Université de Montréal

Enhancing and evolving a rule-based system using historical data: a neuro-fuzzy approach

par

Gang MAI

Département d'Informatique et de Recherche Opérationnelle
Faculté des Arts et des Sciences

Mémoire présenté à la Faculté des Études Supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en Informatique

Mai, 2002

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

Enhancing and evolving a rule-based system using historical data: a neuro-fuzzy approach

présenté par :

Gang MAI

a été évalué par un jury composé des personnes suivantes :

Julie VACHON
Présidente rapporteuse

Houari A. SAHRAOUI
Directeur de recherche

Hakim LOUNIS
Codirecteur de recherche

Philippe LANGLAIS
Membre du jury

Mémoire accepté le : 7 juin 2002

Abstract

Most applications of machine learning (ML) and artificial intelligence (AI) are based on constructing a knowledge model by the rule-based systems which using historical data. The C4.5 algorithm is a machine learning algorithm that implementing the decision tree (DT). This DT is one of the most applied rule-based systems. Fuzzy logic based extensions of the C4.5 algorithm are the popular modifications of these decision learning and making algorithms. The remained problems are that both the predictability and the efficiency are not as good as they should.

To improve the predictability and the efficiency, we propose a new approach that integrates the fuzzy logic and the neural network (NN) in process of generating DT. Our new approach uses the fuzzy logic to improve the prediction rate by focusing on the trends to split the DT rather than the threshold values. Using fuzzy logic aims at combining DT with approximate reasoning offered by fuzzy representation and generates the high knowledge comprehension of DTs. The combination had the ability to deal with inexact and uncertain information of the knowledge base. Moreover, we set up the NN for transferring the continuous variables to the symbolic variables and for the defuzzification procedure.

Our rule-based system is the fuzzy decision system when the classifications are the symbolic variables, and when the classifications are the continuous variables, the system is the fuzzy regression system. In our thesis, in addition to making some modifications to the Marsala's fuzzy decision tree (FDT), we introduce a new method to deal with the continuous regressions: First, we suggest to use self-organizing maps (SOM) transferring continuous classes to symbolic classes; second, we use FDT to classify the symbolic classifications; finally, we propose the NN for defuzzification phrase and get the predicted values. The results are much prompted and some conclusions are given in the end.

To evaluate our approach, we introduce data sets both from the academy and the industry for testing the prediction accuracy and the completeness of learning results. This test helps us to improve the proposed algorithms.

Keywords: rule-based system, fuzzy decision tree, fuzzy regression system, software quality prediction, neural network.

Résumé

C4.5 est un algorithme d'apprentissage permettant de générer un arbre de décision . Les extensions de cet algorithme, basées sur la logique floue de l'algorithme C4.5 sont des modifications populaires des algorithmes d'apprentissage. Il faut cependant souligner que des problèmes demeurent non résolus; ainsi par exemple, la capacité de prédiction et l'efficacité ne sont pas très bonnes.

Pour résoudre ces problèmes, nous proposons une nouvelle approche qui intègre la logique floue et les réseaux de neurones dans le processus de génération des arbres de décisions. Cette nouvelle approche emploie la logique floue pour améliorer le taux de prédiction en se concentrant sur les tendances pour diviser l'arbre de décision plutôt que d'utiliser des valeurs de seuil. Le but de la logique floue vise à combiner les arbres de décision avec le raisonnement approximatif offert par la représentation floue. Cette combinaison produit un haut niveau de compréhension des arbres de décision et offre une capacité de traitement dans le cas des informations inexactes et incertaines de la base de connaissance.

De plus nous employons les réseaux de neurones, d'une part pour transformer la classification continue en une classification symbolique et d'autre part pour la procédure de ''defuzzification''.

Quand les classifications sont des variables symboliques, notre système à base de règles est un système de décisions floues et quand les classifications sont des variables continues notre système est un système de régression flou. Dans notre thèse, en plus de l'introduction de quelques modifications à l'arbre de décision flou de Marsala (FDT), nous présentons une nouvelle méthode pour traiter les classifications continues: d'abord, nous suggérons l'auto organisation des cartes (SOM) de transfert des classifications continues vers les classifications symboliques; deuxièmement, nous employons les arbres de décision flous pour classifier les variables symboliques; finalement, nous proposons

les réseaux de neurones pour la defuzzification afin d'obtenir les valeurs prédites. Les résultats de nos expériences sont présentés et une conclusion est donnée à la fin.

Pour évaluer notre approche, nous présentons des données issues du milieu académique ainsi que des données issues de l'industrie. Nous évaluons l'exactitude de prédiction et la complétude des résultats d'apprentissage. Cet essai nous aide à améliorer l'algorithme proposé.

Mots-clés : système à base de règles, arbre de décision flous, système de régression flous, prédiction de qualité des logiciels, réseaux de neurones.

Contents

Table Contents

Figure Contents

Abbreviations

AI: Artificial intelligence.

CART: Classification and Regression Trees.

COG: Center of gravity.

CRIM: Centre de Recherche Informatique de Montréal.

DT: Decision tree.

FDT: Fuzzy decision tree.

FPMM: Fuzzy Partitioning using mathematical morphology.

HRM: Hydropower resources management group at Alcan Ltd.

ML: Machine learning.

MOM: Mean of maximum.

NN: Neural network.

NSERC: Natural Sciences and Engineering Research Council of Canada.

RT: Regression tree.

SOM: Self-organizing maps.

TDIDT: Top down Induction of Decision Tree.

UQAM: University of Quebec at Montreal.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Houari Sahraoui, for all his support since I started my research on this project. I greatly appreciate the opportunity to work in such an interesting project under his precise comments and valuable guidance. I would like to thank him for all his effort in a very kind and encouraging way by pointing out relevant research, generating interesting ideas.

I would like to thank my co-supervisor, Professor Hakim Lounis of UQAM. He reviews all the topics in this thesis. Without his keen guidance, this thesis is also impossible. I would like to thank Professor Mounir Boukadoum of UQAM, for his academic instruction on machine learning.

Many thanks to Professor Julie Vachon and Professor Philippe Langlais, for their time, patience, and accurate comments.

Thanks to Mr. Robert Cantave of CRIM, for his help on the neural network. Thanks are also due to Salah Bouktif, for giving us the data sets for testing our algorithms, Mohamed Adel Serhani, and Idrissa Konkobo, for many interesting discussions on the topics of my thesis and helping me broaden my perspective.

À mes parents...

Chapter 1 *Introduction*

Machine learning (ML) methods are computer methods for accumulating, changing, and updating knowledge in artificial intelligence (AI). One of the tasks of ML is to set up rule-based systems for class prediction. Normally, the decision tree is for predicting the symbolic classification problems and the regression tree is for predicting the continuous classification problems. The rule-based systems have many practical applications in many areas, such as in this thesis, the prediction of the software quality prediction and the prediction of the power station data. Enhancing and evolving the performance of the rule-based systems is the highly research field and sets of methods are developed. In our thesis, we will mainly discuss one kind of rule-based systems: the fuzzy decision and regression system.

1.1 The big picture

The rule-based systems like decision tree (DT) and regression tree (RT) using the historical data means that they are inductive learning from the examples. These methods assume that a set of examples or instances are known. The instances are the form of (x_i, y_i), where $x_i \in D$ is a state of the domain space D and $y_i \in S$ is the state of the solution space S. The task is to create a system which can learn from the input/output associations $\{(x, y)\}$. Simply speaking, it is given a set of examples (the learning set of associated input/output pairs), deriving a general rule representing the underlying input/output relationship, which may be used to explain the observed pairs and/or predict output values for any new unseen inputs. This belongs to supervised learning.

In contrast to the supervised learning, where the object is clearly defined in terms of modeling input/output relationships, unsupervised learning methods are not oriented towards a particular prediction task. Unsupervised learning eliminates the teacher and requires that the learner form and evaluate concepts on its own. They try to find by themselves the existing relationships among states characterized by a set of attributes. For example, clustering begins with a collection of unclassified objects and some means of measuring the similarity of objects, and then organizes the objects into a hierarchy of classes that meet some standard of quality, such as maximizing the similarity of objects in the same class [LS98].

How to discriminate the classes and what structure should be used for generalizing new examples? A supervised learning technique called the inductive DT "observes" the examples and builds the trees which can discriminate all the classes. The automatic extraction of the knowledge from a set of data enables us to discover relationships among these data. Furthermore, these relationships can be used for generalizing the new data sets. Given a set of data described by means of the values of attributes and associated with the value of the classes, DTs are able to construct a set of rules that express the relationships. The fuzzy decision system is the fuzzy set theory extension of the classical

C4.5 algorithm [Qui93], one of the popular DT models. This system enables us to take into account the imprecision values in the description of data sets.

Fuzzy set theory represents in a straight forward way of the "common sense" knowledge and skills, or the knowledge that is subjective, ambiguous, or vague. This theory uses analog values to realize fuzzy inference which is robust and efficient. They are the macroscopic ways to realize human intelligence at the level of symbols and rules. Fuzzy decision systems provide the convenient and flexible methods of reasoning and decision.

In our approach of fuzzy decision and the regression system, the attributes in the data sets are all the continuous variables. For the special attribute, classes, we have two formats. The first format is that the classes are symbolic variables, for example, the software quality prediction data sets. The second data format is that the classes are continuous variables, in our thesis for example, the hydropower station data sets.

When the classes are symbolic variables, we use our fuzzy decision tree (FDT) to do the inductive learning which is the modification of Marsala's FDT. When the classes are continuous variables, normally people use the regression tree (RT) to do the inductive learning. Our approach is integrating the neuro-fuzzy method with the FDT to do the inductive learning. This classification and regression system is the hybrid neuro-fuzzy approach of the rule-based system. The results show that for the classification this approach is better than the classical methods and for the regression it is the satisfying approach between the accuracy and explicit trade-off.

1.2 Motivations

The first initiation of this approach is from the task for predicting the continuous classes power station data provided by Alcan Ltd. Alcan company, the biggest Canadian player in the aluminum industry, has a hydropower network constitutes a territory larger than the province of New Brunswick (Canada). The network has, on average, an annual energy

capacity of approximately 2000 megawatts; it includes 6 hydroelectric power station, 28 reserve installations, 43 turbine-alternators groups, 850 kilometers of energy transport lines, a network of about thirty hydro-meteorological stations, etc. Alcan Ltd. needs planning a knowledge-based system for hydroelectric power generation management. These big industrial project between Alcan Ltd. and CRIM (Centre de Recherche Informatique de Montréal), is supported by a joint grant from Alcan Ltd. and NSERC (Natural Sciences and Engineering Research Council of Canada), the operation grant number is #CRDPJ 228746-99 [BML02]. They have predicted the same data sets using the domain knowledge and want to compare them with the rules which we get from our rule-based system. We are fortunately the members of this research group.

As usual, the rule-based system can be divided into two sub-systems: One is the classification system which deals with the symbolic classes; the other is the regression system which deals with the continuous classes. The classification system performs well in finding decision rules. But it fails in predicting continuous variables and also the prediction ability for the symbolic variables need to be improved. The regression system can predict efficiently for the continuous variables. But it is hard to extract rules from the RTs, as it is very complex and not explicit.

To solve the problem of predicting the continuous regressions, while we also want to get the explicit rules to store in the rule base. We propose a fuzzy regression system, a combination the FDT and the RT techniques by using the neuro-fuzzy approach. Our new approach is firstly modifying the fuzzy classification system and making it more efficient and dynamic. After that, we add the neural network (NN) to this modified system and make it deal with the continuous regressions. Our motivation for proposing this fuzzy regression system is that the poor results obtained with C4.5 algorithm may be improved by using a fuzzy binary tree instead of the regular tree used in C4.5, and by using a neural network in the defuzzification stage, thus switching from the original binary classification system into a neuro-fuzzy regression system. It is also known that the use of fuzzy sets enhances the understandability and predictive ability of regular decision trees [Mar98],

[Tor99]. The advantages of our new approach are in the regression system we can generate the decision rules, and the discrimination ability is also robust.

Usually, the neuro-fuzzy approach in the fuzzy rule systems refers to the combinations of the NNs and fuzzy systems [NK99], but does not mean they are used together in the same way. Our neuro-fuzzy approach means that introducing the self-organizing maps (SOM) to transfer the continuous classes into the symbolic classes, and using the back propagation to defuzzify the FDTs.

1.3 Goals

Our goals are using the neuro-fuzzy approach to enhance and evolve a rule-based system, the FDT system. The main goals we get from this thesis are:

- Modifying the Marsala's FDTs which is a typical rule-based system, with the changes of the fuzzy partition shape, in order to be more suitable for the mathematical morphology algorithm, the splitting criterion, and adding the enhanced pessimistic pruning process to the building of the FDTs.
- When generating the new cases, we use the min-max algorithm and the vote method to defuzzify the FDTs.
- Given an automatic formula for fuzzy regression the continuous variables. The built models are robust and easy to extract the decision rules.
- Changing the continuous classes to the symbolic classes using the unsupervised learning method: the SOM to generate the classes automatically.
- Embedding the back propagation NN to defuzzify the FDTs, we get the perfect prediction rate for both the symbolic class applications (software quality prediction) and the continuous class applications (hydropower station applications).
- For both the symbolic and the continuous regressions, we provide different but suitable evaluation models, and also the necessary discussions are given.

1.4 The outlines

The rest of the thesis is organized as follows:

Chapter two presents the state of the art, some concepts which concern in this thesis, and the related works in this domain.

Chapter three introduces the general process of building the Marsala's approach of FDT for the symbolic class data.

Chapter four modifies the Marsala's FDT model and builds the new FDT models.

Chapter five proposes the neuro-fuzzy approach of the fuzzy regression procedure, including the SOM method and the back propagation NN.

Chapter six provides the java program which implements our FDT models, especially we introduces how the *C* program embeds in our java program.

Chapter seven gives the application results of the proposed algorithms both for the software quality prediction data sets (symbolic classifications) and the hydropower station data sets (continuous regressions). Some evaluation models are proposed.

Chapter eight draws the conclusion which comes from the discussion of the whole thesis and the future work.

Chapter 2 *State of the Art*

In chapter one, we introduced the motivation of the topics and presented the arrangement of this thesis. In this chapter, we introduce some concepts of the rule-based system and present the current issues of the FDT, one of the rule-based systems that we would like to enhance and evolve.

2.1 Terminology of context

The rule-based system is the most practical and remaining widely used system in AI. The others are the model-based reasoning system and the case-based reasoning system. The model-based reasoning system is based on the knowledge-based system whose analysis is founded directly on the specification and functionality of a physical system. Because they are based on the theoretical understanding of the domain, model-based techniques can break through the limitations of the heuristic approaches. On the other hand, this explicit

model in the knowledge acquisition stage is quite demanded and program is made large [LS98]. The case-based reasoning system uses an explicit database of problem solutions to address the new problem-solving situations. The most difficult issue in it is the selection of salient features for the indexing and retrieval of cases. Therefore, the case-based reasoning system is rarely used comparing to the rule-based system.

In this section, we introduce some concepts of rule-based system and AI. These concepts are used in this thesis thoroughly. Before we discuss the FDT construction, the introduction to some basic definitions and terminology will facilitate the reader's understanding.

Data mining

Data mining is a highly multidisciplinary research field and a set of methods (involving theoretical and applied methods from statistics, computer science, AI) to extract high level synthetic information (knowledge) from databases containing large amounts of low level data.

Machine learning (ML)

In knowledge discovery, ML is most frequently used to meaning the application of induction algorithms, which is one step in the knowledge discovery process. This is similar to the definition of empirical learning or inductive learning. The training examples are "externally supplied", whereas they are assumed to be supplied by a previous stage of the knowledge discovery process. ML is a field of scientific study that concentrates on induction algorithms and other algorithms that can be said to "learn".

Rule-based system

Instead of representing knowledge in a relatively declarative, static way (as a bunch of things that are true), rule-based system represents knowledge in terms of a bunch of rules that tell you what you should do or what you can conclude in different situations. A rule-

based system consists of a bunch of *if-then* rules, with the premises of the rules. The *if* portion, corresponding to the condition, the *then* portion, corresponding to the conclusion: When the condition is satisfying, the system takes the action of asserting the conclusion as true. In our thesis, the rule-based system refers to the fuzzy decision rule (tree) system.

Inductive learning

Inductive learning raises the particular to the general. A set of classes C are considered, representing a physical or a conceptual phenomenon. This phenomenon is described by means of a set of attributes $\mathrm{A} = \{A_1,...,A_N\}$. Each attribute A_j can take a value v_{jl} in a given universe X_j. A description is an N-tuple of attribute-value pairs (A_j, v_{jl}). Each description is associated with a particular class c_k from the set $C = \{c_1,...,c_K\}$ to make up an instance (or example, or case) e_i of the phenomenon. So it is a process to generalize from a training set $\varepsilon = \{e_1,...,e_n\}$ of examples to a general law and bring out relations between descriptions and classes in C.

Top down induction of decision tree (TDIDT)

TDITD method proceeds by successively splitting the training set into subsets. Each partition is done by means of a test on the selected attribute and leads to the definition of a node of the tree. Thanks to the measure of discrimination, a suitable attribute is selected. Such a measure enables us to order the attributes according to an increasing accuracy to split the training set. The partitioning is done by means of a splitting strategy. A stopping criterion enables us to stop splitting the data and to construct a leaf in the tree.

Decision tree (DT)

A DT is a set of nodes, where each node tests the value of an attribute, and each edge from a node is labeled with a particular value of the attribute and a set of leaves, where each leaf gives a class value. It is built from the training set, which consists of objects.

Each object is completely described by a set of attributes and a class label. Attributes can have ordered or unordered values (in this thesis, we only consider the unordered values). A DT contains a root node, zero or more internal nodes, and one or more leaf nodes. All internal nodes have one or more child nodes. All non-terminal nodes contain splits. Each leaf node has a class label associated with it. The number of classes is finite.

An object is misclassified by a tree if the class label output by the tree does not match the class label of the object. The proportion of the objects classified correctly by the tree is called accuracy, and otherwise, it is called error. Note that the tree can be translated into an equivalent set of *if-then* rules. Each rule responds for each rout from the top node to the terminal node. A DT is important not only because it summarizes what we know, i.e. the training set, but also it may classify new cases correctly and efficiently.

C4.5 algorithm

The C4.5 algorithm is the coming from ID3 algorithm; both of them are introduced by Quinlan for inducing classification models in the form of decision trees. The basic idea of Id3 algorithm is:

- In the decision tree, each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf.
- In the decision tree at each node should be associated the non-categorical attribute which is most informative among the attributes not yet considered in the path from the root.
- Entropy is used to measure how informative is a node.

C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute ranges, and pruning of decision trees.

Fuzzy logic theory

Fuzzy logic was invented by Zadeh. The main contention is although probability theory is appropriate for measuring randomness of information, it is inappropriate for measuring the meaning of information. The backbone of fuzzy logic theory is fuzzy set theory which is the basis of fuzzy model and fuzzy inference. In the fuzzy set theory, objects have a grade of membership ranging from *0* to *1*. Fuzzy inference provides the meanings to perform the approximate reasoning. The key to a fuzzy inference system is the fuzzy rule base, which contains information relating the input conditions to the output conditions. A fuzzy model is created when the fuzzy rule base defines the relationship between the input domains and the output domains of some target systems. Most approaches to fuzzy inference can be categorized as generalizations of either logical deduction or approximation theory.

Fuzzy set theory expresses lack of precision in a quantitative fashion by introducing a set of membership functions that can be the real values between *0* and *1*. This notion of a fuzzy set can be described as follows: Let *S* be a fuzzy set and *s* a member of it. A fuzzy subset *F* of *S* is defined by a membership function *mF(s)* that measures the "degree" to which *s* belongs to *F*.

Fuzzy decision tree (FDT)

Classical DTs handle symbolic attributes, but the presence of continuous or continuous/symbol attributes leads to different kinds of DTs. Continuous/symbol modalities are on the universe of values of the continuous attributes. Thus, fuzzy set theory methods have been introduced to handle continuous attributes when constructing DTs. Such methods define the FDT. Most of the FDTs implement through the TDIDT method.

Regression tree (RT)

Regression problem consists on obtaining a functional model that relates the value of a target continuous variable y with the values of variables $x_1, x_2, ..., x_n$ (the predictor). This

model is obtained using samples of the unknown regression function. The RT is similar to the DT in the form. But they are used in different situations and the methods for discrimination are also different. As usual, DTs are used for the symbolic classifications and RTs are used for the continuous regressions. The regression model can be represented in the form of the tree. This tree consists of a hierarchy of nodes, starting with a top node known as the root node. Each node of the tree contains logical tests on the predictor variables, with the exception of the bottom nodes of the hierarchy. The bottom nodes are usually known as the leaves of the tree.

2.2 The limitation of DTs

The rule-based systems are the oldest approach to knowledge representation in AI, and remain as an important technique for building the knowledge-intensive problem solvers. Using the rule-based systems, we can get the knowledge base of certain domain as *if-then* rules, which can be easily understood. These descriptions serve to explain what have been learned and what the basis is for new prediction. Moreover, it will be very helpful for us to know the internal relationship of the domain and the rules can be used in many other systems.

Normally the DT is the first method to be thought about when applying the rule-based systems. The study of the DT is the dynamic part of ML. Most of the DT methods, including the C4.5 algorithm, construct DTs using the TDIDT method which generally represents the inductive knowledge by selecting the best attribute to obtain the compact trees with high predictive accuracy. Using the C4.5 DT, we can get good results from which the attributes are symbolic. But for the continuous attributes, there are two limitations:

- First, when it treats the continuous attributes, the C4.5 processes them as the symbolic attributes: Each continuous value is considered as a symbolic value and it selects a threshold value as the splitting criterion. With such a method, the generated DTs are very large and in details (over-fitted). Moreover, when it

classifies the new cases, continuous values may be different from the values within the tree.

- Second, when the data values are successively distributed and there are noises in the training/testing samples, it is easy to classify the instances into the wrong parts.

In these two situations, errors occur as the testing samples take the wrong branch while they are classified by the DT. So you can not say that the threshold value as the splitting criterion is the best choice.

In order to minimize these errors, people introduce the FDT which making the decision at each node with the fuzzy membership functions. First, we would like to introduce that there is an alternative method to FDT. That is the probabilistic theory approaches. These probabilistic theory approaches are based on the naïve Bayesian model [SBLF00] or the Dempster-Shafer theory [WebSW]. They are testified effective and validate. The remaining problem is that the probabilistic theory is more complex than the fuzzy logic theory and according to Sahraoui's comparison, the results are not better than those of FDT [SBLF00].

The general algorithm to construct a FDT is similar to the building of the C4.5 DT which approaches to the problem by learning from a set of independent instances. The FDT can bridge the gaps of the symbolic and the continuous attributes by fuzzifying the latter. Instead of using the threshold to split the continuous attributes, FDT fuzzifies the continuous attributes, and splits the data according the fuzzy membership function. The branch of the FDT is the fuzzy labels of the continuous attributes. FDT is tolerance to the training/testing data which is incomplete and imprecise and provides good tools to discover knowledge from these data. The FDT will be introduced in the following sections.

2.3 Overview of the current technologies of FDT

As you know, FDT has many advantages over the DT. In this thesis, we focus on the enhancing and evolving the FDT algorithm. Now, let's look at the current stage of FDTs.

All the FDTs follow the TDIDT structure. The FDT methods differ from each other by their choices of a measure of discrimination entropy, a splitting strategy for attributes, and a stopping criterion. Basically speaking, the FDT can be grouped by the answers to the following questions: First, what kind of fuzzy measure it uses; second, how it gets the fuzzy modalities.

2.3.1 FDT based on star entropy measure

There are two families of entropy measures to construct FDT. One is based on the generalized Shannon entropy measurement. It is called star entropy. Another is based on the other entropy measurements.

The most common used discrimination measure of the FDT is the entropy of fuzzy events, the star entropy. It is the extension of the Shannon entropy to the fuzzy events by fuzzy probability:

$$H_S^*(C) = -\sum_{k=1}^{K} p^*(c_k)\log(p^*(c_k)).$$

The chosen definition of probability p^* is introduced by Zadeh in 1968 [Zad68]. The explanations are following.

$$p^*(V) = \sum_{i=1}^{n} \mu(e_i)P(e_i).$$

Given a fuzzy subset V= { $e_1,...,e_n$ } with a membership function μ, each element e_i of V is associated with a classical probability $P(e_i)$. This star entropy $H_S^*(C)$ was introduced by Tanaka et al. in 1979 [TOA79] as a measure of fuzzy information. It has been used in several software systems to construct the FDTs.

The SAFI system [Ram92] is one of the first methods of constructing the FDT in the inductive learning in presence of continuous/symbolic data sets. In this system, the star entropy is used as a measure of discrimination. In SAFI system, the fuzzy modalities of the continuous attributes (including class, the special attribute) are provided by the domain experts. Ramdani introduces a method based on the use of the star entropy to optimize the fuzzy modalities. The problem is, according to his research, the best fuzzy partitions are often the classical partitions. Wehenkel [Weh96] studies this problem too. He shows that the major drawback of the star entropy is that the optimum is its non-fuzzy subsets.

Weber [Web92] and Janikow [Jan94] also use the star entropy to set up their own FDT systems individually. All these systems are using the presence of given fuzzy modalities on the universe of values of continuous attributes. Moreover, in Janikow's system, he optimized the fuzzy partition by means of genetic algorithm.

The star entropy is also implemented in the Marsala's FDT [Mar98]. All these methods are equal in the sense of that using the star entropy to measure the discrimination of the fuzzy sets, although they use modifications on the fuzzy partitions and different methods when classifying the new cases.

2.3.2 FDT based on other fuzzy measures

The star entropy is one of the heuristic methods in constructing the FDT. There are many other types of entropy, besides the star entropy. The star entropy's efficiency in constructing FDT as an information measure has been proved, while others have not been

proved yet. As this thesis is based on the enhancing the ability of the FDT, we would like to describe what these fuzzy classification schemes are.

The first approach is De Luca and Termini's entropy [DT72]. Given a fuzzy subset V= $\{e_1,...,e_n\}$ with a membership function μ. The De Luca and Termini entropy of the fuzzy subset V is defined as:

$$H_D(V) = -\sum_{i=1}^{n}(\mu(e_i)\log(\mu(e_i)) + (1-\mu(e_i))\log(1-\mu(e_i))).$$

This measure can evaluate the fuzziness of the subset V. Another approach is made by Kosko [Kos97]. He defines the fuzzy entropy as:

$$H_K(V) = \frac{\sum_{count}(V \cap \bar{V})}{\sum_{count}(V \cup \bar{V})},$$

Where $\sum_{count}$ is the cardinality of fuzzy subsets introduced by Zadeh, and $\bar{V}$ is the complement of V.

There are another two applications using the fuzzy Kolmogorov-Smirnov measure as the fuzzy entropy. In Araya's approach [Ara94], when there are only two classes to recognize, this measure values the largest distance between the distributions of probability related to each class:

$$K(c_1,c_2) = \max_x(|P_{c1}(x) - P_{c2}(x)|),$$

Where P_{c1} (or P_{c2}) is the distribution of probability of class c_1 (or class c_2).

Boyen proposes two methods to deal with the continuous/symbolic attributes [Boy95], which are Outbound FDTs and Fussy FDTs. These two methods are based on the Kolmogorov-Smirnov measure.

Finally, a very different approach is presented by Yuan et al. [YS95]. In their method, they introduce the cognitive uncertainty, such as vagueness and ambiguity associated with human thinking and perception. Once the fuzzy sets are introduced, the cognitive uncertainties represented by fuzzy sets can be measured. They are vagueness measures E_v and the ambiguity E_a.

Vagueness measurement: Let A denote a fuzzy set on the universe U with membership function $\mu_A(u)$ for all $u \in U$. If U is a discrete set $U = \{u_1, u_2, ..., u_m\}$ and $\mu_i = \mu_A(u_i)$, the vagueness or the fuzziness of fuzzy set A is defined by

$$E_v(A) = -\frac{1}{m}\sum_{i=1}^{m}(\mu_i \ln \mu_i + (1-\mu_i)\ln(1-\mu_i)).$$

$E_v(A)$ measures the fuzziness or vagueness of a fuzzy set A.

Ambiguity or non-specificity measure: Let $\pi = (\pi(x) \mid x \in X)$ denote a normalized possibility distribution of Y on $X = \{x_1, x_2, ..., x_n\}$, the possibility measure of ambiguity or non-specificity is defined as:

$E_a(Y) = g(\pi) = \sum_{i=1}^{n}(\pi_i^* - \pi_{i+1}^*)\ln i$, where $\pi^* = \{\pi_1^*, \pi_2^*, ..., \pi_n^*\}$ is the permutation of the possibility distribution $\pi = \{\pi(x_1), \pi(x_2), ..., \pi(x_n)\}$, sorted so that $\pi_i^* \geq \pi_{i+1}^*$ for $i=1... n$, and $\pi_{n+1}^* = 0$.

The construction of the FDT is the process of reducing classification ambiguity with accumulated fuzzy evidences. Fuzzy evidence is the knowledge of the particular attribute.

The selection of additional fuzzy evidence is based on its contribution in reducing the classification ambiguity. Their approach handles the classification problems with both fuzzy attributes and fuzzy classes represented in linguistic fuzzy terms. It can also handle other situations in a uniform way, where continuous values can be fuzzified to fuzzy terms and crisp categories can be treated as a special case of fuzzy terms with zero fuzziness. The classification ambiguity directly measures the quality of classes without any restrictions. Another advantage of their approach is the use of significant level of evidence and truth level threshold which provide effective control during the induction process.

2.3.3 Determination of fuzzy modalities

The other differences of building the FDT is the procedure of obtaining the fuzzy modalities. There are many methods to obtain the fuzzy modalities. Apparently, some linguistic characterizations of the attributes can be given by an expert (or many experts) of domain. However, in some particular cases, there is no available knowledge from any domain expert, and even the expert in the domain might make mistakes and his knowledge might be imperfect and should not take into account exactly the observed phenomenon. Most of the FDTs are using the expert's knowledge as the bases of the fuzzy modalities, such as the FDT introduced by Koen-Myung et al. [KKJL99], and Yonghong et al. [YF01], and Haskell [Has99].

When there are no fuzzy modalities available by the expert on the studied domain, it seems to infer an automatic way to get the fuzzy modalities. There exist a lot of methods to construct the fuzzy partitions from the continuous attributes. Most of them come from statistical approaches, such as the naïve Bayesian method. Some of the automatic methods are developed by the NN theory or genetic algorithm. In this cases, the results are still fuzzy.

Narazaki et al [NR94] propose a method to construct the membership function in a learning scheme. Given a set of continuous values and a class, the mean value is computed. This value enables to compute the distance for any elements to that class. Thus, the distance can be translated as a membership value for the element to the class.

Sahraoui et al [SSB00] propose the fuzzy modalities when they build the fuzzy decision system (001 system) in the learning scheme. They use the C-means method plus the statistical ways to get the fuzzy modalities.

Some other methods have been developed in the context of FDTs induction. For instance, Yuan et al. [YS95] determine a set of triangular membership functions using the fuzzy clustering based on the SOM algorithm. Assume attribute A has continuous value x, the continuous values of attribute A for all objects $u \in U$ then can be represented by $X = \{x(u), u \in U\}$. They want to cluster X to k linguistic terms $T_i, i = 1,...,k$. Each linguistic term T_i has a triangular membership function as follows:

$$u_{T_1}(x) = \begin{cases} 1, & x \le m_1, \\ (m_2 - x)/(m_2 - m_1), & m_1 < x < m_2, \\ 0, & x \ge m_2, \end{cases}$$

$$u_{T_k}(x) = \begin{cases} 1, & x \ge m_k, \\ (x - m_{k-1})/(m_k - m_{k-1}), & m_{k-1} < x < m_k, \\ 0, & x \le m_{k-1}, \end{cases}$$

$$u_{T_i}(x) = \begin{cases} 0, & x \ge m_{i+1}, \\ (m_{i+1} - x)/(m_{i+1} - m_i), & m_i < x < m_{i+1}, \quad 1 < i < k \\ 0, & x \le m_{i-1}, \end{cases}$$

The slopes of the triangular membership functions are selected in the way that adjacent membership functions cross at the membership value 0.5. In this case, the parameters

needed to be determined are the set of k centers $M = \{m_i, i = 1,...,k\}$. The centers m_i can be calculated by using Kohonen's SOM algorithm. The major drawback of this method lies in the fact that it is not linked to the distribution of the classes.

An automatic method to construct the fuzzy partitions on the universe of values of the continuous attributes is proposed by Marsala [Mar00]. In his method, Marsala introduces a new way which bases on the use of operators from mathematical morphology to make the fuzzy modalities, and these operators allow the filtering of the set of continuous values to highlight kernels for the fuzzy modalities related to the classes. In chapter three, we will focus on this topic.

2.3.4 The applications of FDT

As FDT is an efficient and robust rule-based system, it has many applications in different domains in the range of ML.

In 1994, Umano et al. [UOH94] used FDT in the diagnosis of potential transformers, which can analyze gas in oil and determine the cause of destruction of potential transformers.

The FDT has been used in a robotic task to cut three-dimensional work pieces by a manipulator with six degrees of freedom, which is used to extract important factors for the task [SAT95].

In the approach of Boyen et al. [BW96], FDT has been used in the context of power system security assessment. In this domain, FDT is used to foresee the transient stability of a power system.

Bothorel et al. reported his applications of FDTs to the medical domain in 1997 [BBM97]. Here, in breast cancer recognition context, they are used to determine the

malignancy of a micro-calcification described by fuzzy features in a mammography image.

In chemistry, Marsala et al. [MRT98] had set up the relations of the structures of chemical compounds and the quality of the odors using the FDT in 1998.

Sahraoui et al [SBL00] proposed the prediction model for evaluating the stability of a reusable class library interface using the FDT. The results they got are better than the C4.5 and the other statistic methods. They implemented FDT to build the software quality estimation models [SBL01] in addition.

Other applications are in the fuzzy decision inductive learning processes. For instance, Botta et al. [BGS93] used such an approach to deduce imprecise categories from imprecise descriptions. Maher et al. [MS93] provided an inductive learning algorithm which is robust with regard to knowledge uncertainties. Also, in 1997, Ohno-Machado et al [OLM97] applied the fuzzy decision model to select the measles vaccination strategies in Brazil.

2.4 Regression trees and the remaining problems

Normally, the way to deal with the continuous regressions data sets is using the CART (Classification and Regression Trees) algorithm to build the RT.

The principle of the CART algorithm is growing the maximal tree and then using statistical techniques to prune it and obtain an optimal tree [WebMcC]. There are many modifications of the CART algorithm: One of them is local RTs [Tor99] presented by Torgo. He introduces a hybrid model that integrates tree-based regression with local modeling techniques. His model is more efficient and accurate than the regular regression methods.

The CART algorithm and its modifications are typical techniques for classifying the continuous regressions. But they have some defects too: One is that the trees they generated are very large and in details and have to put more accurate prune process on them, so the rules they generated are too complicated to use for practice; another is that they use the crisp value as the criterion to split the RT. This is the same defect of the DTs.

From what we have presented until now, the problems still remain because the FDTs are easily to explain and retrieve the rules, but hard to deal with the continuous regressions. The RTs treat the continuous regressions easily but hard to give us the sets of rules. In chapter four and five, we will use the neuro-fuzzy approach to integrate the FDT and the continuous regressions.

2.5 Summary

The similarity occurs in all the FDT systems. First of all, all the FDTs are constructed by extending the classical information measurement, the entropy, so that we are able to deal with the continuous/symbolic attributes. During all these extensions, the star entropy is widely used. Second, few of them construct fuzzy modalities on the universe of values of the continuous attributes. Most of them considered the fuzzy modalities are granted given. Therefore, one of the advantages of Marsala's algorithm to build the FDT is using the automatic learning process for the arbitrary fuzzy modalities. In chapter three, we will present Marsala's method on how to build the fuzzy modalities and the FDT, on which our approach is based. In chapter four and five, we will present how we use the neuro-fuzzy approach to modify the Marsala's algorithms.

Chapter 3 *Fuzzy Decision Tree*

In chapter two, we presented the state of the art of the fuzzy rule-based systems. In this chapter, we focus on Marsala's approach of FDT. In the first part, we give a general idea of the Marsala's method for building the FDT. Then, we present how to build the FDT in details accordingly. Finally, we give the conclusion of Marsala's approach.

3.1 Introduction

There are many new methods proposed by Marsala in building the FDT. For example, first, the construction of FDTs is based on the knowledge of a fuzzy partition for each continuous attribute. Marsala introduces an automatic method in generalizing the fuzzy partition on the continuous/symbolic attributes using the mathematical morphology operators. Second, he uses the star entropy, the measure of discrimination during the FDTs' construction and gives a hierarchical model of functions. Third, Marsala classifies

the previously unknown cases using a measure of the satisfying degree to evaluate the satisfaction between the description values of the case and the corresponding values that appear in nodes of the tree.

These methods have been implemented as a computer system named *Salammbo*. This system constructs the binary FDTs and recognizes only two class values: this value versus not this value.

To deal with the classifications with more than two class values, such as the classes are three or four, Marsala implements another computer system named *Tanit*. This system enables us to construct a fuzzy decision forest by means of integrating several *Salammbo* systems.

During the classification of new cases, the membership degrees of the class values given by each *Salammbo* have been aggregated by the *Tanit* system. Both *Salammbo* and *Tanit* systems have been tested on various kinds of training sets. These tests enable us to highlight the greater understandability and explain ability of decisions given by the FDT system. Moreover, it can be shown that the size of fuzzy trees is smaller than that of classical trees, and their classification rate is also improved.

3.2 The algorithms for constructing FDT

The general algorithm to construct a FDT is similar to building the C4.5 DT, the approach of learning from a set of independent instances. Marsala's building the FDT derives from three parts: automatically fuzzy partitioning the continuous attributes; constructing the FDT including discrimination measure, splitting strategy, stopping criterion; and classifying the new instances.

3.2.1 Fuzzy partitioning using mathematical morphology operators

Using FDT to do the inductive inference, we should fuzzify the data set into the fuzzy modality first. Marsala uses the mathematical morphology [Mar95] to generalize the fuzzy partition. The advantage of using the mathematical morphology method is that it integrates the attributes with the classification information in the fuzzification.

- ***Mathematical morphology operators***

The fundamental operators from mathematical morphology theory are the *erosion* operator and the *dilatation* operator. They are combined to produce the *opening* operator and the *closure* operator that can be used to filter a set of bodies. These operators come from the pattern recognition domain and are often used to filter 2D-pictures.

The basic operators: Erosion and dilatation enable us to modify a morphological body (Figure 3.1). This modification is related to a structuring element.

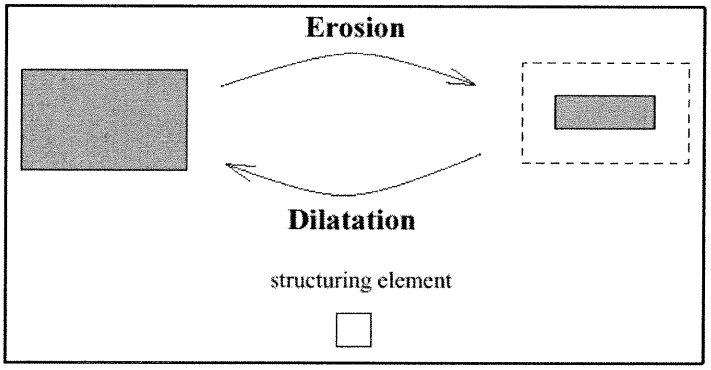

Figure 3.1 Mathematical morphology operators

The combination operators: Opening and closure are a combination of the two basic operators. The opening is the combination of erosion followed by dilatation applied to a morphological body, with the same structuring element (Figure 3.2). It enables to destruct the small bodies in the space, with respect to the size of the chosen structuring element.

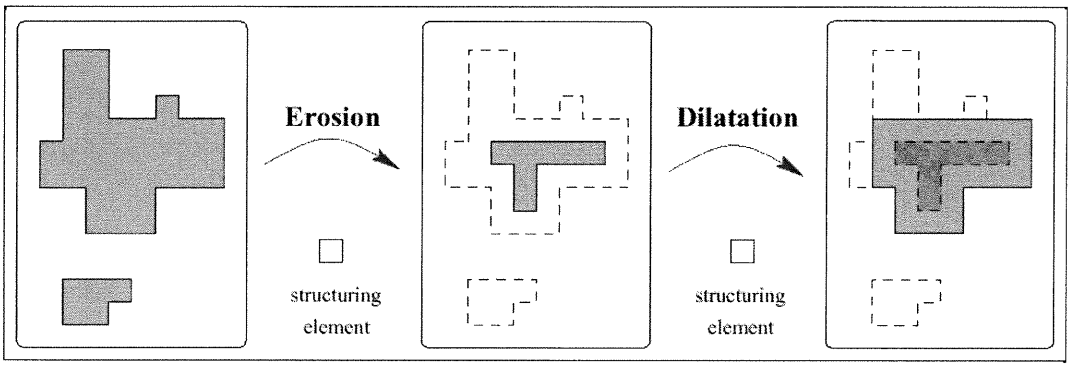

Figure 3.2 Opening operator

The closure is the combination of a dilatation followed by erosion applied to a morphological body, with the same structuring element (Figure 3.3). It enables the destruction of small vacuum places occurring in a body, with respect to the size of the chosen structuring element.

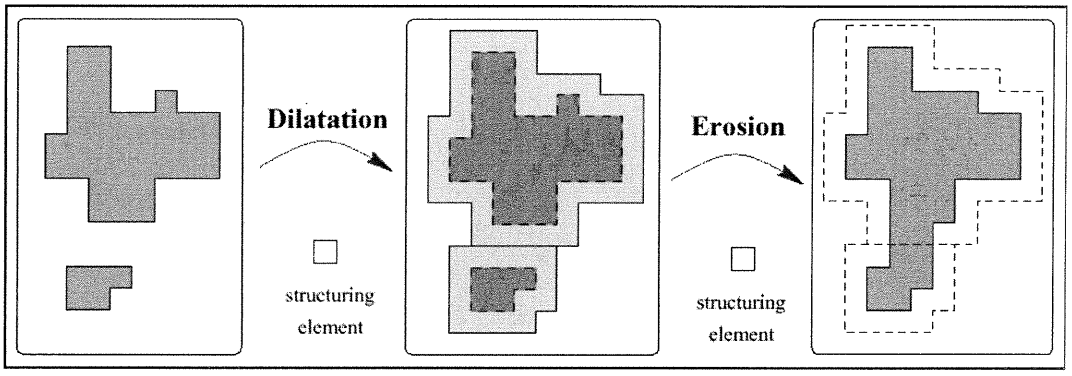

Figure 3.3 Closure operator

The *open-close filter* is a combination of openings and closures. It is composed by k successive openings followed by k successive closures (k=1, 2, ...) applied to all bodies of the space, with the same structuring element. Thus, it enables the destruction of small

bodies presenting in the space with respect to the chosen structuring element. Simultaneously, it enables the filling of small vacuum places occurring in bodies. The value of k enables us to control the power of the modification.

- ***Smoothing a set of values***

Up to now, we know the basic principle of mathematical morphology. In this section, we will see how this principle applies to the fuzzy partitions.

We present the representation of the training set T as a word describing the distribution of classes in T, according to the values of a given attribute which is supposed to take values in an ordered set X.

First, we transform a training set into a word, for a given attribute with values in X. Let L be an alphabet, each letter of L representing one of the classes in T. We construct the alphabet $L_u = L \cup \{u\}$ with $u \notin L$. The letter u is a particular letter in the system. We will use it to determine uncertain sequences. For any alphabet A, A^* denotes the set of all possible words composed by letters from A.

For example, let the training set be T= {(17, cheap), (24, expensive), (29, expensive), (33, expensive), (42, expensive)} with {cheap, expensive} as a set of classes and {17, 24, 29, 33, 42} as a set of values of an attribute (e.g. the attribute "size"). Let L ={c, e}, c representing *cheap* and e representing *expensive*. The word defined in L^* by the training set is *ceeee*.

We want to obtain sequences of letters from L, as homogenous as possible, in order to associate them with fuzzy subsets of X (constituting a fuzzy partitioning of X). Each subset will represent a linguistic modality of the attribute (for instance, *cheap* and *expensive* with the previous example).

We use the operators inspired by mathematical morphology to erase non-representative values within a word in order to smooth it. Let's recall that a transduction is a 6-tuple <*A, B, S, I, E , δ*> where *A* is the input alphabet, *B* is the output alphabet, *S* is a set of states, $I \subseteq S$ is the set of initial state of the transduction, $E \subseteq S$ is the set of terminal states of the transduction and $\delta \subset S \times A^* \times B^* \times S$ is the transition function.

A transduction reads a word $v \in A^*$ and rewrites it in a corresponding word $w \in B^*$. It proceeds sequentially from the first letter to the last one. The rewriting rules to generate *w* are based on *δ*. Let $(s_i, z, t, s_j) \in \delta$, with $s_i, s_j \in S, z \subset A^*$ and $t \subset B^*$, (s_i, z, t, s_j) is called a transition of the transduction. If s_i is the current state and we can read *z* in *v*, we replaced it by *t* and the current state becomes s_j. A convention is to use *$* to match the end of the input word, and *ε* (the null word) is introduced when nothing has to be written.

Let's define the transduction Er_x with $A = B = L_u$. This rewriting system is used for the erosion of a word with a particular letter $x \in L$ as structuring element. It corresponds to the reduction of sequences of *x* in the word.

Di_x is another rewriting system with $A = B = L_u$ and $x \in L$. This system will dilate a sequence in a word when this sequence is surrounded with letter *u*. It can be proven that for any given word *v*, the computed terminal word is unique. Thus, we are sure that $Er_x(v)$ and $Di_x(v)$ exist for all the words $v \in L_u^*$ and for all the letters $x \in L$, and therefore, for all training sets. Moreover, for any word $v \in L_u^*$, we call $Er_x^n(v)$ (resp. $Di_x^n(v)$), with *n*>0, the word obtained from *v* after *n* consecutive erosions (resp. dilatations). With these two rewriting systems, we will define the two usual operations from the mathematical morphology: the opening and the closure.

The opening operation is the composition $Di_x \circ Er_x$ of the two previous operators. The *n*-opening $(n \in N)$ of a word $v \in L_u^*$ with respect to $x \in L$ is defined

as $Op_x^n(v) = Di_x^n(Er_x^n(v))$. The n-opening of a word allows erasing small sequences with length smaller than $2n$. The advantage of this operation is that we can erase all sequences in v with length smaller than a fixed value.

The closure operation is the composition $Er_x \circ Di_x$ of the two operators of erosion and dilatation. This composition allows joining sequences of letters in a word, if these sequences are separated by less than two letters u. the n-closure ($n \in N$) with respect to $x \in L$ of the word $v \in L_u^*$ is defined as $Cl_x^n(v) = Er_x^n(Di_x^n(v))$. With this operation, two sequences separated by less than $2n$ letters u are unified.

Finally, the filter operator is the composition of the previously described word-transforming operation. Let $v \in L_u^*$, $x \in L$ and $n \in N$. The n-filter of the word v with respect to x is defined as:

If n=1: $Fil_x^1(v) = Cl_x^1(Op_x^1(v))$

If n>1: $Fil_x^n(v) = Cl_x^n(Op_x^n(Fil_x^{n-1}(v)))$

The particular combination of these operations has some interesting properties. A filter will allow smoothing a word. First, the sequences with a length of $2n$ letters are erased, and then we unify sequences separated by $2n$ letters.

- ***Fuzzy partition of a set of values***

We apply a filter to the word induced by a training set. We are able to translate small sequences of classes into uncertain sequences. To smooth a training set, we apply an n-filter to it. Then, we obtain the word with large sequences which the length is longer than $2n$, n is ten percent of the training set size.

The sequences of *u* represent *uncertain sequences* where the classes are highly mixed. The *certain sequences* consist of a single letter *x*, $x \in L$. These sequences have a single class and they do not contain any *u* character. We will use these two kinds of sequences to build a fuzzy partition of the training set *T* related to an attribute. *Certain sequences* of letter *x* correspond to the kernels of the fuzzy sets of the partition.

Let *r* be the number of fuzzy modalities we choose for the attribute. We select the *r* largest certain sequences containing one class (Figure 3.4). Suppose *r*=2, to each sequence, we assign intervals from *X*, for instance $[S_1^{\min}, S_1^{\max}]$ and $[S_2^{\min}, S_2^{\max}]$. In the case where we cannot find *r* sequences, we can either reduce the number of applied filters, or select fewer sequences. We summarize this in the following algorithm FPMM (Fuzzy Partitioning using Mathematical Morphology), with *r*=2:

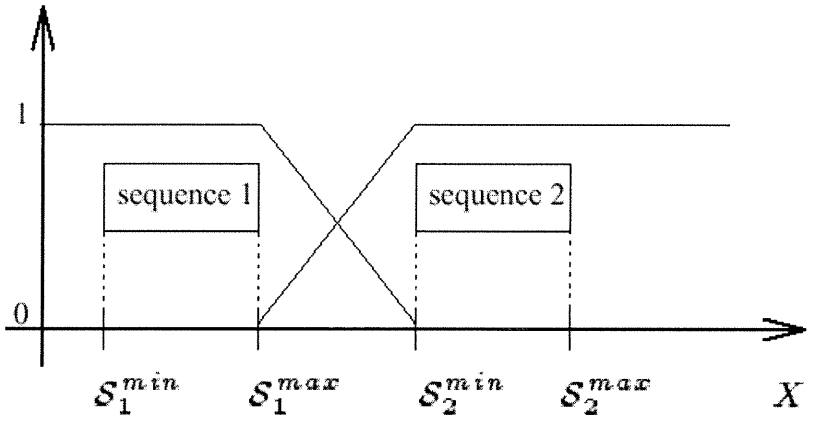

Figure 3.4 Induced fuzzy partition

Algorithm FPMM fuzzify a training set *T* with respect to an attribute defined on *X* in two fuzzy subsets.

1. Transform *T* into a word *v*.
2. For *n* fixed, smooth *v*.
3. Find the two larger certain sequences S_1 and S_2.

4. We denote by S_i^{min} (resp. S_i^{max}) the value associated with the first (resp. last) letter of S_i in X, $i=1, 2$. $S_1 \equiv [S_1^{min}, S_1^{max}]$ (resp. $S_2 \equiv [S_2^{min}, S_2^{max}]$) with $S_1^{max} < S_2^{min}$.
5. The fuzzy partition is defined as a family of two fuzzy subsets. The kernel of the first one is $]-\infty, S_1^{max}]$ and its support is $]-\infty, S_2^{max}[$. The kernel of the second one is $[S_1^{max}, +\infty[$ and its support is $[S_2^{min}, +\infty[$.

Thus, we have an algorithm to induce a fuzzy partition from a training set. This algorithm smoothes a word that induced by a training set. From this smoothed word, we propose a way to define a fuzzy partition.

3.2.2 Constructing the FDT

The FDT is constructed from a set of examples by inductive learning. The set of examples are called training set. Each training set is a case already solved or completely known, associated with the [description, class] pair. A common process of inductive learning is the DTs' construction from a given set of examples [BFO84].

Let A= { $A_1, A_2, ..., A_n$ } be a set of attributes and let C= { $c_1, c_2, ..., c_k$ } be a set of classes. A and C are able to construct from training data: Each training data e_i is composed by an N-tuple of attribute-value pairs (A_j, v_{jl}) associated with a particular class c_k from C. The algorithm is: Given a training set E = { $e_1, e_2, ..., e_n$ } of examples, a FDT is constructed from the root to the leaves, by successive partitioning of the training sets into subsets. The construction process can be done in three fundamental steps. An attribute is selected according to the fuzzy entropy. The partitioning is done by means of splitting the set. The stopping criterion enables us to stop splitting a set and constructed a leaf in the tree. The algorithm is listed in Figure 3.5.

```
Function induce_fuzzy_tree (Example set E, Properties P)
Begin
 if all entries in E are in the same class
  then return a leaf node labeled with that class
   else if P is empty
   then return leaf node labeled with disjunction of all classes in E
    else begin
      fuzzify E;
      select a property pr of P, and make it the root of the current tree;
      for each fuzzy partition f, of pr,
      begin
        create a branch of the tree labeled with f;
        let partition pa be elements of E with f for pr;
       call induce_fuzzy_tree (pa, P), attach result to branch f
      end
    end
end
```

Figure 3.5 FDT algorithm

The algorithm described in Figure 3.5 is discussed in more details:

- **Measure of discrimination**

Haskell said, "In a binary tree classifier, a decision is made at each non-terminal node of the tree based upon the value of one of many possible attributes or features. If the feature value is less than the threshold then the left branch of the tree is taken, otherwise the right branch is taken" [Has98]. This threshold is the measure of discrimination. The choice of an attribute is done by means of a measure of discrimination. It enables us to measure the power of discrimination of an attribute A_j relatively to the class C. It evaluates to which extent the values of this attribute are linked to each modality of the class. Thus, all the attributes can be compared in order to select the best one to split the training set. The choice of the best attribute by means of a measure of discrimination is a heuristic that

enables us to optimize the tree built. This heuristic should minimize the size of the tree. Let's present what heuristic method is in the constructing the FDT.

We suppose that all the values of a continuous attribute A_j are precise in the training set E. Let $\{x_1, x_2, ..., x_n\}$ be the set of values of A_j in the training set E. So the criterion of splitting the fuzzy data is defined as:

$$H_S^*(C \mid A_j) = -\sum_{l=1}^{m_j} P^*(V_{jl}) \sum_{k=1}^{K} P^*(C_k \mid V_{jl}) \log_2(P^*(C_k \mid V_{jl})).$$

In this definition, $P^*(V_{jl})$ is the fuzzy probability, $P^*(C_k \mid V_{jl}) \log_2(P^*(C_k \mid V_{jl}))$ is the star entropy, which the decision C is related to the attribute A_j. $P^*(C_k \mid V_{jl})$ is the fuzzy conditional probability.

The fuzzy probability [Zad68] of modality v_i is defined as:

$$P^*(v_i) = \sum_{1 \le l \le n}^{n} f_{v_i}(x_l) P(x_l).$$

Where the probability $P(x_l)$ is weighted by the frequency of x_l in E, and f_{v_i} is the membership function of the fuzzy set representing v_i. In practice, if n_{x_l} is the number of examples in E with the value x_l for A_j, $P(x_l)$ is approximated with $P(x_l) = \frac{n_{x_l}}{|E|}$ (we denote $|E|$ the total number of examples in E).

Let $\{y_1, y_2, ..., y_q\}$ be the set of continuous values of the decision attribute C in E. The fuzzy conditional probability of modality c_j of C, given v_i, is defined as:

$$P^*(c_j \mid v_i) = \frac{P^*(c_j, v_i)}{P^*(v_i)} \text{ with}$$

$$P^*(c_j, v_i) = \sum_{1 \le l \le q} \sum_{1 \le k \le n} \min(f_{c_j}(y_l), f_{v_i}(x_k)) P(y_l, x_k)$$

In practice, $P(y_l, x_k)$ is approximated with $P(y_l, x_k) = \frac{n_{y_l \wedge x_k}}{|E|}$, where $n_{y_l \wedge x_k}$ is the number of examples in E with both the values x_k for A_j and y_l for C.

This discrimination measure is the uncertainty of the fuzzy decision when the modalities of the continuous/symbolic attribute A_j are known. It is the criterion chosen to order the attributes during the FDTs' construction.

- **Splitting strategy**

The splitting strategy defines how the training set will be split by a question on the value of an attribute. In the case of continuous/symbolic attributes, each modality v_{jl} of an attribute A_j defines a subset ε_{jl} of the training set, composed by the examples that possess value v_{jl} for A_j in which $l = 1...m_j$.

Marsala's splitting solution is using α-cut for each fuzzy set. For instance, with $\alpha = \frac{1}{2}$, the following crisp partition is created: $\forall l = 1,...m_j$, $\varepsilon_{jl} = \{t_i \in \varepsilon \mid \mu_{v_{jl}}(t_i[A_j]) \geq \frac{1}{2}\}$.

- **Stopping criterion**

The stopping criterion T enables us to know if a subset of ε is homogeneous with regard to the class. The process of construction of the FDT will stop the development of a path on a suitable set. The stopping criterions are defined as the following strategies:

1. All the examples of ε, or at least a *majority* of examples, are associated with the same class. Usually, the stopping criterion T is linked to the measure of discrimination of the algorithm. The notion of *majority* is evaluated by means of a threshold for the value of the discrimination measure. The smaller the value of

the discrimination measure, the more homogenous of the training set with regard to the class is.

2. The number of examples in ε is too small to justify a split of ε.
3. There are no more sub-fuzzy sets to use. This stopping criterion is used when the generated length of word is no longer than the two times of the recursion time in the fuzzy partition procedure.

The stopping criterion for the FDTs' construction is quite similar to the stopping criterion in the classical DT.

3.2.3 Classifying with FDT

As mentioned, a path of the tree is equivalent to an *if...then* rule. The premises for such a rule *r* are composed by tests on values of attributes, and the conclusion is that the value of the class labels the leaf of the path:

$$if\ A_{l_1} = v_{l_1}\ and ...and\ A_{l_p} = v_{l_p}\ then\ C = c_k$$

In FDT, a leaf can be labeled by a set of values $\{c_1,...,c_k\}$ for the class, each value c_j associated with a weight computed during the learning phase. Thus, a path of FDT is equivalent to the following rule:

$$if\ A_{l_1} = v_{l_1}\ and ...and\ A_{l_p} = v_{l_p}\ then\ C = c_1\ with\ the\ degree\ P^*(c_1 \mid (v_{l_1},...,v_{l_p}))\ and ...$$

$$and C = c_k\ with\ the\ degree\ P^*(c_k \mid (v_{l_1},...v_{l_p}))$$

A new example *e* to be classified is described by means of values for each attribute $\{A_1 = w_1;...;A_n = w_n\}$. The description enabling us to value the satisfying degree of any observed modality *w* to the edge modality *v* is compared with the premises of the

rule r by means of a measure of satisfying degree [BRB96]. The measure of satisfying degree used here is:

$$Deg(w,v) = \frac{\int_X \mu_{w\cap v} dx}{\int_X \mu_w dx} \quad \text{if} \quad \int_X \mu_w dx \neq 0$$

where μ_w is the membership function associated with the modality w, $\mu_{w\cap v}$ is the membership function associated with the intersection $w \cap v$ between the modality w and the modality v, X is the universe of values where μ_w and $\mu_{w\cap v}$ are defined.

For each premise, a satisfying degree $Deg(w_l, v_l)$ is computed for the corresponding value w_{l_i}. Finally, given a rule r, a description is associated with the class c_j with a final satisfying degree $F\deg_r(c_j)$. The final satisfying degree corresponds to the satisfying degree of the description to the premises of the rule r weighted by the conditional probability for c_j according to the rule r in order to take into account the confidence of the rule:

$$F\deg_r(c_j) = \mathrm{T}_{i=1\ldots p} Deg(w_{l_i}, v_{l_i}) \cdot P^*(c_j \mid (v_{l_1}, v_{l_2}, \ldots, v_{l_p}))$$

Final degrees computed from all the rules are aggregated by means of a triangular co norm $\perp$ (for instance, the *maximum* triangular co norm) to obtain a single degree of satisfy degree $F\deg(c_j)$. If n_ρ is the number of rules given by the FDT:

$$F\deg(c_j) = \perp_{r=1\ldots n_\rho} F\deg_r(c_j)$$

For each value of the class, the description e is associated with such a satisfy degree $F\deg(c_j)$ for each class c_j computed from the whole set of rules. The class c_e associated with e can be chosen as the class with the higher satisfying degree:

$$FDeg(c_e) = \max_{j=1...K} FDeg(c_j)$$

We use such a process of aggregation of degrees in order to have meaningful values of degrees for each class.

3.3 Summary

In this chapter, we introduce Marsala's method to build the FDT. There are many important properties in his approach. Marsala proposes a new formalization of the algorithms to construct FDTs. It highlights various aspects that differentiate an algorithm from the others: The measure of discrimination is used to obtain ideal results during the generalization step; the splitting strategy defines how to split the training set during the construction of the tree; and the stopping criterion defines when the development of a path of the tree has to be stopped. By means of this formalization, FDT is well established.

In chapter four, we will introduce our approach to the FDT, the evolution of Marsala's method.

Chapter 4 *Fuzzy Learning for Symbolic Classification*

From chapter two and three, we know that there are many methods available from the literature to build the FDT. The best method up to now is Marsala's binary FDT [Mar98]. The application of fuzzy set theory enhances the understandability and prediction ability of DTs. By adding the fuzzy partition of the attributes, this algorithm generates the FDTs automatically. The implementation results show that Marsala's FDTs are more robust than the classical ones [Mar98].

In this chapter, we will provide our modifications to Marsala's approach. The modifications focus on the fuzzy membership function and the defuzzification procedure. The results we get are more efficient and explicit.

4.1 Some defects of Marsala's FDT

The C4.5 makes a big success in classification of the symbolic attributes. When the attributes are the continuous variables, Marsala introduces a well-built fuzzy decision model [Mar98] of the TDIDT method, which solves them very well. In this fuzzy decision model, Marsala proposes the mathematical morphology [MB96] operators for automatic fuzzy partition of the continuous attributes. The experiments show that some FDTs lead to a better accuracy rate when classifying the new cases [Mar98].

There are five problems in Marsala's approach. First, when he does fuzzy partition on a set of values, Marsala applies an n-filter to the word induced by the length of the training set, so it can translate small sequences of classes into uncertain sequences. The problem is that the value n is not suitable for our system. Second, in Marsala's method, the shape of fuzzy membership function is shoulder-left and shoulder-right, therefore, the discrimination degrades. We change it to the standard trapezoid shape, and thus, it is more comprehensible. Third, when building the FDT, he uses the crisp partition to split the FDT. We modify it as the trend partition. Furthermore, Marsala's method does not have the pruning procedure. When the trees grow large, the generated rules could be very complicated. We add the enhanced pessimistic pruning procedure into the tree building. At last, instead of using the probability similarity to classify the new cases, we use the min-max algorithm and the vote method to get the predicted value.

4.2 The value of *n*-filter

To smooth a training set, we should apply an n-filter to the training data set. Then we get a word of the large sequences with the length longer than $2n$. According to Marsala's method, n equals to the value of ten percent of the training set size. Each time the program does the recursion for smooth the word for n times. For the regular length data

set, normally n is four or five. Therefore, the computational performances are bad; a lot of time was wasted in the recursion.

According to Marsala, n is the empirical data. We find that the results are a little different whether n is big (e.g. four or five) or small (one or two). So we change n with the logarithm of the data set length. The performance changes a lot and the smooth degrees are little brought down.

4.3 The shape of fuzzy membership function

Now, we would like to discuss the shape of the fuzzy membership function. According to Marsala's FPMM algorithm, we can see that the kernel of the fuzzy membership function is the class characters while the left and the right sides are uncertain parts. But Marsala's method transfers the fuzzy partition as the shoulder-left or shoulder-right (Figure 4.1). Therefore, he omitted the two sides (the left most side and the right most side) of the uncertain parts and transferred them into the certain parts. This will degrade the discrimination of the data set. So we change the FPMM algorithm when we determine the shape of the fuzzy set as the trapezoid shape (Figure 4.2). Therefore, we can exactly get the results from the mathematical morphology transformation. Using the trapezoid shape fuzzy partition, the results are more accurate and the performance is better.

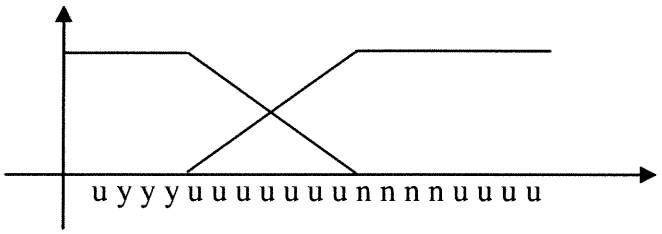

Figure 4.1 Marsala's fuzzy partition

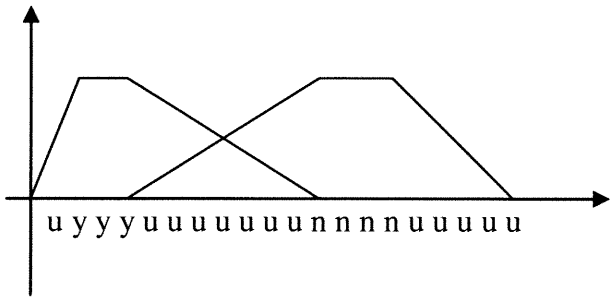

Figure 4.2 Modified fuzzy partition

4.4 Splitting the FDT

Marsala uses the α-cut (=0.5) for splitting the fuzzy sets in his algorithms. It means that he splits the fuzzy sets with the crisp partition. We propose to apply the splitting trend to split the fuzzy sets instead of the crisp value. We split the FDT according to the fuzzy partition got from the mathematical morphology. We think it is more natural and accurate in this fuzzy context. The same idea can also be found in [SBLF00].

4.5 Enhanced pessimistic post pruning procedure

Using TDIDT method to build the DT always continues to subdivide the training set until each subset in the partition contains a single class, or there is no data set to split at all. Therefore, the trees they built are always complex and "over-fit the data". This means that the generated trees have more structures than they should have by inferring the training cases, thus the classification ability may be weak when they treat new cases.

As you know, DTs always apply the pruning procedure to the tree in order to avoid "over fitting the data". But Marsala's method does not provide this procedure, as he thinks that the FDTs he built are shorter than classical ones and therefore avoid the problem and pruning the tree may distort the results.

Although Marsala's FDTs are shorter than the classical DTs and sometimes they do avoid "over fitting the data", we think they need the pruning procedure too, because they are also large and hard to generate the fuzzy decision rules from them. Therefore, after we construct the FDTs, we simplify them with the pruning process. First we provide the pruning procedure in our project, and then we introduce the principle of the enhanced pessimistic pruning method.

A DT is not usually simplified by deleting the whole tree in favor of a leaf. Instead, the idea is to remove parts of the tree that do not contribute to classification accuracy on unseen cases. This procedure is less complex and more comprehensible. In our thesis, we have three steps in our tree pruning procedure after we build the FDTs (Figure 4.3). First, suppose each sub-tree has two branches, of which the classifications are the same. We merge the two branches into one. It is called the branch merging. The process is similar to the one in [XBG00].

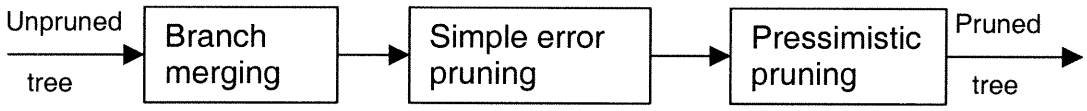

Figure 4.3 The pruning process

Second, when the expected error rate in the sub-tree is larger than that in a single leaf of one of its branches, we replace a whole sub-tree by its leaf node or its branches. This is the simple error pruning process. The procedure of simple error pruning is: starting from the bottom of the tree and examining each non-leaf sub-tree. If the replacement of this sub-tree with a leaf, or with its most frequently used branch, would lead to a lower predicted error rate, then the tree is pruned accordingly. The error rate for the whole tree decreases as the error rate of any of its sub-trees is reduced. This process will lead to a tree whose predicted error rate is minimal with respect to the pruning process.

Third, we prune the tree according to the enhanced pessimistic pruning method. This method is developed by Quinlan [Qui93]. It only uses the training set from which the tree is built as the estimation. The re-substitution error, which is the error rate on pruning a

sub-tree using the observation on the training set from which the tree is built, is estimated and adjusted to reflect this estimate's bias.

Consider a leaf covering N training cases, E of them classified incorrectly, the re-substitution error rate for this leaf is *E/N*. We define this result as the probability of error over the entire population of cases covered by this leaf. For a given confidence level *CF* (in our thesis *CF* equals to 0.25), the upper limit ($U_{CF}(E,N)$) on this probability can be found from the confidence limits for the binomial distribution. The predicted error rate of a leaf is considered to be equal to this upper limit, on the argument that the tree has been constructed to minimize the predicted error rate. Although this conclusion is lack of the proof of the statistical theory, like many heuristics with questionable underpinnings, it seems produces the acceptable results.

To simplify the calculation, error estimates for leaves and sub-trees are computed assuming that they are used to classify a set of unseen cases of the same size as the training set. A leaf covering N training cases with a predicted error rate of $U_{CF}(E,N)$ will give rise to predicted $N \times U_{CF}(E,N)$ errors. Therefore, for a sub-tree with leaves, if the predicted error rate for the leaf ($N \times U_{CF}(E,N)$) is lower than that of the sub-tree ($\sum_{i=1}^{n} N_i \times U_{CF}(E,N_i)$), this sub-tree is replaced by a single leaf.

4.6 The min-max algorithm and the vote method

After generating the FDT, we use different methods to classify the new cases. They are the min-max algorithm and the vote method. These two methods are explicit and easy to be implemented and the results are accurate comparing with the Marsala's satisfying degree method. From Marsala's FPMM algorithm, we get binary fuzzy partitions and thus generate the FDTs which are binary trees. The FDTs' number equals to the class number. For each class, we have the correspondent tree. For example, if the classes are

"*0*" and "*1*", we can generate two trees. The leaves of one tree are labeled with this class "*0*" and not this class "*!0*"; The leaves of the other tree are labeled with this class "*1*" and not this class "*!1*". For each FDT, we use min-max algorithm to get the final value of each branch labeled with this class or not this class. For all the FDTs, we use the vote method to get the final class label of the new case.

The min-max algorithm is similar to the min-max inference of fuzzy rule evaluation procedure [NK99]. Suppose for each tree, from the top node to each leaf, we can generate the fuzzy rules with the number equaling to that of leaves. So the FDT model consists of a set of fuzzy rules R_j like

$$R_j: \textit{if } x_1 \textit{ is } A_j^{(1)} \textit{ and } \ldots \textit{ and } x_n \textit{ is } A_j^{(n)} \textit{ then } y \textit{ is } B_j$$

where $x_1,\ldots,x_n \in \mathrm{IR}$ are the fuzzy values of the non-leaf node, and $y \in \mathrm{IR}$ is the leaf value. $A_j^i: \mathrm{IR} \rightarrow [0,1]$ is the fuzzy value. $B_j(y)$ is the final label of each leaf. $B(y)$ is the output fuzzy set label.

$$B_j(y) = \min\{A_j^1(x_1),\ldots,A_j^n(x_n)\}$$

$$B(y) = \max_j\{B_j(y)\}$$

In a word, the min-max algorithm is minimum truth-value along each tree path, and maximum truth-value for each end leaf.

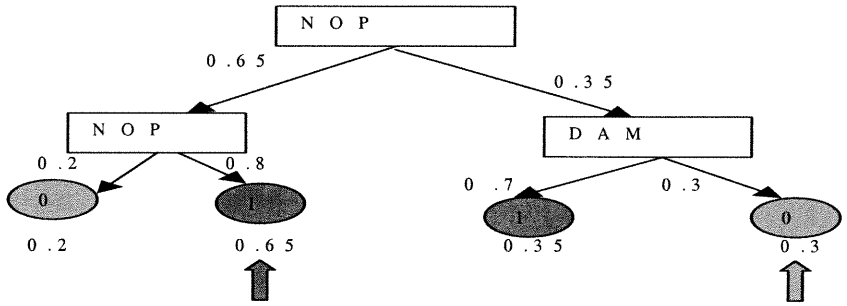

Figure 4.4 The min-max algorithm

For example, in Figure 4.4 according to the min-max algorithm, in the left branch, we choose 0.65 for class "*1*", and 0.2 for class "*0*"; in the right branch, we choose 0.35 for class "*1*", and 0.3 for class "*0*". The leaf value is the minimum of the branch value. For all the leaves, we select the biggest value as the final label of the tree. So for class "*0*", the fuzzy value is 0.3 and for class "*1*", the fuzzy value is 0.65. As for each class, we have the correspondent tree, we have the fuzzy values of every class, and we take these values as the input data to the vote method.

After we get the class value of each tree, the next procedure is using the vote method to get the final class label of the new case considering all the FDTs. The vote method considers the fuzzy value as the weight of the class, and integrates both sides of this class and not this class. The result of the vote method is the class label of the new case.

The vote of the class C is calculated as:

$$\mu(c_i) = \frac{m_{c_i}(\{c_i\}) + \frac{1}{|n-1|}\sum_{c_j \neq c_i} m_{c_j}(\{\overline{c_j}\})}{|n-1|}$$

Figure 4.5 The vote method formula

In Figure 4.5, $m_{c_i}(\{c_i\})$ and $m_{c_j}(\{\bar{c}_j\})$ are the fuzzy values of this class (i.e. "*1*") and not other class (i.e. "*!0*"). *N* is the tree number and $\mu(c_i)$ is the vote of the class. We select the class label with the biggest vote value as the final class label of the new case.

For example, suppose we have fuzzy values in Table 4.1. We have three FDTs, named *Tree1*, *Tree2*, *Tree3*; and three data sets *E1*, *E2*, *E3*. For each tree, it has two classes. Using the vote method, we select the biggest value in vote column: For *E1* we select 0.78, for *E2* we select 0.57, for *E3* we select 0.55, so the correspondent class labels are given to each data set, *E1*'s class label is *C1*, *E2*'s class label is *C1*, *E3*'s class label is *C3*.

Ex	Tree 1		Tree 2		Tree 3		Vote		
	C1	$\bar{C}_1$	C2	$\bar{C}_2$	C3	$\bar{C}_3$	C1	C2	C3
E1	1	0	0	0.3	0	0.8	**0.78**	0.2	0.08
E2	0.55	0.6	0.3	0.7	0.32	0.48	**0.57**	0.42	0.48
E3	0.1	0	0.8	0.1	0	0.6	0.22	**0.55**	0.02

Table 4.1 The vote method example

4.7 Summary

In this chapter, we introduce how we modify the Marsala's FDT approach and set up our own fuzzy decision models. The general inference procedure of the FDT we proposed for symbolic classification shows in Figure 4.6.

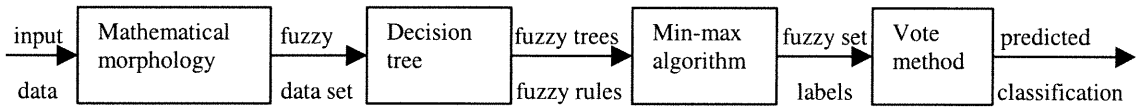

Figure 4.6 The procedure of the FDT

The FDT model that we get is more clear, shorter and easier to explain. The features (or modifications) of our fuzzy decision models are:

- We select the n which equals to the logarithm of the data set length as a recursion time when we smooth the sequence of the continuous values. The length of the word's longest definite part should be longer than $2n$; there should be only one word in each sequence. On the other hand, if we cannot find the word (It often happens when the data set length is too short), we reduce the recursion time n.

- We use mathematical morphology algorithm to do the fuzzy partition inference over a set of continuous values. These fuzzy partition shapes are trapezoid instead of the shape of shoulder-left and shoulder-right. This fuzzy partition is depending on the class, thus this method is better than the normal clustering methods.

- We propose using the splitting trend instead of the splitting crisp value to split the fuzzy partition.

- We introduce the enhanced pessimistic pruning procedure to prune the FDTs. We get more general trees and the rules. Therefore, the rule-based systems we get are more robust and powerful.

- When we apply the FDTs to classify the instances, we implement the min-max algorithm to get the fuzzy values of each tree: One belongs to this class; another belongs to not this class. After that, we use vote method to select the class label of the new case. The main idea is, giving the non-classification fuzzy value to the other classification fuzzy value, and selecting the biggest one as the class label.

Using min-max algorithm and vote method to defuzzify the FDTs is clear and easy to implement. It is comparative to the Marsala's satisfying degree method. Actually, we also use the NN to do the defuzzification procedure, as the NN can deal with the symbolic values. The results are a little better than the defuzzification procedure we just proposed. In chapter five, we will focus on these issues as well as how we use the neuro-fuzzy approach to deal with the continuous classes.

Chapter 5 *Fuzzy Learning for Continuous Regression*

In chapter four, we discussed the fuzzy learning procedure for the symbolic classification data sets. Encouraged by the good results we obtained with them, we tried to deal with the continuous regression data sets. In this chapter, we mainly present the methods such as transferring the continuous classes into the symbolic classes and NN for defuzzification procedures. We also introduce the concept of NN, and the classical defuzzification methods.

5.1 Introduction

Predicting the continuous regressions is a big task in ML. There are many methods in NNs or RT algorithms. Normally, NNs have become equally popular due to the relative ease of application and the ability to provide gradual responses. However, they are lack of similar levels of comprehensibility; they are really black boxes. We can not use them

as generating the decision rules. The problems of the RT are that the models are complex and incomprehensible and the performance is not computationally efficient.

We present a new procedure that can deal well with these defects. This new approach is, first we transfer the continuous classes into the symbolic ones using the SOM algorithm [YS95], and then use the FDTs to classify these symbolic classifications, and at last introduce the NN defuzzification procedure comparing with the classical defuzzification process (MOM model) to get the crisp values we predicted. We call it fuzzy regression process (Figure 5.1).

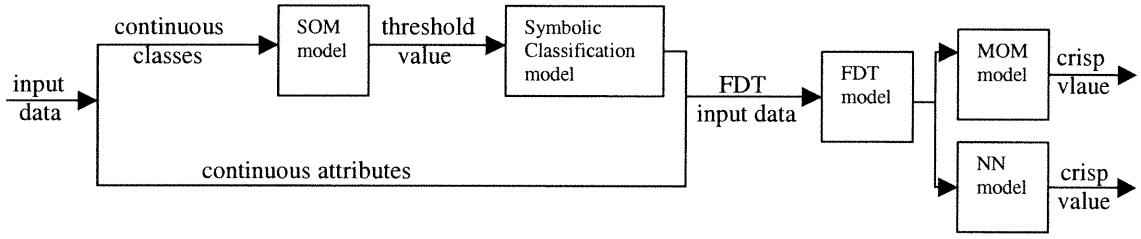

Figure 5.1 The general regression procedure

5.2 Concepts of neural network

As our fuzzy regression process has two kinds of NNs, let's introduce the concepts of the NN first. NN is a model that intelligently arises in the forms of simple, interacting components through a process of learning or adaptation by the connections among components. It processes the data in the layers of the neurons, and the process is independent and simultaneous.

NN is not just another learning algorithm or a regression technique; it represents a new computation model whose idea is borrowed from the human brain. It has thousands of simple processors (called processing units, or neurons) connected by thousands of adaptive weights, as illustrated in Figure 5.2.

A neuron is based on the following parameters:

- Input connections (or inputs): $x_1, x_2, ..., x_n$. There are weights bound to the input connections: $w_1, w_2, ..., w_n$.
- Input function *f*, calculates the aggregated net input signal to the neuron: *sum=f(x,w)*.
- An output function calculates the output signal value emitted through the output of the neuron: *y*.

NN can be used in supervised, unsupervised, and reinforcement learning procedures. When sufficient data sets are available, NN can do classification, clustering, and prediction. In our algorithm, we use two typical NN algorithms, SOM is for the clustering and the back propagation is for the training and prediction.

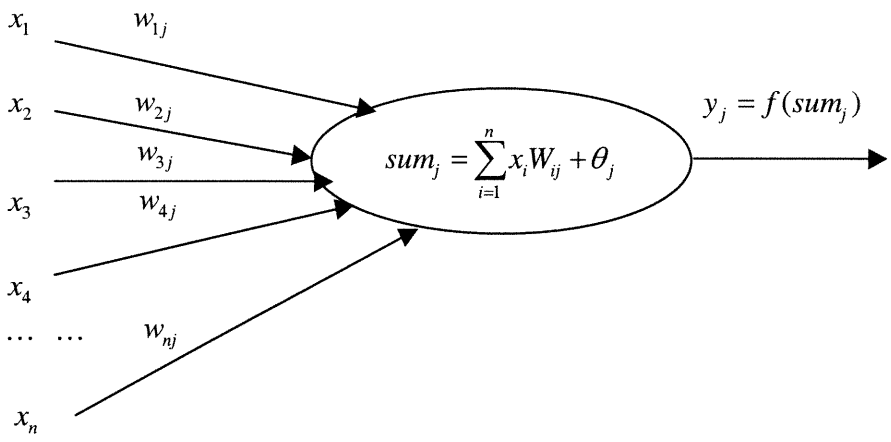

Figure 5.2 An NN process unit

5.3 Self-organizing maps

The transformation of the continuous classes into the symbolic classes is the process of fuzzy conceptualization. This process is used to reduce the information overload in the decision making process. For instance, the numerical length of a human may be transferred into a linguistic terms, such as tall or short. The membership function can be determined by the experts' opinion or people's common perception. It can also be determined by the statistic method [CT86], clustering (the C-means method) [SBC01] and clustering based on NN (SOM method) [YS95].

The C-means clustering is an interesting approach. Sahraoui et al. propose such a method in [SSB00]. Originally, this paper introduces a method to build software quality predictive models which combines both fuzzy decision processes from the software metrics and the heuristic knowledge from the field. In exploring the software metrics, the authors propose that the C-means clustering can be used for fuzzy partition. Using this method, we can transfer the continuous regressions data into the symbolic classifications data.

The C-means method is based on the distribution of the measurement data. We calculate the frequency of each input value and derive homogeneous clusters from the resulting curve. The clusters represent the fuzzy labels of the input metric. If it is hard to derive homogenous clusters, we should apply a logarithmic transformation to the frequency to boost the smaller values and flatten the larger ones. Finally, integrated with the domain knowledge, we can get some (normally two or three) homogeneous clusters of values. Starting from these clusters, we define some fuzzy labels for the selected metric and assign a membership function to each of them. The fuzzy membership function can be trapezoid or triangle shape.

The C-means is a good method in transferring the continuous regression to the symbolic classification: we try to deal with one of the data sets using the C-means method, the results are satisfying. However, it has inconvenience that at present we cannot do the

fuzzification process automatically. So we turn to another method: SOM, the NN clustering method, in which the process can be implemented automatically.

5.3.1 The principle of SOM

The SOM algorithm introduced by Kohonen [Koh95] has become one of the most popular and practical NN models. The SOM represents the result of a vector quantization algorithm that places a number of reference or codebook vectors into a high-dimensional input data space to approximate to its data sets in an ordered fashion. When local-order relations are defined between the reference vectors, the relative values of the latter are made to depend on each other as if their neighboring values would lie along an "elastic surface". This "surface" becomes defined as a kind of nonlinear regression of the reference vectors through the data points. A mapping from a high-dimensional data space $\Re^n$ onto a two-dimensional lattice of points is thereby defined.

The process in which the SOM is formed is an unsupervised learning process. Like any unsupervised classification method, it can be used to find clusters in the input data, and to identify an unknown data vector with one of the clusters. We use this property of the SOM to identify the threshold value from the input vector and use it to partition the continuous classes.

The SOM is a single-layer NN model (Figure 5.3). It has only one input layer and one output layer. Each time an input vector presents to the network, its distance to each unit in the output layer is computed, and the output unit with the smallest distance to the input vector is declared as the "winner". The winning unit and a set of units in the neighborhood weights are adjusted by moving the weights toward the input vector. Initially, the neighborhood and the learning rate are quite large. As the training process makes progresses, the neighborhood shrinks and the learning rate is decreased. At the end, a topographic map is created.

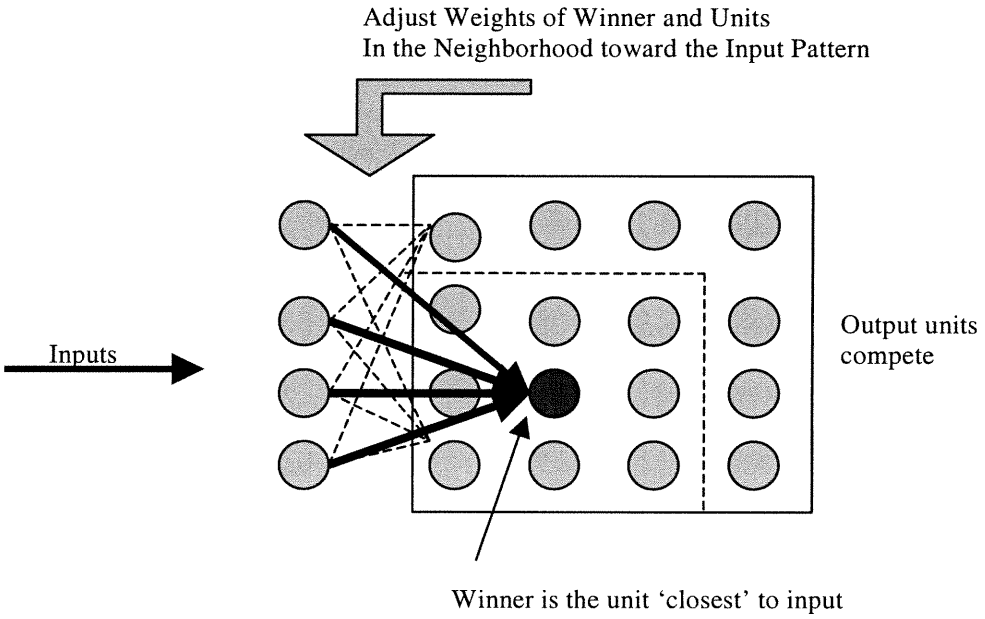

Figure 5.3 SOM

We illustrate how we can get the "winner" in details:

1. The continuous values of classification for all objects $u \in U$ can be represented by $X=\{x(u), u \in U\}$. At time *0*, the winner is initially set to be evenly distributed on the range of X, such as:

$$m_i = min\{x, x \in X\} + (max\{x, x \in X\} - min\{x, x \in X\}) \times (i-1)/k-1),\ i = 1, \ldots, k.$$

2. The winner is then adjusted iteratively in order to reduce the total distance of X to M, which is defined as:

$$D(X, M) = \sum_{x \in X} \min_i \|x - m_i\|.$$

3. Do it recursively, until the $D(X, M)$ converges. Therefore we can get the winner m_i.

5.3.2 Implementation of SOM

We selected this crisp value ("winner") as the threshold value. Now we can generate the linguistic terms of the continuous classes using this threshold value. The class value is the α-cut (=0.5) of the membership function, which means if the class value is smaller than the threshold value, and then it is labeled as "small", otherwise it is labeled as "large". So for all the continuous classes, we have the fuzzy labels on them which are either "large" or "small". After this transformation, we get the input data sets of the FDTs. The format of the transferred data sets is exactly the same as those of the symbolic classification data sets. So we use them to do the fuzzy decision induction that we presented in chapter three and chapter four.

Implementing SOM to all the attributes in the data sets, we can get the linguistic terms of them, as well as the special attribute, the classes. It is useful when we generate the fuzzy decision rules from the FDTs. We can easily put these linguistic terms into the rules construction.

5.4 Comparison of defuzzification methods

Using the transferred data sets as the input ones, we can easily build the FDTs and generate fuzzy rules. We can get the predicted symbolic classifications by using the fuzzy decision method and find that the accuracy rates of them are about eighty percent. It matches the results with the FDTs for the symbolic classifications. But our goals are not to predict the symbolic classifications; we want to predict the continuous ones. Therefore, we should defuzzify the FDTs we have got.

5.4.1 Classical defuzzification methods

The procedure of defuzzifying the FDT is a little bit different from the vote method presented in chapter four. As we want to get the precise prediction of the continuous values, we have to use more precise defuzzification procedure. After we use the min-max algorithm to get the fuzzy values from all the leaves with the class label, we input these fuzzy values into the defuzzification procedure (Figure 5.2 MOM model).

For example, we use the two different methods separately: the center of gravity (COG) and the mean of maximum (MOM) to get the crisp values, the predictions of the classifications. The COG method finds the center of gravity of the union ($\cup$) of the fuzzy sets. It is illustrated in Figure 5.4:

$$\Gamma_{COG}(\mu(x)) = \frac{\int_{x_{\inf}}^{x_{\sup}} \mu(x) \bullet x dx}{\int_{x_{\inf}}^{x_{\sup}} \mu(x) dx}$$

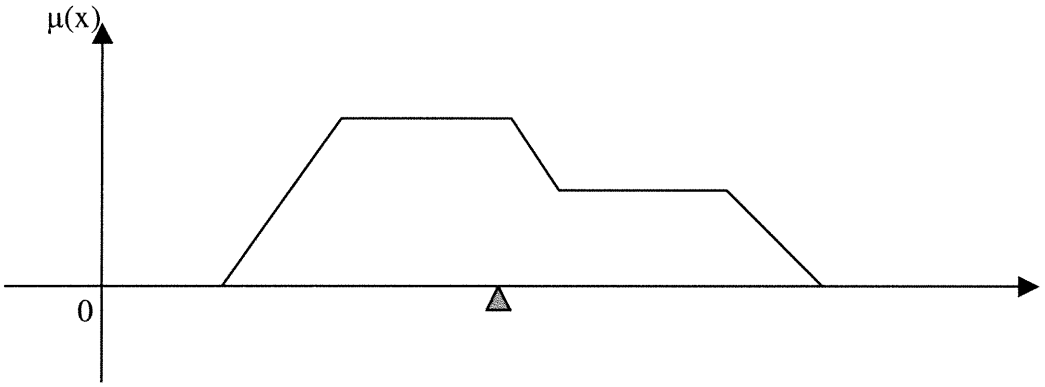

Figure 5.4 The COG method

The MOM takes the mean of those points where the membership function is at a maximum. It is illustrated in Figure 5.5:

$$\Gamma_{MOM}(\mu(x)) = \frac{\int_{\mu(x)=\sup_{x\in\chi(\mu(x))}} xdx}{\int_{\mu(x)=\sup_{x\in\chi(\mu(x))}} dx}$$

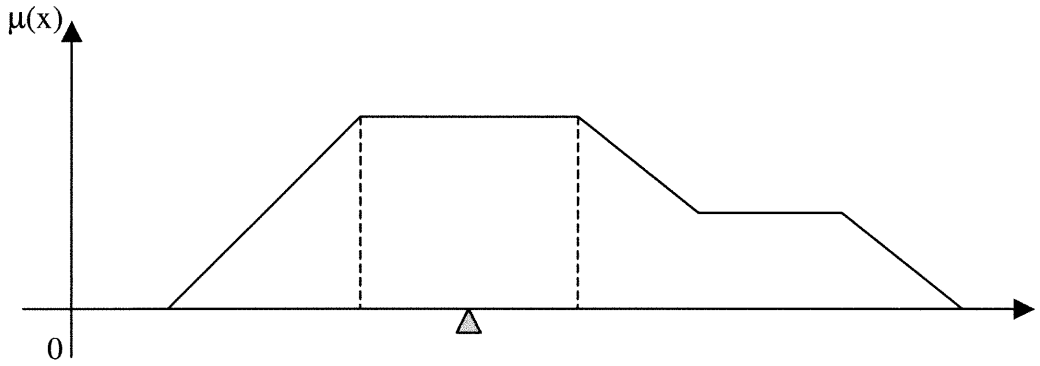

Figure 5.5 The MOM method

We used two groups of data sets to implement these two defuzzification methods, and we found that the results vary heavily depending on the methods we used: results with MOM were better than those with COG. This suggests that there are subjective factors in the prediction. This conclusion matches the conclusion of Veliev et al. [VRS99].

We choose the NN as defuzzification procedure, inspired by many applications of integrating the NN with the fuzzy system [Has98], and the conclusion of [YS95]: the classification accuracy may be further improved if converting membership functions and fuzzy rules into NNs and using the learning mechanism of the NNs to tune the membership function.

5.4.2 Neural network: an objective defuzzification procedure

There are many applications to do the defuzzification procedure using the NN [VRS99] [Web95] [HRG96]. The typical method is presented by Veliev et al. in [VRS99]. In their

approach, they do not model the fuzzy processes using layers of the NN, but taking the outputs of the fuzzy system as inputs to the NN. The NN outputs are used to determine the crisp value. Accepting the idea from their approach, we use NN for defuzzifying the FDTs. The differences between the two approaches are the inputs and the outputs of NN. In their approach, the NN inputs come from the fuzzy inference system, and the NN outputs are the intermediate results then determine the crisp value. In our approach, the NN inputs come from FDTs, and the NN outputs are the crisp value directly.

We select the standard back propagation algorithm as the implementation of the NN. It has the feature of the feed-forward connection topology, meaning that the data flows through the network in a single direction, and uses a technique called the backward propagation of errors to adjust the connection weights through the hidden layers.

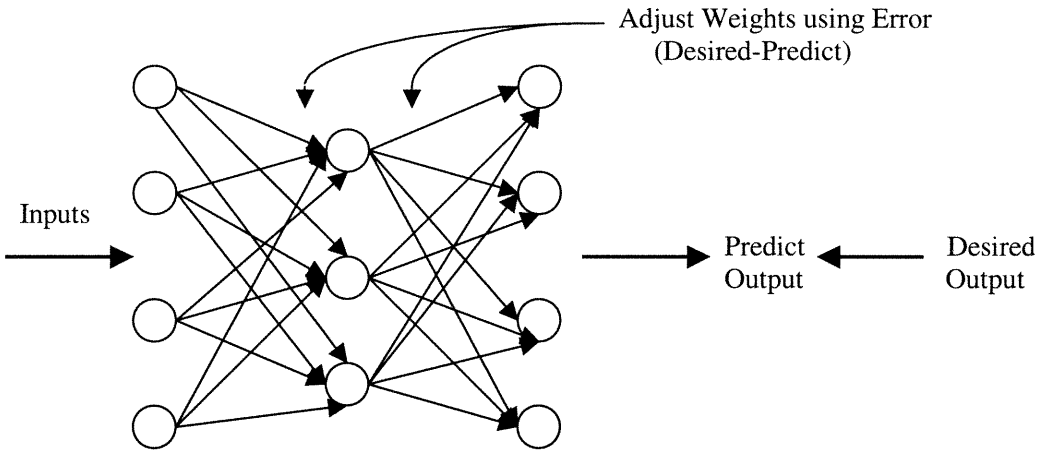

Figure 5.6 The back propagation NN

Figure 5.6 shows a back propagation NN and illustrates the three major steps in the training process. First, input data are presented to the input layer of units on the left, and flow through the network until they reach the network layer of units on the right. This is called the forward pass. The activations or values of the output units represent the actual or predicted output of the network. The desired output value is also presented to the

network, because this is the supervised learning. Next, the difference between the desired and actual output is computed, producing the network error. This error term is then passed backwards through the network to adjust the connection weights.

The implementation training of the back propagation NN is using the data we get from the FDTs. As we explained in chapter four, for each edge of the tree, we can get the fuzzy value of the leaves using the min-max algorithm. Suppose we have two classes *small* and *large*, thus we have two trees labeled with "small" and "large". For each tree, we have four leaves, of which two are the class labels (*small* or *large*), and the others are not the class labels (*! small* or *! large*). Therefore, for "small" tree, we can get two fuzzy values of the *small* label leaves using the min-max algorithm, and so are two *large* leaves of the "large" tree.

We use these four fuzzy values as the input data of the NN. For training procedure we put both of the input data and the desired classes (continuous classes), and for the test procedure we input the same data sets, but the desired classes are only for comparison with the predicted outputs of the NN. The results show that the NN defuzzification procedures are better than the system using the classical defuzzification methods.

5.4.3 Discussion on fuzzy membership function shape

There is a difference between the classical defuzzification and NN method. In classical defuzzification methods, such as COG and MOM, they need the fuzzy partitions as many as they need. When they defuzzify, they generate the crisp value with more fuzzy values, just like considering more various situations and thus their prediction would be more accurate. In some fuzzy systems [Has99], the number of fuzzy partition is six or seven. But in our opinion, it is quite complicated if the fuzzy number is more than three.

Another issue is when we use the classical defuzzification process, such as COG and MOM, the fuzzy membership function shape, which gets from the procedure of the

transferring the continuous classes to the symbolic classes, should have a single peak. If the symbolic output is desired, flat-topped membership functions in which adjacent functions intersect at full confidence are usually desirable; but if the continuous output takes place, membership functions should usually have a single peak, and adjacent functions should usually intersect at about half of full confidence [WebSil]. So we have the fuzzy membership function in Figure 5.7.

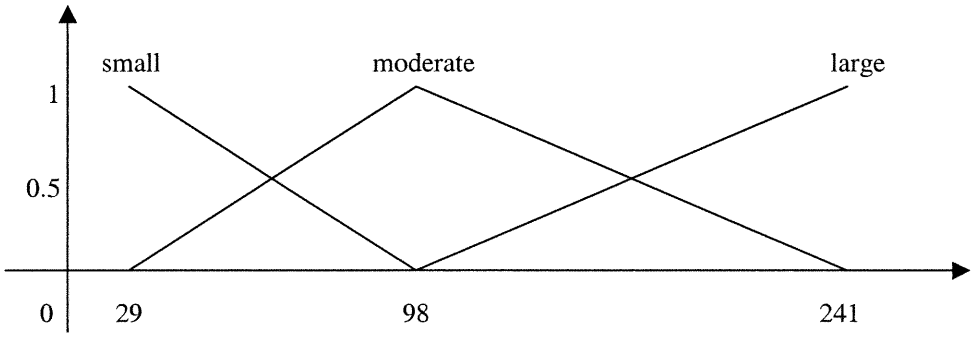

Figure 5.7 The single peak fuzzy partitions

For the NN defuzzification, we only use the threshold value distinguishing the class label *small* and *large*. Here, we do not care how many linguistic terms we have, we only care whether or not we can generate the correct (not too simple) trees. In such a way, we can get the suitable numbers of the input data for the NN.

5.5 Summary

In this chapter, we introduce a new approach to deal with the continuous regressions. We use the fuzzy decision method and integrate with the SOM in transferring the continuous classes into the symbolic classes and the back propagation NN for the defuzzification procedure.

The reason that we adapted the NN as the defuzzification procedure instead of selecting one of the classical defuzzification methods is that the NN uses all the fuzzy set values

which are fired in the FDT, while the classical defuzzification method just uses the representative fuzzy set values. The results suggest that these integrations we proposed are comparable with the technology used in ML.

In chapter six, we will introduce how we implement the algorithms we presented in chapter four and chapter five, and especially how we embed the SOM package, the *C* program in our java program.

Chapter 6 *Implementation*

In chapter four and five, we described the algorithms of how to realize the FDT for the symbolic classification data sets and the fuzzy regression methods for the continuous regressions data sets. In this chapter, we will show that although we describe the two algorithms separately, we implement them in one java program with the different options.

6.1 The procedure

We use java language to implement our program because java has the JVM which is platform free and easy to implement ML algorithms. In addition, this program is part of a big project of CRIM and the other programs of the project are implemented in java.

The procedure of our java program is in Figure 6.1. We have an option for the input data sets with the symbolic classifications or with the continuous regression. When the data sets are with the continuous regressions, first the program changes the continuous classes into the symbolic ones. We have another option using the training data set as the testing data set or using the ten folders cross validations. After we generate the tree, we have the third option of whether post pruning the tree or not. At last, we get the predicted results using both of the classical MOM defuzzification method and the back propagation NN defuzzification method.

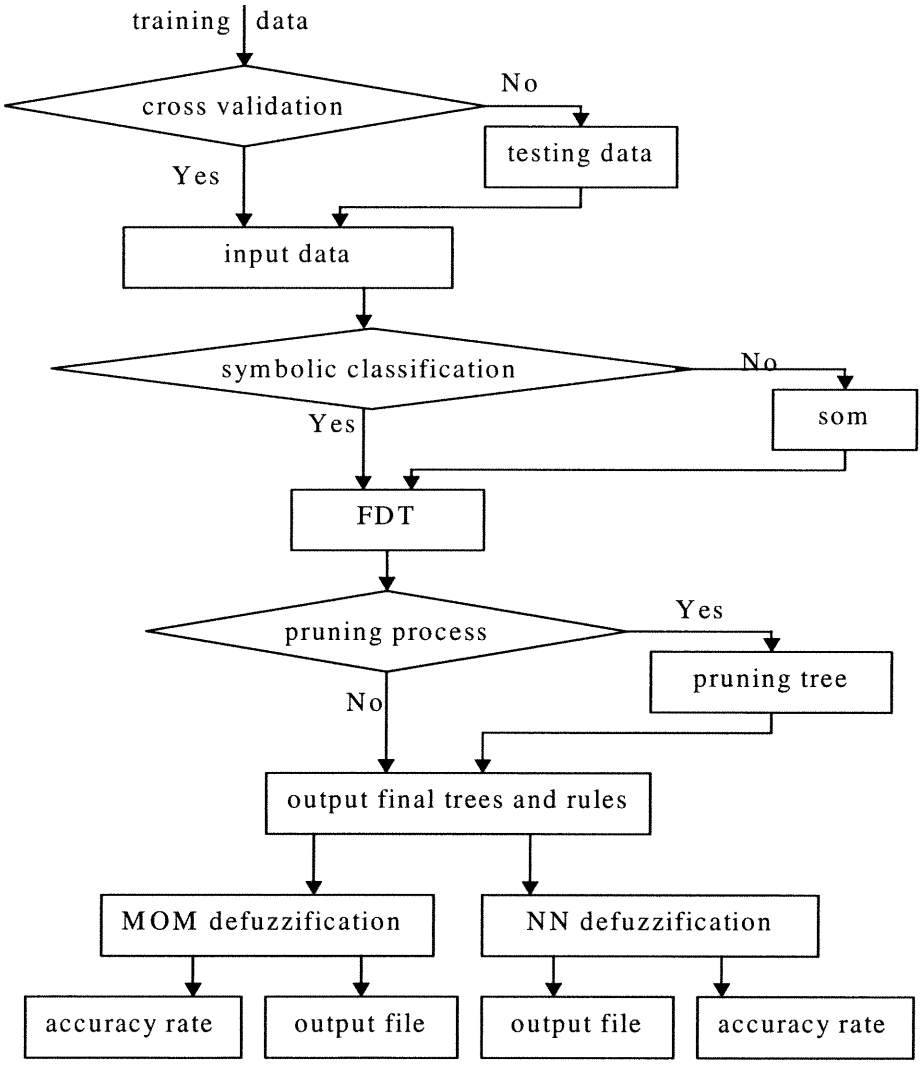

Figure 6.1 The program follow chart

The input data are in training data and testing data forms. Both of them have the results. In training data, results are for the training the algorithm; in testing data results are for the comparison. In fuzzy decision system, we use the software quality prediction data generated from DISCOVER. In fuzzy regression system, we use the hydropower station data from HRM (Hydropower resources management group at Alcan Ltd.).

6.1.1 The tools

We use JBuilder Foundation 4.0 to organize the project, edit the java source files, compile and execute them. It can be downloaded from [WebBor]. For JDK, the 1.3.0 version, we can get from [WebSun]. The platform is Windows 2000 professional version. In our program, we embed the *C* program, the Self-Organizing Map Program Package (SOM _PAK) Version 3.1 [WebSom], into our java program for calculating the threshold value to split the continuous classes. In our program, there are twenty-nine classes, two interfaces, and two execute files.

6.1.2 Embedding SOM_PAK into java program

We use the SOM_PAK to get the split point for transferring the continuous classes into the symbolic classes. The SOM Programming Team of the Helsinki University of the Technology provides the SOM_PAK for the research purpose. All programs in this package are written in ANSI C. This package includes the *C* source code, the makefile and the illustration of how to use it. The principle of the SOM is presented in chapter five. Here, we mainly talk about how it embeds in our java program.

The SOM_PAK can be downloaded for anonymous ftp user at the internet site cochlea.hut.fi (130.233.168.48). All programs and documentation are stored in the directory /pub/som.pak. It has two versions, one for UNIX and the other for DOS. Because we use JBuilder on the Windows 2000, we only consider the DOS version of the

SOM_PAK. After downloading the SOM_PAK, we use makefile to make all the source code in the package and get some executive files. Among these executive files, we only choose two files, raninit.exe that is the initiate file and vsom.exe, which gets the split point value, the result. For implementation, we copy the two files into the working directory of JBuilder.

Figure 6.2 shows how these two command lines are embedded in our java program.

```
public void set_Ls() throws IOException {
        String cmdArray=null;
          Process child=null;
        Runtime rt=Runtime.getRuntime();
            try{
                cmdArray="randinit -din data.dat -cout data.cod -xdim 1 -ydim 1 -topol
hexa -neigh bubble -rand 62";
             child=rt.exec(cmdArray);
             cmdArray="vsom -din data.dat -cin data.cod -cout data.cod -rlen 100000 -
alpha 0.02 -radius 3";
             if(child.waitFor()==0)
             child=rt.exec(cmdArray);
             new OutputPollster(child.getErrorStream()).start();
             new OutputPollster(child.getInputStream()).start();
             } catch(IOException e){
             throw e;
               }
               catch(Exception e) {
                   throw new IOException(e.toString());
                   }
                 }
```

Figure 6.2 The command lines embedded in Java program

Please pay attention to the command line "cmdArray" in Figure 6.2. We use "cmdArray" twice in the try-catch parses: The first cmdArray is the map initialization. The reference vectors of the map are first initialized to tentative values. The second cmdArray is the map training, in this phase, the reference vectors in each unit converge to their 'correct' values. After these two phases of training, the outputs of the *C* program are ready to be used in the java program.

In these command lines, there are some parameters:

-din	name of the input data file.
-cout	name of the file to which the reference vectors are stored.
-xdim	number of units in the x-direction.
-ydim	number of units in the y-direction.
-topol	the topology type used in the map.
-neigh	the neighborhood function type used.
-rand	parameter that defines a new seed for the random-number generator is defined.
-cin	name of the file from which the reference vectors are read.
-rlen	running length (number of steps) in training.
-alpha	initial learning rate parameter. Decreases linearly to zero during training.
-radius	initial radius of the training area in SOM algorithm. Decreases linearly to one during training.

When we embed the command line in our java program, we should start a new thread in the java program [WebDev] [WebVen]. In Figure 6.2, We have two new threads: "new OutputPollster (child.getErrorStream()).start()" and "new OutputPollster (child.getInputSteam()).strat()". The reason is that the outputs of the program we just launched using "(java.lang.Runtime) rt.exec()" are directed to our execution context, not to the system console. In other words, the "Process" object (referenced by "child") is receiving the output. Therefore, we should poll the "InputStream" objects and get the desired output. However, if we do this inside the same thread that executes the "commandArray", we should block until the "Process" object "child" terminates, which may not be desirable. In this case, we should take the polling of the outputs of "Process" object "child" on the separate threads as we do.

6.1.3 Program outputs

The input files of the program are different when the option is cross validation or not. When they select cross validation, the input file is only one file, because the program divides the input file into ten parts and select nine parts as training data and one part as testing data. If they do not need cross validation, the input files are composed of training file and testing file accordingly.

In our program, only when we deal with the continuous regression, we have the disk I/O. The program generates the data.dat file as the input file in the SOM command line, and out puts the file named data.cod that contains the split point value for the program use. The program generates not only the FDTs (Figure 7.3) and fuzzy decision rules (Figure 7.4) but also the fuzzy induced results and NN defuzzification results (Figure 6.3).

```
Please enter the training file name:  iris.txt
Please enter the testing file name:  iris.txt

The Fuzzy induced Result is:      (150/7.0)
The Neural induced Result is:     (150/1.0)
```

Figure 6.3 The output of the program

Using the NN for the defuzzification, we refer to the program from [JJ01]. The interface of our program in JBuilder is Figure 6.4.

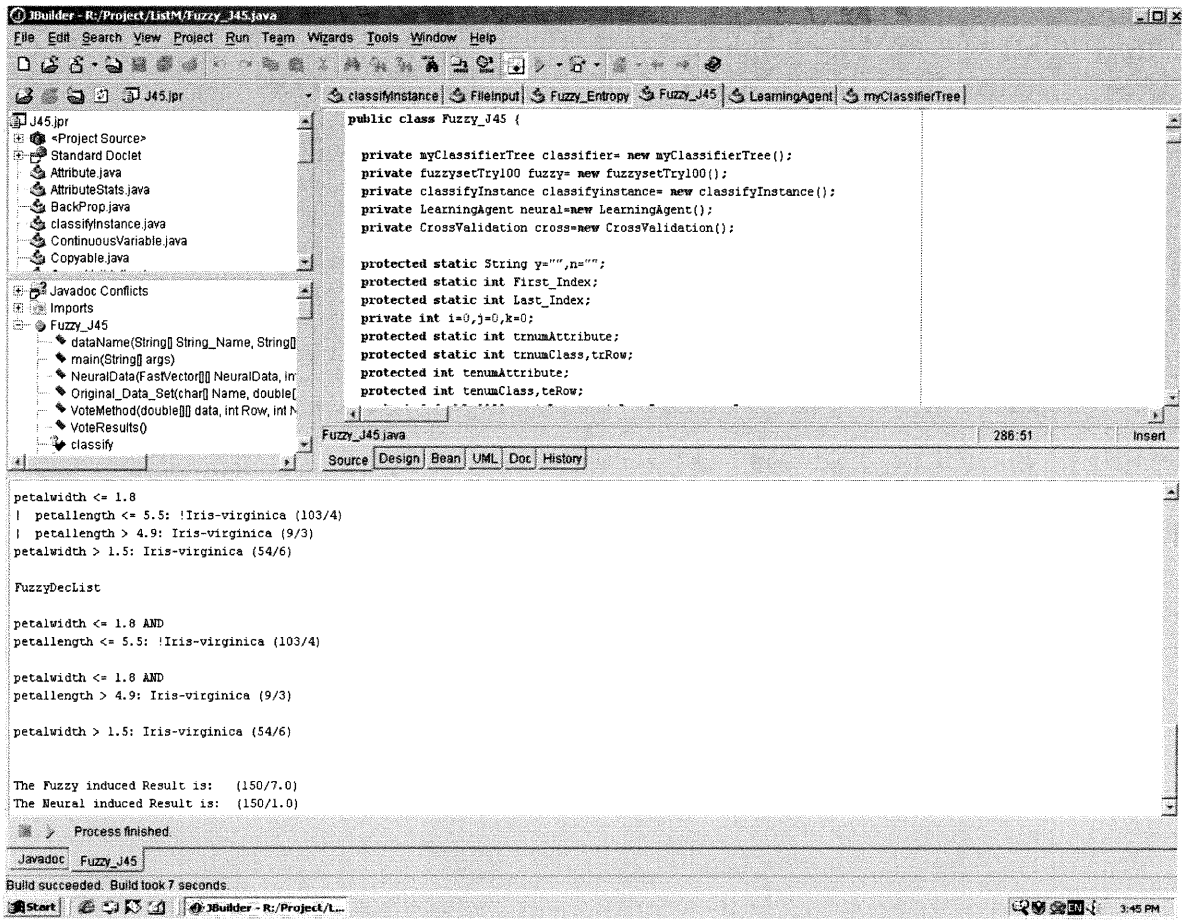

Figure 6.4 The interface of JBuilder

With the output results on the screen, the program out puts the details of the comparison of predicted results and actual values with two defuzzification methods in the files of fuzzy_results.txt and neural_results.txt (Figure 6.5, Figure 6.6). With these two files, we can easily find out which predicted examples are correct.

iris results

Original Class	Predicted Class	Errors
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	
Iris-setosa	Iris-setosa	

Figure 6.5 The output file: fuzzy_results.txt

Neural results

Original Class	Predicted Class	Error
150.0	160.89354497604798	
148.0	137.23702597717374	
145.0	135.9394021397919	
146.0	59.24113867973112	1
145.0	132.01612896854093	
150.0	168.74844634675694	
159.0	172.050443914336	
172.0	157.25723325488119	
184.0	167.25305548585735	
191.0	195.23684785146452	
193.0	175.61353654492865	
188.0	192.96666279853775	
182.0	129.22545255286883	2
81.0	81.33307094708543	
80.0	130.52894966122682	3

Figure 6.6 The output file: neural_results.txt

6.1.4 Cross validation

The obvious method for estimating the reliability of a classification model is to divide the data sets into the training sets and the testing sets, using the training set to build the model and examine its performance on the unseen testing sets. This is satisfying when there are plenty of data. In more common circumstance, where there are less data sets two problems appear: First, in order to get a reasonably accurate fix on error rate, the testing set must be large, so the classification power of the training set becomes worse. Second, when the available amount of data is moderate, different divisions of the data in training sets and testing sets can produce large variations in error rates on the unseen cases.

A more robust estimation of accuracy on unseen cases can be obtained by cross validation. In this procedure, the available data is divided into N blocks so as to make the same number of each block of cases and class distribution as uniform as possible. N different classification models are then built. In each model, one block is omitted from the training data, and the resulting model is tested on the cases in that omitted block. In this way, each case appears exactly once in the testing set. If N is not too small – ten is a common number – the average error rate over the N unseen testing sets is a good predictor of the error rate of a model built from all the data sets.

In our program, we select N equals to ten, thus it is ten folders cross validation. We only apply the cross validation to the symbolic classification data. The results show that the accuracy rate of cross validation is a little lower than that of the same data sets in training and testing. For the continuous data sets, we do not use the cross validation, as the data sets are enough to test the algorithms.

6.2 The options

The input files (Figure 7.1) of this program are the txt files. Therefore, the file name is filename.txt. The first line of the file is the number and name of the attributes. The second line is the number and name of the classes. The third line and the others are the

data sets; one example occupies one line. For the continuous regression, the second line is "2 small large" or "3 small moderate large" according to how many linguistic labels you want to create when you transfer the continuous classes into the symbolic classes. The format of the testing data file is same as that of the training data file.

As mentioned in the first part of this chapter, the program allows three options to be invoked in the command line. We offer the default values of the options that are used if the options are not invoked. The three options are:

- *S* (Default), the symbolic class; *C*, the continuous class.

- *V* (Default), cross validation (ten folders); *N*, no cross validation.

- *U* (Default), not pruning the tree; *P*, post pruning the tree.

The class Fuzzy_J45.java contains the main method. When executing the program in the command lines, the typical command may be *Fuzzy_J45 C N P*.

6.3 Summary

In this chapter, we present the implementations of the java program that are based on the algorithms we described in chapter four and five. In our implementation, if we classify the continuous regression, the first job is to transfer the continuous class into the symbolic ones, so we embed the SOM_PAK into our java program. For induce the FDT, we utilize some options, such as using the cross validation or not, post pruning the tree or not. For the defuzzification procedure, we have two methods, one is using the classical MOM defuzzification method; the other one is using the NN to do the defuzzification.

In chapter seven, we will present our testing results from the program using the real life data sets, which are the symbolic classification data set and continuous regression data set.

Chapter 7 *Experiment Results*

In chapter six, we implemented our own java program for the fuzzy decision process and fuzzy regression process. We described how to use the java program to implement these fuzzy models for the symbolic classification and continuous regression discriminations. In this chapter, we will generate some experiment results with the real life data sets to validate and evaluate the algorithms and get some conclusions from them.

This chapter is constituted by two parts according to the data sets we implement: The first part is using the software quality prediction data sets whose classes are symbolic classifications. We set up the FDTs and fuzzy decision rules and provide the statistic evaluation models. These data sets are retrieved by the DISCOVER, a commercial software system that can analysis software for the software engineering.

The second part is using the nine data sets from the records of the hydropower station to test fuzzy regression procedure. These data sets are the continuous classes. In the end, we perform the statistic evaluations and the useful discussions.

7.1 Experiment with symbolic classification

One of the big topics in software engineering is how to build software quality estimation models. Suppose we have built such models using ML algorithms, we can predict which model is good, and then we can use such conclusions to evaluate our software designs directly. Using the rule-based system, we can setup the prediction model and predict the quality of the new software with the explicit rules. Now, the problem is how we can get the precise prediction model, and we need the powerful tools to help the software engineers to setup such models. Comparing with other ML methods, FDT can simultaneously fire several rules and allow the simultaneous validation of all the rules [SBL01]. We choose our modified FDT as the right tool to setup the prediction models.

First, we use some tools (such as DISCOVER) to get the software quality metrics. It is quite a task to analysis these metrics, to get the useful formula (rules), and to give the suitable prediction to the software quality. Therefore, we need more precise method to generate the metrics.

We have parsed five software systems using the DISCOVER. For each system, we get some metrics of the software quality prediction. Another data set is Iris data which is the typical data set of ML. We get it from the ftp site of Irvine University [WebIri].

7.1.1 DISCOVER

DISCOVER is a development information system consisting of the integrated sets of applications and tools. It analyzes and parses the source code and creates a database

(Information Model) that captures the interrelationships among all entities in the code base. This Information Model provides critical information for both management and the development team of software engineering. It can also organize software by entity type; query software elements; and generate software graphics;

The results from the DISCOVER are in a detailed view and high-level architectural perspective of the entire application. Specifications, documentation, tests, and test plans are also captured as a part of the central data repository. So it is the best tool for us to retrieve the software quality prediction metrics.

7.1.2 Six symbolic classes data sets

As you know, we use FDT to classify the symbolic classifications. The data format is in Figure 7.1. The attributes are the software metrics, and the classes are in the last column, in which *1* represents stable, and *0* represents unstable. All data sets are got from DISCOVER. The meanings of the metrics are presented in Figure 7.2.

```
6      OCAEC   NOP     DIT     NAM     NAA     DAM
2     1    0
0       0       1       3       1       0       1
0       0       2       11      5       0       1
1       0       1       10      8       0.87    0
0       0       2       34      11      1       0
0       1       2       22      3       1       0
0       0       1       19      8       1       1
0       0       1       4       3       1       1
0       0       1       3       4       0       1
0       0       1       4       2       0       1
0       0       1       3       2       1       1
0       0       1       6       4       1       0
0       0       1       10      16      0       1
0       0       1       5       1       1       1
0       0       1       12      11      0       1
0       1       1       9       3       1       0
0       0       1       4       2       0       1
0       0       2       43      21      0.95    0
0       0       2       3       2       1       0
0       0       1       8       1       0       0
0       0       3       5       1       1       0
0       0       1       32      5       0.8     0
0       0       2       40      7       0.85    0
0       0       1       2       0       0       0
0       0       3       15      4       0.75    0
```

Figure 7.1 Sample data format of software metrics

OCAEC: Others class-attribute export coupling.
NOP: Number of parents.
DIT: Depth of inheritance tree.
NAM: Number of available methods.
NAA: Number of available attribute.
DAM: Data access metrics.
COH: Cohesion.
LCOMB: Lack of cohesion in methods.
COM: Cohesion metric.
COMI: Cohesion metric inverse.
OCMAIC: Other class method attribute import coupling.
CUBF: Number of classes that are used by a membership function of class.
CUB: Number of classes that are used by a class.
CUSOMAEC: Other class method attribute export coupling.

Figure 7.2 The definitions of the metrics

The data are from the current using software systems, which are Donne, Beans, Jetty, Free, and Major_version. All the classes are symbolic variables and the binary classifications except the Iris data. For details, please look at the Table 7.1.

Name	Iris	Donne	Beans	Jetty	Free	Major_version
Data size	150	486	391	229	50	2283
Attribute 1	sepallength	OCAEC	OCAEC	OCAEC	OCAEC	COH
Attribute 2	Sepalwidth	NOP	NOP	NOP	NOP	LCOMB
Attribute 3	petallength	DIT	DIT	DIT	DIT	COM
Attribute 4	Petalwidth	NAM	NAM	NAM	NAM	COMI
Attribute 5		NAA	NAA	NAA	NAA	OCMAIC
Attribute 6		DAM	DAM	DAM	DAM	CUBF
Attribute 7						CUB
Attribute 8						CUSOMAEC
Attribute 9						NOC
Attribute 10						NOP
Class 1	Iris-setosa	0	0	0	0	0
Class 2	Iris-versicolor	1	1	1	1	1
Class 3	Iris-virginica					

Table 7.1 Data set in details

After implementing the program, we get the FDT (Figure 7.3) and the fuzzy decision rules (Figure 7.4).

```
DAM <= 0.0: 1 (75/5)
DAM > 0.0
|  DIT <= 0.0
|  |  OCAEC <= 0.0
|  |  |  NAA <= 0.0: !1 (10/5)
|  |  |  NAA > 0.0: 1 (55/2)
|  |  OCAEC > 0.0: 1 (140/4)
|  DIT > 0.0
|  |  NOP <= 5.0: !1 (42/2)
|  |  NOP > 5.0
|  |  |  NAA <= 5.0: 1 (118/8)
|  |  |  NAA > 3.0
|  |  |  |  |  NOP <= 8.0: 1 (29/9)
|  |  |  |  |  NOP > 7.0
|  |  |  |  |  |  NOP <= 8.0: 1 (16/5)
|  |  |  |  |  |  NOP > 8.0
|  |  |  |  |  |  |  NOP <= 9.0: !1 (28/3)
|  |  |  |  |  |  |  NOP > 8.0
|  |  |  |  |  |  |  |  DAM <= 0.01: 1 (21/1)
|  |  |  |  |  |  |  |  DAM > 0.01: 1 (84/3)
```

Figure 7.3 Sample tree

```
Rule 1:

DIT <= 0.0 AND
OCAEC <= 0.0: 1 (45/2)

Rule 2:

DIT <= 0.0 AND
OCAEC > 0.0 AND
DAM <= 0.5 AND
NAA <= 1.0 AND
DAM <= 0.0: 1 (92/4)

Rule 3:

DIT <= 0.0 AND
OCAEC > 0.0 AND
DAM <= 0.5 AND
NAA <= 1.0 AND
DAM > 0.0 AND
NAA <= 0.0: 1 (51/2)

Rule 4:

DIT <= 0.0 AND
OCAEC > 0.0 AND
DAM <= 0.5 AND
NAA <= 1.0 AND
DAM > 0.0 AND
NAA > 0.0: 0 (40/1)
```

Figure 7.4 Sample rules

These fuzzy decision rules are well explained. If we express them in *if ... then* format, such as for rule 1:

If *DIT* is less than or equal to small (=*0*) and *OCAEC* is less than or equal to small (=*0*) then class is stable (=*1*).

7.1.3 An example: the analysis of Iris data

Iris data set, which dates back to seminal work by the eminent statistician Fisher in the mid-1930s, is the typical test data for ML. It contains fifty examples each of the three types of plant: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. One class is linearly separable from the other two; the latter two are not linearly separable from each other. There are four attributes: sepal length, sepal width, petal length, and petal width (all measured in continuous variables). Iris data set is excerpted in Table 7.2.

	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
3	4.7	3.2	1.3	0.2	Iris setosa
4	4.6	3.1	1.5	0.2	Iris setosa
5	5.0	3.6	1.4	0.2	Iris setosa
...					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
53	6.9	3.1	4.9	1.5	Iris versicolor
54	5.5	2.3	4.0	1.3	Iris versicolor
55	6.5	2.8	4.6	1.5	Iris versicolor
...					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
103	7.1	3.0	5.9	2.1	Iris virginica
104	6.3	2.9	5.6	1.8	Iris virginica
105	6.5	3.0	5.8	2.2	Iris virginica
...					

Table 7.2 Iris data

We test our algorithm to the Iris data set and compare it with the algorithm C4.5. We first use the training data set as the testing data set, and then, use ten folders cross validation. We use the two defuzzification methods: both the MOM method and the NN method. We notice that the performances of cross validation are little decreased than using the training data set as the testing data set.

In our algorithm, we have the correspondent tree for each class. So in Iris data, as they have three classes, we have three trees. We present them in Figure 7.5, Figure 7.6, and Figure 7.7.

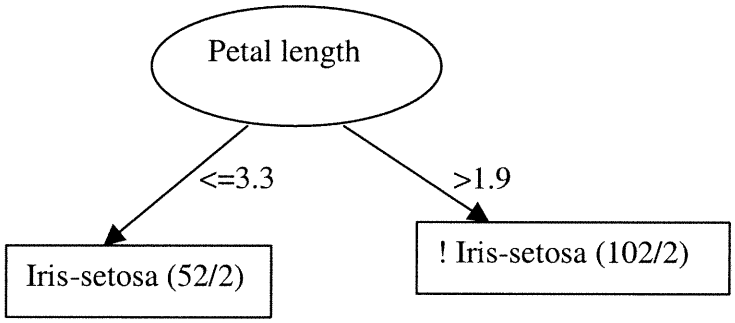

Figure 7.5 Iris-setosa tree

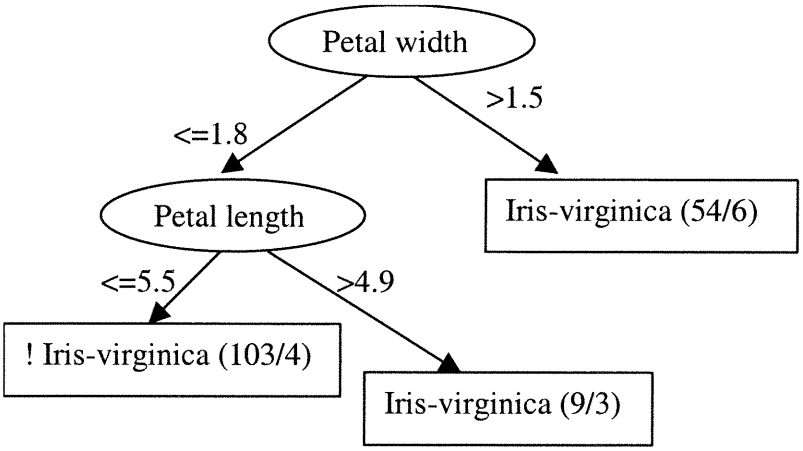

Figure 7.6 Iris-virginica tree

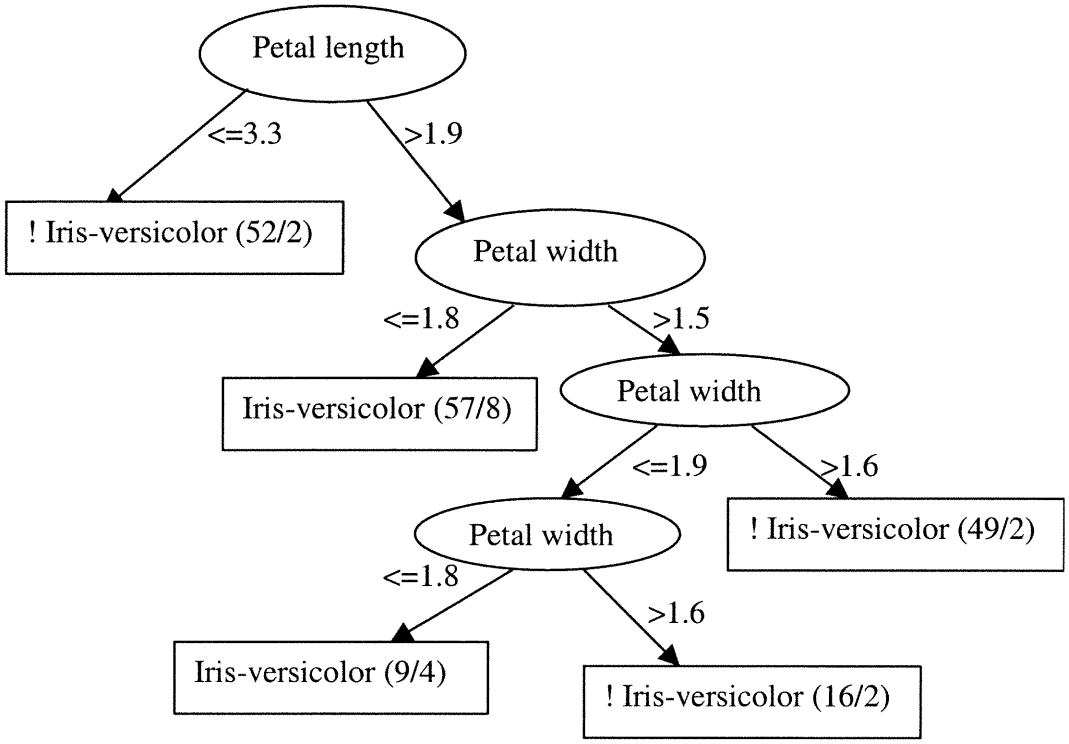

Figure 7.7 Iris-versicolor tree

In FDT, the branch is divided by the fuzzy partition such as "small" or "large". Notice that the values to split the left branch of the tree are always larger or equal to those of the right. This is the main difference between the classical DT and the FDT. The splitting branches show in Figure 7.8.

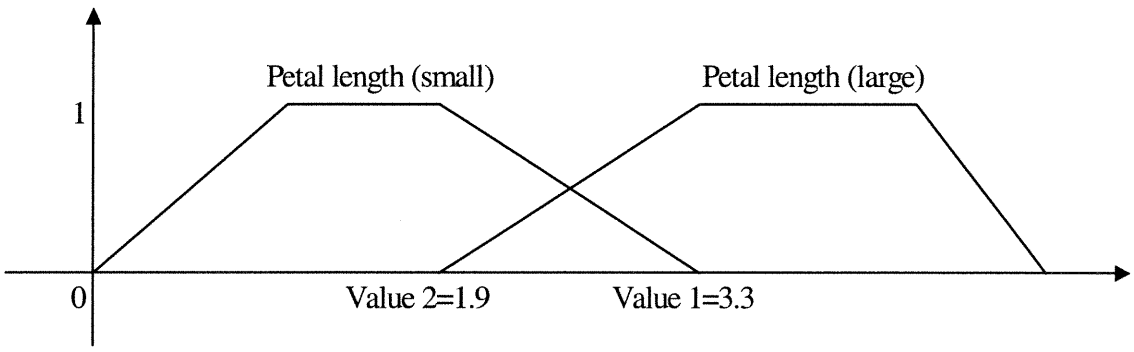

Figure 7.8 Fuzzy values and fuzzy partitions

From these FDTs, we get fuzzy decision rules of the Iris data (Figure 7.9). It is easy to transfer the fuzzy value into the fuzzy label, here we omit this transformation. At the end of each rule and the leaves of each tree are the total number of the examples being fired and the number of misclassification examples.

```
Fuzzy decision rules:
Rule 1:
If Petal length <= 3.3 then Iris-setosa [52/2];

Rule 2:
If Petal length > 1.9 then ! Iris-setosa [102/2];

Rule 3:
If Petal length <= 3.3 then  Iris-versicolor [52/2];

Rule 4:
If Petal length > 1.9 AND Petal width <= 1.8 then Iris-versicolor [57/8];

Rule 5:
If Petal length > 1.9 AND Petal width > 1.5 AND Petal width <= 1.9 AND Petal width <= 1.8
then Iris-versicolor [9/4];

Rule 6:
If Petal length > 1.9 AND Petal width > 1.5 AND Petal width <= 1.9 AND Petal width > 1.6 then
! Iris-versicolor [16/2];

Rule 7:
If Petal length > 1.9 AND Petal width > 1.5 AND Petal width > 1.6 then ! Iris-versicolor [49/2];

Rule 8:
If Petal width <= 1.8 AND Petal length <= 5.5 then ! Iris-virginica [103/4];

Rule 9:
If Petal width <= 1.8 AND > 4.9 then Iris-virginica [9/3];
```

Figure 7.9 Fuzzy decision rules

For inducting the Iris data, we get the accuracy rate of the prediction (Table 7.3). It shows that using the NN defuzzification process and the training data as the testing data, the accuracy rate is higher than that of the C4.5. The FDT is very efficient and robust, and it always gives us the shorter trees than that of C4.5. We use the FDT to classify the Iris data. The results are very prompting: The accuracy rate is ninety four percent. We view the FDT as a way to heuristically find parameters of fuzzy models by processing training data with a learning algorithm. When we do the cross validation and the MOM defuzzification process, the accuracy rate is equal to the C4.5.

Iris	MOM	NN	C4.5
Training Data	95.3	**99.3**	98.0
Cross validation	**95.3**	93.3	95.3

Table 7.3 The comparison results of Iris data set

Therefore, we draw a conclusion that: For some examples, FDTs are more robust than the C4.5 algorithm [SBLE00]. Now we just compare the accuracy rate of the prediction. The more precise statistic evaluation methods will be described in the next section.

7.1.4 Evaluations for the symbolic classification prediction

Although the accuracy rate is important in the evaluations of the models, it does not take into account the cost of making incorrect classifications. Optimizing accuracy rate without considering the cost of the errors often cause a bias. So we need formal measures that comprise objective set of standards. We calculate the confusion matrix from the predicted results. The three outputs of the confusion matrix can thoroughly evaluate the results of the DTs. They are *accuracy*, *correctness*, and *completeness*.

According to [WF99] and [WebKos], suppose we have two-class case with class label *yes* and *no*. we present the five different possible outcomes of a single data set in one table (Table 7.4).

Desired class/predicted class	Yes	No	Completeness
Yes	True positive (a)	False negative (b)	a/(a+b)
No	False positive (c)	True negative (d)	d/(c+d)
Correctness	a/(a+c)	d/(b+d)	Accuracy

Table 7.4 Different outcomes of a two-class prediction evaluation

In this table, true positive and true negative are correct classifications. A false positive is when the outcome is incorrectly predicted as *yes* (or *positive*), it is in fact *no* (or *negative*). A false negative is when the outcome is incorrectly predicted as *no* (or *negative*), it is in fact *yes* (or *positive*). So we get the confusion matrixes for analyzing the results that showing the predicted and desired classifications. A confusion matrix is of size $L \times L$, where L is the number of different label values. For Iris data, L is three, the others, L is two. We have standard evaluations as below:

- *Accuracy* is defined as the percentage of components that were predicted as belonging to certain classification group and actually did belong to that classification group. Accuracy = (a+d)/(a+b+c+d).

- *Correctness* is defined as the percentage of components that were predicted as belonging to certain classification group and actually did belong to that classification group.

- *Completeness* is defined as the percentage of those components that belonged to certain classification group and were identified by the model.

The three standards are all wanted to maximize. For accuracy, it measures how correct the model is. For correctness, if correctness is low, the model identifies more components as being non-faulty, but in fact, they are really faulty. And for completeness, if completeness is low, then more components that were likely to be faulty will not be identified.

Another measure that can be used to evaluate the overall appropriateness of the model is the goodness-of-fit of data. This measure can be obtained via a Chi-square test. This test evaluates whether or not the expected cell frequencies under the respective model are significantly different from the observed cell frequencies. The higher the value of Chi-square test, the better the fit of the data and the model is. The corresponding p-value points out to the statistical significance of the test. The statistical significance of the

results is an estimated measure of the degree to which it is "true", in the sense of "representative of the population". The *p*-value represents a decreasing index of the reliability of the results. The smaller the *p*-value, the more significant the test is and the more you can believe that the observed relation among the variables in the sample. It is a reliable indicator of the relation among the respective variables in the population. The following six tables present the empirical results of six data sets using the statistic methods to evaluate. They use both the MOM method and the NN method as the defuzzification procedures. The common characters of these tables (Table 7.5-Table 7.10) are:

- High overall accuracy achieved. The average value is 81.0% for fuzzy induction and 81.3% for NN across the six experiments.
- Experiments in Iris data, Free data, and Donne data show that the X-sqr values are high; with the perfect statistical significance (*p*-value is small).

Iris data				
Fuzzy induction	Predict 0	Predict 1	Predict 2	Completeness
Desired 0	50	0	0	100
Desired 1	0	48	2	96
Desired 2	0	5	45	90
Correctness	100	90.6	95.7	95.3
X-sqr	261.3408	p <=	0.001	
Neural induction	Predict 0	Predict 1	Predict 2	Completeness
Desired 0	50	0	0	100
Desired 1	0	49	1	98.0
Desired 2	0	0	50	100
Correctness	100	100	98.0	99.3
X-sqr	294.1176	P <=	0.001	

Table7.5 Iris data confusion matrix

Donne data			
Fuzzy induction	Predict 0	Predict 1	Completeness
Desired 0	0	149	0
Desired 1	0	337	100
Correctness	0	69.3	69.3
X-sqr	0.00004	p <=	1
Neural induction	Predict 0	Predict 1	Completeness
Desired 0	16	133	10.7
Desired 1	23	314	93.2
Correctness	41.0	70.2	67.9
X-sqr	2.1437	p <=	0.20

Table 7.6 Donne data confusion matrix

Jetty data			
Fuzzy induction	Predict 0	Predict 1	Completeness
Desired 0	0	56	0
Desired 1	0	173	100
Correctness	0	75.5	75.5
X-sqr	0.00007	P <=	1
Neural induction	Predict 0	Predict 1	Completeness
Desired 0	0	56	0
Desired 1	0	173	100
Correctness	0	75.5	75.5
X-sqr	0.00007	P <=	1

Table 7.7 Jetty data confusion matrix

Beans data			
Fuzzy induction	Predict 0	Predict 1	Completeness
Desired 0	0	96	0
Desired 1	0	295	100
Correctness	0	75.5	75.5
X-sqr	0.00007	P <=	1
Neural induction	Predict 0	Predict 1	Completeness
Desired 0	0	96	0
Desired 1	0	295	100
Correctness	0	75.5	75.5
X-sqr	0.00007	P <=	1

Table 7.8 Beans data confusion matrix

Free data			
Fuzzy induction	Predict 0	Predict 1	Completeness
Desired 0	14	9	60.9
Desired 1	3	24	88.9
Correctness	82.4	72.7	76.0
X-sqr	13.7035	p <=	0.001
Neural induction	Predict 0	Predict 1	Completeness
Desired 0	14	9	60.9
Desired 1	3	24	88.9
Correctness	82.4	72.7	76.0
X-sqr	13.7035	p <=	0.001

Table 7.9 Free data confusion matrix

Major_version data			
Fuzzy induction	Predict 0	Predict 1	Completeness
Desired 0	0	140	0
Desired 1	0	2143	100
Correctness	0	93.9	93.9
X-sqr	0.00067	p <=	1
Neural induction	Predict 0	Predict 1	Completeness
Desired 0	0	140	0
Desired 1	0	2143	100
Correctness	0	93.9	93.9
X-sqr	0.00067	p <=	1

Table 7.10 Major_version data confusion matrix

7.2 Experiment with continuous regression

The hydropower network data of this part experiment is coming from HRM. The used algorithm is that we proposed in the chapter five, the fuzzy regression system. In this section, we introduce the problem we try to solve. And then, we present the experiment results, and the statistic evaluations.

7.2.1 Problem description

The data sets are about hydrological information contained in a historical database to improve the forecast of natural contributions flow in the short run of Chute-du-Diable area for the summer-fall period (June 15 at November 30) from 1992 to 1999. The objective of getting the results from the data sets can be summarized as the satisfaction of the following requirements:

- Effective use of water

- Account of future hydrological uncertainty
- Satisfaction of energy need
- Respect of safety constraints.

To reach these goals, a decision-making process of water stock management is used that consists of four steps:

1) Weather hydro measurements and gathering of the data;
2) Data analysis;
3) Weather and hydrological forecasting;
4) Planning.

An essential component of the decision process is the prediction of natural contributions flow. This helps evaluate the ability of the power system to face various contingencies and to propose appropriate remedial actions. In our thesis, we focus on the step two and step three.

The data sets are grouped into nine separate parts. All data sets are composed of 1186 examples, the class values are continuous variables and range from 300 to 1300 (Table 7.11).

Class	AncCD_J1	AncCD_J2	AncCD_J3	AncCD_J4	AncCD_J5	AncCD_J6	AncCD_J7	Vol_3j	Vol_7j
Range	1290	1290	1290	1290	1290	1290	1290	297	551
Attribute 1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1	AncCD_P1
Attribute 2	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j	AncCD_T1j
Attribute 3	Qman_P1	Qman_4h	Qman_4h	Qman_4h	Qman_4h	Qman_4h	Qman_4h	Qman_4h	Qman_4h
Attribute 4	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h	Qman_T4h
Attribute 5	Qman_T12h	Qserp_4h	Qman_T12h	Qman_T12h	Qman_T12h	Qman_T12h	Qman_T12h	Qman_T12h	Qman_T12h
Attribute 6	Qserp_4h	Qserp_T4h	Qserp_4h	Qserp_4h	Qserp_4h	Qserp_4h	Qserp_4h	Qserp_4h	Qserp_4h
Attribute 7	Qserp_T4h	Qserp_T12h	Qserp_T4h	Qserp_T4h	Qserp_T4h	Qserp_T4h	Qserp_T4h	Qserp_T4h	Qserp_T4h
Attribute 8	Qpper_4h	Qpper_4h	Qpper_4h	Qpper_4h	Qserp_T12h	Qserp_T12h	Qserp_T12h	Qpper_4h	Qpper_4h
Attribute 9	Qpper_T4h	Qpper_T4h	Qpper_T4h	Qpper_T4h	Qpper_4h	Qpper_4h	Qpper_4h	PbvCD_P2	Qpper_T4h
Attribute 10	PbvCD_P4	PbvCD_P3	PbvCD_P2	PbvCD_P2	Qpper_T4h	Qpper_T4h	Qpper_T4h	PstCDP_P2	PbvCD_P2
Attribute 11	PstCD_P2	PstCD_P1	PbvCD_P1	PbvCD_P1	PbvCD_P3	PbvCD_P2	PbvCD_P1	PstCDP_P1	PbvCD_P1
Attribute 12	PstCDP_P3	PstCDP_P2	PstCDP_P2	PstCDP_P2	PbvCD_P2	PbvCD_P1	Pprev_J1	PstMisbi2_P1	Pprev_J1
Attribute 13	PstCDP_P2	PstCDP_P1	PstCDP_P1	PstCDP_P1	PbvCD_P1	Pprev_J1	Pprev_J2	PstManE_P3	Pprev_J2
Attribute 14	PstMisbi2_P1	PstMisbi2_P1	PstMisbi2_P1	Pprev_J1	Pprev_J1	Pprev_J2	Pprev_J3	Pprev_J1	Pprev_J3
Attribute 15	PstManE_P2	PstManE_P1	Pprev_J1	Pprev_J2	Pprev_J2	Pprev_J3	Pprev_J4	Pprev_J2	Pprev_J4
Attribute 16	Pprev_J1	Pprev_J1	Pprev_J2	Pprev_J3	Pprev_J3	Pprev_J4	Pprev_J5	Pprev_J3	Pprev_J5
Attribute 17		Pprev_J2	Pprev_J3	Pprev_J4	Pprev_J4	Pprev_J5	Pprev_J6		Pprev_J6
Attribute 18					Pprev_J5	Pprev_J6	Pprev_J7		Pprev_J6

Table 7.11 Hydropower station data sets

7.2.2 Continuous regression prediction

Through the fuzzy regression induction, for each data set, we use SOM algorithm to transfer the continuous classes into two symbolic classes. Then we generate two FDTs and fuzzy decision rules. By using the min-max algorithm, we get the input data of the NN for defuzzification procedure. There are three layers in back propagation NN: The nodes of the input layer are the results of the leaves number of two trees, which are labeled with the class. The hidden layer has the same nodes as the input nodes. The output layer has one node that gives the crisp value we predicted. The sample input data show in Table 12.

Examples	Tree (small)			Tree (large)			Desired value
	Leaf (small)	Leaf (small)	Leaf (small)	Leaf (large)	Leaf (large)	Leaf (large)	
Ex 1	0.8	...	0.5	0	...	0.1	84
Ex 2	0.4	...	0.7	0.2	...	0.2	498
...	...	...	...	...	...	...	...
Ex 1186	0.2	...	0	0.8	...	0.7	713

Table 7.12 Sample input data of the NN

We implement the algorithm described in chapter five for the continuous regression. We have three different tests in different number of classes and defuzzification methods (Table 7.13).

Test	Number of classes	Defuzzification method
1	3	MOM
2	2	MOM
3	2	NN

Table 7.13 Tests on the data

For continuous classes transferring to symbolic classes, test 1, we use three symbolic classifications: *small*, *moderate*, *large*. The defuzzification method is the MOM. Test 2, we use two symbolic classifications: *small* and *large*. The defuzzification method is the same as the first one. Test 3, we use the second classification transformation and the defuzzification method is the NN. In all these tests, we use the training data sets as the testing data sets. We select ten percent of the data range as the threshold. That means, if the result of the desired value minus the predict value is smaller than the threshold, it indicates that the prediction is successful.

%	CD_J1	CD_J2	CD_J3	CD_J4	CD_J5	CD_J6	CD_J7	Vol_3j	Vol_7j
1	77.1	77.9	79.3	81.2	79.6	77.2	63.2	63.8	63.1
2	55.1	40.8	31.2	54.3	47.2	52.9	73.1	58.5	23.7
3	82.5	77.6	80.2	78.1	76.8	78	76.1	76.5	58.8

Table 7.14 Accuracy rate of three tests

From Table 7.14, we conclude that for the classical defuzzification method, the more fuzzy partition number, the more accuracy the results are. In addition, the NN defuzzification procedure is better than that of MOM method.

Figure 7.10 is the graphic comparison accuracy rates with test 1 and test 3. It shows that NN method is better than MOM method without much deference. In next section, we use the statistic method to evaluate the results of test 1 and test 3, and then we can further testify our conclusions.

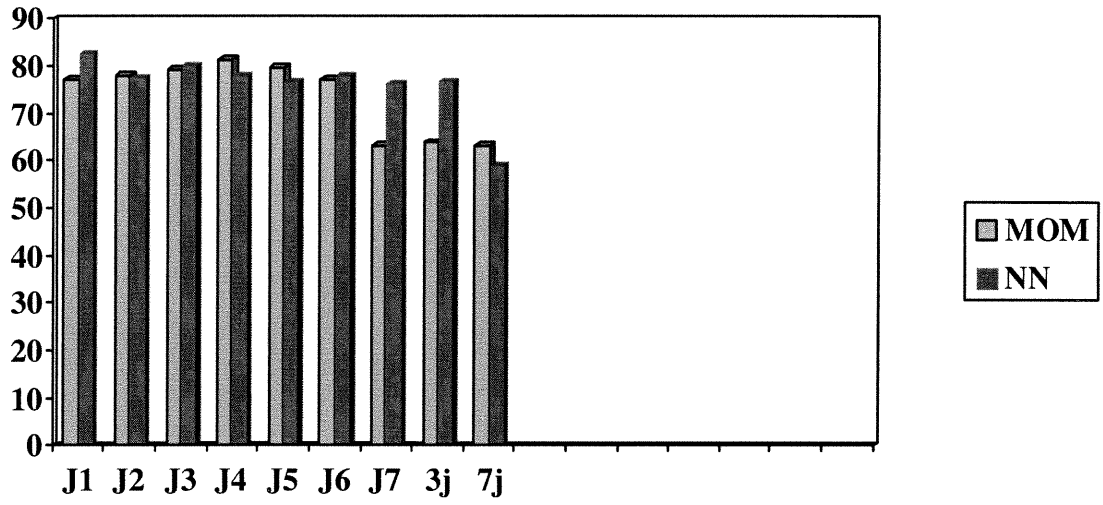

Figure 7.10 Accuracy rates of two defuzzification methods

7.2.3 Statistic evaluation method

In the continuous variable prediction situation, the basic quality measures offered by the error rate or accuracy rate are not enough: Errors are not simply present or absent, as the data sizes may be different.

Several methods, which have more or less relations with the statistic theory, can be used to evaluate the success of the continuous variable prediction. We choose *mean*, *median* and *standard deviation* (*Std.*) three methods (Table 7.15) from them. All these three methods want the small values.

- Mean is also called the mean absolute error. It is the principal and commonly used measure. It is obtained by adding the value of each quantity and divided by the number of quantities without asking account of their sign. It has the feature that all sizes of errors are treated evenly according to their magnitude.

- Median is the value of the variable for the middle member of the population or sample; half of the population would have value above and the other half below.

- Standard deviation is the statistic method that takes into account the whole sample which may therefore be distorted by rogue data.

Methods	Formula
Mean	$\sum x / n$
Median	The middle value.
Std.	$\sqrt{(\sum X^2 - \bar{x}\sum X)/(n-1)}$

Table 7.15 Statistic measures for continuous data prediction

The evaluation results present in Table 7.16. Although both results are satisfying, the result values of NN are smaller than that of the MOM. Therefore, the NN method is better than the MOM method.

MOM	J1	J2	J3	J4	J5	J6	J7	3j	7j
Mean	21.9844	20.4363	20.6642	20.9531	21.3375	20.5900	54.4037	1.3802	18.8675
Median	0	0	0	0	0	0	0	0	0
Std.	80.1131	79.1718	79.8761	81.1224	81.0345	79.3304	118.0583	7.5154	46.9306
NN	J1	J2	J3	J4	J5	J6	J7	3j	7j
Mean	12.1085	15.9230	13.1314	15.4255	11.3932	16.5626	17.9033	0.2803	17.1069
Median	0	0	0	0	0	0	0	0	0
Std.	54.6577	55.8703	46.9230	55.4930	47.9052	64.5570	68.2035	4.6163	42.4060

Table 7.16 Results of statistic evaluation for hydropower station data

It is easy to explain these evaluation results from the practical point of view. For example, the 3j model is the prediction of the coming three days variations; the 7j model is the prediction of the coming seven days variations. In Table 7.15, the values in the 3j column are far smaller than those of the 7j column. That means to predict in three days is easier than to predict in seven days.

7.3 Lessons learned

From the implementation of the fuzzy classification and regression system using the real life data, we can see that some results are satisfying and some are not as good as expected. The following parts are the discussions of these problems.

7.3.1 Symbolic classification

There is mainly a typical error in some results (Beans data, Major_version data, and Jetty data) of the experiments. The class "*0*" was misclassified. The reason is when the number of class "*0*" is small and they are sparsely distributed in the data set, then the FDT has difficulty to classify them, and considering them as a certain part in the fuzzy partition.

Basically speaking, there are two reasons to form such misclassification. First we smooth a word using the dilatation and erosion operators. After using these two operators, the word is smoothed and it is generated into a new word, in which the same class labels are close to each other. But if a class is sparsely distributed in the data sets, there are always one or two class characters among other classes. For example, in Figure 7.11, there are two classes "+", and "-". Before we use the mathematical morphology, the class is distributed like the first figure. After the transformation (the second figure), the classes are generated into three groups; those are sequence "+", sequence uncertain, and sequence "-"; both the sequence "+", and sequence "-" have other classes.

The second reason is there is a limit when the FDT splits. This limit is that the generated length of the word's longest definite part should be longer than two times of the recursion time. If not, the tree stops splitting. Therefore, in the data set, it remains the different classes.

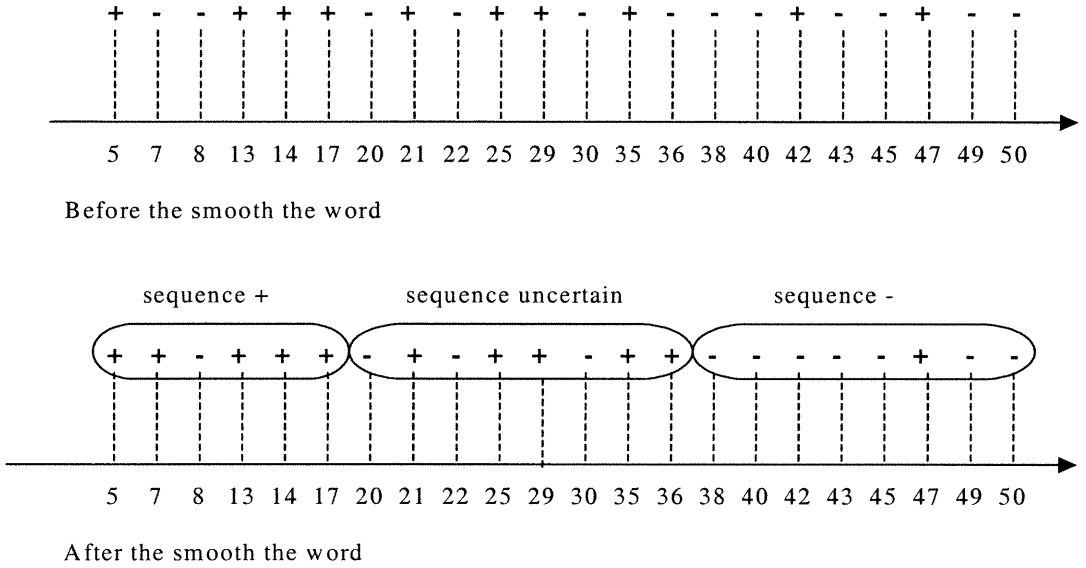

Figure 7.11 Fuzzy partition problem

7.3.2 Continuous regression

Suppose we have two classifications, class "*0*" and class "*1*". Theoretically speaking, the generated class "*!0*" equals to class "*1*", and class "*!1*" equals to class "*0*", thus one tree is enough. We can only use the class "*0*" and "*!0*" or class "*1*" or "*!1*". But if we have more than two classes, we should use all the trees. The same thing is for the symbolic classification.

Another issue is that we use the threshold value to judge the successful prediction. This is the special character of the continuous regression predictions. We should sacrifice the accuracy, if we want to make the program efficient and the results easy to explain with the rules. In neuro-fuzzy techniques, we must deal with this trade-off between precision and readability [BB97].

7.4 Summary

In this chapter, we have presented the experiment results of our data sets both for the symbolic classes and continuous classes. We give not only the accuracy rate of each prediction, but also the statistic evaluations. The results are satisfying for both evaluations. For the symbolic classification, we do training data set as the testing data set and ten folders cross validation separately. The results are similar, but the accurate rate of the cross validation is a little decreased. The part of rules for software quality prediction we generated were used in [SBC01]. For the continuous regressions, we compare the NN and MOM defuzzification methods. We find that the NN method for defuzzification is better than the classical MOM defuzzification methods. And also these results are generated in [BML02].

In chapter eight, we will give conclusions of the whole thesis and the lessons we learned from the experiments and the discussions of the future work.

Chapter 8 *Conclusions and Future Work*

In chapter seven, we present the results of implementation of our rule-based system to both the symbolic classification and continuous regression. We discuss and evaluate the results and draw some conclusions. In this chapter, we will present the conclusions of the thesis and the future work.

8.1 Conclusions

In this thesis, we enhance the Marsala's FDTs for some modifications. We change the shape of the fuzzy partition, and add the enhanced pessimistic pruning process in building the tree. We introduce a neuro-fuzzy approach to deal with the continuous regressions. We implement the two NN algorithms, SOM and back propagation, into these fuzzy regression procedures.

The conclusions of this thesis are:

- From our experiments, there are two advantages of using the mathematical morphology to do the fuzzy partitions. First, this method is better than the normal clustering method, as it associates with the classes. Also sometimes it is better than the fuzzy partition provided by the domain experts as they have the chances to make mistakes. Second, this method can be implemented automatically, thus avoid the interaction with the human.

- The modification of Marsala's FDT algorithm, such as changing the recursion time, the splitting criterion and the fuzzy membership function, and adding the pessimistic pruning procedure, can enhance the FDT's inductive and prediction abilities.

- When we use the classical defuzzification, the more numbers of the fuzzy partition of the FDTs we hold, the more accurate the results are. But if the fuzzy partition numbers are too many, the fuzzification and the defuzzification procedure will be too complicated.

- For the symbolic classification, the min-max algorithm and the vote method to defuzzy the FDTs are good choices. They can be compared with the other methods when we classify the new cases.

- For the continuous regression, the NN method as the defuzzification procedure is better than the classical defuzzification methods. As the NN uses all fuzzy set values which are "fired" in the FDT, while the classical defuzzification method just uses the representative fuzzy set values. The results suggest that this neuro-fuzzy approach we proposed is comparable with the technology which is used in ML.

- Some of the software quality prediction data sets are not predicted well because of the small number of the class and it sparsely distributes in the data sets. For the continuous regression prediction, there is a trade-off between the accuracy and the explicitness.

8.2 Future work

Future work will be focused on implementing more data to test and enhance the algorithm. Using the C-means clustering to transfer the continuous classes in the symbolic ones is also an interesting work. Now we are using the standard back propagation algorithm for defuzzifying the FDTs. The major drawback of it is when a large learning rate occurs. It will prevent the algorithm from descending to the desired optimal solution. Therefore, the performance of convergence can be slow. If we use some modified algorithms, such as resilient propagation algorithm, the performance may be increased.

As the boosting algorithm has been testified effective to the classical DTs, we consider using the boosting algorithms for improving the performance of FDTs. The general procedure of boosting the FDTs can be straight forward to adapt the FDTs to the process of weighted data vectors. The outputs from the FDTs could be combined in a linear combination of the class probabilities. For each tree, it could be taken to give an overall set of probabilities. In this case the weights for each tree would be determined according to the expected performance of the trees. Alternatively, we can also use fuzzy sets to partition the classification probability values obtained from each DT and use these to learn a higher level DT on probability space which could then be used for combination.

8.3 Closing words

Enhancing and evolving the rule-based system is not an easy job. As one of the effective rule-based system, the FDT has many aspects to be thought over. Although there are many modifications about it, it still needs to enhance the performance. For the fuzzy regression procedure, we should enhance the prediction accuracy rate of it. Apparently, there is a trade-off between the explicitness and the accuracy: If we want to get better predicted results, we can use the RT or more complicated algorithms to classify the cases. But it is hard to generate the rules from them. If we want to get the explicit rules, the accuracy rate may be affected. Our approach is a satisfying approach between this trade-off. Many deep thinking should be executed. Also the new approach may be needed.

Here, I personally thank HRM for its financial support and my supervisor Professor Sahraoui for his academic support. The combination of these two supports is the basis of this thesis.

Bibliography

[Ara94] R. Araya. Induction of Decision Trees when Examples are Described with Noisy Measurements and with Fuzzy Class Membership. *INRIA seminar, CLOREC Project,* 1994.

[BB97] H. Bersini, G. Bontempi. Now comes the time to defuzzify neuro-fuzzy models. *Fuzzy Sets and Systems 90,* pp. 161-170, 1997.

[BBM97] S. Bothorel, B. Bouchon-Meunier, S. Muller. A Fuzzy Logic Based Approach for Seminological Analysis of Microcalcifications in Mammographic Images. *International Journal of Intelligent Systems 12,* 11-12, pp. 819-848, 1997.

[BGS93] M. Botta, A, Giordana, L. Saitta. Learning Fuzzy Concept Definitions. *In Proceedings of the 2nd IEEE International Conference on Fuzzy Systems,* San Francisco, pp. 18-22, 1993.

[BM99] B. Bouchon-Meunier, C. Marsala. Learning Fuzzy Decision Rules. *In Fuzzy Sets in Approximate Reasoning and Information Systems,* J. Bezdek, D. Dubois and H. Prade eds, Kluwer Academic Pub., Handbooks on Fuzzy Sets Series, chap. 4, pp. 279-304, 1999.

[BML02] M. Boukadoum, G. Mai, H. Lounis, H. A. Sahraoui, V. Siveton. A Comparison Between Using a Neural Network and a Fuzzy Regression System to Predict the Values of Hydropower System Variables. *Technical report*, 2002.

[Boy95] X. Boyen. Design of the Fuzzy Logic-based Decision Trees Applied to Power System Transient Stability Assessment. *Memoire de fin d'etudes,* University of Liege, Belgium, 1995.

[BRB96] B. Bouchon-Meunier, M. Rifqi, S. Bothorel. Towards general measures of comparison of objects. *Fuzzy Sets and Systems 84(2),* pp. 143-153, December 1996.

[BW96] X. Boyen, L. Wehenkel. Automatic Induction of Continuous Decision Trees. *In Proceedings of the 6th Int. Conf. IPMU.,* Granada, Spain, pp. 419-424, 1996.

[CJT99] M. Ching-Hung, L. Jan-Fu, H. Tzung-Pei, T. Shian-Shyong. A fuzzy inductive learning strategy for modular rules. *Fuzzy Sets and Systems 103,* pp. 91-105, 1999.

[CT86] M. R. Civanlar, H. J. Trussell. Constructing membership functions using statistical data, *Fuzzy Sets and Systems 18,* pp. 1-14, 1986.

[DT72] A. De Luca, S. Termini. A Definition of Non-probabilistic Entropy in the Settings of Fuzzy Sets Theory. *Information and Control 20,* pp. 301-312, 1972.

[Has98] Richard E. Haskell. Neuro-Fuzzy Classification and Regression Trees. *Proc. Third International Conference on Applications of Fuzzy Systems and Soft Computing,* Wiesbaden, Germany, October 5-7, 1998.

[Has99] Richard E. Haskell. Market Sector Selection Using Fuzzy Regression Trees. *Proc. International ICSC Congress on Computational Intelligence -- Methods and Applications; International Symposium on Soft Computing in Financial Markets,* Rochester Institute of Technology, NY, USA, June 22-25, 1999.

[HRG96] S. K. Halgamuge, T. A. Runkler, M. Glesner. On the Neural Defuzzification Methods. *In Proc. 5th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96),* New Orleans, Louisiana, USA, pp. 463-469, IEEE, 1996. ISBN 0-7803-3645-3, Sep 8-11, 1996.

[Jan94] C. Z. Janikow. Fuzzy Decision Trees: FIDMV. *In Proceedings of the Joint Conference on Information Science JCIS'94,* Pinehurst, USA, pp. 232-235. 1994.

[JJ01] B. P. Joseph, B. Jennifer. Constructing intelligent agents using JAVA. *2nd ed. New York; Toronto,* Wiley copyright, 2001.

[KKJ99] L. Koen-Myung, L. Kyung-Mi, L. Jee-Hyong, H. Lee-Kwang. A Fuzzy Decision Tree Induction Method Fuzzy Data. *Fuzzy Systems Conference Proceedings,* FUZZ-IEEE '99. 1999 IEEE International Volume:1, pp. 16-21, 1999.

[Koh95] T. Kohonen. Self-Organizing Maps. *Springer-Verlag Series in Information Sciences,* Vol 30, Heidelberg, 1995.

[Kos97] B. Kosko. Fuzzy Engineering. *Prentice-Hall Inc.,* 1997.

[LS98] G. F. Luger, W. A. Stubblefield. Artificial Intelligence Structures and Strategies for Complex Problem Solving. *Addison Wesley Longman,* Inc., 1998.

[Mar95] C. Marsala. Fuzzy Partition Inference over a Set of Numerical Values. *Technical report,* 1995.

[Mar98] C. Marsala. Apprentissage inductif en présence de données imprécises: construction et utilisation d'arbres de décision flous. *THESE de DOCTORAT,* Université Pierre et marie Curie, Paris, France, Rapport LIP6 no 1998/014, janvier 1998.

[Mar00] C. Marsala. Fuzzy Partitioning Methods. *Technical report,* 2000.

[MB96] C. Marsala, and B. Bouchon-Meunier. Fuzzy Partitioning Using Mathematical Morphology in a Learning Scheme. *In Proc. Of the 5th Int. Conf. On Fuzzy Systems,* Fuzz-IEEE'96, New Orleans, pp. 1512-1517, Sept. 1996.

[MRT98] C. Marsala, M. Ramdani, M. Tollabi, D. Zackarya. Recognition of Odors: a Fuzzy Decision Tree Approach. *In Proceeding of the 7th Int. Conf. IPMU.,* Paris, France, pp. 532-539, 1998.

[MS93] P. E. Maher, D. Saint-Clair. Uncertain Reasoning in an ID3 Machine Learning Framwork. *In Proceedings of the 2nd IEEE Int. Conf. on Fuzzy System,* FUZZ-IEEE, pp. 7-12, 1993.

[NK99] D. Nauck, R. Kruse. Neuro-Fuzzy methods in fuzzy Rule Generation. *In Fuzzy Sets in Approximate Reasoning and Information Systems,* J. Bezdek, D. Dubois and H. Prade eds, Kluwer Academic Pub., Handbooks on Fuzzy Sets Series, chap. 5, pp. 305-332, 1999.

[OLM97] L. Ohno-Machado, R. Lacson, E. Massad. Decision Trees and Fuzzy Logic: A Comparison of Models for the Selection of Measles Vaccination Strategies in Brazil. *Proc AMIA Symp,* pp. 625-629, 2000.

[Qui93] J. R. Quinlan. C4.5: Programs for machine learning. *San Francisco,* Morgan Kaufmann, 1993.

[Ram92] M. Ramdani. Une approche floue pour traiter les valeurs numeriques en apprentissage. *Actes des Journees Francophones d'Apprentissage et d'Explication des Connaisssances,* Dourdan, France, 1992.

[SAT95] T. Shibata, T. Abe, K. Tanie, M. Nose. Motion Planning of a Redundant Manipulator Based on Criteria of Skilled Operators using Fuzzy ID3 and GMDH. *In Proceedings of the 6th IFSA World Congress,* Sao Paulo, Brazil, pp. 613-616, 1995.

[SBC01] H. A. Sahraoui, M. Boukadoum, H. M. Chawiche, G. Mai, M. Serhani. A Fuzzy Logic Framework to Build Rule-Based Quality Prediction Models. *The 26th Annual International Computer Software and Applications Conference (COMPSAC 2002),* Oxford, England, August 2002 (to appear).

[SBL00] H. A. Sahraoui, M. Boukadoum, H. Lounis. Using Fuzzy Threshold Values for Predicting Class Libraries Interface Evolution. *In Proc. of the 4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering,* Lisbon, Portugal, pp. 47-58, 2000.

[SBL01] H. A. Sahraoui, M. Boukadoum, H. Lounis. Building Quality Estimation models with Fuzzy Threshold Values. *L'Objet Vol. 17,* No. 4, Edition Hermès Sciences, 2001.

[SBLF00] H. A. Sahraoui, A. M. Boukadoum. H. Lounis, F. Etheve. Predicting Class Libraries Interface Evolution: an investigation into machine learning approaches. *In Proceedings of the 7th Asia-Pacific Software Engineering Conference (APSEC'00),* pp. 456-464, Singapore, December 2000.

[SSB01] H. Sahraoui, M. A. Serhani, M. Boukadoum. Extending Software Quality Predictive Models Using Domain Knowledge. *In Procedings of 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2001),* 2001.

[TOA79] H. Tanaka, T. Okuda, K. Asai. Fuzzy Information and Decision in Statistical Model. *In Advances in Fuzzy Set Theory and Applications,* M. M. Gupta, R. K. Ragade, R. R. Yager. Eds., North-Holland, pp. 303-320, 1979.

[Tor99] L. F. R. A. Torgo. Inductive Learning of Tree-Based Regression Models. *Ph. D thesis,* Departamento de Ciencia de Computadores Faculdade de Ciencias da Universidade do Porto Setembro de, 1999.

[TS85] T. Takagi, M. Sugeno. Fuzzy Identification of Systema and its Applications to Modeling and Control. *IEEE Trans. Systems, Man and Cybernetics,* 15(1), pp. 116-132, 1985.

[UOH94] M. Umano, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, J. Kinoshita. Fuzzy Decision Trees by Fuzzy ID3 algorithm and Its Application to Diagnosis Systems. *In Proceedings of the 3rd IEEE Int. Conf. On Fuzzy Systems,* Orlando, pp. 2113-2118, 1994.

[VRS99] R. Veliev, A. Rubinov, A. Stranier. Neural Networks for defuzzification of Fuzzy Rules: An Application in Macroeconomic Forecasting. *AI'99,* LNAI 1747, Springer-Verlag, Berlin; Heidelberg, pp. 60-71, 1999.

[Web92] R. Weber. Fuzzy-ID3: a Class of Methods for Automatic Knowledge Acquisition. *In proceedings of the 2nd Int. Conf. On Fuzzy Logic,* Iizuka, pp. 265-268, 1992.

[Web95] K. Weber. Survey on the Defuzzification Element of an Adaptive Neuro-Fuzzy Control System. *In* Proceedings of the ICSC Symposium on Fuzzy Logic, Zürich, 1995.

[WebBor] Borland company website. http://www.borland.com

[WebChi] Web Chi Square Calculator.
http://www.georgetown.edu/cball/webtools/web_chi.html

[WebDev] http://www.dev.com/free/tips/tipview.asp?Content_id=1798

[WebIri] The Iris data set of machine learning. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

[WebKos] R. Koschke. Institut fur Informatik, Universitait Stuttgart. Ph. D thesis. ftp://www.informatik.uni-stuttgart.de/pub/ps/people/koschke/Thesis/Koschke.Thesis.PDF

[WebMcC] C. McConnell. Classification and Regression Trees (CART).
http:// www.cs.cmu.edu/afs/cs.cmu.edu/user/ccm/www/talks/cart.ps

[WebSil] W. Siler. Building Fuzzy Expert Systems.
On line book, http://members.aol.com/wsiler/

[WebSom] SOM Programming Team of the Helsinki University of Technology. SOM_PAK the Self-Organizing Map Program Package User's Guide. http://www.cis.hut.fi/research/som_pak/

[WebSun] Sun company website. http://www.sun.com

[WebSW] Zenon A. Sosnowski, Jarek S. Walijewski. Generating Fuzzy Decision Rules with the Use of Dempster-Shafer Theory. *Online paper.* http://cksr.ac.bialystok.pl/jwal/e-index.html

[WebTor] L. Torgo. Rt4.1 User's Manual. http://www.liacc.up.pt/~ltorgo

[WebVen] http://www.artma.com/jini/cyberspace/CyberspaceConsole.html

[Weh96] L. Wehenkel. On Uncertainty Measures used for Decision Tree Induction. *In Proceeding of the 6th Int. Conf. IPMU.,* Granada, Spain, pp. 413-418, 1996.

[WF99] I. H. Witten, and E. Frank. Data Mining Practical Machine Learning Tools and Techniques with Java Implementations. *Morgan Kaufmann Publishers,* 1999.

[XBG00] W. Xizhao, C. Bin, Q. Guoliang, Y. Feng. On the optimization of fuzzy decision trees. *Fuzzy and Sets and Systems 112,* pp. 117-125, 2000.

[YF01] P. Yonghong, P. A. Flach. Soft Discretization to Enhance the Continuous Decision Tree Induction. *ECML/PKDD01 Workshop IDDM-2001,* Freiburg, September 2001.

[YS95] Y. Yuan, M. Shaw. Induction of fuzzy decision tree. *Fuzzy sets and systems 69,* pp. 125-139, 1995.

[Zad68] L. A. Zadeh. Probability measures of fuzzy events. *Journal Math. Anal. Applic.,* in Fuzzy Sets and Applications: selected papers by L. A. Zadeh. R. R. Yager and S. Ovchinnikov and R. M. Tong and H.T. Nguyen Eds, pp. 45-51, 1968.