Part II

Part II

Chapter 6

Preliminaries and State of the Art: Fuzzy Logic and Fuzzy Decision Trees

This chapter lays out the preliminaries and concepts of fuzzy theory and its applications in the real world. It presents all the elements of fuzzy logic theories necessary for a good understanding of the fuzzy decision tree methods and describes the components of a fuzzy system. It also presents the state of the art and literature in fuzzy decision tree and related techniques.

6.1 Fuzzy Logic and Reasoning

Fuzzy theory deals with the vague and imprecise information present in real life and to cater for these aspects by retaining these properties in a descriptive manner. The information present in real life about a given object or an action is generally imprecise, where nothings seems to be in black and white (as we would like it to be), but the presence of gray areas seem to complicate the expression power and the learning process. Thus, in order to study this type of system, we need to be able to deal with this type of vague data. For us, this complication is catered easily by our ability of intuition and approximation. But in order to automate this type of learning into computers, the conventional techniques and logic are based on discrete and rigid concepts like '2 meters in length', 'temperature of 50 degrees' etc. Conversely, vague data like 'small length','mild temperature' etc are difficult or cannot

be properly dealt with.

Thus, the need to cater for this type of problem led to the invention of fuzzy logic by Zadeh [74] in 1965. It offers a larger ability to deal with imprecision than conventional mathematics and is used for handling uncertain and imprecise knowledge in real world applications. It has been widely and successfully used for decision-making in the presence of imprecise and noisy data. The aim of fuzzy logic is to inculcate the ability of vague and approximations other than precision. This gives rise to the concept of approximate reasoning which is a rule-based system. Knowledge representation in a rule-based system is done by means of IFĚTHEN rules. Furthermore, approximate reasoning systems allow fuzzy inputs, fuzzy antecedents, fuzzy consequents. Thus, this fuzzy representation allows a closer match with many of the important concepts of practical affairs, which lack the sharp boundaries assumed by classical logic.

Fuzzy logic is an extension of the Boolean logic but uses graded statements having different memberships rather than only true or false ones. In fuzzy logic, the value of the truth of any statement is a measured in degrees. In boolean logic the inference rules are of the form p -> q (p implies q). But with fuzzy logic, it is possible to say (.2* p) implies that (.2 * q). Here, we take an example, for the rule if (weather is sunny) then (we can play). For this example both the rules have a ranges of values. Fuzzy logic rely on membership functions to explain to the computer how to calculate the correct value between 0 and 1. The degree to which any fuzzy statement is true is denoted by a value between 0 and 1.

Fuzziness is quite a different notion from probability. Probability describes the objective uncertainty obtained on the basis of a large number of observations. Fuzziness describes the uncertainty that has a subjective meaning. Fuzzy notions describe the degrees of possession of a given property. The popularity of fuzzy set theory in solving control problems results from its ability to treat some situations which are difficult to deal with by the classical control theory. Fuzzy sets are used to control ill-defined, complex, non-linear systems. There are two aspects of this phenomenon. The first aspect concerns the descriptive ability of fuzzy sets. The second important feature is the applicability of fuzzy sets to work with incomplete, contradictory and subjective information. Fuzzy set theory is becoming increasingly important tool in the new fast developing disciplines of artificial intelligence

and expert systems.

6.2 Fuzzy Sets

A fuzzy set can be characterized by a membership function having degrees from 0 to 1. These can be easily implemented by fuzzy conditional statements. For example, if the antecedent is true to some degree of membership, then the consequent is also true to that same degree. $If'antecedent'Then'consequent'$. Thus, for example as a rule, if input one is 'low' and input two is 'high', then output is 'feasible' Else output is 'non-feasible'.

In a fuzzy classification system, a case or an object can be classified by applying a set of fuzzy rules based on the linguistic values of its attributes. Every rule has a weight, which is a number between 0 and 1, and this is applied to the number given by the antecedent. It involves two distinct parts. The first part involves evaluating the antecedent, fuzzifying the input and applying any necessary fuzzy operators. The most common fuzzy operators are listed below:

- $Union : \mu_{A \cup B}(x) = Max[\mu_A(x), \mu_B(x)]$
- $Intersection : \mu_{A \cap B}(x) = Min[\mu_A(x), \mu_B(x)]$
- $Complement : \mu_A - (x) = 1 - \mu_A(x)$

$\top(\mu_A, \mu_B)$ t-norms		$\perp(\mu_A, \mu_B)$ t-conorms	
intersection: $\wedge$, AND		union: $\vee$, OR	
$\mu_{A \cap B} = min(\mu_A, \mu_B)$	Zadeh	$\mu_{A \cup B} = max(\mu_A, \mu_B)$	Zadeh
$\mu_{A \cdot B} = \mu_A \mu_B$	product	$\mu_{A \hat{+} B} = \mu_A + \mu_B - \mu_A \mu_B$	probabilistic sum
$max(0, \mu_A + \mu_B - 1)$	bold $\cap$ Lukasiewicz	$min(1, \mu_A + \mu_B)$	bounded sum
$f(x) = \begin{cases} \mu_A & \text{if } \mu_B = 1 \\ \mu_B & \text{if } \mu_A = 1 \\ 0 & \text{otherwise} \end{cases}$	Weber	$f(x) = \begin{cases} \mu_A & \text{if } \mu_B = 0 \\ \mu_B & \text{if } \mu_A = 0 \\ 1 & \text{otherwise} \end{cases}$	Weber

Figure 6.1: Various T-norms and Co-norms by [21].

Where μ is the membership function. Apart minimum and maximum operators, there exists multiple intersection and union operators. The functions that fulfill a minimum of

requirements for an intersection operator are called tnorms. The functions that fulfill a minimum of requirements for a union operator are called t-conorms and these function are described in detail by Janikow in [17]. Figure 6.1 (taken from [21]) displays some of the t-norms and t-conorms operators defined in the literature. The t-norms are employed for defining AND operations in approximate reasoning, while the t-conorms serve the same role for OR operations. In the majority of fuzzy systems the min and max operators are used.

The second part requires application of that result to the consequent, known as inference. To build a fuzzy classification system, the most difficult task is to find a set of fuzzy rules pertaining to the specific classification problem.

6.3 Fuzzy Systems

A fuzzy inference system is a rule-based system that uses fuzzy logic, rather than boolean logic, to reason about data. It consists of a collection of control laws whose inputs and outputs are both fuzzy values and an inference mechanism so as to determine the control action for a given process state. Its basic structure includes four main components [17, 21]:

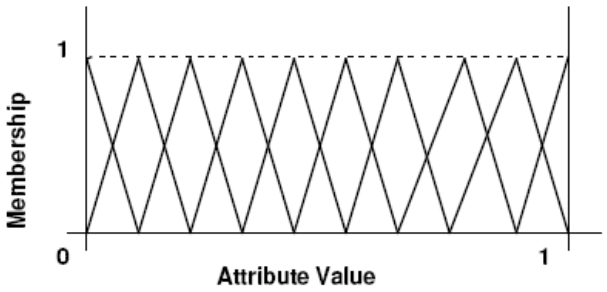

Figure 6.2: Fuzzy memberships.

6.3.1 A Fuzzifier

A fuzzifier translates crisp (real-valued) inputs into fuzzy values. The fuzzy sets are created and defined by the crisp inputs. Thus mappings between the inputs and the membership

degrees of the fuzzy sets are established as shown in figure 6.2. These mappings or transformation procedure can create multiple fuzzy sets which can be identical or different from each other. The shape of the membership function is a priori chosen or can even be decided upon the inputs. These shapes can be triangular, trapeziodal, gaussian or sigmoidal which each one having their own advantages in terms of the information they can contain. Figure 6.2 shows triangular membership functions which is the most frequently used function and are also used in this thesis. As the same figure also shows these fuzzy sets are overlapping sets and this amount of overlapping is important in terms of the degree of fuzziness that has to be introduced. This degree can be chosen a priori or dynamically, by considering the input and fitting the input data like in our techniques.

Thus, the fuzzifier generates fuzzy partitions by any one of the following two ways:

- **A Priori Partitioning**: This type of fuzzy partitioning is done a priori by the expert user.
- **Automatic Partitioning**: This is an automatic fuzzy partitioning of data by taking into into account the nature and structure of the data itself.

6.3.2 An Inference Engine

The inference engine applies a fuzzy reasoning mechanism to obtain a fuzzy output. This inference process is a decision making logic which determines fuzzy outputs corresponding to fuzzified inputs, with respect to the fuzzy rules in the rule base. Fuzzy systems like fuzzy decision trees can fire multiple rules in parallel for example firing rules along different branches of the tree but each can be of a different grade or degree. The inference process of a fuzzy system may be accomplished in three steps as follows [21]:

1. Determining the truth degree for each fuzzy rule of the satisfaction of its antecedent requiring the use of t-norms.
2. Determining the truth degree of the satisfaction of the consequent for each fuzzy rule, by propagation of the satisfaction of the antecedent to the consequent. This step is done by means of a fuzzy implication, like for example a min or product operation.

3. Conflict resolution from multiple consequents, thus aggregation of all the rules consequents requiring the use of t-conorms.

6.3.3 A Defuzzifier

A defuzzifier translates this latter output into a crisp value. The process of defuzzification is launched when a fuzzy set needs to be represented back in a crisp form. Thus, it finds the best crisp representation of the information contained in this fuzzy set. In any defuzzification algorithm there is a tradeoff between the need to find a single point result and the loss of information. There are multiple defuzzification strategies listed in [21] in order to transform a fuzzy set into a crisp value. These include finding the point which present the maximum truth, average of all values where a maximum membership degree is achieved or the weighted mean of the fuzzy region which is the center of gravity of the fuzzy set.

6.3.4 A Knowledge Base

The knowledge base contains both an ensemble of fuzzy rules, known as the rule base, and an ensemble of membership functions known as the database. The decision-making process is performed by the inference engine using the rules contained in the rule base. These fuzzy rules define the connection between input and output fuzzy variables.

6.4 Potential Contributions of Fuzzy Set Theory

The potential contributions of fuzzy theory has been discussed in [37]. Fuzzy theory enhances the process of learning in the following ways:

1. The authors discuss the concept of 'graduality' which is the ability of the fuzzy sets to represent a gray region or approximate concepts. These concepts are vague and uncertain in nature and one will usually agree only to a certain extent that an object belongs to a concept. Thus, an obvious idea is to induce fuzzy concepts, that are formally identified by a fuzzy rather than a crisp subset. Also, in learning problems the patterns of interest have non crisp transitions and thus, this property of fuzzy sets is highly desirable.

2. The ability of fuzzy sets to express discrete and continuous structures in terms of natural language is very desirable for effective representational characteristics. Thus, crisp sets and be presented in linguistic terms and hence in a comprehensible way. Although, in order to increase the accuracy of complex fuzzy models, it has to be represented in a larger number of rules thus, decreasing its interpretability. Thus, there is a trade-off between these two.

3. A significant capability of fuzzy sets is there robustness towards variations of the data. In this thesis, we tend to exploit this characteristic of fuzzy sets. A learning or data mining method is considered robust if a small variation of the observed data does hardly alter the induced model or the evaluation of a pattern. We take an example of a crisp discretization method where, a small shift of the boundary of an interval can have a drastic effect on the membership of the test instance to a particular class, if many data points are located near the boundary. This effect is alleviated when using fuzzy sets instead of intervals.

4. Fuzzy sets have made important contributions to the representation and processing of uncertainty. The data presented to learning algorithms is imprecise, incomplete or noisy most of the time, a problem that can badly mislead a learning procedure. But even if observations are perfect, the generalization beyond that data, the process of induction, is still afflicted with uncertainty.

5. Fuzzy set-based modeling techniques provide a convenient tool for making expert knowledge accessible to computational methods and, hence, to incorporate background knowledge in the learning process. One very obvious approach is to combine rule-based modeling and learning.

6.4.1 Generalized Aggregation Operators

Many data mining methods make use of logical and arithmetical operators for representing relationships between attributes in models and patterns. In decision tree induction, for example, each inner node represents an equality or an inequality predicate, and these predicates are combined in a conjunctive way along a path of a tree. A large repertoire

of generalized logical (e.g. t-norms and t-conorms) and arithmetical (e.g. Choquet- and Sugeno-integral) operators have been developed in fuzzy theory and related fields. Thus, a straightforward way to extend standard learning methods consists of replacing standard operators by their generalized versions.

The general effect of such generalizations is to make models more flexible. For example, while a standard decision tree can only produce axis-parallel decision boundaries, these boundaries can become non-axis-parallel for fuzzy decision trees where predicates are combined by means of a t-norm.

6.5 Some Drawbacks in Fuzzy Learning

One disadvantage arises when a fuzzy predictor is supplemented with a 'defuzzification' mechanism (like a winner-takes-all strategy in classification), many of its merits are lost. In the classification setting, for instance, a defuzzified fuzzy classifier does again produce hard decision boundaries in the input space. Thereby, it is actually reduced to a standard classifier.

The use of linguistic modeling techniques does also produce some disadvantages, however. A first problem concerns the interpretation of fuzzy models: Linguistic terms and, hence, models are highly subjective and context dependent.

A solution to guarantee transparency of a fuzzy model is to let a user of a data mining system specify all fuzzy concepts by hand, including the fuzzy partitions for all of the variables involved in the study under consideration. This has two problems. Firstly, the job is of course tedious and cumbersome if the number of variables is large. Secondly, much flexibility for model adaptation is lost, because it is by no means guaranteed that accurate predictive models or interesting patterns can be found on the basis of the fuzzy partitions as pre-specified by the user.

It is well-known that learning from empirical data will be most successful if the model class under consideration has just the right flexibility, since both over and under fitting of a model can best be avoided in that case. Therefore, the question whether a fuzzy generalization will pay off cannot be answered in general: If the original (non-fuzzy) hypothesis

space is not flexible enough, the fuzzy version will probably be superior. On the other hand, if the former is already flexible enough, a fuzzification might come along with a danger of over-fitting.

6.6 Fuzzy Decision Trees

The most important feature of decision trees is their capability to break down a complex decision-making process into a collection of simpler decisions and thereby, providing an easily interpretable solution. ID3 [65] is a popular and efficient method of decisionŮmaking for classification of symbolic data and is generally not suitable in cases where numerical values are to be operated upon. Since most real life problems deal with nonsymbolic (numeric, continuous) data, they must be discretized prior to attribute selection. Classification and Regression Trees (CART) [75] and C4.5 [66], however, do not require prior partitioning. Here the thresholds are dynamically computed depending on the conditions along a path, and often result in the multiple use of a particular attribute with different thresholds. This can lead to increased simplicity at the cost of reduced comprehensibility.

In the methods listed above, the cut points used in classical decision trees are usually crisp. Real life experience and applications show that these approaches could only work well for well separated classes with clearly defined boundaries. However, due to the existence of vague and imprecise information in real-world problems, the class boundaries may not be defined clearly and the class distributions have significant overlap. A hard-decision approach usually covers up these ambiguities by a forced recognition [138]. In which case, the decision tree may produce high misclassification rates in testing even if they perform well in training. To overcome this drawback, several approaches have been proposed, including probability based approaches [40], and possibility based methods [131]. In classical decision trees, given an unknown object, only one branch of tree (one rule) is fired at a time in classical decision tree, which may lead to opposite following branch when the input is close to the cut point, and thus generate a wrong result. This drawback could be overcome by firing multi-branches but with various certainty degrees. This could be implemented by means of fuzzy set theory, leading to fuzzy decision trees. In fuzzy decision trees [96, 17] the fuzzy reasoning process

allows two or more rules to be simultaneously validated with gradual certainty and the end result will be the outcome of combining several results.

An inherent problem with decision trees using normal discretization or computing dynamic thresholds is that it cannot provide any information about the intersection region where the pattern classes are overlapping. Another disadvantage of classical crisp decision trees is their brittleness. A wrong path taken down the tree can lead to widely erroneous results or error accumulation. Though using heuristics such as proper tree search strategies, this error can be reasonably controlled, but one method to overcome this difficulty altogether is to make the decision tree fuzzy. A fuzzy non-terminal node will allow more than one path to be followed down the tree. The fusion of fuzzy sets with decision trees enables one to combine the uncertainty handling and approximate reasoning capabilities of the former with the comprehensibility and ease of application of the latter. This enhances the representative power of decision trees naturally with the knowledge component inherent in fuzzy logic, leading to better robustness, noise immunity, and applicability in uncertain/imprecise contexts. Most fuzzy decision trees assume that all domain attributes or linguistic variables have pre-defined fuzzy terms, determined in a data-driven manner using fuzzy restrictions. The information gain measure [16], used for splitting a node, is modified for fuzzy representation and a pattern can have nonzero match to one or more leaves.

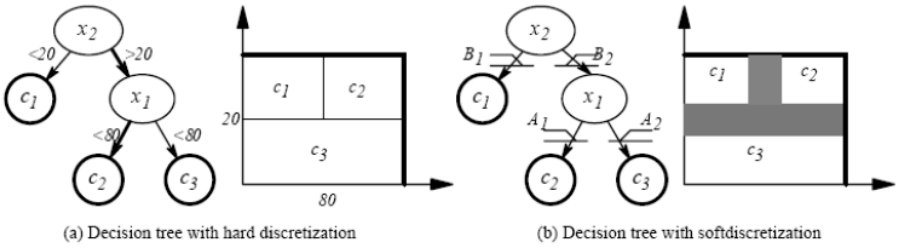

Figure 6.3: Comparison between crisp and fuzzy decision trees with an example.

The potential of fuzzy decision trees in improving the robustness and generalization in

classification is due to the use of fuzzy reasoning. An example of figure 6.2 [138] illustrates the difference between classical and fuzzy decision trees. Figure 6.2(a) shows a classical decision tree based on crisp discretization, while the tree in figure 6.2(b) uses soft or fuzzy discretization. In both decision trees, each path from the root node to a leaf node establishes a classification rule. Using crisp discretization, the decision space is partitioned into a set of non-overlapping subspaces, as shown on the right of figure 6.2(a), in which each data point is assigned to a single class. In contrast, a fuzzy decision tree gives results within [0,1], as the possibility degree of an object matching the class, as shown on the right of figure 6.2(b). Fuzzy decision trees provide a more robust way to avoid misclassification. For example, given an example with attributes $X1$ and $X2$ having values 82 and 23 respectively, the conclusion reached by the classical decision tree is class C3. However, if the sampled data point has moved to a new point with a small change of value due to noise, for example X1=79 and X2=25, then the classical decision tree would give a wrong result: class C2. In contrast, the fuzzy decision tree gives a result about the possibilities that the object belongs to classes C1, C2, C3. According to these results, the human users could make their final decision or perform further investigation, or a high-level meta-learner can be designed to learn further. As a result, the rate of misclassification could be reduced.

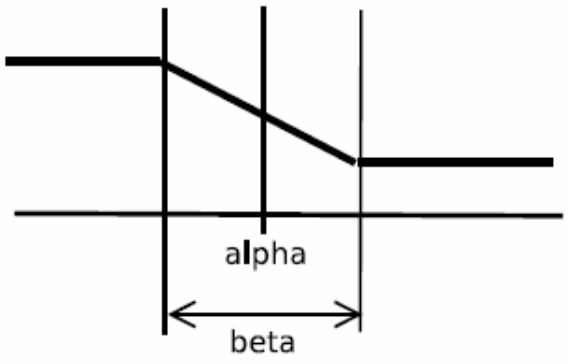

Figure 6.4: A fuzzy transition region.

A hard discretization generates the boundary between two or more crisp sets. Alternatively, a soft or fuzzy discretization is defined by a fuzzy set which forms a fuzzy partition as

shown in figure 6.3 [21]. In contrast to the classical method of non-overlapping partitioning, the fuzzy discretization is overlapped and is defined with two parameters/functions. The two parameters defining it are: α, which is the location of the cut-point and corresponds to the split threshold in a test node of a decision or a regression tree, and β which is the width, the degree of spread that defines the transition region on the attribute chosen in that node. With such a piecewise linear discriminator function identical to the one used in [96], the local input space of a node is split (fuzzy partition) into two overlapping subregions of objects. Some objects go only to the left successor, some only to the right one, and the objects in the overlap region go to both successors. The larger the transition region in a test node, the larger the overlap and the softer the decision in that node.

In consequence, any given instance is in general propagated through the tree by multiple decision paths in parallel: in the simplest case through one path, in the most complex case, through all the paths. This given instance does not effectively belong to a node it passes through, but has a membership degree attached to that node. Thus, the node may be seen as a fuzzy set. Finally, the given instance reaches multiple terminal nodes and the output estimations given by all these terminal nodes are averaged in order to obtain the final estimated membership degree to the target class.

6.7 Construction of Fuzzy Decision Tree and State of the Art

Like classical decision trees with the ID3 algorithm, fuzzy decision trees are constructed in a top-down manner by recursive partitioning of the training set into subnets. In this section, we discuss previous works by various authors related to the generation of fuzzy decision trees. In the previous sections, we identified the different steps and components in the generation of fuzzy decision trees and below we discuss the different research done in each of these components.

6.7.1 Attribute selection criteria or Discriminator Function

Generally, tree-structured approaches use to select a split based on a minimization of an impurity measure. This measure, called also discriminating measure used to select the best

attribute for the partitioning of a (fuzzy) set, may differ from the measure used to fuzzy partition (split) the set in one or more fuzzy sets (the difference between finding the attribute and finding its best threshold for split in a node). Measures presently employed in a fuzzy decision tree to evaluate the most discriminating attribute in a node based on local growing set and sometimes also to search for the best choice of attribute fuzzy partitioning (if this fuzzy partitioning is not already done a priori to tree building, as it is the case of ID3 based fuzzy decision trees) are:

- Information Gain [54, 86, 17, 92]: The most commonly used is information gain, which is computationally simple and shown effective: select an attribute for testing (or a new threshold on a continuous domain) such that the information difference between that contained in a given node and in its children nodes is maximized. The information content is measured according to [17].

- Extended Information Measure: In [134], they suggest a different way of computing information measure in FDT to make information gain ratio applicable as a selection measure.

- Gain Ratio [85]: Gain ratio is used in classical decision trees to select the test attribute in order to reduce the natural bias of information gain, i.e., the fact that it favors attributes with many values (which may lead to a model of low predictive power). In FDT, fuzzy partitions are created for all attributes before the tree induction. To keep the tree simple, usually each partition possesses as few fuzzy sets as possible. The information gain ratio is measured according to [66].

- Yuan and ShawŠs Measure: In [141], a method is introduced to construct FDT by means of a measure of classification ambiguity as a measure of discrimination. This measure is defined from both a measure of fuzzy subset hood and a measure of non-specificity. The measure of ambiguity was introduced by to measure the discriminating power of the attribute.

- Criteria based on Fuzzy-Rough Sets: It has been shown that fuzzy-rough metric is a useful gauger of (discrete and real-valued) attribute information content in datasets.

This has been employed primarily within the feature selection task, to benefit the rule induction that follows this process [92]. Fuzzy-rough measure is comparable with the leading measures of feature importance. Its behavior is quite similar to the information gain and gain ratio metrics. The results show that the fuzzy-rough measure performs comparably to fuzzy ID3 for fuzzy datasets, and better than it for crisp data [92].

- Other fuzzy entropy based include the (fuzzy) star entropy [28], the fuzzy entropy defined by De Luca and Termini [4], the index of fuzziness of Kaufmann, a measure defined by Yager and the fuzzy entropy of Kosko [10] with DombiŠs generalized fuzzy operators;

- Kolmogorov-Smirnov distance based [129].

- Fuzzy sum of squared errors [33].

- Other methods in which a neural network is used to compute the score [6] or a genetic algorithm [115] for structure identification of the fuzzy decision tree.

6.7.2 Fuzzification or Fuzzy Partitioning

The process of construction of fuzzy decision trees (FDT) is based on the knowledge of fuzzy partitions of each numeric attribute which can be difficult to possess. A widely used shape of discriminator function is the piecewise linear one as in [96](see figure 6.3). The two parameters defining it are: β, which is the location of the cut-point and corresponds to the split threshold in a test node of a decision tree, and α which is the width, the degree of spread that defines the transition region on the attribute chosen in that node. With such a piecewise linear discriminator function, the local input space of a node is split (fuzzy partitioned) into two overlapping subregions of objects. Some objects go only to the left successor, some only to the right one, and the objects in the overlap region go to both successors. The larger the transition region in a test node, the larger the overlap and the softer the decision in that node. In consequence, any given instance is in general propagated through the tree by multiple decision paths in parallel: in the simplest case through one path, in the most complex case, through all the paths having a membership degree attached

to that node. Finally, the given instance reaches multiple terminal nodes and the output estimations given by all these terminal nodes are aggregated through some defuzzification scheme in order to obtain the final estimated membership degree to the target class.

Two types of fuzzy partitioning are identified in the literature:

6.7.2.1 A Priori Partitioning

This is a common practice in fuzzy decision trees. In many approaches the fuzzy sets are user defined i.e the fuzziness degree is a fixed one for a given attribute [63, 92, 104]. While, in others they simply evenly fuzzy partition an attribute when the degree of overlap is specified [61]. In [89] Ramdani proposes is a rare technique that comprises of searching for the best degree of fuzziness in a tree. However, there are approaches that perform fuzzy clustering of the attribute, based on KohonenŠs feature-maps [141] or use fuzzy statistics that considers counting the data located nearby several points on the fuzzy set [50].

6.7.2.2 Automatic Partitioning

The drawback of the fuzzy partitioning a priori to tree building is usually that the class does not interfere in the process and the fuzzification is done independently of the target class. The automatic generation as the tree is developed permits a fuzzy partition at each step of the tree building, thus the induced partition being related to the local learning set of the current step of the tree building.

Automatic partitioning methods involve fuzzy partitioning as the tree is being developed. In [96] the fuzzy partitions are generated by finding the optimal width of the transition region β by adapting a Fabonacci searching mechanism. [18] generates membership functions by dynamic optimization based on genetic algorithms at a node while building the tree (the genetic algorithm encodes the center and width of the fuzzy membership functions subject to tuning). [51] locally optimizes them by adjusting t-norms and t-conorms operators, both defined function of a free parameter and [54] locally optimizes them by fuzzy c-means clustering.

Another unique automatic fuzzy partitioning of continuous attributes was developed by Marsala [27] based on mathematical morphology. The learning set of an attribute is treated

as a 'word' so that mathematical morphological operators like erosion and dilation, opening and closure, and filtering can be applied. These operators actually smooth this word forming a fuzzy partition. A discriminator based on fuzzy entropy is used to compare different fuzzy partitions formed.

The encountered shapes of the membership functions used in the fuzzy partitioning of an attribute are: piecewise linear (triangular, trapezoidal), sigmoidal or Gaussian [47, 21].

6.7.3 Stopping Rule

Usually classical tree learning is terminated if all attributes are already used on the current path; or if all examples in the current node belong to the same class. In FDT an example may occur in any node with and membership degree. Thus general more examples are considered per node and fuzzy trees are usually larger than classical trees. To solve this problem, there are many methods mentioned.

The usual criteria encountered are split as the following:

- A threshold limit of improvement of a certain criteria at a node such as fuzzy probability of the node (relative frequency) with respect to one class [17], fuzzy entropy of the node [27, 17]. Limitation of the truth level of classifying into one class [141], if the performance on the test set starts to deteriorate [33] or if the next best splitting does not improve a criteria by [128].

- Limitation of the depth of the tree [128] or the number of attributes already chosen in the grown tree [27].

- A limit of the number of instances in the local learning set of the node [27, 7] or class based where at least 90 percent of the objects in node belong to one class [86] or could also be based on the cardinality of a node by a certain threshold [51, 17].

- Pruning in order to remove branches with little statistical validity [96].

- Look-ahead methods that attempt to establish a decision at a node by analyzing the classifiability of instances along each of the branches of a split [85].

6.7.4 Inference

The most profound differences between fuzzy and classical decision tree are in the process of classifying a new sample. These differences arise from the following two reasons [117]:

- Each leaf of the fuzzy decision tree could contain samples of different classes with different degrees of match.
- Thus, the inference procedure is likely to match the new observations against multiple leaves, with varying degrees of match.

To account for these potential problems, a number of inference routines have been proposed. During classification in a fuzzy decision tree, new examples follow branches following restrictions as each node and reach the leaves. The fuzzy decision tree then combines their decisions into a single crisp response. Such decisions are very likely to have conflicts, which can be found both in a single leaf or across different leaves. Thus, fuzzy decision trees differ in the choice of the inference measures: t-norms (min, product, and), t-conorms (max, sum, or), etc. [17] defines four different inferences. [27] tries several operators with its method: Zadeh, probabilistic and Lukasiewicz.

The most frequent application of FDT is the induction or the adaptation of rule-based models. Plenty of methods has been developed for inducing a fuzzy rule base from the data given. Among them [19, 134, 47], the main difference concerns the way in which individual rules or their condition parts are learned. Thus, we can also classify a new sample or examples depending on the Rule Base [117].

6.7.5 Defuzzification

Defuzzification. The final answer of a fuzzy decision tree is computed with one of the defuzzification methods: max-criterion [127], center-of-area [7], mean-of-maximum [61], etc.

6.8 Some More Fuzzy Decision Tree Methods

Without being exhaustive below we present some fuzzy decision tree methods from the literature that have also been mentioned in the above categories. Some equations below

have been taken from the authors work. They are listed below:

6.8.1 Xizhao Wang et al [143]

The authors focus on an important problem of whether or not there exists an exact algorithm for constructing the smallest-scale fuzzy decision tree. Centering on this optimal problem, this paper discusses the representation of fuzzy decision trees and gives a comparison between fuzzy decision trees and crisp ones; proves that the algorithm complexity for constructing a kind of smallest scale fuzzy decision tree is NP-hard; points out the inherent defect of the fuzzy ID3 and presents a new algorithm for constructing fuzzy decision trees.

6.8.2 J.F. Baldwin et al [61]

The authors describe a method for generating probabilistic decision trees with fuzzy attribute and classification values. The decision trees generated are probabilistic classifiers analogous to those suggested by Quinlan in [16] where the probabilities are calculated according to the mass assignment semantics for fuzzy sets developed by Baldwin (see [3] and [4]). This algorithm has been implemented in Fril [51] which is a logic programming style language with built in capabilities for processing both probabilistic and fuzzy uncertainty. The decision trees can be represented in terms of Fril extended rules the syntax and semantics

6.8.3 Xiaomeng Wang et al [133]

The authors adapted the ID3 algorithm to construct fuzzy decision trees and borrowed some basic ideas from [17]. They go beyond JanikowŠs work and consider how to extract a fuzzy rule base from the induced fuzzy tree, and study heuristics to simplify it. In addition they discuss how to treat missing values.

6.8.4 Ming Dong et al [85]

The authors propose a novel method of evaluating the classifiability of instances along the branches of a node split. Based on this method, they propose a fuzzy decision tree induction algorithm that utilizes look-ahead to produce smaller decision trees.

6.8.5 J.A. Abonyi et al [56]

The authors attempt to obtain compact, accurate and linguistically interpretable fuzzy rule-based classifiers from labeled observation data in an iterative fashion. An initial model is derived from the observation data and subsequently, feature selection and rule base simpli

cation methods [10] are applied to reduce the model. After the model reduction, a real-coded GA is applied to improve the classification accuracy [7,11]. To maintain the interpretability of the rule base, the GA search-space is restricted to the neighborhood of the initial rule base.

6.8.6 Ferenc Peter Pach et al [38]

This work investigates how supervised clustering can be used for the effective partitioning of the input domains. In contrast with other cluster based approaches with most discretization of continuous variables that deal with a single variable only, this approach concerns all the variables discussed at same time. The discretization mechanism is straightforward: project the cluster prototypes on the respective axes (coordinates) and construct the discretization intervals.

6.8.7 C. Z. Janikow et al [17]

The authors have described a number of possible fuzzy inferences were described in their previous works [18]. In this paper, they describe a few different inferences, following the idea of exemplars [19] and implementing the local center-of-gravity inference with various interpretations of those two levels of conflicts. In exemplar-based learning, selected training examples are retained, and a decision procedure returns the class assigned to the 'closest' exemplar. Moreover, these examples can be generalized, and multiple exemplars can be used for making the decision.

6.8.8 Hung Son Nguyen et al [46]

They propose a novel approach based on soft cuts and leads to new efficient strategies in searching for the best cuts (both soft and crisp cuts) using the whole data. They show

some properties of considered optimization measures allowing to reduce the size of searching space. They try to prove that using only O(log N) simple queries, one can construct the partition very close to optimal.

6.8.9 Yoon-Seok Choi et al [137]

The authors optimize the parameters that specify fuzzy partitioning by genetic algorithms. They evolve the number of intervals, the boundaries, and the degrees of overlapping for fuzzy sets. For efficiency, they mix genes of discrete values and continuous values.

6.8.10 Jay Fowdar et al [60]

The authors try to improve the classification accuracy of decision tree induction in non-deterministic domains. The research involved applies the principles of fuzzy theory to the CHAID (Chi-Square Automatic Interaction Detection) algorithm in order to soften the sharp decision boundaries which are inherent in traditional decision tree algorithms. CHAID is a decision tree induction algorithm with the main feature of significance testing at each level, leading to the production of trees which require no pruning. The application of fuzzy logic to CHAID decision trees can represent classification knowledge more naturally and inline with human thinking and are more robust when it comes to handling imprecise, missing or conflicting information.

6.8.11 Christophe Marsala and Bernadette Bouchon-Meunier [26]

The authors propose to infer a fuzzy partition in an automatic way, in an intermediate step of the construction of the decision tree. This method is easy to implement and gives good results. In this paper, they present a solution based on the use of mathematical morphology operators, and formalized by means of tools from theory of formal language. The implementation of this solution is described in [27].

6.8.12 Chengming Qi et al [22]

The authors present a modified fuzzy decision tree model (MFD) in order to diminish the cost of the tree-constructing. Entropy of multi-valued and continuous-valued attributes is

both computed with fuzzy theory after fuzzification, while entropy of other attributes is dealt with general Shannon method.

6.8.13 M.Fajfer and C.Z. Janikow [87]

The authors describe a new bottom-up domain partitioning technique. The previously available technique was aimed at producing the minimal set of partitions locally necessary for minimizing tree size. As such, it was a top-down technique, implementing domain splitting rules while building a tree. However, the fuzzy decision tree has just been extended to a decision forest (described separately), which uses redundant knowledge for performing more elaborate classification. This redundancy requires more global rather than minimal local partitioning. Such a domain partitioning is presented here.

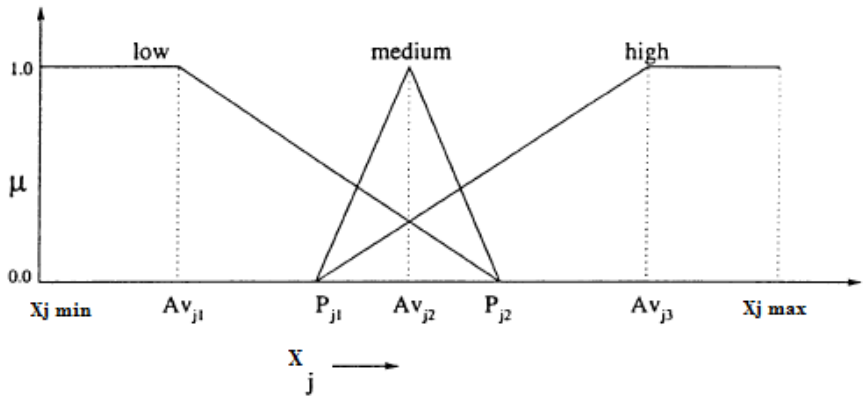

Figure 6.5: Linguistic rules based partitions.

6.8.14 Sushmita Mitra et al [107]

The authors describe the formulation of a fuzzy knowledge-based network using the principle of a fuzzy decision tree. Quantitative measures are defined to evaluate the effectiveness of the fuzzy decision tree and the linguistic rules. Discretization of continuous attributes, based on the distribution of pattern points in the feature space, is made in linguistic terms using quantiles. Unlike other fuzzy decision trees, this discretization to boolean form helps in

reducing the computational complexity while preserving the linguistic nature of the decision in rule form. New fuzziness measures, in terms of class memberships, are used at the node level of the tree to take care of overlapping classes. Pruning is used to minimize noise, resulting in a smaller decision tree with more efficient classification. The extracted rules are mapped onto a fuzzy knowledge-based network. The fuzzy partitioning algoritm is presented below:

Let $X_{j_{max}}$ and $X_{j_{min}}$ denote the maximum and minimum values encountered along feature X_j considering N training patterns $X_{1_j}, X_{2_j}, ..., X_{N_j}$. Let these patterns be sorted in the ascending order of their values along the jth axis. The first quantile P_{j_1} is the value of X_j that exceeds one-third of the measurements and is less than the remaining two-thirds. The second quantile P_{j_2} is the value of X_j that exceeds two-thirds of the measurements and is less than the remaining one-third. In order to determine the two quantiles, we divide the measurements into a number of small class intervals of equal width δ and count the corresponding class frequencies f_i. The position of the kth partition value (here quantile, as 1,2 for three partitions) is calculated as:

$$P_{j_k} = l_i + \frac{R_k - cf_{i-1}}{f_i}.\delta$$

where l_i is the lower limit of the ith class interval, $R_k = N.k/3$ is the rank of the kth partition value, and cf_{i-1} is the cumulative frequency of the immediately preceding class interval, such that $cf_{i-1} < R_k < cf_i$. Then, in figure 6.4 they have $Av_{j_1} = (X_{j_{min}} + P_{j_1})/2$, $Av_{j_1} = (P_{j_1} + P_{j_2})/2$ and $Av_{j_3} = (P_{j_2} + X_{j_{max}})/2$.

The membership values of a pattern along the jth axis, in the corresponding three-dimensional linguistic space, is computed as:

$$\mu_{low}(x_j)(X_i) = \left\{1, for F_{ij} < Av_{j_1}; \frac{P_{j_2} - F_{ij}}{P_{j_2} - Av_{j_1}}, for Av_{j_1} \leq F_{ij} < P_{j_2}; 0, otherwise\right\}$$

$$\mu_{medium}(x_j)(X_i) =$$

$$\left\{0, for F_{ij} < P_{j_1}; \frac{Av_{j_2} - F_{ij}}{Av_{j_2} - P_{j_1}}, for P_{j_1} \leq F_{ij} < Av_{j_2}; \frac{P_{j_2} - F_{ij}}{P_{j_2} - Av_{j_2}}, for Av_{j_2} \leq F_{ij} < P_{j_2}; 0, otherwise\right\}$$

$$\mu_{high}(x_j)(X_i) = \left\{0, for F_{ij} < P_{j_1}; \frac{F_{ij} - P_{j_1}}{Av_{j_3} - P_{j_1}}, for P_{j_1} \leq F_{ij} < Av_{j_3}; 1, otherwise\right\}$$

Fuzziness is incorporated into the ID3 algorithm at the node level by modifying the conventional decision function, with classical Shannon entropy, by the inclusion of different

fuzzy measures. The fuzzy entropy considers the membership of a pattern to a class and helps enhance the discriminative power of an attribute. In order to reduce the effect of noise or exceptions, a node is pruned depending on the number of patterns reaching it. For this purpose, a threshold is defined as a lower bound on the fraction of patterns allowed in an existing node.

6.8.15 C.Olaru and L.Wehenkel [96]

The authors present a complete fuzzy decision tree that caters to the problem of automatic fuzzy partitioning, tree growing, pruning and optimization of the fuzzy decision tree. Here, we discuss their method that implies the automatic generation of a fuzzy split of the most discriminating attribute at each new developed node of the tree. The procedure used to achieve this goal is further detailed below.

Objective: Given S, fuzzy set in a fuzzy decision tree, find attribute X(.), threshold α and width β (parameters defining the discriminator function I) together with successors labels L_L and L_R, so as to minimize the squared error function:

$$E_S = \sum_{\omega \in S} \mu_S(\omega)[\mu_C(\omega) - \omega'_C(\omega)]^2$$

where,

$$\omega'_C(\omega) = \nu(X(\omega), \alpha, \beta)L_L + (\nu(X(\omega), \alpha, \beta))L_R$$

The squared error of equation above called score measure or discriminating measure is in our case employed both for measuring the power of each attribute to split the input space in subsets of the same output, and for fuzzifying each attribute.

The local search is decomposed as follows [21]:

- Searching for the attribute and split location: With a fixed $\beta = 0$ (crisp split) they search among all the attributes for the attribute yielding the smallest crisp E_s, its optimal crisp split threshold α, and its corresponding (provisional) successors labels L_L and L_R, by using crisp heuristics adapted from CART regression trees.

- Fuzzification and labeling: With the optimal attribute and threshold α kept frozen, both already chosen in the previous step, they search for the optimal width β by

FIBONACCI search; for every new value, the two successors labels L_L and L_R are automatically updated to every candidate value of by explicit linear regression formulas.

Chapter 7

Automatic Fuzzy Partitioning for Fuzzy Decision Trees

In this chapter, we discuss our solutions based on automatic fuzzy partitioning during fuzzy decision tree construction. The first one is based on resampling based technique that extracts fuzzy partitions from a discretization point distribution built from repeated resampling and the second is based on a multi-interval recursive fuzzy partitioning method.

7.1 Our Fuzzy Decision Tree

In this section we present the attribute selection criteria or discriminator function and the fuzzification process used in our fuzzy decision tree. As discussed earlier our contribution is regarding the fuzzy partitioning phase of the fuzzy decision tree construction where we present two different techniques.

A solution to guarantee transparency of a fuzzy model is to let a user of a data mining system specify all fuzzy concepts by hand, including the fuzzy partitions for all of the variables involved in the study under consideration. This has two problems. Firstly, the job is of course tedious and cumbersome if the number of variables is large. Secondly, much exibility for model adaptation is lost, because it is by no means guaranteed that accurate predictive models or interesting patterns can be found on the basis of the fuzzy partitions as pre-specified by the user. In fact, in most methods the fuzzy partitions are rather adapted

to the data in an optimal way, so as to maximize the model accuracy or the interestingness of patterns.

We focus on building fuzzy of soft partitions during the process of discretization without any expert knowledge available. This soft discretization is then used by the decision tree such as ID3 [65] which makes use of these fuzzy partitions making them fuzzy or soft decision trees. Our first fuzzy discretization approach is an extension of the RSD discretization discussed in chapter 4. Thus similarly, we estimate the discretization point distribution over an attribute X_i by repeatedly resampling and performing discretization on each bootstrap sample using any of the discretization approaches and thus, creating a histogram density function of the obtained candidate discretization points. Then, we apply a moving average filter to this function with a window size ws and try to obtain distinct regions (peaks) in the distribution most likely to be the exact discretization points towards the entire population. These peaks are regions of high probability of finding the exact discretization point towards the entire population, thus, we extract our partitions from these regions and fuzzify them to obtained fuzzy partitions. This process is explained in the following sections.

In our second fuzzy partitioning approach, we adopt a top-down binary partitioning method that starts with a single interval and recursively partitions into multiple intervals. The reason for partitioning in multiple intervals because sometimes it seems to be better than only binary discretization. It can lead to more compact and more accurate decision trees whereas the explanation capability of the decision tree to the user might be better as well. A heuristic searches for the best split at each node which can be crisp or fuzzy depending on the region where the split is taking place. To know when to perform a fuzzy split we identify fuzzy regions with highly mixed classes and imprecisions. This process is discussed in detail in the following sections.

7.1.1 Discriminator Function

A measure of discrimination has to take into account the degrees of membership to a fuzzy value for each example of a training set. The method that we use is based on the measure of discrimination in the presence of fuzzy values is an extension of the Shannon entropy called the star-entropy [89]. It combines a measure of uncertainty (probability) and a measure of

imprecision (fuzziness). The fuzzy entropy (entropy star) of X is an extension of Shannon entropy to fuzzy event then defined by: $I_f(X) = -\sum_{i=1}^{m} p_f(C_i) log p_f(C_i)$.

This measure is obtained from the classical Shannon measure of entropy by substituting the Zadeh's probability measure P_f of fuzzy events to the classical probability. Given a fuzzy set X of examples $X = \{x_1/\mu_1, x_2/\mu_2, .., x_n/\mu_n\}$ with μ_i is the membership degree of example x_i in the set X, the probability of fuzzy event C_i in X is defined by: $P_f(C_i) = \frac{\sum_{x_j \in C_i} \mu_j}{\sum_1^n \mu_j}$.

The selected fuzzy cut point is the one that minimizes: $I_f(P/A_i) = \sum_{l=1}^{k} \frac{|P_l|}{|P|} I_f(P_l)$. This is the entropy of the sub-training set conditioned by the attribute discretized by the fuzzy cut-point. Note that: $\sum_{l=1}^{k} \frac{|P_l|}{|P|} = 1$. After the discretization of all numerical attributes, the gain of information brought out by numerical attribute A_k, is valued: $G_f(A_k) = I_f(P) - I_f(P_{A_k})$.

7.1.2 Fuzzy Partitioning

We use automatic method for fuzzy partitioning which are discussed in detail in the next section. The two parameters involved are the cut point threshold α and the width of the fuzzy or transition region β. These parameters help to constitute our piecewise linear membership function as discussed in the previous chapter. Thus our piecewise linear membership function can be written as:

$$PiecewiseLinear : \mu_S(\omega) =$$
$$\left\{1, if \omega < \alpha - \frac{\beta}{2}; 0.5 + \frac{\alpha - \omega}{\beta}, if \alpha - \frac{\beta}{2} \leq \omega \leq \alpha + \frac{\beta}{2}; 0, if \omega > \alpha + \frac{\beta}{2}\right\}$$

7.2 Our Fuzzy Partitioning Techniques

In this section we present in detail our two fuzzy partitioning techniques.

7.2.1 Resampling based Smoothed Fuzzy-Partitioning (RSF)

This fuzzy partitioning technique is based on the multi-interval crisp discretization technique known as RSD presented in chapter 4. This method builds a discretization-point distribution by repeatedly resampling and obtaining discretization points for each bootstrap sample similar to RSD. It then identifies peak regions which are the regions of high probabilities

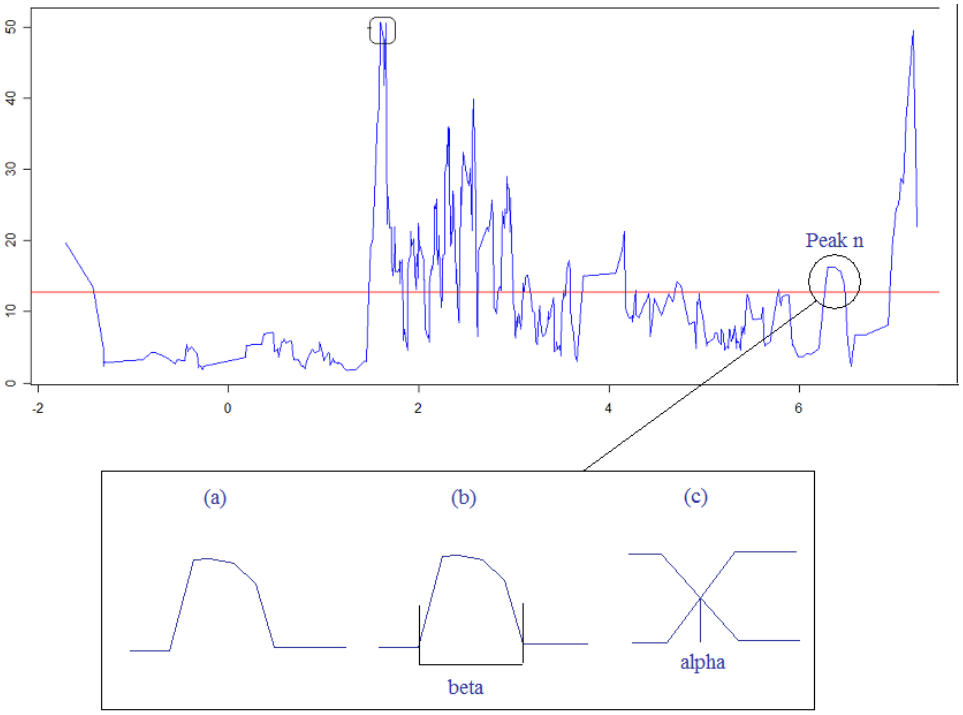

Figure 7.1: Above) Discretization point distribution obtained after the the smoothing of the frequency histogram. Below) Fuzzification of crisp partitions obtained from each peak (Region of high probability of finding exact discretization points).

of finding a discretization point as previously in RSD. We consider these high probability regions as our fuzzy regions such that the individuals (with high probability) falling in these regions will not be classified as totally wrong but will be given a fuzzy membership [0;1]. This method is explained below:

7.2.2 Step 1: Building a Discretization-Point Distribution by Resampling

This technique is carried out in two phases. In the first phase we estimate a discretization point distribution over an attribute X_i by repeated resampling n times and performing discretization on each bootstrap sample Ω_{bs} using an entropy based MDLPC method and thus, creating a histogram density function of the resulting candidate points as shown in figure 7.1 (above). Note that our method supports the use of any discretization method including MDLPC. During the discretization performed on each bootstrap sample, we record

the number of intervals for each sample and build the histogram density function for the number of intervals obtained for n bootstraps. Then from this interval distribution we select the most probable number of intervals obtained denoted as I_{best}. In the second phase we smooth over the discretization point distribution function by applying a moving average filter with a window size ws. The resulting smoothed curve is shown in figure 7.1 (above). We can see distinct regions (or peaks) where the probability of the candidate points to be the exact discretization point is higher. As shown in figure 7.1 (above), the straight line that runs parallel with the x-axis is called the threshold parameter T, which is set as the median of the obtained frequency values. By taking into account this threshold T we define peaks P_{reg} as the regions which lie above this threshold line. The reason for defining such a threshold is to help to elaborate only the regions of higher probabilities (most frequent). The ws is determined as follows: We start by setting $ws = 3$ and we calculate the number of peaks P_{reg} obtained. We continue to increment the ws until the the number of peaks P_{reg} obtained approach the most probable interval number I_{best} as above.

7.2.3 Step 2: Fuzzification of the Crisp Cut Points

Once the crisp cut points are determined we start the fuzzification process which is elaborated in figure 7.1 (below). Here we set the values of our fuzzy parameter set (α, β, h) where h is always set to 1. The starting point (attribute value on the x-axis) on the peak is denoted as P_{start} and the end point as P_{end}. α is the discretization or threshold cut point and is set to $\frac{P_{start}+P_{end}}{2}$ and the transition width β is set to $P_{start} + P_{end}$ and thus a fuzzy partition is build as shown in Fig 7.1 (below). This process is repeated for each peak P_{reg} of the discretization point distribution obtained.

7.2.4 Dual Split Fuzzy-Partitioning (DSF)

As discussed earlier, our fuzzy partitioning technique (DSP) makes use of both crisp and fuzzy partitions based on the level of imprecision of class data in the region around the split. Before starting to search for the best split α over an attribute X, we divide the data into two types of regions i.e crisp and fuzzy regions. This division is done by a heuristic that runs once before the actual top-down partitioning of the data space begins. This heuristic

is explained in the following sub-sections and illustrated with the help of figure 7.3. Below, we give a few definitions and notations of these two types of regions:

- Region: $\gamma = x_r, ..., x_s$, where r and s could be from 1 to a (total number of unique values) and $r \leq s$.
- Fuzzy Region: $\gamma_F = \gamma : \varphi_{unc}(\gamma) < t_{\varphi_{unc}}$, where $\varphi_{unc}(\gamma)$ is the measure of uncertainty or imprecision of a region γ (defined below) and $t_{\varphi_{unc}}$ is an uncertainty threshold which is user defined.
- Crisp Region: $\gamma_C = \gamma : \gamma \neq \gamma_F$.

7.2.4.1 Optimal Fuzzy Partitioning

First we discuss optimal ways to construct fuzzy partitions and see whether they are practical solutions or not. As far as crisp partitions are concerned, many techniques exist that try to find the optimal partitions e.g. dynamic programming algorithms like Fisher's algorithm [122] later used by [81] and [31]. This technique is described in detail in chapter 3. To the best of our knowledge no similar optimal fuzzy partitioning method exists. Thus, here we extend fishers algorithm by using a fuzzy information criteria discussed above.

An optimal fuzzy partition P with two intervals ($k = 2$) is found by finding the best split point α on the attribute X by first sorting in an ascending manner all the individuals of this attribute. After finding the split point that most improves a certain criterion (which forces searches equal to the total values of X i.e $\alpha \in [0; a]$), the process takes into consider this point known as α and searches for the best width of the fuzzy transition region β (for each α) that minimizes the criterion i.e fuzzy entropy I_f. Here, the search space of β is from zero to 2α if $\alpha < a/2$ and from zero to $2(1 - \alpha)$, if $\alpha \geq a/2$ and is a at the most.

While, finding optimal fuzzy partitions is easier in the case of two partitions, it becomes more complicated and time consuming for multiple partitions. The fuzzy discretization into two partitions is denoted by $FDisc(P^{1,k}, k = 2)$. Here, we try to extend Fisher's algorithm (presented in chapter 3) which was originally used for crisp partitions, to find optimal fuzzy partitions with k > 2 and $\beta \succ 0$. The criteria used with this algorithm is based on the star entropy measure I_f given above. Hence, if a fuzzy partition of P into k intervals P_1, P_2, P_k

is an optimal discretization of P, then a fuzzy partition of $P - P_1$ into $k-1$ intervals P_2, P_k is an optimal discretization of $P - P_1$. This interesting additive property is sufficient to adapt the dynamic programming algorithm used by [31] and other authors. We summarize below this dynamic programming algorithm.

Let $P^{i,j}$ be the substring of S consisting of the instances from i to j. $P^{1,n} = P$. Let $FDisc(P^{i,j}, k)$ be the optimal fuzzy discretization of $P^{i,j}$ into exactly k intervals.

- For each k, $1 \leq$ k $\leq$ n ,
- For each j, $1 \leq$ j $\leq$ n ,
- If k=1, $FDisc(P^{1,j}, 1) = P^{1,j}$
- If $k > 1$, $FDisc(P^{1,j}, k)$ is obtained by minimizing the star entropy value of all discretizations $FDisc(P^{1,i}, k-1)UP^{i+1,j}$ for $1 \leq$ i$\leq$ j.

7.2.4.2 A Quasi-Optimal Fuzzy Partitioning Method

This algorithm clearly is not very practical considering its quadratic time complexity. Thus, we have to look for some heuristic to obtain a quasi-optimal fuzzy partition. According to our understanding, the key for obtaining a good automatic fuzzy partition is finding the best threshold split point α and more importantly finding the most appropriate value for β. If β is too large then the partition is too vague and many branches of the fuzzy decision tree will be traversed unnecessarily and thus, increasing the complexity of the decision tree while also effecting its comprehensibility and simplicity.

Here, we point out that any fuzzy partition can contain both fuzzy and crisp splits (crisp being a special case with $\beta = 0$) as shown in fig 1. Our fuzzy multi-partitioning heuristic is based on the idea of recursive partitioning such as a top-down discretization method such as MDLPC that is discussed in chapter 3 and 4 of this thesis. Our method splits the attribute to be discretized at a threshold α with the combination of fuzzy width β that most reduces the fuzzy entropy criterion and continues recursively on the two subsets and so on until a stopping criterion is met. We adopt this technique but with significant alterations to the search and splitting mechanism. The problem with this mechanism is that the search space

for searching is very large if the number of values of the attribute X are large (which is mostly the case in data mining). Thus, we propose two modifications in this case.

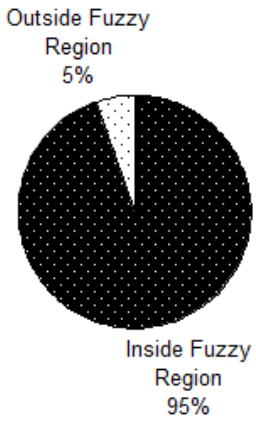

Figure 7.2: The percentage of times that β occurs within the fuzzy region.

The first one relates to the searching of α, in which we exploit the notion that the cut points are always on the boundaries which is used and validated by Fayyad et al in MDLPC [121]. Thus, we only search the boundary points for potential threshold cuts and reduce the search space. The second is concerned with reduction of the possible width of the transition region β. We argue that there is a very high probability of finding β in the region that is exhibits fuzzy characteristics i.e. highly mingled classes. We define an uncertainty criteria that measures the amount of fuzzy characteristics of a particular region and we call this region as fuzzy region and is defined above. We empirically validate this point by performing fuzzy splits on two known datasets and observing how many times the fuzzy splits performed on an attribute are actually contained within the fuzzy regions. Figure 7.2 shows that almost 95 percent of the time the split is contained within our fuzzy regions.

Our method first identifies boundary points and then the crisp and fuzzy regions and illustrated above. It then starts searching for splits on these boundary points. A fuzzy split will be favored only if the split region is within the fuzzy region and else the split is performed with $\beta = 0$ i.e crisp. Now, we try to cater to the problem of defining a region

that favors a fuzzy split and we call this region as a fuzzy region γ. Thus, we define an uncertainty criterion based on Shannon's entropy. If uncertainty of a region γ is more than a certain threshold t_{unc}, the region is termed as fuzzy γ_F. The criterion calculates the entropy for each run (region between two boundary points) divided by the total number of values.

$\varphi_{unc} = \frac{\sum_{j=1}^{p} \frac{(1+\sum_{i=1}^{k} P_i^j log P_i^j)}{2}}{a}$, where p is the number of runs and a is the number of values.

Now, the problem remains of finding the best fuzzy regions in each $X(.)$. An optimal way of finding these regions is to do an exhaustive search thats starts from v_1 and looks for the best fuzzy region γ_f in $v_2, .., v_a$ whose φ_{unc} is greater than a certain threshold t_{unc}. Thus, it selects the best γ_f from $(v_1, v_2), .., (v_1, v_a)$. Similarly, it goes to v_2 and searches in $v_3, ..v_a$ and so on. Here again, we argue that this exhaustive search is very expensive and we propose a heuristic known as dual split partitioning (DSP) to find the best partitions by searching for the best fuzzy regions and then performing a top-down split to find best partitions.

7.2.4.3 Algorithm 1: Fuzzy Region Search

1. All the examples ω are sorted by increasing $X(.)$.

2. A list of $k-1$ boundary points and k runs are identified as explained above and illustrated by figure 7.3a.

3. Define two parameters: **a)** A fuzzy window of size: $F_{ws} = \frac{a}{k}$, where a are the number of distinct values of $X(\Omega)$ and k is the number of runs. This window size gives us an idea about the separation of the class values $c_1, .., c_m$. A small size tells us that the classes are mixed (not easily separable) and vice versa. This window can also be user defined. **b)** Minimum number of values required to be contained in the fuzzy region $F_{R_{min}} = 2$ values.

4. The initial set of k runs are further refined. Starting from the initial value x_1 till $x_a - F_{ws}$, regions of window size F_{ws} are taken into account one by one. If $Region \geq \varphi$ a flag is set as $success = 1$ which is initially set to zero and the window moves to the next value of X. Else if $Region \prec \varphi$ and $success = 1$, the precedent region is set as a

refined run and the window moves to the value immediately after this run. Else if the same condition is true when $success = 0$, then the current region in set as a refined run and the window moves similarly. The result of this process is shown in figure 7.3b, where a window size of approximately 2 is applied.

5. For all runs $R_1, .., R_p$, if $R_i \leq F_{ws}$, it becomes a fuzzy region and if consecutive runs are $\leq F_{ws}$, they are added to the same region, otherwise a new region is formed further along X. Example of figure 7.3(c) shows a single fuzzy region γ_f being formed.

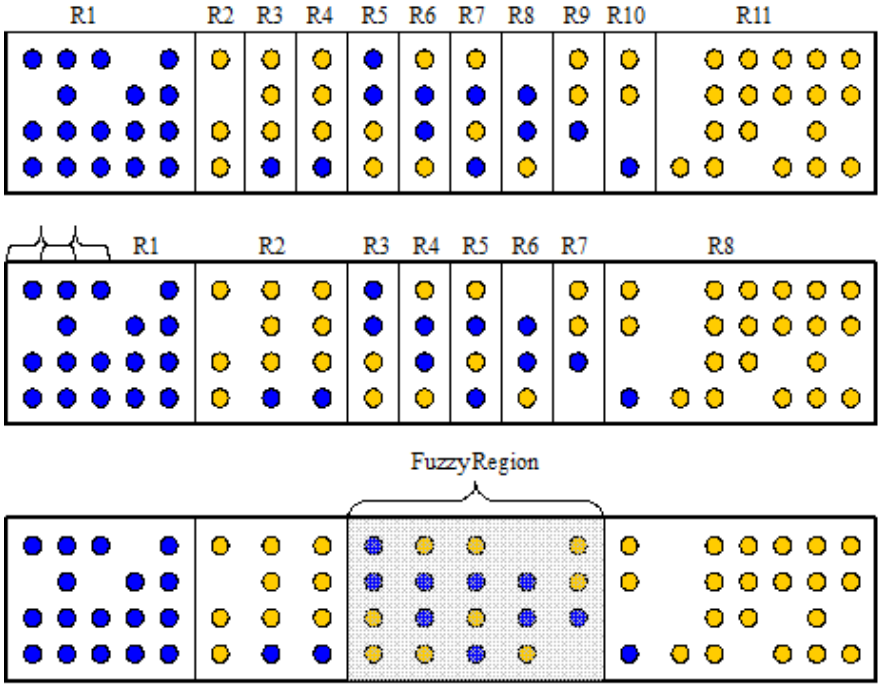

Figure 7.3: **a)** Initial set of runs. **b)**. Refined set of Runs. **c)** Forming of a single fuzzy region.

7.2.4.4 Algorithm 2: Top-Down Dual Recursive Partitioning

1. From each boundary point (situated between two runs), a binary partition is built by splitting the considered population at this point α with parameters α, β. If the split lies in a crisp region set $\beta = 0$, making the split crisp. Else if the region is

identified as fuzzy then the criterion is measured for each β which varies between $0, \alpha + boundary of \gamma_f$. The split which optimizes the fuzzy entropy criterion is then retained as the cut point. The process stops if no improvement can be done.

2. Step 2 is recursively repeated on the two new sub-populations.

At step 2, the chosen discretization point d_t^* leads to the best binary partition on Ω. Let Ω_1 and Ω_2 be the two elements of this binary partition and $m_j = card(C(\Omega_j))$ the number of different classes in the sub-sample Ω_j.

Here we define two stopping criteria to halt the top down fuzzy partitioning.

1. The partitioning stops when the criteria φ_{unc} is below a threshold t_s.

2. Or we can use the stopping criterion used by Fayyad et al [121] by modifying it to serve our purpose i.e. to use fuzzy entropy in the formula instead of Shannon's entropy. Thus, the stopping criterion becomes:

$$G_f(A_k) > \frac{log_2(N-1)}{N} + \frac{log_2(3^m-2)-[m\times I_F(X)-m_1\times I_F(X_1)-m_2\times I_F(X_2)]}{N} \text{ and is stopped otherwise.}$$

7.3 Handling of Multiple classes

The fuzzy decision tree approach is suited for having a single output class C, the result for the complementary class being deduced as:

$$\mu_C(\omega) = 1 - \mu_C(\omega)$$

This idea is taken from C.Oralu et al [21] where for problems with more than two classes, a forest of is built, each tree being dedicated to the recognition of a single class against the union of all the other classes. Suppose each tree from the forest returns a value of the membership degree to one single class for a given individual ω, $\mu_{C_i}(\omega)$, and suppose also that the costs of misclassification are independent of the type of error, then the overall elected class for the individual is the one with the maximal degree over the forest:

$$C^* = argmax_{C_i}\mu_{C_i}(\omega)$$

In the more general case where misclassification costs are not uniform, we could adapt this scheme by interpreting these class-membership degrees as (estimates of) conditional probabilities to belong to the different classes and by choosing the class that minimizes the expected misclassification cost estimated for a given example or individual.

Chapter 8

Experimentation and Results

In this chapter we perform detailed experiments on various crisp and fuzzy decision tree techniques and compare their performances with our fuzzy partitioning based decision tree methods. In doing so we provide analysis and results on each experiment done and try to deduce reasonable conclusions.

8.1 Evaluated Discretization Techniques in Brief

In addition to our fuzzy partitioning based decision tree techniques, we have evaluated 4 other crisp decision tree methods, 3 other fuzzy decision tree methods based on fuzzy partitioning and 2 decision tree techniques based on bagging/boosting which have been described in sufficient detail in the previous chapters. A summary of these methods is given below:

- **CART** [75] is a greedy top-down decision tree method that relies on binary splits. Its split evaluation criteria is the Gini impurity which is based on squared probabilities of membership for each target category in the node. **C4.5** [66] is an algorithm used to generate a decision tree developed by Ross Quinlan. It builds decision trees from a set of training data in the same way as ID3 [65], using the concept of information entropy. **C4.5(MDLPC)** [121] uses MDLPC based discretization (described in chapter 4) on continuous data and uses this discretized data to perform entropy based splits as in C4.5. Similarly, **C4.5(Chimerge)** [103] uses Chi^2 statis-

tics as their criterion for discretization of continuous data before the splits. All these above methods have been implemented by us in software R. **OC1** [106] by Murthy et al is based on an oblique classifier that uses linear combination splits to partition the data. The complete OC1 implemented algorithm is available for free at http://www.cbcb.umd.edu/ salzberg/announce-oc1.html.

- **SDF-FDT** uses a fuzzy partitioning method using fuzzy entropy as the criteria for the split threshold selection. It sets the width of the fuzzy split region equal to the standard deviation of that attribute. **BSF-FDT** uses fuzzy entropy both for the split threshold selection and the width of the fuzzy region. Both these methods use bi-partitioning. Whereas, **Optimal-FDT** searches for an optimal fuzzy partition which is based on Fisher's optimal algorithm using fuzzy entropy based criteria (described in the previous chapter). These methods have been implemented by us in R.

- **Bagging-DT** [76] performs bootstrap aggregation on the C4.5 classifier using 20 decision trees. **Boosting-DT** [102] uses Ada-Boost algorithm on 20 decision trees. Both of these have been utilized from packages provided by R. These are available for free use at 'http://cran.r-project.org/web/packages/ada/index.html' for boosting and 'http://cran.r-project.org/web/packages/rpart/index.html' for bagging.

Dataset	Attributes	Size	Classes
Adult	15	48842	2
Balance	4	625	3
Breast	10	699	2
Heart	13	270	2
Hypothyroid	25	3163	2
Iris	4	150	3
Ionosphere	34	351	2
Pima	8	768	2
Sonar	60	208	2
Waveform	21	5000	3

Figure 8.1: Data sets and their summary.

8.2 Data Sets

All the experiments have been done using the ten data sets laid out in fig 8.1. These datasets comprise of 2 or 3 classes and contain small and large sets, with Iris data having 150 examples and the largest Waveform data containing 5000 examples. Note that the waveform dataset contains significant noise as compared to the others.

8.3 Results - Analysis and Comparisons

Fuzzy decision tree methods have mainly been evaluated using other decision tree methods [96] and to the best of our knowledge they have not been really compared against other fuzzy decision tree techniques. Only in [27] C.Marsala et al compare different fuzzy entropy based criteria. While in their experiments, (C.Olaru and L.Wehenkel) [96] compare their fuzzy decision tree with other aggregation based methods like bagging and boosting in addition to crisp methods. Other than that, as we have mentioned not a work of experiments have been done in terms of comparisons among fuzzy decision tree techniques.

Although these evaluations bring many insights on the impact of fuzzy decision tree methods in dealing with imprecise and fuzzy real life situations, the contribution of the type of fuzzy partitions and the fuzzy criteria used in each fuzzy decision tree is not taken into consideration. In addition the idea of comparing the fuzzy partitions with an optimal fuzzy partitioning based technique and the idea of multi fuzzy partitions itself has not yet been evaluated.

Thus, we perform an extensive evaluation in which we compare our techniques with crisp decision trees and in addition compare among various fuzzy partitioning techniques. Among the various fuzzy partitioning techniques, we also compare with an 'optimal' fuzzy partitioning method and determine the proximity of our results. In addition we also felt an obligation to compare with some existing variance reduction and ensemble classifier techniques like bagging and boosting and to determine the placement of our technique in the existing work regarding accuracy improvement by variance reduction.

We evaluate different aspects of decision tree learning, out of which some are evaluated empirically and some aspects are discussed theoretically. Thus, we evaluate the different

decision tree based models based on the following criteria:

1. Misclassification Error: A decision rule $d(.)$ is a function that maps $\Re$ into $1, 2, ..., m$ classes with $C(X)$ representing the class label of feature vector X. The true misclassification error rate of d, denoted by $R*(d)$, is: $R*(d) = p(d(X) \neq Y)$, where p(.) denotes the probability and Y the true class.

2. Complexity: It is the number of the test nodes. In the case of multi-class problems, for all the methods, the complexities displayed are the total number of test nodes of the models over the forest.

3. Robustness: We introduce a concept of robustness. This is equal to the classification accuracy of the training sample divided by the classification accuracy of the whole population (which is known in our experiments). This measures the degree of accurate estimation of the population from a small training sample.

4. Mean Squared Error: It is the amount of error by which the estimator differs from the quantity to be estimated and we define it as $Variance + Bias^2$.

5. Time Complexity: This is the amount of time required to build and classify.

6. Comprehensibility: This is the simplicity and understandability of the built model.

Because of the instability of the learning algorithms and especially of the tree-based methods, any comparison of two different algorithms must involve multiple runs of the algorithms instead of a single run, in order to diminish the learning set variation and the algorithm internal randomness as sources of variation. That is why all the results simulated are computed as averages over a number of 10 or 20 trees. The variance coming from the test set is also diminished by choosing sufficiently large test sets for all experimented datasets.

Most of the time we used a 3-fold cross validation to measure the geometric mean of the error rate, MSE, complexity and robustness for all the data sets involved. The resulting measures and their results are explained in the following subsections along with figures 8.2 to 8.17. In our experimentations we obtained RSF-FDT technique using the MDLPC discretization obtained from $n = 100$ bootstrap samples and try to build a near optimal

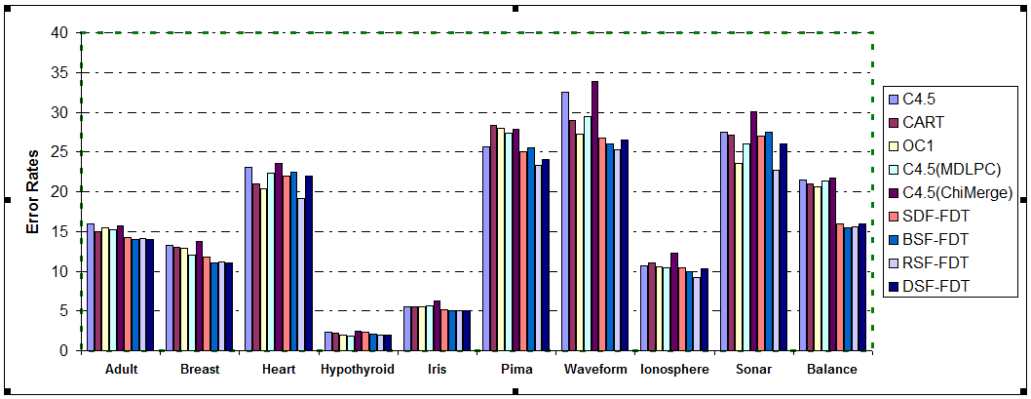

Figure 8.2: Rough Comparison of the Accuracy of all the methods for each dataset.

solution. In the case of more than two classes we build forests of decision trees and aggregate the results of all the trees.

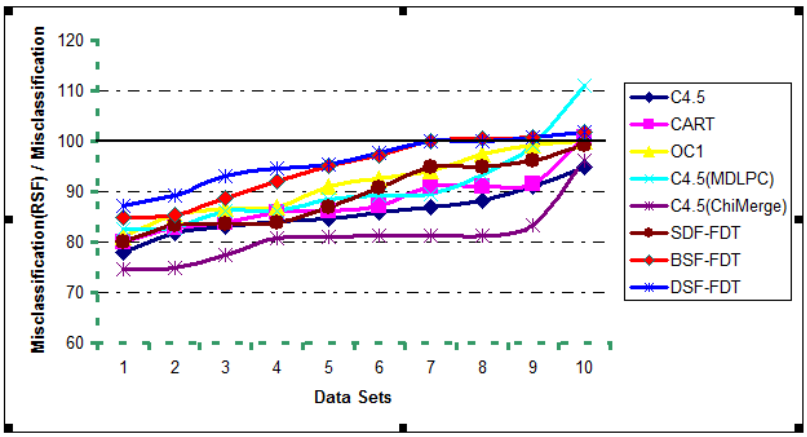

Figure 8.3: Repartition function of RSD vs all the other methods.

8.3.1 Classification Accuracy and Error Rate

We consider the prediction accuracy of any classification technique over a dataset as the most important criteria. Figure 8.2 illustrates a rough estimate of the error rates of all the methods plotted over all the data sets. The only obvious exclusion are the ensemble classifier

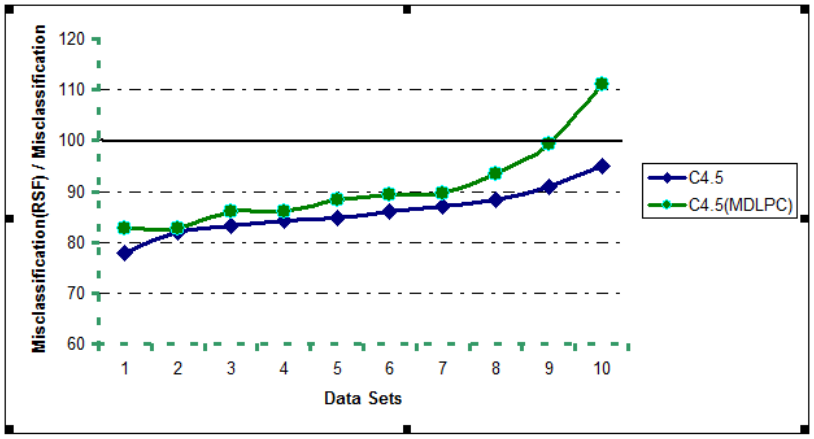

Figure 8.4: Repartition function focusing on RSD vs MDLPC and Fisher.

methods (bagging and boosting), as they are illustrated in separate graphs later on.

The error rates plotted in the graph shows that the decision trees based on fuzzy partitioning seem to perform better on almost all the data sets. Although, the databases of Hypothyroid and Iris seem to display an almost even error rates but this phenomenon could be because of the small size of these databases. Overall RSF-FDT seems to have the minimum error rate while the other three fuzzy techniques are very close to each other. C4.5(Continuous) and C4.5(Chimerge) demonstrate the highest error rates.

Figure 8.3 shows the repartition function of the relative differences of error rate between the RSF-FDT method and the other methods. Each point in this repartition function is the summary of 10 experiments performed on the same data set. While, figure 8.4 illustrates the comparison in particular between RSF-FDT, C4.5(Continuous) and C4.5 (MDLPC).

Such repartition functions have been used and explained already in chapter 5. The curves below the cutoff line (at 100) shows that the RSF-FDT method is dominated by the other methods and, and the curves above the cutoff line shows that it outperforms the other algorithms. Figure 8.3 shows that the DSF-FDT and BSF-FDT curve is almost flat for about 50 percent of the data sets having exactly the same performances that of the RSF-FDT method. However, the other 50 percent of the data sets demonstrate a higher accuracy of RSF-FDT. The four other methods i.e. SDT-FDT, CART, C4.5(MDL), C4.5(ChiMerge) methods are dominated by RSF-FDT and DSF-FDT methods throughout the data sets.

t*	RSF-FDT	C4.5	CART	OC1	C4.5 (MDLPC)	C4.5 (ChiMerge)	BSF-FDT	SDF-FDT	DSF-FDT
RSF-FDT	X	4.209238	4.700232	3.870801	4.019677302	4.799778593	3.597397192	2.320485	2.315027
C4.5		X	0.490232	-0.3392	-0.190322698	0.589778593	-0.61260281	-1.88951	-1.89497
CART			X	-0.8292	-0.680322698	0.099778593	-1.10260281	-2.37951	-2.38497
OC1				X	0.149677302	0.929778593	-0.27260281	-1.54951	-1.55497
C4.5 (MDLPC)					X	0.789778593	-0.41260281	-1.68951	-1.69497
C4.5 (ChiMerge)						X	-1.20260281	-2.47951	-2.48497
BSF-FDT							X	-1.27951	-1.28497
SDF-FDT								X	-0.00497
DSF-FDT									X

Figure 8.5: Comparison of the critical area between all the methods.

Among these methods C4.5(Chimerge) seems to perform the worst.

Figure 8.4 elaborates the curves of RSF-FDT, C4.5(Continuous) and C4.5 (MDLPC) using the same repartition function as above. As our RSF-FDT method is built by aggregating the discretization points of MDLPC, we need to illustrate the improvement in accuracy achieved by our method. The curve of both the C4.5 methods are dominated by RSF-FDT except for one data set. While using C4.5 with MDL discretization tends to improve the performance of C4.5, when used with continuous data.

8.3.1.1 Comparison Summary

Here, compare all the methods using the same statistic defined in chapter 5 for discretization. The computed t^* results are reported in figure 8.5. Positive values of t^* indicate that the method in the row is better than the method in the column. The table shows the performance of C4.5(Continuous) and C4.5 (MDLPC) methods whose results are the worst among all the methods. C4.5 (MDL), CART and SDT-FDT perform better and have relatively smaller differences. BSF-FDT and DSF-FDT improve the accuracy a lot more where DSF-FDT seems to outperform the other by a small margin. While, RSF-FDT significantly, report much better results.

8.3.1.2 Quasi-Optimal Fuzzy Partitioning

We would like to see how far away our solution lies from the optimal solution. For this purpose, we apply the Fisher's optimal fuzzy partitioning algorithm based on fuzzy entropy

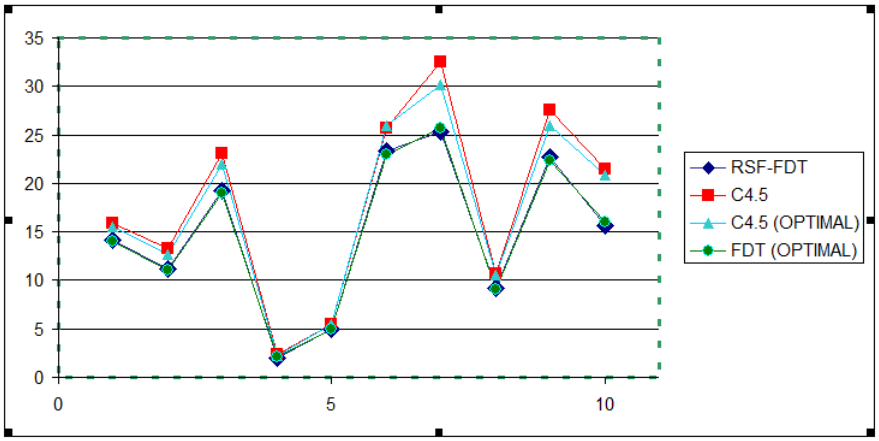

Figure 8.6: Comparison of RSF-FDT with the optimal algorithm.

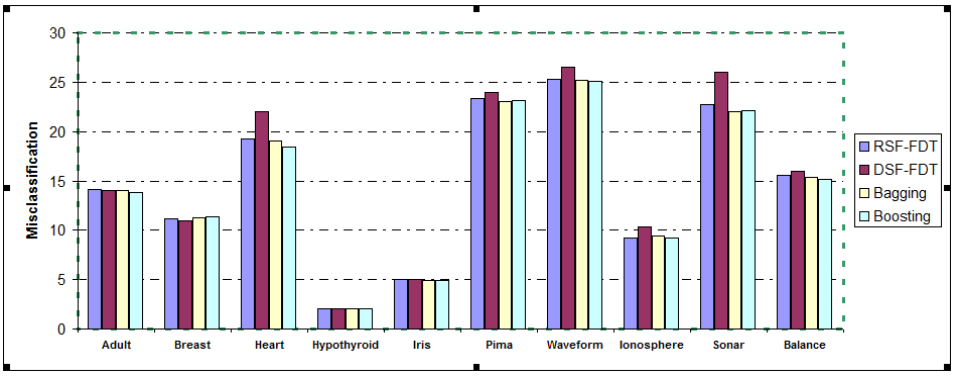

Figure 8.7: Comparison of accuracy with ensemble tree methods.

(described in chapter 7) on each dataset. Figure 8.6 plots the error rates for each data set for RSF-FDT, C4.5(Continuous), C4.5(Optimal) and FDT(Optimal). Here C4.5(Optimal) builds C4.5 classifier on data discretized by the optimal Fisher algorithm's implementation of the FUSINTER criterion (as discussed in chapter 4) and FDT(Optimal) is the fuzzy decision tree built by using Fisher's optimal fuzzy partitioning algorithm of chapter 7. The plot in figure 8.6 shows that the points of RSF-FDT and FDT(Optimal) almost overlap except for two data sets where its accuracy is slightly lesser. While, it performs significantly better than C4.5(Optimal) for 5 data sets.

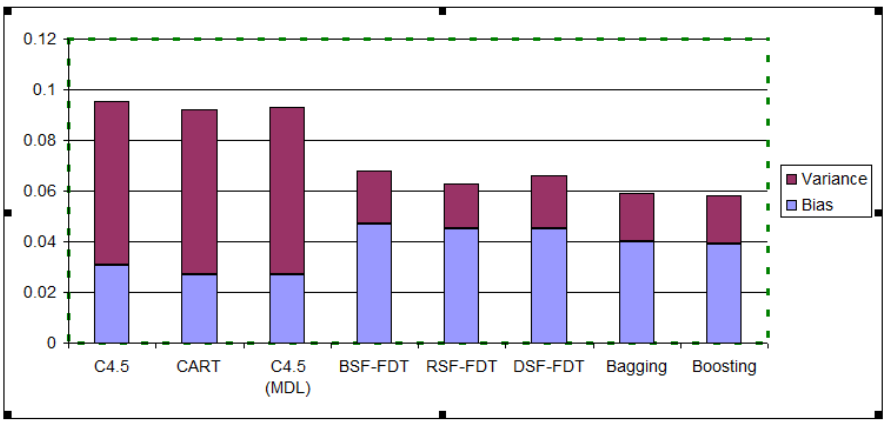

Figure 8.8: Bias/variance decomposition of all the methods.

8.3.1.3 Comparison with Ensemble Tree Learners

We feel the need to compare our fuzzy partitioning based decision tree with ensemble tree learning methods such as bagging and boosting. These techniques have been presented in detail in chapter 3. Figure 8.7 demonstrates the error rates of these methods compared with our techniques over all the data sets. The result does not show any significant difference in error rates except for DSF-FDT that has the highest error rate. In Breast and Waveform data sets RSF-FDT even marginally outperforms both bagging and boosting.

8.3.1.4 MSE Evolution and Decomposition

In figure 5.8 we illustrate the MSE and bias/variance decomposition of the chosen methods from all the 30 trials, comprising of the geometric mean of all the datasets. It demonstrates a small difference in bias between the crisp methods but they display high variance. On the other hand the fuzzy decision tree and ensemble methods shown higher bias but very small variance and an overall much smaller MSE. RSF-FDT, bagging and boosting have the smallest variance out of all the methods while the overall MSE is minimum for the ensemble learning methods.

Furthermore, in figure 5.9 we compare the MSE decomposition of C4.5 using MDLPC discretization and RSF-FDT over all data sets. Notice the decreased variance of the fuzzy

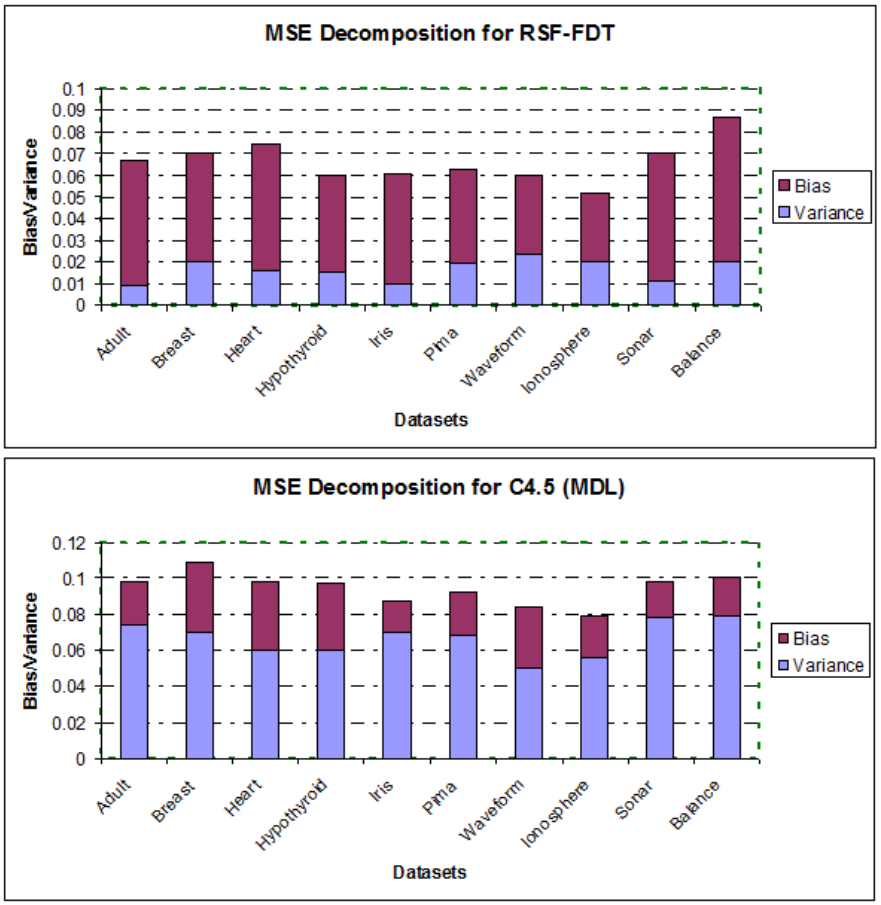

Figure 8.9: Bias/variance decomposition comparison with ensemble tree methods.

counterpart and thus, a much decreased MSE.

8.3.1.5 Other Factors Effecting Accuracy of RSF-FDT

In this subsection we discuss in detail the other factors that effect the accuracy of our fuzzy partitioning based decision tree classifier. Figure 8.10 shows us a steady but a small increase in accuracy of our RSD technique with respect to the sample size measured over the waveform dataset. However, the increase in accuracy is significant until a sample size of 1500 examples but after that the increase becomes smaller and the curve stabilizes.

Since, RSF-FDT technique is based upon bootstrap sample, we study the effect of the number of bootstrap samples generated on the prediction accuracy. Figure 8.11 demonstrates

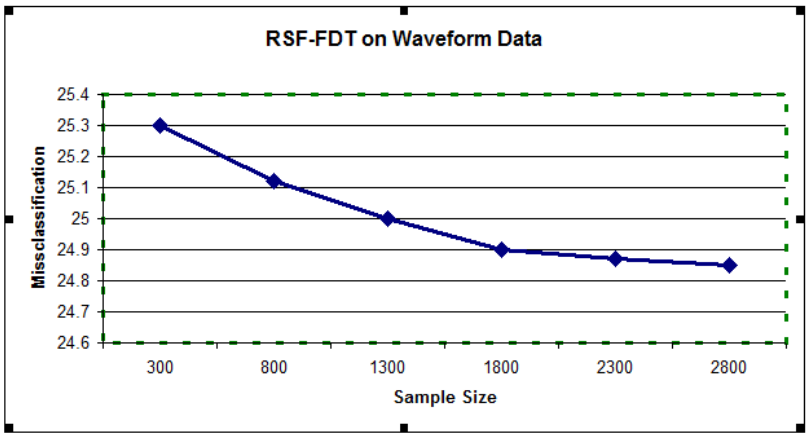

Figure 8.10: Evolution of error rate with increasing sample size.

this effect on the mean accuracy from all the data sets. There is a gradual improvement in accuracy but the increase from 100 to 1000 bootstrap samples is just about 0.6 percent.

As discussed in previous chapters, the RSF-FDT technique uses a moving average filter of size ws to smoothen the discretization point distribution curve, the size of ws does have an effect on the fuzzy partition and thus, on the accuracy of the overall classifier. Figure 8.12 illustrates this effect on the accuracy and complexity of the fuzzy decision tree on the waveform data set. The curve of figure 8.12(top) shows an increase in accuracy with an increase in window size, but then it reaches an optimum window size. After this point a further increase in window size starts decreasing the accuracy. While, the curve of figure 8.12(bottom) illustrates a gradual decrease in the complexity of the decision tree with increase in the window size.

8.3.2 Complexity

The complexity of a decision tree play a vital role in the performance of a decision tree classifier. Figure 8.13 shows the repartition function of the relative differences of the complexity between the RSF-FDT method and the other methods. The curves above the cutoff line (at 100) shows that the RSF-FDT method is dominated by the other methods and, and the curves below the cutoff line shows that it outperforms the other algorithms. RSF-FDT dominates throughout the other fuzzy decision tree methods but is mostly inferior to all the

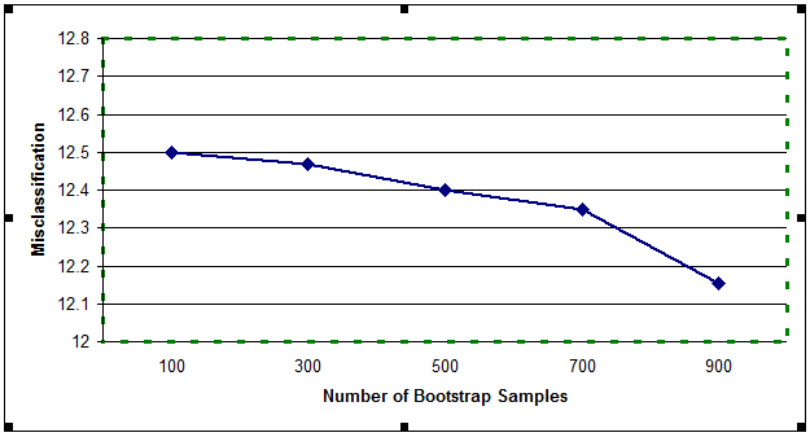

Figure 8.11: Evolution of error rate with increasing bootstrap samples.

crisp decision tree methods. However, we have a flat curve for 5 data sets for all the crisp methods indicating that the performance was not significantly better for those data sets. OC1 performed the best in terms of complexity.

Figure 8.14 demonstrates the MSE evolution and the bias/variance decomposition as a function of increasing complexity. This idea has been taken from [21] where the authors perform similar type of experiments on their FDT. This is done by introducing a strict stopping rule based on the number of nodes. We compare C4.5 method against its counterpart RSF-FDT on the waveform data set. The crisp C4.5 classifier demonstrates the classic bias/variance trade-off with the increased model complexity. After a certain complexity a threshold point is reached which is the optimal point and where the MSE is minimal. A contrast is observed in its counterpart fuzzy classifier where the increased complexity gradually decreasing the MSE and the bias. As far as the variance is concerned, it almost has no effect on the increase complexity which significantly increases the performance of this classifier.

8.3.2.1 Bi-criteria Evaluation

In this sub-section we perform the similar bi-criteria evaluation as discussed in chapter 5. Accuracy is certainly the most important parameter to distinguish a classifier so we have grouped it in both the analysis of figure 8.15 and 8.17. Figure 8.15 reveals a significant

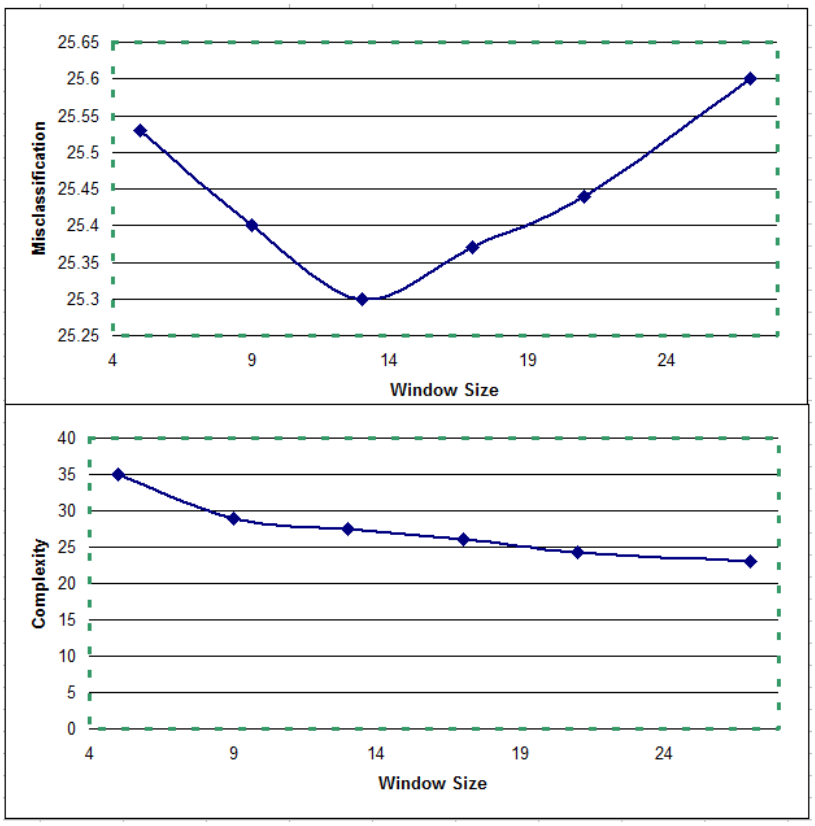

Figure 8.12: Evolution of error rate with increasing window size.

distance between the fuzzy and crisp decision tree methods where the former lie in the top-left corner with high complexities and high accuracies while, the crisp decision tree methods in the bottom-right corner with low complexities and lower accuracies. Among the former methods RSF-FDT has the best performance while, among the other group OC1 and C4.5(MDL) stands out.

8.3.3 Robustness

The robustness is an interesting criterion that allows to estimate whether the performance on the sample train data is a good prediction of the performance on the whole data set (Population). The higher is the robustness, the most accurate will be a ranking of attributes based on their sample accuracy. This can be critical in the case of classifiers such as decision

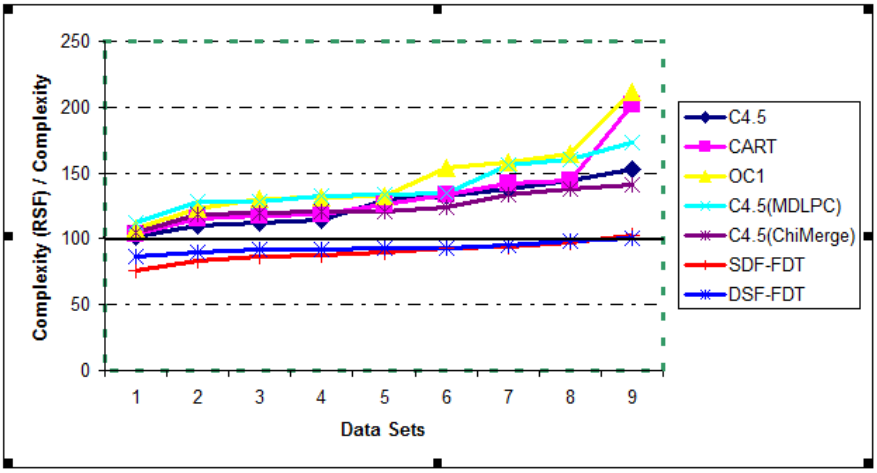

Figure 8.13: Comparison of the complexity between all the methods.

trees that incorporate an attribute selection method to build the next node of the tree. Figure 8.16 plots the curves of the mean robustness for each dataset for all the methods. It can be seen that both our techniques seem to exhibit high robustness in almost all the data sets while overall the fuzzy decision tree methods demonstrate high robustness as compared to the crisp methods.

8.3.3.1 Bi-criteria Evaluation

To sum up this criteria we do a bi-criteria analysis of robustness with accuracy which is shown in figure 8.17. It shows our RSF-FDT method in the top-left corner with the highest robustness and highest accuracy while, in contrast the C4.5 method showing the lowest. The other methods lie in between with our DSF-FDT method standing out of the rest.

8.3.4 Time Complexity

The evaluated soft decision tree methods have the same computational complexity of $O(nlog(n))$. They first sort the attribute values and identify boundary instances in a preprocessing step. The time efficiency of the methods mainly relies on the search direction (top-down or bottom-up), the simplicity of the mathematical criterion. The optimal algorithm has a complexity of $O(N^3)$ while RDD and RSD depend on the number of bootstrap samples B in addi-

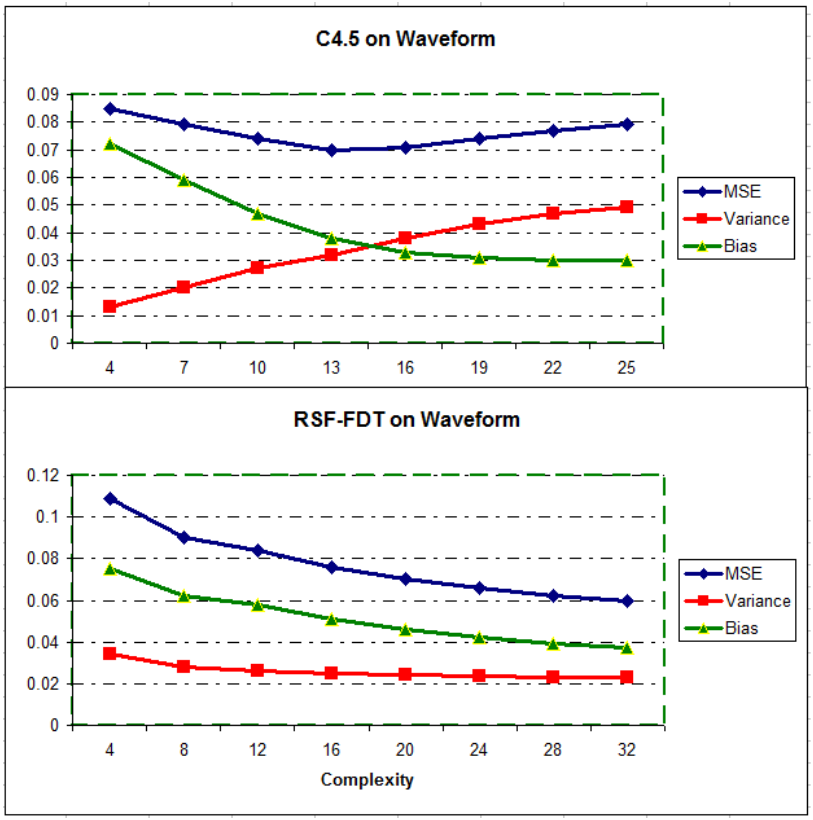

Figure 8.14: Evolution of MSE with increasing complexity.

tion to small post-processing that includes the extraction of discretization points from the distribution. Thus, their complexity becomes $B \times O(nlog(n))$. In the case of TBTD an additional cost is added which consists of generating B bootstrap samples at each node instead of generating them once for the sample of the root node (as in RSD and RDD). While, the complexity of DBD is almost linear $B \times O(N)$. Thus, the comparison of the time complexities is as follows:

$$DBD < (MDLPC < Bgain < Chimerge < Modl) < RDD < RSD < TBTD < Fisher$$

Although here, we argue that there is a trade-off between time complexity and quality. But with vast improvements in computing speeds, we argue that quality could be a much valuable commodity.

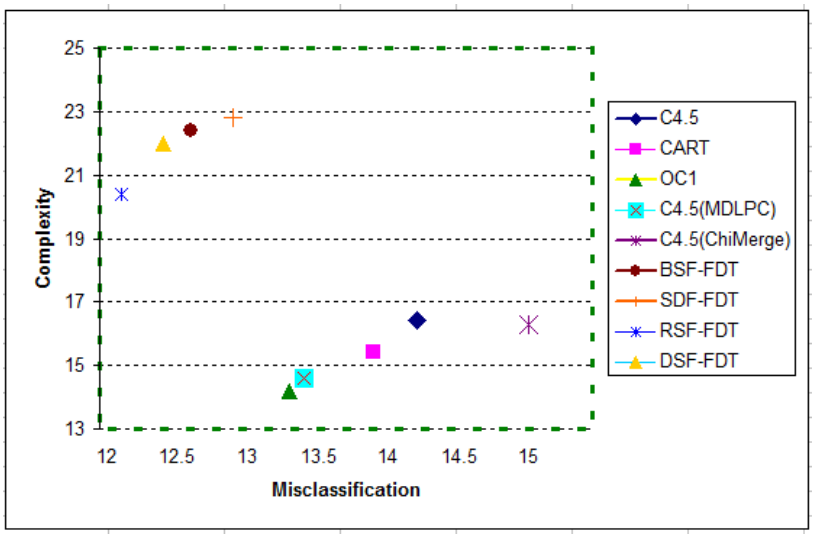

Figure 8.15: Bi-criteria evaluation of the methods for the accuracy and complexity.

The complexity of decision tree growing is upper bounded by $O(A.T.N.log(N))$, where T is the complete tree complexity, N is the number of growing instances and A is the number of candidate attributes. We say it is bounded due to the fact that in the worst case all the objects are propagated through all the test nodes. This is the case where no preprocessing of continuous attributes is performed. If a preprocessing such as discretization is performed a priori on the continuous attributes. Then the values of the search set decrease depending on the particular discretization algorithm. This new search space becomes N_d and the complexity becomes $O(A.T.N_d.log(N_d))$. The discretization itself e.g. MDLPC in bounded by $O(nlog(n))$ and in case of our resampling based technique it is $B \times O(nlog(n))$.

Wherease, while measuring the complexity of a fuzzy decision tree we have to add the complexity of determining the width β to the complexity of the decision tree i.e. $O(A.T.N.log(N) + T.T_\beta)$. The calculation for T_β varies in the different fuzzy techniques from searching all of the candidate values (BSF) to searching only in specific regions as in DSF.

Thus, the comparison of the time complexities is as follows:

$$(C4.5 < CART) < (SDF - FDT < DSF - FDT < BSF - FDT) < RSF - FDT < OC1 < Optimal - FDT$$

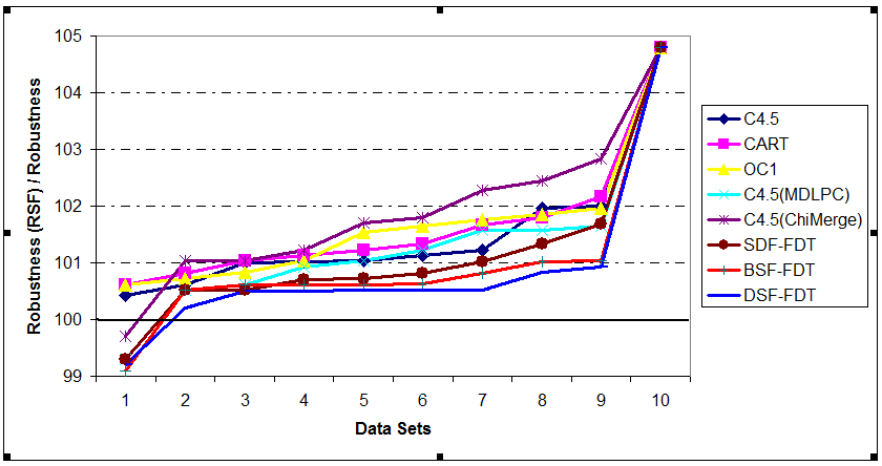

Figure 8.16: Comparison of the robustness between all the methods.

In the case of Bagging and Boosting, they depend heavily on the number of bootstrap samples.

8.3.5 Interpretability

From the point of view of interpretability, all standard decision trees are the most interpretable, since they are the less complex. The aggregation methods (bagging, boosting) are not at all interpretable methods, since they are based integrally on averaging. However, fuzzy decision trees are also easily interpretable because they translate the final classifications into simple decision rules.

8.4 Conclusion of Part II

We presented two fuzzy decision tree techniques that use resampling based fuzzy partitioning and a top-down recursive fuzzy partitioning respectively, during the tree building process. The aim of resampling is to build a discretization point distribution from which close to optimal discretization points can be obtained. We identify regions of high probability of finding discretization points which contribute in building fuzzy partitions during the tree building process. In the later method, we use fuzzy partitions only when necessary i.e. when

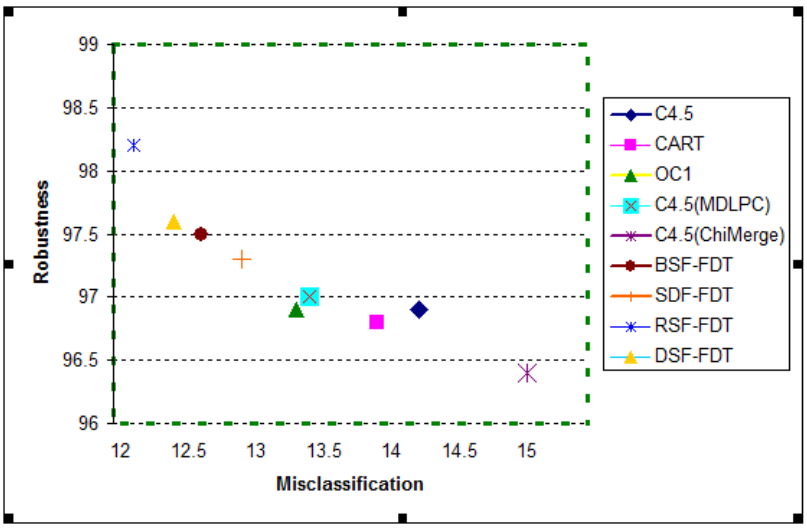

Figure 8.17: Bi-criteria evaluation of the methods for the accuracy and robustness.

the region around the cut point is vague and fuzzy. We justify the use of a heuristic to build fuzzy partitions by showing the high time complexity of optimal partitioning solutions.

We show by comparing with other classical and fuzzy decision tree methods that our fuzzy decision trees produced better accuracy with a greater ability to cater for large data noise and variance. However, the only trade-off is that of time complexity. We also show that our solution is comparable to aggregation techniques such as Bagging and Boosting.