Georgia State University
Digital Archive @ GSU

Computer Science Dissertations | Department of Computer Science

12-2-2009

Integrating Information Theory Measures and a Novel Rule-Set-Reduction Tech-nique to Improve Fuzzy Decision Tree Induction Algorithms

Nael Mohammed Abu-halaweh
Georgia State University, nabuhalaweh1@student.gsu.edu

Follow this and additional works at: http://digitalarchive.gsu.edu/cs_diss

Recommended Citation
Abu-halaweh, Nael Mohammed, "Integrating Information Theory Measures and a Novel Rule-Set-Reduction Tech-nique to Improve Fuzzy Decision Tree Induction Algorithms" (2009). *Computer Science Dissertations*. Paper 48.

This Dissertation is brought to you for free and open access by the Department of Computer Science at Digital Archive @ GSU. It has been accepted for inclusion in Computer Science Dissertations by an authorized administrator of Digital Archive @ GSU. For more information, please contact digitalarchive@gsu.edu.

INTEGRATING INFORMATION THEORY MEASURES AND A NOVEL RULE-SET-REDUCTION TECHNIQUE TO IMPROVE FUZZY DECISION TREE INDUCTION ALGORITHMS

by

NA'EL ABU-HALAWEH

Under the Direction of Dr. Robert W. Harrison

ABSTRACT

Machine learning approaches have been successfully applied to many classification and prediction problems. One of the most popular machine learning approaches is decision trees. A main advantage of decision trees is the clarity of the decision model they produce. The ID3 algorithm proposed by Quinlan forms the basis for many of the decision trees' application. Trees produced by ID3 are sensitive to small perturbations in training data. To overcome this problem and to handle data uncertainties and spurious precision in data, fuzzy ID3 integrated fuzzy set theory and ideas from fuzzy logic with ID3. Several fuzzy decision trees algorithms and tools exist. However, existing tools are slow, produce a large number of

rules and/or lack the support for automatic fuzzification of input data. These limitations make those tools unsuitable for a variety of applications including those with many features and real time ones such as intrusion detection. In addition, the large number of rules produced by these tools renders the generated decision model un-interpretable. In this research work, we proposed an improved version of the fuzzy ID3 algorithm. We also introduced a new method for reducing the number of fuzzy rules generated by Fuzzy ID3. In addition we applied fuzzy decision trees to the classification of real and pseudo microRNA precursors. Our experimental results showed that our improved fuzzy ID3 can achieve better classification accuracy and is more efficient than the original fuzzy ID3 algorithm, and that fuzzy decision trees can outperform several existing machine learning algorithms on a wide variety of datasets. In addition our experiments showed that our developed fuzzy rule reduction method resulted in a significant reduction in the number of produced rules, consequently, improving the produced decision model comprehensibility and reducing the fuzzy decision tree execution time. This reduction in the number of rules was accompanied with a slight improvement in the classification accuracy of the resulting fuzzy decision tree. In addition, when applied to the microRNA prediction problem, fuzzy decision tree achieved better results than other machine learning approaches applied to the same problem including Random Forest, C4.5, SVM and Knn.

INDEX WORDS: Decision tree, Fuzzy, ID3, Fuzzy ID3, FID3, Classification, Prediction, Rule-set reduction, MicroRNA, Pre-microRNA, Machine learning.

INTEGRATING INFORMATION THEORY MEASURES AND A NOVEL RULE-SET-REDUCTION TECHNIQUE TO IMPROVE FUZZY DECISION TREE INDUCTION ALGORITHMS

by

NA'EL M. ABU-HALAWEH

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2010

Copyright by
Na’el Mohammed Abu-halaweh
2010

INTEGRATING INFORMATION THEORY MEASURES AND A NOVEL RULE-SET-REDUCTION TECHNIQUE TO IMPROVE FUZZY DECISION TREE INDUCTION ALGORITHMS

by

NA'EL ABU-HALAWEH

Committee Chair: Robert W. Harrison

Committee: Yanqing Zhang

Yi Pan

Yichuan Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2010

DEDICATION

To My Parents Mohammad Abu-halaweh and Somaya Abu-halaweh,

To my Wife Saeda Mustafa,

To my Brothers Tawfiq, Zaki, Naser and Abdel-jabbar Abu-halaweh,

and to my Children, Mohammad, Lana and Somaya

ACKNOWLEDGEMENTS

I would like to thank my Advisor Dr. Robert W. Harrison, a leading scientist in Machine Learning and Bioinformatics, for providing me the opportunity to work on one of the state of art machine learning techniques “Fuzzy Decsion Trees”. His support, guidance, encouragement and invaluable ideas led me to great achievements. I would also like to thank Dr. Yi Pan, a leading scientist in many fields including Bioinformatics and Machine Learning and Chairman of the Computer Science Department, Dr. Yanqing Zhang, a leading scientist in Data Mining, Computational Intelligence and Machine Learning, and Dr. Yichuan Zhao, a leading scientist in statistics and statistical modeling of fuzzy systems, for their complementary scientific expertise, support and guidance throughout this research and for their invaluable advices and critical reading of this dissertation.

I am very thankful to the Computer Science Department at the Georgia State University (GSU), the GSU Molecular Basis of Disease Initiative (MBD), and the Georgia Cancer Coalition for supporting this research work.

I would like to thank Dr. Raj Sunderraman for his support and invaluable advice. I would like also to thank my colleagues in Dr. Harrison’s lab for the useful discussions we have, especially, Mr. Amit Sabnis.

Finally, I would like to give my deep appreciation to my family, including my parents Mohammad and Somaya Abu-halaweh, my wife Saeda Mustafa, and my brothers for their support and encouragement.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS v

LIST OF TABLES ix

LIST OF FIGURES x

Chapter 1: INTRODUCTION.....1

Chapter 2: BACKGROUND AND LITERATURE REVIEW.....7

2.1 Introduction.....7

2.2 Literature Review.....9

2.3 Information Theory Measures.....13

2.3.1 Information Gain.....14

2.3.2 Classification Ambiguity.....17

2.3.3 Gini Index.....19

2.4 Interactive Dichotomizer 3 (ID3) Algorithm.....21

2.5 Fuzzy ID3 Algorithm.....22

2.6 Inference in Fuzzy ID3.....26

Chapter 3: IMPROVED FUZZY ID3 ALGORITHM.....27

3.1 Improved Fuzzy ID3 Algorithm (IFID3).....28

3.2 Experimental Details.....30

3.2.1 Datasets.....30

3.2.2 Datasets Preprocessing.....31

3.2.3 Data Fuzzification and Fuzzy Membership Functions.....31

3.2.4 Experimental Setup and Results.....40

3.3 Conclusion and Future Work.....44

Chapter 4: RULE SET REDUCTION IN FUZZY DECISION TREES, EXTENDED IFID3 ALGORITHM.....46

4.1 Introduction and Background.....47

4.2 Extended IFID3 Algorithm.....48

4.2.1 A new Syntax for Fuzzy Rules.....48

4.2.2 Object Propagation Threshold.....49

4.2.3 Extended IFID3 Algorithm.....51

4.3 Experiments and Results.....54

4.4 Summary and Conclusion.....57

Chapter 5: PREDICTION AND CLASSIFICATION OF REAL AND PSEUDO MICRORNA PRECURSORS (PRE-MIRNA) VIA DATA FUZZIFICATION AND FUZZY DECISION TREES.....59

5.1 Introduction and Background.....60

5.2 Encoding the Pre-miRNA Hairpin Structure.....64

5.3 Algorithm for Classifying Real and Pseudo pre-miRNAs.....66

5.4 Experiments and Results.....69

5.4.1 Datasets.....69

5.4.2 Parameter Optimization and Fuzzy Model Selection.....70

5.4.3 Performance Evaluation Measures.....71

5.4.4 Experimental Results.....72

5.4.5 Discussion.....76

5.5 Conclusions and Future Work.....78

Chapter 6: IDENTIFYING ESSENTIAL FEATURES FOR THE CLASSIFICATION OF REAL AND PSEUDO MICRORNA PRECURSORS USING FUZZY DECISION TREES.....80

6.1 Extended Triplet Elements Sliding Window Encoding Scheme.....81

6.2 Method.....82

6.3 Experiments and Results.....83

6.3.1 Datasets.....83

6.3.2 Parameter Optimization.....84

6.3.3 Results and Discussion.....84

6.4 Conclusions.....89

Chapter 7: FUZZY DECISION TREE SOFTWARE TOOL (FDT).....92

7.1 Introduction.....92

7.2 FDT Features.....92

7.3 File Format.....95

7.3.1 Dataset File Format.....95

7.3.2 Fuzzy Parameters File Format.....96

7.3.3 Cluster-Centers' File Format.....98

7.3.4 Fuzzy Rules File Format.....98

7.3.5 Example.....99

7.4 Summary and Future Work.....103

Chapter 8: SUMMARY, CONCLUSIONS AND FUTURE WORK.....104

8.1 Summary and Conclusions.....104

8.2 Future Work.....106

REFERENCES.....111

LIST OF TABLES

Table 3.1 Dataset Description 31
Table 3.2 Accuracy Achieved by IFID3 Versus Acuuracy Achived by Other Single-Tree Methods 41
Table 3.3 Accuracy of IFID3 Versus Accuracy of Random Forests (RF) 42
Table 4.1 A BreifDesctrition of Datasets Used During The Experiments 54
Table 4.2 IFID3 and Extended IFID3 Performance Comparison Based on The Nymber of Produced Rules and Accuracy 55
Table 4.3 Execution Times (In Seconds) for Extended IFID3 Algorithm and IFID3 Algorithm 57
Table 5.1 Summary of Datasets 70
Table 5.2 Overall Accuracy Achieved by Fuzzy-Decision-Tree Methotd Compared to Those Reported in the Litrature. Datasets 2 Through 16 73
Table 5.3 Fuzzy Decision Trees Versus Other Machine Learning Approaches. Datasets 2 Through 5 73
Table 5.4 Accuracy Achieved by FDT on Indivisual Human Datasets 2 Through 5 74
Table 5.5 Fuzzy Decision Trees Versus Other Machine Learning Approaches. Other Species: Datasets 6 Through 16 74
Table 5.6 Accuracy Achieved by FDT on Individual Datasets 6 Through 16 75
Table 6.1 Summary of Datasets 83
Table 6.2 Classification Results Achieved by FDT Using the Global Features Combinerd with The Local Features Extracted Using the Window Size Specified in The Table 85
Table 6.3 Classification Results of Different Machine Learning Tools When Applied to Datasets with Local Features Extracted Using The Sliding Window Scheme 86
Table 6.4 Classification Results Comparison: Local Features, Global Features and Combined Global and Local Features Otained Using the Triplet Elements 87
Table 6.5 Comapring Classification Results: The Six Global Features Selected By FDT, Global Features, and Global Features Combined with The Local Features Encoded Using The Triplet Elements 88

LIST OF FIGURES

Figure 3.1 Trapezoidal Fuzzy Membership Functions For the Feature in The Example 34
Figure 3.2 Triangular Fuzzy Membership Functions For The Feature in The Example 37
Figure 3.3 Calculated Triangular Fuzzy Membership Functions Using {2, 5, 8, 11} as Cluster Centers 38
Figure 3.4 Calculated Trapezoidal Fuzzy Membership Functions Using {2, 5, 8, 11} as Cluster Centers 40
Figure 3.5 Summary of Experintal Results, Accuracy of IFID3 Versus Other Methods That Produce A Single Decision Tree 42
Figure 3.6 Confusion Matrices 44
Figure 4.1 The Effect of Using The Object Propagation Threshold On Fuzzy Membership Functions 50
Figure 4.2 Comparing The Number of Fuzzy Of Fuzzy Rules Produced By EIFID3 To Those Produced By the IFID3 Algorithm 56
Figure 5.1 MicroRNA Biogenesis 61
Figure 5.2 Triplet Elements Encoding Scheme 66
Figure 5.3 Flowchart for Classification of Real and Pseudo pre-miRNA Using FDT 68
Figure 6.1 Sliding Window Scheme for a Window Size of 5 82
Figure 7.1 Sample Dataset File 96
Figure 7.2 Sample Fuzzy Parameters File with Associated Fuzzy Membership Functions 97
Figure 7.3 Sample Clusters' Centers File 99
Figure 7.4 Sample Rules' File 99
Figure 7.5 Example Training Dataset 100
Figure 7.6 Example Testing Dataset File 100
Figure 7.7 Fuzzy Parameters File 101
Figure 7.8 Constructed Fuzzy Decision Tree 101
Figure 7.9 Generated Fuzzy Rule File 102
Figure 7.10 Rule File Using the Rule Syntax Presented in Chapter 4 102

Chapter 1

INTRODUCTION

In our daily life we always face situations where we have to make decisions. We often use our past experience to decide on current events. Past experience can be thought of as the experimental data. Some applications are very complex such that it is very hard for us to deduce good decision models based on our experience. Furthermore, experimental approaches to decide on new cases may be too expensive and time-consuming. Machine learning represents an efficient and an automated approach to construct decision models from previously collected data (known cases) and apply the constructed models to unknown newly encountered similar cases to make a decision.

In the past, machine learning approaches have been successfully applied to many decision support problems including classification and prediction problems. Machine learning approaches utilize existing known cases to build a decision model that can be used to decide about unknown cases and thus saving time and reducing the experimental costs. Many machine learning approaches exist, these include neural networks, Bayesian learning, decision trees and others. This dissertation focuses on decision trees specifically fuzzy decision ones as a classification and prediction tool.

Decision trees are one of the most popular machine-learning techniques [3] [4] [5] [6] [29]; they are praised for their ability to represent the decision support information in a human comprehensible form [3] [8], however, they are recognized as a highly unstable classifier with respect to small changes in training data [28].

One of the most popular algorithms for building decision trees is Interactive Dichotomizer 3 **(**ID3) [12]. Generally trees produced by ID3 - known as crisp decision trees- are sensitive to small changes in feature values, cannot handle data uncertainties caused by noise and/or measurement errors [2] [3]. To overcome these problems, the approximate reasoning offered by fuzzy representation was coupled with ID3 [3]. The result of this coupling is the Fuzzy ID3 algorithm [1] [2] [3]. The trees produced by the fuzzy ID3 - known as fuzzy decision trees (FDTs) - are more immune to data uncertainties caused by measurement errors, noise, missing and/or inconsistent information.

In the past, many researchers studied fuzzy decision trees [1] [2] [3] [4] [6] [7] [10]. In most cases the researchers concentrated on designing a good fuzzy representation of the target problem rather than on improving the quality and the performance of produced fuzzy decision tree. Furthermore, in most of the work addressing fuzzy decision trees, the proposed methods were applied to one or few small hypothetical examples. This limits the validity of these methods and the generality of the results obtained. Published fuzzy decision tree freeware tools take too long (3–4 weeks) to produce results in relatively large dataset (about 400 features), which limit the applicability of these tools to many of the classification applications such as

those arising in bioinformatics and security. In addition, since these tools lack the support for the automatic generation of fuzzy membership functions, they require intensive user involvement in determining the fuzzy models used to fuzzify collected data. Determining the fuzzy model is a very challenging task, it requires a high degree of experience and an excellent knowledge in the target application area. In addition, some systems are very complex such that it is hard, even for an expert in the application field, to come up with a good fuzzy representation. In addition to being slow and lacking the support for the automatic fuzzification of input data, fuzzy decision tree applications produce a large number of rules. This sacrifices the interpretability of the produced decision models losing one of the main advantages of decision trees.

This research work focuses on finding solutions for the fuzzy decision tree tools' shortcomings mentioned above. Our research questions are: Can we improve the performance (both accuracy and running time) of fuzzy decision trees? Can we reduce the number of fuzzy rules produced by the Fuzzy ID3 algorithm? Can we automate the generation of fuzzy membership functions? Can we generalize the decision models produced by fuzzy decision trees in other words can we transfer decision trees from being weak learners to strong learners? Our main motivations for studying decision trees are:

- Decision trees produce a human-interpretable decision model. Understanding the decision model make it possible to verify its validity and enable a better understanding of the problem. In addition it helps in identifying the critical attributes needed for making the decision.

- Collected data always contain uncertainties due to measurement errors and/or noise. Fuzzy sets were proposed to deal with such uncertainties.
- Huge amounts of experimentally collected data are available. Fuzzy decision trees represent a data-driven approach to extract fuzzy rules from previously collected data. This can be of great importance to the understanding of the problems and for designing fuzzy logic controllers for complex systems where it is hard for a human being even an expert in the target field to come up with a set of rules that map the system behavior.

A successful approach should improve both the accuracy and the running time of the fuzzy decision tree. It should also improve the human comprehensibility of the produced fuzzy decision model by reducing significantly the number of generated fuzzy rules. In addition, a successful approach should reduce the user involvement by automating the generation of the fuzzy membership functions.

In this dissertation, we proposed an improved FID3 algorithm (IFID3), the IFID3 integrates classification ambiguity [2] and fuzzy information gain [1] to select the branching attribute. The IFID3 algorithm outperformed the existing FID3 algorithm on a wide range of datasets. We also introduced an extended version of the IFID3 algorithm (EIFD3). EIFID3 extends IFID3 by introducing a new threshold on the membership value of a data instance to propagate down the decision tree from a parent node to any of its children nodes during the tree construction phase. Using the new threshold resulted in a significant decrease in the number of rules produced, an improved accuracy and a huge reduction in execution time. We also automated

the generation of the membership functions. To do this, we used two simple approaches, in the first approach the ranges of all numerical features in a dataset are divided evenly into an equal number of fuzzy sets. In the second one, the dataset is clustered and the resulted cluster centers are used to generate fuzzy membership functions. Finally we applied fuzzy decision trees to the microRNA prediction problem, our results showed that fuzzy decision trees achieved a better accuracy than other machine learning approaches such as Support Vector Machines (SVM) [30] and Random Forest (RF) [31]. Currently, we are working on tuning and optimizing the automatically generated fuzzy membership and designing a fuzzy decision forest algorithm. In addition, we are investigating fuzzy rule set reduction further.

The rest of this dissertation is organized as follows: In Chapter 2, we will review the literature, present background information about decision trees and discuss the most common information theory measures utilized by the different decision tree algorithms. In chapter 3, we present the improved Fuzzy ID3 algorithm (IFID3). The improved FID3 integrates fuzzy information gain and classification ambiguity to select the branching feature. The chapter discusses our experimental results and compares the performance of IFID3 with that of FID3 and other decision-tree-based classification algorithms. In chapter 4, we introduce an extended version of IFID3. The new version utilizes a new threshold on the membership of a data instance to propagate down the tree from a parent node to any of its child nodes during the tree building step. The chapter compares the results achieved by EIFID3 with those achieved by the IFID3 algorithm. These results show that EIFID3 produces less fuzzy rules and achieves better accuracy than IFID3. In chapter 5, we apply the extended IFID3 to the classification and prediction of real

and pseudo microRNAs precursors (pre-microRNAs) and compare our results with those of achieved by other machine learning tools reported in the literature. In chapter 6, we present an extension of the work presented in chapter 5. In this chapter we introduced an encoding scheme of pre-microRNAs and their associated predicted secondary structure and we utilized extended IFID3 to identify the essential features for this classification problem. In chapter 7, we introduce the software tool developed during this work and discuss its features and capabilities. In chapter 8, we summarize the work presented in this dissertation, conclude it and present some future directions to improve this work.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

Decision trees are one of the common machine learning tools [3] [4] [5] [6] [29]. They are classified as weak learners [32]. A weak learner builds a strong classifier that fits the training data, but may produce very low accuracy when applied to unknown cases. ID3 algorithm proposed by Quinlan in 1979 [12] forms the basis of many decision tree algorithms. ID3 support only symbolic data [1]. Several ID3 extensions were proposed to handle continuous and multi-valued data [1] [5]. Some statistical approaches and tree pruning are used to overcome the problem of overfitting the training data and to improve the generalization of the decision model produced by decision trees.

ID3 partitions the training data recursively based on a test attribute [12]. To select the test attribute at a non-leaf node, ID3 utilizes an entropy based measure called Information Gain. One of the problems of information gain is being biased toward selecting the attribute that results in a larger number of branches [33]. To overcome this deficiency C4.5 [14] uses Information Gain Ratio [14] to select the test attribute. Trees produced by ID3 – known as crisp decision trees- cannot handle data uncertainties and spurious precision in the data. Fuzzy ID3 [1] (FID3) is an extension of the ID3 algorithm. It integrates ID3, fuzzy set theory and the approximate rea-

soning offered by fuzzy logic [34] [37] [38] to overcome some of the deficiencies in ID3. Trees produced by the FID3 algorithm are known as fuzzy decision trees (FDTs). For a brief discussion of fuzzy sets and fuzzy logic, the user is referred to [3].

Generally, a fuzzy decision tree induction mechanism consists of the following four steps [1] [2] [3] [4] [5] [6] [7] [9] [10]:

1. Data fuzzification.
2. Building a fuzzy decision tree.
3. Converting the fuzzy decision into a set of fuzzy rules.
4. Applying the fuzzy rules to make classification and/or prediction (inference).

The data fuzzification is applied to numerical data. The purpose of this step is to reduce the information overload in the decision support process [2]. Selecting the appropriate fuzzy membership function is crucial to the performance of fuzzy decision trees. The fuzzy membership functions may be provided by experts, or may be defined by some other means such as clustering. Common fuzzy membership functions are triangular, trapezoidal and Gaussian.

The fuzzy decision tree building procedure is very similar to that of ID3 [1] [2] [4] [5] 6] [7]; the fuzzified training dataset is partitioned recursively based on the value of a selected splitting (also called branching or test) feature (attribute). The splitting feature is selected such that a certain information measure of separating data belonging to different classes is maximized [7]. Several information measures exist in the literature, typical information measures used include: Information gain (IG) [12] [13], classification ambiguity (CA) [2] and gini-index [35]. Existing fuzzy

ID3 algorithms use either information gain or classification ambiguity to select a branching feature. A gini-index based fuzzy decision tree algorithm was proposed recently in [36]. A node in the tree is considered a leaf node, when all the objects at the node belong to the same class, the number of objects in the node is less than a certain threshold, the ratio between objects' memberships in different classes is greater than a given threshold, or no more features are available. Tuning these thresholds is crucial to the performance and the quality of the produced fuzzy decision trees. A feature can appear only once on any path from the root node to a leaf node in the tree. Unlike crisp decision trees where an object can propagate only to one child node, fuzzy decision trees allow a data item or an object to propagate to more than a child node.

Inference in fuzzy decision trees starts at the root node; the tested object may fall into several leaves in the decision tree - possibly with conflicting decisions - with some certainty [1] [2]. To give a final decision on an object, several reasoning methods exist [1] [2] [3] [4] [6] [11]. Typical ones include applying the fuzzy rule with the maximum firing threshold, or using the sum of products (X-X-+) reasoning method found in fuzzy logic. Interpolation can be used to reason about test objects that fire no rules by manipulating adjacent fuzzy rules.

2.2 Literature Review

Several fuzzy decision tree induction algorithms exist [4]. These algorithms differ in the information theory measure they use to select the branching attribute and/or the inference me-

thod used to draw a final decision on a test object. In [1] a fuzzy ID3 (FID3) algorithm is proposed, the proposed algorithm utilizes a fuzzy version of information gain to select a test attribute. To build a fuzzy decision tree, the proposed algorithm uses a procedure similar to that of ID3. In [2], an FID3 algorithm is proposed, the proposed algorithm uses the same procedure as in [1] to build the fuzzy decision tree. However, the algorithm uses Classification Ambiguity instead of fuzzy information gain to select the test attribute. In [3] fuzzy decision trees issues including tree construction procedure, fuzzy set theory, fuzzy logic, interpolation and fuzzy reasoning were discussed. In [4], a fuzzy decision tree induction method was proposed; the proposed method is the same as that in [1]. However, the proposed method utilizes a procedure called classwise element set to partition the range of discrete attributes. This procedure is used to overcome the information gain bias toward selecting discrete features with many possible values. In [6] a fuzzy decision tree induction freeware (FID3.4) was introduced. The tool implements the same FID3 proposed in [1] except for using a priority queue data structure to unfold the recursion in FID3. The tool takes too long to run and may crash without producing results. In [7], the idea of fuzzy decision forest (FDF) was introduced. FDF extends FID3 by allowing the selection of more than one test attribute at a non-leaf node. Selecting multiple test attributes is used to overcome noisy or missing values associated with any of the test attributes. In [9], a freeware FDF tool is introduced, the tool is an implementation of the idea presented in [7]. This tool proved to be unstable, and is currently unavailable for download with promises since 2004 to make it available in the future. In [10] a new matching method for fuzzy decision tree is pro-

posed, the proposed method assigns each extracted fuzzy rule a truth degree, which is used as a weight during the decision process.

In the literature of fuzzy logic controllers, several methods exist to address rule set reduction and the optimization of fuzzy membership functions. In [15], the authors studied two fuzzy membership functions tuning techniques; the two tuning techniques utilize genetic algorithms to optimize the membership functions by iteratively examining the dataset and fuzzy rule set base. By optimizing the membership functions the authors were able to achieve a slight reduction in the fuzzy rule set base size. However, the approach assumes that the fuzzy rules exist before hand. In [16], the authors utilized genetic algorithms to develop a method for learning membership functions from the data. The main idea is to merge adjacent membership functions when their overlap is greater than a given threshold. Although the proposed method is capable of learning the membership functions from the data, however, it resulted in the generation of more fuzzy rules. In [17], the authors applied several multi-objective evolutionary algorithms to reduce the number of fuzzy rules and to tune membership functions. In [18], the authors proposed an approach for adjusting the membership functions and for reducing the number of fuzzy rules. The proposed approach, first, generates a fuzzy rule for each data item in the dataset, and then the degree of similarity between rules is calculated. Based on the result of similarity calculations, initial membership functions are merged. The adjusted membership functions are then used to generate a new set of fuzzy rules. In [19], the authors reduced the fuzzy rule base by removing redundant and inconsistent rules. However, the time complexity of the algorithm is high, the authors defined redundancy and inconsistency measures that need to be cal-

culated for each pair of rules. In [20], the authors used singular value decomposition to reduce the fuzzy rule base for a fuzzy filter that estimates motor currents. In [21], the authors combined the concept of fuzzy region and the piecewise Lyapunov stability criterion to reduce the fuzzy rule base size. The authors named their technique regional piecewise approach.

In the literature, fuzzy decision trees were applied to specific problems, to some hypothetical examples or to datasets with a relatively small number of features (< 10). Therefore the obtained results represent a proof of concept but may not be generalized. Existing fuzzy decision tree induction applications like FID3.4 introduced in [6] takes a long time to run (a single run takes few weeks on a protein dataset with 450 extracted features). Taking into consideration the time needed to fine tune the FID3 parameters and the time needed to fine tune the fuzzy membership functions associated with each numerical feature in the data, running times of months are expected in order to generate an optimal fuzzy decision tree. Furthermore, existing fuzzy decision tree algorithms produce a huge number of fuzzy rules. Reducing the number of resulting fuzzy rules is important for improving the falsifiability of the decision model and for reducing the computational time. In addition, reducing the number of rules improves the system interpretability, efficiency and the applicability of the resulting fuzzy systems to real time applications. Although, several rule set reduction methods were proposed to reduce the number of fuzzy rules in fuzzy logic controllers. All these methods examine the existing fuzzy rule set base after they are generated, to either remove redundant and/or inconsistent rules or to adjust the initial fuzzy membership functions.

Using a forest of crisp decision trees has proved to be an effective approach for improving the accuracy of decision-tree-based machine learning algorithms. Applying the same concept may improve the performance of fuzzy decision trees. The fuzzy decision forest proposed in [7] proved to be unstable, and therefore, further research is needed to come up with new efficient fuzzy forest algorithms. Another problem with fuzzy systems is the determination of the number of fuzzy sets and the optimal fuzzy membership functions. Most approaches in the literature assume the number of fuzzy sets for any given feature in dataset is known before hand and try to tune the membership functions by means of genetic algorithms. However, this assumption may not be valid for complex systems with huge number of features. For such systems, it is hard to come up with a good fuzzy representation. On the other hand, many of the humans performing the data processing and analysis may lack the experience in the target application field. Therefore, it is very important to derive methods that automate this task.

2.3 Information Theory Measures

At the heart of any decision tree algorithm is the selection of the branching attribute (also called test attribute and splitting attribute). Decision trees either select the test attribute randomly or utilize one of the information theory measures to select it. In this section we will introduce the most common information theory measures used in decision trees to select the branching attribute; these include information gain, classification ambiguity and gini-index.

2.3.1 Information Gain

The most common information theory measure utilized by the ID3 algorithm and its fuzzy extensions is the Information gain measure. Information gain is an entropy based measure. At each internal node in the tree, the information gain associated with each feature in the dataset is calculated, and the feature with the maximum information gain is selected as the splitting (branching) feature.

As mentioned earlier, the Information Gain measure is an Entropy-based measure. Equation 2.1 below defines Entropy:

$$H(S) = \sum_{i}^{N} -p_i * \log_2 p_i \qquad (2.1)$$

Where p_i is the ratio of examples belonging to class C_i in the set of training examples S = {x_1, x_2, x_3, ..., x_k} at a particular node in the tree and is defined by equation 2.2.

$$p_i = \frac{\sum x_k \in C_i}{|S|} \qquad (2.2)$$

Where |S| is the number of examples in the training dataset S.

The information gain of an attribute A, is the measure of the reduction of the sample set entropy after using A to divide this sample set and is given by equation 2.3.

$$G(S,A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} * H(S_v) \qquad (2.3)$$

Where $\frac{|S_v|}{|S|}$ is the ratio of the number of samples with A = v in the sample data set S to the total number of samples in the sample dataset S.

Fuzzy ID3 utilizes a fuzzy version of the information gain measure. Given a data set D, with attributes A_1, A_2, …, A_L and a classified class C = {C_1, C_2, …, C_n} and fuzzy sets F_{i1}, F_{i2}, …, F_{im} for the attribute A_i (each attribute may have a different value of m). Let D^{C_k} be a fuzzy subset in D with class C_k, and let |D| be the sum of membership values in a fuzzy set of data D. Then, the information gain for an attribute A_i by a fuzzy set of D is defined as follows [1]:

$$G(A_i, D) = I(D) - E(A_i, D) \qquad (2.4)$$

Where:

$$I(D) = -\sum_{k=1}^{n} (p_k . \log_2 p_k) \qquad (2.5)$$

$$E(A_i, D) = \sum_{J=1}^{m} (p_{ij} . I(D_{F_{ij}})) \qquad (2.6)$$

$$p_k = \frac{|D^{C_k}|}{|D|} \tag{2.7}$$

$$P_{ij} = \frac{|D_{F_{ij}}|}{\sum_{j=1}^{m} |D_{F_{ij}}|} \tag{2.8}$$

Finally, Since Information Gain (IG) is an Entropy-Based measure; it has a strong bias toward selecting attributes that result in larger number of branches. In other words, it favors attributes with many values. For example, consider a dataset in which one of the features being used as an identifier, IG will pick this feature as the splitting feature, although this feature may be irrelevant to the classification process. To overcome this deficiency, information gain ratio is used instead of IG. The definition for the Information Gain Ratio (IGR) is given by equation 2.9.

$$IGR(S, A) = G(S, A) / IV(S, A) \tag{2.9}$$

Where IV(S, A) is the gain ratio of splitting the sample dataset S using attribute A and is given by the following equation:

$$IV(S, A) = -\sum \frac{|\{n \in S\}|}{|S|} * \log_2 (\frac{|\{n \in S\}|}{|S|} \tag{2.10}$$

Where n is the number of samples left in the class after testing on attribute A.

2.3.2 Classification Ambiguity

Classification Ambiguity is another information theory measure used by some versions of the Fuzzy ID3 algorithm to select the splitting attribute at a given tree node. This section reviews some of the definitions and terms involved in the calculations of the attribute classification ambiguity.

A fuzzy membership function (μ(x)) of a fuzzy feature Y defined on X can be interpreted as the possibility of Y taking the value x among all element in X. therefore π(x) = μ(x) for all xεX can be considered as a possibility distribution of Y on X. Let π = (π(x) | xεX) denote a normalized distribution of Y on X = {x_1, x_2, …, x_n}, the possiblistic measure of ambiguity (also called nonspecificity and U-uncertainty) is defined as:

$$E_a(Y) = g(\pi) = \sum_{i=1}^{n} (\pi^*_i - \pi^*_{i+1}) \ln i \qquad (2.11)$$

where: $\pi^* = \{\pi^*_1, \pi^*_2, \ldots, \pi^*_n\}$ is the permutation of the possibility distribution π = {π(x_1), π(x_2), …, π(x_n)} sorted so that $\pi^*_i \geq \pi^*_{i+1}$ for i = 1, 2, …, n, and $\pi^*_{n+1} = 0.$ based on the definition, an $E_a(Y) = 0$ indicates no ambiguity and an Ea(Y) = ln(n) indicates the greatest ambiguity [2].

The fuzzy subsethood S(A, B) is defined as the degree to which A is a subset of B [2], and is given by equation 2.12. The fuzzy subsethood can be used to measure the truth degree of a fuzzy rule.

$$S(A,B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u \in U} \min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)} \tag{2.12}$$

where μ_A and μ_B are the membership functions for fuzzy sets A and B respectively.

Fuzzy evidence is a condition fuzzy subset defined on the object space, and represents the linguistic values of one or more attributes, for example, "The Temperature is Hot and the Outlook is Sunny" [2]. Then given fuzzy evidence E, the possibility of an object to be classified as class Ci can be defined as:

$$\pi(C_i \mid E) = \frac{S(E, C_i)}{\max_j S(E, C_j)} \tag{2.13}$$

where $S(E, C_i)$ is the truth degree of the classification rule: "IF E THEN C_i" and is calculated by using the fuzzy subsethood definition, and $\pi(C|E) = \{ \pi(C_i|E), i = 1, \ldots, L\}$ is a normalized possibility distribution on the nonfuzzy label space $C = \{ C_1, \ldots, C_L\}$ [2]. Finally the classification ambiguity associated with a single fuzzy evidence E, such as a particular value of an attribute is given by:

$$G(E) = g(\pi(C \mid E)) \tag{2.14}$$

The reader is referred to [2] for more details.

2.3.3 Gini Index

Gini index (also known as gini impurity) is the information measure used in classification and regression trees (CART) [35] to select the branching feature. Given a dataset D with N examples from n classes $\{c_1, c_2, \ldots, c_n\}$, the gini index, gini(D) is defined as follows:

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.15}$$

Where p_i is the relative frequency of class C_i in D.

If the dataset D is split into two subsets D_1 and D_2 with sizes N_1 and N_2 respectively, the gini index of the split data is given by the following equation:

$$Gini_{split}(D) = \frac{N_1}{N} gini(D_1) + \frac{N_2}{N} gini(D_2) \tag{2.16}$$

For the general case where the dataset D is split into k subsets $\{D_1, D_2, \ldots, D_k\}$ with sizes $\{N_1, N_2, \ldots, N_k\}$ the gini index of the split data is given by the following equation:

$$Gini_{split}(D) = \sum_{i=1}^{k} \frac{N_i}{N} gini(D_i) \tag{2.17}$$

The attribute with the smallest gini index is selected as the branching attribute at the tree node.

The above definition of gini index works for crisp decision trees and need to be formulated using fuzzy terms in order to be use in fuzzy decision trees. Given a dataset D with N ex-

amples from n classes and fuzzy sets F_{i1}, F_{i2}, …, F_{im} for the attribute A_i (each attribute may have a different value of m). Let D^{C_k} be a fuzzy subset in D with class C_k, and let |D| be the sum of membership values in a fuzzy set of data D. The gini index of D gini of D is defined as follows:

$$Gini(D) = 1 - \sum_{k=1}^{n} \left(\frac{|D^{C_k}|}{|D|} \right)^2 \tag{2.18}$$

If the dataset D is split into two fuzzy subsets D_1 and D_2 then the fuzzy gini index of the split is given by:

$$Gini_{split}(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2) \tag{2.19}$$

For the general case where the fuzzy set D is split into k fuzzy subsets $\{D_1, D_2, \ldots, D_k\}$ the gini index of the split data is given by the following equation:

$$Gini_{split}(D) = \sum_{i=1}^{k} \frac{|N_i|}{|N|} gini(D_i) \tag{2.20}$$

The fuzzy gini index information measure was derived by us following a derivation procedure similar to that used in the derivation of the fuzzy information gain.

2.4 Interactive Dichotomizer 3 (ID3) Algorithm

The Interactive Dichotomizer 3 (ID3) [21] algorithm is an efficient algorithm for building decision trees. It forms the basis for many decision tree algorithms and applications. ID3 was proposed by Quinlan in 1979. ID3 builds the decision tree from symbolic data [1]. Several revisions of ID3 were proposed to deal with continuous and/or multi-valued data [1] [5]. ID3 constructs the decision tree by the recursive partitioning of the training dataset based on the values of a selected attribute [2]. ID3 utilizes Information Gain measure to select the branching (test) attribute at a given tree node. It selects the attribute that maximizes the information gain as the split attribute. The recursive partitioning of the dataset continues until either all data instances at a particular node belong to the same class or no more test attributes are available. An attribute can only be used once along the path from the tree root node to particular leaf node. If a leaf node contains data instances belonging to different classes, the node is assigned the name of the majority class at the node. Following are the steps of the ID3 Algorithm:

- ID3 (Data Sample, Target Attribute, Set of Attributes)

  Data Sample: refers to the training data samples at a given tree node. If this node is the root of the decision tree, then the Data Sample contains the entire training dataset.

  Target Attribute: Refers to the attribute whose value is be predicted by the tree (class)

  Attributes: Refers to the set of attributes to be tested.

- Create a root node

- If all examples in the data sample belong to same class, then return the single-node tree Root. Label = the name of the target attribute.
- If attributes is empty, then return the single-node tree Root. Label = The name of the most common target attribute value in examples in the Data Sample.
- Otherwise Begin
    1. Calculate the IG of each attribute in the attributes set and select the attribute that maximizes the IG.
    2. for each value v_i of the selected attribute (A) do
        - Add a tree branch below this tree node ($A = v_i$)
        - Let $Examples_{vi}$ be the subset of Data Sample with $A = v_i$
        - If $Examples_{vi}$ is empty then below this new branch add a leaf node with label = the most common target attribute value in the Data Sample.
        - Else, below this branch add the subtree ID3($Examples_{vi}$, Target Attribute, Attributes – {A})
- END

2.5 Fuzzy ID3 Algorithm

Fuzzy ID3 is an extension of the existing ID3 algorithm; it integrates fuzzy set theory and ID3 to overcome the effects of spurious precision in the data, to treat uncertainties in the data and to reduce the decision tree sensitivity to small changes in attribute values. Fuzzy ID3 algo-

rithm uses a very similar procedure to that of the original ID3 algorithm to build the decision tree, except that it selects the test attribute based on the fuzzy information gain rather than information gain. Several fuzzy decision tree algorithms exist in the literature [4]. These algorithms differ slightly from each other.

Given a data set D, with attributes A_1, A_2, …, A_L and a classified class C = {C_1, C_2, …, C_n} and fuzzy sets F_{i1}, F_{i2}, …, F_{im} for the attribute A_i (each attribute may have a different value of m). Let D^{C_k} be a fuzzy subset in D with class C_k, and let |D| be the sum of membership values in a fuzzy set of data D. The algorithm to generate a fuzzy decision tree works as follows:

1. Generate the root node with a fuzzy set of all training data with membership value of 1.
2. The node is a leaf node, if the fuzzy set of the data at that node satisfies any of the following conditions:
    a. The number of objects in the node data set D is less than a given threshold; this threshold is called leaf control threshold θ_n. that is:

    $$|D| < \theta_n.$$

    b. The proportion of a data set of any class C_k $|D^{C_k}|$ in the node data set D is greater than or equal to a given threshold. This threshold is called fuzziness control threshold θ_r. That is:

    $$\frac{|D^{C_k}|}{|D|} \geq \theta_r$$

   c. No more attributes are available.

3. The class name assigned to a leaf node depends on the inference method and is either the name of the class with the greatest membership value, or the node is assigned all class names along with membership values. (See below for more details)
4. If the node does not satisfy any of the above conditions then do the following:
   a. For all attributes, calculate the fuzzy information gain (notice that the information measure depends on the version Fuzzy ID3, some versions may use other information theory measures such as classification ambiguity); select the attribute that has the greatest fuzzy information gain as the test attribute.
   b. Divide the fuzzy data set at the node into fuzzy subsets using the selected test attribute, with the membership value of an object in a subset set to the product of its' membership value in parent node dataset and the value of the selected attribute fuzzy term.
   c. For each of the subsets, generate new node with the branch labeled with the fuzzy term of the selected attribute.
5. For each new generated node repeat recursively from step 2.

The reader is referred to [1] for more details. Assigning a class label to a leaf node can be achieved using one of the following methods:

- The node is labeled with the class label of the class that has the greatest membership value.

- The node is labeled by all class names along with their membership values.

The Fuzzy ID3 reviewed above is presented in [1]. The authors in [1] pointed out that the only difference between the preseneted fuzzy ID3 and the original ID3 algorithm is in the way the information gain is calculated. However, there is another important difference between the two algorithms. In ID3 an attribute that is selected prviously on a path from a node to the root node in the decision tree as a test attribute is excluded from information gain calculations at the node. On the other hand, in the fuzzy ID3 algorithm, the information gain is calculated for all attributes regardless whether the attribute was selected as a test attribute up the tree or not (refer to 4.a of Fuzzy ID3 description in in this section), it may be the case that the information gain calcualtion at the child node will lead to selecting a test attribute that is already selected previously up in the decision tree; since the algorithm does not allow using an attribute as test attribute more than once, it is not clear how to proceed in such a case. Two approahes can be used to deal with this case; in the first approach, if such case happens we designate the node as a leaf node and proceed. In the second approach, A procedure similar to that of ID3 is followed, that is 4.a of the fuzzy ID3 algorithm is modified as follows: ***For all attributes not selected so far as a test attribute up in the path from current node to root of the tree, calculate the information gain and select the attribute that maximizes the information gain as a test attribute.*** Both option were implmented and found to achieve comparable performance. The details of the experiments will be dicussed later in chapter 3.

2.6 Inference in Fuzzy ID3

Like inference in ID3, inference in Fuzzy ID3 starts at the root of the fuzzy decision tree. However, a data object may propagate across several paths down the tree falling into one or more leaf nodes with some certainty with possibly different classification decisions [1] [10]. In this section we will review two of the inference methods utilized by fuzzy decision tree induction tools. The first method corresponds to labeling the leaf node with the class that has the greatest membership value. And the second method corresponds to labeling the leaf node with all class names along with their membership values. For a brief description of the inference methods, the reader is referred to [3].

In the first method, as the object propagates down the fuzzy decision tree its membership value in all of the decision tree leaf nodes is calculated. Then the object is assigned the class label of the leaf node, in which it has the greatest membership value. In other words, it is assigned the same label of the fuzzy rule with the maximum firing strength (max-min).

In the second method, as the test object propagates down the decision tree; it will reach each leaf node with some certainty or membership value. However, since the leaf nodes are labeled with all class names and their membership values, the certainties are multiplied by the class proportions in the leaf node. Then the certainties of each class are summed, and the test object is assigned the class label with the greatest certainty (In fuzzy logic this method is referred to by X-X-+) refer to [1] for more details.

Chapter3

IMPROVED FUZZY ID3 ALGORITHM

There are several variations of the fuzzy ID3 algorithm [4]. All of these variations use similar procedures in building the fuzzy decision tree. However, they differ in the information theory measure they use to select the branching attribute. This section introduces an improved version of the Fuzzy ID3 [39]. The improved version utilizes both fuzzy information gain and classification ambiguity to select the branching feature. At the root node of the tree, the algorithm uses classification ambiguity to select the branching feature. In the other non-leaf tree nodes, the algorithm uses fuzzy information gain to select the branching feature. The modified algorithm achieved better accuracy than the original Fuzzy ID3 as well as crisp programs such C4.5 [22] on a wide range of datasets. From now on, we will refer to our modified version of fuzzy ID3 as the Improved Fuzzy ID3 algorithm. The rest of this chapter is organized as follows: In section 3.1 we will introduce our modified version of the fuzzy ID3 algorithm. In section 3.2 we will discuss the data sets used to test the algorithm performance, we will also discuss the fuzzy membership functions used during the experiments and the procedures to select the fuzzy membership functions for each of the datasets. In addition we will present the experimental setup. In section 3.3 we will present the experimental results and discuss them. And in section 3.4 we will present our conclusions and suggest ideas for future work.

3.1 Improved Fuzzy ID3 Algorithm (IFID3)

Improved fuzzy ID3 (IFID3) uses the same fuzzy decision tree building procedure as that of ID3 and Fuzzy ID3 algorithms discussed in chapter 2. It uses the classification ambiguity measure introduced in [2] to extend the Fuzzy ID3 algorithm presented in [1]. As mentioned earlier, IFID3 uses attribute classification ambiguity to select the branching attribute at the root node, and fuzzy information gain else where. Given a data set D, with attributes A_1, A_2, …, A_L and a classified class C = {C_1, C_2, …, C_n} and fuzzy sets F_{i1}, F_{i2}, …, F_{im} for the attribute A_i (each attribute may have a different value of m). Let D^{Ck} be a fuzzy subset in D with class C_k, and let |D| be the sum of membership values in a fuzzy set of data D. IFID3 works as follows:

1. Generate the root node with a fuzzy set of all training data with membership value of 1 (This not necessary the case, membership values can be initialized manually, for example instances in the training dataset can be associated with weights).
2. The node is a leaf node, if the fuzzy set of the data at that node satisfies any of the following conditions:
    a. The number of objects in the node data set D is less than a given threshold; this threshold is cold leaf control threshold θ_n. that is:

$$|D| < \theta_n.$$

b. The proportion of a data set of any class C_k $|D^{C_k}|$ in the node data set D is greater than or equal to a given threshold. This threshold is called fuzziness control threshold θ_r. That is:

$$\frac{|D^{C_k}|}{|D|} \geq \theta_r$$

c. No more attributes are available for classification.

3. The class name assigned to a leaf node depends on the inference method and is either the name of the class with the greatest membership value, or the node is assigned all class names along with membership values.

4. If the node does not satisfy any of the above conditions then do the following:

   a. If this node is the root node of the decision tree, then calculate the Attribute classification Ambiguity for all attributes in the dataset and select the attribute with the minimum attribute classification ambiguity as the test attribute.

   b. Else, the node is not the root node of the decision tree, calculate the fuzzy information gain for all attributes; and select the attribute that has greatest fuzzy information gain as the test attribute.

   c. Divide the fuzzy data set at the node into fuzzy subsets using the selected test attribute, with the membership value of an object in a subset set to the product of its' membership value in parent node dataset and the value of the selected attribute fuzzy term.

d. For each of the subsets, generate new node with the branch labeled with the fuzzy term of the selected attribute.

5. For each new generated node repeat recursively from step 2.

3.2 Experimental Details

To evaluate the performance of the IFID3 algorithm, the algorithm was tested on several standard datasets, and its performance was compared to the performance of several decision tree algorithms. This section will introduce the datasets, discuss data preprocessing steps and the fuzzy membership functions used to fuzzify the datasets and other experimental issues.

3.2.1 Datasets

To evaluate the performance of IFID3, we randomly selected six standard UCI-machine-learning repository [42] datasets and one non-standard dataset. The six standard datasets were selected to represent different application areas. These datasets include: Breast Cancer Wisconsin (diagnostic) [44] [45], Waveform Database Generator (V1) [35], Statlog (heart) [42], Ionosphere [46], Statlog (Landsat Satellite) [42], and Glass [42] Datasets. The non standard dataset is an encoded version of the protein dataset RS126 [43] pertains to helices. The features of these datasets are described briefly in Table 3.1 [39].

Table 3.1: Datasets Description

Dataset	Number of Samples	Number of Attributes	Number of Classes
Breast Cancer Wisconsin (Diagnostic)	569	32	2
Waveform Database Generator (VI)	5000	21	3
Statlog (Heart)	270	14	2
Ionosphere	351	34	2
Statlog (Landsat Satellite)	6435	36	7
Glass	214	9	6
RS126 (H/~H)	21834	455	2

3.2.2 Datasets Preprocessing

For experimental purposes, the datasets described in table 3.1 were divided into training and testing datasets. During the experiments; if the obtained dataset is already divided into training and testing datasets, we use the existing partitions, otherwise the dataset is partitioned randomly into three folds, such that the ratio of objects belonging to different classes in the data folds is the same as that in the original dataset.

3.2.3 Data Fuzzification and Fuzzy Membership Functions

All variations of fuzzy ID3 algorithm require both the training and testing datasets to be in a fuzzy form. Therefore if the datasets are not in a fuzzy form, continuous features in the da-

tasets need to be fuzzified. To achieve this task, for each continuous feature in a given dataset, we need to determine the number of fuzzy sets to partition the feature range into, and the appropriate fuzzy membership function. Such information may be provided by experts in the field from which the dataset was obtained. However, such information is seldom available; therefore in our experiments two approaches were examined. In the first approach we divide all the continuous features in the dataset evenly into an equal number of fuzzy sets, and the second approach we utilized cluster center information to construct the membership functions for all continuous features. In both approaches, adjacent fuzzy membership functions cross each other at a membership value of 0.5. We also assumed that the sum of membership values in all fuzzy sets associated with a given datum corresponding to a certain feature value is 1. To determine the optimal number of fuzzy sets, the number of fuzzy sets was treated as one of the parameters that need to be tuned.

To divide the range of a given feature evenly into n fuzzy sets with trapezoidal membership function, we need to calculate the four boundary points for each trapezoid. To do so the following algorithm was used:

- n = number of Fuzzy Sets.
- min = Feature Minimum Value (data in both the training and testing dataset are considered for this purpose).
- max = Feature Maximum Value (data in both the training and testing dataset are considered for this purpose).

- Calculate the range of the feature using the following formula:

$$\text{Range} = \max - \min \tag{3.1}$$

- Calculate the following value (HalfOverlap) using the following formula:

$$\text{HalfOverlap} = \text{Range}/(4 * n) \tag{3.2}$$

- Set the trapezoid boundaries representing the fuzzy membership function of the first fuzzy set (most left) as follows:
    a. First Point = Second Point = min
    b. Third Point = min + (Range/n) – HalfOverlap
    c. Fourth Point = min + (Range/n) + HalfOverlap
- For i = 1 to n – 2
    a. 1st point of the $(i+1)^{th}$ Fuzzy Set = min + (i * Range/n) – HalfOverlap
    b. 2nd Point of the $(i+1)^{th}$ Fuzzy Set = min + (i * Range/n) + HalfOverlap
    c. 3rd Point of the $(i+1)^{th}$ Fuzzy Set = min + ((i + 1) * Range/n) – HalfOverlap
    d. Fourth Point of the $(i+1)^{th}$ Fuzzy Set = min + ((i + 1)* Range/n) + HalfOverlap
- Set the trapezoid boundaries representing the fuzzy membership function of the nth fuzzy set (most right) as follows:
    a. First point = min + ((n -1) * Range)/n) – HalfOverlap
    b. Second point = min + ((n -1) * Range)/n) + HalfOverlap
    c. Third Point = Fourth Point = max

The following example illustrates the above procedure: Assume we have a feature which takes a minimum value of 0 and a maximum value of 16. Assume that we need to divide the

range of this feature into 4 fuzzy sets. First, we set n = 4, min = 0 and max = 16. Next we calculate the range = 16 – 0 = 16. Next we calculate HalfOverlap = 16/(4 *4) = 1. To generate the four points required for representing the membership function of the first fuzzy set, we set first point = second point = 0, third point = 0 + (16/4) – 1 = 3, fourth Point = 0 + (16/4) + 1 = 5. These are the x-coordinates for the four points defining the corners of the trapezoid. Since the membership function value ranges between 0 and 1 we get the following four points {(0,0), (0,1), (3,1), (5,0)}. A similar procedure is used to calculate corners for the membership functions for the rest of the fuzzy sets. Figure 3.1 shows the resulting membership functions.

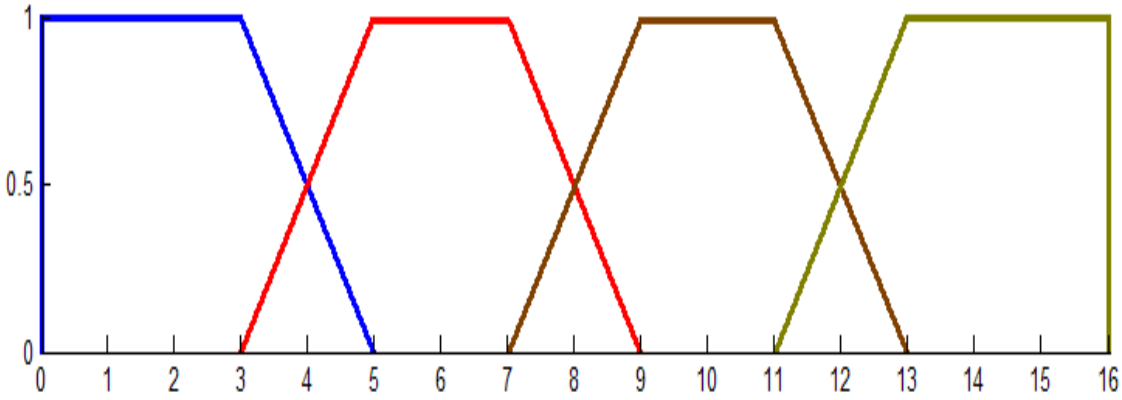


Figure 3.1: Trapezoidal membership function for the feature in the above example

To divide the range of a given feature evenly into n fuzzy sets with triangular membership function, we need to calculate the three boundary points for each triangle. We can also

treat the triangular membership function as a special case of the trapezoidal membership function with the length of the upper side = 0. The following algorithm was used to divide a feature range evenly into n fuzzy sets with triangular membership function (Notice that the membership function for the first and the last fuzzy sets is not triangular, this is because we wanted to keep the sum of the membership value of any data instance = 1):

- n = number of Fuzzy Sets.
- min = Feature Minimum Value
- max = Feature Maximum Value
- Calculate the range of the feature using the following formula:

$$\text{Range} = \max - \min \tag{3.3}$$

- Calculate the following value (HalfOverlap) using the following formula:

$$\text{HalfOverlap} = \text{Range}/(n + 1) \tag{3.4}$$

- Set the boundaries representing the fuzzy membership function of the first fuzzy set (most left) as follows:
    a. First Point = Second Point = min
    b. Third Point = min + HalfOverlap
    c. Fourth Point = min + (2 * HalfOverlap)
- For i = 1 to n – 2
    a. 1st Point of the (i+1)th Fuzzy Set = min + ((Range * i)/(n+1))
    b. 2nd Point of the (i+1)th Fuzzy Set = min + ((Range * i)/(n+1)) + HalfOverlap
    c. 3rd Point of the (i+1)th Fuzzy Set = min + ((Range *(i + 1)/(n+1)) + HalfOverlap.

- Set the boundaries representing the fuzzy membership function of the nth fuzzy set (most right) as follows:
    a. First point = min + ((n -1) * Range)/(n + 1))
    b. Second point = min + ((n -1) * Range)/(n + 1)) + HalfOverlap
    c. Third Point = Fourth Point = max

The following example illustrates the above procedure: Assume we have a feature which takes a minimum value of 0 and a maximum value of 15. Assume that we need to divide the range of this feature into 4 fuzzy sets. First, we set n = 4, min = 0 and max = 15. Next we calculate the range = 15 – 0 = 15. Next we calculate HalfOverlap = 15/(4 + 1) = 3. To generate the four points required for representing the membership function of the first fuzzy set, we set first point = second point = 0, third point = 0 + 3 = 3, fourth Point = 0 + (2 * 3) = 6. These are the x-coordinates for the four points defining the corners of membership function. Since the membership function value ranges between 0 and 1. We have the following four points {(0,0), (0,1), (3,1), (6,0)}. For the second fuzzy set, first point = 0 + ((15 * 1)/(4 + 1)) = 3, second point = 0 + ((15 * 1)/(4 + 1)) + 3 = 6, third point = 0 + ((15 * 2)/(4 + 1)) + 3 = 9. A similar procedure is used to calculate corners for the membership functions for the rest of the fuzzy sets. Figure 3.2.3.2 shows the resulting membership functions.

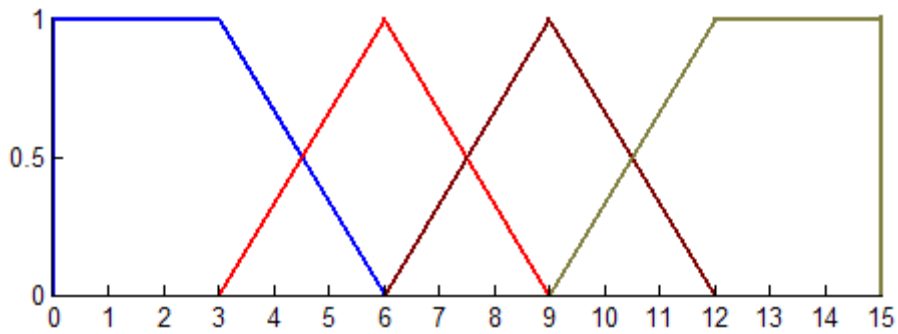


Figure 3.2: Triangular fuzzy membership function for the feature in the above example (Notice the first and last fuzzy membership functions are not triangular)

To use clustering to build the fuzzy membership functions, the data were clustered in n clusters, where n is the number of the desired fuzzy sets used to partition the range of a feature. K-means clustering was used to cluster the data into the desired number of clusters, the calculated cluster centers were used to generate the membership functions. All features were divided into an equal number of fuzzy sets. The following example illustrates how to use the cluster centers to generate either a triangular or trapezoidal membership functions. Assume we want to partition the range of a certain feature into four fuzzy sets, so n = 4, assume that the feature minimum value is 0 and its maximum value is 15. Now, assume the resulting cluster centers values associated with this feature are {2, 5, 8, 11}. For a triangular membership function, each cluster center represents the top of the triangle representing a fuzzy membership of any of

the fuzzy sets. Figure 3.3 shows the resulting triangular fuzzy membership function for the four fuzzy sets.

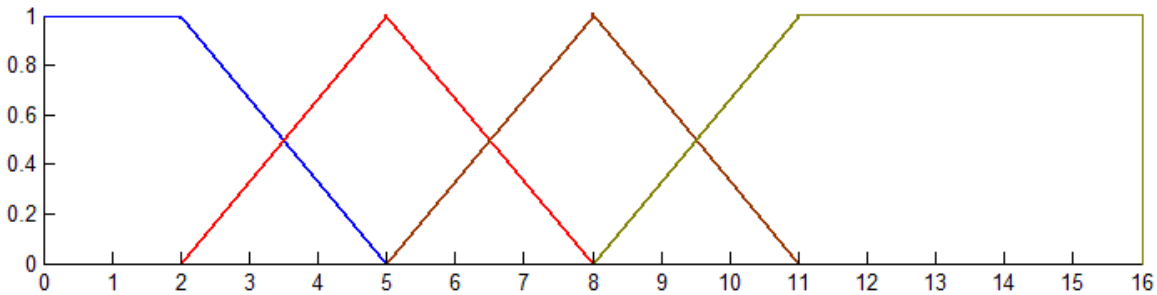


Figure 3.3: Calculated Triangular membership Function using {2, 5, 8, 11} as cluster centers.

To divide the range of a given feature into n fuzzy sets with trapezoidal membership function using the cluster centers information, we need to calculate the four boundary points for each trapezoid. To do so the following algorithm was used:

- n = number of Fuzzy Sets.
- min = Feature Minimum Value
- max = Feature Maximum Value
- Centers[n] stores the n cluster centers.
- Set the trapezoid boundaries representing the fuzzy membership function of the first fuzzy set (most left) as follows:
    a. 1st Point = 2nd Point = min

b. 3rd Point = Centers[0] + ((Centers[1] - Centers[0])/4)

c. 4th Point = Centers[1] - ((Centers[1] - Centers[0])/4)

- For i = 1 to n – 2

  a. 1st Point of the (i+1)th Fuzzy Set=Centers[i-1]+((Centers[i]- Centers[i-1])/4)

  b. 2nd Point of the (i+1)th Fuzzy Set=Centers[i]-((Centers[i]-Centers[i-1])/4);

  c. 3rd Point of the (i+1)th Fuzzy Set=Centers[i]+((Centers[i+1]- Centers[i])/4)

  d. 4th Point of the (i+1)th Fuzzy Set=Centers[i+1]-((Centers[i+1]- Centers[i])/4)

- Set the trapezoid boundaries representing the fuzzy membership function of the nth fuzzy set (most right) as follows:

  a. 1st point=Centers[n-2] + ((Centers[n-1] - Centers[n-2])/4)

  b. 2nd point=Centers[n-1] - ((Centers[n-1] - Centers[n-2])/4)

  c. 3rd Point = 4th Point = max

Assume we want to partition the range of a certain feature into four fuzzy sets, so n = 4, assume that the feature minimum value is 0 and its maximum value is 15. Now, assume the resulting cluster centers values associated with this feature are {2, 5, 8, 11}. Figure 3.4 shows the resulting trapezoidal fuzzy membership function for the four fuzzy sets.

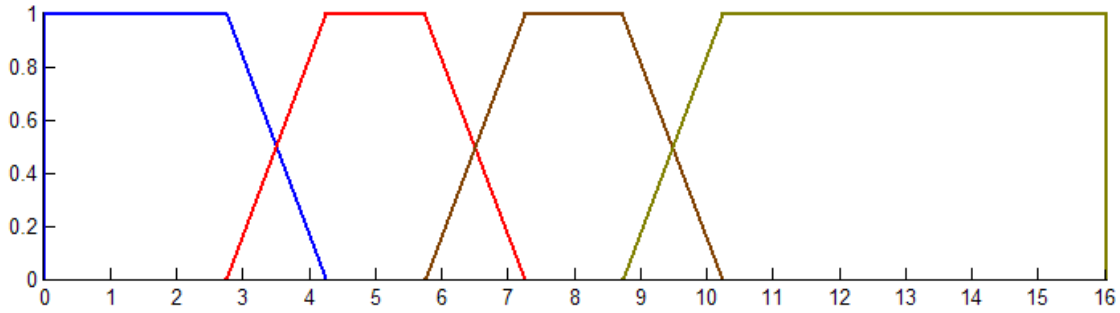


Figure 3.4: Calculated Trapezoidal membership function using {2, 5, 8, 11} as the cluster centers.

3.2.4 Experimental Setup and results

To evaluate the performance of the fuzzy decision tree generated by the improved Fuzzy ID3 algorithm, we compared the performance of our method with the performance of C4.5, GATree [40] (Genetic Algorithm Decision Tree), Random Forest and Fuzzy-EBY [41]; Fuzzy-EBY generates fuzzy ID3 based decision trees. For this purpose, we used six standard UCI-machine-learning repository data sets and one non-standard dataset, refer to table 3.1. The non standard dataset is an encoded version of the protein dataset RS126 pertains to helices. During the experiments, all the datasets -except the Statlog (Landsat Satellite) dataset- were partitioned randomly into three folds, such that the proportions of data items belonging to each of the classes in the data folds are equal to those in the entire dataset. For each dataset the experiment was executed three times, with two of the data folds used as the training dataset and the third one as a testing dataset, each time changing the fold used as the testing dataset (3-fold

cross-validation) and the obtained accuracies were averaged. The number of fuzzy sets per feature was varied from 2 to 14 and the leaf control threshold was varied between 1 and the Square root of the number of instances in the training datasets in steps of 2. The same number of fuzzy sets and the same leaf control threshold was used for all folds belonging to a particular dataset. To decide the class of each object in the testing dataset, the object is assigned the class label of the leaf node, in which it has the greatest membership value. In other words, it is assigned the same label of the fuzzy rule with the maximum firing strength. The experimental results are shown in table 3.2 [39] and table 3.3 [39], where the accuracy is the average value of the three trials.

Table 3.2: Accuracy achieved by IFID3 versus accuracy achieved by other single-tree methods

Dataset	**GATree V2.0**	**C4.5**	**FuzzyEBY**[1]	**FID3**[2] **"IG"**	**FID3**[3] **"IG+Amb"**
Breast Cancer Wisconsin (Diagnostic)	0.930	0.938	0.935	0.970	0.977
Waveform Database Generator (VI)	0.702	0.764	0.751	0.797	0.832
Statlog (Heart)	0.796	0.752	0.745	0.804	0.819
Ionosphere	0.852	0.889	0.903	0.940	0.940
Statlog (Landset Satellite)	0.717	0.857	0.836	0.878	0.880
Glass	0.538	0.619	0.621	0.696	0.706
RS126 (H/~H)	0.678	0.695	0.746	0.758	0.759

1: The Java virtual machine runtime parameters needed to be modified for this software to run.
2: Our implementation of Fuzzy ID3 using Information Gain.
3: Our implementation of Fuzzy ID3 using Information Gain and Attribute Classification Ambiguity.

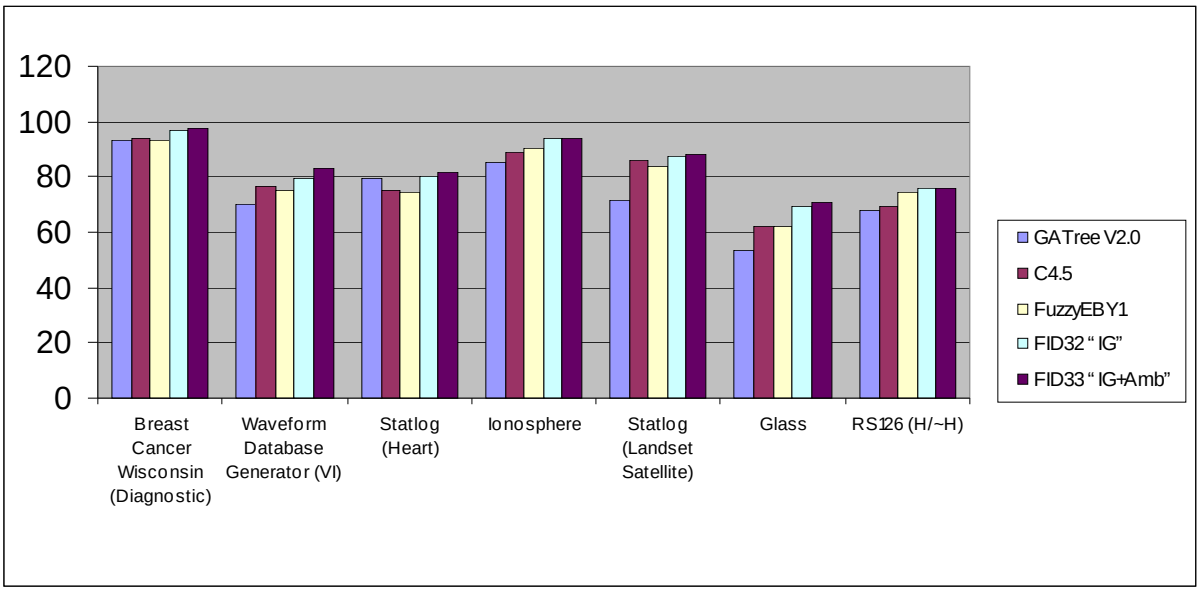


Figure3.5: Summary of the Experimental Results Accuracy of IFID3 versus other methods that produce a single Decision Tree

Table 3.3: Accuracy of IFID3 versus accuracy of Random Forests (RF)

Dataset	**RF 1 Tree**	**RF 5 Trees**	**RF Best**	**FID3[1] "IG + Amb"**
Breast Cancer Wisconsin (Diagnostic)	0.926	0.958	0.969	0.977
Waveform Database Generator (VI)	0.724	0.794	0.857	0.832
Statlog (Heart)	0.707	0.778	0.830	0.819
Ionosphere	0.860	0.912	0.942	0.940
Statlog (Landset Satellite)	0.829	0.878	0.914	0.880
Glass	0.551	0.684	0794	0.706
RS126 (H/~H)	0.646	0.738	0.816	0.759

1: Our implementation of Fuzzy ID3 using Information Gain and Attribute Classification Ambiguity.

In our experiments, we also used clusters-centers information to generate fuzzy membership functions; using these membership functions achieved accuracy results comparable to those produced by fuzzy membership functions produced by dividing the attribute range evenly, with the latter producing slightly better results in most cases. However the clusters centers approach achieved better performance (running time) and resulted in a reduced number of rules, further research is needed to investigate how to use information produced by a clustering algorithm more efficiently to generate better fuzzy membership functions, more over a method for determining the optimal number of clusters is needed. We are investigating these issues.

In section 2.5, we discussed the case when the information gain calculation at a node results in the selection of an attribute that was previously selected up the decision tree on the path to the node, and we suggested two alternative procedures, the first (option 1) was to designate the node as a leaf node, while the second (option 2) was to take an approach similar to that of ID3. We implemented the two options. Although, both options achieved the same accuracy in most cases; the first option runs faster and produced fewer rules than the second option, with the second option producing a slightly better accuracy in very few cases. However the confusion matrices produced by both options were also different; Figure 3.6 shows a representative example of the confusion matrices produced by both options.

$$\begin{pmatrix} 65 & 0 \\ 5 & 119 \end{pmatrix} \quad \begin{pmatrix} 67 & 3 \\ 2 & 117 \end{pmatrix}$$

(a) (b)

Figure 3.6: Confusion Matrices. (a) Confusion matrix produced by option 1. (b) Confusion Matrix produced by option 2.

3.3 Conclusions and Future Work

In this chapter, we presented a modified version of the Fuzzy ID3 algorithm; our algorithm uses Attribute Classification Ambiguity to select the branching attribute at the root node of the decision tree, and fuzzy information gain to select the branching attribute at all the other non-leaf nodes in the tree. Experimental results showed that our algorithm achieved better accuracy than the original fuzzy ID3 algorithm on a wide range of datasets.

Although our algorithm presented a simple approach to integrate two information measures, it showed that such integration can lead to better performance. A number of information measures exist, and we believe that more efficient methods to integrate these measures exit and need to be investigated.

Using a forest of crisp decision trees has proved to be an effective approach in improving the accuracy of decision-tree-based machine learning algorithms. We believe applying the

same approach to fuzzy decision trees will result in boosting the accuracy of these trees. Therefore the fuzzy-based forest approach needs to be investigated more thoroughly.

Finally, one of the drawbacks of our method is the huge number of generated fuzzy rules. Reducing the number of rules required for a decision is very important, both because it increases the computational performance of the fuzzy decision tree induction process and for the more fundamental reason that it improves the falsifiability of decision model and improves its interpretability and its applicability to real time applications. The likelihood that the rules actually reflect an experimentally validateable or causal relationship between the independent and dependent variables of the problem is much higher with a smaller, and often dramatically smaller rule base.

Chapter 4

RULE SET REDUCTION IN FUZZY DECISION TREES

EXTENDED IMPROVED FUZZY ID3 ALGORITHM

The Improved Fuzzy ID3 algorithm [39] discussed in the previous chapter generated a large number of fuzzy rules. Reducing the number of fuzzy rules required for making a decision is of critical importance, both because it increases the computational performance and for the more fundamental reason that it improves the falsifiability and the interpretability of the decision model and reduces its complexity. To reduce the number of generated fuzzy rules, we introduced a modified version of the Improved Fuzzy ID3 algorithm [47]. The modified version introduces a new threshold on the membership value of a given data item to propagate down a fuzzy decision tree from a parent node to any of its child nodes during the fuzzy decision tree generation step. Our experimental results showed that the modified algorithm achieved a significant reduction in the number of the resulting rules. This reduction in the number of rules was associated with a slight improvement in the accuracy and a huge reduction in the computational time. The reduction in computational time is due to the fact that the execution time especially the time associated with inference part is proportional to the number of rules. The rest of this chapter is organized as follows: Section 4.1 defines fuzzy rule set reduction and discusses its importance. Section 4.2 introduces a new syntax for fuzzy rules and presents the new threshold on

the membership value of an object to propagate down a branch. In addition it introduces the modified version of the improved FID3. Section 4.3 presents the experimental procedure and discusses the results and section 4.4 concludes the chapter and presents future research.

4.1 Introduction and Background

Rule set reduction refers to the process of generating a smaller set of rules from a larger set of rules. This is achieved mainly by removing redundant rules and/or by merging adjacent rules leading to same decision. The main purpose of rule reduction is improving the human interpretability of the decision model by reducing its complexity. In addition, rule set reduction makes the process of validating the resulting decision model easier and improves the applicability of the resulting models to real time applications and can improve the system accuracy. In the literature many researchers in the field of fuzzy logic addressed the rule set reduction problem. All proposed methods examine the existing fuzzy rule set base to either remove redundant and/or inconsistent rules or to merge similar fuzzy rules. Most approaches will run at least in two rounds. In the first round a number of fuzzy rules is generated, then these fuzzy rules are examined to fine tune the initial membership functions, these tuned fuzzy membership functions are then used to generate a new set of rules. The complexity of the introduced algorithms is very high and it is only feasible to apply these approaches to simple system. Refer to section 2.1 for more details.

4.2 Extended IFID3 Algorithm

This section will start by introducing a new syntax of the fuzzy rules produced the fuzzy decision tree. Then it introduces and discusses a new threshold on the membership value for a data object to propagate down the tree from a parent node to a child node during the fuzzy decision tree construction step, we called this threshold "**Object Propagation Threshold"**. Finally this section will introduce a modified version of the Improved FID3.

4.2.1 A new Syntax for Fuzzy Rules

Since there are different methods to make inference in the fuzzy decision trees induction process, the new fuzzy rule syntax is intended to be independent of the inference method to be used. The new syntax will keep track of partial membership values of each class in a leaf node rather than assigning the rule the label of the class with the greatest membership value. In addition this syntax can be extended to include the truth degree of the rule. For example, assume we have a total of three classes in our dataset, and only two of these classes present in a certain leaf node, then the syntax of the rule representing this leaf node is as follows:

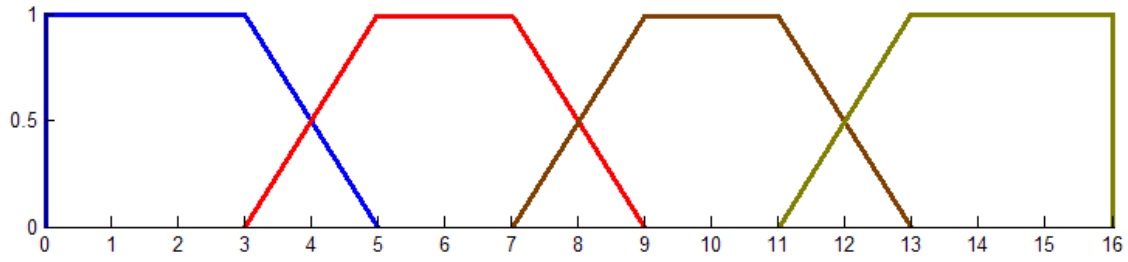
If fuzzyTerm0, fuzzyTerm1, …, FuzzyTermn, 0.05, 0.95, 0

Which means that if fuzzyTerm0, then if fuzzyTerm1, …, then if FuzzyTermn, then the object is 0.05 class 1, 0.95 class 2, and 0.0 class 3. This syntax moves the assignment of the class labels to the fuzzy rules from the tree construction step to the inference step. The advantage of this ap-

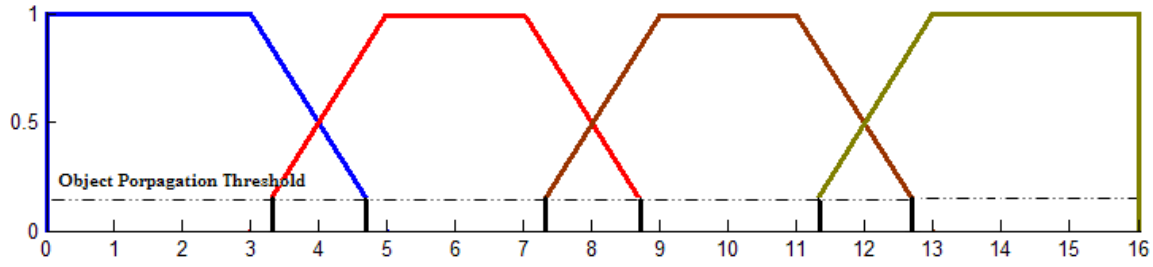
proach is that it allows the user to use the inference method of his preference without having to re-execute the tree construction step again and allows him to utilize the entire information in the leaf node in the decision tree. For example the user can label the rule with the class label of the class with the greatest membership value or apply the (X-X-+) inference approach without having to rebuild the tree and reconvert it into a new set of fuzzy rules.

4.2.2 Object Propagation Threshold

In order to reduce the number of fuzzy rules generated by the Improved FID3, we introduced a new threshold on the membership value for a data object to propagate down the tree from a parent node to a child node during the fuzzy decision tree construction procedure. This threshold was named the Object propagation threshold, and is used as follows: If the membership value of a certain object when propagated from a parent node to any of its child nodes is less than the object propagation threshold, the object will not propagate down to that child node. The justification for this threshold is that, if the object membership value is going to get smaller and smaller as it propagates down the decision tree then the effect of this object on the final decision is going to very low. In addition this threshold can be viewed as fine tuning of the membership functions. Figure 4.1 shows the effect of using this threshold on the initial membership functions. This effect is only applicable during the training phase (tree-building phase).



(a)



(b)

Figure 4.1: The effect of using the Object Propagation Threshold on fuzzy membership functions. (a) Original membership functions. (b) Membership functions after using the threshold. The dotted line represents the threshold values

4.2.3 Extended IFID3 Algorithm

The original FID3 algorithm and the IFID3 algorithm use only two thresholds. These two thresholds are used to stop the recursive partitioning of data at a tree node and to designate that node as a leaf node. The two thresholds are the leaf control threshold and fuzziness control threshold. The leaf control threshold is a threshold on the number of objects (data instances) in a tree node, if the number of objects at a tree node is less than this threshold the recursive partitioning of the node's data is stopped and the node is designated as a leaf node. The fuzziness control threshold is a threshold on the ratio of the sum of membership values of data instances belonging to a given class to sum of the membership values of data instances belonging to all classes at a given tree node. If the ratio of the sum of the membership values of data instances belonging to a particular class is greater than this threshold, the recursive partitioning of the node's data is stopped and the node is designated as a leaf node.

The Extended IFID3 extends the IFID3 algorithm by adding an additional threshold on the membership value of an object (data instance) in a node data to propagate from a parent node to a child node while building the fuzzy decision tree; this threshold was called the object propagation threshold. Unlike the leaf control threshold and the fuzziness control threshold, the object propagation threshold is not used to terminate the data partitioning, but is used as a filter that prevents data instances with membership values less than this threshold to propagate down to child nodes. The use of this threshold represents an online-tree-pruning procedure.

Given a data set D, with attributes A_1, A_2, …, A_L and a classified class C = {C_1, C_2, …, C_n} and fuzzy sets F_{i1}, F_{i2}, …, F_{im} for the attribute A_i (each attribute may have a different value of m). Let D^{C_k} be a fuzzy subset in D with class C_k, and let |D| be the sum of membership values in a fuzzy set of data D. The extended IFID3 works as follows:

1. Generate the root node with a fuzzy set of all training data with membership value of 1.
2. The node is a leaf node, if the fuzzy set of the data at the current node satisfies any of the following conditions:
    a. The number of objects in the node data set D is less than a given threshold; this threshold is cold leaf control threshold θ_n. that is:

    $$|D| < \theta_n.$$

    b. The proportion of a data set of any class C_k $|D^{C_k}|$ in the node data set D is greater than or equal to a given threshold. This threshold is called fuzziness control threshold θ_r. That is:

    $$\frac{|D^{C_k}|}{|D|} \geq \theta_r$$

    c. No more attributes are available for classification.
3. The class name assigned to a leaf node depends on the inference method to be used and is either the label of the class with the greatest membership value, or all the class labels along with their membership ratios in the node data set.

4. If the node does not satisfy any of the above conditions then do the following:
    a. If this node is the root node of the decision tree, then calculate the Attribute classification Ambiguity for all attributes in the dataset and select the attribute with the minimum attribute classification ambiguity as the test attribute.
    b. Else, the node is not the root node of the decision tree, calculate the fuzzy information gain for all attributes; and select the attribute that has greatest fuzzy information gain as the test attribute.
    c. Divide the fuzzy data set at the node into fuzzy subsets using the selected test attribute, with the membership value of an object in a subset set to the product of it's membership value in parent node dataset and the value of the selected attribute fuzzy term.
    d. For each of the fuzzy subsets produced in c do:
        a) For each data instance in the fuzzy subset

        If the membership of this data instance < object propagation threshold then remove this instance from the fuzzy set.

    e. For each of the subsets, generate a new node with the branch labeled with the fuzzy term of the selected attribute.
5. For each new generated node repeat recursively from step 2.

4.3 Experiments and Results

To compare the performance of the Extended IFID3 with that of IFID3, we used the same datasets described in the previous chapter. The datasets included six randomly-selected standard UCI-machine-learning repository datasets from different application fields and one non-standard dataset. The non standard dataset is an encoded version of the protein dataset RS126 pertains to helices. Table 4.1 which is a copy of table 3.1 describes the features of these datasets and is presented here for convenience.

Table 4.1: A brief description of the datasets used during the experiments

Dataset	Number of Samples	Number of Attributes	Number of Classes
Breast Cancer Wisconsin (Diagnostic)	569	32	2
Waveform Database Generator (VI)	5000	21	3
Statlog (Heart)	270	14	2
Ionosphere	351	34	2
Statlog (Landset Satellite)	6435	36	7
Glass	214	9	6
RS126 (H/~H)	21834	455	2

The datasets described above were divided into training and testing datasets. During the experiments; if the obtained data set is already divided into training and testing sets, we use the

existing partitions, otherwise the dataset is partitioned randomly into three folds, such that the ratio of objects belonging to different classes in the data folds is the same as that in the original data set and three-fold cross validation is used using the same IFID3 and Extended IFID3 parameters. Our experimental results show that using the extended IFID3 resulted in a huge reduction in the number of rules produced, and achieved a better accuracy compared to IFID3. Table 4.2 compares the number of rules and accuracy of both Extended IFID3 and IFID3 [47]. The numbers in the table represent the average results of the 3-fold cross validation.

Table 4.2: IFID3 and Extended IFID3 Performance Comparison based on number of produced rules and accuracy

Data Set	Number of Rules, IFID3	Accuracy IFID3	Number of Rules, Extended IFID3	Accuracy Extended IFID3	Relative size of rule set in %
Breast Cancer Wisconsin (Diagnostic)	2994	0.977	94	0.981	3.14
Waveform Database Generator (VI)	106053	0.832	10952	0.834	10.33
Statlog (Heart)	79	0.819	38	0.823	48.10
Ionosphere	88082	0.940	3284	0.943	3.73
Statlog (Landset Satellite)	233306	0.880	16245	0.892	6.96
Glass	291	0.706	222	0.730	76.29
RS126 (H/~H)	681	0.759	616	0.780	90.46

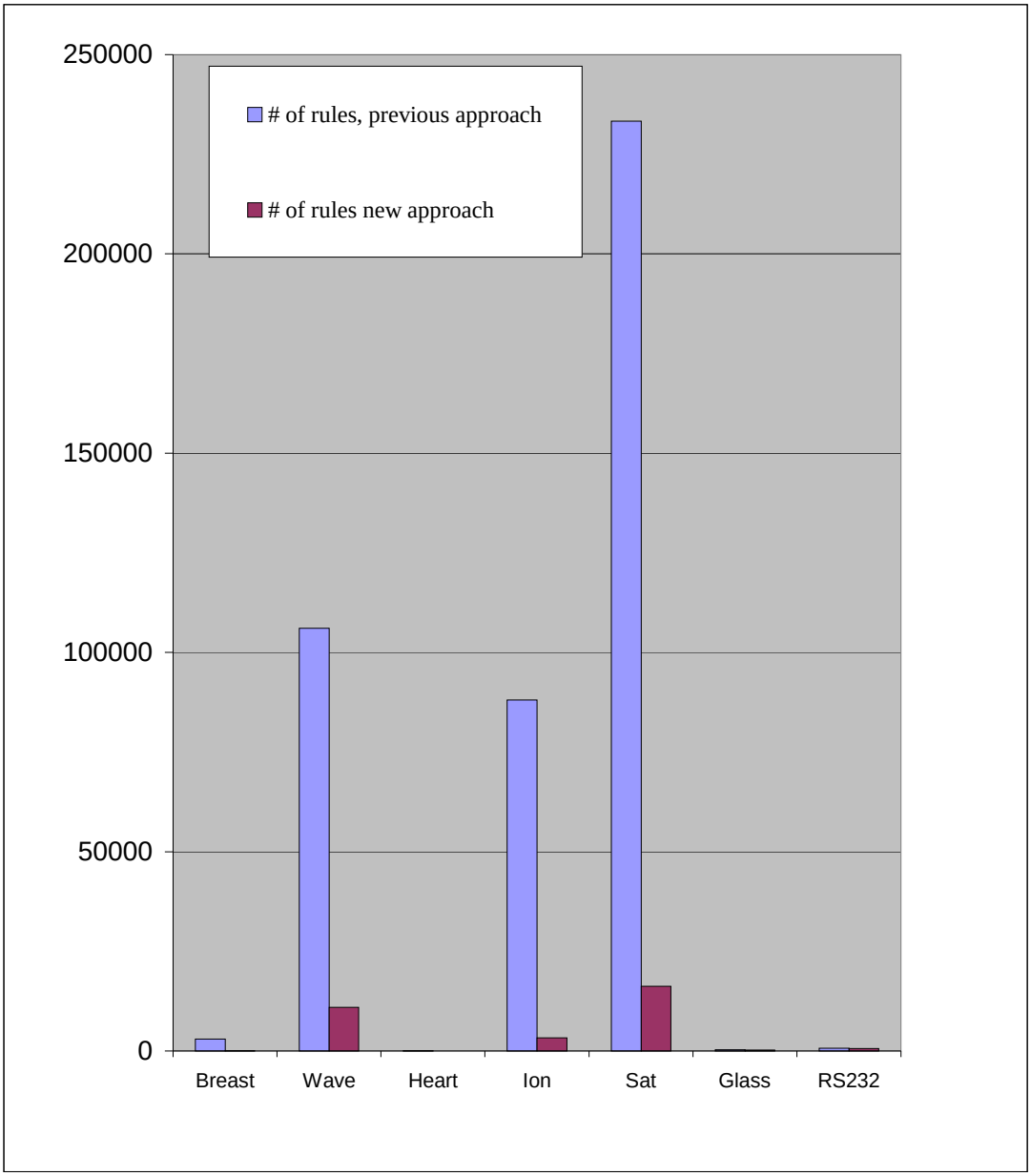


Figure 4.2: Comparing the number of rules produced by EIFID3 algorithm to those generated by the IFID3 algorithm

As mentioned earlier, since the algorithm execution time, especially the inference part is proportional to the size of the generated rule base, the Extended IFID3 executes faster. Table 4.3 compares the execution times of both the IFID3 and the Extended IFID3 algorithms [47]; the execution times represent the average of the three experiments. Since the reduction in execution time is not noticeable by the user on small datasets, table 4.3 [47] shows only the reduction in execution time on relatively large datasets.

Table 4.3: Execution Times (in Seconds) for the Extended IFID3 algorithm and the IFID3 algorithm

Data Set	Execution Time(s) of IFID3	Execution Time(s) Extended IFID3
Breast Cancer Wisconsin (Diagnostic)	5.33	< 1
Waveform Database Generator (VI)	1753.33	106
Statlog (Heart)	425	5.33
Ionosphere	8441	258
Statlog (Landsat Satellite)	268	256

4.4 Summary and Conclusion

In this chapter we presented a modified version of the Improved FID3. the presented algorithm uses, in addition to the two thresholds used in the original ID3, an additional threshold on the membership value of an object in a tree node dataset to propagate to a child node, if the membership value of the object is less than or equal to the given threshold, the data item is dropped. The experimental results showed that the modified version of IFID3 can produce better accuracy and can achieve a significant reduction in the number of resulting fuzzy rules.

In addition we presented a new syntax for the rules; the new syntax incorporates the class membership ratio in a leaf node; using the new syntax makes the extended IFID3 independent of the inference method to be used. We also compared two inference methods, the one in which the leaf node is assigned the name of the class with the greatest membership value, and the one in which the leaf node is assigned the names of all classes in the leaf node with the ratio of their membership values. The experimental results showed that none of the two inference methods is superior to the other.

Reducing the number of rules required for a decision is an important result of this work, both because it increases the computational performance of the program and for the more fundamental reason that it improves the falsifiability of decision model and improves it's interpretability. The likelihood that the rules actually reflect an experimentally validateable or causal relationship between the independent and dependent variables of the problem is much higher with a smaller, and often dramatically smaller, rule base.

Chapter 5

PREDICTION AND CLASSIFICATION OF REAL AND PSEUDO MICRORNA PRECURSORS VIA DATA FUZZIFICATION AND FUZZY DECISION TREES

MicroRNAs (miRNAs) are short non-coding RNA molecules that play a significant role in post-transcriptional gene regulation. Despite the fact that hundreds of miRNAs have been identified, recent studies indicate that more miRNAs exist. Since current experimental approaches to identify novel miRNAs are expensive and time-consuming, computational methods can play an important role in identifying miRNAs candidates for further experimental validation. Most computational approaches utilize features extracted from miRNA precursors (pre-miRNA) sequences and/or their secondary structures to detect miRNAs. A key characteristic of pre-miRNAs is their hairpin structure.

In this chapter, Fuzzy decision trees are applied to the prediction and classification of real and pseudo pre-miRNAs. In our model, a number of features that encode local and global characteristics of pre-miRNA sequence structure are used. A fuzzy model of the extracted features was constructed. The fuzzified data was then fed into a fuzzy decision tree induction algorithm. Our experimental results showed that our method achieved better accuracy than other machine-learning based computational approaches. Analyzing the results revealed that one of

the features –the sequence length to number of basepairs ratio - is very critical to the classification and identification of pre-miRNAs.

5.1 Introduction and Background

MicroRNAs are short non-coding RNAs that play important roles in gene regulation by targeting messenger RNAs (mRNAs) for cleavage or translational inhibition [22] [23] [24] [25] [26] [27]. In addition, microRNAs have been reported to play a role in oncogenesis [48] [49] [50] and to be involved in heart development and diseases [51] [52] [53].

Despite the fact that hundreds of miRNAs have been identified [23], recent studies indicate that many more miRNAs still undiscovered. For example, in [57], the researchers argued that the number of miRNAs encoded in a mammalian genome is around a thousand.

Although, experimental approaches to identifying new miRNAs are time consuming and expensive [25], hundreds of miRNAs have been cloned since the discovery of the very first miRNA. However, only abundant miRNAs can be easily detected by PCR or northern blot due to limitations in those techniques [54]. To overcome the disadvantages of the experimental approaches, computational methods are used to complement experimental approaches [24] and to identify potential miRNAs candidates for further experimental validation [25].

The formation of mature miRNAs starts in the nucleus [25] [54] [55], where larger RNA molecules called primary miRNAs (pri-miRNAs) are processed into hairpin structures called miR-

NA precursors (pre-miRNAs) by nuclear RNase III Drosha [55]. The pre-miRNAs are then transported to the cytoplasm by Exportin-5 [22] [56] [55]. In the cytoplasm another RNase III Dicer cuts these pre-miRNAs to release the ~22 nt mature miRNAs [22] [54]. Figure 5.1 shows the microRNAs Biogenesis [58].

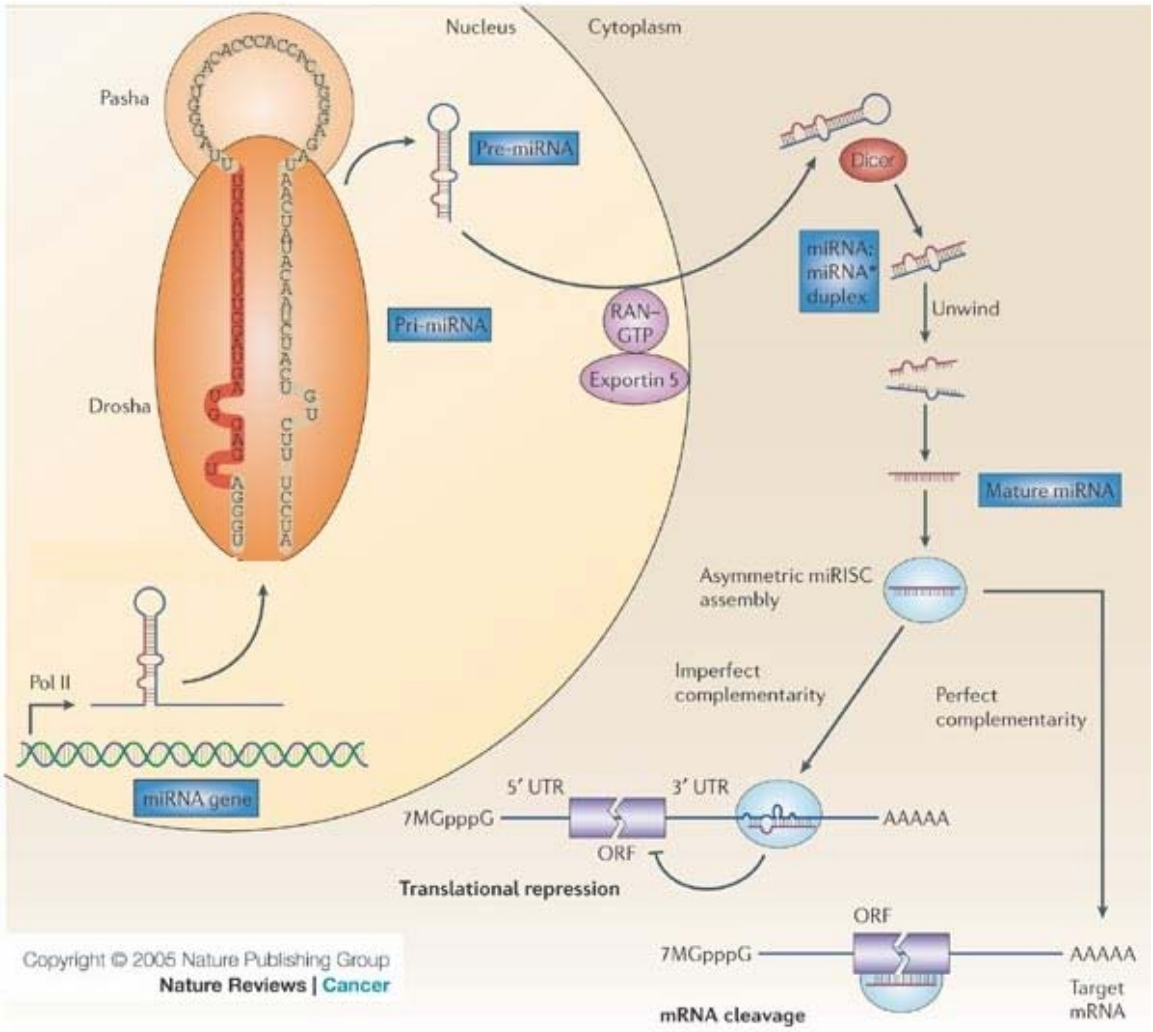


Figure 5.1 [58]: MicroRNA Biogenesis

Since miRNA precursors (pre-miRNAs) and their secondary structure encode more discriminative information than their corresponding mature miRNAs [23] [25]; most computational techniques utilize features extracted from pre-miRNAs sequences and/or their secondary structure [23] [25]. For example, MiRscan [22], a computational tool for miRNAs prediction, utilizes the observation that known miRNAs are derived from phylogenetically conserved stem-loop precursor RNA.

Computational techniques for miRNAs prediction can be grouped into two main categories [25]: gradually hierarchical methods and discriminative ones. Gradually hierarchical methods combine comparative genomics information with features extracted from pre-miRNA sequences and their secondary structure to predict miRNAs. On the other hand, discriminative approaches utilize machine learning to build a classifier, the classifier is trained on features extracted from positive and negative pre-miRNA sequences and/or their secondary structures. The fuzzy decision tree approach falls into this category.

In previous work targeting the prediction of pre-miRNAs, researchers have applied Support Vector Machines (SVM) to the problem. SVM was trained on features extracted from positive and negative pre-miRNA sequences and their secondary structure to build a classifier. The main disadvantage of using SVM is that it is a black box approach, the decision model upon which SVM draws the final decision is unknown to the human user and therefore, it is hard to identify the critical features to this classification problem.

In the past few years, a number of algorithms have been proposed for classifying pre-miRNAs. In [22] a set of local features were extracted from the pre-miRNA sequences and their predicted secondary structure. Support vector machines (SVM) was applied to dataset with the extracted features to build a classifier to classify real and pseudo pre-miRNAs. In [23] a method for identifying clustered miRNAs was introduced. In [25] a string kernel and support vector machines were proposed to identify miRNAs. In [26] the authors combined features consisting of the local structure-sequence, the minimum of free energy of the secondary structure and the P-value of randomization test. Then they used random forest to classify real and pseudo pre-miRNAs. In [54] a computational tool to identify miRNAs was introduced; it uses pre-miRNA sequence and structure alignment to identify miRNAs. In [59] a computational tool called miR-seeker is introduced, it is used to identify miRNAs in Drosophila.

The work presented in this chapter combines the local pre-miRNA features defined by the triplet elements introduced in [22], and the global pre-miRNA features introduced in [27] to encode the pre-miRNA sequences and the predicted secondary structure associated with them, the encoded data is fed into a fuzzification module. The fuzzified data is then fed into a fuzzy-decision-tree induction tool [39] [47] to generate a set of fuzzy rules. The produced fuzzy rules are then used to classify and predict real and pseudo pre-miRNAs.

5.2 Encoding the Pre-miRNA Hairpin Structure

In order to apply the fuzzy decision tree machine learning tool to the classification/prediction of real and pseudo pre-miRNA sequences, the pre-miRNA sequences and their associated predicted secondary structure need to be encoded into a set of features. RNAFold [60] was used to predict the secondary structure associated with these sequences, and then the sequence structure and the associated secondary structure were encoded into a set of forty three features including eleven global features and thirty two local ones. The global features were proposed in [27]. In order to extract these features, the pre-miRNA hairpin structure is divided into two arms, the left arm starts at the 5' end and ends at the center of the central loop, the rest of the nucleotides in the pre-miRNA sequence form the right arm. The eleven global features and their definitions are given below:

1. Symmetric Difference: The symmetric difference is the difference in the length of the two arms.
2. Bulge Size: The bulge size represents the size of the largest mismatch region on the two arms.
3. Bulge number: The number of bulges in the arm containing the larger number of bulges.
4. Tail length: The tail length represent the length of the longer free tail of the two arms.

5. Free energy per nucleotide: The free energy per nucleotide is obtained by dividing the free energy calculated by RNAFold by the number of nucleotides in the sequence.
6. Number of basepairs: Number of basepairs in the sequence.
7. GC Content.
8. Sequence length: Number of nucleotides in the sequence.
9. Length-basepair ratio: this feature is obtained by dividing the sequence length by the number of basepairs.
10. Central loop length: Number of nucleotides in the central loop.
11. Number of tails.

The thirty two local features are defined by the triplet elements introduced in [22]. A triplet element is defined as one nucleotide and the secondary structure associated with its -1, 0, +1 positions, in other words the nucleotide, and the secondary structure associated with it and the secondary structures associated with it's left and right neighbor nucleotides. Since there are four nucleotides: A, C, G, U and only two possible secondary structures: match '(' and mismatch '.', there are 4 X 2^3 possible triplet elements. The count values of them are used as the values of 32 local features. For example the features associated with nucleotide A are: A…, A..(, A.(., A.((, A(.., A(.(, A((. and A(((. To find the values of each of these features, we count the number of occurrences of these patterns in the sequence and its associated secondary structure. For example if the pattern A(((and the associated secondary structure occurs 5 times in the se-

quence, the feature A(((is assigned a value of 5. This encoding scheme is shown in Figure 5.2 [22].

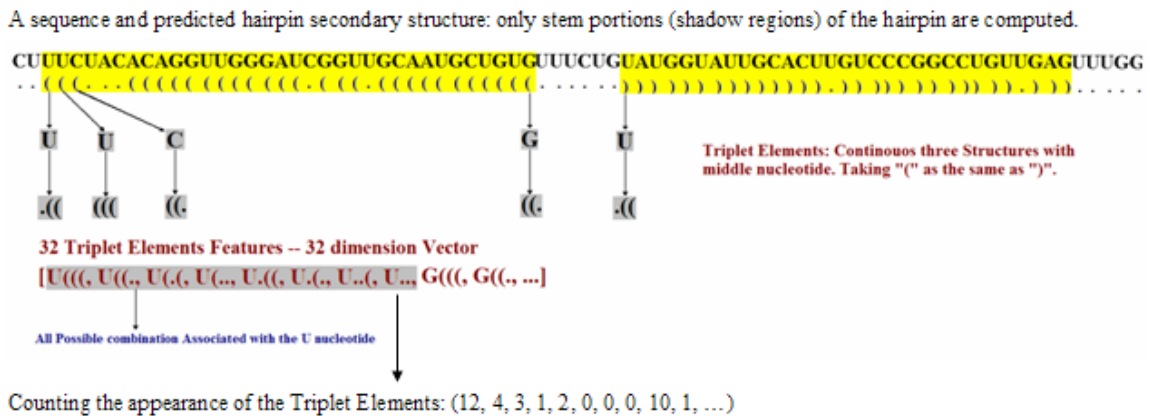


Figure 5.2 [22]: Triplet Elements Encoding Scheme

5.3 Algorithm for Classifying Real and Pseudo pre-miRNAs

In this section, we discuss the application of fuzzy decision trees to the classification and prediction of real and pseudo pre-miRNAs and we present our algorithm. After encoding the sequence and its' associated secondary structure into the 43 features described in the previous section; the resulting encoded dataset is fed into our fuzzy decision tree induction tool; this tool supports only triangular and trapezoidal fuzzy membership functions. All the features' ranges were divided evenly into an equal number of fuzzy sets. The output of this step is a fuzzified version of the original dataset. The fuzzified dataset is fed into our fuzzy decision tree tool [39] [47] to construct a fuzzy decision tree. Then the produced tree is converted into a set of fuzzy

rules. To perform inference about test instances, the testing dataset was fuzzified using the same fuzzy parameters that were used to fuzzify the training dataset. Then the fuzzified testing dataset was tested against the fuzzy rules generated during the training step. Since each of the generated fuzzy rules includes partial membership information of the object firing the rule in each of the classes in the training data set, the partial membership information of the fired rules is recorded. For example, the resulting decision of applying a certain fuzzy rule may be 0.1 pre-miRNA and 0.9 background. To make the final decision on a certain sequence the weights of the partial class membership information assigned by all fired rules are added and the class with the larger weight is considered as the final decision. For Example the sum of the weights assigned by all fired rules may be 0.2 pre-miRNA and 0.8 background, in this case the final decision will be background. Figure 5.3 shows the follow of the suggested method [61]. Following is a summary of the steps of our method:

1. Extract the 43 features from the training and testing datasets using the encoding procedures proposed in [22] and [27].
2. Build a fuzzy model to be used to fuzzify the training dataset. This includes specifying the number of fuzzy sets to be used per feature and the fuzzy membership function.
3. Use the fuzzy model built in step 2 to fuzzify the training dataset.
4. Apply the fuzzy ID3 algorithm to the fuzzified training dataset to build a fuzzy decision tree.
5. Convert the fuzzy decision tree generated in step 4 into a set of fuzzy rules.
6. Use the fuzzy model built in step 2 to fuzzify the testing dataset.

7. Apply the fuzzy rules to each sequence in the testing dataset to classify it.
8. Use the inference-by-partial-membership method (X-X-+) to make the final classification decision.

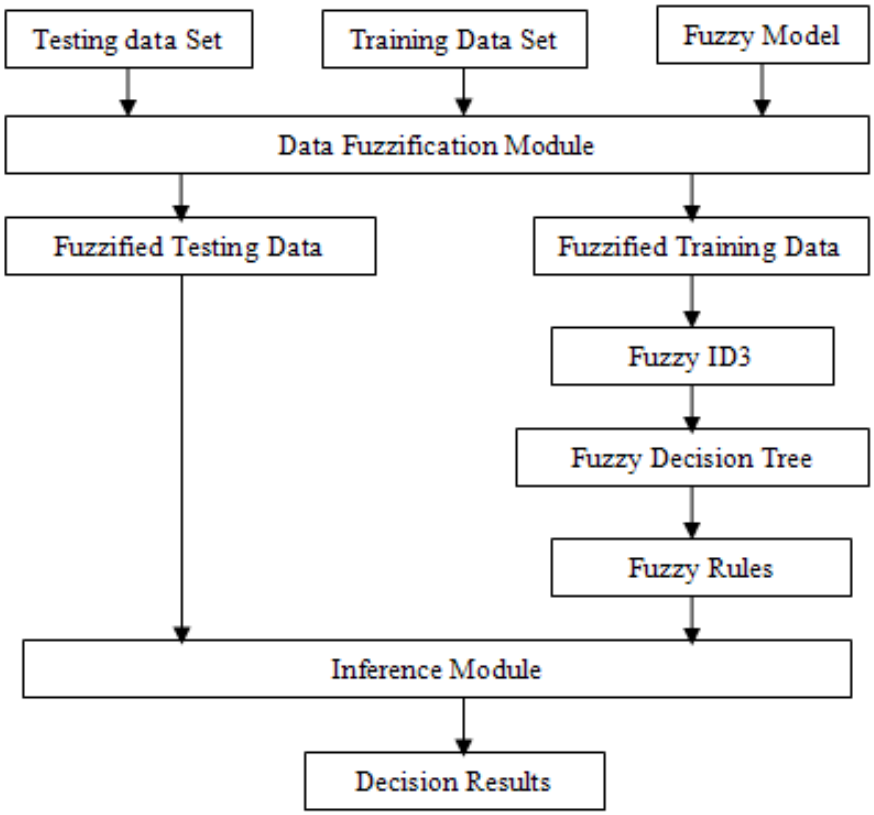


Figure 5.3: Flow chart for classification of real and pseudo pre-miRNAs using Fuzzy Decision Trees. In the figure, the datasets contain the 43 features.

5.4 Experiments and Results

5.4.1 Datasets

In order to compare the performance of our approach to previous approaches introduced by other researchers, we used the same datasets used in literature [22] [27] and compare our results with theirs. Table 5.1 describes these datasets briefly. In table 5.1, Datasets 1 through 5 are from human, and datasets 6 through 16 are from other species. The encoded datasets were downloaded from [62]. These raw positive microRNA samples were obtained from the microRNA registry database version 5.0 (Rfam 5.0) [65] [66]. Only sequences with a single loop were used. The negative samples were obtained from the protein coding regions of human RefSeq genes and the genome region of positions 56,000,001 – 57,000,000 on the human chromosome 19. For more details about these datasets and the preprocessing steps of the raw data, the reader is referred to [22] and [27]. Finally, it is worth mentioning that these datasets have some errors, for example dataset 4 (Conserved-Hairpin (T3)) contains three known positive samples that were verified experimentally. To further validate our approach, we tested it on Rfam database release 12.0 (Rfam 12.0) [63] [64] [65] [66]. Rfam 12.0 was the most recent microRNA registry database at the time of this work and contains 8619 pre-microRNA entries; all sequences in this release were used including those with multiple loops. The secondary structure for these sequences was predicted using RNAfold program. Then the 43 features were extracted from the sequences and their associated secondary structure.

Table5.1: Summary of Datasets [6]

	Dataset	Number of Samples	Class
1	TR-C	163/168	Pre-miRNAs/background
2	TE-C1	30	Pre-miRNAs
3	TE-C2	1000	Background
4	Conserved-Hairpin (T3)	2444	Background
5	Updated(T4)	39	Pre-miRNAs
6	Mus musculus	36	Pre-miRNAs
7	Rattus norvegicus	25	Pre-miRNAs
8	Gallus gallus	13	Pre-miRNAs
9	Danio rerio	6	Pre-miRNAs
10	Caenorhabditis briggsae	73	Pre-miRNAs
11	Caenorhabditis elegans	110	Pre-miRNAs
12	Drosophila pseudoobscura	71	Pre-miRNAs
13	Drosophila melanogaster	71	Pre-miRNAs
14	Oryza sativa	96	Pre-miRNAs
15	Arabidopsis thaliana	75	Pre-miRNAs
16	Epstein barr virus	5	Pre-miRNAs

5.4.2 Parameter Optimization and Fuzzy Model Selection

Fuzzy ID3 algorithm has two parameters; the fuzziness control threshold θ_r and the leaf decision threshold θ_n. In order to optimize these parameters and to select an optimal fuzzy model (membership functions), the training dataset was divided into 10 folds, 9 folds were used as a training dataset and the 10th fold as a validation dataset (10-fold cross validation). The values for the fuzziness control parameter were varied between 0.85 and 1 in step of 0.01, and the values for the number of objects at a node threshold (leaf control threshold) were varied between 5 and 200 in step of 5. For each pair of values of these two thresholds triangular and tra-

pezoidal fuzzy membership functions were tested. To decide on the set of optimal membership functions, each of the features range was divided evenly into an equal number of fuzzy sets, the number of fuzzy sets was varied between 2 and 15. For each combination of the three parameters: Fuzzy model (number of fuzzy sets, trapezoidal or triangular membership function), leaf control threshold and fuzziness control threshold, the experiment was executed 10 times by changing the fold used for validation. The results were averaged and the parameters that maximize the accuracy were selected. We also varied the information measure used to select the branching attribute, we tries information gain, classification ambiguity and the algorithm presented in chapter 2 which uses classification ambiguity and information gain, of the three, the EIFID3 achieved the best results. Based on the results of these experiments, 4, 8 and 12 fuzzy sets achieved comparable results. In all the experiments we tested 4, 8 and 12 fuzzy sets with both triangular and trapezoidal membership functions. The results reported below are the best ones.

5.4.3 Performance Evaluation Measures

Three measures accuracy, sensitivity and specificity were used to evaluate the results of our experiments. The definition of these three measures is given below:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN) \qquad 5.1$$

$$\text{Sensitivity} = TP/(TP + FN) \qquad 5.2$$

$$\text{Specificity} = TN/(FP + TN) \qquad 5.3$$

Where:

- True positives (TP): Positive samples predicted as positive.
- False positives (FP): Negative samples predicted as positive.
- True negatives (TN): Negative samples predicted as negative.
- False negatives (FN): Positive samples predicted as negative.

5.4.4 Experimental Results

In order to evaluate our approach and compare it to the results of other previous approaches, we conducted several experiments. During all the experiments we used the same datasets used in the literature [22] [27]. In the first experiment, dataset 1 (TR-C) was used as a training dataset and datasets 2 through 16 were used as the testing dataset. Each feature was divided evenly into 4 fuzzy sets with a triangular fuzzy membership function. The triangular fuzzy membership functions of adjacent fuzzy sets cross each other at 0.5. Our method achieved a 93.6% overall accuracy, 82.62% sensitivity and a 95.64% specificity. Table 5.2 shows the overall accuracy of the fuzzy decision tree (FDT) approach compared to the other machine learning approaches used in the literature [22] [27] and posted on [62] in addition to random forest (RF). All the entries in this table except the entries for RF and FDT were obtained from the literature [62]. In the table kNN refers to the k-Nearest-Neighbors algorithm [67].

Table 5.2: Overall Accuracy achieved by our method compared to those reported in the Literature. Datasets 2 through 16

	SVM	C4.5	kNN	Ripper [68]	Tri-SVM[1]	RF	FDT
Accuracy	93.1	88.9	88.9	87.2	89.1	90.7	93.6

1: Results achieved by Tri-SVM using the triplet elements features only. This result was reported by [1].

In the second experiment, Dataset 1 (TR-C) was used as the training dataset and datasets 2 through 5 were used as a testing dataset. Both the training and the testing datasets were obtained from humans. The fuzzy membership functions are the same as those used in the first experiment. Our method achieved a **95.64%** overall accuracy, **95.65%** sensitivity and **95.64%** specificity. Table 5.3 [61] compares the results achieved by our method to those reported in the literature. All the entries in table 5.3 except for RF and FDT were obtained from the literature. Table 5.4 [61] shows the accuracy achieved by our method on individual human datasets.

Table 5.3: Fuzzy Decision Trees versus Other machine learning approaches

Human: Datasets 2 through 5

	SVM	C4.5	kNN	Ripper	Tri-SVM[1]	RF	FDT
Sensitivity	94.2	94.2	94.2	94.2	92.8	95.6	95.6
Specificity	93.3	89.7	88.8	87.0	88.7	90.9	95.6
Accuracy	93.3	89.8	88.9	87.1	90.9	91.0	95.6

1: Results achieved by Tri-SVM using the triplet elements features only. This result was reported by [1].

Table 5.4: Accuracy achieved by FDT on individual human datasets 2 through 5

	Dataset	Number of Samples	Accuracy
2	TE-C1	30	100
3	TE-C2	1000	97.4
4	Conserved-Hairpin (T3)	2444	94.9
5	Updated(T4)	39	92.3

In the third experiment, Dataset 1 (TR-C) was used as the training dataset and datasets 6 through 16 were used as a testing dataset. All instances in the testing dataset were obtained from other species. In this experiment, the range of each feature was divided evenly into eight fuzzy sets with a trapezoidal fuzzy membership function. The trapezoidal fuzzy membership functions of adjacent fuzzy sets cross each other at 0.5. Our method achieved a 95.52% overall accuracy and 95.52% sensitivity. Since the testing dataset has no negative samples, specificity cannot be calculated and the accuracy and sensitivity are equal. Table 5.5 [61] compares the results achieved by our method to those reported in the literature. All the entries in table 5.5 except for RF and FDT were obtained from the literature. Table 5.6 [61] shows the accuracy achieved by our method on individual datasets.

Table 5.5: Fuzzy Decision Trees versus Other machine learning approaches

Other Species: Datasets 6 through 16

	SVM	C4.5	kNN	Ripper	Tri-SVM[1]	RF	FDT
Sensitivity	91.7	83.3	87.8	87.8	90.9	89.2	95.5
Accuracy	91.7	83.3	87.8	87.8	90.9	89.2	95.5

1: Results achieved by Tri-SVM using the triplet elements features only. This result was reported by [1].

Table 5.6: Accuracy achieved by FDT on individual other species datasets 6 through 16

	Dataset	Number of Samples	Accuracy
6	Mus musculus	36	94.4
7	Rattus norvegicus	25	84.0
8	Gallus gallus	13	92.3
9	Danio rerio	6	83.3
10	Caenorhabditis briggsae	73	95.9
11	Caenorhabditis elegans	110	93.6
12	Drosophila pseudoobscura	71	97.2
13	Drosophila melanogaster	71	95.8
14	Oryza sativa	96	100
15	Arabidopsis thaliana	75	100
16	Epstein barr virus	5	40.0

In the fourth experiment, Datasets 6 through 16 (Datasets from other species) and negative sample obtained from human Dataset 3 (TE-C2) were used as a training dataset, and dataset 1 (TR-C), dataset 2 (TE-C1), dataset 4 (Conserved-Hairpin) and dataset 5 (Updated) were used as a testing dataset. In this experiment, the range of each feature was divided evenly into twelve fuzzy sets with a trapezoidal fuzzy membership function. The trapezoidal fuzzy membership functions of adjacent fuzzy sets cross each other at 0.5. Our method achieved a 96.2% overall accuracy, a 91.8% sensitivity and a 96.6% specificity.

Finally, to further evaluate our method, we tested it on the most recent Rfam database at the time of this work Rfam Release 12.0. Since the Rfam dataset contains only positive samples, it was combined with the negative samples from TE-C2 dataset (1000 sequences), T3 dataset (2444 sequences) and TR-C dataset (168 sequences). The resulting dataset was then divided randomly into three folds, such that the ratio of negative samples to positive samples in each of

the three folds is the same as that in the entire dataset. Three-fold cross validation was then used and the results were averaged. Our method achieved an average accuracy of **92.28%**, an average sensitivity of **93.92%** and an average specificity of **88.88%**.

5.4.5 Discussion

Analyzing the results reported in the previous subsection shows that the pre-miRNA structure is conserved across species. This conclusion is based on the observation that in cases where we used pre-miRNA obtained from humans, we were able to classify samples obtained from other species with a very high accuracy and vice versa. However, the difference in the fuzzy model (number of fuzzy sets and fuzzy membership function) may indicate slight differences across species or may be due to different error levels during the experimental identification of the pre-miRNA samples.

Examining the fuzzy decision trees produced during all the experiments and the corresponding fuzzy rules revealed the significance of the length basepair ratio (Length of the sequence / the number of basepairs) feature to the classification decision. By using only this feature, our method when trained on dataset 1 and tested on all other datasets achieved a 92.1% accuracy, 65.7% sensitivity and 97.0% specificity. A deep examination of the rules revealed that **if the length basepair ratio value associated with a given sequence is greater than 3.9, then this sequence does not represent a pre-miRNA**. On the other hand, if the value of the length

basepair ratio feature is less than or equal to 3.9, then the sequence may represent either a pre-miRNA sequence or a background sequence depending on the fuzzy membership values of the sequence in each of the three overlapping fuzzy sets spanning the range 0 to 3.9, furthermore, **if the length basepair ratio value is less than or equal to 3.9, the likelihood of the sequence to represent a pre-miRNA increases as the value of length basepair ratio feature decreases**. A further study on release 12.0 of the Rfam database showed that **95.86%** of the pre-miRNA sequences in this database have a length basepair ratio less than or equal to 3.9.

To further investigate the significance of the length basepair ratio feature to the classification of real and pseudo pre-miRNAs, we removed this feature from all the datasets, and then we repeated the first three experiments. In the first experiment, the accuracy went down from 93.6% to 85.3%, in the second experiment our method achieved a 96.64% accuracy, a 76.81% sensitivity and a 97.0% specificity, and in the third experiment the overall accuracy and sensitivity were only 64.4%. These results indicate that the length basepair ratio feature has a significant effect on the classification results, and that by using this feature, our method was able to avoid many of the false positives.

The set of features include a number of dependent features. For example, the length basepair ratio feature is a function of another two features in the dataset, the sequence length and the number of basepairs. Therefore, we believe that removing either one of these two features from the datasets should not affect the results significantly. When the sequence length feature was removed from the datasets, repeating the first experiment achieved the same re-

sults. On the other hand removing the two features from the datasets affects slightly the results, for the first experiment the accuracy went down from 93.6% to 93.3%.

5.5 Conclusions and Future Work

In this chapter, a new method to classify real and pseudo pre-microRNAs was introduced. The new method utilized fuzzy decision tree as a classification tool. The sequences and their associated secondary structure were encoded using the same procedures introduced in [22] and [27]. Based on the results of the experiments, fuzzy decision trees achieved better results than other machine-learning tools applied to the problem. Analyzing the experimental results shows that the pre-miRNA structure is conserved across species.

Investigating the set of rules produced in the experiments, revealed that the length basepair ratio feature was always selected to be the branching feature at the root node of the constructed fuzzy decision tree. This indicates the importance of this feature in the classification process. Further investigation showed that by using only this feature we can exclude many of the negative samples.

The data fuzzification model plays an important role in determining the performance of our method. We believe that the performance of our method can be significantly improved by carefully designing the data fuzzification model. We hypothesize that the more accurate the fuz-

zification model in reflecting the uncertainty or ambiguity in the data the more effective the fuzzy-decision-tree will be as a decision tool.

The triplet elements' encoding scheme introduced in the literature showed to encode discriminative features. However, the use of this encoding scheme was not justified. We propose to further study this encoding scheme. We also believe that some important information may be lost during the data encoding and preprocessing steps. We believe that a fuzzy string kernel capable of processing the raw sequence data and/or secondary structure may result in a significant improvement in the performance of our method.

Chapter 6

IDENTIFYING ESSENTIAL FEATURES FOR THE CLASSIFICATION OF REAL AND PSEUDO MICRO-RNA PRECURSORS USING FUZZY DECISION TREES

In the previous chapter we discussed the application of Fuzzy Decision Trees to the classification of real and pseudo pre-microRNAs. The experimental results presented in the previous chapter showed that FDTs can achieve better performance than other machine-learning approaches applied to the same problem with the advantage of producing human interpretable rules. In this chapter, we extend the work presented in the previous chapters in two ways: First we investigated the rules to identify the features that were selected during the fuzzy tree construction procedure, and then we apply several machine learning tools to datasets with only the selected features. In other words we used the fuzzy decision tree algorithm as a feature selection algorithm. Second, since the use of the triplet elements was not justified, we further investigated it. To do so we extended the triplet elements scheme by varying the number of secondary structure positions used to encode the features, and we studied the classification results achieved when utilizing features extracted by using any of its extensions. We called the triplet element extended encoding scheme the triplet element sliding window encoding scheme. Our results showed that the triplet elements scheme does not encode more discriminative information than any of its' extension and that fuzzy decision trees can be used effectively for feature selection.

6.1 Extended Triplet Elements Sliding Window Encoding Scheme

The extended triplet elements sliding window encoding scheme is an extension to the triplet elements encoding scheme introduced in [22] and discussed in the previous chapter. In the triplet elements encoding scheme the target nucleotide and its' secondary structure and the secondary structure associated with its' right and left neighbor nucleotides are used as features. Since there are four nucleotides and two possible secondary structures match "(" and mismatch ".", the scheme results in a total of 32 features. The number of occurrences of each of these features in a given sequence and the associated predicted secondary structure represent the value assigned to that feature.

The extended triplet elements sliding window encoding scheme extends the triplet elements encoding scheme by varying the number of secondary structure positions to the left and right of the target nucleotide used in encoding the features. For example, for window size of 5, the target nucleotide and its' secondary structure and the secondary structure associated with the target nucleotide's two left neighbors and two right neighbors are used to encode the features. Since each secondary structure position has two possible values (match and mismatch), a window size of 5 results in 128 features. See figure 6.1. Generally the number of features associated with a given window size is given by the following formula:

$$\text{Number of Features} = 2^{w} \times 4 \qquad 6.1$$

Where w is the window size.

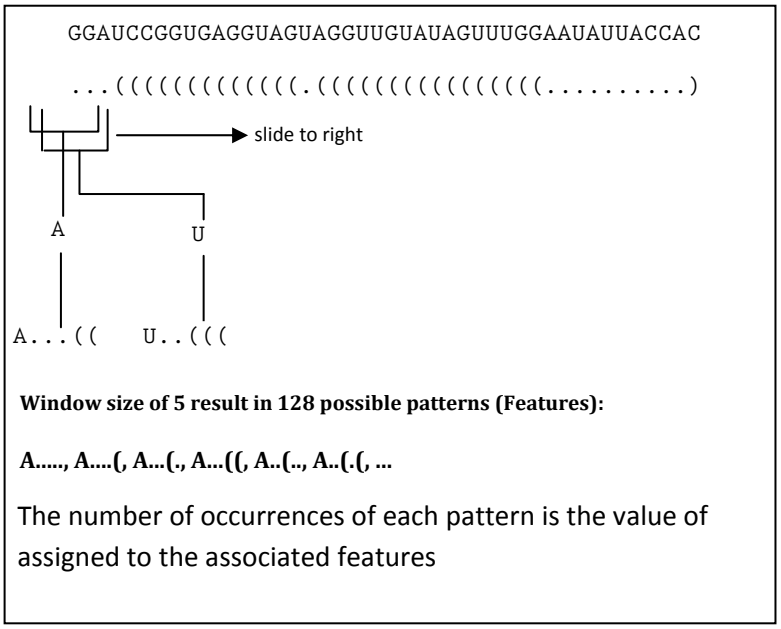


Figure 6.1 [69]: Sliding Window Encoding Scheme for a Window Size of 5.

6.2 Method

In order to identify the essential features for the classification of real and pseudo pre-microRNA sequences, we investigated the fuzzy rules generated in each of the experiments. We also compared the results obtained by using the triplet elements encoding scheme with the results obtained by using its' extensions. The same method described in chapter five was used here. First, RNAFold was used to predict the secondary structure associated with a given pre-mircroRNA or pseudo pre-microRNA sequence. Second, the sequence and its' associated-predicted secondary structure were encoded into a set of local and global features. The global features are those that were described in the previous chapter. The local features are those encoded using the triplet elements or its' sliding window extension. During the experiments the

window size in the sliding window scheme was varied between 1 and 13. Several machine learning tools were applied to the resulting encoded datasets and the results were compared.

To identify the essential features for classifying real and pseudo pre-microRNAs, the fuzzy rules produced by each of the experiments were analyzed, as a result of this analysis six features were identified as being essential to this classification problem. Several machine learning tools were applied to dataset with only these features and the results were compared with the results achieved by using the same dataset with all features. The experiments and results section will address this in more details.

6.3 Experiments and Results

6.3.1 Datasets

In these experiments we used the same datasets described in chapter 5, section 5.4. The datasets were obtained from Rfam 5.0. Datasets in table 5.1 from other species were combined into a single dataset. Table 6.1 describes these datasets briefly.

Table 6.1: Summary of Datasets

	Dataset	Number of Samples	Class
1	TR-C	163/168	Pre-miRNAs/background
2	TE-C1	30	Pre-miRNAs
3	TE-C2	1000	Background
4	Conserved-Hairpin (T3)	2444	Background
5	Updated(T4)	39	Pre-miRNAs
6	Other Species	581	Pre-miRNAs

In addition we performed experiments on datasets obtained from Rfam database release 12.0 (Rfam 12.0).

6.3.2 Parameter Optimization

During the experiments several machine learning tools were used including: Support Vector Machines (SVM), Random Forest (RF), C4.5 and Fuzzy Decision Trees (FDT). The parameters for each of these tools were optimized using a grid search approach. The results presented below are the best results achieved by each of these tools.

6.3.3 Results and Discussion

In order to evaluate the sliding window encoding scheme, several experiments were conducted. In all the experiments, unless otherwise mentioned, dataset 1 from table 6.1 was used as the training dataset and datasets 2 through 6 were used as the testing datasets. In the first experiment we combined the global features with the local features obtained using the sliding window scheme, with window sizes 1, 5, 7, 9, 11 and 13.

Table 6.2: Classification Results achieved by FDT using the global features combined with the local features extracted using the specified window size

Window Size	Accuracy	Sensitivity	Specificity
1	92.79	92.77	92.8
3	93.6 [61]	82.62 [61]	95.64 [61]
5	92.9	88.9	93.7
7	94.1	91.4	94.6

Table 6.2 shows the results of these experiments and compare the results with those of the triplet elements (Window Size = 3) reported in chapter 5. Since for window sizes greater than 7, the data become so sparse with many features having a value of 0, all the tables will only show the results for window sizes 1, 3, 5 and 7. The experimental results reported in table 6.2 showed that the FDT performance has improved when combining the global features with the local features generated with a window size of 7. This leads us to hypothesize that the local features extracted using a window size of 7 encode more discriminative information than those encoded using other window sizes. To examine the correctness of our hypothesis, we conducted several experiments on datasets with local features encoded several window sizes. In these experiments we applied four known machine learning tools (C4.5, SVM, RF and FDT) to these datasets. The results of these experiments are reported in table 6.3.

Table 6.3: Classification Results of different machine learning tools when applied to datasets with local features extracted using the sliding window scheme. In the table the values in each of the cells represent Accuracy/Sensitivity/ Specificity

Window Size	SVM	C4.5	RF	FDT
1	89.3/85.5/90.0	81.5/85.4/79.6	86.6/90.6/80.8	89.1/85.2/89.9
3	89.9/92.0/89.5[27]	81.4/85.2/80.7[27]	88.1/91.5/87.5	88.9/87.8/89.1
5	89.5/91.7/89.1	76.6/82.0/75.5	90.1/90.9/90.0	87.7/80.0/89.1
7	89.5/55.2/96.3	70.8/83.5/67.9	90.0/87.4/90.5	86.2/64.2/90.3

The classification results shown in table 6.3 leads to the following interesting observations: First, The local features encoded using the triplet elements scheme encode comparable discriminative information to that of those encoded using other window sizes. Second, the features encoded with a window size of 7 are not more discriminative than those encoded using other windows sizes. Based on the second observation, it is unclear why the performance of FDT was improved when combining the global features with the local feature extracted using a window size of 7. This leads us to hypothesize that FDT was utilizing only the global features. To check the correctness of this hypothesis, we examined the fuzzy rules generated by FDT when trained on datasets with the global features and local features obtained with a window size of 7. This analysis revealed that the FDT algorithm was utilizing only a subset of the global features. In other words, the performance improvement achieved was not due to the more discriminative information encoded in the local feature extracted by a window size of 7, but due to the removal of the classification confusion introduced by the use of the triplet elements. This result is inconsistent with the results reported in the literature which stated that combining the features

extracted using triplet elements with the global features improves the classification performance of pre-microRNAs. To verify our results we applied four common machine learning tools to datasets with only the global features, and compared the results with those obtained when using datasets with both the global and local features. Table 6.4 shows the results of these experiments.

Table 6.4: Classification Results Comparison: Local Features, Global features versus Combined Global Feature and Local Features obtained using the triplet elements

Tool	Local	Global	Global and local
SVM	89.9/92.0/89.5[27]	92.3/92.8/92.6	93.1/92.0/93.3 [27]
C4.5	81.4/85.2/80.7[27]	90.8/86.5/91.6	88.9/84.5/89.7 [27]
RF	88.1/91.5/87.5	90.4/85.5/91.3	90.7/89.5/90.9 [61]
FDT	88.9/87.8/89.1	94.1/91.4/94.6	93.6/82.6/95.6 [61]

The results in table 6.4 show that some of the machine learning tools performs slightly better on the dataset with the local features encoded using the triplet elements than on datasets with global features only, while others performs slightly better on datasets with global features. Generally the results achieved by using the global features are comparable to those achieved by combining the global features with the local features obtained by the triplet encoding scheme.

Further analysis of the fuzzy rules extracted by the FDT algorithm showed that only six global features were used, these are: number of basepairs, GC content, length basepair ratio,

central loop length, free energy per nucleotide, and the number of tails. To test the significance of the features selected by FDT, we applied a number of classification tools to datasets with only these six features. Table 6.5 compares the performance of these tools over the six selected global features, the eleven global features and the combined global features and local features extracted using the triplet elements. Interestingly, the results indicated that majority of the tools showed improved performance over the reduced datasets.

Table 6.5: Comparing Classification Results: The Six global features selected by FDT, global features, and global features combined with local features encoded using the triplet elements. The values in each cell represent accuracy/sensitivity/specificity.

Tool	Reduced Global	Global	Global and local
SVM	93.6/90.2/94.2	92.3/92.8/92.6	93.1/92.0/93.3 [27]
C4.5	89.7/86.3/90.3	90.8/86.5/91.6	88.9/84.5/89.7 [27]
RF	90.8/87.8/91.4	90.4/85.5/91.3	90.7/89.5/90.9 [61]
FDT	94.1/91.4/94.6	94.1/91.4/94.6	93.6/82.6/95.6 [61]

Finally, we performed several experiments during which we applied the FDT algorithm to pre-microRNAs obtained from Rfam12.0. The goal of these experiments was to test whether using only the six selected global features will improve the classification performance of FDT on Rfam 12.0, and to test whether the fuzzy rules generated by FDT can be generalized or not. In the first experiment, we applied the fuzzy rules obtained using dataset 1 from table 6.1 with the reduced set of features to the pre-microRNA sequences obtained from Rfam database release 12.0; this dataset contains 8619 pre-microRNA sequences; all sequences were used including

those with multiple loops. Out of the 8619 sequences the tool was able to predict 7567 sequences. In other words the prediction accuracy and sensitivity are equal to 87.8%. These results may indicate that the fuzzy rules obtained using FDT can be generalized and are not specific to the training and validation datasets used during the experiments or that the training samples can fully represent the pre-mircoRNA samples in Rfam 12.0.

In the second experiment, we combined the pre-microRNA sequences in release 12.0 with the background sequences in table 6.1. The datasets includes only the six global features mentioned previously in this section. We performed a three fold cross validation over the resulting dataset. FDT achieved 93.6% average accuracy, 95.2% average sensitivity and 89.9% average specificity compared to 92.3% average accuracy, 93.9% average sensitivity and 88.9% average specificity in our previous work [18] presented in chapter 5. These results show that by using the reduced set of features the predictive power of the FDT algorithm was improved.

6.4 Conclusions

In this chapter, we introduced an extension of the triplet element encoding scheme by varying the number of secondary structure positions used for encoding the features. We called this encoding scheme as the extended triplet element sliding window encoding scheme. The triplet elements encoding scheme represent a special case of the sliding window encoding scheme with a window size equal to 3. Examining the features generated by using different win-

dow size revealed that these features encode comparable discriminative information to that of the features encoded using the triplet elements encoding scheme.

Analyzing the fuzzy rules generated by FDT revealed that, the FDT algorithm was utilizing only a subset of the global features. Applying popular classification tools to datasets with only this subset of these features showed that the classification performance of these tools has been improved in most cases.

Applying common machine learning classification tools to the datasets with only the global features showed that the classification performance of some of these tools has been improved. This result is inconsistent with results reported in the literature which state that combining the features extracted using the triplet elements (local features) with the global features improves the classification performance. A deeper analysis of the results showed that only the performance of SVM which was used in the literature was improved when combining global and the local feature. Therefore the results reported in the literature can not be generalized.

The FDT algorithm showed some performance improvement on datasets with the global features and the local features extracted using a window size of 7. However, analyzing the decision model revealed that this improvement was due to the removal of the negative effect caused by the use of the triplet elements rather than being due to the discriminative characteristics of the local features extracted using a window size of 7.

Applying the rules obtained by training FDT on a small dataset obtained from an older microRNA database (Rfam 5.0) to a larger dataset of the newer microRNA database (Rfam 12.0) results in 87.8% prediction accuracy. This indicates that rules generated by FDT can be generalized. This conclusion need to be further investigated.

The experimental results showed that the local features obtained using different window sizes contain comparable discriminative information. This may indicate that the reserved patterns in pre-microRNA sequences and their associated secondary structure are of variable length. We will investigate this option in our future research. Furthermore we think including positional data about where such patterns occur in a microRNA sequence will improve the prediction accuracy of pre-microRNA and may help in identifying microRNA targets.

Chapter 7

FUZZY DECISION TREE SOFTWARE TOOL "FDT"

7.1 Introduction

In this chapter we will introduce the fuzzy decision software tool developed during this research work and discuss its' features. The tool was started off an implementation of FID3 authored by ***Zhiheng Huang*** which was posted on the website of University of Edinburgh and no longer available. The tool implements the FID3, improved FID3 and the extended improved FID3 algorithms. The tool also implements four different variation of FID3: The first utilizes information gain, the second utilizes classification ambiguity, the third utilizes gini-index, and the fourth integrates both information gain and classification ambiguity to select the branching feature.

7.2 FDT Features

The FDT classification tool implements FID3, IFID3 and extended IFID3. Two versions of FID3 are supported; the first version uses Information Gain to select the branching feature while the second uses classification ambiguity for the same purpose. The tool also implements a fuzzy version of the Gini-index information measure. This fuzzy version was defined during this work. The tool enables the user to select one of the four options to do classification. The difference

between these four options is the information measure used to select the branching feature. The available options are:

1. Using fuzzy information gain in all fuzzy decision tree non-leaf nodes to select the branching feature.
2. Using classification ambiguity in all decision tree non-leaf nodes to the branching feature.
3. Using a fuzzy version of the gini-index in all fuzzy decision tree non-leaf nodes to select the branching feature.
4. Using classification ambiguity to select the branching feature at the fuzzy decision tree root node, and information gain elsewhere.

The tool also implements the extended IFID3 algorithm which has an additional threshold, this threshold and the original FID3 two thresholds can be tuned by the user to improve the classification results. The tool supports three data fuzzification options. In the first option the ranges of all features are split evenly into an equal number of fuzzy sets. In the second option, the ranges of all features are split into an equal number of fuzzy sets using cluster centers information. And, in the third option, the data is fuzzified using fuzzy membership functions information provided by the user. The tool supports only triangular and trapezoidal fuzzy membership functions.

In addition the tool supports two inference methods. In the first method, during the generated fuzzy-decision-tree conversion into rules, the rule is assigned the name of the class with the greatest membership value. When doing inference, the test assigned the class label of the rule with the maximum firing threshold. In the second method, all class labels with partial membership values are assigned to the rule. During inference, the partial memberships of all class labels of the fired are summed and the test instance is assigned the class label of the class with the greatest membership value. The tool also implements interpolation to allow inference on data instances firing no fuzzy rules.

The software tool allows the user to use the fuzzy rules automatically generated by mining data, or to use a predefined rule set. The advantage of the later is that the user does not need to train the tool every time he needs to classify new data instances. In addition, this makes the tool suitable for distributed environments. For example if we think a bout a distributed intrusion detection system composed of a central utility and distributed network edge agents. Only the rules need to be distributed to these agents. The computational-intensive data mining step to learn these rules can be done in the central utility. This has two advantages, first, detection at the edge nodes is faster, the edge nodes are not involved in mining the data to learn new patterns. Second, since the edge nodes are not involved in the time and power intensive data mining step, these agents do not need to be high performance machines which lower the cost.

Finally the tool utilizes the bootstrapping technique in attempt to build a fuzzy forest. To make inference, all trees are applied to a given test data instance and voting is used to draw the

final decision. This part is still under development and further work and algorithmic details need to be investigated.

7.3 File Format

In this section, we will cover the file formats of all the files that may be needed for running the FDT tool. These files include dataset files, fuzzy parameters file, rules file, and cluster center information file. The dataset file, fuzzy parameters file and rules file use the same format used by the FID3 implementation authored by ***Zhiheng Huang***. All the files involve a header section and a data section.

7.3.1 Dataset File Format

A dataset file has two sections, the first section is the header information and the second section is the data. The header section contains the information about the attributes in the dataset. The first line specifies the number of attributes (features) in the dataset including the target attribute or the class label. Each feature is then described in the one line. The attribute description lines start the with keyword @attribute followed by the attribute name and type. The attribute type could be numerical or category. If the type is category, the attribute name is followed by all possible values taken by the attribute. Only attributes of numerical type can be fuzzified. The data section starts with the keyword @data in one line. Each data instance

is described in one line with the values assigned to different attributes separated by commas. Missing values are represented by the question mark character “?”. Figure 7.1 shows a sample dataset file.

```
5
@attribute A1 numerical
@attribute A2 numerical
@attribute A3 numerical
@attribute A4 numerical
@attribute class category c1 c2 c3
@data
269,269,981,500,c1
47,981,500,500,c1
118,500,500,881,c1
500,1300,1200,950,c3
400,1250,1250,980,c3
47,881,952,118,c2
47,730,118,118,c2
```

Figure 7.1: Sample Dataset File

7.3.2 Fuzzy Parameters File Format

The fuzzy parameters file stores the information needed to fuzzify data. It stores the number of fuzzy sets associated with each feature and the associated membership functions. This file can be provided by the user or generated automatically by the tool. The first two lines are header lines. The first line stores the string “fuzzify parameters: “. The second line stores information about the number of fuzzy sets associated with each feature and whether a given

feature should be fuzzified or not, the values are separated by commas. To identify a categorical attribute, the number of fuzzy sets which in this case the number of all possible values taken by this attribute is marked with "*". Following this header section is the fuzzy sets definition section. This section is composed of subsections that are separated by blank lines. Each subsection stores the fuzzy membership function information associated with one of the features. Each line in a subsection defines a fuzzy set. Figure 7.2 shows a sample fuzzy parameters file with the associated fuzzy membership functions.

```
fuzzify parameters:
4,4,4,4,3*
23.5,23.5,131.9375,188.5625
131.9375,188.5625,245.1875,301.8125
245.1875,301.8125,358.4375,415.0625
358.4375,415.0625,750.0,750.0

134.5,134.5,462.3125,591.1875
462.3125,591.1875,720.0625,848.9375
720.0625,848.9375,977.8125,1106.6875
977.8125,1106.6875,1950.0,1950.0

59.0,59.0,330.25,471.75
330.25,471.75,613.25,754.75
613.25,754.75,896.25,1037.75
896.25,1037.75,1875.0,1875.0

59.0,59.0,279.625,387.375
279.625,387.375,495.125,602.875
495.125,602.875,710.625,818.375
710.625,818.375,1470.0,1470.0
```

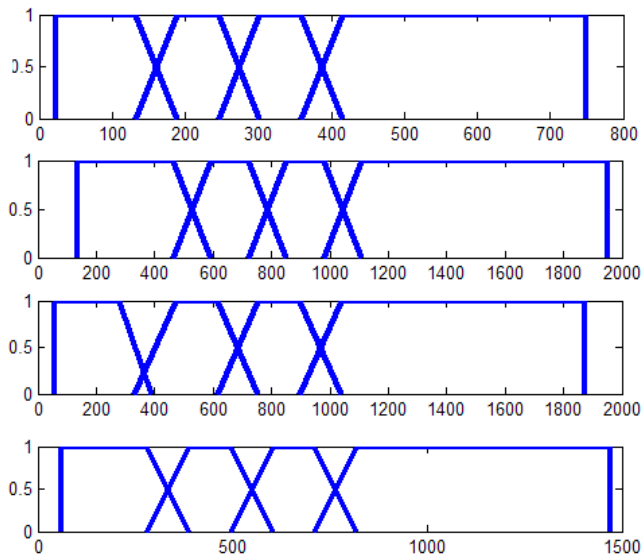


Figure 7.2: Sample Fuzzy Parameters File with the associated fuzzy membership functions

In the figure the line 4,4,4,4,3* indicates that the range all attributed was divided into four fuzzy sets except the last one. The last one is a categorical attribute that can take only three possible values. Fuzzy descriptions are required only for numerical attributes.

7.3.3 Cluster-Centers' File Format

The clusters centers file is an optional file. It allows the user to use the cluster centers coordinates stored in a file to build a fuzzy parameters file. Currently the tool has no clustering module. Any clustering algorithm can be used. The first line in the file stores the number of attributes in the dataset that were clustered. The second line stores the number of clusters. The rest of the lines store the clusters centers information. Each line represents the coordinates of a cluster center. Figure 7.3 shows a sample cluster file.

```
5
4
14.459,19.203,94.342,649.47,0.097093
19.105,21.51,126.32,1139.8,0.10188
23.42,24.304,155.91,1738.2,0.098176
11.335,18.317,72.779,399.31,0.094882
```

Figure 7.3: Sample clusters centers file

7.3.4 Fuzzy Rules File Format

The fuzzy rules file stores the fuzzy rules generated from the constructed tree. The rules' file consists of two parts. The first is the header part and the second is the fuzzy rules. The first line in the header part stores the number of attribute including the target attribute (Class label).

Each attribute is the described in one line. The attribute description line starts with the keyword @attribute followed by the attribute name and the set of linguistic terms assigned to each of the fuzzy sets. The rules part start with the keyword @rule followed by the set of rules. Each rule is described in one line. The rule lists all the linguistic terms of all attributes followed by a class label. Figure 7.4 shows a sample rules file. In the rules file the terms Fterm0, Fterm1, …etc. can be substituted by meaningful linguistic terms such as cold, hot, etc. …

```
5
@attribute A1 {FTerm0 FTerm1 FTerm2 FTerm3 }
@attribute A2 {FTerm0 FTerm1 FTerm2 FTerm3 }
@attribute A3 {FTerm0 FTerm1 FTerm2 FTerm3 }
@attribute A4 {FTerm0 FTerm1 FTerm2 FTerm3 }
@attribute class {c1 c2 c3 }

@rule
null, null, null, FTerm0, c2
null, null, null, FTerm1, c1
null, null, null, FTerm2, c1
null, null, null, FTerm3, c3
```

Figure 7.4: Sample Rules File

7.3.5 Example

In this section we will go over an example. The datasets in this example have five features including the class label; there are three classes in the dataset. The training and testing datasets are shown in figures 7.5 and 7.6 respectively.

```
5
@attribute A1 numerical
@attribute A2 numerical
@attribute A3 numerical
@attribute A4 numerical
@attribute class category c1 c2 c3
@data
269,269,981,500,c1
47,981,500,500,c1
118,500,500,881,c1
500,1300,1200,950,c3
400,1250,1250,980,c3
47,881,952,118,c2
47,730,118,118,c2
```

Figure 7.5: Example Training Dataset

```
5
@attribute A1 numerical
@attribute A2 numerical
@attribute A3 numerical
@attribute A4 numerical
@attribute class category C1 C2 C3
@data
47,500,730,500,C1
118,269,118,118,C2
450,1270,1220,960,C3
```

Figure 7.6: Example Testing Dataset

Each parameter range was divided evenly into three fuzzy sets. The fuzzy parameters file used during this example is shown in figure 7.7. Figure 7.8 shows the generated fuzzy decision tree and figure 7.9 shows the generated rules file.

```
fuzzify parameters:
3,3,3,3,3*
23.5,23.5,160.25,235.75
160.25,235.75,311.25,386.75
311.25,386.75,750.0,750.0

134.5,134.5,526.7500000000001,698.5833333333334
526.7500000000001,698.5833333333334,870.4166666666667,1042.25
870.4166666666667,1042.25,1950.0,1950.0

59.0,59.0,401.0,589.6666666666666
401.0,589.6666666666666,778.3333333333333,967.0
778.3333333333333,967.0,1875.0,1875.0

59.0,59.0,333.5,477.16666666666663
333.5,477.16666666666663,620.8333333333333,764.5
620.8333333333333,764.5,1470.0,1470.0
```

Figure 7.7: Fuzzy Parameters File

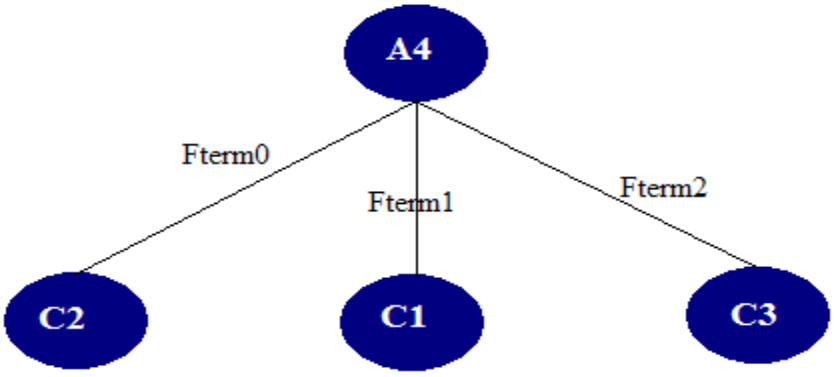


Figure 7.8: Constructed Fuzzy Decision Tree

```
5
@attribute A1 {FTerm0 FTerm1 FTerm2 }
@attribute A2 {FTerm0 FTerm1 FTerm2 }
@attribute A3 {FTerm0 FTerm1 FTerm2 }
@attribute A4 {FTerm0 FTerm1 FTerm2 }
@attribute class {c1 c2 c3 }

@rule
null, null, null, FTerm0, c2
null, null, null, FTerm1, c1
null, null, null, FTerm2, c3
```

Figure 7.9: Generated Fuzzy Rules Files

In the rules file shown in the above figure, each rule is labeled only with one label and does not adhere to fuzzy rule syntax introduced in chapter 4. Figure 7.10 shows the rules file using the syntax introduced in chapter 4. Notice, in this rule file, the rule is not labeled using a given class label but rather the relative of memberships of each class is recorded. This syntax can be used to assign a truth value to each fuzzy rule.

```
5
@attribute A1 {FTerm0 FTerm1 FTerm2 }
@attribute A2 {FTerm0 FTerm1 FTerm2 }
@attribute A3 {FTerm0 FTerm1 FTerm2 }
@attribute A4 {FTerm0 FTerm1 FTerm2 }
@attribute class {c1 c2 c3 }

@rule
null, null, null, FTerm0, 0.0, 1.0, 0.0
null, null, null, FTerm1, 1.0, 0.0, 0.0
null, null, null, FTerm2, 0.3333333333333333, 0.0, 0.6666666666666666
```

Figure 7.10: Rules Files Using the Rule Syntax Presented in Chapter 4

7.4 Summary and Future Work

In this chapter we presented the fuzzy decision tree classification tool that was developed during this research work. The tool implements a fuzzy information gain FID3, a classification ambiguity FID3, a fuzzy gini-index FID3, the IFID3 algorithm presented in chapter 3 and the extended IFID3 presented in chapter 4. The tool supports the automatic generation of fuzzy membership functions by dividing the range of all attributes evenly into an equal number of fuzzy sets or by using clusters centers information. For future work the tool need to support and include the following:

1. Implement a data clustering module.
2. Mine available data in order to automatically generate fuzzy membership functions.
3. Tune the resulting fuzzy membership function.
4. Implement additional techniques for rule set reduction.
5. Support dividing the range of each feature into a number of fuzzy sets independent of the number of fuzzy sets assigned to other feature in the dataset. In other words, the number of fuzzy sets of all features need not be equal.
6. Design and Implement a fuzzy forest algorithm.

Chapter 8

SUMMARY, CONCLUSIONS AND FUTURE WORK

8.1 Summary and Conclusions

Current fuzzy decision tree induction algorithms are slow, generate a huge number of rules and require intensive human involvement in determining the fuzzy parameters used to fuzzify the data at hand. As a result further improvement of these algorithms is needed. A successful approach to improving fuzzy decision tree induction process should improve the accuracy, reduce the running-time, automate the data fuzzification process and generate a smaller number of fuzzy rules.

In this work, we improved the accuracy and execution time of the fuzzy decision induction algorithm by integrating fuzzy information gain and classification ambiguity to select the branching feature. We also reduced significantly the number of fuzzy rules generated by the fuzzy decision tree induction algorithm by introducing a new threshold on the membership value of a data object to propagate down the decision tree from a parent node to any of its' child node during the tree construction step.

Although a number of fuzzy decision tree algorithms exist, our results indicate that further improvements of these algorithms are possible. In the literature, most of the work done concentrates on applying fuzzy decision trees to solving certain problems. The researchers in

their work concentrated on optimizing the fuzzy modeling of the target problem using input from experts rather than concentrating on improving the fuzzy decision tree algorithm. Furthermore, fuzzy decision trees have been applied to datasets with a small number of features, and therefore, the tree computational time was not an issue. In addition, provided test cases were either hypothetical or a targeting the problem at hand, and therefore, the reported results can not be generalized. As a result, further work on improving the fuzzy decision induction algorithms is of critical importance.

Fuzzy Decision Trees are a promising machine-learning technique; our experimental results indicate that this technique can outperform other machine learning approaches. In fact, we have some evidence that fuzzy decision trees can deal efficiently with highly unbalanced dataset. The pre-microRNA dataset represent an example of an unbalanced dataset, which fuzzy decision trees were able to solve efficiently.

A common key problem with fuzzy decision tree induction algorithms is that they require or rely on fuzzy models determined by the experts in the field. However, some systems are very complex and it is sometimes hard for a human expert to build an optimal fuzzy model of the problem at hand. Therefore, it is very important to come up with systematic methods that can learn and extract fuzzy models from the data.

Reducing the number of fuzzy rules is very critical to the interpretability of the resulting model of the problem and to the applicability of this model to real time applications. More re-

search is required to develop method to reduce the number of rules generated by the current fuzzy decision trees induction algorithms.

Decision trees are generally considered as weak learners. A weak learner builds a strong classifier; however, this classifier can not be generalized. In this work we have some evidence that the rules obtained by the fuzzy decision trees algorithms can be generalized. This may indicate that by using fuzzy set theory we may be able to transfer decision trees from being weak learners to being strong learners. This observation need to be further investigated to prove its correctness.

7.2 Future Work

This research work presented two directions of research. The first concentrates on improving the fuzzy decision tree algorithms, while the second concentrates on improving the microRNA prediction and the prediction of microRNA targets.

Given the availability of huge amounts of data, fuzzy decision trees represent a direct approach to extract fuzzy rules from collected data, especially for cases where the systems being modeled are complex or there is no input from an expert in the field of the application. Despite the fact that several fuzzy decision trees algorithms exist, theses algorithms have the following limitations:

- **Performance:** Existing methods have been limited by computational performance. While our approach is about two hundred times faster than applications like FID3.4, it still can be improved. Performance improvement will be implemented by profiling the existing code to identify bottlenecks, using the java threads package for those parts of the process that can be implemented in parallel and algorithmic improvements to reduce the dependency on the number of features in the data. We have preliminary data showing that feature reduction algorithms are effective with fuzzy decision trees.
- **Accuracy:** The accuracy of the FDT critically depends on the implementation of the fuzzy sets. Our current program is competitive in accuracy with the state of the art learners like SVM and random forests. Effective fuzzy set definition algorithms should improve our performance. Existing fuzzy decision tools require the user to provide the fuzzy membership functions for each of the features related to the decision problem. Providing such information is a not a trivial task. In some cases, this information may be provided by experts in the application fields. However, due to the complexity of some systems it is hard even for the experts to decide on the optimal fuzzy membership functions. Furthermore, most of the time, the user analyzing the data has no expertise in the target domain. Therefore, it is important to derive methods that can mine the available data and generate fuzzy membership functions. We propose to examine the use of clustering and genetic algorithms to define fuzzy sets, and to examine the use of bootstrapping algorithms to minimize the dependence of the decision tree on the specific data samples.

- **Rule Set Reduction, Feature Space Reduction:** Existing fuzzy decision tools generate a huge number of fuzzy rules. Rule set reduction improves the system interpretability, efficiency and the applicability of the resulting fuzzy systems to real time application; as a result, it is necessary to derive methods to reduce the resulting rule set base. Extracting a small number of rules based on a critical set of features has the operational advantage of being experimentally falsifiable so that a domain specialist can use this approach to identify new avenues for experimental science. We propose to apply ideas used in fuzzy logic to the generated rule base and to merge of adjacent rules that lead to the same classification/prediction decision. Furthermore, we believe that an optimal definition of the fuzzy membership function will lead to a further reduction in the rule base size.

A fuzzy forest may boost the classification performance of fuzzy decision trees. Bootstrapping the datasets and/or the features may be used for this purpose. In addition to bootstrapping, splitting on multiple features may be utilized to build a fuzzy forest. Our hypothesis is that the use of a forest will improve the accuracy; however, we expect that improvement to be less than the improvement achieved in the crisp case. This is because we believe that the randomness used in generating the trees in the Random Forest algorithm compensates for the fuzziness in the data.

The automatic generation of fuzzy membership functions by mining available data is of critical importance especially when systems in hand are complex or no such information is avail-

able from an expert in the field. A data-driven approach for the automatic generation of fuzzy membership functions is of great importance. A possible approach is to cluster unlabeled data. Each resulting cluster will be represented by a fuzzy set. To identify the optimal number of clusters in the data, we will use one of the clustering-quality measures available in the literature. To determine the boundaries of each fuzzy set we will utilize information derived from the labeled data. One of the drawbacks of using this approach is that it will result in an equal number of fuzzy sets for each of the features in the data. One of the following approaches can be applied to overcome this problem:

1. Clustering the values of each feature individually.
2. Merging adjacent fuzzy sets if the datasets belonging to them belong to the same class.
3. Using genetic algorithms to fine-tune the resulting membership functions.

Another drawback is that in order to determine the optimal number of clusters we have to run the clustering algorithm several times and each time measure the clustering quality. Such approach will be computationally intensive for large datasets. To solve this problem, one of the following approaches can be used:

1. Using a feature selection algorithm to reduce the number of features in a dataset.
2. Using a cluster counting algorithm.

Another possible approach to learn fuzzy membership functions from data is to use genetic algorithms. Several papers in the fuzzy logic field utilize genetic algorithms to fine tune membership functions and reduce the number of rules. However, the main assumptions were that initial fuzzy membership functions and fuzzy rules are provided by an expert in the target field. We believe that genetic algorithms can be used effectively to learn membership functions.

MicroRNAs are now being used as therapies to cure some diseases; therefore identifying microRNAs and their targets is of great importance. Since our results may suggest that pre-microRNAs may contain reserved patterns of variable sizes. Investigating these patterns and finding them may significantly improve the prediction accuracy of microRNAs and their targets.

REFERENCES

[1] Umano M., Okamoto H., Hatono I., Tamura H., Kawachi F., Umedzu S., Kinoshita J., "Fuzzy Decision Trees by Fuzzy ID3 Algorithm and Its Application to Diagnosis Systems," in Proc. 3rd IEEE Conf. on Fuzzy Systems, Orlando, Vol. 3, pages 2113-2118, 1994.

[2] Yuan Y., Shaw M. J., "Induction of Fuzzy Decision Trees," Fuzzy Sets and Systems, Vol. 69, Issue 2, pages 125-139, 1995.

[3] Janikow C. Z., "Fuzzy Decision Trees: Issues and Methods," IEEE Trans. on Man, Systems and Cybernetics, Vol. 28, Issue 1, pages 1-14, 1998.

[4] Lee K., Lee K., Lee J., Lee-Kwang H., "A Fuzzy Decision Tree Induction Method for Fuzzy Data," Proc. IEEE Conf. on Fuzzy Systems, FUZZ-IEEE 99, Seoul, Vol. 1, pages 16-25, 1999.

[5] Janikow C.Z., "Exemplar Learning in Fuzzy Decision Trees," Proc. of 5th IEEE Int, Conf on Fuzzy Systems, New Orleans, Vol 2, pages 1500-1505, 1996.

[6] Janikow C. Z. and Fajfer M., "Fuzzy Partitioning with FID3.1," Proc. of the 18th Int. Conf. of North American Fuzzy Information Society, New York, pages 467-471, 1999.

[7] Janikow C. Z., "Fuzzy Decision Forest," Proc. of 22nd Int. Conf. of the North American Fuzzy Information Processing Society, Chicago, pages 480-483, 2003.

[8] Quinlan J. R., “Decision Trees and Decisionmaking,” IEEE Trans. on Systems, Man and Cybernetics, Vol. 20, Issue 2, pages 339-346, 1990.

[9] Janikow C. Z., “FID4.1: an Overview,” IEEE Annual Meeting of the Fuzzy Information Processing, NAFIPS ’04, Vol. 2, pages 877-881, 2004.

[10] Lee J., Sun J. and Yang L., “A Fuzzy Matching Method of Fuzzy Decision Trees,” Int. Conf. on Machine Learning and Cybernetics, Vol. 3, Issue 2, pages 1569-1573, 2003.

[11] Huang Z. and Shen Q., “Fuzzy Interpolative Reasoning via Scale and Move Transformations,” IEEE Trans. on Fuzzy Systems, Vol. 14, Issue 2, pages 340-359, 2006.

[12] Quinlan J. R., “Discovering Rules by Induction from Large Collections of Examples”, in D. Michie (ed.) : Expert Systems in Micro Electronics Age, Edinburgh University Press, 1979.

[13] Quinlan J. R., “ Induction of Decision Trees”, Machine Learning, Vol. 1, pages 81-106, 1986

[14] Quinlan, J. R., “C4.5: Programs for Machine Learning”. Morgan Kaufmann, San Francisco, CA (1993).

[15] Rafael Alcala, Jesus Alcala-Fdez, Maria Jose Gacto and Francisco Herrera, “Improving Fuzzy Logic Controllers Obtained by Experts: A Case Study in HVAC Systems” Springer Netherlands, Applied Intelligence, 2007.

[16] Jesus Alcala-Fdez, Rafael Alcala, Maria Jose Gacto and Francisco Herrera, “Learning the Membership Function Contexts for Mining Fuzzy Association Rules by Using Genetic Algorithms”, Fuzzy Sets and Systems, Vol. 160, Issue 7, pages 905-921, 2009.

[17] Maria Jose Gacto, Rafael Alcala and Francisco Herrera, “Adaptation and Application of Multi-Objective Evolutionary Algorithms for Rule reduction and Parameter Tuning of Fuzzy Rule-Based Systems”, Soft Computing – A Fusion Foundations, Methodologies and Applications, Vol. 13, Issue 5, 2008.

[18] Raouf Ketata, Hatem Bellaaj, Mohamed Chtourou, and Mohamed Ben Amer, “Adjustment of Membership Functions, Generation and Reduction of Fuzzy Rule Base from Numerical Data”, Malaysian Journal of Computer Science, Vol. 20, Issue 2, pages 147-169, 2007

[19] Kemal Ciliz M., “Rule Base Reduction for Knowledge-Based Fuzzy Controllers with Application to a Vacuum Cleaner”, Expert Systems with Applications, Vol. 28, Issue 1, pages 175-184, 2004.

[20] Dan Simon, “Design and Rule Base Reduction of a Fuzzy Filter for the Estimation of Motor Currents”, International Journal of Approximate Reasoning, Vol. 25, Issue 2, 2000.

[21] Sheng-Ming Wu and Hung-Yuan Chung, “Fuzzy Rules Reduction for Discrete-Time T-S Fuzzy Systems Based on Regional Piecewise Approach”, IEEE SICE Conference Proceedings, pages 574-579, 2007.

[22] Xue, C., Li, F., He, T., Liu, G., Li, Y., Zhang, X., “Classification of Real and Pseudo Micro-RNA Precursors Using Local Structure-Sequence and Support Vector Machine”. BMC Bioinformatics 6(1) (2005) 310

[23] Sewer, A., Paul, N., Landfraf, P., Aravin, A., Pfeffer, S., Brownstein, M., Tuschl, T., van Nimwegan, E., Zavolan, M., “Identification of Clustered MicroRNAs Using an Ab Initio Prediction Method”. BMC Bioinformatics 6(1) (2005) 267

[24] Yoon, S., De Micheli, G., “Computational Identification of MicroRNAs and Their Tragets”. In: Birth Defects Research 2006, vol. 78, pages 118—128, (2006)

[25] Xu, J., Li, F., Sun, Q., “Identification of MicroRNA Precursors with Support Vector Machine and String Kernel”. In: Genomics, Proteomics & Bioinformatics, vol 6, issue 2, pages 121-128, (2008)

[26] Jaing, P., Wu, H., Wang, W., Ma, W., Sun, X., Lu, M., “MiPred: Classification of Real and Pseudo MicroRNA Using Random Forest Prediction Model with Combined Features”. Nucleic Acids Res. 35: W339-344 (2007)

[27] Zheng, Y., Hsu, W., Li Lee, M., Soon Wong, L., “Exploring Essential Attributes For Detecting MicroRNA Precursors From Background Sequences”. In: 32nd International Conference on Very Large Databases Workshop on Data Mining in Bioinformatics, Seoul, Korea, (2006), June, 2009

[28] Olaru C., Wehenkel L.: A Complete Fuzzy Decision Tree Technique. Fuzzy set and systems, pages 221-254, 2003.

[29] Chen Y., Wang T., Wang B., “A Survey Of Fuzzy Decision Tree Classifier”, Fuzzy Information and Engineering, pages 149-159, June (2009)

[30] Vapnik V. and Cortes C., “Support vector networks”, Machine Learning vol 20, no 3, pages 273-293, 1995.

[31] Breiman L., “Random Forests”, Machine Learning. Vol. 45, No.1, pages 5-32, 2001.

[32] Drucker H., “Effect of Pruning and Early Stopping on Performance of a Boosting Ensemble”, Computational Statistics and Data Analysis, Vol. 34. Issue 4, pages 393-406, 2002.

[33] White A., Liu W. Z., “Technical Note: Bias in Information-Based Measures in Decision Tree Induction”, Machine Learning, Vol. 15, pages 321-329, 1994.

[34] Smithson M., Verkuilen J., “Fuzzy Set Theory: Applications in The Social Sciences”, Sage Publication Ltd., ISBN 0-7619-2986-X, 2006.

[35] Breiman L., Friedman J. H., Olshen J. A., & Stone, C. J, “Classification and regression trees. Belmont”, CA: Wadsworth International Group. 1984

[36] Chandra B., Varghese P. P., “Fuzzifying Gini Index Based Decision Trees”, Expert Systems with Applications, Vol. 36, Issue 4, pages 8549-8559, 2009.

[37] Zadeh L. A., “Fuzzy Sets”, Information and Control, Vol. 8, pages 338-353, 1965.

[38] Zadeh L. A., “Fuzzy Sets as A Bases for a Theory of Possibility”, Fuzzy Sets and Systems, Vol. 1, pages 3-28, 1978.

[39] Abu-halaweh N., Harrison R. W., “Practical Fuzzy Decision Trees”, Proceeding of IEEE Symposium on Computational Intelligence and Data Mining (CIDM09), 2009, p.p. 211-216.

[40] Papagelis A., Kalles D., “GA Tree: Genetically Evolved Decision Trees”, Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pages 203-206, 2000.

[41] Huang Z., Fuzzy EBY Software, http://www.eecs.berkeley.edu/~zhiheng/ .

[42] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/ .

[43] Rost, B., "Rising accuracy of protein secondary structure prediction" In: Chasman D.,editor., Protein structure determination, analysis, and modeling for drug discovery. New York: Dekker, pages 207-249, 2003.

[44] Street W. N., Wolberg W. H., Mangasarian O. L.. "Nuclear Feature Extraction for Breast Tumor Diagnosis". International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

[45] Mangasarian O. L., Street W. N., Wolberg W. H., "Breast Cancer Diagnosis and Prognosis via Linear Programming", Operations Research, 43(4), pages 570-577, 1995.

[46] Sigillito V. G., Wing S. P., Hutton L. V., Baker K. B., "Classification of Radar Returns from the Ionosphere Using Neural Networks. Johns Hopkins APL Technical Digest, 10, pages 262-266, 1989.

[47] Abu-halaweh N., Harrison R. W., "Rule Set Reduction in Fuzzy Decision Trees", Proceedings of NAFIPS, 2009, p.p. 1-4.

[48] Hammond S., "MicroRNAs as Oncogenes", Current Opinion in Genetics and Development, Vol. 16, Issue 1, pages 4-9, 2006.

[49] Papagiannakopoulos T., Kosik K., "MicroRNAs: Regulators of Oncogenesis and Stemness", BMC Medicine, Vol. 6, 2008.

[50] Scaria V., Jadhav V., "MicroRNAs in Viral Oncogensis", Retrovirology, Vol. 4, 2007.

[51] Thum T., Catalucci D., Bauersachs J., "MicroRNAs: Novel Regulators in Cardiac Development and Disease", Cardiovascular Research, Vol. 79. Issue 4, pages 562-570, 2008.

[52] Mishra P. K., Tyagi N., Kundu S., Tyagi S. C., "MicroRNAs Are Involved in Homocysteine-Induced Cardiac Remodeling", Cell Biochemistry and Biophysics, Vol. 55, Issue 3, pages 153-162, 2009.

[53] Rooij E. V., Olson E. N., "MicroRNAs: Powerful New Regulators of Heart Disease and Provocative Therapeutic Targets", Science in Medicine, Vol. 117, Issue 9, pages 2369-2376, 2007.

[54] Wang, X., Zhang, J., Li, F., Gu, J., He, T., Zhang, X., Li, Y., "MicroRNA Identification Based on Sequence and Structure Alignment", Bioinformatics, vol. 21, pages 3610--3614, (2005)

[55] Ambion, MiRNA Research Guide, http://www.ambion.com/miRNA

[56] JonesRhoades, M., Bartel, D., "Computational Identification of Plant MicroRNAs and Their Targets Including a Stress-Induced MiRNA", Mol Cell 2004, 14 (6), pages 787—799, (2004).

[57] Berezikov, E., Guryev, V., Van de Belt, J., Weinholds, E., Plasterk, RH., Cuppen, E., "Phylogenetic Shadowing and Computational Identification of Human MicroRNA Genes", Cell 2005, vol. 120, pages 21—24, (2005).

[58] Esquela A., Slack F. J., "Oncomirs- MicroRNAs with A Role in Cancer", Nature Reviews Cancer, Vol. 6, pages 259-269, 2006.

[59] Lai, E., Tomancak, P., Williams, R., Rubin, G, ”Computational Identification of Drosophila MicroRNA Genes”, Genome Biol, 2003, 4(7):R42, (2003)

[60] Hofacker I.L., “Vienna RNA Secondary Structure Server”, Nucleic Acids, 31, 2003, p.p. 3429-3431.

[61] Abu-halaweh N., Harrison R. W., “Prediction and Classification of Real and Pseudo MicroRNA Precursors via Data Fuzzification and Fuzzy Decision Trees”, Proceedings of ISBRA, pages 323-334, 2009.

[62] Zheng, Y., Hsu, W., Li Lee, M., Limsoon, W., “Exploring Essential Attributes for Detecting MicroRNA Precursors from Background Sequences”, http://www.comp.nus.edu.sg/~wongls/projects/miRNA/suppl-info/vldb2006.htm

[63] S. Griffiths-Jones, HK. Saini, S. Van Dongan, “miRBase: Tools for MicroRNA genomics”, NAR, 2008 36 (Database Issue): D154-D158.

[64] S. Griffiths-Jones, RJ. Grocock, S. Van Dongan, A. Bateman, AJ. Enright, “miRBase: microRNA Sequences, Targets and Gene Nomenclature”, NAR, 2006 34 (Database Issue): D140-D144.

[65] S. Griffiths-Jones, “The MicroRNA Registry”, NAR 2004 32 (Database Issue): D109-D111.

[66] V. Ambros, B. Bartel, DP. Bartel, JC. Carrington, X. Chen, G. Dreyfuss, S. Griffiths-Jones, M. Marshall, G. Ruvkun, T. Tuschl, “A Uniform System for MicroRNA Annotation”, RNA, 2003 9(3): 277-279 (2003).

[67] Aha D., Kibler D., Albert M., “Instance-Based Learning Algorithm”, Machine Learning, Vol. 6, pages 37-66, 1991.

[68] Cohen W. W., In Prieditis A., Russell S., “Fast Effective Rule Induction”, Proceedings of the 12th International Conference On Machine Learning, Tahoe City, CA, pages 115-123, Morgan Kaufmann, 1995.

[69] Abu-halaweh Na'el, Harrison Robert W., "Identifying the Essential Features for the Classification of Real and Pseudo MicroRNAs Precursors Using Fuzzy Decision Trees", Accepted to appear in the Proceedings of the IEEE Conference in Bioinformatics and Computational Biology (CIBCB 2010). Montreal, Canada