UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

Fuzzy Methods for Machine Learning. A big data perspective.

Barsacchi Marco

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Fuzzy Methods for Machine Learning. A big data perspective.

Barsacchi Marco

Advisor:

Ing. Alessio Bechini

Advisor:

Prof. Beatrice Lazzerini

Advisor:

Prof. Gigliola Vaglini

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Giovanna Castellano, *Università degli Studi di Bari*
Prof. Alberto Fernandez Hilario, *University of Granada*

XXXI ciclo — October 2019

It is the mark of an instructed mind to rest satisfied with that degree of precision which the nature of the subject admits, and not to seek exactness where only an approximation of the truth is possible.

Aristotle, 384-322 BC

ACKNOWLEDGEMENTS

This thesis would not have been possible without the contributions of many people. First, I want to thank my advisor Alessio Bechini who made this work possible. I appreciate his contributions of time, ideas, to make my Ph.D. experience productive and stimulating. I would like to thank my supporting advisors Beatrice Lazzerini and Gigliola Vaglini for their great support and invaluable advice.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Giovanni Stea and Prof. Enrico Vicario, for their insightful comments and encouragement.

I would like to express my sincere gratitude to Prof. Francesco Marcelloni and Prof. Pietro Ducange who have been supportive of my career goals and whose valuable suggestions have contributed greatly to the improvement of the thesis.

My sincere thanks also goes to Prof. Pietro Lió and to the Computational Biology Group who provided me with the opportunity to join their group as intern, and who hosted me in Cambridge during my visiting period.

Furthermore, I would to thank the reviewers, Prof. Giovanna Castellano and Prof. Alberto Fernandez Hilario, for their detailed comments and suggestions for improving the manuscript.

Last but not the least, I would like to thank my family: my parents, for supporting me throughout my life, my brother and my girlfriend. This dissertation would not have been possible without their warm love, continued patience, and endless support.

SUMMARY

More than fifty years after its introduction, fuzzy sets theory is still thriving and continues to play a relevant role in a wide number of scientific applications. Nevertheless, while the enrichments that fuzzy logic and set theory can provide are manifold, the recognition of fuzzy set and logic inside the machine learning community remains rather moderate. In this thesis, we present several approaches aimed at improving machine learning techniques using tools borrowed from fuzzy set theory and logic. Particularly, we try to focus more on the machine learning perspective, thus inviting machine learning researcher to appreciate the modelling strengths of fuzzy set theory.

We begin presenting **FDT-Boost**, a boosting approach shaped according to the SAMME-Adaboost scheme, which leverages fuzzy binary decision trees as base classifiers; then, we explore a distributed fuzzy random forest **DFRF**, that leverages the Apache Spark framework, to generate an efficient and effective classifier for big data. We also propose a novel approach for generating, out of big data, a set of fuzzy rule-based classifiers characterised by different optimal trade-offs between accuracy and interpretability. The approach, dubbed **DPAES-FDT-GL**, extends a state-of-the-art distributed multi-objective evolutionary learning scheme, implemented in the Apache Spark environment. Lastly, we focus on an application, showing how fuzzy systems could be employed in helping medical decision; we propose a novel pipeline to support tumour type classification and rule extraction based on somatic CNV data. The pipeline outputs an interpretable Fuzzy Rule-Based Classifier (FRBC)

Much work remains to be done, and fuzzy set theory has still a big role to play in machine learning.

LIST OF ABBREVIATIONS

C

CNV	Copy Number Variation.

D

DB	Data Base.
DT	Decision Tree.

F

FDT	Fuzzy Decision Tree.
FL	Fuzzy Logic.
FMDT	Fuzzy Multi-way Decision Tree.
FRBC	Fuzzy Rule-Based Classifier.
FRBS	Fuzzy Rule-Based System.
FRF	Fuzzy Random Forest.

K

KB	Knowledge Base.

M

MOEA	Multi-Objective Evolutionary Algorithm.
MOEL	Multi-Objective Evolutionary Learning.

R

RB	Rule Base.
RF	Random Forest.

T

TCGA	The Cancer Genome Atlas.

NOTATION

Symbols

$\{x_1, ..., x_n\}$	The set of n elements.
$O, \Omega, \Theta, o, \omega$	asymptotic notation.
$T(x, y)$	A fuzzy T-norm.
$X, \mathcal{X}$	instances domain.
$Y, \mathcal{Y}$	labels domain.
$[x]^{\mathrm{T}}$	transpose operator.
$[a, b)$	The real interval including a but excluding b.
$[a, b]$	The real interval including a and b.
$\perp(x, y)$	A fuzzy T-conorm.
$\lceil x \rceil$	round to the upper integer of x.
$\lfloor x \rfloor$	round to the lower integer of x.
$\mathbb{R}$	the set of real numbers.
$\mathbb{R}^+$	the set of non-negatives real numbers.
$\mathbb{R}^d$	the set of d-dimensional real vectors.
$\mathbb{R}^{mxn}$	the set of $m \times n$ matrices.
$\mu_A(\cdot)$	membership function for a set A.
$\mathbf{x}$	vectors are in bold.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER

ONE

INTRODUCTION

1. The world is everything that is the case

Tractatus Logico-Philosophicus —
Ludwig Wittgenstein

1.1 Introduction

More than 50 years have passed since the introduction of the concept of fuzziness by Zadeh (Zadeh, 1965); during these years, the theory of fuzzy sets and logic gradually yet relentlessly crept into many scientific fields, such as computer science, control engineering, decision theory, pattern recognition, and robotics, just to name a few. Among them, the field of artificial intelligence (AI) is certainly one of the most active and thriving.

In fact, the idea of building artificial beings and thinking machines is an old one; the mythology of ancient Greece is sprinkled with tales of golden robots and artificial beings. The quest for artificial intelligence evolved unceasingly through the centuries, up to the birth of modern AI. Modern AI emerged in 1956 at Dartmouth College, and grew ever since, ushering in a new era of big data and machine learning.

Nowadays, it is machine learning that takes the lion's share of artificial intelligence (Figure 1.1). Indeed, machine learning's surge in popularity followed the beginning of the era of *big data*. Being no longer possible to design a knowledge acquisition system by hand, automated systems for knowledge acquisition have come to the fore. Models, automatically constructed from the data, might be used to predict future observations or may provide insights into the inner structure of the data. The birth of *fuzzy machine learning* has been accompanied by the same sudden shift from *knowledge-driven* to *data-driven* fuzzy modelling, where the model is constructed (more properly *induced*) automatically from the training data.

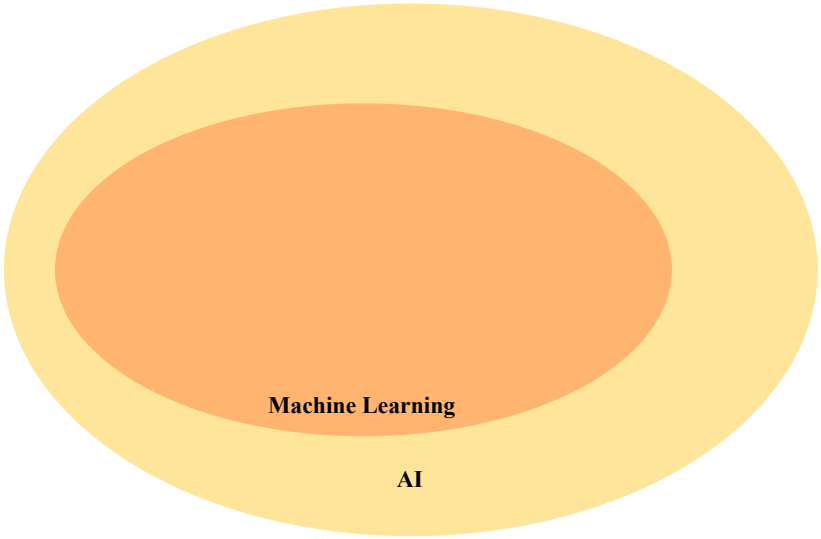


Figure 1.1: *Artificial intelligence studies machines that define a course of action reacting to the environment, aiming to a particular goal. Machine learning devices automatically learn to complete a task from the data, without the need to be programmed to do so.*

According to Mitchell (Mitchell, 1997), a machine learning system may be defined in the following way: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*". Here, we particularly focus on methods, or algorithms, that perform automatically pattern detection from data, and use the discovered patterns to predict future data or perform some kind of decision-making from that data (Murphy, 2012). In this thesis, we are primarily interested in predictive or supervised machine learning, i.e. in the task of learning (or *inducing*) a function f from a set of training data (a set of pairs of the form $(\mathbf{x}_i, y_i)$, being $\mathbf{x}_i$ the i-th input and y_i the i-th output) to a set of outputs. If the response variable is categorical, the problem is known as classification or pattern recognition; the *classifier* is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, being $\mathcal{X}$ the instance space and $\mathcal{Y} = \{1, ..., k\}$ the target space, with k number of classes. The estimated $\hat{f}$ can then be used to make predictions. Different classification approaches make different assumptions, and, as such, focus on diverse classes of functions from which the approximated $\hat{f}$ is learnt.

The contribution of fuzzy set theory and logic to machine learning can be manifold; if used in the data pre-processing stage, fuzzy sets might be employed to model vague data, and fuzzy summaries of subsets of data might also be created (Hüllermeier, 2011). Furthermore, fuzzy methods can be also employed in the analysis phase; in this case, approaches are borrowed from fuzzy set theory and logic and used to analyse data.

The reasons that drive the employment of fuzzy set theory and logic methods in machine learning are multifarious (Hüllermeier, 2011); the paramount factor behind the widespread diffusion of fuzzy set theory is the concept of interpretability. Even though the definition of interpretability is still a debated topic, as fuzzy sets bridge the gap between the

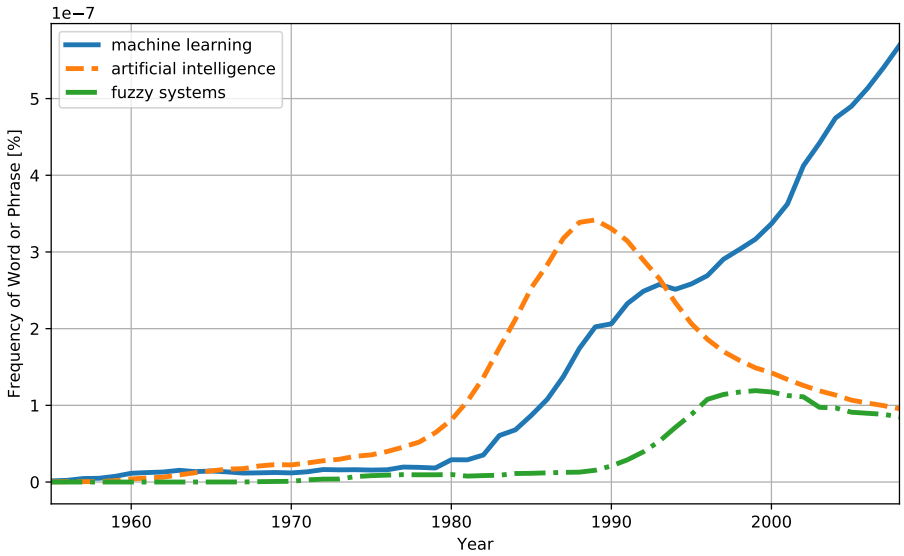


Figure 1.2: *Growth of interest in machine learning, as measured by the frequency of the phrases "artificial intelligence" and "machine learning", according to Google Books. The frequency for the phrase "fuzzy system" is reported as well.*

realm of numbers and the realm of concepts, they provide a way to represent knowledge in a linguistic and thus comprehensible way. For example, linguistic fuzzy rule-based system are generally regarded as highly interpretable, due to their being based on linguistic rules, in which the antecedent and the consequent are made up of linguistic variables comprised of linguistic terms and the associated fuzzy sets defining their meanings. A general taxonomy of interpretability distinguishes complexity from semantics and rule base versus data base (or fuzzy partition) (Gacto et al., 2011).

Furthermore, fuzzy set theory provides a natural framework for dealing with uncertainty and uncertain phenomena; as machine learning is inextricably bound to uncertainty, fuzzy approaches represent a valuable way for dealing with such a problem. Machine learning has to deal with data that is noisy, often incomplete or imprecise; in this scenario fuzzy set theory can complement probability theory and provides a natural approach for exploiting the uncertain characteristics of the input data.

It has often been argued that fuzzy methods are inherently more robust with respect to crisp methods. In this context, robustness is often used with a slightly different meaning with respect to the standard definition of *robustness* in computer science, i.e. "the ability of a computer system to cope with errors during execution, and cope with erroneous input." A robust machine learning method ideally shows stable performances if some (small) noise is added to the data. It is commonly held that fuzzy models are more robust with respect to variations in the boundary points, as the soft boundaries employed in fuzzy systems prevent abrupt changes with small variations in boundary points. In the case of decision trees, for example, the lack of smoothness of the decision surface is mitigated when using

fuzzy decision trees instead.

Other reasons often cited to support the usage of fuzzy set theory in machine learning are the capability to incorporate background knowledge, granularity (Pedrycz et al., 2008), i.e. the capability of dealing with information granules at different level of abstraction, and graduality, the inherent capability of fuzzy system to deal with gradual concepts. Fuzzy concepts, defined as fuzzy (instead of crisp) subset of the universe U, are naturally endowed with the notion of graduality (Dubois and Prade, 2012); furthermore, if defined in terms of fuzzy predicates, the qualifying properties can benefit from a gradual definition too.

In this work, we focus on devising novel machine learning approaches that exploit fuzzy set and logic to either improve the classification performances or solve existing problems in a novel and more efficient way. Two specific families of classifier are of particular interest in this thesis: *fuzzy rule-based systems* (FRBSs) and *fuzzy decision trees* (FDTs).

1.2 Motivation

As machine learning has become increasingly more widespread and pervasive, the number of application in which it is replacing (or at least helping) human decision making is now growing at a staggering rate. Nevertheless, more often than not, we still struggle to explain why a particular machine learning model behaved in the way it did. Even if not all ML systems need interpretability, there are classes of problems and tasks for which it is not enough to get the prediction (the what); rather the model should also be able to explain how it came to the prediction (the why). Those, amongst others, are the paramount reasons that drive the demand for interpretability and explanation.

Since its inception, fuzzy logic has been often praised as a valuable modelling tool (Gacto et al., 2011) that can produce highly interpretable models. Indeed, particularly with linguistic models, fuzzy ML has shown a remarkable aptness in producing ML systems that are not only accurate but also highly interpretable (Cordón, 2011).

The goal of this thesis is to use tools, borrowed from fuzzy logic and set theory, to improve machine learning systems, particularly with respect to the interpretability of the learnt model; furthermore, as we have already entered the era of big data (Anuradha et al., 2015; Segatori et al., 2017b), the need for systems that are able to cope with huge amount of data, yet learning an interpretable model, is even more pressing. Finally, we aim to show how real applications can benefit from the model produced according to the guidelines described so far. We believe that the room for improvement in this area is big enough to justify a research project in this direction.

1.3 Contribution

In this thesis, diverse aspects of machine learning and fuzzy expert systems are explored and improved with respect to the state-of-the-art. The core focus of this work is on the machine learning perspective, with the aim of inviting machine learning researchers to

appreciate the modelling strengths offered by fuzzy set theory and logic. Among the core topics faced up in this work are ensembles of fuzzy systems and fuzzy rule-based classifiers for big data.

First, we concentrate on ensembles of fuzzy systems, developing a boosted ensemble of fuzzy decision trees – dubbed FDT-Boost – that improves state-of-the-art performances (Chapter 3). The system proves to be effective and more robust when noise is injected into the training set; moreover, several tools are used to provide valuable insights into the learning phase.

Second, we focus on fuzzy systems for big data; we begin by i) proposing a novel Distributed Fuzzy Random Forest for Big Data, which leverages the Apache Spark Framework in order to provide an effective classifier (Chapter 4). Then, ii) we significantly improve an evolutionary fuzzy rule-based system for big-data, using a strategy for learning the most suitable number of fuzzy sets (*granularity learning*) for each linguistic variable, concurrently to the learning of the rule base and the parameters of the fuzzy sets (Chapter 5). The proposed system, named DPAES-FDT-GL, is able to generate more interpretable models than the state-of-the-art while preserving comparable accuracies.

Lastly, in order to further emphasise the applicability of fuzzy machine learning systems, we devise a use case, presenting a pipeline for tumour classification from somatic CNV data. The proposed pipeline first performs data extraction and preparation, then implements a classification system built upon a fuzzy rule-based classifier (Chapter 6). We show that the system is able to provide highly accurate yet interpretable results.

1.4 Thesis Outline

Chapter 2: Background In this chapter, we review the preliminaries of fuzzy sets and logic used in the remaining of the thesis; our aim is that of offering the machine learning practitioner the necessary fuzzy logic background needed to understand the rest of the work.

Chapter 3: On Boosting Approaches for Fuzzy Decision Trees This chapter describes FDT-Boost, a boosting approach shaped according to the SAMME-AdaBoost scheme, which leverages fuzzy binary decision trees as base classifiers. Furthermore, new tools for a comprehensive analysis of the learning phase are provided.

Chapter 4: Implicitly Distributed Fuzzy Random Forests In this Chapter, a novel implementation of a Distributed Fuzzy Random Forest (DFRF) induction algorithm is presented and discussed. The proposed approach, although shaped on the MapReduce programming model, takes advantage of the implicit distribution of the computation provided by the Apache Spark framework. A thorough evaluation follows.

Chapter 5: Multi-Objective Evolutionary Fuzzy Classifiers for Big Data This chapter

proposes a novel approach for generating, out of big data, a set of fuzzy rule-based classifiers characterised by different optimal trade-offs between accuracy and interpretability. The approach extends a state-of-the-art distributed multi-objective evolutionary learning scheme, implemented under the Apache Spark environment.

Chapter 6: Fuzzy Rule Based Classifiers: an application to CNV-base Tumour Classification using Fuzzy Rule Based Classifiers This chapter proposes a novel pipeline to support tumour type classification and rule extraction based on somatic CNV data. The pipeline outputs an interpretable Fuzzy Rule Based Classifier (FRBC), on which inference can be made. The results show the potential application of the approach: The method is able to classify between three kidney tumour types, with an accuracy of $\sim$ 93%, using a compact set of $\sim$ 50 interpretable rules.

Chapter 7: Discussion This chapter provides a comprehensive review of the results achieved in this work, as well as some future directions.

CHAPTER

TWO

BACKGROUND

> All traditional logic habitually assumes that precise symbols are being employed. It is, therefore, not applicable to this terrestrial life but only to an imagined celestial existence.
>
> — Bertrand Russel

In this chapter, the necessary background about fuzzy logic and fuzzy set theory is introduced. As the topic of fuzzy logic is broad and lively, we do not attempt to provide a comprehensive review of the subject. We will instead restrict this introduction to the basic concepts needed to properly understand the subsequent chapters.

The basic notions of fuzzy sets and logic, fuzzy operators and relations are provided in Section 2.1. Section 2.2 bridges the gap towards approximate reasoning, while Section 2.3 introduces the notion of fuzzy rule-based systems and fuzzy rule-based classifiers. Section 2.4 briefly introduced evolutionary fuzzy system. Then, Section 2.5 provides a brief discussion of probability and fuzzyness.

2.1 Basics of Fuzzy Logic

2.1.1 The quest for fuzzyness

Most of our traditional tools for formal modelling, reasoning, and computing are crisp, deterministic, and precise in nature (Zimmermann, 2010). Nonetheless, we constantly interact with a reality that is neither crisp nor certain. We routinely process vague and imprecise linguistic terms, such as *hot*, *tall*, and *around midnight*. The ability to proficiently deal with and reason from, imprecise data is indeed a peculiarly human one. The theory of fuzzy sets was first introduced by Zadeh (Zadeh, 1965, 1975a,b,c), who aimed at generalising the classical notion of sets and proposition, in order to include fuzzyness; its seminal

paper "Fuzzy sets" (Zadeh, 1965) opens with the (by now famous) quote, clearly realising the need to account for fuzzyness:

> "More often than not, the classes of objects encountered in the real physical world do not have precisely defined criteria or membership"

He first formalised the concept of vagueness in the scientific community into the form of fuzzy logic. The concept of fuzzy sets emerged when it was realised that it may not be possible to model ill-defined systems with precise mathematical assumptions of the classical methods, such as probability theory. As we strove to describe the reality in a crisp and deterministic fashion, we realised that a complete description of a system is, more often than not, beyond human possibility.

Fuzzy set theory provides the tools for coping with uncertainty, i.e. a strict mathematical framework for dealing with vague conceptual phenomena. It provides a way for embedding approximated reasoning, or imprecise human reasoning, into problems that were heretofore intractable.

2.1.2 Classical Mathematics and Logic

Classical logic defines a *proposition* p as a linguistic, or declarative statement, that can be either true or false. In other words, the set of truth values on which classical logic is based is $\{0, 1\}$. A unary operation $\neg$ (NOT) and two binary operations $\wedge$ (AND), $\vee$ (OR) are defined on the propositions. Compound propositions can be constructed from atomic propositional variables $p, q, \ldots$ using the basic operators; e.g $(p \wedge q), (p \vee q), \neg q$ (Fodor and Rudas, 2015).

Let X, *universe of discourse*, be the universe of all available information about a given problem. Let A be a generic set on this universe, i.e. a collection of object $x \in X$, that might be finite, countable or over countable. Classical set theory satisfies the Aristotelian "Law of Excluded Middle" that is, for a given crisp set A defined over X, each element $x \in X$ either belongs to A or not belongs to A.

For classical sets, three basic operations are defined: i) the union between two sets, $A \cup B$, is defined as the set of objects that belong to the set A, to the set B or both; ii) the intersection $A \cap B$ is defined as the set of elements that belong to both A and B. Finally, iii) the complement of a set $\bar{A}$ is defined as the collection of all elements in the universe that do not belong to that set. We define the characteristic function $\mathcal{X}_A : X \rightarrow 0, 1$ of the set A, as:

$$\mathcal{X}_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \tag{2.1}$$

The characteristic function discriminates between members and non members of a set. Using the characteristic function we are able to describe the set operations in terms of

logical connectives (Ross, 2011):

$$\begin{aligned} \mathcal{X}_{A\cup B}(x) &= \mathcal{X}_A(x) \vee \mathcal{X}_B(x) \\ \mathcal{X}_{A\cap B}(x) &= \mathcal{X}_A(x) \wedge \mathcal{X}_B(x) \\ \mathcal{X}_{\bar{A}}(x) &= \neg\mathcal{X}_A(x) \end{aligned} \tag{2.2}$$

2.1.3 Fuzzy Sets and Membership Functions

A fuzzy set is a generalisation of a classical set, which defines a generalised characteristic function, called *membership function*, allowed to assume various degrees of memberships (Zimmermann, 1996). A membership function μ_A is a mapping from the universe of discourse X to the unit interval, i.e., $\mu_A : X \rightarrow [0, 1]$. As stated, membership functions are direct generalisations of characteristic functions which allow degrees of membership between zero and one. In a logical setting, the degree of membership $\mu_A(x)$ can also be interpreted as the truth value of the statement "x is an element of A". Figure 2.1 portrays the example of a generic fuzzy set A and shows a comparison with a crisp set A'.

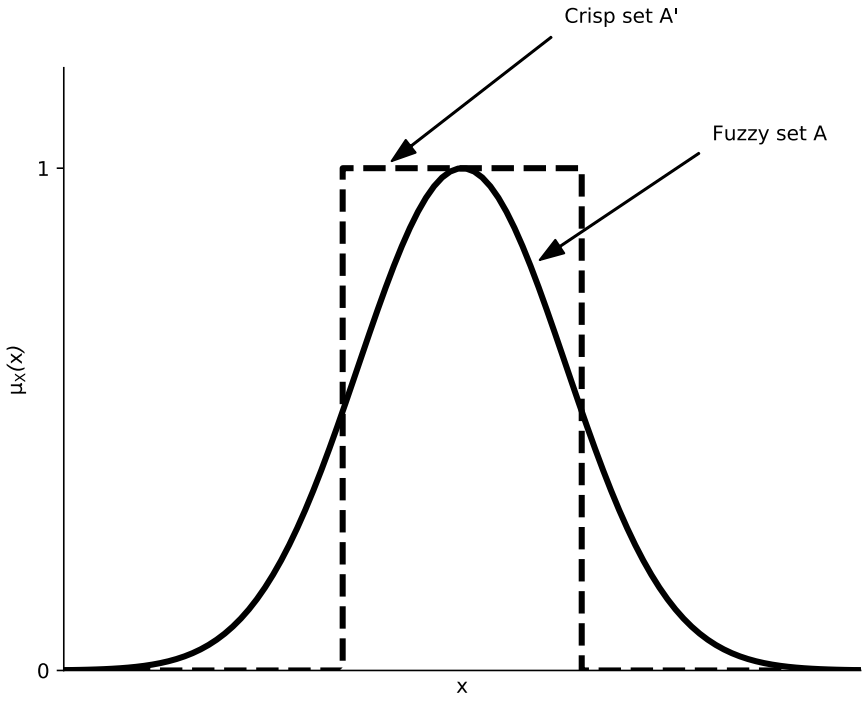


Figure 2.1: *A fuzzy set A and a crisp set A′.*

A fuzzy set can thus be represented as a collection of ordered pairs of elements of X and their membership grades. A fuzzy set defined on a discrete universe can be described using the following notation:

$$A = \left\{ \frac{\mu_A(x_1)}{x1} + \frac{\mu_A(x_2)}{x2} + \ldots \right\} = \left\{ \sum_i \frac{\mu_A(x_i)}{x_i} \right\} \tag{2.3}$$

or, if the universe of the discourse is a continuum as:

$$A = \left\{ \int_x \frac{\mu_A(x)}{x} \right\} \tag{2.4}$$

It usually avoided listing all the elements for which the membership degree is 0. It is also worth noticing that in both notations, the horizontal bar is not a quotient but rather a delimiter; furthermore, the summation, integral, and + symbols represent aggregation or collections operators.

In the following we assume that all fuzzy set are normalised, i.e. $\sup_x \mu_A(x) = 1$. We define the cardinality of a fuzzy set A as :

$$|A| = \sum_{x \in X} \mu_A(x) \tag{2.5}$$

We refer to the *support* of a fuzzy set A as the crisp set of elements $S(A) = \{x | \mu_A(x) \geq 0\}$. The *$\alpha$-level set* A_α is defined as the (crisp) set of elements of A whose membership degree is greater than α:

$$A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\} \tag{2.6}$$

2.1.4 Set-Theoretic operators

The fuzzy set operations generalise the classic set operations, like intersection, union and complement. Nevertheless, unlike classical set theory (and classical logical), several different definitions are possible (Ross, 2011).

The basic set-theoretic operations are illustrated in Figure 2.2. The membership function of the intersection $C = A \cap B$ between A and B is defined for each element x as:

$$\mu_C(x) = \min \{\mu_A(x), \mu_B(x)\} \tag{2.7}$$

The membership function of the union $D = A \cup B$ of A and B is defined for each element x as:

$$\mu_D(x) = \max \{\mu_A(x), \mu_B(x)\} \tag{2.8}$$

The complement E of a normalised fuzzy set A is defined in terms of its membership function:

$$\mu_E(x) = 1 - \mu_A(x) \tag{2.9}$$

As stated, the basic form of the three fuzzy standard operations is not the only possible formulation. In fact, a broad class of functions can be used to generalise each of the standard operators. Functions that qualify as fuzzy intersection, $T : [0, 1]^2 \to [0, 1]$, are referred to as *triangular norm* or *t-norm*, while we refer to functions that qualify as fuzzy union $\perp : [0, 1]^2 \to [0, 1]$ as *triangular conorm* or *t-conorm* or even S-norm. The precise mathematical definition of t-norms and t-conorms is beyond the scope of this thesis; the interested reader is referred to (Zimmermann, 2010). A notable property of operations on fuzzy sets is that the "axiom of the excluded middle" does not hold, i.e. $A \cap A^c \neq \emptyset$ and $A \cup A^c \neq X$.

A non-comprehensive list of operators is reported in Table 2.1. A even wider class of parametric functions exists; please refer to (Fodor János, 2000) for further details.

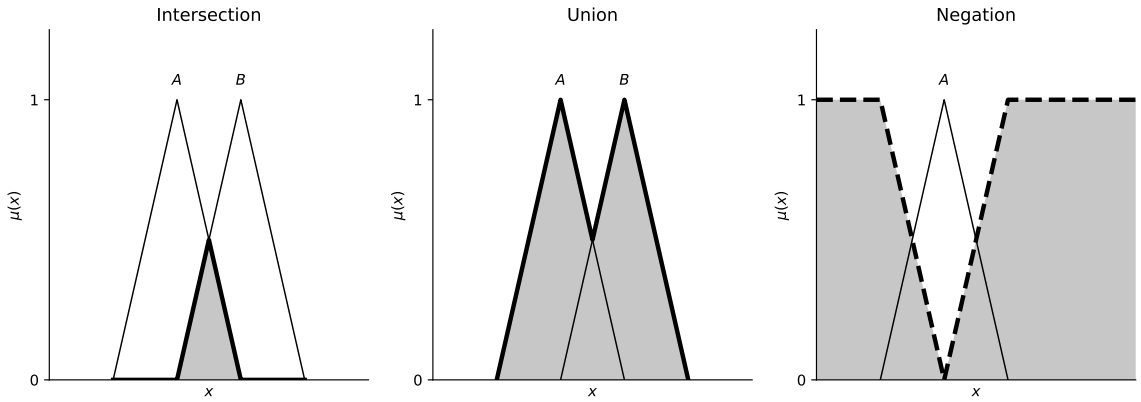


Figure 2.2: *Basic fuzzy set operators:* intersection *between two fuzzy sets A and B (left panel),* union *between two fuzzy sets A and B (middle panel) and* negation *of a fuzzy set A (right panel).*

Table 2.1: *Alternative fuzzy Sets operators*

Name	**T-norm** $T(x,y)$	**T-conorm** $\perp(x,y)$	**Negation** $N(x)$
Gödel	$\min(x,y)$	$\max(x,y)$	$1-x$
Product	$x \cdot y$	$x+y-x\cdot y$	$1-x$
Łukasiewicz	$\max(x+y-1,0)$	$\min(x+y,1)$	$1-x$
Drastic	$\begin{cases} x & \text{if } y=1 \\ y & \text{if } x=1 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} x & \text{if } y=0 \\ y & \text{if } x=0 \\ 1 & \text{otherwise} \end{cases}$	$1-x$
Nilpotent	$\begin{cases} \min(x,y) & \text{if } x+y \geq 1 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} \max(x,y) & \text{if } x+y \leq 1 \\ 1 & \text{otherwise} \end{cases}$	$1-x$
Hamacher	$\frac{x\cdot y}{x+y-x\cdot y}$	$\frac{x+y-2\cdot x\cdot y}{1-x\cdot y}$	$1-x$
Einstein	$\frac{x\cdot y}{2-[x+y-x\cdot y]}$	$\frac{x+y}{1+x\cdot y}$	$1-x$

2.1.5 Fuzzy Relations

The concept of relation plays a vital role in the theory of sets and its applications (Zadeh, 1965); furthermore, it is centrally related to the concept of graphs (Zimmermann, 2010). As such, a generalisation of this concept to the fuzzy realm is of paramount importance.

The definition of *fuzzy relation* strictly follows that of crisp relation: given two sets X and Y, a relation R is defined as a mapping from the Cartesian product $X \times Y$ to $[0,1]$; in other words, a relation is a subset of $X \times Y$. A crisp relation $R(x,y)$ measures whether two

elements $x \in X$ and $y \in Y$ are related; the relationship can either be a *complete relationship* or a *no relationship*. Likewise, a fuzzy relation measures the strength of the connection between the two elements; nevertheless, it differs from a crisp relation in that the degree is measured on the interval $[0, 1]$. A fuzzy relation $R(x, y)$ over $X \times Y$ can be thus described in terms of its membership function:

$$R(x, y) = \{((x, y), \mu_R(x, y))|(x, y) \in X \times Y\} \tag{2.10}$$

There is obviously nothing special about a binary relation; in fact, this definition can be extended to the Cartesian product of more than two sets $X_1 \times X_2 \times ... \times X_n$, leading to the definition of $n-$ary fuzzy relations.

As well as on the crisp equivalent, several operations can be defined on fuzzy relations: T-intersection, S-union, transposition, inversion and complement are amongst the most common ones. Nevertheless, an operation on relations in which we are particularly interested – mostly in virtue of the role it plays in allowing approximate reasoning – is that of *fuzzy composition*: let R be a fuzzy relation on $X \times Y$ and S a fuzzy relation on $Y \times Z$. We define the T-composition between R and S as:

$$R(x, y) \circ_T S(y, z) = \sup_{y \in Y} T(R(x, y), S(y, z)) \ \ \forall (x, z) \in X \times Z \tag{2.11}$$

being $T(\cdot, \cdot)$ a t-norm. If the Zadeh (Zadeh, 1965) definition of t-norm is used, we obtain the widely know max-min composition:

$$R(x, y) \circ S(y, z) = \max_{y \in Y} \min(\mu_R(x, y), \mu_S(y, z)) \ \ \forall (x, z) \in X \times Z \tag{2.12}$$

2.2 Towards approximate reasoning

> "In retreating from precision in the face of overpowering complexity, it is natural to explore the use of what might be called linguistic variables, that is, variables whose values are not numbers but words or sentences in a natural or artificial language. The motivation for the use of words or sentences rather than numbers is that linguistic characterizations are, in general, less specific than numerical ones"
> **Zadeh**

2.2.1 Linguistic Variables

The concept of linguistic variable lies at the very heart of fuzzy approximate reasoning. The notion of linguistic variable attempts to harness the powerful and flexible natural language reasoning of which humans are gifted into a mathematical framework. In other words, a linguistic variable is a quantitative method for dealing with qualitative information.

Firstly introduced by Zadeh (Zadeh, 1975b) under the name of *variable of higher order*, a linguistic variable is defined as quintuple[1] $(x, T(x), U, G, \tilde{M})$; here, x is the name of the variable and $T(x)$ represents the set of (*atomic*) terms of x. For the linguistic variable represented in Figure 2.3, $T(x) = \{\text{Low, Medium-Low, Medium, Medium-High, High}\}$. U is the universe of discourse, e.g. $[l_f, u_f]$ for the example in Figure 2.3 and G is a syntactic rule (usually a grammar) for generating the name, X, of values of x; basically $G(x)$ is a rule for generating the terms of a variable x. $\tilde{M}$ is a *semantic rule* that ties together each term X with its *meaning*, $\tilde{M}$, a fuzzy subset of U.

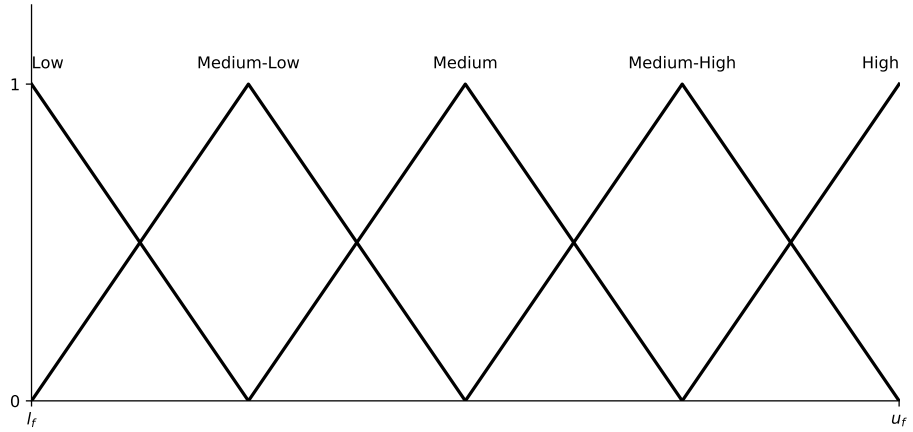


Figure 2.3: *Generic linguistic variables with 5 linguistic terms, defined on a continuous universe of discourse with boundaries l_f and u_f.*

2.2.2 The fuzzy implication

The implication operator $p \implies q$ relates together two proposition p and q, called *hypothesis* (or *antecedent*) and *conclusion* (or *consequent*), respectively. The implication $p \implies q$ is false if and only if p is true and q is false.

The fuzzy implication generalises the classical implication $p \implies q$; the most common form of fuzzy implication is, as first defined by Zadeh(Zadeh, 1965), the following:

$$I(x, y) = \max(1 - x, \min(x, y)) \tag{2.13}$$

As was the case for t-norms and t-conorms, a formal definition of fuzzy implication is given in terms of a function $I : [0, 1]^2 \to [0, 1]$ that needs to satisfy a set of conditions (Baczyński Micheal, 2008). More than one function that satisfies the constraints might be defined; several common implication operators are listed in Table 2.2.

[1]The characterisation of linguistic variable we provide here is the one given in (Zimmermann, 2010).

Table 2.2: *Fuzzy Implication Operators*

Name	Fuzzy Implication $I(x, y)$
Zadeh	$\max(1 - x, \min(x, y))$
Lukasiewicz	$\min(1, 1 - x + y)$
Mamdani	$\min(x, y)$
Gödel	$\begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases}$

2.2.3 Fuzzy Inference

A fuzzy inference system deals with approximate reasoning, i.e. it allows reasoning about imprecise propositions (Zadeh, 1975c). The modus ponens $(A \wedge (A \implies B)) \implies B$, one of the staples of classical logic, allows inferring the truth of a proposition B given the truth of another proposition A and an implication $A \implies B$ [2].

Indeed, we would like to be able to perform approximated reasoning in the following sense: consider the implication $A \implies B$; furthermore, let us know that A', some approximation of A, is true. Can we then conclude that B is approximately true?

Two steps are required in order to provide the desired generalisation. The first necessary step is to define the *generalised modus ponens* (Luis, 2015):

$$\begin{array}{ll} \text{Implication} & A \implies B \\ \text{Premise} & A' \\ \hline \text{Conclusion} & B' \end{array} \tag{2.14}$$

The second step complements the generalised modus ponens in order to allow to reason from unequal premises. Zadeh proposed the *compositional rule of inference* to bridge the gap from inference to approximate inference (Zadeh, 1975a).

The first term in the generalised modus ponens is indeed a fuzzy implication, i.e. a fuzzy relation R defined on $U \times V$:

$$\mu_R(x, y) = I(\mu_A(x), \mu_B(y)) \forall x \in U, y \in V \tag{2.15}$$

The second term in the generalised modus ponens, is a unary fuzzy relation A', defined on U. We can thus apply the compositional rule for the fuzzy relations:

$$\mu_{B'}(y) = A'(x) \circ_T R(x, y) = \sup_{x \in U} T(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \tag{2.16}$$

[2]Given the substantial equivalence between the membership degree of an element x to a set A, $\mu_A(x)$, and the degree of truth of the proposition "x IS A", we will, in the following, refer to fuzzy sets and propositions without distinction.

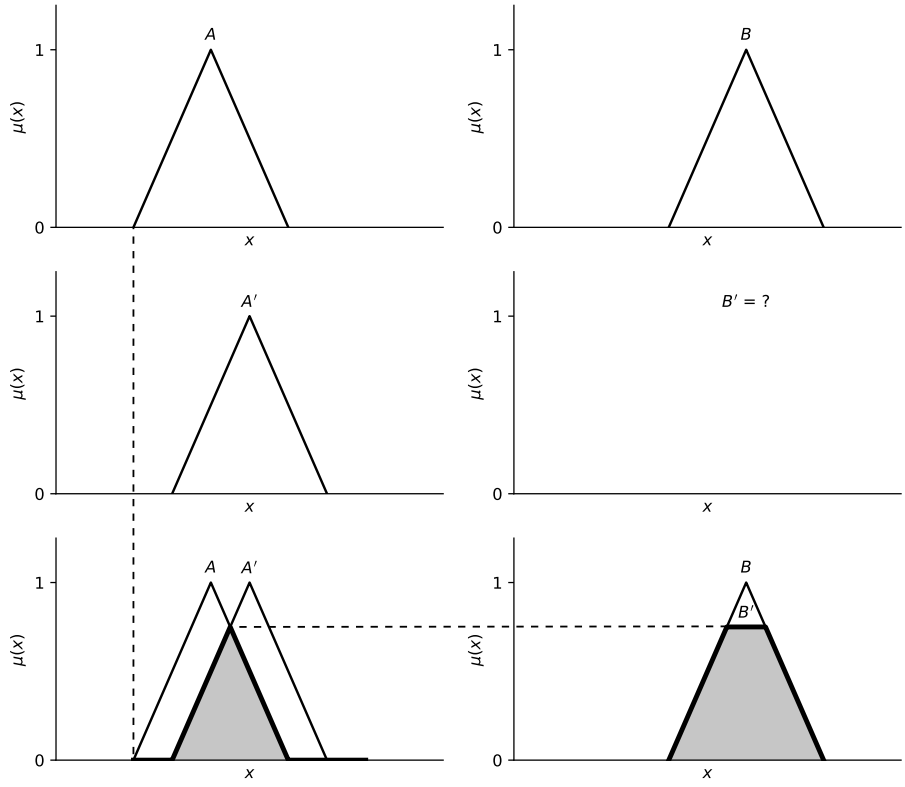


Figure 2.4: *Example of fuzzy inference using the compositional rule of inference with* max-min *inference. We consider the implication* $A \implies B$*, where A and B are shown in the upper left and right panel respectively. The premise* A' *is shown in the middle left panel. Can we infer* B'*? (central right panel). The application of the max-min compositional inference is shown in the lower panels.*

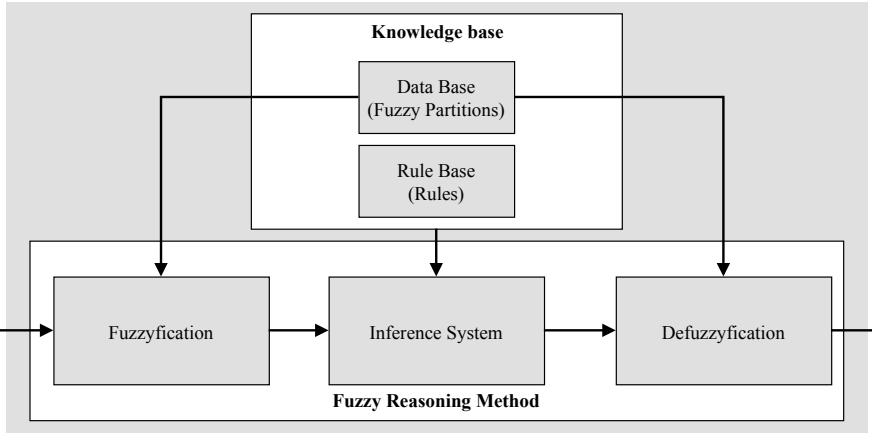


Figure 2.5: *Architecture of a generic fuzzy rule-based system.*

An example of fuzzy inference, is reported in Figure 2.4.

Approximate reasoning is at the core of several fuzzy machine learning approaches such as Fuzzy Rule-Based Systems (FRBSs) and Fuzzy Decision Trees (FDTs) which are the subjects of the next chapters.

2.3 Fuzzy Rule-Based Systems

In this section, an introduction to (fuzzy) rule-based systems (FRBSs) is provided. A rule-based system, also known as *expert system*, is a simple knowledge-based system, which codes knowledge about a particular problem under the form of a set of IF-THEN rules (Grosan and Abraham, 2011). As both the knowledge representation and the reasoning method rely on bivalent logic, rule-based system are inherently inadequate to deal with uncertain or imprecise information. FRBSs generalise rule-based systems, employing linguistic variables and fuzzy sets to provide a more powerful knowledge representation, and fuzzy logic inference to reason from uncertainty.

A generic FRBS, as first introduced by Mamdani (Mamdani and Assilian, 1975), is made up of a knowledge base, where both the rules (rule base) and the fuzzy partitions (data base) are stored, and a fuzzy inference system, implementing the fuzzy reasoning process (Cordón, 2011). The architecture of a generic FRBS is depicted in Figure 2.5

2.3.1 Knowledge Base

The knowledge base (KB) encodes the problem-specific knowledge under the form of a set of linguistic rules (rule base, or RB) and a set of fuzzy partitions (data base, or DB) that describe the semantics of the Fuzzy subsets associated to the linguistic labels in the if-part of the rules (Cordón et al., 1999).

The RB is made up of a set of linguistic rules, joined by the *also* operator (Luis, 2015). Several types of rules have been defined; Mamdani and Takagi-Sugeno-Kang are amongst

the most common ones. Furthermore, different subtypes of Mamdani and Takagi-Sugeno-Kang rules have been proposed in the literature. As a thorough description is beyond the scope of this chapter, the interested reader is referred elsewhere (Luis, 2015; Cordón et al., 1999; Cordón, 2011).

A generic (Mamdani type) k-th rule in the RB has the following form:

$$\begin{aligned} R_k : \textbf{If } X_1 \textbf{ is } A_{1,k_{m,1}} \textbf{ and } \ldots \textbf{ and} \\ X_F \textbf{ is } A_{F,k_{m,F}} \textbf{ then } Y \textbf{ is } B_{j_m} \end{aligned} \tag{2.17}$$

where X_i and Y are the system linguistic input and output variables, respectively, and A_i, $k_{m,1}$ and B_{j_m} are the linguistic labels associated with fuzzy sets specifying their meaning. In a fuzzy rule-based classifier (FRBC), the fuzzy sets associated with the output variable are singleton fuzzy sets $C_j \in \{C_1, \ldots, C_M\}$ representing the (discrete) set of possible class labels.

The DB contains the membership functions of the fuzzy partition associated to each linguistic variable (See Figure 2.3 for an example of a strong fuzzy partition with 5 linguistic terms for a generic linguistic variable).

2.3.2 Fuzzy Reasoning Method

The flow of information through an FRBS follows the scheme reported in Figure 2.5; an input (usually crisp), is presented to the system; the data is fuzzified and the information is then processed by the inference system; finally, the output is defuzzified and a crisp response is generated by the system.

The fuzzification step is paramount as it allows the inference engine to process the input information using approximated inference; a mapping between crisp input values and fuzzy sets defined over the universe of the input is used in this process. The singleton fuzzification is commonly employed (Luis, 2015), i.e. given an input x_0 the corresponding input fuzzy set if defined as:

$$\mu_{A'}(x) = \begin{cases} 1, & x = x_0 \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

The standard fuzzy inference scheme uses the generalised modus ponens described in Section 2.2:

$$\begin{array}{lll} \text{Implication} & \text{IF } X \text{ is } A \textit{ THEN} & Y \text{ is } B \\ \text{Premise} & X \text{ is } A' & \\ \text{Conclusion} & & Y \text{ is } B' \end{array} \tag{2.19}$$

Then, recalling the compositional rule of inference, the membership of a given output set B' is written down as:

$$\mu_{B'}(y) = A'(x) \circ_T R(x, y) = \sup_{x \in U} T(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \tag{2.20}$$

If the input set is a singleton, being $\mu_{A'}(x) = 0 \; \forall x \neq x_0$, we then have:

$$\mu_{B'}(y) = I(\mu_A(x_0), \mu_B(y))) \tag{2.21}$$

To extend the inference to a rule with multiple variables and conditions in the IF part, such as that in Equation 2.17, we then proceed as follows: consider an input vector $\mathbf{x}_0 = \left(x_0^1, ..., x_0^F\right)$; the matching degree of the IF part, $\mu_A(\mathbf{x}_0)$, being a conjunction between several statements x_0^i is A_i, is thus written down as:

$$\mu_A(\mathbf{x}_0) = T(\mu_{A_{1,k_{m,1}}}(x_0^1), ..., \mu_{A_{F,k_{m,F}}}(x_0^F)) \tag{2.22}$$

The membership degree defining the output set is then recovered from Equation 2.21 as $\mu_{B'}(y) = I(\mu_A(\mathbf{x}_0), \mu_B(y)))$; here we assume a single output variable y. The outputs sets for the M rule in the system (in an FRBC there are M singleton fuzzy sets) are aggregated using an aggregator function and the output is defuzzified in order to provide a crisp value (*First Aggregate, Then Infer* (Luis, 2015)). Sometimes, the order is reversed and the input can be first defuzzified and then aggregated (*First Infer, Then Aggregate* (Luis, 2015)). A wide spectrum of diverse aggregation functions exists; the interested reader is referred to the literature (Cordón et al., 1999). In the following, the aggregation function suitable to a particular problem will be defined in the relevant section.

2.4 Evolutionary Fuzzy Systems

Since the beginning of the 1990s, the idea of combining fuzzy systems with evolutionary computation has been explored; from those first proposals the concept of Evolutionary Fuzzy System (EFS) had born. EFSs are a family of approaches built on the top of the top of the FRBSs described in the previous section. The components of the FRBS are improved (evolved) through an evolutionary process (Fernández et al., 2015). The example of an EFS, integrated on the top of a FRBS, is reported in Figure 2.6.

In the EFS setting, the design of the fuzzy system can be formulated as a search problem in high-dimensional space where each point represents a rule set and membership functions (Shi et al., 1999). Evolutionary algorithms (EAs) and genetic algorithms (GAs) have been employed to carry the optimisation out, thanks to their ability to deal with large search spaces and to find near-optimal solutions without a precise description of the problem (Fazzolari et al., 2013).

Typically, the design of an EFS begins specifying which components of the FRBS should undergo the optimisation process. As such, two specific categories of EFSs can be defined: tuning and learning. Furthermore, the objective optimised can either by a single objective (e.g. accuracy) or a trade-off between different objectives; in the latter, a multi-objective evolutionary algorithm (MOEA) is needed (Fazzolari et al., 2013).

In general, the EFS can either learn the KB components anew or tune the components of a preexisting KB. The KB learning can in turn be distinguished in: i) rule selection, ii)

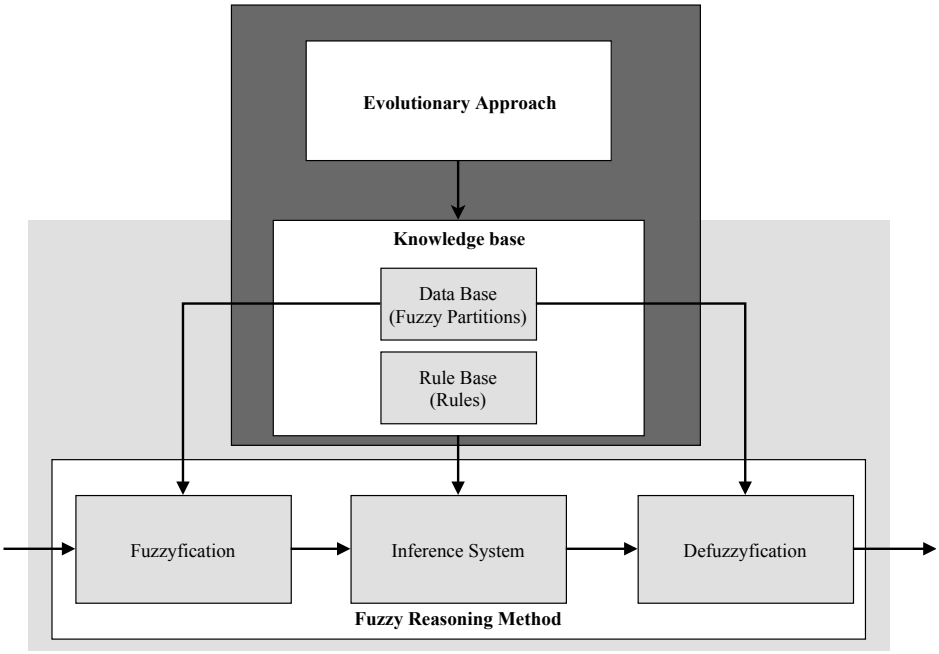


Figure 2.6: *Architecture of a generic fuzzy rule-based system with an EFS (dark grey) integrated to evolve the KB.*

simultaneous learning of the KB components, iii) rule learning and iv) DB learning. The evolutionary tuning can be further classified in: i) KB parameters tuning (such as tuning of the membership functions), ii) adaptive inference systems and iii) adaptive defuzzification methods.

In this thesis we primarily focus on rule (and condition) selection as well as on KB parameters tuning, using a MOEA that optimises a trade-off between accuracy and interpretability. A comprehensive taxonomy of EFSs is beyond the scope of this work, and can be found elsewhere (Fernández et al., 2015).

2.5 Randomness And Fuzzyness

Since its first steps, Fuzzy Logic has been often criticised for being just probability in disguise (Kosko, 1990); a fierce debate has been going on about whether probability is the sole sensible way to describe uncertainty. This view, often held by Bayesianists, obliquely challenged Fuzzy Set Theory, asserting that every other description of uncertainty but probability is inadequate (Lindley, 1987). There are, however, several conceptual and theoretical differences between the two theories.

Indeed, both fuzzyness and randomness deal with uncertainty; they both did so by quantifying uncertainty using numbers in the $[0, 1]$ interval. But a fundamental philosophical distinction lies hidden at their roots: fuzzyness describes the ambiguity of an event, i.e. the degree to which that particular event occurs, while probability deals with the un-

certainty in the occurrence of an event (*stochastic uncertainty* (Zimmermann, 2010)). In other words, while fuzzy logic deals with degrees of truth (*fuzziness* and partial or relative truths) probability is about making predictions about the state of an event given a state of partial knowledge.

Consider the example of describing the weather on the next day. A probabilistic approach would produce a statement of the form: "There is a 70% chance of having a shower tomorrow". A fuzzy logic statement is instead of the form: "There will be a 0.7 light shower (or a 0.2 heavy shower) tomorrow". In the probabilistic case, we assert that it is more likely to have a shower than not having a shower, without providing any information about how heavy the shower will be. On the other hand, the fuzzy set based approach tells us that for sure, there will be a shower; furthermore, information regarding the amount of rainfall is given as well.

As such fuzzy logic is not particularly apt to deal with partial knowledge and, on the other hand, probability theory does not capture the essential property of meaning (partial truth) which is the goal of fuzzy logic.

CHAPTER

THREE

ON BOOSTING APPROCHES FOR FUZZY DECISION TREES

> How is it that a committee of blockheads can somehow arrive at highly reasoned decisions, despite the weak judgment of the individual members? How can the shaky separate views of a panel of dolts be combined into a single opinion that is very likely to be correct?
>
> Boosting - Robert E. Schapire and Yoav Freund

This chapter contains material from the following publications:

- Barsacchi, M., Bechini, A., & Marcelloni, F. (2017). Multi-class boosting with fuzzy decision trees. In 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1–6).
  https://doi.org/10.1109/FUZZ-IEEE.2017.8015567
- Barsacchi, M. Bechini, A. and Marcelloni F. (2018). Using Fuzzy Decision Trees in Boosting: An Ensemble Multi-class Classifier and its Experimental Evaluation. *Under Review.*

In this chapter, we propose FDT-Boost, a boosting approach shaped according to the SAMME-AdaBoost scheme, which employs fuzzy binary decision trees as base classifiers. An experimental evaluation of FDT-Boost is carried out using sixteen classification benchmarks. Comparing FDT-Boost with FURIA, one of the most popular fuzzy classifiers, with a fuzzy binary decision tree, and with a fuzzy multi-way decision tree, we show that FDT-Boost is accurate, getting to results that are statistically better than those achieved by the other approaches.

This chapter is organised as follows. Section 3.2 recalls basic concepts on FDTs and fuzzy discretization. Section 3.3 is devoted to the description of the proposed FDT-Boost procedure. The results of the experimental assessment of FDT-Boost are reported in Section 3.4. Section 3.5 summarise the results.

3.1 Decision Trees and Ensembles

Among the most popular machine learning algorithms are decision trees. Their interpretability lies at the heart of their widespread diffusion, and the simplicity of their supervised learning process has contributed to their success as well (Quinlan, 1986). Yet another positive aspect of decision trees is that the tuning of its learning process involves only a very limited number of parameters, if any. Decision trees are constructed partitioning the feature space into a set of rectangles, and fitting a simple model in each one. The training dataset is recursively split: a tree node is associated with a subset of the whole dataset, and in turn, the subsets of its children correspond to a partition of the father's data subset. When traversing the tree from the root down to the leaves, subsets become increasingly homogeneous with respect to the label associated with the enclosed examples.

As discussed in the foregoing chapters, fuzzy systems have been often proposed for dealing with vague/uncertain information, (Klir and Yuan, 1995); among the various applications, the ones in the field of decision trees, through the proposal of *fuzzy decision trees* (FDTs) (Janikow, 1998), are among the most successful. Unlike classical decision trees, a node in an FDT corresponds to a *fuzzy* set, and each instance can activate more than one branch, reaching multiple leaves. In general, fuzzy classification leads to label an instance with multiple classes, each with a different confidence degree. Subsequently, FDTs may also serve as robust rankers (Hüllermeier and Vanderlooy, 2009).

The first fuzzy decision tree algorithm is attributed to Chang and Pavlidis, in 1977 (Chang and Pavlidis, 1977). Since then, many different fuzzy decision tree induction algorithms have been proposed (Chiang and jen Hsu, 2002), with successful applications in a growing number of fields (Boyen and Wehenkel, 1999; Crockett et al., 2017). Among them, the most common approaches are the Fuzzy ID3 Umanol et al. (1994) and the fuzzy SLIQ decision tree (Chandra and Varghese, 2008) algorithms. Other notable approaches are the soft decision trees (SDT) (Olaru and Wehenkel, 2003), that provides a complete method combining tree growing and pruning, to determine the structure of the soft decision tree, with refitting and backfitting, to improve its generalisation capabilities and look-ahead fuzzy decision trees (Dong and Kothari, 2001), that jointly optimise the node splitting criterion (information gain or gain ratio) and the classifiability of instances along each branch of the node. C-fuzzy decision trees Pedrycz and Sosnowski (2005) exploit information granules (multivariable entities characterised by high homogeneity) developed via fuzzy clustering and use them in the growth of the decision trees, The literature around FDT is still lively: a recent contribution explored fuzzy rule based decision tree (FRDT), whose nodes involve a fuzzy rule which in turn involves multiple features (Wang et al., 2015) in a way

alike to oblique decision trees (Manwani and Sastry, 2012). Multi flexible fuzzy decision tree (Isazadeh et al., 2016) has been recently proposed to handle online streams of data, while hesitant fuzzy decision trees, based on the concept of hesitant fuzzy sets, have been successfully applied in the case of imbalance classification (Sardari et al., 2017). Another recent contribution has investigated the application of fuzzy decision trees in the big data setting (Segatori et al., 2017b) with promising results.

In a classification problem, a good classifier should be able to predict the right class with satisfying accuracy. In order to achieve high performances, a proper learning process has a primary role. A classifier is usually defined as either *binary* or *multi-class*, depending on the so-called arity of categorical feature to be considered in the defined problem (two or more, respectively). Beginning with the remarkable results on the possibility to improve the performance of any mediocre classifier found by Schapire (Schapire, 1990), several other approaches have been suggested to build one accurate predictive model by fusing together multiple models (Rokach, 2010). In *ensemble methods*, a set of different *base learners* is generated, and used for classification by collecting their predictions to determine a unique, agreed result. Among them, *boosting* plays a prominent role (Freund and Schapire, 1997; Schapire and Freund, 2012). Boosting is also known in the literature as "arcing", i.e. Adaptive Resampling and Combining (Breiman, 1998). In this approach, base classifiers are generated iteratively, and the base classifiers generated in previous iterations are used to properly manipulate the complete training set so to derive the specific training set to be used in the next iteration (Schapire and Freund, 2012). Notably, boosting with decision trees as base learners has been soon considered as one of the most effective choices (Breiman, 1998). As decision trees are renown for their handiness, it is not surprising that they have often been chosen as base learners in boosting procedures (Roe et al., 2005; De'ath, 2007)

In the field of binary classification, AdaBoost is a popular boosting meta-algorithm known to be simple yet very effective (Freund and Schapire, 1997). It can operate with any kind of base learner (that is, obtained by any learning algorithm), and its tuning is based on one single parameter. Anyway, many real-world problems involve multiple classes, and the employment of AdaBoost in such contexts asks for specific algorithmic adaptations. Since the introduction of AdaBoost, the proposed multi-class boosting procedures have been shaped according to two main schemes (Saberian and Vasconcelos, 2011). According to the first one, named *binary reduction*, a multi-class problem is recast to a combination of a number of binary sub-problems, each to be solved with the original AdaBoost. Unfortunately, binary reduction is subject to a number of problems (Zhai et al., 2014). The second option plans to directly make use of native multi-class base classifiers, and use them in the context of a boosting procedure. Theoretical aspects of this option have been thoroughly investigated (Mukherjee and Schapire, 2013), and SAMME-AdaBoost (Hastie et al., 2009) can be regarded as the most used method that follows this approach.

Moreover, such an approach has been paired with a pre-discretization phase of numerical attributes. The very preliminary results obtained in this attempt lead us to the

development of "FDT-Boost" classifiers, which are fully described in this work for the first time. In this context, the use of size-constrained binary FDTs let us bound the complexity of the overall model. Moreover, here a strong assessment of the performance of the algorithm let us understand its real potential. It is important to underline that a fair comparison with other models has to take into account not only the classification accuracy but also the model complexity and the ability to deal with noisy data.

In this chapter, after a detailed description of the FDT-Boost approach, we experimentally investigate on the characteristics of its learning process and on its ability to deliver accurate and robust results, in a cross-comparison with other state-of-the-art fuzzy classifiers. Moreover, we show that the complexity of models generated by FDT-Boost is lower than their non-fuzzy counterparts, created by using SAMME-AdaBoost with classical multi-class binary decision trees.

3.2 From DTs to FDTs

In this section we recall and expand the notation described in Chapter 2. Let each instance $\mathbf{x} = [x_1 \ldots, x_F]$ be characterised by a set $X = \{X_1, \ldots, X_F\}$ of F features (or "attributes"). Let y be a categorical label that takes values out of the set of M classes $\{C_1, \ldots, C_M\}$. We define the training set as $TR = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$, where N is the number of labelled instances. We recall that in each iteration of the boosting procedure the used training set is obtained by sampling TR according to a given discrete distribution; hence, the actual "sampled" training set used in the t-th iteration is indicated in the following as $TR^{(t)}$. Where used, the notation $\hat{\mathbf{x}}$ refers to an unlabelled instance, to which a label must be assigned by the classifier.

A decision tree approximates a discrete-valued function in the feature space; it does so by inducing a hierarchical partition of the feature space, utilising the information contained in the training set. The partition corresponds to a multistage decision system that is followed in a sequential manner.

The partitioning procedure follows a recursive approach: each set, corresponding to a node in the tree, is partitioned into k subsets until a termination condition is met (Janikow, 1998). If $k = 2$, decision trees are said *binary*, and for $k > 2$ they are said *multi-way*. Interestingly, learnt trees can also be re-represented as sets of if-then rules; this feature will be exploited in later chapters (Chapter 5 and 6). Fuzzy decision trees (FDTs) are built by generalising the idea to the fuzzy context (Altay and Cinar, 2016).

At first, as the learning procedure was defined for *categorical* attributes, *continuous* attributes required a discretization, to be performed either before or along the tree construction; notably, it is the case for the well-known ID3 algorithm (Quinlan, 1986) and for CART (Breiman, 1993), respectively. Furthermore, Fayyad and Irani have shown that the application of entropy-based heuristics can lead to effective prior multi-interval discretization (Fayyad and Irani, 1993a).

All the different algorithms for decision trees induction rely on a core algorithm that

employs a top-down, greedy search through the space of possible decision trees. At each node, the splitting that generates the relative children depends on the local choice of a discriminating attribute. In the specific context of FDTs, the attribute selection may resort to different types of metrics; among them, the most frequently used are the fuzzy Kolmogorov-Smirnov discrimination quality measure (Boyen and Wehenkel, 1999), the minimal ambiguity of a possibility distribution (Yuan and Shaw, 1995), the fuzzy Gini index (Chandra and Varghese, 2008), and the maximum classification importance of attribute contributing to its consequent (Wang et al., 2001). In more recent years, fuzzy information (Zeinalkhani and Eftekhari, 2014) became one of the most popular.

The tree generation goes on branching until a termination condition occurs. Usually, the termination condition is the logical disjunction of several *stopping predicates* that account for different kinds of imposed constraints. In the following, the chosen stopping conditions are:

- the node contains less than n_{min} instances;
- the node contains only instances of the same class;
- the fuzzy information gain for the split is lower than a threshold ε
- the maximum tree depth β has been reached.

The choice between binary and multi-way decision trees determines different properties for the final tree: in *multi-way* splitting of a node, one child is generated for any distinct linguistic value defined over the current splitting attribute. Subsequently, in multi-way DTs, no single attribute is tested twice on a path from the root to a leaf, while this is not guaranteed in binary DTs.

The distinct properties of classical and fuzzy sets, outlined in Chapter 2, leads to a different classification process between DTs and FDTs. Given an unlabeled instance $\hat{\mathbf{x}}$, a classical decision tree is traversed following a unique path from the root node to a leaf, thus outputting a single class. When considering an FDT, an instance $\hat{\mathbf{x}}$ can, by definition, belong to different fuzzy sets with different membership degrees. Consequently, when traversing the tree from the root downwards, multiple paths are activated, getting to multiple leaf nodes. We can thus express the activation level of a leaf by its *matching degree*. Considering a generic node $\mathcal{N}$ whose parent node is $\mathcal{PN}$, we define the matching degree $md^{\mathcal{N}}(\hat{\mathbf{x}})$ of instance $\hat{\mathbf{x}}$ with $\mathcal{N}$ as:

$$md^{\mathcal{N}}(\hat{\mathbf{x}}) = T(\mu^{\mathcal{N}}(\hat{x}_f), md^{\mathcal{PN}}(\hat{\mathbf{x}})) \tag{3.1}$$

where T is a T-norm (as described in Chapter 2), $\mu^{\mathcal{N}}(\hat{x}_f)$ is the membership degree of $\hat{x}_f$ to node $\mathcal{N}$ considering X_f as splitting attribute, and $md^{\mathcal{PN}}(\hat{\mathbf{x}})$ is the matching degree of $\hat{\mathbf{x}}$ to $\mathcal{PN}$, being each node associate with a fuzzy set.

At each activated (i.e. having *matching degree geq*0) leaf node $\mathcal{LN}$, an instance $\hat{\mathbf{x}}$ can be labelled to a certain extent with a class C_m. We can compute the *association degree* $AD_m^{\mathcal{LN}}(\hat{\mathbf{x}})$

accordingly:

$$AD_m^{\mathcal{LN}}(\hat{\mathbf{x}}) = md^{\mathcal{LN}}(\hat{\mathbf{x}}) \cdot w_m^{\mathcal{LN}} \tag{3.2}$$

where $w_m^{\mathcal{LN}}$ is the *class weight* associated with class C_m at leaf node $\mathcal{LN}$. The usage of class weights is supported by a series of works, that have shown how they improve the performances in fuzzy classifiers (Ishibuchi et al., 2005).

Finally, in order to compute the class weights, we proceed as follows. Let G be the set of training examples represented in the leaf node, andG_{C_m} its subset of elements labelled with class C_m; we define the class weight $w_m^{\mathcal{LN}}$ as:

$$w_m^{\mathcal{LN}} = \frac{|G_{C_m}|}{|G|}. \tag{3.3}$$

being $|\cdot|$ the set cardinality. The actual output class label is finally obtained by the classifier by combining the *association degrees* for all the leaves in the FDT. The different possible ways of combing the *association degrees* are discussed elsewhere (Section 2.3).

3.3 The approach: FDT-Boost

The structure of the approach follows the SAMME-AdaBoost scheme; a set of base learners is trained: for each learner, we use a fuzzy entropy-based discretizer to induce a fuzzy partition for continuous attributes.

In the following subsections, the components phases of FDT-Boost are thoroughly described; they are: i) the fuzzy discretizer, ii) the construction of a weak learner, i.e. the fuzzy binary decision tree (FBDT), and iii) the ensembling using SAMME-AdaBoost.

3.3.1 (Fuzzy) Discretization for Continous Attributes

The majority of fuzzy approaches assume that a fuzzy partition is defined on each continuous attribute; the fuzzy decision tree we employ is no exception. As such, a discretizer, which adapts a solution from the literature (Segatori et al., 2017b), is employed. This discretization approach is based a fuzzy generalisation of what has been proposed by Fayyad and Irani (Fayyad and Irani, 1993a), and is built on a recursive evaluation of the fuzzy entropy associated with possible fuzzy partitions. Each continuous attribute is recursively partitioned using strong triangular fuzzy partitions.

The choice of the discretizer has been partially driven by its inherent ability to perform *feature selection*; if no partition has been generated at the termination point while splitting the target domain of a given attribute, such an attribute can be discarded.

Let us define a generic training set as *TR*. Wherever the training set varies across iteration, as in the boosting procedure, $TR^{(t)}$ is used to represent the training set at the t-th iteration.

Let X_f be the f-th attribute, and thus let $x_{f,i}$ be the value of X_f in the i-th sample in *TR*. Hereafter is assumed, without loss of generality, that all the values $x_{f,i}, i = 1 \dots N$,

are sorted in ascending order. Let I_f be a generic interval on the universe of discourse U_f for the feature X_f, and let S_f be the relative set of examples in TR whose values for feature X_f fall in I_f, i.e. the support of I_f. A fuzzy partition over I_f is represented as $P_{I_f} = \left\{B_{f,1}, \ldots B_{f,|P_{I_f}|}\right\}$, being $B_{f,p}$ the p-th fuzzy set defined over a sub-interval of I_f, and $|P_{I_f}|$ the number of fuzzy sets in the given partition.

For each fuzzy set $B_{f,p}$, the related *support set* $S_{f,p}$ is defined as the subset of the elements of TR that are members of $B_{f,p}$ or, more formally, those whose values for the f-th feature have a membership degree to $B_{f,p}$ strictly greater than zero: $S_{f,p} = \left\{\mathbf{x} \mid \mathbf{x} \in TR \ \wedge \ \mu_{B_{f,p}}(x_f) > 0\right\}$.

The number of elements in the support set, $|S_{f,p}|$, is known as *support value* or simply *support*, and the *fuzzy cardinality* $|B_{f,p}|$ of a fuzzy set $B_{f,p}$ is defined as:

$$|B_{f,p}| = \sum_{j=1}^{|S_{f,p}|} \mu_{B_{f,p}}(x_{f,j}) \tag{3.4}$$

where $x_{f,j}$ is the value for the f-th feature of the j-th element in $S_{f,p}$.

The definition of *fuzzy entropy* for a fuzzy set $B_{f,p}$ strictly follows the definition of entropy:

$$FEnt(B_{f,p}) = -\sum_{m=1}^{M} \frac{|B_{f,p,C_m}|}{|B_{f,p}|} \log_2 \left(\frac{|B_{f,p,C_m}|}{|B_{f,p}|}\right) \tag{3.5}$$

where $|B_{f,p,C_m}|$ represents the fuzzy cardinality of the set $B_{f,p}$ restricted to just examples with class label C_m.

Given a fuzzy partition P_{I_f} over a given interval I_f, the *weighted fuzzy entropy* $WFEnt(P_{I_f}; I_f)$ can be defined as the weighted average of the *fuzzy entropy* for all the fuzzy set pertaining to the partition:

$$WFEnt(P_{I_f}; I_f) = \sum_{i=1}^{|P_{I_f}|} \frac{|B_{f,i}|}{|S_f|} FEnt(B_{f,i}). \tag{3.6}$$

The definition of *weighted fuzzy entropy* suggests a straightforward way for comparing different partitions over the same interval I_f. Given a "base" partition P_{I_f} we can quantify the effect of the substitution with another partition P'_{I_f} by means of the fuzzy information gain defined as follows:

$$FGain(P'_{I_f}; I_f) = WFEnt(P_{I_f}; I_f) - WFEnt(P'_{I_f}; I_f). \tag{3.7}$$

The discretization algorithm proceeds by recursively partitioning each numeric feature on the training set TR. For each numeric feature, the initial partition is made up of a pair of triangular "fuzzy" set. A generic partitioning step is described as follows: given a generic sub-interval I_f, it is partitioned with a partition formed of three triangular fuzzy sets, fully described by an internal "cut-point" (see Figure 3.1). The selected cut-point induces two sub-intervals I_f^1 (left) and I_f^2 (right) of I_f, and the procedure is recursively applied to them.

Given an interval I_f, along with its relative current partition P_{I_f}, the discretization procedure choose the cut-point that yields the further partition P'_{I_f} of I_f with the most

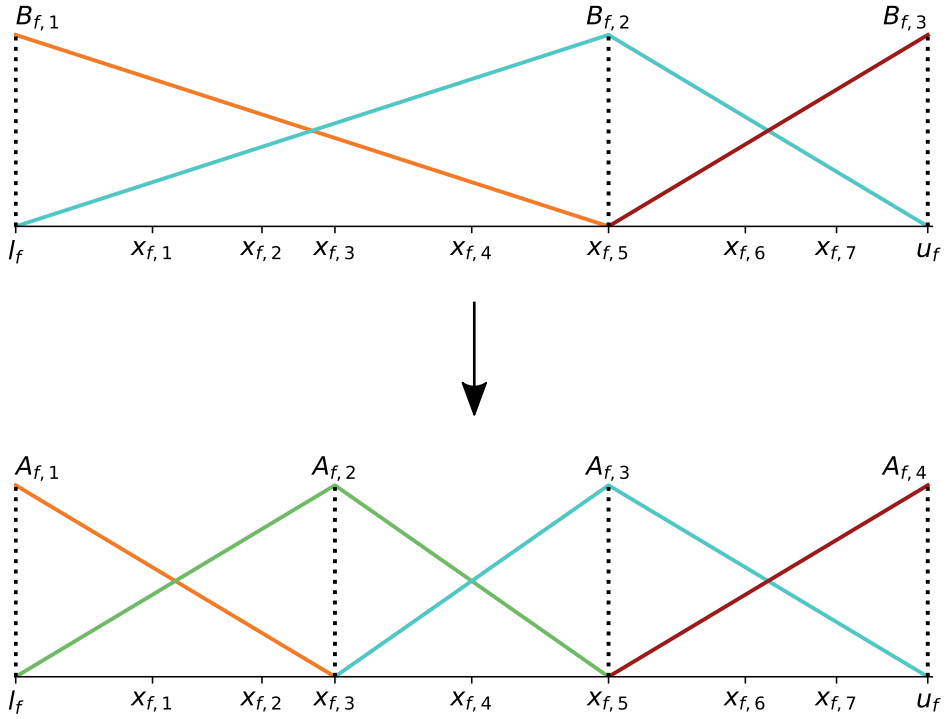


Figure 3.1: *An example of the application of the fuzzy discretization procedure for attribute X_f on $I_f = [l_f, u_f]$. First, the procedure selects $x_{f,5}$ as cut point, and generates the triangular fuzzy partition shown in the upper panel. Then, it selects $x_{f,3}$ as cut point for the interval $[l_f, x_{f,5}]$ thus generating the triangular fuzzy partition shown at the bottom.*

favourable *FGain*. Whenever the information gain falls below a given threshold, the partition is not applied (this is the stopping condition for the recursive procedure). Figure 3.1 report an example of two steps of the described procedure on a given interval, and then on its obtained left sub-interval.

The stopping condition can be formally expressed as

$$FGain(P'_{I_f}; I_f) < \frac{\log_2(|S_f| - 1)}{|S_f|} + \frac{\Delta(P'_{I_f}; I_f)}{|S_f|} \tag{3.8}$$

being $\Delta(P'_{I_f}; I_f)$ calculated as:

$$\begin{aligned} \Delta(P'_{I_f}; I_f) = \log_2(3^{k_f} - 2) - [&k_{I_f} \cdot WFEnt(P_{I_f}; I_f) \\ &- k_{I_f^1} \cdot WFEnt(P'^{1}_{I_f}; I_f^1) \\ &- k_{I_f^2} \cdot WFEnt(P'^{2}_{I_f}; I_f^2)] \end{aligned} \tag{3.9}$$

where k_{I_f} is the number of class labels over the whole interval I_f, and $k_{I_f^1}$, $k_{I_f^2}$ are the numbers of class labels over the subintervals I_f^1 and I_f^2, respectively.

If the stopping condition is immediately met in the first step of the discretization procedure for an attribute, that attribute does not influence the class label (at least in a univariate

fashion), and it is thus discarded. In that in datasets with up to a few hundred examples the proposed stopping condition often leads to an excessive selection of features, while in large datasets the possible problem is an unreasonable discretization level.

In order to cope with this limitation, a heuristics has been implemented to obtain a more effective discretization process. The number of splits can be limited to a maximum s_{max}, i.e. constraining the number of generated fuzzy sets. This can be obtained by tracing back the recursive splitting procedure, and selecting the top s_{max} splits in terms of fuzzy entropy gain, yet keeping the partitioning procedure consistent. This possibility is due to the additive formulation of entropy, and it needs only to keep track of the splitting tree; then the splitting tree can be traversed one step at a time, keeping at each iteration the node with the highest gain, among the available ones.

3.3.2 Fuzzy Decision Tree

The proposed approach builds an ensemble of weak fuzzy decision trees (FDTs); the decision trees are built following the overall indications reported in (Segatori et al., 2017b). Each FDT is built resorting to a recursive node-splitting procedure that selects the split that maximises the relative fuzzy information gain and is subject to a set of stopping conditions as well. The pseudocode for the learning algorithm is reported in Figure 3.2.

The construction begins considering a root node containing all the data in the training set $\widehat{TR}$. As such, the root node is associated with a fuzzy set containing all the data with a membership degree of 1. Then, the recursive SplitNode function is called on the root node. SplitNode first verifies whether at least one of the stopping conditions over the current node is satisfied. In our work, we used the stopping conditions described in Section 3.2.

If at least one of the stopping conditions is satisfied, the node is labelled as a leaf and the recursive branching is stopped. Contrariwise, the SelectSplit function is called; the function generates two child nodes, choosing the split that maximises the fuzzy information gain *FGain* (Eq. 3.7) referring to the partitioning of the node fuzzy set induced by the node split. The best split is found iterating over all possible splits for all the possible features, and the split with the highest *FGain* is returned.

In order to be able to perform a *binary* splitting on a partition $\mathbf{P}$ (being $\mathbf{P}$ the outcome of the discretization described in Section 3.3.1) $|\mathbf{P}| - 1$ candidates must be evaluated, being $|\mathbf{P}|$ the number of fuzzy sets in $\mathbf{P}$. Each candidate binary split is obtained by merging the first adjacent k fuzzy sets in $\mathbf{P}$ to form the first part, and all the remaining fuzzy sets to form the second one, leading to the definition of two trapezoidal fuzzy sets. Figure 3.3, portrays an example of candidate binary splits that correspond to the partition generated in the lower panel of Figure 3.1.

Categorical attributes are dealt with according to a one-vs-all approach. When the function SelectSplit is called, all the possible candidates are evaluated over all the attributes, and the best split made of the two fuzzy sets $Z_{f,1}$ and $Z_{f,2}$ is returned. Based on the outcome of SelectSplit, two child nodes, G_1 and G_2, are generated by calling the Membership

```
1: function LearnFDT(TR̂, P, β, n_min, γ, η)
2:     Create a root node with all the set of data TR̂, i.e. a fuzzy set of all the data, with
   all the membership values = 1.
3:     tree ← SplitNode(node, TR̂, P, β, n_min, γ, η )
4:     return tree
5: end function
6: function SplitNode(node, X, P, β, n_min, γ, η)
7:     if StoppingCondition(node, β, n_min, γ, η) then
8:         node ← mark node as leaf
9:     else
10:        splits ← SelectSplit(X, P)
11:        for split_k in splits do
12:            child_k, X_k ← Membership(X, P, split_k)
13:            child_k ← SplitNode(child_k, X_k, P, β, n_min, γ, η)
14:            connect child_k with node
15:        end for
16:    end if
17:    return node
18: end function
```

Figure 3.2: *Pseudocode of the tree construction procedure LearnFDT. As input, the algorithm requires the complete training set $\widehat{TR}$, the discretization* **P***, and the parameters β, n_{min}, γ, and η. The output is the FBDT.*

function; it assigns to each node the examples belonging to the support of the respective fuzzy sets, with a membership value computed as the t-norm of the membership on the set and the membership on the father node. In all the experiments the product t-norm has been used.

3.3.3 Boosting scheme: SAMME-AdaBoost

The boosting procedure used in the proposed classifier is aimed at generating an ensemble of base classifiers (namely binary FDTs) by following the SAMME-AdaBoost algorithm described in (Hastie et al., 2009). SAMME-AdaBoost extends AdaBoost to the multi-class case, avoiding to reduce the classification problem to multiple two-class problems. It is worth recalling that SAMME just requires a base-learner to provide better predictions than an M-class random guess.

In the following, the overall procedure will be referred to as *FDT-Boost*; the relative pseudo-code is presented in Figure 3.4. The algorithm requires a base model (FBDT), a fixed number of iterations T, a constraint on the maximum number β of levels of the base model, the minimum number n_{min} of instances in a node, the minimum value γ of the

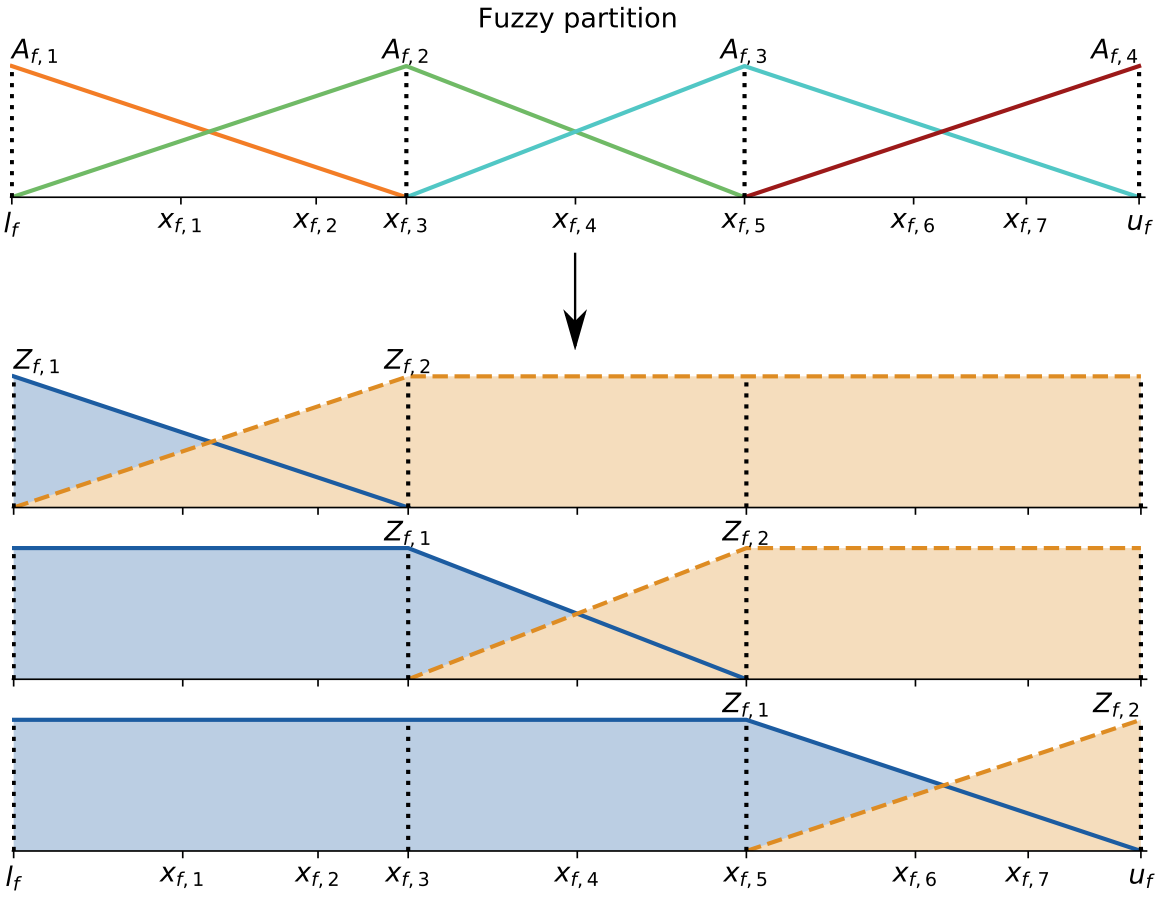


Figure 3.3: *The application of the binary splitting procedure for attribute X_f on $I_f = [l_f, u_f]$ is shown. All possible candidates obtained by grouping together adjacent fuzzy sets into two disjoint groups are tested, producing the trapezoidal fuzzy partitions shown at the bottom (Z_{f_1} and Z_{f_2}).*

proportion of examples of a class in a node, the minimum threshold value η for the fuzzy information gain, and the maximum number s_{max} (for the initial discretization). The base model, described in the previous sub-section, is grown up to its maximum depth β, with no post-pruning. The optimal values for the parameter T and β depend on the characteristics of the specific problem; further details are given below (Section 3.4).

Following the notation introduced before, the boosting procedure starts from considering the original training set with N samples,
$TR = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$, and one fuzzy partition (or "discretization") for each of the numerical attributes $\mathbf{X} = \{X_1, X_2, ..., X_F\}$ as generated by the fuzzy discretizer (Section 3.3.1) ; it gives back $\mathbf{P}$, a set of partitions for the attributes that have not been filtered out during the discretization process.

The ensemble of T classifiers is built up by means of an iterative procedure; at each iteration t, the actual training set $TR^{(t)}$ to work on (still with N examples) is obtained by

function FDT-BOOST(TR, T, β, n_{min}, γ, η, s_{max})
  $\mathbf{P} \leftarrow$ FUZZYDISCRETIZER($\mathbf{X}$, TR, s_{max})
  $BoostEns \leftarrow \varnothing$
  $\mathbf{w} = \{w_i \leftarrow 1/N, i = 1, \ldots N\}$
                                                ▷ Uniform sampling
  for $t \leftarrow 1$ to T **do**
    $TR^{(t)} \leftarrow$ WEIGHTEDSAMPLING(TR, $\mathbf{w}$)
                                                ▷ TR is sampled using weight vector $\mathbf{w}$
    $FDT^{(t)} \leftarrow$ LEARNFDT($TR^{(t)}$, $\mathbf{P}$, β, n_{min}, γ, η)
                                                ▷ The t-th Fuzzy Decision Tree is built
    $eer^{(t)} \leftarrow$ computation of Eq. 3.10
    $\alpha^{(t)} \leftarrow$ computation of Eq. 3.11
                                                ▷ Tree weight $\alpha^{(t)}$ depends on error rate $eer^{(t)}$
    $BoostEns \leftarrow BoostEns \cup (FDT^{(t)}, \alpha^{(t)})$
    $\mathbf{w} = \{w_i \leftarrow$ value as per Eq. 3.12, $i = 1, \ldots N\}$
    $\mathbf{w} \leftarrow \mathbf{w}/\sum_{i=1}^{N} w_i$
                                                ▷ Sample weight vector updated & normalized
  end for
  return *BoostEns*
end function

Figure 3.4: *Pseudocode for the overall boosting procedure* FDT-Boost. *As input, the algorithm requires the complete training set TR and all the model parameters. The output is the ensemble of FDTs.*

sampling TR according to the current sampling distribution, defined by its weight vector $\mathbf{w}$ with N components. In the first iteration, $TR^{(1)} = TR$. A base model $FDT^{(t)}$ is induced out of a dataset $TR^{(t)}$, and a weighted error rate $eer^{(t)}$ is computed as:

$$eer^{(t)} = \frac{\sum_{i=1}^{N} w_i \cdot \mathcal{I}(y_i \neq FDT^{(t)}(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i} \tag{3.10}$$

being $\mathcal{I}(\texttt{true}) = 1$ and $\mathcal{I}(\texttt{false}) = 0$, and $FDT^{(t)}(\mathbf{x}_i)$ the classification output of the t-th FDT for the i-th sample in TR.

The contribution of the t-th model $FDT^{(t)}$ to the boosted ensemble is accounted by the weight $\alpha^{(t)}$:

$$\alpha^{(t)} = \log \frac{1 - eer^{(t)}}{eer^{(t)}} + \log(M - 1). \tag{3.11}$$

The $\log(M - 1)$ term (M indicates the number of classes), is necessary in order to avoid negative values for α in the multiclass case for better-than-random guesses, as stated by the original SAMME algorithm (Hastie et al., 2009).

The sample weights are updated at each iteration, by increasing the values of samples that have been misclassified by the last built FDT:

$$w_i^{(t+1)} = w_i^{(t)} \cdot e^{\left[\alpha^{(t)} \cdot \mathcal{I}\left(y_i \neq FDT^{(t)}(\mathbf{x}_i)\right)\right]}, i = 1, \ldots N. \tag{3.12}$$

A normalization across all the sample weights is then applied.

After the completion of the whole learning phase, the ensemble prediction for any unlabeled example $\hat{\mathbf{x}}$ is calculated taking into account all the votes from all the base learners in the ensemble. In the proposed fuzzy algorithm, each base $FDT^{(t)}$ produces as output, for each class C^m, a relative vote C_t^m. The ensemble-wide classification $Cl(\hat{\mathbf{x}})$ for $\hat{\mathbf{x}}$ consists in the single class that gets the highest cumulative vote, considering also the weights associated with the trees:

$$Cl(\hat{\mathbf{x}}) = \arg\max_m \sum_{t=1}^{T} \alpha^{(t)} \cdot C_t^m \tag{3.13}$$

.

3.4 Experimental Comparison

In all the experiments performed in this section, a binary FDTs has been selected as a base model for FDT-Boost. The maximum tree depth has been fixed to 4, i.e. $\beta = 4$, and the number of boosting iterations has been fixed to $T = 500$. The complete parametrization is reported in Table 3.1. These parameters reflect an acceptable tradeoff between accuracy and model complexity, according to experimental results reported hereafter in Subsection 3.4.3.

Table 3.1: *Parameters used in FDT-Boost and their values*

Parameter	Description	Value
T	number of boosting iterations	500
β	maximum tree depth	4
n_{min}	min # inst. per leaf	2
γ	max fraction inst. per leaf	1.0
η	min gain	0.0001
s_{max}	max number of fuzzy sets	7

The set of experiments have been performed on a set of 16 datasets gathered from the KEEL repository[1]. Table 3.2 summarises their reporting cardinality, number of attributes, and number of classes. In order to perform a cross-validation, we used the pre-specified 5-fold cross-validation splitting.

The experimental section is organised as follow: first, a comparison with a set of fuzzy classifiers is performed; the results are reported in Subsection 3.4.1. Then, FDT-Boost is

[1] `http://sci2s.ugr.es/keel/datasets.php`

Table 3.2: *Datasets used in the experimental phase, listed in alphabetical order*

Dataset (init.)	Cardinality	# Attr.	# Classes
appendicitis (APP)	106	7	2
australian (AUS)	690	14	2
bands (BAN)	365	19	2
dermatology (DER)	358	34	6
glass (GLA)	214	9	7
hayes (HAY)	160	4	3
iris (IRI)	150	4	3
mammographic (MAM)	830	5	2
newthyroid (NEW)	215	5	3
segment (SEG)	2310	19	7
tae (TAE)	151	5	3
vehicle (VEI)	846	18	4
vowel (VOW)	990	13	11
wdbc (WDC)	569	30	2
wine (WIN)	178	13	3
wisconsin (WIS)	683	9	2

compared to crisp SAMME-AdaBoost in Subsection 3.4.2. Finally, in Subsection 3.4.3 an investigation of the convergence behaviour of the proposed model is performed, in order to get suggestions for a correct parametrization and to study possible overfitting issues.

3.4.1 Fuzzy Classifiers

In this section, the results of an experimental comparison of FDT-Boost and a set of state-of-the-art fuzzy classifiers are described; we selected FBDT, a fuzzy binary decision tree (Segatori et al., 2017b), FMDT, a fuzzy multi-way decision tree (Segatori et al., 2017b) and FURIA, a fuzzy rule-based classifier (Hühn and Hüllermeier, 2009). For what concerns the parametrization of FBDT and FMDT, we selected the value according to the indications reported in (Segatori et al., 2017b); the maximum depth has been fixed to 15 for FBDT and 5 for FMDT. FURIA has been used as provided in WEKA (Eibe et al., 2016) with its default parametrization.

The average accuracies and their standard deviations obtained on the test set on all the selected datasets are reported in Table 3.3.

In order to detect statistically relevant differences among the accuracies achieved by the different approaches, a set of statistical tests have been performed. First, the distribution with the average accuracies evaluated on the test set for all the datasets are defined, and then non-parametric statistical tests are applied. The Friedman test (Friedman, 1937), allows calculating the rankings among the distribution; the mean ranks are reported in

Table 3.3: *Average accuracy and standard deviation achieved by FDT-Boost, FURIA, FBDT and FMDT over the test set (5-fold cross-validation)*

Datasets	Algorithms			
	FDT-Boost	FURIA	FBDT	FMDT
appendicitis	85.80 ± 7.43	86.84 ± 5.38	83.07 ± 2.14	**87.71** ± 4.57
australian	84.49 ± 0.35	**85.22** ± 2.57	83.91 ± 1.34	84.20 ± 2.57
bands	**74.51** ± 1.28	67.20 ± 6.33	67.93 ± 1.64	66.85 ± 3.09
dermatology	**97.78** ± 2.23	95.81 ± 1.79	94.69 ± 2.20	93.01 ± 1.88
glass	**74.31**± 6.73	70.55 ± 3.53	71.10 ± 5.07	71.95 ± 8.66
hayes	**85.00** ± 5,00	81.87 ± 4.16	76.87 ± 4.24	60.62 ± 4.24
iris	**94.67** ± 2.49	93.33 ± 4.71	94.00 ± 3.89	**94.67** ± 2.50
mammographic	**83.59** ± 1.63	83.57 ± 2.44	80.55 ± 2.00	80.44 ± 1.43
newthyroid	94.88 ± 3.42	94.42 ± 2.37	93.49 ± 6.31	**96.28** ± 6.31
segment	96.80 ± 1.05	**97.24** ± 0.54	96.79 ± 0.63	95.67 ± 0.51
tae	**53.01** ± 7.48	45.63 ± 5.06	48.30 ± 7.71	51.61 ± 4.87
vehicle	**72.21** ± 3.11	68.09 ± 1.78	71.87 ± 1.98	72.11 ± 1.71
vowel	86.46 ± 2.93	79.80 ± 2.09	93.43 ± 1.94	**94.44** ± 1.59
wdbc	**97.18** ± 1.02	95.96 ± 2.12	95.08 ± 1.71	95.08 ± 1.18
wine	**98.87** ± 1.38	93.78 ± 3.37	94.38 ± 2.79	95.50 ± 2.52
wisconsin	97.01 ± 0.79	96.05 ± 0.58	96.49 ± 1.44	**97.36** ± 0.53
	86.04 ± 3.02	83.46 ± 3.05	83.87 ± 2.94	83.59 ± 2.74

Table 3.4. The p-value relative to the Iman and Davenport test (Iman and Davenport, 1980) is also reported. These tests reveal whether statistical differences between the distribution exists.

The Iman and Davenport p-value is marked as significant if it is below the significance level, here assumed as the ordinary value 0.05; if this is the case, the null hypothesis is rejected, and the claim statistical differences among the distributions can be made.

Table 3.4: *Outcome of the Friedman statistical test on the accuracy obtained by FDT-Boost, FURIA, FBDT, and FMDT. The Iman-Davenport p-value is also reported.*

Algorithm	Mean Rank	Iman-Davenport p-value
FDT-Boost	1.5312	0.00191
FURIA	2.875	
FBDT	3.0625	
FMDT	2.5312	

Among the selected algorithms FDT-Boost is the one with the lowest mean rank. Furthermore, the results in Table 3.4 suggests that given the fact that p-value $= 0.00191 < 0.05$, the null hypothesis of statistical equivalence can be rejected. In order to unravel statistical differences among the pairs of approaches, an Holm post hoc test have been performed; in the analysis FDT-Boost has been used as control algorithm, being the one with the smaller rank. The results are reported in Table 3.5; statistical differences exist between FDT-Boost and all the other approaches.

Table 3.5: *Results of the pairwise Holm post hoc test for significance level* $= 0.05$

Algorithm	z-value	p-value	Holm	Hypothesis
FBDT	3.54801	0.000794	0.016667	Rejected
FURIA	2.94401	0.00324	0.025	Rejected
FMDT	2.19089	0.02846	0.05	Rejected

3.4.2 Comparing with Crisp AdaBoost

In this section, experimental comparisons are performed in order to investigate whether a *fuzzy* base learner, combined with an interpretable discretizer, outperforms SAMME-AdaBoost when using "crisp" decision tree as base learner. First, FDT-Boost and a SAMME-AdaBoost classifiers are compared on the datasets described in the previous section; the results are reported in Table 3.6.

At first glance, the two approaches show comparable classification accuracies. Nevertheless, measuring the complexity of the resulting models, FDT-Boost seems to produce significantly less complex models. In order to use a complexity measure that is consistent

Table 3.6: *Average accuracy achieved by FDT-Boost and crisp AdaBoost on the test set for the 5-fold cross-validation and average number of nodes per base learner.*

Datasets	Accuracy (%)		Number of nodes	
	FDT-Boost	AdaBoost	FDT-Boost	AdaBoost
APP	**85.80**	84.89	**10.48**	20.56
AUS	84.49	**86.66**	**7.93**	28.14
BAN	74.51	**75.02**	**19.53**	26.54
DER	**97.78**	96.36	**14.73**	17.23
GLA	74.31	**77.10**	**25.66**	26.76
HAY	**85.00**	84.38	**9.09**	20.08
IRI	**94.67**	91.19	**7.85**	15.00
MAM	**83.59**	78.63	**1.88**	13.96
NEW	94.88	**95.81**	19.78	**14.66**
SEG	96.80	**98.18**	27.82	**24.11**
TAE	53.01	**55.03**	**15.15**	28.55
VEI	72.21	**77.54**	29.02	**28.77**
VOW	86.46	**91.12**	29.77	**28.24**
WDC	**97.18**	91.17	**22.44**	24.56
WIN	**98.87**	95.46	15.35	**15.00**
WIS	97.01	**97.37**	**16.47**	23.27
	86.04	85.99	**15.15**	22.21

across both systems, model complexity is defined as the average number of non-empty nodes across the whole ensemble; furthermore, the same tree depth and the same number of iterations are used, in order to perform a fair comparison. Analysing the results in Table 3.6, FDT-Boost induces models that are, on average, $\sim 31\%$ less complex than the ones generated by SAMME-AdaBoost.

Then, in order to analyse whether the claim that, as discussed in Chapter 1, fuzzy models are more apt to deal with noisy data (Klir and Yuan, 1995; Chiang and jen Hsu, 2002), the robustness of both FDT-Boost and the classical SAMME-AdaBoost under an increasing level of noise has been tested. This analysis is of utmost importance as the adaptation mechanism at the basis of AdaBoost is not expected to easily discriminate noise along the learning phase.

The following mechanism of noise injection in the training data, by corrupting only the class labels, has been devised; for each sample in the training set, the class label is perturbed with probability p, by choosing a random label across the remaining ones. In order the evaluate the trends with increasing levels of noise, we varied from 0% to 35% with 5% increments. When noise is injected both FDT-Boosts and Samme-AdaBoost behave similarly in several datasets; we observed a different trend on several other datasets; we reported some of them in Fig. 3.5.

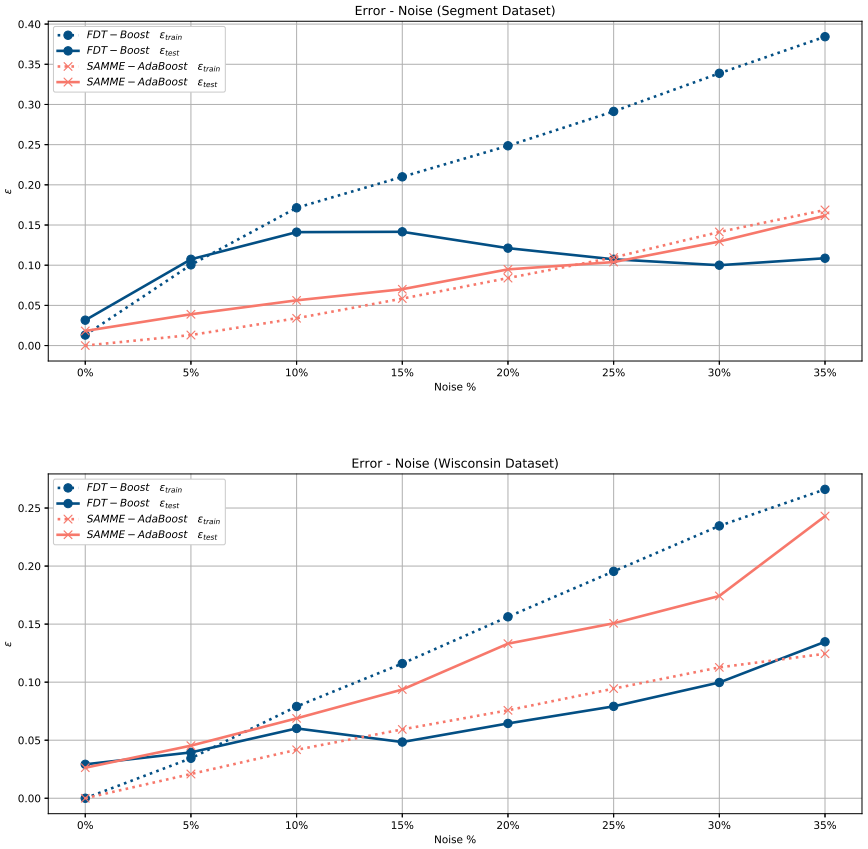


Figure 3.5: *The classification error on the Segment and the Wisconsin datasets, for both FDT-Boost and SAMME-AdaBoost, as a function of the percentage of noisy data. On these datasets, the noisier data are, the better the fuzzy model behaves with respect to its crispy counterpart.*

On Segment (one of the biggest), for example, as the noise ratio increases FDT-Boost becomes growingly more accurate and, when the noise goes beyond 25%, it outperforms SAMME-AdaBoost on the test set (as shown in the upper panel of Fig. 3.5); interestingly, in absence of noise is the crisp model that provides higher performances. On Wisconsin (lower panel in Fig. 3.5) this trend is even more visible: FDT-Boost outperforms the crisp approach as soon as the minimal amount of noise is injected (less than 5%).

3.4.3 Analysis of the convergence

In this section, the results of a thorough analysis of the convergence of FDT-Boost are provided. The curves for the training and test error ratios (ε_{train} and ε_{test}) have been evaluated with respect to a growing number of iterations. There are several reasons that drive this analysis: first and foremost, by observing how the boosting procedure behaves along its progression, insight into the learning procedure might be gained. Furthermore, a quantita-

tive evaluation of the learning process might help in devising the correct parametrization for the model.

Firstly, the average learning curves across all the datasets are evaluated; since the learning curve depends on various factors, as the complexity of the adopted base learner, all the maximum tree depths β in the interval $[1, 6]$ have been tested. The results, reported in Figure 3.6, provide some guidance in the parameter selection.

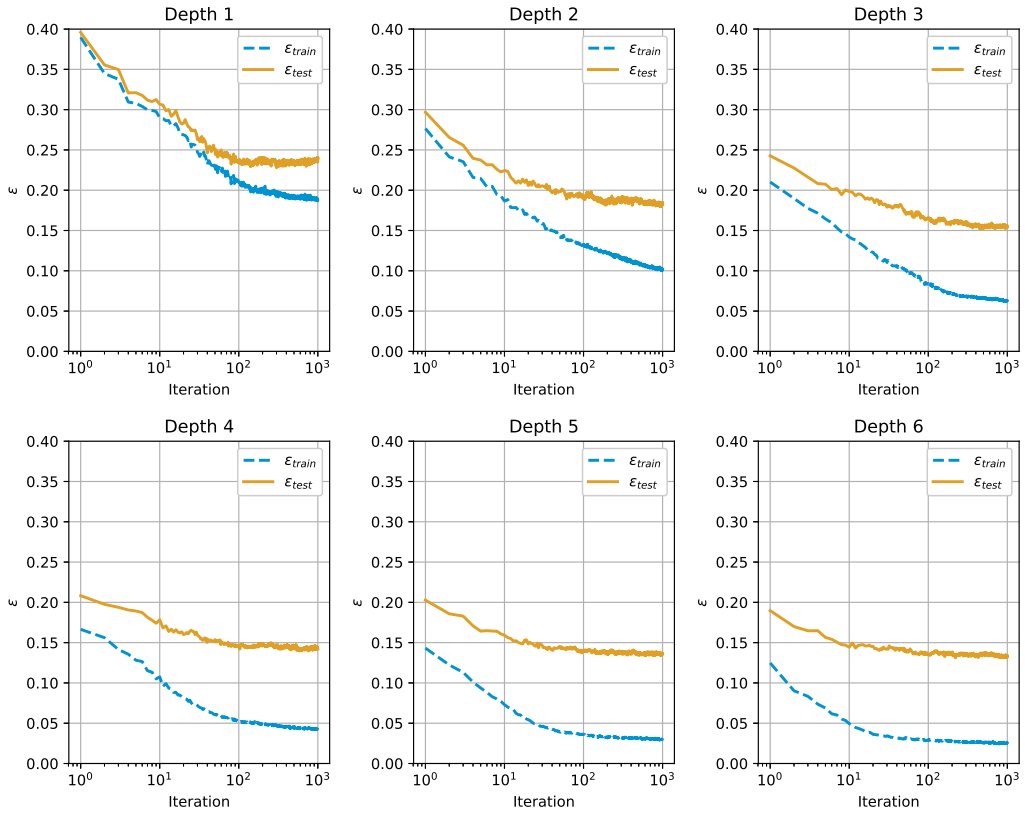


Figure 3.6: *In figure, the average learning curves on all the datasets, for different values of the maximum depth for the base learner (a fuzzy decision tree) are reported.*

The choice of $T = 500$ is adequate enough to reach the accuracy plateau, even for a value of $\beta \geq 2$ (decision stump are thus left out in the following considerations, as they do not seem to provide sufficient accuracy). Concerning β, a choice $\beta = 4$ seems appropriate to reach a minimal error, without letting the model complexity grows considerably. As these are general considerations, the best values might vary in specific cases. As has been often pointed out, AdaBoost defaults its ability to resist overfitting (Schapire et al., 1998) if the base classifier is too complex with respect to the dataset size, or if it is too weak i.e. its margin against a random classifier is too small (this issue will be discussed below) (Schapire, 2013). Indeed, previous works in the literature have also shown that boosting algorithms may be prone to a growing generalisation error under certain conditions (Schapire and Freund, 2012). To better understand the issue, we performed a wide set of experiments

to understand how the selection of fuzzy decision tree as base learner influences this issue, considering that it is a more complex model than its crisp counterparts, i.e. it learns more parameters during its training. As a case study, an analysis of the learning process in the case of two datasets that have been selected as representatives is reported in this section. In order to perform a fair analysis we selected both "Vowel" a multi-class case where FDT-Boost is able, if properly parametrized, to generate an effective model, without overtraining along its learning and "Mammographic" that is likely the most problematic dataset considered here, where the proposed algorithm struggles even in learning examples in the training set.

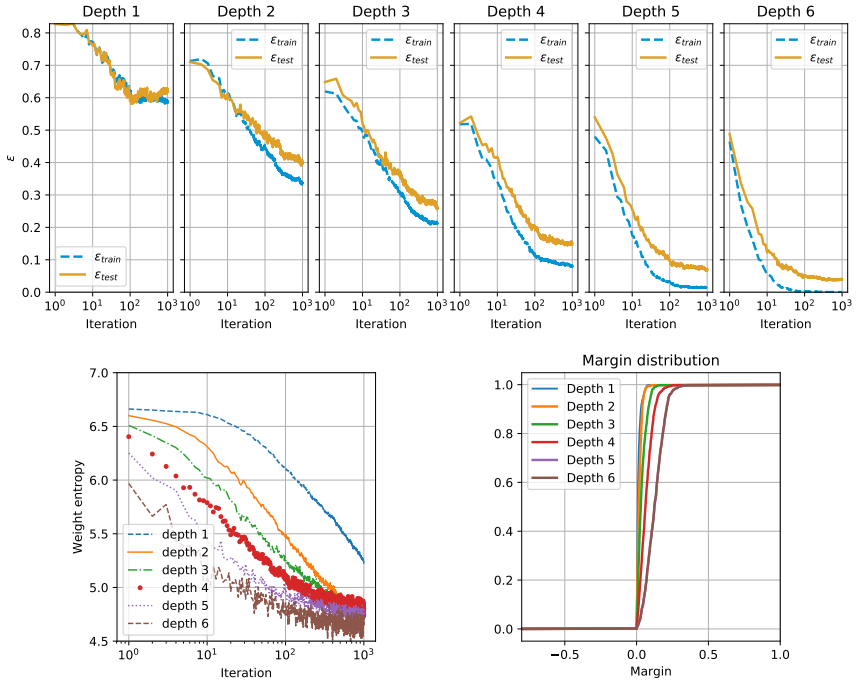
The upper rows of panels (a) and (b) in Figure 3.7 report the error ratio on both the training and the test sets for growing number of iterations (the x-axis are in log scale). The curves show very different convergence patterns between the datasets.

In the case of Vowel (a), no overtraining seems to occur even when the number of iterations approaches 1000; moreover, when the complexity of the base model is raised, FDT-Boost seems to converge faster. Mammographic tells a completely different story: as shown in Figure 3.7 (b), the training set can never be completely learned (as the training error ε_{train} never approaches the x-axis); consequently, the classification performances of the learnt model on the test set are negatively affected as well. As expected, however, the usage of a simpler (i.e. less complex) base model reduces the overfitting issue. These examples clearly depict how a dataset-dependent parametrization could definitively improve the accuracy. The selected maximum depth of 4 provides a global trade-off for all the datasets, but it is a suboptimal choice in both cases.

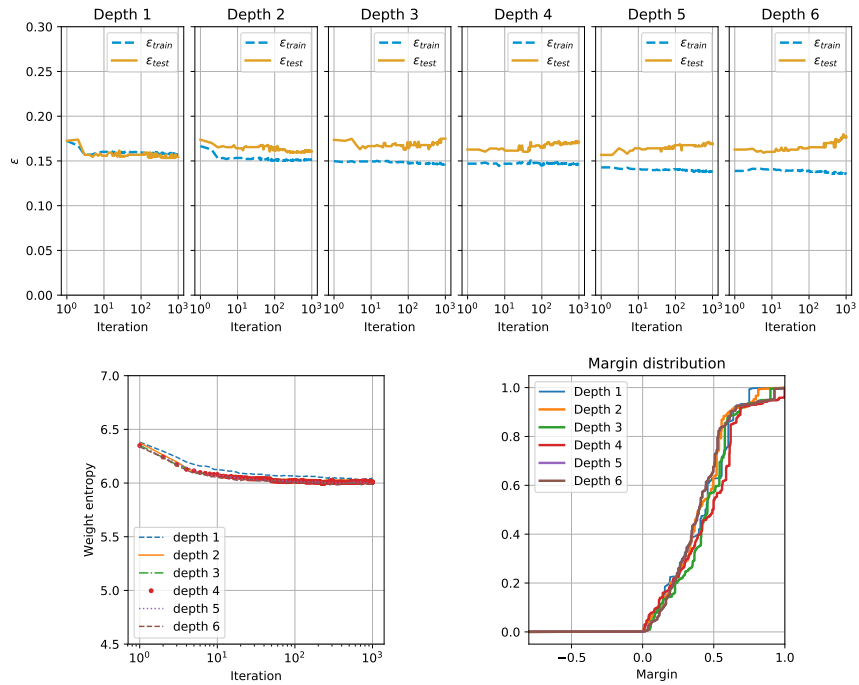
Furthermore, in order the better shed light on the learning behaviour of FDT-Boost, two additional analysis tools have been used: i) the entropy of the weight distribution, and ii) the margin distribution (which nicely fits with the explanation of boosting proposed by Shapire (Schapire et al., 1998)). Both of this indices have been evaluated as the system evolves along successive iterations, for trees bounded to varying maximal depth.

The entropy of the distribution of sample weights is, as portrayed in the lower left panel of Figure 3.7, maximal in the initial phases of the evolution when all the samples are equally probable. As the boosting procedure goes on, entropy decreases, as expected; the rate is proportional to the rate at which the set of incorrectly classified examples dwindles. The lower left plot of panels (a) and (b) of Figure 3.7 reports the entropy of the weight distribution for various maximum tree depths as a function of the iteration number, for both Vowel and Mammographic. Interestingly, the stark decrease shown in Vowel, both with growing number of iterations and with a deeper base model, is not paralleled in Mammographic.

With reference to the second index, the classification margin, it is defined as the difference between the sum of the weights assigned to the correct label and the maximal sum of weights assigned to any single incorrect label. Consequently, the margin is defined in the range $[-1, 1]$ and a sample is classified correctly if and only if its margin is greater than 0. Furthermore, the greater the margin, the more confident the classification; nevertheless, the margin is obviously influenced by the number of classes, as the greater the number of



(a) *Vowel*



(b) *Mammographic*

Figure 3.7: *Results of the convergence analysis for the Vowel (a) and the Mammographic (b) datasets. In the upper row, the classification error on the training and the test sets (represented with dashed and continuous lines, respectively), as a function of the number of boosting rounds are reported; the lower row shows the trend of the weight entropy and the classification margin for different tree depths.*

labels, the lower the margin. In the lower left plots of panel (a) and (b) of Figure 3.7, the cumulative of the margin distribution over the training set is reported. In the ideal case, such as Vowel in panel (a), the deeper the base model the better the margin distribution; furthermore, a greater margin helps to achieve lower generalisation error. In Mammographic, reported in panel (b), the margin remains the same when the base model grows deeper; this phenomenon fits with the stable (or slightly increasing) generalisation error shown in the upper panels of (b).

Lastly, to better inspect how the ensemble grows during the execution of the boosting algorithm, a graphical representation is provided in Figure 3.8; here, we plot a sample of the generated trees, via their Pythagorean representations (Beck et al., 2015). Each node of the tree is drawn as a square, and each split as a right triangle; the father node lays on the hypotenuse, while the child nodes lay on the cathetus. The number of samples in each node is proportional to the areas of the square it is associated with. The colour of each node matches that of the dominant class, and the transparency is related to its margin.

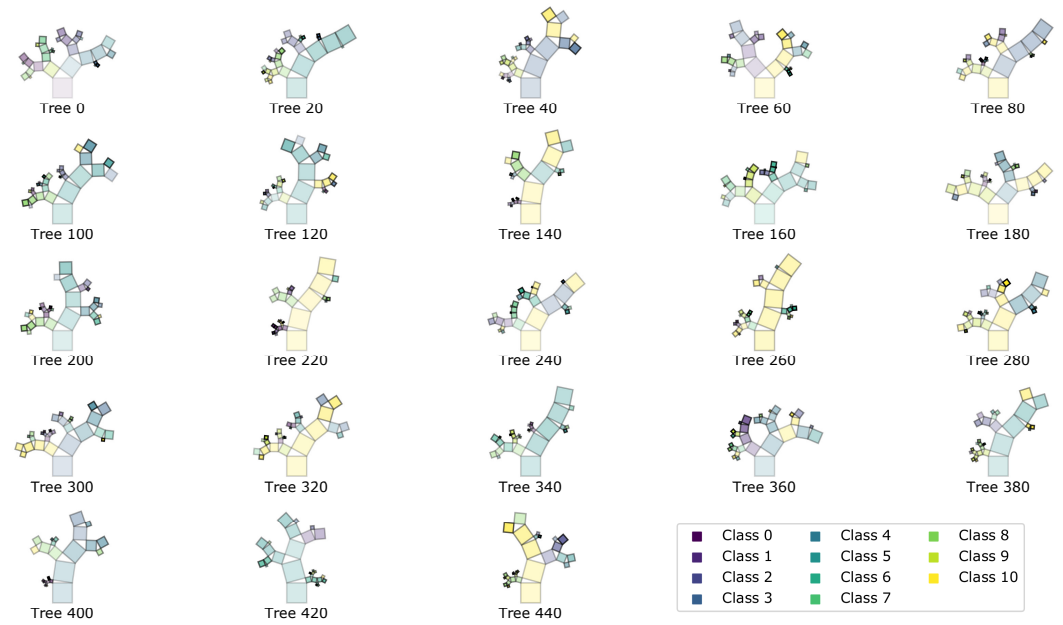
Representing a sample of the ensemble in term of Pythagoras trees may help in understanding how the boosted ensemble has evolved; in the case of Vowel (panel (a) in Figure 3.8), for example, the sampled training set at the time t, $TR^{(t)}$, exhibits a swinging behaviour, being alternately dominated by examples from one of the leading classes. Consequently, trees get unbalanced and more specialised as they are grown over increasingly specialised training sets. In Mammographic (panel (b) in Figure 3.8), the trees struggle to the reach the maximal depth, mostly stopping to the root node after the very first iterations; in other words, they seem to strive to separate the hardest examples.

3.5 Summary

In this chapter we have described FDT-Boost, a novel ensemble-based fuzzy classifier; a comprehensive analysis of the approach is also provided. FDT-Boost is made up of several components: first, a fuzzy discretizer for continuous attributes; second, a depth-constrained binary fuzzy decision tree, which is used as a weak learner. Third, the adopted boosting approach, which follows the SAMME-AdaBoost meta-algorithm. To the best of our knowledge, this is also the most exhaustive investigation of boosting with fuzzy learners.

According to a comprehensive set of experiments, SAMME-AdaBoost ensembles of binary fuzzy decision trees, limited to a maximum depth of 4, are typically able to provide high accuracy value. These results are comparable to those obtained by state of the art fuzzy classifiers, and often even better ones. Indeed, on a set of sixteen benchmarks, FDT-Boost outperformed FURIA, a fuzzy binary decision tree, and a fuzzy multi-way decision tree. Statistical analysis established those differences as significant.

In order to better understand the contribution of a fuzzy base learner, FDT-Boost has been compared with its crisp counterpart; FDT-Boost produces way simpler models, with a reduction of 30% in the number of nodes and leaves, while preserving a comparable



(a) *Vowel*

(b) *Mammographic*

Figure 3.8: *A sample of the tree composing the ensemble trained on the Vowel (a) and Mammographic (b) datasets is shown by means of the Pythagoras tree representation. The two datasets lead to a different population of trees.*

accuracy. Even if AdaBoost is not known for generally providing robustness against noise, a set of noise injection experiments showed that FDT-Boost behaves better than SAMME-AdaBoost for some datasets. Furthermore, the use of a unique fuzzification valid for all the trees in the ensemble, consisting of a triangular strong fuzzy partitioning for each feature,

endows the model with higher interpretability.

Then, the convergence properties of FDT-Boost have been investigated by means of different instruments: margin analysis and entropy of weight distribution; those tools showed that different datasets may lead to very different behaviours. Finally, the model evolution has been also visualized using an effective representation of the generated trees, which helps in understanding the model behaviour.

CHAPTER

FOUR

IMPLICITLY DISTRIBUTED FUZZY RANDOM FOREST

> The intelligence of the creature known as a crowd, is the square root of the number of people in it.
>
> Jingo – Terry Pratchett

In this chapter, we present a novel distributed fuzzy random forest induction algorithm, based on a fuzzy discretizer for continuous attributes. The proposed approach, although shaped on the MapReduce programming model, takes advantage of the implicit distribution of the computation provided by the Apache Spark framework. The chapter is organised as follows: first, Section 4.1 and 4.2 provide a brief description of the state of the art, as well as some background information about fuzzy decision trees and fuzzy discretization. Then, in Section 4.3 fuzzy random forests are thoroughly described. Section 4.4 details the parallel implementation we provide. Section 4.5 gives experimental results for the assessment of the proposed approach. Finally, Section 4.6 summarises the results obtained.

4.1 From Random Forests to Big Data

The ongoing quest of artificial intelligence is that of unearthing complex relationships in the data. In other words, the learnt models should be able to generate accurate predictions while also allowing to extract knowledge in an intelligible way. Pursuing this double goal, research in machine learning has given rise to an extensive body of works in a myriad of directions. As briefly discussed in Section 3.1, a particularly popular solution is represented by *decision trees* (Quinlan, 1986), which are conceptually very simple models with a high level of interpretability. Their founding idea is to define a proper partition of the feature space, with every single partition containing a homogeneous subset of the learning dataset, and corresponding to a classification label. The model is constructed according to a hierarchical approach, and several algorithms have been proposed to this aim (Quinlan, 1986).

The result of the hierarchical partitioning is a tree that provides also an efficient way to assign an example pattern to the target partition, represented by a tree leaf. Intermediate nodes correspond to intermediate partitions.

As also discussed in Chapter 3, as single classifier might often be too simple of specific, *ensemble methods* have been proposed; indeed, they have shown to offer improved performance with respect to base learners (Rokach, 2010). Particularly, this seems to be due to the fact that a committee of classifiers drives down the prediction variance, yet maintaining the same bias (Opitz and Maclin, 1999). In fact, as (deep) decision trees are generally believed to be high-variance low-bias models, they are ideally suited for ensembling. Several ensemble methods have been proposed: boosting was the topic of Chapter 3; random forests are the topic of this chapter. Unlike boosting, which evolves a committee of weak learners over time, random forests (Breiman, 2001) make use of *bagging* (i.e. each single decision tree is learnt from a different training set, sampled with replacement from the complete dataset) and *random attribute selection* at the node level. Random forests substantially modify bagging, in that the trees they grow are de-correlated. In fact, the advent of random forest ushered in a new era of machine learning. Nowadays, random forests are one of the most popular classification and regression algorithms and have found application in a plethora of scientific fields (Díaz-Uriarte and Alvarez de Andrés, 2006; Rodriguez-Galiano et al., 2012; Svetnik et al., 2003).

Recalling from Chapter 3, fuzzy decision trees (FDTs) have been proposed to incorporate the notion of fuzziness (Janikow, 1998), associating each node of the FDT to a fuzzy set, rather than a classical set. In general, fuzzy approaches help in dealing with uncertainties (Zimmermann, 2010), and fuzzy classification of an instance leads to multiple labels with different confidence values. In an FDT, each instance can activate different branches and thus reach multiple leaves. A thorough description of FDTs has been given in Section 3.3.2. Random forest with fuzzy decision trees have been proposed before, showing good accuracy classification, comparable to that of the best classifiers when tested with conventional data sets, and high robustness to noise (Bonissone et al., 2010; De Matteis et al., 2015). The first comprehensive analysis of a fuzzy random forest (FRF) classifier was given in (Bonissone et al., 2010); the authors explored different combination methods to obtain the final decision and proposed a thorough comparison with other approaches. Later, in (De Matteis et al., 2015), a novel approach to the problem of fuzzy partitioning in FRFs generation was proposed; the fuzzy partitions were created during the generation of the tree by adopting an approach which iteratively zooms in on specific intervals of the universe. Lastly, an FRF composed of C-fuzzy decision tree was also proposed (Gadomer and Sosnowski, 2016).

As the era of Big Data swept in, it brought a wealth of data that required computer scientists to rethink the way in which they solved their problems; indeed, Big data refers to datasets so big and complex that require specialised data processing approaches. During the last decades, a plethora of approaches, exploiting different solutions, have been proposed to tackle big data; cluster computing is one of the most explored areas. Amongst the most popular cluster computing frameworks, it is worth recalling Apache Hadoop (White,

2009) and Apache Spark (Zaharia et al., 2010). The former implements the MapReduce programming model introduced to simplify the distribution of a computational flow across clusters of computing machines (Dean and Ghemawat, 2008). The latter implements the concept of in-memory cluster computing, thereby speeding up computation, notably in executing iterative or interactive algorithms, and supporting integrated processing of large datasets (Zaharia et al., 2010). Plenty of machine learning approaches have been adapted to take advantage of these new paradigms; SVMs (Lin et al., 2014) and decision trees (Meng et al., 2016) are among them. It is also believed that fuzzy models are particularly well suited for handling the variety and veracity that define Big Data. Recently, several fuzzy techniques have been proposed for dealing with huge amount of data (Elkano et al., 2017; Fernández et al., 2017; Ferranti et al., 2017; López et al., 2015; Márquez et al., 2017; del Río et al., 2015a; Segatori et al., 2017a, 2018). A much wider and comprehensive introduction to the problems connected with the analysis of big data is provided later in the book, in Section 5.1.

4.2 Preliminaries

In the following, the necessary notation is recalled; TR is a training set of N training examples, $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i$ and y_i are respectively the feature vector and the label for the i-th example, a supervised learning algorithm looks for a function $h : X \rightarrow Y$ from the input space to the output space. In the case of classification, y_i can assume only a limited number of values $\{C_1, ..., C_m\}$.

Here, it is assumed that: i) each numerical attribute X_f is defined on a universe $U_f \subset \mathbb{R}$; ii) given a fuzzy set v_f, the degree to which an element $x_f \in U_f$ belongs to the set v_f is defined by the membership function $\mu^v(x_f)$. Furthermore, let us introduce the notion of linguistic variable V, an attribute defined over a domain of linguistic values, which are labels for fuzzy subsets.

Amongst the most common supervised learning techniques, decision tree learning works by recursively partitioning the sample space according to a data-driven approach, representing the partition as a tree (Janikow, 1998). In other words, a decision tree learns a piece-wise continuous function in the feature space. Although discrete attributes are handled "out-of-the-box", continuous ones must be discretized, either prior to or during the tree induction. Unlike the classical ID3 algorithm (Quinlan, 1986), most of the tree induction algorithms proposed so far, such as CART (Breiman, 1993), do not require prior partitioning. However, it has been shown that the use of prior multi-interval discretization, combined with properly shaped heuristics, can bring several advantages (Fayyad and Irani, 1993a). As in Boolean decision trees, at each internal node, one attribute is chosen and tested, and its child nodes are organised according to the test outcome. Then, each leaf node holds one or more classes.

The tree generation proceeds recursively from the root node downwards. At each node, an attribute is selected according to a proper criterion, and a split is performed. The selec-

tion criterion in FDT is typically based on metrics like fuzzy information gain (Zeinalkhani and Eftekhari, 2014), fuzzy Gini index (Chandra and Varghese, 2008), maximum classification importance of attribute contributing to its consequent (Wang et al., 2001), minimal ambiguity of a possibility distribution (Yuan and Shaw, 1995), and fuzzy Kolmogorov-Smirnov discrimination quality measure (Boyen and Wehenkel, 1999). In our work, we used the fuzzy information gain as actual splitting metrics. A formal definition will be given in Section 4.3.

The tree induction recursively proceeds in partitioning nodes, until one of the stopping conditions, discussed in Section 3.2, is met.

Regarding the inference procedure, in classical decision trees each unlabelled instance $\hat{\mathbf{x}}$ activates a unique path, and as such, is assigned a unique class $C_m \in C$. Conversely, in FDTs, $\hat{\mathbf{x}}$ can activate multiple paths in the tree, because at each node it may correspond to multiple fuzzy subsets. We recall that each leaf is reached with a *matching degree*; given a node $\mathcal{N}$, the matching degree $md^{\mathcal{N}}(\hat{\mathbf{x}})$ of $\hat{\mathbf{x}}$ with $\mathcal{N}$ is defined as in Equation 3.1.

Here TN is a T-norm, $\mu^{\mathcal{N}}(\hat{x}_f)$ is the membership degree of $\hat{x}_f$ with the node N considering X_f as splitting attribute and $md^{\mathcal{PN}}(\hat{\mathbf{x}})$ is the matching degree of $\hat{\mathbf{x}}$ with the parent node of $\mathcal{N}$, $\mathcal{PN}$.

Then, for each activated leaf node, an *association degree* is computed as $AD_m^{\mathcal{LN}}(\hat{\mathbf{x}}) = md^{\mathcal{LN}}(\hat{\mathbf{x}}) \cdot w_m^{\mathcal{LN}}$ being $md^{\mathcal{LN}}(\hat{\mathbf{x}})$ the matching degree of $\hat{\mathbf{x}}$ with the leaf node $\mathcal{LN}$ and $w_m^{\mathcal{LN}}$ the class weight associated with class C_m at leaf node $\mathcal{LN}$. Class weights are used as they have been shown to provide higher performances (Ishibuchi et al., 2005).

The class weight for the class C_m at the leaf node $\mathcal{LN}$ is $w_m^{\mathcal{LN}} = \frac{|G_{C_m}|}{|G|}$, being G_{C_m} the set of training instances in G having class label C_m. Then the *association degrees* are properly combined to produce an output class label. More precisely, a weighted voting method is used, where the class weights for each leaf, weighted by their respective *association degrees* are aggregated into an unique vector of class votes.

4.3 Proposed Fuzzy Random Forest

In this Section, the proposed FRF algorithm is described. The approach is structured in two main steps: i) a fuzzy discretization of continuous variables is performed, then ii) a distributed FRF is induced.

4.3.1 Distributed discretization

The fuzzy discretization approach used here follows a modified Fuzzy Partitioning based on Fuzzy Entropy (FPFE) procedure. This has been adapted from (Segatori et al., 2017b). The description given here follows the one given in Section 3.3.1; as such only the necessary details are recalled here. Please refer to Section 3.3.1 and to (Segatori et al., 2017b) for further information.

The chief aim of this section is that of extending the concepts described in Section 3.3.1 in the context of big data, using a distributed implementation; the topic of discretization in the context of big data is thoroughly discussed in (Ramírez-Gallego et al., 2016).

The discretization algorithm recursively partitions continuous attributes, defining strong triangular fuzzy partitions, until a stopping criterion is met; the stopping criterion is a fuzzy adaptation of the minimum description length principle (MDLP) (Fayyad and Irani, 1993a). Unlike the discretizer examined in Section 3.3.1, the one described here works in a distributed fashion. The partitioning, done independently of the tree induction step, considers each feature separately. The proposed approach brings several advantages: first it can lead to feature selection if no splits are selected for a feature; furthermore, it produces a strong and highly interpretable fuzzy partition.

The notation will be briefly recalled in the following. Let us focus on the generic feature X_f; let $x_{i,f}$ be the value the f-feature in the i-th sample of the training set, and let us assume values sorted in ascending order. Let I_F be an interval on the universe of the feature X_f.

Consider a fuzzy partition $P_{I_f} = \left\{B_{f,1}, ..., B_{f,|P_{I_f}|}\right\}$ over I_f; let $B_{f,i}$ be the i-th fuzzy set, and let $|P_{I_f}|$ be the number of fuzzy sets defined in the partition P_{I_f} over the interval I_f.

Let $S_{f,i}$ be the crisp subset of points of S_f contained in the support of $B_{f,j}$ and let $|B_{f,j}|$ be the fuzzy cardinality of the fuzzy set $B_{f,j}$, defined in Equation 3.4.

Let also $FEnt(B_{f,j})$ be the fuzzy entropy for a fuzzy set $B_{f,j}$, as in Equation 3.5, and let $WFEnt(P_{I_f}; I_f)$ be the weighted fuzzy entropy of a fuzzy partition P_{I_f} over an interval I_f (Equation 3.6).

Furthermore, let us recall the definition of information gain $FGain$, given by a partition P_f over an interval I_f for an attribute X_f:

$$FGain(P_f; I_G) = FEnt(G) - WFEnt(P_f; I_G) \tag{4.1}$$

Starting from an initial interval, the discretization approach proceeds by evaluating all the candidate cut-points for an interval, selecting the cut point with maximum fuzzy information gain, if the stopping criterion is not met. In the context of big data, to diminish the number of evaluations, the candidate cut-points are reduced by computing an equi-frequency binning of the universe for each continuous attribute.

Here, the stopping criterion is recalled:

$$FGain(x^1_{i,f}; I_f) < \frac{\log_2(|S_f - 1|)}{|S_f|} + \frac{\Delta(x^1_{i,f}; I_f)}{|S_f|} \tag{4.2}$$

being $\Delta(x^1_{i,f}; I_f)$ given in Equation 3.9.

The MDLP, as well as its fuzzy twin, can induce a feature selection. If no splits for a given variable is selected, the feature will be removed.

The distributed implementation of the discretization phase is described in Section 4.4; the distribution of the workload across a cluster of computing nodes operates both i) the selection of candidate partitions to be analysed, and ii) the recursive discretization step.

4.3.2 Fuzzy Random Forest

The construction of the fuzzy random forest (FRF) is carried out building a set of NT independent decision trees, grown out to their maximum depth size β (defined as a parameter of the model), without post-pruning.

Let $|TR|$ be the size of the training set; following the classical definition of the algorithm, each tree is trained on a bagged subset of the training set (Breiman, 2001), i.e. for each tree, $|TR|$ sample are randomly chosen with replacement from the set TR. Thus, on average, a bagged set contains $\sim 63\%$ of the training set (Domingos, 1997). Moreover, for each tree, the remaining samples are aggregated in the so-called *out-of-bag (OOB) sample*.

As such, NT decision trees are grown, each one on its bagged training set. Furthermore, in order to decrease the correlation between trees, each node uses a *random attribute selection*: while the tree is growing, a random subset of $m < f$ attributes is considered at each node (Breiman, 2001). Several feature selection strategies have been proposed in the literature, the most common ones being $sqrt$, $\log_2$ and $onethird$. Further details about the implemented decision tree algorithm can be found in (Segatori et al., 2017b).

The pseudo-code for a naive FRF learning algorithm is reported in Figure 4.1. The implementation defined here works in a parallel fashion, where all the trees are trained concurrently, with parallelisation at the node level; the implementation details are provided in Section 4.4.

```
1:  function FRFLearn(TR, NT)
2:      FuzzyForest ← ∅
3:      for i ← 1 to NT do
4:          TR^(i) ← Bagging(TR,|TR|)
5:                                          ▷ TR is bagged to obtain TR^(i)
6:          FDT^(i) ← FDTLearning(TR^(i))
7:                                  ▷ i-th fuzzy decision tree generated from TR^(i)
8:          FuzzyForest ← FuzzyForest ∪ FDT^(i)
9:      end for
10:     return FuzzyForest
11: end function
```

Figure 4.1: *Representation of a simple algorithm that induces a Fuzzy Random Forest. It requires the complete training set TR and the number of trees NT as inputs. Furthermore, the algorithm assumes that the continuous variables have been previously discretized.*

Combination of Results from Single Trees

Each leaf node $\mathcal{LN}$ of each tree casts a vote for each class m, weighted by the *association degree* previously defined, $AD_m^{LN}(\hat{\mathbf{x}})$. As such, each decision tree in the ensemble of classi-

fiers produces a set of aggregated votes, called confidence values; the *confidence value* C_m for the class m, is obtained by summation of the leaf votes grouped by class m.

Accordingly, the confidence value of the t-th tree for the class m is defined as:

$$CV_m^t = \sum_{\mathcal{LN} \in t} AD_m^{\mathcal{LN},t} \tag{4.3}$$

In other words, each tree produces a list of class-vote pairs, thus assigning a vote to each class. These votes are then aggregated by class for the whole forest and weighted by a tree weight. Our implementation provides the possibility of weighting the t tree by using the OOB accuracy for that tree, $1 - OOBError_t$. Then, the forest votes for the class having the max value.

4.4 Distributed Fuzzy Random Forests

The approach described in the preceding sections has been implemented on the Apache Spark framework. In the following, it is described according to the MapReduce paradigm (Dean and Ghemawat, 2008). First, we introduce the necessary notation: let V be the number of chunks (partitions) in which the training set is split, and let Q be the number of executors (or computing units) available in the cluster. It must be recalled that, in the Map-Reduce paradigm used here, each chunk feeds only one Map Task.

The implementation of the fuzzy partitioning approach follows the one described in (Segatori et al., 2017b); to reduce the complexity of the discretization step described in Section 4.3.1, the number of candidate partitions to be analysed is lowered. For each chunk of the dataset, the values for each feature are sorted and the domain is divided into L equi-frequency bins. The choice of the right L value involves a trade-off between the coarseness of the approximation and the computational complexity.

The generated list of bin boundaries is then aggregated and, for each pair of consecutive bin boundaries, a new bin is generated; later, the distribution of the classes among the instances belonging to such a bin is computed. Then a candidate fuzzy partition is generated for each bin boundary and the class distribution is used for computing fuzzy entropy and fuzzy information gain at each step of the algorithm. In the following each generic bin l for the feature f will be referred to as $b_{f,l}$, and it will be represented by its central value $\bar{b}_{f,l}$. So, Equation 3.4 can be rewritten accordingly:

$$|B_{f,j}| = \sum_{l=1}^{L_{f,j}} \mu_{B_{f,j}}(\bar{b}_{i,f}) \tag{4.4}$$

being $\mu_{B_{f,j}}(\bar{b}_{i,f})$ the membership degree of the central value of the bin $\bar{b}_{f,l}$ to the fuzzy set $B_{f,j}$. Here, $L_{f,j}$ is the number of bins in $S_{f,j}$, the support of the j-th fuzzy set for the f-th feature. The fuzzy entropy of the set $B_{f,j}$ is computed using Equation 3.5, where $|B_{f,j,C_m}|$ is extracted considering the distribution of class C_m in each bin contained in $S_{f,j}$.

The discretization can be described as two consecutive Map-Reduce steps: in the first step the training set is scanned, yielding at most $\Omega = V \cdot (L+1)$ bin boundaries. Here, L can be selected to be a fixed fraction of the dataset size, let say 0.1%. Each Map-Task loads the v-th chunk of the dataset and, for each continuous attribute X_f, computes bin boundaries of equi-frequency bins. Being $BB_{v,f} = \{b_{v,f}^{(1)}, ..., b_{v,f}^{(L)}\}$ the sorted list of bin boundaries for the f-th feature in the v-th chunk, each Map-Task outputs $\langle key = f, value = BB_{v,f} \rangle$. Then, the Reduce-Task gets a list of $BB_{v,f}$ for all the v, and emits $\langle key = f, value = BB_f \rangle$, where $BB_f = \{b_f^{(1)}, ..., b_f^{(\Omega)}\}$ is the sorted list of bin boundaries for the f-th attribute, thus aggregating all the bin boundaries for each feature. The Map-Task has a space complexity of $O(\lceil \frac{V}{Q} \rceil \cdot N/V)$ and a time complexity of $O(\lceil \frac{V}{Q} \rceil \cdot (F \cdot N \cdot (log(N/V))/V))$, while for the Reduce-Task the space and time complexities are $O(F \cdot \Omega/Q)$ and $O(F \cdot (\Omega \cdot log(\Omega))/Q)$, respectively. The pseudo code for the first Map-Reduce step is shown in Figure 4.2.

Require: *TR*, split into V chunks
1: **Function** MAP-TASK(TR_v, γ)
2: **for each** feature X_f in **X do**
3: sort X_f values
4: $BB_{v,f} \leftarrow$ bin boundaries of equi-frequency bins, according to γ
5: **output** $\langle key = f, value = BB_{v,f} \rangle$
6: **end for**
7: **end Function**
8: **Function** REDUCE-TASK(f, $List(BB_{v,f})$)
9: $BB_f \leftarrow$ sort elements of $List(BB_{v,f})$
10: **output** $\langle key = f, value = BB_f \rangle$
11: **end Function**

Figure 4.2: *Pseudo code of the first MapReduce Task for the discretization.*

The second Map-Reduce step generates the distributed fuzzy sets. The Map-Task is fed with the v-th chunk of the dataset, and for each feature X_f, a vector $W_{v,f}$ of size $\Omega - 1$ is initialized. Each element in the vector, let say $W_{v,f}^{(i)}$, represents the corresponding bin, $(b_f^i, b_f^{r+1}]$, and is itself a vector of size M containing, for each class, the number of instances for that class in the i-th bin for the f-th feature. The vector $W_{v,f}$ is updated for each instance in the chunk, and a key value pair is emitted $\langle key = f, value = W_{v,f} \rangle$. Then a Reduce-Task for each attribute is run, that produces a vector W_f by element-wise addition of all the V vector $W_{v,f}$; later on the Reduce-Task applies the fuzzy partitioning described before, thus emitting $\langle key = f, value = P_f \rangle$, being P_f the strong fuzzy partitioning for the f-th attribute. For the Map task space and time complexities are $O(\lceil \frac{V}{Q} \rceil \cdot N/V)$ and $O(\lceil \frac{V}{Q} \rceil \cdot (N \cdot log(\Omega)/V))$ respectively, while, for the Reduce phase we space and time complexities are $O(F \cdot (\Omega - 1)/Q)$ and $O(F \cdot (2 \cdot max(T_f) - 3) \cdot (\Omega - 1)^2)/Q)$.

The DFRF learning generalises the procedure described in (Segatori et al., 2017b), distributing the computation of the best split for each node across the CUs. The following

Require: TR split into V chunks, and BB matrix whose f-th row is BB_f
1: **Function** MAP-TASK(TR_v, BB, M)
2: $W_{v,f} \leftarrow$ initialise F arrays according to BB and M
3: **for each** instance $\mathbf{x}_n, y_n$ in TR_v **do**
4: **for each** attribute X_f in $\mathbf{X}$ **do**
5: $W^{(r)}_{v,f} \leftarrow$ update number of instances of y_n
6: **end for**
7: **end for**
8: **for each** attribute X_f in $\mathbf{X}$ **do**
9: **output** $\langle key = f, value = WW_{v,f} \rangle$
10: **end for**
11: **end Function**
12: **Function** REDUCE-TASK(f, $List(WW_{v,f})$, BB_f)
13: $WW_f \leftarrow$ addition of $List(WW_{v,f})$
14: $P_f \leftarrow$ FuzzyPartitioning(W_f, B_f)
15: **output** $\langle key = f, value = P_f \rangle$
16: **end Function**

Figure 4.3: *Pseudo code of the second MapReduce Task for the discretization.*

Map-Reduce steps are executed iteratively. Let R be the set of nodes to be split. In the first iteration, R is initialised with NT elements, the root nodes for each tree in the ensemble. At each generic iteration h, a set of Y nodes is retrieved from R; we refer to that set as R_h. The size of the set is determined as min(size(R), $maxY$). $maxY$ is a parameter of the model, depending on the amount of memory available on the cluster, as well as on the number of categorical values and fuzzy sets defined by the fuzzy partitioning. The v-th map task loads the v-th chunk of the data and, for each node NT_y in R_h, a vector $D_{v,y}$ of $|D| = \sum_{\forall f \in F_y} T_f$ instances is initialised. Here F_y represents the subset of features selected for the node y. Even if the feature selection strategy is fixed, and is the same for all the nodes, the set F_y depends on the particular node, and the value of $|D|$ changes from node to node. Nevertheless, for a given feature selection strategy, $|D|$ is bounded by $D_{min} \leq |D| \leq D_{max}$ being $D_{min} = \min_{F_y} \sum_{\forall f \in F_y} T_f$ and $D_{max} = \max_{F_y} \sum_{\forall f \in F_y} T_f$.

Then, for each attribute of each instance in the chunk, the Map-Task, updates all the $D_{v,y}$ vectors. Finally, each Map-Task outputs the key values pairs $\langle key = y, value = D_{v,y} \rangle$, being y the index of the y-th node in R_h. Then the Reduce-Task gets a list of vectors $D_{v,y}$ all pertaining to the same node, and creates a vector D_y by performing an element-wise addiction of all V vectors in the list. As a result, D_y stores the cardinality for each attribute value from the root to NT_y. Finally, the Reduce-Task produces the child nodes using a binary splitting method, as described before. Children are used to update the tree and are inserted in R.

Require: *TR* split into V chunks
1: **Function** MAP-TASK(TR_v,R_h)
2: **for each** node NT_y in R_h **do**
3: $D_{v,y} \leftarrow$ create vector of size $|D|$
4: **for each** instance $\mathbf{x}_n, y_n$ in TR_v **do**
5: $D_{v,y} \leftarrow$ update statistics using $x_{f,n}$
6: **end for**
7: **output** $\langle key = y, value = D_{v,f} \rangle$
8: **end for**
9: **end Function**
10: **Function** REDUCE-TASK(f, $List(D_{v,y})$)
11: $D_y \leftarrow$ addition of $List(D_{v,y})$
12: $children \leftarrow$ SPLITTING(NT_y, D_y)
13: **output** $\langle key = y, value = children \rangle$
14: **end Function**

Figure 4.4: *Pseudo code of the MapReduce Tasks for the distributed node splitting.*

The following procedure is repeated until R is empty. Regarding time and space complexity of the Map phase, we have $O(\lceil \frac{V}{Q} \rceil \cdot N/V)$ and $O(\cdot \lceil \frac{V}{Q} \rceil \cdot (N \cdot Y \log(D_{max})/V))$ for space and time complexity, respectively. The Reduce phase has a space complexity $O(Y/Q)$ and a time complexity $O(Y \cdot |allSplits|/Q)$. The value of $|allSplits| \leq D_{max}$ depends on the feature subset strategy selected. The complexity of the forest construction algorithm can be approximated as: $O(NT \cdot H \cdot (\lceil \frac{V}{Q} \rceil \cdot (N \log(D_{max})/V)))$.

4.5 Experimental Results

In this section, the results obtained in the experimental characterisation of the proposed algorithm are presented. A first subsection is devoted to evaluating the performance of the approach against three other state-of-the-art fuzzy classifiers: DFDT, a Distributed Fuzzy Decision Tree (Segatori et al., 2017b), DFAC-FFP, an associative fuzzy classifier for big data (Segatori et al., 2017c), and ChiFRBCS-BigData (del Río et al., 2015b). Then, the scalability of the algorithm is evaluated using a growing number of computing units (CUs). In all the experimental tests performed to evaluate the proposed approach, a selection of seven big datasets available from the *UCI Machine Learning Repository*[1] have been used. The different datasets are characterised by different cardinalities (ranging from 1 up to 11 million), numbers of attributes, number of classes, and overall sizes, as reported in Table 4.1.

A five-fold cross-validation has been performed on each dataset. The algorithm has been executed on Apache Spark 2.2.0 on a small computer cluster, using up to 7 machines,

[1] https://archive.ics.uci.edu/ml/datasets.html

Table 4.1: *(Big) Datasets used in the experimental validation of DFRF*

Dataset	# Instances	# Attributes	# Classes	Size
COVTYPE	581,012	54	7	75.2 MB
ECO_E	4,178,504	16	10	534 MB
ECO_CO	4,178,504	16	21	534 MB
EM_E	4,178,504	16	10	532.5 MB
Higgs	11,000,000	28	2	8.04 GB
KDDcup 1999 5	4,898,431	41	5	480 MB
KDDcup 1999 23	4,898,431	41	23	484 MB
Poker-Hand	1,025,010	10	10	24.5 MB
Susy	5,000,000	18	2	2.4 GB

one master node and up to 6 workers. Each machine has 4 vCPU, 8GB of RAM and 160 GB Hard Drive. All the machines run Ubuntu 14.04. The training set is stored on the Hadoop Distributed File System (HDFS), with a data node running on each worker machine. In all the experiments the standalone cluster manager provided by Apache Spark has been used.

4.5.1 Comparison with other state-of-the-art fuzzy classifiers for big data

In this section, the performances of DFRF, in terms of accuracy and execution time, are analysed and compared to those of DFDT (proposed in (Segatori et al., 2017b)), DFAC-FFP (Segatori et al., 2017c), and ChiFRBCS-BigData (del Río et al., 2015b).

Table 4.2 lists the parameters used in the experiments. The DFDT, DFAC-FFP and ChiFRBCS-BigData are parametrized according to indications present in the literature (Segatori et al., 2017b,c).

Table 4.2: *Values of the parameters used in the experiments for DFRF*

Parameter	Description	Value
λ	Min instances per node	$10^{-4} \cdot N$
ϕ	Min support Fuzzy Set	$0.02 \cdot N$
γ	Fraction bin discretization	0.1%
Impurity	Impurity measure	entropy
β	Maximum Tree Depth	15
featSel	Feature selection strategy	*sqrt*
maxBins	Maximum number of bins	32
T	T-norm used	product
NT	Number of trees	50
V	Number of partitions	$2 \cdot Q$

The number NT of trees composing the ensemble has been fixed to 50 in the current experiment. The feature selection strategy we used is $sqrt$, where the number of features randomly selected in each node is $\sqrt{F'}$, being F' is the number of remaining features after the discretization step. As suggested before in (Segatori et al., 2017b)), the minimum number λ of instances belonging to each leaf is at last $10^{-4} \cdot N$, and the support of each fuzzy set should contain at least $\phi = 0.02 \cdot N$ instances. The maximum tree depth β has been fixed to 15.

For each dataset and for each algorithm, a five-fold cross-validation is performed; Table 4.3 lists, for each dataset and for each algorithm, the average values and the standard deviation of the accuracy, measured both on the train (ACC_{Tr}) and test (ACC_{Ts}) set; furthermore, the highest accuracy for each dataset is reported in bold.

According to the results reported in the table, DFRF always outperforms DFDT, DFAC-FFP and ChiFRBCS-BigData. Furthermore, DFRF shows little or no sign of overtraining, the only notable exception being POKER dataset. This behaviour might be related to the structure of the dataset, being POKER composed of only categorical features with a high number of possible different categories. Indeed, both DFDT ChiFRBCS-BigData suffer from overtraining in the same dataset.

In order to perform a statistical comparison of the approaches, a distribution consisting of the average accuracy value on the test set for all the datasets is generated for DFRF, DFDT, DFAC-FFP and ChiFRBCS-BigData. Then, the Wilcoxon test is applied to uncover statistical difference among the distributions; the results of the Wilcoxon test, in terms of ranks and p-values are reported in Table 4.4.

In order to account for multiple comparisons, the Bonferroni correction has been applied; here, considering 6 comparisons (even in we are only interested in three of them), the corrected $\alpha' = \frac{\alpha}{m}$, being m the number of comparisons, is $\alpha' = 0.008334$. The findings are reported in Table 4.5: DFRF is shown to statistically outperform all the other approaches considered here.

The execution times for the algorithm analysed here are reported in Table 4.6. The run times have been measured on a cluster with one master node equipped with 4 CUs and 8GB of RAM and 6 workers, up to a total amount of 24 CUs and 48 GB RAM. Nevertheless, the amount of RAM available to Spark is reduced to 6GB per worker; the total amount of executor memory is 36GB.

As expected, DFRF is much slower than DFDT; furthermore, it is, on average, slightly slower than ChiFRBCS, while being much faster than DFAC-FFP.

The ratio between the FDRF and the DFDT execution times is on average 34.241. The departure from the ideal value $NT = 50$, is due both to the discretization time (included in the overall values shown in Table 4.6) and to the speedup caused by the feature selection strategy.

Table 4.3: *Experimental results*

	FDRF		DFDT		DFAC-FFP		ChiFRBCS	
Dataset	ACC_{Tr}	ACC_{Ts}	ACC_{Tr}	ACC_{Ts}	ACC_{Tr}	ACC_{Ts}	ACC_{Tr}	ACC_{Ts}
COV	82.136 ± 0.313	**81.869** ± 0.259	81.923 ± 0.235	81.592 ± 0.256	77.227 ± 0.043	77.190 ± 0.212	82.773 ± 2.547	81.174 ± 4.027
ECO_E	94.981 ± 0.018	**94.695** ± 0.030	93.581 ± 0.029	93.578 ± 0.050	73.672 ± 0.008	73, 698 ± 0.037	54.454 ± 7.813	54.485 ± 7.819
ECO_CO	95.970 ± 0.036	**95.956** ± 0.056	95.590 ± 0.161	95.561 ± 0.158	62.315 ± 0.120	62.305 ± 0.118	73.639 ± 6.415	73.604 ± 6.406
EM_E	94.980 ± 0.060	**94.961** ± 0.063	94.961 ± 0.111	94.949 ± 0.109	86.948 ± 0.248	86.944 ± 0.230	67.285 ± 8.962	67.277 ± 8.952
HIGGS	72.601 ± 0.022	**72.426** ± 0.023	72.343 ± 0.020	72.159 ± 0.025	66.019 ± 0.062	66.005 ± 0.078	55.933 ± 0.080	55.897 ± 0.119
KDD_5	99.980 ± 0.001	**99.980** ± 0.001	99.973 ± 0.002	99.972 ± 0.004	99.941 ± 0.031	99.939 ± 0.032	96.395 ± 0.043	96.302 ± 0.044
KDD_23	99.957 ± 0.001	**99.956** ± 0.002	99.942 ± 0.001	99.941 ± 0.001	99.887 ± 0.152	99.886 ± 0.150	99.988 ± 0.001	99.61 ± 0.01
POKER	85.068 ± 0.358	**79.205** ± 0.459	66.580 ± 0.495	62.818 ± 0.536	50.654 ± 0.031	50.630 ± 0.042	99.711 ± 0.002	50.178 ± 0.005
SUSY	79.981 ± 0.015	**79.791** ± 0.024	79.915 ± 0.009	79.713 ± 0.031	78.274 ± 0.004	78.267 ± 0.050	55.747 ± 0.110	55.751 ± 0.157
AVERAGE	89.517	88.760	87.201	86.689	79.439	79.435	76.214	70.475

Table 4.4: *Results obtained by the Wilcoxon test for algorithm FDRF*

vs.	R^+	R^-	Exact P-value	Asymptotic P-value
DFDT	45.0	0.0	0.003906	0.006434
DFAC-FPP	45.0	0.0	0.003906	0.006434
ChiFRBCS	45.0	0.0	0.003906	0.006434

Table 4.5: *Summary of the Wilcoxon test.* •*= the method in the row improves the method of the column.* ∘ *= the method in the column improves the method of the row. Upper diagonal of level significance* $\alpha = 0.95$ *(non-adjusted), lower diagonal level of significance* $\alpha = 0.99167$ *(adjusted for* $m = 6$ *comparisons).*

	FDRF	DFDT	DFAC-FPP	ChiFRBCS
FDRF	-	•	•	•
DFDT	∘	-	•	•
DFAC-FPP	∘	∘	-	
ChiFRBCS	∘	∘		-

Table 4.6: *Average execution time (s) for all the algorithms.*

Dataset	**DFRF**	**DFDT**	**DFAC-FFP**	**ChiFRBCS**
COV	1,830	65	1742	46,817
ECO_E	12,160	402	18,800	1,263
ECO_CO	18,210	417	19,915	1,491
EM_E	11,690	324	56,279	1,276
HIGGS	17,410	617	24,077	19,889
KDD_5	5,366	212	23,508	3,615
KDD_23	11,162	241	70,925	6,551
POKER	2,333	109	810	18,918
SUSY	9,181	272	11,132	1,540

4.5.2 Scalability

In this section we provide an investigation of the scalability of the proposed approach by using an increasing number of computing units (CUs). We do so by using a metric, *speedup* σ, which is the most common metric used to evaluate scalability. The speedup is defined as the ratio of the sequential execution time to the parallel execution time. Nevertheless, the size of the datasets used here is such that the sequential version of the overall algorithm would take an unreasonable amount of time to run. We then redefine the speedup using a reference execution with Q^* computing units:

$$\sigma_{Q^*}(Q) = \frac{Q^* \cdot \tau(Q^*)}{\tau(Q)} \tag{4.5}$$

being $\tau(Q)$ the runtime using Q computing units, and Q^* the number of computing units used in the reference execution. Here $Q^* \cdot \tau(Q^*)$ is an estimation of an ideal single computing unit runtime. Indeed, it is worth noticing the $\tau(Q^*)$ also includes the overhead due to the Apache Spark framework. Furthermore, the formulation we provide makes sense only if $Q > Q^*$.

Table 4.7: *Average execution time, speedup $\sigma_8(Q)$ and utilisation $\sigma_8(Q)/Q$, obtained on the susy dataset. Table shows both the Fuzzy Partitioning and Forest Learning processes.*

	Fuzzy Partitioning			Forest Building		
# Cores	Time (s)	$\sigma_8(Q)$	$\sigma_8(Q)/Q$	Time (s)	$\sigma_8(Q)$	$\sigma_8(Q)/Q$
8	335	8.0	1.0	16490	8.0	1.0
16	226	11.87	0.74	8024	16.4	1.03
24	179	14.98	0.62	6395	20.6	0.86

In this work we chose $Q^* = 8$, corresponding to 2 slaves in our cluster. Furthermore, we split the RDD in a number of partitions that is twice the number of cores available on the cluster. Table 4.7 reports the average execution time, speedup $\sigma_8(Q)$ and utilisation $\sigma_8(Q)/Q$, obtained on the Susy dataset. The results are detailed for both the Fuzzy Partitioning and Forest Learning processes, to detail how the different components of the approach scale. As expected, the fuzzy partitioning step benefits less and less from an increased number of cores, as the limiting factor is the number of attributes. Conversely, the DFRF learning phase shows a good scalability; the trend is approximately linear up to 16 cores, showing only a slight decrease when the number of available cores goes up to 24.

4.6 Summary

In this chapter, we propose a distributed fuzzy random forest (DFRF) learning scheme that takes advantage of Apache Spark to generate an efficient and effective classifier for big

data. The approach is built upon a distributed fuzzy discretizer for big data, which provides strong fuzzy partitions for each continuous attribute; then, an ensemble of distributed fuzzy decision trees is built, employing the fuzzy information gain as splitting criterion.

By performing a set of experiments on eight big datasets, the approach is thoroughly evaluated. A comparison with a distributed fuzzy decision tree (DFDT), a distributed fuzzy associative classifier (DFAC-FFP), and ChiFRBCS-BigData shows that the approach provides very competitive results; DFRF statistically outperforms the other approaches. Concerning run times, DFRF is obviously slower than DFDT, and requires comparable execution times with respect to ChiFRBS-BigData; furthermore, DFRF is generally faster than DFAC-FFP. Finally, a scalability analysis shows that the approach is well-behaved with respect to an increment in the number of CUs.

CHAPTER

FIVE

MULTI-OBJECTIVE EVOLUTIONARY FUZZY CLASSIFIERS FOR BIG DATA

> Model building is the art of selecting those aspects of a process that are relevant to the question being asked. As with any art, this selection is guided by taste, elegance, and metaphor; it is a matter of induction, rather than deduction. High science depends on this art.
>
> Hidden Order - How Adaptation Builds Complexity — John Henry Holland

This chapter contains material from the following publications:

- Barsacchi, M. Bechini, A., Ducange, P. and Marcelloni F. (2019). Optimizing Partition Granularity, Membership Function Parameters, and Rule Bases of Fuzzy Classifiers for Big Data by a Multi-objective Evolutionary Approach. *in Cognitive Computing pp. 1-21, January 2019.*

In this chapter, a novel approach for generating, out of big data, a set of fuzzy rule-based classifiers characterised by different optimal trade-offs between accuracy and interpretability is introduced. A recently proposed distributed fuzzy decision tree learning approach is used to generate an initial rule base that serves as input to the evolutionary process. Then, the evolutionary learning scheme is integrated with an ad-hoc strategy for the granularity learning of the fuzzy partitions, along with the optimisation of both the rule base and the fuzzy set parameters. A thorough experimental investigation shows that the proposed approach is able to generate fuzzy rule-based classifiers that are significantly less com-

plex than the ones generated by the original multi-objective evolutionary learning scheme while keeping the same accuracy levels.

The chapter is structured as follows. In Sections 5.1 the problem is introduced; some background concepts are provided in Section 5.2. Section 5.3 describes the overall approach. We report the results of our experimental analysis in Section 5.4. Section 5.5 provides a brief summary of the chapter.

5.1 Fuzzy Rule Based Classifiers for Big Data: Background and State of The Art

The last decade ushered in the era of *Big Data* (Mayer-Schönberger and Cukier, 2013); classical data mining algorithms appear to be inadequate to manage Big Data. Indeed, Big Data are usually characterised in terms of the so-called four "V"s, namely volume, variety, velocity and veracity: large *volumes* of data, which are often produced at very high speed and need to be elaborated in almost real time (*velocity)*, are generated by different sources and may have different formats (*variety)* (Anuradha et al., 2015) and trustworthiness (*veracity*).

As the inadequateness of the classical algorithm for dealing with huge amount of data became increasingly clear, several researchers have introduced data mining approaches purposely designed and implemented for Big Data (Wu et al., 2014). The majority of these approaches have relied on specific distributed frameworks, such as Apache Hadoop (White, 2012) and Apache Spark (Zaharia et al., 2010), which have been recently proposed with the aim of offering the practitioner simpler ways for dealing with data storage and elaboration of Big Data. Moreover, most of the recent contributions in the field stemmed from the MapReduce paradigm (Dean and Ghemawat, 2008); it allows the implementation of both descriptive and predictive models, with the additional benefit of the possibility to make use of computing resources on the Cloud (Fernández et al., 2014).

Among the wide variety of algorithm developed using the Apache Hadoop framework, several distributed implementation of clustering algorithm have been proposed, such as DB-SCAN (Kim et al., 2014) and Fuzzy C-Means (Ludwig, 2015). Among the class of supervised approaches, a fuzzy version of Random Forests has been implemented over the same framework as well (Bechini et al., 2016a).

More recently, Apache Spark implementations of associative classification models (Bechini et al., 2016b) and of a KNN classifier (Maillo et al., 2017), respectively, have been proposed. Furthermore, big social data analysis has also taken advantage of such distributed computing frameworks (Oneto et al., 2017). A recent highlights of the main advances, challenges and objectives in designing, developing and exploiting data mining and machine learning algorithms for Big Data can be found in (Zhou et al., 2017). In the last years, a wide number of contributions of predictive models exploiting Fuzzy Models (FMs) for handling Big Data have been proposed (Elkano et al., 2017; Fernández et al., 2017; Ferranti et al., 2017; López et al., 2015; Márquez et al., 2017; del Río et al., 2015a; Segatori et al.,

2017a, 2018). It is believed that FMs are particularly well suited for handling the variety and veracity of Big Data (Fernández et al., 2016a), mainly due to their inherent capability of coping with vague, imprecise, and uncertain concepts (a broader discussion of the subject can be found in Chapter 1). Moreover, fuzzy logic has been often recognised also as an important tool for preserving the fidelity of psychological interpretation of emotion (Ayesh and Blewitt, 2015), opening up new ways to analyse the sentiment contents of huge amounts of data available today from the web. Mainly, it is the use of overlapped fuzzy labels that ensures a good coverage of the problem space; this issue is especially relevant when dealing with very large datasets.

Among the first FMs for Big Data classification is Chi-FRBCS-BigData (del Río et al., 2015a); it is a fuzzy rule-based classifier (FRBC), that exploits the MapReduce paradigm to build a classifier according to the approach previously described by Chi et al. (Chi et al., 1996). Chi-FRBCS-BigData algorithm has been later adapted for handling imbalanced big datasets (López et al., 2015); furthermore, the effects of the granularity of fuzzy partitions on the very same algorithm, have been studied in (Fernández et al., 2016b). Recently, the CHI-BD algorithm has been introduced: it is a novel distributed version of the Chi et al's approach (Chi et al., 1996), with improved results with respect to Chi-FRBCS-BigData (Elkano et al., 2017). More details can be found in (Elkano et al., 2017) and (del Río et al., 2015a), respectively.

Other recent approaches for Big Data are Fuzzy Associative Classifiers Segatori et al. (2017a) and Distributed Multi-Way Fuzzy Decision Trees (DMFDTs) Segatori et al. (2018). Even if both of them produce highly accurate models, the complexity of the relative models, in terms of both the number of rules and number of decision nodes, is very high. In general, the greater the complexity of the model, the lower the interpretability of the FMs. Indeed, the interpretability is a very important feature that characterises FMs and is believed to be highly relevant in the context of Big Data (Fernández et al., 2016a; Wang et al., 2017). As such, new methods that generate both accurate and interpretable FMs are currently under investigation in the research community on fuzzy models (Duţu et al., 2018).

The subject of interpretability has been introduced and briefly discussed in Chapter 1; as interpretability is a subjective concept, it is still hard to find an agreed definition and consequently a valid and universal measure of interpretability. A general taxonomy of interpretability measures for FRBSs has been proposed (Gacto et al., 2011); it considers the two distinct dimensions of semantics and complexity, both at the rule base (RB) and data base (DB) levels. It thus partitions the interpretability space into 4 quadrants. Concerning the DB, the semantic interpretability is usually evaluated in terms of the integrity of the fuzzy partitions, while the complexity is evaluated in terms of the number of fuzzy sets. The interpretability of the RB is usually analysed in terms of complexity and one of the most used metrics is the Total Rule Length (*TRL*) (Cococcioni et al., 2007; Ishibuchi et al., 2001; Ishibuchi and Yamamoto, 2004), that is, the total number of conditions used in the RB. More recently, the importance of other factors, like *rule relevance*, in the learning of interpretable Fuzzy Rule-based Systems, has been experimentally studied as well (Rey et al.,

2017).

In the context of FMs, multi-objective evolutionary algorithms (MOEAs) have been often used to generate models that are both accurate and interpretable. Several approaches (Antonelli et al., 2016a; Fazzolari et al., 2013) exploited the MOEAs to in order to evolve both the DB and the RB of the fuzzy rule-based systems. Nevertheless, as the computation of the accuracy of each individual requires the entire training set, large datasets require specialised approaches in order to properly deal with Big Data. As such, a distributed implementation that exploits a cluster of computing nodes may come in handy.

Recent works have focused on evolutionary-based methods for learning FMs for Big Data (Fernández et al., 2017; Márquez et al., 2017); among them is a recent implementation of an MOEA that learns concurrently the RB and DB of FRBCs, by maximising accuracy and minimising complexity (Ferranti et al., 2017). Dubbed DPAES-RCS, it extends PAES-RCS (Antonelli et al., 2014, 2016b) to the Big Data setting, by leveraging on the Apache Spark framework. The RB is learnt through a rule and condition selection strategy: for each iteration of the evolutionary process, a reduced number of rules is selected from a heuristically generated set of candidate rules; furthermore, for each rule, a reduced number of conditions is selected. Additionally, the parameters of the fuzzy sets are learnt concurrently with the RB. PAES-RCS has been thoroughly evaluated, generating satisfactory approximations of the Pareto front using a limited number of iterations (Antonelli et al., 2014).

The work described in this chapter extends DPAES-RCS with two main novelties: first, the initial rule set is now generated using a fuzzy decision tree learning algorithm, rather than the standard C4.5 used before; the rules are extracted by considering each leaf of the tree, mapping back the path to the root node. As a second novelty, a granularity learning approach has been employed: the best number of fuzzy sets composing the fuzzy partition for each continuous variable (granularity) is learnt during the evolutionary process as well. In order to do so, the *virtual partition* method introduced in (Antonelli et al., 2009b) has been exploited.

5.2 Preliminaries on FRBCs

In this chapter, the necessary notation is recalled. A generic classification task consists of assigning a class label C_m, out of a given set $C = \{C_1, \ldots, C_M\}$ of M classes, to a given unlabeled instance. A generic instance is described by a set $\mathbf{X} = \{X_1, \ldots, X_F\}$ of attributes having cardinality F. Each attribute can be either *categorical* or *numerical*; in the case of a generic categorical attribute f, X_f takes values out of a set $L_f = \{L_{f,1}, \ldots, L_{f,T_f}\}$ of T_f distinct values.

Each numerical attribute is instead associated with a universe U_f of X_f; the universe is a bounded interval in $\mathbb{R}$. For each universe U_f, a corresponding fuzzy partition can be defined: considering the attribute X_f, let $P_f = \{A_{f,1}, \ldots, A_{f,T_f}\}$ be a fuzzy partition over the relative universe U_f, where $A_{f,i}$ is a generic fuzzy set, and T_f is the number of fuzzy sets in the partition. A linguistic label $L_{f,j}$ is then assigned to each fuzzy set $A_{f,j}$; linguistic

labels bridge the gap with linguistic variables (Chapter 2) and allow dealing with both categorical and numerical attributes in a homogeneous fashion.

Here, a partition made of triangular fuzzy sets is assumed, i.e each fuzzy set $A_{f,i}$ is identified by the 3-tuple $(a_{f,i}, b_{f,i}, c_{f,i})$, where $a_{f,i}$ and $c_{f,i}$ are to the left and right extremes of the support, respectively, and $b_{f,i}$ indicates the core. Furthermore, when using strong fuzzy partitions, the following two conditions are satisfied: i) $a_{f,1} = b_{f,1}, b_{f,T_f} = c_{f,T_f}$, and ii) $b_{f,i} = c_{f,i-1}$ and $b_{f,i} = a_{f,i+1}$ for $i = 2, ..., T_f - 1$.

Let the number T_f of fuzzy sets making up the partition of X_f be used as a measure of the *granularity*. In the following, the notation $P_f(T_f)$ is used to emphasise the granularity level for P_f.

A fuzzy rule based classifier (FRBC) is an FRBS that infers the output value for an unlabelled instance using the fuzzy rules that compose the RB. Here a slightly reworked version of Eq.2.17 is used in order to define a generic m-th rule:

$$\begin{aligned} R_m : \textbf{if } X_1 \textbf{ is } L_{1,i_{m,1}} \textbf{ and } \ldots \textbf{ and} \\ X_F \textbf{ is } L_{F,i_{m,F}} \textbf{then } Y \textbf{ is } C_{k_m} \end{aligned} \tag{5.1}$$

Here, the generic linguistic label $L_{f,i_{m,f}}$ is used in place of the fuzzy set $A_{f,i_{m,f}}$, in order to provide a more general treatment. In fact, depending on the nature of X_f (that can either numerical or categorical), such a label may refer to either a fuzzy set in partition P_f or a categorical value. Furthermore, the output class C_{k_m} is used in place of the linguistic labels associated with the output.

In order to manage the situation in which the attribute does not provide any information in choosing the outcome, an additional dummy label $L_{f,0}$ for each attribute X_f, is introduced; this fictitious label expresses that X_f does not contribute to the classification (a "do not care" condition). Formally, this "dummy" label is associated with a set that has a unitary membership across the whole universe: thus, $L_{f,0}$ allows maintaining the generic rule structure of (5.1) even for rules in which the actual outcome depends only on a subset of the attributes.

A training set may be defined as a set of N tuples:
$TR = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$; here, $(\mathbf{x}_n, y_n)$ indicates the n-th input-output pair, where $\mathbf{x}_n$ is the input vector with F values (each, either numerical or categorical, for the relative attribute), and y_n is the relative classification label.

For a generic rule R_m, and given an input $\mathbf{x}_n$, the *matching degree* of the rule with the input is defined as strength of activation of the rule. If the product t-norm is used (Chapter 2), the matching degree can be written down as:

$$w_m(\mathbf{x}_n) = \prod_{f=1}^{F} \mu_{L_{f,i_{m,f}}}(x_{n,f}) \tag{5.2}$$

where $\mu_{L_{f,j_{m,f}}}(x_{n,f})$ is, in case of numerical attributes, the membership value of $x_{n,f}$ to the fuzzy set $A_{f,j_{m,f}}$ represented by label $L_{f,j_{m,f}}$ and, in case of categorical attributes, is either 0 or 1.

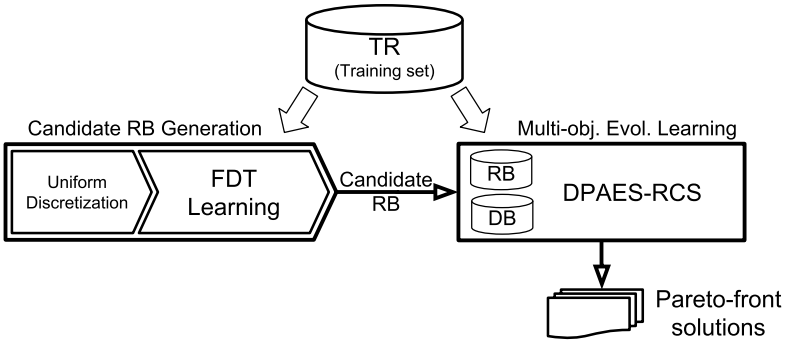


Figure 5.1: *Structure the proposed approach. In the* Candidate RB Generation*, an initial candidate RB is generated by means of a distributed FDT learning algorithm applied to uniformly partitioned attributes; then, a multi-objective evolutionary algorithm finds optimal FRBCs with different trade-offs between accuracy and complexity.*

Concerning, the many measures of complexity proposed for a FRBS, the total rule length *TRL* is among the most simple and effective ones. It quantifies the model complexity directly counting the number of conditions; as such it represents a proxy for the interpretability of the system.

Several reasoning methods (Chapter 2.3) for FRBS exists; here the "maximum matching method" has been selected; it assigns the label to an unlabelled instance, selecting the consequent of the rule that has been maximally activated by such instance. Furthermore, in case of a tie, the first one among the equally-matching rules is chosen; eventually, if no rule is activated, the instance is classified with the most frequent class.

5.3 DPAES-FDT-GL

The overall proposed approach, dubbed DPAES-FDT-GL, is structured according to the scheme reported in Figure 5.1. Two major phases are defined: i) *Candidate RB Generation*, and ii) *Multi-Objective Evolutionary Learning*. In the former, a *candidate rule base*, to be used in the following evolutionary phase, is generated. In the latter, the rule base undergoes an optimisation process, yielding a set of FRBCs that approximate the Pareto front.

As the problem at hand involves dealing with Big Data, a set of efficient and properly design algorithms must be used. In order to fulfil these requirements, all the steps of the approach have been designed to exploit the Apache Spark framework (Zaharia et al., 2010); this framework automatically deals with distribution by means of a predefined container type (known as RDD, Resilient Distributed Dataset).

In order to generate a candidate RB several approaches are possible; in DPAES-RCS the RB is generated using a C4.5 algorithm (Ferranti et al., 2017); the generated RB is then translated in a fuzzy RB. Nevertheless, as the C4.5 treats the input and output data as crisp, the candidate RB is not tuned to the fuzzy partitions used as input; thus, the usage of an

FDT algorithm should, in principle, easy the subsequent fuzzy manipulations. Indeed, in practical contexts, the RBs generated by traversing all the paths from the root down to each leaf of an FDT have proven to be both compact and interpretable (Ricatto et al., 2018). As a preliminary step, a uniform discretization is performed. The FDT learning is then carried out by means of a distributed algorithm(Segatori et al., 2018).

The Multi-Objective Evolutionary Learning (MOEL) phase uses a MOEA to evolve the initial population of individuals; in this work, a Pareto Archive Evolutionary Strategy (PAES) (Coello Coello et al., 2007) is used. The distributed implementation, which leverages Apache Spark, follows the one proposed with DPAES-RCS (Ferranti et al., 2017). The proposed DPAES implementation generates a set of satisfactory solutions in a reduced number of iterations; its fast convergence rate is of particular relevance in the setting of Big Data since a fitness evaluation is usually extremely costly.

In the following subsections a description of both the phases i) *Candidate RB Generation*, and ii) *Multi-Objective Evolutionary Learning* is provided.

5.3.1 Generating the Candidate Rule Base

The Candidate RB generation phase establishes the bases for the following evolutionary phase; as such, it is of paramount importance to generate a Candidate RB that makes the subsequent step as simple and efficient as possible. To this aim, a distributed FDT (DFDT) learning approach, suitable for dealing with Big Data (Segatori et al., 2018), is exploited. The multi-way version of the FDT is particularly appealing because simple Mamdami type rules (as in Equation 5.1) are easily extracted through a tree traversal, deriving one rule for each possible path from the root down to a leaf. Furthermore, as the tree operates on fuzzy data, the extracted RB is tuned to the input fuzzy partitions. The decision tree algorithm (Segatori et al., 2018) has been modified to better suit our needs: first, a uniform discretization with T_{max} evenly-spaced triangular fuzzy sets that make up strong partitions, has been used in place of the origin discretization. Furthermore, while each path from the root down to a leaf can be seen a Mamdani-type rule with certainty degree for all the classes in the consequent, the extracted rules have been simplified using the class with the highest certainty only (a different approach, in which all classes are retained, is discussed in Section 6.2.3).

Each node $\mathcal{N}$ of the FDT is associated with a fuzzy set A_n, and thus with a subset of TR (the set of examples in the support of A_n), and the root corresponds to the whole TR. The FDT construction starts from the root and follows a top-down approach; unless a termination condition is not satisfied, a newly-generated node gives rise to T_{max} child nodes according to the fuzzy partition of the attribute chosen for that specific splitting. The selected termination conditions are: The i) all the instances in the node belong to the same class; ii) the number of instances in the node is lower than a fixed threshold λ, iii) the tree has reached a maximum fixed depth β, and iv) the value of the fuzzy information gain is lower than a fixed threshold ε.

In this procedure, an attribute can be considered only once in the same path from the root to a leaf, thus producing rules that are simple and interpretable. The attribute that drives the splitting is selected as the one that yields the best *fuzzy information gain*, which will be defined below.

Given a parent node $\mathcal{PN}$, let $\mathcal{CN}_j$ indicate the generic j-th child node, $j = 1, \ldots, T_{max}$; the number of child nodes for a parent node depends on the granularity of the fuzzy partition associated with the splitting attribute X_f. Each child node $\mathcal{CN}_j$ contains only the instances that belong to the support of the fuzzy set $A_{f,j}$; here, let $S_{f,j}$ be the set of instances for $\mathcal{CN}_j$, and S_f be the set of instances in the parent node. We measure the cardinality of the fuzzy set G_j associated with the node $\mathcal{CN}_j$ as:

$$\left|G_j\right| = \sum_{i=1}^{N_j} \mu_{G_j}(\mathbf{x}_i) = \sum_{i=1}^{N_j} TN(\mu_{A_{f,j}}(x_{f,i}), \mu_G(\mathbf{x}_i)) \tag{5.3}$$

being N_j the number of instances in the support $S_{f,j}$, the operator TN a T-norm and $\mu_G(\mathbf{x}_i)$ is the membership degree of instance $\mathbf{x}_i$ to parent node $\mathcal{PN}$ (for the root node, $\mu_G(\mathbf{x}_i) = 1$).

The fuzzy information gain *FGain* is used for selecting the splitting attribute; it is computed, for a generic attribute X_f with partition P_f, as

$$FGain(P_f; I_G) = FEnt(G) - WFEnt(P_f; I_G) \tag{5.4}$$

being I_G the support of fuzzy set G. The Fuzzy Entropy $FEnt(G)$ has been defined in Eq. 3.5 and the weighted fuzzy entropy $WFEnt(P_f; I_G)$ has been reported in Eq. 3.6.

In the case of categorical attributes, the parent node is split into a number of child nodes $\mathcal{CN}_j$ equal to the number of possible values for the attribute. Each node $\mathcal{CN}_j$ is thus characterised by a "dummy" fuzzy set G_j, whose cardinality is :

$$\left|G_j\right| = \sum_{i=1}^{N_j} \mu_{G_j}(\mathbf{x}_i) = \sum_{i=1}^{N_j} TN(1, \mu_G(\mathbf{x}_i)) \tag{5.5}$$

Figure 5.2 summarises the distributed implementation of the candidate RB generation phase. The distribution of the computation across a cluster of Computing Units (CUs) is highlighted.

The distributed FDT learning proceeds by iteratively executing a MapReduce step; the mapper computes the fuzzy entropies needed in order to find the best split, and the reducer aggregates the data and actually performs the split. The nodes to be possibly split are kept in a list, updated in each iteration; at the beginning of each iteration at most *MaxY* elements at a time are fetched to be processed in a MapReduce step. Furthermore, for each selected split, all the child nodes are pushed into the list. The good scalability figures of the distributed algorithm have been reported before (Segatori et al., 2018). Still, the most critical aspect relates to the system requirements for the used cluster of computers. Notably, the maximum number of nodes *MaxY* that can be processed in parallel depends on the amount of RAM available on the cluster.

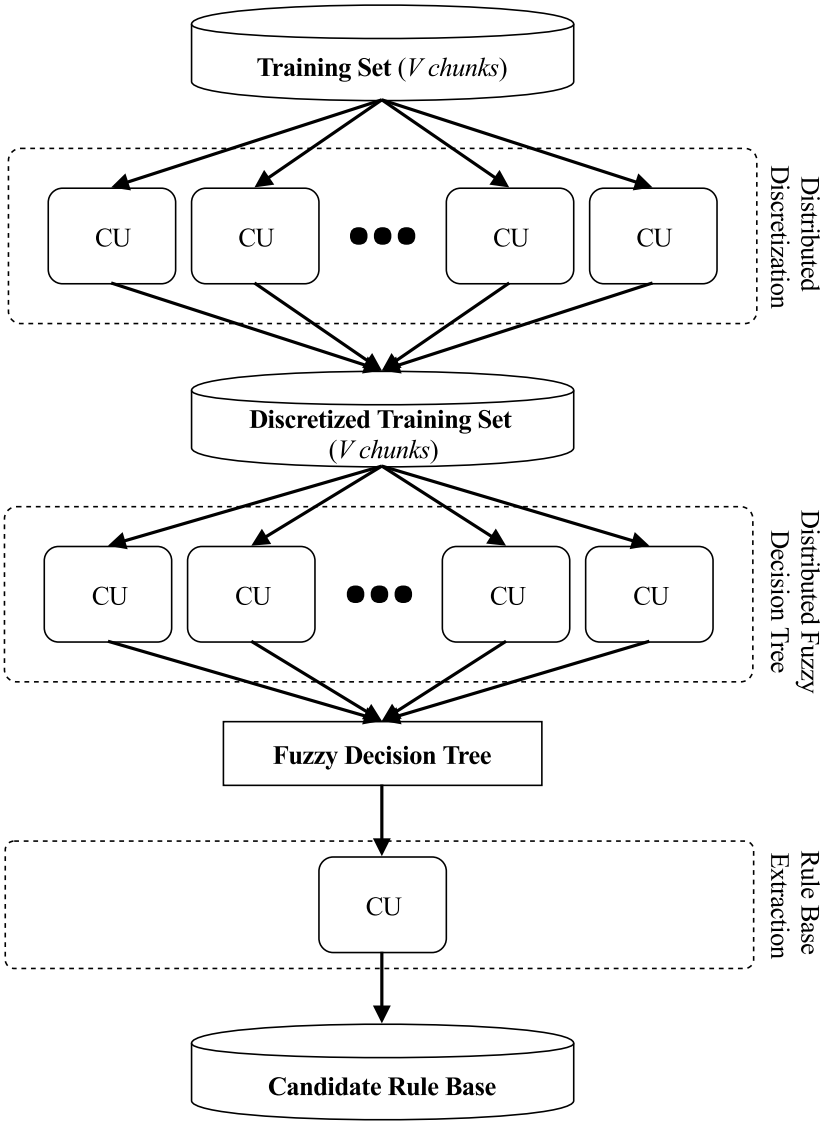


Figure 5.2: *Schematic of the distributed RB generation phase; the three major steps are: i) discretization, ii) fuzzy decision tree learning, and iii) rule base extraction.*

5.3.2 Distributed MOEL

The evolutionary process

The multi-objective evolutionary learning phase is used to generate a satisfactory set of solutions, which approximates the accuracy-*TRL* Pareto front. The two objectives used here are i) maximisation of the classification accuracy on the training set, and ii) minimisation of the complexity of the FRBC.

To this aim, a distributed MOEA is employed; the algorithm consumes the candidate RB, produced by the previous phase, as input, and outputs the set of solutions. The base structure of the algorithm is that proposed for DPAES-RCS (Ferranti et al., 2017); it is based on (2+2)M-PAES (Cococcioni et al., 2007), which, in turn, relies on a modified version of the well-known (2+2)PAES (Knowles and Corne, 2000). (2+2)M-PAES is a steady-state evolutionary algorithm that stores the non-dominated solutions in an archive and uses two current solutions at each iteration. Nevertheless, unlike classical (2+2)PAES, the current solutions at extracted randomly at each iteration. Furthermore, instead of considering uniform fuzzy partitions with a pre-fixed number of fuzzy sets, the granularity of each partition here is learned as well. Thus, the chromosome coding has to accommodate this additional requirement.

The proposed evolutionary process works by evolving: i) the set of rules and conditions, ii) the cores of the fuzzy sets and, iii) the granularity of each partition. For (i), a subset of rules is selected out of the initial set of candidate rules; likewise, rule conditions are activated or deactivated as well. In (ii) the fuzzy partitions are rearranged by moving the cores of triangular fuzzy sets composing the partitions. Finally, in (iii) the number T_f of fuzzy sets for a given partition (i.e. the granularity level) varies in the range $[T_{min}, T_{max}]$.

The initial set of candidate rules is constructed by exploiting a uniform strong triangular fuzzy partition for each numeric variable X_f, each containing T_{max} fuzzy sets. Furthermore, the evolutionary phase evolves the RB, via rule and condition selection, still referring to the original partitions. As such, a strategy is needed in order to interface the *virtual RBs* (Antonelli et al., 2009a), the and *virtual partitions* (Antonelli et al., 2009b) with the actual RB (defined with the actual granularity) and the *real partitions*. Such mapping strategy, needed both for the RB and for the parameters of the fuzzy sets, allows a efficient execution of crossover and mutation operators.

Even though during the evolutionary process we generate RBs, denoted as virtual RBs, and tune the fuzzy set parameters by using the virtual partitions, the mapping needs to be performed each time a fitness evaluation is requested.

RB The strategy adopted to map a virtual RB defined on (virtual) partitions with T_{max} fuzzy sets onto a concrete RB defined on variables partitioned with T_f fuzzy sets follows the one proposed in (Antonelli et al., 2009a,b).

A generic RB is made up of rules with generic propositions of the form: X_f **is** $\widehat{A}_{f,h}$,

$h \in [1, T_{max}]$. Such a proposition is mapped to X_f **is** $\widetilde{A}_{f,s}$, with $s \in [1, T_f]$; $\widetilde{A}_{f,s}$ is defined as the the fuzzy set that is *most similar* to $\widehat{A}_{f,h}$ (amongst the T_f fuzzy sets $\widehat{A}_{f,h}$ defined on X_f). The specific form of the similarity metric is an open problem; when dealing with strong triangular fuzzy partitions, the simplest similarity measure is the distance between the centres of the cores of the two fuzzy sets. The trivial case in which two different cores are at the same distance from the centre of $\widehat{A}_{f,h}$ is handled by randomly choosing on of the two.

The structure of the mapping strategy defined above may lead to a situation in which distinct fuzzy sets, defined on the partitions of the virtual RB, do map onto the same fuzzy set on the partitions used in the concrete RB; as such, different rules in the virtual RB may correspond to the same rule in the concrete RB. For this reason, the concrete RB is searched against duplicate rules. Thus, the concept of virtual RB allows us to explore the search space and, at the same time, exploit the (sub)optimal solutions found during the evolutionary process.

Fuzzy set parameter The mapping strategy also applies to the fuzzy set parameters; to do so, a piecewise non-decreasing linear transformation (Antonelli et al., 2009b) is applied. Given the initial partition of the input variables, the parameters of the fuzzy sets (the cores) making up the partition are tuned by employing the transformation. The necessary notation is provided below; let $\widetilde{P}_f = \left\{\widetilde{A}_{f,1}, \ldots, \widetilde{A}_{f,T_f}\right\}$ be the initial partition, and $P_f = \left\{A_{f,1}, \ldots, A_{f,T_f}\right\}$ be the transformed partitions. Let also assume that the two partitions are defined on the same universe (i.e. $\widetilde{U}_f \equiv U_f$),

Furthermore, let $t(x_f) : U_f \rightarrow \widetilde{U}_f$ be the piecewise linear transformation. Thus, the transformed fuzzy set $\widetilde{A}_{f,i}$ is obtained by applying the transformation to the original one $A_{f,i}$, such that $A_{f,i}(x_f) = \widetilde{A}_{f,i}\left(t\left(x_f\right)\right) = \widetilde{A}_{f,i}\left(\widetilde{x}_f\right)$.

The piecewise linear transformation operates on a sequence of representatives for each fuzzy partition; the representatives are linked to variation of slopes of the piecewise linear transformation $t(x_f)$, for each variable X_f. When triangular fuzzy sets are employed, the cores can be used as representatives. Formally, let $\widetilde{b}_{f,1}, \ldots, \widetilde{b}_{f,T_f}$ and $b_{f,1}, \ldots, b_{f,T_f}$ be the representatives of $\widetilde{A}_{f,1}, \ldots, \widetilde{A}_{f,T_f}$ and $A_{f,1}, \ldots, A_{f,T_f}$, respectively.

For each pair of representatives, the transformation $t(x_f)$ is defined on such an interval $b_{f,i-1} \leq x_f \leq b_{f,i}, i = 1 \ldots T_f$ as:

$$t(x_f) = \frac{\widetilde{b}_{f,i} - \widetilde{b}_{f,i-1}}{b_{f,i} - b_{f,i-1}} \cdot (x_f - b_{f,i-1}) + \widetilde{b}_{f,i-1} \tag{5.6}$$

An example of such a transformation is reported in Figure 5.3; the transformation $t(x_f)$, as per Equation 5.6, is applied assuming a uniform initial partition and a maximum granularity $T_{max} = 7$.

As $b_{f,1}$ and b_{f,T_f} are the extremes of the universe U_f of X_f, the transformation $t(x_f)$ depends on $T_f - 2$ parameters; thus, the transformation $t(x_f)$ can be rewritten as $t(x_f) =$

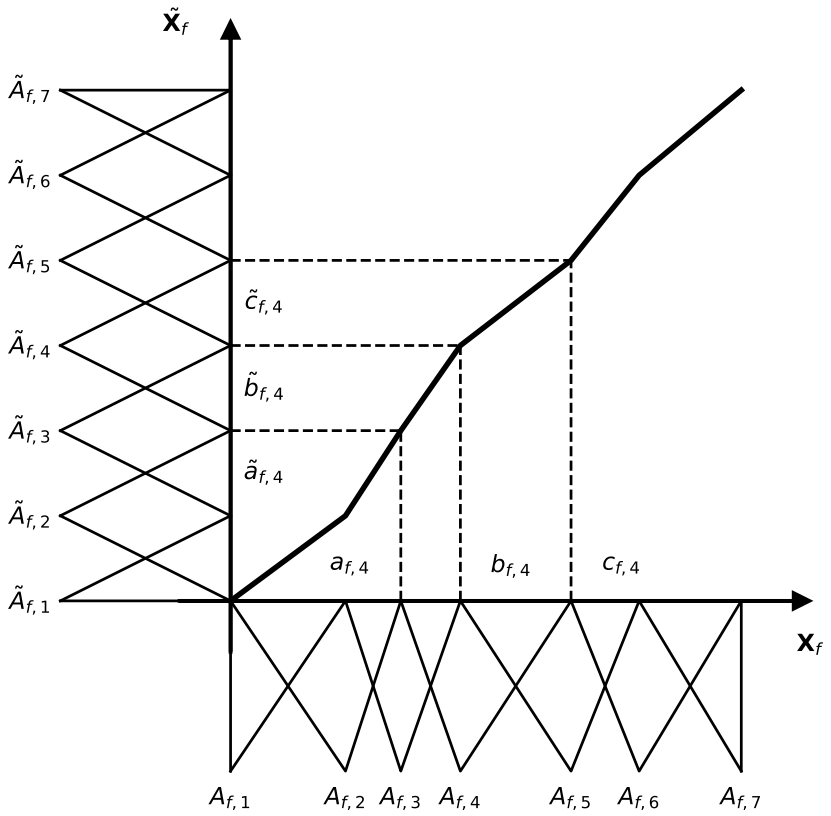


Figure 5.3: *An example of piecewise linear transformation; the transformation $t(x_f)$, as per Equation 5.6, is applied assuming a uniform initial partition and a maximum granularity $T_{max} = 7$*

.

$t\left(x_f; b_{f,2}, \ldots, b_{f,T_f-1}\right)$ (Antonelli et al., 2009b). The transformation proceeds as follows: given the values $b_{f,2}, \ldots, b_{f,T_f-1}$, the partition P_f is generated by transforming the three points $(\widetilde{a}_{f,i}, \widetilde{b}_{f,i}, \widetilde{c}_{f,i})$ that define the generic triangular fuzzy set $\widetilde{A}_{f,j}$, into $(a_{f,i}, b_{f,i}, c_{f,i})$ using the inverse transformation $t^{-1}(\widetilde{x}_f)$.

An example of a transformation, with a granularity of $T_f = 5$, is reported in Figure 5.4; the very same function of Figure 5.3 is used, but with a different granularity, lower than the initial value of $T_{max} = 7$.

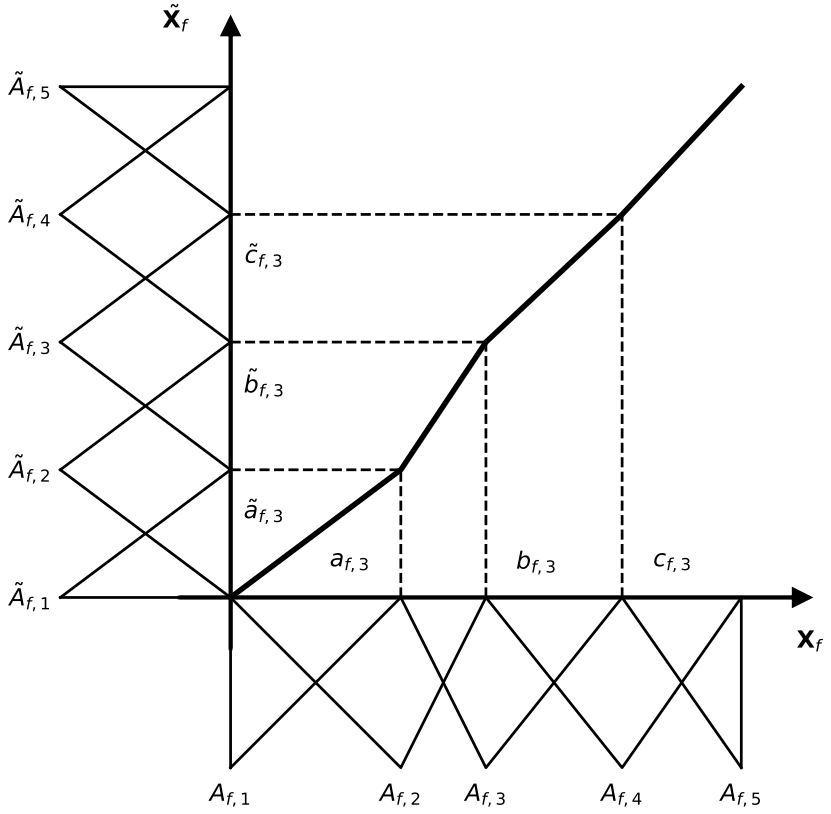


Figure 5.4: *An example of piecewise linear transformation $t(x_f)$ (the same as in Figure 5.3) applied with maximum granularity $T_f = 5$, lower than the initial value of $T_{max} = 7$.*

Objectives, chromosomes and genetic operators

In this section, a description of the objective functions, chromosome coding and mating operators is provided.

Here, each individual is associated with a chromosome and with a bi-dimensional *objective vector*; as discussed before, the two elements of the objective vector are i) the complexity of the actual RB (measured as the *TRL*), and ii) its accuracy (measured as classification rate over the training set).

A generic chromosome, C hereafter, is made up of three sub-vectors: (C_R, C_T, C_G); the first C_R encodes to the virtual RB, the second C_T defines the parameters of the fuzzy sets,

and the third C_G codifies the number of fuzzy sets.

In the following, the initial virtual RB obtained in the first phase is denoted as J_{FDT} (RBs are defined with T_{max} fuzzy sets for each partition.), and the corresponding number of rules is reported as M_{FDT}. Furthermore, to generate RBs that are both compact and interpretable, the coding of the virtual RB is constrained to contain no more than M_{max} rules.

The first sub-vector C_R is composed of M_{max} pairs $\mathbf{p}_m = (k_m, \mathbf{v}_m)$, one for each possible rule in the RB. For a generic pair, k_m identifies the index of the selected rule in J_{FDT}, and thus it is allowed to assume values in the interval $[0, M_{FDT}]$; as the system should be allowed to generate RBs with a number of rules lower than M_{max}, k_m can be set to 0 to exclude the m-th rule from the RB. Moreover, the second element $\mathbf{v}_m$, encodes a boolean"mask" vector $\mathbf{v}_m = \left[v_{m,1}, \ldots, v_{m,F}\right]$, in which the generic element $v_{m,f}$ indicates, for attribute X_f, whether to consider or not the relative condition in the rule (if not, it becomes a "don't care" condition, as previously discussed in Section 5.2).

As for the second sub-vector C_T, it encodes the position of the T_{max} distinct fuzzy sets within each strong fuzzy partition for all the F attributes; as each partition is a strong triangular fuzzy partition, C_T is a vector of F vectors, each containing the $T_{max}-2$ positions of the cores; to allow the transformation to occur, C_T needs to contain enough information to define the slope of the piecewise linear transformation $t(x_f)$ (and, as such, also t^{-1}). A further constraint is needed to impose that $b_{f,i} < b_{f,i+1}$, $\forall i \in [2, T_{max} - 1]$, and to avoid an excessive departure of the cores with respect to the uniform partition; as such the value for the generic core $b_{f,i}$ is restricted to vary in the interval

$$\left[\widetilde{b}_{f,i} - \frac{\widetilde{b}_{f,i} - \widetilde{b}_{f,i-1}}{2}, \widetilde{b}_{f,i} + \frac{\widetilde{b}_{f,i+1} - \widetilde{b}_{f,i}}{2}\right], \forall i \in [2, T_{max} - 1]\,. \tag{5.7}$$

The third sub-vector C_G specifies the number of fuzzy sets defining the partition for each attribute, i.e. its granularity. It is a vector of length F, whose f-th elements T_f vary in the range $l\,[2, T_{max}]$, coding the number of fuzzy sets to be used in the actual partition $P_f(T_f)$. As discussed in Section 5.3.2, the values contained in C_G are used to generate the concrete RB from the virtual RB coded in C_R.

Concerning the genetic operators, MOEL generates the offspring population using both crossover and mutation operations. The RB sub-vector C_R undergoes a two-point crossover, with the crossover point always placed between two rules, and two mutation operators: i) first, a random rule (actually, a pair $\mathbf{p}_m$) is selected; then, the rule in $\mathbf{p}_m$ is replaced with another rule randomly chosen out from the candidate rule base; ii) for a random rule, each position $\mathbf{v}_{m,f}$ of its condition mask is scanned, and complemented with probability $P_{cond} = \frac{2}{F}$.

The C_T sub-vector is subject to a BLX-α-crossover, with $\alpha = 0.5$ and to a mutation operator; the mutation is applied as follows: first, a feature index f is randomly chosen in $[1, F]$; an index i, identifying the fuzzy set in the partition, is randomly selected in the interval $[2, T_{max} - 1]$. Then the value $b_{f,i}$ is randomly chosen in the interval defined in Equation 5.7.

Lastly, C_G undergoes an one-point crossover, with the crossover point random selected in $[1, F]$ and a simple mutation: first, a gene (feature) f in randomly selected in the interval $[1, F]$; then the value is (randomly) either increased or decrease by one. If the resulting value is outside the allowed interval $[2, T_{max}]$, the mutation is escaped.

The aforementioned set of mating operators has experimentally shown a good balance between exploration and exploitation, thus being suitable for driving the evolutionary algorithm towards good approximations of the Pareto fronts.

The Distributed MOEL

The topic of parallel/distributed MOEA implementation has been highly discussed, even before the widespread availability of cluster/cloud computing frameworks (Van Veldhuizen et al., 2003); a plethora of approaches, exploiting different solutions, have been proposed during the last decades. Nevertheless, both the recent availability of efficient frameworks like Apache Spark, and the growing handiness of cloud resources, favoured certain solutions w.r.t others. Notably, "master-slave" paradigm (Van Veldhuizen et al., 2003), inherent in typical Spark programs, has been among the most thriving ones; it is mainly attributable to its good scalability w.r.t. to the size of the training set, and its ability to deal with big datasets.

Even if other paradigms, such as the "islands" and "diffusion" ones, exist, they are much more suited with other distributed computing frameworks (Coello Coello et al., 2007; Van Veldhuizen et al., 2003); furthermore, it is often the case that the accuracy may be affected by the number of used CUs.

The proposed implementation of the distributed MOEL phase for DPAES-FDT-GL is presented in Figure 5.5; the picture focuses on the distribution of the workload across a cluster of computing nodes. Both the initialisation of the archive required by the genetic algorithm (upper part of the figure), and the evolutionary procedure itself (lower part) are implemented in such a way to benefit from more computing nodes.

The master task, running on the driver node in the Apache Spark cluster, manages the execution of the algorithm; also, given the limited size of the rule base, it computes the *TRL* for the current solutions at each iteration.

The evaluation of the accuracy is the most resource-intensive phase of the algorithm since it requires the evaluation of the RB over the whole TR. Thus, in order to exploit the Apache Spark framework, the dataset TR is split into V chunks; the RB is then separately evaluated on each chunk, by a slave task running on a cluster CU (as reported in Figure 5.5). The slave task, which has the same structure for all the CUs, works by consuming as input the two solutions to be evaluated and producing, for each of them, the number of successful classifications over the target chunk of TR. Then, the driver aggregates the intermediate results, generating the final accuracy for each solution.

Two factors influence the complexity of the accuracy evaluation phase: i) the number of instances in the training set $|TR|$, and ii) the complexity of the currently evaluated rule

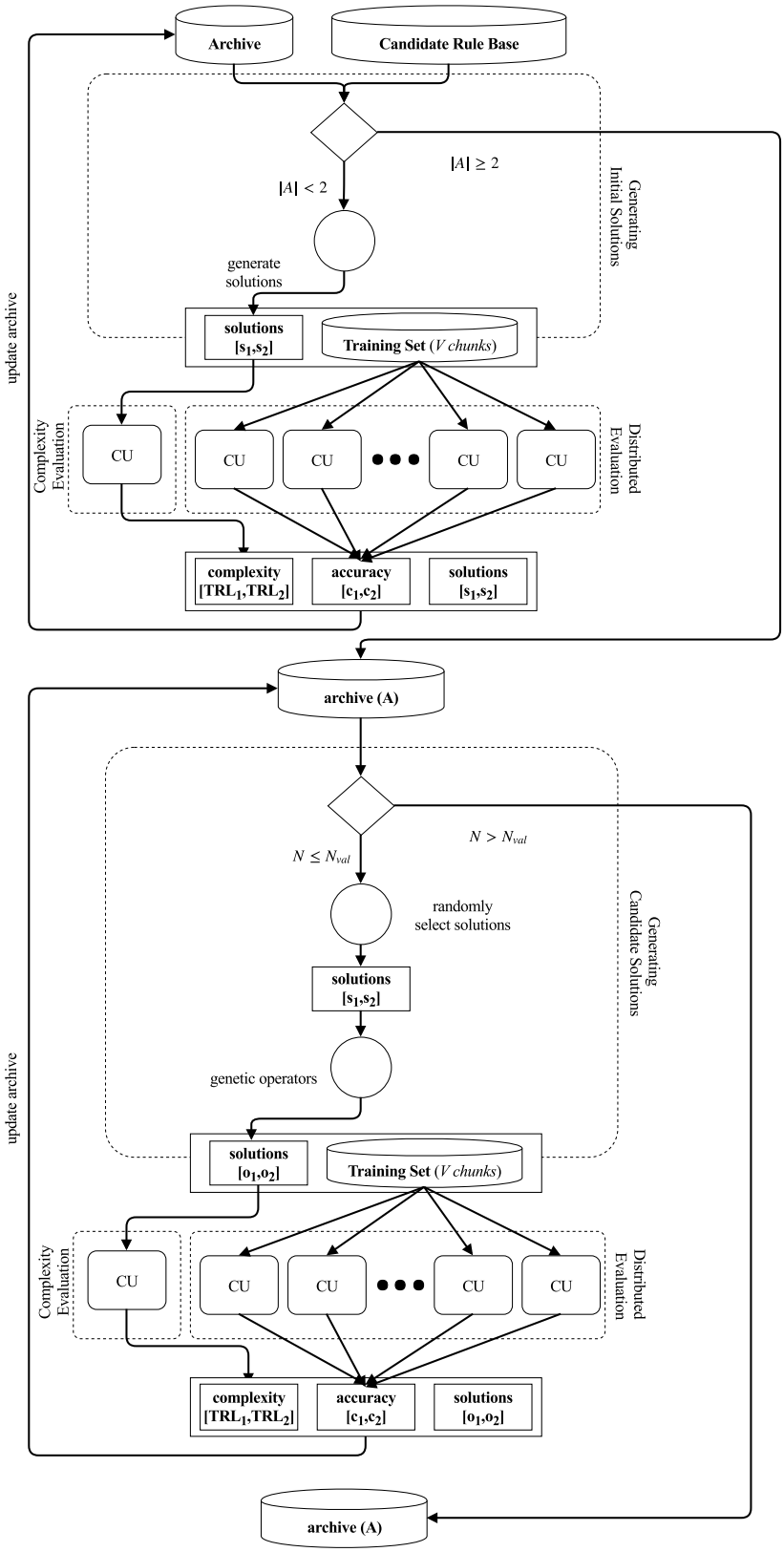


Figure 5.5: *Schematic of the distributed multi-objective evolutionary learning implementation.*

base (two RBs are evaluated at each iteration); as the average complexity gets smaller as the population evolves (the MOEA pushes towards simpler solutions, as well as on more accurate ones), the runtime for each iteration decreases during the execution of the algorithm.

As a final remark, the scalability of the adopted MOEL with respect to the used CUs has been shown before to be almost linear (Ferranti et al., 2017). This result allows using additional CUs to effectively reduce the runtimes of the algorithm.

5.4 Experimental Results and Comparisons

This section describes the results of an experimental study designed to thoroughly evaluate DPAES-FDT-GL. First, i) the solutions provided by DPAES-FDT-GL are evaluated in terms of classification accuracy, complexity, and interpretability; then, ii) the algorithm is compared with the original DPAES-RCS, from which it has been derived.

Furthermore, an additional comparison with DPAES-FDT, which adopts the FDT in the candidate RB generation phase but employs no granularity learning, is performed, to better understand the relative contributions of the FDT and of the granularity learning.

All the experiments described here have been performed on eight datasets. The datasets are reported in Table 5.1; their number of instances, attributes, classes and their sizes are reported as well. For the sake of reproducibility, the datasets have been extracted from UCI[1] and the LIBSVM[2] repositories, and have been split into 5 folds, thus performing a 5-fold cross-validation.

Table 5.1: *Datasets used in the experiments: here n and c denote numerical and categorical attributes, respectively.*

Dataset Name	# Instances	# Attributes	# Classes	Size
Covertype 2 (COV_2)	581 012	54 (n:10, c:44)	2	75.2 MB
Covertype 7 (COV_7)	581 012	54 (n:10, c:44)	7	75.2 MB
eCO (ECO)	4 178 504	16 (n:16)	10	534 MB
eME (EME)	4 178 504	16 (n:16)	10	535.2 MB
Higgs (HIG)	11 000 000	28 (n:28)	2	8.04 GB
Kddcup 2 (KDD_2)	4 856 151	41 (n:26, c:15)	2	476 MB
PokerHand (POK)	1 025 010	10 (c:10)	10	24.5 MB
Susy (SUS)	5 000 000	18 (n:18)	2	2.40 GB

All the experiments described here have been executed on a cluster of up to 7 machines (6 worker nodes and one master node); the cluster run Apache Spark 2.2.0. All the machines used in the cluster (*master* and the *workers*) are equipped with 4 vCPU, 8GB of RAM, and

[1] Available at `https://archive.ics.uci.edu/ml/datasets.html`
[2] Available at `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

160 GB Hard Drive; furthermore, all the machines run Ubuntu 14.04. The datasets are stored on the Hadoop Distributed File System (HDFS), running with a datanode on each worker. As regards the cluster manager, the standalone cluster manager provided by Apache Spark has been used in all the experiments.

Table 5.2 lists the parameter values used for both DPAES-FDT-GL and DPAES-FDT; starting with the values proposed in DPAES-RCS (Ferranti et al., 2017), the best values have been extracted both from previous experiences and experimental evaluations. The selected number of evaluations (here fixed to 50,000) reflects the results obtained in (Antonelli et al., 2014): there, it has been shown that 50,000 fitness evaluations allow obtaining Pareto fronts statistically equivalent to the ones achieved after 1 million evaluations.

The values for the granularity learning (the number of fuzzy sets per partition varies between $T_{min} = 3$ and $T_{max} = 7$) have been selected according to the following considerations: first, as a strong triangular fuzzy partitioning scheme is used, both the first and last fuzzy sets of each partition are fixed to the ends of the universe; as such, a partition is meaningful if contains at least three fuzzy sets. Second, several studies (Miller, 1956) reported that in order to preserve the interpretability, in terms of human processing capability, the number of linguistic terms per linguistic variable should be approximately 7 ± 2. As such, the value $T_{max} = 7$ has been used; further, it has also been found before that using 7 or 9 as T_{max} yields very similar results (Antonelli et al., 2009a).

Table 5.2: *The parametrization used in the experiments for DPAES-FDT-GL and DPAES-FDT is reported.*

Parameter	Description	Value
N_{val}	Total number of fitness evaluations	50000
AS	(2+2)M-PAES archive size	64
M_{max}	Maximum number of rules in a virtual RB	100
T_f	Number of fuzzy sets for each continous attribute X_f	7
P_{C_R}	Probability of applying crossover operator to C_R	0.6
P_{C_T}	Probability of applying crossover operator to C_T	0.5
P_{C_G}	Probability of applying crossover operator to C_G	0.5
P_{MRB_1}	Probability of applying first mutation operator to C_R	0.1
P_{MRB_2}	Probability of applying second mutation operator to C_R	0.7
P_{M_T}	Probability of applying mutation operator to C_T	0.6
P_{M_G}	Probability of applying mutation operator to C_G	0.2
T_{max}	Maximum number of fuzzy sets for each linguistic variable	7
T_{min}	Minimum number of fuzzy sets for each linguistic variable	3

The candidate RB, used as initial population, has been obtained using the multi-way DFDT algorithm (Segatori et al., 2017a); here, a uniform discretization, with $T_{max} = 7$ linguistic values for each numeric attribute, has been used. The parameter values for the DFDT follow the ones used in the original manuscript (Segatori et al., 2017a): i) the minimum number of instances per leaf is limited to the 0.1% of the total number of instances,

and ii) the maximum tree depth β has been set to 10, in order to allow rules with an adequate number of antecedent, without generating too many of them. In Table 5.3, the average number of *generated* rules, as well as the average number of selected features, are reported.

Table 5.3: *Min number of instances per leaf used for the DFDT algorithm, and average numbers of rules and attributes in the RBs extracted from the generated FDTs.*

Dataset	min # inst. per leaf	$\overline{\text{Rules}}$	$\overline{\text{Attributes}}$
COV_2	1	13 392.8	12.0
COV_7	1	18 176.2	53.0
ECO	334	6 683.0	13.0
EME	334	9 226.6	16.0
HIG	880	4 138.0	21.0
KDD_2	391	451.6	22.2
POK	80	28 561.0	5.0
SUS	400	10 770.0	18.0

5.4.1 DPAES-FDT-GL: an Experimental Evaluation

In this section, a thorough characterisation of DPAES-FDT-GL is provided. First, an analysis of the Pareto front approximation generated during the optimisation process is given; to this aim, previously proposed methods (Ferranti et al., 2017) are employed. For each fold, the set of solutions constituting the Pareto front is extracted; the front is then sorted by decreasing accuracy, and three solutions are collected. The FIRST, MEDIAN and LAST, are the most accurate solution, the median in the set and the least accurate, respectively, with regards to the accuracy.

The characteristics of these three solutions are reported in Table 5.4; there, for each dataset and for each representative solution, the average values and the standard deviations of the accuracy on both the training (Acc_{Tra}) and test (Acc_{Tst}) sets are reported. Furthermore, the average values and the standard deviations of the complexity (measured as *TRL*) and of the number (N_{NDS}) of non-dominated solutions contained in the archive at the end of the evolutionary process are reported as well.

The results provided in Table 5.4 are highly competitive (a comparison with the state-of-the-art is provided in Section 5.4.2), while reasonably simple (low *TRL*). This seems to suggest that DPAES-FDT-GL generates both accurate and interpretable systems; furthermore, when the accuracies obtained on the training and test sets are compared, there are no signs of overtraining.

With the aim of providing a more fine-grained analysis of the interpretability, Table 5.5 lists the average number M of rules, the average number $\widehat{F}$ of selected features, and the average number $\#F_{set}$ of fuzzy sets obtained via granularity learning (for each selected

Table 5.4: *Average values and standard deviations of the accuracy on the training and test sets and of TRL of the FIRST, MEDIAN and LAST solutions and average values and standard deviations of the number of non-dominated solutions generated by DPAES-FDT-GL.*

	FIRST			MEDIAN			LAST			
Dataset	Acc_{Tra}	Acc_{Tst}	*TRL*	Acc_{Tra}	Acc_{Tst}	*TRL*	Acc_{Tra}	Acc_{Tst}	*TRL*	N_{NDS}
COV_2	75.843 ± 0.002	75.767 ± 0.003	107.4 ± 23.6	75.378 ± 0.002	75.320 ± 0.003	43.4 ± 11.8	66.153 ± 0.097	66.203 ± 0.096	12.2 ± 7.5	37.0 ± 7.5
COV_7	67.649 ± 0.012	67.618 ± 0.012	11.4 ± 3.0	67.611 ± 0.011	67.582 ± 0.012	8.8 ± 1.8	67.172 ± 0.007	67.157 ± 0.007	5.8 ± 0.8	7.0 ± 2.8
ECO	76.261 ± 0.005	76.266 ± 0.005	101.2 ± 24.9	74.069 ± 0.007	74.074 ± 0.007	34.6 ± 10.3	56.816 ± 0.089	56.801 ± 0.089	5.0 ± 0.0	37.8 ± 8.6
EME	81.225 ± 0.005	81.193 ± 0.005	136.8 ± 23.0	78.997 ± 0.007	78.981 ± 0.007	50.2 ± 26.4	62.793 ± 0.039	62.793 ± 0.039	5.8 ± 1.8	44.8 ± 6.6
HIG	65.040 ± 0.003	65.035 ± 0.004	48.4 ± 23.2	63.625 ± 0.007	63.610 ± 0.007	22.0 ± 1.7	58.718 ± 0.003	58.697 ± 0.003	6.2 ± 1.5	24.2 ± 7.0
KDD_2	99.886 ± 0.008	99.886 ± 0.010	24.6 ± 6.5	99.883 ± 0.008	99.882 ± 0.009	13.6 ± 2.5	94.423 ± 0.094	94.415 ± 0.094	5.4 ± 0.5	15.0 ± 4.8
POK	61.778 ± 0.011	61.806 ± 0.001	90.2 ± 7.1	56.061 ± 0.008	55.989 ± 0.008	37.6 ± 5.7	49.870 ± 0.009	49.822 ± 0.009	9.2 ± 3.3	55.2 ± 4.3
SUS	78.628 ± 0.004	78.608 ± 0.004	63.0 ± 17.1	78.362 ± 0.006	78.361 ± 0.006	28.2 ± 8.3	72.525 ± 0.052	72.521 ± 0.052	7.4 ± 3.8	28.0 ± 5.3

numerical feature), for the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL.

The results in Table 5.5 show that both the number of rules and the number of features are reasonably low, thus suggesting that the learnt RB is highly interpretable; the interpretability is even strengthened by the fact that the mean number of fuzzy sets per feature is lower than 5. Furthermore, comparing the number of rules M with TRL, it is noted that the average rule length (not reported here) is very low; thus, the evolved RBs are composed by generic rules.

As regards the DPAES-FDT-RCS runtime, the average execution time for DPAES-FDT-RCS (in seconds), as well as its standard deviation are reported in Table 5.6 for all the datasets. For the sake of reproducibility, we remark that the execution times have been measured on a cluster with 6 slaves, with 4 cores each (for a total of 24 cores). Table 5.6 reports both the total execution time and the runtime for the distributed evolutionary optimisation phase (DEO); as expected, the DEO phase is the most time consuming one.

In the following paragraph, an example of the results provided by DPAES-FDT-GL is given. Here, the MEDIAN solution obtained on the first fold of SUSY [3] dataset is reported.

According to the results listed in Table 5.4, the MEDIAN solution offers an optimal trade-off in terms of accuracy and TRL. SUSY is a widely used binary classification dataset, in which a signal process that produces supersymmetric particles has to be discriminated from a background process (which does not produce anything); all the data have been generated via Monte Carlo simulations. Of the 18 attributes, the first 8 are kinematic properties directly measured by the particle detectors in the accelerator. The remaining ten features are high-level features derived by physicists to help discriminate between the two classes. In the following figures and tables, the features will be labelled as: lepton 1 p_T, lepton 1 η, lepton 1 ϕ, lepton 2 p_T, lepton 2 η, lepton 2 ϕ, missing energy magnitude, missing energy ϕ, MET_rel, Axial MET, M_R, M_TR_2, R, M_{T2}, $\sqrt{\hat{S}_R}$, M_{Δ_R}, $\Delta\Phi_{R^\beta}$ and $cos(\theta_{R+1})$. Additional information can be found in the original manuscript (Baldi et al., 2014).

The original uniform fuzzy partition (dashed line) of the MEDIAN solution, are reported, as well as the learned fuzzy partition (solid line), in Figure 5.6; each one of the 18 attributes is reported in a different panel. Furthermore, the corresponding RB is provided in Figure 5.7. The RB is composed of 7 rules, with a maximum of 3 antecedents each.

The linguistic terms used to label the fuzzy set depends on the granularity of the partitions; in the case of 3 fuzzy sets the terms *low*, *medium* and *high* are used, while in the case 7 fuzzy sets the term set is: *very_low*, *low*, *medium-low*, *medium*, *medium-high*, *high* and *very_high*. The labelling for 4, 5 and 6 fuzzy sets has been obtained by interpolating in between.

[3] The SUSY dataset is available at `https://archive.ics.uci.edu/ml/datasets/SUSY`

Table 5.5: *Average values and standard deviations of the number of rules (M), the number of attributes ($\widehat{F}$), and the average number of fuzzy sets in the partition ($\#F_{set}$) for the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL.*

	FIRST			MEDIAN			LAST		
Dataset	M	$\widehat{F}$	$\#F_{set}$	M	$\widehat{F}$	$\#F_{set}$	M	$\widehat{F}$	$\#F_{set}$
COV_2	20.8 ± 3.3	11.4 ± 0.5	4.4 ± 0.2	12.4 ± 2.4	9.4 ± 0.5	4.5 ± 0.3	7.6 ± 3.1	6.4 ± 2.7	4.0 ± 0.3
COV_7	7.0 ± 1.2	9.2 ± 2.4	3.7 ± 0.2	6.8 ± 1.1	7.2 ± 1.9	3.9 ± 0.3	5.8 ± 0.8	4.6 ± 0.9	3.8 ± 0.3
ECO	21.2 ± 3.3	11.6 ± 0.5	4.6 ± 0.2	10.2 ± 2.3	10.2 ± 0.9	4.5 ± 0.2	5.0 ± 0.0	3.6 ± 0.5	4.7 ± 0.3
EME	28.6 ± 4.7	14.2 ± 0.8	4.8 ± 0.3	14.4 ± 5.1	11.0 ± 2.3	4.7 ± 0.2	5.4 ± 0.9	2.8 ± 1.9	4.7 ± 0.3
HIG	14.0 ± 1.7	12.2 ± 2.1	3.7 ± 0.2	9.0 ± 1.7	9.4 ± 0.6	3.7 ± 0.1	6.4 ± 1.5	5.4 ± 0.6	3.7 ± 0.2
KDD_2	10.8 ± 1.8	9.8 ± 2.2	3.9 ± 0.3	7.2 ± 1.1	7.6 ± 1.7	3.9 ± 0.3	5.4 ± 0.5	4.2 ± 0.4	3.9 ± 0.3
POK	41.6 ± 3.1	5.0 ± 0.0	-	19.4 ± 3.0	5.0 ± 0.0	-	6.6 ± 1.3	4.2 ± 0.4	-
SUS	14.6 ± 2.7	12.4 ± 0.9	4.3 ± 0.2	9.0 ± 1.6	10.2 ± 1.3	4.2 ± 0.2	6.6 ± 2.5	4.8 ± 1.5	4.2 ± 0.2

Table 5.6: *Average run times (measured in seconds) and standard deviations for the distributed evolutionary optimization (DEO) phase and the overall algorithm (Tot).*

	Execution Time (s)	
Datasets	DEO	Tot
COV_2	6 245 ± 1 115	7 165 ± 1 191
COV_7	4 965 ± 718	5 147 ± 671
ECO	23 895 ± 6 449	24 836 ± 6 416
EME	27 189 ± 3 060	28 088 ± 3 047
HIG	53 749 ± 9 780	54 821 ± 9 805
KDD_2	13 470 ± 1 033	14 310 ± 1 033
POK	3 935 ± 422	3 964 ± 421
SUS	32 010 ± 6 697	32 611 ± 6 690

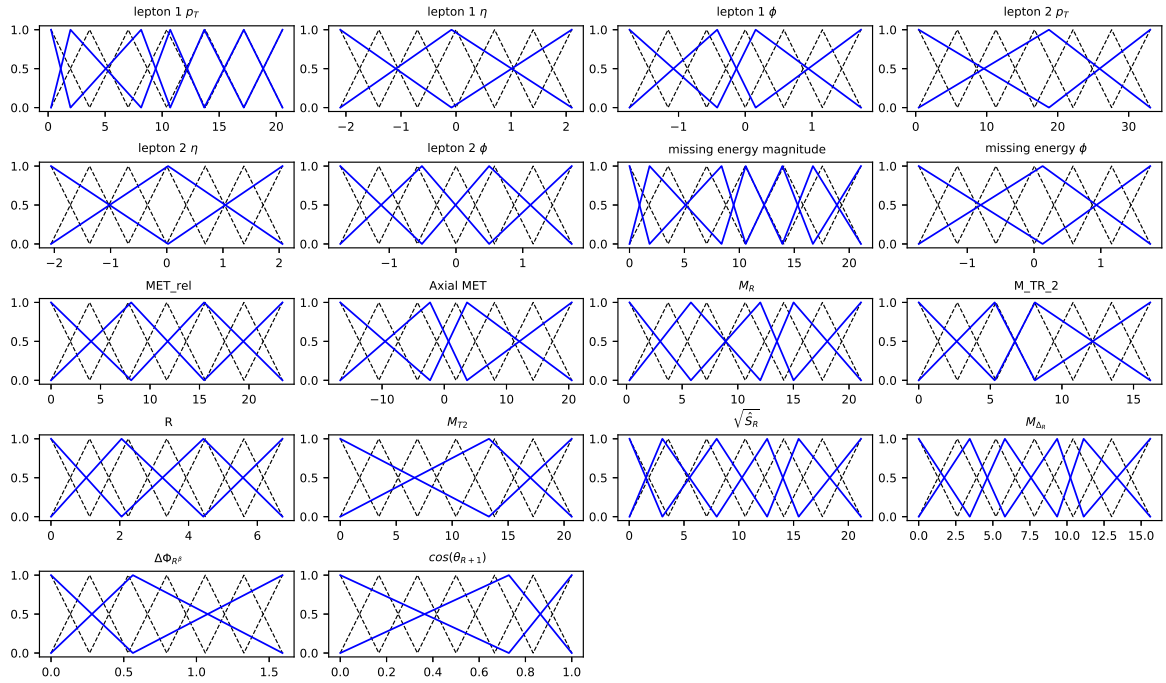


Figure 5.6: *The uniform fuzzy partitions (dashed line) and learnt fuzzy partitions (solid line) the MEDIAN solution obtained at the end of the evolutionary process are reported for each of the attributes of the SUSY dataset.*

IF missing energy magnitude **IS** 'very_low' **AND** M_R **IS** 'very_high' **AND** $\cos(\theta_{R+1})$ **IS** 'low' **THEN** Y **IS** TYPE_1
IF lepton 1 p_T **IS** 'very_low' **AND** missing energy magnitude **IS** 'very_low' **AND** M_{Δ_R} **IS** 'very_high' **THEN** Y **IS** TYPE_1
IF lepton 1 p_T **IS** 'low' **AND** lepton 1 η **IS** 'low' **AND** lepton 2 η **IS** 'low' **THEN** Y **IS** TYPE_2
IF Axial MET **IS** 'high' **AND** $\sqrt{\hat{S}_R}$ **IS** 'very_high' **AND** $cos(\theta_{R+1})$ **IS** 'low' **THEN** Y **IS** TYPE_2
IF lepton 1 p_T **IS** 'very_low' **AND** missing energy magnitude **IS** 'very_low' **AND** $cos(\theta_{R+1})$ **IS** 'medium' **THEN** Y **IS** TYPE_1
IF R **IS** 'high' **AND** $cos(\theta_{R+1})$ **IS** 'high' **THEN** Y **IS** TYPE_2
IF missing energy magnitude **IS** 'medium-low' **THEN** Y **IS** TYPE_2

Figure 5.7: *RB of the MEDIAN solution obtained on the first fold of SUSY. The RB, composed of 7 rules, and characterized by a TRL of 18, achieved a classification accuracy of* $\sim$ 78.776% *on the test set.*

5.4.2 Comparing DPAES-FDT-GL with DPAES-FDT and DPAES-RCS

This section provides an experimental comparison of the performances of DPAES-FDT-GL, DPAES-FDT and DPAES-RCS. Here, DPAES-RCS is the baseline MOEL scheme from which DPAES-FDT-GL has been derived, while DPAES-FDT is obtained from DPAES-RCS by activating the FDT learning but avoiding any kind of granularity learning. As the effectiveness of DPAES-RCS, compared to other state-of-the-art algorithms (such as DDTs and the Chi-FRBCS-BigData) has been reported before(Ferranti et al., 2017), it has been chosen as a benchmark.

Table 5.7 lists the average values (and the standard deviations) of the accuracy on the training (Acc_{Tra}) and test (Acc_{Tst}) sets for the FIRST, MEDIAN, and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and DPAES-RCS. Furtheremore, Table 5.8 reports the average number M of rule and the *TRL*.

According to the results reported in Table 5.7, the three algorithms provide comparable accuracies across the three solutions. Furthermore, considering the complexity indices from Table 5.8, the solutions generated by DPAES-FDT-GL and DPAES-FDT are generally more compact than those produced by DPAES-RCS. Nevertheless, it is worth noticing that DPAES-FDT-GL solutions are characterised, on average, by a lower *TRL* and fewer rules than those of DPAES-FDT.

In order to discover if any statistical difference exists among the accuracies and the complexities of the three algorithms so far discussed, the following approach has been followed: first, for each algorithm and on all datasets, a distribution consisting of the average accuracy values obtained on the test set, and a distribution consisting of the average complexity values have been defined. Then, non-parametric statistical tests are applied.

Table 5.7: *Average accuracies ± standard deviations achieved by the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and by DPAES-RCS.*

	DPAES-FDT-GL (FIRST)		**DPAES-FDT (FIRST)**		**DPAES-RCS (FIRST)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	75.843 ± 0.002	75.767 ± 0.003	**75.999 ± 0.002**	**75.988 ± 0.001**	75.753 ± 0.004	75.732 ± 0.003
COV_7	67.649 ± 0.012	67.618 ± 0.012	68.156 ± 0.006	68.232 ± 0.007	**72.383 ± 0.003**	**72.374 ± 0.003**
ECO	76.261 ± 0.005	76.266 ± 0.005	**78.498 ± 0.005**	**78.493 ± 0.005**	77.133 ± 0.004	77.115 ± 0.004
EME	81.225 ± 0.005	81.193 ± 0.005	**82.882 ± 0.004**	**82.855 ± 0.004**	80.600 ± 0.008	80.570 ± 0.008
HIG	**65.040 ± 0.003**	**65.035 ± 0.004**	65.013 ± 0.003	65.010 ± 0.003	65.008 ± 0.012	64.998 ± 0.012
KDD99_2	99.886 ± 0.008	99.886 ± 0.010	99.866 ± 0.000	99.865 ± 0.000	**99.948 ± 0.012**	**99.947 ± 0.012**
POK	**61.778 ± 0.011**	**61.806 ± 0.001**	60.535 ± 0.010	60.504 ± 0.011	60.233 ± 0.006	60.221 ± 0.006
SUS	**78.628 ± 0.004**	**78.608 ± 0.004**	77.968 ± 0.003	77.954 ± 0.003	78.123 ± 0.001	78.110 ± 0.001
	DPAES-FDT-GL (MEDIAN)		**DPAES-FDT (MEDIAN)**		**DPAES-RCS (MEDIAN)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	**75.378 ± 0.002**	75.320 ± 0.003	75.3754 ± 0.003	**75.335 ± 0.004**	74.968 ± 0.005	74.909 ± 0.005
COV_7	67.611 ± 0.011	67.582 ± 0.012	67.9249 ± 0.006	67.988 ± 0.006	**71.940 ± 0.004**	**71.924 ± 0.004**
ECO	74.069 ± 0.007	74.074 ± 0.007	**76.6335 ± 0.008**	**76.635 ± 0.007**	74.995 ± 0.011	74.984 ± 0.011
EME	78.997 ± 0.007	78.981 ± 0.007	**80.9819 ± 0.010**	**80.957 ± 0.010**	78.221 ± 0.010	78.201 ± 0.010
HIG	63.625 ± 0.007	63.610 ± 0.007	63.7108 ± 0.005	63.710 ± 0.005	**64.389 ± 0.008**	**64.370 ± 0.008**
KDD_2	99.883 ± 0.008	99.882 ± 0.009	99.8651 ± 0.000	99.864 ± 0.000	**99.933 ± 0.008**	**99.934 ± 0.008**
POK	56.061 ± 0.008	55.989 ± 0.008	55.4280 ± 0.010	55.360 ± 0.010	**58.423 ± 0.008**	**58.430 ± 0.009**
SUS	**78.362 ± 0.006**	**78.361 ± 0.006**	77.7293 ± 0.003	77.707 ± 0.003	77.658 ± 0.003	77.659 ± 0.003
	DPAES-FDT-GL (LAST)		**DPAES-FDT (LAST)**		**DPAES-RCS (LAST)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	66.153 ± 0.097	66.203 ± 0.096	70.907 ± 0.035	70.955 ± 0.035	**72.708 ± 0.007**	**72.681 ± 0.006**
COV_7	67.172 ± 0.007	67.157 ± 0.007	**67.243 ± 0.005**	**67.299 ± 0.006**	57.921 ± 0.106	57.907 ± 0.106
ECO	56.816 ± 0.089	56.801 ± 0.089	**59.864 ± 0.032**	**59.851 ± 0.032**	56.228 ± 0.078	56.244 ± 0.078
EME	**62.793 ± 0.039**	**62.793 ± 0.039**	62.233 ± 0.043	62.230 ± 0.043	61.407 ± 0.061	61.391 ± 0.061
HIG	58.718 ± 0.003	58.697 ± 0.003	58.524 ± 0.016	58.530 ± 0.015	**59.825 ± 0.017**	**59.849 ± 0.017**
KDD_2	94.423 ± 0.094	94.415 ± 0.094	94.351 ± 0.094	94.347 ± 0.094	**98.508 ± 0.017**	**98.514 ± 0.017**
POK	**49.870 ± 0.009**	**49.822 ± 0.009**	48.313 ± 0.012	48.331 ± 0.012	48.772 ± 0.031	48.749 ± 0.032
SUS	**72.525 ± 0.052**	**72.521 ± 0.052**	71.377 ± 0.051	71.414 ± 0.051	68.131 ± 0.083	68.128 ± 0.082

Table 5.8: *Average M and TRL ± standard deviations achieved by the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and by DPAES-RCS.*

	DPAES-FDT-GL (FIRST)		**DPAES-FDT (FIRST)**		**DPAES-RCS (FIRST)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	**20.8 ± 3.3**	107.4 ± 23.6	22.4 ± 3.8	113.8 ± 36.5	33.6 ± 8.4	**74.4 ± 23.0**
COV_7	7.0 ± 1.2	**11.4 ± 3.0**	**6.6 ± 1.9**	11.4 ± 4.4	36.2 ± 7.3	145.0 ± 37.0
ECO	**21.2 ± 3.3**	**101.2 ± 24.9**	28.0 ± 3.5	136.6 ± 21.4	54.0 ± 16.5	168.4 ± 79.6
EME	**28.6 ± 4.7**	**136.8 ± 23.0**	34.2 ± 4.0	165.2 ± 35.3	58.6 ± 5.7	187.4 ± 39.8
HIG	**14.0 ± 1.7**	**48.4 ± 23.2**	18.6 ± 3.4	77.2 ± 22.3	30.2 ± 8.2	125.2 ± 40.2
KDD99_2	**10.8 ± 1.8**	24.6 ± 6.5	11.4 ± 0.5	**21.8 ± 1.3**	21.8 ± 4.1	35.4 ± 8.0
POK	41.6 ± 3.1	90.2 ± 7.1	**39.0 ± 3.0**	**83.8 ± 6.5**	50.0 ± 4.6	113.2 ± 13.3
SUS	**14.6 ± 2.7**	**63.0 ± 17.1**	18.0 ± 6.0	73.0 ± 35.6	28.0 ± 8.6	80.4 ± 33.4
	DPAES-FDT-GL (MEDIAN)		**DPAES-FDT (MEDIAN)**		**DPAES-RCS (MEDIAN)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	12.4 ± 2.4	43.4 ± 11.8	**11.2 ± 0.8**	**37.2 ± 6.0**	21.7 ± 7.3	38.7 ± 17.3
COV_7	6.8 ± 1.0	8.8 ± 1.8	**6.2 ± 1.6**	**7.8 ± 2.5**	29.4 ± 6.8	84.2 ± 25.1
ECO	**10.2 ± 2.3**	**34.6 ± 10.3**	15.0 ± 2.3	51.0 ± 11.2	45.4 ± 17.3	117.7 ± 73.4
EME	**14.4 ± 5.1**	**50.2 ± 26.4**	15.4 ± 1.5	56.8 ± 13.2	48.1 ± 5.9	112.0 ± 27.2
HIG	**9.0 ± 1.7**	**22.0 ± 1.7**	10.6 ± 0.9	26.8 ± 6.0	25.8 ± 6.8	78.7 ± 28.6
KDD_2	**7.2 ± 1.1**	13.6 ± 2.5	7.4 ± 0.9	**13.4 ± 0.5**	13.2 ± 2.5	19.5 ± 4.3
POK	19.4 ± 3.0	37.6 ± 5.7	**17.2 ± 2.9**	**33.8 ± 6.3**	35.2 ± 6.3	68.1 ± 11.8
SUS	**9.0 ± 1.6**	28.2 ± 8.3	9.6 ± 2.8	**26.6 ± 12.3**	19.9 ± 7.7	45.6 ± 25.5
	DPAES-FDT-GL (LAST)		**DPAES-FDT (LAST)**		**DPAES-RCS (LAST)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	7.6 ± 3.1	12.2 ± 7.5	**5.2 ± 0.4**	**6.4 ± 1.7**	9.2 ± 2.6	10.0 ± 3.2
COV_7	**5.8 ± 0.8**	**5.8 ± 0.8**	5.8 ± 1.8	5.8 ± 1.8	28.0 ± 6.4	58.2 ± 19.9
ECO	**5.0 ± 0.0**	**5.0 ± 0.0**	6.4 ± 2.1	7.6 ± 4.2	35.2 ± 10.9	54.4 ± 24.2
EME	**5.4 ± 0.9**	**5.8 ± 1.8**	**5.4 ± 0.9**	6.2 ± 2.7	44.6 ± 4.6	75.2 ± 17.3
HIG	6.4 ± 1.5	6.2 ± 1.5	**5.6 ± 1.3**	**5.8 ± 1.3**	23.2 ± 7.2	48.6 ± 21.4
KDD_2	**5.4 ± 0.5**	**5.4 ± 0.5**	5.4 ± 0.9	5.6 ± 1.3	8.0 ± 1.4	8.2 ± 1.3
POK	6.6 ± 1.3	9.2 ± 3.3	**5.6 ± 1.3**	**6.0 ± 2.2**	25.4 ± 3.1	34.2 ± 8.4
SUS	**6.6 ± 2.5**	**7.4 ± 3.8**	7.0 ± 1.2	7.6 ± 1.9	15.0 ± 6.9	22.0 ± 14.0

The Friedman test allows ranking the distributions, while the Iman and Davenport test uncovers whether there exists a statistical difference. Notably, if it is the case that the Iman and Davenport p-value is lower than the level of significance α [4], then the null hypothesis can be rejected, and the claim that there exist statistical differences among the multiple distributions can be made. If not, no statistical difference exists.

If the null hypothesis is rejected, a post-hoc procedure, the Holm test, is applied, allowing the detection of statistical differences between the *control approach*, i.e. the one with the lowest Friedman rank, and the remaining approaches. Details on the aforementioned tests may be found in (García et al., 2009).

Table 5.9: *Results of the Friedman and of the Iman and Davenport tests on the accuracy computed on the test set.*

	Algorithm	*Friedman rank*	*Iman and Davenport p-value*	*Hypothesis*
FIRST	DPAES-FDT-GL	1.875	0.7145	Not Rejected
	DPAES-FDT	1.875		
	DPAES-RCS	2.25		
MEDIAN	DPAES-FDT	1.875	0.714	Not Rejected
	DPAES-RCS	2		
	DPAES-FDT-GL	2.25		
LAST	DPAES-FDT-GL	1.75	0.7145	Not Rejected
	DPAES-FDT	2.125		
	DPAES-RCS	2.125		

The results of the Friedman and of the Iman and Davenport tests on the accuracy values obtained over the test set are reported in Table 5.9. Here, according to the p-values, the null hypothesis can never be rejected (being the p-values always greater than 0.05). As such, all the three algorithms provide solutions that are statistically equivalent in terms of classification accuracy. Nevertheless, it is worth noticing that DPAES-FDT-GL and DPAES-FDT achieve the highest ranks for the FIRST solutions.

The results of the application of the Friedman and of the Iman and Davenport tests on the complexities are reported in Table 5.10; here, being the p-values always lower than 0.05, the null hypothesis associated with the Iman and Davenport test is rejected in all three cases. Performing a Holm post-hoc procedure, and considering DPAES-FDT-GL, DPAES-FDT and PDAES-FDT-GL as control approaches for the FIRST, MEDIAN and LAST solutions, respectively, it is observed that the DPAES-RCS solutions are always statistically more complex than those of the control algorithms (the results for the Holm post hoc procedure are reported in 5.11). Furthermore, the complexity of the solutions gen-

[4]Here it is assumed the standard threshold value $\alpha = 0.05$

Table 5.10: *Results of the Friedman and of the Iman and Davenport tests on the complexity.*

	Algorithm	*Friedman rank*	*Iman and Davenport p-value*	*Hypothesis*
FIRST	DPAES-FDT-GL	1.437	0.0139	Rejected
	DPAES-FDT	1.812		
	DPAES-RCS	2.75		
MEDIAN	DPAES-FDT	1.375	0.0013	Rejected
	DPAES-FDT-GL	1.75		
	DPAES-RCS	2.875		
LAST	DPAES-FDT-GL	1.562	0.0025	Rejected
	DPAES-FDT	1.562		
	DPAES-RCS	2.875		

erated by DPAES-FDT-GL and DPAES-FDT are always statistically equivalent. As such, both DPAES-FDT-GL and DPAES-FDT provides less complex results than DPAES-RCS, yet providing comparable accuracies.

As a last remark, in most of the cases, the complexities of the DPAES-FDT-GL solutions are lower than those generated by DPAES-FDT.

Table 5.11: *Results of the Holm post hoc procedures on the complexity for* $\alpha = 0.05$

	i	**algorithm**	z**-value**	p**-value**	**alpha**/i	**Hypothesis**
FIRST	2	DPAES-RCS	2.625	0.0086	0.025	Rejected
	1	DPAES-FDT	0.75	0.4532	0.05	Not Rejected
MEDIAN	2	DPAES-RCS	3	0.0027	0.025	Rejected
	1	DPAES-FDT-GL	0.75	0.4532	0.05	Not Rejected
LAST	2	DPAES-RCS	2.62	0.0086	0.025	Rejected
	1	DPAES-FDT	0	1	0.05	Not Rejected

In order to allow a visual inspection of the approximated Pareto fronts, a plot of the average values achieved by the three representative solutions, for all the datasets, on both the training and test sets is provided (Figure 5.8); in the plot, which depicts the classification rate/*TRL* plane, the solutions generated by DPAES-FDT-GL, DPAES-FDT and DPAES-RCS are reported as blue diamond, empty black circle and red plus markers, respectively.

In short, the usage of a distributed FDT, in place of a distributed version of the C4.5, allows the MOEL to generate simpler (less complex) FRBCs while leaving the accuracies unchanged. Besides, the activation of the granularity allows to further reduce the number of rules and the *TRL* of the generated classifiers, even though this contribution is not strong enough to be statistical detected.

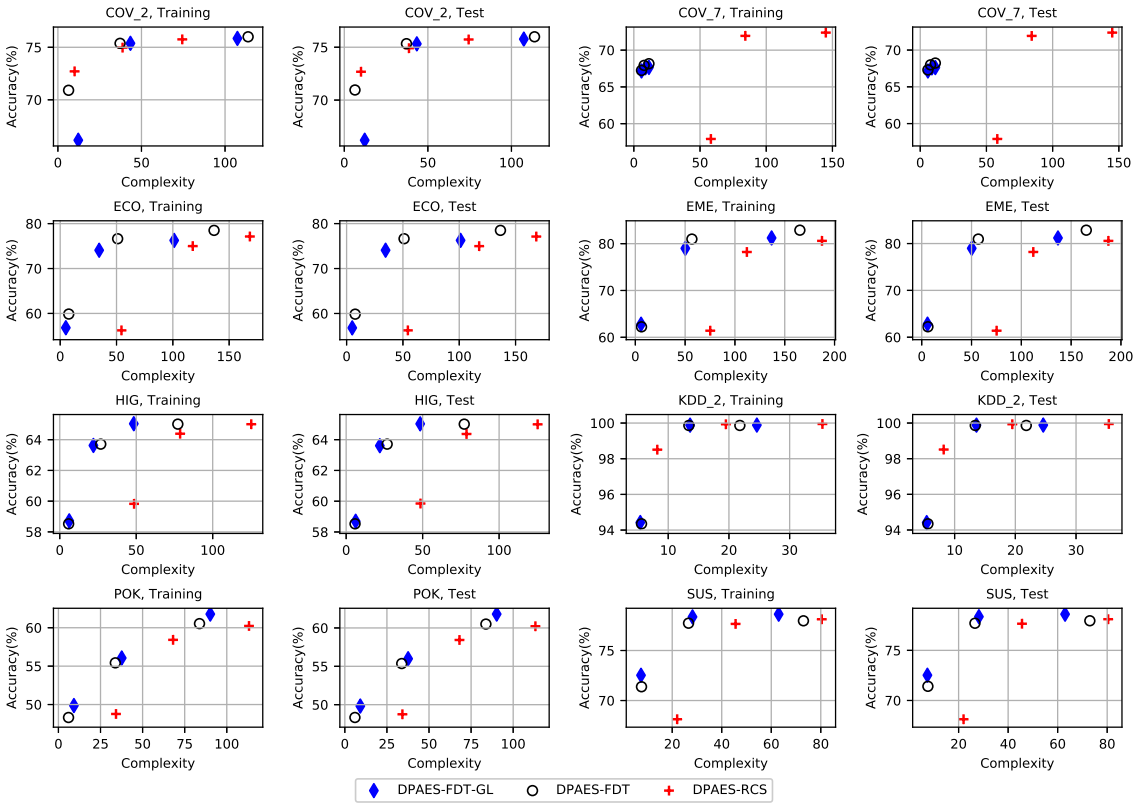


Figure 5.8: *Plots of the average accuracy on the training and test sets and average TRL of the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL (blue diamond markers), DPAES-FDT (empty black circle markers) and DPAES-RCS (red plus symbol markers).*

Summing up the results, the following considerations may be given in order to explain the behaviour of the proposed approach: first, as the FDT learning algorithm generates fuzzy decision trees directly from fuzzy partitions, the tree is inherently tuned to the fuzzy partitions. On the other hand, the C4.5 learning algorithm used in DPAES-RCS generates decision trees from crisp partitions. The fuzzy partitions are actually considered crisp for the execution of the learning algorithm, and only once the tree is learnt, the rules are extracted from the tree and the labels re-assigned to the original fuzzy sets. As such, the decision tree (differently from FDT) is not tuned to the final fuzzy partitions. Second, the granularity learning process allows reducing the number of fuzzy sets for each linguistic variable. The lower the number of fuzzy sets that describe each partition, the lower the number of combinations that can be obtained for generating classification rules. This result is achieved thanks to the synergy among the initial set of fuzzy rules extracted from the FDT, granularity learning, rule and condition selection, and fuzzy set parameter learning. Indeed, the membership function parameter learning allows adapting the fuzzy partitions to the dataset, also when using a low number of fuzzy sets for each linguistic attribute. Thus, the number of rules can decrease and the accuracy increase during the evolutionary process.

Lastly, a brief comparison of the the results of DPAES-FDT-GL with those of a distributed multi-way fuzzy decision tree (DMFDT) [5] learning algorithm (Segatori et al., 2018), and of a distributed fuzzy associative classifier for big data (DFAC-FFP) (Segatori et al., 2017a) is provided in Table 5.12.

Considering HIGGS, it is the DMFDT that achieves the highest accuracy; there, the lower complexity of both DPAES-FDT-GL and DFAC-FFP is equalised by a lower classification accuracy. Furthermore, while the accuracies of DPAES-FDT-GL and DFAC-FFP are comparable, the model complexities are different by about 2 order of magnitudes.

On KDD_2, the three algorithms achieve similar accuracy, but the complexity of DPAES-FDT-GL is one order of magnitude smaller than the other ones.

On Susy DMFDT achieves a classification accuracy of $\sim$ 79.6%; it is $\sim$ 1.1% higher of that achieved by DPAES-FDT-GL, yet it has been obtained with 805, 076 nodes and 758, 064 leaves, thus with a system of 4 orders of magnitude more complex than the one generated by DPAES-FDT-GL. Finally, it is worth noticing that DPAES-FDT-GL achieves better results than DFAC-FFP, with a complexity smaller by two orders of magnitude.

5.5 Summary

In this chapter, we have proposed DPAES-FDT-GL; it is a novel approach for generating sets of fuzzy rule-based classifiers with different optimal trade-offs between accuracy and interpretability from big data. The approach is built upon an extension of DPAES-RCS, which

[5] While DMFDT exploits the same FDT learning algorithm used to generate the initial set of fuzzy rules in DPAES-FDT-GL, it employs fuzzy partitions generated by a distributed fuzzy discretizer, leaves labelled with different classes and a weighted voting inference strategy.

Table 5.12: *Comparison of the average accuracies on the test set and average complexities for DPAES-FDT-GL, DMFDT and DFAC-FFP. Complexity is measured as average number (M) of rules and average TRL for DPAES-FDT-GL, average number of nodes and leaves for DMFDT, and average number (M) of rules for DFAC-FFP.*

	DPAES-FDT-GL			**DMFDT**			**DFAC-FFP**	
Dataset	Acc_{Tst}	M	TRL	Acc_{Tst}	$\#Leaves$	$\#Nodes$	Acc_{Tst}	M
HIG	65.035 ± 0.004	14.0	48.4	71.253 ± 0.029	920,942	972,779	66.005 ± 0.078	9,365
KDD_2	99.886 ± 0.010	10.8	24.6	99.986 ± 0.005	703	630	99.998 ± 0.001	890
SUS	78.608 ± 0.004	14.6	63.0	79.639 ± 0.016	758,064	805,076	78.267 ± 0.050	10,970

is a distributed multi-objective evolutionary algorithm recently proposed on the Apache Spark framework. Two main novelties characterise DPAES-FDT-GL; first, the initial set of candidate rules used in the multi-objective evolutionary learning is extracted from a fuzzy decision tree (FDT) rather than a crisp decision tree. Second, the granularity of each numerical attribute (that is, the number of fuzzy sets in the partition) can be learnt during the evolutionary process as well.

The approach has been evaluated and compared with DPAES-RCS, by performing a set of experiments on 8 big datasets. First, the accuracy achieved by the fuzzy rule-based classifiers generated by DPAES-FDT-GL is statistically equivalent to the one obtained by the classifiers generated by DPAES-RCS; nevertheless, the FRBCs generated by DPAES-FDT-GL are characterised by the lowest number of rules, conditions, and fuzzy sets. As such, DPAES-FDT-GL represents an important step forward in getting more interpretable fuzzy classifiers in the setting of big data. Since there exists a number of real applications that require not only high accuracy but also high interpretability, DPAES-FDT-GL can be a very interesting and promising approach for such applications.

Then, in order to untie the contribution of the FDT from that of the adopted granularity learning, we also performed a comparison with DPAES-FDT, a version of DPAES-FDT-GL that adopts the FDT for generating the initial rule set, but no granularity learning during the evolutionary process. The results suggest that, when extracting the initial set of rules from an FDT, the resulting models are always statistically less complex. Moreover, even though no statistical differences between the complexities of the FRBCs generated by DPAES-FDT-GL and DPAES-FDT can be found, the activation of the granularity learning allows reducing, in most of the cases, the number of rules and the *TRL* of the generated classifiers.

CHAPTER

SIX

FUZZY RULE BASED CLASSIFIERS: AN APPLICATION TO CNV-BASED TUMOUR CLASSIFICATION

> Words strain, Crack and sometimes break, under the burden, Under the tension, slip, slide, perish, Decay with imprecision, will not stay in place, Will not stay still.
>
> — T.S. Eliot.

This chapter contains material from the following publications:

- Ricatto, M., Barsacchi, M., & Bechini A. (2018). Interpretable CNV-based Tumour Classification using Fuzzy Rule Based Classifiers. In ACM SAC 2018. https://doi.org/10.1145/3167132.3167135.

In this chapter we move towards applied fuzzy systems, proposing a novel pipeline to support tumour type classification and rule extraction based on somatic CNV data. The pipeline builds an interpretable Fuzzy Rule Based Classifier (FRBC), on which inference can be made.

The chapter is organised as follows: Section 6.1 provides an introduction to the problem; then Section 6.2 describes the proposed pipeline, as well as the classification algorithm. Section 6.3 provides a case study and benchmarks the approach. Finally, Section 6.4 concludes the chapter, summarising the results.

6.1 Copy Number Variations, Cancer, and Expert Systems

The once posited 99.9% genetic identicalness of all humans has been challenged by a series of recent discoveries. We now know that the DNA sequence of the genome undergoes

continuous variations, and this is the process that endows organisms with the ability to evolve and adapt. Variations in the human genome encompass a wide range of sizes, from the single base pair to entire chromosomal events (Alkan et al., 2011). Located on the larger end of the spectrum, structural variations (SVs) are generally defined as genomic regions of DNA approximately 1Kb (1000 bases) or greater in size and can include insertions, deletions and inversions (Feuk et al., 2006). Even though SVs play a major role in a wide class of illnesses, their presence is not necessarily associated with diseases or particular phenotype.

The family of DNA structural variants is numerous and multifarious; among them, copy number variations (CNVs) can be considered one of the most common classes: according to recent studies, up the 10% of the human genome is currently believed to be altered by copy number variations (Zarrei et al., 2015). Certainly, CNV can be held responsible for the majority of base pair variations between genomes. CNVs were initially defined as SVs with copy number change, affecting regions with size greater than 1 kilobase (Kb) and typically less than five megabases (Mb) (Freeman et al., 2006). As of today, the lower end of the range has been pushed down to include variants larger than 50 bp (Zarrei et al., 2015). The widespread presence of CNVs seems to suggest that they play a relevant contribution in phenotypic variation (Zarrei et al., 2015); CNV and phenotype were first shown to be associated in 1936, when a duplication of the Bar gene in *Drosophila melanogaster* was related to the Bar eye phenotype (Nowakowska, 2017). Furthermore, thanks to several recent studies, CNVs are known to play an important role in human diseases (Girirajan et al., 2011), altering the diploid status of particular loci in the genome.

The relationship between copy number variations and cancer has been recently investigated (Shlien and Malkin, 2009, 2010), and it has been suggested that CNVs could play a non-trivial role in cancer evolution. It is only recently that researchers started exploiting somatic CNVs. For example, a recent work has shown how CNVs can be used as a means for cancer prognostic (Poniah et al., 2017). Moreover, it has been proposed to use CNV data for the classification of tumours, and the first attempts (Zhang et al., 2016; Li et al., 2014) look promising. However, this particular research field it is still in its infancy, and the relationship between CNVs and different types of cancer remains unclear: this work is aimed at providing a further contribution.

Data mining (and machine learning) is a thriving field; among the wide variety of approaches, fuzzy rule-based systems (FRBS) offer highly interpretable results (Baldwin and Xie, 2005), yet providing nearly-state-of-the-art accuracies. The interpretability of the system increases the confidence in its outputs, and this is of utmost importance in critical fields, such as that of medical applications (a brief discussion of the definition of interpretability is provided in Chapter 1). Furthermore, fuzzy systems are naturally endowed with the ability to manage noisy and uncertain data, and sequencing data can be regarded as uncertain proxy measures for the state of the underlying biological system.

This work focuses on the definition of an approach for tumour classification from somatic CNV data. We define a powerful classification pipeline able to provide high classification accuracies and to produce a small set of understandable simple fuzzy logic rules

(i.e. IF ... THEN... rules), based on triangular shaped fuzzy sets. Rules can provide insights into the roles of specific genes in specific classes of tumours and can be of valuable help in understanding genes interactions.

6.2 Methods description

The proposed methodology is described in this section; the approach is arranged into three phases: in the first phase, CNV data are filtered, preprocessed and rearranged (Section 6.2.1) outputting a representation amenable for the following classification step; subsequently, the training set is discretized using a Fuzzy Minimum Description Length Principle discretizer (Fuzzy-MDL), and the dataset is then used to learn a Distributed Fuzzy Decision Tree (DFDT), thus training a model (Section 6.2.2). Finally, given the discretization and the learnt model, a Fuzzy Rule Based Classifier (FRBC) is extracted (Section 6.2.3) and enables inference on the system.

6.2.1 Data extraction and preprocessing

In this section, a description of the data extraction and pre-processing pipeline is given. The goal of the whole procedure is to attain a database structured so that every row is a specific patient, every feature is a gene affected by a CNV and the value of the feature is the segment mean of that CNV. A scheme of the whole process is given in Figure 6.1.

Data extraction

The first step of the pipeline involves the retrieval of a set of preprocessed (somatic) CNV data; the description of the CNV extraction process from raw data is beyond the scope of this work and can be found elsewhere (Kendall and Krasnitz, 2014; Zare et al., 2017). CNVs can be detected and CNV data can be generated with different approaches; initially conventional karyotyping was used, while nowadays the two primary technologies are high-throughput next generation sequencing (NGS) and comparative genomic hybridization (aCGH) (Nowakowska, 2017). We aim at proposing an approach that is independent with respect to the method used in CNV detection. Nonetheless, samples must be normalized prior to the usage, in order to avoid biasing the machine learning algorithm. Furthermore, particular care must be taken when combining samples generated with different approaches.

In the following, we refer to data organised as in TCGA level 3[1] data (Chang et al., 2013); the data are structured such as that each row pertains to a specific CNV, and contains information about the patient id, the chromosome affected, the start and end coordinates,

[1] A thorough description of the relationship between TCGA data types and data levels can be found here `https://cancergenome.nih.gov/abouttcga/aboutdata/datalevelstypes`.

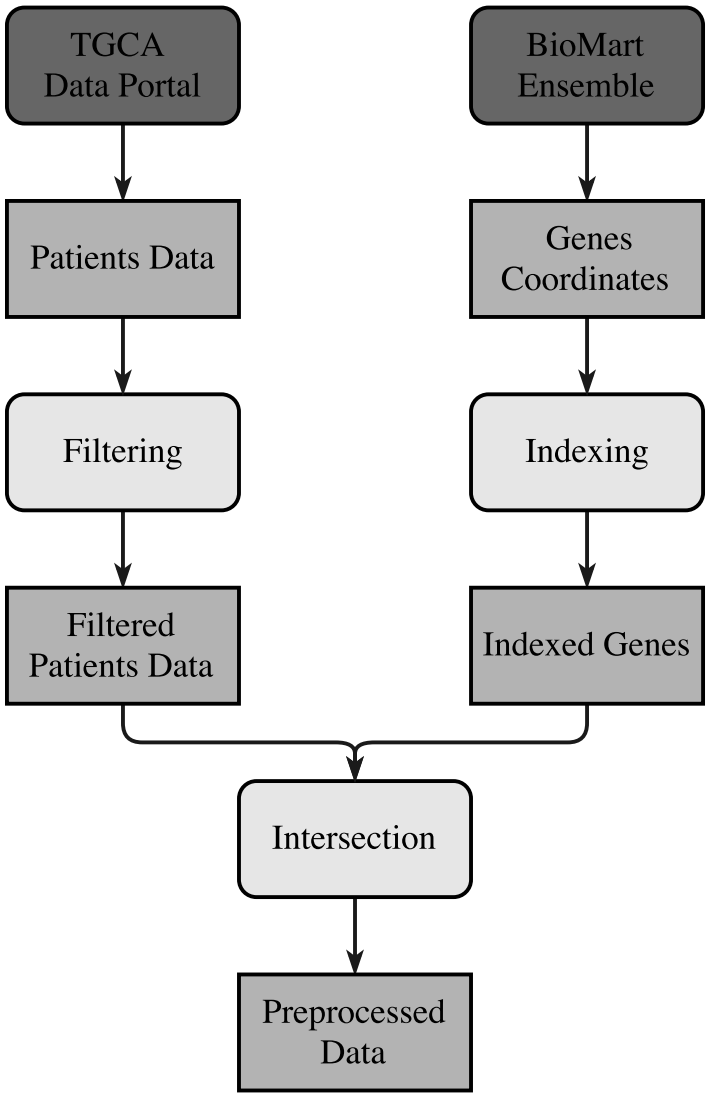


Figure 6.1: *Scheme of the proposed data extraction and pre-processing protocol. Data sources (TGCA data portal and Biomart Ensemble) are pictured in dark grey, intermediate or final data products are coloured in grey while operations on the data (such as filtering, indexing and intersection) are reported in light gray.*

the number of probes and the segment mean [2].

The genes coordinates are retrieved by means of the BioMart tool from the Ensembl genome browser (Yates et al., 2016); the selected reference genome is the GRCh37/hg19, that is, the same used as a reference for TCGA patients data. In the following it is assumed that genes data are structured in a tabular format; each row should refer to a gene and should be defined by four attributes: chromosome, start index, end index, gene name.

Data preparation

The preprocessing phase consists of a filtering step, in which both CNV and genes data are filtered, and of an intersection step, in which CNVs data and genes data are intersected. With regards to the filtering step, the following three criteria have been used:

Segment Mean value First, aiming at extracting a set of high confidence CNVs, filtering criteria on the segment mean value has been defined; thresholds of 0.2 for amplifications and -0.2 for deletions have been used. These thresholds have been proposed by examining the distribution of segment mean values from both tumoural and normal samples, and their effectiveness has been proved before (Laddha et al., 2014).

CNV Length The second criterion relates to the CNV length. Only CNVs shorter than 1 megabase (Mb) have been used here, while the longer ones have been filtering out. We used a slightly stricter interval, w.r.t the 50b - 3Mb proposed in the CNV map (Zarrei et al., 2015). This step is necessary as segments that cover large regions, e.g. a whole chromosome (aneuploidy), may be found; in such cases, the whole region will be covered with a unique value of segment mean, so that all the genes in the region have the same copy number value. Then, genes will be indistinguishable in the following operations.

Chromosome Another filtering criterion is based on the selection of the chromosome to investigate. In fact, data analysis is often restricted only to autosomes, because of several factors; first, CNV calling in X chromosome has been proved problematic due to the presence of pseudoautosomal regions (Conrad et al., 2010); moreover, there is a severe lack of available data for sex chromosomes. Finally, limiting the analysis to autosomes makes this work comparable to the majority of the researchers' efforts in this field (Qiu et al., 2017).

In the second phase, an intersection between filtered CNVs data and genes data is generated; for each patient, its segmented data are compared with the genes extracted from BioMart. If the start/end coordinates of the segmented data overlap with the coordinates of a gene (either both coordinates are inside the genomic interval containing the gene, both are outside the interval, or one coordinate is inside and the other is outside), the CNV segment mean of that segment data will be associated to that gene. This operation has

[2] The *segment mean* is defined as $\log_2\left(\frac{CN}{2}\right)$, being CN the *copy number*.

been carried out using `bedtools` (Quinlan and Hall, 2010). The intersection process is portrayed in Figure 6.2

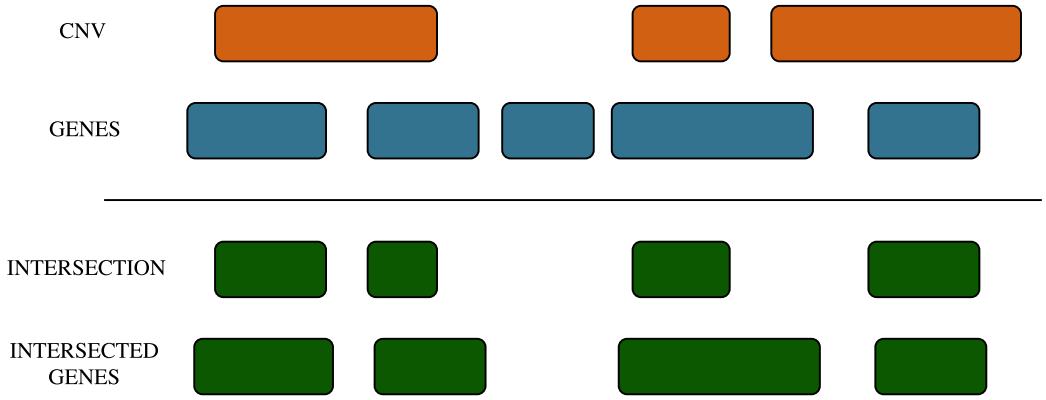


Figure 6.2: *Scheme of the intersection process. The start/end coordinates of the segmented data (first row) are compared with the coordinates of the genes (second row). All the genes for which overlap occurs, either with both coordinates inside the genomic interval containing the gene, with both outside the interval, or with one coordinate inside and the other outside, are selected (last row).*

As a side-product, each intersection produces a value quantifying the amount of "overlap" between the CNV and the gene region; these values can be used to quantify the relative importance of each feature.
Besides, if more than one intersection has been found for a given gene, the segment mean value will be an average of all the segment mean values for the intersecting CNVs; this strategy has been proposed before and has been found appropriate (Qiu et al., 2017). Finally, intersected data has been reorganised in `libsvm` format:
`<label>` $\texttt{<gene}_1\texttt{>:<segMean}_1\texttt{>}$ ... $\texttt{<gene}_N\texttt{>:<segMean}_N\texttt{>}$.

6.2.2 Fuzzy Discretization and Fuzzy (multi-way) Decision Tree

In this section, a brief description of both the discretization approach and of the selected classification algorithm are provided.

The Distributed Fuzzy Decision Tree (DFTF henceforth) algorithm, as proposed in (Segatori et al., 2017c), has proved to be very effective, and has been selected due to its peculiarities; particularly, it is a suitable starting point for generating an FRBS, as also observed in Chapter 5. The workflow of the proposed DFDT learning process consists of the two following main steps:

1. *Fuzzy Partitioning*: a strong fuzzy partition is generated on each continuous attribute by using a novel discretizer based on fuzzy entropy (Al-sharhan et al., 2001); features for which no discretization occurs will be discarded (Section 3.3.1 provides a complete description of the approach).

2. *FDT Learning*: a (multi-way) Fuzzy Decision Tree (FDT) is induced from the training data by using a multi-way splitting mechanism based on the concept of fuzzy information gain. The approach differs from the one described in Chapter 3 and 4 in that it follows a multi-way splitting criterion instead of a binary one.

In the following, the essential notation will be recalled. Let C_m be a class from a set of possible classes $C = \{C_1, ..., C_J\}$. A training set **TR** can be described as a set of tuples $\mathbf{TR} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, with each instance $\mathbf{x} = [x_1, ..., x_F]$, described by a set of features $\mathbf{X} = \{X_1, ..., X_F\}$ and each label y defined on the set of possible classes; here we used F to represent the number of features, and J to denote the number of classes.

The Fuzzy Partitioning step takes care of producing, for each continuous feature X_f from the set of features $\mathbf{X}$, a fuzzy partition P_f on the universe of discourse U_f. A partition P_f, with T_f fuzzy sets on U_f, can be described as $P_f = A_{f,1}, ..., A_{f,T_F}$. Details about fuzzy partitions are provided in Chapter 2.2.
The strong triangular fuzzy partition is induced by recursive partitioning, using a "Fuzzy Partitioning based on Fuzzy Entropy" criteria. The stopping condition is similar to the Entropy Minimization method proposed by Fayyad and Irani (Fayyad and Irani, 1993b) and the process is repeated for each generated subset until a stopping condition is met. We recall that a complete description of the algorithm, as well as its detailed mathematical formulation, is given in Chapter 3.3.1. Interestingly enough, the partitioning procedure induces a simultaneous *feature selection*, as the features for which no discretization has been performed will be discarded.

The Fuzzy Decision Tree is built by means of a recursive node-splitting approach; each split is chosen such as to maximise a given criterion, in this case the Fuzzy Information Gain (Zeinalkhani and Eftekhari, 2014). Each split selects a feature f, and induce T_f child nodes, where each node corresponds to a linguistic value of the fuzzy partitioning P_f. Each feature can be selected only once on a given path from the root to the leaf. The procedure is iterated until a stopping condition is met (Segatori et al., 2017c)

6.2.3 Building a Fuzzy-Rule-Based-Classifier

Here, the notion of fuzzy rule-based system is briefly recalled; a thorough description is provided in Chapter 2. We then proceed by extending the definitions where necessary.

A fuzzy rule-based system (or a classifier, in this case) consists of two main components: 1) the fuzzy inference system, implementing the fuzzy reasoning process, and 2) the fuzzy knowledge base (KB) which formalises the knowledge about the problem; the knowledge base is, in turn, composed of a rule base (RB) containing the set of rules, and the data base (DB) containing the parameters of the linguistic partitions (Alcalá et al., 2007).

In the following, the DB is directly extracted from the discretizer described in Section 6.2.2; the RB is instead extracted from the learnt fuzzy decision tree, by mapping each path from the root node to a leaf into a Mamdani type rule (Mamdani and Assilian, 1975).

In both section 2.3 and 5.2, simple fuzzy rules with a single consequence have been presented. Conversely, this approach uses a set of rules with a certainty degree for all the classes in the consequent (Cordón et al., 1999); the rules have the following structure:

$$\begin{aligned} &\textbf{If } X_1 \textbf{ is } A_{1,k_{m,1}} \textbf{ and } \ldots \textbf{ and} \\ &\quad X_F \textbf{ is } A_{F,k_{m,F}} \textbf{ then } \left(r_1^m, \ldots, r_J^m\right) \end{aligned} \tag{6.1}$$

where $k_{m,f}$ uniquely identify the fuzzy set selected for the feature f in the m-th rule, and r_j^m is the soundness degree for the m-th rule to output the class $C_j \in C$, given the pattern defined by the antecedent. The soundness degrees for the m-th rule, $r_j^m,\ j = 1, \ldots, J$, being J the number of classes, depends on the fraction of examples in the m-th leaf of the tree with class C_{j_m}; we can thus compute r_j^m as:

$$r_j^m = \frac{|G_{C_j}^{(m)}|}{|G^{(m)}|},\ j = 1, \ldots, J \tag{6.2}$$

where $|G^{(m)}|$ is the total membership of the examples in the m-th leaf, and $|G_{C_j}^{(m)}|$ is the total membership of the examples in the m-th leaf restricted to the class C_j. As such, the $r_j^m,\ j = 1, \ldots, J$ are real numbers in the interval $[0, 1]$ and sum up to one. Moreover, if only examples from a single class h are found in the m-th leaf, such that:

$$r_h^m = 1,\ r_j^m = 0,\ j \neq h,\ j = 1, \ldots, J, \tag{6.3}$$

the rule reduces to a classic Madmani rule without certainty degree, as described in Section 2.3:

$$\begin{aligned} &\textbf{If } X_1 \textbf{ is } A_{1,k_{m,1}} \textbf{ and } \ldots \textbf{ and} \\ &\quad X_F \textbf{ is } A_{F,k_{m,F}} \textbf{ then } Y \textbf{ is } C_h. \end{aligned} \tag{6.4}$$

In the inference phase, given an input pattern $\hat{\mathbf{x}} = \{\hat{x}_1, \ldots, \hat{x}_F\}$, the m-th rule activates with a strength (*matching degree*) $R^m(\hat{\mathbf{x}})$ defined as:

$$R^m(\hat{\mathbf{x}}) = T(\mu_{A_{1,k_{m,1}}}(\hat{x}_1), \ldots, \mu_{A_{F,k_{m,F}}}(\hat{x}_F)) \tag{6.5}$$

where $T(\cdot)$ is the t-norm operator and $\mu_{A_{f,k_{m,1}}}(\hat{x}_f)$ is the membership degree of $\hat{x}_f$ in the fuzzy set uniquely identified for the feature f in the m-th rule. In the following experimentation we used the product as t-norm operator.
Then, the association degree of the m-th rule with the class C_j, b_j^m, is computed as:

$$b_j^m = h(R^m(\hat{\mathbf{x}}), r_j^m) \tag{6.6}$$

being $h()$ a combination operator. We used the product operator as h operator.
Given the association degrees for all the rules in the rule base, we adopted a *weighted vote* approach to combine the association degrees; first each association degree is weighted by

a function g: $B_j^m = g(b_j^m)$. In our work we used the identity function as weight function: $g(x) = x$.
Then, for each class C_j, the weighted votes for all the classes are aggregated:

$$V_j = \sum_m B_j^m \tag{6.7}$$

Eventually, the class label l is selected by applying a decision function D to all the classes: $D(V_1, ..., V_J)$; we selected the simplest decision function, the max function:

$$C_l = \arg\max_{j=1,...,J} V_j. \tag{6.8}$$

6.3 A potential application to Kidney Cancer

In order to show a potential application of our method, and to measure its effectiveness, we propose a case study in which the approach is applied to the classification of kidney cancer.

Kidney cancer is known to affect nearly 270,000 patients annually worldwide and is held responsible for over 115,000 deaths each year (Linehan, 2012). As true for other tumours, kidney cancer (or Renal Cancer) is not a single disease; indeed, it comprises a number of different cancers that occur in the kidney, each with a different histology and clinical course; importantly, each cancer type responds differently to therapy and is caused by mutations in different genes. As such, the ability to classify different tumour types on the base of their CNV footprints could provide clinicians with a powerful instrumental in deciding the best therapy.

We began collecting from TCGA (Chang et al., 2013) the data of about 890 patients affected by one among Kidney Renal Clear Cell Carcinoma (KIRC), Kidney Renal Papillary Cell Carcinoma (KIRP) and Kidney Chromophobe Renal Cell Carcinoma (KICH), which are all subtypes of Renal Cell Carcinoma. The instance label (tumour type) has been assigned by the TCGA sources after histological examination of the affected tissue. The tumour types, as well as the number of instances for each tumour are reported in Table 6.1.

Table 6.1: *Summary of the data collected from TCGA.*

Type of Carcinoma	Acronym	# Samples
Chromophobe Renal Cell Carcinoma	KICH	66
Renal Papillary Cell Carcinoma	KIRP	290
Renal Clear Cell Carcinoma	KIRC	532

It is worth mentioning that TCGA provides, for each tumoural tissue sample, a sample from the adjacent normal tissue; in order to account for these information, two datasets

are generated: the first (dubbed *Not Normal*) using only the tumoural samples, the second (dubbed *All Samples*) combining, for each patient, the normal and tumoural samples.
When the proposed pipeline is applied, it is first observed that total number genes affected by a CNV is 19349, even if a high fraction of them might be unrelated to the specific tumoural event; the number of affected genes for every patient varies between 90 and 779 [3]. As such, the data matrix is a sparse one.

The data have been preprocessed according to the pipeline described in Section 6.2.1; the FRBC algorithm used here has been compared to a random forest classifier (Breiman, 2001). The average accuracies and the standard deviation obtained on a 10-fold cross-validation are shown in Table 6.2. Here, the results appear fairly stable across different folds. Interestingly, the FRBC extracted from the DFDT produces higher mean accuracy and lower standard deviation than a RF algorithm; indeed, it does so by using a small set of rules, being more interpretable than an RF. Interestingly, the number of fuzzy rules of the FBRC is even lower than the number of trees composing the RF.

Table 6.2: *Average accuracies and standard deviations obtained by RF and FRBC on the two datasets, using a 10-fold cross-validation. Not Normal refers to the usage of tumoural sample only, while in All Samples the data from adjacent tissues is used as additional information.*

	Not Normal		**All Samples**	
Algorithm	μ	σ	μ	σ
RF	89.57%	3.42%	91.57%	2.75%
FRBC	**92.41**%	2.87%	**92.92**%	2.29%

Interestingly enough, if the information provided by the normal and tumoural samples is naively combined, the classification accuracy increases and the standard deviation decreases; we suggest three possible explanation: 1) adjacent tissues can be affected by the same similar CNVs of the tumoral tissue; combining the data could thus causing an increase of the signal to noise ratio (SNR). 2) Some authors suggested that normal tissue may not be normal as believed (Russo, 2006), and thus, again, this gives rise to an increased SNR. Finally, 3) from this kind of data, it can not be known which CNVs have been developed during lifetime and which are inherited from the family tree; thus, the algorithm can automatically use these data to spot out if all of the CNVs that patients have in common are linked to the kidney disease or not.

Beyond the sheer classification accuracy, the proposed approach produces a set of highly interpretable rules. The average number of rules is ~ 50. As an example, the first

[3] It should be noted that, due to the filtering step in the pipeline, this number does not consider the effects of aneuploidies, or CNV larger than 1Mb

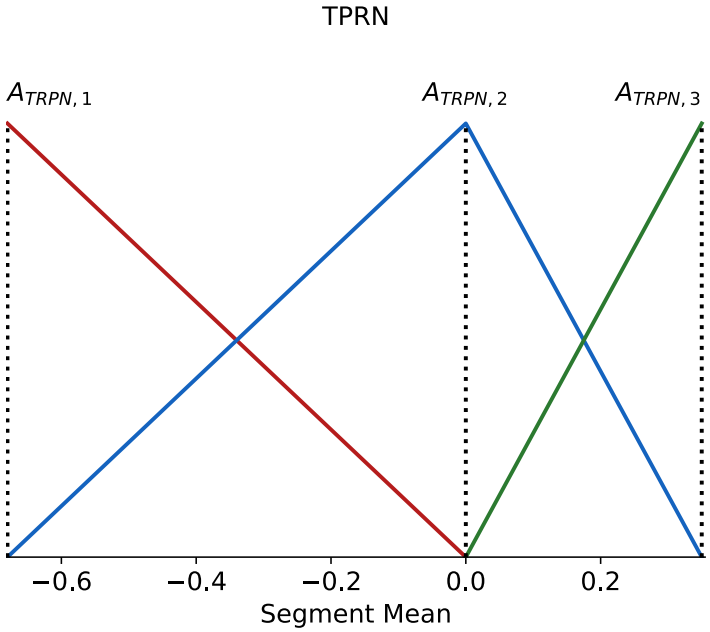


Figure 6.3: *The outcome of the fuzzy partitioning process on the TRPN gene. The three fuzzy sets $A_{TRPN,1}$, $A_{TRPN,2}$, $A_{TRPN,2}$ can be associated with decreased, normal and increased copy number respectively.*

three mined rules are reported in the box below:

```
R1: IF TPRN IS 1 AND IF ATXN7L2 IS 1 AND
IF TBC1D3C IS 1 THEN 3
R2: IF TPRN IS 2 AND IF TBC1D3B IS 1 AND
IF ATXN7L2 IS 3 AND IF TBC1D3C IS 1 THEN 1
R3: IF TPRN IS 3 AND IF ATXN7L2 IS 3 AND
IF TBC1D3C IS 1 THEN 3
```

As described in Section 6.2.2, each rule is made up of a set of antecedent conditions, i.e. a set of genes with their associated segment mean values, and a set of consequent values, i.e. the tumour classes, in this case; if the confidence value for the dominant class is 1, only the dominant class is reported. An example of fuzzy partitioning for the TRPN gene is shown in Figure 6.3.

The automatic discretization provides interpretable partitions, that can be mapped to linguistic values such as deleted and amplified gene.

6.4 Summary

In this chapter, a novel pipeline for tumour type discrimination from somatic CNV data is presented and thoroughly described. Inasmuch as CNVs have been proved to be related to

tumours, the study of CNVs and their relationship with cancer started maturing quickly; notwithstanding, most of the research still revolve around single nucleotide variants (SNVs) for tumour characterisation.

It is widely known that classifying tumour from CNV data is a challenging problem, notably due to the high number of affected genes and to the low number of samples. In the machine learning parlance, it is a member of the set of so-called "*Large p Small N*" problems. , To make matters worse, the sparsity of the data matrix further raises the complexity of the classification task. Here, we tried tackling the problem with a simple pipeline for analysing CNV data, classifying tumours and extracting a compact set of understandable rules. First, we generate a set of affected genes, by filtering CNV data and intersecting CNV coordinates with the coordinates of the genes. The feature sparsity has been handled by using an automated fuzzy partitioner that outputs interpretable partitions and performs a concurrent feature selection. The fuzzy decision tree produces a model from which a set of rules can be extracted, thus building a Fuzzy Rule Based Classifier (FRBC). The proposed pipeline is simple to implement and adaptable to a wide range of NGS data, yet it has demonstrated to be extremely effective.

The approach is benchmarked on a set of tumoural samples from TCGA, showing its effectiveness; the approach classifies the tumours with an accuracy of $\sim$ 93%, yet using a compact set of $\sim$ 50 rules. We believe that fuzzy techniques can be very useful in addressing cancer classification problems from CNV data. The set of understandable rules extracted from the system can guide clinicians in selecting the best therapy, and help researchers in understanding the mechanisms beyond the specific tumoural event.

Despite the good results illustrated here, several questions remain unanswered; first, the positive effect of normal samples on the classification accuracy must be clearly explained. Moreover, further discriminative power could be achieved considering in which position inside the gene a CNV occurs (e.g. intronic or exonic); its impact must be carefully addressed.

A further point that deserves an investigation is whether the set of rules can be shrunk and mapped back to gene circuits, to better understand the mechanisms underpinning these diseases.

CHAPTER

SEVEN

DISCUSSION

More than fifty years after its introduction, fuzzy sets theory is still thriving and continues to play a relevant role in a wide number of scientific applications. Nevertheless, while the enrichments that fuzzy logic and set theory can provide are manifold, as thoroughly discussed in Chapter 1, the recognition of fuzzy set and logic inside the machine learning community remains somewhat limited.

In this thesis, we presented several approaches aimed at improving machine learning techniques using tools borrowed from fuzzy set theory and logic. Particularly, we tried to focus more on the machine learning perspective, thus inviting machine learning researcher to appreciate the modelling strengths of fuzzy set theory.

We begin presenting **FDT-Boost** in Chapter 3, a boosting approach shaped according to the SAMME-Adaboost scheme, which leverages fuzzy binary decision trees as base classifiers. Such trees are kept compact by constraining their depth, without a degradation of the classification accuracy. The experimental evaluation of FDT-Boost has been carried out using sixteen classification benchmarks. Comparing our approach with FURIA, one of the most popular fuzzy classifiers, with a fuzzy binary decision tree, and with a fuzzy multi-way decision tree, we show that FDT-Boost is accurate, getting to results that are statistically better than those achieved by the other approaches. Moreover, compared to a crisp SAMME-AdaBoost implementation, FDT-Boost shows equivalent performances, but the relative produced models are significantly less complex.

Then, in Chapter 4, we present a distributed fuzzy random forest **DFRF**, that leverages the Apache Spark framework, to generate an efficient and effective classifier for big data. The approach is built upon a distributed fuzzy discretizer for big data that provides strong fuzzy partitions for each continuous attribute; then, an ensemble of distributed fuzzy decision trees is built, employing the fuzzy information gain as splitting criterion. By performing a set of experiments on eight big datasets, the approach has been thoroughly evaluated.

In Chapter 5, we propose a novel approach for generating, out of big data, a set of fuzzy rule-based classifiers characterised by different optimal trade-offs between accuracy and

interpretability. The approach, dubbed **DPAES-FDT-GL**, extends a state-of-the-art distributed multi-objective evolutionary learning scheme, implemented in the Apache Spark environment. In particular, we exploit a recently proposed distributed fuzzy decision tree learning approach for generating an initial rule base that serves as input to the evolutionary process. Furthermore, we integrate the evolutionary learning scheme with an ad-hoc strategy for the granularity learning of the fuzzy partitions, along with the optimisation of both the rule base and the fuzzy set parameters. Experimental investigations show that the proposed approach is able to generate fuzzy rule-based classifiers that are significantly less complex than the ones generated by the original multi-objective evolutionary learning scheme, while keeping the same accuracy levels.

Lastly, we move into a real case scenario, showing how fuzzy systems could be employed in helping medical decision making; in Chapter 6 we propose a novel pipeline to support tumour type classification and rule extraction based on somatic CNV data. The pipeline outputs an interpretable Fuzzy Rule-Based Classifier (FRBC), on which inference can be made. The pipeline benchmarking is performed over a set of samples of kidney cancer from TCGA. The results show the potential application of the approach: the method is able to classify between three kidney tumour types, with an accuracy of $\sim$ 93%, using a compact set of $\sim$ 50 interpretable rules.

Much work remains to be done, and fuzzy set theory has still a big role to play in machine learning.

7.1 Directions for future work

The following subsections present a concise list of future directions which build on the work presented in this thesis. Here we mainly focus on two particular aspects: FRBSs and real case applications of fuzzy models.

7.1.1 The future of FRBS

In an age of deep complex black-box models, FRBSs may seem especially out of date; yet, their continuance is primarily due to their being simple and understandable. Even if the problem of finding an agreed definition of interpretability is far from being solved — and maybe even nonsensical, being user and problem dependent — the fact that a system with only 7 rules is able to achieve nearly state-of-the-art accuracies on a 10M examples dataset remains astonishing (Chapter 5). Thus, despite having been jettisoned by the machine learning research community in the nineties, rule-based systems may still prove a worthy research topic. The extremely fast pace at which the ML community is moving means that a lot of nook and crannies are left unexplored. More often than not, however, something worthy is found there; the several recent advancements in FRBCs are there to prove it.

Considering the context of big data, there is still a lot of work to be done; future works should address the problem, in the specific setting of FRBCs, of bounding the size of the

training set without experiencing losses in the achieved accuracy. In fact, this aspect is crucial in dealing with Big Data, and effective solutions can extend the practical applicability of DPAES-FDT-GL to extremely big datasets, with no significant additional penalties in the runtimes for the learning phase. Considering EFS, for example, the limiting factor to overcame when coping with Big Data is the computation of the accuracy on the overall training set. This computation depends on the number of instances in the training set and on the dimensionality of each instance. In Big Data generally both these numbers are high and then require long runs before achieving satisfactory solutions. Thus, techniques for reducing the number of attributes and the numerosity of the datasets, preserving the accuracy achieved by the models, are very appealing.

7.1.2 Applications of fuzzy modelling

In the several decades that have passed since its inception, fuzzy set theory has evolved as a very powerful and general modelling tool, able to deal with uncertainty in real life scenarios. In fact, among the many contributions of fuzzy set theory, is that it made researchers more aware of uncertainty. Uncertainty is unavoidable, and, as such, is better to manage it properly than to simply ignore it.

We believe that, particularly in the application realm (for example, bioinformatics), there is still a significant contribution to be made. Because of its generality, modelling uncertain phenomena is indeed where fuzzy set theory really shines. As stated, fuzzyness describes event ambiguity, rather than uncertainty in event occurrence: since a wide class of biologically related problems are based on vague assumptions, they may benefit from fuzzy set theory and logic.

APPENDIX

A

LIST OF PUBLICATIONS

International Journals

1. Renda A., Barsacchi M., Bechini A., and Marcelloni F. , "Comparing ensemble strategies for deep learning: An application to facial expression recognition". *Submitted to Expert System with Applications, 2018*

2. Barsacchi M., Andres-Terre H., and Pietro L. , "Geese: Metabolically driven latent space learning for gene expression data". *Submitted to Bioinformatics*, pre-print available at bioRxiv `https://doi.org/10.1101/365643`, 2018

3. Barsacchi, M. Bechini, A., Ducange, P. and Marcelloni F. (2019). Optimizing Partition Granularity, Membership Function Parameters, and Rule Bases of Fuzzy Classifiers for Big Data by a Multi-objective Evolutionary Approach. *in Cognitive Computation, pp. 1-21, January 2019.*

4. Barsacchi, M. Bechini, A. and Marcelloni F. (2018). Using Fuzzy Decision Trees in Boosting: An Ensemble Multi-class Classifier and its Experimental Evaluation. *Under Review.*

5. Barsacchi M., Novoa E. M., Kellis M., and Bechini A., "SwiSpot: Modeling Riboswitches by Spotting Out Switching Sequences," *in Bioinformatics, pp. 3252–3259, November. 2016*

International Conferences with Peer Review

1. Renda A., Barsacchi M., Bechini A. and Marcelloni F. "Comparing ensemble strategies for Deep Learning. An application to Facial Expression Recognition", *in Proc. of 4th Int'l Conf. on Machine Learning, Optimization, and Data Science - Sept. 13-16, 2018 - Volterra, Italy*

2. Ricatto, M., Barsacchi, M., & Bechini A. (2018). “Interpretable CNV-based Tumour Classification using Fuzzy Rule Based Classifiers”, *in ACM SAC 2018.* https://doi.org/10.1145/3167132.3167135.

3. Barsacchi, M., Bechini, A., & Marcelloni, F. (2017). “Multi-class boosting with fuzzy decision trees” *in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1–6)* https://doi.org/10.1109/FUZZ-IEEE.2017.8015567

4. Barsacchi M., Baù, A. and Bechini A., “Extensive Assessment of Metrics on RNA Secondary Structures and Relative Ensembles” *in Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 44–47.*

BIBLIOGRAPHY

Al-sharhan, S., Karray, F., Gueaieb, W., and Basir, O. (2001). Fuzzy entropy: a brief survey. In *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, volume 3, pages 1135–1139.

Alcalá, R., Alcalá-Fdez, J., Herrera, F., and Otero, J. (2007). Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning*, 44(1):45 – 64.

Alkan, C., Coe, B. P., and Eichler, E. E. (2011). Genome structural variation discovery and genotyping. *Nature Reviews Genetics*, 12(5):363–376.

Altay, A. and Cinar, D. (2016). Fuzzy decision trees. In Kahraman, C. and Kabak, Ö., editors, *Fuzzy Statistical Decision-Making: Theory and Applications*, pages 221–261. Springer International Publishing, Cham.

Antonelli, M., Ducange, P., Lazzerini, B., and Marcelloni, F. (2009a). Learning concurrently partition granularities and rule bases of Mamdani fuzzy systems in a multi-objective evolutionary framework. *International Journal of Approximate Reasoning*, 50(7):1066–1080.

Antonelli, M., Ducange, P., Lazzerini, B., and Marcelloni, F. (2009b). Multi-objective evolutionary learning of granularity, membership function parameters and rules of Mamdani fuzzy systems. *Evolutionary Intelligence*, 2(1-2):21–37.

Antonelli, M., Ducange, P., Lazzerini, B., and Marcelloni, F. (2016a). Multi-objective evolutionary design of granular rule-based classifiers. *Granular Computing*, 1(1):37–58.

Antonelli, M., Ducange, P., and Marcelloni, F. (2014). A fast and efficient multi-objective evolutionary learning scheme for fuzzy rule-based classifiers. *Information Sciences*, 283:36–54.

Antonelli, M., Ducange, P., and Marcelloni, F. (2016b). Multi-objective evolutionary design of fuzzy rule-based systems. In *HANDBOOK ON COMPUTATIONAL INTELLIGENCE: Volume 2: Evolutionary Computation, Hybrid Systems, and Applications*, pages 635–670. World Scientific.

Anuradha, J. et al. (2015). A brief introduction on big data 5vs characteristics and hadoop technology. *Procedia computer science*, 48:319–324.

Ayesh, A. and Blewitt, W. (2015). Models for computational emotions from psychological theories using type I fuzzy logic. *Cognitive Computation*, 7(3):285–308.

Baczyński Micheal, J. B. (2008). An introduction to fuzzy implications. In *Fuzzy Implications. Studies in Fuzziness and Soft Computing, vol 231*, chapter 1, pages 1–35. Springer, Berlin, Heidelberg.

Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5.

Baldwin, J. F. and Xie, D. (2005). Intelligent information processing ii. In Shi, Z. and He, Q., editors, *Intelligent Information Processing II*, chapter Simple Fuzzy Logic Rules Based on Fuzzy Decision Tree for Classification and Prediction Problem, pages 175–184. Springer-Verlag, London, UK, UK.

Bechini, A., De Matteis, A. D., Marcelloni, F., and Segatori, A. (2016a). Spreading fuzzy random forests with MapReduce. In *Proc. of IEEE SMC 2016*. IEEE Computer Society.

Bechini, A., Marcelloni, F., and Segatori, A. (2016b). A MapReduce solution for associative classification of big data. *Information Sciences*, 332:33–55.

Beck, F., Burch, M., Munz, T., Di Silvestro, L., and Weiskopf, D. (2015). Generalized pythagoras trees: A fractal approach to hierarchy visualization. In Battiato, S., Coquillart, S., Pettré, J., Laramee, R. S., Kerren, A., and Braz, J., editors, *Computer Vision, Imaging and Computer Graphics - Theory and Applications*, volume 550 of *Comm. in Computer and Inf. Science*, chapter 8, pages 115–135. Springer.

Bonissone, P., Cadenas, J. M., Garrido, M. C., and Díaz-Valladares, R. A. (2010). A fuzzy random forest. *International Journal of Approximate Reasoning*, 51(7):729 – 747.

Boyen, X. and Wehenkel, L. (1999). Automatic induction of fuzzy decision trees and its application to power system security assessment. *Fuzzy Sets and Systems*, 102(1):3–19.

Breiman, L. (1993). *Classification and regression trees*. Chapman & Hall.

Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *Ann. Statist.*, 26(3):801–849.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Chandra, B. and Varghese, P. (2008). Fuzzy SLIQ decision tree algorithm. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(5):1294–1301.

Chang, K., Butterfield, Y. S. N., Chu, A., Chuah, E., et al. (2013). The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*, 45(10):1113–1120.

Chang, R. L. P. and Pavlidis, T. (1977). Fuzzy decision tree algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(1):28–35.

Chi, Z., Yan, H., and Phạm, T. (1996). *Fuzzy algorithms: with applications to image processing and pattern recognition*, volume 10 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific.

Chiang, I.-J. and jen Hsu, J. Y. (2002). Fuzzy classification trees for data analysis. *Fuzzy Sets and Systems*, 130(1):87 – 99.

Cococcioni, M., Ducange, P., Lazzerini, B., and Marcelloni, F. (2007). A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. *Soft Computing*, 11(11):1013–1031.

Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2nd edition.

Conrad, D. F., Pinto, D., Redon, R., Feuk, L., et al. (2010). Origins and functional impact of copy number variation in the human genome. *Nature*, 464(7289):704–712.

Cordón, O. (2011). A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *Int. J. Approx. Reasoning*, 52(6):894–913.

Cordón, O., del Jesus, M. J., and Herrera, F. (1999). A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning*, 20(1):21 – 45.

Crockett, K., Latham, A., and Whitton, N. (2017). On predicting learning styles in conversational intelligent tutoring systems using fuzzy decision trees. *International Journal of Human-Computer Studies*, 97:98 – 115.

De Matteis, A. D., Marcelloni, F., and Segatori, A. (2015). A new approach to fuzzy random forest generation. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Comm. of the ACM*, 51(1):107–113.

De'ath, G. (2007). Boosted trees for ecological modeling and prediction. *Ecology*, 88(1):243–251.

del Río, S., López, V., Benítez, J. M., and Herrera, F. (2015a). A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *Int'l Journal of Computational Intelligence Systems*, 8(3):422–437.

del Río, S., López, V., Benítez, J. M., and Herrera, F. (2015b). A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *International Journal of Computational Intelligence Systems*, 8(3):422–437.

Díaz-Uriarte, R. and Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3.

Domingos, P. (1997). Why does bagging work? a bayesian account and its implications. In *Proc. of the Third Int'l Conf. on Knowledge Discovery and Data Mining*, pages 155–158. AAAI Press.

Dong, M. and Kothari, R. (2001). Look-ahead based fuzzy decision tree induction. *IEEE Transactions on Fuzzy Systems*, 9(3):461–468.

Dubois, D. and Prade, H. (2012). Gradualness, uncertainty and bipolarity: Making sense of fuzzy sets. *Fuzzy Sets and Systems*, 192:3 – 24. Fuzzy Set Theory — Where Do We Stand and Where Do We Go?

Duţu, L. C., Mauris, G., and Bolon, P. (2018). A fast and accurate rule-base generation method for Mamdani fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 26(2):715–733.

Eibe, F., Hall, M. A., and Witten, I. H. (2016). Appendix B: The WEKA workbench. In *Data Mining: Practical Machine Learning Tools and Techniques*, page 553–571. Morgan Kaufmann, 4 edition.

Elkano, M., Galar, M., Sanz, J., and Bustince, H. (2017). CHI-BD: A fuzzy rule-based classification system for big data classification problems. *Fuzzy Sets and Systems*.

Fayyad, U. M. and Irani, K. B. (1993a). Multi-interval discretization of continuous-valued attributes for classification learning. *Proc. of the 13th Int'l Joint Conference on Artificial Intelligence*, pages 1022–1027.

Fayyad, U. M. and Irani, K. B. (1993b). Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029.

Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., and Herrera, F. (2013). A review of the application of multi-objective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy Systems*, 21(1):45–65.

Fernández, A., Almansa, E., and Herrera, F. (2017). Chi-Spark-RS: an Spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

Fernández, A., Carmona, C. J., del Jesus, M. J., and Herrera, F. (2016a). A view on fuzzy systems for Big Data: Progress and opportunities. *Int'l Journal of Computational Intelligence Systems*, 9(sup1):69–80.

Fernández, A., del Río, S., Bawakid, A., and Herrera, F. (2016b). Fuzzy rule based classification systems for big data with MapReduce: granularity analysis. *Advances in Data Analysis and Classification*, pages 1–20.

Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M. J., Benítez, J. M., and Herrera, F. (2014). Big data with cloud computing: an insight on the computing environment, MapReduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(5):380–409.

Fernández, A., López, V., del Jesus, M. J., and Herrera, F. (2015). Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. *Knowledge-Based Systems*, 80:109 – 121. 25th anniversary of Knowledge-Based Systems.

Ferranti, A., Marcelloni, F., Segatori, A., Antonelli, M., and Ducange, P. (2017). A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. *Information Sciences*, 415:319–340.

Feuk, L., Carson, A. R., and Scherer, S. W. (2006). Structural variation in the human genome. *Nature Reviews Genetics*, 7(2):85–97.

Fodor, C. J. and Rudas, I. J. (2015). Basics of fuzzy sets. In Janusz Kacprzyk, W. P., editor, *Springer Handbook of Computational Intelligence.*, chapter 10, pages 159–170. Springer, Berlin, Heidelberg.

Fodor János, Y. R. R. (2000). Fuzzy set-theoretic operators and quantifiers. In Didier Dubois, H. P., editor, *Fundamentals of Fuzzy Sets. The Handbooks of Fuzzy Sets Series, vol 7.*, chapter 4, pages 125–193. Springer, Boston, MA.

Freeman, J. L., Perry, G. H., Feuk, L., Redon, R., et al. (2006). Copy number variation: New insights in genome diversity. *Genome Research*, 16(8):949–961.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

Gacto, M. J., Alcalá, R., and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20):4340–4360.

Gadomer, Ł. and Sosnowski, Z. A. (2016). Fuzzy random forest with c–fuzzy decision trees. In Saeed, K. and Homenda, W., editors, *Computer Information Systems and Industrial Management*, pages 481–492, Cham. Springer International Publishing.

García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644.

Girirajan, S., Campbell, C. D., and Eichler, E. E. (2011). Human copy number variation and complex genetic disease. *Annual Review of Genetics*, 45:203–226.

Grosan, C. and Abraham, A. (2011). Rule-based expert systems. In *Intelligent Systems. Intelligent Systems Reference Library, vol 17*, chapter 7, pages 149–185. Springer, Berlin, Heidelberg.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class AdaBoost. *Statistics and Its Interface*, 2(3):349–360.

Hühn, J. and Hüllermeier, E. (2009). FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319.

Hüllermeier, E. and Vanderlooy, S. (2009). Why fuzzy decision trees are good rankers. *IEEE Transactions on Fuzzy Systems*, 17(6):1233–1244.

Hüllermeier, E. (2011). Fuzzy sets in machine learning and data mining. *Applied Soft Computing*, 11(2):1493 – 1505. The Impact of Soft Computing for the Progress of Artificial Intelligence.

Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595.

Isazadeh, A., Mahan, F., and Pedrycz, W. (2016). Mflexdt: multi flexible fuzzy decision tree for data stream classification. *Soft Computing*, 20(9):3719–3733.

Ishibuchi, H., Nakashima, T., and Murata, T. (2001). Three-objective genetics-based machine learning for linguistic rule extraction. *Information Sciences*, 136(1-4):109–133.

Ishibuchi, H. and Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy sets and systems*, 141(1):59–88.

Ishibuchi, H., Yamamoto, T., Member, S., Ishibuchi, P. H., Ishibuchi, H., Yamamoto, T., and Member, S. (2005). Rule weight specification in fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Systems*, 13:428–435.

Janikow, C. (1998). Fuzzy decision trees: issues and methods. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 28(1):1–14.

Kendall, J. and Krasnitz, A. (2014). *Computational Methods for DNA Copy-Number Analysis of Tumors*, pages 243–259. Springer New York, New York, NY.

Kim, Y., Shim, K., Kim, M.-S., and Lee, J. S. (2014). DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce. *Information Systems*, 42:15–35.

Klir, G. J. and Yuan, B. (1995). *Fuzzy sets and fuzzy logic: theory and applications*. Prentice Hall.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172.

Kosko, B. (1990). Fuzziness vs. probability. *International Journal of General Systems*, 17(2-3):211–240.

Laddha, S. V., Ganesan, S., Chan, C. S., and White, E. (2014). Mutational Landscape of the Essential Autophagy Gene BECN1 in Human Cancers. *Molecular Cancer Research*, 12(4):485–490.

Li, B., You, J., Huang, T., and Cai, Y.-D. (2014). Classification of non-small cell lung cancer based on copy number alterations. *PLOS ONE*, 9(2):1–8.

Lin, C., Tsai, C., Lee, C., and Lin, C. (2014). Large-scale logistic regression and linear support vector machines using spark. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 519–528.

Lindley, D. V. (1987). The probability approach to the treatment of uncertainty in artificial intelligence and expert systems. *Statist. Sci.*, 2(1):17–24.

Linehan, W. M. (2012). Genetic basis of kidney cancer: Role of genomics for the development of disease-based therapeutics. *Genome Research*, 22(11):2089–2100.

López, V., del Río, S., Benítez, J. M., and Herrera, F. (2015). Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. *Fuzzy Sets and Systems*, 258:5–38.

Ludwig, S. A. (2015). MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability. *International journal of machine learning and cybernetics*, 6(6):923–934.

Luis, M. (2015). Fuzzy rule-based systems. In Janusz Kacprzyk, W. P., editor, *Springer Handbook of Computational Intelligence.*, chapter 13, pages 203–218. Springer, Berlin, Heidelberg.

Maillo, J., Ramírez, S., Triguero, I., and Herrera, F. (2017). kNN-IS: An iterative Spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, 117:3–15.

Mamdani, E. and Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1 – 13.

Manwani, N. and Sastry, P. S. (2012). Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):181–192.

Márquez, A., Márquez, F., and Peregrín, A. (2017). A scalable evolutionary linguistic fuzzy system with adaptive defuzzification in big data. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

Mayer-Schönberger, V. and Cukier, K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Eamon Dolan/Houghton Mifflin Harcourt.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh, R., Zaharia, M., and Talwalkar, A. (2016). Mllib: Machine learning in apache spark. *J. Mach. Learn. Res.*, 17(1):1235–1241.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Mukherjee, I. and Schapire, R. E. (2013). A theory of multiclass boosting. *Journal of Machine Learning Research*, 14(2):437–497.

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

Nowakowska, B. (2017). Clinical interpretation of copy number variants in the human genome. *Journal of Applied Genetics*, 58(4):449–457.

Olaru, C. and Wehenkel, L. (2003). A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221 – 254.

Oneto, L., Bisio, F., Cambria, E., and Anguita, D. (2017). Semi-supervised learning for affective common-sense reasoning. *Cognitive Computation*, 9(1):18–42.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *J. Artif. Int. Res.*, 11(1):169–198.

Pedrycz, W., Skowron, A., and Kreinovich, V. (2008). *Handbook of Granular Computing*. Wiley-Interscience, New York, NY, USA.

Pedrycz, W. and Sosnowski, Z. A. (2005). C-fuzzy decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):498–511.

Poniah, P., Zain, S. M., Razack, A. H. A., Kuppusamy, S., et al. (2017). Genome-wide copy number analysis reveals candidate gene loci that confer susceptibility to high-grade prostate cancer. *Urologic Oncology: Seminars and Original Investigations*.

Qiu, Z., Bi, J., Gazdar, A. F., and Song, K. (2017). Genome-wide copy number variation pattern analysis and a classification signature for non-small cell lung cancer. *Genes, Chromosomes & Cancer*, 56(7):559–569.

Quinlan, A. R. and Hall, I. M. (2010). Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ramírez-Gallego, S., García, S., Mouriño Talín, H., Martínez-Rego, D., Bolón-Canedo, V., Alonso-Betanzos, A., Benítez, J. M., and Herrera, F. (2016). Data discretization: Taxonomy and big data challenge. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(1):5–21.

Rey, M., Galende, M., Fuente, M., and Sainz-Palmero, G. (2017). Multi-objective based fuzzy rule based systems (FRBSs) for trade-off improvement in accuracy and interpretability: A rule relevance point of view. *Knowledge-Based Systems*, 127:67 – 84.

Ricatto, M., Barsacchi, M., and Bechini, A. (2018). Interpretable cnv-based tumour classification using fuzzy rule based classifiers. In *Proc. of the 33rd ACM Symposium on Applied Computing*, SAC 18, New York, NY, USA. ACM.

Rodriguez-Galiano, V., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:93 – 104.

Roe, B. P., Yang, H.-J., Zhu, J., Liu, Y., Stancu, I., and McGregor, G. (2005). Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2):577 – 584.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

Ross, J. T. (2011). *Fuzzy Logic with Engineering Applications, Third Edition.* John Wiley & Sons, Hoboken, NJ, USA.

Russo, P. (2006). Evaluation of the nonneoplastic pathology in tumor nephrectomy specimens: Predicting the risk of progressive renal failure. *Urologic Oncology: Seminars and Original Investigations*, 24(6):558.

Saberian, M. J. and Vasconcelos, N. (2011). Multiclass boosting: Theory and algorithms. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2124–2132. Curran Associates, Inc.

Sardari, S., Eftekhari, M., and Afsari, F. (2017). Hesitant fuzzy decision tree approach for highly imbalanced data classification. *Applied Soft Computing*, 61:727 – 741.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schapire, R. E. (2013). Explaining AdaBoost. In Schölkopf, B., Luo, Z., and Vovk, V., editors, *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 37–52. Springer Berlin Heidelberg.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms.* MIT press.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686.

Segatori, A., Bechini, A., Ducange, P., and Marcelloni, F. (2017a). A distributed fuzzy associative classifier for big data. *IEEE Transactions on Cybernetics*.

Segatori, A., Marcelloni, F., and Pedrycz, W. (2017b). On distributed fuzzy decision trees for big data. *IEEE Trans. on Fuzzy Systems*, PP(99).

Segatori, A., Marcelloni, F., and Pedrycz, W. (2017c). On distributed fuzzy decision trees for big data. *IEEE Transactions on Fuzzy Systems*, PP(99):1–1.

Segatori, A., Marcelloni, F., and Pedrycz, W. (2018). On distributed fuzzy decision trees for big data. *IEEE Transactions on Fuzzy Systems*, 26(1):174–192.

Shi, Y., Eberhart, R., and Chen, Y. (1999). Implementation of evolutionary fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 7(2):109–119.

Shlien, A. and Malkin, D. (2009). Copy number variations and cancer. *Genome medicine*, 1(6):62.

Shlien, A. and Malkin, D. (2010). Copy number variations and cancer susceptibility. *Current Opinion in Oncology*, 22(1):55–63.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., and Feuston, B. P. (2003). Random forest:a classification and regression tool for compound classification and qsar modeling. *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958. PMID: 14632445.

Umanol, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S., and Kinoshita, J. (1994). Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, pages 2113–2118 vol.3.

Van Veldhuizen, D. A., Zydallis, J. B., and Lamont, G. B. (2003). Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):144–173.

Wang, H., Xu, Z., and Pedrycz, W. (2017). An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities. *Knowledge-Based Systems*, 118:15–30.

Wang, X., Liu, X., Pedrycz, W., and Zhang, L. (2015). Fuzzy rule based decision trees. *Pattern Recognition*, 48(1):50 – 59.

Wang, X.-Z., Yeung, D., and Tsang, E. (2001). A comparative study on heuristic algorithms for generating fuzzy decision trees. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(2):215–226.

White, T. (2009). *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 3 edition.

White, T. (2012). *Hadoop: The definitive guide*. O'Reilly Media, Inc.

Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107.

Yates, A. et al. (2016). Ensembl 2016. *Nucleic Acids Research*, 44(D1):D710–D716.

Yuan, Y. and Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2):125–139.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338 – 353.

Zadeh, L. (1975a). The concept of a linguistic variable and its application to approximate reasoning—i. *Information Sciences*, 8(3):199 – 249.

Zadeh, L. (1975b). The concept of a linguistic variable and its application to approximate reasoning—ii. *Information Sciences*, 8(4):301 – 357.

Zadeh, L. (1975c). The concept of a linguistic variable and its application to approximate reasoning—iii. *Information Sciences*, 8(4):301 – 357.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, volume 10, page 10.

Zare, F., Dow, M., Monteleone, N., Hosny, A., et al. (2017). An evaluation of copy number variation detection tools for cancer using whole exome sequencing data. *BMC Bioinformatics*, 18(1):286.

Zarrei, M., MacDonald, J. R., Merico, D., and Scherer, S. W. (2015). A copy number variation map of the human genome. *Nat Rev Genet*, 16(3):172–183.

Zeinalkhani, M. and Eftekhari, M. (2014). Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers. *Information Sciences*, 278:715–735.

Zhai, S., Xia, T., and Wang, S. (2014). A multi-class boosting method with direct optimization. In *Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, KDD '14, pages 273–282, New York, NY, USA. ACM.

Zhang, N., Wang, M., Zhang, P., and Huang, T. (2016). Classification of cancers based on copy number variation landscapes. *Biochimica et Biophysica Acta (BBA) - General Subjects*, 1860(11):2750 – 2755.

Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361.

Zimmermann, H.-J. (1996). *Fuzzy Set Theory—and Its Applications, Third Edition.* Kluwer Academic Publishers, Norwell, MA, USA.

Zimmermann, H.-J. (2010). Fuzzy set theory. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3):317–332.