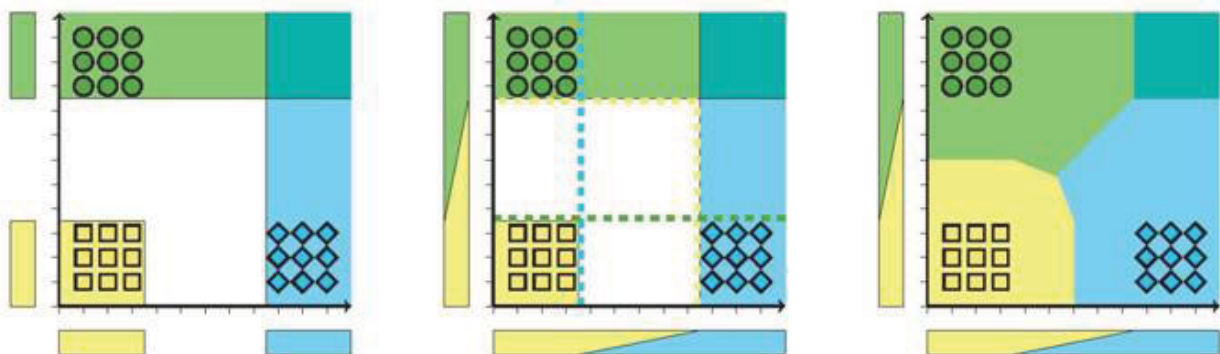

Induction and Fuzzification of Classification Rules



Induction and Fuzzification of Classification Rules



Dissertation

zur Erlangung des Doktorgrades
der Naturwissenschaften
(Dr. rer. nat.)

dem Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt

von

Jens Christian Hühn

aus Marburg

Marburg 2009

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen
Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.

1. Aufl. - Göttingen: Cuvillier, 2009

Zugl.: Marburg, Univ., Diss., 2009

978-3-86955-186-9

Vom Fachbereich Mathematik und Informatik der Philipps-Universität Marburg
als Dissertation angenommen am: 1.12.2009

Erstgutachter: Prof. Dr. Eyke Hüllermeier

Zweitgutachter: Prof. Dr. Johannes Fürnkranz

Tag der mündlichen Prüfung: 8.12.2009

Diese Dissertation wurde durch ein Graduiertenstipendium der Begabtenförderung
der Konrad-Adenauer-Stiftung e.V. gefördert.

© CUVILLIER VERLAG, Göttingen 2009

Nonnenstieg 8, 37075 Göttingen

Telefon: 0551-54724-0

Telefax: 0551-54724-21

www.cuvillier.de

Alle Rechte vorbehalten. Ohne ausdrückliche Genehmigung
des Verlages ist es nicht gestattet, das Buch oder Teile
daraus auf fotomechanischem Weg (Fotokopie, Mikrokopie)
zu vervielfältigen.

1. Auflage, 2009

Gedruckt auf säurefreiem Papier

978-3-86955-186-9

To my parents

Abstract

The ability to learn new things is one of the main characteristics of intelligent behavior. It is this that enables us humans to recognize patterns and to generalize following the structures we encounter. We are able to learn riding a bike, playing the piano, speaking a foreign language and so much more. But when confronted with too much information, it might be impossible for us to find a pattern. This is where the computers come in: A computer might not be able to learn the examples mentioned above, but it is definitely capable of crunching numbers. In fact, there are specific learning tasks that a computer can tackle very effectively.

The discipline in which such kind of intelligent algorithms are sought is called “machine learning”. One challenging task from this domain is classification learning which is considered to be a supervised learning scenario: Considering examples that belong to distinct classes, the task is to learn a generalizing pattern — one which is capable of predicting the true class membership of an instance for which the class is unknown. This is a very general setup. Indeed, every problem that can be formulated accordingly, can be solved with a classification algorithm. For example, diagnosing the disease a patient suffers from, predicting whether loans will be defaulted on, recognizing the digits from a handwritten number. For this generalizing leap there exists a wealth of different methods that all have different characteristics.

One classical approach for this problem are classification rules. Such a rule consists of two parts: The antecedent part as a conditional expression and the consequence as a class assignment. While for a conventional rule the antecedent part consists of selectors on the attributes, the antecedent of a linguistic rule contains fuzzy sets. The latter are combined by means of fuzzy logic connectives. A key feature of rule-based classification is the comprehensible reasoning mechanism. This is a priceless quality for a human expert who must rely on the prediction. He can intervene in an informed way if the reasoning is not proper. This holds for linguistic fuzzy rule-based classifiers especially: They are even more readable and comprehensible. Basically so far, the development of conventional and fuzzy rule learners has been conducted independently. While discriminative abilities have been in the focus for the conventional algorithms, the aspect of interpretability has been the most prominent design maxime for the linguistic ones.

The purpose of this thesis is to bring the scientific realms of conventional and fuzzy rule learning together. For achieving this objective we will combine the best of both worlds for developing three new classifiers. Two of them — the FURIA and the FR3 algorithm — will combine novel fuzzy methods with an improved variant of a proven conventional rule learner and one — the HELLFIRE algorithm — will be an all new linguistic fuzzy rule learner using a new discretization framework and new boundary softening techniques.

The FURIA algorithm swiftly learns a well-classifying fuzzy ruleset. Its main contribution are (A) a novel data-driven rule boundary softening that finds better decisions in borderline regions and (B) a novel method for efficiently stretching the rules if a query instance is not covered at all. The FR3 classifier is based on FURIA but it has the ability to distinguish between different kinds of uncertainty in classification decisions. For every query instance FR3 creates a so-called fuzzy preference structure — based on coverage degrees — that discerns two types of uncertainty called conflict and ignorance respectively. Complementary to these two approaches, the HELLFIRE algorithm induces linguistic fuzzy rule-based models. It uses an innovative strategy for obtaining both a data discretization and a rule model simultaneously. The outcome of this procedure are conventional grid-based rules that are made fuzzy using a technique which is slightly related to the one of FURIA and FR3. The background of all three fuzzification techniques is to provide rule coverage degrees which match the support of the class found in the training data.

We will show that our novel fuzzification techniques are very effective at improving the discriminative abilities of the introduced classifiers. We will point out in detail that softened rule boundaries are both more flexible and more reliable. All three algorithms are comparable or superior to state-of-the-art conventional and fuzzy rule-based classification algorithms in terms of classification and ranking performance. The result is remarkable: Conventional rule learning techniques and fuzzy methods are a fertile combination. Even though rule learning has been researched in both the fuzzy set and the machine learning communities for years, this dissertation will be the first to make a well-founded investigation of a constructive union of both fields.

Zusammenfassung

Die Fähigkeit neue Dinge zu erlernen ist eine der Haupteigenschaften intelligenten Verhaltens. Dies ermöglicht uns Menschen Muster zu erkennen und gemäß diesen zu verallgemeinern. Wir lernen das Fahrradfahren, das Klavierspielen, eine Fremdsprache zu beherrschen und vieles mehr. Sobald aber der Umfang der Informationen überhand nimmt, sind wir nicht mehr in der Lage die Muster zu finden. An dieser Stelle kommen Computer ins Spiel. Sie sind vielleicht noch nicht in der Lage die oben genannten Beispiele zu erlernen, aber dafür können sie mit vielen Zahlen umgehen. Tatsächlich gibt es spezielle Lernaufgaben, die ein Computer sehr effektiv handhaben kann.

Die Disziplin, in der solche intelligente Algorithmen gesucht werden, heißt ‚Maschinelles Lernen‘. Eine anspruchsvolle Aufgabe aus diesem Bereich ist das überwachte Lernproblem des Klassifizierens: Hierbei soll aus zu unterschiedlichen Klassen zugeordneten Beispielen ein generalisierendes Schema abgeleitet werden, mit dem die Klassifikation neuer Beispiele möglich wird. Da diese Aufgabenstellung sehr allgemein gehalten ist, kann jedes entsprechende Problem mit einem Klassifikationsalgorithmus gelöst werden, beispielsweise das Diagnostizieren einer Krankheit, das Vorhersagen ob Schulden nicht beglichen werden oder die Erkennung handgeschriebener Ziffern. Für diese Art von Vorhersage wurden verschiedene Algorithmen entworfen.

Ein klassischer Ansatz, dieses Problem zu lösen, ist der Einsatz von Klassifikationsregeln. Diese bestehen aus zwei Teilen: Der Antezedens als konditionellem Ausdruck und der Konsequenz als Klassenzuweisung. Während für eine konventionelle Klassifikationsregel die Antezedens aus Bedingungen an die Attribute besteht, enthält die Antezedens einer unscharfen Fuzzy-Klassifikationsregel linguistisch interpretierbare Fuzzy-Mengen. Letztere werden mit Hilfe von fuzzy-logischen Operatoren verknüpft. Der Kern der regelbasierten Ansätze ist das verständliche Schlussfolgern. Dies ist eine unschätzbare Eigenschaft, wenn ein menschlicher Experte der Vorhersage vertrauen muss. Er kann einschreiten, wenn die Schlüsse nicht sinnvoll sind. Dies gilt ganz besonders für die linguistischen Fuzzy-Regelklassifizierer, welche als besonders interpretierbar und verständlich gelten. Die Entwicklung von konventionellen und unscharfen Regellern verlief bisher jedoch unabhängig voneinander. Während die Unterscheidungsfähigkeit im Fokus für die konventionellen stand, war die Interpretierbarkeit die bedeutendste Design-Maxime für die linguistischen Ansätze.

Der Zweck dieser Dissertation ist das Vereinen der wissenschaftlichen Felder des konventionellen und unscharfen Regellerns. Um dieses Ziel zu erreichen, wird das Beste beider Felder in drei neuen Algorithmen kombiniert. Zwei davon — der FURIA- und der FR3-Algorithmus — kombinieren neue Fuzzy-Methoden mit einer erweiterten Variante des ursprünglichen RIPPER-Regellerns. Der dritte, der HELLFIRE-Algorithmus, ist ein neuer linguistischer Fuzzy-Regellerner. Er lernt die Regeln mit Hilfe eines neuen Diskretisierungssystems und Grenzaufweichungsverfahrens.

Der FURIA-Algorithmus ist in der Lage, gut klassifizierende Fuzzy-Regelsätze effizient zu lernen. Die Hauptbeiträge von FURIA sind (A) eine neue datengetriebene Technik um Regelgrenzen aufzuweichen, was zu besseren Entscheidungen in Grenzregionen führt und (B) eine neue effiziente Methode, die Regeln zu dehnen, wenn die zu klassifizierende Instanz von keiner Regel überdeckt wird. Der FR3-Klassifizierer baut auf FURIA auf, aber hat die Fähigkeit, verschiedene Arten von Unsicherheiten zu unterscheiden. Für jede zu klassifizierende Instanz erzeugt er anhand der Überdeckungsgrade eine Fuzzy-Präferenzstruktur, die zwischen Konflikt und Ignoranz — als Ausprägungen von Unsicherheit — unterscheiden kann. Als Ergänzung zu diesen Verfahren erzeugt der HELLFIRE-Algorithmus linguistische Fuzzy-Regel-Modelle. Dies geschieht anhand einer innovativen Strategie, die sowohl eine Diskretisierung als auch Regeln gleichzeitig lernt. Das Produkt sind konventionelle, rasterbasierte Regeln, die mit einem Verfahren — ähnlich denen von FURIA und FR3 — fuzzyfiziert gemacht werden. Der Hintergrund aller drei Fuzzyfizierungsstrategien ist es Regelüberlappungsgrade zu bestimmen, die mit der Sicherheit der Regel übereinstimmen.

Die neuen Fuzzifizierungstechniken verbessern die Vorhersagegüte der bisher konventionellen Klassifizierer. Dies liegt an den weichen Regelgrenzen, die flexibler und zuverlässiger sind. Alle drei Algorithmen sind — in Bezug auf Klassifikationsgüte und Rangordnungsgüte — ähnlich oder überlegen zu modernen konventionellen und unscharfen Regellernern. Das Ergebnis ist beachtenswert: Konventionelle Regellertechniken und Fuzzy-Methoden bilden eine fruchtbare Kombination. Obwohl diese Methoden sowohl in der Fuzzy- und der Machine-Learning-Gemeinde seit Jahren untersucht werden, ist dies die erste fundierte Untersuchung einer Vereinigung beider Felder.

Contents

Abstract	i
Zusammenfassung	iii
Contents	v
1. Introduction	1
1.1. A Brief History of Conventional and Fuzzy Rule-Based Classification	2
1.2. Purpose of this Thesis	4
1.3. Contribution of this Thesis	4
1.4. Publications in the Context of this Thesis	5
1.5. Software Developments in the Context of this Thesis	6
1.6. Outline	7
2. Foundations	9
2.1. Classification	9
2.2. Binary Decomposition Techniques for Multi-Class Problems .	11
2.2.1. 1-vs-All Decomposition	11
2.2.2. All-vs-All Decomposition	11
2.2.3. Comparison of 1-vs-All and All-vs-All Decomposition	13
2.3. Rule-Based Classification	15
2.3.1. Separate-and-Conquer Rule Learning	15
2.3.2. Rule Learning with FOIL, REP and IREP	17
2.3.3. RIPPER	20
2.4. Fuzzy Rule-Based Classification	22
2.4.1. Fuzzy Logic	23

2.4.2.	Fuzzy Classification Rules	29
2.4.3.	Linguistic Fuzzy Classification Rules	29
2.4.4.	Fuzzy Reasoning Methods	31
2.5.	Experimental Settings	32
2.5.1.	Testing Environment	32
2.5.2.	Data Sets	33
2.5.3.	Benchmark Classifiers	33
2.5.4.	Performance Measures	35
2.5.5.	Test Setup	38
2.5.6.	Statistical Evaluation	39
2.6.	Summary	42
3.	FURIA: Fuzzy Unordered Rule Induction Algorithm	43
3.1.	Introduction	43
3.2.	Fuzzy Unordered Rule Induction Algorithm	44
3.2.1.	Learning Unordered Rulesets	44
3.2.2.	Pruning Modifications	45
3.2.3.	Rule Fuzzification	46
3.2.4.	Classifier Output	53
3.2.5.	Rule Stretching	55
3.3.	Experiments	57
3.3.1.	Classification Performance Analysis	57
3.3.2.	Ranking Performance Analysis	59
3.3.3.	Fuzzification Analysis	61
3.3.4.	Model Complexity Analysis	67
3.3.5.	Rule Stretching Analysis	67
3.3.6.	Runtime Analysis	69
3.4.	Summary	73
4.	HELLFIRE: Learning Linguistic Fuzzy Classification Rules	75
4.1.	Introduction	75
4.2.	High-End Learning of Linguistic Fuzzy Interval Rule Expertise	76
4.2.1.	Rule Learning	76
4.2.2.	From Split-Based Rules to Interval-Based Rules	86

4.2.3.	Pruning	90
4.2.4.	Rule Selection	91
4.2.5.	Interval Fuzzification	91
4.2.6.	Rule Weighting	95
4.2.7.	Handling of Missing Values	95
4.2.8.	Classification with Rule Stretching	96
4.2.9.	Complexity Analysis	96
4.3.	Comparing HELLFIRE with Other Discretization Techniques	98
4.4.	Experiments	100
4.4.1.	Classification Performance Analysis	100
4.4.2.	Ranking Performance Analysis	102
4.4.3.	Fuzzification Analysis	104
4.4.4.	Comparison between the Discretization Procedures of HELLFIRE and MDLP	113
4.4.5.	Interpretability Analysis	114
4.5.	Exemplary Model	122
4.6.	Summary	124
5.	FR3: Learning Fuzzy Preference Structures using Fuzzy Rules	127
5.1.	Introduction	128
5.2.	Preference Relations and Structures	130
5.3.	Learning Valued Preference Structures for Classification . . .	133
5.4.	Fuzzy Round Robin RIPPER	134
5.4.1.	Pairwise Decomposition	135
5.4.2.	Fuzzy Rules for Learning Fuzzy Preference Structures	136
5.4.3.	Making Classification Decisions Based on Fuzzy Pref- erence Structures	140
5.5.	Visualization of Fuzzy Preference Structures	141
5.6.	Experiments	142
5.6.1.	Classification Performance Analysis	142
5.6.2.	Ranking Performance Analysis	144
5.6.3.	Fuzzification Analysis	146
5.6.4.	Model Complexity Analysis	147

5.6.5. Analysis of Conflict and Ignorance as Measures of Uncertainty	147
5.7. Summary	149
6. Comparison of FURIA, HELLFIRE and FR3	153
6.1. Comparison of Motivations and Methods	153
6.1.1. Motivation	153
6.1.2. Problem Decomposition Technique	154
6.1.3. Rule Learning Strategy	154
6.1.4. Fuzzy Partitioning	154
6.1.5. Fuzzification Technique	154
6.1.6. Handling of Uncovered Instances	155
6.1.7. Score Aggregation	155
6.2. Experimental Comparison	156
6.2.1. Binary Data Sets	156
6.2.2. Multi-class Data Sets	157
6.2.3. Summary	159
6.3. Discussion	160
7. Related Work	163
7.1. Conventional Rule-Based Classifiers	163
7.1.1. Separate-and-Conquer Rule Learning	163
7.1.2. Decision Tree Rule Learning	165
7.1.3. AUC-Optimizing Rule Learning	166
7.1.4. Nearest Generalized Examples	167
7.2. Fuzzy Rule-Based Classifiers	168
7.2.1. Grid-Oriented Approaches	168
7.2.2. Non-Grid-Oriented Approaches	169
7.2.3. Fuzzy Propositional and Fuzzy First-Order Logic Rule Learning	170
7.2.4. Fuzzy Set Covering	170
7.2.5. Hybrid Approaches	171
7.2.6. Clustering-Based Techniques	175
7.2.7. Divide-and-Conquer	176

7.2.8. Conventional vs. Fuzzy Partitions	177
7.2.9. Rule Weights and Confidence Factors	177
7.3. Dealing with Uncertainty	178
7.4. Interpretability	179
7.5. Discussion	180
7.5.1. FURIA	180
7.5.2. HELLFIRE	181
7.5.3. FR3	182
7.6. Summary	182
8. Conclusion and Outlook	183
8.1. Conclusion	183
8.2. Future Work	184
8.2.1. FURIA	184
8.2.2. FR3	184
8.2.3. HELLFIRE	185
8.2.4. General Directions	185
8.2.5. Outlook	186
A. Tables	187
Acknowledgements	197
Glossary	199
List of Tables	201
List of Figures	205
Bibliography	207
Erklärung	229
Resume	231
Index	233

1

Introduction

The advent of the computer provided mankind with a new tool that was able to solve problems which were infeasible so far. The computer made possible the solving of difficult and extensive calculations in shorter time. Machines were increasingly improved and today they are able to make millions of calculations in the blink of an eye. Even though machines have already been calculating in speeds that outperformed human capability for a long time, they are unable to act intelligently or in a self-aware manner. So far, really intelligent and self-aware machines only exist in science fiction, such as HAL 9000 or the Terminator. But there are niches where a computer or device might learn to act “intelligently”.

So far, it is not known how a computer could learn in a human way, how it could learn any arbitrary concept from its experience or its failures. Nevertheless, there are special tasks which a computer can learn to solve, e.g. trying to find treatments for a new disease from medical records or saving energy costs of a house by setting the heating apparatus according to the usage patterns of the occupants. Despite the fact that those tasks are in general quite specific and must be clearly defined, a computer can make use of its calculating speed and create new insights that the human user or even expert in that field is unaware of yet [Mit97].

The question of how a computer can be enabled to solve certain problems on its own is dealt with in *machine learning* research. The purpose of this field is to find new algorithms that let a computer create generalizing knowledge from the experience or data that was available so far.

One of the most popular and interesting problems in machine learning is *classification*. It assumes a set of examples from which each belongs to exactly one distinct class. The generalization desired from the machine learning algorithm is a mapping that is able to predict an unknown instance's true class.

Researchers from various backgrounds have developed a wealth of algorithms that are able to cope with classification problems. The perhaps most famous one might be the *neural network* that emerged from the Perceptron idea introduced by the psychologist Rosenblatt in the late 1950s [Ros58]. Others include Support Vector Machines, Logistic Regression or probabilistic approaches such as the family of Bayesian classifiers. Even though the mentioned techniques apply very different learning strategies, they all have a lack of interpretability in common. This materializes in classification decisions that can neither be understood nor explained. The reasoning mechanism within those *black box* algorithms remains untransparent. But in addition to these algorithms, symbolic approaches such as decision trees and rule-based models have also been developed: Machine learning algorithms that are considered to be interpretable. When classification decisions can be explained with the model, a human expert could intervene if the reasoning does not fit. To make those algorithms even more interpretable, researchers have conceived linguistic variants based on Zadeh's *fuzzy logic*. Fuzzy logic is an extension of the classical two-valued logic that allows intermediate degrees of truth [Zad65]. In contrast to decision trees and rule-based models the linguistic ones do not operate on the attribute values directly but on linguistically interpretable fuzzy sets instead. This meaning is attached to the fuzzy set in form of a label. Consequently, linguistic fuzzy models are very readable and understandable even without an underlying knowledge of the attribute domains. This is especially useful when a human expert is interacting with such a system.

1.1. A Brief History of Conventional and Fuzzy Rule-Based Classification

The domain of rule-based classification emerged in the 1960s, most notably with expert systems — a realm pioneered by Edward A. Feigenbaum [Fei80]. These systems were built to reproduce specific human knowledge, e.g. through

inference rules. While in the beginning the rule logic had to be elicited within expert interviews, Ryszard S. Michalski developed the idea of separate-and-conquer learning, an inductive technique which is able to infer rules from data automatically. At roughly the same time, Lotfi A. Zadeh was working on a multi-valued logic which he coined fuzzy logic. Fuzzy logic was mainly intended as a formal framework of a human-like approximate reasoning and “computing with words”.

From the late 1980s until the mid 1990s the realm of separate-and-conquer rule learning was studied by a large number of researchers in the machine learning community. Competition concerning efficiency and predictive quality led to significant improvements in this field. During that time a large number of heuristics and strategies to learn simple but well-classifying models emerged. A cornerstone of separate-and-conquer rule learning was developed by Johannes Fürnkranz with the IREP algorithm, which was used by William W. Cohen in the RIPPER classifier from 1995, which remains a state-of-the-art algorithm even today.

The task of classification learning was also discovered by researchers from the fuzzy set community. In 1992, Li-Xin Wang and Jerry M. Mendel published their work on how to learn a linguistic fuzzy rule-based classification model. In contrast to the discoveries from the machine learning field, Wang and Mendel used a rather simple grid-partitioning scheme that was able to yield interpretable rules but which was lacking effectivity and efficiency. Researchers from the fuzzy set community developed new algorithms as a remedy for the weakness of the initial idea. The main objective was to maintain interpretability while improving discriminative power. One of the dominating approaches since then has been the idea of using evolutionary algorithms for improving the grid-oriented linguistic fuzzy rules. In contrast to the fast and steady improvements in the machine learning community, no benchmark state-of-the-art learner within the fuzzy set community evolved. Albeit dozens of evolutionary and other hybrid algorithms for linguistic fuzzy rule learning have been published, there is none which excels. While the machine learning community designed lean but effective methods and heuristics, the fuzzy set community applied even more gargantuan approaches which were both ineffective and slow. The direct comparison of conventional with these fuzzy rule learning techniques shows clearly that the latter are not yet competitive in terms of predictive power. It is by no means an exaggeration to describe the efforts to build these gargantuan systems as a scientific dead end so far.

Basically, there has been little effort in establishing a link between the possibilities that fuzzy logic offers with the experience and efficiency from the conventional rule-learning realm. In this direction there are still opportunities for improvements for either fields.

1.2. Purpose of this Thesis

So far, the realms of conventional and fuzzy classification rule learning have been researched separately. Researchers from the machine learning community developed rule-based classification algorithms with a notable attention on the discriminative abilities and to the efficient use of processor time and memory consumption. The focus of researchers from the fuzzy logic community emphasized the aspect of interpretability when developing linguistic fuzzy rule-based classification algorithms. As a result, both communities are relatively disjunct: There are no remarkable efforts to bring the best of both worlds together. The consequence is that in the world of machine learning the fuzzy rule-based approaches are not taken seriously due to their weak predictive qualities.

This thesis will revolve around the combination of conventional rule learning methods and techniques for fuzzifying the rules. Therefore, we will introduce procedures to soften conventional rule boundaries. We will investigate the consequences of that fuzzification process and analyze the differences between the original conventional and the fuzzy version.

1.3. Contribution of this Thesis

In this thesis the following aspects will find consideration:

New data-driven fuzzy rule-based classifiers The main contribution of this thesis will be the development of novel methods for fuzzy rule-based classification. We will introduce three algorithms, two will improve on an existing state-of-the-art conventional rule learner from the separate-and-conquer realm and one will be a completely novel approach that is related to both decision tree learning and data discretization techniques.

Effective fuzzification techniques The transformation of conventional into fuzzy rules plays the key role in this work. We introduce two procedures that soften existing rule boundaries making a conventional rule fuzzy: (A) a fuzzification strategy for ordinary classification rules that do not grid-partition the numeric attributes, (B) a fuzzification strategy for rules that partition the data set into a grid.

Deep investigation into the effects of rule fuzzification This thesis encompasses serious investigations into the effects of fuzzification. In the focus are the consequences that are caused by the fuzzification process. As we will see, fuzzification comes with side effects that influence the classification decision. In a step-by-step examination the side effects will be peeled off in order to go to the very heart of fuzzification influence.

Comprehensive experimental analysis In machine learning, benefits of new algorithms are typically proven experimentally in a statistically sound way. Due to the concentration on objective, quantitative characteristics — e.g. the predictive accuracy — a comparison of two or more classifiers is rather simple. This enables researchers to compare their algorithms on a large number of data sets without making the evaluation more complicated or time consuming. Researchers are encouraged to further improve their algorithms in order to be competitive. This governance is also a kind of selection that keeps the standard high. In this work, we analyze the introduced algorithms extensively using 45 data sets from different domains.

Apart from testing the classifiers on a large testbed we will evaluate their strengths and weaknesses. To this end, we will consider classification and ranking performance for investigating the discriminative power and e.g. the number of rules and the average rule length to measure the model complexity.

The analyses will encompass thorough statistical evaluations that allow us to reach a convincing and significant conclusion.

1.4. Publications in the Context of this Thesis

Parts of this thesis have already been published in international journals or at international conferences. The following list gives an overview of the publications related to the topics of this thesis.

- J.C. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. In L. Magdalena, M. Ojeda-Aciego and J.L. Verdegay, editors: *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU*, Torremolinos (Málaga), Spain, pages 1543–1550, June 22–27, 2008.
- J.C. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions Fuzzy Systems*, 17(1):138–149, 2009.
- J.C. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, 2009.
- J.C. Hühn and S.A. Vinterbo. HELLFIRE: Learning interpretable and effective fuzzy rule-based classification models. *Fuzzy Sets and Systems*, with editor.
- J.C. Hühn and E. Hüllermeier. An analysis of the FURIA algorithm for fuzzy rule induction. In J. Koronacki, Z. Ras, S.T. Wierzbach and J. Kacprzyk, editors, *Advances in Machine Learning I: Dedicated to the memory of Professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*. Springer, Berlin, Germany, 2010.

1.5. Software Developments in the Context of this Thesis

The main part of this thesis is related to the development of fuzzy rule-based classification algorithms. All three new algorithms — FURIA, HELLFIRE and FR3 — were implemented in JAVA for the WEKA machine learning framework [WF05]. These implementations are publicly available in the software repository of the Knowledge Engineering & Bioinformatics Lab at Marburg University¹:

<http://www.uni-marburg.de/fb12/kebi/research>

¹The HELLFIRE implementation will be made available for download as soon as the article is accepted for publishing.

Moreover, the FURIA algorithm will be part of the official WEKA package in a contribution later than 3.7.0. Today, it is already available in the nightly build via the WEKA subversion system:

`http://www.cs.waikato.ac.nz/~ml/weka/`

1.6. Outline

This thesis will be structured following the algorithms introduced. First of all, we will provide the foundation for this dissertation in Chapter 2. This chapter will encompass the necessary theoretical and methodical background for the remaining thesis. In the next three chapters, we will introduce three different fuzzy rule-based classification algorithms: The FURIA algorithm in Chapter 3, the HELLFIRE algorithm in Chapter 4 and the FR3 algorithm in Chapter 5. In Chapter 6 we will contrast the three algorithms both from a formal but also from an experimental point of view. A comprehensive survey of related work will be given in Chapter 7. We will conclude this dissertation and give an outlook on future work in Chapter 8.

2 Foundations

In this thesis, three different fuzzy rule-based classification algorithms will be proposed. However, first of all, we will provide the foundations necessary for understanding the later chapters without repeating ourselves.

In Section 2.1, we will explain the task of classification and will then introduce reduction techniques to binary problems in Section 2.2. In Section 2.3 and Section 2.4, we will focus on conventional and fuzzy rule-based classification, respectively. Finally, we will explain the experimental setup and statistical methods used throughout this dissertation in Section 2.5.

2.1. Classification

The classification task is a classic machine learning scenario: The challenge is to assign an instance with unknown class label a class membership that is based on training examples for which this information is known. The fascination of this task is that it can be applied to a legion of different problems with different characteristics, which range from being rather simple to being incredibly difficult to solve. Formally, the classification scenario can be described as follows:

Let

$$\mathbb{L} \stackrel{\text{df}}{=} \{\lambda_1, \dots, \lambda_m\}$$

for $m \geq 2$ be the set of class labels. Let

$$\mathbb{D} \stackrel{\text{df}}{=} A_1 \times \dots \times A_k$$

be the data space, where A_i is a nominal or numerical attribute. Let

$$\mathbf{x} \stackrel{\text{df}}{=} (x_1, \dots, x_k) \in \mathbb{D}$$

be an instance. Let

$$(\mathbf{x}, \lambda) \in \mathbb{D} \times \mathbb{L}$$

be an example. A data set

$$D \stackrel{\text{df}}{=} \{(\mathbf{x}_1, \lambda_{\mathbf{x}_1}), \dots, (\mathbf{x}_n, \lambda_{\mathbf{x}_n})\} \subseteq \mathbb{D} \times \mathbb{L}$$

is a collection of examples.

Let p be a discrete probability distribution on $\mathbb{D} \times \mathbb{L}$ with $p : 2^{\mathbb{D} \times \mathbb{L}} \rightarrow [0, 1]$. Extracting p from a data set D is called *inference*. This can be achieved by *learning* an inference model \mathcal{M} .

In classification, the task is to assign a class λ to an instance \mathbf{x} , for which this information is not known. The quality of such an assignment is measured by a loss function

$$L : \mathbb{L} \times \mathbb{L} \rightarrow \{\mathbb{R}^+ \cup \{0\}\} .$$

$L(\lambda, \bar{\lambda})$ measures the cost when λ is the true class and $\bar{\lambda}$ is the prediction. A typical loss function in classification is

$$L(\lambda, \bar{\lambda}) \stackrel{\text{df}}{=} \begin{cases} 0 & \text{if } \lambda = \bar{\lambda} \\ 1 & \text{else} \end{cases} .$$

A classification model $\mathcal{M} : \mathbb{D} \rightarrow \mathbb{L}$ is a function from the data space \mathbb{D} to the set of classes \mathbb{L} . The true risk R of \mathcal{M} as predictor of $p(\mathbb{D}, \mathbb{L})$ is defined using the expected loss as follows:

$$R(\mathcal{M}) \stackrel{\text{df}}{=} \sum_{(\mathbf{x}, \lambda) \in \mathbb{D} \times \mathbb{L}} L(\lambda, \mathcal{M}(\mathbf{x})) \cdot p(\mathbf{x}, \lambda)$$

Since probabilities $p(\mathbf{x}, \lambda)$ are unknown, the empirical risk \hat{R} can be deter-

mined from the known data set D only:

$$\hat{R}(\mathcal{M}) \stackrel{\text{df}}{=} \sum_{(\mathbf{x}, \lambda) \in D} L(\lambda, \mathcal{M}(\mathbf{x})) \cdot \frac{1}{|D|}$$

In order to minimize the true risk R , a classification model \mathcal{M} is sought by minimizing the empirical risk \hat{R} .

2.2. Binary Decomposition Techniques for Multi-Class Problems

The multi-class classification scenario is a very common problem in machine learning research. The task is to assign an instance with unknown class exactly one from $m > 2$ class labels. To make existing binary classifiers applicable to this kind of problem, decomposition techniques were conceived. These break down the original multi-class problem into multiple two-class problems.

2.2.1. 1-vs-All Decomposition

A very simple technique for decomposing a polychotomous classification problem into binary ones is *1-vs-All*. This means that the m -class problem is decomposed into m binary problems, where the respective classifier \mathcal{M}_i decides whether the query instance \mathbf{x} belongs to λ_i or not. In the optimal case, exactly one classifier \mathcal{M}_i returns true while the others return false. In this case the classification decision is in favor of λ_i . However, when there is a conflict — multiple classifiers returning true or all classifiers returning false — this conflict must be broken. A common solution for a scoring classifier is to choose the class with the highest score. For classifiers which do not return scores, such as rule learners, the conflict can be broken by selecting the most frequently observed class in the training data.

2.2.2. All-vs-All Decomposition

The notion of *pairwise learning*, aka *round robin learning*, *All-vs-All* or *1-vs-1* as an approach to solve polychotomous classification problems was introduced

by Friedman [Fri96]. The main idea behind pairwise learning is to decompose an m -class classification problem with $m > 2$ into $m(m-1)/2$ binary problems. The decomposition process creates pairwise problems that contain exactly two classes λ_i, λ_j with $1 \leq i < j \leq m$. To distinguish between instances with labels from either class a classifier $\mathcal{M}_{i,j}$ is trained. Therefore, an original example $(\mathbf{x}, \lambda_\alpha) \in D$, which means that \mathbf{x} belongs to class λ_α , is considered as a positive example for all models $\mathcal{M}_{\alpha,j}$, $\alpha < j$ and as a negative example for all models $\mathcal{M}_{i,\alpha}$, $i < \alpha$. At classification time, a query instance \mathbf{x} is submitted to all learners. The result of one of those base classifiers $\mathcal{M}_{i,j}(\mathbf{x})$ can be interpreted as a vote for label λ_i or λ_j . The simplest assumption is that the output of $\mathcal{M}_{i,j}$ is a real-valued score $s_{i,j} \in [0, 1]$. A score close to 1 could be considered as evidence in favor of λ_i , while a score close to 0 reflects a support of class λ_j . Given the $m(m-1)/2$ pairwise evidence, a classification decision could be inferred from the sum of votes

$$s_i(\mathbf{x}) \stackrel{\text{df}}{=} \sum_{1 \leq i \neq j \leq m} s_{i,j} \quad (2.1)$$

assuming that $s_{i,j} = \mathcal{M}_{i,j}(\mathbf{x})$ for $i < j$ and $s_{j,i} = 1 - \mathcal{M}_{j,i}(\mathbf{x})$ for $i > j$. The class λ_i maximizing $s_i(\mathbf{x})$ will be predicted.

A faster evaluation of the winner class was put forward by Park and Fürnkranz with *Quick Weighted Voting* [PF07]. This algorithm is able to reduce the quadratic number of comparisons to nearly linear in the number of classes during classification. The trick is to evaluate the classifier $\mathcal{M}_{i,j}$ next, where λ_i and λ_j have the smallest loss. In this regard, the loss is the sum of (weighted) votes λ_i and λ_j respectively. This procedure tries to avoid an evaluation of $\mathcal{M}_{k,\ell}$ where λ_k or λ_ℓ have no chance of winning the (weighted) voting ballot.

An analysis of weighted voting as aggregation technique for pairwise decomposition can be found in [HV10].

Friedman proposed to vote for the class which obtains the largest number of wins in the pairwise comparison [Fri96]. In contrast to (2.1) this strategy ignores the magnitude of the results.

A different take on aggregating the pairwise results — known as *vote-against* — was proposed by Cutzu [Cut03]. He reasoned that if a classifier $\mathcal{M}_{i,j}$ makes a prediction for an instance \mathbf{x} from class λ_k with $k \neq i, j$ it inevitably makes a mistake. However, when letting $\mathcal{M}_{i,j}$ vote against either λ_i or λ_j in the same situation, this would reduce the number of wrong pre-

dictions in comparison to conventional voting — assuming that the pairwise classifiers are trained properly.

Another strategy to combine the pairwise predictions $s_{i,j}$ is the *pairwise coupling* strategy introduced by Hastie and Tibshirani [HT97]. This approach follows the idea of Bradley and Terry that $s_{i,j}$ and $s_{j,i}$ with $s_{i,j} + s_{j,i} = 1$ are probability estimates [BT52]. In such a case, the question would be whether there is a probability distribution s_i which is compatible with the pairwise probability observations. However, a solution for this does not necessarily exist: The problem is over-constrained, since the number of probabilities s_i is m and the number of pairwise models is $m(m-1)/2$. Hastie and Tibshirani proposed to find pairwise probabilities \hat{s}_i such that the average weighted Kullback-Leibler distance δ between $s_{i,j}$ and $\hat{s}_{i,j} \stackrel{\text{df}}{=} \frac{\hat{s}_i}{\hat{s}_i + \hat{s}_j}$ is minimized. For this purpose they suggested an iterative method that strictly decreased δ . A more stable approach for this problem was put forward by Wu et al. who suggested to obtain the probabilities with the help of convex quadratic programming [WLW04].

An approach based on the Dempster-Shafer theory of evidence to combine pairwise classifiers was proposed by Quost et al. [QuM07]. The authors propose to combine the pairwise results as non-probabilistic evidence in favor of the classes involved. Comparable to the approach of pairwise coupling is the need to find a non-conditional belief function which is consistent with the belief functions from the binary classifiers.

2.2.3. Comparison of 1-vs-All and All-vs-All Decomposition

Fürnkranz found that the pairwise decomposition technique is superior to 1-vs-All decomposition. He explained this through smaller problems involving less examples which are simpler to separate in general [Für02, Für03]. Another revelation was made by Rifkin and Klautau who found that there might be no clear winner between all pairs and 1-vs-All, when the 1-vs-All algorithm is well-tuned [RK04].

One clear advantage of decomposition in pairwise learning is that it creates very small problems containing two classes only. In general, it holds that the smaller the number of classes, the easier they can be separated. Consequently, the decision boundaries for pairwise problems should be simpler than multi-class decision boundaries, cf. Figure 2.1 [Für03].

One drawback of the pairwise approach is that it needs a quadratic number

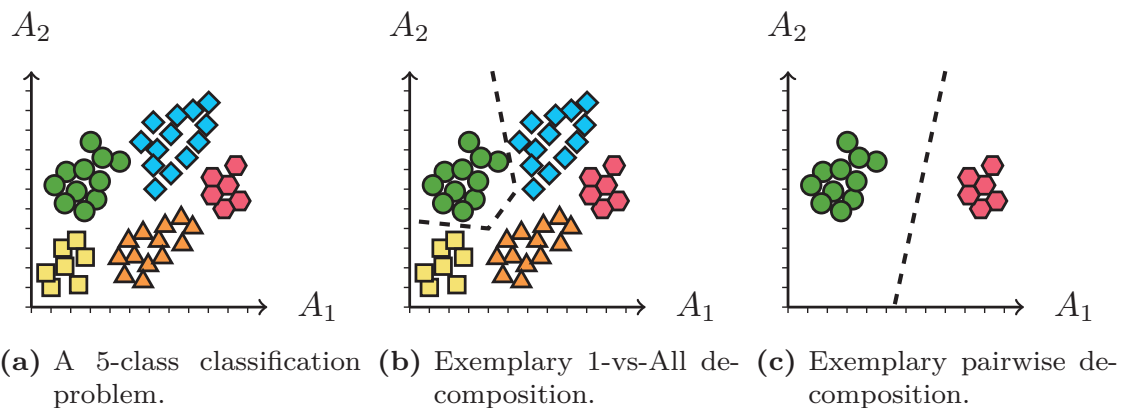


Figure 2.1.: Decision boundaries of different decomposition schemes according to Fürnkranz [Für03].

of $m(m-1)/2$ models in comparison to the m models in the 1-vs-All scheme. However, the total effort of building the classifier using pairwise decomposition is smaller than the effort necessary when using 1-vs-All, assuming that the base learner has at least super-linear complexity. This can be explained by the decreased pairwise problem sizes. Interestingly, when the complexity of the learning algorithm rises, the gain of pairwise learning in comparison to 1-vs-All will increase [Für02].

Decomposition techniques for reducing multi-class to binary classification problems have been investigated quite extensively in recent years. Many standard decomposition schemes, including the all-pairs (round robin) and the 1-vs-All scheme (cf. Section 2.2), are special cases of the more general approach of *Error Correcting Output Codes (ECOC)* [DB95] or, more precisely, their generalization that has been introduced by Allwein et al. [ASS01].

Even though ECOC allows for a more flexible decomposition of the original problem into simpler ones, the all-pairs approach has the advantage that it provides a fixed, domain-independent and non-stochastic decomposition with a good overall performance. In several experimental studies, including the work of Allwein et al. [ASS01], it performed en par or better with competing decoding matrices.

2.3. Rule-Based Classification

The idea in rule-based classification is to learn a model that generalizes the data with simple IF-THEN-rules. A single classification rule typically has the form $r = \langle r_A \mid r_C \rangle$, consisting of an antecedent part r_A and a consequent part r_C . In the realm of conventional rule-based classification, the antecedent part is a logic expression that combines conditions constraining the attributes. For a numerical attribute A , a condition can be expressed in the form $v \in I$, where $I = [b, c] \subset A$. A value v is covered if and only if $b \leq v \leq c$ holds. For nominal attributes, the conditions are of the form $v \theta A$ for $\theta \in \{=, \neq\}$. To distinguish this interval-based rule and antecedent from other ones introduced below, we write r^I and r_A^I , respectively. Formally, we consider antecedents

$$r_A^I \stackrel{\text{df}}{=} (A_{i_1} \in I_{i_1,1} \vee \dots \vee A_{i_1} \in I_{i_1,i_k}) \wedge \dots \wedge (A_{j_1} \in I_{j_1,j_1} \vee \dots \vee A_{j_1} \in I_{j_1,j_\ell}) ,$$

where A_i is an attribute, $\{i_1, \dots, i_k\} \subseteq \{1, \dots, |I_i|\}$ are indices of occurring intervals and $|I_i|$ is the number of intervals for A_i .

The consequent part r_C is a class assignment of the form $(\text{class} = \lambda)$, where $\lambda \in \mathbb{L}$. A rule $r^I = \langle r_A^I \mid r_C \rangle$ is said to *cover* an instance $\mathbf{x} = (x_1, \dots, x_n)$ if the antecedent r_A^I evaluates to true.

2.3.1. Separate-and-Conquer Rule Learning

In order to learn a well-performing classification model effectively, the idea of separate-and-conquer rule learning has been popular throughout the last decades [PH90, Für99]. The general strategy of separate-and-conquer learning (cf. Algorithm 1) is to create a rule r^I for a specific class λ_i that explains a part of the training set D by covering examples $\{(\mathbf{x}, \lambda_i) \in D\}$ from the same class: The *separating* step. Those examples are removed from D , such that the process may be repeated to cover the remainder: The *conquering* step. This procedure creates a single rule in every loop thus forming a ruleset $RS^I \stackrel{\text{df}}{=} \{r_1^I, \dots, r_k^I\}$.

An important distinction, when dealing with rule-based classifiers, is the one between a list of rules and a ruleset. In a list of rules the order, in which the rules were learned, plays an important role during classification: The first rule covering the query instance is the one which decides the classification in favor of its class, while subsequently learned rules will be ignored. Typically, for the largest class there is one rule only which has an empty antecedent and

Algorithm 1 Separate-And-conquer(D)

```

1:  $RS \leftarrow \emptyset$ 
2: for  $i = 1, \dots, m$  do
3:    $D_{\text{Grow}} \leftarrow D$ 
4:   while  $\{(\mathbf{x}, \lambda_i) \in D_{\text{Grow}}\} \neq \emptyset$  do
5:     // Separate:
5:     Find rule  $r = \langle r_A \mid \lambda_i \rangle$  covering  $D_r$  such that  $D_r \cap D_{\text{Grow}} \neq \emptyset$ 
6:      $RS \leftarrow RS \cup r$ 
7:     // Conquer:
7:      $D_{\text{Grow}} \leftarrow D_{\text{Grow}} \setminus \{(\mathbf{x}, \lambda_j) \in D_{\text{Grow}} \mid r_A(\mathbf{x}) = \text{TRUE}\}$ 
8:   end while
9: end for
10: return  $RS$ 

```

which is placed at the very end of the list. Instances that are not covered by earlier rules are then covered and classified by this *default rule*.

On the contrary, the learning order is of no effect in a ruleset: Every rule covering the query instance can have an influence on the classification decision. In fact, only when rules from one class cover the query instance, a clear decision can be made. Otherwise, if rules from different classes cover the instance, a remedy must be found, e.g. by letting the rules that cover the query instance vote in favor of their consequent classes. If no rule covers the instance, a typical solution is to select the most frequent class in the training data.

The concept of separate-and-conquer learning introduced so far is too general yet. To make it applicable in practice, one has to define the learning bias that makes the generalization possible. According to Fürnkranz, three different types of bias for a separate-and-conquer learner exist [Für99]:

Language Bias The space of hypotheses, which contains concepts that may describe the training data, is constituted by the hypothesis language. This language sets a frame of which concepts are allowed and of which are not. Consequently, it is important to select a hypothesis language that is able to model a well-generalizing concept. This makes the hypothesis language an important bias for rule learning.

Search Bias Since the hypothesis space is typically too large to be enumerated, a good concept must be searched selectively. To make such a search fast and effective, it has to be guided by an appropriate search bias.

Overfitting Avoiding Bias The training set for a learning problem is typically of limited size, while the number of possible concepts that describes the data flawlessly is unlimited. Yet, such a perfect concept becomes problematic, if the data is noisy or when the training sample is too small to reflect the true distribution. A well fitted model reflecting the setbacks of the training has a poor generalizing performance on unseen data. In concordance with Occam's razor, the overfitting avoiding bias should prefer the simple models to more complex ones.

Fürnkranz pointed out that the latter bias is — strictly seen — a form of search bias. His argument to list it separately is due to the fact that in separate-and-conquer algorithms both types of bias are applied more or less independently.

2.3.2. Rule Learning with FOIL, REP and IREP

One representative algorithm of the separate-and-conquer realm is the *FOIL*-algorithm, which was presented by Quinlan [Qui90]. It is able to learn a logical concept description from relational data. FOIL uses two nested loops for learning. The outer loop is the classical separate-and-conquer loop that tries to explain a part of the training data, removes that part and then starts over to explain the rest. The inner loop of FOIL builds a single rule by greedily adding literals to a function-free Horn clause. This is done by adding one literal at a time according to a selection criterion.

The original purpose of FOIL was to learn general logical descriptions. Therewith it is also able to learn classification rules. This can be done by replacing the notion of the function-free Horn clause with a rule antecedent r_A and by using a conjunction of predicates (or “selectors” [Mic73]) $A_i \theta v_i$ as literals given $\theta \in \{\leq, \geq\}$. Note that this type of rule can also be written as an interval-based rule r^I using

$$A_i \leq v_i \Leftrightarrow A_i \in (-\infty, v_i]$$

and

$$A_i \geq v_i \Leftrightarrow A_i \in [v_i, \infty) .$$

Algorithm 2 FOIL-Separate-And-conquer(D)

```

1:  $RS \leftarrow \emptyset$ 
2: for  $i = 1, \dots, m$  do
3:    $D_{\text{Grow}} \leftarrow D$ 
4:   while  $\{(\mathbf{x}, \lambda_i) \in D_{\text{Grow}}\} \neq \emptyset$  do
5:      $r^I \leftarrow \langle \text{TRUE} \mid \lambda_i \rangle$ 
6:      $D_{\text{Local}} \leftarrow D_{\text{Grow}}$ 
7:     while  $\{(\mathbf{x}, \lambda_j) \in D_{\text{Local}} \mid i \neq j\} \neq \emptyset$  do
8:       Find selector  $A_k \theta v_k$  according to some criterion
9:        $D_{\text{Local}} \leftarrow \{(\mathbf{x}, \lambda) \in D_{\text{Local}} \mid \mathbf{x} = \{x_1, \dots, x_n\} \wedge A_k \text{ covers } x_k\}$ 
10:       $r^I \leftarrow \langle r_A^I \wedge A_k \theta v_k \mid \lambda_i \rangle$ 
11:    end while
12:     $RS \leftarrow RS \cup r^I$   $D_{\text{Grow}} \leftarrow D_{\text{Grow}} \setminus \{(\mathbf{x}, \lambda_j) \mid r_A^I(\mathbf{x}) = \text{TRUE}\}$ 
13:  end while
14: end for
15: return  $RS$ 

```

Algorithm 2 shows a separate-and-conquer algorithm, which uses the FOIL strategy to grow the rules. Note that FOIL never uses disjunctions as operators. This assures that the rule coverage monotonically shrinks with the rule length.

Let p and n be the number of positive and negative examples in D_{Local} , respectively. The information, which is required to signal that among $(p+n)$ examples are p positives is given by:

$$I(D_{\text{Local}}) \stackrel{\text{df}}{=} -\log_2 (p / (p+n)) \quad (2.2)$$

Quinlan proposed the *Information Gain* as a criterion to evaluate the utility of the i^{th} condition in the rule:

$$\text{Gain}(D_{\text{Local}_i}, D_{\text{Local}_{i-1}}) \stackrel{\text{df}}{=} |D_{\text{Local}_i}| \times (I(D_{\text{Local}_i}) - I(D_{\text{Local}_{i-1}})) \quad (2.3)$$

The Information Gain describes how much information was won by adding up another condition to the original rule.

To avoid overfitting in noisy domains, Quinlan added a premature stopping criterion to the learning process. For the case that the coverage of additional

examples is more costly than the description of the examples themselves in terms of the *Minimum Description Length (MDL)* [Ris83], the learning process will be aborted.

Another way to avoid overfitting due to noise was proposed by Brunk and Pazzani: The *Reduced Error Pruning (REP)* is a post-pruning procedure for a model learned by FOIL [BP91]. The authors proposed splitting the training data into a growing set that is used for learning the model and a pruning set to cut selectors from the rule antecedent. The pruning procedure works in a simple, hill-climbing fashion by trying to maximize the accuracy on the pruning set. Therefore, the algorithm evaluates the removal of the last condition or of a whole rule. REP keeps removing conditions and rules until a further simplification would lead to an accuracy reduction.

Very serious issues with the REP algorithm were found by Fürnkranz and Widmer [FW94]: (A) Complexity of REP is $\Omega(n^4)$ which is worse than the complexity for learning with $\Omega(n^2 \log n)$ [Coh93]. (B) The rule learning procedure fails to exploit the information in the pruning data and makes pruning mistakes when the split into growing and pruning data was somehow unfortunate. (C) Post-pruning of a rule might affect other rules as well. These other rules would be learned using the “wrong” examples and, thus, contain the wrong conditions, (D) due to the immense search space of REP, the pruning is likely to get stuck in a local optimum.

To redeem the shortcomings of REP, Fürnkranz and Widmer proposed the *Incremental Reduced Error Pruning (IREP)* algorithm [FW94]. The strategy is to prune every rule right after learning in a greedy way. In contrast to REP, the removal of any condition is allowed instead of only the last one. Apart from this, IREP uses the pruning also as a stopping criterion for the rule learning process. As soon as the empty rule has a better accuracy than any of the pruned variants, the learning is stopped. This tackles the mentioned problems of REP in the following ways: (A) Complexity goes down to $\Omega(n \log^2 n)$ due to the rule-wise pruning strategy. (B) The failure to exploit the information in the pruning data is reduced from the model learning level to the rule learning level and, thus, (C) The IREP pruning is done after learning a rule and, to avoid affecting other, already learned rules. (D) The risk of getting stuck in a local optimum is reduced since there is no overly complex model as a starting point for a complicated pruning problem.

2.3.3. RIPPER

The *RIPPER* algorithm was introduced by Cohen as a successor to the IREP algorithm [Coh95].

The result of RIPPER — and also of most conventional rule learners — is a decision list. To produce such a list, rules are learned for each class in turn, starting with the smallest (in terms of relative frequency of occurrence in the training data) and ending with the second largest one. Finally, a default rule is added for the majority class. A new query instance is then classified by the first rule in the list by which it is covered.

Cohen's modifications to the IREP algorithm were more of evolutionary than of revolutionary nature. There are three main changes: (A) The pruning phase is updated with a new metric. (B) A new stopping criterion is introduced and (C) an optimization strategy is added.

2.3.3.1. Pruning Modifications

Pruning is changed in several ways. In contrast to IREP, RIPPER was allowed to not only prune one condition from a rule, but instead may prune any final sequence of conditions from a rule. The criterion to find the position at which the rule is cut is the rule-value metric

$$V(r) \stackrel{\text{df}}{=} \frac{p_r - n_r}{p_r + n_r} .$$

Therewith all antecedents that were learned after the antecedent maximizing $V(r)$ will be pruned. Shorter rules are preferred in the case of a tie.

The motivation for this new criterion is that IREP is sometimes unable to converge when the training data increases. According to Cohen, this is due to the original pruning criterion, which prefers a rule which covers 2000 positive examples and 1000 negative examples to a rule covering 1000 positive examples and only 1 negative example — a quite unintuitive behavior.

2.3.3.2. New Stopping Criterion

The stopping criterion of IREP ceases the learning process, when the last rule constructed has an error rate larger than 50%. Cohen found that this is often too early, especially for low coverage rules which may have an error that is larger only by chance. This stopping criterion seems to be too sensitive to the “small disjunct problem” [HAP89].

The proposed solution uses the idea of description length as stopping criterion [Qui93, Qui95]: The learning procedure is aborted in case of the last learned rule having been too specific. This is the case if the total description length of the ruleset is more than d bits larger than the shortest description found so far. Cohen proposes to set $d = 64$.

2.3.3.3. Rule Optimization

The two changes proposed above improve the classification accuracy of IREP significantly. In order to distinguish the novel version, it is denoted as *IREP**. Nevertheless, the optimization of the rules is the most significant change from IREP to RIPPER.

The ruleset RS , which was learned by *IREP** so far, is taken as a starting point for a subsequent optimization process. This process re-examines the rules $r_i \in RS$ in the order in which they were learned. For each r_i , two alternative rules r'_i and r''_i are created. The *replacement rule* r'_i is an empty rule, which is grown and pruned in a way that minimizes the error of the modified ruleset $(RS \cup \{r'_i\}) \setminus \{r_i\}$. The *revision rule* r''_i is created in the same way, except that it starts from r_i instead of the empty rule. To decide which version of r_i to retain, the MDL criterion is used [Qui93]. Afterward, the remaining positives are covered using the *IREP** algorithm.

The *RIPPER k* algorithm iterates the optimization of the ruleset and the subsequent covering of the remaining positive examples with *IREP** k times, hence the name *RIPPER* (*Repeated Incremental Pruning to Produce Error Reduction*). See Algorithm 3 for a simplistic overview which omits unnecessary details.

The classification of new instances having an unknown class label with the list of rules learned by *RIPPER* is done as follows: The rules are scanned in the order they were learned. The first rule covering the query instance decides the classification in favor of its consequence class. Rules subsequently learned will be ignored. If no learned rule covers the example, then the final default rule, having as consequence the label of the most frequent class in the training data, determines the classification.

Algorithm 3 $\text{RIPPER}_k(D)$

```

1:  $RS \leftarrow \emptyset$ 
2: Sort classes  $\{\lambda_1, \dots, \lambda_m\}$  by class frequency ascendingly
3: for  $i = 1, \dots, m - 1$  do
4:   // Build phase
5:    $RS \leftarrow RS \cup \text{GrowAndPruneRulesWithIREP}^*(D)$ 
6:   // Optimization phase
7:   for  $j = 1, \dots, k$  do
8:     // Consider replacement  $r'$  and revision  $r''$  rules as alternatives
9:      $RS \leftarrow \text{GrowAndPruneRuleAlternativesWithIREP}^*(RS, D)$ 
10:    // Cover remaining positive instances
11:     $RS \leftarrow RS \cup \text{CoverRemainderWithIREP}^*(D)$ 
12:   end for
13: end for
14:  $RS \leftarrow RS \cup \langle \text{TRUE} \mid \lambda_m \rangle$ 
15: return  $RS$ 

```

2.4. Fuzzy Rule-Based Classification

A different way of dealing with classification problems is to use fuzzy instead of conventional rules. Even though both types are rules, there are significant differences between these two approaches:

Fuzzy rules are more general than conventional rules and have a number of advantages. For example, conventional (non-fuzzy) rules produce models with “sharp” decision boundaries resulting in two arbitrarily close points being assigned to different classes. This problem is compounded by the boundaries often being chosen arbitrarily in some interval. Alternatively, one could expect the support for a class provided by a rule to decrease from “full” (inside the core of the rule) to “zero” (near the boundary) in a gradual rather than an abrupt way. Fuzzy rules have “soft” boundaries, which is one of their main characteristics. In addition, conventional boundaries constructed from combinations of fuzzy ones are potentially more flexible. For example, by using suitable operators for combining fuzzy rules, those boundaries are not necessarily axis-parallel [PFTV92].

2.4.1. Fuzzy Logic

Fuzzy logic was introduced by Zadeh as an extension to the classical binary (or *Boolean*) logic [Zad65]. While in Boolean Logic a variable can be either true or false, in fuzzy logic it can assume arbitrary degrees of truth. The advantage of this concept is the ability to cope with imprecise definitions that cannot describe unambiguously which expressions are true and which are not.

Boolean logic can be connected with classical set theory using a so-called indicator function

$$\mathbb{I}_A(v) \stackrel{\text{df}}{=} \begin{cases} 1 & \text{if } v \in A \\ 0 & \text{if } v \notin A \end{cases} ,$$

which states whether element v belongs to the set A or not.

However, this sharp distinction is problematic, when both $\mathbb{I}_A(v) = 0$ and $\mathbb{I}_A(v) = 1$ seem to be inappropriate. Consider the following example:

Example 2.4.1 (Sharp and not so sharp descriptions)

The description “people taller than 185 cm” is a sharp set description, because every person’s body height can be precisely determined with a yardstick for example. Either the body height is taller than 185 cm or not. In contrast, the description “tall people” is not sharp. Here, it is not so clear whether a person belongs to this set or not. The reason in this example is the lack of a clear threshold value. In such a case one might think that a person with a body height of 180 cm might be “rather tall” and, thus, belong to the concept of “tall people” only partially.

2.4.1.1. Fuzzy Set

As an extension to the classical set, Zadeh introduced the *fuzzy set*. Let $v \in X$, where X is the reference set. For this work we assume $X = \mathbb{R}$. To denote to which degree the value v belongs to a fuzzy set A , the *membership function* $\mu_A(v)$ was introduced.

Definition 1 (Membership Function)

A fuzzy set A of the real numbers is characterized by its membership function

$$\mu_A : \mathbb{R} \rightarrow [0, 1] ,$$

which maps the reference set \mathbb{R} to the unit interval. $\mu_A(v)$ describes to which

degree v belongs to A . As an abbreviation

$$A(v) \stackrel{\text{df}}{=} \mu_A(v)$$

will be used.

The membership function μ is a generalization of the indicator function \mathbb{I} .

Example 2.4.2 (Fuzzy sets of sharp and not so sharp descriptions)

The two notions of tall people that were given in the previous example need different membership functions to be represented as fuzzy sets. The set for the description of “people taller than 185 cm” results in the indicator function

$$\mathbb{I}_{>185 \text{ cm}}(v) = \begin{cases} 1 & v > 185 \text{ cm} \\ 0 & \text{else} \end{cases}$$

that is depicted in Figure 2.2a.

The fuzzy set for “tall people” can be described through the following membership function

$$\mu_{\text{tall people}}(v) = \begin{cases} 1 & v > 190 \text{ cm} \\ \frac{v-180}{10} & 180 \leq v \leq 190 \\ 0 & \text{else} \end{cases}$$

that is able to express memberships gradually, as it can be seen in Figure 2.2b. Of course, since there exists an infinitive number of reasonable membership functions for the concept of “tall people”, it is clear that this is only one example.

The exemplary membership functions show the advantage of fuzzy sets in comparison to ordinary sets. In the latter framework, a person with a body height of 185.1 cm might be considered as tall, while a person with a body height of 185 cm does not belong to this group. This transition from “full” to “zero” membership due to the difference of only 0.1 cm is — in this case — unnatural to distinguish tall from other people. In general, the difference between two instances from which one belongs to a certain set, while the other does not, could be arbitrarily small. A more natural characteristic would be that points lying near to each other should also have similar membership degrees. In the example this could mean that a person with a height of 185 cm belongs to the concept “tall people” to the degree of 0.5, while a slightly taller person with 185.1 cm has a membership of 0.51.

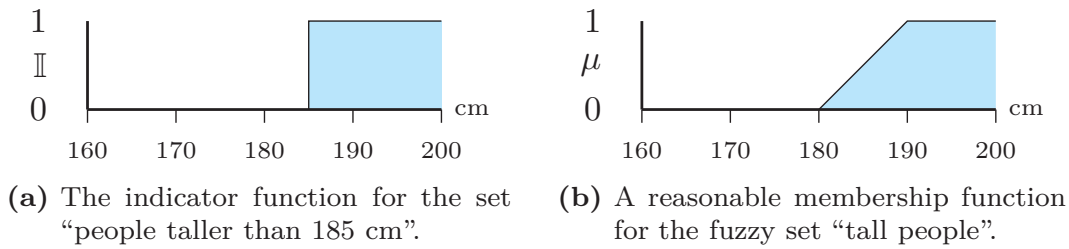


Figure 2.2.: Membership representations for concepts of “tall people”.

The flexibility of fuzzy membership functions allows unlimited shapes of fuzzy sets. Yet, only some of them fulfill the following demands:

Definition 2 (Fuzzy Set Normalization)

A fuzzy set A is said to be *normalized* if and only if there exists a $v \in \mathbb{R}$ such that $\mu_A(v)=1$.

Definition 3 (Fuzzy Set Convexity)

A fuzzy set A is said to be *convex* if and only if $\mu_A(\lambda \cdot v_1 + (1 - \lambda) \cdot v_2) \geq \min(\mu_A(v_1), \mu_A(v_2))$ for all $v_1, v_2 \in \mathbb{R}$ and $\lambda \in [0, 1]$ holds.

Definition 4 (Fuzzy Set Continuity)

A fuzzy set A is said to be *continuous* if and only if its membership function $\mu_A(\cdot)$ is continuous.

There exists an abundance of fuzzy set shapes that can be described through their membership function. However, for our purposes, we concentrate on fuzzy intervals which are simple, trapezoidal fuzzy sets:

Definition 5 (Fuzzy Interval)

A fuzzy interval $I^F = [a, b, c, d]$ is a fuzzy set with a trapezoidal membership function of the form

$$I^F(v) \stackrel{\text{df}}{=} \mu_{I^F}(v) = \begin{cases} 1 & b \leq v \leq c \\ \frac{v-a}{b-a} & a < v < b \\ \frac{d-v}{d-c} & c < v < d \\ 0 & \text{else} \end{cases},$$

where b and c are, respectively, the lower and upper bound of the core of the fuzzy set; likewise, a and d are, respectively, the lower and upper bound of the support.

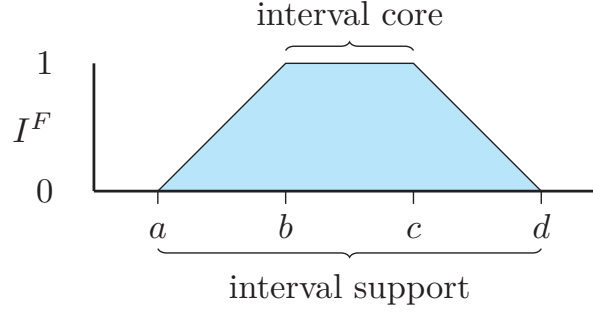


Figure 2.3.: A fuzzy interval I^F .

Figure 2.3 depicts the membership function for a general fuzzy interval. The fuzzy interval is normalized, convex and also continuous.

The fuzzy interval $I^F = [b, b, c, c]$ can be seen as a generalization of the conventional interval $I = [b, c]$. As it will be shown below, this characteristic will be helpful for transforming conventional rules into fuzzy ones.

In practice, the use of fuzzy logic is often seen with a partition of the data space into several fuzzy sets.

Definition 6 (Fuzzy Partition)

A fuzzy partition of an attribute is a set of fuzzy sets $\{A_1, \dots, A_k\}$ on that attribute.

This definition of a fuzzy partition is very general and a single fuzzy set can already be considered to be a fuzzy partition. As it was already the case for the fuzzy set, certain characteristics of fuzzy partitions exist as well:

Definition 7 (Fuzzy Partition Completeness)

A fuzzy partition $\{A_1, \dots, A_k\}$ is said to be complete if and only if for all $v \in \mathbb{R}$ there exists a $A \in \{A_1, \dots, A_k\}$ such that $\mu_A(v) > 0$ holds.

Definition 8 (Ruspini Fuzzy Partition)

A fuzzy partition $\{A_1, \dots, A_k\}$ is said to be a Ruspini partition if and only if $\mu_{A_1}(v) + \dots + \mu_{A_k}(v) = 1$ holds for all $v \in \mathbb{R}$.

2.4.1.2. Fuzzy Operators

The extension from a conventional set to a fuzzy set makes it necessary to also extent the Boolean logic to a fuzzy logic. Consequently, it is necessary

to introduce fuzzy operators that are able to cope with the notion of gradual set memberships.

As a fuzzy AND-operator the *T-norm* will be used:

Definition 9 (Fuzzy T-norm)

Let $\top : [0, 1] \times [0, 1] \rightarrow [0, 1]$ be a function. \top is called T-norm if the following conditions hold for all $0 \leq a, b, c \leq 1$:

$\top(a, 1) = a$	Neutral element
$\top(a, \top(b, c)) = \top(\top(a, b), c)$	Associativity
$\top(a, b) = \top(b, a)$	Commutativity
$\top(a, b) \leq \top(c, b), \text{ if } a \leq c$	Monotonicity

Example 2.4.3 (Fuzzy T-norms)

$\top(a, b) = \min(a, b)$	Minimum
$\top(a, b) = a \cdot b$	Product
$\top(a, b) = \max(0, a + b - 1)$	Łukasiewicz

As a fuzzy OR-operator the *T-conorm* will be used:

Definition 10 (Fuzzy T-conorm)

Let $\perp : [0, 1] \times [0, 1] \rightarrow [0, 1]$ be a function. \perp is called T-conorm if the following conditions hold for all $0 \leq a, b, c \leq 1$:

$\perp(a, 0) = a$	Neutral element
$\perp(a, \perp(b, c)) = \perp(\perp(a, b), c)$	Associativity
$\perp(a, b) = \perp(b, a)$	Commutativity
$\perp(a, b) \leq \perp(c, b), \text{ if } a \leq c$	Monotonicity

Example 2.4.4 (Fuzzy T-conorms)

$\perp(a, b) = \max(a, b)$	Maximum
$\perp(a, b) = a + b - a \cdot b$	Product
$\perp(a, b) = \min(1, a + b)$	Łukasiewicz

Finally, the fuzzy negation operator is introduced.

Definition 11 (Fuzzy negation)

Let $\neg : [0, 1] \rightarrow [0, 1]$ be a function. \neg is called fuzzy negation if the following conditions holds:

$$\begin{array}{ll} \neg(1) = 0 \text{ and } \neg(0) = 1 & \text{Boundary condition} \\ \neg(a) \leq \neg(b) \text{ if } b \leq a & \text{Monotonicity} \end{array}$$

Example 2.4.5 (Fuzzy negation)

$$\neg(a) = 1 - a \quad \text{Zadeh}$$

Definition 12 (De Morgan triplet)

Let \top a fuzzy T-norm, \perp a fuzzy T-conorm and \neg a fuzzy negation. (\top, \perp, \neg) is a De Morgan triplet if for all $u, v \in [0, 1]$ the following conditions hold:

$$\begin{array}{l} \neg \top(u, v) = \perp(\neg u, \neg v) \\ \neg \perp(u, v) = \top(\neg u, \neg v) \end{array}$$

Using T-norm, T-conorm and fuzzy negation the following fuzzy set operations can be introduced:

Definition 13 (Fuzzy set union)

Let A, B be two fuzzy sets, \perp a fuzzy T-conorm and $v \in \mathbb{R}$. Then the fuzzy set union \cup_{\perp} is defined as:

$$\mu_{A \cup_{\perp} B}(v) \stackrel{\text{df}}{=} \perp(\mu_A(v), \mu_B(v))$$

Definition 14 (Fuzzy set intersection)

Let A, B be two fuzzy sets, \top a fuzzy T-norm and $v \in \mathbb{R}$. Then the fuzzy set intersection \cap_{\top} is defined as:

$$\mu_{A \cap_{\top} B}(v) \stackrel{\text{df}}{=} \top(\mu_A(v), \mu_B(v))$$

Definition 15 (Fuzzy set complement)

Let A be a fuzzy set, \neg a fuzzy negation and $v \in \mathbb{R}$. Then the fuzzy set complement \overline{A} is defined as:

$$\mu_{\overline{A}}(v) \stackrel{\text{df}}{=} \neg \mu_A(v)$$

2.4.2. Fuzzy Classification Rules

The general form of a fuzzy classification rule $r^F = \langle r_A^F | r_C \rangle$ resembles the interval-based classification rule r^I introduced in Section 2.3. The main difference is the antecedent part r_A^F which is now a fuzzy logic expression for which the Boolean logic operators have been replaced with their fuzzy logic counterparts. For this work we consider only fuzzy rules with antecedents in conjunctive normal form (CNF):

$$r_A^F \stackrel{\text{df}}{=} (A_i \in I_{i,i_1}^F \perp \dots \perp A_i \in I_{i,i_k}^F) \top \dots \top (A_j \in I_{j,j_1}^F \perp \dots \perp A_j \in I_{j,j_\ell}^F) ,$$

where A_i is an attribute, $\{i_1, \dots, i_k\} \subseteq \{1, \dots, |I_i^F|\}$ are indices of occurring fuzzy intervals and $|I_i^F|$ is the number of fuzzy intervals for A_i .

In contrast to the conventional case an instance can be partially covered by a fuzzy rule. This is denoted as coverage degree or activation level

$$\mu_{r^F}(\mathbf{x}) \stackrel{\text{df}}{=} (I_{i,i_1}^F(x_{i_1}) \perp \dots \perp I_{i,i_k}^F(x_{i_k})) \top \dots \top (I_{j,j_1}^F(x_{j_1}) \perp \dots \perp I_{j,j_\ell}^F(x_{j_\ell})) , \quad (2.4)$$

where $I_{i,i_j}^F(x_{i_j})$ is the membership of the j -th fuzzy interval of attribute A_i .

2.4.3. Linguistic Fuzzy Classification Rules

The idea behind linguistic fuzzy classification rules is closely related to fuzzy rules in general. Rules of these kind use fuzzy sets with attached linguistic meanings for the given attributes, e.g. the set “tall people” that was introduced on the measurements of body heights in Example 2.4.1. The origin of the fuzzy set definitions and the attached labels can be a domain expert or a data-driven technique [INN05, pp. 7,8]. While the domain expert can ensure that fuzzy sets and attached labels are appropriate, the data-driven way is unable to achieve this. For the latter, the common approach is to discretize numeric attributes and to attach labels in a generic way *low*, ..., *high* as it is exemplary shown in Figure 2.4. This semantical drawback is problematic for all fuzzy set definitions that are not legitimated by domain-specific knowledge. This holds for data-driven fuzzy sets and partitions but also for homogeneous ones. Consequently, one might argue that only legitimated fuzzy sets might be considered as truly linguistic fuzzy sets. Ishibuchi et al. claimed that a homogeneous (or generic) fuzzy partition instead of a data-driven one can be “more easily understood by humans” [INN05, p. 8].

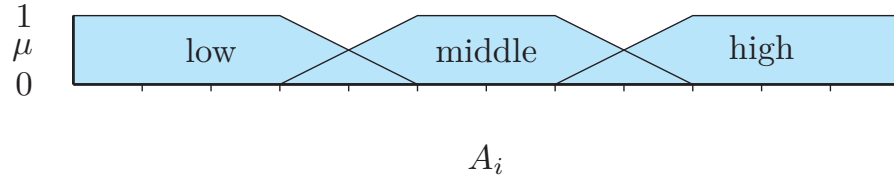


Figure 2.4.: An exemplary fuzzy partition with generic linguistic labels for each fuzzy set.

A discussion of this claim would be beyond the scope of the thesis, but it is important to mention that Ishibuchi et al. used the minimum value and the maximum value of the numeric domains as a starting point for introducing the generic partition. Ironically, these values have to be determined from the known data and, thus, the generic partition is data-driven none the less. With an increasing amount of data this weakness might be mitigated but it remains prone to outliers. Moreover, such a generic partition is problematic, when the number of fuzzy sets is too small or too large to model the underlying pattern.

In summary, it is important to underline that linguistic fuzzy classification rules which are obtained in a data-driven way might not be legitimated through a domain expert. These fuzzy sets and labels might be inappropriate for the considered domain.

In order to obtain readable rules, the fuzzy set descriptions are replaced by their linguistic meaning and the T-norm and T-conorm are replaced with *and* and *or*. The development of methods to learn linguistic fuzzy rule-based classifiers has been a very hot topic since the work of Wang and Mendel in 1992, who introduced the idea using fuzzy grid-partitioning [WM92]. The reason for the popularity of these models are manifold. Not only is understanding a classification decision very attractive, but the option to intervene or modify the decision in an informed way is what makes interpretable models special in real-life situations. This is arguably the main strength and it is this that distinguishes them from black box approaches. Linguistic fuzzy rule-based classifiers are not only interpretable, but also very readable. Instead of working on attributes directly, those classifiers make use of linguistic terms.

The grid-partitioning of linguistic fuzzy rule-based classifiers is both a blessing and a curse. While the grid is a corset which makes a tight fitting of the model to the training data difficult, it is this that helps avoiding

overfitting effects. The use of grid cells in the antecedents of a classification rule has the advantage of reducing the number of distinct conditions. This is due to the fact that non-grid-based rules fit the fuzzy sets on a per-rule basis. Consequently, it is likely that fuzzy sets, which could be represented through the same grid cell, are slightly different. However, aggregating these fuzzy sets might be rather hard. Furthermore, when using a grid-partitioning of an attribute it is much simpler to comply with demands such as the Ruspini partition or the completeness criterion, see above.

2.4.4. Fuzzy Reasoning Methods

The evaluation of conventional classification rules is of course non-trivial but also not too difficult. The reasoning becomes a little more complex when allowing variable degrees of coverage. In such fuzzy cases not only gradual memberships themselves, but generalized logical operators allow a larger flexibility when evaluating fuzzy rules and making classification decisions. The distinction between an ordered list of rules and an unordered ruleset, as it was made in Section 2.3.1, is irrelevant. The case of lists can typically be ignored because every rule is evaluated independently. The question whether it applies to a gradually covered instance leads to an imminent defuzzification which could also be reproduced with a conventional rule. For this reason, fuzzy reasoning deals only with rulesets for classification.

When considering fuzzy reasoning methods for classification with fuzzy rules there are two dominant schemes for obtaining classification decisions [INN05]: (A) single-winner, (B) voting.

In the single-winner scenario a classification decision is caused by one rule r^F only, namely the one that covers the instance \mathbf{x} to the largest degree $\mu_{r^F}(\mathbf{x})$. Other rules are completely ignored in this setting such that overlapping rules are not taken into consideration. This makes understanding classification decisions simple, but it has the cost of decreased predictive performance [CdH99].

The voting technique for obtaining classification decisions with fuzzy classification rules is quite similar to the one proposed for conventional rulesets, cf. Section 2.3.1. The major difference is that in the fuzzy case the gradual coverages lead to gradual votes in this kind of ballot. In order to obtain results in the unit interval, the scores can be normalized. Alternatively, T-conorms can be used to aggregate the gradual votes — making the normalization step superfluous.

A full account of the fuzzy reasoning methods used for the novel algorithms will be separately given in the subsequent chapters.

Cordon et al. employed an analysis of different fuzzy reasoning methods related to voting [CdH99]. Recently, Fernández et al. investigated the influence of fuzzy reasoning techniques in the context of imbalanced data [FGHd07].

2.5. Experimental Settings

The practical utility of a novel classification algorithm is a very important aspect. To be useful in real life situations, a classifier should deliver correct predictions and comply with certain qualitative demands. The classifiers proposed in this work will not be application-specific. They will be able to process any typical classification problem. Unfortunately, this generality makes designing a good classifier a difficult task since there are many different challenges to cope with, e.g. small problems, large problems, noisy data, missing attributes, skewed distributions, etc. In order to deal with these complications, we will evaluate the algorithms we propose very thoroughly. Therefore, we run the tests on up to 45 different real-world data sets. This setup will be very demanding, however, it will disclose the strengths and weaknesses of the algorithms.

2.5.1. Testing Environment

The experimental tests in this thesis were conducted using the WEKA 3.5.8 framework [WF05]. WEKA is a comprehensive machine learning suite that allows the experimental testing of classification algorithms. All proposed algorithms in this work were implemented under the WEKA framework to allow a fair comparison. The benchmark classifiers used within this thesis were all original WEKA implementations, with two exceptions: The fuzzy rule learners CHI and SLAVE, see below, were implementations from the KEEL suite [AFSG⁺09]. To ensure fairness, these classifiers were wrapped within the WEKA framework. Since some of the classifiers are non-deterministic, the order of the seed values for the random number generators was fixed.

2.5.2. Data Sets

The testbed for the experiments was created by selecting or engineering 45 real-world classification data sets. Most of the data sets, 24 in total, were taken from the UCI Repositories [AN07], thirteen data sets belong to the Statlib Repositories [MV07] and three come from an agricultural domain [Bul07, Bar07, Har07]. The remaining five data sets were generated from meteorological station data, published by the German Weather Service (DWD). Table 2.1 shows an overview of the 2- and m -class data sets and their main characteristics. The selection of the data sets was guided by the criterion that the number of numeric attributes should have been at least as large as the number of nominal attributes, since the fuzzy techniques presented in this work are not effective for nominal data.

2.5.3. Benchmark Classifiers

The classifiers which will be developed in the current work will be tested experimentally. To analyze potential benefits of the novel algorithms, a comparison with existing classification algorithms will be undertaken. Since a wealth of algorithms exist that might be considered for this purpose, we had to be selective. The selection of benchmark algorithms was, first of all, guided by the availability of the implementations. This criterion might seem trivial, but unfortunately many algorithms exist only on paper and, due to imprecise descriptions, it is impossible to implement them in a way that reproduces the original results.

2.5.3.1. RIPPER

The RIPPER algorithm is a conventional rule learner introduced in Section 2.3.3. For the experiments we used the WEKA implementation “JRIP”. The minimum number of covered examples per antecedent was set to 2 and for the number of folds and the number of optimizations in RIPPER we used values 3 and 2, respectively (which is the default setting in WEKA leading to RIPPER2).

2.5.3.2. CHI

The *CHI*-algorithm is a grid-based fuzzy classifier [CWY95, CYP96]. To learn a fuzzy rule-base, it requires a fixed fuzzy partition for every input

Table 2.1.: The data sets used in the experiments. The values “Num.”, “Nom.” and “Miss.” denote the numeric and nominal attributes and the ones involving missing values.

Data set	# Inst.	# Classes	# Num.	# Nom.	# Miss.	Origin
analcatdata-authorship	841	4	70	0	0	Statlib
analcatdata-bankruptcy	50	2	5	1	0	Statlib
analcatdata-cyyoung8092	97	2	7	3	0	Statlib
analcatdata-cyyoung9302	92	2	6	4	0	Statlib
analcatdata-esr	32	2	2	0	0	Statlib
analcatdata-halloffame	1340	3	15	2	1	Statlib
analcatdata-lawsuit	264	2	3	1	0	Statlib
analcatdata-votesurvey	48	4	3	1	0	Statlib
biomed	209	2	7	1	2	Statlib
cars	406	3	6	1	2	Statlib
collins	500	15	20	3	0	Statlib
ecoli	336	8	7	0	0	UCI
eucalyptus	736	5	14	5	9	agricult.
glass	214	6	9	0	0	UCI
haberman	306	2	2	1	0	UCI
heart-statlog	270	2	13	0	0	UCI
ionosphere	351	2	34	0	0	UCI
iris	150	3	4	0	0	UCI
liver-disorders	345	2	6	0	0	UCI
metStatCoordinates	4748	16	3	0	0	own
metStatRainfall	4748	16	12	0	0	own
metStatRST	336	12	3	0	0	own
metStatSunshine	422	14	12	0	0	own
metStatTemp	673	15	12	0	0	own
mfeat-factors	2000	10	216	0	0	UCI
mfeat-fourier	2000	10	76	0	0	UCI
mfeat-karhunen	2000	10	64	0	0	UCI
mfeat-morphological	2000	10	6	0	0	UCI
mfeat-zernike	2000	10	47	0	0	UCI
optdigits	5620	10	64	0	0	UCI
page-blocks	5473	5	10	0	0	UCI
pasture	36	3	21	1	0	agricult.
pendigits	10992	10	16	0	0	UCI
pima diabetes	768	2	8	0	0	UCI
prnn-synth	250	2	2	0	0	Statlib
schizo-	340	2	12	2	11	Statlib
segment	2310	7	19	0	0	UCI
sonar	208	2	60	0	0	UCI
squash-unstored	52	3	20	3	8	agricult.
synthetic control	600	6	60	1	0	UCI
vehicle	846	4	18	0	0	UCI
vowel	990	11	10	2	0	UCI
waveform	5000	3	40	0	0	UCI
wine	178	3	13	0	0	UCI
wisconsin-breast-cancer	699	2	9	0	1	UCI

dimension. These partitions form a grid on the data space. A single fuzzy rule consists of such a grid cell as antecedent part and a class as consequent part. The consequence is determined by finding the most frequent class in that grid cell according to the fuzzy memberships of the training data. A problem for this strategy concerns empty grid cells lacking evidence in favor of any class. Chi proposed that a classifier should abstain for query instances falling in such areas. In this work we will instead assign the most frequent class according to the training data.

For the experiments we used an implementation provided from the KEEL group [AFSG⁺09]. We used the following parameter setting: 3 fuzzy sets, product T-norm, maximum inference and weighting scheme number 2 in accordance with Ishibuchi and Yamamoto [IY05].

2.5.3.3. SLAVE

A genetic fuzzy rule-based classifier was proposed by González and Perez under the acronym *SLAVE* which is short for *Structural Learning Algorithm in Vague Environment* [GP99, GP01]. This algorithm learns rules in an iterative way by using a genetic algorithm. For the experiments we used an implementation provided from the KEEL group [AFSG⁺09]. We used the following parameter setting: 5 fuzzy sets, 500 iterations without change, mutation probability 0.01, use weights true, population size 100.

2.5.3.4. C4.5

The *C4.5* decision tree learner was introduced by Quinlan [Qui93]. The main characteristics, which make it very attractive as a benchmark classifier, are the good classification accuracy and the small amount of time that is needed to build it. The C4.5 algorithm splits the data into subsets recursively. For each split it considers all attributes and selects the one for splitting that maximizes the information gain. In order to avoid overfitting effects, C4.5 uses an effective pruning strategy without separating the data into growing and pruning data.

2.5.4. Performance Measures

The perhaps most important aspect of a classifier in the realm of machine learning is the quality of the prediction it gives. It is legitimate to demand

that such a classifier should always return the true class of an instance with unknown class label. Unfortunately, this is often not easy and sometimes impossible, e.g. due to noisy data or missing values. To measure how good the prediction of a classifier is on average, we will use two different measures for reasons that will be explained below: (A) We will measure the classical classification accuracy. (B) We will also measure the area under the ROC curve which has a sound theoretical justification.

2.5.4.1. Classification Performance

The first performance measure we choose is *classification rate*, which we synonymously denote as classification accuracy. It reflects the relative number of correctly classified instances. This measure has its place in the history of machine learning. It was used in countless works on different classification algorithms. In some papers, classification accuracy appears as classification error, denoting the relative number of incorrectly classified instances. Note that we will deal with classifiers that are allowed to abstain from a classification decision within this thesis. These special cases, in which the accuracy and the error are not reciprocal, will be mentioned explicitly.

Unfortunately, classification accuracy is a questionable quality measure when the number of classes is very large or when the class distribution is skewed. Moreover, classification accuracy is unable to take the confidences of the predictions into consideration. For these reasons, a novel measure entered the stage in recent times.

2.5.4.2. Ranking Performance

As an alternative measure, the area under the ROC curve (AUC) is often calculated [PF97, PFK98, LHZ03, PD03, HL05]. To this end, the ROC (*Receiver Operating Characteristic*) curve and the area beneath will be considered.

In binary classification, when the task of a binary classifier \mathcal{M} is to distinguish between a positive class λ_{+1} and a negative class λ_{-1} , four different cases for a prediction can be distinguished:

True Positive The true class and the prediction were positive

True Negative The true class and the prediction were negative

False Positive The true class was negative, the prediction was positive

False Negative The true class was positive, the prediction was negative

Let TP be the number of true positives, let TN be the number of true negatives, let FP be the number of false positives and let FN be the number of false negatives.

The true positive rate TPR (or *sensitivity*) measures the fraction of positive instances that were classified correctly:

$$TPR \stackrel{\text{df}}{=} \frac{TP}{TP + FN}$$

Similarly, the false positive rate FPR (or *specificity*) describes the fraction of negative instances that were classified falsely:

$$FPR \stackrel{\text{df}}{=} \frac{FP}{FP + TN}$$

Assume that \mathcal{M} returns a score or pseudo-probability for a query instance. The ROC curve measures the sensitivity and the specificity of the prediction by ordering the predictions according to these scores and plotting TPR versus FPR , see Figure 2.5 for an example. The larger the area beneath the ROC curve, the better the predictive scores will rank.

Formally, the AUC can be introduced as follows: Let p be a discrete probability distribution on $\mathbb{D} \times \{\lambda_{+1}, \lambda_{-1}\}$ with $p : 2^{\mathbb{D} \times \{\lambda_{+1}, \lambda_{-1}\}} \rightarrow [0, 1]$. And let $s(\mathbf{x}) \in [0, 1]$ be the score that was returned by classifier \mathcal{M} so that $s(\mathbf{x})$ reflects the (pseudo-) probability that \mathbf{x} belongs to class λ_{+1} . The AUC is defined as the probability that a randomly drawn positive example $(\mathbf{x}, \lambda_{+1})$ has a larger score $s(\cdot)$ than a randomly drawn negative $(\mathbf{x}, \lambda_{-1})$. However, in order to approximate p with data $D \subseteq \mathbb{D} \times \{\lambda_{+1}, \lambda_{-1}\}$ the non-parametric Wilcoxon-Mann-Whitney statistic can be used [Wil45, MW47, HM82]. In this case the empirical AUC is defined as

$$\begin{aligned} \text{AUC}(\mathcal{M}, D) \stackrel{\text{df}}{=} & \frac{\left| \left\{ ((\mathbf{x}_i, \lambda_{+1}), (\mathbf{x}_j, \lambda_{-1})) \mid s(\mathbf{x}_i) > s(\mathbf{x}_j) \right\} \right|}{\left| \left\{ ((\mathbf{x}_i, \lambda_{+1}), (\mathbf{x}_j, \lambda_{-1})) \right\} \right|} \\ & + \frac{1}{2} \cdot \frac{\left| \left\{ ((\mathbf{x}_i, \lambda_{+1}), (\mathbf{x}_j, \lambda_{-1})) \mid s(\mathbf{x}_i) = s(\mathbf{x}_j) \right\} \right|}{\left| \left\{ ((\mathbf{x}_i, \lambda_{+1}), (\mathbf{x}_j, \lambda_{-1})) \right\} \right|}, \end{aligned}$$

where $i \neq j$ and $(\mathbf{x}_i, \lambda_{+1}), (\mathbf{x}_j, \lambda_{-1}) \in D$. Ties are counted with 0.5.

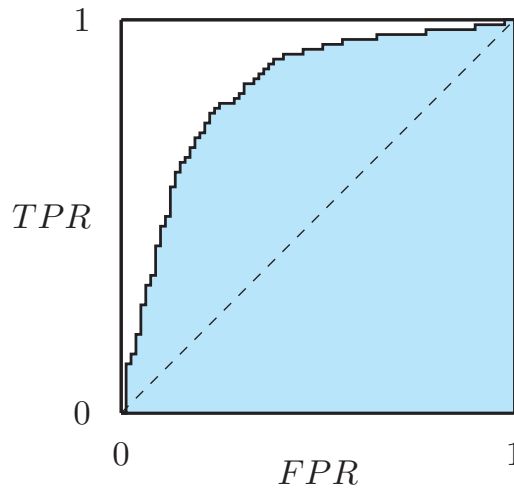


Figure 2.5.: A sample ROC curve (solid) and the ROC curve for random guessing (dashed). The size of the tinted area corresponds to the AUC.

To make the two-class AUC measure applicable for the multi-class case we report the weighted average AUC of the m 1-vs-All AUC values proposed by Provost and Domingos [PD03].

A characteristic of the ROC curve is that ties between instances deteriorate the AUC value [HV09]. It was found that the larger the spectrum of scores, the better the AUC tends to be [HV09, ZBHH08]. This variety of scores helps to dissolve ties in a sound way that works better than a random ordering of the predictions having the same score provided the classifier does better than random guessing.

According to this behavior, it is reasonable to assume that a fuzzy rule-based classifier has a fundamental advantage over its conventional counterpart. For the latter, the number of different scores is bound by the number of rules. For a fuzzy rule learner, the number of return values is unbound due to the infinite number of fuzzy membership values. We will investigate this assumption in our experiments.

2.5.5. Test Setup

In order to evaluate a classifier in terms of discriminative power, the data set is randomly split into two groups. The first group contains two thirds of the data set and is called *training data*. The remaining examples are called

testing data. The classifier will learn a model using the training data, while the testing data is not used. Afterwards, the classifier will use its learned model to predict the class memberships of the testing data instances one after the other. In this work, this test will be repeated 100 times to reduce the standard deviation and, thus, to stabilize the effects of the random data set splitting.

2.5.6. Statistical Evaluation

The experiments conducted in this work must be verified from a statistical point of view. The motivation for this is to assure that the observations in the experiments did not occur by chance and are, therefore, statistically significant. Again, it is important to note here that the data sets contain real-world information, for which no further assumptions concerning commensurability of the results (e.g. for classification accuracy) can be made. This has to be considered for the selection of the statistical tests. Thus, we will follow the recommendations of Demšar [Dem06]. These tests will not analyze classifier differences on single data sets, since a classifier should be able to achieve good results on (nearly) any data set given. Consequently, classifiers are to be compared according to the results achieved on a variety of data sets.

2.5.6.1. Comparison of Two Classifiers

In order to test two classifiers for significant differences, e.g. in terms of classification accuracy, a sign test will be conducted [Sal97, She07]. The idea is to count the number of wins, losses and ties. Let N be the number of data sets, then the null hypothesis states that each one gains approximately $N/2$ wins. Since the distribution of wins is — provided a sufficiently large N — normally distributed according to $N(N/2, \sqrt{N}/2)$, a significant difference in classifier performance is found if the z -test rejects the null-hypothesis. This is the case as soon as one classifier exceeds a critical number of wins. Ties are split equally.

2.5.6.2. Comparison of Multiple Classifiers

In order to test multiple classifiers for significant differences, e.g. in terms of classification accuracy, the Friedman Test will be conducted as a first step [Fri37, Fri40]. If the Friedman Test confirms significant differences among

the tested classifiers in general, the classifiers will be tested for significant differences in more detail using a post-hoc test.

The Friedman test is a non-parametric test which is based on the relative performance of classifiers in terms of their ranks: For each data set, the methods to be compared are sorted according to their performance, i.e., each method is assigned a rank (in case of ties, average ranks are assigned). Let k be the number of classifiers and N the number of data sets. Let r_i^j be the rank of classifier j on data set i and $R_j = \frac{1}{N} \sum_{i=1}^N r_i^j$ the average rank of classifier j . Under the null-hypothesis of equal classifier performance, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{j=1}^k (R_j)^2 - \frac{k \cdot (k+1)^2}{4} \right]$$

is asymptotically χ^2 distributed with $k - 1$ degrees of freedom. If N and k are not large enough, it is recommended to use the following correction which is F-distributed with $(k - 1)$ and $(k - 1)(N - 1)$ degrees of freedom [ID80]:

$$\frac{(N - 1) \cdot \chi_F^2}{N \cdot (k - 1) - \chi_F^2} \quad (2.5)$$

If that statistic for the given experiment is larger than the critical value, the null-hypothesis of equal classifier performance can be rejected. This means that there are significant differences among the classifiers, which have to be analyzed in a post-hoc test separately.

To test the outcome of the Friedman Test for specific differences statistically, Demšar proposed two test:

1. To analyze differences in between pairs of classifiers, the Nemenyi Test as a post-hoc will be conducted [Nem63]. According to this test, the performance of two classifiers is significantly different, if the distance of the average ranks has at least the critical distance

$$CD_{\alpha}^{\text{Nemenyi}} = q_{\alpha, k, \infty} \cdot \frac{1}{\sqrt{2}}, \quad (2.6)$$

where the q -value is from the Studentized Range Statistic [New39].

2. To compare competitors against a control classifier the Bonferroni-

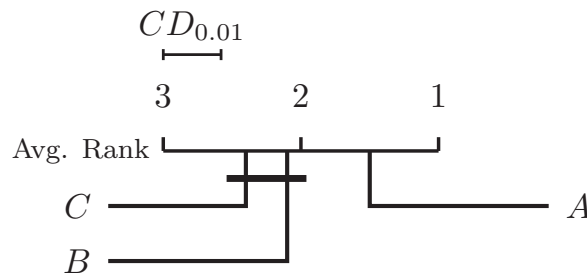


Figure 2.6.: Nemenyi Test visualization according to Demšar [Dem06]. Classifiers with no significant differences in terms of their ranks are connected.

Dunn Test is the post-hoc analysis to chose [Dun61]. According to this test, the performance of the control classifier and another classifier are significantly different if the distance of the average ranks exceeds the critical distance

$$CD_{\alpha}^{\text{Bonferroni-Dunn}} = q_{\frac{\alpha}{k-1}, k, \infty} \cdot \frac{1}{\sqrt{2}}, \quad (2.7)$$

where the q -value is from the Studentized Range Statistic [New39].

To visualize the statistical test results, Demšar proposed a number line from 1 to k . Every single classifier is marked on that number line at the position of its average rank R . As an example assume a test between three classifiers A, B and C . The best result with an average rank of roughly 1.5 was achieved by A , while B and C scored a rank between 2 and 2.5. Figure 2.6 shows the visualization for the Nemenyi Test results. Classifiers differ significantly, when the distance of the average ranks is larger than the critical distance. In the example this does not hold for B and C denoted by the horizontal bar connecting either. Classifier A is unconnected denoting that it performs better than both B and C .

The visualization of the Bonferroni-Dunn Test is very similar to the one before. The main difference is that the test is concentrated on the control classifier which is A in this example. Consequently, Figure 2.7 shows an interval around A that has twice the width of the critical distance. Another classifier is said to be non-significantly different if and only if it lies within this interval. As this is not the case in the example, the outcome is that A is significantly better than B and C . Note that the Bonferroni-Dunn Test does not analyze the difference between the latter.

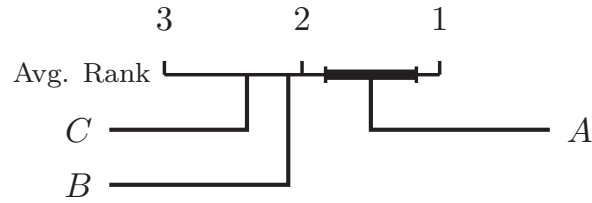


Figure 2.7.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. Classifiers lying within the interval around the control classifier *A* do not differ significantly from it.

2.6. Summary

In this chapter we introduced the theoretical and methodical foundations for this thesis. We outlined the problem setting of classification learning within the field of machine learning. We explained the fundamental ideas of separate-and-conquer rule learning and also of rules based on fuzzy logic. For the practical parts of this thesis we presented the experimental framework and the statistical tests we will use in detail.

3

FURIA: Fuzzy Unordered Rule Induction Algorithm

In this chapter we introduce the novel fuzzy rule-based classification method called *Fuzzy Unordered Rule Induction Algorithm*, or *FURIA* for short, which is a modification and extension of the state-of-the-art rule learner RIPPER [Coh95], cf. Section 2.3.3.

This chapter is structured as follows: In Section 3.1 we will introduce the idea behind FURIA. In Section 3.2 we will present the novel FURIA algorithm in depth. In the next part, Section 3.3, FURIA will be evaluated experimentally. Finally, we will conclude this chapter in Section 3.4 with a summary.

3.1. Introduction

The history of the learners FOIL, REP, IREP and finally RIPPER revealed that there was always room for some improvement — even when the current baseline was high. Regarding this development, one can definitely claim that RIPPER is a sophisticated algorithm which contains the findings from several years of research. Taking this into consideration makes the intention to build an even better algorithm based on RIPPER a challenging yet promising plan.

The FURIA algorithm is a successor to RIPPER that introduces novel techniques and modifications for improving the discriminative power of this algorithm without sacrificing too much of its effectiveness.

The main novelty of FURIA will be the use of flexible fuzzy or soft classification rules (for more information about fuzzy rule characteristics cf. Section 2.4). For this purpose we will introduce a novel rule fuzzification technique, which will soften RIPPER’s rule boundaries making it a fuzzy rule-based classifier.

As we mentioned before, RIPPER learns an ordered list of classification rules. This approach has advantages, but also some disadvantages. For example, it may come along with an unwanted bias since classes are no longer treated in a symmetric way. Moreover, sorting rules by priority compromises comprehensibility (the condition part of each rule implicitly contains the negated conditions of all previous rules). To avoid these problems, FURIA learns an unordered set of rules, namely a set of rules for each class in a 1-vs-All scheme. This, however, means that the resulting model is not necessarily complete, i.e., it may happen that a new query is not covered by any rule (in this regard, decision lists are obviously less problematic). To deal with such cases, we propose a novel rule stretching method which is based on Eineborg and Boström’s rule stretching method [EB01]. The idea is to generalize the existing rules until they cover the example. As an advantage over the use of a default rule, note that rule stretching is a local strategy that exploits information in the vicinity of the query.

3.2. Fuzzy Unordered Rule Induction Algorithm

This section introduces the novel algorithm. FURIA is mainly build upon the RIPPER rule learning procedure, cf. Section 2.3.3: While the separate-and-conquer rule induction remains mostly the same, other parts of the algorithm were modified or enhanced.

3.2.1. Learning Unordered Rulesets

A first modification of RIPPER concerns the type of rule model that is learned and, related to this, the use of default rules. As already mentioned in the introduction, learning a decision list and using one class as a default prediction has some disadvantages. In particular, it comes along with a systematic bias in favor of the default class. To avoid this problem, Eineborg and Boström proposed an unordered version of RIPPER’s predecessor IREP [Bos04, FW94]. Likewise, we propose learning a ruleset for every single class,

using a 1-vs-All decomposition. Consequently, FURIA learns to separate each class from all other classes, which means that no default rule is used and the order of the classes is irrelevant.²

When using an unordered ruleset without default rule, two problems can occur in connection with the classification of a new query instance: First, a conflict may occur if the instance is equally well covered by rules from different classes. As will be seen in Section 3.2.4, this problem is rather unlikely to occur and, in case it still does, can easily be resolved. Second, it may occur that the query is not covered by any rule. To solve this problem, we propose a novel rule stretching method. The idea, which will be discussed in more detail in Section 3.2.5, is to modify the rules in a local way that makes them applicable to the query.

A drawback of the unordered approach concerns its efficiency. The unordered rule learning of RIPPER is a 1-vs-Rest approach. In contrast to the 1-vs-All, this means that examples from the earlier learned classes are not considered as negatives during the subsequent learning process. The consequence is three-fold: (A) The number of examples shrink with every new rule learned throughout the training process. (B) The rules learned are simpler since there are less negative examples that these rules must be distinguished from. (C) The number of rules remains relatively small since the largest class is not described in terms of rules.

3.2.2. Pruning Modifications

The RIPPER algorithm can be divided into the building and the optimization phase. Rule building is done via the IREP* algorithm, which essentially consists of a propositional FOIL algorithm, the pruning strategy (cf. Section 2.3.3.1) and the stopping conditions. Interestingly, we found that the pruning strategies in IREP* have a negative influence on classification accuracy of FURIA. We therefore omitted the pruning step and instead learned the initial ruleset on the whole training data directly. To explain this finding, note that, without pruning, IREP* produces more specific rules that better fit the data. More importantly, small rules provide a better starting point for our fuzzification procedure, to be detailed in Section 3.2.3, in which rules can be made more general but not more specific.

²It is worth mentioning that, while Release 1 based on [Coh95] only supported ordered rule lists, an unordered approach is also included in a more recent RIPPER implementation of Cohen (Release 2.5).

In the optimization phase, the pruning was retained, as its deactivation was not beneficial. This is in agreement with the goal to minimize the MDL. The coverage of the remaining positive examples, which is again accomplished by IREP*, also benefited from omitting the pruning, in the same manner as IREP* in the building phase.

FURIA still applies pruning when it comes to creating the replacement and the revision rule. Here, the original pruning strategy is applied, except in case of the pruning strategy trying to remove all antecedents from a rule, thereby generating a default rule. In this case, pruning will be aborted and the unpruned rule will be used for the MDL comparison in the optimization phase. We found that these pruning strategies are still sufficient for avoiding overfitting. Thus, the removal of pruning in the IREP* part has no negative impact on classification accuracy.

3.2.3. Rule Fuzzification

To obtain fuzzy rules, the idea is to fuzzify the final rules from our modified RIPPER algorithm. More specifically, using the training set $D \subseteq \mathbb{D}$ for evaluating candidates, the idea is to search for the best fuzzy extension of each rule, where a fuzzy extension is understood as a rule of the same structure, but with intervals replaced by fuzzy intervals. Taking an interval I from an original RIPPER rule as the core $[b, c]$ of the sought fuzzy intervals $I^F = [a, b, c, d]$, the problem is to find optimal bounds for the respective supports, i.e., to determine a and d .

For the fuzzification of a single interval I_i on attribute A_i it is important to consider only the relevant training data D^i , i.e., to ignore those examples that are excluded by any other fuzzy intervals I_j^F , $j \neq i$:

$$D^i \stackrel{\text{df}}{=} \{(\mathbf{x}, \lambda) \in D \mid \mathbf{x} = (x_1, \dots, x_k) \wedge I_j^F(x_j) > 0 \text{ for all } j \neq i\} \quad (3.1)$$

We partition D^i into the subset of positive examples, D_+^i and negative examples, D_-^i . To measure the quality of a fuzzification, the purity will be used:

$$\text{pur} = \frac{p_i}{p_i + n_i} \quad , \quad (3.2)$$

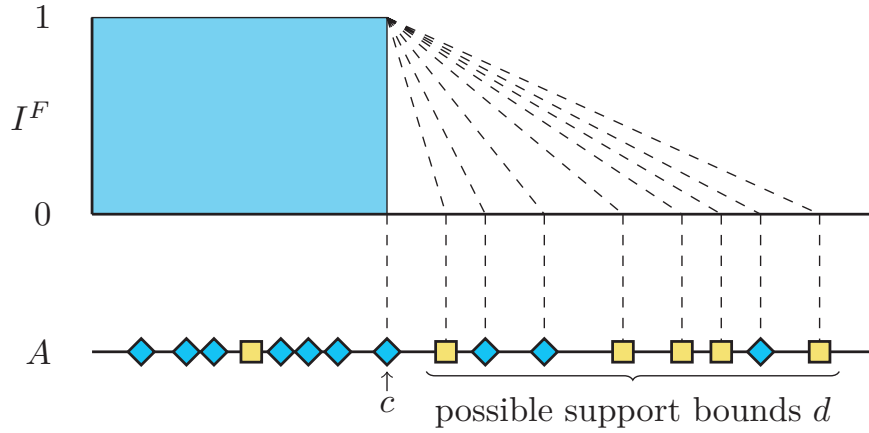


Figure 3.1.: Examination of possible support bounds d of a conventional interval $[-\infty, c]$.

where

$$p_i \stackrel{\text{df}}{=} \sum_{(\mathbf{x}, \lambda) \in D_+^i} I_i(\mathbf{x})$$

$$n_i \stackrel{\text{df}}{=} \sum_{(\mathbf{x}, \lambda) \in D_-^i} I_i(\mathbf{x}) \quad .$$

Rules are fuzzified in a greedy way, as shown by Algorithm 4. In each iteration, a fuzzification is computed for every antecedent, namely the best fuzzification in terms of (3.2). This is done by testing all values

$$\{x_i \mid (\mathbf{x}, \lambda) \in D^i, \mathbf{x} = (x_1, \dots, x_k), x_i < b_i\}$$

as candidates for a_i and, likewise, all values

$$\{x_i \mid (\mathbf{x}, \lambda) \in D^i, \mathbf{x} = (x_1, \dots, x_k), x_i > c_i\}$$

as candidates for d_i (see Figure 3.1). Ties are broken in favor of larger fuzzy sets, that is, larger distances from the core.

Fuzzification is then realized for the antecedent with the largest purity, cf. Figure 3.2. This is repeated until all antecedents have been fuzzified.

Note that the fuzzification of a single antecedent may change the relevant training data (3.1), which is hence recomputed in each iteration. In fact,

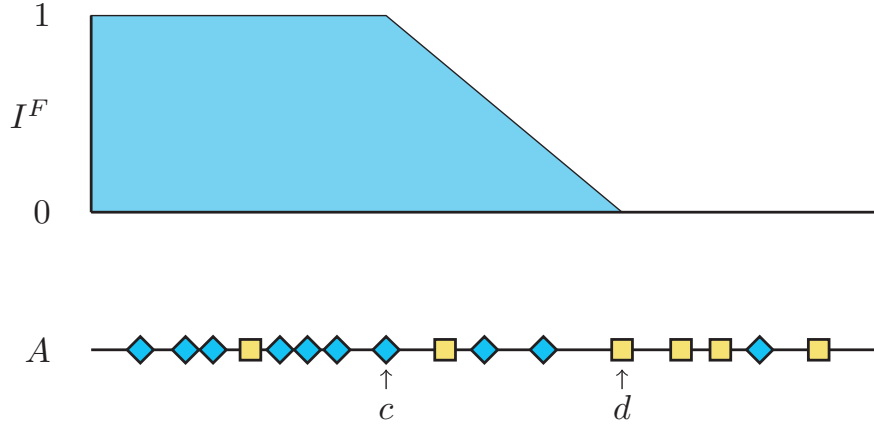


Figure 3.2.: The fuzzified interval $[-\infty, -\infty, c, d]$ maximizing the fuzzy purity.

each fuzzification may increase the number of covered examples, which in turn may also influence the rule purity. Furthermore, note that, after the complete premise part of a rule has been fuzzified, the whole procedure could in principle be repeated until convergence is achieved (convergence is guaranteed, as purity can only increase in each iteration). We did not implement this option, however, as we observed that, except for very rare cases, convergence is already achieved after the first iteration.

To analyze the complexity of the above fuzzification procedure, note that, in each iteration, $|D|$ examples (support bounds) — at the most — are checked for every candidate attribute. Since the total number of iterations is bound by the number of attributes, n , the overall complexity is $\mathcal{O}(|D|n^2)$.

3.2.3.1. Effectivity of the Rule Fuzzification Procedure

One might argue that there is no rule support found, claiming that the original conventional rule has the maximum purity. But if this is the case, then there exists a trivial fuzzification which is always found, namely the one that sets the support bound to the first example behind the core bound, see Figure 3.3. Even though this fuzzification does not change the purity on the training data, it is meaningful when it comes to classifying new instances. The reason is that it extrapolates the original rule very carefully to an area that might not have been covered by any rule at all.

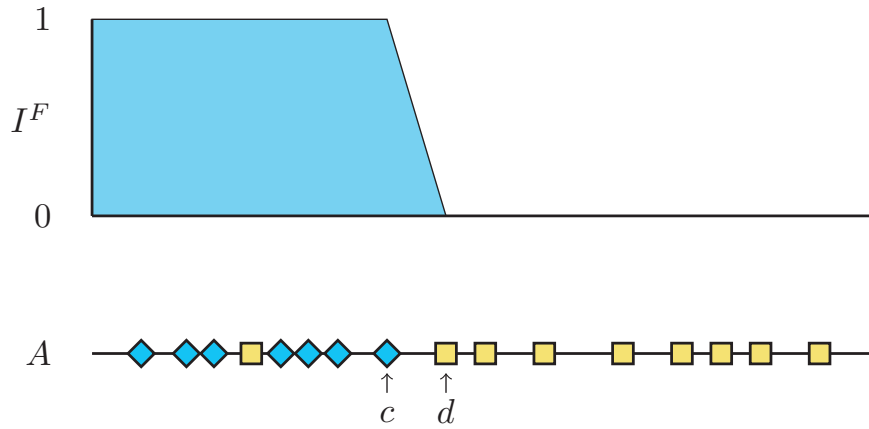
Following this line of argumentation, the next logical question to ask would be whether there is any non-trivial fuzzification for which the support thresh-

Algorithm 4 The antecedent fuzzification algorithm for a single rule r

```

1: Let  $A$  be the set of numeric antecedents of  $r$ 
2: while  $A \neq \emptyset$  do
3:    $a_{\max} \leftarrow \text{null}$  //  $a_{\max}$  denotes the antecedent with the highest purity
4:    $\text{pur}_{\max} \leftarrow 0$  //  $\text{pur}_{\max}$  is the highest purity value, so far
5:   for  $i \leftarrow 1$  to  $\text{size}(A)$  do
6:     compute the best fuzzification of  $A[i]$  in terms of purity
7:      $\text{pur}_{A[i]} \leftarrow$  be the purity of this best fuzzification
8:     if  $\text{pur}_{A[i]} > \text{pur}_{\max}$  then
9:        $\text{pur}_{\max} \leftarrow \text{pur}_{A[i]}$ 
10:       $a_{\max} \leftarrow A[i]$ 
11:    end if
12:  end for
13:   $A \leftarrow A \setminus a_{\max}$ 
14:  Update  $r$  with  $a_{\max}$ 
15: end while

```

**Figure 3.3.:** Trivially fuzzified interval $[-\infty, -\infty, c, d]$.

old is not set to the very first example beyond the original rule core. In fact, when a rule contains only one fuzzy interval as numerical antecedent, then there is no non-trivial fuzzification. The reason is that RIPPER learned this rule with the objective to maximize the information gain (2.3) that implicitly contains the purity (2.2). Consequently, the last example — marking the core boundary — covered by that interval must be from the rule's class. There

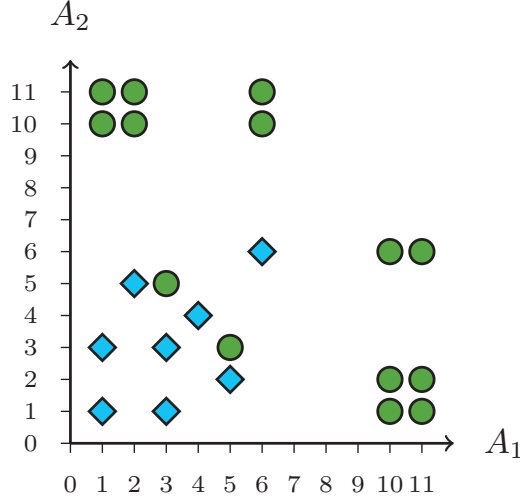


Figure 3.4.: A two-dimensional classification problem showing the myopic failure of separate-and-conquer learning and the correction through fuzzification.

can be no non-trivial fuzzification since the growing procedure would have selected this non-trivial support bound as core position.

A non-trivial fuzzification can be found when a rule uses at least two numeric attributes within its antecedent. In such cases it might happen that the myopic rule-learning strategy hinders the RIPPER rules from finding the global solution. Note that this is not a failure specific to RIPPER, since it might happen to any separate-and-conquer rule learner without look-ahead. Consider the following example:

Example 3.2.1 (Non-trivial fuzzification)

Assume examples from classes \blacksquare and \blacklozenge lying in $A_1 \times A_2 \subseteq \mathbb{R}^2$, cf. Figure 3.4. Moreover, let us consider a rule learner that tries to optimize the purity. The optimal conventional rule for the latter class is given by the rule

$$\text{IF } A_1 \in (-\infty, 6] \wedge A_2 \in (-\infty, 6] \text{ THEN } \blacklozenge$$

with a purity of $\frac{4}{5}$, cf. Figure 3.5.

The myopic separate-and-conquer strategy is not able to find this rule, because it adds one antecedent after the other. It first finds the rule

$$\text{IF } A_1 \in (-\infty, 5] \text{ THEN } \blacklozenge$$

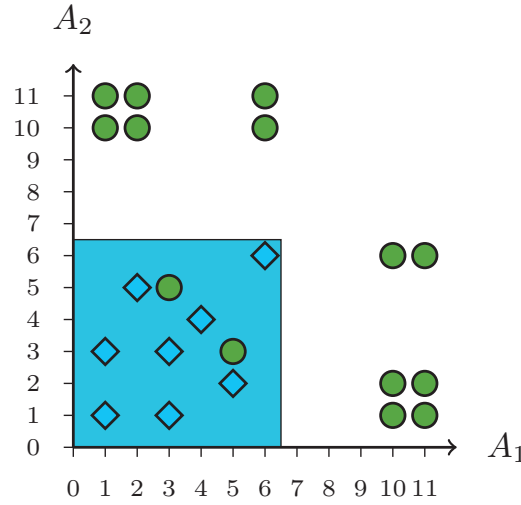


Figure 3.5.: Optimal conventional solution.

having a purity of $\frac{7}{13}$. The optimal first antecedent

$$\text{IF } A_1 \in (-\infty, 6] \text{ THEN } \blacklozenge$$

is disguised since it has a purity of only $\frac{1}{2}$ which is smaller than $\frac{7}{13}$.

The next step in the rule learning procedure cannot recover the mistake as it finds the rule

$$\text{IF } A_1 \in (-\infty, 5] \wedge A_2 \in (-\infty, 5] \text{ THEN } \blacklozenge$$

having a purity of $\frac{7}{9}$ which is smaller than $\frac{4}{5}$, cf. Figure 3.6. Note that due to the symmetry of the data set, the same values hold when considering A_2 first.

In this situation, a non-trivial fuzzification leads to the rule

$$\text{IF } A_1 \in (-\infty, -\infty, 5, 10] \wedge A_2 \in (-\infty, -\infty, 5, 10] \text{ THEN } \blacklozenge .$$

The example at (6,6) is covered to a degree of $\frac{10-6}{10-5} = 0.8$ by each condition such that the rule's purity is $\frac{39}{49}$ when using the minimum T-Norm, see Figure 3.7.

This example shows that the fuzzification, which is also a generalization

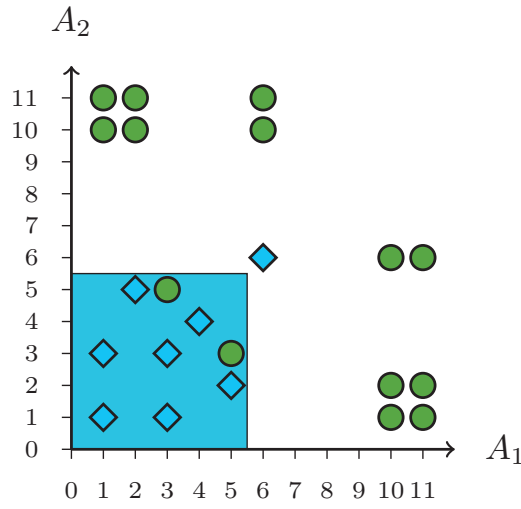


Figure 3.6.: FURIA solution without fuzzification.

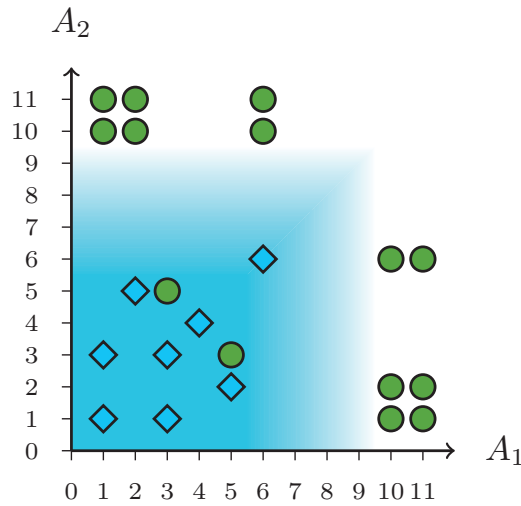


Figure 3.7.: FURIA solution with fuzzification.

procedure, of FURIA might help to recover mistakes that were made due to the short-sighted learning strategy.

3.2.3.2. Rule Readability and Linguistic Interpretability

With regard to the readability of the rules, we consider our fuzzy extension as uncritical. Essentially, the difference is that sharp boundaries of a rule are

replaced by “soft” boundaries: A fuzzy rule is uniquely characterized by its core and its support values. It is valid inside the core and invalid outside the support; in-between, validity drops in a gradual way. Consider, for example, the rule $\langle A \leq 5 \mid + \rangle$, which indicates that if attribute A is smaller or equal to 5, then the class is positive. Here, the rule is valid for $A \leq 5$ and invalid for $A > 5$. Similarly, a fuzzy rule $\langle A \in (-\infty, -\infty, 5, 8) \mid + \rangle$ suggests that the rule is completely valid for $A \leq 5$, invalid for $A > 8$ and partially valid in-between.

The models induced by FURIA are not linguistically interpretable in the classical sense of linguistic fuzzy rule learning as it was described by Ishibuchi et al. [INN05], see above Section 2.4.3. This is due to several reasons. First of all, FURIA learns rules which are not based on a common knowledge base containing fuzzy sets. Instead, the fuzzy sets learned by FURIA are rule-specific. Consequently, it is impossible to find a reasonable and small set of labels valid for all these fuzzy sets in general. Moreover, the fuzzy sets induced by FURIA do not comply with certain demanding characteristics, which are considered to improve interpretability, such as the completeness or Ruspini criteria for fuzzy partitions, cf. Definitions 7 and 8. For these reasons, models learned by FURIA cannot be considered being linguistic. For more information about linguistic fuzzy rules, interpretability and the problems with FURIA, see Chapter 4 and especially Section 4.1.

3.2.4. Classifier Output

The classification output of the RIPPER algorithm is determined in a simple way: The rules are examined in the order they were learned until a rule covering the query instance is found. This scheme leaves no room for interpretation since there is no interaction between the rules. FURIA abolishes rule ordering. Consequently, there might be none, one or multiple rules covering the query instance. To cope with such different situations a prod-sum rule voting procedure is used.

The degree to which a query instance \mathbf{x} is covered by a rule r^F having r_A^F in conjunctive normal form is given by (2.4). The rules learned by FURIA are plain conjunctions of conditions. As fuzzy T-norm we select the product operator since it enables non-linear decision boundaries. Thus, the rule

coverage of FURIA is given by

$$\mu_{r^F}(\mathbf{x}) \stackrel{\text{df}}{=} I_i^F(x_i) \cdot \dots \cdot I_j^F(x_j) \ .$$

Suppose that fuzzy rules $r_1^{F(j)} \dots r_k^{F(j)}$ have been learned for class λ_j . For a new query instance \mathbf{x} , the score of this class is defined by

$$s_j(\mathbf{x}) \stackrel{\text{df}}{=} \sum_{i=1, \dots, k} \mu_{r_i^{F(j)}}(\mathbf{x}) \cdot \text{CF}(r_i^{F(j)}) \ , \quad (3.3)$$

where $\text{CF}(r_i^{F(j)})$ is the *certainty factor* of the rule $r_i^{F(j)}$. It is defined as

$$\text{CF}(r_i^{F(j)}) = \frac{2 \frac{|D^{(j)}|}{|D|} + \sum_{(\mathbf{x}, \lambda_j) \in D^{(j)}} \mu_{r_i^{F(j)}}(\mathbf{x})}{2 + \sum_{(\mathbf{x}, \lambda) \in D} \mu_{r_i^{F(j)}}(\mathbf{x})} \ , \quad (3.4)$$

where $D^{(j)}$ denotes the subset of training examples with label λ_j . Ishibuchi argued that weighing the rules allows for modeling more flexible decision boundaries and thereby improves classification accuracy [IN01, IY05]. The rule weight or *certainty factor* (3.4) is the m -estimate for $m = 2$ [PFTV92].

The calculation of the scores $\{s_j \mid 1 \leq j \leq m\}$ according to (3.3) can be interpreted as a voting procedure in which every rule covering the instance has a vote.

The class predicted by FURIA is the one with maximal score s_j . In the case where \mathbf{x} is not covered by any rule, which means that $s_j(\mathbf{x}) = 0$ for all classes λ_j , a classification decision is derived in a separate way; see Section 3.2.5 below. In the case of a tie, a decision in favor of the class with highest frequency in the training data is made.

Example 3.2.2 (Flexible decision boundaries)

The interval fuzzification can lead to non-linear decision boundaries. Consider the three-class problem in Figure 3.8a. For the top-left and bottom-right class a one-dimensional rule is sufficient. The rule cores tightly fit the data, while the rule supports stretch to the lower left class. To separate the latter, a rule with conditions on both attributes is necessary. Again, the rule core covers the data in a firm way, while the support decreases to the edge of the other classes. As can be seen, all rule conditions are parallel to the axes. But the effective decision boundaries, which are showing where a transition

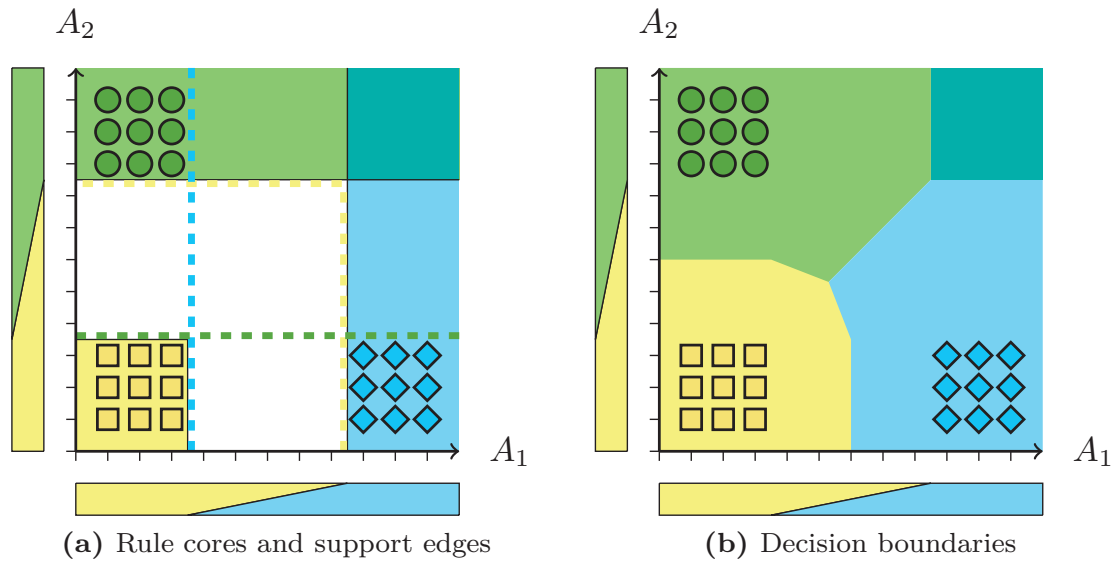


Figure 3.8.: A three-class problem generalized through three fuzzy rules with axis-parallel conditions and the respective, more complex decision boundaries.

from one class to the other occurs, look very different, see Figure 3.8b: The regions, in which instances are assigned to the same class, are neither rectangular nor fully parallel to the axes. Even more flexibility is possible through the selection of appropriate fuzzy operators, e.g. the product T-Norm, which allow non-linear decision boundaries. In comparison to conventional rules, this is a clear improvement in terms of flexibility.

3.2.5. Rule Stretching

To handle the aforementioned non-covering problem, Eineborg and Boström proposed replacing all rules by their *minimal generalizations* for the given instance [EB01]. A generalization or “stretching” of a rule is obtained by deleting one or more of its antecedents and a generalization is minimal if it does not delete more antecedents than necessary to cover the query instance. Thus, the minimal generalization of a rule is simply obtained by deleting all antecedents that are not satisfied by the query instance. Having derived all minimal generalizations, the authors re-evaluate each rule by its Laplace accuracy on the training data and then classify the query by the rule with the highest evaluation. Experimentally, it has been shown that this strategy,

which we subsequently refer to as EB-stretching, is better than using a default rule, i.e., simply predicting the most frequent class.

Unfortunately, EB-stretching has a high computational complexity, as it requires generalizing and re-evaluating every rule. Doing this on demand, for a fixed query, has a complexity of $\mathcal{O}(|RS| \cdot |D_T|)$, with $|RS|$ the number of rules and $|D_T|$ the size of the training set. Besides, it is worth mentioning that all training examples have to be stored. Alternatively, it is possible to pre-compute the evaluation of each possible generalization. However, since a rule r with an antecedent part r_A can be generalized in $2^{|r_A|}$ different ways, this goes hand in hand with large storage requirements.

To avoid these disadvantages, we propose an alternative approach that exploits the order in which the antecedents were learned, treating them as a list $\langle \alpha_1, \alpha_2 \dots \alpha_m \rangle$ instead of a set $\{\alpha_1, \alpha_2 \dots \alpha_m\}$. The idea is that the ordering reflects the importance of the antecedents, an assumption that is clearly justified in light of the underlying rule learning algorithm. As generalizations, we then only allow lists of the form $\langle \alpha_1, \alpha_2 \dots \alpha_k \rangle$ with $k \leq m$. For the minimal generalization, k is simply given by $j - 1$, where α_j is the first antecedent which is not satisfied by the query instance. To re-evaluate generalized rules, we use the measure

$$\frac{p+1}{p+n+2} \times \frac{k+1}{m+2} ,$$

where p is the number of positive and n the number of negative examples covered by the rule. The second factor accounts for the degree of generalization: Heavily pruned rules are discounted, as pruning is likely to decrease the rule's relevance for the query. Furthermore, by Laplace-correcting the relative number of remaining antecedents, k/m , preference is given to longer and, hence, more specific rules.³

Example 3.2.3 (Rule stretching procedures)

Consider an instance $\mathbf{x} = (5, 6, 7, 8) \in (A_1, A_2, A_3, A_4)$ and the following conventional rule:

$$r_{\text{orig}} = \text{IF } A_1 \in [0, 10] \wedge A_2 \in [0, 10] \wedge A_3 \in [0, 5] \wedge A_4 \in [0, 10] \text{ THEN } \lambda_j$$

³In order to present the following more concisely, we combined two selectors (half-intervals) referring to the same attribute into a single fuzzy interval, cf. Section 3.2.3. It is important to mention that, in the context of rule stretching, the two selectors are still treated as different antecedents.

In fact, \mathbf{x} is not covered by this rule since $7 \notin [0, 5]$. The minimum generalization according to Eineborg and Boström deletes only this condition:

$$r_{\text{EB}} = \text{IF } A_1 \in [0, 10] \wedge A_2 \in [0, 10] \wedge A_4 \in [0, 10] \text{ THEN } \lambda_j$$

The novel stretching procedure instead cuts the rule after the second condition:

$$r_{\text{Novel}} = \text{IF } A_1 \in [0, 10] \wedge A_2 \in [0, 10] \text{ THEN } \lambda_j$$

Computationally, the above outlined rule stretching strategy is much more efficient than EB-stretching. The complexity for re-evaluating a rule r is $\mathcal{O}(|r_A|)$. Moreover, since the evaluations of all generalizations of a rule can be calculated and stored directly in the course of the rule learning process, in which antecedents are learned in a successive way, there is no need for storing the training data.

3.3. Experiments

In order to analyze the performance of FURIA, we conducted several experimental studies. As a starting point, we used the RIPPER implementation of WEKA (“JRip”), cf. Section 2.3.3, for implementing the rule learning part of FURIA.

3.3.1. Classification Performance Analysis

In the first experiment, we compared FURIA to other classifiers with respect to classification accuracy. For RIPPER, we used the WEKA default settings according to Section 2.5.3.1. Moreover, we added two fuzzy rule-based classifiers from the KEEL suite [AFSG⁺09]: The fuzzy grid-based CHI algorithm and the genetic fuzzy rule learner SLAVE, cf. Sections 2.5.3.2 and 2.5.3.3. Finally, we also included the C4.5 decision tree learner as a benchmark classifier, cf. Section 2.5.3.4.

The testbed for this comparison consisted of the binary- and multi-class data sets, cf. Table 2.1. The experimental settings followed the setup proposed in Section 2.5.5. Table 3.1 summarizes the results in terms of mean classification accuracies⁴.

⁴The classifier FURIAc, which also appears in the table, will be analyzed in Section 3.3.3.

Table 3.1.: Average classification accuracies and ranks for FURIA and its competitors.

Data set	FURIA	RIPPER	C4.5	CHI	SLAVE	FURIAc
acd-authorship	95.67(1)	93.05(3)	93.50(2)	71.60(5)	91.87(4)	95.26
acd-bankruptcy	82.57(1)	81.97(2)	81.29(3)	74.40(5)	77.80(4)	83.83
acd-cyyoung8092	80.02(2)	80.04(1)	79.86(3)	70.72(5)	79.32(4)	80.17
acd-cyyoung9302	82.64(2)	82.01(3)	80.82(4)	80.27(5)	83.90(1)	82.96
acd-esr	80.90(2)	82.38(1)	80.36(3)	79.55(4)	77.72(5)	81.73
acd-halloffame	92.92(1)	92.87(3)	92.87(2)	92.18(5)	92.68(4)	92.89
acd-lawsuit	98.00(1)	97.54(3)	97.94(2)	94.93(4)	94.81(5)	97.96
acd-votesurvey	36.92(3)	34.40(4)	38.75(2)	40.19(1)	29.51(5)	36.35
biomed	88.31(1)	87.40(3)	87.80(2)	80.64(5)	84.74(4)	88.12
cars	79.08(2)	75.93(3)	82.15(1)	68.97(5)	70.68(4)	78.51
collins	96.35(1)	92.89(3)	96.10(2)	42.63(5)	50.87(4)	95.29
ecoli	83.12(1)	80.57(4)	81.35(2)	77.43(5)	81.03(3)	82.44
eucalyptus	60.62(1)	58.69(3)	59.98(2)	54.09(5)	58.16(4)	60.29
glass	68.22(1)	63.18(3)	66.69(2)	61.39(5)	61.83(4)	67.01
haberman	72.72(3)	72.16(4)	71.75(5)	73.08(2)	73.31(1)	72.80
heart-statlog	79.75(1)	78.44(2)	77.08(4)	68.66(5)	78.44(3)	79.56
ionosphere	89.59(2)	88.64(4)	88.72(3)	66.40(5)	89.83(1)	89.40
iris	94.76(2)	93.45(4)	94.25(3)	92.27(5)	94.92(1)	94.10
liver-disorders	67.15(1)	65.93(2)	63.40(3)	58.75(5)	59.77(4)	66.76
metStatCoord.	93.02(1)	92.04(3)	92.87(2)	46.79(5)	58.77(4)	92.83
metStatRainfall	64.51(1)	60.66(2)	59.47(3)	24.51(5)	29.35(4)	63.79
metStatRST	33.56(4)	36.08(3)	38.60(2)	25.24(5)	42.02(1)	33.31
metStatSunshine	49.05(1)	44.48(3)	46.78(2)	37.93(4)	28.83(5)	48.50
metStatTemp	50.71(2)	47.45(3)	53.18(1)	30.63(4)	22.10(5)	50.39
mfeat-factors	92.09(1)	87.05(4)	87.96(3)	89.19(2)	86.83(5)	91.76
mfeat-fourier	76.69(1)	71.37(4)	74.42(2)	69.27(5)	73.49(3)	76.07
mfeat-karhunen	86.47(1)	79.13(4)	80.20(3)	82.55(2)	78.37(5)	85.57
mfeat-morpholog.	72.09(1)	70.74(3)	71.60(2)	57.93(5)	67.08(4)	72.08
mfeat-zernike	73.67(1)	67.58(5)	69.11(3)	72.37(2)	68.26(4)	72.80
optdigits	94.78(1)	89.68(3)	89.51(4)	45.90(5)	93.45(2)	94.42
page-blocks	97.02(1)	96.79(3)	96.89(2)	91.96(5)	93.58(4)	96.91
pasture-prod.	74.67(1)	68.46(3)	73.67(2)	44.23(5)	53.63(4)	73.23
pendigits	97.77(1)	95.54(4)	95.92(3)	97.45(2)	87.26(5)	97.32
pima diabetes	74.71(1)	74.56(2)	73.43(4)	72.55(5)	73.65(3)	74.76
prnn-synth	83.57(2)	82.50(4)	83.18(3)	84.14(1)	81.51(5)	83.46
schizo-	80.52(1)	75.33(2)	74.93(3)	56.08(5)	56.29(4)	79.97
segment	96.50(1)	94.53(3)	95.95(2)	83.65(5)	88.87(4)	96.04
sonar	77.01(1)	72.41(3)	72.09(4)	74.61(2)	68.50(5)	76.34
squash-unstored	76.44(1)	71.74(3)	76.08(2)	70.56(4)	65.56(5)	77.10
synthetic control	89.75(2)	82.85(4)	90.00(1)	68.33(5)	89.23(3)	88.60
vehicle	70.10(2)	67.80(3)	71.38(1)	61.99(5)	64.08(4)	69.75
vowel	75.43(2)	64.71(3)	75.60(1)	59.49(5)	63.84(4)	71.87
waveform	82.24(1)	78.72(2)	75.05(4)	72.38(5)	75.34(3)	82.23
wine	93.25(1)	90.02(5)	91.22(4)	92.77(2)	92.46(3)	92.21
w.-breast-cancer	95.68(1)	95.58(2)	94.51(4)	90.20(5)	95.49(3)	95.53
average	1.40	3.07	2.60	4.24	3.69	

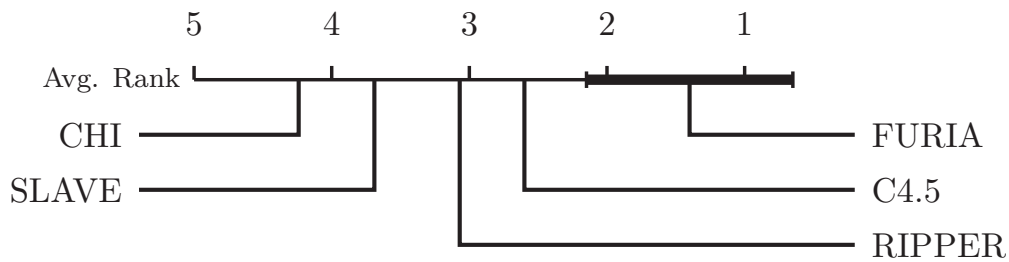


Figure 3.9.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. FURIA is significantly better than all competitors in terms of classification accuracy. Significance level $\alpha = 0.1$

The overall picture conveyed by the results is clearly in favor of FURIA, which outperforms the other methods on most data sets. To evaluate the performances in more detail, we conducted the Friedman Test, cf. Table 3.1. The corrected Friedman statistic for large N and k according to (2.5) is 39.77, while the critical value for the significance level $\alpha = 0.01$ is only 3.43. Thus, the null-hypothesis can quite safely be rejected, which means that there are significant differences in the classifiers' performance.

Given the result of the Friedman Test, we conducted the Bonferroni-Dunn Test as a post-hoc test to compare the competing classifiers to FURIA as the control classifier [Dun61]. The critical distance according to (2.7) between two classifier ranks is $CD_\alpha = 0.75$. The results of this test are summarized in Figure 3.9: FURIA is significantly better than all other classifiers at the significance level $\alpha = 0.1$.

3.3.2. Ranking Performance Analysis

The second analysis concerned with the discriminative power was conducted by measuring the area under the ROC curve (AUC). We used the same competitors, the same settings and the same choice of data sets as in the test before. Table 3.2 summarizes the results in terms of mean AUC⁴.

The analysis of the results suggests the very same result on the first peek as the previous one: FURIA outperforms the field. In fact, the Friedman Test again finds significant differences with a value of 7.97 for (2.5), while the critical value for the significance level $\alpha = 0.01$ is only 3.43. Thus, the null hypothesis is rejected.

The critical distance according to (2.7) between FURIA as control classifier

Table 3.2.: Average AUC and ranks for FURIA and its competitors.

Data set	FURIA	RIPPER	C4.5	CHI	SLAVE	FURIAc
acd-authorship	0.98(1)	0.96(3)	0.96(4)	0.90(5)	0.98(2)	0.98
acd-bankruptcy	0.87(1)	0.82(5)	0.83(4)	0.87(2)	0.86(3)	0.87
acd-cyyoung8092	0.74(2)	0.71(3)	0.67(4)	0.65(5)	0.77(1)	0.74
acd-cyyoung9302	0.77(2)	0.71(4)	0.70(5)	0.75(3)	0.79(1)	0.77
acd-esr	0.67(1)	0.62(3)	0.63(2)	0.60(4)	0.59(5)	0.67
acd-halloffame	0.83(3)	0.79(5)	0.81(4)	0.87(2)	0.89(1)	0.83
acd-lawsuit	0.95(2)	0.92(4)	0.93(3)	0.81(5)	0.96(1)	0.95
acd-votesurvey	0.52(3)	0.50(4)	0.54(2)	0.55(1)	0.49(5)	0.52
biomed	0.90(3)	0.86(5)	0.87(4)	0.93(1)	0.91(2)	0.90
cars	0.88(2)	0.82(4)	0.90(1)	0.86(3)	0.81(5)	0.87
collins	1.00(1)	0.97(3)	0.98(2)	0.88(4)	0.83(5)	1.00
ecoli	0.91(3)	0.89(4)	0.89(5)	0.94(1)	0.91(2)	0.91
eucalyptus	0.79(3)	0.82(2)	0.82(1)	0.79(4)	0.77(5)	0.79
glass	0.81(1)	0.76(5)	0.79(3)	0.77(4)	0.79(2)	0.81
haberman	0.62(2)	0.60(3)	0.56(5)	0.65(1)	0.60(4)	0.62
heart-statlog	0.82(1)	0.79(3)	0.78(4)	0.76(5)	0.81(2)	0.82
ionosphere	0.91(2)	0.88(4)	0.88(5)	0.89(3)	0.92(1)	0.91
iris	0.97(3)	0.96(5)	0.96(4)	1.00(1)	0.98(2)	0.97
liver-disorders	0.68(1)	0.64(2)	0.62(3)	0.57(5)	0.60(4)	0.68
metStatCoord.	0.98(2)	0.98(1)	0.98(3)	0.78(5)	0.82(4)	0.98
metStatRainfall	0.87(1)	0.85(2)	0.81(3)	0.76(4)	0.66(5)	0.87
metStatRST	0.65(3)	0.70(2)	0.71(1)	0.59(4)	0.58(5)	0.65
metStatSunshine	0.77(2)	0.75(4)	0.74(5)	0.80(1)	0.75(3)	0.77
metStatTemp	0.76(3)	0.77(2)	0.77(1)	0.66(5)	0.68(4)	0.76
mfeat-factors	0.98(2)	0.95(4)	0.94(5)	0.98(1)	0.97(3)	0.98
mfeat-fourier	0.92(2)	0.92(4)	0.88(5)	0.92(1)	0.92(3)	0.92
mfeat-karhunen	0.96(2)	0.92(4)	0.90(5)	0.98(1)	0.95(3)	0.96
mfeat-morpholog.	0.88(5)	0.94(1)	0.92(3)	0.93(2)	0.89(4)	0.88
mfeat-zernike	0.91(2)	0.90(4)	0.87(5)	0.95(1)	0.90(3)	0.91
optdigits	0.99(1)	0.96(3)	0.95(4)	0.82(5)	0.96(2)	0.99
page-blocks	0.95(1)	0.93(3)	0.93(2)	0.82(5)	0.83(4)	0.95
pasture-prod.	0.86(1)	0.79(3)	0.83(2)	0.64(5)	0.71(4)	0.86
pendigits	1.00(2)	0.98(4)	0.98(5)	1.00(1)	0.99(3)	1.00
pima diabetes	0.73(4)	0.71(5)	0.74(3)	0.80(1)	0.76(2)	0.73
prnn-synth	0.85(3)	0.84(5)	0.84(4)	0.90(1)	0.90(2)	0.85
schizo-	0.86(1)	0.78(3)	0.81(2)	0.56(5)	0.59(4)	0.86
segment	0.99(1)	0.98(2)	0.98(3)	0.97(4)	0.96(5)	0.99
sonar	0.81(2)	0.74(4)	0.72(5)	0.82(1)	0.75(3)	0.81
squash-unstored	0.83(1)	0.77(3)	0.81(2)	0.71(5)	0.76(4)	0.83
synthetic control	0.97(3)	0.93(5)	0.95(4)	0.99(1)	0.99(2)	0.97
vehicle	0.85(1)	0.85(2)	0.85(4)	0.85(3)	0.83(5)	0.85
vowel	0.93(1)	0.88(5)	0.91(3)	0.91(2)	0.89(4)	0.93
waveform	0.91(2)	0.88(4)	0.83(5)	0.91(3)	0.91(1)	0.91
wine	0.97(3)	0.93(5)	0.93(4)	0.98(1)	0.97(2)	0.97
w.-breast-cancer	0.97(3)	0.96(4)	0.95(5)	0.98(1)	0.97(2)	0.97
average	2.02	3.53	3.51	2.84	3.09	

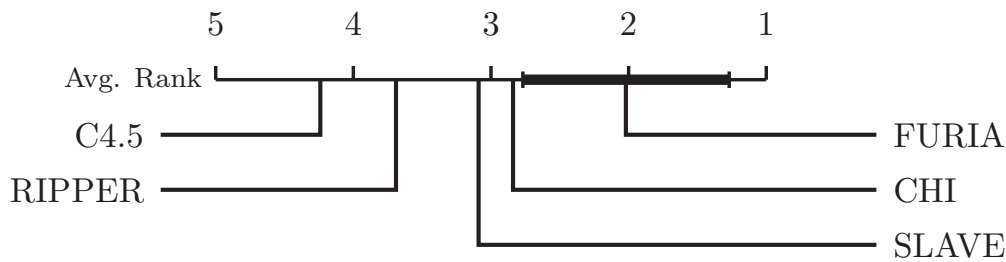


Figure 3.10.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. FURIA is significantly better than all competitors in terms of AUC. Significance level $\alpha = 0.1$

and any competitor is $CD_\alpha = 0.75$ for a significance level of $\alpha = 0.1$. The results of this test are summarized in Figure 3.10: FURIA is significantly better than all other classifiers at the significance level $\alpha = 0.1$.

Another interesting observation can be made in this context: The ranking of the competing classifiers changed completely from classification accuracy to AUC. We conjecture that the weak performance in terms of AUC of C4.5 and RIPPER can be explained through the limited spectrum of scores, cf. Section 2.5.4.2. The larger score variability of CHI and SLAVE improved their ranking position dramatically. From this point of view, we can confirm the findings of Hüllermeier and Vanderlooy as well as Zhang et al. [HV09, ZBHH08], see also Section 2.5.4.2.

3.3.3. Fuzzification Analysis

The previous results have demonstrated that FURIA is a significant improvement in comparison to RIPPER. Since FURIA differs from RIPPER in several ways, it is interesting to investigate the influence of the different modifications. One may wonder, for example, to what extent the improvements can be attributed to the use of fuzzy instead of conventional rules. To answer this question, we conduct some additional experiments with a conventional (*crisp*) variant of FURIA under the name FURIAc, cf. Table 3.1 and Table 3.2. To optimize an interval as originally produced by RIPPER, this variant conducts a search process quite similar to the search for an optimal fuzzy interval (cf. Section 3.2.3). Instead of a trapezoid, however, it is again only allowed to use intervals, i.e., it simply tries to optimize the original decision boundary in terms of the rule's purity.

Even though FURIAc still compares favorably to RIPPER (42 wins and 3 losses for accuracy and 40 wins and 5 losses for AUC) and C4.5 (34 wins and 11 losses for accuracy and 38 wins and 7 losses for AUC), the gains are less clear than those of FURIA. More importantly, in a direct comparison FURIA achieves 38 wins in terms of accuracy and 36 in terms of AUC. Besides, six of the seven data sets won by FURIAc in terms of accuracy are two-class data sets and the remaining one is a three-class data set, suggesting that fuzzy rules are especially useful for problems with many classes. This is confirmed from the AUC analysis, where 7 out of the 9 wins of FURIAc over FURIA are on data sets with four classes or less.

3.3.3.1. Effect of Fuzzification

Unfortunately, it is very hard to grasp the influence of fuzzification in a theoretically sound way. The reason is that fuzzification affects classification in several different ways, which all might influence the final outcome. To cope with this difficulty, we will conduct tests using modified versions of FURIA. The idea is to locate the reason for the predictive strength of FURIA by looking at the performance differences. We are hoping to explain the consequences of fuzzification by examining its individual effects with this sort of differential diagnosis. For this test we will focus on classification accuracy only.

Setup

To examine the impact of fuzzification (or more general: *post-generalization*), we measure the discriminative power of the overall rule model. To minimize the interference, we will disable the rule stretching technique for this test. Since FURIA is now allowed to abstain from classification for uncovered instances, we have to take this into consideration and measure both classification accuracy and classification error, i.e. the relative number of correctly and incorrectly classified instances. For this analysis, we will compare four different versions of FURIA that differ only slightly:

FURIA-prod The FURIA algorithm using the product T-norm as it was presented.

FURIA-min The FURIA algorithm using the minimum T-norm.

Table 3.3.: Wins and losses in terms of classification accuracy and error on the test data for variants of FURIA.

	Accuracy				Error			
	prod	min	crisp	w/o	prod	min	crisp	w/o
FURIA-prod	-	22	37	45	-	23	37	0
FURIA-min	8	-	37	45	7	-	37	0
FURIA-crisp	7	7	-	45	7	7	-	0
FURIA-w/o	0	0	0	-	45	45	45	-

Table 3.4.: Wins and losses in terms of classification accuracy and error on the training data for variants of FURIA.

	Accuracy				Error			
	prod	min	crisp	w/o	prod	min	crisp	w/o
FURIA-prod	-	1	6	43	-	1	6	28
FURIA-min	9	-	6	43	9	-	6	28
FURIA-crisp	26	26	-	42	26	26	-	29
FURIA-w/o	0	0	1	-	10	10	9	-

FURIA-crisp This version is very similar to the original FURIA version. It learns the very same model. The only difference is that during classification all interval memberships $I^F(x_i) > 0$ are rounded to 1. Note that this algorithm is different from the FURIAc variant discussed above.

FURIA-w/o This FURIA version does not apply any post-generalization technique. The rules remain in the same shape in which they were learned by RIPPER.

The complete results of this test can be found in Table A.1, Table A.2, Table A.3 and Table A.4. A short overview can be found in Table 3.3 and Table 3.4. To cope with the large amount of information, we analyze these tables step by step:

The effect of rule post-generalization on predictive performance

For this analysis we will have to compare FURIA-w/o to the rest. We know that the rules of FURIA-w/o are the original, unmodified RIPPER rules. Furthermore, recall that FURIA can only increase the coverage of the rules in the process of fuzzification. Consequently, FURIA will abstain on less

instances during classification than FURIA-w/o. In fact, when looking at classification accuracy and classification error on the test data, cf. Table 3.3, FURIA-w/o has the lowest classification accuracy and also the lowest classification error always. From this we cannot conclude whether FURIA-w/o is better or worse than the rest.

The effect of rule post-generalization on myopia

In Section 3.2.3.1 and Example 3.2.1 we argued that generalizing the rules after learning could correct an improper placement of the interval cores. We found so far that FURIA-w/o is a very careful variant that has the smallest classification accuracy and also the smallest classification error on the test data. Unfortunately, this information is not very telling and does not explain whether the other FURIA-variants that apply the post-generalization of the rules behave less myopic. For this examination we have to concentrate on the results on the training data. The reason is that if the generalization process had no beneficial impact, the accuracies and errors of FURIA-w/o and the rest should be the same. But surprisingly, we find that FURIA-w/o has a worse classification accuracy than the other FURIA variants on practically all data sets, cf. Table 3.4. On top of that, we see that FURIA-w/o has also a worse classification error as well. The variants have a lower classification error on at least 28 data sets and are tying on 7. In accordance with the null-hypothesis that the performances are equal, we can conduct the sign test, cf. Section 2.5.6.1. The sign test rejects this hypothesis with an error probability of $\alpha = 0.01$. From this we can conclude that the post-generalization has a beneficial impact on both accuracy and error on the training data. Especially the observation that the training error is smaller adds weight to the claim that the post-generalization mitigates the initial myopia.

The effect of the gradual rule coverage on classification accuracy

One of FURIA's key features is the fuzzification in the post-generalization procedure. The membership function of a fuzzy interval decreases linearly from its core suggesting that "nearer" instances belong to a larger degree to the interval than others farther away. While this notion makes sense intuitively, the consequences in terms of predictive power have to be examined. For this purpose we will compare the fuzzy versions FURIA-prod and FURIA-min to FURIA-crisp. Following this approach, we will analyze the

influence of the fuzziness alone. Table 3.3 displays the results on the test data: FURIA-prod and FURIA-min win 37 and lose only 7 data sets over FURIA-crisp in terms of both classification accuracy and classification error.

A surprising observation can be made in the results on the training data, cf. Table 3.4. FURIA-crisp wins 26 and ties 13 data sets over both fuzzy competitors FURIA-prod and FURIA-min in terms of classification accuracy and classification error. This is worth noting since this was not to be expected from the results on the test data. A very similar observation has been made by Kuwajima et al. [KNI08]. In a — to some extend — related analysis they also found that a stronger fuzzification leads to worse results on the training data, but not on the test data. For more information about the work of Kuwajima et al. see Section 7.2.8.

The results show that the gradual instead of the conventional extrapolation causes significant improvements. This can be explained by the fact that instances which are nearer to the core obtain a larger support from the interval than instances farther away. What happens in the conventional variant, where this is not the case, is that if two conventional rules from different classes overlap, the rule with the larger confidence prevails. Consequently, the conventional classification expands the influence of the stronger rule to the core of the weaker rule, overriding the weaker rule's support. An instance which is arbitrarily near to the weaker rule's core would always be classified through the stronger rule — a behavior that we criticized for conventional rules in general. Figure 3.11 depicts the membership degrees according to Example 3.2.2 (see page 54). We see that for the fuzzy variants, there are only smooth, continuous changes. But FURIA-crisp realizes hard decision boundaries that would treat instances in the pink, grey and ocher area in the same manner, respectively.

Comparing the effect of the product and the minimum T-norm on classification accuracy

As a conjunction operator to combine the coverage degrees of multiple fuzzy intervals, we proposed using the product T-norm. One of the characteristics of the product T-norm is that the decision boundaries have a smooth, round shape which extrapolate more carefully in terms of interpolation: Suppose an instance \mathbf{x} covered by I_1 and I_2 with $0 < I_1(x_1) \leq I_2(x_2) < 1$, then $\top_{\text{prod}}(I_1(x_1), I_2(x_2)) < \top_{\text{min}}(I_1(x_1), I_2(x_2))$. We assumed this to be benefi-

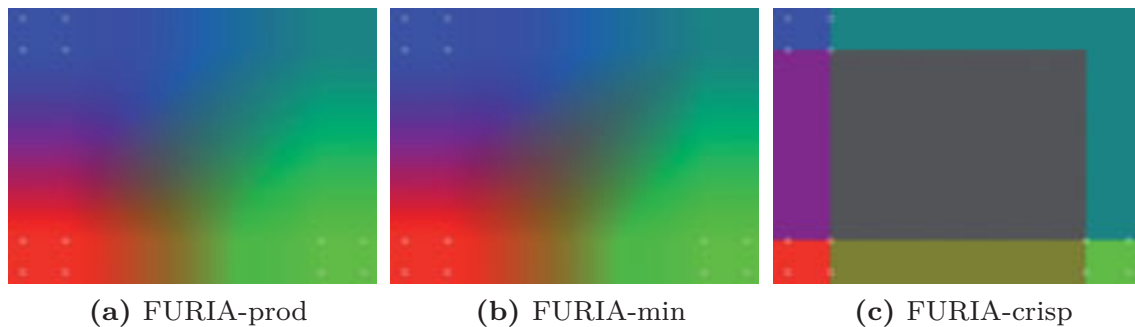


Figure 3.11.: Visualization of the decision boundaries. The different colors resemble the (fuzzy) coverage to the classes.

cial for classification accuracy in comparison to a T-norm with more rectangular decision boundaries such as the minimum T-Norm, see Figure 3.11.

Table 3.3 shows the test data results for the respective versions of FURIA, namely FURIA-prod and FURIA-min. We find that FURIA-prod wins over FURIA-min on 22 data sets and ties on 15 for classification accuracy. For classification error 23 wins and 15 ties convey a similar impression. Following the sign test, cf. Section 2.5.6.1, we cannot confirm the null-hypothesis of equal classifier performance according to an error probability of $\alpha = 0.05$: FURIA-prod is significantly better than FURIA-min. The results on the training data are different: FURIA-min is better than FURIA-prod on 9 data sets and worse on only one for both classification accuracy and error. Here, the sign test fails to reject the null-hypothesis of equal classifier performance with an error probability of $\alpha = 0.1$.

Synthesis

The previous analysis revealed very interesting results concerning the post-generalization procedure of FURIA: (A) Post-generalization is able to correct the myopic behavior of greedy rule learning. (B) Soft boundaries tend to improve the decision boundaries even when the underlying rule model is the same. (C) The product T-norm is significantly better than the minimum T-norm according to the results on the test data.

One might argue that FURIA-crisp had a structural disadvantage in comparison to FURIA-prod. The reason is that the conventional and fuzzy generalizations are different as we learned from the comparison of FURIA and

FURIAc. Thus, one could assume that using the “wrong” fuzzification could be the reason for the bad results of FURIA-crisp. But this is not the case: We ran the conventional optimization of the model as we did for FURIAc (FURIAc-crisp), but made the final classification decision in a fuzzy way (FURIAc-prod). We found FURIAc-prod outperforming FURIAc-crisp with 36 wins and 8 losses in terms of classification accuracy and classification error on the test set. Actually, the fuzzy set community might assign this latter insight even more importance: It is a proverbial away victory for the fuzzy variant.

3.3.4. Model Complexity Analysis

Since FURIA disables the pruning step in IREP*, it learns more specialized rules. Therefore, it is likely to produce models that are more complex, in terms of the number of rules and their lengths, than those produced by RIPPER. Indeed, while FURIA learns 25.4 rules on average, RIPPER generates only 15.5 rules.⁵ Moreover, while a FURIA rule has 2.5 conditions on average, a RIPPER rule has only 1.7; see Table 3.5 for detailed statistics. Consequently, the performance gain of FURIA in comparison with RIPPER comes at the cost of slightly more complex models.

Still, however, FURIA compares favorably with the other algorithms. Its average model size is quite comparable to the one of SLAVE, which creates 19.8 rules per model. Besides, the rules of FURIA are much shorter than the rules of SLAVE, which consist of 4.4 conditions on average. Since the CHI classifier uses a grid-based approach, every rule contains all attributes. In general, this leads to very large rulesets with long condition parts.

3.3.5. Rule Stretching Analysis

To investigate the effectiveness of our novel rule stretching method, we compared it to the original EB-stretching of (**author?**) [EB01]. More specifically, we compared the performance of FURIA with the performance of a variant of FURIA that uses EB-stretching instead of our rule stretching method, cf. Table A.5. The results, 19 wins for the variant, 26 losses and one tie, suggest that both methods are comparable in terms of classification accuracy. Furthermore, we can confirm that rule stretching works more effectively than

⁵Including RIPPER’s default rule.

Table 3.5.: FURIA model statistics. The number of rules per rule set and the average number of antecedents per rule.

Data set	FURIA		RIPPER		CHI		SLAVE	
	rules	cond.	rules	cond.	rules	cond.	rules	cond.
acd-authorship	15.9	2.7	9.6	1.7	555.1	3.0	12.1	5.9
acd-bankruptcy	3.8	1.8	2.5	0.7	23.9	3.0	2.5	1.6
acd-cyyoung8092	3.7	1.5	2.6	0.7	55.0	3.0	3.4	2.0
acd-cyyoung9302	3.5	1.3	2.8	0.8	49.5	3.0	3.1	2.0
acd-esr	2.1	1.1	2.0	0.5	6.8	3.0	2.6	1.1
acd-halloffame	14.3	2.8	6.5	1.8	458.9	3.0	7.2	3.4
acd-lawsuit	3.7	1.5	2.0	1.0	24.7	3.0	2.6	1.7
acd-votesurvey	1.7	1.4	2.3	0.8	13.6	3.0	7.3	2.1
biomed	8.6	2.0	4.4	1.0	55.1	3.0	4.3	2.7
cars	12.9	2.4	7.1	1.8	54.4	3.0	12.5	3.3
collins	15.9	1.1	15.2	1.0	321.7	3.0	45.8	6.6
ecoli	13.8	2.5	8.3	1.6	47.2	3.0	11.3	3.0
eucalyptus	14.7	2.6	10.2	1.8	375.0	3.0	38.3	5.7
glass	11.3	2.2	6.7	1.7	42.7	3.0	12.3	3.3
haberman	4.4	1.5	2.0	0.8	15.8	3.0	4.0	1.7
heart-statlog	8.4	2.5	4.3	1.5	164.9	3.0	7.0	3.6
ionosphere	8.3	2.0	4.7	1.1	168.9	3.0	8.0	3.8
iris	4.4	1.5	3.3	0.8	14.9	3.0	3.1	1.2
liver-disorders	8.2	2.2	4.3	1.8	42.1	3.0	5.9	3.4
metStatCoord.	69.7	2.3	38.8	2.1	15.6	3.0	12.8	2.4
metStatRainfall	123.9	4.5	82.7	3.6	215.6	3.0	30.3	4.4
metStatRST	9.9	2.2	10.0	1.7	15.0	3.0	9.5	2.3
metStatSunshine	25.0	2.7	17.0	1.9	91.0	3.0	39.3	4.3
metStatTemp	31.5	2.8	22.4	2.2	36.4	3.0	15.6	3.5
mfeat-factors	45.0	3.0	28.5	2.2	1317.2	3.0	44.3	12.2
mfeat-fourier	52.4	3.8	29.2	2.6	1317.2	3.0	73.2	10.6
mfeat-karhunen	59.1	3.2	38.4	2.6	1314.4	3.0	64.7	9.7
mfeat-morpholog.	25.1	2.6	19.0	2.1	31.4	3.0	15.7	3.1
mfeat-zernike	44.9	3.7	30.6	2.8	1257.6	3.0	77.7	10.0
optdigits	97.8	4.9	59.6	3.9	3708.5	3.0	68.6	8.0
page-blocks	25.6	3.2	14.7	2.2	47.6	3.0	10.1	3.5
pasture-prod.	3.4	1.4	3.2	0.7	24.0	3.0	3.6	3.3
pendigits	110.9	4.8	67.6	3.4	2745.2	3.0	37.0	7.6
pima diabetes	8.5	2.6	3.9	1.8	98.6	3.0	9.3	3.7
prnn-synth	4.4	1.4	3.5	1.0	8.0	3.0	2.4	1.6
schizo-	15.1	1.7	6.5	1.2	136.7	3.0	7.9	5.3
segment	26.9	3.1	17.0	2.2	275.1	3.0	15.8	4.5
sonar	8.1	2.3	4.3	1.4	137.1	3.0	6.9	4.7
squash-unstored	4.0	1.5	3.2	0.8	33.8	3.0	4.1	2.6
syntheticcontrol	17.3	2.6	10.6	1.8	394.3	3.0	9.1	6.3
vehicle	20.7	3.3	13.8	2.2	314.4	3.0	26.4	6.5
vowel	53.8	3.3	34.2	2.5	251.5	3.0	51.1	5.6
waveform	79.9	5.9	27.9	3.8	2874.7	3.0	50.7	9.3
wine	6.2	1.9	3.5	1.1	101.2	3.0	3.8	2.9
w.-breast-cancer	12.2	2.9	4.8	1.5	172.4	3.0	5.8	3.7
average	25.4	2.5	15.5	1.7	431.7	3.0	19.8	4.4

default classification (predicting the most frequent class): Both FURIA and EB-stretching achieve 42 wins against this strategy.

The rule stretching procedure applies only in cases in which the given instance is not covered by any rule. Since the number of uncovered instances depends on the data set (see Table 3.6), a theoretical comparison between the complexity of the two methods is difficult: Our approach conducts a fixed number of weight calculations, whereas EB-stretching recalculates the weights only on demand. Therefore, we compared the number of times a rule weight has to be calculated in EB-stretching with the total number of all antecedents, which corresponds to the number of calculations conducted by our rule stretching procedure. To avoid repetitious calculation of the same weights, we cached the weights in EB-stretching. Table 3.6 shows the results of this comparison and Figure 3.12 plots the number of calculations as a function of the number of antecedents. As can be seen from the corresponding regression curves, this dependency is super-linear for EB-stretching, while being linear for our approach. This is in perfect agreement with the theoretical considerations in Section 3.2.5. Thus, from a complexity point of view, our approach is especially advantageous for complex models.

As another advantage, recall that our approach is not required to store the training data. In this regard, it is interesting to note that, if the complete training data is kept in memory during classification, as done by Eineborg and Boström’s approach, uncovered examples could also be handled by a simple nearest neighbor (NN) classifier [AKA91]. We tested this idea and, interestingly, found that FURIA in combination with a simple 1-NN classifier outperforms FURIA with EB-stretching for 35 out of 45 data sets.

3.3.6. Runtime Analysis

It is clear that FURIA, as an extension of RIPPER, which encompasses more complex strategies such as fuzzification and rule stretching, will pay for its improved accuracy with an increase in runtime. To elaborate on this aspect, we compared the following algorithms:

- RIPPER
- RIPPER without IREP (RIP*)
- RIPPER learning an unordered ruleset again without IREP (RIP**)
- FURIA without post-generalization or fuzzification of rules (FUR*)

Table 3.6.: Absolute rule weight calculation frequencies. The first column shows the relative amount of uncovered data.

Data set	Uncov.	Ours	EB	Diff.
acd-authorship	0.05	42.70	49.40	6.70
acd-bankruptcy	0.02	6.80	0.80	-6.00
acd-cyyoung8092	0.08	5.50	1.80	-3.70
acd-cyyoung9302	0.07	4.70	1.10	-3.60
acd-esr	0.04	2.30	0.20	-2.10
acd-halloffame	0.03	39.80	48.50	8.70
acd-lawsuit	0.00	5.50	0.70	-4.80
acd-votesurvey	0.77	2.40	1.70	-0.70
biomed	0.05	17.60	9.80	-7.80
cars	0.11	31.20	34.50	3.30
collins	0.01	18.00	7.20	-10.80
ecoli	0.05	34.70	26.30	-8.40
eucalyptus	0.30	38.50	61.70	23.20
glass	0.15	24.60	22.40	-2.20
haberman	0.07	6.40	2.40	-4.00
heart-statlog	0.08	21.20	21.40	0.20
ionosphere	0.04	17.00	9.90	-7.10
iris	0.01	6.60	0.90	-5.70
liver-disorders	0.17	18.20	19.90	1.70
metStatCoordinates	0.02	163.20	120.30	-42.90
metStatRainfall	0.23	556.60	1737.60	1181.00
metStatRST	0.49	22.10	24.10	2.00
metStatSunshine	0.26	67.60	91.10	23.50
metStatTemp	0.33	89.00	121.40	32.40
mfeat-factors	0.06	134.40	227.60	93.20
mfeat-fourier	0.13	200.60	622.00	421.40
mfeat-karhunen	0.10	189.30	426.00	236.70
mfeat-morphological	0.08	65.70	72.60	6.90
mfeat-zernike	0.18	167.60	489.20	321.60
optdigits	0.04	482.10	1949.60	1467.50
page-blocks	0.01	82.00	87.90	5.90
pasture-production	0.23	4.60	1.40	-3.20
pendigits	0.01	528.00	1429.30	901.30
pima diabetes	0.13	22.00	30.60	8.60
prnn-synth	0.08	6.10	2.00	-4.10
schizo-	0.13	25.20	25.90	0.70
segment	0.02	83.70	91.10	7.40
sonar	0.13	18.80	20.00	1.20
squash-unstored	0.06	6.10	1.40	-4.70
synthetic control	0.08	45.50	49.70	4.20
vehicle	0.22	68.60	118.60	50.00
vowel	0.11	175.00	291.00	116.00
waveform	0.10	474.70	2645.80	2171.10
wine	0.04	11.50	4.30	-7.20
w.-breast-cancer	0.02	34.90	26.11	-8.79

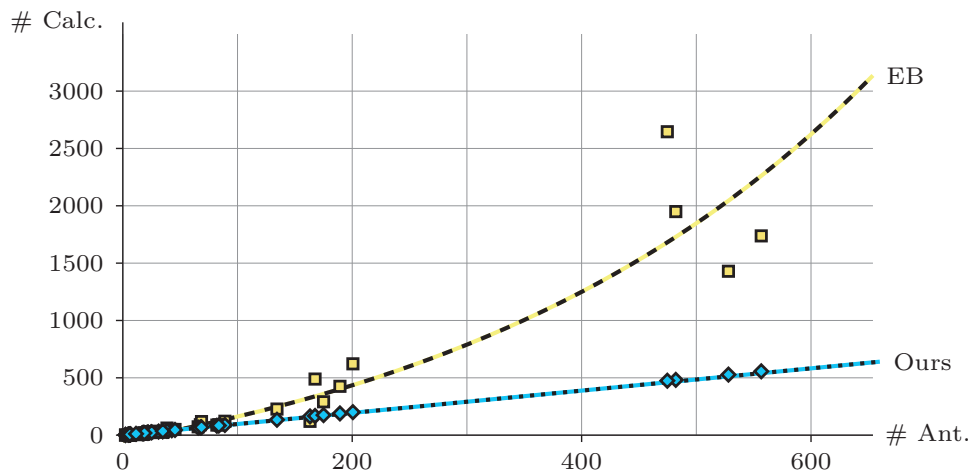


Figure 3.12.: The number of rule weight calculations as a function of the number of antecedents for our rule stretching method (◆) and EB-stretching (■). Trends are shown in terms of corresponding regression curves.

• FURIA

Table 3.7 shows the runtime results in seconds⁶. As expected, RIPPER is the most efficient variant. Disabling the IREP procedure (RIP*) does indeed slow down the algorithm (keep in mind that, since the pruning set is now empty, the growing data is larger at the beginning). A further increase in runtime is caused by switching from an ordered rule list to the unordered ruleset (RIP**). This is also expected since the unordered version learns rules in a 1-vs-All fashion, while for the ordered variant, the training data becomes successively smaller (training examples from already covered classes are dropped). There is not much difference between the unordered RIPPER without IREP (RIP**) and FURIA without fuzzification or conventional generalization after rule learning (FUR*). The difference between RIP** and FUR* can be explained by the rule stretching procedure that needs additional time to determine the weights during classification.

The quintessential outcome of this study is that, compared to RIPPER, the extensions and modifications of FURIA (disabling of the IREP procedure, the change from an ordered rule list to an unordered list, the calculation of the rule stretching weights and the fuzzification procedure) cause an increase of runtime by a factor between 1.5 and 7.7 (average 3.4).

⁶All measurements were performed on an Intel Core2Duo 2.4Ghz.

Table 3.7.: Average model building times in seconds for variants of FURIA. FUR* = FURIA w/o fuzzification process, R* = RIPPER unordered w/o IREP, R** = RIPPER w/o IREP.

Data set	RIPPER	RIP*	RIP**	FUR*	FURIA
acd-authorship	0.348	0.588	0.828	0.840	0.873
acd-bankruptcy	0.001	0.002	0.003	0.003	0.003
acd-cyyoung8092	0.004	0.005	0.012	0.010	0.010
acd-cyyoung9302	0.003	0.004	0.008	0.008	0.008
acd-esr	0.000	0.000	0.001	0.001	0.001
acd-halloffame	0.461	0.706	0.996	1.008	1.046
acd-lawsuit	0.003	0.003	0.006	0.005	0.007
acd-votesurvey	0.002	0.004	0.003	0.004	0.004
biomed	0.012	0.013	0.027	0.030	0.031
cars	0.045	0.094	0.125	0.138	0.146
collins	0.210	0.300	0.329	0.321	0.327
ecoli	0.032	0.066	0.084	0.089	0.095
eucalyptus	0.280	0.659	0.890	0.953	0.976
glass	0.034	0.054	0.079	0.084	0.090
haberman	0.010	0.016	0.034	0.045	0.046
heart-statlog	0.023	0.033	0.068	0.077	0.080
ionosphere	0.082	0.115	0.221	0.228	0.233
iris	0.002	0.004	0.007	0.006	0.006
liver-disorders	0.024	0.049	0.100	0.121	0.124
metStatCoordinates	1.327	4.199	4.834	7.481	10.266
metStatRainfall	11.287	24.868	29.186	35.720	42.819
metStatRST	0.065	0.134	0.164	0.170	0.181
metStatSunshine	0.321	0.566	0.669	0.658	0.684
metStatTemp	0.423	0.887	1.033	1.108	1.156
mfeat-factors	10.623	23.466	25.879	25.537	25.762
mfeat-fourier	9.733	30.836	33.859	32.685	33.236
mfeat-karhunen	7.973	19.593	21.658	20.720	21.164
mfeat-morphological	0.571	1.808	2.059	2.263	2.826
mfeat-zernike	6.563	17.002	18.605	17.943	18.383
optdigits	14.817	39.101	44.720	50.118	58.243
page-blocks	1.574	2.599	3.426	3.954	5.667 9
pasture-production	0.002	0.005	0.006	0.006	0.007 1
pendigits	10.716	30.488	35.114	47.218	77.755
pima diabetes	0.073	0.132	0.278	0.335	0.358
prnn-synth	0.007	0.010	0.024	0.027	0.030
schizo-	0.050	0.081	0.173	0.196	0.203
segment	0.940	2.178	2.666	2.818	3.074
sonar	0.082	0.103	0.224	0.230	0.233
squash-unstored	0.005	0.006	0.009	0.009	0.009
synthetic control	0.650	1.301	1.563	1.476	1.495
vehicle	0.282	0.787	0.900	0.949	0.988
vowel	0.715	1.755	1.949	2.013	2.164
waveform	7.778	29.359	42.688	50.521	57.809
wine	0.010	0.015	0.025	0.025	0.026
w.-breast-cancer	0.025	0.053	0.085	0.102	0.117

3.4. Summary

In this chapter we introduced a fuzzy rule-based classifier called FURIA, which is an advancement of the famous RIPPER algorithm. FURIA differs from RIPPER in several respects, notably in the use of fuzzy instead of conventional rules. Thus, it becomes possible to model decision boundaries in a more flexible way. Besides, FURIA makes use of a novel rule stretching technique which is computationally less complex than a hitherto existing alternative and improves performance in comparison to the use of a default rule. Combined with the sophisticated rule induction techniques employed by the original RIPPER algorithm, these improvements have produced a rule learner with a superb discriminative performance in both classification accuracy and AUC, which comes at the price of an acceptable increase in runtime. In fact, extensive experiments on a large number of benchmark data sets have shown that FURIA significantly outperforms the original RIPPER, as well as other fuzzy rule learning methods included for comparison purposes.

Apart from the practical innovations we also investigated the influence of fuzzification empirically. Therefore, we compared different FURIA variants and discovered that fuzzification had a beneficial impact on classification accuracy. Our analysis found two main reasons for the improvement: (A) Fuzzification, which is also a kind of post-generalization, mitigates the myopia of the greedy rule learning framework and (B) Fuzzification is a careful extrapolation because it assigns instances in the fuzzy support of the rule smaller values with increasing distance from the core.

4

HELLFIRE: Learning Linguistic Fuzzy Classification Rules

In this chapter we develop a novel fuzzy rule-based classifier that improves on FURIA in terms of interpretability. We found that the non-grid-oriented partitioning scheme of FURIA is problematic for the assignment of labels to the fuzzy set. We will discuss this problem and show an alternative approach that is able to create linguistic fuzzy rule-based classifiers.

We will give a short introduction into the motivation and the background of the new algorithm in Section 4.1. In Section 4.2 we will present the novel approach. After comparing it to other discretization approaches in Section 4.3, an experimental analysis will be done in Section 4.4. Before concluding this chapter, we will show an exemplary model and discuss the interpretability in Section 4.5. Finally, we will conclude this chapter in Section 4.6 with a summary.

4.1. Introduction

In the previous chapter we introduced the FURIA algorithm which is a fuzzy rule-based classifier. Fuzziness was obtained through a fuzzification of RIPPER rules. It was shown that this fuzzification had a beneficial im-

pact on the classifier’s discriminative power. But the facet of interpretable linguistic fuzzy rules has yet to become the focus so far. Due to the RIPPER-based fuzzy rules, it is difficult to assign the fuzzy sets reasonable linguistic labels. Some reasons for this difficulty are: (A) The fuzzy sets are rule- and not model-specific. So there could be multiple intervals of the form $[-\infty, -\infty, c, d]$ that could be represented through the label “low”. (B) The fuzzy sets might overlap a lot which also leads to the previous consequence. (C) The options of how to resolve those overlaps are manifold, especially when there are more than two rules that lie in different attribute subspaces. Without going into too much detail, we state that the conversion of a FURIA-based model into a grid-based linguistic rule-based model — without losing the discriminative abilities — is extraordinarily complex or perhaps even impossible.

A rule learning scheme which avoids the overlapping problem through a grid-orientation would help create a linguistic algorithm. Interval-based classification rules obtained from a decision tree would solve this, if there was no recursion creating independent subtrees making those intervals overlap. In this chapter we will present the *High-End Learning of Linguistic Fuzzy Interval Rule Expertise* — or *HELLFIRE* for short — algorithm that learns fuzzy rules based on a combination of rule induction and data discretization. The idea behind this algorithm is the coordination of independent subproblems from a decision tree. This rather simple idea leads to an algorithm that learns conventional classification rules and a data discretization parallelly. The transformation from conventional to fuzzy rules is done via a separate fuzzification step.

In contrast to FURIA, this approach will lead to a linguistic fuzzy rule-based classifier that is both interpretable and comprehensible.

4.2. High-End Learning of Linguistic Fuzzy Interval Rule Expertise

We can conceptualize learning the HELLFIRE classifier as follows.

4.2.1. Rule Learning

The rule learning of HELLFIRE is done via a coordinated splitting approach to ensure certain rule characteristics. In order to introduce this novel ap-

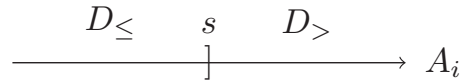


Figure 4.1.: The data is split along attribute A_i at position s into D_{\leq} and $D_{>}$.

proach, we first consider the divide-and-conquer strategy of decision tree learners. This will help introduce HELLFIRE since there are common characteristics of both.

In the typical decision tree learning algorithm [BF⁺84], the data D is separated by a split $A_i\theta s$ at position s of a numerical attribute A_i into two subsets $D_{\theta} \stackrel{\text{df}}{=} \{(\mathbf{x}, \lambda) \in D \mid x_i\theta s\}$ for $\theta \in \{\leq, >\}$ for a numerical attribute A_i or $\theta \in \{=, \neq\}$ for a nominal attribute A_i , cf. Figure 4.1. This is recursively repeated until the resulting subsets are pure enough to form the leaves of the tree.

In the usual classification tree generation method there is no coordination between different branches of the tree. This independence makes this approach able to locally adjust the decision boundaries to the data. It is well-known that a path from a leaf to the root in such a tree can be considered as a classification rule. The ruleset RS^S induced by an ordinary decision tree consists of rules $r^S \stackrel{\text{df}}{=} \langle r_A^S \mid r_C \rangle$, where the antecedent part

$$r_A^S \stackrel{\text{df}}{=} \{A_i\theta_{i,i_1}s_{i,i_1} \wedge \dots \wedge A_i\theta_{i,i_k}s_{i,i_k} \wedge \dots \wedge A_j\theta_{j,j_1}s_{j,j_1} \wedge \dots \wedge A_j\theta_{j,j_\ell}s_{j,j_\ell}\}$$

is a conjunction of the splits of attributes $\{A_i, \dots, A_j\}$ that occurred in the path from the leaf to the root and the consequent part r_C is the most frequent class of the data covered by r_A^S ; where $\{i_1, \dots, i_k\} \subseteq \{1, \dots, |x_i\theta s|\}$ are indices of occurring splits and $|x_i\theta s|$ is the number of splits for A_i .

Linguistic fuzzy rules are typically build upon fuzzy sets instead of conditions on the underlying attributes. A typical fuzzy ruleset RS^F contains fuzzy rules $r^F = \langle r_A^F \mid r_C \rangle$, where the antecedent part r_A^F is a fuzzy logic expression and the consequent part r_C is again a class assignment.

In order to learn linguistic fuzzy rules with a conventional decision tree, several adjustments have to be made. Initially, it has to be assumed that the fuzzy sets in the rules are made of simple, conventional intervals $I = [b, c]$. This constraint makes it possible to map the splits in the antecedent part r_A^S from a decision tree rule to a fuzzy rule antecedent r_A^F by a small detour over

an interval-based antecedent r_A^I :

$$\text{data set} \xrightarrow[\text{decision tree like learning}]{\rightsquigarrow} RS^S \xrightarrow[\text{transformation}]{\rightsquigarrow} RS^I \xrightarrow[\text{fuzzification}]{\rightsquigarrow} RS^F$$

Unfortunately, the rules derived from an ordinary decision tree are not useful for a direct transformation of that kind. The reason is that every rule was learned locally which might lead to an abundance of distinct intervals with potentially many overlaps. This makes assigning linguistic variables to those intervals awkward. Accordingly, the decision tree learning algorithm has been altered in such a way that the individual subtrees are not learned independently anymore. Instead, splitting is coordinated layer-wise. This assures that the intervals, which are induced by the splitting, are pairwise disjunct and cover the attribute domain completely. From another point of view, this approach is a data-driven discretization technique that decomposes all relevant numeric attributes into bins. Compare Figure 4.2 where a simple classification problem (4.2a) and the different data-driven solutions of decision tree learning (4.2b) and of the coordinated learning of HELLFIRE (4.2c) are shown. Due to its simplicity the decision tree solution is of course more elegant, but the discretization of HELLFIRE alleviates the labeling of the generated intervals.

Algorithm 5 and Algorithm 6 outline the learning procedure of HELLFIRE. The MAIN function contains the general rule learning procedure. The tree is learned in an implicit way by only keeping track of the data in the leaf layer and their respective rules in \mathcal{P} . A tuple $\langle D \mid r^S \rangle \in \mathcal{P}$ represents a leaf's single split rule r^S and the data that is covered by r^S .

4.2.1.1. Tree Splitting

In classical decision tree learning the data is split recursively into two subsets. Well-known criteria for that purpose are e.g. the *Gini Index* or *Shannon's entropy* which may be used to measure the imbalance in distributions [Gin21, Sha48]. Shannon's entropy is defined as

$$H(p_1, \dots, p_n) \stackrel{\text{df}}{=} - \sum_{i=1}^n p_i \cdot \log_2 p_i \ ,$$

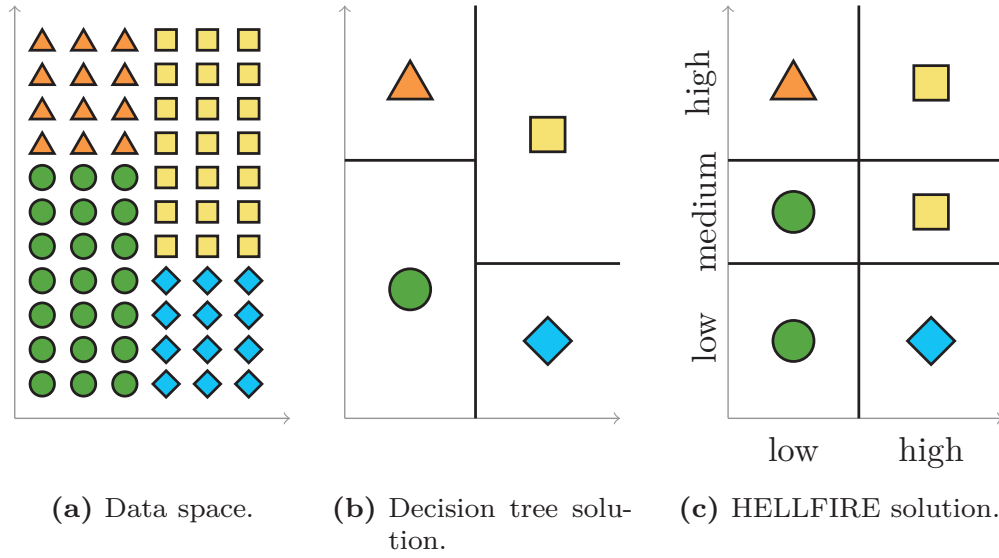


Figure 4.2.: A comparison between an ordinary decision tree and the data-driven discretization induced by HELLFIRE that can be easily labeled with linguistic variables.

where p_1, \dots, p_n with $\sum_{i=1}^n p_i = 1$ is a discrete probability distribution and $0 \cdot \log_2 0 \stackrel{\text{df}}{=} 0$. Shannon's entropy will reach its maximum value when the probabilities are all the same describing a situation of "maximum entropy" and will be zero for a distribution where exactly one outcome has a probability of 1. Entropy measures have been extensively used and studied in the field of decision tree research in the recent past [SJ06, OBG06, MZR08, MD08]. Generally, the procedure to split the data D in a node into two halves $D_{A_i \leq s}$ and $D_{A_i > s}$ involves an examination of all possible split positions s of all attributes A_i and selecting the split position and its respective attribute that minimizes the entropy:

$$\text{split}(D) \stackrel{\text{df}}{=} \underset{A_i, s}{\operatorname{argmin}} \frac{|D_{A_i \leq s}|}{|D|} \cdot H(D_{A_i \leq s}) + \frac{|D_{A_i > s}|}{|D|} \cdot H(D_{A_i > s}) ,$$

where $H(D_{A_i \theta s})$ with $\theta \in \{\leq, >\}$ denotes the entropy of the distribution of the elements $(\mathbf{x}, \lambda) \in D$ for which $x_i \theta s$ holds. The set of split positions $\{s_{i,1}, \dots, s_{i,n'-1}\}$ can be determined e.g. by taking the middle position between two consecutive points $s_{i,j} = (x_{i,j} + x_{i,j+1})/2$, where $x_{i,j} < x_{i,j+1}$ for

every $1 \leq j \leq n' - 1$ holds, assuming n' is the number of distinct values $|\{x_i \mid (\mathbf{x}, \lambda) \in D\}|$.

HELLFIRE coordinates the split across a whole layer in this tree-like structure all at once. This makes the standard entropy measures not directly applicable. Let $\mathcal{P} \stackrel{\text{df}}{=} \langle D^{L_1} \mid r^{L_1} \rangle \dots \langle D^{L_k} \mid r^{L_k} \rangle$ be the data sets in the nodes $1, \dots, k$ of the layer L with its respective rules containing the respective information of previous splits. To coordinate the splitting across L , the split position and the attributes have to be determined in a way respecting all D^{L_1}, \dots, D^{L_k} :

$$\begin{aligned} \text{coordinatedSplit}(D^{L_1}, \dots, D^{L_k}) &\stackrel{\text{df}}{=} \\ \operatorname{argmin}_{A_i, s} \sum_{\ell=1}^k \frac{|D_{A_i \leq s}^{L_\ell}|}{|D^{L_\ell}|} \cdot H(D_{A_i \leq s}^{L_\ell}) &+ \frac{|D_{A_i > s}^{L_\ell}|}{|D^{L_\ell}|} \cdot H(D_{A_i > s}^{L_\ell}) \end{aligned} \quad (4.1)$$

The selection of a split according to (4.1) looks for the best split on average. In general, this split selection is not the same as determining the individual splits for an attribute and averaging those positions afterwards.

Example 4.2.1 (Individual splits vs. coordinated split)

In this example we will show the benefits of a coordinated split in comparison to the individual splits. Figure 4.3 shows an example in which three data sets $D_{A_j=A}, D_{A_j=B}, D_{A_j=C}$ have to be split in a coordinated way. For starters, it is important to note that multiple optimal split positions in terms of Shannon's entropy H exists for each of these data sets, e.g. $\{s_5, \dots, s_{11}\}$ for $D_{A_j=A}$. When considering one of these subsets $D_{A_j=*}$ individually, it is reasonable to select the middle point that maximizes the margin.⁷ However, only one point is optimal in this example when splitting in a coordinated way: All three data sets can be perfectly separated when splitting at s_{11} . Consequently, a coordinated split is able to partition A_i in only two intervals $(-\infty, s_{11}]$ and (s_{11}, ∞) , while the three individual splits would lead to four intervals $(-\infty, s_8], (s_8, s_{10}], (s_{10}, s_{13}]$ and (s_{13}, ∞) .

When plotting entropy versus the split position, this “curve” will have potentially multiple plateaus of minimum values because of symmetric splits. Moreover, this “curve” will show two potential plateaus of maximum values on the lower and upper end of the numeric attribute. This is due to the

⁷For more information on margin maximization see Boser et. al. for example [BGV92].

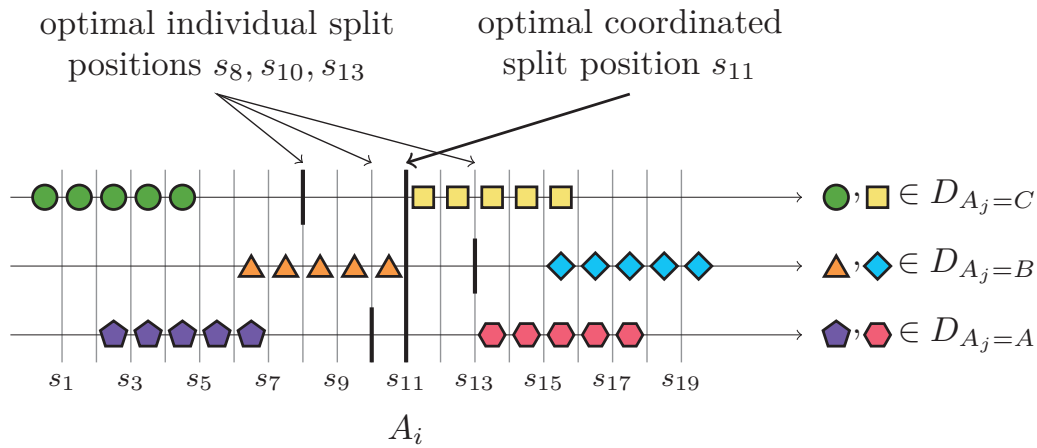


Figure 4.3.: Splitting three data sets $D_{A_j=A}$, $D_{A_j=B}$ and $D_{A_j=C}$ along A_i in a coordinated way. The coordinated split is different from the conventional splits for each independent data set.

fact that for all subsets D_θ the set of split positions is the same, namely the ones that were defined using the initial data D . The potential plateau of $H(D^{L_j})$ on the lower (upper) end of attribute A_i is a result of split points $\{s \mid s \leq \min(x_i)\}$ ($\{s \mid s > \max(x_i)\}$) for all $(\mathbf{x}, \lambda) \in D^{L_j}$ since in that case $H(D_{A_i > s}^{L_j}) = H(D^{L_j})$ ($H(D_{A_i \leq s}^{L_j}) = H(D^{L_j})$). Another consequence of this definition of split points is that for a subset D_θ there might be several split points leading to the very same split entropy, see Figure 4.4 for an example.

The averaging of split entropies can destroy the concavity characteristic of the respective averaged curve. Consequently, there might be more than one coordinated split point having an averaged minimal entropy, cf. Figure 4.4 for an example. In such a case, HELLFIRE selects a split having minimal averaged entropy randomly. In fact, it might even happen that the coordinated split of the averaged curve is not a minimum in any of the individual curves, but a “concession” split point that would not be selected by an ordinary decision tree algorithm.

Example 4.2.2 (Concession split)

This example demonstrates that the coordinated split is not necessarily an optimal split for the individual subproblems. Figure 4.5 shows an example in which three data sets $D_{A_j=A}$, $D_{A_j=B}$, $D_{A_j=C}$ have to be split in a coordinated way. The numbers below the examples from data set $D_{A_j=A}$ denote

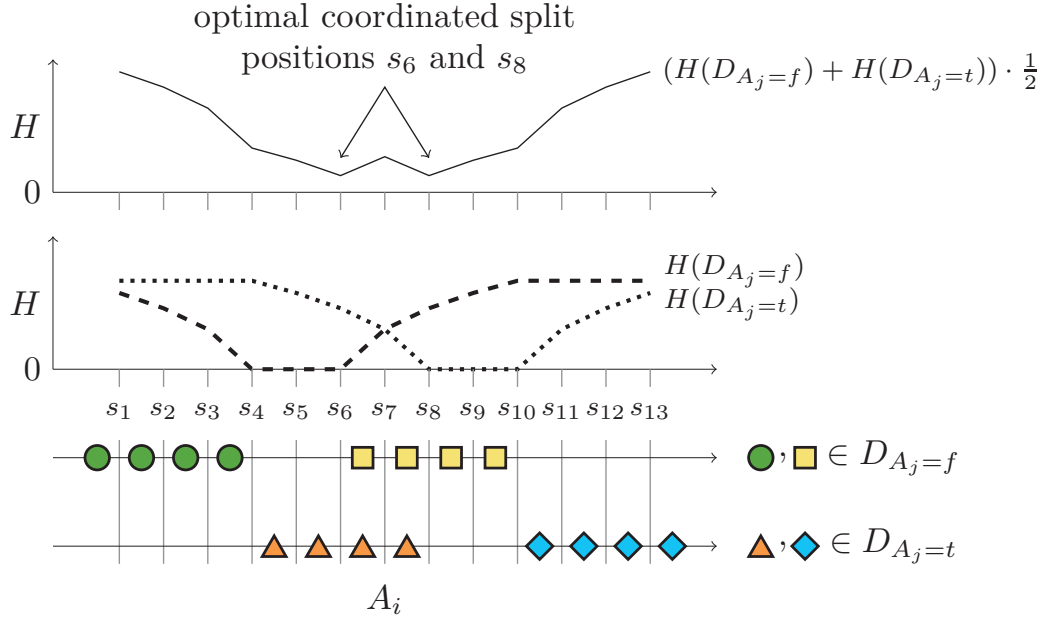


Figure 4.4.: Splitting two data sets $D_{A_j=t}$ and $D_{A_j=f}$ along A_i in a coordinated way. The entropy and averaged entropy are plotted versus the split values. (Note the interpolation for better visualization.)

multiple occurrences. The best individual split points can be found at s_5, s_{10} and s_{15} for the three data sets but the optimal coordinated splits are at s_7 and s_{13} .

Finding a situation in which such a coordinated split is concessive is quite difficult, but the shown data sets are a working example.

4.2.1.2. Avoiding Exponential Ruleset Growth

The learning approach of HELLFIRE creates a structure for which the splits are the same for every node in a given layer according to the coordinated split criterion that was introduced in Section 4.2.1.1. Without effective countermeasures this would lead to an exponential rule growth with the number of coordinated splits. We will avoid this scenario through pre-pruning mechanisms that abort the learning process early.

The first strategy `SPLITISALLOWED(split, D_θ)` examines whether the layer-coordinated split can be conducted for the node data D_θ , cf. Algorithm 6, Line 4. It tests two criteria: (A) hinders the procedure from branching into

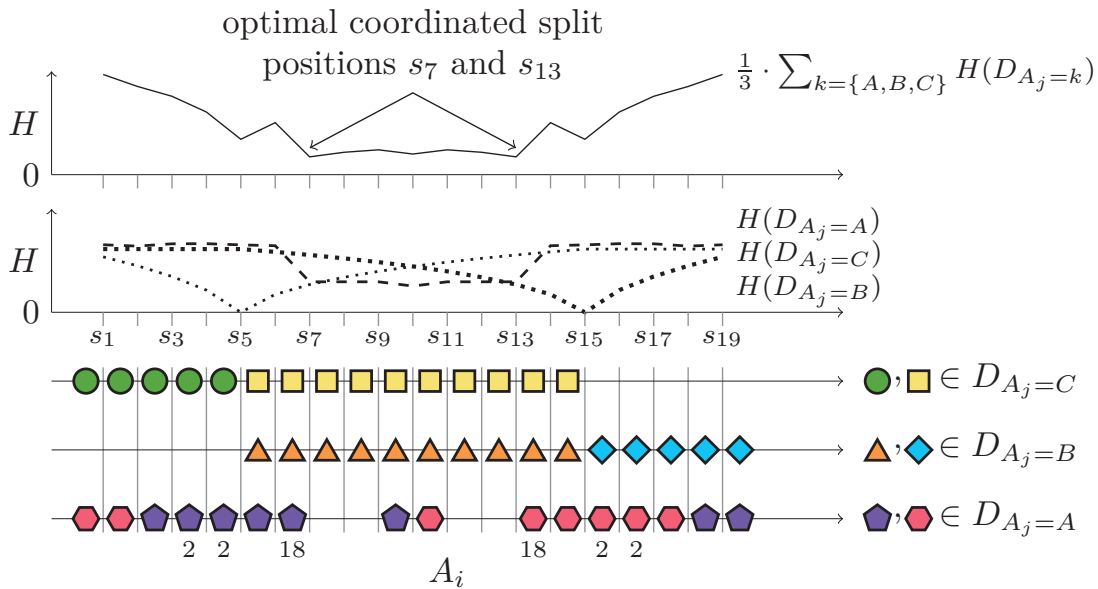


Figure 4.5.: Splitting three data sets $D_{A_j=A}$, $D_{A_j=B}$ and $D_{A_j=C}$ along A_i in a coordinated way. The entropy and averaged entropy are plotted versus the split values. (Note the interpolation for better visualization.)

D_{\leq} and $D_{>}$ if at least one would not be sufficiently large after the split. This would be the case if no class has more than one example in D_{θ} since no reasonable rule could be induced from that branch. (B) checks whether the separation would lead to a deterioration in the accuracy on the training data. Thus, instead of the two new nodes the unsplit data D of the original node will be passed on to the next lower level of the tree, assuring that it is considered for that lower level split, if it is not sorted out by the following mechanism.

The second strategy $\text{ISFINE}(r^S)$ is used to evaluate a rule r^S resembling a path from its leaf to the root whether it is already specific enough, cf. Algorithm 5, Line 6. This is the case if r overlaps not more than one example from classes other than its consequence class r_C or if r^S overlaps less than twice the number of attributes occurring in its antecedent r_A^S . If at least one of either condition is true then the respective branch will be removed from the learning process and r^S will be added to set the of rules RS^S .

The final simplification criterion forces the transformation of the remaining leaves into rules in the case that the coordinated split for layer L could not be applied to any of the data sets D^{L_1}, \dots, D^{L_k} , cf. Algorithm 5, Line 12.

Algorithm 5 MAIN(Data set D)

```

1:  $\mathcal{P} \leftarrow \{\langle D \mid \text{empty rule} \rangle\}$ 
2:  $RS^S \leftarrow \emptyset$ 
3:  $f \leftarrow \text{false}$ 
4: while  $\mathcal{P} \neq \emptyset$  do
5:   for all  $\langle D \mid r^S \rangle \in \mathcal{P}$  do
6:     if ISFINE( $r^S$ ) or  $f = \text{true}$  then
7:        $RS^S \leftarrow RS^S \cup r_i$ 
8:        $\mathcal{P} \leftarrow \mathcal{P} \setminus \langle D \mid r \rangle$ 
9:     end if
10:   $\mathcal{P}_{old} \leftarrow \mathcal{P}$ 
11:   $\mathcal{P} \leftarrow \text{SPLIT}(\mathcal{P})$ 
12:   $f \leftarrow \mathcal{P}_{old} = \mathcal{P}$ 
13: end for
14: end while
15:  $RS^S \leftarrow \text{PRUNE}(RS^S, D)$ 
16:  $RS^I \leftarrow \text{CONVERTSPLITSINTOINTERVALS}(RS^S)$ 
17:  $RS^I \leftarrow \text{CALCULATERULEWEIGHTS}(RS^I, D)$ 
18:  $RS^I \leftarrow \text{SETCOVERRULES}(RS^I, D)$ 
19:  $RS^I \leftarrow \text{JOINADJACENTINTSAPPEARINGALWAYSTOGETHER}(RS^I)$ 
20:  $RS^F \leftarrow \text{FUZZIFY}(RS^I, D)$ 
21:  $RS^F \leftarrow \text{CALCULATERULEWEIGHTS}(RS^F, D)$ 

```

The tree-like learning stops as soon as \mathcal{P} is empty. This means that all decision tree leaves were transformed into rules.

Example 4.2.3 (Coordinated splitting in action)

We explain the learning process of HELLFIRE on the simple artificial problem involving three labels shown in Figure 4.6. The initial data and the first split can be found in Figure 4.6a; here the split is an ordinary decision tree split because there is nothing to coordinate so that $\mathcal{P} = \{D_{A_1 \leq 6.5}, D_{A_1 > 6.5}\}$. This split is unable to represent any subset of \mathcal{P} as a classification rule in a reasonable way. Since $\mathcal{P} \neq \emptyset$, another split will be conducted. This time the split will be a coordinated one according to (4.1). It separates the data at $A_2 = 4.5$, cf. Figure 4.6b, leading to $\mathcal{P} = \{D_{A_1 \leq 6.5, A_2 \leq 4.5}, D_{A_1 \leq 6.5, A_2 > 4.5}, D_{A_1 > 6.5, A_2 \leq 4.5}, D_{A_1 > 6.5, A_2 > 4.5}\}$. The new split is still incapable of inducing a classification rule. Again, $\mathcal{P} \neq \emptyset$ holds and another coordinated split will

Algorithm 6 SPLIT(Partition \mathcal{P})

```

1:  $split \leftarrow \operatorname{argmin}_{\langle A_i | s \rangle} \sum_{\langle D | r \rangle \in \mathcal{P}} \text{SPLITENTROPY}(D, A_i, s)$ 
2:  $\mathcal{P}_{new} \leftarrow \emptyset$ 
3: for all  $\langle D | r \rangle \in \mathcal{P}$  do
4:   if SPLITISALLOWED( $split, D$ ) then
5:     for all  $\theta \in \{\leq, >\}$  do
6:        $D_\theta \leftarrow \{(\mathbf{x}, \lambda) \in D \mid x_i \theta s\}$ 
7:        $r_A^S \leftarrow r_A^S \cup \langle split \mid \theta \rangle$ 
8:        $r_C \leftarrow \text{MOSTFREQUENTCLASS}(D_\theta)$ 
9:        $\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup \langle D_\theta \mid r_\theta \rangle$ 
10:    end for
11:  else
12:     $\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup \langle D \mid r \rangle$ 
13:  end if
14: end for
15: return  $\mathcal{P}_{new}$ 

```

be conducted. Note that this second split is already different from what a decision tree would do: In this setting after the first split, a perfect second split for the subset $D_{A_1 > 6.5}$ would be at $A_2 = 2.5$ inducing a rule $\langle A_1 > 6.5 \wedge A_2 \leq 2.5 \mid \blacklozenge \rangle$. For coordinated splitting, the next step is shown in Figure 4.6c. This split is able to separate some homogeneous areas in terms of the class distribution in such a way that the respective subsets can be removed from \mathcal{P} and transformed into split-based rules:

$$\begin{aligned}
r_1^S &= \langle A_1 \leq 6.5 \wedge A_2 > 4.5 \wedge A_1 \leq 4.5 \mid \blacklozenge \rangle \\
r_2^S &= \langle A_1 \leq 6.5 \wedge A_2 > 4.5 \wedge A_1 > 4.5 \mid \bullet \rangle \\
r_3^S &= \langle A_1 \leq 6.5 \wedge A_2 \leq 4.5 \wedge A_1 > 4.5 \mid \bullet \rangle
\end{aligned}$$

This process continues with $\mathcal{P} = \{D_{A_1 \leq 6.5, A_2 \leq 4.5, A_1 \leq 4.5}, D_{A_1 > 6.5, A_2 \leq 4.5}, D_{A_1 > 6.5, A_2 > 4.5}\}$ ignoring the removed data for the subsequent splits, see Figures 4.6d–4.6f. The learning ends here after the induction of eleven rules $RS^S = \{r_1^S \dots r_{11}^S\}$ as \mathcal{P} runs empty.

Note that the only purpose of this example was to give a rough impres-

sion on the learning process using the coordinated splitting technique of HELLFIRE. We are well aware that for this data, a decision tree algorithm could induce a very similar set of rules using the very same split points.

4.2.2. From Split-Based Rules to Interval-Based Rules

The ruleset RS^S that is learned through the coordinated splitting approach must be converted to an interval-based set of rules RS^I , cf. CONVERTSPLITS-INTOINTERVALS(RS^S) in Algorithm 6, Line 16. The transformation process will remove obsolete splits from the rules $r^S \in RS^S$ which might occur when an attribute is split more than once.

The transformation from splits to intervals is straightforward: Given an attribute A_i , the coordinated splits $\{s_{i,\ell_1}, \dots, s_{i,\ell_k}\} \subseteq \{s_{i,1}, \dots, s_{i,k} \mid s_{i,j} < s_{i,j+1}\}$ are considered in ascending order such that the intervals I can be defined in the following way:

$$\begin{aligned} I_{\text{Low}} &\stackrel{\text{df}}{=} (-\infty, s_{i,\ell_1}] & j = 1 \\ I_{\text{Middle } j-1} &\stackrel{\text{df}}{=} (s_{i,\ell_j}, s_{i,\ell_{j+1}}] & 1 \leq j \leq k-1 \\ I_{\text{High}} &\stackrel{\text{df}}{=} (s_{i,\ell_k}] & j = k \end{aligned} \quad (4.2)$$

A characteristic of this interval definition is that it covers the underlying attribute A_i completely as soon as at least one split was made. Because of globally coordinated splitting in the learning process, all interval edges are directly data-driven and appear as split points in the rules of RS^S . To obtain the same coverage as a rule r^S using intervals, it might be necessary to disjunct more than one interval, however. The reason for this is that r^S might be removed from \mathcal{P} early and that its intervals were split further after that. The splitting of intervals occurring in r_A^S which are done after its removal have no effect on r^S .

The way the intervals are derived from the splits in (4.2) alleviates the mapping from the split rule r^S to a semantically identical interval rule r^I . The rule r^I is created by adding all intervals I which are covered by the rule r^S . To allow the use of several intervals from the attribute, another rule syntax has become necessary. In a rule r^S , all conditions — regardless of their attributes — were conjuncted and, thus, an instance \mathbf{x} would only be covered if every split condition $A_i \theta s_i$ from r_A^S covered \mathbf{x} . In the interval-based rule antecedent the intervals from the same attribute have to be disjuncted

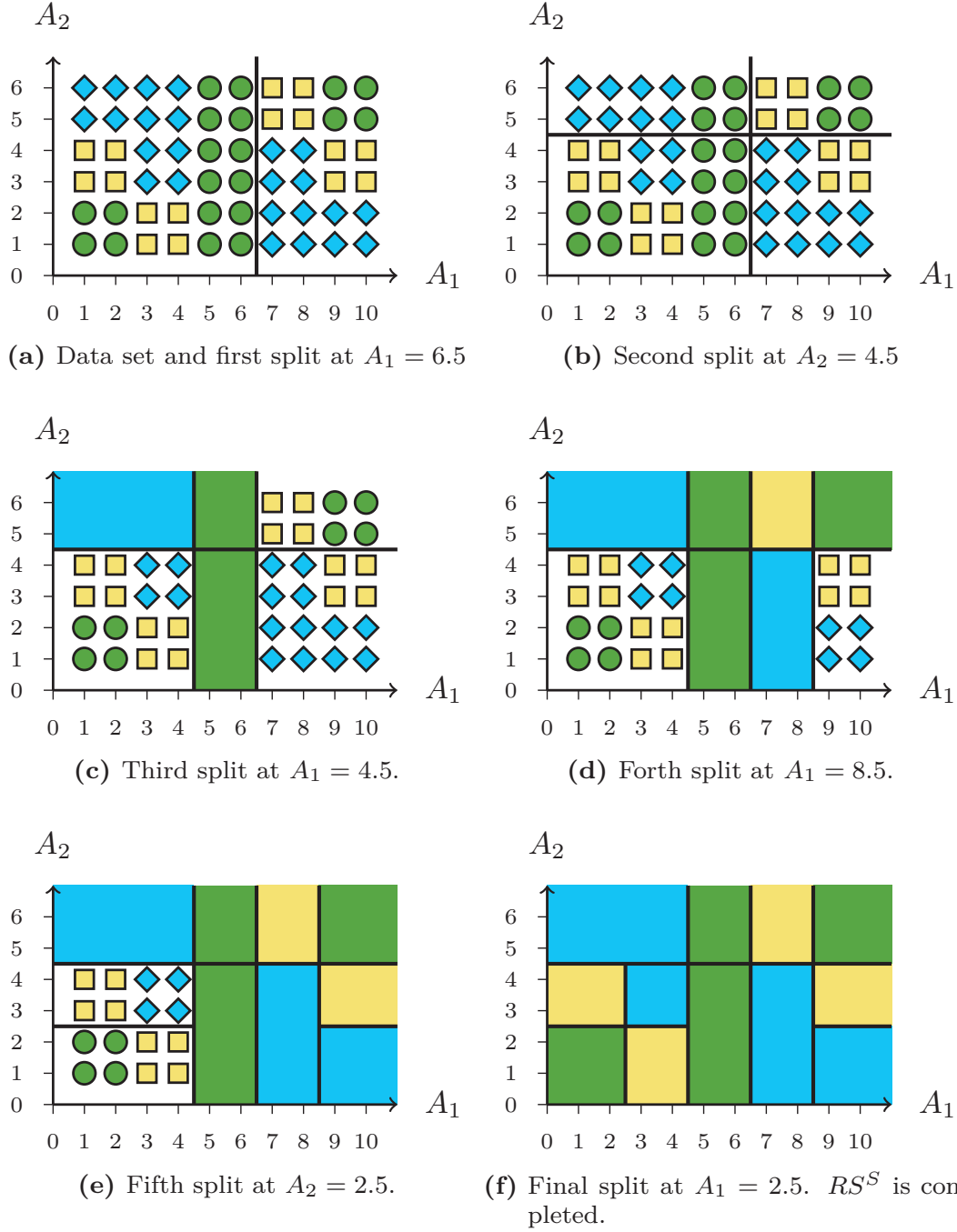


Figure 4.6.: The learning process using coordinated splitting of HELLFIRE for inducing the set of split-based rules RS^S .

to clauses and the latter have to be conjuncted:

$$r_A^I \stackrel{\text{df}}{=} (A_i \in I_{i,i_1} \vee \dots \vee A_i \in I_{i,i_k}) \wedge \dots \wedge (A_j \in I_{j,j_1} \vee \dots \vee A_j \in I_{j,j_\ell}) ,$$

where A_i is an attribute, $\{i_1, \dots, i_k\} \subseteq \{1, \dots, |I_i|\}$ are indices of occurring intervals and $|I_i|$ is the number of intervals for A_i .

It is necessary to allow disjunctions in the antecedent since multiple disjunct intervals from the same attribute would otherwise be mutually exclusive. An interval rule r^I covers an instance \mathbf{x} if and only if the evaluation of r_A^I on \mathbf{x} satisfies all conditions.

Example 4.2.4 (Transforming split-based into interval-based rules)

This brief example illustrates the transformation from the split-based ruleset RS^S from the previous example into an interval-based ruleset RS^I , see Figure 4.7. Initially, the split points $\{2.5, 4.5, 6.5, 8.5\} \subset A_1$ and $\{2.5, 4.5\} \subset A_2$ of the rules in Figure 4.7a are mapped to intervals, e.g. $I_{\text{Low}} = (-\infty, 2.5]$, $I_{\text{Middle } 1} = (2.5, 4.5]$, \dots for A_1 , cf. Figure 4.7b and Figure 4.7c. Now, every split-based rule r^S induces an interval-based rule r^I with exactly the same coverage. To illustrate this more effectively we dotted the lines between the intervals which appear in rules together, cf. Figure 4.7d. As the example shows, a complete grid-partitioning including non-atomic classification rules could be induced.

From the discretization point of view, it is important to take the flexibility of this approach into consideration. The data-driven splitting leads to data-driven intervals. This characteristic is important for the predictive quality of the rules [ER99]. It is interesting to see that many fuzzy rule learners are using non-data-driven fuzzy partitions for inducing classification rules, cf. Section 7.2.1. The problem with learning rules on a predefined partition are that (A) it is difficult to define the correct number of bins, (B) sometimes there has no data been observed within a bin, (C) that the split points are inept to reflect the true data distribution. It is clear that these problems become even worse the more number of dimensions are considered. Of course, using a fuzzy partition might mitigate these lacks to a certain degree. Yet, for an illustrative example, even one dimension is enough.

Example 4.2.5 (Data-driven vs. pre-defined discretization)

A quite frequently seen discretization in various fields of research, e.g. medicine, is the partition of “age” into decades $[0, 9)$, $[10, 19)$, \dots . Typically, such a

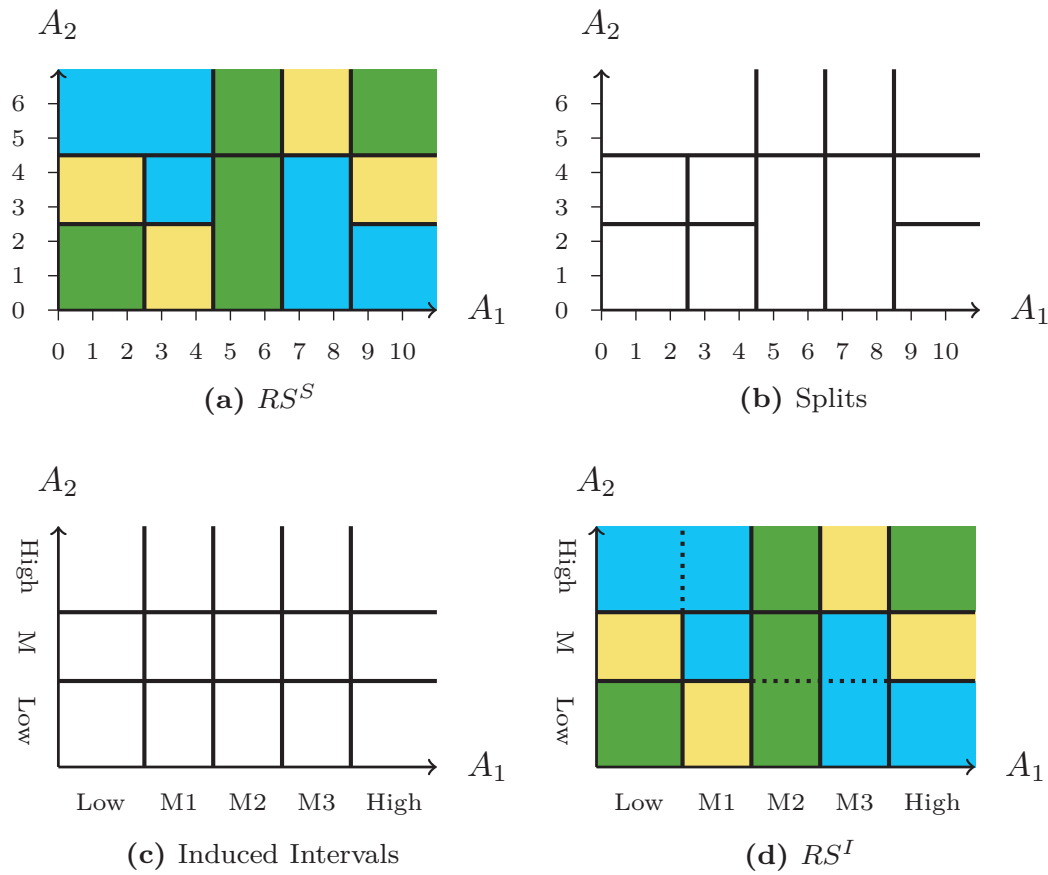


Figure 4.7.: The transformation of a split-based into an interval-based ruleset.

discretization is rather ad hoc, which is not a bad thing per se. But sometimes it may happen that this discretization is unable to grasp the underlying logic. Consider the question of whether a person is in puberty or not. For this concept the discretization into decades is treacherous. Consider Figure 4.8a where this discretization leads to puberty for $[0 - 20)$ year old persons and to no puberty for persons between $[20 - 40)$ and $[50 - 70)$ years. This concept is problematic due to several factors: (A) This generalization concludes that small children and also adults older than 17 years are in puberty although there is no such evidence, (B) It is unknown whether people in the 40s or in the 70s are in puberty, (C) The number of bins is larger than actually needed.

A data-driven discretization for this problem is shown in Figure 4.8b. This concept does not suffer from the problems that were found for the pre-defined

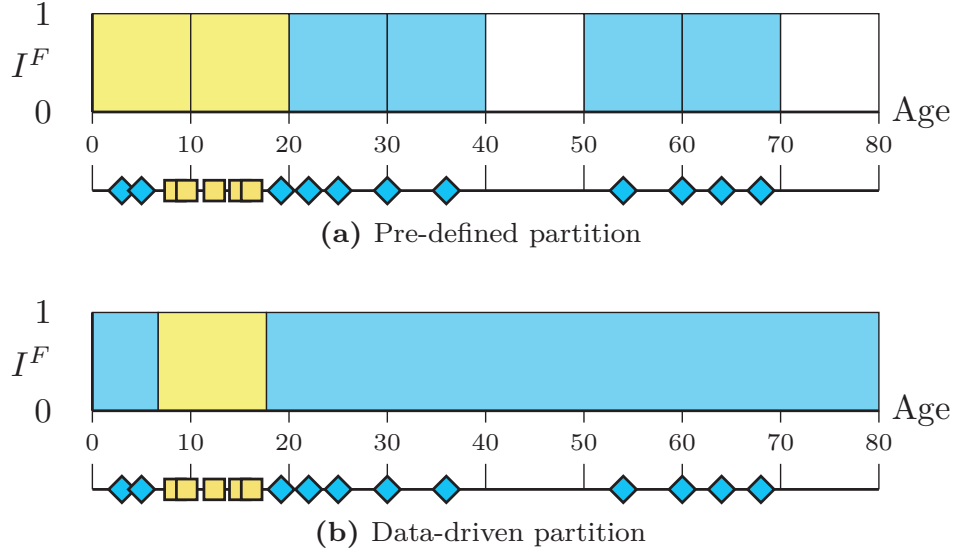


Figure 4.8.: Describing the concept of puberty using different forms of discretization. People in puberty are denoted by \blacksquare and not in by \blacklozenge .

partitioning: All ages are covered, neither are small children in puberty nor people in their 40s or 70s. Moreover, the concept is much simpler, since there are only two split points which enclose the people in puberty.

Note that this example was a rather simplistic demonstration of the benefits of data-driven fuzzification. This example was not meant to illustrate the multi-dimensional data-driven discretization of HELLFIRE.

4.2.3. Pruning

A common approach to avoiding overfitting and to enhancing generalization capabilities is to prune rule conditions [BP91, FW94, Für97, Bos04]. The pruning approach used for the HELLFIRE algorithm is applied on the split ruleset RS^S . Essentially, it serves two purposes: (A) The pruning should shorten the rule length by removing conditions. (B) The pruning should remove unnecessary splits which leads to a smaller number of intervals. This is the case if a distinct type of split is removed from all rules.

The pruning procedure considers each rule $r^S \in RS^S$ individually and examines whether the deletion of a condition, i.e. split, does not worsen the rule accuracy. In such a case the condition will be removed. The order in

which the conditions are considered is from the back towards the beginning of the rule since the conditions at the end are typically the ones that might be responsible for overfitting [Coh95]. Each time a condition could be removed, the pruning process restarts in order to take the modified overall rule coverage into consideration.

Even though this kind of pruning is not conducted on a separate pruning set, it is quite effective at reducing rule size and at increasing classification accuracy on the testing data.

4.2.4. Rule Selection

The pruning process generalized the rules in RS^S by removing split conditions. The increased rule coverage leads to situations where instances might be covered by several rules, potentially of the same class. This redundancy should be taken into consideration by finding a sufficient subset of rules without sacrificing classification ability. For this purpose a very simple set covering approach $SETCOVERRULES(RS^I, D)$ is used that selects rules until the training data is sufficiently covered, cf. Algorithm 5, Line 18. The selection criterion looks for the rule with the largest positive coverage among the still uncovered examples. It stops selecting rules as soon as the coverage on the uncovered data is less than 5% of the rules' positive coverage on the training data.

In Algorithm 5, the ruleset covering is applied after transforming the split-based ruleset RS^S into the interval-based ruleset RS^I . This order is not mandatory since the transformation does not change the rule coverage in any way.

4.2.5. Interval Fuzzification

Up until now, the ruleset RS^I learned by the HELLFIRE algorithm consists of rules containing conventional intervals in the antecedent part. To soften the decision boundaries, leading to a fuzzy ruleset RS^I , the conventional intervals $I = [b, c]$ will be replaced by fuzzy intervals of the form $I^F \stackrel{\text{df}}{=} [a, b, c, d]$, where b and c denote the fuzzy interval core bounds and a and d the fuzzy interval support bounds, see Figure 2.3 on page 26.

The fuzzification procedure starts with the interval rules RS^I that have been learned so far. The transformation from the conventional rule $r^I \in RS^I$ to the fuzzy rule $r^F \in RS^F$ is done as follows: First of all, the conventional

intervals $I = [b, c]$ were replaced with fuzzy intervals $I^F = [b, b, c, c]$ by simply assuming that the support boundaries of the lower and upper side of the interval equal the respective core boundaries. Secondly, the logic operators \vee and \wedge were replaced with a fuzzy T-conorm \perp and a fuzzy T-norm \top . Given fuzzy memberships μ_1, μ_2 we propose the product T-norm $\mu_1 \top \mu_2 \stackrel{\text{df}}{=} \mu_1 \cdot \mu_2$ as conjunction. As disjunction operator we suggest the Łukasiewicz T-conorm $\mu_1 \perp \mu_2 \stackrel{\text{df}}{=} \min(1, \mu_1 + \mu_2)$. This leads to the fuzzy rule membership function:

$$\mu_{r^F}(\mathbf{x}) \stackrel{\text{df}}{=} \min \left(1, I_{i,i_1}^F(x_{i_1}) + \min \left(1, \dots \min \left(1, I_{i,i_{k-1}}^F(x_{i_{k-1}}) + I_{i,i_k}^F(x_{i_k}) \right) \right) \right) \\ \cdot \dots \cdot \\ \min \left(1, I_{j,j_1}^F(x_{j_1}) + \min \left(1, \dots \min \left(1, I_{j,j_{\ell-1}}^F(x_{j_{\ell-1}}) + I_{j,j_\ell}^F(x_{j_\ell}) \right) \right) \right) ,$$

where $\{i_1, \dots, i_k\} \subseteq \{1, \dots, |I_i^F|\}$ are indices of occurring fuzzy intervals and $|I_i^F|$ is the number of fuzzy intervals for A_i .

The fuzzification of the — at this point still conventional — interval I^F has the purpose of softening the decision boundary of the rule in which I^F occurs as a condition in the antecedent part. However, to be precise, there might be more than one rule involving I^F . This makes the fuzzification process more difficult since multiple regions of the data space are involved at once. Consequently, the fuzzification process should be conducted in a careful way that does not harm the role of I^F in any of the rules in which it occurs.

To soften the boundaries of the conventional fuzzy intervals, the existing core and support boundaries have to be manipulated. This process is shown in Figure 4.9 and consists of two steps. It starts with the interval partitioning (a) and tries to shrink the interval core of interval I_i^F in a data-driven way (b). This leads to soft boundaries but dead spots at the position of the original split. To reestablish the full attribute coverage, the support bounds of I_i^F are set using the core location of its neighbor intervals I_{i-1}^F and I_{i+1}^F (c).

The critical part of that procedure is the data-driven step (b) that shrinks the rule cores. The idea behind shrinking I^F is to find the most essential part of that fuzzy interval that should be fully covered. For that purpose HELLFIRE tries to find a subinterval $I_{\text{sub}}^F \subseteq I^F$ that has the maximum m -measured purity for $m = 2$. To deal with multiple occurrences of I^F in several rules, the purity will be averaged over those rules. Note that one

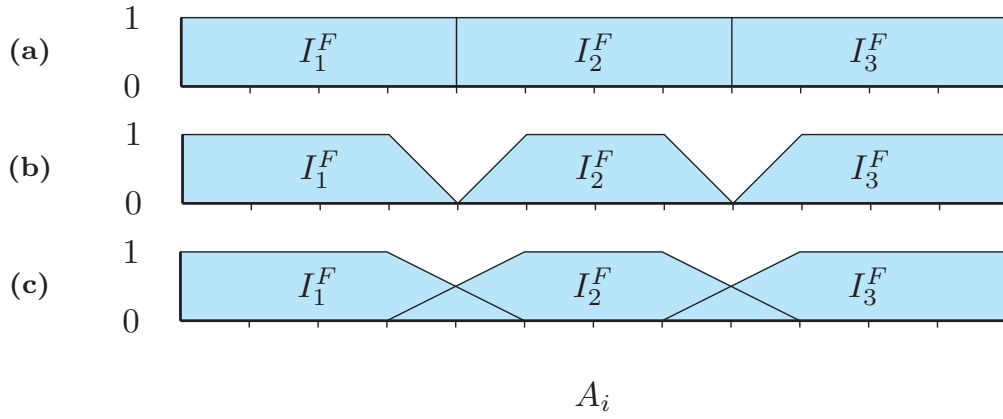


Figure 4.9.: A simplified sketch of the interval fuzzification strategy.

could also conceive other aggregation techniques. The set of subintervals for attribute A_i that should be considered is determined by all valid combinations of examples $\mathbf{x}_u, \mathbf{x}_v$ covered by I_i^F :

$$\{I_{i,\text{sub}}^F\} \stackrel{\text{df}}{=} \left\{ [x_{i,u}, x_{i,v}] \mid (\mathbf{x}_u, \lambda_{\mathbf{x}_u}), (\mathbf{x}_v, \lambda_{\mathbf{x}_v}) \in D \wedge \right. \\ \left. I_i^F(x_{i,u}) = I_i^F(x_{i,v}) = 1 \wedge x_{i,u} \leq x_{i,v} \right\}$$

Note that the simplified sketch in Figure 4.9 leaves the decision boundaries at the very same position throughout the process which, here, can be explained by the symmetry of the core shrinkage of adjacent fuzzy intervals. In practice this will not necessarily be the case since the shrinking process only incorporates local data of the currently considered interval. Consequently, the decision boundary could be shifted, cf. Figure 4.10a: Here, the original decision boundaries can still be found where the memberships are zero and where the skewed decision boundaries are at the crossings of the dashed lines. A very simple but effective solution is to mirror one core point along the old decision boundary. We propose to mirror the one that is farther away from

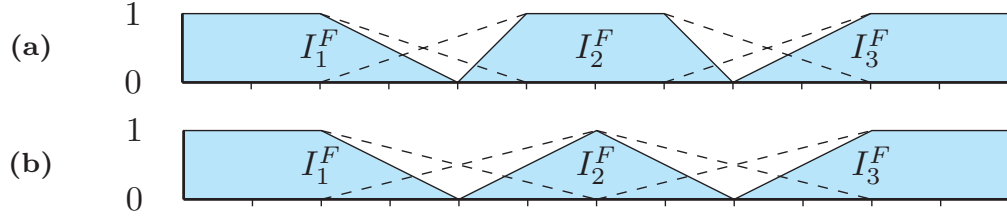


Figure 4.10.: (a) Different shrinkage of the cores lead to shifted decision boundaries. (b) Mirrored shrinkage resolves the shift and keeps the decision boundary in place.

the split s_i leading to smaller cores, cf. Figure 4.10b:

$$c_i \stackrel{\text{df}}{=} s_i - \max(\delta_{i,i+1}, \delta_{i+1,i})$$

$$b_{i+1} \stackrel{\text{df}}{=} s_i + \max(\delta_{i,i+1}, \delta_{i+1,i})$$

The exact shifting distances $\delta_{i,i+1}, \delta_{i+1,i}$ for the intervals I_i^F and I_{i+1}^F is given by:

$$\delta_{i,i+1} \stackrel{\text{df}}{=} \min\left(d_i - c_i, \frac{1}{2}(c_{i+1} - b_{i+1}) + b_{i+1} - a_{i+1}\right)$$

$$\delta_{i+1,i} \stackrel{\text{df}}{=} \min\left(b_{i+1} - a_{i+1}, \frac{1}{2}(c_i - b_i) + d_i - c_i\right)$$

Note that this definition takes care of situations involving swapped core boundaries $b_i > c_i$ which might occur if the upper and lower core shifts of an interval would be larger than its initial width.

When considering two adjacent fuzzy intervals $I_j^F = [a_j, b_j, c_j, d_j]$ and $I_{j+1}^F = [c_j, d_j, c_{j+1}, d_{j+1}]$ the reason for applying the Łukasiewicz T-conorm becomes clear: For every value $v \in [b_j, c_{j+1}]$, $I_j^F(v) + I_{j+1}^F(v) = 1$ holds, which can be easily proven by using the symmetry characteristics mentioned above. From this follows that the Łukasiewicz T-conorm keeps the disjunction of adjacent fuzzy intervals convex.

The order in which the intervals are dealt with plays a role since it affects the coverage of other intervals and therewith their fuzzification. In order to cope with this situation, we propose determining the examples that are cov-

ered by the respective intervals before starting shrinking. Then, the ordering has no effect.

4.2.6. Rule Weighting

HELLFIRE uses rule weights to cope with rules of different certainty. It was shown that those weights improve classification accuracy and also lead to more flexible boundaries [IN01, IY05]. The rule weight or certainty factor $CF(r^F)$ is determined by the performance of the rule on the training set. The assumption is that rules that worked very well on the training set D should also perform in the same manner on the test set. To avoid an overfitting effect caused by this idea, the m -measure with $m = 2$ is applied to determine the confidence factor. It prefers larger over smaller coverage when the accuracy is the same [PFTV92]:

$$CF(r^F) = CF(\langle r_A | \lambda_j \rangle) \stackrel{\text{df}}{=} \frac{2 \frac{|D^{(j)}|}{|D|} + \sum_{(\mathbf{x}, \lambda_j) \in D^{(j)}} \mu_{r^F}(\mathbf{x})}{2 + \sum_{(\mathbf{x}, \lambda) \in D} \mu_{r^F}(\mathbf{x})}, \quad (4.3)$$

where $D^{(j)} \subseteq D$ denotes the set of examples belonging to class j .

The rule weight must be calculated twice for every rule, cf. Algorithm 5, Line 17 and 21. The first time before selecting the rules with the set cover algorithm and the second time after fuzzification. The former procedure incorporates the rule confidence as a weight for the positive coverage that is used to determine the next rule in the set covering part. The second time the rule weight has to be calculated is after the fuzzification process to adjust for the changed coverage of the fuzzified rules.

4.2.7. Handling of Missing Values

In practice, the convenience of a complete data set without any missing attribute values is rarely encountered. Consequently, a classifier should only be able to treat missing values and to learn from the available example values. The decision tree-like learning approach of HELLFIRE is able to deal with missing values. There are two situations for which the treatment of missing values is essential: (A) during the calculation of the averaged entropies, since this is sensitive to missing values. Here HELLFIRE assumes that an example (\mathbf{x}, λ) which has a missing value x_i for the attribute A_i under investigation

is considered for the entropy of both sides of the split with half of its initial weight, (B) during the split procedure, since an interval having a missing value for the split attribute must not be put to either D_{\leq} or $D_{>}$. Instead it is added to both sides with half of its initial weight. The benefit of that approach is that a false assignment of (\mathbf{x}, λ) to either side is avoided.

4.2.8. Classification with Rule Stretching

In order to classify an unseen instance \mathbf{x} , all rules in RS^F are involved in a bound ballot, in which every rule r^F has a specific voting weight according to its certainty factor $\text{CF}(r^F)$ from (4.3) and the coverage degree $\mu_{r^F}(\mathbf{x})$. For combining multiple rules of the same class in this ballot the Łukasiewicz T-conorm $\perp(r_1^F, r_2^F) \stackrel{\text{df}}{=} \min(1, \mu_{r_1^F} \cdot \text{CF}(r_1^F) + \mu_{r_2^F} \cdot \text{CF}(r_2^F))$ is selected. This leads to a possibility distribution

$$s_j = \perp(r_1^{F(j)}, \dots, r_\ell^{F(j)}) ,$$

where $r^{F(j)}$ denotes a fuzzy rule of class j . HELLFIRE will predict the class

$$\lambda^* \stackrel{\text{df}}{=} \underset{\lambda_j \in \mathbb{L}}{\text{argmax}} s_j$$

having maximal possibility.

It may occur that \mathbf{x} is not covered by any rule at all. In such cases we apply the novel rule stretching procedure that was already explained for FURIA, cf. Section 3.2.5.

4.2.9. Complexity Analysis

A very important point when designing new classifiers is to keep the efficiency in mind. In order to analyze HELLFIRE in such terms we conduct a complexity analysis. To this end, let n be the number of training examples, k the number of attributes and d the depth of the induced tree-like structure — or in other terms: the number of splits made.

The training phase for building the model searches for the best coordinated split. For this purpose, the entropy of subsets of D must be calculated. Since the overall number of examples across those subsets is at most n for every single structure layer, the overall effort therefore is $O(d \cdot k \cdot n)$. To calculate

the split points, the data must be sorted to find the middle position between two adjacent examples, resulting in a complexity of $O(d \cdot n \log n)$.

The pruning procedure tries to remove splits from every rule as long as the quality does not decrease. Since the pruning restarts if a condition is removed, there might be up to d^2 weight calculations. Since the weight calculation must determine the coverage of all splits for every example, this could lead up to $n \cdot d$ checks. In total, the complexity of pruning is $O(d^3 \cdot n \cdot |RS^S|)$.

The set covering is done in a greedy way that determines the coverage of the available rules on the uncovered training data. In the worst case, when all rules have to be selected, there will be a quadratic number of sweeps over the training data. Consequently, the complexity is given by $O(|RS^I|^2 \cdot n)$.

The fuzzification process essentially consists of finding a subinterval with a maximum m -measure for $m = 2$. Since all numerical attributes are fully covered with disjunct fuzzy sets, every example must be considered only once per attribute. This can be effectively done with a simple sweep algorithm if the underlying data is sorted. Consequently, the complexity of fuzzifying all intervals boils down to $O(d \cdot n \log n)$.

The rule weight calculation is comparable to the performance measure during pruning. Due to the rule stretching procedure, the rule weights are precalculated for every rule stump, in comparison to the pruning this is not quadratically in the number of splits but linear. Consequently, the complexity is $O(d^2 \cdot n \cdot |RS^S|)$.

The overall complexity of HELLFIRE for building the model is

$$\begin{aligned} & O(d \cdot k \cdot n) + O(d \cdot n \log n) + O(d^3 \cdot n \cdot |RS^S|) + \\ & O(|RS^I|^2 \cdot n) + O(d \cdot n \log n) + O(d^2 \cdot n \cdot |RS^S|) \\ & = O(d^3 \cdot n \cdot |RS^S| + d \cdot n \log n + |RS^I|^2 \cdot n) \end{aligned}$$

under the assumption that $k < d^2 \cdot |RS^S|$.

To classify an unseen instance, all intervals of all rules have to be checked. Since the number of intervals per attribute is the same as the number of splits in terms of Landau-notation, it can be expressed as $O(d \times |RS^F|)$.

4.3. Comparing HELLFIRE with Other Discretization Techniques

The realm of discretization deals with the problem of partitioning numeric attributes into intervals. The reasoning behind this procedure is that the number of continuous values is typically much larger than the one of discrete values. Actually, the latter is an interval in a continuous spectrum. A discretized attribute can be considered as a generalization within the realm of classification and, thus, less sensitive to overfitting than a single value⁸ [LHTD02].

Liu et al. suggested a hierarchical framework in order to structure the different types of discretization [LHTD02]: They distinguish whether the discretization is: (A) supervised or unsupervised, (B) dynamic or static, (C) local or global, (D) top-down (splitting) or bottom-up (merging), (E) direct or incremental, (F) univariate or multivariate. The authors proposed to structure the techniques as shown in Figure 4.11.

This figure shows that the first characterization for a discretization algorithm is whether it uses splitting or merging and whether it is supervised or not. For the supervised splitting approaches the authors see four different techniques for obtaining the split values: entropy-based, binning, dependency and accuracy. Without going into detail, we will focus on the entropy-based techniques since HELLFIRE belongs to that category. Apart from HELLFIRE there are other techniques within this group which we will introduce shortly: (A) ID3 / C4.5, the decision tree discretization methods from Quinlan [Qui86, QCJ93], (B) *D2*, a recursive partitioning algorithm from Catlett [Cat91], (C) *MDLP*, an algorithm based on minimum description length from Fayyad and Irani [FI93].

Before digging deeper, we explain the remaining dichotomizations: (A) The aspect of local vs. global dichotomy describes whether the discretization considers the whole data space or only a local subset, while finding the split points. (B) The aspect of direct vs. incremental dichotomy describes whether all split points are found at once or whether one split is made after the other. (C) The aspect of static vs. dynamic describes whether the discretization is finished before building the model or whether the discretization is conducted while building the model.

⁸A single value v might be more prone to overfitting when it is used as a selector $A = v$ in comparison to an interval $I = [v - \epsilon, v + \epsilon]$ for $\epsilon > 0$ and a selector $A \in I$.

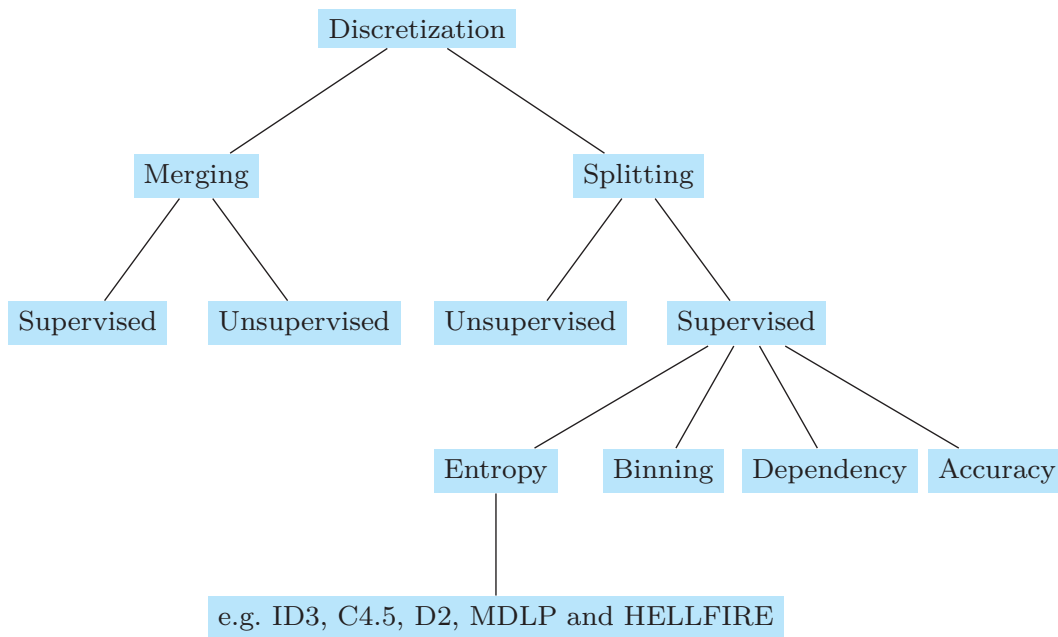


Figure 4.11.: Discretization framework (shortened) according to Liu et al. [LHTD02].

An overview of the entropy-based supervised splitting discretization algorithms and also *HELLFIRE* can be found in Table 4.1.

Note that *HELLFIRE* is somehow similar to ID3 since it also splits incrementally in a dynamic way, while building the model. The difference between *HELLFIRE* and ID3 is that ID3 is a pure decision tree approach which considers subtrees independently. This is the reason why ID3 can be considered as a local approach. By contrast, *HELLFIRE* uses a coordinated split criterion, which considers the whole data space at once. This makes *HELLFIRE* a global approach — on the one hand. However, the removal of adequately covered areas of the data space concentrates the coordinated split search on potentially disjunct regions. This makes *HELLFIRE* a local approach — on the other hand.

The literature on supervised discretization techniques is vast and a comprehensive survey is beyond the scope of this analysis. Nevertheless, to the best of the authors knowledge, no discretization method doing anything comparable to the coordinated splitting of *HELLFIRE* has been proposed yet.

Table 4.1.: Categorization of entropy-based supervised splitting discretization algorithms.

Method	Global / Local	Direct / Incremental	Static / Dynamic
ID3/C4.5	Local	Incremental	Dynamic
D2	Local	Incremental	Static
MDLP	Local	Incremental	Static
HELLFIRE	Global \rightarrow Local	Incremental	Dynamic

4.4. Experiments

To analyze the performance of HELLFIRE, we conduct several experimental studies. We implemented the HELLFIRE algorithm from scratch within the WEKA suite. As competitors we used the rule-based classifiers RIPPER, CHI and SLAVE, cf. Section 2.3.3, Section 2.5.3.2 and Section 2.5.3.3. The testbed for this comparison were the 45 binary- and multi-class data sets, cf. Table 2.1. The experimental settings followed the setup proposed in Section 2.5.5.

4.4.1. Classification Performance Analysis

In the first experiment, we compared HELLFIRE to other classifiers with respect to classification accuracy. Table 4.2 summarizes the results in terms of mean classification accuracies⁹.

The results of this test suggest that HELLFIRE is able to deliver good predictions in terms of classification accuracy. It is competitive on most data sets, cf. Table 4.2. To analyze the differences between the classifiers more closely, we followed the statistical procedures introduced in Section 2.5.6 and conducted the Friedman Test, cf. Table 4.2 for the classifier ranks. The corrected Friedman statistic for large N and k according to (2.5) is 12.31, while the critical value for the significance level $\alpha = 0.01$ is only 3.94. Thus, the null-hypothesis can quite safely be rejected, which means that there are significant differences in the classifiers' performance.

Given the result of the Friedman Test, we conduct the Bonferroni-Dunn Test as a post-hoc test to compare the competitors to the control classifier HELLFIRE [Dun61]. The critical distance according to (2.7) between two classifier ranks is $CD_\alpha = 0.58$. The results of this test are summarized in

⁹The classifier HELLFIRE^I, which also appears in the table, will be analyzed in Section 4.4.3.

Table 4.2.: Average classification accuracies and ranks for HELLFIRE and its competitors.

Data set	HELLFIRE	RIPPER	CHI	SLAVE	HELLFIRE ^I
acd-authorship	91.40(3)	93.05(1)	71.60(4)	91.87(2)	90.35
acd-bankruptcy	82.15(1)	81.97(2)	74.40(4)	77.80(3)	81.97
acd-cyyoung8092	80.28(1)	80.04(2)	70.72(4)	79.32(3)	80.10
acd-cyyoung9302	81.87(3)	82.01(2)	80.27(4)	83.90(1)	81.00
acd-esr	82.62(1)	82.38(2)	79.55(3)	77.72(4)	80.59
acd-halloffame	91.51(4)	92.87(1)	92.18(3)	92.68(2)	91.37
acd-lawsuit	96.94(2)	97.54(1)	94.93(3)	94.81(4)	96.25
acd-votesurvey	35.47(2)	34.40(3)	40.19(1)	29.51(4)	35.02
biomed	86.69(2)	87.40(1)	80.64(4)	84.74(3)	86.16
cars	71.84(2)	75.93(1)	68.97(4)	70.68(3)	74.31
collins	93.26(1)	92.89(2)	42.63(4)	50.87(3)	93.11
ecoli	79.67(3)	80.57(2)	77.43(4)	81.03(1)	79.33
eucalyptus	55.04(3)	58.69(1)	54.09(4)	58.16(2)	55.36
glass	64.04(1)	63.18(2)	61.39(4)	61.83(3)	65.53
haberman	73.00(3)	72.16(4)	73.08(2)	73.31(1)	72.82
heart-statlog	79.63(1)	78.44(2)	68.66(4)	78.44(3)	79.24
ionosphere	88.39(3)	88.64(2)	66.40(4)	89.83(1)	88.13
iris	94.82(2)	93.45(3)	92.27(4)	94.92(1)	94.82
liver-disorders	63.81(2)	65.93(1)	58.75(4)	59.77(3)	62.69
metStatCoord.	85.18(2)	92.04(1)	46.79(4)	58.77(3)	84.58
metStatRainfall	48.28(2)	60.66(1)	24.51(4)	29.35(3)	47.10
metStatRST	38.55(2)	36.08(3)	25.24(4)	42.02(1)	37.93
metStatSunshine	41.36(2)	44.48(1)	37.93(3)	28.83(4)	41.43
metStatTemp	47.72(1)	47.45(2)	30.63(3)	22.10(4)	47.86
mfeat-factors	83.30(4)	87.05(2)	89.19(1)	86.83(3)	79.34
mfeat-fourier	72.86(2)	71.37(3)	69.27(4)	73.49(1)	71.01
mfeat-karhunen	76.57(4)	79.13(2)	82.55(1)	78.37(3)	70.96
mfeat-morpholog.	70.95(1)	70.74(2)	57.93(4)	67.08(3)	70.26
mfeat-zernike	63.27(4)	67.58(3)	72.37(1)	68.26(2)	59.08
optdigits	82.69(3)	89.68(2)	45.90(4)	93.45(1)	79.35
page-blocks	93.08(3)	96.79(1)	91.96(4)	93.58(2)	92.85
pasture-prod.	71.74(1)	68.46(2)	44.23(4)	53.63(3)	71.99
pendigits	88.25(3)	95.54(2)	97.45(1)	87.26(4)	86.77
pima diabetes	72.92(3)	74.56(1)	72.55(4)	73.65(2)	72.49
prnn-synth	83.81(2)	82.50(3)	84.14(1)	81.51(4)	82.44
schizo-	70.52(2)	75.33(1)	56.08(4)	56.29(3)	72.20
segment	90.40(2)	94.53(1)	83.65(4)	88.87(3)	91.70
sonar	72.35(3)	72.41(2)	74.61(1)	68.50(4)	72.14
squash-unstored	76.40(1)	71.74(2)	70.56(3)	65.56(4)	75.84
synthetic control	82.49(3)	82.85(2)	68.33(4)	89.23(1)	86.40
vehicle	66.99(2)	67.80(1)	61.99(4)	64.08(3)	67.38
vowel	47.55(4)	64.71(1)	59.49(3)	63.84(2)	48.89
waveform	77.31(2)	78.72(1)	72.38(4)	75.34(3)	73.34
wine	91.88(3)	90.02(4)	92.77(1)	92.46(2)	91.40
w.-breast-cancer	94.51(3)	95.58(1)	90.20(4)	95.49(2)	93.78
average rank	2.31	1.82	3.27	2.6	

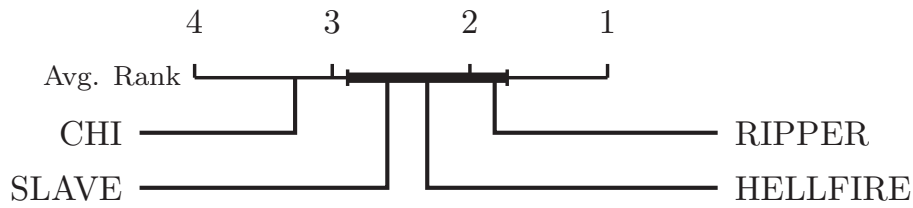


Figure 4.12.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. HELLFIRE is significantly better than CHI in terms of accuracy. Significance level $\alpha = 0.1$.

Figure 4.12: HELLFIRE is better than CHI at the significance level $\alpha = 0.1$. In the comparison to SLAVE and RIPPER there are no significant differences.

From a qualitative point of view the results are satisfying: (A) HELLFIRE is not much worse than RIPPER. This is astonishing since RIPPER is more flexible in choosing the rule boundaries. HELLFIRE could achieve RIPPER's flexibility only with doing more splits in the initial learning phase, leading to a large number of fuzzy sets which is bad for the interpretability. As will be shown below this is not the case. (B) HELLFIRE is a little better than SLAVE which uses a genetic search that is much slower than the decision tree-like learning.

4.4.2. Ranking Performance Analysis

The second analysis of the discriminative power is conducted by measuring the area under the ROC curve (AUC). We used the same competitors, the same settings and the same choice of data sets as in the test before. Table 4.3 summarizes the results in terms of mean AUC⁹.

The results from the AUC analysis are quite different from the ones of the accuracy analysis. Now HELLFIRE ranks at the very first position and RIPPER descends to the last place. Besides, CHI overtakes SLAVE and scores second in this test. Cf. Section 3.3.2 for an explanation this kind of observation.

To evaluate the performances in more detail, we conduct the Friedman Test, cf. Table 4.3 for the classifier ranks. The corrected Friedman statistic for large N and k according to (2.5) is 6.84, while the critical value for the significance level $\alpha = 0.01$ is only 3.94. Thus, the null-hypothesis can be rejected, which means that there are significant differences in the classifiers'

Table 4.3.: Average AUC and ranks for HELLFIRE and its competitors.

Data set	HELLFIRE	RIPPER	CHI	SLAVE	HELLFIRE ^I
acd-authorship	0.98(1)	0.96(3)	0.90(4)	0.98(2)	0.95
acd-bankruptcy	0.88(1)	0.82(4)	0.87(2)	0.86(3)	0.82
acd-cyyoung8092	0.78(1)	0.71(3)	0.65(4)	0.77(2)	0.82
acd-cyyoung9302	0.77(2)	0.71(4)	0.75(3)	0.79(1)	0.72
acd-esr	0.67(1)	0.62(2)	0.60(3)	0.59(4)	0.72
acd-halloffame	0.78(4)	0.79(3)	0.87(2)	0.89(1)	0.70
acd-lawsuit	0.95(2)	0.92(3)	0.81(4)	0.96(1)	0.70
acd-votesurvey	0.54(2)	0.50(3)	0.55(1)	0.49(4)	0.62
biomed	0.91(2)	0.86(4)	0.93(1)	0.91(3)	0.62
cars	0.85(2)	0.82(3)	0.86(1)	0.81(4)	0.75
collins	0.99(1)	0.97(2)	0.88(3)	0.83(4)	0.93
ecoli	0.92(2)	0.89(4)	0.94(1)	0.91(3)	0.93
eucalyptus	0.81(2)	0.82(1)	0.79(3)	0.77(4)	0.53
glass	0.80(1)	0.76(4)	0.77(3)	0.79(2)	0.87
haberman	0.63(2)	0.60(3)	0.65(1)	0.60(4)	0.82
heart-statlog	0.84(1)	0.79(3)	0.76(4)	0.81(2)	0.97
ionosphere	0.90(2)	0.88(4)	0.89(3)	0.92(1)	0.78
iris	0.98(3)	0.96(4)	1.00(1)	0.98(2)	0.78
liver-disorders	0.66(1)	0.64(2)	0.57(4)	0.60(3)	0.60
metStatCoord.	0.96(2)	0.98(1)	0.78(4)	0.82(3)	0.60
metStatRainfall	0.82(2)	0.85(1)	0.76(3)	0.66(4)	0.83
metStatRST	0.71(1)	0.70(2)	0.59(3)	0.58(4)	0.83
metStatSunshine	0.77(2)	0.75(4)	0.80(1)	0.75(3)	0.88
metStatTemp	0.76(2)	0.77(1)	0.66(4)	0.68(3)	0.97
mfeat-factors	0.97(2)	0.95(4)	0.98(1)	0.97(3)	0.64
mfeat-fourier	0.95(1)	0.92(4)	0.92(2)	0.92(3)	0.92
mfeat-karhunen	0.96(2)	0.92(4)	0.98(1)	0.95(3)	0.89
mfeat-morpholog.	0.91(3)	0.94(1)	0.93(2)	0.89(4)	0.89
mfeat-zernike	0.92(2)	0.90(4)	0.95(1)	0.90(3)	0.93
optdigits	0.96(1)	0.96(3)	0.82(4)	0.96(2)	0.79
page-blocks	0.83(3)	0.93(1)	0.82(4)	0.83(2)	0.74
pasture-prod.	0.81(1)	0.79(2)	0.64(4)	0.70(3)	0.74
pendigits	0.98(3)	0.98(4)	1.00(1)	0.99(2)	0.85
pima diabetes	0.77(2)	0.71(4)	0.80(1)	0.76(3)	0.85
prnn-synth	0.89(3)	0.84(4)	0.90(1)	0.90(2)	0.76
schizo-	0.75(2)	0.78(1)	0.56(4)	0.59(3)	0.76
segment	0.98(1)	0.98(2)	0.97(3)	0.96(4)	0.97
sonar	0.78(2)	0.74(4)	0.82(1)	0.75(3)	0.74
squash-unstored	0.85(1)	0.77(2)	0.71(4)	0.77(3)	0.74
synthetic control	0.97(3)	0.93(4)	0.99(1)	0.99(2)	0.80
vehicle	0.86(1)	0.85(2)	0.85(3)	0.83(4)	0.94
vowel	0.85(4)	0.88(3)	0.91(1)	0.89(2)	0.83
waveform	0.92(1)	0.88(4)	0.91(3)	0.91(2)	0.76
wine	0.97(3)	0.93(4)	0.98(1)	0.97(2)	0.90
w.-breast-cancer	0.98(2)	0.96(4)	0.98(1)	0.97(3)	0.96
average rank	1.89	2.96	2.38	2.78	

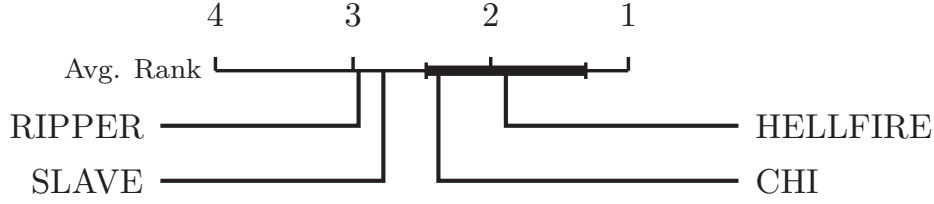


Figure 4.13.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. HELLFIRE is significantly better than RIPPER and SLAVE in terms of AUC. Significance level $\alpha = 0.1$.

performance. The critical distance according to (2.7) between HELLFIRE as control classifier and any competitor is $CD_\alpha = 0.58$ for a significance level of $\alpha = 0.1$.

The results of this test are summarized in Figure 4.13: HELLFIRE is significantly better than SLAVE and RIPPER, while there is no such difference in comparison to CHI at the significance level $\alpha = 0.1$.

4.4.3. Fuzzification Analysis

The fuzzification strategy proposed for HELLFIRE builds upon a rather simple principle that only concentrates on the interval cores. The question is whether this approach improves the overall performance. In analyzing this question, we conduct a test using the rules RS^I induced by just omitting the fuzzification step denoted as $HELLFIRE^I$. The comparison of HELLFIRE with $HELLFIRE^I$ leaves no doubt: 33 wins versus 11 losses (1 tie) for HELLFIRE show that fuzzification has a very significant impact on classification, cf. Table 4.2. A very similar picture is conveyed when conducting the same comparison for the AUC measurements: HELLFIRE wins 30 data sets and loses 15 over $HELLFIRE^I$. According to the sign test, cf. Section 2.5.6.1, under the assumption of equal performance we can reject this null-hypothesis with an error probability of $\alpha = 0.02$ for each analyses individually.

4.4.3.1. Effect of Fuzzification

To investigate the improvements that are due to fuzzification, we will follow the same approach we used for the FURIA algorithm, cf. Section 3.3.3. This setup will be slightly different since HELLFIRE uses another fuzzification

scheme for another type of rule. The fuzzification procedure of HELLFIRE modifies the fuzzy interval core and support boundaries in such a way that formerly adjacent but disjunct intervals will now overlap. The consequences are that the rule weighing might lead to a shifting of the decision boundaries and that the coverage of the fuzzified rules increases. To cope with this situation, we will conduct tests using modified variants of HELLFIRE. We hope to explain the consequences of fuzzification by examining its individual effects with this sort of differential diagnosis. For this test we will focus on classification accuracy only.

Setup

In analyzing the impact of fuzzification, we consider the discriminative power of the overall rule model. To minimize the interference of other effects, we will disable the rule stretching technique for this test. Since HELLFIRE is now allowed to abstain from classifying for uncovered instances, we have to take this into consideration and measure both classification accuracy and classification error, i.e. the relative number of correctly and incorrectly classified instances. To explain the effects due to fuzzification we have now four different variants that have to be compared. These variants resemble all possibilities of the dichotomies conventional vs. fuzzy and weighted vs. unweighted rules:

HELLFIRE The classifier as it was suggested (weighted and fuzzy).

HELLFIRE^I HELLFIRE weighted but without fuzzification.

HELLFIRE^U HELLFIRE with fuzzification but without weighting.

HELLFIRE^{I,U} HELLFIRE unweighted without fuzzification.

The complete results of this test can be found in Table A.6 and Table A.7. A short overview can be found in Table 4.4. To cope with the large amount of information, we will analyze those tables step by step:

The effect of rule weights on the fuzzified and on the conventional variant of HELLFIRE

A frequently encountered opinion in the literature is that rule weights improve classification accuracy [IN01, IY05, ACH03, JT08, FJ09]. The typical reason given is that a larger rule weight resembles a better rule since it is

Table 4.4.: Wins and losses in terms of classification accuracy and error on the test data for variants of HELLFIRE.

	Accuracy				Error			
	HF.	HF. ^I	HF. ^U	HF. ^{I,U}	HF.	HF. ^I	HF. ^U	HF. ^{I,U}
HELLFIRE	-	35	42	44		27	42	41
HELLFIRE ^I	10	-	32	42	18	-	37	42
HELLFIRE ^U	3	12	-	40	3	7	-	33
HELLFIRE ^{I,U}	1	1	4	-	4	1	11	-

often defined through a quality measure taking both purity and coverage into consideration, e.g. Laplace-weighted purity.

When considering conventional classification rules, the rule weight serves as a tie-breaker when multiple rules cover the query instance in a single-winner scenario. When considering a voting scheme, the rule weight works as a discounting effect on weaker rules. HELLFIRE determines the winning class using a voting procedure that combines the coverage of different rules from the same class using the Łukasiewicz T-conorm.

The results suggest that the weighted conventional variant HELLFIRE^I outperforms the unweighted conventional variant HELLFIRE^{I,U} with 42 wins and 1 loss in terms of both classification accuracy and classification error, cf. Table 4.4.

For the fuzzy variant HELLFIRE, the rule weight has a secondary effect: In the previous test with the conventional variants, adjacent intervals were disjunct. But now adjacent intervals overlap due to fuzzification. The consequence is that the effective decision boundaries are shifted according to the rule weight: A stronger rule can push the decision boundary away from its core and increase its area of influence into the territory of weaker rules. Figure 4.14 depicts the membership degrees according to Example 3.2.2 (see page 54). We see that for the fuzzy variants, there are only smooth and continuous changes. Besides, we find that in the unweighted case in Figure 4.14a, the influence of the three HELLFIRE fuzzy rules is somehow balanced. Yet, in the weighted case Figure 4.14b, the rule for the green class has a larger rule weight that decreases the other rules' territory. As we expected, the shift of the rule boundaries cannot be found with the conventional variants even if there is some imbalance in the rule weights, see Figure 4.14c.

The outcome displays that the weighted fuzzy variant HELLFIRE clearly outperforms the unweighted fuzzy variant HELLFIRE^U with 42 wins and 3

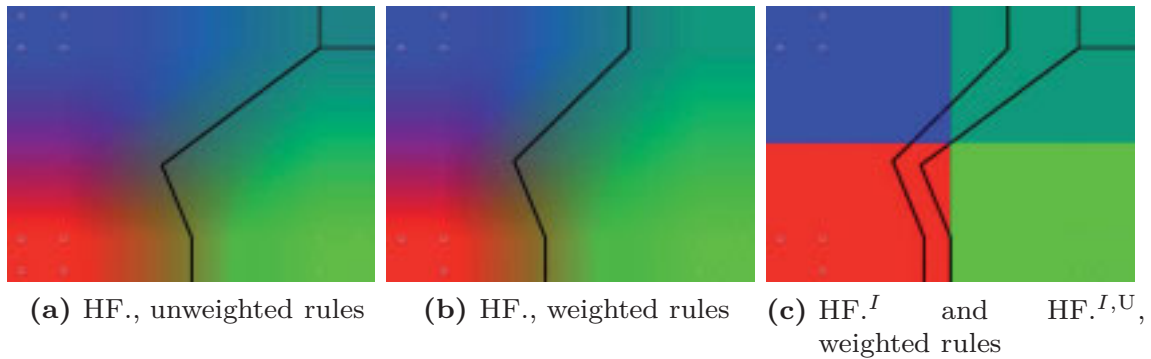


Figure 4.14.: Visualization of the decision boundaries. The different colors resemble the (fuzzy) coverage to the classes. The line corresponds to the decision boundary between the green and the remaining classes.

loss in terms of both classification accuracy and classification error, cf. Table 4.4. But these results do not shed any light on whether the improvement of using weighted rules is due to the discounting effect — which we also found for the weighted conventional variant — or because of decision boundary shift. Thus, we have to refine the analytical perspective:

The effect of fuzzification on the rule coverage

A characteristic of the fuzzification process is that it increases the width of the intervals leading to gradual overlaps. Consequently, the number of examples covered through a rule cannot shrink because of fuzzification. In fact, the experimental results can confirm this assumption: For all but three data sets, HELLFIRE covers more instances than HELLFIRE^I . The three exceptions are all completely covered by both variants. See Table A.6 and Table A.7 for more information.

This insight might be interesting, however it is not very helpful for the investigation into fuzzification benefits. Even worse, this observation shows that in order to avoid a Pareto-optimal situation, a variant has to outperform another variant in terms of both accuracy and error.

The effect of fuzzification on the discriminative quality

From the first analysis we learned that the rule weight has a beneficial effect on classification accuracy and classification error. But for the fuzzy case, we

could not separate the reason for the improvements due to some interference with the gradual coverage degrees.

To find out whether there is any benefit that can be attributed to fuzzification and fuzzification only, we must conduct further tests. We start with comparing the weighted fuzzy variant HELLFIRE with the weighted conventional variant HELLFIRE^I. The results in terms of accuracy are very clear: HELLFIRE wins 35 and loses 10 data sets over HELLFIRE^I. In terms of classification error there are only 27 wins over 18 losses. The null-hypothesis of equal performance in terms of classification error can be rejected using a sign test with an error probability of 0.12, cf. Section 2.5.6.1.

A reasonable assumption would be that the increased rule coverage would lead to a larger number of both correctly and incorrectly covered instances, explaining those improvements. But the results show that the absolute number of incorrectly classified instances is smaller for the fuzzy version HELLFIRE on 27 data sets. Furthermore, on all those data sets, HELLFIRE has also a better classification accuracy than HELLFIRE^I, cf. Table A.6 and Table A.7. Consequently, the aspect of fuzziness and not the increased coverage must be responsible for the improvements.

Following this line of reasoning, we can again consider the two unweighted variants for which the coverage examination is valid in exactly the same way. When analyzing the difference between HELLFIRE^U and HELLFIRE^{I,U} we might find the improvement of fuzzification caused neither by the shifting of the decision boundaries nor by the discounting of weaker rules. In fact, the results show that HELLFIRE^U wins 40 data sets with 1 loss in terms of accuracy and wins 33 data sets with 11 losses in terms of classification error over HELLFIRE^{I,U}. Again, we find that for every data set, for which HELLFIRE^U is better in terms of classification error, it is also better in terms of classification accuracy. Combining this outcome with the previous one, we find that there are benefits from fuzzification and fuzzification only. These improvements are neither due to increased coverage nor due to shifting rule boundaries, nor due to discounting weaker rules.

Synthesis

From the preceding analysis, we learned that fuzzification has a significant beneficial impact on classification accuracy and classification error. In this kind of differential diagnosis we revealed that there were improvements that were neither caused by the rule weights nor by the generalizing effect. From

this we can conclude that the gradual coverage, which assigns instances being nearer to the interval core higher membership degrees, must have a beneficial effect.

4.4.3.2. Comparing HELLFIRE Fuzzification with a Generic Fuzzification

The fuzzification procedure of HELLFIRE is a data-driven process that tries to find a reasonable degree of fuzzification for each fuzzy interval individually. Therefore, it considers the data covered by an interval as well as the interval's occurrence in potentially multiple rules. One might ask whether this effort has any advantages over a non-data-driven fast generic fuzzification procedure. Kuwajima et al. investigated different degrees of fuzzification $f \in [0, 1]$ to a grid-based rule model [KNI08]: A fuzzification degree of $f = 0$ resembled the conventional interval, while a degree of $f = 1$ is the maximal possible fuzziness.

Starting with two adjacent conventional intervals $I_i = (b_i, c_i]$ and $I_{i+1} = (c_i, c_{i+1}]$ we would set

$$\begin{aligned} I_i^F &\stackrel{\text{df}}{=} [b_i, b_i, a_{i+1}, b_{i+1}] \\ I_{i+1}^F &\stackrel{\text{df}}{=} [a_{i+1}, b_{i+1}, c_{i+1}, c_{i+1}] \end{aligned}$$

with

$$\begin{aligned} a_{i+1} &\stackrel{\text{df}}{=} c_i - f \cdot \max(c_i - b_i, c_{i+1} - c_i) \\ b_{i+1} &\stackrel{\text{df}}{=} c_i + f \cdot \max(c_i - b_i, c_{i+1} - c_i) \quad . \end{aligned}$$

See Figure 4.15 for an example. Note that the fuzziness is limited through the fuzzy set normality criterion according to Definition 2. Note that in Figure 4.15 the interval I_{i+1}^F is not fuzzified any more at the maximal degree of $f = 1$ because of this normality demand. Besides, this kind of fuzzification leaves the decision boundary at the position of the conventional split.

We conducted two tests to investigate whether the effort of the data-driven fuzzification is justified. In the first test we concentrated on classification accuracy and in the second on AUC. We compared the original HELLFIRE algorithm with generic fuzzification variants HELLFIRE^f , where $f \in \{0, 0.2, \dots, 1\}$ are the degrees of fuzzification we tested.

A win-loss overview of the accuracy tests are listed in Table 4.5 and the

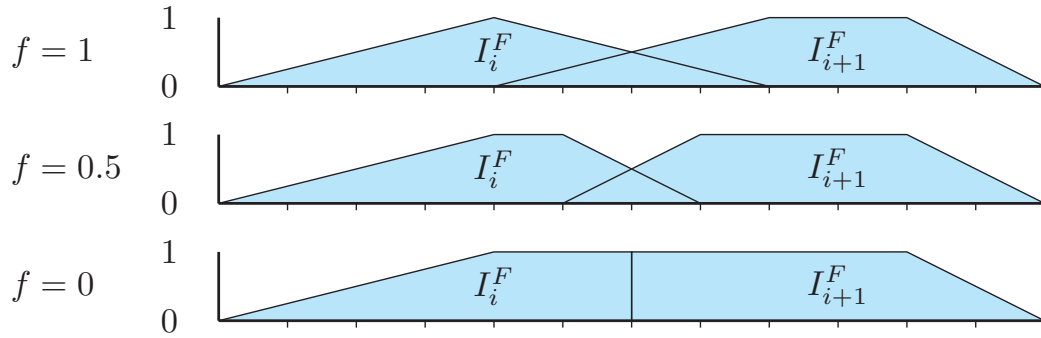


Figure 4.15.: Different fuzzification degrees f for two adjacent fuzzy intervals.

Table 4.5.: Wins and losses in terms of classification accuracy for HELLFIRE with different fuzzification degrees.

	HF.	HF. ⁰	HF. ^{0.2}	HF. ^{0.4}	HF. ^{0.6}	HF. ^{0.8}	HF. ¹
HELLFIRE	-	33	30	30	35	39	41
HELLFIRE ⁰	12	-	14	20	27	30	34
HELLFIRE ^{0.2}	15	31	-	30	33	37	41
HELLFIRE ^{0.4}	15	25	15	-	39	41	41
HELLFIRE ^{0.6}	10	18	12	6	-	45	45
HELLFIRE ^{0.8}	6	15	8	4	0	-	43
HELLFIRE ¹	4	11	4	4	0	2	-

complete results can be found in Table A.8. We see that the original variant of HELLFIRE is able to outperform all other generic variants with at least 30 wins over 15 losses.

At this point we are not conducting the multi-classifier evaluation using the Friedman Test and some post-hoc test on purpose. The reason is that we want to know whether there is one HELLFIRE^f variant that is able to dominate the original HELLFIRE algorithm. Making this observation would reject our claim that the data-driven fuzzification method is preferable in comparison to one generic variant. For this study we are considering a “domination plot” that plots the number of data sets where a classifier scored under the top k ranks, see Figure 4.16. We find that HELLFIRE is able to dominate the generic competitors in terms of 1st, 2nd and 3rd positions on up to 37 data sets. Interestingly, the moderately fuzzified $\text{HELLFIRE}^{0.4}$ and $\text{HELLFIRE}^{0.2}$ are the generic variants that work the best. Interestingly, too much or too less fuzzification is not optimal. It is quite remarkable that

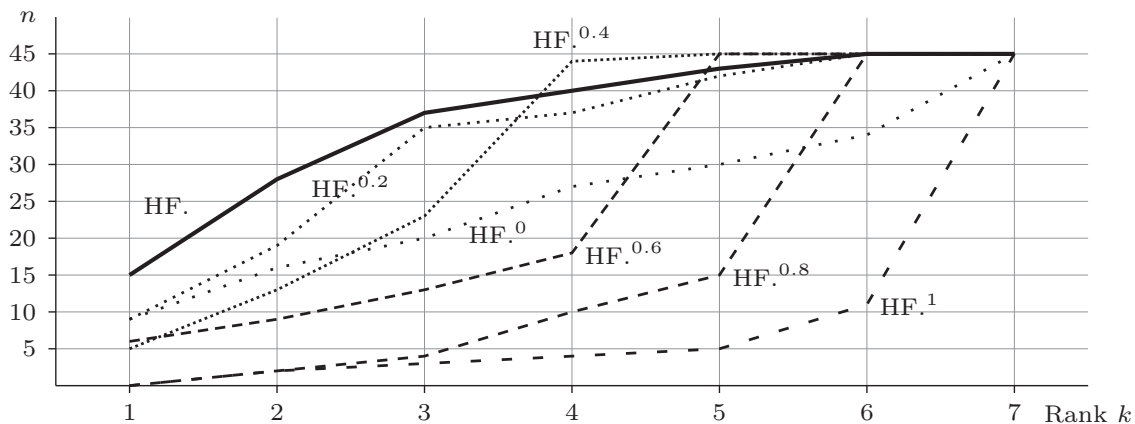


Figure 4.16.: Domination plot: Number of data sets n a classifier ranked among the top k ranks in terms of classification accuracy. (Note the linear interpolation between the ranks for a better visualization.)

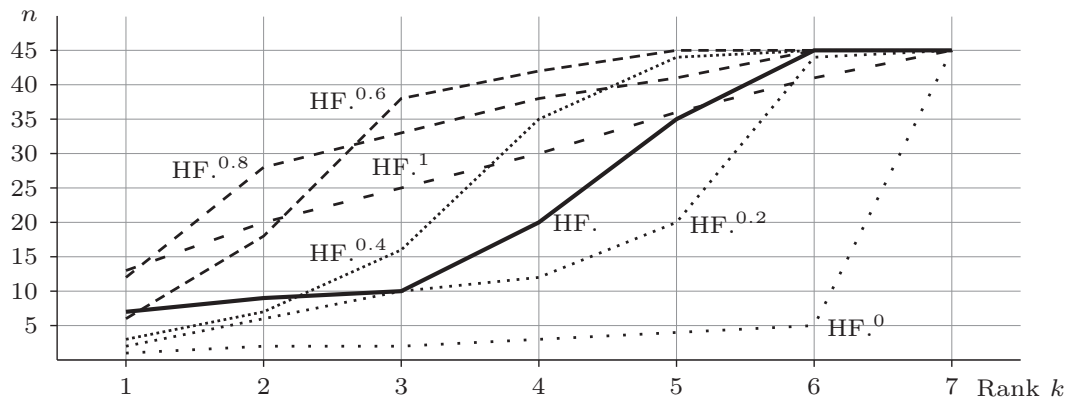
the variant HELLFIRE¹, which is using the softest boundaries, is dominated by all competitors. A potential explanation is that the effective decision boundary shifting caused by the rule weights is more severe, the more fuzzy the rules are, cf. Section 4.4.3.1. Moreover, the reduced coverage degree in safely covered areas could also have a detrimental effect on the final voting procedure.

We also conducted the same test for the ranking performance measure AUC. Our expectation is that larger fuzzification degrees f improve the AUC due to the increased score value spectrum according to the findings of Hüllermeier and Vanderlooy [HV09]. We find this characteristic here because of the smaller fuzzy interval core which always return a fuzzy membership of 1, whereas the fuzzy support area that returns gradual coverage increases. A win-loss overview of the AUC tests are listed in Table 4.6 and the complete results can be found in Table A.9.

This table displays that HELLFIRE is unable to outperform the generic variants in terms of ranking performance. It loses clearly against the ones having a fuzzification degree of $f > 0.6$. The first impression is that the variants with a higher fuzzification outperform the ones with a lower one. What is clear from this analysis is that a small degree of fuzzification deteriorates the AUC performance, which can be found for the variants with $f < 0.2$. But at a second glance it seems that a degree of $f = 0.8$ is even better than $f = 1$. From the literature we know that a larger spectrum of return

Table 4.6.: Wins and losses in terms of AUC for HELLFIRE with different fuzzification degrees.

	HF.	HF. ⁰	HF. ^{0.2}	HF. ^{0.4}	HF. ^{0.6}	HF. ^{0.8}	HF. ¹
HELLFIRE	-	44	31	17	10	11	13
HELLFIRE ⁰	1	-	2	2	3	4	5
HELLFIRE ^{0.2}	14	43	-	9	7	9	13
HELLFIRE ^{0.4}	28	43	36	-	10	14	20
HELLFIRE ^{0.6}	35	42	38	35	-	19	25
HELLFIRE ^{0.8}	34	41	36	31	26	-	29
HELLFIRE ¹	32	40	32	25	20	16	-

**Figure 4.17.:** Domination plot: Number of data sets n a classifier ranked among the top k ranks in terms of AUC. (Note the linear interpolation between the ranks for a better visualization.)

values improves the AUC, but the 29 wins and 16 losses of HELLFIRE^{0.8} over HELLFIRE¹ suggest that there is some kind of saturation effect. This impression is also supported when considering the domination plot, cf. Figure 4.17: HELLFIRE¹ does not dominate the field as one could expect. The saturation effect might be due to the fact that at some level the variety of the scores is already sufficient to resolve the ties and that a further fuzzification discount the very heart of the fuzzy intervals with the instances that definitely belong to that interval.

From an overall point of view, the performance of the data-driven fuzzification strategy of HELLFIRE is very convincing. It outperforms the generic variants in terms of classification accuracy significantly. Since HELLFIRE

was designed for classification tasks, this is a very convincing result. Unfortunately, the outcome of the fuzzification analysis concerning AUC performance was not very surprising: For a well-ranking classifier, a generic fuzzification degree of $f \in [0.6, 1]$ achieves very good results. HELLFIRE is here a midfielder: It dominates the conventional variant HELLFIRE⁰ and HELLFIRE^{0.2}. However, by comparison with the more fuzzy ones it is the data-driven variant that is dominated most of the time. In general, it seems that a concession has to be made: Either choosing a strong classification performance with a smaller degree of fuzzification or choosing a strong ranking performance with a larger degree of fuzzification.

4.4.4. Comparison between the Discretization Procedures of HELLFIRE and MDLP

We already mentioned the MDLP discretization from Fayyad and Irani in Section 4.3 as a representative from the group of entropy-based, supervised, splitting approaches — the group to which HELLFIRE belongs as well [FI93].

The MDLP algorithm partitions each numeric attribute recursively in such a way that the entropy is minimized. The splitting of a branch halts as soon as the gain for the split is lower than a certain threshold value that is based on minimum description length [FI93].

In order to compare HELLFIRE with MDLP a concession has to be made. While HELLFIRE learns the discretization and the rules simultaneously, MDLP only discretizes. Unfortunately, it is impossible to separate the rule learning procedure from the discretization process of HELLFIRE and to combine it with the discretization process of MDLP. Consequently, a comparison can be made between the discretization procedures only. For this purpose we will use the RIPPER rule learner in order to learn a rule-based model using the discretized data from MDLP and HELLFIRE, respectively. As a measure of discriminative performance we will evaluate classification accuracy and as a measure of model complexity we consider the number of rules (including RIPPER's default rule).



The results of this comparison are presented in Table 4.7. In order to evaluate these results statistically we will use the sign test, cf. Section 2.5.6.1, under the null-hypotheses of equal classifier performance and equal model sizes. For classification accuracy the sign test is unable to reject the null-hypothesis: The discretization of MDLP wins 23 data sets while the discretization of HELLFIRE (H-Dis) wins 22. However, when comparing the

model sizes, H-Dis learns a smaller model on 29 and a bigger model on only 14 data sets. The sign test rejects the null-hypothesis, indicating that the models of H-Dis are significantly smaller than the ones of MDLP with an error probability of $\alpha = 0.05$.

In summary, the outcome of this experiment is quite satisfying: The discretization from HELLFIRE is very comparable to the one of MDLP in terms of classification accuracy while requiring a much number of rules.

When comparing the discretization of HELLFIRE with the one of MDLP it is interesting that both algorithms are doing feature selection: HELLFIRE discretizes only the attributes which appear in the rule set. MDLP does in a first step a discretization for all numeric attributes, but in a second step it uses the MDL criterion to reject irrelevant splits. Consequently, discretizations from HELLFIRE and MDLP can concentrate on different subspaces of the initial data space. This makes an experimental comparison of the number of partitions per attribute difficult. However, we claim that the one-dimensional evaluation of numeric attributes for MDLP may miss multi-dimensional structures due to a loss of information. A deeper analysis of this claim is beyond the scope of this thesis. However, we want to give a brief example of what kind of problem we are referring to.

Example 4.4.1 (A comparison of discretizations from HELLFIRE and MDLP)

This example shows the discretizations of HELLFIRE (H-Dis) and MDLP for an artificial, two-dimensional data set, cf. Figure 4.18a. When projecting the examples to attributes A_1 and A_2 respectively, there is no discretization which is able to partition the data flawlessly. The solution that was proposed by MDLP is shown in Figure 4.18b. It consists of two cuts, one for each attribute. While this 2×2 grid is able to separate examples from classes  and , the remaining examples cannot be discerned. The solution of HELLFIRE is not affected by this problem, cf. Figure 4.18c. It is able to find an additional split for the second attribute at $A_2 = 3.5$ after cutting the first attribute at $A_1 = 3.5$. The discretization of HELLFIRE partitions the data perfectly.

4.4.5. Interpretability Analysis

Interpretability is a very important aspect for a linguistic fuzzy rule-based classifier. Good interpretability is essential when it comes to humans exam-

Table 4.7.: Comparison in terms of number of rules and classification accuracy using RIPPER of discretizations obtained through HELLFIRE (H-Dis) and MDLP.

Data set	Number of rules		Accuracy	
	H-Dis	MDLP	H-Dis	MDLP
acd-authorship	7.54	11.67	91.40	92.50
acd-bankruptcy	2.27	2.07	81.31	81.97
acd-cyyoung8092	2.84	2.49	79.41	78.45
acd-cyyoung9302	2.39	2.18	82.05	79.83
acd-esr	1.87	1.26	80.05	80.09
acd-halloffame	2.82	6.25	91.94	92.72
acd-lawsuit	2.00	2.00	97.93	98.54
acd-votesurvey	2.61	1.04	35.55	37.18
biomed	4.42	4.61	87.58	88.20
cars	6.77	4.21	75.61	71.55
collins	14.88	15.00	93.50	97.02
ecoli	5.91	6.87	79.30	80.22
eucalyptus	9.97	11.58	56.78	59.31
glass	6.45	6.52	64.23	64.08
haberman	2.26	1.88	71.78	71.86
heart-statlog	3.83	3.98	79.82	80.69
ionosphere	5.43	6.08	88.28	89.15
iris	3.28	3.15	94.90	94.27
liver-disorders	3.78	1.32	63.25	57.58
metStatCoord.	29.62	110.65	89.38	87.52
metStatRainfall	41.56	84.99	49.35	51.86
metStatRST	9.59	3.55	34.58	27.66
metStatSunshine	9.46	10.51	37.76	35.15
metStatTemp	11.76	11.58	41.32	40.89
mfeat-factors	22.47	43.10	81.38	84.70
mfeat-fourier	23.25	55.27	71.56	68.18
mfeat-karhunen	27.04	67.34	73.40	74.18
mfeat-morpholog.	16.25	23.88	68.01	66.58
mfeat-zernike	22.37	56.81	57.89	63.62
optdigits	46.08	116.39	84.54	82.93
page-blocks	4.44	20.57	94.82	96.21
pasture-prod.	3.20	3.20	72.25	69.16
pendigits	70.90	202.23	93.49	90.87
pima diabetes	3.41	3.89	74.13	74.67
prnn-synth	3.72	3.19	82.34	83.94
schizo-	6.28	3.92	74.85	64.41
segment	16.64	37.27	93.44	91.83
sonar	3.91	4.08	72.66	71.38
squash-unstored	3.47	3.27	74.06	70.98
synthetic control	9.61	14.59	87.55	85.52
vehicle	9.19	12.14	65.85	63.83
vowel	39.28	55.78	50.56	64.12
waveform	11.56	44.71	75.25	75.85
wine	3.13	4.32	91.48	89.77
w.-breast-cancer	4.01	6.05	94.43	95.38
wins	29	14	22	23

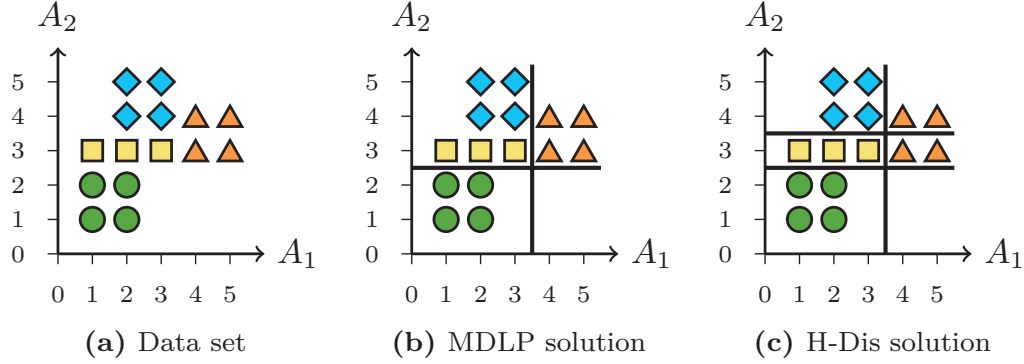


Figure 4.18.: Comparison of discretizations from MDLP and HELLFIRE (H-Dis).

ining the ruleset in an attempt to discover how the classification decisions are derived. Unfortunately, often those aspects depend on the data set and lie in the eye of the beholder. Nevertheless, there have been attempts to grasp what can be considered good interpretability: Zhou and Gan listed several criteria separated into low-level and high-level attributes that shall determine how interpretable a fuzzy rule-based classifier is [ZG08].

4.4.5.1. Low-level Criteria

This part of the interpretability analysis deals with the low-level criteria that concentrate on the fuzzy sets.

Distinguishability The aspect of distinguishability deals with the questions of how much the fuzzy sets overlap and how clear the semantic meaning of every fuzzy set is. The HELLFIRE algorithm excels in this regard. This is due to the fact that the fuzzy sets were obtained from disjunct intervals and fuzzified in such a way that there will be no fuzzy set cores overlapping. Even more appealing is the fact that there is no point covered by more than two fuzzy intervals per attribute at the same time. This can be explained by the fuzzification process that first shrinks the interval core and then stretches the support to the adjacent intervals' cores.

Completeness of the Fuzzy Partitioning This criterion asks for a complete coverage of an attribute such that every data point is covered by at least

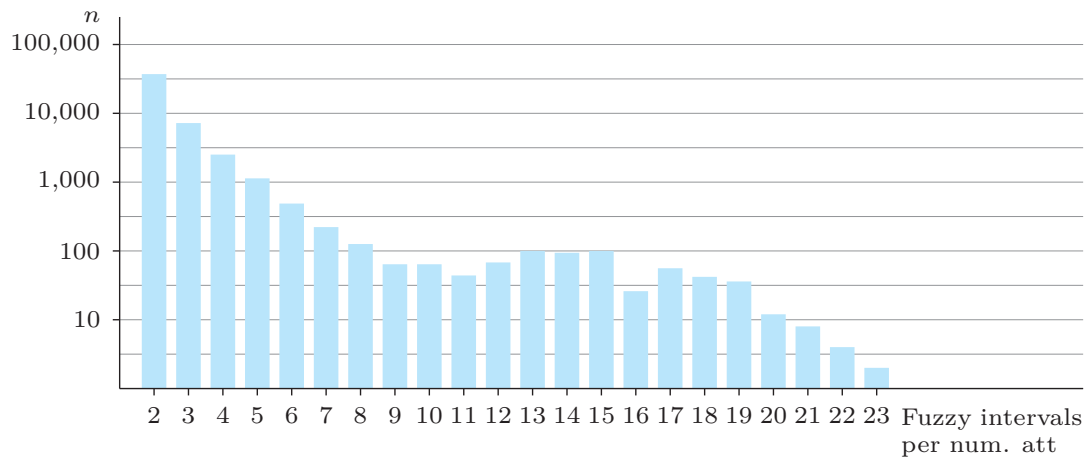


Figure 4.19.: Distribution of fuzzy sets per interval for HELLFIRE.

one fuzzy set or linguistic representation respectively. The decision tree-like learning process of HELLFIRE induces a set of disjunct intervals which covers the area between all adjacent splits. Since all lower intervals enclose the split towards the larger direction, there is no gap. Because the fuzzification process does not reduce on the coverage, it is clear that HELLFIRE covers all numeric attributes that occur in the ruleset completely. But note that not all intervals of an attribute necessarily appear in the ruleset.

Number of Fuzzy Sets Another important factor for interpretability is the number of fuzzy sets that are used per attribute. The recommendation of Zhou and Gan defined this number with 7 ± 2 according to a “rule of thumb in cognitive psychology”¹⁰ [ZG08].

The distribution of the number of fuzzy intervals per numeric attribute over all data sets displays that among a total of 49,556 numeric attributes only 846 contained more than 7 fuzzy intervals, cf. Figure 4.19. Even better, HELLFIRE uses less than 4 fuzzy intervals per numeric attribute on 35 out of the 45 data sets on average, cf. Table 4.8.

¹⁰Unfortunately, no reference is given, but it is likely that the authors refer to Miller’s work concerning this topic [Mil56].

Normalization The normalization criterion demands that every fuzzy set should cover at least one point to a degree of one. Since HELLFIRE learns fuzzy intervals, this condition clearly holds for every value $v \in [b, c] \subseteq I^F$.

Complementarity The aspect of complementarity demands that the sum of coverages for an instance is exactly one for every attribute. This is the case — if the ruleset is complete — for the interval-based ruleset RS^I since every point is covered by exactly one interval to a degree of 1. And this condition is also true for the fuzzy-intervals since there is no overlapping of cores and the gradual overlapping between two adjacent fuzzy intervals $I_i^F = [a_i, b_i, c_i, d_i]$ and $I_{i+1}^F = [c_i, d_i, c_{i+1}, d_{i+1}]$ is symmetric to their old interval boundary $s = \frac{1}{2}(c_i + d_i)$, which leads to $I_i^F(v) + I_{i+1}^F(v) = 1$ for every $v \in [c_i, d_i]$.

If the ruleset is not complete, this condition does not hold. Then the sum of coverages is smaller or equal 1.

4.4.5.2. High-level Criteria

The five criteria of high-level interpretability concentrate on individual rules but also on the whole ruleset.

Rule Base Parsimony and Simplicity The demand for small but reasonable performing models following Occam's Razor is summarized by Zhou and Gan under this point. Unfortunately, the authors do not give any recommendation or threshold for this criterion. In fact, it is questionable whether this criterion can be quantified since it is very problem dependent. But due to the fact that every rule can be read separately in the unordered ruleset of HELLFIRE, a motivated human expert should be able to grasp even the largest ruleset with nearly 40 rules. Fortunately, most of the rulesets are much smaller, cf. Table 4.8.

Readability of a Single Rule To keep the rules readable, the authors apply the same characteristic that was already applied to the number of fuzzy sets per attribute. The recommendation is that a rule should contain less than 7 ± 2 fuzzy intervals. The distribution of the number of fuzzy intervals per rule over all data sets displays that among a total of 56,256 rules only 4,500 contained more than 7 fuzzy intervals, cf. Figure 4.20. Most of these rules (nearly 4,000) were learned for metStatCoordinates, metStatRainfall, mfeat-morphological, pendigits, metStatTemp and vowel. For a vast majority of

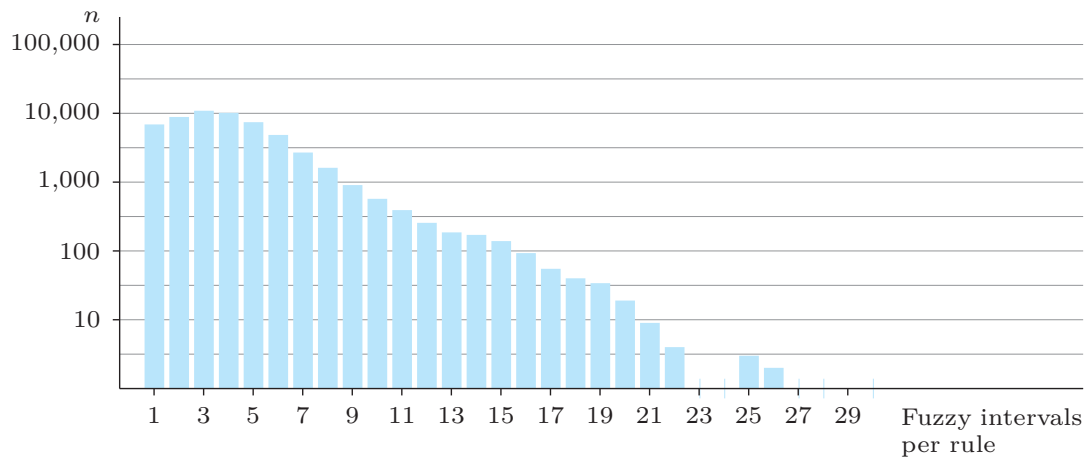


Figure 4.20.: Distribution of fuzzy sets per interval for HELLFIRE.

data sets the number of intervals per rule is rather small. Table 4.8 shows the average rule length for HELLFIRE.

Consistency A very intuitive criterion is the absence of inconsistent rules. A rule is said to be inconsistent if another rule exists with the same antecedent but with another class assignment. The decision tree-like learning of HELLFIRE induces distinct rules for every path from the leaves to the root. Consequently, every antecedent is unique. Only during pruning there might be duplicate antecedents that could occur due to the removal of splits. Since the removal of splits is only applied when there is no deterioration in terms of the m -measure, it is highly unlikely that two or more rules end up with the same antecedent. Even if that should nevertheless be the case, only one rule would be selected in the set covering step since the remaining rule would find no examples to cover.

Completeness The completeness criterion is very demanding. It asks for the full coverage with a non-arbitrarily low membership value of the whole data space by at least one rule. In fact, sometimes it is very difficult to learn reasonable rules for areas where no example or just noise has been observed in the training data. Learning a rule for such an area might need a tremendous extrapolation effort that might not be justified by data. This situation becomes even more complicated as soon as the curse of dimensionality strikes.

Unfortunately, this point was not addressed by Zhou and Gan. HELLFIRE is not able to deliver a ruleset with full coverage in general.

Transparency of Rule Structure The final criterion deals with the clarity in which the fuzzy rule-based system describes expert knowledge. According to Zhou and Gan, every fuzzy rule-base fulfills this criterion trivially due to plain class-assignment rule consequences. Thus, the classification rules of HELLFIRE can be considered to have a transparent structure.

4.4.5.3. Interpretability Discussion

In the first part of the interpretability analysis, the focus was on the fuzzy sets. Here we showed that all five criteria are reasonably fulfilled. There were only two data sets for which HELLFIRE was not able to keep the number of fuzzy intervals as small as demanded. From an overall point of view, the claim that HELLFIRE induces models with a sound low-level interpretability can be confirmed. The analysis of the high-level interpretability shows a less clear picture. While the models introduced by HELLFIRE excel in terms of rule size and consistency, they are not able to guarantee the parsimony and completeness criteria. While the ruleset parsimony and simplicity are somehow subjective characteristics that depend strongly on the data set, completeness is a much clearer demand. It comes down to question of how much interpolation the human expert is willing to accept. In fact, it is questionable whether such a model lacking data-backed evidence is desirable.

A weakness of HELLFIRE in terms of interpretability is the rule stretching procedure: If an instance is not covered by any of the rules, then the rules are stretched until they cover the instance, as it was explained in Section 4.2.8. Even though the stretched rules are readable and interpretable, this approach deteriorates the comprehensibility in such cases. Fortunately, for 38 out of the 45 data sets, the relative number of uncovered instances is smaller than 1%. And for the remainder it is smaller than 5%, cf. Table 4.9. From this we can conclude that the rule stretching is the rare exception and not the rule.

Generally, interpretability has to be considered on multiple layers. While it is typically simpler to interpret a single rule, the consideration of the ruleset is much more challenging. For a single rule the question is only whether it covers the instance and if yes to which degree and with which confidence. However, when multiple rules from different classes cover the instance, it becomes much harder to follow the fuzzy reasoning. The fuzzy inference and the aggregation

Table 4.8.: HELLFIRE model statistics.

Data set	Number of rules	Fuzzy intervals per rule	Fuzzy intervals per num. att.
acd-authorship	8.03 \pm 1.5	2.54 \pm 0.3	2.04 \pm 0.1
acd-bankruptcy	2.93 \pm 0.8	1.27 \pm 0.3	2.13 \pm 0.3
acd-cyyoung8092	4.49 \pm 1.0	1.77 \pm 0.4	2.16 \pm 0.2
acd-cyyoung9302	3.96 \pm 1.0	1.63 \pm 0.3	2.10 \pm 0.3
acd-esr	2.01 \pm 0.6	1.03 \pm 0.2	1.95 \pm 0.4
acd-halloffame	3.03 \pm 2.5	1.65 \pm 0.7	2.04 \pm 0.1
acd-lawsuit	2.49 \pm 0.5	1.43 \pm 0.1	2.00 \pm 0.0
acd-votesurvey	4.53 \pm 1.4	1.91 \pm 0.5	2.30 \pm 0.6
biomed	6.72 \pm 1.4	2.23 \pm 0.5	2.47 \pm 0.4
cars	9.97 \pm 3.1	3.34 \pm 0.6	2.87 \pm 0.3
collins	16.42 \pm 1.7	1.62 \pm 0.5	7.12 \pm 3.5
ecoli	7.44 \pm 2.0	3.04 \pm 0.5	2.22 \pm 0.2
eucalyptus	16.03 \pm 3.2	3.41 \pm 0.5	2.59 \pm 0.2
Glass	11.25 \pm 2.0	3.22 \pm 0.4	2.31 \pm 0.2
haberman	5.91 \pm 2.5	2.63 \pm 0.9	3.29 \pm 1.2
heart-statlog	6.31 \pm 1.5	1.94 \pm 0.4	2.02 \pm 0.1
ionosphere	7.65 \pm 1.7	2.06 \pm 0.4	2.04 \pm 0.1
iris	4.00 \pm 0.7	1.36 \pm 0.3	2.58 \pm 0.3
liver-disorders	8.34 \pm 2.1	2.79 \pm 0.6	2.52 \pm 0.3
metStatCoordinates	25.30 \pm 7.6	7.97 \pm 1.1	12.11 \pm 2.2
metStatRainfall	31.46 \pm 9.0	7.29 \pm 1.1	3.78 \pm 0.3
metStatRST	17.64 \pm 3.2	4.34 \pm 0.6	4.38 \pm 0.6
metStatSunshine	17.60 \pm 3.3	3.81 \pm 0.5	2.36 \pm 0.2
metStatTemp	23.90 \pm 4.9	4.74 \pm 0.7	2.93 \pm 0.4
mfeat-factors	22.13 \pm 2.6	3.74 \pm 0.2	2.00 \pm 0.0
mfeat-fourier	24.76 \pm 4.1	4.15 \pm 0.3	2.07 \pm 0.1
mfeat-karhunen	27.11 \pm 3.4	4.19 \pm 0.3	2.09 \pm 0.1
mfeat-morphological	18.47 \pm 3.7	5.32 \pm 1.1	4.38 \pm 0.7
mfeat-zernike	26.81 \pm 4.9	4.13 \pm 0.3	2.05 \pm 0.1
optdigits	24.92 \pm 3.1	4.01 \pm 0.3	2.04 \pm 0.1
page-blocks	3.43 \pm 2.5	2.34 \pm 0.7	2.36 \pm 0.4
pasture-production	3.44 \pm 0.5	1.59 \pm 0.2	2.05 \pm 0.2
pendigits	30.68 \pm 3.8	5.20 \pm 0.4	2.24 \pm 0.1
pima diabetes	5.83 \pm 2.2	2.91 \pm 0.7	2.46 \pm 0.3
prnn-synth	6.56 \pm 1.9	2.69 \pm 0.8	4.41 \pm 1.1
schizo-	10.74 \pm 2.9	2.58 \pm 0.7	3.82 \pm 0.8
segment	15.60 \pm 2.5	4.15 \pm 0.5	2.50 \pm 0.2
sonar	8.45 \pm 1.9	2.47 \pm 0.4	2.02 \pm 0.1
squash-unstored	4.39 \pm 0.6	1.72 \pm 0.2	2.13 \pm 0.2
synthetic control	14.69 \pm 2.0	3.25 \pm 0.2	2.04 \pm 0.1
vehicle	16.46 \pm 3.6	3.66 \pm 0.4	2.27 \pm 0.2
vowel	35.83 \pm 3.9	4.91 \pm 0.4	2.45 \pm 0.2
waveform	6.26 \pm 0.6	1.77 \pm 0.3	2.09 \pm 0.1
wine-weka.filters.unsupe	4.53 \pm 0.7	1.99 \pm 0.2	2.05 \pm 0.1
w.-breast-cancer	4.16 \pm 1.0	1.61 \pm 0.3	2.08 \pm 0.2

Table 4.9.: Average rate of unclassified instances for HELLFIRE.

Data set	Unclassified	Data set	Unclassified
acd-authorship	0.2064	metStatTemp	0.2409
acd-bankruptcy	0.0000	mfeat-factors	0.0250
acd-cyyoung8092	0.0313	mfeat-fourier	0.0191
acd-cyyoung9302	0.0313	mfeat-karhunen	0.0191
acd-esr	4.5000	mfeat-morpholog.	0.2059
acd-halloffame	3.7602	mfeat-zernike	0.0029
acd-lawsuit	0.8019	optdigits	0.1450
acd-votesurvey	0.2463	page-blocks	2.6570
biomed	0.4099	pasture-prod.	0.0000
cars	0.8974	pendigits	0.0211
collins	0.0826	pima diabetes	0.4748
ecoli	0.1762	prnn-synth	0.0940
eucalyptus	2.2702	schizo-	1.9104
glass	0.3166	segment	0.1019
haberman	0.8763	sonar	0.0851
heart-statlog	0.2827	squash-unstored	3.5041
ionosphere	0.7627	synthetic control	0.0392
iris	0.5882	vehicle	0.0452
liver-disorders	0.1707	vowel	0.0147
metStatCoord.	1.3554	waveform	0.4577
metStatRainfall	0.1518	wine	0.2282
metStatRST	0.6212	w.-breast-cancer	0.6690
metStatSunshine	0.1459		

of coverage degrees and rule weights is a non-trivial process that must not be underestimated. It is not unlikely that classification problems exist for which a domain expert has problems interpreting the linguistic fuzzy rule-based model. This, of course, holds true for HELLFIRE-models as well.

From an overall point of view, considering the model statistics, it seems that HELLFIRE is able to achieve interpretable models in terms of low- and high-level interpretability. However, we have to point out that interpretability of linguistic fuzzy rule models is highly subjective and problem-dependent.

4.5. Exemplary Model

In the previous section, the interpretability aspect was handled on an abstract, data set comprehensive level. Now, we will show a specific model learned by HELLFIRE. Therefore, we choose the glass data set due to its reasonable data set characteristics and model size. This data set contains different kinds of glass, described by chemical and refractive characteristics. The task is settled in the domain of crime scene investigation where the type

Table 4.10.: The rules induced by HELLFIRE for the glass data set. Note that “M x ” denotes the x^{th} smallest medium fuzzy interval. Note that the intervals’ attributes are sorted according to the order of the data set.

Antecedent	Conseq.	CF
RI is (M1 or M2 or High) and Mg is High	bwf	0.63
Mg is High and Al is High and Si is High and Fe is High	bwf	0.54
Mg is High and Al is High	bwf	0.57
Na is High and Mg is Low and K is High and Ba is Low	bwnf	0.69
RI is (Low or M1 or M2) and Al is Low and Si is Low	bwnf	0.81
RI is (M2 or High) and Na is Low and Mg is High	bwnf	0.48
Mg is Low and Al is Low	bwnf	0.53
Ba is High	h	0.85
Mg is Low and Al is High and K is High and Ba is Low	c	0.53
Na is High and Mg is Low and K is Low and Ba is Low	t	0.74
RI is Low and Na is (Low or M) and Mg is High and Al is Low	vwf	0.25
Na is M and Mg is High and Si is High and Ba is Low	vwf	0.47

of glass found at car accident scenes is in need of identification. The data set consists of 214 examples and 9 numeric attributes. The first attribute contains the refractive index, while the remainder are of chemical nature. The examples belong to six different classes: building windows float processed (bwf), building windows non float processed (bwnf), vehicle windows float processed (vwf), containers (c), tableware (t) and headlamps (h).

The rule-base for this data set consists of twelve linguistic rules, compare Table 4.10. The antecedents consist of clause conjuncts where each of these clauses is a disjunction of adjacent fuzzy intervals of distinct attributes. The fuzzy intervals refer to the fuzzy partitioning shown in Figure 4.21. All attributes except RI, Na, Ca are partitioned in only two fuzzy sets, namely “Low” and “High”. Attribute Na was partitioned into three fuzzy sets “Low”, “Medium” (M) and “High” and attribute RI into four “Low”, “Medium Low” (M1), “Medium High” (M2) and “High”. The attribute Ca is not used in the rule antecedents and, thus, it is not partitioned at all.

The data-driven fuzzy partitioning learned by HELLFIRE is far from generic, which is easy to see for the attributes which are partitioned into only two fuzzy sets: (A) The splits are at very different positions considering the respective attribute’s range of values, often far from the mean value. (B) The relative fuzzy set overlap varies tremendously.

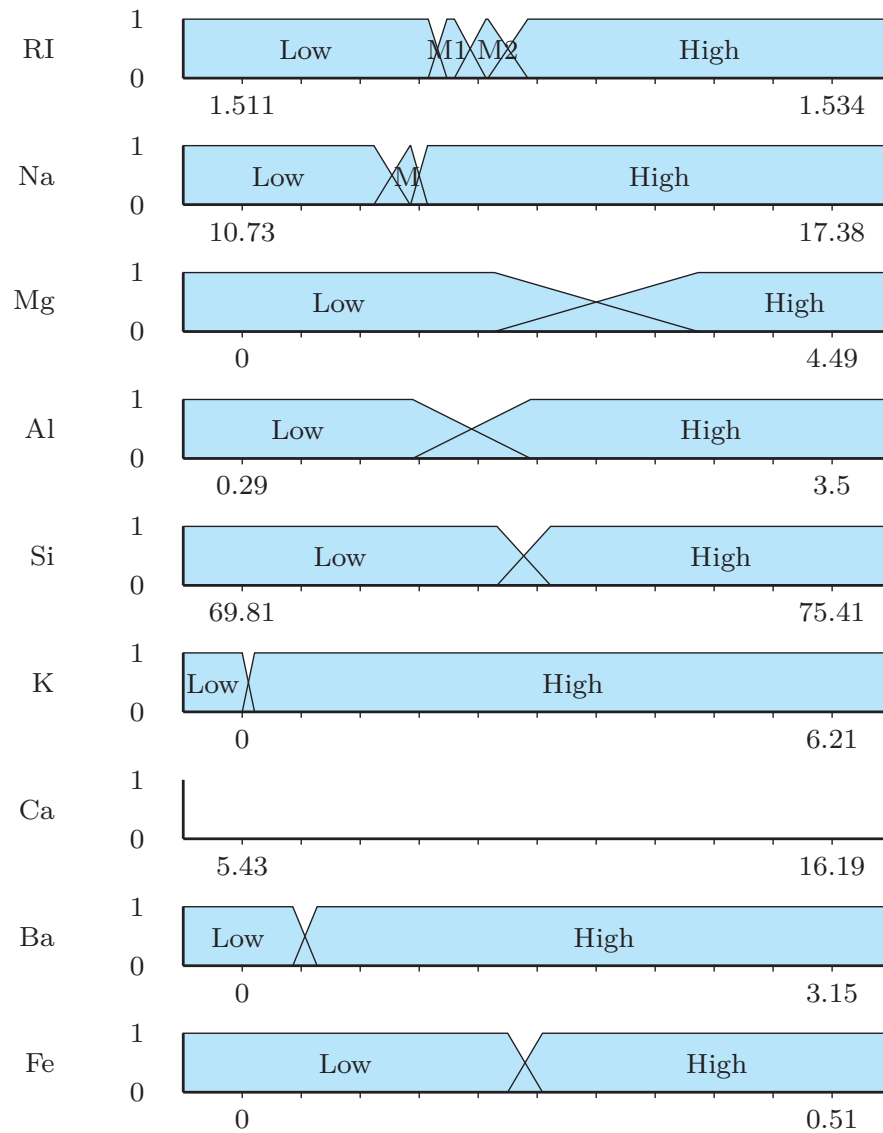


Figure 4.21.: The attributes of the glass data set and the fuzzy intervals induced by HELLFIRE. The numbers given denote the minimum and maximum attribute values.

4.6. Summary

In this paper, we proposed the novel linguistic fuzzy rule-based classifier HELLFIRE. It displays strong classification performance in terms of classification accuracy, while at the same time being interpretable and comprehen-

sible. The algorithm uses a three-step learning process with (A) inducing a decision tree-like structure, (B) transforming it into interval-based rules and (C) fuzzifying the intervals into linguistic fuzzy sets.

Future research should concentrate on the decision tree-like learning process. Even though HELLFIRE uses working heuristics, the learning process should be tuned in such a way that it becomes as sophisticated as today's state-of-the-art decision trees.

5

FR3: Learning Fuzzy Preference Structures using Fuzzy Rules

In Chapter 3 the novel fuzzy rule-based classifier FURIA was introduced, having the main purpose of solving conventional classification tasks. In this chapter we will explain how FURIA's fuzzy rules can be embedded in a more complex algorithm, that is able to learn a *fuzzy preference structure* — a potential starting point for various decision making procedures. Therefore, we will introduce the novel *FR3* algorithm which is short for *Fuzzy Round Robin RIPPER*. As the name suggests, it is based on fuzzy RIPPER rules applied in an all-pairs class decomposition scenario.

In Section 5.1 an introduction into this chapter will be given. Fuzzy preference relations, preference structures will then be presented formally in Section 5.2. After introducing the current state of the art approach in Section 5.3, the novel FR3 algorithm will then be explained in Section 5.4. Subsequently, in Section 5.5, a way of visualizing fuzzy preference structures will be shown. A larger experimental analysis of FR3 will be conducted in Section 5.6 before concluding this chapter in Section 5.7.

5.1. Introduction

One of the classical problems in decision theory addresses is to decide which of two alternatives to prefer, e.g. strawberry flavor versus chocolate flavor when it comes to comparing ice cream. For this purpose, the *weak preference relation* is able to express that one alternative is at least as preferable as the other one.

A more complex construction called *preference structure* is able to express the pairwise relationship between alternatives in a more distinguished way. It is able to express strict preference, indifference and incomparability. The strict preference states that one alternative is strictly preferred over the other. The indifference describes the preference for both alternatives and cannot be attributed to only one of them. The incomparability between two alternatives is expressed when neither strict preferences nor indifference can be found. A *fuzzy preference structure* allows to express strict preference, indifference and incomparability in a gradual way.

Hüllermeier and Brinker connected the idea of decision making to machine learning [HB08]. The authors suggested reducing a problem of polychotomous classification to a problem of decision making based on a fuzzy preference structure.

When a binary classifier predicts a score in the unit interval, this can be interpreted as a (*weak*) *fuzzy preference*. A *fuzzy preference relation* can be obtained by decomposing a polychotomous classification problem into pairwise problems, for which the binary classifiers return fuzzy preferences. This fuzzy preference relation may serve as starting point for a classification decision.

For making a classification prediction based on a fuzzy preference structure, the All-vs-All decomposition scheme can still be used for obtaining pairwise classifiers. However, these binary classifiers have to be modified for being able to express strict fuzzy preference, fuzzy indifference and fuzzy incomparability. Hüllermeier and Brinker proposed to model strict fuzzy preference as degree of evidence for one class only, while considering fuzzy indifference and fuzzy incomparability as two different types of uncertainty. The fuzzy indifference between two alternatives in decision making is seen as conflicting evidence in favor of both classes in classification. The fuzzy incomparability is seen as a lack of evidence for both classes.

Following this transformation, a polychotomous classification problem becomes to a decision making problem based on a fuzzy preference structure.

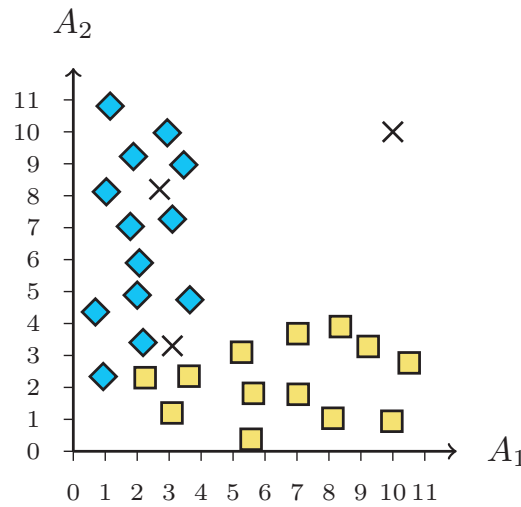


Figure 5.1.: Exemplary classification scenario: Observations from two classes \blacklozenge and \blacksquare and new query instances \times .

Moreover, this structure can also serve as foundation for other types of decision making (e.g. to abstain or to gather additional information).

A key feature of this approach is its ability to represent ignorance in a reliable way. In fact, even though many machine learning methods are able to reflect conflict in one way or the other, for example in terms of probability distributions, the same is not true for ignorance. To illustrate the meaning of conflict and ignorance in the context of classification, consider the simple scenario shown in Figure 5.1: Given observations from two classes, \blacksquare and \blacklozenge , three new instances marked by a cross need to be classified. Obviously, given the current observations, the upper left instance can quite safely be classified as \blacklozenge . The case of the lower left instance, however, involves a high level of conflict, since both classes, \blacksquare and \blacklozenge , appear plausible. The third situation is an example of ignorance: The upper right instance is located in a region of the instance space in which no observations have been made so far. Consequently, there is neither evidence in favor of class \blacksquare nor in favor of class \blacklozenge .

5.2. Preference Relations and Structures in Decision Making and Classification Learning

In the realm of decision making, when comparing two alternatives — for example classes λ_i and λ_j — the notion of *weak preference*

$$R(\lambda_i, \lambda_j) \equiv \lambda_i \succeq \lambda_j$$

means that “ λ_i is not worse than λ_j ” according to Fodor and Roubens [FR94]. Assuming that $R(\lambda_i, \lambda_j) \in [0, 1]$ this relation is called *fuzzy* or *valued* weak preference relation

$$R = \begin{bmatrix} 1 & R(\lambda_1, \lambda_2) & \cdots & R(\lambda_1, \lambda_m) \\ R(\lambda_2, \lambda_1) & 1 & \cdots & R(\lambda_2, \lambda_m) \\ \vdots & \vdots & \ddots & \vdots \\ R(\lambda_m, \lambda_1) & R(\lambda_m, \lambda_2) & \cdots & 1 \end{bmatrix}.$$

Of course, the reflexivity $R(\lambda_i, \lambda_i) = 1$ holds. To express indifference between λ_i and λ_j , it is reasonable to assume $R(\lambda_i, \lambda_j) = R(\lambda_j, \lambda_i) = 1$ since either of them are not worse than the other.

A *preference structure* $(P, \mathcal{I}, \mathcal{J})$ offers a more sophisticated option to express the relationship between λ_i and λ_j :

- strict preference $P(\lambda_i, \lambda_j)$ meaning λ_i is strictly preferred over λ_j ,
- indifference $\mathcal{I}(\lambda_i, \lambda_j)$ meaning that the alternatives are indifferent in terms of preference,
- incomparability $\mathcal{J}(\lambda_i, \lambda_j)$ denoting the impossibility to compare the alternatives.

When constraining the view on Boolean relations, we can describe the preference structure in terms of the weak preference relation [FR94]:

strict preference	$\lambda_i \succ \lambda_j \stackrel{\text{df}}{=} (\lambda_i \succeq \lambda_j) \wedge (\lambda_i \not\succeq \lambda_j)$
indifference	$\lambda_i \sim \lambda_j \stackrel{\text{df}}{=} (\lambda_i \succeq \lambda_j) \wedge (\lambda_j \succeq \lambda_i)$
incomparability	$\lambda_i \parallel \lambda_j \stackrel{\text{df}}{=} (\lambda_i \not\succeq \lambda_j) \wedge (\lambda_j \not\succeq \lambda_i)$

Following Fodor and Roubens, we allow a *valued* or *fuzzy* preference structure containing real values between 0 and 1 for $P, \mathcal{I}, \mathcal{J}$. For a more formal introduction, let (\top, \perp, \neg) be a De Morgan triple, where \top is a continuous fuzzy T-norm, where \perp is a continuous fuzzy T-conorm and where \neg is the strict negation. Together $(P, \mathcal{I}, \mathcal{J})$ form a fuzzy preference structure on the set of class labels $\mathbb{L} = \{\lambda_1, \dots, \lambda_m\}$, if the following conditions hold [Fod94]:

1. P, \mathcal{J} are irreflexive
2. \mathcal{I} is reflexive
3. P is \top -asymmetric $(\forall i, j \in \{1, \dots, m\} : \mathcal{P}(i, j) \top \mathcal{P}(j, i) = 0)$
4. \mathcal{I}, \mathcal{J} are symmetric $(\forall i, j \in \{1, \dots, m\} : \mathcal{I}(i, j) = \mathcal{I}(j, i), \mathcal{J}(i, j) = \mathcal{J}(j, i))$
5. $P \cap_{\top} \mathcal{I} = \emptyset, P \cap_{\top} \mathcal{J} = \emptyset, \mathcal{I} \cap_{\top} \mathcal{J} = \emptyset$
6. $P \cup_{\perp} P^t \cup_{\perp} \mathcal{I} \cup_{\perp} \mathcal{J} = \mathbb{L} \times \mathbb{L}$

When comparing more than two alternatives, a decomposition into all pairs keeps this binary concept applicable to multi-class problems. Consequently, this leads to the following matrices:

$$\begin{aligned}
 P &= \begin{bmatrix} 0 & P(\lambda_1, \lambda_2) & \cdots & P(\lambda_1, \lambda_m) \\ P(\lambda_2, \lambda_1) & 0 & \cdots & P(\lambda_2, \lambda_m) \\ \vdots & \vdots & \ddots & \vdots \\ P(\lambda_m, \lambda_1) & P(\lambda_m, \lambda_2) & \cdots & 0 \end{bmatrix} \\
 \mathcal{I} &= \begin{bmatrix} 1 & \mathcal{I}(\lambda_1, \lambda_2) & \cdots & \mathcal{I}(\lambda_1, \lambda_m) \\ \mathcal{I}(\lambda_1, \lambda_2) & 1 & \cdots & \mathcal{I}(\lambda_2, \lambda_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{I}(\lambda_1, \lambda_m) & \mathcal{I}(\lambda_2, \lambda_m) & \cdots & 1 \end{bmatrix} \\
 \mathcal{J} &= \begin{bmatrix} 0 & \mathcal{J}(\lambda_1, \lambda_2) & \cdots & \mathcal{J}(\lambda_1, \lambda_m) \\ \mathcal{J}(\lambda_1, \lambda_2) & 0 & \cdots & \mathcal{J}(\lambda_2, \lambda_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{J}(\lambda_1, \lambda_m) & \mathcal{J}(\lambda_2, \lambda_m) & \cdots & 0 \end{bmatrix}
 \end{aligned}$$

One standard way of introducing the fuzzy preference structure $(P, \mathcal{I}, \mathcal{J})$

based on the weak preference relation was introduced in [Fod94]:

$$\begin{aligned} P(\lambda_i, \lambda_j) &= \top (R(\lambda_i, \lambda_j), \neg(R(\lambda_j, \lambda_i))) \\ \mathcal{I}(\lambda_i, \lambda_j) &= \top (R(\lambda_i, \lambda_j), R(\lambda_j, \lambda_i)) \\ \mathcal{J}(\lambda_i, \lambda_j) &= \top (\neg(R(\lambda_i, \lambda_j)), \neg(R(\lambda_j, \lambda_i))) \end{aligned}$$

According to Hüllermeier and Brinker, a classification problem can be understood as a decision problem: Assume that two alternatives in a decision making scenario can be seen as classes λ_i and λ_j in a binary classification scenario, where only one class can be predicted. The weak preference $R(\lambda_i, \lambda_j)$ has now a special interpretation; it now describes that “ λ_i is at least as likely as λ_j ” [HB08]. In that case the notion of strict preference remains the same, but there is no indifference and no incomparability between λ_i and λ_j in classification since one class and one class only has to be predicted. The semantic equivalences would be the following ones:

$$\begin{aligned} \text{indifference } \mathcal{I} &\equiv \text{conflict } C \\ \text{incomparability } \mathcal{J} &\equiv \text{ignorance } I \end{aligned}$$

When dealing with a classification problem, the fuzzy preference structure (P, C, I) has the following meaning:

- strict preference $P(\lambda_i, \lambda_j)$: the degree of evidence supporting λ_i but not λ_j ,
- conflict $\mathcal{I}(\lambda_i, \lambda_j)$: the degree of evidence in favor of both classes that cannot be assigned more specifically,
- ignorance $\mathcal{J}(\lambda_i, \lambda_j)$: the degree of missing evidence in favor of any of the two classes.

The idea of relating a binary classification decision for a query instance \mathbf{x} to the learning of a fuzzy preference structure means that a single model $\mathcal{M}_{i,j}$ involving classes λ_i and λ_j does not return a weak preference relation R which would be similar to a reciprocal single score in the unit interval within an ordinary classification scenario. Instead, the model $\mathcal{M}_{i,j}$ returns a quadruple $(P(\lambda_i, \lambda_j), P(\lambda_j, \lambda_i), C(\lambda_i, \lambda_j), I(\lambda_i, \lambda_j))$ (as abbreviation *quadruple* will be used below synonymously). The leap from modeling binary classification problems using fuzzy preference structures to the polychotomous

case can be achieved by applying the pairwise decomposition technique: The multi-class problem will be decomposed into $m(m-1)/2$ pairwise problems, cf. Section 2.2.2.

The set of all these quadruples form the starting point for various decision making procedures, e.g. polychotomous classification decisions or abstention decisions [HB08].

5.3. State-of-the-art: Learning Valued Preference Structures for Solving Classification Problems

The approach of Hüllermeier and Brinker to learning a fuzzy preference structure for classification was to decompose a polychotomous classification algorithm into pairwise problems and then to solve the pairwise problems through an ensemble of Perceptrons [HB08, Ros58]. The variation in the ensemble was achieved through permutations in the training data. The score distribution of this ensemble was used to induce the fuzzy preference structure. The theoretical foundation for this procedure is that the ensemble $\mathbb{M}_{i,j}$ is a finite sample of the version space \mathfrak{M} of the Perceptron for the given problem:

$$\mathbb{M}_{i,j} = \{\mathcal{M}_{i,j}^{(1)}, \dots, \mathcal{M}_{i,j}^{(K)}\} \subseteq \mathfrak{M}$$

This ensemble returns a finite sample of the true score distribution that will be used as the starting point for creating the fuzzy preference structure. It is assumed that an ensemble member $\mathcal{M}_{i,j}^{(k)}$ delivers a score $s_{i,j}^{(k)} \in [0, 1]$ that can be interpreted as a weak fuzzy preference. A value near to one corresponds to the support for λ_i , while a value near to zero denotes the support for λ_j . Given those scores, the authors proposed the weak preference

$$R(\lambda_i, \lambda_j) \stackrel{\text{df}}{=} \min_{k=1, \dots, K} s_{i,j}^{(k)}.$$

This definition is quite critical when the data contains noise or outliers. To diminish such effects the authors proposed replacing the minimum operator by the α -quantile of the distribution $s_{i,j}^{(k)}$. It is assumed that the models in $\mathfrak{M}_{i,j}$ are reciprocal and, thus, $s_{i,j}^{(k)} = 1 - s_{j,i}^{(k)}$ holds. Consequently, the weak preference $R(\lambda_j, \lambda_i)$ is given by the $(1 - \alpha)$ -quantile of the distribution $s_{i,j}^{(k)}$. Given the weak preferences $R(\lambda_i, \lambda_j)$ and $R(\lambda_j, \lambda_i)$, the fuzzy preference

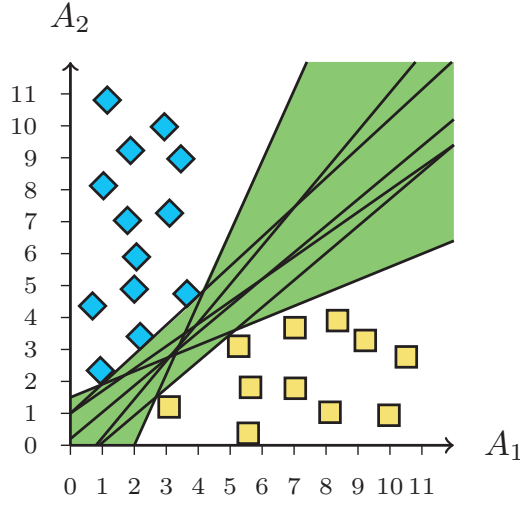


Figure 5.2.: Perceptron hyperplanes that were sampled from the version space and classify the data correctly. The shaded area is the “region of ignorance” [HB08].

structure is established in the following way:

$$\begin{aligned}
 P(\lambda_i, \lambda_j) &= R(\lambda_i, \lambda_j) (1 - R(\lambda_j, \lambda_i)) \\
 C(\lambda_i, \lambda_j) &= 2 R(\lambda_i, \lambda_j) R(\lambda_j, \lambda_i) \\
 I(\lambda_i, \lambda_j) &= 1 - (R(\lambda_i, \lambda_j) + R(\lambda_j, \lambda_i))
 \end{aligned}$$

A graphical illustration of this approach is given in Figure 5.2. It shows two classes that are linearly separable. Since a wealth of Perceptrons exists that solve this problem flawlessly, creating an ensemble of those Perceptrons corresponds to sampling the version space. A query instance lying in the shaded area in the illustration exhibits some degree of uncertainty, due to the fact that it is not fully clear to which class it belongs.

5.4. Fuzzy Round Robin RIPPER

Hüllermeier and Brinker mentioned that a rule-based system would be ideally suited for obtaining a fuzzy preference structure [HB08]. The main reason for this suitability is that, in contrast to standard discriminative classification methods (such as linear discriminant functions), rule-based models are able to represent conflict and, more importantly, ignorance in a natural way: A

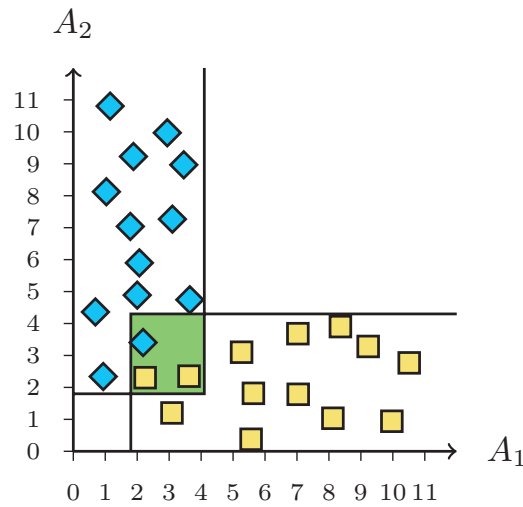


Figure 5.3.: Exemplary classification scenario: Regions of conflict (green area) and ignorance (area not covered by any rule) in case of a rule-based model (rules indicated as rectangular lines).

situation of conflict occurs if an instance \mathbf{x} is simultaneously covered by two (or more) conflicting rules, while a situation of ignorance occurs if it is not covered by any rule; see Figure 5.3.

In the present section we will describe how this idea can be realized: We will present a novel rule-based algorithm that is able to learn fuzzy preference structures: The Fuzzy Round Robin RIPPER (FR3) algorithm.

Since we already found that fuzzy rule learners overcome the unnatural conventional step between full coverage and zero coverage, we prefer this characteristic also for the learning of fuzzy preference structures: The gradual rule coverage will allow a plausible representation of preference, conflict and ignorance.

5.4.1. Pairwise Decomposition

The quadruple $(P(\lambda_i, \lambda_j), P(\lambda_j, \lambda_i), C(\lambda_i, \lambda_j), I(\lambda_i, \lambda_j))$ considers classes λ_i and λ_j only. This makes it inapplicable to polychotomous classification problems without further effort. We follow the scheme of Hüllermeier and Brinker who suggested applying the All-vs-All decomposition to create pairwise problems [HB08], cf. Section 2.2.2.

Thus, we will decompose the m -class classification problem into $m(m-1)/2$

binary problems involving classes λ_i, λ_j with $1 \leq i < j \leq m$. We will learn the quadruple using the classifier $\mathcal{M}_{i,j}$, as we will explain below.

5.4.2. Fuzzy Rules for Learning Fuzzy Preference Structures

The fundamental idea behind learning the quadruple for two classes λ_i and λ_j using classification rules was already mentioned. We will now go into detail on how this can be realized in practice.

5.4.2.1. Starting Point: FURIA

The learning of the quadruple with classification rules will be based on the FURIA algorithm. Consequently, several adjustments have to be made. A direct use of the FURIA algorithm as a binary classifier $\mathcal{M}_{i,j}$ is not viable for the following reasons: (A) The degree of ignorance is expressed through the uncovered area. Consequently, the rule stretching technique cannot be used. (B) FURIA calculates a score $s \in [0, |RS|]$ to obtain a classification decision which cannot be easily transformed into the quadruple. (C) The rule learning procedure of FURIA builds well generalizing rules which extrapolate extensively. This is problematic when decomposing the problem in a pairwise manner.

Obviously, when dealing with a classification scenario involving classes λ_i and λ_j , FURIA itself has no use for constructing a preference structure for the given reasons. Thus, we will just use FURIA's rule learning strategy to learn two sets of fuzzy rules $RS_{i,j}$ and $RS_{j,i}$, one for each class. Of course, this will sideline (A) and (B), but it makes a novel way of calculating the class scores necessary. We will go into detail below.

But first of all, we will deal with (C): We will limit the extrapolation by restricting the rules in a fuzzy way.

Note that setting up RIPPER without any modifications would also lead to a very similar set of failures: (A) RIPPER learns only rules for the minority class and classifies the rest as default class. (B) There is no score calculation in RIPPER since there is just the ordered list of rules. (C) RIPPER rules are also very extrapolative.

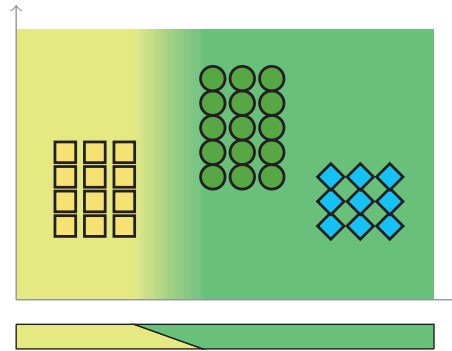


Figure 5.4.: The unbound rules in a pairwise model \mathcal{M} ■ ● cover class ◆ accidentally.

5.4.2.2. Limiting the Extrapolation

A key characteristic of fuzzy preference structures is that both indifference and ignorance are represented in a reasonable way. Especially for the notion of ignorance, a classifier has to be careful with its prediction. For a rule based-classifier this means that areas not supported by proper evidence shall not be covered by any rule. This demand is typically not supported by a rule learner due to the learning scheme that only tries to separate the classes from each other. Covering sparse areas is a generalization bias which is fatal here. This topic becomes even more severe when considering the pairwise decomposition scheme. This is because a ruleset $RS_{i,j}$ for class λ_i tries to separate examples from this class from the ones of class λ_j . Since examples from another class λ_k are not taken into consideration at this point, there is no guarantee that $RS_{i,j}$ does not cover λ_k -examples accidentally, see Figure 5.4 for an illustrative example.

The consequence would be that instances which actually belong to λ_k gather evidence for λ_i from $RS_{i,j}$. The main risk of such malignant coverage is due to the unbound support from intervals of the form $A_\ell \in [a, b, \infty, \infty)$ and $A_\ell \in (-\infty, -\infty, c, d]$ that might cover huge areas of the data space where no evidence in favor of their class was found. As this is not in agreement with the “cautious” extrapolation strategy, these intervals are finally closed: If $b_\ell = -\infty \in A_\ell$ is considered, this core bound is set to

$$b_\ell = \min\{x_\ell \mid (\mathbf{x}, \lambda) \in D^{(i)}, \mathbf{x} = (x_1, \dots, x_k), \mu(\mathbf{x}) > 0\} .$$

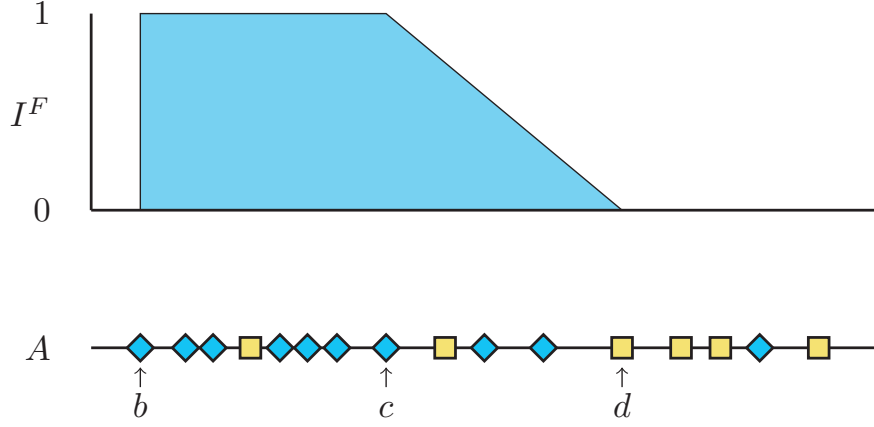


Figure 5.5.: In case a fuzzy interval $[a = -\infty, b = -\infty, c, d]$ is open to one side (here the left one), the core bound b is determined by the last positive example. (Note that $a = -\infty$ is ignored for a clearer visualization.)

In this way, the core of the rule is restricted to the region in which positive examples have indeed been observed, cf. Figure 5.5.

Moreover, the support bound a_ℓ is set to the minimal value in \mathbb{D}_ℓ , the domain of attribute A_ℓ . In practice we have to estimate $\min(\mathbb{D}_\ell)$ with the known training data D . Consequently, to avoid any misses, we set a_ℓ beyond $\min(D_\ell)$ in a conservative fashion, as a rule of thumb by 50% of the width of D_ℓ , which is $\max(D_\ell) - \min(D_\ell)$. Applying this procedure lets the interval support decrease as a linear function of the distance from the interval core, cf. Figure 5.6. Analogous modifications are made in the case where $c = \infty$. See Figure 5.7 showing the previous example (but with bound rules): The instances from the third class are still mistakenly covered, but now the mistake is less severe.

5.4.2.3. Output of the Pairwise Models

Suppose that fuzzy rules r_1^i, \dots, r_k^i and r_1^j, \dots, r_ℓ^j have been learned for classes λ_i and λ_j respectively. For a new query instance \mathbf{x} , the supports of these classes are then given, respectively, by

$$s_i \stackrel{\text{df}}{=} \bigwedge_{s=1, \dots, k} (\mu_{r_s^i}(\mathbf{x}) \cdot \text{CF}(r_s^i))$$

$$s_j \stackrel{\text{df}}{=} \bigwedge_{t=1, \dots, \ell} (\mu_{r_t^j}(\mathbf{x}) \cdot \text{CF}(r_t^j)) \quad ,$$

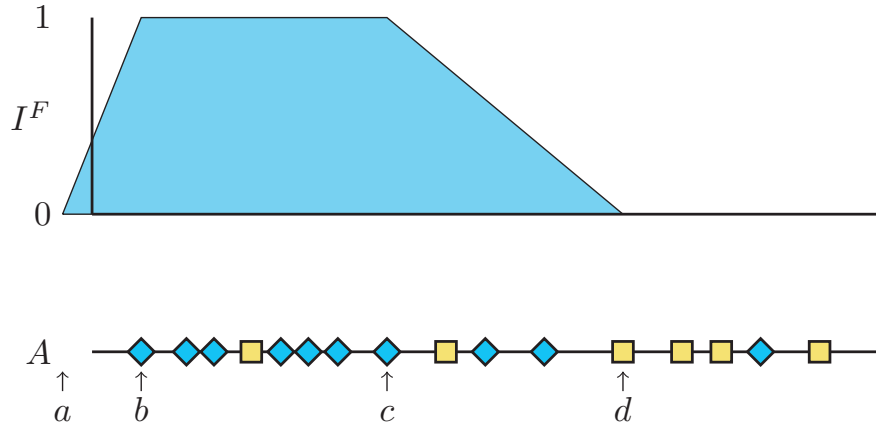


Figure 5.6.: The final fuzzy interval $[a, b, c, d]$ with a support reaching beyond the known data space. Note the position of a in this illustration is not true to scale according to the 50% rule of thumb.

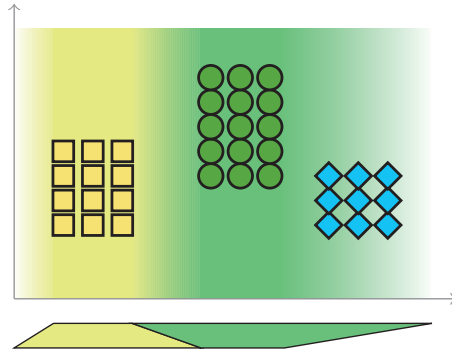


Figure 5.7.: The bound and fuzzified rules in a pairwise model $\mathcal{M}_{\square, \bullet}$ cover class \blacklozenge accidentally, but only gradually.

where

$$\text{CF}(r) = \text{CF}(\langle r_A \mid \lambda_j \rangle) \stackrel{\text{df}}{=} \frac{\sum_{(\mathbf{x}, \lambda_j) \in D^{(j)}} \mu_r(\mathbf{x})}{\sum_{(\mathbf{x}, \lambda) \in D} \mu_r(\mathbf{x})}$$

is a measure of the confidence or validity of a rule. For FR3 we choose the max-operator as T-conorm \perp .

From these two support degrees, we derive the quadruple as the output of

the fuzzy rule-based model $\mathcal{M}_{i,j}$:

$$\begin{aligned} P(\lambda_i, \lambda_j) &= s_i - \min\{s_i, s_j\} \\ P(\lambda_j, \lambda_i) &= s_j - \min\{s_i, s_j\} \\ C(\lambda_i, \lambda_j) &= \min\{s_i, s_j\} \\ I(\lambda_i, \lambda_j) &= 1 - \max\{s_i, s_j\} \end{aligned} \quad (5.1)$$

$C(\lambda_i, \lambda_j)$ is the degree of *conflict*, namely the degree to which both classes are supported. Likewise, $I(\lambda_i, \lambda_j)$ is the degree of *ignorance*, namely the degree to which none of the classes is supported. Finally, $P(\lambda_i, \lambda_j)$ and $P(\lambda_j, \lambda_i)$ denote, respectively, the strict preference for λ_i and λ_j . Note that at least one of these two degrees is zero and that $P(\lambda_i, \lambda_j) + P(\lambda_j, \lambda_i) + C(\lambda_i, \lambda_j) + I(\lambda_i, \lambda_j) \equiv 1$. In passing, we also remark that (5.1) is actually a standard decomposition scheme used in fuzzy preference modeling [FR94].

5.4.3. Making Classification Decisions Based on Fuzzy Preference Structures

To obtain a classification decision from the quadruples an aggregation has to be made. Let \mathbf{x} be a query instance submitted to each model. As explained in Section 5.4.2.3, the output of model $\mathcal{M}_{i,j}$ is a quadruple $\mathcal{M}_{i,j}(\mathbf{x}) = (P(\lambda_i, \lambda_j), P(\lambda_j, \lambda_i), C(\lambda_i, \lambda_j), I(\lambda_i, \lambda_j))$.

These relations provide the basis for sophisticated classification and decision making policies. In the standard scenario where a single prediction is sought, the following classification practice could be used:

$$\lambda^* = \arg \max_{\lambda_i \in \mathbb{L}} \sum_{1 \leq i \neq j \leq m} P(\lambda_i, \lambda_j) + \frac{1}{2} \cdot C(\lambda_i, \lambda_j) + \frac{N_i}{N_i + N_j} \cdot I(\lambda_i, \lambda_j) \quad , \quad (5.2)$$

where N_i is the number of examples from class λ_i in the training data (and hence an unbiased estimate of the class probability). This decision rule, which proved to perform well in practice (cf. Section 5.6), evaluates each candidate label in terms of the sum of strict preferences over all other labels, distributes the corresponding degrees of conflict in a uniform way and the degrees of ignorance in proportion to the size of the classes (in other words, prior probabilities are used in the case of no further information).

Going beyond the conventional classification setting, a preference structure (P, C, I) can be especially useful in generalized settings in which, for example,

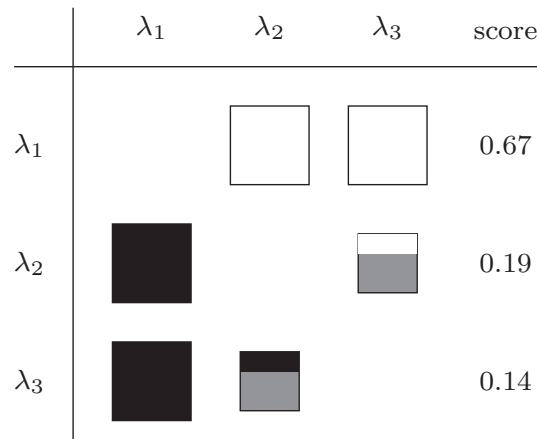


Figure 5.8.: Graphical illustration of a preference structure predicted by FR3 for a query instance on the iris data. The size of a box is proportional to the degree of non-ignorance (1 minus ignorance). The size of the white (black) area is proportional to the degree of preference in favor of the row-class (column-class). The gray area shows the corresponding degree of conflict. The rightmost column shows the final score (5.2) for every class.

more than one class can be predicted in cases of conflict, or a classification decision can be refused in cases of ignorance (cf. Section 5.6.5).

5.5. Visualization of Fuzzy Preference Structures

An FR3 prediction reduces complexity by providing information on two levels of abstraction: On the “relational level”, the preference structure gives a rough picture of the situation, including uncertainties and potential conflicts. Information on this level becomes especially comprehensible when being presented in a graphical form, as shown in Figure 5.8. If the need arises, each entry in the corresponding relations can then be “explained” by an underlying pairwise model, which resembles the second level of abstraction. As an advantage, note that each pairwise model itself will typically be much simpler than a single polychotomous model, as it refers to only two instead of all classes simultaneously.

5.6. Experiments

To analyze the performance of our FR3 approach, we conduct several experimental studies. As a starting point, we used the RIPPER implementation of WEKA (“JRip”), cf. Section 2.3.3, both for re-implementing Fürnkranz’s R3 and our FR3.

5.6.1. Classification Performance Analysis

In a first study, we compared RIPPER, R3 and FR3 with respect to classification accuracy. For RIPPER we used the WEKA default settings according to Section 2.5.3.1. R3 was used with the weighted voting variant, i.e., the vote of a pairwise classifiers is weighted in terms of rule purity; Fürnkranz pointed out that this method outperformed binary (0/1) voting and Laplace weighted voting [Für03]. Apart from that, the RIPPER-specific settings in R3 were set to default values.

We also included the C4.5 decision tree learner, cf. Section 2.5.3.4. Moreover, we added two fuzzy rule learners from the KEEL suite [AFSG⁺09]: The fuzzy grid-based CHI classifier and the genetic fuzzy rule learner SLAVE, cf. Section 2.5.3.2 and 2.5.3.3.

The testbed for this comparison were the 30 multi-class data sets, cf. 2.1. We did not use the binary data sets since FR3 is a multi-class classifier only. The experimental settings followed the setup proposed in Section 2.5.5. Table 5.1 summarizes the results in terms of mean classification accuracies.¹¹

The overall picture conveyed by the results is clearly in favor of FR3, which outperforms the other methods on most data sets. In order to evaluate the performances, we conduct the Friedman Test, cf. Table 5.1 for the classifier ranks. The corrected Friedman statistic for large N and k according to (2.5) is 49.98, while the critical value for the significance level $\alpha = 0.01$ is only 4.70. Thus, the null-hypothesis can quite safely be rejected, which means that there are significant differences in the classifiers’ performance.

Given the result of the Friedman Test, we conduct the Bonferroni-Dunn Test as a post-hoc test to compare the competitors to the control classifier HELLFIRE [Dun61]. The critical distance according to (2.7) between two classifier ranks is $CD_\alpha = 1.12$. The results of this test are summarized in Figure 5.9: FR3 is better than the rest in terms of accuracy — except R3

¹¹The classifier FR3c, which also appears in the table, will be analyzed in Section 5.6.3.

Table 5.1.: Average classification accuracy and ranks for FR3 and its competitors.

Data set	FR3	R3	RIPPER	C4.5	CHI	SLAVE	FR3c
acd-authorship	95.17(1)	94.37(2)	93.05(4)	93.50(3)	71.60(6)	91.87(5)	94.66
acd-halloffame	93.13(2)	93.22(1)	92.87(4)	92.87(3)	91.73(6)	92.68(5)	93.06
acd-votesurvey	36.06(3)	35.58(4)	34.40(5)	38.75(2)	42.88(1)	29.51(6)	36.63
cars	81.48(2)	79.52(3)	75.93(4)	82.15(1)	67.66(6)	70.68(5)	80.93
collins	94.53(2)	92.87(3)	92.67(4)	96.10(1)	43.37(6)	50.87(5)	93.12
ecoli	82.91(1)	82.46(2)	80.57(5)	81.35(3)	78.26(6)	81.03(4)	82.04
eucalyptus	64.25(1)	63.72(2)	58.69(4)	59.98(3)	52.43(6)	58.16(5)	63.88
glass	72.98(1)	69.61(2)	63.18(4)	66.69(3)	61.70(6)	61.83(5)	71.13
iris	94.78(2)	94.25(3)	93.45(5)	94.25(3)	92.55(6)	94.92(1)	94.08
metStatCoord.	93.30(1)	92.85(3)	92.04(4)	92.87(2)	32.64(6)	58.77(5)	92.97
metStatRainfall	69.68(1)	68.13(2)	60.66(3)	59.47(4)	22.71(6)	29.35(5)	68.86
metStatRST	43.22(2)	44.13(1)	36.08(5)	38.60(4)	20.17(6)	42.02(3)	43.30
metStatSunshine	52.94(1)	51.17(2)	44.48(4)	46.78(3)	32.18(5)	28.83(6)	52.25
metStatTemp	57.38(1)	56.10(2)	47.45(4)	53.18(3)	24.07(5)	22.10(6)	57.23
mfeat-factors	93.35(1)	92.64(2)	87.05(5)	87.96(4)	89.19(3)	86.83(6)	93.06
mfeat-fourier	80.48(1)	79.26(2)	71.37(5)	74.42(3)	69.29(6)	73.49(4)	80.16
mfeat-karhunen	91.51(1)	89.70(2)	79.13(5)	80.20(4)	83.30(3)	78.37(6)	91.13
mfeat-morpholog.	72.31(1)	72.25(2)	70.74(4)	71.60(3)	41.63(6)	67.08(5)	72.09
mfeat-zernike	77.18(1)	76.13(2)	67.58(6)	69.11(4)	73.44(3)	68.26(5)	77.05
optdigits	96.22(1)	95.55(2)	89.68(4)	89.51(5)	45.90(6)	93.45(3)	96.01
page-blocks	97.04(2)	97.13(1)	96.79(4)	96.89(3)	92.41(6)	93.58(5)	96.92
pasture-prod.	71.28(2)	67.91(4)	68.46(3)	73.67(1)	44.53(6)	53.63(5)	67.47
pendigits	98.07(1)	97.48(3)	95.54(5)	95.92(4)	97.64(2)	87.26(6)	97.67
segment	96.97(1)	96.14(2)	94.53(4)	95.95(3)	85.32(6)	88.87(5)	96.52
squash-unstored	75.52(2)	74.92(3)	71.74(4)	76.08(1)	60.90(6)	65.56(5)	75.02
synthetic control	92.24(1)	90.78(2)	82.85(6)	90.00(3)	86.59(5)	89.23(4)	90.89
vehicle	72.78(1)	71.66(2)	67.80(4)	71.38(3)	61.05(6)	64.08(5)	72.63
vowel	84.03(1)	77.85(2)	64.71(4)	75.60(3)	47.93(6)	63.84(5)	81.58
waveform	79.79(2)	80.58(1)	78.72(3)	75.05(5)	74.98(6)	75.34(4)	78.56
wine	92.70(2)	92.63(3)	90.02(6)	91.22(5)	93.08(1)	92.46(4)	91.84
average rank	1.40	2.23	4.37	3.07	5.13	4.77	

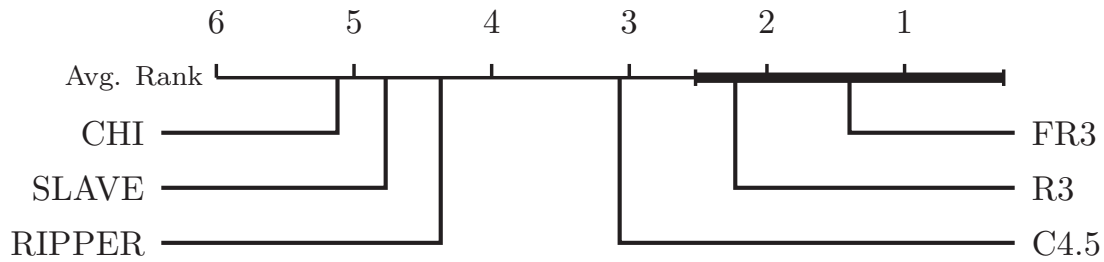


Figure 5.9.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. FR3 is significantly better than the rest in terms of classification accuracy — except R3. Significance level $\alpha = 0.1$

— at the significance level $\alpha = 0.1$. Although this test was unable to reject the null-hypothesis of equal classifier performance between FR3 and R3, the direct comparison is very clear: FR3 outperforms R3 with 26 wins and only 4 losses.

5.6.2. Ranking Performance Analysis

In this study, we compare FR3 to the algorithms from the analysis before. The settings and data sets remain the same, while the focus is now on the area under the curve, AUC. Table 5.2 summarizes the results in terms of mean AUC.¹¹

The overall picture conveyed by the results is clearly in favor of FR3, which outperforms the other methods on most data sets. In order to evaluate the performances, we conduct the Friedman Test, cf. Table 5.2 for the classifier ranks. The corrected Friedman statistic for large N and k according to (2.5) is 41.05, while the critical value for the significance level $\alpha = 0.01$ is only 4.70. Thus, the null-hypothesis can quite safely be rejected, which means that there are significant differences in the classifiers' performance.

Given the result of the Friedman Test, we conduct the Bonferroni-Dunn Test as a post-hoc test to compare the competitors to the control classifier HELLFIRE [Dun61]. The critical distance according to (2.7) between two classifier ranks is $CD_\alpha = 1.12$. The results of this test are summarized in Figure 5.10: FR3 is better than the rest in terms of AUC — except R3 — at the significance level $\alpha = 0.1$. Although this test was unable to reject the null-hypothesis of equal classifier performance between FR3 and R3, the

Table 5.2.: Average AUC and ranks for FR3 and its competitors.

Data set	FR3	R3	RIPPER	C4.5	CHI	SLAVE	FR3c
acd-authorship	0.99(1)	0.99(2)	0.96(4)	0.96(5)	0.90(6)	0.98(3)	0.99
acd-halloffame	0.89(2)	0.90(1)	0.79(6)	0.81(5)	0.87(4)	0.89(3)	0.88
acd-votesurvey	0.54(3)	0.53(4)	0.50(5)	0.54(2)	0.55(1)	0.49(6)	0.55
cars	0.92(1)	0.91(2)	0.82(5)	0.90(3)	0.86(4)	0.81(6)	0.91
collins	1.00(1)	0.99(2)	0.97(4)	0.98(3)	0.88(5)	0.83(6)	0.99
ecoli	0.94(1)	0.94(2)	0.89(5)	0.89(6)	0.94(3)	0.91(4)	0.94
eucalyptus	0.89(1)	0.88(2)	0.82(4)	0.82(3)	0.79(5)	0.77(6)	0.89
glass	0.88(1)	0.86(2)	0.76(6)	0.79(4)	0.77(5)	0.79(3)	0.87
iris	0.99(2)	0.98(4)	0.96(6)	0.96(5)	1.00(1)	0.98(3)	0.98
metStatCoord.	1.00(1)	1.00(2)	0.98(3)	0.98(4)	0.78(6)	0.82(5)	1.00
metStatRainfall	0.95(1)	0.94(2)	0.85(3)	0.81(4)	0.76(5)	0.66(6)	0.95
metStatRST	0.82(1)	0.82(2)	0.70(4)	0.71(3)	0.59(5)	0.58(6)	0.82
metStatSunshine	0.88(1)	0.87(2)	0.75(5)	0.74(6)	0.80(3)	0.75(4)	0.88
metStatTemp	0.88(1)	0.88(2)	0.77(4)	0.77(3)	0.66(6)	0.68(5)	0.88
mfeat-factors	0.99(1)	0.99(2)	0.95(5)	0.94(6)	0.98(3)	0.97(4)	0.99
mfeat-fourier	0.97(1)	0.97(2)	0.92(5)	0.88(6)	0.92(3)	0.92(4)	0.97
mfeat-karhunen	0.99(1)	0.98(2)	0.92(5)	0.90(6)	0.98(3)	0.95(4)	0.99
mfeat-morpholog.	0.96(1)	0.96(2)	0.94(3)	0.92(5)	0.93(4)	0.89(6)	0.96
mfeat-zernike	0.96(1)	0.96(2)	0.90(5)	0.87(6)	0.95(3)	0.90(4)	0.96
optdigits	1.00(1)	1.00(2)	0.96(4)	0.95(5)	0.82(6)	0.96(3)	1.00
page-blocks	0.98(1)	0.98(2)	0.93(4)	0.93(3)	0.82(6)	0.83(5)	0.98
pasture-prod.	0.86(1)	0.85(2)	0.79(4)	0.83(3)	0.64(6)	0.71(5)	0.85
pendigits	1.00(2)	1.00(3)	0.98(5)	0.98(6)	1.00(1)	0.99(4)	1.00
segment	1.00(1)	0.99(2)	0.98(3)	0.98(4)	0.97(5)	0.96(6)	0.99
squash-unstored	0.83(1)	0.82(2)	0.77(4)	0.81(3)	0.71(6)	0.76(5)	0.82
synthetic control	0.99(1)	0.99(4)	0.93(6)	0.95(5)	0.99(2)	0.99(3)	0.99
vehicle	0.90(1)	0.90(2)	0.85(3)	0.85(5)	0.85(4)	0.83(6)	0.90
vowel	0.98(1)	0.97(2)	0.88(6)	0.91(4)	0.91(3)	0.89(5)	0.97
waveform	0.94(2)	0.94(1)	0.88(5)	0.83(6)	0.91(4)	0.91(3)	0.93
wine	0.98(2)	0.97(3)	0.93(6)	0.93(5)	0.98(1)	0.97(4)	0.98
average rank	1.23	2.20	4.57	4.47	3.97	4.57	

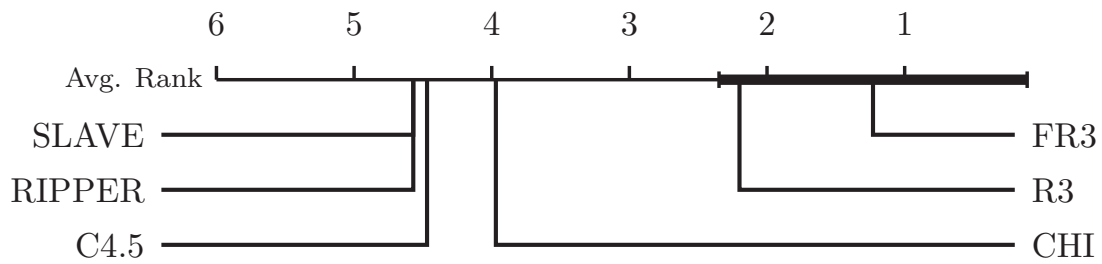


Figure 5.10.: Bonferroni-Dunn Test visualization according to Demšar [Dem06]. FR3 is significantly better than the rest in terms of AUC — except R3. Significance level $\alpha = 0.1$

direct comparison is very clear: FR3 outperforms R3 with 28 wins and only 2 losses.

5.6.3. Fuzzification Analysis

The previous results show that FR3 is a significant improvement in comparison to RIPPER and R3. To explain this improvement, we conjecture that the scores produced by fuzzy rules are superior to those produced by conventional rules, which in turn is beneficial for the voting scheme that is used by the round robin learner to determine a prediction.

To examine whether rule fuzzification is indeed the main factor, or whether the improvements should perhaps be attributed to other modifications, we conduct some additional experiments with a conventional (*crisp*) variant of FR3, included in Table 5.1 and in Table 5.2 under the name FR3c. To optimize an interval as originally produced by RIPPER, this variant conducts a search process quite similar to the search for an optimal fuzzy interval (cf. Section 3.2.3). Instead of a trapezoid, however, it is again only allowed to use intervals, i.e. it simply attempts to optimize the original decision boundary in terms of the rule's purity. When comparing FR3c to R3, it achieves 21 wins in terms of accuracy and 24 in terms of AUC, which is less than the 26 and 28 wins achieved by FR3. The importance of the fuzzy rules become even more obvious when comparing FR3c to FR3 itself. Here, the latter has a higher classification rate for all except two data sets and a better AUC for all but one data set.

From this analysis we conclude that the use of fuzzy rules is indeed essential for the superb performance of FR3.

5.6.4. Model Complexity Analysis

Since FR3 disables the pruning step in IREP, it learns more specialized rules. Therefore, it is likely to produce models that are more complex — in terms of the number of rules and their lengths — than those produced by R3.

Indeed, FR3 produces more specific rules than R3 for all but one data set and also the average rule length (number of attributes in the antecedent part) of FR3 (1.78) is slightly larger than the average length for R3 (1.53); see Table 5.3 for detailed statistics. Likewise, FR3 uses more rules for all but one data set and again the average number of rules is slightly higher for FR3 (3.00) than for R3 (2.31).

As demonstrated, the performance gain comes at the cost of slightly more complex models, even though the average differences (less than one additional rule and about 0.3 additional attributes per rule) are admissible.

5.6.5. Analysis of Conflict and Ignorance as Measures of Uncertainty

The ability to represent uncertainty involved in a classification decision — in terms of measures of conflict and ignorance — is arguably one of the main advantages of FR3. To test whether FR3 does indeed provide a basis for implementing classifier that are more “reliable”, we conduct another series of experiments in a setting of classification with reject option. Roughly speaking, the idea is that, if γ is a reliable index of classification uncertainty, then the value of γ should correlate with the probability of making a correct decision. Or, stated differently, when abstaining from the classification of all instances, the γ -value of which exceeds a threshold t , classification accuracy should improve on the remaining instances. The dependency between the threshold t and classification accuracy is typically depicted in the form of so-called *accuracy-rejection* curves.

In our experiments, we test two very simple uncertainty indexes (needless to say, various other indexes are conceivable) directly related to the two types of uncertainty reflected by FR3: γ_c is the degree of conflict between the top-class as suggested by FR3 (in terms of the score (5.2)) and the second-best class. Likewise, γ_i is the degree of ignorance between these two classes. Again, each data set is randomly split, in proportion 2:1, for training and testing. This is repeated 100 times and each instance (occurring in potentially many of the 100 test sets) is associated with its average γ -index.

Table 5.3.: FR3 model statistics. The number of rules per rule set and the number of attributes per rule.

Data set	rules per rule set		conditions per rule	
	FR3	R3	FR3	R3
analcatauthorship	2.51	2.20	1.76	1.54
analcatauthority	4.7	2.53	2.31	1.62
analcatauthority	1.47	1.16	1.30	1.12
cars	4.28	3.14	1.86	1.64
collins	1.0	1.2	1.0	1.0
ecoli	1.45	1.20	1.34	1.12
eucalyptus	4.3	2.90	2.10	1.67
glass	1.64	1.37	1.46	1.23
iris	1.23	1.17	1.22	1.12
metStatCoordinates	1.51	1.40	1.36	1.29
metStatRainfall	4.35	3.37	2.37	2.1
metStatRST	1.68	1.40	1.35	1.22
metStatSunshine	1.59	1.35	1.42	1.23
metStatTemp	1.85	1.60	1.51	1.37
mfeat-factors	1.99	1.75	1.57	1.31
mfeat-fourier	2.69	2.14	1.93	1.54
mfeat-karhunen	2.91	2.56	1.87	1.63
mfeat-morphological	1.79	1.55	1.50	1.36
mfeat-zernike	3.30	2.63	2.9	1.70
optdigits	3.87	3.42	2.29	2.0
page-blocks	2.76	2.38	1.99	1.73
pasture-production	1.11	1.7	1.7	1.4
pendigits	3.77	3.41	2.31	2.7
segment	1.97	1.74	1.72	1.51
squash-unstored	1.33	1.20	1.19	1.10
synthetic control	1.45	1.52	1.36	1.15
vehicle	4.45	3.49	2.38	2.6
vowel	2.54	2.29	1.77	1.61
waveform	19.92	10.73	4.61	3.59
wine	1.58	1.47	1.39	1.18
average	3.00	2.31	1.78	1.53

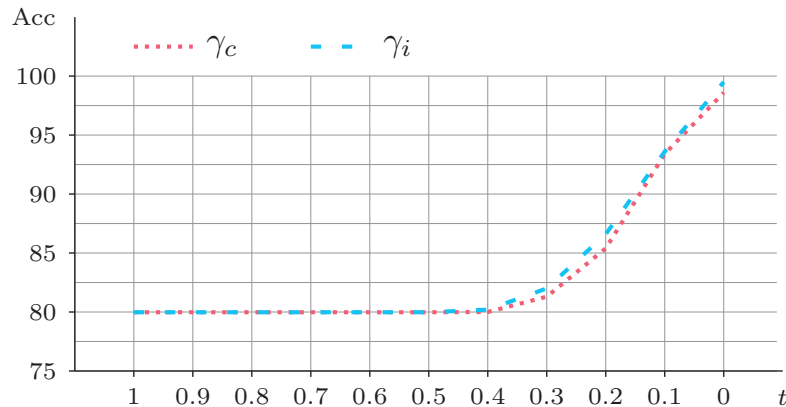


Figure 5.11.: Accuracy-rejection curves for the data set waveform.

The monotonicity expected of the dependence between rejection threshold t and classification accuracy is confirmed by the experimental results summarized in Tables 5.4 and 5.5. Using γ_c , an improvement is obtained for all data sets and γ_i leads to an improvement in all but one case. Typical accuracy-rejection curves are shown in Figure 5.11 (the plateaus in these curves are caused by the absence of instances with corresponding γ -values). In summary, these experiments clearly show that both measures of uncertainty derived by FR3, conflict and ignorance, are reliable indicators of the uncertainty involved in a classification decision.

5.7. Summary

In this chapter we proposed the FR3 method that learns a fuzzy preference structure using fuzzy rule-based models as a foundation. The idea was to derive degrees of strict preference, conflict and ignorance from the coverage degrees of a rule-based model. Through pairwise decomposition, this approach was able to learn a fuzzy preference structure for multi-class problems. It was shown that the uncertainty distinction between conflict and ignorance make sense because of the different semantic meaning. It was explained that this approach can be used for various decision making techniques, e.g. for a classification task or for an abstention decision. In the experiments we found that the discriminative ability of FR3 is considerable in terms of both accuracy and AUC.

Table 5.4.: Classification rates (acc) on the test set for different rejection thresholds and the coverage (cov) in terms of the percentage of non-rejected instances, for conflict (using γ_c) as rejection criteria. For statistical reasons, results for less than 10 instances are not reported.

Data set	rejection threshold							
	1		0.6		0.2		0	
	acc	cov	acc	cov	acc	cov	acc	cov
acd-authorship	95.2	100.0	95.2	100.0	96.2	94.9	99.4	54.9
acd-halloffame	93.1	100.0	93.1	100.0	94.1	97.8	98.4	76.3
acd-votesurvey	36.8	100.0	36.8	100.0	36.8	100.0		
cars	81.5	100.0	81.5	100.0	85.5	88.2	98.9	43.1
collins	94.5	100.0	94.5	100.0	97.2	94.6	99.3	75.0
ecoli	83.1	100.0	83.1	100.0	84.8	93.8	94.2	55.4
eucalyptus	64.5	100.0	64.5	100.0	66.2	91.6	89.5	21.7
glass	72.8	100.0	72.8	100.0	74.2	90.1	97.7	14.6
iris	94.4	100.0	94.4	100.0	94.6	98.6	99.7	80.3
metStatCoordinates	93.3	100.0	93.3	99.9	95.4	94.7	99.6	70.9
metStatRainfall	69.7	100.0	69.7	100.0	75.9	77.2	96.3	13.3
metStatRST	43.4	100.0	43.4	100.0	43.9	96.7	57.1	8.9
metStatSunshine	52.6	100.0	52.6	100.0	55.0	86.5	86.6	3.8
metStatTemp	57.3	100.0	57.3	100.0	59.2	94.2	84.5	21.8
mfeat-factors	93.3	100.0	93.3	99.9	95.1	94.5	99.6	58.3
mfeat-fourier	80.4	100.0	80.4	99.9	84.8	87.1	99.6	32.2
mfeat-karhunen	91.5	100.0	91.5	99.9	94.8	87.2	99.9	32.9
mfeat-morphological	72.2	100.0	72.2	99.9	73.4	95.5	91.6	46.2
mfeat-zernike	77.1	100.0	77.1	99.9	80.1	85.9	97.9	22.8
optdigits	96.2	100.0	96.3	99.9	98.1	93.8	99.9	62.7
page-blocks	97.1	100.0	97.1	100.0	97.7	98.5	99.4	88.3
pasture-production	71.6	100.0	71.6	100.0	72.3	91.7		
pendigits	98.1	100.0	98.1	99.9	99.0	96.5	99.9	73.3
segment	96.9	100.0	96.9	100.0	98.1	96.5	99.8	76.7
squash-unstored	74.8	100.0	74.8	100.0	74.6	96.2	76.7	25.0
synthetic control	92.2	100.0	92.2	100.0	94.4	89.2	98.2	43.0
vehicle	72.6	100.0	72.6	100.0	74.2	91.8	95.1	24.9
vowel	84.0	100.0	84.0	100.0	88.7	83.7	98.9	13.7
waveform	79.8	100.0	79.8	100.0	85.4	77.9	99.3	14.6
wine	92.8	100.0	92.8	100.0	95.1	92.1	99.7	52.2

Table 5.5.: Classification rates (acc) on the test set for different rejection thresholds and the coverage (cov) in terms of the percentage of non-rejected instances, for ignorance (using γ_i) as rejection criteria. For statistical reasons, results for less than 10 instances are not reported.

Data set	rejection threshold							
	1		0.6		0.2		0	
	acc	cov	acc	cov	acc	cov	acc	cov
acd-authorship	95.2	100.0	95.2	100.0	96.4	96.0	99.3	60.6
acd-halloffame	93.1	100.0	93.1	99.9	95.2	93.6		
acd-votesurvey	36.8	100.0	35.5	87.5				
cars	81.5	100.0	81.5	100.0	88.0	75.4		
collins	94.5	100.0	94.5	100.0	97.2	94.6	99.7	70.6
ecoli	83.1	100.0	83.1	100.0	87.1	86.0		
eucalyptus	64.5	100.0	64.7	98.6	81.0	40.4	99.8	10.1
glass	72.8	100.0	72.8	99.5	78.6	73.2		
iris	94.4	100.0	94.4	100.0	95.2	97.3	99.8	28.6
metStatCoordinates	93.3	100.0	93.3	100.0	94.3	97.2	97.7	5.6
metStatRainfall	69.7	100.0	69.7	100.0	78.6	68.8		
metStatRST	43.4	100.0	43.5	99.7	53.5	27.7		
metStatSunshine	52.6	100.0	52.6	100.0	61.0	55.5		
metStatTemp	57.3	100.0	57.4	99.9	70.1	57.5		
mfeat-factors	93.3	100.0	93.3	100.0	94.8	95.9	99.7	23.0
mfeat-fourier	80.4	100.0	80.5	99.8	89.5	73.8	99.6	12.5
mfeat-karhunen	91.5	100.0	91.5	100.0	94.0	92.5	99.4	8.4
mfeat-morphological	72.2	100.0	73.8	93.9	86.0	61.9	100.10	6.
mfeat-zernike	77.1	100.0	80.2	91.7	90.1	71.9	99.2	6.0
optdigits	96.2	100.0	96.2	100.0	97.3	96.4	99.8	51.7
page-blocks	97.1	100.0	97.1	100.0	97.7	98.4		
pasture-production	71.6	100.0	71.6	100.0	75.3	77.8		
pendigits	98.1	100.0	98.1	100.0	98.5	98.3	99.8	49.2
segment	96.9	100.0	96.9	100.0	98.0	97.0	99.9	40.7
squash-unstored	74.8	100.0	76.0	94.2				
synthetic control	92.2	100.0	92.2	100.0	94.1	90.8	98.9	24.8
vehicle	72.6	100.0	74.0	94.0	95.1	45.7	100.0	2.6
vowel	84.0	100.0	84.0	100.0	87.8	84.4	98.5	7.4
waveform	79.8	100.0	79.8	100.0	87.1	75.9	99.1	24.4
wine	92.8	100.0	92.8	100.0	93.2	97.2	99.5	20.8

6

Comparison of FURIA, HELLFIRE and FR3

In the previous three chapters novel techniques for learning and fuzzifying classification rules were proposed. This chapter has the purpose of contrasting FURIA, HELLFIRE and FR3 in order to better display their inherent characteristics. We explain the differences and similarities in the following sections.

6.1. Comparison of Motivations and Methods

6.1.1. Motivation

The purposes of the proposed classifiers are quite different. On the one hand we have the rule-based classifiers FURIA and HELLFIRE and on the other hand we have the preference learner FR3. While FURIA and HELLFIRE were conceived for solving classification problems, FR3 is more powerful since it is able to derive a fuzzy preference structure — instead of a mere class prediction — for each query instance. This can be the starting point for making classification decisions, but it may also help to tackle more complex decision scenarios.

Another distinction can be found between FURIA and HELLFIRE: The latter is a linguistic rule-based classifier, which is using rule conditions that are based on linguistically labeled fuzzy grid cells. In contrast to that FURIA

uses rule-specific fuzzy sets which are not based on a global grid-partition. While FURIA solves classification problems only, HELLFIRE claims to learn linguistic models that are interpretable by domain experts.

6.1.2. Problem Decomposition Technique

As FURIA and HELLFIRE were designed for being fuzzy rule-based classifiers, they decompose a classification problem in a 1-vs-All fashion, learning one set of rules for every class. The purpose of FR3, however, was to obtain fuzzy preference structures for query instances. In order to express the pairwise relationships, FR3 decomposes polychotomous classification problems using All-vs-All decomposition, learning one set of rules for every ordered pair of classes.

6.1.3. Rule Learning Strategy

A key distinction between FURIA and FR3 on the one side and HELLFIRE on the other side is the rule learning strategy. The rules which are used by the former are learned with a modified version of RIPPER [Coh95]. As a contrast to that, HELLFIRE learns rules with its own, novel strategy that is based on a coordinated splitting approach.

6.1.4. Fuzzy Partitioning

A major difference between the rules learned by FURIA and FR3 in contrast to the ones learned by HELLFIRE is that rules of the latter form a Ruspini fuzzy grid partition on the numeric attributes. Both FURIA and FR3 are more flexible in that concern, meaning that both do not generate a global fuzzy partition. Instead FURIA and FR3 create fuzzy sets for each rule independently.

6.1.5. Fuzzification Technique

The fuzzification technique to obtain fuzzy rules is different for FURIA, HELLFIRE and FR3. While HELLFIRE creates grid-based rules, are FURIA and FR3 models based on more flexible RIPPER rules [Coh95]. The technique to soften the rule boundaries is the same for FURIA and FR3: They

are fuzzified by carefully and gradually extrapolating the rule coverage. However, FR3 limits the extrapolation of the rules in a fuzzy way whenever an interval condition is unbound on one side, e.g. $I = [b, \infty]$ or $I = [-\infty, c]$. The motivation behind that additional fuzzification is due to the pairwise decomposition: Limiting the extrapolation helps avoiding accidental coverage of instance which do not belong to either of the classes of the current pairwise model.

The fuzzification technique of HELLFIRE is different. This is due to the fact that rules contain grid cells in the antecedent part. Basically, one grid cell can occur in more than one rule. The consequence is that the focus shifts from fuzzifying rule intervals to fuzzifying grid cells. A major difference between the fuzzification of a condition from a RIPPER rule and a grid cell is that in the former case this can be done for each rule independently. However, when fuzzifying a grid cell, one has to take adjacent grid cells into consideration. The process of fuzzifying grid cells is essentially to soften split points of a numeric attribute discretization. While the fuzzification of a RIPPER rule does not decrease the coverage within the original intervals, softening split points of a discretization decreases the area of full coverage while increasing the area of gradual coverage.

6.1.6. Handling of Uncovered Instances

For the rule-based classifiers in this work it might happen that a query instance is not covered by any rule at all. In order to give a class prediction the classifiers use different solutions. FURIA and HELLFIRE use rule stretching as fallback strategy. This means that all rules are generalized so that the query instance can be covered. In a subsequent evaluation these generalized rules are used for classification. For FR3 the lack of coverage within a pairwise model is considered as ignorance. During score calculation this (gradual) ignorance is distributed according to the class frequencies observed in the training data.

6.1.7. Score Aggregation

All three classifiers use a kind of scoring systems for the classification of an unknown instance: FURIA uses weighted voting to aggregate the coverage degrees, while FR3 aggregates the quadruples from the pairwise models using a sum operation. In contrast to these approaches, HELLFIRE applies

the Łukasiewicz T-conorm upon the weighted coverage degrees. All three classifiers predict the class that has the largest score.

6.2. Experimental Comparison

The comparison in the previous section displayed that *FURIA*, *HELLFIRE* and *FR3* are rule learning algorithms. However, when comparing the three more closely, significant differences can be found. Now we compare *FURIA*, *HELLFIRE* and *FR3* experimentally and analyze the outcome with respect to the classifiers' characteristics.

This comparison will focus on discriminative power, which is arguably one of the most important features of a (linguistic) fuzzy rule learning algorithm. To this end, we will distinguish not only between classification accuracy and AUC but also between binary and multi-class data sets. This is because of the fact that *FR3* is a polychotomous classifier only.

6.2.1. Binary Data Sets

The first experimental comparison of the introduced classifiers is concerned with *FURIA* and *HELLFIRE* only. We will focus on the 15 two-class data sets that were introduced in Section 2.1. The results for classification accuracy and AUC of this comparison are shown in Table 6.1. *FURIA* wins the direct comparison with 11 and 10 wins for classification accuracy and AUC respectively. From a statistical point of view, a sign test rejects the null-hypothesis of equal performance with an error probability of $\alpha = 0.1$, cf. Section 2.5.6.1, for classification accuracy only. However, the outcome of AUC is not that much worse.

Interestingly, the winning algorithm differs on more than one data set. However, for a two-class problem the AUC winner should also win the accuracy comparison — when tuning the decision threshold of the predicted class distribution appropriately [LF03]. Note that — not only for this purpose — a pseudo class distribution could be obtained by normalizing the class scores. Yet this tuning was neither done for *FURIA* nor *HELLFIRE*.

Table 6.1.: Average classification accuracy and AUC of FURIA and HELLFIRE.

Data set	Accuracy		AUC	
	FURIA	HELLFIRE	FURIA	HELLFIRE
acd-bankruptcy	82.57	82.15	0.88	0.87
acd-cyyoung8092	80.02	80.28	0.78	0.74
acd-cyyoung9302	82.64	81.87	0.77	0.77
acd-esr	80.90	82.62	0.67	0.67
acd-lawsuit	98.00	96.94	0.95	0.95
biomed	88.31	86.69	0.91	0.90
haberman	72.72	73.00	0.63	0.62
heart-statlog	79.75	79.63	0.84	0.82
ionosphere	89.59	88.39	0.90	0.91
liver-disorders	67.15	63.81	0.66	0.68
pima diabetes	74.71	72.92	0.77	0.73
prnn-synth	83.57	83.81	0.89	0.85
schizo-	80.52	70.52	0.75	0.86
sonar	77.01	72.35	0.78	0.81
w.-breast-cancer	95.68	94.51	0.98	0.97
wins	11	4	10	5

6.2.2. Multi-class Data Sets

The second experimental comparison of the introduced classifiers is focused on polychotomous problems: FURIA, HELLFIRE and FR3 are compared on the 30 multi-class data sets, cf. Section 2.1. The results in terms of classification accuracy and AUC can be found in Table 6.2.

To evaluate the performances we conduct the Friedman Test, cf. Table 6.2 for the classifier ranks. The corrected Friedman statistic for large N and k according to (2.5) is 38.1 for accuracy and 37.5 for AUC, while the critical value for the significance level $\alpha = 0.01$ is only 4.99. Thus, the null-hypothesis of equal performance can quite safely be rejected for both accuracy and AUC, which means that there are significant differences in the classifiers' performance.

Given the results of the Friedman Tests, we conduct Nemenyi Tests as post-hoc tests in order to compare the classifiers in a pairwise manner [Nem63]. The critical distance according to (2.6) between two classifier ranks is $CD_\alpha = 0.53$. The results of these tests are summarized in Figure 6.1 and Figure 6.2: For classification accuracy, FR3 and FURIA are both significantly better than HELLFIRE. For AUC, FR3 is significantly better than FURIA and HELLFIRE.

Table 6.2.: Average classification accuracy, AUC and ranks of *FURIA*, *HELLFIRE* and *FR3*.

Data set	Accuracy			AUC		
	FURIA	HELLFIRE	FR3	FURIA	HELLFIRE	FR3
acd-authorship	95.67(1)	91.40(3)	95.17(2)	0.98(2)	0.98(3)	0.99(1)
acd-halloffame	92.92(2)	91.51(3)	93.13(1)	0.83(2)	0.78(3)	0.89(1)
acd-votesurvey	36.92(1)	35.47(3)	36.06(2)	0.52(3)	0.54(1)	0.54(2)
cars	79.08(2)	71.84(3)	81.48(1)	0.88(2)	0.85(3)	0.92(1)
collins	96.35(1)	93.26(3)	94.53(2)	1.00(1)	0.99(3)	1.00(2)
ecoli	83.12(1)	79.67(3)	82.91(2)	0.91(3)	0.92(2)	0.94(1)
eucalyptus	60.62(2)	55.04(3)	64.25(1)	0.79(3)	0.81(2)	0.89(1)
glass	68.22(2)	64.04(3)	72.98(1)	0.81(2)	0.80(3)	0.88(1)
iris	94.76(3)	94.82(1)	94.78(2)	0.97(3)	0.98(2)	0.99(1)
metStatCoord.	93.02(2)	85.18(3)	93.30(1)	0.98(2)	0.96(3)	1.00(1)
metStatRainfall	64.51(2)	48.28(3)	69.68(1)	0.87(2)	0.82(3)	0.95(1)
metStatRST	33.56(3)	38.55(2)	43.22(1)	0.65(3)	0.71(2)	0.82(1)
metStatSunshine	49.05(2)	41.36(3)	52.94(1)	0.77(3)	0.77(2)	0.88(1)
metStatTemp	50.71(2)	47.72(3)	57.38(1)	0.76(3)	0.76(2)	0.88(1)
mfeat-factors	92.09(2)	83.30(3)	93.35(1)	0.98(2)	0.97(3)	0.99(1)
mfeat-fourier	76.69(2)	72.86(3)	80.48(1)	0.92(3)	0.95(2)	0.97(1)
mfeat-karhunen	86.47(2)	76.57(3)	91.51(1)	0.96(2)	0.96(3)	0.99(1)
mfeat-morpholog.	72.09(2)	70.95(3)	72.31(1)	0.88(3)	0.91(2)	0.96(1)
mfeat-zernike	73.67(2)	63.27(3)	77.18(1)	0.91(3)	0.92(2)	0.96(1)
optdigits	94.78(2)	82.69(3)	96.22(1)	0.99(2)	0.96(3)	1.00(1)
page-blocks	97.02(2)	93.08(3)	97.04(1)	0.95(2)	0.83(3)	0.98(1)
pasture-prod.	74.67(1)	71.74(2)	71.28(3)	0.86(2)	0.81(3)	0.86(1)
pendigits	97.77(2)	88.25(3)	98.07(1)	1.00(2)	0.98(3)	1.00(1)
segment	96.50(2)	90.40(3)	96.97(1)	0.99(2)	0.98(3)	1.00(1)
squash-unstored	76.44(1)	76.40(2)	75.52(3)	0.83(2)	0.85(1)	0.83(3)
synthetic control	89.75(2)	82.49(3)	92.24(1)	0.97(3)	0.97(2)	0.99(1)
vehicle	70.10(2)	66.99(3)	72.78(1)	0.85(3)	0.86(2)	0.90(1)
vowel	75.43(2)	47.55(3)	84.03(1)	0.93(2)	0.85(3)	0.98(1)
waveform	82.24(1)	77.31(3)	79.79(2)	0.91(3)	0.92(2)	0.94(1)
wine	93.25(1)	91.88(3)	92.70(2)	0.97(2)	0.97(3)	0.98(1)
average rank	1.8	2.83	1.37	2.46	2.4	1.13

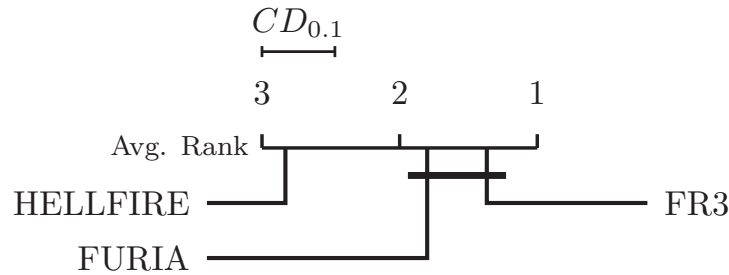


Figure 6.1.: Nemenyi Test visualization according to Demšar [Dem06]. FR3 and FURIA are significantly better than HELLFIRE in terms of classification accuracy. Significance level $\alpha = 0.1$.

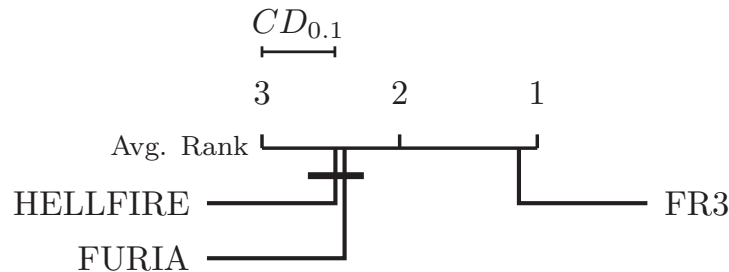


Figure 6.2.: Nemenyi Test visualization according to Demšar [Dem06]. FR3 is significantly better than FURIA and HELLFIRE in terms of AUC. Significance level $\alpha = 0.1$.

6.2.3. Summary

The results from the previous analysis of FURIA, HELLFIRE and FR3 are quite clear: For both classification accuracy and AUC is the best performance delivered by FR3. It is superior to FURIA and HELLFIRE. For the latter two we find that FURIA outperforms HELLFIRE, again in terms of classification accuracy and AUC on both two-class and multi-class data sets.

The outcome of these experiments is in concordance with the model constraints for the three classifiers. When taking into consideration that the pairwise models of FR3 have similarities with FURIA, it becomes clear that FR3 profits from its All-vs-All decomposition. As Fürnkranz pointed out, this decomposition obtains better results than 1-vs-All when applying the same base learner [Für02, Für03]. Moreover, when comparing the constraints of FURIA and HELLFIRE, the largest difference is the fact that HELLFIRE

uses a grid corset while *FURIA* does not. This grid limits the flexibility of the rule boundaries and explains why *FURIA* is better.

6.3. Discussion

In this section we discussed the differences and similarities of the new classifiers. We contrasted *FURIA*, *HELLFIRE* and *FR3* along several criteria and showed major differences. A concise overview can be found in Table 6.3, which summarizes the most important aspects. Moreover, we compared the three classifiers experimentally with focus on the discriminative power. The results reflect the constraints of the classifiers perfectly: The most flexible classifier, which is *FR3* due to its pairwise decomposition and its non-grid-oriented rules, shows the best performance. Contrary to that displays *HELLFIRE* worse results, which are due to its grid-corset and the applied 1-vs-All decomposition. *FURIA* as a 1-vs-All learner with non-grid-oriented rules scores in between.

Table 6.3.: Contrasting FURIA, HELLFIRE and FR3.

Algorithm	FURIA	HELLFIRE	FR3
Type	Conventional rule learner with fuzzified boundaries	Linguistic fuzzy rule learner	Algorithm for learning fuzzy preference structures using conventional rules with fuzzified boundaries
Problem Decomposition Technique	1-vs-All	1-vs-All	All-vs-All
Rule learning strategy	Separate-and-conquer	Coordinated splitting	Separate-and-conquer
Ruspini grid partition	No	Yes	No
Fuzzification technique	Soft post-generalization of rule intervals	Softening grid cells	Soft post-generalization of rule intervals + limiting the extrapolation
Handling of Uncovered Instances	Rule stretching	Rule stretching	Uncoverage \equiv Ignorance
Score Aggregation	Weighted voting	Łukasiewicz T-conorm	Preference Structure transformation into scores leading to weighted voting

7

Related Work

In the realm of conventional, fuzzy and linguistic rule-based classification a wealth of work has been published so far. In order to deal with this abundance, introducing related work will require distinguishing into conventional and fuzzy techniques. Besides, we will also introduce other related techniques and fields that have a significant relationship to our presented algorithms. We will present various approaches and attempt to outline the motivation for the algorithms introduced in this thesis.

7.1. Conventional Rule-Based Classifiers

The idea of using rule-based systems in machine learning is very old. The initial thoughts about logical rules go back to the area of expert systems in the 1960s. At that time human specialists defined logical inference rules as a means of modeling knowledge. Automatic learning of these rules is subject to various machine learning approaches which generalize data by IF-THEN-rules. There have been different approaches conceived and, subsequently, we introduce a selection of learning strategies that are used to learn classification rules, especially.

7.1.1. Separate-and-Conquer Rule Learning

The perhaps most popular way to induce a set of rules is to learn them following the separate-and-conquer scheme that was introduced by Michalski

with his AQ algorithm [Mic69]. We already introduced this concept in Section 2.3.1 and mentioned some algorithms based on FOIL, which is a quite popular representative for this field.

Boström proposed a variant of IREP which is called Unordered-IREP (U-IREP) [Bos04]. It has some similarities to FURIA, in fact. As its name suggests, U-IREP does not learn an ordered decision list, but instead a set of independent rules for all classes, so that there is no use for the default class anymore. Boström works on pruning and rule filtering criteria for the original IREP algorithm from Fürnkranz and not on IREP* and RIPPER from Cohen, which he mentioned as future work.

As a successor to RIPPER, Cohen introduced the *SLIPPER* rule learner for two-class problems [CS99]. *SLIPPER* combines the idea of separate-and-conquer learning with the idea of boosting. This is realized through considering an individual rule as a weak learner, which can be boosted using a generalized version of Freund and Schapire's AdaBoost algorithm [FS99].

Another RIPPER variant was introduced by Dain et al. [DCB04], namely the IREP++ algorithm which emphasizes the speed over the discriminative power. Therefore, the authors used data structures, which reduced the number of resorting occurrences and they also removed the optimization step to save time.

Another classical rule learning approach is the CN2 induction algorithm from Clark and Niblett [CN89]. It is a combination of features from the AQ family and the ID3 decision tree [Mic73, Qui86]. CN2 is a beam-search algorithm that works in a general-to-specific direction. It uses Shannon's entropy to guide the search and a significance test as a stopping criterion to handle noise effectively. CN2 learns a rule list with default class.

A very general take on separate-and-conquer rule learning was presented by Theron and Cloete with the *BEXA* algorithm that learns an unordered set of rules [TC93, TC96]. The key contribution of *BEXA* is the use of disjunctions in rule clauses. Its concept learning process starts with the most general conjunction *mgc*, which is a rule that contains a conjunct for every attribute. Internally, the conjunct consists of a disjunction of all selectors on its attribute. Initially, the *mgc* covers all examples, but is successively specialized by removing selectors from the disjunctions. The search is guided using a Laplace weighted accuracy within a beam search scenario. To avoid overfitting, it employs stopping criteria, e.g. the significance test from CN2 [CN89] and also pre- and post-pruning techniques. One of the main contri-

butions of Theron and Cloete is to subsume different covering algorithms as special variants within the BEXA rule learning framework.

Recently, Janssen and Fürnkranz investigated the influence of the search strategy and the search heuristics on the predictive quality of a rule learner [JF09]. One of the key results of the authors is that the Laplace-weighted purity (the authors use the term “precision”) search outperforms a search guided by the purity alone for a hill-climbing strategy, whereas this difference cannot be found for an extensive search. This was explained through the decreased noise-sensitivity of the Laplace-weighting.

A comprehensive overview of separate-and-conquer learning in general can be found in the work of Fürnkranz which also contains a large enumeration of corresponding algorithms [Für99].

7.1.2. Decision Tree Rule Learning

Another popular way of learning a rule-based classification algorithm is to apply decision tree learning techniques. A typical decision tree algorithm partitions the data set in a divide-and-conquer scheme recursively. This leads to a tree structure in which every inner node splits along an attribute. The leaf nodes contain a class label. To classify an unseen example, the tree is traversed from the root to a leaf. The path taken is the one that evaluates the conditions in the inner nodes to true, given the instance’s attribute values. To obtain classification rules, all those paths are converted into rules by conjuncting the conditions of the inner nodes as antecedents and setting the consequences to the most frequent classes of the respective leaf nodes.

Quinlan’s C4.5 algorithm, with its rule learner variant C4.5rules, is a well-known representative for the field of decision tree learning [Qui93]. Rules obtained using that strategy cover disjunct areas in the data space. A post-pruning and post-processing technique removes conditions from the rules, as long as this is not detrimental to predictive accuracy. The resulting rules are no longer mutually exclusive.

Another C4.5 successor is PART that was introduced by Frank and Witten [FW98]. PART uses the general separate-and-conquer learning approach as a framework. To induce a single classification rule, it builds a C4.5 decision tree from the current growing data. After adding the best rule from this tree to the ruleset, the decision tree is discarded. Then, the covered data is removed and another rule is learned with a new C4.5 decision tree. Since building a complete C4.5 decision tree requires a lot of effort for generating a single rule,

Frank and Witten proposed the generation of partial trees such that the best rule is found faster. The motivation behind this whole approach is threefold: (A) It is simple. (B) PART avoids the overfitting of typical separate-and-conquer learning. (C) It has no need for any global optimization.

7.1.3. AUC-Optimizing Rule Learning

In the recent past, measuring the area under the ROC curve (AUC) has become an alternative to measuring classification accuracy [Bra97, PF01b]. As it was explained in Section 2.5.4.2, there are some advantages of AUC over classification accuracy, e.g. for measuring the performance on data sets with skewed class distributions [PFK98].

An early empirical study on optimizing the AUC performance for rule-based classifiers was conducted by Fawcett [Faw01]. He found that weighted voting is significantly superior to other strategies, such as random selection, first rule selection or voting, when the instance is covered by multiple rules. Besides, Fawcett noted that — if appropriately used — classification rule learners are also effective at predicting probabilities. He concluded this from an AUC comparison of a rule learner to Naive Bayes.

A very interesting analysis on the relationship between rule learning and the AUC can be found in the work of Fürnkranz and Flach [FF05]. They showed that the ROC-space can also be considered as a PN-space, where P and N resemble the absolute number of positive and negative examples in the training data. Instead of plotting the true positive rate versus the false positive, they proposed plotting P versus N . According to this point of view, the ROC-space is a normalization of the PN-space to $[0, 1]^2$. Adding a new rule to the ruleset can be plotted as a linear segment in the PN-space. The segment starts with the current ruleset at the coordinate of currently covered positive and negative examples to the coordinate which resembles the new ruleset. This new coordinate is simply calculated by adding the newly covered positive and negative examples respectively. Since every new rule cannot decrease the number of covered examples, this approach creates a path through the PN-space from $(0, 0)$ to (N, P) . A separate-and-conquer rule learner, which covers all positive and negative examples at the beginning, would build that path from (N, P) to $(0, 0)$ in opposite direction, where every rule and its segment in PN-space is chosen through a greedy search heuristic.

Further research on AUC in rule learning can be found in the work of Boström [Bos05, Bos07]. One of his findings was that unordered rulesets

outperform ordered rule lists in terms of AUC due to a more “fine-grained” scale. This is in concordance with the observations made by Fawcett which state that weighted voting is better than random selection or first rule selection, see above.

Two further AUC-optimizing rule-based classifiers were put forward by Fawcett and also by Prati and Flach [Faw08, PF05].

7.1.4. Nearest Generalized Examples

An interesting combination of two different classification schemes was proposed by Salzberg under the term *Nearest Generalized Examples* [Sal91]. The proposal considers classification rules as generalizations of examples. This approach allows it to measure the distance from an instance to a rule within the Euclidean space such that lazy instance-based algorithms — such as Nearest Neighbor — can be applied [WD94].

A realization of this idea was proposed by Domingos with the RISE algorithm [Dom96]. The idea of Domingos was to consider examples as maximally specific classification rules that serve as a starting point for the model building. The learning process of RISE generalizes these specific rules and increases their coverage. Domingos proposed a nearest neighbor search on the learned rules to classify an unseen instance. Therefore, he introduced the following distance measure between a rule r and an instance \mathbf{x} for $p \in \mathbb{N}$:

$$\Delta_{\text{num}}(r, \mathbf{x}) = \sum_{I_i \in r_A} \delta_{\text{num}}^p(i)$$

The *component distance* $\delta_{\text{num}}(i)$ describes the distance between \mathbf{x}_i and $I_i = [b, c]$ on attribute A_i :

$$\delta_{\text{num}}(i) = \begin{cases} 0 & \text{if } b \leq x_i \leq c \\ \frac{x_i - c}{\max(D_i) - \min(D_i)} & \text{if } x_i > c \\ \frac{b - x_i}{\max(D_i) - \min(D_i)} & \text{if } x_i < b \end{cases},$$

where $\max(D_i) - \min(D_i)$ is used for normalization according to the largest and smallest value for attribute A_i according to the training data D . Note that Domingos introduced the distance $\Delta(r, \mathbf{x})$ both for numerical and nominal attributes. For the sake of simplicity, we will not introduce the nominal measure since it does not affect the fuzzification analysis here.

The classification decision of RISE is given by the class of the nearest rule. Ties can occur quite frequently when an instance is fully covered by multiple rules. They are solved by preferring the rule with the highest Laplace accuracy.

7.2. Fuzzy Rule-Based Classifiers

To cope with the abundance of fuzzy rule-based classifiers, we will attempt to cluster the algorithms according to the learning techniques applied. This overview will not concern Takagi-Sugeno fuzzy rules, which have a function of the input values as rule consequence. Instead, we will fully concentrate on rule-based classifiers, where the rule consequences are plain class-assignments.

7.2.1. Grid-Oriented Approaches

A very intuitive and simple early approach (denoted as *WM* approach) for learning linguistic fuzzy rules was proposed by Wang and Mendel [WM92]. Wang and Mendel suggested partitioning the attributes of the input space with a fixed number of (e.g. triangular) fuzzy sets into a fuzzy grid. From every example they derived a fuzzy classification rule that had the grid cell which covered the example maximally as antecedent and the example's class label as a consequence. In order to avoid conflicting rules, the authors proposed selecting the most frequent one, which also has the maximum support from the training data. Furthermore, the rules can be weighted by a human expert. As a generalization, Wang and Mendel mentioned melting neighboring rules in the grid to cover whole columns or rows. An important note is that Wang and Mendel did not focus on classification learning here, but on other learning tasks such as truck backer-upper control.

Chi et al. proposed setting up the initial fuzzy partition of the data space by using self-organizing maps as a starting point for the WM rule generation process [CWY95, CYP96].

Nakashima et al. expanded the WM approach in such a way that it copes with weighted examples [NSYI07]. The novel aspect of this work is that the fuzzy membership is multiplied with the example's weight.

Another grid-based approach was proposed by Hong and Chen [HC00]. They suggested a three-step procedure for learning the fuzzy rule-base: (A) finding the relevant attributes, (B) partitioning the relevant attributes into

a fuzzy grid, (C) learning the rules. For finding the relevant attributes, they count the number of occurrences of every value and then select the attributes with low entropy. The number of attributes chosen is set according to an error threshold. For every attribute, the number of initial fuzzy sets is set in accordance with a statistic argument. Afterwards, neighboring fuzzy sets are merged. For learning the rules, the authors proposed the most frequent class of a grid cell as fuzzy classification rule consequence. A four step procedure for obtaining a grid-based linguistic fuzzy rule-base for classifying gene expression data was suggested by Vinterbo et al. [VKOM05]. The procedure consists of: (A) selecting relevant attributes using the Wilcoxon rank, (B) partitioning those attributes using training data quantiles, (C) using the notion of minimum discerning sets for inducing a rule-base and (D) filtering redundant rules.

Carmona et al. proposed a post-processing technique for fuzzy rule-based systems that deals with conflicting rules and rule extrapolation [CCZ04]. The main concern of this approach is to solve the conflicts between rules that were derived from examples from the same grid cell. This is done by introducing exceptions to the ruleset. The authors argued that this strategy allows for more general rules which are more interpretable. As a further step to increase interpretability, the authors suggested techniques such as rule reduction, rule merging and exception merging.

A typical difficulty of grid-based approaches is that the number of partitions must be defined a priori. One way of avoiding this problem is to use multiple partitions parallelly. Ishibuchi and Yamamoto introduced an approach that uses multiple homogeneous partitions in an evolutionary learning scheme [IY02] (for more information on evolutionary fuzzy rule-based systems see below). The drawback of this approach is that it can increase the hypothesis space tremendously when the number of partitions is enlarged.

7.2.2. Non-Grid-Oriented Approaches

A critical analysis of grid-oriented approaches was conducted by Alcalá et al. [ACH01]. The authors discussed the trade-off between grid-oriented and more flexible non-grid-oriented fuzzy rule-based models. Therefore, they distinguished two types of soft rules: (A) the *linguistic rules* with fuzzy sets coming from a global grid and (B) the *fuzzy rules* with fuzzy sets that are rule specific. While the former are typically used to increase interpretability, the latter are used to improve accuracy.

7.2.3. Fuzzy Propositional and Fuzzy First-Order Logic Rule Learning

A fuzzy extension to Quinlan’s classical FOIL algorithm, cf. Section 2.3.2, was proposed by Drobics et al. [DBK03]. The authors criticized the sharp decision boundaries that are caused by the interval-like splitting of FOIL which is due to the use of Boolean logic. The new algorithm *FS-FOIL* overcomes this problem with the help of fuzzy logic. Furthermore, it uses a beam search instead of a greedy one. The authors showed that FS-FOIL is capable of doing classification both in theory and on practical examples. Further related work of this domain be found in [PRS03, SP07].

An interesting take on first-order logic is the *Label Semantics* framework of Qin and Lawry [QL08]. This framework is based on random-sets with the idea that a value from a domain can be represented through multiple gradually assigned labels. Semantically the authors switched from “membership degree” to “appropriateness degree” and defined a fuzzy classification rule through a multi-dimensional appropriateness degree. The learning itself is conducted through the FOIL algorithm which is applied onto this new data representation.

7.2.4. Fuzzy Set Covering

Several approaches to learning a fuzzy rule-based classifier based on a fuzzy set covering approach were introduced by van Zyl and Cloete in terms of their *FUZZYBEXA* framework [Cv06]. This framework is a fuzzy generalization of the BEXA framework [TC93, TC96]. As a novelty, the fuzzy variant is able to work on fuzzy sets with gradual set coverage.

One variant of the FUZZYBEXA framework is the *FuzzConRI* classifier that is a fuzzy generalization of CN2 [vC04, CN89]. The novel version introduces coverage degrees and replaces the conventional junction operators through fuzzy T- and T-conorms. Examples are considered as belonging to all linguistic terms to a certain degree. An α -cut is used to set a minimum threshold on the membership degrees. van Zyl and Cloete showed that FuzzConRI is sensitive to this threshold on some data sets. To obtain the fuzzy sets for the numeric attributes, the authors put bell-shaped membership functions on top of the cluster centroids that were learned through a clustering procedure according to Surmann [Sur00].

A deeper investigation on the impact of different evaluation functions for FuzzyBEXA was also conducted by the same authors [Cv04].

7.2.5. Hybrid Approaches

The perhaps most popular way to obtain a linguistic fuzzy rule-based classifier is to use a hybrid scheme that combines the fuzzy rules with another learning structure. Many strategies have been proposed concerning how to accomplish this. We will concentrate on the main approaches, but also mention the ones encountered less frequently.

7.2.5.1. Evolutionary Algorithms

An extremely popular family of techniques for the learning of linguistic fuzzy rule-based classifiers are the evolutionary algorithms, most prominently the Genetic Fuzzy Systems. The strategy behind these is to apply a learning technique that is inspired by natural evolution [Gol89]. To enable a computer to learn in this way, the solution is considered as a chromosome that has to be encoded in an appropriate scheme. Furthermore, the genetic operators must be defined. The genetic algorithm starts with a population of random chromosomes. Through the application of the genetic operators, this population is refined to better solve the given task. The idea behind this follows Darwin's well-known "Survival of the Fittest" notion, which states that a better solution shall be retained, while others can be removed from the population. The creation of an appropriate fitness function for selecting the best individuals plays the most important role at this stage. In order to create new solutions, a recombination of solutions is made, namely, either a crossover of existing chromosomes or a mutation thereof. According to Cordón et al. the fitness evaluation, the selection operator and the recombination are the three elements found in every genetic fuzzy system [CGH⁺04].

In a very recent publication, García et al. discussed different techniques and measures for analyzing the discriminative ability and the interpretability of *Genetics-Based Machine Learning* algorithms [GFLH09]. This paper resembles the work of Demšar due to the introduction and explanation of various statistical methods, but also expounds the detail of how to grasp the quality of a genetic machine learning algorithms. The authors proposed measuring the discriminative power in terms of accuracy and Cohen's Kappa

[Coh60]. As a measure for interpretability, the authors suggested measuring the number of rules and the rule length.

Cordón et al. distinguished three main genetic fuzzy systems: (A) the Pittsburgh approach [Smi80], (B) the Michigan approach [HR78] and (C) iterative rule learning [Ven93]. (A) considers the whole set of rules as a single chromosome and the population as set of rulesets; (B) assumes that a single rule is a chromosome and the population to be a set of rules, (C) also considers the rule as a chromosome, but it learns them in an iterative fashion.

According to Cordón et al., one can distinguish the following genetic algorithm settings for classification learning:

1. tuning fuzzy memberships when rules are fixed
2. learning fuzzy rules using given fuzzy membership functions
3. 1. + 2. in an alternating order
4. memberships and rules simultaneously
5. rules, membership function, membership function shape, parameters, operators, etc. simultaneously

Pittsburgh Genetic Fuzzy Rule Learning A combination of grid-oriented fuzzy rule learning and a Pittsburgh-style genetic algorithm was proposed by Ishibuchi and Yamamoto [IY04]. The authors proposed to induce a set of candidate rules in a grid-based manner which is then used for extracting a representative subset of rules. For this extraction, a *multi-objective genetic algorithm* (MOGA) is designed that is able to find a non-dominated set of rules with respect to three different objectives: (A) maximizing the discriminative power, (B) minimizing the ruleset, (C) minimizing the rule length. The authors suggested that such a ruleset cannot be found in general. Instead, they wanted to have their algorithm return all Pareto-optimal rulesets with respect to the three objectives. Then, the authors extended MOGA to MOGLS (Multi-Objective Genetic Local Search) which combines the global genetic algorithm with a local search that helps to converge to solutions faster.

Bacardit and Krasnogor presented a combination of Pittsburgh-style learning and ensemble techniques [BK07]. They proposed building a voting ensemble of genetically learned rulesets for obtaining a classification decision

according to the idea of Breiman [Bre96]. Apart from this, a hierarchical approach for ordinal classification (meaning $\lambda_i < \lambda_j$ for $1 \leq i < j \leq m$) was presented, which recursively splits the examples into lower $\{\lambda_1, \dots, \lambda_i\}$ and upper classes $\{\lambda_{i+1}, \dots, \lambda_m\}$, keeping the number of examples in both halves as balanced as possible. This is repeated until there are only binary problems which then are solved using the ensemble of genetic rule learners proposal.

Michigan Genetic Rule Learning A Michigan style linguistic fuzzy rule learner was proposed by Orriols-Puig et al. [OPCBM09]: *Fuzzy-UCS*. This algorithm is a fuzzy variant of the interval-based UCS algorithm, which is a variant of Wilson's XCS rule learning, system [BMGG03, Wil95].

Fuzzy-UCS is capable of online learning, which means that it is able to learn from a stream of examples. Apart from ordinary classification learning the authors showed that Fuzzy-UCS is also capable of learning a model for a problem involving nearly 500,000 examples and 23 classes.

Iterative Genetic Rule Learning A representative of iterative genetic rule learning was proposed in Section 2.5.3.3: The SLAVE algorithm [GP99, GP01].

A combination of iterative genetic rule learning and boosting was introduced by Hoffmann and also by del Jesus et al. [Hof01, dHNS04]. These approaches apply Kearns' idea of boosting to the evolutionary rule learning scheme [Kea88]. More specifically, both algorithms use the AdaBoost algorithm proposed by Freund and Schapire [FS99]: After learning a single rule, the boosting procedure reweighs the examples. This leads to a set of weighted linguistic fuzzy rules.

Mansoori et al. pointed out that a typical drawback of existing iterative genetic rule learners is the fact that the problem size grows with the number of dimensions. To cope with this problem, they introduced the steady-state *SGERD* evolutionary rule learner that limits the number of generations in dependence of the problem dimensionality [MZK08].

Genetic Rule-Based Model Tuning A genetic post-processing technique for fuzzy rule-based classification algorithms is the tuning of fuzzy rules, parameters, membership functions, etc. The purpose of this tuning is to improve the model in terms of accuracy or interpretability most often.

A genetic tuning strategy was proposed by Alcalá et al. [ACH03]. The proposed approach combines rule selection with rule weighting for improving the cooperation between the rules.

Casillas et al. proposed a genetic tuning algorithm that optimizes the membership functions through non-linear scaling, parameter modification and linguistic hedges [CCdH05].

Another way to improve classification accuracy of fuzzy rule-based systems was proposed by Alcalá et al. [AAFH07]. The authors introduced a genetic algorithm that optimizes the fuzzy membership functions through a lateral shifting. In contrast to more general tuning approaches, the shifting strategy keeps the search space smaller and preserves the interpretability aspects of the fuzzy sets.

7.2.5.2. Neural Networks

A very common way to obtain a linguistic fuzzy rule-based classification algorithm is through the use of a neural network.

Neuro-fuzzy methods encode a fuzzy system as a neural network and apply corresponding learning methods (like backpropagation) [MH00, NKK97]. Fuzzy rules are then extracted from a trained network.

The self-organizing neuro-fuzzy multilayered classifier *SONeFMUC* was introduced by Mitrakis et al. in [MTP08]. Their approach combines fuzzy neuron classifiers (FNC) layer wise. Every layer is constructed using a combination of parent FNCs to enhance the predictive quality. The structure of this network is developed while learning. Therefore, it considers only the attributes with the largest amount of information. After learning, another genetic algorithm is used to tune the created model for better discriminative power. To obtain classification decisions, this approach has to make successive decisions and feature transformations.

A combination of Neuro-fuzzy learning and a flexible quasi-triangular norm was proposed by Rutkowski and Cpałka [Rk05]. The *H*-function they proposed can be used to blend T-norms and T-conorms through a parameter setting. Both the model and this parameter are learned in the building phase.

An approach to fuzzy neural network classification with support-vector-based learning was made by Lin et al. [LYL⁺06]. The proposed support-vector-based fuzzy neural network (*SVFNN*) learns the parameters for the fuzzy neural network using a support vector approach.

7.2.5.3. Support Vector and Spectral Fuzzy Rule Learning

Chen and Wang proposed a strategy for learning a fuzzy rule-based classifier by using a support vector approach [CW03]. The authors showed that, according to some assumptions, it is possible to connect an additive fuzzy rule-based classification system with the Mercer kernel to a *positive definite fuzzy classifier*.

In a recent article, Evsukoff et al. proposed learning a fuzzy rule-based classifier by using spectral analysis [EGdE09]. The rule learning is conducted using a clustering approach, so that every cluster resembles a fuzzy classification rule.

7.2.5.4. Simulated Annealing

An improvement to the approach of Wang and Mendel was made by Casillas et al., who criticized that the WM algorithm had a good local reasoning but no global cooperation between the rules [CCH00]. The idea is to assign a grid cell not the most frequent class among the covered examples, but another one that better fits into the overall model. Therefore, the authors proposed using simulated annealing as a heuristic within the search space. Simulated annealing starts with an initial solution and explores a neighboring solution. If the neighbor has a smaller mean square error (MSE), the neighbor solution is chosen and the process repeated. To avoid local maxima, a worse neighbor is accepted as a better solution with a probability that decreases with the number of iterations.

7.2.6. Clustering-Based Techniques

Another way of inducing fuzzy membership functions that might be used for classification is fuzzy clustering [HKKR99]. The idea is that every cluster is represented through a fuzzy membership function that is used within the fuzzy classification rules. One approach from that field was introduced by Höppner and Klawonn who used a combination of fuzzy c-means clustering and memberships based on the Voronoi Distance to find the fuzzy membership functions [HK04].

7.2.7. Divide-and-Conquer

A very early divide-and-conquer approach for obtaining a fuzzy decision tree was introduced by Cios et al. as a fuzzy variant of Quinlan's ID3 algorithm [CTLL91, CS92, CL92]. The fuzzy ID3 algorithm applies the iterative tree learning approach to pre-defined fuzzy sets by using an entropy measure. Another fuzzy ID3 variant that assumes a human way of dealing with uncertainties was proposed by Yuan and Shaw [YS95]. It starts with a partition of the numerical attributes into triangular fuzzy sets by using a Kohonen feature-map algorithm [Koh82]. The center points of the fuzzy sets are set to the cluster centroid points. Then it induces the fuzzy decision tree that splits according to the fuzzy ambiguity which measures a degree of unspecification. The algorithm transforms the tree into rules which are further simplified by deleting conditions as long as no decline in terms of truth (i.e. purity) can be observed. An important parameter for this algorithm is the α -cut that is used as a threshold to define the minimally allowed membership degree. The classification is done according to the rule that covers the instance to the largest degree. The fuzzy ID3 algorithm has been quite popular throughout the last years [ISNM96, BX05, ZC06, HY08].

Chiang and Hsu introduced the fuzzy classification tree algorithm *FCT* that — given ordinary classification data — predicts possibility distributions instead of single classes [CjH02]. The tree learner works without initial fuzzy sets. It creates the fuzzy set, while splitting using a c-means clustering approach. In the experiments the authors discovered that their approach improves over the results from Yuan and Shaw [YS95]. According to the authors, the lack of flexibility is due to the pre-partitioning of Yuan and Shaw's ID3 variant that is unable to adapt to local regions.

A quite general overview of fuzzy decision trees was written by Jarnikow [Jan98]. Further research on fuzzy decision tree learning can be found in [WCQY00, MJG05].

A different take on decision trees was made by Olaru and Wehenkel who introduced the *soft decision tree (SDT)* which contains both conventional and fuzzy learning aspects [OW03]. The idea is to learn a fuzzy decision tree which is not based on linguistic fuzzy sets, but which uses fuzzy splitting. This is conducted in two steps: (A) searching an optimal split-point according to a squared error function and then (B) softening the split by basically replacing the intervals with fuzzy intervals in a symmetric way. Olaru and Wehenkel found improvements in terms of discriminative ability due to the softness of

the split. The SDT is built to solve binary problems, i.e. learning a tree for one class only and using the score reciprocity to determine the prediction for the other class. Consequently, a decomposition has to be conducted in order to solve multi-class problems. Olaru and Wehenkel suggest 1-vs-All.

Other related approaches to fuzzification of decision tree splits can be found in the literature [JLL97, SL99, PF01a, TEF08].

7.2.8. Conventional vs. Fuzzy Partitions

Ishibuchi et al. analyzed the difference between a grid partition and a fuzzy partition [INN05, pp. 142-160]. The authors considered the case in which the (fuzzy) partition is determined before the rule learning process. The difference between the fuzzy and the conventional partition is that in the fuzzy one the grid cells cover a larger area in the data space. According to the authors, this is beneficial since one example can be used for the learning of multiple rules, while in the conventional partition this is not possible. Further positive effects can be expected for sparse areas of the data space in which the increased coverage helps to induce rules for more grid cells. Besides, the adjustment of the decision boundaries using rule weighing may also be considered as a positive reason for fuzzy over conventional grid-partitioning.

A general examination of conventional interval fuzzification was given by Kuwajima et al. [KNI08]. This article partly resembles the work of Ishibuchi et al. [INN05, pp. 142-160], see above. Kuwajima et al. made an experimental analysis of different degrees of interval fuzzification. The proverbial lesson to learn from this work is that the interval fuzzification improves the accuracy on the test data. Unfortunately, there is no guideline given concerning “how fuzzy” an interval has to be for this improvement. Furthermore, the authors made no attempts at explaining the beneficial effects of the fuzzification.

7.2.9. Rule Weights and Confidence Factors

An iterative weight learning strategy for linguistic fuzzy rules was proposed by Nozaki et al. [NIT96]. The authors proposed the *Reward and Punishment* (*R&P*) algorithm that increases the rule weight if a rule classifies an instance correctly and decreases it in the other case.

A very interesting article about rule weights and the geometrical consequences thereof was written by Ishibuchi and Nakashima [IN01]. The authors

showed that unweighted grid-based fuzzy rules may create decision boundaries which are not parallel to the axes if there are cells in the grid for which no fuzzy rule was learned. Furthermore, the authors found the same flexibility also for complete grids if the rules are weighted. It was shown that the decision areas of the fuzzy rules vary a lot if different rule weights are assigned.

A proposal for two novel rule weighting heuristics was done by Ishibuchi and Yamamoto [IY05]. The core aspect of the first new heuristic is to exploit the difference in terms of fuzzy purity between the most frequent class and the second most frequent class with the same antecedent. The second heuristic calculates the difference in terms of purity between the most frequent class and the sum of all other classes with the same antecedent.

Jahromi and Taheri proposed a strategy for assigning rule weights to linguistic fuzzy classification rules for both single-winner and voting classification schemes [JT08]. The suggested algorithm applies a hill-climbing search for an optimal set of rule weights. The authors proposed a method that is able to find an at least as good or even better neighbor solution for one single rule weight assuming that the other rule weights remain steady. Transforming a multi-class problem at that point into a two-class problem where the class label of the current rule is positive and the remaining ones are negative is essential. By using a threshold and assuming a scored predictions, the authors are able to calculate the optimal rule weight. The search process iterates over all rules until no improvements can be found. The drawback of this strategy is that it is sensitive to the ordering of the rules.

More proposals on how to learn the weights for classification rules can be found in the literature [ZM07, FJ09].

Nauck and Kruse suggested a noteworthy take on rule weights [NK98]. They explained that rule weights can be considered as a modification of the fuzzy membership function. A consequence of this is that a weighted rule would e.g. be non-normalized if it is transformed to an unweighted one. The authors criticized that this would damage interpretability.

7.3. Dealing with Uncertainty

Issues of uncertainty and reliable classification have been addressed under various perspectives in the machine learning literature and remain to be an active area of research [KK02, VGS03]. Even though the focus is definitely on

probabilistic methods, alternative frameworks for modeling and representing uncertainty have also been investigated [Den95, Hül03]. A distinction between different types uncertainty has been made, for example, in connection with reject options for nearest neighbor classification [Hel70], where a distance reject (non-existence of neighbors close enough to the query) is distinguished from an ambiguity reject (existence of close neighbors from different classes).

7.4. Interpretability

The interpretability of fuzzy rule systems was discussed by Mikut et al. [MJG05]. The authors were in agreement with Zhou and Gan (cf. Section 4.4.5) concerning most of the model characteristics necessary to be considered as interpretable [ZG08]. In contrast to Zhou and Gan, they did not distinguish between low-level and high-level interpretability and also kept the discussion rather concise. They preferred rules without certainty factors or weights. This demand is reasonable, but it typically conflicts with the discriminative quality of the model since rule weights shift the decision boundaries and improve classification accuracy often, as was mentioned before. Mikut et al. also proposed that fuzzy membership functions and the classification reasoning should be intuitively comprehensible.

We already mentioned the work of García et al. which focused on statistical evaluation of genetics-based machine learning algorithm [GFLH09]. As a measure of interpretability the authors suggested the product of the number of rules and the total number of antecedents as *complexity*. The authors highlighted that this interpretability analysis must be conducted carefully since there are many traps to avoid, e.g. different rule types. The statistical analysis of interpretability might only be conducted under appropriate circumstances.

A relevant criticism to the utility of linguistic rules and models can be found in the survey of Hüllermeier [Hül05]. He argued that linguistic terms and also the models are highly subjective mainly because of context-dependencies. Moreover, he mentioned the problem that even if the rules are interpretable this does not have to be the case for the overall model. A large number of rules (~ 40) with a reasonable number of antecedents ($\sim 5 - 7$) might be too hard to grasp.

7.5. Discussion

In the following section we will discuss the novel algorithms in the context of the related work. We show similarities, differences as well as the new ideas emerging in comparison to existing approaches.

7.5.1. FURIA

We introduced related techniques from the realm of conventional rule learning. The existing approaches from this domain are very different in comparison to the techniques proposed in this thesis: In contrast to FURIA, the focus of the other algorithms is not on soft coverage degrees. Consequently, there is no discussion about the difference between the shape of the rules and the effective rule boundaries that are obtained through the classification process. The only classification algorithm that is somehow comparable to FURIA is Domingos' RISE.

When comparing RISE to FURIA, we will find some aspects that are similar, but we will encounter very different characteristics. The following list contrasts both approaches:

- When FURIA covers an instance within its rule core, it behaves similar to RISE. FURIA assigns the maximum membership of $\mu_r = 1$, while RISE observes $\delta_{\text{num}} = 0$. The assigned values are reciprocal for this point.
- When a RISE rule does not cover the query instance, it considers the distances $\delta_{\text{num}} > 0$. But when FURIA does not cover the query instance with a rule core two things might happen:
 1. The instance might be covered within the fuzzy extension to a degree larger zero. The farther away from the initial core, the lower the membership value would be.
 2. The instance might not be covered at all.

In the first case, the FURIA rule behaves in a comparable fashion to the RISE rule. FURIA will find a gradual membership degree that decreases with the distance to the rule core linearly. RISE would return the distance directly. Since RISE considers the whole data space, it normalizes the distance to the unit interval. Since FURIA returns a membership function, its return value is also in the unit interval. Here,

the values — although not necessarily reciprocal — are at least anti-proportional. In the case that FURIA does not cover the instance we see the first dissimilarity: In this case FURIA limits the influence of the rule to a certain region to the data space.

- Both δ_{num} and I^F are monotonic continuous functions.
- RISE uses the p -norm to calculate distances in the data space. FURIA uses a T-norm to combine membership values. The T-norm is not a metric since it violates the triangular inequality.
- To classify an instance, FURIA applies a weighted voting scheme in which the instance's coverage degrees are taken into consideration. RISE would apply a very similar scheme, if it would not only consider the nearest-neighbor rule but the n -nearest-neighbor rules according to their weight and their distance, where n is the overall number of rules.

From this analysis we can conclude that RISE looks similar to FURIA but it is different nevertheless.

The realm of soft decision trees, e.g. the SDT algorithm from Olaru and Wehenkel [OW03], seems to have some similarities with FURIA. Both approaches fuzzify a conventional splitting point according to some optimization criterion to obtain soft boundaries. The difference is that the one is a decision tree split, while the other one is a rule boundary split.

To the best of the author's knowledge, there has been no effort made so far to soften boundaries of conventional classification rules. Yet, as we learned from this thesis, this is a direction that brings significant improvements to this scientific field.

7.5.2. HELLFIRE

The motivation behind the HELLFIRE algorithm was the fact that there is a gap between the linguistic grid-oriented and the non-grid-oriented fuzzy learners. While the non-grid-oriented approaches are flexible in choosing their boundaries, such as Olaru and Wehenkel's SDT, the others have a very strict grid corset. The consequence is that many grid-based algorithms do not even try to learn good decision boundaries. Instead, they take the fuzzy partition as fixed input. Often, this is done by merely dividing the attribute into triangular fuzzy sets in a uniform fashion. The consequence is that the

curse of dimensionality strikes hard and that the rules have a poor discriminative power. The grid-based algorithms that try to find a good partitioning also prove to be not very effective. They typically use some kind of genetic algorithm that is unable to cope with the incredibly large search space of even moderately sized problems. The consequence is that the predictive performance is extremely poor — especially when considering how much effort in terms of computation time is invested. The approach of HELLFIRE introduced a more efficient learning strategy to the field of linguistic fuzzy rule-based classifiers. It combines aspects of well-known veteran decision tree algorithms with a novel search strategy that assures linguistic interpretability. The idea to learn the rules, while discretizing the data is completely new. In addition, something like the coordinated splitting cannot yet be found in the literature. And this is what separates HELLFIRE from locally acting decision trees. For the supervised discretization of multiple dimensions, existing approaches use splitting or multi-splitting procedures that are unable to grasp the true underlying structure in the data. The removal of adequately covered areas in the learning approach of HELLFIRE assures that the focus remains on the interesting spots only. The removed parts of the data space can no longer distract further splitting.

7.5.3. FR3

The FR3 algorithm is based on FURIA and uses the majority of its techniques. The novelty of FR3 is the capability to learn a fuzzy preference structure using fuzzy classification rules that was briefly mentioned by Hüllermeier and Brinker [HB08]. In this thesis we brought this idea to fruition and showed its capabilities.

7.6. Summary

In this chapter we compared our three novel approaches to the ones existing in the literature. We found similarities existing between both groups. But from an overall point of view, we can confirm that the techniques introduced can be seen as a significant advancement in this field.

8 Conclusion and Outlook

In order to conclude this dissertation, we mention some final words which will highlight the most interesting findings and developments from our research. In a list of suggested future research, we will mention potential directions in which we can go from here.

8.1. Conclusion

In this thesis we introduced new procedures for and insights about the interface between conventional rule learning techniques and fuzzy logic. Three new fuzzy rule-based classification algorithms were proposed: (A) We introduced FURIA as a very powerful fuzzy rule learner which uses fuzzy rules that are based on ordinary conventional classification rules. We showed that fuzzification is beneficial for both accuracy and area under the ROC curve. We also introduced a novel rule stretching method that outperforms the existing technique in terms of memory and time consumption without deteriorating discriminative power. (B) We demonstrated how FURIA rules can be used to learn fuzzy preference structures which can be used as a starting point for manifold decision making processes. For that purpose, we developed the new FR3 classifier that is able to distinguish different kinds of uncertainty in a very intuitive fashion. (C) We developed the novel HELLFIRE algorithm that is learning interpretable and comprehensible linguistic fuzzy classification rules.

In extensive practical experiments we were able to show the qualities of the new algorithms and in addition, were able to draw comparisons to established — both conventional and fuzzy — classification algorithms. We found that the suggested classifiers are quite strong in terms of discriminative power. For FURIA we found that the improvement over RIPPER came at a very reasonable price of slightly larger models and runtime. For FR3 we found that conflict and ignorance are two semantically different kinds of uncertainty that can be used for further decision making processes. And finally we found that HELLFIRE constructs very interpretable and comprehensible models. All of these outcomes show that the new algorithms are significant improvements on state-of-the-art conventional, fuzzy and linguistic fuzzy rule learners.

8.2. Future Work

As is nearly always the case with research, there is still room for improvement. The same holds true for the approaches and techniques proposed in this thesis. We will now give some pointers of what scientific treasures might still be out there to be excavated.

8.2.1. FURIA

The FURIA algorithm is the one for which not many special future plans exist. There are two main aspects that need further attention: (A) The examination whether appropriate data structures might improve the fuzzification strategy. Since efficiency is an important point to deal with large amounts of data, this research direction is definitely important. (B) The investigation whether there is some way to make the FURIA models linguistically interpretable. Since FURIA's discriminative ability is already quite strong, improvements in this direction would enhance its use.

8.2.2. FR3

For the FR3 algorithm it would be interesting to analyze the decision making possibilities that are available due to the discerning between conflict and ignorance. While this research direction is highly application specific, there are still decision making types which are somewhat general in nature, e.g. whether or not to abstain.

8.2.3. HELLFIRE

For HELLFIRE it would be interesting to find out whether the grid-based approach is capable of achieving discriminative abilities that are comparable to the ones of FURIA. Such an insight could determine whether the interpretability of linguistic fuzzy rule-based classifiers comes at the price of a worse classification performance. Furthermore, the implementation of better stopping and pruning strategies for HELLFIRE is an area where new discoveries will certainly be made. This expectation would be in agreement with the direction in which research of conventional decision trees and conventional rule learners went.

8.2.4. General Directions

The following list of potential research directions is in no way exhaustive. It is more of an effort to show in which direction future work could be heading.

8.2.4.1. Weighing Examples for Boosting

A very hot topic in recent years has been boosting, especially the AdaBoost algorithm. The meta techniques originating from this field are very strong in improving a weak learner's discriminative ability. A preliminary characteristic of such an algorithm is that it is capable of handling weighted examples. In fact, all three algorithm FURIA, FR3 and HELLFIRE should be able to cope with weighted examples. Nevertheless, a thorough analysis should clarify whether the algorithms or perhaps the rules could be subjected to boosting. Besides, weighed examples could also occur in a classification task. This makes this direction an important one in order to allow the algorithms to exploit all data information in a reasonable way.

8.2.4.2. Parallelization

For all three algorithms the advent of multi processor computers offers new opportunities. The parallel execution might improve the building time of the classifiers. For FR3 one strategy could be to parallelize the learning of the pairwise problems that could result in a speed-up which is quadratic in the number of classes. For FURIA the parallel learning of the 1-vs-All rulesets could achieve a speed-up that is linear in the number of classes. For HELLFIRE a parallelization of the split calculation for the different subsets

would be a reasonable opportunity. Denoting the possible speed-up is here harder, however. The first problem is that the broader the search becomes, the more splits can be analyzed parallelly. The second problem is that this parallelization has to be synchronized for the split coordination. Without going into detail, there should be plenty of parallelization opportunities that could follow this work.

8.2.4.3. Streaming Data

The wide distribution of mini-computers or sensors has nowadays led to an increase in data and especially data streams, e.g. real time stock trading data, traffic lane car counting and measuring data, etc. In order to learn the time-varying patterns early enough, there is a large need for machine learning algorithms that are able to cope with data streams. This could also be a direction for further investigation.

8.2.5. Outlook

The results which were found in this thesis are by no means the end of this scientific path. In fact, it is really the beginning. There are still plenty of opportunities to learn well-performing linguistic fuzzy rule-based classifiers using lean but effective algorithms. Moreover, the combination of fuzzy logic and conventional machine learning algorithms will be a fertile field for future research.

A

Tables

Table A.1.: Average classification accuracy on the test data for variants of FURIA.

Data set	prod	min	crisp	w/o
analcatauthorship	92.809	92.809	92.557	91.228
analcatabankruptcy	81.326	81.329	82.580	76.127
analcata-cyyoung8092	76.012	76.042	76.192	73.308
analcata-cyyoung9302	78.742	78.709	79.059	76.491
analcata-esr	80.091	80.091	80.918	75.971
analcata-halloffame	90.878	90.876	90.841	89.383
analcata-lawsuit	97.770	97.770	97.726	97.024
analcata-votesurvey	8.561	8.561	8.561	7.336
biomed	85.425	85.411	85.188	82.399
cars	74.605	74.591	74.518	72.339
collins	96.017	96.029	95.103	89.676
ecoli	80.590	80.581	79.963	76.893
eucalyptus	47.929	47.929	47.721	46.766
glass	60.661	60.647	59.691	54.498
haberman	67.246	67.246	67.323	66.939
heart-statlog	73.329	73.329	73.144	71.923
ionosphere	86.644	86.627	86.510	84.339
iris	94.000	94.000	93.333	91.745
liver-disorders	57.667	57.667	57.471	56.123
metStatCoordinates	92.189	92.183	91.981	90.287
metStatRainfall	56.119	56.118	55.691	51.634
metStatRST	22.540	22.540	22.232	20.241
metStatSunshine	41.363	41.370	41.128	36.623
metStatTemp	42.041	42.037	41.850	40.391
mfeat-factors	88.999	88.993	88.746	86.356
mfeat-fourier	70.696	70.693	70.278	66.624
mfeat-karhunen	82.403	82.404	81.678	77.450
mfeat-morphological	68.099	68.099	68.097	66.804
mfeat-zernike	68.084	68.093	67.572	63.879
optdigits	92.560	92.560	92.314	90.400
page-blocks	96.529	96.528	96.376	95.902
pasture-production	64.653	64.653	63.300	58.215
pendigits	97.073	97.070	96.701	95.021
pima diabetes	68.942	68.942	68.912	67.832
prnn-synth	81.045	81.045	80.999	79.500
schizo-	72.386	72.377	71.903	69.932
segment	95.536	95.538	95.132	93.694
sonar	70.210	70.195	69.631	66.048
squash-unstored	72.878	72.878	73.196	67.906
synthetic control	85.529	85.520	84.480	80.260
vehicle	62.848	62.845	62.626	61.075
vowel	71.108	71.082	67.912	61.323
waveform	75.343	75.343	75.349	70.524
wine	90.574	90.557	89.612	86.271
wisconsin-breast-cancer	94.257	94.257	94.144	93.643

Table A.2.: Average classification error on the test data for variants of FURIA.

Data set	prod	min	crisp	w/o
analcatauthorship	2.686	2.686	2.937	2.528
analcatabankruptcy	16.785	16.781	15.530	12.460
analcata-cyyoung8092	17.398	17.367	17.218	15.792
analcata-cyyoung9302	14.828	14.861	14.511	13.197
analcata-esr	17.359	17.359	16.532	15.545
analcata-halloffame	5.720	5.722	5.757	5.160
analcata-lawsuit	1.939	1.939	1.984	1.750
analcata-votesurvey	18.969	18.969	18.969	16.911
biomed	9.424	9.438	9.662	8.172
cars	14.392	14.406	14.479	12.125
collins	2.970	2.958	3.884	0.124
ecoli	13.900	13.908	14.527	12.340
eucalyptus	23.425	23.425	23.633	21.919
glass	23.123	23.137	24.092	18.709
haberman	22.996	22.996	22.919	22.737
heart-statlog	16.254	16.254	16.440	15.654
ionosphere	8.511	8.528	8.645	7.456
iris	4.706	4.706	5.373	4.275
liver-disorders	25.181	25.181	25.377	24.302
metStatCoordinates	6.135	6.142	6.343	5.234
metStatRainfall	20.598	20.599	21.026	16.577
metStatRST	27.899	27.899	28.207	22.689
metStatSunshine	30.615	30.607	30.850	23.062
metStatTemp	25.784	25.788	25.975	22.897
mfeat-factors	4.866	4.872	5.119	4.138
mfeat-fourier	16.025	16.028	16.443	13.050
mfeat-karhunen	8.481	8.481	9.207	6.587
mfeat-morphological	22.510	22.510	22.512	21.504
mfeat-zernike	15.500	15.499	16.157	12.971
optdigits	3.394	3.394	3.640	2.844
page-blocks	2.651	2.652	2.804	2.323
pasture-production	15.125	15.125	16.478	12.536
pendigits	1.691	1.694	2.063	1.449
pima diabetes	20.779	20.779	20.810	20.086
prnn-synth	14.575	14.575	14.621	13.655
schizo-	13.297	13.305	13.780	12.191
segment	2.412	2.409	2.815	1.851
sonar	18.648	18.662	19.226	16.595
squash-unstored	20.369	20.369	20.051	17.383
synthetic control	6.912	6.922	7.961	5.858
vehicle	16.907	16.910	17.129	15.509
vowel	16.826	16.853	20.022	11.633
waveform	14.076	14.076	14.069	12.212
wine	4.358	4.375	5.320	3.105
wisconsin-breast-cancer	3.660	3.660	3.774	3.504

Table A.3.: Average classification accuracy on the training data for variants of FURIA.

Data set	prod	min	crisp	w/o
analcatauthorship	99.580	99.580	99.580	99.537
analcatabankruptcy	97.064	97.064	97.064	96.912
analcata-cyyoung8092	90.537	90.537	90.537	90.272
analcata-cyyoung9302	92.343	92.343	92.343	92.212
analcata-esr	88.247	88.247	88.247	88.059
analcata-halloffame	95.405	95.405	95.379	95.190
analcata-lawsuit	99.105	99.105	99.105	99.099
analcata-votesurvey	24.060	24.060	24.060	24.060
biomed	96.042	96.042	96.056	95.853
cars	87.738	87.738	87.653	87.484
collins	99.730	99.730	99.730	99.427
ecoli	90.764	90.764	90.782	90.503
eucalyptus	61.476	61.476	61.519	61.140
glass	81.522	81.522	81.543	80.814
haberman	72.389	72.389	72.409	72.250
heart-statlog	85.617	85.617	85.634	85.449
ionosphere	96.347	96.347	96.347	96.196
iris	97.869	97.869	97.939	97.768
liver-disorders	72.343	72.343	72.387	71.829
metStatCoordinates	96.766	96.767	96.763	96.459
metStatRainfall	73.551	73.552	73.612	72.427
metStatRST	36.896	36.896	36.918	36.508
metStatSunshine	69.692	69.695	69.717	69.118
metStatTemp	63.474	63.474	63.478	63.219
mfeat-factors	98.977	98.977	98.977	98.936
mfeat-fourier	89.992	89.992	90.008	89.557
mfeat-karhunen	97.185	97.186	97.189	96.932
mfeat-morphological	74.550	74.550	74.580	74.282
mfeat-zernike	82.136	82.137	82.166	81.691
optdigits	99.351	99.351	99.351	99.270
page-blocks	97.863	97.860	97.717	97.747
pasture-production	89.065	89.065	89.065	89.065
pendigits	99.642	99.642	99.643	99.531
pima diabetes	75.019	75.019	75.069	74.489
prnn-synth	87.121	87.121	87.260	86.891
schizo-	88.357	88.357	88.392	88.058
segment	98.831	98.832	98.837	98.648
sonar	95.696	95.696	95.696	95.507
squash-unstored	93.042	93.042	93.042	93.013
synthetic control	99.134	99.134	99.136	98.980
vehicle	74.770	74.770	74.795	74.422
vowel	93.583	93.584	93.607	92.874
waveform	91.319	91.320	91.332	90.222
wine	98.978	98.978	98.978	98.774
wisconsin-breast-cancer	98.298	98.298	98.283	98.236

Table A.4.: Average classification error on the training data for variants of FURIA.

Data set	prod	min	crisp	w/o
analcatauthorship	0.005	0.005	0.005	0.005
analcatabankruptcy	1.610	1.610	1.610	1.761
analcata-cyyoung8092	4.574	4.574	4.574	4.558
analcata-cyyoung9302	2.982	2.982	2.982	3.015
analcata-esr	5.066	5.066	5.066	5.066
analcata-halloffame	1.161	1.161	1.187	0.983
analcatalawsuit	0.597	0.597	0.597	0.597
analcata-votesurvey	2.825	2.825	2.825	2.825
biomed	1.138	1.138	1.124	1.189
cars	3.786	3.786	3.872	3.577
collins	0.000	0.000	0.000	0.000
ecoli	4.424	4.424	4.406	4.450
eucalyptus	10.259	10.259	10.216	10.208
glass	4.538	4.538	4.517	4.517
haberman	17.357	17.357	17.337	17.272
heart-statlog	5.159	5.159	5.142	5.176
ionosphere	1.399	1.399	1.399	1.447
iris	1.242	1.242	1.172	1.273
liver-disorders	10.959	10.959	10.915	11.016
metStatCoordinates	1.891	1.890	1.894	1.924
metStatRainfall	4.297	4.296	4.236	4.334
metStatRST	11.508	11.508	11.485	11.499
metStatSunshine	4.617	4.613	4.592	4.674
metStatTemp	7.765	7.765	7.760	7.769
mfeat-factors	0.055	0.055	0.055	0.061
mfeat-fourier	1.210	1.210	1.194	1.233
mfeat-karhunen	0.291	0.290	0.286	0.317
mfeat-morphological	15.983	15.983	15.953	15.845
mfeat-zernike	4.178	4.177	4.154	4.198
optdigits	0.018	0.018	0.018	0.018
page-blocks	1.394	1.396	1.539	1.144
pasture-production	1.129	1.129	1.129	1.129
pendigits	0.037	0.037	0.037	0.042
pima diabetes	14.120	14.120	14.071	13.767
prnn-synth	7.997	7.997	7.857	8.021
schizo-	2.403	2.403	2.368	2.426
segment	0.225	0.224	0.219	0.230
sonar	0.794	0.794	0.794	0.816
squash-unstored	2.752	2.752	2.752	2.752
synthetic control	0.063	0.063	0.061	0.073
vehicle	5.751	5.751	5.726	5.769
vowel	0.805	0.804	0.781	0.853
waveform	0.332	0.332	0.320	0.342
wine	0.059	0.059	0.059	0.085
wisconsin-breast-cancer	0.427	0.427	0.442	0.436

Table A.5.: Average Classification accuracies and ranks. FURIA-MFC votes for the most frequent class for uncovered instances.

Data set	FURIA	FURIA-EB	FURIA-1NN	FURIA-MFC
analcatauthorship	95.67	95.73	97.21	94.37
analcatabankruptcy	82.57	82.51	82.62	82.10
analcata-cyyoung8092	80.02	79.81	79.93	79.94
analcata-cyyoung9302	82.64	82.65	82.07	82.55
analcata-esr	80.90	80.81	82.00	80.81
analcata-halloffame	92.92	92.88	92.94	92.87
analcatalawsuit	98.00	97.97	97.93	97.94
analcata-votesurvey	36.92	33.81	33.26	36.16
biomed	88.31	88.64	89.08	88.59
cars	79.08	78.37	79.27	77.22
collins	96.35	96.57	96.26	96.02
ecoli	83.12	83.45	84.00	81.52
eucalyptus	60.62	60.06	59.83	59.18
glass	68.22	68.28	69.55	67.27
haberman	72.72	72.63	72.67	72.51
heart-statlog	79.75	79.55	79.45	78.89
ionosphere	89.59	89.38	90.02	89.23
iris	94.76	94.88	95.06	94.00
liver-disorders	67.15	67.16	68.08	67.00
metStatCoordinates	93.02	93.01	93.28	92.38
metStatRainfall	64.51	64.24	69.94	60.62
metStatRST	33.56	33.10	37.81	33.04
metStatSunshine	49.05	48.85	53.82	46.87
metStatTemp	50.71	50.09	56.40	46.62
mfeat-factors	92.09	92.33	94.21	89.29
mfeat-fourier	76.69	76.42	78.62	70.90
mfeat-karhunen	86.47	86.72	90.33	83.20
mfeat-morphological	72.09	72.03	72.16	68.17
mfeat-zernike	73.67	72.86	77.24	68.59
optdigits	94.78	94.82	96.26	93.01
page-blocks	97.02	97.00	97.05	97.00
pasture-production	74.67	75.87	75.33	74.02
pendigits	97.77	97.75	98.20	97.21
pima diabetes	74.71	74.79	74.71	74.37
prnn-synth	83.57	83.65	84.04	83.22
schizo-	80.52	81.77	81.27	81.76
segment	96.50	96.44	96.98	95.72
sonar	77.01	77.15	78.90	76.78
squash-unstored	76.44	76.22	76.74	76.19
synthetic control	89.75	89.81	92.46	85.91
vehicle	70.10	69.50	72.73	66.49
vowel	75.43	75.23	82.69	71.91
waveform	82.24	82.31	81.92	79.01
wine	93.25	93.68	94.80	93.26
wisconsin-breast-cancer	95.68	95.57	95.89	95.04

Table A.6.: Average classification accuracy on the test data for variants of HELLFIRE.

Data set	HELLFIRE	HF. ^I	HF. ^U	HF. ^{I,U}
acd-authorship	91.286	89.713	89.292	84.770
acd-bankruptcy	82.147	81.973	82.025	81.679
acd-cyyoung8092	80.251	79.917	79.519	79.580
acd-cyyoung9302	81.867	80.934	81.131	80.618
acd-esr	78.968	76.941	76.941	76.941
acd-halloffame	90.004	89.680	90.042	89.539
acd-lawsuit	96.848	96.123	97.460	96.123
acd-votesurvey	35.346	34.834	35.795	34.749
biomed	86.479	85.816	84.559	81.859
cars	71.511	73.388	70.909	70.627
collins	93.212	92.977	92.759	92.194
ecoli	79.571	78.285	77.741	75.432
eucalyptus	54.984	55.007	53.224	52.012
glass	63.962	65.321	60.711	59.092
haberman	72.410	72.113	72.381	72.304
heart-statlog	79.455	79.020	75.551	74.571
ionosphere	87.925	87.516	87.065	85.107
iris	94.745	94.725	94.431	94.235
liver-disorders	63.697	62.452	60.165	55.877
metStatCoord.	84.900	83.906	81.424	79.527
metStatRainfall	48.260	46.690	43.700	39.353
metStatRST	38.476	37.701	36.160	34.994
metStatSunshine	41.321	41.129	39.439	38.634
metStatTemp	47.693	47.721	46.509	44.045
mfeat-factors	83.284	78.893	80.407	71.632
mfeat-fourier	72.857	70.571	70.694	65.782
mfeat-karhunen	76.559	70.551	74.832	60.596
mfeat-morpholog.	70.891	70.044	67.360	66.794
mfeat-zernike	63.268	58.599	61.688	52.247
optdigits	82.655	78.917	78.693	69.200
page-blocks	92.054	91.685	91.419	90.956
pasture-prod.	71.735	71.985	71.242	70.832
pendigits	88.248	86.342	86.495	78.644
pima diabetes	72.673	71.991	69.644	68.039
prnn-synth	83.766	82.386	79.814	77.757
schizo-	69.386	70.726	62.805	62.839
segment	90.383	91.374	88.520	88.528
sonar	72.281	71.853	70.440	68.688
squash-unstored	73.128	72.161	72.154	72.092
synthetic control	82.451	86.059	81.569	83.593
vehicle	66.968	67.127	63.172	61.746
vowel	47.539	48.684	46.121	45.371
waveform	77.125	73.045	72.135	60.623
wine	91.811	91.283	91.428	89.929
wisconsin-breast-cancer	94.123	93.244	92.058	88.957

Table A.7.: Average classification error on the test data for variants of HELLFIRE.

Data set	HELLFIRE	HF. ^I	HF. ^U	HF. ^{I,U}
acd-authorship	8.507	9.034	10.502	13.978
acd-bankruptcy	17.853	18.027	17.975	18.321
acd-cyyoung8092	19.717	19.899	20.449	20.236
acd-cyyoung9302	18.102	18.935	18.838	19.250
acd-esr	16.532	18.559	18.559	18.559
acd-halloffame	6.236	6.119	6.198	6.260
acd-lawsuit	2.350	1.972	1.738	1.972
acd-votesurvey	64.408	64.795	63.959	64.879
biomed	13.111	13.336	15.031	17.293
cars	27.592	24.166	28.194	26.927
collins	6.705	6.735	7.158	7.517
ecoli	20.253	18.745	22.083	21.598
eucalyptus	42.745	41.771	44.506	44.766
glass	35.721	34.073	38.972	40.302
haberman	26.713	26.828	26.742	26.637
heart-statlog	20.263	20.610	24.167	25.059
ionosphere	11.312	11.461	12.172	13.870
iris	4.667	4.627	4.980	5.118
liver-disorders	36.133	37.147	39.665	43.722
metStatCoord.	13.744	12.812	17.221	17.190
metStatRainfall	51.588	50.361	56.148	57.698
metStatRST	60.903	60.723	63.219	63.429
metStatSunshine	58.533	57.058	60.415	59.553
metStatTemp	52.066	50.672	53.250	54.349
mfeat-factors	16.691	19.475	19.568	26.735
mfeat-fourier	27.124	27.804	29.287	32.593
mfeat-karhunen	23.422	27.609	25.149	37.565
mfeat-morpholog.	28.903	29.254	32.434	32.504
mfeat-zernike	36.729	39.331	38.309	45.682
optdigits	17.200	19.322	21.162	29.038
page-blocks	5.289	5.364	5.924	6.093
pasture-prod.	28.265	28.015	28.758	29.169
pendigits	11.731	11.928	13.484	19.626
pima diabetes	26.853	27.098	29.882	31.050
prnn-synth	16.140	17.472	20.092	22.102
schizo-	28.704	26.716	35.285	34.604
segment	9.515	7.447	11.378	10.293
sonar	27.634	27.722	29.475	30.887
squash-unstored	23.368	23.822	24.342	23.890
synthetic control	17.510	13.279	18.392	15.745
vehicle	32.986	32.031	36.783	37.412
vowel	52.446	50.443	53.865	53.756
waveform	22.417	26.210	27.407	38.632
wine	7.961	8.307	8.344	9.661
wisconsin-breast-cancer	5.208	5.590	7.273	9.877

Table A.8.: HELLFIRE fuzzification comparison in terms of accuracy. The generic fuzzification ranges from 0 for crisp to 1 for maximally fuzzy.

Data set	HF.	HF. ⁰	HF. ^{0.2}	HF. ^{0.4}	HF. ^{0.6}	HF. ^{0.8}	HF. ¹
acd-authorship	91.398	90.350	90.891	89.265	86.361	83.182	80.318
acd-bankruptcy	82.147	81.973	82.306	82.174	79.671	77.491	75.623
acd-cyyoung8092	80.283	80.101	80.703	80.796	80.637	80.571	80.028
acd-cyyoung9302	81.867	81.001	81.900	82.695	83.137	82.830	82.828
acd-esr	82.621	80.594	82.541	85.123	86.696	86.130	86.139
acd-halloffame	91.508	91.368	91.447	91.438	91.401	91.317	91.247
acd-lawsuit	96.937	96.245	97.493	96.346	95.678	95.031	94.251
acd-votesurvey	35.467	35.018	35.253	35.985	37.441	37.068	37.072
biomed	86.692	86.156	86.113	85.099	84.045	83.463	82.678
cars	71.844	74.307	71.476	70.208	69.354	68.356	67.075
collins	93.260	93.107	92.947	92.164	90.847	89.816	86.755
ecoli	79.668	79.335	80.034	79.858	79.109	77.442	74.888
eucalyptus	55.037	55.363	54.910	54.446	53.798	53.239	52.496
glass	64.045	65.527	63.144	59.645	56.223	53.198	50.513
haberman	72.998	72.816	73.219	73.257	73.257	73.066	73.027
heart-statlog	79.629	79.238	79.651	79.531	78.428	77.840	77.621
ionosphere	88.387	88.128	86.585	86.408	86.174	85.570	84.816
iris	94.824	94.824	94.902	94.686	94.588	92.824	90.765
liver-disorders	63.808	62.691	63.237	62.547	61.720	61.463	61.301
metStatCoord.	85.180	84.585	85.290	85.351	85.140	84.795	84.435
metStatRainfall	48.280	47.098	47.344	45.768	43.708	41.603	39.508
metStatRST	38.546	37.929	38.284	38.311	38.036	37.184	36.340
metStatSunshine	41.356	41.429	42.047	40.927	39.633	38.230	36.251
metStatTemp	47.723	47.857	45.538	41.576	38.255	35.723	33.594
mfeat-factors	83.296	79.335	82.141	82.996	82.546	80.803	78.247
mfeat-fourier	72.862	71.013	72.957	73.594	73.377	72.603	71.379
mfeat-karhunen	76.566	70.959	74.991	76.574	76.574	75.457	73.612
mfeat-morpholog.	70.954	70.259	70.938	70.734	70.277	69.715	68.352
mfeat-zernike	63.268	59.077	61.799	63.219	63.497	62.835	61.424
optdigits	82.690	79.351	79.851	79.727	79.672	79.641	79.446
page-blocks	93.081	92.852	91.362	91.126	90.785	90.700	90.642
pasture-prod.	71.735	71.985	71.658	70.767	70.286	69.632	69.388
pendigits	88.253	86.770	88.091	87.279	85.960	84.448	82.737
pima diabetes	72.922	72.493	72.749	73.060	72.773	72.524	72.022
prnn-synth	83.813	82.445	83.283	84.025	83.860	83.624	82.838
schizo-	70.518	72.204	69.907	68.093	66.616	65.242	64.499
segment	90.398	91.698	89.979	85.161	80.385	76.064	70.888
sonar	72.352	72.137	72.675	72.592	71.121	70.259	69.224
squash-unstored	76.397	75.838	75.436	74.815	74.373	73.817	72.573
synthetic control	82.485	86.402	86.196	83.809	79.956	75.044	70.098
vehicle	66.989	67.385	63.819	58.496	55.846	53.990	52.452
vowel	47.545	48.895	49.407	48.134	46.143	44.026	41.463
waveform	77.313	73.343	74.503	74.903	74.827	74.583	74.187
wine	91.876	91.398	92.321	91.827	91.295	88.155	83.690
w.-breast-cancer	94.506	93.778	94.392	94.418	94.683	94.636	94.456

Table A.9.: HELLFIRE fuzzification comparison in terms of AUC. The generic fuzzification ranges from 0 for crisp to 1 for maximally fuzzy.

Data set	HF.	HF. ⁰	HF. ^{0.2}	HF. ^{0.4}	HF. ^{0.6}	HF. ^{0.8}	HF. ¹
acd-authorship	0.979	0.951	0.979	0.985	0.985	0.983	0.980
acd-bankruptcy	0.880	0.824	0.911	0.924	0.921	0.920	0.917
acd-cyyoung8092	0.784	0.721	0.775	0.807	0.829	0.836	0.838
acd-cyyoung9302	0.769	0.700	0.776	0.804	0.833	0.843	0.854
acd-esr	0.669	0.625	0.713	0.713	0.711	0.706	0.691
acd-halloffame	0.779	0.748	0.766	0.778	0.787	0.797	0.803
acd-lawsuit	0.948	0.926	0.945	0.949	0.978	0.979	0.979
acd-votesurvey	0.543	0.526	0.534	0.537	0.544	0.545	0.549
biomed	0.906	0.867	0.912	0.912	0.913	0.914	0.912
cars	0.853	0.823	0.851	0.863	0.866	0.866	0.866
collins	0.989	0.972	0.990	0.994	0.995	0.995	0.994
ecoli	0.918	0.881	0.910	0.927	0.934	0.936	0.937
eucalyptus	0.815	0.780	0.806	0.822	0.829	0.832	0.833
glass	0.799	0.778	0.793	0.793	0.789	0.778	0.767
haberman	0.632	0.604	0.628	0.632	0.633	0.633	0.630
heart-statlog	0.839	0.831	0.839	0.844	0.858	0.858	0.858
ionosphere	0.899	0.879	0.889	0.887	0.892	0.891	0.884
iris	0.981	0.966	0.989	0.993	0.994	0.994	0.993
liver-disorders	0.657	0.637	0.656	0.649	0.643	0.643	0.643
metStatCoord.	0.965	0.947	0.961	0.968	0.972	0.974	0.975
metStatRainfall	0.822	0.765	0.817	0.830	0.833	0.832	0.829
metStatRST	0.712	0.672	0.709	0.725	0.731	0.735	0.736
metStatSunshine	0.768	0.699	0.764	0.788	0.797	0.799	0.798
metStatTemp	0.765	0.716	0.770	0.769	0.763	0.756	0.749
mfeat-factors	0.971	0.924	0.956	0.970	0.975	0.976	0.974
mfeat-fourier	0.947	0.889	0.925	0.943	0.950	0.952	0.950
mfeat-karhunen	0.958	0.894	0.938	0.956	0.962	0.964	0.963
mfeat-morpholog.	0.914	0.889	0.907	0.918	0.926	0.927	0.931
mfeat-zernike	0.923	0.842	0.891	0.917	0.927	0.930	0.930
optdigits	0.964	0.935	0.951	0.959	0.964	0.967	0.969
page-blocks	0.829	0.821	0.817	0.818	0.819	0.819	0.819
pasture-prod.	0.807	0.791	0.814	0.816	0.817	0.815	0.816
pendigits	0.985	0.960	0.978	0.983	0.984	0.984	0.981
pima diabetes	0.769	0.738	0.764	0.778	0.784	0.784	0.783
prnn-synth	0.889	0.852	0.887	0.905	0.915	0.917	0.919
schizo-	0.749	0.764	0.740	0.728	0.716	0.706	0.695
segment	0.984	0.972	0.986	0.983	0.977	0.968	0.958
sonar	0.785	0.742	0.789	0.798	0.793	0.784	0.778
squash-unstored	0.845	0.803	0.861	0.882	0.888	0.875	0.855
synthetic control	0.971	0.938	0.979	0.984	0.981	0.974	0.964
vehicle	0.856	0.833	0.852	0.835	0.821	0.808	0.798
vowel	0.851	0.759	0.826	0.856	0.869	0.873	0.870
waveform	0.919	0.897	0.905	0.909	0.909	0.907	0.905
wine	0.970	0.939	0.981	0.987	0.989	0.989	0.988
w.-breast-cancer	0.975	0.958	0.971	0.972	0.978	0.979	0.980

Acknowledgements

This thesis is the result of my research that I performed as a PhD student in the Knowledge Engineering & Bioinformatics Lab of the University of Marburg and as a visiting research fellow in the Decision Systems Group at Harvard Medical School. During my scientific journey I have had the distinct honor of becoming acquainted with many people whom I would like to thank for their helpful advice and their friendship.

First of all, I would like to sincerely thank my thesis advisor Prof. Dr. Eyke Hüllermeier for his insight concerning the work in this thesis and his generous support. I am continually grateful for the opportunity to research as a member of his group for which he has always provided ready support and discussion. I value the high scientific standards he constantly sets and the excellent quality of his scientific mentorship.

My gratitude also belongs to Prof. Dr. Johannes Fürnkranz for supervising this thesis and also for many fruitful discussions about separate-and-conquer rule learning. I thank him for his invaluable advice and his friendly support of this thesis.

I thank Ronilda Lacson, M.D., Ph.D., Lucila Ohno-Machado, M.D., Ph.D. and Staal Vinterbo, Ph.D. for having me as a visiting research fellow in the Decision Systems Group at Harvard Medical School. It was an enlightening experience to research with Dr. Vinterbo and I appreciate his scientific genius and esteem his support.

My deep gratitude belongs to the Konrad-Adenauer-Foundation which supported my research through a scholarship both financially and ideally. It has been an honor to be a member of this group of brilliant minds. Moreover, I thank the International Conference on Machine Learning 2009 for supporting me financially.

I want to express my gratefulness to my colleagues at the Knowledge Engineering & Bioinformatics Lab for an enriching time both personally and scientifically. I thank Weiwei Cheng for being a great office neighbor. I also thank Krzysztof Dembczyński, Florian Finkernagel, Thomas Fober, Frederick Kämpfer, Marc Koch, Marco Mernberger, Robin Senge, Dr. Stijn Vanderlooy and Dr. Yu Yi for many constructive discussions and a general great time. I also thank Mechthild Keßler for her support.

I wish to thank my colleagues at Harvard Medical School for the pleasant atmosphere, most notably Kumiko Ohashi, Ph.D., Jihoon Kim and Richard Lu, M.D.. I also thank Dr. Jan Pruszek and Thomas Rost for being great fellows.

I acknowledge Alberto Fernández from the SCI2S Group of the University of Granada for making the CHI and SLAVE program code available. Alberto has been a great sparring partner for discussions about evolutionary fuzzy rule learners.

My gratitude for proofreading this thesis and for giving me advice on using proper English goes to Barbara Güldenring.

I thank my brother Dr. Markus Hühn for his great guidance *behind the scenes* of Academia.

I also thank Susanne Kleinmann for heartily supporting me through all the years of my dissertation.

Last but definitely not least, I wish to express my humble gratitude to my parents Annemarie and Albert Hühn for their unlimited encouragement and support.

Glossary

- AUC** Area under the ROC curve. A ranking performance measure.
- Bias** The assumptions made. Limiting the space of possibilities, which is on the one hand restricting but on the other hand simplifying.
- C4.5** A decision tree algorithm that recursively partitions the data until the class distribution in the leaves is pure enough. C4.5 is fast and is strong in terms of classification accuracy.
- Classification accuracy** A measure for the predictive quality. The relative number of correctly classified instances. Note that the relative number of correctly, incorrectly and unclassified instances sums up to 1.
- Classification error** A measure for the predictive quality. The relative number of incorrectly classified instances. Note that the relative number of correctly, incorrectly and unclassified instances sums up to 1.
- Classification problem** The task of finding a generalizing pattern, which assigns every instance a single class label according to some unknown underlying distribution.
- Conventional** Denoting the fact only two degrees of truths are used: TRUE and FALSE. In contrast to fuzzy.
- Crisp** *see Conventional.*
- Curse of dimensionality** The fact that the problem complexity grows with the number of attributes exponentially.
- Decision boundary** The borderline between two area for which instances become assigned to different classes respectively.

Discretization A gapless partition of numeric values into discrete intervals.

Divide-and-conquer A learning strategy for decision trees. The data is recursively split following a criterion that e.g. tries to reduce the entropy of the split partitions. This process is stopped when the class distribution in the leaves is pure enough. To avoid overfitting pruning techniques can be applied.

FR3 Fuzzy Round Robin RIPPER. An algorithm for inducing a fuzzy preference structure for decision making tasks. It is based on fuzzy rules based on methods from FURIA and RIPPER.

FURIA Fuzzy Unordered Rule Induction Algorithm. A RIPPER-based separate-and-conquer learner. It was modified in several ways and uses a set of rules instead of a list. It uses rule stretching to classify uncovered instance. It contains a data-driven fuzzification method that softens the rule boundaries.

Fuzzification A method to soften conventional rule boundaries. A hard threshold is replaced through a soft and gradual transition from the covered to the uncovered area.

Fuzzy Denoting the fact all shades of truths are used. In contrast to conventional or crisp.

Fuzzy logic A multi-valued logic which does not only distinguish between TRUE and FALSE but between all gradual degrees of truths. Moreover, it uses so-called T-norms and T-conorms as generalizations of the two-valued AND and OR operations. For the negation there are also generalizations of the binary NOT operation.

Genetic fuzzy rule learner A rule learning algorithm that learns fuzzy rules through a search strategy that is inspired by nature selection and evolution.

HELLFIRE High-End Learning of Linguistic Fuzzy Interval Rule Expertise. A linguistic fuzzy rule classifier which learns a data discretization and classification rules simultaneously. The grid-based rules are softened in a second data-driven fuzzification step.

- IREP** Incremental Reduced Error Pruning. A rule pruning technique that simplifies a rule directly after its learning. It is faster than its predecessor REP and it is less vulnerable to local optima.
- Machine learning** The realm of learning generalizing patterns from data.
- Noise** Data points which are not following a given distribution, e.g. due to false attribute values, wrong class labels, etc.
- Occam's Razor** A rule of thumb that a simpler model generalizes better. This rule is often used in the context of model overfitting in noisy domains.
- Overfitting** Learning a model which reproduces the training data — including noise — very precisely, but which generalizes on unseen data poorly.
- Pruning** A strategy to avoid overfitting on noisy data through model simplification.
- REP** Reduced Error Pruning. A post-pruning technique that simplifies a ruleset through deleting rules or conditions.
- RIPPER** Repeated Incremental Pruning to Produce Error Reduction. A separate-and-conquer rule learner using an ordered list of rules. It is known for its small rulesets and its learning speeds.
- ROC** Receiver operating characteristic. A plot of the true-positive-rate vs. the false-positive-rate according to a given ordering of the predictions, typically the score or the (pseudo-) probability.
- Separate-and-conquer** A strategy to learn classification rules for a specific class. The idea is to cover examples, remove those examples and repeat this process until there are basically no examples left. To avoid overfitting pruning techniques can be applied.

List of Tables

2.1.	The data sets used in the experiments.	34
3.1.	Average classification accuracies and ranks for FURIA and its competitors.	58
3.2.	Average AUC and ranks for FURIA and its competitors. . . .	60
3.3.	Wins and losses in terms of classification accuracy and error on the test data for variants of FURIA.	63
3.4.	Wins and losses in terms of classification accuracy and error on the training data for variants of FURIA.	63
3.5.	FURIA model statistics.	68
3.6.	Absolute rule weight calculation frequencies.	70
3.7.	Average model building times in sec. for variants of FURIA. .	72
4.1.	Categorization of entropy-based supervised splitting discretization algorithms.	100
4.2.	Average classification accuracies and ranks for HELLFIRE and its competitors.	101
4.3.	Average AUC and ranks for HELLFIRE and its competitors. . .	103
4.4.	Wins and losses in terms of classification acc. and error on the test data for variants of HELLFIRE.	106
4.5.	Wins and losses in terms of classification acc. for HELLFIRE with different fuzzification degrees.	110
4.6.	Wins and losses in terms of AUC for HELLFIRE with different fuzzification degrees.	112
4.7.	Comparison in terms of number of rules and classification accuracy using RIPPER of discretizations obtained through HELLFIRE (H-Dis) and MDLP.	115
4.8.	HELLFIRE model statistics.	121
4.9.	Average rate of unclassified instances for HELLFIRE.	122
4.10.	The rules induced by HELLFIRE for the glass data set. . . .	123

5.1. Average classification accuracy and ranks for FR3 and its competitors.	143
5.2. Average AUC and ranks for FR3 and its competitors.	145
5.3. FR3 model statistics.	148
5.4. Classification rates (acc) on the test set for different rejection thresholds.	150
5.5. Classification rates on the test set for different rejection thresholds.	151
6.1. Average classification accuracy and AUC of FURIA and HELLFIRE.	157
6.2. Average classification accuracy, AUC and ranks of FURIA, HELLFIRE and FR3.	158
6.3. Contrasting FURIA, HELLFIRE and FR3.	161
A.1. Average classification accuracy on the test data for variants of FURIA.	188
A.2. Average classification error on the test data for variants of FURIA.	189
A.3. Average classification accuracy on the training data for variants of FURIA.	190
A.4. Average classification error on the training data for variants of FURIA.	191
A.5. Average Classification accuracies and ranks.	192
A.6. Average classification accuracy on the test data for variants of HELLFIRE.	193
A.7. Average classification error on the test data for variants of HELLFIRE.	194
A.8. HELLFIRE fuzzification comparison in terms of accuracy. . .	195
A.9. HELLFIRE fuzzification comparison in terms of AUC.	196

List of Figures

2.1.	Decision boundaries of different decomposition schemes. . . .	14
2.2.	Membership representations for concepts of “tall people”. . .	25
2.3.	A fuzzy interval I^F	26
2.4.	An exemplary fuzzy partition with generic linguistic labels for each fuzzy set.	30
2.5.	A sample ROC curve.	38
2.6.	Nemenyi Test visualization.	41
2.7.	Bonferroni-Dunn Test visualization.	42
3.1.	Examination of possible support bounds of a conventional in- terval.	47
3.2.	The fuzzified interval maximizing the fuzzy purity.	48
3.3.	Trivially fuzzified interval.	49
3.4.	Myopic failure of separate-and-conquer learning: Data.	50
3.5.	Myopic failure of separate-and-conquer learning: Optimal con- ventional solution.	51
3.6.	Myopic failure of separate-and-conquer learning: FURIA so- lution without fuzzification.	52
3.7.	Myopic failure of separate-and-conquer learning: FURIA so- lution with fuzzification.	52
3.8.	Three-class problem generalized through three fuzzy rules. . .	55
3.9.	Bonferroni-Dunn Test visualization.	59
3.10.	Bonferroni-Dunn Test visualization.	61
3.11.	Visualization of the decision boundaries.	66
3.12.	Rule weight calculations vs. number of antecedents.	71
4.1.	Attribute splitting.	77
4.2.	A comparison between an ordinary decision tree and the dis- cretization induced by HELLFIRE	79
4.3.	Splitting three data sets in a coordinated way.	81

4.4. Splitting two data sets in a coordinated way.	82
4.5. Splitting three data sets in a coordinated way.	83
4.6. The learning process using coordinated splitting of HELLFIRE for inducing the set of split-based rules RS^S	87
4.7. The transformation of a split-based into an interval-based rule- set.	89
4.8. Describing the concept of puberty using discretization.	90
4.9. A simplified sketch of the interval fuzzification strategy.	93
4.10. Shifted vs. unshifted decision boundaries.	94
4.11. Discretization framework.	99
4.12. Bonferroni-Dunn Test visualization.	102
4.13. Bonferroni-Dunn Test visualization.	104
4.14. Visualization of the decision boundaries.	107
4.15. Different fuzzification degrees for two adjacent fuzzy intervals.	110
4.16. Domination plot for classification accuracy.	111
4.17. Domination plot for AUC.	112
4.18. Comparison of discretizations from MDLP and HELLFIRE.	116
4.19. Distribution of fuzzy sets per interval for HELLFIRE.	117
4.20. Distribution of fuzzy sets per interval for HELLFIRE.	119
4.21. The attributes of the glass data set and the fuzzy intervals induced by HELLFIRE.	124
5.1. Exemplary classification scenario: Different query instance sit- uations.	129
5.2. Perceptron hyperplanes.	134
5.3. Exemplary classification scenario: Rule-based model solution.	135
5.4. The unbound rules in a pairwise model.	137
5.5. Bounding a one-sided fuzzy interval.	138
5.6. Extrapolating a bound fuzzy interval.	139
5.7. Bound and fuzzified rules in a pairwise model.	139
5.8. Graphical illustration of a preference structure predicted.	141
5.9. Bonferroni-Dunn Test visualization.	144
5.10. Bonferroni-Dunn Test visualization.	146
5.11. Accuracy-rejection curves for the data set waveform.	149
6.1. Nemenyi Test visualization.	159
6.2. Nemenyi Test visualization.	159

Bibliography

- [AAFH07] R. Alcalá, J. Alcalá-Fernandez, and F. Herrera. A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. *IEEE Transactions on Fuzzy Systems*, 15(4):616–635, 2007.
- [ACH01] R. Alcalá, J. Casillas, and O. Cordón F. Herrera. Building fuzzy graphs: Features and taxonomy of learning for non-grid-oriented fuzzy rule-based systems. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, 11(3,4):99–119, 2001.
- [ACH03] R. Alcalá, O. Cordón, and F. Herrera. Combining rule weight learning and rule selection to obtain simpler and more accurate linguistic fuzzy models. In J. Lawry, J.G. Shanahan, and A.L. Ralescu, editors, *Modelling with Words*, volume 2873 of *Lecture Notes in Computer Science*, pages 44–63. Springer, 2003.
- [AFSG⁺09] J. Alcalá-Fernandez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, and F. Herrera. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.
- [AKA91] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
- [AN07] A. Asuncion and D.J. Newman. UCI machine learning repository. <http://archive.ics.uci.edu/ml/index.html>, 2007. Obtained on 22th of August 2007.
- [ASS01] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

- [Bar07] D. Barker. Dataset: Pasture production. <http://weka.sourceforge.net/wiki/index.php/Datasets>, 2007. Obtained on 20th of October 2007.
- [BF⁺84] L. Breiman, , J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.
- [BGV92] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory, COLT*, pages 144–152, New York, NY, USA, 1992. ACM.
- [BK07] J. Bacardit and N. Krasnogor. Empirical evaluation of ensemble techniques for a pittsburgh learning classifier system. In Jaume Bacardit, Ester Bernadó-Mansilla, Martin V. Butz, Tim Kovacs, Xavier Llorà, and Keiki Takadama, editors, *IWLCS*, volume 4998 of *Lecture Notes in Computer Science*, pages 255–268. Springer, 2007.
- [BMGG03] E. Bernadó-Mansilla and J.M. Garrell-Guiu. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary Computing*, 11(3):209–238, 2003.
- [Bos04] H. Boström. Pruning and exclusion criteria for unordered incremental reduced error pruning. *Proceedings of the Workshop on Advances in Rule Learning, ECML*, pages 17–29, 2004.
- [Bos05] H. Boström. Maximizing the area under the roc curve using incremental reduced error pruning. In *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning*, 2005.
- [Bos07] H. Boström. Maximizing the area under the ROC curve with decision lists and rule sets. In C. Apte, B. Liu, and S. Parthasarathy, editors, *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 27–34, Minneapolis, MN, USA, 2007. Society for Industrial Mathematics.
- [BP91] C.A. Brunk and M.J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. In L. Birnbaum and

- G. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 389–393, Evanston, IL, USA, 1991. Morgan Kaufmann.
- [Bra97] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
- [Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [BT52] R.A. Bradley and M.E. Terry. The rank analysis of incomplete block designs. i. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.
- [Bul07] B. Bulloch. Dataset: Eucalyptus soil conservation. <http://weka.sourceforge.net/wiki/index.php/Datasets>, 2007. Obtained on 20th of October 2007.
- [BX05] J.F. Baldwin and D. Xie. *Simple fuzzy logic rules based on fuzzy decision tree for classification and prediction problem*, pages 175–184. Springer, London, UK, 2005.
- [Cat91] J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Machine Learning, EWSL*, pages 164–178, London, UK, 1991. Springer.
- [CCdH05] J. Casillas, O. Cordon, M.J. del Jesus, and F. Herrera. Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, 13(1):13–29, 2005.
- [CCH00] O. Casillas, O. Cordon, and F. Herrera. Improving the Wang and Mendel’s fuzzy rule learning method by inducing cooperation among rules. In *Proceedings of the 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference, IPMU*, volume 3, pages 1682–1688, Madrid, Spain, 2000.

-
- [CCZ04] P. Carmona, J.L. Castro, and J.M. Zurita. FRIwE: fuzzy rule identification with exceptions. *IEEE Transactions on Fuzzy Systems*, 12(1):140–151, 2004.
- [CdH99] O. Cordón, M.J. del Jesus, and F. Herrera. A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximative Reasoning*, 20(1):21–45, 1999.
- [CGH⁺04] O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, 2004.
- [CjH02] I-J. Chiang and J.Y. j. Hsu. Fuzzy classification trees for data analysis. *Fuzzy Sets and Systems*, 130(1):87–99, 2002.
- [CL92] K.J. Cios and N. Liu. A machine learning method for generation of a neural network architecture: a continuous ID3 algorithm. *IEEE Transactions on Neural Networks*, 2(3):280–291, 1992.
- [CN89] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
- [Coh60] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960.
- [Coh93] W.W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 988–994, Chambéry, France, 1993. Morgan Kaufmann.
- [Coh95] W.W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the 12th International Conference on Machine Learning, ICML*, pages 115–123, Tahoe City, CA, USA, July 9–12, 1995. Morgan Kaufmann.
- [CS92] K.J. Cios and L.M. Sztandera. Continuous ID3 algorithm with fuzzy entropy measures. In *IEEE International Conference on Fuzzy Systems*, pages 469–476, San Diego, CA, USA, 1992.

- [CS99] W.W. Cohen and Y. Singer. A simple and fast and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence, AAAI*, pages 335–342, Orlando, FL, USA, 1999. American Association for Artificial Intelligence.
- [CTLL91] K.J. Cios, R. Tjia, N. Liu, and R.A. Langenderfer. Study of continuous ID3 and radial basis function algorithms for recognition of defects in glass. In *International Joint Conference on Neural Networks, IJCNN*, volume 1, pages 49–54, Seattle, 1991.
- [Cut03] F. Cutzu. Polychotomous classification with pairwise classifiers: A new voting principle. In *Multiple Classifier Systems*, pages 115–124, 2003.
- [Cv04] I. Cloete and J. van Zyl. Evaluation function guided search for fuzzy set covering. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE*, volume 2, pages 1007–1012, Budapest, Hungary, 2004. IEEE Computer Society.
- [Cv06] I. Cloete and J. van Zyl. Fuzzy rule induction in a set covering framework. *IEEE Transactions on Fuzzy Systems*, 14(1):93–110, 2006.
- [CW03] Y. Chen and J.Z. Wang. Support vector learning for fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 11(6):716–728, 2003.
- [CWY95] Z. Chi, J. Wu, and H. Yan. Handwritten numeral recognition using self-organizing maps and fuzzy rules. *Pattern Recognition*, 28(1):59–66, 1995.
- [CYP96] Z. Chi, H. Yan, and T. Pham. *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.
- [DB95] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

-
- [DBK03] M. Drobics, U. Bodenhofer, and E.P. Klement. FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. *International Journal of Approximative Reasoning*, 32(2–3):131–152, 2003.
- [DCB04] O. Dain, R. Cunningham, and S. Boyer. IREP++, a faster rule learning algorithm. In M.W. Berry, U. Dayal, C. Kamath, and D.B. Skillicorn, editors, *Proceedings of the Fourth SIAM International Conference on Data Mining, SDM*, Lake Buena Vista, FL, USA, 2004. Society for Industrial & Applied Mathematics.
- [Dem06] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
- [Den95] T. Dencœux. A k-nearest neighbor classification rule based on Dempster-Shafer Theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5):804–813, 1995.
- [dHNS04] M.J. del Jesus, F. Hoffmann, L.J. Navascues, and L. Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems*, 12(3):296–308, 2004.
- [Dom96] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.
- [Dun61] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.
- [EB01] M. Eineborg and H. Boström. Classifying uncovered examples by rule stretching. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming, ILP*, pages 41–50, London, UK, 2001. Springer.
- [EGdE09] A.G. Evsukoff, S. Galichet, B.S.L.P. de Lima, and N.F.F. Ebecken. Design of interpretable fuzzy rule-based classifiers using spectral analysis with structure and parameters optimization. *Fuzzy Sets and Systems*, 160(7):857–881, 2009.
- [ER99] T. Elomaa and J. Rousu. General and efficient multisplitting of numerical attributes. *Machine Learning*, 36(3):201–244, 1999.

- [Faw01] T. Fawcett. Using rule sets to maximize ROC performance. In N. Cercone, T.Y. Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM*, pages 131–138, Washington, DC, USA, 2001. IEEE Computer Society.
- [Faw08] T. Fawcett. PRIE: a system for generating rulelists to maximize roc performance. *Data Mining and Knowledge Discovery*, 17(2):207–224, 2008.
- [Fei80] E.A. Feigenbaum. Expert Systems: Looking Back and Looking Ahead. In *GI — 10. Jahrestagung*, pages 1–14, London, UK, 1980. Springer, Berlin, Germany.
- [FF05] J. Fürnkranz and P.A. Flach. Roc ‘n’ rule learning—towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.
- [FGHd07] A. Fernández, S. García, F. Herrera, and M.J. del Jesus. An analysis of the rule weights and fuzzy reasoning methods for linguistic rule based classification systems applied to problems with highly imbalanced data sets. In *Applications of Fuzzy Sets Theory*, volume 4578 of *Lecture Notes in Computer Science*, pages 170–178. Springer, Berlin / Heidelberg, Germany, 2007.
- [FI93] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambéry, France, 1993. Morgan Kaufmann.
- [FJ09] S.M. Fakhrahmad and M.Z. Jahromi. A new rule-weight learning method based on gradient descent. In S.I. Ao, L. Gelman, D.W.L. Hukins, A. Hunter, and A.M. Korsunsky, editors, *Proceedings of the World Congress on Engineering, WCE*, volume 1, London, UK, 2009. International Association of Engineers.
- [Fod94] J. Fodor. Valued preference structures. *European Journal of Operational Research*, 79:277–286, 1994.

-
- [FR94] J. Fodor and M. Roubens. *Fuzzy preference modelling and multicriteria decision support*. Kluwer Academic Publishers, 1994.
- [Fri37] M. Friedman. 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, 1937.
- [Fri40] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.
- [Fri96] J.H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.
- [FS99] Y. Freund and R.E. Schapire. A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.
- [Für97] J. Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.
- [Für99] J. Fürnkranz. Separate-and-Conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.
- [Für02] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.
- [Für03] J. Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–403, 2003.
- [FW94] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In W.W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning, ICML*, pages 70–77, New Brunswick, NJ, USA, 1994. Morgan Kaufmann.
- [FW98] E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In J.W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning, ICML*, pages 144–151, San Francisco, CA, USA, 1998. Morgan Kaufmann.

-
- [GFLH09] S. García, A. Fernández, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, 13(10):959–977, 2009.
- [Gin21] C. Gini. Measurement of inequality of income. *Economic Journal*, 31:22–43, 1921.
- [Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, USA, 1989.
- [GP99] A. González and R. Perez. Slave: a genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.
- [GP01] A. González and R. Perez. Selection of relevant features in a fuzzy genetic learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(3):417–425, 2001.
- [HAP89] R.C. Holte, L.E. Acker, and B.W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818, Detroit, MI, USA, 1989. Morgan Kaufmann.
- [Har07] W. Harvey. Dataset: Squash harvest stored / unstored. <http://weka.sourceforge.net/wiki/index.php/Datasets>, 2007. Obtained on 20th of October 2007.
- [HB08] E. Hüllermeier and K. Brinker. Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems*, 159(18):2337–2352, 2008.
- [HC00] T.P. Hong and J.B. Chen. Processing individual fuzzy attributes for fuzzy rule induction. *Fuzzy Sets and Systems*, 112(1):127–140, 2000.
- [Hel70] M.E. Hellman. The nearest neighbor classification rule with a reject option. *Transactions on Systems, Man, and Cybernetics*, SMC-6:179–185, 1970.

- [HH08] J.C. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. In L. Magdalena, M. Ojeda-Aciego, and J.L. Verdegay, editors, *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU*, pages 1543–1550, Torremolinos (Málaga), Spain, 2008.
- [HH09a] J.C. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems*, 17(1):138–149, 2009.
- [HH09b] J.C. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, 2009.
- [HH10] J.C. Hühn and E. Hüllermeier. An analysis of the FURIA algorithm for fuzzy rule induction. In J. Koronacki, Z. Ras, S.T. Wierzbach, and J. Kacprzyk, editors, *Advances in Machine Learning I: Dedicated to the memory of Professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*. Springer, Berlin, Germany, 2010.
- [HK04] F. Höppner and F. Klawonn. Learning fuzzy systems – an objective-function approach. *Mathware and Soft Computing Journal*, 11(5):143–162, 2004.
- [HKKR99] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. Wiley, 1999.
- [HL05] J. Huang and C.X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.
- [HM82] J. A. Hanley and B. J. Mcneil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.
- [Hof01] F. Hoffmann. Boosting a genetic fuzzy classifier. In *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference of NAFIPS and IFSA*, volume 3, pages

- 1564–1569, Vancouver, BC, Canada, 2001. IEEE Computer Society.
- [HR78] J.H. Holland and J.S. Reitman. Cognitive systems based on adaptive algorithms. In D.A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*, pages 313–329. Academic Press, New York, NY, USA, 1978.
- [HT97] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *NIPS*, Denver, CO, USA, 1997. The MIT Press.
- [Hül03] E. Hüllermeier. Possibilistic instance-based learning. *Artificial Intelligence*, 148(1–2):335–383, 2003.
- [Hül05] E. Hüllermeier. Fuzzy-methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems*, 156(3):387–407, 2005.
- [HV09] E. Hüllermeier and S. Vanderlooy. Why fuzzy decision trees are good rankers. *IEEE Transactions on Fuzzy Systems*, 2009.
- [HV10] E. Hüllermeier and S. Vanderlooy. Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, 43(1):128–142, 2010.
- [HVor] J.C. Hühn and S.A. Vinterbo. HELLFIRE: Learning interpretable and effective fuzzy rule-based classification models. *Fuzzy Sets and Systems*, 2009 – with editor.
- [HY08] J. Hongxia and H. Yao. Classroom teaching quality evaluation based on neuro-fuzzy ID3 algorithm. In B. Werner, editor, *Proceedings of the 2008 International Symposium on Computational Intelligence and Design, ISCID*, pages 166–169, Washington, DC, USA, 2008. IEEE Computer Society.
- [ID80] R.L. Iman and J.M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, 9(6):571–595, 1980.

-
- [IN01] H. Ishibuchi and T. Nakashima. Effect of rule weights in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 9(4):506–515, 2001.
- [INN05] H. Ishibuchi, T. Nakashima, and M. Nii. *Classification and Modeling with Linguistic Information Granules*. Advanced Information Processing. Springer, Berlin / Heidelberg, Germany, 2005.
- [ISNM96] H. Ichihashi, T. Shirai, K. Nagasaka, and T. Miyoshi. Neuro-fuzzy ID3: a method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning. *Fuzzy Sets and Systems*, 81(1):157–167, 1996.
- [IY02] H. Ishibuchi and T. Yamamoto. Fuzzy rule selection by data mining criteria and genetic algorithms. In W. B. Langdon, E. Cantú-Paz, K.E. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V.G. Honavar, G. Rudolph, I. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*, pages 399–406, San Francisco, CA, USA, 2002. Morgan Kaufmann.
- [IY04] H. Ishibuchi and T. Yamamoto. 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, 2004.
- [IY05] H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 13(4):428–436, 2005.
- [Jan98] C.Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(1):1–14, 1998.
- [JF09] F. Janssen and J. Fürnkranz. A re-evaluation of the over-searching phenomenon in inductive rule learning. In *SDM*, pages 329–340, Sparks, NV, USA, 2009. SIAM.

-
- [JJL97] B. Jeng, Y.-M. Jeng, and T.-P. Liang. Film: a fuzzy inductive learning method for automated knowledge acquisition. *Decision Support Systems*, 21(2):61–73, 1997.
- [JT08] M.Z. Jahromi and M. Taheri. A proposed method for learning rule weights in fuzzy rule-based classification systems. *Fuzzy Sets and Systems*, 159(4):449–459, 2008.
- [Kea88] M. Kearns. Thoughts on hypothesis boosting. ML class project, 1988.
- [KK02] M. Kukar and I. Kononenko. Reliable classifications with machine learning. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the European Conference on Machine Learning, ECML*, pages 219–231, Berlin / Heidelberg, Germany, 2002. Springer.
- [KNI08] I. Kuwajima, Y. Nojima, and H. Ishibuchi. Effects of constructing fuzzy discretization from crisp discretization for rule-based classifiers. *Artificial Life and Robotics*, 13(1):294–297, 2008.
- [Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
- [LF03] N. Lachiche and P.A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning, ICML*, pages 416–423, Washington, DC, USA, 2003. AAAI Press.
- [LHTD02] H. Liu, F. Hussain, C.L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.
- [LHZ03] C.X. Ling, J. Huang, and H. Zhang. AUC: A better measure than accuracy in comparing learning algorithms. In Y. Xiang and B. Chaib-draa, editors, *Advances in Artificial Intelligence, 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI*, Lecture Notes in Computer Science, pages 329–341, Halifax, Canada, 2003. Springer.

- [LYL⁺06] C. Lin, C. Yeh, S. Liang, J. Chung, and N. Kumar. Support-vector-based fuzzy neural network for pattern classification. *IEEE Transactions on Fuzzy Systems*, 14(1):31–41, 2006.
- [MD08] T. Maszczyk and W. Duch. Comparison of Shannon, Renyi and Tsallis entropy used in decision trees. In L. Rutkowski, R. Tadeusiewicz, L.A. Zadeh, and J.M. Zurada, editors, *Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing, ICAISC*, volume 5097 of *Lecture Notes in Computer Science*, pages 643–651, Zakopane, Poland, 2008. Springer.
- [MH00] S. Mitra and Y. Hayashi. Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Transactions on Neural Networks*, 11(3):748–768, 2000.
- [Mic69] R.S. Michalski. On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing, FCIP*, volume A3 (Switching Circuits), pages 125–128, Bled, Yugoslavia, 1969.
- [Mic73] R.S. Michalski. AQVAL/1-Computer implementation of a variable-valued logic system and the application to pattern recognition. In K.S. Fu, editor, *Proceedings of the First International Joint Conference on Pattern Recognition*, pages 3–17, Washington, DC, USA, 1973. IEEE Computer Society.
- [Mil56] G.A. Miller. The magical number seven, plus or minus two. *The Psychological Review*, 63(2):81–97, 1956.
- [Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [MJG05] R. Mikut, J. Jäkel, and L. Gröll. Interpretability issues in data-based learning of fuzzy systems. *Fuzzy Sets and Systems*, 150(2):179–197, 2005.
- [MTP08] N.E. Mitrakis, J.B. Theocharis, and V. Petridis. A multilayered neuro-fuzzy classifier with self-organizing properties. *Fuzzy Sets and Systems*, 159(23):3132–3159, 2008.

- [MV07] M. Meyer and P. Vlachos. Statlib. <http://lib.stat.cmu.edu/>, 2007.
- [MW47] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.
- [MZK08] E.G. Mansoori, M.J. Zolghadri, and S.D. Katebi. SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data. *IEEE Transactions on Fuzzy Systems*, 16(4):1061–1071, 2008.
- [MZR08] S. Marcellin, D.A. Zighed, and G. Ritschard. Evaluating decision trees grown with asymmetric entropies. In A. An, S. Matwin, Z.W. Ras, and D. Slezak, editors, *Proceedings of the 17th International Symposium on Foundations of Intelligent Systems, ISMIS 2008*, volume 4994 of *Lecture Notes in Computer Science*, pages 58–67, Toronto, Canada, 2008. Springer.
- [Nem63] P. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.
- [New39] D. Newman. The distribution of range in samples from a normal population, expressed in terms of an independent estimate of standard deviation. *Biometrika*, 31:20–30, 1939.
- [NIT96] K. Nozaki, H. Ishibuchi, and H. Tanaka. Adaptive fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 4(3):238–250, 1996.
- [NK98] D. Nauck and R. Kruse. How the learning of rule weights affects the interpretability of fuzzy systems. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE*, volume 2, pages 1235 – 1240, Anchorage, AU, 1998. IEEE Computer Society.
- [NKK97] D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, UK, 1997.
- [NSYI07] T. Nakashima, G. Schaefer, Y. Yokota, and H. Ishibuchi. A weighted fuzzy classifier and its application to image processing tasks. *Fuzzy Sets and Systems*, 158(3):284–294, 2007.

- [OBG06] K.M. Osei-Bryson and K. Giles. Splitting methods for decision tree induction: An exploration of the relative performance of two entropy-based families. *Information Systems Frontiers*, 8(3):195–209, 2006.
- [OPCBM09] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Fuzzy-UCS: A Michigan-style fuzzy-learning classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, 13(2):260–283, 2009.
- [OW03] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221–254, 2003.
- [PD03] F.J. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.
- [PF97] F.J. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In D. Heckerman, H. Mannila, and D. Pregibon, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, KDD*, pages 43–48, Newport Beach, CA, USA, 1997. AAAI Press.
- [PF01a] Y. Peng and P. Flach. Soft discretization to enhance the continuous decision tree induction. In C. Giraud-Carrier, N. Lavrac, and S. Moyle, editors, *Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 109–118. ECML/PKDD’01 workshop notes, 2001.
- [PF01b] F.J. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
- [PF05] R.C. Prati and P.A. Flach. ROCCER: An algorithm for rule learning based on roc analysis. In L.P. Kaelbling and A. Saffioti, editors, *International Joint Conference on Artificial Intelligence, IJCAI*, pages 823–828, Edinburgh, Scotland, UK, 2005. Professional Book Center.
- [PF07] S.H. Park and J. Fürnkranz. Efficient pairwise classification. In J.N. Kok, J. Koronacki, R.L. Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Proceedings of the 18th*

- European Conference on Machine Learning, ECML*, pages 658–665, Warsaw, Poland, 2007. Springer.
- [PFK98] F.J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In J.W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning, ICML*, pages 445–453, Madison, WI, USA, 1998. Morgan Kaufmann.
- [PFTV92] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 2nd edition, 1992.
- [PH90] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.
- [PRS03] H. Prade, G. Richard, and M. Serrurier. Enriching relational learning with fuzzy predicates. In N. Lavrac, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD*, pages 399–410, Cavtat-Dubrovnik, Croatia, 2003. Springer.
- [QCJ93] J.R. Quinlan and R.M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the 6th European Conference on Machine Learning, ECML*, pages 3–20, London, UK, 1993. Springer.
- [QL08] Z. Qin and J. Lawry. LFOIL: Linguistic rule induction in the label semantics framework. *Fuzzy Sets and Systems*, 159(4):435–448, 2008.
- [Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [Qui90] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.
- [Qui93] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993.

- [Qui95] J.R. Quinlan. MDL and categorial theories (continued). In A. Prieditis and S.J. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning, ICML*, pages 464–470, Lake Tahoe, CA, USA, 1995. Morgan Kaufmann.
- [QuM07] B. Quost, T. Denc  ux, and M.-H. Masson. Pairwise classifier combination using belief functions. *Pattern Recognition Letters*, 28(5):644–653, 2007.
- [Ris83] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.
- [RK04] R.M. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
- [Rk05] L. Rutkowski and K. Cpa  ka. Designing and learning of adjustable quasi-triangular norms with applications to neuro-fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 13(1):140–151, 2005.
- [Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- [Sal91] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6(3):251–276, 1991.
- [Sal97] S. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328, 1997.
- [Sha48] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.
- [She07] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 4 edition, 2007.
- [SJ06] D.A. Simovici and S. Jaroszewicz. Generalized conditional entropy and a metric splitting criterion for decision trees. In W.K. Ng, M. Kitsuregawa, J. Li, and K. Chang, editors, *Proceedings of the 10th Pacific-Asia Conference on Advances in*

- Knowledge Discovery and Data Mining, PAKDD*, volume 3918 of *Lecture Notes in Computer Science*, pages 35–44, Singapore, 2006. Springer.
- [SL99] A. Suárez and J.F. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999.
- [Smi80] F.S. Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA, 1980.
- [SP07] M. Serrurier and H. Prade. Introducing possibilistic logic in ILP for dealing with exceptions. *Artificial Intelligence*, 171(16–17):939–950, 2007.
- [Sur00] H. Surmann. Learning a fuzzy rule based knowledge representation. In H. Bothe and R. Rojas, editors, *Proceedings of the Second ICSC Symposium on Neural Computation*, pages 349–355, Berlin / Heidelberg, Germany, 2000. Springer.
- [TC93] H. Theron and I. Cloete. An empirical evaluation of beam search and pruning in BEXA. In *Fifth International Conference on Tools with Artificial Intelligence, ICTAI*, pages 132–139, Boston, MA, USA, 1993. IEEE Computer Society.
- [TC96] H. Theron and I. Cloete. BEXA: A covering algorithm for learning propositional concept descriptions. *Machine Learning*, 24(1):5–40, 1996.
- [TEF08] M.G. Tsipouras, T.P. Exarchos, and D.I. Fotiadis. A methodology for automated fuzzy model generation. *Fuzzy Sets and Systems*, 159(23):3201–3220, 2008.
- [vC04] J. van Zyl and I. Cloete. Fuzzconri - a fuzzy conjunctive rule inducer. *Proceedings of the Workshop on Advances in Rule Learning, ECML*, pages 17–29, 2004.
- [Ven93] G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In

- P. Brazdil, editor, *Proceedings of the European Conference on Machine Learning, ECML*, pages 280–296, London, UK, 1993. Springer.
- [VGS03] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer, 2003.
- [VKOM05] S.A. Vinterbo, E.-Y. Kim, and L. Ohno-Machado. Small, fuzzy and interpretable gene expression based classifiers. *Bioinformatics*, 21(9):1964–1970, 2005.
- [WCQY00] X. Wang, B. Chen, G. Qian, and F. Ye. On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems*, 112(1):117–125, 2000.
- [WD94] D. Wettschereck and T.G. Dietterich. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In F. Bergadano and L. de Raedt, editors, *Proceedings of the European Conference on Machine Learning, ECML*, pages 323–335, Catania, Italy, 1994. Springer.
- [WF05] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, USA, 2nd edition, 2005.
- [Wil45] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, 1945.
- [Wil95] S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computing*, 3(2):149–175, 1995.
- [WLW04] T.-F. Wu, C.-H. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.
- [WM92] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, 1992.
- [YS95] Y. Yuan and M.J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2):125–139, 1995.

-
- [Zad65] L.A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.
- [ZBHH08] J. Zhang, J.W. Bala, A. Hadjarian, and B. Han. Learning to rank cases with classification rules. Preference Learning ECML/PKDD-08 Workshop, 2008.
- [ZC06] J. Zhao and Z. Chang. Neuro-fuzzy decision tree by fuzzy ID3 algorithm and its application to anti-dumping early-warning system. In *International Conference on Information Acquisition*, pages 1300–1304, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
- [ZG08] S. Zhou and J.Q. Gan. Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. *Fuzzy Sets and Systems*, 159(23):3091–3131, 2008.
- [ZM07] M.J. Zolghadri and E.G. Mansoori. Weighting fuzzy classification rules using receiver operating characteristics (roc) analysis. *Information Sciences: an International Journal*, 177(11):2296–2307, 2007.

Erklärung

Ich versichere, dass ich meine Dissertation

Induction and Fuzzification of Classification Rules

selbständig, ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen und Hilfen bedient habe.

Die Dissertation wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Resume

- 02/2009 – 05/2009 Visiting Research Fellow
Decision Systems Group, Harvard Medical School /
Brigham and Women's Hospital, Boston, MA, USA
Research advisor: Dr. Staal A. Vinterbo
- 04/2007 – 12/2009 Scholarship from the Konrad-Adenauer-Foundation
- 11/2006 – 12/2009 Doctoral studies:
Computer Sciences
Department of Mathematics and Computer Sciences
Philipps-Universität Marburg, Germany
Thesis advisor: Prof. Dr. Eyke Hüllermeier
- 10/2000 – 06/2006 Undergraduate/Graduate studies:
Computer Sciences and Business Administration
Department of Mathematics and Computer Sciences
Philipps-Universität Marburg, Germany
Degree: Diploma (roughly equivalent to M.Sc. C.S.)
- 08/1991 – 06/2000 High school:
Alfred-Wegener-Schule, Kirchhain, Germany
Degree: Abitur (university entrance qualification)

Index

0–9

1-vs-1 *see* All-vs-All
 1-vs-All **11**, **13–14**, 38, 44, 45, 71,
 177, 185
 1-vs-Rest 45

A

AdaBoost 164, 173, 185
 All-vs-All **11–14**, 131, 133,
 135–141, 185
 AQ 164
 Area under the ROC curve....*see*
 AUC
 AUC **36–38**,
 58, 62, 102, 104, 110–112,
 144–146, 166–167

B

BEXA 164–165, 170
 Black box 30

C

C4.5 **35**, 57, 62, 98, 142, 165
 CHI . 32, **33–35**, 57, 67, 100, 102,
 104, 142, 168
 Classification **9–11**
 accuracy 21, **36**, 54,
 57–58, 62, 64–66, 69, 83,
 91, 95, 100–102, 105–110,
 142–144, 146, 147

error . **36**, 62, 64–66, 105–109
 rate *see* Classification
 accuracy
 rule **15**, 17, 44, 76, 77, 84, 88,
 106, 136, 165, 167

CN2 **164**, 170
 Cohen’s Kappa 171
 Concession split 81–82
 Confidence factor .*see* Rule weight
 Conflict... 128–129, 132, 134, 135,
 140–141, 147–149

D

De Morgan triplet 28
 Decision tree 76–79, 81, 84–86, 98,
 99, 165–166, 181, 182

E

ECOC 14
 Error Correcting Output Codes*see*
 ECOC
 Evolutionary Algorithm.. 171–174

F

FCT 176
 FOIL **17–19**, 43, 45, 164, 170
 FR3 **127–149**, 182, 184–185
 FS-FOIL 170

- FURIA **43**–
73, 75–76, 104, 127, 136,
 164, 180–182, 184–185
- FuzzConRi 170
- Fuzzification 45, **46–52**, 54,
 61–67, 69, 71, 73, 75, 76,
 90, **91–95**, 97, 104–113,
 117, **137–138**, 146, 167,
 177, 184
- Fuzzy
- UCS 173
 - classification rule **29**, 88, 120,
 172
 - decision tree 176–177
 - first-order logic 170
 - ID3 **176**
 - intersection 28
 - interval **25–26**, 46,
 49, 56, 62, 64, 65, 91–94,
 109, 111, 112, 117, 119,
 120, 123, 146
 - logic **23–28**, 186
 - membership function. **23–25**,
 64, 92, 172, 175, 178, 179
 - negation 27, 131
 - operator 26–28, 172
 - partition .. **26**, 30, 88, 92, 98,
 123, 168, 176, 177, 181
 completeness 26
 - preference structure 127–134,
 141, 182
 - propositional Logic 170
 - reasoning methods 31
 - Round Robin RIPPER... *see*
 FR3
 - rule .. *see* Fuzzy classification
 rule
 - rule-based classification....2,
22–32, 38, 168–178
 - Ruspini partition 26, 31
 - set **23–26**, 109, 116–118, 120,
 123
 continuity 25
 convexity 25
 normalization..... 25
 - set covering 170–171
 - T-conorm . **27**, 30, 92, 94, 96,
 106, 131, 139, 174
 - T-norm ... **27**, 30, 53, 55, 63,
 65–66, 92, 131, 174
 - union 28
 - Unordered Rule Induction Al-
 gorithm..... *see*
 FURIA
- FUZZYBEXA..... 170
- G**
- Genetic fuzzy system 171
 - Gini Index 78
 - Grid partition *see* Fuzzy partition
- H**
- HELLFIRE **75–124**, 181–182, 185
- I**
- ID3 98, 99, 164
 - Ignorance. 128–129, 132, 134, 135,
 140–141, 147–149
 - Incomparability 130, 132
 - Incremental Reduced Error Prun-
 ing *see*
 IREP
 - Indifferentiability 130, 132
 - Information Gain **18**, 49
 - Interval fuzzification *see*
 Fuzzification

IREP...**19**, 20, 21, 43, 44, 71, 73,
147, 164

IREP* **21**, 45–46, 67, 164

IREP++ 164

Iterative genetic rule learning. 173

K

KEEL 32, 35

L

Language bias 16

Learning bias 16–17

Linguistic fuzzy classification rule .. **29–31**, 77, 116–122,
see also Fuzzy classification rule

M

MDL 19, 21, 46, 98

Michigan genetic fuzzy rule learning 173

Minimum Description Length . *see* MDL

MOGA 172

MOGLS 172

Multi-objective genetic algorithm
see MOGA

Multi-objective genetic local search
see MOGLS

Myopia 50–52, 64

N

Nearest Neighbor 167

Neural network 174

Non-trivial fuzzification 48, 50

O

One-vs-All *see* 1-vs-All

One-vs-Rest *see* 1-vs-Rest

Overfitting 18, 31, 46, 166

Overfitting avoiding bias 17

P

Pairwise learning ... *see* All-vs-All

PART 165

Perceptron 133, 134

Pittsburgh genetic fuzzy rule learning 172–173

Preference structure 128

Problem decomposition 11–14

Pruning . 19–20, 45–46, 67, 71, 82,
90–91, 97, 119, 147, 164,
165

R

R3 142, 147

Ranking performance ... *see* AUC

Receiver Operating Characteristic
see ROC

Reduced Error Pruning . *see* REP

REP **19**, 43

RIPPER **20–21**, 33,
43–46, 49, 50, 53, 67, 69–
73, 75–76, 100, 102, 104,
136, 142, 164

RISE **167–168**, 180–181

ROC **36–38**, 58, 102, 166

Round robin learning *see* All-vs-All

Round Robin RIPPER *see* R3

Rule

Fuzzification *see* Fuzzification
stretching 44, 45, **55–57**,
62, 67–69, 71, 73, 96–97,
105, 120, 136

weight 177–178

Rule-based classification 2, **15–21**

S

- SDT 176
- Search bias 16–17
- Separate-and-conquer **15–21**,
163–165
- SGERD 173
- Shannon’s entropy 78
- SLAVE .. 32, **35**, 57, 67, 100, 102,
104, 142
- SLIPPER 164
- Stopping criterion 20–21
- Strict preference 130, 132, 140
- Support Vector Machine . 174, 175

T

- Takagi-Sugeno rule 168
- Trivial fuzzification 48

U

- U-IREP 164

V

- Valued Preference Structure .. *see*
Fuzzy Preference Structure

W

- Wang-Mendel classifier ... 30, **168**
- Weak preference 128, 130, 132, 133
- WEKA 32–33, 57, 100, 142

X

- XCS 173

