An Analysis of the FURIA Algorithm for Fuzzy Rule Induction

Jens Christian Hühn and Eyke Hüllermeier

Department of Mathematics and Computer Science
University of Marburg
Hans-Meerwein-Straße
35039 Marburg
Germany
huehnj@informatik.uni-marburg.de, eyke@informatik.uni-marburg.de

Abstract. This paper elaborates on a novel fuzzy rule-based classification method called FURIA, which is short for "Fuzzy Unordered Rule Induction Algorithm". FURIA has recently been developed as an extension of the well-known RIPPER algorithm. It learns fuzzy rules instead of conventional rules and unordered rule sets instead of rule lists. Moreover, to deal with uncovered examples, it makes use of an efficient rule stretching method. First experimental results have shown that FURIA significantly outperforms the original RIPPER in terms of classification accuracy. Elaborating on the advantages of a fuzzy approach, this paper makes an attempt to distill and quantify the influence of rule fuzzification on the performance of the algorithm. Moreover, going beyond the conventional classification problem, we investigate the performance of FURIA in the context of bipartite ranking, in which a fuzzy approach appears to be even more appealing.

1 Introduction

The interest in learning rule-based models for classification goes far beyond the field of machine learning itself and also includes other research areas, notably fuzzy systems (Hül05). This is hardly surprising, given that rule-based models have always been a cornerstone of fuzzy systems and a central aspect of research in that field. To a large extent, the popularity of rule-based models can be attributed to their comprehensibility, a distinguishing feature and key advantage in comparison to many other (black-box) classification models. Despite the existence of many sound algorithms for rule induction, the field still enjoys great popularity and, as shown by recent publications (IY05; Cv06; JCC07; FGHd07), offers scope for further improvements.

This paper investigates some properties of a novel fuzzy rule-based classification method called *Fuzzy Unordered Rule Induction Algorithm*, or FURIA for short, which was recently introduced in (HH09). FURIA is a fuzzy rule learner that builds upon RIPPER, a state-of-the-art rule induction method developed by Cohen (Coh95). The main modifications of RIPPER include changes in the pruning procedure, the learning of an unordered instead of an ordered rule set, a novel rule stretching mechanism and, perhaps most importantly, a fuzzification strategy that turns conventional rules into fuzzy rules with soft boundaries.

J. Koronacki et al. (Eds.): Advances in Machine Learning I, SCI 262, pp. 321–344.
springerlink.com

First experimental studies suggest that FURIA is superior to the original RIPPER in terms of classification accuracy, and that the rule fuzzification has an important part in the improvements. In this paper, we elaborate in more detail on the influence of rule fuzzification. In particular, we try to isolate the effect of rule fuzzification from the effect of other modifications. Moreover, going beyond the conventional classification problem, we investigate the performance of FURIA in the context of bipartite ranking.

To make the paper self-contained, we recall the basics of both the FURIA and RIPPER algorithms in the next section. In Section 3, we give some general arguments in favor of using fuzzy rules. An experimental evaluation of FURIA as well as an empirical analysis of the influence of rule fuzzification are presented in Section 4. The paper ends with a summary and concluding remarks in Section 5.

2 FURIA: Fuzzy Unordered Rule Induction Algorithm

This section gives a brief introduction of FURIA; for technical details, we refer to (HH09). As FURIA builds upon the separate-and-conquer rule induction procedure of RIPPER, we start with a recapitulation of the basics of this algorithm and give an overview of the main modifications afterward.

2.1 RIPPER

RIPPER was introduced by (Coh95) as a successor of the IREP algorithm for rule induction (FW94). Even though the key principles remained unchanged, RIPPER improves IREP in many details and, moreover, is able to cope with multi-class problems.

Consider a polychotomous (multi-class) classification problem with m classes $\mathbb{L} \stackrel{\text{df}}{=} \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$. Suppose instances to be represented in terms of attributes A_i, $(i = 1, \ldots, n)$, which are either numerical (real-valued) or nominal, and let $\mathbb{D}_i$ denote the corresponding domains. Thus, an instance is represented as an n-dimensional attribute vector

$$x = (x_1, x_2, \ldots, x_n) \in \mathbb{D} \stackrel{\text{df}}{=} \mathbb{D}_1 \times \ldots \times \mathbb{D}_n.$$

A single RIPPER rule is of the form $r = \langle r_A \,|\, r_C \rangle$, consisting of a premise part r_A and a consequent part r_C. The premise part r_A is a conjunction of predicates (selectors) which are of the form $(A_i = v_i)$ for nominal and $(A_i \,\theta\, v_i)$ for numerical attributes, where $\theta \in \{\leq, =, \geq\}$ and $v_i \in \mathbb{D}_i$. The consequent part r_C is a class assignment of the form $(\text{class} = \lambda)$, where $\lambda \in \mathbb{L}$. A rule $r = \langle r_A \,|\, r_C \rangle$ is said to *cover* an instance $x = (x_1, x_2, \ldots, x_n)$ if the attribute values x_i satisfy all the predicates in r_A.

RIPPER learns such rules in a greedy manner, following a separate-and-conquer strategy (Für99). Prior to the learning process, the training data is sorted by class labels in ascending order according to the corresponding class frequencies. Rules are then learned for the first $m-1$ classes, starting with the smallest one. Once a rule has been created, the instances covered by that rule are removed from the training data, and this is repeated until no instances from the target class are left. The algorithm then proceeds

with the next class. Finally, when RIPPER finds no more rules to learn, a default rule (with empty antecedent) is added for the last (and hence most frequent) class.

Rules for single classes are learned until either all positive instances are covered or the last rule r that has been added was "too complicated". The latter property is implemented in terms of the total description length (Qui95).

2.2 Learning Individual Rules

Each individual rule is learned in two steps. The training data, which has not yet been covered by any rule, is therefore split into a *growing* and a *pruning* set. In the first step, the rule will be specialized by adding antecedents which were learned using the growing set. Afterward, the rule will be generalized by removing antecedents using the pruning set.

When RIPPER learns a rule for a given class, the examples of that class are denoted as *positive* instances, whereas the examples from the remaining classes are denoted as *negative* instances.

A new rule is learned on the growing data, using a propositional version of the FOIL algorithm (Qui90; QCJ93). It starts with an empty conjunction and adds selectors until the rule covers no more negative instances, i.e., instances not belonging to the target class. The next selector to be added is chosen so as to maximize FOIL's information gain criterion (IG), which is a measure of improvement of the rule in comparison with the default rule for the target class:

$$\mathrm{IG}_r \stackrel{\mathrm{df}}{=} p_r \times \left(\log_2 \left(\frac{p_r}{p_r + n_r} \right) - \log_2 \left(\frac{p}{p+n} \right) \right), \tag{1}$$

where p_r and n_r denote, respectively, the number of positive and negative instances covered by the rule; likewise, p and n denote the number of positive and negative instances covered by the default rule.

The above procedure typically produces rules that overfit the training data. To remedy this effect, a rule is simplified so as to maximize its performance on the pruning data through a cut-off at the position that maximizes the rule-value metric

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

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

The ruleset *RS* produced by the learning algorithm outlined so far, called IREP*, is taken as a starting point for a subsequent optimization process in which all rules are re-examined. For each rule r, two alternative rules – the *replacement rule* and the *revision rule* – are considered. To decide which version of r to retain, the MDL (Minimum Description Length (Qui93)) criterion is used. Afterward, the remaining positives are covered using the IREP* algorithm.

The RIPPERk 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).

2.3 Modifications of RIPPER

A first modification of RIPPER concerns the type of rule model that is learned and, related to this, the use of default rules: 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.

When using an unordered rule set without default rule, two problems can occur in connection with the classification of a new query instance. First, a conflict may occur since the instance is equally well covered by rules from different classes. As will be seen in Section 2.6, this problem is rather unlikely to occur and, in case it still does, can easily be resolved. Second, it may happen that the query is not covered by any rule. To solve this problem, we propose a novel rule stretching method. The idea is to modify the rules in a local way so as to make them applicable to the query.

FURIA uses a rule stretching approach that exploits the order in which the antecedents were learned, treating them as a list $\langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$ instead of a set $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. 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 $\langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle$ 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. By Laplace-correcting the relative number of remaining antecedents, k/m, preference is given to longer and, hence, more specific rules.

Computationally, the above rule stretching strategy is much more efficient than the original proposal of Eineborg and Boström (Bos04). The complexity for re-evaluating a rule r is linear in its number of antecedents. 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.

The RIPPER algorithm can be divided into the building and the optimization phase. The rule building is done via the IREP* algorithm, which essentially consists of a propositional FOIL algorithm, the pruning strategy and the stopping conditions. Interestingly, we found that the pruning strategies in IREP* have a negative influence on the performance of FURIA. We therefore omitted the pruning step and instead learned the initial ruleset on the whole training data directly.

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 instances, which is again accomplished by IREP*, also benefited from omitting the pruning, just like 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 the pruning strategy tries to remove all antecedents from a rule, thereby generating a default rule. In this case, the pruning will be aborted, and the unpruned rule will be used for the MDL comparison in the optimization phase. We found that those pruning strategies are still sufficient to avoid overfitting. Thus, the removal of the pruning in the IREP* part has no negative impact on classification accuracy.

2.4 Representation of Fuzzy Rules

A selector constraining a numerical attribute A_i (with domain $\mathbb{D}_i = \mathbb{R}$) in a RIPPER rule can obviously be expressed in the form $(A_i \in I)$, where $I \subseteq \mathbb{R}$ is an interval: $I = (-\infty, c]$ if the rule contains a selector $(A_i \leq c)$, $I = [b, \infty)$ if it contains a selector $(A_i \geq b)$, and $I = [b, c]$ if it contains both (in the last case, two selectors are combined).

Essentially, a fuzzy rule is obtained through replacing intervals by fuzzy intervals, namely fuzzy sets with trapezoidal membership function.

A fuzzy interval of that kind is specified by four parameters and will be written $I^F = [a, b, c, d]$:

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

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. Note that, as in the non-fuzzy case, a fuzzy interval can be open to one side ($a = b = -\infty$ or $c = d = \infty$.) In fact, as will be seen later on, the fuzzy antecendents successively learned by FURIA are fuzzy half-intervals of exactly that kind.

A fuzzy selector $(A_i \in I_i^F)$ *covers* an instance $x = (x_1 \ldots x_n)$ to the degree $I_i^F(x_i)$. A fuzzy rule r^F involving k selectors $(A_i \in I_i^F)$, $i = 1, \ldots, k$, covers x to the degree

$$\mu_{r^F}(x) = \bigotimes_{i=1 \ldots k} I_i^F(x_i) \quad , \tag{2}$$

where $\otimes$ is a so-called t-norm operator such as the product or the minimum.

2.5 Rule Fuzzification

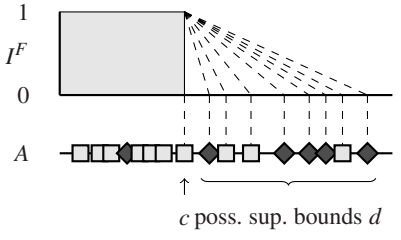
To obtain fuzzy rules, the idea is to fuzzify the final rules from our modified RIPPER algorithm. More specifically, using the training set $D \subseteq \mathbb{D}$ for evaluating candidates, the idea is to search for the best fuzzy extension of each rule, where a fuzzy extension is understood as a rule 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 interval I^F, 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 instances that are excluded by any other fuzzy interval I_j^F, $j \neq i$:

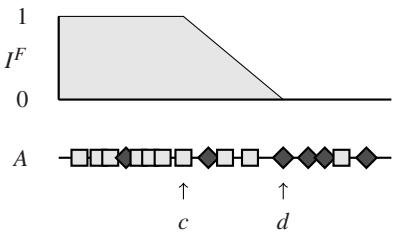
$$D^i \stackrel{\text{df}}{=} \left\{ x = (x_1, x_2, \ldots, x_k) \in D \,|\, I_j^F(x_j) > 0 \text{ for all } j \neq i \right\} \subseteq D \tag{3}$$

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

$$\text{pur} = \frac{p_i}{p_i + n_i}, \tag{4}$$



(a) Examination of possible support bounds of a crisp interval.



(b) Fuzzification that maximizes the interval's fuzzy purity.

Fig. 1. Fuzzification process

where

$$p_i \stackrel{\text{df}}{=} \sum_{x \in D^i_+} I_i(x) \quad \text{and} \quad n_i \stackrel{\text{df}}{=} \sum_{x \in D^i_-} I_i(x).$$

Rules are fuzzified in a greedy way, as shown by Algorithm 1. In each iteration, a fuzzification is computed for every antecedent, namely the best fuzzification in terms of (4). This is done by testing all values

$$\{x_i \,|\, x = (x_1, x_2, \ldots, x_k) \in D^i, x_i < b_i\}$$

as candidates for a_i and, likewise, all values

$$\{x_i \,|\, x = (x_1, x_2, \ldots, x_k) \in D^i, x_i > c_i\}$$

as candidates for d_i (see Fig. 1a). Ties are broken in favor of larger fuzzy sets, that is, larger distances from the core. The fuzzification is then realized for the antecedent with the highest purity, cf. Figure 1b. This is repeated until all antecedents have been fuzzified.

Algorithm 1. The antecedent fuzzification algorithm for a single rule r

```
Let A be the set of numeric antecedents of r
while A ≠ ∅ do
    a_max ← null // a_max denotes the antecedent with the highest purity
    pur_max ← 0 // pur_max is the highest purity value, so far
    for i ← 1 to size(A) do
        compute the best fuzzification of A[i] in terms of purity
        pur_A[i] ← be the purity of this best fuzzification
        if pur_A[i] > pur_max then
            pur_max ← pur_A[i]
            a_max ← A[i]
        end if
    end for
    A ← A \ a_max
    Update r with a_max
end while
```

Note that the fuzzification of a single antecedent may change the relevant training data (3), which is hence recomputed in each iteration. In fact, each fuzzification may increase the number of covered instances, 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.

2.6 Classifier Output

Suppose that fuzzy rules $r_1^{(j)}, r_2^{(j)}, \ldots, r_k^{(j)}$ have been learned for class λ_j. For a new query instance x, the support of this class is defined by

$$s_j(x) \stackrel{\text{df}}{=} \sum_{i=1,\ldots,k} \mu_{r_i^{(j)}}(x) \cdot CF\left(r_i^{(j)}\right) \quad , \tag{5}$$

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

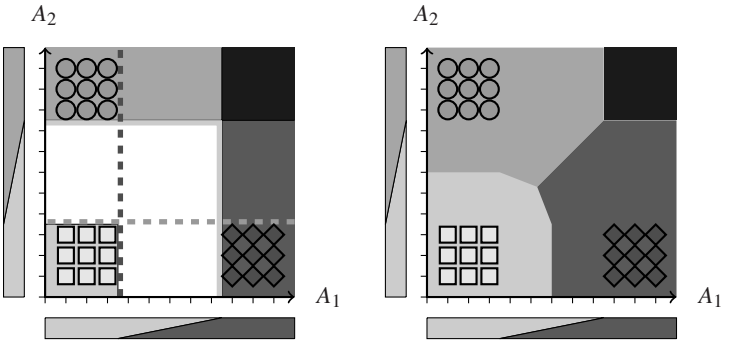
$$CF\left(r_i^{(j)}\right) = \frac{2\frac{|D_T^{(j)}|}{|D_T|} + \sum_{x \in D_T^{(j)}} \mu_{r_i^{(j)}}(x)}{2 + \sum_{x \in D_T} \mu_{r_i^{(j)}}(x)} \quad , \tag{6}$$

where $D_T^{(j)}$ denotes the subset of training instances with label λ_j. The authors in (IN01, IY05) argued that weighing rules according to (5) allows for modeling more flexible decision boundaries and thereby improves classification accuracy. The certainty factor (6) is the m-estimate for $m = 2$ (PFTV92).

The class predicted by FURIA is the one with maximal score. In the case where x is not covered by any rule, which means that $s_j(x) = 0$ for all classes λ_j, a classification decision is derived using the rule stretching procedure (see above). In the case of a tie, a decision in favor of the class with highest frequency is made.

3 Advantages of Fuzzy Rules

Fuzzy rules are more general than conventional rules and have a number of potential advantages. For example, conventional (non-fuzzy) rules produce models with "sharp" decision boundaries and, correspondingly, abrupt transitions between different classes. This property is questionable and not very intuitive. Instead, one would 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. As explained above, fuzzy rules have "soft" boundaries, which is one of their main characteristics. Admittedly, if a definite classification decision has to be made, soft boundaries have again to be turned into crisp boundaries. Interestingly, however, these boundaries are potentially more flexible in the fuzzy case. For example, by using suitable aggregation operators for combining fuzzy rules, they are not necessarily axis-parallel (PFTV92).



(a) Rule cores and support edge **(b)** Decision boundaries

Fig. 2. Three axis-parallel fuzzy rules (light gray region, dark gray region, and region marked by the intersection of the dotted lines) learned for a three-class problem (left) and the induced decision boundaries (right)

As an illustration, consider the three-class problem in Fig. 2a. For the top-left and bottom-right class, a simple one-dimensional rule is sufficient. The rule cores tightly fit the data, while the rule supports reach 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 toward the other classes. As can be seen in Fig. 2b, the decision boundaries induced by the rules are non-linear, even though all rule conditions are axis-parallel.

In general, one may wonder to what extent rule fuzzification in FURIA can actually become effective in the sense of increasing the rule purity. To answer this question, one should first notice that a trivial fuzzification does always exist, namely the one that puts the support bound to the first instance behind the core bound. 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 other rule. The previous example was exactly of that kind.

The question remains, of course, whether there are non-trivial fuzzifications. Admittedly, such fuzzifications cannot exist when a rule contains only one fuzzy interval as numerical antecedent. In this case, RIPPER has learned this rule with the objective to maximize the information gain (1) which implicitly contains the purity. Consequently, the last instance – marking the core boundary – covered by that interval has to be from the class represented by the rule. A non-trivial fuzzification cannot exist, since the growing procedure would have selected the support bound as core position.

A non-trivial fuzzification can be found, however, when a rule uses at least two numeric attributes in its antecedence part. In such cases, it can happen that the myopic rule-learning strategy disables RIPPER (like any separate-and-conquer rule learner without look-ahead) to find the globally optimal solution. As an illustration, consider the following example.

Example 1. Assume examples from classes □ and ◆ lying in $A_1 \times A_2 \subseteq \mathbb{R}^2$, cf. Fig. 3a. Moreover, consider a rule learner that tries to optimize the purity. The optimal non-fuzzy rule for the latter class is

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

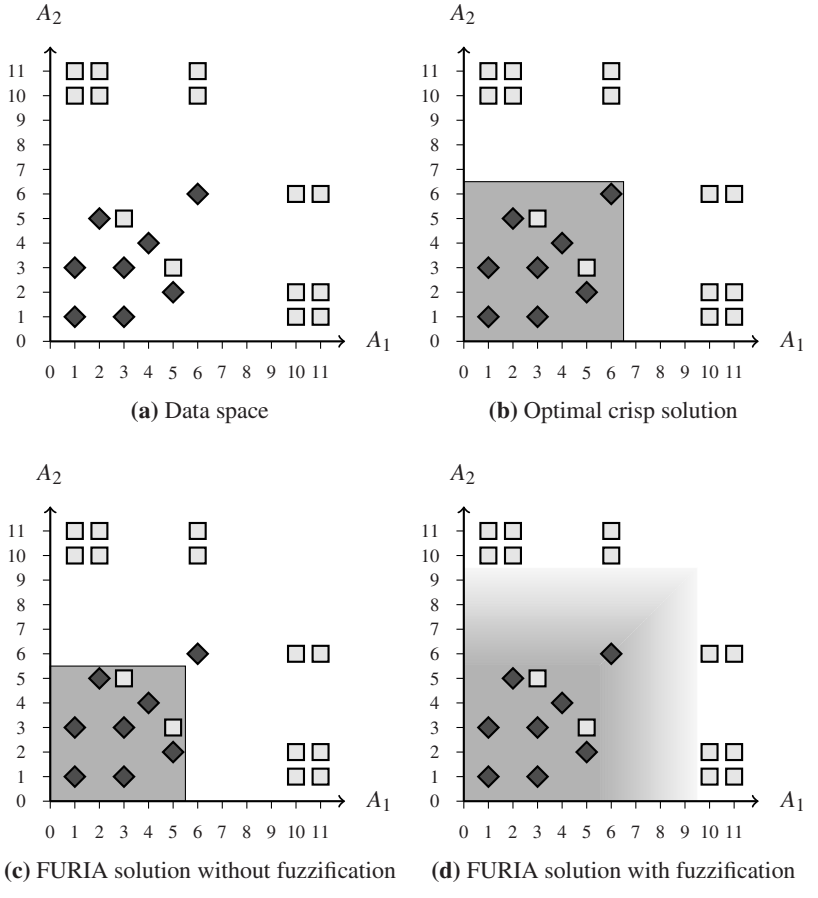
with a purity of $\frac{4}{5}$, cf. Fig. 3b. The myopic separate-and-conquer strategy is not able to find this rule, however, because it adds antecedents successively. Thus, it first finds the rule

$$\text{IF } A_1 \in (-\infty, 5] \text{ THEN class} = ◆$$

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

$$\text{IF } A_1 \in (-\infty, 6] \text{ THEN class} = ◆$$

is refused since it has a purity of only $\frac{1}{2}$.



(a) Data space **(b)** Optimal crisp solution

(c) FURIA solution without fuzzification **(d)** FURIA solution with fuzzification

Fig. 3. A two-dimensional classification problem showing the failure of myopic separate-and-conquer learning and the correction through post-generalization (fuzzification)

The next step in the rule learning procedure can not undo the mistake and finds the rule

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

with a purity of $\frac{7}{9} < \frac{4}{5}$, cf. Figure 3c. Note that, due to the symmetry of the problem, the same values result when considering A_2 first.

In this situation a non-trivial fuzzification will lead to the rule

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

The example at position $(6,6)$ is now covered to a degree of $\frac{10-6}{10-5} = 0.8$ by each condition, resulting in a purity of $\frac{39}{49}$ when using the minimum t-norm in (2).

This example suggests that the rule fuzzification of FURIA, which is also a kind of generalization, may help to undo mistakes that were made due to short-sighted (myopic) learning strategies. In the next section, we shall investigate this conjecture on an experimental basis.

4 Experiments

To analyze the performance of FURIA, several experimental studies were conducted. As a starting point, we used the RIPPER implementation of WEKA ("JRip"), cf. Section 2.1, for implementing the rule learning part of FURIA.

Our testbed for the experiments consists of 45 real-world classification datasets. Most of the datasets, 24 in total, were taken from the UCI repository (AN07), 13 datasets originate from the Statlib repository (MV07) and 3 come from an agricultural domain (Bul07; Bar07; Har07). The remaining five datasets were generated from meteorological station data, published by the German Weather Service (DWD). Table 1 gives an overview of the datasets and their main characteristics. We only used datasets with at least as many numeric as nominal attributes, since the fuzzy techniques presented in this work are ineffective for nominal data.

Table 1. An overview of the datasets used in the experiments

Dataset	# Inst.	# Classes	# Attributes Cont.	Nom.	Miss.	Origin
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-lawsuit	264	2	3	1	0	Statlib
biomed	209	2	7	1	2	Statlib
haberman	306	2	2	1	0	UCI
heart-statlog	270	2	13	0	0	UCI
ionosphere	351	2	34	0	0	UCI
liver-disorders	345	2	6	0	0	UCI

Table 1. (*continued*)

Dataset	# Inst.	# Classes	# Attributes Cont.	Nom.	Miss.	Origin
pima diabetes	768	2	8	0	0	UCI
prnn-synth	250	2	2	0	0	Statlib
schizo-	340	2	12	2	11	Statlib
sonar	208	2	60	0	0	UCI
wisconsin-breast-cancer	699	2	9	0	1	UCI
analcatdata-authorship	841	4	70	0	0	Statlib
analcatdata-halloffame	1340	3	15	2	1	Statlib
analcatdata-votesurvey	48	4	3	1	0	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
iris	150	3	4	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
segment	2310	7	19	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	5000	3	40	0	0	UCI
wine	178	3	13	0	0	UCI

To evaluate a classifier in terms of its predictive performance, a dataset is randomly split into two parts, $2/3$ for training and $1/3$ for testing. To reduce random effects, the performance measure (e.g., classification rate or AUC) is averaged over 100 repetitions of this procedure.

To analyze the results, we follow the two-step procedure recommended by (Dem06): First, a Friedman test is conducted to test the null hypothesis of equal classifier

performance (Fri37; Fri40). In case this hypothesis is rejected, which means that the classifiers' performance differs in a statistically significant way, a posthoc test is conducted to compare the classifiers in a pairwise way. Here, we apply the Bonferroni-Dunn test (Dun61) which, just like the Friedman test, is a rank-based statistic: On each dataset, the classifiers are ranked according to their performance, i.e., the best classifier receives rank 1, the second-best rank 2, and so on. The overall performance of a classifier is then measured in terms of its average rank, and what the aforementioned tests do is essentially looking at the differences between these averages.

To visualize the statistical test results, Demšar proposed a number line from 1 to k, which represents the average ranks of the k classifiers according to the Friedman test. Every single classifier is marked on that number line at the position of its average rank, and an interval that has twice the width of the critical distance is centered at the rank of the reference classifier. Thus, another classifier falling outside this interval differs from the reference classifier in a statistically significant way.

4.1 Classification Performance

In the first experiment, we compared FURIA to other classifiers using the classification rate as a performance measure, i.e., the percent of correct classifications. For RIPPER we used the WEKA default settings. 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. The CHI algorithm is based on (CWY95, CYP96) and uses rule weighing as proposed by (IY05).[1] The SLAVE algorithm makes use of genetic algorithms to learn a fuzzy classifier (GP99; GP01).[2] Finally, we also included the well-known C4.5 decision tree learner as a benchmark classifier (Qui93; Qui95).

Table 2. Average accuracies and ranks

data set	FURIA	RIPPER	C4.5	CHI	SLAVE
acd-authorship	95.67(1)	93.05(3)	93.50(2)	71.60(5)	91.87(4)
acd-bankruptcy	82.57(1)	81.97(2)	81.29(3)	74.40(5)	77.80(4)
acd-cyyoung8092	80.02(2)	80.04(1)	79.86(3)	70.72(5)	79.32(4)
acd-cyyoung9302	82.64(2)	82.01(3)	80.82(4)	80.27(5)	83.90(1)
acd-esr	80.90(2)	82.38(1)	80.36(3)	79.55(4)	77.72(5)
acd-halloffame	92.92(1)	92.87(3)	92.87(2)	92.18(5)	92.68(4)
acd-lawsuit	98.00(1)	97.54(3)	97.94(2)	94.93(4)	94.81(5)
acd-votesurvey	36.92(3)	34.40(4)	38.75(2)	40.19(1)	29.51(5)
biomed	88.31(1)	87.40(3)	87.80(2)	80.64(5)	84.74(4)
cars	79.08(2)	75.93(3)	82.15(1)	68.97(5)	70.68(4)
collins	96.35(1)	92.89(3)	96.10(2)	42.63(5)	50.87(4)

[1] We used the following parameter setting: 3 fuzzy sets, product t-norm, maximum inference, and weighting scheme number 2 from (IY05).

[2] We used the following parameter setting: 5 fuzzy sets, 500 iterations without change, mutation probability 0.01, use weights, population size 100.

Table 2. (*continued*)

data set	FURIA	RIPPER	C4.5	CHI	SLAVE
ecoli	83.12(1)	80.57(4)	81.35(2)	77.43(5)	81.03(3)
eucalyptus	60.62(1)	58.69(3)	59.98(2)	54.09(5)	58.16(4)
glass	68.22(1)	63.18(3)	66.69(2)	61.39(5)	61.83(4)
haberman	72.72(3)	72.16(4)	71.75(5)	73.08(2)	73.31(1)
heart-statlog	79.75(1)	78.44(2)	77.08(4)	68.66(5)	78.44(3)
ionosphere	89.59(2)	88.64(4)	88.72(3)	66.40(5)	89.83(1)
iris	94.76(2)	93.45(4)	94.25(3)	92.27(5)	94.92(1)
liver-disorders	67.15(1)	65.93(2)	63.40(3)	58.75(5)	59.77(4)
metStatCoord.	93.02(1)	92.04(3)	92.87(2)	46.79(5)	58.77(4)
metStatRainfall	64.51(1)	60.66(2)	59.47(3)	24.51(5)	29.35(4)
metStatRST	33.56(4)	36.08(3)	38.60(2)	25.24(5)	42.02(1)
metStatSunshine	49.05(1)	44.48(3)	46.78(2)	37.93(4)	28.83(5)
metStatTemp	50.71(2)	47.45(3)	53.18(1)	30.63(4)	22.10(5)
mfeat-factors	92.09(1)	87.05(4)	87.96(3)	89.19(2)	86.83(5)
mfeat-fourier	76.69(1)	71.37(4)	74.42(2)	69.27(5)	73.49(3)
mfeat-karhunen	86.47(1)	79.13(4)	80.20(3)	82.55(2)	78.37(5)
mfeat-morpholog.	72.09(1)	70.74(3)	71.60(2)	57.93(5)	67.08(4)
mfeat-zernike	73.67(1)	67.58(5)	69.11(3)	72.37(2)	68.26(4)
optdigits	94.78(1)	89.68(3)	89.51(4)	45.90(5)	93.45(2)
page-blocks	97.02(1)	96.79(3)	96.89(2)	91.96(5)	93.58(4)
pasture-prod.	74.67(1)	68.46(3)	73.67(2)	44.23(5)	53.63(4)
pendigits	97.77(1)	95.54(4)	95.92(3)	97.45(2)	87.26(5)
pima diabetes	74.71(1)	74.56(2)	73.43(4)	72.55(5)	73.65(3)
prnn-synth	83.57(2)	82.50(4)	83.18(3)	84.14(1)	81.51(5)
schizo-	80.52(1)	75.33(2)	74.93(3)	56.08(5)	56.29(4)
segment	96.50(1)	94.53(3)	95.95(2)	83.65(5)	88.87(4)
sonar	77.01(1)	72.41(3)	72.09(4)	74.61(2)	68.50(5)
squash-unstored	76.44(1)	71.74(3)	76.08(2)	70.56(4)	65.56(5)
synthetic control	89.75(2)	82.85(4)	90.00(1)	68.33(5)	89.23(3)
vehicle	70.10(2)	67.80(3)	71.38(1)	61.99(5)	64.08(4)
vowel	75.43(2)	64.71(3)	75.60(1)	59.49(5)	63.84(4)
waveform	82.24(1)	78.72(2)	75.05(4)	72.38(5)	75.34(3)
wine	93.25(1)	90.02(5)	91.22(4)	92.77(2)	92.46(3)
w.-breast-cancer	95.68(1)	95.58(2)	94.51(4)	90.20(5)	95.49(3)
average	1.40	3.07	2.60	4.24	3.69

The overall picture conveyed by the results, summarized in Table 2, is clearly in favor of FURIA, which outperforms the other methods on most data sets. In fact, the Friedman test rejects the null-hypothesis quite safely (at a significance level < 0.01) and, hence, indicates that there are significant differences between the classifiers' performance. The results of the Bonferroni-Dunn test are summarized graphically in Fig. 4. As can be seen, FURIA is significantly better than all other classifiers at the significance level of 0.05.

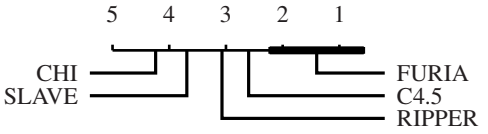


Fig. 4. Visualization of the Bonferroni-Dunn test according to Demšar (Dem06). FURIA is significantly better than its competitors in terms of classification rate (significance level 0.05).

4.2 Ranking Performance

In a second study, we analyzed the ranking performance of the classifiers, using the AUC (Area Under the ROC Curve) as an evaluation measure (PF97; PFK98). This study is motivated by the recent paper (HV09), according to which fuzzy models are especially suitable for this kind of problem. In bipartite ranking, the problem is to order the test instances from the most likely positive to the most likely negative. Typically, a ranking of this kind is established by sorting the instances according to the scores they receive from the classifier. A ranking error occurs when a negative instance is ranked higher than a positive one, and the AUC essentially counts the number of such errors. For multi-class problems, we used an extension of the AUC as proposed in (PD03).

Table 3. Average AUC and ranks

data set	FURIA	RIPPER.	C4.5	CHI	SLAVE.
acd-authorship	0.98(1)	0.96(3)	0.96(4)	0.90(5)	0.98(2)
acd-bankruptcy	0.87(1)	0.82(5)	0.83(4)	0.87(2)	0.86(3)
acd-cyyoung8092	0.74(2)	0.71(3)	0.67(4)	0.65(5)	0.77(1)
acd-cyyoung9302	0.77(2)	0.71(4)	0.70(5)	0.75(3)	0.79(1)
acd-esr	0.67(1)	0.62(3)	0.63(2)	0.60(4)	0.59(5)
acd-halloffame	0.83(3)	0.79(5)	0.81(4)	0.87(2)	0.89(1)
acd-lawsuit	0.95(2)	0.92(4)	0.93(3)	0.81(5)	0.96(1)
acd-votesurvey	0.52(3)	0.50(4)	0.54(2)	0.55(1)	0.49(5)
biomed	0.90(3)	0.86(5)	0.87(4)	0.93(1)	0.91(2)
cars	0.88(2)	0.82(4)	0.90(1)	0.86(3)	0.81(5)
collins	1.00(1)	0.97(3)	0.98(2)	0.88(4)	0.83(5)
ecoli	0.91(3)	0.89(4)	0.89(5)	0.94(1)	0.91(2)
eucalyptus	0.79(3)	0.82(2)	0.82(1)	0.79(4)	0.77(5)
glass	0.81(1)	0.76(5)	0.79(3)	0.77(4)	0.79(2)
haberman	0.62(2)	0.60(3)	0.56(5)	0.65(1)	0.60(4)
heart-statlog	0.82(1)	0.79(3)	0.78(4)	0.76(5)	0.81(2)
ionosphere	0.91(2)	0.88(4)	0.88(5)	0.89(3)	0.92(1)
iris	0.97(3)	0.96(5)	0.96(4)	1.00(1)	0.98(2)
liver-disorders	0.68(1)	0.64(2)	0.62(3)	0.57(5)	0.60(4)
metStatCoord.	0.98(2)	0.98(1)	0.98(3)	0.78(5)	0.82(4)
metStatRainfall	0.87(1)	0.85(2)	0.81(3)	0.76(4)	0.66(5)
metStatRST	0.65(3)	0.70(2)	0.71(1)	0.59(4)	0.58(5)
metStatSunshine	0.77(2)	0.75(4)	0.74(5)	0.80(1)	0.75(3)
metStatTemp	0.76(3)	0.77(2)	0.77(1)	0.66(5)	0.68(4)

Table 3. (*continued*)

data set	FURIA	RIPPER	C4.5	CHI	SLAVE
mfeat-factors	0.98(2)	0.95(4)	0.94(5)	0.98(1)	0.97(3)
mfeat-fourier	0.92(2)	0.92(4)	0.88(5)	0.92(1)	0.92(3)
mfeat-karhunen	0.96(2)	0.92(4)	0.90(5)	0.98(1)	0.95(3)
mfeat-morpholog.	0.88(5)	0.94(1)	0.92(3)	0.93(2)	0.89(4)
mfeat-zernike	0.91(2)	0.90(4)	0.87(5)	0.95(1)	0.90(3)
optdigits	0.99(1)	0.96(3)	0.95(4)	0.82(5)	0.96(2)
page-blocks	0.95(1)	0.93(3)	0.93(2)	0.82(5)	0.83(4)
pasture-prod.	0.86(1)	0.79(3)	0.83(2)	0.64(5)	0.71(4)
pendigits	1.00(2)	0.98(4)	0.98(5)	1.00(1)	0.99(3)
pima diabetes	0.73(4)	0.71(5)	0.74(3)	0.80(1)	0.76(2)
prnn-synth	0.85(3)	0.84(5)	0.84(4)	0.90(1)	0.90(2)
schizo-	0.86(1)	0.78(3)	0.81(2)	0.56(5)	0.59(4)
segment	0.99(1)	0.98(2)	0.98(3)	0.97(4)	0.96(5)
sonar	0.81(2)	0.74(4)	0.72(5)	0.82(1)	0.75(3)
squash-unstored	0.83(1)	0.77(3)	0.81(2)	0.71(5)	0.76(4)
synthetic control	0.97(3)	0.93(5)	0.95(4)	0.99(1)	0.99(2)
vehicle	0.85(1)	0.85(2)	0.85(4)	0.85(3)	0.83(5)
vowel	0.93(1)	0.88(5)	0.91(3)	0.91(2)	0.89(4)
waveform	0.91(2)	0.88(4)	0.83(5)	0.91(3)	0.91(1)
wine	0.97(3)	0.93(5)	0.93(4)	0.98(1)	0.97(2)
w.-breast-cancer	0.97(3)	0.96(4)	0.95(5)	0.98(1)	0.97(2)
average	2.02	3.53	3.51	2.84	3.09

The results of this study, summarized in Table 3 and Fig. 5, convey the same message as the previous one: FURIA is significantly better than the other classifiers. Worth mentioning is also the improved performance of the other two fuzzy classifiers, CHI and SLAVE. The fact that the fuzzy approaches outperform the non-fuzzy ones in terms of ranking performance is completely in agreement with the results in (HV09).

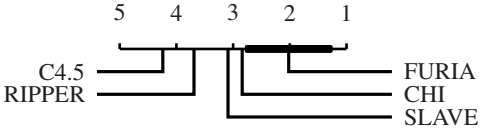


Fig. 5. Visualization of the Bonferroni-Dunn test according to Demšar (Dem06). FURIA is significantly better than its competitors in terms of AUC (significance level 0.1).

4.3 The Effect of Fuzzification

The previous results have shown 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 or, more generally, to the post-generalization step which involves the

fuzzification of rules as a key component. Needless to say, distilling and quantifying the influence of single modifications is rather difficult, if not impossible, given that all components of an algorithm interact in a complex and non-trivial way. Nevertheless, to get a rough idea, we shall at least make an attempt in this direction. To this end, we compare different variants of FURIA in terms of their predictive performance:

FURIA-prod. The original FURIA algorithm using the product t-norm in (2) to combine rule antecedents.
FURIA-min. The FURIA algorithm using the minimum t-norm instead of the product.
FURIA-crisp. This is a "crisp" version of FURIA, in which the fuzzy intervals are turned back into normal intervals. To this end, all membership degrees $I^F(x_i) > 0$ are lifted to the degree 1.
FURIA-w/o. This version does not apply any post-generalization technique; the rules remain in the form as they were learned by RIPPER.

To minimize the interference with other extensions, we disable the rule stretching technique. This, however, means that test instances may remain uncovered, i.e., that FURIA will abstain from the classification of some instances. To take this into consideration, we measure three instead of only a single performance metric: The *accuracy* is the percentage of correct classifications, the *error* the percentage of incorrect classifications, and the *abstention* the percentage of unclassified instances; obviously, these three quantities sum to 1.

The complete results of this test can be found in Tables 6–9. A synopsis can also be found in Table 4 and Table 5. To cope with the large amount of information, we try to answer a number of relevant questions step by step.

4.3.1 Benefit of Post-Generalization

In Section 3 and Example 1, we argued that post-generalizing the rules may correct an improper placement of the interval boundaries. Our first question therefore concerns the

Table 4. Wins and losses in terms of 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 5. Wins and losses in terms of 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	-

contribution of the post-generalization for avoiding the myopia of separate-and-conquer rule induction. To answer this question we have to look at the performance on the training data. If the generalization process is beneficial, the performance of FURIA-w/o should be worse than the performance of the other variants.

In agreement with this expectation, we find that FURIA-w/o has the worst accuracy on virtually all data sets, cf. Table 5. Moreover, FURIA-w/o has also a higher error. The other variants have a lower classification error on at least 28 data sets and are tying on 7. A simple sign test rejects the hypothesis of equal performance with an error probability of 0.01. From this we can conjecture that the post-generalization has a beneficial impact on both accuracy and error on the training data. Especially the observation of a smaller training error gives evidence to the claim that the post-generalization mitigates the initial myopia.

4.3.2 Benefit of Fuzzy Rules

Our second question concerns the impact of the use of fuzzy instead of conventional rules on the classification performance. Making use of fuzzy intervals, a fuzzy rule can cover an instance to a certain degree: While instances in the core are definitely covered by the rule, the degree of coverage decreases linearly toward the boundary of the rule. Of course, even though this notion makes sense intuitively, it is a-priori not clear that it also leads to an improved performance. Therefore, we compare the fuzzy versions FURIA-prod and FURIA-min to the non-fuzzy variant FURIA-crisp. As can be seen in Table 4, FURIA-prod and FURIA-min win 37 and lose only 7 data sets over FURIA-crisp in terms of both accuracy and error on the test data.

A surprising observation is that, in contrast to the test data, FURIA-crisp wins 26 over both fuzzy competitors FURIA-prod and FURIA-min and ties 13 data sets in terms of accuracy and error on the training data (cf. Table 5). Interestingly, a similar observation has been made by Kuwajima et al. (KNI08). In a related analysis, they also found that a stronger fuzzification leads to worse results on the training data but not on the test data.

In summary, these results clearly suggest that a gradual extrapolation, which gives a higher support to instances in the core of a rule than to instances in the boundary region, is not only intuitively appealing but also advantageous with regard to generalization performance.

4.3.3 Choice of Fuzzy Aggregation Operator

Finally, we investigated whether the product t-norm is preferable to the minimum t-norm for fuzzy inference. Since the product has isobars with a round shape, it leads to decision boundaries that are smoother than the boundaries produced by the minimum. Moreover, it extrapolates in a more "careful" way, since the product yields membership degrees that are smaller than those produced by the minimum.[3] Finally, the product gives rise to an interesting probabilistic interpretation: Looking at the membership degrees as (cumulative) probabilities of the location of the interval boundaries, which are considered as random variables, and making a simplifying assumption of independence

[3] For an instance x covered by I_1 and I_2 with $0 < I_1(x_1) \neq I_2(x_2) < 1$, we have $\top_{\text{prod}}\big(I_1(x_1), I_2(x_2)\big) < \top_{\min}\big(I_1(x_1), I_2(x_2)\big)$.

between the boundaries of different attributes, the product combination corresponds to the probability that an instance is covered by the rule.

In terms of predictive performance, we indeed find that FURIA-prod is significantly better than FURIA-min (at a significant level of 0.05): It wins on 22 data sets and ties on 15 for accuracy, and for the error measure it wins even 23 data sets. It is worth mentioning that the results are again different on the training data: FURIA-min is better than FURIA-prod on 9 data sets and worse on only one for both classification accuracy and error. Again, it seems that a more "careful" extrapolation, as realized by the product t-norm, is advantageous in terms of generalization performance.

5 Summary and Conclusion

A major goal of this paper is to show that rule induction, an important subfield of machine learning, can strongly benefit from tools and concepts that have been developed in the field of fuzzy logic. To this end, we analyzed FURIA, a fuzzy rule-based classification method that builds on RIPPER. Our experimental results have shown that FURIA significantly outperforms the original RIPPER, as well as other rule learners used for comparison, in terms of both classification and ranking performance.

Elaborating on the advantages of a fuzzy approach, we tried to distill and quantify the influence of rule fuzzification on the strong classification performance of FURIA. Conducting experiments with different variants of the algorithm, we indeed found strong evidence for the importance of fuzzy rules. In particular, it seems that a gradual extrapolation as realized by fuzzy rules is advantageous with regard to the generalization performance of a rule-based classifier.

Going beyond the conventional classification problem, we also investigated the performance of FURIA in the context of bipartite ranking. As noted in the recent literature, fuzzy models appear to be especially suitable for this type of problem, a supposition which is fully supported by our results. In future work, we therefore plan to investigate the advantages of a fuzzy approach in the context of ranking in more detail.

Further Tables

Table 6. Average accuracy on the test data for variants of FURIA

data set	prod	min	crisp	w/o
analcatdata-authorship	92.80921	92.80921	92.55734	91.22824
analcatdata-bankruptcy	81.32557	81.32925	82.58047	76.12663
analcatdata-cyyoung8092	76.01159	76.04189	76.19168	73.30849
analcatdata-cyyoung9302	78.74153	78.70927	79.05907	76.49093
analcatdata-esr	80.09091	80.09091	80.91818	75.97121
analcatdata-halloffame	90.87807	90.87588	90.84069	89.38324
analcatdata-lawsuit	97.77013	97.77013	97.72594	97.02380
analcatdata-votesurvey	8.56054	8.56054	8.56054	7.33627
biomed	85.42540	85.41131	85.18755	82.39893

Table 6. (*continued*)

data set	prod	min	crisp	w/o
cars	74.60520	74.59071	74.51809	72.33939
collins	96.01711	96.02881	95.10308	89.67611
ecoli	80.58999	80.58138	79.96263	76.89322
eucalyptus	47.92876	47.92878	47.72092	46.76565
glass	60.66062	60.64654	59.69095	54.49754
haberman	67.24621	67.24621	67.32295	66.93906
heart-statlog	73.32883	73.32883	73.14369	71.92296
ionosphere	86.64396	86.62716	86.51006	84.33937
iris	94.00000	94.00000	93.33333	91.74510
liver-disorders	57.66717	57.66717	57.47095	56.12342
metStatCoordinates	92.18901	92.18282	91.98087	90.28669
metStatRainfall	56.11860	56.11798	55.69124	51.63397
metStatRST	22.53966	22.53966	22.23236	20.24124
metStatSunshine	41.36317	41.37031	41.12752	36.62300
metStatTemp	42.04130	42.03699	41.84980	40.39076
mfeat-factors	88.99853	88.99265	88.74559	86.35588
mfeat-fourier	70.69559	70.69265	70.27794	66.62353
mfeat-karhunen	82.40294	82.40441	81.67794	77.45000
mfeat-morphological	68.09853	68.09853	68.09706	66.80441
mfeat-zernike	68.08382	68.09265	67.57206	63.87941
optdigits	92.55994	92.55993	92.31394	90.40031
page-blocks	96.52939	96.52831	96.37623	95.90225
pasture-production	64.65293	64.65293	63.30037	58.21520
pendigits	97.07328	97.07007	96.70082	95.02126
pima diabetes	68.94244	68.94244	68.91185	67.83203
prnn-synth	81.04536	81.04536	80.99885	79.49978
schizo-	72.38598	72.37736	71.90262	69.93216
segment	95.53575	95.53829	95.13213	93.69444
sonar	70.20954	70.19525	69.63149	66.04831
squash-unstored	72.87754	72.87754	73.19556	67.90609
synthetic control	85.52941	85.51961	84.48039	80.25980
vehicle	62.84809	62.84462	62.62561	61.07522
vowel	71.10840	71.08169	67.91235	61.32328
waveform-5000	75.34261	75.34261	75.34909	70.52370
wine	90.57375	90.55709	89.61175	86.27115
wisconsin-breast-cancer	94.25726	94.25726	94.14377	93.64308

Table 7. Average error on the test data for variants of FURIA

data set	prod	min	crisp	w/o
analcatdata-authorship	2.68560	2.68560	2.93748	2.52823
analcatdata-bankruptcy	16.78513	16.78145	15.53023	12.45956
analcatdata-cyyoung8092	17.39773	17.36742	17.21764	15.79217
analcatdata-cyyoung9302	14.82836	14.86062	14.51082	13.19684

Table 7. (*continued*)

data set	prod	min	crisp	w/o
analcatdata-esr	17.35909	17.35909	16.53182	15.54545
analcatdata-halloffame	5.71984	5.72203	5.75722	5.16018
analcatdata-lawsuit	1.93935	1.93935	1.98355	1.74996
analcatdata-votesurvey	18.96895	18.96895	18.96895	16.91095
biomed	9.42371	9.43779	9.66155	8.17166
cars	14.39157	14.40606	14.47868	12.12514
collins	2.97016	2.95846	3.88419	0.12375
ecoli	13.89950	13.90812	14.52686	12.33990
eucalyptus	23.42483	23.42482	23.63268	21.91884
glass	23.12261	23.13669	24.09228	18.70914
haberman	22.99615	22.99615	22.91941	22.73690
heart-statlog	16.25442	16.25442	16.43956	15.65444
ionosphere	8.51115	8.52796	8.64505	7.45617
iris	4.70588	4.70588	5.37255	4.27451
liver-disorders	25.18079	25.18079	25.37701	24.30197
metStatCoordinates	6.13532	6.14151	6.34346	5.23401
metStatRainfall	20.59847	20.59909	21.02583	16.57657
metStatRST	27.89949	27.89949	28.20679	22.68913
metStatSunshine	30.61461	30.60746	30.85025	23.06217
metStatTemp	25.78395	25.78826	25.97545	22.89670
mfeat-factors	4.86618	4.87206	5.11912	4.13824
mfeat-fourier	16.02500	16.02794	16.44265	13.05000
mfeat-karhunen	8.48088	8.48088	9.20735	6.58676
mfeat-morphological	22.51029	22.51029	22.51176	21.50441
mfeat-zernike	15.50000	15.49853	16.15735	12.97059
optdigits	3.39419	3.39419	3.64018	2.84412
page-blocks	2.65105	2.65213	2.80421	2.32270
pasture-production	15.12546	15.12546	16.47802	12.53571
pendigits	1.69079	1.69400	2.06325	1.44917
pima diabetes	20.77898	20.77898	20.80957	20.08571
prnn-synth	14.57498	14.57498	14.62149	13.65491
schizo-	13.29685	13.30547	13.78021	12.19070
segment	2.41159	2.40904	2.81521	1.85139
sonar	18.64786	18.66214	19.22590	16.59549
squash-unstored	20.36859	20.36859	20.05057	17.38321
synthetic control	6.91176	6.92157	7.96078	5.85784
vehicle	16.90668	16.91015	17.12915	15.50935
vowel	16.82586	16.85257	20.02191	11.63290
waveform-5000	14.07581	14.07581	14.06934	12.21165
wine	4.35848	4.37514	5.32048	3.10506
wisconsin-breast-cancer	3.66027	3.66027	3.77375	3.50447

Table 8. Average accuracy on the training data for variants of FURIA

data set	prod	min	crisp	w/o
analcatdata-authorship	99.58021	99.58021	99.58021	99.53697
analcatdata-bankruptcy	97.06350	97.06350	97.06350	96.91199
analcatdata-cyyoung8092	90.53704	90.53704	90.53704	90.27212
analcatdata-cyyoung9302	92.34319	92.34319	92.34319	92.21176
analcatdata-esr	88.24697	88.24697	88.24697	88.05866
analcatdata-halloffame	95.40479	95.40479	95.37877	95.18996
analcatdata-lawsuit	99.10482	99.10482	99.10482	99.09907
analcatdata-votesurvey	24.05953	24.05953	24.05953	24.05953
biomed	96.04186	96.04186	96.05630	95.85345
cars	87.73836	87.73836	87.65253	87.48447
collins	99.73038	99.73038	99.73038	99.42730
ecoli	90.76378	90.76378	90.78166	90.50300
eucalyptus	61.47577	61.47577	61.51902	61.14015
glass	81.52163	81.52163	81.54296	80.81389
haberman	72.38894	72.38894	72.40874	72.25012
heart-statlog	85.61704	85.61704	85.63392	85.44872
ionosphere	96.34704	96.34704	96.34704	96.19581
iris	97.86869	97.86869	97.93939	97.76768
liver-disorders	72.34301	72.34301	72.38699	71.82926
metStatCoordinates	96.76641	96.76704	96.76321	96.45910
metStatRainfall	73.55085	73.55245	73.61245	72.42661
metStatRST	36.89565	36.89565	36.91824	36.50843
metStatSunshine	69.69173	69.69530	69.71691	69.11754
metStatTemp	63.47378	63.47378	63.47820	63.21942
mfeat-factors	98.97727	98.97727	98.97727	98.93636
mfeat-fourier	89.99167	89.99167	90.00758	89.55682
mfeat-karhunen	97.18485	97.18561	97.18939	96.93182
mfeat-morphological	74.55000	74.55000	74.57955	74.28182
mfeat-zernike	82.13636	82.13712	82.16591	81.69091
optdigits	99.35111	99.35111	99.35084	99.26996
page-blocks	97.86254	97.86032	97.71748	97.74737
pasture-production	89.06472	89.06472	89.06472	89.06472
pendigits	99.64230	99.64244	99.64285	99.53106
pima diabetes	75.01928	75.01928	75.06860	74.48856
prnn-synth	87.12079	87.12079	87.26008	86.89067
schizo-	88.35659	88.35659	88.39222	88.05802
segment	98.83123	98.83188	98.83713	98.64758
sonar	95.69622	95.69622	95.69622	95.50670
squash-unstored	93.04192	93.04192	93.04192	93.01250
synthetic control	99.13384	99.13384	99.13636	98.97980
vehicle	74.76969	74.76969	74.79478	74.42219
vowel	93.58258	93.58411	93.60706	92.87406
waveform-5000	91.31896	91.31956	91.33168	90.22170
wine	98.97811	98.97811	98.97811	98.77371
wisconsin-breast-cancer	98.29834	98.29834	98.28316	98.23550

Table 9. Average error on the training data for variants of FURIA

data set	prod	min	crisp	w/o
analcatdata-authorship	0.00540	0.00540	0.00540	0.00540
analcatdata-bankruptcy	1.60974	1.60974	1.60974	1.76125
analcatdata-cyyoung8092	4.57378	4.57378	4.57378	4.55791
analcatdata-cyyoung9302	2.98185	2.98185	2.98185	3.01464
analcatdata-esr	5.06558	5.06558	5.06558	5.06558
analcatdata-halloffame	1.16123	1.16123	1.18725	0.98257
analcatdata-lawsuit	0.59672	0.59672	0.59672	0.59672
analcatdata-votesurvey	2.82503	2.82503	2.82503	2.82503
biomed	1.13803	1.13803	1.12359	1.18860
cars	3.78576	3.78576	3.87160	3.57683
collins	0.00000	0.00000	0.00000	0.00000
ecoli	4.42350	4.42350	4.40562	4.45045
eucalyptus	10.25930	10.25930	10.21605	10.20780
glass	4.53830	4.53830	4.51697	4.51702
haberman	17.35651	17.35651	17.33671	17.27228
heart-statlog	5.15872	5.15872	5.14183	5.17557
ionosphere	1.39920	1.39920	1.39920	1.44680
iris	1.24242	1.24242	1.17172	1.27273
liver-disorders	10.95915	10.95915	10.91518	11.01625
metStatCoordinates	1.89078	1.89014	1.89397	1.92398
metStatRainfall	4.29721	4.29562	4.23562	4.33423
metStatRST	11.50757	11.50757	11.48498	11.49858
metStatSunshine	4.61678	4.61321	4.59160	4.67405
metStatTemp	7.76461	7.76461	7.76018	7.76904
mfeat-factors	0.05455	0.05455	0.05455	0.06136
mfeat-fourier	1.20985	1.20985	1.19394	1.23258
mfeat-karhunen	0.29091	0.29015	0.28636	0.31742
mfeat-morphological	15.98258	15.98258	15.95303	15.84545
mfeat-zernike	4.17803	4.17727	4.15379	4.19773
optdigits	0.01752	0.01752	0.01779	0.01806
page-blocks	1.39415	1.39637	1.53921	1.14417
pasture-production	1.12862	1.12862	1.12862	1.12862
pendigits	0.03722	0.03708	0.03667	0.04218
pima diabetes	14.12000	14.12000	14.07068	13.76685
prnn-synth	7.99655	7.99655	7.85726	8.02090
schizo-	2.40335	2.40335	2.36772	2.42565
segment	0.22497	0.22431	0.21907	0.23021
sonar	0.79367	0.79367	0.79367	0.81557
squash-unstored	2.75223	2.75223	2.75223	2.75223
synthetic control	0.06313	0.06313	0.06061	0.07323
vehicle	5.75072	5.75072	5.72563	5.76862
vowel	0.80510	0.80357	0.78063	0.85253
waveform-5000	0.33240	0.33180	0.31968	0.34210
wine	0.05939	0.05939	0.05939	0.08496
wisconsin-breast-cancer	0.42705	0.42705	0.44223	0.43569

References

[AFSG+09] Alcalá-Fernandez, J., Sánchez, L., García, S., del Jesus, M.J., Ventura, S., Garrell, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F.: KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Computing 13(3), 307–318 (2009)

[AN07] Asuncion, A., Newman, D.J.: UCI machine learning repository (2007), `http://archive.ics.uci.edu/ml/index.html` (Obtained on 22nd of August 2007)

[Bar07] Barker, D.: Dataset: Pasture production (2007), `http://weka.sourceforge.net/wiki/index.php/Datasets` (Obtained on 20th of October 2007)

[Bos04] Boström, H.: Pruning and exclusion criteria for unordered incremental reduced error pruning. In: Proceedings of the Workshop on Advances in Rule Learning, ECML, pp. 17–29 (2004)

[Bul07] Bulloch, B.: Dataset: Eucalyptus soil conservation (2007), `http://weka.sourceforge.net/wiki/index.php/Datasets` (Obtained on 20th of October 2007)

[Coh95] Cohen, W.W.: Fast effective rule induction. In: Prieditis, A., Russell, S. (eds.) Proceedings of the 12th International Conference on Machine Learning, ICML, Tahoe City, CA, USA, July 9-12, pp. 115–123. Morgan Kaufmann, San Francisco (1995)

[Cv06] Cloete, I., van Zyl, J.: Fuzzy rule induction in a set covering framework. IEEE Transactions on Fuzzy Systems 14(1), 93–110 (2006)

[CWY95] Chi, Z., Wu, J., Yan, H.: Handwritten numeral recognition using self-organizing maps and fuzzy rules. Pattern Recognition 28(1), 59–66 (1995)

[CYP96] Chi, Z., Yan, H., Pham, T.: Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition. World Scientific Publishing Co., Inc., River Edge (1996)

[Dem06] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

[Dun61] Dunn, O.J.: Multiple comparisons among means. Journal of the American Statistical Association 56, 52–64 (1961)

[FGHd07] Fernández, A., García, S., Herrera, F., del Jesús, M.J.: 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: Masulli, F., Mitra, S., Pasi, G. (eds.) WILF 2007. LNCS (LNAI), vol. 4578, pp. 170–178. Springer, Heidelberg (2007)

[Fri37] Friedman, M.: 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] Friedman, M.: A comparison of alternative tests of significance for the problem of *m* rankings. The Annals of Mathematical Statistics 11(1), 86–92 (1940)

[Für99] Fürnkranz, J.: Separate-and-Conquer rule learning. Artificial Intelligence Review 13(1), 3–54 (1999)

[FW94] Fürnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Cohen, W.W., Hirsh, H. (eds.) Proceedings of the 11th International Conference on Machine Learning, ICML, New Brunswick, NJ, USA, pp. 70–77. Morgan Kaufmann, San Francisco (1994)

[GP99] González, A., Perez, R.: Slave: a genetic learning system based on an iterative approach. IEEE Transactions on Fuzzy Systems 7(2), 176–191 (1999)

[GP01] González, A., Perez, R.: Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B 31(3), 417–425 (2001)

[Har07] Harvey, W.: Dataset: Squash harvest stored / unstored (2007), `http://weka.sourceforge.net/wiki/index.php/Datasets` (Obtained on 20th of October 2007)

[HH09] Hühn, J.C., Hüllermeier, E.: Furia: an algorithm for unordered fuzzy rule induction. In: Data Mining and Knowledge Discovery (2009)

[Hül05] Hüllermeier, E.: Fuzzy sets in machine learning and data mining: Status and prospects. Fuzzy Sets and Systems 156(3), 387–406 (2005)

[HV09] Hüllermeier, E., Vanderlooy, S.: Why fuzzy decision trees are good rankers. IEEE Transactions on Fuzzy Systems (2009)

[IN01] Ishibuchi, H., Nakashima, T.: Effect of rule weights in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 9(4), 506–515 (2001)

[IY05] Ishibuchi, H., Yamamoto, T.: Rule weight specification in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 13(4), 428–436 (2005)

[JCC07] Juang, C., Chiu, S., Chang, S.: A self-organizing TS-Type fuzzy network with support vector learning and its application to classification problems. IEEE Transactions on Fuzzy Systems 15(5), 998–1008 (2007)

[KNI08] Kuwajima, I., Nojima, Y., Ishibuchi, H.: Effects of constructing fuzzy discretization from crisp discretization for rule-based classifiers. Artificial Life and Robotics 13(1), 294–297 (2008)

[MV07] Meyer, M., Vlachos, P.: Statlib (2007), `http://lib.stat.cmu.edu/`

[PD03] Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning 52(3), 199–215 (2003)

[PF97] Provost, F.J., Fawcett, T.: Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: Heckerman, D., Mannila, H., Pregibon, D. (eds.) Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, KDD, Newport Beach, CA, USA, pp. 43–48. AAAI Press, Menlo Park (1997)

[PFK98] Provost, F.J., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. In: Shavlik, J.W. (ed.) Proceedings of the Fifteenth International Conference on Machine Learning, ICML, Madison, WI, USA, pp. 445–453. Morgan Kaufmann, San Francisco (1998)

[PFTV92] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1992)

[QCJ93] Quinlan, J.R., Cameron-Jones, R.M.: FOIL: A midterm report. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 3–20. Springer, Heidelberg (1993)

[Qui90] Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5(3), 239–266 (1990)

[Qui93] Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)

[Qui95] Quinlan, J.R.: MDL and categorial theories (continued). In: Prieditis, A., Russell, S.J. (eds.) Proceedings of the 12th International Conference on Machine Learning, ICML, Lake Tahoe, CA, USA, pp. 464–470. Morgan Kaufmann, San Francisco (1995)