Data Min Knowl Disc (2009) 19:293–319
DOI 10.1007/s10618-009-0131-8

FURIA: an algorithm for unordered fuzzy rule induction

Jens Hühn · Eyke Hüllermeier

Received: 21 November 2008 / Accepted: 3 April 2009 / Published online: 29 April 2009
Springer Science+Business Media, LLC 2009

Abstract This paper introduces a novel fuzzy rule-based classification method called FURIA, which is short for Fuzzy Unordered Rule Induction Algorithm. FURIA extends the well-known RIPPER algorithm, a state-of-the-art rule learner, while preserving its advantages, such as simple and comprehensible rule sets. In addition, it includes a number of modifications and extensions. In particular, FURIA 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. Experimental results show that FURIA significantly outperforms the original RIPPER, as well as other classifiers such as C4.5, in terms of classification accuracy.

Keywords Classification · Rule learning · Fuzzy logic · Rule stretching

1 Introduction

The learning of rule-based classification models has been an active area of research for a long time. In fact, the interest in rule induction goes far beyond the field of machine learning itself and also includes other fields, notably fuzzy systems (Hüllermeier 2005). 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

Responsible editor: Charles Elkan.

J. Hühn · E. Hüllermeier (✉)
Department of Mathematics and Computer Science, Philipps-Universität Marburg,
Mehrzweckgebäude, Hans-Meerwein-Straße, Marburg 35043, Germany
e-mail: eyke@informatik.uni-marburg.de

J. Hühn
e-mail: huehnj@informatik.uni-marburg.de

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 (Ishibuchi and Yamamoto 2005; Cloete and Van Zyl 2006; Juang et al. 2007; Fernández et al. 2007), offers scope for further improvements.

This paper proposes a 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 (Cohen 1995). In particular, FURIA 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.

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 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. 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 (Press et al. 1992).

The result 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) 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.[1] This approach has advantages but 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 one-vs-rest 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 2001). 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.

In the next section, we recall the basics of the RIPPER algorithm. In Sect. 3, we introduce FURIA and give a detailed explanation of its novelties. An experimental evaluation is presented in Sect. 4. Here, it is shown that FURIA significantly

[1] An interesting probabilistic interpretation of rules in a rule list was recently proposed by Fawcett (2008).

outperforms the original RIPPER, as well as other classifiers such as C4.5, in terms of classification accuracy. Besides, the impact of the different modifications distinguishing FURIA from RIPPER are investigated. Section 5 is devoted to related work. The paper ends with a summary and concluding remarks in Sect. 6.

2 Outline of RIPPER

RIPPER was introduced by Cohen (1995) as a successor of the IREP algorithm for rule induction (Fürnkranz and Widmer 1994). Even though the key principles remained unchanged, RIPPER improves IREP in many details and is also able to cope with multi-class problems.

Consider a polychotomous classification problem with m classes $\mathbb{L} \overset{\mathrm{df}}{=} \{\lambda_1, \ldots, \lambda_m\}$. 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

$$\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{D} \overset{\mathrm{df}}{=} \mathbb{D}_1 \times \ldots \times \mathbb{D}_n.$$

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 (class $= \lambda$), where $\lambda \in \mathbb{L}$. A rule $r = \langle r_A \,|\, r_C \rangle$ is said to *cover* an instance $\boldsymbol{x} = (x_1, \ldots, x_n)$ 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 (Fürnkranz 1999). 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 (Quinlan 1995): The stopping condition is fulfilled if the description length of r is at least d bits longer than the shortest description length encountered so far; Cohen suggests choosing $d = 64$.[2]

[2] Essentially, the description length of a rule depends on the number selectors in its premise part; see Quinlan (1993) for more details.

2.1 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.

Rule growing: A new rule is learned on the growing data, using a propositional version of the FOIL algorithm (Quinlan 1990; Quinlan and Cameron-Jones 1993).[3] 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:

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

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.

Rule pruning: 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.

For the pruning procedure, the antecedents are considered in the order in which they were learned, and pruning actually means finding a position at which that list of antecedents is cut. The criterion to find that position is the rule-value metric:

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

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.

2.2 Rule optimization

The ruleset *RS* produced by the learning algorithm outlined so far, called IREP*, 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

[3] Apart from RIPPER, several other rule learners have been built upon FOIL, for example the HYDRA algorithm by Kamal and Pazzani (1993).

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 (Minimum Description Length (Quinlan 1993) criterion is used. 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).

3 FURIA

This section presents the novel FURIA algorithm. Since FURIA builds upon the RIPPER algorithm, the corresponding modifications and extensions will be especially highlighted.

3.1 Learning unordered rule sets

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, Boström (2004) has proposed an unordered version of RIPPER's predecessor IREP (Fürnkranz and Widmer 1994). Likewise, we propose to learn a rule set for every single class, using a one-vs-rest 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.[4]

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 Sect. 3.5, 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, to be detailed in Sect. 3.6, is to modify the rules in a local way so as to make them applicable to the query.

3.2 Pruning modifications

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 (cf. Sect. 2.1) and the stopping conditions. Interestingly, we found that the pruning strategies in IREP* have a negative

[4] It is worth mentioning that, while Release 1 based on (Cohen 1995) only supported ordered rule lists, an unordered approach is also included in a more recent RIPPER implementation of Cohen (Release 2.5).

influence on the performance 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 Sect. 3.4, in which rules can be made more general but not more specific.

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.

3.3 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: $I = (-\infty, v]$ if the rule contains a selector $(A_i \leq v)$, $I = [u, \infty)$ if it contains a selector $(A_i \geq u)$, and $I = [u, v]$ 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 $I^F = (\phi^{s,L}, \phi^{c,L}, \phi^{c,U}, \phi^{s,U})$:

$$I^F(v) \overset{\text{df}}{=} \begin{cases} 1 & \phi^{c,L} \leq v \leq \phi^{c,U} \\ \frac{v-\phi^{s,L}}{\phi^{c,L}-\phi^{s,L}} & \phi^{s,L} < v < \phi^{c,L} \\ \frac{\phi^{s,U}-v}{\phi^{s,U}-\phi^{c,U}} & \phi^{c,U} < v < \phi^{s,U} \\ 0 & \text{else} \end{cases}$$

$\phi^{c,L}$ and $\phi^{c,U}$ are, respectively, the lower and upper bound of the core (elements with membership 1) of the fuzzy set; likewise, $\phi^{s,L}$ and $\phi^{s,U}$ are, respectively, the lower and upper bound of the support (elements with membership > 0), see Fig. 1.

Note that, as in the non-fuzzy case, a fuzzy interval can be open to one side ($\phi^{s,L} = \phi^{c,L} = -\infty$ or $\phi^{c,U} = \phi^{s,U} = \infty$). In fact, as will be seen later on, the fuzzy antecedents successively learned by FURIA are fuzzy half-intervals of exactly that kind.

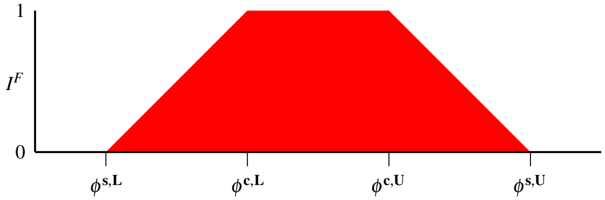

Fig. 1 A fuzzy interval I^F

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

$$\mu_{r^F}(\boldsymbol{x}) = \prod_{i=1\dots k} I_i^F(x_i). \tag{1}$$

3.4 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 $D_{\mathrm{T}} \subseteq \mathbb{D}$ 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 the intervals I_i of the original rules as the cores $[\phi_i^{c,L}, \phi_i^{c,U}]$ of the sought fuzzy intervals I_i^F, the problem is to find optimal bounds for the respective supports, i.e., to determine $\phi_i^{s,L}$ and $\phi_i^{s,U}$.

For the fuzzification of a single antecedent $(A_i \in I_i)$ it is important to consider only the relevant training data D_{T}^i, i.e., to ignore those instances that are excluded by any other antecedent $(A_j \in I_j^F)$, $j \neq i$:

$$D_{\mathrm{T}}^i = \left\{ \boldsymbol{x} = (x_1 \dots x_k) \in D_{\mathrm{T}} \mid I_j^F(x_j) > 0 \text{ for all } j \neq i \right\} \subseteq D_{\mathrm{T}} \tag{2}$$

We partition D_{T}^i into the subset of positive instances, $D_{\mathrm{T}+}^i$, and negative instances, $D_{\mathrm{T}-}^i$. To measure the quality of a fuzzification, the rule purity will be used:

$$\mathrm{pur} = \frac{p_i}{p_i + n_i}, \tag{3}$$

where

$$p_i \overset{\mathrm{df}}{=} \sum_{\boldsymbol{x} \in D_{\mathrm{T}+}^i} \mu_{A_i}(\boldsymbol{x})$$

$$n_i \overset{\mathrm{df}}{=} \sum_{\boldsymbol{x} \in D_{\mathrm{T}-}^i} \mu_{A_i}(\boldsymbol{x})$$

Algorithm 1 The antecedent fuzzification algorithm for a single rule r

```
Let $A$ be the set of numeric antecedents of $r$
while $A \neq \emptyset$ do
  $a_{\max} \leftarrow$ null {$a_{\max}$ denotes the antecedent with the highest purity}
  $\mathrm{pur}_{\max} \leftarrow 0$ {$\mathrm{pur}_{\max}$ is the highest purity value, so far}
  for $i \leftarrow 1$ to size($A$) do
    compute the best fuzzification of $A[i]$ in terms of purity
    $\mathrm{pur}_{A[i]} \leftarrow$ be the purity of this best fuzzification
    if $\mathrm{pur}_{A[i]} > \mathrm{pur}_{\max}$ then
      $\mathrm{pur}_{\max} \leftarrow \mathrm{pur}_{A[i]}$
      $a_{\max} \leftarrow A[i]$
    end if
  end for
  $A \leftarrow A \setminus a_{\max}$
  Update $r$ with $a_{\max}$
end while
```

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 (3). This is done by testing all values

$$\{x_i \mid \boldsymbol{x} = (x_1, \ldots, x_k) \in D_{\mathrm{T}}^i,\ x_i < \phi_i^{c,L}\}$$

as candidates for $\phi_i^{s,L}$ and, likewise, all values

$$\{x_i \mid \boldsymbol{x} = (x_1, \ldots, x_k) \in D_{\mathrm{T}}^i,\ x_i > \phi_i^{c,U}\}$$

as candidates for $\phi_i^{s,U}$ (see Fig. 2). 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 largest purity. This is repeated until all antecedents have been fuzzified. It is important to mention that there exists a trivial fuzzification which is always found, namely the one that sets 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.

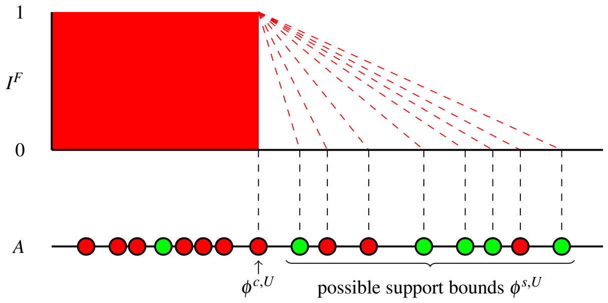

Fig. 2 Examination of possible support bounds given a crisp antecedent

Note that the fuzzification of a single antecedent may change the relevant training data (2), 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.

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

With regard to the readability of 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. It is valid inside the core and invalid outside the support; in-between, the validity drops in a gradual way. Consider, for example, the rule $\langle A \leq 5 \,|\, + \rangle$, 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 $\langle A \in (-\infty, -\infty, 5, 8) \,|\, + \rangle$ suggests that the rule is completely valid for $A \leq 5$, invalid for $A > 8$, and partially valid in-between.

3.5 Classifier output

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

$$s_j(\boldsymbol{x}) \overset{\mathrm{df}}{=} \sum_{i=1\ldots k} \mu_{r_i^{(j)}}(\boldsymbol{x}) \cdot CF\left(r_i^{(j)}\right) \ , \tag{4}$$

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

$$CF\left(r_i^{(j)}\right) = \frac{2\frac{|D_T^{(j)}|}{|D_T|} + \sum_{\boldsymbol{x} \in D_T^{(j)}} \mu_{r_i^{(j)}}(\boldsymbol{x})}{2 + \sum_{\boldsymbol{x} \in D_T} \mu_{r_i^{(j)}}(\boldsymbol{x})} \ , \tag{5}$$

where $D_T^{(j)}$ denotes the subset of training instances with label λ_j. Ishibuchi and Nakashima (2001); Ishibuchi and Yamamoto (2005) argued that weighing rules according to (4) allows for modeling more flexible decision boundaries and thereby improves classification accuracy. The certainty factor (5) is the m-estimate for $m = 2$ (Press et al. 1992).

The class predicted by FURIA is the one with maximal support. 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 in a separate way; see Sect. 3.6 below. In the case of a tie, a decision in favor of the class with highest frequency is made.

3.6 Rule stretching

To handle the aforementioned non-covering problem, Eineborg and Boström (2001) replace all rules by their *minimal generalizations* for the given instance. 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. 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, that 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 instances have to be stored. Alternatively, it is possible to pre-compute the evaluation of each possible generalization, but since a rule r with antecedent set $\mathscr{A}(r)$ can be generalized in $2^{|\mathscr{A}(r)|}$ different ways, this comes along with large storage requirements.

To avoid these disadvantages, we propose an alternative approach that exploits the order in which the antecedents had been 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. 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.[5]

Computationally, the above rule stretching strategy is much more efficient than EB-stretching. The complexity for re-evaluating a rule r is $\mathcal{O}(|\mathscr{A}(r)|)$. Moreover, since the

[5] For ease of presentation, we combined two selectors (half-intervals) referring to the same attribute into a single fuzzy interval in Sect. 3.3. It is important to mention that, in the context of rule stretching, the two selectors are still treated as different antecedents.

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.

4 Experimental results

To analyze the performance of our FURIA approach, we conducted several experimental studies under the WEKA 3.5.5 framework (Witten and Frank 2005). As a starting point, we used the RIPPER implementation of WEKA ("JRip") for re-implementing FURIA.

4.1 Classification accuracy

In a first study, we compared FURIA to other classifiers with respect to classification accuracy. The minimum number of covered instances per premise was set to 2, and for the number of folds and the number of optimizations in FURIA and RIPPER we used, respectively, values 3 and 2 (which is the default setting in WEKA and leads to RIPPER2).

Additionally, we also included the C4.5 decision tree learner (Quinlan 1993) as a well-known benchmark classifier and, moreover, added two fuzzy rule-based classifiers from the KEEL suite (Alcalá-Fernandez et al. 2009): The CHI algorithm is based on Chi et al. (1995, 1996) and uses rule weighing as proposed by Ishibuchi and Yamamoto (2005).[6] The SLAVE algorithm makes use of genetic algorithms to learn a fuzzy classifier (Gonzalez and Perez 1999, 2001).[7] Both algorithms are frequently used for experimental purposes (e.g., Fernández et al. 2007; Ishibuchi and Yamamoto 2003; Cordon et al. 2004; Zolghadri and Mansoori 2007).

We collected 40 data sets from the UCI (Asuncion and Newman 2007) and the STATLIB (Meyer and Vlachos 2007) repositories and from (Bulloch 2007; Barker 2007; Harvey 2007); see Table 1 for an overview. Additionally, we created five data sets with data from a German meteorological institute (DWD).[8] In these data sets, the task is to predict the origin (one of the federal states in Germany) of a set of measurements (e.g., sunshine duration, temperature,...). As our fuzzy extension is not applicable to nominal attributes, we only selected data sets having at least as many numeric as nominal attributes.

The experiments were conducted by randomly splitting each data set into 2/3 for training and 1/3 for testing, and deriving the classification accuracy on the testing data for each learner. This procedure was repeated 100 times to stabilize the results. Table 2 summarizes the classification accuracies.[9] The overall picture conveyed by the

[6] We used the following parameter setting: 3 fuzzy sets, product t-norm, maximum inference, and weighting scheme number 2 from (Ishibuchi and Yamamoto 2005).

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

[8] Available at: http://www.uni-marburg.de/fb12/kebi/research/repository.

[9] The classifier FURIA-c, which also appears in the table, will be analyzed in Sect. 4.2.

Table 1 Properties of the data sets used in the experiments: number of instances and classes, continuous (c) and nominal (n) attributes, and attributes with missing instances (m)

Data set	# Inst.	# Classes	# Attributes		
			c	n	m
analcatdata-authorship	841	4	70	0	0
analcatdata-bankruptcy	50	2	5	1	0
analcatdata-cyyoung8092	97	2	7	3	0
analcatdata-cyyoung9302	92	2	6	4	0
analcatdata-esr	32	2	2	0	0
analcatdata-halloffame	1340	3	15	2	1
analcatdata-lawsuit	264	2	3	1	0
analcatdata-votesurvey	48	4	3	1	0
biomed	209	2	7	1	2
cars	406	3	6	1	2
collins	500	15	20	3	0
ecoli	336	8	7	0	0
eucalyptus	736	5	14	5	9
glass	214	6	9	0	0
haberman	306	2	2	1	0
heart-statlog	270	2	13	0	0
ionosphere	351	2	34	0	0
iris	150	3	4	0	0
liver-disorders	345	2	6	0	0
metStatCoordinates	4748	16	3	0	0
metStatRainfall	4748	16	12	0	0
metStatRST	336	12	3	0	0
metStatSunshine	422	14	12	0	0
metStatTemp	673	15	12	0	0
mfeat-factors	2000	10	216	0	0
mfeat-fourier	2000	10	76	0	0
mfeat-karhunen	2000	10	64	0	0
mfeat-morphological	2000	10	6	0	0
mfeat-zernike	2000	10	47	0	0
optdigits	5620	10	64	0	0
page-blocks	5473	5	10	0	0
pasture-production	36	3	21	1	0
pendigits	10992	10	16	0	0
pima diabetes	768	2	8	0	0
prnn-synth	250	2	2	0	0
schizo-	340	2	12	2	11
segment	2310	7	19	0	0
sonar	208	2	60	0	0

Table 1 continued

Data set	# Inst.	# Classes	# Attributes		
			c	n	m
squash-unstored	52	3	20	3	8
synthetic control	600	6	60	1	0
vehicle	846	4	18	0	0
vowel	990	11	10	2	0
waveform-5000	5000	3	40	0	0
wine	178	3	13	0	0
wisconsin-breast-cancer	699	2	9	0	1

results is clearly in favor of FURIA, which outperforms the other methods on most data sets. To analyze the differences between the classifiers more closely, we followed the two-step procedure recommended by Demšar (2006): First, a Friedman Test is conducted to test the null hypothesis of equal classifier performance (Friedman 1937, 1940). In case this hypothesis is rejected, which means that the classifiers' performance differs in a statistically significant way, a posthoc test is conducted to analyze these differences in more detail.

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); see Table 3. 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, 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 (Iman and Davenport 1980):

$$\frac{(N-1)\cdot\chi_F^2}{N\cdot(k-1)-\chi_F^2} \tag{6}$$

In our case, the value of (6) 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 Nemenyi Test (Nemenyi 1963) as a posthoc test to compare classifiers in a pairwise manner. According to this test, the performance of two classifiers is significantly different if the distance of the

Table 2 Estimation of classification accuracies in terms of averages on the testing data (best per data set in bold, excluding separately analyzed FURIA-c)

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

Table 2 continued

Data set	FURIA	RIPPER	C4.5	CHI	SLAVE	FURIA-c
synthetic control	89.75	82.85	**90.00**	68.33	89.23	88.60
vehicle	70.10	67.80	**71.38**	61.99	64.08	69.75
vowel	75.43	64.71	**75.60**	59.49	63.84	71.87
waveform	**82.24**	78.72	75.05	72.38	75.34	82.23
wine	**93.25**	90.02	91.22	92.77	92.46	92.21
wisconsin-breast-cancer	**95.68**	95.58	94.51	90.20	95.49	95.53

average ranks exceeds the critical distance $CD_\alpha = q_{\alpha,k,\infty} \cdot \frac{1}{\sqrt{2}}$, where the q-value is taken from the Studentized Range Statistic (Newman 1939). The results of this test are summarized in Fig. 3: FURIA is significantly better than all other classifiers at the significance level $\alpha = 0.01$.

4.2 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. To answer this question, we conducted some additional experiments with a “crisp” variant of FURIA, included in Table 2 under the name FURIA-c. 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. Sect. 3.4). 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 FURIA-c still compares favorably to RIPPER (42 wins and 3 losses) and C4.5 (34 wins and 11 losses), the gains are less clear than those of FURIA. More importantly, in a direct comparison, FURIA achieves 38 wins. Besides, six of the seven data sets won by FURIA-c 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. A possible explanation for this finding is that fuzzy rules are able to generate more flexible decision boundaries which are smooth and not necessarily axis-parallel (see Fig. 4 for an illustration), which is especially advantageous for difficult problems.

4.3 Model complexity

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

Table 3 Ranks of the classifiers

Data set	FURIA	RIPPER	C4.5	CHI	SLAVE
analcatdata-authorship	1	3	2	5	4
analcatdata-bankruptcy	1	2	3	5	4
analcatdata-cyyoung8092	2	1	3	5	4
analcatdata-cyyoung9302	2	3	4	5	1
analcatdata-esr	2	1	3	4	5
analcatdata-halloffame	1	3	2	5	4
analcatdata-lawsuit	1	3	2	4	5
analcatdata-votesurvey	3	4	2	1	5
biomed	1	3	2	5	4
cars	2	3	1	5	4
collins	1	3	2	5	4
ecoli	1	4	2	5	3
eucalyptus	1	3	2	5	4
glass	1	3	2	5	4
haberman	3	4	5	2	1
heart-statlog	1	2	4	5	3
ionosphere	2	4	3	5	1
iris	2	4	3	5	1
liver-disorders	1	2	3	5	4
metStatCoordinates	1	3	2	5	4
metStatRainfall	1	2	3	5	4
metStatRST	4	3	2	5	1
metStatSunshine	1	3	2	4	5
metStatTemp	2	3	1	4	5
mfeat-factors	1	4	3	2	5
mfeat-fourier	1	4	2	5	3
mfeat-karhunen	1	4	3	2	5
mfeat-morphological	1	3	2	5	4
mfeat-zernike	1	5	3	2	4
optdigits	1	3	4	5	2
page-blocks	1	3	2	5	4
pasture-production	1	3	2	5	4
pendigits	1	4	3	2	5
pima diabetes	1	2	4	5	3
prnn-synth	2	4	3	1	5
schizo-	1	2	3	5	4
segment	1	3	2	5	4
sonar	1	3	4	2	5
squash-unstored	1	3	2	4	5
synthetic control	2	4	1	5	3

Table 3 continued

Data set	FURIA	RIPPER	C4.5	CHI	SLAVE
vehicle	2	3	1	5	4
vowel	2	3	1	5	4
waveform	1	2	4	5	3
wine	1	5	4	2	3
wisconsin-breast-cancer	1	2	4	5	3
average	1.40	3.07	2.60	4.24	3.69

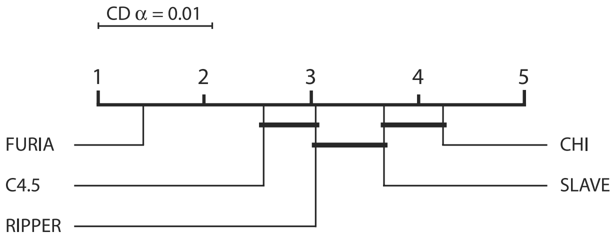

Fig. 3 Average classifier ranks depicted on a number line. Connections between classifiers indicate non-significant differences at significance level $\alpha = 0.01$

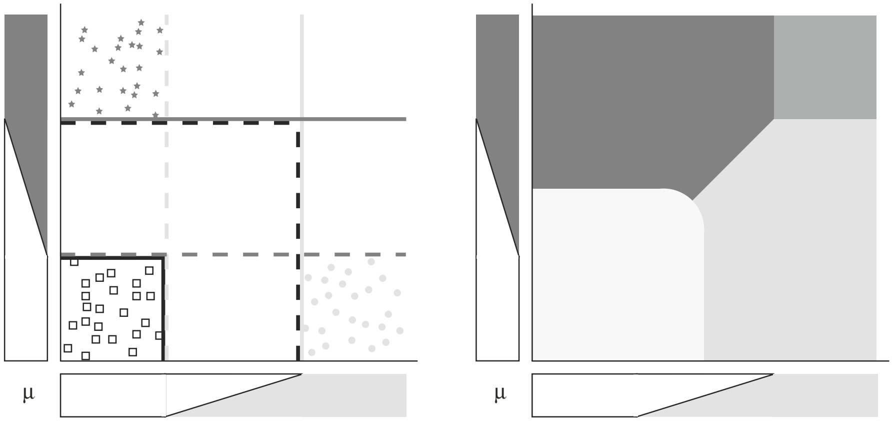

Fig. 4 The left figure shows three classes in the upper left, lower left, and lower right corner of the data space. Classification rules for every class are shown by the solid lines bounding the rule core and the dashed lines bounding the rule support. The membership functions are given on the bars on the bottom and on the left. The right picture shows the decision boundaries using the product-sum combination. As can be seen, there is a non-axis-parallel decision boundary between the three classes. The square in the upper right belongs to equal parts to the upper left and lower right classes

learns 25.4 rules on average, RIPPER generates only 15.5 rules.[10] Moreover, while a FURIA rule has 2.5 conditions on average, a RIPPER rule has only 1.7; see Table 4

[10] Including RIPPER's default rule.

Table 4 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.
analcatdata-authorship	15.9	2.7	9.6	1.7	555.1	3	12.1	5.9
analcatdata-bankruptcy	3.8	1.8	2.5	0.7	23.9	3	2.5	1.6
analcatdata-cyyoung8092	3.7	1.5	2.6	0.7	55	3	3.4	2.0
analcatdata-cyyoung9302	3.5	1.3	2.8	0.8	49.5	3	3.1	2.0
analcatdata-esr	2.1	1.1	2	0.5	6.8	3	2.6	1.1
analcatdata-halloffame	14.3	2.8	6.5	1.8	458.9	3	7.2	3.4
analcatdata-lawsuit	3.7	1.5	2	1	24.7	3	2.6	1.7
analcatdata-votesurvey	1.7	1.4	2.3	0.8	13.6	3	7.3	2.1
biomed	8.6	2	4.4	1	55.1	3	4.3	2.7
cars	12.9	2.4	7.1	1.8	54.4	3	12.5	3.3
collins	15.9	1.1	15.2	1	321.7	3	45.8	6.6
ecoli	13.8	2.5	8.3	1.6	47.2	3	11.3	3.0
eucalyptus	14.7	2.6	10.2	1.8	375	3	38.3	5.7
glass	11.3	2.2	6.7	1.7	42.7	3	12.3	3.3
haberman	4.4	1.5	2	0.8	15.8	3	4	1.7
heart-statlog	8.4	2.5	4.3	1.5	164.9	3	7	3.6
ionosphere	8.3	2	4.7	1.1	168.9	3	8	3.8
iris	4.4	1.5	3.3	0.8	14.9	3	3.1	1.2
liver-disorders	8.2	2.2	4.3	1.8	42.1	3	5.9	3.4
metStatCoordinates	69.7	2.3	38.8	2.1	15.6	3	12.8	2.4
metStatRainfall	123.9	4.5	82.7	3.6	215.6	3	30.3	4.4
metStatRST	9.9	2.2	10	1.7	15	3	9.5	2.3
metStatSunshine	25	2.7	17	1.9	91	3	39.3	4.3
metStatTemp	31.5	2.8	22.4	2.2	36.4	3	15.6	3.5
mfeat-factors	45	3	28.5	2.2	1317.2	3	44.3	12.2
mfeat-fourier	52.4	3.8	29.2	2.6	1317.2	3	73.2	10.6
mfeat-karhunen	59.1	3.2	38.4	2.6	1314.4	3	64.7	9.7
mfeat-morphological	25.1	2.6	19	2.1	31.4	3	15.7	3.1
mfeat-zernike	44.9	3.7	30.6	2.8	1257.6	3	77.7	10.0
optdigits	97.8	4.9	59.6	3.9	3708.5	3	68.6	8.0
page-blocks	25.6	3.2	14.7	2.2	47.6	3	10.1	3.5
pasture-production	3.4	1.4	3.2	0.7	24	3	3.6	3.3
pendigits	110.9	4.8	67.6	3.4	2745.2	3	37	7.6
pima diabetes	8.5	2.6	3.9	1.8	98.6	3	9.3	3.7
prnn-synth	4.4	1.4	3.5	1	8	3	2.4	1.6
schizo-	15.1	1.7	6.5	1.2	136.7	3	7.9	5.3
segment	26.9	3.1	17	2.2	275.1	3	15.8	4.5
sonar	8.1	2.3	4.3	1.4	137.1	3	6.9	4.7
squash-unstored	4	1.5	3.2	0.8	33.8	3	4.1	2.6

Table 4 continued

Data set	FURIA		RIPPER		CHI		SLAVE	
	Rules	Cond.	Rules	Cond.	Rules	Cond.	Rules	Cond.
synthetic control	17.3	2.6	10.6	1.8	394.3	3	9.1	6.3
vehicle	20.7	3.3	13.8	2.2	314.4	3	26.4	6.5
vowel	53.8	3.3	34.2	2.5	251.5	3	51.1	5.6
waveform	79.9	5.9	27.9	3.8	2874.7	3	50.7	9.3
wine	6.2	1.9	3.5	1.1	101.2	3	3.8	2.9
wisconsin-breast-cancer	12.2	2.9	4.8	1.5	172.4	3	5.8	3.7
average	25.4	2.5	15.5	1.7	431.7	3	19.8	4.4

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 rule sets with long condition parts.

4.4 The effect of the rule stretching algorithm

To investigate the effectiveness of our novel rule stretching method, we compared it to the original EB-stretching of Eineborg and Boström (2001). 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. 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 better than 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 5), 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 a repeated calculation of the same weights, we cached the weights in EB-stretching. Table 5 shows the results of this comparison, and Fig. 5 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

Table 5 The number of times the rule weight has to be calculated for the rule stretching procedure

Data set	Uncovered	Novel	(Eineborg and Boström 2001)	Diff.
analcatdata-authorship	0.05	42.70	49.40	6.70
analcatdata-bankruptcy	0.02	6.80	0.80	−6.00
analcatdata-cyyoung8092	0.08	5.50	1.80	−3.70
analcatdata-cyyoung9302	0.07	4.70	1.10	−3.60
analcatdata-esr	0.04	2.30	0.20	−2.10
analcatdata-halloffame	0.03	39.80	48.50	8.70
analcatdata-lawsuit	0.00	5.50	0.70	−4.80
analcatdata-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

Table 5 continued

Data set	Uncovered	Novel	(Eineborg and Boström 2001)	Diff.
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
wisconsin-breast-cancer	0.02	34.90	26.11	−8.79

The first column shows the relative number of testing instances that were not covered by any rule. The second column shows the number of calculations for our approach and the third column for the one of Eineborg and Boström (2001). The last column shows the difference of these two

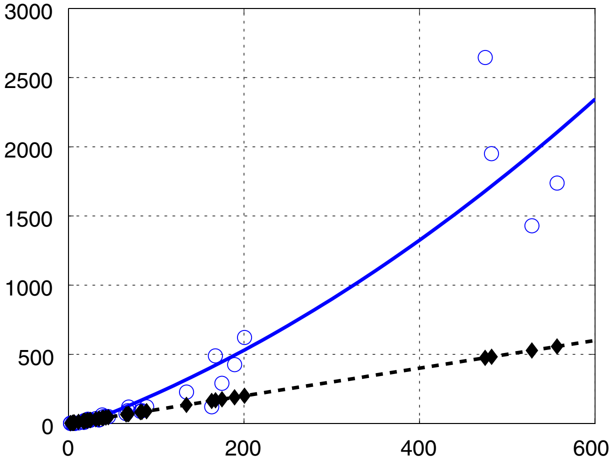

Fig. 5 The number of weight calculations (y-axis) as a function of the number of antecedents (x-axis) for our rule stretching method (*diamonds*) and EB-stretching (*circles*). Trends are shown in terms of corresponding regression curves

with the theoretical considerations in Sect. 3.6. Thus, from a complexity point of view, our approach is especially advantageous for complex models.

As another advantage, recall that our approach does not have 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 (Aha et al. 1991). We tested this idea and, interestingly enough, found that FURIA in combination with a simple 1-NN classifier outperforms FURIA with EB-stretching for 35 out of 45 data sets.

4.5 Runtime

It is clear that FURIA, as an extension of RIPPER encompassing more complex strategies such as fuzzification and rule stretching, will pay 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 rule set again without IREP (RIP**)
- FURIA without post generalization or fuzzification of rules (FUR*)
- FURIA

Table 6 shows the runtime results in seconds (all measurements were performed on a Intel Core2Duo 2.4 Ghz). 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, growing data is larger at the beginning). A further increase in runtime is caused by changing from an ordered rule list to the unordered rule set (RIP**). This is also expected since the unordered version learns rules in a one-vs-all fashion, while for the ordered variant, the training data becomes successively smaller (training instances from already covered classes are dropped). There is not much difference between the unordered RIPPER without IREP (RIP**) to FURIA without fuzzification or crisp 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 quintessence 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).

5 Related work

Since the literature on (fuzzy) rule learning abounds, a comprehensive survey of this field is clearly beyond the scope of this paper. Nevertheless, this section is meant to convey a rough picture of the field and to briefly mention some related work.

The field of fuzzy rule learning can be roughly separated into several subfields. Firstly, there are fuzzy extensions of conventional rule learning techniques, not only for the propositional case but also for the case of first-order logic (Drobics et al. 2003; Prade et al. 2003; Serrurier and Prade 2007). Quite popular in the fuzzy field are grid-based approaches as popularized by Wang and Mendel (1992), which proceed from fixed fuzzy partitions of the individual dimensions. They are not very flexible and suffer from the "curse of dimensionality" in the case of many input variables but may have advantages with respect to interpretability (Guillaume 2001). A well-known representative of this kind of approach is the CHI algorithm that we also used in our experiments (Chi et al. 1995, 1996). It proceeds from a fuzzy partition for each attribute and learns a rule for every grid cell. This is done by searching the training instance with maximal degree of membership in this cell (matching degree of the rule premise) and adopting the corresponding class attribute as the rule consequent. Another

Table 6 Average model building times in seconds

Data set	RIPPER	RIP*	RIP**	FUR*	FURIA
analcatdata-authorship	0.348	0.588	0.828	0.840	0.873
analcatdata-bankruptcy	0.001	0.002	0.003	0.003	0.003
analcatdata-cyyoung8092	0.004	0.005	0.012	0.010	0.010
analcatdata-cyyoung9302	0.003	0.004	0.008	0.008	0.008
analcatdata-esr	0.000	0.000	0.001	0.001	0.001
analcatdata-halloffame	0.461	0.706	0.996	1.008	1.046
analcatdata-lawsuit	0.003	0.003	0.006	0.005	0.007
analcatdata-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	5.6679
pasture-production	0.002	0.005	0.006	0.006	0.0071
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

Table 6 continued

Data set	RIPPER	RIP*	RIP**	FUR*	FURIA
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
wisconsin-breast-cancer	0.025	0.053	0.085	0.102	0.117

FUR* = FURIA w/o fuzzification process, R* = RIPPER unordered w/o IREP, R** = RIPPER w/o IREP

approach of this subfield, which prevails the literature on conventional rule learning but has received less attention in the fuzzy field so far, is rule covering algorithms (Cloete and Van Zyl 2006). In this category, the FR3 rule learner that has recently been proposed by Hühn and Hüllermeier (2009) deserves special mentioning. Just like FURIA, FR3 draws on the RIPPER algorithm and modifies it in a quite similar way. However, FR3 has a completely different focus and embeds the modified RIPPER algorithm in a round robin learning scheme, i.e., it learns an ensemble of binary classification models, one for each pair of classes. As opposed to this, FURIA learns a single multi-class model.

Secondly, several fuzzy variants of *decision tree learning*, following a divide-and-conquer strategy and producing rule sets of a special (hierarchical) structure, have been proposed (Wang et al. 2007). As this direction is only indirectly related to our work, we do not go into further details.

Thirdly, *hybrid methods* that combine fuzzy set theory with other (soft computing) methodologies, notably evolutionary algorithms and neural networks, are especially important in the field of fuzzy rule learning. For example, evolutionary algorithms are often used to optimize ("tune") a fuzzy rule base or for searching the space of potential rule bases in a (more or less) systematic way (Cordon et al. 2004). One of these classifiers, which was also included in our experimental comparison, is the SLAVE classifier (Gonzalez and Perez 1999, 2001). It uses a genetic learning approach to create a fuzzy rule-based system by following a covering scheme. SLAVE represents each rule as a single chromosome. It uses an iterative approach, which means that the result of the genetic algorithm is not meant to cover all positive examples. Instead, the genetic algorithm is repeated until the iteratively generated set of rules is sufficient to represent the training set. Another interesting approach in this area is the one proposed by del Jesus et al. (2004), which applies the idea of boosting (Kearns 1988) to the evolutionary learning of rule-based classifiers. *Neuro-fuzzy* methods (Mitra and Hayashi 2000; Nauck et al. 1997) encode a fuzzy system as a neural network and apply corresponding learning methods (like backpropagation). Fuzzy rules are then extracted from a trained network.

Finally, with regard to the idea of rule stretching that we proposed in this paper, it is worth mentioning that some other approaches have been proposed in the literature that are closely related, in particular the idea to combine instance-based and rule learning. This idea has been realized in the RISE system (Domingos 1995), in which single instances are considered as maximally specific rules. The learning procedure essentially tries to aggregate specific rules into more general ones. At classification time, an instance is classified by the nearest rule. Another combination of instance-based and rule learning is proposed by Hendrickx and van den Bosch (2005), who make use of the rule set to create new features, indicating whether or not a rule is activated for an instance. A classification is made by searching the query's nearest neighbor in the novel feature space and assigning its class.

6 Concluding remarks

In this paper, 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. This way, 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 classification performance, 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 purpose.

A Java implementation of FURIA, running under the open-source machine learning toolkit WEKA, can be downloaded at http://www.uni-marburg.de/fb12/kebi/research/.

Acknowledgements This research was supported by the German Research Foundation (DFG) and the Konrad Adenauer Foundation (KAS). The CHI and SLAVE classifiers were made available to us by the developers of the KEEL software (Alberto Fernández). We gratefully acknowledge this support. We also thank Johannes Fürnkranz for insightful discussions about rule fuzzification and generalization.

References

Aha D, Kibler D, Albert M (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66

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

Asuncion A, Newman D (2007) UCI machine learning repository. http://archive.ics.uci.edu/ml/index.html. Accessed 20 Aug 2007

Barker D (2007) Dataset: pasture production. http://weka.sourceforge.net/wiki/index.php/Datasets. Accessed 20 Oct 2007

Boström H (2004) Pruning and exclusion criteria for unordered incremental reduced error pruning. Proceedings of the workshop on advances in rule learning, ECML, pp 17–29

Bulloch B (2007) Dataset: eucalyptus soil conservation. http://weka.sourceforge.net/wiki/index.php/Datasets. Accessed 20 Oct 2007

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

Chi Z, Yan H, Pham T (1996) Fuzzy algorithms: with applications to image processing and pattern recognition. World Scientific Publishing Co., Inc., River Edge

Cloete I, Van Zyl J (2006) Fuzzy rule induction in a set covering framework. IEEE Trans Fuzzy Syst 14(1): 93–110

Cohen W (1995) Fast effective rule induction. In: Prieditis A, Russell S (eds) Proceedings of the 12th international conference on machine learning, ICML. Morgan Kaufmann, Tahoe City, pp 115–123

Cordon O, Gomide F, Herrera F, Hoffmann F, Magdalena L (2004) Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst 141(1):5–31

del Jesus M, Hoffmann F, Navascues L, Sd́fnchez L (2004) Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Syst 12(3):296–308

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Domingos P (1995) Rule induction and instance-based learning: a unified approach. In: Proceedings of the fourteenth international joint conference on artificial intelligence, IJCAI. Morgan Kaufmann, Montral, pp 1226–1232

Drobics M, Bodenhofer U, Klement E (2003) FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. Int J Approx Reason 32(2–3):131–152

Eineborg M, Boström H (2001) Classifying uncovered examples by rule stretching. In: ILP '01: Proceedings of the 11th international conference on inductive logic programming. Springer-Verlag, London, pp 41–50

Fawcett T (2008) Prie: a system for generating rulelists to maximize roc performance. Data Min Knowl Discov 17(2):207–224

Fernández A, García S, Herrera F, del Jesus M (2007) 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, vol 4578, Lecture notes in computer science. Springer, Berlin/Heidelberg, pp 170–178

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of *m* rankings. Ann Math Stat 11(1):86–92

Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intell Rev 13(1):3–54

Fürnkranz J, Widmer G (1994) Incremental reduced error pruning. In: Proceedings of the 11th international conference on machine learning, ICML, pp 70–77

Gonzalez A, Perez R (1999) Slave: a genetic learning system based on an iterative approach. IEEE Trans Fuzzy Syst 7(2):176–191

Gonzalez A, Perez R (2001) Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Trans Syst Man Cyber B 31(3):417–425

Guillaume S (2001) Defining fuzzy inference systems from data: an interpretability-oriented review. IEEE Trans Fuzzy Syst 9(3):426–443

Harvey W (2007) Dataset: squash harvest stored/unstored. http://weka.sourceforge.net/wiki/index.php/Datasets. Accessed on 20 Oct 2007

Hendrickx I, van den Bosch A (2005) Hybrid algorithms for instance-based classification. In: Gama J, Camacho R, Brazdil P, Jorge A, Torgo L (eds) Proceedings of the sixteenth European conference on machine learning, ECML. Springer, Berlin, pp 158–169

Hühn J, Hüllermeier E (2009) FR3: a fuzzy rule learner for inducing reliable classifiers. IEEE Trans Fuzzy Syst 17(1):138–149

Hüllermeier E (2005) Fuzzy sets in machine learning and data mining: status and prospects. Fuzzy Sets Syst 156(3):387–406

Iman R, Davenport J (1980) Approximations of the critical region of the friedman statistic. Commun Stat 9(6):571–595

Ishibuchi H, Nakashima T (2001) Effect of rule weights in fuzzy rule-based classification systems. IEEE Trans Fuzzy Syst 9(4):506–515

Ishibuchi H, Yamamoto T (2003) Performance evaluation of three-objective genetic rule selection. In: The 12th IEEE international conference on fuzzy systems, vol 1, pp 149–154

Ishibuchi H, Yamamoto T (2005) Rule weight specification in fuzzy rule-based classification systems. IEEE Trans Fuzzy Syst 13(4):428–436

Juang C, Chiu S, Chang S (2007) A self-organizing ts-type fuzzy network with support vector learning and its application to classification problems. IEEE Trans Fuzzy Syst 15(5):998–1008

Kamal M, Pazzani M (1993) Hydra: a noise-tolerant relational concept learning algorithm. In: Proceedings of the thirteenth international joint conference on artificial intelligence, IJCAI. Morgan Kaufmann, Chambry, pp 1064–1071

Kearns M (1988) Thoughts on hypothesis boosting. ML class project

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

Mitra S, Hayashi Y (2000) Neuro-fuzzy rule generation: survey in soft computing framework. IEEE Trans Neural Netw 11(3):748–768

Nauck D, Klawonn F, Kruse R (1997) Foundations of neuro-fuzzy systems. Wiley, Chichester

Nemenyi P (1963) Distribution-free multiple comparisons. PhD thesis, Princeton University

Newman D (1939) The distribution of range in samples from a normal population expressed in terms of an independent estimate of standard deviation. Biometrika 31:20–30

Prade H, Richard G, Serrurier M (2003) Enriching relational learning with fuzzy predicates. In: Proceedings of PKDD–03 European conference on principles and practice of knowledge discovery in databases, pp 399–410

Press W, Flannery B, Teukolsky S, Vetterling W (1992) Numerical recipes in FORTRAN: the art of scientific computing, 2nd edn. Cambridge University Press

Quinlan J (1990) Learning logical definitions from relations. Mach Learn 5(3):239–266

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

Quinlan J (1995) MDL and categorial theories (continued). In: Proceedings of the 12th international conference on machine learning, ICML. Morgan Kaufmann, Lake Tahoe, pp 464–470

Quinlan J, Cameron-Jones R (1993) Foil: a midterm report. In: Proceedings of the 6th European conference on machine learning, ECML. Springer-Verlag, London, pp 3–20

Serrurier M, Prade H (2007) Introducing possibilistic logic in ILP for dealing with exceptions. Artif Intell 171(16–17):939–950

Wang L, Mendel J (1992) Generating fuzzy rules by learning from examples. Trans Syst Man Cyber 22(6):1414–1427

Wang T, Li Z, Yan Y, Chen H (2007) A survey of fuzzy decision tree classifier methodology. In: Proceedings of the second international conference of fuzzy information and engineering, vol 40 of advances in soft computing. Springer-Verlag, Berlin/Heidelberg

Witten I, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco

Zolghadri M, Mansoori E (2007) Weighting fuzzy classification rules using receiver operating characteristics (roc) analysis. Inf Sci 177(11):2296–2307