16th World Congress of the International Fuzzy Systems Association (IFSA)
9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)

Interpretability improvement of fuzzy rule-based classifiers via rule compression

Andri Riid[1] **Jürgo-Sören Preden**[2]

[1] Laboratory for Proactive Technologies, Tallinn University of Technology Ehitajate tee 5, Tallinn 19086, Estonia, Email: andri.riid@ttu.ee
[2] Laboratory for Proactive Technologies, Tallinn University of Technology Ehitajate tee 5, Tallinn 19086, Estonia, Email: jurgo.preden@ttu.ee

Abstract

Rule-level feature selection, also termed as rule compression, is an important technique for improving interpretability of fuzzy rule-based classifiers. In this paper we present three different rule compression algorithms and analyze their performance and characteristics on the classifiers identified from well-known classification benchmarks, namely the Iris, Wine and two versions of Wisconsin Breast Cancer data sets. Our study shows that the classifiers, in which the overlap between either the rules representing different classes or all rules is eliminated, can be usually compressed at a higher rate and that the interpretation of such classifiers is more insightful.

Keywords: Rule-based classification, feature selection, interpretability

1. Introduction

In last two decades, fuzzy rule based classifiers have earned substantial recognition because of their ability to explain the reasoning that lies behind assigning an object into any given class. However, intepretability is not a default property of fuzzy systems and over a decade or so, considerable effort has been made to find out what comprises interpretability of fuzzy systems and how to preserve it [2].

Interpretability of fuzzy systems is divided into low-level and high-level interpretability [14] Low-level interpretability can be tracked down to fuzzy set and partition properties such as normality, continuity, convexity, coverage, distinguishability, complementarity, partition cardinality, etc. and is usually achieved by imposing constraints on membership function (MF)/partition parameters. High-level interpretability, on the other hand, is associated with rule base properties such as the number of features, number of rules, number of conditions, consistency of the rule base, etc. and is generally obtained by complexity reduction.

Today, the weapon of choice for obtaining interpretable fuzzy systems seems to be multi-objective evolutionary optimization [2, 4, 5] where one or several interpretability measures plus a measure of accuracy are included in the optimization criterion to strike a good balance between accuracy and interpretability (not a trivial task because of the well-known interpretability-accuracy tradeoff).

The described concept of interpretability leads to a partition-driven interpretation of fuzzy systems - one can view the partitions of input variables composed of fuzzy sets that are labelled by linguistic labels setting a global semantics and read the IF-THEN rules that describe the relationships between these labels quite like the sentences in everyday human language.

However, the number of rules in a fuzzy system grows exponentially as the number of variables increases (curse of dimensionality) and low granularity of input partitions is often a necessity that results in a limited understanding of the modelled phenomenon. This particularly concerns classification applications where we typically deal with a large number of features.

On the other hand, it has been shown that when implemented with the most common single-winner method, classification rules do not cooperate in producing the output for the fuzzy system but compete with each other [8], which has lead to the suggestion that low-level interpretability is not so important for classification systems [9] - this applies most notably for such partition properties as distinguishability and complementarity. It then allows one to create the rules directly in product space that helps to cope with the curse of dimensionality [7] (the number of rules is relatively low and does not depend on the number of features). This, however, comes at the cost of loss of global semantics, making it impossible to use meaningful linguistic labels for fuzzy sets and invalidates the partition-driven interpretation – therefore one has to revert to the rule-driven interpretation of a fuzzy system [10].

In the context of the latter, interpretability improvement is a matter of finding a small number of concise fuzzy rules (not too spread out in product space) with a limited number of conditions [9]. The degree at which the rules in a classifier overlap is also an important factor that influences how understandable the rules of the system can be [10].

Depending on the type of rule overlap, rule-based classifiers can be divided into three categories:

1. Classifiers in which the rule overlap is not reg-

ulated at all (type 0 overlap control);
2. Classifiers in which the overlap between the rules that represent different classes is not permitted (type 1 overlap control);
3. Classifiers in which no rules overlap (type 2 overlap control).

In general, fuzzy rule-based classification is a prime example of the first approach. It takes advantage of the rule overlap that provides oblique decision boundaries that allows one to get high classification accuracy even with a small number of rules. The shape of the boundary mostly depends on the placement of overlapping rules in respect to each other. The main effort in this line of research therefore boils down to the optimization of MF parameters so as to improve classification accuracy.

Perhaps the best known representative of the second approach are the fuzzy min-max networks [11] that allow one to obtain fuzzy classifiers with more concise, well separated rules that can be more easily interpreted. However, higher level of granularity is usually required to achieve the same level of accuracy than in decision boundary optimization.

The third approach is represented by (non-fuzzy) classification trees (C4.5, CART etc.) that can be presented in rule-based format [3] with even higher level of granulation.

In this context, we focus on the part of complexity reduction/interpretability improvement that aims at reducing the number of conditions in classification rules (rule compression) and considers all three types of classification systems that are obtained by applying the algorithms described in Sections 2 and 3). The goal of rule compression is to detect the redundant conditions that can be removed from the rules without accuracy loss. We have developed three rule compression algorithms (Section 4) that exhibit slightly different characteristics, which are analyzed in more detail in Section 5.

Note that we employ full benchmark data sets and identify the classifiers that are 100% accurate before and after compression. This is deliberate for two reasons. First, we want to compare the considered compression methods on equal grounds - using the same and as complete as possible data sets. Our second goal is to emphasize the ability of compressed classifiers to explain the class distributions in a comprehensible manner (see Section 5), which, again, makes more sense on full data sets with no misclassified samples.

2. Preliminary classifiers

A fuzzy classifier is a fuzzy rule-based system that utilizes fuzziness only in its reasoning mechanism and groups the examples presented to it into a small number of distinct classes that are labelled with discrete values $(1, 2, ..., T)$ where T is the number of classes. Note that the actual numerical value assigned to a class is irrelevant, it just functions as a label. A fuzzy classifier consists of rules in the following format

$$\begin{array}{c}\text{IF } x_1 \text{ is } A_{1r} \text{ AND } x_2 \text{ is } A_{2r} \text{ AND ... AND} \\ x_N \text{ is } A_{Nr} \text{ THEN } y \text{ belongs to class } c_r\end{array} \quad (1)$$

where c_r is a class assigned to the r-th rule ($c_r \in \{1, ..., T\}$) and A_{ir} denote the linguistic labels of the i-th feature associated with the r-th rule ($i = 1, ..., N$).

Each A_{ir} has its representation in the numerical domain - a typically normal and convex membership function μ_{ir} such as a triangular MF determined by three parameters a_{ir}, b_{ir} and c_{ir}:

$$\mu_{ir}(x_i) = \begin{cases} \frac{x_i - a_{ir}}{b_{ir} - a_{ir}}, & a_{ir} < x_i < b_{ir} \\ \frac{c_{ir} - x_i}{c_{ir} - b_{ir}}, & b_{ir} \leq x_i < c_{ir} \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

The reasoning mechanism of a fuzzy rule-based classifier is usually implemented by the single winner approach that selects the class label c_r, associated with the rule that provides the highest rule activation degree (τ_r) for the given set of feature values x_i

$$y = c_r, \arg \max_{1 \leq r \leq R}(\tau_r), \quad (3)$$

where

$$\tau_r = \bigcap_{i=1}^{N} \mu_{ir}(x_i), \quad (4)$$

where $\cap_i^N$ is the conjunction operator corresponding to the linguistic operator AND (most often a minimum or product operator) that has a marked effect on how the decision boundary is drawn between highly overlapping rules. In present paper we apply product implication that yields smoother decision boundaries [9].

The goal in fuzzy rule-based classification is to obtain the maximum possible classification accuracy with as simple classifier as possible. Classification accuracy that a data driven fuzzy rule-based classifier is able to achieve first and foremost depends on the properties of the data set. A class that is separated from other classes in product space is easy to classify correctly whereas high overlap of classes can make it very difficult to obtain a rule placement that would result in an accurate classifier and typically, such class distributions need to be modeled with increased level of granularity. For this purpose we employ the algorithms of rule granulation and consolidation [10] to obtain the zero-error classifiers on which to verify the performance of compression methods described in Section 4.

The classifiers of considered data sets are initialized as minimal rule classifiers (MRC) that specify only one rule for each class. To obtain these, the training data set is divided into T subsets so that each subset S_r contains only the samples belonging to one of T classes.

Given a subset of data S_r that contains K_r observations and its mean $\mathbf{m}_r = (m_{1r}, m_{2r}, ..., m_{Nr})$

that is the geometric centroid of the data points in S_r

$$\mathbf{m}_r = \sum_{k \in S_r} \mathbf{x}_k / K_r, \tag{5}$$

the MFs μ_{ir} of form (2) are created in all dimensions i. Given a predefined value of $\alpha < 1$, the MF parameters a_{ir}, b_{ir}, c_{ir} are obtained as follows. For each i

$$\begin{aligned} a_{ir} &= \frac{\min\limits_{k \in S_r}(x_i(k)) - \alpha \cdot m_{ir}}{1-\alpha}, \\ b_{ir} &= m_{ir}, \\ c_{ir} &= \frac{\max\limits_{k \in S_r}(x_i(k)) - \alpha \cdot m_{ir}}{1-\alpha}. \end{aligned} \tag{6}$$

Following this a rule of format (1) is constructed. Note that what we described here is a general subset modeled rule generation/modification procedure used as well at the later stages of the algorithm.

Also note that as the rules of a MRC are usually overlapping, rule competition steps in and as a result a number of samples are redistributed among the rules ending up as misclassified samples. The number of misclassified samples in a r-th rule is denoted by η_r and called local error. The global error (η) is given by

$$\eta = \sum_{r=1}^{R} \eta_r. \tag{7}$$

Classification error reduction via rule granulation is carried out by a sequence of rule splits so that at each iteration a parent rule is selected and split into two offspring rules. The offspring rules replace the parent rule, which means that at each iteration the number of classification rules increases by one. Usually there is a number of choices on which parent rule to pick and how to make the split. The first choice for the parent rule is a rule p with the highest local error

$$p = r, \arg\max_{1 \le r \le R}(\eta_r). \tag{8}$$

If there are several rules with the same local error, we simply choose the one with the highest K_r of those.

The rule splitting cut can be made around each erroneous sample under the parent rule. At given iteration, a single cut is allowed at one of N coordinates, thus the overall number of potential rule splits at the iteration equals $N \times \eta_p$.

A cut divides the K_p samples of the parent rule into two subsets S_o and S_q that form the basis of two offspring rules, R_o and R_q. Note that the erroneous sample is always sided with the offspring rule that contains less samples. Of available cuts the one that results in the best performing classifier (yielding the smallest η) is selected. It is possible that there are several cuts that result in classifiers with the same number of erroneous samples. In this case we choose the cut that has the minimal value of $\max(\eta_o, \eta_q)$ - generally this leads to faster convergence. If this still leaves us several equally good candidates, we choose the cut that has the smallest value of $\min(K_o, K_q)$. The granulation continues until η reaches zero.

This procedure, however, usually creates too many rules the number of which can be substantially reduced by rule base consolidation [10]. During the consolidation, weaker rules (governing few samples) are constantly losing their samples to stronger rules (those governing many samples) and as a natural result, many of the weaker rules become obsolete.

The rules are ranked by their strength (the number of samples they govern) in ascending order $p \in \{1, ..., R\}$. The process starts from the lowest ranked rule ($p = 1$).

1. pick a rule R_r with the rank p
2. pick k-th sample ($k = 1, ..., K_r$) from the subset S_r governed by rule R_r.
3. transfer this sample from S_r to the subset S_q corresponding to R_q, the next rule in the ranking that matches the class of the sample ($c_q = y_k$).
4. update the MFs of both R_r and R_q on the basis of modified subsets S_r and S_q, respectively.

First, prior to accepting the transfer, we need to verify that there is no accuracy loss. Secondly, depending on in what form the overlap control is applied, we need to verify that the consolidated rule (R_q) is not overlapping with any other non-singleton rules (type 2 overlap control), just the non-singleton rules that represent classes other than c_r (type 1 overlap control) or skip this step of verification (type 0 overlap control).

There are a number of different scenarios on what to do next.

- if the transfer is accepted and $k < K_r$, increment k (select the next sample from S_r). If k, however, already equals K_r, delete rule R_r along with associated MFs, update the ranking, increment p and go back to step 1.
- if the transfer is rejected, first discard the changes to the MFs of R_r and R_q, pick the next matching rule from the ranking and go back to step 3. If we already have reached the last matching rule in the ranking, select the next sample from subset S_r (increment k) and go to step 2. If k already equals K_r as well, increment p and return to step 1.

The process comes to a natural end when we have reached the last rule in the ranking ($p = R$) and can then be repeated by returning to the first rule in the ranking until the consolidation stabilizes (i.e. there are no more accepted transfers).

Table 1 shows the number of rules after the rule splits (R_0) and after consolidation (R_c^0) of zero-error classifiers of type 0 overlap identified from benchmark data sets, the value of parameter α and number of samples in these data sets (K).

data set	N	T	K	α	R_0	R_c^0
Iris [6]	4	3	150	0.05	10	7
Wine [1]	13	3	178	0.005	6	4
WDBC [12]	30	2	569	0.005	23	9
WBC [13]	9	2	683	0.05	39	13

Table 1: The data sets (the number of features (N), classes (T) and samples (K)) and the number of rules in corresponding classifiers with overlapping rules before and after consolidation (R_0 and R_c^0, respectively).

3. Elimination of rule overlap

In general, after error-reducing rule splits we obtain a classifier with overlapping rules. This is true for all considered classifiers in Table 1 even before the application of the overlap-ignorant consolidation. To remove the overlap, the procedure introduced in current section is applied.

Note that two rules R_p and R_q are not overlapping if there exists at least one feature $i \in \{1, ..., N\}$ for which the corresponding MFs μ_{ip} and μ_{iq} do not intersect, i.e. either $c_{ip} < a_{iq}$ or $a_{ip} > c_{iq}$ is true. To obtain a classifier with non-overlapping rules we need to identify the existing overlap situations and eliminate the overlap.

If two rules R_p and R_q overlap in N-dimensional space, there exist the samples for which both $\tau_p > 0$ and $\tau_q > 0$ are true. In order to get rid of the overlap we need to shrink one or both of the involved rules and release the samples located in the original overlap area. This can be done iteratively. The question is, which rule to pick for correction, in which dimension to shrink it and in what order to pick the samples to be released.

For this purpose, we identify the subsets of samples $Z_{ip} \subset Sp$ and $Z_{iq} \subset S_q$ that are located between the overlap margins x_i^l and x_i^r for each i

$$\begin{aligned} Z_{ip} &= \{x_{ik} | k \in S_p, x_{ik} \geq x_i^l, x_{ik} \leq x_i^r\} \\ Z_{iq} &= \{x_{ik} | k \in S_q, x_{ik} \geq x_i^l, x_{ik} \leq x_i^r\} \end{aligned} \quad (9)$$

Note that if μ_{ip} is at the left from μ_{iq} then the margins are $x_i^l = a_{iq}$ and $x_i^r = c_{ip}$, while in the opposite case, margins would be $x_i^l = a_{ip}$ and $x_i^r = c_{iq}$ (for the special occasion where one of MFs is inside another, the margins are the edge parameters of the "embedded" MF).

We pick the rule and the dimension/feature according to the cardinality of subset Z_{ir}

$$\underset{1 \leq i \leq N, r \in [p,q]}{\arg\min} |Z_{ir}|. \quad (10)$$

Having identified the dimension i and the rule (p or q), the next step is to exclude the sample from the subset S_r corresponding to the picked rule which has the minimum value of μ_{ir} and appoint it to a newly created singleton rule (Figure 1). This is carried out until the current overlap situation has been eliminated, after what the next similar situation can be handled.

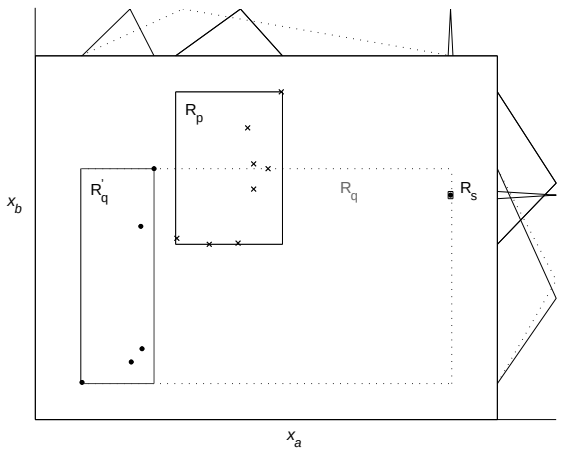

Figure 1: Initial rules R_p and R_q (depicted by dotted line) are substantially overlapping. After appointing the sample at the right to a singleton rule R_s, R_q is shrunk into R'_q which solves the original overlap situation.

data set	overlap control type			
	1		2	
	R_1	R_c^1	R_2	R_c^2
Iris	9	7	7	7
Wine	17	10	10	10
WDBC	49	20	43	25
WBC	33	20	27	22

Table 2: The number of rules in classifiers with different kind of overlap removal before and after consolidation (R_1, R_c^1 and R_2, R_c^2, respectively). Note that for Iris and Wine data sets, the overlap-free classifiers were obtained after resolving the type 1 overlap situations.

Overlap elimination usually creates a number of singleton rules, most of which, however, can be consolidated with existing stronger rules with the consolidation algorithm. Table 2 shows the overlap elimination and subsequent consolidation results in terms of number of rules for benchmark data sets. Note that overlap control of type 1 means that only the overlap situations involving different class rules were resolved, whereas overlap control of type 2 means that all rules in these classifiers are ultimately separated from each other.

4. Rule compression

Rule compression can be considered a special method of feature selection that is implemented as an iterative procedure based on trial and error in which we remove the features/conditions from the rules that do not affect accuracy of the classifier and overlap of the rules (actual features that can be removed vary from rule to rule). This can be ac-

complished in several ways and we have developed three such algorithms (of which first two ignore the aspect of overlap) presented in the following subsections.

4.1. Naive compression algorithm

This algorithm is based on simple trial and error and is described as follows:

1. pick the rule R_r $(r = 1, ..., R)$
2. rank the antecedents $i = 1, ..., N$ by MF spread $(c_{ir} - a_{ir})$, in descending order (by this the features in which the subset of samples governed by r-th rule is less compact, are removed first)
3. discard the conditions applied to the antecedents one by one, in the order of ranking, cancelling those removals on the run that would result in loss of accuracy.

4.2. Template based compression

The template based compression, although also based on trial and error, is more sophisticated and can discard several features simultaneously whereas compressions at higher rates are prioritized higher.

1. pick the number of features in the template M $(M = 1, ..., N - 1)$
2. pick rule R_r $(r = 1, ..., R)$
   - make a list all possible unique combinations of M variables (the total number of these amounts to $N_c = N!/(M!(N-M)!)$) that serve as compression templates, so to speak

Next we cycle through these N_c items in the list starting from the first one. At each iteration:

- compress R_r according to the picked compression template
- compute the classification error of the resulting classifier
  - if there is no accuracy loss - increment r and go back to step 2 which means that the compression is accepted.
  - if accuracy loss is detected and we have not yet reached the end of the template list - restore original R_r and pick the next compression template from the list.
  - if accuracy loss is detected and and we have reached the end of the list - restore original R_r, increment r and return to step 2.

Following this, increment M (meaning that all subsequent compressions will be done at lower compression rate) and start from step 1 again (for already compressed rules further compression, of course, is no longer applied). The process ends when we have managed to compress all rules.

	R_1	R_2	R_3	R_6
R_1	$\emptyset$	{3,4}	{1,3,4}	{3,4}
R_2	{3,4}	$\emptyset$	{3}	{4}
R_3	{1,3,4}	{3}	$\emptyset$	{4}
R_6	{3,4}	{4}	{4}	$\emptyset$

Table 3: The rule compression look-up table for Iris classifier

The main shortcoming of the method is its computational cost because the list of possible combinations grows exponentially as the number of features increases (e.g. for a 20-feature classification problem there would be 184756 different 10-feature templates alone) and is a practical approach only if the compressions are made in the very beginning of the process (at high compression rates).

4.3. Analytical compression algorithm

This algorithm is applicable only to the classifiers to which overlap control of type 1 or type 2 has been applied.

In first stage of the algorithm non-singleton rules are compressed. For this purpose a look-up table is constructed in which each entry in the given row contains the set of features in which the MFs of the given rule do not intersect with MFs of the rule determined by the column. The entries in the main diagonal are always empty sets and are ignored.

The compression is based on the analysis of the table. When attempting to compress a rule corresponding to a given row, a feature that is most frequently represented in the entries of the row is picked first and all the entries it was found at are excluded from the further analysis of the rule. Next we pick the feature that is most frequently represented in the remaining entries and exclude corresponding entries. In the end of processing this row there are no more entries left and the rule can be compressed into the picked features.

Consider, for example, the look-up table constructed for the four non-singleton rules of the Iris data classifier (Table 3). It appears from the table that R_1 can be compressed into feature 3 (or 4) because those are represented in all entries of first row; R_2 can be compressed into features 3 and 4 because either 3 or 4 is always present in row 2 entries. R_3 will be compressed into features 3 and 4 and R_6 into feature 4 for now obvious reasons (Figure 2). Features 3 and 4 (petal width and length) are indeed relevant because Iris classes are poorly separated in features 1 and 2.

In the case where overlap of same class rules is permitted, the entries in the table corresponding to same-class overlap are neglected from the analysis just like the entries in the main diagonal. In current example, R_3 and R_6 both represent class 3 and if their separation is not necessary, the last entry in the 3rd row of Table 3 is not taken into account,

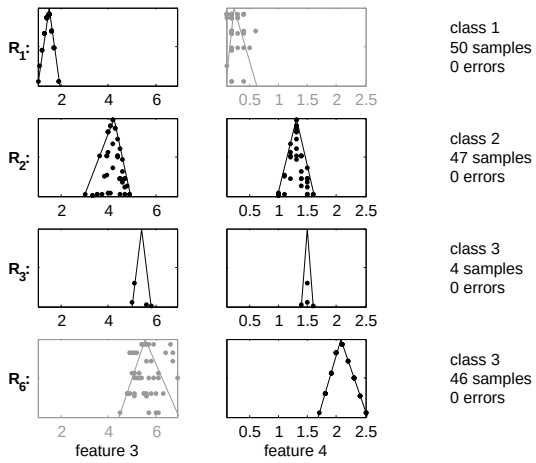

Figure 2: Compressed non-singleton rules of Iris data classifier. The compressed features in rules 1 and 6 are drawn in grey color.

which permits us to compress R_3 into a single feature (feature 3). The third entry in the last row is simlarly ignored.

The compression of singleton rules is done separately and is (similarly to the naive compression method) based on pure trial and error

1. pick a singleton rule R_p
2. determine the subset of samples S_p governed by the p-th rule (S_p, of course, consists of a single sample)
3. remove the antecedents one by one, cancelling those removals on the run that would increase the number of samples in the subset S_p.

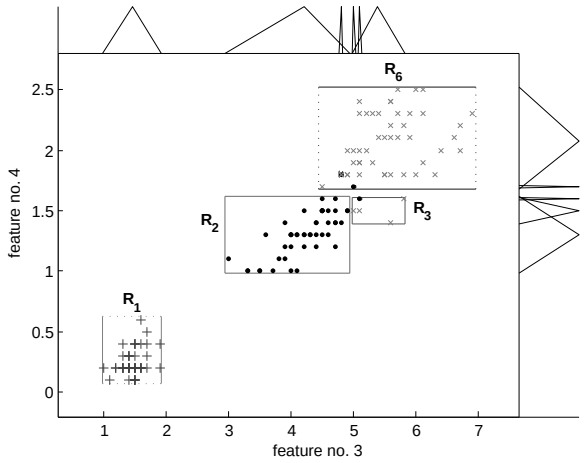

Figure 3: The scatterplot of Iris classifier rules. The three Iris classes are: + - Iris setosa, • - Iris versicolor, x - Iris virginica

In Figure 3 that depicts all rules of the classifier in the most relevant features we can spot three major rules describing majority of samples belonging to each of three classes. In addition, there is one minor rule (R_3) that describes four class 3 samples that

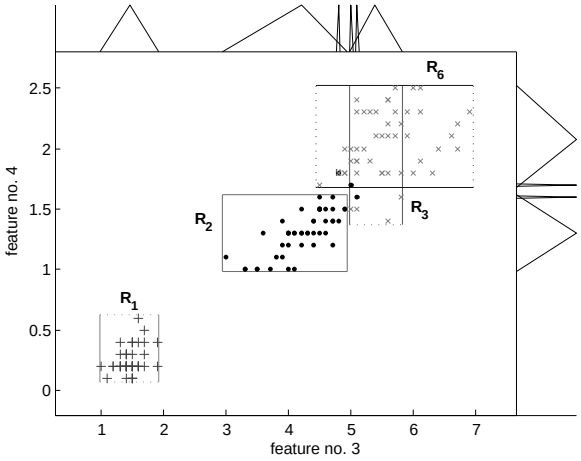

Figure 4: Compression with type 1 overlap control has introduced the overlap between the same class rules (R_3 and R_6).

are not compatible with the major rule of the same class (R_6) and three singleton rules R_5, R_7 (located inside R_6) and R_4 (located inside R_3) representing evident outliers.

In Figure 4 that depicts the classifier that has been compressed with type 1 overlap control, the only difference in comparison with the classifier in Figure 3 is the removal of the MF in feature 4 from R_3, which leads to its overlap with R_6 and sample redistribution among these rules (R_3 acquires 12 additional samples from R_6). The high overlap of R_3 and R_6 makes them virtually indistinguishable from each other in rule view, which is unfortunate from the interpretational aspect.

5. Results and discussion

The compression results are given in Table 4, where classification by a tree (CART) is added for comparison (the number of rules in a classifier converted from a tree is equal to the number of leaf nodes of the latter, whereas the number of conditions in a rule is usually smaller than the number of internal nodes met on the path from the root node to the leaf node because in a rule, the conditions concerning the same variable can be combined into the very same condition). The following observations can be made:

- Template based compression that obtains 12, 13 and 54 conditions for Iris, Wine and WBC data set classifiers, respectively, has only a slight advantage over the naive algorithm (respective measures are 14, 16 and 56, see Table 4). It is unable to complete the compression of the WDBC data classifier in reasonable time because of the high number of original features.
- The classifiers, in which the overlap control of either type 1 or type 2 is applied, are com-

pressed at a higher rate than classifiers in which the overlap control is not enforced. This is clearly evidenced in Table 4 when dividing the number of conditions with the number of rules. Moreover, the conditions in those classifiers are much more evenly distributed over the rules (some, usually the major rules, are compressed at a very low compression rate in classifiers in which the overlap control is not enforced).

- The classifiers with type 1 or type 2 overlap control have usually considerably more rules than those classifiers in which overlap control has not been enforced. The classifiers with type 1 overlap control, however, do not have considerably more rules than the classifiers with type 2 overlap control.
- Compression of overlapping rules generally redistributes the samples among the rules.
- The classifiers with type 1 or type 2 overlap control usually contain a healthy dose of singleton rules. These singleton rules may represent noise or measuring errors and can be conveniently isolated from the rest of data this way.
- The classification tree results should be directly compared the classifiers with type 2 overlap control. It can be seen that the proposed algorithms yield much more compact classifiers.

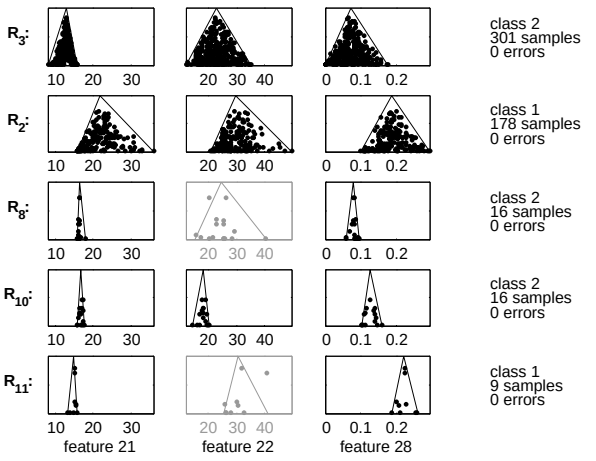

Figure 6: Five strongest rules of the WDBC data classifier that cover over 91% of samples.

Let us have a closer look at Wine and WDBC type 2 overlap classifiers. From Figure 5 it is clear that class 1 and class 3 wines (in R_1 and R_4, respectively) can be distinguished from each other by the value of flavanoids (feature 7). Class 2 wines (R_2), however, have mixed values of flavanoids and therefore, to separate class 2 wines from class 3 wines we need to look at their color intensity (feature 10) and from class 1 wines at proline values (feature 13). Three minor rules describe the small groups of wines that do not fit this pattern. For example, six samples in R_3 have too high values of proline to be distinguished from R_1 and use the value of color intensity instead, whereas in R_7 the value of

data set	overlap control			CART
	0	1	2	
Iris	7/14	7/11	7/12	9/21
Wine	4/16	10/16	10/20	12/43
WDBC	9/93	19/41	22/52	22/103
WBC	13/56	20/44	22/55	32/139

Table 4: Summary of compression results. Each table entry consists of two numbers separated by a slash, which stand for the number of rules and number of conditions in compressed classifiers, respectively.

the latter is too high and feature 3 (ash) is used. In R_8, the color intensity is too low to provide separation from class 2 wines and the value of flavanoids is used instead. In addition, there are four single specimens that do not fit the above reasoning and are defined using singleton rules.

The WDBC data classifier has even more singleton rules (10 of 22). Due to space limitations we only look at a selection of stronger classifier rules (Figure 6). First of all, the compression algorithm has decided that the most relevant features are the extreme values of cell radius (feature 21), texture (feature 22) and concave points (feature 28). This harmonizes well with the claim "the extreme values are the most intuitively useful for the problem at hand since only a few malignant cells may occur in a given sample" [12].

As a rule, the healthy patients are separated from sick ones by lower values in feature 21. However, there exist healthy persons that have high values of that feature (R_8 and R_{10}) that are distinguished from sick persons in features 22 and 28, respectively. And there are sick persons that have deceptively low high value of feature 22 but distinguish from healthy persons by feature 28 (R_{11}).

As those two examples demonstrate, interpretation of classification rules in which there is no overlap, is very intuitive.

6. Conclusions

The interest in interpretable fuzzy classifiers can be largely credited to our increasing need to understand and reason about data. Present study of rule compression methods clearly demonstrates that rule compression can improve interpretability of fuzzy classifiers to a great extent by bringing our attention to the features that are crucial in assigning the samples of the data set to the proper classes. It is particularly helpful when the number of available features is high. The study also suggests that the most easily interpretable fuzzy classifiers are such in which the overlap between the rules is eliminated. This, however, is a somewhat inconvenient conclusion because essentially, such classifiers are no longer fuzzy classifiers.

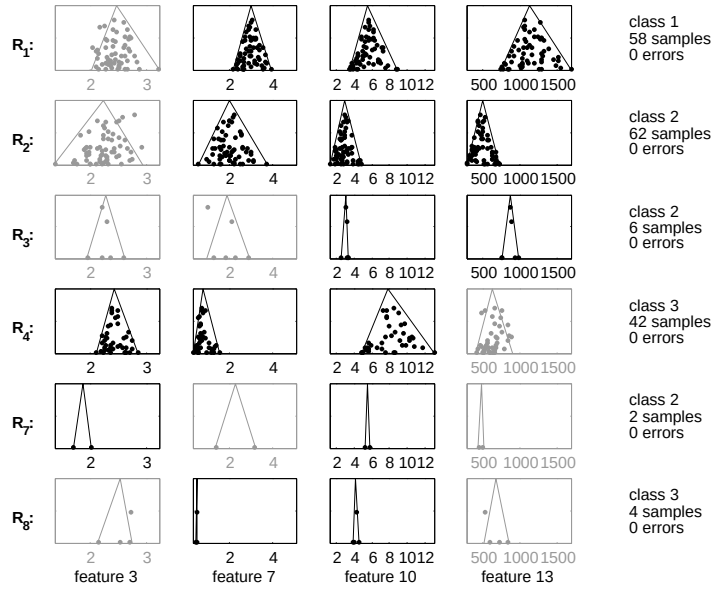

Figure 5: Compressed non-singleton rules of wine data classifier.

References

[1] S. Aeberhard, D. Coomans, and O. de Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 27(8):1065–1077, 1994.

[2] J. M. Alonso, C. Castiello, and C. Mencar. Interpretability of fuzzy systems: Current research trends and prospects. In J. Kacprzyk and W. Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 181–199. Springer-Verlag, Berlin Heidelberg, 2015.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, 1984.

[4] E. H. Cardenas and H. A. Camargo. Multiobjective genetic generation of fuzzy classifiers using the iterative rule learning. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 1–8, Brisbane, Australia, 2012.

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

[6] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

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

[8] A. Riid and E. Rüstern. An integrated approach for the identification of compact, interpretable and accurate fuzzy rule-based classifiers from data. In *Proceedings of the 15th International Conference on Intelligent Engineering Systems*, pages 101–107, Poprad, Slovakia, 2011.

[9] A. Riid and E. Rüstern. Adaptability, interpretability and rule weights in fuzzy rule-based systems. *Information Sciences*, 257:301–312, 2014.

[10] A. Riid and M. Sarv. Determination of regional variants in the versification of Estonian folksongs using an interpretable fuzzy rule-based classifier. In *Proceedings of the 8th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2013)*, pages 61–66, Milan, Italy, 2013.

[11] P. K. Simpson. Fuzzy min-max neural networks–part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, 1992.

[12] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Proceedings of the IS&T 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870, San Jose, CA, 1993.

[13] W. H Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87:9193–9196, 1990.

[14] S. Zhou and J. Gan. Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling. *Fuzzy Sets and Systems*, 159(23):3091–3131, 2008.