Induction of Fuzzy-Rule-Based Classifiers With Evolutionary Boosting Algorithms

María José del Jesus, Frank Hoffmann, Luis Junco Navascués, and Luciano Sánchez

***Abstract*—This paper proposes a novel Adaboost algorithm to learn fuzzy-rule-based classifiers. Connections between iterative learning and boosting are analyzed in terms of their respective structures and the manner these algorithms address the cooperation-competition problem. The results are used to explain some properties of the former method. The evolutionary boosting scheme is applied to approximate and descriptive fuzzy-rule bases. The advantages of boosting fuzzy rules are assessed by performance comparisons between the proposed method and other classification schemes applied on a set of benchmark classification tasks.**

***Index Terms*—Boosting algorithms, evolutionary algorithms, fuzzy-rule-based classifiers, iterative learning.**

I. Introduction

BOOSTING algorithms are statistical additive modeling techniques that combine different low-quality classifiers to obtain a compound classifier that performs better than any of its components. Adaboost [8] is a boosting algorithm, which repeatedly invokes a learning algorithm to successively generate a committee of simple, low-quality classifiers. Each time a new simple classifier is added to the compound one, the examples in the training set are re-weighted (so that future classifiers will focus on the most difficult examples,) and a voting strength is assigned to the classifier. The number of votes a classifier is given depends on the confidence in its classification accuracy, as measured on the training set. Adaboost generates a compound classifier which decision is a linear threshold of the outputs of the simple classifiers.

Adaboost is commonly used to combine linear classifiers, but other learning algorithms can be boosted well. This paper proposes the application of the Adaboost algorithm to learn a fuzzy-rule-based classifier, and experimentally shows the advantages of the approach over other fuzzy and nonfuzzy classification methods. The methodology proposed in the following unifies and extends previous approaches to boost fuzzy rules [13], [17].

In particular, the Adaboost algorithm is compared with the incremental learning of fuzzy-rule-based classifiers [1]–[3], [5], [9], [11]. Similar to Adaboost, iterative learning consists in the recurring invocation of an algorithm that identifies the fuzzy rule that best matches the current set training examples. The main difference between iterative learning and Adaboost is, that those training examples that are correctly classified by a new rule are removed from the training set instead of being down-weighted. Therefore, future rules are not aware of previously removed instances, such that conflicting rules might emerge in the rule base. These conflicts are resolved by means of a genetic post processing stage that weeds out contradicting rules from the initial base. As a second difference, iterative learning does not assign individual votes to the rules (all rules are equally important) but instead tunes the membership functions to improve classification accuracy. Adaboost does not require these two post processing steps of the compound classifier with respect to conflict resolution and tuning.

This paper is organized as follows. Section II briefly introduces the Adaboost algorithm and describes the type of fuzzy classifiers that are boosted. Section III proposes methods for boosting descriptive and approximate fuzzy rule bases, and explains the technical details of the approach, and how it is applied to two-class as well as multiclass problems. Finally, in Section IV the classification accuracy of the proposed Adaboost algorithms on a number of data sets is compared with the genetic iterative learning and other fuzzy and statistical classifiers.

II. Adaboost and Fuzzy-Rule-Based Classifiers

A. Fuzzy Classifiers

At this point we introduce the basic notation employed throughout this paper. Let $\mathbf{X}$ be the feature space, and let $\mathbf{x}$ be a feature vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{X}$. Let p be the number of classes. The training set is a sample of m classified examples $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbf{X}$, $1 \leq y_i \leq p$, $1 \leq i \leq m$.

The antecedents of all fuzzy rules in the classifier form a fuzzy partition $\mathcal{A}$ of the feature space $\mathcal{A} = \{A^j\}_{j=1\ldots N}$, with $A^j \subset \widetilde{\mathcal{P}}(\mathbf{X})$, where $\widetilde{\mathcal{P}}(\mathbf{X})$ stands for "fuzzy parts of $\mathbf{X}$". In the remainder of this paper, we assume that the training examples are indexed by the letter i, the rules by j, the features by f and the classes by k; the ranges of these variables are $1 \leq i \leq m$, $1 \leq j \leq N$, $1 \leq f \leq n$ and $1 \leq k \leq p$. For example, "for all $\mathbf{x}_i$" means $\mathbf{x}_i$, $1 \leq i \leq m$; from now on, unless necessary, the ranges of indexes are not explicitly stated.

A fuzzy-rule-based classifier is defined by means of a fuzzy relationship defined on $\mathcal{A} \times \{1, \ldots, p\}$. Values of this relationship describe the degrees of compatibility among the fuzzy subsets of the feature space collected in $\mathcal{A}$, and each of the classes. In other words, for every antecedent A^j there are p numbers between 0 and 1 that represent the confidence in the assertion "All

Manuscript received November 20, 2001; revised November 22, 2002 and September 8, 2003. This work was supported by the Spanish Ministry of Science and Technology and by European Fund FEDER under Projects TIC-04036-C05-05 and TIC-04036-C05-04.

M. J. del Jesus is with the Computer Science Department, Jaén University, 23071 Jaén, Spain.

F. Hoffmann is with the University of Dortmund, 44227 Dortmund, Germany.

L. Junco Navascués and L. Sánchez are with the Computer Science Department, Oviedo University, 33204 Oviedo, Spain (e-mail: luciano@lsi.uniovi.es).

Digital Object Identifier 10.1109/TFUZZ.2004.825972

Input data:

Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m), \quad \mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{-1, +1\}$

Number of hypotheses $H \in \{1, \ldots, N\}$

Local Variables:

$w \in \mathbf{R}^m$ (weights of the examples in the training set)

$\alpha \in \mathbf{R}^N$ (votes of the weak hypotheses)

begin

Initialize $w_i \leftarrow 1/m, \alpha^j = 0$

Select the number of weak hypotheses H

Repeat H times

Identify the weak hypothesis $g^h \in \{g^1, \ldots g^N\}$ that best classifies the weighted data

Calculate the number of votes α^h of g^h

Update the weights w_i of the examples

end-repeat

Output the classifier: sign $\left(\sum_{j=1}^{N} \alpha^j g^j(\mathbf{x})\right)$

end

Fig. 1. Outline of the generalized Adaboost algorithm. Two-class version.

elements in the fuzzy set A^j belong to class number k". Values close to 1 indicate "high confidence," and values close to 0 denote "absence of knowledge about the assertion."

1) Linguistic Interpretation of Fuzzy Classifiers: Fuzzy-rule-based classifiers are comprehensible to humans as they can be expressed in form of linguistic sentences. There are different standards when translating the former fuzzy relationship into linguistic statements. In this paper, we combine p instances of the fuzzy relationship

$$\text{compatibility}(A^j, c_k) = s_k, \qquad k = 1, \ldots, p$$

into a single sentence, as follows:

$$\text{if } \mathbf{x} \text{ is } A^j \text{ then truth}(c_1) = s_1^j \text{ and } \cdots \text{ and truth}(c_p) = s_p^j.$$

The antecedents A^j are decomposed in a Cartesian product of fuzzy sets defined over each feature, $A^j = A_1^j \times A_2^j \times \ldots \times A_n^j$, thus the rules are

$$\text{if } x_1 \text{ is } A_1^j \text{ and } \ldots \text{ and } x_n \text{ is } A_n^j \\ \text{then truth}(c_1) = s_1^j \text{ and } \cdots \text{ and truth}(c_p) = s_p^j.$$

The linguistic expression of the fuzzy classifier does not include those terms for which confidence values are null. In the case of fuzzy subsets for which all confidence values are null, the rule base comprises fewer sentences (fuzzy rules,) than elements in the fuzzy partition $\mathcal{A}$.

We can further restrict the definition by defining n linguistic variables (one linguistic variable for every feature) and requiring that all terms sets A_f^j in the antecedents are associated with one linguistic term in its corresponding linguistic variable. In this case, we obtain a fuzzy-rule-based *descriptive* classifier. If we do not apply the latter restriction, we obtain an *approximate* classifier.

Notice, that in a descriptive fuzzy classifier the set of possible rules is finite due to the discrete number of possible linguistic labels associated to each rule. Conversely, there is an infinite number of possible approximate classifiers as fuzzy rules use continuous parameters to define the characteristic points of their underlying fuzzy sets.

2) Fuzzy Inference: Fuzzy reasoning methods define how rules are combined and how to infer from a given input to the corresponding output. The actual inference method is solely defined in terms of the fuzzy relationship, and is therefore independent of the approximate or descriptive nature of the classifier. An instance $\mathbf{x}$ is assigned to the class

$$\arg\max_{k=1,\ldots,p} \bigvee_{j=1}^{N} A^j(\mathbf{x}) \wedge s_k^j \tag{1}$$

where "$\wedge$" and "$\vee$" can be implemented by different operators; for example, "$\vee$" can be the maximum operator [18] or the arithmetic sum, so called "maximum voting scheme" [16]. "$\wedge$" is always a t-norm, usually the minimum or the product. In this paper, fuzzy inference employs the product operator and the maximum vote scheme.

B. Adaboost Algorithm

Let us define a set $\{g^1, g^2, \ldots, g^N\}$ of simple, but possibly unreliable binary classifiers. Boosting consists in combining these low quality classifiers (so called "weak hypotheses" in the boosting literature) with a voting scheme to generate an overall classifier that performs better than any of its individual constituents alone. We will show later that a fuzzy rule can be regarded as a particular case of a weak hypothesis, and a fuzzy rule base can be interpreted as a weighted combination of weak hypotheses.

Weak hypotheses take feature values as input and produce both a class number as well as a degree of confidence in the given classification. In two-class problems, these two outputs are encoded by a single real number, $g^j(\mathbf{x}) \in \mathbf{R}$, whose sign is interpreted as the label of $\mathbf{x}$ and whose absolute value is interpreted as the confidence in the classification. The higher this

value the more confidence is given to the classification. Adaboost is intended to produce a linear threshold of all hypotheses

$$\operatorname{sign}\left(\sum_{j=1}^{N} \alpha^j g^j(\mathbf{x})\right). \tag{2}$$

An outline of the Adaboost algorithm is shown in Fig. 1. Observe that Adaboost can operate with any learning algorithm that generates a confidence rated classifier on a given weighted data set. There are different algorithms for assigning the number of votes to a weak hypothesis, and for adjusting the weights of the examples. For example, in confidence-rated Adaboost [26] the number of votes of the weak hypothesis g^h is given by the value α^h that minimizes the following function:

$$Z(\alpha) = \sum_{i=1}^{m} w_i \exp\left(-\alpha y_i g^h(\mathbf{x}_i)\right) \tag{3}$$

and the weights of the examples are updated according to

$$w_i \leftarrow \frac{w_i \exp\left(-\alpha^h y_i g^h(\mathbf{x}_i)\right)}{v} \tag{4}$$

where v is the normalization factor such that $\sum w_i = 1$. There are analytical approximations and heuristics that may replace this formula in specific problems.

C. Adaboost Compared to Genetic Iterative Learning

Fuzzy classifier induction with genetic iterative learning [5] operates in three stages. The first stage consists of a "fuzzy rule generation" algorithm, which evolves a population of candidate fuzzy rules that correctly classify the examples in the training set. The rule with the largest covering degree, highest frequency value and highest consistency is added to the intermediate rule base [3]. The examples that are correctly classified by the selected rule are removed from the training set. The basic rule generation mechanism is repeated until the training set becomes empty as all examples in the original training set are covered by at least one rule.

Rules generated in later stages are unaware of the previously removed examples and therefore might be in conflict with rules generated earlier on. This undesirable effect is an instance of the well-known cooperation-competition problem in genetic fuzzy systems. In order to improve the cooperation among the fuzzy rules the intermediate rule set is post-processed, with the objective to resolve conflicts among fuzzy rules with the same antecedent but contradicting classifications. In the second, so called "genetic multiselection", stage a binary coded genetic algorithm removes conflicting rules. The third and final stage, "genetic tuning," adjusts the membership functions of the linguistic terms in order to improve the classification rate of the rule base, but does not add or delete rules.

Adaboost is applicable to boost fuzzy rules as an individual rule can be interpreted as a confidence-rated weak, albeit incomplete hypothesis. To show that fuzzy rules are in fact equivalent to weak hypotheses, recall the definition of a fuzzy classifier given in (1). If the maximum vote scheme is combined with the product t-norm, the output of a fuzzy classifier becomes

$$\arg\max_{k=1,\ldots,p} \sum_{j=1}^{N} A^j(\mathbf{x}) \cdot s_k^j. \tag{5}$$

Let us particularize this expression for two-class problems

$$\arg\max\left\{\sum_{j=1}^{N} A^j(\mathbf{x}) \cdot s_1^j, \sum_{j=1}^{N} A^j(\mathbf{x}) \cdot s_2^j\right\} \tag{6}$$

therefore, if class numbers are 1 and -1, the output of the classifier becomes

$$\operatorname{sign}\left(\sum_{j=1}^{N} A^j(\mathbf{x}) \cdot \left(s_1^j - s_2^j\right)\right). \tag{7}$$

If this equation is compared to the Adaboost decision in (2), the analogy between A^j and g^j on the one hand, and $s_1^j - s_2^j$ and α^j on the other hand becomes obvious.

The first stage of iterative learning shares a common structure with Adaboost. Iterative learning identifies the fuzzy rule that best describes a subset of the training examples, and determines its number of votes (later, we will relate the votes to the degree of truth for the rule consequent.) Then, it alters the weight of training examples, depending on how well they are classified by the new rule. This process is repeated until either a desired number of rules has been retrieved or a given classification accuracy on the training set has been achieved. Notice that Adaboost, unlike the iterative genetic rule learning, down-weights the importance of training examples, rather than discarding them altogether once they are covered by a fuzzy rule. In the iterative rule learning scheme, it might happen that a deleted training instance is incorrectly classified by another fuzzy rule that emerges in a later invocation of the rule learner. The "cooperation-competition problem" is addressed in a subsequent stage which globally refines the initial rule base. As Adaboost never completely discards a training instance cooperation is implicitly achieved and the refinement stage becomes obsolete, in other words the rules generated by the incremental search already constitute the final rule base.

From the above we can conclude that boosting and iterative rule learning are similar techniques. The most important differences are: a) in iterative learning, the weight of an instance is either 0 or 1, and b) in Adaboost the multiselection stage becomes obsolete, as instances are never completely removed from the training, but rather down-weighted according to the accuracy that a new rule achieves. This property allows Adaboost to identify potential conflicts among rules already during rule generation.

III. Boosting Fuzzy Rules

A. Outline of the Algorithm

For the sake of simplicity, we restrict the discussion for the time being to two-class problems. Following with the similarities between confidence rated weak learners and fuzzy rules

Input data:

Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m), \quad \mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{-1, +1\}$

Number of fuzzy rules $H \in \{1, \ldots, N\}$

Local Variables:

$w \in \mathbf{R}^m$ (weights of the examples in the training set)

$\alpha \in \mathbf{R}^N$ (votes of the weak hypotheses)

$c \in \{-1, 1\}^m$ (consequents of the weak hypotheses)

$s : \mathcal{A} \times \{-1, 1\} \rightarrow [0, 1]$ (fuzzy relationship = consequents of rules)

Local Procedures:

Generate a new fuzzy rule: **begin**

Generate by means of an evolutionary algorithm the rule " if $\mathbf{x}$ is A^h then c^h"

with $A^h \in \mathcal{A}$, that minimizes the fitness

end

Convert votes into confidences: **begin**

for all j do

if ($\alpha^j > 0$ and $c^j = 1$) or ($\alpha^j < 0$ and $c^j = -1$) $s_1^j = |\alpha^j| / \max |\alpha^j|$ else $s_{-1}^j = |\alpha^j| / \max |\alpha^j|$

end

begin of main algorithm

Initialize $w_i \leftarrow 1/m$, $\alpha^j = 0$, $s_k^j = 0$

repeat H times

Generate a new fuzzy rule " if $\mathbf{x}$ is A^h then c^h"

Calculate the number of votes of the rule: votes(if $\mathbf{x}$ is A^h then c^h) $= \alpha^h$

Update the weights of the examples

end-repeat

Convert votes into confidences

Output the classifier: for all j, if $s_1^j \neq 0$ or $s_{-1}^j \neq 0$ emit the rule

if x_1 is A_1^j and $\ldots x_n$ is A_n^j then tr(class $= 1$) $= s_1^j$ and tr(class $= -1$) $= s_{-1}^j$

end of main algorithm

Fig. 2. Adaboost algorithm applied to the induction of a descriptive, two-class, fuzzy-rule-based classification system.

introduced in (7), the space of weak hypotheses is identified with the product of the fuzzy partition and the class labels, $\mathcal{A} \times \{1, \ldots, p\}$.

Fig. 2 shows an outline of the algorithm. In the following, the individual steps of the method are explained in detail: namely how to obtain a new fuzzy rule, how to compute its number of votes and how to adjust the weights of the training examples once the new rule has been added to the base. Notice, that the values of α^j (the number of votes) that Adaboost generates are not restricted to the interval [0,1]; some of the schemes to assign the number of votes may produce negative results, and all of them can produce a vote weight larger than 1. Therefore, vote weights generated by Adaboost can not be coherently interpreted as confidence rates. The degrees α^j in the consequents are normalized to a range [0,1] before the entire rule base is generated.

In descriptive rule bases $\mathcal{A}$ is finite, thus the best weak hypothesis can be found by exhaustive search, as long as the number of features N is small. As the number of features increases, one has to use more efficient albeit potentially suboptimal search and optimization techniques. In approximate representations, membership functions are defined by continuous parameters. Therefore, the solution space is infinite and the identification of the best rule becomes an optimization problem. We propose evolutionary algorithms for identifying fuzzy rules that are well adapted to the training set in the rule generation step.

B. Objective (Fitness) Function

There are several criteria that determine how well a fuzzy rule captures the weighted training examples. Given the results in [26], the fitness of a fuzzy rule is

$$\text{fitness}(\text{if } \mathbf{x} \text{ is } A^j \text{ then } c^j) = \sum_i w_i \exp\left(-y_i c^j\, A^h(\mathbf{x}_i)\right). \quad (8)$$

This expression can be interpreted as a measure of the weighted difference between the sum of the memberships of "class 1" and "class -1" examples. This objective function is, in essence, similar to other heuristics previously suggested for fuzzy rule learning. In particular, it is related to criteria in which new rules are selected depending on the maximum difference between the sum of the memberships of "positive" and "negative" examples [10]. We evaluated criteria like the above in combination with boosting and observed that in terms of final classification accuracy there are no significant between (8) and the conventional heuristics. In particular, in [4], the authors propose to evaluate different aspects of the quality of a fuzzy classification rule, such as a high frequency value, a high covering degree over positive examples and the k-consistency property to penalize negative

examples covered by a rule. The fitness function employed in that approach considers two objectives, namely the number of training instances covered by the rule j compared to the overall number of training instances with the rules' class label c^j

$$f_1 = \frac{\sum_{i:y_i=c^j} w_i A^j(\mathbf{x}_i)}{\sum_{i:y_i=c^j} w_i} \tag{9}$$

and the frequency of negative examples covered by a rule

$$f_2 = \frac{\sum_{i:y_i\neq c^j} w_i A^j(\mathbf{x}_i)}{\sum_i w_i A^j(\mathbf{x}_i)}. \tag{10}$$

Both objectives are aggregated into a scalar fitness value

$$f = \begin{cases} 0, & f_2 > k_{\max} \\ f_1 * \left(1 - \frac{f_2}{k_{\max}}\right), & f_2 \leq k_{\max} \end{cases} \tag{11}$$

where the parameter $k_{\max}$ denotes the maximal rule inconsistency that is still tolerated.

C. Number of Votes of a Rule

Adaboost can determine the number of votes assigned to a weak hypothesis by different means. Confidence-rated Adaboost employs an optimization-based approach which determines the confidence degree by minimizing the following expression:

$$Z(\alpha) = \sum_i w_i \exp\left(-\alpha y_i A^j(\mathbf{x}_i)\right). \tag{12}$$

For this problem, classical optimization techniques, such as Brent's linear search method [19], turn out to be more efficient than evolutionary search algorithms. $Z(\alpha)$ is convex except for the case in which the rule antecedent does not cover any negative examples; in which case the above expression is minimized for $\alpha \to \infty$. The problem that α is unbounded is common to all implementations of Adaboost, see for example [26]. Brent's method converges in a few iterations in the convex case. However, in practice low error rates also lead to numerical instabilities. To avoid these numerical instabilities during optimization, a term that penalizes large values of α is added

$$Z(\alpha) = \sum_i w_i \exp\left(-\alpha y_i A^j(\mathbf{x}_i)\right) + \sum_{i:A^j(\mathbf{x}_i)=0} w_i \exp\left(|\alpha\epsilon|\right) \tag{13}$$

where ϵ is a small manually determined parameter.

Other implementations of Adaboost do not solve the direct optimization problem but instead optimize an analytic approximation as follows: The boosting algorithm computes the error E_j of the fuzzy rule number generated in iteration j. In our case, each fuzzy classification rule constitutes an incomplete, weak classifier. Incomplete in the sense, that it is able to classify those instances covered by its antecedent but makes no prediction about the other training examples. Therefore, the classification error E_j of a fuzzy rule is weighted by the degree of matching $A^j(\mathbf{x}_i)$ between the ith training instance $(\mathbf{x}_i, y_i)$ and the rule antecedent as well as the weight w_i

$$E_j = \frac{\sum_{i:y_i\neq c^j} w_i A^j(\mathbf{x}_i)}{\sum_i w_i A^j(\mathbf{x}_i)}. \tag{14}$$

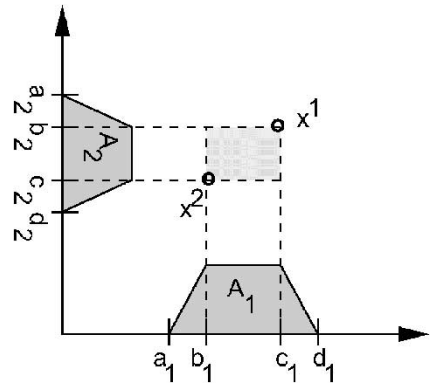


Fig. 3. Initialization of two trapezoidal fuzzy sets A_1, A_2 using a pair of training instances x_1, x_2.

In other words, the goal of the fuzzy rule generation method is to find classification rules that perform well over the current distribution of training examples. By means of the previously defined error measure, one can apply the approximation given in [25] to calculate the number of votes assigned to the jth rule

$$\alpha^j = \log\left(\frac{1 - E_j}{E_j}\right). \tag{15}$$

Rules with small classification error E_j obtain a larger weight. To avoid numerical instabilities, the same correction term that was applied in (13) is included in the calculation of the error

$$E_j = \frac{\sum_{i:y_i\neq c^j} w_i \max\left(\epsilon, A^j(\mathbf{x}_i)\right)}{\sum_i w_i \max\left(\epsilon, A^j(\mathbf{x}_i)\right)} \tag{16}$$

where the parameter ϵ is small and determined in advance.

D. Updating the Weights of the Examples

The original implementation of Adaboost only updates the weights of correctly classified examples: The weight of an instance $(\mathbf{x}_i, y_i)$ is multiplied by some factor β_i if the jth rule classifies the example correctly and is left unchanged otherwise

$$w_i \leftarrow \begin{cases} w_i, & \text{if } y_i \neq c^j \\ \beta_i w_i, & \text{if } y_i = c^j \end{cases}. \tag{17}$$

The factor

$$\beta_i = \left(\frac{E_j}{1 - E_j}\right)^{A^j(\mathbf{x}_i)} \tag{18}$$

depends on the error E_j of the fuzzy rule and the degree of matching $A^j(\mathbf{x}_i)$ between the fuzzy rule and the training example. Effectively, examples that are classified correctly and which match the rule antecedent obtain a lower weight, and misclassified or uncovered examples get relatively higher weights. Thereby, the boosting algorithm increases the weight of those examples which are most difficult to classify correctly for the genetic fuzzy system.

The most up to date implementations of Adaboost (confidence-rated Adaboost) simultaneously update the importance of all examples not just the correctly classified ones: the weight of

Input data:

Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m), \quad \mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{1, \ldots, p\}$

Number of fuzzy rules $H \in \{1, \ldots, N\}$

Local Variables:

$w \in \mathbf{R}^m$ (weights of the examples in the training set)

$\alpha \in \mathbf{R}^N$ (votes of the weak hypotheses)

$c \in \{1, \ldots, p\}^m$ (consequents of the weak hypotheses)

$s : \mathcal{A} \times \{-1, 1\} \rightarrow [0, 1]$ (fuzzy relationship = consequents of rules)

Local Procedures:

Generate a new fuzzy rule: **begin**

Generate by means of an evolutionary algorithm the rule " if $\mathbf{x}$ is A^h then c^h"

with $A^h \in \mathcal{A}$, that minimizes the fitness

end

Convert votes into confidences: **begin**

$v = \max |\alpha^j|$

for all j do

if $(\alpha^j > 0)$ $s_1^j = 0, \ldots, s_{c^h-1}^j = 0, s_{c^h}^j = |\alpha^j|/v, s_{c^h+1}^j = 0, \ldots, s_p^j = 0$

else $s_1^j = |\alpha^j|/v, \ldots, s_{c^h-1}^j = |\alpha^j|/v, s_{c^h}^j = 0, s_{c^h+1}^j = |\alpha^j|/v, \ldots, s_p^j = |\alpha^j|/v$

end

begin of main algorithm

Initialize $w_i \leftarrow 1/m$, $\alpha^j = 0$, $s_k^j = 0$

repeat H times

Generate a new fuzzy rule " if $\mathbf{x}$ is A^h then c^h"

Calculate the number of votes of the rule: votes(if $\mathbf{x}$ is A^h then c^h) $= \alpha^h$

Update the weights of the examples

end-repeat

Convert votes into confidences

Output the classifier: for all j, if any $s_f^j \neq 0$ emit the rule

if x_1 is A_1^j and $\ldots x_n$ is A_n^j then tr(class $= 1) = s_1^j$ and $\ldots$ and tr(class $= p) = s_p^j$

end of main algorithm

Fig. 4. Adaboost algorithm applied to the induction of a descriptive, fuzzy-rule-based classification system. Straightforward extension of the two-class version to the multiclass version.

correctly classified instances is lowered, but that of misclassified examples is increased, by means of

$$w_i \leftarrow \frac{w_i \exp\left(-\alpha^j y_i A^j(\mathbf{x}_i)\right)}{\sum_i w_i \exp\left(-\alpha^j y_i A^j(\mathbf{x}_i)\right)}. \tag{19}$$

Notice, that an instance is never completely removed unless $\alpha^j \rightarrow \infty$, which is prevented by the penalty term in (13).

E. Implementation Issues of the Evolutionary Algorithms

Adaboost depends on a procedure that fits a weak learner to the weighted training set. We have seen that in our case the weak learning process generates the pair (A^h, c^h) that minimizes the fitness function explained in Section III-B.

The optimization problem is different for approximate and descriptive fuzzy rules. Descriptive rule bases require to search a finite hypothesis space, but approximate rule bases require it to explore a continuous search space defined by the parameters of the fuzzy sets that form the antecedents of the rules. According to the nature of the optimization problem, an integer coded genetic algorithm solves the weak learning of descriptive bases, and an evolution strategy learns approximate rules. Details about population sizes, type and probability of operators algorithms are given in Section IV; this section focuses on the coding of fuzzy rules and membership functions and the generation of the initial population.

1) Coding of Fuzzy Memberships in Descriptive Rules: Descriptive fuzzy rules are usually coded by a pair of integers: the index j of the antecedent A^j in $\mathcal{A}$, and the consequent c^j, but the search process is faster if the rule instead is encoded by a sequence of n integers: that refer to labels of the linguistic terms in the underlying fuzzy partitions.

In addition to the concrete linguistic labels such as "LOW" and "HIGH," each linguistic variable includes a wild card label "ANY VALUE," with a membership degree of 1 across

Input data:

Training set $(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_m, z_m), \quad \mathbf{x}_i \in \mathbf{R}^n,\ z_i \in \{1, \ldots, p\}$

Number of fuzzy rules $H \in \{1, \ldots, N\}$

Local Variables:

$w \in \mathbf{R}^m$ (weights of the examples in the training set)

$\alpha \in \mathbf{R}^{N \times p}$ (votes of the weak hypotheses)

$s : \mathcal{A} \times \{-1, 1\} \rightarrow [0, 1]$ (fuzzy relationship = consequents of rules)

$c \in \{-1, 1\}^m$ (consequents of the weak hypotheses)

$y \in \{-1, 1\}^m$ (labels of the examples in the two-class problems)

Local Procedures:

Generate a new fuzzy rule: see Figure 4

Convert votes into confidences: **begin**

$s_k^j \leftarrow 0$

for j in $1 \ldots N$

for k in $1 \ldots p$

if $(\alpha_k^j > 0)$ $s_k^j \leftarrow s_k^j + \alpha_k^j$

else $s_1^j \leftarrow s_1^j - \alpha_k^j, \ldots s_{k-1}^j \leftarrow s_{k-1}^j - \alpha_k^j, s_{k+1}^j \leftarrow s_{k+1}^j - \alpha_k^j, \ldots s_p^j \leftarrow s_p^j - \alpha_k^j$

end-for

end-for

Normalize consequents: $s_k^j \leftarrow s_k^j / \max s_k^j$

end

begin of main algorithm

for $k = 1, \ldots, p$

Initialize $w_i \leftarrow 1/m$, $\alpha_k^j = 0$; if $(z_i = k)$ $y_i = 1$ else $y_i = -1$

repeat H/p times

Generate a new fuzzy rule " if $\mathbf{x}$ is A^h then c^h"

Calculate the number of votes of the rule: votes(if $\mathbf{x}$ is A^h then c^h) $= \alpha_k^h$

Update the weights of the examples

end-repeat

end-for

Convert votes into confidences

Output the classifier: for all j, if any $s_k^j \neq 0$ emit the rule

if x_1 is A_1^j and $\ldots x_n$ is A_n^j then tr$(c_0) = s_0^j, \ldots,$ tr$(c_p) = s_p^j$

end of main algorithm

Fig. 5. Adaboost algorithm applied to the induction of a descriptive, multiclass, fuzzy-rule-based classification system.

the entire universe of discourse. The linguistic expression of a fuzzy rule that contains a wild card can be simplified, as illustrated by the following example: Assume a classification problem with two features (weight, height), where height={low, high} and weight={light, heavy}. The two linguistic variables are extended with a wild card term such that height={low, high, anyvalue} and weight={light, heavy, anyvalue}. If the rule antecedent is described by "anyvalue $\times$ heavy," the linguistic expression

$$\text{if height is anyvalue and weight is heavy then class} = 1$$

is simplified to

$$\text{if weight is heavy then class} = 1.$$

This property of the genetic representation allows it to code general rules that only refer to a subset of all possible features, thus enabling the boosting algorithm to take advantage of feature selection.

The last rule is coded by the sequence (3,2,1), as "anyvalue" is the third linguistic term in the first linguistic variable, "heavy" is the second term in the second linguistic variable and "1" is the class label in the rule consequent.

2) Coding of Fuzzy Membership in Approximate Rules: The membership function of trapezoidal fuzzy sets is parameterized by four characteristic points: A_f^j is defined by the points a_f^j, b_f^j, c_f^j, and d_f^j thus A^j is encoded by the sequence $a_1^j, b_1^j, c_1^j, d_1^j, \ldots a_n^j, b_n^j, c_n^j, d_n^j$. Rather than encoding the absolute values, the chromosome encodes the left most point and the distances between successive points. This representation preserves the order of points as long as the transformed parameters

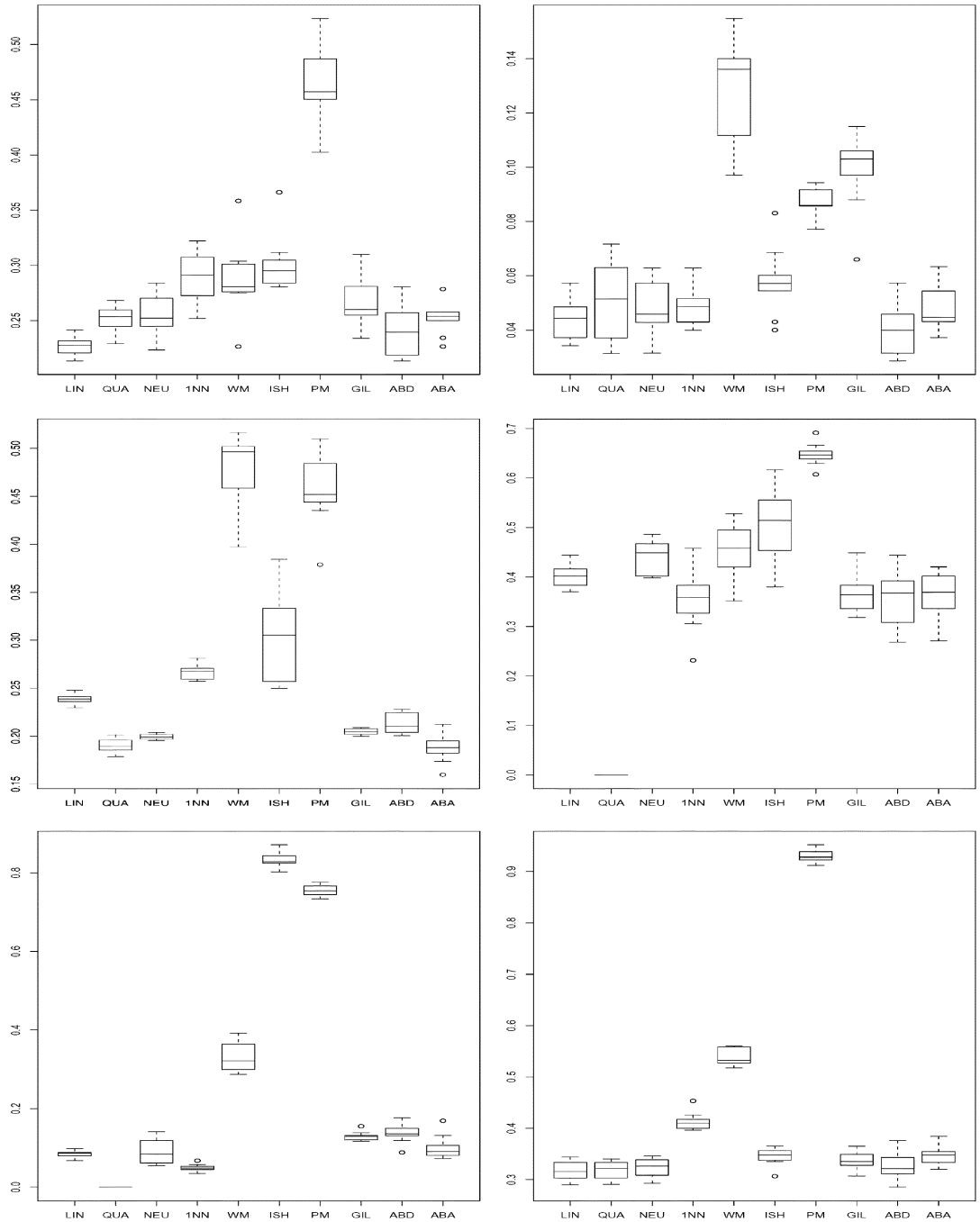


Fig. 6. Dispersion of test errors for Pima, Cancer, Gauss, Glass, Image, and Gauss5 datasets, over ten experiments with 50% train/test partitions (5 × 2 cv statistical test settings). Columns are named as follows. LIN: Linear. QUA: Quadratic. NEU: Neural Network. 1NN: Nearest Neighbor. WM: Wang and Mendel. ISH: Ishibuchi. PM: Pal and Mandal. GIL: Genetic Iterative Rule Learning. ABD, ABA: Fuzzy Adaboost, descriptive and approximate rules. The first three datasets are two-class problems, the last three are multiclass. (5 × 2 cv statistical test settings).

are restricted to positive values. The entire rule chromosome is formed by a real-valued vector that is the concatenation of the individual fuzzy set code segments.

As the number of rules required to cover the entire input space grows rapidly with the number of features the coding scheme provides for general rule antecedents that only refer to a subset of attributes. The chromosome contains an additional bitstring $S = \{s_1, \ldots, s_n\}$ in which the bit s_f indicates whether the input clause x_f is A_f^j occurs or is omitted in the rule antecedent. Adaptation of the bitstring S enables the evolutionary algorithm to take advantage of feature selection and to focus on fuzzy rules based on the most relevant attributes for classification.

3) Initial Population in Approximate Rules: The initial population is randomly generated for descriptive classifiers. This approach is less efficient for approximate classifiers due to the more complex search space.

The initialization scheme uses the training instances as prototypes to seed the membership function parameters of individuals in the first generation. The initialization scheme randomly picks a pair of training instances $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ with the same

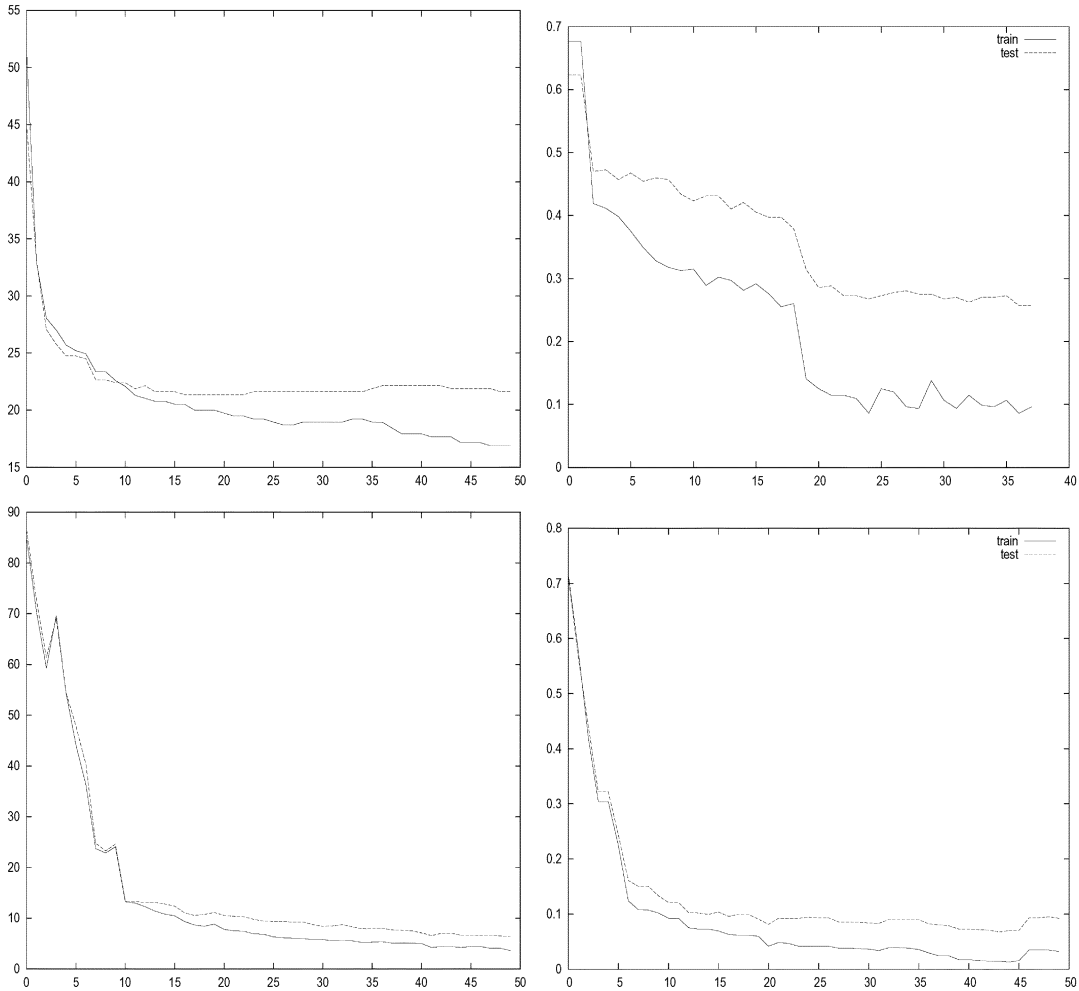


Fig. 7. Evolution of train and test errors versus the number of rules in ABD (left) and ABA (right) methods in a typical execution, for PIMA (upper part) and IMAGE (lower part) datasets. Observe that Adaboost based learning is robust to overfitting.

TABLE I
MEAN VALUES OF TEST RESULTS OVER THE SIX DATASETS STUDIED. DIFFERENCES BETWEEN VALUES IN THIS TABLE ARE ONLY MEANINGFUL IF THEIR CORRESPONDING P-VALUE IS LOWER THAN 0.05 (SEE TABLES II, III, IV, V, VI, AND VII)

	LIN	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
pima	0.227	0.252	0.255	0.289	0.287	0.301	0.464	0.269	0.241	0.252
cancer	0.044	0.051	0.047	0.048	0.129	0.058	0.087	0.099	0.040	0.049
gauss	0.239	0.190	0.200	0.267	0.477	0.304	0.457	0.205	0.213	0.187
glass	0.403	-	0.439	0.354	0.453	0.503	0.647	0.363	0.359	0.359
image	0.084	-	0.090	0.049	0.329	0.833	0.755	0.130	0.136	0.100
gauss5	0.317	0.317	0.323	0.413	0.539	0.344	0.931	0.338	0.327	0.347

class label $y_1 = y_2$. The likelihood of a training instance $\mathbf{x}_i$ to serve as a prototype is proportional to the weight w_i assigned to it by the boosting algorithm. Let $A = A_1 \times A_2 \times \ldots \times A_n \in \mathcal{A}$ be a generic antecedent of a rule in the initial population. Trapezoidal sets A_f are selected such that the two training examples span the core of the fuzzy sets

$$\begin{aligned} b_f &= \min\{x_{1f}, x_{2f}\} \\ c_f &= \max\{x_{1f}, x_{2f}\} \end{aligned} \tag{20}$$

as depicted in Fig. 3. The left and right most points a_f, d_f of the fuzzy set A_f are computed such that the support of A_f is twice as large as its core

$$\begin{aligned} a_f &= b_f - \frac{1}{2}|x_{1f} - x_{2f}| \\ d_f &= c_f + \frac{1}{2}|x_{1f} - x_{2f}|. \end{aligned} \tag{21}$$

The width of the core $|x_{1f} - x_{2f}|$ also determines the initial values of the mutation rates $\sigma_{a_f,b_f,c_f,d_f} = (1/5)|x_{1f} - x_{2f}|$ in the evolution strategy.

TABLE II
P-VALUES OF 5 × 2 CV STATISTICAL TEST, PIMA DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	0.17	0.08	0.01	0.11	0.00	0.00	0.14	0.08	0.08
QUA		0.62	0.12	0.47	0.00	0.01	0.57	0.40	0.87
NEU			0.08	0.73	0.03	0.01	0.79	0.73	0.33
1NN				0.84	0.03	0.01	0.68	0.71	0.00
WM					0.05	0.03	0.92	0.90	0.47
ISH						0.14	0.01	0.01	0.01
PM							0.05	0.02	0.00
GIL								0.96	0.49
ABD									0.40

TABLE III
P-VALUES OF 5 × 2 CV STATISTICAL TEST, CANCER DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	0.86	0.63	0.47	0.16	0.36	0.03	0.22	0.05	0.49
QUA		0.91	0.80	0.17	0.58	0.14	0.31	0.63	0.77
NEU			0.78	0.11	0.10	0.01	0.13	0.59	0.54
1NN				0.05	0.20	0.01	0.02	0.75	0.76
WM					0.19	0.68	0.20	0.07	0.08
ISH						0.05	0.50	0.03	0.21
PM							0.06	0.01	0.00
GIL								0.04	0.05
ABD									1.00

TABLE IV
P-VALUES OF 5 × 2 CV STATISTICAL TEST, GAUSS DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.10	0.03
QUA		0.13	0.00	0.00	0.10	0.00	0.06	0.17	0.02
NEU			0.00	0.00	0.10	0.00	0.18	0.40	0.30
1NN				0.00	0.36	0.00	0.00	0.00	0.01
WM					0.02	0.80	0.00	0.00	0.00
ISH						0.11	0.11	0.16	0.14
PM							0.00	0.00	0.00
GIL								0.68	0.45
ABD									0.87

F. Extension to Multiclass Problems

The extension of two-class Adaboost to multiclass problems is not unique. Let us first write (see Fig. 4) an straightforward extension of the algorithm given in Fig. 2. The fitness function derived from confidence-rated Adaboost is defined by

$$\text{fitness}(A^h, c^h) = \sum_{i=1,\ldots,m:y_i=c^h} w_i \exp\left(-A^h(\mathbf{x}_i)\right) + \sum_{i=1,\ldots,m:y_i\neq c^h} w_i \exp\left(A^h(\mathbf{x}_i)\right). \tag{22}$$

TABLE V
P-VALUES OF 5 × 2 CV STATISTICAL TEST, GLASS DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	-	0.14	0.42	0.79	0.11	0.00	0.18	0.73	0.54
QUA		-	-	-	-	-	-	-	-
NEU			0.19	0.35	0.50	0.00	0.05	0.28	0.25
1NN				0.41	0.03	0.00	0.88	0.80	0.75
WM					0.14	0.00	0.11	0.72	0.58
ISH						0.20	0.03	0.21	0.09
PM							0.00	0.01	0.00
GIL								0.75	0.75
ABD									0.89

The formulas for determining the number of votes and for weight updates in the same approach also need to be adapted. The number of votes is the value α^h that minimizes

$$Z(\alpha) = \sum_{i=1,\ldots,m:y_i=c^h} w_i \exp\left(-\alpha A^h(\mathbf{x}_i)\right) + \sum_{i=1,\ldots,m:y_i\neq c^h} w_i \exp\left(\alpha A^h(\mathbf{x}_i)\right) \tag{23}$$

and the weights are updated according

$$\textbf{if}\,(y_i = c^h) w_i \leftarrow \frac{w_i \exp\left(-\alpha^h A^h(\mathbf{x}_i)\right)}{\sum w_i} \quad \textbf{else}\, w_i \leftarrow \frac{w_i \exp\left(\alpha^h A^h(\mathbf{x}_i)\right)}{\sum w_i}. \tag{24}$$

The analytical approximation in (17) and (18) can be used without modifications. Apart from these modifications, resulting from the differences in the notation (classes numbered from 1 to p instead of -1 and 1), there is a significant alteration in the routine that assigns confidences to the consequents of the fuzzy rules. Observe that a negative number of votes produces a type 3 fuzzy rule because in two-classes problems the sentence "$\mathbf{x}$ does not belong to class 1" implies that $\mathbf{x}$ is in class -1, but in multiclass problems the same assertion only implies that $\mathbf{x}$ can be in classes 2 to p.

A theoretically more efficient extension of the two-class problem, inspired by the relationship between probabilistic classifiers and boosting stated in [7], is outlined in Fig. 5. Following the results in this article, it can be concluded that the solution of a p-class problem is equivalent to solve p corresponding two-class problems. In each two-class problem, one class is classified against all the other classes. The final rule base for the multiclass is obtained by merging the rule bases generated for the m two-class problems.

Let us define the "two-class problem number k" ($k = 1\ldots,p$) as that of labeling with the number "1" all examples in class k, and labeling with the number "-1" all other examples in the sample. After applying the algorithm discussed in the preceding section to the problem number k, we obtain a classification system comprised of fuzzy rules such as

$$\text{if } x_1 \text{ is } A_1^j \text{ and } \ldots \text{ and } x_n \text{ is } A_n^j \text{ then } \text{tr}(\text{class} = 1) = s_1^j \text{ and } \text{tr}(\text{class} = -1) = s_{-1}^j. \tag{25}$$

TABLE VI
P-VALUES OF 5 × 2 CV STATISTICAL TEST, IMAGE DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	-	0.46	0.13	0.00	0.00	0.00	0.03	0.17	0.97
QUA		-	-	-	-	-	-	-	-
NEU			0.80	0.00	0.00	0.00	0.09	0.18	0.72
1NN				0.00	0.00	0.00	0.01	0.03	0.49
WM					0.00	0.00	0.01	0.03	0.01
ISH						0.16	0.00	0.00	0.00
PM							0.00	0.00	0.00
GIL								0.61	0.33
ABD									0.29

TABLE VII
P-VALUES OF 5 × 2 CV STATISTICAL TEST, GAUSS-5 DATASET. LOW VALUES (<0.05) MEAN THAT THE DIFFERENCE BETWEEN TWO ALGORITHMS IS STATISTICALLY RELEVANT

	QUA	NEU	1NN	WM	ISH	PM	GIL	ABD	ABA
LIN	1.00	0.03	0.02	0.00	0.13	0.00	0.23	0.15	0.10
QUA		0.04	0.01	0.00	0.06	0.00	0.23	0.13	0.10
NEU			0.02	0.00	0.48	0.00	0.88	0.60	0.57
1NN				0.00	0.04	0.00	0.01	0.02	0.06
WM					0.00	0.00	0.00	0.00	0.00
ISH						0.00	0.51	0.37	0.92
PM							0.00	0.00	0.00
GIL								0.96	0.57
ABD									0.49

These rules can be replaced by their equivalent p-class consequents, by substituting the "all-other-classes"-label c_B with the corresponding class label c_k of the multiclass problem, i.e.,

if x_1 is A_1^j and $\ldots$ and x_n is A_n^j then

$$\mathrm{tr}(c_1, \ldots, c_k, \ldots, c_p) = \left(s_{-1}^j, \ldots, s_{-1}^j, s_1^j, s_{-1}^j, \ldots, s_{-1}^j\right). \tag{26}$$

We obtain p fuzzy rule bases, that are merged into the overall multiclass rule base. It is important not to normalize the number of votes until all p binary problems are solved as all confidence degrees must be divided by the same normalization factor to obtain a meaningful decomposition; the procedure used to convert votes into confidences is shown in Fig. 5.

IV. NUMERICAL RESULTS

Three two-class and three multiclass problems were selected to assess the performance of the proposed fuzzy boosting method. Four of the datasets stem from the UCI ML database and the other two are artificially generated Gaussian distributions of which the theoretical error bound is known. The two-class problems are Pima and Breast Cancer from UCI and an artificial dataset defined in [12], comprising 4000 samples of two equiprobable bidimensional Gaussian distributions with centers in (0,0) and (2,0) and covariances matrices I and $4I$, which makes a quadratic problem. Multiclass problems are glass and image recognition from UCI and a synthetic problem called "Gauss-5," comprising 50, 100, 150, 200, and 250 samples from five bidimensional Gaussian distributions with centers in (0,0), $(-1,-1)$, $(-1,1)$, $(1,-1)$, and (1,1) and unity covariances matrix.

An evolution strategy was used to minimize the fitness function in approximate rule bases, and an integer coded genetic algorithm in descriptive bases. For this last case, Ruspini's partitions of size 3 were used for all datasets but Image, with only two linguistic elements. All fuzzy sets are triangular and their support set is given by the range of their corresponding variable in the training set.

The rule generation processes were terminated after 30 rules for Pima, 40 for Cancer, 16 for Gauss, 100 for Glass, 125 for Image, and 50 for Gauss5; the size of the rule base for Adaboost problems was determined after applying the tests in [24]. We combined the optimization-based fitness, number of votes and weight updating (derived from confidence-rated Adaboost) with descriptive rules, and the analytic approximation of these three quantities with approximate rule bases. In multiclass problems, the first version of the algorithm (Fig. 4) was used, and the second one was combined with descriptive bases (Fig. 5).

The evolution strategy in approximate Adaboost uses a deterministic (μ, λ) selection scheme, with $\mu = 10$ and $\lambda = 200$, which run over 40 generations to obtain the best rule. The genetic algorithm in descriptive Adaboost is steady-state, with a population size of 100 and every rule is obtained from the best individual in the population after 1000 crossover operations.

We used Dietterich's 5 × 2 cv statistical contrast, which is considered to be superior to paired k-fold cross validation in classification problems [6]. Four statistical methods (linear and quadratic discriminant analysis, two layer perceptron—with a hidden layer of size equal to half the number of features—and nearest neighbor) plus four well-known fuzzy descriptive rule-based classifiers (Wang and Mendel's [27], Ishibuchi's [15], Pal and Mandal's [22] and genetic iterative learning [5]) were compared to fuzzy Adaboost.

There is not statistical evidence against boosted classifiers being equal to the best classifiers for all datasets, neither there is evidence about the settings derived from confidence rated Adaboost being superior to that of the original Adaboost when evolving fuzzy rule bases. It seems that the version of the algorithm in Fig. 5 is more efficient in multiclass problems, even though this observation was not true for all experiments and the relevance of the difference is not statistically significant, thus further experimentation is needed before stating a definite conclusion.

The performance of Adaboost is better than Wang and Mendel's, Ischibuchi, or Pal–Mandal's algorithms, besides the variance in the output of these methods can be so high that it sometimes prevents statistical contrasts from giving useful information (see Fig. 6). Adaboost also improves GIL, besides the statistical difference is only significant in one of the experiments. Notice that current implementations of GIL algorithm are slower than Adaboost by more than one order of magnitude. The size of the final rule base is also much larger in GIL, but it is not meaningful to compare them, because GIL adjusts membership functions (Adaboost do not) but Adaboost weights the consequents of the rules and GIL produces type 1 fuzzy rules.

Fig. 7 provides graphical insight into the average performance of the method and about the evolution of training and test errors as the number of rules increases, showing robustness toward overfitting either in two-class and multiclass, approximate and descriptive problems. Table I contains the means of the test errors of all experiments. Tables II–VII contain the p-values of all statistical contrasts. Differences between values in Table I are only meaningful when the corresponding value in their p-values table is lower than 0.05.

V. Concluding Remarks

Boosting of fuzzy rule bases can be regarded as an extension to the Iterative Learning. Boosting rather down-weights instances instead of removing them, and thereby implicitly promotes cooperation among rules when building the rule base in an incremental fashion. Due to this property of the Adaboost scheme the otherwise mandatory simplification stage of iterative rule learning becomes obsolete.

The Adaboost approach was compared with other fuzzy and nonfuzzy classification methods and demonstrated no statistically relevant difference in performance to the best classifier on the respective datasets. Also, the Adaboost algorithm produced more compact results than other genetic fuzzy systems. Rule bases depend on fewer parameters than those induced with GIL or other rule generation methods. This obviously favors the interpretability, but the non standard fuzzy reasoning method of Adaboost is a detrimental to the objective of rule comprehensibility. In applications for which interpretability is more crucial than classification accuracy nonweighted fuzzy rules may be preferred over the weighted representation generated by Adaboost. Anyway, there exists a clear advantage of Adaboost over their competitors: learning time. Even if we lack a detailed study by now, our experimentation seems to show important differences, from minutes to days, in the time required to solve some of the classification problems included in this paper.

References

[1] R. Alcalá, J. Casillas, J. L. Castro, A. González, and F. Herrera, "A multicriteria genetic tuning for fuzzy logic controllers," *Mathware Soft Comput.*, vol. 8, no. 2, pp. 179–201, 2001.

[2] L. Castillo, A. González, and A. Pérez, "Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm," *Fuzzy Sets Syst.*, vol. 120, pp. 309–321, 2001.

[3] O. Cordón and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples," *Int. J. Approx. Reason.*, vol. 17, no. 1, pp. 369–407, 1997.

[4] O. Cordón, M. J. del Jesus, and F. Herrera, "Genetic learning of fuzzy-rule-based classification systems co operating with fuzzy reasoning methods," *Int. J. Intell. Syst.*, vol. 13, no. 1011, pp. 1025–1053, Nov. 1998.

[5] ——, "A proposal on reasoning methods in fuzzy-rule-based classification systems," *Int. J. Approx. Reason.*, vol. 20, no. 1, pp. 21–45, 1999.

[6] G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1924, 1998.

[7] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–374, 2000.

[8] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Machine Learning*, pp. 148–156.

[9] A. González and F. Herrera, "Multi-stage genetic fuzzy systems based on the iterative rule learning approach," *Mathware Soft Comput.*, vol. 4, no. 3, pp. 233–249, 1997.

[10] A. González and R. Pérez, "Completeness and consistency conditions for learning fuzzy rules," *Fuzzy Sets Syst.*, vol. 96, pp. 37–51, 1998.

[11] A. Gonzalez and R. Perez, "SLAVE: a genetic learning system based on the iterative approach," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 176–191, Apr. 1999.

[12] S. Haykin, *Neural Networks*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[13] F. Hoffmann, "Boosting a genetic fuzzy classifier," in *Proc. Joint 9th IFSA World Congr. 20th NAFIPS Int. Conf.*, vol. 3, Vancouver, BC, Canada, July 2001, pp. 1564–1569.

[14] T. P. Hong and C. Y. Lee, "Induction of fuzzy rules and membership functions from training examples," *Fuzzy Sets Syst.*, vol. 84, pp. 33–47, 1996.

[15] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets Syst.*, vol. 52, pp. 21–32, 1992.

[16] H. Ishibuchi, T. Nakashima, and T. Morisawa, "Voting in fuzzy-rule-based systems for pattern classification problems," *Fuzzy Sets Syst.*, vol. 103, no. 2, pp. 223–239, 1999.

[17] L. Junco and L. Sánchez, "Using the Adaboost algorithm to induce fuzzy rules in classification problems," in *Proc. ESTYLF 2000*, Seville, Spain, 2000, pp. 297–301.

[18] L. I. Kuncheva, *Fuzzy Classifier Design*. New York: Springer-Verlag, 2000.

[19] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.

[20] L. Magdalena, "A fuzzy logic controller with learning through the evolution of it knowledge base," *Int. J. Approx. Reason.*, vol. 16, pp. 335–358, 1997.

[21] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy-rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 238–250, Apr. 1996.

[22] S. K. Pal and D. P. Mandal, "Linguistic recognition system based on approximate reasoning," *Inform. Sci.*, vol. 61, pp. 135–161, 1992.

[23] E. H. Ruspini, "A new approach to clustering," *Inform. Control*, vol. 15, pp. 22–32, 1969.

[24] L. Sánchez, J. Casillas, O. Cordón, and M. J. Del Jesus, "Some relationships between fuzzy and random set-based classifiers and models," *Int. J. Approx. Reason.*, vol. 29, pp. 175–213, 2002.

[25] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learn.*, vol. 37, no. 3, pp. 297–336, 1999.

[26] R. E. Schapire, *Theoretical Views of Boosting and Applications*, ser. Lecture Notes in Artificial Intelligence. New York: Springer-Verlag, 1999, vol. 1720, pp. 13–25.

[27] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 353–361, Mar. 1992.

María José del Jesus received the M.Sc. and Ph.D. degrees in computer sciences from the University of Granada, Granada, Spain, in 1994 and 1999, respectively.

Currently, she is an Associate Professor with the Department of Computer Sciences, University of Jaén, Jaén, Spain. Her research interests are in the fields of fuzzy-rule-based systems, fuzzy and linguistic modeling, fuzzy classification, machine learning, and genetic fuzzy systems.

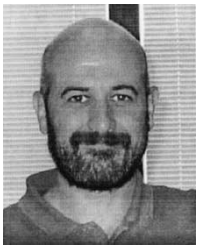
Frank Hoffmann received the Ph.D. degree in physics from the University of Kiel, Kiel, Germany in 1996.

From 1996 to 2000, he was with University of California at Berkeley as a Visiting Researcher. He was a Lecturer in computer science at the Royal Institute of Technology, Stockholm, Sweden, from 2000 to 2002. He is currently Interim Chair for control system design at the University of Dortmund, Dortmund, Germany. His research interests are in the area of machine learning, fuzzy control, soft computing and robotics.

Dr. Hoffman is General Chair of WSC8, Founding Member of the World Federation of Soft Computing, and Associate Editor of the IEEE Transactions on Systems, Man, and Cybernetics B.

Luis Junco Navascués received the M.Sc. degree in industrial engineering from the University of Oviedo, Oviedo, Spain, in 1990.

Currently, he is an Associate Professor with the Department of Computer Sciences, the University of Oviedo. His research interests are in the fields of fuzzy-rule-based systems, genetic algorithms, and scientific metrology.



Luciano Sánchez received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Oviedo, Oviedo, Spain, in 1991 and 1994, respectively.

Currently, he is an Associate Professor with the Department of Computer Sciences, the University of Oviedo. His research interests are in the fields of fuzzy-rule-based systems, fuzzy and random sets-based classification, scientific metrology, genetic algorithms, and genetic programming.