Soft Comput (2006) 10: 825–835
DOI 10.1007/s00500-005-0011-0

**FOCUS**

**José Otero · Luciano Sánchez**

# Induction of descriptive fuzzy classifiers with the Logitboost algorithm

Published online: 12 October 2005

**Abstract** Recently, Adaboost has been compared to greedy backfitting of extended additive models in logistic regression problems, or "Logitboost". The Adaboost algorithm has been applied to learn fuzzy rules in classification problems, and other backfitting algorithms to learn fuzzy rules in modeling problems but, up to our knowledge, there are not previous works that extend the Logitboost algorithm to learn fuzzy rules in classification problems.

In this work, Logitboost is applied to learn fuzzy rules in classification problems, and its results are compared with that of Adaboost and other fuzzy rule learning algorithms. Contradicting the expected results, it is shown that the basic extension of the backfitting algorithm to learn classification rules may produce worse results than Adaboost does. We suggest that this is caused by the stricter requirements that Logitboost demands to the weak learners, which are not fulfilled by fuzzy rules. Finally, it is proposed a prefitting based modification of the Logitboost algorithm that avoids this problem.

**Keywords** Genetic fuzzy systems · Descriptive fuzzy rules · Fuzzy adaboost · Fuzzy LogitBoost

## 1 Introduction

Boosting consists in combining low quality classifiers with a voting scheme to produce a classifier better than any of its components. The most common version of Boosting is called AdaBoost [5]. Recently, a close relationship between this method and Generalized Additive Models has been shown. Following [6], Adaboost is a specialization of the backfitting algorithm – used since the 80's to induce generalized additive models – whose greedy version is also known as "matching pursuit" in signal processing related works [17, 23]. This relationship explains the mechanisms of Adaboost in terms of iterative approximations to maximum likelihood estimation over a family of additive models.

As a consequence of the new theoretical justification, some corrections were introduced to Adaboost and a new boosting method, called LogitBoost, was proposed in [6]. Logitboost should pose less numerical problems than Adaboost, and it was experimentally shown to improve the former method, being specially efficient in multiclass problems, to which Adaboost was difficult to extend.

Matching pursuit methods have been used to induce fuzzy classifiers and models in different ways. In fact, Iterative Rule Learning of models [8] and classifiers [2] are closely related to matching pursuit algorithms and can be regarded as the precursors of these algorithms, and Adaboost itself was directly applied to induce fuzzy classifiers [15,12,3]. Backfitting has also been regarded as the counterpart of Adaboost in model estimation and also used to induce fuzzy models in previous works [20].

The structure of this paper is as follows: in the next section, fuzzy classifiers are introduced and it is explained how Adaboost can be applied to induce them from data. Then, the Logitboost algorithm is explained, compared to Adaboost and adapted to learn fuzzy classifiers. Some problems with this adaptation are discussed, and an extension to Logitboost is introduced. The paper finishes with an empirical evaluation of the new algorithm.

## 2 Boosting fuzzy classifiers

### 2.1 Notation

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

The antecedents of all fuzzy rules in the classifier form a fuzzy partition $\mathcal{A}$ of the feature space $\mathcal{A} = \{A^j\}_{j=1,\dots,N}$, with $A^j \subset \widetilde{\mathcal{P}}(\mathbf{X})$, where $\widetilde{\mathcal{P}}(\mathbf{X})$ stands for "fuzzy parts of $\mathbf{X}$".

J. Otero · L. Sánchez (✉)
Universidad de Oviedo, Depto. Informática, Sedes departamentales, despacho 1.1.28, Campus de Viesques, Gijón 33203, Spain
E-mail: luciano@ccia.uniovi.es

In the remaining part of this paper, we will assume that the training examples will be 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, if we write "for all $\mathbf{x}_i$" we mean $\mathbf{x}_i$, $1 \leq i \leq m$; from now on, this range will not be explicitly stated unless its absence leads to confusion.

We will define a fuzzy rule based classifier by means of a fuzzy relationship defined on $\mathcal{A} \times \{1, \ldots, p\}$. Values of this relationship describe the degrees of compatibility between the fuzzy subsets of the feature space collected in $\mathcal{A}$, and each one of the classes. In other words, for every antecedent $A^j$ we have $p$ numbers between 0 and 1 that represent our degree of knowledge about the assert "All elements in the fuzzy set $A^j$ belong to class number $k$". Values near to 1 mean "high confidence," and values near 0 mean "absence of knowledge about the assertion."

## 2.2 Linguistic interpretation of fuzzy classifiers

Fuzzy rule based classifiers are understandable to humans as they can be expressed as 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,

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

into a single sentence, as follows:

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

Furthermore, the antecedents of various rules with the same consequent

$$\begin{array}{l}\text{if } \mathbf{x} \text{ is } A \text{ then truth}(c_1) = s_1 \text{ and } \cdots \text{ and truth}(c_p) = s_p \\ \text{if } \mathbf{x} \text{ is } A' \text{ then truth}(c_1) = s_1 \text{ and } \cdots \text{ and truth}(c_p) = s_p\end{array}$$

can be combined with the help of the "or" connective, giving a compound rule:

$$\begin{array}{l}\text{if } (\mathbf{x} \text{ is } A) \text{ or } (\mathbf{x} \text{ is } A') \text{ then truth}(c_1) = s_1 \\ \qquad \text{and } \cdots \text{ and truth}(c_p) = s_p.\end{array}$$

In practical cases, we work with asserts $A^j$ that can be decomposed in a Cartesian product of fuzzy sets defined over each feature, $A^j = A_1^j \times A_2^j \times \cdots \times A_n^j$, thus the rules are

$$\begin{array}{l}\text{if } (x_1 \text{ is } A_1^j \text{ and } \cdots \text{ and } x_n \text{ is } A_n^j) \text{ or } (x_1 \text{ is etc. }) \\ \text{then truth}(c_1) = s_1^j \text{ and } \cdots \text{ and truth}(c_p) = s_p^j.\end{array}$$

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

We can restrict the definition further 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.

Observe 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.

## 2.3 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 classifier being approximate or descriptive. 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 [16] or the arithmetic sum, so called "maximum voting scheme" [14]. "$\wedge$" is always a $t$-norm, usually the minimum or the product. In this paper, we will combine the product with the maximum vote scheme to do the fuzzy inference.

## 2.4 The 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 boosting literature) with a voting scheme to produce 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 weak hypothesis, and a fuzzy rule base can be compared to 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 can be encoded with 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 the better. AdaBoost is intended to produce a linear threshold of all hypotheses:

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

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, given a weighted data set. There are different algorithms for assigning a number of votes to a weak hypothesis, and for adjusting

```
Given: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m), \quad \mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/m$
Select the number of weak hypotheses $N$
For $j = 1, \ldots, N$:
    1. Get weak hypothesis $g^j : \mathbf{X} \to \mathbf{R}$
    2. Find numerically the value $\alpha_j$ that minimizes
       $Z_j(\alpha) = \sum_{i=1}^{m} D_j(i) \exp(-\alpha y_i g^j(\mathbf{x}_i))$
    3. Update the weights:
       $D_{j+1}(i) = \frac{D_j(i) \exp(-\alpha_j y_i g^j(\mathbf{x}_i))}{Z_j}$
       where $Z_j$ is a normalization factor, so that $D_{j+1}$ is a distribution.
Output the final hypothesis
    $H(\mathbf{x}) = \operatorname{sign}\left(\sum_{j=1}^{N} \alpha_j g^j(\mathbf{x})\right)$
```

**Fig. 1** Generalized Adaboost algorithm. Two classes version

the weights of the examples. For example, in confidence-rated Adaboost [22] the number of votes of the weak hypothesis $g^h$ is given by the value $\alpha^h$ that minimizes the following function:

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

and the weights of the examples are updated according to the formula

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

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

### 2.5 Boosting fuzzy rules

Fuzzy rules are weak learners in fuzzy boosting. Each fuzzy rule is a confidence rated classifier that can produce the output '0' if the pattern is not covered by its antecedent, or both a class number and a confidence value between 0 and 1 else [3]. Therefore, boosting fuzzy rules can be based on an algorithm able to fit one single fuzzy rule to a set of weighted examples. This algorithm will be repeated so many times as rules in the base, and the Adaboost algorithm produces the number of votes each rule is assigned and recalculates the weight of every example when the rule is added to the base.

For the sake of simplicity, we restrict the discussion for the time being to two-class problems. A function $R_j(\cdot)$ can be assigned to the rule

$$\text{if } x_1 \text{ is } A_1^j \text{ and } \cdots \text{ and } x_n \text{ is } A_n^j \text{ then } \mathrm{t}(c_1) = s_1 \text{ and } \mathrm{t}(c_2) = s_2$$

$R_j(\mathbf{x})$ is defined as the product of the membership degree of instance $\mathbf{x}$ with the rule antecedent and the difference between the degrees of truth of the two classes in its consequent: $R_j(\mathbf{x}) = A^j(\mathbf{x})(s_1 - s_2)$. Assuming the product as the conjunction operator $\wedge$, the output of the fuzzy classifier given in Eq. 1 can be written as

$$\operatorname{sign}\left(\sum_{j=1}^{N} R_j(\mathbf{x})\right).$$

Noticing, the similarity between the above expression and Eq. 2, it allows us to apply the boosting mechanism to descriptive fuzzy rules. The space of weak hypotheses becomes identified with the fuzzy partition $\mathcal{A}$. A linear threshold of elements of $\mathcal{A}$ is

$$\operatorname{sign}\left(\sum_{j=1}^{N} \alpha_j A_j(\mathbf{x})\right)$$

and the values of $\alpha_j$, along with the $N$ elements $A_j$ selected from $\mathcal{A}$ are obtained by the usual Adaboost algorithm. Positive values of $\alpha$ correspond to rules for which $s_1 > s_2$ and negative ones to rules with $s_2 > s_1$. Since the values of $\alpha_j$ that Adaboost produces are not constrained to the interval [0, 1], it may happen that they no longer constitute valid confidence rates. Therefore, the degrees $\alpha_j$ in the consequents are normalized to a range $[-1, 1]$ once the entire rule base has been generated.

Figure 2 shows the outline of the final algorithm, as proposed in [15,3].

## 3 Backfitting additive logistic classifiers: the Logitboost algorithm

The Logitboost algorithm justifies the exponential bound introduced in the preceding section as an approximation to the objective function originated when a Generalized Additive Linear Model is used to fit a classification problem after certain logistic transform. Logitboost tries to minimize the likelihood of the classifier, which in turn is restricted to a parametric family of density functions. Before the algorithm is introduced, we will revisit the notions of statistical classification problem and generalized additive model.

### 3.1 Definition of an statistical classification problem

Let us suppose we have a set $\Omega$ that contains objects $\omega$, each one of them belonging to a class $c_i, i = 1, \ldots, p$, and we perform the set of measurements $X(\omega) = (x_1(\omega), \ldots, x_n(\omega))$ over every object (i.e., the features of an object are its image under a transform given by certain random variable that models the measuring process.) Let us also assume that the mapping $X$ fulfills all necessary conditions to be a random variable. We will say that a classification system is a decision rule that maps every element of $X(\Omega)$ to a class $c_i$, whose main objective is to produce a low number of errors.

```
Given: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$,   $\mathbf{x_i} \in \mathbf{R}^n$, $y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/m$, $s_1^j = 0$, $s_2^j = 0$
Select the number of rules $N$
For $j = 1, \ldots, N$:
  – Find the fuzzy membership $A^j \in \mathcal{A}$ that minimizes
    $Z = \min_{A\in\mathcal{A}} \left(\sum_{i=1}^m D_j(i)\exp(-y_i A(\mathbf{x}_i)),\ \sum_{i=1}^m D_j(i)\exp(y_i A(\mathbf{x}_i))\right)$
  – Find numerically the value $\alpha_j$ that minimizes
    $Z_j(\alpha) = \sum_{i=1}^m D_j(i)\exp(-\alpha y_i A^j(\mathbf{x}_i))$
  – If $\alpha_j > 0$ then $s_1^j = \alpha_j$ else $s_2^j = -\alpha_j$.
  – Update the weights:
    $D_{j+1}(i) = \frac{D_j(i)\exp(-\alpha_j y_i A^j(\mathbf{x}_i))}{K}$
    where $K$ is another normalization factor, so that $D_{j+1}$ is a distribution.
End For
$s_1^j = s_1^j / \max_{k,j}(s_k^j)$, $s_2^j = s_2^j / \max_{k,j}(s_k^j)$; $k = 1, 2$; $j = 1, \ldots, N$
Generate the rules
  if $x_1$ is $A_1^j$ and $\ldots x_n$ is $A_n^j$ then $\mathrm{tr}(c_1) = s_1^j$ and $\mathrm{tr}(c_2) = s_2^j$
```

**Fig. 2** Adaboost algorithm applied to the induction of a descriptive, fuzzy rule based classification system. Two classes version

Since we did not assume that $\omega_1 \neq \omega_2 \Rightarrow X(\omega_1) \neq X(\omega_2)$ perhaps a decision rule that never fails cannot be defined for this problem. Usually we evaluate the expectation of a new random variable that quantifies the mean number of errors, and try to optimize it. If the classifier is a decision rule, $D(X)$, and "class$(\omega)$" is the class of the object $\omega$ then the error is

$$\mathrm{err}(D) = \int_\Omega \mathrm{cost}(D(X(\omega)), \mathrm{class}(\omega))\,\mathrm{dP}$$

where $\mathrm{cost}(a, b) = 1$ when $a \neq b$ and 0 else. This value is called "minimum Bayes error", and it is a lower bound of the expected error of any classifier. It can be demonstrated that the classifier that reaches this bound has the form [9]

$$D(\mathbf{x}) = \arg\max_k P(\mathrm{class}(\omega) = c_k | X = \mathbf{x}).$$

## 3.2 Generalized and extended additive models

As a consecuence of the last assert, the objective of the learning process can be rewritten as "estimate $P(\mathrm{class}(\omega) = c_k | X = \mathbf{x})$." Alternatively, we can also define $p$ random variables

$$y_k(\omega) = \begin{cases} 1 & \text{if class } (\omega) = c_k \\ 0 & \text{else} \end{cases} \tag{5}$$

and reformulate the classification problem as a regression problem, that of estimating the conditional expectations $E(y_k|x) = P(\mathrm{class}(\omega) = c_k | X = \mathbf{x})$. We will show next that this allows us to apply certain statistical regression techniques to the classification problem.

## 3.3 Additive models

Additive models were introduced in the 80's to improve precision and interpretability of classical nonparametric regression techniques in problems with a large number of inputs. These models estimate an additive approximation to the multivariate regression function, where each of the additive terms is estimated using a univariate smoother.

Individual terms explain the dependence of the output variable with respect to their corresponding input variables, thus there exists a certain degree of interpretability in the model. While this kind of estimation avoids the curse of dimensionality, it is not able to approximate universally. Hastie and Tibshirani [10] addressed this issue and proposed generalized additive models. With these last models it is assumed that the mean of the output depends on a sum of terms through a nonlinear link function, and it is permitted that the response probability distribution is any distribution in the exponential family. Many statistical models belong to this class, including additive models for Gaussian data and nonparametric logistic models for binary data like the one we are interested in.

More formally, let $y$ be the output random variable we wish to model, and let $x = (x_1, \ldots, x_n)$ be the input random vector. The objective of the modeling process consists in estimating the conditional expectation of $y$ given $x$. Linear regression assumes

$$E(y|x) = f(x_1, \ldots, x_n) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{6}$$

and obtains $\beta_0, \ldots, \beta_n$ by least squares. Additive models generalize this schema by allowing the use of a sum of nonlinear univariate regressors

$$E(y|x) = f(x_1, \ldots, x_n) = r_0 + r_1(x_1) + \cdots + r_n(x_n) \tag{7}$$

where $r_i$ are smooth functions that are estimated in a nonparametric fashion. Generalized additive models extend additive models by not assuming a Gaussian distribution of the output, but any probability distribution in the exponential family,

$$f_y(t; \theta; \phi) = \exp\left\{\frac{t\theta - b(\theta)}{a(\phi)} + c(t, \phi)\right\} \tag{8}$$

and making the additive component

$$f(x_1, \ldots, x_n) = r_0 + r_1(x_1) + \cdots + r_n(x_n) \tag{9}$$

to depend on the mean of the output by means of a link function $l$, so that $g(E(y|x)) = f(x_1, \ldots, x_n)$. The most commonly used link function in practice is the canonical link $l(E(y|x)) = \theta$.

Additive models can be generalized furthermore. In extended additive models, the univariate regressors $r_i$ are replaced by functions of more than one feature. In our context, these functions usually depend on a set of parameters $\gamma$ and a multiplier $\beta$,

$$r_j = \beta_j r(x; \gamma_j) \tag{10}$$

thus the additive model becomes

$$E(y|x) = f(x_1, \dots, x_n) = r_0 + \sum_{j=1}^{N} \beta_j r((x_1, \dots, x_n); \gamma_j). \tag{11}$$

For example, in radial basis neural networks the functions $s(x, \gamma_j) = \exp\{||x - \gamma_j||^2\}$ are the "basis functions"; $\gamma_j$ are their centers and $\beta_j$ are the weights that connect the input layer with the output. In support vector machines, $r(x, \gamma)$ is a kernel, and $\gamma_j$ are the support vectors. In our case, we will propose a model where $(x, \gamma_j)$ will be the membership $A^j$ of the antecedent of the $i$-th fuzzy rule, $\gamma_j$ identifies the linguistic terms that participate in the rule and $\beta_j$ is the degree of truth of the consequent of the rule.

## 3.4 Backfitting and the Logitboost algorithm

Extended additive models can be learned with a generalized backfitting algorithm [6]. Given a cost function $d$, that measures the differences between the conditional expectation and its approximation, this algorithm consists in finding $N$ pairs of values $\{\beta_j, \gamma_j\}$ minimizing each

$$E\left[d\left(y, \sum_{\substack{\alpha=1,\dots,N \\ j\neq\alpha}} \beta_\alpha r(x; \gamma_\alpha) + \beta r(x; \gamma)\right)\right] \tag{12}$$

with respect to $\beta, \gamma$ [6]. A greedy approach, where the expectation of the output is incrementally approximated, produces good results in practice. Let $f_0(x), f_1(x), \dots$ be successive approximations to $E(y|x)$; then, let us define

$$\{\beta_\alpha, \gamma_\alpha\} \leftarrow \arg\min_{\beta,\gamma} E[d(y, f_{\alpha-1}(x) + \beta r(x; \gamma))] \tag{13}$$

where $\{\beta_k, \gamma_k\}_1^{\alpha-1}$ are fixed at their corresponding solution values at earlier iterations.

Algorithms that learn a weighted sum of basis functions, by sequentially appending functions to an initially empty basis, to approximate a target function in the least-squares sense, are contained in the family of the *matching pursuit* algorithms [17]. These algorithms have been compared to support vector machines [25] and radial basis neural networks in machine learning problems [23]. One of the most interesting properties of matching pursuit algorithms is that they are good in keeping the sparsity of the solution; this improves the generalization properties of the method and we will also see in the following sections that the same property guarantees a small number of rules in the fuzzy case that will be described later.

We have mentioned that the objective of a binary classification problem is to approximate the value $E(y|x) = p(c = 1|x)$, which we will denote by $p(x)$. The response variable in a classification problem follows the binomial distribution, and the link function is $\log(p(x)/(1 - p(x)))$ [10]; therefore, the additive model is

$$\log \frac{p(\text{class}(x) = 1)}{p(\text{class}(x) = 0)} = f(x_1, \dots, x_n) = r_0 + \beta_1 r_1(x) + \cdots \tag{14}$$

and the output of the model, reversing the logistic transform,

$$p(x) = \frac{e^{f(x)}}{1 + e^{f(x)}} \tag{15}$$

If the greedy version of generalized backfitting, mentioned in the preceding subsection, is applied to this model, it is obtained the Logitboost algorithm [6]. An outline of this algorithm, extended to multiclass problems, is shown in Fig. 3. Observe that our multiclass extension consists in fitting $p$ uncoupled additive logistic models, each class against the rest. The term $y_{ik}$ is defined as follows (recall Eq. 5:)

$$y_{ik} = \begin{cases} 1 & \text{if class } (\mathbf{x}_i) = k \\ 0 & \text{else} \end{cases} \tag{16}$$

## 3.5 The smoothing operation

The "smooth" operation [10], consists in estimating the values $\beta_j$ and $\gamma_j$ on which the $j$-th additive term depends, by means of a suitable statistical or machine learning procedure. Every step can be understood as fitting a new term to a weighted set of residuals of the previous submodel. This residual is

$$z = \frac{y - p_{j-1}(x)}{p_{j-1}(x)(1 - p_{j-1}(x))},$$

and the weight of the residual at the element $x$ in the sample is $p_{j-1}(x)(1 - p_{j-1}(x))$. Recall that this, in our case, is equivalent to find the fuzzy rule that best fits the residual of the $j - 1$ first rules in the knowledge base (this rule is given by the parameter $\gamma_j$) and assigning a degree of confidence to this rule (the parameter $\beta_j$.)

We have conducted the search of $\gamma_j$ by means of a Genetic Algorithm, hybridized with an analytical procedure. It is clear that, once $\gamma_j$ is selected, the value of $\beta_j$ that minimizes the squared error over the residual can be found by differentiating with respect to $\beta_j$ and equating to 0. Let $\omega_i$ be the weight

```
1. Set $f_{0k}(x) = 0$, $p_{0k}(x) = 1/2$.
2. For step number $j = 1, \dots, N$
   (a) For class number $k = 1, \dots, p$
       i. Compute $\beta_{jk} r_{jk}(x) = \text{smooth}\left[\frac{y_{ik} - p_{j-1k}(x)}{p_{j-1k}(x)(1 - p_{j-1k}(x))}\right]$
          with weights $p_{j-1k}(x)(1 - p_{j-1k}(x))$.
       ii. Update $f_{jk}(x) = f_{j-1k}(x) + \beta_{jk} r_{jk}(x)$
       iii. Compute $p_{jk}(x) = \frac{e^{f_{jk}(x)}}{1 + e^{f_{jk}(x)}}$
3. Output class$(x) = \arg\max_k(p_{Nk}(x))$
```

**Fig. 3** Pseudocode of backfitting applied to a logistic extended additive model or *Logitboost*. After solving the step (2.a.i) as discussed in the text, the algorithm shares certain similarities with Adaboost. For two classes problems it is not needed the second loop, labelled (a), as $p_{j1}(x) = 1 - p_{j2}(x)$

of the $i$-th example and $z_i$ its corresponding residual. The factor $\beta_j$ is

$$\beta_j = \frac{\sum_i \omega_i z_i r(\mathbf{x}_i; \gamma_j)}{\sum_i \omega_i r^2(\mathbf{x}_i; \gamma_j)} \tag{17}$$

Therefore, we will let the GA to select the combinations of linguistic terms that form the antecedent of the rules, and calculate the importance of the rules by means of the expression 17. The fitness value of a candidate rule is its squared error over the residuals $z_i$, for its optimal value of $\beta_i$. Further details about the genetic algorithm will be given in the next section.

## 4 Logitboost-based learning of fuzzy rule based classifiers

The basic version of learning algorithm proposed here is shown in Fig. 4, and implements the outline depicted in Fig. 3. Our algorithm produces rules with a single consequent in two-class problems, and rules with more than one consequent in multiclass problems. Observe that the antecedent $A^j$ of the fuzzy rule number $j$ plays the role of the regressor $r_j$ in the preceding section, and the confidence degrees $s^j$ are the values $\beta_j$, as given by Eq. 17.

The line "Find $A^j$ ... " implements the *smooth* operation described in the preceding section. The search is solved with a genetic algorithm, that finds the combination of antecedents (the fuzzy set $A$) which, in combination with its optimal value of $s^j$ best approximates the residual, in the weighted least squares sense.

```
$f_{i0k} = 0$
For step number $j = 1, \dots, N$
    For class number $k = 1, \dots, p$
        for $i = 1, \dots, n$ do $p_{ijk} = e^{f_{ij-1k}}/(1 + e^{f_{ij-1k}})$
        for $i = 1, \dots, n$ do $w_{ijk} = p_{ijk}(1 - p_{ijk})$.
        Find $A^j$ that minimizes

$$\text{fitness}(A^j) = \sum_i^n w_{ijk} \left( s^j \cdot A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}} \right)^2$$

$$\text{where } s^j = \frac{\sum_i (y_{ik} - p_{ijk}) A^j(x_i)}{\sum_i w_{ijk} [A^j(x_i)]^2}$$

        for $i = 1, \dots, n$ do $f_{ijk} = f_{ij-1k} + s^j \cdot A^j(x_i)$
        if $s^j > 0$ then Emit the Rule "**if x is** $A^j$ **then t**$(c_k)$**=**$s^j$"
        else Emit the Rule "**if x is** $A^j$ **then t**$(c_1)$**=-**$s^j$ ... **t**$(c_k)$**=0** ... **t**$(c_p)$**=-**$s^j$"
    End for
End for
```

**Fig. 4** Outline of the basic version of backfitting applied to a logistic extended additive model or *Logitboost.* For two classes problems it is not needed the second loop, as $p_{j1}(x) = 1 - p_{j2}(x)$

### 4.1 Genetic search of antecedents

Binary coded genetic algorithms are a natural choice for this problem, and we have experimentally checked that the rules that the GA finds are close to the optimal ones. But the choose of a genetic algorithm is not mandatory for this problem. Many other approaches could be used, including exhaustive search, as the search space is finite and rather small for many practical problems.

#### *4.1.1 Coding scheme*

We will use a coding scheme based in [7]. Let us codify a linguistic term with a '1' bit in a chain of so many bits as different terms in the linguistic partition. For example, let {Low, Med, High} be the linguistic labels of all features in a problem involving three input variables and two classes. The antecedent of the rule

```
For $k = 1, \dots, p$ and $i = 1, \dots, n$ do
$f_{i1k} = -2 + \frac{4}{n} \sum_{\alpha=1}^{n} y_{\alpha k}$
For step number $j = 2, \dots, N$
    For class number $k = 1, \dots, p$
        for $i = 1, \dots, n$ do $p_{ijk} = e^{f_{ij-1k}}/(1 + e^{f_{ij-1k}})$
        for $i = 1, \dots, n$ do $w_{ijk} = p_{ijk}(1 - p_{ijk})$.
        Find $A^j$ that minimizes

$$\text{fitness}(A^j) = \sum_i^n w_{ijk} \left( s_0^j + s^j \cdot A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}} \right)^2$$

$$\text{where } s_0^j = -\frac{\sum_i w_{ijk} A^j(x_i)}{\sum_i w_{ijk}}$$

$$\text{and } s^j = \frac{\sum_i (y_{ik} - p_{ijk})[s_0^j + A^j(x_i)]}{\sum_i w_{ijk}[s_0^j + A^j(x_i)]^2}$$

        for $i = 1, \dots, n$ do $f_{i1k} = f_{i1k} + s_0^j$
        for $i = 1, \dots, n$ do $f_{ijk} = f_{ij-1k} + s^j \cdot A^j(x_i)$
        if $s^j > 0$ then Emit the Rule "**if x is** $A^j$ **then t**$(c_k)$**=**$s^j$"
        else Emit the Rule "**if x is** $A^j$ **then t**$(c_1)$**=-**$s^j$ ... **t**$(c_k)$**=0** ... **t**$(c_p)$**=-**$s^j$"
    End for
End for
For step number $k = 1, \dots, p$
    if $f_{11k} > 0$ then Emit the Rule "**if true then t**$(c_k)$**=**$f_{11k}$"
    else Emit the Rule "**if true then t**$(c_1)$**=**$-f_{11k}$ ... **t**$(c_k)$**=0** ... **t**$(c_p)$**=**$-f_{11k}$"
End for
```

**Fig. 5** Pseudocode of the prefitting version of backfitting applied to a logistic extended additive model or *Logitboost.* For two classes problems it is not needed the second loop, labelled (a), as $p_{j1}(x) = 1 - p_{j2}(x)$

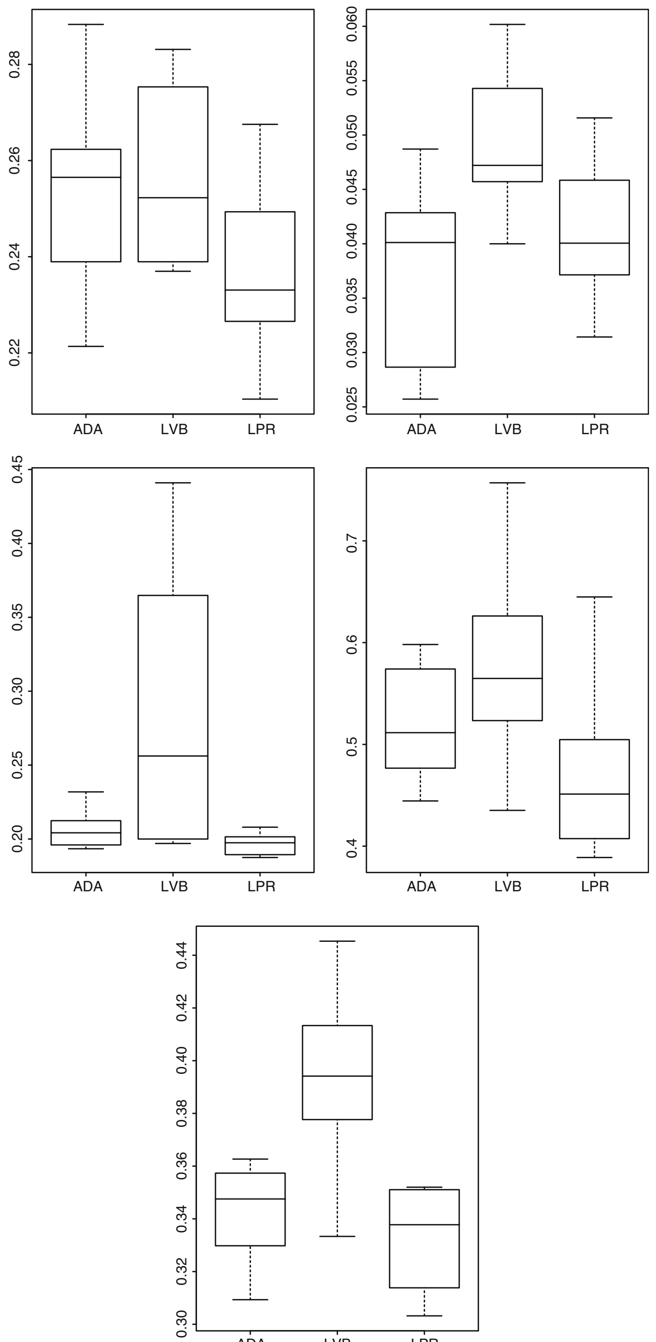

**Fig. 6** The basic implementation of Logitboost (see Fig. 4) does not improve the results of Adaboost. The prefitting based implementation (see Fig. 5) improves significantly the results of this initial implementation. The figures show the results for Pima, Cancer, Gauss, Glass and Gauss-5. Every *boxplot* contains the results for Adaboost, the Basic Version of Logitboost and the Prefitting Version of Logitboost.

$$\texttt{If } x_1 \texttt{ is High and } x_2 \texttt{ is Med and } x_3 \texttt{ is Low}$$
$$\texttt{then class is C1 with seg} = \texttt{S1, C2 with seg} = \texttt{S2}$$

is codified with the chain 001 010 100. We could use this encoding to represent rules for which not all variables appear in the antecedent and 'OR' combinations of terms in the antecedent. For example, the antecedent of the rule

$\texttt{If } x_1 \texttt{ is High and } x_3 \texttt{ is Low then} \ldots$

is codified with the chain 001 000 100, and

$$\texttt{If } x_1 \texttt{ is( High or Med) and } x_3 \texttt{ is Low then } \ldots,$$

will be assigned the chain 011 000 100. With this structure, the GA is also exploited to integrate a rule-wise feature selection process into the search scheme.

Since "OR" combinations of rules increase the complexity of the knowledge base we have decided not to allow them, but we permit rules where not all input variables are present, as we think that this improves the readability of the output.

### 4.1.2 Fitness function

The fitness of a fuzzy rule "if $A^j(x)$ then class(x)=$c_k$ with confidence $s^j$" is the weighted squared error between the logistic transform $p_{ijk}$ of the output of fuzzy classifier and the desired output $y_{ik}$ [that was defined in Eq. (16)]

$$\sum_{i=1}^{n} w_{ijk}\left(s_j \cdot A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}}\right)^2$$

and the expression 17 is rewritten as follows:

$$s_j = \frac{\sum_i (y_{ik} - p_{ijk})A(\mathbf{x}_i)}{\sum_i \omega_i [A^j(\mathbf{x}_i)]^2} \tag{18}$$

## 4.2 Prefitting version

The Logitboost algorithm did not improve the results of the Adaboost algorithm (the experimentation is detailed in Sect. 5.2). This seems to contradict some of the claims made by the Logitboost authors. It can be argued that neither Adaboost nor Logitboost were designed to deal with classifiers that leave most of the feature space uncovered, as is the case with single fuzzy rules. But, since our straight extension of Adaboost was able to properly learn fuzzy rules in classification problems [3] we can conclude that the Logitboost algorithm is more restrictive than Adaboost with respect to the valid families of weak learners.

Interestingly enough, the Logitboost algorithm derives from previous works about backfitting of generalized additive models where it was stated that the expectation of the multivariate regressors $r_j$ must be null [10]. The fuzzy memberships $A^j(\mathbf{x})$, being positive, do not fulfill this requirement.

The simplest modification needed to convert a fuzzy membership into an unbiased weak learner consists in adding it a constant term. This way, the weak learner has not the form $s^j \cdot A^j(\mathbf{x})$, but $s_0^j + s^j \cdot A^j(\mathbf{x})$, where $-s_0^j$ is the weigthed mean of the membership $A^j$ over the feature space. These constant terms can be absorbed into a fuzzy rule that fully covers the feature space, as shown in the algorithm in Fig. 5. Observe that this modification of the Logitboost algorithm can be understood as a partial prefitting, as defined in [23].

## 5 Numerical results

The datasets used in this article to test the accuracy of the proposed algorithm are taken from the UCI Repository Of Machine Learning Databases and Domain Theories [18], from the literature [11] or synthetic [3]. The following datasets are used:

- PIMA (Pima Indians Diabetes Database): two classes problem. The patient shows signs of diabetes according to World Health Organization criteria or not. Eight numerical attributes (related to blood pressure, number of pregnancies, age,...). The number of instances are 768, many attributes have missing values and these have been encoded with the numerical value 0.
- Cancer (Wisconsin Breast cancer): the so called "original dataset" in [18]. Two classes problem, malignant or benign, nine integer attributes (cell size, cell shape, and so on) from 1 to 10, 699 instances.
- Gauss: two classes problem, proposed in [11]. 4000 points taken from two overlapping bi-dimensional Gaussian distributions (centered in (0, 0) and (2, 0))with different covariance matrix ($I$ and $4I$).
- Glass (Glass Identification Database): six class problem, the type of glass. Ten attributes (different oxide content, refractive index), all numerical.
- Gauss-5: synthetic five class problem proposed in [3], comprising 50, 100, 150, 200 and 250 samples from five bi-dimensional Gaussian distribution with centers in (0, 0), $(-1, -1)$, $(-1, 1)$, $(1, -1)$, $(1, 1)$, and unity covariances matrix.

Adaboost and Logitboost were terminated after the generation of seven rules for Pima, four for Cancer, five for Gauss, ten for Glass and 25 for Gauss5. The number of linguistic labels discretizing input variables are 3, 2, 5, 3 and 5, respectively. The genetic algorithm in both descriptive Adaboost and Logitboost is steady-state, with ten subpopulations of size 100 each. Every rule is obtained from the best individual after 2500 crossover operations.

## 5.1 Pairwise comparisons, 5×2cv $f$ test

In order to compare the accuracy of two learning algorithms, Dietterich analyzes in [4] five statistical tests and states that 5×2cv $t$-test has low type I error and good power. Later, in [1], a new test called 5×2cv-f, that improves both type I error and power, is proposed. We have adopted this last test in all our experiments.

This test performs five replications of twofold cross-validation. In each replication, the data set is divided into two sets of equal size. The statistic used in this test is:

$$f = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}[p_i^{(j)}]^2}{2\sum_{i=1}^{5} s_i^2}$$

where $p_i^{(j)}$ is the difference between the error of the two learning algorithms on the $j$-th fold of the $i$-th replication. The $f$ statistic is $F$ distributed with 10 and five degrees of freedom.

**Table 1** $P$-values of the statistical tests that assess the influence of the prefitting term. The influence of the term was significant in all cases but Gauss, where the dispersion of the results prevent us from making a decision

| Pima | Cancer | Gauss | Glass | Gauss5 |
|---|---|---|---|---|
| 0.05 | 0.07 | 0.50 | 0.04 | 0.03 |

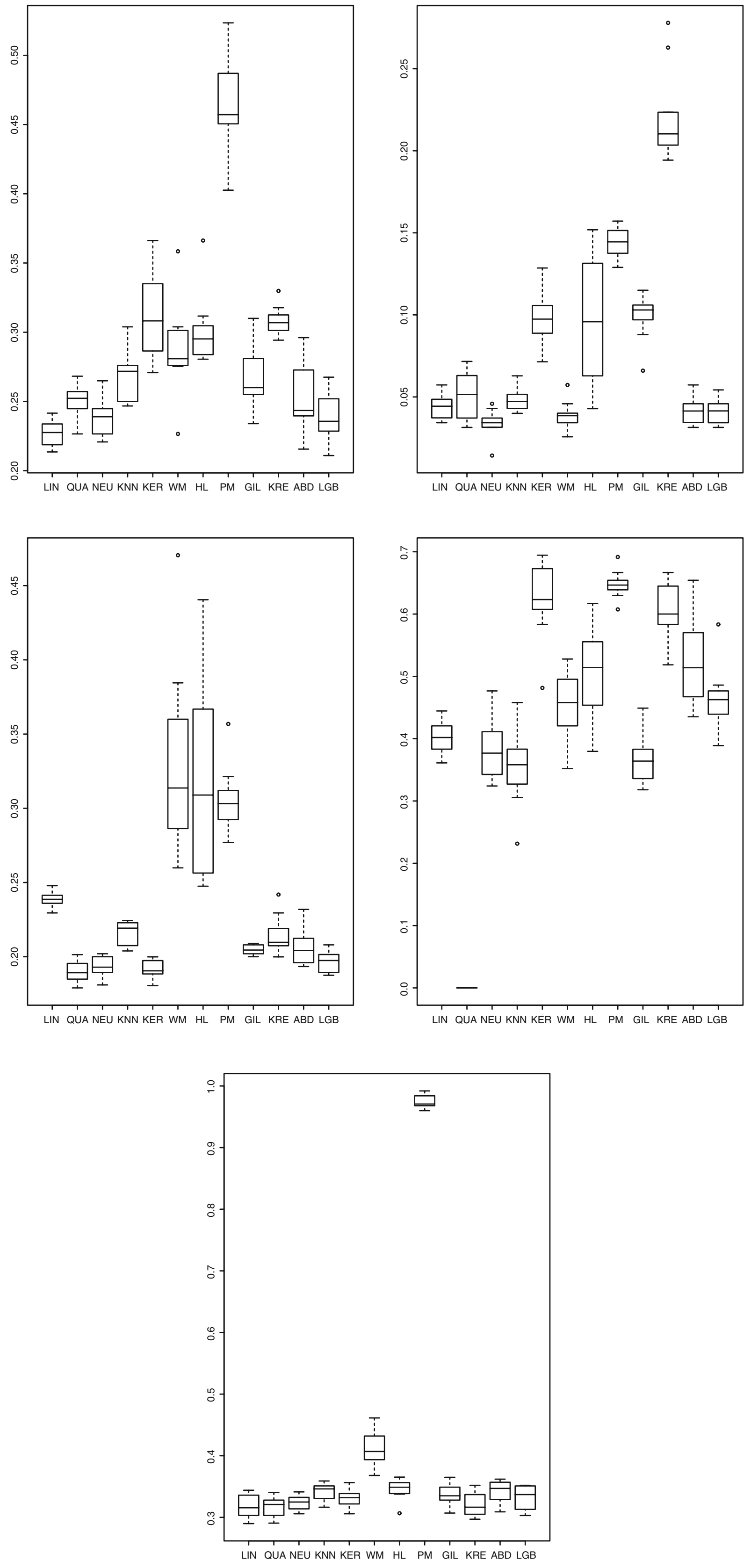

**Fig. 7** *Boxplots* with a comparison between *black-boxes* (linear and quadratic discriminant analysis, three layer perceptron, *k*-nearest neighbours, kernel estimation of densities) and fuzzy rule based classifiers (Wang and Mendel's[24], Ishibuchi[13], Pal and Mandal[19], Genetic Iterative Learning, Random Set based, Fuzzy Adaboost and Fuzzy Logitboost) The datasets are Pima, Cancer, Gauss, Glass and Gauss5

**Table 2** Mean values of the experiments shown in Figure 7

| | LIN | QUA | NEU | KNN | KER | WM | HL | PM | GIL | KRE | ABD | LGB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pima | 0.227 | 0.251 | 0.238 | 0.270 | 0.313 | 0.287 | 0.301 | 0.464 | 0.269 | 0.308 | 0.253 | 0.237 |
| cancer | 0.044 | 0.051 | 0.034 | 0.048 | 0.099 | 0.039 | 0.096 | 0.145 | 0.099 | 0.221 | 0.041 | 0.041 |
| gauss | 0.239 | 0.190 | 0.193 | 0.216 | 0.191 | 0.329 | 0.322 | 0.306 | 0.205 | 0.215 | 0.206 | 0.197 |
| glass | 0.404 | - | 0.382 | 0.354 | 0.621 | 0.453 | 0.503 | 0.647 | 0.363 | 0.606 | 0.522 | 0.463 |
| gauss5 | 0.318 | 0.317 | 0.324 | 0.343 | 0.332 | 0.410 | 0.345 | 0.974 | 0.338 | 0.321 | 0.343 | 0.332 |

**Table 3** $p$-values of the differences between Adaboost and LogitBoost. It is clear from the previous table that Logitbost produces better results than Adaboost, but not all the differences are statistically significant

| Pima | Cancer | Gauss | Glass | Gauss5 |
|---|---|---|---|---|
| 0.07 | 0.28 | 0.19 | 0.10 | 0.16 |

### 5.2 Influence of the prefitting term

The first set of experiments is designed to decide whether the prefitting version of the Logitboost algorithm (see Fig. 5) improves the basic version. In Fig. 6 the results for Pima, Cancer, Gauss, Glass and Gauss-5 are shown. Every boxplot plots the median, the 25 and 75% quartiles and the outliers (points outside 1.5 times the length of the inner quartiles). This way, we graphically describe the tests errors over the ten partitions defined by the 5×2cv-f method, for Adaboost, the Basic Version of Logitboost and the Prefitting Version of Logitboost. The $p$-values of the statistical tests that assess the differences between the two versions of Logitboost are shown in Table 1. The influence of the term was significant in all cases but Gauss, where the dispersion of the results prevent us from making a decision.

### 5.3 Benchmark results

Five statistical methods (linear and quadratic discriminant analysis, neural networks, kernel estimation of densities and $k$-nearest neigbours) plus six fuzzy descriptive rule based methods (Wang and Mendel's [24], Ishibuchi's [13], Pal and Mandal's [19], Iterative Genetic Learning [2], Random Sets Based [21], Fuzzy Descriptive Adaboost [3]) were compared to Logitboost. The combined boxplots are shown in Fig. 7. The Logitboost method was better than fuzzy Adaboost in all datasets, and also the best fuzzy rule learning method in all datasets but one, as shown in Table 2, but the differences between Logitboost and Adaboost were not significant in all cases, as shown in Table 3.

As a reference, the reader can compare the results here with those in [21] and [3] for the same datasets. In these papers it is shown that lower error rates are attainable when using more fuzzy rules.

## 6 Concluding remarks

The Logitboost algorithm has a solid statistical background, that makes it an interesting choice between boosting algorithms. It has been shown that fuzzy rules are not valid weak learners under Logitboost, contrary to Adaboost. A prefitting based algorithm that solves this problem is proposed, and the behaviour of this algorithm is moderately better than our former Adaboost based genetic fuzzy classification system.

The advantages of boosting methods when learning fuzzy classifiers are two: as far as we know, the size of the rule base is between the lowest values attainable with other genetic fuzzy classisiers, and the learning is very fast (between seconds and minutes for the problems used in this paper). But there are also drawbacks: the inference is not standard, and the quality of the rule base is low, because the interaction between rules is very high. In future works, we intend to modify the boosting algorithm in order to produce knowledge rules with higher linguistic quality.

**Acknowledgements** This work was funded by Spanish M. of Science and Technology and by FEDER funds, under the grant TIC-04036-C05-05.

## References

1. Alpaydin E (1999) Combined 5×2cv $F$ test for comparing supervised classification learning algorithms. Neural Comput 11(8):1885–1982
2. Cordón, O, Herrera F (1997)A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples. Int J Approx Reason 17(4):369–407
3. Del Jesus MJ, Hoffmann F, Junco L, Sánchez L Induction of fuzzy rule based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Sets Syst (Admitted for publication)
4. Dietterich G (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural comput 10(7):1895–1924
5. Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: Machine learning, proceedings of the 13th international conference, pp 148–156
6. Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 38(2):337–374
7. Gonzalez A, Perez R (1996) Completeness and consistency conditions for learning fuzzy rules. Fuzzy Sets Syst 96:37–51
8. González A, Herrera F (1997) Multi-stage genetic fuzzy systems based on the iterative rule learning approach. Mathware Soft Comput 4:233–249
9. Hand DJ (1981) Discrimination and classification. Wiley, New York
10. Hastie TJ, Tibshirani R (1986) Generalized additive models. Stat Sci 1:297–318
11. Haykin S (1999) Neural networks. Prentice Hall, Englewood Cliffs
12. Hoffmann F (2001) Boosting a genetic fuzzy classifier. In: Proceeding of joint 9th IFSA world congress and 20th NAFIPS international conference, vol 3. Vancouver, Canada, pp 1564–1569
13. Ishibuchi H (1992) Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy Sets Syst 52:21–32

14. Ishibuchi H, Nakashima T, Morisawa T (1999) Voting in fuzzy rule-based systems for pattern classification problems. Fuzzy Sets Syst 103(2):223–239
15. Junco L, Sanchez L (2000) Using the Adaboost algorithm to induce fuzzy rules in classification problems. In: Proceeding of ESTYLF 2000, Sevilla, pp 297–301
16. Kuncheva LI (2000) Fuzzy Classifier design. Springer, Berlin Heidelberg New York
17. Mallat S, Zhang Z (1993) Matching pursuits with time-frequency dictionaries. IEEE Trans Signal Process 41:3397–3415
18. Merz CJ, Murphy PM (1998) UCI repository of machine learning databases. Available at: http://www.ics.uci.edu/mlearn/MLRepository.html
19. Pal SK, Mandal DP (1992) Linguistic recognition system based in approximate reasoning. Inf Sci 61:135–161
20. Sánchez L (2001) A fast genetic method for inducting linguistically understandable fuzzy models. In: Proceeding of IFSA NAFIPS, pp 1559–1563
21. Sánchez L, Casillas J, Cordón O, del Jesus MJ (2002) Some relationships between fuzzy and random classifiers and models. Int J Approx Reason 29:175–213
22. Schapire R, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336
23. Vincent P, Bengio Y (2002) Kernel matching pursuit, Machine Learning Journal, Special Issue on New Methods for Model Combination and Model Selection, pp 165–187
24. Wang LX, Mendel J (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 25(2):353–361
25. Zhu J, Hastie T (2001) Kernel logistic regression and the import vector machine. In: Proceeding of NIPS 2001, Vancouver, Canada, pp 1081–1088