Boosting Fuzzy Rules in Classification Problems under Single-Winner Inference

Luciano Sánchez,* José Otero†
Universidad de Oviedo, Departmento Informática, Sedes departamentales, Edificio 1, Campus de Viesques, 33203 Gijón, Spain

In previous studies, we have shown that an Adaboost-based fitness can be successfully combined with a Genetic Algorithm to iteratively learn fuzzy rules from examples in classification problems. Unfortunately, some restrictive constraints in the implementation of the logical connectives and the inference method were assumed. Alas, the knowledge bases Adaboost produces are only compatible with an inference based on the maximum sum of votes scheme, and they can only use the t-norm product to model the "and" operator. This design is not optimal in terms of linguistic interpretability. Using the sum to aggregate votes allows many rules to be combined, when the class of an example is being decided. Because it can be difficult to isolate the contribution of individual rules to the knowledge base, fuzzy rules produced by Adaboost may be difficult to understand linguistically. In this point of view, single-winner inference would be a better choice, but it implies dropping some nontrivial hypotheses. In this work we introduce our first results in the search for a boosting-based genetic method able to learn weighted fuzzy rules that are compatible with this last inference method. © 2007 Wiley Periodicals, Inc.

1. INTRODUCTION

The first application of a boosting algorithm to learn fuzzy rules combined a search algorithm with a fitness function taken from Real Adaboost to incrementally learn descriptive fuzzy rules from examples in classification problems.[1] There are subsequent works in which approximate rules are also learned.[2] A comprehensive description of the use of boosting in fuzzy classifiers is given in Ref. 3.

In other publications,[4,5] following the work of Friedman et al.,[6] Adaboost is regarded as a forward stepwise estimation of the statistical parameters defining a logit transform of a Generalized Additive Model, and this property is used to extend this last estimation to learn fuzzy models in regression problems. A similar statistical interpretation was used later to improve the fuzzy Adaboost algorithm, again in classification problems. Adaboost was considered as the application of the same

*Author to whom all correspondence should be addressed: e-mail: luciano@ccia.uniovi.es.
†e-mail: jotero@lsi.uniovi.es.

© 2007 Wiley Periodicals, Inc. Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20236

forward stepwise procedure, a so-called matching pursuit in signal theory related works,[7,8] and an instance of the matching pursuit algorithm was successfully used to extend the LogitBoost algorithm to learn descriptive fuzzy rules in classification problems,[9] solving some difficulties the AdaBoost algorithm poses in multiclass problems.

Although all these methods are fast and produce accurate classifiers and models, they all share a common problem: their output has a low degree of interpretability. This is rooted in their own definition. Fuzzy systems can only be compared to Generalized Additive models when the sum of votes scheme[10] is adopted. But the use of the sum to aggregate rules allows the existence of rules that have no linguistic meaning by themselves, only when combined with other, overlapping ones. In other words, one cannot isolate the contribution of a single rule to the fuzzy classifier; they can be thought of as weights in a neural network.

A better inference method, in terms of linguistic interpretability, is the "single winner."[11] This last mechanism is compatible with the idea of a fuzzy rule being an imprecise assertion, which states that all patterns in a given fuzzy region belong to the same class. But the single-winner inference does not combine the votes of the rules with the arithmetic sum, but with the maximum operator. Apparently, this prevents us from using the analogy between fuzzy classifiers and additive models on which fuzzy Adaboost depends. Conversely, we will show later in this article that this problem can be reformulated so that a matching pursuit, with a prefitting stage, can be used to solve it.

The structure of this article is as follows: in the next section, fuzzy classifiers are introduced and we explain how boosting can be applied to induce them from data. Then, we explain how single-winner inference can be expressed in terms of additive models, and a new algorithm is proposed. We finish with an empirical evaluation of the new algorithm and some preliminary numerical results.

2. BOOSTING FUZZY CLASSIFIERS

2.1. Notation

At this point we introduce the basic notation employed throughout the article. 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 $A^j \subset \tilde{\mathcal{P}}(\mathbf{X})$, where $\tilde{\mathcal{P}}(\mathbf{X})$ stands for "fuzzy parts of $\mathbf{X}$." In the remaining part of this article, 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.

2.1.1. Linguistic Interpretation of Fuzzy Classifiers

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 may have up to p numbers between 0 and 1 that represent our degree of knowledge about the assertion "All elements in the fuzzy set A^j belong to class number k." Values near 1 mean "high confidence" and values near 0 mean "absence of knowledge about the assertion."

In practical cases, we work with antecedents 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

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

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. This work deals with descriptive classifiers.

2.1.2. Fuzzy Inference

Fuzzy reasoning methods define how rules are combined and how to infer from a given input to the corresponding output.

An instance $\mathbf{x}$ is assigned to the class

$$\arg\max_k \bigvee_j A^j(\mathbf{x}) \wedge s_k^j \tag{1}$$

where "$\wedge$" and "$\vee$" can be implemented by different operators. "$\wedge$" is always a t-norm, usually the minimum or the product. In this work, we have chosen to use the product.

Selecting an implementation of the "$\vee$" operator is not immediate. Fuzzy Adaboost relies on the use of the "maximum voting scheme,"[11] because the assignment of confidence degrees to fuzzy rules derived from Adaboost is only meaningful when the outputs of all rules are added to produce the output of the classifier (see Ref. 3). It was mentioned in the Introduction that this scheme may be criticized because of interpretability reasons. Consequently, in this article we are not interested in defining "$\vee$" by means of the sum operation but instead use the maximum operator.[10] Observe that, in this case, all terms s_k^j in rule number j except the maximum one can be removed without affecting the output of the classifier, which will be formed by rules as follows:

$$\text{if } x_1 \text{ is } A_1^j \text{ and } \ldots \text{ and } x_n \text{ is } A_n^j$$
$$\text{then truth}(c_{q(j)}) = s_{q(j)}^j$$

where $q(j) = \arg\max_k s_k^j$. There is an alternate linguistic expression for this rule:

$$\text{if } x_1 \text{ is } A_1^j \text{ and } \ldots \text{ and } x_n \text{ is } A_n^j$$
$$\text{then class} = q(j) \text{ with weight } [s_{q(j)}^j]$$

2.2. Learning Fuzzy Rules

The learning algorithm that will be introduced in this article is based on the similarities that exist between estimating additive models and boosting rules.[6,9] In the following sections, the concepts needed to understand these similarities are summarized.

The statistical problem solved when learning a classifier is "estimate $P(\text{class}(\mathbf{x}) = c_k)$." When additive models are used to solve it, it is reformulated by means of p random variables

$$y_k(\mathbf{x}) = \begin{cases} 1 & \text{if class}(\mathbf{x}) = c_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

that allow us to transform the classification problem into a regression one, that of estimating the conditional expectations $E(y_k|\mathbf{x}) = P(\text{class}(\mathbf{x}) = c_k)$, which is solved as shown in the next subsection.

2.2.1. Additive Models

Additive models were introduced in the 1980s 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.

Let $E(y|\mathbf{x}) = p(\text{class}(\mathbf{x}) = 1)$ be the output random variable we wish to model, and let $\mathbf{x} = (x_1, \ldots, x_n)$ be the input random vector. The objective of the modeling process consists of estimating the conditional expectation of y given $\mathbf{x}$.

Linear regression assumes

$$E(y|\mathbf{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{3}$$

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

$$E(y|\mathbf{x}) = u_0 + u_1(x_1) + \cdots + u_n(x_n) \tag{4}$$

where u_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 and making the additive component to depend on the mean of the output by means of a link function g, so that

$$g(E(y|\mathbf{x})) = u_0 + u_1(x_1) + \cdots + u_n(x_n) \tag{5}$$

Additive models can be generalized furthermore. In extended additive models, the n univariate regressors u_i are replaced by N functions r_j of more than one feature. In our context, these functions usually depend on a set of parameters γ and a multiplier β,

$$r_j = \beta_j r(\mathbf{x}; \gamma_j) \tag{6}$$

Thus the additive model becomes

$$g(E(y|\mathbf{x})) = r_0 + \sum_{j=1}^{N} \beta_j r(\mathbf{x}; \gamma_j) \tag{7}$$

Generalized additive models embody many machine-learning algorithms. For example, in radial basis neural networks the functions $r(x, \gamma_j) = \exp\{\|x - \gamma_j\|^2\}$ are the "basis functions," γ_j are their centers, and β_j are the weights that connect the input layer with the output. In support vector machines, $r(x, \gamma_j)$ is a kernel and γ_j are the support vectors. In our case, we will propose a model where $r(x, \gamma_j)$ is an expression that contains the membership A^j of the antecedent of the jth fuzzy rule, γ_j identifies the linguistic terms that participate in the rule, and β_j is the weight of the rule (that we have called $s^j_{q(j)}$ before). The precise expression will be made clear later.

2.2.2. *Backfitting and the Logitboost Algorithm*

Extended additive models are usually estimated by maximum likelihood. The numerical techniques involved can be very different, but they all share a common objective: given a cost function d that measures the differences between the conditional expectation and its approximation, the learning algorithm consists of finding N pairs of values $\{\beta_j, \gamma_j\}$ minimizing each

$$E\left[d\left(y, \sum_{\substack{\alpha=1\ldots N \\ j\neq\alpha}} \beta_\alpha r(\mathbf{x}; \gamma_\alpha) + \beta r(\mathbf{x}; \gamma)\right)\right] \tag{8}$$

with respect to β, γ.[6] We are interested in a greedy learning algorithm that finds $\{\beta_1, \gamma_1\}$ first, then $\{\beta_2, \gamma_2\}$, and so on.

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.[7] These algorithms have been compared to those used to learn support vector machines[12] and radial basis neural networks in machine-learning problems,[8] and also to Genetic Iterative Learning of fuzzy rules.[3]

One of the most interesting properties of matching pursuit algorithms is that they are good in keeping the sparsity of the solution; this explains the good generalization properties of the methods listed before. 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.

As we mentioned in the preceding section, the objective of a binary classification problem is to approximate the value $E(y|\mathbf{x}) = p(\text{class}(\mathbf{x}) = 1)$, which we will abbreviate by $p(\mathbf{x})$. The response variable in a classification problem follows the binomial distribution, whose corresponding extended additive model is[13]

$$\log \frac{p(\text{class}(\mathbf{x}) = 1)}{p(\text{class}(\mathbf{x}) = 0)} = g(E(y|\mathbf{x})) = r_0 + \beta_1 r(\mathbf{x}, \gamma_1) + \cdots \tag{9}$$

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

$$p(x) = \frac{e^{g(E(y|\mathbf{x}))}}{1 + e^{g(E(y|\mathbf{x}))}} \tag{10}$$

If the greedy version of generalized backfitting (the "matching pursuit" algorithm), also mentioned in the preceding subsection, is applied to this model, the Logitboost algorithm is obtained.[6] At each iteration, an adjusted dependent variable is fitted by weighted least squares. The adjusted variable that arises from the binomial distribution is

$$z = g(E(y|\mathbf{x})_0) + \frac{1_{y=1} - p(\mathbf{x})}{p(\mathbf{x})(1 - p(\mathbf{x}))} \tag{11}$$

This response depends on the values $p(\mathbf{x})$, defined by means of Equation (10). $g(E(y|\mathbf{x})_0)$ is the output of the additive model in the previous iteration, as defined in Equation (9). When the method is particularized to learn fuzzy rules, $g(E(y|\mathbf{x})_0)$ is the output of the fuzzy rule base before the new rule is added, and the values s_k^j and the membership function A^j are chosen to minimize

$$\text{fitness}(A^j) = \sum_i^n p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))\left(s_k^j \cdot A^j(\mathbf{x}_i) - \frac{d(\mathbf{x}_i) - p(\mathbf{x}_i)}{p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))}\right)^2 \tag{12}$$

and are searched by means of a genetic algorithm. An outline of the adaptation of this method to learn fuzzy rules, as described in Ref. 9, is shown in Figure 1.

3. PROPOSED ALGORITHM

3.1. Definition of the Weak Learners

Let us recall Equation (1). We mentioned that an instance $\mathbf{x}$ is assigned to the class

$$\arg\max_k \bigvee_j A^j(\mathbf{x}) \wedge s_k^j \tag{13}$$

```
$f_{i0k} = 0$
For step number $j = 1, \ldots, N$
    For class number $k = 1, \ldots, p$
        for $i = 1, \ldots, n$ do $p_{ijk} = e^{f_{ij-1k}}/(1 + e^{f_{ij-1k}})$
        for $i = 1, \ldots, n$ do $w_{ijk} = p_{ijk}(1 - p_{ijk})$.
        Find with a Genetic Algorithm the antecent $A^j$ 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 $i = 1, \ldots, n$ 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
```

Figure 1. Outline of the basic version of backfitting applied to a logistic extended additive model or *Logitboost.* For two classes problems the second loop is not needed, as $p_{j1}(x) = 1 - p_{j2}(x)$. The knowledge base that this algorithm produces requires the "sum of votes" inference.

where "$\wedge$" and "$\vee$" could be implemented by different operators. In fuzzy boosting, this last expression became

$$\arg\max_k \sum_j A^j(\mathbf{x}) \cdot s_k^j \tag{14}$$

and that allowed us to use the fuzzy memberships A^j in antecedents as weak learners and obtain the weights of the rules s_k^j by means of a boosting algorithm, as shown in Figure 1.

To use single-winner inference, we want to use the max operator instead of the sum. We have to obtain the weights s_k^j of the expression that follows:

$$\arg\max_k \left\{ \max_j A^j(\mathbf{x}) \cdot s_k^j \right\} \tag{15}$$

which is not a sum of terms and therefore not an additive model; thus boosting makes no sense here. But, we can define a function

$$I(\mathbf{x}, j) = \begin{cases} 1 & \text{if } j = \arg\max A^j(\mathbf{x}) \cdot s_k^j \\ 0 & \text{elsewhere} \end{cases} \tag{16}$$

(in words, $I(\mathbf{x}, j) = 1$ if the rule number j is the winner when classifying the input $\mathbf{x}$, and 0 if not) and rewrite Equation (15) as

$$\arg\max_k \sum_j I(\mathbf{x},j)\cdot A^j(\mathbf{x})\cdot s_k^j \tag{17}$$

The rewritten expression is related to Equation (7) as follows. Observe that the products

$$r(\mathbf{x},\gamma_j) = I(\mathbf{x},j)\cdot A^j(\mathbf{x}) \tag{18}$$

can be regarded as weak learners, and their weights s_k^j determined by backfitting. Therefore, we have transformed the initial problem into another that can be solved after estimating the function *I*. In the next subsection, we propose an iterative algorithm to complete the task.

3.2. Recurrent Estimation of the Function I

Let us suppose for the time being that we have an initial rule base, comprising $N-1$ rules, to which we want to add a new fuzzy rule. The additive model we want to solve is

$$g(E(y_k|\mathbf{x})) = \sum_j I(\mathbf{x},j)\cdot A^j(\mathbf{x})\cdot s_k^j \tag{19}$$

We know the values of $I(\mathbf{x}_i,j)$ for all the rules in the initial base, at the points in the training set $I(\mathbf{x}_i,j)=1$ if the rule number *j* was the winner in the example *i*, 0 otherwise. Now we want to improve the base by adding the *N*th rule. It is clear that some of these values of *I* will change from 1 to 0 after the new rule is added (otherwise, the new rule would not win at any example in the training set and it would be useless to add it to the base). But *I* participates in the definition of the weak learners; therefore all values s_k^j must be recalculated every time a rule is added. Clearly, a prefitting version of the matching pursuit algorithm[8] is mandatory for this problem: the consequents of all the rules in the initial base are affected after a new rule is added to it.

If all the values $I(\mathbf{x}_i,j)$ and $A^j(\mathbf{x}_i)$ were known, the least squares election of s_k^j would be a standard problem of linear regression that can be solved by means of a pseudoinverse. Let us suppose that the initial rule base is empty and define $f_{i0k}=0$, $p_{i0k}=1/2$, and $d_{ik}=1$ if $\text{class}(x_i)=k$ and 0 otherwise. The adjusted dependent variable is

$$z_{ik} = 0 + \frac{d_{ik}-1/2}{1/2(1-1/2)} \tag{20}$$

and, given *a set of N membership functions* A^j, the $N\cdot p$ values s_k^j can be found by minimizing

$$\text{fitness}(A^1,\ldots,A^N) = \left(z_{ik} - \sum_j I(\mathbf{x}_i,j)\cdot A^j(\mathbf{x}_i)\cdot s_k^j\right)^2 \tag{21}$$

In matrix form, let $S=[\sigma_{kj}]$, $Z=[\theta_{ki}]$, and $F=[\phi_{ji}]$, where $\sigma_{kj}=s_k^j$, $\theta_{ki}=z_{ik}$, and $\phi_{ji}=f_{i1k}$. Then, the least squares solution of *S* is

```
procedure **AddOneRule**
**Input:** A rule base of size $N$ and the antecedent of the fuzzy rule $A^{N+1}$
**Output:** A rule base of size $N+1$ and a numerical value of fitness
 $S_{kj} = s_k^j \quad k = 1, \ldots, p,\ j = 1, \ldots, N$
 Initialize $S_{k,N+1}$ at random
 Repeat
    $I' = I$
    For $j = 1, \ldots, N+1,\ i = 1, \ldots, m$ do
        $I_{ij} = \begin{cases} 1 & \text{rule } j \text{ wins in example } x_i \\ 0 & \text{else} \end{cases}$
    End For
    $F_{ji} = A^j(\mathbf{x}_i) \cdot I_{ij} \quad j = 1, \ldots, N+1,\ i = 1, \ldots, m$
    $Z_{ki} = 4(y_k(\mathbf{x}_i) - 0.5) \quad k = 1, \ldots, p,\ i = 1, \ldots, m$
    $S' = S$
    $S = Z \cdot F^t \cdot (F \cdot F^t)^{-1}$
    $S = \alpha S + (1 - \alpha) S'$
 Until $||S - S'|| < \epsilon$
 Output $S$ and fitness$=||Z - S \cdot F||$
```

Figure 2. The procedure AddOneRule takes as inputs a fuzzy classifier of size *N* and the antecedent of a fuzzy rule. Its output consists of a new fuzzy classifier, of size $N + 1$, and a numerical fitness value that measures the merit of the new rule. Adding one rule to the base implies recalculating the importance of all consequents.

$$S = ZF^t(F \cdot F^t)^{-1} \tag{22}$$

We can use a search algorithm (a genetic algorithm, in this case) to find the set of values of A^j and I that minimize Equation (21), but the simultaneous search of A^j and I would not be efficient. We propose to use instead the recurrent algorithm shown in Figure 2 to do the task. In the next section we will explain where this function is called from and give an outline of the complete procedure.

Notice that the values of s_k^j are randomly generated first to obtain an initial guess of I; then s_k^j are calculated by means of Equation (22). I is estimated again; if old and new values are the same, the procedure stops. Otherwise, the process is repeated. In practical simulations we have observed that a smoothing term α, as shown in Figure 2, improves the speed of convergence.

3.3. Scheme of the Algorithm

An outline of the algorithm is shown in Figure 3. Fitness values computed by the function "AddOneRule" are optimized by a Genetic Algorithm, which is launched once every time a new rule is added to the base. The algorithm is incremental, because antecedents of rules do not change between iterations, but their weights can be modified by the mentioned function.

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. We use a coding scheme based on Ref. 14. Let us codify a linguistic

```
For step number $j = 1, \ldots, N$
    Call **AddOneRule** from a GA and select
        the rule base of size $j$ with the minimum fitness value.
End For
For $j = 1, \ldots, N$
    Make all $s_k^j = 0$ but the maximum one, $s_{q(j)}^j$
    Emit the Rule "**if x is** $A^j$ **then class** $= q(j)$ **with confidence** $[s_{q(j)}^j]$"
End For
```

Figure 3. Outline of the backfitting algorithm applied to a logistic extended additive model under single-winner inference, or *Max-Fuzzy Logitboost*.

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

$$x_1 \texttt{ is High and } x_2 \texttt{ is Med and } x_3 \texttt{ is Low}$$

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

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

is codified with the chain 001 000 100, and

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

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.

4. PRELIMINARY BENCHMARK RESULTS

The data sets 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,[15] from the literature,[16] or are synthetic.[3] The following data sets 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, etc.). 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 data set in Ref. 15. Two classes problem, malignant or benign, nine integer attributes (cell size, cell shape, etc.) from 1 to 10, 699 instances.

- Gauss: two classes problem, proposed in Ref. 16. Four thousand points taken from two overlapping bidimensional Gaussian distributions (centered in $(0,0)$ and $(2,0)$) with different covariance matrix (I and $4I$).
- Glass (Glass Identification Database): six classes problem, the type of glass. Ten attributes (different oxide content, refractive index), all numerical.
- Gauss-5: synthetic five classes problem proposed in Ref. 3, comprising 50, 100, 150, 200, and 250 samples from five bidimensional Gaussian distributions with centers in $(0,0)$, $(-1,-1)$, $(-1,1)$, $(1,-1)$, $(1,1)$, and unity covariances matrix.

Four statistical methods were evaluated: linear (LIN) and quadratic discriminant analysis (QUA), kernel estimation of densities (KER), and *k*-nearest neighbors (KNN). In addition, neural networks (NEU) and six fuzzy descriptive rule-based methods were compared to Max-Fuzzy Logitboost (AMM). These methods are Wang and Mendel's (WM),[17] Ishibuchi's (ISH),[18] Pal and Mandal's (PM),[19] Iterative Genetic Learning (GIL),[20] Random Sets Based (KRE),[21] and Fuzzy Descriptive Adaboost (ADB).[3]

To compare the accuracy of two learning algorithms, Dietterich[22] analyzed five statistical tests and stated that 5x2cv t-test has low type I error and good power. In subsequent works, Alpaydin[23] proposed a new test called 5x2cv-f that improves both type I error and power. We have adopted this last test in all our experiments. We have also decided to give a graphical view of the relevance of the differences between the different methods by means of boxplots (see Figure 4). Every boxplot summarizes (median, interquartile range, and outliers) the 10 test results produced by the 5x2cv-f experimental design and allow us to judge the significance of the differences in the mean error rates of the different algorithms being compared.

The genetic algorithm mentioned in Figure 3 is of the steady-state type, with 10 subpopulations of size 100, each evolving in parallel. The value of the smoothing parameter α in Figure 2 (see Section 3.2) is 0.75 for all experiments. Every rule is obtained from the best individual after 5000 evaluations of the objective function. Special care was taken to select the minimum number of rules that produce a meaningful classification for all data sets in order to keep the rule bases linguistically understandable. We decided to stop the learning in either Adaboost and Max-Fuzzy Logitboost when knowledge bases are of the following sizes:

- 7 rules for Pima (3 linguistic labels in all input variables)
- 4 rules for Cancer (2 labels)
- 5 rules for Gauss (5 labels)
- 10 rules for Glass (3 labels)
- 10 rules for Gauss5 (3 labels).

Observe that the numerical values of the error could be further lowered if the number of rules had been allowed to increase. As a reference, the reader can compare the results here with those in Refs. 21 and 3 for the same data sets.

After examining the boxplots in Figure 4 (see also the values in Table I and the rule base in Figure 5), we can conclude that the extra linguistic quality has little cost in accuracy; differences are never statistically significant (after applying the 5x2cv-f test, with 95% confidence), and, in fact, this method improves Fuzzy Adaboost in multiclass problems, as expected (this is one of the advantages of

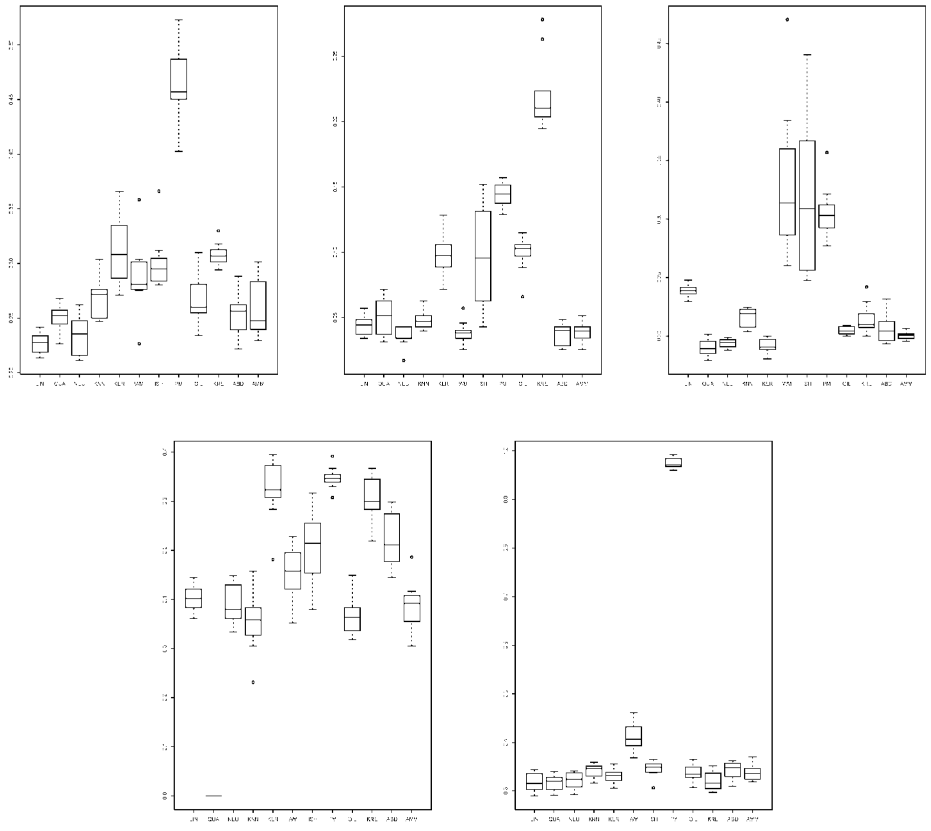

Figure 4. Boxplots with a comparison between linear and quadratic discriminant analysis, three-layer perceptron, *k*-nearest neighbors, kernel estimation of densities, and fuzzy rule-based classifiers (Wang and Mendel's, Ishibuchi, Pal and Mandal, Genetic Iterative Learning, Random Set based, Fuzzy Adaboost, and Max-Fuzzy Logitboost). The three graphics in the upper part are, from left to right, Pima, Cancer, and Gauss. The lower part shows the results of Glass and Gauss5.

Logitboost over Adaboost, according to its author[6]). Therefore, Max-Fuzzy Logitboost is the preferred learning algorithm when linguistic quality is the primary concern. Unfortunately, the learning time of the proposed method is much longer that that needed by fuzzy Adaboost (between 6 and 10 times, in our implementation) and ranks between 15 and 30 min on a personal computer (Intel Pentium III, 500 MHz) for each problem contained in the proposed benchmark.

5. CONCLUDING REMARKS

The advantages of boosting methods when learning fuzzy classifiers are two: as far as we know, the size of the rule base can be made very small and the learning is very fast (between seconds and minutes for the problems used in this article).

Table I. Mean values of the experiments shown in Figure 4.

	LIN	QUA	NEU	KNN	KER	WM	ISH	PM	GIL	KRE	ABD	AMM
Pima	0.227	0.251	0.234	0.270	0.313	0.287	0.301	0.464	0.269	0.308	0.255	0.257
Cancer	0.044	0.051	0.035	0.048	0.099	0.039	0.096	0.145	0.099	0.221	0.038	0.039
Gauss	0.239	0.190	0.194	0.216	0.191	0.329	0.322	0.306	0.205	0.215	0.206	0.200
Glass	0.404	—	0.389	0.354	0.621	0.453	0.503	0.647	0.363	0.606	0.522	0.388
Gauss5	0.318	0.317	0.321	0.343	0.332	0.410	0.345	0.974	0.338	0.321	0.344	0.337

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.

The high interaction between rules in Adaboost and Logitboost is a consequence of the "sum of votes" inference scheme. The preferred inference method, in terms of linguistic interpretability, is the "single winner" one. This last mechanism is compatible with the idea of a fuzzy rule being an imprecise assertion, which states that all patterns in a given fuzzy region belong to the same class. But the single-winner inference does not combine the votes of the rules with the arithmetic sum, but the maximum operator, and this makes the application of boosting algorithms difficult. We have solved the problem by the introduction of an intermediate function in the definition of the weak learner and a recurrent algorithm to estimate it. The final algorithm produces fuzzy rule bases of high descriptive quality while preserving a good accuracy. A drawback of this new procedure is related to its computational complexity, which is higher than that of fuzzy Adaboost and Logitboost.

Acknowledgments

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

```
if x1 is LOW and x2 is LOW and x3 is LOW and
   x5 is LOW and x6 is LOW and x7 is LOW and
   x8 is LOW then class=1 with weight 0.69

if x6 is HIGH then class=2 with weight 0.51

if x1 is HIGH and x3 is HIGH and x6 is LOW and
   x7 is HIGH then class=2 with weight 0.97

if x1 is LOW and x2 is HIGH and x4 is LOW and
   x5 is LOW and x7 is LOW and x8 is LOW
   then class=1 with weight 1
```

Figure 5. Example of rule base obtained by means of Max-Fuzzy Logitboost for the problem "Cancer," mentioned in the text.

References

1. Junco L, Sanchez L. Using the Adaboost algorithm to induce fuzzy rules in classification problems. In: Proc X Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF), Sevilla, Spain; 2000. pp 297–301.
2. Hoffmann F. Boosting a genetic fuzzy classifier. In: Proc Joint Ninth IFSA World Congress and 20th NAFIPS Int Conf; 2001. pp 1564–1569.
3. Del Jesus MJ, Hoffmann F, Junco L, Sánchez L. Induction of fuzzy rule based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Syst 2004;12:3.
4. Sánchez L. A fast genetic method for inducting linguistically understandable fuzzy models. In: Joint 9th Int Fuzzy Systems Association World Congress and 20th North American Fuzzy Information Processing Society Int Conf (IFSA/NAFIPS), Vancouver, Canada; 2001. pp 1559–1563.
5. Sánchez L, Otero J. A fast genetic method for inducting descriptive fuzzy models. Fuzzy Set Syst 2004;141:33–46.
6. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: A statistical view of boosting. Ann Stat 2000;38:337–374.
7. Mallat S, Zhang Z. Matching pursuits with time-frequency dictionaries. IEEE Trans Signal Process 1993;41:3397–3415.
8. Vincent P, Bengio Y. Kernel matching pursuit. Mach Learn 2002;48:165–187.
9. Otero J, Sánchez L. Induction of descriptive fuzzy classifiers with the Logitboost algorithm. Soft Comput 2006;10:825–835.
10. Kuncheva LI. Fuzzy classifier design. New York: Springer-Verlag; 2000.
11. Ishibuchi H, Nakashima T, Morisawa T. Voting in fuzzy rule-based systems for pattern classification problems. Fuzzy Set Syst 1999;103:223–239.
12. Zhu J, Hastie T. Kernel logistic regression and the import vector machine. In: Proc Neural Information Processing Systems; 2001. pp 1081–1088.
13. Hastie TJ, Tibshirani R. Generalized additive models. Stat Sci 1986;1:297–318.
14. Gonzalez A, Perez R. Completeness and consistency conditions for learning fuzzy rules. Fuzzy Set Syst 1996;96:37–51.
15. Merz CJ, Murphy PM. UCI repository of machine learning databases. Available at: http://www.ics.uci.edu/mlearn/MLRepository.html; 1998.
16. Haykin S. Neural networks. Englewood Cliffs, NJ: Prentice Hall; 1999.
17. Wang LX, Mendel J. Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 1992;25:353–361.
18. Ishibuchi H. Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy Set Syst 1992;52:21–32.
19. Pal SK, Mandal DP. Linguistic recognition system based in approximate reasoning. Inform Sci 1992;61:135–161.
20. Cordón O, Del Jesus MJ, Herrera F. A proposal on reasoning methods in fuzzy rule-based classification systems. Int J Approx Reason 1999;20:21–45.
21. Sánchez L, Casillas J, Cordón O, del Jesus MJ. Some relationships between fuzzy and random classifiers and models. Int J Approx Reason 2002;29:175–213.
22. Dietterich G. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput 1998;10:1895–1924.
23. Alpaydin E. Combined 5x2cv F test for comparing supervised classification learning algorithms. Neural Comput 1999;11:1885–1982.