Preliminary results on the application of boosting to learn weighted fuzzy rules under single-winner inference

Luciano Sánchez, José Otero, M. del Rosario Suárez.
Universidad de Oviedo. Depto. Informática.
Sedes departamentales, Edificio 1.
Campus de Viesques, 33203 Gijón.

Resumen— **In previous works, we have shown that Adaboost and Logitboost can be used to learn fuzzy rules from examples in classification problems. Boosting-based algorithms are significantly faster than other genetic fuzzy systems, and they produce compact rule bases with good accuracy. Unfortunately, both fuzzy Adaboost and fuzzy Logitboost exploit certain similarities between generalized additive models and fuzzy classifiers. As a consequence of its definition, the knowledge bases they produce are only compatible with an inference based on the maximum sum of votes scheme. This procedure is not standard in fuzzy classifiers, thus rule bases produced by fuzzy boosting have a low degree of interpretability.**

In this work we try to avoid this problem and propose a boosting-like algorithm able to learn weigthed fuzzy rules that are compatible with a single-winner method.

I. Introduction

The first application of a boosting algorithm to learn fuzzy classifiers is given in [16]. In this work, it was proposed to combine a search algorithm with a fitness function taken from Real Adaboost to incrementally learn descriptive fuzzy rules from examples in classification problems. There are subsequent works in which approximate rules [12] are also learned. A comprehensive description of the use of boosting in fuzzy classifiers is given in [4].

In [22][24], following the work of [7], Adaboost is regarded as an 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 has been used later to improve the fuzzy Adaboost algorithm, again in classification problems. Adaboost was considered as the application of the same forward stepwise procedure, so called "Matching Pursuit" in signal theory related works [18][26], and an instance of the matching pursuit algorithm was successfully used to extend the LogitBoost algorithm to learn descriptive fuzzy rules in classification problems [20], solving some difficulties the AdaBoost algorithm poses in multiclass problems.

Besides 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 [17] is adopted. But the use of the sum to aggregate rules allows the existence of rules that have not linguistical meaning by themselves, but when combined with other, overlapping ones. In other words, one can not isolate the contribution of a single rule to the fuzzy classifier; they can be thought of as weights in a neural network.

The prefered inference method, in terms of linguistic interpretability, is called "single winner" [14]. This last mechanism is compatible with the idea of a fuzzy rule being an imprecise assert, which states that all patterns in a given fuzzy region belong to the same class. But the single winner inference does not combines the votes of the rules with the arithmetic sum, but the maximum operator. Apparently, this leaves out the analogy between fuzzy classifiers and additive models that originated fuzzy Adaboost. We will show later in this paper that this is not always true, and that this problem can be solved by means of a matching pursuit, with a prefitting stage, that shares a common structure with that used in fuzzy Logitboost.

A. Summary

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, it is explained how single winner inference can be casted in terms of additive models and a new algorithm proposed. The paper finishes with an empirical evaluation of the new algorithm and some preliminary numerical results.

II. Boosting Fuzzy Classifiers

A. 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 $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{X}$. 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 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.

B. 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 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 assert."

In practical cases, we work with antecedents A^j that can be decomposed in a Cartesian product of fuzzy sets defined over each feature, $A^j = A^j_1 \times A^j_2 \times \ldots \times A^j_n$, thus the rules are

$$\begin{gathered}\text{if } x_1 \text{ is } A^j_1 \text{ and } \ldots \text{ and } x_n \text{ is } A^j_n \\ \text{then truth}(c_1) = s^j_1 \text{ and } \cdots \text{ and truth}(c_p) = s^j_p.\end{gathered}$$

We can restrict the definition further by defining n linguistic variables (one linguistic variable for every feature) and requiring that all terms sets A^j_f 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.

C. 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

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

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" [15] (because of reasons explained in [4].) It was mentioned in the introduction that this scheme may be criticized, because of interpretability reasons. Therefore, in this paper define "$\vee$" to be the maximum operator [17] and study the consequences of this definition when boosting fuzzy rules.

D. Generalized Additive Models, Backfitting and Logitboost

According to [7], Adaboost can be assimilated to a statistical inference problem. The fuzzy Logitboost algorithm [20] exploits this idea and learns fuzzy rules by fitting a logit transform of a Generalized Additive Linear Model to data by means of a matching pursuit algorithm.

The original statistical problem being solved is "estimate $P(\text{class}(\mathbf{x}) = c_k)$." Alternatively, we define p random variables

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

and reformulate the classification problem as a regression problem, 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.

E. 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.

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

$$E(y|x) = f(x_1, \ldots, x_n) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n \tag{3}$$

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

$$E(y|x) = f(x_1, \ldots, x_n) = r_0 + r_1(x_1) + \ldots + r_n(x_n) \tag{4}$$

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,

$$f_y(t; \theta; \phi) = \exp\left\{\frac{t\theta - b(\theta)}{a(\phi)} + c(t, \phi)\right\} \tag{5}$$

and making the additive component

$$f(x_1, \ldots, x_n) = r_0 + r_1(x_1) + \ldots + r_n(x_n) \tag{6}$$

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 γ and a multiplier β,

$$r_j = \beta_j r(x; \gamma_j) \tag{7}$$

thus the additive model becomes

$$E(y|x) = f(x_1, \ldots, x_n) = r_0 + \sum_{j=1}^{N} \beta_j r((x_1, \ldots, x_n); \gamma_j). \tag{8}$$

For example, in radial basis neural networks the functions $s(x, \gamma_j) = \exp\{||x - \gamma_j||^2\}$ 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, $r(x, \gamma)$ is a kernel, and γ_j are the support vectors. In our case, we will propose a model where (x, γ_j) is an expression that contains the membership A^j of the antecedent of the i-th fuzzy rule, γ_j identifies the linguistic terms that participate in the rule and β_j is the degree of truth of the consequent of the rule.

F. Backfitting and the Logitboost Algorithm

Extended additive models can be learned with a generalized backfitting algorithm [7]. 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

$$E\left[d\left(y, \sum_{\substack{\alpha=1\ldots N \\ j\neq\alpha}} \beta_\alpha r(x; \gamma_\alpha) + \beta r(x; \gamma) \right) \right] \tag{9}$$

with respect to β, γ [7].

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 [18]. These algorithms have been compared to support vector machines [28] and radial basis neural networks in machine learning problems [26]. 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 short number of rules in the fuzzy case that will be described later.

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 $\log \frac{p(x)}{1-p(x)}$ [10]; therefore, the additive model is

$$\log \frac{p(\text{class}(x) = 1)}{p(\text{class}(x) = 0)} = f(x_1, \ldots, x_n) = r_0 + \beta_1 r_1(x) + \ldots \tag{10}$$

```
$f_{i0k} = 0$
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 $A^j$ that minimices
            $\text{fitness}(A^j) = \sum_{i}^{n} w_{ijk} \left( s^j \cdot A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}} \right)^2$
            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
```

Fig. 1. 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 $p_{j1}(x) = 1 - p_{j2}(x)$.

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

$$p(x) = \frac{e^{f(x)}}{1 + e^{f(x)}} \tag{11}$$

If the greedy version of generalized backfitting, mentioned in the preceding subsection, is applied to this model, it is obtained the Logitboost algorithm [7]. An outline of the adaptation of this method to learn fuzzy rules, as described in [20], is shown in Figure 1. This last algorithm will be extended in this paper, as shown in the next section, to deal with the max-combination of models mentioned in the introduction.

III. Proposed algorithm

A. Definition of the weak learners

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

$$\arg\max_{k=1,\ldots,p} \bigvee_{j=1}^{N} A^j(\mathbf{x}) \wedge s_k^j \tag{12}$$

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

$$\arg\max_{k=1,\ldots,p} \sum_{j=1}^{N} A^j(\mathbf{x}) \cdot s_k^j \tag{13}$$

and that allowed 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 obtain the weights s_k^j of the expression that follows:

$$\arg\max_{k=1,\ldots,p} \{\max_{j=1}^{N} A^j(\mathbf{x}) \cdot s_k^j\} \tag{14}$$

which is not a sum of terms and therefore not an additive model. But, if we define a function

$$I(\mathbf{x}, j) = \begin{cases} 1 & \text{if } j = \arg\max A^j(\mathbf{x}) \cdot s_k^j \\ 0 & \text{elsewhere} \end{cases} \qquad (15)$$

(in words, $I(\mathbf{x}, j) = 1$ if the rule number j is the winner in the point $\mathbf{x}$, and 0 if not) it is clear that eq. 14 can be rewritten as

$$\arg\max_{k=1,\ldots,p} \sum_{j=1}^{N} I(\mathbf{x}, j) \cdot A^j(\mathbf{x}) \cdot s_k^j \qquad (16)$$

and the products $I(\mathbf{x}, j) \cdot A^j(\mathbf{x})$ can be regarded as weak learners. Anyway, this change in the notation does not completely solve the problem; it is needed to estimate the function I.

B. Recurrent estimation of the function I

Let us suppose for the time being that we have an incomplete rule base, to which we want to add a new fuzzy rule. It is immediate that we can calculate the values of $I(\mathbf{x}_k, j)$ for all the rules in the incomplete base, at the points in the training set.

But it is also evident that some of these values I will change after the new rule is added: otherwise, that new rule would not win at any example in the training set. Since I participates in the definition of the weak learners, this means that all values s_k^j must be recalculated every time a rule is added. In other words, the prefitting version of the algorithm is mandatory for this problem, because the consequents of all the rules in the incomplete base need to be recalculated after a new rule is added to the base.

If all the values $I(\mathbf{x}_k, j)$ are known, the least squares election of s_k^j is an standard problem of linear regression, that can be solved by means of a pseudoinverse. We propose to use the recurrent algorithm shown in Figure 2, which is a prefitting version of the algorithm in Figure 1: the values of s_k^j are obtained first for an initial estimation of I, and then they are used to build a new definition of this function, which in turn is used to define a new set of values of I, and so on. Observe that we have included an smoothing term α, and that the algorithm finishes when the difference between succesive values of I and s_k^j are small enough.

C. Outline 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 lauched 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

```
procedure **AddFuzzyRule**
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 $||S - S'|| < \epsilon$ and $||I - I'|| < \epsilon'$
  Output $S$ and fitness=$||Z - S \cdot F||$
```

Fig. 2. The procedure AddFuzzyRule takes as inputs a fuzzy classificator of size N and the antecedent of a fuzzy rule. Its output consists of a new fuzzy classificator, 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.

```
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
   Emit the Rule "**if x is** $A^j$ **then t**$(c_k)$=$s_k^j$"
End For
```

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

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.

We will use a coding scheme based in [8]. 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

```
If $x_1$ is High and $x_2$ is Med and $x_3$ is Low
then class is C1 with seg = S1, C2 with seg = 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

```
If $x_1$ is High and $x_3$ is Low then ...
```

is codified with the chain 001 000 100, and

```
If $x_1$ is( High or Med) and $x_3$ 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.

IV. Preliminary benchmark 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 [19], from the literature [11] or synthetic [4].

Boosting algorithms were terminated after the generation of 7 rules for Pima, 4 for Cancer, 5 for Gauss, 10 for Glass and 10 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 Max-Logitboost is steady-state, with ten subpopulations of size 100 each. Every rule is obtained from the best individual after 2500 crossover operations.

In order to compare the accuracy of two learning algorithms, Dietterich analyzes in [5] five statistical tests and states that 5x2cv t-test has low type I error and good power. Later, in [1], a new test called 5x2cv-f, that improves both type I error and power, is proposed. We have adopted this last test in all our experiments.

5 statistical methods (linear and quadratic discriminant analysis, neural networks, kernel estimation of densities and k-nearest neigbours) plus 6 fuzzy descriptive rule based methods (Wang and Mendel's [27], Ishibuchi's [13], Pal and Mandal's [21], Iterative Genetic Learning [2], Random Sets Based [23], Fuzzy Descriptive Adaboost [4]) were compared to Max-Logitboost. The combined boxplots are shown in figure 4. Special care was taken to select the minimum number of rules that produce a meaningful classification for all datasets, in order to keep the rule bases linguistically understandable. Observe that the numerical values of the error could be further lowered if the number of rules was allowed to increase. As a reference, the reader can compare the results here with those in [23] and [4] for the same datasets.

Pima, Cancer and Gauss5 behave as expected; the extra linguistic quality has a cost in accuracy. The difference is not statistically significant, but it is still visible in the boxplots. Gauss results are not different between fuzzy Adaboost and Max-fuzzy Logitboost, and are rather similar to that of GIL, which uses the same inference but adjusted membership functions, without weighting the rules. The behavior of Glass dataset seems anomalous to us (Max-Logitboost improves Adaboost with 88% of confidence.) In particular, the joint results of GIL and Max-Logitboost, confronted to that of Interpretable-Kernel and Adaboost seem to show similarities between groups of methods in terms of the inference used (max-based or sum-based,) that need to be researched further.

V. Concluding Remarks

The advantages of boosting methods when learning fuzzy classifiers are two: as far as we know, the size of the rule base is much smaller than the obtained with any other genetic fuzzy classisier, 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.

The high interaction between rules in Adaboost and Logitboost is a consequence of the "sum of votes" inference scheme. The prefered 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 assert, which states that all patterns in a given fuzzy region belong to the same class. But the single winner inference does not combines the votes of the rules with the arithmetic sum, but the maximum operator, and this difficulties the application of boosting algorithms. 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.

Acknowledgements

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

Referencias

[1] E. Alpaydin (1999) "Combined 5x2cv F Test for Comparing Supervised Classification Learning Algorithms," Neural Computation, 11(8), 1885-1982.

[2] Cordón, O., Del Jesus, M. J., Herrera, F. "A proposal on reasoning methods in fuzzy rule-based classification systems". International Journal of Approximate Reasoning **20**(1), pp. 21-45, 1999.

[3] Cordón, O., Herrera, F. (2000) "A proposal for improving the accuracy of linguistic modeling". IEEE Transactions on Fuzzy Systems, 8, 3, pp. 335-344.

[4] Del Jesus, M. J., Hoffmann, F., Junco, L., Sánchez, L. Induction of Fuzzy Rule Based Classifiers with Evolutionary Boosting Algorithms. IEEE Transactions on Fuzzy Sets and Systems. Admitted for publication.

[5] Dietterich, G. (1998) "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". Neural Computation, 10, 7, pp 1895-1924.

[6] Freund, Y., Schapire, R. "Experiments with a new boosting algorithm". In Machine Learning, Proc. 13th International Conference, pp. 148-156

[7] Friedman, J., Hastie, T., Tibshirani, R. (1998) "Additive Logistic Regression: A Statistical View of Boosting". Machine Learning.

[8] Gonzalez, A., Perez, R. (1996) "Completeness and consistency conditions for learning fuzzy rules," Fuzzy Sets and Systems, vol 96, pp 37-51.

[9] Hand, D. J. *Discrimination and Classification*. Wiley. 1981

[10] Hastie, T. J., Tibshirani, R. (1986) "Generalized Additive Models". Statistical Science, 1, pp 297-318

[11] Haykin, S. *Neural Networks*. Prentice Hall, 1999.

[12] Hoffmann, F., Boosting a Genetic Fuzzy Classifier. in Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol. 3, (Vancouver, Canada), pp. 1564–1569, July 2001

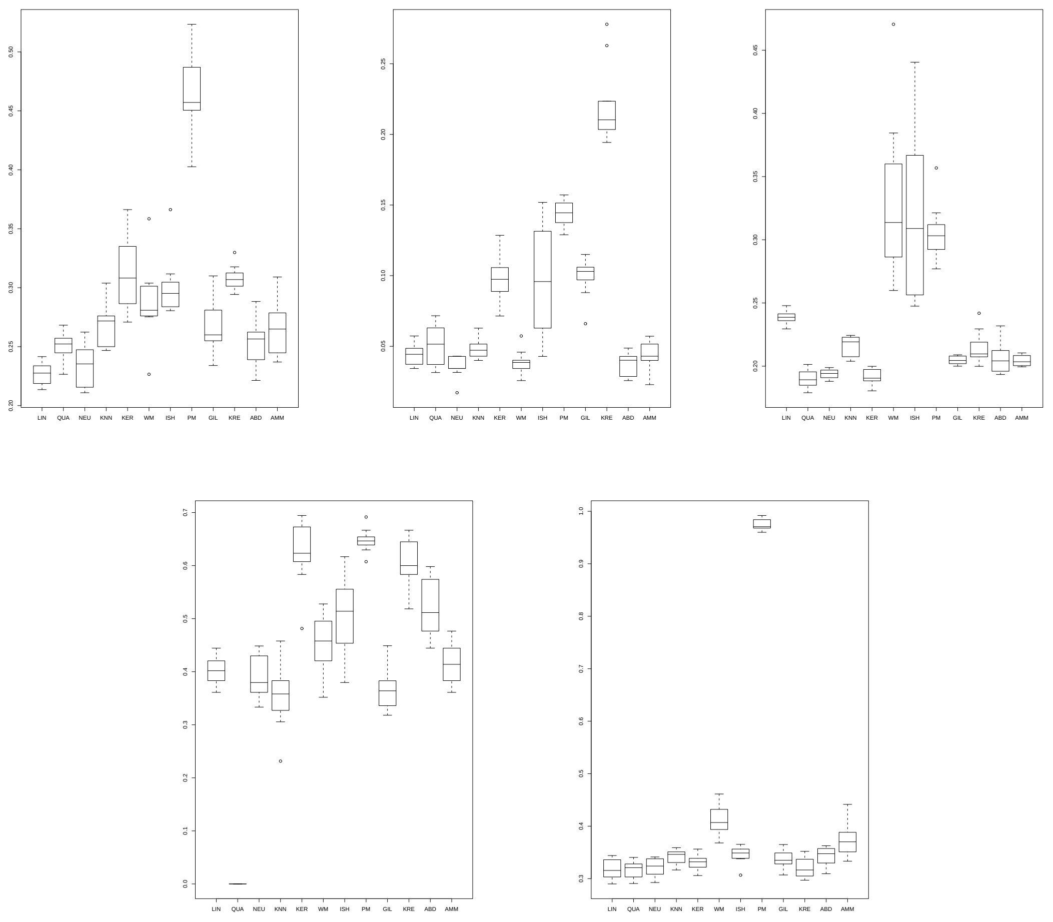

Fig. 4. Boxplots with a comparison between black-boxes (linear and quadratic discriminant analysis, 3 layer perceptron, k-nearest neighbours, kernel estimation of densities) and fuzzy rule based classifiers (Wang and Mendel's, Ishibuchi, Pal and Mandal, Genetic Iterative Learning, Random Set based, Fuzzy Adaboost and Max-Fuzzy Boosting.) The datasets are Pima, Cancer, Gauss, Glass and Gauss5.

	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	0.266
cancer	0.044	0.051	0.035	0.048	0.099	0.039	0.096	0.145	0.099	0.221	0.038	0.043
gauss	0.239	0.190	0.194	0.216	0.191	0.329	0.322	0.306	0.205	0.215	0.206	0.204
glass	0.404	-	0.389	0.354	0.621	0.453	0.503	0.647	0.363	0.606	0.522	0.415
gauss5	0.318	0.317	0.321	0.343	0.332	0.410	0.345	0.974	0.338	0.321	0.344	0.373

Fig. 5. Mean values of the experiments shown in the preceding figure

[13] Ishibuchi, H., "Distributed representation of fuzzy rules and its application to pattern classification", Fuzzy Sets and Systems 52, pp. 21-32, 1992.

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

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

[16] Junco, L., Sanchez, L."Using the Adaboost algorithm to induce fuzzy rules in classification problems", Proc. ESTYLF 2000, Sevilla, pp 297-301.

[17] Kuncheva, L. I. Fuzzy Classifier Design. Springer-Verlag, NY, 2000.

[18] Mallat, S. Zhang, Z. (1993) "Matching pursuits with time-frequency dictionaries". IEEE Trans on Signal Processing 41, pp 3397-3415.

[19] Merz, C. J., Murphy, P.M. (1998). UCI repository of machine learning databases. Available at: http://www.ics.uci.edu/mlearn/MLRepository.html.

[20] Otero, J., Sánchez, L. Induction of descriptive fuzzy classifiers with the Logitboost algorithm. Submitted to Soft Computing.

[21] Pal, S. K., Mandal, D. P. "Linguistic recognition system based in approximate reasoning". *Information Sciences* **61**, pp. 135-161. 1992.

[22] Sánchez, L. "A Fast Genetic Method for Inducting Linguistically Understandable Fuzzy Models". Proc. IFSA NAFIPS, 2001.

[23] L. Sánchez, J. Casillas, O. Cordón, M. J. del Jesus (2002) "Some relationships between fuzzy and random classifiers and models". International Journal of Approximate Reasoning 29, 175-213.

[24] Sánchez, L., Otero, J. A fast genetic method for inducting descriptive fuzzy models. Fuzzy Sets and Systems. Admitted for publication.

[25] Schapire, R., Singer, Y. Improved Boosting Algorithms Using

Confidence-rated Predictions. Machine Learning 37(3): 297-336. 1999

[26] P. Vincent and Y. Bengio, (2002) “Kernel Matching Pursuit”, Machine Learning Journal, Special Issue on New Methods for Model Combination and Model Selection.

[27] Wang, L. X., Mendel, J. (1992) “Generating fuzzy rules by learning from examples”. IEEE Trans. on Systems, Man and Cybernetics, 25, 2, pp. 353-361.

[28] Zhu, J., Hastie, T. (2001) “Kernel Logistic Regression and the Import Vector Machine”. Proc. NIPS 2001, Vancouver, Canada.