Tuning fuzzy partitions or assigning weights to fuzzy rules: which is better?

Luciano Sánchez[1] and José Otero[2]

Departamento de Informática, Universidad de Oviedo
Sedes Departamentales, Edificio 1. Campus de Viesques
33204 Gijón, Asturias, Spain

Summary. The accuracy of linguistic classifiers can be improved with several techniques, but they all compromise the interpretability of the rule base up to a certain degree. Assigning weights to fuzzy rules and tuning the memberships associated to linguistic variables are two of the most common methods. In this work we study whether tuning the membership functions in a linguistic classifier is better or not than adjusting rule weights, in terms of the interpretability of the rule base and the complexity of the output.

To make this analysis independent of the learning method, we will use statistical techniques of estimation over a random sets based linguistic classifier (introduced in previous works) that produces a rule base with weighted rules. The loss of precision produced when weights are restricted to binary values will be evaluated by comparing the output of the former method with an integer programming extension of it which is proposed in this chapter. The relative performances of classifiers with tuned membership and binary rules will be compared to those of uniform partition-based weighted rules, and the statistical relevance of these differences measured.

1 Introduction

It is commonly assumed that some precision must be sacrificed to achieve interpretability. Fuzzy classifiers in which memberships are tuned, or rules are weighted, are expected to be more precise than those in which there are not degrees of confidence in the consequents, and also more precise than those in which the semantic values of the linguistic terms (the fuzzy membership functions) are not modified by the learning algorithm. Nevertheless, a comparative study of the gain of performance obtained when memberships are tuned, or rules are weighted, is not immediate [12][9][10]. Neither adhoc fuzzy learning methods like [17][8][13][1], nor modern genetic methods [2] produce the optimum classifier; therefore it is difficult to separate that gain of performance which is inherent to the extra representative power in adjustable classifiers from the gain that is due to the specific properties of the learning algorithm. A method that is not based in heuristics neither stochastic techniques, and that is valid to induce both weighted and not weighted fuzzy classifiers, is needed.

In [15] a random sets based model, numerically identical to a weighted fuzzy rule based classifier, was proposed. All parameters in this last model

have statistical sense, thus they can be estimated with conventional procedures: in particular, maximum likelihood can be used to estimate all of them. The numerical algorithm that originates here involves finding the solution of a constrained optimization problem, that can be transformed into an unconstrained one with the help of Lagrange multipliers, and then solved by nonlinear programming (projected gradient method.) We propose now to extend the statistical procedures used in the former work to obtain solutions in which weights are 1 or 0, by means of nonlinear integer programming techniques, so that the difference of performance between fuzzy classifiers with and without degrees of confidence in their consequents can be studied without dependence of the learning algorithm used to induce them from data. It will also be decided whether the loss of linguistic interpretability that is introduced when memberships are tuned is significant.

This chapter is organized as follows: first, we define the concept of random set based, linguistically understandable classifiers, and propose one method for estimating these classifiers' parameters from samples. Then we insert this algorithm into a branch and bound process that produces the best non weighted classifier from that data, and study the statistical significance of the difference between real and integer solutions, corresponding to weighted and not weighted classifiers, in synthetic and real world problems. Finally, we study the effect of tuning the antecedents of the real solution before calculating the integer solution, and decide whether the gain of quality worths the loss of interpretability.

2 Random set based classifiers

2.1 Linguistically understandable classifiers

Let us introduce the notation we will use in this chapter. Suppose we have a set Ω that contains objects ω, each one of them belonging to a class c_i, $i = 1 \dots C$, and we perform the set of measurements $X(\omega) = (X_1(\omega), \dots, X_M(\omega))$ over every object. 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 [6].

Sometimes it is needed to obtain a linguistically understandable classifier. This means we need to setup a decision rule that can be codified in a language that allows it to be linguistically communicated. Fuzzy logic based classifiers have this property [19]. The semantic of a fuzzy classifier depends on the equivalence between linguistic values of attributes and certain fuzzy sets defined over the range of every feature [18], and on a fuzzy inference procedure. For example, the sentence

`if x is` $\widetilde{A}$ `then class = (`c_1 `with conf` p_1, ..., c_C `with conf` p_C`)`

means

$$\text{truth}(\widetilde{A} \rightarrow c_1) = p_1,\ \ldots,\ \text{truth}(\widetilde{A} \rightarrow c_C) = p_C.$$

where the concept "$\widetilde{A}$" is linked to a fuzzy subset of the feature space.

The same sentence can be given a probabilistic meaning [16]: if A is a *crisp* subset of $X(\Omega)$, then the sentence means

$$P(c_1|A) = p_1, \ldots P(c_C|A) = p_C,$$

with $\sum_{i=1}^{C} p_i = 1$. The probabilistic logic-based view has some advantages. It can be used to give a statistical meaning to the learning of a rule-based classifier from examples, as we will show below. But it is not immediate to compare it to fuzzy logic based rules, in which "$\widetilde{A}$" is a fuzzy set. On the contrary, we will show that fuzzy classifiers can be compared to certain random set-based classifiers, which in turn are defined as the expectation of the probabilistic logic-based ones for a given sample distribution.

2.2 Fuzzy and Random set based classifiers

Suppose we have a machine learning procedure to estimate a crisp partition $\{A_j\}_{j=1,\ldots,S}$ of the feature space, $A_j \cap A_k = \phi$ for $j \neq k$, plus the values $P(c_i|\mathbf{x} \in A_j) = p_{ij}$ that define a probabilistic logic-based classifier. The machine learning task takes as input a random sample $\mathbf{X}$ of classified examples and produces a partition $\{A_1^{\mathbf{X}}, A_2^{\mathbf{X}}, \ldots\}$ and the values $P(c_i|\mathbf{x} \in A_j^{\mathbf{X}}) = p_{ij}^{\mathbf{X}}$. In turn, for an input value $\mathbf{x}$, the classifier that was learned from the sample $\mathbf{X}$ outputs the probabilities of all classes according to the formula

$$P(c_i|\mathbf{x}, \mathbf{X}) = \sum_j p_{ij}^{\mathbf{X}} A_j^{\mathbf{X}}(\mathbf{x}) \tag{1}$$

where $A_j^{\mathbf{X}}(\mathbf{x})$ is 1 if $\mathbf{x} \in A_j^{\mathbf{X}}$ and 0 else.

It is well known that the expected error of this classifier (we will use the notation $\langle\rangle_{\mathbf{X}}$ to denote expectation with respect to the sample distribution of $\mathbf{X}$) is the sum of two positive terms, *bias* plus *variance* [5], where the bias is the error of the average classifier

$$P(c_i|\mathbf{x}) = \left\langle \sum_j p_{ij}^{\mathbf{X}} A_j^{\mathbf{X}}(\mathbf{x}) \right\rangle_{\mathbf{X}}. \tag{2}$$

Since the variance term is positive, the error of this average classifier is lower that the mean error of the individual classifiers. We can reorder the terms in (2):

$$P(c_i|\mathbf{x}) = \sum_j \left\langle p_{ij}^{\mathbf{X}} A_j^{\mathbf{X}}(\mathbf{x}) \right\rangle_{\mathbf{X}} \tag{3}$$

and, if the random variables $p_{ij}^{\mathbf{X}}$ and $A_j^{\mathbf{X}}(\mathbf{x})$ (both defined with respect to $\mathbf{X}$) were independent,

$$P(c_i|\mathbf{x}) = \sum_j \left\langle p_{ij}^{\mathbf{X}} \right\rangle_{\mathbf{X}} \left\langle A_j^{\mathbf{X}}(\mathbf{x}) \right\rangle_{\mathbf{X}} = \sum_j \left\langle p_{ij}^{\mathbf{X}} \right\rangle_{\mathbf{X}} \Phi_j(\mathbf{x}) \tag{4}$$

where $\Phi_j(\mathbf{x})$ is the one-point coverage function of the random set $A_j^{\mathbf{X}}$. Observe that $\sum_j A_j^{\mathbf{X}}(\mathbf{x}) = 1$ for all $\mathbf{x}$, so $\left\langle \sum_j A_j^{\mathbf{X}}(\mathbf{x}) \right\rangle_{\mathbf{X}} = 1$ and then $\sum_j \Phi_j = 1$ for all values of $\mathbf{x}$ (in words, the sum of the memberships is always equal to 1).

Expression (4) is very similar to the fuzzy inference formula applied to a fuzzy classifier defined by the rules "truth$(\widetilde{A}_j \to c_i) = t_{ij}$", $j = 1, \ldots, S$:

$$\text{truth}(c_i) = \bigvee_j \text{truth}(\widetilde{A}_j \to c_i) \wedge \widetilde{A}_j(\mathbf{x}) = \bigvee_j t_{ij} \wedge \widetilde{A}_j(\mathbf{x}). \tag{5}$$

The truth value t_{ij} is the counterpart of the value $\left\langle p_{ij}^{\mathbf{X}} \right\rangle_{\mathbf{X}}$, and the one-point coverage function $\Phi_j(\mathbf{x})$ is the counterpart of the membership function $\widetilde{A}_j(\mathbf{x})$; the t-norm $\wedge$ is restricted to the product, and the t-conorm $\vee$ is replaced by the sum.

The value $\left\langle p_{ij}^{\mathbf{X}} \right\rangle_{\mathbf{X}}$ is the expected probability of class i given the set $\hat{A}_j^{\mathbf{X}}$, this is the mean value of $P(c_i|A_j^{\mathbf{X}})$ for all probabilistic logic-based classifiers with respect to the sample distribution. It can be regarded as the degree of truth of the assertion "All elements in A_j belong to class i". The function $\Phi_j(\mathbf{x})$ is the probability of $\mathbf{x}$ being covered by the random set $A_j^{\mathbf{X}}$, and thus can be associated to the truth of the assertion "$\mathbf{x}$ belongs to A_j". It is possible to assign linguistic labels to the random sets $A_i^{\mathbf{X}}$ and draw their coverage functions in a form that closely resemble a Ruspini fuzzy partition (see Figure 1). In Figure 2 the three types of classifiers that have been discussed (probabilistic, random sets based and fuzzy sets based) are represented along with their inference procedures.

2.3 Approximate and descriptive rules

C rules of the form truth$(\widetilde{A}_j \to c_i)$=$p_{ij}$ can be combined into the assert

$$\begin{array}{l} \texttt{if x is } \widetilde{A}_j \texttt{ then} \\ \texttt{class = (}c_1 \texttt{ with conf } p_{1j},\ \ldots,\ c_C \texttt{ with conf } p_{Cj}\texttt{)}. \end{array} \tag{6}$$

Unless the concept $\widetilde{A}_j$ can be expressed as the conjunction of independent properties defined over every feature, it is difficult to understand the meaning of this last assert; it is easier to use expressions like

$$\begin{array}{l} \texttt{if } x_1 \texttt{ is } \widetilde{A}_{j1} \texttt{ and ... and } x_M \texttt{ is } \widetilde{A}_{jM} \texttt{ then} \\ \texttt{class = (}c_1 \texttt{ with conf } p_{1j}\texttt{, ... ,}c_C \texttt{ with conf } p_{Cj}\texttt{)} \end{array} \tag{7}$$

where all fuzzy sets $\widetilde{A}_{jf}$, $j = 1, \ldots, S$, belong to a given fuzzy partition of the feature f. This is a typical rule structure in the field of fuzzy classifiers.

We will call *linguistic* or *descriptive* to classifiers based on expression 7, and *approximate* fuzzy classifiers to those based on expression 6, following the nomenclature in this book.

Not all approximate fuzzy classifiers can be expressed with linguistic classification rules. The conditions they must fulfill are immediate in fuzzy logic:

$$\widetilde{A}_j(\mathbf{x}) = \widetilde{A}_{j1}(x_1) \wedge \ldots \wedge \widetilde{A}_{jM}(x_M)$$

and exactly the same in probabilistic logic-based rules,

$$\begin{array}{c} \texttt{if } x_1 \texttt{ is } A_{j1} \texttt{ and } \ldots \texttt{ and } x_M \texttt{ is } A_{jM} \texttt{ then} \\ \texttt{class = } (c_1 \texttt{ with conf } p_{1j}\texttt{, } \ldots \texttt{ ,} c_C \texttt{ with conf } p_{Cj}) \end{array} \tag{8}$$

where the antecedents A_j must be hypercubes in the feature space. If $A_j(\mathbf{x}) = 1$ for all $\mathbf{x} \in A$ and 0 else,

$$A_j(\mathbf{x}) = A_{j1}(x_1) \wedge \ldots \wedge A_{jM}(x_M)$$

and all intervals A_{jk} must be elements of the same crisp partition of the feature k.

Recalling equation (1), a probabilistic rule based classifier obtained from the sample $\mathbf{X}$ outputs the probabilities of all classes according to the formula

$$P(c_i|\mathbf{x}, \mathbf{X}) = \sum_j p_{ij}^{\mathbf{X}} \prod_{k=1}^{M} A_{jk}^{\mathbf{X}}(x_k) \tag{9}$$

where $\prod_{k=1}^{M} A_{jk}^{\mathbf{X}}(x_k)$ is 1 or 0. We obtain that

$$P(c_i|\mathbf{x}) = \sum_j \left\langle p_{ij}^{\mathbf{X}} \right\rangle_{\mathbf{X}} \left\langle \prod_{k=1}^{M} A_{jk}^{\mathbf{X}}(x_k) \right\rangle_{\mathbf{X}}. \tag{10}$$

The condition we need to obtain a descriptive random set-based classifier is

$$\Phi_j(\mathbf{x}) = \prod_{k=1}^{M} \Phi_{jk}(x_k) = \left\langle \prod_{k=1}^{M} A_{jk}^{\mathbf{X}}(x_k) \right\rangle_{\mathbf{X}} \tag{11}$$

which is fulfilled when random variables $P(x_k \in A_{jk}^{\mathbf{X}})$ are independent. In this last case,

$$\Phi_{jk}(x_k) = \left\langle A_{jk}^{\mathbf{X}}(x_k) \right\rangle_{\mathbf{X}} \tag{12}$$

and the descriptive classifier can then be expressed as a set of rules of the form

$$\begin{array}{c} \texttt{if } x_1 \texttt{ is } \Phi_{j1} \texttt{ and } \ldots \texttt{ and } x_M \texttt{ is } \Phi_{jM} \texttt{ then} \\ \texttt{class = } (c_1 \texttt{ with conf } \langle p_{1j}^{\mathbf{X}} \rangle_{\mathbf{X}}, \ \ldots, \ c_C \texttt{ with conf } \langle p_{Cj}^{\mathbf{X}} \rangle_{\mathbf{X}}) \end{array}$$

Selecting a parametric family of coverage functions Φ_{jk} is equivalent to define a family of sample distributions over $\mathbf{X}$. It is immediate that all functions Φ_{jk} can be interpreted as elements of a Ruspini's fuzzy partition of feature k. We will use triangular coverage functions in all numerical examples, as shown in Figure 1.

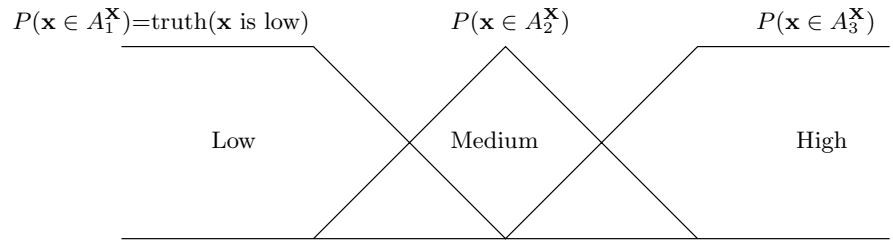

Fig. 1. Graph of the degrees of truth of a numeric value being compatible with the properties "low", "medium" and "high". These labels are associated to random sets $A_1^{\mathbf{X}}$,$A_2^{\mathbf{X}}$,$A_3^{\mathbf{X}}$, respectively, and the degree of truth of the assertion "$\mathbf{x}$ is label" is the probability of $\mathbf{x}$ being covered by the corresponding random set. For example: truth("80 is low")= $P(80 \in A_1^{\mathbf{X}})$.

3 Inducting weighted rule based classifiers

Let us suppose we know the antecedents of the rules (the functions "truth(A_j is $\mathbf{x}$)" or Φ_j, $j = 1, \ldots, S$) and we wish to estimate their consequents (the values "truth($A_j \rightarrow c_i$)" or $\langle p_{ij}^{\mathbf{X}} \rangle_{\mathbf{X}}$.) For simplicity in the notation, let $\theta_{ij} = \langle p_{ij}^{\mathbf{X}} \rangle_{\mathbf{X}}$ and Θ be the parameter vector of all unknown parameters,

$$\Theta = \left(\langle p_{11}^{\mathbf{X}} \rangle_{\mathbf{X}}, \ldots, \langle p_{1S}^{\mathbf{X}} \rangle_{\mathbf{X}}, \ldots, \langle p_{CS}^{\mathbf{X}} \rangle_{\mathbf{X}} \right) = (\theta_{11}, \ldots, \theta_{1S}, \ldots, \theta_{CS}) \,. \quad (13)$$

The likelihood function V_Θ is defined as follows:

$$V_\theta = \prod_{\mathbf{x} \in \mathbf{X}} \sum_{j=1}^{S} \theta_{class(\mathbf{x}),j} \Phi_j(\mathbf{x}) \quad (14)$$

and its maximum, restricted to the conditions $\sum_{i=1}^{C} \theta_{ij} = 1$ for $j = 1, \ldots, S$ and $\theta_{ij} \geq 0$, is a good estimation of the unknowns.

It is easier to work with the logarithm of this function:

$$L(\Theta) = \log(V_\theta) = \sum_{\mathbf{x} \in \mathbf{X}} \log \sum_{j=1}^{S} \theta_{\mathrm{class}(\mathbf{x}),j} \Phi_j(\mathbf{x}). \quad (15)$$

Using Lagrange multipliers to cope with the restrictions, we have to minimize

$$L_1(\Theta, \lambda_1, \ldots, \lambda_S) = \sum_{\mathbf{x} \in \mathbf{X}} \log \sum_{j=1}^{S} \theta_{\mathrm{class}(\mathbf{x}),j} \Phi_j(\mathbf{x}) + \sum_{j=1}^{S} \lambda_j (1 - \sum_{i=1}^{C} \theta_{ij}) \quad (16)$$

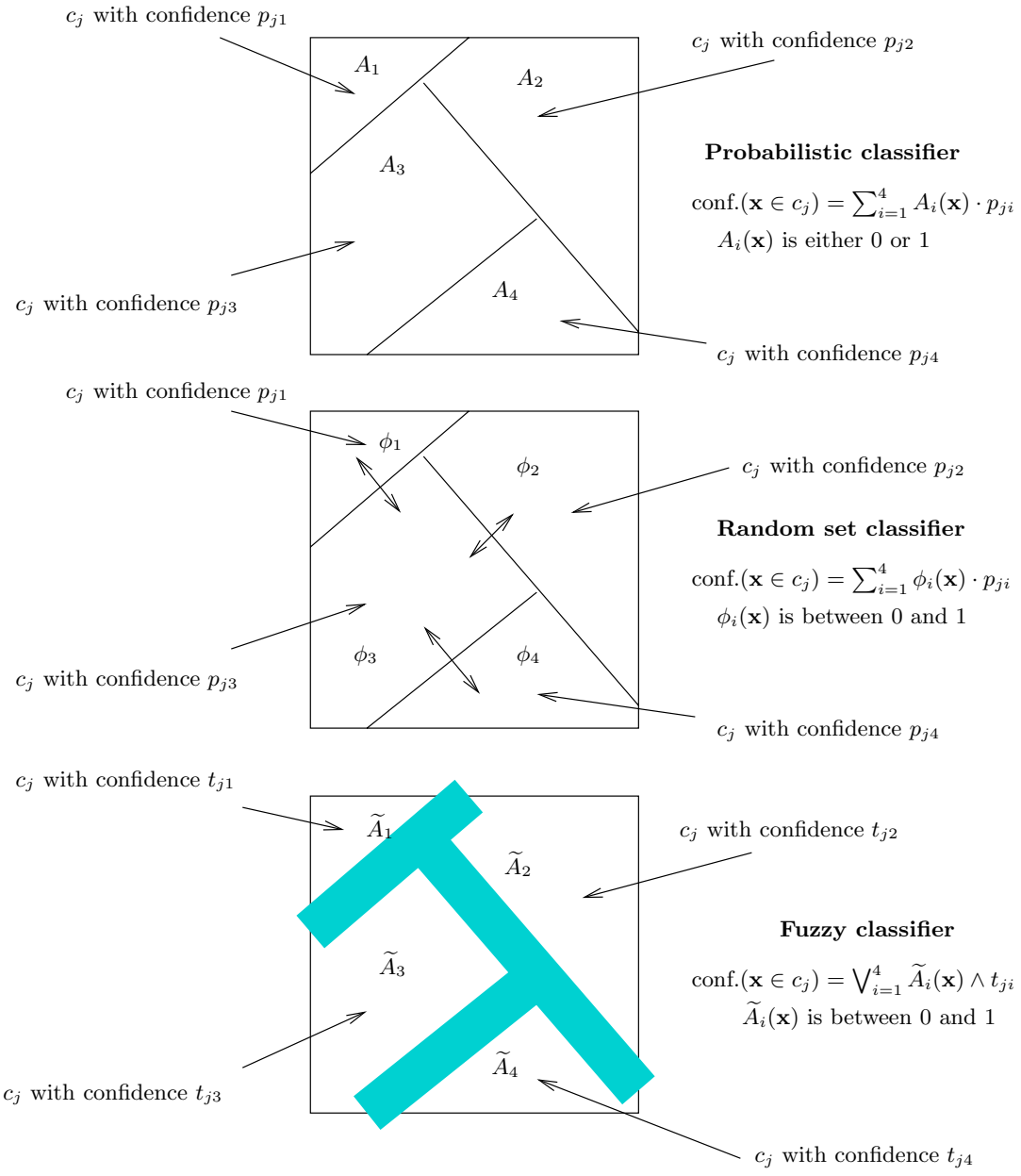

Fig. 2. Three types of classifiers are used in this chapter: Probabilistic classifiers output degrees of confidence depending on a crisp partition. The output of random set classifiers is the expectation of probabilistic classifiers' and confidences are added after multiplying them by the coverage functions. Fuzzy sets use t-norms and t-conorms instead of products and sums. We will compare random set classifiers with fuzzy classifiers in experiments for which fuzzy memberships and coverage functions are numerically identical.

Taking derivatives with respect to θ_{ij} and λ_j, we obtain the following conditions that are true in the minimum:

$$\sum_{\substack{\mathbf{x}\in\mathbf{X}\\ \text{class}(\mathbf{x})=i}} \frac{\Phi_j(\mathbf{x})}{\sum_{j=1}^{S}\theta_{ij}\Phi_j(\mathbf{x})} = \sum_{\substack{\mathbf{x}\in\mathbf{X}\\ \text{class}(\mathbf{x})=k}} \frac{\Phi_j(\mathbf{x})}{\sum_{j=1}^{S}\theta_{kj}\Phi_j(\mathbf{x})} \tag{17}$$

$$\sum_{i=1}^{C}\theta_{ij} = 1, \quad \text{for } j = 1,\ldots,S;\ i,k = 1,\ldots,C. \tag{18}$$

The first set of equalities (equation 17) produces $C-1$ equations and the second one (equation 18) leads to S, thus the search of the parameters consists in numerically solving a system of $C \cdot S$ nonlinear equations.

3.1 Incomplete rule bases

Whilst fuzzy rule banks can be incomplete, probabilistic ones can not. But all consequents can be initialized to the value $1/C$ to express the initial absence of knowledge. If the learning algorithm finishes and there are still rules like these, they can be skipped when doing an inference (eq. 4) without affecting the output of the classifier. We can define an "equivalent number of rules" to compare the complexity of a complete probabilistic rule bank to that of an incomplete fuzzy rule bank as the number of rules for which the degree of confidence in any of their consequent parts is different from $1/C$.

Determining which set of $N_r' < S$ rules produces the best classifier has practical relevance. When the number of features is high, the number of parameters grows above all practical linguistic interpretability, and then it is useful to decide whether an approximate solution in which most of the parameters are $1/C$ (thus they can be ignored) is precise enough. Moreover, many fuzzy rule learning algorithms produce incomplete rule banks and it is reasonable to compare them to random set-based classifiers with the same structure, but also with the same "equivalent number of rules".

As far as we know, finding the best set of N_r' rules in polynomial time is an open problem. An heuristic method to obtain banks with a reduced number of rules is shown in Figure 3. This algorithm does not guarantee that there is not a different set of rules with higher likelihood, but has good properties in practical problems. It just selects rules containing the parameters for which the partial derivatives of eq. 16 are higher in the initial point of the minimization ($\theta_{ij} = 1/C$).

3.2 Adjusting the membership functions

For a fuzzy classifier to be linguistically understandable, all fuzzy memberships have to be related to linguistic values of variables, so that a human intuitively can relate a concept with an imprecise range of values. If some

$\mathbf{G}, \mathbf{D} \in \mathbf{R}^{C \times S}$, $\alpha \in \mathbf{R}$, $i \in 1 \ldots C$, $j \in 1 \ldots S$

function learn-with-weights$(\theta \in \mathbf{R}^{C \times S}, S' \subset 1 \ldots S)$ **returns** $(\theta' \in \mathbf{R}^{C \times S})$
  $\theta' =$ **learn-V**$(\theta,$**calculate-V**$(S'))$
end of learn-with-weights

function calculate-V$(S' \in \mathbf{N})$ **returns** $(\mathbf{V} \subset 1 \ldots S)$
  $\theta_{ij} = 1/C$
  $\mathbf{G}_{ij} = C \sum_{\mathbf{x} \in \mathbf{X}, \text{class}(\mathbf{x})=i} \Phi_j(\mathbf{x}) / \sum_{j=1}^{S} \Phi_j(\mathbf{x})$
  $\mathbf{D}_{ij} = \mathbf{G}_{ij} - (S \cdot C)^{-1} \sum_{i=1}^{C} \sum_{j=1}^{S} \mathbf{G}_{ij}$
  $\texttt{importance}(j) = \sum_{\substack{i=1 \ldots C \\ \mathbf{D}_{ij} > 0}} \mathbf{D}_{ij}$
  $\mathbf{V} =$ `set of values` j `of the` S' `rules with higher importance`
end of calculate-V

function learn-V$(\theta \in \mathbf{R}^{C \times S}, \ \mathbf{V} \subset 1 \ldots S)$ **returns** $(\theta' \in \mathbf{R}^{C \times S})$
  `repeat`
    $\mathbf{G}_{ij} = \sum_{\mathbf{x} \in \mathbf{X}, \text{class}(\mathbf{x})=i} \Phi_j(\mathbf{x}) / \sum_{j=1}^{S} \theta_{ij} \Phi_j(\mathbf{x})$
    $\mathbf{D}_{ij} = \mathbf{G}_{ij} - (S \cdot C)^{-1} \sum_{i=1}^{C} \sum_{j=1}^{S} \mathbf{G}_{ij}$
    `if` $j \notin \mathbf{V}$ `then` $\mathbf{D}_{ij} = 0$
    `search` α `that minimizes` $L(\texttt{normalize}(\theta + \alpha \cdot \mathbf{D}))$
    $\theta = \texttt{normalize}(\theta + \alpha \cdot \mathbf{D})$
  `until` $\alpha ||D|| < \epsilon$
  $\theta' = \theta$
end of learn-V

function normalize$(\mathbf{X} \in \mathbf{R}^{C \times S})$ **returns** $(\mathbf{X}' \in \mathbf{R}^{C \times S})$
  `if` $(\mathbf{X}_{ij} < 0)$ $\mathbf{X}_{ij} = 0$
  $\mathbf{X}_{ij} = \mathbf{X}_{ij} / \sum_{i=1}^{C} \mathbf{X}_{ij}$
  $\mathbf{X}' = \mathbf{X}$
end of normalize

Fig. 3. Pseudo code of the numerical algorithm used to approximately solve the set of equations (18), producing at most S' rules. The linear search (determination of the value of α) was implemented with Brent's method. All points examined fulfill eq. (18) because of the function `normalize`, and the algorithm stops when the conditions (17) are approximately true.

changes are allowed to these ranges, this last association is less clear, but the precision of the classifier is increased.

The same relationship between random sets and fuzzy sets that has been used up to now, can be extended to estimate parameters defining the semantic values of the linguistic terms as well. Therefore, antecedents can be induced from data along with the consequent parts of the rules. There are some numerical difficulties with this adjust, arising from the fact of functions 15 and 16 being not differentiable with respect to these parameters: $\phi(\cdot)$ is piecewise linear with respect to them and thus gradient based optimization algorithms are not applicable.

We propose to combine the process shown in Figure 3 with a robust deterministic evolutionary algorithm, Nelder and Mead's simplex [11]. We can setup an objective function for the simplex that depends only on the parameters defining the linguistic partitions. This function calls a full gradient descent algorithm to obtain the consequents that fit the partition being evaluated, and eq. 15 is applied to them to get the return value. This way, the evolutionary search selects changes in the membership function and the right consequents for them are estimated with gradient descent.

4 Inducting rules with binary weights

4.1 Random set based rules with binary weights

Extending the usual nomenclature in fuzzy modelling, three type of fuzzy rules in classification problems can be considered. Let us name "Type 1" fuzzy rules to those that have the form

$$\texttt{if x is } \widetilde{A} \texttt{ then class = } c_k.$$

"Type 2" have one consequent, weighted by a degree of confidence

$$\texttt{if x is } \widetilde{A} \texttt{ then class = } c_k \texttt{ with conf. } p_k$$

and rules not in these two classes (more than one consequent, with or without weights) are "type 3" rules. The interpretability of "type 1" fuzzy rules is higher than those of weighted or "type 3" rules, because they do not carry numerical information in their linguistic expression. Random set based rules are, by definition, equivalent to "Type 3" rules, which are uncommon in fuzzy sets literature. In this section we discuss how to adapt the maximum likelihood estimation discussed before to obtain "Type 1" (we will also call them "not weighted", or "binary") rules.

Two considerations must be taken before:

1. Under the maximum vote scheme (which is the fuzzy inference method equivalent to that of random sets based descriptive classifiers, as we have mentioned) all confidences in the whole rule base can be multiplied by a

constant, and this does not change the description surfaces of the classifier. As a consequence of this, any base comprising "type 3" rules can be written in terms of "type 1" rules. This will be made clear with an example: observe the following three "type 1" fuzzy rules:

$$\texttt{if x is } \widetilde{A} \texttt{ then class = 1}$$

$$\texttt{if x is } \widetilde{A} \texttt{ then class = 1}$$

$$\texttt{if x is } \widetilde{A} \texttt{ then class = 2.}$$

This set of rules is equivalent to this single one:

$$\texttt{if x is } \widetilde{A} \texttt{ then class = (1 with conf 2s, 2 with conf s)}$$

where the value of the constant "s" depends on the remaining rules in the rule base. It is not difficult to prove that one can always convert a rule base comprising "type 3" rules with rational weights into a "type 1" rule base by splitting each rule and making an adequate number of copies of it. Therefore, the linguistic interpretability of a rule bank in which the antecedents can be repeated is exactly the same as the interpretability of a rule containing "type 3" rules.

2. The addition of a constant to all confidences in one isolated rule does not modify the bank. This means that the rule

$$\texttt{if x is } \widetilde{A} \texttt{ then class = (1 with conf s, 2 with conf s)}$$

can be replaced by

$$\texttt{if x is } \widetilde{A} \texttt{ then class = (1 with conf 0, 2 with conf 0)}$$

and removed from the rule base.

Because of these considerations, we decided that random set based rules equivalent to "type 1" fuzzy rules can belong to two categories:

1. if $\mathbf{x}$ is ϕ_j then class $= c_k$
2. if $\mathbf{x}$ is ϕ_j then class $=$ (1 with conf $1/C$, ..., C with conf $1/C$)

where the second type of rules do not appear in the linguistic expression of the rule base. It is remarked again that we will not admit that two different rules in the base have the same antecedent; it is clear for us that, if this restriction is not enforced, all experiments would conclude that there is not a loss of classification power when weights are binary with respect to that of real valued weights.

4.2 Learning algorithm

Learning the non-weighted rule base is a nonlinear integer programming problem. Standard branch and bound techniques can be applied, as shown in the pseudo code in figure 4, to obtain the best set of "type 1" rules that approximate a base of "type 3" rules.

The crux of the algorithm consists in deciding whether each weighted rule will be replaced by either the rule

$$\text{if } \mathbf{x} \text{ is } \phi_j \text{ then class} = c_k$$

or

$$\text{if } \mathbf{x} \text{ is } \phi_j \text{ then class} = (\ 1 \text{ with conf } 1/C, \ldots, C \text{ with conf } 1/C),$$

i.e., removed from the base. Let us suppose we replace the first rule of a weighted classifier comprising S' rules by a "type 1" rule; there are $C+1$ different replacements (C binary rules and removing the rule from the base). We can think that each one of these substitutions originates a new weighted subproblem, where the first rule is fixed and the remaining $S'-1$ rules should be modified to find a new maximum likelihood estimation. Let us solve all these $C+1$ weighted problems, write down the final values of the likelihood in each case, and recursively repeat the process for every one of them (the second rule is replaced by a binary one or removed, and so on), finishing when all S' rules have been replaced or removed. If we arrange the result of all experiments in a tree, the leaves are solutions to the integer problem and the internal nodes are solutions to the weighted subproblems.

Obviously, we only need to search a part of this tree, because the likelihoods of intermediate weighted classifiers (the internal nodes of the tree) are lower bounds of the likelihood of the binary weighted classifiers (the leaves of the subtree originated in the internal node). Therefore, as soon as we know the likelihood of any binary solution, we can skip all recursive calls for which the real solution is higher than the likelihood of the binary solution, and prune the search tree as it is shown in the pseudo code in figure 4.

There are three further improvements to the speed of convergence of this algorithm:

1. If we know that the likelihood of the binary solution is in a certain range of the real solution (for example, we usually can expect that the binary classifier log-likelihood is not worse than the real classifier's one +10%) we can skip the recursive calls for which the lower bound is higher than this value, even if a better binary solution has not been reached yet.
2. The order in which the intermediate problems are solved is important: if the problems with a lower bound are solved first, many paths will be removed from the search.
3. If we admit that any binary solution within a certain range of the real solution is precise enough, we can stop the search as soon as this value is reached.

```
best-L ∈ R, best-θ ∈ R^{C×S}, low-bound ∈ R^{C+1}

function learn-bin-weights(θ ∈ R^{C×S}, S' ∈ N) returns (θ' ∈ R^{C×S})
      branch-and-bound(θ, calculate-V(S'))
      θ' = best-θ
end of learn-bin-weights

procedure branch-and-bound(θ ∈ R^{C×S}, V ⊂ 1...S)
   if (V = ∅) then
         if (L(θ) < best-L) then
         best-L=L(θ)
         best-θ = θ
       end if
   else
      r = first element of V
      for k ∈ 0...C
         if (k = 0) θ_ir = 1/C else θ_ir = δ_ir
         low-bound_k = learn-V(θ, V − {r})
         if (low-bound_k < best-L) branch-and-bound(θ, V − {j})
      end for
   end if
end of branch-and-bound
```

Fig. 4. Simplified pseudo code of the numerical algorithm used to approximately solve the set of equations (18), producing at most S' rules with binary weights. Heuristics used to shorten the search (described in section 4.2) are not shown.

5 Numerical examples

5.1 Pure linguistic classification problem

The behavior of the algorithm will be illustrated first with a synthetic example, by means of a data set generated so that the Bayes solution can be described without error by means of a descriptive classifier comprising the following set of type 3 rules:

If x_1 is $\widetilde{R}_1$ and x_2 is $\widetilde{R}_1$ then class$_1$=0.90 and class$_2$=0.10
If x_1 is $\widetilde{R}_1$ and x_2 is $\widetilde{R}_2$ then class$_1$=0.85 and class$_2$=0.15
If x_1 is $\widetilde{R}_1$ and x_2 is $\widetilde{R}_3$ then class$_1$=0.60 and class$_2$=0.40
If x_1 is $\widetilde{R}_2$ and x_2 is $\widetilde{R}_1$ then class$_1$=0.40 and class$_2$=0.60
If x_1 is $\widetilde{R}_2$ and x_2 is $\widetilde{R}_2$ then class$_1$=0.80 and class$_2$=0.20
If x_1 is $\widetilde{R}_2$ and x_2 is $\widetilde{R}_3$ then class$_1$=0.40 and class$_2$=0.60
If x_1 is $\widetilde{R}_3$ and x_2 is $\widetilde{R}_1$ then class$_1$=0.20 and class$_2$=0.80
If x_1 is $\widetilde{R}_3$ and x_2 is $\widetilde{R}_2$ then class$_1$=0.10 and class$_2$=0.90
If x_1 is $\widetilde{R}_3$ and x_2 is $\widetilde{R}_3$ then class$_1$=0.00 and class$_2$=1.00

where the memberships $\widetilde{R}_1$, $\widetilde{R}_2$, $\widetilde{R}_3$ are shown in Figure 5. An algorithm that can generate examples for this problem is shown in Figure 6. Let us generate 1000 examples and apply the algorithms in Figures 3 and 4 to infer the values of the coefficients, with both weighted and not weighted versions.

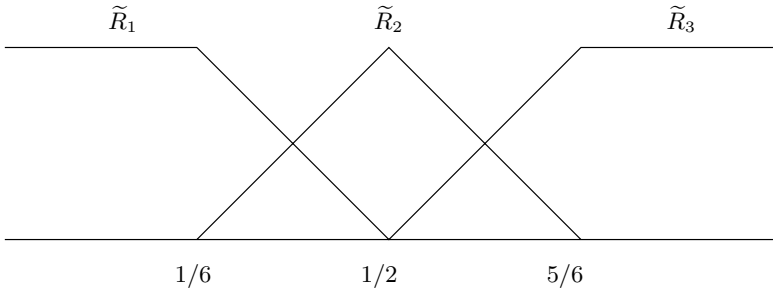

Fig. 5. Membership functions of the example in Section 5.1

x_1=random(0,1); x_2=random(0,1)
$p_1 = 0.90\widetilde{R}_1(x_1)\cdot\widetilde{R}_1(x_2) + 0.85\widetilde{R}_1(x_1)\cdot\widetilde{R}_2(x_2) + 0.60\widetilde{R}_1(x_1)\cdot\widetilde{R}_3(x_2) +$
$0.40\widetilde{R}_2(x_1)\cdot\widetilde{R}_1(x_2) + 0.80\widetilde{R}_2(x_1)\cdot\widetilde{R}_2(x_2) + 0.40\widetilde{R}_2(x_1)\cdot\widetilde{R}_3(x_2) +$
$0.20\widetilde{R}_3(x_1)\cdot\widetilde{R}_1(x_2) + 0.15\widetilde{R}_3(x_1)\cdot\widetilde{R}_2(x_2) + 0.00\widetilde{R}_3(x_1)\cdot\widetilde{R}_3(x_2)$
if (random(0,1)< p_1) output (x_1, x_2, c_1) else output (x_1, x_2, c_2)

Fig. 6. Algorithm used to output a point of the learning sample in the problem discussed in Section 5.1.

0.90	0.40	0.20
0.10	0.60	0.80
0.85	0.80	0.10
0.15	0.20	0.90
0.60	0.40	0
0.40	0.60	1

0.887	0.345	0.198
0.112	0.654	0.801
0.756	0.744	0.100
0.243	0.255	0.899
0.664	0.354	0
0.335	0.645	1

0.5	0.5	0.5
0.5	0.5	0.5
1	0.5	0.5
0	0.5	0.5
0.5	0.5	0
0.5	0.5	1

Fig. 7. True (left), estimated weighted rules (center) and estimated binary rules (right) values for the example explained in Section 5.1.

The inferred rule banks are summarized in Figure 7. The weighted version recovers the original base, while the not weighted one, which is the most

precise "type 1" base, (in fact, since it includes rules with consequents "0.5-0.5" it can be argued that this is not strictly a type 1 classifier; see comments in section 4.1) is clearly suboptimal and comprises only two rules (i.e, only two rules with different values in their consequent part; recall the comments in section 3.1:)

$$\texttt{If } x_1 \texttt{ is } \widetilde{R}_2 \texttt{ and } x_2 \texttt{ is } \widetilde{R}_1 \texttt{ then class}_1$$
$$\texttt{If } x_1 \texttt{ is } \widetilde{R}_3 \texttt{ and } x_2 \texttt{ is } \widetilde{R}_3 \texttt{ then class}_2\texttt{.}$$

This classifier has an estimated error rate of 0.41, while the real solution has an error of 0.26. This example shows us that it is not immediate to pass from the real solution to the best binary solution. In particular, one can not apply an heuristic method to obtain "type 1" rules from "type 3" rules, one by one: such a conversion would depend not only on the single rule being considered, but on rules surrounding it.

5.2 Graphical analysis: Haykin's two Gaussian problem

With this second example we intend to study the differences in the decision surface between "type 1" and "type 3" rule bases. To be able to do a graphical representation, we are going to analyze the data set proposed in [7]: 4000 points taken from two overlapping Gaussian distributions with different variances. The optimal decision surface is a circle, and the Bayesian test error is 0.185. The error of the linear classifier is 0.24, which is near enough the optimal solution to confuse many rule learning algorithms. The shape of the decision surface in areas with low density of examples (i.e., the left side of the circle) does not contribute too much to the classification error.

In Figure 8, descriptive classifiers are compared when the number of linguistic terms in every partition ranges from 3 to 5. Uniform, unadjusted fuzzy partitions were used. In that Figure we observe that the decision surface of the "type 1" descriptive random set-based classifier is nearer to the "type 3" surface than we could intuitively think. In the worst case (the leftmost one) the difference between "type 1" and "type 3" banks produces less than a 2% increase in the classification error, even while the "type 1" base has two rules less and they all are less complex.

5.3 Significance of the loss of classification power

In third place, to judge whether the loss of power produced when "type 3" bases are downgraded to "type 1" bases, we will study 5 cases: the problem introduced in the preceding section, a multi class synthetic problem similar to "Gauss" but involving five classes (it will be named "Gauss-5"), and three more real-world problems from UCI [14]: Pima, Cancer and Glass.

The experimental framework is as follows: 5x2cv Dietterich's test [4] will be applied to assess the statistical relevance of the differences between the

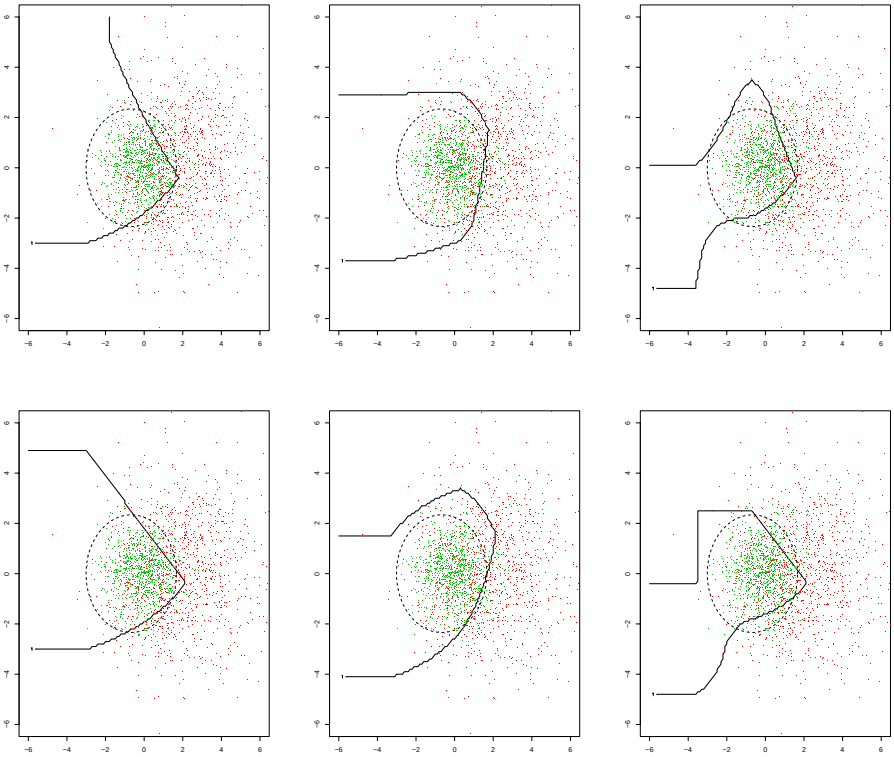

Fig. 8. Effect of the number of terms in the fuzzy partition in the weight removal process. Upper part, from left to right: decision surfaces induced by RSB for classifiers with 3, 4, and 5 terms/feature (9, 16 and 25 "type 3" rules) in the "Gauss" problem. Lower part: the same rule bases, downgraded to "type 1". The dashed line is the optimal decision surface.

two types of rule bases being considered. Data sets are randomly permuted first; the first half of samples is used to train the method, and the second half to test it. Training and test sets are swapped and the learning and test phase repeated. This is repeated for 5 different random permutations. Training errors are discarded, so that box plots only show the dispersion of the error in test sets.

Other statistical (linear, quadratical, nearest neighbor) and artificial intelligence based (neural networks, Wang and Mendel's [3,1], Hong and Lee's [8], Pal and Mandal's [13] and Cordón, Del Jesus' et al. [2] fuzzy classifiers) are included so that the reader can judge the magnitude of the differences between the two methods being considered here. Results of the genetic method in [2] are among the state of the art results in fuzzy classification, and involve

tuning of the linguistic partitions in the antecedent; all other fuzzy classification algorithms are based on heuristics and are included as a reference only; besides some of them achieve good results in some data sets, their results are not as consistent as the former genetic method.

	LIN	QUA	NEU	1NN	WM	HL	PM	GIL	KRE	KBI
pima	0.227	0.252	0.255	0.289	0.287	0.301	0.464	0.269	0.238	0.237
cancer	0.044	0.051	0.047	0.048	0.129	0.058	0.087	0.099	0.043	0.043
gauss	0.239	0.190	0.200	0.267	0.477	0.304	0.457	0.205	0.217	0.220
glass	0.403	-	0.439	0.354	0.453	0.503	0.647	0.363	0.392	0.384
gauss5	0.317	0.317	0.321	0.413	0.539	0.344	0.759	0.338	0.328	0.388

Fig. 9. Mean test values of problems in section 5.3

The mean values of test errors are included in Figure 9. Random set based classifiers comprise 200 rules in Pima, Cancer and Glass, and 9 in Gauss and Gauss-5. All features have three linguistic terms but in Cancer data set, where only two values were needed.

The only statistically significant difference between "type 3" and "type 1" data sets is in Gauss-5 (the p-value of the contrast is 0.08, thus we reject the hypotheses of binary and weighted bases producing similar results, with a 92% level). The difference in Gauss has a p-value of 50%. Cancer produced the same results in 9 of the 10 repetitions, thus box plots are roughly the same. Binary results with Pima and Glass seem to be better than the weighted, but the difference is well under the expected deviation of the results.

5.4 Importance of the membership tuning process

Finally, we will study whether an adjust in the memberships recovers the information lost, in the cases in which downgrading to "type 1" rules made a difference; for Gauss, Cancer, Pima and Glass there are not relevant dissimilarities between either the real or the binary solution and the black boxes, thus a membership tuning makes no sense for them.

Let us focus on the problem for which there are significant differences, Gauss-5. This data set comprises 5 classes and there is a maximum of 16 rules in it, thus the weight removal process remove a lot of information and the "type 1" classifier does not perform properly. One may question whether there exists a definition of the membership functions for which the behavior of the "type 1" rule base is comparable to that of black boxes and "type 3" rules. To check this, we have tuned the membership functions, as explained in section 3.2, before launching the weight removal process, and compared the results of weighted rules with unadjusted (uniform partitions) memberships in the antecedents with that of not weighted rules with tuned memberships. Results are plotted in Figure 11. While tuning the membership always improves the

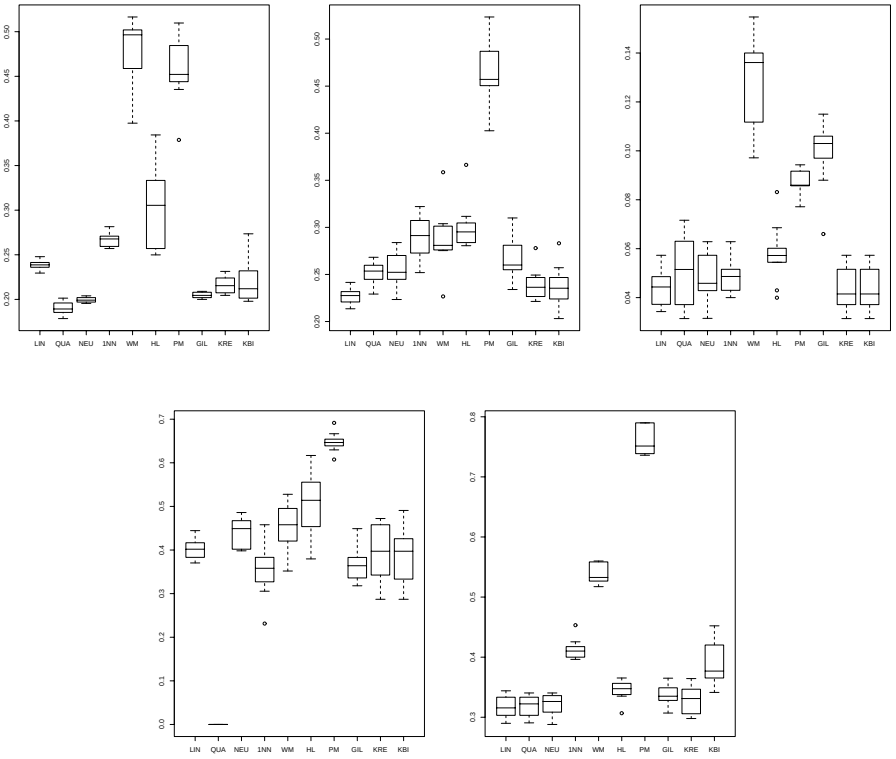

Fig. 10. From left to right and from upper to lower: Box plots of the differences between weighted and not weighted classifiers in gauss, pima, cancer, glass and gauss-5 problems. The columns are: linear, quadratic, neuronal, nearest neighbor, Wang and Mendel's, Hong and Lee's, Pal and Mandal's, Genetic Iterative Learning, Random sets based with "type 3" and "type 1" rules. The bars represent the dispersion of the test results in the 10 repetitions of the experiment.

final results, the gain of classification power does not compensate the loss produced in the weight removal.

6 Concluding remarks and future work

Experimental results have shown that, most of times, there exists very little difference between black boxes and weighted probabilistic rules. These differences decrease with the number of rules, and are statistically significant only when the rule base is rather small. Taking into account that random set based classifiers did not modify the fuzzy memberships in the antecedents, we doubted that the effect of tuning the antecedents was important against the right selection of weights in the rule consequents.

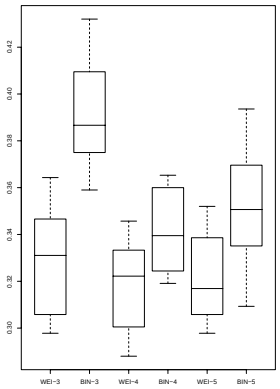

Fig. 11. Comparison between unadjusted memberships + weighted rule bases and adjusted memberships + not weighted rule bases in Gauss-5. From left to right: box plots of weighted and binary solutions of Gauss 5, with 3, 4 and 5 elements/partition. In this case, tuning the memberships does not recover the classification power lost in the weight removal process. The bars represent the dispersion of the test results in the 10 repetitions of the experiment.

Against our initial thought, the results of this study show that the use of weighted or "type 3" does not uniformly produce results significantly better than those obtained by simpler (and easier to interpret) rules without weights. It is remarkable that neither weights in the rules nor multi consequent rules achieved significant improvements in the representation power in real world data sets (where the rule base comprised more than one hundred rules.) In the set of experiments that we performed, the tradeoff between precision and interpretability was best achieved when memberships were uniform and confidences were not used in rule bases with size from moderate to large; by the contrary, both weighted rules and adjusted memberships improved the results of type 1 rules in small bases. In this last case, using weights on uniform partitions produces a gain of precision similar to that achieved when both the definition of the linguistic terms are adjusted, and type 1 rules used.

Acknowledgments

The authors wish to thank the anonymous reviewers for their effort revising this chapter and their valuable suggestions for future works.

References

1. 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.
2. Cordón O., del Jesus M. J., Herrera F. y Lozano M. (1999) Mogul: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. *International Journal of Intelligent Systems* 14(9).
3. Chi, Z., Yan, H., Pham, T. *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. World Scientific. 1996.
4. Dietterich, G. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". Neural Computation, 10(**7**), pp 1895-1924. 1998
5. Geman, S., Bienenstock, E., Doursat, R. "Neural networks and the bias/variance dilemma". Neural Computation, 4, pp. 1-58. 1992.
6. Hand, D. J. *Discrimination and Classification*. Wiley. 1981
7. Haykin, S. *Neural Networks*. Prentice Hall, 1999.
8. Hong, T. P., Lee, C. Y. Induction of fuzzy rules and membership functions from training examples. *Fuzzy Sets and Systems* 84. pp 33-47. 1996.
9. Ishibuchi, H. and Nakashima, T. "Effect of rule weights in fuzzy rule-based classification systems," *Proc. of 9th International Conference on Fuzzy Systems*, pp 59-64 (San Antonio, May 7-10, 2000).
10. Ishibuchi, H., Nakashima, T.: Effect of Rule Weights in Fuzzy Rule-Based Classification Systems, *IEEE Trans. on Fuzzy Systems*, vol. 9, no. 4, pp. 506-515, August 2001.
11. Nelder, J.A. and Mead, R., A simplex method for function minimization, Computer J., 7 (1965), 308-313.
12. Nauck, D. and Kruse, R. "How the learning of rule weights affects the interpretability of fuzzy systems". *Proc. of the 7th IEEE International Conference on Fuzzy Systems*, pp. 1235-1240 (Anchorage, May 4-9, 1998).
13. Pal, S. K., Mandal, D. P. "Linguistic recognition system based in approximate reasoning". *Information Sciences* **61**, pp. 135-161. 1992.
14. Prechelt, L. "PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms". Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
15. Sánchez, L., Casillas, J., Cordón, O., Del Jesus, M. J. "Some relationships between fuzzy and random set-based classifiers and models". Accepted for publication in IJAR, 2001.
16. Trillas, E., Alsina, C., Terricabras, J. *Introducción a la lógica borrosa*. Ariel Matemática. 1995.
17. Wang, L. X., Mendel, J. "Generating fuzzy rules by learning from examples". IEEE Trans. on Systems, Man and Cybernetics, **25**(2), pp. 353-361, 1992.
18. Zadeh, L.A. "The concept of a linguistic variable and its application to approximate reasoning". Information Science, Part I: vol. 8, pp. 199-249, 1975; Part II: vol. 8, pp. 301-357, 1975; Part III: vol. 9, pp. 43-80, 1975.
19. Zadeh, L.A. "Fuzzy Languages and Their Relation to Human and Machine Intelligence", in *Fuzzy Sets, Fuzzy Logic and Fuzzy Systems*, Klir, Yuan, eds. pp 148-179. World Scientific, 1996.