Soft Comput (2009) 13:467–479
DOI 10.1007/s00500-008-0362-4

FOCUS

Obtaining linguistic fuzzy rule-based regression models from imprecise data with multiobjective genetic algorithms

Luciano Sánchez · José Otero · Inés Couso

Published online: 27 August 2008

Abstract Backfitting of fuzzy rules is an Iterative Rule Learning technique for obtaining the knowledge base of a fuzzy rule-based system in regression problems. It consists in fitting one fuzzy rule to the data, and replacing the whole training set by the residual of the approximation. The obtained rule is added to the knowledge base, and the process is repeated until the residual is zero, or near zero. Such a design has been extended to imprecise data for which the observation error is small. Nevertheless, when this error is moderate or high, the learning can stop early. In this kind of algorithms, the specificity of the residual might decrease when a new rule is added. There may happen that the residual grows so wide that it covers the value zero for all points (thus the algorithm stops), but we have not yet extracted all the information available in the dataset. Focusing on this problem, this paper is about datasets with medium to high discrepancies between the observed and the actual values of the variables, such as those containing missing values and coarsely discretized data. We will show that the quality of the iterative learning degrades in this kind of problems, because it does not make full use of all the available information. As an alternative to sequentially obtaining rules, we propose a new multiobjective Genetic Cooperative Competitive Learning (GCCL) algorithm. In our approach, each individual in the population codifies one rule, which competes in the population in terms of maximum coverage and fitting, while the individuals in the population cooperate to form the knowledge base.

L. Sánchez (✉) · J. Otero
Computer Science Department, University of Oviedo,
Campus de Viesques, 33071 Gijón, Asturias, Spain
e-mail: luciano@uniovi.es

J. Otero
e-mail: jotero@uniovi.es

I. Couso
Statistics Department, Facultad de Ciencias,
University of Oviedo, 33071 Oviedo, Asturias, Spain
e-mail: couso@uniovi.es

1 Introduction

Since the last few years, there has been a growing interest in the use of imprecise data in intelligent systems. For instance, in Ferson et al. (2007), up to eight categories of real-world imprecise data suitable for a random sets based model are discussed: plus-or-minus reports, significant digits, intermitent measurement, non-detects, censoring, data binning, missing data and gross ignorance. Also, recent works in fuzzy random variables prop up using a fuzzy representation when the data is known through a family of confidence intervals (Couso and Sánchez 2008), including the preceding examples and others, like reconciling different measurements of a feature in a given object (Sánchez et al. 2007, 2009).

For the most part, the application of Genetic Fuzzy Systems (GFSs) to fuzzy data is still in their beginnings (Sánchez and Couso 2007). According to Herrera (2008), there are four genetic techniques for learning KB components: Pittsburgh, Michigan, Iterative Rule Learning (IRL) and Genetic Cooperative-Competitive Learning (GCCL). In previous works (Sánchez et al. 2006), we have extended our own implementations of backfitting and boosting, that can be considered part of the IRL approach, to fuzzy data (del Jesus et al. 2004; Sánchez and Otero 2004). In this paper, we will do the same with an algorithm of the GCCL type.

In particular, in Sánchez et al. (2006), we have shown that a fuzzy data-based IRL algorithm is efficient in problems where the observation error is small. That GFS adopted a fuzzy representation of the imprecise data, and (as a consequence of this) a fuzzy-valued accuracy. To balance accuracy and linguistic quality, the linguistic quality was measured by a real number, and we used a multicriteria Genetic Algorithm (GA) for jointly optimizing a mix of crisp and fuzzy objectives (Sánchez et al. 2007). The GA, in turn, was related to previous approaches about the use of fuzzy fitness functions (Koeppen et al. 2003) and precedence operators between imprecise values (Limbourg 2005; Teich 2001).

Nevertheless, in this paper we will show that IRL does not make full use of low quality data. As we will discuss later (see Sect. 4.2) backfitting-based algorithms cause the residual to be less specific when new rules are added. In certain cases, this prevents discovering enough rules. As an alternative, we propose a new implementation of a multiobjective Genetic Cooperative Competitive Learning (GCCL) algorithm (Herrera 2005), where each individual in the population codifies one rule, and the individuals in the population comprise the knowledge base. The individuals compete in a fitness space defined by the coverage and the fitting of the rules, and are assigned credit in the cooperation phase by means of a new algebraic procedure, related to the Singular Value Decomposition (Press et al. 1992).

The structure of this paper is as follows: in the next section we discuss certain issues about the representation of vague data that affect the performance of GFS. In Sect. 3 we explain the fuzzy reasoning method. In Sect. 4 we recall the use of backfitting techniques to learn fuzzy models from interval data. In Sect. 5 we describe the new algorithm, and benchmark its results in Sect. 6. The paper finishes with the concluding remarks, in Sect. 7.

2 Preliminaries: representation issues with low quality data

Although the use of intervals or fuzzy sets for codifying ranges or families of tolerances may seem trivial, there exist problems where the correspondence between a vague input and an interval, or a fuzzy set, is counterintuitive. As we will show later (see Sect. 6) a wrong representation guides the learning to an incorrect objective.

For example, imagine the following piece of vague information about a function $f: \mathbf{R} \rightarrow \mathbf{R}$: "The input value is missing, and the value of the output is 3." Following Ferson et al. (2007), we can represent the missing input by an interval spanning the whole range of the variable, and produce the granule $((-\infty, \infty), 3)$. Depending on how we calculate the fitting between the model and the data, this piece of information could be interpreted by the learning algorithm as "the function is constant and takes the value 3," while the actual information provided by the granule is other: "the value 3 is in the image of the function."

Let us consider another example: an input–output pair (1.5, 3), and a tolerance of the measurements in the input space of ± 0.5 units. We will assume that the measurements are accurate in the output space (see Figure 1). One might think in representing this information through the pair $(1.5 \pm 0.5, 3)$. But, if we included that granule of information in the training set, we could be stating that the output of the model must be 3 for all the values in 1.5 ± 0.5, and that is not true. What we really know is that there exist *at least one value* in the interval [1, 2] whose output is 3.

Hence, the proper representation of the granule of information in Fig. 1 is $(1.5 \pm 0.5, [3-\delta_1, 3+\delta_2])$, for certain values δ_1 and δ_2 that depend on the excursion of f in [1, 2]. Observe also that δ_1 and δ_2 are not related to the tolerance in the measurements of the output variable.

With these examples we want to highlight that, when the input data has low quality (high observation or quantization error, missing values, etc.) the output variables must be preprocessed to have their supports extended, before the learning begins. For the sake of clarity, in this paper we will label the datasets according to the relative uncertainties between inputs and outputs, as follows (see Fig. 2):

I. The uncertainty in the outputs is lower than the excursion of the function in the range of the inputs.
II. The uncertainty in the outputs is higher than the excursion of the function in the range of the inputs.
III. The uncertainty in the outputs matches the set of all the images of the values compatible with the input.

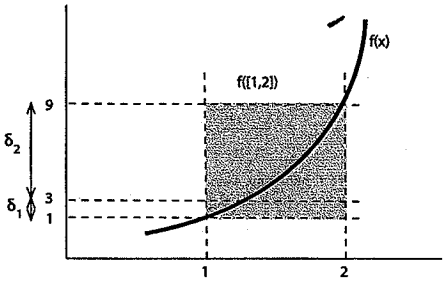

Fig. 1 The proper representation of the training example (1.5, 3) when the tolerance of the measurements in the input space is ± 0.5, is a pair $(1.5 \pm 0.5, [3-\delta_1, 3+\delta_2])$ for certain values δ_1 and δ_2 that depend on the excursion of f in [1, 2]

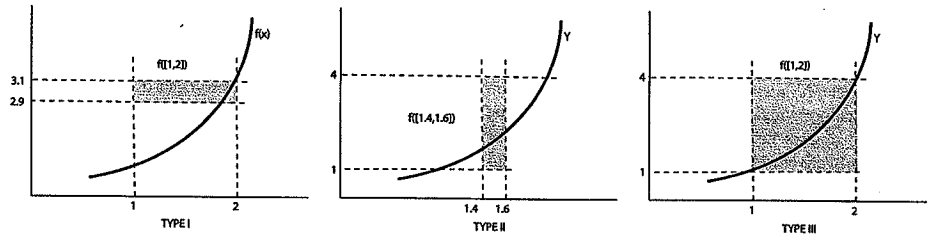

Fig. 2 Categories of vague datasets. We are given three granules of information about an unknown function $f(x) = x^2$. **a** The input is in the interval [1, 2], and the output is in [2.9, 3.1]; **b** the input is in [1.4, 1.6] and the output is in [1, 4]; **c** the input is in [1, 2] and the output is in [1, 4]. The granule (a) is of type I, because $f([1, 2]) \supset [2.9, 3.1]$. The granule (b) is of type II, because the $f([1.4, 1, 6]) \subset [1, 4]$. The granule (x) is of type III, because $f([1, 2]) = [1, 4]$

Problems of type III are rare. In point of fact, type I problems appear the most, since we seldom can quantify how the observation error in the inputs is transferred to the output variable. Nevertheless, the most convenient case (from the point of view of rule learning) is that of a type II dataset. We will restrict ourselves to this category of problems, because any dataset can be transformed into a type II problem without losing much information. For example, the transformation from an assert "There is one point in [1, 2] whose output is 3" into a sentence like "The output of the value 1.5 is in the interval [1, 4]" involves selecting a characteristic point of that input (the center point, for instance) and then extending the output until we are certain that it covers the actual output of the function at this characteristic point.

3 Fuzzy reasoning

In this paper, fuzzy knowledge bases comprise M fuzzy if—then rules of the following type:

$$\begin{aligned} &\text{Rule } m: \texttt{If } x_1 \texttt{ is } A_{m1} \text{ and} \ldots \text{and } x_n \texttt{ is } A_{mn} \\ &\qquad \texttt{then } y \texttt{ is } B_m \texttt{ with weight } w_m, \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ and y are, respectively, the input and the output values. $A_{m1}, \ldots, A_{mn}$ are linguistic labels or "or" combinations of labels, whose corresponding fuzzy sets are arranged in a prespecified fuzzy grid (that will not change during the learning). The consequents B_m can be either singletons or fuzzy numbers. For example:

$$\begin{aligned} &\text{Rule 1: If } x_1 \text{ is (SMALL or MEDIUM) and } x_2 \text{ is LARGE} \\ &\qquad \text{then } y \text{ is 3 with weight 0.8}, \end{aligned} \tag{2}$$

The rule base is interpreted according to the First Inference Then Aggregation (FITA) principle (Cornelis and Kerre 2003). Let us define the fuzzy set

$$A_m(x) = \min\{A_{m1}(x_1), \ldots, A_{mn}(x_n)\}. \tag{3}$$

Each fuzzy rule defines a function f_m,

$$f_m(x) = C_Y(I(A_m(x), B_m)) \tag{4}$$

where I is a fuzzy inference (we have used the product operator) and C_Y stands for "centroid",

$$C_Y(h(x, \cdot)) = \frac{\int_Y y h(x,y)\, dy}{\int_Y h(x,y)\, dy}. \tag{5}$$

The aggregated output is

$$g(x) = G_{m=1}^{M} w_m f_m(x) \tag{6}$$

The output of the rule base, for a fuzzy input X, is computed through the extension principle, and it is the fuzzy set

$$Y(y) = \max\{X(x) \mid y = g(x)\}. \tag{7}$$

Observe that the output of a crisp value x is also a crisp number $g(x)$, even though B_m was a fuzzy number.

4 Backfitting fuzzy models

If the aggregation operator G is the sum, then the expression in Eq. (6) describes a weighted sum of functions f_m, which can be regarded as an ensemble of models. The weights w_m of their members match the confidence degrees of the consequents of the corresponding rules.

The catalog of functions f_m is finite, because there is also a finite number of sets A_m and B_m (we have stated that the fuzzy grid of linguistic terms does not change during the learning). Therefore, the learning problem can be thought of as that of finding the vector of weights (one for each function in the catalog) that best approximates the dataset. However, the size of the catalog prevents us from storing the whole vector of weights in memory. We are not

interested in the optimal approximation, though. What we really want is to find the best *sparse* approximation to the optimal vector, i.e. that most of the weights are zero.

The *backfitting* algorithm [some of whose variants are called matching pursuit (Mallat and Zhang 1993), arcing and, arguably, boosting (Friedman et al. 1998)] is one of the most effective methods for finding a sparse set of weights. We discuss its crisp and fuzzy implementations in the sections that follow.

4.1 Backfitting in crisp datasets

For a crisp dataset

$$\{(x^i, y^i)\}_{i\in\{1,\ldots,N\}} \tag{8}$$

the backfitting algorithm recurrently calls a procedure that finds both a function f in the mentioned catalog, and a weight w minimizing the norm of the residual:

$$f = \arg\min \sum_{i=1}^{N} (y^i - wf(x_i))^2. \tag{9}$$

We proposed in Sánchez et al. (2006) that the best weight w is analitically determined,

$$w = \frac{\sum_{i=1}^{N} f(x^i)^2}{\sum_{i=1}^{N} y^i f(x^i)}, \tag{10}$$

and used a genetic procedure that searches for the list of sets $A_{m1},\ldots,A_{mn}$ and B_m, which, in combination with the value of w determined in Eq. (10), produces the best function f.

This generic procedure is wrapped in an iterative search, defined as follows: In a first step, we find the pair (w_1, f_1), as defined in Eq. (9), that best fits the dataset (8). The fuzzy rule associated to f_1 is added to the (empty) rule base, and assigned a weight w_1.

In a second step, the training set (8) is replaced by the residual of the approximation,

$$\{(x^i, y^i - wf(x^i))\}_{i\in\{1,\ldots,N\}} \tag{11}$$

and the procedure that finds the best f in the catalog is called again, to obtain f_2 and w_2:

$$f_2 = \arg\min \sum_{i=1}^{N} \left((y^i - w_1 f_1(x_i)) - w_2 f_2(x_i)\right)^2. \tag{12}$$

The fuzzy rule related to f_2 is added to the knowledge base, and the dataset is replaced by the new residual. Once we reach this point, this last step is repeated until the residual is small enough.

From a conceptual point of view, observe that replacing the output variable of the training set by the residual is numerically the same as assigning certain weights to the examples in the dataset. The weighted least squares problem, applied to the original dataset, is equivalent to the least squares approximation to the residual. These weights would range from a perfect fit (weight 0) to an uncovered example (weight 1), therefore the rationale of this process can also be explained as "fit a rule to the dataset, remove the examples that are well explained by this rule and repeat until all examples are explained." In other words, the backfitting is an IRL method.

In previous works, we have used a genetic algorithm for finding the function f, and tested the preceding search scheme for extracting fuzzy rules from crisp data in both classifiers and regression models (del Jesus et al. 2004; Otero and Sanchez 2006; Sánchez and Otero 2007). The method is accurate and has been shown to be as fast as some ad-hoc learning methods (Sánchez and Otero 2004). On the other side, when the datasets are imprecise the backfitting results may not be appropriate, as we explain in the next section.

4.2 Backfitting in interval and fuzzy datasets

If the dataset is interval-valued or fuzzy,

$$\{(X^i, Y^i)\}_{i\in\{1,\ldots,N\}} \tag{13}$$

we can not accurately compute the residual of the model. Following the ideas introduced in Sánchez and Couso (2007), we assume that the residual of the model is contained in the set

$$\bigoplus_{i=1}^{N} (Y^i \ominus w \otimes F(X_i))^2 \tag{14}$$

where

$$F(X)(y) = \max\{X(x) \mid y = f(x)\} \tag{15}$$

and

$$X^2(y) = \max\{X(x) \mid y = x^2\}. \tag{16}$$

The best function f must minimize the fuzzy valued function defined in Eq. (14), as we will show in Sect. 5.

According to our experimentation, the definition (14) is not well suited for Type I problems. Observe that

$$Y^i \ominus w \otimes F(X_i) \tag{17}$$

is less specific than Y^i. The uncertainty in the inputs is propagated to the residual, because of the right term of the fuzzy subtraction in Eq. (17). Each time the output data is replaced by the residual of a new rule, the situation is made worse. Eventually, the residual will cover the value zero in all the points, and the search ends. As we will discuss later, (see Sect. 5) a method that evaluates groups of rules at the same time (Pittsburgh, Michigan or GCCL) is potentially better than IRL for problems with fuzzy inputs.

5 A multiobjective GCCL algorithm for obtaining fuzzy regression models from imprecise data

Genetic Cooperative-Competitive Learning (GCCL) algorithms (Herrera 2005, 2008; Ishibuchi et al. 1999) encode the rule base in a subset of the population. Rather than iteratively adding new members to the ensemble, a GCCL algorithm evolves the whole ensemble. Hence, we do not longer have a growing rule base; rules can be replaced and exit the ensemble. Likewise, at each generation new weights are assigned to all rules. Since all the weights are assigned simultaneously, the specificity of the residual does not influence the learning. Besides, the weight of a member may change when a new function is added to the ensemble.

The algorithm that we propose is called Fuzzy Genetic Cooperative Competitive Learning, or FGCCL. It makes the individuals in the population to compete in terms of coverage and fitting, and also to cooperate building a rule base. The cooperation involves removing redundant rules, and also reweighing the others, so that the fitting between the rule base and the dataset is improved.

The pseudocode of the method is shown in Fig. 3. We reuse the same genetic operators and representation of the backfitting approach (Sánchez et al. 2006). The multiobjective genetic algorithm applied to generate nondominated subsets of rules is based on the NSGA-II (Deb et al. 2002). We have added a new stage to this genetic algorithm, to implement the cooperation phase where the redundant rules are removed, as mentioned. This new stage is called "Null Space Filter" because of reasons that will be made clear later.

It is emphasized that FGCCL is related to the symbiotic evolution of fuzzy systems described in (Juang et al. 2000). Symbiotic evolution also assumes that each individual in the population represents only a partial solution to the problem. Complete solutions are formed by combining several individuals. In the same way, our individuals compete for survival (in terms of coverage and fitting), and cooperate to form the final rule base. Nevertheless, our approach differs from symbiotic evolution, since we do not use specializations to assign the fitness, neither reinforcement learning, but assign credit to the different individuals in the population in a global stage. Furthermore, our assignment of the coverage criteria is similar to that of other GCCL algorithm, COGIN (Greene and Smith 1993). Conversely, our work is different from previous approaches in that (a) we will use two objectives when assigning credit to the rules (the coverage of the rule, and its fitting to the set of examples), and (b) one of the objectives is a fuzzy valued function, because the algorithm is intended to run on imprecise data.

```
Initialize P
Evaluate coverage and fitting in P
Null Space Filter P
Create Intermediate Population Q
while iter ≤ maxiter
    Evaluate coverage and fitting in P+Q
    Null Space Filter P+Q
    non dominated sort P+Q
    compute crowding distance P+Q
    P ← selection of P+Q
    Null Space Filter P
    non dominated sort P
    Create Intermediate Population Q
end while
Output the nondominated elements of P
```

Fig. 3 Pseudocode of the FGCCL algorithm

In the following sections we will explain this algorithm in detail, and give a comprehensive explanation of the following points:

1. The computation of the coverage and fitting of a rule.
2. How to extend the NSGA-II algorithm for finding nondominated sets of rules under two different criteria, one of which is crisp and the other fuzzy.
3. The "Null Space Filtering", that removes redundant rules and weights the members of the ensemble.

5.1 Fitness function: coverage and fitting of a rule

The fitness function of FGCCL has two components: the coverage and the fitting. The first component counts how many examples are explained by the rule, and the second is intended to measure how well the consequent of the rule fits the data.

To compute the coverage, each example is allocated at the most to one rule. Let us recall that our rule base comprises M rules, as mentioned in Sect. 3:

$$\text{Rule } m: \texttt{If } x_1 \texttt{ is } A_{m1} \text{ and} \ldots \text{and } x_n \texttt{ is } A_{mn} \quad \texttt{then } y \texttt{ is } B_m \texttt{ with weight } w_m. \tag{18}$$

The antecedent of a rule is associated to the fuzzy set

$$A_m(x) = \min\{A_{m1}(x_1), \ldots, A_{mn}(x_n)\}, \tag{19}$$

and let the vague dataset contain N pairs of fuzzy sets (X^i, Y^i). If the consequents B_m are fuzzy sets, then the example number i is allocated to the rule whose firing strength

$$\max_{x,y,t} \{\min\{X^i(x), Y^i(y)\} \mid t = A_m(x) \cdot B_m(y)\} \tag{20}$$

is higher. In case that B_m are singletons, the example is allocated to the rule whose antecedent has the higher degree of truth:

$$\max_{x,t} \{X^i(x) \mid t = A_m(x)\}. \tag{21}$$

The calculus of the second component, the fitting, is more troublesome. Since we are aggregating rules with the sum,

we cannot easily isolate the contribution of a single rule to the approximation error. We have assigned to each rule a "replacement error", defined as the increment of the error of the whole rule base when that rule is removed *and the remaining rules are assigned new weights*, with the procedure that we explain in the next section. For example, the replacement error of a duplicate rule must be zero.

Since the residual of the model is imprecisely known, we define the replacement error of the mth rule as the fuzzy set

$$E_m = \bigoplus_{i=1}^{N} (Y^i \ominus F(X_i))^2 \ominus (Y^i \ominus F_{-m}(X_i))^2 \tag{22}$$

where

$$F(X)(t) = \max\left\{ X(x) \mid t = \sum_k w_k f_k(x) \right\} \tag{23}$$

and

$$F_{-m}(X)(t) = \max\left\{ X(x) \mid t = \sum_{k \neq m} w'_k f_k(x) \right\} \tag{24}$$

where w_k are the weights of the rule base and w'_k are those of the reduced base.

5.2 Using NSGA-II for optimizing uncertain objectives

The population of rules is evolved by means of a genetic algorithm that is derived from NSGA-II (Deb et al. 2002). The main difference between the genetic algorithm in FGCCL and NSGA-II is the use of the Null Space Filter. Besides, the NSGA-II has to work with a non-standard set of operators in order to manage fuzzy criteria (Deb et al. 2002). We need to define new precedence (dominance) operators, an alternate nondominated sorting of the individuals, and a crowding distance between fuzzy values. These operators are detailed in the sections that follow.

5.2.1 Precedence under imprecise fitness

To determine whether one individual precedes another, it is needed to set up a procedure that, given two fuzzy observations X_1 and X_2 of two unknown fitness values x_1 and x_2, estimates whether the probability of $x_1 < x_2$ is greater than that of $x_1 \geq x_2$, thus $X_1 \prec X_2$. In this sense, the criteria that we pursue can be regarded as a special case of fuzzy ranking. However, we also want to find those cases where there is not statistical evidence in X_1 and X_2 that makes us to prefer one of them (thus $X_1 \| X_2$).

If a joint probability $P(x_1, x_2)$ was known, comparing two individuals would be a statistical decision problem, i.e. $X_1 \prec X_2$when

$$P(x_1 < x_2) > P(x_1 \geq x_2) \tag{25}$$

or

$$P(x_1 \geq x_2) < 1/2. \tag{26}$$

Nevertheless, a fuzzy set provides us with less information than this probability distribution. We will interpret a fuzzy membership as a possibility, i.e. an upper probability that dominates the probability of the crisp fitness. In other words, given a fuzzy value X, the information we have about the value x is limited to

$$X(x) = P^*(x) \geq P(x). \tag{27}$$

Hence, $X_1 \prec X_2$ when

$$P_*(x_1 < x_2) > P^*(x_1 \geq x_2) \tag{28}$$

and, applying that $P^*(A) = 1 - P_*(A^c)$, the expression (28) is reduced to

$$P^*(x_1 \geq x_2) < 1/2. \tag{29}$$

Since $P^*(\cdot)$ is a possibility,

$$P^*(A) = \max\{P^*(x) : x \in A\} \tag{30}$$

therefore, to decide whether $X_1 \prec X_2$, we check that

$$\max\{X_1(x_1) \cdot X_2(x_2) : x_1 < x_2\} \geq 1/2 \tag{31}$$

and also that

$$\max\{X_1(x_1) \cdot X_2(x_2) : x_1 \geq x_2\} < 1/2 \tag{32}$$

It is remarked that rejecting $X_1 \prec X_2$ does not imply that $X_2 \prec X_1$, because it may happen that Eq. (29) is higher than 1/2 and also that $P^*(x_1 < x_2) \geq 1/2$. In that case, we conclude that the fuzzy memberships X_1 and X_2 do not contain enough information to appreciate significant differences between them, i.e. $X_1 \| X_2$.

Lastly, the precedence between heterogeneous pairs (n_1, X_1) and (n_2, X_2) comprising an integer and a fuzzy number, is as follows:

$$(n_1, X_1) \preceq (n_2, X_2) \Leftrightarrow \begin{cases} n_1 < n_2 \text{ and } X_1 \preceq X_2 \\ n_1 \leq n_2 \text{ and } X_1 \prec X_2 \end{cases} \tag{33}$$

5.2.2 Non dominated sorting

Finding the best individual is a generalization of the precedence between fuzzy values seen before. Observe that we can bound the lower probability of the assert "the ith individual has the minimum fitness in the population" by

$$M_i = \prod_{j=1}^{M} P_*(x_i < x_j) \quad i \neq j$$

where $P_*(x_i < x_j) = 1 - \max\{X_1(x_1) \cdot X_2(x_2) : x_1 \geq x_2\}$. Sorting the population with respect to the fuzzy criterion is the same as ordering the values of M_i.

5.2.3 Crowding distance

We have chosen the Hausdorff distance between the mean values of both fitness values. The mean value of the fuzzy number X is an interval $[l(X), u(X)]$ (Dubois and Prade, 1987). Let $[l_\alpha, u_\alpha]$ be the α-cut of the fuzzy set X. Then,

$$u(X) = \frac{\int_0^1 \alpha u_\alpha d\alpha}{\int_0^1 u_\alpha d\alpha} \tag{34}$$

and

$$l(X) = \frac{\int_0^1 \alpha l_\alpha d\alpha}{\int_0^1 l_\alpha d\alpha}. \tag{35}$$

Given two intervals A and B, the Haussdorff distance is defined as

$$d_H(A, B) = \max\{|a_1 - b_1|, |a_2 - b_2|\}. \tag{36}$$

The crowding distance for a given individual is defined as the distance between the nearest (as defined by the Hausdorff metric) individual preceding it and the nearest individual following it. The first and the last individuals are assigned a high crowding distance. The meaning of 'precede', 'follow', 'first' and 'last' is given by the order defined by the values M_i mentioned before.

5.3 Cooperation–competition problem: null space filtering

Assuming that the expected value of the fitness function tends to increase during the genetic evolution, it is clear that the cooperation between the rules in the population arises when the sum of the fitness values of the individuals is comonotonic with the fitness of the rule base that they form. In our case, on the one side, the sum of the coverage values equates the number of points in the dataset that fire at least one rule, thus the genetic evolution tends to increase the number of explained examples; the cooperation arises in a natural way. On the other side, imagine a set of linearly dependent rules: the replacement error of any of the rules in the set is zero, because any rule can be removed without affecting the fitting. Obviously, this does not mean that we can remove all the rules in the set. We can not determine which rules are superfluous on the basis on the replacement error. Further processing of the fitting criterion is needed to achieve cooperation.

In point of fact, if the size of the population is large enough, there will be infinitely many different settings of the confidence degrees that produce an optimal fit, in the least squares sense. Since our purpose is to obtain the most compact rulebase, we have to choose between those different weight assignments in a way that the number of redundant rules is as high as possible, thus the size of the rule base is small. Resembling credit assignment in fuzzy classifier systems, where two identical rules may be assigned different credit (Ishibuchi et al. 1999), we want that most, but not all of the rules with zero replacement error are assigned a null weight.

Let us derive first the expression of the set of weights that minimizes the squared error. Let $A = [a_{nm}]$ be the matrix of the memberships of the antecedents of all rules in the population, at the modal points of the inputs, $a_{nm} = A_m(\overline{X}_n)$. Let $Y = [y_n]$, $y_n = \overline{Y}_n$ be a column vector with the centers of the desired outputs of the model, and let $W = [W_n]$ be another column vector formed by the weights of these rules, those that we want to obtain. The assignment of weights that minimizes the error (and therefore solves the cooperation problem) is

$$K = (A^t A)^{-1} A^t Y \tag{37}$$

provided that the rank r_A of A coincides with its number of columns, the number of individuals in the population. In most cases, r_A is lower than the population size, therefore $C = A^t A$ does not have inverse. The common solution to this problem is to apply a Singular Value Decomposition (SVD) of the matrix C, so that $C = U D V^t$, for canceling the eigenvalues of D lower than 10^{-6} times the highest, and take the inverse of the remaining ones, and by last define

$$K = [V \cdot (1/D) \cdot U^t] A^t Y. \tag{38}$$

This assignment solves the cooperation problem. We want to modify it for improving the competition part of the problem too. Observe that the SVD solution does not discard rules. For example, if the rule base has two identical rules, the definition in eq. (38) will assign them the same weight. In conflict with this, we prefer that one of them takes all the credit.

It is easy to purge the duplicated rules, but it is difficult to remove rules that are (almost) linear combination of others. The value of K in the preceding equation is that of minimum norm of the weight vector, but what we really need is the definition of the matrix $(1/D)$ that produces *the most sparse* definition of K. In fact, the number of individuals we want to have assigned weights different than zero is the same as the number of not cancelled eigenvalues in D. Observe that, the columns of the matrix U associated to null eigenvalues form a basis of the nullspace of A. That means that each individual (each column of A), if expressed in the base formed by the columns of U, will have at most r_A non-null coefficients. In other words, we will not find more than r_A independent elements in the population. Therefore, we know that we can set to zero the weights of all the rules but r_A. The problem here is, how can we determine which columns of A can be zeroed without losing accuracy.

The solution is given by the stage that we call "Null Space Filtering": we compute the eigenvalues of $C' = (A')^t$

A', for all the submatrices A' of A formed by removing only one of its columns. When it is found a submatrix A' whose corresponding non null eigenvalues are the same as those of A, the column is removed and the process restarted from A'. We will end up with a matrix with r_A columns and full rank. Now it is possible to use the pseudoinverse solution in Eq. (37) to obtain the weights of the r_A rules in the base, assigning a weight zero to a number of rules equal to the dimension of the null space of A.

6 Numerical analysis

In this section we include some numerical experiments, to provide statistical evidence supporting the main claims of this paper. These claims can be summarized in the following four points:

1. IRL-based learning from Type-I vague datasets is prone to overtrain (see Sect. 6.1).
2. The FGCCL algorithm makes better use of vague datasets than IRL in problems with moderate and high observation error (Sect. 6.2).
3. The FGCCL algorithm is not worse than other fuzzy rule learning algorithms in the literature when the nature of the uncertainty is stochastic (Sect. 6.3).
4. The "Null Space Filter" stage in the FGCCL algorithm permits discovering more compact rule bases than IRL (Sect. 6.4).

Nine laboratory problems and four real world problems have been used to benchmark the algorithms proposed in this paper (see Table 1). The laboratory problems are:

- f_1: high slope function $f_1(x, y) = x^2 + y^2$, no noise added (Sánchez et al. 2002). $f_1 - 10$, $f_1 - 20$, $f_1 - 50$ are the same function, with 10, 20 and 50% of gaussian noise.
- f_2: low slope function $f_2(x, y) = 10(x-xy)/(x-2xy+y)$, no noise added (Sánchez et al. 2002). $f_2 - 10$, $f_2 - 20$, $f_2 - 50$ are the same function, with 10, 20 and 50% of gaussian noise.
- *Friedman*: Synthetic benchmark dataset proposed in (Friedman 1991) $f(x, y, z, t, u) = 10 \sin(\pi xy) + 20(z - 0.5)^2 + 10t + 5u$, with added gaussian noise.

The real world problems are:

- *elec* Relationship between the population and the radius of a village and the length of its electrical line (Cordón et al. 1999).
- *machine-CPU* Relative performance of CPUs based on other computer characteristics (Ein-Dor and Feldmesser 1987).
- *daily-elec* Daily price of electrical energy in Spain, in 2003, as a function of the technology mix (Alcala et al. 2008; Marín and Sánchez 2004).
- *building* problem taken from the benchmark (Prechelt 1994), with eight binary and six continuous input variables.

In all experiments, 50% of the points were used to train the models, which were tested against the remaining 50%. The roles of training and test sets were interchanged and the process repeated, and this was replicated five times for different permutations of the dataset, which gives ten repetitions of the learning algorithm for each dataset. The results shown are the mean test values of ten repetitions of the experiments. The parameters used in the genetic learning are summarized in Table 2.

6.1 Overtraining in Type-I datasets

In Sect. 2 we explained that vague data in Type-I datasets is easily misrepresented. If the uncertainty in the output is lower than the excursion of the function in the range of values of the input, the learning algorithm will learn a different function than expected. This effect will be perceived as an overtraining, i.e. reducing the train error causes the test error to increase.

In Table 3 we have quantified the importance of this problem. We have computed the test error of the FGCCL algorithm for the synthetic problems f_1 and f_2 (and their noisy versions) and the real world problem "elec" for interval-valued input data, with tolerances ±1, ±2 and ±3% in the input variables, and assuming that the output

Table 1 Properties of the datasets used in the numerical analysis

Name	Inputs	Examples
f_1	2	675
$f_1 - 10$	2	675
$f_1 - 20$	2	675
$f_1 - 50$	2	675
f_2	2	675
$f_2 - 10$	2	675
$f_2 - 20$	2	675
$f_2 - 50$	2	675
elec	2	490
Friedman	5	1200
machine-CPU	6	209
daily-elec	6	365
building	14	4208

f_1, f_2 and Friedman are synthetic datasets. $f_1 - *$ and $f_2 - *$ contain different amounts of stochastic noise. The remaining problems contain real-world data

Table 2 Parameters used in computational experiments

	FGCCL	BFT (IRL)
Evolutionary scheme	NSGA-II w/ fuzzy fitness + Null Space Filter (Sect. 5.3)	NSGA-II with fuzzy fitness (Sánchez et al. 2007)
Coding scheme	DNF fuzzy rules, binary codification, fixed length (Sánchez and Otero 2004)	
Fitness: first criterion	Fitting (Sect. 5.1)	Fuzzy MSE on the residual (Sánchez et al. 2006)
Fitness: second criterion	Coverage (Sect. 5.1)	N/A
Crossover probability	0.9	0.9
Tournament size	Binary tournament	Binary tournament
Mutation probability	0.02	0.02
Population size	10–25	50
Evaluations of fitness	1,000–2,000 Generations	10–30 × 100 Generations

BFT is an IRL algorithm, where the GA is lauched once for each rule. FGCCL obtains the whole population in a single execution of the GA. Depending on the problem, the GA in BFT is launched between 10 and 30 times, and then stopped after 100 generations. The GA in FGCCL has a small population (10–25 individuals) in order to limit the number of rules and the time needed to process the null space filter described in Sect. 5.3. It is remarked that, although the number of fitness evaluations in FGCCL is lower than that of BFT, the overhead imposed by the null space filter makes that the computing time of GCCL is roughly four times higher than that of BFT

Table 3 Test error when the input data has tolerances ±1, ±2 and ±3%

	0%	1%	2%	3%
f_1	0.13	0.22	1.34	**5.30**
$f_1 - 10$	1.59	1.67	2.75	**6.69**
$f_1 - 20$	5.98	6.08	7.17	**11.05**
$f_1 - 30$	14.37	14.53	15.76	**19.92**
$f_1 - 50$	39.22	39.30	40.31	**44.16**
f_2	0.22	0.23	0.34	0.76
$f_2 - 10$	0.35	0.37	0.50	0.93
$f_2 - 20$	0.86	0.88	1.00	1.43
$f_2 - 30$	1.40	1.42	1.55	1.97
$f_2 - 50$	3.69	3.71	3.83	4.25
elec × 10^{-3}	416	416	416	**890**

When the observation error is combined with a high slope in the function (function f_1 and "elec" problem,) the learning algorithm overtrains. The abnormal values are shown in boldface

Table 4 Comparison of FGCCL and backfitting in datasets with 1, 5 and 10% of interval-valued imprecision in the outputs, and tested with crisp data with non-zero mean observation error

	1%		5%		10%	
	BFT	FGCCL	BFT	FGCCL	BFT	FGCCL
f_1	0.89	**0.35**	6.64	**6.25**	24.82	**24.77**
$f_1 - 10$	2.66	**1.86**	9.39	**8.31**	29.03	**28.60**
f_2	0.52	**0.23**	0.60	**0.47**	1.41	**1.23**
$f_2 - 10$	0.56	**0.37**	0.97	**0.68**	**1.67**	1.70
elec	440	**421**	581	**558**	1003	**988**

FGCCL improved the results in all the tests for which the observation error was the most relevant source of noise

variable can be precisely observed. Each granule of information comprises (wrongly) two interval-valued inputs and a crisp output. Under these circumstances, when the observation error is combined with a high slope in the function (function f_1 and "elec" problem,) an overtraining happens, as can be seen in the column "3%".

6.2 Comparison between FGCCL and IRL in imprecise datasets

To compare FGCCL and IRL in imprecise datasets, the functions f_1 and f_2 were chosen. We want to show that the improvements of GCCL are more significant in f_1 (because its slope magnifies the effect of the imprecision in the inputs) than they are in f_2. Consequently, we have prepared datasets with 1, 5 and 10% of interval-valued imprecision in the outputs, superimposed on different amounts of stochastic uncertainty.

All the obtained rule bases have been tested with crisp data, assuming a biased (non-zero mean) observation error: all the test points were in the upper extreme of the tolerance. Notice that we have only used datasets for which the stochastic error is lower than the observation error.

The results are shown in Table 4. As expected, GCCL improved the results in the tests for which the observation error was the most relevant source of noise, almost uniformly.

6.3 Influence of the stochastic noise

The third benchmark checks whether FGCCL can be applied to crisp problems. We have compared the accuracy

Table 5 Comparative test error of FGCCL and the synthetic datasets f_1 and f_2 (reproduced from reference Sánchez et al. 2006)

	WM1	WM2	WM3	CH1	CH2	CH3	NIT	LIN	QUA	NEU	TSK	BFT	BMO	GCCL
f_1	5.65	5.73	5.57	5.82	8.90	6.93	5.63	130.5	**0.00**	0.35	0.095	0.327	0.30	**0.14**
f_1-10	6.89	7.19	6.54	6.84	10.15	8.20	7.16	133.91	**1.40**	1.78	1.62	1.86	1.71	**1.65**
f_1-20	11.07	10.99	11.06	11.33	13.45	12.42	10.63	135.6	**5.29**	6.42	5.90	6.04	5.98	**5.89**
f_1-50	51.78	46.40	47.80	53.48	48.94	48.16	39.65	166.64	**33.53**	41.18	36.76	39.62	**38.66**	**38.66**
f_2	0.41	0.48	0.45	0.40	0.59	0.45	0.43	1.54	1.61	1.48	**0.15**	0.24	0.26	**0.25**
f_2-10	0.64	0.68	0.68	0.59	0.68	0.60	0.58	1.71	1.75	1.81	**0.29**	0.42	0.41	**0.38**
f_2-20	1.27	1.16	1.17	1.29	1.15	1.17	0.97	2.04	2.09	0.90	**0.76**	0.87	**0.87**	0.90
f_2-50	4.34	3.98	3.94	4.47	3.90	3.97	**3.59**	4.67	4.78	3.76	3.62	3.67	3.72	3.70

The algorithms include: heuristics (WM, CH), statistical regression (LIN, QUA), neural networks (NEU), rules with a real-valued consequent (NIT), TSK rules (TSK), genetic backfitting (BFT) and MOSA-based backfitting (BMO). BFT, BMO and GCCL runs were limited to 25 fuzzy rules. The best of WM, CH, NIT, BFT, BMO and GCCL, plus the best overall model, were highlighted for every dataset

of FGCCL with the selection of learning algorithms in Sánchez et al. (2006). We have intentionally dropped from that list the real-world datasets[1], and focused in the synthetic datasets, whose degree of contamination with stochastic noise is known. It is emphasized that, in this case, we have assumed that the observation error is zero, since none of the algorithms to which we will compare ours can use vague data.

The fuzzy rule learning algorithms in Sánchez et al. (2006) are: Wang and Mendel (1992) with importance degrees 'maximum', 'mean' and 'product of maximum and mean' (WM1, WM2 and WM3, respectively) the same three versions of Cordón and Herrera's method (Cordón and Herrera 2000) (CH1, CH2, CH3). Nozaki, Ishibuchi and Tanaka's fuzzy rule learning (Nozaki et al. 1997) (NIT), TSK rules (Takagi and Sugeno 1985) optimized with Weighted Least Squares and Genetic Backfitting (Sánchez and Otero 2004), and MOSA-based backfitting (Sánchez and Villar 2008; Sánchez et al. 2006) (BMO). The same reference includes Linear Regression (LIN), Quadratic Regression (QUA) and a Conjugate-Gradient trained Multilayer Perceptron (NEU). This 13 algorithms have been compared to the Genetic Cooperative Competitive FGCCL.

The best overall result and the most accurate fuzzy rule base are boldfaced in Table 5. We have also included a selection of boxplots in Fig. 4. These provide a graphical insight of the relevance of the differences between the median, the mean and the variance of the results. Observe that the performances of the heuristic algorithms are always worse than those of the GFS. There are not, in general, significant differences in accuracy between GFS, statistical nonlinear regression and neural networks, although there seems to be a small improvement in accuracy of FGCCL over the remaining GFS. The significance of the difference is never statistically sound (95% level). Observe that, if we had used the results in the first column of Table 3 (which were obtained after a different set of ten independent runs of the algorithm) the conclusions are the same.

However, the most important gain, if FGCCL is to be used in crisp datasets, is not the improved accuracy but the compactness of the rule base, as we show in the next section.

6.4 Performance of GCCL over machine learning databases

The fourth benchmark uses standard Machine Learning benchmarks to assess the accuracy and the compactness of the FGCCL algorithm, in problems with different sizes and number of inputs. In order to keep the discussion simple, we have chosen a selection of the algorithms in the preceding section. CH1, CH2 and CH3 were discarded because they produced a much higher number of rules than WM, without a noticeable increment of the accuracy. Also, since we are measuring the linguistic quality by the number of rules, without taking into account the complexities of the consequent part of the rules, we discard NIT, because it is less accurate than weighted linear squares-based TSK, for the same number of fuzzy rules. The quadratic regression is also removed, because most of times the corresponding polynomial has more terms than the multilayer perceptron needed for obtaining an equivalent accuracy.

Summarizing, linear regression and the neural network are left for reference purposes. Other than this, we have compared: one example-guided heuristic method (Wang and Mendel algorithm), one TSK grid-based algorithm (we fitted with weighted least squares a plane to each set of elements covered by one cell of the fuzzy grid; the weights

[1] The results of FGCCL on the two dropped problems *building* and *elec* can be found in Table 6. In both cases, FGCCL is better than all GFSs in reference (Sánchez et al. 2006), but the difference is not relevant in this context.

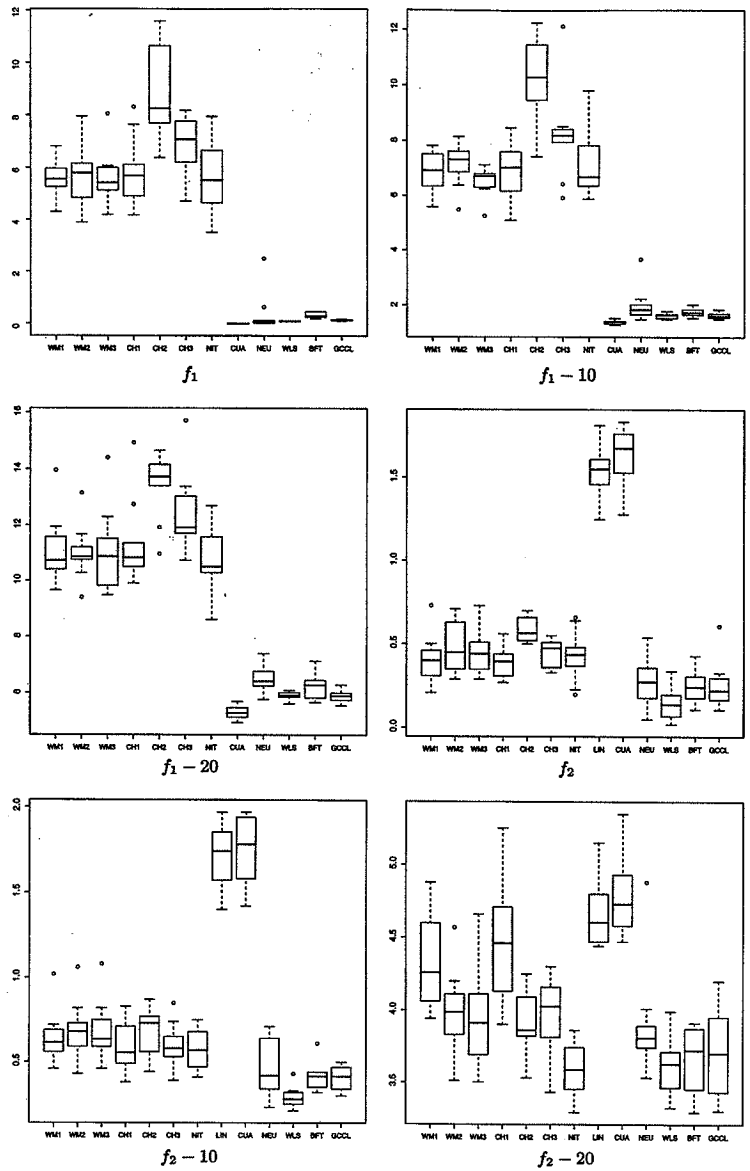

Fig. 4 Statistical relevance of the differences between FGCCL and the selection of statistical and fuzzy rule-based regression algorithms found in Sánchez et al. (2006), in the datasets f_1 and f_2. The performance of the heuristic algorithms is worse than that of the GFS. In turn, there are not, in general, significant differences in accuracy between GFS, statistical nonlinear regression and neural networks

Table 6 Performance of a selection of statistical models, heuristic rule learning and GFS in some benchmarks

	Statistical		Fuzzy rule learning			
	LIN	NEU	WM	WLS-TSK	BFT	FGCCL
elec $\times 10^{-3}$	419	616	832	485	444	**399**
machine-CPU	**6204**	8955	18057	7533	17857	9536
daily-elec	**0.171**	0.195	0.305	0.179	0.224	0.196
Friedman	7.33	1.22	7.11	1.80	2.28	**1.61**
building $\times 10^3$	4.77	2.76	4.44	**2.68**	3.15	2.98

The most accurate fuzzy rule learning algorithm is Weighted Least Squares TSK, followed by FGCCL

Table 7 The number of fuzzy rules generated by FGCCL is much lower than those produced by either example-guided or grid-guided learning methods

	Labels	WM	WLS-TSK	BFT	FGCCL
elec	3	7	8	10	**4**
machine-CPU	3	20	91	25	**4**
daily-elec	3	64	427	25	**5**
Friedman	3	192	242	25	**10**
building	3/2*	789	896*	30	**20**

FGCCL improves grid-based learning algorithms by one order of magnitude in the number of parameters. We have used three linguistic labels in all the variables (but the discrete ones). In the dataset "building", the grid-based algorithm depends on partitions of size 2. Otherwise, the number of rules is too high for practical purposes

are the memberships of the elements to the cell), an IRL method (genetic backfitting) and Fuzzy GCCL. The results are given in Tables 6, 7, and Fig. 5.

As expected, the example-guided heuristic uses a moderately high number of rules, and it is not very precise (equivalent or worse than linear regression). The weighted linear squares obtention of TSK rules is the most precise method, similar to that of the neural network. Backfitting and FGCCL both perform well and offer a good compromise between compacteness and accuracy. FGCCL was significantly better than backfitting in four of five tests, where the fuzzy knowledge bases it generated were both more compact and more precise.

7 Concluding remarks

This paper addressed some problems of learning fuzzy rules from imprecise data. We have proposed a new Genetic Cooperative Competitive Learning algorithm, designed to make full use of low quality data, and shown that this paradigm has inherent advantages over IRL when the inputs are vague. Our algorithm includes an algebraic stage, where the redundant rules are filtered. We have demonstrated that this stage contributes to obtain compact rulebases. We have also shown that an FGCCL algorithm has good properties when applied to crisp data.

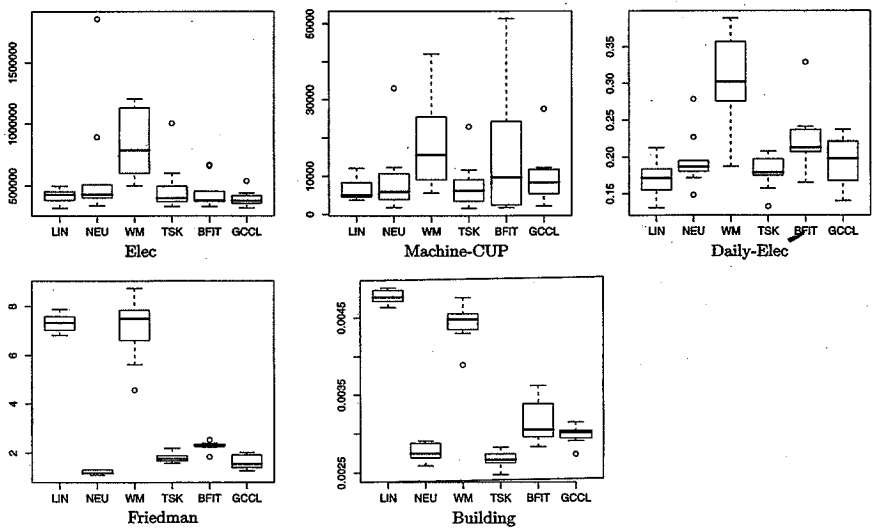

Fig. 5 Compared accuracy between Linear Regression, Neural Networks, Wang and Mendel, TSK rules, Genetic Backfitting and FGCCL. The accuracy of FGCCL is better, in general, than that of backfitting, and the differences are statistically relevant in four of the five datasets

Besides, there are some basic questions that remain unanswered, such as the best way of preprocessing a dataset with a high degree of imprecision in the input. In particular, it would be convenient to know how to compute the spread of the output when the input is shrunk to one point (generally speaking, how to convert type I problems into type II). In this paper we have benchmarked the algorithm over type II problems, but future works should emphasize the learning of the more common type I problems, including those with totally unknown input values (i.e., missing data).

On a different subject, there is room for certain improvements in the computer algorithm of FGCCL. It is worth mentioning that the initialization of the population is currently random. Hence, it may happen that some rules in the initial population do not cover any examples. This slows the convergence in high dimensional problems. We are currently studying the initialization of the population with a heuristic rule learning algorithm.

Acknowledgments This work was supported by the Spanish Ministry of Education and Science, under grants TIN2005-08036-C05-05 and TIN2005-08036-C05-01.

References

Alcala J et al (2008) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput (in press)
Cordón O, Herrera F (2000) A proposal for improving the accuracy of linguistic modeling. IEEE Trans Fuzzy Syst 8(3):335–344
Cordón O, Herrera F, Sánchez L (1999) Solving electrical distribution problems using hybrid evolutionary data analysis techniques. Appl Intell 10(1):5–24
Cornelis C, Kerre E (2003) A fuzzy inference methodology based on the fuzzification of set inclusion. In: Recent advances in intelligent paradigms and applications, Physica-Verlag, pp 71–89
Couso I, Sánchez L (2008) Higher order models for fuzzy random variables. Fuzzy Sets Syst 159:237–258
Deb K, Pratap A, Agarwal S, Meyarevian T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
del Jesus MJ, Hoffmann F, Junco L, Sánchez L (2004) Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Trans Fuzzy Syst 12(3):296–308
Dubois D, Prade H (1987) The mean value of a fuzzy number. Fuzzy Sets Syst 24(3):279–300
Ein-Dor P, Feldmesser J (1987) Attributes of the performance of central processing units: a relative performance prediction model. Commun ACM 30(4):308–317
Ferson S, Kreinovich V, Hajagos J, Oberkampf W, Ginzburg L (2007) Experimental uncertainty estimation and statistics for data having interval uncertainty. Technical Report SAND2007-0939, Sandia National Laboratories
Friedman J (1991) Multivariate adaptive regression splines. Ann Stat 19:1–141
Friedman J, Hastie T, Tibshirani R (1998) Additive logistic regression: a statistical view of boosting. Mach Learn
Greene DP, Smith SF (1993) Competition-based induction of decision models from examples. Mach Learn 3:229–257
Herrera F (2005) Genetic fuzzy systems: status, critical considerations and future directions. Int J Comput Intell Res 1(1):59–67
Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol Intell 1:27–46
Ishibuchi H, Nakashima T, Murata T (1999) Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans Syst Man Cybern Cybern 29(5):601–618
Juang CF, Lin JY, Lin CT (2000) Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. IEEE Trans Syst Man Cybern B Cybern 30(2):290–302
Koeppen M, Franke K, Nickolay B (2003) Fuzzy-Pareto-dominance driven multi-objective genetic algorithm. In: Proceedings of 10th international fuzzy systems assotiation world congress (IFSA), Istanbul, pp 450–453
Limbourg P (2005) Multi-objective optimization of problems with epistemic uncertainty. EMO 2005:413–427
Mallat S, Zhang Z (1993) Matching pursuits with time-frequency dictionaries. IEEE Trans Signal Process 41:3397–3415
Marín E, Sánchez L (2004) Supply estimation using coevolutionary genetic algorithms in the Spanish electrical market. Appl Intell 21(1):7–24
Nozaki K, Ishibuchi H, Tanaka H (1997) A simple but powerful heuristic method for generating fuzzy rules from numerical data. Fuzzy Sets Syst 86:251–270
Otero J, Sanchez L (2006) Induction of descriptive fuzzy classifiers with the Logitboost algorithm. Soft Comput 10(9):825–835
Prechelt L (1994) PROBEN1—a set of benchmarks and benchmarking rules for neural network training algorithms. Tech. Rep. 21/94, Fakultat fur Informatik, Universitat Karlsruhe
Press W et al (1992) Numerical recipes in C. The art of scientific computing. Cambridge University Press, New York
Sánchez L, Couso I (2007) Advocating the use of imprecisely observed data in genetic fuzzy systems. IEEE Trans Fuzzy Syst 15(4):551–562
Sánchez L, Otero J (2004) A fast genetic method for inducting descriptive fuzzy models. Fuzzy Sets Syst 141(1):33–46
Sánchez L, Otero J (2007) Boosting fuzzy rules in classification problems under single-winner inference. Int J Intell Syst 22(9):1021–1034
Sánchez L, Villar JR (2008) Obtaining transparent models of chaotic systems with multiobjective simulated annealing algorithms. Inform Sci 178(4):952–970
Sánchez L, Casillas J, Cordón O et al (2002) Some relationships between fuzzy and random set-based classifiers and models. Int J Approx Reason 29(2):175–213
Sánchez L, Otero J, Villar JR (2006) Boosting of fuzzy models for high-dimensional imprecise datasets. In: Proceedings of IPMU 2006, Paris, pp 1965–1973
Sánchez L, Couso I, Casillas J (2007) Modelling vague data with genetic fuzzy systems under a combination of crisp and imprecise criteria. In: Proceedings of 2007 IEEE symposium on Computational Intellignece in multicriteria decision making, Honolulu, pp 30–37
Sánchez L, Couso I, Casillas J (2009) Genetic learning of fuzzy rules based on low quality data. Fuzzy Sets Syst (submitted)
Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132
Teich J (2001) Pareto-front exploration with uncertain objectives. EMO 2001:314–328
Wang LX, Mendel J (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 25(2):353–361