Evol. Intel. (2009) 2:73–84
DOI 10.1007/s12065-009-0024-1

SPECIAL ISSUE

Extending a simple genetic cooperative-competitive learning fuzzy classifier to low quality datasets

Ana M. Palacios · Luciano Sánchez · Inés Couso

Received: 18 June 2009 / Revised: 23 September 2009 / Accepted: 25 September 2009 / Published online: 24 October 2009

Abstract Exploiting the information in low quality datasets has been recently acknowledged as a new challenge in Genetic Fuzzy Systems. Owing to this, in this paper we discuss the basic principles that govern the extension of a fuzzy rule based classifier to interval and fuzzy data. We have also applied these principles to the genetic learning of a simple cooperative-competitive algorithm, that becomes the first example of a Genetic Fuzzy Classifier able to use low quality data. Additionally, we introduce a benchmark, comprising some synthetic samples and two real-world problems that involve interval and fuzzy-valued data, that can be used to assess future algorithms of the same kind.

Keywords Genetic fuzzy systems · Vague data · Fuzzy-knowledge-based rules · Cooperative-competitive learning · Possibilistic data

1 Introduction

Fuzzy data is the main object of study in fuzzy statistics [3], but this kind of information is seldom considered in Genetic Fuzzy Systems (GFS) [2, 6]. Indeed, GFSs obtain Fuzzy Rule Based Systems (FRBS) from data, but the role of fuzzy sets in a FRBS is to model vague asserts, using fuzzy logic-based tools. Fuzzy logic techniques are not, generally speaking, compatible with those fuzzy statistical techniques used for modeling vague observations of variables. As a consequence of this, most GFSs can only extract FRBS from crisp data [13].

To this we should add that there are many different interpretations of the meaning of a fuzzy membership [11], thus there are many different approaches for relating "fuzzy data" and "low quality data". In this paper, we are choosing a possibilistic interpretation, because it matches a large amount of practical situations. This consists in understanding a fuzzy membership function as a nested family of sets (see Fig. 1), each one of them containing the true value of the variable with a probability greater or equal than certain bound [3]. For instance, it can be used to model datasets with missing values (one interval that spans the whole range of the variable), left and right censored data (the value is greater or lower than a cutoff value, or it is between a couple of bounds), compound data (each item comprises a disperse list of values), mixes of punctual and set-valued measurements (as produced by certain sensors, for instance GPS receivers) etc. All these cases share in common a certain degree of ignorance about the actual value of a variable, and assume less prior knowledge than the standard model, thus we will refer to all of them with the generic term "low quality data".

In this paper, we will devise an augmented GFS that can operate with low quality (possibilistic) data. In our context, this means that we want a classifier that is able to operate when we cannot accurately observe all the properties of the object. In the most simple case (interval-valued data) we will perceive sets that contain these values. In the general case, we will be given a nested family of sets, each one of them containing the true value with a probability greater or equal than its level.

A. M. Palacios · L. Sánchez (✉)
Departamento de Informática, Universidad de Oviedo,
33071 Gijón, Asturias, Spain
e-mail: luciano@uniovi.es

A. M. Palacios
e-mail: apalaciosjimenez@gmail.com

I. Couso
Departamento de Estadística e I.O. y D.M,
Universidad de Oviedo, 33071 Gijón, Asturias, Spain
e-mail: couso@uniovi.es

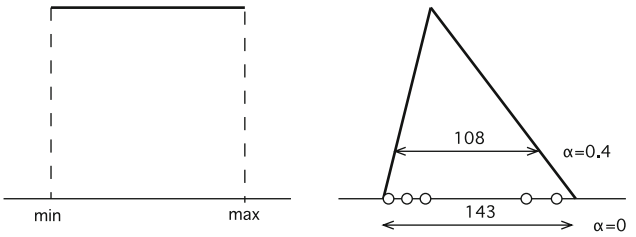

Fig. 1 Fuzzy representation of vague data. *Left* A missing value is codified with an interval that spans the whole range of the variable, or $P([\min, \max]) \leq 1$. *Right* A compound value (in this example, five different measurements of the variable) can be described by a fuzzy membership, that can also be understood as an upper probability. Each α-cut contains the true value of the variable with probability at least $1 - \alpha$

In the following sections we will study how this kind of data can be routed through a FRBS to produce a set of outputs, how can we measure the performance of a FRBS with this data, and genetically optimize it. In Sect. 2 we will propose a new reasoning method that is compatible with the possibilistic view of the imprecision in the data. In Sect. 3 we will review the algorithm that is being generalized, and propose different extensions for some of its modules. In Sect. 4 we will apply the new algorithm to different synthetic and real-world problems, and compare the results to those of the crisp algorithm. In Sect. 5 we conclude the paper and discuss future work in the subject.

2 An extension principle-based reasoning method

In this section we discuss how to compute the output of an FRBS, given a vague input. At a first glance, this should consist in computing the cylindrical extension of the input, intersecting it with the fuzzy graph implicitely defined by the FRBS, and projecting this intersection on the output space.

However, this reasoning method does not preserve the possibilistic meaning of the data. That is to say, it may happen that, given an input that has a possibilistic meaning, we come out with an output that has not that kind of interpretation. In order to obtain meaningful results, in this section we adapt a reasoning method, that was proposed in [17] for fuzzy models, to the classification case.

Let us make clear the problem with the help of a particular case; consider a fuzzy classifier comprising M rules:

$$\textbf{if } (x \textbf{ is } \widetilde{A}_i) \textbf{ then class is } C_i, \tag{1}$$

and let us use the single-winner inference mechanism. In the first place, let us suppose that we have a crisp perception x of the properties of an object. Its class is, therefore,

$$\text{class(x)} = C_{\text{argmax}_i\{\widetilde{A}_i(x)\}}. \tag{2}$$

In the second place, let the object be imprecisely observed, thus all our information is "$x \in X$." If we apply the fuzzy logic based approach mentioned before, the class of the object is still a singleton:

$$\text{class}'(X) = C_{\text{argmax}_i\{\min\{\widetilde{A}_i(x)|x\in X\}\}} \tag{3}$$

which is not the result we need. We want to obtain the set of labels that follows:

$$\text{class(X)} = \{C_{\text{argmax}_i\{\widetilde{A}_i(x)\}} \mid x \in X\} \tag{4}$$

or, in other words,

$$\text{class(X)} = \{\text{class}(x) \mid x \in X\}. \tag{5}$$

which is different than Eq. 3.

To solve this discrepancy, we propose to use the reasoning method that follows: Let X be the input space, let N_c be the number of classes, thus $K = \{1, ..., N_c\}$ is the output space, and let $\{\widetilde{A}_i \rightarrow C_i\}_{i=1,...,M}$ be a set of M fuzzy rules. Recall that, given a crisp input $x \in X$, the most common reasoning method for computing the output of a FRBS takes two stages [2]:

1. An intermediate fuzzy set is composed:

$$\widetilde{\text{out}}(x)(k) = \max_{i=1,...,M} \min\{\widetilde{A}_i(x), \delta^k_{C_i}\}, \tag{6}$$

where $\delta^k_{C\_i} = 1$ if $C_i = k$, 0 else.

2. This intermediate fuzzy set is transformed in a crisp value $\text{defuz}(\widetilde{\text{out}}(x)) \in K$ by means of a suitable defuzzification operator. In classification problems, the "maximum" defuzzification is mostly used. Therefore, the value $\text{defuz}(\widetilde{\text{out}}(x)) \in K$ is often equivalent to

$$\text{defuz}(\widetilde{\text{out}}(x)) = \arg\max_k \{\widetilde{\text{out}}(x)(k)\}. \tag{7}$$

The extension to set valued inputs is as follows: Given an input $A \subseteq X$ (that, in our context, means "all we know about the input is that it is in the set A"),

1. We determine a family of intermediate fuzzy sets in the universe $\mathcal{F}(K)$, $\widetilde{\text{out}}(A) \in \wp(\mathcal{F}(K))$—where $\wp(\mathcal{F}(K))$ is the set of all crisp subsets of $\mathcal{F}(K)$—as

$$\widetilde{\text{out}}(A) = \{\widetilde{\text{out}}(x) \text{ s. t. } x \in A\} \tag{8}$$

2. An element of $\wp(K)$ (that is to say, a set of crisp outputs $\text{defuz}(\widetilde{\text{out}}(A)) \in \wp(K)$) is obtained, according to the following definition:

$$\text{defuz}(\widetilde{\text{out}}(A)) = \{\text{defuz}(\widetilde{\text{out}}(x)) \text{ s. t. } x \in A\}. \tag{9}$$

Lastly, given a fuzzy input $\widetilde{A} \in \mathcal{F}(X)$, we will assign it, according to the Extension Principle (which is compatible

with the possibilistic interpretation of fuzzy sets) a fuzzy set computed as follows:

1. We determine an intermediate fuzzy set on the universe $\mathcal{F}(K), \widetilde{\text{out}}(\widetilde{A}) \in \mathcal{F}(\mathcal{F}(K))$, defined as

$$\widetilde{\text{out}}(\widetilde{A})(\widetilde{B}) = \sup\{\widetilde{A}(x) \text{ s. t. } \widetilde{\text{out}}(x) = \widetilde{B}\}, \quad \forall \widetilde{B} \in \mathcal{F}(K) \tag{10}$$

2. An element of $\mathcal{F}(K)$ (that is to say, a fuzzy output) $\text{defuz}(\widetilde{\text{out}}(\widetilde{A})) \in \mathcal{F}(K)$ is obtained as follows:

$$\text{defuz}(\widetilde{\text{out}}(\widetilde{A}))(k) = \sup\{\widetilde{A}(x) \text{ s. t. } \text{defuz}(\widetilde{\text{out}}(x)) = k\}, \quad \forall k \in K. \tag{11}$$

Observe that the fuzzy set $\text{defuz}(\widetilde{\text{out}}(\widetilde{A}))$ is associated to the nested family of sets $\{\text{defuz}(\widetilde{\text{out}}(A_\alpha))\}_{\alpha\in[0,1]}$, and that explains the possibilistic interpretation of this procedure.

3 Definition of the extended genetic fuzzy system

We have seen in the preceding section that an imprecise knowledge about the input variables means that the output of the FRBS will not be completely determined: it is a fuzzy set of classes (or a crisp subset, if the input is set-valued). From the foregoing it can be deduced that the number of errors of the FRBS in the training data will be also a set. The same happens if any other quality function is used instead of the number of errors, i.e. likelihood, logistic loss functions, etc. Let us use an example: we have a classification system, defined by these rules:

$$\begin{aligned} &\text{if } x<1 \text{ then class is } A \\ &\text{if } x \in [1,2] \text{ then class is } B \\ &\text{if } x>2 \text{ then class is } C \end{aligned} \tag{12}$$

and the input that follows:

$$x<1.8. \tag{13}$$

The output of the classifier is the set of classes {A, B}. If the object being classified is of class C, we know that the classifier has failed. Otherwise, we cannot know. Nonetheless, we can use a set-valued variable "number of errors", and state that the error of the classifier in that example is the set {0, 1}.

It is remarked too that, if a point is labeled as "class {A, B}" we are not stating that it belongs to both categories at the same time (which is not an imprecise assert). We are expressing that we are not sure about the class of the object, i.e. we only know that it is not in class "C". Therefore, if the output of the classifier is the set of classes {A, B} and the point is also labeled as "class {A, B}", the error in this point is still {0, 1} and not 0. Because of this, in this paper, each rule will contain a single consequent. We will not consider FRBS like, for instance,

$$\begin{aligned} &\text{if } x<1 \text{ then class is } \{A,B\} \\ &\text{if } x \in [1,2] \text{ then class is } B \\ &\text{if } x>2 \text{ then class is } C \end{aligned} \tag{14}$$

because, according to our interpretation, the first rule necessarily will have a non-zero error at any example; therefore, for any dataset we can conceive, we could find an FRBS comprising only single-consequent rules whose error is more specific than that of (14).

3.1 Crisp GFS

The GFS that we will generalize to vague data was introduced in [7]. We have chosen this algorithm because of its balance between simplicity and performance. In future works we intend to extend the procedure described in this section to more recent algorithms [16].

The pseudocode of this algorithm is shown in Fig. 2. It can be seen that it depends on two functions: "assignConsequent" (line 6) and "assignFitness" (line 9). These functions are also listed in Figs. 3 and 4.

Observe that this algorithm does not codify the consequent of the fuzzy rules in the genetic individual neither we are assigning weights to the rules. The function "assignConsequent" determines the class label that matches an antecedent with a maximum confidence. The function "assignFitness," in turn, determines the winner rule for each object in the training set and increments the fitness of the corresponding individual if its consequent matches the class of the object.

3.2 Generalized GFS

Generalizing a GFS to imprecise data involves changes to the inference mechanism, that we have discussed in Sect. 2,

```
function GFS
1    Initialize population
2    for iter in {1, ..., Iterations}
3      for sub in {1, ..., subPop}
4        Select parents
5        Crossover and mutation
6        assignConsequent(offspring)
7      end for sub
8      Replace the worst subPop individuals
9      assignFitness(population,dataset)
10   end for iter
11   Purge unused rules
return population
```

Fig. 2 Outline of the GFS that will be generalized [7]. Each chromosome codifies one rule. The fitness of the classifier is distributed among the rules at each generation

```
function assignConsequent(rule)
1    for example in {1,...,N}
2      m = membership(Antecedent,example)
3      weight[class[example]] = weight[class[example]] + m
4    end for example
5    mostFrequent = 0
6    for c in {1,...,Nc}
7      if (weight[c]>weight[mostFrequent]) then
8        mostFrequent = c
9      end if
10   end for c
11   Consequent = mostFrequent
return rule
```

Fig. 3 The consequent of a rule is not codified in the GA, but it is assigned the most frequent class label, between those compatible with the antecedent of the rule [7]

```
function assignFitness(population,dataset)
1    for example in {1,...,N}
2      winnerRule = 0
3      bestMatch = 0
4      for rule in {1,...,M}
5        m = membership(Antecedent[rule],example)
6        if (m>bestMatch) then
7          winnerRule = rule
8          bestMatch = m
9        end if
10     end for rule
11     if (consequent(winnerRule)==class(example)) then
12       fitness[winnerRule] = fitness[winnerRule] + 1
13     end if
14   end for example
return fitness
```

Fig. 4 The fitness of an individual is the number of examples that it classifies correctly. Single-winner inference is used, thus at most one rule changes its fitness when the rule base is evaluated in an example [7]

and also to the fitness function, as we have introduced in the preceding paragraphs (see also [13] for a deeper explanation). In the remainder of this subsection, we will study how to alter these functions "assignConsequent" and "assignFitness". This comprises

1. new procedures to assign the consequents,
2. computing set-valued fitness functions, and
3. the genetic selection and replacement of the worst individuals, including a short discussion about the meaning of "best" and "worst" when the fitness is a set-valued function.

3.2.1 *Assignment of consequents*

The assignment of consequents seen in Fig. 3 is extended in Fig. 5. The original assigment consists in computing the confidences of the rules "if (x is $\widetilde{A}$) then class is C" for all the values of "C", then selecting the alternative with

```
function assignImpreciseConsequent(rule)
1    for example in {1,...,N}
2      m̃ = fuzMembership(Antecedent,example)
3      weight[{class[example]}] = weight[{class[example]}] ⊕ m̃
4    end for example
5    mostFrequent = {1,...,Nc}
6    for c in {1,...,Nc}
7      for c1 in {c+1,...,Nc}
8        if (weight[c] dominates weight[c1]) then
9          mostFrequent = mostFrequent - { c1}
10       end if
11     end for c1
12   end for c
13   Consequent = mostFrequent
return rule
```

Fig. 5 If the examples are imprecise, we might not know the most frequent class label—lines 5–12. In this paper we have used the dominance proposed in [10] to reduce this set to one element

maximum confidence. In this case, the confidence of a rule is a set of values.

The operation "dominates" used in line 8 can have different meanings, ranging from the strict dominance (A dominates B iff $a < b$ for all $a \in A$, $b \in B$) [18] to other definitions that induce a total order in the set of confidences. Generally speaking, we have to select one of the values in the set of nondominated confidences and use its corresponding consequent. In this paper, we have used the uniform dominance defined in [10], that induces a total order and thus the set of nondominated consequents has size 1. This issue is further discussed at the end of this section.

3.2.2 *Computation of fitness*

We have mentioned at the beginning of this section that the error of the FRBS at an imprecisely perceived object is an interval or a fuzzy set. The number of errors of the whole classifier can be obtained by adding these individual errors with interval or fuzzy arithmetic operators.

Roughly speaking, estimating a classifier from data requires a numerical technique that finds the minimum of the classification error with respect to the free parameters of the classifying system. In our case, this function is interval-valued or fuzzy. But there are not many techniques for optimizing interval-valued or fuzzy valued functions. In the genetic algorithms field, the solutions are related to precedence operators between imprecise values [8, 10, 18]. In previous works, we have jointly optimized a mix of crisp and fuzzy objectives with genetic algorithms [14]. We have also proposed a number of different algorithms for learning regression models from low quality data and the fuzzy representation mentioned before [12, 16, 17]. However, to the best of our knowledge there have not been previous GFSs where those principles have been applied to learn classification problems.

In case that the *i*th object of the training set is perceived through a crisp set, the output of the FRBS is a set of classes:

$$C_{\mathrm{FRBS}}(X_i) = \{C_{\underset{\mathrm{argmax}_j\{\widetilde{A}_j(x)\}}{\sim}} \mid x \in X_i\}. \quad (15)$$

Accordingly, for a fuzzy value $\widetilde{X}_i$ the output is the fuzzy subset of {1, ..., N_c} that follows:

$$\widetilde{C}_{\mathrm{FRBS}}(X_i)(k) = \max\{\alpha \mid k \in C_{\mathrm{FRBS}}([X_i]_\alpha)\} \quad (16)$$

for $k \in \{1, ..., N_c\}$. It can be inferred that the theoretical expression of the fitness function of the FRBS is:

$$\widetilde{f} = \bigoplus \widetilde{e}_i \quad (17)$$

where $\widetilde{e}_i$ is a fuzzy subset of {0, 1}, whose α-cuts are:

$$[\widetilde{e}_i]_\alpha = \begin{cases} 1 & C_{\mathrm{FRBS}}([X_i]_\alpha) = C_i \text{ and } \#(C_i) = 1 \\ 0 & C_{\mathrm{FRBS}}([X_i]_\alpha) \cap C_i = \emptyset \\ \{0,1\} & \text{else} \end{cases} \quad (18)$$

In words, if the output of the FRBS is a single class label that matches the class label of the example, this point scores 1. If the set of classes emitted by the FRBS does not intersect with that of the object, this point scores 0. Otherwise, it scores the set {0, 1}.

The evaluation of this function is computationally very expensive, and we will use an approximation, described in Fig. 6 for interval-valued data. This algorithm computes an interval of values of matching between each rule and the input, then discards all rules that can not be the winner rule, and approximates the output of the FRBS by the set of the consequents of the non-discarded rules. This set includes the theoretical output, but sometimes it also includes extra class labels. In Fig. 7 we have also included a more accurate approximation which is based on a sample of values of the support of the input. This second approximation will be used in the next section to better determine the quality of a classifier, but our learning will be guided by the function in Fig. 6, because of its lower cost.

```
function assignImpreciseFitnessApprox(population,dataset)
1    for example in {1, ..., N}
2      setWinnerRule = ∅
3      for rule in {1, ..., M}
4        dominated = FALSE
5        rule.m̃ = fuzMembership(Antecedent[r],example)
6        for sRule in setWinnerRule
7          if (sRule dominates rule) then
8            dominated = TRUE
9          end if
10       end for sRule
11       if (not dominated and rule.m̃ > 0) then
12         for sRule in setWinnerRule
13           if (rule.m̃ dominates sRule) then
14             setWinnerRule = setWinnerRule −{ sRule }
15           end if
16         end for sRule
17         setWinnerRule = setWinnerRule ∪{ rule }
18       end if
19     end for r
20     if (setWinnerRule == ∅) then
21       setWinnerRule = setWinnerRule ∪{ rule_freq_class }
22     setOfCons= ∅
23     for sRule in setWinnerRule
24       setOfCons= setOfCons ∪{ consequent(sRule) }
25     end for sRule
26     deltaFit= 0
27     if ({class(example)} == setOfCons and
           size(setOfCons)==1) then
28       deltaFit = {1}
29     else
30       if ({class(example)}∩ setOfCons ≠ ∅) then
31         deltaFit = {0, 1}
32       end if
33     end if
34     Select winnerRule ∈ setWinnerRule
35     fitness[winnerRule] = fitness[winnerRule] ⊕ deltaFit
36   end for example
return fitness
```

Fig. 6 Generalization of the function "assignFitness" to imprecise data. If the example is imprecisely perceived, there are three ambiguities that must be resolved: **a** some different crisp values compatible the same example might correspond to different winner rules—lines 3–19, **b** these rules might have different consequent, thus we do not know if the rule base fails in the example—lines 22–33 and **c** we must assign credit to just one of these rules—lines 34 and 35

3.2.3 Genetic selection and replacement

There are two other parts in the original algorithm that must be altered in order to use an imprecise fitness function: (a) the selection of the individuals in [7] is based on a tournament, that depends on a total order on the set of fitness values. And (b) the same happens with the removal of the worst individuals. We leave for future works the application of a multicriteria genetic algorithm similar to those used in our previous works in regression modeling [16, 17]. In both cases, we have used the uniform dominance defined in [10] to impose such a total order. This definition is recalled in the next subsection.

3.2.4 Precedence in imprecise fitness-based genetic algorithms

In this section we detail how we define an ordering between fitness values, to be used in the parts of the mentioned algorithm. Let f_1, f_2 be the fitnesses of two FRBSs. We will assume that f_1 and f_2 are unknown, but we know two fuzzy sets $\widetilde{f}_1$ and $\widetilde{f}_2$ that describe them. Let θ_1 and θ_2 be the interval-valued expectations [4] of $\widetilde{f}_1$ and $\widetilde{f}_2$.

We want to determine whether one individual precedes the other, thus we need a procedure that estimates whether the probability of $f_1 < f_2$ is greater than that of $f_1 \geq f_2$ (thus $\theta_1 \prec \theta_2$) or not. We also want to find those cases where there is no statistical evidence in θ_1 and θ_2 that makes us prefer one of them (thus $\theta_1 \| \theta_2$). Our approach can be regarded as a PQI interval order [19], where there is a zone

```
function assignImpreciseFitnessExhaustive(population,dataset)
1     for example in {1,...,N}
2       S = sample(example)
3       maxScore =∅
4       for s in S
5         winnerRule = 0
6         bestMatch = 0
7         for rule in {1,...,M}
8           m = membership(Antecedent[rule],s)
9           if (m > bestMatch) then
10            winnerRule = rule
11            bestMatch = m
12          end if
13        end for rule
14        if (consequent(winnerRule) == class(example)) then
15          score[winnerRule] = score[winnerRule] ⊕ 1
16        else
17          if consequent(winnerRule) ⊂ class(example)) then
18            score[winnerRule] = score[winnerRule] ⊕ {0,1}
19          end if
20        end if
21        if (maxScore ∩ score[winnerRule] = ∅)
22          then maxScore = maxScore ⊕ score[winnerRule]
23        end if
24      end for s
23        if (maxScore == size(S)) then
24          fitness = fitness ⊕ 1
25        else
26          if (maxScore ∩ 0 ≠ ∅ and maxScore ∩ 1 ≠ ∅ ) then
27            fitness = fitness ⊕ {0,1}
28          end if
29        end if
30    end for example
return fitness
```

Fig. 7 Other generalization of the function "assignFitness" to interval-valued data. This function is computationally too expensive for being used as a fitness function; it will be used instead for obtaining better estimations of the train and test errors of the final rule bases. Lines 14–18 deal with the case where an object has imprecise output, i.e. "the class is A or C"; otherwise, the value of the variable "score" is crisp

of hesitation between strict difference and strict similarity. We have considered two different scenarios:

1. Strong Dominance
   Without further assumptions, when θ_1 and θ_2 are non-disjoint intervals, we do not have evidence to prefer one of them. Otherwise, the decision is trivial. This criterion has been called *strong dominance* in [10].
2. Probabilistic Prior
   We introduce prior knowledge about the probability distribution of the fitness. If a joint probability $P(f_1, f_2)$ was known, comparing two individuals would be a statistical decision problem. For instance, we can decide that $\theta_1 \prec \theta_2$ when

$$\frac{P(\{(f_1,f_2) : f_1 < f_2\})}{P(\{(f_1,f_2) : f_1 \geq f_2\})} > 1. \tag{19}$$

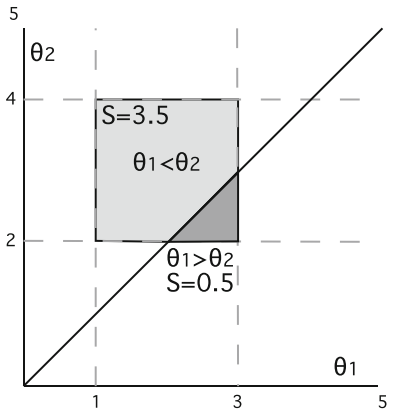

Fig. 8 Graphical representation of the example 2, showing that $[1, 3] \prec [2, 4]$

For instance, in [18] it was assumed that $P(f_1, f_2)$ was uniform. This use of a uniform prior will be made clear in the examples below.

Example 1 Let $\theta_1 = [1, 3]$ and $\theta_2 = [2, 4]$ two non-disjoint intervals. If we assume that $P(f_1, f_2)$ is uniform in $[1, 3] \times [2, 4]$ (see Fig. 8) we obtain

$$\frac{P(\{(f_1,f_2) : f_1 < f_2\})}{P(\{(f_1,f_2) : f_1 \geq f_2\})} = \frac{3.5/4}{0.5/4} > 1 \tag{20}$$

thus we can state that $\theta_1 \prec \theta_2$.

Example 2 Let $\theta_1 = [1, 5]$ and $\theta_2 = [1.9, 4]$ two non-disjoint intervals. The application of the same principle produces

$$\frac{P(\{(f_1,f_2) : f_1 < f_2\})}{P(\{(f_1,f_2) : f_1 \geq f_2\})} = \frac{4.095}{4.305} < 1 \tag{21}$$

therefore $\theta_2 \prec \theta_1$.

The uniform prior defines a total order in the population, since every pair of intervals is comparable. We may question the consistency of this order, though. In the last example, there might be situations where a fitness [1, 5] could be prefered to [1.9, 4], and it is also reasonable to state that these two intervals cannot be compared. We have proposed a more complex ordering in [17], albeit we have decided not to use it in this initial algorithm for simplifying our explanation.

4 Numerical results

This section contains a numerical analysis of the generalized algorithm. To the best of our knowledge, there are not previous publications with compared results of machine learning algorithms over vague datasets. Therefore, we have collected our own data, and propose to use the

datasets that will be described in this section for future developments in the field. We have considered three categories of problems:

1. Synthetic datasets: we have generated a sample of data with Gaussian distribution and known Bayesian error. To this data we have added different amounts of observation error.
2. Realistic problems: a dataset comprising actual measurements for a problem whose statistical distribution is not known, but knowing the optimal solution.
3. Real-world problems: datasets originated in open problems of medical diagnosis and high performance athletics, whose optimal solution neither the statistical distribution of the data nor the amount of observation error are known.

Additionally, we have also tested the new algorithm over some standard, crisp datasets. This is intended for checking that the extended algorithm has the same performance as the original version in crisp problems.

4.1 Settings

All the experiments have been run with a population size of 100, probabilities of crossover and mutation of 0.9 and 0.1, respectively, and limited to 200 generations. The fuzzy partitions of the labels are uniform and their size is 3, except when mentioned otherwise. We have used the precedence criteria called 'Probabilistic Prior' in Sect. 3.2.4 for comparing fitness values and also for assigning the consequents to the rules. All the datasets used in this paper will be made freely available in the website of the KEEL project: http://www.keel.es.

4.1.1 Compared results between crisp and low quality data-based algorithms

In this section, we include compared results between crisp and low quality data-based algorithms. We are not aware of previous works in the field, and this comparison is difficult, as it involves a method for removing the observation error from the data. This is necessarily an arbitrary choice. Depending on the problem, the same prior assumptions about the probability distribution of the data can be effective or not.

In this section we have applied the following rules when comparing crisp and interval-valued data:

- If the imprecision is in the input, each interval has been replaced by its midpoint. For instance, a vague example ($X = [1, 3]$, $C = A$) is converted into ($x = 2$, $C = A$).
- If the imprecision is in the output, each sample has been replicated for the different alternatives. Each replication is assigned a degree of importance such that the contribution of the example to the total fitness is not multiplied by the number of replicas. For instance, an example ($x = 2$, $C = \{A, B\}$) is converted in two examples ($x = 2$, $C = A$) and ($x = 2$, $C = B$), and each one of them is assigned an importance 0.5. This requires a small change of the original algorithm, which is shown in Figs. 9 and 10.

For removing the imprecision in fuzzy data, we have replaced each fuzzy set by its modal point. For instance, a vague example ($\widetilde{X} = [1; 2; 4], C = A$) (where [1;2;4] is a triangular fuzzy set with support [1, 4] and modal point 2) is defuzzified in the crisp value ($x = 2$, $C = A$).

4.2 Experimentation with crisp datasets

In the first place, we have solved some crisp datasets, that have been included for checking how the algorithm performs on crisp data. The results show that there are not

```
function assignConsequent(rule)
1     for example in {1, ..., N}
2       m = membership(Antecedent,example) * ω(example)
3       weight[class[example]] = weight[class[example]] + m
4     end for example
5     mostFrequent = 0
6     for c in {1, ..., Nc}
7       if (weight[c]>weight[mostFrequent]) then
8         mostFrequent = c
9       end if
10    end for c
11    Consequent = mostFrequent
return rule
```

Fig. 9 The original algorithm in [7] is altered as shown in line 2, so that is able to learn from a database where each example has a fractional degree of importance "ω(example)"

```
function assignFitness(population,dataset)
1     for example in {1, ..., N}
2       winnerRule = 0
3       bestMatch = 0
4       for rule in {1, ..., M}
5         m = membership(Antecedent[rule],example)
6         if (m>bestMatch) then
7           winnerRule = rule
8           bestMatch = m
9         end if
10      end for rule
11      if (consequent(winnerRule)==class(example)) then
12        fitness[winnerRule] = fitness[winnerRule] + * ω(example)
13      end if
14    end for example
return fitness
```

Fig. 10 The fitness of an individual is the number of examples that it classifies correctly. Single-winner inference is used, thus at most one rule changes its fitness when the rule base is evaluated in an example. The algorithm in [7] has also been altered (see line 12) for dealing with weighted examples

differences between the original algorithm in [7] and the generalized version proposed here when the datasets are crisp. We have included these experiments in Table 1.

4.3 Experimentation with synthetic datasets

The dataset that we have called "Gaussian" comprises 699 points of two classes. The distribution of both classes is bidimensional Gaussian, with unity covariance matrix, and centered in (0, 0) and (3, 0), respectively. To this data we have added interval-valued imprecision of sizes $\beta = 0.03, 0.05, 0.1, 0.2, 0.5$.

A 10cv experimental design was applied, and the mean values of the test errors are shown in Table 2 and Fig. 11. The training error has been also included, to show the differences between the approximation of the fitness function seen before and the exhaustive computation that has been used to compute the test error. Observe that the approximate error computed by the fitness function is less specific than the actual error, and the difference is relevant when the observation error is high ($\beta = 0.2$ and $\beta = 0.5$), nevertheless it still guides the evolution correctly.

4.4 Experimentation with realistic datasets

The dataset "Screws" comprises 21 objects of three different classes. Each object has two features, weight and length. If all the measurements were accurate, this problem could have been solved without error. There is no ambiguity in the class labels, and each feature is an interval. Since the misclassification rate is entirely originated in the observation error, the results shown in Table 3 indicate that the algorithm proposed here has exploited better the information contained in the imprecise data than the crisp algorithm.

Table 1 Classification error in some crisp benchmarks, where the imprecise fitness function is the same as the crisp fitness function

Dataset	Crisp		Low quality	
	Train	Test	Train	Test
Pima	0.253	0.283	[0.254,0.258]	[0.271,0.278]
Glass	0.332	0.356	[0.328,0.329]	[0.356,0.356]
Haberman	0.241	0.261	[0.232,0.243]	[0.254,0.261]

The results of "crisp" and "low quality" columns are similar

4.5 Experimentation with real world datasets

In this section we will describe two different real-world problems. The first one is a medical diagnosis problem, and the second one is related to the composition of teams in

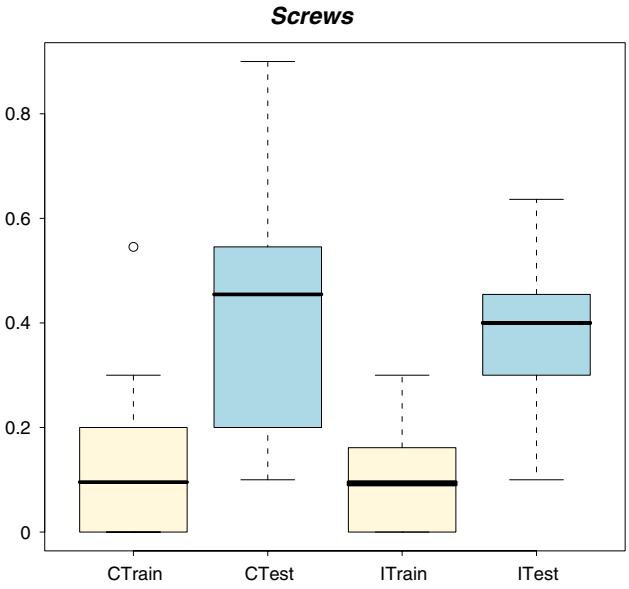

Fig. 11 Boxplots illustrating the error of crisp (columns 'CTrain' and 'CTest') and extended GFS (columns 'ITrain' and 'ITest') in the problem "Screws"

Table 2 Results of the extended GFS in the synthetic dataset "Gauss" for crisp data ('Crisp' columns) and different degrees of observation error ('Low Quality' columns)

β	Crisp			Low quality		
	Theoretical	Train	Test	Approx. train	Exh. test	Exh. train
0	0.084	0.083	0.086	[0.086,0.086]	[0.082,0.082]	[0.086,0.086]
0.03				[0.076,0.091]	[0.083,0.094]	[0.047,0.086]
0.05				[0.071,0.094]	[0.081,0.098]	[0.075,0.089]
0.1				[0.076,0,093]	[0.068,0.104]	[0.070,0.103]
0.2				[0.075,0.089]	[0.055,0.128]	[0.052,0.116]
0.5				[0.014,0.225]	[0.022,0.179]	[0.022,0.183]

Training error for low quality data has been computed twice, with a slow, precise algorithm ('Exhaustive Train') or a fast approximation ('Approximate Train') (see Fig. 7). The approximate fitness function (see Fig. 6) has guided the evolution. The test error is always computed with the precise algorithm ('Exhaustive test'). The approximate error is less specific than the actual error, and the difference is relevant when the observation error is high ($\beta = 0.2$ and $\beta = 0.5$), nevertheless it still converges to a good FRBS

Table 3 Results of the generalized GFS for the imprecise datasets "Screws"

Crisp			Low quality		
Dataset	Train	Test	Approx. train	Exh. test	Exh. train
Screws	0.133	0.427	[0.068,0.106]	[0.377,0.377]	[0.096,0.096]

The algorithm in this paper has exploited the low quality information better than the crisp algorithm

high performance athletics. Both problems are open; we do not know the best attainable error with a Genetic Fuzzy Classifier.

4.5.1 Diagnosis of dyslexia

Dyslexia is a learning disability in people with normal intellectual coefficient, and without further physical or psychological problems explaining such disability. It has been estimated that between 4 and 5% of schoolchildren have dislexia, with reading and writing problems [1]. The average number of children in a Spanish classroom is 25, therefore most of them have dyslexic children. Dyslexia may become apparent in early childhood, with difficulty putting together sentences and a family history. Recognition of the problem is very important in order to give the infant an appropriate teaching.

Using Soft Computing techniques for diagnosing dyslexia seems to us a natural choice, because of the properties of our data (linguistic terms, and vague measurements). As a matter of fact, there are many references where fuzzy techniques were used to learn medical diagnosis models from data. In particular, in [5] and [9], fuzzy techniques have been used in the diagnosis of disabilities in language. However, in all of the preceding works, the data was crisp or categorical. Instead, most of our measurements (see Table 4) are not crisp. Some of our responses are linguistic ("low", "high"), others are subjective (like the "squareness" of a hand-drawn shape—see Fig. 12) or interval valued (f.e. a dyslexia degree "between 2 and 4"). Lastly, a high percentage of cases have missing values. None of the preceding approaches are directly applicable to the problem at hand.

The dataset used in this section is called "Dyslexia-12". It has 65 objects, 4 classes and 12 features. This is a selection of the original dataset described in [15], where the 12 most relevant variables have been hand-picked by a psychologist. There are imprecision in both the input and the output. The theoretical error is unknown.

We have used a 10cv design, and the boxplots of the compared results, in both train and test sets, are depicted in Fig. 13. Observe that the boxplots of the imprecise experiments are not standard. We propose using a box showing the 75% of the maximum and 25% percentile of the minimum fitness (thus the box displays at least the 50% of data) and also drawing two marks inside the box, because the median of the data is an interval. In Fig. 14 the ranges and means of all repetitions of the learning are shown, for both the crisp and the imprecise versions of the algorithm. The upper bound of the mean imprecise fitness is consistently lower than the mean of the crisp fitness.

Table 4 Categories of the tests currently applied in Spanish schools for detecting dyslexia in children between 5 and 8 years

Category	Test	Description
Verbal comprehension	BAPAE	Vocabulary
	BADIG	Verbal orders
	BOEHM	Basic concepts
Logic reasoning	RAVEN	Color
	BADIG	Visual memory
Sensory-motor skills	BENDER	Visual-motor coordination
	BADIG	Perception of shapes
	BAPAE	Spatial relations, shapes, orientation
	STAMBACK	Auditive perception, rhythm
	HARRIS/HPL	Laterality, pronunciation
	GOODENOUGHT	Spatial orientation, body scheme
Reading-writing	TALE	Analysis of reading and writing

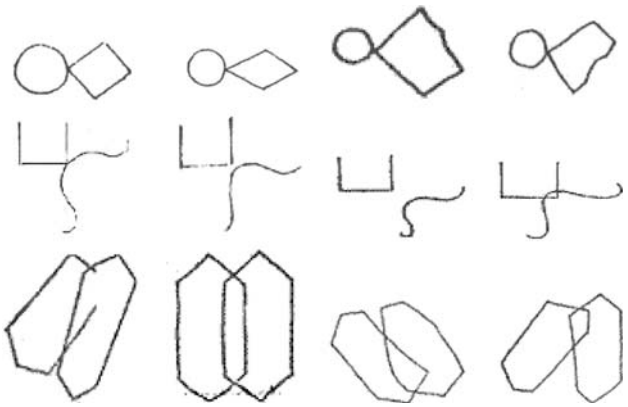

Fig. 12 Example of some of Bender's tests for detecting dyslexia. *Upper part* The angles of the shape in the right are qualified by a list of adjectives that can contain the words "right," "incoherent," "acceptable," "regular" and "extra." *Middle* and *lower part*: The relative position between the figures can be "right and separated," "right and touching," "intersecting", etc

4.5.2 High performance athletics

The score of an athletics team is the sum of the individual scores of the athletes in the different events. It is the coach's responsibility to balance the capabilities of the

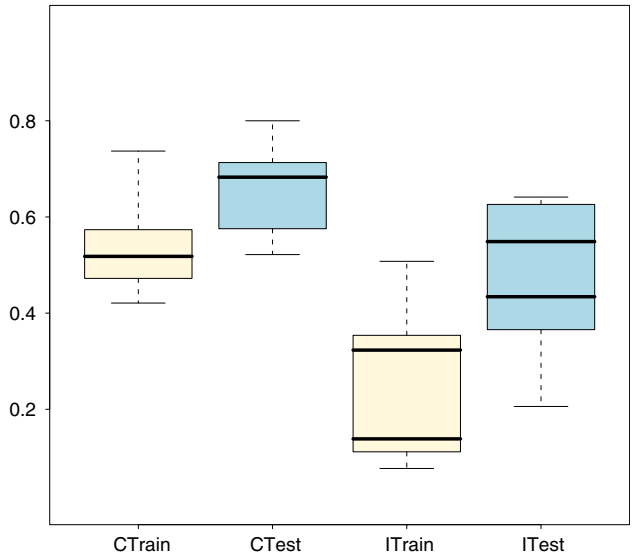

Fig. 13 Boxplots illustrating the results of crisp (columns 'CTrain' and 'CTest') and extended GFS (columns 'ITrain' and 'ITest') in the problem "dyslexia-12", with 4 labels/partition

different athletes in order to maximize the score with a team according to the regulations. In this practical application, an algorithm is used for choosing the best team. The algorithm makes use of the expected marks of the athletes at each event, and also of the confidence degrees in the achievement of these marks. These expected marks are determined by the trainer according to the past performance of the athlete, a set of indicators that are described in this section and, optionally, the marks of rival teams. The objective is to choose the subset of athletes that will get a given score, with the highest confidence.

The objective of the FRBS that is evaluated in this section is to determine whether an athlete will improve a certain mark, in two different events: long jump and 100 m. The variables that define each problem are as follows:

1. There are four indicators in long jump that are used to predict whether an athlete will pass a given threshold [20]: the ratio between the weight and the height, the maximum speed in the 40 m race and the tests of central (abdominal) muscles and lower extremities. The first two indicators are determined by the coach, who was allowed to use numbers, intervals or linguistic values (fuzzy intervals) at his convenience. The two last tests are repeated three times, and produce numbers. The abdominal muscle test consists in counting how many flexion movements the athlete can repeat in a minute. Lastly, the lower extremities test measures how much the athlete can stretch.
2. There are also four indicators in the 100 m race: the ratio between weight and height, the reaction time, the starting or 20 m speed, and the maximum or 40 m speed. We have collected two different databases for this problem. In the first database, three different people measure the actual reaction time, starting and maximum speed of the athletes. These three measurements are joined to form an imprecise value. On the contrary, in the second database the trainer has graded each speed and time with a mark between 0 and 10. He was allowed to express his grades with numbers, intervals or linguistic values. This second database has a high subjective component; it serves to assess the expert knowledge of the trainer about the athletes, by comparing this results with the actual measurements.

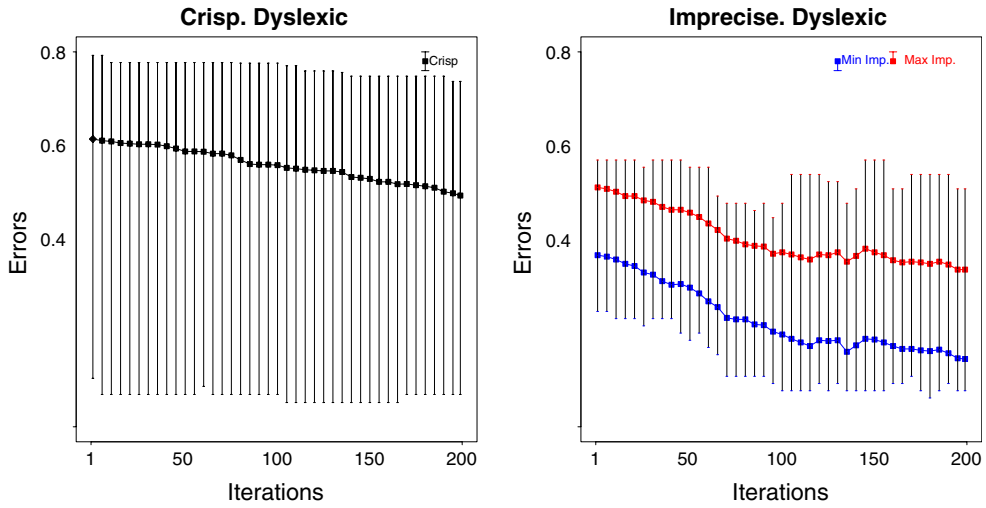

Fig. 14 Compared evolution of crisp (*left*) and imprecise GFS (*right*) in the dataset "dyslexia-12". The minimum, mean and maximum classification error at every generation are shown, for both the crisp and the imprecise versions of the algorithm. The upper bound of the imprecise fitness is consistently lower than the crisp fitness

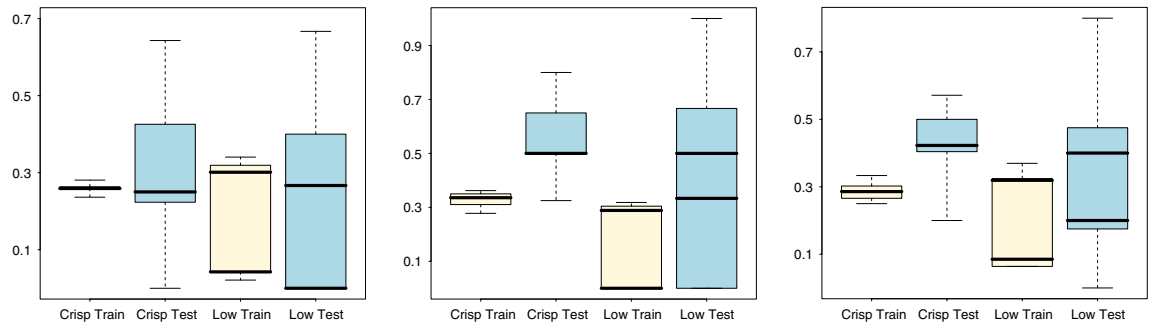

Fig. 15 Boxplots illustrating the classification error of crisp and extended GFS in the high performance athletics problem. From *left* to *right*,: Problems "100ml-4-I", "Long-4", and "100ml-4-P"

Finally, the datasets that are used in this application are:

1. Dataset "Long-4": This dataset is used to predict whether an athlete will improve certain threshold in the long jump, given the indicators mentioned before. We have measured 25 athletes, thus the set has 25 instances, 4 features, 2 classes, no missing values. All the features, and also the output variable, are interval-valued.
2. Dataset "100ml-4-I": Used for predicting whether a mark in the 100 metres sprint race is being achieved. Actual measurements are taken by three observers, and are combined into the smallest interval that contains them. 25 instances, 4 features, 2 classes, no missing data. All input and output variables are intervals.
3. Dataset "100ml-4-P": Same dataset as "100ml-4-I", but the measurements have been replaced by the subjective grade the trainer has assigned to each indicator (i.e. "reaction time is low" instead of "reaction time is 0.1 seg").

We have compared the performance of the generalized algorithm to that of the original crisp algorithm, as mentioned before. We have used a 10cv design for all datasets. The boxplots with all the results are shown in Fig. 15. Observe that the boxplots of the imprecise experiments are not standard, as before. We propose to use an extended boxplot that can describe a sample of interval data. We will be using a box showing the upper bound of the 75% percentile of the maximum and the lower bound of the 25% percentile of the minimum fitness (thus the box displays at least the 50% of data). The also interval-valued median is drawn with two marks inside this box. In addition, the numerical values of the classification error have also been included in Tables 5 and 6.

The results are promising in all the experiments. We expected that the extra freedom for the coach has when he is allowed to use ranges of values and linguistic terms instead of numbers would allow us to capture better his expertise, and the results seem to confirm this intuition (Table 6, column "Test, Low Quality").

Table 5 Results of the generalized GFS for the imprecise datasets "Dyslexia-12" with 4 and 5 labels/variable

Crisp			Low quality	
Dataset	Train	Test	Approx. train	Exh. test
Dyslexia-12 (4 labels)	0.541	0.657	[0.144,0.335]	[0.421,0.558]
Dyslexia-12 (5 labels)	0.672	0.694	[0.155,0.355]	[0.490,0.609]

Table 6 Expected classification error of the generalized GFS for the imprecise datasets "Long-4", "100ml-4-P" and "100ml-4-I"

Crisp			Low quality	
Dataset	Train	Test	Train	Test
Long-4 (5 labels)	0.327	0.544	[0.0,0.279]	[0.349,0.616]
100ml-4-P (5 labels)	0.288	0.419	[0.076,0.320]	[0.17,0.406]
100ml-4-I (5 labels)	0.259	0.384	[0.089,0.346]	[0.189,0.476]

5 Concluding remarks

Extending a GFS to imprecise data in classification problems is based on the use of an interval or fuzzy valued fitness function. Most GFSs can be extended to low quality data if some changes are made in their reasoning method, and the genetic algorithm can deal with an imprecisely known fitness function. We have shown in detail how to apply this changes to a simple GCCL-type algorithm, and evaluated it with some synthetic, realistic and real-world benchmarks. The numerical results are as expected for an elementary algorithm like this; there is room for improvement and future works will address more complex GFSs that are based on a multicriteria fitness function.

Acknowledgments This work was supported by the Spanish Ministry of Education and Science, under grants TIN2008-06681-C06-04, TIN2007-67418-C03-03, and by Principado de Asturias, under grant PCTI 2006-2009.

References

1. Ajuriaguerra J (1976) Manual de psiquiatrí a infantil (in Spanish). Toray-Masson, Barcelona
2. Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic fuzzy systems Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, Singapore
3. Couso I, Sánchez L (2008) Higher order models for fuzzy random variables. Fuzzy Sets Syst 159:237–258
4. Dubois D, Prade H (1987) The mean value of a fuzzy number. Fuzzy Sets Syst 24(3):279–300
5. Georgopulos V (2003) A fuzzy cognitive map to differential diagnosis of specific language impairment. Artif Intell Med 29:261–278
6. Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol Intell 1:27–46
7. Ishibuchi H, Nakashima T, Murata T (1995) A fuzzy classifier system that generates fuzzy if-then rules for pattern classification problems. In: Proceedings of 2nd IEEE international conference on evolutionary computation, pp 759–764
8. 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, Turkey, pp 450–453
9. Lakov DV (2004) Soft computing agent approach to remote learning of disables. In: 2nd IEEE international conference on intelligent systems, pp 250–255
10. Limbourg P (2005) Multi-objective optimization of problems with epistemic uncertainty. In: EMO 2005, pp 413–427
11. Medasani S, Kim J, Krishnapuram S (1998) An overview of membership function generation techniques for pattern recognition. Int J Approx Reason 19(3–4):391–417
12. 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
13. Sánchez L, Couso I (2007) Advocating the use of imprecisely observed data in genetic fuzzy systems. IEEE Trans Fuzzy Syst 15(4):551–562
14. 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 intelligence in multicriteria decision making, Honolulu, pp 30–37
15. Sánchez L, Palacios A, Couso I (2008) A minimum risk wrapper algorithm for genetically selecting imprecisely observed features, applied to the early diagnosis of dyslexia. Lect Notes Comput Sci 5271:608–615
16. Sánchez L, Otero J, Couso I (2009) Obtaining linguistic fuzzy rule-based regression models from imprecise data with multiobjective genetic algorithms. Soft Comput 13(5):467–479
17. Sánchez L, Couso I, Casillas J (2009) Genetic learning of fuzzy rules based on low quality data. Fuzzy Sets Syst 160(17):2524–2552
18. Teich J (2001) Pareto-front exploration with uncertain objectives. In: EMO 2001, pp 314–328
19. Öztürk M, Tsoukias A (2007) Valued Hesitation in intervals comparison. In: Proceedings of the SUM-07 conference, LNAI 4772, Springer, pp 157–170
20. Vinuessa M, Coll J (1984) Tratado de atletismo. Servicio Geográfico del Ejercito Español