A baseline learning genetic fuzzy classifier based on low quality data

Ana M. Palacios[1] Luciano Sánchez[1] Inés Couso[2]

1. Departamento de Informática, Universidad de Oviedo, Gijón, Asturias, Spain
2. Departamento de Estadística e I.O. y D.M, Universidad de Oviedo, Gijón, Asturias Spain
Email: apalaciosjimenez@hotmail.com, luciano@uniovi.es, couso@uniovi.es

Abstract— Obtaining fuzzy rules from low quality data is a topic that has been recently formalized. This paper contains the first application of these principles to classification problems. We intend that the classifier proposed here serves as a baseline for future developments in the field. For that reason, we have extended a simple crisp genetic fuzzy classifier to imprecise data, paying special attention to the computational details. In particular, we will discuss some issues about the fuzzy-valued fitness function that is used in our formalism. A synthetic problem, plus two real-world datasets of low and medium complexities are also proposed, and used to evaluate the algorithm.

1 Introduction

The term "low quality data" refers to datasets where some or all of the features are imprecisely perceived. It embodies different concepts: censored data, binning, missing values, significant digits, confidence intervals, contradictory information from different sources, fuzzy numbers, linguistic information, and others.

We are interested in those cases where the imprecision in the perception of the data is defined by a family of confidence intervals. This includes most of the preceding situations as particular cases. For instance, we can model few significant digits, censored or missing data by means of a single interval that spans the range of the unknown measure: "the weight of an object is in $[1.1, 1.2]$" or "the pressure is lower than 3". We can also reconcile different measurements of the same feature by means of a set of confidence intervals [8]. Certain (but not all) kinds of fuzzy data can also be regarded as imprecise perceptions of crisp values, known through a family of α-cuts [1].

To make clear the extent of this concept, let us recall the statistical framework of a standard Artificial Intelligence-based classifier. The purpose of a classifier is to predict the class of an object, given the values of other properties of the same object. To that end, a probability distribution is defined on the set of objects, and the mentioned properties are assumed to be random variables. The classifier, in turn, is a decision rule that depends on the posterior probability of each class, given a vector of properties.

The algorithms in this paper differ from the standard case because we do not suppose that we can accurately observe all the properties of the object. Otherwise, the same assumptions hold. 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 certain probability. From a theoretical point of view, we hence understand that low quality data is a kind of data for which we can not achieve a precise knowledge about the posterior probability distributions induced by the mentioned random variables, but we can obtain families of probability distributions that are compatible with them. Accordingly, the mentioned decision rule can be inconclusive if the input data is not specific enough.

Recent works in fuzzy statistics suggest using a fuzzy representation when the data is known through a family of confidence intervals [1]. This representation assumes that a fuzzy set can be interpreted as a possibility distribution (which, in turn, is a family of probability distributions) and each α-cut of a fuzzy feature is a random set that contains the unknown crisp value of the feature with probability $1-\alpha$. [7, 8].

1.1 GFS and Fuzzy fitness functions

Our particular fuzzy representation of low quality data is tied to the use of a fuzzy or interval-valued measurement of the accuracy of a classifier or model. Let us use an example: we have a classification system, defined by these rules:

$$\begin{array}{l}\text{if } x<1 \text{ then class is } A\\ \text{if } x\in[1,2] \text{ then class is } B\\ \text{if } x>2 \text{ then class is } C\end{array} \qquad (1)$$

and the input that follows:

$$x < 1.8 \qquad (2)$$

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 $\{0, 1\}$. The number of errors of the whole classifier can be obtained by adding these individual errors with interval arithmetic operators. If the output of the classifier is a fuzzy set, the number of errors is a fuzzy number too, and we must use fuzzy arithmetic [6].

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 [3, 4, 11]. We have previous works where we have jointly optimized a mix of crisp and fuzzy objectives with genetic algorithms [7]. We have also proposed a number of different algorithms for learning regression models from low quality data and the fuzzy representation mentioned before [5, 10, 8]. However, to the best of our knowledge there

ISBN: 978-989-95079-6-8

```
function GFS
  Initialize population
  for iter in $\{1,\ldots,\mathrm{Iterations}\}$
    for sub in $\{1,\ldots,\mathrm{subPop}\}$
      Select parents
      Crossover and mutation
      assignConsequent(offspring)
    end for sub
    Replace the worst $\mathrm{subPop}$ individuals
    assignFitness(population,dataset)
  end for iter
  Purge unused rules
return population
```

Figure 1: Outline of the GFS that will be generalized [2]. Each chromosome codifies one rule. The fitness of the classifier is distributed among the rules at each generation.

```
function assignConsequent(rule)
  for example in $\{1,\ldots,\mathrm{N}\}$
    m = membership(Antecedent,example)
    weight[class[example]] $=$ weight[class[example]] + m
  end for example
  mostFrequent = 0
  for c in $\{1,\ldots,\mathrm{N_c}\}$
    if (weight[c]$>$weight[mostFrequent]) then
      mostFrequent = c
    end if
  end for c
  Consequent = mostFrequent
return rule
```

Figure 2: 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 [2].

have not been previous GFSs where those principles have been applied to learn classification problems.

1.2 Summary

The structure of this paper is as follows: in the next section we generalize the crisp GFS defined in [2] to low quality data. In Section 3 we evaluate the generalized algorithm in both crisp and imprecise datasets, and propose two real-world datasets of classification with imprecise data. The paper finishes with the concluding remarks, in Section 4.

2 Generalizing a Genetic Fuzzy Classifier to imprecise data

Generalizing a GFS to imprecise data involves, at the very least, changes in the inference mechanism and the fitness function, as we have discussed in [6]. In this section we will generalize the GFS outlined in Figure 1, which was introduced in [2]. This is a very compact algorithm that allows us to focus in the subject of this paper (extending Genetic Fuzzy Classifiers to imprecise data) without getting lost in the details.

Observe that this algorithm depends on two functions: “assignConsequent” (line 6) and “assignFitness” (line 9). These functions are also listed in Figures 2 and 3. This algorithm does not codify the consequent of the fuzzy rules in the genetic individual. Instead, the function “assignConsequent”

```
function assignFitness(population,dataset)
  for example in $\{1,\ldots,\mathrm{N}\}$
    winnerRule = 0
    bestMatch = 0
    for rule in $\{1,\ldots,\mathrm{M}\}$
      m = membership(Antecedent[rule],example)
      if (m$>$bestMatch) then
        winnerRule = rule
        bestMatch = m
      end if
    end for rule
    if (consequent(winnerRule)==class(example)) then
      fitness[winnerRule] = fitness[winnerRule] + 1
    end if
  end for example
return fitness
```

Figure 3: 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 [2].

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. In the remainder of this section, we will study the impact of the imprecise knowledge about the independent and dependent variables in the structure of the rules, and how to extend these two functions. That is to say: we analyze the reasoning method, the assignment of the consequents, the computation of a set-valued fitness and the genetic selection and replacement of the worst individuals.

2.1 Analysis of the reasoning method

The objective of the extended GFS is to obtain a fuzzy rule base from objects, when there is imprecise knowledge about some or all of the attributes of these objects. The set of classes that is produced when the input value is a fuzzy set can be computed in some different ways, but not all of them are consistent with our representation of an imprecise value.

Having an imprecise knowledge about the input variables differs from the standard case because:

1. The output of the FRBS will not be completely determined.
2. The number of errors of the FRBS in the training data will be partially known. The same happens if any other quality function is used instead of the number of errors, i.e. likelihood, logistic loss functions, etc.

Both issues have been introduced with the crisp classifier commented in the introduction: if the input is a set X that contains a range of inputs $x \in X$, the output is not a class but a set of classes:

$$\mathrm{class(X)} = \{\mathrm{class}(x) \mid x \in X\}. \tag{3}$$

The first difference is apparently a trivial issue: after all, one of the advantages of fuzzy rules is dealing with imprecision. Nevertheless, the standard reasoning method does not produce the set of classes that we need. To make our point clearer, let us study a fuzzy classifier comprising M rules:

$$\textbf{if } (x \textbf{ is } \widetilde{A}_i\ ) \textbf{ then class is } C_i, \tag{4}$$

and let us use the single-winner inference mechanism:

$$\text{class(x)} = C_{\arg\max_i\{\widetilde{A}_i(x)\}}. \tag{5}$$

Observe that, classifying the set X by means of the standard mechanism produces the class label that follows:

$$\text{class'(X)} = C_{\arg\max_i\{\min\{\widetilde{A}_i(x)|x\in X\}\}} \tag{6}$$

while we need this set of labels:

$$\text{class(X)} = \{C_{\arg\max_i\{\widetilde{A}_i(x)\}} \mid x \in X\} \tag{7}$$

which is different than 6. Furthermore, the fuzzy set of classes obtained when aggregating the rules, before the defuzzification stage, is not what we need either, because our set class(X) does not coincide with the fuzzy set

$$\sum C/\max\{\min\{\widetilde{A}_i(x) \mid x \in X\}|C_i = C\}. \tag{8}$$

The code we propose to use is included in lines 2–23 in Figure 5, that we will explain later.

It is remarked that, in this paper, each rule will contain a single consequent. This is a result of our interpretation of “low quality” as a family of confidence intervals. In words: in this paper, if a point is labeled as “class {A,C}” 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 “B”. Therefore, it makes not sense in this context to produce a rule with a double consequent, since this rule necessarily will have non-zero error at any example.

2.2 Assignment of consequents

```
function assignImpreciseConsequent(rule)
1    for example in $\{1,...,N\}$
2      $\widetilde{m}$ = fuzMembership(Antecedent,example)
3      weight[class[example]] $=$ weight[class[example]] $\oplus\ \widetilde{m}$
4    end for example
5    mostFrequent = $\{1,...,N_c\}$
6    for c in $\{1,...,N_c\}$
7      for $c_1$ in $\{c+1,...,N_c\}$
8        if (weight[c] dominates weight[$c_1$]) then
9          mostFrequent = mostFrequent - { $c_1$ }
10       end if
11     end for $c_1$
12   end for c
13   Consequent = select(mostFrequent)
return rule
```

Figure 4: If the examples are imprecise, we might not know the most frequent class label –lines 5 to 12–. In this paper we have used the dominance proposed in [4] to reduce this set to one element.

The assignment of consequents seen in Figure 2 is extended in Figure 4. 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 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 $a \in A, b \in B$) [11] 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 [4], that induces a total order and thus the set of nondominated consequents has size 1.

2.3 Computation of fitness

```
function assignImpreciseFitnessApprox(population,dataset)
1    for example in $\{1,...,N\}$
2      setWinnerRule = $\emptyset$
3      for r in $\{1,...,M\}$
4        dominated = FALSE
5        rule.$\widetilde{m}$ = 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) then
12         for sRule in setWinnerRule
13           if ($\widetilde{m}$ dominates sRule) then
14             setWinnerRule = setWinnerRule $-$\{ sRule \}
15           end if
16         end for sRule
17         setWinnerRule = setWinnerRule $\cup$\{ rule \}
18       end if
19     end for r
20     setOfCons= $\emptyset$
21     for sRule in setWinnerRule
22       setOfCons= setOfCons $\cup$\{ consequent(rule) \}
23     end for sRule
24     deltaFit= 0
25     if (\{class(example)\} == setOfCons and
           size(setOfCons)==1) then
26       deltaFit = \{1\}
27     else
28       if (\{class(example)\}$\cap$ setOfCons $\neq \emptyset$) then
29         deltaFit = $\{0, 1\}$
30       end if
31     end if
32     Select winnerRule $\in$ setWinnerRule
33     fitness[winnerRule] $=$ fitness[winnerRule] $\oplus$ deltaFit
34   end for example
return fitness
```

Figure 5: 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 to 19—, (b) these rules might have different consequent, thus we do not know if the rule base fails in the example –lines 20 to 31– and (c) we must assign credit to just one of these rules –lines 32 and 33–.

The output of the FRBS at the i-th object of the training set is a set of classes:

$$C_{\text{FRBS}}(X_i) = \{C_{\arg\max_j\{\widetilde{A}_j(x)\}} \mid x \in X_i\}. \tag{9}$$

The theoretical expression of the fitness function of the FRBS is:

$$\text{fitness} = \bigoplus e_i \tag{10}$$

where

$$e_i = \begin{cases} 1 & C_{\text{FRBS}}(X_i) = C_i \text{ and } \#(C_i) = 1 \\ 0 & C_{\text{FRBS}}(X_i) \cap C_i = \emptyset \\ \{0,1\} & \text{else} \end{cases} \tag{11}$$

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 Figure 5. 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 Figure 6 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 Figure 5, because of its lower cost.

2.4 *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 [2] 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. In both cases, we have used the uniform dominance defined in [4] to impose such a total order. We leave for future works the application of a multicriteria genetic algorithm similar to those used in our previous works in regression modeling [8, 10].

3 Numerical results

This section contains a brief numerical analysis of the generalized algorithm. We have performed the experiments that follow:

1. Synthetic datasets: Gaussian distribution, known Bayesian error, and different amounts of observation error.

2. Crisp datasets: three standard benchmarks, for testing that the extended algorithm has the same performance as the original version in crisp problems.

3. Imprecise, real world datasets: we propose two real world datasets, of small and medium size. One of them has been specifically designed for the purpose of this research, and the other is part of a practical problem of medical diagnosis.

```
function assignImpreciseFitnessExhaustive(population,dataset)
1    for example in $\{1, \ldots, N\}$
2      S = sample(example)
3      maxScore = 0
4      for s in S
5        winnerRule = 0
6        bestMatch = 0
7        for rule in $\{1, \ldots, M\}$
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] $\oplus$ 1
16       elif (consequent(winnerRule) $\subset$ class(example)) then
17         score[winnerRule] = score[winnerRule] $\oplus \{0, 1\}$
18       end if
19       if (max(score[winnerRule]) $>$ max(score[maxScore]))
20         then maxScore = winnerRule
21       end if
22     end for s
23     if (score[maxScore] $>$ 0) then
24       if (score[maxScore] == size(S)) then
25         fitness[maxScore] $=$ fitness[maxScore] $\oplus$ 1
26       else
27         fitness[maxScore] $=$ fitness[maxScore] $\oplus \{0, 1\}$
28       end if
29     end if
30   end for example
return fitness
```

Figure 6: 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.

All the datasets used in this paper are available in the website of the KEEL project: `http://www.keel.es`. 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.

3.1 *Synthetic datasets*

The set "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 10-cv experimental design was applied, and the mean values of the test errors are shown in Table 1. 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

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

Table 1: Results of the extended GFS in the synthetic dataset "Gauss" for crisp data (first column) and different degrees of observation error (second column). 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.

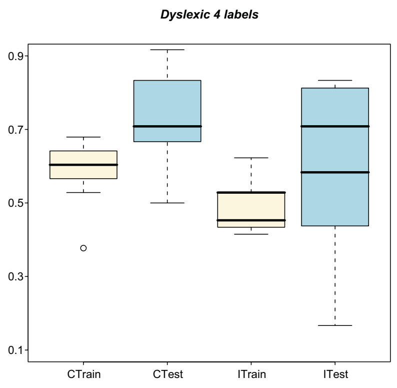

Figure 7: Boxplots illustrating the dispersion of the 10 repetitions of crisp and extended GFS in the problem "dyslexia-12", with 4 labels/partition. The boxplot of the imprecise experiments is not standard: we show respectively the 75% of the maximum and 25% percentile of the minimum fitness, thus the box displays at least the 50% of data; there are two marks inside the box, because the median of the data is an interval.

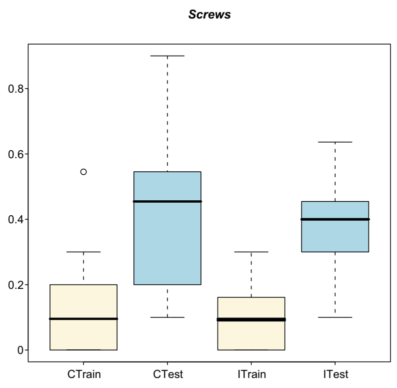

Figure 8: Boxplots illustrating the dispersion of the 10 repetitions of crisp and extended GFS in the problem "screws-50".

when the observation error is high ($\beta = 0.2$ and $\beta = 0.5$), nevertheless it still guides the evolution correctly.

3.2 *Crisp datasets*

Crisp datasets are included for assessing the performance of the generalized algorithm in standard problems. Since, in this case, the particularization of the algorithm to crisp data recovers the original algorithm in [2], the results are expected to be adequate. Nonetheless, we have included some estimations in Table 2. As expected, the results of both algorithms are similar; the differences are originated in the different random seeds.

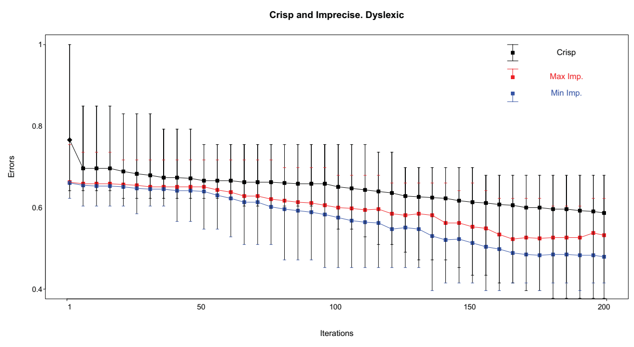

Figure 9: Compared evolution of crisp and imprecise GFS in the dataset "dyslexia-12". The ranges and means of 10 repetitions of the learning are shown, for both the crisp and the imprecise versions of the algorithm. The upper bound of the mean imprecise fitness is consistently lower than the mean of the crisp fitness.

3.3 *Real world datasets*

We propose two datasets for testing this and future learning algorithms with low quality data:

1. Dataset "Screws-50": 21 objects, 3 classes, 2 features (weight and length). We have weighed and measured 21 screws of three different types, taking into account the accuracy of the physical measurement: each feature is an interval. The class labels are precise. There are not outliers, i.e. 0% of error is attainable in absence of imprecision.

2. Dataset "Dyslexia-12": 65 objects, 4 classes, 12 features. This is a selection of the original dataset described in [9], 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

	Crisp		Low Quality		
Dataset	Train	Test	Exh. Train	Exh. Test	Approx. Train
Pima	0.254	0.287	[0.258,0.258]	[0.288,0.288]	[0.258,0.258]
Glass	0.323	0.365	[0.321,0.321]	[0.352,0.352]	[0.321,0.321]
Haberman	0.239	0.255	[0.238,0.238]	[0.248,0.248]	[0.238,0.238]

Table 2: Results in some crisp benchmarks, where the imprecise fitness function reduces to the crisp fitness function. The results of "crisp" and "low quality" columns are similar.

Crisp			Low Quality		
Dataset	Train	Test	Exh. Train	Exh. Test	Approx. Train
Screws-50	0.133	0.427	[0.096,0.096]	[0.377,0.377]	[0.068,0.106]
Dyslexia-12 (4 labels)	0.584	0.724	[0.481, 0.616]	[0.541, 0.675]	[0.477,0.516]
Dyslexia-12 (5 labels)	0.745	0.658	[0.581,0.667]	[0.608,0.641]	[0.537,0.549]

Table 3: Means of 10 repetitions of the generalized GFS for the imprecise datasets "Screws-50" and "Dyslexia-12" with 4 and 5 labels/variable

unknown.

We have compared the performance of the generalized algorithm to that of the original crisp algorithm. To that end, we have built a crisp dataset by removing the uncertainty in the imprecise dataset: each imprecise measurement was replaced by the mid-point of the corresponding interval, and those examples with imprecision in the independent variable were replicated for the different options. For instance, a point $(X = [1,3], C = \{A,B\})$ is converted into two points $(x = 2, c = A)$, $(x = 2, c = B)$.

We have used a 5x2cv design for the first problem, because of its small size, and 10cv for the second. The boxplots of the compared results, in both train and test sets, are depicted in Figures 7 and 8. 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 Figure 9 the ranges and means of 10 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.

4 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 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.

Acknowledgements

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, PCTI 2006-2009.

References

[1] Couso, I., Sánchez, L. Higher order models for fuzzy random variables. Fuzzy Sets and Systems 159: 237–258 (2008)

[2] Ishibuchi, H., Nakashima, T., Murata, T, A fuzzy classifier system that generates fuzzy if-then rules for pattern classification problems. In Proc. of 2nd IEEE International Conference on Evolutionary Computation, 759-764 (1995)

[3] Koeppen, M., Franke, K., and Nickolay, B., Fuzzy-Pareto-Dominance driven multi-objective genetic algorithm. in Proc. 10th International Fuzzy Systems Assotiation World Congress (IFSA), Istanbul, Turkey, 2003: 450–453. (2003)

[4] Limbourg, P., Multi-objective optimization of problems with epistemic uncertainty. in EMO 2005: 413–427. (2005)

[5] Sánchez, L., Otero, J., Villar, J. R., Boosting of fuzzy models for high-dimensional imprecise datasets. Proc. IPMU 2006, Paris, France: 1965–1973. (2006)

[6] Sánchez L., Couso I. Advocating the use of imprecisely observed data in genetic fuzzy systems IEEE Transactions on Fuzzy Systems 15 (4): 551–562. (2007)

[7] Sánchez, L., Couso, I., Casillas, J. Modelling vague data with genetic fuzzy systems under a combination of crisp and imprecise criteria Proc. 2007 IEEE Symp. on Comp. Int. in Multicriteria Decision Making, Honolulu, USA: 30–37. (2007)

[8] Sánchez, L., Couso, I., Casillas, J. Genetic Learning of Fuzzy Rules based on Low Quality Data. Fuzzy Sets and Systems. Submitted.

[9] Sánchez, L., Palacios, A., Couso, I., A Minimum Risk Wrapper Algorithm for Genetically Selecting Imprecisely Observed Features, applied to the Early Diagnosis of Dyslexia. Lecture Notes in Computer Science 5271, 608–615 (2008)

[10] Sánchez, L., Otero, J., Couso, I., Obtaining linguistic fuzzy rule-based regression models from imprecise data with multiobjective genetic algorithms. Soft Computing. 13(5): 467–479 (2008)

[11] Teich, J., Pareto-front exploration with uncertain objectives. in EMO, 2001: 314–328. (2001)