Learning Fuzzy Linguistic Models from Low Quality Data by Genetic Algorithms

Luciano Sánchez, *Member, IEEE,* and José Otero

***Abstract*— Incremental rule base learning techniques can be used to learn models and classifiers from interval or fuzzy-valued data. These algorithms are efficient when the observation error is small. 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, and that it does not make full use of all the available information. As an alternative, we propose a new implementation of a mutiobjective Michigan-like algorithm, where each individual in the population codifies one rule and the individuals in the Pareto front form the knowledge base.**

I. Introduction

There are many practical problems requiring to learn models from uncertain data. Coarse-grained digital data, as obtained when weighing small objects in a low resolution scale, routinely appears. Or, occasionally we are provided with values comprising both a numerical measure and one or more confidence intervals defining its imprecision (the position given by a GPS sensor, for example.) In either case, there is an unknown difference between the true measure and the observed one, but assuming that this difference is stochastic noise is an oversimplification. Intervals or fuzzy sets are best suited for representing the uncertainty in the observation. Furthermore, an interval or fuzzy-based representation can also be used to to reconcile different measurements of a feature in a given object [8] and, lastly, to describe incomplete knowledge about a value. For example, a missing input value can codified by an interval spanning the whole range of the variable.

In [7] we advocated the use of fuzzy data to learn and evaluate Genetic Fuzzy Systems, and raised the use of fuzzy-valued fitness functions to formulate the kind of problems mentioned before. When using imprecise data, the accuracy of a model becomes a fuzzy number, thus it is needed to optimize a fuzzy valued function, and sometimes a combination of crisp and fuzzy objectives [8]. Besides, we also proposed to combine backfitting and boosting techniques with the Iterative Rule Learning method in genetic fuzzy models [6] and classifiers [3], and lastly both approaches (backfitting and imprecise data handling) were combined in the definition of a learning algorithm for fuzzy models, based in incremental rule learning and the optimization a fuzzy-valued fitness function [9].

Luciano Sánchez and José Otero are with the Computer Science Department, University of Oviedo, Campus de Viesques, 33203 Gijón, Asturias, Spain (email: [luciano,jotero]@uniovi.es).

This paper is about datasets with medium to high discrepancies between the observed and the true values of the variables, such as those containing missing values or coarsely discretized data. It has been shown that the iterative algorithm in [9] is efficient in problems where the observation error is small. Unfortunately, we will show that it does not make full use of low quality data. As discussed in section III-B, the residual becomes less and less specific when new rules are added, and in certain cases this prevents discovering enough rules. As an alternative, we propose a new implementation of a mutiobjective Michigan-style algorithm, where each individual in the population codifies one rule, but all the individuals in the Pareto front comprise the knowledge base. In this paper, we will use an extension of the NSGA-II algorithm [2][8] rather than the simulated annealing introduced in [9].

The structure of this paper is as follows: in the next section we discuss certain issues about the representation of vague data with intervals. In section III we recall the use of backfitting techniques to learn fuzzy models from interval data, and summarize the modifications that must be done to the NSGA-II algorithm in order to optimize fuzzy valued functions. In section IV we describe the new algorithm, and benchmark its results in section V. The paper finishes with the concluding remarks, in section VI.

II. Low Quality Data

There exist problems where the correspondence between a vague input and an interval is not direct. For example, let us be given a training example $(X, Y) = (1.5, 3)$ and let us suppose that the tolerance of the measures in the input space is ± 0.5 units, and the measures are accurate in the output space (see Figure 1.) If we represent this imprecise example by mean of the interval-valued pair $(1.5 \pm 0.5, 3)$ we are stating that the desired value of the output for all the values in 1.5 ± 0.5 is 3, which is not true. What we really know is that there exist *at least one value* in the interval $[1, 2]$ whose output is 3. The proper representation of the granule of information in Figure 2 is the pair $([1, 2], [1, 4])$.

According to the relative uncertainty between inputs and outputs, we can classify all imprecise datasets into one of these three categories (see Figure 2):

I. The uncertainty in the outputs is lower than the span of the function in the range of the inputs.
II. The uncertainty in the outputs is higher than the span 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.

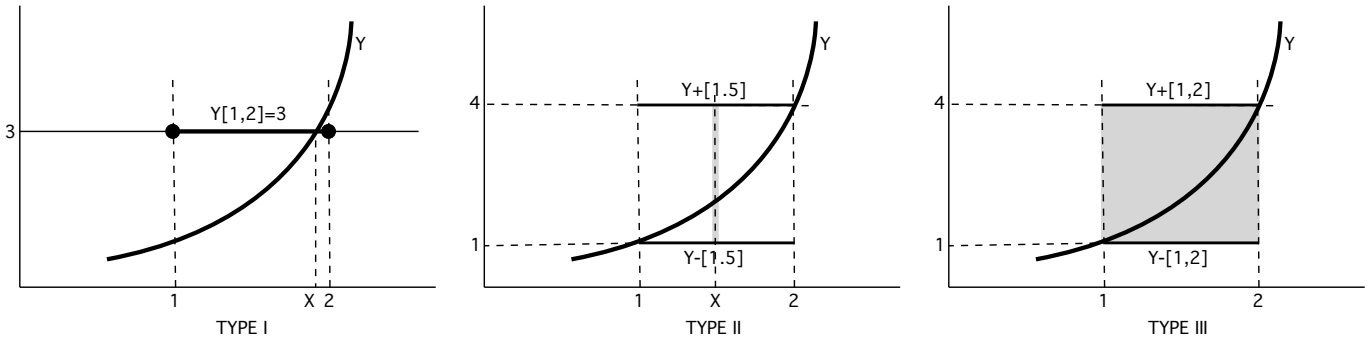

Fig. 2. Categories of vague datasets. Type I: The uncertainty in the outputs is lower than the span of the function in the range of the inputs: $Y[1,2] = 3$. Type II: The uncertainty in the input spans an interval contained in the range of the output: $Y(1.5) = [1,4]$. Type III: The output data is the set of all the images of the values compatible with the input: $Y[1,2] = [1,4]$

Problems of type III are rare. In point of fact, type I problems appear the most, since we seldom can quantify how much the observation error in the inputs is mapped to the output variable. Nevertheless, the most convenient case (from the point of view of rule learning) is that of a type II dataset with near crisp inputs. This is the case we will study in this paper, because any dataset can be transformed into that case without losing much information. 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. Think, for instance, in a problem with missing values. These unknown values can be replaced by their characteristic values with the method of choice, provided that the output is also expanded accordingly so that we are certain that it contains the true output of the model in that characteristic point.

III. Fuzzy Extended Additive Models

In this section we define the fuzzy reasoning method we will use, and recall the backfitting algorithm to which our proposal will be compared. We will restrict ourselves to linguistic additive fuzzy rule-based models, comprising M rules as the one that follows:

$$\texttt{If } x \texttt{ is } A_m \texttt{ then } y \texttt{ is } B_m \texttt{ with weight } w_m, \quad (1)$$

where x and y are the feature and the output vectors, respectively, and A_m are conjunctions of linguistic labels, which in turn are associated to fuzzy sets. B_m can be either a singleton or a fuzzy number. In this paper, none of the sets A_m, B_m will be modified during the learning, to preserve the linguistic interpretability, but we will admit that each rule is assigned a weight w_m. The output y of the fuzzy model is then computed as

$$y = G_{m=1}^{M}(w_m \cdot I(A_m(x), B_m(y))) \quad (2)$$

where I is a fuzzy inference operator, and G is an operator that combines the outputs of all the rules. Let us define I to be the product, and G the sum of the centroids. If the input x is crisp, the output of the fuzzy model is a real number, which can be written as

$$y(x) = \sum_{m=1}^{M} f_m(x) \quad (3)$$

Each function $f_m(x)$ is a product $\beta_m A_m(x)$. $A_m(x)$ is the membership of x to a linguistic expression whose terms are labels of the linguistic variables defined over the input variables, connected by the operators "AND" and "OR". β_m is the product of the centroid of B_m and the weight w_m assigned to the rule.

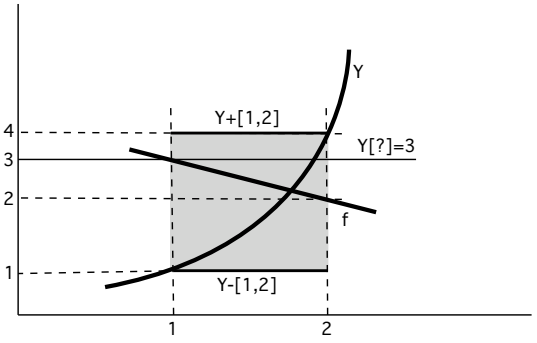

Fig. 1. The proper representation of the training example $(1.5 \pm 0.5, 3)$ is a pair $(1.5 \pm 0.5, [3 - \delta_1, 3 + \delta_2])$ for certain values $\delta_1(\epsilon)$ and $\delta_2(\epsilon)$ that depend on the excursion of f in $[1,2]$.

A. Backfitting Fuzzy Models

To learn the model (3) from data with the backfitting algorithm, we first fit one rule f_1 to the train set. Then, we replace the output variable in the train set by the residuals of the output of this rule, repeat the process to obtain f_2 and so on. A fuzzy rule f_m will be obtained in every iteration, and the process finishes when its β_m is near zero. Because of the limitations of space, the reader is refered to [6] for further details in the numerical procedure.

When the output data in the train set is fuzzy, the residual of the approximation of a rule to the data consists in minimizing a fuzzy-valued function. Following the ideas introduced in [7], given two fuzzy samples $\{\widetilde{X}_1, \ldots \widetilde{X}_N\}$

and $\{\widetilde{Y}_1, \ldots \widetilde{Y}_N\}$ of the input-output data, The residual of the model is

$$\widetilde{R}_k(\theta, \beta) = \widetilde{Y}_n \ominus \sum_{m=1}^{M} \beta_m A_m(\widetilde{X}_n) \tag{4}$$

and the best rule will be the one that minimices the *fuzzy valued* function

$$\tilde{F}_\beta(\theta) = \bigoplus_{n=1}^{N} \left(\widetilde{R}_k(\theta, \beta)\right)^2 \tag{5}$$

where $[X]_\alpha^2 = \{x^2 \mid x \in [X]_\alpha\}$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_M)$ is the parameter vector. The membership value $\tilde{F}_\beta(\theta)$ will be understood as the possibility of the value θ is the true error of the rule, thus θ is an unknown real number whose possibility distribution is given by the fuzzy fitness $\tilde{F}_\beta$:

$$\Pi(\theta|\beta) = \tilde{F}_\beta(\theta). \tag{6}$$

B. Incremental learning of rules in low quality data

Unless the input is crisp, the residual becomes less specific each time a new rule is added. Observe that the uncertainty in the input propagates to the output because of the right term of the subtraction in eq. (4), and each time the output data is replaced by the residual of a rule the situation is made worse, as $\widetilde{R}_k$ will be wider. Eventually, the residual will cover the null value for all the points. As we will discuss later, a method that evaluates all rules at the same time (Pittsburgh or Michigan) would be potentially better for problems with fuzzy inputs.

C. Using NSGA-II for optimizing uncertain objectives

In [8] the NSGA-II algorithm [2] was extended so that it can find a set of nondominated solutions for a two-objective problem, where one of the objectives is crisp (the complexity of the fuzzy rule base) and the other one is an interval. This extension, which involves changes in three modules, is used in this paper. These modules are the precedence (dominance) operator, the non-dominated sorting of the individuals, and the crowding distance.

1) Precedence under imprecise fitness: To determine whether one individual precedes another, it is needed to set up a procedure that, given two imprecise observations $\tilde{F}_{\beta_1}$ and $\tilde{F}_{\beta_2}$ of two unknown fitness values θ_1 and θ_2, estimates whether the probability of $\theta_1 < \theta_2$ is greater than that of $\theta_1 \geq \theta_2$, thus $\tilde{F}_{\beta_1} \prec \tilde{F}_{\beta_2}$. 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 $\tilde{F}_{\beta_1}$ and $\tilde{F}_{\beta_2}$ that makes us to prefer one of them (thus $\tilde{F}_{\beta_1} \parallel \tilde{F}_{\beta_2}$.)

If a joint probability $P((\theta_1, \theta_2)|(x_1, x_2))$ were known, comparing two individuals would be an statistical decision problem. The imprecise fitness provides us with less information than this probability distribution, because it is only an upper probability, that dominates the posterior probability of the crisp fitness. In other words, given an individual x, the information we have about its fitness takes a value θ is limited to

$$\tilde{F}_\beta(\theta) = P^*(\theta|\beta) \geq P(\theta|\beta) \tag{7}$$

and the decision rule is

$$\frac{P_*(\{(\theta_1, \theta_2) : \theta_1 < \theta_2\}|\beta)}{P^*(\{(\theta_1, \theta_2) : \theta_1 \geq \theta_2\}|\beta)} > 1. \tag{8}$$

In the reference [8] three different implementations of this decision rule were introduced. We will use here the one that introduces an uniform prior, i.e $\tilde{F}_{\beta_1} \prec \tilde{F}_{\beta_2}$ if

$$\frac{\sum_{\theta_1 < \theta_2} \tilde{F}_{\beta_1}(\theta_1)\tilde{F}_{\beta_2}(\theta_2)}{\sum_{\mathbf{R}} \tilde{F}_{\beta_1}(\theta_1)\tilde{F}_{\beta_2}(\theta_2)} > 1. \tag{9}$$

2) Non Dominated Sorting: Finding the best individual is a procedure of statistical decision that generalizes the rule shown in the preceding section. Observe that we can bound the lower probability of the assert "the i-th individual has the best fitness in the population" by

$$M_i = P_*(\theta_i \text{ is the minimum}) = \prod_{j=1}^{s} P_*(\theta_i < \theta_j) \quad i \neq j$$

where $P_*(\theta_i < \theta_j) = 1 - \max\{\tilde{F}_{\beta_1}(\theta_1) \cdot F_{\beta_2}(\theta_2) : \theta_1 \geq \theta_2\}$. Therefore, sorting the population w. r. t. an imprecise criterion is the same as ordering the values of M_i.

3) Crowding distance: We have chosen the Hausdorff distance between the expectation of both fitness values. Given two intervals A and B, this distance is defined as

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

The crowding distance is defined as the distance between the nearest (as defined by the Hausdorff metric) individual preceding I_i and the nearest individual following I_i. 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.

IV. A multiobjective Michigan-style Genetic Fuzzy System able to learn Fuzzy Models from imprecise data

In this section we propose to use a Michigan-stype algorithm instead of the iterative approach discussed before. While the catalog of this kind of algorithms for classification problems is wide, their application to modeling problems is less studied [1], [5], [10], [11], and there are not, up to our knowledge, previous formulations of Michigan models for imprecise data.

The rationale behind our method is as follows: we assume that the best model will comprise rules that are in nondominated sets under confidence and support measures [4]. Therefore, we will use the extension of the NSGA-II algorithm described in the preceding section to obtain successive approximations to this set of nondominated rules. Every generation, by means of a weighting and selection procedure that will be explained later in this section, we

```
Initialize P
Evaluate confidence and support in P
SVD select P
Create Intermediate Population Q
while iter ≤ maxiter
    Evaluate confidence and support in P+Q
    SVD select P+Q
    non dominated sort P+Q
    compute crowding distance P+Q
    P ← selection of P+Q
    SVD select P
    non dominated sort P
    Create Intermediate Population Q
end while
Output the nondominated elements of P
```

Fig. 3. Pseudocode of the NMIC algorithm

assign a weight to each rule in the Pareto front and solve the cooperation-competition problem. The set of rules that remain with non-null weights will comprise the knowledge base of the fuzzy model. The pseudocode of the NSGA-II based Michigan algorithm (NMIC) is shown in Figure 3. The same codification and genetic operators described in [9] were used.

A. Fitness function: Confidence and support

The fitness of an individual has three components: the support of the antecedent of the rule, its confidence and the weight of the rule.

The usual definition of support (the sum of the memberships of the antecedent for all the points in the sample) is applicable. It is remarked that, in our case, the support is an imprecise value when the inputs are interval or fuzzy.

The confidence is used to compare the degree to which a rule explains all the examples that it covers. In modeling problems, and assuming that the consequent of a rule is constant, we will consider that the concept "fraction of negative examples," matches the variance of the covered examples, weighed by the membership of these examples to the antecedent of the rule. Thus, we propose to use this approximate expression to assign a confidence value to a rule (the lower the better):

$$\begin{cases} \bigoplus [\widetilde{Y}_n \ominus w A_m(\widetilde{X}_n)]^2 \otimes A_m(\widetilde{X}_n) & \exists\ n \mid A_m(\widetilde{X}_n) > 0 \\ \bigoplus [\widetilde{Y}_n]^2 & \text{otherwise} \end{cases} \tag{10}$$

where w is the solution to the weighted least squares problem defined by the centers $\overline{X}_n$ of the data:

$$w = \frac{\sum_{n=1}^{N} Y_n A_m(\overline{X}_n)^2}{\sum_{n=1}^{N} A_m(\overline{X}_n)^3}. \tag{11}$$

It is remarked that the values w are only used to compute the confidence. The weights of the rules are computed as described in the section that follows.

B. Cooperation-competition problem: SVD selection

The crux of this method is the assignment of weights and selection of rules from the Pareto front. This stage is needed because the cooperation of the rules only arises if the sum of the fitness values of the individuals is comonotonic with the fitness of the rule base they form. Otherwise, the genetic evolution would not improve the error or the model. However, this is not the case with our confidence and support based measures: each rule in the population models the part of the space covered by its antecedent, but nothing prevents that more than one rule covers the same area while other areas are uncovered.

We have solved this problem by assigning a weight to each rule in the Pareto front. Resembling the fitness assignment in Michigan classifier systems, where two identical rules may be given different credit, we intend to assign a null weight to all the redundant rules. Our purpose is to obtain the most compact rulebase, and also that these weights produce the best matching between the data and the model.

Again, as we have done in the preceding section, we can obtain these weights by least squares. Let $A = [a_{nm}]$ be the matrix of the memberships of the antecedents of all rules in the Pareto front, at the centerpoint 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

$$K = (A^t A)^{-1} A^t Y \tag{12}$$

provided that the rank r_A of A coincides with its number of columns, the number of individuals in the Pareto front. In most cases, r_A is lower than this, therefore $C = A^t A$ does not have inverse. The common solution to this problem is to apply a singular value decomposition $C = UDV^t$, then cancel the eigenvalues of D lower than 10^{-6} times the highest, and take the inverse of the remaining ones, and by last define

$$K = [V \cdot (1/D) \cdot U^t] A^t Y. \tag{13}$$

While this assignment solves the cooperation problem, it does not solve the competition problem, because the redundant rules are not discarded. The value of K in the preceding equation if that of minimum norm, but what we really need is the definition of the matrix $(1/D)$ that produce *the most sparse* definition of K, not that with the lowest weights. For example, observe that the definition in eq. 13 will assign the same weight to identical rules, but we want 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 in the Pareto front.

As a matter of fact, the number of individuals we want to assign 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 and 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, i.e, we will not find more than r_A independent elements in

the Pareto front. 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 will be set to nonzero weight.

The solution is trivial, but computationally intensive: we just need to compute the eigenvalues of all the submatrices A' of A formed by removing only one of its columns. When it is found a submatrix A' whose 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. (12) to obtain the weights of the r_A rules in the base, thus solving the competition problem.

C. Non dominated sorting

The precedence between individuals is based on weight, confidence and support. The treatment of the three criteria is not symmetric:

1) An individual which has a weight different than zero precedes another whose weight is cero.
2) In case that two individuals have both zero weight or both nonzero weight, we use a nondominance-based order depending on the two criteria 'confidence' and 'support' defined before. At least one of them is imprecise, thus we use the ranking defined in [8].

V. Numerical analysis

8 synthetic problems and two real world problems have been used to benchmark the algorithms proposed in this paper. The synthetic problems are based on the functions $f_1(x,y) = x^2 + y^2$ (high slope) and $f_2(x,y) = 10(x - xy)/(x - 2xy + y)$ (low slope). It is expected that the improvements of the GFS in this paper are more significant in f_1, for its slope magnifies the effect of the imprecision in the inputs. f_i-y are the functions f_i with y% of gaussian noise, and are used to model the combined effects of stochastic error and observation error. "Building" and "Cable" are real-world problems, of moderate and small sizes, respectively. The results shown are the mean test values of 10 repetitions of the experiments.

In the first place we have quantified the effect of using type-I datasets without preprocessing the output, that we claimed in section II that could cause overtraining. In Table I we have plotted the test error when the input data has tolerances $\pm 1\%$, $\pm 2\%$ and $\pm 3\%$. As expected, when the observation error is combined with a high slope in the function (function f_1 and "cable" problem,) that overtraining happened. The train errors (not displayed in the table) were not affected.

The second benchmark measures the accuracy of this method when train data is crisp, so we can compare it to different fuzzy rule learning and statistical methods in the literature. Wang and Mendel with importance degrees 'maximum' (WM1), 'mean' (WM2) and 'product maximum-mean' (WM3), the same three versions of Cordón and Herrera's method (CH1, CH2, CH3), Nozaki, Ishibuchi and Tanaka's (NIT), Linear (LIN) and Quadratic regression (QUA) , Neural Networks (NN) and TSK rules induced with Weighted Least Squares (WLS) are compared to genetic backfitting (BFT), MOSA backfitting (BMNO) and NSGA-II based Michigan (NMIC). The best overall result and the most accurate fuzzy rule base were boldfaced in Table III. All the methods and datasets not described in this paper are referenced in [6] and [9]. Because of space reasons we do not include the statistical tests, but a selection of boxplots in Figure 4.

	0%	1%	2%	3%
f_1	0.13	0.22	1.34	5.30
f_1-10	1.60	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
cable $\cdot 10^{-3}$	416	416	416	890

TABLE I

Test error when the input data has tolerances $\pm 1\%$, $\pm 2\%$ and $\pm 3\%$. When the observation error is combined with a high slope in the function (function f_1 and "cable" problem,) the learning algorithm overtrains.

	1% BMO	1% NMIC	5% BMO	5% NMIC	10% BMO	10% NMIC
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
cable	440	**421**	581	**558**	1003	**988**

TABLE II

Comparison of NMIC and BMO 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. NMIC improved the results in all the tests for which the observation error was the most relevant source of noise.

Lastly, to compare the performance of the NMIC method with that of the iterative learning, we have prepared datasets with 1%, 5% and 10% of interval-valued imprecision in the outputs, and tested the rule bases with crisp data with non-zero mean observation error (all the test points were in the upper extreme of the tolerance.) We have only used datasets for which the stochastic error is lower than the observation error, and the results are shown in Table II. As expected, NMIC improved the results in all the tests for which the observation error was the most relevant source of noise.

VI. Concluding remarks

This paper addressed some problems of learning fuzzy rules from imprecise data (namely, how to make full use of low quality data). Besides, there are some basic questions that remain unanswered, as the best way of preprocessing a

	WM1	WM2	WM3	CH1	CH2	CH3	NIT	LIN	CUA	NEU	WLS	BFT	BMO	NMIC
f_1	5.65	5.73	5.57	5.82	8.90	6.93	5.63	130.5	**0.00**	0.17	0.09	0.45	0.30	**0.13**
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	**1.59**
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**	**5.98**
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**	39.22
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	**0.22**
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	**0.35**
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	**0.86**
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	3.69
cable $\cdot 10^{-3}$	778	720	723	673	663	655	548	418	**393**	522	486	441	437	**416**
building $\cdot 10^{2}$	1.113	1.051	1.023	0.983	1.753	1.465	0.432	0.477	-	0.276	**0.246**	**0.375**	0.389	0.389

TABLE III

COMPARATIVE RESULTS BETWEEN ADDITIVE REGRESSION + GENETIC BACKFITTING (BFT), MOSA-BASED BACKFITTING (BMO) AND OTHER APPROACHES. BFT METHOD WAS LIMITED TO 25 FUZZY RULES. THE BEST OF WM, CH, NIT, BFT AND BMO, PLUS THE BEST OVERALL MODEL, WERE HIGHLIGHTED FOR EVERY DATASET.

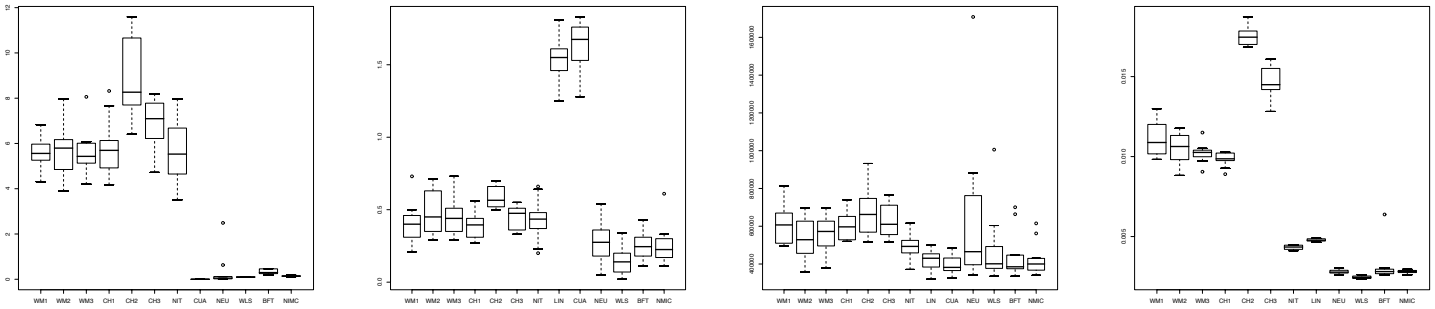

Fig. 4. Comparative results between additive regression + backfitting (BFT) and multiobjective fuzzy Michigan (NMIC). over f_1, f_2, cables and building datasets.

dataset with missing values, i.e., how to compute the spread of the output when an interval in the input is replaced by one point (generally speaking, how 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.

The implementation of the new algorithm NMIC could also be improved in future releases. It worths mentioning that the initialization the population is currently random, so most points are uncovered in problems with many inputs. We might study an hypothetical 'interval-valued' Wang-Mendel initialization, that would be good in terms of confidence but would produce rules far from Pareto-optimal ones in terms of support. In addition, the search of the Michigan algorithm could be further guided toward points with high residuals.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Bonarini, A. (1993). "ELF: learning incomplete fuzzy sets for an autonomous robot". In Proc. EUFIT' 93. Aachen, Germany, pp. 69-75.

[2] Deb, K., Pratap, A., Agarwal, S., and Meyarevian,T., (2002) *A fast and elitist multiobjective genetic algorithm: NSGA-II,* IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197.

[3] del Jesús, M. J., Hoffmann, F. , Junco, L., Sánchez, L. (2004) *Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms*. IEEE T. Fuzzy Systems 12(3): pp. 296-308.

[4] Ishibuchi, H, Kuwajima, O., Nojima, Y. (2007) "Relation between Pareto-optimal fuzzy rules and Pareto-optimal fuzzy rule sets". Proc. 2007 IEEE Symp. on Comp. Int. in Multicriteria Decision Making, Honolulu, USA, To appear.

[5] Parodi, A. and Bonelli, P (1993) A new approach to fuzzy classifier systems. In Proc Fifth International Conference on Genetic Algorithms (ICGA'93). pp. 223-230.

[6] Sánchez, L. Otero, J. (2004) A fast genetic method for inducting descriptive fuzzy models. Fuzzy Sets and Systems 141(1): 33-46

[7] Sánchez, L., Couso, I. (2007) *Advocating the use of Imprecisely Observed Data in Genetic Fuzzy Systems*. IEEE Trans Fuzzy Sys, 2007. In press.

[8] Sánchez, L., Couso, I., Casillas, J. (2007) "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, To appear.

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

[10] Valenzuela-Rendón, M. (1991) "The fuzzy classifier system: A classifier system for continuosly varying variables." In Proc. ICGA'91, San Diego, USA, pp. 346-353

[11] Velasco, J. R., (1998) *Genetic-based on-line learning for fuzzy process control*. International Journal of Intelligent Systems. 13(10-11), 891-903