Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Fuzzy Sets and Systems 141 (2004) 33–46

FUZZY sets and systems

www.elsevier.com/locate/fss

A fast genetic method for inducting descriptive fuzzy models

Luciano Sánchez*, José Otero

Depto. Informática, Universidad de Oviedo, Campus de Viesques, 33203 Gijón, Spain

Abstract

Under certain inference mechanisms, fuzzy rule bases can be regarded as extended additive models. This relationship can be applied to extend some statistical techniques to learn fuzzy models from data. The interest in this parallelism is twofold: theoretical and practical. First, extended additive models can be estimated by means of the matching pursuit algorithm, which has been related to Support Vector Machines, Boosting and Radial Basis neural networks learning; this connection can be exploited to better understand the learning of fuzzy models. In particular, the technique we propose here can be regarded as the counterpart to boosting fuzzy classifiers in the field of fuzzy modeling. Second, since matching pursuit is very efficient in time, we can expect to obtain faster algorithms to learn fuzzy rules from data. We show that the combination of a genetic algorithm and the backfitting process learns faster than ad hoc methods in certain datasets.

Keywords: Descriptive fuzzy models; Genetic fuzzy systems; Boosting algorithms; Backfitting; Matching pursuit

1. Introduction

Fuzzy modeling is used to model a system making use of a descriptive language based on fuzzy logic with fuzzy predicates [22]. Most of times, fuzzy rule based systems are considered. Rule based models can be constructed by means of different system identification techniques, and their quality can be measured in terms of interpretability and accuracy.

As pointed out in [2], current tendencies in fuzzy modeling look for a good balance between interpretability and accuracy. This paper can be included in this tendency. We propose a new method for inducting descriptive fuzzy models. Our model is based on fuzzy propositions that only use prespecified semantic values of linguistic terms. Rules combining these propositions do not necessarily use all input variables, as the algorithm performs a rule-based feature selection. Furthermore, to preserve linguistic interpretability, membership functions are not tuned, besides accepting that rule consequents are weighted.

* Corresponding author. Tel.: +34-85-182130; fax: +34-85-181986.
E-mail address: luciano@lsi.uniovi.es (L. Sánchez).

0165-0114/$ - see front matter
doi:10.1016/S0165-0114(03)00112-X

Our method is based on the similarities that exist, under certain fuzzy reasoning methods, between fuzzy models and extended additive models. If a fuzzy model can be regarded as an extended additive model, matching pursuit algorithms [14] can be applied to it: this means we can obtain a sparse solution, which results in a low number of fuzzy rules, using a fast algorithm. Moreover, there exist connections between this family of algorithms and state of the art machine learning methods such us support vector machines [1] and boosting [20] which would justify by themselves the comparative study of additive and fuzzy models.

1.1. Summary

The summary of this paper is as follows: in the next section, additive models, as well as generalized and extended additive models, are introduced, and their relationship with boosting algorithms and fuzzy models explained. Section 3 contains a detailed description of all steps in the learning algorithm, paying special attention to the search of the intermediate models, which will be accomplished by a genetic algorithm. Section 4 contains numerical results contrasting the performance of our algorithm, in both the speed of the process and the accuracy of the result.

2. Additive models and boosting

2.1. Generalized additive models

Additive models were introduced in the 1980s to improve precision and interpretability of classical nonparametric regression techniques in problems with a large number of inputs [21]. These models estimate an additive approximation to the multivariate regression function, where each of the additive terms is estimated using a univariate smoother.

Individual terms explain the dependence of the output variable with respect to their corresponding input variables, thus there exists a certain degree of interpretability in the model. While this kind of estimation avoids the curse of dimensionality, it is not able to approximate universally. Hastie and Tibshirani addressed this issue and proposed generalized additive models [11]. With these later models it is assumed that the mean of the output depends on a sum of terms through a nonlinear link function, and it is permitted that the response probability distribution is any distribution in the exponential family. Many statistical models belong to this class, including additive models for Gaussian data and nonparametric logistic models for binary data.

More formally, let y be the output random variable we wish to model, and let $x=(x_1,\ldots,x_n)$ be the input random vector. The objective of the modeling process consists in estimating the conditional expectation of y given x. Linear regression assumes

$$E(y|x) = f(x_1,\ldots,x_n) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{1}$$

and obtains $\beta_0,\ldots,\beta_n$ by least squares. Additive models generalize this schema by allowing the use of a sum of nonlinear univariate regressors

$$E(y|x) = f(x_1,\ldots,x_n) = s_0 + s_1(x_1) + \cdots + s_n(x_n), \tag{2}$$

where s_i are smooth functions that are estimated in a nonparametric fashion. Generalized additive models extend additive models by not assuming a Gaussian distribution of the output, but any probability distribution in the exponential family,

$$f_y(t;\theta;\phi) = \exp\left\{\frac{t\theta - b(\theta)}{a(\phi)} + c(t,\phi)\right\} \tag{3}$$

and making the additive component

$$f(x_1,\ldots,x_n) = s_0 + s_1(x_1) + \cdots + s_n(x_n) \tag{4}$$

to depend on the mean of the output by means of a link function g, so that $g(E(y|x)) = f(x_1,\ldots,x_n)$. The most commonly used link function in practice is the canonical link $g(E(y|x)) = \theta$.

2.2. Extended additive models and matching pursuit

Additive models can be generalized furthermore. In extended additive models, the univariate regressors s_i are replaced by functions of more than one feature. In our context, these functions usually depend on a set of parameters γ and a multiplier β,

$$s_i = \beta_i s(x;\gamma_i) \tag{5}$$

thus the additive model becomes

$$E(y|x) = f(x_1,\ldots,x_n) = s_0 + \sum_{i=1}^{n} \beta_i s((x_1,\ldots,x_n);\gamma_i). \tag{6}$$

For example, in radial basis neural networks the functions $s(x,\gamma_i) = \exp\{\|x-\gamma_i\|^2\}$ are the "basis functions"; γ_i are their centers and β_i are the weights that connect the input layer with the output. In support vector machines, $s(x,\gamma)$ is a kernel, and γ_i are the support vectors. We will show later that a fuzzy rule base can be casted in the same schema under certain mechanisms of approximate reasoning.

Extended additive models can be learned with a generalized backfitting algorithm [9]. Given a cost function d, that measures the differences between the conditional expectation and its approximation, this algorithm consists in finding n pairs of values $\{\beta_m,\gamma_m\}$ minimizing each

$$E\left[d\left(y, \sum_{\substack{k=1\ldots n\\ k\neq m}} \beta_k s(x;\gamma_k) + \beta s(x;\gamma)\right)\right] \tag{7}$$

with respect to β,γ [9]. A greedy approach, where the expectation of the output is incrementally approximated, produces good results in practice. Let $f_0(x), f_1(x),\ldots$ be successive approximations

to $E(y|x)$; then, let us define

$$\{\beta_m, \gamma_m\} \leftarrow$$
$$\arg\min_{\beta,\gamma} E[d(y, f_{m-1}(x) + \beta s(x;\gamma))], \tag{8}$$

where $\{\beta_k, \gamma_k\}_1^{m-1}$ are fixed at their corresponding solution values at earlier iterations.

Algorithms that learn a weighted sum of basis functions, by sequentially appending functions to an initially empty basis, to approximate a target function in the least-squares sense, are contained in the family of the "matching pursuit" algorithms [14]. These algorithms have been compared to support vector machines [27] and radial basis neural networks in machine learning problems [23]. One of the most interesting properties of matching pursuit algorithms is that they are good in keeping the sparsity of the solution; this improves the generalization properties of the method and we will also see in the following sections that the same property guarantees a small number of rules in the fuzzy case that will be described later.

2.3. Fuzzy models and extended additive models

Under certain fuzzy reasoning methods, fuzzy models are extended additive models. Consider a fuzzy rule based model comprising M weighted rules

$$\texttt{If } X \texttt{ is } A_m \texttt{ then } Y \texttt{ is } B_m \texttt{ with } [w_m], \tag{9}$$

where X and Y are the feature and the output vectors, respectively; A_m and B_m are conjunctions of linguistic labels, which in turn are associated to fuzzy sets, and w is a real-valued rule weight. This notation has been used before, to extend fuzzy inference methods [25], explain fuzzy neural networks [3] and improve linguistic modeling [16]. The semantic of (9) can also be expressed with an alternate linguistic expression. Following the syntax used in quantified fuzzy sentences [7], the sentence

$$w_m \texttt{ of } A_m \texttt{ are } B_m \tag{10}$$

gives the same information as rule (9), besides w_m is a real value here, instead of a linguistic modifier. For example, the rule "If weight is HIGH then height is MEDIUM with [0.2]" can also be written as "20% of HIGH weighted are MEDIUM heighted." This last expression can also be given a probabilistic interpretation [19], and is compatible with the original definition by Zadeh of a conditioned πp-fuzzy granule [26]

$$\texttt{If } X \texttt{ is } A_m \texttt{ then } Y \texttt{ is } B_m \texttt{ is } \lambda \tag{11}$$

with a degenerate fuzzy probability λ expressed by the following possibility distribution over the unit interval:

$$\Pi(x) = \begin{cases} 1 & x = w_m, \\ 0 & \text{else.} \end{cases} \tag{12}$$

Given a linguistic model comprising rules like (10) (or (11)) and an input value x, the output can be computed with different methods. Let us use the "first infer, then aggregate" fuzzy

reasoning [6]. The inference process begins calculating the set B' which is the result of the inference process for every rule and the input x:

$$\mu_{B'}(y) = I(\mu_{A_m}(x), [\mu_{B_m}(y) \wedge w_m]), \tag{13}$$

where I is a fuzzy inference operator, and the membership of the consequent is limited by the degree of truth of the rule. The output $f(x)$ of the fuzzy model is then computed as

$$f(x) = G_{m=1}^{M}(D(\mu_{B'_m}(y))), \tag{14}$$

where D is a defuzzification operator, and G is an operator that combines all values $D(\mu_{B'_m}(x, y))$ to produce the final output. Let I and $\wedge$ be the product, D the centroid and G the weighted sum. Then

$$f(x) = \sum_{m=1}^{M} w_m \cdot \mu_{A_m}(x) \cdot D(\mu_{B_m}(y)). \tag{15}$$

Since $w_m \cdot D(\mu_{B_m}(y))$ is a real value and $\mu_{A_m}(x_0)$ is a nonlinear function of the inputs, it is immediate that the function $f(x)$, for this particular choice of t-norm, aggregation and defuzzification operators, is an extended additive model.

2.3.1. Backfitting rule bases

The particularization of the backfitting algorithm to such a fuzzy model consists in identifying:

- The membership functions $\mu_{A_m}(x)$ with the multivariate predictors $s_m(x) = s(x, \gamma_m)$.
- the products $w_m \cdot D(\mu_{B_m}(y))$ with the multipliers β_m.

If least-squares is used as a fitting criterion, back-fitting consists in finding $\{\beta_m, \gamma_m\}$ minimizing

$$E\left[y - \sum_{k \neq m} \beta_k s(x; \gamma_k) - \beta s(x; \gamma)\right]^2 \tag{16}$$

with respect to β, γ. There will be a finite number of values for γ, each one of them pointing to one of the fuzzy sets that can be constructed by combining linguistic terms in the input. Observe that, after obtaining the value of β, it must be converted into a product of two constants: a confidence w_m and the defuzzified value of one of the terms in the output variable. Finally, the greedy approach is

$$\{\beta_m, \gamma_m\} \leftarrow$$
$$\arg\min_{\beta,\gamma} E[y - f_{m-1}(x) - \beta s(x; \gamma)]^2, \tag{17}$$

where $\{\beta_k, \gamma_k\}_1^{m-1}$ are fixed at their corresponding solution values at earlier iterations, as mentioned before.

2.4. Matching pursuit in additive models and boosting

While the preceding analysis considered regression models, generalized additive models have been used to solve classification problems. In particular, the output of a committee of boosted classifiers [20] can be expressed nearly as shown in the preceding section.

1. Set $f_0(x) = 0$, $p_0(x) = 1/2$.
2. For step number $i = 1, \ldots, M$
   (a) Compute $\beta_i s_i(x) = \text{smooth}[\frac{y - p_{i-1}(x)}{p_{i-1}(x)(1-p_{i-1}(x))}]$ with weights $p_{i-1}(x)(1 - p_{i-1}(x))$.
   (b) Update $f_i(x) = f_{i-1}(x) + \beta_i s_i(x)$
   (c) Compute $p_i(x) = \frac{e^{f_i(x)}}{1+e^{f_i(x)}}$
3. Output class$(x) = \arg\max(p_M(x), 1 - p_M(x))$

Fig. 1. Pseudocode of backfitting applied to a logistic extended additive model. After solving step (2) as discussed in the text, the outline of the algorithm is nearly identical to that of Adaboost procedure.

The analogy between boosting and extended additive models is strong, because there exist similarities in both their structure and their respective learning algorithms. The objective of a binary classification problem is to approximate the value $p(y = 1|x)$, which we will denote by $p(x)$. The response variable in a classification problem follows the binomial distribution, and the link function is $g(p(x)) = \log(p(x)/(1 - p(x)))$ [11]; therefore, the additive model is

$$\log \frac{p(\text{class}(x) = 1)}{p(\text{class}(x) = 0)} = f(x_1, \ldots, x_n) = s_0 + \beta_1 s_1(x) + \cdots \tag{18}$$

and the output of the model, reversing the logistic transform,

$$p(x) = \frac{e^{f(x)}}{1 + e^{f(x)}}. \tag{19}$$

If the greedy version of generalized backfitting, mentioned in the preceding section, is applied to this model, one obtains an algorithm very similar to Adaboost. An outline of this algorithm is shown in Fig. 1. The "smooth" operation [11], consists in estimating the values β_i and γ_i on which the i-th additive term depends, by means of a suitable statistical or machine learning procedure. Every step can be understood as fitting a new term to a weighted set of residuals of the previous submodel. This residual is $z = (y - p_{i-1}(x))/(p_{i-1}(x)(1 - p_{i-1}(x)))$, and the weight of the residual at the element x in the sample is $p_{i-1}(x)(1 - p_{i-1}(x))$.

It is controversial to decide whether this algorithm (which eventually was renamed to "Logitboost" by its author) is a new boosting algorithm, based on different principles than Adaboost, or, on the contrary, Adaboost can be regarded as an approximation to the combination of the local modeling and backfitting algorithms used in Logitboost, as claimed in [9]. Since in this paper we apply the backfitting algorithm to learn linguistic fuzzy rules from data, we can consider that our method can be regarded as a boosting-like algorithm applied to a committee of models (instead of being applied to a committee of classifiers) and in this sense it complements previous works where either approximate or descriptive fuzzy rules were learned with fuzzy extensions of the Adaboost algorithm [13,12].

3. Proposed methodology

Let us revisit the greedy method introduced at the end of Section 2.3.1. Observe that the term $\beta_m s(x;\gamma_m)$ is chosen to minimize the squared difference

$$E[y - f_{m-1}(x) - \beta s(x;\gamma)]^2, \tag{20}$$

where $y - f_{m-1}(x)$ is the residual of the previous model at x. Having this point in mind, let us rewrite the procedure in algorithmic nomenclature:

```
residual[1..N] = Y[1..N]
rule base = emptyset
repeat
     R = fit one rule (X, residual)
     do i=1..N
         residual [i] =
         residual[i] - inference (R,X[i])
     end do
     rule base = rule base + R
until rule base contains enough rules
```

`X` is a vector of N examples, `Y` contains the desired outputs and the procedure "`fit one rule`" returns the values of β_m and γ_m that minimize the square error between $\beta s(x,\gamma)$ and the vector "`residual`," evaluated over the points given by `X`. Remember that we have identified the membership $\mu_{A_m}(x)$ with the function $s(x,\gamma_m)$, and the real number β_m with the product of the confidence w_m and the defuzzified value of the consequent B_m, both in rule number m.

The procedure "`fit one rule`" is an algorithm able to fit one single submodel to the set of data. This means that we just need to devise an efficient method for fitting one fuzzy rule to a dataset, without worrying about interferences between rules. Once a rule that models the residual is found, the set of data is replaced by the residuals of the whole model,

$$y \leftarrow y - \sum_{k\neq m} \beta_k s_k(x) \tag{21}$$

which in this case reduces to

$$y_m = y_{m-1} - \beta_{m-1} s_{m-1}(x) \tag{22}$$

and the process is repeated. A fuzzy rule is obtained in every iteration, and the process terminates once the best β_m is zero or the accuracy of the model is high enough, whatever come first.

It is remarked that there exist similarities between this process and other learning fuzzy systems. In particular, between backfitting and the selection stage in genetic iterative learning (GIL) [4]. In GIL, rules are selected one at a time and iteratively added to the rule base. In short, the process is as follows: the rule which best explains the data is found and added to the rule base. Then the examples covered by this rule are deleted from the training sample and the process repeated until the sample is empty. Backfitting is similar: besides examples are not removed, the value of the objective function in them is lowered, so that every new rule will focus in the points with high

error. It is easy to show that lowering the objective function in an example is equivalent to assign a real valued weight to it, and to find the best rule by weighted least squares. In that sense, these methods differ in one point: backfitting uses real valued weights, while GIL uses binary weights.

3.1. Fitting one rule

Fitting a submodel consists here in selecting one rule from the set of combined memberships of antecedent and consequent (i.e. the set of fuzzy memberships that can be used for a set $A_m \times B_m$, given the linguistic partitions of the features) and determining the value of w_m (the truth of the rule) that best fits it to the residuals.

If we know the membership of the antecedent, $s(x, \gamma_m) = A_m(x)$, the value of the product $\beta_m = w_m \cdot D(B_m)$ can be found analytically. Differentiating Eq. (20) and equating to 0, we obtain that the optimum value of β_m for a function $s(x, \gamma_m)$ is

$$\beta_m = \frac{E_X[Y \cdot (f_{m-1} + s(x, \gamma_m))]}{E_X^2[f_{m-1} + s(x, \gamma_m)]}. \tag{23}$$

Given the list of linguistic labels B_i in the output, one must find a pair of a real value $w_m \in [0, 1]$ and a linguistic term B_i such that $\beta_m = w_m \cdot D(B_m)$. Since the obtained value of β_m can be negative, it is needed that there exist linguistic labels for which $D(B_m) < 0$ for this algorithm to have a solution in the general case.

3.1.1. Determining the best antecedent

The preceding discussion can be used to assign a real value of merit to every antecedent A_m. This value is the square error obtained when this antecedent is combined with its optimal value β_m, given by Eq. (23). Therefore, the procedure "`fit one rule`" discussed here must include a search directed to find the combination of linguistic values that give that rule the best overall error for the current residuals.

A simple binary coded genetic algorithm is able to perform this discrete optimization part efficiently. Moreover, the GA structure can be exploited to integrate the feature selection process into the search scheme. We use a common coding scheme, based in [10]: a linguistic term is represented with a chain of bits. There are as many bits in the chain as different terms in the linguistic partition. If a term appears in the rule, its bit has the value '1', '0' otherwise.

For example, let {Low, Med, High} be the linguistic labels of all features in a problem involving three input variables. The rule

`If` x_1 `is High and` x_2 `is Med and` x_3 `is Low then` Y `is Med,`

is codified with the chain 001 010 100 010. It is immediate to extend this encoding: it can represent rules for which not all variables appear in the antecedent, and also 'OR' combinations of terms in the antecedent. For example, the rule

`If` x_1 `is High and` x_3 `is Low then` Y `is Med,`

is codified with the chain 001 000 100 010, and the rule

$$\begin{aligned}&\texttt{If } x_1 \texttt{ is (High or Med)}\\ &\quad\texttt{and } x_3 \texttt{ is Low}\\ &\texttt{then } Y \texttt{ is Med},\end{aligned}$$

will be assigned the chain 011 000 100 010.

Since we are aggregating rules with the sum, chains "001 111 100 010" and "001 000 100 010" are equivalent; the second representation will be preferred. As a general rule of thumb, "OR" combinations of rules increase the complexity of the knowledge base and we desire to minimize their number in the final result. To promote simpler individuals, it was decided that in case of tie when evaluating the squared error of two different individuals, the one with a lower number of bits is preferred; therefore, the search is guided towards rule banks that might not use all features.

4. Numerical results

4.1. Learning time

To assess the learning time of this method a synthetic problem was designed. It consists in generating 100 points $(x_1,\ldots,x_n)$ uniformly distributed in the interval $[0,1]^n$, with n varying from 1 to 15. Then, a linear model, a neural network, Wang and Mendel's method and Backfitting were applied to approximate the functions

$$s_n(x_1,\ldots,x_n)=\sum_{i=1}^{n} x_i. \tag{24}$$

Results are shown in Fig. 2. The left part shows that Wang–Mendel's method complexity is exponential in the number of features, while backfitting is approximately linear, provided that the genetic algorithm is stopped after a specified number of iterations (1000, for this experiment). The right part shows the final error of all models; noise was not added, therefore it should be 0 in all cases. In practice, since the number of samples is constant, the density of examples decreases with the number of features and conversely the error of non linear models is expected to increase. This effect is shown in the right part of the same figure.

4.2. Numerical accuracy

To study the accuracy of the backfitted fuzzy models we are going to compare the performance of the method proposed here with eight fuzzy learning methods and three black box models. Fuzzy algorithms considered are:

- Wang and Mendel [24], with selection of importance degrees by maximum (WM1), mean (WM2) and product maximum-mean (WM3) (the original definition of the method is in [24]; the two other ones are from [6]).
- Cordón and Herrera's method, also with maximum, mean and product maximum-mean (CH1, CH2, CH3) [6].

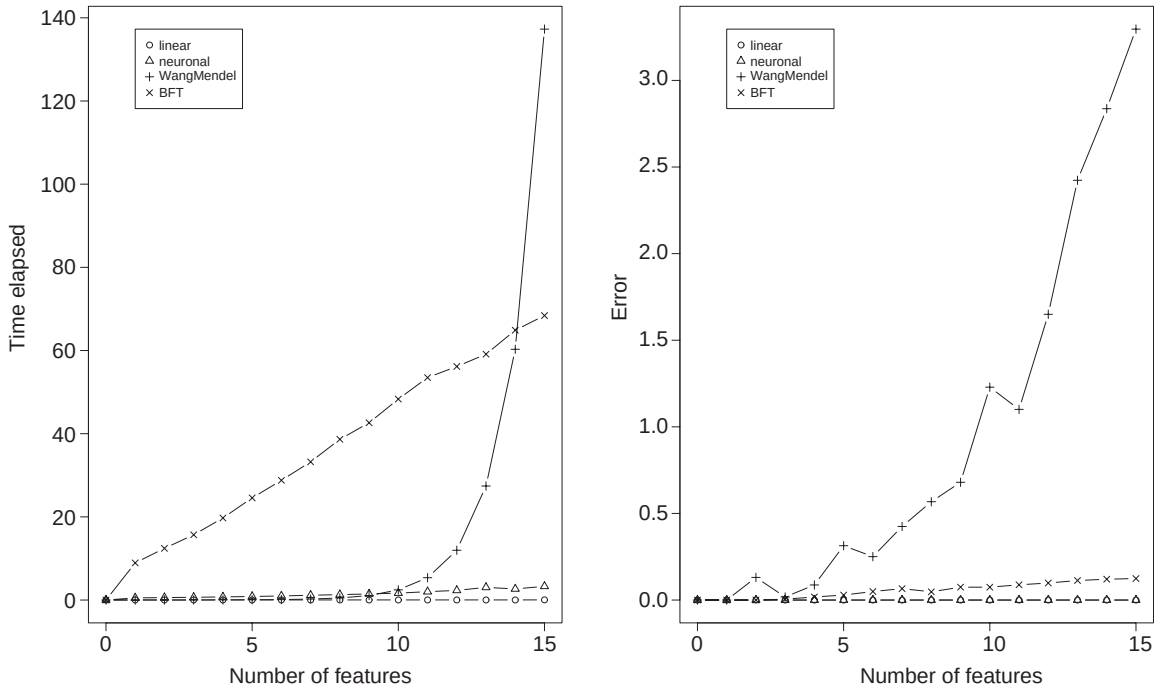

Fig. 2. Learning time in seconds (left) and train error (right) of linear regression, neural networks, Wang and Mendel and Backfitting algorithms on the problem discussed in Section 4.1. Neuronal and linear train error curves are indistinguishable.

- TSK rules, induced by locally weighted linear squares over all possible combinations of antecedents (WLS).
- Nozaki, Ishibuchi and Tanaka method (NIT) [15]

and black boxes are:

- Linear regression (LIN).
- Quadratic regression (QUA).
- Two layer perceptron (NN). These neural networks were trained with the conjugate gradient algorithm. The number of elements in the hidden layer was the same for all experiments over the same dataset. This number was designed to minimize a validation sample taken from a permutation of the sample that was not used to measure the test error.

Ten modeling problems were used in this comparison. Eight of them are synthetic, two are real world problems. These problems are:

- "f1": 676 examples of the function $z = x^2 + y^2$. "f1-10": The same function, with 10% Gaussian noise added. "f1-20": 20% Gaussian noise. "f1-50": 50% noise. The dataset is taken from [5].
- "f2": 676 examples of the function $10(x - xy)/(x - 2xy + y)$. "f2-10", "f2-20" and "f2-50" were added 10, 20 and 50% Gaussian noise [5].
- "Building". A real-world problem taken from [17].
- "Cable". Another real-world modeling problem, taken from [18].

Table 1
Comparative results between additive regression + backfitting and other approaches

	WM1	WM2	WM3	CH1	CH2	CH3	NIT	LIN	CUA	NEU	WLS	BFT
f1	5.65	5.73	5.57	5.82	8.90	6.93	5.63	130.5	**0.00**	0.17	0.09	**0.55**
f1-10	6.89	7.19	6.54	6.84	10.15	8.20	7.16	133.91	**1.40**	1.78	1.62	**2.17**
f1-20	11.07	10.99	11.06	11.33	13.45	12.42	10.63	135.6	**5.29**	6.42	5.90	**6.47**
f1-50	51.78	46.40	47.80	53.48	48.94	48.16	39.65	166.64	**33.53**	41.18	36.76	**38.94**
f2	0.41	0.48	0.45	0.40	0.59	0.45	0.43	1.54	1.61	1.48	**0.15**	**0.26**
f2-10	0.64	0.68	0.68	0.59	0.68	0.60	0.58	1.71	1.75	1.81	**0.29**	**0.42**
f2-20	1.27	1.16	1.17	1.29	1.15	1.17	0.97	2.04	2.09	0.90	**0.76**	**0.91**
f2-50	4.34	3.98	3.94	4.47	3.90	3.97	**3.59**	4.67	4.78	3.76	3.62	3.62
cable $\times 10^{-3}$	778	720	723	673	663	655	548	418	**393**	522	486	**441**
building $\times 10^{2}$	1.113	1.051	1.023	0.983	1.753	1.465	0.432	0.477	-	0.276	**0.246**	**0.299**

Size 3 partitions were used in all cases. BFT method was limited to 25 fuzzy rules. The best of WM, CH, NIT and BFT, plus the best overall model, were highlighted for every data set. f1 and f2 are synthetic data with Gaussian noise, cable and building are real world problems.

In all our experiments we have removed the mean from the datasets, so that $E(Y) = 0$ and there is a linguistic label Z for which $D(Z) = 0$, plus negative and positive values. Ruspini fuzzy partitions with three terms each were used for both input and output variables.

To measure the statistical significance of the differences between algorithms, 5x2cv experimental framework [8] was used: 50% of points were used to train the model, that was tested against the remaining 50%; roles of training and test sets are interchanged and the process done again. This is repeated 5 times, for different permutations of the data set, which gives 10 repetitions of the learning algorithm for every data set. The means of the 10 test errors obtained for every pair learning algorithm-data set are shown in Table 1. We decided to include the boxplot of the test error (see Fig. 3) instead of the p-values of the contrast, because this graph gives a deeper insight into the relative merits of the algorithms; we can consider that nonoverlapping boxes indicate that there exists a statistically significant difference between the algorithms involved.

Observe that backfitting was always among the best linguistic methods in both synthetic and real problems. There exist a significant difference between Wang–Mendel and Cordón–Herrera methods and the remaining ones, but these methods do not use weights in the rules neither non standard reasoning. NIT has a similar performance than these two first classes of methods in clean synthetic problems, and its relative performance improves under the presence of noise, as can be observed in the two upper rows in the figure. Since BFT was not statistically different to black boxes neither in synthetic nor in real problems, we can affirm that the properties of matching pursuit algorithms are also verified in the genetic fuzzy version we have proposed in this paper.

5. Concluding remarks and future work

Not all fuzzy reasoning methods are suitable for this implementation of matching pursuit: the t-norm must be the product, rules must be defuzzified before aggregated and the aggregation

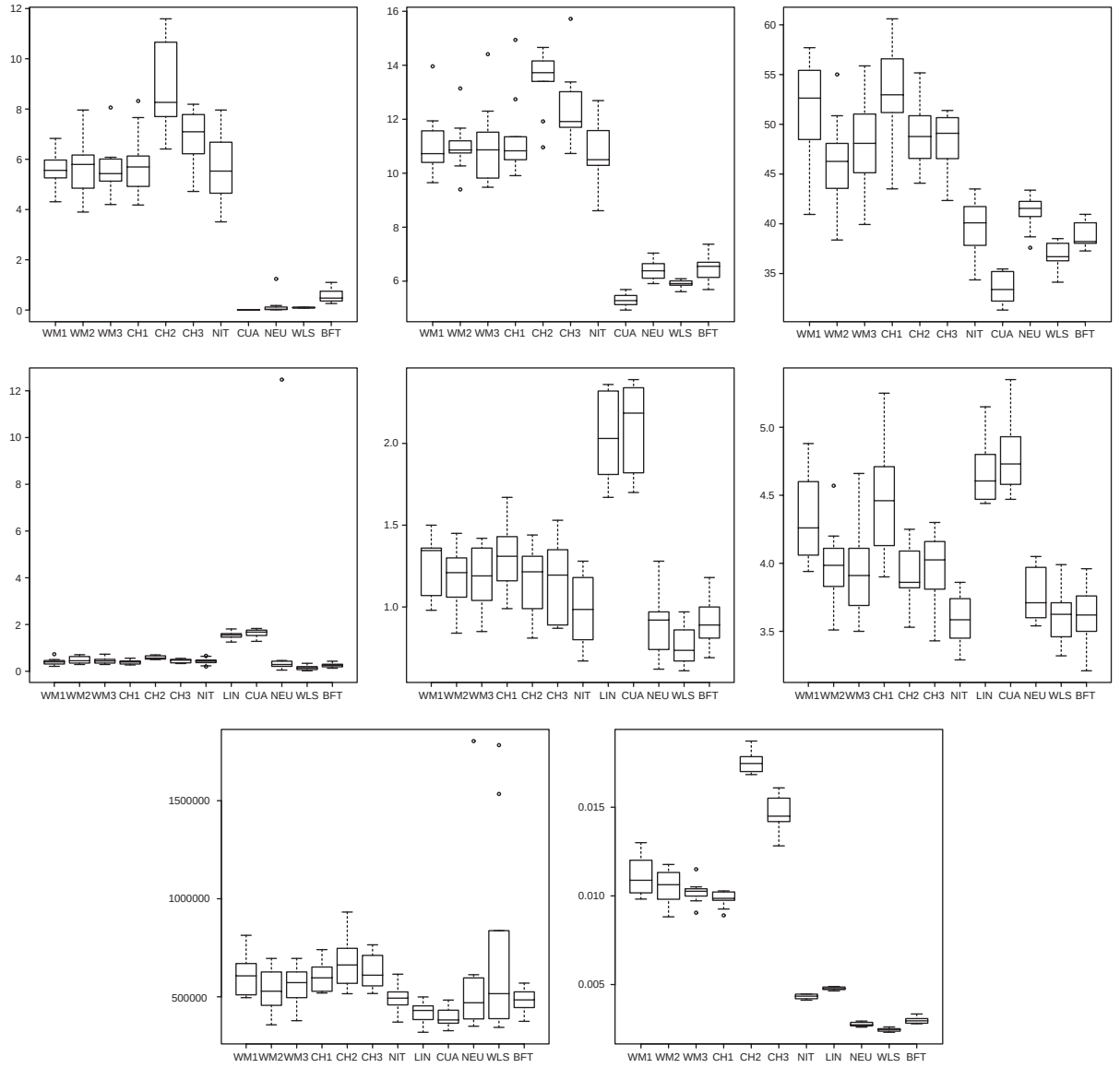

Fig. 3. Comparative results between additive regression + backfitting and other approaches. Size 3 partitions were used in all cases. BFT method was limited to 25 fuzzy rules. Upper part: f1, f1-20 and f1-50 data sets. Center: f2, f2-20 and f 2-50. Lower part: cable and building data sets.

operator must be the sum. These conditions restrict the applicability of this method, which cannot be considered as a general approach to fuzzy rules learning.

Provided that these conditions can be assumed, the use of matching pursuit and, in particular, the greedy backfitting algorithm, produces sparse rule bases in a very short time and achieves a very good performance without the need of tuning the membership functions of the antecedent. Up to

our knowledge, this is the computationally most efficient genetic fuzzy system up to date: it solves many practical problems in a few seconds, and it is faster than many "ad-hoc" methods while being comparable in precision to neural networks.

The work in this field of research is not finished. It was commented in Section 2.3 that the type of weighted rules we were inducting was a degenerate class of Quantified Fuzzy Sentences. It should be desirable to extend backfitting to cope with the general definition of these sentences, because this would replace the real valued weight by a linguistic quantifier, improving the descriptive properties of the result. Additionally, in Section 2.4, a relationship was introduced between matching pursuit and classification systems that could be studied in depth, to check whether the application of the LogitBoost to learn fuzzy classifiers algorithm improves the results already obtained with the application of AdaBoost to the same problem; our primary concern will be to reducing the number of rules even more, to setup rule bases comprising at most 10 or 20 rules. Finally, the relation that intuitively seems to exist between matching pursuit, support vector machines and the random set based classifiers based on maximum verosimility estimation [19] is not completely clear by now and we intend to study it in detail in a near future.

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: D. Haussler (Ed.), Proc. 5th Annual ACM Workshop on Computational Learning Theory, ACM Press, New York, 1992, pp. 144–152.

[2] J. Casillas, O. Cordón, F. Herrera, L. Magdalena, Finding a balance between interpretability and accuracy in fuzzy rule-based modeling: an overview, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Trade-Off Between Accuracy and Interpretability in Fuzzy Rule-Based Modeling, Physica-Verlag, 2002, to appear.

[3] J.S. Cho, D.J. Park, Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network, J. Intell. Fuzzy Systems 8 (2000) 99–110.

[4] O. Cordón, F. Herrera, A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples, Int. J. Approx. Reason. 17 (1) (1997) 369–407.

[5] O. Cordón, F. Herrera, A proposal for improving the accuracy of linguistic modeling, IEEE Trans. Fuzzy Syst. 8 (3) (2000) 335–344.

[6] O. Cordón, F. Herrera, A. Peregrin, Applicability of the fuzzy operators in the design of fuzzy logic controllers, Fuzzy Sets and Systems 86 (1997) 15–41.

[7] M. Delgado, D. Sánchez, J.M. Serrano, M.A. Vila, A survey of methods to evaluate quantified sentences, Math. Soft Comput. 7 (2000) 149–158.

[8] G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (7) (1998) 1895–1924.

[9] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Statist. 28 (2) (2000) 337–374.

[10] A. Gonzalez, R. Perez, Completeness and consistency conditions for learning fuzzy rules, Fuzzy Sets and Systems 96 (1996) 37–51.

[11] T.J. Hastie, R. Tibshirani, Generalized additive models, Statist. Sci. 1 (1986) 297–318.

[12] F. Hoffmann, Boosting a fenetic fuzzy classifier, Proc. IFSA 2001, Vancouver, Canada, 2001.

[13] L. Junco, L. Sanchez, Using the Adaboost algorithm to induce fuzzy rules in classification problems, Proc. ESTYLF 2000, Sevilla, 2001, pp. 297–301.

[14] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. 41 (1993) 3397–3415.

[15] K. Nozaki, H. Ishibuchi, H. Tanaka, A simple but powerful heuristic method for generating fuzzy rules from numerical data, Fuzzy Sets and Systems 86 (1997) 251–270.

[16] N.R. Pal, K. Pal, Handling of inconsistent rules with an extended model of fuzzy reasoning, J. Intelligent and Fuzzy Systems 7 (1998) 55–73.
[17] L. Prechelt, PROBEN1–A set of benchmarks and benchmarking rules for neural network training algorithms, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.
[18] L. Sánchez, Interval-valued GA-P algorithms, IEEE Trans. Evol. Comput. 4 (1) (2000) 64–72.
[19] L. Sánchez, J. Casillas, O. Cordón, M.J. del Jesus, Some relationships between fuzzy and random classifiers and models, Int. J. Approx. Reason. 29 (2002) 175–213.
[20] R.E. Schapire, The strength of weak learnability, Mach. Learning 5 (2) (1990) 197–227.
[21] C.J. Stone, Additive regression and other nonparametric models, Ann. Statist. 13 (1985) 689–705.
[22] N. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Trans. Fuzzy Systems 1 (1) (1993) 7–31.
[23] P. Vincent, Y. Bengio, Kernel matching pursuit, Mach. Learning 48 (2002) 165–187.
[24] L.X. Wang, J. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybernet. 25 (2) (1992) 353–361.
[25] W. Yu, Z. Bien, Design of fuzzy logic controller with inconsistent rule base, J. Intelligent Fuzzy Systems 2 (1994) 147–159.
[26] L.A. Zadeh, Fuzzy sets and information granularity, in: M.M. Gupta, R.K. Ragade, R.R. Yager (Eds.), Advances in Fuzzy Set Theory and Applications, North-Holland, New York, 1979, pp. 3–18.
[27] J. Zhu, T. Hastie, Kernel logistic regression and the import vector machine, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14, MIT Press, Cambridge, MA, 2002, pp. 1081–1088.