A fast genetic method for inducting linguistically understandable fuzzy models

Luciano Sánchez.
Universidad de Oviedo. Depto. Informática.
Sedes departamentales, despacho 1.1.28.
Campus de Viesques, 33203 Gijón.

Abstract

Fuzzy rule bases can be regarded as mixtures of experts, and boosting techniques can be applied to learn them from data. In particular, provided that adequate reasoning methods are used, fuzzy models are extended additive models, thus backfitting can be applied to them. We propose to use an implementation of backfitting that uses a genetic algorithm for fitting submodels to residuals and we also show that it is both more accurate and faster than other fuzzy rule learning methods.

1 Introduction

1.1 Fuzzy models and extended additive models

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

$$\texttt{If } X \texttt{ is } A_m \texttt{ then } Y \texttt{ is } B_m, \tag{1}$$

where X and Y are the feature and the output vectors, respectively, and A_m, B_m are conjunctions of linguistic labels, with in turn are associated to fuzzy sets. Let B' be the result of the inference process for the preceding rule; given an input x_0,

$$\mu_{B'}(y) = I(\mu_{A_m}(x_0), \mu_{B_m}(y)) \tag{2}$$

where I is a fuzzy inference operator. Let these sets be aggregated after defuzzified [3]. The output $F(x_0)$ of the fuzzy model is then computed as

$$F(x_0) = G_{m=1}^{M}(D(\mu_{B'_m}(y))) \tag{3}$$

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 us define I to be the product, D the centroid and G the sum. Then

$$F(x_0) = \sum_{m=1}^{M} \mu_{A_m}(x_0) \cdot D(\mu_{B_m}(y)) \tag{4}$$

and $D(\mu_{B_m}(y))$ is a constant, thus the output of the fuzzy model has the form

$$F(x) = \sum_{m=1}^{M} f_m(x) \tag{5}$$

where each $f_m(x)$ is a mapping $f_m(x) : \mathcal{X} \to R$. This is what we know as an extended additive model [2].

1.2 Backfitting algorithm

Formulating the regression problem for a model like this (i.e., modeling the mean $E(Y|x)$ from training data $(x_1, y_1), \ldots, (x_N, y_N)$, with x_i a vector valued feature and y_i scalar) can be quite simple if a back-fitting algorithm is used. Let the functions $f_m(x)$ be characterized by a set of parameters γ and a multiplier β_m,

$$f_m(x) = \beta_m b(x, \gamma_m). \tag{6}$$

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

$$E\left[y - \sum_{k \neq m} \beta_k b(x; \gamma_k) - \beta b(x; \gamma)\right]^2 \tag{7}$$

with respect to β, γ [2]. A greedy approach is frequently used to solve the problem:

$$\begin{array}{l} \{\beta_m, \gamma_m\} \leftarrow \\ \arg\min_{\beta,\gamma} E[y - F_{m-1}(x) - \beta b(x; \gamma)]^2 \end{array} \tag{8}$$

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

In this paper we propose to define $b(x, \gamma_m) := \mu_{A_m}(x)$ and $\beta_m := D(\mu_{B_m}(y))$ and to apply generalized backfitting to induce a fuzzy model. The following section explains how to adapt, with the help of a genetic algorithm, the greedy backfitting algorithm to fuzzy rule learning. Both learning time and numerical accuracy of the method will be experimentally contrasted to other fuzzy rule learning procedures in section 3.

0-7803-7078-3/01/$10.00 (C)2001 IEEE.

2 Proposed methodology

Generalized back-fitting requires an algorithm that 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. Then, the set of data will replaced by the residuals of the whole model,

$$y \leftarrow y - \sum_{k \neq m} f_k(x) \tag{9}$$

which is this case reduces to

$$y_m = y_{m-1} - f_{m-1}(x) \tag{10}$$

and the process is repeated. A fuzzy rule will be obtained in every iteration, and the process finishes when the best β_m is zero or the accuracy of the model is high enough, whichever come first. An outline of this process is shown in Figure 1.

```
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
```

Figure 1. Backfitting is a greedy method that is based on an algorithm that fits one rule to the residual of the incomplete rule base. A discrete coded genetic algorithm will be used to find the best fuzzy rule, whose degree of truth can be analytically determined.

2.1 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 β_m (the truth of the rule) that best fits it to the residuals.

We propose to find β_m analytically. Differentiating the square error and equating to 0, we obtain that the optimum value of β_m for a rule $A_m \rightarrow B_m$ is

$$\beta_m = \frac{E_X\, Y \cdot (F_{m-1} \cup [A_m \rightarrow B_m])}{E_X^2\, F_{m-1} \cup [A_m \rightarrow B_m]} \tag{11}$$

where $[A_m \rightarrow B_m](x)$ is the defuzzified output of the rule and the union symbol means "add one rule to the base". Since this value is not restricted to $[0, 1]$, values of β_m greater than 1 are replaced by repetitions of the same rule and negative importances have their sign changed and their linguistic label swapped with other one, centered in a negative point.

$[A_m \rightarrow B_m]$ is assigned the square error obtained when the optimal value of β_m, given by eq. 11. It only remains 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 can do this discrete optimization part, as we will show in the next subsection.

2.2 GA search of linguistic terms

The GA structure can be exploited to integrate the feature selection process into the search scheme. We will use a coding scheme based in [5]. Let us codify a linguistic term with a '1' bit in a chain of so many bits as different terms in the linguistic partition. 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. We extend this encoding to represent rules for which not all variables appear in the antecedent and '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

If x_1 is(High or Med)
and x_3 is Low
then Y is Med,

will be assigned the chain 011 000 100 010.

Observe that, under the fuzzy reasoning method used here, chains "001 111 100 010" and "001 000 100 010" are equivalent. "OR" combinations of rules increase the complexity of the knowledge base and we desire to minimize their number in the final result. Therefore, 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. This way, the search is guided toward rule banks that might not use all features.

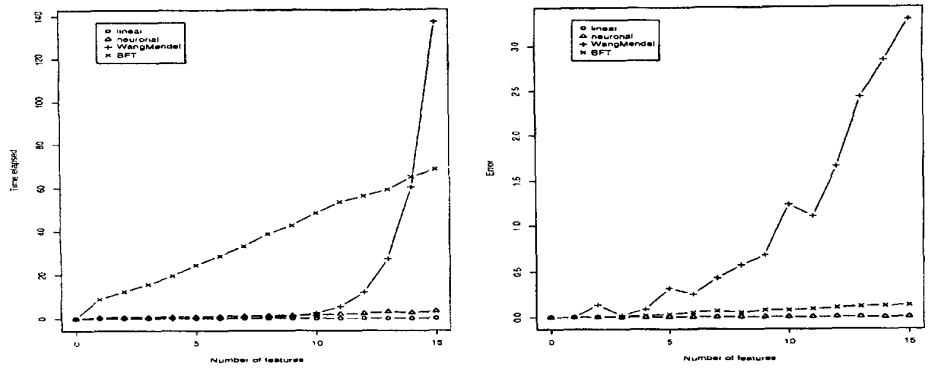

Figure 2. Learning time in seconds (left) and train error (right) of linear regression, neural networks, Wang and Mendel and Backfitting algorithms on the problem discussed in section 3.1. Neuronal and linear train error curves are the same.

3 Numerical results

3.1 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

$$s_n(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i. \tag{12}$$

Results are shown in Figure 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.

3.2 Numerical accuracy

Wang and Mendel with importance degrees 'maximum' (WM1), 'mean' (WM2) and 'product maximum-mean' (WM3) [6], the same three versions of Cordón and Herrera's method (CH1, CH2, CH3) [4], NIT [8], Linear (LIN) and Quadratic regression (QUA) , Neural Networks (NN) and TSK rules induced with Weighted Least Squares (WLS) are compared to this method (BFT) over 8 synthetic problems and two real world problems. "f1" is $z = x^2 + y^2$ and "f2" is $10(x - xy)/(x - 2xy + y)$ (both were taken from [4]). "fx-y" is the function fx with y% of gaussian noise. "Building" was taken from [9] and "Cable" from [7]. 5x2cv experimental framework [10] was used: 50% of points were used to train the model, that was tested against the remaining 50%; roles of training and test sets were interchanged and the process repeated, and this was replicated 5 times for different permutations of the dataset, which gives 10 repetitions of the learning algorithm for every dataset. The mean of the test errors is shown in Table 1 and the boxplot of the results are shown in figure 3. p-values assessing significance of the statistical contrasts as indicated in 5x2cv method are not included, but they can be deduced from the graphs: non overlapping boxes indicate that there exist a statistically significant difference between the algorithms involved. Observe that backfitting was more accurate in both synthetic and real problems.

4 Concluding Remarks

Not all fuzzy reasoning methods are suitable for backfitting: the t-norm must be the product, rules must be defuzzified before aggregated and the aggregation operator must be the sum. This limits the application of this method as a general approach to fuzzy rules learning. But, when this limitations are accepted, backfitting is very precise and faster than other genetic fuzzy systems; it is remarkable than this genetic fuzzy system can also be faster than many "ad-hoc" methods while being comparable in precision and time to neural networks.

	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 $\cdot 10^{-3}$	778	720	723	673	663	655	548	418	**393**	522	486	**441**
building $\cdot 10^{2}$	1.113	1.051	1.023	0.983	1.753	1.465	0.432	0.477	–	0.276	**0.246**	**0.299**

Table 1. 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. The best of WM, CH, NIT and BFT, plus the best overall model, were highlighted for every dataset. f1 and f2 are synthetic data with gaussian noise, cable and building are real world problems.

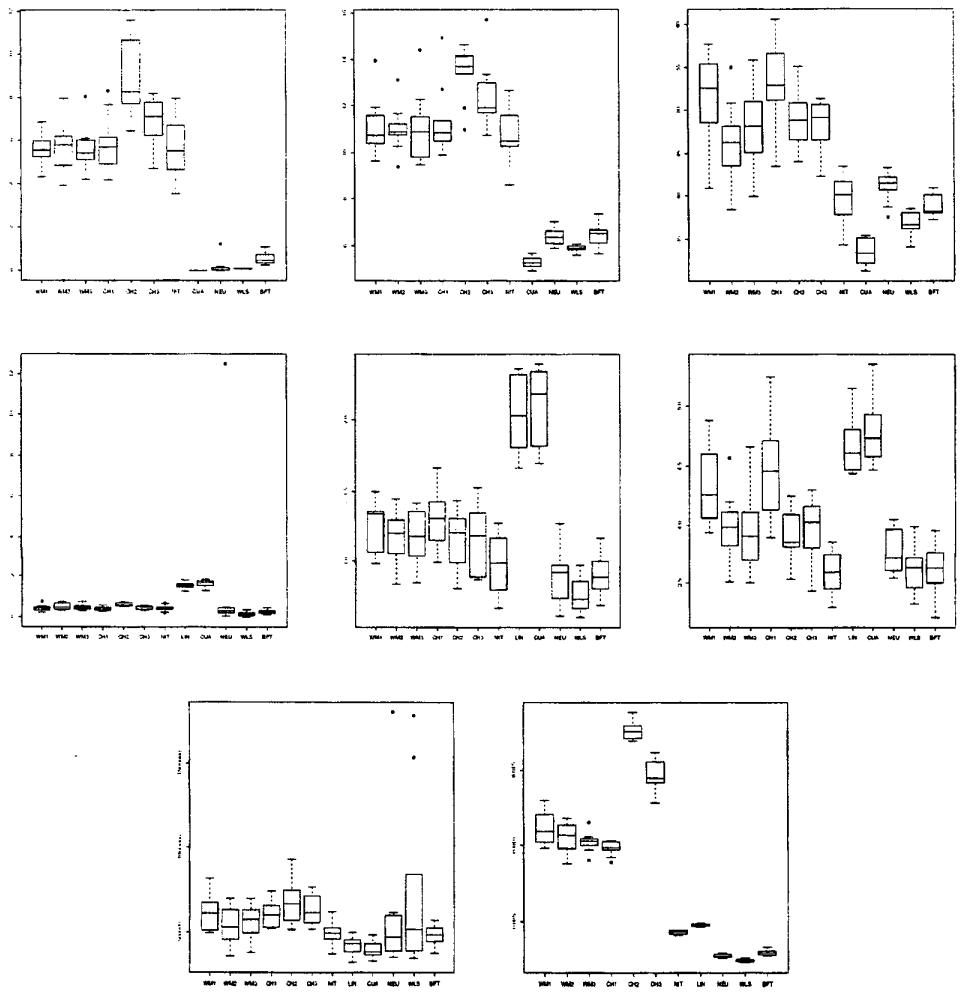

Figure 3. Comparative results between additive regression + backfitting and other approaches. Size 3 partitions were used in all cases. BFT method was limited to 25 fuzzy rules. Upper part: f1, f1-20 and f1-50 datasets. Center: f2, f2-20 and f2-50. Lower part: cable and building datasets.

References

[1] Mallat, S. Zhang, Z. (1993) "Matching pursuits with time-frequency dictionaries". IEEE Trans on Signal Processing 41, pp 3397-3415.

[2] Friedman, J., Hastie, T., Tibshirani, R. (1998) "Additive Logistic Regression: A Statistical View of Boosting". Machine Learning.

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

[4] Cordón, O., Herrera, F. (2000) "A proposal for improving the accuracy of linguistic modeling". IEEE Transactions on Fuzzy Systems, 8, 3, pp. 335-344.

[5] Gonzalez, A., Perez, R. (1996) "Completeness and consistency conditions for learning fuzzy rules," Fuzzy Sets and Systems, vol 96, pp 37-51.

[6] Wang, L. X., Mendel, J. (1992) "Generating fuzzy rules by learning from examples". IEEE Trans. on Systems, Man and Cybernetics, 25, 2, pp. 353-361.

[7] Sánchez, L. (2000) "Interval-Valued GA-P Algorithms". IEEE Transactions on Evolutionary Computation, 4, 1, pp 64-72.

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

[9] Prechelt, Lutz. (1994) "PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms". Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe.

[10] Dietterich, G. (1998) "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". Neural Computation, 10, 7, pp 1895-1924.