Building an interpretable fuzzy rule base from data using Orthogonal Least Squares Application to a depollution problem

Sébastien Destercke, Serge Guillaume, Brigitte Charnomordic

Building an interpretable fuzzy rule base from data using Orthogonal Least Squares Application to a depollution problem

Sébastien Destercke [b], Serge Guillaume [a],
Brigitte Charnomordic [b,*],

[a] *Umr Itap, Cemagref, BP 5095, 34196 Montpellier Cedex, France*
[b] *Umr ASB, INRA, 2 place Viala, 34060 Montpellier Cedex, France*

Abstract

In many fields where human understanding plays a crucial role, such as bioprocesses, the capacity of extracting knowledge from data is of critical importance. Within this framework, fuzzy learning methods, if properly used, can greatly help human experts. Amongst these methods, the aim of orthogonal transformations, which have been proven to be mathematically robust, is to build rules from a set of training data and to select the most important ones by linear regression or rank revealing techniques. The OLS algorithm is a good representative of those methods. However, it was originally designed so that it only cared about numerical performance. Thus, we propose some modifications of the original method to take interpretability into account. After recalling the original algorithm, this paper presents the changes made to the original method, then discusses some results obtained from benchmark problems. Finally, the algorithm is applied to a real-world fault detection depollution problem.

Key words: Learning, rule induction, fuzzy logic, interpretability, OLS, orthogonal transformations, depollution, fault detection

1 Introduction

Fuzzy learning methods, unlike "black-box" models such as neural networks, are likely to give interpretable results, provided that some constraints are respected.

* Corresponding author.
Email address: bch@ensam.inra.fr (Brigitte Charnomordic).

While this ability is somewhat meaningless in some applications such as stock market prediction, it becomes essential when human experts want to gain insight into a complex problem (e.g. industrial [17] and biological [28] processes, climate evolution [13]).

These considerations explain why interpretability issues in Fuzzy Modeling have become an important research topic, as shown in recent literature [2]. Even so, the meaning given to interpretability in Fuzzy Modeling is not always the same. By interpretability, some authors mean mathematical interpretability, as in [1] where a structure is developed in Takagi-Sugeno systems, that leads to the interpretation of every consequent polynomial as a Taylor series expansion about the rule center. Others mean linguistic interpretability, as in [11], [10]. The present paper is focused on the latter approach. Commonly admitted requirements for interpretability are a small number of consistent membership functions and a reasonable number of rules in the fuzzy system.

Orthogonal transformation methods provide a set of tools for building rules from data and selecting a limited subset of rules. Those methods were originally designed for linear optimization, but subject to some conditions they can be used in fuzzy models. For instance, a zero order Takagi Sugeno model can be written as a set of r fuzzy rules, the qth rule being:

$$R_q: \text{ if } x_1 \text{ is } A_1^q \text{ and } x_2 \text{ is } A_2^q \text{ and } \ldots \text{ then } y = \theta^q \tag{1}$$

where $A_1^q, A_2^q \ldots$ are the fuzzy sets associated to the $x_1, x_2, \ldots$ variables for that given rule, and θ^q is the corresponding crisp rule conclusion.

Let (x, y) be N input-output pairs of a data set, where $x \in \mathbb{R}^p$ and $y \in \mathbb{R}$. For the ith pair, the above Takagi Sugeno model output is calculated as follows:

$$\widehat{y^i} = \frac{\sum_{q=1}^{r} \theta^q \left(\bigwedge_{j=1}^{p} \mu_{A_j^q}(x_j^i) \right)}{\sum_{q=1}^{r} \left(\bigwedge_{j=1}^{p} \mu_{A_j^q}(x_j^i) \right)} \tag{2}$$

In equation 2, $\bigwedge$ is the conjunction operator used to combine elements in the rule premise, $\mu_{A_j^q}(x_j^i)$ represents, within the qth rule, the membership function value for $x_j^i, j = 1 \ldots p$.

Let us introduce the rule firing strength $w^q(x^i) = \bigwedge_{j=1}^{p} \mu_{A_j^q}(x_j^i)$. Thus equation 2 can

be rewritten as:

$$\widehat{y^i} = \frac{\sum_{q=1}^{r} \theta^q w^q(x^i)}{\sum_{q=1}^{r} w^q(x^i)} \tag{3}$$

Once the fuzzy partitions have been set, and provided a given data set, the $w^q(x^i)$ can be computed for all x^i in the data set. Then equation 3 allows to reformulate the fuzzy model as a linear regression problem, written in matrix form as: $y = P\theta + E$. In that matrix form, y is the sample output vector, P is the firing strength matrix, θ is the rule consequent vector and E is an error term. Orthogonal transformation methods can then be used to determine the θ^q to be kept, and to assign them optimal values in order to design a zero order Takagi Sugeno model from the data set.

A thorough review of the use of orthogonal transformation methods (SVD, QR, OLS) to select fuzzy rules can be found in [29]. They can be divided into two main families: the methods that select rules using the P matrix decomposition only, and others that also use the output y to do a best fit. The first family of methods (rank revealing techniques) is particularly interesting when the input fuzzy partitions include redundant or quasi redundant fuzzy sets. The orthogonal least squares (OLS) technique belongs to the second family and allows a rule selection based on the rule respective contribution to the output inertia or variance. With respect to this criterion, it gives a good summary of the system to be modeled, which explains why it has been widely used in Statistics, and also why it is particularly suited for rule induction, as shown for instance in [26].

The aim of the present paper is to establish, by using the OLS method as an example, that orthogonal transformation results can be made interpretable, without suffering too much loss of accuracy. This is achieved by building interpretable fuzzy partitions and by reducing the number of rule conclusions. This turns orthogonal transformations into useful tools for modeling regression problems and extracting knowledge from data. Thus they are worth a careful study as there are few available techniques for achieving this double objective, contrary to knowledge induction in classification problems.

In section 2, we recall how the original OLS works. Section 3 introduces the learning criteria that will be used in our modified OLS algorithm. Section 4 presents the modifications necessary to respect the interpretability constraints. In the next section, the modified algorithm is applied to benchmark problems, compared to the original one and to reference results found in the literature. A real-world application is presented and analyzed in section 6. Finally we give some conclusions and perspectives for future work.

2 Original OLS algorithm

The OLS (orthogonal least squares) algorithm [3,4] can be used in Fuzzy Modeling to make a rule selection using the same technique as in linear regression. Wang and Mendel [27] introduced the use of Fuzzy Basis Functions to map the input variables into a new linear space. We will recall the main steps used in the original algorithm.

Rule construction

First N rules are built, one from each pair in the data set. Hohensohn and Mendel [15] proposed the following Gaussian membership function for the jth dimension of the ith rule.

$$\mu_{A_j^i}(u) = e^{\left[-\frac{1}{2}\left(\frac{(u-x_j^i)}{\sigma_j}\right)^2\right]} \tag{4}$$

with $\sigma_j = s.[\max_{i=1,2,...,N}(x_j^i) - \min_{i=1,2,...,N}(x_j^i)]$, s being a scale factor whose value depends on the problem.

Rule selection

Once the membership functions have been built, the Fuzzy Inference System (FIS) optimization is done in two steps. The first step is non-linear and consists in fuzzy basis function (FBF) construction; the second step, which is linear, is the orthogonal least square application to the FBF.

A FBF $p^i(x^i)$ is the relative contribution of the ith rule, built from the ith example, to the inferred output:

$$p^i(x^i) = \frac{w^i(x^i)}{\sum_{q=1}^{N} w^q(x^i)}$$

Thus the fuzzy system output (see equation 3) can be written and viewed as a linear combination: $\widehat{y^i} = \sum_q p^q(x^i)\,\theta^q$, where $\theta^q \in \mathbb{R}$ are the parameters to optimize (they correspond to the rule conclusions). The system is equivalent to the matrix form $y = P\theta + E$, y being the observed output while E is the error term, supposed to be uncorrelated with the $p^i(x)$ or P.

The element p_{ji} of the matrix P represents the ith rule firing strength for the jth pair, i.e. the jth component of the p^i vector.

The OLS procedure transforms the p^i regressors into a set of orthogonal ones using the Gram-Schmidt procedure. The P matrix can be decomposed into an orthogonal one, M, and an upper triangle one, A.
The system becomes $y = MA\theta + E$. Let $g = A\theta$, then the orthogonal least square solution of the system is $\widehat{g}_i = \dfrac{m_i^T y}{m_i^T m_i}$, $1 \leq i \leq r$ where m_i is the ith column of the orthogonal matrix M.

Optimal $\widehat{\theta}$ can be computed using the triangular system $A\widehat{\theta} = \widehat{g}$.

Thanks to the orthogonal characteristic of M, there is no covariance, hence vector (i.e. rule) individual contributions are additive. This property is used to select the rules. At each step, the algorithm selects the vector m_i that maximizes the explained variance of the observed output y. The selection criterion is the following one:

$$[xVar]_i = \frac{g_i^2 \; m_i^T m_i}{y^T y}$$

The selection stops when the cumulated explained variance is satisfactory. This occurs at step $r \leq N$ when

$$1 - \sum_{i=1}^{r} [xVar]_i < \epsilon \tag{5}$$

ϵ being a threshold value (e.g. 0.01).

Conclusion optimization

As the selected m_i still contain some information related to unselected rules, Hohensohn and Mendel [15] propose to run the algorithm a second time. No selection is made during this second pass, the aim being only to optimize the rule conclusions.

The original algorithm, as described here, results in models with a good numerical accuracy. However, as we'll see later on, it has many drawbacks when the objective is not only numerical accuracy but also knowledge extraction.

3 Learning criteria

Two numerical criteria are presented: the coverage index, based upon an activation threshold, and the performance index. We will use them to assess the overall system

quality. The performance index is an error based index that will allow us to measure the numerical accuracy of our results, while the coverage index, together with the activation threshold, will give us information related to the system completeness with respect to the learning data. To some extent, the coverage index reflects the potential quality of the extracted knowledge. Linguistic integrity, for its part, is insured by the proposed method, and thus does not need to be evaluated.

The two criteria are actually independent of the OLS algorithm and can be used to assess the quality of any Fuzzy Inference System (FIS).

3.1 Coverage index and activation threshold

Consider a rule base RB_r containing r rules such as the one given in equation 1.

Definitions

Let I_i be the interval corresponding to the *ith* input range and $I^p \subseteq I_1 \times \ldots \times I_p$ be the subset of $\mathbb{R}^p$ covered by the rule base ($I_1 \times \ldots \times I_p$ is the Cartesian product).

Definition 1 *An activation threshold $\alpha \in [0, 1]$ defines the following constraint: given α, a sample x^i is said active iff* $\exists\ R_q \in RB_r$ *s.t.* $w^q(x^i) > \alpha$.

Definition 2 *Let n be the number of active samples. The coverage index $CI_\alpha = n/N$ is the proportion of active samples for the activation threshold α.*

Note: Increasing the activation threshold reduces the amount of active samples and transforms I^p into a subset $I^p_\alpha \subseteq I^p$

The threshold choice depends on the conjunctive operator used to compute the rule firing strength: the use of a **prod** operator yields lesser firing strengths which will decrease with the input dimension, while a **min** operator results in higher and less dependent firing strengths.

The two-dimensional rule system depicted by figure 1 illustrates the usefulness of the activation threshold and coverage index in the framework of knowledge extraction.

3.1.1 Maximum coverage index ($\alpha = 0$)

CI_0 is the maximum coverage index and it gives us two kinds of information:

- Completeness: for a so called complete system, where each data set item activates at least one rule, we have $CI_0 = 1$ while an empty rule base yields

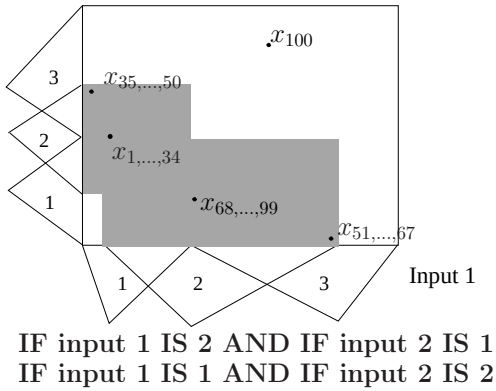


Fig. 1. Input domain rule coverage

$CI_0 = 0$. The coverage index can thus be used to measure the completeness of the rule base, with respect to a given data set.

- Exception data: $CI_\alpha \approx 1$ is often the consequence of exception samples. We call exception an isolated sample which is not covered by the rule base. Sample x_{100} in figure 1 is such an exception.

The maximum coverage index of the system shown in figure 1 is 99%, meaning that there are a few exceptions.

3.1.2 Coverage index for $\alpha > 0$

Figure 2 shows the previous system behaviour with an activation threshold $\alpha = 0.1$. Unfortunately, the coverage index drastically drops from 99% to 66%.

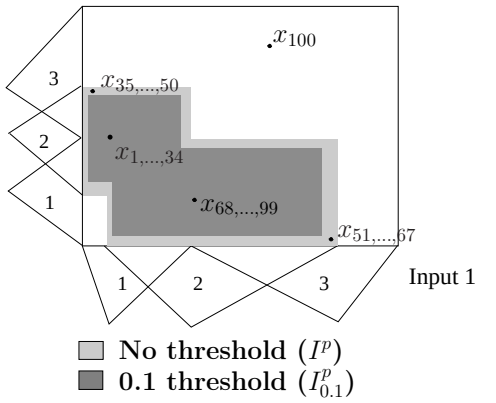


Fig. 2. Input domain with $\alpha = 0.1$

Generally speaking, the use of a coverage index gives indications as to:

- System robustness (see Figure 2).
- Reliability of extracted knowledge: Figure 1 shows that a system can have a good accuracy and be unreliable.

- Rule side effect: if too many samples are found in the rule borders, the rule base reliability is questionable. Studying the evolution of coverage versus the activation threshold allows to quantify this "rule side effect".

As we shall see in section 5, a blind application of OLS may induce fairly pathological situations (good accuracy and a perfect coverage index dropping down as soon as an activation threshold constraint is added).

The coverage index and activation threshold are easy-to-use, easy-to-understand tools with the ability to detect such undesirable rule bases.

3.2 Performance index

The performance index reflects the numerical accuracy of the predictive system. In this study, we use $PI = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \left\| \widehat{y^i} - y^i \right\|^2}$.

The performance index only takes account of the active samples ($n \leq N$, see definition 2), so a given system may have good prediction results on only a few of the available samples (i.e. good PI but poor $CI\alpha$), or cover the whole data set, but with a lower accuracy.

4 Proposed modifications for the OLS

In this section, we propose changes that aim to improve induced rule interpretability. Rule premises, through variable partitioning, and rule conclusions are both subject to modification.

Figure 3 is a flowchart describing the method used in the original OLS and the proposed modifications.

The fuzzy partitioning readability is a prerequisite to build an interpretable rule base [11]. In the original OLS algorithm, a rule is built from each item of the training set, and a Gaussian membership function is generated from each value of each variable. Thus a given fuzzy partition is made up of as many fuzzy sets as there are distinct values within the data distribution. The result, illustrated in figure 4, is not interpretable. Some membership functions are quasi redundant, and many of the corresponding fuzzy sets are not distinguishable, which makes it impossible to give them a semantic meaning. Moreover, Gaussian functions have another drawback for our purpose: their unlimited boundaries, which yield a perfect coverage index, likely to drop down as soon as an activation threshold is set.

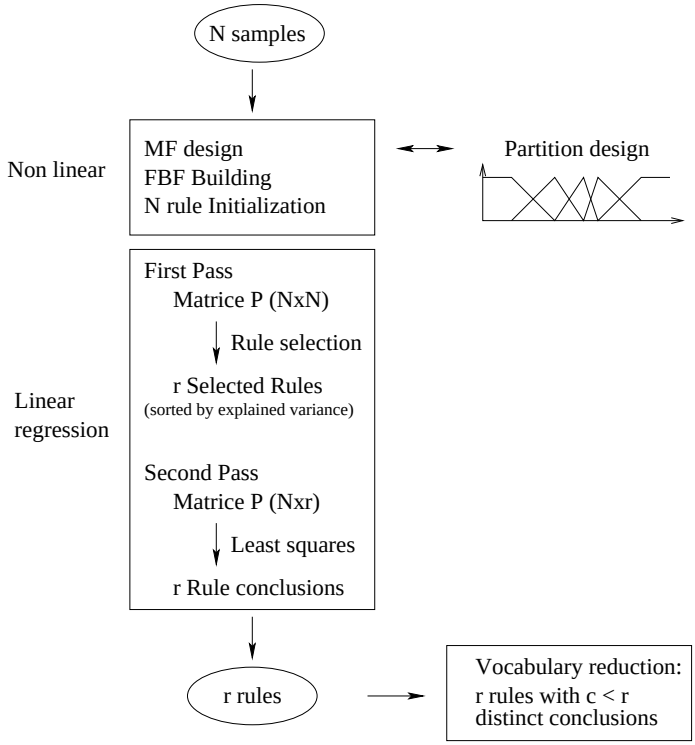


Fig. 3. Flowchart for the modified OLS algorithm

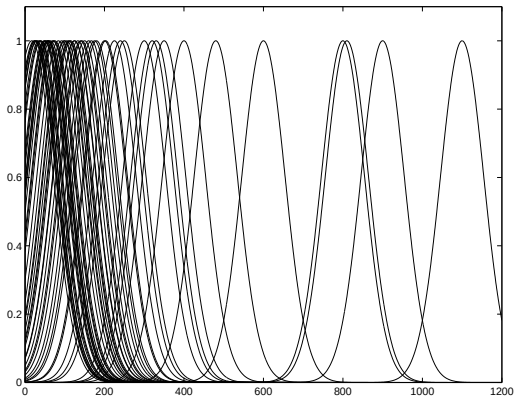


Fig. 4. Original fuzzy partition generated from a 106-item sample

4.1 Fuzzy partition readability

The necessary conditions for a fuzzy partition to be interpretable by a human expert have been studied by several authors [5, 8, 9]. Let us recall the main points:

- Distinguishability: Semantic integrity requires that the membership functions represent linguistic concepts different from each other.
- A justifiable number of fuzzy sets [19].
- Normalization: All the fuzzy sets should be normal.
- Overlapping: All the fuzzy sets should significantly overlap.

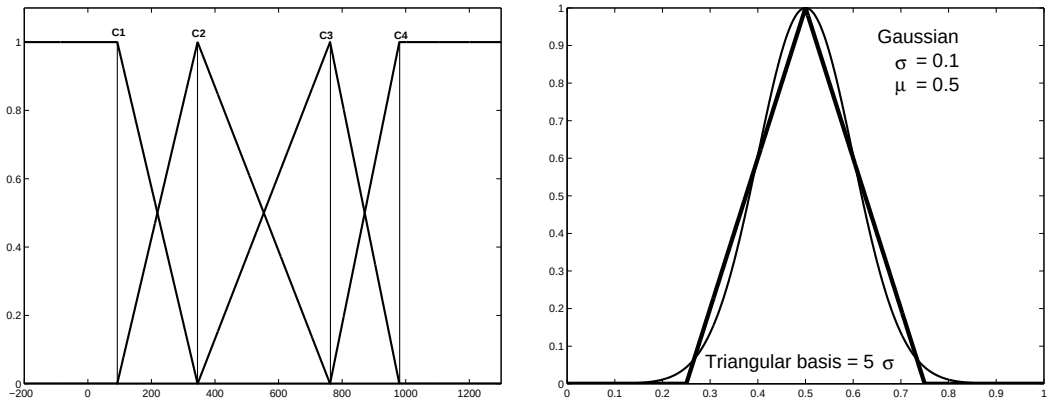
- Domain coverage: Each data point, x, should belong significantly, $\mu(x) > \epsilon$, at least to one fuzzy set. ϵ is called the coverage level [21].

We implement these constraints within a standardized fuzzy partition as proposed in [24]:

$$\begin{cases} \forall x \sum\limits_{f=1,2,..,M} \mu_f(x) = 1 \\ \forall f \; \exists \, x \; such \; as \; \mu_f(x) = 1 \end{cases} \tag{6}$$

where M is the number of fuzzy sets in the partition and $\mu_f(x)$ is the membership degree of x to the fth fuzzy set. Equation 6 means that any point belongs at most to two fuzzy sets when the fuzzy sets are convex.

Due to their specific properties [22] we choose fuzzy sets of triangular shape, except at the domain edges, where they are semi trapezoidal, as shown in figure 5.a. Such a M-term standardized fuzzy partition is completely defined by M points, the fuzzy set centers. With an appropriate choice of parameters, symmetrical triangle MFs approximately cover the same range as their Gaussian equivalent (see figure 5.b).



(a) A standardized fuzzy partition (b) Triangle equivalent to a Gaussian MF

Fig. 5. New fuzzy partitions

4.2 Fuzzy partition design

Various methods are available to build fuzzy partitions [18]. In this paper, we want to use the OLS algorithm to build interpretable rule bases, while preserving a good numerical accuracy. To be sure that this is the case, we have to compare results obtained with the original partitioning design in OLS and those achieved with an interpretable partitioning. To that effect, we need a simple and efficient way to design standardized fuzzy partitions from data, as the one given below. The fuzzy set centers are not equidistant as in a regular grid, but are estimated according to the data distribution, using the well known $k - means$ algorithm [14]. The multidimensional k-means is recalled in Algorithm 1, we use it here independently in each

input dimension.

1: Let N multidimensional data points denoted $x_i, i = 1 \dots N$
Let C the number of clusters to build
2: Initialization: choose k centroids E(k), $k = 1 \dots C$
(random or uniformly spaced)
3: Assign each data point to the nearest cluster:
$cluster(x_i) = argmin_k(dist(x_i, E(k))$
4: Compute cluster centroids m(k)
5: **while** $\exists k$ such as $m(k) \neq E(k)$ **do**
6: FOR (k=1 to C) $E(k) = m(k)$
7: GOTO 3
8: **end while**

Algorithm 1. k-means algorithm

How to choose the number of fuzzy sets for each input variable? There are several criteria to assess partition quality [11] but it is difficult to make an a priori choice. In order to choose the appropriate partition size, we first generate a hierarchy of partitions of increasing size in each input dimension j, denoted $FP_j^{n_j}$ for a n_j size, n_j^{max} being the maximum size of the partition (limited to a reasonable number (≈ 7) [19]).

Note: $FP_j^{n_j}$ is uniquely determined by its size n_j, the fuzzy set centers being the coordinates computed by the $k - means$ algorithm.

The best suited number of terms for each input variable is determined using a refinement procedure based on the use of the hierarchy of fuzzy partitions. This iterative algorithm is presented below. It calls a FIS generation algorithm to be described later. It is not a greedy algorithm, unlike other techniques. It does not implement all possible combinations of the fuzzy sets, but only a few chosen ones.

Table 1 illustrates the first steps of a refinement procedure for a four inpur system. Detailed procedures are given in Algorithm 2(refinement procedure) and Algorithm 3 (FIS generation).

The key idea is to introduce as many variables, described by a sufficient number of fuzzy sets, as necessary to get a good rule base.

The initial FIS is the simplest one possible, having only one rule (Algorithm 2, lines 1-2; Table 1, line 1). The search loop (algorithm lines 5 to 12) builds up temporary fuzzy inference systems. Each of them corresponds to adding to the initial FIS one fuzzy set in a given dimension. The selection of the dimension to retain is based upon performance and is done in lines 14-15 of the algorithm. If we go back to table 1, we see that the second iteration corresponds to lines 2 to 5, and that the best configuration is found by refining input variable # 2. Following this selection, a FIS

Line #	Iteration #	#MF per variable	PI	CI
1	1	1 1 1 1	PI_1	CI_1
2	2	2 1 1 1	PI_2^1	CI_2^1
3	2	1 2 1 1	PI_2^2	CI_2^2(best)
4	2	1 1 2 1	PI_2^3	CI_2^3
5	2	1 1 1 2	PI_2^4	CI_2^4
6	3	2 2 1 1	PI_3^1	CI_3^1
7	3	1 3 1 1	PI_3^2	CI_3^2
8	3	1 2 2 1	PI_3^3	CI_3^3 (best)
9	3	1 2 1 2	PI_3^4	CI_3^4
10	4	2 2 2 1	PI_4^1	CI_4^1
11	4	...		...

Table 1
An example of ongoing refinement procedure

1: iter = 1; $\forall j\ n_j = 1$
2: CALL *FIS Generation* (Algorithm 3)
3: **while** $iter \leq iter_{max}$ **do**
4: Store system as base system
5: **for** $1 \leq j \leq p$ **do**
6: **if** $n_j = n_j^{max}$ **then** next j (partition size limit reached for input j)
7: $n_j = n_j + 1$
8: CALL *FIS Generation* (Algorithm 3)
9: $PI_j = PI$
10: $n_j = n_j - 1$
11: Restore base system
12: **end for**
13: **if** $\forall j\ n_j = n_j^{max}$ **then** exit (no more inputs to refine)
14: s = argmin $\{PI_j,\ j = 1, \ldots, p, \quad n_j < n_j^{max}\}$ (Select input to refine)
15: $n_s = n_s + 1$
16: CALL *FIS Generation* (Algorithm 3): return FIS_{iter}
17: iter = iter + 1
18: **end while**

Algorithm 2. Refinement procedure

to be kept is built up. It will serve as a base to reiterate the sequence (Algorithm 2, lines 3 to 18).

When necessary, the procedure calls a FIS generation algorithm, referred to as Algorithm 3, which is now detailed.

The rule generation is done by combining the fuzzy sets of the $FP_j^{n_j}$ partitions for $j = 1, \ldots, p$, as described by Algorithm 3. The algorithm then removes the less influential rules and evaluates the rule conclusions, using output training data values $y^i, i = 1 \ldots N$. The condition stated in line 5, where α, the activation threshold

Require: $\{n_j \mid j = 1, ..., p\}$
1: get $FP_j^{n_j} \quad \forall j = 1, ..., p$
2: Generate the $\prod_{j=1}^{p} n_j$ rule premises
3: **for all** Rule $r \in FIS$ **do**
4: $\quad \alpha_r = \max_k w^r(x^k)$
5: $\quad$ **if** $\alpha_r < \alpha$ **then** remove rule r
6: $\quad$ **else** initialize rule conclusion $C_r = \frac{1}{N} \sum_{i=1}^{N} w^r y^i$
7: **end for**
8: Compute PI

Algorithm 3. FIS generation

defined in section 3.1, ensures that the rule is significantly fired by the examples of the training set.

The procedure does not yield a single fuzzy inference system, but K FIS of increasing complexity. The selection of the best one takes into consideration both performance and coverage indices. The selected FIS_k corresponds to:

$k = argmin(PI_k, k = 1, \ldots, K \; such \; as \; CI_\alpha(FIS_k) \geq thres)$, where PI_k and $CI_\alpha(FIS_k)$ are the FIS_k performance and coverage indices.

In the following, only the fuzzy partition corresponding to the *best* FIS will be kept. The initial rules are ignored as they will be determined by the OLS.

The use of standardized fuzzy partitions, with a small number of linguistic terms, ensures that rule premises are interpretable. Moreover, that choice eliminates the problem of *quasi redundant* rule selection, due to MF redundancy and underlined by authors familiar with these procedures, as [26].

However, the OLS brings forth a different conclusion for each rule. It makes rules difficult to interpret. We will now propose another modification of the OLS procedure to improve that point.

4.3 Rule conclusions

Reducing the number of distinct output values improves interpretability as it makes rule comparison easier. Rule conclusions may be assigned a linguistic label if the number of distinct conclusions is small enough. The easiest way to reduce the num-

ber of distinct output values is to adjust conclusions upon completion of the algorithm. We use the following method based on the k-means algorithm.

Given a number of distinct conclusions, c, and the set of training output values $y^i,\ i = 1, 2, \ldots, N$, the reduction procedure consists in:

- Applying the k-means method [14] to the N output values with c final clusters.
- For each rule of the rule base, replacing the original conclusion by the nearest one obtained from the k-means.

The vocabulary reduction worsens the system numerical accuracy on training data. However, rule conclusions are no more computed only with a least square optimization, and the gap between training and test errors may be reduced.

5 Results on benchmark data sets

This section presents, compares and discusses results obtained on two well known cases chosen in the UCI repository [16].

5.1 Data sets

The data sets are the following ones:

- *cpu-performance* (209 samples):
  Published by Ein-Dor and Feldmesser [7], this data set contains the measured CPU performance and 6 continuous variables such as main memory size or machine cycle time.
- *auto-mpg* (392 samples):
  Coming from the StatLib library maintained at Carnegie Mellon University, this case concerns the prediction of city-cycle fuel consumption in miles per gallon from 4 continuous and 3 multi-valued discrete variables.

The cpu-performance and auto-mpg datasets are both regression problems. Many results have been reported for them in the previous years [6, 20, 23, 25].

Experimental method

For the experiments, we use on each dataset a ten-fold cross validation method. The entire dataset is randomly divided into ten parts. For each part, the training is done on the nine others while testing is made on the selected one. Besides the stop criterion based upon the cumulated explained variance (equation 5), another

one is implemented: the maximum number of selected rules. The algorithm stops whenever any of them is satisfied.

5.2 Results and discussion

Tables 2 and 3 summarize the results for original and modified OLS on test subsets, for each data set. The original OLS algorithm is applied with only a slight modification: the conjunction operator in rule premises is the minimum operator instead of the product. Tests have been carried out to check that results are not significantly sensitive to the choice of the conjunction operator. The choice of the minimum allows a fair comparison between data sets with a different number of input variables.

Both tables have the same structure: the first column gives the average number of membership functions per input variable, the following ones are grouped by three. Each group of three corresponds to a different value of the allowed maximum number of rules, ranging from unlimited to five. The first one of the three columns within each group is the average number of rules, the second one the average performance index PI and the third one is the average coverage index CI_α, which corresponds to the activation threshold α given in the row label between parentheses. The first group of three columns corresponds to an unlimited number of rules, the actual one found by the algorithm being given in the #R column.

	#MF	#R	Perf.	Cov.	#R	Perf.	Cov.	#R	Perf.	Cov.	#R	Perf.	Cov.
orig. OLS (0)	27.8	39.8	69.78	1.00	15	74.54	1.00	10	98.11	1.00	5	150.38	1.00
orig. OLS (0.1)	27.8	39.8	32.52	0.75	15	33.32	0.40	10	46.65	0.23	5	113.26	0.03
orig. OLS (0.2)	27.8	39.8	32.30	0.59	15	40.67	0.21	10	61.99	0.09	5	113.26	0.03
mod. OLS (0)	2.7	11.3	41.95	0.99	11.3	41.95	0.99	10	45.57	0.97	5	71.96	0.47
mod. OLS (0.1)	2.7	11.3	41.95	0.99	11.3	41.95	0.99	10	46.01	0.95	5	71.71	0.45
mod. OLS (0.2)	2.7	11.3	41.92	0.98	11.3	41.92	0.98	10	45.07	0.91	5	69.27	0.40

Table 2
Cpu data comparison of original and modified OLS (averaged on 10 runs)

	#MF	#R	Perf.	Cov.	#R	Perf.	Cov.	#R	Perf.	Cov.	#R	Perf.	Cov.	#R	Perf.	Cov.
orig. OLS (0)	86.8	182.9	3.31	1.00	20	3.88	1.00	15	4.32	1.00	10	5.47	1.00	5	9.35	1.00
orig. OLS (0.1)	86.8	182.9	2.91	0.84	20	3.08	0.40	15	3.22	0.34	10	3.35	0.25	5	3.55	0.18
orig. OLS (0.2)	86.8	182.9	2.75	0.77	20	2.92	0.32	15	3.11	0.27	10	3.21	0.21	5	3.22	0.15
mod. OLS (0)	3.3	19.3	3.03	1.00	19.3	3.03	1.00	15	3.05	1.00	10	2.99	0.99	5	3.33	0.90
mod. OLS (0.1)	3.3	19.3	3.03	1.00	19.3	3.03	1.00	15	3.05	1.00	10	2.99	0.99	5	3.36	0.85
mod. OLS (0.2)	3.3	19.3	3.03	1.00	19.3	3.03	1.00	15	3.05	1.00	10	3.00	0.98	5	3.36	0.81

Table 3
Auto-mpg data comparison of original and modified OLS (averaged on 10 runs)

The discussion includes considerations about complexity, coverage and numerical accuracy of the resulting FIS.
Let us first comment the FIS structures. Clearly the original OLS yields a more complex system than the modified one, with a much higher number of membership functions per input variable. When the number of rules is not limited, the original OLS systematically has many more rules than the modified one. As to the performances, let us focus on rows one and four, which correspond to $\alpha = 0$, and on the first three columns, to allow an unlimited number of rules. This configuration allows a fair comparison between both algorithms. We see that, for both data sets, the modified algorithm has an enhanced performance. For the cpu data set, this has a slight coverage cost, with a loss of one percent, meaning that an average of two items in the data set is not managed by the systems obtained by the modified algorithm.
Examination of the next rows ($\alpha = 0.1$) shows that the modified algorithm systems have the same PI and CI_α than for the zero threshold. It is not at all the case for the original algorithm systems, where the coverage loss can be important (from 16 to 25 percent). This well demonstrates the lack of robustness of the original algorithm, as a slight change in input data may induce a significant output variation. The modified algorithm does not have this drawback.

Figure 6 shows the evolution of CI_0 and PI with the number of rules for each data set. As expected, the coverage index CI_0 is always equal to 1 for the original version. For the modified version, CI_0 quasi linearly increases with the number of rules. It means that each newly selected rule covers a set of data items, so that rules are likely to be used for knowledge induction, as will be shown in more details in section 6.

For a reasonable number of rules (≥ 10), we see that, while $CI_0 \approx 1$, the modified OLS has a much better accuracy than the original one.
For a low number of rules, the performance index PI has a very different behaviour for the two OLS versions. The poor accuracy (high values of PI) of the original algorithm can be explained by a low cumulated explained variance, and the good accuracy observed for the modified algorithm must be put into perspective of its poor CI_0. As the number of rules increases, both systems display a similar behaviour.

Another advantage of the modified OLS noticed in the benchmark results is the reduced execution time. When averaged over ten runs with an unlimited number of selected rules, for the CPU and auto cases, it respectively took 1.16 s and 5.65 s CPU on a 32 bit Xeon 3.2 GHz processor for the original OLS algorithm to complete, while it respectively took 1.03 s and 3.72 s for the modified version.

Table 4 compares the results of the modified OLS method and of other methods used in the literature (see [23]), in terms of Mean Absolute Error (criterion used in that reference paper), computed as $MAE = \frac{1}{n}\sum_{i=1}^{n} |\widehat{y}_i - y_i|$, n being the number of

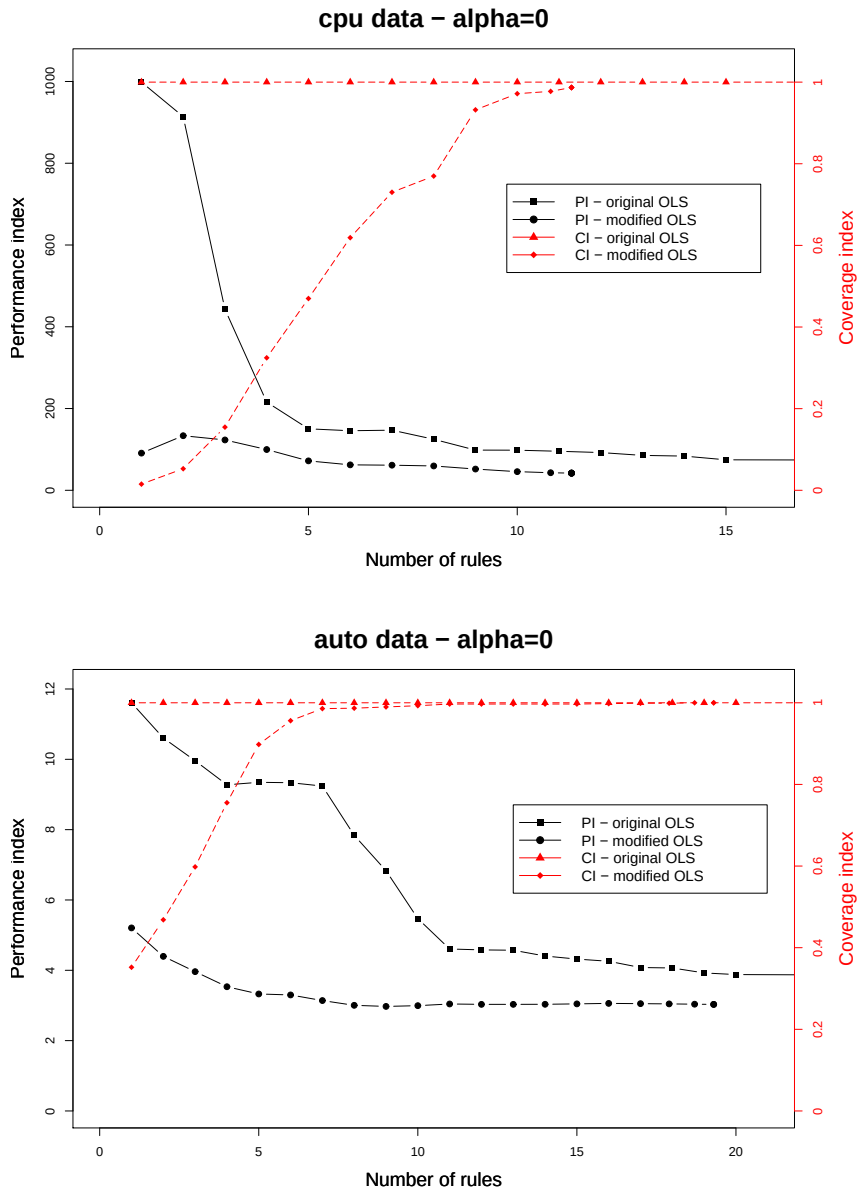


Fig. 6. Evolution of performance PI and coverage index CI_0 versus number of rules

active samples. The first method is a multivariate linear regression (LR), the second one is a regression tree (RT) and the third one is a neural network (NN). In all cases, the modified OLS average error is comparable to those of competing methods, or even better.

Data set	Mod.OLS	Linear regression	Decision tree	Neural network
CPU-Performance	28.6	35.5	28.9	28.7
Auto-mpg	2.02	2.61	2.11	2.02

Table 4
Comparison of mean absolute error of the modified OLS and other methods on test sets

We showed in this section that the proposed modifications of the OLS algorithm

Name	Description
pH	pH in the reactor
vfa	volatile fatty acid conc.
$qGas$	biogas flow rate
qIn	input flow rate
$ratio$	alkalinity ratio
CH_4Gas	CH_4 concentration in biogas
qCO_2	CO_2 flow rate

Table 5
Input variables

yield good results on benchmark data sets. We will thus use the modified OLS to deal with a real world case.

6 A real world problem

The application concerns a fault diagnosis problem in a wastewater anaerobic digestion process, where the ”living” part of the biological process must be monitored closely. Anaerobic digestion is a set of biological processes taking place in the absence of oxygen and in which organic matter is decomposed into biogas.

Anaerobic processes offer several advantages: capacity to treat slowly highly concentrated substrates, low energy requirement and use of renewable energy by methane combustion. Nevertheless, the instability of anaerobic processes (and of the attached microorganism population) is a counterpart that discourages their industrial use. Increasing the robustness of such processes and optimizing fault detection methods to efficiently control them is essential to make them more attractive to industrials. Moreover, anaerobic processes are in general very long to start, and avoiding breakdowns has significant economic implications.

The process has different unstable states: hydraulic overload, organic overload, underload, toxic presence, acidogenic state. The present study focuses on the acidogenic state. This state is particularly critical, and going back to a normal state is time consuming, thus it is important to detect it as soon as possible. It is mainly characterized by a low pH value (< 7), a high concentration in volatile fatty acid and a low alkalinity ratio (generally < 0.3).

Our data consist of a set of 589 samples coming from a pilot-scale up-flow anaerobic fixed bed reactor (volume=0.984 m^3). Data are provided by the LBE, a laboratory situated in Narbonne, France. Seven input variables summarized in table 5 were used in the case study.

The output is an expert assigned number from 0 to 1 measuring to what extent the actual state can be considered as acidogenic.

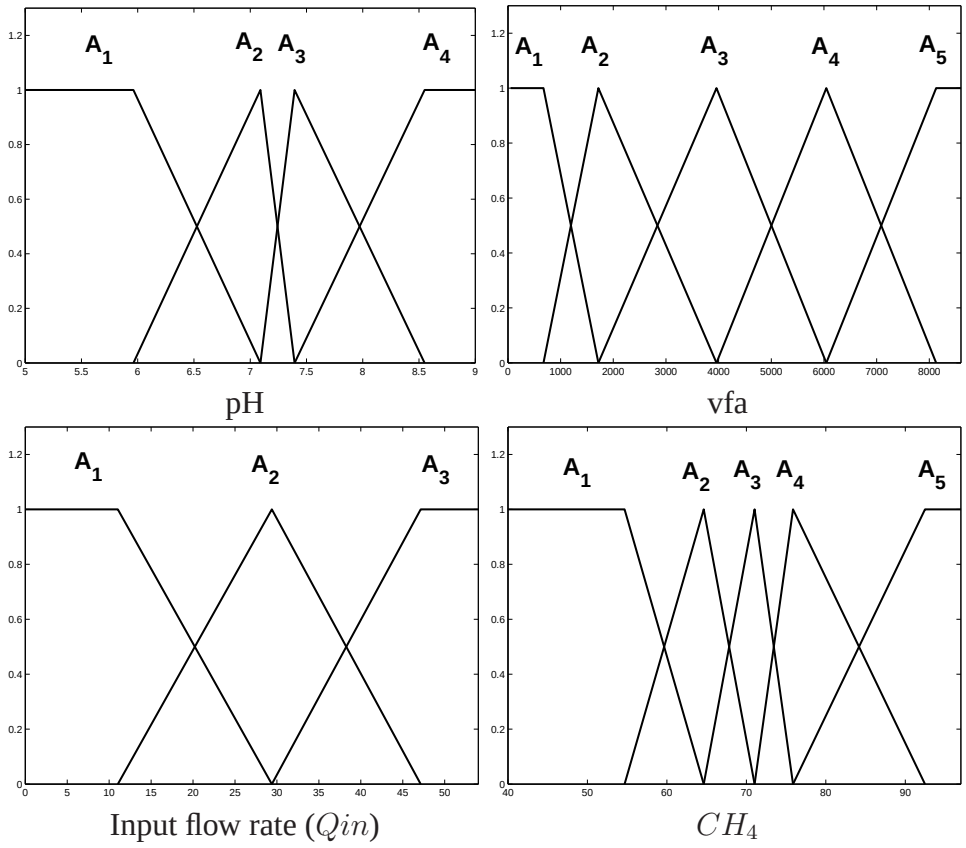


Fig. 7. Fuzzy partitions for wastewater treatment application

Fault detection systems in bioprocesses are usually based on expert knowledge. Multidimensional interactions are imperfectly known by experts. The OLS method allows to build a fuzzy rule base from data, and the rule induction can help experts to refine their knowledge of fault-generating process states.

Before applying the OLS, we select the fuzzy partition with the refinement algorithm described in section 4, which yields the selection of four input variables : pH, vfa, Qin and CH_4Gas. The membership functions are shown in Figure 7. Notice that each membership function can be assigned an interpretable linguistic label.

Results and discussion

We apply the OLS procedure to the whole data set, obtaining a rule base of 53 rules and a global performance $PI = 0.046$.

Rule base analysis

Analyzing a rule base is usually a very long task, and must be done anew with each different problem. Here are some general remarks:

- **Rule ordering**: amongst the 589 samples, only 35 have an output value greater than 0.5 (less than 10%), while there are 12 rules out of 53 that have a conclusion

greater than 0.5 (more than 20%). Moreover, 8 of these rules are in the first ten selected ones (the first six having a conclusion very close to one). This shows that the algorithm first select rules corresponding to "faulty" situations. It can be explained by the fact that the aim of the algorithm is to reduce variance, a variance greatly increased by a "faulty" sample. This highlights a very interesting characteristic of the OLS algorithm, which first selects rules related to rare samples, often present in fault diagnosis.

- **Out of range conclusions**: each output in the data set is between 0 and 1. This is no more the case with the rule conclusions, some of them being greater than 1 or taking negative values. It is due to the least-square optimization method trying to improve the accuracy by adjusting rule conclusions, without any constraint. This is one of the deficiencies of the algorithm, at least from an interpretability driven point of view.

Removing outliers

The fact that rules corresponding to rare samples are favored in the selection process has another advantage: the ease with which outliers can be identified and analyzed. In our first rough analysis of the rule base, two specific rules caught our attention :

- **Rule 5** : If pH is A_3 and vfa is A_1 and Qin is A_1 and CH_4 is A_1, then output is 0.999
- **Rule 6** : If pH is A_4 and vfa is A_3 and Qin is A_3 and CH_4 is A_5, then output is 1

Both rules indicate a high risk of acidogenesis with a high pH, which is inconsistent with expert knowledge of the acidogenic state. Further investigation shows that each of these two rules is activated by only one sample, which does not activate any other rule. Indeed, one sample has a pH value of 8.5 (clearly not acid) and the other one has a pH of 7.6, together with an alkalinity ratio (which should be low in an acidogenic state) greater than 0.8.

These two samples being labeled as erroneous data (maybe a sensor disfunction), we remove them from the data set in further analysis.

This kind of outliers cannot be managed using traditional noise removal filtering techniques, it requires expert examination to decide whether they should be removed from learning data.

We renew the OLS procedure on the purified data, and we also perform a reduction of the output vocabulary, to improve interpretability.

Performance with reduced output vocabulary

The final rule base has 51 rules, the two rules induced by erroneous data having disappeared. The output vocabulary is reduced from 49 distinct values to 6 differ-

ent ones, all of them constrained to belong to the output range. Figure 8 shows the rule conclusion distribution before and after vocabulary reduction. On the left subfigure, two dotted lines have been added to show the observed output range $[0-1]$. Rules are easier to interpret, while the distribution features are well conserved. The new system performance is PI=0.056, which corresponds to an accuracy loss of 15 percent.

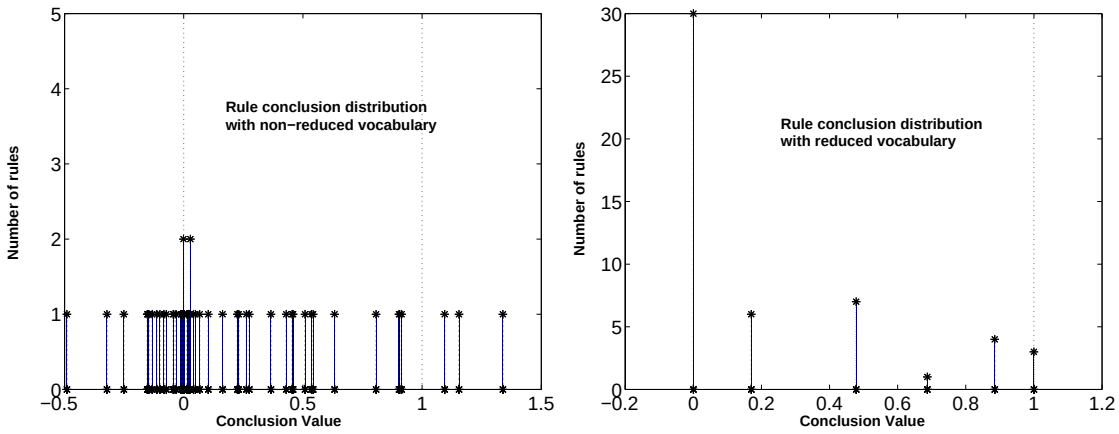


Fig. 8. Impact of vocabulary reduction on rule conclusions

To test the rule base representativity, we did some experiments on increasing the activation threshold α. Up to $\alpha = 0.5$, only one sample amongst the 587 ones is not covered by the rule base, which is a good sign as to the robustness of our results.

Another interesting feature is that 100% of the samples having an output greater than 0.2 are covered by the first twenty rules, allowing one to first focus on this smaller set of rules to describe critical states.

Figure 9 illustrates the good qualitative predictive quality of the rule base: we can expect that the system will detect a critical situation soon enough to prevent any collapse of the process. From a function approximation point of view, the prediction would be insufficient. However, for expert interpretation, figure 9 is very interesting. Three clusters appear. They can be labeled as *Very low risk*, *Non neglectable risk* and *High risk*. They could be associated to three kinds of action or alarms.

From a fault detection point of view, some more time should be spent on the few *faulty* samples that wouldn't activate a fault detection trigger set at 0.2 or 0.3. They have been signaled to experts for further investigation. Each rule fired by those five samples (asterisk and diamond in figure 9) is also activated by about a hundred other samples which have a very low acidogenic state. It may be difficult to draw conclusions from these five samples.

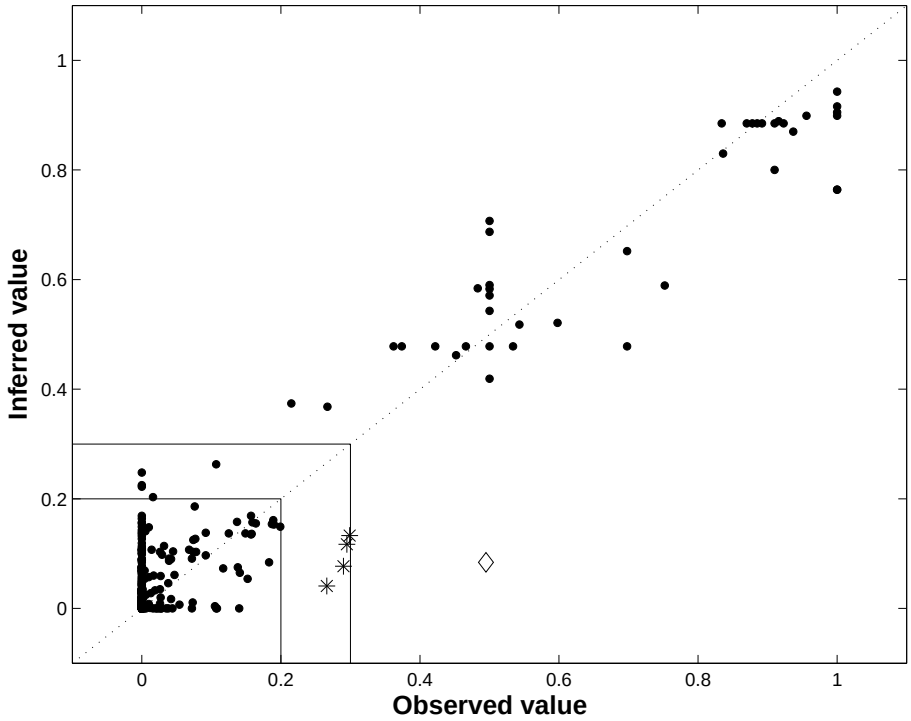


Fig. 9. Prediction with 6 conclusion values. • : Detection with trigger > 0.2 ; $\{*, \diamondsuit\}$: Non-detection with trigger = 0.2 ; $\diamondsuit$: Non-detection with trigger = 0.3

7 Conclusion

Orthogonal transform methods are used to build compact rule base systems. The OLS algorithm is of special interest as it takes into account input data as well as output data.

Two modifications are proposed in this paper. The first one is related to input partitioning. We propose to use a standardized fuzzy partition with a reduced number of terms. This obviously improves linguistic interpretability but also avoids the occurrence of an important drawback of the OLS algorithm: redundant rule selection. Moreover, it can even enhance numerical accuracy.

The second way to improve linguistic interpretability is to deal with rule conclusions. Reducing the number of distinct values used by the rules has some effect on the numerical accuracy measured on the training sets, but very little impact on the performance obtained on test sets.

We have successfully applied the modified OLS to a fault detection problem. Our results are robust, interpretable, and our predictive capacity is more than acceptable. The OLS was also shown able to detect some erroneous data after a first brief analysis. When dealing with applications where the most important samples are rare, OLS can be very useful.

We would like to point out the double interest of properly used fuzzy concepts in a numerical technique. Firstly, linguistic reasoning with input data, which is only relevant with readable input partitions, takes into account the progressiveness of biological phenomena which have a high intrinsic variability. Secondly, a similar symbolic reasoning can be used on output data. Though interesting for knowledge extraction, this is rarely considered.

Let us also underline the proposed modifications could benefit to all the similar algorithms based on orthogonal transforms, for instance the TLS (Total Least Squares) method [29] which seems to be of particular interest.

A thorough study of the robustness of this kind of models is still to be carried out. It should include a sensitivity analysis of both algorithm parameters and data outliers with respect to the generalization ability. The sensitivity analysis could be sampling-based or be based on statistical techniques (for instance decomposition of variance). Similarly the rule selection procedure could be refined by extending classical backward-forward stepwise regression procedures to the fuzzy OLS algorithm.

Contrary to other methods, OLS does not perform a variable selection, which can be a serious drawback. Future work should also focus on combining an efficient variable selection method with the OLS rule selection.

References

[1] Marwan Bikdash. A highly interpretable form of sugeno inference systems. *IEEE Transactions on Fuzzy Systems*, 7 (6):686–696, December 1999.

[2] Jorge Casillas, Oscar Cordon, Francisco Herrera, and Luis Magdalena. *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*. Springer, 2003.

[3] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *Int. J. Control*, 50:1873–1896, 1989.

[4] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2:302–309, March 1991.

[5] J. Valente de Oliveira. Semantic constraints for membership functions optimization. *IEEE Transactions on Systems, Man and Cybernetics. Part A*, 29(1):128–138, 1999.

[6] W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for rule-based data understanding. In *Proceedings of the IEEE*, volume 92 (5), pages 771–805, April 2004.

[7] P. Ein-Dor and J. Feldmesser. Attributes of the performance of central processing units: a relative performance prediction model. *Communications of the ACM*, 30 (4):308–317, 1987.

[8] Jairo Espinosa and Joos Vandewalle. Constructing fuzzy models with linguistic integrity from numerical data-afreli algorithm. *IEEE Transactions on Fuzzy Systems*, 8 (5):591–600, October 2000.

[9] Pierre-Yves Glorennec. *Algorithmes d'apprentissage pour systèmes d'inférence floue*. Editions Hermès, Paris, 1999.

[10] Pierre-Yves Glorennec. *Constrained optimization of fuzzy decision trees*, pages 125–14, in [2], 2003.

[11] Serge Guillaume. Designing fuzzy inference systems from data: an interpretability-oriented review. *IEEE Transactions on Fuzzy Systems*, 9 (3):426–443, June 2001.

[12] Serge Guillaume and Brigitte Charnomordic. Generating an interpretable family of fuzzy partitions. *IEEE Transactions on Fuzzy Systems*, 12 (3):324– 335, June 2004.

[13] D. Han, I.D. Cluckie, D. Karbassioum, J. Lawry, and B. Krauskopf. River flow modeling using fuzzy decision trees. *Water Resources Management*, 16(6):431–445, 2002.

[14] John A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[15] J. Hohensohn and J. M. Mendel. Two pass orthogonal least-squares algorithm to train and reduce fuzzy logic systems. In *Proc. IEEE Conf. Fuzzy Syst.*, pages 696–700, Orlando, Florida, June 1994.

[16] http://www.ics.uci.edu/∼mlearn/MLRepository.html. UCI repository of machine learning databases, 1998.

[17] F. Kossak, M. Drobics, and T. Natschlger. Extracting knowledge and computable models from data - needs, expectations, and experience. In *Proc. IEEE Int. Conf. on Fuzzy Systems*, pages 493–498, Budapest, 2004.

[18] Swarup Medasani, Jaeseok Kim, and Raghu Krishnapuram. An overview of membership function generation techniques for pattern recognition. *International Journal of Approximate Reasoning*, 19:391–417, 1998.

[19] G. Miller. The magical number seven, plus or minus two. *The Psychological Review*, 63:81–97, 1956.

[20] Witold Pedrycz. Logic-driven fuzzy modeling with fuzzy multiplexers. *Engineering Applications of Artificial Intelligence*, 17:383–391, 2004.

[21] Witold Pedrycz. *Fuzzy control and fuzzy systems*. Studies in Fuzziness. Research Studies Press Ltd, Taunton, Somerset, England, 2nd edition, 1993.

[22] Witold Pedrycz. Why triangular membership functions? *Fuzzy sets and Systems*, 64 (1):21–30, 1994.

[23] J. Quinlan. Combining instance-based model and model-based learning. In *Proceedings of the 10th ICML*, pages 236–243, San Mateo, CA, 1993.

[24] Enrique H. Ruspini. A new approach to clustering. *Information and Control*, 15:22–32, 1969.

[25] R. Setiono and J.Y.L Thong. An approach to generate rules from neural networks for regression problems. *European Journal of Operational Research*, 155:239–250, 2004.

[26] Magne Setnes. *Simplification and reduction of fuzzy rules*, pages 278–302. in [2].

[27] Li-Xin Wang and Jerry M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least squares learning. *IEEE Transactions on Neural Networks*, 3:807–814, 1992.

[28] P.J. Woolf and Y. Wang. A fuzzy logic approach to analyzing gene expression data. *Physiological Genomics*, 3:9–15, 2000.

[29] John Yen and Liang Wang. Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Transactions on Systems, Man and Cybernetics*, 29 (1):13–24, February 1999.