Using the OLS algorithm to build interpretable rule bases: an application to a depollution problem

S. Destercke, Serge Guillaume, Brigitte Charnomordic

Using the OLS algorithm to build interpretable rule bases: an application to a depollution problem

Sebastien Destercke, Serge Guillaume and Brigitte Charnomordic

***Abstract*— One of the main advantages of fuzzy modeling is the ability to yield interpretable results. Amongst these modeling methods, the OLS algorithm is a mathematically robust technique that allows to induce a fuzzy rule base from a set of training data. It does so by using linear regression to select the most important rules. However, the original OLS algorithm only relies upon numerical accuracy, and doesn't take interpretability matters into account. Thus, we propose some modifications to the original method so that it builds interpretable rule bases.**

I. Introduction

Unlike "black-box" models (e.g. neural networks), fuzzy modeling techniques are likely to give interpretable results, provided that some constraints are respected. This feature of fuzzy models is a real asset in domains where human understanding of processes is essential (e.g. climate evolution, biological industry).

This explains why interpretability issues in fuzzy modeling have deserved special attention in the literature [1]. It is commonly accepted that interpretability requires a small number of consistent membership functions for each input and a reasonable number of rules in the fuzzy system.

On the other hand, efficient and robust numerical methods are needed to deal with large amount of data. The OLS algorithm, a particular case of more general techniques using orthogonal transformation [2], is among such methods. Given input membership functions, the OLS algorithm selects the most important rules by using linear regression techniques. However, the original OLS algorithm was designed on accuracy criteria, without taking account of interpretability.

In this paper, we propose some modifications to make the OLS algorithm bring on interpretable rule bases, without suffering too much loss of accuracy. After a brief reminder of the original method in section II, these proposals and their application to benchmark problems are developed in sections III and IV. Finally, section V shows an application of the algorithm to a real-world fault detection depollution problem.

II. Original OLS algorithm

After introducing some notations, we recall how the original algorithm works.

Sebastien Destercke is with the Institute of Radiological Protection and Nuclear Safety (IRSN), Cadarache, France (email: sebastien.destercke@irsn.fr).

Serge Guillaume is with the Cemagref, Umr Itap, BP 5095, 34196 Montpellier Cedex, France (email: serge.guillaume@montpellier.cemagref.fr).

Brigitte Charnomordic is with the INRA, Umr ASB, 2 place Viala, 34060 Montpellier Cedex, France (email: bch@ensam.inra.fr)

A. Notations

We write a zero order Takagi Sugeno model as a set of r fuzzy rules such as:

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

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

(x, y) denote N input-output pairs of a data set, where $x \in \mathbb{R}^p$ and $y \in \mathbb{R}$. The zero order Takagi Sugeno model output for the ith pair can then be written as follows:

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

where $\mu_{A_j^q}(x_j^i)$ is the membership function value of x_j^i in the q^{th} rule, and $\bigwedge$ the conjunction operator used to combine rule premise elements. In the sequel, $\bigwedge \mu_{A_j^q}(x_j^i)$ is called the rule standardized firing strength and is noted $w^q(x^i)$, the previous equation then becomes:

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

B. Original algorithm

In the original algorithm [3], N rules are first built from the samples (one for each pair in the data set). Hohensohn and Mendel [4] proposed the following Gaussian membership functions:

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

for the jth dimension of the ith rule, with the optimal value of σ_j depending on the problem at hand.

Once these membership functions have been built, a first step consist of mapping input variables into a linear space, by using fuzzy basis functions (FBF) [3]. Given the membership functions, a FBF $p^i(x^i)$ is the relative contribution of the ith rule, built from the ith example, to the inferred output:

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

After the inputs have been mapped by FBF, equation (1) can be written $\widehat{y^i} = \sum_q p^q(x^i)\ \theta^q$, where the only unknown value, at this stage, is θ^q. Thus, we have a linear combination, and the rule conclusions (θ^q) are the parameters to optimize. The overall system can be rewritten in the matrix form:

$$y = P\theta + E$$

where y is the true system output, P is a matrix where the column i is the FBF $p^i(x)$, θ are the parameters to optimize, and E is an error matrix, supposed to be uncorrelated with P. Let us note that the element p_{ji} of the matrix P represents the ith rule firing strength for the jth pair.

We thus have a linear form, to which an orthogonal least square can be applied. P is decomposed by a Gram-Schmidt procedure into an orthogonal matrix M and an upper triangular matrix A. The system then becomes:

$$y = MA\theta + E$$

If we write $g = A\theta$, then the orthogonal least square solution of this system is:

$$\widehat{g_i} = \frac{m_i^T y}{m_i^T m_i}, 1 \leq i \leq r$$

where m_i is the ith column of the orthogonal matrix M. Optimal $\widehat{\theta}$ is then computed from $\widehat{g}$.

Thanks to the orthogonal nature of M (i.e. no covariance), each individual vector (i.e. rule) contribution to the explained variance of the observed output can be computed. At each iteration, the algorithm selects the vector m_i that maximizes the explained variance (i.e. the most important rule not already selected). The explained variance, which is also the selection criterion, is computed as follows:

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

and the rule selection stops when the cumulated explained variance ($\sum_{i=1}^{r} [xVar]_i$) reaches a satisfactory level ϵ (typically, 0.99).

On completion of the selection procedure, selected m_i still contain some information about the unselected rules. Also, Hohensohn and Mendel [4] propose to re-run the algorithm, but only with the optimization phase (no selection is done during this phase).

The OLS algorithm, as described here, is numerically efficient, but has many drawbacks when one also wants interpretable results with the aim of knowledge extraction.

III. Requirements for building and analyzing interpretable rule bases

This section presents the requirement for results to be interpretable and which criteria we use to analyze a given rule base.

A. Requirements for interpretability

A first requirement for fuzzy rule bases to be interpretable is a system with a reasonable number of rules. This requirement is already fulfilled by the OLS algorithm, which consists of selecting a limited number of rules. Moreover, if one accepts to lower the numerical accuracy, the stopping criterion can be related to the number of selected rules, instead of a cumulated explained variance.

A second requirement is to use interpretable membership functions as input fuzzy sets [5]. The necessary conditions for the membership functions to be interpretable have been studied by many authors in the past (see, e.g. [6]), and can be achieved by the use of standardized fuzzy partitions, defined as follows:

$$\begin{cases} \forall x \sum_{f=1,2,...,M} \mu_f(x) = 1 \\ \forall f\ \exists\ x\ such\ as\ \mu_f(x) = 1 \end{cases} \quad (3)$$

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 3 means that any point belongs at most to two fuzzy sets when the fuzzy sets are convex. A standardized fuzzy partition is shown on Figure3(b).

The last requirement is to impose a small number of distinct output value in the zero order Takagi Sugeno system.

B. Evaluation Criteria

We first introduce what we call the coverage index, parameterized by an activation threshold. As shown below, we use this criterion as a practical tool, both to measure the robustness of the system and to evaluate the reliability of the rule base with respect to the data.

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 there's a rule in the rule base 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.

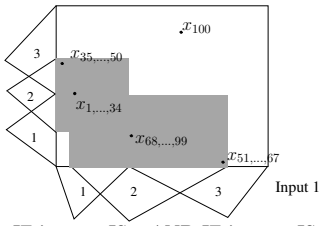


Fig. 1. Input domain rule coverage

Figures 1 and 2 illustrate coverage indices on a toy example. We see that increasing the activation threshold can induce an important decrease in the coverage index. For instance, here CI drops from 0.99 to 0.66 if the activation

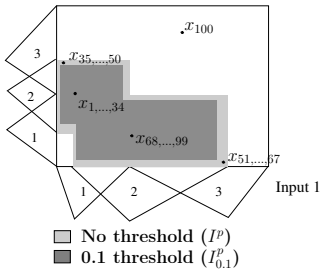


Fig. 2. Input domain rule coverage with $\alpha = 0.1$

threshold increases by 0.1. Using the coverage index gives indications as to:

- Exception data: a $CI_\alpha \approx 1$ is often the consequence of isolated samples not covered by the rule base. Their detection is facilitated by the use of CI,
- System robustness and knowledge reliability: sensitivity of CI_α to α is a way to measure system robustness to small changes. A fast decreasing CI_α when increasing α indicates that at least some of the rules are not really representative of the data, and thus of the system. The reliability of the knowledge represented by such rules is questionable.

From our point of view, the advantage of the coverage index is that it allows a quick and easy first analysis of the rule base that provides useful information.

To measure the numerical accuracy of our systems, and thus their predictive capacity, we use the following error index (n is the number of active samples) :

$$PI = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left\|\widehat{y^i} - y^i\right\|^2}$$

IV. Proposed modifications

After describing the changes made to the original method, we compare the modified and original OLS on two benchmark problems.

A. Proposed modifications

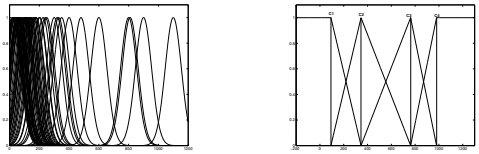
From an interpretability standpoint, the OLS algorithm has two main drawbacks: too many input fuzzy membership functions and distinct rule conclusions. We thus propose two changes:

- Building and using interpretable membership functions in the selection and the least-square optimization steps,
- Reducing the number of distinct output values by a clustering process.

Figure 3(a) shows membership functions generated by the original algorithm. The result is clearly not interpretable. Some membership functions are quasi-redundant and many fuzzy sets are not distinguishable. The unlimited boundary feature of Gaussian functions is here a disadvantage: it yields $CI_0 = 1$, even if there's only one rule, but this perfect coverage index is likely to drop as soon as α will increase.

Due to their properties [7], we choose to build standardized partitions with triangular fuzzy sets (except at the domain edges, where we build semi-trapezoidal fuzzy sets). Such a M-term standardized fuzzy partition is totally defined by M values, corresponding to the fuzzy set centers. There are many ways to build interpretable fuzzy partitions. We want to have an interpretable result while preserving a good numerical accuracy. We choose to use a non greedy refinement based algorithm for partition design, tailored to calculate the fuzzy set bounds and the number of terms in the fuzzy partition. The algorithm starts with the simplest possible system (a single fuzzy set for each input), and works by successive refinement of the input dimensions inducing the best accuracy gain. The reader is referred to [8] for details. The outcome of the algorithm is an interpretable fuzzy set partition for each input.

Figure 3.b shows a 4-term standardized partition induced from the same data used for figure 3.a.



(a) Original partition (b) Standardized partition

Fig. 3. Fuzzy partitions

Let us note that the use of standardized partitions eliminates the problem of quasi-redundant rule selection, a known drawback of the OLS procedure.

The OLS algorithm is then applied, with the purpose of building fuzzy rules, thus producing a system with interpretable rule premises. Nevertheless, it still gives forth as many distinct output values as there are rules in the rule base. This is why we use the following simple method to reduce the number of distinct rule conclusions:

1) Set the desired number of final distinct rule conclusion to c
2) Apply the k-means method with c final clusters to the N data output values
3) For each rule, replace the conclusion value by the closest one found by a k-means clustering procedure.

B. Benchmark results

To be sure that our changes to the original method do not induce too much loss of accuracy, we compare both algorithms on the CPU-performance and auto-mpg benchmarks, two regression problems taken from the UCI repository [9]. CPU-performance case has 6 continuous variables as its input, and the CPU-performance as its output. The data set contains 206 samples. Auto-mpg case has 4 continuous and 3 multi-valued discrete variables as its input, and the measured city-cycle fuel consumption as its output. The data set contains 392 samples

Tests were achieved by doing a ten-fold cross validation. Data sets were randomly divided into ten parts. For each

part, training was done on the nine others, and testing was achieved on the selected one. Besides the cumulated explained variance stop criterion, we also imposed a maximum number of selected rules.

TABLE I

RESULT COMPARISON BETWEEN MODIFIED AND ORIGINAL OLS METHODS ON THE CPU AND AUTO-MPG PROBLEMS (AVERAGED ON 10 RUNS)

	CPU problem						
	#MF	#R	PI	CI	#R	PI	CI
Orig. OLS (α=0)	27.8	39.8	69.8	1.00	10	98.1	1.00
Orig. OLS (α=0.1)	27.8	39.8	32.5	0.75	10	46.6	0.23
Mod. OLS (α=0)	2.7	11.3	41.9	0.99	10	45.6	0.97
Mod. OLS (α=0.1)	2.7	11.3	41.9	0.99	10	46	0.95
Auto-mpg problem							
Orig. OLS (α=0)	86.8	182.9	3.31	1.00	10	5.47	1.00
Orig. OLS (α=0.1)	86.8	182.9	2.91	0.84	10	3.35	0.25
Mod. OLS (α=0)	3.3	19.3	3.03	1.00	10	2.99	0.99
Mod. OLS (α=0.1)	3.3	19.3	3.03	1.00	10	2.99	0.99

Table I summarizes the results obtained from the test. Values are mean values computed over 10 cross-validation runs. The first column contains the mean number of fuzzy sets per input data, and the following columns are grouped by three. For each group, the first column contains the mean number of rules selected by the algorithm, the second one the average performance index and the third one the average coverage index value. The first group corresponds to an unlimited number of rules, while the second one is obtained by considering only the first ten rules (at most). Tests were done for two different activation thresholds: $\alpha = 0$ and $\alpha = 0.1$.

Please note that the order of magnitude of PI depends on the data set variance, which is much higher for the CPU case than for the auto-mpg one.

Table I shows that, in both cases, the original algorithm yields a more complex fuzzy system: inputs have more membership functions and, on average, more rules are selected (this is particularly true for the auto-mpg case). If we compare the performances obtained with an unlimited number of rules and $\alpha = 0$, we see that the modified version gives better results than the original OLS (41.9 against 69.8 for the CPU case, and 3.03 against 3.31 for the auto-mpg case). In the CPU case, the modified version induces a slight coverage index loss when compared to the original version.

If we set the activation threshold to 0.1, we see that a lot of data are not covered by the original version (CI drops from 100 to 75 % in the CPU case, and to 84 % in the auto-mpg case), while the modified version is not sensitive to this change. The same effect can be observed with a drastic drop in coverage when we limit the number of selected rules to 10 (3 last columns of table I), which can be interesting to make the rule base analysis easier.

Figure 4 shows the evolution of CI and PI with the number of rules in the system for both methods and for the CPU problem (behavior for the Auto-mpg problem is similar). As expected, $CI_0 = 1$ for the original version

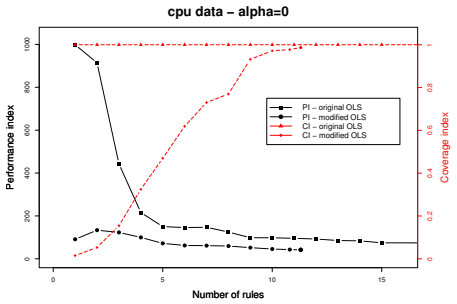


Fig. 4. Evolution of PI and CI_0 versus number of rules

whatever the number of rules. For the modified version, CI increases quasi linearly, which means that each added rule covers a significant amount of samples. Hence it can be used for knowledge induction. The difference of behavior between the PI of the different versions for a low number of rules can be easily explained: the original version has a low PI because of the poor amount of explained variance, and the good PI of the modified version must be balanced by the poor CI_0. As the number of rules increases, the two algorithms display a similar behavior.

Table IV-B compares the results of the modified OLS method and of other methods (see [10]), 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 active samples. LR stands for multivariate linear regression, RT for regression tree and NN for neural network. In all cases, the modified OLS average error is comparable or even better than those of competing methods.

TABLE II

COMPARISON OF THE MODIFIED OLS AND OTHER METHODS

Data set	Mod.OLS	LR	RT	NN
CPU-Performance	28.6	35.5	28.9	28.7
Auto-mpg	2.02	2.61	2.11	2.02

The results presented above show that we can use the modified OLS on a real-world problem to extract knowledge.

V. REAL-WORLD APPLICATION

A. Presentation

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

TABLE III
INPUT VARIABLES

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 biogas concentration
qCO_2	CO_2 flow rate

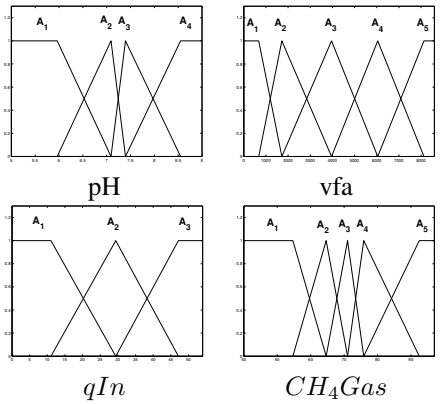


Fig. 5. Fuzzy partitions for wastewater treatment application

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. Data are provided by the LBE, an INRA laboratory located in Narbonne, France. Seven input variables summarized in table III were used in the case study.

The output varies from 0 to 1 indicating to what extent the actual state can be considered as acidogenic. Fault detection systems in bioprocesses are usually based on expert knowledge. Multidimensional interactions are imperfectly known by experts. The modified 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.

B. First analysis

The refinement algorithm described in section IV (see [8] for details) yields the selection of four input variables: pH, vfa, qIn and CH_4Gas.

We first run the original OLS on our data restricted to those four input variables and to the output. It generates a system with 53 rules, but where each fuzzy partition counts more than 500 fuzzy sets. The error index is $PI = 0.074$. Moreover, the CI of the system built with the original OLS drops from 100% to 35% as soon as the activation threshold increases from 0 to 0.1.

We then apply our algorithm, with the membership functions automatically determined as explained in section IV. They are shown in Figure 5. Notice that each membership function can be assigned an interpretable linguistic label.

The modified OLS gives us also a system of 53 rules. The error index is now $PI = 0.046$. Some conclusions of general interest can be drawn from these first results.

- **Rule ordering**: amongst the 589 samples, only 35 have an output value greater than 0.5, while there are 12 rules out of 53 that have a conclusion greater than 0.5. Moreover, 8 of these rules are in the first ten selected ones (the first six having a conclusion very close to one): the algorithm first selects rules corresponding to "faulty" situations. The explanation is that the algorithm is based on explained 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, very important 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, done without any constraints on the conclusion values. 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. These two samples being labeled as erroneous data (maybe a sensor dysfunction), we remove them from the data set in further analysis.

C. Final system

The final rule base (after a new application of the modified OLS) has 51 rules, the two rules induced by erroneous data having disappeared. In an extra step, the output vocabulary is reduced from 49 to 6 distinct values, all of them constrained

to belong to the output range. The new system performance is PI=0.056 (i.e. a 15 percent accuracy loss). This loss was judged acceptable for our purpose (i.e. knowledge discovery). Concerning coverage index and activation threshold, tests showed that 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 6 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 6 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.

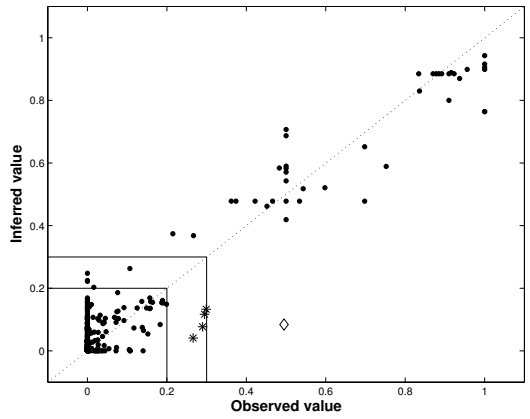


Fig. 6. Prediction with 6 conclusion values. •: Detection with trigger > 0.2 ; $\{*, \diamondsuit\}$: Non-detection with trigger = 0.2 ; $\diamondsuit$: Non-detection with trigger = 0.3

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 signalled to experts for further investigation. Each rule fired by those five samples (asterisk and diamond in figure 6) 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.

VI. Conclusions

The OLS algorithm (and orthogonal transform method in general) was originally designed in order to build compact rule bases with an efficient numerical accuracy, but with almost no interpretability power.

In this paper, two modifications are proposed. The first one consists in using standardized input partitions. This improves linguistic interpretability and avoids the selection of quasi-redundant rules by the OLS. The second proposition is to reduce the number of distinct conclusions to a handful. Tests have shown that, if the effect of this reduction on the training data can lower accuracy, it is hardly true on the test data.

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

The modified OLS is a simple and efficient numerical tool that allows to build relatively small interpretable rule bases for regression problems (which is interesting, since most of the existing algorithms focus on classification problems). As is shown by the application, it can be very useful as a support for expert analysis, particularly in fault detection problems.

The proposed modifications could benefit to all orthogonal transforms (see, e.g. [2]), and a next step of this work would be to undertake a thorough study of the advantages of such methods, together with a study of robustness and sensitivity to the algorithm parameters. Moreover, it would be interesting to see how classical backward-forward stepwise regression procedures could help in the result analysis. Another perspective is to apply this method in conjunction with an efficient variable selection method.

Acknowledgment

The authors would like to thank the LBE laboratory of INRA Narbonne for allowing us to use their data, and Laurent Lardon for the helpful interpretation of our results.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

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

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

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

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