Parameter optimization of a Fuzzy Inference System using the FisPro open source software

Serge Guillaume, Brigitte Charnomordic

in : IEEE International Conference on Fuzzy Systems, Brisbane, AUS,10-15 June 2012

Parameter optimization of a Fuzzy Inference System using the FisPro open source software

Serge Guillaume, Brigitte Charnomordic

***Abstract*—This paper proposes a flexible optimization sequence that can be applied to any parameter of a fuzzy inference system. Interrelated parameters can be optimized together, and criteria include system accuracy and coverage. The fuzzy inference system structure is preserved and constraints are imposed to respect the fuzzy partition semantics. The procedure described here uses a Solis & Wets based algorithm, but the approach remains valid for other optimization techniques, provided that they accept semantic constraints. The optimization sequence is implemented in an open source software, *FisPro*, made for fuzzy inference system design and tuning.**

I. Introduction

Fuzzy Inference Systems (FIS) are widely used in system modeling either for classification or regression purposes. Their acknowledged originality stems from the linguistic interpretability of the fuzzy rules while they are able to reach a comparable level of accuracy to the one gained by alternative methods.

Whatever the technique used to design the FIS, their numerical accuracy can be improved by optimizing their different parts. This objective should be carried out without loosing the system interpretability [2], [10].

Historically, research teams have been interested in different levels of FIS optimization falling into two main categories: structure and parameter optimization. Various methods are available for both levels: analytical methods, mainly based on gradient descent [3], or evolutionist ones, typically genetic algorithms [1] as well as ant colony techniques [20].

Structure optimization deals with the definition of variables and rules. Feature selection can be carried out at the global level as well as at the local rule level. The difference between local and global levels is that the removed input variable is no more available for any of the rules when unselected at the global level. The number of membership functions (MF) can also be tuned by appropriate algorithms. Rule base optimization can be achieved using incremental procedures, fuzzy set or cluster merging, or else rule selection can be done using statistical criteria. A survey of those structure optimization methods can be found in [14].

This paper focuses on parameter optimization, MF parameters and rule conclusions. It does not introduce a new algorithm, but proposes an innovative optimization sequence for the different FIS parameters. The optimization procedure includes a ten-fold cross validation and a simple way to define the final optimized system. The procedure is generic, and it is illustrated using a simple and efficient evolutionist algorithm.

All of the developments are implemented in an open source software called *FisPro*[1], which is used to illustrate the approach with two benchmark data sets from the UCI repository.

The next section recalls the main features of *FisPro*, and the importance of fuzzy partitioning for system interpretability. The optimization process is detailed in Section 3. Section 4 is dedicated to the experimental setup and results. Finally, the main conclusions are summarized in Section 5.

Serge Guillaume is with the Cemagref, UMR ITAP, BP 5095, 34196 Montpellier, France (email: serge.guillaume@irstea.fr).

Brigitte Charnomordic is with the INRA/SupAgro, UMR MISTEA, 34060 Montpellier, France (email:bch@supagro.inra.fr).

II. FisPro: design and main features

FisPro is a toolkit for FIS design and optimization. Four points are of interest:

- the FIS interpretability. This is the main originality of FisPro, as interpretability is guaranteed in each step: variable partitioning, rule induction, rule base simplification, optimization.
- a modular, portable software architecture that allows platform independence and facilitates extension writing.
- the open source license.
- the C++ function library, which can be used independently of the interface, for instance to design batch sequences of optimization runs.

Let us recall the main functionalities of *FisPro*.

A. Designing FIS from expert knowledge and data

FisPro allows to design fuzzy systems from the expert knowledge available in a given field. It also implements the automatic design of a fuzzy inference system from the numerical data related to the problem under study.

- FIS design from expert knowledge
  Two inference mechanisms are available in *FisPro*: conjunctive rules, that represent joint sets of possible input and output values, and another kind of fuzzy rules [26], [8], called implicative rules, which model the input

[1] http://www.inra.fr/mia/M/fispro/

output relationship using a fuzzy implication, and generalize the classical logic rules. They encode constraints and each rule defines a fuzzy restriction on the set of possible values. The difference of nature between conjunctive and implicative rules impacts rule combination: while several conjunctive rules are combined disjunctively (as they widen the scope of a single rule), implicative rules are combined conjunctively, because several constraints lead to a more restricted feasible set of allowed situations than a single constraint. All details can be found in [19]. Implicative rules are ususally designed by experts.

- Learning FIS from data
  To guarantee interpretability, *FisPro* proposes a learning approach [16] decomposed into several steps: fuzzy input (plus fuzzy output if defined) partitioning, rule induction and rule base simplification. Fuzzy partitioning will be discussed in Section II-B, a brief discussion on rule learning is done in Section II-C.
- Assessing the FIS behavior
  To help users to assess the rule representativity, an option that evaluates the *links between rules and examples* is available. A detailed cross-summary is accessible giving, for each rule, the samples that fire this rule above a given matching degree, and for each sample, the rules that are fired.
  Inference can be done manually or on the current data file, with evaluation criteria which take into account the numerical accuracy as well as the significance of data items regarding the FIS. Response surfaces are also available for an exploratory analysis of the system behavior.
- Sampling from a data file
  Random sampling from a data file is included in *FisPro* to facilitate the setting of robust learning and optimization procedures, and to avoid overfitting. Two possibilities are given for sample generation: to generate learning and test pairs, or blocks.
  In the first case, each pair includes a sample file and its complement, with a given relative sample file size. In the second case, the procedure splits the data file into K blocks for K-fold cross validation procedures.
  Sampling can be done with respect to the class proportions in a data file. A *FisPro* snapshot is shown in Figure 1.

Semantic constraints are implemented in *FisPro* learning techniques. This is to avoid a common drawback of many automatic learning methods, which, in the absence of such constraints, unfortunately may lead to uninterpretable systems.

B. Linguistic variable and fuzzy partitioning

The readability of fuzzy partitioning is a pre-requisite condition to build an interpretable rule base [17]. The necessary conditions for interpretable fuzzy partitions have been studied by several authors [21], [6], [13]. Let us recall the main points:

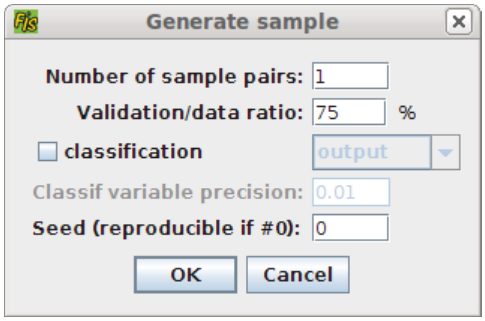

Fig. 1. The sample generation window in *FisPro*

- Distinguishability: Semantic integrity requires that each of the membership functions represents a distinct linguistic concept.
- A justifiable number of fuzzy sets.
- Coverage: Each data point, x, should belong significantly, $\mu(x) > \epsilon$, at least to one fuzzy set. ϵ is called the coverage level.
- Normalization: All the fuzzy sets should be normal.
- Overlapping: All the fuzzy sets should significantly overlap.

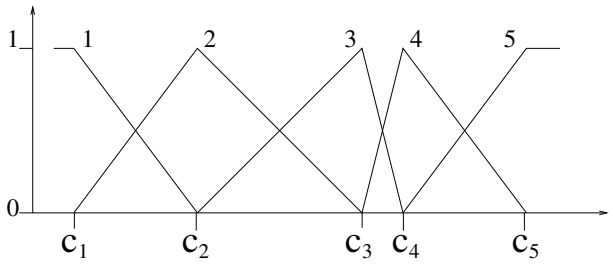

Fig. 2. A five linguistic strong fuzzy partition

These requirements are all fulfilled by the strong fuzzy partitions, illustrated in Figure 2, such as:

$$\begin{cases} \forall x \quad \sum_{f=1,2,\ldots,m} \mu_f(x) = 1 \\ \forall f \quad \exists\, x \quad \mu_f(x) = 1 \end{cases} \tag{1}$$

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 1 means that any point belongs at most to two fuzzy sets when the fuzzy sets are convex.

These partitions are fully defined by a set of m points, m being equal to the number of linguistic terms, denoted by c_i, $i = 1, \ldots, 5$, in Figure 2. Various techniques are available to identify these characteristic points from the data set.

In *FisPro*, fuzzy partitions can be manually built, by specifying the MF shape and bounds. Or else, three methods are available for automatic design . The first one generates regular grids without taking into account the data distribution. Another option is to use the classical *k-means* algorithm

in : IEEE International Conference on Fuzzy Systems, Brisbane, AUS,10-15 June 2012

with various numbers of groups. The last available method is called *hfp*, which stands for Hierarchical Fuzzy Partitioning. It is described in [15]. According to the generation method, the partitions are either independent one from the other (*k-means*) or the *k-1* term partition is derived from the k term one by fuzzy set merging (*hfp*).

In Section IV, the *k-means* algorithm will be used to generate the input fuzzy partitions of the FIS to be optimized.

C. Rule learning

In *FisPro*, all rule learning procedures produce conjunctive rules. In the study presented in Section IV, conjunctive FIS will be used in the optimization sequence, because it is applied to FIS learned from data.

All the rule learning methods implemented in *FisPro* only use the linguistic labels defined at the previous step, in order to guarantee fuzzy set sharing and thus interpretability. Two broad categories of rule learning methods can be distinguished: region based methods and prototype based ones. In the case of region based methods, the rule premises are chosen by splitting the input domains into regions and selecting the relevant regions according to a given criterion applied to the data set.

In the case of prototype based methods, the rule premises are initialized from data by analyzing each row and building the corresponding rule. The rule is kept if it satisfies a given criterion, and assigned an appropriate conclusion.

Well known methods from both categories are available in *FisPro*: Fuzzy Decision Trees [25] for the region based family, Wang and Mendel algorithm [24], Orthogonal least squares (OLS) [4], [5], [23], [18] for the prototype based family. When necessary, they have been revisited to ensure interpretable results. The key idea, which is valid for all these revisited methods, is the use of predefined strong fuzzy partitions for the rule generation, instead for instance of the data based Gaussian membership functions used in the original OLS.

A simple rule generation method, called FPA which stands for *Fast Prototyping Algorithm*, is also available. Despite its name, it belongs to the region based family.

Independently of the way the FIS was initially designed, all of its parameters can be optimized. In the following, two rule generation methods will be used to demonstate the utility of optimization procedures: FPA and OLS. More details will be given in Section IV.

III. The optimization procedure

The optimization algorithm, proposed by Glorennec [12], is based upon the work by Solis and Wets [22]. It is summarized in Algorithm 1.

The vector of parameters is denoted by $X(.)$ and the system error by $E(.)$.

The algorithm is a randomised hill-climber with an adaptive step size. Each step starts at a current vector $X(.)$. A deviate $G^{(k)}$ is chosen from a normal distribution whose standard deviation is function of the $Nmag$ parameter. This is a simple method that can adapt its step size very quickly,

Algorithm 1: The Solis and Wets algorithm

input : Initial vector $X^{(0)}$, noise magnitude $Nmag$
output: The optimized vector, $X = X^{(Max)}$

1 Initialization: $k = 0$, $M^{(0)} = 0$.
2 **while** $k \leq Max$ **do**
3 Generate a Gaussian vector $G^{(k)}$,
4 with mean $M^{(k)}$ and noise magnitude $Nmag$
5 **if** $E(X^{(k)} + G^{(k)}) < E(X^{(k)})$ **then**
6 $X^{(k+1)} = X^{(k)} + G^{(k)}$
7 $M^{(k+1)} = 0.2M^{(k)} + 0.4G^{(k)}$
8 **else if** $E(X^{(k)} - G^{(k)}) < E(X^{(k)})$ **then**
9 $X^{(k+1)} = X^{(k)} - G^{(k)}$
10 $M^{(k+1)} = M^{(k)} - 0.4G^{(k)}$
11 **else**
12 $X^{(k+1)} = X^{(k)}$
13 $M^{(k+1)} = 0.5M^{(k)}$
14 **end**
15 $k = k + 1$
16 **end**

because it differs from pure random search algorithms in the following way. The *good* directions in the research space are memorized through the M vector (lines 7 and 10). This biases the search process. In case of failure, the bias is decreased in an exponential way (line 13).

The coefficient values 0.4, 0.2 and 0.5 are retained from Solis and Wets' experimental results in [22]. In the following experiments, 0.005 is chosen for $Nmag$.

Solis and Wets established that convergence to a region surrounding the optimum is inevitable under some mild conditions (see [22] for details). As all random optimization methods, it does not require the gradient of the problem to be optimized and can therefore be used on functions that are not continuous or differentiable.

All of the FIS parameters: input or output partitions, rule conclusions, can be optimized by the process.

A. Control parameters and constraints

Several definitions are given below to introduce the *Loss of Coverage* control parameter.

Definition 3.1: Data row items are labeled active or inactive for a given rule base. A row item is *active* if its maximum matching degree over all the rules is greater than a user-defined threshold, denoted by *blank threshold*, *inactive* otherwise.

Definition 3.2: The *coverage index value* for a given FIS is $CI^k = \frac{A}{N}$, where A is the number of active rows, N the file size, and k the iteration number. CI^0 stands for the initial coverage index value.

Definition 3.3: The *Loss of coverage* for the kth iteration is $\frac{CI^0 - CI^k}{CI^0}$.

The *Loss of coverage* parameter is a control parameter, used during the optimization procedure to reject fuzzy

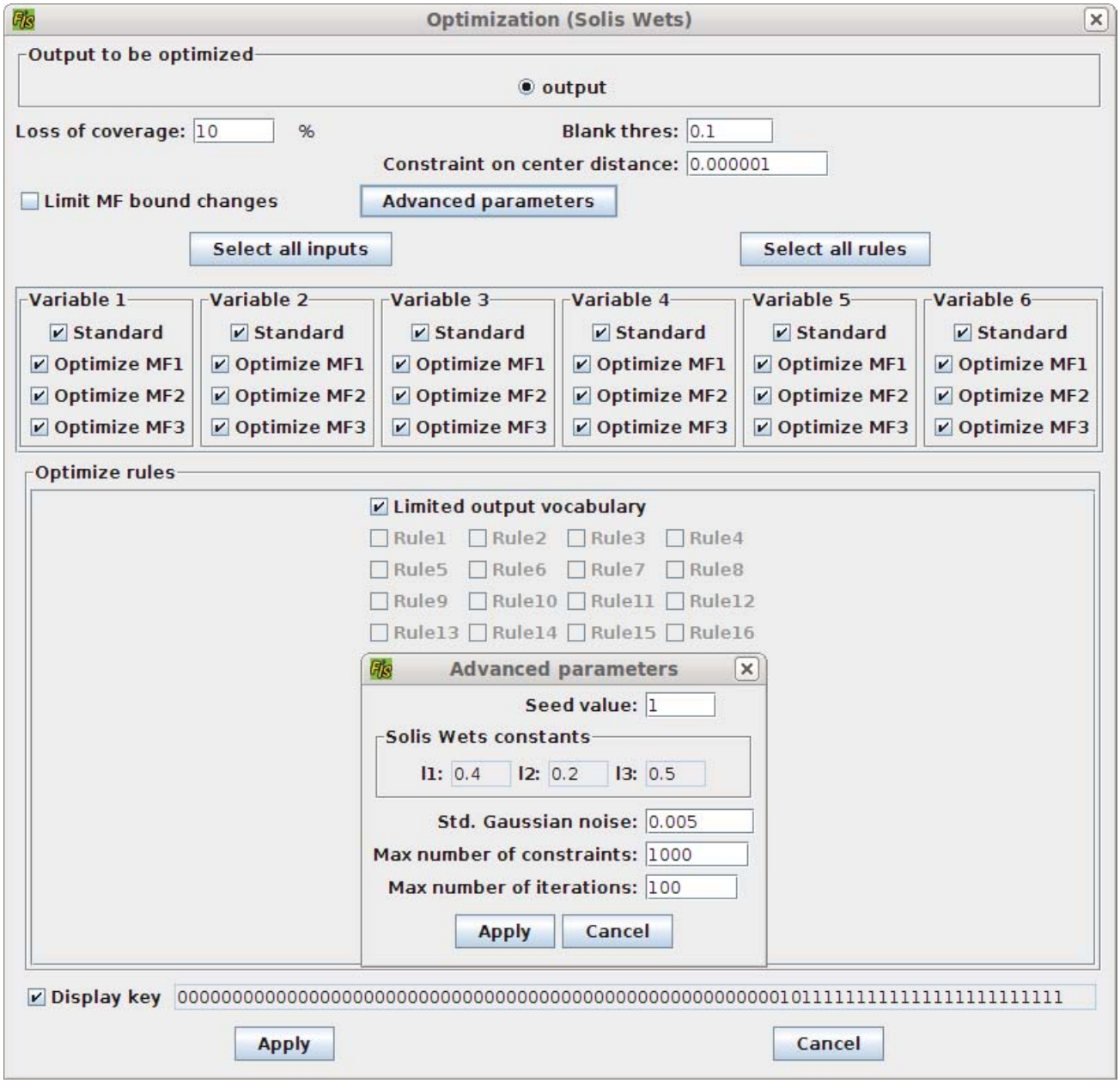

Fig. 3. The custom optimization window in *FisPro*

partition modifications that would degrade too much the FIS representativeness. A decreased coverage may happen because of an incomplete rule base.

Constraints can easily be imposed to guarantee semantic properties. When optimizing MF parameters, all solutions that do not lead to strong fuzzy partitions may be rejected by being considered as failures. In the same way, it is also possible to impose a minimum distance between characteristic points.

B. Customizable procedure

FisPro Learning menu includes two optimization options based on the Solis and Wets' algorithm: either a very general customizable procedure or a guided sequence. Figure 3 corresponds to the former one. It allows for individual selection of elements to be optimized together, either variables, output (if fuzzy), MFs or/and rules. The advanced parameter popup can be used to set algorithm parameters, like the seed value for the random generator, if one wants repeatable trials.

However, because the space exploration is biased by the M vector, it is better not to optimize all of the parameters

in a single pass. Otherwise some components may behave in opposite ways and penalize the search.

C. Guided sequence

Therefore it can be more efficient to design an optimization sequence. This sequence is composed of several steps, each of them optimizing a set of interrelated parameters. How to define these sets? All rule conclusions have to be optimized in the same process because a given sample is likely to fire several rules. Similarly, fuzzy terms in an input fuzzy partition come into mutual relationship and must be optimized together. As a strong fuzzy partition is uniquely defined by as many fuzzy set centers as linguistic terms, these center values constitute the vector to be optimized.

The Solis & Wets algorithm does not guarantee a global optimum. Consequently, the order in which the various steps are done may change the optimization results. The optimization procedure can be re-iterated in order to improve the results. In the following, each iteration is called a loop.

The optimization procedure proposed in *FisPro* is summarized below:

1) Call *Fistree*, the *FisPro* fuzzy decision tree, to sort the input variables by order of importance. The most influential variables come first in the tree hierarchy,
2) Optimize each of the selected input variables in turn according to the sort order,
3) If the output is fuzzy, optimize the output partition,
4) Optimize the rule conclusions.

Note that the procedure does not require the variables to be independent.

Figure 4 is a snapshot of the guided optimization window in *FisPro*. User-defined parameters include the *blank threshold* and the allowed loss of coverage, as well as the number of loops or iterations. Advanced parameters allow users to change the noise magnitude default value and to set the random generator seed. The default number of loops is set to 2.

As desired, the optimization procedure does not modify the system structure: the number of linguistic terms remains the same for each variable, no input variable is added nor removed in any rule.

This procedure also maintains the system interpretability. Constraints are imposed to ensure the new partitions are strong fuzzy partitions, and to keep the relative order of linguistic terms within the partition: the kth fuzzy set center value cannot be higher than the *(k+1)th* one.

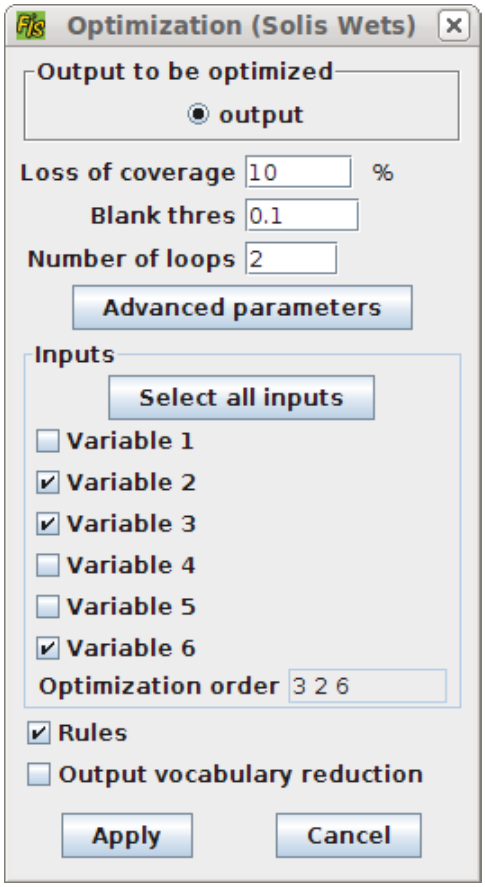

Fig. 4. The guided optimization window in *FisPro*

IV. Experiments

This section presents, compares and discusses results obtained on two well known cases chosen in the UCI repository [11]. The data sets are the following ones:

- *Cpu-performance* (209 samples):
  Published by Ein-Dor and Feldmesser [9], 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.

For the experiments, a repeated random sub-sampling cross validation method is used on each dataset. Ten pairs of learning and test subsets are generated by random sampling from the data set. For each pair, the learning subset size is 75% of the data set size, and the test subset is the complement of the learning subset. The noise magnitude for Gaussian vector generation is 0.005. The tolerated loss of coverage is 10%.

The error index is the Mean Absolute Error (MAE) defined in Equation 2, y_i and $\widehat{y_i}$ being respectively the observed and inferred output for the ith example.

$$MAE = \frac{1}{A}\sum_{i=1}^{A} |\widehat{y_i} - y_i| \qquad (2)$$

The index only takes into account *active* row items (see Definition 3.1), A being the number of *active* items.

Tests have been carried out in the range $0.01 - 0.2$ to evaluate the sensitivity to the *blank threshold* defined in

Section III and results proved not to be sensitive to this threshold. In the following tables, this value is set to 0.1.

A. Two rule induction methods

In order to illustrate its generic behavior, the optimization process has been tested with two rule bases generated with two different induction algorithms. The first one is the *FisPro* implementation of the *fuzzy Orthogonal Least Squares*, denoted by OLS, and the second one is called FPA which stands for *Fast Prototyping Algorithm*. Some elements are given below for each algorithm.

1) OLS: the algorithm is well known and thoroughly described in [7]. It is a statistical based algorithm restricted to regression models, and it yields uncorrelated rules which explain a new part of variance. The higher the sample size, the better the results.

2) FPA: the algorithm is simple and efficient for summarizing data sets. It acts in two steps: first, all of the rules corresponding to the input combinations are generated, and second their conclusions are initialized according to the data values.
In the regression case, this is achieved by Equation 3.

$$C_r = \frac{\sum\limits_{i \in E_r} \mu_r(x_i) * y_i}{\sum\limits_{i \in E_r} \mu_r(x_i)} \tag{3}$$

$\mu_r(x_i)$ is the matching degree of the ith example for the rth rule. E_r is a subset of examples chosen according to their matching degree to the rule. Its size (cardinality) is user-defined. If there are not enough items which fire the rth rule with a degree higher than the threshold, the rule is removed. Both parameters, threshold and cardinality, control the number of rules as well as their generalization ability. A snapshot of *FisPro* rule generation using FPA is shown in Figure 5.

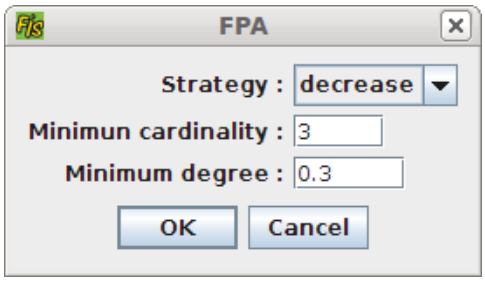

Fig. 5. The FPA rule generation window in *FisPro*

Strategy for the choice of the E_r subset

The selection of E_r elements can be done in two different ways, depending of the chosen strategy.

The first strategy is called *decrease*. It retains the examples which most activate the rule. The user can specify two parameters : the cardinality threshold *CardMin*, and the *MatchMin* matching degree threshold. If the number of examples matching the rule to a degree $\geq MatchMin$ is lower than *CardMin*, the required matching degree is decreased according to a given step. The default step value is equal to *0.1*, and the default matching degree is set to 0.7 in *FisPro*. The decrease procedure stops as soon as the required cardinality is reached, or when the required matching degree goes below the *MatchMin* value.

This strategy privileges the rule prototypes. It is assumed that the examples with a lower matching degree will be dealt with by means of interpolating during the inference procedure.

The other strategy is called *minimum*. it retains all the examples whose matching degree for the rule is greater than the $MatchMin$ threshold.

The FPA algorithm can manage classification as well as regression cases. Once the input space regions are defined by rule premises, only the samples which best match a rule premise are used to set the rule conclusion. The algorithm is faster than OLS, but less accurate when large data sets are available.

B. Initial system

Obviously, optimization results highly depend upon the initial system. As it is easy to improve a *poor* system, it is important to start from an accurate initial system, in order to illustrate the procedure efficiency.

- For each data set, the input fuzzy partitions are generated from data with the *k-means* algorithm, with $k = 3$ fuzzy sets per partition.
  The output is crisp, e.g. we have a *Sugeno-zero-order FIS*.
  The initial rule base is built using the OLS algorithm. The number of rules is selected to get an accuracy, measured by MAE (see Equation 2), averaged over the ten test sets, comparable to the one published in [7], and given in Table 1.

TABLE I
INITIAL OLS FIS FEATURES

	MAE	# rules
Auto-mpg	2.02	54
Cpu	28.77	9

- For each dataset, the same fuzzy partitions are used for FPA and OLS, and the FPA algorithm is run, with its parameters tuned to yield a FIS with the number of rules given in Table I.
- The fuzzy decision tree procedure is run to rank variables, by decreasing order of importance. The results are:
  - Auto-mpg: 1 6 4 5
  - Cpu: 3 6 2
- The optimization procedure is run on each of the four FIS, as described in Section III.

Table II summarizes the results. The optimization algorithm improves the performance of both systems for both data

in : IEEE International Conference on Fuzzy Systems, Brisbane, AUS,10-15 June 2012

sets. However, the FPA-based optimized FIS remains less accurate than the initial OLS-based FIS. The optimization procedure does not erase the differences between the rule generation methods.

TABLE II
OPTIMIZATION RESULTS: MAE AND RELATIVE GAIN AVERAGED OVER THE TEST SETS

		Ols	Fpa
Auto-mpg	Initial	2.02	2.22
	Optimized	1.96	2.14
	Gain (%)	3.0	3.6
Cpu	Initial	28.77	31.1
	Optimized	27.47	30.49
	Gain (%)	4.5	2.0

C. Final FIS selection

The cross validation procedure yields not a single FIS, but ten different ones, which share the same structure: same number and type of fuzzy sets in the fuzzy partitions, same number of rules, identical rule premises.

It seems a good policy to present the users with a unique FIS, resulting of the combination of all optimized ones. Therefore, we compute a *median FIS*, where the various optimized parameters are replaced by their median value, which is a more robust statistic than the mean.

The averaged results of the *median FIS* over the ten test samples are given in Table III. On average, the median FIS has a better accuracy than the individually optimized FIS with their corresponding test samples. As the FIS parameters are optimized according to the learning sample, the performance may decrease when assessed using the test set. This phenomenon happens as shown in Table IV.

TABLE III
OPTIMIZED MEDIAN FIS RESULTS: MAE AND RELATIVE GAIN AVERAGED OVER THE TEST SETS

		Ols	Fpa
Auto-mpg	MAE	1.91	2.01
	Gain (%)	5.4	9.5
Cpu	MAE	27.05	29.83
	Gain (%)	6.0	4.1

TABLE IV
NUMBER OF SAMPLES WITH POORER PERFORMANCE

		Ols	Fpa
Auto-mpg	Optim	2	3
	Median	0	0
Cpu	Optim	4	4
	Median	2	3

Thanks to the robustness due to the median statistics, the number of samples affected by this side effect decreases. Moreover, further investigation showed than the magnitude of the degradation is reduced, e.g. for two of the three samples of CPU and FPA with poorer accuracy, the accuracy degradation becomes lower than 1 %.

D. Discussion

The optimization procedure described above yields interpretable FIS, with the same structure as the initial one, making the comparison easy. Let us give some details about the FIS built using the OLS method for Cpu data. In both tables V and VI, I stands for initial, while F corresponds to final, i.e. the *median FIS*.

Table V compares the fuzzy partition characteristic points (defined in Section II-B) before and after optimization.

TABLE V
CPU OLS FIS: INITIAL (I) AND FINAL (F) FUZZY SET CHARACTERISTIC POINTS

	C_1		C_2		C_3	
	I	F	I	F	I	F
V2	2292	2634	14484	14259	31310	31379
V3	10404	7789	27254	27838	63110	63168
V6	12021	6916	65833	64972	144000	144000

Table VI shows the evolution of the rule conclusions. In this table, the rule premise is defined by the MF numbers of input variables.

TABLE VI
CPU OLS FIS: INITIAL (I) AND FINAL (F) RULE BASES

Premise					I	F
1	3	2	2	3	1033	1023
3	3	2	2	2	1144	1140
2	2	1	2	1	339	322
1	2	1	1	1	146	133
2	3	1	2	1	658	639
1	1	1	1	1	37	37
2	2	2	2	1	332	314
1	2	2	3	2	274	272
2	2	3	2	1	395	396

The final FIS is, as expected, not very different from the initial one and as much interpretable. However the slight modifications in the FIS parameters resulting from the optimization procedure systematically improve the performance, even though the initial FIS already had a good accuracy.
A snapshot of the *FisPro* inference window is shown in Figure 6.

V. CONCLUSION

This work has presented a flexible optimization procedure for FIS parameters, input or output MF parameters and rule conclusions. The guiding criteria include accuracy and coverage. The example developed in the present paper is based upon a classic random optimization technique, the Solis and Wets algorithm, but the principles of the guided optimization sequence remain valid whatever the optimization algorithm.

The Solis and Wets algorithm is a mono-agent evolutionist method. Its complexity, either in time or space, is reduced compared to genetic algorithms. There is no need to maintain a population because there are no genetic operations (neither crossover nor mutation), and the time convergence to the

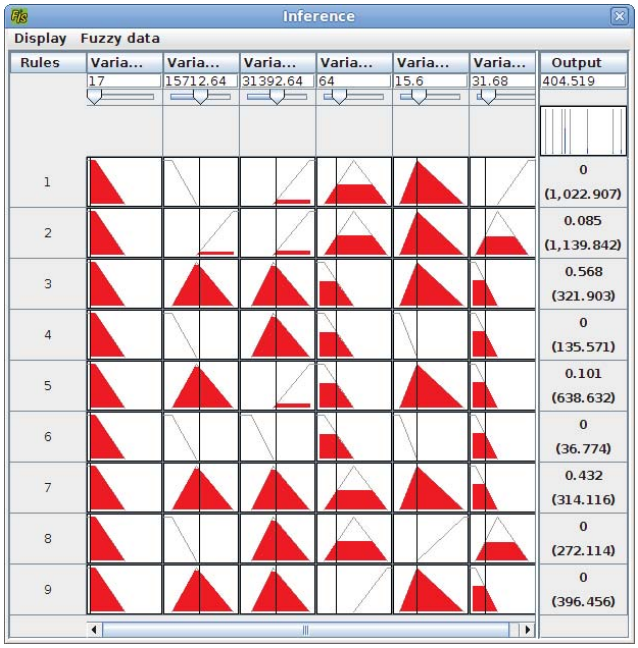

Fig. 6. The inference window in *FisPro*

local optimum is short (the default number of iterations is set to 100 in *FisPro*) because the *good* search directions are memorized. As interpretability constraints are imposed to the algorithm, the system accuracy can be improved without loosing the semantics attached to the partitions or rules.

The optimization algorithm has a few parameters only, which makes it easy to use. The software implementation in *FisPro* makes it available in two versions, an entirely customizable version and a guided sequential one.

The proposed sequence processes, at a given step, a set of interrelated parameters, such as the MF bounds of a single variable.

Applied to two well known data sets from the UCI repository, the overall procedure proved to be robust and efficient, independently from the initial rule base. For each of the two data sets, two systems have been built using different rule generation methods. Robustness has been checked using a cross-validation procedure, and the final system parameters are the median values of the cross validation runs. In all cases, the optimization leads to a significant improvement of accuracy preserving system interpretability.

Through the *FisPro*[2] implementation, the optimization procedure is freely available to users by the means of either a user-friendly interface or batch scripts.

REFERENCES

[1] R. Alcala, P. Ducange, F. Herrera, B. Lazzerini, and F. Marcelloni, "A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems," *IEEE Transactions on Fuzzy Systems*, vol. 17(5), pp. 1106–1122, 2009.

[2]http://www.inra.fr/Internet/Departements/MIA/M/fispro

[2] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, "Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: an overview," in *Interpretability Issues in Fuzzy Modeling*. Springer, 2003, pp. 3–22.

[3] L. Chen, C. Chen, and W. Pedrycz, "A gradient-descent-based approach for transparent linguistic interface generation in fuzzy models," *IEEE Transactions on Systems, Man and Cybernetics, part B: Cybernetics*, vol. 40(5), pp. 1219–1230, 2010.

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

[5] 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*, vol. 2 (No 2), pp. 302–309, March 1991.

[6] J. V. de Oliveira, "Semantic constraints for membership functions optimization," *IEEE Transactions on Systems, Man and Cybernetics. Part A*, vol. 29, no. 1, pp. 128–138, 1999.

[7] S. Destercke, S. Guillaume, and B. Charnomordic, "Building an interpretable fuzzy rule base from data using orthogonal least squares-application to a depollution problem," *Fuzzy Sets and Systems*, vol. 158, pp. 2078–2094, 2007.

[8] D. Dubois, H. Prade, and L. Ughetto, "A new perspective on reasoning with fuzzy rules," *International Journal of Intelligent Systems*, vol. 18, pp. 541–567, 2003.

[9] P. Ein-Dor, "Grosch's law re-revisited: Cpu power and the cost of computation," *Commun. ACM*, vol. 28, no. 2, pp. 142–151, 1985.

[10] A. G. Evsukoff, S. Galichet, B. S. de Lima, and N. F. Ebecken, "Design of interpretable fuzzy rule-based classifiers using spectral analysis with structure and parameters optimization," *Fuzzy sets and Systems*, vol. 160, pp. 857–881, April 2009.

[11] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[12] P.-Y. Glorennec, *Constrained optimization of FIS using an evolutionary method*, ser. Studies in Fuzziness and Soft Computing. Physica-Verlag, 1996, pp. 349–368.

[13] ——, *Algorithmes d'apprentissage pour systèmes d'inférence floue*. Editions Hermès, Paris, 1999.

[14] S. Guillaume, "Designing fuzzy inference systems from data: an interpretability-oriented review," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 426–443, 2001.

[15] S. Guillaume and B. Charnomordic, "Fuzzy models to deal with sensory data in food industry," *Journal of Donghua University*, vol. 21, no. 3, pp. 43–48, June 2004.

[16] ——, "Learning interpretable fuzzy inference systems with fispro," *International Journal of Information Sciences*, vol. 181, pp. 4409–4427, 2011.

[17] S. Guillaume and L. Magdalena, "Expert guided integration of induced knowledge into a fuzzy knowledge base," *Soft computing*, vol. 10, no. 9, pp. 773 – 784, 2006.

[18] 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.*, Orlando, Florida, June 1994, pp. 696–700.

[19] H. Jones, B. Charnomordic, D. Dubois, and S. Guillaume, "Practical inference with systems of gradual implicative rules," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 61–78, 2009.

[20] C.-F. Juang and P.-H. Chang, "Designing fuzzy-rule-based systems using continuous ant-colony optimization," *IEEE Transactions on Fuzzy Systems*, vol. 18(1), pp. 138–149, 2010.

[21] E. H. Ruspini, *Recent developments in fuzzy clustering.* Pergamon Press, New York, 1982, pp. 133–147.

[22] F. J. Solis and R. J. Wets, "Minimization by random search techniques," *Mathematics of Operation Research*, vol. 6, pp. 19–30, 1981.

[23] L.-X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," *IEEE Transactions on Neural Networks*, vol. 3, pp. 807–814, 1992.

[24] ——, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22 (6), pp. 1414–1427, November/December 1992.

[25] R. Weber, "Fuzzy-id3: A class of methods for automatic knowledge acquisition," in *2nd International conference on fuzzy logic and neural networks*, 1992, pp. 265–268.

[26] J. Weisbrod, "A new approach to fuzzy reasoning," *Soft Computing*, vol. 2, pp. 89–99, 1998.