Learning interpretable fuzzy inference systems with FisPro

S. Guillaume, Brigitte Charnomordic

Learning interpretable Fuzzy Inference Systems with FisPro

Serge Guillaume[a], Brigitte Charnomordic[b]

[a]*Cemagref, UMR ITAP, BP 5095, 34196 Montpellier, France*
[b]*INRA/SupAgro, UMR MISTEA, 34060 Montpellier, France*

Abstract

Fuzzy inference systems (FIS) are likely to play a significant part in system modeling, provided that they remain interpretable following learning from data. The aim of this paper is to set up some guidelines for interpretable FIS learning, based on practical experience with fuzzy modeling in various fields. An open source software system called *FisPro* has been specifically designed to provide generic tools for interpretable FIS design and learning. It can then be extended with the addition of new contributions. This work presents a global approach to design data-driven FIS that satisfy certain interpretability and accuracy criteria. It includes fuzzy partition generation, rule learning, input space reduction and rule base simplification. The *FisPro* implementation is discussed and illustrated through several detailed case studies.

Keywords: Fuzzy rule bases, interpretability, modeling, rule induction, fuzzy partitioning

1. Introduction

Fuzzy Inference Systems (FIS) have been shown to perform well in input-output mapping identification because they satisfy the universal approximation property and can be designed from expertise or data driven methods.

Furthermore, their inference engine implements approximate reasoning, which provides a good framework for representing and manipulating a wide body of linguistically expressed information, as pointed out in a recent survey [41]. The im-

---

Email addresses: serge.guillaume@montpellier.cemagref.fr (Serge Guillaume), bch@supagro.inra.fr (Brigitte Charnomordic)

plementation is done in the form of a set of potentially understandable IF-THEN rules.

Nevertheless, these advantages are not sufficient for FIS to be truly useful in modeling real world systems.

Indeed, generalization ability and interpretability are fundamental prerequisites for a model to be used. By interpretability, we mean transparency and intelligibility.

Although no one can question the need for automatic learning from data, data-driven approaches still suffer from many drawbacks, as their performance strictly depends both on the volume and on the quality of the data.

Even if FIS have the potential ability to express the system behavior in a linguistic way, the use of a fuzzy formalism is not sufficient to ensure the transparency and intelligibility of a fuzzy rule base. As shown in [20], several conditions are required for interpretability, concerning the fuzzy partitioning, the consistency and number of rules of the rule base, and the presence of incomplete rules.

As the interpretability constraints may conflict with the objective of automatic learning methods to minimize numerical error, several works have proposed a tradeoff between interpretability and accuracy [10].

In this paper, our goal is to propose tools that can facilitate a modeling approach using supervised learning and data-driven FIS. The *FisPro* toolbox that we present is a step in that direction. It stands out among fuzzy software products because of the care taken to ensure the interpretability of the fuzzy systems that are automatically learnt from data, at all steps of the design.

The original contribution of the paper is that it provides guidelines for interpretable FIS learning based on practical experience with fuzzy modeling, acquired during several years of work in this field. The modeling was supported by the *FisPro* software.

The structure of the paper is as follows.

First, in Section 2, we investigate the learning steps required for interpretable FIS design: the objectives and evaluation criteria, fuzzy partitioning methods, and rule induction techniques. Special attention is paid to variable granularity and selection.

Then, in Section 3, we present the *FisPro* open source software features that are particularly useful for learning. Section 4 is dedicated to three case studies, each illustrating some particular points of our approach. Finally, we offer some concluding remarks and topics for further discussion.

2

2. Learning steps for interpretable FIS

Since it was first introduced by Takagi and Sugeno [36], FIS design using data-driven approaches has been a favorite topic of fuzzy logic researchers. Data mining methods are available in large quantities, either originating from the field of Statistics or inspired by Artificial Intelligence learning techniques, and they are easily transferable to FIS with a few changes to adapt them to the particular structure of FIS.

Following Sugeno's early work, significant efforts have been dedicated to the development of data-driven FIS design, focusing on numerical accuracy but weakening the key innovation and competitive advantage of fuzzy logic. An interpretability-oriented review of fuzzy inference system design from data can be found in [36]. It analyzes the main methods of automatic rule generation and structure optimization from that perspective.

At the beginning of the twenty-first century, a new research trend appeared of a return to genuine principles of fuzzy logic. Renewed attention was paid to system interpretability while taking advantage of the specific learning capabilities of these methods, see [10].

Even so, not all researchers attribute the same meaning to the concepts of *interpretability* and *transparency* for FIS. A detailed study can be found in [33].

Our purpose here is not to provide an in-depth discussion of these concepts. Instead, we shall try to define some general learning principles and steps for an interpretable FIS learning approach.

Both the fuzzy partition and the rule base are involved in the design of FIS, and three main factors emerge as the primary determinants of interpretability.

First, all fuzzy partitions should obey some semantic integrity constraints, so that each fuzzy set is associated with a non ambiguous meaning, and in order to obtain a complete coverage of the variable domain ranges.

Second, the number of rules should be small and the rule base consistent. The curse of dimensionality is a real problem for FIS learning from data, as a full set of complete rules quickly leads to a combinatorial explosion when the number of variables and of fuzzy sets rises. This is annoying as some rules may be generated that cover a very small part of the input space, are matched only by a few examples, and lack generality.

The third condition is specific to complex systems with a large number of input variables: rules must not systematically include all input variables, but only the important ones in the context of the rule. These rules are often called incomplete rules.

3

FIS learning methods are essentially rule generation techniques, and rule generation can be decomposed into two main steps: rule induction and rule base optimization. Variable selection and rule simplification are two important stages of the optimization process.They are usually referred to as structure optimization. Apart from structure optimization, a FIS has many parameters that can also be optimized, such as membership function (MF) parameters and rule conclusions. This is called parameter optimization. A thorough study has been done by various authors [26, 18], and the respective advantages and drawbacks of these methods are well known.

Many methods perform fuzzy partitioning, rule induction and optimization in the same procedure. Nevertheless, this approach has serious drawbacks.

It makes it difficult to reuse the fuzzy partitioning for other learning procedures. However, a fuzzy partition can be valuable by itself, as it represents a mapping between numerical values and symbolic concepts in the input space. Furthermore, the use of the same fuzzy partitions in several rule bases makes it easier to compare these bases.

For similar reasons, it is desirable for the optimization procedure to be distinct from the induction procedure: new data sets can be used for the optimization as they become available, and different criteria can be used depending on the context of use.

All of this leads us to propose a FIS learning approach decomposed into several steps: fuzzy input (plus fuzzy output if defined) partitioning, rule induction and simplification. Optimization will not be discussed in this work, although *FisPro* offers parameter optimization procedures based on evolutionist algorithms, which can be applied to any FIS. The *FisPro* implementation pays attention to preserving semantics, as do some other works. For instance, in [15], an index is proposed to maintain the semantic interpretability in a multi-objective evolutionary optimization algorithm. For the reasons given above, we concentrate our presentation on the other points.

The following sections present each learning step in detail, and the last one focuses on the choice of variable granularity and selection. As a preliminary to presenting the method, we give some learning objectives and evaluation criteria.

2.1. Objectives and evaluation criteria

Machine learning techniques cannot produce results better than what they find in the training data. The task of a supervised learner is to predict a value for any valid input object after having seen a number of training examples (i.e., pairs of inputs and target outputs). To achieve this, the learner has to generalize from the

4

presented data to unseen situations in a *reasonable* way. FIS partition and rule learning have the same objectives and constraints.

A reasonable level accuracy must be achieved while avoiding overfitting. Overfitting occurs when a model describes random errors or noise instead of the underlying relationship. Overfitting generally happens when a model is excessively complex, such as having too many degrees of freedom, in relation to the amount of available data.

A model that has been overfit will generally have poor predictive performance on a data set that is significantly different from the learning set, as it can exaggerate minor fluctuations in the data.

To limit this risk, some parameters will be made defined for the learning methods; in particular, the learning will often be guided by a minimum rule matching degree. Test sets will be used whenever possible.

The coverage index and the performance indices, which are different for regression and classification cases, will be used as evaluation indices to assess the prediction capabilities of a FIS for a given dataset.

Denote by (x_i, y_i) the ith row of the data set, where x_i is a multidimensional input vector and y_i the corresponding output.

2.1.1. Coverage index

Data rows are labeled *active* or *inactive* for a given rule base. A row is active if its maximum matching degree over all of the rules is greater than a user defined threshold, and otherwise it is inactive.

Following this definition, a *coverage index value* is calculated by applying the formula $CI = \dfrac{A}{N}$ where A is the number of active rows, and N is the file size. The *coverage index value* is a quality index that is complementary to the classical accuracy index.

2.1.2. Performance indices

In the following, the error index only considers the number of active items defined previously, denoted A.

For regression cases, the performance index available in *FisPro* is based on the square root of the mean squared error:

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

where $\widehat{y_i}$ is the inferred value.

It allows to compute another error index that is often found in the literature, called the Mean Absolute Error (MAE):

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

For classification cases, the error index used in *FisPro* is the sum of classification errors (MC):

$$MC = \sum_{i=1}^{A} (err_i | err_i = 1 \; if \; \widehat{C}_i \neq C_i, \; 0 \; otherwise) \qquad (3)$$

where C_i is the observed class and $\widehat{C}_i$ the inferred one.

Another classification index, not yet available in *FisPro*, that is differentiable so that it can be used in gradient-based approaches, is the squared classification error:

$$SCE = \frac{1}{A} \frac{1}{K} \sum_{i=1}^{A} \sum_{k=1}^{K} (\alpha_i^k - \mu_i^k)^2 \qquad (4)$$

K is the number of classes, and α_i^k is the activation of the kth class over the ith item, with $\mu_i^k = 1$ if the correct class for the ith item is k, and 0 otherwise.

2.2. *Linguistic variable and fuzzy partitioning*

Variable partitioning is the first step of FIS design. The necessary conditions for fuzzy partitions to be interpretable and to implement the linguistic variable concepts have been studied by several authors [35, 30, 18, 28]. The main points are distinguishability, a justifiable number of fuzzy sets, normalization, sufficient overlapping and coverage: each data point, x, should significantly belong ($\mu(x) > \epsilon$, ϵ is called the coverage level) to at least one fuzzy set.

Even if other membership function shapes are available and if the fuzzy partitions can be freely adjusted, the *FisPro* automatic procedures only generate strong input fuzzy partitions (see Figure 1 for an example, with semi-trapezoidal shapes at the edges and either triangular or trapezoidal MFs elsewhere).

The overlap between two neighboring MFs is:

$$\forall x, \sum_{f=1}^{c} \mu_f(x) = 1 \text{ and } \forall f, \; \exists \, x \; such \; as \; \mu_f(x) = 1$$
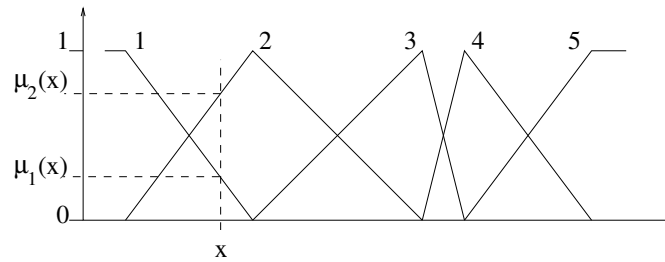
6

Figure 1: Example of a strong fuzzy partition

where c is the number of fuzzy sets in the partition and $\mu_f(x)$ is the membership degree of x in the fth fuzzy set.

Using strong fuzzy partitions ensures semantic integrity, and compared to partitions made up of unbounded membership functions, such as the widely used Gaussian partitions, they also lead to more robust systems because the number of simultaneously fired rules is limited [12].

What should be the number of linguistic terms in the partition? The correct answer is the one required for reasoning with rules. However, in an automatic learning procedure, this number is unknown, so the question becomes: what is the most suitable number of terms according to the data?

The *FisPro* approach is to generate a collection of fuzzy partitions of various sizes from two to a user-defined maximum value, e.g., seven, allowing the decision to be made later, based upon indices or an objective function. Three methods are available. The first one generates regular grids without considering the data distribution. Another option is to use the *k-means* algorithm with different numbers of groups. The last proposed method is called *hfp*, which stands for Hierarchical Fuzzy Partitioning. It is described in [22]. According to the generation method, the partitions are either independent of each other (*k-means*), or the *k-1* term partition is derived from the k term partition by fuzzy set merging (*hfp*).

Several indices have been defined to characterize fuzzy partitions. The following indices are implemented in the *GUAJE* open source software [1] described in [3, 4]. *GUAJE* is an upgraded version of *KBCT* (Knowledge Base Configuration Tool). Let N be the data set size and $\mu_i(k)$ the membership degree of the ith item in group k; we then have:

7

$$PC = \frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{c} \mu_i^2(k)$$

$$PE = -\frac{1}{N} \left\{ \sum_{k=1}^{N} \sum_{i=1}^{c} [\mu_i(k) \log_a(\mu_i(k))] \right\}$$

$$CI = \frac{1}{N} \sum_{k=1}^{N} \max_i \mu_i(k) - \frac{2}{c(c-1)} \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} \frac{1}{N} \sum_{k=1}^{N} \min(\mu_i(k), \mu_j(k))$$

The partition coefficient (PC) and the partition entropy (PE) were proposed by Bezdek[6] in 1981; the Chen index (CI) is more recent[11].

The three indices described above can be applied to any partition, independently of the derivation method. According to these criteria a good partition should minimize the entropy and maximize the partition coefficient and the Chen index.

As an illustration, consider data distributions for two variables of the *auto-mpg* and *wine* data sets, which are described at the beginning of Section 4. The three indices are computed for partitions of two to seven terms generated from these data by the three available methods. Two fifteen class histograms are plotted in Figure 2, with the x-axis representing the input values and the y-axis the number of data items. The histogram on the left corresponds to the fifth variable of the *auto-mpg* data set, and the one on the right to the tenth variable of the *wine* data set. These two distributions are dissimilar in shape: a quasi Gaussian distribution for the fifth variable, and a multimodal skewed one for the tenth variable.

The index behavior is also quite different between these two variables.

Tables 1 and 2 show the best partition size for the corresponding variable according to each of the three indices. The index value is given in parentheses following the partition size.

auto-v5	PC	PE	CI
km	2 (0.80)	2 (0.30)	5 (0.75)
hfp	7 (0.63)	7 (0.54)	7 (0.70)
reg	3 (0.69)	3 (0.47)	5 (0.75)

Table 1: Optimal partition size and index value for auto-v5 according to the indices
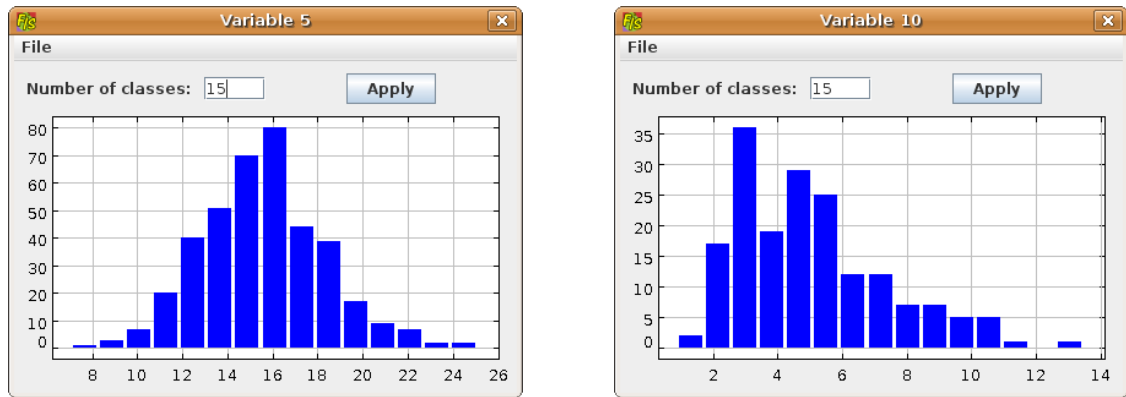
Figure 2: Data distribution for v5 from auto-mpg (left) and v10 from wine (right) data sets: v5 is the car weight and v10 is the wine color intensity.

wine-v10	PC	PE	CI
km	2 (0.82)	2 (0.26)	4 (0.80)
hfp	2 (0.86)	2 (0.21)	2 (0.80)
reg	7 (0.67)	7 (0.49)	7 (0.75)

Table 2: Optimal partition size and index value for wine-v10 according to the indices

For the *auto-v5* variable, the best partitions according to PC and PE are the *k-means* ones, while the CI index cannot differentiate between *k-means* and *regular grids*. For *wine-v10*, PC and PE agree that *hfp* is the best, and CI yields the same values for *k-means* and *hfp*. However the *hfp* partition size is smaller.

Recall that the indices cannot be compared with each other, the only valid comparison is between the behavior of a given index for several partitions.

This simple example shows that none of the proposed methods can be considered as uniformly better than the others in all cases. Moreover, even if the indices are useful for characterizing the fuzzy partitions, they are likely to differ in their results. One way to use them is to select one index and to follow its evolution throughout an optimization process.

2.3. Rule induction

There are many FIS rule learning methods, and our purpose in the present paper is not to provide a review of them all, but to recall the principles of a few techniques that yield interpretable FIS, which are implemented in *FisPro*. FIS

learning involves the fitting of many parameters, so we find it reasonable to consider only single output systems in the present study. Whatever their complexity and their origin (fuzzy clustering, statistical methods, machine learning or ad hoc data-driven ones specifically designed for fast fuzzy rule learning), rule learning methods may be classified into two broad categories: region based methods and prototype based ones. Indeed, once the input fuzzy partitions have been defined, the naive method of generating the complete set of rules corresponding to all fuzzy set combinations must be definitively rejected because it causes a very fast combinatorial explosion. The learning question then reduces to selecting the relevant rule premises and to assigning them appropriate conclusions.

In the case of the region based methods, the rule premises are chosen by splitting the input domains into regions and selecting the relevant regions by applying a given criterion to the data set.

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

We now present some methods that are available in *FisPro*. We discuss their advantages and drawbacks, and the parameters available for reaching our objectives: generalization, robustness, accuracy.

2.3.1. Notations

A fuzzy rule is defined as:

IF x^1 is A_r^1 and x^2 *is* A_r^2 ... and x^p is A_r^p THEN y *is* C^r

The rth rule matching degree for x is calculated as:

$$m_r(x) = \bigwedge_{j=1}^{p} \mu_{A_r^j}(x^j) \tag{5}$$

where $\wedge$ is the conjunction operator.

2.3.2. Region based rule learning methods

Two methods are discussed in this section, first a fast but efficient technique and then a more elaborate one.

- The Fast Prototyping Algorithm (*FPA*) [17] consists of generating the rules that, out of all possible combinations of antecedents, satisfy the following

10

criterion: the rule matching degree w is higher than a given threshold μ_t for more than a given number of data rows.

Two strategies are available to compute the conclusion for a particular rule: the first only retains the data rows that most activate the rule, the second one all data rows that match it above μ_t. The subset so defined is called D_r.

The conclusion C_r of the rth rule is computed from the selected data in the subset D_r, which are used as prototypes, whence the method name.

  – for classification cases, the conclusion is assigned the majority class,

  – for regression cases, it is equal to:

$$C_r = \frac{\sum\limits_{i \in D_r} m_r(x_i) * y_i}{\sum\limits_{i \in D_r} m_r(x_i)} \tag{6}$$

FPA gives a quick summary of the data set in the form of fuzzy rules that can be further tuned through optimization procedures on a test set. It is efficient for large data sets.

• Fuzzy Decision Trees (*FDT*), which are an extension of classical decision trees [9, 31], constitute a popular elaborate application of region based methods. The *FDT* proposed in *FisPro* are based on the algorithm presented in [40]. The *FisPro* implementation relies on a predefined fuzzy partition of the input variables, which is left untouched by the tree growing algorithm.

Starting from a root node including all data set items, the FDT uses a recursive procedure to split each node into M_j child nodes, where M_j is the number of fuzzy sets in the jth input variable partition selected for the split. For each node, the algorithm selects the variable that maximizes the gain according to a *discriminant* criterion:

$$G_n^j = I_n - \sum_{m=1}^{M_j} w_m I_{n,m} \tag{7}$$

where $I_{n,l}$ is the criterion for the node, created by the split onto the jth variable into M_j MFs, and corresponding to the mth MF.

w_m is the relative weight of the mth MF, $\mu_m(x_i^j)$ the membership of x_i^j in the mth MF.

11

Let $\mu_i^n = \mu_n(x_i)$ denote the membership of the ith example in the nth node, the child of the *(n-1)th* node. It is recursively defined as:

$$\begin{cases} \mu_i^0 = 1 \\ \mu_i^n = \mu_i^{n-1} \wedge \mu_m(x_i^j) \end{cases}$$

An interesting feature of *FDT* is their ability to adapt to different kind of outputs: continuous numerical outputs or discretized outputs that represent classes, by using a suitable splitting criterion.

   – If the output is a *class*, the *absolute gain* criterion is based on the definition of the fuzzy entropy for the nth node:

$$I_n = - \sum_{k=1}^{K} p_k log(p_k) \tag{8}$$

where k is an output class, K is the total number of classes and p_k is the fuzzy standardized kth class ratio at node n. It is calculated as follows.

Let $\mu_k(y_i)$ denote the membership of the output value y_i in the MF associated to the output class k.

Then p_k is given by Equation 9:

$$p_k = \frac{\sum_{i=1}^{A} \mu_k(y_i) \wedge \mu_i^n}{\sum_{c=1}^{K} \sum_{i=1}^{A} \mu_c(y_i) \wedge \mu_i^n} \tag{9}$$

where A is the number of active rows defined in section 2.1.1.

A *relative entropy gain* criterion is also available to avoid the bias against variables with an unequal distribution of examples (not classes) between the various MFs.

   – If the output is *continuous*, the criterion is based on the output variance of all examples attracted by the node:

$$I_n = \sum_{i=1}^{N} \mu_i^n (y_i - \bar{y}_i)^2 \tag{10}$$

FDT have several assets. They provide a flexible, compact and interpretable representation. They also determine a sorting order for influential variables, with the most discriminating variables appearing near the root. Therefore, they are useful for selecting the more relevant variables prior to applying another learning technique.

Currently available parameters include the tree depth, the minimum membership μ_t for a data row to be considered in the process, and the accuracy loss allowed for pruning a branch. A test set can be used for pruning, which may be different from the set used for tree learning.

A *tree equivalent FIS* is obtained by creating a fuzzy rule for each path leading to a terminal node (leaf). The assigned conclusion is based on the majority class or average value, computed over all examples attracted by the leaf. The *tree equivalent FIS* rule base is composed of incomplete rules, where only the influential variables appear in each rule premise.

Their main drawbacks are their sensitivity to small variations in the data set and the fast growth of the tree size, especially for continuous outputs. Therefore a good practice is to prune the tree by transforming a node into a leaf node, according to an accuracy criterion computed on a test set.

2.3.3. *Prototype based rule learning methods*

In contrast to region based learning methods, where it is easier to guarantee interpretability because of the use of predefined regions, this is not so easy for prototype based learning methods. Many works found in the literature use the Gaussian unbounded MF or assign different MF centers for each data row.

In *FisPro*, we constrain the prototype based learning techniques to use a predefined interpretable partitioning to overcome these difficulties. We now introduce two such techniques: the first one was especially designed for fuzzy systems and the second is inherited from statistical methods.

- Wang & Mendel [39] originally proposed the following procedure, called *WM*:
    1. Each variable of the input space is automatically divided into a user defined number of triangular membership fuzzy sets.
    2. One fuzzy rule is generated for each data row, in the form given in Section 2.3.1.
       The fuzzy sets A_i^j are those that maximize the matching degree of x_i^j for each input variable j from the ith row. The fuzzy set C^i is the one that maximizes the observed output matching degree.

3. A degree m_r is assigned to each rule according to Equation 5. In case two rules have identical premises, only the one with the higher degree is kept.
4. The output is computed by centroid defuzzification.

In the *FisPro* implementation, to comply with our interpretability requirements, the first step is removed and the fuzzy rules are generated for the preexisting fuzzy partitions.

Therefore the generated rules are not centered on the data set examples, and *WM* is no longer a purely prototype based method, as the fuzzy partitions delineate fuzzy regions. However, if we compare it with *FPA*, the main difference stems from the way the rule conclusions are initialized. With *FPA*, they are calculated using a subset of examples, whereas *WM* only considers a single item.

The *WM* procedure allows the rule base to be adaptive: new rules compete with existing ones. One drawback is the rough management of conflicts between rule conclusions, which are *resolved* by selecting the more representative one in the data set.

- Fuzzy Orthogonal Least squares (*OLS*) is an advanced example of a prototype based learning method. The technique is inspired by linear regression model fitting. Wang and Mendel [38] introduced the use of Fuzzy Basis Functions to map the input variables into a new linear space. In their original implementation, *OLS* are even more prototype oriented than *WM*, with each data row being used for straightforward rule initialization using a Gaussian MF centered on the corresponding data values.

Figure 3 illustrates a flowchart describing the two pass method used in the original *OLS* and the modifications introduced in the *FisPro* implementation to increase interpretability. All details can be found in [12].

The algorithm allows us to use a test set for the second pass that is different from the learning set used for the first pass. Two stopping criteria are available: the amount of explained variance and the number of rules.

Indeed a great advantage of fuzzy *OLS* rule learning is its ranking of the rules in the rule base in decreasing order of explained output variance.

A side effect is the easy detection of outliers, by checking whether the rules that come out first are matched by only a few data items. The sensitivity to outliers could be a drawback, if it is not carefully monitored.
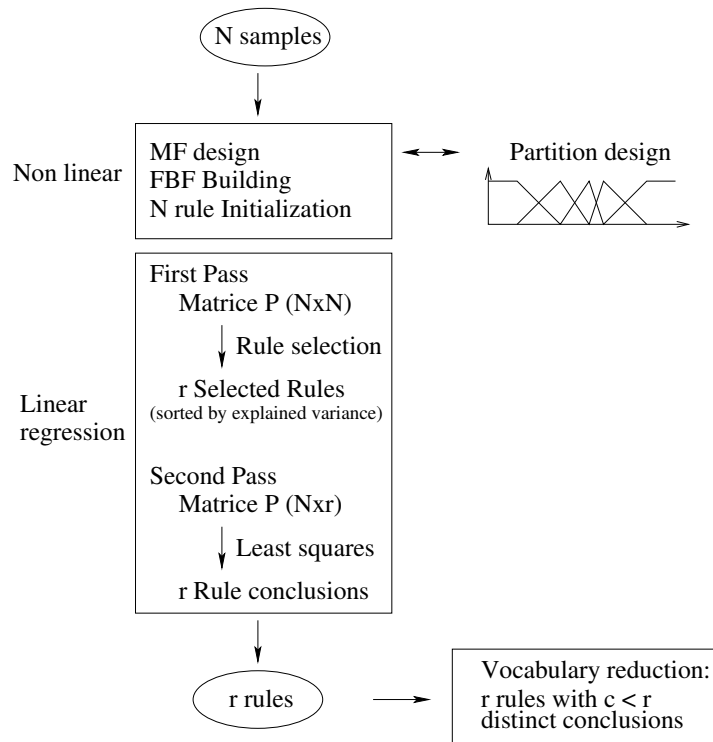
14

Figure 3: Flowchart for the modified *OLS* algorithm implemented in *FisPro*

Table 3 gives a summary of the various features and parameters for the four methods presented above. The method category and rule nature, complete or incomplete, is specified. The minimum matching degree μ is relevant to all methods, except *WM*, and refers to the minimum rule matching degree for a data item to be significant. It increases the method robustness. The parameter *#ro.*, which is the minimum number of data rows that must match the rule to keep it in the rule base, appears only in *FPA*. It could easily be introduced into *WM*, for the removal of rules that are weakly represented within the data set, or into *FDT*.

Other *specific* parameters mentioned previously for each method are indicated in the third parameter related column.

When the observed output is a continuous value, we have a regression case, denoted R. When the objective is to predict a class label of the observed output, we have a classification case, denoted C.

The FIS output type and the aptitude to model regression/classification cases are indicated in the last two columns.

For all methods except *OLS*, which is designed for crisp outputs only, the *FIS output type* is either *crisp*, corresponding to Sugeno FIS, or *fuzzy* for Mamdani FIS. Because of the way rule conclusions are assigned, *WM* is well suited to classification cases. Although classical decision trees are often used for classification purposes, fuzzy ones are of great interest also for modeling regression cases because of their interpolation capabilities. FDT were also found to be good rankers, as shown in [24].

Method	Category	Rules	Parameters			Test set	Output	R/C
			μ	#ro.	*Specific*			
FPA	region	comp.	x	x	strategy		crisp/fuzzy	R/C
FDT	region	incomp.	x		gain	x	crisp/fuzzy	R/C
WM	prototype	comp.					crisp/fuzzy	C/R
OLS	prototype	comp.	x		expl. var. #rules	x	crisp	R

Table 3: Main features and parameters of some rule learning methods - R denotes regression, and C denotes classification

2.4. Variable granularity and selection

Independently of the rule induction and partitioning methods, the selection of variables and the choice of the appropriate granularity are recurrent questions

in FIS design, and more generally in Machine Learning. The goal of this section is not to offer a general and thorough view, which can be found in [23], but to highlight original approaches implemented in *FisPro* to dealing with variable granularity and rule base simplification.

Recall that *FDT* includes a selection variable process, as mentioned in section 2.3.2 and it yields incomplete rules.

2.4.1. Refinement

The refinement procedure aims to determine the suitable number of terms for a given variable by exploiting the hierarchies of fuzzy partitions presented in section 2.2.

The key idea is to introduce as many variables, described by a sufficient number of fuzzy sets, as are necessary to get a good rule base. A *good* FIS represents a reasonable trade-off between complexity, determined by the number of rules, and accuracy, measured by the performance index, denoted *Perf* in the algorithms.

The refinement procedure is responsible for the selection of the variables or fuzzy sets to be introduced in the FIS. Let p be the number of input variables, $FP_j^{n_j}$ the fuzzy partition of the variable j of size n_j, where the fuzzy set centers are the coordinates given by the hierarchy, $FP_j^{n_j} = \{MF_j^{k/n_j}, k = 1, \ldots, n_j\}$, where MF_j^{k/n_j} refers to the kth membership function of the n_j-term fuzzy partition for the jth variable.

The initial FIS is the simplest one possible with a single rule (Algorithm 1, lines 1-2). The search loop (lines 5 to 14) builds up temporary fuzzy inference systems, each of which corresponds to adding to the initial FIS one fuzzy set in a given dimension. The dimension to retain is selected in lines 18-19. Following this selection, the FIS to be kept is built up. It will serve as a base to iterate the sequence (lines 3 to 23).

Thus, the result of the procedure is not a single FIS but a series $FIS_1, FIS_2, \ldots$ of increased complexity.

When necessary, the procedure calls a FIS generation algorithm, denoted as Algorithm 2, which is now detailed.

The rules are generated by combining the fuzzy sets of the $FP_j^{n_j}$ partitions for $j = 1, \ldots, p$, as described by Algorithm 2. The algorithm then removes the less influential rules and evaluates the rule conclusions.

$w^r(x_k)$ is the matching degree of example k with rule r. The condition stated in line 4, where CV_t is a given threshold, ensures that the rule is significantly fired by the training set examples.

Algorithm 1: Refinement procedure

---

1 Initialization: $iter = 1, \forall j \; n_j = 1$

2 CALL *FIS Generation* (Algorithm 2)

3 **while** $iter \leq iter_{max}$ **do**

4 $\quad$ Store system as base system

5 $\quad$ **for** $1 \leq j \leq p$ **do**

6 $\quad\quad$ **if** $n_j = n_j^{max}$ **then**

7 $\quad\quad\quad$ next j

8 $\quad\quad$ // partition size limit for input j

9 $\quad\quad$ $n_j = n_j + 1$

10 $\quad\quad$ CALL *FIS Generation* (Algorithm 2)

11 $\quad\quad$ $Perf_j = Perf$

12 $\quad\quad$ $n_j = n_j$ - 1

13 $\quad\quad$ Restore base system

14 $\quad$ **end**

15 $\quad$ **if** $\forall j \; n_j = n_j^{max}$ **then**

16 $\quad\quad$ exit // no more inputs to refine

17 $\quad$ // Select input to refine

18 $\quad$ s = argmin $\{Perf_j, \; j = 1, \ldots, p, \quad n_j < n_j^{max}\}$

19 $\quad$ $n_s = n_s + 1$

20 $\quad$ CALL *FIS Generation* (Algorithm 2)

21 $\quad$ keep FIS_{iter}

22 $\quad$ $iter = iter + 1$

23 **end**

Algorithm 2: FIS generation

    input : $\{n_j \mid j = 1, ..., p\}$, $FP_j^{n_j} \forall j = 1, ..., p$

1   Generate the $\prod_{j=1}^{p} n_j$ rule premises

2   **forall the** *Rule* $r \in FIS$ **do**

3      $CV_r = \sum\limits_{k=1}^{n} w^r(x_k)$

4   **if** $CV_r < CV_t$ **then**

5   remove rule r

6   **else**

7   initialize rule conclusion

8   **end**

9   **end**

10   Compute Perf

The rule conclusion initialization, line 7, depends on both the rule induction method - either *WM* or *FPA* can be used - and on the system output type, regression or classification.

As noted above, the outcome of the procedure is not a single fuzzy inference system, but K FIS of increasing complexity. The best one is selected based on the performance and the coverage indices.

2.4.2. Simplification

Most of the induction methods, fuzzy decision trees being a noticeable exception, yield bases consisting of rules described by the same set of input variables. We call such bases complete rule bases.

In these *complete rule bases*, it is somewhat difficult to give a meaning to the rules. All variables are showing up equally in all rules. If we wish to interpret the rules as interaction rules, it is important to think of a means to privilege the strongest interactions. A good way to do so is to try out a simplification procedure leading to an incomplete rule base, where some variables (one or more) appear in some rules only.

The elimination of variables in order to obtain incomplete rules could be undertaken at different levels. Many existing methods remove variables from the whole rule base, on the faith of overall indicators which could be misleading. Other techniques remove variables from one rule at a time, not considering any relationship that could exist between the rules. We favor an intermediate selection

19

level, which is an attempt to make up for these difficulties. This intermediate level is chosen as the level of a *group of rules* with a *common context*. Our main axis in the simplification procedure focuses on the merging of some rules into a more generic incomplete rule.

A group of rules is a set of rules whose premises only differ by a single fuzzy set label, corresponding to the same variable v.

The procedure consists of examining each group of rules to see if it can be replaced by a generic incomplete rule, formed by removing the variable v in each of the premises of the rules that constitute the group. These rules are all identical within the group, so they can be replaced by a single new rule.

In the illustration of Figure 4, the group consists of three rules that only differ by the label of V_2. The procedure merges them into the R_g rule defined only by the variables V_1 and V_3.

Rule *3 1 2* $\iff$ IF V_1 is *3* and V_2 is *1* and V_3 is *2* THEN ...

$$
\begin{cases}
\text{R1}: & 3 \quad 1 \quad 2 \\
\text{R2}: & 3 \quad 2 \quad 2 \\
\text{R3}: & 3 \quad 3 \quad 2
\end{cases}
\iff R_g : 3\,0\,2 \quad \text{If } V_1 \text{ is 3 and } V_3 \text{ is 2 then} \ldots
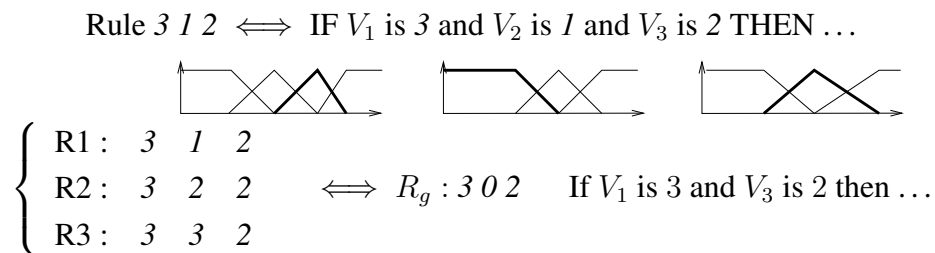$$

Figure 4: Group rule merging

The merging is guided by the performance index (as defined in section 2.1.2 for regression and classification cases), but also by a careful examination of the heterogeneity of the output. For regression cases, it is computed as the ratio of the output variance of the N^r items matching the rule standardized by the similar computation for the whole data set, according to Equation 11.

$$H^r = \frac{\sigma^r}{\sigma} \tag{11}$$

The calculations of σ^r and σ are detailed in Equation 12.

$$\sigma = \frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y}_i)^2, \ \sigma^r = \frac{\displaystyle\sum_{i=1}^{N^r} w^r(x_i)(y_i - \bar{y}_i)^2}{\displaystyle\sum_{i=1}^{N^r} w^r(x_i)} \tag{12}$$

20

For classification cases the index is computed as a normalized entropy as shown in Equation 13, where p_k is the ratio of examples with class k among those matching the rule.

$$H^r = \frac{-\sum_{k=1}^{K} p_k log(p_k)}{log K}, \ p_k = \frac{\sum_{i=1}^{N^r} w^r(x_i)d_k(x)}{\sum_{i=1}^{N^r} w^r(x_i)} \tag{13}$$

where d_k is the characteristic function of class k; $d_k(x) = 1$ if x belongs to class k, 0 otherwise.

Why bother with this index?

To gain interpretability, we can tolerate a loss of performance. However, we must be cautious about the consequences of the widening of the space potentially covered by the new rule. If the output heterogeneity associated to the new rule is too high, then the replacement should not be made. Nevertheless, some heterogeneity is unavoidable and even desirable. It keeps the rules from being too specific.

The simplification procedure is an iterative one. If the rule belongs to at least one group that is to be replaced by a generic rule, the replacement will be made and the original rule will be removed from the base. Indeed, the generic rule covers a wider multidimensional space than the original one.

Redundancy is likely to occur in the generated rule base, as there is no redundancy control during the generation phase. To overcome this drawback, rules of the base issued from the simplification procedure are tentatively removed one after the other.

The simplification procedure can be applied to any rule base, whatever the induction method.

These three approaches, fuzzy decision trees, refinement and simplification are illustrated with case studies in Section 4.

2.5. Concluding remarks

Some general points are worth noting when using FIS learning methods. Some regions of the input domains may not be handled by the FIS. For instance, this could happen for regions that are hardly (or not at all) represented in the learning data set. This is the price to pay to improve generalization.

Regarding the output domain, as the fuzzy inference engine classically interpolates between rules, some output values may be inferred even if they do not

correspond to a rule conclusion. This possibility is interesting for modeling continuous phenomena, it but must be avoided in non-ordered classification cases by choosing a suitable defuzzification operator, as illustrated in Section 4.2.

Learning methods implemented in *FisPro* yield coherent non-contradictory rules.

However, the simplification of the rule base may lead to systems where a rule is included within another. In that case, both rules are considered during the inference process, which does not include any checking mechanism for such a phenomenon. *FisPro* actually proposes another inference mechanism (see [27] for details), adapted to fuzzy implicative rules, where partially redundant rules are handled in a more proper way. The *KBCT* [2] software also has an option to eliminate redundant rules from a fuzzy rule base.

A flowchart is displayed in Figure 5 to provide some guidelines for the main choices during FIS generation. The main steps are summarized from top to bottom. If expert knowledge is available, input partitioning can be done by hand; otherwise, the input partitions can be designed entirely from the data, using *k-means* or *hfp*.

The output design depends on the nature of output data: categorical data (classes) or numerical continuous data.
In the first case, the output nature is *crisp*, whereas in the second case it can be *crisp* for *Sugeno FIS*, or *fuzzy* for *Mamdani FIS*. The choice of the appropriate defuzzification operator is described in Figure 5.

In *FisPro*, the rule aggregation operator is selected at the output level, with a choice between *max* and *sum* (for conjunctive rule systems).

There is no universal best choice of the rule learning method. However, if the number p of input variables is large, it is recommended to do first a variable selection procedure, for which *Fuzzy Decision Trees* are particularly suitable. Once p is sufficiently small, the appropriate rule method may be *WM* for classification problems. For continuous outputs, when the number n of examples is not too high, *Fuzzy Decision Trees* can be the best choice, as they have a good interpolating capability. When n is large, a statistically inspired method like *OLS* will do very well to extract the most significant rules and pinpoint the outliers, while *FPA* will run faster and give an image of the data prototypes.

Independently of the way the FIS was designed, the rule base can be simplified, as explained in section 2.4.2. *FisPro* also includes an optimization module, which allows us to optimize any FIS component with interpretability constraints. The simplification and optimization procedures can be performed using a validation data set.
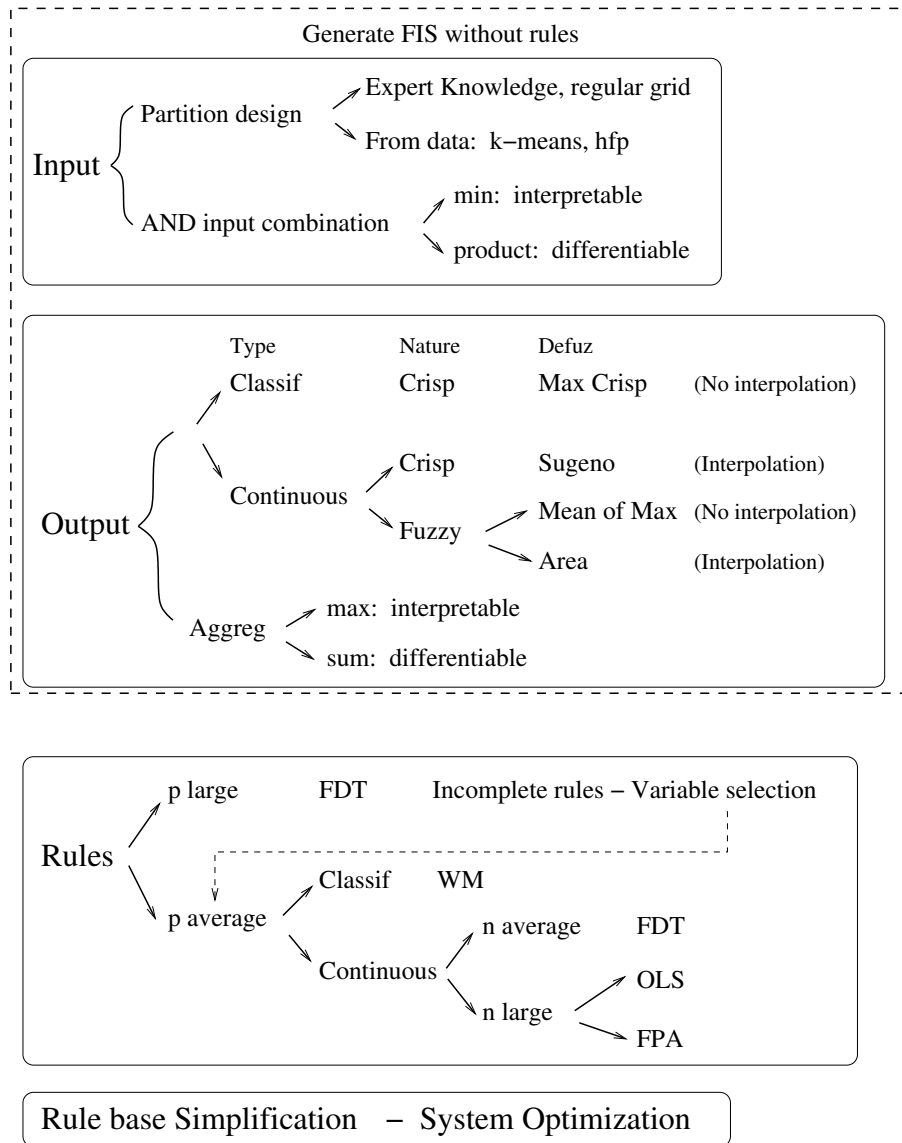
22

Figure 5: Flowchart for learning with *FisPro* - n is the number of data rows; p is the number of input variables.

Let us briefly consider the complexity of the rule learning methods. If we neglect the number of MFs per variable, the time complexity only depends on the number of rows n in the data set, and the number p of input variables. *FPA* and *WM* have a complexity of $O(np)$, *FDT* has a complexity of $O(np!)$, while *OLS* has a complexity of $O(n^3)$.

Obviously, *FisPro* current limits will first be reached for the rule induction using the *OLS* algorithm, which may also be limited by the available memory because it requires the storage of a square size n matrix. On an Intel Pentium 4 CPU 3.40GHz with 1 Gigabyte memory, an *OLS* rule learning procedure takes about 30 s for a data set with 54000 rows.

Parallel computing could be used to circumvent the current limits of *FisPro*. It is already available to speed up inference calculations with an *OpenMp* compliant implementation.

3. FisPro

Fuzzy software was first developed for the needs of fuzzy control, the popularity of which has been asserted in the 1990s. Industrial as well as academic software became available, and the targeted audience was control engineers, who used fuzzy software as an alternative in the domain of control system design. When elaborate learning methods became more mature and when fuzzy logic expanded to other fields of interest, more general fuzzy software appeared to provide these methods.

A special session on software for soft computing was organized at the 2007 FuzzIEEE conference [25]. A comprehensive review of fuzzy software, an interesting discussion of useful features, and a call for building a fuzzy tool kit that supports the take-up of fuzzy systems in business applications can be found in [29], which appeared in the proceedings. During the same conference, some advanced software projects were presented, such as FrIDA [7], a free intelligent data analysis toolbox, or Xfuzzy [5]. Xfuzzy is a development environment that integrates a set of tools to help the user through the several stages involved in the process of designing fuzzy logic-based inference systems.

The *FisPro* toolbox is an open source toolkit for interpretable FIS design using expert knowledge and data. It stands out among fuzzy software products, because of the interpretability of the fuzzy systems automatically learnt from data, guaranteed in each step of the FIS design, according to the principles detailed in

Section 2: variable partitioning, rule induction, optimization. We now present the main features that are useful for FIS learning.

3.1. Sample generation

Sample files may be generated by random sampling from a data file. There are two possibilities: 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 so as to respect the class proportions in a data file.

3.2. Fuzzy partitioning and FIS with no rules

Visualization tools are available to examine partitions and data jointly. An example is given in Figure 6. The X axis range is set to the input variable range, while the Y axis ranges from 0 to 1 for the fuzzy partition (bottom part) and from 0 to the maximum number of elements in a class (top part). This visualization is useful to examine the concept significance in relation to actual data.
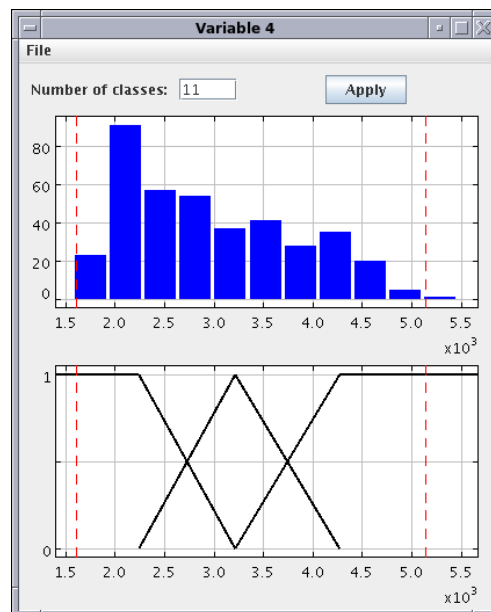

Figure 6: Examples of fuzzy partition and data distribution - *auto* data set, fourth variable (horse-power)

FIS with no rules can be generated by all partitioning methods: k-means, regular grids, or hfp hierarchy, and it can be reused in any further rule learning procedure.

3.3. FIS learning

When a data file is open in *FisPro*, it becomes available as a learning set. Variable histograms, two and three dimensional plots are proposed to examine the data distribution and the correlations between variables. Data rows can be individually activated or deactivated from the plots or from the data array viewing window.

Likewise, input variables may be activated or deactivated, and all modifications are automatically passed on to the learning set that will be used for learning.

Rule learning methods are assigned their parameters through a user-friendly interface, with the possibility to use test files when they are relevant to the method.

Links between rules and data can be evaluated using a utility function that creates several files. One contains the number of items that activate each rule beyond the μ threshold parameter and the cumulative weight of the rule over the data file. Another one indicates the rules matched by each data row, and the third one contains a square matrix, whose size is equal to the number of rules, and for which the i, j cell gives the corresponding linkage level, calculated as:

$$L_{i,j} = \frac{N_{i,j}}{N_i}$$

where N_i is the size of the subset E_i of the items that activate the ith rule, and $N_{i,j}$ is the size of $E_i \cap E_j$.

3.4. Viewing results

After learning, the FIS performance can be calculated on a whole data set, and a summary of coverage and accuracy results is displayed. For regression cases, several plots are available: histograms, X-Y plots with or without a regression line, error plots, and response surfaces. For small data files, an exploratory analysis may be done by moving the mouse over a data point and displaying its row number and the activated rule numbers (beyond a given μ).

For a classification system, the result is presented as a *confusion matrix* (an example is given in Figure 9, Section 4.2).

Specific graphical representations are available for FDT, in table or graph form, with relevant information including the number of examples attracted by a leaf, the fuzzy cardinality per leaf, and the entropy or deviance per leaf.

Examples of specific plots are given in section 4.

4. Case studies

In this section, the ability of *FisPro* to design accurate and interpretable FIS is illustrated in three case studies. The two first case studies are well known data sets from the UCI repository [14], one is a regression problem, and the other one deals with classification. The data sets are the following:

- *auto-mpg* (392 samples):
  From the StatLib library maintained at Carnegie Mellon University, this case concerns the prediction of city-cycle fuel consumption in miles per gallon from four continuous and three multi-valued discrete variables.

- *wine* (178 samples):
  The data set contains samples that are grown in the same region of Italy but derived from three different cultivars. The numbers of instances in each class are: 59, 71 and 48. Each pattern consists of thirteen continuous features resulting from chemical analysis.

The last data set used is related to pesticide losses during spraying.

4.1. Regression case

The first step for FIS design is to build the fuzzy partitions. This is done using the *Generate FIS without rules* option. The number of terms is set to *three* for all of the input variables, and the *k-means* algorithm is used. The *min* is used for premise combination, the output is *crisp*, and the defuzzification operator is the *Sugeno* weighted average, given in Equation 14.

$$\widehat{y} = \frac{\sum\limits_{r=1}^{R} m_r(x)C^r}{\sum\limits_{r=1}^{m} m_r(x)} \tag{14}$$

$m_r(x)$ is defined in Equation 5, and R is the number of rules.

The rules are then induced with the *OLS* method using a ten-fold cross validation. Ten pairs of training (75%) and test (25%) sets are randomly chosen. For

27

each pair, the rule induction is based on the training set, while the performance evaluation is based on the corresponding test one.

The number of induced rules is chosen to be equal to 19 in order to ensure a sufficient accuracy. It is measured by the Mean Absolute Error (MAE) defined in Equation 2.

The average results over the ten test sets yield a MAE of 2.23 with a coverage index of 97%. The standard deviations of the MAE and coverage index are equal to 0.18 and 1.9, respectively, which show a satisfactory robustness of the learning procedure.

To illustrate the system's behavior, we use the FIS generated by the $third$ pair, for which the results are close to the average ones.

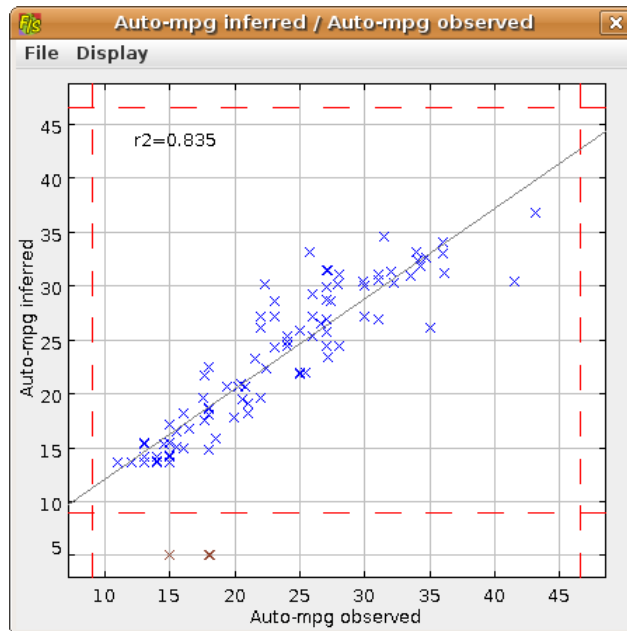Figure 7 shows a screenshot from *FisPro*.

Figure 7: Auto-mpg: Inference on test sample#3 with FIS built from training sample #3

Three examples do not fire any rules, and are set to a user defined *default* output value of 5, in order to easily identify them. Mouse-over events are enabled to get row number information (44, 54 and 70).

As said previously, *OLS* generates complete rules. We do not expect all of the variables to be necessary in all of the rules. The simplification procedure is

applied to the FIS induced from sample #3, with the same training data.

The new rule base is simpler: it has 12 rules instead of 19, and 5 of them are incomplete rules. Nevertheless, the system is still accurate: the new RMSE, measured over the whole data set is 0.0101 instead of 0.0098, i.e. a 3% loss. Readers can refer to [12] for a compared study of performance results on these data.

4.2. Classification case

First *a FIS without rules* is generated with three fuzzy sets by variable, using a k-means procedure. The output is *crisp* with a *classification* flag enabled, meaning that the output is a class discrete value. To avoid rule interpolation the defuzzification operator chosen is a special one called *max crisp*, defined below.

Let m be the number of distinct rule conclusion values and C^r the conclusion of the rth rule. A given x input vector matches the rth rule to a degree denoted $w^r(x)$. The cumulative weights are calculated for all distinct values j, and the inferred output is the j value that corresponds to the maximum weight.

$$\begin{cases} W^j = \sum_r m_r(x) \mid C^r = j \\ \widehat{y} = argmax\left(W^j\right) \mid j = 1\dots m \end{cases} \tag{15}$$

WM is used for rule induction. As the input space is relatively large, the number of cells is 3^{13} and there are few items in a given cell (subspace). This yields 133 rules. As the number of rules is comparable with the training set size, the rule base cannot claim to be general.

FPA is an alternative for rule induction. The *FisPro* default parameters, at least 3 samples that match a rule to a degree higher than 0.3, lead to a huge rule base with 2506 rules. The number of rules could be reduced using the control parameter, but this would give a poor image of the data. Despite the high number of rules, the inference results are not really good: 50 misclassified items with *FPA* and 15 with *WM*.

In such a case, it is better to reduce the input space before rule induction. The refinement procedure is then applied, using *WM* as the rule induction method.

Table 4 summarizes the main results. The first variable to be introduced is $V12$. At this step, there are only two rules in the rule base, and 79 items are misclassified.

As new variables are automatically introduced or refined, the number of error cases decreases. The coverage index, as defined in Section 2.1.1, remains equal to 1 for all of these configurations.

29

MC	V1	V2	V3	V4	V8	V10	V11	V12	V13	#R
79	1	1	1	1	1	1	1	2	1	2
32	1	1	1	1	1	1	1	2	2	4
19	1	1	1	1	1	1	2	2	2	7
12	1	1	1	1	1	2	2	2	2	12
11	1	2	1	1	1	2	2	2	2	19
9	1	2	2	1	1	2	2	2	2	33
8	1	2	2	2	1	2	2	2	2	52
5	2	2	2	2	1	2	2	2	2	67
3	2	2	2	2	1	2	3	2	2	78
2	3	2	2	2	1	2	3	2	2	94
1	3	2	2	2	2	2	3	2	2	108

Table 4: Wine refinement procedure

The final choice results from a trade-off between accuracy and model complexity. The 12 rule FIS, presented in Figure 8, appears to be a good compromise, with a misclassification rate equal to $12/178$, i.e. 6.7%.

Figure 8: Wine: The selected rule base

Figures 8 and 9 show the selected rule base and the corresponding confusion matrix. Reference results are available in the literature [34]. We do not present a detailed study here, but, as in the regression case introduced in Section 4.1, cross validation procedures can be applied to sampled data sets to test the robustness of

30

Figure 9: Wine: The confusion matrix

the algorithms.

4.3. Pesticide loss modeling

These data are part of an experiment whose goal is to study the influence of micro-meteorological factors on pesticide loss to the air during vine spraying. A thorough description of both the problem statement and the results can be found in [16].

The spraying is achieved using air assisted devices to aid the transport of the droplets toward the target. The modeling objective is to propose a relationship between the proportion of product lost in the atmosphere, given as a percentage of the total volume (%), and some micro-meteorological variables. The following variables are considered:

- W : Wind speed (m/s)

- T : Air temperature (^{o}C)

- ΔT : Wet bulb temperature depression (^{o}C)

- z/L : Atmosphere stability parameter

The data are difficult to measure, and the sample size is small: 32 experiments. We proceed as follows: first we use the refinement procedure described in Section 2.2 to select the *best* number of fuzzy sets per input variable, except for the *Wind speed*, which is a three-term expert designed partition, shown in Figure 10. It is in agreement with the Beaufort scale, leading to highly interpretable rules, as each linguistic label corresponds to a Beaufort degree.

The refinement procedure selects four fuzzy sets for the second and fourth variable, and two for the third one.

Then, a fuzzy regression decision tree is generated with a minimum μ equal to 0.3. It is pruned to allow different temporary accuracy losses: 5% and 30%. The
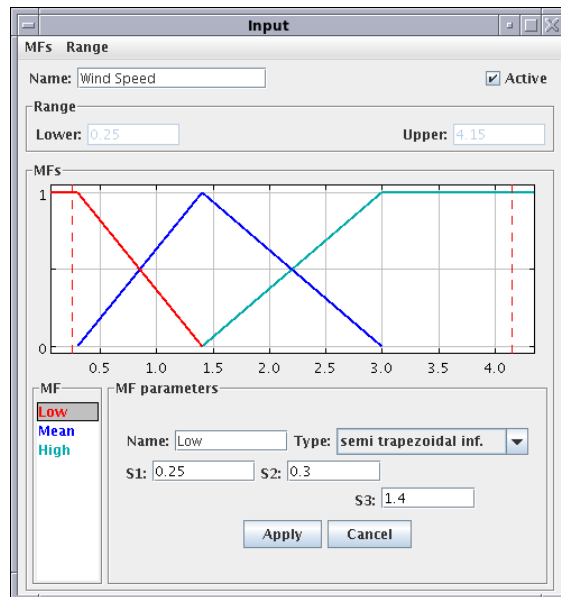
31

Figure 10: Wind Speed fuzzy input partition for pesticide loss data

#rules	Tree	avg #var. per rule	RMSE
29	initial tree	3.4	0.021
14	first pruned tree	2.4	0.016
6	final pruned tree	2.0	0.019

Table 5: *Tree equivalent FIS* features for pesticide loss - initial and pruned trees

results are shown in Table 5, including the accuracy and the average number of variables per rule, and the six leaf pruned decision tree is displayed in Figure 11, with each leaf labeled by the number of attracted data rows and their mean value.

The tree is very easy to interpret: two rules include only one variable, two others include two variables, and the last two rules include three variables. This kind of system can be used to recommend suitable spraying periods: avoid windy times, but even if the wind velocity is moderate, prefer times when air temperature is high to minimize losses.
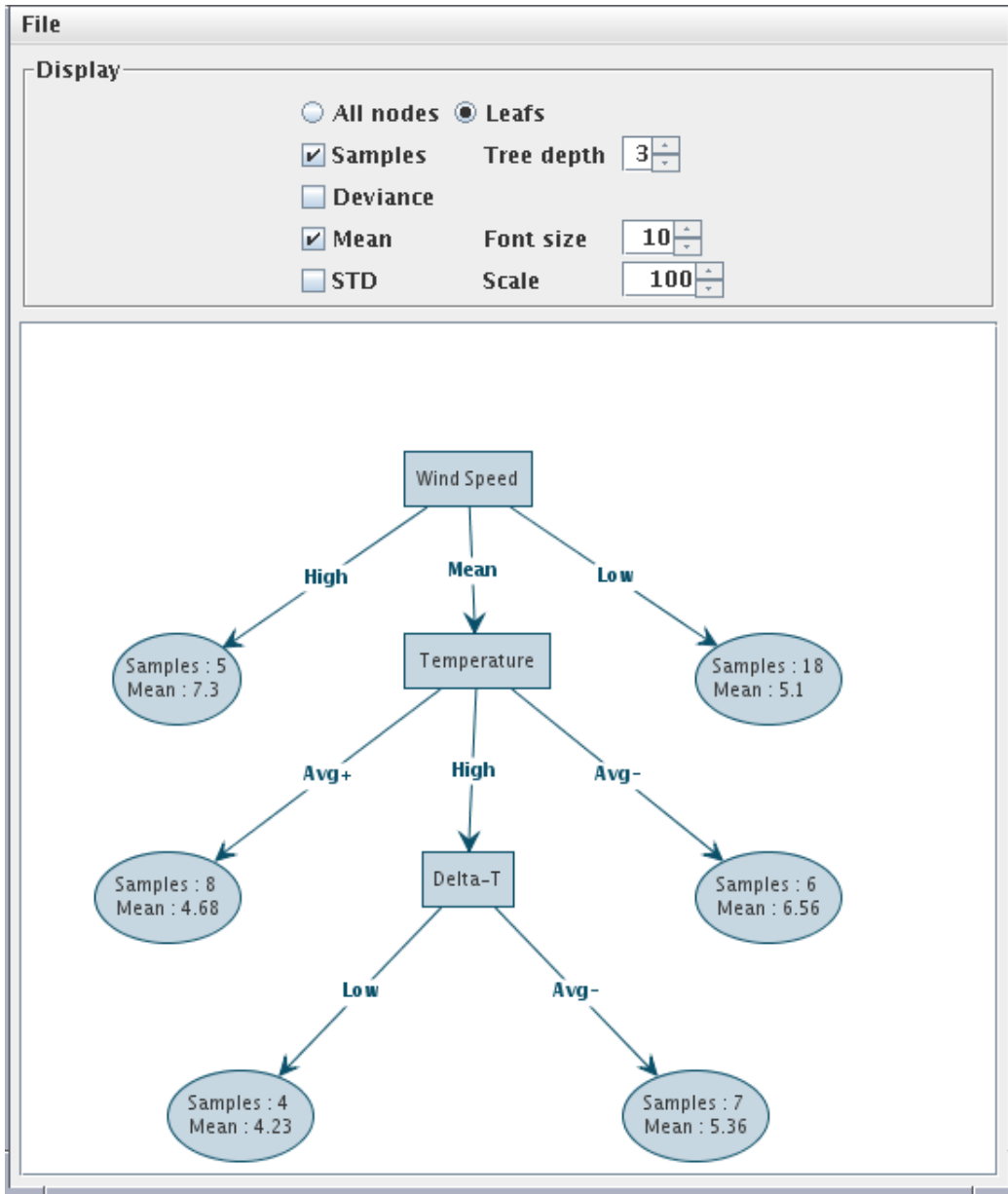
32

Figure 11: Fuzzy decision tree for pesticide loss data

5. Conclusion

This work has endeavored to present a generic approach to the design of interpretable data-driven FIS for system modeling and its implementation in an *open source* software called *FisPro*[1].

The approach addresses several FIS design related topics, including fuzzy partitioning, rule learning, variable selection and rule base simplification.

It has been illustrated on data sets of varying natures: regression and classification benchmarks, and a short case study in the field of environmental modeling.

Several real world modeling problems in various fields - food science, image analysis and agriculture - were treated in collaboration with the authors of *FisPro*. The reader can find more details in the following references [16], [21], [19], [13], [37].

The main features of *FisPro* related to FIS learning were described, as well the available learning methods and the dynamical interface and exploratory analysis tools that allow interactive design.

In addition to the user-friendly *FisPro* interface, the learning programs are also available as C++ self contained source code, which allows them to be called from a line command and easily included in batch scripts for cross validation procedures. This functionality, associated with automatic sampling procedures, allows the system to run intensively computational procedures and facilitates the testing of the algorithm's robustness and efficiency.

Modular and open source, the *FisPro* software project welcomes contributions from artificial intelligence scientists or engineers. It has already been used for modeling projects in application fields different from the ones cited above and by researchers from different communities, as shown by the recent references appearing in the literature [32, 8]. More references are available on the *FisPro* Web Site. Hopefully the present paper will serve as a reference for users and will help them accomplish their modeling tasks.

References

[1] Alonso, J.M., 2010. Guaje: Generating understandable and accurate fuzzy models in a java environment. http://www.softcomputing.es/guaje.

---

[1]freely available at http://www.inra.fr/Internet/Departements/MIA/M/fispro

[2] Alonso, J.M., Guillaume, S., Magdalena, L., 2003. Kbct: A knowledge management tool for fuzzy inference systems. `http://www.mat.upm.es/projects/advocate/kbct.htm"`.

[3] Alonso, J.M., Magdalena, L., 2010. Guaje - a java environment for generating understandable and accurate models, in: XV Spanish conference for Fuzzy Logic and Technology, Universidad de Huelva, Spain. pp. 399–404.

[4] Alonso, J.M., Magdalena, L., Guillaume, S., 2008. Hilk:a new methodology for designing highly interpretable linguistic knowledge bases using the fuzzy logic formalism. International Journal of Intelligent Systems 23, 761–794.

[5] Baturone, I., Moreno-Velo, F.J., Snchez-Solano, S., Barrios, A.B., Jimnez, P.B., Gersnoviez, A., Brox, M., 2007. Using xfuzzy environment for the whole design of fuzzy systems., in: [25]. pp. 1–6. pp. 1–6.

[6] Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Functions Algorithms. Plenum Press, New York.

[7] Borgelt, C., Gonzáles-Rodríguez, G., 2007. Frida - a free intelligent data analysis toolbox, in: [25]. pp. 1–5. pp. 1–5.

[8] Bossomaier, T., Standish, R.K., Harré, M., 2010. Simulation of trust in client-wealth management adviser relationships. International Journal of Simulation and Process Modelling 6, 40–49.

[9] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth International Group, Belmont CA.

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

[11] Chen, M.Y., 2002. Establishing interpretable fuzzy models from numerica data, in: Proceedings of the 4th World Congress on Intelligent Control and Automation, IEEE. pp. 1857–1861.

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

[13] El Hajj, M., Bégué, A., Guillaume, 2009. Integrating spot-5 time series, crop growth modeling and expert knowledge for monitoring agricultural practices - the case of sugarcane harvest on reunion island. Remote Sensing of Environment 113 (10), 2052–2061.

[14] Frank, A., Asuncion, A., 2010. UCI machine learning repository.

[15] Gacto, M.J., Alcalá, R., Herrera, F., 2010. Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems. IEEE Transactions on Fuzzy Systems 18 (3), 515–531.

[16] Gil, Y., Sinfort, C., Guillaume, S., Brunet, Y., Palagos, B., 2008. Influence of micrometeorological factors on pesticide loss to the air during vine spraying: Data analysis with statistical and fuzzy inference models. Biosystems Engineering 100(2), 184–197.

[17] Glorennec, P.Y., 1996. Quelques aspects analytiques des systèmes d'inférence floue. Journal Européen des Systèmes automatisés 30 (2-3), 231–254.

[18] Glorennec, P.Y., 1999. Algorithmes d'apprentissage pour systèmes d'inférence floue. Editions Hermès, Paris.

[19] Goelzer, A., Charnomordic, B., Colombié, S., Fromion, V., Sablayrolles, J., 2009. Simulation and optimization software for alcoholic fermentation in winemaking conditions. Food Control 20, 635 – 642.

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

[21] Guillaume, S., Charnomordic, B., 2004a. Fuzzy Inference Systems to Model Sensory Evaluation. Intelligent Sensory Evaluation- Methodologies and Applications, Springer. pp. 197–216.

[22] Guillaume, S., Charnomordic, B., 2004b. Fuzzy models to deal with sensory data in food industry. Journal of Donghua University 21 (3), 43–48.

[23] Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182.

36

[24] Hüllermeier, E., Vanderlooy, S., 2009. Why fuzzy decision trees are good rankers. IEEE Transactions on Fuzzy Systems 17(6), 1233–1244.

[25] IEEE (Ed.), 2007. FUZZ-IEEE 2007, IEEE International Conference on Fuzzy Systems, Imperial College, London, UK, 23-26 July, 2007, Proceedings, IEEE.

[26] Jang, J.S.R., Sun, C.T., Mizutani, E., 1997. Neuro-Fuzzy and Soft Computing. Prentice Hall.

[27] Jones, H., Charnomordic, B., Dubois, D., Guillaume, S., 2009. Practical inference with systems of gradual implicative rules. IEEE Transactions on Fuzzy Systems 17 (1), 61–78.

[28] Mencar, C., Fanelli, A.M., 2008. Interpretability constraints for fuzzy information granulation. Information Sciences 178 (24), 4585–4618.

[29] Nauck, D.D., 2007. Gnu fuzzy, in: [25]. pp. 1019–1024. pp. 1019–1024.

[30] de Oliveira, J.V., 1999. Semantic constraints for membership functions optimization. IEEE Transactions on Systems, Man and Cybernetics. Part A 29, 128–138.

[31] Quinlan, J.R., 1986. Induction of decision trees. Machine Learning 1, 81–106.

[32] Rajaram, T., Das, A., 2010. Modeling of interactions among sustainability components of an agro-ecosystem using local knowledge through cognitive mapping and fuzzy inference system. Expert Systems with Applications 37, 1734–1744.

[33] Riid, A., Rüstern, E., 2003. Transparent Fuzzy Systems in Modelling and Control. Interpretability Issues in Fuzzy Modeling, Studies in Fuzziness and Soft Computing, Vol 128, Springer. pp. 452–476.

[34] Roubos, J.A., Setnes, M., Abonyi, J., 2003. Learning fuzzy classification rules from labeled data. Inf. Sci. Inf. Comput. Sci. 150, 77–93.

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

37

[36] Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on System Man and Cybernetics 15, 116–132.

[37] Tremblay, N., Bouroubi, M., Panneton, B., Guillaume, S., Vigneault, P., Bélec, C., 2010. Development and validation of fuzzy logic inference to determine optimum rates of n for corn on the basis of field and crop features. Precision Agriculture 11, 621–635.

[38] Wang, L.X., Mendel, J.M., 1992a. Fuzzy basis functions, universal approximation, and orthogonal least squares learning. IEEE Transactions on Neural Networks 3, 807–814.

[39] Wang, L.X., Mendel, J.M., 1992b. Generating fuzzy rules by learning from examples. IEEE Transactions on Systems, Man and Cybernetics 22 (6), 1414–1427.

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

[41] Yager, R.R., 2010. A framework for reasoning with soft information. Information Sciences 180, 1390–1406.