Designing Fuzzy Inference Systems from Data: An Interpretability-Oriented Review

Serge Guillaume

***Abstract*—Fuzzy inference systems (FIS) are widely used for process simulation or control. They can be designed either from expert knowledge or from data. For complex systems, FIS based on expert knowledge only may suffer from a loss of accuracy. This is the main incentive for using fuzzy rules inferred from data. Designing a FIS from data can be decomposed into two main phases: automatic rule generation and system optimization. Rule generation leads to a basic system with a given space partitioning and the corresponding set of rules. System optimization can be done at various levels. Variable selection can be an overall selection or it can be managed rule by rule. Rule base optimization aims to select the most useful rules and to optimize rule conclusions. Space partitioning can be improved by adding or removing fuzzy sets and by tuning membership function parameters. Structure optimization is of a major importance: selecting variables, reducing the rule base and optimizing the number of fuzzy sets. Over the years, many methods have become available for designing FIS from data. Their efficiency is usually characterized by a numerical performance index. However, for human-computer cooperation another criterion is needed: the rule interpretability. An implicit assumption states that fuzzy rules are by nature easy to be interpreted. This could be wrong when dealing with complex multivariable systems or when the generated partitioning is meaningless for experts. This paper analyzes the main methods for automatic rule generation and structure optimization. They are grouped into several families and compared according to the rule interpretability criterion. For this purpose, three conditions for a set of rules to be interpretable are defined.**

***Index Terms*—Fuzzy inference systems, fuzzy partitioning, interpretability, rule induction, system optimization.**

I. Introduction

FUZZY inference systems (FIS) are one of the most famous applications of fuzzy logic and fuzzy sets theory [1]. They can be helpful to achieve classification tasks, offline process simulation and diagnosis, online decision support tools and process control.

The strength of FIS relies on their twofold identity. On the one hand, they are able to handle linguistic concepts. On the other hand, they are universal approximators able to perform nonlinear mappings between inputs and outputs. These two characteristics have been used to design two kinds of FIS.

The first kind of FIS to appear focused on the ability of fuzzy logic to model natural language [2]. These FIS contain fuzzy rules built from expert knowledge and they are called fuzzy expert systems or fuzzy controllers, depending on their final use. Prior to FIS, expert knowledge was already used to build expert systems for simulation purposes. These expert systems were based on classical boolean logic and were not well suited to managing the progressiveness in the underlying process phenomena. Fuzzy logic allows gradual rules to be introduced into expert knowledge based simulators. It also points out the limitations of human knowledge, particularly the difficulties in formalizing interactions in complex processes. This kind of FIS offers a high semantic level and a good generalization capability. Unfortunately, the complexity of large systems may lead to an insufficient accuracy in the simulation results. Expert knowledge only based FIS may show poor performances.

Another class of simulation tools is based on automatic learning from data. This study is restricted to supervised learning and observed outputs are part of the training data. Thus, a numerical performance index can be defined which is usually based on the mean square error. Neural networks have become very popular. Their main advantage is the numerical accuracy while a major drawback is their *black box* behavior. Indeed, they provide a numerical model, whose coefficients have no meaning for experts. Sugeno [3] was one of the first to propose self-learning FIS and to open the way to a second kind of FIS; those designed from data. Even if the fuzzy rules, which are automatically generated from data, are expressed in the same form as expert rules, there is generally a loss of semantic. Since Sugeno's early work, a lot of researchers have been involved in designing fuzzy systems from databases. This paper aims to introduce the main methods for designing fuzzy inference systems from data. All these methods can be considered as rule generation techniques. Rule generation can be decomposed into two main steps: 1) rule induction and 2) rule-base optimization. Originally, automatic induction methods were applied to simple systems with a few variables. In these conditions, there is no need for optimizing the rule base. The situation is different for large systems. The number of induced rules becomes enormous and the rule description is complex because of the number of variables. Obviously, the rules will be easier to interpret if they are defined by the most influential variables and the system behavior will be easier to understand as the number of rules is getting smaller. Variable selection and rule reduction are, thus, two important steps of the rule generation process. They are ususally referred as structure optimization.

Apart from structure optimization, a FIS has many parameters that can also be optimized, i.e., membership functions parameters and rule conclusion adjustment. This is called parameter optimization. A thorough study has been done by various authors [4], [5], their respective advantages and drawbacks are well known.

In this review, rule induction and rule-base optimization methods will be compared and analyzed according to the most important criterion for human-computer cooperation:

Manuscript received September 20, 2000; revised November 26, 2000.
The author is with Cemagref, 34033 Montpellier, France.
Publisher Item Identifier S 1063-6706(01)04535-0.

their interpretability. Many authors seem to consider that interpretability is automatically given by the fuzzy formalism, but when dealing with large systems it is not true. The rule-base legibility is an important condition to take full advantage of fuzzy inference systems. That means providing a good framework for cooperation between two kinds of information: expert knowledge and hidden knowledge in data.

Section II gives the notation used in this paper, which consists of two main parts dealing with rule induction and structure optimization. Rule induction techniques are gathered in three main families, each of them analyzed in one separate section. Section III introduces the shared partitioning induction methods. Section IV is devoted to clustering. Hybrid methods are presented in Section V. In Section VI, the different approaches are compared and summarized. The optimization part is divided into two sections. Section VII deals with variable selection and Section VIII discusses rule-base optimization methods. Finally, the paper is concluded in Section IX, which recalls the main features from an interpretability point of view.

II. Notation

Let us give some basic definitions. The training set contains n data pairs. Each pair is made of a *p-dimensional* input-vector x and a *q-dimensional* output-vector y. The number of rules in the FIS rule base is r.

Mamdani's rule i within this system is written as follows:

$$\begin{aligned}&\mathrm{If\ x_1\ is\ A_1^i\ and\ x_2\ is\ A_2^i \ldots and\ x_p\ is\ A_p^i}\\&\mathrm{Then\ y_1\ is\ C_1^i \ldots and\ y_q\ is\ C_q^i}\end{aligned}$$

where A_j^i and C_j^i are fuzzy sets that define an input and output space partitioning.

In Sugeno's model, the conclusion of the rule i for output j is computed as a linear function of the inputs: $y_j^i = b_{jo}^i + b_{j1}^i x_1 + b_{j2}^i x_2 + \cdots + b_{jp}^i x_p$, also written as: $y_j^i = f_j^i(x)$.

A fuzzy rule is called an incomplete rule if its premise is defined by a subset of the available variables only. Let us consider a two input, one output system. The rule: $\mathrm{If\ x_2\ is\ A_2^1\ Then\ y\ is\ C_2}$, is an incomplete one because it does not use the x_1 input variable. Expert rules are mainly incomplete rules, they contain only the most influential variables. Formally, an incomplete rule uses implicit and and or logical connectors. If the x_1 input variable space is partitioned into three fuzzy sets, the incomplete rule given above can be written as

$$\begin{aligned}&\mathrm{If\ (x_1\ is\ A_1^1\ or\ x_1\ is\ A_1^2\ or\ x_1\ is\ A_1^3)}\\&\mathrm{and\ x_2\ is\ A_2^1\ Then\ y\ is\ C_2.}\end{aligned}$$

For a given rule i, its firestrength, also called weight and written w_i, is computed as a conjunction operation between the premise elements: $w_i = \mu_{A_1^i}(x_1) \wedge \mu_{A_2^i}(x_2) \wedge \ldots \wedge \mu_{A_p^i}(x_p)$, where $\mu_{A_j^i}(x_j)$ is the membership degree of x_j to the fuzzy set A_j^i and $\wedge$ is the *and* operator. Minimum and product are the most common *and* operators.

The X_j input variable partitioning is called a strong partitioning, if $\forall x \in X_j$, $\sum_i \mu_{A_j^i}(x) = 1$.

The mean square error (MSE) is computed as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|\hat{y}_i - y_i\|^2$$

$\hat{y}_i$ being the inferred output for example i.

Part I—Rule Induction

There are two kinds of rule induction methods. The first kind uses a grid partitioning of the multidimensional space. The partitioning can be generated from data or given by experts. It defines a number of fuzzy sets for each variable, which are interpreted as linguistic labels and shared by all the rules. A training procedure optimizes the grid structure, as well as the rule consequences, according to data samples. These methods are introduced in Section III.

The second kind is the clustering introduced in Section IV. The training pairs are gathered into homogeneous groups and a rule is associated to each group. The fuzzy sets are not shared by the rules, but each of them is tailored for one particular rule.

Section V presents another family called hybrid methods. They are based on soft computing techniques. Their group is more heterogenous than the others, the results are highly dependent on implementation and encoding.

III. The Fuzzy Sets Shared By All the Rules

A common way to generate a grid partitioning consists in dividing each input variable domain into a given number of intervals whose limits do not necessarily have any physical meaning and do not take into account a data density repartition function. We will introduce several approaches.

The first, and most intuitive approach implements all possible combinations of the given fuzzy sets as rules. This way of doing shows some drawbacks, which are handled by additional methods. Due to an insufficient work-space coverage, some rules may never be fired. However, a diffusion procedure can be used to initialize the unfired rules.

The choice of the number of fuzzy sets in each dimension carries significant consequences: it can be dynamically chosen within the second approach.

When the number of combinations increases, it is necessary to limit the number of rules: the third approach initializes one rule per data pair.

At last, the decision trees are introduced at the end of this section. They generate incomplete rules but require a predetermined fuzzy partitioning.

A. All the Rules Implemented

Ishibuchi *et al.* [6] consider a multiple input, single output system. They assume the input and output spaces to be [0, 1]^p and [0, 1]. For the ith input variable x_i, its domain interval is evenly divided into K_i fuzzy sets labeled as $A_i^1, A_i^2, \cdots, A_i^{K_i}$ as shown in Fig. 1 for the x_1 variable with $K_1 = 5$. Any kind of

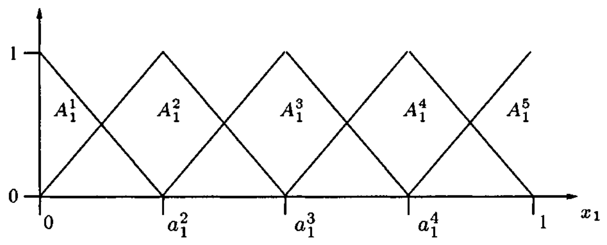


Fig. 1. Automatic partitioning with 5 triangular membership functions.

membership function can be used, the most common being the triangle-shaped one

$$\mu_{ij}(x) = \max\left\{1 - \frac{|x - a_j^{K_i}|}{b^{K_i}}, 0\right\}$$
$$a_j^{K_i} = \frac{j-1}{K_i - 1}, \qquad j = 1, \ldots, K_i; \qquad b^{K_i} = \frac{1}{K_i - 1}$$

All the rules corresponding to the possible combinations of the inputs are implemented. The total number of rules for a p input system is: $K_1 \times K_2 \times \cdots \times K_p$.

Nozaki *et al.* [7] propose a simple heuristic to calculate the rule conclusions, which are real numbers. For rule i, the conclusion is written b_i and computed as

$$b_i = \frac{\sum_{j=1}^{n} w_i(j) * y(j)}{\sum_{j=1}^{n} w_i(j)}$$

$y(j)$ being the j pair observed output and $w_i(j)$ the i rule fire-strength for the j pair.

The j pair inferred output is then

$$\hat{y}(j) = \frac{\sum_{i=1}^{r} w_i(j) * b_i}{\sum_{i=1}^{r} w_i(j)}. \tag{1}$$

Depending on both the choice of the K_i values and the input space covering, some rules may never be fired by training examples. Glorennec [5] proposes a diffusion procedure in order to initialize the corresponding conclusions.

Let S be the set of rules whose conclusions have already been initialized; let us define the N_{S_i} neighborhood, N_i being the neighborhood of rule i

$$N_{S_i} = \begin{cases} \{i\}, & \text{if } i \in S \\ N_i, & \text{otherwise} \end{cases}$$

N_i is basically defined by the sets of rules whose premises differ from rule i by only one fuzzy set. According to this definition, each rule has at most two neighbors in each input space dimension and, thus, $|N_i| \leq 2^p$. The diffusion is done according to the following series $g^{(n)}$

$$g^{(0)}(i) = \begin{cases} b_i, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$
$$g^{(n+1)}(i) = \tfrac{1}{2} \sup_{j \in N_{S_i}} g^{(n)}(j) + \tfrac{1}{2} \inf_{j \in N_{S_i}} g^{(n)}(j).$$

The series converges when n goes toward infinity and the diffusion procedure is stable.

B. Number of Fuzzy Sets Dynamically Chosen

The former method requires the K_i to be set. For a given input variable the choice of K_i carries significant consequences. If K_i is too small, the system won't be able to model a nonlinear behavior, it won't be accurate enough. Conversely it is difficult to increase K_i too much, due to the following reasons: 1) if K_i is too large, the corresponding fuzzy sets tend to be too specific, resulting in a loss of generality and 2) the number of rules is the product of all the K_i coefficients.

To avoid fixing the K_i values, some authors propose to derive them from the data.

1) Partition Refinement: Bortolet [8] uses a partition refinement. At each step of the algorithm, a fuzzy set is added on the input that is responsible for the greatest part of the error.

Initially, each input is divided into two triangle-shaped fuzzy sets. They are centered on the minimum and the maximum values of the considered input domain.

At each step, all the rules corresponding to the possible combinations are implemented. The ith rule conclusion is first estimated using the least square regression. Let us note n_{r_i} the number of linearly independent pairs whose weight for rule i is greater than a given threshold, typically set to 0.5

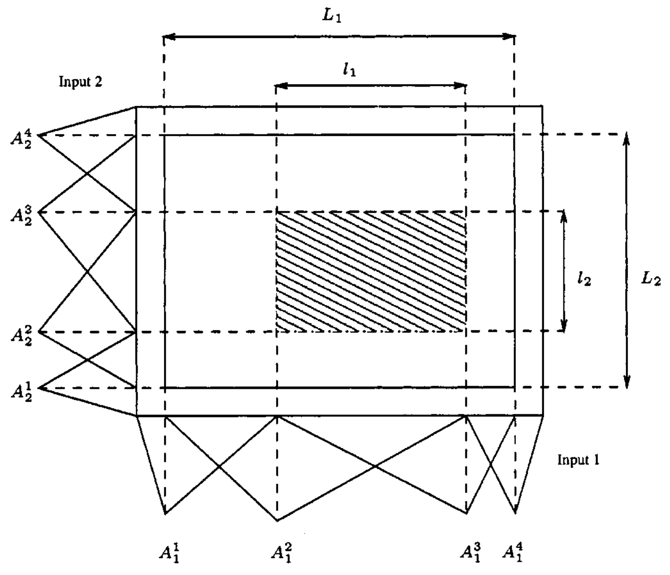
$$y_{\text{train}}(j) = a_0 + \sum_{i=1}^{p} a_i * x_i(j). \tag{2}$$

The a_i coefficients are those that minimize the difference between y_{train} and the observed output for the n_{r_i} pairs[1]. The rule conclusion is obtained by replacing the x_i values in (2) by the centers of the corresponding fuzzy sets.

The system for the corresponding fuzzy partitioning is, thus, completely defined. It is now possible to process all of the n pairs. A new fuzzy set is added to prepare the next step of the algorithm. This is done by identifying the region of the input space, then the input variable, and finally the center of the new fuzzy set. A region of the input space is bounded by the vertices of two consecutive membership functions on each input variable.

An error index is associated to each region. It is computed as the product of the mean error for the pairs belonging to the region (n_i for region i) by the ratio of the input domain covered by the considered region. An error index associated to each input variable within the considered region is computed in the same way. As an example, Fig. 2 shows the region defined by four

[1]If $n_{r_i} < p + 1$, the regression cannot be achieved and less precise methods are proposed.



Error of region i:

$$E_i = \frac{\sum_{k=1}^{n_i} |y_k - \hat{y}_k|}{n_i} * \frac{l_1}{L_1} * \frac{l_2}{L_2}$$

Error of input j within region i:

$$E_j = \frac{\sum_{k=1}^{n_i} |y_k - \hat{y}_k|}{n_i} * \frac{l_j}{L_j}$$

Fig. 2. Partition refinement.

fuzzy sets (A_1^2, A_1^3, A_2^2, A_2^3) in a two-input system and the formulae corresponding to the error indices.

The selected region and input are the ones for which the indices are the greatest. The new center coordinate on the selected input, j, is computed as

$$A_j^{\text{new}} = \frac{\sum_{k=1}^{n_i} x_{kj} |y_k - \hat{y}_k|}{\sum_{k=1}^{n_i} |y_k - \hat{y}_k|.}$$

The limits of the new triangular membership function are the centers of the fuzzy sets between which the new fuzzy set has been inserted. So, the input partitioning is still a strong partitioning.

Finally, the rule conclusions corresponding to the modified part of the input space are updated. The system is ready for the next iteration.

The algorithm stops when the error reaches a minimum or when it becomes smaller than a given threshold. The method does not contain any protection against introducing into the model the noise included in data: the only criterion to add a new fuzzy set is the error and it does not take into account the current system.

In [9], the refinement is based upon a controversy index. This index, defined at the rule level, indicates the difference between the rule conclusion and the observed output for the datapoints that activate the corresponding rule. It is computed as follows for rule i

$$CI(i) = \sum_{k=1}^{n} \left([(y_k - R_i) w_i(k)]^2 \right)^{1/2}$$

R_i	ith rule conclusion;
y_k	kth example observed output;
$w_i(k)$	ith rule firestrength for the kth data point.

This definition can be extended to a membership function. The index is then called the sum of controversies associated with a given membership function and written as

$$\text{SCMF}(X_v^j) = \sum_{r \in R} CI(r)$$

R is the set of rules whose antecedent in the X_v variable refers to the jth membership function.

To make the index values comparable it is normalized by the product of the number of fuzzy sets for the remaining variables

$$\widehat{\text{CI}}(X_v^j) = \frac{\text{CI}(X_v^j)}{\prod_{\substack{m=1 \\ m \neq v}}^{p} n_m}$$

n_m is the number of membership functions of the mth variable.

A new membership function is added for variables whose controversy index variance is high. The authors use a strong triangular partitioning, so that only the centers have to be stored. The new center location is computed as

$$c_v^* = \frac{\sum_{j=1}^{n_v} c_v^j \, \text{SCMF}(X_v^j)}{\sum_{j=1}^{n_v} \text{SCMF}(X_v^j)}.$$

Contrary to Bortolet's method, the criterion is not evaluated in an input space region, but at the rule level.

2) Using a Genetic Algorithm: Ishibuchi *et al.* [10] deal with classification problems. They want to generate fuzzy rules that divide the input space into C disjoint decision areas, C being the number of classes to discriminate.

Different fuzzy partitions are automatically generated with different values of K_i, the number of fuzzy sets for the ith variable. The coarser partitioning corresponds to small values of K_i. The genetic algorithm is used to select the best suited level for each of the input space regions according to the data. All rules corresponding to a given partitioning are considered as possible and the objective function of the genetic algorithm takes into account both the performance of the system and the rule base size. Through the evolution process the selected systems are those which maximize

$$f(S) = W_c C_S - W_r R_S$$

C_S	pairs well classified by system S;
R_S	rules in the system;
W_c, W_r	corresponding weights.

C. Only One Rule per Data Pair

In the method introduced by Wang and Mendel [11], the number of rules is limited by the number of training pairs. It does not depend on the fuzzy partition resolution level, i.e., the number of fuzzy sets for each input variable. They propose the following five step procedure:

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 pair, the ith pair one is written

$$\text{if } \mathrm{x}_1 \text{ is } \mathrm{A}_1^{\mathrm{i}} \text{ and } \mathrm{x}_2 \text{ is } \mathrm{A}_2^{\mathrm{i}} \ldots \text{and } \mathrm{x}_\mathrm{p} \text{ is } \mathrm{A}_\mathrm{p}^{\mathrm{i}} \text{ then } \mathrm{y} \text{ is } \mathrm{C}^{\mathrm{i}}.$$

The fuzzy sets A_j^i are those for which the degree of match of x_j^i is maximum for each input variable j from pair i. The fuzzy set C_i is the one for which the degree of match of the observed output, y_i, is maximum.
3) A degree is assigned to each rule. For a given rule it is equal to the rule firestrength for the considered pair. If some *a priori* information is available, the confidence level of each pair will be used too, the degree being the product of the firestrength by the confidence level. In case of identical premises for two rules, only the one with the higher degree is kept.
4) Experts rules are allowed. The *and* rules induced from data may be combined with *or* rules given by experts. The membership degree for the missing variables is set to one, the neutral element for the product operation. *Or* and *and* type rules are equally managed.
5) The output is computed through the centroid defuzzification.

This procedure allows the rule base to be adaptive: new rules competing with existing ones.

D. Decision Trees

The decision trees were proposed by Quinlan [12]. Their application is restricted to classification approaches. The objective is to design paths leading to pure leaves with each leaf corresponding to an incomplete rule. The tree represents a subspace of all the possible rules.

Ichihashi *et al.* [13] propose a neuro-fuzzy implementation of Quinlan's interactive dichotomizer (ID3) algorithm. Unlike the other methods mentioned in Section III, the input space partitioning must be user defined prior to running the algorithm.

The tree induction is an iterative process. At each step a new node is added. A node corresponds to an input variable and generates a number of subnodes equal to the number of fuzzy sets (also called attributes) of the selected variable. The process is repeated until all leaves are pure, i.e., they contain elements belonging to the same given class.

The selected variable at a given step, is the one that maximizes the information gain. The tree can be regarded as a source of a message. The information needed to generate this message is the sum for all the nodes, of the node entropies. The rule associated to a given node b is written as

$$\text{If } \mathrm{x}_{\mathrm{i}_1} \text{ is } \mathrm{A}_{\mathrm{i}_1}^{\mathrm{j}_1} \text{ and } \mathrm{x}_{\mathrm{i}_2} \text{ is } \mathrm{A}_{\mathrm{i}_2}^{\mathrm{j}_2} \ldots \text{Then } \mathrm{y} \text{ is } \mathrm{C}_\mathrm{b}$$

$A_{i_1}^{j_1}$ corresponds to the first node of the path starting from the root and leading to the node b, meaning that the first selected variable is i_1 and the subtree leading to node b starts from the j_1 attribute of this variable. C_b is the most represented class in node b. An illustration is shown in Fig. 3.

The premise of the rule corresponding to node b is defined by the set, Q, of the couples (i, j), the jth attribute of the ith input variable, along the branch from the root to node b.

The entropy for node b is defined as

$$H_b = -\sum_k p_k^b * \log(p_k^b)$$

p_k^b is the k class density within b node, that means the proportion ratio of elements belonging to class k. The cardinalities are fuzzy and computed as the sum of the rule firestrengths for all the elements in the node

$$p_k^b = \frac{|D_k^b|}{|D^b|} \quad \text{with} \quad |D^b| = \sum_{x \in D^b} \left(\prod_{(i,j) \in Q} \mu_{i,j}(x) \right)$$

$\mu_{i,j}(x)$ is the membership degree of pair x input i value to the j_{th} fuzzy set of input i. $|D_k^b|$ is defined in the same way but with the subset of $x \in D^b$ which belongs to class k.

Let H be the node entropy and V the number of fuzzy sets of the considered input variable. The new entropy is the weighted sum of the subnode entropies

$$E = \sum_{v=1}^{V} q^{b_v} * H_v \quad \text{with} \quad q^{b_v} = |D^{b_v}|/|D^b|.$$

The information gained by selecting the considered input variable is $G = H - E$.

To cope with expert uncertainty, the algorithm is adapted to deal with belief functions within the evidence theory formalism proposed by Dempster and Shafer [14].

The main advantage of fuzzy decision trees is that they generate incomplete rules constrained to a given partitioning. Incomplete rules were introduced in Section II. They offer a compact description of a given context by using only the locally most significant variables. The rules generated by decision trees will be informative for experts provided that the initial partitioning was carefully defined.

IV. Fuzzy Clustering

Fuzzy clustering algorithms form a well- identified family of rule induction techniques. They are used to organize and categorize data. The result is a partition of the data into homogeneous groups. The space partitioning is derived from the data partitioning and a rule is associated to each cluster. Unlike within

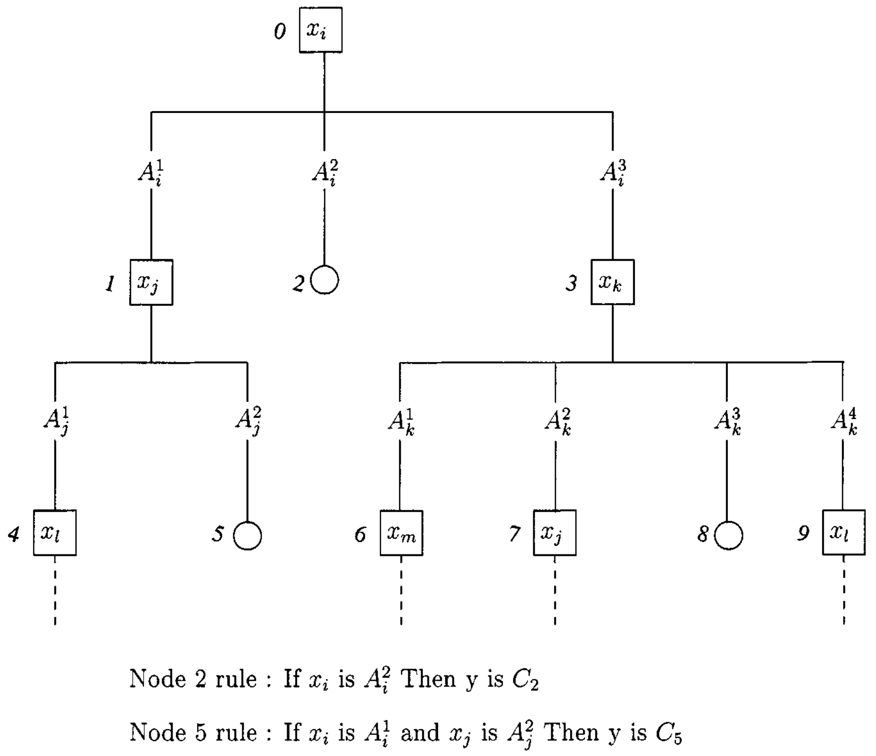

0 x_i
A_i^1
A_i^2
A_i^3
1 x_j
2
3 x_k
A_j^1
A_j^2
A_k^1
A_k^2
A_k^3
A_k^4
4 x_l
5
6 x_m
7 x_j
8
9 x_l
Node 2 rule : If x_i is A_i^2 Then y is C_2
Node 5 rule : If x_i is A_i^1 and x_j is A_j^2 Then y is C_5

Fig. 3. An illustration of a fuzzy decision tree.

the previous section, the fuzzy sets are not shared by the set of rules. For a given dimension, each of them is tailored for one rule only. The resulting fuzzy sets are usually difficult to interpret.

A. Fuzzy C-Means Clustering

The first method, called Fuzzy C-means, was introduced by Dunn in 1973 [15]. Bezdek demonstrated its properties and proposed the first cluster validity criteria [16], [17]. Each of the n data pairs belongs to each of the c groups with a membership coefficient, u_{ik} being the membership degree of pair k to cluster i. Let D_{ki}^2 be the distance between pair k and cluster i, basically defined as the Euclidean norm and more generally as

$$D_{ki}^2 = ||x_k - v_i||_A^2 = (x_k - v_i)A(x_k - v_i)^T$$

x_k being the k_{th} data pair used for the clustering, A being a positive definite symmetric matrix, and v_i being the prototype of cluster i.

Let U be the u_{ik} coefficient matrix and V the center coordinate matrix. The algorithm yields U and V which minimize the following loss function

$$J_{\text{FCM}} = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^m D_{ki}^2$$

under the probabilistic constraint:

$$\sum_{i=1}^{c} u_{ik} = 1 \quad \forall k = 1, \cdots, n$$

$m \geq 1$, is the fuzzy exponent.

The function optimization is done by an alternating optimization procedure. First, the u_{ik} coefficients are randomly initialized. Then, at each step, the two following operations are successively carried out.

1) Compute the fuzzy centers v_i, assuming the u_{ik} degrees are constant numbers, using the following equation:

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}. \tag{3}$$

2) Compute the memberships u_{ik}, assuming the v_i centers are constant vectors, using

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{D_{ik}}{D_{jk}}\right)^{2/m-1}}. \tag{4}$$

These operations are reiterated until convergence when the center coordinates are stable with respect to a given tolerance.

The FCM algorithm is suitable for clusters with comparable size and shape (spherical when using the identity matrix) or when the clusters are well separated. The cluster prototypes are data points chosen as the cluster centers.

1) Variations of the Original Algorithm: A lot of improvements or generalizations of the basic algorithm have been proposed.

Krishnapuram [18], [19] introduced the possibilistic C-means by releasing the probabilistic constraint and by adding a punishment term in the loss function in order to penalize low membership degrees. The PCM loss function is written as

$$J_{\mathrm{PCM}} = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik}^m\|x_k - v_i\|^2 + \eta_i(1-u_{ik}^m))$$

$$\eta_i = \frac{\sum_{k=1}^{n} u_{ik}^m\|x_k - v_i\|^2}{\sum_{k=1}^{n} u_{ik}^m}.$$

In the Gustafson–Kessel algorithm [20], the A matrix is defined according to the data. The covariance matrix, for group i, is

$$C_i = \frac{\sum_{k=1}^{n} u_{ik}^m (x_k - v_i)^T (x_k - v_i)}{\sum_{k=1}^{n} u_{ik}^m}.$$

And the distance between pair k and group i becomes

$$D_{ki}^2 = (x_k - v_i)[\det(C_i)^{1/n} C_i^{-1}](x_k - v_i)^T.$$

The fuzzy C-regression model (FCRM) [21]–[23] produces hyperplane-shaped clusters, instead of hypersphere-shaped ones for FCM and the prototypes are hyperplanes instead of datapoints. The prototype of the ith group is

$$y_i = X^T P^i \qquad X = [1 \quad x_1 \quad \dots \quad x_m]^T$$
$$P^i = [a_0^i \quad a_1^i \quad \dots \quad a_m^i]^T.$$

The premise membership functions are generalized Gaussians expressed as

$$A(x) = e^{-(x-p_1/p_2)^2}$$

p_1 and p_2 are tuned by a gradient method.

2) Which Data for Fuzzy Clustering?: Fuzzy clustering can be done using input–output data, input data only, or output data only. Depending on this choice the induced rules may or may not be completely defined.

Some authors [24], [25] want to take advantage of all the available information and apply the clustering to the product space, $X \times Y$. Therefore, the corresponding rule is completely defined: the premise corresponds to the input part, and the conclusion to the output part. Input plus output based clustering could be confusing. Some items could belong to the same cluster while being neither close in the input space nor in the output space. Their closeness in the cluster is due to distances compensating each other in the input-output space.

Sugeno and Emami [26], [27] run the clustering in the output space. The rule premises are then defined by projecting clusters onto the input space. This operation is not trivial and the result is usually affected by some noise. It can happen that several rules be generated from a single cluster. Indeed, projection of the multidimensional cluster onto one input dimension may yield more than one fuzzy set. This feature could reflect a real property as there exist different premises leading to the same conclusion.

When using only the input part of the data pairs, a conflict management procedure is needed: some pairs with different output values may belong to the same group because their input parts are similar.

The FCM algorithm and its derivatives requires some parameters such as the number of clusters and the value of the fuzzy exponent.

B. Cluster Validity

Since Bezdek's early work, many teams have been involved in finding the optimal number of groups, also called the cluster validity problem. Two main techniques are available: run the FCM algorithm with an increased number of clusters ($c = 2, \dots, n-1$) and characterize each partition using indexes or, run one time only an algorithm which determines by itself the best suited number of groups.

1) Indices to Characterize Fuzzy Partitions: Xie and Beni [28] define the best partition as the one that minimizes the ratio of compactness, $C(c)$ to separation $S(c)$. These measures ares defined as follows

$$C(c) = \frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^m \|x_k - v_i\|_A^2$$
$$S(c) = \min_{i\neq j}\|v_i - v_j\|_A^2.$$

Sugeno [26] suggests choosing the number of groups which minimizes the following criterion

$$V(c) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m(\|x_k - v_i\|_A^2 - \|v_i - \overline{x}\|_A^2)$$

$\overline{x}$ being the centroid of the data set. The first term is the *within* group variance, the second one is the *between* group variance.

Emami *et al.* [27] use a similar formula, the centroid $\overline{x}$ is replaced by its fuzzy extension $\overline{v}$. The difference between Enami's and Sugeno's criteria gets larger as the fuzzy exponent increases.

Burrough *et al.* [29] use a coefficient partition F and a classification entropy H, to characterize each partition. They are defined as

$$F = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^2, \qquad \frac{1}{c} < F < 1$$
$$H = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{c} -u_{ik} * \ln(u_{ik})$$
$$1 - F < H < \ln(c).$$

Their values depend on the number of clusters. In order to make F and H independent, they can be scaled as

$$F_s = \frac{F - \frac{1}{c}}{1 - \frac{1}{c}} \quad \text{and} \quad H_s = \frac{H - (1-F)}{\ln(c) - (1-F)}.$$

For good partitions F_s values are expected to be large while H_s values are expected to be small. The user has to find a compromise.

2) Subtractive Clustering: the algorithm proposed by Chiu [24] is an improvement of the one called "mountain method" developed by Yager [30].

Each data point is considered as a potential cluster center. A measure of the potential is associated to each point according to its neighborhood, itself being defined by a radius, r_a. For the point i, it is written as

$$P_i = \sum_{j=1}^{n} e^{-4\|x_i - x_j\|^2/r_a^2} \quad \text{with} \quad r_a > 0.$$

The point with the highest potential, P_1^* written x_1^*, is selected to become the first cluster center. Once the center is selected, the potential of each pair is decreased according to its distance to x_1^*. The new potential for pair i becomes

$$P_i = P_i - P_1^* e^{-4\|x_i - x_1^*\|^2/r_b^2}, \quad \text{with} \quad r_b \approx 1.5 r_a.$$

The process is repeated. To avoid introducing the noisy part of the data into the model, two thresholds are defined. $s+$ and $s-$ typically set to 0.5 and 0.15, and a new center candidate x_k^* at step k with its associated potential P_k^*, is managed as follows:

```
if $P_k^* > P_1^* * s+$, $x_k^*$ is accepted as a new cluster center;
if $P_k^* < P_1^* * s-$, $x_k^*$ is rejected as a new cluster center and the
  algorithm stops;
else
    let $d_{\min}$ be the shortest distance between $x_k^*$ and all previously found cluster centers;
        if $(d_{\min}/r_a) + (P_k^*/P_1^*) \geq 1$ $x_k^*$ is accepted as a new cluster center;
            (that means if it is far enough from the closest cluster)
    else $x_k^*$ is rejected, its potential is set to 0 and the algorithm goes on.
```

This algorithm is quite sensitive to the different parameters such as the neighborhood radius and the potential thresholds. Unfortunately, there is no theoretical guidance for choosing them.

Fig. 4 shows the results of the subtractive clustering on a 100 random pair set. The five centers found by the algorithm are highlighted in Fig. 4(a). They correspond to five rules whose premises are defined by Gaussian membership functions shown in Figs. 4(b) and 4(c). These functions are computed by projecting the points belonging to each cluster onto each dimension.

C. Tuning the Fuzzy Exponent

The value of the fuzzy exponent controls the amount of fuzziness in the clustering process. The larger it is, the fuzzier the partition. When m tends toward infinity, all cluster centers tend toward the centroid of the data. Many authors recommend a fixed value of m, usually 1.5 or 2.

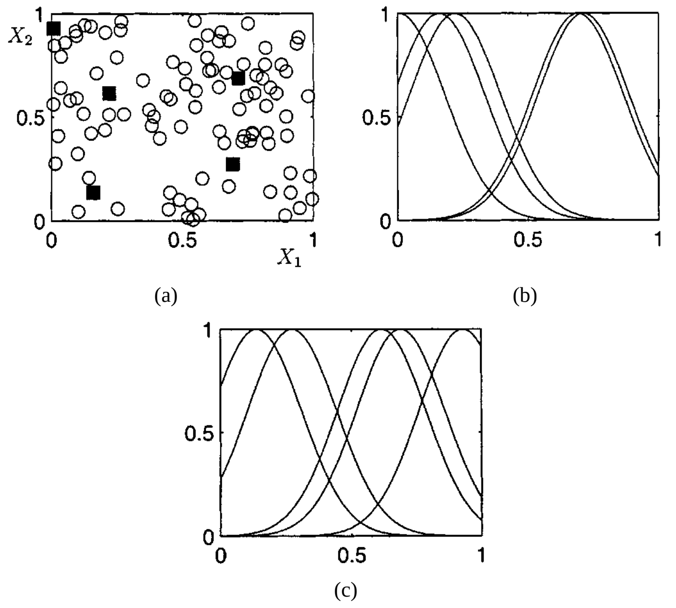


Fig. 4. The subtractive clustering. (a) Data pairs and clusters. (b) X_1 variable membership functions. (c) X_2 variable membership functions.

Chen and Wang [31] propose an iterative method to tune the fuzzy exponent. The membership function associated to cluster i is

$$\text{MF}_i(x) = \frac{1}{1 + \left(\frac{x - v_i}{\sigma_i}\right)^{b_i}}$$

v_i, σ_i, b_i	center, width, and crossover slope of the function;
σ_i	square root of the trace of the i group covariance matrix;
b_i	chosen according to σ_i and cluster center locations to make sure membership functions overlap enough to avoid inference breaking.

The objective is to find a value of m, such as there exists for each dimension of the work space; at least one cluster for which the inner deviation for the kth dimension, σ_{ik}, is greater than the training set deviation for the given dimension, σ_k. The value of m, initially set to 1.5, is increased by 0.1 at each step of the algorithm. This costly method needs to run the FCM algorithm and to compute the covariance matrices for each increment. An alternative way would be to check the sensitivity of the final model to the fuzzy exponent.

Another method, proposed by Li and Mukaidono [32], does not use any fuzzy exponent. The loss function to minimize is written

$$J = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} D_{ki}^2$$

$$\sum_{i=1}^{c} u_{ik} = 1 \quad \forall k = 1, \ldots, n.$$

This last method called Gaussian-clustering method, maximizes the entropy with respect to each input pair k, under the two following constraints: (a) minimization of the loss function for pair

k, and (b) normalization of membership degrees. The problem becomes

$$\begin{aligned} &\text{maximize} && \left\{-K\sum_{i=1}^{c}u_{ik}\log(u_{ik})\right\} \qquad K > 0\\ &\text{subject to (a)} && \sum_{i=1}^{c}u_{ik}D_{ki}^2 = \kappa\\ & && \kappa \text{ being a small positive number}\\ &\text{and (b)} && \sum_{i=1}^{c}u_{ik} = 1. \end{aligned} \tag{5}$$

Their algorithm is similar to the FCM one, the cluster prototypes v_i are updated using (3), while (4) to update the u_{ik} is replaced by (6) which is the solution of the optimization problem (5)

$$u_{ik} = \frac{e^{-(D_{ki}^2/2\sigma^2)}}{\sum_{j=1}^{c}e^{-(D_{ki}^2/2\sigma^2)}} \tag{6}$$

$2\sigma^2$ is called the temperature, σ being related to κ by constraint (a).

V. Hybrid Methods

This set of methods integrates many different tools, the most famous and widely used being the genetic algorithms and the neural networks. Neural networks brought their learning algorithms and numerical accuracy to FIS without paying much attention to the semantic. Genetic algorithms are more likely to find a global optimum and may optimize both the structure and the parameters of the corresponding FIS. These tools prove useful when there is no available expert knowledge and for applications for which semantic is not a prime concern.

A. Neuro-Fuzzy Modeling

Neuro-fuzzy models [33], including adaptive neuro fuzzy inference systems (ANFIS) [34], [4] are fuzzy inference systems implemented as neural nets. Each layer in the network corresponds to a part of the FIS: input fuzzification, rule inference and firestrength computation, and output defuzzification. The main advantage of this kind of representation is that the FIS parameters are encoded as weights in the neural network and, thus, can be optimized via powerful well known neural net learning methods (Hebbian rule, back-propagation, etc.).

In this paper, we first focus on a particular type called radial basis functions (RBF) networks. The main idea of RBF relies on a local tuning of the process units, each of them corresponding to a local model. The architecture was first proposed by Moody and Darken [35], since then a lot of work has been done to bridge the gap between neural nets and FIS. Jang showed that RBF are equivalent to FIS under few restrictive conditions [36], the most important being that the rule conclusion are scalars. More recently Cho and Wang [37] suggested improvements to deal with polynomial or fuzzy conclusions.

An RBF is a three layer network: 1) the input layer of size equal to the input vector size p; 2) the output layer of size q; and 3) one hidden layer. The number of nodes in the hidden layer corresponds to the number of rules and it is upper bounded by the number of pairs.

Hidden layer units are locally tuned radial receptive fields. Learning aims to setup the network so that a hidden unit recognizes one and only one kind of pattern. The hidden layer is fully connected to the input layer[2] and unit i performs the following operation: $R_i(x) = e^{-((x-c_i)^2)/\sigma_i^2}$, c_i and σ_i being the center and the standard deviation of the Gaussian membership function,[3] respectively.

The output layer is fully connected to the hidden layer. Within the configuration where rule conclusion are scalars, the defuzzification is easy. For each output, the corresponding unit computes the weighted sum of the connections and the weight is the firestrength of the rule for the current pair. When the rule conclusions are polynomial, $y_i = b_o^i + b_1^i x_1 + b_2^i x_2 + \cdots + b_p^i x_p$, the weights between input and hidden layers are not constant. The weights correspond to the $b_1^i, \cdots, b_p^i$ coefficients, and a fixed input, set to 1, is artificially added with a b_0^i weight.

Learning consists of determining the minimum number of units in the hidden layer, i.e., the number of rules, the corresponding vectors c_i and σ_i, and their weights.

First, the number of rules is set to 0. Then, at each step all the pairs of the training set are processed in turn. For each pair i, at step k, where $r_k(j)$ is the jth hypersphere radius and ϵ is a tolerance value

```
if $|y_i - \hat{y}_{ik}| > \epsilon$, then
  if there exists node $j$ such as $(x_i - c_j)^2/\sigma_j^2 < r_k(j)$, then
    modify $c_j$ and $\sigma_j$ using the gradient method
  else create new node whose center is $x_i$[4]
else train the hidden nodes using the gradient method.
```

Once all the pairs have been processed, the hypersphere radii are decreased before the next step.

The algorithm terminates when the squared sum of errors is less than a given tolerance or after a predefined number of iterations have been done. Note that this algorithm may be sensitive to the data processing order.

This technique is close to the subtractive algorithm introduced in Section IV-B.2 and, thus, could be classified as a clustering one.

Recent work attempts to use neural networks in a different way, with interpretability in mind. In [38], [39] the authors build a network for classification purposes. Each input is partitioned into three fuzzy sets; each fuzzy set being in turn modified by three hedges. The network has two output nodes per class or convex subclass. The first node is used to classify items which belong to the output class (positive items), and the second one to recognize negative items for the same class. The interpretability effort consists in generating a single rule for each output node.

[2]That means each unit of the hidden layer is connected to each unit of the input layer.

[3]Any kind of radial basis function can be used, the Gaussian one is given as an example.

[4]The initial value of σ is computed using the standard deviation of the data set.

TABLE I
MAIN METHODS FOR FUZZY RULE INDUCTION FROM DATA

Family	**Method**[a]	**Drawbacks**	**Answers**
Shared partitions	All the possible rules *(III-A)*	Curse of dimensionality	Partition refinement *(III-B.1)*
	One rule per pair *(III-C)*	Completeness not guaranteed	
	Decision trees *(III-D)*	Pre-requisite partitioning	
Clustering :	FCM *(IV-A)*	Number of clusters?	Subtractive clustering *(IV-B.2)*
Hybrid Methods	Neuro-fuzzy including RBF *(V-A)* Genetic Algorithms *(V-B)*	Depending on implementation, on encoding.	

[a] The number in parenthesis is the section number which describes the approach.

This is achieved by a backtracking procedure that selects the maximal weighted path from the input layer to the output node.

B. Genetic Algorithms

Since they were proposed by Goldberg [40], genetic algorithms (GA) have been widely used to learn input output relations and to design fuzzy controllers [41]–[45].[5] As an illustration, let us examine the model introduced by Russo [47] which aims to combine the respective advantages of fuzzy logic, neural networks and genetic algorithms.

Its evolutionary algorithm considers a population of neural networks. Training consists of adjusting the different weights, unit removal is allowed. Once defined, the network is encoded as a chromosome and evolves within the population using selection, crossover and mutation operations.

Its evolutionary algorithm considers a population of neural networks. Training consists of adjusting the different weights, unit removal is allowed. Once defined, the network is encoded as a chromosome and evolves within the population using selection, crossover and mutation operations.

The network, corresponding to a r-rule FIS, is made of 4 layers:

1) *Input layer*: it has at most p neurons, p being the input vector size.
2) *Fuzzification layer*: the number of neurons is at most rp. There is one fuzzy set for each active input and for each rule. Its Gaussian membership depends on c and γ, the center and the inverse of the standard deviation respectively.[6] These values are encoded as weights and learnt through a back-propagation algorithm. An important choice is done in this layer: the fuzzy sets are tailored for each rule. The complexity is decreased compared with fuzzy sets shared by all rules, but the induced partitioning is less suitable for human cooperation.
3) *Inference layer*: the rule firestrength is computed using the *min* operator. All weights are set to one.
4) *Output layer*: rule conclusion are scalars. The defuzzification is either done using the weighted mean of rule contributions [see (1)] or using their weighted sum.

[5] Although GA ae very popular other stochastic techniques can be used such as simulated annealing[46]

[6]The use of γ instead of σ allows the optimization of the learning time (by replacing the division operation by a multiplication), and may avoid singularities in the neighborhood of $\sigma = 0$.

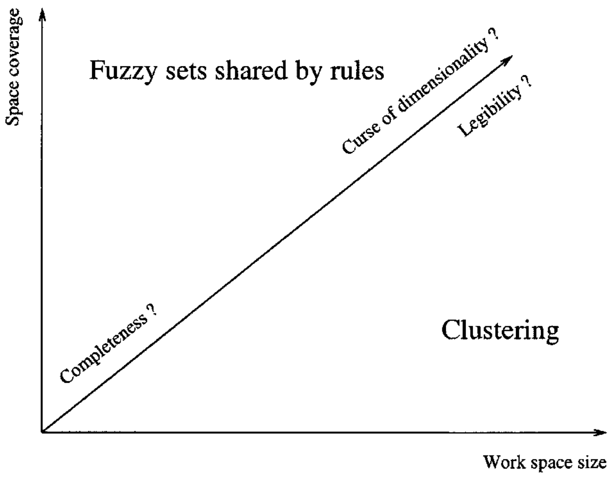


Fig. 5. Choosing a rule induction method according to data characteristics.

The fitness function of the genetic algorithm is not restricted to accuracy performance. It also rewards compact systems, which use a minimum number of input variables and favors incomplete rules.

VI. CONCLUSION

The three rule induction technique families are quite different and may correspond to specific needs. Table I summarizes the most important conclusions.

Fig. 5 shows the applicability of the first two families of methods. Each one is better on one or the other of the plane areas defined by the training set characteristics, work space size, and coverage.

The methods that use shared fuzzy sets for the rule base are appropriate within a small size work space with a good coverage. Otherwise, in case of a weak coverage the rule base completeness is not guaranteed and, when dealing with large systems, the number of combinations to manage is huge.

Clustering is well adapted for large work spaces with a small amount of training examples. However, the induced rule legibility gets worse as the work space size gets larger.

The hybrid methods, including neuro-fuzzy modeling techniques and genetic algorithm based ones, are not easy to locate on the figure. They cannot be viewed as a homogeneous group; all of them are not on the same side. Their performance highly depends on their implementation and particularly on the problem encoding. Thus, their global evaluation remains difficult. The main reason for using such techniques is their universal approximator property. They allow to optimize all the FIS parameters including the membership function parameters. If only guided by numerical accuracy the tuning algorithms may generate an unreadable partitioning. In these conditions, there is not much to be expected in terms of interpretability. Even if the partitioning is carefully respected, other difficulties occur due to the great number of tunable parameters. A new research trend aims to produce a readable set of rules with hybrid methods, by trying to extract the most significant rules.

Once rule induction is done, whatever the technique used, the different parts of the FIS should be optimized to improve the interpretability.

Part II—System Optimization

Historically research teams have been interested in different levels of FIS optimization falling into two main categories: 1) parameter and 2) structure optimization.

Methods for the parameter optimization, membership function fine tuning and rule conclusion optimization, are widely used. Their respective advantages and drawbacks are well known [4], [5].

In this paper, we will focus on structure optimization, input variable selection, and rule base reduction. Defining the FIS using the most useful variables only would benefit to interpretability and stability. Removing extra variables leads to a more compact set of rules and improves the rule interpretability. Moreover, rule base and parameter optimization are easier to achieve once extra variables have been removed. These extra variables are also likely to bring more noise than useful information.

As the available databases are getting larger and larger, FIS will be helpful for the increased needs of knowledge discovery if automatic procedures for variable selection and rule base reduction are included in their design.

VII. Variable Selection

Variable selection can be achieved in a global or in a local way. In the first case, the variable is removed and none of the rules can use it. In the second case, the selection is done at the rule level leading to incomplete rules.

Some of the previously introduced rule induction methods are dealing with variable selection. In a decision tree the path from the root to each leaf node only involves the few variables necessary for defining the associated rule. The genetic algorithm objective function used by Russo [47] aims to minimize the number of variables. Neural networks may be helpful too: the output sensitivity to the input variables can be used to rank the input variables [48], [49].

A. Regularity Criterion

Sugeno, [26], proposed to make the selection using a cross-validation procedure. The training set is randomly split into two groups, A and B, and the criterion to be minimize is

$$\mathrm{RC} = \frac{1}{2}\left[\frac{1}{k_A}\sum_{i=1}^{k_A}(y_i^A - y_i^{AB})^2 + \frac{1}{k_B}\sum_{i=1}^{k_B}(y_i^B - y_i^{BA})^2\right]$$

y_i^A (respectively, y_i^B) is the observed output for pair i of group A (respectively, B), and y_i^{AB} (respectively, y_i^{BA}) is the inferred output, for pair i of group A (respectively, B) after training using group B (respectively, A) sample.

The variables are selected using an ascending procedure. At the first step, p models made of a single variable are considered. The first selected variable is that for which the corresponding model minimizes the regularity criterion. At the second step, $p-1$ models of two variables, the already selected one and each of the remaining candidate ones, have to be assessed. The procedure ends when the criterion increases. The maximum number of models is bounded, $p(p+1)/2$. Even if this number is large, it is still less than the number of all possible combinations, $2^p - 1$.

B. Geometric Criteria

Once the clustering is done, Emami *et al.* [27] obtained the fuzzy sets by projecting the groups onto each input. If a membership function is equal to one on a wide range for a given rule, then the corresponding variable is neutral, one being the neutral element for *and* operators. An index of input nonsignificance for a given rule is defined as the ratio to the entire range of the interval in which its membership function is one. Note that this index is local to a rule. However, the authors use a global combination of the local indices, their product, to make the variable selection for the whole set of rules.

Another static and geometric method was proposed by Lin and Cunningham [50]. Its complexity is linear with respect to the number of inputs and the number of training pairs. Each pair k in each input j is fuzzified as follows:

$$\mu_{jk}(x) = e^{-(x_{jk} - x/b)^2} \qquad b \approx \frac{x_j^{\max} - x_j^{\min}}{10}.$$

A fuzzy rule is associated to each training pair. For each input j, the output of each training pair l is computed as

$$c_{jl} = \frac{\sum_{k=1}^{n}\mu_{jk}(x_{jl})\cdot y_k}{\sum_{k=1}^{n}\mu_{jk}(x_{jl})}.$$

The set of c_{jl} values is the fuzzy curve of input j. Significant input variables are supposed to have a wider range for their fuzzy curves, $(c_j^{\max} - c_j^{\min})$.

The process of the fuzzy curve building is shown in Fig. 6.

The fuzzy curve looks like a kernel estimator projected onto one dimension. These estimators have been thoroughly studied by mathematicians and none of their results is related to the significance of such isolated projections. Moreover, dealing with isolated variables relies on the assumption that they are independent. This assumption is not usually satisfied in real world problems, local contexts being defined by a subset of some interacting variables.

C. Individual Discrimination Power

The originality of the method proposed by Hong and Chen [51] is to make the selection before defining the space parti-

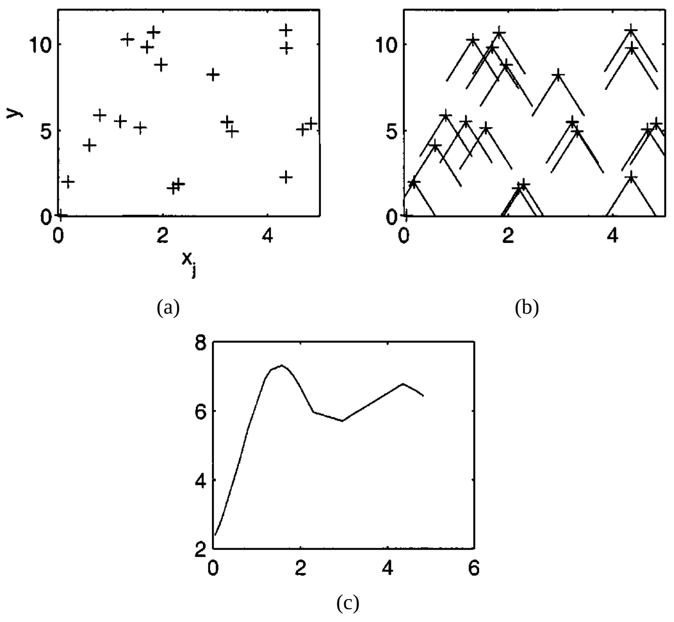


Fig. 6. Fuzzy curve process.

tioning. However, it is restricted to classification problems. It is a five step procedure.

1) Let q_i be the number of unique values for input i: A_{ij}, $j = (1, \ldots, q_i)$.
2) Let n_{ij} be the number of instances whose i input value is A_{ij}. Let n_{ijk}, the number of instances belonging to class k whose value is A_{ij}, $n_{ijk} \leq n_{ij}$. The discrimination power is based on the number of instances for which an input value corresponds to only one class. This number, t_i, is computed as

$$t_i = \sum_j n_{ijk} \qquad \{j = 1, \cdots, q_i | \exists k, n_{ijk} = n_{ij}\}.$$

3) Compute the discrimination power index of each input variable, i. The following two formulae are proposed:

$$f_i = \frac{t_i}{n}$$

$$f_i = 1 - \left\{ -\frac{1}{q_i} \sum_{j=1}^{q_i} \sum_{k=1}^{c} \left[\left(\frac{n_{ijk}}{\sum_{k=1}^{c} n_{ijk}} \right) \log_p \left(\frac{n_{ijk}}{\sum_{k=1}^{c} n_{ijk}} \right) \right] \right\}.$$

The second formula corresponds to an entropy definition, c being the number of classes.

4) Sort the variables in descending order of discrimination power.
5) Select the relevant input variables: the variables are selected in the order mentioned above till the error becomes less than a threshold (0.1 given as an example). The error variable, err, is initialized to 1 and updated as $err = err * (1 - f_i)$ when input i is selected.

This approach also makes the assumption of variable independence, so that their individual contributions are additive.

D. Entropy Variation Index

This approach, proposed by Pal [49], is similar to the previous one; it also deals with classification problems and it implicitly assumes variable independence.

The entropy is a measure of fuzziness. For a given fuzzy set A, it may be expressed as

$$H(A) = \frac{1}{n \log 2} \sum_{i=1}^{n} -\mu_A(x_i) \log(\mu_A(x_i)) - (1 - \mu_A(x_i)) \log(1 - \mu_A(x_i))$$

$H(A)$ reaches a maximum when A is most fuzzy, i.e., when $\mu_A(x_i)$ = 0.5 $\forall i$, and a minimum when $\mu_A(x_i) = 0$ or 1 $\forall i$.

The author uses an S-type function membership for modeling μ, defined as follows on the interval $[a, c]$, b being the crossover point for which the function value is 0.5, $b = (a + c)/2$

$$\mu_A(x_i; a, b, c) = \begin{cases} 0, & x_i \leq a \\ 2\left[\frac{x_i - a}{c - a}\right]^2, & a \leq x_i \leq b \\ 1 - 2\left[\frac{x_i - c}{c - a}\right]^2, & b \leq x_i \leq c \\ 1, & x_i \geq c\ . \end{cases}$$

Let x_{qj} be the values of input variable q for the pairs belonging to class j. Let H_{qj} be the H value of the fuzzy set defined by the following parameters:

$$\begin{cases} b = (x_{qj})_{\text{av}} \\ c = b + \max(|(x_{qj})_{\text{av}} - (x_{qj})_{\max}|, |(x_{qj})_{av} - (x_{qj})_{\min}|) \\ a = 2b - c \end{cases}$$

av, min, and max being, respectively, the average, the minimum, and the maximum value of x_{qj}.

The highest values of H_{qj} are reached when a great number of pairs have a membership degree close to 0.5; that means, when the pairs are grouped in the neighborhood of the average. In other words, H_{qj} varies in the reverse order of the within group variance of input variable q for class j.

If two classes are merged, j and k, and once the parameters a, b and c have been computed according to the new group, the corresponding entropy, H_{qjk}, will be smaller as the average values of the two classes, x_{qj} and x_{qk} are further from each other. In other words, H_{qjk} varies in the reverse order of the between group variance of input variable q for the classes j and k.

Thus, the most discriminant variable for the two classes, is the one that minimizes the variable evaluation index

$$\text{VEI}_q = \frac{H_{qjk}}{H_{qj} + H_{qk}}.$$

TABLE II
VARIABLE SELECTION METHODS

Method[a]	**Technique**	**H**[b]	**Advantage or *drawback***
Regularity criterion *(VII-A)*	Cross validation		Incremental method- *Time consuming*
Genetic Algorithm *(V-B)*	Selection for all the rules or Rule by rule selection		*Poor information*
Geometric criteria *(VII-B)*			
Fuzzy set kernel width	Rule level measurement	●	*Combination of local indices at the rule base level*
Fuzzy curve	Kernel estimator	●	*Criterion interpretation?*
Restricted to classification problems:			
Decision trees *(III-D)*			Incomplete rules
Individual discrimination power *(VII-C)*	All the classes	●	Selection done before partitioning
Entropy variation *(VII-D)*	For pairs of classes	●	*Rule base selection*

[a] The number in parenthesis is the section number which describes the approach.
[b] Variable independence implicit hypothesis.

In order to deal with more than two classes, the following generalization is proposed:

$$\mathrm{OVEI}_q = \frac{\sum_{\substack{j,k=1\\ j\neq k}}^{c} H_{qjk}}{\sum_{j=1}^{c} H_{qj}}.$$

The overall variable evaluation index yields an average: a variable which separates one class from all the others will not be assigned a high value. Its value does not depend on the cardinality of the classes, which can be considered sometimes as an advantage, sometimes as a drawback, depending on the context.

The characteristics of the available variable selection methods are summarized in Table II.

VIII. Rule Base Optimization

Three properties are usually required for the rule base: continuity, consistency and completeness. The continuity guarantees that small variations of the input do not induce big variations for the output. Consistency means that if two or more rules are simultaneously fired their conclusions are coherent. Completeness means that for any possible input vector, at least one rule is fired, there is no inference breaking. When the interpretability is of major importance, it is also necessary to eliminate redundancy.

Some of the previously introduced rule induction methods deal with the rule-base size. The objective function of the genetic algorithm is partly defined by the number of rules. The RBF and the subtractive algorithm tend to minimize the number of generated rules by starting from a small size rule base and incrementally adding rules when needed.

The converse method is also possible: generate a high number of rules, at most one for each training pair, and then reduce the rule base. The rule base reduction methods are also useful when two or more bases are to be merged; for example, an expert knowledge based rule base and some rules induced from data.

Two kinds of techniques are available. The first one consists of merging compatible elements: clusters, fuzzy sets, or variables. The second family of methods is based upon statistic input domain transformation.

A. Merging

Generate a high number of rules using a clustering method makes the resulting partition less sensitive to the initial conditions. Babuska and his co-workers [52], [53], [25], [54] propose an improvement of the compatible cluster merging procedure first introduced by Krishnapuram and Freg [55]. The cluster shape is defined by the eigenvectors (ellipsoid direction) and the corresponding eigenvalues (axis length). The clustering is done using the Gustafson–Kessel algorithm: the distance function uses the covariance matrix.

For a given cluster i, the hyperplane is defined by the following equation $(x - v_i) \cdot \phi_{is} = 0$, where ϕ_{is} is the smallest eigenvalue of cluster i.

The two merging criteria are for clusters i and j.

1) Their hyperplanes are almost parallel: $|\phi_{is} \cdot \phi_{js}| \geq k_1, k_1$ close to one.
2) Their centers are close: $||c_i - c_j|| \leq k_2$, k_2 close to 0.

Two matrices are computed, $C1$ and $C2$. $c1_{ij}$ (respectively $c2_{ij}$) is the degree of similarity of cluster i and j according to the first (respectively, the second) criterion. These values are fuzzified into a two-dimensional (2-D) space so that the ideal candidate coordinates become (1, 1) leading to new matrices $\tilde{C}1$ and $\tilde{C}2$. The two criteria may partially compensate each other. Two clusters whose hyperplanes are not so parallel but whose centers are very close can be merged, conversely. To take this fact into account the criteria are combined into a single matrix using the geometric mean: $\mu_{ij} = \sqrt{\tilde{c}1_{ij} \cdot \tilde{c}2_{ij}}$. These compatibility degrees are then thresholded with a given value (0.7 as an example). Finally, the remaining candidates are merged if they do not contain in their common neighborhood any incompatible cluster. This condition is formalized as

$$\min_{\substack{c_i \in M \\ c_k \notin M}} \max \mathrm{d_{ik}} > \max_{c_i, c_j \in M} \mathrm{d_{ij}}$$

d_{ij} being the distance between cluster i and j in the premise space; the clustering being done in the product space.

Cluster merging is strictly equivalent to rule merging as a rule is associated to a cluster. In another method, also proposed by the researchers of Delft university [56], the elements to be merged are the fuzzy sets, the rule-base reduction being a consequence. The authors highlight three kinds of unwanted similarities between fuzzy sets produced by automatic rule induction: 1) similarity between two fuzzy sets for a given input variable; 2) similarity of a fuzzy set to the universal set U ($\mu_U(x) = 1$ $\forall x \in X$); and 3) similarity of a fuzzy set to a singleton set.

The paper proposes automatic methods to manage the first two types but not for the last one. The corresponding rules may rarely be fired, but this situation may also correspond to exception handling, thus, the removal of close to singleton fuzzy sets has to be confirmed by experts.

An example of a similarity measure between two fuzzy sets, A and B, is

$$S_{(A,B)} = \frac{|A \cap B|}{|A \cup B|}$$

where $|\cdot|$ stands for the fuzzy cardinality and $\cap$ and $\cup$ operators represent the intersection and union, respectively.

The algorithm consists of merging the two most similar fuzzy sets into a new one and then updating the rule base. This operation is repeated until there exist compatible fuzzy sets, those for which the similarity measure is greater than a given threshold. Finally, sets that are close to being universal sets are removed, the closeness being defined by another threshold.

When the fuzzy sets are trapezoidal, a_1, a_2, a_3, a_4 being the parameters for fuzzy set A, the resulting fuzzy set C is defined from $A \cup B$ by

$$\begin{aligned} c_1 &= \min(a_1, b_1) \\ c_2 &= \lambda_2 a_2 + (1 - \lambda_2) b_2 \\ c_3 &= \lambda_3 a_3 + (1 - \lambda_3) b_3 \\ c_4 &= \max(a_4, b_4) \end{aligned}$$

$\lambda_2, \lambda_3 \in [0, 1]$, both set at 0.5 in the example.

The result of the process depends on the thresholds for merging fuzzy sets and for removing universal sets. The interpretability improves as the thresholds get lower.

It is sometimes possible to combine input variables and, thus, to reduce significantly the rule-base size. Before combining the variables, the user has to check if the new variable is still meaningful. Within a control framework, Lacrose [57] combined the error and all its derivatives into a single variable.

The use of multidimensional membership function also leads to a small number of rules. The input space partitioning is done by a Delaunay meshing, i.e., triangulation for a 2-D space. The definition of meaningful multidimensional membership functions may be difficult. Foulloy [58], [59] designed in this way symbolic sensors for color evaluation.

B. Statistic-Based Methods

These methods also initialize a great number of rules, one rule per pair and select the most influential ones using statistic based methods. These methods are powerful and mathematically well established. However, some of them perform an input domain transform which yields a loss of semantic.

1) Orthogonal Least Squares (OLS) Methods: The OLS family [60], [61] makes the selection using a linear regression. To use linear methods for nonlinear optimization the problem must be rewrittem. A FIS can be seen as a two-layer system. First, the input variables are mapped through a nonlinear projection into a new space and second, the output is computed as a linear combination of this new space components. For Wang and Mendel [62], a FIS is a linear combination of fuzzy basis functions (FBF), each of them performing a nonlinear mapping of the input vector.

First, a rule per data pair (Sec. III-C) is generated. The rule i membership function for dimension j is a Gaussian function centered around x_j^i

$$\mu_{A_j^i}(x_j) = a_j^i e^{-(1/2)\left(\left(x_j - x_j^i\right)/\sigma_j^i\right)^2}, \quad \text{with} \quad 0 < a_j^i \leq 1.$$

The inferred output for a given input x is

$$f(x) = \frac{\sum_{i=1}^{r} c^i \left(\prod_{j=1}^{p} \mu_{A_j^i}(x_j) \right)}{\sum_{i=1}^{r} \left(\prod_{j=1}^{p} \mu_{A_j^i}(x_j) \right)}.$$

The FBF, $f_i(x)$, is the relative contribution of rule i for the x example inferred output

$$f_i(x) = \frac{\prod_{j=1}^{p} \mu_{A_j^i}(x_j)}{\sum_{i=1}^{r} \left(\prod_{j=1}^{p} \mu_{A_j^i}(x_j) \right)}.$$

Thus, the fuzzy system can be written as: $\hat{y} = \sum_i f_i(x)\theta_i$, where $\theta_i \in R$ are the scalar parameters to optimize, or $y = F\theta + E$; y being the observed output vector, and E the error.

Each regressor, f_i, is a r*-dimensional* vector, the general term f_{ji} being the firestrength of rule i for pair j.

The OLS learning algorithm transforms the f_i vectors into a set of orthogonal ones using the Gram–Schmidt procedure. The F matrix is decomposed into an orthogonal matrix W and an upper triangular matrix A. The space spanned by the set of orthogonal vectors is the same that spanned by the f_i vectors, so the problem can be written as: $y = Wg + E$. The orthogonal least square solution is $\hat{g}_i = w_i^T y / w_i^T w_i$, $1 \leq i \leq r$. The quantities $\hat{g}$ and $\hat{\theta}$ satisfy the triangular system: $A\hat{\theta} = \hat{g}$. The

TABLE III
RULE BASE OPTIMIZATION METHODS

Family	**Major approaches**[a]	**Technique**	**Advantage or *drawback***
Incremental procedures	Clustering *(IV)*	Rule addition	*Choice of the number of rules*
	RBF *(V-A)*		*Introduction of noise*
	Partition refinement *(III-B.1)*		*into the model.*
Merging *(VIII-A)*	Clusters merging	Heuristic	Less sensitive to initial conditions
	Fuzzy sets merging	Similarity measures	Formally defined fuzzy arithmetic
	Mathematical merging	Variable recombination	*Keep aware of semantic*
	Symbolic merging	Multidimensional fuzzy sets	*Difficult to design*
Statistic based methods	OLS *(VIII-B.1)*	Regression after a non linear projection	Mathematically established (robust)
	SVD *(VIII-B.2)*	Work space size reduction	*Legibility?*
	PCA *(VIII-B.2)*		*Inadvisable*

[a] The number in parenthesis is the section number which describes the approach.

w_i vectors being orthogonal, their individual contributions are additive (no covariance). At each step the algorithm selects the vector w_i, which maximizes the explained variance of the observed output y, i.e., the following criterion

$$[err]_i = \frac{g_i^2 w_i^T w_i}{y^T y}.$$

The algorithm stops when the output has been reconstructed well enough. This occurs at step r_f such as

$$1 - \sum_{i=1}^{rf} [err]_i < \epsilon, \epsilon$$

being a threshold value.

Once the rules have been selected, Hohensohn and Mendel [63] propose to rerun the algorithm with the only objective to optimize rule conclusions, without doing any selection. They note that after the first pass the selected vectors w_i still contain information related to removed rules.

2) Multivariate Data Analysis Based Methods: Multivariate data analysis provides tools for working space reduction, the most popular being the principal component analysis (PCA). These methods are all based on a rectangular matrix property named singular value decomposition (SVD).

The decomposition is written as

$$X = \sum_{i=1}^{l} \sqrt{\lambda_i} u_i v_i' \quad \text{or} \quad X = U \Sigma V^T$$

$l \leq \min(\mathrm{n}, \mathrm{p})$	rank of matrix X;
λ_i	singular values of X sorted in a descending order;
u_i	*n-dimensional* eigenvectors within the row space;
v_i	*p-dimensional* eigenvectors within the column space.

All of them are orthonormal.

Some recent work shows interest in this technique [64], [65]. The model of Yen *et al.* [64] is of the form $Y = Xb$. X is initialized from data pairs like in the former section. The rule i conclusion is computed as $y_i(x) = b_i^0 + b_i^1 x_1 + \cdots + b_i^p x_p$. Thus, the jth line of matrix X contains n blocks, one for each rule. Each block is made of $p + 1$ values corresponding to the jth pair coordinates weighted by the firestrength $w_i(j)$ of each rule i for the jth pair. The values of the rule i block are

$$w_i(j) \quad w_i(j)x_1(j) \quad w_i(j)x_2(j) \quad \ldots \quad w_i(j)x_p(j).$$

The final space size, r, $r \leq \mathrm{rank(X)}$ is determined after checking the singular values. Then the V matrix is partitioned as: $V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$, where V_{11} is a $r \times r$ matrix. Let $\overline{V}^T = [V_{11}^T \;\; V_{21}^T]$. Applying the QR algorithm[7] to $\overline{V}$, Q being a $r \times r$ orthogonal matrix and R_{11} an upper triangular matrix, yields the permutation matrix Π: $Q^T \overline{V}^T \Pi = [R_{11} \;\; R_{12}]$. The first r columns of Π indicate the corresponding fuzzy partitions.

The PCA is used by Kim *et al.* [23] to build new uncorrelated components from the input variables. The rules are initialized by a clustering procedure and the transformation is done within each cluster. For each rule, the covariance matrix is computed, and the rule is defined in the eigenvector space, each eigenvector being a linear combination of the p input variables.

While the merging techniques preserve the semantic, the input domain transform based methods produce rules that cannot be read by an expert, so they are not suited to human cooperation. The characteristics of the rule base reduction methods are summarized in Table III.

IX. CONCLUSION

Many techniques to design FIS from data are available, they all take advantage of the property of FIS to be universal approximators. In order to compare FIS with other modeling techniques, their performance is usually measured by a numerical index, the mean square error. But the blind improvement of the performance may conflict with the originality of fuzzy logic: its

[7]This method is similar to the Gram–Schmidt procedure.

TABLE IV
RULE BASE NEEDS ACCORDING TO FIS APPLICATIONS

Constraint	Application	Semantic	Consistency	Continuity	Completeness
In line	Control	+	+++	+++	+++
	Classification	+	++	+	+++
	Decision support	++	+++	+++	++
Off line	Diagnosis	+++	+++	++	+
	Simulation	+++	+++	+++	+

TABLE V
INTERPRETABILITY OF FUZZY RULE GENERATION METHODS

Family[a]	Interpretability
Rule induction	
Shared partitions *(III-A,III-B.1,III-C)*	High: Fuzzy sets can be interpreted as linguistic labels
Decision trees *(III-D)*	High: Each generated rule only uses a subset of input variables
Clustering *(IV)*	Low: By construction the fuzzy sets are different for each rule, which makes rule comparison and interpretation difficult
Hybrid Methods *(V)*	Low to average: Low in case of membership function parameter tuning, average when rule generation is done with fixed partitioning
Rule base optimization	
Merging *(VIII-A)*	Average to high: Depending on the elements to be merged
Statistic based methods *(VIII-B)*	Poor: Due to the transformed input space

[a] The number in parenthesis is the section number which describes the approach.

interpretability. What are the necessary conditions for a set of induced rules to be interpretable? First, the fuzzy partition must be readable, in the sense that the fuzzy sets can be interpreted as linguistic labels. These labels must be meaningful for experts of the problem under study, so as to allow the rules to be compared to each other, and to lead to knowledge discovery. Second the set of rules must be as small as possible. The reduction of a set of rules results in a loss of numerical performance on the training dataset, but a more compact set has a better generalization capability while being easier to read. For large systems a third condition is required: the rules should be incomplete rules. If the rule premisses involve the whole set of variables, there is a loss of interpretability without a corresponding increase of performance, when the rule context can be defined by a subset of the available variables only. The systematic presence of all variables in all rules can be considered as a drawback of most automatic rule induction methods, due to the techniques themselves. It is not an intrinsic characteristic of the problem.

The interpretability needs depend on the final use of the FIS. Table IV summarizes the main potential applications and the corresponding rule base needs.

Table V compares the main families of rule induction and rule base optimization methods in terms of interpretability. Generally speaking, the methods where all the rules share the same partitioning yield a higher degree of interpretability as they fulfill the first condition stated above. Nevertheless, as shown in Fig. 5, most of these methods become redhibitory for large systems. Indeed the curse of dimensionality prevents the use of methods which generate all the possible rules. The techniques that generate one rule per pair either suffer from an insufficient space coverage or have a great number of data points at their disposal, which also leads to a curse of dimensionality. The only method from that family that at once escapes from that inconvenience and has a good interpretability level is the fuzzy decision tree. Recall, that it needs a prior fuzzy partitioning, which is a bearable constraint when one searches for an interpretable system.

Clustering approaches are very effective in large systems with a low-space coverage. However, as the induced fuzzy sets are different for each rule, this forbids rule comparison and considerably reduces the interpretability.

The third family of methods is characterized by a variable interpretability level due to its heterogeneity. Historically, these methods were not designed with an interpretability concern. Recent work noticeably improved that side.

The second condition to be met for a good interpretability is the reduction of the rule base. The first step is, of course, the variable selection. Other optimization methods can follow; their interpretability level is summarized in Table V. Statistic based methods yield a set of rules difficult to be interpreted as the partitioning is defined onto the transformed input domain. Merging techniques are more suitable for interpretability purposes. However, the interpretability depends on the elements to be merged, it is higher for fuzzy set merging than for cluster merging.

The only approach that deals with the third interpretability condition is the fuzzy decision tree. Most of the available variable selection methods operate in a global way. Unselected variables are completely removed and cannot be used by any rule. Only fuzzy decision trees are able to generate incomplete rules but in the restricted context of classification.

Recent work [66] showed that the selection and simplification can also be done within a rule neighborhood that includes a small group of rules, using reasoning based methods in order to produce reusable knowledge.

The set of procedures able to generate and merge incomplete rules, data induced as well as expert rules, is still an open way of research.

Acknowledgment

The author would like to send special thanks to Brigitte Charnomordic for her powerful and faithful accompaniment throughout this work.

References

[1] L. A. Zadeh, "Fuzzy sets," *Inform. Control*, vol. 8, pp. 338–353, 1965.

[2] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, pp. 1–13, 1975.

[3] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Tran. Syst., Man, Cybern.*, vol. SMC 15, pp. 116–132, 1985.

[4] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice Hall, 1997.

[5] P.-Y. Glorennec, *Algorithmes d'apprentissage pour systèmes d'inférence floue*. Paris, France: Hermès, 1999.

[6] H. Ishibuchi, K. Nozaki, H. Tanaka, Y. Hosaka, and M. Matsuda, "Empirical study on learning in fuzzy systems by rice test analysis," *Fuzzy Sets Syst.*, vol. 64, pp. 129–144, 1994.

[7] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," *Fuzzy Sets Syst.*, vol. 86, pp. 251–270, 1997.

[8] P. Bortolet, "Modelization et commande multivariable floues: Application a la commande d'un moteur thermique," Ph.D. dissertation, Inst. Nat. Sci. Appl., Toulouse, LAAS-CNRS, Dec. 1998.

[9] I. Rojas, H. Pomares, J. Ortega, and A. Prieto, "Self-organized fuzzy system generation from training examples," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 23–26, Feb. 2000.

[10] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, Aug. 1995.

[11] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, Nov./Dec. 1992.

[12] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, Aug. 1986.

[13] H. Ichihashi, T. Shirai, K. Nagasaka, and T. Miyoshi, "Neuro-fuzzy id3: A method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning," *Fuzzy Sets Syst.*, vol. 81, pp. 157–167, 1996.

[14] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.

[15] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, 1973.

[16] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Functions Algorithms*. New York: Plenum , 1981.

[17] T. A. Runkler and J. C. Bezdek, "Alternating cluster estimation: A new tool for clustering and function approxiamtion," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 377–393, Aug. 1999.

[18] R. Krishnapuram and J. M. Keller, "A possibilistic approach to clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 98–110, May 1993.

[19] R. Krishnapuram and J. Kim, "A note on the Gustafson–Kessel and adaptive fuzzy clustering algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 453–461, Aug. 1999.

[20] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proc. IEEE CDC*, San Diego, CA, 1979, pp. 761–766.

[21] R. Hathaway and J. Bezdek, "Switching regression model and fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 195–204, Aug. 1993.

[22] E. Kim, M. Park, S. Ji, and M. Park, "A new approach to fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 328–337, Aug. 1997.

[23] E. Kim, M. Park, S. Kim, and M. Park, "A transformed input-domain approach to fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 596–604, Nov. 1998.

[24] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.

[25] R. Babuska and H. B. Verbruggen, "An overview of fuzzy modeling for control," *Control Eng. Practice*, vol. 4, no. 11, pp. 1593–1606, 1996.

[26] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Aug. 1993.

[27] M. R. Emami, I. B. Türksen, and A. A. Goldenberg, "Development of a systematic methodology of fuzzy logic modeling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 346–361, Aug. 1998.

[28] X. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 841–847, Aug., 1991.

[29] P. A. Burrough, P. F. M. van Gaans, and R. A. MacMillan, "High-resolution landform classification using fuzzy k-means," *Fuzzy Sets Syst.*, vol. 113, no. 1, pp. 37–52, July 2000.

[30] R. R. Yager and D. P. Filev, "Generation of fuzzy rules by mountain clustering," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 209–219, 1994.

[31] M.-S. Chen and S.-W. Wang, "Fuzzy clustering analysis for optimizing fuzzy membership functions," *Fuzzy Sets Syst.*, vol. 103, pp. 239–254, 1999.

[32] R.-P. Li and M. Mukaidono, "Gaussian clustering method based on maximum-fuzzy-entropy interpretation," *Fuzzy Sets Syst.*, vol. 102, pp. 253–258, 1999.

[33] P.-Y. Glorennec, "Un reseau "neuro-flou" evolutif," in *Neuro-Nimes, Fourth Int. Conf. Neural Networks Applicat.*, Nanterre, France, Nov. 1991, EC2.

[34] J.-S. R. Jang, "Anfis: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, 1993.

[35] J. Moody and C. Darken, "Fast learning in networks of locally-tuned process units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.

[36] J.-S. R. Jang and C.-T. Sun, "Functional equivalence between radial basis function network and fuzzy inference systems," *IEEE Trans. Neural Net.*, vol. 4, pp. 156–159, 1993.

[37] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction," *Fuzzy Sets Syst.*, vol. 83, pp. 325–339, 1996.

[38] S. Mitra, R. K. De, and S. K. Pal, "Knowledge-based fuzzy mlp for classification and rule generation," *IEEE Trans. Neural Networks*, vol. 8, pp. 1338–1350, 1997.

[39] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Networks*, vol. 11, pp. 748–768, 2000.

[40] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[41] C. L. Karr, "Design of a cart-pole balancing fuzzy logic controller using a genetic algorithm," in *Conf. Applicat. Artificial Intell.* Bellingham, WA, 1991.

[42] C.-K. Chiang, H.-Y. Chung, and J.-J. Lin, "A self-learning fuzzy logic controller using genetic algorithms with reinforcements," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 460–467, Aug. 1997.

[43] D. Leitch and P. Probert, "New techniques for genetic development of fuzzy controllers," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 112–123, Aug. 1998.

[44] A. Gonzales and R. Perez, "Slave: A genetic learning System Based on an Iterative Approach," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 176–191, Apr. 1999.

[45] C.-C. Wong and S.-M. Her, "A self-generating method for fuzzy systems design," *Fuzzy Sets Syst.*, vol. 103, pp. 13–25, 1999.

[46] F. Guély, R. La, and P. Siarry, "Fuzzy rule base learning through simulated annealing," *Fuzzy Sets Syst.*, vol. 105, pp. 353–363, 1999.

[47] M. Russo, "Fugenesys—A fuzzy genetic neural system for fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 373–388, Aug. 1998.

[48] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayer perceptron," *J. Neural Network Comput.*, vol. 1, pp. 40–48, 1990.

[49] N. R. Pal, "Soft computing for feature analysis," *Fuzzy Sets Syst.*, vol. 103, pp. 201–221, 1999.
[50] Y. Lin and G. A. Cunningham, "A fuzzy approach to input variable identification," in *Proc. IEEE Conf. Fuzzy Syst.*, Orlando, FL, 1994, pp. 2031–2036.
[51] T.-P. Hong and J.-B. Chen, "Finding relevant attributes and membership functions," *Fuzzy Sets Syst.*, vol. 103, pp. 389–404, 1999.
[52] U. Kaymak and R. Babuska, "Compatible cluster merging for fuzzy modeling," in *Proc. Fourth IEEE Int. Conf. Fuzzy Syst.*, Yokohama, Japan, Mar. 1995, pp. 897–904.
[53] R. Babuska and H. B. Verbruggen, "A new identification method for linguistic fuzzy models," in *Proc. Fourth IEEE Int. Conf. Fuzzy Syst.*, Yokohama, Japan, Mar. 1995, pp. 905–912.
[54] R. Babuska, J. A. Roubos, and H. B. Verbruggen, "Identification of mimo systems by input-outputs its fuzzy models," in *Fuzz-IEEE 98*, Anchorage, AK, May 1998, pp. 657–662.
[55] R. Krishnapuram and C.-P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognit.*, vol. 25, no. 4, pp. 385–400, 1992.
[56] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification," *IEEE Trans., Syst., Man, Cybern.*, vol. 28, pp. 376–386, 1998.
[57] V. Lacrose, "Réduction de la complexité des contrôleurs flous: Application a la commande multivariable," Ph.D. dissertation, Inst. Nat. Sci. Appl., Toulouse, LAAS-CNRS, Nov. 1997.
[58] E. Benoit and L. Foulloy, "Exemple de capteur symbolique flou en reconnaissance des couleurs," *RGE*, vol. 3, pp. 22–27, Mar. 1993.
[59] L. Foulloy, S. Galichet, and E. Benoit, "Fuzzy control with fuzzy state sensors," in *EUFIT'94*, Aachen, Germany, Sept. 1994, pp. 1156–1160.
[60] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to nonlinear system identification," *Int. J. Control*, vol. 50, pp. 1873–1896, 1989.
[61] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, Mar. 1991.
[62] L.-X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, 1992.
[63] J. Hohensohn and J. M. Mendel, "Two pass orthogonal least-squares algorithm to train and reduce fuzzy logic systems," in *Proc. IEE Conf. Fuzzy Syst.*, Orlando, FL, 1994, pp. 696–700.
[64] J. Yen, L. Wang, and C. W. Gillepsie, "Improving the interpretability of tsk fuzzy models by combining global learning and local learning," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 530–537, Nov. 1998.
[65] Y. Yam, P. Baranyi, and C.-T. Yang, "Reduction of fuzzy rule base via singular value decomposition," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 120–132, Apr. 1999.
[66] S. Guillaume and B. Charnomordic, "Knowledge discovery for control purposes in food industry databases," *Fuzzy Sets Syst.*, to be published.