A CONSTRUCTIVE METHOD FOR BUILDING FUZZY RULE-BASED SYSTEMS

IGNACIO REQUENA, ARMANDO BLANCO and MIGUEL DELGADO

Dept. Computer Science and Artificial Intelligence
University of Granada, E.T.S.I. Informática
C/ Daniel Aranda Saucedo s.n. 18071 Granada (Spain)
requena@decsai.ugr.es, armando@decsai.ugr.es, mdelgado@ugr.es

Received April 2002
Revised October 2002

This paper proposes a new method for identifying unknown systems with Fuzzy Rule-Based Systems (FRBSs). The method employs different methodologies from the discipline of Soft Computing (Artificial Neural Networks, Fuzzy Clustering) and follows a three-stage process. Firstly, the structure of the FRBS rules is determined using a feature selection process. A fuzzy clustering procedure is then used to establish the number of fuzzy rules. In the third step, the fuzzy membership functions are constructed for the linguistic labels. Finally, the empirical performance of the algorithm is studied by applying it to a number of classification and approximation problems.

Keywords: System identification; fuzzy rule construction; fuzzy approximation; fuzzy modeling.

1. Introduction

Fuzzy Logic was originally introduced as a way of formally describing and manipulating linguistic information [1, 2]. Soon, however, it became clear that it could also be used to model complex systems, where classical mathematical methods failed. This led to a great interest in Fuzzy Rule-Based Systems (FRBSs), which have successfully been applied in control and modeling.

An FRBS has four components: a fuzzification interface, a knowledge base, an inference engine, and a defuzzification interface. The roles of the fuzzification and defuzzification interfaces are reciprocal: while the first converts a real value into a fuzzy value, the second performs the inverse task. The knowledge base holds all the knowledge the system needs to perform its work, and includes the rules describing the relationships between inputs and outputs. Finally, the inference engine is the module which computes the fuzzy outputs corresponding to a set of given inputs.

In order to design an FRBS, the four components must be specified. Regarding the fuzzification and defuzzification interfaces, there is a set of widely known and
used possibilities, from which the ones best suited to the application at hand are selected. Possible choices are direct fuzzification (the real number is considered as a singleton fuzzy number) and Center of Gravity defuzzification [3]. The inference engine is built from the compositional rule of inference proposed by Zadeh [2]. The conjunctive, disjunctive, implication, and aggregation operators must be stated. Finally, the most important part of the system must be designed: the knowledge base. The variables and outputs to be considered are given by the system to be modeled.

In this paper, we propose a three-stage method in order to obtain the fuzzy rules. In the first of these stages, the structure of the different fuzzy rules is obtained; in the second, the number of rules for each structure is determined; and in the last stage, the membership functions are built.

Fuzzy rule-based systems have two basic requirements: interpretability (so that the modeled system behavior may be expressed in a way which the user can understand), and accuracy (so that the modeled system may be represented as precisely as required by the problem being considered). In general, it is difficult for a method to simultaneously achieve both (ideal) goals. The proposed method focuses on a minimum number of rules, but which have sufficient precision, i.e. it focuses on interpretability, but without forgoing accuracy.

Previous ideas and some initial results for the iris problem are showed in [4].

The paper is organized as follows: the second section describes the method in detail; section three empirically analyses its performance; and section four includes final remarks and conclusions.

2. Constructive Method

The framework for the problem considered here is an unknown system with n inputs \( \{x_1, \ldots, x_n\} \), and s outputs \( \{y_1, \ldots, y_s\} \). We will consider a Mamdani-type FRBS [5]. As we have already mentioned, our objective is to build the knowledge base. It is therefore necessary to design both the data and the rule bases.

The system will be modeled with (one or more) MISO-type fuzzy rules (Multiple Input, Single Output), which place no restriction on the kind of system that can be modeled. These rules have the following structure:

\[
R_j: \text{If } x_1^j, \ldots, x_n^j \in A_1^j, \ldots, A_n^j \text{ and } y_j \in B^j \text{, then } y_j \text{ is } B^j. \tag{1}
\]

with \( x_1^j, \ldots, x_n^j \in \{x_1, \ldots, x_n\} \) and \( A_1^j, B^j \) fuzzy sets on the respective domains of variables. As each rule involves only one output, we consider, without lack of generality, the system to have only one output.

The Constructive Method includes the following three steps:

(i) Feature selection: For each output, the most relevant variables are selected, yielding a set of pre-rules.

(ii) Computation of the number of rules for each pre-rule.

(iii) Construction of the fuzzy set membership functions.
2.1. Feature selection

Identifying relevant features is a very important task in the modeling process. This yields great advantages such as a reduction in the complexity of the built model, thereby making it easier to handle and understand. Feature selection is carried out by an extension for continuous values of a method based on Neural Networks (NN) and proposed by Sestito and Dillon [6], which uses only binary data. In short, the method employs two neural networks, which are trained with conveniently modified training sets. An NN is trained with the backpropagation algorithm, using the outputs as additional inputs. The second NN is trained with Hebbian learning, using the complementary of the original inputs as inputs. By combining the weights of both networks, a measure of the relevance of each input to each output is obtained. This measure is then used to rank the inputs' influence on each output variable and to determine which are of real importance and which are irrelevant. The original method was extended so that it could work on continuous valued attributes.

As a result of this feature selection process, pre-rules are obtained. There is only one pre-rule per output, indicating the inputs which most influence that output. These are the only input variables which will appear in the antecedents of the rules describing the output.

A pre-rule indicates the structure of a rule, and has the following form, where the variables \( x^i_1, x^i_2, ... x^i_n \) are the inputs that most influence the output \( y^i \):

\[
x^i_1, x^i_2, ... x^i_n \rightarrow y^i.
\]

The rules obtained from a pre-rule can only contain the variables in the pre-rule.

2.2. Computation of the number of rules

Feature selection yields the structure of the rules associated to a given output, but it does not state how many there are nor which they are. The number of rules will be obtained by a fuzzy clustering algorithm. The clustering will not be applied on the original data, but on the data which result from projecting according to the pre-rule. So if, for instance, the following pre-rule was obtained for output \( y_1 \):

\[
x_1, x_3, x_6 \rightarrow y_1.
\]

then every sample \( \bar{x}_i \) in the data set is transformed into \( \bar{x}'_i = (x_{i1}, x_{i3}, x_{i6}, y_{i1}) \), and the clusters are formed from these samples. The Fuzzy Kohonen Clustering Network [7] or the Fuzzy C-Means algorithm [8] is used to obtain the fuzzy clusters. These methods must be initialized with the number of clusters and the initial centers for them. Initialization is carried out using Chiu's method [9, 10].

Chiu's proposal for fuzzy clustering initialization is a variant of the mountain clustering algorithm [11]. The procedure uses four prefixed parameters: \( r_b \) set the
distance between two possible centers, \( r'_a \) set the radius of the cluster, \( \varepsilon \) and \( \varepsilon' \) set the stopping criterion (the distance between two centers is \( > \varepsilon \) and the distance of each sample to one of the centers is \( < \varepsilon \)).

The numeric variables are normalized in the interval \([0,1]\) and every possible value of the non-numeric variables is assigned a numeric value in the unit interval, preserving the order if necessary (in qualitative variables, where the values have no order, we can use the alphabetic order, or simpler, the order in which these values are first written, as we have used here).

As the metric depends on the number of components and this is not equal for every pre-rule, but depends on the number of selected relevant inputs, \( \nu \), the effective clustering radius will be \( r_a = \nu r'_a \), where \( r'_a \) is the same for all variables.

The clustering induces a fuzzy pseudopartition of data in \( c \) clusters characterized by their centers \( \vec{v}_k \), \( k=1, \ldots, c \). Every sample \( x_i \) from the data belongs to the \( k \)-th cluster with degree \( u_{ik} \). There are different ways of defining this membership degree. The following have been considered in this paper (\( \| \cdot \| \) represents a norm in the data space):

- **Gaussian membership function:**
  \[
  u_{ik}^{(a)} = e^{-\phi\|\vec{x}_i - \vec{v}_k\|^2} .
  \]
  \( \phi \) is a positive constant. Both \( \phi \) and \( \| \cdot \| \) are related to those used in the clustering, as described in [10] (\( \phi = r_a^2/4 \) and \( \| \cdot \| \) is the same as that used in the potential computation).

- **Fuzzy C-Means membership function:**
  \[
  u_{ik}^{(k)} = \left[ \sum_{j=1}^{c} \left( \frac{\left\| x_i - \bar{v}_k \right\|^2}{\left\| x_i - \bar{v}_j \right\|^2} \right)^{\frac{1}{s-1}} \right]^{-1} .
  \]
  where \( s \in (1, \infty) \) regulates the function fuzziness degree. The closer this is to 1, the crisper the partition becomes.

- **Distance reciprocal membership function:**
  \[
  u_{ik}^{(c)} = \frac{1}{1 + \left\| x_i - \bar{v}_k \right\|^2} .
  \]

- **Nearest center membership function:**
  \[
  u_{ik}^{(c)} = \begin{cases} 
  1 & \text{if } \left\| x_i - \bar{v}_k \right\| = \min_j \left\| x_i - \bar{v}_j \right\| \\
  0 & \text{otherwise}
  \end{cases} .
  \]
For each definition, the membership function for the k-th cluster verifies these two properties:

- The center belongs completely to its group: \( \mu_k(\overline{v}_k) = 1 \).
- Monotony with respect to the distance:

\[
\forall \overline{v}_k, \overline{x}_i, \overline{x}_j, \quad \|\overline{x}_i - \overline{v}_k\| < \|\overline{x}_j - \overline{v}_k\| \quad \Rightarrow \quad \mu_k(\overline{x}_i) \geq \mu_k(\overline{x}_j).
\]

2.3. Determination of the semantics for linguistic labels

In order to complete the fuzzy rule definition, it only remains to indicate the membership functions for the fuzzy sets \( A^j, B^j \). Each cluster identified in the previous step corresponds to a rule. In order to build the fuzzy sets for a rule, samples belonging to that group are employed.

The idea is that every fuzzy cluster induces a fuzzy set for each sample component, and that the label which will be assigned to each variable in the rule will be an approximation of the fuzzy set induced by the cluster. For instance, let us consider the first cluster from the clustering for the pre-rule (2), and that there are \( n_1 \) samples in this cluster:

\[
\{(\overline{x}_i, u_{i1}), i=1,\ldots,n_1\}.
\] (8)

This cluster induces 4 fuzzy sets: one on the domain of each component \( x_1, x_3, x_6 \) and \( y_1 \). The fuzzy set on the \( x_1 \) component is:

\[
\{(x_{i1}, u_{i1}), i=1,\ldots,n_1\}.
\] (9)

Henceforth, the label \( A_1 \) for the rule built from this cluster will be an approximation of (9). The specific process will depend on the type of fuzzy set chosen. For reasons of simplicity and efficiency, we will restrict ourselves here to trapezoidal fuzzy sets, whose membership function is usually characterized by four parameters, as depicted in fig. 1.

```
Figure 1. Trapezoidal fuzzy set A (a, b, c, d)
```

In order to construct the fuzzy sets, a procedure based on simple statistical measures has been designed. Let’s consider the values \( x_{ij} \) of variable \( x_j \) within the fuzzy cluster (k-th) corresponding to the rule we are building. Let the k-th cluster obtained contain \( n_k \) samples, and \( \{x_{ijk}, i=1,\ldots,n_k\} \) be the values for variable \( x_j \) of the samples in the k-cluster.

This set of values is considered as a statistical distribution by assigning a frequency to every value. The frequency assigned to value \( x_{ijk} \) is proportional to the
membership degree of the sample to the cluster. If a concrete value is repeated (e.g., \( x'_{ijk} = x_{ijk} \)), we consider the greatest \( f_{ijk} \) assigned. This is expressed by (\( r \) goes round all of the values \( x'_{ijk} \) which are equal to \( x_{ijk} \)):

\[
 f_{ijk} = \max_{r, x'_{ik} = x_{ijk}} \left( \frac{u_{rk}}{\sum_{s=1}^{n_k} u_{sk}} \right) . 
\]

(10)

We therefore have a statistical distribution, and the fuzzy set parameters are:

- The support \((a, d)\):
  
  Let \( a' = \min_i \{x'_{ijk}\} \) and \( d' = \max_i \{x'_{ijk}\} \), and to prevent bound values from having a null membership degree, we consider a support extension parameter \( \eta \) (\( 0.05 < \eta < 0.3 \)). Then, the support is

\[
 a = a' - \eta \ (d' - a') \quad ; \quad d = d' + \eta \ (d-a) .
\]

(11)

If \((a', d')\) is considered to be the support, it will be possible for the test set to frequently contain values outside this domain (then the membership degree is 0), or, in real problems, new values (some of which are possibly outside this domain) can appear (we can say that the training set is not sufficiently representative). It is therefore necessary to extend the support, and \( \eta \) must be set according to the knowledge the user has about the domain of the variable.

- The core \((b, c)\):
  
  The sorted values \( x_{ij} \) are considered as a statistical distribution, and then:

\[
 b = \text{quartile 1} \quad ; \quad c = \text{quartile 3} .
\]

(12)

2.4. Membership function construction to \( \alpha \) level

In this section, we propose a different approach to building the membership functions. It is raised according the following consideration: membership functions are built with data belonging to the cluster with a minimum degree of \( \alpha \), hence the bounds in every component should have a membership degree of \( \alpha \). This is described graphically in fig. 2.

![Figure 2. Trapezoidal fuzzy set A(a, b, c, d) built at \( \alpha \) level](image)

In order to calculate the bounds of support, \( a \) and \( d \), it is only necessary to extrapolate the lines defined by the core bounds and the component value set bounds
(b and c are obtained as in section 2.3).

- The parameter \( a \) is the abscissa of the point at which the line passing through the points \((\text{min, } \alpha)\) and \((b, 1)\) cuts the abscissas axis:

\[
a = \min - \alpha \frac{b - \min}{1 - \alpha}.
\]

- The parameter \( d \) is the abscissa of the cut point of the x axis and the line defined by \((c, 1)\) and \((\text{max, } \alpha)\):

\[
d = \max + \alpha \frac{\max - c}{1 - \alpha}.
\]

Membership functions built in this way perform better in approximation problems than in classification problems, when the attributes are discrete-valued (they usually appear in classification tasks) and the difference between the bound and the closest quartile is very large.

The three steps of the Constructive Method have already been detailed and are summarized in the flowchart below.

**Constructive Method in brief**

I. Identify relevant variables for each output by a feature selection method. A pre-rule represents the association inputs \(\rightarrow\) output:

\[x_1^h, \ldots, x_{mh}^h \rightarrow y^h\]

II. For each pre-rule \(h\):

A) Transform the data set by projecting samples according to the variables involved.

B) Identify the number of meaningful clusters (let be \(p\)) with Chiu's algorithm. This yields an initial fuzzy clustering.

C) Optimize the clustering with Fuzzy C-Means or Kohonen Fuzzy Clustering Net. The clusters are fuzzy sets as in (15)

D) For every identified fuzzy cluster, \(k = 1, \ldots, p\):

i) For every data component in the pre-rule, \(j = 1, \ldots, m_h\), and the output:

a) Construct the fuzzy set corresponding to the projection on the \(j\)-th component of the \(k\)-th fuzzy cluster:

\[\{x'_{ijk}, u_{ik}\}, i = 1, \ldots, n_k\] (15).

b) Associate frequencies \(f_{ijk}\) to \(x'_{ijk}\), with (10)

c) Compute the trapezoidal fuzzy set \(A_{jk}^h\) (\(B_{jk}^h\) for the output) using the statistical distribution, according to: (11) (or (13,14)) and (12).

ii) Add to the fuzzy rule base, the rule:

If \(x_1^h\) is \(A_1^h\) \ldots and \(x_{mh}^h\) is \(A_{mh}^h\), then \(y^h\) is \(B^h\). (16)
2.5. Adaptation to classification problems

The Constructive Method is general and may be applied to system identification in a very general framework, and this includes classifiers. The method could, however, perform better if advantage were taken of its peculiar features.

Classifiers may be reproduced by FRBSs according to a pair of basic schemes:

(i) Classification with crisp output. Class boundaries may be blurred, but a given object only belongs to a single class. Classical classifiers are included as particular instances of this type. Used fuzzy rules have the form:

\[ \text{If } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \ldots \text{ and } x_n \text{ is } A_n \text{, then } c_{l_j} \]  

(17)

Each rule characterizes a subgroup of the class \( c_{l_j} \), or may be the whole class. This FRBS inference mechanism is fairly simple: each rule in the base is fired and the output is the class of the rule with the highest firing level.

(ii) Classification with fuzzy output. This approach offers total flexibility, allowing situations to be modeled in which a given object belongs to more than one class with different degrees of membership. Fuzzy rules adopt the following form:

\[ \text{If } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \ldots \text{ and } x_n \text{ is } A_n \text{, then } c_{l_j} \text{ is } B_j \]  

(18)

where \( B_j \) is a fuzzy set on \([0, 1]\) , which gives a fuzzy estimation on the degree to which the fuzzy rule describes objects of class \( c_{l_j} \).

The peculiarity of classification problems (something which approximation problems do not have) lies in the fact that an initial clustering is prepared. Using MISO rules we can then obtain independent descriptions of each class. In this way, the construction of rules may be driven by samples from a single class.

This may be used to adapt the Constructive Method as follows: Every pre-rule indicates which variables are necessary to characterize a given class. Hence, the clustering is not carried out on all the data, but only on those of the class being modeled. Moreover, in order to compute the effective clustering radius \( r_a \) the output is not taken into account. If pre-rule (2) were for a classifier, its effective clustering radius would be

\[ r_a = 3 \cdot r_a' \quad \text{instead of} \quad r_n = 4 \cdot r_a' \].

Finally, when constructing rules of type (17), there are no output labels to be built.

2.6. Constructive method in action

When applying the Constructive method, some preliminary experimentation must be carried out by trying different values for the clustering process parameters. The clustering radius \( r_a \) has the highest influence on the resulting clustering. Once a small number of representative values for \( r_a \) have been found, they are combined with the four possible cluster membership functions, \( h_k \).
3. Empirical Evaluation

In order to validate the Constructive Method and test its performance, an empirical study has been carried out by applying the method to a set of problems.

We have selected problems commonly found in benchmarks employed by researchers, and whose descriptions and datasets are available on Internet. A real-world problem has also been included. We have experimented with the following problems:

- Classification problems:
  - Iris plants
  - Wine
  - Diabetes diagnosis: PIMA
- Approximation problems:
  - Function approximation
  - Apple purchase policy

3.1. Experimental design

Conclusions drawn from experimental studies are only meaningful if these are conducted rigorously. This is why the experimental set up was carefully designed. All the experiments described here follow the same schedule.

The initial data set is randomly divided into two parts: a training part, including 2/3 of the whole set, and a test set including 1/3 of the whole set. These are the proportions in which the UCI's (University of California, Irvine) machine learning repository datasets are organized. We respected this criterion and adopted it as a general convention. The selection is uniformly randomly performed according to the implementation described in [12]. There is no per class sampling nor selective sampling. The training set is used during the rule construction stages: training of neural nets, cluster identification, and construction of the membership functions.

After a rule base has been obtained, its performance is measured by contrasting the outputs the system gives to the test samples against the actual outputs. For classification problems, the percentage of correct classification is used as a quality measure. A more detailed description of the behavior is available with the confusion matrix (a squared matrix with dimension size the number of classes) $\mathbf{M} = (m_{ij})$, such that $m_{ij}$ indicates the number of samples of class $i$ which have been classified as being of class $j$. Obviously, the more diagonal the matrix, the better the rule base.

For approximation problems, the measure is the Mean Square Error:

$$MSE = \frac{1}{P} \sum_{p=1}^{P} (o_p - y_p)^2 .$$

with $o_p$ the expected output to the $p$-th datum, $y_p$ the output calculated by the FRBS, and $P$ the number of samples.
The randomness is present in the building process and comes from two main sources. On one hand, the data set partition (in training and test parts) is random and could greatly influence the results. On the other hand, there are some steps in the method where an element of chance exists, such as in the neural nets initialization.

In order to offer sufficiently robust results, it is convenient to carry out a number of executions. To contrast with the first source of randomness, the cross validation technique is employed. Cross validation implies making different partitions and applying the method on each partition. In the experiments, 10 partitions over every problem have been considered. So as to face the second random source, once the different parameters have been fixed, 10 executions are carried out.

Experimental results are showed in the tables, indicating the number of rules, and the errors in training and the test sets. We would like to mention that these results were obtained without a rule tuning process.

### 3.2. Classification problems

#### 3.2.1. Iris plants

The first problem considered is the widely-known *iris* plant problem. The dataset comprises 150 samples, each characterized by four continuous attributes: petal-length, sepal-length, petal-width, and sepal-width. There are three classes: setosa, virginica, and versicolor, with 50 samples of each within the data set. We applied the constructive method to this problem, following the established steps. As each class corresponds to an output, the feature selection step yielded the following dependencies (pre-rules):

- petal-length $\rightarrow$ setosa
- sepal-width, petal-length, petal-width $\rightarrow$ versicolor
- petal-length, petal-width $\rightarrow$ virginica

The next step is to determine the number of rules for each pre-rule. This is achieved using a fuzzy clustering process on the data resulting from the projection of original data samples on the variables indicated by the feature selection process. In the classification adapted version of the method, the clustering is performed only on data from a single class. Therefore, the clustering associated to the first pre-rule is only carried out on training data samples belonging to the setosa class. Similarly, for the second and third pre-rules, only data from classes versicolor and virginica, respectively, were employed (previous results were showed in [4]).

The definitive selection of parameters for the clustering was:

<table>
<thead>
<tr>
<th>$r_a' = 0.07$</th>
<th>$r_b = 1.5 \times r_a$</th>
<th>$\varepsilon = 0.5$</th>
<th>$\varepsilon = 0.15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{ik} = u_{ik}^{(b)}$, FCM mf. (5)</td>
<td>$\alpha = 0.5$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The last step was to build fuzzy sets for each variable. In order to keep a reduced number of rules and guarantee a good cover of data, the support of fuzzy sets had to be widened to 50%, so the value of $\eta$ was set to 0.25.

The initial set of 150 samples was randomly divided 10 times into partitions comprising a training set (100 samples) and a test set (50 samples). The method was applied 10 times, producing the average results indicated in Table 1. This table includes the number of rules obtained (N.R.) along with the Percentage of Correctly Classified samples in the training set ($PCC_{\text{Train}}$) and in the test set ($PCC_{\text{Test}}$), and their means. In several partitions, the best run results only have 3-4 errors on the 150 samples (training + test), which indicate a less than 2% error. The samples not learned are the same in all partitions, independently they belong to the training or test sets.

<table>
<thead>
<tr>
<th>Partition</th>
<th>N.R.</th>
<th>$PCC_{\text{Train}}$</th>
<th>$PCC_{\text{Test}}$</th>
<th>No. Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>94.0</td>
<td>100.0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>97.0</td>
<td>98.0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>96.0</td>
<td>98.0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>99.0</td>
<td>90.0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>96.0</td>
<td>94.0</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>98.0</td>
<td>94.0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>95.0</td>
<td>100.0</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>97.0</td>
<td>96.0</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>97.0</td>
<td>96.0</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>97.0</td>
<td>94.0</td>
<td>6</td>
</tr>
<tr>
<td>Average</td>
<td>3</td>
<td>96.6</td>
<td>96.0</td>
<td>5.4</td>
</tr>
</tbody>
</table>

As an example, here is one of the rule bases constructed for partition number 1:

(i) If petal-length is (0.73, 1.40, 1.50, 2.17), then setosa
(ii) If sepal-width is (1.84, 2.6, 3.0, 3.76) and petal-length is (2.76, 4.0, 4.7, 5.64) and petal-width is (0.76, 1.20, 1.50, 2.04), then versicolor
(iii) If petal-length is (4.17, 5.1, 5.97, 7.53) and petal-width is (1.07, 1.80, 2.30, 2.83) then virginica

Confusion matrices on training ($M_{\text{TR}}$) and test ($M_{\text{TE}}$) sets for this rule base are (matrices are similar for other rule bases; confusions are always between classes 2-3):

$$
M_{\text{TR}} = \begin{bmatrix}
30 & 0 & 0 \\
0 & 36 & 3 \\
0 & 0 & 31
\end{bmatrix} \quad M_{\text{TE}} = \begin{bmatrix}
20 & 0 & 0 \\
0 & 11 & 0 \\
0 & 1 & 18
\end{bmatrix}
$$

(20)

The results obtained are good and in line with those reached by other methods. For example in [13, 14], the best overall accuracy reported is 95.8%. Halgamuge et
al. [15] need 13 rules to reach a 99% of hits. With a more compact representation, the percentage is reduced to 96%. Nauk et al. [16] report 96% and 97.3% in the training and test sets, respectively. However, in all of these papers, the result is only for one run, they do not offer the average results of different independent runs.

Average results on several runs are showed in other papers, f. e., [17,18,19,20]. The best (several parameter configurations) average results, using different methodologies, are similar to our results: In [17], there is 94.67% success with 3 rules and 98.67% with 6 rules (using all the samples, that is to say, without test to obtain the rules); in [18], 96.22% with 70 rules and after tuning, 96.71% with 48 rules; in [19], 98.58% (train) 96.22% (test) and after tuning, 99.65% (train) – 96.710 (test) (there is no indication of the number of rules); and in [20], 96.6% (on the test set with 20% samples) with 13 rules, and 96.64% with 4.6 rules after a rule reduction process.

The advantage of the Constructive Method may be summarized as follows:
(i) The number of rules is the smallest possible: only one rule per class.
(ii) The results, without tuning, are along the lines of (or better than) other published methods.
(iii) The computational effort is sufficiently small. Each running of the process was performed on a PC (Intel Pentium 400 MHz) in less than 20 seconds.

3.2.2. Wine classification

The goal of this problem is to classify wines. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 178 samples, with 59, 71, 48 for classes 1, 2, 3, respectively.

Observing the criterion of splitting the data set into 2/3 (training) and 1/3 (test), 10 different partitions are randomly generated with 120 and 58 samples.

The Constructive Method allowed small rule bases to be obtained with very good accuracy. The results averaged over the 10 executions are displayed in Table 2 (in [18] the best results on several configurations are 98.38 (train) and 92.97 (test)).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>N.R.</th>
<th>PCC_{Train}</th>
<th>PCC_{Test}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_k = u_k^{(0)}) , (r_k = 1.0) , (\eta = 0.05)</td>
<td>3</td>
<td>96.67</td>
<td>67.24</td>
</tr>
<tr>
<td>(u_k = u_k^{(0)}) , (r_k = 1.0) , (\eta = 0.20)</td>
<td>3</td>
<td>98.33</td>
<td>91.38</td>
</tr>
<tr>
<td>(u_k = u_k^{(0)}) , (r_k = 1.0) , (\eta = 0.25)</td>
<td>3</td>
<td>97.50</td>
<td>94.83</td>
</tr>
</tbody>
</table>

This problem has been quite easy to solve with a minimum set of rules. The main handicap (\(\eta = 0.05\)) was the lack of representativity of certain training sets with respect to the original domain of attributes. This is why the best results are achieved by widening the label membership functions by enlarging parameter \(\eta\).
3.2.3. Diabetes diagnostic

This problem is aimed at diagnosing diabetes in human beings. This is a real problem\(^1\) with data gathered from a population of Pima Indian women. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e. if the 2-hour post-load plasma glucose was at least 200 mg/dl in any survey examination or if found during routine medical care).

The data set comprises 768 samples, each described by 8 attributes besides the binary-valued class. There are 500 and 258 instances in classes 0 and 1, respectively. In the experiments we have conducted, the training set included 512 samples and the remaining 256 went to the test set.

The feature selection process yields the following pre-rules:

\[
x_2, x_3, x_6 \rightarrow \text{class 0}
\]

\[
x_2, x_3, x_6 \rightarrow \text{class 1}
\]

Averaged results of the application of the Constructive Method to this problem are displayed in Table 3. As the best overall performance, we should mention a base with 2 rules which achieved 74\% success on the training set and 76.95\% on the test set. The results are similar for several values of \( \eta \). In [18], an 81.91\% (train) – 74.68 (test) of success are indicated; in [19], after a tuning process, 83.27\% (train) – 75.81\% (test) are showed, and in [20], 68.25\% (test) with 128 rules and 66.20\% with 18 rules.

As is obvious from Table 3, the Constructive Method was not able to directly achieve a good identification on this problem. Nevertheless, in each case, the best performance achieved more than 74\% success over the 768 samples (training + test sets). We performed a deeper analysis of wrong cases and reached the conclusion that the simple way of constructing the membership functions is not suitable directly for every problem, but a tuning process (which is usual) must be enough\(^2\).

<table>
<thead>
<tr>
<th>( t^* )</th>
<th>N.R.</th>
<th>PCC(_{\text{Train}})</th>
<th>PCC(_{\text{Test}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>2.0</td>
<td>71.13</td>
<td>70.70</td>
</tr>
<tr>
<td>0.06</td>
<td>3.0</td>
<td>71.13</td>
<td>70.59</td>
</tr>
<tr>
<td>0.05</td>
<td>5.3</td>
<td>71.11</td>
<td>70.66</td>
</tr>
<tr>
<td>0.04</td>
<td>12.3</td>
<td>70.92</td>
<td>70.35</td>
</tr>
<tr>
<td>0.03</td>
<td>27.3</td>
<td>70.88</td>
<td>70.43</td>
</tr>
</tbody>
</table>

\(^1\) Data were gathered by researchers at the Nati. Institute of Diabetes and Digestive and Kidney Diseases.

\(^2\) With FRUTSA (J.M. Benitez, J. L. Castro, I. Requena. FRUTSA: Fuzzy Rule Tuning by Simulated Annealing. Submitted to IJARS) the average results (10 runs of FRUTSA) achieved 78.20\% and 76.40\% success on training and test sets, respectively, for a rule base with 3 rules and 70.89\%, 71.48\% success. For a rule base with 13 rules, and 68.75\% (training), 73.82\% (test), after FRUTSA application, the average results were 83.20\% (training) and 78.20 (test).
3.3. Approximation problems

3.3.1. Function approximation

In order to test the effectiveness of the Constructive Method in approximation problems, we decided to apply it to the modeling of two simple functions. The selected functions were:

- \( f_1(x) = \sin(x), \quad x \in [0,1] \)
- \( f_2(x) = x^2, \quad x \in [0,1] \)

The datasets were built by sampling the functions in 1000 points distributed uniformly on the domain [0, 1]. Each dataset is randomly partitioned devoting 700 points for the training set and 300 points for the test set.

Once again, the quality of modeling depends on the number of rules, which is regulated by changing the value of \( r_s \). This fact can be checked by inspecting the tables with results. Another point to mention is that the building of labels to an \( \alpha \) level - extrapolating the support bounds - yields better results. This can be inferred by contrasting Table 4 with Table 5, both corresponding to the function \( f_1(x) \) and, from the comparison of Tables 6 and 7, which corresponds to \( f_2(x) \).

**Table 4** Constructive Method applied to \( f_1(x) = \sin(x) \)

(1) Membership function building based on frequencies.

<table>
<thead>
<tr>
<th>( r_s )</th>
<th>N.R.</th>
<th>( \text{MSE}_{\text{Train}} )</th>
<th>( \text{MSE}_{\text{Test}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>2.0</td>
<td>0.1031</td>
<td>0.1035</td>
</tr>
<tr>
<td>0.15</td>
<td>3.4</td>
<td>0.0621</td>
<td>0.0630</td>
</tr>
<tr>
<td>0.10</td>
<td>6.0</td>
<td>0.0369</td>
<td>0.0405</td>
</tr>
<tr>
<td>0.05</td>
<td>12.1</td>
<td>0.0293</td>
<td>0.0357</td>
</tr>
</tbody>
</table>

**Table 5** Constructive Method applied to \( f_1(x) = \sin(x) \)

(II) Membership function building to level \( \alpha \).

<table>
<thead>
<tr>
<th>( r_s )</th>
<th>N.R.</th>
<th>( \text{MSE}_{\text{Train}} )</th>
<th>( \text{MSE}_{\text{Test}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>2.0</td>
<td>0.0805</td>
<td>0.0803</td>
</tr>
<tr>
<td>0.15</td>
<td>3.4</td>
<td>0.0425</td>
<td>0.0426</td>
</tr>
<tr>
<td>0.10</td>
<td>6.0</td>
<td>0.0229</td>
<td>0.0236</td>
</tr>
<tr>
<td>0.05</td>
<td>12.1</td>
<td>0.0117</td>
<td>0.0120</td>
</tr>
</tbody>
</table>

**Table 6** Constructive Method applied to \( f_2(x) = x^2 \)

(I) Membership function building based in frequencies.

<table>
<thead>
<tr>
<th>( r_s )</th>
<th>N.R.</th>
<th>( \text{MSE}_{\text{Train}} )</th>
<th>( \text{MSE}_{\text{Test}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>2.0</td>
<td>0.1187</td>
<td>0.1152</td>
</tr>
<tr>
<td>0.15</td>
<td>3.5</td>
<td>0.0717</td>
<td>0.0719</td>
</tr>
<tr>
<td>0.10</td>
<td>6.0</td>
<td>0.0425</td>
<td>0.0444</td>
</tr>
<tr>
<td>0.05</td>
<td>12.6</td>
<td>0.0278</td>
<td>0.0332</td>
</tr>
</tbody>
</table>
Table 7 Constructive Method for $f_2(x) = x^2$

<table>
<thead>
<tr>
<th>$r^*$</th>
<th>N.R.</th>
<th>MSE(_{\text{Train}})</th>
<th>MSE(_{\text{Test}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30</td>
<td>2.0</td>
<td>0.1091</td>
<td>0.1070</td>
</tr>
<tr>
<td>0.15</td>
<td>3.5</td>
<td>0.0575</td>
<td>0.0561</td>
</tr>
<tr>
<td>0.10</td>
<td>6.0</td>
<td>0.0287</td>
<td>0.0286</td>
</tr>
<tr>
<td>0.05</td>
<td>12.6</td>
<td>0.0124</td>
<td>0.0230</td>
</tr>
</tbody>
</table>

3.3.2. Apple purchase policy

This is a real-world problem. The aim is to obtain the purchase policy for a company which uses apples as its raw material. The firm works with several suppliers with which orders are placed. By considering several features of the product and quality of service, the firm decides how many apples to buy from each supplier.

The data set comprises 625 samples, each featuring 5 attributes:

- Price: Expressed in monetary units (e.g., Euros) within the domain $[0,100]$.
- Weight: Average weight per apple, measured in grams (ranging from 0 to 80).
- Sugar level: Percentage of sugar content.
- Delivery delay: Measured in days with minimum and maximum values of 0, 10.
- Output: Percentage of purchase to order.

According to the experimental set up, the data set is randomly sampled to select training and test sets with 416 and 209 samples respectively. After the feature selection, all the variables remained relevant, so no simplification of rules took place here.

The average results of 10 runs are showed in Table 8. The membership degrees were obtained according to the nearest center, equation (7). Results obtained with Gaussian or FCM membership functions are quite similar to those in Table 8. Notwithstanding, when using either of these two cluster membership functions, the role of the $\alpha$ parameter, which regulates the data set employed in constructing the memberships of the labels, is very important. As the value of the radius is decreased, the value of $\alpha$ should also be decreased to counteract its effect since the reduction in the radius value implies both accepting a higher number of clusters and smaller clusters resulting in a reduced number of data employed in the membership function construction of the labels.

Table 8. Constructive Method applied to Apple Purchase Policy.

<table>
<thead>
<tr>
<th>$r^*$</th>
<th>$\alpha$</th>
<th>N.R.</th>
<th>MSE(_{\text{Train}})</th>
<th>MSE(_{\text{Test}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>0.25</td>
<td>1.0</td>
<td>24.24</td>
<td>24.88</td>
</tr>
<tr>
<td>0.15</td>
<td>0.15</td>
<td>2.2</td>
<td>16.40</td>
<td>16.73</td>
</tr>
<tr>
<td>0.10</td>
<td>0.13</td>
<td>14.4</td>
<td>8.93</td>
<td>9.64</td>
</tr>
<tr>
<td>0.09</td>
<td>0.12</td>
<td>21.7</td>
<td>6.77</td>
<td>7.48</td>
</tr>
<tr>
<td>0.08</td>
<td>0.11</td>
<td>33.8</td>
<td>5.77</td>
<td>7.34</td>
</tr>
<tr>
<td>0.07</td>
<td>0.10</td>
<td>55.2</td>
<td>3.24</td>
<td>6.00</td>
</tr>
</tbody>
</table>
4. Conclusions

A new procedure called the Constructive Method for building FRBSs has been proposed. There are three stages to this method. Firstly, the structure of the rules (pre-rules) is established using a feature selection process. We have performed an extension for continue data of a method for binary data [6]

The number of rules is then determined with a clustering algorithm. Our contribution has been the idea of obtaining the clusters with only the data involved in the pre-rule, by projecting the initial samples on the pre-rules.

Finally, the membership functions for the fuzzy sets involved in the fuzzy rules are constructed. By considering data as a statistical distribution, two approaches for building the membership functions have been proposed. This approach does not need the membership functions of the values of each variable to be prefixed, since the process builds them directly, using a simple original procedure.

The method has been tested on a set of problems including both classification and approximation problems. There have been selected problems usually found in benchmarks and a real-world problem as well. In general, the accuracy and generalization reached by the rule bases generated with the Constructive Method are good. They usually achieved results obtained with other methods, but we must emphasize that the results are obtained with a fairly minor number of rules, and as far as we know, with similar or at least no greater computational effort. In short, the real advantage can be appreciated when the size of the rule base is a key issue, which is very important in real situations (i.e. real-life problems).

The results have been equally good in classification and approximation problems. Some complex problems, such as the Pima, cannot be quite so well modeled directly with this method. This is mainly due to the simple approaches to membership function construction, and can be solved by a tuning process (see footnote 2).

Finally, we would briefly like to mention that the proposed method is able to obtain good results with a very low number of rules. These results can easily be improved with a tuning process.

Acknowledgements

The authors acknowledge to the anonymous referees their comments which have permitted to improve the quality of this paper.

This research has been partially supported under project TIC99-0563, DGICYT, MECD, Madrid, Spain.

References


