3 Methodology

Our purpose is to obtain an intelligible fuzzy model from a set of input-output data in a fast and simple way. Thus, not only the models must be intelligible but also the method itself. Although the method will be presented as an automatic process, its simplicity will also allow users to tune it in a fast manner depending on the requirements of the problem.

In summary our method will identify, from a set of input-output data, the necessary partitions to be applied to each input variable in order to model the data in a satisfactory manner. Later it will compute the rules which relate these partitions and furthermore, we will apply some techniques in order to simplify the final model and also to assure its intelligibility. The resulting model will be computed as an output singleton type FRBS due to the reasons which have been explained in the previous chapter.

In this section we will detail all the aspects related to these guidelines. Furthermore the whole algorithm is given in the appendix in order to facilitate its understanding.

3.1 Outline of the method

The method we propose is based on the fact that a fuzzy system can be analyzed as a system which interpolates a function between different regions defined by fuzzy sets. Observe, for instance, the transfer functions of the simple fuzzy systems plotted in figure 3.1 where the fuzzy partitions define the regions between which the output values are interpolated.

In this sense, our method will divide the universe of scope of each input in different partitions which will be assigned to fuzzy sets. Furthermore, and by being interested in obtaining a simple and fast technique, we will only work with one-dimensional functions by dividing the samples of the original

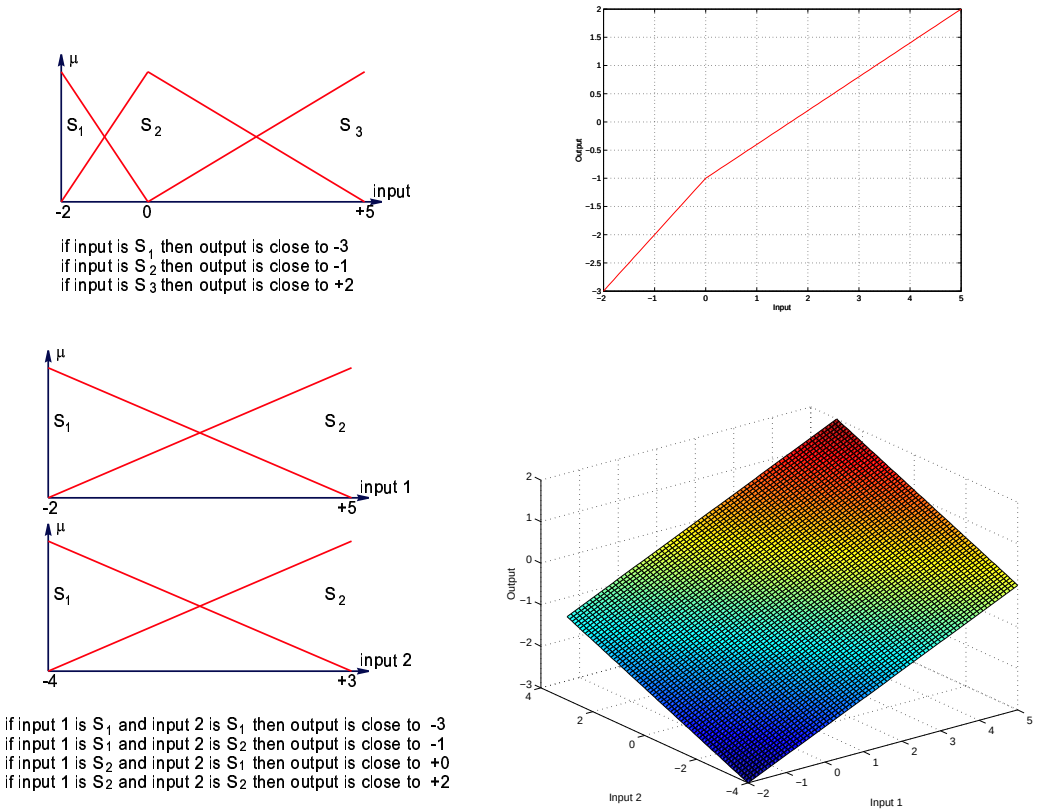


Figure 3.1: Two examples of simple fuzzy models.

MISO system of N inputs in N SISO systems, one for each input. Later we will join the different variables when defining the rules. At the end of the work we will explain the problems which can arise with this option.

Let us introduce the method a little more.

In one of the first steps we will find an optimal relation between the output variable and each possible input. These relations will be computed with an inverse distance weighted average method. The resulting functions are called fuzzy curves and they will be optimized by means of the square error.

In this way, the following steps will work with optimal one-dimensional functions in order to operate in a fast manner in comparison with the actions which would have been required with the original cloud of samples.

Once we have the optimal fuzzy curves, for each input variable we will find the necessary fuzzy sets to obtain a linear-piecewise approximation of the estimated optimal fuzzy curve.

Then we will compute the possible output set assigned to each input partition and consequently the necessary rules to define our fuzzy model.

These two steps, the linearization of the fuzzy curves and the search of the output values, are developed in a hierarchical process. Therefore we will increase the input partitions and also the corresponding number of sets in every step. This will also increase the model's complexity but will reduce its error and the process ends when we are satisfied with the trade-off between accuracy and intelligibility.

In fact this trade-off is adjusted by users with a parameter called *desired error* which defines the maximum error which can be accepted. In fact this parameter will be used in several steps of the method (when optimizing the fuzzy curves, when linearizing them, when clustering some values, ...) and it will be one of the few optional parameters of the whole method apart from the alternatives which can be considered when a fuzzy system is computed such as the implication method, the defuzzification technique, the T-norms, the S-norms, ... and also some necessary values to round different numbers which will be described later.

Apart from these basic steps (optimal fuzzy curves, linearization and output sets) there are some complementary operations in order to accelerate the whole process, for example the use of rounded numbers in some points of the method, or to assure an intelligible final model, for example by considering some clustering techniques.

The overall method is summarized in figure 3.2.

In the following sections we will give the details, the refinements and also the different options to be considered. Nevertheless some aspects will not be explained, basically those related to the search of an efficient computational cost and some minor points of the method, in order to focus on the basic steps by avoiding the dispersion of the guidelines. Anyway we provide the whole algorithm in the appendix.

3.2 Grid partition of the UoS with rounded numbers

By not being interested in a high accuracy method but in a fast, simple and intelligible solution, the values of either the inputs or the output will be rounded to the closest number from a set of values which is computed for every variable.

Each one of these sets will vary based on the demanded accuracy, which is defined with the *desired error* parameter and thus, the higher accuracy the more possible values will be considered for each set.

Thus, for every variable we build a set of equidistant values which cover

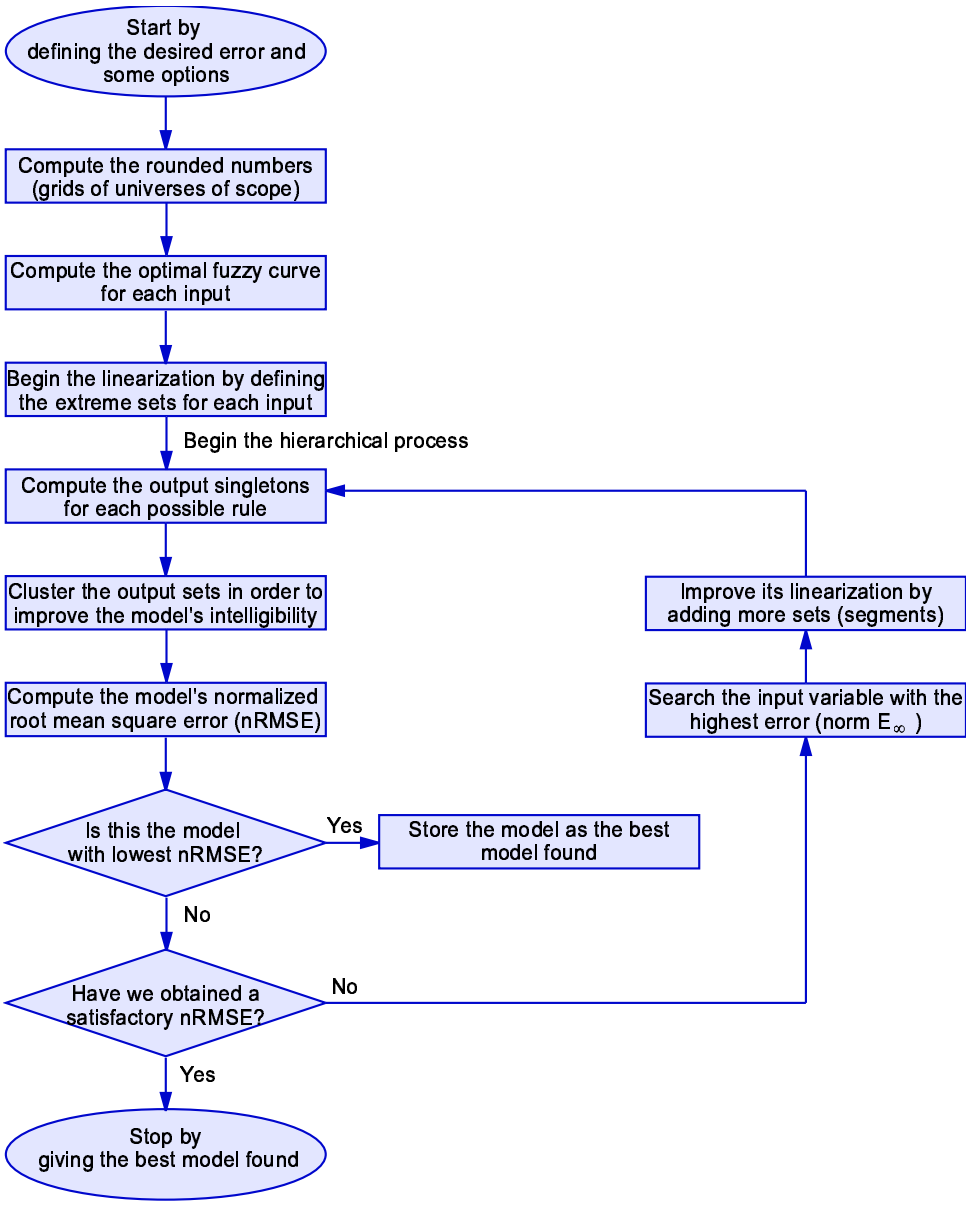


Figure 3.2: A look to the overall method.

the whole universe of scope, from the lowest sample to the highest one, whose step is computed as

$$(max(variable) - min(variable)) \times (desired\ error\ parameter) \tag{3.1}$$

Finally these values are also rounded to their closest decade in order to improve the intelligibility of the resulting models.

Example

Suppose a variable whose samples vary between 0.4562 and 5.3246. If the *desired error* parameter was 15%, the previous formula would give $(5.3246 - 0.4562) \times 0.15 = 0.7303 \simeq 1$, reason why any later value related to this variable, like for example the cores of its fuzzy sets, would be placed to the closest value between 0, 1, 2, 3, 4, 5 or 6 because only these values could define the discrete values of the universe of scope.
Otherwise if the *desired error* parameter was only 5% then the previous formula would have given $(5.3246 - 0.4562) \times 0.05 = 0.2434 \simeq 0.1$ and in this case any result would have been rounded to 0.4, 0.5, 0.6, ... , 5.2, 5.3 or 5.4.

Furthermore and in order to reduce the size of these possible rounded numbers but while keeping the most representative values, if none of the samples is close to one of these possible numbers by being all the samples closer to some other values then we remove this number from the set. This would reduce the necessary time to search the rounded values without degrading the representativity of the set.

3.3 Optimal fuzzy curves

The fact to compute the fuzzy sets from the original data can be very arduous if we have many samples. Therefore recall that we prefer to work with a simplified relation between the output variable and each possible input. This relation will be defined as a fuzzy curve.

3.3.1 Fuzzy curves

There are many techniques in the literature able to give a relation between the output variable and each possible input. The most popular among them are:

- Linear regression
- Weighted average
- Spline (Bézier curves)

Several alternatives coexist because they have different advantages and disadvantages [23, 29]. The most important differences are summarized in table 3.1.

We prefer in this case the weighted average prior to the other alternatives because

Method	Main advantage	Main disadvantage
Linear regression	Optimal result which diminishes the square error	Lack of a method able to discern the best order of the model (model's structure)
Weighted average	Low computational cost	Lack of a compact equation as result
Spline (Bézier curves)	Smooth, continuous, derivable and robust results even with few samples, and therefore suitable for CAD designs more than mathematical analyses	High computational cost if many samples are considered (a cloud of samples) or the requirement of windowed samples (a degree) in order to reduce it

Table 3.1: Advantages and disadvantages between linear regression, weigthed average and Spline.

⇒ we are not interested in an extremely accurate result but in a fast solution which shows the tendency that the output variable adopts when the input varies

⇒ we do not need any algebraic equation as result because the following steps work with numerical methods

⇒ we may not have any previous intuition of the possible relation between the input and the output which could provide a reliant model's structure

Furthermore, the weighted average has an important similitude with fuzzy logic because both methods evaluate the result by considering not only the current situation but also its *neighborhood*.

Given a set of samples $\{x, y\}$, the weighted average computes the output $\hat{y}_i$ when the input is x_i as

$$\hat{y}_i = \frac{\sum_{k=1}^{N} \omega_{ik} y_k}{\sum_{k=1}^{N} \omega_{ik}} \tag{3.2}$$

where ω_{ik} are the weights used to average the output samples y_k based on different possible criteria. Among them the distance $d(x_i, x_k)$ between x_i

and the input value x_k is frequently accepted, then $\omega_{ik} = d(x_i, x_k)$ and

$$\hat{y}_i = \frac{\sum_{k=1}^{N} d(x_i, x_k)\, y_k}{\sum_{k=1}^{N} d(x_i, x_k)} \tag{3.3}$$

where the distance $d(x_i, x_k)$ can be defined in many ways as shows table 3.2.

$d(x_i, x_k)$	**Absolute**	**Relative**
Linear norm	$\lvert x_i - x_k \rvert$	$\lvert x_i - x_k \rvert / \beta$
Square norm	$(x_i - x_k)^2$	$(x_i - x_k)^2 / \beta$

Table 3.2: Distance measurements.

Furthermore these alternatives can be part of another function in order to emphasize some samples. For instance $\omega_{ik} = \exp(-d(x_i, x_k))$ is commonly accepted in order to enhance the closest values.

From the previous options,

$\Rightarrow$ we choose the square norm in order to diminish the square error (optimization procedure)

$\Rightarrow$ we choose only the use of a negative exponential function in order to compute ω_{ik} and then we will have less parameters to optimize by fixing ω_{ik}

$\Rightarrow$ we choose a relative function because otherwise there wouldn't be any possible parameter to optimize

At this point it seems necessary to clarify the apparent incoherence of diminishing the error when in fact we have bet for a simple, although not accurate, method. Certainly the precision of the resulting function is not its most valuable characteristic, but obviously, if the final result displayed a good performance in terms of error no one would reject it. The method we propose will search for a weighted function with a low square error but without assuring its lowest value, as we will just find a statistic of it and thus, this optimized function will have a certain tolerance, while reducing the necessary time to assure the optimal value. This tolerance will be the trade-off between the desired precision and the required computational cost and, in general, we will assume a high tolerance but with a satisfactory error according to our objectives.

When one works with a weighted average one always doubts about the window's length which is defined by a relative measurement of the distance between points. This one will be the parameter that we will optimize.

Therefore, the proposed weighted function is the following one where the β parameter can be adjusted in order to diminish the square error as will be detailed later:

$$\hat{y}_i = \frac{\sum_{k=1}^{N} \phi_{ik} y_k}{\sum_{k=1}^{N} \phi_{ik}} \tag{3.4}$$

$$\phi_{ik} = \exp\left(-\left(\frac{x_k - x_i}{\beta}\right)^2\right) \tag{3.5}$$

In fact this function was proposed by Y. Lin and G. A. Cunningham III [69, 70] and it was called *fuzzy curve* because of its similarities with a fuzzy system with N rules, one per sample, where ϕ_{ik} represents the antecedent set placed at x_k and evaluated when $x = x_i$.

Lin *et al.* used them as a fuzzy modeling method able to identify the significant input variables, determine the model's structure and set the initial weights in a neuro-fuzzy model. They can be interpreted like radial basis functions (RBF) by means of which the value in a point comes near by the average of the samples near this point according to its distance, exponentially measured. Consequently, the resulting functions display a very smooth behavior which allows us to know the relation between the variables of the system, from which we can develop many analyses.

These functions have been used in several applications in which, from input/output data pairs, the fuzzy curves are generated in order to discern those variables which are more relevant when modeling the operation of the system, in addition to helping the development of neural networks or fuzzy systems to foretell the behavior of the system.

The main area where most applications have been developed by using fuzzy curves is in the engineering of mines and geology due to the specialization of its precursors in this field. We can mention in this sense the applications developed in the New Mexico Tech's Petroleum Recovery Research Center which improve the chances of locating oil, better assess the risks of prospecting and drilling for oil and also reduce the costs of oil exploration, for example by computing an average of hydrocarbon equivalent production in the Brushy Canyon wells of New Mexico, USA [121]. Another application consists in ranking the variables applied to restimulation of gas storage wells in the Clinton sand of Ohio, USA [80]. We can find more recent applications of different nature where these curves are used to model a wastewater plant [31] or to predict the algal blooms in the Ortobello lagoon of Italy [77].

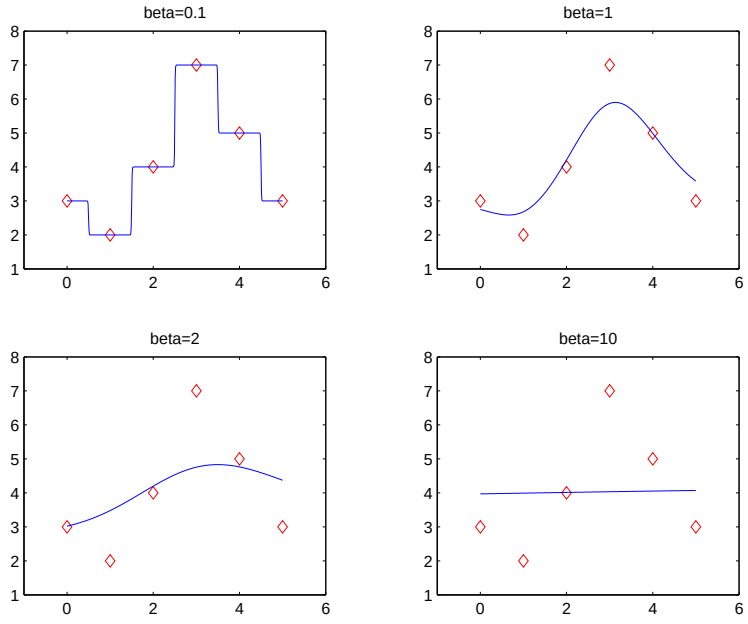
3.3.2 Optimal fuzzy curves

Until now β has usually been adjusted empirically and in fact Lin *et al.* suggest a value equal to the 20% of the variable's range. But as we have introduced before, it is possible to work with an optimized fuzzy curve because the β of the fuzzy curve can be adjusted in such a way that this curve diminishes the square error defined as the difference between the value of the curve and the original sample [32].

Nevertheless, we can not define the error from the sum of all the input samples because then the trivial solution for the parameter β would be equal to zero. Observe that if β is close to zero then any $\phi_{ik} = 0$ except for $k = i$ because then $\phi_{ii} = 1$ and $\hat{y}_i = y_i$. Thus, the error defined as $\varepsilon = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = 0$ but this would give rise to very sharp curves. On the contrary, if β tends to infinity then ϕ_{ik} will be close to one and then $\hat{y}_i$ will be the mean of the output values. In this case we will obtain too flat curves.

Example

Consider a function with only 6 samples: (0,3), (1,2), (2,4), (3,7), (4,5) and (5,3), which is implemented with a fuzzy curve with different values for the β equal to 0.1, 1, 2 and 10. The resulting fuzzy curves are given in the following figure:



If β is very low in comparison with the range of the input values then the fuzzy curve is very sharp. On the contrary, if β is very high the fuzzy curve is very flat and close to 4 by being the mean between 3, 2, 4, 7, 5 and 3. Thus, the optimal β is expected to be proportional to the range of the input values.

Consequently, we will have to divide the samples in two groups: those that will be used to search the optimal parameter and the rest that will be used to evaluate the square error which has been committed with this adjustment.

This can arise a problem if the number of samples is not enough big to keep the original *form* of the fuzzy curve when they are split in two groups and in this case, the β which diminishes the error is in general lower than the value found with the method we propose. Furthermore this division of the samples will give strongly sensible values of the parameter β to this partition, reason why later a statistical study will be made in order to fit its value.

Therefore, we begin by dividing the N samples in N_1 samples which are used to calculate the fuzzy curve (train points) and $N_2 = N - N_1$ samples which are used to test the error (test points). In this way the values of the fuzzy curve in each test point will be computed by using only the N_1 values with the equation 3.6.

$$\hat{y}_i = \frac{\sum_{k=1}^{N_1} \phi_{ik} y_k}{\sum_{k=1}^{N_1} \phi_{ik}} \tag{3.6}$$

Once the fuzzy curve has been computed, the error in each one of the N_2 test points is $\varepsilon_i = y_i - \hat{y}_i$ and the global square error is

$$\varepsilon = \frac{1}{2} \sum_{i=1}^{N_2} {\varepsilon_i}^2 \tag{3.7}$$

As we look for the necessary value of β to diminish this error, we come to calculate the equation 3.8.

$$\frac{\partial \varepsilon}{\partial \beta} = \sum_{i=1}^{N_2} (y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial \beta} = 0 \tag{3.8}$$

Once developed the previous equation we would obtain the condition of the equation 3.9

$$\sum_{i=1}^{N_2} \frac{[A]\left[-(B)(C) + (D)(E)\right]}{[C]^3} = 0 \tag{3.9}$$

where

$$A = \sum_{k=1}^{N_1} \phi_{ik} (y_i - y_k) \tag{3.10}$$

$$B = \sum_{k=1}^{N_1} \phi_{ik} (x_i - x_k)^2 y_k \tag{3.11}$$

$$C = \sum_{k=1}^{N_1} \phi_{ik} \tag{3.12}$$

$$D = \sum_{k=1}^{N_1} \phi_{ik} (x_i - x_k)^2 \tag{3.13}$$

$$E = \sum_{k=1}^{N_1} \phi_{ik} y_k \tag{3.14}$$

by which we can obtain the parameter β which diminishes the global error.

Observe how the previous equation does not present a trivial solution, reason why it must be solved by numerical methods.

In order to accelerate the process, it would be interesting to know the boundaries between which the β can be optimal. In this sense we can observe how the error tends to become stable either for low values of β or for high values and consequently the value which diminishes the error must be found between these limits.

Therefore, if β is very small then $\hat{y}_i \simeq y_k$ for $|x_i - x_k|_{min}$ and only the closest train point to each test point affects to the computation of the fuzzy curve, reason why the error tends to become stable from a certain value of β.

In order to determine this value we have to analyze for which value the effect of the second train point closest to each test point is insignificant and only the effect of the closest train point prevails. If d_1 is the distance from a test point to its closest train point and d_2 is the distance from the same test point to the second train point closest to it, it is interesting to find the β for which

$$\omega_1 = \exp\left[-\left(\frac{d_1}{\beta}\right)^2\right] \gg \omega_2 = \exp\left[-\left(\frac{d_2}{\beta}\right)^2\right] \tag{3.15}$$

or at least for a given $\eta \simeq 0$ find the β for which $\eta\omega_1 > \omega_2$. In this case we would find that

$$\beta > \sqrt{-\frac{{d_2}^2 - {d_1}^2}{\ln(\eta)}} = \beta_{min} \tag{3.16}$$

The most critical situation occurs for the test point having the two train points with the minimum value for ${d_2}^2 - {d_1}^2$.

There is also a big value of β for which the error tends to become stable because then $\hat{y_i}$ is similar to the mean of all the train points.

In this case for each test point we search a value of β for which, if d_1 is the distance from the test point to the train point closest to it and d_∞ is the distance from the same test point to its farthest train point,

$$\omega_1 = \exp\left[-\left(\frac{d_1}{\beta}\right)^2\right] \geq \omega_\infty = \exp\left[-\left(\frac{d_\infty}{\beta}\right)^2\right] \tag{3.17}$$

or at least for a given $\eta \simeq 1$ find the β for which $\eta\omega_1 < \omega_\infty$. Similarly to the previous process we would find that

$$\beta < \sqrt{-\frac{{d_\infty}^2 - {d_1}^2}{\ln(\eta)}} = \beta_{max} \tag{3.18}$$

In this case the most critical situation will occur for the test point with the maximum value for ${d_\infty}^2 - {d_1}^2$.

Once we have the values between which the optimal β can be found, we use a numerical method to determine it. Nevertheless, the fact to divide the samples in two groups can give more than one local minimum which solves the equation 3.9, or even none. Thus, for each group of samples, the range between β_{min} and β_{max} is logarithmically divided in order to locate and to reduce the range (or ranges) where an optimal value can be found. Then we find the local minimum of each range by using a dichotomist method and once all the local minima have been found, the resulting error with each minimum is computed in order to choose the one with the lowest global error.

By dividing the samples in two groups, the optimal value obtained in every partition may not be the optimal one for the whole group of samples, reason why we will not be able to assure the optimal value but at least a statistic.

For this reason we will create different partitions of the samples in an iterative way, from which we will obtain, for each partition, an optimal value of $\beta = \beta_i$ with which we will compute, according to the theorem of the central limit, a confidence interval for this parameter. We will stop the process after n iterations when the confidence interval allows us to assure the optimal value of β with a lower error than an established level η. According to our objectives this level η should not be very low because we do not seek very accurate results. Thus, we will obtain fuzzy curves with a satisfactory error but without increasing our method's computational cost.

In fact we can know, in every iteration, the number of remaining iterations because the confidence interval with a level of confidence α, typically $\alpha = 5\%$, is

$$\overline{\beta} \pm t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \tag{3.19}$$

where

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\beta_i - \overline{\beta}\right)} \tag{3.20}$$

and then

$$t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} < \eta\overline{\beta} \tag{3.21}$$

or

$$n > \left[\frac{t_{\alpha/2,n-1}s}{\overline{\beta}\eta}\right]^2 \tag{3.22}$$

Nevertheless at least 20 iterations should always be considered in order to guarantee the theorem of central limit.

Example

We want to compute the optimal fuzzy curve from the samples given in the following table:

Sample	x	y	Sample	x	y
1	0.0579	3.8529	**8**	0.2026	4.6040
2	0.8132	6.2282	**9**	0.0099	4.3126
3	0.1389	4.2557	**10**	0.6721	6.0068
4	0.1987	4.6400	**11**	0.8462	6.6147
5	0.5252	5.3736	**12**	0.4660	5.5162
6	0.0153	3.8437	**13**	0.7468	6.3632
7	0.4451	5.4368	**14**	0.4186	5.1272

First, we must generate the test points and the train points. In fact these sets of points are computed by taking randomly two samples close to each rounded value of the universe of scope, one of them as train point and the other one as test point. In the next chapter we will explain in detail a whole example and we will tell exactly how these sets are computed. Thus, if in this case the universe of scope is rounded to 0.1, these sets could be:

Test points			Train points		
Sample	x	y	Sample	x	y
1	0.0153	3.8437	**8**	0.0099	4.3126
2	0.1389	4.2557	**9**	0.0579	3.8529
3	0.2026	4.6040	**10**	0.1987	4.6400
4	0.4451	5.4368	**11**	0.4186	5.1272
5	0.5252	5.3736	**12**	0.4660	5.5162
6	0.6721	6.0068	**13**	0.7468	6.3632
7	0.8462	6.6147	**14**	0.8132	6.2282

Now we compute the boundaries between which the optimal β is found $\beta_{min} \leq \beta_{opt} \leq \beta_{max}$. For β_{min} we search the distance between the first and the second of the train points closest to each test point in order to find the points for which this distance is the minimum. This occurs for the fourth test point (0.4451) which has the first and second of the train points closest to it at 0.4660 and 0.4186. Consequently, d_1=0.0209 and d_2=0.0264 and this is the case with the minimum $d_2^2 - d_1^2$. Thus, if the parameter $\eta = 10\%$ then,

$$\beta_{min} = \sqrt{-\frac{0.0264^2 - 0.0209^2}{\ln 0.1}} = 0.0107$$

For β_{max} we search the test point having the train point farthest to it and also the train point closest to it with the maximum distance between them. This occurs for the seventh test point (0.8462) which has the farthest train point placed at 0.0099 and the closest one placed at 0.8132. Thus, d_1=0.0331 and d_∞=0.8364, and if the parameter $\eta = 10\%$ then,

$$\beta_{max} = \sqrt{-\frac{0.8364^2 - 0.0331^2}{\ln 0.9}} = 2.5746$$

As the range between β_{min} and β_{max} is usually very large we compute the derivative of the square error $\frac{\partial \varepsilon}{\partial \beta}$ with some values for the β parameter inside this range in order to reduce the range between which the optimal value is found. For instance if 3 points per decade were considered then the following 9 values would be computed:

β	0.0107	0.0212	0.0421	0.0835	0.1658	0.3292	0.6534	1.2970	2.5746
$\partial\varepsilon/\partial\beta$	-0.0033	-0.0147	-0.0805	-0.0399	0.0099	0.3843	0.9838	0.4522	0.1296

Therefore, the optimal β is between 0.0835 and 0.1658. We finally use the bisection method in order to fit the optimal value with an error lower than the 10%. In this case only three iterations are necessary:

Iteration	Initial	1^{st}	2^{nd}	3^{rd}
$[\beta_{min}, \beta_{max}]$	[0.0835,1658]	[0.1247,0.1658]	[0.1453,0.1658]	[0.1555,0.1658]
$[\partial\varepsilon/\partial\beta_{min}, \partial\varepsilon/\partial\beta_{max}]$	[-0.0399,0.0099]	[-0.0241,0.0099]	[-0.0135,0.0099]	[-0.0034,0.0099]

Once the optimal β is between 0.1555 and 0.1658, we can assure it with an error lower than 10% and we conclude an optimal β equal to the mid point of this range, that is 0.1607.

3.4 Input partition

The fact to work with one-dimensional functions (fuzzy curves) instead of the original cloud of samples will simplify this step. For this purpose we suggest a piecewise linearization of each fuzzy curve because its mapping with the final fuzzy sets is very simple and in fact any error can be reached based on the number of segments we consider. In general the more segments the less error but the less intelligibility.

3.4.1 Linearization of the fuzzy curves

Remember that any linear segment can be obtained with a fuzzy system without any error as can be observed in figure 3.3. Thus, the overall error will be due to the linearization of the curve but not to the fuzzy approximation.

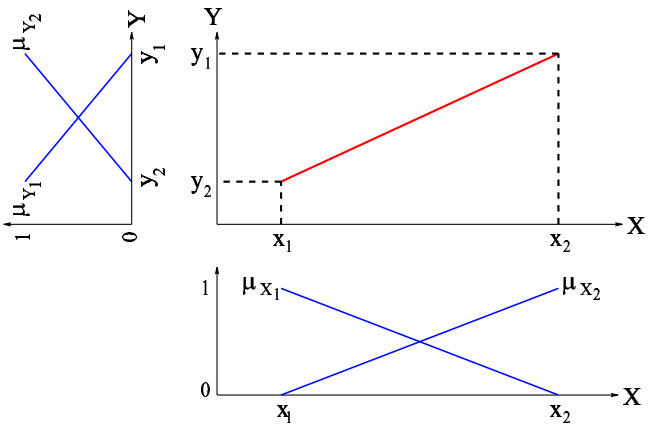


Figure 3.3: A straight line implemented with a fuzzy system.

Observe that if we define

$$\mu_{X_1}(x) = \frac{x_2 - x}{x_2 - x_1} \tag{3.23}$$

as the fuzzification of the input value x in the set X_1,

$$\mu_{X_2}(x) = \frac{x - x_1}{x_2 - x_1} \tag{3.24}$$

as the fuzzification of the input value x in the set X_2, y_1 as the core[1] of the set Y_1, y_2 as the core of the set Y_2 and the defuzzification technique is defined as

$$y = \frac{\mu_{X_1}(x)\, y_1 + \mu_{X_2}(x)\, y_2}{\mu_{X_1}(x) + \mu_{X_2}(x)} \tag{3.25}$$

then, with the fact that $\mu_{X_1}(x) + \mu_{X_2}(x) = 1$, we conclude that

$$y = \frac{\mu_{X_2}(x)(y_2 - y_1) + y_1}{1} = \frac{x - x_1}{x_2 - x_1}(y_2 - y_1) + y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 \tag{3.26}$$

which is the equation of the original line. In fact we have computed an output singleton FRBS with the sum-product operator.

[1]Point of the fuzzy set with the highest degree by being its most representative value.

Therefore we can use triangular membership functions by being interested in a simple fuzzy model able to fit the linearized fuzzy curve without any error.

In fact L.X. Wang [117] proved the *universal approximation theorem* for fuzzy systems from which any real continuous function can be implemented with bell-shaped Gaussians membership functions (or any square integrable functions).

This theorem is not in contrast to the solution we propose because the use of Gaussians membership functions do not guarantee an optimal solution in terms of the number of sets. On the contrary, the number of triangular membership functions which are necessary to implement a linear function without error is optimal for the FRBS we take into account.

This kind of implementation of a linear segment with a fuzzy system was already used by C.M. Higgins and R.M. Goodman [40] but with a more complex procedure if the overall method is compared. It has also been used in computed aided geometric design CADG [29] where the fuzzy curves are called hat functions.

The method we propose is similar to the one suggested by C.M. Higgins and R.M. Goodman [40]. The main difference is that we work with only one input variable every time in order to avoid the required simplification that C.M. Higgins and R.M. Goodman need after all their fuzzy sets and consequently all their linguistic rules are stated.

Therefore, we suggest the following procedure:

1. We plot a straight line to join the first and the last point of the fuzzy curve. In fact these points will fix the boundaries of the universe of scope and also the extreme fuzzy sets.

2. We choose the farthest point of the fuzzy curve to the linear approximation in order to be part of the next linearization and to reduce the current maximum error.

   Thus, a straight line joins this point and the next point of the scope which was considered in the previous linear approximation while another straight line joins the selected point and the previous point of the scope which was considered in the previous linear approximation.

3. We modify the fuzzy sets in order to implement this linearization with fuzzy partitions.

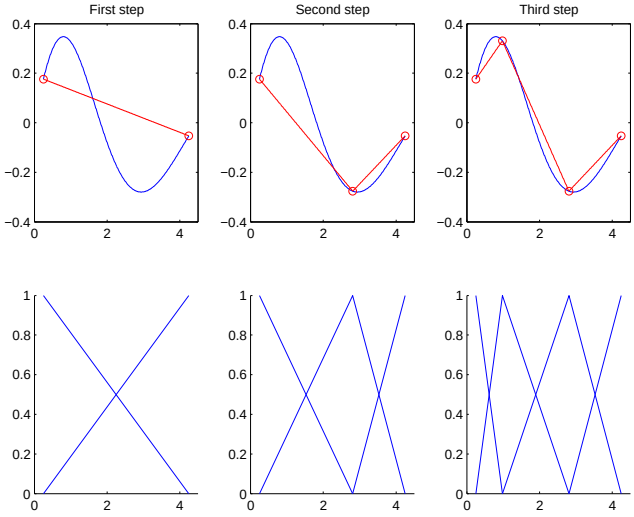
4. We repeat the process until assuring the desired precision.

Finally the fuzzy curve defined with N points $fc = \{(x_i, y_i) \mid i = 1...N\}$ is piecewised linearized by considering the boundaries of each one of the segments $\{(x_j, y_j) \in fc\}$ as

$$l(x_i) = \left\{ \frac{y_j - y_{j-1}}{x_j - x_{j-1}} (x_i - x_{j-1}) + y_{j-1} \mid x_{j-1} \leq x_i \leq x_j \right\} \tag{3.27}$$

Example

In the following figure we have an example of the method and also the necessary fuzzy sets in order to define each linear segment:



The error may not be a monotonically decreasing sequence but it can be easily concluded that if the fuzzy curve is defined with N points, the proposed method will remove the error in N-1 iterations at most because then the linearization will be defined with N-1 straight lines by joining the N points and consequently, N fuzzy sets will be considered for the input variable, one for each point.

Nevertheless, we will fix a certain error different from zero, by using the *desired error* parameter, in order to avoid this situation and to stop the process before. Obviously the lower accepted error the higher number of sets and also the higher complexity of the model.

The measurement of the error can be computed again in many ways, like those described in table 3.2, but in this case:

⇒ it is not necessary to optimize any parameter

⇒ there is no reason to penalize those points with a high error more than with the error itself

⇒ nevertheless, it is necessary a relative error because the process must stop when the linearization relative to the fuzzy curve shows a good approximation

Therefore the error in this step will be defined with equation 3.28

$$\varepsilon = \frac{\sum_k |\hat{y_k} - y_k|}{\sum_k |y_k|} \tag{3.28}$$

where y_k is the output value of the k-th point of the fuzzy curve and $\hat{y_k}$ is the value of the linear approximation for the same point.

Anyway we also compute the square error and the root mean square error in order to compare our method with the precision achieved in the most popular examples found in the literature.

Observe that the solution we propose is based on fuzzy partitions (Ruspini partitions) in order to guarantee:

- the distinguishability property
- the completeness property
- the coverage of the scope
- the normal property
- the transparency of the model

which are some of the necessary conditions among the intelligibility criteria that we have analyzed in the previous chapter and thus, we satisfy several properties of intelligible fuzzy models.

3.4.2 Round-off errors and rounded values

At this point it is interesting to expose the problems which can appear due to the round-off errors and their solution. We have already explained the need to work with rounded numbers in the universe of scope but we will require a similar solution in the axis of ordinate.

That is because computers can not store exact values when dealing with floating point numbers except for powers of two. The ANSI/IEEE standard 754 suggests that any non-zero number should be computed by using the form $\pm(1+f)\times 2^e$ where either the mantissa f or the exponent e must have finite expansions in base two. This finite nature of f brings about round-off errors because these computed numbers have a limited accuracy while the

finite nature of e may involve underflow and overflow problems. We must take into account this problem.

Here some aspects should be considered. The first one refers to the samples and should be considered before starting the process. It is the machine epsilon (epsmch) defined as the smallest positive number such that 1+epsmch is not equal to 1.

Example

When using C library files this parameter can be found in the header file *float.h* as a constant called DBL_EPSILON. Instead of fixing its value we consider more properly to compute it with the following simple sequence in order to avoid working with samples close to it:

```
temp=1.0
do {
      temp=temp/2
    } while (1.0<(temp+1.0))
epsmch=temp
```

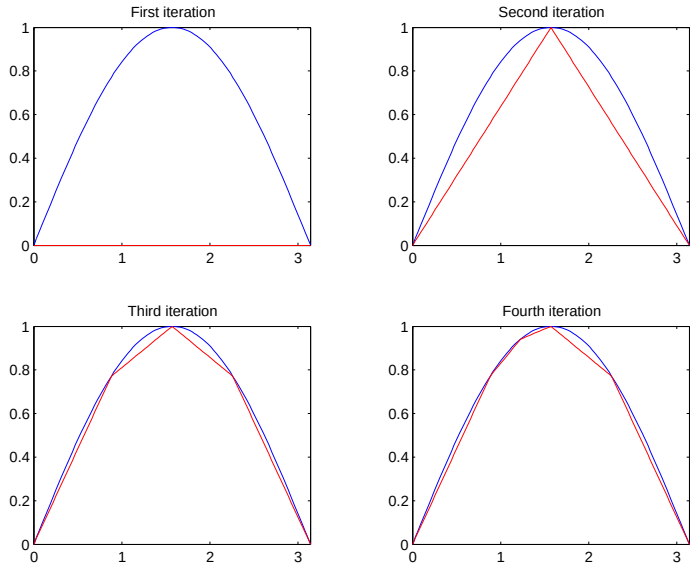
Other problems due to round-off errors can be minimized or even avoided with a proper program, basically by controlling the loops or the comparisons. Recall that the potential errors in a loop is much greater than in sequential code and after computing N arithmetic operations the total round-off error can be on the order of $\sqrt{N}$epsmch if the round-off errors come in randomly up or down.

We have tried to avoid accumulating floating point values through repeated addition or subtraction in loops as much as possible. We also have avoided direct comparisons between numbers and before doing a test for equality, we have compared the difference of the operands to check them against a tolerance.

In general these problems can be avoided if working with rounded values. Moreover, by not being very interested in the exactitude of the models but in its global intelligibility, rounded values seem to have more benefits than disadvantages. For example if we must search the point of the linearized fuzzy curve with the maximum error and two of these points have an error very similar (equal if they were rounded) and clearly higher than the error of the rest of points, then it would be plausible to choose both points simultaneously and not only one of them.

Example

This phenomenon can be better understood with the following example. Suppose that the proposed method is applied with Matlab to the sine function defined between 0 and π with 101 equidistant samples. The first four iterations are plotted in the following figure:



Observe that in the fourth iteration only the point $\{1.2252, 0.9409\}$ has been chosen in order to implement the new segment instead of taking it but also the point $\{1.9164, 0.9409\}$ by being apparently at the same distance than the previous linearization, like the points $\{0.8796, 0.7705\}$ and $\{2.2619, 0.7705\}$ were chosen in the third iteration for the same reason.
This problem is due to the round-off errors. In the example, the distance between the point $\{1.2252, 0.9409\}$ or the point $\{1.9164, 0.9409\}$ and the previous linearization is apparently the same except for my computer.

For this reason either the input values or the output values must be rounded and therefore the corresponding axis of abscissa and the axis of ordinate.

These rounded axes obviously can differ between users. In general the lower step the more precision but the more complexity and consequently a less comprehensible model. Thus, every user must define it by averaging the precision and the understanding.

The fact to work with rounded values can give more than one point with the same maximum error in some iterations. We suggest considering all of them for the next linearization except for when these points are successive. In this case we will only consider the point which really displays the highest error without being rounded or the mean between those which have the maximum rounded error.

Figure 3.4 shows this phenomenon where we would choose two points for the next linearization because both of them have the maximum rounded error and, as each one of them has more than one value with the same rounded error, we choose the value with the highest error or their mean.

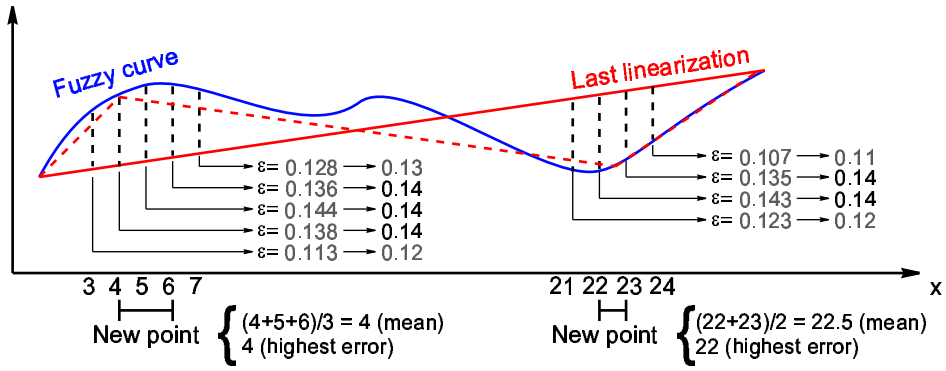


Figure 3.4: Effect of the rounded values when linearizing.

Another conclusion related with the use of rounded values is that they bound the maximum number of sets and consequently also the linear model.

For this reason the process will stop whether the desired error is accomplished or if the number of iterations is equal to the minimum between the number of possible segments and the number of different possible levels of error due to the rounded values.

Example

This phenomenon can be clearly observed again with the sine function example. Consider that the x-axis is rounded with a step of $\pi/10$ and thus, a maximum of 10 segments can be implemented. Also consider that the y-axis is rounded with a step of 0.2 and thus, a maximum of 5 different levels of error can be perceived. In this case the linearization would admit a maximum of 5 segments by being the minimum between 5 and 10.

3.4.3 Odd fuzzy curves

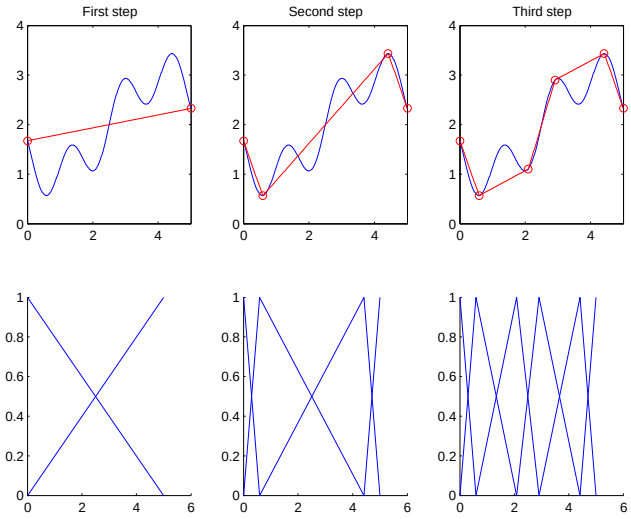
The odd fuzzy curves require a special treatment because the mid-point of the linearized fuzzy curve passes through the optimal fuzzy curve and therefore, the error in this point is always zero. According to the proposed method this point would never be chosen as the core of a new set because in every iteration only those points with the highest error are selected.

Nevertheless this point can be very significant when the process is described because it defines its symmetry. Furthermore in control systems it typically defines the desired situation when the target has been accomplished, i.e. the error is zero.

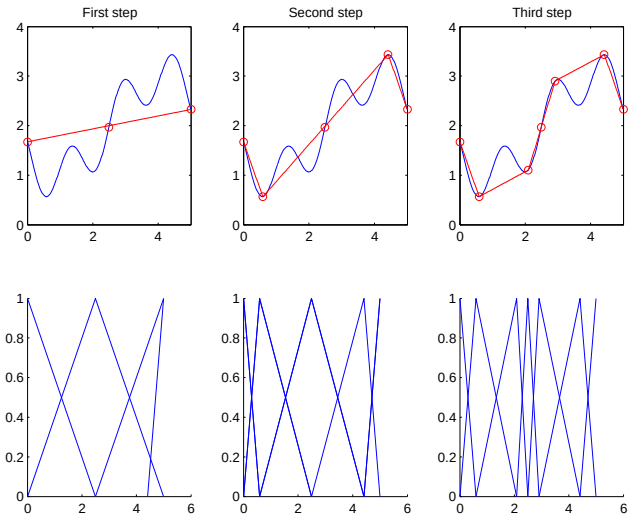
For these reasons, if an odd fuzzy curve is identified, its mid-point is chosen at the beginning together with the first and the last points and thus, we begin with three fuzzy sets instead of only two.

Example

Observe in the following figure how the odd fuzzy curve never considers its midpoint:



Therefore we choose it at the beginning in order to define this situation in the model:



This special consideration is necessary to guarantee the *natural zero positioning*. Recall that in some problems this is a necessary condition among the intelligibility criteria that we have analyzed in the previous chapter.

3.5 Possible output sets

Once defined the input fuzzy sets and consequently after fixing the maximum number of rules as the product of the number of sets of each variable, it is only necessary to define the linguistic rules by computing the output set for each possible rule.

In fact first we will search the possible output sets and later we will cluster them in order to reduce the number of output sets and thus, to increase the model's intelligibility.

This first problem, the search of the possible output sets, is commonly solved by finding the value which best fits from the set of samples. Here we can consider the two most popular alternatives, the Wang&Mendel's method and the Takagi&Sugeno's method. In fact both of them have been proved in our method and any of them could be considered. We will detail both but nevertheless, later we will justify why we prefer the first one.

3.5.1 Wang&Mendel's alternative

The first alternative was proposed by L.X. Wang and J.M. Mendel [120] who designed a simple method to generate a set of *"if ... then ..."* rules by working directly with the input-output data. The method requires the division of the universe of scope into different regions having membership functions, either for the input variables or for the output. Then for each possible rule formed by a combination of input regions they choose the output set closest to the output value of the sample with the highest implication level. The method is summarised as follows:

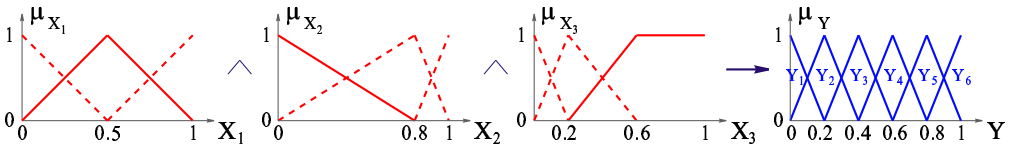
1. First, they assign each sample to the possible rule with the highest degrees of membership.

2. In order to resolve conflicting rules, i.e. rules with more than one sample assigned to it whose output values are closest to different output sets, they compute a degree for each sample as the product of the degrees of membership, either for the input variables or for the output. The sample with the highest degree is kept in order to assign the output set closest to it as the output set of the rule.

3. Finally they generate the combined rulebase comprised both of numerically generated fuzzy rules as described above and linguistic information provided by experts.

Example

Suppose that we must decide which output set among $Y_1 \dots Y_6$ must be assigned to the following rule



from the following samples:

Sample	x_1	x_2	x_3	y
1	0.6	0.4	0.7	0.58
2	0.8	0.6	0.5	0.38
3	0.7	0.2	0.5	0.65
4	0.3	0.0	0.8	0.29
5	0.2	0.7	0.9	0.46
6	0.8	0.2	0.1	0.17
7	0.4	0.2	0.4	0.72
8	0.5	0.7	0.0	0.24

First of all we search the samples that applied to the rule under test show degrees of membership higher than other rules. In fhis case we will just take into account the samples with $0.25 \leq x_1 \leq 0.75$, $0.00 \leq x_2 \leq 0.40$ and $0.40 \leq x_3 \leq 1.00$. Thus, we will only consider the samples number 1, 3, 4 and 7 for the current rule.
Then we search the output set closest to the output value of each one of these four samples. In this case there are conflicting rules because the sample number 1 and 3 would assign the output set Y_3, but the sample number 4 would assign the output set Y_2 and the sample number 7 would assign the output set Y_5. Thefefore, we come to compute the degrees of these four samples as:

$$\begin{aligned}
\text{Sample 1} \quad &\rightarrow \quad \mu_{X_1}(0.6) \times \mu_{X_2}(0.4) \times \mu_{X_3}(0.7) \times \mu_{Y_4}(0.58) = \\
&= 0.80 \times 0.50 \times 1.00 \times 0.90 = 0.36 \\
\text{Sample 3} \quad &\rightarrow \quad \mu_{X_1}(0.7) \times \mu_{X_2}(0.2) \times \mu_{X_3}(0.5) \times \mu_{Y_4}(0.65) = \\
&= 0.60 \times 0.75 \times 0.75 \times 0.75 = 0.25 \\
\text{Sample 4} \quad &\rightarrow \quad \mu_{X_1}(0.3) \times \mu_{X_2}(0.0) \times \mu_{X_3}(0.8) \times \mu_{Y_2}(0.29) = \\
&= 0.60 \times 1.00 \times 1.00 \times 0.55 = 0.33 \\
\text{Sample 7} \quad &\rightarrow \quad \mu_{X_1}(0.4) \times \mu_{X_2}(0.2) \times \mu_{X_3}(0.4) \times \mu_{Y_5}(0.72) = \\
&= 0.80 \times 0.75 \times 0.50 \times 0.60 = 0.18
\end{aligned}$$

Thus, the output set applied to the rule must be Y_4 by being the output set closest to the output value of the sample with the highest degree.

As we do not have at this point of the method any division of the universe of scope of the output variable, we propose a similar version of this method which basically consists in computing the implication value of each sample like Wang&Mendel but without considering the bag of output sets. Furthermore we do not split the samples in different groups because all of them are considered for each possible rule.

Example

In the last example, if we compute the T-norms with the product, the following implication values would be computed for each sample as,

$$\begin{array}{lcl}
\text{Sample 1} & \rightarrow & \mu_{X_1}(0.6) \times \mu_{X_2}(0.4) \times \mu_{X_3}(0.7) = 0.40 \\
\text{Sample 2} & \rightarrow & \mu_{X_1}(0.8) \times \mu_{X_2}(0.6) \times \mu_{X_3}(0.5) = 0.23 \\
\text{Sample 3} & \rightarrow & \mu_{X_1}(0.7) \times \mu_{X_2}(0.2) \times \mu_{X_3}(0.5) = 0.34 \\
\text{Sample 4} & \rightarrow & \mu_{X_1}(0.3) \times \mu_{X_2}(0.0) \times \mu_{X_3}(0.8) = 0.60 \\
\text{Sample 5} & \rightarrow & \mu_{X_1}(0.2) \times \mu_{X_2}(0.7) \times \mu_{X_3}(0.9) = 0.05 \\
\text{Sample 6} & \rightarrow & \mu_{X_1}(0.8) \times \mu_{X_2}(0.2) \times \mu_{X_3}(0.1) = 0.00 \\
\text{Sample 7} & \rightarrow & \mu_{X_1}(0.4) \times \mu_{X_2}(0.2) \times \mu_{X_3}(0.4) = 0.30 \\
\text{Sample 8} & \rightarrow & \mu_{X_1}(0.5) \times \mu_{X_2}(0.7) \times \mu_{X_3}(0.0) = 0.00
\end{array}$$

thus, we conclude that the core of a possible output set could be placed at 0.29 by being the output value of the sample with the highest implication level.

Thus, the main difference between the alternative we propose and the original method defined by Wang&Mendel is that we do not use any predefined bag of output sets because we just need to define where the output sets may be placed (possible output sets and not the final output sets). Later we will use clustering techniques and rounded values in order to define the final output sets and consequently the final rules.

3.5.2 Takagi&Sugeno's alternative

The second alternative proposed by T. Takagi and M. Sugeno [112, 109, 110] finds the optimum rules for a fixed membership function configuration. In spite of diminishing the error, its computational cost is significantly higher if there are many input partitions, as can be derived from the following operations [95].

Let us consider K samples of a system F with N inputs $(X_1, \ldots, X_N)$ and one output (Y) given by K vectors as $\left(x_1^k, x_2^k, \ldots, x_N^k; y^k\right)\ k = 1, 2, \ldots, K$ with $\vec{x}^k \in \Re$ and $y^k \in \Re$. We are interested in approximating these data with a fuzzy system of the form:

$$\text{If } x_1 \text{ is } X_1^{i_1} \text{ and } x_2 \text{ is } X_2^{i_2} \text{ and } \ldots \text{ and } x_N \text{ is } X_N^{i_N} \text{ then } y \text{ is } Y_{i_1 i_2 \ldots i_N}$$

where $X_m^{i_m} \in \{X_m^1, X_m^2, \ldots, X_m^{n_m}\}$ with n_m being the number of membership functions of the input m while $Y_{i_1 i_2 \ldots i_N} \in \Re$ is the numeric consequent of the rule.

In our approach we compute the T-norm with the product and the defuzzification with the sum-product operator. Thus, the implication value $\mu_{i_1 i_2 \ldots i_N}(\vec{x})$ of the rule with $Y_{i_1 i_2 \ldots i_N}$ is computed by

$$\mu_{i_1 i_2 \ldots i_N}(\vec{x}) = \prod_{m=1}^{N} \mu_{X_m^{i_m}}(x_m) \tag{3.29}$$

where $\mu_{X_m^{i_m}}(x_m)$ is the degree of membership of the m component of the vector $\vec{x}$ related to the linguistic function i_m.

By using the above notation the fuzzy function can be expressed as follows:

$$\widetilde{F}\left(\vec{x}^k;Y\right)=\frac{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(Y_{i_1i_2\ldots i_N}\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)\right)} \tag{3.30}$$

We are interested in finding all the consequents $Y_{i_1i_2\ldots i_N}$ of $\widetilde{F}\left(\vec{x}^k;R\right)$ in order to minimize the objective function

$$O(Y)=\sum_k\left(F\left(\vec{x}^k\right)-\widetilde{F}\left(\vec{x}^k;Y\right)\right)^2 \tag{3.31}$$

Thus, we come to compute

$$\frac{\partial O(Y)}{\partial Y_{j_1j_2\ldots j_N}}=-2\sum_k\left[\left(F\left(\vec{x}^k\right)-\widetilde{F}\left(\vec{x}^k;Y\right)\right)\left(\frac{\partial\,\widetilde{F}\left(\vec{x}^k;Y\right)}{\partial Y_{j_1j_2\ldots j_N}}\right)\right] \tag{3.32}$$

And by introducing the following notation

$$\begin{aligned}S_{i_1i_2\ldots i_N,j_1j_2\ldots j_N} \;=\; &\sum_k\frac{\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)\right)}\times\\ &\times\frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}\left(x_m^k\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)\right)}\end{aligned} \tag{3.33}$$

$$S_{F,j_1j_2\ldots j_N}=\sum_k F\left(\vec{x}^k\right)\cdot\frac{\prod_{m=1}^{N}\mu_{X_m^{j_m}}\left(x_m^k\right)}{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(\prod_{m=1}^{N}\mu_{X_m^{i_m}}\left(x_m^k\right)\right)} \tag{3.34}$$

we obtain the following system of linear equations:

$$\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\left(Y_{i_1 i_2 \ldots i_N}\cdot S_{i_1 i_2 \ldots i_N, j_1 j_2 \ldots j_N}\right) = S_{F, j_1 j_2 \ldots j_N} \tag{3.35}$$

where $1 \leq j_1 \leq n_1$, $1 \leq j_2 \leq n_2$, and so on. Therefore, by solving this system of $n_1 \times n_2 \times \cdots \times n_N$ linear equations we obtain the consequents $Y_{i_1 i_2 \ldots i_N}$ of all the rules which minimize the square error.

This matrix is always two-dimensional and symmetrical, reason why it is recommended to be solved with the Cholesky factorisation [93] by being one of the fastest decomposition available. It constructs a lower triangular matrix as a particular case of the LU decomposition from which we can solve the linear system.

Example

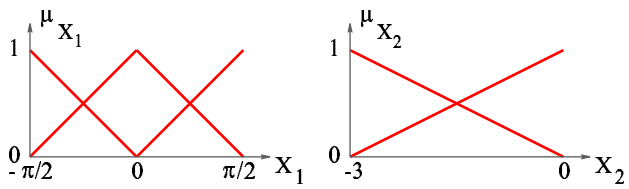
We will compare Wang&Mendel and Takagi&Sugeno with this example. Consider that we are approaching the following function:

$$y = |\sin(x_1)e^{x_2}| \qquad \text{where } x_1 \in [-\pi/2, \pi/2] \text{ and } x_2 \in [-3, 0]$$

with the following 30 samples:

Samples							
k	x_1	x_2	y	k	x_1	x_2	y
1	-1.0672	-0.9205	+0.3489	16	-1.0819	-1.1376	+0.2830
2	+0.4581	-1.3420	+0.1156	17	+1.1183	-0.2856	+0.6760
3	+1.1636	-1.1244	+0.2983	18	-0.1232	-2.2968	+0.0124
4	+1.0014	-2.9704	+0.0432	19	-0.2758	-1.9944	+0.0371
5	+0.7380	-0.6184	+0.3625	20	-0.6365	-1.1181	+0.1943
6	-1.0492	-1.8967	+0.1301	21	-0.8235	-0.0510	+0.6970
7	-0.2436	-2.4183	+0.0215	22	+0.5181	-2.4036	+0.0448
8	+0.9925	-1.1046	+0.2775	23	-1.1404	-1.8723	+0.1397
9	-0.1549	-0.2052	+0.1257	24	+1.2262	-0.7390	+0.4495
10	-1.5533	-1.8243	+0.1613	25	-0.4835	-0.4658	+0.2917
11	+1.1714	-1.0497	+0.3225	26	-0.9704	-0.8062	+0.3685
12	+1.4668	-1.7998	+0.1644	27	-0.0306	-1.2924	+0.0084
13	-1.5396	-0.7999	+0.4492	28	-0.1340	-1.3537	+0.0345
14	-0.2194	-1.7404	+0.0382	29	+1.2617	-1.0334	+0.3389
15	+0.5885	-0.2401	+0.4366	30	-1.4163	-0.9028	+0.4006

Suppose that we have already divided the universes of scope of both inputs with the following fuzzy sets



and we are interested in finding the output singletons for its $3 \times 2 = 6$ possible rules.
If we used the Wang&Mendel's method, we would search, for each possible rule, the sample with the highest implication value in order to assign its output value as the output singleton of the rule.
In this case, the rules would have been fixed as follows:
Rule 1 with highest implication value from the 10th sample:
if x_1 is close to $-\pi/2$ and x_2 is close to -3 then y is close to 0.1613
Rule 2 with highest implication value from the 13th sample:
if x_1 is close to $-\pi/2$ and x_2 is close to 0 then y is close to 0.4492
Rule 3 with highest implication value from the 18th sample:
if x_1 is close to 0 and x_2 is close to -3 then y is close to 0.0124
Rule 4 with highest implication value from the 9th sample:
if x_1 is close to 0 and x_2 is close to 0 then y is close to 0.1257
Rule 5 with highest implication value from the 4th sample:
if x_1 is close to $\pi/2$ and x_2 is close to -3 then y is close to 0.0432
Rule 6 with highest implication value from the 17th sample:
if x_1 is close to $\pi/2$ and x_2 is close to 0 then y is close to 0.6760
Otherwise if we used the Takagi&Sugeno's optimal method, we would first prepare the following system of linear equations,

$$\begin{array}{l}\text{Rule 1}\\ \text{Rule 2}\\ \text{Rule 3}\\ \text{Rule 4}\\ \text{Rule 5}\\ \text{Rule 6}\end{array}\begin{bmatrix} X_{11,11} & X_{12,11} & X_{21,11} & X_{22,11} & X_{31,11} & X_{32,11} \\ X_{11,12} & X_{12,12} & X_{21,12} & X_{22,12} & X_{31,12} & X_{32,12} \\ X_{11,21} & X_{12,21} & X_{21,21} & X_{22,21} & X_{31,21} & X_{32,21} \\ X_{11,22} & X_{12,22} & X_{21,22} & X_{22,22} & X_{31,22} & X_{32,22} \\ X_{11,31} & X_{12,31} & X_{21,31} & X_{22,31} & X_{31,31} & X_{32,31} \\ X_{11,32} & X_{12,32} & X_{21,32} & X_{22,32} & X_{31,32} & X_{32,32} \end{bmatrix} \begin{bmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{bmatrix} = \begin{bmatrix} X_{F,11} \\ X_{F,12} \\ X_{F,21} \\ X_{F,22} \\ X_{F,31} \\ X_{F,32} \end{bmatrix}$$

where

$$X_{i_1 i_2, j_1 j_2} = \sum_{k=1}^{30} \frac{\mu(k) \text{ in the rule } i_1 i_2}{\sum_{r=1}^{6} \mu(k) \text{ in the r-rule}} \cdot \frac{\mu(k) \text{ in the rule } j_1 j_2}{\sum_{r=1}^{6} \mu(k) \text{ in the r-rule}}$$

$$X_{F,j_1 j_2} = \sum_{k=1}^{30} \text{output value of the k-sample} \cdot \frac{\mu(k) \text{ in the rule } j_1 j_2}{\sum_{r=1}^{6} \mu(k) \text{ in the r-rule}}$$

By computing the different values we have the following system

$$\begin{bmatrix} 1.0925 & 1.1785 & 0.5385 & 0.4796 & 0.0000 & 0.0000 \\ 1.1785 & 2.2244 & 0.4796 & 1.0682 & 0.0000 & 0.0000 \\ 0.5385 & 0.4796 & 2.6104 & 1.7181 & 0.5569 & 0.3888 \\ 0.4796 & 1.0682 & 1.7181 & 3.3435 & 0.3888 & 1.0182 \\ 0.0000 & 0.0000 & 0.5569 & 0.3888 & 1.1268 & 0.9485 \\ 0.0000 & 0.0000 & 0.3888 & 1.0182 & 0.9485 & 2.0750 \end{bmatrix} \begin{bmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{bmatrix} = \begin{bmatrix} 0.7117 \\ 1.6627 \\ 0.6457 \\ 1.9470 \\ 0.6520 \\ 1.6526 \end{bmatrix}$$

from which we obtain the rules:
Rule 1 with output singleton equal to Y_{11}:
if x_1 is close to $-\pi/2$ and x_2 is close to -3 then y is close to -0.3136

Rule 2 with output singleton equal to Y_{12}:
if x_1 is close to $-\pi/2$ and x_2 is close to 0 then y is close to 0.8498
Rule 3 with output singleton equal to Y_{21}:
if x_1 is close to 0 and x_2 is close to -3 then y is close to -0.0338
Rule 4 with output singleton equal to Y_{22}:
if x_1 is close to 0 and x_2 is close to 0 then y is close to 0.1483
Rule 5 with output singleton equal to Y_{31}:
if x_1 is close to $\pi/2$ and x_2 is close to -3 then y is close to -0.1142
Rule 6 with output singleton equal to Y_{32}:
if x_1 is close to $\pi/2$ and x_2 is close to 0 then y is close to 0.7822

If we compare both methods by considering the results of the last example, we could observe that

⇒ Obviously Takagi&Sugeno's optimal solution is better in terms of error because this is its main objective[2].

⇒ Wang&Mendel's method is clearly faster because it avoids the need to build the linear system and to solve it[3].

⇒ Takagi&Sugeno can fail if the system is badly conditioned[4].

⇒ Takagi&Sugeno can give absurd results[5].

Therefore, due to these aspects and according to our objectives, we prefer in general the Wang&Mendel's option rather than the Takagi&Sugeno's option.

3.5.3 Removing rules not according to the samples from the rule matrix

After applying one of the previous alternatives, the Wang&Mendel or the Takagi&Sugeno, we can also find out those rules to be omitted by not having any sample directly related to it. This occurs if the implication value of all the samples applied to a possible rule is zero.

[2]In this case the $RMSE_{Takagi\&Sugeno}$=6.5% while the $RMSE_{Wang\&Mendel}$=10.4%.

[3]In this case the elapsed time with Takagi&Sugeno was only the double than the Wang&Mendel's option but it increases significantly basically with the number of input partitions.

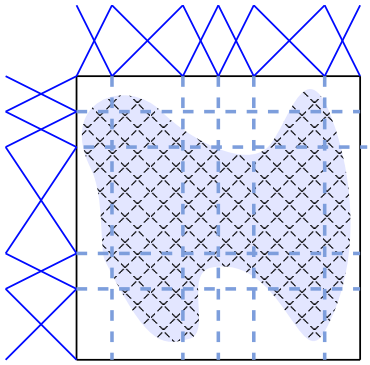
[4]We must control this phenomenon and if it occurs we change to the Wang&Mendel's option.

[5]Observe how in this example there are output sets placed outside the reasonable limits of the original function [0,1].

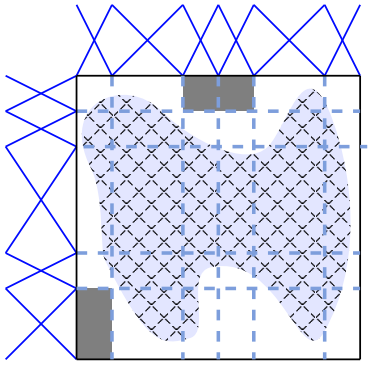
In this way the rule matrix can be simplified and only the rules suited with the original set of samples are preserved.

Example

Suppose a system with 2 inputs from which we have already decided the input sets for a certain set of samples and also the possible output sets. The input values of the different samples are also plotted over the rule matrix as a shadow:



Observe how there are two rules without any sample directly related to them and they may be removed from the rule matrix.



Thus, there will be only 40 rules instead of the possible 42.

By considering this option we improve the *compactness* of the fuzzy rules, which is one of the criteria for a satisfactory intelligibility of the final model. Anyway recall that then the *completeness* of fuzzy partitions, another of these criteria, could be degraded if finally there were no rules related to an input set. Nevertheless the final model adapts better to the input-output data. Therefore, we suggest that the users consider this option or not based on the nature of the problem and their preferences.

3.6 Clustering the possible output sets

Clustering can be considered as the process of grouping objects which are similar in some way. A cluster is therefore a collection of objects which are similar between them and dissimilar to the objects belonging to other clusters. Thus, we can cluster the possible output values of the fuzzy rules in order to reduce the number of output sets and consequently the labels of the output variable.

This is a necessary step if we are interested in satisfying a *justifiable number of labels*, which is a necessary condition among the intelligibility criteria which have been analyzed in the previous chapter.

Clustering methods can be classified as:

- Exclusive clustering: data are grouped in an exclusive way so that each datum belongs to only one cluster because a clear neighbor is defined between them. The typical algorithm is K-means.

  Despite being very simple it has some disadvantages: the results depend on some values which in general are chosen randomly and moreover we often have no way of knowing how many clusters exists. For these reasons we avoid this method.

- Overlapping clustering: on the contrary this method uses fuzzy sets in order to cluster the data and thus, each point may belong to more than one cluster with different degrees of membership.

  The most popular algorithm is Fuzzy C-means which will be discussed later in detail because we consider it a suitable solution.

- Hierarchical clustering: this method is based on the union between the two nearest clusters at each iteration. The beginning condition is realized by setting every datum as a cluster and after some iterations the final number of clusters is reached.

  Here we suggest Chiu's clustering which will be explained later by being our preferred alternative.

- Probabilistic clustering: the clusters are given by the probabilistic membership of each datum to possible clusters. A good example is the mixture of gaussian.

  Nevertheless we avoid this method because we seek finite values but not probabilistic values and moreover its computational cost is in general higher than the other methods.

3.6.1 Fuzzy C-means

This method developed by J. C. Dunn in 1973 [24] and improved by J. C. Bezdek in 1981 [4] is frequently used nowadays in order to find the optimum C clusters of a given data of length L. It is based on the minimization of the following objective function

$$O_m = \sum_{i=1}^{L}\sum_{j=1}^{C} \mu_{ij}^m \|x_i - c_j\|^2 \qquad 1 \leq m \leq \infty \tag{3.36}$$

where m is any real number greater than 1, x_i is the i-th of N-dimensional data to cluster, c_j is the N-dimensional center of the j-th cluster and μ_{ij} is the degree of membership of x_i in the cluster j. Fuzzy partition is carried out through an iterative optimization of the objective function shown above, with the update of the degrees μ_{ij} and the cluster centers c_j given by

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|^2}{\|x_i - c_k\|^2} \right)^{\frac{2}{m-1}}} \tag{3.37}$$

$$c_j = \frac{\sum_{i=1}^{L} \mu_{ij}^m \cdot x_i}{\sum_{i=1}^{L} \mu_{ij}^m} \tag{3.38}$$

The process ends when $\max_{ij} \left\{ |\mu_{ij}^{(k+1)} - \mu_{ij}^{(k)}| \right\} < \varepsilon$ where ε is a termination criterion between 0 and 1.

It must be noticed that an initial matrix with all μ_{ij} must be randomly realized at the beginning. Thus, different initializations cause different evolutions which can converge to a different local minimum of the cost function O_m.

We have tested this method in some experiments which will be shown later with $m = 2$ and ε equal to the *desired error* parameter ($\varepsilon = 1\%$ in this case) from which we have observed two disadvatages.

The first one is the need of fixing the number of clusters. Here we suggest a number proportional to the number of the input partitions, for example the sum of the input sets.

The second problem is the high computational cost if the dimension of the matrix with μ_{ij} is very high basically if many rules must be considered.

3.6.2 Hierarchical clustering: Chiu's method

Here we present a hierarchical method proposed by S. L. Chiu in 1994 [18] which was originally developed in order to obtain the initial values for iterative optimization-based clustering methods such as Fuzzy C-means.

Nevertheless we have observed that its performance is in general very satisfactory and there is no need to combine it with other techniques in order to obtain representative clusters.

Consider N data points $x_i \mid_{1 \leq i \leq N}$. We consider each point as a potential cluster center and define a measure of the potential of each x_i as

$$P_i = \sum_{j=1}^{N} e^{-\alpha \|x_i - x_j\|^2} \tag{3.39}$$

where $\alpha = 4 \backslash r_\alpha^2$ being r_α a positive constant. The measure of potential for a data point is a function of its distances to all other data points. Thus, a data point with many neighboring data points will have a high potential value. In fact r_α is the radius which defines a significant neigborhood.

After the potential of every data point has been computed we choose the data point with the highest potential as the first cluster center. Let x_1^* be the location of the first cluster center and P_1^* its potential value. We then revise the potential of each data point x_i as

$$P_i = P_i - P_1^* e^{-\beta \|x_i - x_1^*\|^2} \tag{3.40}$$

where $\beta = 4 \backslash r_\beta^2$ being r_β a positive constant. Thus, we substract an amount of potential from each data point as a function of its distance from the first cluster center. Those points close to x_1^* will have greatly reduced its potential and therefore will unlikely be selected as the next cluster centers. Now r_β defines the reighborhood which will have measurable reductions in potential. Chiu suggests $r_\beta = 1.5 r_\alpha$ but we usually use the same value ($r_\beta = r_\alpha$).

In fact we prefer $r_\alpha = r_\beta = \varepsilon \left(\max(x) - \min(x)\right)$ and $\alpha = \beta = -\ln(\varepsilon)/{r_a}^2$ where ε is the value of the *desired error* parameter.

Once the potential of all the data points has been revised we choose the data point with the highest remaining potential as the second cluster center. We then further reduce the potential of each data point according to their distance to the last cluster center.

In general, after the k-th cluster center has been obtained, we revise the potential of each data point as

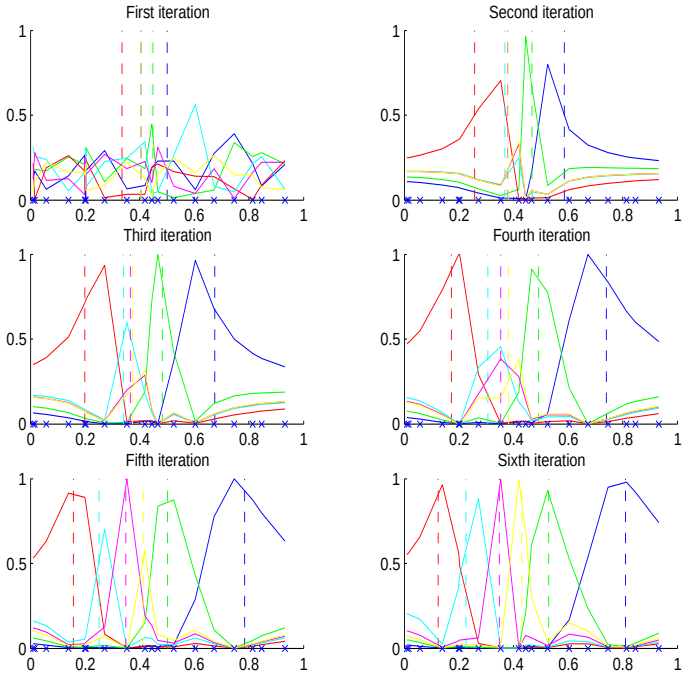
$$P_i = P_i - P_k^* e^{-\beta \|x_i - x_k^*\|^2} \tag{3.41}$$

where x_k^* is the location of the k-th cluster and P_k^* its potential value.

We stop when $P_k^* < \varepsilon P_1^*$ where ε is equal to the *desired error* parameter.

Example

In order to clarify both alternatives we can suppose an example where a set of twenty possible output sets, obtained with Wang&Mendel's or Takagi&Sugeno's methods, will be clustered in order to improve the model's intelligibility.
Suppose that these possible output sets are placed at: 0.0099, 0.0153, 0.0579, 0.1389, 0.1987, 0.1988, 0.2026, 0.2028, 0.2722, 0.3529, 0.4186, 0.4451, 0.4660, 0.5252, 0.6038, 0.6721, 0.7468, 0.8132, 0.8462 and 0.9318.
If we consider fuzzy C-means with 6 clusters, first we would generate a 6×20 matrix with random values where we could find, for each cluster, the degree of membership computed in every possible output set. Obviously the sum of the values of each column must be equal to one.
The following figure shows the first iterations with a certain initial matrix:



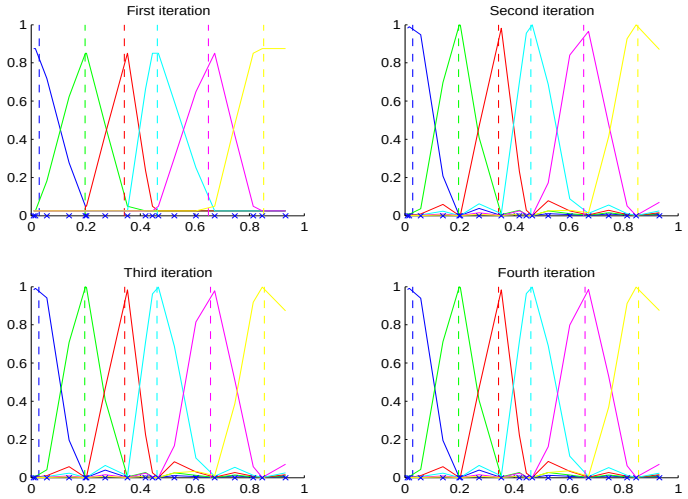
In this case the cluster centers changed as follows:

Iteration 1 →	0.3346	0.4034	0.4054	0.4057	0.4491	0.5005
Iteration 2 →	0.2574	0.3677	0.3784	0.3803	0.4681	0.5862
Iteration 3 →	0.1986	0.3407	0.3661	0.3736	0.4834	0.6748
Iteration 4 →	0.1721	0.3065	0.3533	0.3812	0.4922	0.7394
Iteration 5 →	0.1565	0.2517	0.3496	0.4133	0.5024	0.7840
Iteration 6 →	0.1245	0.2263	0.3483	0.4311	0.5281	0.8092

Otherwise if we considered Chiu's alternative we should compute the potential P of each possible point in order to choose as cluster the point with the highest P. Remember that once we have chosen a cluster then the potentials are computed again by substracting a magnitude based on its distance to the last cluster.
The first clusters are given as follows based on their potential and $\varepsilon = 10\%$:

Point	P_1	P_2	P_3	P_4	P_5	P_6
0.0099	+2.5386	+2.5383	+2.5383	-0.0603	-0.0603	-0.0603
0.0153	+2.6198	+2.6193	**+2.6193**	+0.0000	-0.0000	-0.0000
0.0579	+2.3319	+2.3106	+2.3106	+0.7092	+0.7092	+0.7092
0.1389	+2.6250	+0.8826	+0.8826	+0.8409	+0.8409	+0.8409
0.1987	+4.6088	-0.0001	-0.0001	-0.0004	-0.0004	-0.0004
0.1988	**+4.6089**	+0.0000	-0.0000	-0.0003	-0.0003	-0.0003
0.2026	+4.5999	+0.0094	+0.0094	+0.0092	+0.0092	+0.0092
0.2028	+4.5993	+0.0098	+0.0098	+0.0097	+0.0097	+0.0097
0.2722	+2.1881	+1.1161	+1.1152	+1.1152	+1.1152	+1.1152
0.3529	+1.6202	+1.6128	+1.3139	+1.3139	+1.3139	**+1.3139**
0.4186	+2.7313	+2.7313	+0.2547	+0.2547	+0.2547	+0.2547
0.4451	+2.9932	**+2.9932**	+0.0000	-0.0000	-0.0000	-0.0000
0.4660	+2.8579	+2.8579	+0.1986	+0.1986	+0.1986	+0.1986
0.5252	+1.8004	+1.8004	+1.2730	+1.2730	+1.2730	+1.2687
0.6038	+1.4804	+1.4804	+1.4771	+1.4771	+1.4771	+1.0535
0.6721	+1.5109	+1.5109	+1.5109	+1.5109	**+1.5014**	+0.0000
0.7468	+1.5968	+1.5968	+1.5968	+1.5968	+0.9683	+0.6365
0.8132	+2.0736	+2.0736	+2.0736	**+2.0736**	+0.0000	-0.0069
0.8462	+1.9502	+1.9502	+1.9502	+1.9502	+0.4079	+0.4075
0.9318	+1.1596	+1.1596	+1.1596	+1.1596	+1.1138	+1.1138

The first cluster would be 0.1988 with P=4.6089, the second cluster would be 0.4451 with P=2.9932, the third cluster would be 0.0153 with P=2.6193, the fourth cluster would be 0.8132 with P=2.0736, the fifth cluster would be 0.6721 with P=1.5014 and the sixth cluster would be 0.3529 with P=1.3139.
As explained before, the Chiu's cluster selection is compatible with fuzzy C-means because we could apply first the Chiu's method in order to define the number of clusters and also a first fuzzification matrix. In this way the fuzzy C-means converges faster as can be observed in the following figure:



Iteration 1 →	0.0304	0.1980	0.3417	0.4630	0.6494	0.8514
Iteration 2 →	0.0288	0.1969	0.3424	0.4616	0.6541	0.8530
Iteration 3 →	0.0284	0.1967	0.3424	0.4613	0.6571	0.8542
Iteration 4 →	0.0284	0.1966	0.3423	0.4614	0.6592	0.8550

Despite being compatible the use of Chiu's method with fuzzy C-means, we usually prefer to compute the possible output sets only with the Chiu's alternative and thus, we avoid fuzzy C-means because of its high computational cost.

Finally it must be noticed that the final output sets will be rounded by using the method we have explained before in order to improve the intelligibility of the model.

3.7 Criteria to stop the process

The proposed hierarchical process (linarization → input partition → output sets → model's rules) goes on until the fuzzy model is enough satisfactory in terms of the error. Obviously the less error the more partitions and the less intelligibility. Therefore, we compute different models until the error reaches the *desired error* parameter which is fixed at the beginning based on our subjective trade-off between accuracy and intelligibility.

When we talk about the error we refer to the normalized root mean square error (NRMSE). This is the empirical standard deviation of the (non mean) error normalized by the square error of the empirical variance of the model we are trying to estimate. Thus, the NRMSE is defined as equation 3.42.

$$NRMSE = \sqrt{\frac{\sum_k \left(y_{model}\left(k\right) - y_{measured}\left(k\right)\right)^2}{\sum_k \left(y_{model}\left(k\right) - mean\left(y_{measured}\right)\right)^2}} \tag{3.42}$$

This criterion was defined by D. H. Nash and B. Sutcliffe [81] together with the modeling efficiency $ME = 1 - NRMSE^2$ in order to evaluate the performance of a model in a way independent of either the scale factors or the number of data.

But in fact we take into account two criteria in order to decide if the process should finish. The first is the logical fact that the overall model reaches an error equal or lower than the *desired error* parameter. The second is the fact that all the fuzzy curves have an error equal or lower than the *desired error* parameter because then we can not increase the number of partitions by having obtained a satisfactory linearization in every fuzzy curve. We stop the process when any of both alternatives happens.

Anyway, it could occur that the best model in terms of either the error or the intelligibility might have been reached before the last iteration. Recall that the error may not be a monotonically decreasing sequence. Therefore, when the process stops, we always give as result the model with the lowest error, even if it had been obtained before the last iteration. Moreover its intelligibility must be in general better by having less partitions.

3.8 Summary

In this chapter we have detailed the method we propose in order to obtain an intelligible fuzzy model from a set of input-output data.

We have explained the basic steps: the computation of the optimal fuzzy curves for each possible input, the input partition, the choice of the possible output values and the clustering solution in order to assure intelligible models. Several alternatives were considered in some steps which have been compared in order to offer criteria when applying the method.

We have also explained some aspects which are very important if we seek intelligible models, like the proper use of rounded numbers or the special consideration of the odd fuzzy curves.

Furthermore, we have chosed the different solutions by considering not only the computational cost, but the necessary criteria to satisfy the intelligible models explained in the previous chapter.

One should take into account that some aspects related to the search of an efficient computation have not been explained in order to avoid the dispersion of the explanations. For example, we have not mentioned that the points of the linearized fuzzy curve are only computed between the boundaries of the segment having the maximum error because the rest of points do not change their value and are already stored in memory. Anyway, this and other aspects can be found in the appendix where the whole algorithm is provided.