2

Constructing Fuzzy Models from Input-Output Data

In the previous chapter we presented a discussion of the approximation capabilities of fuzzy models. In summary, we have shown that fuzzy models can be used to reproduce the behavior of any continuous function. This chapter presents some of the methods used to construct fuzzy models that replicate the behavior of a given function. The information about the function is presented in the form of input–output data, which means that a set of points over the domain of the function (input) is selected and then evaluated in the function (output).

The construction of fuzzy models involves the selection of several parameters: position, shape and the distribution of the membership functions, rule base construction, selection of the logical operations, consequences of the rules, etc. This large number of "degrees of freedom" makes it very difficult to implement a unique method to select all these parameters at once. A typical approach is to set in advance the logical operations and the type of membership functions using certain criteria (differentiability, linguistic integrity, implementability, etc.). The remaining parameters can be estimated from the data using different strategies, but in general all are based on a single objective, which is to minimize the approximation error between the output values and the values given by the fuzzy model.

According to the tuned parameters and the strategies, different methods have been proposed in the literature. This chapter presents the following strategies:

- Mosaic or table lookup scheme [18]
- Using gradient descent [18] [19]
- Using clustering and gradient descent [12] [4]
- Using evolutionary strategies [20] [21]

The *mosaic or table lookup scheme* fixes in advance the type, number and position of the membership functions and calculates only the consequences of the rules. The methods based on *gradient descent* fix in advance the type and number of the membership functions and calculate their positions and the value of the consequences. The methods based on *clustering and gradient*

descent fix only the type of membership functions and by means of clustering algorithms select the number and initial positions of the membership functions. Consequences and refined positions of the membership functions are found by means of a gradient descent algorithm.

Evolutionary strategies deserve a different comment since they can be used to optimize all possible aspects integrated in a fuzzy model including the set of inputs used to construct the model. Some interesting features of the *evolutionary strategies* are the fact that they can introduce complex constraints to enforce some desired features into the model and also the fact that they perform a gradient-free optimization.

Table 2.1 summarizes the methods and the parameters that are adjusted by the method. The following sections are dedicated to explain these methods. Finally, the chapter closes with an example of an industrial application of the fuzzy models constructed from input–output data.

Table 2.1. Parameters Adjusted by the Different Training Methods

Method	**Type of MFs**	**Number of MFs**	**Location of the MFs**	**Consequences**
Mosaic scheme	Fixed	Fixed	Fixed	Adjusted
Gradient descent	Fixed	Fixed	Adjusted	Adjusted
Clustering + gradient descent	Fixed	Adjusted	Adjusted	Adjusted
Evolutionary strategies (1)	Adjusted	Adjusted	Adjusted	Adjusted

Summary:
Fuzzy inference systems (FIS) can be systematically constructed from "pure" input–output data. All methods are based on the optimization of a cost function to minimize the "distance" between the predictions of the FIS and the output data. The main differences among the methods are the initialization and the adjusted parameters.

2.1 Mosaic or Table Lookup Scheme

The basic scheme of the method was proposed by Wang [18]. Here some simple modifications are introduced, and these modifications are related to the consequence calculation. In this method the position, the shape and the distribution of the membership functions are choices for the designer. The rule base is composed and the method finds only the consequences of the rules.

Assume a sequence of input–output $\{x^i, y^i\}\, i = 1, \ldots, N$ data is collected, the inputs $x^i \in U \subset \Re^p$ and the output $y^i \in V \subset \Re$. The subset U is a portion of the space $\Re^p$ and is defined as $U = [a_1, b_1] \times \ldots \times [a_p, b_p]$. The procedure to construct the model is laid out in the following.

- For each of the p inputs of the system distribute over the interval $[a_i, b_i]$ N_i membership functions. The shape, the position and the distribution is a user's choice. The only condition is that the full interval is covered and at least two membership functions are placed on each point of the input domain. As shown in the previous sections, the shape and the distribution affect the smoothness and the accuracy of the approximation.
- Generate the rule base using all possible combinations among the antecedents and the AND operator (choosing in advance "min" or "product" operator). The rule l of the rule base for Mamdani fuzzy systems is

$$\text{IF } x_1^i \text{ is } A_1^l \text{ AND } \ldots \text{ AND } x_p^i \text{ is } A_p^l \text{ THEN } y \text{ IS } \bar{y}^l$$

and for Takagi–Sugeno fuzzy systems

$$\text{IF } x_1^i \text{ is } A_1^l \text{ AND } \ldots \text{ AND } x_p^i \text{ is } A_p^l \text{ THEN } y = a_1^l x_1^i + \ldots + a_p^l x_p^i + b^l$$

- Calculate the inference of each rule. For rule l of the form

$$\mu_l(x^i) = \min\{\mu_l^1(x_1^i), \mu_l^2(x_2^i), \ldots, \mu_l^p(x_p^i)\} \tag{2.1}$$

or

$$\mu_l(x^i) = \mu_l^1(x_1^i).\mu_l^2(x_2^i).\ldots.\mu_l^p(x_p^i) \tag{2.2}$$

the general expressions for these fuzzy system with L rules will be given by

$$f(x^i) = \frac{\sum_{l=1}^{L} \bar{y}^l \mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} \tag{2.3}$$

for the Mamdani models and

$$f(x^i) = \frac{\sum_{l=1}^{L} (a_1^l x_1^i + \ldots + a_p^l x_p^i + b^l)\mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} \tag{2.4}$$

for Takagi–Sugeno models.
- Calculate the consequence parameters
  - In the Mamdani model the parameter to be calculated is $\bar{y}^l$ $l = 1, \ldots, L$ such that $f(x^i) \approx y^i$. Observe that Equation (2.3) can be written as

$$f(x^i) = \sum_{l=1}^{L} \bar{y}^l w_l(x^i) \tag{2.5}$$

$$w_l(x^i) = \frac{\mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} = w_l^i \tag{2.6}$$

The N output values can be represented as the vector Y in terms of the inference process:

$$\underbrace{\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} w_1^1 & w_2^1 & \dots & w_L^1 \\ w_1^2 & w_2^2 & \dots & w_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^N & w_2^N & \dots & w_L^N \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} \bar{y}^1 \\ \bar{y}^2 \\ \vdots \\ \bar{y}^L \end{bmatrix}}_{\theta} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_{E} \tag{2.7}$$

– In the Takagi–Sugeno model the parameters to be calculated are $a_1^l \dots a_p^l$ and b^l $l = 1, \dots, L$ such that $f(x^i) \approx y^i$. Using the reasoning applied for the Mamdani models, Equation (2.4) can be written as

$$f(x^i) = \sum_{l=1}^{L} (a_1^l x_1^i + \dots + a_p^l x_p^i + b^l) w_l(x^i) \tag{2.8}$$

where $w_l(x^i)$ has the same form shown in Equation (2.6).
The N output values can be represented as the vector Y in terms of the inference process

$$\underbrace{\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} w_1^1 x_1^1 & \dots & w_1^1 x_p^1 & w_1^1 & w_2^1 x_1^1 & \dots & w_L^1 x_p^1 & w_L^1 \\ w_1^2 x_1^1 & \dots & w_1^2 x_p^1 & w_1^2 & w_2^2 x_1^1 & \dots & w_L^2 x_p^1 & w_L^2 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_1^N x_1^1 & \dots & w_1^N x_p^1 & w_1^N & w_2^N x_1^1 & \dots & w_L^N x_p^1 & w_L^N \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} a_1^1 \\ a_2^1 \\ \vdots \\ a_p^1 \\ b^1 \\ a_1^2 \\ \vdots \\ a_p^L \\ b^L \end{bmatrix}}_{\theta}$$

$$+ \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}}_{E} \tag{2.9}$$

In both cases the vector E is the approximation error and the aim is to reduce the norm of this vector as much as possible. Using the quadratic norm to measure the approximation error, we obtain

$$\min_{\theta} ||E||_2 = \min_{\theta} ||Y - W\theta||_2 \tag{2.10}$$

It is a least-squares problem and the consequences can be calculated using least squares. The solution to this least-squares problem is

$$\theta = \arg\min_{\theta} ||E||_2 = (W^T W)^{-1} Y^T W \tag{2.11}$$

This solution is applicable as far as the $\text{rank}(W^T W) = \dim(\theta)$; otherwise other methods must be applied to guarantee a reliable set of consequences for the rules. In Section 2.5, a method based on recursive least squares is detailed.

Summary:
A mosaic or table lookup scheme is probably the simplest method to construct fuzzy models from data. The method demands from the user the definition of the antecedent of the rules and finds the consequences by using least squares.

2.1.1 Illustrative Example

In this example we show a simple application of the method *mosaic or table lookup scheme* to approximate the function $f(x) = sin(x)$ over the interval $[0, 2\pi]$ using 629 points equidistant along the domain of x. In this case we illustrate the results using six membership functions over the input domain. Four models are presented: three of the Mamdani type and one Takagi–Sugeno. The three Mamdani models are created with three different types of membership functions: *triangular*, *polynomial* and *Gaussian*. For the model using Takagi–Sugeno rules only the results using *triangular* membership are illustrated.

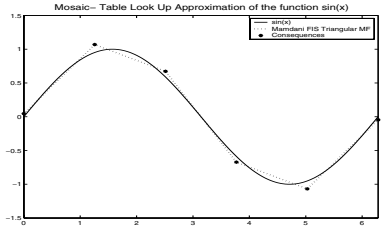
Observe that the interpolations generated by the Mamdani models are mentioned in previous Chapter: linear for the *triangular* membership functions (see Figure 2.1), *polynomial* for the polynomial membership functions (see Figure 2.2) and between the *neighborhoods* of the consequences for the Gaussian membership functions.

The best model is by far the Takagi–Sugeno model (see Figure 2.4). In fact, in the figure it is difficult to distinguish the approximation from the original function. Figure 2.4 shows some straight segments corresponding to the consequences of the rules. The successful result of the Takagi–Sugeno can be explained in part because the model exhibits 12 degrees of freedom (two adjustable parameters per rule a_1^l and b^l) in contrast with the Mamdani models with only 6 degrees of freedom (only one adjustable parameter per rule $\bar{y}^l$). Having more degrees of freedom can be beneficial as long as the number of points is large enough and as long as they are well spread over the input domain (*persistent excitation*). Otherwise the generalization capabilities of the model can be compromised.

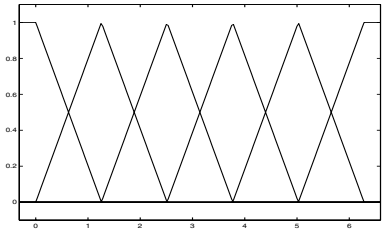
Among the Mamdani models the best model is the model generated using *Gaussian* membership functions. The result is explained in part for the

resemblance between the shapes of the sine function and the Gaussian membership functions. Also, it is interesting to observe how the "flat" sections of the polynomial membership functions affect the approximation by generating local plateaus in the function approximation.

Observe that the results of this example are only an illustration of the method and are by no means a benchmark to judge the capabilities of some membership functions or model types.



(a)



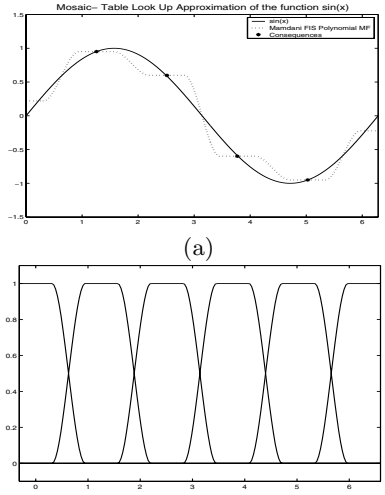
(b)

Figure 2.1. (a) Approximation generated by a Mamdani fuzzy model trained using the *mosaic or table lookup scheme* using triangular membership functions with 6 membership functions. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions

2.2 Using Gradient Descent

This method requires the definition of the number of membership functions and their shape. Normally the AND function is fixed to be the "product" because an analytical expression for the gradient of the cost function is needed. The initial position of the membership functions is another element that must be chosen. The method proceeds as follows:

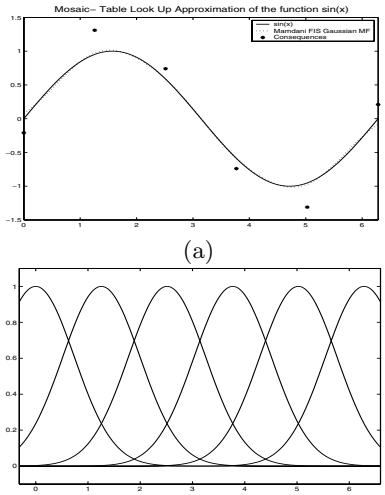
- For each of the p inputs of the system, distribute over the interval $[a_i, b_i]$, N_i membership functions. The shape, the initial positions and the distribution are user's choices. The membership functions must cover the input



(a)

(b)

Figure 2.2. (a) Approximation generated by a Mamdani fuzzy model trained using the *mosaic or table lookup scheme* using polynomial membership functions with 6 membership functions. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions



(a)

(b)

Figure 2.3. (a) Approximation generated by a Mamdani fuzzy model trained using the *mosaic or table lookup scheme* using polynomial membership functions with 6 membership functions. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions

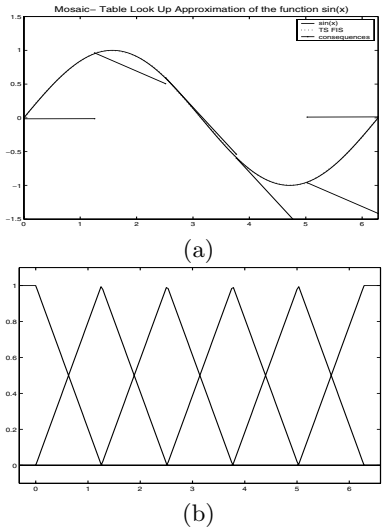


Figure 2.4. (a) Approximation generated by a Takagi–Sugeno fuzzy model trained using the *mosaic or table lookup scheme* using triangular membership functions with 6 membership functions. (-) Original function (- -) Approximation generated by the fuzzy model (.-) Consequences of the rules (b) Membership functions

interval, and at least two membership functions should be placed on each input domain.

- Generate the rule base using all possible combinations among the antecedents and the AND operator using "product".
- Initialize the value of the consequences using prior knowledge, least squares or recursive least squares.
- Optimize the value of the consequences $\bar{y}^l$ and the parameters of the membership functions. The criteria will be to minimize the cost function described in the previous section, but now the optimization will also adjust the membership functions of the antecedents. The cost function can be described as

$$J = \frac{1}{2}\sum_{i=1}^{N}(y^i - f(x^i, \theta))^2 \tag{2.12}$$

where θ is a vector representing all the "adjustable" parameters (consequences, parameters of the membership functions) of the fuzzy system $f(.,.)$. The problem will be the minimization of the cost function J. This minimization is a nonlinear, nonconvex optimization problem. The objective is to obtain an "acceptable" solution and not necessarily "the global minima" of this cost function. Different schemes for optimization can be applied to find this solution. Probably the simplest one will be the gradient descent method. This method consists of an iterative calculation of the parameters oriented to the negative direction of the gradient. The ex-

planation behind this method is that by taking the negative direction of the gradient, the steepest route toward the minimum will be taken. This descent direction does not guarantee convergence of the scheme; for this reason, the α parameter is introduced and it can be modified to improve the convergence rate and properties. Some choices of α are given by Newton and quasi-Newton methods [22].

$$\theta(k+1) = \theta(k) + \alpha \frac{\partial J}{\partial \theta} \tag{2.13}$$

α is sometimes called the "learning rate." The gradient descent method can be modified; for example, by calculating the consequences by means of least squares (see ANFIS scheme [19]).

The gradient of the cost function will be in general

$$\frac{\partial J}{\partial \theta} = \sum_{i=1}^{N} (y^i - f(x^i, \theta)) \frac{\partial f(x^i, \theta)}{\partial \theta} = \sum_{i=1}^{N} e^i \frac{\partial f(x^i, \theta)}{\partial \theta} \tag{2.14}$$

Using the general expression of the fuzzy system can be parameterized as

$$f(x^i) = \frac{\sum_{l=1}^{L} \bar{y}^l \mu_l(x^i)}{\sum_{l=1}^{L} \mu_l(x^i)} = \frac{A}{B} \tag{2.15}$$

The updating of the consequence parameters will be independent of the parameterization of the membership functions and will be given by

$$\bar{y}^l(k+1) = \bar{y}^l(k) + \alpha \sum_{i=1}^{N} (y - f(x^i)) \frac{\mu_l(x^i)}{B} \tag{2.16}$$

The expressions to update the parameters of the membership functions are different for each parameterization. Special attention is devoted to the gradient calculation to guarantee a 0.5 overlap between contiguous membership functions. The updating formulas for some of the membership functions are shown in the next sections.

Summary:
The *gradient descent* method calculates parameters on the antecedents and the consequences of the fuzzy inference system. The method demands from the user the definition of the initial location of the membership functions of the antecedents. The method can be combined with a calculation of the consequences by using least squares. In this case the method is known as the ANFIS scheme. Such a method exhibits faster convergence, especially for Takagi–Sugeno models.

2.2.1 Gradient Updating for Trapezoidal Membership Functions

Assuming the parameterization given in the expression

$$\mu_j^i(x_i, a_j^i, b_j^i, c_j^i, d_j^i) = \min\left[\max\left(\frac{x_i - a_j^i}{b_j^i - a_j^i}, 0\right), \max\left(1 - \frac{x_i - c_j^i}{d_j^i - c_j^i}, 0\right), 1\right] \tag{2.17}$$

the updating formulas will be

$$a_j^i(k+1) = a_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} \tag{2.18}$$

$$b_j^i(k+1) = b_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} \tag{2.19}$$

$$c_j^i(k+1) = c_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} \tag{2.20}$$

$$d_j^i(k+1) = d_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} \tag{2.21}$$

where the set $\mathcal{U}$ is the set of rules that includes the function $\mu_j^i(x)$ in the antecedents and

$$\frac{\partial \mu_j^i(x_i^t)}{\partial a_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ \frac{x_i^t - b_j^i}{(b_j^i - a_j^i)^2} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \tag{2.22}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ \frac{a_j^i - x_i^t}{(b_j^i - a_j^i)^2} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \tag{2.23}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ \frac{d_j^i - x_i^t}{(d_j^i - c_j^i)^2} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \tag{2.24}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ \frac{x_i^t - c_i^j}{(d_i^j - c_j^i)^2} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \tag{2.25}$$

It is important to remark that the updating should preserve the condition $a_j^i \leq b_j^i \leq c_j^i \leq d_j^i$. This adaptation rule can be applied to triangular membership functions by just making $b_j^i = c_j^i$.

2.2.2 Gradient Updating for Triangular Membership Functions with Overlap $\frac{1}{2}$

The membership functions are parameterized by using only their modal values. This parameterization not only preserves the overlap but also reduces the number of parameters to be tuned. Triangular membership functions are parameterized by the position of their three vertices; but the condition of overlap $\frac{1}{2}$ makes the lower right vertex of one membership function to be at the same position as the modal value of the next membership function. So, instead of tuning three parameters (the vertices), only one parameter is tuned for each membership function.

The parameterization for a triangular membership function using the modal values as parameters is

$$\mu_j^i(x_i, m_{j-1}^i, m_j^i, m_{j+1}^i) = \max\left[0, \min\left(\frac{x_i - m_{j-1}^i}{m_j^i - m_{j-1}^i}, 1 - \frac{x_i - m_j^i}{m_{j+1}^i - m_j^i}\right)\right] \tag{2.26}$$

The updating formula will be

$$\begin{aligned} m_j^i(k+1) = m_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \\ \left[\sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j-1}^i(x_i^t)} \frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} + \sum_{l \in \mathcal{V}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i}\right. \\ \left. + \sum_{l \in \mathcal{W}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j+1}^i(x_i^t)} \frac{\partial \mu_{j+1}^i(x_i^t)}{\partial m_j^i}\right] \end{aligned} \tag{2.27}$$

where the sets $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ are the set of rules that includes the functions $\mu_{j-1}^i(.)$, $\mu_j^i(.)$ and $\mu_{j+1}^i(.)$, respectively, and with

$$\frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} = \begin{cases} 0 & \text{if } x_i^t < m_{j-1}^i \\ \frac{x_i^t - m_j^i}{(m_j^i - m_{j-1}^i)^2} & \text{if } m_{j-1}^i < x_i^t < m_j^i \\ 0 & \text{if } x_i^t > m_j^i \end{cases} \tag{2.28}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i} = \begin{cases} 0 & \text{if } x_i^t < m_{j-1}^i \\ \frac{m_{j-1}^i - x_i^t}{(m_j^i - m_{j-1}^i)^2} & \text{if } m_{j-1}^i < x_i^t < m_j^i \\ \frac{m_{j+1} - x_i^t}{(m_{j+1}^i - m_j^i)^2} & \text{if } m_j^i < x_i^t < m_{j+1}^i \\ 0 & \text{if } x_i^t > m_{j+1}^i \end{cases} \tag{2.29}$$

$$\frac{\partial \mu^i_{j+1}(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_j \\ \frac{x^t_i - m^i_{j+1}}{(m^i_{j+1} - m^i_j)^2} & \text{if } m^i_j < x^t_i < m^i_{j+1} \\ 0 & \text{if } x^t_i > m^i_{j+1} \end{cases} \tag{2.30}$$

Here the adaptation must be constrained such that the condition $m^i_j \leq m^i_{j+1}$ is preserved.

2.2.3 Gradient Updating for Polynomial Membership Functions

Assuming the parameterization given in the expression

$$\mu^i_{j_i}(x_i) = \begin{cases} 0 & \text{if } x_i < a^i_{j_i} \\ (x_i - a^i_{j_i})^2(2x_i + a^i_{j_i} - 3b^i_{j_i})/(a^i_{j_i} - b^i_{j_i})^3 & \text{if } a^i_{j_i} < x_i < b^i_{j_i} \\ 1 & \text{if } b^i_{j_i} \leq x_i \leq c^i_{j_i} \\ -(x_i - d^i_{j_i})^2(2x_i + d^i_{j_i} - 3c^i_{j_i})/(c^i_{j_i} - d^i_{j_i})^3 & \text{if } c^i_{j_i} < x_i < d^i_{j_i} \\ 0 & \text{if } x_i > d^i_{j_i} \end{cases} \tag{2.31}$$

the expressions to update the parameters a^i_j, b^i_j, c^i_j and d^i_j are similar to the ones used for the trapezoidal membership functions. Only the expression for the gradient of the membership functions changes.

$$a^i_j(k+1) = a^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu^i_j(x^t_i)} \frac{\partial \mu^i_j(x^t_i)}{\partial a^i_j} \tag{2.32}$$

$$b^i_j(k+1) = b^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu^i_j(x^t_i)} \frac{\partial \mu^i_j(x^t_i)}{\partial b^i_j} \tag{2.33}$$

$$c^i_j(k+1) = c^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu^i_j(x^t_i)} \frac{\partial \mu^i_j(x^t_i)}{\partial c^i_j} \tag{2.34}$$

$$d^i_j(k+1) = d^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu^i_j(x^t_i)} \frac{\partial \mu^i_j(x^t_i)}{\partial d^i_j} \tag{2.35}$$

with

$$\frac{\partial \mu^i_j(x^t_i)}{\partial a^i_j} = \begin{cases} 0 & \text{if } x^t_i < a^i_j \\ 6\frac{(b^i_j - x^t_i)^2(a^i_j - x^t_i)}{(a^i_j - b^i_j)^4} & \text{if } a^i_j < x^t_i < b^i_j \\ 0 & \text{if } x^t_i > b^i_j \end{cases} \tag{2.36}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial b_j^i} = \begin{cases} 0 & \text{if } x_i^t < a_j^i \\ -6\frac{(a_j^i - x_i^t)^2(b_j^i - x_i^t)}{(a_j^i - b_j^i)^4} & \text{if } a_j^i < x_i^t < b_j^i \\ 0 & \text{if } x_i^t > b_j^i \end{cases} \tag{2.37}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial c_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ -6\frac{(d_j^i - x_i^t)^2(c_j^i - x_i^t)}{(c_j^i - d_j^i)^4} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \tag{2.38}$$

$$\frac{\partial \mu_j^i(x_i^t)}{\partial d_j^i} = \begin{cases} 0 & \text{if } x_i^t < c_j^i \\ 6\frac{(c_j^i - x_i^t)^2(d_j^i - x_i^t)}{(c_i^j - d_j^i)^4} & \text{if } c_j^i < x_i^t < d_j^i \\ 0 & \text{if } x_i^t > d_j^i \end{cases} \tag{2.39}$$

The adaptation algorithm should preserve the condition $a_j^i \leq b_j^i \leq c_j^i \leq d_j^i$.

2.2.4 Gradient Updating for Polynomial Membership Functions with Overlap $\frac{1}{2}$ and $b_j^i = c_j^i = m_j^i$

The parameterization of the membership functions is made using only their modal values. This parameterization guarantees the overlap of $\frac{1}{2}$ with the neighboring membership functions. The number of adjusted parameters is reduced: instead of adjusting four parameters $(a_j^i, b_j^i, c_j^i, d_j^i)$, for each membership function, only one parameter m_j^i is adjusted. Observe that the overlap $\frac{1}{2}$ is preserved only if the parameters $a_{j+1}^i, b_j^i, c_j^i, d_{j-1}^i$ describing the polynomial membership function are equal among each other and equal to the modal value m_j^i. The parameterization using the modal values is as follows:

$$\mu_{j_i}^i(x_i) = \begin{cases} 0 & \text{if } x_i < m_{j-1}^i \\ \frac{(x_i - m_{j-1}^i)^2(2x_i + m_{j-1}^i - 3m_j^i)}{(m_{j-1}^i - m_j^i)^3} & \text{if } m_{j-1}^i < x_i < m_j^i \\ \frac{-(x_i - m_{j+1}^i)^2(2x_i + m_{j+1}^i - 3m_j^i)}{(m_j^i - m_{j+1}^i)^3} & \text{if } m_j^i < x_i < m_{j+1}^i \\ 0 & \text{if } x_i > m_{j+1}^i \end{cases} \tag{2.40}$$

The parameters m_j^i are updated with a similar formula as the one used for the triangular membership functions with overlap $\frac{1}{2}$:

$$\begin{aligned} m_j^i(k+1) = m_j^i(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \\ \left[\sum_{l \in \mathcal{U}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j-1}^i(x_i^t)} \frac{\partial \mu_{j-1}^i(x_i^t)}{\partial m_j^i} + \sum_{l \in \mathcal{V}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_j^i(x_i^t)} \frac{\partial \mu_j^i(x_i^t)}{\partial m_j^i} \right. \\ \left. + \sum_{l \in \mathcal{W}} (\bar{y}^l - f(x^t)) \frac{\mu_l(x^t)}{\mu_{j+1}^i(x_i^t)} \frac{\partial \mu_{j+1}^i(x_i^t)}{\partial m_j^i} \right] \end{aligned} \tag{2.41}$$

where the sets $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{W}$ are the set of rules that includes the functions $\mu^i_{j-1}(.)$, $\mu^i_j(.)$ and $\mu^i_{j+1}(.)$, respectively, and with

$$\frac{\partial \mu^i_{j-1}(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_{j-1} \\ 6\frac{(m^i_{j-1}-x^t_i)^2(m^i_j-x^t_i)}{(m^i_{j-1}-m^i_j)^4} & \text{if } m^i_{j-1} < x^t_i < m^i_j \\ 0 & \text{if } x^t_i > m^i_j \end{cases} \tag{2.42}$$

$$\frac{\partial \mu^i_j(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_{j-1} \\ -6\frac{(m^i_{j-1}-x^t_i)^2(m^i_j-x^t_i)}{(m^i_{j-1}-m^i_j)^4} & \text{if } m^i_{j-1} < x^t_i < m^i_j \\ -6\frac{(m^i_{j+1}-x^t_i)^2(m^i_j-x^t_i)}{(m^i_j-m^i_{j+1})^4} & \text{if } m^i_j < x^t_i < m^i_{j+1} \\ 0 & \text{if } x^t_i > m^i_{j+1} \end{cases} \tag{2.43}$$

$$\frac{\partial \mu^i_{j+1}(x^t_i)}{\partial m^i_j} = \begin{cases} 0 & \text{if } x^t_i < m^i_j \\ 6\frac{(m^i_{j+1}-x^t_i)^2(m^i_j-x^t_i)}{(m^i_j-m^i_{j+1})^4} & \text{if } m^i_j < x^t_i < m^i_{j+1} \\ 0 & \text{if } x^t_i > m^i_{j+1} \end{cases} \tag{2.44}$$

Observe that the adaptation must preserve the condition $m^i_j \leq m^i_{j+1}$.

2.2.5 Gradient Updating for Gaussian Membership Functions

The parameterization of the membership functions is given by

$$\mu^i_j(x^t_i) = \exp(-(\frac{x^t_i - \bar{x}^i_j}{\sigma^i_j})^2) \tag{2.45}$$

The updating formula for the parameters $\bar{x}^i_j$ and σ^i_j will be given by

$$\bar{x}^i_j(k+1) = \bar{x}^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} 2(\bar{y}^l - f(x^t))\mu_l(x^t)\frac{x^t_i - \bar{x}^i_j}{{\sigma^i_j}^2} \tag{2.46}$$

$$\sigma^i_j(k+1) = \sigma^i_j(k) + \alpha \sum_{t=1}^{N} \frac{(y^t - f(x^t))}{B} \sum_{l \in \mathcal{U}} 2(\bar{y}^l - f(x^t))\mu_l(x^t)\frac{(x^t_i - \bar{x}^i_j)^2}{{\sigma^i_j}^3} \tag{2.47}$$

where $\mathcal{U}$ is the set of rules with the antecedent term $\mu^i_j(.)$.

2.2.6 Illustrative Example

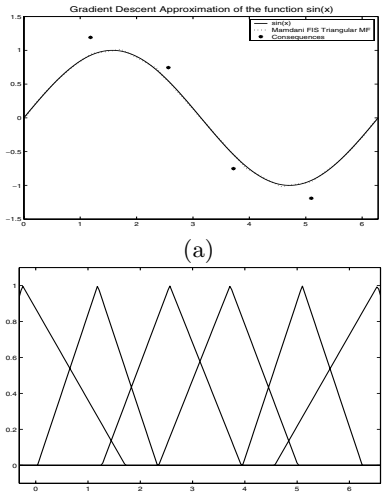
This example uses the same simple sine function presented in Section 2.1.1. The same 629 equidistant points were used to approximate the function $f(x) = sin(x)$ over the interval $[0, 2\pi]$. In this case the number of membership functions is 6 and they were initially equally distributed along the

input domain in the same way as in the example of Section 2.1.1. Again, we prepared four models: three of the Mamdani type and one Takagi–Sugeno. The three Mamdani models were created with three different types of membership functions *triangular*, *polynomial* and *Gaussian*, and they were trained during 400 iterations (epochs) using pure gradient descent. For the model using Takagi–Sugeno rules, only the results using *triangular* membership are illustrated. The Takagi–Sugeno model was trained during 400 iterations using a combination of *gradient descent* and *least squares* (ANFIS Scheme [19]). The ANFIS scheme was more efective in the Takagi–Sugeno scheme showing a faster convergence. For the Mamdani models, the use of ANFIS or "pure" *gradient descent* did not show major differences.

Observe that all the approximations are better than the approximations given by the models obtained with the method of *mosaic or table lookup*. The Mamdani model with *triangular* membership functions improves the approximation by extending the overlap of the most external membership functions (see Figure 2.5). Observe that the function no longer crosses the points of the consequences and the interpolation is no longer linear, all because the overlap of the membership functions is no longer $\frac{1}{2}$.

On the other hand, the Mamdani models using *polynomial* and *Gaussian* membership functions improve the approximation by narrowing the central membership functions and putting their centers closer (see Figures 2.6 and 2.7). The improvement shown by the approximation using polynomial membership functions (see Figure 2.6) is very remarkable compared with the approximation obtained with the simple *mosaic or table lookup* method.
The Takagi–Sugeno model shows again a good approximation with some improvement as shown in Table 2.2, but compared with the other models the improvement brought by the gradient descent method was not as significant as it was for the other models. However, observe that even that the membership functions did not have significant changes; the functions describing the consequences show completely different slopes.

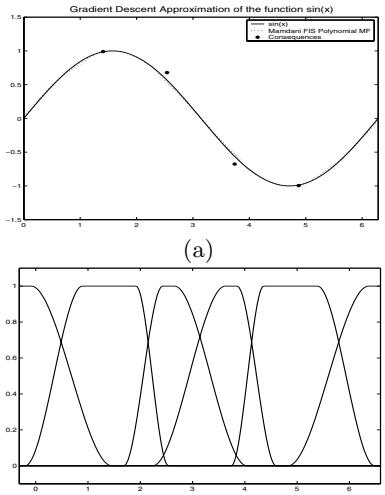
In general, the improvement in the approximation provided by the tuning of the membership functions using the *gradient descent* method is clear. The observed improvement, which in one case (Mamdani polynomial model) was of almost two orders of magnitude, is partially explained by the increased number of degrees of freedom (consequences + parameters of the membership functions) introduced in the gradient descent method (see Table 2.2).



(a)

(b)

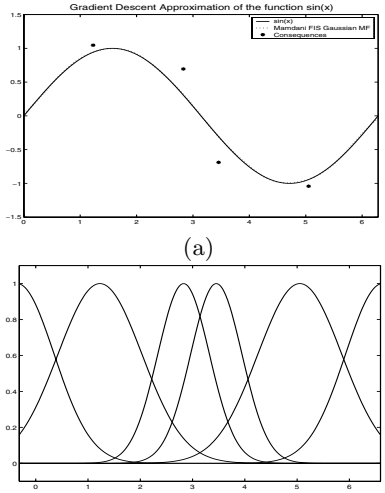
Figure 2.5. (a) Approximation generated by a Mamdani fuzzy model trained using the *gradient descent method* using triangular membership functions with 6 membership functions initially equally spaced. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions after training



(a)

(b)

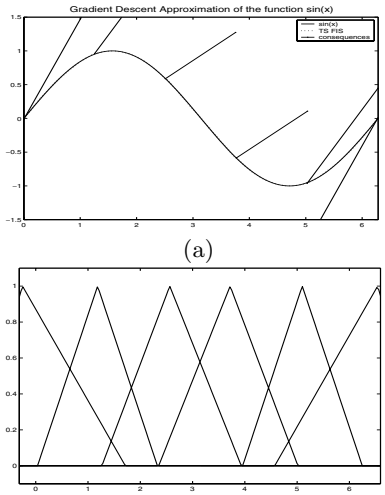
Figure 2.6. (a) Approximation generated by a Mamdani fuzzy model trained using the *gradient descent method* using polynomial membership functions with 6 membership functions initially equally spaced. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions after training



(a)

(b)

Figure 2.7. (a) Approximation generated by a Mamdani fuzzy model trained using the *gradient descent method* using polynomial membership functions with 6 membership functions initially equally spaced. (-) Original function (- -) Approximation generated by the fuzzy model (*) Consequences of the rules (b) Membership functions after training



(a)

(b)

Figure 2.8. (a) Approximation generated by a Takagi–Sugeno fuzzy model trained using the *mosaic or table lookup scheme* using triangular membership functions with 6 membership functions equally spaced. (-) Original function (- -) Approximation generated by the fuzzy model (.-) Consequences of the rules (b) Membership functions after training

Table 2.2. Approximation Error ($\sum_{i=1}^{629}[f(x) - \hat{f}(x)]^2$)of the Different Models Trained with the *Mosaic or Table Lookup* Scheme and the *Gradient Descent* Method for 400 Steps.

Model	Table look up		Gradient Descent	
	Error	DOF	Error	DOF
Mamdani triangular M.F.	1.3392	6	0.0615	24
Mamdani Gaussian M.F.	0.2378	6	0.0449	18
Mamdani polynomial M.F.	12.9666	6	0.1755	30
Tak.Sug. triangular M.F.	0.0206	12	0.0117	30

DOF=degrees of freedom number of adjustable parameters

2.3 Using Clustering and Gradient Descent

The methods studied so far had placed the fuzzy sets of the input domains on their initial positions according to the choice made by the designer (typically equally distributed). Two choices has been made by the designer – the number of membership functions and their initial distribution. The methods based on clustering aim to obtain both parameters at the same time, the number of fuzzy sets needed to make the function approximation and their distribution along the input domains.

The methods based on clustering are considered as data-driven methods. The main idea of these methods is to find structures (clusters) among the data according to their distribution in the space of the function and assimilate each cluster as a multidimensional fuzzy set representing a rule. The cluster prototypes can be either a point (to construct *Mamdani* models) or a hyperplane (to construct *Takagi–Sugeno* models).

The fuzzy inference system is constructed by means of projecting the clusters into the input space and approximating the projected cluster with a one-dimensional fuzzy set. The advantage of these methods is that they generate automatically the membership functions, leaving as the user's choices only the parameters of the clustering algorithms (number of clusters and distance function). According to the type of model to be constructed the method will be slightly different. Here is a summary of the methods:

2.3.1 Algorithm for Mamdani Models

- Collect the data and construct a set of vectors $Z^t = \{x^{t^T}, y^t\}$ where x^t and y^t are, respectively, the inputs and the output of the function. Observe that here we assume $x^t \in \Re^n$ and $y^t \in \Re$.
- Search for clusters using the Fuzzy C-means algorithm [2] or the mountain-clustering algorithm [4] for problems where the dimension of the input

space is small. Appendix B includes a description of the mentioned algorithms.
- Project the membership functions from the partition matrix U into the input space.
- Approximate the projected membership function using convex membership functions (triangular, Gaussian, polynomial, trapezoidal, etc.)
- Construct the rules with the projected membership functions.
- Calculate the consequences using recursive least squares.
- Adjust the parameters of the antecedents (if needed) using gradient descent.

2.3.2 Algorithm for Takagi–Sugeno Models

- Collect the data and construct a set of vectors $Z^t = \{x^{t^T}, y^t\}$ where x^t and y^t are, respectively, the inputs and the output of the function. Observe that here we assume $x^t \in \Re^n$ and $y^t \in \Re$.
- Search for clusters using the Gustafson and Kessel (G-K) algorithm [3]. Appendix B describes the G-K algorithm.
- Check for similarities among the clusters. Do two clusters describe a similar hyperplane?
- Project the membership functions from the partition matrix U into the input space.
- Approximate the projected membership function using convex membership functions (triangular, Gaussian, polynomial, trapezoidal, etc.)
- Construct the rules with the projected membership functions.
- Generate the consequences using the covariance matrices of each cluster.
- Calculate the consequences that are not covered by the clusters using recursive least squares.
- Adjust the parameters of the antecedents (if needed) using gradient descent.

Summary:
The *clustering + gradient descent* method calculates the initial location of the membership function by projecting the partition matrices obtained from a clustering applied to the input–output data. The consequences are generated from the centers of the clusters and for the Takagi–Sugeno models from the centers and their covariance matrices. The parameters can be refined to improve the approximation by applying gradient descent.

2.3.3 Illustrative Example

This example uses the same function ($f(x) = sin(x)$) presented in Sections 2.1.1 and 2.2.6. The data are composed of 629 equidistant points that were used to approximate the function $f(x) = sin(x)$ over the interval $[0, 2\pi]$. In this case the models were constructed based in two clustering procedures.

Fuzzy C-Means to construct Mamdani models and Gustafson and Kessel to construct a Takagi–Sugeno model. For both procedures the number of clusters selected *a priori* was 6 and the stopping criteria $\epsilon = 5 \times 10^{-5}$. This selection was made such that the results are comparable with the ones shown in previous examples. Both clustering algorithms were executed and they generated the clusters shown in Figures 2.9(a) and 2.10(a). Observe that the cluster of the G-K algorithm are characterized by their center and "main direction" of its covariance matrix. The partition matrix was projected over the input domain obtaining the membership functions shown in Figures 2.9(b) and 2.10(b).

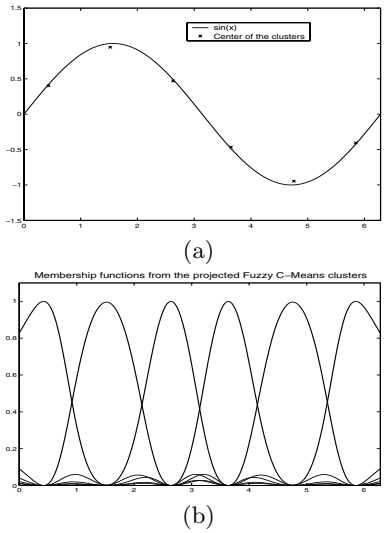


Figure 2.9. (a) Center of the clusters found by the Fuzzy C-means algorithm. Original function (-) Center of the clusters (*) (b) Membership functions projected from the partition matrix U

The projected membership functions obtained from the partition matrix U are approximated by convex membership functions as they are shown in Figures 2.11 and 2.12.

The rule base was constructed and the models were further optimized using gradient descent for 400 steps. Figures 2.13 and 2.14 show the approximation of the function. It is important to comment that for the Mamdani models there are little differences with the models shown in previous examples, but it is not the case of the Takagi–Sugeno models. Observe the orientation of the consequences of the rules, which are almost tangent to the function.

Table 2.3 summarizes the results obtained with the three methods shown. Perhaps the most remarkable results are the improvement of the models using Gaussian functions. The reason for such a benefit from the clustering can be

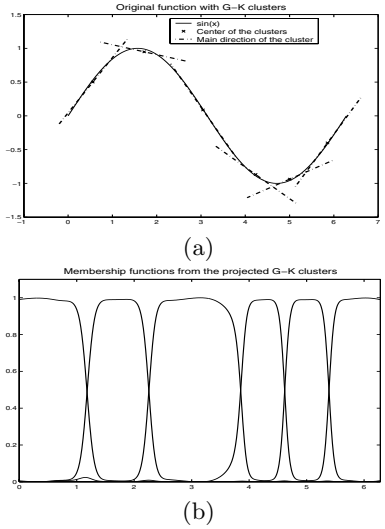


Figure 2.10. (a) Clusters found by the G-K algorithm. Original function (-) Center of the clusters (*) Main direction of the covariance matrix (.-) (b) Membership functions projected from the partition matrix U

explained by the strong similarity between the projected partition function from the clusters and the Gaussian membership functions. Observe that these results are simple illustrations of the methods and do not represent an absolute benchmark. For other functions the performance exhibit by the models will be different.

Table 2.3. Approximation error $(\sum_{i=1}^{629}[f(x) - \hat{f}(x)]^2)$of the Different Models Trained with the *Mosaic or Table Lookup* Scheme, the *Gradient Descent* Method for 400 Steps and *Clustering Gradient Descent* Method for 400 steps)

Model	Table lookup app. error	Gradient descent app. error	Clustering + GD app. error
Mamdani triangular M.F.	1.3392	0.0615	0.0858
Mamdani Gaussian M.F.	0.2378	0.0449	0.0037
Mamdani polynomial M.F.	12.9666	0.1755	0.1959
Takagi–Sugeno model.	0.0206	0.0117	0.5058

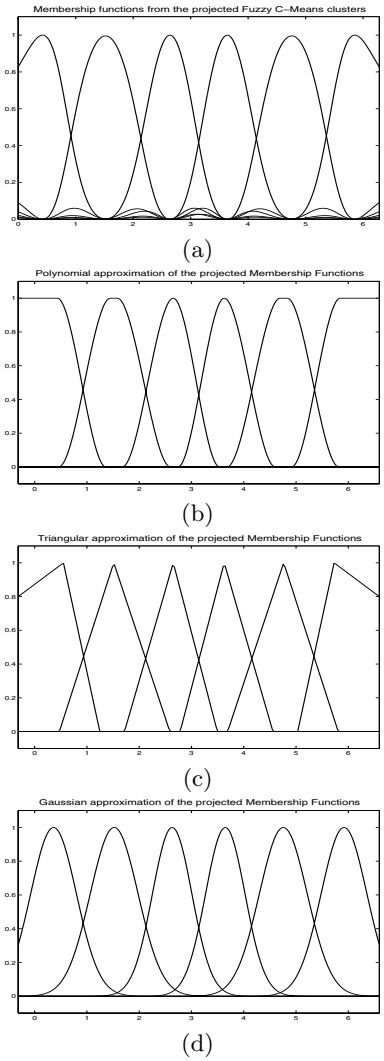


Figure 2.11. Membership functions for Mamdani models.(a) Membership functions projected from the partition matrix U (b) Approximation with polynomial M.Fs. (c)Approximation with triangular M.Fs. (d) Approximation with Gaussian M.Fs.

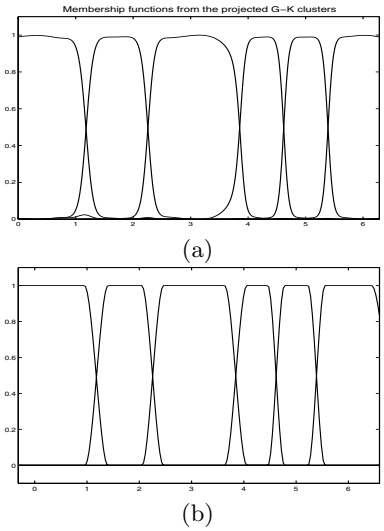


Figure 2.12. Membership functions for Takagi–Sugeno models.(a) Membership functions projected from the partition matrix U from the G–K clustering (b) Approximation with polynomial M.Fs.

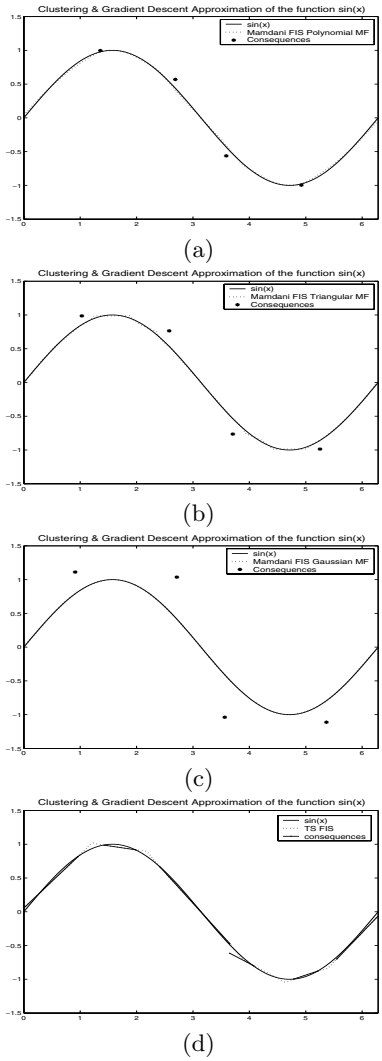


Figure 2.13. Function approximation of the models obtained using clustering and gradient. Original function (-) Approximated function (–) Consequences (*) Consequence of the TS model (-.). (a) Membership functions projected from the partition matrix U (b) Approximation with polynomial M.Fs. (c)Approximation with triangular M.Fs. (d) Approximation with Gaussian M.Fs.

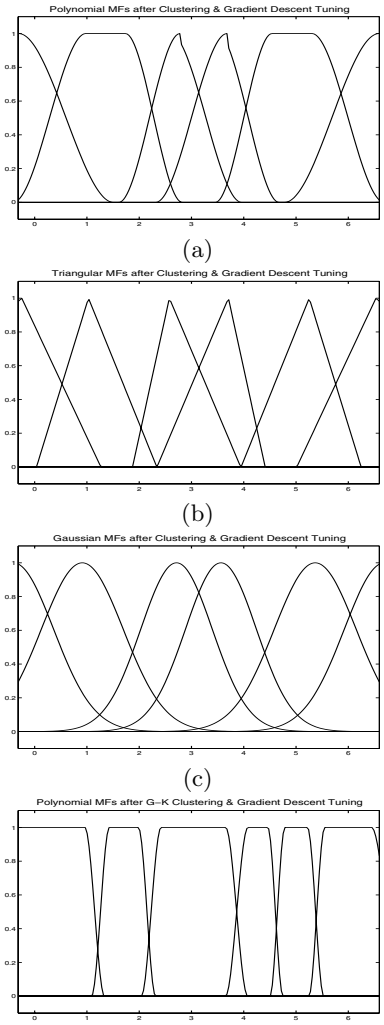


Figure 2.14. Membership functions for the models obtained by clustering and gradient descent optimization.(a) Mamdani model with polynomial M.Fs. (b) Mamdani model triangular M.Fs. (c) Mamdani model with Gaussian M.Fs.(d) Takagi–Sugeno model with polynomial M.Fs.

2.4 Using Evolutionary Strategies

The evolutionary strategies are computational algorithms that use methods derived from the concept of "natural evolution." Some of the methods include reproduction, mutation and selection. The use of these algorithms has been oriented to the search of parameters such that a certain computational entity can achieve some goals.

In this case the computational entity will be a fuzzy system, the goal will be to approximate a function with certain accuracy and a limited complexity and the parameters could be the number of membership functions, their distribution, etc.

Basic methods in these strategies are the so-called genetic algorithms [23]. In genetic algorithms, the data are represented as binary strings. The parameters are encoded on these binary strings. It is important to remark that the efficiency of these techniques is strongly affected by the "code book" used to construct the strings [24]. Initiallty, a group of these strings is generated as the initial "population." The fulfillment of the goal is tested for each element of the population (cost evaluation) and a "fitness" value is generated such that, if the value is larger, the objective is better achieved. The procedure can be outlined as follows:

- Take the initial population N and evaluate the "fitness" of the individuals (binary strings).
- Reproduce the population according to the "fitness," such that those individuals with higher values of fitness will have a higher probability of being reproduced.
- Make random couples among the individuals of the reproduced population and apply the "crossover" operation. The crossover operation takes two individuals and generates a random number $l \leq L$ where L is the length of the string. This operation generates two new individuals by taking the first l elements of one string and the remaining $L - l$ element from the other string. For example, take the first string $A_1A_2A_3A_4A_5A_6$ and the second string $B_1B_2B_3B_4B_5B_6$. In this case $L = 6$. Suppose $l = 2$. The crossover will be represented as

$$\begin{array}{c} A_1A_2A_3A_4A_5A_6 \\ B_1B_2B_3B_4B_5B_6 \\ --------- \\ A_1A_2B_3B_4B_5B_6 \\ B_1B_2A_3A_4A_5A_6 \end{array}$$

- Finally, some members of the population are selected for "mutation." A random number l is generated such that $0 < l \leq L$ for each selected member and the bit l is flipped. Suppose the string $A_1A_2A_3A_4A_5A_6 = 101100$ is selected for mutation and $l = 4$. The string after mutation is $A_1A_2A_3A_4A_5A_6 = 101000$.

- The "fitness" of the generated population is evaluated and the procedures of reproduction, crossover and mutation are repeated for a given number of times (generations).

These algorithms are very powerful for the search of "global" solutions in the search space, and there is a probability equal to 1 that the algorithm will find the "global solution" after a number of generations given by [20]

$$\frac{1}{1-(1-p_M\frac{N_{opt}}{2^L})^N} \tag{2.48}$$

where p_M is the probability of mutation, N_{opt} is the number of global solutions in the final population, L is the length of the strings and N is the number of strings in the population.

The application of these algorithms to the design of fuzzy systems is mainly oriented to the generation of the number and distribution of the membership functions. One example of codification is: Assume a number of triangular, trapezoidal or polynomial membership functions with overlap $\frac{1}{2}$ have been fixed for each input. Then the string will represent the distance from the previous point, as shown in Figure- 2.15. The length of the string is $L = 28$, four groups of seven bits. An example of mutation is shown in Figure- 2.16, where

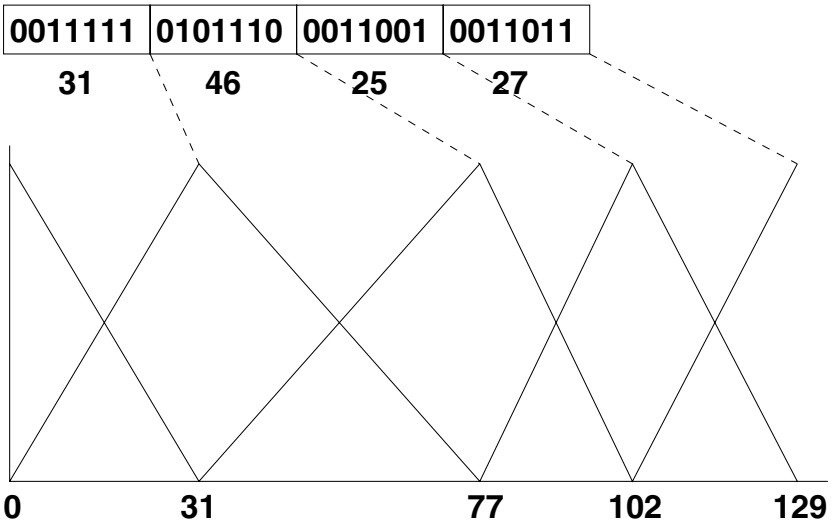


Figure 2.15. Codification of the membership functions

$l = 10$. Finally, an example of the effect of the crossover operation is shown in Figure- 2.17, where $l = 10$. Other codification methods and details can be seen in [20].

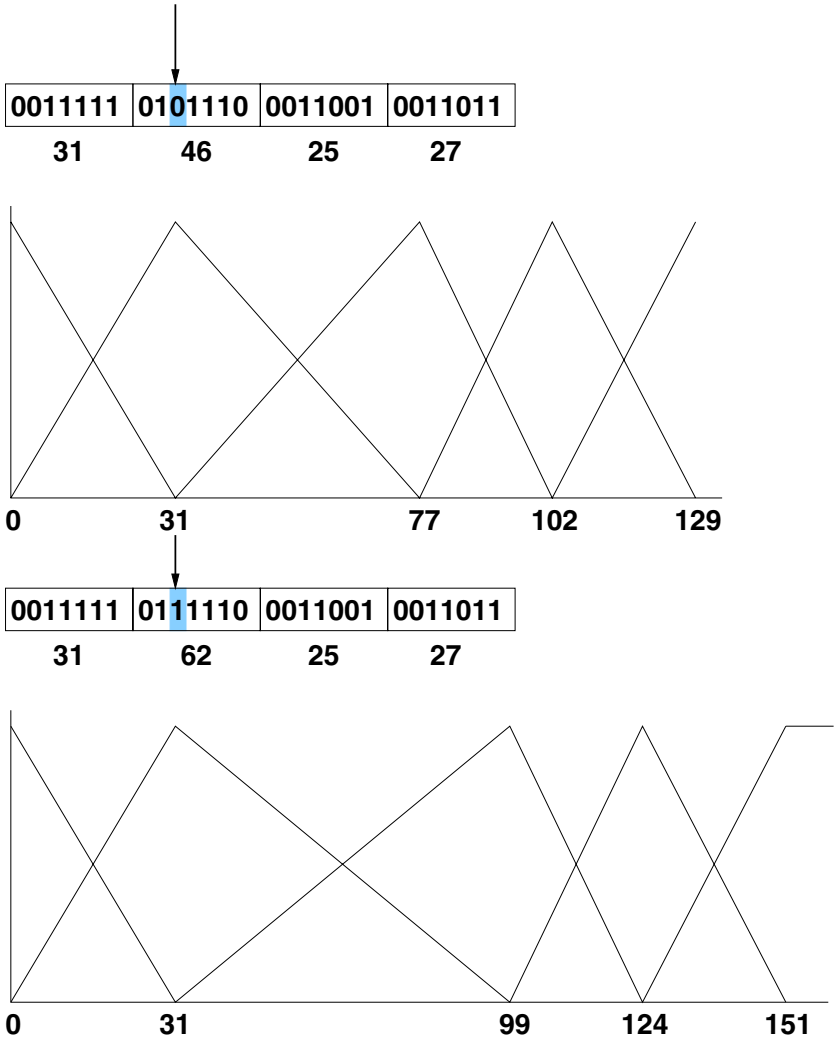


Figure 2.16. Mutation operation in a fuzzy partitions

Summary:
The *evolutionary strategies* are mainly based on discrete optimization algorithms such as the genetic algorithms. The method can calculate parameters such as number and location of the membership functions.

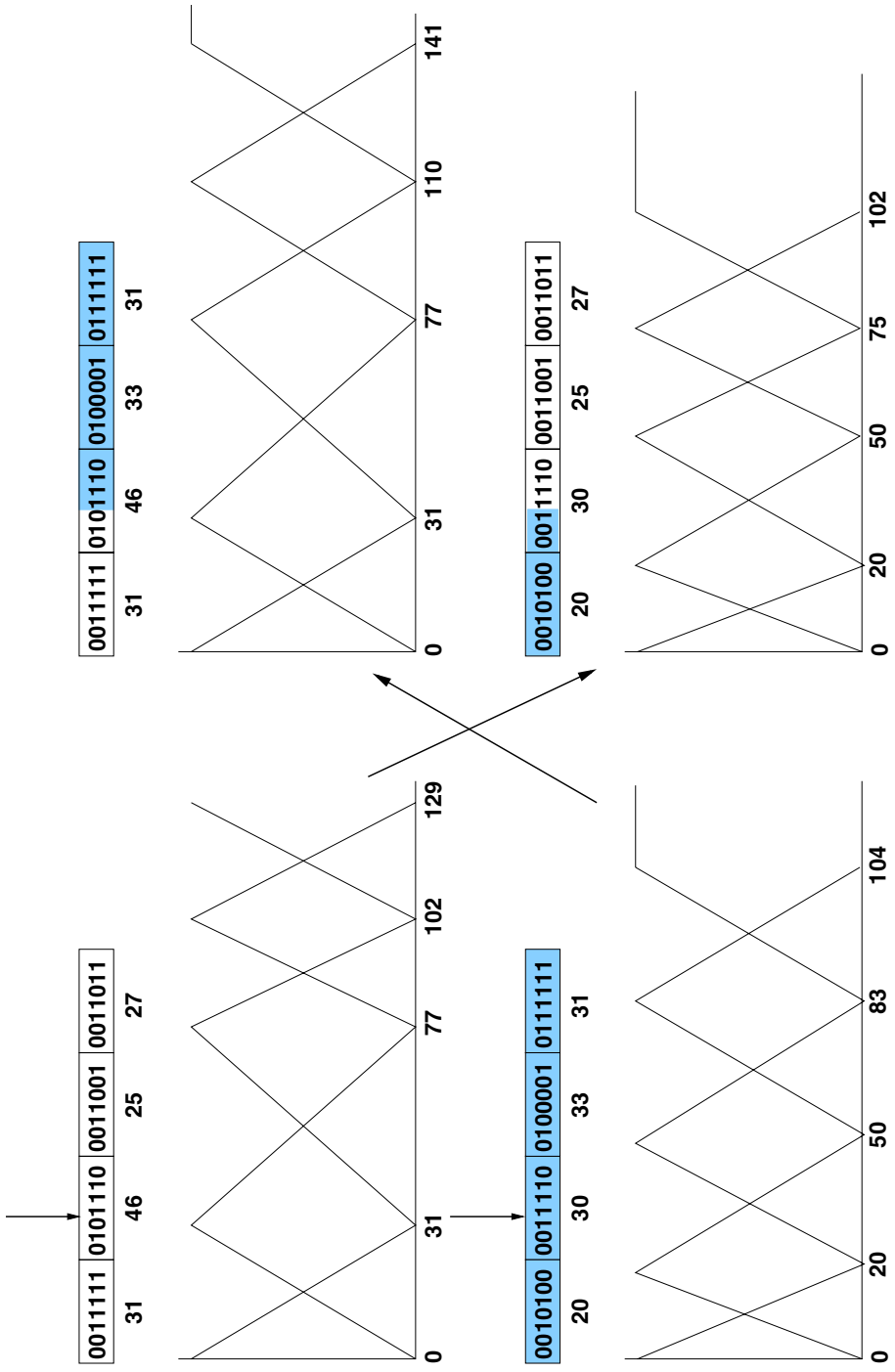


Figure 2.17. Crossover operation between two fuzzy partitions

2.5 Generalization and Consequences Estimation

The issue of generalization is quite related with the issue of consequence estimation in fuzzy systems, as will be explained in the following lines. Generalization is the capacity of the designed system (the fuzzy inference system) to generate "good" output values when new inputs are presented to the system. Two issues limit the capacity of generalization:

- Lack of excitation during model construction
- Too many degrees of freedom in the model

These two issues are strongly related because as the degrees of freedom grow the data must excite all the new modes introduced by the new degrees of freedom. There are two ways to improve the generalization:

- Reducing the degrees of freedom with the drawback of reducing the accuracy of the approximation.
- Generating many data for all possible operating modes. For some practical cases, this is almost an impossible task.

Assuming the input–output data are given in advance, the challenge is to design a system with good approximation properties and good generalization. The application of the methods reviewed in previous sections postulates the generation of the consequences of the rules by means of least squares. As mentioned in Section 2.1, the calculation of the least squares using Equation (2.11) will be possible only if $\text{rank}(W^T W) = \dim \theta$. In cases where $\text{rank}(W^T W) \leq \dim \theta$, the estimation will be very poor and the consequences of those unexcited rules will be very far from their real value. A reasonable solution is to initialize the rules using information given by a simpler model (with very few degrees of freedom) and to improve the estimation of the consequences of those rules that have been excited using recursive least squares. The advantage of the recursive least-squares algorithm is that it only updates those terms that have been excited. The procedure can be detailed in two steps.

2.5.1 Consequence Initialization

The initialization of the consequences can be done in two ways using the information given by a simple model with sufficient excitation or using expert knowledge. The use of expert knowledge demands, the designer that initialize those rules with empirical knowledge. The initialization using a simple model with sufficient excitation operates as follows:

- The smallest fuzzy model $\hat{f}(x^t)$ is constructed by placing only two membership functions (triangular or polynomial) on each input with their modal values placed, respectively, in the maximum and the minimum values of the universe of discourse and fixing the overlap value in $\frac{1}{2}$. This distribution

of the membership functions will generate a fuzzy system with 2^N rules where N is the number of inputs. This fuzzy system has the property that any input presented will excite the whole set of rules. This property guarantees enough excitation such that the 2^N consequences can be estimated using the least-squares solution given in the Equation (2.11).

- If the constructed model uses triangular or polynomial membership functions with overlap $\frac{1}{2}$, the consequences of the rules can be initialized using the model $\hat{f}(x^t)$ as follows:

$$\bar{y}^{j_1 j_2 \cdots j_N} = \hat{f}(M^{j_1 j_2 \cdots j_N}) \tag{2.49}$$

with

$$M^{j_1 j_2 \cdots j_N} = \{m^1_{j_1}, m^2_{j_2}, \ldots, m^N_{j_N}\}^T$$

where $m^i_{j_i}$ are the modal values of the membership functions of the fuzzy model $f(x^t)$. If the model is not constructed as described above, the initial consequences can be estimated by using a data set generated from the model $\hat{f}(x^t)$, so that the condition of sufficient excitation is guaranteed. This can be done just by generating input data regularly distributed and with "enough" density over the input space U.

This initialization method guarantees that the constructed fuzzy model will be at least as good as the best multilinear model, if the smaller model is constructed with triangular membership functions, or at least as good as the best third-order multipolynomial model. These bounds guarantee the quality of the generalization even if the training data have no information about some of the regions described in the rule base.

2.5.2 Consequence Estimation

Once the consequences have been initialized, the recursive least-squares algorithm can be applied to improve the estimation. The algorithm is described as follows using the notation presented in Section 2.1:

$$\theta(k+1) = \theta(k) + \gamma(k)[y^t - W^t\theta(k)] \tag{2.50}$$

with $W^t = \{w^t_1, w^t_2, \ldots, w^L_t\}$, $\theta(k) = \{\bar{y}^1(k), \bar{y}^2(k), \ldots, \bar{y}^L(k)\}$ and:

$$\gamma(k) = P(k+1)W_{k+1} \tag{2.51}$$

$$= \frac{1}{W^t P(k) W^{t^T} + 1} P(k) W^t \tag{2.52}$$

$$P(k+1) = [I - \gamma(k)W^t]P(k) \tag{2.53}$$

with the initial value $P(0) = \alpha I$, where α is a large scalar value. The procedure is repeated and each time the index k is incremented. Also, the index t is incremented until it reaches the value N, and then the value of t is reset to

$t = 1$. The initial values of $\theta(0)$ are the initialization values given by the procedure described before. The following example is presented in order to illustrate how the present method improves the generalization.

Example 2.1. The objective is to approximate the function of two variables $f(x, y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$ on the interval $(x, y) \in U$ $U = [-2, 2] \times [-2, 2]$. The function is plotted in Figure- 2.18.

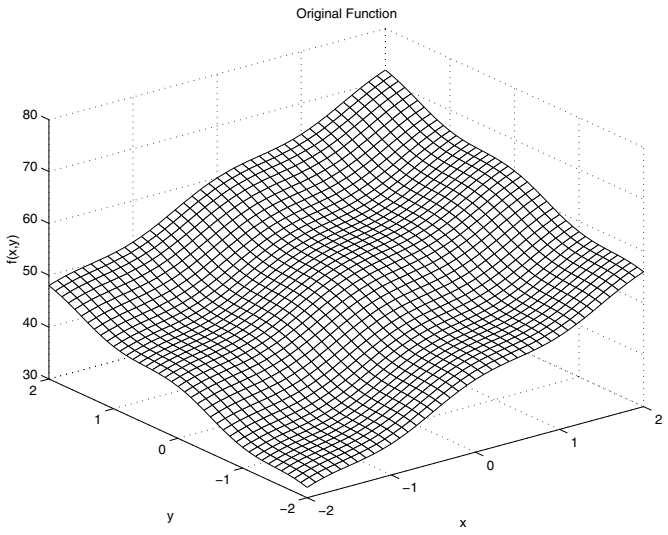


Figure 2.18. Function to be approximated $f(x, y) = 6x+4y+\cos(\pi x)+\cos(\pi y)+50$

The function is sampled at 153 points. The sampling was done such that only one point falls in the interval $V = [-2, 0] \times [-2, 0]$. The data points are depicted in Figure 2.19.

The function will be approximated by a fuzzy system using five triangular membership functions equally distributed on each domain with overlap 0.5. A total of 25 rules is generated and the consequences will be estimated using three methods: least squares (LS), recursive least squares (RLS) and RLS with the consequence initialization method explained in Section 2.5.1. Observe in Figure 2.20 that the LS method and the "pure" RLS fail to approximate the function in the domain V and even the LS solution fails to make a good approximation in the region where the "training" points were selected.

The third method as was explained in Section 2.5.1 first calculates the smallest fuzzy model $\hat{f}(x, y)$ with only two membership functions with overlap 0.5 covering the whole domain on each input. The model has four rules that are excited by all the points such that the consequence estimation does not represent any numerical problem. The approximation generated by this model

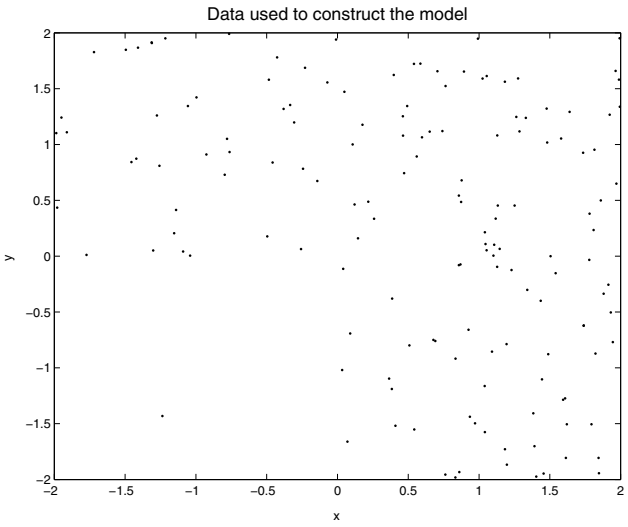
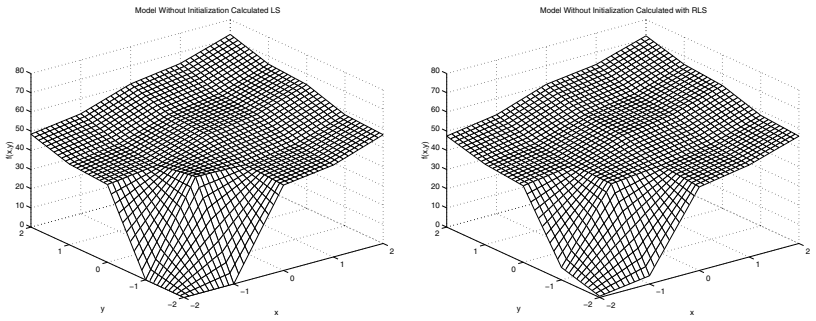


Figure 2.19. Sampled points to approximate the function $f(x, y) = 6x + 4y + \cos(\pi x) + \cos(\pi y) + 50$



(a) (b)

Figure 2.20. (a) Approximation obtained with the consequences calculated using LS (b)Approximation obtained with the consequences calculated using RLS

is shown in Figure 2.21. Equation (2.49) is used to initialize the consequences of the rules in the model with 25 rules such that it generates an approximation perfectly equivalent to the approximation given by the model $\hat{f}(x, y)$.

Then the consequences are estimated using RLS. The results are shown in Figure 2.22. Observe that the approximation is good in the whole domain U and there are no big changes in the region V where almost no data exist during the training. A final comparison was performed by generating 141 points in the domain U but excluding the region V (the same conditions used for the training) to observe the approximation error in the "well-excited" region. The results are presented in Table 2.4, and the error index is calculated as $E =$

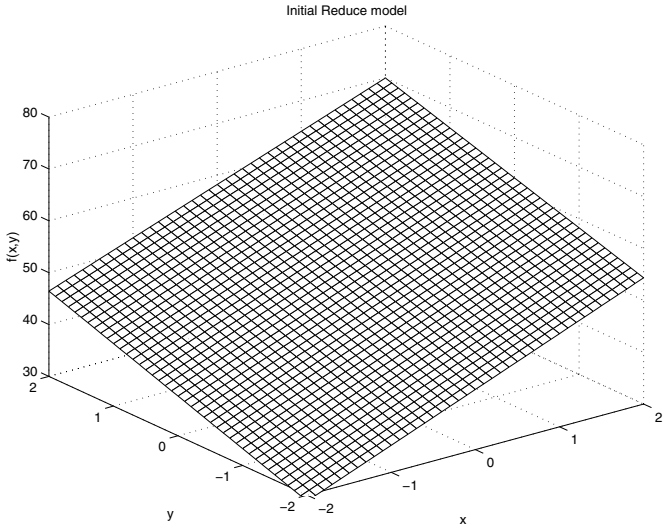


Figure 2.21. Approximation generated by the "smallest" fuzzy model $\hat{f}(x, y)$ with only 4 rules

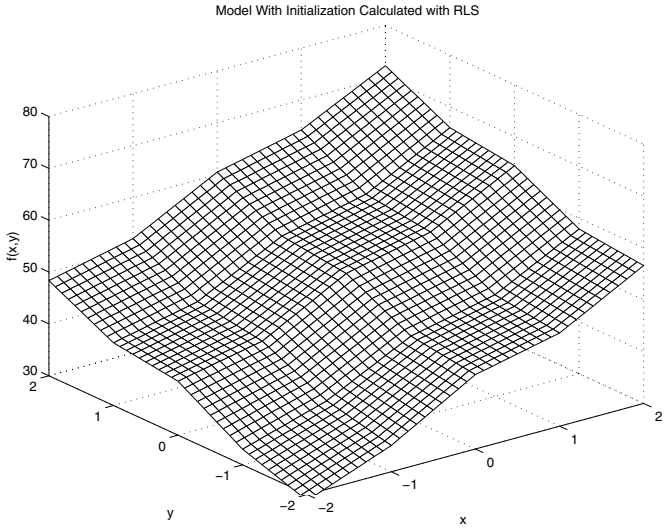


Figure 2.22. Approximation generated by the fuzzy model initialized with $\hat{f}(x, y)$ and with the consequences calculated using RLS

$\frac{1}{141}\sum_{i=1}^{141} e^2$, where E is the difference between the real and the estimated values. From these results, it is clear that the LS method is the worst. The reason is that the inputs selected for the estimation did not excite some of the rules and in this specific case the rank$(W^T W) = 22$; therefore, the LS

solution is badly conditioned. The RLS solution is better because it updates only the excited rules but the badly excited rules are not updated, making a bad generalization on "poorly" excited regions. Finally, the best performance is by far the one of the proposed method. The reason is that this method assumes the generalization given by a "well"-excited model ($\hat{f}(x, y)$) and the tuning will only improve the approximation on these regions where there is enough excitation.

Table 2.4. Example: Comparison Between Methods for Consequences Calculation

Method	Approx. error
Least Squares	0.3455
Recursive Least Squares	0.1659
RLS with initialization using $\hat{f}(x, y)$	0.0146

Summary:
Fuzzy models should make good predictions even when they are asked to predict on regions that were not excited during the construction of the model. The generalization capabilities can be controlled by an appropriate initialization of the consequences (prior knowledge) and the use of the recursive least squares to improve the prior choices. The prior knowledge can be obtained from the data.

2.6 Example of an Industrial Application

This section presents an industrial application of a static model. In this case the system helps to supply hot water for domestic use. The water is heated using steam coming from the cooling circuit of an electric power plant. The heat is transferred to the cold water by a heat exchanger (see Figure 2.23). Since the demand of hot water (F_{hw}) and the supply of steam change (F_{steam} and T_{steam}), the system must be commanded by a control system to guarantee a supply of hot water at a constant temperature (T_{hw}) (Set-point = 60°C). This objective is achieved by combining a feedback controller constructed with a PID (proportional integral derivative) and a feed-forward controller constructed using a fuzzy model (see Figure 2.24). The fuzzy model is constructed using experimental data supplied by the manufacturer of the heat exchanger. The fuzzy model is constructed to map the flow of water (F_{hw}), the temperature of the steam (T_{steam}) and the temperature of the cold water (T_{cw}) into a steam flow (F_{steam}) to guarantee that the hot water is supplied at the correct temperature (60°C).

$$F_{steam} = f(F_{hw}, T_{steam}, T_{cw})$$

Since the function is constructed using nominal data and the controller is not supposed to be "fine-tuned" on each installation, the feed-forward action will be insufficient to guarantee the supply of the water at the correct temperature. For this reason an additional feedback controller is put in place.

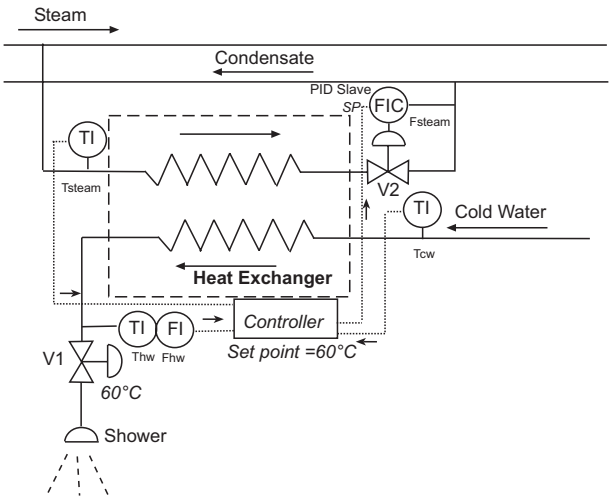


Figure 2.23. Diagram of the installation of the heat exchanger including the instrumentation and the control system

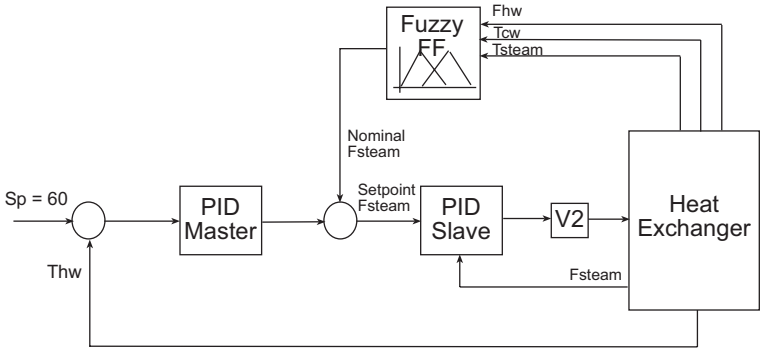


Figure 2.24. Diagram of the control system for the heat exchanger

The data supplied by the manufacturer of the heat exchanger are shown in Figure 2.25 together with the result of the approximation [see Figure 2.25(d)].

The system was implemented using triangular membership functions since the memory and the computational time available in the microcontroller were limited. Figure 2.26 shows the membership functions of the feed-forward controller.

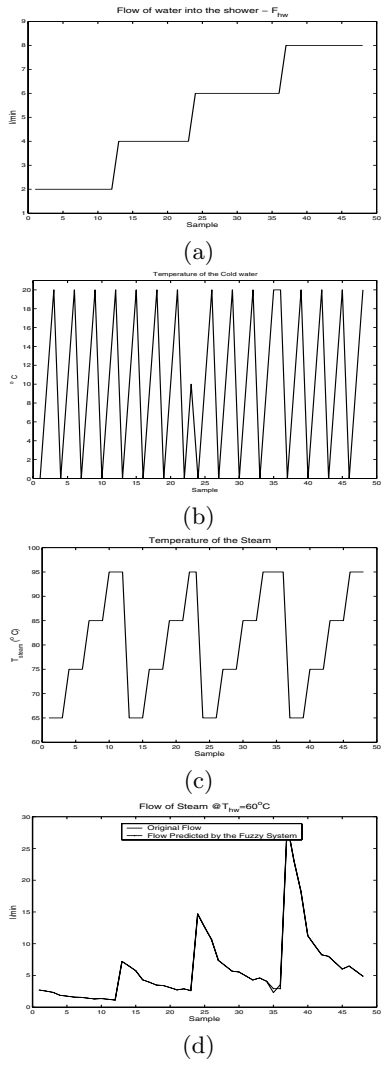


Figure 2.25. Signal collected from the heat exchanger to guarantee a nominal temperature of 60°C (a) Flow of hot water F_{hw} (b) Temperature of the cold water T_{cw} (c) Temperature of the steam T_{steam} (d) Flow of steam F_{steam} (-) Original value (.-) Value generated by the fuzzy system

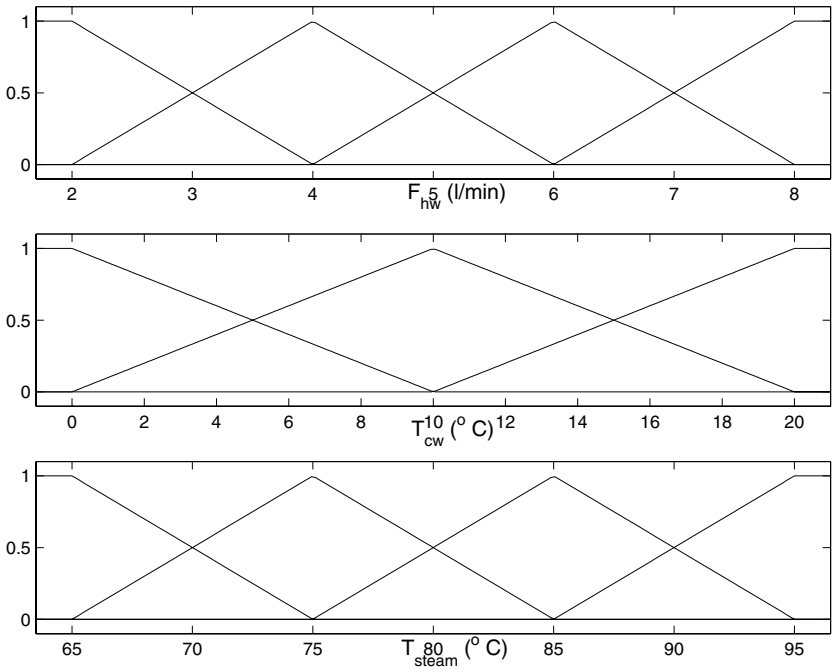


Figure 2.26. Membership functions of the feed-forward controller

2.7 Conclusions

This chapter has presented different methods to construct fuzzy models that approximate nonlinear functions. The issue of lack of excitation and generalization has been analyzed and a method to guarantee good generalization has been proposed. This method guarantees a lower bound in the quality of the model (the fuzzy model will be at least as good as the best multilinear approximation). The example of the industrial application shows a method to construct feed-forward controllers using fuzzy inference systems for function approximation.

http://www.springer.com/978-1-85233-828-2

Fuzzy Logic, Identification and Predictive Control
Espinosa Oviedo, J.J.; Vandewalle, J.P.L.; Wertz, V.
2005, XX, 264 p., Hardcover
ISBN: 978-1-85233-828-2