Chapter 7

FUZZY LOGIC

Javid Taheri and Albert Y. Zomaya
The University of Sydney

The principles of Fuzzy Logic were introduced several decades ago by Lotfi Zadeh [1]. The thrust of Zadeh's work was in the realization that decision making in the real world is not crisp. Most of the time, decisions are not "binary" in nature, such as yes/no, black/white, up/down, etc. Events and decisions tend to be "fuzzy," and a good example is the case of a glass of water that can be described as full or empty. Now, if one is to take a sip of water, then the glass is neither empty nor full, but in between. If the process continues until the glass is empty, then one can say that the glass has undergone different states from the time it was full to the time it became empty. It is obvious that the above phenomenon cannot be described by using binary logic and different rules need to be adopted to account for the different levels of "fuzziness" that any a decision process can take.

1 FUZZY PRINCIPLES

1.1 Multivalue Algebra

The most important difference between fuzzy and binary representations is the way a variable is quantized. The binary world uses two values (0 or 1) to represent each phenomenon, while in the fuzzy world variables are quantized by a function that takes a smooth shape ranging from 0 to 1 [1, 2].

1.2 Simplicity versus Accuracy

Fuzzy logic attempts to formulate an environment not accurately but in a simple manner. In modern sciences, especially mathematics and physics, there is an accurate formulation for every event. On the other hand, if an event cannot be explained accurately, a decision can be made with a given probability. Fuzzy logic tends to simplify the process of making a decision, especially in cases where an exact formula is very difficult to derive or does not exist.

1.3 Probability versus Possibility

To explain the interplay between probability and possibility, let's return to our earlier example, the glass of water. If one is to say that this is "a glass containing water with the probability of 0.5," it means that the whole glass might contain water or some other liquid like gasoline. On the other hand, if one uses the expression that this is "a glass containing water with the possibility of 0.5," it means that the liquid is definitely a mixture of water and another unknown liquid. Another distinguishing factor between these two expressions is the sample spaces they represent. In probability, the sum of all events that could happen should add up to 1.0, while in the case of possibility, the sum can be smaller or larger than 1.0.

1.4 Fuzzy Sets

A fuzzy set is a fundamental component of a fuzzy system [2]. Traditionally, a *set* is a collection of elements or objects that can be of finite or infinite size. In this case, a given element, x, can be a member of set A, or otherwise. So the answer to the question "Does x belong to set A?" is either true or false. In contrast, each fuzzy set is a set of ordered pairs and is usually defined as follows:

$$\tilde{A} = \left\{ \left(x, \mu_{\tilde{A}}(x) \right) \right\}$$

where $\mu_{\tilde{A}}(x)$ is the *membership function* and represents the degree of truth or compatibility of variable x with the set. Figure 7.1 shows a simple fuzzy set with following definition:

$$\tilde{A} = \left\{ \left(\left(x, \mu_{\tilde{A}}(x) \right), \mu_{\tilde{A}}(x) = \left(1 + (x-5)^2 \right)^{-2} \right) \right\}$$

1.5 Fuzzy Numbers

A fuzzy number $\tilde{M}$ is called positive (negative) if its membership function is such that [2]

$$\mu_{\tilde{M}}(x) = 0, \forall\, x < 0 \; (x > 0)$$

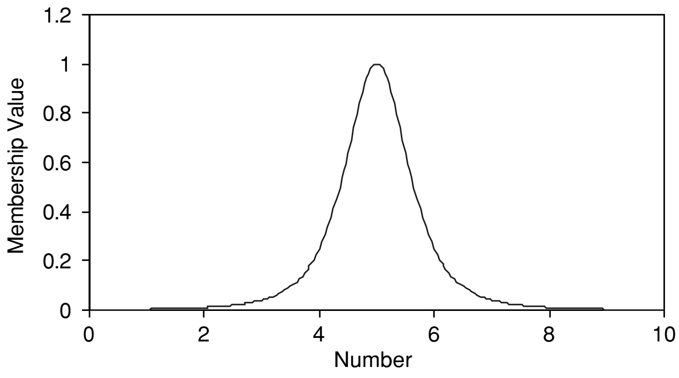


Figure 7.1. Representation of the fuzzy numeral "approximately 5"

1.6 Basic Set-Theoretic Operations

Different logic operations are defined for fuzzy sets and numbers. The basic logic operations of union, intersection, and complement are usually defined as follows [2].

1.6.1 Union

The union of two fuzzy sets $\tilde{A}$ and $\tilde{B}$ is

$$\tilde{C} = \tilde{A} \cup \tilde{B} = \left\{ \left(x, \mu_{\tilde{C}}(x)\right), \mu_{\tilde{C}}(x) = \max\left(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right) \right\}$$

1.6.2 Intersection

The intersection of two fuzzy sets $\tilde{A}$ and $\tilde{B}$ is

$$\tilde{D} = \tilde{A} \cap \tilde{B} = \left\{ \left(x, \mu_{\tilde{D}}(x)\right), \mu_{\tilde{D}}(x) = \min\left(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\right) \right\}$$

1.6.3 Complement

The complement of a fuzzy set $\tilde{A}$ is

$$\overline{\tilde{A}} = \left\{ \left(x, \mu_{\overline{\tilde{A}}}(x)\right), \mu_{\overline{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x) \right\}$$

Figure 7.2 shows the results of the above operations on fuzzy sets $\tilde{A}$ and $\tilde{B}$. These definition are simple and don't obey advance set-theoretic operations such as monotonicity, commutativity, and associativity. To overcome this problem, several complex definitions have been proposed in the literature [2].

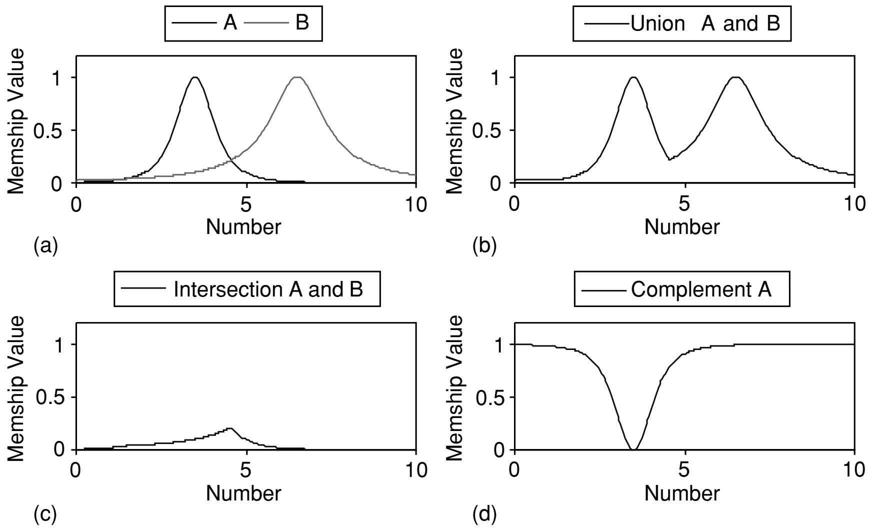


Figure 7.2. Fuzzy operations. (a) Representation of two fuzzy numbers A and B; (b) union of A and B; (c) intersection of A and B; (d) complement of A

2 FUZZY SYSTEMS

Figure 7.3 shows a generic fuzzy system. In all fuzzy systems, there are three main components: Rule Database, Fuzzification, and Defuzzification.

2.1 Fuzzy rules

Fuzzy systems are based on the preliminary information given to the system as fuzzy rules. These rules, which are written as linguistic commands, are usually not so precise. In fact, they are written to enable decision to be made in cases where there is imprecise or no preliminary information about the system under considerations. The following rules represent instances of generic fuzzy rules:

- IF "Salary is High" then "Tax is High"
- IF "Speed is Low" then "Accident Probability is Low"
- IF "Left Obstacle is Near" and "Front Obstacle is Near" then "Turn Right Quickly" and "Reduce Speed"

The above rules may have single or multiple antecedents and/or consequences.

2.2 Fuzzification

One of the most important components of every fuzzy system is the fuzzification phase, during which the crisp values from a real-world system are managed so that they can be processed by the fuzzy system [2]. Fuzzy rules, as seen earlier, are linguistic expressions that need to be further clarified, as in the case of the following rule:

IF "Salary is High" then "Tax is High"

So what does "High" mean? How high does the salary need to be so that it is considered "High"? Also, what is "High" in the context of how much tax needs to be paid? The process of defining this kind of information for a fuzzy system is known as *fuzzification*. To achieve this, knowledge-based information is

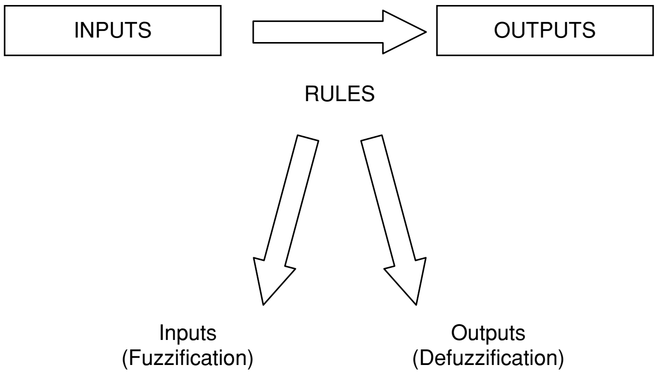


Figure 7.3. A generic fuzzy system

categorized into several parts as membership functions or fuzzy sets. Then a label is assigned to each part. For example, "Salary" could be categorized as shown in Figure 7.4. Note that membership functions designed to separate the different classes of salary earnings are overlapped smoothly to reduce the sensitivity of the fuzzy system.

2.3 Defuzzification

This process attempts to generate a crisp value for each fuzzy output generated by the fuzzy system. The following methods are the most popular for the *defuzzification* process.

2.3.1 Center of Area (COA)

In this case, the crisp value is calculated as the integral of the output fuzzy number weighted by the value of the membership function, which can be defined as follows:

$$u^{COA} = \frac{\int_U u.\mu^C(u)\,du}{\int_U \mu^C(u)\,du}$$

where $\mu^C(u)$ is the membership function of the fuzzy value.

2.3.2 Center of Sum (COS)

This defuzzification method is a simplified version of COA and is defined as follows:

$$u^{COA} = \frac{\int_U u.\mu^C(u)\,du}{\int_U \mu^C(u)\,du}$$

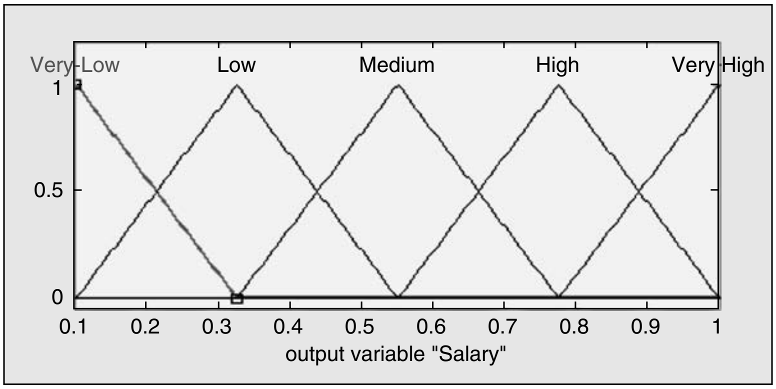


Figure 7.4. Fuzzification of the variable "Salary"

2.3.3 Mean of Maximum (MOM)

The maximum of the fuzzy number is computed, and then the average of both the maximum and the actual number is the defuzzified version.

2.3.4 Smallest of Maximum (SOM)

The maximum of the fuzzy number is computed, and then the smallest value is considered as the defuzzified number [2].

2.3.5 Largest of Maximum (LOM)

The maximum of the fuzzy number is computed, and then the largest value is considered as the defuzzified number [2]. To clarify the above definitions, Figure 7.5 shows how a fuzzy variable can be defuzzified.

2.4 Mamdani Fuzzy Systems

The Mamdani system is one of the two most famous fuzzy systems and is usually used for making fuzzy decisions [2–5]. In this system, the input and output variables are all fuzzified with several membership functions. For example, assume that a fuzzy system is designed to define the salary of an employee. Also suppose that the salary of an employee is related to his/her work experience and education level.

Figure 7.6 provides an overview of the above system. Although the output of this system is the level of salary, the first step is to fuzzify the input variables with membership functions. Towards this end, work experience (WrkExp) is fuzzified by three triangular membership function (Figure 7.7) as Beginner, Intermediate, or Expert, and the Education level (Edu) is fuzzified by three membership functions (Figure 7.8) as High School Diploma, Bachelor Degree, or Post Graduate Degree. The output of the system, Salary, is fuzzified by five labels (Figure 7.9) as Very-Low, Low, Medium, High, and Very-High. Note that, to generalize the controller, all variables are normalized to 1.0.

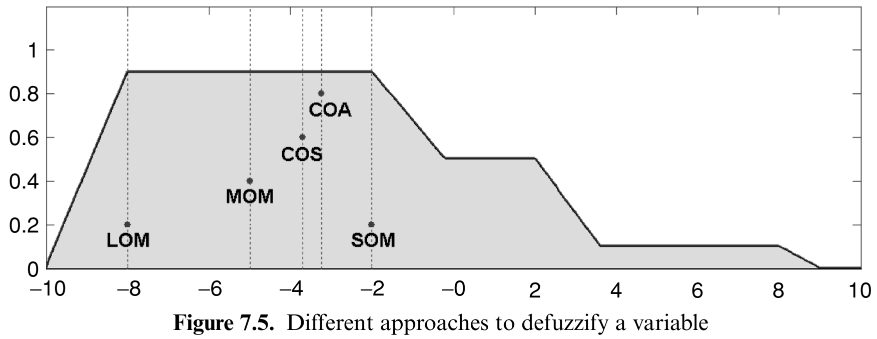
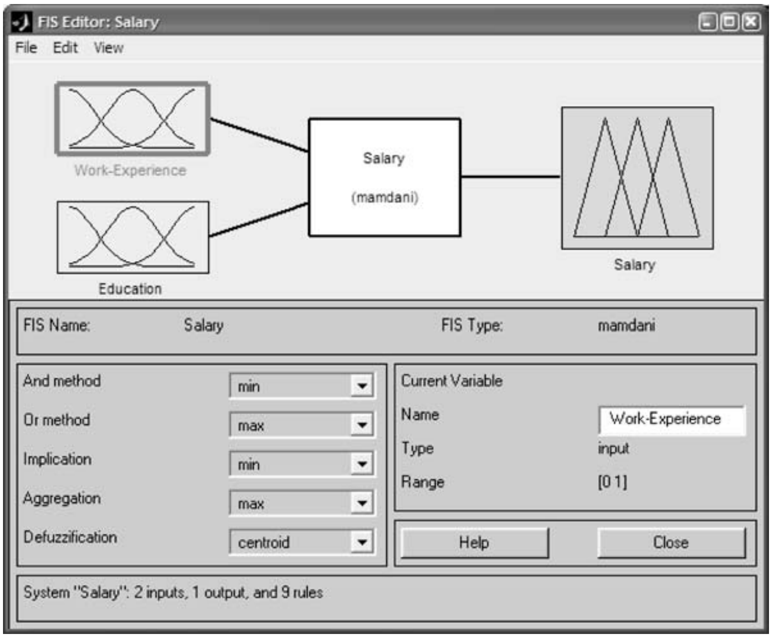


Figure 7.5. Different approaches to defuzzify a variable



FIS Name: Salary
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: Work-Experience
Type: input
Range: [0 1]
System "Salary": 2 inputs, 1 output, and 9 rules

Figure 7.6. The general overview of Mamdani's salary system

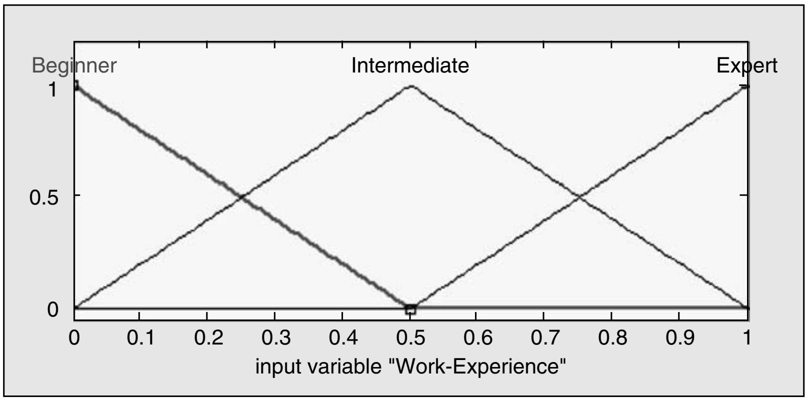


Figure 7.7. Fuzzification of the "Work-Experience" variable

Table 7.1 lists the rules of this system. To clarify how this fuzzy system computes the salary of an employee, the general data flow of this system is shown in Figure 7.10, while the general surface view of this system is shown in Figure 7.11.

Note that there are two other logic operations that need to be performed to compute the final fuzzy answer: *implication* and *aggregation*. These two operators are usually defined as AND and OR operators [2]. In this example, the COA is chosen as the defuzzification method. The Work Experience and Education Level variables are set to 0.1 and 0.3, respectively. Therefore, the Salary output for these inputs is 0.365. The general Surface View of this controller is presented in Figure 7.11.

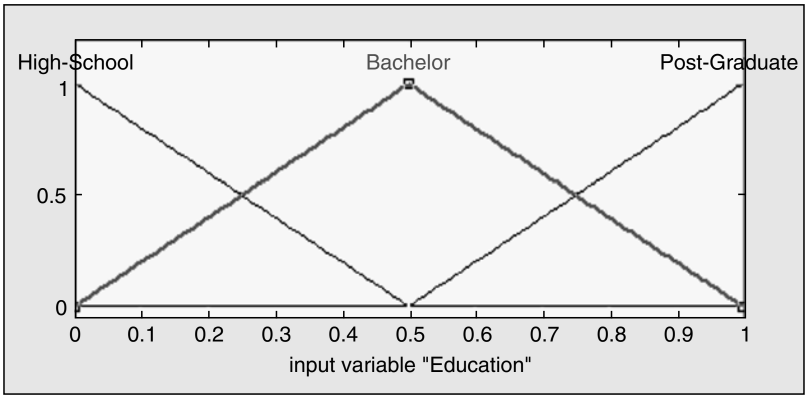


Figure 7.8. Fuzzification of the "Education" variable

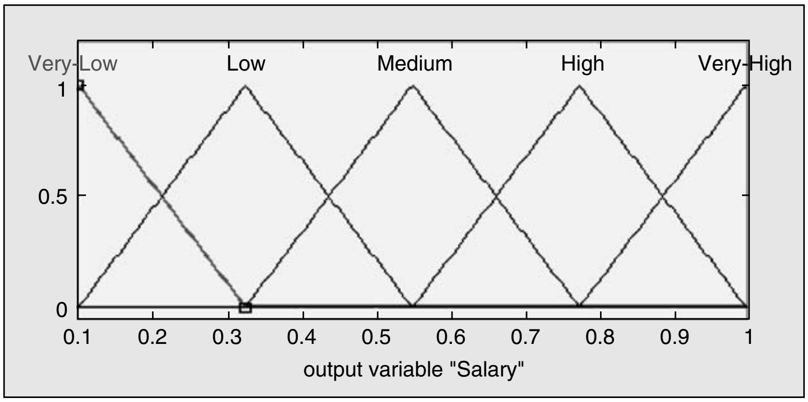


Figure 7.9. Fuzzification of the "Salary" variable

Table 7.1: Fuzzy rules for the system of Figure 7.6

Antecedent			→		Consequence
IF	WrkExp is Beginner	And	Edu is High School	Then	Salary is Very-Low
IF	WrkExp is Beginner	And	Edu is Bachelor	Then	Salary is Low
IF	WrkExp is Beginner	And	Edu is Post Graduate	Then	Salary is Medium
IF	WrkExp is Intermediate	And	Edu is High School	Then	Salary is Low
IF	WrkExp is Intermediate	And	Edu is Bachelor	Then	Salary is Medium
IF	WrkExp is Intermediate	And	Edu is Post Graduate	Then	Salary is High
IF	WrkExp is Expert	And	Edu is High School	Then	Salary is Medium
IF	WrkExp is Expert	And	Edu is Bachelor	Then	Salary is High
IF	WrkExp is Expert	And	Edu is Post Graduate	Then	Salary is Very High

2.5 Sugeno Fuzzy Systems

The Sugeno fuzzy system is another class of fuzzy systems that is usually used for control system applications [2, 6]. The output of each rule in this system is a linear, or in some cases a nonlinear, combination of its inputs. The output of the different rules is augmented to calculate the final output, which is actually the weighted sum of the rules.

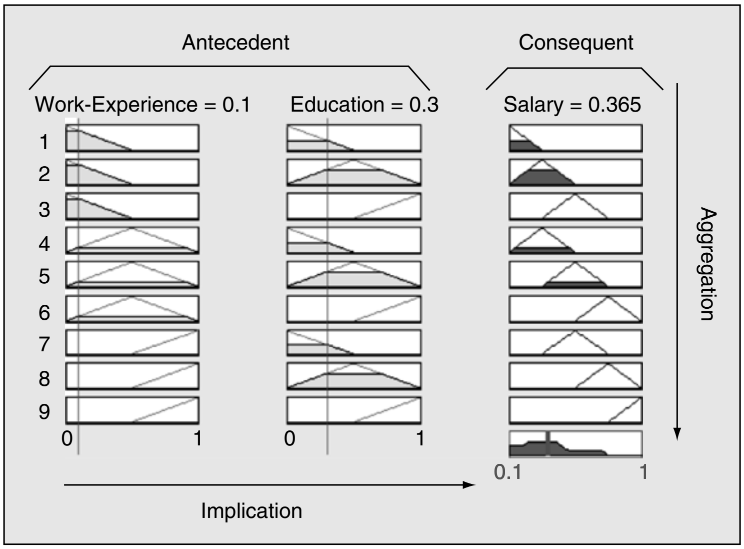


Figure 7.10. A flow diagram for Mamdani's fuzzy system

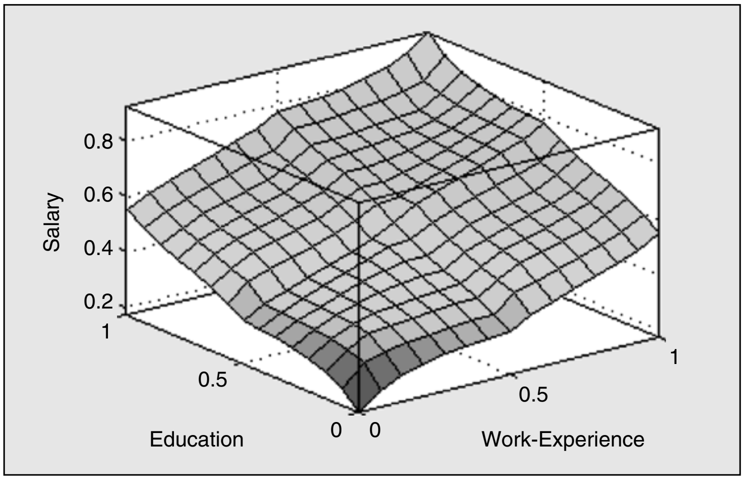
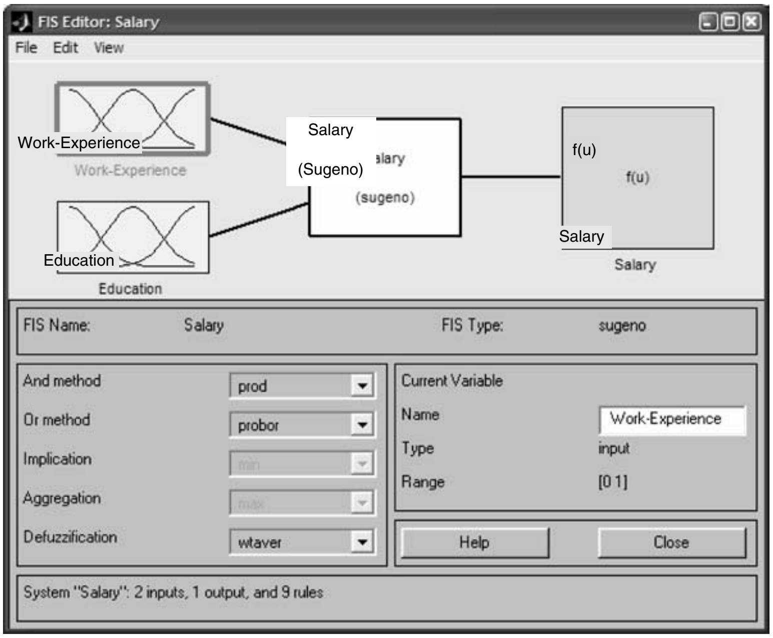


Figure 7.11. A surface view of Mamdani's salary system

To clarify the above, a Sugeno fuzzy system is designed to solve the salary problem given previously. Figure 7.12 shows the general overview of the system. The way the input variables are fuzzified is exactly the same as in Mamdani's version of this controller. The only difference is in defining the output for each fuzzy rule. In this case, five different formulas are defined to determine the salary category. To simplify the problem, these formulas are selected as constant numbers (although they can be any linear or nonlinear combination of the inputs) labeled as Very-Low, Low, Medium, High, and Very-High, with the following definitions:

Very-Low = 0.1



FIS Name: Salary
FIS Type: sugeno
And method: prod
Or method: probor
Implication: min
Aggregation: max
Defuzzification: wtaver
Current Variable
Name: Work-Experience
Type: input
Range: [0 1]
System "Salary": 2 inputs, 1 output, and 9 rules

Figure 7.12. A general overview of Sugeno's salary system

Low	=	0.25
Medium	=	0.5
High	=	0.75
Very-High	=	1.0

The rules of Table 7.1 are applicable here, with the only difference being how the output is defined. Figure 7.13 shows a general overview of the rules firing scheme when the input variables are 0.1 and 0.3 for Work Experience and Education Level, respectively. In this case, the salary output is 0.232. The general Surface View of this system is given in Figure 7.14.

2.6 Fuzzy Decision Makers

Fuzzy decision makers are another class of fuzzy systems used for real-world applications [7-9]. In these systems, a predefined number of simple rules are embedded into the system, and then the system is allowed to make its own decisions, even in the case of unknown events for which the system was never trained.

To demonstrate the general idea of such systems, assume that one knows how the system must behave in extreme conditions, as shown in Figure 7.15, which is drawn for the examples provided in the last two sections to set the amount of salary for an employee. Then the aim of the whole system is to decide for all conditions inside the plate shown in Figure 7.15, while the rules are actually written for the known conditions that are marked with spheres (the system is trained for these points as its rules).

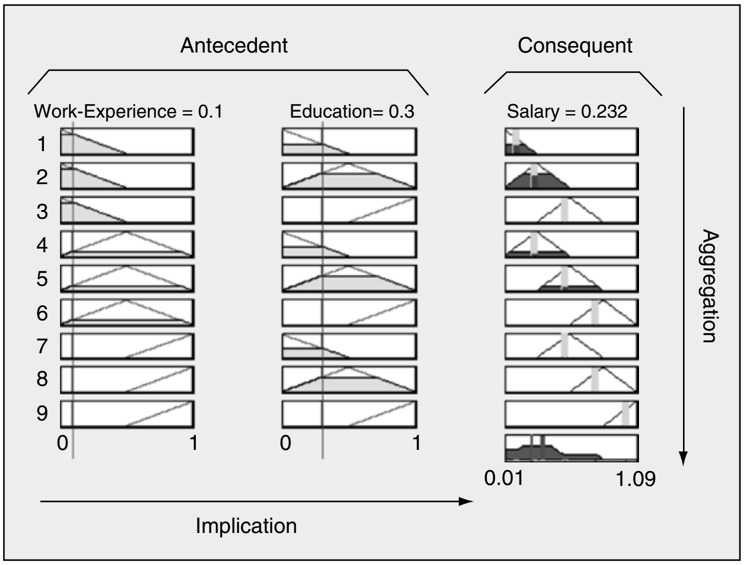


Figure 7.13. A flow diagram of Sugeno's fuzzy system

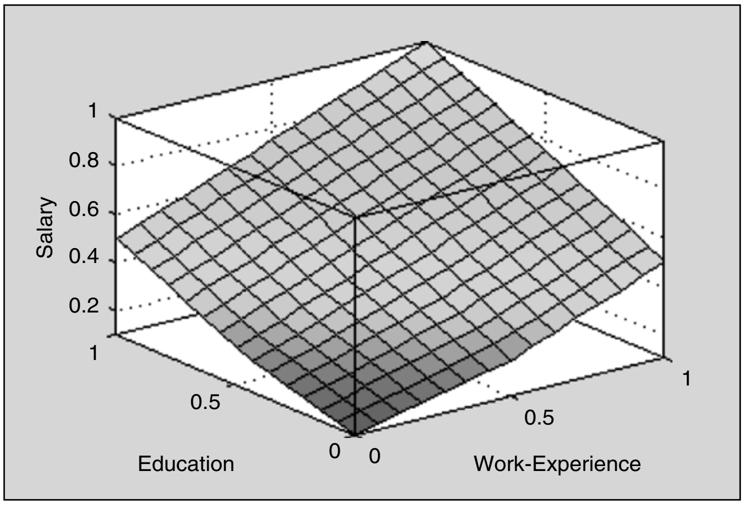


Figure 7.14. A surface view of Sugeno's salary system

2.7 Fuzzy Controller

Fuzzy Controllers are the other type of system employed for systems control [10–13]. The most famous example of this kind of system is reverse car parking. This example is one of the Demos of the Matlab® Releases, Version 13, Fuzzy Toolbox [14]. Figure 7.16 shows the initial conditions of a car to be parked, while Figure 7.17 shows the trajectory of the car position when the fuzzy controller is parking the car.

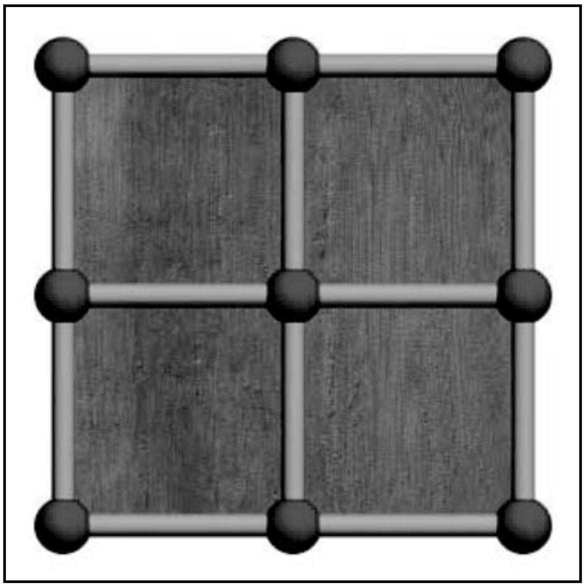


Figure 7.15. The rules are composed for the marked areas, although the system is able to make its decision for all the points

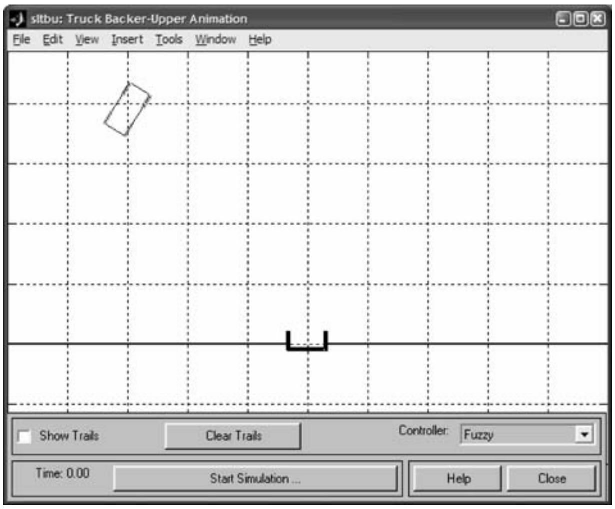


Figure 7.16. The relative positions of a car and the parking spot

2.8 Fuzzy Classifiers

Fuzzy classifiers are other classes of systems with different functionalities. [16, 17]. The aim here is to cluster objects, for example, in cases of system identification, time-series prediction, and noise cancellation. For further information, please refer to the Fuzzy Toolbox of the Matlab®, released version 13 [14].

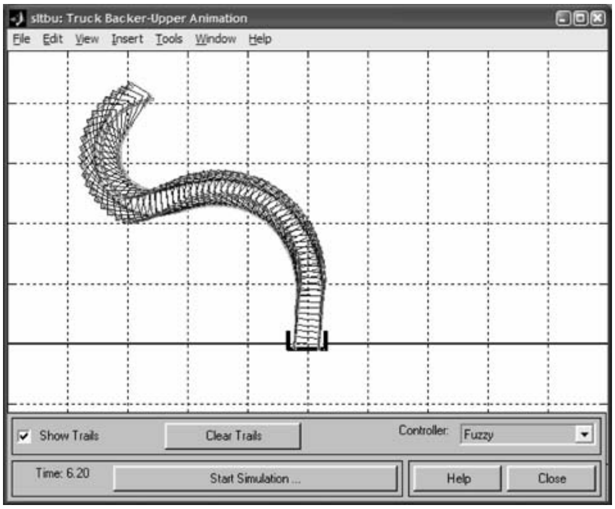


Figure 7.17. The trajectory of the car in reverse parking mode

3 DATA CLUSTERING ALGORITHMS

Clustering algorithms are used extensively not only to organize and categorize data but also to compress them in order to construct a model [17–24]. Through use of clustering techniques, data are partitioned into several groups such that the similarity within a group is larger than the similarities with other groups. These techniques are usually used in conjunction with radial basis functions or fuzzy modeling to determine the initial locations of the radial basis functions or fuzzy *IF – THEN* rules. In this case, a similarity function is usually defined to take two variables and generate a small output for similar inputs and large numbers for nonsimilar ones. It is important to note that clustering techniques used for structure identification in neural or fuzzy models are highly heuristic, and it is possible to find a data set in which none of the clustering techniques is applicable.

3.1 K-Means Clustering

The K-means algorithm partitions a group of n vectors x_j: $j = 1,...,n$ into c groups G_i:$i = 1,...,c$, and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized [19,20]. To achieve this outcome, let's assume that

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\left(\sum_{k,\, x_k \in G_i} \left\| x_k - c_i \right\|^2 \right)$$

where $J_i = \sum_{k,\, x_k \in G_i} \left\| x_k - c_i \right\|^2$ is a cost function within group i.

The partitioned groups are typically defined by a $c \times n$ binary membership matrix U, where the elements u_{ij} are 1 if the j^{th} data point x_j belongs to group i and 0 otherwise.

$$u_{ij} = \begin{cases} 1 & \|x_j - c_i\|^2 \le \|x_j - c_k\|^2 \\ & k \ne i \\ 0 & \text{otherwise} \end{cases}$$

The membership matrix U has the following properties:

1. $\sum_{i=1}^{c} u_{ij} = 1 \qquad \forall j = 1,\ldots,n$
2. $\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n$

Finally, after every iteration, c_i should be updated as follows:

$$c_i = \frac{1}{|G_i|} \sum_{k,\, x_k \in G_i} x_k \text{ where } \qquad |G_i| = \sum_{j=1}^{n} u_{ij}$$

Note that the algorithm is inherently iterative, and no guarantee can be made that it will converge to an optimum solution. The performance of the K-means algorithm depends on the initial position of the cluster centers.

3.2 Fuzzy C-Means Clustering

Fuzzy C-means clustering (FCM), also known as fuzzy ISODATA, is a data clustering algorithm in which each data point belongs to a cluster to a degree specified by a membership grade [20,21].

FCM partitions a collection of n vectors x_j: $j = 1,\ldots,n$ into c fuzzy groups G_i:$i = 1,\ldots,c$, and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized. To accommodate the introduction of fuzzy partitioning, the membership matrix U is allowed to have elements with values ranging between 0.0 and 1.0 such that

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad \forall j = 1, \ldots, n$$

The cost function for FCM is then a generalization of

$$J(U, c_1, c_2, \ldots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \times d_{ij}^2$$

where u_{ij} is between 0 and 1; c_i is the cluster center of fuzzy group i; $d_{ij} = \|c_i - x_j\|$; and $m \in [1,\infty)$ is a weighting exponent.

The necessary conditions for the above equation to reach a minimum can be determined by

$$\bar{J}(U, c_1, c_2, \ldots, c_c, \lambda_1, \lambda_2, \ldots, \lambda_n) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \times d_{ij}^2 + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=}^{c} u_{ij} - 1\right)$$

where λ_j; $j = 1,\ldots,n$ are the Lagrange multipliers for the n constraints. A solution of the above problem should lead to the following formulas:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m \times x_j}{\sum_{j=1}^{n} u_{ij}^m}$$

and

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{m-1}}}$$

As in the previous case, no guarantee ensures that FCM will converge to an optimum solution. The performance depends on the initial cluster centers.

3.3 Mountain Clustering Method

The mountain clustering method is a relatively simple and effective approach to approximate estimation of cluster centers on the basis of a density measure called the *mountain function* [22, 23]. This method can be used to obtain initial cluster centers that are required by more sophisticated cluster algorithms such as fuzzy C-mean. This clustering method involves three major steps. The first step forms a grid over the data space. The second step entails constructing a mountain function representing a data density measure:

$$m(v) = \sum_{i=1}^{N} \exp\left(-\frac{\left\|v - x_i\right\|^2}{2\sigma^2}\right)$$

where x_i is the i^{th} data point and σ is an application specific constant. The mountain function can be viewed as a measure of *data density*, since it tends to be higher if more data points are located nearby and lower if fewer data points are around. The third step involves selecting the cluster centers by sequentially destructing the mountain function. First, the point in the candidate centers $v \in V$ that has the greatest value for the mountain function is found. This point will be considered as the first cluster center c_1.

Now let

$$m_{new}(v) = m(v) - m(c_1)\exp\left(-\frac{\left\|v - c_1\right\|}{2\beta^2}\right)$$

After the subtraction operation, the second cluster center is selected as the point $v \in V$ that has the greatest value for the new mountain function. This process of revising the mountain function and finding the next cluster center continues until a sufficient number of cluster centers are reached.

Mountain clustering can also be applied to identify the structure of a fuzzy model. To do this, firstly, a training data set is used to find cluster centers (x_i, y_i), and then a zero-order Sugeno fuzzy model is formed in which the i^{th} rule is expressed as

IF X is close to x_i *THEN Y is close to* y_i

Then other tuning methods can be used to tune the rules further.

3.4 Subtracting Clustering

A new approach in fuzzy clustering is *subtractive clustering*, in which data points (not grid points) are considered as candidates for cluster centers [24]. With this method, the computation is simply proportional to the number of data points and independent of the dimensional of the problem under consideration, since each data point is potentially a candidate for a cluster center. Then, a *density measure* at data point x_i is defined as

$$D_i = \sum_{j=1}^{n} \exp\left(- \frac{\left\| x_i - x_j \right\|^2}{\left(r_a/2 \right)^2} \right)$$

where r_a is a positive constant. The radius r_a defines a neighborhood; data points outside this radius contribute only slightly to the density measure.

When the density measurement for each data point has been calculated, the data point with the highest density measure is selected as the first cluster center. Let x_{c_1} be the point selected, with D_{c_1} as its density measure. Now the density measure for each data point x_i is revised by the formula

$$D_i = D_i - D_{c_1} \exp\left(- \frac{\left\| x_i - x_{c_1} \right\|^2}{\left(r_b/2 \right)^2} \right)$$

where another r_b is a positive constant. Note that the constant r_b is normally larger than r_a to prevent closely spaced cluster centers. In general, $r_b = 1.5\ r_a$.

After the density measure for each data point is revised, the next cluster center x_{c_2} is selected and all the density measures for data points are revised again. This process is repeated until sufficient cluster centers have been generated.

Like the mountain clustering algorithm, the subtractive clustering algorithm can be launched to determine fuzzy rules. For instance, assume that the center for the i^{th} cluster is c_i in an M-dimensional and that the consequent parts are assumed to have RBFN membership. In this case, the membership function μ can be assigned as

$$\mu_i = \exp\left(- \frac{\| x_i - p_i \|^2}{\left(r_b/2 \right)^2} \right)$$

3.5 Fuzzy Rules Generation

As explained earlier, each fuzzy system consists of three main components: input variables that must be fuzzified, output variables that must be defuzzified, and the most important part, namely, the rules database. The rules of a fuzzy system are the part of the system that actually relates the outputs to the inputs. It is obvious that without appropriate rules, the system may function inefficiently. Although rule generation is the most important part of a fuzzy system, it has rarely been considered because of its complexity.

Several approaches have been presented to help designers of fuzzy systems develop their rules in an efficient and concise way. However, most of these approaches have limited applicability. This section attempts to introduce some effective approaches to generate fuzzy rules [25–31]. Further, appropriate fuzzification and defuzzification methods are also important because they are correlated with the rules of the system.

3.6 Fuzzy Rules from Fuzzy Knowledge

The first approach employed for generating fuzzy rules is based on the experience of actual system operators, who usually intuitively know how to control the system. In this case, the fuzzy designer codes the ideas of an expert user into linguistic expressions, as seen earlier. The only thing the designer must consider is the consistency of the coding process so as to achieve maximum robustness of the system.

To clarify this situation, suppose a controller must be designed to control the temperature and flow of a shower using Hot and Cold values as inputs. In this case, the simplest controller can be that of Figures 7.18 and 7.19. Note that this system is a simple feedback controller that tries to reduce the difference between the actual temperature and flow rates and the desired ones (Feedback Errors).

To achieve this result, temperature and flow errors are both fuzzified by three triangular membership functions, as shown in Figures 7.20 and 7.21, while the outputs of the system are represented with three trapezoidal membership functions, as shown in Figure 7.22. Figures 7.23 and 7.24 show the temperature and flow rate of the system when their desired values are changes with square waveforms. Table 7.2 lists the rules for this system. This example is one of the Matlab® Fuzzy Logic Toolbox Demos [14].

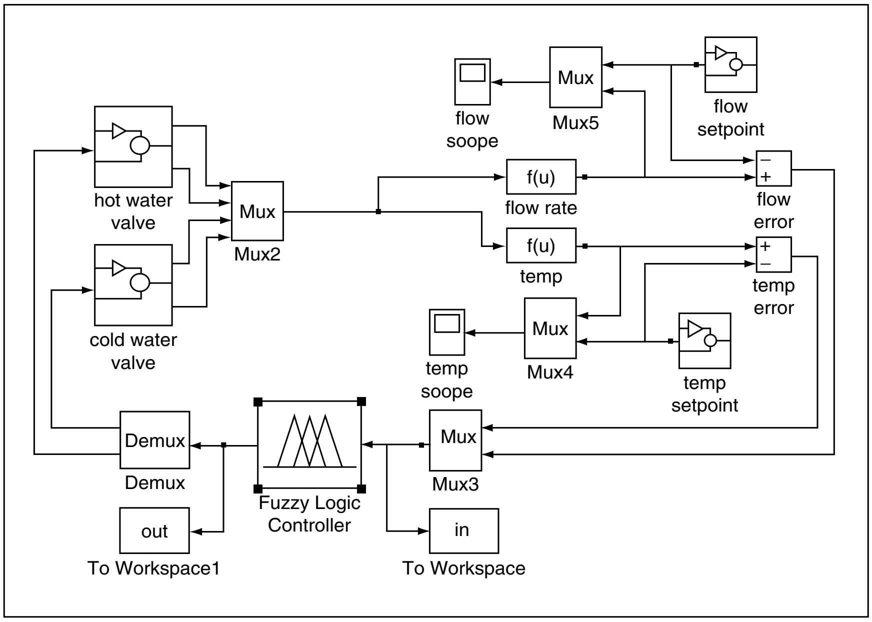
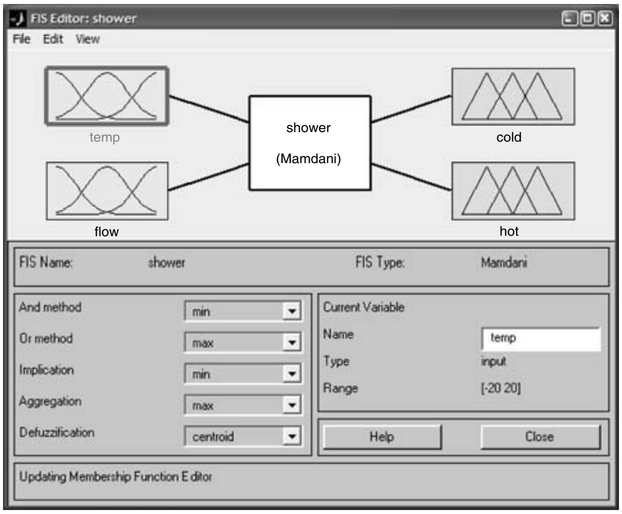


Figure 7.18. General overview of the shower system



FIS Name: shower

FIS Type: Mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Current Variable

Name: temp

Type: input

Range: [-20 20]

Updating Membership Function Editor

Figure 7.19. General overview of Mamdani's shower fuzzy system

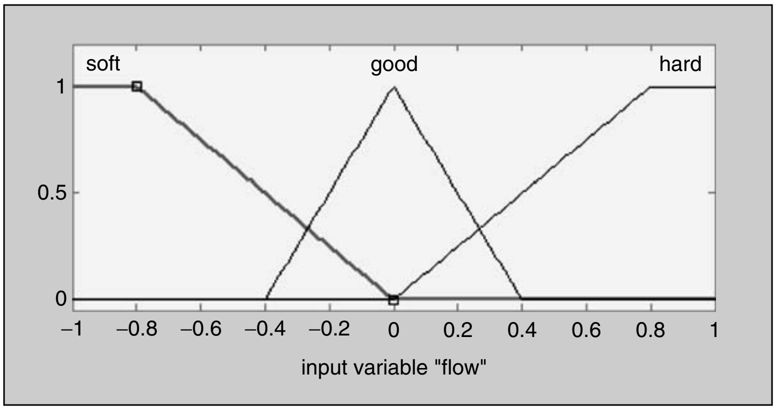


Figure 7.20. Fuzzy membership function for the Flow variable

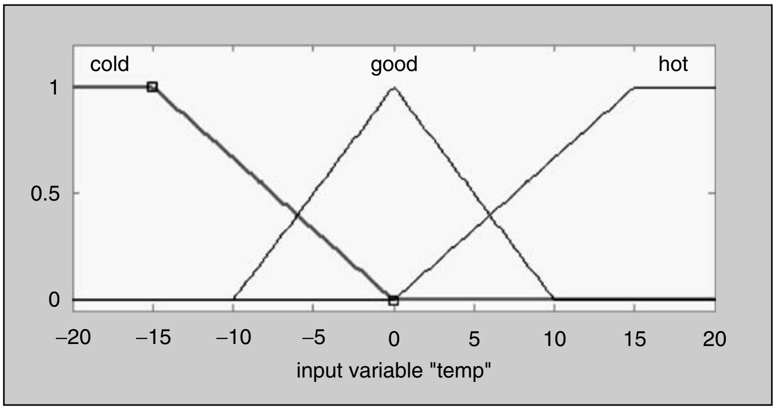


Figure 7.21. Fuzzy membership function for the Temp variable

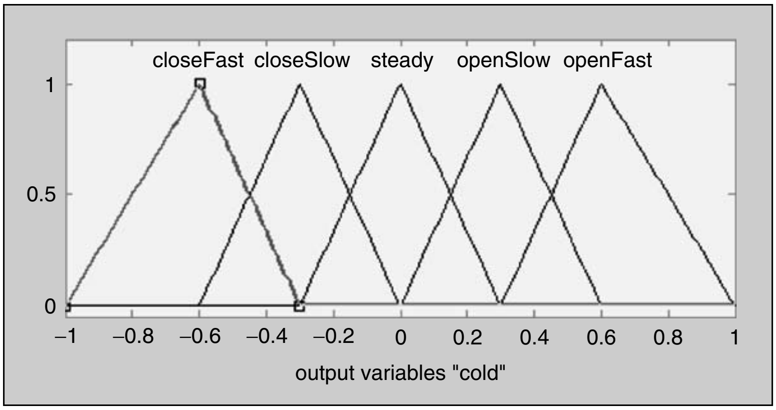


Figure 7.22. Fuzzy membership function for the Cold and Hot variables

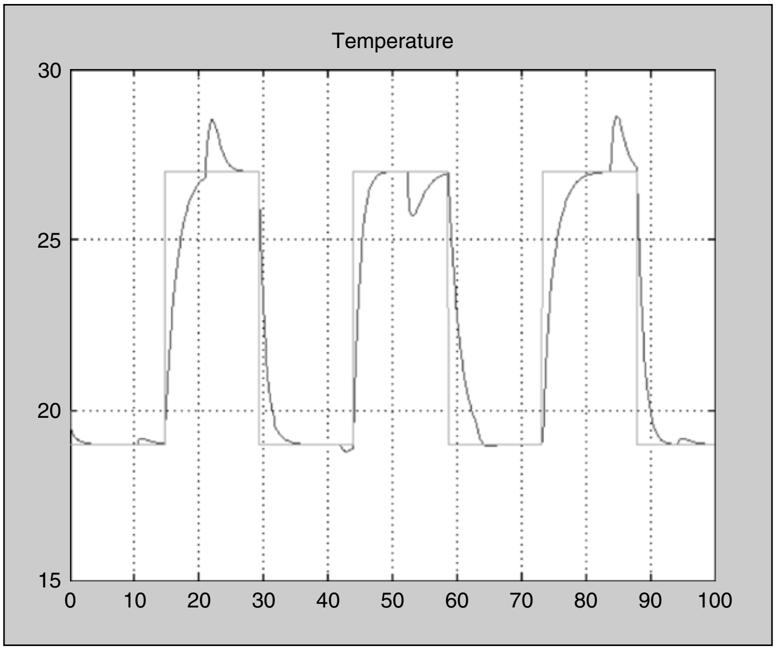


Figure 7.23. Simulation results of the system to follow the temperature curve

3.7 Fuzzy Rules from Fuzzy Patches

Fuzzy patches are actually fuzzy clusters that are generated by a given fuzzy clustering technique. Then a rule is written for each patch to imitate the behavior of the system in that condition. These patches can also be used to design a controller. In fact, the controller is designed so that it compensates the behavior of the system for each one of the patches. Then some other fuzzy rules are added to the system just to achieve overall stability for the system.

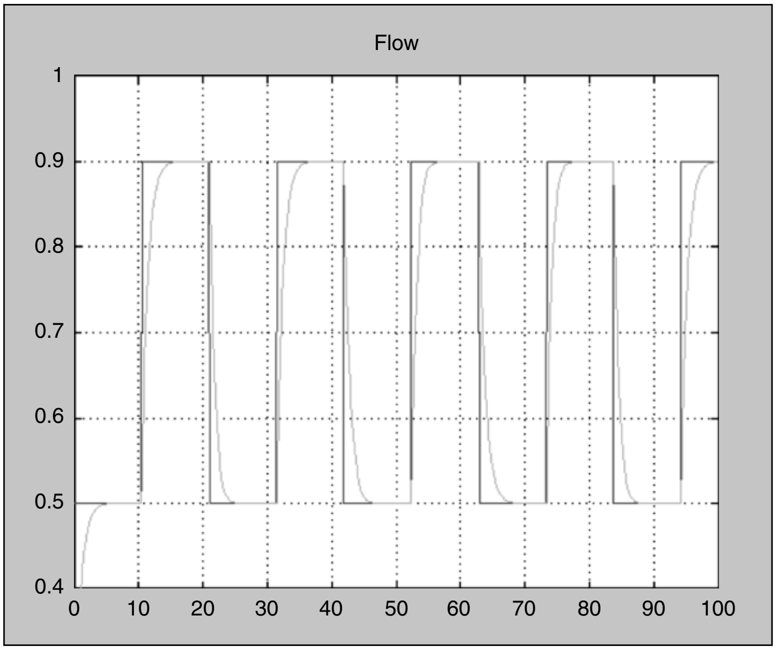


Figure 7.24. Simulation results of the system to follow the flow rate signal

3.8 Tuning Fuzzy Rules

Once the general structure of a fuzzy system has been determined, the system must be tuned to have the best performance. This process is usually performed by some optimal control routines that tune the parameters of the membership functions. In some cases, these routines even change the whole structure of the fuzzification and defuzzification processes [29].

3.9 Tuning Fuzzy Systems Using Gradient Descent Training

In this section, it is assumed that the structure of the fuzzy system is known and that the aim is to tune the parameters. In this case, a fuzzy system with a Gaussian membership function and COA defuzzification method is considered:

$$f(x) = \frac{\sum_{l=1}^{M} \overline{y}^{1} \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \overline{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \overline{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}$$

where M is the number of rules and $\overline{y}^l, \overline{x}_i^l, \sigma_i^l$ are free parameters to be adjusted. Note that, although the structure of the system is chosen, the whole system has not been designed yet because of the $\overline{y}^l, \overline{x}_i^l, \sigma_i^l$ parameters. To determine these

Table 7.2. Fuzzy rules for the system shown in Figure 7.18

	Antecedent			➔	Consequent		
IF	Temp is Cold	And	Flow is Soft	Then	CldVlv is OpenSlow	And	HotVlv is OpenFast
IF	Temp is Cold	And	Flow is Good	Then	CldVlv is CloseSlow	And	HotVlv is OpenSlow
IF	Temp is Cold	And	Flow is Hard	Then	CldVlv is CloseFast	And	HotVlv is CloseSlow
IF	Temp is Good	And	Flow is Soft	Then	CldVlv is OpenSlow	And	HotVlv is OpenSlow
IF	Temp is Good	And	Flow is Good	Then	CldVlv is Steady	And	HotVlv is Steady
IF	Temp is Good	And	Flow is Hard	Then	CldVlv is CloseSlow	And	HotVlv is CloseSlow
IF	Temp is Hot	And	Flow is Soft	Then	CldVlv is OpenFast	And	HotVlv is OpenSlow
IF	Temp is Hot	And	Flow is Good	Then	CldVlv is OpenSlow	And	HotVlv is CloseSlow
IF	Temp is Hot	And	Flow is Hard	Then	CldVlv is CloseSlow	And	HotVlv is CloseFast

parameters, it is helpful to represent the fuzzy system $f(x)$ as a feedforward network—specifically, the mapping from the input $x \in U \subset \Re^n$ to the output, $f(x) \in V \subset \Re$.

Now in order to design the parameters by the Gradient Descent Method, the matching error of the system is assigned as follows:

$$e^p = \frac{1}{2}(f(x_0^p) - y_0^p)^2$$

Considering a minimization problem, the $\bar{y}^l, \bar{x}_i^l, \sigma_i^l$ parameters should be adjusted such that e^p is minimized. In this case, using the gradient descent algorithm, the following formulas are used to tune these parameters. $\bar{y}^l$ would be adjusted as follows:

$$\bar{y}^l(q+1) = \bar{y}^l(q) - \alpha \left.\frac{\partial e}{\partial \bar{y}^1}\right|_q$$

$$\frac{\partial e}{\partial \bar{y}^l} = (f - y)\frac{\partial f}{\partial a}\frac{\partial a}{\partial \bar{y}^l} = (f - y)\frac{1}{b}z^l$$

$$\bar{y}^l(q+1) = \bar{y}^1(q) - \alpha \frac{f - y}{b} z^l$$

$\bar{x}_i^l$ as follows:

$$\bar{x}_i^l(q+1) = \bar{x}_i^l(q) - \alpha \left.\frac{\partial e}{\partial \bar{x}_i^l}\right|_q$$

$$\frac{\partial e}{\partial \bar{x}_i^l} = (f - y)\frac{\partial f}{\partial z^l}\frac{\partial z^l}{\partial \bar{x}_i^l}$$

$$\bar{x}_i^l(q+1) = \bar{x}_i^l(q) - \alpha \frac{f - y}{b}(\bar{y}_i^l(q) - f)z^l \frac{2(x_{0i}^p - \bar{x}_i^l(q))}{\sigma_i^{l^2}(q)}$$

and finally σ_i^l as follows:

$$\sigma_i^l(q+1) = \sigma_i^l(q) - \alpha \left.\frac{\partial e}{\partial \sigma_i^l}\right|_q$$

$$\frac{\partial e}{\partial \sigma_i^l} = (f - y)\frac{\partial f}{\partial z^l}\frac{\partial z^l}{\partial \sigma_i^l}$$

$$\sigma_i^l(q+1) = \sigma_i^l(q) - \alpha \frac{f - y}{b}(\bar{y}^l(q) - f)z^l \frac{2(x_{0i}^p - \bar{x}_i^l(q))^2}{\sigma_i^{l^3}(q)}$$

This algorithm is also called the *error back-propagation training algorithm*. The following algorithm is the final procedure that can be used to adjust the parameters of a fuzzy system using the gradient descent technique.

Step 1: Structure determination and initial parameter setting

Choose the fuzzy system in the above form and determine the M. Have in mind that larger values for M need more computation as well, but better accuracy. The initial parameters $\bar{y}^l$ (0), $\bar{x}_i^l$ (0), σ_i^l (0) must be chosen carefully, too. These initial parameters may be determined according to the linguistic rules from experts or any other clustering technique.

Step 2: Present input and calculate the output of the fuzzy system

For a given input–output pair (x_0^p, y_0^p), p = 1,2,..., the following auxiliary parameters are calculated, where q is the iteration cycle:

$$z^l = \prod_{i=1}^{n} \exp\left(-\left(\frac{x_{0i}^p - \bar{x}_i^l(q)}{\sigma_i^l(q)}\right)^2\right)$$

$$b = \sum_{l=1}^{M} z^l$$

$$a = \sum_{l=1}^{M} \bar{y}^l(q) z^l$$

$$f = \frac{a}{b}$$

Step 3: Update the parameters

Modify the parameters $\bar{y}^l(q+1)$, $\bar{x}_i^l(q+1)$, σ_i^l (q + 1) based on the results of Step 2, where $y = y_0^p$.

Step 4: Repeat Steps 2 and 3 with $q = q + 1$ for a predefined number of iterations, or until the output error of the system $|f - y_0^p|$ becomes less than another predefined value ε.

Step 5: Repeat Steps 2 through 4 with $p = p + 1$, that is, update parameters using the next input–output pair (x_0^{p+1}, y_0^{p+1}).

Step 6: Repeat the whole training procedure if applicable.

If desirable and feasible, set p = 1 and repeat Steps 2–5 until the designed fuzzy system is satisfactory. For online control and dynamic system identification, this step is not feasible because the input–output pairs are provided one-by-one in a real-time fashion. However, for pattern recognition problems where the input–output pairs are provided offline, this step is desirable.

Note that, because of the nature of the above training algorithms, choosing the initial parameters is crucial to the success of the algorithm. If the initial parameters are chosen close to the optimal ones, the algorithm has a good chance of converging to the optimal solution; otherwise, the algorithm may converge to a nonoptimal solution or even diverge.

Setting the Initial Parameters

The choice of initial parameters is detrimental to the overall quality of the final solution. In some cases, these parameters can be selected by experts, but in other occasions this is not possible. So, to solve the above identification problem, the following method is proposed for setting the initial parameters [29].

An online initial parameter choosing method:

Step 1: Collect the input–output pairs

$$(x_0^{k+1}, y_0^{k+1})$$

where

$$x_0^{k+1} = (y(k),...,y(k-n+1),u(k),...,u(k-m+1))$$

for the first

$$y_0^{k+1} = y(k+1)$$

M points ($k = 1,\ldots,M-1$).

Note that the training algorithm is actually started when $k = M - 1$.

Step 2: Choose the initial parameters

These parameters are chosen as $\bar{y}^l(0) = y_0^l$ and $\bar{x}^l(0) = x_{0i}^l$, while $\sigma_i^l(0)$ can be

set according to one of the following criteria:

1. Set σ_i^l (0) to a small number
2. Set $\sigma_i^l(0) = \dfrac{\max_{l=1,\ldots,M}(x_{0i}^l) - \min_{l=1,\ldots,M}(x_{0i}^l)}{M}, i = 1,\ldots,n+m$
3. Set σ_i^l (0) so that it makes the membership functions uniformly cover the range of $x_{0i}{}^l$ from $l = 1$ to $l = M$.

The following lemma is a stability proof of the presented technique.

Lemma: For any arbitrary $\varepsilon > 0$, there exist $\sigma^* > 0$ such that the fuzzy system $\hat{f}(x)$, with the preceding initial parameters $\bar{y}^l$, $\bar{x}_i^l$, and $\sigma_i^l = \sigma^*$, has the property that

$$\left|\hat{f}(x_0^{k+1}) - y_0^{k+1}\right| < \epsilon \qquad k = 0, 1, \ldots, M-1.$$

Note that, by using this method, the first M input–output pairs will be properly matched. Thus, if these first M input–output pairs contain important features of the unknown system $f(x)$, it is very likely that, after training, the fuzzy identifier will converge rapidly and determine the unknown parameters of the system.

3.10 Design of Fuzzy Systems Using Recursive Least Squares

The gradient descent algorithm in the previous section tries to minimize the criterion $e^p(e^p = \frac{1}{2}(f(x_0^p) - y_0^p)^2)$, which actually accounts for the matching error

of only one input–output pair ($x_0{}^p$, $y_0{}^p$). In other words, the training algorithm updates the parameters to match one input–output pair at a time. In this new approach, a training algorithm that minimize the summation of the matching errors for all the input–output pairs up to p is used to adjust the training parameters; that is, the objective here is to design a fuzzy system $f(x)$ to minimize the following cost function:

$$J_p = \sum_{j=1}^{p}\left(f(x_0^j) - y_0^j\right)^2$$

Moreover, the fuzzy system is designed iteration by iteration in a recursive manner; that is, if f_p is the fuzzy system designed to minimize J_p, then f_p should be represented as a function of f_{p-1}. To accomplish this, the recursive least squares algorithm is used as follows:

Step 1: Suppose that $U = [\alpha_1, \beta_1] \times \ldots \times [\alpha_n, \beta_n] \subset \Re^n$. Then, for each $[\alpha_i, \beta_i]$, $i = 1,2,\ldots,n$, define N_i fuzzy sets as $A_i^{l_i}$, $l_i = 1, 2,\ldots, N_i$, which cover $[\alpha_i, \beta_i]$ homogenously.

Step 2: Construct the fuzzy system from the following $\prod_{i=1}^{n} N_i$ fuzzy *IF – THEN* rules as follows:

$$IF\ x_1\ is\ A_1^{l_1}\ and \ldots and\ x_n\ is\ A_n^{l_n}\ THEN\ y\ is\ B^{l_1,\ldots,l_n}$$

where $l_i = 1, 2,\ldots, N_i$, $i = 1, 2,\ldots, n$, and $B^{l_1,\ldots,l_n}$ is any fuzzy set with center at $\bar{y}^{l_1,\ldots,l_n}$ (which is free to change). In particular, when the fuzzy system with product inference engine, singleton fuzzifier, and COA defuzzifier is chosen with the following formula:

$$f(x) = \frac{\sum_{l_1=1}^{N_1}\cdots\sum_{l_n=1}^{N_n}\bar{y}^{l_1,\ldots,l_n}\left[\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right]}{\sum_{l_1=1}^{N_1}\cdots\sum_{l_n=1}^{N_n}\left[\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right]}$$

where $\bar{y}^{l_1,\ldots,l_n}$ are free parameters (that need to be properly chosen).

Step 3: Collect the free parameters $\bar{y}^{l_1,\ldots,l_n}$ into the $\prod_{i=1}^{n} N_i$ -dimensional vector as follows:

$$\theta = (\bar{y}^{1\ldots1},\ldots,\bar{y}^{N_1 1\ldots1},\bar{y}^{121\ldots1},\ldots,\bar{y}^{N_1 21\ldots1},\bar{y}^{1N_2\ldots N_n},\ldots,\bar{y}^{N_1 N_2\ldots N_n})^T$$

to form $f(x) = b^T(x)\cdot\theta$, where

$$b(x) = (b^{1\ldots1},\ldots,b^{N_1 1\ldots1},b^{121\ldots1},\ldots,b^{N_1 21\ldots1},b^{1N_2\ldots N_n},\ldots,b^{N_1 N_2\ldots N_n})^T$$

and

$$b^{l_1,\ldots,l_n}(x) = \frac{\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)}{\sum_{l_1=1}^{N_1}\cdots\sum_{l_n=1}^{N_n}\left[\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right]}$$

Step 4: Choose the initial parameters $\theta(0)$ as follows:

If there are linguistic rules from experts whose *IF* parts agree with the *IF* parts of one of the existing rules, then choose $\bar{y}^{l_1,\ldots,l_n}(0)$ to be the centers of the *THEN* part fuzzy sets in these linguistic rules; otherwise, choose $\theta(0)$ arbitrary in the output space $V \subset \Re$; or from a clustering algorithm.

Step 4: For $p = 1, 2,\ldots,$ compute the parameter θ using the following recursive least squares algorithm:

$$\theta(p) = \theta(p-1) + K(p)\cdot[y_0^p - b^T(x_0^p)\,\check{}\,\theta(p-1)]$$
$$K(p) = P(p-1)\cdot b(x_0^p)\cdot[b^T(x_0^p)\cdot P(p-1)\cdot b(x_0^p) + 1]^{-1}$$
$$P(p) = P(p-1) - P(p-1)\cdot b(x_0^p).$$
$$[b^T(x_0^p)\cdot P(p-1)\cdot b(x_0^p) + 1]^{-1}\, b^T(x_0^p)\cdot P(p-1)$$

where $\theta(0)$ is chosen from Step 4, and $P(0) = \sigma I$, where σ is a large constant. In this fuzzy system, the parameters $\bar{y}^{l_1,\ldots,l_n}$ are equal to the corresponding elements in $\theta(p)$.

4 DESIGN OF FUZZY SYSTEMS USING CLUSTERING

In this section, the input–output pairs are used to design the rules for the fuzzy system. Basically, the input–output pairs are grouped into clusters and one rule is formulated for each cluster [32–37].

4.1 An Adaptive Fuzzy System

Suppose that N input–output pairs (x_0^l, y_0^l), $l = 1, 2,..., N$, are given and the task is to construct a fuzzy system $f(x)$ that can match all the N pairs with a given accuracy. That is, for any given $\varepsilon > 0$, it is required to satisfy $|f(x_0^l) - y_0^l| < \varepsilon$ for all $l = 1,2,..., N$. In this case, the optimal fuzzy system is considered as

$$f(x) = \frac{\sum_{l=1}^{N} y_0^l \exp\left(-\frac{|x - x_0^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{N} \exp\left(-\frac{|x - x_0^l|^2}{\sigma^2}\right)}$$

while the membership functions are

$$\mu_{A_i^l}(x_i) = \exp\left(-\frac{|x - x_0^l|^2}{\sigma^2}\right)$$

In this case, the designed optimal fuzzy system will have one rule for one input–output pair. Therefore, the larger the number of input–output pairs, the larger the number of rules in the system. To solve this problem, various clustering techniques can be used to categorize the input–output pairs and, consequently, reduce the number redundant rules.

4.2 Design of Fuzzy System Using Nearest-Neighbor Clustering

Use of the nearest-neighbor technique is one of the most effective ways to design fuzzy systems [34, 35]. This technique can be summarized as follows:

Step 1: Starting with the first input–output pair (x_0^l, y_0^l), establish a cluster center x_c^1 at x_0^1, and set $A^1(1) = y_0^1$, $B^1(1) = 1$. Select a radius r.

Step 2: Suppose that the algorithm is going to assign the k^{th} input–output pair (x_0^k, y_0^k), $k = 2, 3,...$, to a cluster when there are M clusters with centers at $x_c^1, x_c^2,..., x_c^M$.

Step 3: Compute the distance of x_0^k to those M cluster centers, and then find the nearest cluster to x_0^k, namely, $x_c^{l_k}$. Then:

- If $|x_0^k - x_c^{l_k}| > r$, establish x_0^k as a new cluster center $x_c^{M+1} = x_0^k$, set $A^{M+1}(k) = y_0^k$, $B^{M+1}(k) = 1$

  and keep

  $$\left.\begin{aligned} A^l(k) &= A^l(k-1) \\ B^l(k) &= B^l(k-1) \end{aligned}\right. \quad \text{for all } l = 1, 2,...,M.$$

- If $|x_0^k - x_c^{l_k}| \le r$, do the following:

$A^{l_k}(k) = A^{l_k}(k-1) + y_0^k$

$B^{l_k}(k) = B^{l_k}(k-1) + 1$

and set

$A^l(k) = A^l(k-1)$
$B^l(k) = B^l(k-1)$ for all $l = 1,2,...,M$.

Step 3: If x_0^k does not establish a new cluster, then the designed fuzzy system based on the k input–output pairs (x_0^j, y_0^j), $j = 1, 2,...,k$ is

$$f_k(x) = \frac{\sum_{l=1}^{M} A^l(k)\exp\left(-\frac{|x-x_c^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{M} B^l(k)\exp\left(-\frac{|x-x_c^l|^2}{\sigma^2}\right)}$$

Step 4: If x_0^k establishes a new cluster, then the designed fuzzy system is

$$f_k(x) = \frac{\sum_{l=1}^{M+1} A^l(k)\exp\left(\frac{-|x-x_c^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{M+1} B^l(k)\exp\left(\frac{-|x-x_c^l|^2}{\sigma^2}\right)}$$

Step 5: Repeat by returning to Step 2 with $k = k+1$ until the process converges to a satisfactory solution.

5 FUZZY APPLICATIONS

This section presents two popular applications that demonstrate the potential of fuzzy logic in solving complex problems [36–38].

6 APPLICATION TO NONLINEAR DYNAMIC SYSTEM IDENTIFICATION

System identification is a process of determining an appropriate model for a system based on measurement form sensors [36, 37]. This process is important because many applications in science and engineering depend on the accurate modeling of a real-world system. In this section, a fuzzy system is used to approximate the unknown nonlinear components of a dynamic system. Now, consider a discrete-time nonlinear dynamic system as follows:

$$y(k+1) = f(y(k),..., y(k-n+1), u(k),..., u(k-m+1))$$

where f is an unknown function that needs to be "identified", u and y are the inputs and outputs of the system, respectively, and n and m are positive integers. Now let $\hat{f}(x)$ be the fuzzy system that is supposed to be an approximate of the real system f.

$$y(k+1) = \hat{f}(y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1))$$

Based on the identification scheme given in Figure 7.25, the aim is to adjust the parameters of $\hat{f}(x)$ such that the output of the identification model $\hat{y}(k+1)$ converges to the output of the real system $y(k+1)$ as $k \to \infty$.

To achieve this outcome, any of the previously presented tuning algorithms can be used with the following formulation. The input–output pairs in this problem are (x_0^{k+1}, y_0^{k+1}), where

$$x_0^{k+1} = (y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-m+1))$$
$$y_0^{k+1} = y(k+1) \qquad\qquad k = 0, 1, 2, \ldots$$

Now the system parameters are modified iteration by iteration to follow the real output.

6.1 Fuzzy robot navigator

Robot control is another area that benefited from advances in fuzzy logic [38]. A fuzzy navigator is designed to control a robot that moves around a room containing several static obstacles (chairs, tables, etc) and dynamic obstacles (humans). Now the idea is that a fuzzy navigator will aid the robot to get to any arbitrary point in the room from any other arbitrary point without colliding with any static or dynamic obstacle [39].

In summary, the robot is equipped with ultrasonic sensors to detect its surrounding obstacles. These sensors are mounted on the front, left, and right side of the robot. Three completely individual controllers were designed to seek the goal, avoid obstacles, and follow edges in the room. Figure 7.26 shows the general overview of the controller, while Figures 7.27 and 7.28 are two examples of launching the proposed algorithm in the presence of dynamic and static obstacles. In these figures, the robot starts from the "S" point to get the target point "T".

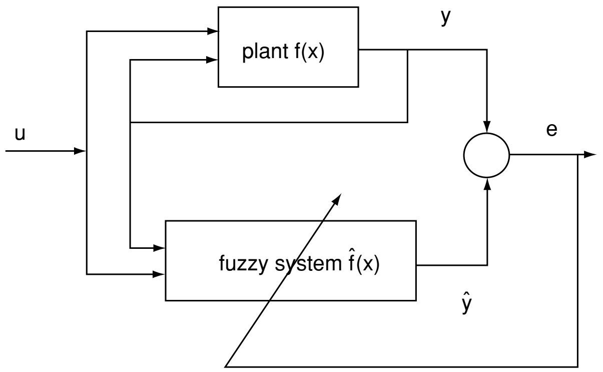


Figure 7.25. A fuzzy identification system

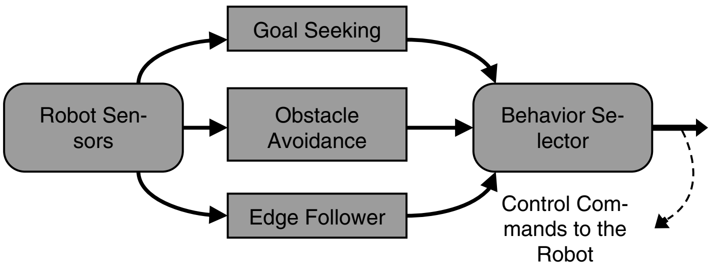


Figure 7.26. A fuzzy robot navigator

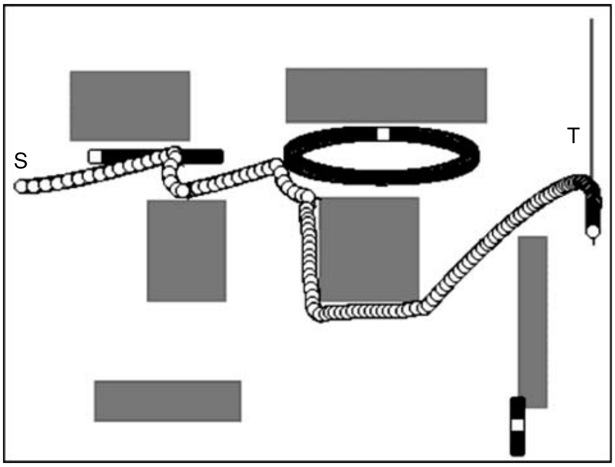


Figure 7.27. A path generated in the presence of static and dynamic obstacles with a moving target

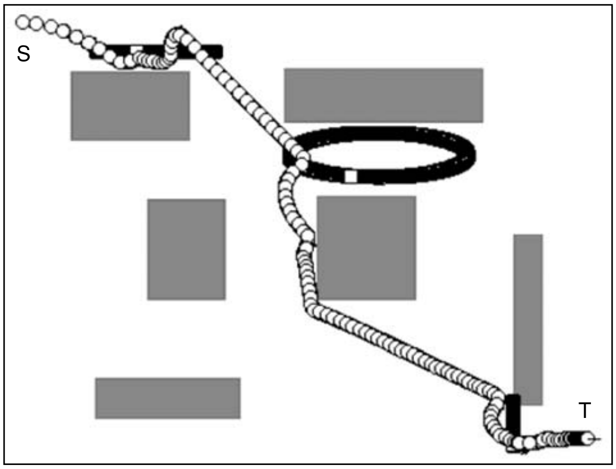


Figure 7.28. A path generated in the presence of static and dynamic obstacles

Static obstacles are shown in gray, are dynamic (moving) obstacles are shown in black. The robot itself is shown as a circle with a tick to show its head angle. Note that, in Figure 7.27, the target is also a moving point, such as a carriage.

7 CONCLUSION

In this chapter, a general overview of the fuzzy logic has been presented. The premise of fuzzy logic relies on the fact that decisions in the real world may not be clear-cut, especially in complex scenarios. Fuzzy logic is a powerful tool that can be applied to a wide range of applications ranging from fuzzy control to fuzzy decision makers and fuzzy classifiers.

REFERENCES

[1] B. Kosko (1994): Fuzzy Thinking: The New Science of Fuzzy Logic. *Hyperion*, Reprint edition.

[2] H.-J. Zimmermann (2001): Fuzzy Set Theory and its Applications, 4th ed. Kluwer Academic Publishers.

[3] H. Ying, Y. Ding, S. Li, and S. Shao (1999): Comparison of necessary conditions for typical Takagi-Sugeno and Mamdani fuzzy systems as universal approximators, *IEEE Transactions on Systems, Man and Cybernetics (Part A)*, *29*(5), 508–514.

[4] Y. Ding, H. Ying, and S. Shao (2000): Necessary conditions on minimal system configuration for general MISO Mamdani fuzzy systems as universal approximators, *IEEE Transactions on Systems, Man and Cybernetics (Part B), 30*(6), 857–864.

[5] P. Liu (2002): Mamdani fuzzy system: universal approximator to a class of random processes, *IEEE Transactions on Fuzzy Systems*, *10* (6), 756–766.

[6] K. Tanaka, T. Taniguchi, and H. O. Wang (2000): Generalized Takagi-Sugeno fuzzy systems: rule reduction and robust control, in *Proc. Ninth IEEE International Conference on Fuzzy Systems*, *2*, 688–693.

[7] V. Catania, G. Ficili, S. Palazzo, and D. Panno (1995): A fuzzy decision maker for source traffic control in high speed networks, in *Proc. International Conference on Network Protocols*, pp. 136–143.

[8] Q. M. Wu and C. W. de Silva (1993): Automatic adjustment of the cutting position of a vision-based fish processing machine, in *Proc. IEEE Pacific Rim Conference on Communications*, Computers and Signal Processing, *2*, 702–705.

[9] H. R. Beom and H. S. Cho (2000): Sonar-based navigation experiments on a mobile robot in indoor environments, in *Proc. IEEE International Symposium on Intelligent Control*, pp. 395–401.

[10] H.M. Tai and S. Shenoi (1994): Robust fuzzy controllers, in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, *1*, 85–90.

[11] S. Galichet and L. Foulloy (1995): Fuzzy controllers: synthesis and equivalences, *IEEE Transactions on Fuzzy Systems*, *3*(2), 140–148.

[12] W. Barra, Jr. (1998): A practical and useful self-learning fuzzy controller, in *Proc. Int. Conf. Control (Control '98)*, Sept. 1-4, 1998, *1*, 290–295.

[13] C. W. Tao and J. Taur (1999): Design of fuzzy controllers with adaptive rule insertion, *IEEE Transactions on Systems, Man and Cybernetics (Part B), 29*(3), 389–397.

[14] Fuzzy Toolbox, Matlab Released Version 13.

[15] S. Abe (1998): Dynamic cluster generation for a fuzzy classifier with ellipsoidal regions, *IEEE Transactions on Systems, Man and Cybernetics (Part B)*, *28*(6), 869–876.

[16] J. G. Marin-Blazquez and Q. Shen (2002): From approximative to descriptive fuzzy classifiers, *IEEE Transactions on Fuzzy Systems*, *10*(4), 484–497.

[17] O. Takata, S. Miyamoto, and K. Umayahara (2001): Fuzzy clustering of data with uncertainties using minimum and maximum distances based on L1 metric, in *Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, *5*, 2511–2516.

[18] L.-J. Kau (2003): Adaptive predictor with dynamic fuzzy K-means clustering for lossless image coding, in *Proc. 12th IEEE International Conference on Fuzzy Systems*, *2*, 944–949.

[19] N. Watanabe and T. Imaizumi (2001): Fuzzy k-means clustering with crisp regions, in *Proc. 10th IEEE International Conference on Fuzzy Systems*, *1*, 199–202.

[20] Y. Bo, G. J. Klir, and J. F. Swan-Stone (1995): Evolutionary fuzzy C-means clustering algorithm, in *Proc. Joint Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium*, *4*, 2221–2226.

[21] M.-C. Hung and D.-L. Yang (2001): An efficient Fuzzy C-Means clustering algorithm, in *Proc. IEEE International Conference on Data Mining*, pp. 225–232.

[22] J. W. Lee, S. H. Son, and S. H. Kwon (2001): Advanced mountain clustering method, in *Proc. Joint 9th IFSA World Congress and 20th NAFIPS, July 1*, 275–280.

[23] P. J. Costa Branco, N. Lori, and J. A. Dente (1995): An autonomous approach to the mountain-clustering method, in *Proc. Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society*, pp. 649–654.

[24] W.-Y. Liu, C.-J. Xiao, B.-W. Wang, Y. Shi, and S.-F. Fang (2003): Study on combining subtractive clustering with fuzzy C-means clustering, in *Proc. Int. Conf. Machine Learning and Cybernetics*, *5*, 2659–2662.

[25] S. Mitra and Y. Hayashi (2000): Neuro-fuzzy rule generation: survey in soft computing framework, *IEEE Transactions on Neural Networks*, *11*(3), 748–768.

[26] C.-S. Fahn, K.-T. Lan, and Z.-B. Chern (1999): Fuzzy rules generation using new evolutionary algorithms combined with multilayer perceptrons, *IEEE Transactions on Industrial Electronics*, *46*(6), 1103–1113.

[27] T.M. McKinney and N. Kehtarnavaz (1997): Fuzzy rule generation via multi-scale clustering, in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, *4*, 3182–3187.

[28] J. Wang, L. Shen, and J.-F. Chao (1997): An efficient method of fuzzy rules generation, in *Proc. IEEE International Conference on Intelligent Processing Systems*, *1*, 295–299.

[29] Y. Shi, M. Mizumoto, N. Yubazaki and M. Otani (1996): A learning algorithm for tuning fuzzy rules based on the gradient descent method, in *Proc. IEEE International Conference on Fuzzy Systems*, *1*, 55–61.

[30] X. Chang, W. Li, and J. Farrell (2000): A C-means clustering based fuzzy modeling method, in *Proc. Ninth IEEE International Conference on Fuzzy Systems*, *2*, 937–940.

[31] M.-S. Chen and R.-J. Liou (1999): An efficient learning method of fuzzy inference system, in *Proc. IEEE International Fuzzy Systems*, *2*, 634–638.

[32] T.-W. Hung, S.-C. Fang, and H. L. W. Nuttle (1999): An easily implemented approach to fuzzy system identification, in *Proc. 18th International Conference of the North American Fuzzy Information Processing Society*, pp. 492–496.

[33] Y. Wang and G. Rong (1997): A self-organizing neural-network-based fuzzy system, in *Proc. Fifth International Conference on Artificial Neural Networks*, pp. 106–110.

[34] I. Burham Turksen, B. A. Sproule, and C. A. Naranjo (2001): A k-nearest neighborhood based fuzzy reasoning schema, in *Proc. 10th IEEE International Conference on Fuzzy Systems*, *1*, 236–239.

[35] L.-X. Wang (1993): Training of fuzzy logic systems using nearest neighborhood clustering, in *Proc. Second IEEE International Conference on Fuzzy Systems*, *1*, 13–17.

[36] F. Wan, L.-X. Wang, H.-Y. Zhu, and Y.-X. Sun (2001): Generating persistently exciting inputs for nonlinear dynamic system identification using fuzzy models, in *Proc. IEEE International Conference on Fuzzy Systems*, *1*, 505–508.

[37] A. Lo Schiavo and A. M. Luciano (2001): Powerful and flexible fuzzy algorithm for nonlinear dynamic system identification, *IEEE Transactions on Fuzzy Systems*, *9*(6), 828–835.

[38] J. Taheri and N. Sadati (2003): A fully modular online controller for robot navigation in static and dynamic environments, in *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, *1*, 163–168.