6 Fuzzy Logic Systems

Together with neural networks, fuzzy logic models constitute the modeling tools of soft computing, and it seems appropriate to start with a short definition of fuzzy logic: *Fuzzy logic is a tool for embedding structured human knowledge into workable algorithms.*

One can say, paraphrasing Zadeh (1965; 1973), that the concept of fuzzy logic is used in many different senses. In a narrow sense, fuzzy logic (FL) is considered a logical system aimed at providing a model for modes of human reasoning that are approximate rather than exact. In a wider sense, FL is treated as a fuzzy set theory of classes with unsharp or fuzzy boundaries. Fuzzy logic methods can be used to design intelligent systems on the basis of knowledge expressed in a common language. The application areas of intelligent systems are many. There is practically no area of human activity left untouched by these systems today. The main reason for such versatility is that this method permits the processing of both symbolic and numerical information. Systems designed and developed utilizing FL methods have often been shown to be more efficient than those based on conventional approaches.

Here, the interest is chiefly in the role of FL as a technique for mathematical expression of linguistic knowledge and ambiguity. In order to follow the presentation, it is first useful to understand the relation between human knowledge and basic concepts such as sets and functions. A graphical presentation of these relations is given in figure 6.1. It seems natural to introduce FL modeling following the bottom-up approach outlined in the figure: from sets, their operations and their Cartesian products to relations, multivariate functions, and IF-THEN rules as a linguistic form of structured human knowledge.

Consequently, in section 6.1, the basics of fuzzy set theory are explained and compared with classic crisp logic. The important concept of the membership function is discussed. The representation of fuzzy sets by membership functions will serve as an important link with neural networks. Basic set operations (notably intersection and union) are presented and connected with the proper operators (for example, MIN and MAX). Then the concepts of (fuzzy) relations, the relational matrix, and the composition of fuzzy relations are examined. A formal treatment of fuzzy IF-THEN statements, questions of fuzzification and defuzzification, and the compositional rule of fuzzy inference with such statements concludes the section on the basics of fuzzy logic theory.

In earlier chapters it was mentioned that neural networks and fuzzy logic models are based on very similar, sometimes equivalent, underlying mathematical theories. This very important and remarkable result, which has been discovered by different researches independently, is discussed in section 6.2. The development follows a paper by Kecman and Pfeiffer (1994), which shows when and how learning of fuzzy

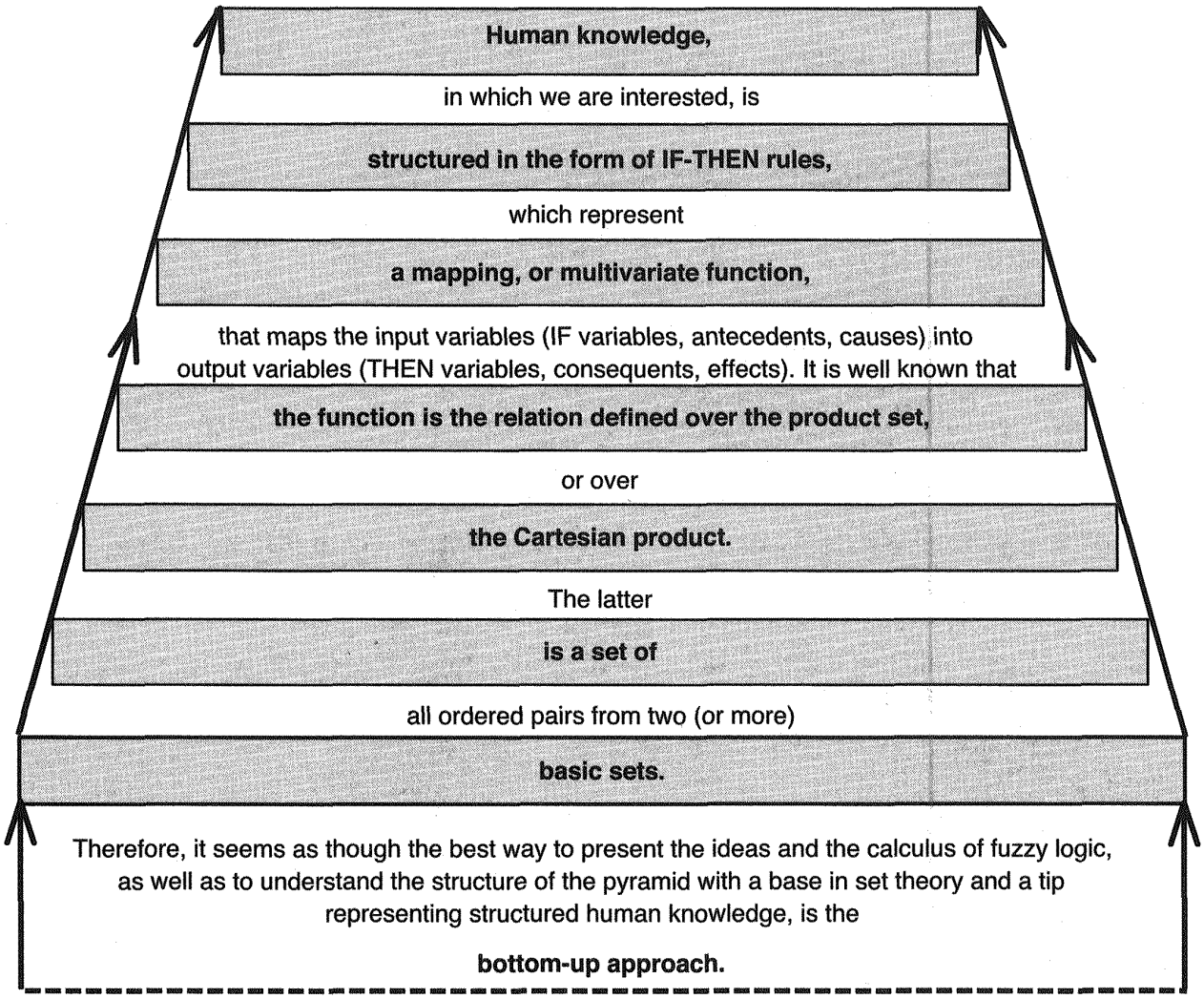


Figure 6.1
Pyramid of structured human knowledge in the world of fuzzy logic.

rules from numerical data is mathematically equivalent to the training of a radial basis function, or regularization, network. Although these approaches originate in different paradigms of intelligent information processing, it is demonstrated that the mathematical structure is the same. The presentation in section 6.2 can be readily extended to other, not necessarily radial, activation functions.

Finally, in section 6.3 fuzzy additive models (FAMs) are introduced. They are naturally connected with, and represent an extension of, the soft-radial basis models from section 6.2. FAMs are universal approximators. They are very powerful fuzzy modeling tools, and unlike early fuzzy models that used the MAX operator, FAMs add the THEN-parts of all active rules, that is, they use the SUM operator.

6.1 Basics of Fuzzy Logic Theory

The theory of fuzzy sets is a theory of graded concepts—"a theory in which everything is a matter of degree, or everything has elasticity" (Zadeh 1973). It is aimed at dealing with complex phenomena that "do not lend themselves to analysis by a classical method based on bivalent logic and probability theory." Many systems in real life are too complex or too ill-defined to be susceptible to exact analysis. Even where systems or concepts seem to be unsophisticated, the perception and understanding of such seemingly unsophisticated systems are not necessarily simple. Using fuzzy sets or classes that allow intermediate grades of membership in them, opens the possibility of analyzing such systems both qualitatively and quantitatively by allowing the system variables to range over fuzzy sets.

6.1.1 Crisp (or Classic) and Fuzzy Sets

Sets[1] or *classes* in a *universe of discourse* (universe, domain) U could be variously defined:

1. By a list of elements:

$S_1 = \{Ana, John, Jovo, Mark\}$. $S_2 = \{beer, wine, juice, slivovitz\}$.

$S_3 = \{horse, deer, wolf, sheep\}$. $S_4 = \{1, 2, 3, 5, 6, 7, 8, 9, 11\}$.

2. By definition of some property:

$S_5 = \{x \in N \mid x < 15\}$. $S_6 = \{x \in R \mid x^2 < 25\}$.

Note that $S_4 \in S_5$.

$S_7 = \{x \in R \mid x > 1 \wedge x < 7\}$. $S_8 = \{x \in R \mid$ "x is much smaller than 10"$\}$.

(The symbol $\wedge$ stands for logical AND, the operation of *intersection*.)

3. By a *membership function* (in crisp set theory also called a *characteristic*). For crisp sets (see fig. 6.2, left graph),

$$\mu_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}.$$

For fuzzy sets (see fig. 6.2, right graph), $\mu_S(x)$ is a mapping of X on $[0, 1]$, that is, the degree of belonging of some element x to the universe X can be any number $0 \leq \mu_S(x) \leq 1$.

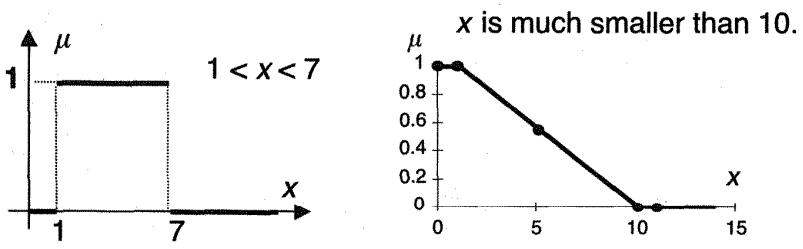


Figure 6.2
Membership functions of (*left*) crisp and (*right*) fuzzy sets.

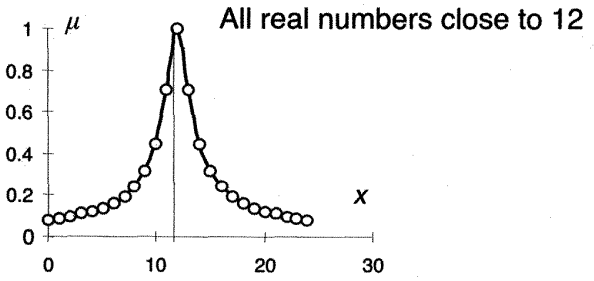


Figure 6.3
Membership function of fuzzy set S "all real numbers close to 12".

In engineering applications, the universe of discourse U stands for the domain of (linguistic) input and output variables, i.e., for antecedent and consequent variables, or for the IF and the THEN variables of the rule. Membership functions (possibility distributions, degrees of belonging) of two typical fuzzy sets are represented in figure 6.2, right graph, and in figure 6.3. The latter shows a fuzzy set S of "all real numbers close to 12":

$$S = \{(x, \mu_S(x)) \,|\, \mu_S(x) = (1 + (x - 12)^2)^{-1}\}.$$

Note the similarities between the two membership functions and sigmoidal and radial basis activation functions given in previous chapters.

In human thinking, it is somehow natural that the maximal degree of belonging to some set cannot be higher than 1. Related to this is a definition of *normal* and *not-normal* fuzzy sets. Both sets are shown in figure 6.4. Typically, the fuzzy set of input variables (the IF variables of IF-THEN rules) is a normal set, and the fuzzy set of output variables (the THEN variables of IF-THEN rules) is a not-normal fuzzy set.

There is an important difference between crisp sets and fuzzy sets (see table 6.1). Fuzzy logic is a tool for modeling human knowledge, or human understanding and concepts about the world. But the world is not binary: there is an infinite number of numbers between 0 and 1; outside of Hollywood movies, people are not divided into

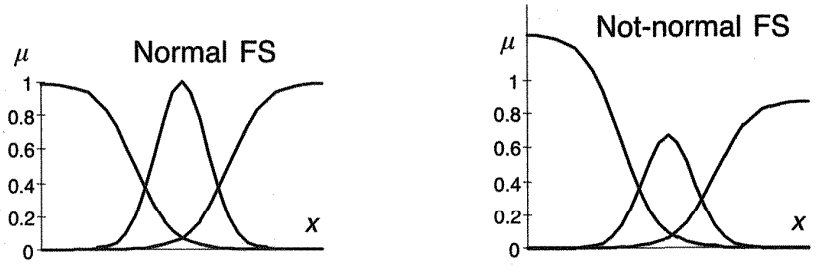


Figure 6.4
Membership functions of normal and not-normal fuzzy sets.

Table 6.1
Differences Between Crisp Sets and Fuzzy Sets

Crisp Sets	Fuzzy Sets
either or	and
bivalent	multivalent
yes or no	more or less

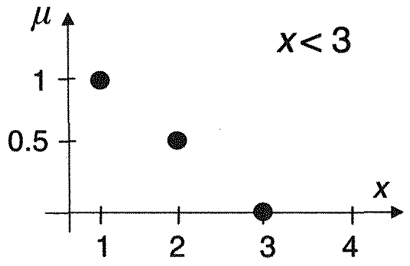


Figure 6.5
Membership functions of the set "x smaller than 3" as discrete pairs μ/x.

only *good* and *bad*; there is a spectrum of colors between *black* and *white*; we are usually not *absolutely healthy* or *terminally ill;* our statements are not *utterly false* or *absolutely true*. Thus, binary concepts like *yes-no* or 0-1, as well as the very wording while dealing with such graded concepts, should be extended to cover a myriad of vague states, concepts, and situations.

In fuzzy logic an *element* can be a member of two or more sets at the same time. Element x belongs to A AND to B, not only to one of these two sets. The very same x is just *more* or *less* a member of A and/or B. See table 6.1

Another notation for *finite* fuzzy sets (sets comprising a finite number of elements) is when a set S is given as a set of pairs μ/x (see fig. 6.5). Note that μ is a function of x

$$\mu = \mu(x)\colon S = \{\mu/x \,|\, x < 3\}, \quad \text{e.g.,} \quad S = \{(1/1), (0.5/2), (0/3)\}.$$

Usually, human reasoning is very approximate. Our statements depend on the contents, and we describe our physical and spiritual world in a rather vague terms. Imprecisely defined "classes" are an important part of human thinking. Let us illustrate this characteristic feature of human reasoning with two more real-life examples that partly describe the subjectivity with which we conceive the world.

The modeling of the concept "young man" is both imprecise and subjective. Three different membership functions of this fuzzy set, or class, depending on the person using it, are given in figure 6.6. (Clearly, the two dashed membership functions would be defined by persons who are in their late thirties or in their forties. The author personally prefers a slightly broader membership function, centered at age = 45.) Similarly, the order given in a pub, "Bring me a cold beer, please,"[2] may have different meanings in different parts of the world. It is highly subjective, too. The author's definition of this fuzzy class is shown in figure 6.7. The membership functions may have different shapes. The choice of a shape for each particular linguistic variable (attribute or fuzzy set) is both subjective and problem-dependent. The most common ones in engineering applications are shown in figure 6.8.

Any function $\mu(x) \rightarrow [0, 1]$ describes a membership function associated with some fuzzy set. Which particular membership function is suitable for fuzzy modeling can

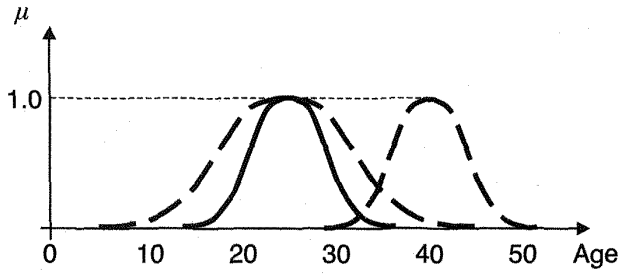


Figure 6.6
Three different membership functions $\mu(x)$ of the class "young man".

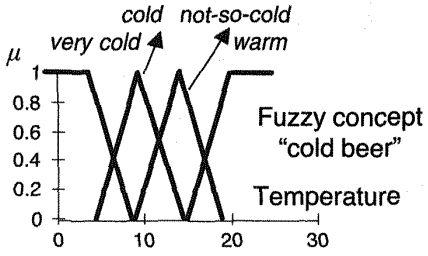


Figure 6.7
Membership functions $\mu(x)$ for S ("cold beer") = {*very cold*, *cold*, *not-so-cold*, *warm*}.

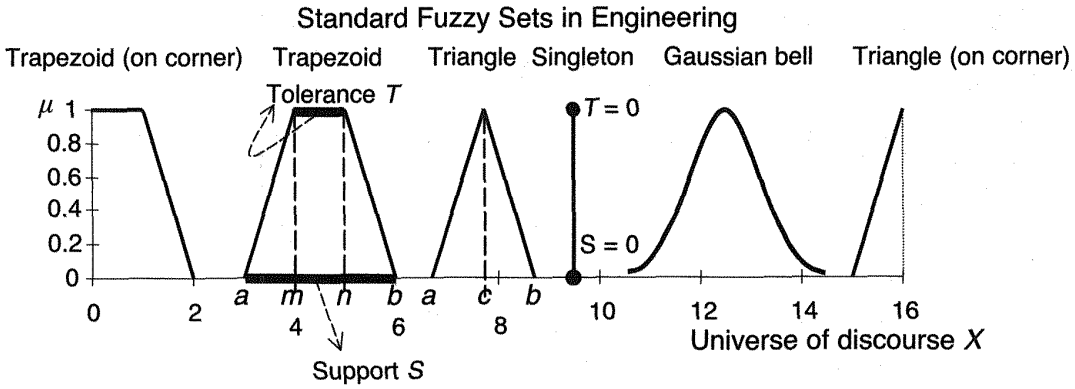


Figure 6.8
The most common fuzzy sets in engineering applications.

be determined in a specific context. Here, the most general triangular and trapezoidal membership functions are defined. Note that all membership functions in figure 6.8 (except the Gaussian one) are specific cases of the following expressions.

Triangular Membership Functions

$$\mu(x) = \begin{cases} 0 & \text{if } x < a \\ \dfrac{x-a}{c-a} & \text{if } x \in [a,c] \\ \dfrac{b-x}{b-c} & \text{if } x \in [c,b] \\ 0 & \text{if } x > b \end{cases}, \quad (6.1a)$$

Trapezoidal Membership Functions

$$\mu(x) = \begin{cases} 0 & \text{if } x < a \\ \dfrac{x-a}{m-a} & \text{if } x \in [a,m] \\ 1 & \text{if } x \in [m,n] \\ \dfrac{b-x}{b-n} & \text{if } x \in [n,b] \\ 0 & \text{if } x > b \end{cases}, \quad (6.1b)$$

where a and b denote lower and upper bounds (i.e., they are "coordinates" of a support S), c is a "center" of a triangle, and m and n denote "coordinates" of a tolerance (see fig. 6.8).

6.1.2 Basic Set Operations

Out of many set operations the three most common and important are *complement* S^C (or *not-S*), *intersection*, and *union*. Figure 6.9 shows the complement S^C of crisp and fuzzy sets in Venn diagrams and using membership functions. Figure 6.10 shows the intersection, union, and complement operations using membership functions. The graphs in figure 6.10 are obtained by using the MIN operator for an intersection (interpreted as logical AND) and the MAX operator for a *union* (interpreted as

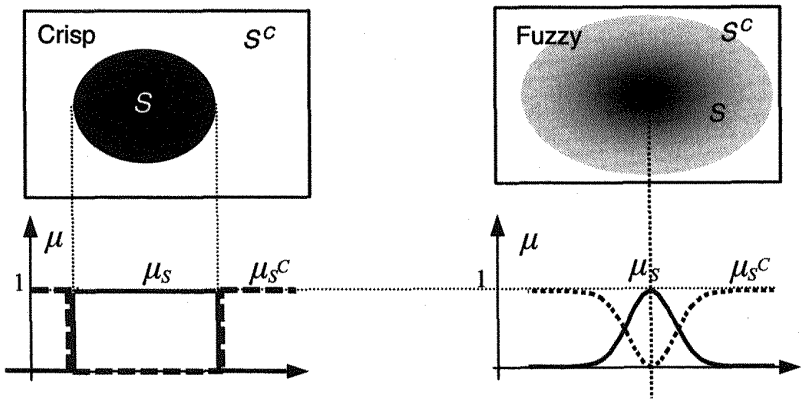


Figure 6.9
Two different ways of representing (*left*) crisp and (*right*) fuzzy sets S and corresponding complement sets S^C. *Top*, Venn diagrams. *Bottom*, membership functions. The brightness of the fuzzy set patch in the right graph denotes the degree of belonging, or membership degree μ, of elements of U to the fuzzy set S (black $-\ \mu = 1$ and white $-\ \mu = 0$). For the complement set, the following is true: $\mu_{S^C} = \mu_{not\text{-}S} = 1 - \mu_S$.

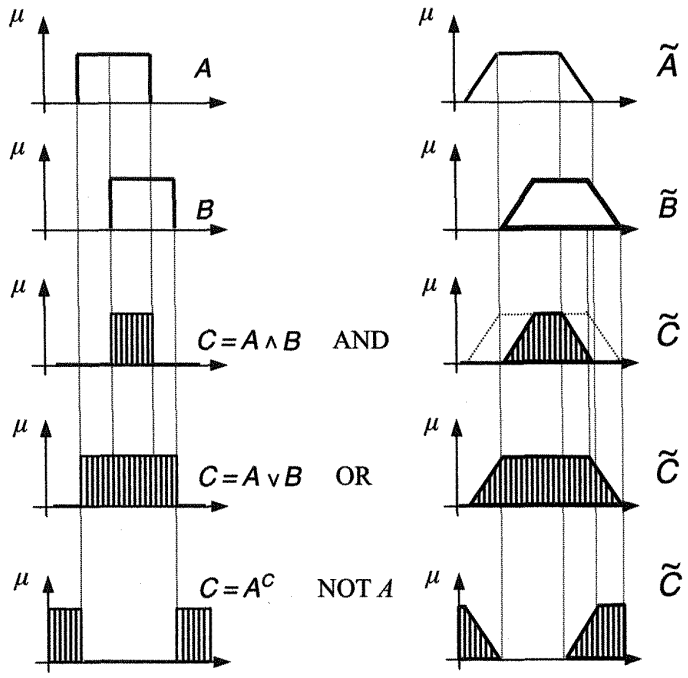


Figure 6.10
Intersection and union as well as complement of A operations for (*left*) crisp and (*right*) fuzzy sets represented by the corresponding membership functions.

logical OR):

$$\mu_{A \wedge B} = \text{MIN}(\mu_A, \mu_B), \tag{6.2}$$

$$\mu_{A \vee B} = \text{MAX}(\mu_A, \mu_B). \tag{6.3}$$

These are not the only operators that can be chosen to model the intersection and union of a fuzzy set, but they are the most commonly used ones in engineering applications. For an intersection, a popular alternative to the MIN operator is the *algebraic product*

$$\mu_{A \wedge B} = \mu_A \cdot \mu_B, \tag{6.4}$$

which typically gives much smoother approximations. In addition to MIN, MAX, and product operators there are many others that can be used. In fuzzy logic theory, intersection operators are called *T-norms*, and union operators are called *T-conorms* or *S-norms*. Table 6.2 lists only some classes of *T-norms* and *S-norms*.

Before closing this section on basic logical operators, it is useful to point out some interesting differences between crisp and fuzzy set calculus. Namely, it is well known that the intersection between a crisp set S and its complement S^C is an empty set, and that the union between these two sets is a universal set. Calculus is different in fuzzy

Table 6.2
T-Norms and *S*-Norms

AND *T*-Norm $T(\mu_A(x), \mu_B(x))$	OR *S*-Norm $S(\mu_A(x), \mu_B(x))$
Minimum $\text{MIN}(\mu_A(x), \mu_B(x))$	Maximum $\text{MAX}(\mu_A(x), \mu_B(x))$
Algebraic product $\mu_A(x)\mu_B(x)$	Algebraic sum $\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$
Drastic product $\text{MIN}(\mu_A(x), \mu_B(x))$ if $\text{MAX}(\mu_A(x), \mu_B(x)) = 1$ 0 otherwise	Drastic sum $\text{MAX}(\mu_A(x), \mu_B(x))$ if $\text{MIN}(\mu_A(x), \mu_B(x)) = 0$ 1 otherwise
Lukasiewicz AND (Bounded Difference) $\text{MAX}(0, \mu_A(x) + \mu_B(x) - 1)$	Lukasiewicz OR (Bounded Sum) $\text{MIN}(1, \mu_A(x) + \mu_B(x))$
Einstein product $\mu_A(x)\mu_B(x)/(2 - (\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)))$	Einstein sum $(\mu_A(x) + \mu_B(x))/(1 + \mu_A(x)\mu_B(x))$
Hamacher product $\mu_A(x)\mu_B(x)/(\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x))$	Hamacher sum $(\mu_A(x) + \mu_B(x) - 2\mu_A(x)\mu_B(x))/(1 - \mu_A(x)\mu_B(x))$
Yager operator $1 - \text{MIN}(1, ((1 - \mu_A(x))^b + (1 - \mu_B(x)^b)^{1/b}))$	Yager operator $\text{MIN}(1, (\mu_A(x)^b + \mu_B(x)^b)^{1/b})$

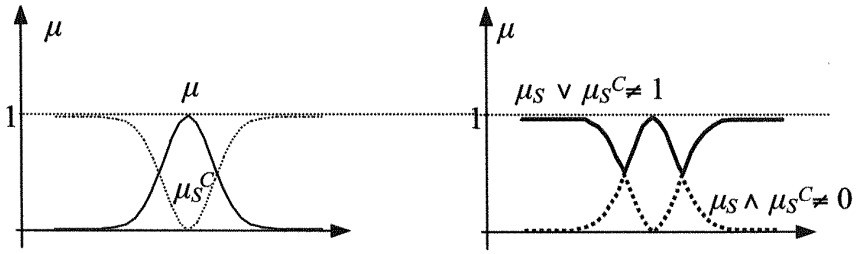


Figure 6.11
Interesting properties of fuzzy set calculus.

logic. Expressed by membership degrees, these facts are as follows:

Crisp Set Calculus		*Fuzzy Set Calculus*	
$\mu \wedge \mu^C = 0.$	$\mu \vee \mu^C = 1.$	$\mu \wedge \mu^C \neq 0.$	$\mu \vee \mu^C \neq 1.$

This can be verified readily for fuzzy sets, as shown in figure 6.11.

6.1.3 Fuzzy Relations

Let us consider the notion of an ordered pair. When making pairs of anything, the order of the elements is usually of great importance (e.g., the points $(2,3)$ and $(3,2)$ in an (x, y) plane are different). A pair of elements that occur in a specified order is called an *ordered pair*. A *relation* is a *set of ordered pairs*.

Relations express connections between different sets. A crisp relation represents the presence or absence of association, interaction, or interconnectedness between the elements of two or more sets (Klir and Folger 1988).

If this concept is generalized, allowing various degrees or strengths of relations between elements, we get fuzzy relations. Because a *relation itself is a set*, all set operations can be applied to it without any modifications. Relations are also subsets of a Cartesian product, or simply of a product set. In other words, relations are defined over *Cartesian products or product sets*.

The *Cartesian product* of two crisp sets X and Y, denoted by $X \times Y$, is the crisp set of all ordered pairs such that the first element in each pair is a member of X and the second element is a member of Y:

$$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}.$$

Let $X = \{1, 2\}$ and $Y = \{a, b, c\}$ be two crisp sets. The Cartesian product is given as,

$$X \times Y = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}.$$

Now, one can choose some subsets at random, or one can choose those that satisfy specific conditions in two variables. In both cases, these subsets are relations. One typically assumes that variables are somehow connected in one relation, but the random choice of, say, three ordered pairs $\{(1,b),(2,a),(2,c)\}$, being a subset of the product set $X \times Y$, is also a relation.

A Cartesian product can be generalized for n sets, in which case elements of the Cartesian product are n-tuples $(x_1, x_2, \ldots, x_n)$. Here, the focus is on relations between two sets, known as a *binary relation* and denoted $R(X, Y)$, or simply R. Thus the binary relation R is defined over a Cartesian product $X \times Y$.

If the elements of the latter come from discrete universes of discourse, this particular relation R can be presented in the form of a *relational matrix* or graphically as a discrete set of points in a three-dimensional space $(X, Y, \mu_R(x, y))$.

Example 6.1 Let X and Y be two sets given as follows. Present the relation R: "x is smaller than y" graphically and in the form of a relational matrix.

$$X = \{1,2,3\}, \qquad Y = \{2,3,4\}, \qquad R\text{: } x < y.$$

$$R = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}.$$

Note that R is a set of pairs and a binary relation. The relational matrix, or membership array, in this crisp case comprises only 1's and 0's. Figure 6.12 shows the discrete membership function $\mu_R(x, y)$ of this relation.

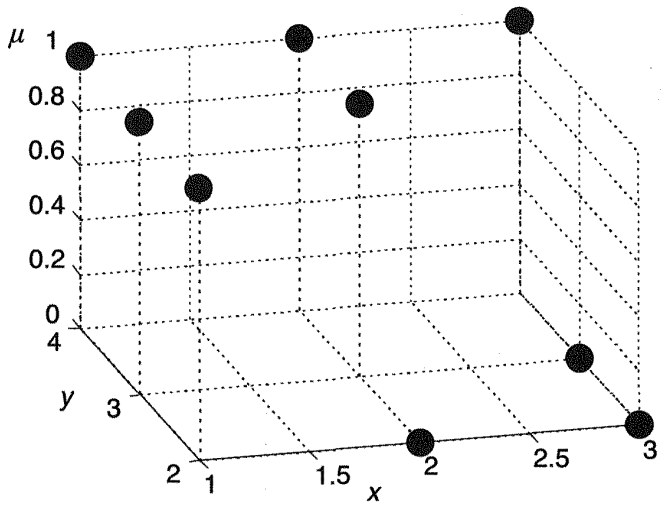


Figure 6.12
The discrete membership function $\mu_R(x,y)$ of the relation R: "x is smaller than y".

$x \setminus y$	2	3	4
1	1	1	1
2	0	1	1
3	0	0	1

■

The elements of the relational matrix are degrees of membership $\mu_R(x, y)$, that is, possibilities, or degrees of belonging, of a specific pair (x, y) to the given relation R. Thus, for example, the pair $(3, 1)$ belongs with a degree 0 to the relation "x is smaller than y", or the possibility that 3 is smaller than 1 is zero. The preceding relation is a typical example of a crisp relation. The condition involved in this relation is precise and one that is either fulfilled or not fulfilled.

The common mathematical expression "x is approximately equal to y", or the relation R: $x \approx y$, is different. It is a typical example of an imprecise, or fuzzy, relation. Example 6.2 is very similar to example 6.1, the difference being that the degree of belonging of some pairs (x, y) from the Cartesian product to this relation can be any number between 0 and 1.

Example 6.2 Let X and Y be two sets given as follows. Present the relation R: "x is approximately equal to y" in the form of a relational matrix.

$X = \{1, 2, 3\}, Y = \{2, 3, 4\}, \qquad R$: $x \approx y$

$R = \{(1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,2), (3,3), (3,4)\}$

x $\setminus$ y	2	3	4
1	0.66	0.33	0
2	1	0.66	0.33
3	0.66	1	0.66

The discrete membership function $\mu_R(x, y)$ is again a set of discrete points in a three-dimensional space $(X, Y, \mu_R(x, y))$ but with membership degrees that can have any value between 0 and 1. ■

When the universes of discourse (domains) are *continuous sets* comprising an infinite number of elements, the membership function $\mu_R(x, y)$ is *a surface* over the Cartesian product $X \times Y$, not a curve as in the case of one-dimensional fuzzy sets.[3] Thus, the relational matrix is an (∞, ∞) matrix and has no practical meaning. This is a common situation in everyday practice, which is resolved by appropriate discretization of the universes of discourse. Example 6.3 illustrates this.

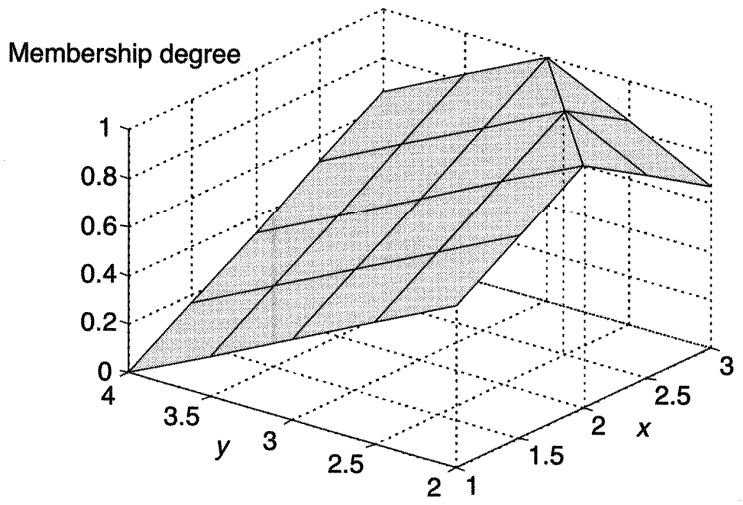


Figure 6.13
Membership function $\mu_R(x,y)$ of the relation R: "x is approximately equal to y" over the Cartesian product of two continuous sets X and Y.

Example 6.3 Let X and Y be two sets given as follows. Show the membership function of the relation R: "x is approximately equal to y" and present the relational matrix after discretization.

$$X = \{x \in R \mid 1 \leq x \leq 3\}, \qquad Y = \{y \in R \mid 2 \leq x \leq 4\}, \qquad R\text{: } x \approx y.$$

Figure 6.13 shows the membership function, and the relational matrix after discretization by a step of 0.5 is

$x \setminus y$	2	2.5	3	3.5	4
1	0.6667	0.5000	0.3333	0.1667	0.0000
1.5	0.8333	0.6667	0.5000	0.3333	0.1667
2	1.0000	0.8333	0.6667	0.5000	0.3333
2.5	0.8333	1.0000	0.8333	0.6667	0.5000
3	0.6667	0.8333	1.0000	0.8333	0.6667

■

In the preceding examples, the sets are defined on the same universes of discourse. But relations can be defined in linguistic variables expressing a variety of different associations or interconnections.

Example 6.4 Relation R is given as an association or interconnection between fruit color and state. Present R as a crisp relational matrix.

$$X = \{green, yellow, red\}, \qquad Y = \{unripe, semiripe, ripe\}.$$

R	*unripe*	*semiripe*	*ripe*
green	1	0	0
yellow	0	1	0
red	0	0	1

■

This relational matrix can be interpreted as a notation, or model, of an existing empirical set of IF-THEN rules:

R_1: IF (the tomato is) *green*, THEN (it is) *unripe*.

R_2: IF *yellow*, THEN *semiripe*.

R_3: IF *red*, THEN *ripe*.

In fact, relations are a convenient tool for modeling IF-THEN rules. However, the relational matrix in example 6.4 is a crisp one and not in total agreement with our experience. A better interconnection between fruit color and state may be given by the following fuzzy relational matrix:

R	*unripe*	*semiripe*	*ripe*
green	1	0.5	0
yellow	0.3	1	0.4
red	0	0.2	1

Example 6.5 Present the fuzzy relational matrix for the relation R that represents the concept "very far" in geography. Two crisp sets are given as

$X = \{\text{Auckland, Tokyo, Belgrade}\}$.

$Y = \{\text{Sydney, Athens, Belgrade, Paris, New York}\}$.

R: "very far"	*Sydney*	*Athens*	*Belgrade*	*Paris*	*New York*
Auckland	0.2	0.8	0.85	0.90	0.55
Tokyo	0.5	0.5	0.5	0.55	0.4
Belgrade	0.8	0.1	0	0.15	0.5

■

Hence, the relational matrix does not necessarily have to be square. Many other concepts can be modeled using relations on different universes of discourse. Note that, as for crisp relations, fuzzy relations are fuzzy sets in product spaces. As an example, let us analyze the meaning of the linguistic expression "*a young tall man*" (Kahlert and Frank 1994).

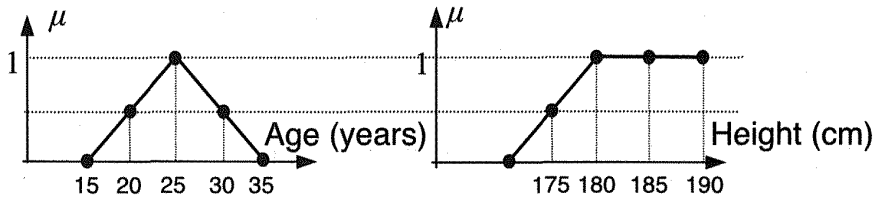


Figure 6.14
The fuzzy sets, or linguistic terms, "young man" and "tall man" given by corresponding membership functions.

Example 6.6 Find the relational matrix of the concept "a young tall man".

Implicitly, the concept "a young tall man" means "young AND tall man". Therefore, two fuzzy sets, "young man" and "tall man", are defined first, and then the intersection operator is applied to these two sets defined on different universes of discourse, age and height. One out of many operators for modeling a fuzzy intersection is the MIN operator. (Another commonly used one is the algebraic product.) Thus,

$\mu_R(\text{age, height}) = \text{MIN}(\mu_1(\text{age}), \mu_2(\text{height}))$.

After discretization, as in figure 6.14,

$(S_1 = \{15, 20, 25, 30, 35\}, \qquad S_2 = \{170, 175, 180, 185, 190\},$

the relational matrix follows from

$$\mu_1 = \begin{bmatrix} 0 \\ 0.5 \\ 1 \\ 0.5 \\ 0 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 0 \\ 0.5 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{R} = \mu_1 \times \mu_2^T = \begin{bmatrix} 0 \\ 0.5 \\ 1 \\ 0.5 \\ 0 \end{bmatrix} \times [0 \quad 0.5 \quad 1 \quad 1 \quad 1], \qquad (6.5)$$

or

R	170	175	180	185	190
15	0	0	0	0	0
20	0	0.5	0.5	0.5	0.5
25	0	0.5	1	1	1
30	0	0.5	0.5	0.5	0.5
35	0	0	0	0	0

■

The relational matrix in example 6.6 is actually a surface over the Cartesian product age × height, which represents the membership function, or a possibility distribution of a given relation. Generally, one can graphically obtain this surface utilizing the extension principle given the different universes of discourse (cylindrical extension, in particular). However, this part of fuzzy theory is outside the scope of this book.

6.1.4 Composition of Fuzzy Relations

Fuzzy relations in different product spaces can be combined with each other by *composition*. Note that fuzzy sets can also be combined with any fuzzy relation in the same way. A composition is also a fuzzy set because relations are the fuzzy sets. (This is the same attribute as the product of matrices being a matrix.)

Many different versions of the compositional operator are possible. The best known one is a MAX-MIN composition. MAX-PROD and MAX-AVERAGE can also be used. The MAX-PROD composition is often the best alternative. A discussion of these three most important compositions follows.

Let $R_1(x, y), (x, y) \in X \times Y$ and $R_2(y, z), (y, z) \in Y \times Z$ be two fuzzy relations. The MAX-MIN composition is a fuzzy set

$$R_1 \circ R_2(x, z) = \left\{ \left[(x, z), \max_y \{\min\{\mu_{R_1}(x, y), \mu_{R_2}(y, z)\}\} \right] \middle| x \in X, y \in Y, z \in Z \right\}, \tag{6.6}$$

and $\mu_{R_1 \circ R_2}$ is a membership function of a fuzzy composition on fuzzy sets.

The MAX-PROD composition is

$$R_1 \underset{\otimes}{\circ} R_2(x, z) = \left\{ \left[(x, z), \max_y \{\mu_{R_1}(x, y) \cdot \mu_{R_2}(y, z)\} \right] \middle| x \in X, y \in Y, z \in Z \right\}. \tag{6.7}$$

The MAX-AVE composition is

$$R_1 \underset{\text{ave}}{\circ} R_2(x, z) = \left\{ \left[(x, z), \frac{1}{2} \max\{\mu_{R_1}(x, y) + \mu_{R_2}(y, z)\} \right] \middle| x \in X, y \in Y, z \in Z \right\}. \tag{6.8}$$

Later, while making a *fuzzy inference*, a composition of a fuzzy set and a fuzzy relation (and not one between two fuzzy relations) will be of practical importance. Example 6.7 will facilitate the understanding of the three preceding compositions.

Example 6.7 R_1 is a relation that describes an interconnection between color x and ripeness y of a tomato, and R_2 represents an interconnection between ripeness y and

taste z of a tomato (Kahlert and Frank 1994). Present relational matrices for the MAX-MIN and MAX-PROD compositions.

The relational matrix $\mathbf{R}_1$ (x–y connection) is given as

$\mathbf{R}_1(x, y)$	*unripe*	*semiripe*	*ripe*
green	1	0.5	0
yellow	0.3	1	0.4
red	0	0.2	1

The relational matrix $\mathbf{R}_2$ (y–z connection) is given as

$\mathbf{R}_2(y, z)$	*sour*	*sweet-sour*	*sweet*
unripe	1	0.2	0
semiripe	0.7	1	0.3
ripe	0	0.7	1

The MAX-MIN composition $\mathbf{R} = \mathbf{R}_1 \circ \mathbf{R}_2$ results in the relational matrix

$\mathbf{R}(x, z)$	*sour*	*sweet-sour*	*sweet*
green	1	0.5	0.3
yellow	0.7	1	0.4
red	0.2	0.7	1

The entries of the relational matrix $\mathbf{R}$ were calculated as follows:

$$r_{11} = \text{MAX}(\text{MIN}(1, 1), \text{MIN}(0.5, 0.7), \text{MIN}(0, 0)) = \text{MAX}(1, 0.5, 0) = 1,$$

$$r_{23} = \text{MAX}(\text{MIN}(0.3, 0), \text{MIN}(1, 0.3), \text{MIN}(0.4, 1)) = \text{MAX}(0, 0.3, 0.4) = 0.4.$$

The MAX-PROD composition will give a slightly different relational matrix:

$$\mathbf{R} = \mathbf{R}_1 \underset{\otimes}{\circ} \mathbf{R}_2$$

$$= \text{MAX} \begin{bmatrix} (1 \cdot 1, 0.5 \cdot 0.7, 0 \cdot 0) & (1 \cdot 0.2, 0.5 \cdot 1, 0 \cdot 0.7) & (1 \cdot 0, 0.5 \cdot 0.3, 0 \cdot 1) \\ (0.3 \cdot 1, 1 \cdot 0.7, 0.4 \cdot 0) & (0.3 \cdot 0.2, 1 \cdot 1, 0.4 \cdot 0.7) & (0.3 \cdot 0, 1 \cdot 0.3, 0.4 \cdot 1) \\ (0 \cdot 1, 0.2 \cdot 0.7, 1 \cdot 0) & (0 \cdot 0.2, 0.2 \cdot 1, 1 \cdot 0.7) & (0 \cdot 0, 0.2 \cdot 0.3, 1 \cdot 1) \end{bmatrix}$$

$$= \text{MAX} \begin{bmatrix} (1, 0.35, 0) & (0.2, 0.5, 0) & (0, 0.15, 0) \\ (0.3, 0.7, 0) & (0.06, 1, 0.28) & (0, 0.3, 0.4) \\ (0, 0.14, 0) & (0, 0.2, 0.7) & (0, 0.06, 1) \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.15 \\ 0.7 & 1 & 0.4 \\ 0.14 & 0.7 & 1 \end{bmatrix} \quad (6.9)$$

Note that the resulting MAX-MIN and MAX-PROD relational matrices differ a little only in two elements; r_{13} and r_{31}. It is also interesting to compare the result of the MAX-PROD composition with the classical multiplication of the two matrices $\mathbf{R}_1 \times \mathbf{R}_2$. Recall that in standard matrix multiplication, the SUM operator is used instead of the MAX operator, after the multiplication of the specific elements in corresponding rows and columns. Thus, matrix multiplication would result in

$$\mathbf{R}_1 \times \mathbf{R}_2$$

$$= \begin{bmatrix} (1\cdot 1+0.5\cdot 0.7+0\cdot 0) & (1\cdot 0.2+0.5\cdot 1+0\cdot 0.7) & (1\cdot 0+0.5\cdot 0.3+0\cdot 1) \\ (0.3\cdot 1+1\cdot 0.7+0.4\cdot 0) & (0.3\cdot 0.2+1\cdot 1+0.4\cdot 0.7) & (0.3\cdot 0+1\cdot 0.3+0.4\cdot 1) \\ (0\cdot 1+0.2\cdot 0.7+1\cdot 0) & (0\cdot 0.2+0.2\cdot 1+1\cdot 0.7) & (0\cdot 0+0.2\cdot 0.3+1\cdot 1) \end{bmatrix}$$

$$= \begin{bmatrix} 1.35 & 0.7 & 0.15 \\ 1 & 1.34 & 0.7 \\ 0.14 & 0.9 & 1.06 \end{bmatrix}$$

The linguistic interpretation of the resulting relational matrix $\mathbf{R}$ is a straightforward one, corresponding to our experience, and can be given in the form of IF-THEN rules. This example clearly shows that fuzzy relations are a suitable means of expressing fuzzy (uncertain, vague) implications. A linguistic interpretation in the form of rules for the relational matrices (6.9) is as follows:

R_1: IF the tomato is *green*, THEN it is *sour*, less likely to be *sweet-sour*, and unlikely to be *sweet*.

R_2: IF the tomato is *yellow*, THEN it is *sweet-sour*, possibly *sour*, and unlikely to be *sweet*.

R_3: IF the tomato is *red*, THEN it is *sweet*, possibly *sweet-sour*, and unlikely to be *sour*.

The fuzzy sets (also known as attributes or linguistic variables) are shown in *italics*. Note the multivalued characteristic of the fuzzy implications. Compare the crisp relational matrix in example 6.4 with the $\mathbf{R}_1$ given here, and compare their corresponding crisp and fuzzy implications. ■

6.1.5 Fuzzy Inference

In classical propositional calculus there are two basic inference rules: the *modus ponens* and the *modus tollens*. Modus ponens is associated with the implication "*A*

implies B" or "B follows from A", and it is the more important one for engineering applications.

Modus ponens can typically be represented by the following inference scheme:

Fact or premise	"x is A"
Implication	"IF x is A, THEN y is B"
Consequence or conclusion	"y is B"

In modus tollens inference, the roles are interchanged:

Fact or premise	"y is not B"
Implication	"IF x is A, THEN y is B"
Consequence or conclusion	"x is not A"

The modus ponens from standard logical propositional calculus cannot be used in the fuzzy logic environment because such an inference can take place if, and only if, the fact or premise is exactly the same as the antecedent of the IF-THEN rule. In fuzzy logic the *generalized modus ponens* is used. It allows an inference when the antecedent is only partly known or when the fact is only similar but not equal to it. A typical problem in fuzzy approximate reasoning is as follows:

Implication	IF the tomato is *red*, THEN it is *sweet*, possibly *sweet-sour*, and unlikely to be *sour*.
Premise or fact	The tomato is *more* or *less red* ($\mu_{\mathrm{Re}\,d} = 0.8$).
Conclusion	Taste = ?

The question now is, Having a state of nature (premise, fact) that is not exactly equal to the antecedent, and the IF-THEN rule (implication), what is the conclusion?

In traditional logic (classical propositional calculus, conditional statements) an expression such as "IF A, THEN B" is written as $A \Rightarrow B$, that is, A implies B, or B follows from A. Such an *implication* is defined by the following truth table:

A	B	$A \Rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

The following identity is used in calculating the truth table:

$$A \Rightarrow B \equiv A^C \vee B.$$

Note the "strangeness" of the last two rows. Conditional statements, or implications, sound paradoxical when the components are not related. In everyday human reasoning, implications are given to combine somehow related statements, but in the use of the conditional in classical two-valued logic, there is no requirement for relatedness. Thus, "unusual" but correct results could be produced using the preceding operator. Example 6.8 illustrates this curious character of standard Boolean logic.

Example 6.8 The statement "IF $2 \times 2 = 5$, THEN cows are horses" is *true* (row 4 in the truth table), but "IF $2 \times 2 = 4$, THEN cows are horses" is *false* (row 2 in the truth table). ■

In Boolean logic there does not have to be any real causality between the antecedent (IF part) and the consequent (THEN part). It is different in human reasoning. Our rules express cause-effect relations, and fuzzy logic is a tool for transferring such structured knowledge into workable algorithms. Thus, *fuzzy logic cannot be and is not Boolean logic*. It must go beyond crisp logic. This is because in engineering and many other fields, there is no effect (output) without a cause (input).

Therefore, which operator is to be used for fuzzy conditional statements (implications) or for fuzzy IF-THEN rules? In order to find an answer to this question, consider what the result would be of everyday (fuzzy) reasoning if the *crisp implication* algorithm were used. Starting with the crisp implication rule

$$A \Rightarrow B \equiv A^C \vee B, \qquad A^C = 1 - \mu_A(x),$$

and

$$A \vee B = \text{MAX}(\mu_A(x), \mu_B(y)) \quad \text{(fuzzy OR operator)},$$

the *fuzzy implication* would be

$$A \Rightarrow B \equiv A^C \vee B = \text{MAX}(1 - \mu_A(x), \mu_B(y)).$$

This result is definitely not an acceptable one for the related fuzzy sets that are subjects of everyday human reasoning because in the cases when the premise is not fulfilled ($\mu_A(x) = 0$), the result would be the truth value of the conclusion $\mu_B(y) = 1$. This doesn't make much sense in practice, where a system input (cause) produces a system output (effect). Or, in other words, if there is no cause, there will be no effect. Thus, for $\mu_A(x) = 0$, $\mu_B(y)$ must be equal to zero. For *fuzzy implication*, the implication rule states that *the truth value of the conclusion must not be larger than that of the premise*.

There are many different ways to find the truth value of a premise or to calculate the relational matrix that describes a given implication. The minimum and product implications are the two most widely used today. (They were used by Mamdani and Larsen, respectively).

$$\mu_{A\Rightarrow B}(x, y) = \text{MIN}(\mu_A(x), \mu_B(y)) \quad \text{(Mamdani)}, \tag{6.10}$$

$$\mu_{A\Rightarrow B}(x, y) = \mu_A(x)\mu_B(y) \quad \text{(Larsen)}. \tag{6.11}$$

6.1.6 Zadeh's Compositional Rule of Inference

If R is a fuzzy relation from the universe of discourse X to the universe of discourse Y, and x is a fuzzy subset of X, then the fuzzy subset y of Y, which is induced by x, is given by the *composition*

$$y = x \circ R.$$

As mentioned earlier, the operator of this composition is MAX-MIN, with alternatives MAX-PROD or MAX-AVE.

Example 6.9 Show a compositional rule of inference using the MAX-MIN operator. R represents the relation between color x and taste z of a tomato, as given in example 6.7, and the state of nature (premise, fact, or input x) is

The tomato is *red.*

First, this premise should be expressed as the input vector **x**. Note that X has three possible linguistic values: *green, yellow*, and *red.* Thus, the fact that the tomato is red is expressed by the vector $\mathbf{x} = [0 \quad 0 \quad 1]$. This is a *fuzzification step*, which transforms a crisp value into a vector of membership degrees.

Premise or fact: $\mathbf{x} = [0 \quad 0 \quad 1]$.

Implication **R**:

$x \setminus z$	*sour*	*sweet-sour*	*sweet*
green	1	0.5	0.3
yellow	0.7	1	0.4
red	0.2	0.7	1

The linguistic interpretation of this implication (or of this relational matrix) is given in the form of IF-THEN rules in example 6.7.

The conclusion is a result of the following composition (m denotes a MIN operator):

$$y = \mathbf{x} \circ \mathbf{R} = [0 \quad 0 \quad 1] \circ \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.7 & 1 & 0.4 \\ 0.2 & 0.7 & 1 \end{bmatrix}$$

$$= \text{MAX}[m(0,1), m(0,0.7), m(1,0.2), m(0,0.5), m(0,1),$$
$$m(1,0.7), m(0,0.3), m(0,0.4), m(1,1)]$$
$$= [0.2 \quad 0.7 \quad 1]. \tag{6.12}$$

■

Example 6.9 showed a composition between $\mathbf{x}$ and a given relational matrix $\mathbf{R}$. Thus, when modeling structured human knowledge, the IF-THEN rules (in their most common form for expressing this knowledge) *should first be transformed into relational matrices*. Only after the appropriate relational matrix $\mathbf{R}$ of the rules is calculated can a fuzzy inference take place. How to find this relational matrix $\mathbf{R}$ of the IF-THEN rules is shown in example 6.10. In section 6.3, however, the FAMs that are introduced do not use relational matrices in modeling human knowledge at all. They are closer to the ways in which neural networks model data.

Example 6.10 Find the relational matrix $\mathbf{R}$ of the following rule (implication);

R: IF $x = small$, THEN $y = high$.

First, the fuzzy sets *low* and *high* should be defined. These are shown in figure 6.15 by their membership functions. In order to obtain a matrix $\mathbf{R}$ of finite dimension, each membership function must be discretized. The discrete points shown in figure 6.15 (but not the straight lines of the triangles) now represent the fuzzy sets *low* and

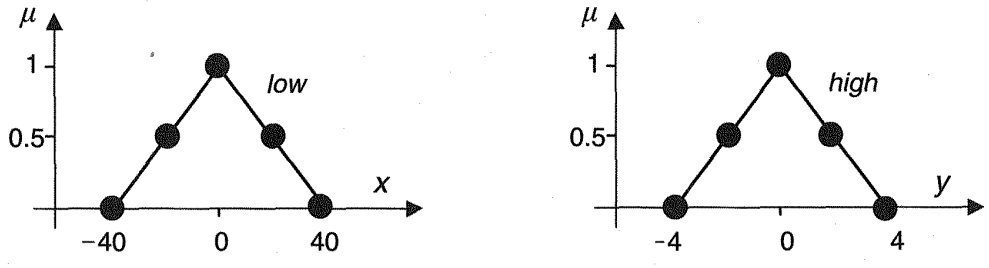


Figure 6.15
Fuzzy sets, or linguistic terms, "low" and "high" given by corresponding membership functions in different universes of discourse.

high (see fig. 6.5). Thus, the universes of discourse X and Y now have five (or a finite number of) elements each:

$$X = \{-40, -20, 0, 20, 40\}, \qquad Y = \{-4, -2, 0, 2, 4\}.$$

In order to calculate the entries of the relational matrix $\mathbf{R}$, recall that *the truth value of the conclusion must be* smaller than, or equal to, *the truth value of the premise*. This is ensured by using the MIN or PROD operator, for example. The result obtained by the MIN operator (the Mamdani implication) is

$$\mu_R(x, y) = \text{MIN}(\mu_L(x), \mu_H(y)). \tag{6.13}$$

The relational matrix $\mathbf{R}$ can be calculated by a vector product, using the same procedure as in example 6.6:

$$\mathbf{R} = \text{MIN}\{\mu_L(x)^T \mu_H(y)\} = \text{MIN}\{[0 \quad 0.5 \quad 1 \quad 0.5 \quad 0]^T [0 \quad 0.5 \quad 1 \quad 0.5 \quad 0]\}.$$

For example, for $x = -20$ and $y = 0$, the membership degree of the relational matrix will be

$$\mu_R(x = -20, y = 0) = \text{MIN}\{\mu_L(-20)\mu_H(0)\} = \text{MIN}\{0.5, 1\} = 0.5.$$

The whole $\mathbf{R}$ is

$\mathbf{R}$	−40	−20	0	20	40
−40	0	0	0	0	0
−20	0	0.5	0.5	0.5	0
0	0	0.5	1	0.5	0
20	0	0.5	0.5	0.5	0
40	0	0	0	0	0

The fuzzy inference for $x' = -20$ (the framed row of $\mathbf{R}$) is the result of the following composition:

$$\mu_{L' \circ R}(y) = \mu_{H'}(y) = \underset{x \in X}{\text{MAX}}\ \text{MIN}(\mu_{L'}(x), \mu_R(x, y)), \tag{6.14}$$

$$\mu_{L' \circ R}(y) = \mu_{H'}(y) = [0 \quad 1 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= [0 \quad 0.5 \quad 0.5 \quad 0.5 \quad 0].$$

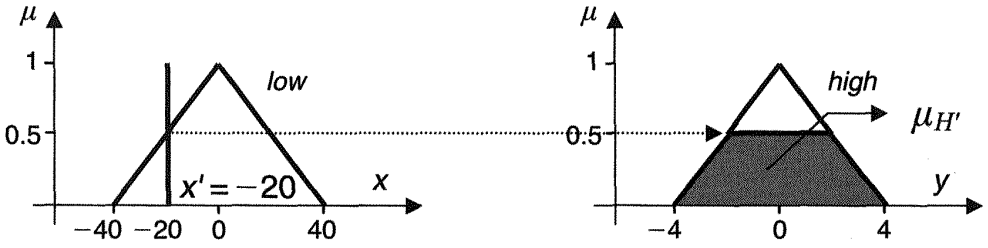


Figure 6.16
MAX-MIN fuzzy inference. The conclusion is not a crisp value but a not-normal fuzzy set.

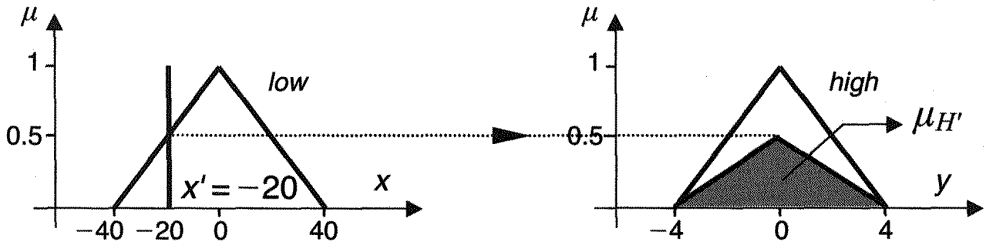


Figure 6.17
MAX-PROD fuzzy inference. The conclusion is not a crisp value but a not-normal fuzzy set.

Note that the crisp value $x' = -20$ was *fuzzified*, or *transformed into a membership vector* $\mu_{L'} = [0 \quad 1 \quad 0 \quad 0 \quad 0]$, first. This is because x is a singleton at x' (see fig. 6.16).

Another popular fuzzy inference scheme employs MAX-PROD (the Larsen implication), which typically results in a smoother model. The graphical result of the MAX-PROD inference is given in figure 6.17, and the relational matrix **R** is as follows:

R	−40	−20	0	20	40
−40	0	0	0	0	0
−20	0	0.25	0.5	0.25	0
0	0	0.5	1	0.5	0
20	0	0.25	0.5	0.25	0
40	0	0	0	0	0

■

Typical real-life problems have more input variables, and the corresponding rules are given in the form of a rule table:

R_1: IF $x_1 = low$ AND $x_2 = medium$, THEN $y = high$.

R_2: IF $x_1 = low$ AND $x_2 = high$, THEN $y = very\ high$.

⋮

Now, the rules R are three-tuple fuzzy relations having membership functions that are hypersurfaces over the three-dimensional space spanned by x_1, x_2, and y. For instance, for rule R_1,

$$\mu_R(x_1, x_2, y) = \text{MIN}(\mu_L(x_1), \mu_M(x_2), \mu_H(y)). \tag{6.15}$$

The relational matrix has a third dimension now. It is a cubic array. This is studied in more detail later but is illustrated in example 6.11 with two inputs and one output.

Example 6.11 Find the consequent of the rule R_1. The membership functions of two fuzzy sets "low" and "medium" are shown in figure 6.18, and rule R_1 is

R_1: IF $x_1 = low$ AND $x_2 = medium$, THEN $y = high$.

Figure 6.18 shows the results of a fuzzy inference for the two crisp values $x_1' = 2$ and $x_2' = 600$.[4] The objective is to find the output for the two given input values, or $y(x_1' = 2, x_2' = 600) = ?$ At this point, nothing can be said about the crisp value of y. A part of the tool, the *defuzzification method*, is missing at the moment. It is discussed in section 6.1.7. But the consequent of rule R_1 can be found. First note that the antecedents 1 and 2 (*small* and *medium*) are connected with an AND operator, meaning that the fulfillment degree of rule R_1 will be calculated using a MIN operator, $H = \text{MIN}(\mu_L(2), \mu_M(600)) = 0.5$. Thus, the resulting consequent is a not-normal fuzzy set μ_H', as shown in figure 6.18.

In actual engineering problems there are typically more input variables and fuzzy sets (linguistic terms) for each variable. In such a situation, there are $N_R = n_{FS1} \times n_{FS2} \times \cdots \times n_{FSu}$ rules, where n_{FSi} represents the number of fuzzy sets for the ith input variable x_i, and u is the number of input variables. For example, when there are three ($u = 3$) inputs with two, three, and five fuzzy sets respectively, there are $N_R = 2 \times 3 \times 5 = 30$ rules. During the operation of the fuzzy model, more rules are generally active simultaneously. It is important to note that all the rules make a *union* of rules. In other words, the rules are implicitly connected by an OR operator.

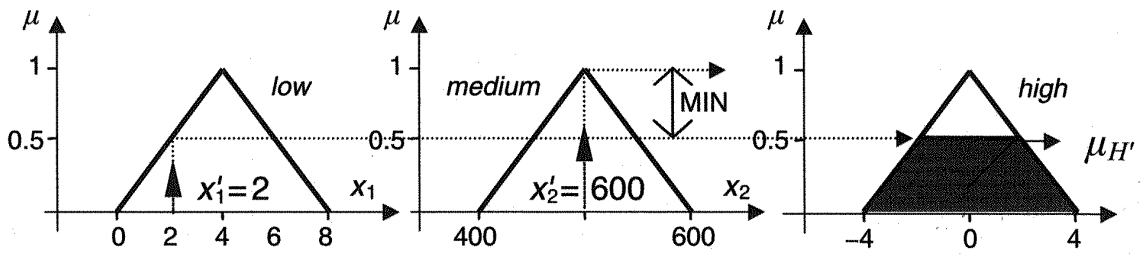


Figure 6.18
Construction of the consequent membership function $\mu_{H'}$ for the rule R_1.

Example 6.12 shows a fuzzy inference in a simple case with one input and one output (an $\mathfrak{R}^1 \rightarrow \mathfrak{R}^1$ mapping). A slightly more complex generic situation with two inputs and one output (an $\mathfrak{R}^2 \rightarrow \mathfrak{R}^1$ mapping) pertains in example 6.13.

Example 6.12 For $x' = 20$, find the output fuzzy set of the single-input, single-output system described by the two following rules:

R_1: IF $x = S$, THEN $y = H$.

R_2: IF $x = M$, THEN $y = M$.

Figure 6.19 illustrates the consequent of these rules, and the equations are as follows:

Rule R_1:

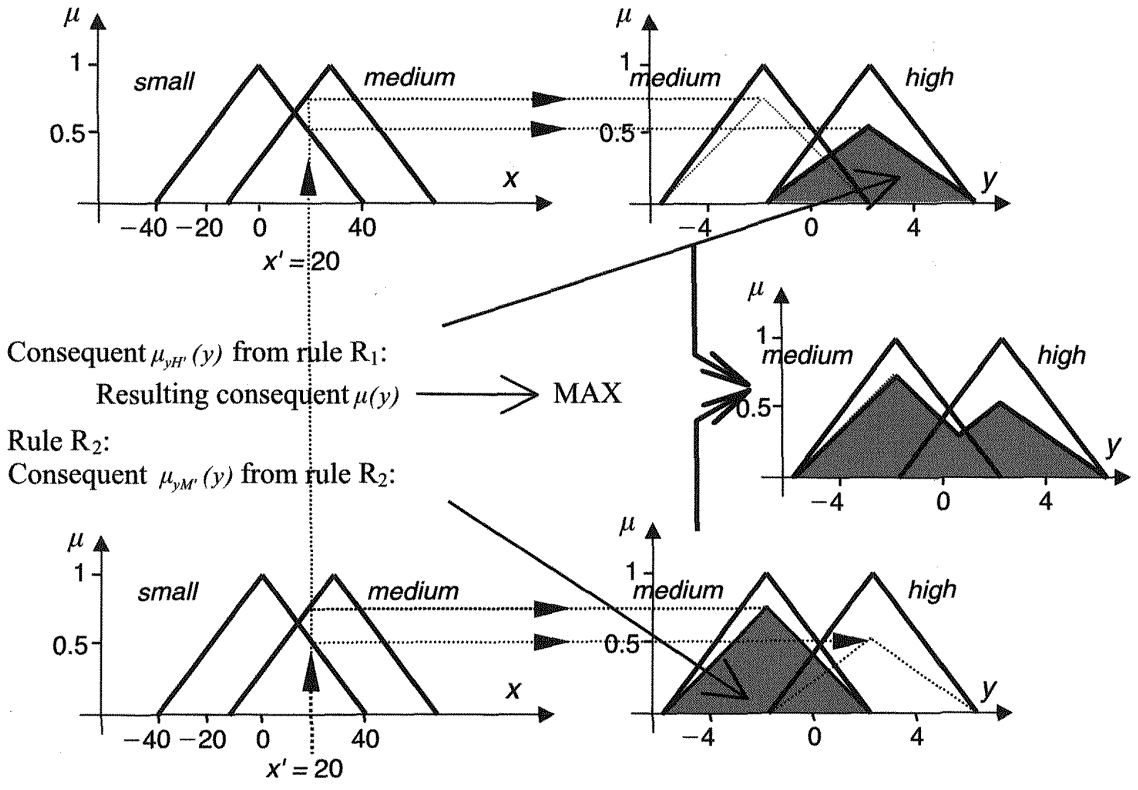


Figure 6.19
Construction of the consequent membership function from the two active rules for a single-input, single-output system (MAX-PROD inference).

$$y(x' = 20) = ? \qquad \mu_x = S(20) = 0.5, \qquad \mu_x = M(20) = 0.75,$$

$$\mu(y) = \text{MAX}(\text{PROD}(0.5, \mu_{y=H}(y), \text{PROD}(0.75, \mu_{y=M}(y))$$

$$= \text{MAX}(\mu_{yH'}(y), \mu_{yM'}(y)). \qquad \blacksquare$$

Note that the result of this fuzzy inference is a *not-normal* fuzzy set. In real-life problems, one is more interested in the single crisp value of the output variable. How to find this crisp value y is discussed in section 6.1.7.

Example 6.13 Find the output fuzzy set for a system with two inputs (having two fuzzy sets for each input) and one output, described by the following four rules:

R_1: IF $x_1 = low$ AND $x_2 = low$, THEN $y = low$.

R_2: IF $x_1 = low$ OR $x_2 = high$, THEN $y = medium$.

R_3: IF $x_1 = zero$ AND $x_2 = low$, THEN $y = medium$.

R_4: IF $x_1 = zero$ OR $x_2 = high$, THEN $y = high$.

The output fuzzy set is shown in figure 6.20.

Finally, how one finds the crisp output value y from the resulting not-normal sets $\mu(y)$, or how one can *defuzzify* $\mu(y)$, will be introduced below.

6.1.7 Defuzzification

In the last few examples, the conclusions happened to be not-normal fuzzy sets. For practical purposes, a crisp output signal to the actuator or decision maker (classifier) is needed. The procedure for obtaining a crisp output value from the resulting fuzzy set is called *defuzzification*. Note the subtle difference between fuzzification (as in examples 6.9 and 6.10) and defuzzification: *Fuzzification* represents the transformation of a crisp input into a vector of membership degrees, and *defuzzification* transforms a (typically not-normal) fuzzy set into a crisp value.

Which method is to be used to find the crisp output value? Several methods are in use today. Here the four most popular are presented. It may be useful first to get an intuitive insight into defuzzification. What would be a crisp output value for the resulting not-normal fuzzy set in example 6.13? Just by observing the geometry of the resulting fuzzy set (fig. 6.21) one could conclude that the resulting output value y might be between $y = 50$ and 80 and that the right value could be $y = 58$. At this value is actually the *center-of-area* (or *center-of-gravity*) of the resulting consequent from the four rules given in example 6.13. This is one of the many methods of defuzzification. Figure 6.22 shows three of the most common defuzzification methods.

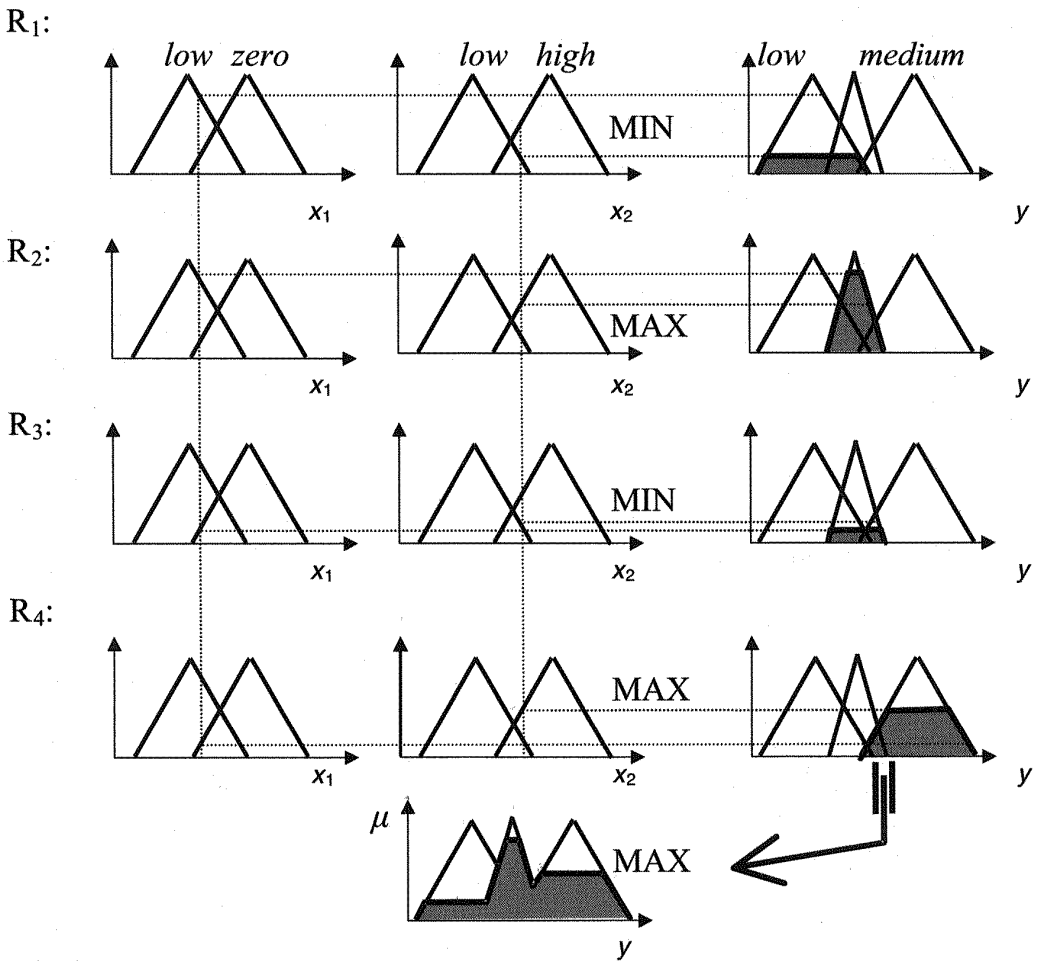


Figure 6.20
Construction of the consequent membership function from four active rules for a system with two inputs and one output.

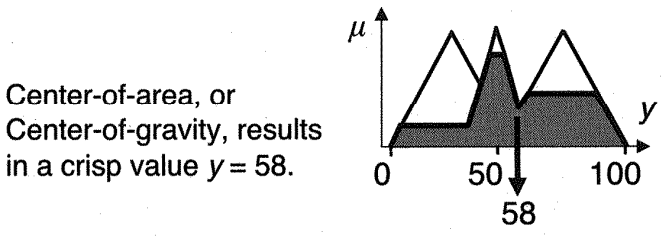


Figure 6.21
Defuzzification, or obtaining a crisp value from a fuzzy set: center-of-area (or center-of-gravity) method.

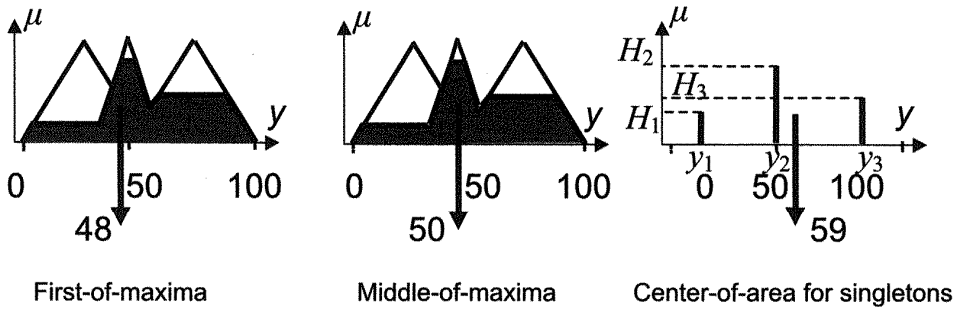


Figure 6.22
Graphical representation of three popular defuzzification methods.

Each of these methods possesses some advantages in terms of, for example, complexity, computing speed, and smoothness of the resulting approximating hypersurface. Thus, *the first-of-maxima method* is the fastest one and is of interest for real-time applications, but the resulting surface is rough. The *center-of-area for singletons method* is eventually the most practical because it has similar smoothness properties as the *center-of-area method* but is simpler and faster.

When the membership functions of the output variables are singletons and when the PROD operator is used for inference, it is relatively easy to show the equality of the neural network and fuzzy logic models (see section 6.2). The resulting crisp output in this particular case is calculated as

$$y' = \frac{\sum_{i=1}^{N} y_i H_i}{\sum_{i=1}^{N} H_i}, \tag{6.16}$$

where N is the number of the output membership functions. Equation (6.16) is also valid if the MIN operator is used when singletons are the consequents. Note the important distinction between the relational matrices models used in section 6.1 and the fuzzy additive models (FAMs) used in section 6.3. Here, MAX-PROD or MAX-MIN implication is used, but FAMs use SUM-PROD or SUM-MIN or SUM-any-other-T-norm implication. The practical difference regarding (6.16) is that in the case of FAMs, N stands for the number of rules.

At the beginning of this chapter it was stated that human knowledge structured in the form of IF-THEN rules represents a mapping, or a multivariate function, that maps the input variables (IF variables, antecedents, causes) into the output ones (THEN variables, consequents, effects). Now, this is illustrated by showing how our common knowledge in controlling the distance between our car and the vehicle in

front of us while driving is indeed a function. It is a function of which we are usually not aware. To show it graphically, the input variables are restricted to the two most relevant to this control task: the distance between the two vehicles and the speed.

The surfaces shown in figure 6.23 are *surfaces of knowledge* because all the control actions (producing the braking force, in this task) are the results of sliding on this surface. Normally, we are totally unaware of the very existence of this surface, but it is stored in our minds, and all our decisions concerning braking are in accordance with this two-dimensional function. In reality, this surface is a projection of one hypersurface of knowledge onto a three-dimensional space. In other words, additional input variables are involved in this control task in the real world. For example, visibility, wetness, or the state of the road, our mood, and our estimation of the quality of the driver in the car in front. Taking into account all these input variables, there is a mapping of five more input variables besides the two already mentioned into the one output variable (the braking force). Thus, in this real-life situation, the function is a hypersurface of knowledge in eight-dimensional space (it is actually an $\Re^7 \rightarrow \Re^1$ mapping). Let us stay in the three-dimensional world and analyze a fuzzy model for controlling the distance between two cars on a road.

Example 6.14 Develop a fuzzy model for controlling the distance between two cars traveling on a road. Show the resulting surface of knowledge graphically.

There are two input variables (distance and the speed) and one output variable (braking force), and five chosen fuzzy subsets (membership functions, attributes) for each linguistic variable. The membership functions for the input (the IF) variables are triangles. The fuzzy subsets (attributes) of the output variable are singletons.

Fuzzy subsets (attributes) of distance	[*very small, small, moderate, large, very large*]
Fuzzy subsets (attributes) of speed	[*very low, low, moderate, high, very high*]
Fuzzy subsets (attributes) of braking force	[*zero, one-fourth, one-half, three-fourths, full*]

Now, the rule basis comprises 25 rules of the following type:

R_1: IF distance *very small* AND speed *very low*, THEN braking force *one-half.*

The inference was made using a MAX-MIN operator. Two surfaces of knowledge are shown in figure 6.23. The smooth one (left graph) is obtained by center-of-gravity defuzzification, and the rough one (right graph) is obtained by first-of-maxima defuzzification. ■

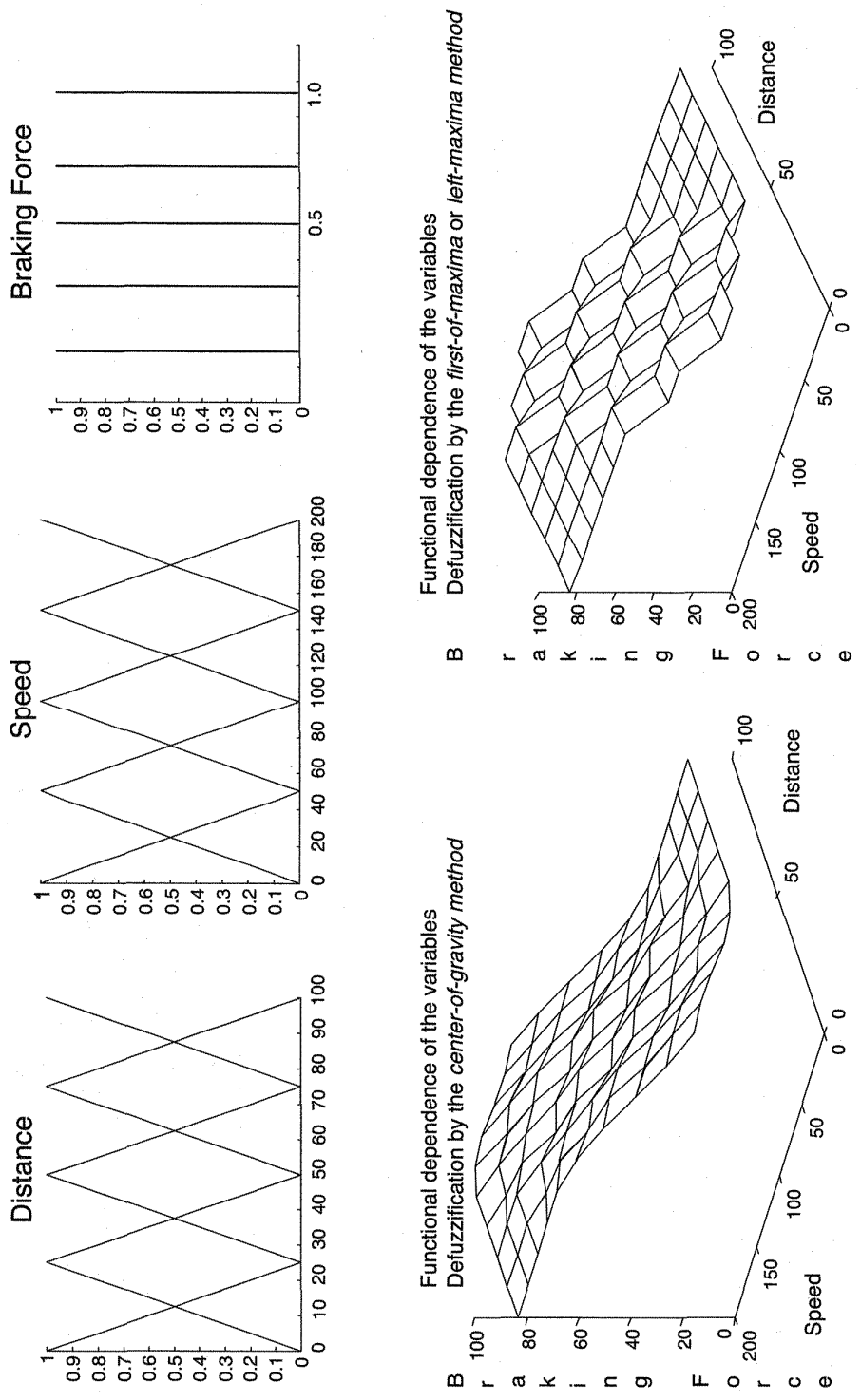


Figure 6.23
Fuzzy model for controlling the distance between two cars. *Top*, the fuzzy subsets of the two input variables (distance and speed) and one output variable (braking force). *Bottom*, the two surfaces of knowledge obtained by different defuzzification methods.

Some comments and conclusions can now be stated. First, using fuzzy logic models, one tries to model structured human knowledge. This knowledge is highly imprecise. We all drive a car differently. Even at the very first step, each of us would differently define the universes of discourse, that is, the domains of the input and output variables. Younger or less cautious drivers would definitely consider distances of 100 meters intolerably large. They would drive *very close*, meaning that the maximal value of the distance's fuzzy subsets (*very large*) would perhaps be 50 m. On the other hand, more cautious drivers would probably never drive at velocities higher than 120 km/h. Second, the choice (shapes and positions) of the membership functions is highly individual. Third, the inference mechanism and the defuzzification method applied will also have an impact on the final result. Despite all these fuzzy factors, the resulting surface of knowledge that represents our knowledge with regard to the solution of the given problem is usually an acceptable one. If there is usable knowledge, fuzzy logic provides the tools to transfer it into an efficient algorithm.

Compare the two surfaces shown in figure 6.23. Both surfaces model the known facts: a decrease in distance or an increase in driving speed, or both, demands a larger braking force.

Note that where there are several input and output variables, nothing changes except required computing time and required memory. If the resulting hypersurfaces reside in four- or higher-dimensional space, visualization is not possible but the algorithms remain the same.

6.2 Mathematical Similarities between Neural Networks and Fuzzy Logic Models

As mentioned, neural networks and fuzzy logic models are based on very similar, sometimes equivalent, underlying mathematics. To show this very important result the presentation here follows a paper by Kecman and Pfeiffer (1994) showing when and how the *learning of fuzzy rules* (LFR) from numerical data is mathematically equivalent to the training of a *radial basis function* (RBF) or regularization, network. Although these approaches originate in different paradigms of intelligent information processing, their mathematical structure is the same. These models also share the property of being a universal approximator of any real continuous function on a compact set to arbitrary accuracy. In the LFR algorithm proposed here, the subjects of learning are the rule conclusions, that is, the positions of the membership functions of output fuzzy sets (also called attributes) that are in form of singletons. For the fixed number, location, and shape of the input membership functions in the FL model or the basis functions in an RBF network, LFR and RBF training becomes a least-squares optimization problem that is linear in unknown parameters. (These

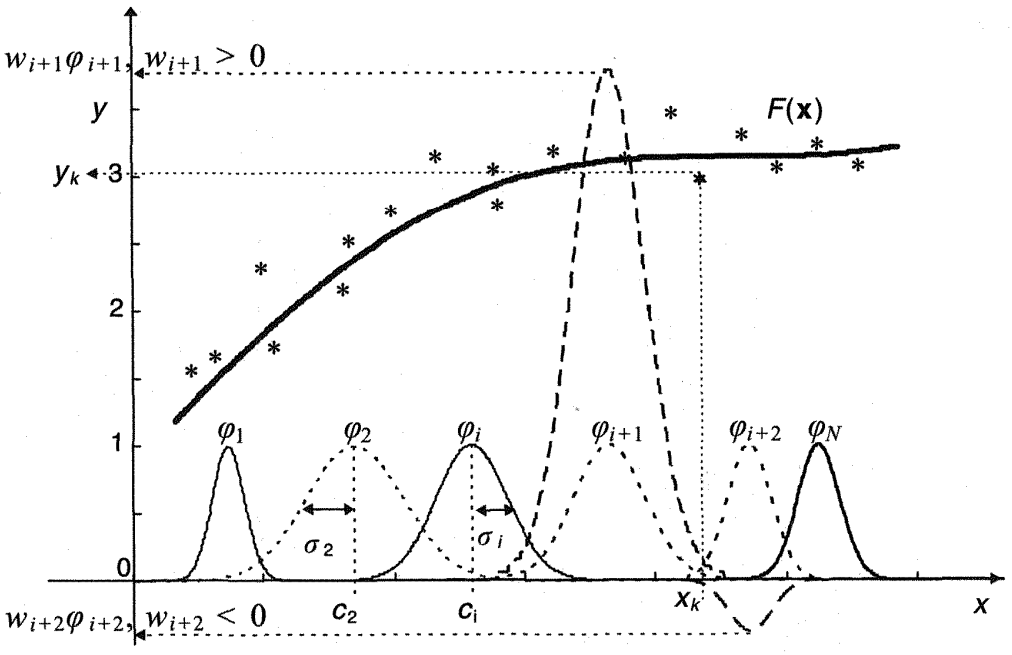


Figure 6.24
Training data set (asterisks), basis functions φ or membership functions μ, and nonlinear approximating function $F(x)$. Note that with fixed φ (or μ) this approximation is linear in parameters.

parameters are the OL weights **w** for RBFs or rules **r** for LFR). In this case, the solution boils down to the pseudoinversion of a rectangular matrix. The presentation here can be readily extended to the other, not necessarily radial, activation functions.

Using these two approaches, the general problem of approximating or learning mapping f from an n-dimensional input space to an m-dimensional output space, given a finite training set of P examples of $f\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_P, \mathbf{y}_P), \mathbf{y}_p = f(\mathbf{x}_p)\}$, is exemplified by the one-to-one mapping presented in figure 6.24.

For real-world problems with noisy data, one never tries to find the function F that interpolates a training set passing through each point in the set, that is, one does not demand $\boldsymbol{F}(\boldsymbol{x}_p) = \boldsymbol{y}_p$, $\forall p \in \{1, \ldots, P\}$. The approximating function $\boldsymbol{F}$ of the underlying function f will be obtained on the relaxed condition that $\boldsymbol{F}$ does not have to go through all the training set points but should merely be as close as possible to the data.

Usually, the criterion of such closeness is least squares. It is clear that in the case of noisy free data, interpolation is the better solution. This will be true as long as the size of the data set is not too large. Generally, in the case of large data sets (say, more than 1,000 data patterns) because of numerical problems, one is forced to find an approximation solution.

In RBF approximation techniques (which resulted from the regularization theory of Tikhonov and Arsenin (1977), a good theoretical framework for the treatment of approximation or interpolation problems), after making some mild assumptions the expression for an approximating function $\mathbf{F}$ has the following simple form:

$$\mathbf{y} = \boldsymbol{F}(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi_i(\mathbf{x}, \mathbf{c}_i), \tag{6.17}$$

where w_i are weights to be learned and $\mathbf{c}_i$ are the centers of the radial basis functions φ_i. RBF φ_i can have different explicit forms (e.g., spline, Gaussian, multiquadric).

It is important to realize that when the number N, the positions $\mathbf{c}_i$, and the shapes (defined by the parameter σ and by the covariance matrix $\boldsymbol{\Sigma}$ for one-dimensional and higher-dimensional Gaussian basis functions, respectively) are fixed before learning, the problem of approximation is linear in the parameters (weights w_i), which are the subject of learning. Thus, the solution boils down to the pseudoinversion of matrix $\Phi(P, N)$. This matrix is obtained using (6.17) for the whole training set. If any of the parameters $\mathbf{c}_i$ or σ_i, which are "hidden" behind the nonlinear function φ_i, become part of the training for any reason, the problem of learning will have to be solved by nonlinear optimization. Certainly then it will be much more involved.

Consider a scalar output variable in order to show the equality of neural networks and fuzzy logic models without loss of generality. The RBF network modeling the data set is given as

$$y = F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi_i(\mathbf{x}, \mathbf{c}_i). \tag{6.18}$$

If φ is a Gaussian function (usually the normalized Gaussian with amplitude $G(c_i, c_i) = 1$ is used), one can write

$$y = F(\mathbf{x}) = \sum_{i=1}^{N} w_i G_i(\mathbf{x}, \mathbf{c}_i). \tag{6.19}$$

Figure 6.24 presents (6.19) graphically. For $N = P$ and $N < P$ an interpolation or an approximation, respectively, will occur.

The same approximation problem can be considered a problem of learning fuzzy rules from examples. Figure 6.24 still represents the problem setup but now the Gaussian bumps are interpreted as membership functions μ_i of the linguistic attributes (fuzzy subsets) of the input variable x (input is now a one-dimensional variable).

For reasons of computational efficiency, the attributes of the linguistic output variable are defuzzified off-line by replacing the fuzzy set of each attribute with a singleton at the center of gravity of the individual fuzzy set (as in fig. 6.23, the braking force graph).

The parameters to be learned are the positions r_i of the singletons describing the linguistic rule conclusions. The corresponding continuous universes of discourse for linguistic input and output variables $\text{Input}_1, \ldots, \text{Input}_n$, and Output, are called $X_1, \ldots, X_n$, Y, respectively. Rule premises are formulated as fuzzy AND relations on the Cartesian product set $X = X_1 \times X_2 \times \cdots \times X_n$, and several rules are connected by logical OR. Fuzzification of a crisp scalar input value x_1 produces a column vector of membership grades to all the attributes of Input_1, and similarly for all other input dimensions, for instance,

$$\mu_1 = \begin{bmatrix} \mu_{1,\text{“attr1”}}(x_1) \\ \mu_{1,\text{“attr2”}}(x_1) \\ \vdots \end{bmatrix}, \qquad \mu_2 = \begin{bmatrix} \mu_{2,\text{“attr1”}}(x_2) \\ \mu_{2,\text{“attr2”}}(x_2) \\ \vdots \end{bmatrix}. \tag{6.20}$$

The degrees of fulfillment of all possible AND combinations of rule premises are calculated and written into a matrix **M**. For ease of notation, the following considerations are formulated for only two input variables, but they can be extended to higher-dimensional input spaces. If the *algebraic product* is used as an AND operator, this matrix can be directly obtained by the multiplication of a column and a row vector:

$$\mathbf{M}(\mathbf{x}) = \mu_1(x_1)\mu_2^T(x_2). \tag{6.21}$$

Otherwise, the minimum or any other appropriate operator from table 6.2 can be applied to all pairs of membership values. Because the attributes of the linguistic output variable are singletons, they appear as crisp numbers in the fuzzy rule base. The first rule, for example, reads

R_1: IF Input_1 is *attribute*$_1$ AND Input_2 is *attribute*$_2$, THEN Output is r_{11}.

and its conclusion is displayed as the element r_{11} in the relational (rule) matrix

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots \\ r_{21} & r_{22} & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix}. \tag{6.22}$$

R has the same dimensions as **M**. IF-THEN rules are interpreted as AND relations on $X \times Y$, that is, the degree of membership in the output fuzzy set of a rule is limited to the degree up to which the premise is fulfilled. A crisp output value y is computed by the center-of-area for singletons, or center-of-singletons, algorithm (6.16) as a weighted mean value

$$y = F(x) = \frac{\sum_{jl} \mu_{jl}(x) r_{jl}}{\sum_{jl} \mu_{jl}(x)}, \tag{6.23}$$

where $\mu_{jl} = H_i$ and $r_{jl} = y_i$. The sum covers all elements of the two matrices **M** and **R**. If the membership functions of the input attributes are Gaussians, the μ_{jl} are space bumps $G_i(\mathbf{x}, \mathbf{c}_i)$ representing the joint possibility distribution of each rule. Moreover, if the elements of matrix **R** are collected in a column vector

$$\mathbf{r} = (r_{11}, r_{12}, \ldots r_{21}, r_{22}, \ldots)^T = (r_1, r_2, \ldots, r_N)^T, \tag{6.24}$$

the approximation formula becomes

$$y = F(\mathbf{x}) = \frac{\sum_{i=1}^{N} G(\mathbf{x}, \mathbf{c}_i) r_i}{\sum_{i=1}^{N} G(\mathbf{x}, \mathbf{c}_i)}. \tag{6.25}$$

The structural similarity of (6.19) and (6.25) is clearly visible: the rule conclusions r_i correspond to the trainable weights w_i. These two equations could be given a graphical representation in the form of "neural" networks. For bivariate functions $y = f(x_1, x_2)$ this is done in figure 6.25.

The structures of both networks are the same in the sense that each has just one hidden layer and the connections between the input and the hidden layer are fixed and not the subject of learning. The subjects of learning are the connections w or r between the hidden layer and the output layer. It must be stressed that the seemingly second hidden layer in a fuzzy (or soft RBF) network is not an additional hidden layer but the normalization part of the only hidden layer. Because of this normalization, the sum of the outputs from the hidden layer in a soft RBF network is equal to 1, that is, $\Sigma o_{iF} = 1$. This is not the case in a classic RBF network.

The equivalence of these two approximation schemes is clear if (6.19) and (6.25) are compared. The only difference is that in fuzzy approximation the output value from the hidden layer y is "normalized." The word *normalized* is in quotation marks because y is calculated using the normalized output signals o_{iF} (fig. 6.25) from neurons

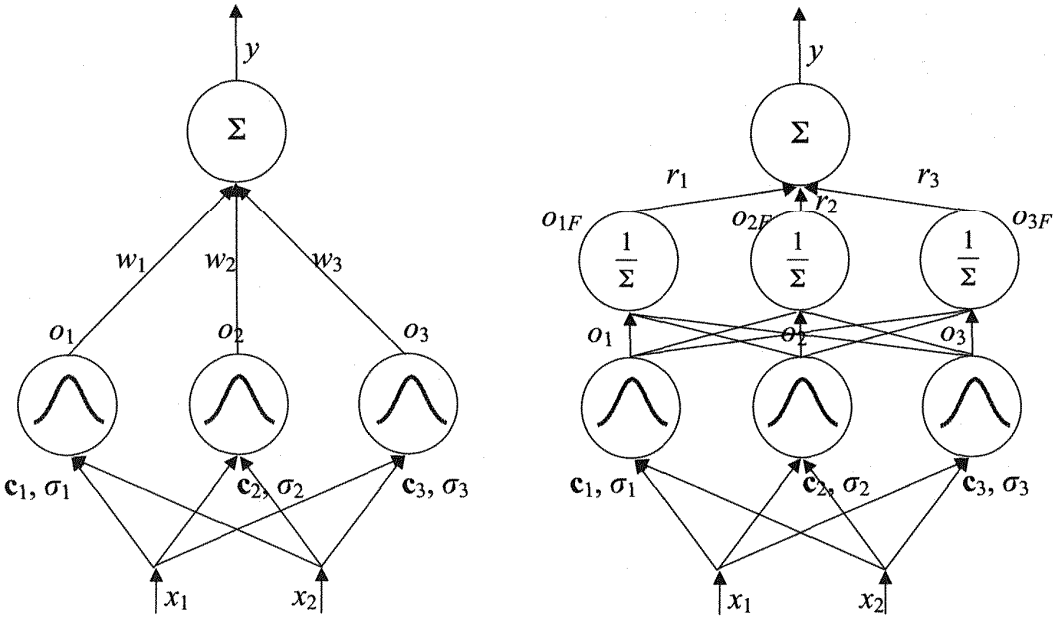


Figure 6.25
Networks for the interpolation ($N = P$) or approximation ($N < P$) of a bivariate function $y = f(x_1, x_2)$. *Left*, an RBF. *Right*, a fuzzy network or soft RBF. Here $N = 3$.

whose sum is equal to 1. This is not the case with a standard RBF network. Because of the effect of normalization, fuzzy approximation is a kind of soft approximation, with the approximating function always going through the middle point between the two training data. As an analogy to the softmax function introduced to the neural network community for the sigmoidal type of activation function (Bridle 1990), fuzzy approximation is called a soft RBF approximation scheme.

The mathematical side of the solution is defined as follows: for a fixed number N, positions c_i, and width σ_i of the basis function φ_i or membership function μ_i, the problem of approximation is linear in learned parameters w or r and will be solved by the simple inversion of the matrix **A** given in (6.30). (In the case of interpolation, i.e., when the number N of basis or membership functions (attributes) for each input variable is equal to the number P of data pairs, **A** is a square matrix. When there are fewer basis or membership functions than data pairs, **A** is rectangular. The latter type of approximation is more common in real-life problems.) This property of being linear in parameters is not affected by the choice of the algebraic product as a fuzzy AND operator. This algorithm remains the same for the minimum operator.

It seems as though the soft RBF is more robust with respect to the choice of the width parameter and has better approximation properties in cases when there is no large overlapping of basis functions φ or membership functions μ. In such a situation

(for small σ in the case of Gaussian functions) the approximation obtained with the classic RBF given by (6.19) will be much more spiky than the one obtained with fuzzy approximation, or the soft RBF given by (6.25).

There is a significant difference in the physical meaning of the learned (or trained) weights w_i or rules r_i in these two paradigms. Approaching the problem from a fuzzy perspective, the rules have from the very start of problem formulation a clear physical meaning, stating that an output variable must take a certain value under specified conditions of input variables. There is no such analogy in the classic RBF approach to functional approximation. In the latter case, the meaning of weights w_i is more abstract and depends on such small subtleties as whether normalized Gaussians $G(c_i, c_i) = 1$ are used. Generally, in both methods, with increased overlapping of the basis or membership functions, the absolute values of the parameters w or r will increase. But, in the fuzzy case, when the resulting output variables are rules r, we are aware of their physical limits, and these limits will determine the actual overlapping of the membership functions in input space. There are no such caution signs in a classic RBF because that approach is derived from a mathematical domain.

In order to apply a standard least-squares method in the spirit of parameter estimation schemes, the dedicated fuzzy identification algorithm for the center-of-singletons defuzzification method must be slightly reformulated by collecting the elements of $\mathbf{M}$ in a column vector

$$\mu = [\mu_{11} \quad \mu_{12} \quad \ldots \quad \mu_{21} \quad \mu_{22} \quad \ldots]^T, \tag{6.26}$$

and by defining a vector of 1's with the same dimension $\mathbf{1} = (1 \quad 1 \quad \ldots \quad 1)^T$. Using these vectors, (6.23) can be written with the numerator and denominator calculated as scalar products

$$y = \frac{\mu^T \mathbf{r}}{\mu^T \mathbf{1}}, \tag{6.27}$$

which is equivalent to

$$\mu^T \mathbf{r} = \mu^T \mathbf{1} y. \tag{6.28}$$

The input data are fuzzified according to the attributes of the linguistic variables Input_1 and Input_2. For each sample p, and input data set $\mathbf{x}_p$, a corresponding vector μ_p is obtained by applying formulas (6.20), (6.21), and (6.26) successively, and an equation of the form (6.28) is stated as

$$\mu_p^T \mathbf{r} = \mu^T \mathbf{1} y_p. \tag{6.29}$$

From this equation a system of linear equations is constructed for $p = 1, \ldots, P$

$$\mathbf{Ar} = \mathbf{b} \Rightarrow \begin{bmatrix} \mu_1^T \\ \mu_2^T \\ \vdots \\ \mu_P^T \end{bmatrix} \mathbf{r} = \begin{bmatrix} \mu_1^T \mathbf{1} y_1 \\ \mu_2^T \mathbf{1} y_2 \\ \vdots \\ \mu_P^T \mathbf{1} y_P \end{bmatrix}. \tag{6.30}$$

This system is in linear form, with a known rectangular matrix $\mathbf{A}$ and a known vector $\mathbf{b}$.

$$\mathbf{Ar} = \mathbf{b}. \tag{6.31}$$

Now (6.31) can be solved for the unknown vector $\mathbf{r}$ by any suitable numerical algorithm, for instance, by taking the pseudoinverse as an optimal solution in a least-squares sense:

$$\mathbf{r} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^+\mathbf{b}. \tag{6.32}$$

Finally, the elements of vector $\mathbf{r}$ can be regrouped into the rule matrix $\mathbf{R}$. The matrix $\mathbf{A}$ actually contains degrees of fulfillment of all rules. For a system with N rules, its dimensions are (P, N). Therefore the matrix $\mathbf{A}^T\mathbf{A}$ is of the dimension (N, N) and can be easily inverted, even for very large numbers of data samples. This explains the equivalence of the RBF and FL models and also shows how the weights or rules can be adapted (trained). The final learning rule (the matrix $\mathbf{R}$ in (6.22)) was a relatively simple one because the hidden layer weights were fixed.

The structural equivalence of a certain type of learning fuzzy system with trainable RBF neural networks is of considerable importance to the neural network and fuzzy logic communities.

A regularization (RBF) network can be interpreted in terms of fuzzy rules after learning, providing an insight into the physical nature of the system being modeled that cannot be obtained from a black-box neural network. Moreover, the "linear" training algorithm (6.32) can be transformed to recursive notation according to method 5 in section 3.2.2. This opens the door to recursive training of RBF networks in real time with much better convergence properties than error backpropagation networks. From an RBF perspective, a recursive formulation will avoid the problems of pseudoinversion of large matrices when dealing with a large number of basis functions. Also, in using soft RBFs there is no requirement for basis functions to be radial. Experiments by the author and colleagues used logistic and tangent hyperbolic functions with an approximation quality comparable to radial Gaussian functions.

The relevance of this result for the fuzzy logic community is of another nature. It suggests preference for a certain type of membership function (e.g., Gaussian); fuzzy operator (e.g., algebraic product); and a specific kind of inference and defuzzification scheme (e.g., the center-of-singletons algorithm) for modeling tasks. Namely, for fuzzy models, good approximation qualities are guaranteed by the equivalence to regularization networks, whose mathematical properties are firmly established. Moreover, this equivalence provides new insight into the interpolation/approximation aspects of fuzzy models and into such questions as the selection of a suitable number of membership functions and degrees of overlapping. If the nonlinear learning techniques known from neural networks are applied to such fuzzy systems, they allow the learning of input membership functions as well.

6.3 Fuzzy Additive Models

A fuzzy model given by (6.23) or (6.25) is equivalent to an RBF model. This means that fuzzy logic models are also universal approximators in the sense that they can model any multivariate function to any desirable degree of accuracy. The higher the required accuracy, the more rules are needed. This expression is valid for any $\Re^n \rightarrow \Re^m$ mapping. In the case of an $\Re^n \rightarrow \Re^1$ function, (6.25) becomes

$$y = F(\mathbf{x}) = \frac{\sum_{i=1}^{N} \mu(\mathbf{x}, \mathbf{c}_i) r_i}{\sum_{i=1}^{N} \mu(\mathbf{x}, \mathbf{c}_i)}, \tag{6.33}$$

where the notation μ is used for a membership function (degree of belonging) instead of G. N denotes the number of rules, and r_i stands for the center of area (center of gravity, centroid) of the ith output singleton. When modeling an $\Re^1 \rightarrow \Re^1$ relation, equation (6.33) describes a standard fuzzy IF-THEN rule: IF $x = S_{xi}$, THEN $y = S_{yi}$, where S_{xi} and S_{yi} are the antecedent and consequent membership functions, respectively.

This is the simplest form of a *fuzzy additive model* (FAM), also known as a *standard additive model* (SAM).[5] The adjective *additive* refers to the summations that take place in (6.33). Hence, this model can also be called a SUM-PROD or SUM-MIN implication model. All the models in section 6.1 are based either on the Mamdani implication (MAX-MIN) or the Larsen inference method (MAX-PROD). Consequently, they are not additive models. It is important to realize that so far only additive models are proven universal approximators for fuzzy membership functions of any shape.

The simplest FAM model as given by (6.33) is valid when the output membership functions are singletons. A more general model for an $\mathfrak{R}^n \rightarrow \mathfrak{R}^1$ mapping (when the output fuzzy subsets, i.e., attributes or membership functions, are fuzzy subsets of any shape) is given by

$$y = F(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i \mu(\mathbf{x}, \mathbf{c}_i) A_i m_i}{\sum_{i=1}^{N} w_i \mu(\mathbf{x}, \mathbf{c}_i) A_i}. \tag{6.34}$$

where N denotes the number of rules, w_i stands for the relative rule weight (if the ith rule is more relevant than the jth rule, $w_i > w_j$), A_i is the area of the corresponding ith output fuzzy subset (membership function), and m_i stands for the mode, that is, the center of area (center of gravity, centroid) of the ith output fuzzy subset.

In the case of an $\mathfrak{R}^n \rightarrow \mathfrak{R}^m$ mapping, a FAM given by (6.34) becomes

$$\mathbf{y} = F(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i \mu(\mathbf{x}, \mathbf{c}_i) V_i \mathbf{m}_i}{\sum_{i=1}^{N} w_i \mu(\mathbf{x}, \mathbf{c}_i) V_i}, \tag{6.35}$$

where V_i is the volume of the corresponding ith output fuzzy subset (membership function). When all rules are equally relevant, $w_i = w_j$, $i = 1, N, j = 1, N$, and when the output fuzzy subsets have equal volumes (or areas in the case of an $\mathfrak{R}^m = \mathfrak{R}^1$ mapping), (6.35) reduces to (6.33).

The basic description of how this additive model achieves the desired accuracy is given in example 6.15 for an $\mathfrak{R}^1 \rightarrow \mathfrak{R}^1$ mapping. The design steps in fuzzy modeling are shown in box 6.1.

Box 6.1
Design Steps in Fuzzy Modeling

Step 1. Define the universes of discourse (domains and ranges, i.e., input and output variables).

Step 2. Specify the fuzzy membership functions (fuzzy subsets or attributes) for the chosen input and output variables.

Step 3. Define the fuzzy rules (i.e., the rule base).

Step 4. Perform the numerical part (SUM-PROD, SUM-MIN, MAX-MIN, or some other) inference algorithm.

Step 5. Defuzzify the resulting (usually not-normal) fuzzy subsets.

Example 6.15 Design a fuzzy controller for controlling a distance between two cars traveling on a road.

Example 6.14 demonstrated how the braking force B depends upon the distance D and the velocity v, but it did not give details. Here, in order to get a geometrical impression of a FAM works and to enable visualization, it is assumed that the distance D is constant, and only the mapping, $B = f(v)$, is modeled. Figure 6.26 shows the scheme for this model.

In the case of a mapping $B = f(v)$, a very simple (rough) model can be obtained by having three rules only. Both the fuzzy subsets and the rule base are shown in figure 6.27. The rule base is linguistically expressed everyday driving expertise on how to brake depending upon the velocity of the car.

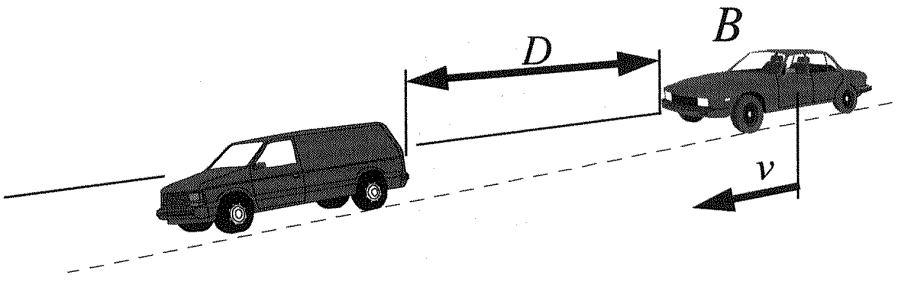
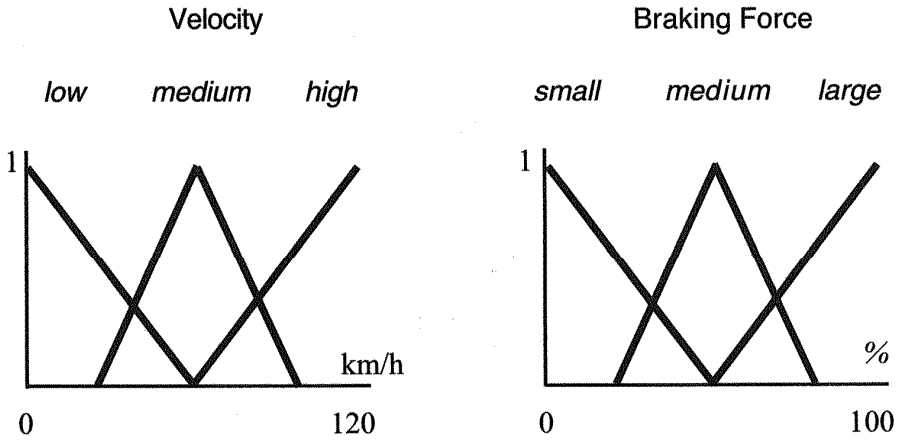


Figure 6.26
Scheme for modeling a fuzzy braking force controller.



IF velocity is *low*, THEN braking force is *small*.
IF velocity is *medium*, THEN braking force is *medium*.
IF velocity is *high*, THEN braking force is *large*.

Figure 6.27
Fuzzy subsets (membership functions) and a rule base for a fuzzy braking force controller.

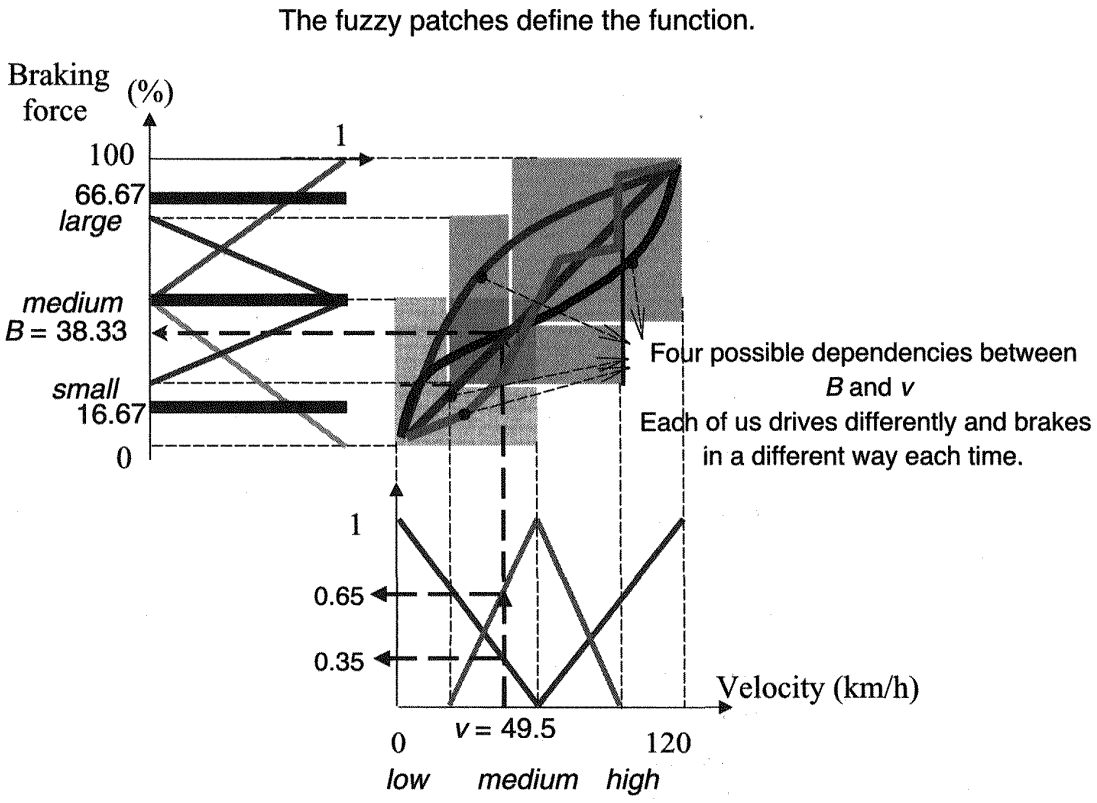


Figure 6.28
Fuzzy rules define the fuzzy patches. More rules result in smaller patches. This means finer granulation, i.e., more precise knowledge.

Figure 6.28 shows the four possible mapping curves that may result from a FAM's having three rules only. A much finer (more accurate) model could have been obtained with finer discretization (more membership functions and rules). The rules and the fuzzy subsets with high overlapping produce the three square patches in a (B, v) plane inside which must lie the function $B = f(v)$, the desired $\Re^1 \rightarrow \Re^1$ mapping. The very shape of a function $B = f(v)$ depends on the shapes of the membership functions applied, on the implication mechanism, and (heavily) on the defuzzification method used. Figure 6.28 shows four possible solutions. This is the soft part of the fuzzy modeling that models the basic dependency stating that with an increase in velocity there must be an increase in braking force. Patch size generally defines the vagueness or uncertainty in the rule. It is related to the number of rules: more rules, smaller patches. There is no unique or prescribed way to brake. We all drive differently, and the way each of us applies the force to the brakes is different each time. Hence, many possible functions result from different experiences, and the fuzzy models that are based on expertise will also be different. Some of the possible variety can be seen in figure 6.28.

Consider now how the FAM models the braking force controller. For the output fuzzy subsets, three singletons are chosen, placed as follows: *small* ($r_1 = 16.67\%$), *medium* ($r_2 = 50\%$), *large* ($r_3 = 66.67\%$). These singleton membership functions are chosen to be at the centers of gravity of the corresponding triangle fuzzy subsets (i.e., $r_i = m_i$) and are shown as thick lines exactly at these given locations. (Note they are not shown in the fig. 6.27.) For a particular crisp input $v = 49.5$ km/h, shown in figure 6.28, only two rules, namely R_1 and R_2, will be active. In other words, their corresponding degrees of belonging μ_i, $i = 1, 2$, will be different from zero, or only these two rules will "fire." The output—a braking force B—for this particular input $v = 49.5$, follows from the FAM model (6.33) as

$$B = f(v = 49.5) = \frac{(0.35 \cdot 16.67) + (0.65 \cdot 50) + (0 \cdot 66.7)}{0.35 + 0.65} = 38.33\%. \quad \blacksquare$$

It is important to realize that the FAMs do not perform very complex calculations. After the membership functions are selected, the volumes V_i and centroids m_i can be computed in advance. Now, the N membership degrees $\mu_i(\mathbf{x})$, $i = 1, N$, are calculated for each specific input $\mathbf{x}$. Finally, having defined the rule weights w_i, the corresponding output value y is found using (6.35) and (6.34) for an $\Re^n \rightarrow \Re^m$ mapping and for an $\Re^n \rightarrow \Re^1$ mapping, respectively. Note that only a part of the corresponding degrees μ_i will be different from zero, meaning that only a part of rules will be active.

The most serious problem in applying FAMs is a *rule explosion* phenomenon. The number of rules increases exponentially with the dimensions of the input and output spaces. Thus, for example, if there are four input variables ($\mathbf{x}$ is a four-dimensional vector) and a single output y, and if one chooses five fuzzy subsets for each input variable, one has to define $5^4 = 625$ rules. In other words, according to figure 6.25, this represents a network with 625 hidden layer neuron. Another serious problem is the learning part in fuzzy models. Theoretically, because of the equivalence of RBF networks and FL models, one can apply any learning approach from the neural field, including the gradient descent method. However, the standard membership functions are not smooth differentiable functions, and error backpropagation is not as popular in the fuzzy logic field as it is in the neural field. Genetic algorithms may be viable techniques, as may other methods for RBF networks training. In particular, a linear programming approach, as given in section 5.3.4, seems promising for selecting the most relevant rules in FAM.

The single important computational step in applying FAMs is the calculation of membership degrees $\mu_i(\mathbf{x})$, $i = 1, N$. In performing this task the most popular oper-

ators are MIN and PROD. A standard fuzzy rule is expressed separately for each input variable, for instance,

R: IF x_1 is *small* AND x_2 is *large* AND, ..., x_n is *medium*, THEN y is *positive*.

In other words, a typical IF-THEN rule operation is a *conjunction* (interpreted as logical AND), and any T-norm operator can be used in the calculation of membership degrees $\mu_i(\mathbf{x})$, $i = 1, N$. The two most popular operators (shown here for an n-dimensional input vector $\mathbf{x}$) are the MIN operator $\mu_{x_1 \wedge x_2 \wedge \cdots \wedge x_n} = \text{MIN}(\mu_{x_1}, \mu_{x_2}, \ldots, \mu_{x_n})$ and the PROD (algebraic product) operator $\mu_{x_1 \wedge x_2 \wedge \cdots \wedge x_n} = \mu_{x_1}\mu_{x_2} \cdots \mu_{x_n}$. If the IF part contains an OR connection, the MIN operator must be replaced with a MAX operator or some other S-norm. However, an application of the OR in an IF part rarely happens in practice.

The algebraic product gives much smoother approximations because it does not ignore information contained in the IF part as the MIN operator does. How the two operators use the information contained in an input vector is shown by a simple example here.

Suppose that an input vector results in following membership degrees (activations) $\boldsymbol{\mu}_1 = [\mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4 \quad \mu_5]^T = [0.7 \quad 0.4 \quad 0.4 \quad 0.5 \quad 0.5]^T$ and that another input vector gives $\boldsymbol{\mu}_2 = [0.7 \quad 0.4 \quad 0.9 \quad 0.9 \quad 1.0]^T$. The MIN operator results in $\boldsymbol{\mu}_1(\mathbf{x}) = \boldsymbol{\mu}_2(\mathbf{x}) = 0.4$, while the PROD operator gives $\boldsymbol{\mu}_1(\mathbf{x}) = 0.028$ and $\boldsymbol{\mu}_2(\mathbf{x}) = 0.2268$. Hence, the MIN operator does not differentiate the joint strength of the inputs and in both cases results in the same activation despite the obvious fact that $\boldsymbol{\mu}_2$ contains much stronger activations.

The product $\mu(\mathbf{x}) = \mu_1(x_1)\mu_2(x_2) \cdots \mu_n(x_n)$ gets smaller for larger input dimension n, but this does not affect the FAM output because $\mu(\mathbf{x})$ is a part of both the numerator and denominator in (6.33)–(6.35). Note an important fact that by using the product of n scalar membership functions $\mu(\mathbf{x}) = \prod_{i=1}^{n} \mu_i(x_i)$, the possible correlations among the input components x_i were ignored.

The relational matrix approach as given in section 6.1 does not add the resulting output (typically) not-normal fuzzy sets, and instead of the SUM operator it uses MAX. Figure 6.19 shows the resulting consequent not-normal fuzzy membership function after the MAX operator has been applied. This MAX-PROD resulting consequent fuzzy subset is shown in figure 6.29 together with the resulting consequent not-normal fuzzy membership function after applying the SUM operator.

There are two distinct advantages in using FAMs (SUM-MIN or SUM-PROD or SUM-any-other-T-norm model) with respect to an application of the relational

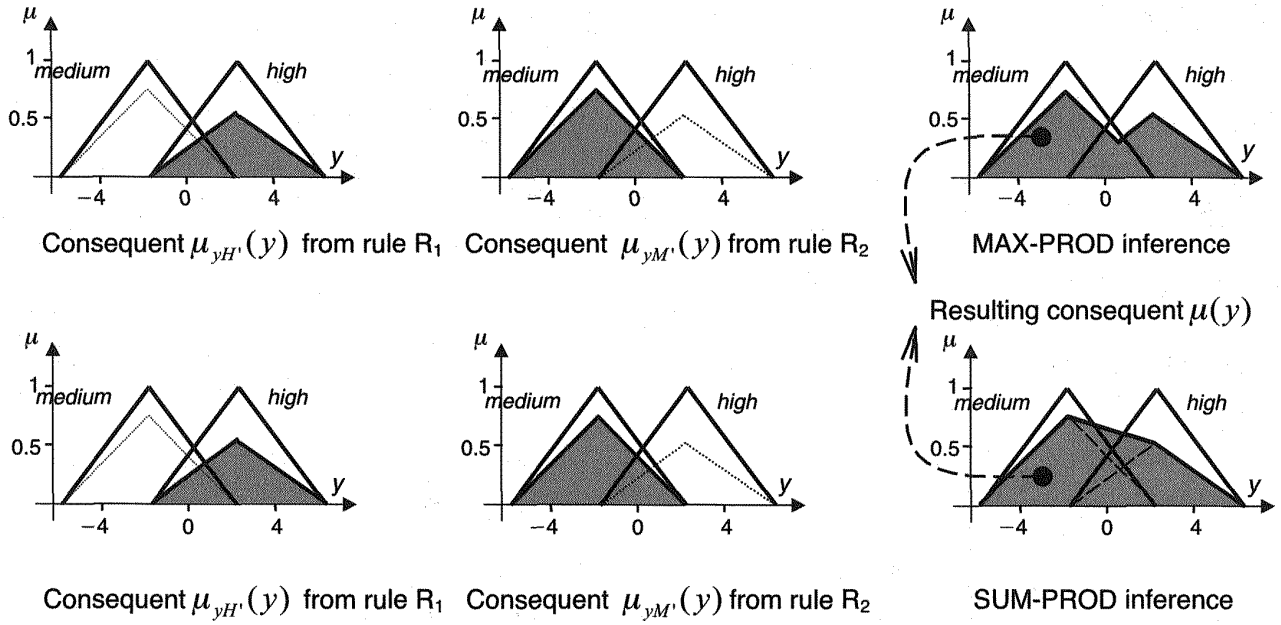


Figure 6.29
Construction of the consequent membership functions from example 6.12 for a single-input, single-output system. *Top*, MAX-PROD inference. *Bottom*, SUM-PROD inference.

models presented in section 6.1 (MAX-MIN or MAX-PROD models). First, on theoretical level, FAMs are universal approximators, and there is no proof of this capacity for the relational models yet. Second, the computational part of a reasoning scheme is simplified through bypassing the relational matrix calculus.

Problems

6.1. On a cold winter morning, your mother tells you, "The temperature is about −10 °C today." Represent this piece of information by

a. a crisp set (a crisp membership function),

b. a fuzzy set.

6.2. Human linguistic expressions depend upon both the context and individual perceptions. Represent the following expressions by membership functions:

a. "large stone" (while you are in the mountains)

b. "large stone" (while you are in a jewelry store)

c. "high temperature today" (winter in Russia)

d. "high temperature today" (summer in Greece)

e. "high temperature today" (summer in Sweden)

6.3. Given is the fuzzy set S for a power plant boiler pressure P (bar) with the following membership function:

$$S(P) = \begin{cases} \frac{1}{25}(P - 200) & \text{if } 200 \leq P \leq 225 \\ -\frac{1}{25}(P - 200) & \text{if } 225 < P \leq 250 \\ 0 & \text{otherwise} \end{cases}$$

a. Sketch the graph of this membership function, and comment on its type.

b. Give the linguistic description for the concept conveyed by S.

6.4. Let three fuzzy sets be defined by an ordered set of pairs, where the first number denotes the degree of belonging (the membership degree) and the second number is the element:

$A = \{1/3, 0.2/4, 0.3/5, 0.4/6, 0.6/7, 0.8/8, 1/10, 0.8/12, 0.6/14\}$.

$B = \{0.4/2, 0.6/3, 0.8/4, 1/5, 0.8/6, 0.6/7, 0.4/8\}$.

$C = \{0.4/2, 0.8/4, 1/5, 0.6/7\}$.

Determine the intersections and unions of

a. the fuzzy sets A, B, and C,

b. the complements of fuzzy sets B and C if both sets are defined on the universe of discourse $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. (*Hint:* First express the complements B^C and C^C, taking into account X.)

6.5. Let the two fuzzy sets $A = \{x \text{ is considerably larger than } 10\}$ and $B = \{x \text{ is approximately } 11\}$ be defined by the following membership functions:

$$\mu_A(x) = \begin{cases} 0 & \text{for } x \leq 10 \\ [1 + (x - 10)^{-2}]^{-1} & \text{for } x > 10 \end{cases}.$$

$$\mu_B(x) = [1 + (x - 11)^4]^{-1}.$$

a. Sketch the graphs of these fuzzy sets, and draw the graphs of a fuzzy set $C = \{x \text{ is considerably larger than } 10 \text{ AND } x \text{ is approximately } 11\}$; and a fuzzy set $D = \{x \text{ is considerably larger than } 10 \text{ OR } x \text{ is approximately } 11\}$.

b. Express analytically the membership functions μ_C and μ_D.

6.6. Let two fuzzy sets be defined as follows:

$A = \{0.4/2, 0.6/3, 0.8/4, 1/5, 0.8/6, 0.6/7, 0.4/8\}$.

$B = \{0.4/2, 0.8/4, 1/5, 0.6/7\}$.

Determine the intersections of A and B by applying three different T-norms:

a. minimum,

b. product,

c. Lukasiewicz AND (bounded difference).

6.7. Determine the unions of A and B from problem 6.6 by applying three different T-conorms (S-norms):

a. maximum,

b. algebraic sum,

c. Lukasiewicz OR (bounded sum).

6.8. Prove that the following properties are satisfied by Yager's S-norm:

a. $\mu_{A \vee B}(x) = \mu_A(x)$ for $\mu_B(x) = 0$.

b. $\mu_{A \vee B}(x) = 1$ for $\mu_B(x) = 1$.

c. $\mu_{A \vee B}(x) \geq \mu_A(x)$ for $\mu_A(x) = \mu_B(x)$.

d. For $b \to 0$, the Yager's union operator (S-norm) reduces to a drastic sum.

6.9. Show that the drastic sum and drastic product satisfy the law of excluded middle and the law of contradiction. (*Hint:* The law of excluded middle states that $A \cup A^C = X$, and the law of contradiction says that $A \cap A^C = \varnothing$).

6.10. Prove that De Morgan's laws are satisfied if we take the union MAX operator and the intersection MIN operator, with the negation defined as

a. $N(x) = \dfrac{1 - x}{1 + \lambda x}$, $\lambda \in (-1, \infty)$.

b. $N(x) = \sqrt[r]{1 - x^r}$, $r \in (0, \infty)$.

(*Hint:* De Morgan's laws state that $\overline{A \cup B} = \bar{A} \cap \bar{B}$ and $\overline{A \cap B} = \bar{A} \cup \bar{B}$).

6.11. Let $X = \{8, 3, 10\}$ and $Y = \{2, 1, 7, 6\}$. Define the relational matrices for the following two relations: R_1: "x is considerably larger than y" and R_2: "y is very close to x". Now find the relational matrices for these two relations:

a. "x is considerably larger OR is very close to y"

b. "x is considerably larger AND is very close to y"

6.12. Consider a fuzzy rule: IF x is A, THEN y is B. The two fuzzy sets are given as follows:

$A = \{0/1, 0.1/2, 0.4/3, 0.8/4, 1/5\}, \qquad B = \{0/-2, 0.6/-1, 1/0, 0.6/1, 0/2\}.$

Find the relational matrices representing this rule by applying

a. MIN (Mamdani) implication (R_m),

b. Lukasiewicz implication (R_L),

c. Fuzzy implication $\text{MIN}(1, 1 - \mu_A(x) + \mu_B(x))$ (R_F).

6.13. Consider the input fuzzy set for the rule in problem 6.12. $A' = \{0/1, 0.2/2, 0.8/3, 1/4, 0.1/5\}$. Apply the three compositional rules of inference, and find the output fuzzy set (consequent) for a

a. MAX-MIN composition by using R_m from problem 6.12,

b. MAX-Lukasiewicz T-norm by using R_L from problem 6.12,

c. MAX-Lukasiewicz T-norm by using R_F from problem 6.12.

6.14. Two fuzzy relations are given as

$$\mathbf{R}_1 = \begin{bmatrix} 0.3 & 0 & 0.7 & 0.3 \\ 0 & 1 & 0.2 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0.5 & 0.4 \\ 0.7 & 0.9 & 0.6 \\ 0 & 0 & 0 \end{bmatrix}.$$

Find the composition of these two relations using

a. MAX-MIN composition,

b. MAX-PROD composition,

c. MAX-AVERAGE composition.

6.15. Find and present graphically the output fuzzy set for the system in figure P6.1 with two inputs (having two fuzzy sets per each input) and one output described by

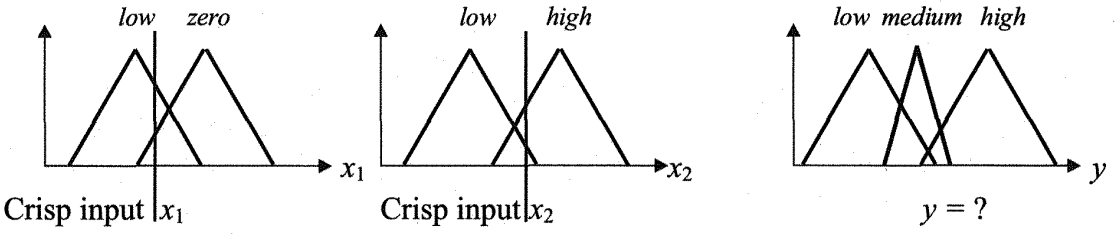


Figure P6.1
Graphs for problem 6.15.

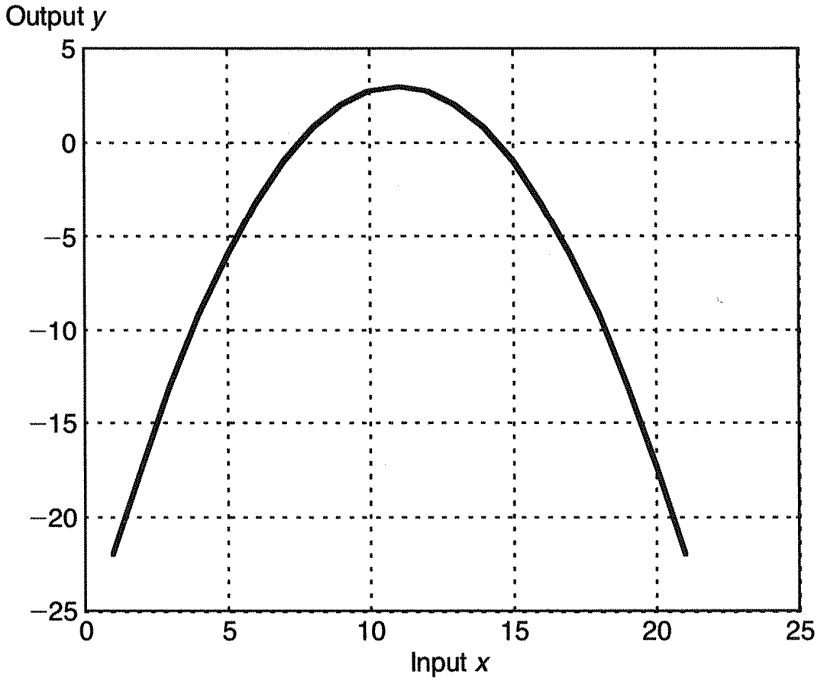


Figure P6.2
Graph for problem 6.16.

following four rules:

R_1: IF $x_1 = low$ AND $x_2 = low$, THEN $y = low$.

R_2: IF $x_1 = low$ AND $x_2 = high$, THEN $y = medium$.

R_3: IF $x_1 = zero$ AND $x_2 = low$, THEN $y = medium$.

R_4: IF $x_1 = zero$ OR $x_2 = high$, THEN $y = high$.

6.16. Figure P6.2 shows the functional dependency between two variables: $y = y(x)$. Make a fuzzy model of this function by using proper fuzzy tools and algorithms. In particular, use three membership functions for the Input x and three membership functions for the Output y. Choose the shapes and positions of the membership functions that you think can solve the problem. Make the corresponding rule base, find the relational matrices if needed, and for $x = 10$, using your fuzzy model, find the crisp value of y. Use any operator, inference rule, or defuzzification method you think is proper for modeling the given function.

6.17. The fuzzy controller is acting according to the following rule basis ($N = negative$, $M = medium$, $P = positive$):

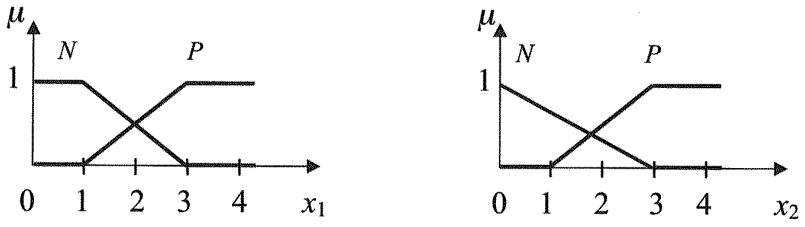


Figure P6.3
Input (antecedent) membership functions for problems 6.17 and 6.18.

R_1: IF x_1 is N AND x_2 is N, THEN u is N.

R_2: IF x_1 is N OR x_2 is P, THEN u is M.

R_3: IF x_1 is P OR x_2 is N, THEN u is M.

R_4: IF x_1 is P AND x_2 is P, THEN u is P.

The membership functions (possibility distributions) of the input variables are given in figure P6.3, and the membership functions of the output variable (which is a controller action) u are singletons placed at u is equal to 1, 2, and 3 for N, M, and P, respectively. Actual inputs are $x_1 = 2.5$ and $x_2 = 4$. Which rules are active, and what will be the controller action u? Find u by applying both the relational models (MAX-MIN or MAX-PROD) and the FAM. Comment whether there is any difference between the MAX-MIN and MAX-PROD here?

6.18. Consider a fuzzy controller acting according to the following rule basis ($N =$ *negative*, $M =$ *medium*, $P =$ *positive*):

R_1: IF x_1 is N OR x_2 is N, THEN u is N.

R_2: IF x_1 is N AND x_2 is P, THEN u is P.

R_3: IF x_1 is P AND x_2 is N, THEN u is P.

R_4: IF x_1 is P OR x_2 is P, THEN u is N.

The membership functions of the input variables are same as in problem 6.17 and are shown in figure P6.3. The membership functions of the output variable (which is a controller action) u are singletons placed at u is equal to 2 and 4 for N and P, respectively. Actual inputs are $x_1 = 2$ and $x_2 = 4$. Which rules are active, and what will be the controller action u? Find u by applying both the relational models (MAX-MIN or MAX-PROD) and the FAM.

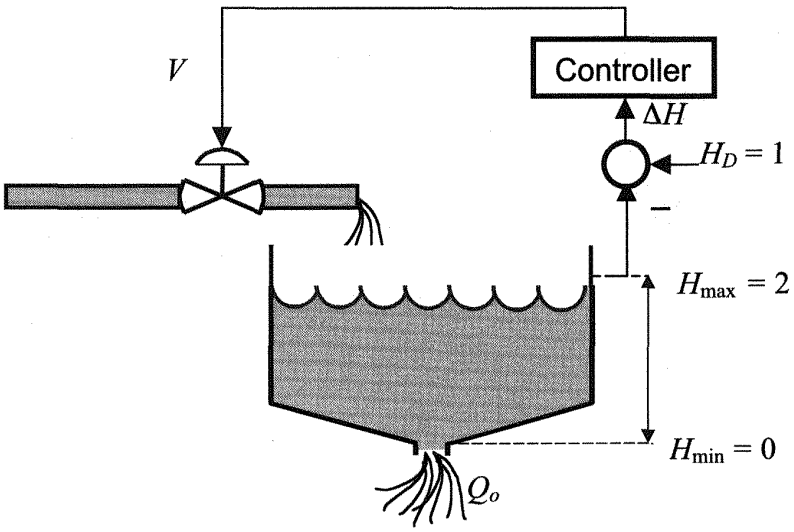


Figure P6.4
Plant scheme for problem 6.19.

6.19. Design the fuzzy controller for a water level control shown in figure P6.4 using three rules only. The input value to the controller is an actual water level perturbation ΔH (meters) $\in [-1, 1]$, and the controller output is the valve opening V (%) $\in [0, 100]$. For $\Delta H = -0.25$, calculate the actual valve opening V by using a FAM. (*Hint:* Follow the design steps in box 6.1.)

6.20. Equation (6.32) can be used for off-line (batch) learning of the output singletons' positions r_i having fixed input fuzzy subsets and data (the "activations" μ_i and the desired outputs y_{di}, namely, a matrix $\mathbf{A}$ and a vector $\mathbf{b}$, are known). Derive the on-line gradient descent error backpropagation adapting algorithm for the output singletons' positions r_i given by (6.33) when the error function is a sum of error squares $E = 1/2(y_d - y)^2$. (*Hint:* Start with $r_{iNew} = r_{iOld} - \eta \nabla_r E$, and find the gradient $\nabla_r E$.)

6.21. A Cauchy bell-shaped function may be a good candidate for an input membership function. In the case of an $\Re^1 \rightarrow \Re^1$ mapping, this function is given by

$$\mu_i(x) = \frac{1}{1 + \left(\frac{x - m_i}{d_i}\right)^2}.$$

It is placed at m_i and acts locally, and the area of activation is controlled by the width parameter d_i, which corresponds to the standard deviation at the Gaussian function.

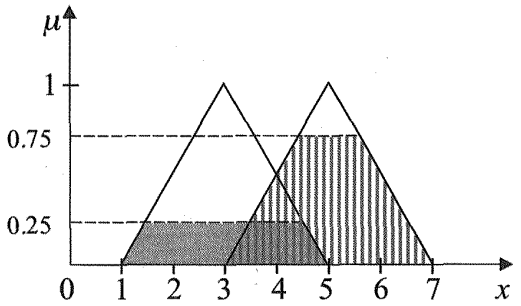


Figure P6.5
Graph for problem 6.23.

In addition, it is differentiable, and both the centers m_i and the width parameter d_i can be adapted by applying the error backpropagation (EBP) algorithm. (Note that m_i and d_i correspond to the hidden layer weights of neural networks). Derive the EBP learning laws for adapting m_i and d_i in a FAM with Cauchy membership functions. The error function is a sum of error squares $E = 1/2(y_d - y)^2$. (*Hint:* Start with the learning algorithm, e.g., for m_i with $m_{i(p+1)} = m_{ip} - \eta \nabla_{m_i} E$, and use the FAM model (6.33). This means that the output membership functions are singletons.)

6.22. Derive the gradient descent EBP learning laws for a FAM to adapt both the centers m_i and the width parameter d_i of the sinc membership function defined as

$$\mu_i(x) = \frac{\sin\left(\dfrac{x - m_i}{d_i}\right)}{\dfrac{x - m_i}{d_i}}.$$

The error function is a sum of error squares $E = 1/2(y_d - y)^2$. Use the FAM model (6.33).

6.23. Consequents (output membership functions) are given in figure P6.5. Find the crisp output y' by applying

a. center-of-gravity defuzzification method for a MAX-MIN inference,

b. FAM method,

c. height method for a MAX-MIN inference that calculates the crisp output as

$$y' = \frac{\sum_{i=1}^{N} c_i H_i}{\sum_{i=1}^{N} H_i},$$

where the c_i are the centers of gravity or means of the resulting rule consequents, and H_i are their maximal heights. N stands for the number of output membership functions. If the consequents are singletons, the preceding equation is equal to (6.16).

6.24. Approximate the two functions presented in figure P6.6 by fuzzy models. Choose the membership functions and type of fuzzy inference you think will work best. Make one rough (small number of rules) and one finer approximation for each function.

6.25. Design a fuzzy logic pattern recognition model for a classification of two letters V and U, shown in figure P6.7. First, make a class description and define only two features based on this description. These two features will be your two inputs to the fuzzy classifier. Then define the membership functions. Choose two membership functions for each input. Define the rules.

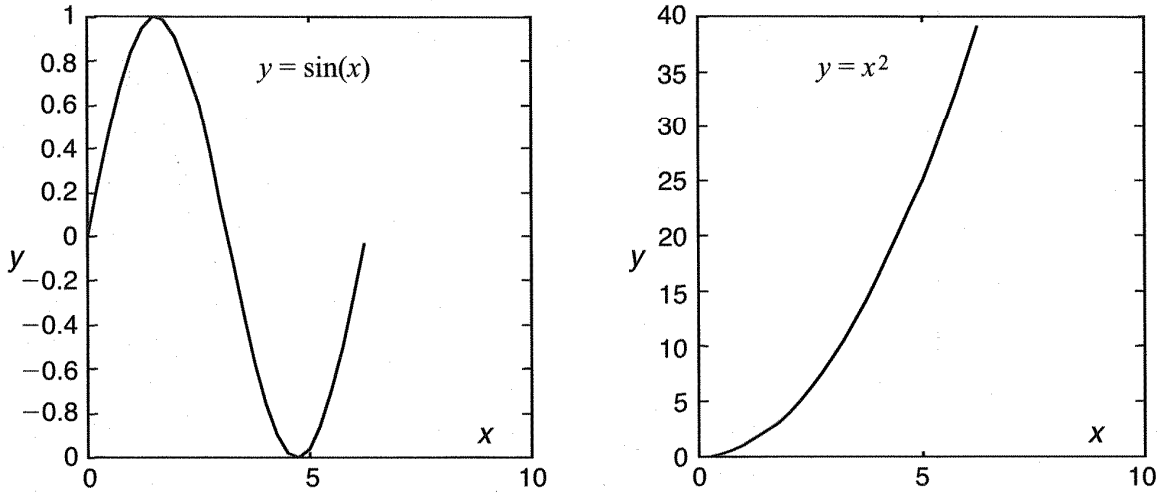


Figure P6.6
Graphs for problem 6.24.

Figure P6.7
Graphs for problem 6.25.

Simulation Experiments

The simulation experiments in chapter 6 have the purpose of familiarizing the reader with the fuzzy logic modeling tools. There are two programs for performing a variety of fuzzy modeling tasks. They can be found in two directories: `fuzzy1` and `fuzzy2`. In addition, there is a program `fuzfam` in `aproxim` file. Both programs were developed as the final-year projects at the University of Auckland under the supervision, guidance, and gentle cooperation of the author. (It is interesting to mention that the students had only a half-semester's introduction to fuzzy logic before commencing the final-year thesis.)

The `fuzzy1` program was created by D. Simunic and G. Taylor, and it was aimed at the application of fuzzy logic to a vehicle turning problem. The `fuzzy2` program was developed by W. M. Chen and G. Chua for guidance of mobile robots using fuzzy logic theory.

`Fuzzy1` can be used to develop other fuzzy logic models, whereas `fuzzy2` is merely a demo program for simulation of a given problem and cannot be used by the reader to create models. However, `fuzzy2` can be used to explore various aspects of FL modeling. Both programs have a nice graphic interface, and they are user-friendly. The user need only follow the pop-up menus and graphic windows.

You can perform various experiments aimed at reviewing many basic facets of fuzzy logic modeling, notably the influence of the membership functions' shape and overlap on the accuracy of model, the influence of the rule basis on model performance, and the effect of inference and defuzzification operators on the final modeling results.

Experiment with the programs `fuzzy1` and `fuzzy2` as follows:

1. Launch MATLAB.

2. Connect to directory `learnSC` (at the `matlab` prompt, type `cd learnsc` ⟨RETURN⟩). `learnSC` is a subdirectory of `matlab` as `bin`, `toolbox`, and `uitools` are. While typing `cd learnsc`, make sure that your working directory is `matlab`, and not `matlab/bin`, for example).

To start the program type `start` ⟨RETURN⟩. Pop-up menus will lead you through a design procedure. There are several options. You can either design your own fuzzy model or run one of several demo programs. It may be best to begin with the simplest *heating* demo. This is a model of how one controls the temperature in a room by changing the heat supplied.

Click to `file` — `open` — `heating.mat`. The input and output membership functions will be displayed. Click on `model` — `inference`, and you will see

surface of knowledge, or in the case of a one-dimensional input, curve of knowledge. By activating the slide bar, you can follow the fuzzy calculations. Active rules are shown by red bars over the corresponding output membership functions.

To see the effects of applying various inference and defuzzification mechanisms, go to options and select any of the given operators. Choose merely one change at time, that is, do not change both inference and defuzzification operators at the same time (unless you really want to). Analyze the change in the resulting curve of knowledge.

Note that all changes during the simulation should go through the pop-up menu. Hence, if you want to run another example, do not kill the existing window by clicking the x-corner button. Rather, click options — main menu, and begin a new simulation.

When you are done with the one-dimensional example, you may run the application of fuzzy logic to a vehicle turning problem. Select one of the demos starting with car**.mat, e.g., click file — open — cartes55.mat. Click model — animation for 2-D car, and drive the car around the corner from various initial positions. You can trace the car paths and keep the traces. Just try out various options of the program. Choose various operators, and keep the traces to compare them. Note that the car is not allowed to go backward, and this makes some initial positions impossible to solve, even for humans.

You can also repeat example 6.14 by selecting one of the two prepared demos, namely brake55.mat or brake35.mat. Choose some operators from options and analyze the surfaces of knowledge obtained.

Program fuzzy2 controls the movement of several mobile robots in a workshop. They service several machines and must avoid collision with each other.

Run several simulations, trying out different numbers of robots on the floor and different numbers of machines. Repeat the simulations with various inference and defuzzification operators. Carefully analyze the three-dimensional graphs of the surfaces of knowledge obtained.

There are small programming bugs in both routines. None is of crucial importance, but some do influence the performance of the fuzzy model created. This will be readily visible in following the trajectories of the mobile robots. Note that all robots have different, constant, and randomly chosen velocities. There will be odd solutions in the situations when the faster robot is closing the distance to the slower one. The very overtaking will be unusual because all robots are programmed to turn to the right only in order to avoid collision.

7 Case Studies

7.1 Neural Networks–Based Adaptive Control

This section focuses on neural networks–based adaptive control and also addresses the class of fuzzy logic models that are equivalent to neural networks (see section 6.2). In particular, after a review of the basic ideas of NN-based control, the *adaptive backthrough control* (ABC) scheme is introduced. ABC is one of the most serious candidates for the future control of the large class of nonlinear, partially known, time-varying systems. Recently, the area of NN-based control has been exhaustively investigated, and there are many different NN-based control methods. Rigorous comparisons show that NN-based controllers perform far better than well-established conventional alternatives when plant characteristics are poorly known (Bošković and Narendra 1995). A systematic classification of the different NN-based control structures is a formidable task (Agarwal 1997). Here, the focus is on an approach based on feedforward networks having static neurons, as given in figures 4.4 and 5.6. This section follows the presentation in Kecman (1997).

A standard control task and basic problem in controlling an unknown dynamic plant is to find the proper, or desired, control (actuation) value u_d as an input to the plant that would ensure

$$y(t) = y_d(t), \qquad \forall t, \tag{7.1}$$

where the subscript d stands for desired. $y(t)$ and $y_d(t)$ denote the plant output and desired (reference) plant output, respectively. The best controller would be one that could produce the value u_d that ensures (7.1), when the output of the plant exactly follows the desired trajectory y_d. In linear control, (7.1) will be ensured when

$$G_{ci}(s) = G_p^{-1}(s). \tag{7.2}$$

Hence, the ideal controller transfer function $G_{ci}(s)$ should be the inverse of the plant transfer function $G_p(s)$. Because of many practical constraints, this is an idealized control structure (Kecman 1988). However, one can try to get as close as possible to this ideal controller solution, $G_{ci}(s)$. The ABC approach, which is presented in section 7.1.4, can achieve a great deal (sometimes even nearly all) of this ideal controller. The block diagram of the ideal control of any nonlinear system is given in figure 7.1. $\mathbf{f}(\mathbf{u}, \mathbf{y})$ in the figure stands for any nonlinear mapping between an input $\mathbf{u}(t)$ and an output $\mathbf{y}(t)$. In the general case of a dynamic system, $\mathbf{f}(\mathbf{u}, \mathbf{y})$ represents a system of nonlinear differential equations. Here, the focus is primarily on discrete-time systems, and the model of the plant in the discrete-time domain is in the form of a nonlinear discrete equation $\mathbf{y}(k+1) = \mathbf{f}(\mathbf{u}(k), \mathbf{y}(k))$. Now, the basic problem is how to learn, or obtain, the inverse model of the unknown dynamic plant by using an NN.