2

Soft Computing Essentials

Andre de Korvin, Hong Lin, and Plamen Simeonov

Department of Computer and Mathematical Sciences, University of Houston, Downtown, One Main Street, Houston, Texas 77002, USA. dekorvina@uhd.edu

The main purpose of this chapter is to give an overview of some of the soft computing methods that are currently applied to a variety of problems. One of the characteristics of soft computing methods is that they are typically used in problems where mathematical models are not available or are intractable or too cumbersome to be viable. Another characteristics is that uncertainty inherent in many situations under study is taken into account rather than ignored. Often a human expert has partial knowledge and it is this partial knowledge (as opposed to complete knowledge) that soft computing uses to advantage. Another characteristic is that soft computing often provides a good solution as opposed to an optimal solution. In this chapter we present some of the mainstream methods of soft computing: Neural Nets, Fuzzy Logic, Neuro-Fuzzy Systems, The Theory of Evidence, Rough Sets, and Genetic Algorithms.

2.1 Neural Nets

A large number of structures have been used and we will highlight a small sample of the great variety of networks used. For a small fraction of the material available for neural nets we refer the reader to [6], [8], [11], [5], [9], [23], and [39]. A neural net is a set of neurons. Each neuron is as shown in Figure 2.1.

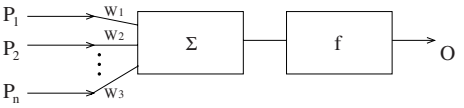

Fig. 2.1

An n-dimensional input $(p_1, \ldots, p_n)^T$ is provided. The first component of the neuron carries a weighted average called the net input:

C. Hu et al. (eds.), *Knowledge Processing with Interval and Soft Computing*,
DOI: 10.1007/978-1-84800-326-2_2,

$$N = \sum_{i=1}^{n} w_i p_i.$$

The second neuron transforms N into the output

$$O = f(N).$$

We refer to f as the transfer function. Commonly used transfer functions are

$$\begin{aligned} f(N) = \text{Hardlim}(N) &= \begin{cases} 1 \text{ if } N \geq 0 \\ 0 \text{ otherwise,} \end{cases} \\ f(N) = \text{Hardlims}(N) &= \begin{cases} 1 \text{ if } N \geq 0 \\ -1 \text{ otherwise,} \end{cases} \\ f(N) = \text{LogSig}(N) &= 1/(1+e^{-N}). \end{aligned}$$

Often an additional input of 1 is provided as well as an additional weight called the bias. The net input is then given by

$$N = \sum_{i=1}^{n} w_i p_i + b,$$

where b is the bias. The set of neurons forming a neural net is often organized into layers as shown for example in Figure 2.2.

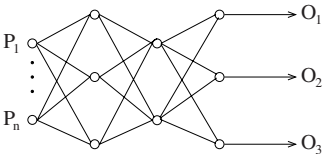

Fig. 2.2

In Figure 2.2 an n-dimensional input generates a 3-dimensional output. A commonly used notation for weights is $w^l_{i,j}$, which stands for the weight associated with the i-th neuron at layer l receiving input from the j-th neuron at layer $l-1$. So, for example, $w^2_{1,1}$, $w^2_{1,2}$, and $w^2_{1,3}$ would be the weights for the first neuron (i.e. the top one) at layer 2. A similar notation holds for the biases. Thus b^2_2 would be the bias of the second neuron (i.e., the bottom one) at layer 2. Under supervised learning, a set of inputs with desired targets is given $\{(p_1, t_1), \ldots, (p_n, t_n)\}$. Here $p_1, \ldots, p_n$ denote inputs (thus, p_i is the i-th input vector) and $t_1, \ldots, t_n$ denote the desired targets for these inputs. Initially the weights and biases are set somewhat arbitrarily (usually to small random numbers) and are then continuously adjusted in terms of the error produced. This process is called learning under supervision.

We now outline a small sample of existing neural net structures.

2.1.1 Adaline

The Adaptive Linear Element (or Adaline) is the simplest example of a neural net. It is shown in Figure 2.3.

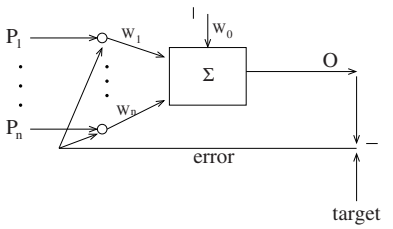

Fig. 2.3

In Figure 2.3 we have one neuron and we have denoted its bias by w_0. The transfer function is taken to be the identity function $O = f(N) = N$. Thus,

$$O = \sum_{i=1}^{n} w_i p_i + w_0.$$

For each input $(p_{1,q}, \ldots, p_{n,q})^T$ we specify a target t_q, $q = 1, \ldots, Q$. For the q-th input, the squared error is

$$E_q = (t_q - O_q)^2.$$

We would like to adjust the weights and the bias so as to minimize E_q. The partial derivatives are

$$\frac{\partial E_q}{\partial w_i} = -2(t_q - O_q)\frac{\partial O_q}{\partial w_i} = -2(t_q - O_q)p_{i,q}.$$

In order to minimize E_q we take small steps opposite ∇E_q (the gradient of E_q). If $w_1(0), \ldots, w_n(0)$, $w_0(0)$ are the initial values given to the weights and the bias, the update from step k to step $k+1$ is given by

$$w_i(k+1) = w_i(k) + \eta(t_q - O_q)p_{i,q}$$

provided input $\mathbf{p}_q = (p_{1,q}, \ldots, p_{n,q})^T$ is presented at step $k+1$. Another way to write this is

$$\triangle_q w_i(k) = \eta(t_q - O_q)p_{i,q}, \qquad 0 \le i \le n,$$

where $\triangle_q w_i(k)$ denotes the change of weight w_i at iteration $k+1$ when the q-th input is presented, $t_q - O_q$ is, of course, the error produced when the q-th input is presented, and η is a small positive number.

The Adaline is trained by presenting the input from the training set many times over and updating the weights as shown. Under certain conditions, it can be shown that the weights and the bias will converge to values so that specified inputs will produce outputs close to the specified targets.

2.1.2 Adaptive Nets

Adaptive nets generalize the concept of neural nets in the sense that no weights need to be involved in the computation of the output. Adaptive nets use a training set to update parameters of adaptive nodes. An example of an adaptive net is given in Figure 2.4.

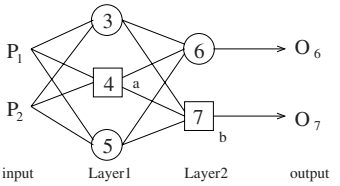

Fig. 2.4

Let $x_{i,j}$ denote the output produced by node j at layer i. The square nodes 4 and 7 produce an output depending on parameters a and b, respectively. The basic idea is to set a and b to some initial values and then use the training set to minimize the sum of squared errors. Let $N(i)$ denote the number of nodes at layer i. In Figure 2.4, $N(0) = 2$, $N(1) = 3$, and $N(2) = 2$. When input $\mathbf{p}$ is presented, the corresponding error function is

$$E_{\mathbf{p}} = \sum_{i=1}^{N(L)} (t_i^{\mathbf{p}} - x_{L,i}^{\mathbf{p}})^2.$$

Here L denotes the last layer, $t_i^{\mathbf{p}}$ denotes the specified target for $\mathbf{p}$ at neuron i of the last layer, and $x_{L,i}^{\mathbf{p}}$ denotes the output of that neuron. To simplify notation we now omit the superscript $\mathbf{p}$. Each sensitivity is defined as

$$\mathcal{E}_{l,i} = \frac{\partial E}{\partial x_{l,i}}.$$

We need to compute the gradient of E, ∇E, and then update the values of the parameters by adding to the current parameter vector $-\eta \nabla E$, where η is a small positive number. Let $f_{i,j}$ be the function giving the output for node $x_{i,j}$. In the example shown in Figure 2.4 we have

$$\begin{aligned}
&x_{1,3} = f_{1,3}(p_1, p_2), \quad x_{1,4} = f_{1,4}(a, p_1, p_2), \quad x_{1,5} = f_{1,5}(p_1, p_2),\\
&x_{2,6} = f_{2,6}(x_{1,3}, x_{1,4}, x_{1,5}), \quad x_{2,7} = f_{2,7}(b, x_{1,3}, x_{1,4}, x_{1,5}),\\
&O_6 = x_{2,6}, \quad O_7 = x_{2,7}.
\end{aligned}$$

Furthermore,

$$\nabla E = \left(\frac{\partial E}{\partial a}, \frac{\partial E}{\partial b} \right)^T$$

and

$$\mathcal{E}_{2,i} = -2(t_i - x_{2,i}),$$
$$\mathcal{E}_{1,i} = \frac{\partial E}{\partial x_{1,i}} = \sum_m \frac{\partial E}{\partial x_{2,m}} \frac{\partial x_{2,m}}{\partial x_{1,i}}.$$

So, for our example, we have

$$\mathcal{E}_{2,6} = -2(t_6 - x_{2,6}), \qquad \mathcal{E}_{2,7} = -2(t_7 - x_{2,7}),$$
$$\mathcal{E}_{1,i} = \mathcal{E}_{2,6} \frac{\partial f_{2,6}}{\partial x_{1,i}} + \mathcal{E}_{2,7} \frac{\partial f_{2,7}}{\partial x_{1,i}}, \qquad i = 3, 4, 5,$$
$$\mathcal{E}_{0,i} = \mathcal{E}_{1,3} \frac{\partial f_{1,3}}{\partial p_i} + \mathcal{E}_{1,4} \frac{\partial f_{1,4}}{\partial p_i} + \mathcal{E}_{1,5} \frac{\partial f_{1,5}}{\partial p_i}, \qquad i = 1, 2.$$

Thus,

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial x_{1,4}} \frac{\partial f_{1,4}}{\partial a} = \mathcal{E}_{1,4} \frac{\partial f_{1,4}}{\partial a}$$

and similarly

$$\frac{\partial E}{\partial b} = \mathcal{E}_{2,7} \frac{\partial f_{2,7}}{\partial b}.$$

This completes the computation of ∇E and, therefore, the necessary updates of a and b when input p is presented. This process is called backpropagation, as the computation of the sensitivities proceeds from the last layer to the first layer.

2.1.3 The Standard Backpropagation

The standard backpropagation typically refers to a special case of adaptive nets where the parameters are weights and biases and the output produced by neuron i at layer m, $0 \le n \le L$, is given by

$$a_i^m = f_m(n_i^m).$$

Here f_m refers to the transfer function at layer m (applied to each neuron at layer m) and n_i^m is the net input to neuron i at layer m. So,

$$n_i^m = \sum_j w_{i,j}^m a_j^{m-1} + b_i^m.$$

Here a_j^{m-1} refers to the output of neuron j at layer $m-1$, $w_{i,j}^m$ is the weight connecting neuron j at layer $m-1$ to neuron i at layer m, and b_i^m is the bias of neuron i at layer m. The sensitivities $\mathcal{E}_{l,i}$ defined in the previous subsection are now replaced by sensitivities defined as

$$\zeta_i^m = \frac{\partial \hat{E}}{\partial n_i^m},$$

where $\hat{E} = \mathbf{e}^T(k)\mathbf{e}(k)$ and $\mathbf{e}(k)$ is the error vector at layer L (i.e., the output layer) committed on trial k. Using the chain rule in a manner somewhat similar to that in Section 2.1.2, it can be shown that

$$\zeta_i^L = -2(t_i - a_i^L) f_L'(n_i^{L-1}),$$
$$\zeta_i^m = \sum_j w_{j,i}^{m+1} f_m'(n_i^m) \zeta_j^{m+1}.$$

Thus, as in the previous subsection the sensitivities ζ_i^{m+1}, $0 \leq m \leq L-1$, are computed starting from layer L and proceeding step by step to previous layers. The updates are given by

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \eta \zeta_i^m a_j^{m-1},$$
$$b_i^m(k+1) = b^m(k) - \eta \zeta_i^m.$$

As earlier, the inputs in the training set are presented over and over with appropriate updates given above. If the positive number η is small enough, the weights and biases will often converge (albeit somewhat slowly) to the specified targets.

Multilayers nets are very flexible. For example, it can be shown that a two-layer network can approximate any reasonable function where the transfer function for the first layer is LogSig and the transfer for the second layer is the identity function. This can be done with any degree of flexibility provided there are enough neurons in the first layer. Even if there are enough neurons, the net may not be able to approximate a given function if the weights and biases are poorly chosen. The reason the net may not converge is that the function to be minimized (i.e., the square of the error) would possibly be highly nonlinear and not quadratic and hence have many local minima. It is good practice to apply the algorithm outlined earlier starting with a number of different initial conditions. It is also important to choose the training set to be representative of the total dataset so that the net is able to successfully generalize the input-output relation to that total dataset. Also, it is worth pointing out that if too much flexibility is given to the net (i.e., too many neurons), the net may generalize in a way not intended by the designer.

Another problem is that if the learning factor η is taken to be too large, then convergence may fail. If η is taken too small, convergence may be too slow for many applications. Some of the methods to address this problem include the use of methods different from the steepest descent (e.g., the conjugate gradient or the Levenberg-Marquard algorithms). Other methods use heuristic techniques and we now mention a few of these. If the learning factor η is taken too large, the weights and biases may oscillate and fail to converge. To reduce the oscillations, one may use a first order filter. The filter works as follows: If at time k the input is $w(k)$, then the output is given by

$$y(k) = \gamma y(k-1) + (1-\gamma) w(k),$$

where $0 \leq \gamma \leq 1$. Note that $\sum y(k) = \gamma \sum y(k-1) + (1-\gamma) \sum w(k)$. Therefore, $E(y) = \gamma E(y) + (1-\gamma)E(w)$, that is

$$E(y) = E(w).$$

Thus, the average value of y is equal to the average value of w. Now, the variation of the weights and biases for the standard backpropagation is

$$\triangle w^m(k) = -\eta \zeta^m (a^{m-1})^T,$$
$$\triangle b^m(k) = -\eta \zeta^m.$$

Thus, at time k the variations are redefined as

$$\triangle w^m(k) = \gamma \triangle w^m(k-1) - (1-\gamma)\eta \zeta^m (a^{m-1})^T,$$
$$\triangle b^m(k) = \gamma \triangle b^m(k-1) - (1-\gamma)\eta \zeta^m.$$

The average values of the oscillations for the weights and the biases remain the same but the amplitude is reduced. In fact, if γ is close to 1, $\triangle w^m$ and $\triangle b^m$ change very little from time $k-1$ to time k.

Another heuristic technique to speed up convergence is to use a variable learning rule. For example, if the squared error (over the entire training set) increases by say more than 5%, then the update is discarded and the learning value η is multiplied by some number less than 1, say by 0.75, and the momentum γ is set to zero. If the squared error decreases after the weight update, then the update is accepted and the learning rule is multiplied by a number greater than 1, say by 1.5, and the momentum γ is reset to its original value. Finally, if the squared error increases by less than 5%, the weight update is accepted but the learning rule and the momentum are left unchanged. The motivation to accept an increase in error (less than 5%) is to be able to explore different values for weights that may potentially lead to a better value although initially the error is slightly higher. Multiplying the learning factor by 1.5 speeds up the updates. On the other hand multiplying the learning rule by 0.75 and setting the momentum to zero is mathematically expressing caution about moving on the current path to an optimal point. It should be pointed out that sometimes this approach fails to obtain convergence even though the standard approach gives convergence. Most of the time, however, this heuristic method yields faster convergence than the standard algorithm.

2.1.4 A Brief Overview of Additional Networks

In this subsection we present a brief overview of a few additional networks.

Counter-Propagation

The Counter-Propagation net is based on joining a Kohonen net with an outstar net. Such a net is sketched in Figure 2.5.

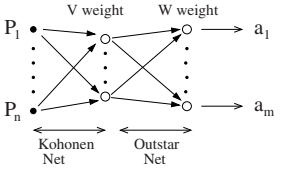

Fig. 2.5

Counter-Propagation sets up a map between n-dimensional vectors and m-dimensional vectors. The neurons at the Kohonen level are competitive. This means that the neuron that has the highest net input is the only neuron that fires and updates its weight. The other neurons have activation zero and do not update their weights. All inputs are first normalized so the highest net input corresponds to the cosine made by the input vector and the neuron's weight vector. Thus, the highest net input corresponds to the best match in direction between the input and the neuron's weight. If neuron i at the Kohonen level is the winner, its weight vector is updated at time k by

$$\mathbf{v}_i(k) = \mathbf{v}_i(k-1) + \eta(\mathbf{p} - \mathbf{v}_i(k-1))$$

if input $\mathbf{p}$ has been presented. The equilibrium position is $\mathbf{v}_i = \mathbf{p}$, or more accurately, $\mathbf{v}_i$ converges to the center of gravity of inputs similar to $\mathbf{p}$ (i.e., inputs for which the same neuron i is the winner). If O_j denotes the output of neuron j at the Kohonen level, then

$$O_j = \begin{cases} 1 \text{ if neuron } j \text{ is the winner} \\ 0 \text{ otherwise.} \end{cases}$$

The second net is the outstar net. The update of the weights at that level is given by

$$\triangle w_{i,j} = \eta'(t_i - w_{i,j})O_j,$$

where t_i represents the i-th component of the desired target. At equilibrium, $w_{i,j} = t_i$ if j is the winning neuron. Since the transfer function is linear (i.e., $f(N) = N$) and $O_j = 0$ except for the winning neuron where $O_j = 1$, the net input to neuron i is $w_{i,j}$. So, $a_i = t_i$ at equilibrium. The right target is obtained at equilibrium. For further information on Counter-Propagation we refer the reader to [9] and [39].

Adaptive Resonance Theory (ART)

We first discuss the outstar nets and instar nets and show how memory patterns might be accessed. The outstar net is shown in Figure 2.6.

As shown, the neuron $\mathbf{p}$ sends an output of 1 to all the neurons in the next layer. The weight vector is made equal to the specified output $(a_1, \ldots, a_n)^T$.

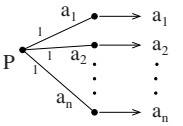

Fig. 2.6

The transfer function here is, of course, linear. Thus, when **p** is active, a specified output vector is emitted by the output layer. The instar net is shown in Figure 2.7.

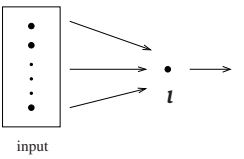

Fig. 2.7

Here, neuron l is only activated by certain inputs. The input **p** needs to be close enough to the weight vector of neuron l for neuron l to be activated. Often instars are structured in a competitive layer, as the following net shows. In order to access memory patterns a competitive layer is paired off with outstar structures, as shown in Figure 2.8.

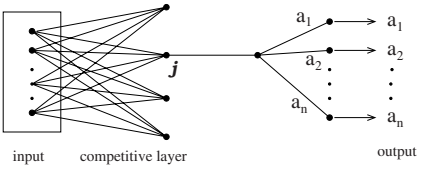

Fig. 2.8

Here, the vector input **p** is sent to the competitive layer. The competition works as follows: The only neuron to be activated is the one where the net input is the largest. The neuron is labeled j in Figure 2.8. Neuron j sends an output of 1 to the next layer, where the weight vector $(a_1, \ldots, a_n)^T$ is chosen to be the vector that one wishes to associate with **p**. This structure is thus able to build an association between vectors. At the end of the process neurons in the competitive layer will represent "clusters" (i.e., will be activated by vectors falling into a similar class). One could slightly change the philosophy of competition by having several winners (the weight of each winner determined

by the strength of match between its weight vector and the input vector). This then will allow the user "interpolation" to determine the output.

We now have the background to define the ART structure. Inputs can be binary or analog. For simplicity, we focus the discussion on binary inputs. The analog case is very similar in its broad features but technically more complex. One problem with the classical backpropagation algorithm is that once a net is trained, no incremental training is possible. If an additional (input, output) pair is to be added to the training set at a later time, the network must be retrained on the old training set with the new pair added. Using only the new pair will train the net on that pair only and hence render the net invalid for the initial training set. The ART structure is a complex dynamical structure and, for this reason, we focus on the functional aspects of ART rather than the mathematical machinery behind these functional aspects.

A common problem with nets using backpropagation is the stability or plasticity problem: When an input comes in, should it be classified as similar to previous inputs (i.e., does it belong to an established cluster or should a new cluster be initiated for that input)? The ART structure is sketched in Figure 2.9.

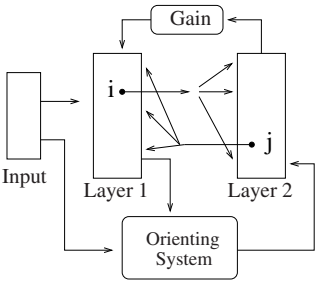

Fig. 2.9

Such a structure has limited capability to generalize from the training set. Because of its capability to perform incremental training, it is useful in spatio-temporal pattern recognition. The main idea is to use "resonance" between Layer 1, the input layer, and Layer 2, the output layer. Layer 1 receives and holds the input pattern. Layer 2 sends a response to Layer 1. If the response is similar to the input, then there is a match. If response does not match them, the two layers will resonate, seeking a match. If the input fails to match any pattern stored in the neurons of Layer 2 within a specified tolerance, a new stored pattern is formed. The orienting system disables the winning neuron (labeled j) in Layer 2 and a new competition takes place without the winner. A new match is attempted with the new winner. If the match does not take place within the described tolerance, the orienting system disables the new winner and a competition takes place without the two previous winners. If

a no-point tolerance is met, the input is then viewed as new and an unused neuron in Layer 2 stores its pattern. The connection from neurons in Layer 1 to a neuron of Layer 2 forms an instar net, with Layer 2 being a competitive layer. Competition here has a slightly different meaning, in that normalization of inputs does not take place. Some form of normalization is performed by Layer 1. The connections from a neuron in Layer 2 to neurons in Layer 1 form an outstar net. Denote by V the connections from Layer 1 to Layer 2 and by W the connections from Layer 2 to Layer 1. The match takes place within tolerance α if the following inequality holds:

$$\sum_i w_{j,i} a_i / \sum_i a_i > \alpha.$$

Here, a_i represents the activation of neuron i. The left-hand side of the inequality reflects how similar the (binary) input vector is to the response. If, for example,

$$(a_1, a_2, a_3, a_4) = (1, 0, 1, 1)$$

and

$$(w_{j,1}, w_{j,2}, w_{j,3}, w_{j,4}) = (0, 0, 1, 1)$$

then the left-hand side is 2/3 (i.e., two out of the three 1's of the input match the response). The gain unit outputs 1 if at least one component of the input is 1. Each neuron in Layer 1 has three inputs: the data input, the gain unit, and the response sent by j. For a neuron in Layer 1 to output 1, at least two of the three inputs must be 1 (the two-third rule). If any component in the response is 1, the gain unit is forced to 0. Thus, a neuron in Layer 1 will fire only if its input matches the response (both are 1). If the match test is successful, the input is then associated with the winning neuron in Layer 2. The W weights are updated. The V weights are similarly updated but with normalization

$$w_{i,j}(k+1) = w_{i,j}(k) a_i,$$
$$v_{j,i}(k+1) = \frac{L w_{i,j}(k) a_i}{L - 1 + \sum_k w_{k,j} a_k},$$

where L is some constant. Thus, $v_{j,i}$ is a scaled-down version of $w_{i,j}$. The reason $v_{j,i}$ are updated in a normalized way is to avoid normalizing the prototypes.

For further information on the ART structure, we refer the reader to [5] and [39].

Hebbian Learning

Hebbian learning is inspired by a natural law formulated by Donald Hebb:

> When the axon of a cell A is near enough to excite cell B and repeatedly takes part on firing it, some changes take place in one or both cells such that the efficiency of A firing B is increased. [A need not be the only cell involved in firing B.]

Another way to formulate the above law is as follows: If two neurons on either side of a synapsis are activated simultaneously, the strength of that synapsis increases. Consider the situation sketched in Figure 2.10.

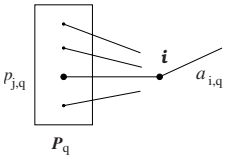

Fig. 2.10

Input $\mathbf{p}_q$ is presented at time $k+1$. Let $p_{j,q}$ denote the j-th component of $\mathbf{p}_q$, $1 \leq q \leq Q$. Let $a_{i,q}$ be the action of neuron i (located at the next layer). Then

$$w_{i,j}(k+1) = w_{i,j}(k) + \alpha a_{i,q} p_{j,q}.$$

Often, α is taken to be 1. Also, if we focus on supervised learning, $a_{i,q}$ is replaced by $t_{i,q}$ (the i-th component of the specified target $\mathbf{t}_q$ for input $\mathbf{p}_q$). The update is then

$$w_{i,j}(k+1) = w_{i,j}(k) + t_{i,q} p_{j,q}.$$

This is conveniently written in matrix form as

$$W = \sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T,$$

provided we assume that the initial values of $w_{i,j}$ are all 0 and we have gone through the cycle of presenting inputs $\mathbf{p}_1, \ldots, \mathbf{p}_Q$. Note that if the dimension of the input is r and the dimension of the target is s, then W is an $r \times s$ matrix. This in turn can be conveniently rewritten as

$$W = TP^T,$$

where T denotes a matrix whose columns are the target vectors $\mathbf{t}_1, \ldots, \mathbf{t}_Q$ and P denotes the matrix whose columns are the inputs vectors $\mathbf{p}_1, \ldots, \mathbf{p}_Q$. We consider here a linear associator. This means that if the input is $\mathbf{p}$, the resulting activation is $W_{\mathbf{p}}$. Assume that the inputs form an orthonormal set; then

$$\mathbf{a}_q = W_{\mathbf{p}_q} = \left(\sum_j \mathbf{t}_j \mathbf{p}_j^T \right) \mathbf{p}_q = \sum_j \mathbf{t}_j (\mathbf{p}_j^T \mathbf{p}_q).$$

Since all the terms $\mathbf{p}_j^T \mathbf{p}_q$ are 0 except when $j = q$, in which case the value is 1, we have

$$\mathbf{a}_q = \mathbf{t}_q.$$

In the case when the inputs form an orthonormal set, the linear associator produces an exact recall, and its activation matches the target. If the inputs are normalized (i.e., $||\mathbf{p}_q|| = 1$) but not orthogonal, then

$$\mathbf{a}_q = \mathbf{t}_q + \sum_{j \neq q} \mathbf{t}_j (\mathbf{p}_j^T \mathbf{p}_q).$$

An error in the recall takes place and that error depends on the amount of correlation between the inputs of the training set.

One way to produce a better recall is to use the pseudo-inverse method. If the number of rows of P is greater than the number of columns of P (i.e., the number of inputs in the training set is less the dimension of the inputs) and the inputs are linearly independent, then one seeks to minimize the squared error $||T - WP||^2$. It can be shown that under the above assumptions the minimum is given by

$$W = TP^+,$$

where P^+ denotes the pseudo-inverse of P, that is, $P^+ = (P^T P)^{-1} P^T$. Note that in the (very unlikely) case when P^{-1} exists then $P^+ = P^{-1}$. Note also that, in contrast to previous structures, no learning takes place. The weights are set to TP^T, or TP^+ if the pseudo-inverse method is required.

Particle Swarm Optimization (PSO)

An interesting methodology that, in particular, could be used to train neural nets was developed by Kennedy and Eberhart in 1995, [4, 16]. For applications using this approach, we refer to [4] and [16]. The PSO approach is similar to genetic algorithms (see a later section) in that the starting point is a population, more or less randomly selected of potential solutions. The difference, in contrast to genetic algorithms, comes in assigning random velocities to each member of the population. Each member of the population is called a particle. Particles are then moved toward a near optimal solution by evaluating the "fitness value" of each particle. In fact, a possible set of equations describing the dynamics of the solution is

$$v_{i,d}(\text{new}) = v_{i,d}(\text{old}) + c_1 \text{ rand}(\ )(p_{i,d} - x_{i,d}) + c_2 \text{ rand}(\ )(p_{g,d} - x_{i,d}),$$
$$x_{i,d}(\text{new}) = x_{i,d}(\text{old}) + v_{i,d}.$$

Here, x_i refers to the position of the i-th particle and $x_{i,d}$ refers to the d-th component of x_i. The variables $p_{i,d}$ and $p_{g,d}$ refer to the best position so far

occupied by the i-th particle and the global best position over all particles, respectively. The constants c_1 and c_2 have often been set to some values close to 2. The velocities in each direction are bounded by some constant $V_{\max}$. If $V_{\mathbf{max}}$ is high, particles may move past the optimal point. If $V_{\mathbf{max}}$ is too low, exploration becomes limited. The concept of best here refers to relative to some fitness function. The function rand( ) refers to a randomly generated number between 0 and 1. The algorithm then keeps track of the best for each particle and the global best, both of these, of course, updated as needed. The differences $p_{i,d} - x_{i,d}$ and $p_{g,d} - x_{i,d}$ are the distances of the current position of the particle i to its best and to the global best, respectively. The first previous update can be replaced by

$$v_{i,d} = w \times v_{i,d} + c_1 \text{ rand}()(p_{i,d} - x_{i,d}) + c_2 \text{rand}()(p_{g,d} - x_{i,d}),$$

where w denotes an inertia coefficient. With that new formulation, $V_{\max}$ can be taken to be the dynamic range of each variable (thus $V_{\max}$ here is a vector). To optimize the weights in a neural net, the variable $x_{i,d}$ consists of all current values for all weights and biases. Additional structure can be incorporated. For example, if the transfer function is $\text{LogSig}(N) = 1/(1 + e^{-cN})$, then the parameter c can be thrown in as one component of the variables $x_{i,d}$. The fitness function could be the sum of squared errors computed over some fixed training set.

2.1.5 Conclusion

We have presented a small sample of some of the important types of neural nets whose weights are determined by a training set. Once a network is trained, given some input that is not part of the training set, the system produces an output. The system is a black box. In general, it is difficult to know what exactly the net has learned. An important property is that neural nets have the capability to adapt. The weights change so that the system gradually learns to reproduce the specified input-output pairs. It is important to choose the right training set so that the net is able to generalize to the whole dataset.

2.2 Fuzzy Logic

Let X denote the universal set. The concept of a "fuzzy subset of X" is a generalization of subset of X. Let A be any subset of X. The subset A can be identified with its characteristic function, which we will often denote by the same symbol A. Thus, the subset A viewed as a characteristic function is defined by

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

The values taken by A are either 1 or 0. If $A(x) = 1$, then x belongs to A, and if $A(x) = 0$, then x does not belong to A. Thus, A viewed as a characteristic function is "a membership function" since the value of A at x indicates if x is a member of A or not.

We define A to be a fuzzy subset of X if A is a function defined on X that takes values in $[0, 1]$. For example, $A(x) = 0.25$ indicates that the membership of x in A is 0.25 - intuitively x does not belong very much to A. The value $A(x)$ can be interpreted as "on a scale of 0 to 1, how much x fits A." Often, fuzzy sets have a linguistic interpretation. In Figure 2.11 we show an example of how the set "Around 10" might be defined.

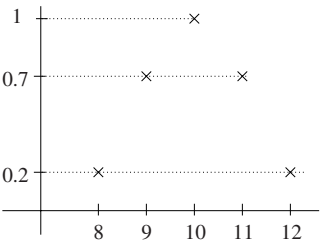

Fig. 2.11

Here, the universal set, often called the "universe of discourse," is $X = \{6, \ldots, 12\}$. The membership of 10 in that set is 1 (i.e., 10 fits perfectly "Around 10"). The memberships of 9 and 11 fit 0.7 the concept of "Around 10." The memberships of $6, 7, 13$, and 14 are 0; these numbers are too far away from 10 to fit the concept of "Around 10." Similarly Figures 2.12, 2.13, and 2.14 could represent "Young," "Old," and "Few," respectively.

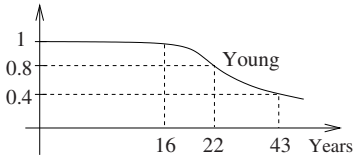

Fig. 2.12

If the universe of discourse X is finite and A is a fuzzy subset of X, the notation

$$A = \sum_i \alpha_i / x_i, \qquad \alpha_i \in [0, 1],$$

signifies that A is the supremum of functions f_i such that $f_i(x_i) = \alpha_i$ and $f_i(x_j) = 0$ if $j \neq i$. In particular, if all the x_i are distinct, $A(x_i) = \alpha_i$. If $x_i = x$ for several distinct values of i, then

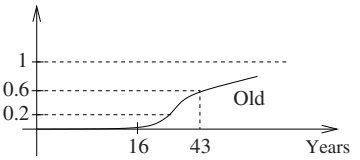

Fig. 2.13

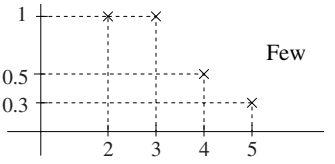

Fig. 2.14

$$A(x) = \sup\{\alpha_i \,|\, x_i = x\}.$$

For example $0.7/x_1 + 0.3/x_1 + 0.5/x_2 + 0.9/x_2 = 0.7/x_1 + 0.9/x_2$. For the continuous case, notation such as

$$A = \int_{x\in[5,10]} e^{-3(x-1)^2}/x$$

signifies that $X = [5, 10]$ and that for each $x \in X$, $A(x) = e^{-3(x-1)^2}$. In the notation $A = \sum_i \alpha_i/x_i$, if $x_j \in X$ and $x_j \neq x_i$ for all i, it is assumed that $A(x_j) = 0$.

2.2.1 Operations of Fuzzy Sets

If A and B denote two subsets of X, then the characteristic functions of $A \cap B$, $A \cup B$, and $\bar{A}$ (the complement of A) are given by

$$(A \cap B)(x) = A(x) \wedge B(x), \qquad (A \cup B)(x) = A(x) \vee B(x),$$
$$\bar{A}(x) = 1 - A(x).$$

Here, $\wedge$ and $\vee$ denote the operations Min and Max, respectively. This definition extends without any change to the case where A and B are fuzzy subsets of X, for example,

$$(0.3/x_1 + 0.8/x_2) \cap (0.4/x_1 + 0.7/x_2) = 0.3/x_1 + 0.7/x_2$$

and

$$(0.3/x_1 + 0.8/x_2) \cup (0.4/x_1 + 0.7/x_2) = 0.4/x_1 + 0.8/x_2.$$

These operations on fuzzy subsets can be generalized by introducing functions from $[0, 1]^2$ into $[0, 1]$ called t-norm and s-norm. For additional information on t-norms and s-norms, see [43].

A t-norm is a function from $[0,1]^2$ into $[0,1]$ satisfying the following:
1. $t(x,y) = t(y,x)$,
2. $t(t(x,y),z) = t(x,t(y,z))$,
3. t is monotone and nondecreasing,
4. $t(0,x) = 0$, $t(1,x) = x$.

Clearly, $\wedge$ satisfies the four properties. An s-norm is a function from $[0,1]^2$ into $[0,1]$ satisfying items 1 to 3 and 5. $s(x,0) = x$, $s(1,x) = 1$. Clearly, $\vee$ satisfies these four properties. Thus, a more general definition of intersection and union of fuzzy sets could be

$$(A \cap B)(x) = t(A(x), B(x)), \qquad (A \cup B)(x) = s(A(x), B(x)),$$

where t and s are arbitrary t-norm and s-norm. There are many examples of t-norms and s-norms, we list a few:

$$t(x,y) = xy, \qquad t(x,y) = 1/(1/x^p + 1/y^p - 1)^{1/p}$$

and

$$s(x,y) = x + y - xy, \qquad s(x,y) = 1 - 1/((1-x)^{-p} + (1-y)^{-p} - 1)^{1/p}.$$

Once a t-norm (or s-norm) is defined, the dual s-norm (or dual t-norm) can be constructed by $s(x,y) = 1 - t(1-x, 1-y)$ (or $t(x,y) = 1 - s(1-x, 1-y)$). If dual norms are used, then DeMorgan's laws hold for fuzzy subsets (i.e., $\overline{(A \cup B)} = \bar{A} \cap \bar{B}$ and $\overline{(A \cap B)} = \bar{A} \cup \bar{B}$). Just as intersections and unions can be generalized using the t and s-norms, the complement can be generalized using the negation operator. A negation operator is a nonincreasing map from $[0,1]$ to $[0,1]$ with the additional properties

$$N(0) = 1, \qquad N(1) = 0.$$

Clearly, $N(x) = 1 - x$ is a special case. Other examples could be

$$N(x) = \frac{1-x}{1+\underline{a}x}, \qquad \underline{a} > -1,$$

or

$$N(x) = (1 + x^w)^{1/w}, \qquad w > 0.$$

2.2.2 Construction of Membership Functions

In applications it is important to be able to define fairly well the membership functions corresponding to linguistic terms. In a given situation what do terms such as Rich, Poor, Middle Age, ..., etc. mean?

Polling Experts

Perhaps the simplest way to define a membership function is to poll experts. For example, suppose we have 10 experts on aging questions such as "Is 30 middle age?" "Is 40 middle age?" "Is 60 middle age?" The answer to be given by the experts should be Yes or No. The number of Yes answers is counted. If all 10 experts classified 40 as middle age, then the membership of 40 in middle age would be 1. If 6 experts classified 60 as middle age, then the membership of 60 in middle age would be 0.6.

Pointwise Comparison

To determine the membership of each element of the universe of discourse in some linguistic set A, one compares all pairs (x_i, x_j) of the universe of discourse and forms the ratios

$$r_{i,j} = A(x_i)/A(x_j).$$

Thus, if $r_{i,j} = 2$, then the expert has determined that x_i fits twice as well the concept of A than x_j does. Clearly, $r_{i,k} = r_{i,j}r_{j,k}$ and $r_{j,i} = 1/r_{i,j}$. Also, $r_{i,i} = 1$. It can easily be checked that if P denotes the matrix whose entries are $r_{i,j}$ and if $1 \leq i, j \leq n$, then

$$P\mathbf{a} = n\mathbf{a}, \qquad \mathbf{a} = (A(x_1), \ldots, A(x_n))^T.$$

If $\mathbf{a}$ is normalized so that $\sum_i A(x_i) = 1$, then

$$A(x_j) = 1/\sum_i p_{i,j}$$

and this determines the membership function of A. For an extensive discussion of this type of methodology, we refer the reader to [37] and [38].

Parametrized Membership Functions

Often the general form of a membership function is known. In this case, one uses data to determine the parameters. The membership function is of the form $A(\alpha, \boldsymbol{e}, \ldots, \mathbf{x})$, where $\alpha, \boldsymbol{e}, \ldots$ are parameters and $\mathbf{x}$ is a vector. Then one seeks to minimize

$$\sum_{i=1}^{n} (A(\alpha, \boldsymbol{e}, \ldots, \mathbf{x}_i) - y_i)^2$$

as a function of the parameters $\alpha, \boldsymbol{e}, \ldots$. If one specifies that $A(\alpha, \boldsymbol{e}, \ldots, x_i) = y_i$ for all i, then, of course, in most cases an analytical solution cannot be obtained and methods such as the steepest descent are used to get an approximate solution. In order to avoid landing on a local minimum, different initial values for $\alpha, \boldsymbol{e}, \ldots$ are set.

Using Neural Nets

If the value of the membership function is specified at certain points, then this defines a training set and, for example, backpropagation as described earlier could be used. Given a value x in the universe of discourse, the output provided by the net will be interpreted as $A(x)$.

We now list some of the standard membership functions. The triangular function is defined as

$$A(x) = \begin{cases} 0 & \text{if } x \le a \\ (x-a)/(m-a) & \text{if } a \le x \le m \\ (b-x)/(b-m) & \text{if } m \le x \le b \\ 0 & \text{if } x \ge b. \end{cases}$$

The triangular function conveys the idea of "around m."

The S-function is defined as

$$A(x) = \begin{cases} 0 & \text{if } x \le a \\ 2[(x-a)/(b-a)]^2 & \text{if } a \le x \le (a+b)/2 \\ 1 - 2[(x-b)/(b-a)]^2 & \text{if } (a+b)/2 \le x \le b \\ 1 & \text{if } x > b. \end{cases}$$

The S-function is shown in Figure 2.15 and it is used for concepts such as Old, Rich, Heavy, Large, ..., etc. The point at which $A(x) = 0.5$ is $(a+b)/2$.

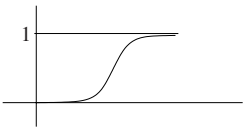

Fig. 2.15

The Gaussian function is defined as

$$A(x) = e^{-c(x-m)^2} \qquad \text{with } c > 0.$$

2.2.3 Fuzzy Relations

Fuzzy relations play a key role in fuzzy inference schemes. The extension from the standard case is very straightforward.

If X and Y denote two universes of discourse, a relation R from X to Y is simply a function from $X \times Y$ into $[0, 1]$. Thus, R is simply a fuzzy subset of $X \times Y$. If X and Y are finite, then R can be represented by a matrix whose entries are $R(x_i, y_j)$. Crucial for establishing the fuzzy inference procedure is the sup-min composition. If R_1 is a fuzzy relation from X into Y and R_2 is a fuzzy relation from Y into Z, then we define

$$R_1 \circ R_2(x,z) = \sup_{y \in Y} R_1(x,y) \wedge R_2(y,z).$$

Thus, $R_1 \circ R_2$ is a relation from X into Z, and if R_1 and R_2 are represented by matrices, then the matrix of $R_2 \circ R_1$ is obtained by "multiplying" the matrix of R_1 by the matrix of R_2 where the product is replaced by $\wedge$ and the sum is replaced by taking the supremum. For example,

$$\begin{bmatrix} 0.9 & 0.1 \\ 0.6 & 0.2 \\ 0.4 & 0.5 \end{bmatrix} \circ \begin{bmatrix} 0.7 & 0.4 \\ 0.2 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.4 \\ 0.6 & 0.4 \\ 0.4 & 0.5 \end{bmatrix}.$$

Similarly, if A is a fuzzy set subset of X and R is a fuzzy relation from X to Y, then $A \circ R$ is defined by

$$A \circ R(y) = \sup_{x \in X} A(x) \wedge R(x,y).$$

Of course, $\wedge$ and sup could be replaced by any t-norm and any s-norm, respectively, to obtain a more general operation.

We now proceed to define the implication relation. In general, if A is a fuzzy subset of X and B is a fuzzy subset of Y, $A \to B$ should be defined as a relation from X into Y:

$$(A \to B)(x,y) = f(A(x), B(y)),$$

where f is a function satisfying certain conditions that define an "implication function." Some of the common choices for f are

$$f(x,y) = x \wedge y, \qquad f(x,y) = xy, \qquad f(x,y) = (1-x) \vee y,$$

and

$$f(x,y) = \sup\{c \in [0,1] \,|\, x \wedge c \leq y\}, \qquad x, y \in [0,1].$$

It is easy to see that if we take A and B to be standard subsets of X and Y, then the first two examples of implication functions imply

$(A \to B)(x,y) = 1$ if and only if $x \in A$ and $y \in B$.

Thus, in the first two cases, $(A \to B) = A \times B$. In the third example, it is easy to see that $(A \to B) = \bar{A} \vee B$ (which is the standard definition of implication) and in the last example. In the third example where $f(x,y) = (1-x) \vee y$, if one takes $X = Y$ and if A and B are crisp subsets of X, then $(A \to B)$ becomes identical to $A \subset B$. For work dealing with fuzzy relations, see [34] and [1].

2.2.4 Fuzzy Reasoning

If we know that the variable U is some fuzzy subset A' of X and if we know that the rule: If U is A then V is B, where A and B are fuzzy subsets of X

and Y holds, then what inference can be made about V? The answer is to use the sup-min composition:

$$V \text{ is } A' \circ (A \to B).$$

To have an intuitive idea why this works, let A and B be standard sets and let $A' = A$. Then the above formula becomes

$$V(y) = \sup_x A(x) \wedge A(x) \wedge B(y) = B(y).$$

So we obtain the standard implication: U is A and (if U is A then V is B) implies V is B. In particular, if U is a number a, then V is

$$V(y) = \sup_x X_a(x) \wedge A(x) \wedge B(y) = A(a) \wedge B(y).$$

Here, X_a denotes a function defined on X that is 1 if $x = a$ and 0 otherwise.

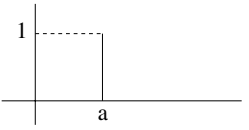

Fig. 2.16a

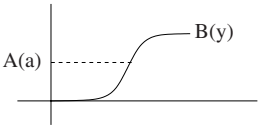

Fig. 2.16b

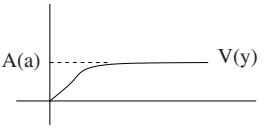

Fig. 2.16c

Figure 2.16a shows U identified with X_a. Figure 2.16b shows the membership function of B. Fig. 2.16.c shows the membership of V when U is a and the presence of the rule: if U is A then V is B.

Similar conclusions are drawn with different choices of the implication function. In the above example, $A(a)$ denotes the strength of the rule given the input $U = a$. If the input is $U = A'$, where A' is a fuzzy subset of X, then

$$V(y) = \sup_x A'(x) \wedge A(x) \wedge B(y).$$

The quantity $\sup_x A'(x) \wedge A(x)$ is called the possibility of A' and A and is denoted by $\text{Poss}(A, A')$. So, $V(y) = \text{Poss}(A, A') \wedge B(y)$. In this case, the strength of the rule (relative to the input U is A') is $\text{Poss}(A, A')$. It should be noted that $\text{Poss}(A, A')$ represents "the largest intersection of A and A'." Of course, the rule may have multiple antecedents and consequents such as:

If U_1 is A_1 and U_2 is A_2 and U_3 is A_3, then V_1 is B_1 and V_2 is B_2.

The input for this rule is U_1 is A'_1, U_2 is A'_2, U_3 is A'_3. The inferred fuzzy set is

$$W(y_1, y_2) = \text{Poss}(A_1, A'_1) \wedge \text{Poss}(A_2, A'_2) \wedge \text{Poss}(A_3, A'_3) \wedge B_1(y_1) \wedge B_2(y_2).$$

It specifies the membership of an arbitrary pair (y_1, y_2) in the fuzzy output W. In the more general case, any legal implication function is used and intersection is replaced by an arbitrary t-norm. Suppose that we have the input

U_1 is A'_1, U_2 is A'_2,..., U_n is A'_n

and the rule

If U_1 is A_1 and U_2 is A_2 ...and U_n is A_n, then V_1 is B_1 and V_2 is B_2 ... and V_m is B_m.

The multiplicative input is defined by

$$P_i(x_1, \ldots, x_n) = t[A'_1(x_1), \ldots, A'_n(x_n)],$$

the multiplicative antecedent is defined by

$$P_a(x_1, \ldots, x_n) = t[A_1(x_1), \ldots, A_n(x_n)],$$

the multiplicative consequent is defined by

$$P_c(y_1, \ldots, y_m) = t[B_1(y_1), \ldots, B_m(y_m)],$$

and

$$R(x_1, \ldots, x_n; y_1, \ldots, y_m) = f(P_a(x_1, \ldots, x_n), P_c(y_1, \ldots, y_m))$$

defines the fuzzy relation generated by the rule. The output under these conditions is then $W = P_i \circ R$; that is,

$$W(y_1, \ldots, y_m) = \sup_{x_1, \ldots, x_n} P_i(x_1, \ldots, x_n) t R(x_1, \ldots, x_n; y_1, \ldots, y_m).$$

In summary, given an if-then rule with fuzzy antecedent and fuzzy consequent, this rule defines a fuzzy relation R. Given then a fuzzy input, the rule produces a fuzzy output obtained by applying the sup-min composition to the

(input, rule) pair. Consider the following example:

Rule: If the Pressure is High, then the Volume is Small.

Input: Pressure is Average.

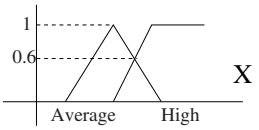

Fig. 2.17

Figure 2.17 shows the memberships of the fuzzy sets High and Average. In this example, $\text{Poss}(\text{Average}, \text{High}) = 0.6$ since this is the largest intersection of Average and High. Figure 2.18 shows the membership function of the fuzzy set $(W(y) = 0.6) \wedge \text{Small}(y)$ if the implication function is $f(x, y) = x \wedge y$ or $(W(y) = 0.6)\text{Small}(y)$ if the implication function is $f(x, y) = xy$.

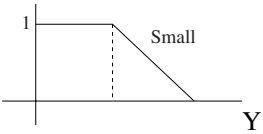

Fig. 2.18

Note that any $y \in Y$ will belong less to the output W than it belongs to Small. Intuitively, Small is implied by High is the only rule we have and the pressure does not match High. This example points to two items that need to be addressed:

1. The output produced is fuzzy. In many situations one needs a single number as an output (e.g., How many cm^3 does the Volume occupy?). This will be addressed in a later section; see defuzzification.

2. It is clear that more than one rule is needed since any single rule is only partially applicable to a given input.

We now assume that we have a set of rules of the form

R_k: If U is A_k, then V is B_k, $\qquad k = 1, \ldots, N$.

The fuzzy input is U is A'. The sets $A_1, \ldots, A_N$, $B_1, \ldots, B_N$ are in general fuzzy. Each rule produces a fuzzy output as described above. Thus, we have $W_1, \ldots, W_N$, the outputs produced by the N rules. The answer is obtained by aggregating these N outputs:

$$W(y) = \mathcal{A}_{k=1}^{N} W_k(y),$$

where $\mathcal{A}$ denotes an aggregation operation. By an aggregation operation we mean a map $\mathcal{A}$ from $[0,1]^p$ into $[0,1]$, with p is a positive integer - that is, nondecreasing in each of its p variables and

$$\mathcal{A}(0,\ldots,0) = 0, \qquad \mathcal{A}(1,\ldots,1) = 1.$$

An often used aggregation operation is the V operation. Thus, we may define

$$W(y) = V_{k=1}^{N} W_k(y).$$

This may be generalized to multiple antecedents and consequents. Using notations previously defined in this section and generalizing intersection and the V operations to t-norms and s-norms, respectively, we have

$$\begin{aligned}
&P_i(x_1,\ldots,x_n) = t[A'_1(x_1),\ldots,A'_n(x_n)],\\
&P_{a,k}(x_1,\ldots,x_n) = t[A_{1,k}(x_1),\ldots,A_{n,k}(x_n)],\\
&P_{c,k}(y_1,\ldots,y_m) = t[B_{1,k}(y_1),\ldots,B_{m,k}(y_m)],\\
&R_k(x_1,\ldots,x_n;y_1,\ldots,y_m) = f[P_{a,k}(x_1,\ldots,x_n), P_{c,k}(y_1,\ldots,y_m)],
\end{aligned}$$

and then

$$W = P_i \circ R, \quad \text{where} \quad R = \mathcal{A}_{k=1}^{N}.$$

An alternative way to combine the N rules is

$$W_k = P_i \circ R_k \quad \text{and} \quad W = \mathcal{A}_{k=1}^{N} W_k.$$

Thus, the two ways to aggregate are

$$P_i \circ \mathcal{A}_{k=1}^{N} R_k \quad \text{and} \quad \mathcal{A}_{k=1}^{N}(P_i \circ R_k).$$

In general, these two ways generate different outputs. The outputs will coincide if $\vee$ is taken for aggregation and $\wedge$ is taken for t-norm. Consider the following two rules:

If Pressure is High then Volume is Small,

If Pressure is Low then Volume is Large,

and say the input is Pressure is Average. Assume $\text{Poss}(\text{High}, \text{Average}) = 0.6$ and $\text{Poss}(\text{Low}, \text{Average}) = 0.3$. Then $W_1(y) = 0.6\text{Small}(y)$, $W_2(y) = 0.3\text{Large}(y)$, where the implication function was chosen to be the product, and $R_1(x,y) = \text{High}(x)\text{Small}(y)$, $R_2(x,y) = \text{Low}(x)\text{Large}(y)$, and $R(x,y) = R_1(x,y) \vee R_2(x,y)$.

Much work has been published on "approximate reasoning", examples of which were sketched above. For further reading on approximate reasoning and related matters see [27], [26], [44], [21], [22], [34], and [36].

2.2.5 Other Rules

Although the standard if-then rules are the most commonly used, at times different types of rules are needed. In this section we list a small sample of alternate rules. The first rule we list addresses the problem of the rule working most of the time except when some condition occurs. The general form is

If U is A, then V is B unless W is C.

The input to this rule should be of the form (U, V) is D, where D is a fuzzy subset of $Y \times Z$, Y is the universe of discourse for U, and Z is the universe of discourse for W. The above rule can then be replaced by the following two rules:

(i) If U is A and W is not C, then V is B.

(ii) If U is A and W is C, then V is not B.

Rule (i) is the default case. If W is not C, the rule must hold; that is, if U is A then V is B holds. Rule (ii) expresses the exception i.e. if U is A then we do not want V to be B because W is C. Each of these rules generates a fuzzy relation. That relation will depend on the implication function chosen. Suppose we pick

$$f(x, y) = \begin{cases} 1 \text{ if } x \leq y \\ y \text{ otherwise.} \end{cases}$$

Then the relations generated by rules (i) and (ii) are

$$R_1(x, y, z) = \begin{cases} 1 & \text{if } A(x) \wedge \bar{C}(z) \leq B(y) \\ B(y) & \text{otherwise} \end{cases}$$

and

$$R_2(x, y, z) = \begin{cases} 1 & \text{if } A(x) \wedge C(z) \leq \bar{B}(y) \\ \bar{B}(y) & \text{otherwise.} \end{cases}$$

The relation generated by the two rules is

$$R(x, y, z) = R_1(x, y, z) \wedge R_2(x, y, z).$$

The output generated by the unless rule is

$$V(y) = \sup_{x,z} D(x, z) \wedge R(x, y, z).$$

In particular, if the input is numerical, that is,

$$D(x, z) = \begin{cases} 1 \text{ if } x = x^* \text{ and } z = z^* \\ 0 \text{ otherwise,} \end{cases}$$

it is easy to see that

$$R(x,y,z) = \begin{cases} B(y) \text{ if } B(y) < A(x^*) \wedge \bar{C}(z^*) \text{ and } A(x^*) \wedge C(z^*) \leq \bar{B}(y) \\ \bar{B}(y) \text{ if } A(x^*) \wedge \bar{C}(z^*) \leq B(y) \text{ and } A(x^*) \wedge C(z^*) > \bar{B}(y). \end{cases}$$

Otherwise, $R(x,y,z) = 1$. Intuitively, this expresses the idea that V under different conditions behaves as B or $\bar{B}$.

Another type of rule is a rule that expresses uncertainty about the rule itself. One form of this is

If U is A, then V is B with certainty $\underline{a}$.

Here, $\underline{a} \in [0,1]$. If $\underline{a} = 1$, we are certain about the rule. If $\underline{a} = 0$, we are totally uncertain about the rule. The fuzzy relation generated by this rule is

$$R^*(x,y) = [R(x,y) \wedge \underline{a}] + (1 - \underline{a}).$$

Here, R denotes the relation generated by the rule: If U is A, then V is B. So, $R(x,y) = f[A(x), B(y)]$, where f is the selected implication function. Note that if $\underline{a} = 1$, $R^*(x,y) = R(x,y)$. If, on the other hand, $\underline{a} = 0$, then $R^*(x,y) = 1$. In that case, $V(y) = \sup_x A'(x) \wedge R^*(x,y) = 1$ if we assume that the fuzzy input A' is normal (i.e., $\sup_x A'(x) = 1$). This implies $V = Y$, which reflects total uncertainty, as we are not able to pin down what elements or Y are more or less believed to be in the output V. Intuitively, every element of Y is equally possible as an output in a case of total uncertainty.

Another type of uncertainty rule is one that is truth-qualified. The general form is

If U is A, then V is B is S.

Here, S stands for a fuzzy set representing a truth qualification. An example is If U is A, then V is B is very true. In this context, true is a fuzzy subset of $[0,1]$, whose membership is $\text{true}(x) = x$. Very true might have a membership function like $\text{Verytrue}(x) = x^2$, and Somewhat true might have a membership function like $\text{Somewhattrue}(x) = \sqrt{x}$. The statement If U is A, then V is B is S generates a fuzzy relation R^* defined by

$$R^*(x,y) = S(R(x,y)),$$

where R is the relation defined by the statement If U is A, then V is B. Note that if S = true, then $S(R(x,y)) = R(x,y)$, so $R^* = R$ and the two statements: If U is A then V is B is true and If U is A, then V is B are equivalent statements.

The next type of statement involving uncertainty that we discuss in this section has the form

If Probability(U is A) is P then V is B.

Here, P denotes a fuzzy probability. For example, if P = Likely, P is a fuzzy subset of $[0,1]$ whose membership function might be as shown in Figure 2.19.

Any probability exceeding 0.8 is a perfect example of Likely. The input to such a rule is a probability distribution on X, the universe of discourse of U. Let f be such a probability distribution. Let

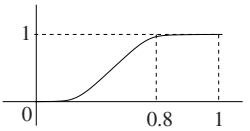

Fig. 2.19

$$W_f = \sum_x A(x)f(x).$$

The quantity W_f represents the average membership in A relative to f. The output is then

$$V(y) = P(W_f) \wedge B(y).$$

What this says is that if the average membership of A relative to the input probability distribution f perfectly fits P, then the output is B.

The last type of statement involving uncertainty we illustrate in this section is one that uses qualifiers. An example of such a rule could be

If Most Big Trucks are Heavy, then the Supply of Gas would be Small.

Here the qualifier is Most. The membership function of Most could be as shown in Figure 2.19. The output in this case could be

$$\text{GasSupply}(y) = \text{Most}\left(\frac{|\text{BigTruckandHeavy}|}{|\text{BigTruck}|}\right) \wedge \text{Small}(y).$$

Here, $|A|$ refers to the cardinality of the fuzzy set A (i.e., $|A| = \sum_x A(x)$). The universe of discourse is the set of trucks under consideration. Each truck has a membership value in Big Truck and in Heavy. The ratio in the parentheses is the fuzzy version of the fraction of Big Trucks that are Heavy.

2.2.6 Defuzzification

In previous subsections we have shown how a set of rules, when provided with some input, generate and output. The output was a fuzzy subset of the appropriate universe of discourse. Most of the time we need to defuzzify the output (i.e., to select an element that in some sense best represents the fuzzy output) Several methods are available and we briefly indicate some of these. In this subsection, B_k will denote the fuzzy output generated by the k-th rule, $1 \leq k \leq N$. We assume that the input is a numerical vector. In a previous subsection we have pointed out that one often aggregates the sets B_k. Let B denote that aggregation. (For example, $B = V_{k=1}^N$ if V is used for aggregation.)

The Centroid Method

The domain of B is partitioned into points $y_1, \ldots, y_M$ and we set

$$y_c(x) = \sum_{i=1}^{M} y_i B(y_i) / \sum_{i=1}^{M} B(y_i).$$

Here, x denotes the input, to stress that the expression on the right-hand side depends on the input x since each B_k and hence B depends on the input x. In practice, the computation of y_i may take some time.

The Center of Sums Defuzzifier

Here, the sets B_k are first combined by addition; that is,

$$B(y) = \sum_{k=1}^{N} B_k(y).$$

Then the centroid of B is found by the formula

$$y_a(x) = \sum_{k=1}^{N} c_{B_k} a_{B_k} / \sum_{k=1}^{N} a_{B_k}.$$

Here, c_{B_k} denotes the centroid for B_k and a_{B_k} denotes the area under B_k. It should be noted in this approach that the areas of overlap between distinct B_k are counted at least twice. However, y_a is easier to compute than y_c.

The Height Defuzzifier

The formula giving this defuzzification is

$$y_h(x) = \sum_{k=1}^{N} \bar{y}_k B_k(\bar{y}_k) / \sum_{k=1}^{N} B_k(\bar{y}_k).$$

Here, $\bar{y}_k$ is the point at which B_k assumes its largest value. If there is more than one point at which B_k is maximum, then $\bar{y}_k$ is the average of all these points.

The Center of Sets Defuzzifier

The defuzzification formula is

$$y_{\mathrm{CoS}}(x) = \sum_{k=1}^{N} c_k \prod_{j=1}^{p} F_j^k(x_j) / \sum_{k=1}^{N} \prod_{j=1}^{p} F_j^k(x_j).$$

Here, it is assumed that the antecedent of the k-th rule is

If X_1 is F_1^k and X_2 is F_2^k and ... and X_p is F_p^k,

and the strength of the rule under input $\mathbf{x} = (x_1, \ldots, x_p)^T$ is $\prod_{j=1}^{p} F_j^k(x_j)$. The number c_k denotes the centroid of B_k. Note that if the consequents of all rules are fuzzy sets G_k and each G_k is symmetric, normal, and convex, then the computations are greatly simplified since $c_k = \bar{y}_k$ and $G_k(\bar{y}_k) = 1$. The quantities $\bar{y}_k$ are known ahead of time and we then have $y_{\mathrm{CoS}} = y_h$ for every input x.

For additional information on defuzzification we refer the reader to [10], [25], and [45].

2.2.7 Two Applications

Numerous applications have been studied using fuzzy rules. In this subsection we sketch two rather different applications.

Forecasting Time Series

Given p measurements, we would like to predict what the next one will be. Assume we have N measurements recorded, $x_1, \ldots, x_N$. This generates $N - p$ elements of a training set of the form

$$(x_k, \ldots, x_{p+k-1});\ x_{p+k}, \qquad k = 1, \ldots, N - p.$$

Then we define $N - p$ rules, each rule being defined by one of the training elements

R_k: If X_1 is A_k^1 and ... and X_p is A_k^p, then Y is B_k,

$1 \leq k \leq N - p$. Each A_k^i is a fuzzy set, say a triangular or a Gaussian membership function centered at x_{k+i-1}. The set B_k is centered at x_{p+k}. Using an additional training set together with a least squares fit, one could later refine the parameters of the membership functions.

An alternate approach to the same problem is to locate an interval $[x_L, x_U]$ where all x_i fall. Partition this interval into overlapping subintervals A_1, A_2, ... as shown in Figure 2.20.

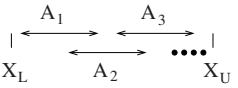

Fig. 2.20

Consider, for example, triangular membership functions, the bases of the triangles being A_1, A_2, ... Determine the degree to which each x_i belongs to the triangular functions. Let the degree be the largest at the triangle based

on $A_{j(i)}$. Then the rules generated are

R_k: If X_1 is $A_{j(k)}$ and X_2 is $A_{j(k+1)}$ and ...and X_p is $A_{j(k+p-1)}$, then Y is $A_j(k+p)$,

$1 \leq k \leq N-p$. Conflicts may be present. The same antecedent might be generated by $(x_k, \ldots, x_{k+p-1})$ and by $(x_l, \ldots, x_{l+p-1})$, yet a different consequent generated by x_{k+p} and x_{l+p}. In that case, the conflict might be resolved by selecting the strongest rule (i.e., by comparing $A_{j(k)}(x_k) \wedge \ldots \wedge A_{j(k+p)}(x_{k+p})$ and $A_{j(l)}(x_l) \wedge \cdots \wedge A_{j(l+p)}(x_{l+p})$). Thus, when a sequence such as y_1,...,y_p is fed to the rules, a fuzzy output in produced and then defuzzified as described in the previous subsections. This output will be the predicted next element of the sequence.

A Fuzzy Controller

We have a mobile robot. The goal is to have the robot move so that it avoids obstacles. In this example, the robot has four sensors. The sensors measure distances front, rear, left, and right. The diagram in Figure 2.21 presents the robot under discussion.

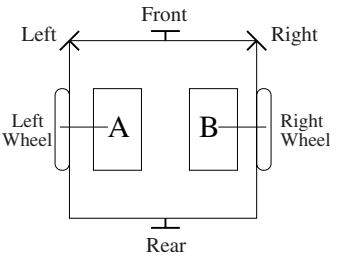

Fig. 2.21

The blocks A and B represent motors contributing to the left and the right wheel. Front, Rear, Left, and Right represent the four sensors that evaluate their respective distances to the obstacle. The distances can have three fuzzy values: Short, Average, and Large. Each motor speed could have seven fuzzy values: Back Fast, Back Average, Back Slow, Stand Still, Forward Slow, Forward Average, and Forward Fast. Rules could be defined as follows:

R_1: If Left Distance is Large and Front Distance is Large and Right Distance is Large, then Left Motor is Fast Forward and Right Motor is Fast Forward.

R_2: If Front Distance is Short, then Left Motor is Forward Average and Right Motor is Backward Average.

The second rule states that if there is an obstacle a short distance ahead, then make a left turn. Other rules such as go to the right if there is an obstacle on the left can easily be written down.

Fuzzy controllers form the bulk of the applications in the area of fuzzy logic. A small fraction of the literature dealing with Fuzzy Controllers can be found in [42], [2], [18], [27], and [26].

2.2.8 The Takayi-Sugeno-Kang Model (TSK Model)

A typical rule in this system is of the form

If X is A and Y is B, then $Z = f(x, y)$.

The input is typically numerical and often the values of the input components are precisely x and y. Often studied are the cases where f is a constant function or a linear function $px + qy + r$. Suppose we have two such rules:

R_1: If X is A_1 and Y is B_1 then Z is $p_1x + q_1y + r_1$.

R_2: If X is A_2 and Y is B_2, then Z is $p_2x + q_2y + r_2$.

If the input is $(x, y)^T$, then the output is

$$\frac{w_1(p_1x + q_1y + r_1) + w_2(p_2x + q_2y + r_2)}{w_1 + w_2},$$

where $w_1 = A_1(x)tB_1(y)$ and $w_2 = A_2(x)tB_2(y)$ and t denotes a t-norm, say $\wedge$ or the product. The interesting part is that no defuzzification is performed; thus the output is not computationally intensive. The TSK model is well adapted to be the fuzzy component of a neuro-fuzzy system. These systems will be discussed in a later section.

2.2.9 Fuzzy Sets of Type 2

A fuzzy set of type 2 is defined by its membership function that maps $X \times [0, 1]$ into $[0, 1]$. Thus, if $\hat{A}$ denotes a fuzzy set of type 2, $\hat{A}(x, u)$ is a number in $[0, 1]$ and the interpretation is: The belief that the membership of x in $\hat{A}$ is u is $\hat{A}(x, u)$. The diagram in Figure 2.22a describes the situation.

The diagram in Figure 2.22b shows a section of $\hat{A}$ when x is kept constant. For every x fixed, $\hat{A}(x, u) = \hat{A}_x(u)$ and $\hat{A}_x$ is a fuzzy subset of $[0, 1]$. In Figure. 2.22b, $\hat{A}_x$ has a Gaussian-like membership function.

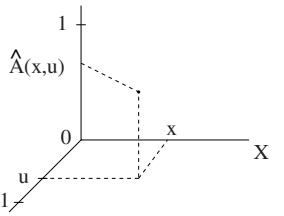

Fig. 2.22a

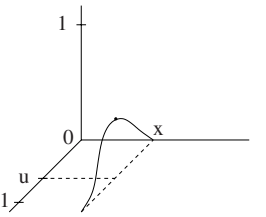

Fig. 2.22b

Thus, $\hat{A}$ factors in the uncertainty that the membership of x in $\hat{A}$ is u, that is, the possibility that x has membership u in $\hat{A}$ is $\hat{A}(x,u)$. Most applications using fuzzy sets of type 2 deal with the case where $\hat{A}_x$ is a subinterval I_x of $[0,1]$ depending on x; that is

$$\hat{A}_x(u) = \begin{cases} 1 \text{ if } u \in I_x \\ 0 \text{ otherwise.} \end{cases}$$

The diagram in Figure 2.23 depicts a situation where $X = \{1,2,3\}$ and a type 2 fuzzy subset $\hat{A}$ of X, where the membership of 1 in X is some number between 0.3 and 0.6 (but it is unknown which number that is), the membership of 2 in $\hat{A}$ is between 0 and 0.3, and the membership of 3 in $\hat{A}$ is 0.6 and 1.

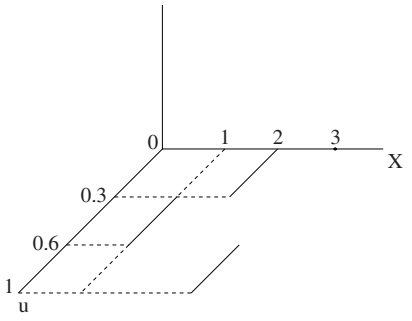

Fig. 2.23

Using previous notation we have $I_1 = [0.3, 0.6]$, $I_2 = [0, 0.3]$, and $I_3 = [0.6, 1]$. Standard operations may be defined using standard interval computations:

$$(\hat{A} \cup \hat{B})_x = [a_x^1 \vee b_x^1, a_x^2 \vee b_x^2],$$
$$(\hat{A} \cap \hat{B})_x = [a_x^1 \wedge b_x^1, a_x^2 \wedge b_x^2],$$
$$c(\hat{A})_x = [1 - a_x^2, 1 - a_x^1].$$

Here, $\hat{A}_x = [a_x^1, a_x^2]$, $\hat{B}_x = [b_x^1, b_x^2]$, and $c(\hat{A})$ denotes the complement of $\hat{A}$. Clearly, the family $\{A_x \,|\, x \in X\}$ determines the set $\hat{A}$.

Given a type 2 set $\hat{A}$, one can define a fuzzy set $\hat{A}_e$ by setting $\hat{A}_e(x) = u_x$, where u_x is a number selected from the interval I_x. The set $\hat{A}_e$ is said to be embedded in $\hat{A}$. If X and U are discretized and X has n elements, and I_{x_i} has M_i elements, the total number of embedded sets is $\prod_{i=1}^{n} M_i$.

We now consider a set of if-then rules where the antecedents and the consequents are fuzzy sets of type 2 and the input is a number denoted by x^*. First, we look at one rule:

If X is $\hat{A}$, then Y is $\hat{B}$,
where $\hat{A}$ and $\hat{B}$ are fuzzy sets of type 2. We assume, moreover, that all relevant spaces have been discretized. Let $\hat{A}_{e,h}$ and $\hat{B}_{e,j}$ be typical embedded sets in $\hat{A}$ and $\hat{B}$, as defined previously, $1 \leq h \leq n_A$, $1 \leq j \leq n_B$. We then consider the rule
If X is $\hat{A}_{e,h}$, then Y is $\hat{B}_{e,j}$.

There are $n_A n_B$ such rules. Given the numerical input x^*, each of these rules produces a fuzzy output, as explained in a previous subsection. Denote the output by $G_{h,j}$. In fact,

$$G_{h,j}(y) = \hat{A}_{e,h}(x^*) t \hat{B}_{e,j}(y),$$

where t denotes a t-norm. The output produced by the rule If X is $\hat{A}$, then Y is $\hat{B}$ is then $\{G_{h,j} \,|\, 1 \leq h \leq n_A, 1 \leq j \leq n_B\}$. Thus, if G denotes the output, we have

$$G(y) = \{G_{h,j}(y) \,|\, 1 \leq h \leq n_A, 1 \leq j \leq n_B\}.$$

Given any type 2 set $\hat{A}$, one can define $\underline{\hat{A}}$ and $\overline{\hat{A}}$ by $(\underline{\hat{A}})_x = a_x$ and $(\overline{\hat{A}})_x = b_x$, where $I_x = [a_x, b_x]$. $(\underline{\hat{A}})$ and $(\overline{\hat{A}})$ are the lower and the upper membership functions of $\hat{A}$, respectively. It can be shown that $\underline{G}$ and $\overline{G}$ are given by the outputs of the rules If X is $\underline{\hat{A}}$, then Y is $\underline{\hat{B}}$ and If X is $\overline{\hat{A}}$, then Y is $\overline{\hat{B}}$. Then, $G(y) = [\underline{G}(y), \overline{G}(y)]$. The formula can be naturally extended to multiple antecedents; no discretization required:

$$\underline{G}(y) = \left(T_{m=1}^{p} \underline{\hat{A}_m}(x_m^*)\right) t \underline{B}(y),$$

$$\overline{G}(y) = \left(T_{m=1}^{p} \overline{\hat{A}_m}(x_m^*)\right) t\overline{B}(y),$$
$$\hat{G}(y) = [\underline{G}(y), \overline{G}(y)].$$

Here, T and t denote t-norms that may or may not be the same. Finally, if there are N such rules, each rule produces an output G_l, as indicated earlier, and the set of N rules produces an output G that is an aggregation of G_l, $1 \leq l \leq N$. Typically,

$$\hat{G}(y) = \cap_{l=1}^{N} \hat{G}_l(y),$$

where the union is as defined earlier.

The Defuzzification Process

How is an interval valued function, such as G defuzzified? The process is accomplished in two steps: type reduction and defuzzification. Type reduction extracts a standard fuzzy set from G that is then defuzzified by any method indicated in a previous subsection. A number of methods leading to type reduction are available, just as a number of defuzzification methods that can be used. A very straightforward type reduction could be obtained by setting

$$T(y) = (\underline{G}(y) + \overline{G}(y))/2.$$

This is followed by one of the standard defuzzification methods applied to T.

We now briefly sketch one type of defuzzification: the height defuzzification. The k-th rule of type 2 produces an output as described earlier, $\hat{B}_k(y) = [\underline{B_k}(y), \overline{B_k}(y)]$, $1 \leq k \leq N$. Obtain the midpoint function $T_k(y) = (\underline{B_k}(y) + \overline{B_k}(y))/2$. Obtain y_k, a point at which T_k is maximum. If T_k is maximum on a set of points, take the average; see Figure 2.24.

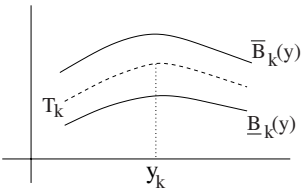

Fig. 2.24

Reorder the N rules if necessary so that we have $y_1 \leq y_2 \leq \cdots \leq y_N$. For each y_i, consider the interval I_{y_i} and partition that interval into M_i points. Consider the set

$$B_h(x) = \left\{ \sum_{i=1}^{N} y_i u_i / \sum_{i=1}^{N} u_i \mid u_i \in I_{y_i} \right\}.$$

A typical element in that set is the center of gravity of the function whose value at y_i is u_i. There are $\prod_{i=1}^{N} M_i$ such centers. It can be shown that $B_h(x)$ is actually an interval. Here, the notation $B_h(x)$ is used to stress that the formula that yields $B_h(x)$ depends on the input x. Thus, $B_h(x) = [c_l, c_r]$, where $c_l = \inf\{\sum y_i u_i / \sum u_i\}$ and $c_r = \sup\{\sum y_i u_i / \sum u_i\}$ over $u_i \in I_{y_i}$. The defuzzification is obtained by taking the midpoint of $[c_l, c_r]$. In the present context, type reduction refers to the defuzzification of fuzzy sets whose membership at y_i is u_i. That defuzzification produces the corresponding centers of gravity $\sum y_i u_i / \sum u_i$, where $u_i \in I_{y_i}$. The process in illustrated in Figure 2.25.

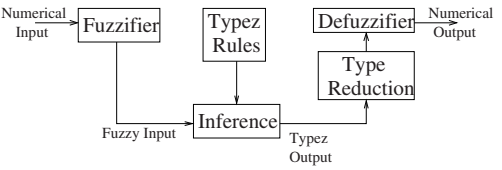

Fig. 2.25

Consider the rule If x is Low, then y is Average, where the membership functions of Low and Average are shown in Figure 2.26a. The idea is to factor in uncertainty due to the difference in expert opinions by replacing the above rule by If x is Low$\hat{}$, then y is Average$\hat{}$. Memberships of type 2 of Low$\hat{}$ and Average$\hat{}$ are shown in Figure 2.26b.

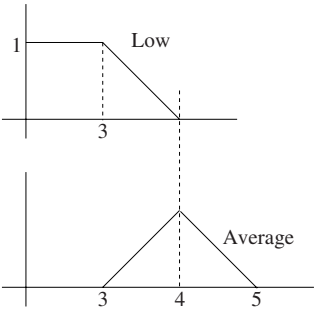

Fig. 2.26a

For more information on fuzzy sets of type 2 see [28], [29], and [30].

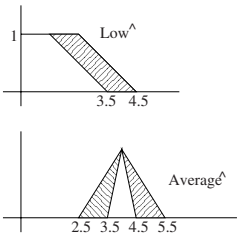

Fig. 2.26b

2.3 Neuro-Fuzzy Systems

2.3.1 Introduction

Neural networks and fuzzy systems have been discussed in the two previous sections. Both systems are highly parallel and use parametric representations for weights and for membership functions. In neural nets, knowledge representation and extraction is a difficult process. In a fuzzy system, knowledge representation takes the natural form of if-then statements. On the other hand, the success of fuzzy systems depends on how accurate the membership functions are. In addition, it is not always clear how to translate knowledge into if-then rules. It is therefore natural to seek a hybrid technology that would combine the advantage of neural nets (i.e., their capability to adapt to a given environment with the advantage of fuzzy systems i.e., their natural knowledge representation). The TSK model has been defined in a previous subsection. Recall that a typical rule has the form

If X is A and Y is B, then $Z = f(x, y)$.

Here, A and B denote fuzzy sets and f is often either a constant or a first-order polynomial. It was pointed out that no defuzzification was necessary. The output is given by the single formula

$$C(x_1, \ldots, c_n) = \sum_{i=1}^{N} \bar{w}_i f_i(x_1, \ldots, x_n)$$

if there are N rules and if f_i is the consequent of the i-th rule. Here, $\bar{w}_i$ denote the normalized weights

$$\bar{w}_i = w_i / \sum_{j=1}^{N} w_j,$$

where w_i is the strength of the i-th rule when the input is $(x_1, \ldots, x_n)^T$.

2.3.2 Fuzzy Neurons

We now extend the idea of a neuron to define "fuzzy neuron." Although a general definition could be given, we mainly will use three types of neuron. The first type has a membership function stored that it uses to weight a fuzzy input, whereas the other two types use boolean operations. The three types are shown in Figures 2.27a – 2.27c respectively.

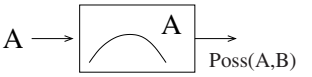

Fig. 2.27a

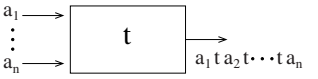

Fig. 2.27b

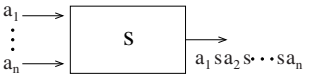

Fig. 2.27c

The input to the first type is a fuzzy set B and the output is $\text{Poss}(A, B)$. In particular, if the input is a number x, then the output is $A(x)$. The other two types perform a t-norm and an s-norm operation, respectively. Inputs here are numbers in $[0, 1]$. Often t is the minimum or the product operation and s is often the maximum operation.

2.3.3 The Adaptive Neuro-Fuzzy Inference System (ANFIS)

It is straightforward to convert TSK-type rules into an ANFIS architecture. For example, suppose that we have the following two rules

R_1: If x is A_1 and y is B_1, then z is $f_1(x, y)$.

R_1: If x is A_2 and y is B_2, then z is $f_2(x, y)$.

Here, $f_1(x, y) = p_1x + q_1y + r_1$ and $f_2(x, y) = p_2x + q_2y + r_2$. This system of rules translates into a five-layer ANFIS shown in Figure 2.28.

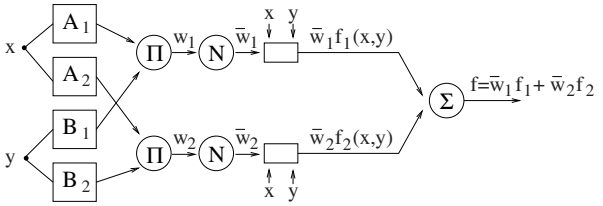

Fig. 2.28

In this case, the input is a numerical pair (x, y). The first layer consists of fuzzy neurons that output $A_1(x)$, $A_2(x)$, $B_1(y)$, and $B_2(y)$. The second layer, whose nodes are labeled Π, outputs the t products. For example, it could output $A_1(x)B_1(y)$ and $A_2(x)B_2(y)$. The third layer, whose nodes are labeled N, simply normalizes the previous strengths of the two rules. The next layer simply outputs $f_1(x, y)$ and $f_2(x, y)$ scaled by the normalized strengths. Finally, the last layer's single node labeled Σ outputs what the TSK system would output.

2.3.4 A Comparison Among Three Approaches

In Figure 2.21 we have shown a simple vehicle whose goal is to avoid obstacles. We have stated the type of fuzzy rules that may control this vehicle. A purely neural net approach to this could be to define two neural nets for which the input is the 4-dimensional vector describing the four distances to the obstacle $(\text{Left}, \text{Front}, \text{Right}, \text{Rear})^T$ and the output would be the corresponding forces applied to the left motor and to the right motor. Thus, the two neural nets would connect the four sensors input to the two motors. The weight could initially be defined by, for example, the matrix W.

	Left	Front	Right	Rear	Bias
A	0.2	−0.5	−0.3	0.2	0.3
B	−0.4	0.1	0.2	0.3	0.3

At every collision, the matrix W should be updated, using, for example, backpropagation. The target would be provided by the desired trajectory before collision has occurred. It is certainly not clear, by looking at the weights, how to characterize the behavior of the vehicle. That characterization is very apparent using the fuzzy approach.

Consider now the neuro-fuzzy approach. Rules could take the following forms:

R_1: If Left Distance is Large and Front Distance is Large, then Force applied to Left Motor is $f_1(x, y)$.

R_2: If Front Distance is Short, then Force applied to Left Motor is $f_2(y)$.

Here, x is the Left distance and y is the Front distance: $f_1(x, y) = p_1x+q_1y+r_1$ and $f_2(y) = q_2y+r_2$. What is needed is a learning algorithm to estimate p_1, q_1, r_1, q_2, and r_2 as well as the parameters involved in the membership functions of Large and Short.

Learning Algorithm for ANFIS

We need a learning algorithm for systems such as the one sketched in Figure 2.28. This system could be viewed as an example of an adaptive net and its parameters estimated as described in section 2.1. Perhaps a better way to proceed is to partition the parameter set into the consequent parameters – those coming from

$$f_i(x, y, z, \ldots) = p_{i,1}x + p_{i,2}y + p_{i,3}z + \cdots + r_i$$

– and the antecedent parameters, i.e. those coming from the defining membership functions of $A_{i,1}, \ldots, A_{i,p}$, assuming that there are p variables in the antecedent. We note that the output $f = \sum \bar{w}_i f_i$ is a linear function in $p_{i,1}$, $p_{i,2}$, $p_{i,3}$,...,r_i, $1 \leq i \leq N$, where N is the number of rules. These parameters can therefore be estimated by a least squares fit. The antecedent parameters can be estimated by using the backpropagation algorithm for adaptive nodes.

2.3.5 Organizing Rules and Architectures

Consider the four rules:

If x is A_1 and y is B_1, then $z_1 = c_{1,1}$.

If x is A_2 and y is B_2, then $z_1 = c_{2,1}$.

If x is D_1 and y is E_1 then $z_2 = c_{1,2}$.

If x is D_2 and y is E_2, then $z_2 = c_{2,2}$.

Here, $c_{i,k}$ refers to the formula giving z_k in the i-th rule and

$$c_{i,k} = p_{i,k}x + q_{i,k}y + r_{i,k}.$$

The extension of this notation to the case of more than two variables is obvious. The diagram in Figure 2.29 shows a possible structure for these four rules. The idea is to put together in parallel ANFIS structures for each output. Here, O_1 and O_2 obviously refer to the outputs z_1 and z_2 generated by these four rules when the input is the numerical vector $(x, y)^T$. Such a structure is called "Coactive Adaptive Neuro-Fuzzy Inference System" or CANFIS in the

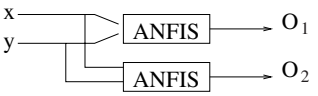

Fig. 2.29

literature. Thus, one way to construct a CANFIS is to put several ANFIS in parallel (one ANFIS for each output). This way of proceeding is often inefficient since no parameter sharing takes place and one is typically faced with a large number of parameters to estimate. A better way to proceed is to use multiple consequents. For example, consider the following two rules:

If x is A_1 and y is B_1, then $z_1 = c_{1,1}$ and $z_2 = c_{1,2}$,

If x is A_2 and y is B_2, then $z_1 = c_{2,1}$ and $z_2 = c_{2,2}$.

The diagram in Figure 2.30 shows a possible structure for these two rules.

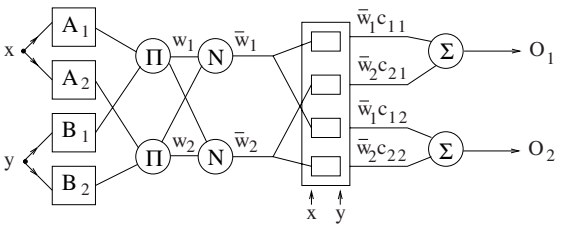

Fig. 2.30

Clearly, in this approach, parameter sharing takes place and fewer parameters are to be estimated for this CANFIS. On can draw an interesting comparison between a CANFIS and a standard neural net. The part following the N layer in Figure 2.30 could be replaced by the system shown in Figure 2.31.

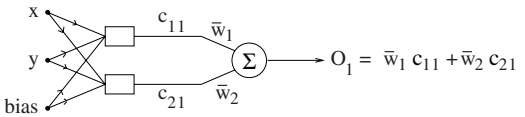

Fig. 2.31

Here, the weights connecting x, y, and the bias to the first neuron of the first layer are $p_{1,1}$, $q_{1,1}$, and $r_{1,1}$, whereas the weights connecting x, y, and the bias to the second neuron are $p_{2,1}$, $q_{2,1}$, and $r_{2,1}$. Thus, the two neurons will

produce $c_{1,1}$ and $c_{2,1}$ as outputs. The weights connecting these to the Σ node are $\bar{w}_1$ and $\bar{w}_2$, obtained as outputs from the first part of the system shown in Figure 2.30. Here, the system produces the output O_1. A similar net, not shown in the diagram, is needed to produce the second output O_2. Training of this set will determine the weights $p_{i,j}$, $q_{i,j}$, and $r_{i,j}$. From this it is therefore clear that CANFIS can be viewed as appending the linguistic component in front of a standard net. In fact, the neural component may be modified to handle nonlinear functions of x and y. Consider the general m-th rule:

R_m: If x_1 is A_1 and x_2 is A_2 and ..., then $z_1 = c_{m,1}$, $z_2 = c_{m,2}, \cdots, z_p = c_{m,p}$.

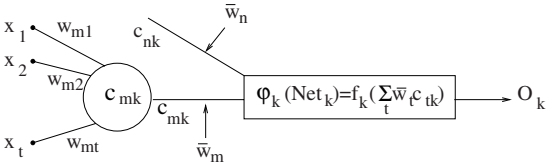

Fig. 2.32

Figure 2.32 shows how a set of rules of type R_m generate the outputs O_k.

We have a training set and the weights $w_{m,k}$, $1 \leq k \leq t$ are adjusted for $1 \leq m \leq N$ if we have N rules of type R_m. In the case discussed previously, f was the identity function. Here, f_k could, for example, be $f_k(u) = 1/(1+e^{-u})$. The net prior to the $\varphi_k(\text{Net}_k)$ box corresponds to the action of one rule (R_m in this case) and could, in principle, be trained by itself. The $\varphi_k(\text{Net}_k)$ box plays the role of consensus builder by weighting every rule according to its normalized strength. One could still generalize this by allowing the transfer function at $c_{m,k}$ to be other than the identity function.

2.3.6 Updating the Consequents and the Antecedents

In order to update the consequents, one needs to update the weights $w_{m,k}$, $1 \leq k \leq t$, $1 \leq m \leq N$. In order to do this, the function

$$E = (t_k - O_k)^2/2$$

needs to be minimized, where t_k is the corresponding target and $O_k = \varphi_k(\text{Net}_k)$. We apply the steepest descend method: $\triangle w_{m,i} = -\eta_m \partial E/\partial w_{m,i}$, where η_m is a small positive number. So,

$$\triangle w_{m,i} = -\eta_m \frac{\partial E}{\partial \text{Net}_k} \frac{\partial \text{Net}_k}{\partial w_{m,i}}$$

$$= -\eta_m \frac{\partial E}{\partial O_k} \frac{\partial O_k}{\partial \mathrm{Net}_k} \frac{\partial \mathrm{Net}_k}{\partial c_{m,k}} \frac{\partial c_{m,k}}{\partial w_{m,i}}$$
$$= -\eta_m \frac{\partial E}{\partial O_k} \varphi_k'(\mathrm{Net}_k) \bar{w}_m x_i$$
$$= \eta_m (t_k - O_k) f_k'(\mathrm{Net}_k) \bar{w}_m x_i.$$

Let a be a parameter involved in the antecedent. We have $\triangle a = -\eta_a \partial E/\partial a$, where η_a is a small positive number. Then

$$\triangle a = -\eta_a \frac{\partial E}{\partial O_k} \frac{\partial O_k}{\partial a} = -\eta_a \frac{\partial E}{\partial O_k} \frac{\partial O_k}{\partial \mathrm{Net}_k} \frac{\partial \mathrm{Net}_k}{\partial w_m} \frac{\partial w_m}{\partial a}.$$

Since

$$\frac{\partial \mathrm{Net}_k}{\partial w_m} = \frac{\partial \mathrm{Net}_k}{\partial \bar{w}_m} \frac{\partial \bar{w}_m}{\partial w_m},$$

it follows that

$$\triangle a = -\eta_a \frac{\partial E}{\partial O_k} \varphi_k'(\mathrm{Net}_k) c_{m,k} \frac{\partial \bar{w}_m}{\partial a}$$
$$= \eta_a (t_k - O_k) \varphi_k'(\mathrm{Net}_k) c_{m,k} \frac{\partial \bar{w}_m}{\partial w_m} \frac{\partial w_m}{\partial a}.$$

Recall that w_m denotes the strength of the m-th rule that was defined by

$$w_m = A_1(x_1) t A_2(x_2) t \cdots t A_l(x_l),$$

where t denotes a t-norm. Since a is a parameter involved in the membership functions, w_m is a function of a. Also, $\bar{w}_m = w_m / \sum w_i$.

For further readings on neuro-fuzzy systems, the reader is referred to the works in [13], [14], [15], [31], [32], [40], and [41].

2.4 The Theory of Evidence

2.4.1 Introduction

There are three types of uncertainty. The fuzziness comes about from the absence of sharp boundaries separating classes of objects. The nonspecificity comes about from the size of the set of alternatives and the strife comes about from the conflicts among the available alternatives. Entropy measures fuzziness and is more or less a direct extension of entropy as known in probability theory. If p_i, $1 \leq i \leq n$, is a discrete probability distribution, the entropy function is defines as

$$H(p_1, \ldots, p_n) = -\sum p_i \log_2 p_i.$$

This function is zero when $p_{i^*} = 1$ and $p_j = 0$, $j \neq i^*$ (i.e., when we are certain that alternative i^* will occur, and can be shown to reach a maximum when $p_i = 1/n$ for all i (i.e., when all outcomes have equal probabilities).

If A is a fuzzy set defined on a finite universe of discourse and $A = \sum \alpha_i / x_i$ with all the x_i distinct, then the entropy is defined by

$$H(A) = \sum h(A(x_i)),$$

where h is a function from $[0,1]$ to $[0,1]$ satisfying the following properties:

$h(x) = 0$ if and only if $x = 0$ or $x = 1$.

$h(x)$ is maximum at $x = 1/2$.

$h(x) = h(1-x)$.

h is monotone increasing on $[0, 1/2]$ and monotone decreasing on $[1/2, 1]$.

Examples of such functions are $h(x) = 1 - |2x - 1|$, $h(x) = 4x(1-x)$, and $h(x) = -x \log_2 x - (1-x) \log_2 (1-x)$.

It follows that $H(A)$ is maximum at $A(x) = 1/2$ for all x; that is H is maximum at the "maximal fuzzy set." Also, $H(A) = 0$ if $A(x) \in \{0, 1\}$ for all x; that is H is 0 on standard nonfuzzy sets. Let $A_{1/2}$ denote the standard non-fuzzy set $\{x \mid A(x) \geq 1/2\}$. If $A_{1/2}$ denotes the membership in that set and if $h(x) = 1 - |2x - 1|$, it can be shown that

$$H(A) = 2 \sum_i |A(x_i) - A_{1/2}(x_i)|.$$

So, $H(A)$ is twice the Hamming distance of A to $A_{1/2}$. Intuitively, the more A is removed from the standard set $A_{1/2}$, the higher the entropy of A is and the more fuzzy A is.

In the continuous case, the entropy H can be defined as

$$H(A) = \int h(A(x))\, dx.$$

2.4.2 Evidence Theory

The concepts of non specificity and strife are best defined in the general context of evidence theory.

Definition 1. *Let $\mathcal{A}$ be the nonempty family of subsets of the set X. A "fuzzy measure" on $< X, \mathcal{A} >$ is a function g from $\mathcal{A}$ into $[0,1]$ such that the following hold:*

(1) $g(\emptyset) = 0$ *and* $g(X) = 1$.

(2) *If* $A \subset B \subset X$ *then* $g(A) \leq g(B)$.

(3) If A_n is an increasing sequence of subsets in $\mathcal{A}$ (i.e., $A_n \subset A_{n+1}$ for every n), and if $\cup A_n \in \mathcal{A}$, then $g(\cup A_n) = \lim g(A_n)$.

(4) If A_n is a decreasing sequence of subsets in $\mathcal{A}$ and $\cap A_n \in$ A, then $g(\cap A_n) = \lim g(A_n)$.

Of course, if X is a finite set, then properties (3) and (4) are vacuous. The interpretation of $g(A)$ is "the total evidence" that some unknown element belongs to the set A. From the definition it follows that

$$g(A \cap B) \leq g(A) \wedge g(B) \quad \text{and} \quad g(A \cup B) \geq g(A) \vee g(B).$$

Note that a probability function is a special case of a fuzzy measure. At times, properties (1), (2), and (3) will be the only properties assumed, and sometimes properties (1), (2), and (4) will be the only properties assumed. Two fuzzy measures are of particular importance: the belief and the plausibility. These measures will be denoted by Bel and Pls.

Definition 2. *Bel is defined as a function from the subsets of X to $[0,1]$ with the following properties:*

(1) $\text{Bel}(\emptyset) = 0$ *and* $\text{Bel}(X) = 1$.

(2) $\text{Bel}(A_1 \cup \cdots \cup A_n) \geq \sum_j \text{Bel}(A_j) - \sum_{j<k} \text{Bel}(A_j \cap A_k) + \cdots + (-1)^{n+1}\text{Bel}(A_1 \cap \cdots \cap A_n)$.

If X is infinite we also require property (4) of fuzzy measures. Note that $\text{Bel}(A) + \text{Bel}(\bar{A}) \leq 1$.

Pls is also defined as a function from subsets of X to $[0,1]$ satisfying the following properties:

(1) $\text{Pls}(\emptyset) = 0$ *and* $\text{Pls}(X) = 1$.

(2) $\text{Pls}(A_1 \cap \cdots \cap A_n) \leq \sum_j \text{Pls}(A_j) - \sum_{j<k} \text{Pls}(A_j \cup A_k) + \cdots + (-1)^{n+1}\text{Pls}(A_1 \cup \cdots \cup A_n)$.

If X is infinite, we also require property (3) of fuzzy measures. Note that $\text{Pls}(A) + \text{Pls}(\bar{A}) \geq 1$.

Both Bel and Pls can be characterized by a third function called mass.

Definition 3. *A mass m is a function from subsets of X to $[0,1]$ satisfying*

$$m(\emptyset) = 0 \quad \textit{and} \quad \sum_{A \subset X} m(A) = 1.$$

It can be shown that if

$$h_1(A) = \sum_{B \subset A} m(B) \quad \text{and} \quad h_2(A) = \sum_{B \cap A \neq \emptyset} m(B),$$

then h_1 is a belief function and h_2 is a plausibility function. The mass m is said to generate belief h_1 and plausibility h_2. Conversely, we have the following:

If Bel is any belief function, then

$$m(A) = \sum_{B \subset A} (-1)^{|A-B|} \text{Bel}(B)$$

is a mass function that generates belief Bel.

If Pls is any plausibility function, then

$$m(A) = \sum_{B \subset A} (-1)^{|A-B|} (1 - Pls(\bar{B}))$$

is a mass function that generates plausibility Pls.

Whereas, as pointed out earlier, a fuzzy measure applied to a set A measures the total evidence that a partially known element belongs to A, $m(A)$ measures the evidence (not total) that a partially known object is precisely in A ($m(A)$ does not include, for example, any evidence that the object might be in some subset of A). In this connection, it should be stressed that $A \subset B$ does not necessarily imply $m(A) \leq m(B)$. For example, if we have three types of aircraft: bomber (B), fighter (F), and passenger plane (PP) – and we are almost sure that the aircraft we are focusing on is a military plane, the corresponding mass defined on $X = \{\text{B}, \text{F}, \text{PP}\}$ could be

$$m(\{\text{B}, \text{F}\}) = 0.9, \qquad m(\{\text{B}, \text{F}, \text{PP}\}) = 0.1.$$

The reason m is small on $\{\text{B}, \text{F}, \text{PP}\}$ is that evidence is strong the aircraft is not any one of the three aircraft since PP is practically ruled out. On the other hand, the total evidence for X is

$$\text{Bel}(\{\text{B}, \text{F}, \text{PP}\}) = m(\{\text{B}, \text{F}\}) + m(\{\text{B}, \text{F}, \text{PP}\}) = 1.$$

2.4.3 Composition of Masses

Given a mass m, those subsets of X on which m is not zero are called the focal elements of m. If m_1 and m_2 denote masses generated by independent sources, then the composition of m_1 and m_2 is defined to be a mass whose focal elements are the intersections of the focal elements of m_1 and m_2. The composition mass is defined by

$$(m_1 \oplus m_2)(A) = \sum_{B \cup C = A} m_1(B)m_2(C) / \sum_{B \cap C \neq \emptyset} m_1(B)m_2(C).$$

In other words, we discount conflicting focal elements. We can rewrite the above as follows:

$$(m_1 \oplus m_2)(A) = \sum_{B \cup C = A} m_1(B)m_2(C)/(1 - K),$$

where

$$K = \sum_{B \cap C = \emptyset} m_1(B)m_2(C).$$

One can define $m_1 \oplus \cdots \oplus m_n$ inductively by

$$m_1 \oplus \cdots \oplus m_n = (m_1 \oplus \cdots \oplus m_{n-1}) \oplus m_n.$$

Example 1. Assume we have three persons who are suspected of some crime. The three persons are John, James, and Jane. James and Jane are siblings. There is a piece of evidence that indicates a male was involved in the crime and the strength of the evidence (on a scale of 0 to 1) is 0.7. There is a slight amount of evidence that Jane is actually the guilty one and the strength of that evidence is 0.2. A totally different piece of evidence seems to indicate that siblings were involved and the strength of that is 0.6. That second piece of evidence seems to confirm what the previous evidence had shown: a slight possibility that Jane was involved. The strength to support this assumption is still 0.2. Putting these two pieces together, how strong is the evidence against any single individual? Let m_1 and m_2 be the masses corresponding to the two pieces of evidence. We have

$$m_1\{\text{John, James}\} = 0.7, \qquad m_1\{\text{Jane}\} = 0.2.$$

There is a floating .1 that we assign to the whole set as that .1 could be of concern to any one of the individuals, so

$$m_1\{\text{John, James, Jane}\} = 0.1.$$

Similarly,

$$m_2\{\text{James, Jane}\} = 0.6, \quad m_2\{\text{Jane}\} = 0.2, \quad m_2\{\text{John, James, Jane}\} = 0.2.$$

We put the two pieces together by combining m_1 and m_2. Since the focal elements of $m_1 \oplus m_2$ are the intersections of the focal elements of m_1 and m_2 and since $\{\text{John}\}$ is not an intersection of two such focal elements, $(m_1 \oplus m_2)\{\text{John}\} = 0$. So, there is no evidence pointing to John being the sole guilty person. Let

$$A = \{\text{John, James}\}, \quad B = \{\text{Jane}\}, \quad C = \{\text{James, Jane}\},$$

and $U = \{\text{John, James, Jane}\}$. Then, $K = m_1(A)m_2(B) = 0.14$, $1-K = 0.86$ and

$$(m_1 \oplus m_2)(A) = m_1(A)m_2(U)/(1-K) = 0.14/0.86 \approx 0.16.$$

Thus, although there is no evidence against John alone, there is approximately 0.16 evidence that John or James did commit the crime, but we cannot determine which of those two. Next, we have

$$(m_1 \oplus m_2)(\{\text{James}\}) = m_1(A)m_2(C)/(1-K) = 0.42/0.86 \approx 0.49.$$

Although none of the two pieces of evidence pointed to James acting alone, pulling all the information together yields an evidence of approximately 0.49 that James was the sole perpetrator. Finally,

$$\begin{aligned}&(m_1 \oplus m_2)\{\text{Jane}\} = \\ &(m_1(B)m_2(U) + m_1(B)m_2(b) + m_1(B)m_2(U)) + m_1(U)m_2(B))/(1-K) \\ &= 0.22/0.86.\end{aligned}$$

Whereas the separate pieces of evidence gave 0.2 evidence against Jane, the combined evidence against Jane is slightly higher, approximately 0.26. Obviously, this type of reasoning can be used in object recognition problems where features are known with specified probabilities.

2.4.4 Possibility Theory

A special case of evidence theory is provided by possibility theory. Here, the mass has nested focal elements. Thus, if a mass has focal elements $A_1, \cdots, A_n$, we have $A_1 \subset \cdots \subset A_n$. If this is so, it is straightforward to show that

$$\text{Bel}(A \cap B) = \min\{\text{Bel}(A), \text{Bel}(B)\}$$

and

$$\text{Pls}(A \cup B) = \max\{\text{Pls}(A), \text{Pls}(B)\}.$$

In this case, one often uses the terms necessity (Nec) and possibility (Pos) instead of belief and plausibility. Again, it is straightforward to show that

$$\begin{aligned}&\text{Nec}(A) = 1 - \text{Pos}(\bar{A}), \\ &\text{Nec}(A) > 0 \quad \text{implies} \quad \text{Pos}(A) = 1, \\ &\text{Pos}(A) < 1 \quad \text{implies} \quad \text{Nec}(A) = 0.\end{aligned}$$

The following is an important theorem that we state without a proof.

Theorem 1. *Every possibility function is uniquely determined by a possibility distribution function* $r : X \to [0,1]$ *so that* $\text{Pos}(A) = \sup\{r(x) \,|\, x \in A\}$. *The function* r *is defined by* $r(x) = \text{Pos}\{x\}$. *If, in addition,* X *is discrete (i.e.,* $X = \{x_1, \ldots, x_n\}$*), then* $1 = r_1 \geq r_2 \geq \cdots \geq r_{n+1} = 0$ *and* $m(A_i) = r_i - r_{i+1}$ *where* $A_i = \{x_1, \ldots, x_i\}$ *and* $r_i = r(x_i) = \text{Pos}\{x_i\}$.

Example 2. Let $X = \{x_1, \ldots, x_7\}$. Let m be a mass on the subsets of X whose focal elements are $A_2 = \{x_1, x_2\}$, $A_3 = \{x_1, x_2, x_3\}$, $A_6 = \{x_1, \ldots, x_6\}$, and $A_7 = X$. Note that the focal elements of m are nested: $A_2 \subset A_3 \subset A_6 \subset A_7$. Assume that

$$m(A_2) = 0.3, \quad m(A_3) = 0.4, \quad m(A_6) = 0.1, \quad m(A_7) = 0.2.$$

Then since $r_i = \text{Pos}\{x_i\} = \sum_{j=i}^{7} m(A_j)$, we obtain $r_1 = 1$, $r_2 = 1$, $r_3 = 0.7$, $r_4 = 0.3$, $r_5 = 0.3$, $r_6 = 0.3$, and $r_7 = 0.2$. Now, for example,

$$\text{Poss}\{x_3, x_4, x_5\} = \sup\{r_3, r_4, r_5\} = 0.7.$$

2.4.5 Relation of Possibility to Fuzzy Sets

Let F be a fuzzy subset of X. If F is used as a possibility distribution function, taking into account the discussion in the previous section, it is natural to define a possibility associated with F by

$$\text{Pos}_F(A) = \sup_{x \in A} F(x).$$

It is also natural to define

$$\text{Nec}_F(A) = 1 - \text{Pos}_F(\bar{A}).$$

Note that when A is a standard crisp set, then (see Section 2.4)

$$\text{Pos}_F(A) = \sup_x (A(x) \wedge F(x)) = \text{Poss}(A, F).$$

Example 3. Let F be a fuzzy set defined by $F = 0.2/1 + 0.4/2 + 0.5/3 + 1/4 + 0.5/5 + 0.4/6 + 0.2/7$. We now order the elements of X in decreasing order of their memberships in F. We then have the list $4, 3, 5, 2, 6, 1, 7$. The sets A_i are: $A_1 = \{4\}$, $A_2 = \{4, 3\}$, $A_3 = \{4, 3, 5\}$, $A_4 = \{4, 3, 5, 2\}$, $A_5 = \{4, 3, 5, 2, 6\}$, $A_6 = \{4, 3, 5, 2, 6, 1\}$, and $A_7 = \{4, 3, 5, 2, 6, 1, 7\}$. Using the formula $m(A_i) = r_i - r_{i+1}$, where $r_i = F(x_i)$ (with $r_1 = 1$ and $r_8 = 0$), we see that the focal elements of m are A_1, A_3, A_5, and A_7. Then, for example,

$$\text{Poss}\{x_2, x_5, x_7\} = \max\{r_2, r_5, r_7\} = \max\{F(2), F(5), F(7)\} = 0.5.$$

On the other hand, $m(A_1) = 0.5$, $m(A_3) = 0.1$, $m(A_5) = 0.2$, and $m(A_7) = 0.2$, so

$$\text{Pls}\{x_2, x_5, x_7\} = \sum_{j=2}^{7} m(A_j) = .5.$$

The two computations yield the same value.

It is clear that if focal elements are single points, the mass is then a probability distribution and Bel = Pls and coincide with probability measures. Thus, in general, Bel < Pls and can be viewed as the lower and the upper bound of some unknown probability measure. In some sense, Pls − Bel is a measure of the uncertainty regarding the determination of some appropriate probability measure.

2.4.6 Nonspecificity

Let A be a standard crisp set. The nonspecificity of A is defined as

$$N(A) = \log_2 |A|,$$

where, of course, $|A|$ denotes the cardinality of A. Intuitively, this is how many splittings of A into equal halves is required to arrive at a single answer. Clearly, the larger the cardinality of A is, the higher the nonspecificity is. Another interpretation is the number of bits required to represent all of the elements of A. If A represents the set of currently viable alternatives and if new information comes in that reduces the number of viable alternatives to a set B where $B \subset A$, then the reduction of uncertainty is given by $N(A) - N(B) = \log_2(A/B)$. This could be used as a measure of the amount of information received, since the information received could reasonably be identified with the reduction in uncertainty.

How can this be generalized to the case where A is a fuzzy set? We define

$$N(A) = \frac{1}{h(A)} \int_0^{h(A)} \log_2 |A_\alpha| \, d\alpha.$$

Here, $h(A)$ denotes the height of A (i.e., $h(A) = \sup_x A(x)$). Furthermore, $|A_\alpha|$ denotes the α-cut of A; that is A_α is a crisp set defined by $A_\alpha = \{x \mid A(x) \geq \alpha\}$ and $|A_\alpha|$ denotes the length of A_α (assuming A is such that A_α is an interval for every $\alpha \geq 0$).

The obvious analog for the discrete case is

$$N(A) = \frac{1}{h(A)} \sum_{i=0}^{n-1} (\log_2 |A_{\alpha_i}|)(\alpha_{i+1} - \alpha_i),$$

where $A = \sum_{i=1}^{n} \alpha_i / x_i$, with $\alpha_n \geq \cdots \geq \alpha_1 \geq \alpha_0 = 0$. Clearly, if A is a crisp set, then $h(A) = 1$, $1 = \alpha_1 \geq \alpha_0 = 0$, $|A_\alpha| = |A|$, and we get back the nonspecificity formula for crisp sets.

If we have a possibility distribution $r =< r_1, \ldots, r_n >$ with $1 = r_1 \geq \cdots \geq r_n \geq r_{n+1} = 0$, the nonspecificity of r is defined by

$$N(r) = \sum_{i=2}^{n} (r_i - r_{i+1}) \log_2 i.$$

The reason $N(r)$ is defined as above is the following: Let F be a fuzzy set. Denote by r_F a possibility distribution associated with F to be defined later. Then an interesting result is

$$N(r_F) = N(F),$$

provided these quantities are redefined in the context of the theory of evidence. Given a mass m, the nonspecificity of m is defined as

$$N(m) = \sum_{A \mid m(A) \neq 0} m(A) \log_2 |A|,$$

that is, the sum is over all focal elements of m. Thus, $N(m)$ is the averaged nonspecificity over all focal elements of m. Now, given a fuzzy set F over a finite universe of discourse X, a mass m_F is generated that has focal elements $A_i = \{x_1, \ldots, x_i\}$, $i = 1, \ldots, n$, and a possibility distribution r_F is generated with $r_F(x_i) = \text{Pos}_F(x_i)$ and $m_F(A_i) = r_i - r_{i+1}$ (see Section 4.4). Let $r_{F,i} = r_F(x_i)$. Clearly, $|A_i| = i$. Then

$$N(m_F) = \sum_{i=1}^{n} m(A_i) \log_2 |A_i| = \sum_{i=2}^{n} (r_{F,i} - r_{F,i+1}) \log_2 i = N(r_F).$$

We thus define the nonspecificity of a fuzzy set F by

$$N(F) = N(m_F) = N(r_F).$$

Note that if m is a probability distribution, its focal elements are singletons and m, and in that case, since $\log_2 1 = 0$, we have $N(m) = 0$. Thus, a probability distribution has zero nonspecificity .

2.4.7 Strife

The strife function will generalize the Shanon entropy function and is again best defined in the context of the theory of evidence. We start by defining conflict over a subset A of X as

$$\text{Con}(A) = \sum_{B} m(B) \frac{|A - B|}{|A|}.$$

Note that if $A \subset B$, then $|A - B| = 0$; that is sets that contain A are not counted in the conflict because $A \subset B$ implies $A \to B$ (A implies B). So if B follows from A, it should not contribute to the conflict generated by A. We would like the strife function to generalize the concept of entropy function so we start by looking at the special case where m is a probability distribution p. In this case,

$$\text{Con}(\{x\}) = \sum_{y \neq x} p(y) = 1 - p(x).$$

So for the probability distribution the entropy is

$$-\sum_{x} p(x) \log_2 p(x) = -\sum_{x} p(x) \log_2 (1 - \text{Con}(\{x\})).$$

Then it is natural to define the strife function, relative to a mass m by

$$S(m) = -\sum_{A} m(A) \log_2 (1 - \text{Con}(A)),$$

where the sum is, of course, over the focal sets of m. Note that as $\mathrm{Con}(A)$ increases from 0 to 1, $-\log_2(1-\mathrm{Con}(A))$ increases from 0 to ∞. Also, note that $\mathrm{Con}(A)$ is the averaged proportion of elements of A that fail to be in B where B is a focal subset of A. The strife $S(m)$ can be rewritten as

$$\begin{aligned} S(m) &= -\sum_A m(A)\log_2[\sum_B m(B)(1-|A-B|/|A|)] \\ &= -\sum_A m(A)\log_2[\sum_B m(B)|A\cap B|/|A|] \\ &= \sum_A m(A)\log_2|A| - \sum_A m(A)\log_2[\sum_B m(B)|A\cap B|]. \end{aligned}$$

If we denote by $R(m)$ the second sum on the right-hand side, we have

$$S(m) = N(m) - R(m).$$

In particular, we can now define strife in the context of possibility theory, and therefore in the context of fuzzy sets. Recall that in the context of possibility theory we have nested focal elements $A_1 \subset \cdots \subset A_n$, where $A_i = \{x_1, \cdots, x_i\}$, $r_i = r(x_i)$, $1 = r_1 \geq \cdots \geq r_n \geq r_{n+1} = 0$ with $m(A_i) = r_i - r_{i+1}$. So in this case,

$$S(m) = N(m) - R(m) = \sum_{i=2}^{n}(r_i - r_{i+1})\log_2 i - R(m).$$

The quantity $R(m)$ can be written as

$$R(m) = \sum_{i+1}^{n}(r_i - r_{i+1})\log_2\left(\sum_{j=1}^{n}(r_j - r_{j+1})\min\{j,i\}\right).$$

So the final form of $S(m)$ is

$$S(m) = N(m) - \sum_{i=1}^{n}(r_i - r_{i+1})\log_2\left(i/\sum_{j=1}^{i} r_j\right).$$

As a special case, the strife generated by a fuzzy set can be obtained (assuming the set is normal and defined on a discrete universe of discourse) by setting

$$r_1 = F(x_1) \geq r_2 = F(x_2) \geq \cdots \geq r_n = F(x_n) \geq r_{n+1} = 0.$$

The total uncertainty present in a decision problem is the sum of the nonspecificity and the strife (i.e., $N(m) + S(m)$). Often a viable decision is one that minimizes uncertainty. Our formulation therefore allows for decision making;for example, if alternate outputs are fuzzy sets F_i, we select the decision that minimizes $N(m_{F_i}) + S(m_{F_i})$.

For additional work on the theory of evidence and related topics, we refer the reader to [19], [33], and [7].

2.5 Rough Sets and Fuzzy Sets

2.5.1 Introduction

Let X be the universe of discourse. Assume that we have an equivalence relation R on X. Then R is a function from $X \times X$ into $\{0, 1\}$ that satisfies

$$\begin{aligned} &R(x, x) = 1 \quad \text{for all} \quad x \in X, \\ &R(x, y) = 1 \quad \text{implies} \quad R(y, x) = 1, \\ &R(x, y) = 1 \quad \text{and} \quad R(y, z) = 1 \quad \text{implies} \quad R(x, z) = 1. \end{aligned}$$

Let $[x]$ denote the class of equivalence of x (i.e., $x = \{y \,|\, R(x, y) = 1\}$). It is well known that distinct classes are disjoint and, moreover, classes of equivalence form a partition of X. Let A be crisp subset of X. We define the lower approximation and the upper approximation of A by

$$\underline{R}(A) = \bigcup_{[x] \subset A} [x] \quad \text{and} \quad \overline{R}(A) = \bigcup_{[x] \cap A \neq \emptyset} [x].$$

A rough set is defined as a representation of a set A by its lower approximation and its upper approximation. Thus, as in fuzzy sets, the boundary is not sharp; it is caught somewhere between the lower and the upper approximation. Generalization of this concept can take two forms: The set A is fuzzy or the set A is crisp but the relation R is fuzzy. If R is a fuzzy relation, it is taken to be a similarity relation; that is,

$$\begin{aligned} &R(x, x) = 1 \quad \text{for all} \quad x \in X, \\ &R(x, y) = 1 \quad \text{implies} \quad R(y, x) = 1, \\ &R(x, z) \geq \max_y \min\{R(x, y), R(y, z)\}. \end{aligned}$$

If R is a similarity relation, we define classes of similarity. If $x \in X$, then $[x]$ denotes the function whose value at y is $R(x, y)$. Of course, it is not true any more that distinct classes are disjoint. In general, if R and A are crisp, then

$$\underline{R}(A) \subset A \subset \overline{R}.$$

If $\underline{R}(A) = A = \overline{R}(A)$, then A is said to be definable. Rough sets are naturally generalized in the context of an information system. By an information system we mean the quadruple $< X, Q, V, \rho >$, where Q is partitioned into two sets C and D (conditions and decisions). The set V is the set of values and ρ is a function from $X \times Q$ into V, where the interpretation of $\rho(x, q)$ is the value of q for x. Table 2.1 illustrates these concepts.

A possible interpretation is the following: There are five patients $x_1, \cdots, x_5$. Two conditions denoted by c_1 and c_2 are observed in these patients. In column c_1, the values 0 and 1 denote the absence or presence of some symptom. In column c_2, L and S denote a large or a small presence of a second symptom.

	C		D
	c_1	c_2	d
x_1	0	L	0
x_2	1	S	1
x_3	1	S	0
x_4	1	L	1
x_5	0	L	0

Table 2.1: patients-symptoms-disease

In the column d, the values 0 and 1 denote the absence or the presence of some disease. The space X is then partitioned into sets of patients having the same symptoms. The partition is $[x_1] = [x_5] = \{x_1, x_5\}$, $[x_2] = [x_3] = \{x_2, x_3\}$, and $[x_4] = \{x_4\}$. Let A denote the set of sick patients. It is clear that A is not definable in terms of symptoms, as x_2 and x_3 have the same symptoms but $x_2 \in A$ and $x_3 \notin A$. How does such a discrepancy come about? Evidently, the physician has used indicators other than c_1 and c_2 to make such a judgment. It may even be the case that the physician is not conscious of how such a judgment was made. Thus, A is not definable in terms of c_1 and c_2 and the boundary of A is not clearly determined. In this case,

$$A = \{x_2, x_4\}, \quad \underline{R}(A) = \{x_4\}, \quad \overline{R}(A) = \{x_2, x_3, x_4\}.$$

It is precisely the size of the set $\overline{R}(A) - \underline{R}(A)$ that reflects the nondefinibility of A in terms of the conditions of Q. In that sense, the lower and the upper approximation of A play a role somewhat analogous to $\mathrm{Bel}(A)$ and $\mathrm{Pls}(A)$, where $\mathrm{Pls}(A) - \mathrm{Bel}(A)$ is a reflection of the uncertainty of A. It should be noted that whereas A is not definable in terms of the conditions, $\underline{R}(A)$ and $\overline{R}(A)$ are, in fact,

$$\underline{R}(A) = \{x_i \,|\, c_1 = 1,\, c_2 = L\},$$
$$\overline{R}(A) = \{x_i \,|\, c_1 = 1,\, c_2 = S\} \cap \{x_i \,|\, c_1 = 1,\, c_2 = L\}.$$

2.5.2 Two Operations on Interval Type 2 Sets

The purpose of this subsection is to illustrate by a simple example how one can extract rules from examples using rough sets. The example deals with linking symptoms to diagnosis. In Section 2.2.3 we have seen a number of ways to define $A \to B$, where A and B are fuzzy sets. In this example, for illustrative purposes, we define

$$(A \to B)(x) = (1 - A(x)) \vee B(x).$$

Unfortunately, uncertainty is all too often present in symptoms as well as the diagnosis. Symptoms and diagnosis fail to partition their universes of

discourses and overlaps are often present. We now define two operators on fuzzy sets that will be of importance in this subsection. Let A and B be fuzzy sets. We set

$$I(A,B) = \min_x \max\{1 - A(x), B(x)\},$$
$$J(A\#B) = \max_x \min\{A(x), B(x)\}.$$

Note that if A and B are crisp sets, then

$$I(A,B) = \begin{cases} 1 \text{ if } A \subset B \\ 0 \text{ otherwise} \end{cases}$$

and

$$J(A\#B) = \begin{cases} 1 \text{ if } A \cap B \neq \emptyset \\ 0 \text{ otherwise.} \end{cases}$$

Thus, $I(A,B)$ and $J(A,B)$ should intuitively measure how much A is a subset of B and much A intersects B (when A and B are fuzzy). One could also rewrite $I(A,B)$ and $J(A\#B)$ as

$$I(A,B) = \min(A \to B)(x) \quad \text{and} \quad J(A\#B) = \text{Poss}(A,B).$$

It can also be easily checked that

$$I(A,B) = 1 - J(A\#\bar{B}).$$

In a number of situations it might be difficult to exactly determine the membership functions of sets A and B. We thus assume now that A and B are interval type 2 sets; for example,

$$A = \sum [a_i, b_i]/x_i.$$

Thus, the membership of x_i in A is some unknown value in the interval $[a_i, b_i]$. We use interval arithmetic to define the complement of A as

$$1 \ominus A = \sum [1 - b_i, 1 - a_i]/x_i.$$

We also define

$$\tilde{\max}\{[a_i, b_i], [a_j, b_j]\} = [\max(a_i, a_j), \max(b_i, b_j)],$$
$$\tilde{\min}\{[a_i, b_i], [a_j, b_j]\} = [\min(a_i, a_j), \min(b_i, b_j)].$$

Then when A and B are interval type 2 sets, we extend the operators I and J as follows:

$$\tilde{I}(A,B) = \tilde{\min}_x \tilde{\max}(1 \ominus A(x), B(x)),$$
$$\tilde{J}(A\#B) = \tilde{\max}_x \tilde{\min}(A(x), B(x)).$$

Note that for fixed x, $1 \ominus A(x)$ and $B(x)$ are intervals. Thus, $\tilde{\max}(1 \ominus A(x), B(x))$ is an interval that depends on x and, therefore, $\tilde{\min}_x \tilde{\max}(1 \ominus A(x), B(x))$ is also an interval (that does not depend on x). The same observation holds for $\tilde{J}$. Thus, in this more general case, the degree to which A is a subset of B and the degree to which A and B intersect are subintervals of interval $[0, 1]$.

2.5.3 Extracting Rules from Data

We illustrate rule extraction from data by a simple example of two conditions. Condition 1 is the size of a tumor, which could be large or small. Condition 2 is the texture of the tumor, which can be hard or pliable. The diagnosis is that patient is in stage A or stage B. Of course, no sharp boundaries exist between large and small, between hard and pliable, and between stage A and stage B. Intuitively speaking, the operators $\tilde{I}$ and $\tilde{J}$ will generate the lower and the upper approximations to stages A and B. The conditions and the corresponding stages on five patients are shown in the table below.

Patient	Condition 1	Condition 2	Diagnosis
x_1	[0.2,0.4]/L+[0.7,0.9]/S	[0.1,0.3]/H+[0.8,1]/P	[0.2,0.4]/D_A+[0.5,0.7]/D_B
x_2	[0.3,0.5]/L+[0.6,0.8]/S	[0.9,1]/H+[0.6,0.9]/P	[0.6,0.9]/D_A+[0.4,0.6]/D_B
x_3	[0.6,0.9]/L+[0.2,0.5]/S	[0.5,0.7]/H+[0.5,0.7]/P	[0.5,0.6]/D_A+[0.8,0.9]/D_B
x_4	[0.7,0.8]/L+[0.4,0.6]/S	[0.2,0.4]/H+[0.7,0.8]/P	[0.5,0.8]/D_A+[0.1,0.2]/D_B
x_5	[0.1,0.4]/L+[0.6,0.7]/S	[0.1,0.2]/H+[0.5,0.9]/P	[0.3,0.4]/D_A+[0.1,0.2]/D_B

This above table generates the following interval type 2 sets:

$$
\begin{aligned}
L &= [0.2, 0.4]/x_1 + [0.3, 0.5]/x_2 + [0.6, 0.9]/x_3 + [0.7, 0.8]/x_4 + [0.1, 0.4]/x_5,\\
S &= [0.7, 0.9]/x_1 + [0.6, 0.8]/x_2 + [0.2, 0.5]/x_3 + [0.4, 0.6]/x_4 + [0.6, 0.7]/x_5,\\
H &= [0.1, 0.3]/x_1 + [0.9, 1]/x_2 + [0.5, 0.7]/x_3 + [0.2, 0.4]/x_4 + [0.1, 0.2]/x_5,\\
P &= [0.8, 1]/x_1 + [0.6, 0.9]/x_2 + [0.5, 0.7]/x_3 + [0.7, 0.8]/x_4 + [0.5, 0.9]/x_5,\\
D_A &= [0.2, 0.4]/x_1 + [0.6, 0.9]/x_2 + [0.5, 0.6]/x_3 + [0.5, 0.8]/x_4 + [0.3, 0.4]/x_5,\\
D_B &= [0.5, 0.7]/x_1 + [0.4, 0.6]/x_2 + [0.8, 0.9]/x_3 + [0.1, 0.2]/x_4 + [0.1, 0.2]/x_5.
\end{aligned}
$$

Thus, conditions and diagnoses become interval type 2 sets over the universe of patients. The interpretation is straightforward; for example, x_3 is an example of large tumor L with membership between 0.6 and 0.9. These sets in turn generate rules. For example,

$$
\begin{aligned}
L \cap H &= [0.1, 0.3]/x_1 + [0.3, 0.5]/x_2 + [0.5, 0.7]/x_3 + [0.2, 0.4]/x_4\\
&\quad + [0.1, 0.2]/x_5,\\
\tilde{\max}(1 \ominus (L \cap H), D_A) &=\\
&[0.7, 0.9]/x_1 + [0.6, 0.9]/x_2 + [0.5, 0.6]/x_3 + [0.6, 0.8]/x_4 + [0.8, 0.9]/x_5,
\end{aligned}
$$

and, therefore, $\tilde{I}((L \cap H), D_A) = [0.5, 0.6]$. This generates the following certain rule (lower approximation): If the tumor is large and hard, the patient's stage is A with belief between 0.5 and 0.6. Proceeding in this way, a total of eight certain (or belief) rules would be collected (four for D_A and four for D_B). A similar computation would produce $\tilde{J}((L \cap P) \# D_A) = [0.5, 0.8]$ and generate the possible rule: If the tumor is large and pliable, the patient's stage is A with possibility between 0.5 and 0.8. This way a total of 16 rules would be generated. Eight of them are certain and eight of them are possible.

For additional information on rough sets, see [35], and for combinations of rough sets and fuzzy sets see, [3].

2.6 Genetic Algorithms

2.6.1 Introduction

Genetic algorithms imitate natural evolution to solve optimization problems by finding a good solution through a search procedure. The basic construct of a generic algorithm is a chromosome that encodes a possible solution. An initial population of chromosomes is initiated and the population is evolved through three operations: reproduction, crossover, and mutation. The reproduction involves the selection of chromosomes that generate high values for the fitness function. A number of steps need to be taken to run a generic algorithm. The first step is to pick a representation for possible solutions. Typically, potential solutions are represented by a string of numbers and/or characters. Such a string is called a chromosome. For example, a neural net could be represented as a string of weights and biases. Thus, the corresponding chromosome would be a string of numbers. The fitness function measures how good a solution represented by a chromosome is. For neural nets, an appropriate fitness function could be defined through some training set $\{(I_1, d_1), \ldots, (I_m, d_m)\}$, where d_i denotes the desired target when input I_i is presented, and the fitness function might be defined as

$$F(\text{SpecificNet}) = -\sum_{i=1}^{m}(d_i - a_i)^2,$$

where a_i is the activation of the net for input I_i.
Reproduction: Given an initial population, chromosomes are selected with probability proportional to their fitness. The selected chromosomes are copied into a set (known as the mating pool).
Crossover: Pairs of chromosomes in the mating pool are selected randomly and corresponding elements of each are swapped randomly. The diagram in Figure 2.33 shows a commonly used crossover.

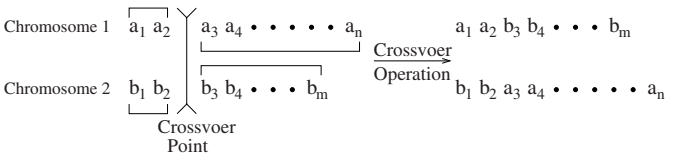

Fig. 2.33

Typically, crossover takes place with some probability and the crossover point is chosen randomly.

em Mutation: After crossover, each element of the string is changed with small probability. Thus, the final result after crossover is performed might be as shown in Figure 2.34.

$a_1\ a_2\ b_3\ a_4\ b_5 \bullet \bullet \bullet b_m$

$b_1\ b_2\ a_3\ a_4 \bullet \bullet \bullet \bullet \bullet \bullet a_n$

Fig. 2.34

In this example the first string obtained after crossover was subject to a mutation, the b_4 element having changed into a_4 while no mutation took place in the second string.

A typical generic algorithm can be outlined as follows:
1. Create a random initial population of chromosomes (representing potential solutions).

2. Repeat until some termination criterion is met:
   a. Evaluate the fitness of each chromosome.
   b. Reproduce (i.e., select chromosomes with probability proportional to their fitness and enter them in the mating pool).
   c. Crossover pair.
   d. Perform mutation (with small probability).
These new individuals form the new mating pool. Go back to step 2.

The termination criterion could be reached when the best chromosome in the mating pool is fit enough and/or the number of iterations reaches a specified number.

2.6.2 Designing a Fuzzy System

The success of the previously outlined algorithm depends on a number of factors. A key factor is a viable encoding of potential solutions into chromosomes. We illustrate a possible way to encode fuzzy systems (i.e., fuzzy sets of rules). When designing a fuzzy system, it is often not obvious how to partition the input and output spaces, that is, it is not obvious how many rules to use. Typically, the general shape of the membership functions is determined but the exact functions are not, as they depend on parameters. For example, we may decide to use trapezoidal functions, but the widths of their bases could be specified as unknown parameters. A possible chromosome corresponding to a set of fuzzy rules could be constructed as a string consisting of three pairs: the input parameters, the output parameters, and the rule identifiers. To keep the example simple, assume we are dealing with rules of the following form: If x is A, then y is B (i.e., one input, one output). A typical chromosome could be as shown in Figure 2.35.

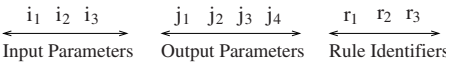

Fig. 2.35

Assume i_1, i_2, and i_3 are the values of the parameters identifying the fuzzy sets A_1, A_2, and A_3, respectively. The values j_1, j_2, j_3, and j_4 identify the fuzzy sets B_1, B_2, B_3, and B_4, respectively. Cells labeled r_1, r_2, and r_3 refer to the rules 1, 2, and 3, respectively. A value $r_i = 0$ means rule i should be taken out.

Let $r_1 = 2$, $r_2 = 0$, and $r_3 = 4$. This chromosome corresponds to the following set of rules:

If x is A_1, then y is B_2.
If x is A_3, then y is B_4.

Assume this chromosome is paired off with the chromosome

$$i'_1\ i'_2\ i'_3\ i'_4\ i'_5\ j'_1\ j'_2\ r'_1\ r'_2\ r'_3\ r'_4\ r'_5,$$

where $i'_1, \cdots, i'_5$ are values of parameters determining the fuzzy sets $A'_1, \cdots, A'_5$, j'_1 and j'_2 determine B'_1 and B'_2, respectively, and $r'_1 = 1$, $r'_2 = 2$, $r'_3 = 1$, $r'_4 = 0$, and $r'_5 = 2$. Then the corresponding system is
If x is A'_1, then y is B'_1.
If x is A'_2, then y is B'_2.
If x is A'_3, then y is B'_1.
If x is A'_5, then y is B'_2.

If we pick a crosspoint between position 2 and 3, after crossover we have the chromosomes

$$i_1\ i_2\ i'_3\ i'_4\ i'_5\ j'_1\ j'_2\ r'_1\ r'_2\ r'_3\ r'_4\ r'_5$$

and

$$i'_1\ i'_2\ i_3\ j_1\ j_2\ j_3\ j_4\ r_1\ r_2\ r_3.$$

Perform a mutation on the first chromosome changing $r'_3 = 1$ to $r'_3 = 0$. The corresponding systems are as following:

If x is A_1, then y is B'_1.
If x is A_2, then y is B'_2.
If x is A'_5, then y is B'_2

and

If x is A'_1, then y is B_2.
If x is A_3, then y is B_4.

The fitness function could be the negative of the error made by the system on a specified training set. After a number of iterations, the best chromosome in the current mating pool would generate the best current fuzzy system.

For more detailed information on genetic algorithms, see [12], and for combining genetic algorithms with fuzzy logic, see [24].

2.7 Conclusion

We presented some of the important methodologies of soft computing. These important tools include neural nets, fuzzy logic, neuro-fuzzy systems, the theory of evidence, rough sets, and genetic algorithms. The main idea that should emerge from this overview is that these methods are not competing but are complementary methods. Adaptive nets use input/output specifications. Such nets form a black box and there is usually no clear indication on how the decisions are made. The advantage of this methodology is that the system learns to adapt as input/output specifications are defined. Fuzzy logic, on the other hand, is a very convenient tool to represent knowledge as a set of rules. Neuro-fuzzy systems combine the advantages of the two previous methods and seamlessly integrate linguistic and numerical information. Two applications were presented: forecasting time series and designing a controller. Although we indicated several ways to construct membership functions, in certain cases such constructions might be difficult to make and some uncertainty regarding membership functions must be taken into consideration. Fuzzy sets of type 2 is a possible approach to this problem.

Fuzzy sets, neural nets, and neuro-fuzzy systems were then compared as possible approaches to designing an autonomous vehicle. The theory of evidence was then presented. This methodology has a built-in capability to use the following type of rules:

If (object$_i$, atribute$_i$, value$_i$), $i = 1, \ldots, n$, then (decision$_1$,...,decision$_k$) (x confirm).

Here, $x \in [0, 1]$. Such a rule generates a mass m. If the antecedent is satisfied, then $m\{\text{decision}_1, \ldots, \text{decision}_k\} = x$. A set of such rules generates a set of masses, and composing these masses yields a mass representing the set of rules. That resulting mass highlights the appropriate decision to consider. Nonspecificity, which is an uncertainty related to how large the set of choices is was discussed in the context of evidence theory. Similarly, the concept of strife, that is the uncertainty related to the conflict between choices, was also cast in the concept of evidence theory. Rough sets were then introduced. As with fuzzy sets, the boundary of a rough set is not a crisp set. There is a lower and an upper approximation. In was then shown how a diagnosis problem in which no sharp boundaries exist between conditions and between diagnoses could generate rules by use of the rough sets methodology. We then have two

types of rules: the certain and the possible rules corresponding to lower and upper approximations. Finally, genetic algorithms were introduced. It was shown how a good set of fuzzy rules could be obtained for some specified problem using genetic algorithms.

It is clear that complex problems typically require hybrid methods as the sensible approach. For example, in certain cases it may be convenient to construct the fuzzy part of a neuro-fuzzy system using genetic algorithms and then evolve the neural part using, for example, the swarm method. Future directions might also incorporate the methodology of fuzzy sets of type 2 into such hybrid methods. For hybrid methods and various approaches to uncertainty, we refer the reader to [17] and [20].

Aknowledgment: The authors would like to thank Dr. Youn Sha-Chan who created all figures in this chapter.

References

1. Bandler, W., Kohout, L.J.: On the general theory of relational morphisms. International Journal of General Systems 13, 47–68 (1986)
2. Berenji, H.R., Khedkar, P.: Learning and tuning fuzzy logic controllers through reinforcements. IEEE Transactions on Neural Networks 3(5), 724–740 (1992)
3. Dubois, D., Prade, H.: Putting rough sets and fuzzy sets together. In: R. Slovinski (ed.) Intelligent Decision Support, pp. 203–232. Kluwer Academic Publishers, Norwell, MA (1992)
4. Eberhart, R.C., Shi, Y.: Particle swarm optimization: Developments, applications, and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, pp. 81–86. IEEE Press, Los Alamitos, CA (2001)
5. Grossberg, S.: A neural model of attention, reinforcement and discrimination learning. International Review of Neurobiology 18, 263–327 (1975)
6. Hagan, M., Demuth, H., Beale, M.: Neural Network Design. PWS Publishing Company, Boston, MA (1996)
7. Harmanec, D., Klir, G.J.: Measuring total uncertainty in Dempster-Shafer theory: A novel approach. International Journal of General Systems 22(4), 405–419 (1994)
8. Haykin, S.: Neural Networks: A Comprehensive Foundation. MacMillan, New York (1994)
9. Hecht–Nielsen, R.: Counterpropagation networks. In: M. Caudill, C. Butler (eds.) IEEE First International Conference on Neural Networks (ICNN'87), Vol. II, pp. II-19–32. IEEE, San Diego, CA (1987)
10. Hellendorn H., Thomas C.: Defuzzification in fuzzy controllers. Journal of Intelligent and Fuzzy Systems, 1(2), 109-123 (1993)
11. Hinton, G., Sejnowski, T.: Learning and relearning in Boltzmann machines. In: Rummelhart, D., McClelland, J. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, pp. 283–335, MIT Press, Cambridge, MA. (1986)
12. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

13. Jang, J.S.R.: Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In: Proceedings of the Ninth National Conference on Artificial Inteligence (AAAI–91), pp. 762–767 (1991)
14. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics 23, 665–684 (1993)
15. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing. Prentice-Hall, Englewood Cliff, NJ (1997)
16. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Service Center, Piscataway, NJ (1995)
17. Klir, G.J.: Developments in uncertainty-based information. Advances in Computers 36, 255–332 (1993)
18. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic. Prentice-Hall, Englewood Cliff, NJ (1995)
19. de Korvin, A., Deeba, E., Kleyle, R.: Knowledge acquisition using rough sets when membership values are intervals. Mathematical Modeling and Scientific Computing 1, 470–479 (1993)
20. de Korvin, A., Hashemi, S., Sirisaengtaksin, O.: A body of evidence approach under partially specified environment. Journal of Neural, Parallel and Scientific Computation 13, 91-106 (2005)
21. de Korvin, A., Kleyle, R., Lea, R.: An evidence approach to problem solving when a large number of knowledge systems are available. International Journal of Intelligent Systems 5, 293–306 (1990)
22. de Korvin, A., Modave, F., Kleyle, R.: Paradigms for decision making under increasing levels of uncertainty. International Journal of Pure and Applied Mathematics 21, 419–430 (2005)
23. Kosko, B.: Bidirectional associative memories. IEEE Transactions on Systems, Man, and Cybernetics 18, 49–60 (1988)
24. Lee, M.A., Takagi, H.: Dynamic control of genetic algorithms using fuzzy logic techniques. In: S. Forrest (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 76–83. Morgan Kaufmann, San Mateo, CA (1993)
25. Mabuchi, S.: A proposal for defuzzification strategy by the concept of sensitivity analysis. Fuzzy Sets and Systems 55, 1–14 (1993)
26. Mamdami, E.H., Gaines, B.R. (eds.): Fuzzy Reasoning and Its Applications. Academic Press, London (1981)
27. Mamdani, E.H.: Applications of fuzzy logic to approximate reasoning using linguistic systems. IEEE Transactions on Computing 26, 1182–1191 (1977)
28. Mendel, J.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper Saddle River, NJ (2001)
29. Mendel, J.: On the importance of interval sets in type-2 fuzzy logic systems. In: Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 1647–1652 (2001)
30. Mendel, J., John, R.: Type-2 fuzzy sets made simple. IEEE Transactions on Fuzzy Systems 10(2), 117–127 (2002)
31. Mizutani, E., Jang, J.S.R.: Coactive neural fuzzy modeling. In: IEEE International Conference on Neural Networks (ICNN'95), Vol. 2, pp. 760–765. IEEE, Perth, Western Australia (1995)
32. Mizutani, E., Jang, J. S. R., Nishio, K., Takagi, H., Auslander, D. M.: Coactive neural networks with adjustable fuzzy membership functions and their appli-

cations. In: International Conference on Fuzzy Logic and Neural Networks, pp. 581–582 (1994)
33. Mrózek, A.: Rough sets and some aspects of expert systems realization. In: Seventh Workshop on Expert Systems and their Applications, pp. 597–611 (1987)
34. Ovchinikov S. V.: Representation of transitive fuzzy relations. In: Skala, Termini, and Trillas (eds.) Aspects of Vagueness, 105–118, Boston, MA (1984)
35. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
36. Ruan, D., Kerre, E.E.: Fuzzy implication operators and generalized fuzzy method of cases. Fuzzy Sets Systems 54(1), 23–37 (1993)
37. Saaty, T.L.: Modeling unstructured decision problems: A theory of analytical hierarchies. In: Proceedings of the First International Conference on Mathematical Modeling, University of Missouri- Rolla, Vol. 1, pp. 59–77 (1977)
38. Saaty, T.L.: The Analytic Hierarchy Process. McGraw–Hill, New York (1980)
39. Skapura, D.M.: Building Neural Networks. ACM Press/Addison–Wesley Publishing Co., New York, NY, USA (1995)
40. Takagi, H., Hayashi, I.: NN–driven fuzzy reasoning. International Journal Approximate Reasoning 5, 191–212 (1991)
41. Takagi, H.: Fusion techniques of fuzzy systems and neural networks, and fuzzy systems and genetic algorithms. In: B. Bosacchi, J.C. Bezdek (eds.) Applications of Fuzzy Logic Technology, Proc. of the Society of Photo–Optical Instrumentation Engineers (SPIE) Conference, SPIE Vol. 2061, pp. 402–413 (1993)
42. Tong, R.M.: An annotated bibliography of fuzzy control. In: M. Sugeno (ed.) Industrial Applications of Fuzzy Control, pp. 249–269. Elsevier Science Publishers, Amsterdam (1985)
43. Weber, S.: A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. Fuzzy Sets and Systems 11, 115–134 (1983)
44. Yager, R.R.: Approximate reasoning as a basis for rule based expert systems. IEEE Transactions on Systems, Man and Cybernetics 14 (1984)
45. Yager, R.R., Filev, D.P.: On the issue of defuzzification and selection based on a fuzzy set. Fuzzy Sets and Systems 55, 255–273 (1993)

http://www.springer.com/978-1-84800-325-5

Knowledge Processing with Interval and Soft Computing
Hu, C.; Kearfott, R.B.; de Korvin, A.; Kreinovich, V. (Eds.)
2008, XII, 233 p. 61 illus., Hardcover
ISBN: 978-1-84800-325-5