

SOFT COMPUTING

Syllabus

Course No.	Course Name	L-T-P Credits	Year of Introduction
CS361	SOFT COMPUTING	3-0-0-3	2015
Course Objectives To introduce the concepts in Soft Computing such as Artificial Neural Networks, Fuzzy logic based systems, genetic algorithm-based systems and their hybrids.			
Syllabus Introduction to Soft Computing, Artificial Neural Networks, Fuzzy Logic and Fuzzy systems, Genetic Algorithms, hybrid systems.			
Expected Outcome Student is able to <ol style="list-style-type: none"> 1. Learn about soft computing techniques and their applications. 2. Analyze various neural network architectures. 3. Define the fuzzy systems. 4. Understand the genetic algorithm concepts and their applications. 5. Identify and select a suitable Soft Computing technology to solve the problem; construct a Solution and implement a Soft Computing solution. 			
Text Books <ol style="list-style-type: none"> 1. S.N.Sivanandam and S.N.Deepa, Principles of soft computing-Wiley India. 2. Timothy J. Ross, Fuzzy Logic with engineering applications-Wiley India. 			
References <ol style="list-style-type: none"> 1. N. K. Sinha and M. M. Gupta, Soft Computing & Intelligent Systems: Theory & Applications-Academic Press /Elsevier. 2009. 2. Simon Haykin, Neural Network- A Comprehensive Foundation- Prentice Hall International, Inc. 3. R. Eberhart and Y. Shi, Computational Intelligence: Concepts to Implementation, Morgan Kaufman/Elsevier, 2007. 4. Ross T.J. , Fuzzy Logic with Engineering Applications- McGraw Hill. 5. Driankov D., Hellendoorn H. and Reinfrank M., An Introduction to Fuzzy Control- Narosa Pub. 6. Bart Kosko, Neural Network and Fuzzy Systems- Prentice Hall, Inc., Englewood Cliffs 7. Goldberg D.E., Genetic Algorithms in Search, Optimization, and Machine Learning AddisonWesley. 			

Course Plan			
Module	Contents	Hours	Sem.Exam Marks%
I	Introduction to Soft Computing Artificial neural networks - biological neurons, Basic models of artificial neural networks – Connections, Learning, Activation Functions, McCulloch and Pitts Neuron, Hebb network.	08	15%
II	Perceptron networks – Learning rule – Training and testing algorithm, Adaptive Linear Neuron, Back propagation Network – Architecture, Training algorithm.	08	15%
FIRST INTERNAL EXAM			
III	Fuzzy logic - fuzzy sets - properties - operations on fuzzy sets, fuzzy relations - operations on fuzzy relations.	07	15%
IV	Fuzzy membership functions, fuzzification, Methods of membership value assignments – intuition – inference – rank ordering, Lambda –cuts for fuzzy sets, Defuzzification methods.	07	15%
SECOND INTERNAL EXAM			
V	Truth values and Tables in Fuzzy Logic, Fuzzy propositions, Formation of fuzzy rules - Decomposition of rules – Aggregation of rules, Fuzzy Inference Systems - Mamdani and Sugeno types, Neuro-fuzzy hybrid systems – characteristics – classification.	08	20%
VI	Introduction to genetic algorithm, operators in genetic algorithm - coding - selection - cross over – mutation, Stopping condition for genetic algorithm flow, Genetic neuro hybrid systems, Genetic-Fuzzy rule based system.	08	20%
END SEMESTER EXAMINATION			

Question Paper Pattern

1. There will be five parts in the question paper – A, B, C, D, E

2. Part A

a. Total marks : 12

b. **Four** questions each having 3 marks, uniformly covering modules I and II; All four questions have to be answered.

3. Part B

a. Total marks : 18

b. **Three** questions each having 9 marks, uniformly covering modules I and II; Two questions have to be answered. Each question can have a maximum of three subparts

4. Part C

a. Total marks : 12

b. **Four** questions each having 3 marks, uniformly covering modules III and IV; All four questions have to be answered.

5. Part D

a. Total marks : 18

b. **Three** questions each having 9 marks, uniformly covering modules III and IV; Two questions have to be answered. Each question can have a maximum of three subparts

6. Part E

a. Total Marks: 40

b. **Six** questions each carrying 10 marks, uniformly covering modules V and VI; four questions have to be answered.

c. A question can have a maximum of three sub-parts.

7. There should be at least 60% analytical/numerical/design questions.

Module – 1

- Introduction to Soft Computing
- Artificial neural networks
- Biological neurons
- Basic models of artificial neural networks
 - Connections
 - Learning
 - Activation Functions
- McCulloch and Pitts Neuron
- Hebb network.

1.1 Introduction to Soft Computing

Two major problem solving techniques are:

- **Hard computing**

It deals with precise model where accurate solutions are achieved.

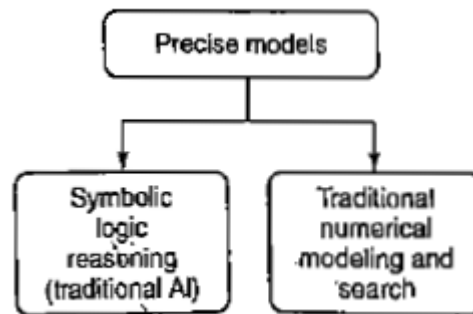


Figure 1.1: Hard Computing

- **Soft computing**

It deals with approximate model to give solution for complex problems. The term "soft computing" was introduced by Professor Lotfi Zadeh with the objective of exploiting the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness, low solution cost and better rapport with reality. The ultimate goal is to be able to emulate the human mind as closely as possible. It is a combination of Genetic Algorithm, Neural Network and Fuzzy Logic.

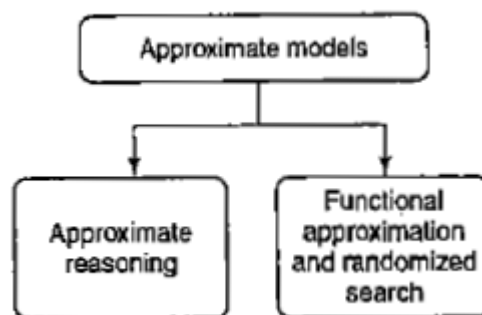


Figure 1.2: Soft Computing

1.2 Biological Neurons

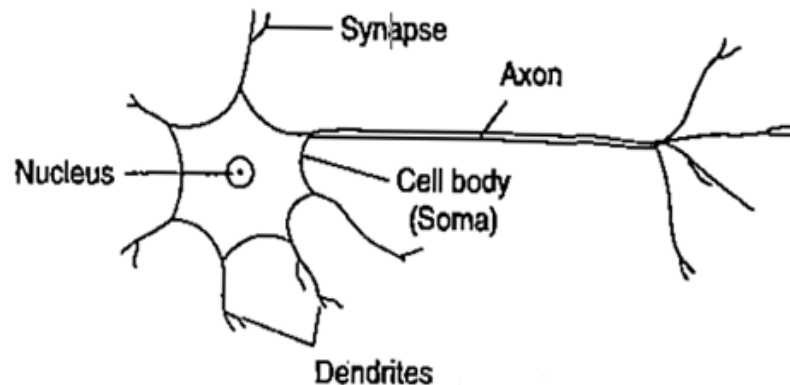


Figure 1.3: Schematic diagram of a biological neuron

The biological neuron consists of main three parts:

- **Soma or cell body**-where cell nucleus is located
- **Dendrites**-where the nerve is connected to the cell body
- **Axon**-which carries the impulses of the neuron

Dendrites are tree like networks made of nerve fiber connected to the cell body. An Axon is a single, long connection extending from the cell body and carrying signals from the neuron. The end of axon splits into fine strands. It is found that each strand terminated into small bulb like organs called as synapse. It is through synapse that the neuron introduces its signals to other nearby neurons. The receiving ends of these synapses on the nearby neurons can be found both on the dendrites and on the cell body. There are approximately 10^4 synapses per neuron in the human body. Electric impulse is passed between synapse and dendrites. It is a chemical process which results in increase/decrease in the electric potential inside the body of the receiving cell. If the electric potential reaches a thresh hold value, receiving cell fires & pulse / action potential of fixed strength and duration is send through the axon to synaptic junction of the cell. After that, cell has to wait for a period called refractory period.

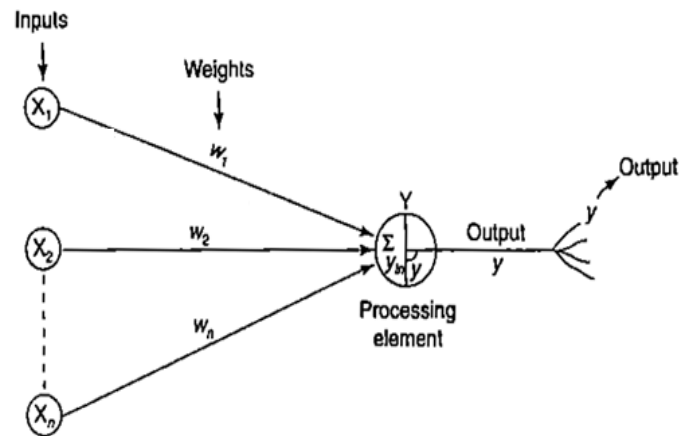


Figure 1.4: Mathematical model of artificial neuron

Biological neuron	Artificial neuron
Cell	Neuron
Dendrites	Weights or interconnections
Soma	Net input
Axon	Output

Table 1.1: Terminology relationships between biological and artificial neurons

In this model net input is calculated as

$$y_{in} = x_1w_1 + x_2w_2 + \dots + x_nw_n = \sum_{i=1}^n x_iw_i$$

Where, i represents i^{th} processing element. The activation function applied over it to calculate the output. The weight represents the strength of synapses connecting the input and output.

1.3 Artificial neural networks

An artificial neural network (ANN) is an efficient information processing system which resembles the characteristics of biological neural network. ANNs contain large number of highly interconnected processing elements called nodes or neurons or units. Each neuron is connected with other by connection link and each connection link is associated with weights which contain information about the input signal. This information is used by neuron net to solve a particular problem. ANNs have ability to learn, recall and generalize training pattern or

data similar to that of human brain. The ANN processing elements called neurons or artificial neurons.

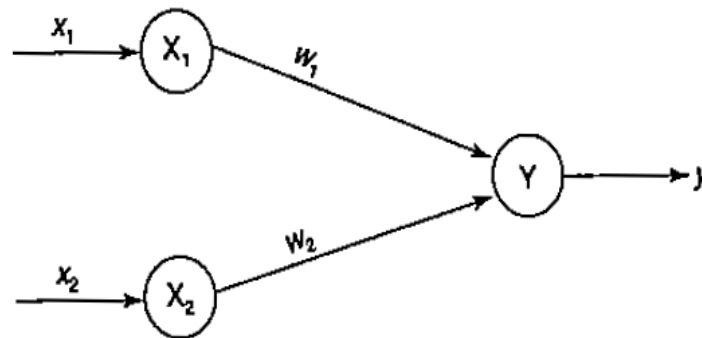


Figure 1.5: Architecture of a simple artificial neuron net

Each neuron has an internal state of its own, called activation or activity level of neuron which is the function of the inputs the neuron receives. The activation signal of a neuron is transmitted to other neurons. A neuron can send only one signal at a time which can be transmitted to several neurons.

Consider the figure 1.5, here X_1 and X_2 are input neurons, Y is the output neuron W_1 and W_2 are the weights net input is calculated as

$$y_{in} = x_1w_1 + x_2w_2$$

where x_1 and x_2 are the activation of the input neurons X_1 and X_2 , i.e., is the output of the input signals. The output y of the output neuron Y can be obtained by applying activations over the net input.

$$y = f(y_{in})$$

Output = Function (net input calculated)

The function to be applied over the net input is called activation function. The net input calculation is similar to the calculation of output of a pure linear straight line equation $y=mx$

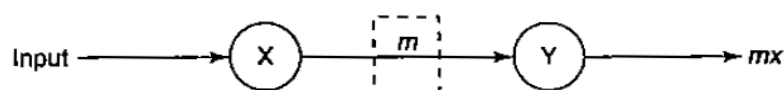
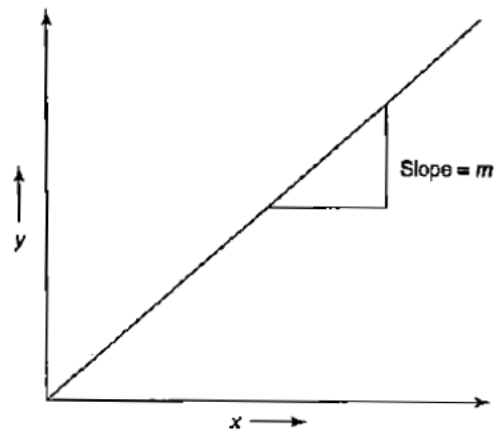


Figure 1.6: Neural net of pure linear equation

Figure 1.7: Graph for $y = mx$

The weight involve in the ANN is equivalent to the slope of the straight line.

1.4 Comparison between Biological neuron and Artificial neuron

Term	Brain	Computer
Speed	Execution time is few milliseconds	Execution time is few nano seconds
Processing	Perform massive parallel operations simultaneously	Perform several parallel operations simultaneously. It is faster the biological neuron
Size and complexity	Number of Neuron is 10^{11} and number of interconnections is 10^{15} . So complexity of brain is higher than computer	It depends on the chosen application and network designer.
Storage capacity	<ul style="list-style-type: none"> Information is stored in interconnections or in synapse strength. New information is stored without destroying old one. Sometimes fails to recollect information 	<ul style="list-style-type: none"> Stored in continuous memory location. Overloading may destroy older locations. Can be easily retrieved

Tolerance	<ul style="list-style-type: none"> • Fault tolerant • Store and retrieve information even interconnections fails • Accept redundancies 	<ul style="list-style-type: none"> • No fault tolerance • Information corrupted if the network connections disconnected. • No redundancies
Control mechanism	Depends on active chemicals and neuron connections are strong or weak	CPU Control mechanism is very simple

Table 1.2: Comparison between Biological neuron and Artificial neuron

Characteristics of ANN:

- It is a neurally implemented mathematical model
- Large number of processing elements called neurons exists here.
- Interconnections with weighted linkage hold informative knowledge.
- Input signals arrive at processing elements through connections and connecting weights.
- Processing elements can learn, recall and generalize from the given data.
- Computational power is determined by the collective behavior of neurons.
 - ANN is a connection models, parallel distributed processing models, self-organizing systems, neuro-computing systems and neuro - morphic system.

1.5 Evolution of neural networks

Year	Neural network	Designer	Description
1943	McCulloch and Pitts neuron	McCulloch and Pitts	Arrangement of neurons is combination of logic gate. Unique feature is thresh hold
1949	Hebb network	Hebb	If two neurons are active, then their connection strengths should be increased.
1958, 1959, 1962, 1988,	Perceptron	Frank Rosenblatt, Block, Minsky and Papert	Here the weights on the connection path can be adjusted.

1960	Adaline	Widrow and Hoff	Here the weights are adjusted to reduce the difference between the net input to the output unit and the desired output.
1972	Kohonen self-organizing feature map	Kohonen	Inputs are clustered to obtain a fired output neuron.
1982, 1984, 1985, 1986, 1987	Hopfield network	John Hopfield and Tank	Based on fixed weights. Can act as associative memory nets
1986	Back propagation network	Rumelhart, Hinton and Williams	<ul style="list-style-type: none"> • Multilayered • Error propagated backward from output to the hidden units
1988	Counter propagation network	Grossberg	Similar to kohonen network.
1987- 1990	Adaptive resonance Theory(ART)	Carpenter and Grossberg	Designed for binary and analog inputs.
1988	Radial basis function network	Broomhead and Lowe	Resemble back propagation network, but activation function used is Gaussian function.
1988	Neo cognitron	Fukushima	For character recognition.

Table 1.3: Evolution of neural networks

1.6 Basic models of artificial neural networks

Models are based on three entities

- The model's synaptic interconnections.
- The training or learning rules adopted for updating and adjusting the connection weights.
- Their activation functions

1.6.1 Connections

The arrangement of neurons to form layers and the connection pattern formed within and between layers is called the network architecture. There exist five basic types of connection architecture.

They are:

1. Single layer feed forward network
2. Multilayer feed-forward network
3. Single node with its own feedback
4. Single-layer recurrent network
5. Multilayer recurrent network

Feed forward network: If no neuron in the output layer is an input to a node in the same layer / proceeding layer.

Feedback network: If outputs are directed back as input to the processing elements in the same layer/proceeding layer.

Lateral feedback: If the output is directed back to the input of the same layer.

Recurrent networks: Are networks with feedback networks with closed loop.

1. Single layer feed forward network

Layer is formed by taking processing elements and combining it with other processing elements. Input and output are linked with each other Inputs are connected to the processing nodes with various weights, resulting in series of outputs one per node.

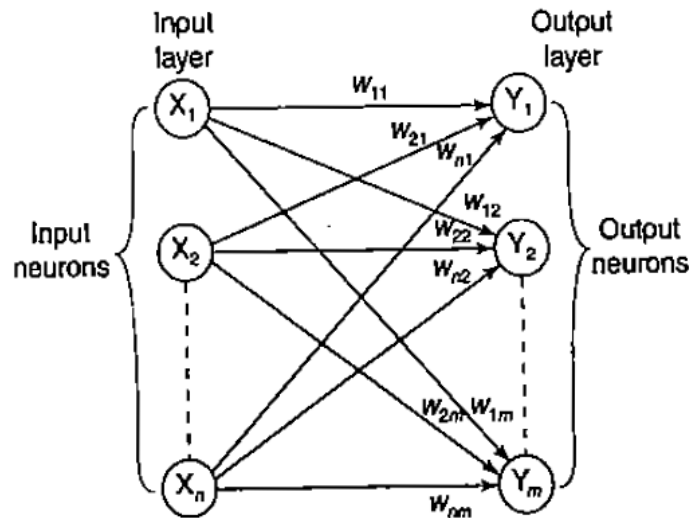


Figure 1.8: Single-layer feed-forward network

When a layer of processing nodes is formed the inputs can be connected to these nodes with various weights, resulting in a series of outputs, one per node. This is called single layer feedforward network.

2. Multilayer feed-forward network

This network is formed by the interconnection of several layers. Input layer receives input and buffers input signal. Output layer generated output. Layer between input and output is called hidden layer. Hidden layer is internal to the network. There are Zero to several hidden layers in a network. More the hidden layer more is the complexity of network, but efficient output is produced.

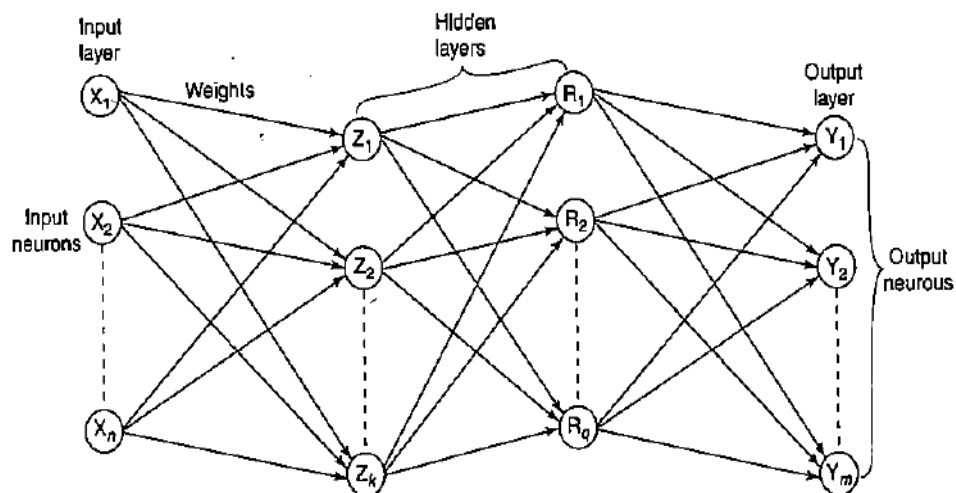


Figure 1.9: Multilayer feed-forward network

3. Single node with its own feedback

It is a simple recurrent neural network having a single neuron with feedback to itself.

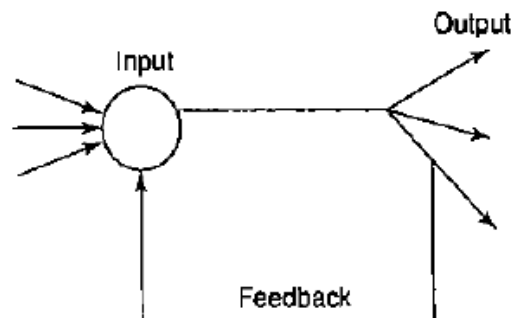


Figure 1.10: Single node with own feedback

4. Single layer recurrent network

A single layer network with feedback from output can be directed to processing element itself or to other processing element/both.

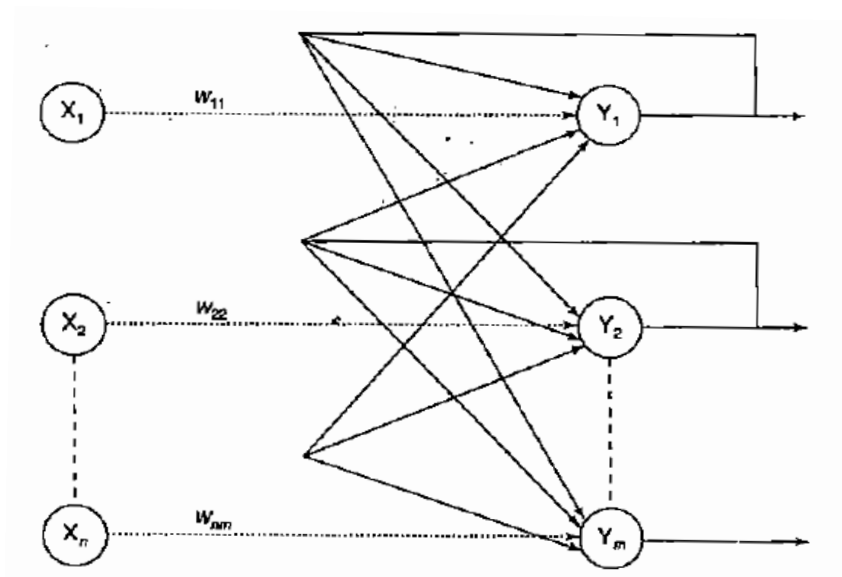


Figure 1.11: Single-layer recurrent network

5. Multilayer recurrent network

Processing element output can be directed back to the nodes in the preceding layer, forming a multilayer recurrent network.

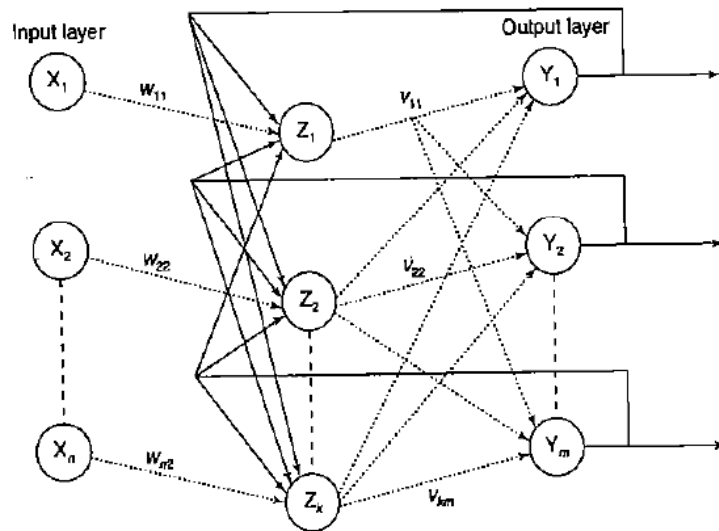


Figure 1.12: Multilayer recurrent network

- **Maxnet** –competitive interconnections having fixed weights.

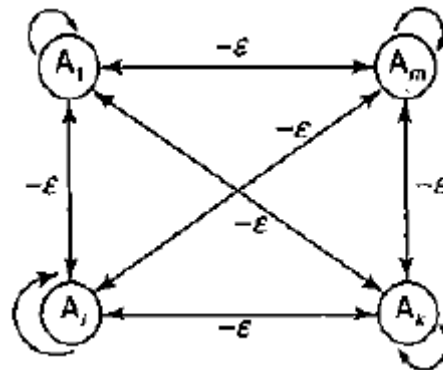


Figure 1.13: Competitive nets

- **On-center-off-surround/lateral inhibition structure** – each processing neuron receives two different classes of inputs- “excitatory” input from nearby processing elements & “inhibitory” elements from more distantly located processing elements. This type of interconnection is shown below

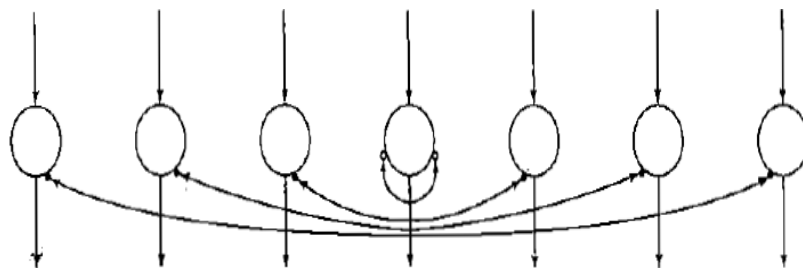


Figure 1.14: Lateral inhibition structure

1.6.2 Learning

Learning or Training is the process by means of which a neural network adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response.

Two broad kinds of learning in ANNs is:

- i) **Parameter learning** – updates connecting weights in a neural net.
- ii) **Structure learning** – focus on change in the network.

Apart from these, learning in ANN is classified into three categories as

- i) Supervised learning
- ii) Unsupervised learning
- iii) Reinforcement learning

i) Supervised learning

The Learning here is performed with the help of a teacher. Example: Consider the learning process of a small child. Child doesn't know how to read/write. Their each and every action is supervised by a teacher. Actually a child works on the basis of the output that he/she has to produce. In ANN, each input vector requires a corresponding target vector, which represents the desired output. The input vector along with target vector is called training pair. Input vector results in output vector. The actual output vector is compared with desired output vector. If there is a difference means an error signal is generated by the network. It is used for adjustment of weights until actual output matches desired output.

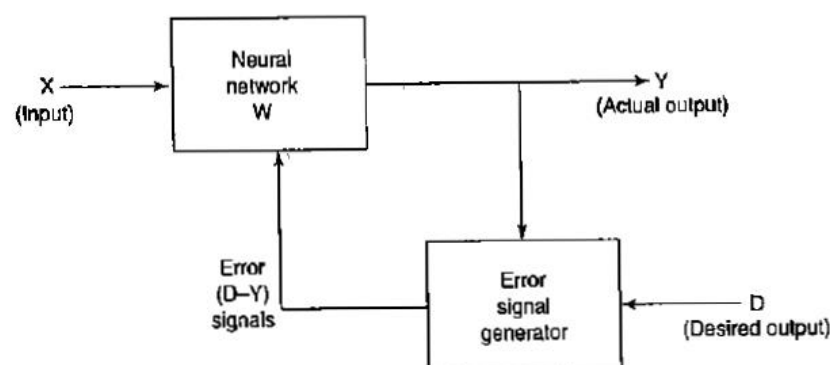


Figure 1.15: Supervised learning

ii) Unsupervised learning

Learning is performed without the help of a teacher. Example: tadpole – learn to swim by itself. In ANN, during training process, network receives input patterns and organize it to form clusters.

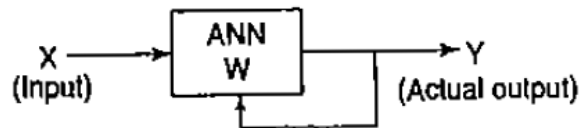


Figure 1.16: Unsupervised learning

From the above Fig.1.16 it is observed that no feedback is applied from environment to inform what output should be or whether they are correct. The network itself discover patterns, regularities, features/ categories from the input data and relations for the input data over the output. Exact clusters are formed by discovering similarities & dissimilarities so called as self – organizing.

iii) Reinforcement learning

It is similar to supervised learning. Learning based on critic information is called reinforcement learning & the feedback sent is called reinforcement signal. The network receives some feedback from the environment. Feedback is only evaluative.

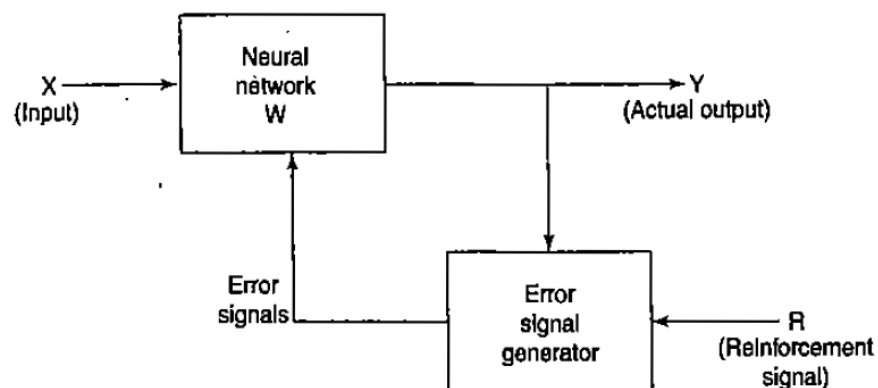


Figure 1.17: Reinforcement learning

The external reinforcement signals are processed in the critic signal generator, and the obtained critic signals are sent to the ANN for adjustment of weights properly to get critic feedback in future.

1.6.3 Activation Functions

To make work more efficient and for exact output, some force or activation is given. Like that, activation function is applied over the net input to calculate the output of an ANN. Information processing of processing element has two major parts: input and output. An integration function (f) is associated with input of processing element.

Several activation functions are there.

1. Identity function: It is a linear function which is defined as

$$f(x) = x \text{ for all } x$$

The output is same as the input.

2. Binary step function: This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$$

Where, θ represents thresh hold value. It is used in single layer nets to convert the net input to an output that is binary (0 or 1).

3. Bipolar step function: This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$$

Where, θ represents threshold value. It is used in single layer nets to convert the net input to an output that is bipolar (+1 or -1).

4. Sigmoid function: It is used in Back propagation nets.

Two types:

a) Binary sigmoid function: It is also termed as logistic sigmoid function or unipolar sigmoid function. It is defined as

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

where, λ represents steepness parameter. The derivative of this function is

$$f'(x) = \lambda f(x)[1 - f(x)]$$

The range of sigmoid function is 0 to 1.

b) Bipolar sigmoid function: This function is defined as

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$

Where λ represents steepness parameter and the sigmoid range is between -1 and +1.

The derivative of this function can be

$$f'(x) = \frac{\lambda}{2} [1 + f(x)][1 - f(x)]$$

It is closely related to hyperbolic tangent function, which is written as

$$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$h(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The derivative of the hyperbolic tangent function is

$$h'(x) = [1 + h(x)][1 - h(x)]$$

5. Ramp function: The ramp function is defined as

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$$

The graphical representation of all these function is given in the upcoming figure 1.18

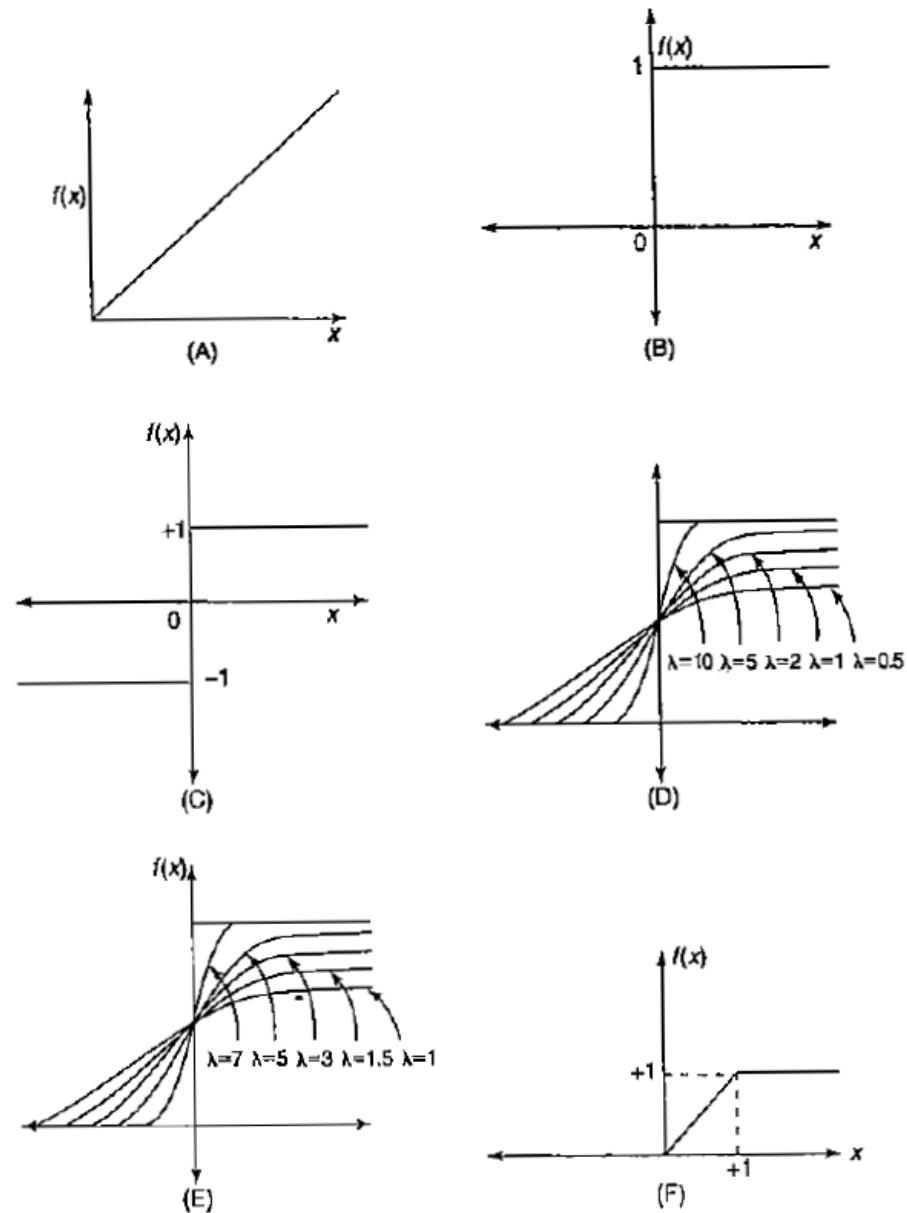


Figure 1.18: Depiction of activation functions: (A) identity function; (B) binary step function; (C) bipolar step function; (D) binary sigmoidal function; (E) bipolar sigmoidal function; (F) ramp function.

1.7 McCulloch and Pitts Neuron

It is discovered in 1943 and usually called as M-P neuron. M-P neurons are connected by directed weighted paths. Activation of M-P neurons is binary (i.e) at any time step the neuron may fire or may not fire. Weights associated with communication links may be excitatory (wgts are positive)/inhibitory (wgts are negative). Threshold plays major role here. There is a fixed threshold for each neuron and if the net input to the neuron is greater than the threshold then the neuron fires. They are widely used in logic functions. A simple M-P neuron is shown in the figure. It is excitatory with weight w ($w>0$) / inhibitory with weight $-p$ ($p<0$). In the Fig.,

inputs from x_1 to x_n possess excitatory weighted connection and x_{n+1} to x_{n+m} has inhibitory weighted interconnections.

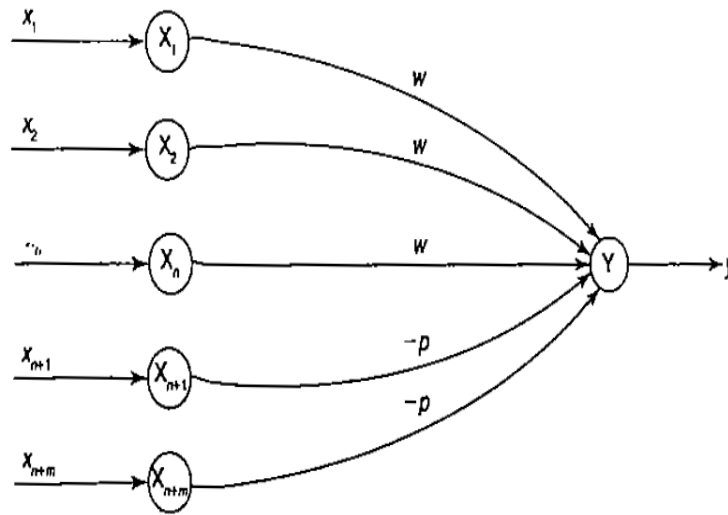


Figure 1.19: McCulloch-Pitts neuron model

Since the firing of neuron is based on threshold, activation function is defined as

$$f(x) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:

$$\theta > nw - p$$

Output will fire if it receives “k” or more excitatory inputs but no inhibitory inputs where

$$kw \geq \theta > (k-1)w$$

The M-P neuron has no particular training algorithm. An analysis is performed to determine the weights and the threshold. It is used as a building block where any function or phenomenon is modeled based on a logic function.

1.8 Hebb network

Donald Hebb stated in 1949 that “In brain, the learning is performed by the change in the synaptic gap”. When an axon of cell A is near enough to excite cell B, and repeatedly or permanently takes place in firing it, some growth process or metabolic change takes place in

one or both the cells such that A's efficiency, as one of the cells firing B, is increased. According to Hebb rule, the weight vector is found to increase proportionately to the product of the input and the learning signal. In Hebb learning, two interconnected neurons are 'on' simultaneously. The weight update in Hebb rule is given by

$$w_i(new) = w_i(old) + x_i y$$

Hebb's network is suited more for bipolar data. If binary data is used, the weight updation formula cannot distinguish two conditions namely:

1. A training pair in which an input unit is "on" and the target value is "off".
2. A training pair in which both the input unit and the target value is "off".

Training algorithm

The training algorithm is used for the calculation and adjustment of weights. The flowchart for the training algorithm of Hebb network is given below

Step 0: First initialize the weights. Basically in this network they may be set to zero, i.e., $w_i = 0$, for $i = 1$ to n where " n " may be the total number of input neurons.

Step 1: Steps 2-4 have to be performed for each input training vector and target output pair, $s: t$.

Step 2: Input units activations are set. Generally, the activation function of input layer is identity function: $x_i = s_i$ for $i = 1$ to n

Step 3: Output units activations are set: $y = t$.

Step 4: Weight adjustments and bias adjustments are performed:

$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$

In step 4, the weight updation formula can be written in vector form as

$$w(new) = w(old) + y$$

Hence, Change in weight is expressed as

$$\Delta w = xy$$

As a result,

$$w(new) = w(old) + \Delta w$$

Hebb rule is used for pattern association, pattern categorization, pattern classification and over a range of other areas.

Flowchart of Training algorithm

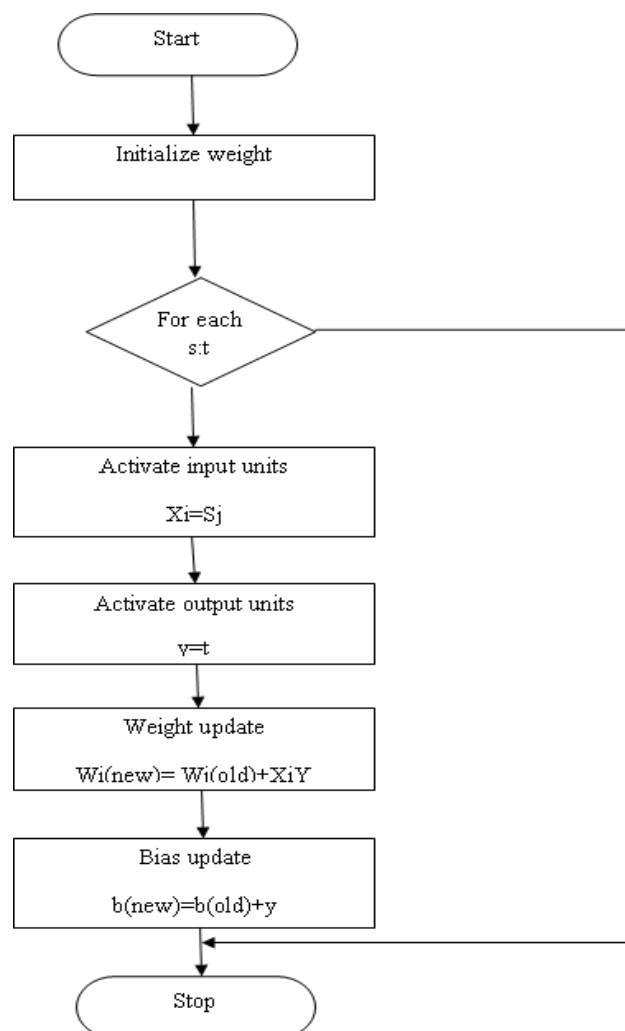


Figure 1.20: Flowchart of Hebb training algorithm

Module – 2

- Perceptron networks
 - Learning rule
 - Training and testing algorithm
- Adaptive Linear Neuron
- Back propagation Network
 - Architecture
 - Training algorithm

2.1 Perceptron networks

2.1.1 Theory

Perceptron networks come under single-layer feed-forward networks and are also called simple perceptrons. Various types of perceptrons were designed by Rosenblatt (1962) and Minsky-Papert (1969, 1988).

The key points to be noted in a perceptron network are:

1. The perceptron network consists of three units, namely, sensory unit (input unit), associator unit (hidden unit), and response unit (output unit).
2. The sensory units are connected to associator units with fixed weights having values 1, 0 or -1, which are assigned at random.
3. The binary activation function is used in sensory unit and associator unit.
4. The response unit has an activation of 1, 0 or -1. The binary step with fixed threshold θ is used as activation for associator. The output signals that are sent from the associator unit to the response unit are only binary.
5. The output of the perceptron network is given by

$$y = f(y_{in})$$

where $f(y_{in})$ is activation function and is defined as

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

6. The perceptron learning rule is used in the weight updation between the associator unit and the response unit. For each training input, the net will calculate the response and it will determine whether or not an error has occurred.
7. The error calculation is based on the comparison of the values of targets with those of the calculated outputs.
8. The weights on the connections from the units that send the nonzero signal will get adjusted suitably.
9. The weights will be adjusted on the basis of the learning rule an error has occurred for a particular training patterns .i.e.,

$$w_i(new) = w_i(old) + \alpha tx_i$$

$$b(new) = b(old) + \alpha t$$

If no error occurs, there is no weight updation and hence the training process may be stopped. In the above equations, the target value " t " is +1 or -1 and α is the learning rate. In general, these learning rules begin with an initial guess at the weight values and then successive adjustments are made on the basis of the evaluation of an objective function. Eventually, the learning rules reach a near optimal or optimal solution in a finite number of steps.

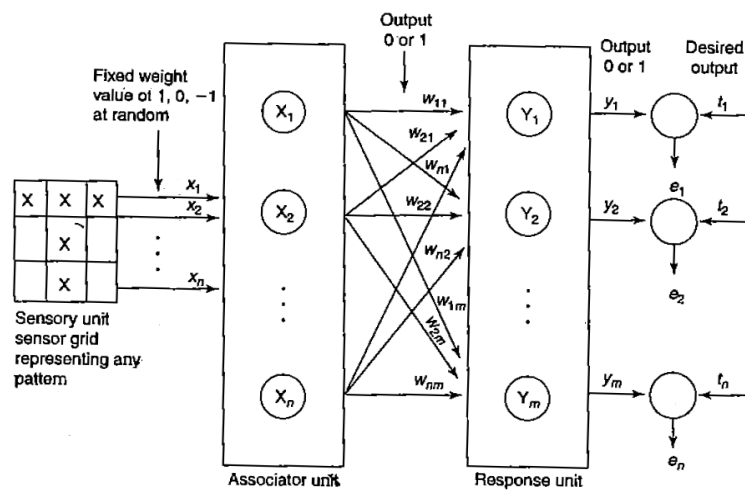


Figure 2.1: Original perceptron network

A Perceptron network with its three units is shown in above figure. The sensory unit can be a two-dimensional matrix of 400 photodetectors upon which a lighted picture with geometric black and white pattern impinges. These detectors provide a binary (0) electrical signal if the input signal is found to exceed a certain value of threshold. Also, these detectors are connected randomly with the associator unit. The associator unit is found to consist of a set of subcircuits called feature predicates. The feature predicates are hardwired to detect the specific feature of a pattern and are equivalent to the feature detectors. For a particular feature, each predicate is examined with a few or all of the responses of the sensory unit. It can be found that the results from the predicate units are also binary (0 or 1). The last unit, i.e. response unit, contains the pattern recognizers or perceptrons. The weights present in the input layers are all fixed, while the weights on the response unit are trainable.

2.1.2 Perceptron Learning Rule

Learning signal is the difference between desired and actual response of a neuron. The perceptron learning rule is explained as follows:

Consider a finite "n" number of input training vectors, with their associated target (desired) values $x(n)$ and $t(n)$, where "n" ranges from 1 to N. The target is either +1 or -1. The output "y" is obtained on the basis of the net input calculated and activation function being applied over the net input.

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

The weight updation in case of perceptron learning is as shown.

If $y \neq t$, then

$$w(\text{new}) = w(\text{old}) + \alpha tx \quad (\alpha - \text{learning rate})$$

else,

$$w(\text{new}) = w(\text{old})$$

The weights can be initialized at any values in this method. The perceptron rule convergence theorem states that " If there is a weight vector W such that $f(x(n)W) = t(n)$, for all n then for any starting vector w_1 , the perceptron learning rule will convergence to a weight vector that gives the correct response for all training patterns, and this learning takes place within a finite number of steps provided that the solution exists".

2.1.3 Architecture

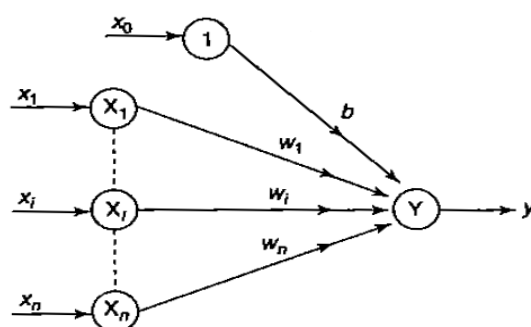


Figure 2.2: Single classification perceptron network

Here only the weights between the associator unit and the output unit can be adjusted, and the weights between the sensory and associator units are fixed.

2.1.4 Flowchart for Training Process

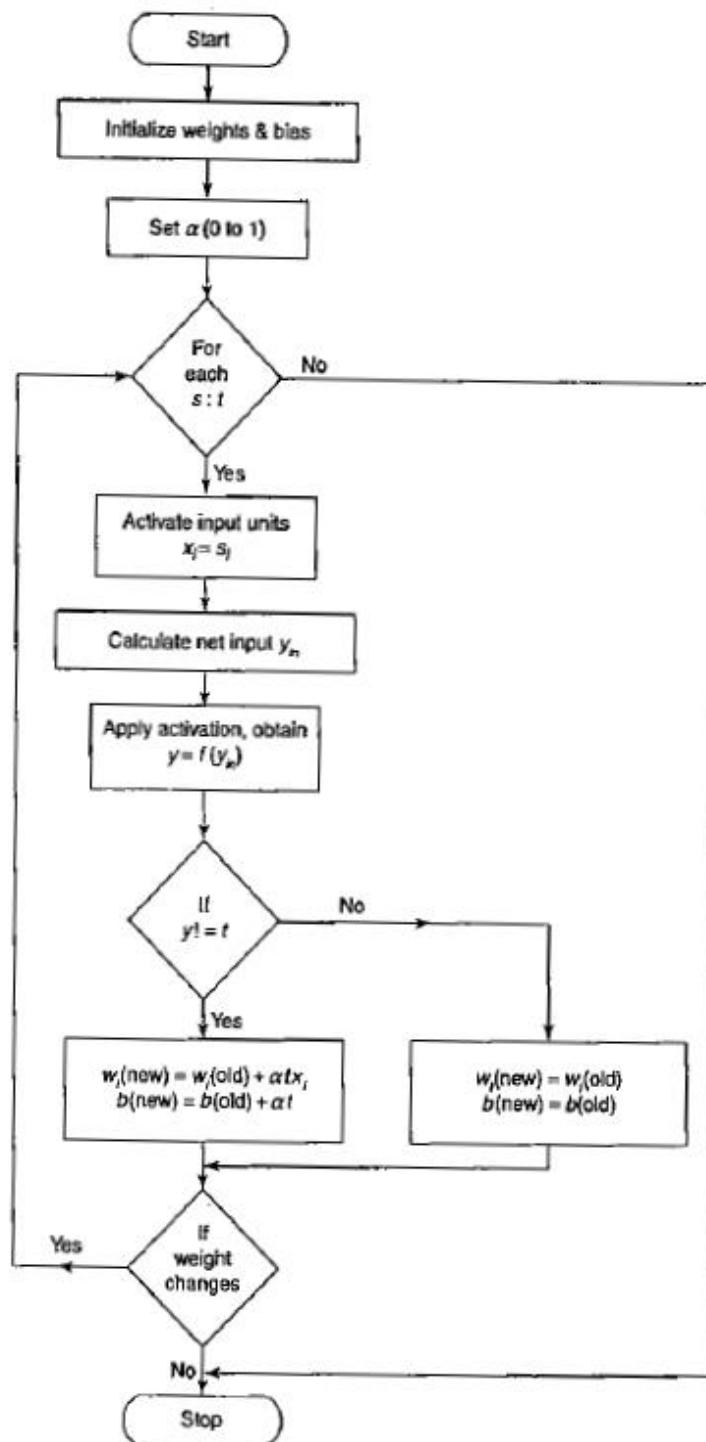


Figure 2.3: Flowchart for perceptron network with single output

2.1.5 Perceptron Training Algorithm for Single Output Classes

Step 0: Initialize the weights and the bias (for easy calculation they can be set to zero). Also initialize the learning rate α ($0 < \alpha \leq 1$). For simplicity α is set to 1.

Step 1: Perform Steps 2-6 until the final stopping condition is false.

Step 2: Perform Steps 3-5 for each training pair indicated by s:t.

Step 3: The input layer containing input units is applied with identity activation functions:

$$x_i = s_i$$

Step 4: Calculate the output of the network. To do so, first obtain the net input:

$$y_{in} = b + \sum_{i=1}^n x_i w_i$$

Where "n" is the number of input neurons in the input layer. Then apply activations over the net input calculated to obtain the output:

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Step 5: Weight and bias adjustment: Compare the value of the actual (calculated) output and desired (target) output.

If $y \neq t$, then

$$w_i(new) = w_i(old) + \alpha t x_i$$

$$b(new) = b(old) + \alpha t$$

else,

$$w_i(new) = w_i(old)$$

$$b(new) = b(old)$$

Step 6: Train the network until there is no weight change. This is the stopping condition for the network. If this condition is not met, then start again from Step 2.

2.1.6 Perceptron Network Testing Algorithm

Step 0: The initial weights to be used here are taken from the training algorithms (the final weights obtained during training).

Step 1: For each input vector X to be classified, perform Steps 2-3.

Step 2: Set activations of the input unit.

Step 3: Obtain the response of output unit.

$$y_{in} = \sum_{i=1}^n x_i w_i$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Thus, the testing algorithm tests the performance of network. In the case of perceptron network, it can be used for linear separability. Here the separating line may be based on the value of threshold that is, the threshold used in the activation function must be a non negative value.

The condition for separating the response from the region of positive to region of zero is

$$w_1 x_1 + w_2 x_2 + b > \theta$$

The condition for separating the response from the region of zero to region of negative is

$$w_1 x_1 + w_2 x_2 + b < -\theta$$

The conditions above are stated for a single layer perceptron network with two input neurons and one output neuron and one bias.

2.2 Adaptive Linear Neuron

2.2.1 Theory

The units with linear activation function are called linear units. A network with a single linear unit is called an Adaline (adaptive linear neuron). That is, in an Adaline, the input-output relationship is linear. Adaline uses bipolar activation for its input signals and its target output. The weights between the input and the output are adjustable. The bias in

Adaline acts like an adjustable weight, whose connection is from a unit with activations being always 1. Adaline is a net which has only one output unit. The Adaline network may be trained using delta rule. The delta rule may also be called as least mean square (LMS) rule or Widrow-Hoff rule. This learning rule is found to minimize the mean squared error between the activation and the target value.

2.2.2 Delta Rule for Single Output Unit

The perceptron learning rule originates from the Hebbian assumption while the delta rule is derived from the gradient descent method (it can be generalized to more than one layer). Also, the perceptron learning rule stops after a finite number of leaning steps, but the gradient-descent approach continues forever, converging only asymptotically to the solution. The delta rule updates the weights between the connections so as to minimize the difference between the net input to the output unit and the target value. The major aim is to minimize the error over all training patterns. This is done by reducing the error for each pattern, one at a time.

The delta rule for adjusting the weight of i th pattern ($i = 1$ to n) is

$$\Delta w_i = \alpha(t - y_{in})x_i$$

Where Δw_i is the weight change; α the learning rate; x the vector of activation of input unit; y_{in} the net input to output unit, i.e., $y_{in} = \sum_{i=1}^n x_i w_i$; t the target output. The delta rule in case of several output units for adjusting the weight from i th input unit to the j th output unit (for each pattern) is

$$\Delta w_{ij} = \alpha(t_j - y_{in_j})x_i$$

2.2.3 Architecture

Adaline is a single unit neuron, which receives input from several units and also from one unit called bias. The basic Adaline model consists of trainable weights. Inputs are either of the two values (+ 1 or -1) and the weights have signs (positive or negative). Initially, random weights are assigned. The net input calculated is applied to a quantizer transfer function (possibly activation function) that restores the output to + 1 or -1. The Adaline

model compares the actual output with the target output and on the basis of the training algorithm, the weights are adjusted.

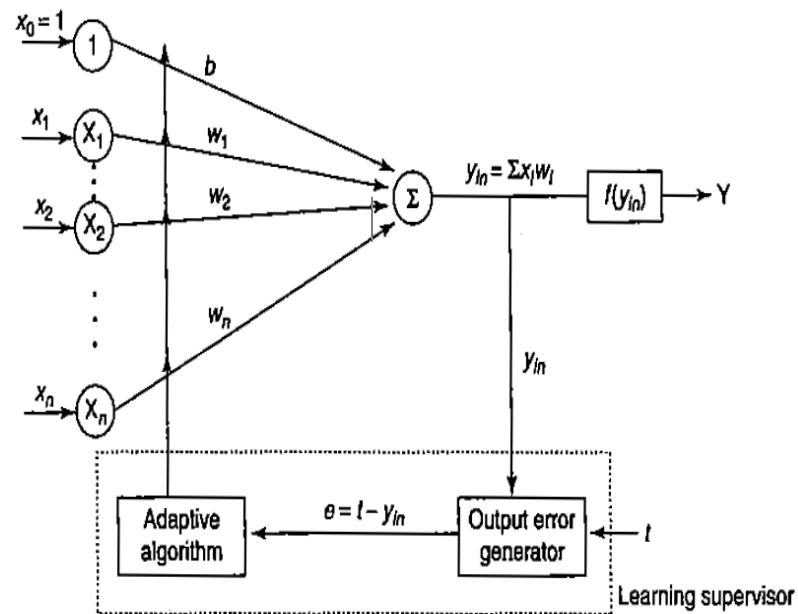


Figure 2.4: Adaline model

2.2.4 Flowchart for Training Process

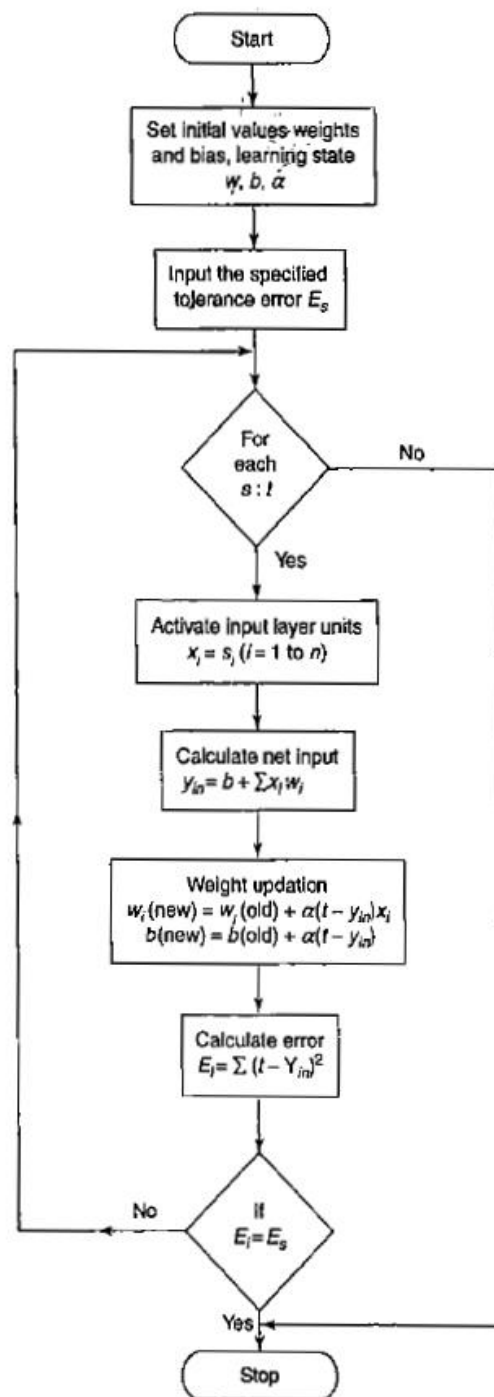


Figure 2.5: Flowchart for Adaline training process

2.2.5 Training Algorithm

Step 0: Weights and bias are set to some random values but not zero. Set the learning rate parameter α .

Step 1: Perform Steps 2-6 when stopping condition is false.

Step 2: Perform Steps 3-5 for each bipolar training pair $s:t$.

Step 3: Set activations for input units $i = 1$ to n .

$$x_i = s_i$$

Step 4: Calculate the net input to the output unit.

$$y_{in} = b + \sum_{i=1}^n x_i w_i$$

Step 5: Update the weights and bias for $i = 1$ to n

$$w_i(\text{new}) = w_i(\text{old}) + \alpha(t - y_{in})x_i$$

$$b(\text{new}) = b(\text{old}) + \alpha(t - y_{in})$$

Step 6: If the highest weight change that occurred during training is smaller than a specified tolerance then stop the training process, else continue. This is the rest for stopping condition of a network.

2.2.6 Testing Algorithm

Step 0: Initialize the weights. (The weights are obtained from the training algorithm.)

Step 1: Perform Steps 2-4 for each bipolar input vector x .

Step 2: Set the activations of the input units to x .

Step 3: Calculate the net input to the output unit:

$$y_{in} = b + \sum x_i w_i$$

Step 4: Apply the activation function over the net input calculated:

$$y = \begin{cases} 1 & \text{if } y_{in} \geq 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

2.3 Back propagation Network

2.3.1 Theory

The back propagation learning algorithm is one of the most important developments in neural networks (Bryson and Ho, 1969; Werbos, 1974; Lecun, 1985; Parker, 1985;

Rumelhart, 1986). This learning algorithm is applied to multilayer feed-forward networks consisting of processing elements with continuous differentiable activation functions. The networks associated with back-propagation learning algorithm are also called back-propagation networks. (BPNs). For a given set of training input-output pair, this algorithm provides a procedure for changing the weights in a BPN to classify the given input patterns correctly. The basic concept for this weight update algorithm is simply the gradient descent method. This is a methods were error is propagated back to the hidden unit. Back propagation network is a training algorithm.

The training of the BPN is done in three stages - the feed-forward of the input training pattern, the calculation and back-propagation of the error, and updation of weights. The testing of the BPN involves the computation of feed-forward phase only. There can be more than one hidden layer (more beneficial) but one hidden layer is sufficient. Even though the training is very slow, once the network is trained it can produce its outputs very rapidly.

2.3.2 Architecture

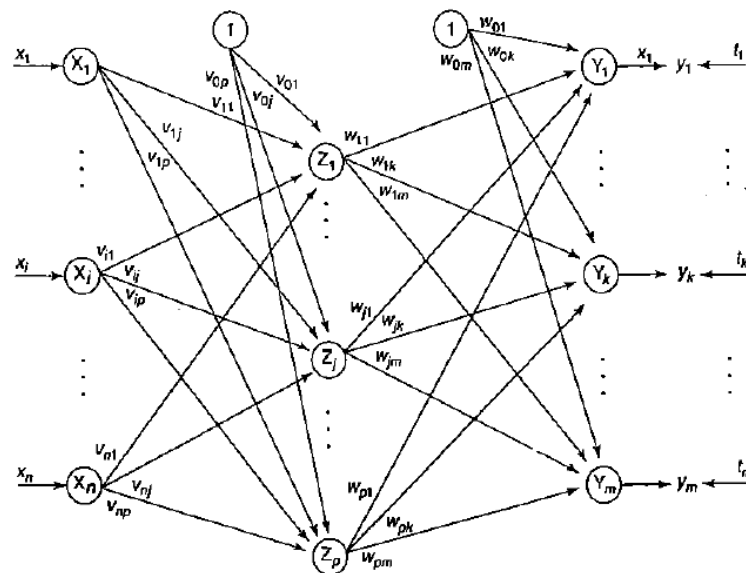


Figure 2.6: Architecture of a back propagation network

A back-propagation neural network is a multilayer, feed-forward neural network consisting of an input layer, a hidden layer and an output layer. The neurons present in the hidden and output layers have biases, which are the connections from the units whose activation is always 1. The bias terms also acts as weights. During the back propagation phase of learning, signals are sent in the reverse direction. The inputs sent to the BPN and the output

obtained from the net could be either binary (0, 1) or bipolar (-1, +1). The activation function could be any function which increases monotonically and is also differentiable.

2.3.3 Flowchart

The terminologies used in the flowchart and in the training algorithm are as follows:

x = input training vector ($x_1, \dots, x_i, \dots, x_n$)

t = target output vector ($t_1, \dots, t_k, \dots, t_m$)

α = learning rate parameter

x_i = input unit i . (Since the input layer uses identity activation function, the input and output signals here are same.)

v_{0j} = bias on j th hidden unit

w_{0k} = bias on k th output unit

z_j = hidden unit j . The net input to z_j is

$$z_{inj} = v_{0j} + \sum_{i=1}^n x_i v_{ij}$$

and the output is

$$z_j = f(z_{inj})$$

y_k = output unit k . The net input to y_k is

$$y_{ink} = w_{0k} + \sum_{j=1}^p z_j w_{jk}$$

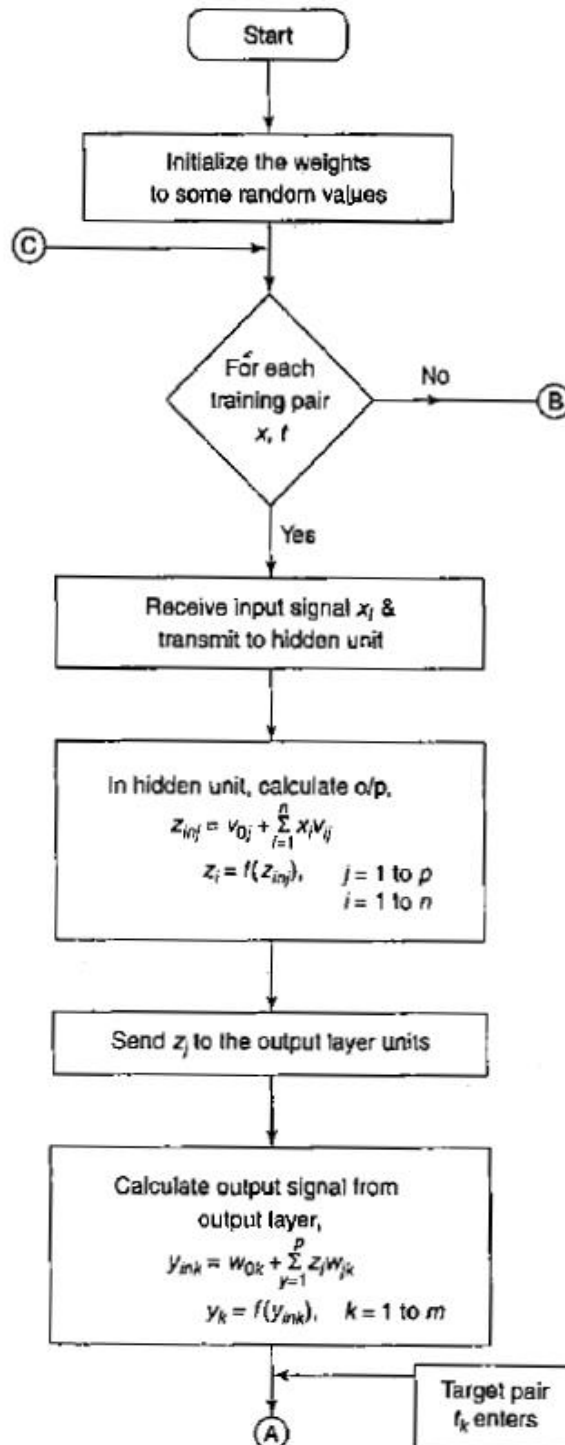
and the output is

$$y_k = f(y_{ink})$$

δ_k = error correction weight adjustment for w_{jk} that is due to an error in unit y_k , which is back-propagated to the hidden units that feed into unit y_k

δ_j = error correction weight adjustment for v_{ij} that is due to the back-propagation of error to the hidden unit is z_j .

The commonly used activation functions are binary, sigmoidal and bipolar sigmoidal activation functions. These functions are used in the BPN because of the following characteristics: (i) Continuity (ii) Differentiability (iii) Non decreasing monotonic.



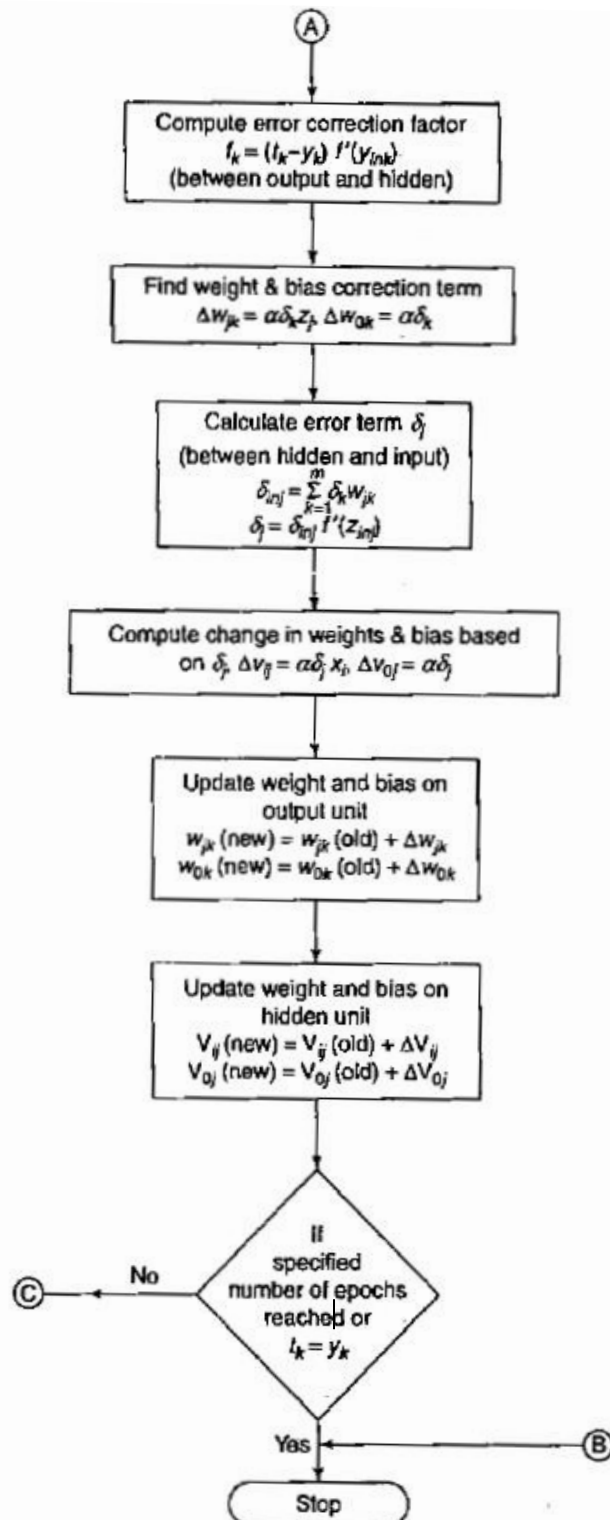


Figure 2.7: Flowchart for back - propagation network training

2.3.4 Training Algorithm

Step 0: Initialize weights and learning rate (take some small random values).

Step 1: Perform Steps 2-9 when stopping condition is false.

Step 2: Perform Steps 3-8 for each training pair.

Feedforward Phase I

Step 3: Each input unit receives input signal x_i and sends it to the hidden unit ($i = 1$ to n).

Step 4: Each hidden unit z_j ($j = 1$ to p) sums its weighted input signals to calculate net input:

$$z_{inj} = v_{0j} + \sum_{i=1}^n x_i v_{ij}$$

Calculate output of the hidden unit by applying its activation functions over z_{inj} (binary or bipolar sigmoidal activation function):

$$z_j = f(z_{inj})$$

and send the output signal from the hidden unit to the input of output layer units.

Step 5: For each output unit y_k ($k = 1$ to m), calculate the net input:

$$y_{ink} = w_{0k} + \sum_{j=1}^p z_j w_{jk}$$

and apply the activation function to compute output signal

$$y_k = f(y_{ink})$$

Back-propagation of error (Phase II)

Step 6: Each output unit y_k ($k=1$ to m) receives a target pattern corresponding to the input training pattern and computes the error correction term:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

The derivative $f'(y_{ink})$ can be calculated as in activation function section. On the basis of the calculated error correction term, update the change in weights and bias:

$$\Delta w_{jk} = \alpha \delta_k z_j; \quad \Delta w_{0k} = \alpha \delta_k$$

Also, send δ_k to the hidden layer backwards.

Step 7: Each hidden unit ($z_j = 1$ to p) sums its delta inputs from the output units:

$$\delta_{inj} = \sum_{k=1}^m \delta_k w_{jk}$$

The term δ_{inj} gets multiplied with the derivative of $f(z_{inj})$ to calculate the error term:

$$\delta_j = \delta_{inj} f'(z_{inj})$$

The derivative $f'(z_{inj})$ can be calculated as activation function section depending on whether binary or bipolar sigmoidal function is used. On the basis of the calculated δ_j , update the change in weights and bias:

$$\Delta v_{ij} = \alpha \delta_j x_i; \quad \Delta v_{0j} = \alpha \delta_j$$

Weight and bias updation (Phase III):

Step 8: Each output unit ($y_k, k = 1$ to m) updates the bias and weights:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

$$w_{0k}(new) = w_{0k}(old) + \Delta w_{0k}$$

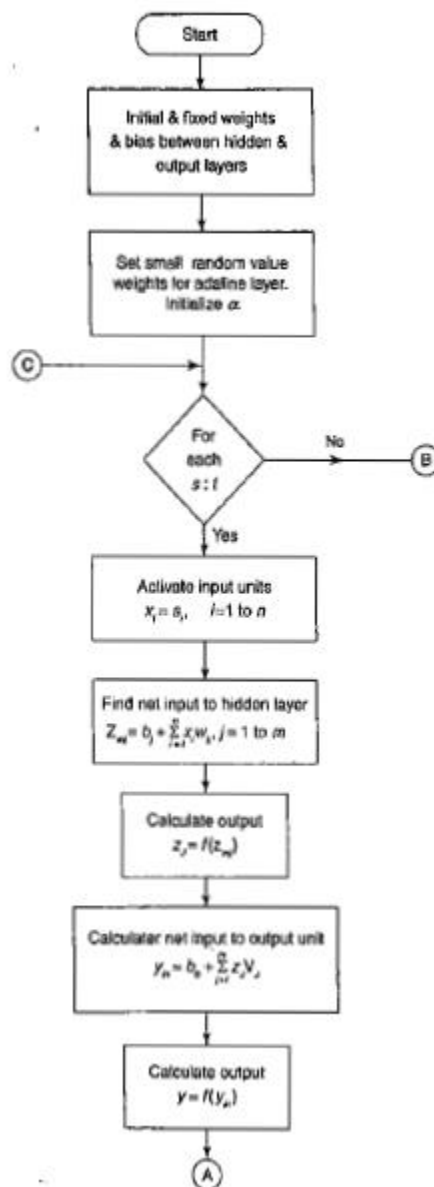
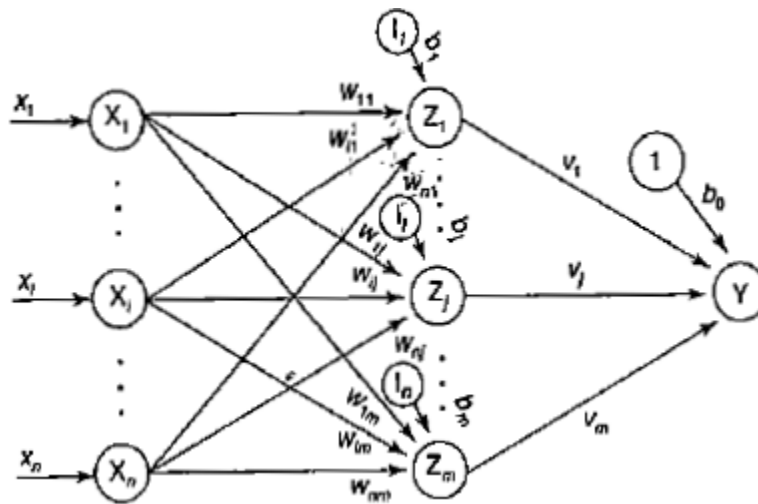
Each hidden unit ($z_j, j = 1$ to p) updates its bias and weights:

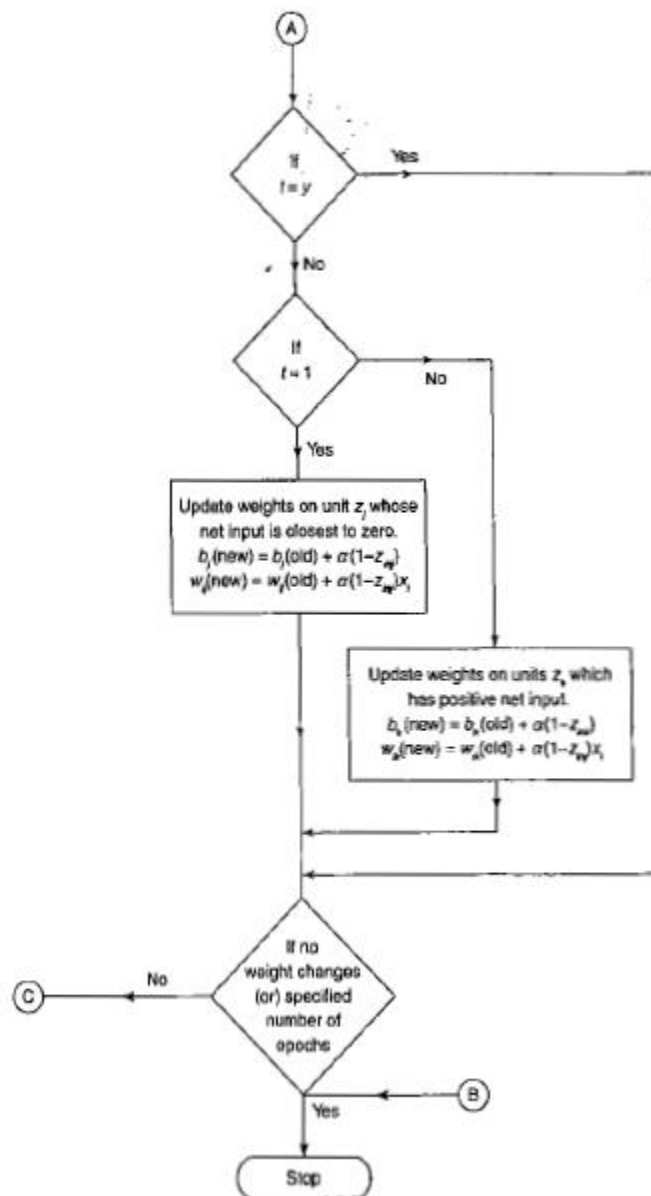
$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}$$

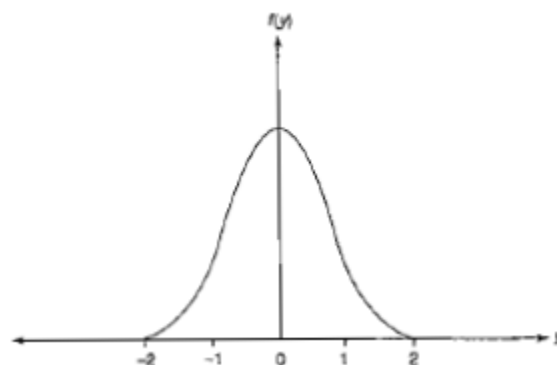
Step 9: Check for the stopping condition. The stopping condition may be certain number of epochs reached or when the actual output equals the target output.

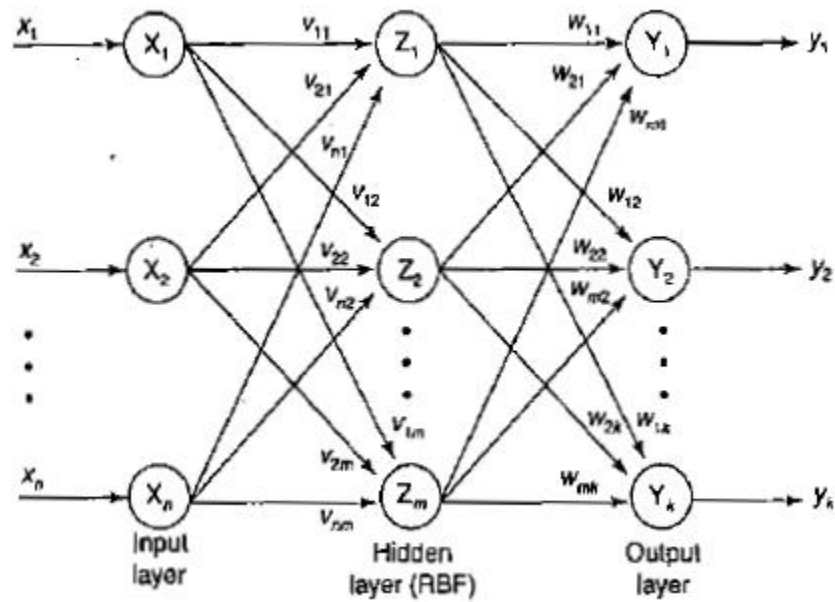
2.4 Madaline



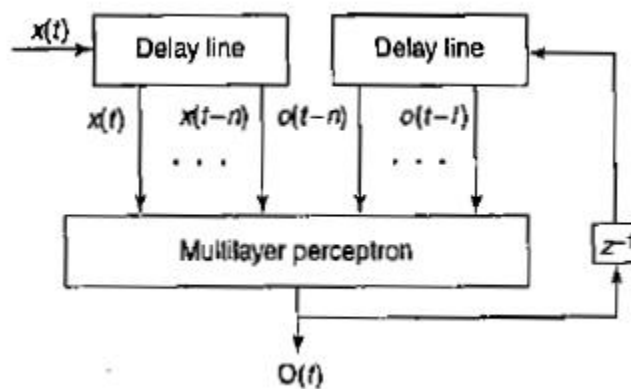
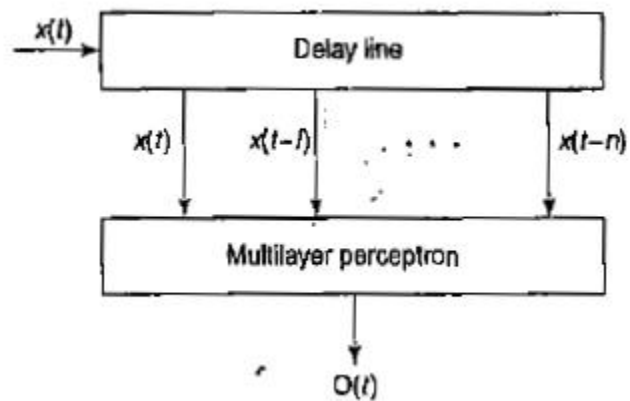


2.5 Radial Basis Function Network

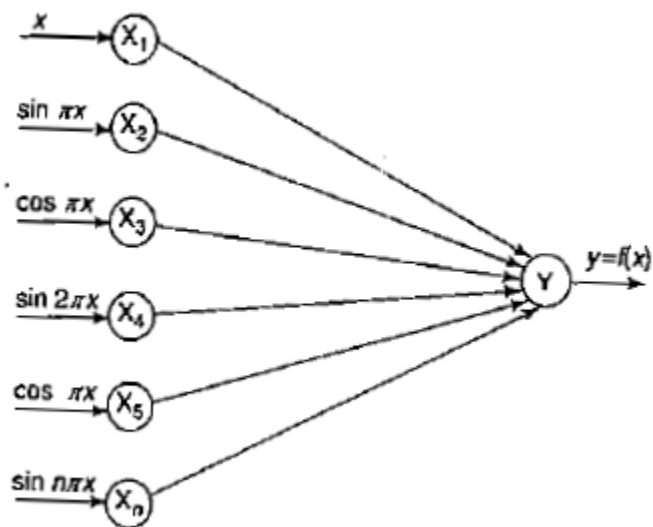




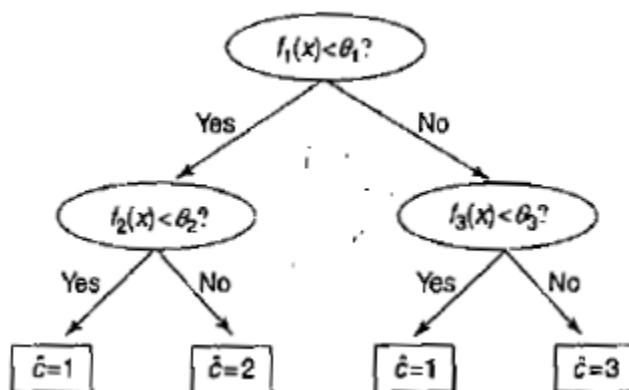
2.6 Time Delay Neural Network



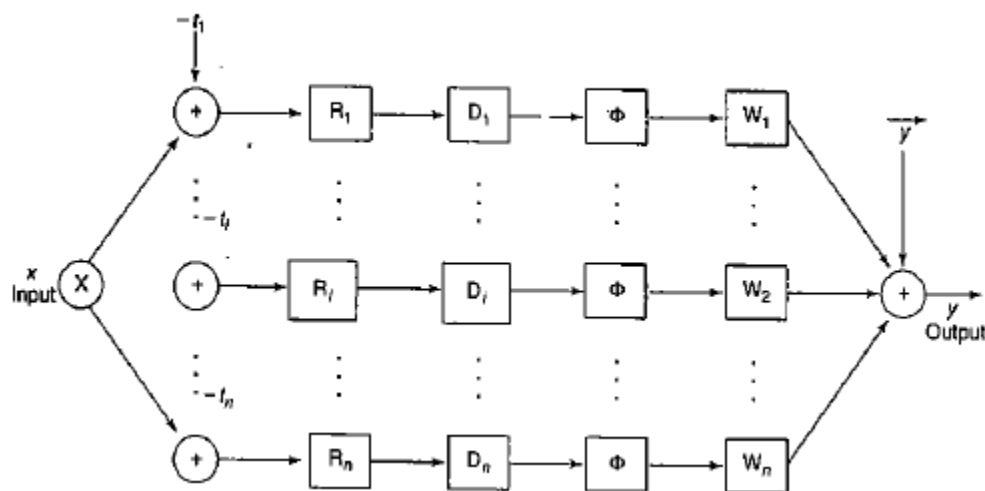
2.7 Functional Link Networks



2.8 Tree Neural Networks



2.9 Wavelet Neural Networks



2.10 Advantages of Neural Networks

1. Mimicks human control logic.
2. Uses imprecise language.
3. Inherently robust.
4. Fails safely.
5. Modified and tweaked easily.

2.11 Disadvantages of Neural Networks

1. Operator's experience required.
2. System complexity.

2.12 Applications of Neural Networks

1. Automobile and other vehicle subsystems, such as automatic transmissions, ABS and cruise control (e.g. Tokyo monorail).
2. Air conditioners.
3. Auto focus on cameras.
4. Digital image processing, such as edge detection.
5. Rice cookers.
6. Dishwashers.
7. Elevators.

Module – 3

- Fuzzy logic
- Fuzzy sets
 - Properties
 - Operations on fuzzy sets
- Fuzzy relations
 - Operations on fuzzy relations

2.13 Fuzzy logic

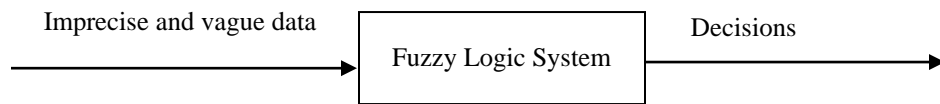
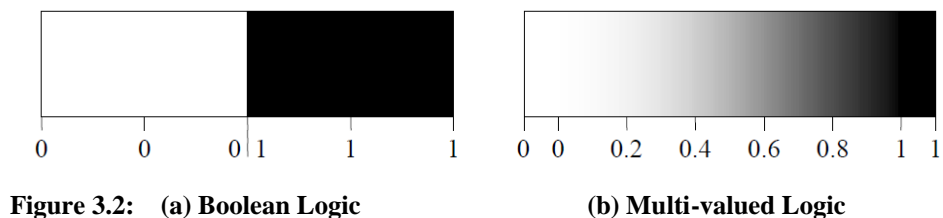


Figure 3.1: A fuzzy logic system accepting imprecise data and providing a decision

In 1965 Lotfi Zadeh, published his famous paper “Fuzzy sets”. This new logic for representing and manipulating fuzzy terms was called fuzzy logic, and Zadeh became the Master/Father of fuzzy logic.

Fuzzy logic is the logic underlying approximate, rather than exact, modes of reasoning. It operates on the concept of membership. The membership was extended to possess various "degrees of membership" on the real continuous interval $[0, 1]$.

In fuzzy systems, values are indicated by a number (called a truth value) ranging from 0 to 1, where 0.0 represents absolute falseness and 1.0 represents absolute truth.



2.14 Classical sets(Crisp sets)

A classical set is a collection of objects with certain characteristics. For example, the user may define a classical set of negative integers, a set of persons with height less than 6 feet, and a set of students with passing grades. Each individual entity in a set is called a member or an element of the set.

There are several ways for defining a set. A set may be defined using one of the following:

1. The list of all the members of a set may be given.

Example $A = \{2, 4, 6, 8, 10\}$ (**Roaster form**)

2. The properties of the set elements may be specified.

Example $A = \{x | x \text{ is prime number} < 20\}$ (**Set builder form**)

3. The formula for the definition of a set may be mentioned. Example

$$A = \left\{ x_i = \frac{x_i + 1}{5}, i = 1 \text{ to } 10, \quad \text{where } x_i = 1 \right\}$$

4. The set may be defined on the basis of the results of a logical operation.

Example $A = \{x | x \text{ is an element belonging to } P \text{ AND } Q\}$

5. There exists a membership function, which may also be used to define a set. The membership is denoted by the letter χ and the membership function for a set A is given by (for all values of x).

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

The set with no elements is defined as an empty set or null set. It is denoted by symbol \emptyset .

The set which consist of all possible subset of a given set A is called power set

$$P(A) = \{x | x \subseteq A\}$$

2.14.1 Properties

1. Commutativity

$$A \cup B = B \cup A; \quad A \cap B = B \cap A$$

2. Associativity

$$A \cup (B \cap C) = (A \cup B) \cap C; \quad A \cap (B \cup C) = (A \cap B) \cup C$$

3. Distributivity

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

4. Idempotency

$$A \cup A = A; \quad A \cap A = A$$

5. Transitivity

$$\text{If } A \subseteq B \subseteq C, \text{ then } A \subseteq C$$

6. Identity

$$A \cup \emptyset = A, \quad A \cap \emptyset = \emptyset$$

$$A \cup X = X, \quad A \cap X = A$$

7. Involution (double negation)

$$\bar{\bar{A}} = A$$

8. Law of excluded middle

$$A \cup \bar{A} = X$$

9. Law of contradiction

$$A \cap \bar{A} = \emptyset$$

10. DeMorgans law

$$\overline{A \cap B} = \bar{A} \cup \bar{B}; \quad \overline{A \cup B} = \bar{A} \cap \bar{B};$$

2.14.2 Operations on Classical sets**1. Union**

The union between two sets gives all those elements in the universe that belong to either set A or set B or both sets A and B. The union operation can be termed as a logical OR operation. The union of two sets A and B is given as

$$A \cup B = \{x | x \in A \text{ or } x \in b\}$$

The union of sets A and B is illustrated by the Venn diagram shown below

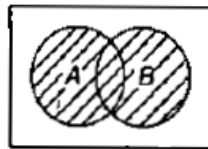


Figure 3.3: Union of two sets

2. Intersection

The intersection between two sets represents all those elements in the universe that simultaneously belong to both the sets. The intersection operation can be termed as a logical AND operation. The intersection of sets A and B is given by

$$A \cap B = \{x | x \in A \text{ and } x \in b\}$$

The intersection of sets A and B is illustrated by the Venn diagram shown below

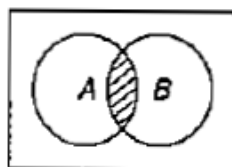


Figure 3.4: Intersection of two sets

3. Complement

The complement of set A is defined as the collection of all elements in universe X that do not reside in set A, i.e., the entities that do not belong to A. It is denoted by \bar{A} and is defined as

$$\bar{A} = \{x | x \notin A, x \in X\}$$

where X is the universal set and A is a given set formed from universe X. The complement operation of set A is shown below

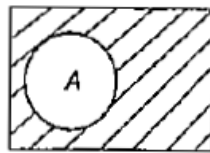


Figure 3.5: Complement of set A

4. Difference (Subtraction)

The difference of set A with respect to set B is the collection of all elements in the universe that belong to A but do not belong to B, i.e., the difference set consists of all elements that belong to A but do not belong to B. It is denoted by $A \setminus B$ or $A - B$ and is given by

$$A \setminus B \text{ or } (A - B) = \{x | x \in A \text{ and } x \notin B\} = A - (A \cap B)$$

The vice versa of it also can be performed

$$B \setminus A \text{ or } (B - A) = B - (B \cap A) = \{x | x \in B \text{ and } x \notin A\}$$

The above operations are shown below

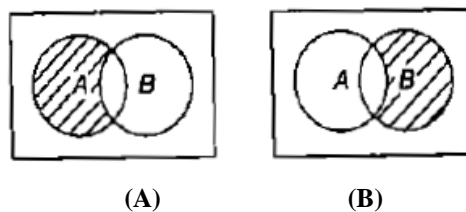


Figure 3.6: (A) Difference $A \setminus B$ or $(A - B)$; (B) Difference $B \setminus A$ or $(B - A)$

2.14.3 Function Mapping of Classical Sets

Mapping is a rule of correspondence between set-theoretic forms and function theoretic forms. A classical set is represented by its characteristic function $\chi_A(x)$ where x is the element in the universe.

Now consider X and Y as two different universes of discourse. If an element x contained in X corresponds to an element y in Y , it is called mapping from X to Y , i.e., $f: X \rightarrow Y$. On the basis of this mapping, the characteristics function is defined as

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

where χ_A is the membership in set A for element x in the universe. The membership concept represents mapping from an element x in universe X to one of the two elements in universe Y (either to element 0 or 1).

Let A and B be two sets in universe X . The function-theoretic forms of operations performed between these two sets are given as follows:

1. Union

$$\chi_{A \cup B}(x) = \chi_A(x) \vee \chi_B(x) = \max[\chi_A(x), \chi_B(x)]$$

where \vee indicates max operator.

2. Intersection

$$\chi_{A \cap B}(x) = \chi_A(x) \wedge \chi_B(x) = \min[\chi_A(x), \chi_B(x)]$$

where \wedge indicates min operator.

3. Complement

$$\chi_{\bar{A}}(x) = 1 - \chi_A(x)$$

2.15 Fuzzy sets

A fuzzy set \tilde{A} in the universe of discourse U can be defined as

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

where $\mu_{\tilde{A}}(x)$ is the degree of membership of x in \tilde{A} and it indicates the degree that x belongs to \tilde{A} . In the fuzzy theory, fuzzy set A of universe X is defined by function $\mu_{\tilde{A}}(x)$ called the membership function of set A .

$$\mu_{\tilde{A}}(x): X \rightarrow [0, 1], \quad \text{where} \quad \begin{aligned} \mu_{\tilde{A}}(x) &= 1 && \text{if } x \text{ is totally in } A; \\ \mu_{\tilde{A}}(x) &= 0 && \text{if } x \text{ is not in } A; \\ 0 < \mu_{\tilde{A}}(x) < 1 && \text{if } x \text{ is partly in } A. \end{aligned}$$

This set allows a continuum of possible choices. For any element x of universe X , membership function $\mu_A(x)$ equals the degree to which x is an element of set A . This degree, a value between 0 and 1, represents the degree of membership, also called membership value, of element x in set A .

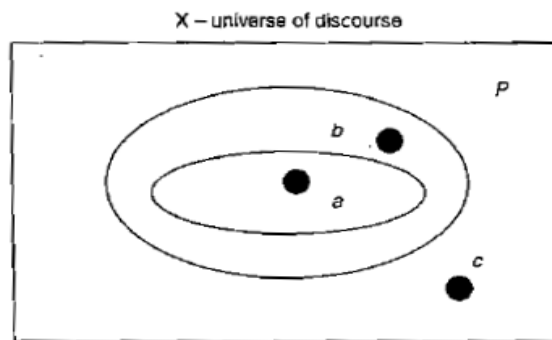


Figure 3.7: Boundary region of a fuzzy set

From figure 3.7 it can be noted that "a" is clearly a member of fuzzy set P , "c" is clearly not a member of fuzzy set P and the membership of "b" is found to be vague. Hence "a" can take membership value 1, "c" can take membership value 0 and "b" can take membership value between 0 and 1 [0 to 1], say 0.4, 0.7, etc. This is said to be a partial membership of fuzzy set P .

There are other ways of representation of fuzzy sets; all representations allow partial membership to be expressed. When the universe of discourse U is discrete and finite, fuzzy set \tilde{A} is given as follows:

$$\tilde{A} = \left\{ \frac{\mu_{\tilde{A}}(x_1)}{x_1} + \frac{\mu_{\tilde{A}}(x_2)}{x_2} + \frac{\mu_{\tilde{A}}(x_3)}{x_3} + \dots \right\} = \left\{ \sum_{i=1}^n \frac{\mu_{\tilde{A}}(x_i)}{x_i} \right\}$$

2.15.1 Properties

Fuzzy sets follow the same properties as crisp sets except for the law of excluded middle and law of contradiction.

That is, for fuzzy set \tilde{A}

$$\tilde{A} \cup \bar{\tilde{A}} = U; \quad \tilde{A} \cap \bar{\tilde{A}} = \emptyset$$

1. Commutativity

$$\tilde{A} \cup \tilde{B} = \tilde{B} \cup \tilde{A}; \quad \tilde{A} \cap \tilde{B} = \tilde{B} \cap \tilde{A}$$

2. Associativity

$$\begin{aligned} \tilde{A} \cup (\tilde{B} \cup \tilde{C}) &= (\tilde{A} \cup \tilde{B}) \cup \tilde{C} \\ \tilde{A} \cap (\tilde{B} \cap \tilde{C}) &= (\tilde{A} \cap \tilde{B}) \cap \tilde{C} \end{aligned}$$

3. Distributivity

$$\begin{aligned} \tilde{A} \cup (\tilde{B} \cap \tilde{C}) &= (\tilde{A} \cup \tilde{B}) \cap (\tilde{A} \cup \tilde{C}) \\ \tilde{A} \cap (\tilde{B} \cup \tilde{C}) &= (\tilde{A} \cap \tilde{B}) \cup (\tilde{A} \cap \tilde{C}) \end{aligned}$$

4. Idempotency

$$\tilde{A} \cup \tilde{A} = \tilde{A}; \quad \tilde{A} \cap \tilde{A} = \tilde{A}$$

5. Transitivity

$$\text{If } \tilde{A} \subseteq \tilde{B} \subseteq \tilde{C}, \text{ then } \tilde{A} \subseteq \tilde{C}$$

6. Identity

$$\begin{aligned} \tilde{A} \cup \emptyset &= \tilde{A} \text{ and } \tilde{A} \cup U = U \\ \tilde{A} \cap \emptyset &= \emptyset \text{ and } \tilde{A} \cap U = \tilde{A} \end{aligned}$$

7. Involution (double negation)

$$\bar{\bar{\tilde{A}}} = \tilde{A}$$

8. DeMorgans law

$$\overline{\tilde{A} \cap \tilde{B}} = \bar{\tilde{A}} \cup \bar{\tilde{B}}; \quad \overline{\tilde{A} \cup \tilde{B}} = \bar{\tilde{A}} \cap \bar{\tilde{B}};$$

2.15.2 Operations on fuzzy sets

1. Union

The union of fuzzy sets \tilde{A} and \tilde{B} , denoted by $\tilde{A} \cup \tilde{B}$ is defined as

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = \max [\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)] = \mu_{\tilde{A}}(x) \vee \mu_{\tilde{B}}(x) \text{ for all } x \in U$$

where \vee indicates max operator. The Venn diagram for union operation of fuzzy sets \tilde{A} and \tilde{B} is shown below figure.

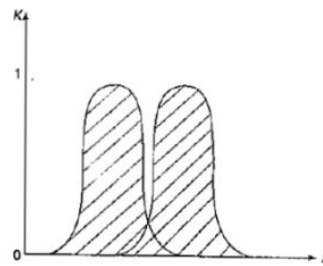


Figure 3.8: Union of fuzzy sets \tilde{A} and \tilde{B}

2. Intersection

The intersection of fuzzy sets \tilde{A} and \tilde{B} , denoted by $\tilde{A} \cap \tilde{B}$, is defined as

$$\mu_{\tilde{A} \cap \tilde{B}}(x) = \min [\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)] = \mu_{\tilde{A}}(x) \wedge \mu_{\tilde{B}}(x) \text{ for all } x \in U$$

where \wedge indicates min operator. The Venn diagram for intersection operation of fuzzy sets \tilde{A} and \tilde{B} is shown below figure.

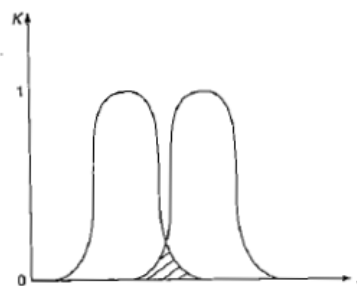


Figure 3.9: Intersection of fuzzy sets \tilde{A} and \tilde{B}

3. Complement

When $\mu_{\tilde{A}}(x) \in [0,1]$, the complement of \tilde{A} , denoted as $\bar{\tilde{A}}$ is defined by,

$$\mu_{\bar{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x) \text{ for all } x \in U$$

The Venn diagram for complement operation of fuzzy set \tilde{A} is shown below figure.

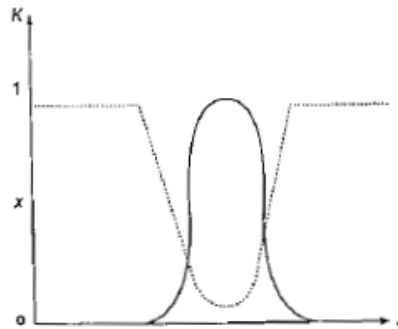


Figure 3.10: Complement of fuzzy set \tilde{A}

4. More Operations on Fuzzy Sets

a. Algebraic sum

The algebraic sum $(\tilde{A} + \tilde{B})$ of fuzzy sets, fuzzy sets \tilde{A} and \tilde{B} is defined as

$$\mu_{\tilde{A} + \tilde{B}}(x) = \mu_{\tilde{A}}(x) + \mu_{\tilde{B}}(x) - \mu_{\tilde{A}}(x) \cdot \mu_{\tilde{B}}(x)$$

b. Algebraic product

The algebraic product $(\tilde{A} \cdot \tilde{B})$ of fuzzy sets, fuzzy sets \tilde{A} and \tilde{B} is defined as

$$\mu_{\tilde{A} \cdot \tilde{B}}(x) = \mu_{\tilde{A}}(x) \cdot \mu_{\tilde{B}}(x)$$

c. Bounded sum

The bounded sum $(\tilde{A} \oplus \tilde{B})$ of fuzzy sets, fuzzy sets \tilde{A} and \tilde{B} is defined as

$$\mu_{\tilde{A} \oplus \tilde{B}}(x) = \min \{1, \mu_{\tilde{A}}(x) + \mu_{\tilde{B}}(x)\}$$

d. Bounded difference

The bounded difference $(\tilde{A} \odot \tilde{B})$ of fuzzy sets, fuzzy sets \tilde{A} and \tilde{B} is defined as

$$\mu_{\tilde{A} \odot \tilde{B}}(x) = \max \{0, \mu_{\tilde{A}}(x) - \mu_{\tilde{B}}(x)\}$$

2.16 Classical relations

A classical binary relation represents the presence or absence of a connection or interaction between the elements of two sets.

- **Cartesian Product of Relation**

An ordered r-tuple is an ordered sequence of r-elements expressed in the form $(a_1, a_2, a_3, \dots, a_r)$. An unordered tuple is a collection of r-elements without any restrictions in order. For $r = 2$, the r-tuple is called an ordered pair. For crisp sets A_1, A_2, \dots, A_r , the set of all r-tuples $(a_1, a_2, a_3, \dots, a_r)$, where $a_1 \in A_1, a_2 \in A_2, \dots, a_r \in A_r$ is called the Cartesian product of A_1, A_2, \dots, A_r and is denoted by $A_1 \times A_2 \times \dots \times A_r$.

Consider two universes X and Y; their Cartesian product $X \times Y$ is given by

$$X \times Y = \{(x, y) | x \in X, y \in Y\}$$

Here the Cartesian product forms an ordered pair of every $x \in X$ with every $y \in Y$. Every element in X is completely related to every element in Y. The characteristic function, denoted by χ , gives the strength of the relationship between ordered pair of elements in each universe. If it takes unity as its value, then complete relationship is found; if the value is zero, then there is no relationship, i.e.,

$$\chi_{X \times Y}(x, y) = \begin{cases} 1, & (x, y) \in X \times Y \\ 0, & (x, y) \notin X \times Y \end{cases}$$

When the universes or sets are finite, then the relation is represented by a matrix called relation matrix. An r-dimensional relation matrix represents an r-ary relation. Thus, binary relations are represented by two-dimensional matrices.

Consider the elements defined in the universes X and Y as follows:

$$X = \{2, 4, 6\} \quad Y = \{p, q, r\}$$

The Cartesian product of these two sets leads to

$$X \times Y = \{(p, 2), (p, 4), (p, 6), (q, 2), (q, 4), (q, 6), (r, 2), (r, 4), (r, 6)\}$$

From this set one may select a subset such that

$$R = \{(p, 2), (q, 4), (r, 4), (r, 6)\}$$

Subset R can be represented using a coordinate diagram as shown in below figure

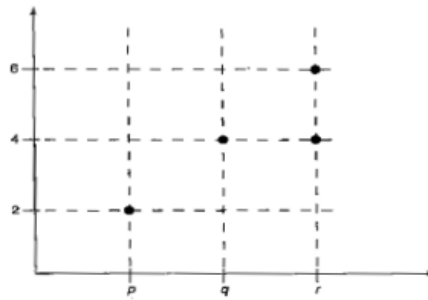


Figure 3.11: Coordinate diagram of a relation

The relation could equivalently be represented using a matrix as follows

R	P	Q	R
2	1	0	0
4	0	1	1
6	0	0	1

The relation between sets X and Y may also be expressed by mapping representations as shown in below figure.

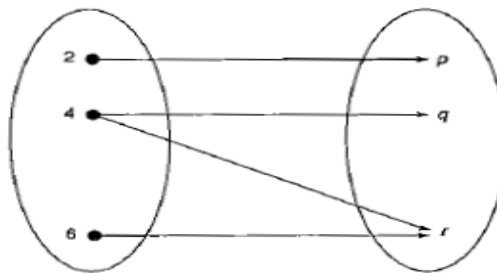


Figure 3.12: Mapping representation of a relation

A binary relation in which each element from set X is not mapped to more than one element in second set Y is called a function and is expressed as

$$R: X \rightarrow Y$$

The characteristic function is used to assign values of relationship in the mapping of the Cartesian space $X \times Y$ to the binary values $(0, 1)$ and is given by

$$\chi_R(x, y) = \begin{cases} 1, & (x, y) \in R \\ 0, & (x, y) \notin R \end{cases}$$

The figure 3.12 (A) and (B) show the illustration of $R: X \rightarrow Y$

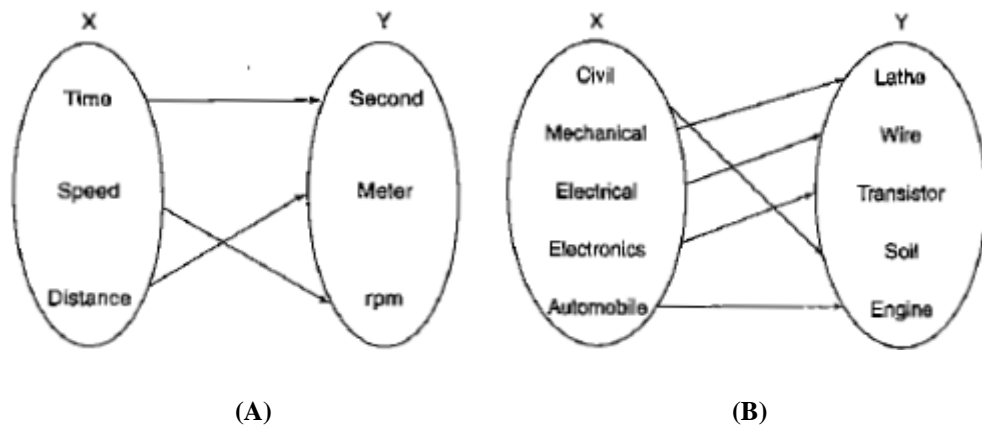


Figure 3.13: Illustration of $R: X \rightarrow Y$

The constrained Cartesian product for sets when $r = 2$ (i.e., $A \times A = A^2$) is called identity relation, and the unconstrained Cartesian product for sets when $r = 2$ is called universal relation.

Consider set $A = \{2, 4, 6\}$.

Then universal relation (U_A) and identity relation (I_A) are given as follows:

$$U_A = \{(2,2), (2,4), (2,6), (4,2), (4,4), (4,6), (6,2), (6,4), (6,6)\}$$

$$I_A = \{(2,2), (4,4), (6,6)\}$$

- **Cardinality of Classical Relation**

Consider n elements of the universe X being related to m elements of universe Y . When the cardinality of $X = n_X$ and the cardinality of $Y = n_Y$, then the cardinality of relation R between the two universe is

$$n_{X \times Y} = n_X \times n_Y$$

The cardinality of the power set $P(X \times Y)$ describing the relation is given by

$$n_{P(X \times Y)} = 2^{(n_X n_Y)}$$

2.16.1 Operations on classical relations

Let R and S be two separate relations on the Cartesian universe $X \times Y$. The null relation and the complete relation are defined by the relation matrices \emptyset_R and E_R . An example of a 3 X 3 form of the \emptyset_R and E_R matrices is given below:

$$\emptyset_R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad E_R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

1. Union

$$R \cup S \rightarrow \chi_{R \cup S}(x, y): \chi_{R \cup S}(x, y) = \max[\chi_R(x, y), \chi_S(x, y)]$$

2. Intersection

$$R \cap S \rightarrow \chi_{R \cap S}(x, y): \chi_{R \cap S}(x, y) = \min[\chi_R(x, y), \chi_S(x, y)]$$

3. Complement

$$\bar{R} \rightarrow \chi_{\bar{R}}(x, y) : \chi_{\bar{R}}(x, y) = 1 - \chi_R(x, y)$$

4. Containment

$$R \subset S \rightarrow \chi_R(x, y): \chi_R(x, y) \leq \chi_S(x, y)$$

5. Identity

$$\emptyset \rightarrow \emptyset_R \quad \text{and} \quad X \rightarrow E_R$$

• Composition of Classical Relations

Let R be a relation that maps elements from universe X to universe Y and S be a relation that maps elements from universe Y to universe Z

$$R \subseteq X \times Y \quad \text{and} \quad S \subseteq Y \times Z$$

The composition operations are of two types:

1. Max-min composition

The max-min composition is defined by the function theoretic expression as

$$T = R \circ S$$

$$\chi_T(x, z) = \bigvee_{y \in Y} [\chi_R(x, y) \wedge \chi_S(y, z)]$$

2. Max-product composition

The max-product composition is defined by the function theoretic expression as

$$T = R \circ S$$

$$\chi_T(x, z) = \bigvee_{y \in Y} [\chi_R(x, y) \cdot \chi_S(y, z)]$$

2.17 Fuzzy relations

A fuzzy relation is a fuzzy set defined on the Cartesian product of classical sets $\{X_1, X_2, \dots, X_n\}$ where tuples (x_1, x_2, \dots, x_n) may have varying degrees of membership $\mu_R(x_1, x_2, \dots, x_n)$ within the relation.

$$R(X_1, X_2, \dots, X_n) = \int_{x_1, x_2, \dots, x_n} \mu_R(x_1, x_2, \dots, x_n) | (x_1, x_2, \dots, x_n), \quad x_i \in X_i$$

A fuzzy relation between two sets X and Y is called binary fuzzy relation and is denoted by $R(X, Y)$. A binary relation $R(X, Y)$ is referred to as bipartite graph when $X \neq Y$. The binary relation on a single set X is called directed graph or digraph. This relation occurs when $X = Y$ and is denoted as $R(X, X)$ or $R(X^2)$.

Let

$$\tilde{X} = \{x_1, x_2, \dots, x_n\} \quad \text{and} \quad \tilde{Y} = \{y_1, y_2, \dots, y_n\}$$

Fuzzy relation $\tilde{R}(\tilde{X}, \tilde{Y})$ can be expressed by an $n \times m$ matrix as follows:

$$\tilde{R}(\tilde{X}, \tilde{Y}) = \begin{bmatrix} \mu_R(x_1, y_1) & \mu_R(x_1, y_2) & \dots & \mu_R(x_1, y_m) \\ \mu_R(x_2, y_1) & \mu_R(x_2, y_2) & \dots & \mu_R(x_2, y_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_n, y_1) & \mu_R(x_n, y_2) & \dots & \mu_R(x_n, y_m) \end{bmatrix}$$

2.17.1 Operations on fuzzy relations

1. Union

$$\mu_{\tilde{R} \cup \tilde{S}}(x, y) = \max [\mu_{\tilde{R}}(x, y), \mu_{\tilde{S}}(x, y)]$$

2. Intersection

$$\mu_{\tilde{R} \cap \tilde{S}}(x, y) = \min [\mu_{\tilde{R}}(x, y), \mu_{\tilde{S}}(x, y)]$$

3. Complement

$$\mu_{\tilde{R}}(x, y) = 1 - \mu_R(x, y)$$

4. Containment

$$\tilde{R} \subset \tilde{S} \rightarrow \mu_{\tilde{R}}(x, y) \leq \mu_{\tilde{S}}(x, y)$$

5. Inverse

The inverse of a fuzzy relation R on $X \times Y$ is denoted by R^{-1} . It is a relation on $Y \times X$ defined by $R^{-1}(y, x) = R(x, y)$ for all pairs $(y, x) \in Y \times X$.

6. Projection

For a fuzzy relation $R(X, Y)$, let $[R \downarrow Y]$ denote the projection of R onto Y . Then $[R \downarrow Y]$ is a fuzzy relation in Y whose membership function is defined by

$$\mu_{[R \downarrow Y]}(x, y) = \max_x \mu_{\tilde{R}}(x, y)$$

- Fuzzy Composition**

Let \tilde{A} be a fuzzy set on universe X and \tilde{B} be a fuzzy set on universe Y . The Cartesian product over \tilde{A} and \tilde{B} results in fuzzy relation \tilde{R} and is contained within the entire (complete) Cartesian space, i.e.,

$$\tilde{A} \times \tilde{B} = \tilde{R}$$

where

$$\tilde{R} \subset X \times Y$$

The membership function of fuzzy relation is given by

$$\mu_{\tilde{R}}(x, y) = \mu_{\tilde{A} \times \tilde{B}}(x, y) = \min [\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y)]$$

There are two types of fuzzy composition techniques:

1. Fuzzy max-min composition

There also exists fuzzy min-max composition method, but the most commonly used technique is fuzzy max-min composition. Let \tilde{R} be fuzzy relation on Cartesian space $X \times Y$, and \tilde{S} be fuzzy relation on Cartesian space $Y \times Z$.

The max-min composition of $R(X,Y)$ and $S(Y,Z)$, denoted by $R(X,Y) \circ S(Y,Z)$ is defined by $T(X,Z)$ as

$$\begin{aligned}\mu_{\tilde{T}}(x, z) &= \mu_{\tilde{R} \circ \tilde{S}}(x, z) = \max_{y \in Y} \left\{ \min \left[\mu_{\tilde{R}}(x, y), \mu_{\tilde{S}}(y, z) \right] \right\} \\ &= \bigvee_{y \in Y} \left[\mu_{\tilde{R}}(x, y) \wedge \mu_{\tilde{S}}(y, z) \right] \quad \forall x \in X, z \in Z\end{aligned}$$

The min-max composition of $R(X,Y)$ and $S(Y,Z)$, denoted by $R(X,Y) \circ S(Y,Z)$ is defined by $T(X,Z)$ as

$$\begin{aligned}\mu_{\tilde{T}}(x, z) &= \mu_{\tilde{R} \circ \tilde{S}}(x, z) = \min_{y \in Y} \left\{ \max \left[\mu_{\tilde{R}}(x, y), \mu_{\tilde{S}}(y, z) \right] \right\} \\ &= \bigwedge_{y \in Y} \left[\mu_{\tilde{R}}(x, y) \vee \mu_{\tilde{S}}(y, z) \right] \quad \forall x \in X, z \in Z\end{aligned}$$

From the above definition it can be noted that

$$\overline{R(X,Y) \circ S(Y,Z)} = \overline{R(X,Y)} \circ \overline{S(Y,Z)}$$

2. Fuzzy max-product composition

The max-product composition of $R(X,Y)$ and $S(Y,Z)$, denoted by $R(X,Y) \circ S(Y,Z)$ is defined by $T(X,Z)$ as

$$\begin{aligned}\mu_{\tilde{T}}(x, z) &= \mu_{\tilde{R} \cdot \tilde{S}}(x, z) = \min_{y \in Y} \left[\mu_{\tilde{R}}(x, y) \cdot \mu_{\tilde{S}}(y, z) \right] \\ &= \bigvee_{y \in Y} \left[\mu_{\tilde{R}}(x, y) \cdot \mu_{\tilde{S}}(y, z) \right]\end{aligned}$$

The properties of fuzzy composition can be given as follows:

$$\begin{aligned}\tilde{R} \circ \tilde{S} &\neq \tilde{S} \circ \tilde{R} \\ (\tilde{R} \circ \tilde{S})^{-1} &= \tilde{S}^{-1} \circ \tilde{R}^{-1} \\ (\tilde{R} \circ \tilde{S}) \circ \tilde{M} &= \tilde{R} \circ (\tilde{S} \circ \tilde{M})\end{aligned}$$

2.18 Advantages of Fuzzy logic

6. Mimicks human control logic.
7. Uses imprecise language.
8. Inherently robust.

9. Fails safely.
10. Modified and tweaked easily.

2.19 Disadvantages of Fuzzy logic

3. Operator's experience required.
4. System complexity.

2.20 Applications of Fuzzy logic

8. Automobile and other vehicle subsystems, such as automatic transmissions, ABS and cruise control (e.g. Tokyo monorail).
9. Air conditioners.
10. Auto focus on cameras.
11. Digital image processing, such as edge detection.
12. Rice cookers.
13. Dishwashers.
14. Elevators.
15. Washing machines and other home appliances.
16. Video game artificial intelligence.
17. Language filters on message boards and chat rooms for filtering out offensive text.
18. Pattern recognition in Remote Sensing.
19. Fuzzy logic has also been incorporated into some microcontrollers and microprocessors.
20. Bus Time Tables.
21. Predicting genetic traits. (Genetic traits are a fuzzy situation for more than one reason).
22. Temperature control (heating/cooling).
23. Medical diagnoses.
24. Predicting travel time.
25. Antilock Braking System.

Module – 4

- Fuzzy membership functions
- Fuzzification
- Methods of membership value assignments
 - Intuition
 - Inference
 - Rank ordering
- Lambda –cuts for fuzzy sets
- Defuzzification methods

4.1 Fuzzy membership functions

Membership function defines the fuzziness in a fuzzy set irrespective of the elements in the set, which are discrete or continuous. A fuzzy set \tilde{A} in the universe of discourse X can be defined as a set of ordered pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$$

where $\mu_{\tilde{A}}(\cdot)$ is called membership function of \tilde{A} . The membership function $\mu_{\tilde{A}}(\cdot)$ maps X to the membership space M , ie., $\mu_{\tilde{A}} : X \rightarrow M$. The membership value ranges in the interval $[0, 1]$ ie., the range of the membership function is a subset of the non-negative real numbers whose supremum is finite.

The three main basic features involved in characterizing membership function are the following.

1. Core

The core of a membership function for some fuzzy set \tilde{A} is defined as that region of universe that is characterized by complete membership in the set \tilde{A} . The core has elements x of the universe such that

$$\mu_{\tilde{A}}(x) = 1$$

The core of a fuzzy set may be an empty set.

2. Support

The support of a membership function for a fuzzy set \tilde{A} is defined as that region of universe that is characterized by a non zero membership in the set \tilde{A} .

$$\mu_{\tilde{A}}(x) > 0$$

A fuzzy set whose support is a single element in X with $\mu_{\tilde{A}}(x) = 1$ is referred to as a fuzzy singleton.

3. Boundary

The support of a membership functions as the region of universe containing elements that have a non zero but not complete membership. The boundary comprises those elements of x of the universe such that

$$0 < \mu_A(x) < 1$$

The boundary elements are those which possess partial membership in the fuzzy set \tilde{A} .

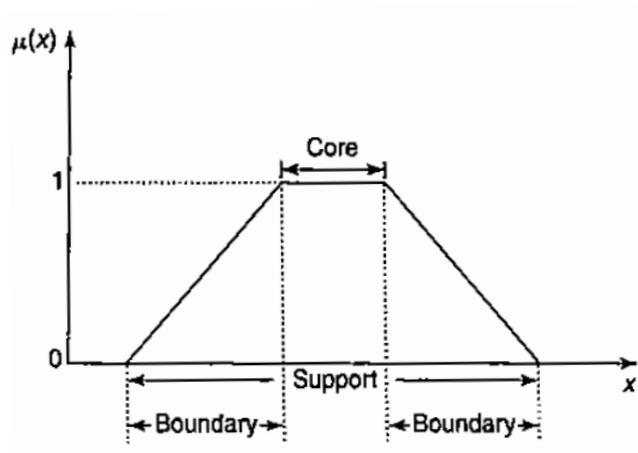


Figure 4.1: Features of membership functions

A fuzzy set whose membership function has at least one element x in the universe whose membership value is unity is called normal fuzzy set. The element for which the membership is equal to 1 is called prototypical element. A fuzzy set where no membership function has its value equal to 1 is called subnormal fuzzy set.

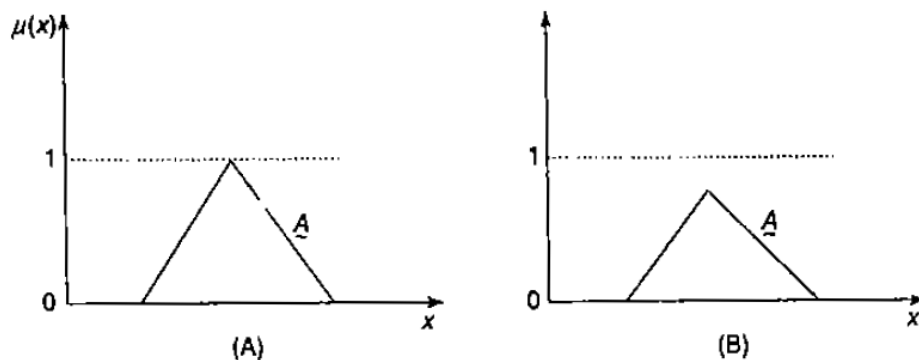


Figure 4.2: (A) Normal fuzzy set and (B) subnormal fuzzy set

A convex fuzzy set has a membership function whose membership values are strictly monotonically increasing or strictly monotonically decreasing or strictly monotonically

increasing than strictly monotonically decreasing with increasing elements in the universe. A fuzzy set possessing characteristics opposite to that of convex fuzzy set is called non convex fuzzy set.

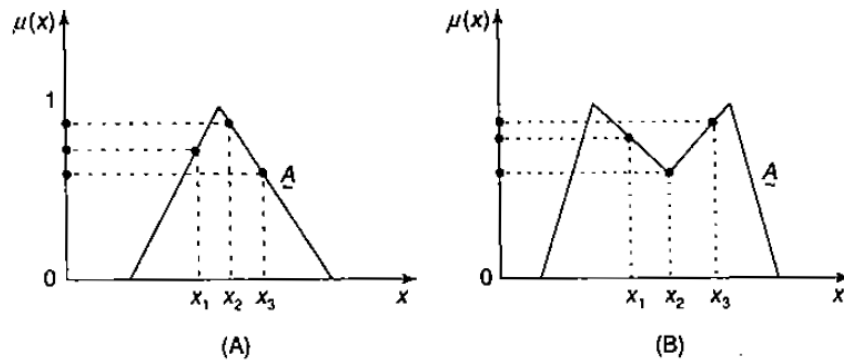


Figure 4.3: (A) Convex normal fuzzy set and (B) Nonconvex normal fuzzy set

The convex normal fuzzy set can be defined in the following way. For elements x_1 , x_2 and x_3 in a fuzzy set \tilde{A} , if the following relation between x_1 , x_2 and x_3 holds. i.e.,

$$\mu_{\tilde{A}}(x_2) \geq \min[\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_3)]$$

The element in the universe for which a particular fuzzy set \tilde{A} has its value equal to 0.5 is called crossover point of a membership function. The membership value of a crossover point of a fuzzy set is equal to 0.5, ie., $\mu_{\tilde{A}}(x) = 0.5$. There can be more than one crossover point in a fuzzy set. The maximum value of the membership function in a fuzzy set \tilde{A} is called as the height of the fuzzy set. For a normal fuzzy set, the height is equal to 1, because the maximum value of the membership function allowed is 1. Thus, if the height of a fuzzy set is less than 1, then the fuzzy set is called subnormal fuzzy set.

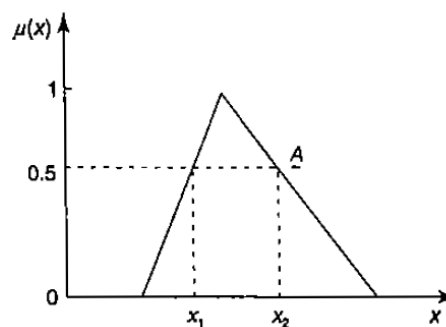


Figure 4.4: Crossover point of a fuzzy set

4.2 Fuzzification

It is the process of transforming crisp set to a fuzzy set or a fuzzy set to a fuzzifier set. For a fuzzy set $\tilde{A} = \left\{ \left(x, \mu_{\tilde{A}}(x) \right) \mid x \in X \right\}$, a common fuzzification algorithm is performed by keeping μ_i constant and x_i being transformed to a fuzzy set $Q(x_i)$ depicting the expression about x_i . The fuzzy set $Q(x_i)$ is referred to as the kernel of fuzzification. The fuzzified set \tilde{A} can be expressed as

$$\tilde{A} = \mu_1 Q(x_1) + \mu_2 Q(x_2) + \cdots + \mu_n Q(x_n)$$

where the symbol \sim means fuzzified. This process of fuzzification is called support fuzzification (s-fuzzification). There is another method of fuzzification called grade fuzzification (g-fuzzification) where x_i is kept constant and μ_i is expressed as a fuzzy set. Thus, using these methods, fuzzification is carried out.

4.3 Methods of membership value assignments

4.3.1 Intuition

Intuition method is based upon the common intelligence of human. It is the capacity of the human to develop membership functions on the basis of their own intelligence and understanding capacity. There should be an in-depth knowledge of the application to which membership value assignment is to be made.

Figure 4.5 shows various shapes of weights of people measured in kilogram in the universe. Each curve is a membership function corresponding to various fuzzy (linguistic) variables, such as very lighter, light, normal, heavy and very heavy. The curves are based on context functions and the human developing them. For example, if the weights are referred to range of thin persons we get one set of curves, and if they are referred to range of normal weighing persons we get another set and so on.

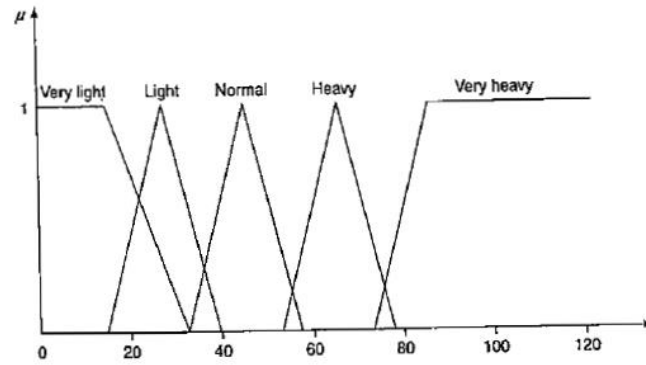


Figure 4.5: Membership function for the fuzzy variable “weight”

4.3.2 Inference

The inference method uses knowledge to perform deductive reasoning. Deduction achieves conclusion by means inference. There are various methods for performing deductive reasoning. Here the knowledge of geometrical shapes and geometry is used for defining membership values. The membership functions may be defined by various shapes: triangular, trapezoidal, bell-shaped, Gaussian and so on. The inference method here is discussed via triangular shape.

Consider a triangle, where X, Y and Z are angles such that $X \geq Y \geq Z \geq 0$, and let U be the universe of triangles i.e.,

$$U = \{(X, Y, Z) | X \geq Y \geq Z \geq 0; X + Y + Z = 180\}$$

There are various types of triangles available.

I = isosceles triangle (approximate)

E = equilateral triangle (approximate)

R = right-angle triangle (approximate)

IR = isosceles and right-angle triangle (approximate)

T = other triangles

The membership values of approximate isosceles triangle is obtained using the following definition, where $X \geq Y \geq Z \geq 0; X + Y + Z = 180^\circ$:

$$\mu_{\tilde{I}}(X, Y, Z) = 1 - \frac{1}{60^\circ} \min(X - Y, Y - Z)$$

The membership value of approximate right-angle triangle is given by

$$\mu_{\tilde{R}}(X, Y, Z) = 1 - \frac{1}{90^\circ} |X - 90^\circ|$$

Membership value of appropriate isosceles right angled triangle is obtained by taking the logical intersection of the approximate isosceles and approximate right-angle triangle membership function i.e.,

$$\tilde{IR} = \tilde{I} \cap \tilde{R}$$

and it is given by

$$\begin{aligned} \mu_{\tilde{IR}}(X, Y, Z) &= \min [\mu_{\tilde{I}}(X, Y, Z), \mu_{\tilde{R}}(X, Y, Z)] \\ \mu_{\tilde{IR}}(X, Y, Z) &= \min [\mu_{\tilde{I}}(X, Y, Z), \mu_{\tilde{R}}(X, Y, Z)] \\ &= 1 - \max \left[\frac{1}{60^\circ} \min(X - Y, Y - Z), \frac{1}{90^\circ} |X - 90^\circ| \right] \end{aligned}$$

The membership function for a fuzzy equilateral triangle is given by

$$\mu_{\tilde{E}}(X, Y, Z) = 1 - \frac{1}{180^\circ} |X - Z|$$

The membership function of other triangles, denoted by \tilde{T} , is the complement of the logical union of \tilde{I}, \tilde{R} and \tilde{E} , i.e.

$$\tilde{T} = \tilde{I} \cup \tilde{R} \cup \tilde{E}$$

By using De Morgans law, we get

$$\tilde{T} = \bar{\tilde{I}} \cap \bar{\tilde{R}} \cap \bar{\tilde{E}}$$

The membership value can be obtained using the equation

$$\begin{aligned}\mu_{\tilde{T}}(X, Y, Z) &= \min \{1 - \mu_{\tilde{I}}(X, Y, Z), 1 - \mu_{\tilde{E}}(X, Y, Z), 1 - \mu_{\tilde{R}}(X, Y, Z)\} \\ &= \frac{1}{180^\circ} \min \{3(X - Y), 3(Y - Z), 2|X - 90^\circ|, X - Z\}\end{aligned}$$

4.3.3 Rank ordering

The formation of government is based on the polling concept; to identify a best student, ranking may be performed; to buy a car, one can ask for several opinions and so on.

4.4 Lambda –cuts for fuzzy sets

Consider a fuzzy set \tilde{A} . The set A_λ ($0 < \lambda < 1$), called the lambda (λ) -cut (or alpha [α] -cut) set, is a crisp set of the fuzzy set and is defined as follows:

$$A_\lambda = \left\{ \left(x, \mu_{\tilde{A}}(x) \geq \lambda \right) \right\} \quad \lambda \in [0,1]$$

The set A_λ is called weak lambda-cut set if it consists of all the elements of a fuzzy set whose membership function have values greater than or equal to the specified value. The set A_λ is called strong lambda cut if it consist of all elements of a fuzzy set whose membership functions have values strictly greater than a specified value. A strong λ – cut set is given by

$$A_\lambda = \left\{ \left(x, \mu_{\tilde{A}}(x) > \lambda \right) \right\} \quad \lambda \in [0,1]$$

The properties of λ cut are as follows

1. $\left(\tilde{A} \cup \tilde{B} \right)_\lambda = A_\lambda \cup B_\lambda$
2. $\left(\tilde{A} \cap \tilde{B} \right)_\lambda = A_\lambda \cap B_\lambda$
3. $\left(\tilde{A} \right)_\lambda \neq \left(\overline{A_\lambda} \right)$ except when $\lambda = 0.5$
4. For any $\lambda \leq \beta$, where $0 \leq \beta \leq 1$, it is true that $A_\beta \subseteq A_\lambda$, where $A_0 = X$

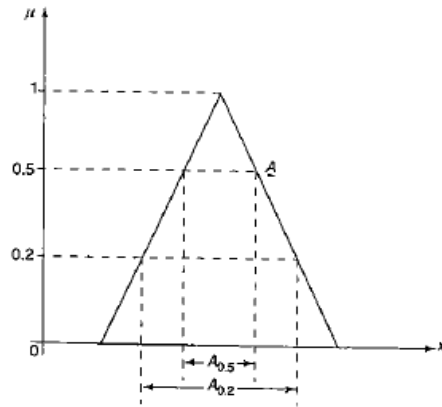


Figure 4.6: Two different λ -cut sets for a continuous-valued fuzzy set

4.5 Defuzzification methods

Defuzzification is the process of conversion of a fuzzy quantity into a precise quantity. The output of a fuzzy set process may be union of two or more fuzzy membership functions defined on the universe of discourse of the output variable.

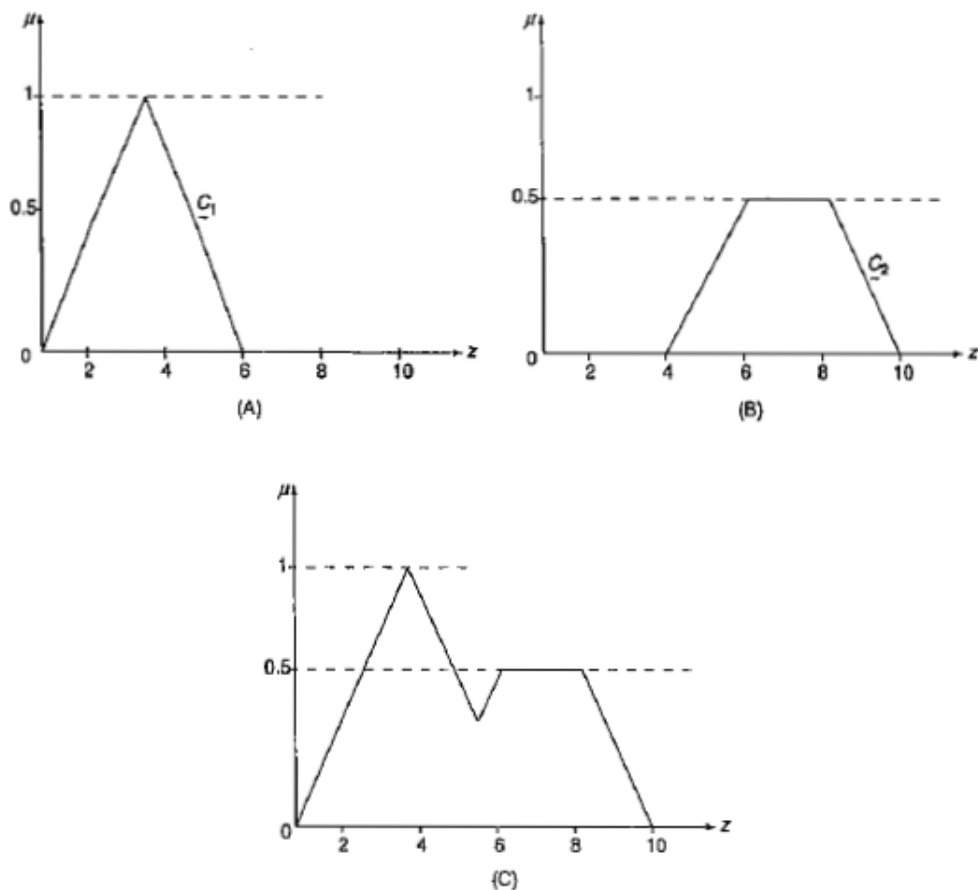


Figure 4.7: (A) First part of fuzzy output, (B) second part of fuzzy output (C) union of parts (A) and (B)

Defuzzification methods include the following:

1. Max membership principle
2. Centroid method
3. Weighted average method
4. Mean-max membership
5. Center of sums
6. Center of largest area
7. First of maxima, last of maxima

4.5.1 Max-Membership Principle

This method is also known as height method and is limited to peak output functions. This method is given by the algebraic expression

$$\mu_{\tilde{c}}(x^*) \geq \mu_{\tilde{c}}(x) \text{ for all } x \in X$$

The method is illustrated in below figure

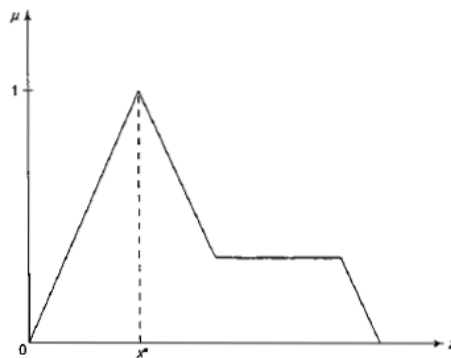


Figure 4.8: Max-membership defuzzification method

4.5.2 Centroid method

This method is also known as center of mass, center of area or center of gravity method. It is the most commonly used defuzzification method. The defuzzified output x^* is defined as,

$$x^* = \frac{\int \mu_{\tilde{c}}(x) \cdot x dx}{\int \mu_{\tilde{c}}(x) dx}$$

where the symbol \int denotes an algebraic integration. This method is illustrated in below figure

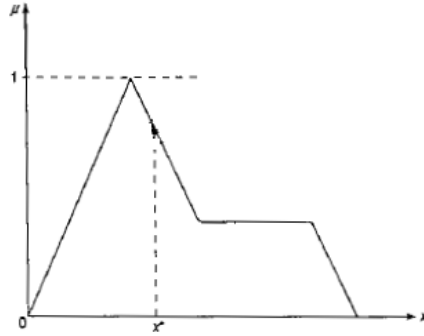


Figure 4.9: Centroid defuzzification method

4.5.3 Weighted average method

This method is valid for symmetrical output membership functions only. Each membership is weighted by its maximum membership value. The output is given by,

$$x^* = \frac{\sum \mu_c(\bar{x}_i) \cdot \bar{x}_i}{\sum \mu_c(\bar{x}_i)}$$

where \sum denotes algebraic sum and \bar{x}_i is the maximum of the i th membership function. The method is illustrated in figure 4.10, where two fuzzy sets are considered. From the figure the defuzzified output is given by

$$x^* = \frac{0.5a + 0.8b}{0.5 + 0.8}$$

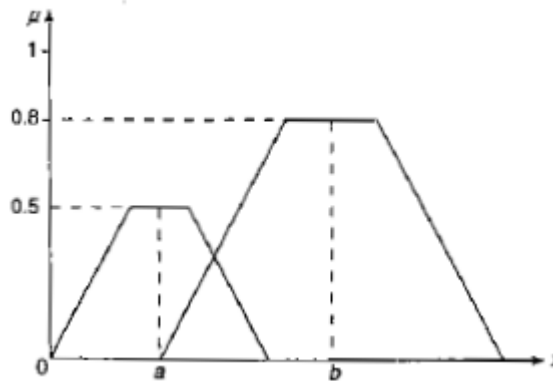


Figure 4.10: Weighted average defuzzification method (two symmetrical functions)

4.5.4 Mean-max membership

This method is also known as the middle of the maxima. This is closely related to method, except that the locations of the maximum membership can be nonunique. The output here is given by

$$x^* = \frac{\sum_{i=1}^n \bar{x}_i}{n}$$

The method is illustrated in figure 4.11, where two fuzzy sets are considered. From the figure the defuzzified output is given by

$$x^* = \frac{a + b}{2}$$

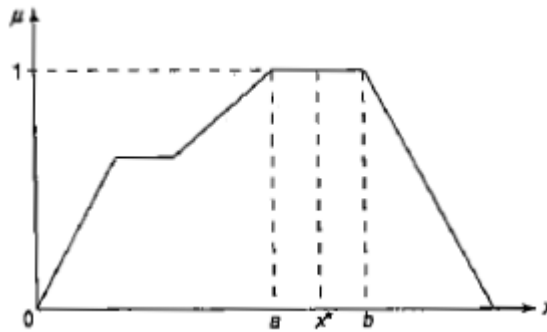


Figure 4.11: Mean-max defuzzification method

4.5.5 Center of sums

This method employs sum of the individual fuzzy subsets instead of their union. The calculations here are very fast, but the main drawback is that intersecting areas are added twice. The defuzzified value x^* is given by

$$x^* = \frac{\int_x \sum_{i=1}^n \mu_{\tilde{c}}(x) dx}{\int_x x \sum_{i=1}^n \mu_{\tilde{c}}(x) dx}$$

Figure 4.12 illustrates the center of sums method. In center of sums method, the weights are the areas of the respective membership functions, whereas in the weighted average method the weights are individual membership values.

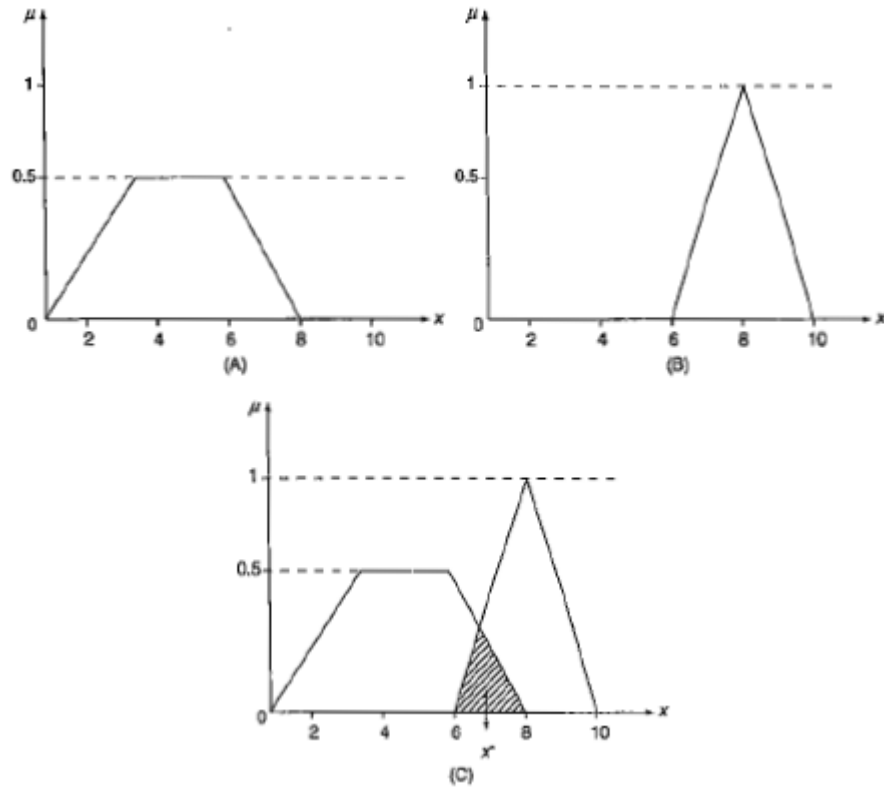


Figure 4.12: (A) First and (B) second membership functions, (C) defuzzification

4.5.6 Center of largest area

This method is adopted when the output consist of atleast two convex fuzzy subsets which are not overlapping. The output is biased towards a side of one membership function. When the output of fuzzy set has atleast two convex regions, then the center of gravity of the convex fuzzy sub region having the largest area is to obtain the defuzzified value x^* .

$$x^* = \frac{\int \mu_{\tilde{c}_i}(x) \cdot x dx}{\int \mu_{\tilde{c}_i}(x) dx}$$

where \tilde{c}_i is the convex subregion that has the largest area making up \tilde{c}_i . Figure 4.13 illustrates the center of largest area.

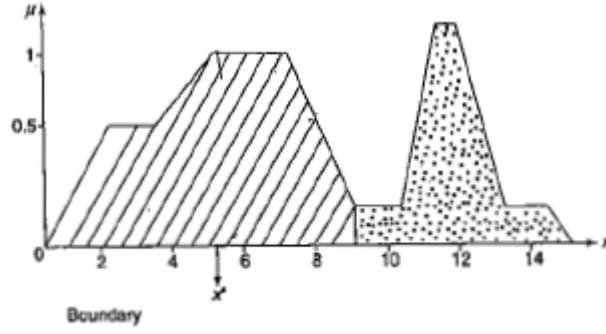


Figure 4.13: Center of largest area method

4.5.7 First of maxima, last of maxima

This method uses the overall output or union of all individual output fuzzy set \tilde{c}_i for determining the smallest value of the domain with maximized membership in \tilde{c}_j . The steps used for obtaining x^* are:

1. Initially, the maximum height in the union is found:

$$hgt(\tilde{c}_i) = \sup_{x \in X} \mu_{\tilde{c}_i}(x)$$

where sup is the supremum that is the least upper bound.

2. Then the first of maxima is found:

$$x^* = \inf_{x \in X} [x \in X | \mu_{\tilde{c}_i}(x) = hgt(\tilde{c}_i)]$$

where inf is the infimum that is the greatest lower bound.

3. After this the last maxima is found:

$$x^* = \sup_{x \in X} [x \in X | \mu_{\tilde{c}_i}(x) = hgt(\tilde{c}_i)]$$

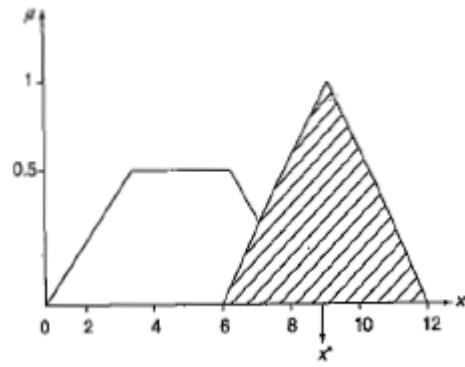


Figure 4.14: First of maxima (last of maxima) method

Module – 5

- Truth values and Tables in Fuzzy Logic
- Fuzzy propositions
- Formation of fuzzy rules
- Decomposition of rules
- Aggregation of rules
- Fuzzy Inference Systems
 - Mamdani types
 - Sugeno types
- Neuro-fuzzy hybrid systems
- Characteristics
- Classification





5.1 Truth values and Tables in Fuzzy Logic

Fuzzy logic uses linguistic variables. The values of a linguistic variable are words or sentences in a natural or artificial language. For example, height is a linguistic variable if it takes values such as tall, medium, short and so on. Consider the statement “John is tall” implies that the linguistic variable John takes the linguistic value tall. The linguistic variable provides approximate characterization of a complex problem. The name of the variable, the universe of discourse and a fuzzy subset of universe of discourse characterize a fuzzy variable. The range of possible values of a linguistic variable represents the universe of discourse of that variable. For example, the universe of discourse of the linguistic variable speed might have the range between 0 and 220 km/h and may include such fuzzy subsets as very slow, slow, medium, fast, and very fast.

A linguistic variable is a variable of a higher order than a fuzzy variable and its values are taken to be fuzzy variables. A linguistic variable is characterized by

1. name of the variable (x);
2. term set of the variable t (x);
3. syntactic rule for generating the values of x;
4. semantic rule for associating each value of x with its meaning.

A linguistic variable carries with it the concept of fuzzy set qualifiers, called hedges. Hedges are terms that modify the shape of fuzzy sets. In the fuzzy set "very tall", the word "very" is a linguistic hedge. A few popular linguistic hedges include: very, highly, slightly, moderately, plus, minus, fairly, rather.

<i>Hedge</i>	<i>Mathematical Expression</i>	<i>Graphical Representation</i>
A little	$[\mu_A(x)]^{1.3}$	
Slightly	$[\mu_A(x)]^{1.7}$	
Very	$[\mu_A(x)]^2$	
Extremely	$[\mu_A(x)]^3$	



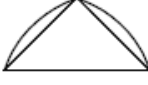

<i>Hedge</i>	<i>Mathematical Expression</i>	<i>Graphical Representation</i>
Very very	$[\mu_A(x)]^4$	
More or less	$\sqrt{\mu_A(x)}$	
Somewhat	$\sqrt{\mu_A(x)}$	
Indeed	$\begin{aligned} &2[\mu_A(x)]^2 \\ &\text{if } 0 \leq \mu_A \leq 0.5 \\ &1 - 2[1 - \mu_A(x)]^2 \\ &\text{if } 0.5 < \mu_A \leq 1 \end{aligned}$	

Table 5.1: Table showing the mathematical and graphical representation of Hedges

If it is not take the complement of membership value. For example not very short then take the complement of very short.

Truth tables define logic functions of two propositions. Let X and Y be two propositions, either of which can be true or false. The basic logic operations performed over the propositions are the following:

1. Conjunction (\wedge): X AND Y.
2. Disjunction (\vee): XOR Y.
3. Implication or conditional (\Rightarrow): IF X THEN Y.
4. Bidirectional or equivalence (\Leftrightarrow): X IF AND ONLY IF Y.

On the basis of these operations on propositions, inference rules can be formulated. Few inference rules are as follows:

$$\begin{aligned} [X \wedge (X \Rightarrow Y)] &\Rightarrow Y \\ [\bar{Y} \wedge (X \Rightarrow Y)] &\Rightarrow \bar{X} \\ [(X \Rightarrow Y) \wedge (Y \Rightarrow Z)] &\Rightarrow (X \Rightarrow Z) \end{aligned}$$

The above rules produce certain propositions that are always true irrespective of the truth values of propositions X and Y. Such propositions are called tautologies.

The truth values of propositions in fuzzy logic are allowed to range over the unit interval [0, 1]. The truth value of the proposition "Z is A," or simply the truth value of A, denoted by

$tv(A)$ is defined by a point in $[0, 1]$ (called the numerical truth value} or a fuzzy set in $[0, 1]$ (called the linguistic truth value).

$$\begin{aligned} tv(X \text{ AND } Y) &= tv(X) \wedge tv(Y) = \min\{tv(X), tv(Y)\} && (Intersection) \\ tv(X \text{ OR } Y) &= tv(X) \vee tv(Y) = \max\{tv(X), tv(Y)\} && (Union) \\ tv(NOT X) &= 1 - tv(X) && (Complement) \\ tv(X \Rightarrow Y) &= tv(X) \Rightarrow tv(Y) = \max\{1 - tv(X), \min[tv(X), tv(Y)]\} \end{aligned}$$

5.2 Fuzzy propositions

1. Fuzzy predicates

In fuzzy logic the predicates can be fuzzy, for example, tall, short, quick. Hence, we have proposition like "Peter is tall." It is obvious that most of the predicates in natural language are fuzzy rather than crisp.

2. Fuzzy-predicate modifiers

In fuzzy logic, there exists a wide range of predicate modifiers that act as hedges, for example, very, fairly, moderately, rather, slightly. These predicate modifiers are necessary for generating the values of a linguistic variable. An example can be the proposition "Climate is moderately cool," where "moderately" is the fuzzy predicate modifier.

3. Fuzzy quantifiers: The fuzzy quantifiers such as most, several, many, frequently are used in fuzzy logic. Employing these we can have proposition like "Many people are educated." A fuzzy quantifier can be interpreted as a fuzzy number or a fuzzy proposition.

4. Fuzzy qualifiers: There are four modes of qualification in fuzzy logic, which are as follows:

- **Fuzzy truth qualification**

It is expressed as " x is τ ," in which τ is a fuzzy truth value. A fuzzy truth value claims the degree of truth of a fuzzy proposition. Consider the example,

(Paul is Young) is NOT VERY True.

Here the qualified proposition is (Paul is Young) and the qualifying fuzzy truth value is "NOT Very True."

- **Fuzzy probability qualification**

It is denoted as " x is λ ," where λ is fuzzy probability. In conventional logic, probability is either numerical or an interval. In fuzzy logic, fuzzy probability is expressed by terms such as likely, very likely, unlikely, around and so on. Consider the example,

(Paul is Young) is Likely.

Here qualifying fuzzy probability is "Likely." These probabilities may be interpreted as fuzzy numbers, which may be manipulated using fuzzy arithmetic.

- **Fuzzy possibility qualification**

It is expressed as " x is π ," where π is a fuzzy possibility and can be of the following forms: possible, quire possible, almost impossible. These values can be interpreted as labels of fuzzy subsets of the real line. Consider the example,

(Paul is Young) is Almost Impossible.

Here the qualifying fuzzy possibility is "Almost Impossible."

- **Fuzzy usuality qualification**

It is expressed as "usually (X) = usually (X is F)," in which the subject X is a variable taking values in a universe of discourse U and the predicate F is a fuzzy subset of U and interpreted as a usual value of X denoted by $U(X) = F$. The propositions that are usually true or the events that have high probability of occurrence are related by the concept of usuality qualification.

5.3 Formation of fuzzy rules

The general way of representing human knowledge is by forming natural language expressions given by

IF antecedant THEN consequent.

The above expression is referred to as the IF- THEN rule based form. There are three general forms that exist for any linguistic variable. They are: (a) assignment statements; (b) conditional statements; (c) unconditional statements.

1. Assignment statements: They are of the form

y =small

Orange color = orange

a=s

Paul is not tall and not very short

Climate = autumn

Outside temperature = normal

These statements utilize "=" for assignment.

2. Conditional statements

The following are some examples.

IF y is very cool THEN stop.

IF A is high THEN B is low ELSE B is not low.

IF temperature is high THEN climate is hot.

The conditional statements use the "IF.THEN" rule-based form.

3. Unconditional statements

They can be of the form

Goto sum.

Stop.

Divide by a.

Turn the pressure low.

5.4 Decomposition of rules (Compound Rules)

A compound rule is a collection of many simple rules combined together. Any compound rule structure may be decomposed and reduced to a number of simple canonical rule forms. The rules are generally based on natural language representations.

1. Multiple conjunctive antecedents

IF x is $\tilde{A}_1, \tilde{A}_2, \dots, \tilde{A}_n$ THEN y is \tilde{B}_m

Assume a new fuzzy subset \tilde{A}_m defined as

$$\tilde{A}_m = \tilde{A}_1 \cap \tilde{A}_2 \cap \dots \cap \tilde{A}_n$$

and expressed by means of membership function

$$\mu_{\tilde{A}_m}(x) = \min [\mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x), \dots, \mu_{\tilde{A}_n}(x)]$$

In view of the fuzzy intersection operation, the compound rule may be rewritten as

$$IF \tilde{A}_m THEN \tilde{B}_m$$

2. Multiple disjunctive antecedents

$$IF x \text{ is } \tilde{A}_1 OR x \text{ is } \tilde{A}_2, \dots OR x \text{ is } \tilde{A}_n THEN y \text{ is } \tilde{B}_m$$

This can be written as

$$IF x \text{ is } \tilde{A}_n THEN y \text{ is } \tilde{B}_m$$

where the fuzzy set \tilde{A}_m is defined as

$$\tilde{A}_m = \tilde{A}_1 \cup \tilde{A}_2 \cup \tilde{A}_3 \cup \dots \cup \tilde{A}_n$$

The membership function is given by

$$\mu_{\tilde{A}_m}(x) = \max [\mu_{\tilde{A}_1}(x), \mu_{\tilde{A}_2}(x), \dots, \mu_{\tilde{A}_n}(x)]$$

which is based on fuzzy union operation.

3. Conditional statements (with ELSE and UNLESS)

Statements of the kind

$$IF \tilde{A}_1 THEN (\tilde{B}_1 ELSE \tilde{B}_2)$$

can be decomposed into two simple canonical rule forms, connected by "OR":

$$\begin{aligned} &IF \tilde{A}_1 THEN \tilde{B}_1 \\ &OR \\ &IF NOT \tilde{A}_1 THEN \tilde{B}_2 \end{aligned}$$

$$IF \tilde{A}_1 (THEN \tilde{B}_1) UNLESS \tilde{A}_2$$

can be decomposed as

$$\begin{aligned} &IF \tilde{A}_1 THEN \tilde{B}_1 \\ &OR \\ &IF \tilde{A}_2 THEN NOT \tilde{B}_1 \end{aligned}$$

$$IF \tilde{A}_1 THEN (\tilde{B}_1) ELSE IF \tilde{A}_2 THEN (\tilde{B}_2)$$

can be decomposed into the form

$$\begin{aligned} &IF \tilde{A}_1 THEN \tilde{B}_1 \\ &OR \\ &IF NOT \tilde{A}_1 AND IF \tilde{A}_2 THEN \tilde{B}_2 \end{aligned}$$

4. Nested-IF-THEN rules

The rule can be of the form " $IF \tilde{A}_1 THEN [IF \tilde{A}_2 THEN (\tilde{B}_1)]$ " can be of the form

$$IF \tilde{A}_1 AND \tilde{A}_2 THEN \tilde{B}_1$$

Thus, based on all the above mentioned methods compound rules can be decomposed into series of canonical simple rules.

5.5 Aggregation of rules

Aggregation of rules is the process of obtaining the overall consequents from the individual consequents provided by each rule. The following two methods are used for aggregation of fuzzy rules:

1. Conjunctive system of rules

For a system of rules to be jointly satisfied, the rules are connected by "and" connectives. Here, the aggregated output, y , is determined by the fuzzy intersection of all individual rule consequents y_i where $i = 1$ to n , as

$$y = y_1 \text{ and } y_2 \text{ and } \dots \text{ and } y_n$$

or

$$y = y_1 \cap y_2 \cap \dots \cap y_n$$

This aggregated output can be defined by the membership function

$$\mu_y(y) = \min[\mu_{y_1}(y), \mu_{y_2}(y), \dots, \mu_{y_n}(y)] \text{ for } y \in Y$$

2. Disjunctive system of rules

In this case, the satisfaction of at least one rule is required. The rules are connected by "or" connectives. Here, the fuzzy union of all individual rule contributions determines the aggregated output, as

$$y = y_1 \text{ or } y_2 \text{ or } \dots \text{ or } y_n$$

or

$$y = y_1 \cup y_2 \cup \dots \cup y_n$$

Again it can be defined by the membership function

$$\mu_y(y) = \max[\mu_{y_1}(y), \mu_{y_2}(y), \dots, \mu_{y_n}(y)] \text{ for } y \in Y$$

5.6 Fuzzy Inference Systems

Fuzzy rule based systems, fuzzy models, and fuzzy expert systems are generally known as systems. The key unit of a fuzzy logic system is FIS. The primary work of this system is decision making. FIS uses "IF ... THEN" rules along with connectors "OR" or "AND" for making necessary decision rules. The input to FIS may be fuzzy or crisp, but the output from FIS is always a fuzzy set.

Construction and Working Principle of FIS:

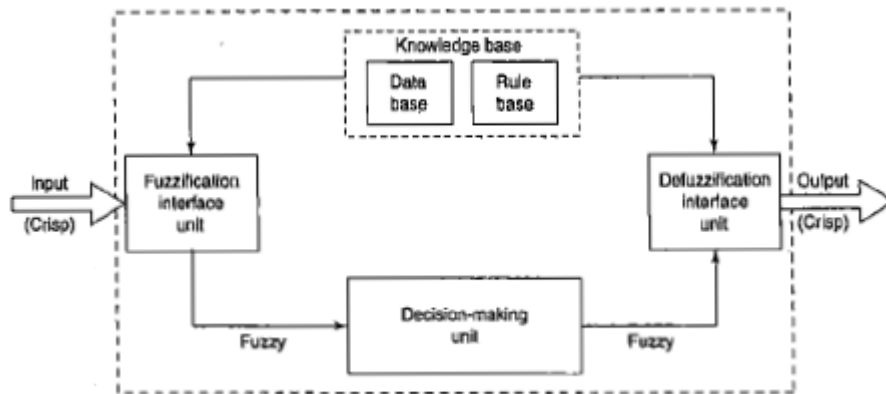


Figure 5.1: Block diagram of FIS

1. A rule base that contains numerous fuzzy IF-THEN rules.
2. A database that defines the membership functions of fuzzy sets used in fuzzy rules.
3. Decision-making unit that performs operation on the rules.
4. Fuzzification interface unit that converts the crisp quantities into fuzzy quantities.
5. Defuzzification interface that converts the fuzzy quantities into crisp quantities.

Initially, in the fuzzification unit, the crisp input is converted into a fuzzy input. Various fuzzification methods are employed for this. After this process, rule base is formed. Database and rule base are collectively called the knowledge base. Finally, defuzzification process is carried out to produce crisp output. Mainly, the fuzzy rules are formed in the rule base and suitable decisions are made in the decision-making unit.

5.6.1 Methods of FIS

There are two important types of FIS. They are

1. Mamdani FIS(1975);
2. Sugeno FIS(1985);

5.6.1.1 Mamdani types

Ehsahim Mamdani proposed this system in the year 1975 to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system. In this case, the output membership functions are expected to be fuzzy sets. After aggregation process, each output variable is a fuzzy set, hence defuzzification is important at the output stage. The steps include:

Step 1: Determine a set of fuzzy rules.

Step 2: Make the inputs fuzzy using input membership functions.

Step 3: Combine the inputs according to the fuzzy rules for establishing a rule strength.

Step 4: Determine the consequent of the rule by combining the rule strength and the output membership function.

Step 5: Combine all the consequents to get an output distribution.

Step 6: Finally, a defuzzified output distribution is obtained.

The fuzzy rules are formed using "IF-THEN" statements and "AND/OR" connectives. The consequence of the rule can be obtained in two steps:

1. By computing the rule strength completely using the fuzzified inputs from the fuzzy combination.
2. By clipping the output membership function at the rule strength

The outputs of all the fuzzy rules are combined to obtain one fuzzy output distribution. From FIS, it is desired to get only one crisp output. This crisp output may be obtained from defuzzification process. The common techniques of defuzzification used are center of mass and mean of maximum.

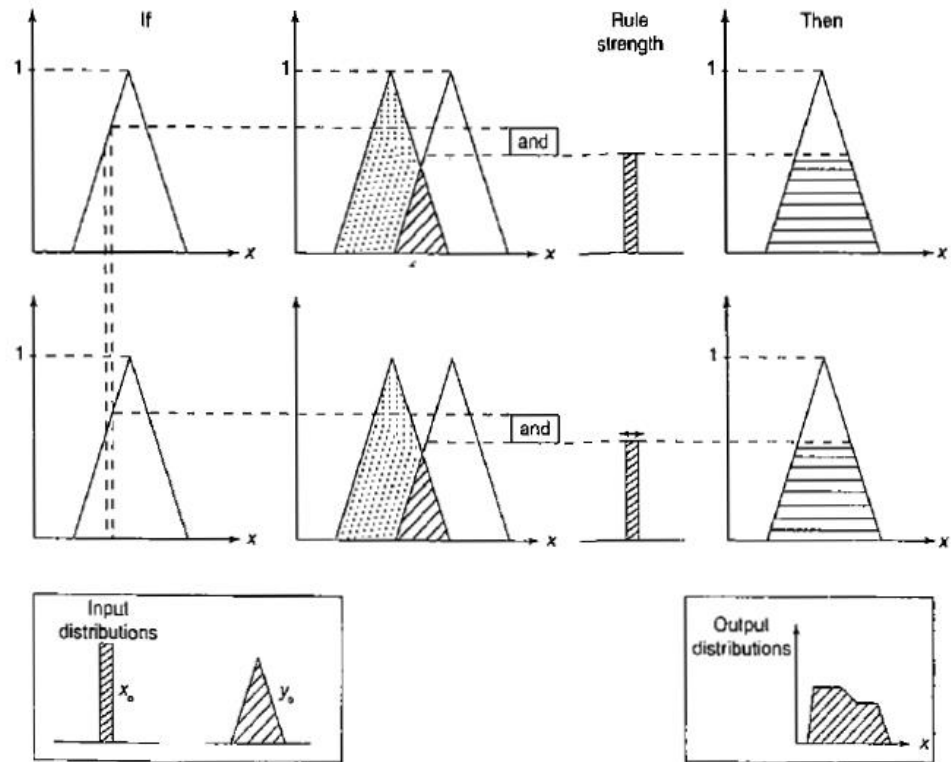


Figure 5.2: A two-input, two-rule Mamdani FIS with a fuzzy input

5.6.1.2 Sugeno types Takagi-Sugeno Fuzzy Model (TS Method)

Sugeno fuzzy method was proposed by Takagi, Sugeno and Kang in the year 1985. The format of the fuzzy rule of a Sugeno fuzzy model is given by

$$\text{IF } x \text{ is } A \text{ and } y \text{ is } B \text{ THEN } z = f(x, y)$$

where A and B are fuzzy sets in the antecedents and $z = f(x, y)$ is a crisp function.

The main steps of the fuzzy inference process namely,

1. Fuzzifying the inputs.
2. Applying the fuzzy operator.

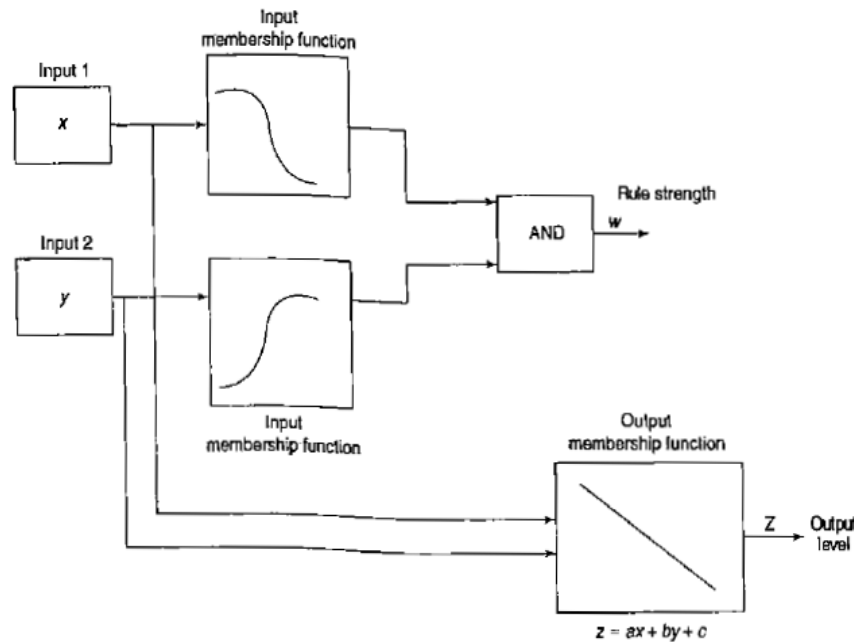


Figure 5.3: Sugeno rule

Sugeno's method can act as an interpolating supervisor for multiple linear controllers, which are to be applied, because of the linear dependence of each rule on the input variables of a system. A Sugeno model is suited for smooth interpolation of linear gains that would be applied across the input space and for modeling nonlinear systems by interpolating between multiple linear models. The Sugeno system uses adaptive techniques for constructing fuzzy models. The adaptive techniques are used to customize the membership functions.

5.6.1.3 Comparison between Mamdani and Sugeno model

The main difference between Mamdani and Sugeno methods lies in the output membership functions. The Sugeno output membership functions are either linear or constant. The difference also lies in the consequents of their fuzzy rules as a result their aggregation and defuzzification procedures differ suitably. A large number of fuzzy rules must be employed in Sugeno method for approximating periodic or highly oscillatory functions. The configuration of Sugeno fuzzy systems can be reduced and it becomes smaller than that of Mamdani fuzzy systems if nontriangular or nontrapezoidal fuzzy input sets are used. Sugeno controllers have more adjustable parameters in the rule consequent and the number of parameters grows exponentially with the increase of the number of input variables. There exist several mathematical results for Sugeno fuzzy controllers than for Mamdani controllers. Formation of Mamdani FIS is easier than Sugeno FIS.

- The main advantage of Mamdani method are:
 1. It has widespread acceptance.
 2. It is well-suitable for human input.
 3. It is intuitive.
- The advantages of Sugeno method include:
 1. It is computationally efficient.
 2. It is compact and works well with linear technique, optimization technique and adaptive technique.
 3. It is best suited for analysis.
 4. It has a guaranteed continuity of the output surface.

5.7 Neuro-fuzzy hybrid systems

It is a learning mechanism that utilizes the training and learning algorithms from neural networks to find parameters of a fuzzy system (i.e., fuzzy sets, fuzzy rules, fuzzy numbers, and so on). The neuro-fuzzy is divided into two areas:

1. Linguistic fuzzy modeling focused on interpretability (Mamdani model).
2. Precise fuzzy modeling focused on accuracy [mainly the Takagi-Sugeno-Kang (TSK) model].

5.7.1 Comparison of Fuzzy Systems with Neural Networks:

When neural networks are concerned, if one problem is expressed by sufficient number of observed examples then only it can be used. These observations are used to train the black box. Though no prior knowledge about the problem is needed extracting comprehensible rules from a neural network's structure is very difficult.

A fuzzy system, on the other hand, does not need learning examples as prior knowledge, rather linguistic rules are required. Moreover, linguistic description of the input and output variables should be given. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. This is a time consuming process.

Neural processing	Fuzzy processing
Mathematical model not necessary	Mathematical model not necessary
Learning can be done from search	A priori knowledge is needed
There are several learning algorithms	Learning is not possible
Black-box behavior	Simple interpretation and implementation

Table 5.2: Comparison of neural and fuzzy processing

5.7.2 Characteristics

An NFS approximates an n-dimensional unknown function, partly represented by training examples. Thus fuzzy rules can be interpreted as vague prototypes of the training data

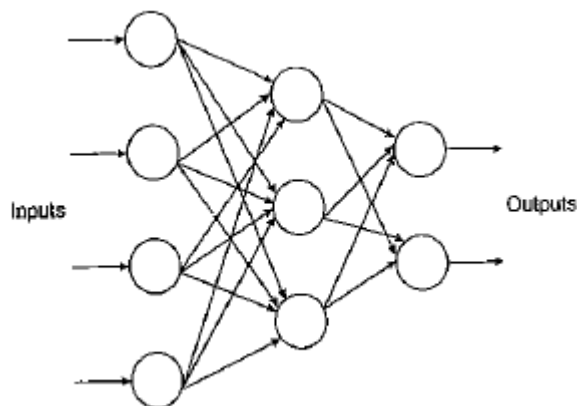


Figure 5.4: Architecture of neuro-fuzzy hybrid system

It can be represented by a three-layer feed forward neural network model. It can also be observed that the first layer corresponds to the input variables, and the second and third layers correspond to the fuzzy rules and output variables, respectively. The fuzzy sets are converted to (fuzzy) connection weights. NFS can also be considered as a system of fuzzy rules wherein the system can be initialized in the form of fuzzy rules based on the prior knowledge available. Some researchers use five layers- the fuzzy sets being encoded in the units of the second and the fourth layer, respectively

5.7.3 Classification

NFSs can be classified into the following two systems:

1. Cooperative NFSs.

2. General neuro-fuzzy hybrid systems.

5.7.3.1 Cooperative Neural Fuzzy Systems

In this type of system, both artificial neural network (ANN) and fuzzy system work independently from each other. Four different kinds of cooperative fuzzy neural networks are shown in figure 5.5.

The FNN in figure 5.5(A) learns fuzzy set from the given training data. This is done, usually, by finding membership functions with a neural network; the fuzzy sets then being determined offline. This is followed by their utilization in form the fuzzy system by fuzzy rules that are given, and not learned. The NFS in figure 5.5 (B) determines, by a neural network, the fuzzy rules from the training data. Here again, the neural networks learn offline before the fuzzy system is initialized. The rule learning happens usually by clustering on self-organizing feature maps. There is also the possibility of applying fuzzy clustering methods to obtain rules.

For the neuro-fuzzy model shown in figure 5.5 (C), the parameters of membership function are learnt online, while the fuzzy system is applied. This means that, initially, fuzzy rules and membership functions must be defined beforehand. Also, in order to improve and guide the learning step, the error has to be measured. The model shown in figure 5.5 (D) determines the rule weights for all fuzzy rules by a neural network. A rule is determined by its rule weight-interpreted as the influence of a rule. They are then multiplied with the rule output.

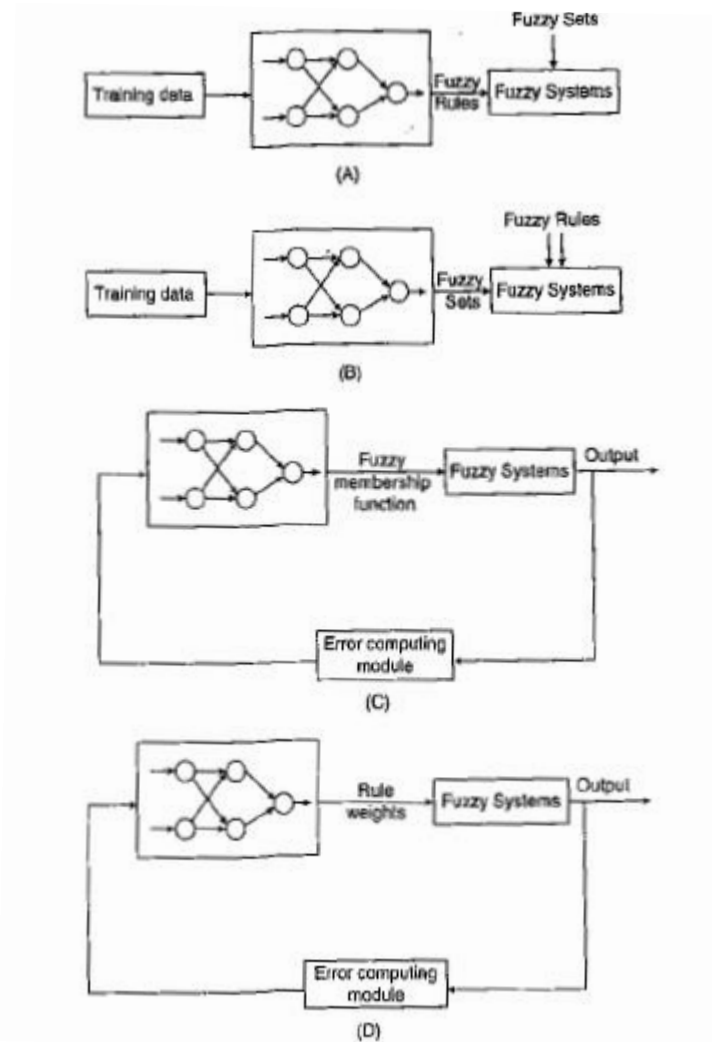


Figure 5.5: Cooperative neural fuzzy systems

5.7.3.2 General Neuro-Fuzzy Hybrid Systems (General NFHS)

The architecture of general NFHS gives it an advantage because there is no communication between fuzzy system and neural network. Figure 5.6 illustrates an NFHS. In this figure the rule base of a fuzzy system is assumed to be a neural network; the fuzzy sets are regarded as weights and the rules and the input and output variables as neurons. The choice to include or discard neurons can be made in the learning step. Also, the fuzzy knowledge base is represented by the neurons of the neural network; this overcomes the major drawbacks of both underlying systems.

Membership functions expressing the linguistic terms of the inference rules should be formulated for building a fuzzy controller. However, in fuzzy systems, no formal approach exists to define these functions. Any shape, such as Gaussian or triangular or bell shaped or

trapezoidal, can be considered as a membership function with an arbitrary set of parameters. Thus for fuzzy systems, the optimization of these functions in terms of generalizing the data is very important; this problem can be solved by using neural networks. Using learning rules, the neural network must optimize the parameters by fixing a distinct shape of the membership functions; for example, triangular. But regardless of the shape of the membership functions, training data should also be available.

The neuro fuzzy hybrid systems can also be modeled in another method. In this case, the training data is grouped into several clusters and each cluster is designed to represent a particular rule. These rules are defined by the crisp data points and are not defined linguistically. The testing can be carried out by presenting a random testing sample to the trained neural network.

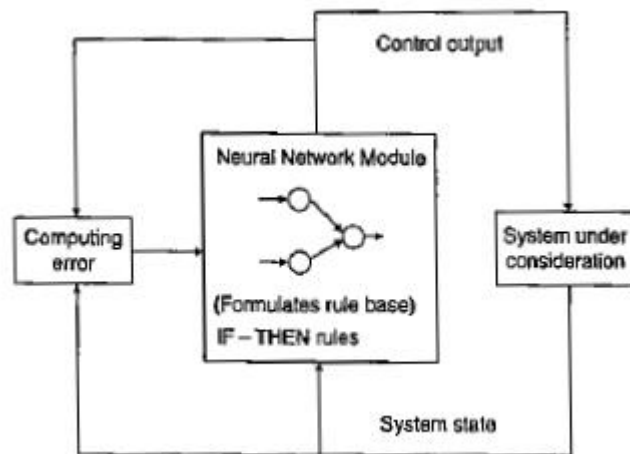


Figure 5.6: A general neuro-fuzzy hybrid system

Module – 6

- Introduction to genetic algorithm
- Operators in genetic algorithm
 - Coding
 - Selection
 - Cross over
 - Mutation
 - Stopping condition for genetic algorithm flow
- Genetic neuro hybrid systems
- Genetic-Fuzzy rule based system

6.1 Introduction to genetic algorithm

6.1.1 Genetic Algorithms

Genetic algorithm (GA) is reminiscent of sexual reproduction in which the genes of two parents combine to form those of their children. When it is applied to problem solving, the basic premise is that we can create an initial population of individual's representing possible solutions to a problem we are trying to solve. Each of these individuals has certain characteristics that make them more or less fit as members of the population. The more fit members will have a higher probability of mating and producing offspring that have a significant chance of retaining the desirable characteristics of their parents than the less fit members. This method is very effective at finding optimal or near-optimal solutions in a wide variety of problems

6.1.2 Biological Background

The science that deals with the mechanisms responsible for similarities and differences in a species is called Genetics. The word "genetics" is derived from the Greek word "genesis" meaning "to grow" or "to become."

- **Cell**

Every animal/human cell is a complex of many "small" factories that work together. The center of all this is the cell nucleus. The genetic information is contained in the cell nucleus.

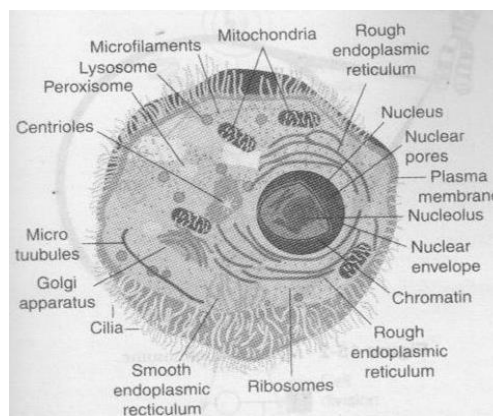


Figure 6.1: Anatomy of the animal cell

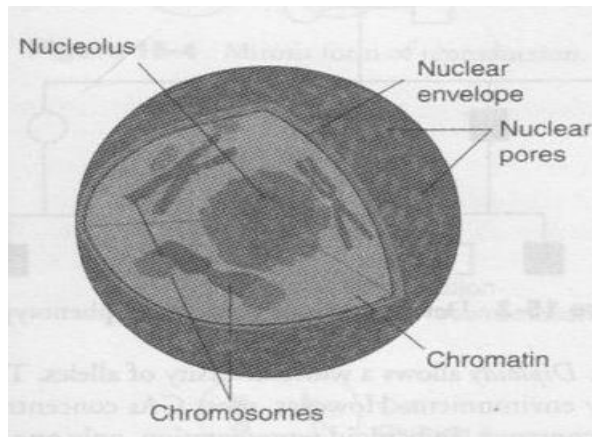


Figure 6.2: The cell nucleus

- **Chromosomes**

All the genetic information gets stored in the chromosomes. Each chromosome is build of deoxyribonucleic acid (DNA). In humans, chromosomes exist in pairs (23 pairs found). The chromosomes are divided into several parts called **genes**. Genes code the properties of species, i.e., the characteristics of an individual. The possibilities of combination of the genes for one property are called **alleles**, and a gene can take different alleles. For example, there is a gene for eye color, and all the different possible alleles are black, brown, blue and green (since no one has red or violet eyes!). The set of all possible alleles present in a particular population forms a **gene pool**. This gene pool can determine all the different possible variations for the future generations. The size of the gene pool helps in determining the diversity of the individuals in the population. The set of all the genes of a specific species is called **genome**. Each and every gene has a unique position on the genome called **locus**.

- **Genetics**

A particular individual, the entire combination of genes is called **genotype**. The phenotype describes the physical aspect of decoding a genotype to produce the phenotype. Chromosomes contain two sets of genes. These are known as **diploids**.

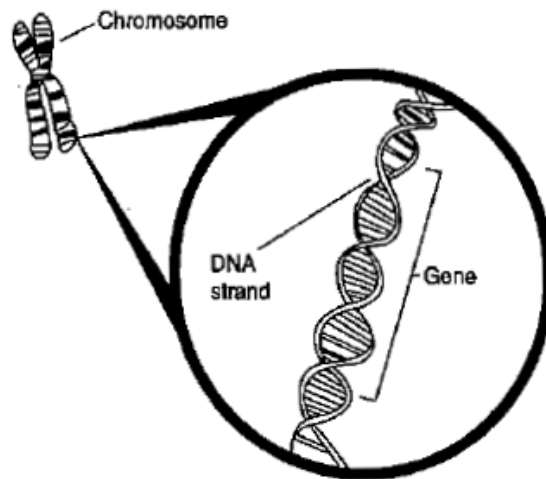


Figure 6.3: Model of chromosome

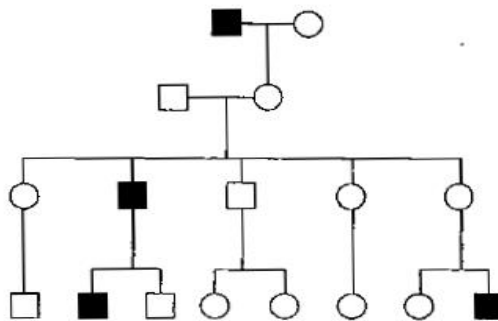


Figure 6.4: Development of genotype to phenotype

- **Reproduction**

Reproduction of species via genetic information is carried out by the following;

1. **Mitosis:** In mitosis the same genetic information is copied to new offspring. There is no exchange of information. This is a normal way of growing of multicell structures, such as organs.

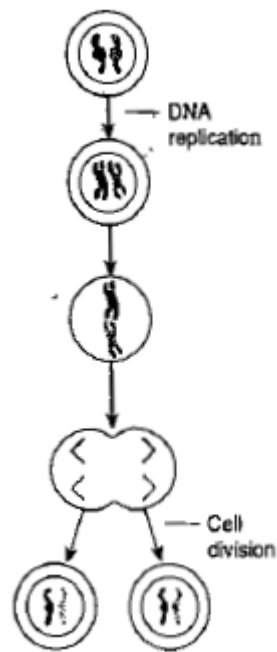


Figure 6.5: Mitosis form of reproduction

2. **Meiosis:** Meiosis forms the basis of sexual reproduction. When meiotic division takes place, two gametes appear in the process. When reproduction occurs, these two gametes conjugate to a zygote which becomes the new individual.

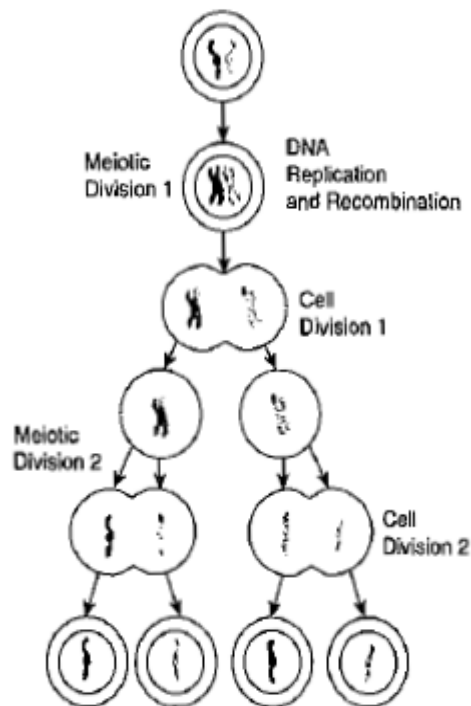


Figure 6.6: Meiosis form of reproduction

<u>Natural evolution</u>	<u>Genetic algorithm</u>
Chromosome	String
Gene	Feature or Character
Allele	Feature value
Locus	String position
Genotype	Structure or coded string
Phenotype	Parameter set, a decoded structure

Table 6.1: Comparison of natural evolution and genetic algorithm terminology

- **Natural Selection**

The origin of species is based on "Preservation of favorable variations and rejection of unfavorable variations." The variation refers to the differences shown by the individual of a species and also by offspring's of the same parents. There are more individuals born than can survive, so there is a continuous struggle for life. Individuals with an advantage have a greater chance of survival, i.e., the survival of the fittest.

For example,

Giraffe with long necks can have food from tall trees as well from the ground; on the other hand, goat and deer having smaller neck can have food only from the ground. As a result, natural selection plays a major role in this survival process.

6.1.3 Basic Terminologies in Genetic Algorithm

- **Individuals**

An individual is a single solution.

- **Genes**

Genes are the basic "instructions" for building a GA. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem, without actually being the

solution. A gene is a bit string of arbitrary lengths. The bit string is a binary representation of number of intervals from a lower bound.

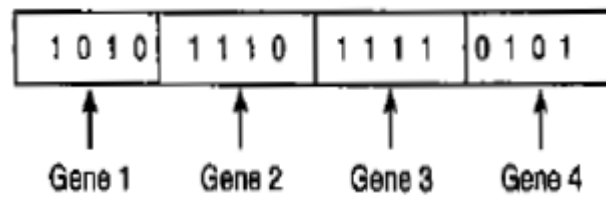


Figure 6.7: Representation of a gene

- **Fitness**

The fitness of an individual in a GA is the value of an objective function for its phenotype. For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one.

- **Populations**

A population is a collection of individuals. A population consists of a number of individuals being tested, the phenotype parameters defining the individuals and some information about the search space. The two important aspects of population used in GAs are:

1. The initial population generation.
2. The population size.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure 6.8: Population

6.1.4 General Genetic Algorithm

Step 1: Create a random initial state

An initial population is created from a random selection of solutions (which are analogous to chromosomes). This is unlike the situation for symbolic AI systems, where the initial State in a problem is already given.

Step 2: Evaluate fitness

A value for fitness is assigned to each solution (chromosome) depending on how close it actually is w solving the problem (thus arriving to the answer of the desired problem). (These "solutions" are not to be confused with "answers" to the problem; think of them as possible characteristics that the system would employ in order to reach the answer.)

Step 3: Reproduce (and children mutate)

Those chromosomes with a higher fitness value are more likely to reproduce offspring (which can mutate after reproduction). The offspring is a product of the father and mother, whose composition consists of a combination of genes from the two (this process is known as "crossover").

Step 4: Next generation

If the new generation contains a solution that produces an output that is close enough or equal to the desired then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached.

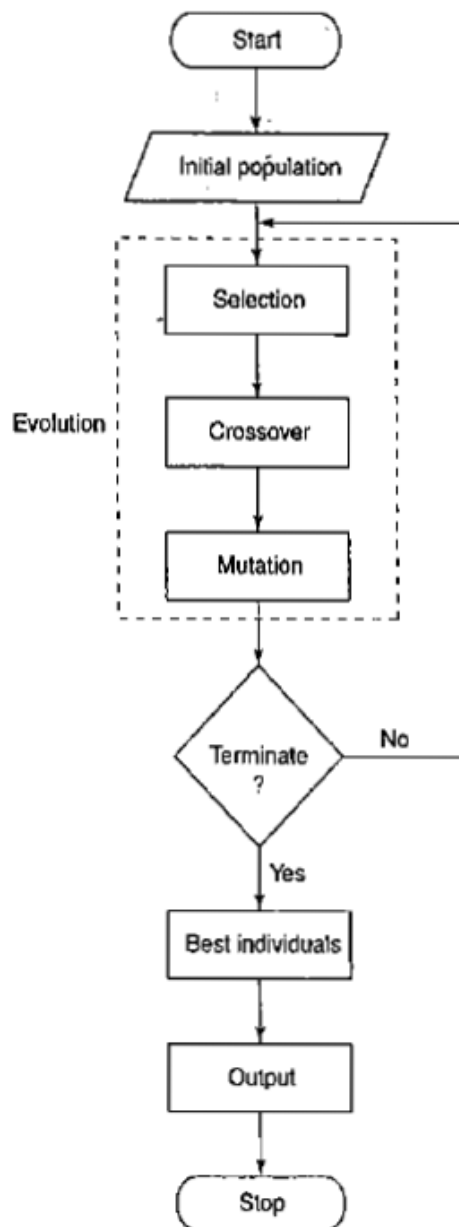


Figure 6.9: Flowchart for generic algorithm

6.2 Operators in genetic algorithm

The basic operators include: encoding, selection, recombination and mutation operators. The operators with their various types are explained with necessary examples.

6.2.1 Encoding

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other object.

- **Binary Encoding**

Each chromosome encodes a binary (bit) string. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. Another possibility is that the whole string can represent a number. The way bit strings can code differs from problem to problem.

Binary encoding gives many possible chromosomes with a smaller number of alleles. Binary coded strings with 1s and 0s are mostly used. The length of the string depends on the accuracy.

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 0 0

Figure 6.10: Binary encoding

- **Octal Encoding**

This encoding uses string made up of octal numbers (0-7).

Chromosome 1	03467216
Chromosome 2	15723314

Figure 6.11: Octal encoding

- **Hexadecimal Encoding**

This encoding uses string made up of hexadecimal numbers (0-9, A-F).

Chromosome 1	9CE7
Chromosome 2	3DBA

Figure 6.12: Hexadecimal encoding

- **Permutation Encoding (Real Number Coding)**

Every chromosome is a string of numbers, represented in a sequence. In permutation encoding, every chromosome is a string of integer/real values, which represents number in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Figure 6.13: Permutation encoding

- **Value Encoding**

Every chromosome is a string of values and the values can be anything connected to the problem. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or characters to some complicated objects.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Figure 6.14: Value encoding

- **Tree Encoding**

This encoding is mainly used for evolving program expressions for genetic programming. Every chromosome is a tree of some objects such as functions and commands of a programming language

6.2.2 Selection

Selection is the process of choosing two parents from the population for crossing. After deciding on an encoding, the next step is to decide how to perform selection, i.e., how to choose individuals in the population that will create offspring for the next generation and how many offspring each will create. The purpose of selection is to emphasize fitter individuals in the population in hopes that their offspring have higher fitness. According to Darwin's theory of evolution the best ones survive to create new offspring.

Selection is a method that randomly picks chromosomes out of the population according to their evaluation function. The higher the fitness function, the better chance that an individual will be selected. The selection pressure is defined as the degree to which the better individuals are favored. The higher the selection pressure, the more the better individuals are favored. This selection pressure drives the GA to improve the population fitness over successive generations.

Two types of selection:

- **Proportionate-based selection**

Proportionate-based selection picks out individuals based upon their fitness values relative to the fitness of the other individuals in the population.

- **Ordinal-based selection**

Ordinal-based selection schemes select individuals not upon their raw fitness, but upon their rank within the population. This requires that the selection pressure is independent of the fitness distribution of the population, and is solely based upon the relative ordering (ranking) of the population.

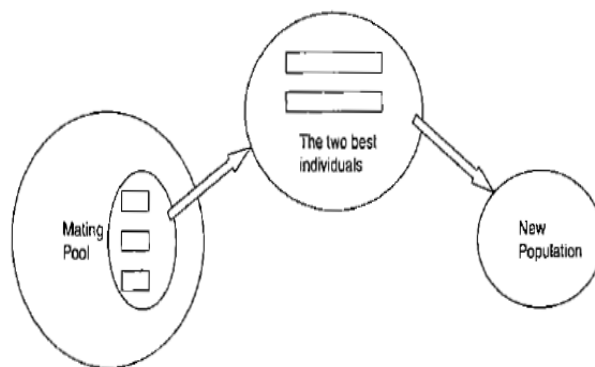


Figure 6.15: Selection

6.2.2.1 Roulette Wheel Selection

The commonly used reproduction operator is the proportionate reproductive operator where a string is selected from the mating Pool with a probability proportional to the fitness. The principle of Roulette selection is a linear search through a Roulette wheel with the slots in the wheel weighted in proportion to the individual's fitness values. A target value is set, which is a random proportion of the sum of the fitness's the population. The population is stepped through until the target value is reached. This is only a moderately strong selection technique, since fit individuals are not guaranteed to be selected for, but somewhat have a greater chance. A fit individual will contribute more to the target value, but if it does not exceed it, the next chromosome in line has a chance, and it may be weak. It is essential that the population not be sorted by fitness, since this would dramatically bias the selection.

The Roulette process can also be explained as follows: The expected value of an individual is individual's fitness divided by the actual fitness of the population. Each individual is assigned a slice of the Roulette wheel, the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation.

This method is implemented as follows:

5. Sum the total expected value of the individuals in the population. Let it be T.
6. Repeat N times:
 - i. Choose a random integer "r" between 0 and T.
 - ii. Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to "r." The individual whose expected value puts the sum over this limit is the one selected.

6.2.2.2 Random Selection

This technique randomly selects a parent from the population. In terms of disruption of generic codes, random selection is a little more disruptive, on average, than Roulette wheel selection.

6.2.2.3 Rank Selection

Rank Selection ranks the population and every chromosome receives fitness from the ranking. The worst has fitness 1 and the best has fitness N. It also keeps up selection pressure when the fitness variance is low. In effect, potential parents are selected and a tournament is held to decide which of the individuals will be the parent.

There are many ways this can be achieved and two suggestions are:

1. Select a pair of individuals at random. Generate a random number R between 0 and 1. If $R < r$ use the first individual as parent. If $R \geq r$ then use second individual as parent. This process is repeated to select second parent.
2. Select two individuals at random. The individual with the highest evaluation function becomes the parent. This process is repeated to select second parent.

6.2.2.4 Tournament Selection

The best individual from the tournament is the one with the highest fitness, who is the winner of N_u . Tournament competitions and the winner are then inserted into the mating pool. The tournament competition is repeated until the mating pool for generating new offspring is filled. The mating pool comprising the tournament winner has higher average population fitness. The fitness difference provides the selection pressure, which drives GA to improve the fitness of the succeeding genes. This method is more efficient and leads to an optimal solution.

6.2.2.5 Boltzmann Selection

In Boltzmann selection, a continuously varying temperature controls the rate of selection according to a preset schedule. The temperature starts high, which means that the selection pressure is low. The temperature is gradually lowered, which gradually increases the selection pressure, thereby allowing the GA to narrow in more closely to the best part of the search space while maintaining the appropriate degree of diversity.

6.2.2.6 Stochastic Universal Sampling

Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in Roulette wheel selection. Here equally spaced pointers are placed

over the line, as many as there are individuals to be selected. Consider N Pointer the number of individuals to be selected, then the distance between the pointers are $1/N$ Pointer and the position of the first pointer is given by a randomly generated number in the range $[0, 1/N \text{ Pointer}]$. For 6 individuals to be selected, the distance between the pointers is $1/6 = 0.167$. Figure 6.16 shows the selection for the above example. Sample of 1 random number in the range $[0, 0.167]$: 0.1. After selection the mating population consists of the individuals, 1,2,3,4,6,8.

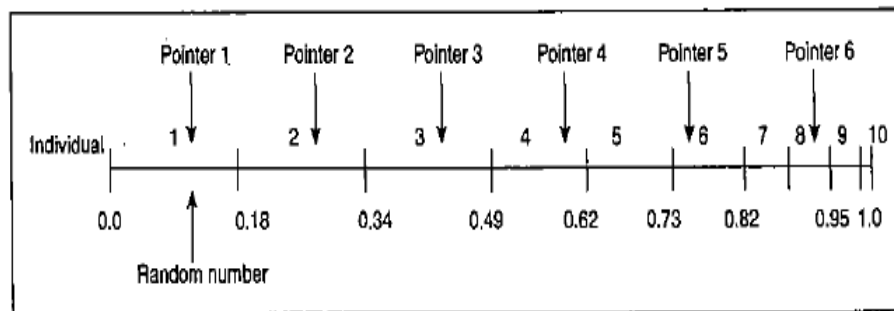


Figure 6.16: Stochastic universal sampling

Stochastic universal sampling ensures selection of offspring that is closer to what is deserved as compared to Roulette wheel selection.

6.2.3 Crossover (Recombination)

Crossover is the process of taking two parent solutions and producing from them a child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Crossover is a recombination operator that proceeds in three steps:

1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

6.2.3.1 Single-Point Crossover

The two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanged. Here, a cross site or crossover point is selected randomly along the length of the mated strings and bits next to the cross sites are exchanged. If appropriate site is chosen, better children can be obtained by combining good parents, else it severely hampers string quality.

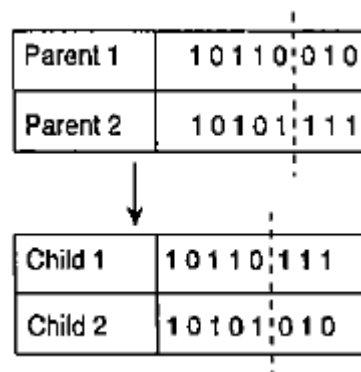


Figure 6.17: Single – point crossover

6.2.3.2 Two-Point Crossover

In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents.

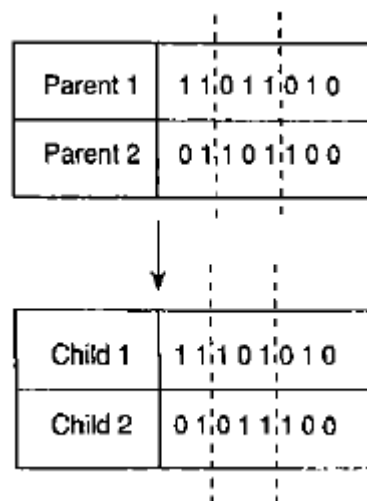


Figure 6.18: Two – point crossover

Originally, GAs were using one point crossover which cuts two chromosomes in one point and splices the two halves to create new ones. But with this one-point crossover, the head

and the tail of one chromosome cannot be passed together to the offspring. If both the head and the tail of a chromosome contain good genetic information, none of the offspring obtained directly with one-point crossover will share the two good features. Using a two-point crossover one can avoid this drawback, and so it is generally considered better than one-point crossover.

5.2.3.3 Multipoint Crossover (N-Point Crossover)

There are two ways in this crossover. One is even number of cross sites and the other odd number of cross sites. In the case of even number of cross sites, the cross sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross sites, a different cross point is always assumed at the string beginning.

5.2.3.4 Uniform Crossover

Each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosomes. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 in the mask the gene is copied from the second parent. A new crossover mask is randomly generated for each pair of parents. In the below figure, while producing child 1, when there is a 1 in the mask, the gene is copied from parent 1 else it is copied from parent 2. On producing child 2, when there is a 1 in the mask, the gene is copied from parent 2, and when there is a 0 in the mask, the gene is copied from the parent 1.

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

Figure 6.19: Uniform crossover

6.2.3.5 Three-Parent Crossover

In this crossover technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring, otherwise the bit from the third parent is taken for the offspring.

Parent 1	1 1 0 1 0 0 0 1
Parent 2	0 1 1 0 1 0 0 1
Parent 3	0 1 1 0 1 1 0 0
Child	0 1 1 0 1 0 0 1

Figure 6.20: Three - parent crossover

6.2.3.6 Crossover with Reduced Surrogate

The reduced surrogate operator constraints crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.

6.2.3.7 Shuffle Crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

6.2.3.8 Precedence Preservative Crossover

The operator passes on precedence relations of operations given in two parental permutations to one offspring at the same rate, while no new precedence relations are introduced. PPX is illustrated below for a problem consisting of six operations A-F. The operator works as follows:

1. A vector of length Sigma, sub $i = 1$ to m_i , representing the number of operations involved in the problem, is randomly filled with elements of the set [1, 2].

2. This vector defines the order in which the operations are successively drawn from parent 1 and parent 2.
3. We can also consider the parent and offspring permutations as lists, for which the operations "append" and "delete" are defined.
4. First we start by initializing an empty offspring.
5. The leftmost operation in one of the two parents is selected in accordance with the order of parents given in the vector.
6. After an operation is selected, it is deleted in both parents.
7. Finally the selected operation is appended to the offspring.
8. Step 7 is repeated until both parents are empty and the offspring contains all operations involved.

Parent permutation 1	A	B	C	D	E	F
Parent permutation 2	C	A	B	F	D	E
Select parent no. (1/2)	1	2	1	1	2	2
Offspring permutation	A	C	B	D	F	E

Figure 6.21: Precedence preservative crossover (PPX)

6.2.3.9 Ordered Crossover

Ordered two-point crossover is used when the problem is order based, for example in assembly line balancing. Given two parent chromosomes, two random crossover points are selected partitioning them into a left, middle and right portions. The ordered crossover behaves in the following way: child 1 inherits its left and right section from parent 1, and its middle section is determined by the genes in the middle section of parent 1 in the order in which the values appear in parent 2. A similar process is applied to determine child 2.

Parent 1: 4	2		1	3		6	5	Child 1: 4	2		3	1		6	5
Parent 2: 2	3		1	4		5	6	Child 2: 2	3		4	1		5	6

Figure 6.22: Ordered crossover

6.2.3.10 Partially Matched Crossover

Partially matched crossover (PMX) can be applied usefully in the TSP. Indeed, TSP chromosomes are simply sequences of integers, where each integer represents a different city and the order represents the time at which a city is visited. Under this representation, known as permutation encoding, we are only interested in labels and not alleles. PMX proceeds as follows:

1. The two chromosomes are aligned.
2. Two crossing sites are selected uniformly at random along the strings, defining a matching section.
3. The matching section is used to effect a cross through position-by-position exchange operation.
4. Alleles are moved to their new positions in the offspring.

```
Name 9 8 4 . 5 6 7 . 1 3 2 1 0   Allele 1 0 1 . 0 0 1 . 1 1 0 0
Name 8 7 1 . 2 3 1 0 . 9 5 4 6   Allele 1 1 1 . 0 1 1 . 1 1 0 1
```

Figure 6.23: Given strings

```
Name 9 8 4 . 2 3 1 0 . 1 6 5 7       Allele 1 0 1 . 0 1 0 . 1 0 0 1
Name 8 1 0 1 . 5 6 7 . 9 2 4 3       Allele 1 1 1 . 1 1 1 . 1 0 0 1
```

Figure 6.24: Partially matched crossover

The matching section defines the position-wise exchanges that must take place in both parents to produce the offspring. The exchanges are read from the matching section of one chromosome to that of the other. In the example illustrate in Figure 6.23, numbers that exchange places are 5 and 2, 6 and 3, and 7 and 10.

6.2.3.11 Crossover Probability

Crossover probability is a parameter to describe how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parents chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new- generation is made from exact copies of chromosomes from old population.

6.2.4 Mutation

After crossover, the strings are subjected to mutation. Mutation prevents the algorithm to be trapped in a local minimum. Mutation plays the role of recovering the lost genetic materials as well as for randomly distributing generic information. Mutation is viewed as a background operator to maintain genetic diversity in the population. It introduces new generic structures in the population by randomly modifying some of its building blocks. Mutation helps escape from local minima's trap and maintains diversity in the population. It also keeps the gene pool well stocked, thus ensuring ergodicity. A search space is said to be ergodic if there is a non-zero probability of generating any solution from any population state.

6.2.4.1 Flipping

Flipping of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated. A parent is considered and a mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in parent chromosome is flipped (0 to 1 and 1 to 0) and child chromosome is produced. In the case illustrated in Figure 6.25, 1 occurs at 3 places of mutation chromosome, the corresponding bits in parent chromosome are flipped and the child is generated.

Parent	1 0 1 1 0 1 0 1
Mutation chromosome	1 0 0 0 1 0 0 1
Child	0 0 1 1 1 1 0 0

Figure 6.25: Mutation flipping

6.2.4.2 Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.

Parent	1 0 1 1 0 1 0 1
Child	1 1 1 1 0 0 0 1

Figure 6.26: Interchanging

6.2.4.3 Reversing

A random position is chosen and the bits next to that position are reversed and child chromosome is produced.

Parent	1 0 1 1 0 1 0 1
Child	1 0 1 1 0 1 1 0

Figure 6.27: Reversing

6.2.4.4 Mutation Probability

It decides how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed; if it is 0%, nothing is changed.

6.2.5 Stopping Condition for Generic Algorithm Flow

1. Maximum generations: The GA stops when the specified number of generations has evolved.
2. Elapsed time: The generic process will end when a specified time has elapsed.

Note: If the maximum number of generation has been reached before the specified time has elapsed, the process will end.

3. No change in fitness: The genetic process will end if there is no change to the population's best fitness for a specified number of generations.
4. Stall generations: The algorithm stops if there is no improvement in the objective function for a sequence of consecutive generations of length "Stall generations."
5. Stall time limit. The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to "Stall time limit."

6.2.5.1 Best Individual

A best individual convergence criterion stops the search once the minimum fitness in the population drops below the convergence value. This brings the search to a faster conclusion, guaranteeing at least one good solution.

6.2.5.2 Worst Individual

Worst individual terminates the search when the least fit individuals in the population have fitness less than the convergence criteria. This guarantees the entire population be of minimum standard, although the best individual may not be significantly better than the worst. In this case, a stringent convergence value may never be met, in which case the search will terminate after the maximum has been exceeded.

6.2.5.3 Sum of Fitness

In this termination scheme, the search is considered to have satisfaction converged when the sum of the fitness in the entire population is less than or equal to the convergence value in the population record. This guarantees that virtually all individuals in the population will be within a particular fitness range, although it is better to pair this convergence criteria with weakest gene replacement, otherwise a few unfit individuals in the population will blow out the fitness sum. The population size has to be considered while setting the convergence value.

6.2.5.4 Median Fitness

Here at least half of the individuals will be better than or equal to the convergence value, which should give a good range of solutions to choose from.

6.2 Genetic-neuro hybrid systems

A neuro-genetic hybrid or a genetic-neuro hybrid system is one in which a neural network employs a genetic algorithm to optimize its structural parameters that define its architecture.

6.2.1 Properties of Genetic Neuro-Hybrid Systems

- a. The parameters of neural networks are encoded by generic algorithms as a string of properties of the network, that is, chromosomes. A large population of chromosomes is

generated, which represent the many possible parameter sets for the given neural network.

- b. Genetic Algorithm- Neural Network, or GANN, has the ability to locate the neighborhood of the optimal solution quickly, compared to other conventional search strategies.

6.3.2 Genetic Algorithm Based Back-Propagation Network (BPN)

BPN is a method of reaching multi-layer neural networks how to perform a given task. Here learning occurs during this training phase.

The limitations of BPN are as follows:

1. BPN do not have the ability to recognize new patterns; they can recognize patterns similar to those they have learnt.
2. They must be sufficiently trained so that enough general features applicable to both seen and unseen instances can be extracted; there may be undesirable effects due to over training the network.

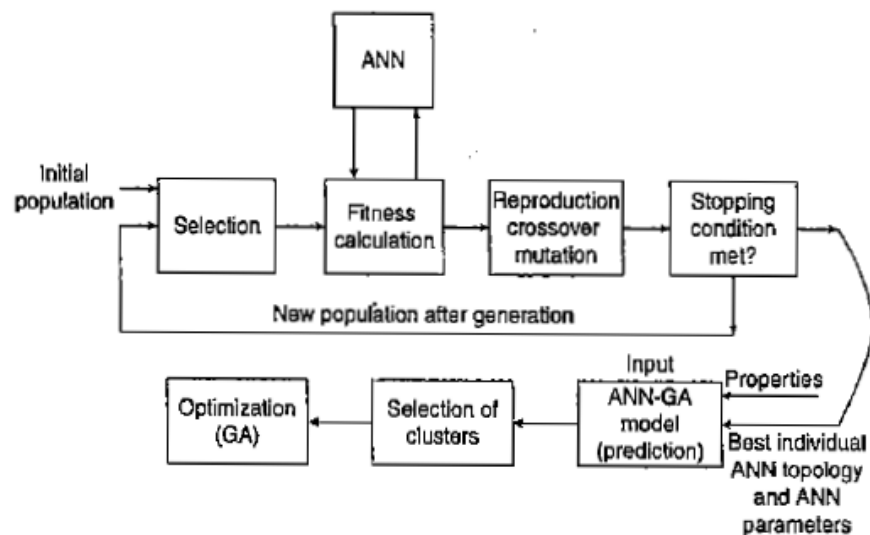


Figure 6.28: Block diagram of genetic-neuro hybrids

6.3.2.1 Coding

Assume a BPN configuration $n-l-m$ where n is the number of neurons in the input layer, l is the number of neurons in the hidden layer and m is the number of output layer neurons. The number of weights to be determined is given by

$$(n+m)l$$

Each weight (which is a gene here) is a real number. Let d be the number of digits (gene length) in weight.

6.3.2.2 Weight Extraction

In order to determine the fitness values, weights are extracted from each chromosome. Let $a_1, a_2, \dots, a_d, \dots, a_l$ represent a chromosome and let $a_{pd+1}, a_{pd+2}, \dots, a_{(p+1)d}$ represent p^{th} gene ($p \geq 0$) in the chromosomes.

The actual weight w_p is given by

$$w_p = \begin{cases} -\frac{a_{pd+2}10^{d-2} + a_{pd+3}10^{d-3} + \dots + a_{(p+1)d}}{10^{d-2}}, & \text{if } 0 \leq a_{pd+1} < 5 \\ +\frac{a_{pd+2}10^{d-2} + a_{pd+3}10^{d-3} + \dots + a_{(p+1)d}}{10^{d-2}}, & \text{if } 5 \leq a_{pd+1} \leq 9 \end{cases}$$

6.3.2.3 Fitness Function

Consider the matrix given by

$$\begin{pmatrix} (x_{11}, x_{21}, x_{31}, \dots, x_{n1}) & (y_{11}, y_{21}, y_{31}, \dots, y_{n1}) \\ (x_{12}, x_{22}, x_{32}, \dots, x_{n2}) & (y_{12}, y_{22}, y_{32}, \dots, y_{n2}) \\ (x_{13}, x_{23}, x_{33}, \dots, x_{n3}) & (y_{13}, y_{23}, y_{33}, \dots, y_{n3}) \\ \vdots & \vdots \\ (x_{1m}, x_{2m}, x_{3m}, \dots, x_{nm}) & (y_{1m}, y_{2m}, y_{3m}, \dots, y_{nm}) \end{pmatrix}$$

where X and Y are the inputs and targets, respectively. Compute initial population I_0 of size ' j '. Let $O_{10}, O_{20}, \dots, O_{j0}$ represent ' j ' chromosomes of the initial population I_0 . Let the weights extracted for each of the chromosomes upto j th chromosome be $w_{10}, w_{20}, w_{30}, \dots, w_{j0}$. For n number of inputs and m number of outputs, let the calculated output of the considered BPN be

$$\left\{ \begin{array}{l} (c_{11}, c_{21}, c_{31}, \dots, c_{n1}) \\ (c_{12}, c_{22}, c_{32}, \dots, c_{n2}) \\ (c_{13}, c_{23}, c_{33}, \dots, c_{n3}) \\ \vdots \\ (c_{1m}, c_{2m}, c_{3m}, \dots, c_{nm}) \end{array} \right\}$$

As a result, the error here is calculated by

[illegible]

The fitness function is further from this root mean square error given by

$$FF_n = \frac{1}{E_{rmse}}$$

6.3.2.4 Reproduction of Offspring

In this process, before the parents produce the offspring with better fitness, the mating pool has to be formulated. This is accomplished by neglecting the chromosome with minimum fitness and replacing it with a chromosome having maximum fitness, In other words, the fittest individuals among the chromosomes will be given more chances to participate in the generations and the worst individuals will be eliminated.

6.3.2.5 Convergence

The convergence for generic algorithm is the number of generations with which the fitness value increases towards the global optimum. Convergence is the progression towards increasing uniformity. When about 95% of the individuals in the population share the same fitness value then we say that a population has converged.

6.2.2 Advantages of Neuro-Genetic Hybrids

The various advantages of neuro-genetic hybrid are as follows:

1. GA performs optimization of neural network parameters with simplicity, ease of operation, minimal requirements and global perspective.

2. GA helps to find out complex structure of ANN for given input and the output data set by using its learning rule as a fitness function.
3. Hybrid approach ensembles a powerful model that could significantly improve the predictability of the system under construction.

6.3 Genetic-Fuzzy rule based system(GFRBSs)

For modeling complex systems in which classical tools are unsuccessful, due to them being complex or imprecise, an important tool in the form of fuzzy rule based systems has been identified.

The main objectives of optimization in fuzzy rule based system are as follows:

1. The task of finding an appropriate knowledge base (KB) for a particular problem. This is equivalent to parameterizing the fuzzy KB (rules and membership functions).
2. To find those parameter values that are optimal with respect to the design criteria.

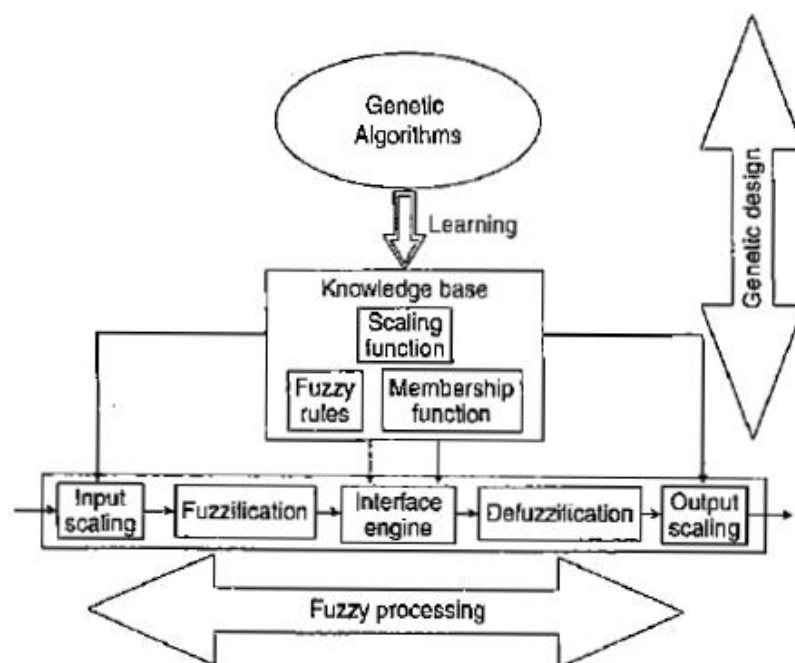


Figure 6.29: Block diagram of genetic fuzzy systems

Considering a GFRBS, one has to decide which parts of the knowledge base (KB) are subject to optimization by the GA. The KB of a fuzzy system is the union of qualitatively different components and not a homogeneous structure.

Tuning	Learning
It is concerned with optimization of an existing FRRS.	It constitutes an automated design method for fuzzy rule sets that start from scratch.
Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and/or the scaling functions, DB parameters	Learning processes perform a more elaborated search in the space of possible RBs or whole KB and do not depend on a predefined set of rules.

Table 6.2: Tuning versus learning problems

6.3.1 Genetic Tuning Process

The task of tuning the scaling functions and fuzzy membership function is important in FRBS design.

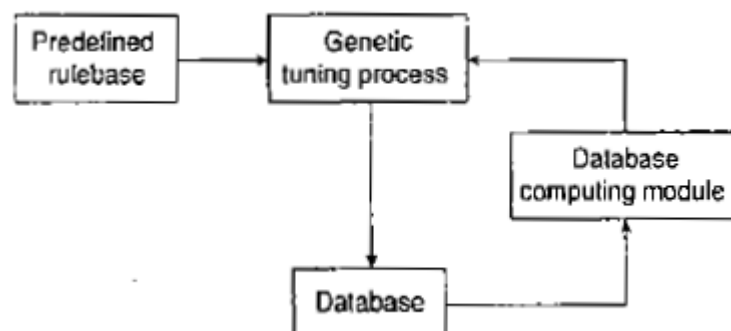


Figure 6.30: Process of using the DB

- **Tuning Scaling Function**

The universes of discourse where fuzzy membership function are defined are normalized by scaling functions applied to the input and output variables of FRBSs. In case of linear the scaling functions are parameterized by a single factor or either by specifying a lower and upper bound. In case of non-linear scaling, the scaling functions are parameterized by one or several contradiction/dilation parameters. These parameters are adapted such that the scaled universe of discourse matches the underlying variable range.

- **Tuning Membership Functions:**

For FRBSs of the descriptive (using linguistic variables) or the approximate (using fuzzy variables) type, the structure of the chromosome is different. In the process of tuning the membership functions in a linguistic model, the entire fuzzy partitions are encoded into the chromosome and in order to maintain the global semantic in the RB, it is globally adapted.

6.4.2 Genetic Learning of Rule Bases

When considering a rule based system and focusing on learning rules, there are three main approaches that have been applied in the literature:

1. Pittsburgh approach.
2. Michigan approach.
3. Iterative rule learning approach

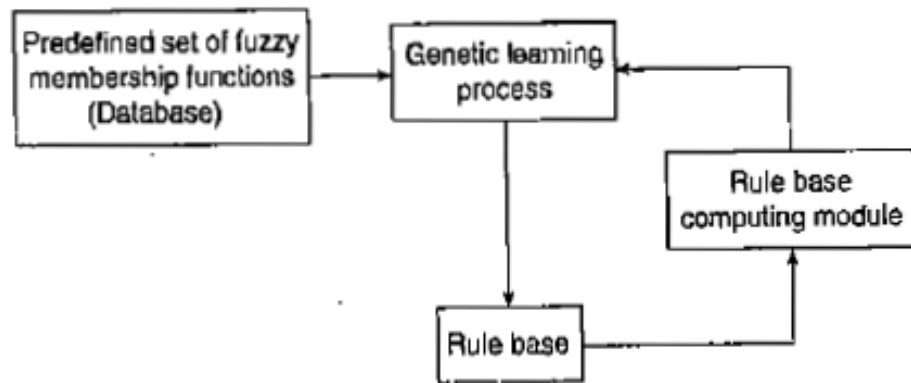


Figure 6.31: Genetic learning of rule base

The Pittsburgh approach is characterized by representing an entire rule set as a generic code (chromosome), maintaining a population of candidate rule sets and using selection and generic operators to produce new generations of rule sets. The Michigan approach considers a different model where the members of the population are individual rules and a rule set is represented by the entire population. In the third approach, the iterative one, chromosomes code individual rules, and a new rule is adapted and added to the rule set, in an iterative fashion, in every run of the genetic algorithm.

6.4.3 Genetic Learning of Knowledge Base

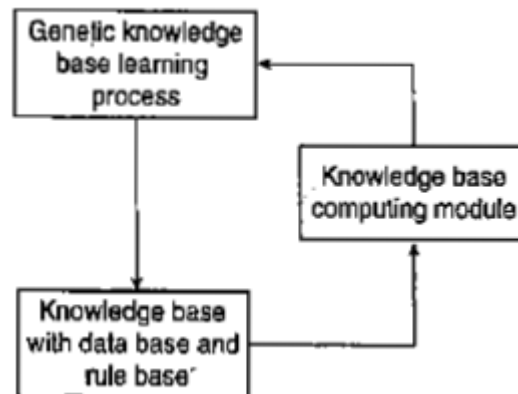


Figure 6.32: Genetic learning of the knowledge base

Genetic learning of a KB includes different generic representations such as variable length chromosomes, multi-chromosome genomes and chromosomes encoding single rules instead of a whole KB as it deals with heterogeneous search spaces. As the complexity of the search space increases, the computational cost of the generic search also grows. To overcome this issue an option is to maintain a GFRBS that encodes individual rules rather than entire KB. In this manner one can maintain a flexible, complex rule space in which the search for a solution remains feasible and efficient. The three learning approaches as used in case of rule base can also be considered here: Michigan, Pittsburgh, and iterative rule learning approach.

Problems
