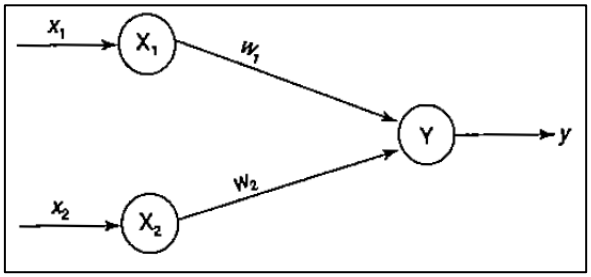
UNIT - I

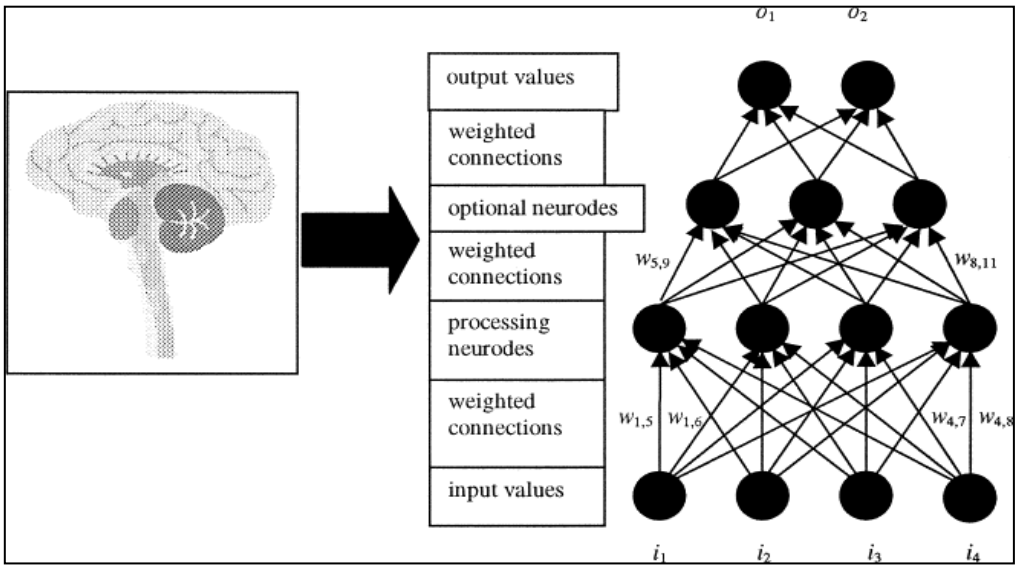
1. ARTIFICIAL NEURAL NETWORKS:

- Neural Networks – objective of the NN research is to develop a computational device for modeling the brain to perform various computational task at a faster rate than the traditional system

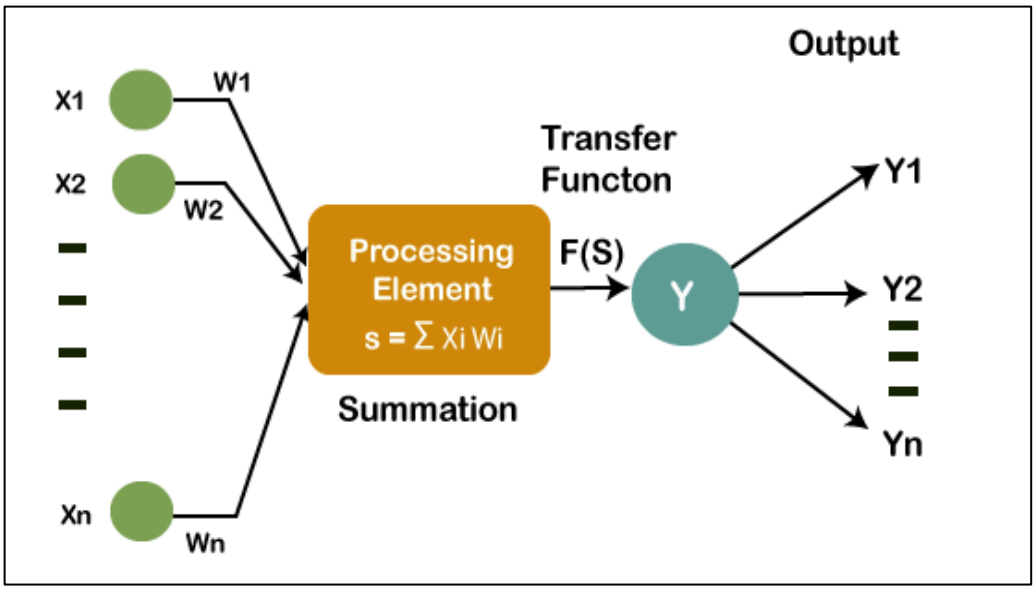


Architecture of a simple artificial neuron net

- It comprises of three main layers
- **Input layer:** The input layer accepts all the inputs that are provided by the programmer.
- **Hidden layer:** In between the input and output layer, there is a set of hidden layers on which computations are performed that further results in the output.
- **Output layer:** After the input layer undergoes a series of transformations while passing through the hidden layer, it results in output that is delivered by the output layer.

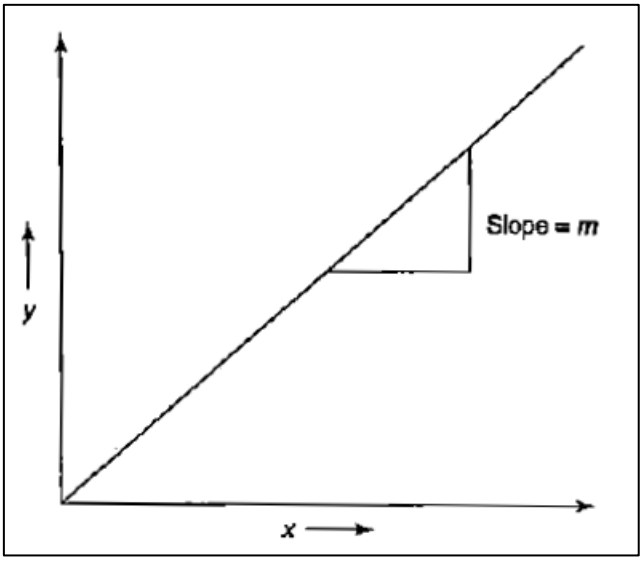


- ANN – Performs various tasks such as
  - pattern matching and classification,
  - approximation function,
  - approximation,
  - vector quantization and
  - data clustering
- ANN – Possess large number of highly interconnected processing elements called nodes or units or neurons which usually operate in parallel and are configured in regular infrastructures
- Each neuron is connected with other by a connection link
- Each connection link is associated with weights which contains the information about the input signal
- This information is used by neuron net to solve a particular problem
- ANN behavior is characterized by their ability to learn, recall and generalize training pattern or data similar to that of human brain
- ANN processing elements are called neurons or artificial neuron
- Each neuron has an internal state of its own, is called activation or activity level of neuron which is the fn. of the i/p the neuron receives
- Instead of directly getting into the working of Artificial Neural Networks, lets breakdown and try to understand Neural Network's basic unit, which is called a **Perceptron**.
- So, a perceptron can be defined as a neural network with a single layer that classifies the linear data. It further constitutes four major components, which are as follows;
  - Inputs
  - Weights and Bias
  - Summation Functions
  - Activation or transformation function
    - Linear activation function and Nonlinear activation function



Y(input)= +x1wI1+.x2w2+…

Y(out)= function (Yin)



Graph for y = mx

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

Pros of ANNs:

- Adaptability,
- Parallel Processing,
- Generalization,
- Fault Tolerance,

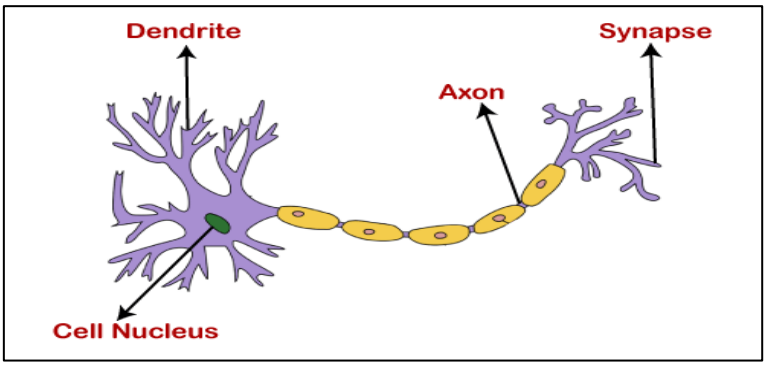
- Feature Extraction.

Cons of ANNs:

- Black Box Nature,
- Training Complexity,
- Overfitting,
- Data Dependency,
- Hyperparameter Sensitivity.

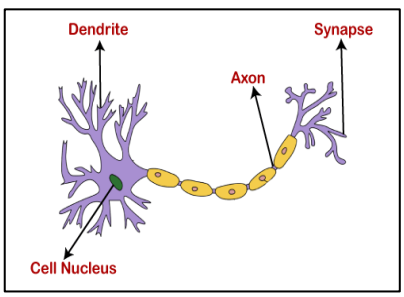
2. BIOLOGICAL NEURONS:

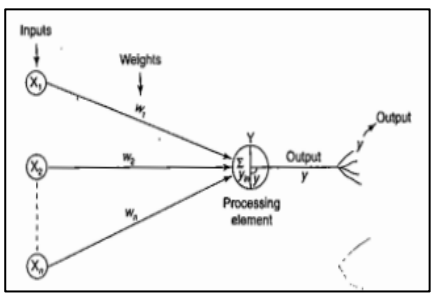
- Human brain consists of huge number of neurons, approximately 10 power 11 with interconnection
- Three main parts:
  - Soma or Cell body – Where the cell nucleus is located
  - Dendrites – Where the nerve is connected to the cell body – Are tree-like networks made of nerve fiber connected to the cell body
  - Axon – Which carries the impulses of the neuron – An axon is a single long connection extending from the cell body and carrying signals from the neuron – The end of the axon splits in to fine strands
  - Each strand terminates in to a small bulb like organ called Synapse – through synapse the neuron introduces signals to other nearby neurons, the receiving end of the synapse on the nearby neuron can be found both on dendrites and on the cell body- 10 power 4 of synapse per neuron in the human brain



- The **dendrites** will act as a receiver that receives signals from other neurons, which are then passed on to the **cell body**.
- The cell body will perform some operations that can be a summation, multiplication, etc. After the operations are performed on the set of input, then they are transferred to the next neuron via **axon**, which is the transmitter of the signal for the neuron.
- Electric pulses are passed b/w synapse and dendrites – this type of signal transmission involves a chemical process in which specific transmitter substances are released from the sending side of the junctions. These releases decrease or increase in the electric potential inside the body of the receiving cell
- If the electric potential reaches threshold, then the receiving cell fires and a pulse or action potential of fixed strength and duration is sent out through axon to the synaptic junction of the other cell,
- After firing the cell has to wait for a period of time is called **Refractory Period** before it can fire again. The Synapse are said to be
  - Inhibitory – let passing impulses hinder the firing of the receiving cell
  - Excitatory – let passing impulses cause the firing of the receiving cell

Biological Neuron	Artificial Neuron
Cell	Neuron
Dendrites	Weights or Interconnection
Axon	Output





Pros of Biological Neurons:

- Complex processing,
- Adaptability,
- Energy efficiency,
- Robustness,
- Integration with biological systems.

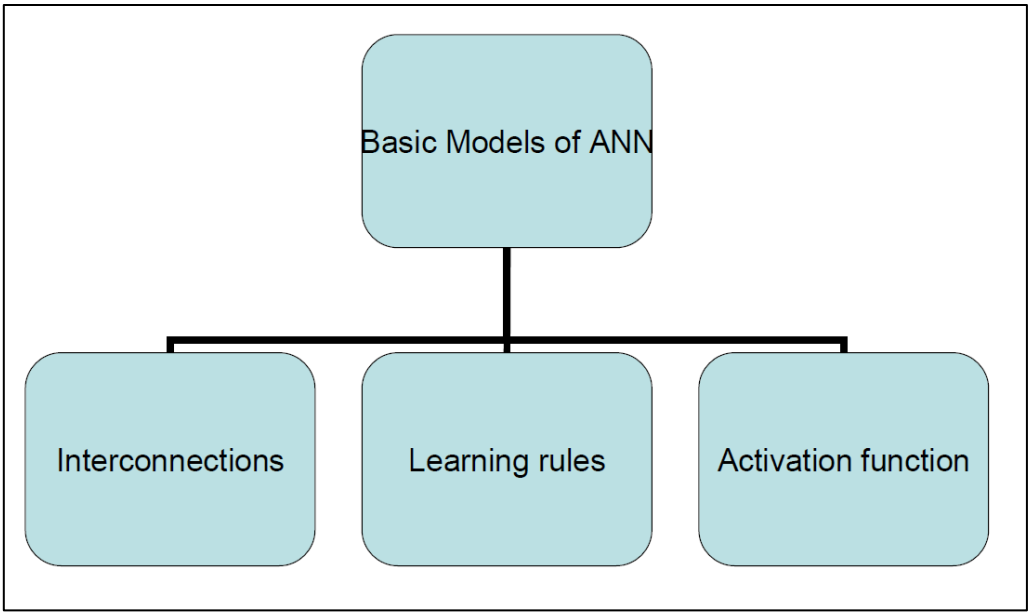
Cons of Biological Neurons:

- Limited speed,
- Vulnerability to disease and aging,
- Lack of control and reproducibility,
- Ethical concerns,
- Complexity.

Comparison between Biological neuron and Artificial neuron:

Criteria	Biological Neuron	Artificial Neuron
Speed	Milliseconds	Nanoseconds
Processing	Massive Parallel operation simultaneously	Parallel operation but faster
Size and Complexity	10^11 and interconnection is 10^15 Complexity is more	Based on the chosen application and the n/w designer
Storage capacity (memory)	Stores the info. in interconnection or in synapse strength	Contiguous memory locations`
Tolerance	Fault tolerant capability	No fault tolerance
Control Mechanism	No control unit for the brain	Control unit in CPU

3. BASIC MODELS OF ANN:



- Specified by three basic entities:
  - The models synaptic interconnections
  - The training or learning rule adopted for updating or adjusting connection weights
  - Their activation functions
- Connections: ANN has set of highly interconnected processing elements called neurons.
- The interconnection with their weighted linkages holds the informative knowledge with itself or others.
- Arrangements and geometry of their interconnections are essential and the function each processing element should be noted.
- The arrangement of neuron to form layers and the connection pattern formed within and b/w layers is called network architecture.

Connections:

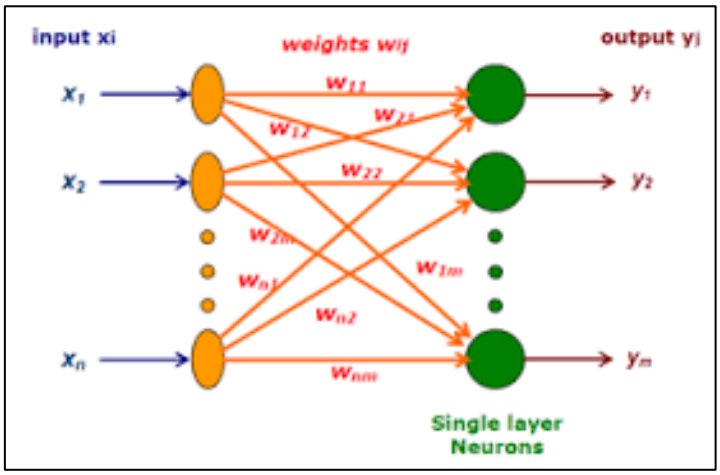
The arrangement of neurons to form layers and the connection pattern formed within and between layers is called the network architecture. There exist five basic types of connection architecture.

They are:

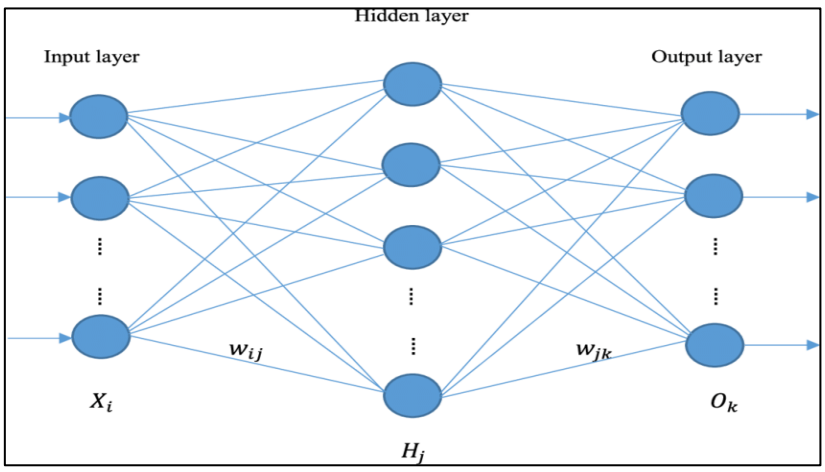
1. **Single layer feed forward network.**
2. **Multilayer feed-forward network.**
3. **Single node with its own feedback.**
4. **Single-layer recurrent network.**
5. **Multilayer recurrent network.**

- **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.

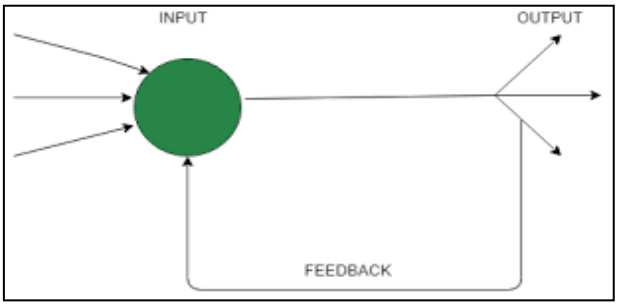
1. ***Single layer feed-forward network*** - The simplest kind of neural network is a single-layer perceptron network, which **consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights**. In this way it can be considered the simplest kind of feed-forward network.



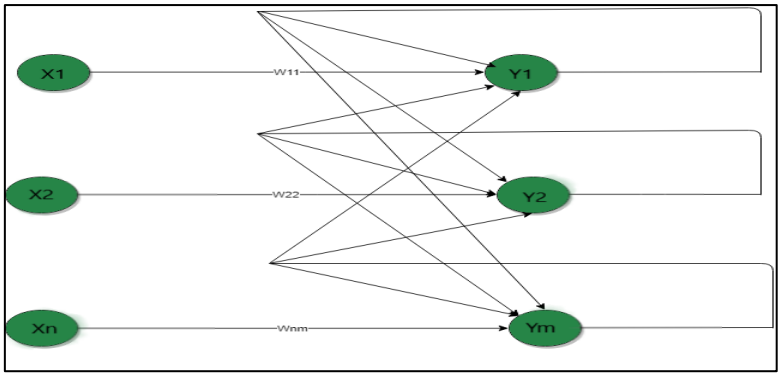
2. ***Multi-layer feed-forward network*** - A multilayer feedforward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons.



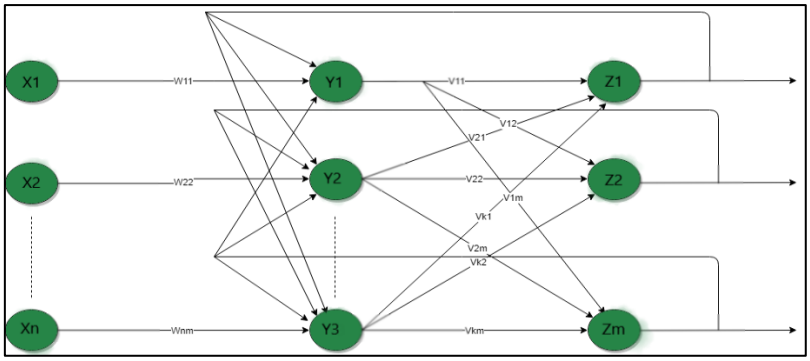
3. Single node with its own feedback - **A network is said to be a feed forward network if no neuron in the output layer is an input to a node in the same layer or in the preceding layer**. When outputs can be directed back as inputs to same or preceding layer nodes then it results in the formation of feedback networks.



4. Single layer recurrent network - A recurrent neural network (RNN) is **a type of artificial neural network which uses sequential data or time series data**. - A single-layer network with a feedback connection in which the processing element's output can be directed back to itself or to another processing element or both. RNNs can use their internal state (memory) to process sequences of inputs.



5. Multi-layer recurrent network -In this type of network, processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network. They perform the same task for every element of a sequence, with the output being dependent on the previous computations. Inputs are not needed at each time step. The main feature of a Recurrent Neural Network is its hidden state, which captures some information about a sequence.



Learning:

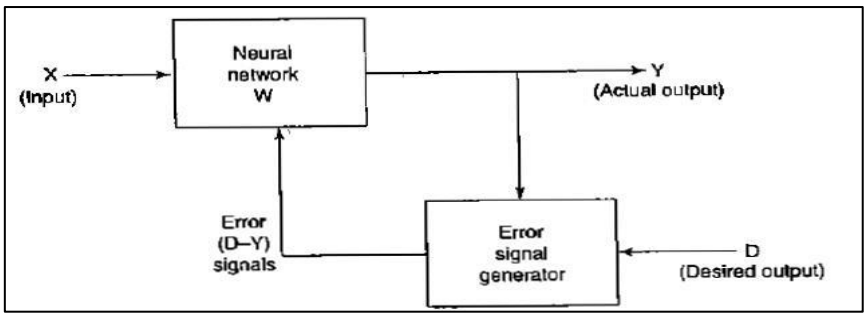
The main property of ANN is capability of learn – learning / Training is a process by means of which a NN adapts itself to a stimulus by making parameter adjustments, resulting in the production of desired response

- Two kinds of learning
  - Parameter learning - It update the connection weights in neural net
  - Structure learning – Focus on the change in the network structure
- General classification of ANN
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

1.Supervised learning:

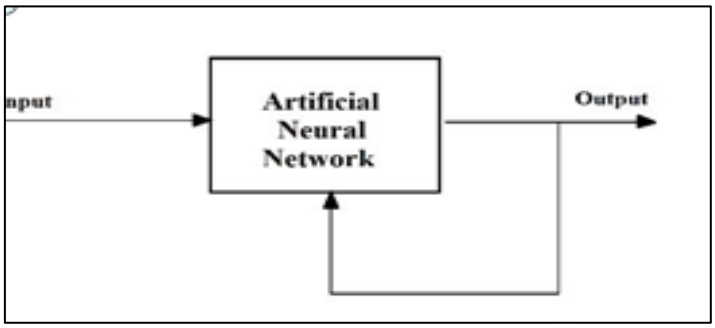
- As the name suggests, **supervised learning** takes place under the supervision of a teacher. This learning process is dependent.
- During the training of ANN under supervised learning, the input vector is presented to the network, which will produce an output vector. This output vector is compared with the desired/target output vector. The input vector along with the target vector is called training pair.

- An error signal is generated if there is a difference between the actual output and the desired/target output vector. On the basis of this error signal, the weights would be adjusted until the actual output is matched with the desired output.



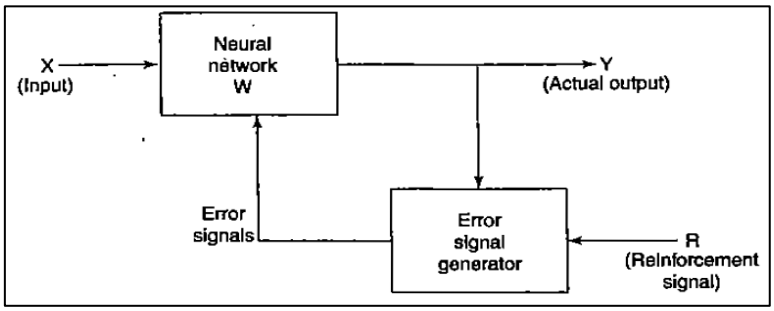
2. Unsupervised learning:

- As the name suggests, this type of learning is done without the supervision of a teacher. This learning process is independent.
- During the training of ANN under unsupervised learning, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which input pattern belongs.
- In this, there would be no feedback from the environment as to what should be the desired output and whether it is correct or incorrect. Hence, in this type of learning the network itself must discover the patterns (Self organizing), features from the input data and the relation for the input data over the output.



3.Reinforcement learning:

- Reinforcement learning is **a goal-directed computational approach where an agent learns to perform a task by interacting with an unknown dynamic environment (learning based on the critic information)**. During training, the learning algorithm updates the agent policy parameters. (Feedback sent is called reinforcement signal)



Activation Functions:

- Bipolar binary and unipolar binary are called as hard limiting activation functions used in discrete neuron model.
- Unipolar continuous and bipolar continuous are called soft limiting activation functions are called sigmoidal characteristics.
- Several activation functions are there:

1. Identity Function
   f(x)=x for all x
2. Binary Step function

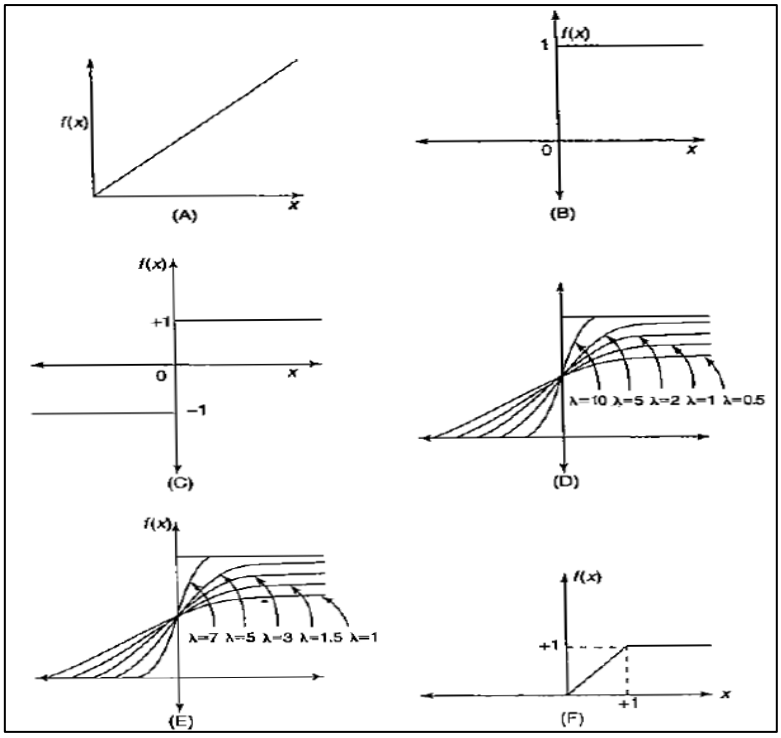
$$f(x) = \begin{cases} 1 if x \geq \theta \\ 0 if x < \theta \end{cases}$$

3. Bipolar Step function

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

4. Sigmoidal Functions:- Continuous functions
5. Ramp functions:-

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



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 functions

4. IMPORT TERMINOLOGIES OF ANN:

- **Weights:**
  1. Each neuron is connected to every other by means of directed links
  2. Links are associated with weights
  3. Weights contain information about the input signal and is represented as a matrix
  4. Weight matrix also called connection matrix

- **Bias:**
  1. Bias is like another weight. It's included by adding a component *x0=1* to the input vector X.

     $$X = (1, X1, X2 \ldots Xi, \ldots Xn)$$

  2. Bias is of two types
     - ☐ Positive bias: increase the net input
     - ☐ Negative bias: decrease the net input

- **Threshold:**
  1. Set value based upon which the final output of the network may be calculated.
  2. Used in activation function.
  3. The activation function using threshold can be defined as

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

- **Learning rate:**
  1. Denoted by α.
  2. Used to control the amount of weight adjustment at each step of training
  3. Learning rate ranging from 0 to 1 determines the rate of learning in each time step

- **Momentum factor:**
  1. used for convergence when momentum factor is added to weight updating process.

- **Vigilance parameter:**
  1. Denoted by ρ
  2. Used to control the degree of similarity required for patterns to be assigned to the same cluster

- **Neuron/Node:**
  1. The fundamental unit of an artificial neural network. It receives inputs, performs a computation, and generates an output.

- **Input Layer:**

  1.The layer of neurons that receives the initial input data for the network.

- **Hidden Layer:**

  The layer(s) of neurons between the input and output layers. It performs intermediate computations and captures complex patterns in the data.

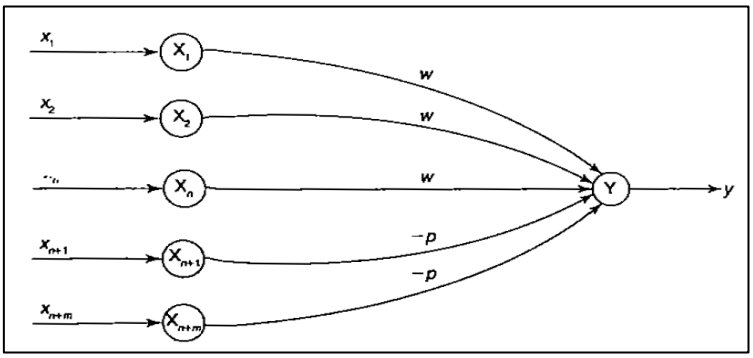
- **Output Layer:**
  The final layer of neurons that produces the network's output or prediction.

- **Forward Propagation:**
  The process of passing the input data through the network layer by layer, computing the outputs of each neuron, until the final output is generated.

- **Backpropagation:**
  The algorithm used to train neural networks by iteratively adjusting the weights and biases based on the calculated error between the predicted output and the actual output.

5. MUCULLOCH AND PITTS NEURON:

- ☐ The McCulloch-Pitts (Walter Pitts) neuron was the earliest neural network discovered in 1943.
- ☐ It is usually called as M-P neurons connected by weighted paths
- ☐ Activation function is binary that is at any time step the neuron may free or may not free
- ☐ Weight may be excitatory (Positive) or Inhibitory (Negative) and all are entering in to the neuron with same weights
- ☐ Threshold plays a vital role, if input is greater that threshold the neuron fires and any non-zero inhibitory prevent the firing
- ☐ Mostly used in Logic functions



McCulloch-Pins neuron model

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

$$f(x)= \{1 \; if \; yin \geq \theta$$
$$0 \; if \; yin < \theta$$

- 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 were

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

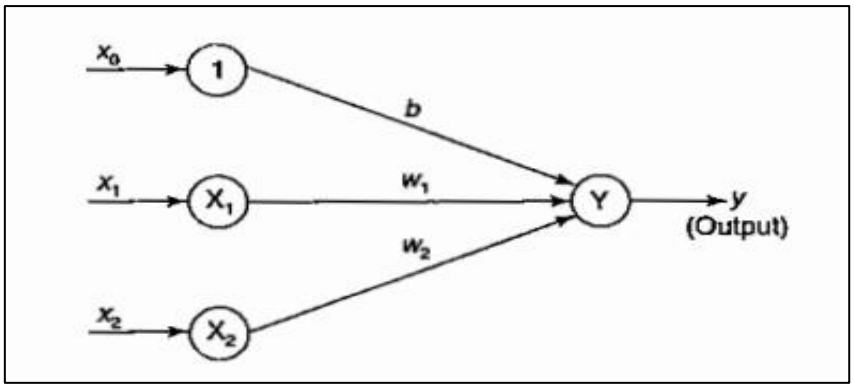
The M-P neuron has no particular training algorithm. An analysis is performed to determine the weights and the threshold. It is used as a building block where any function or phenomenon is modeled based on a logic function.

6. LINEAR SEPARABILITY:

- ANN does not give an exact solution for a nonlinear problem. However, it provides possible approximate solutions nonlinear problems.
- Linear separability, is the concept wherein the separation of the input space into regions is based on whether the network response is positive or negative.
- A decision line is drawn to separate positive and negative responses. The decision line may also be called as the decision-making line or decision-support line or linear-separable line.
- The necessity of the linear separability concept was felt to classify the patterns based upon their output responses. Generally, the net input calculation to the output unit is given as

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

☐ Single layer neural net:

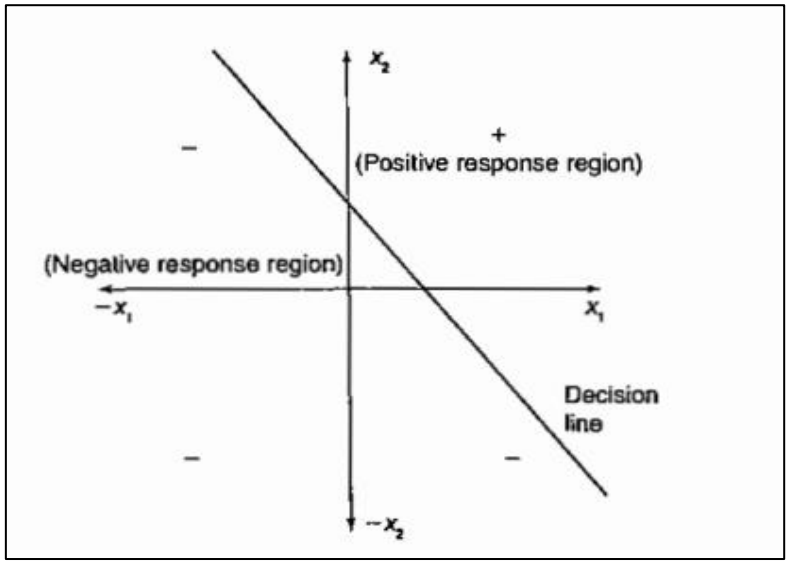


☐ Consider a network having positive response in the first quadrant and negative response in all other quadrants (AND function) with either binary or bipolar data, then the decision line is drawn separating the positive response region from the negative response region.

☐ Separation of the input space into regions is based on whether the network response is positive or negative. Line of separation is called linear separable line.

☐ Example: -

- AND function & OR function is linear separable
- XOR function Linearly inseparable



DECISION BOUNDARY LINE

7. HEBB NETWORK:

- Donald Hebb stated in 1949, learning rule is the simplest one.
- The learning in the brain is performed by the change in the synaptic gap.
- When an axon of cell A is near enough to excite cell B and repeatedly keep firing it, some growth process takes place in one or both cells
- According to Hebb rule, weight vector is found to increase proportionately to the product of the input and learning signal.

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

- The Hebb rule is more suited for data than bipolar data. If binary data is used, the above weight updation formula cannot distinguish two conditions namely;
  - A training pair in which an input unit is "on" and target value is "off. "
  - A training pair in which both the input unit and the target value are "off. "
- Thus, there are limitations in Hebb rule application over binary data. Hence, the representation using bipolar data is advantageous.

TRAINING ALOGRITHM:

- The training algorithm is used for the calculation and of weights.

Step 0: Initialize weights: Set all weights (wi) to zero for i = 1 to n, where n is the total number of input neurons.

Step 1: For each input training vector and target output pair (s:t), perform the following steps.

Step 2: Set input units activations: x = s.

Step 3: Set output units activations: y = t.

Step 4: Adjust weights and biases: Update the weights using the formula:

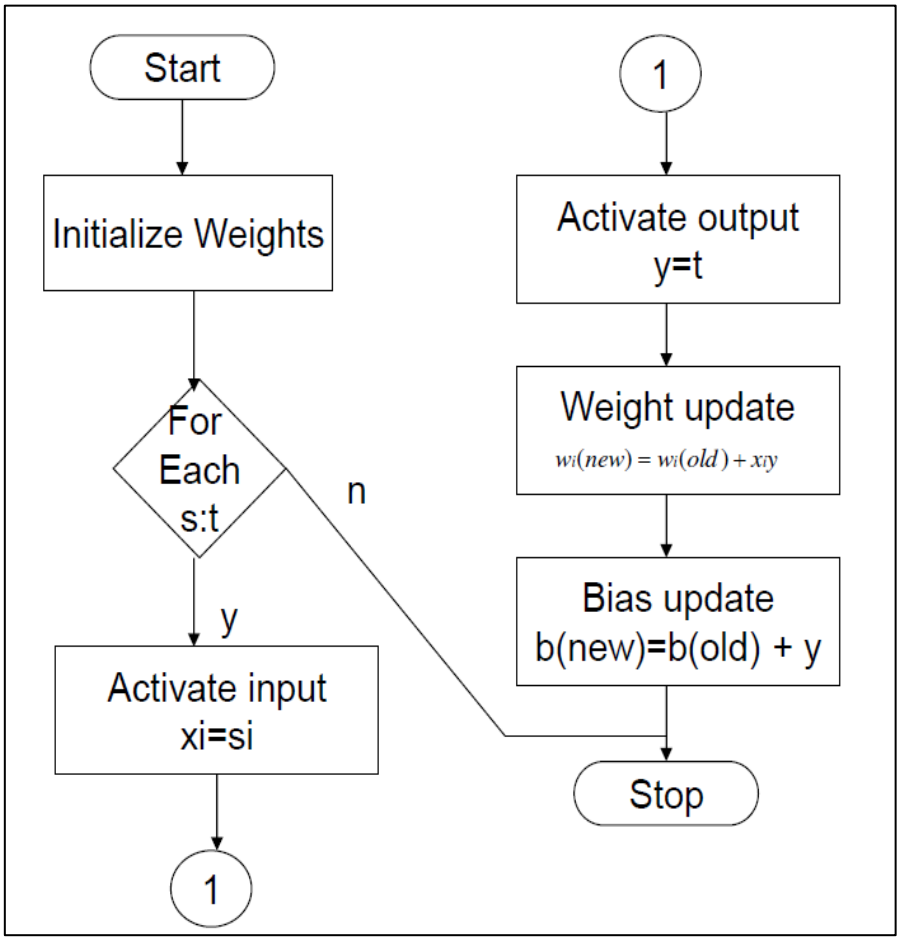
$$\mathbf{wi(new) = wi(old) + x * y,}$$

and update the bias using the formula:

$$\mathbf{b(new) = b(old) + y.}$$

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

FLOWCHART OF TRAINING PROCESS:



UNIT – II

1.PERCEPTRON NETWORKS:

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 -l, 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 l, 0 or -1. The binary step with fixed threshold ɵ 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

$$y=f(yin)$$

where $f(yin)$ is activation function and is defined as

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

6. The perceptron learning rule is used in the weight updating 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 ca1culated outputs.

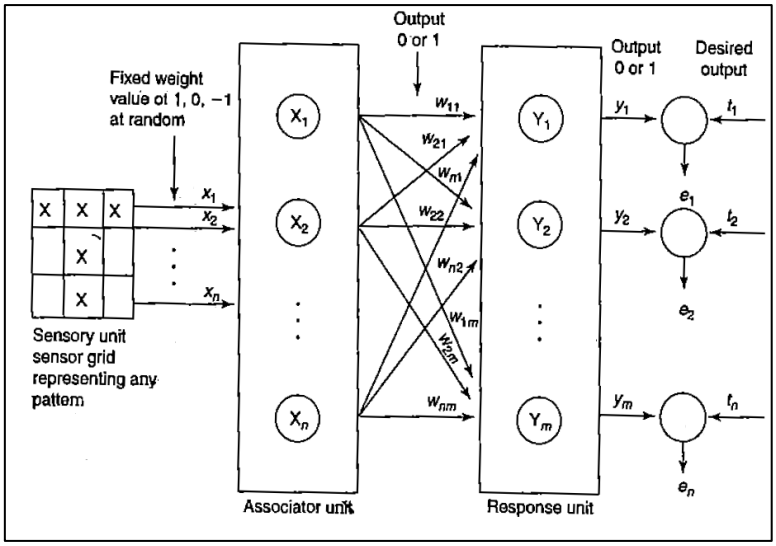
8. The weights on the connections from the units that send the nonzero signal will get adjusted suitably.

9. The weights will be adjusted on the basis of the learning rule an error has occurred for a particular training patterns .i.e..,

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

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

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-l 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.



Original perceptron network

The perceptron network consists of three units: sensory, associator, and response. The sensory unit is a matrix of photodetectors that detect a binary signal based on a threshold. The associator unit has feature predicates that detect specific features of the pattern. The results from the predicates are binary. The response unit contains pattern recognizers with trainable weights. The input layer weights are fixed, while the response unit weights can be adjusted.

Perceptron Learning Rule:

- **Links** − It would have a set of connection links, which carries a weight including a bias always having weight 1.
- **Adder** − It adds the input after they are multiplied with their respective weights.
- **Activation function** − It limits the output of neuron. The most basic activation function is a Heaviside step function that has two possible outputs. This function returns 1, if the input is positive, and 0 for any negative input.

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

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

If $y \neq t$**, then**

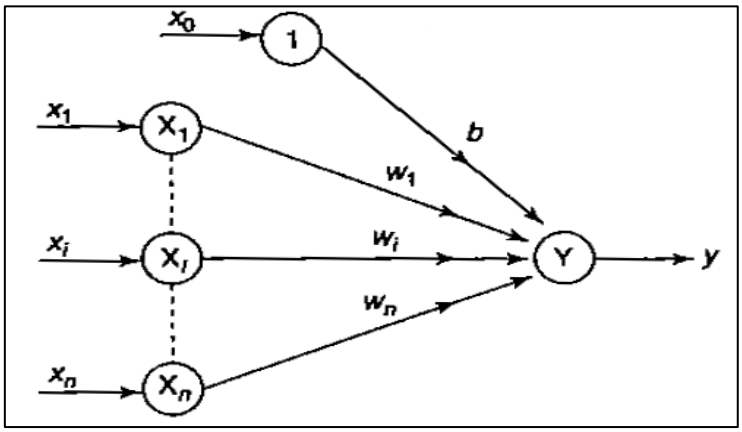
$$w(new) = w(old) + \alpha\, tx\ (\alpha - learning\ rate)$$

else,

$$w(new) = w(old)$$

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 w1, 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".

Architecture:



Single classification perceptron network

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

Flowchart for Training Process:

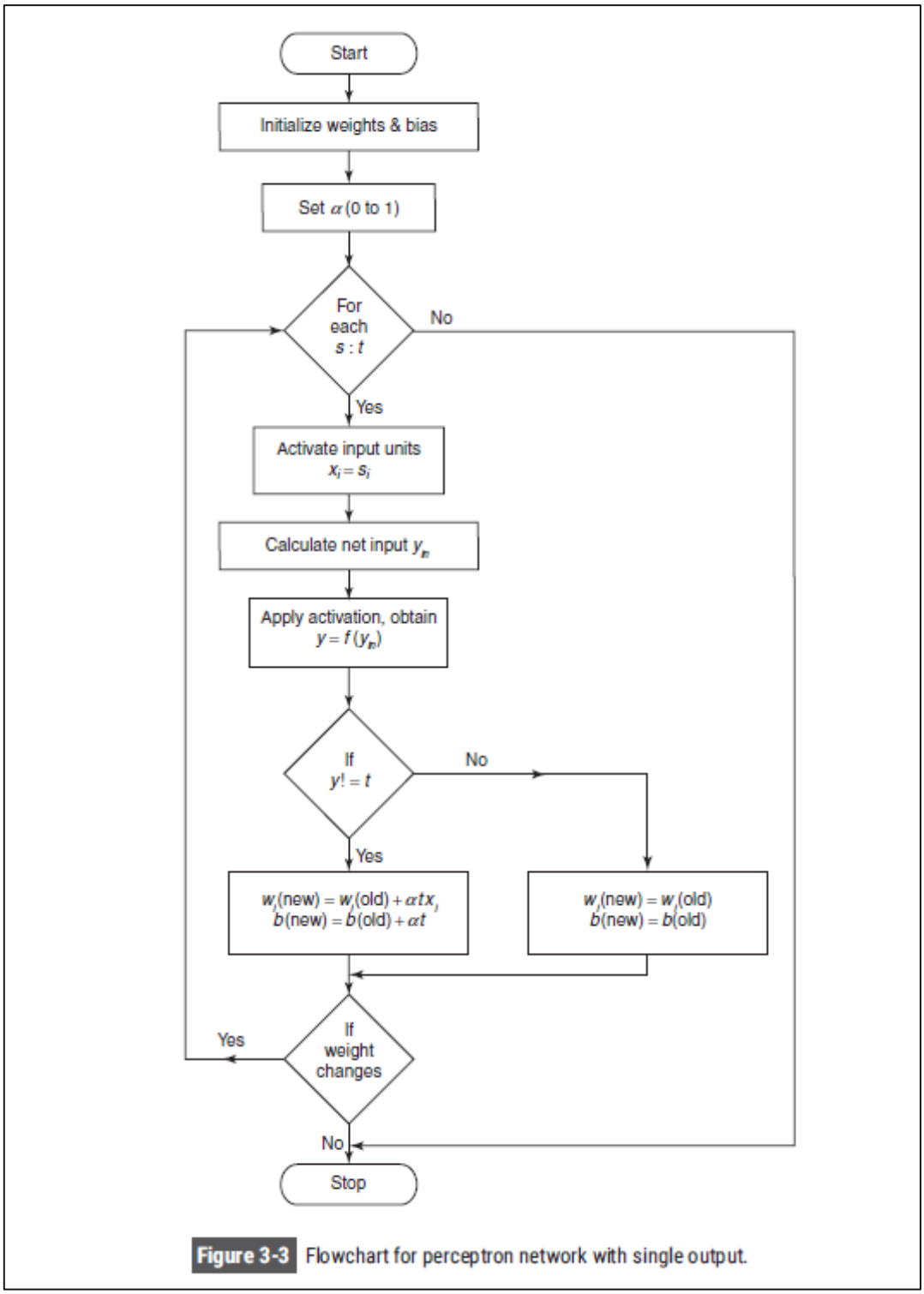


Figure 3-3 Flowchart for perceptron network with single output.

Flowchart for perceptron network with single output

Perceptron Training Algorithm for Single Output Classes:

Step 1: Initialize the weights (w) and bias (b) to small random values or zeros.

Step 2: For each input training sample (x) and its associated target output (t), do the following steps:

a. Calculate the net input (z) by taking the dot product of the input features (x) and weights (w), and adding the bias (b):

$$z = dot(x, w) + b.$$

b. Apply the activation function (typically a step function) to the net input to obtain the predicted output (y): **y = activation_function(z).**

c. Compute the error (e) as the difference between the target output (t) and the predicted output (y): **e = t - y.**

d. Update the weights and bias based on the error and learning rate (α):

Update the weights: $w = w + \alpha * e * x.$

Update the bias: $b = b + \alpha * e.$

Step 3: Repeat steps 2 until the desired level of accuracy is achieved or a maximum number of iterations is reached.

The algorithm updates the weights and bias in the direction that reduces the error, gradually improving the perceptron's ability to classify the input samples correctly.

Perceptron Network Testing Algorithm

Step 1: Given a trained perceptron network with weights (w) and bias (b).

Step 2: For each input test sample (x), calculate the net input (z) as the dot product of the input features (x) and weights (w), plus the bias (b).

Step 3: Apply the activation function to the net input to obtain the predicted output (y).

Step 4: Compare the predicted output (y) to a threshold value (e.g., 0.5) to determine the classification result. If y >= threshold, assign the positive class label; otherwise, assign the negative class label.

Step 5: Repeat for each test sample to obtain the classification results.

This algorithm evaluates the trained perceptron network on new test samples by calculating the net input, applying the activation function, and making a classification decision based on a threshold value.

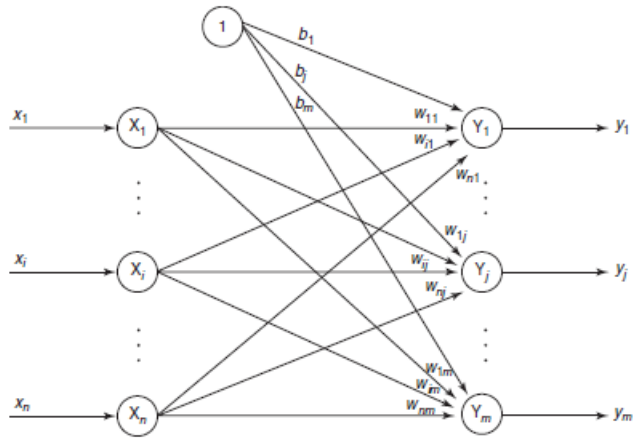


Figure 3-4 Network architecture for perceptron network for several output classes.

2. ADAPTIVE LINER NEURON:

Adaline (Adaptive Linear Neuron) is a network with a single linear unit. It uses a linear activation function, and the input-output relationship is also linear. Adaline employs adjustable weights and a bias, which acts as an adjustable weight connected to a unit with a constant activation of 1. The target output and input signals in Adaline are bipolar.

The training of Adaline is done using the delta rule, also known as the least mean square (LMS) or Widrow-Hoff rule. The delta rule minimizes the mean squared error between the predicted activation and the target value.

Delta Rule for Single Output Unit

Perceptron Learning Rule:

- Originates from Hebbian assumption.
- Stops after a finite number of learning steps.
- Updates weights based on a binary classification error.
- Aim is to find a linear decision boundary.

Delta Rule:

- Derived from the gradient descent method.
- Can be generalized to more than one layer.
- Continues indefinitely, converging asymptotically.
- Updates weights to minimize the difference between net input and target value.
- Aims to minimize error over all training patterns by reducing error for each pattern, one at a time.

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

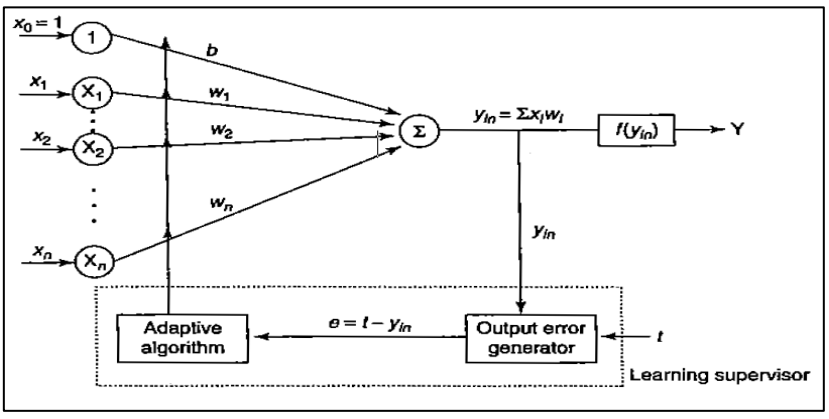
$$\Delta wi = \alpha(t - yin)xi$$

Where Δwi is the weight change; α the learning rate; x the vector of activation of input unit; yin the net input to output unit; 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

$$\Delta wij = \alpha(tj - yinj)xi$$

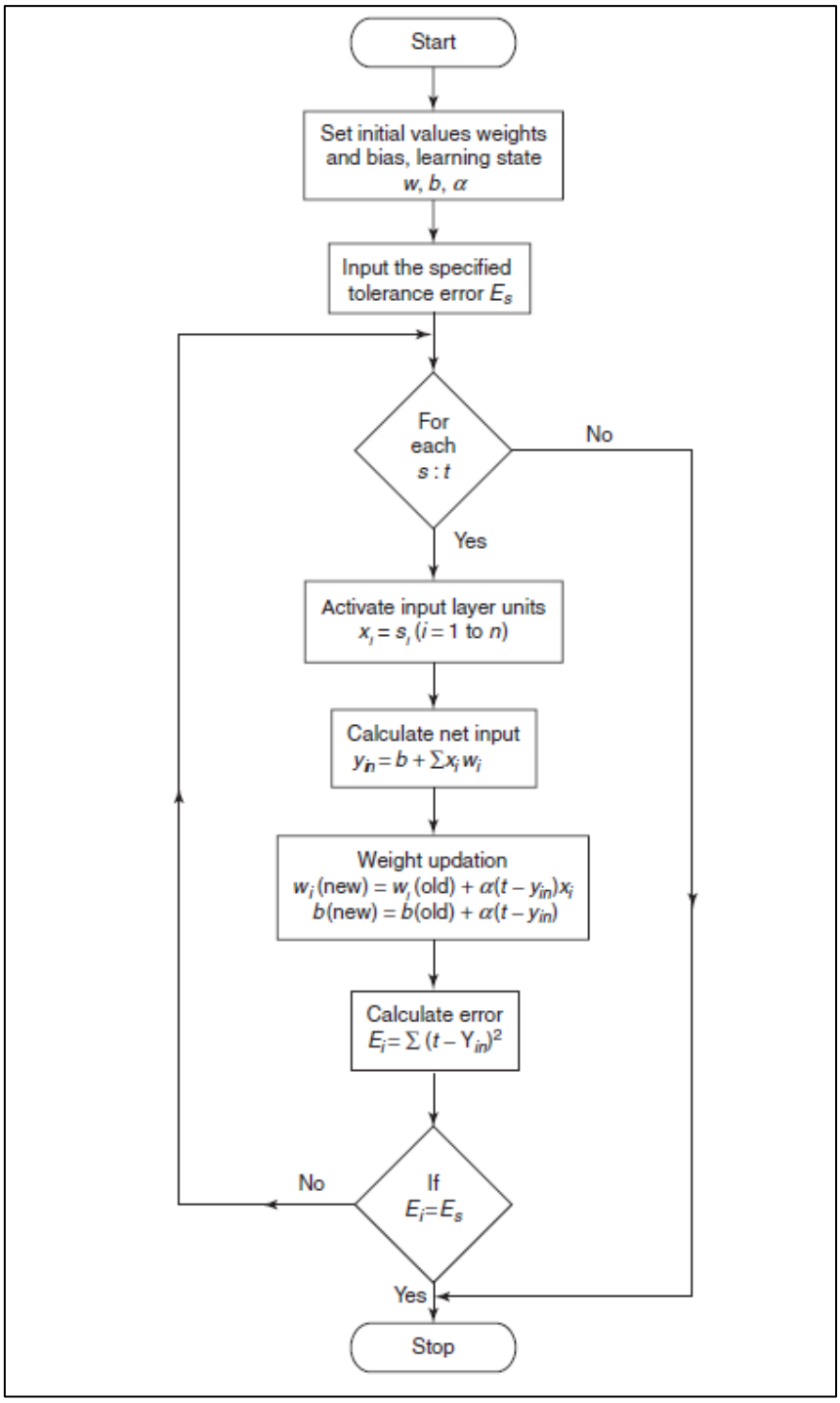
Architecture:

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.



Adaline model

Flowchart for Training Process:



Training Algorithm

Step 1: Set the learning rate (α) to a small positive value.

Step 2: Repeat the following steps until convergence or a maximum number of iterations:

a. For each input training sample (x) and its associated target output (t), do the following:

i. Calculate the net input (z) by taking the dot product of the input features (x) and weights (w).

ii. Compute the error (e) as the difference between the target output (t) and the net input (z).

iii. Update the weights using the formula: $w = w + \alpha * e * x$.

Step 3: End.

The training algorithm for Adaptive Linear Neurons adjusts the weights iteratively to minimize the error between the predicted output and the target output. It calculates the net input, computes the error, and updates the weights by multiplying the error with the learning rate and the input features. This process is repeated until convergence or a maximum number of iterations is reached.

Note: The Adaptive Linear Neurons algorithm is a simplified version of the Widrow-Hoff or Delta rule and is suitable for linear regression or binary classification tasks. For more complex tasks or nonlinear data, other algorithms or network architectures may be more appropriate.

Testing algorithm:

Step 1: Given a trained Adaptive Linear Neuron with weights (w).

Step 2: For each input test sample (x), do the following steps:

a. Calculate the net input (z) by taking the dot product of the input features (x) and weights (w).

b. Obtain the predicted output (y) by passing the net input (z) through an activation function (if applicable).

c. Compare the predicted output (y) to a threshold value (if applicable) or apply any decision rule to determine the final classification or regression result.

d. Store or output the predicted output or classification result.

Step 3: Repeat step 2 for each test sample to obtain the predictions for all input samples.

The testing algorithm for Adaptive Linear Neurons evaluates the trained neuron on new, unseen test samples. It calculates the net input, applies an activation function (if applicable), makes a decision based on a threshold value or decision rule, and stores or outputs the predicted output or classification result.

3.MULTIPLE ADAPTIVE LINEAR NEURON (MADALINE)

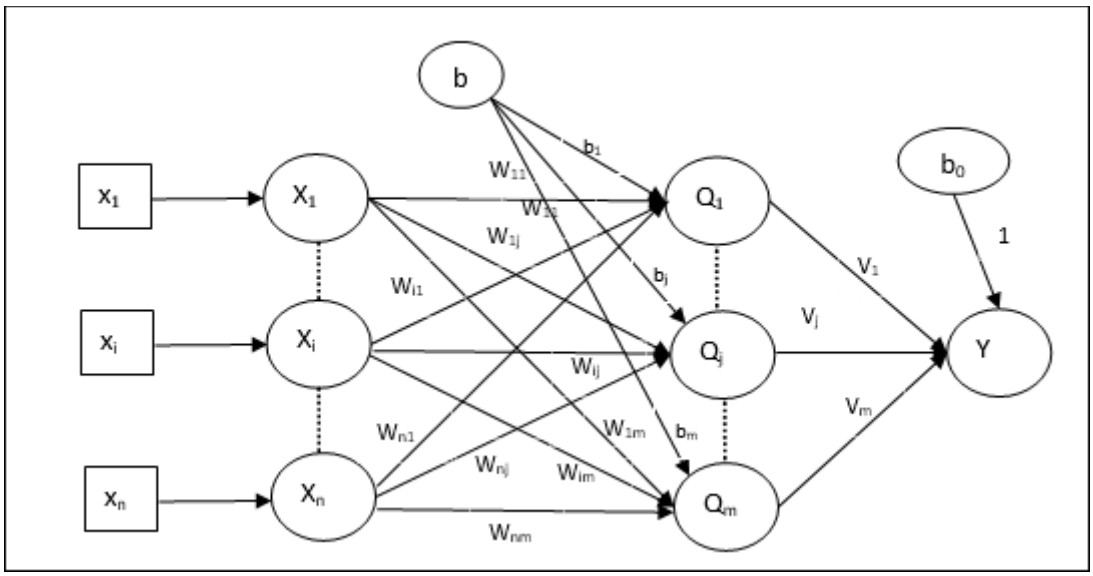
Madaline which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Some important points about Madaline are as follows

- It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.
- The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.
- The Adaline and Madaline layers have fixed weights and bias of 1.
- Training can be done with the help of Delta rule.

Architecture

The architecture of Madaline consists of **"n"** neurons of the input layer, **"m"** neurons of the Adaline layer, and 1 neuron of the Madaline layer.

The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.



TRANING ALGORITHM:

Step 0: Initialize weights, learning rate, and small random values for Adaline weights.

Step 1: Repeat steps 2-3 until the stopping condition is met.

Step 2: For each training pair (s, t), perform steps 3-7.

Step 3: Activate the input layer units.

Step 4: Calculate the net input for each hidden Adaline unit.

Step 5: Calculate the output of each hidden unit.

Step 6: Calculate the net input of the output unit and find the output of the net.

Step 7: Calculate the error and update the weights.

- If the target (t) is equal to the output (y), no weight update is required.
- If the target is not equal to the output:
  1. If the target is +1, update weights on the hidden unit (z) with the net input closest to 0.
  2. If the target is -1, update weights on units (z) with positive net inputs.

Step 8: Test the stopping condition.

- Check if there is no weight change or if weights reach a satisfactory level.
- Check if a specified maximum number of iterations has been reached.

Flowchart of Training Process:

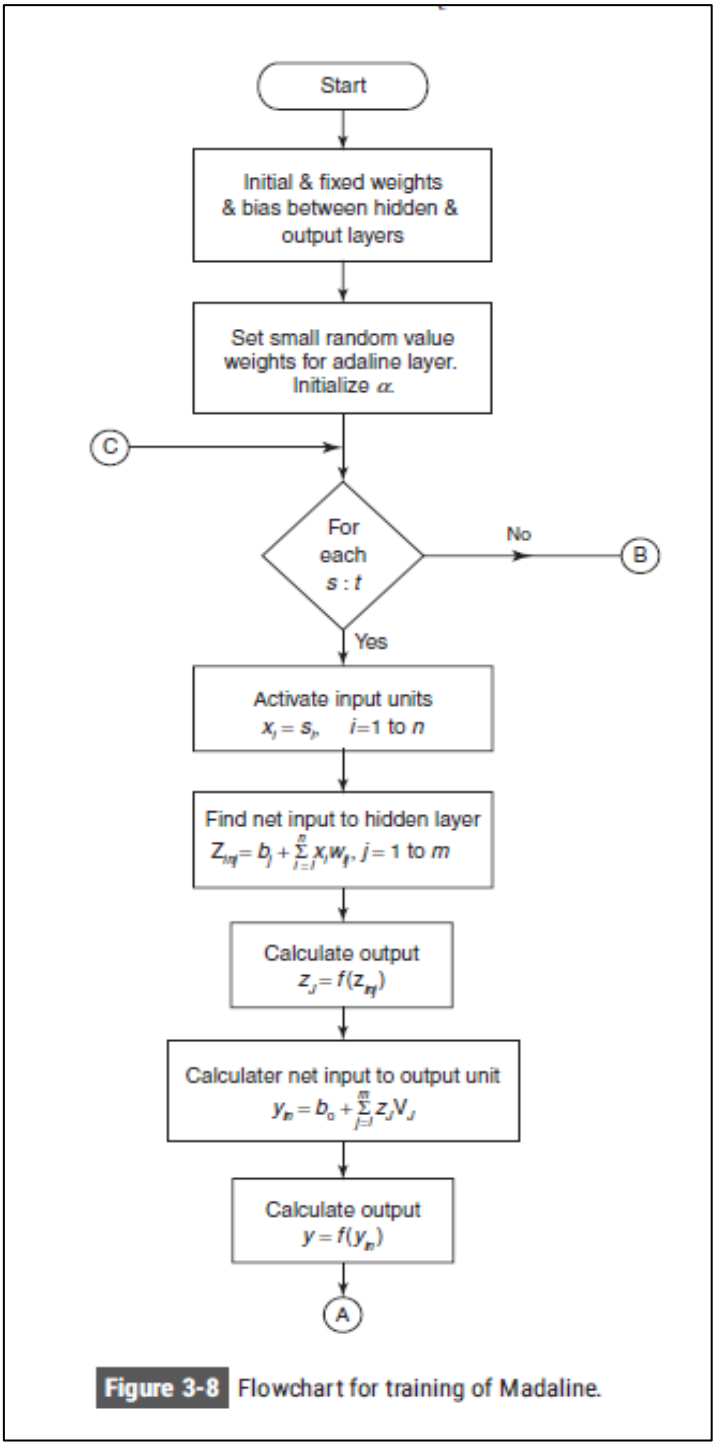


Figure 3-8 Flowchart for training of Madaline.

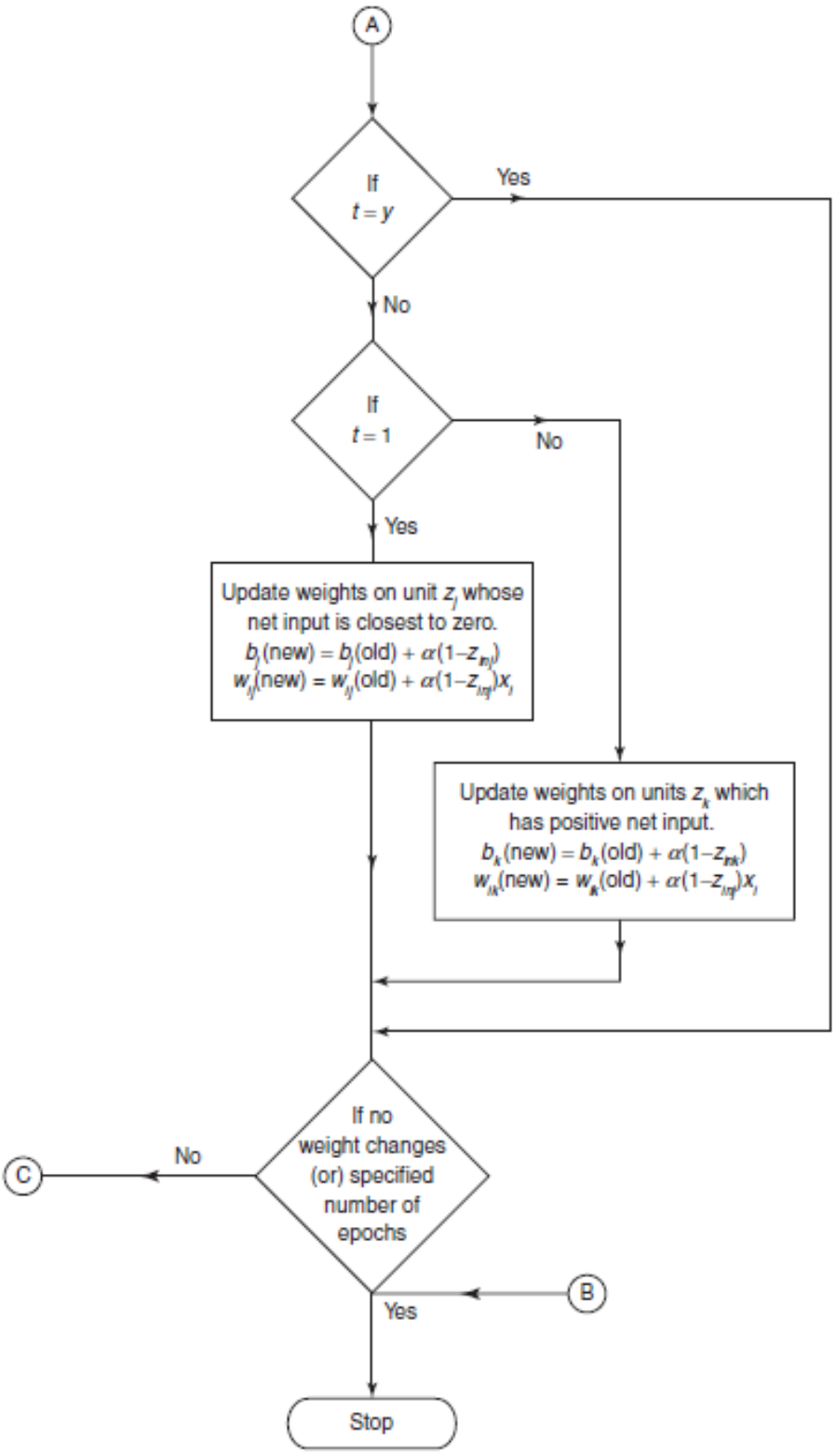


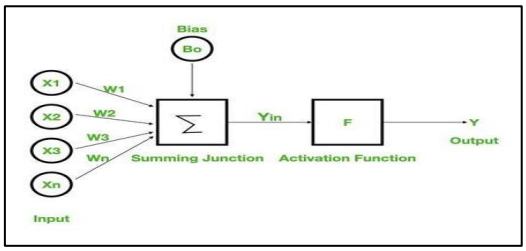
Figure 3-8 (Continued).

3. BACKPROPAGATION

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

Neural Network:

Neural networks are an information processing paradigm inspired by the human nervous system. Just like in the human nervous system, we have biological neurons in the same way in neural networks we have artificial neurons, artificial neurons are mathematical functions derived from biological neurons. The human brain is estimated to have about 10 billion neurons, each connected to an average of 10,000 other neurons. Each neuron receives a signal through a synapse, which controls the effect of the sign concerning on the neuron.



Why Backpropagation:

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Features of Backpropagation:

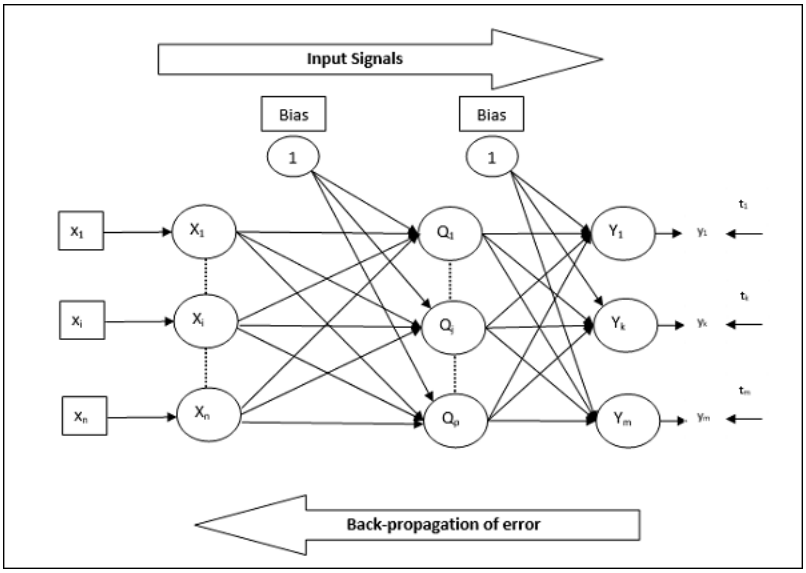
1.it is the gradient descent method as used in the case of simple perceptron network with the differentiable unit.

2. it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
3. training is done in the three stages:
   - the feed-forward of input training pattern
   - the calculation and backpropagation of the error
   - updation of the weight

Working of Backpropagation:

Neural networks use supervised learning to generate output vectors from input vectors that the network operates on. It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get your desired output.

Architecture:



BACKPROPAGATION ALGORITHM:

Step 1: Inputs X, arrive through the preconnected path.

Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.

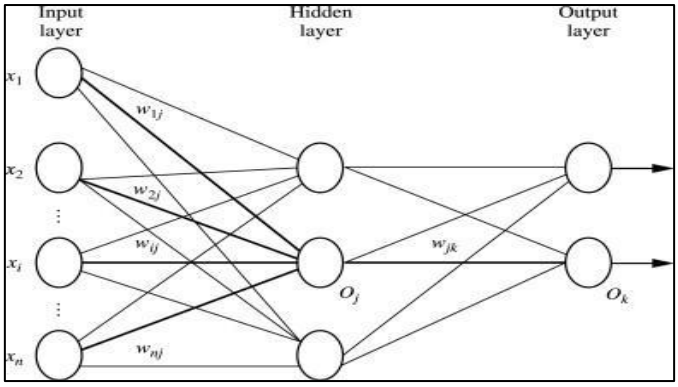
Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Step 4: Calculate the error in the outputs

Backpropagation Error= Actual Output – Desired Output

Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6: Repeat the process until the desired output is achieved.



Training Algorithm:

1. Initialize weights with small random values.
2. Repeat steps 3 to 10 until the stopping condition is met.
3. Perform feed-forward propagation for each training pair.
4. Calculate the net input for each hidden unit by summing weighted input signals and applying an activation function.
5. Calculate the net input for each output unit by summing weighted input signals from the hidden units and applying an activation function.
6. Calculate the error for each output unit by comparing the actual output with the target output.
7. Calculate the error information for each hidden unit by summing the error contributions from the units in the layer above.

8. Update the weights and biases for each output unit based on the error and input signals.
9. Update the weights and biases for each hidden unit based on the error information and input signals.
10. Test the stopping condition, such as reaching a desired error level or a specified number of epochs.

Backpropagation is a powerful training technique that allows for the adjustment of weights in neural networks. It enables the network to learn and improve its performance without prior knowledge of the network structure.

Types of Backpropagations

There are two types of backpropagation networks.

- **Static backpropagation:** Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
- **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

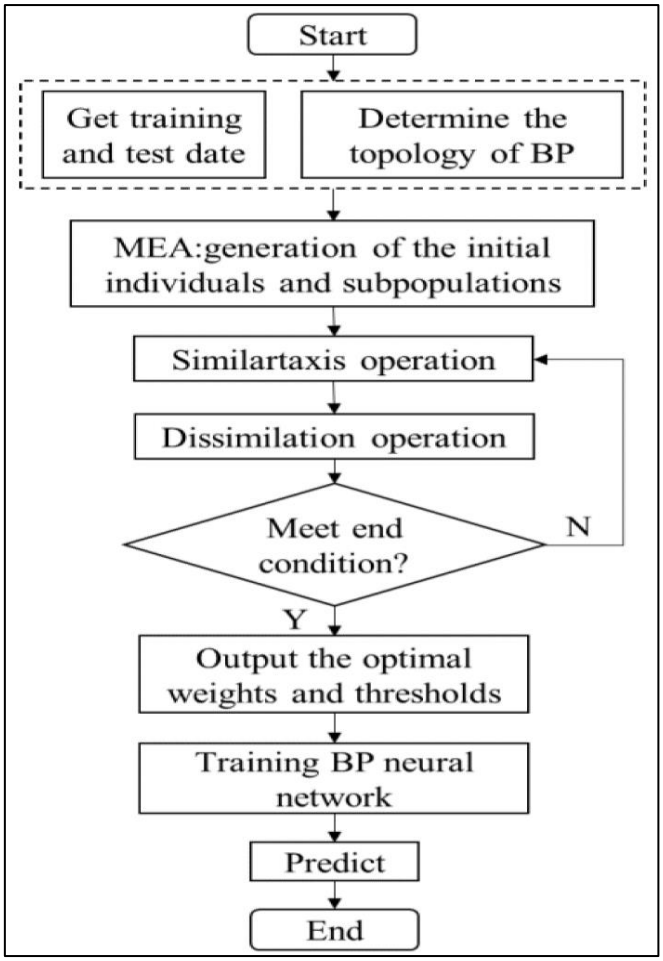
Advantages:

- It is simple, fast, and easy to program.
- Only numbers of the input are tuned, not any other parameter.
- It is Flexible and efficient.
- No need for users to learn any special functions.

Disadvantages:

- It is sensitive to noisy data and irregularities. Noisy data can lead to inaccurate results.
- Performance is highly dependent on input data.
- Spending too much time training.
- The matrix-based approach is preferred over a mini-batch.

FLOW CHART:



Learning Factors of a Backpropagation Network:

1. Initial Weights:

- Randomly initialize the weights at small values.
- Avoid high initial weights to prevent early saturation and convergence to local minima.
- Consider using Nyugen-Widrow initialization for faster convergence.

2. Learning Rate (a):

- Determines the step size of weight updates during training.
- A larger learning rate can speed up convergence but may result in overshooting.
- A smaller learning rate leads to slower learning.
- Commonly used range: 10^ (-3) to 10.

3. Momentum Factor:

- Helps avoid oscillations and allows for larger learning rates.
- Value of 0.9 is often used for the momentum factor.
- Can be used with pattern-by-pattern or batch-mode weight updating.

4. Generalization:

- Backpropagation networks are known for good generalization.
- Monitor error on a test set and terminate training when error increases (to avoid overfitting).
- Introduce variations in the input space of training patterns to improve generalization.

5. Number of Training Data:

- Sufficient and proper training data is essential.
- Training data should cover the entire input space.
- Randomly select training patterns during training.
- Rule of thumb: Number of training patterns (T) should be T / L1, where L is the number of disjoint regions in the input space.

6. Number of Hidden Layer Nodes:

- Experimentally determine the appropriate number of hidden units.
- Size of hidden layer is usually a relatively small fraction of the input layer.
- Increase hidden nodes if the network doesn't converge; decrease if it converges too quickly.

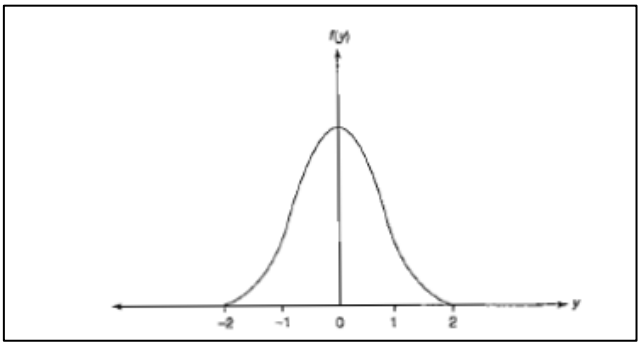
These learning factors play a crucial role in the convergence, performance, and generalization abilities of a backpropagation network.

4. RADIAL BASIS FUNCTION NETWORK:

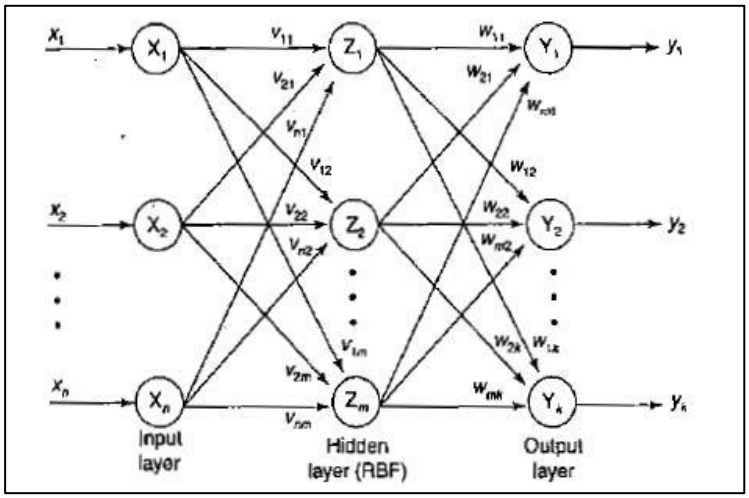
The Powell network utilizes common nonlinearities like sigmoidal and Gaussian kernel functions. The Gaussian function is often used in regularization networks and has a positive response for all y values, decreasing to 0 as |y| approaches 0. The derivative of the Gaussian function is -2ye^(-y) = -2yf(y)^2. The Gaussian potential functions in the network create identical outputs within a fixed radial distance from the kernel center, resulting in radial symmetry. The network combines these nonlinear basis functions through a linear combination.

Architecture

The architecture for the radial basis function network (RBFN) is shown in Figure 3-12. The architecture consists of two layers whose output nodes form a linear combination of the kernel (or basis) functions computed by means of the RBF



Gaussion Potential



RBFN

Flowchart for Training Process

The flowchart for the training process of the RBF is shown in Figure below. In this case, the center of the RBF functions has to be chosen and hence, based on all parameters, the output of network is calculated.

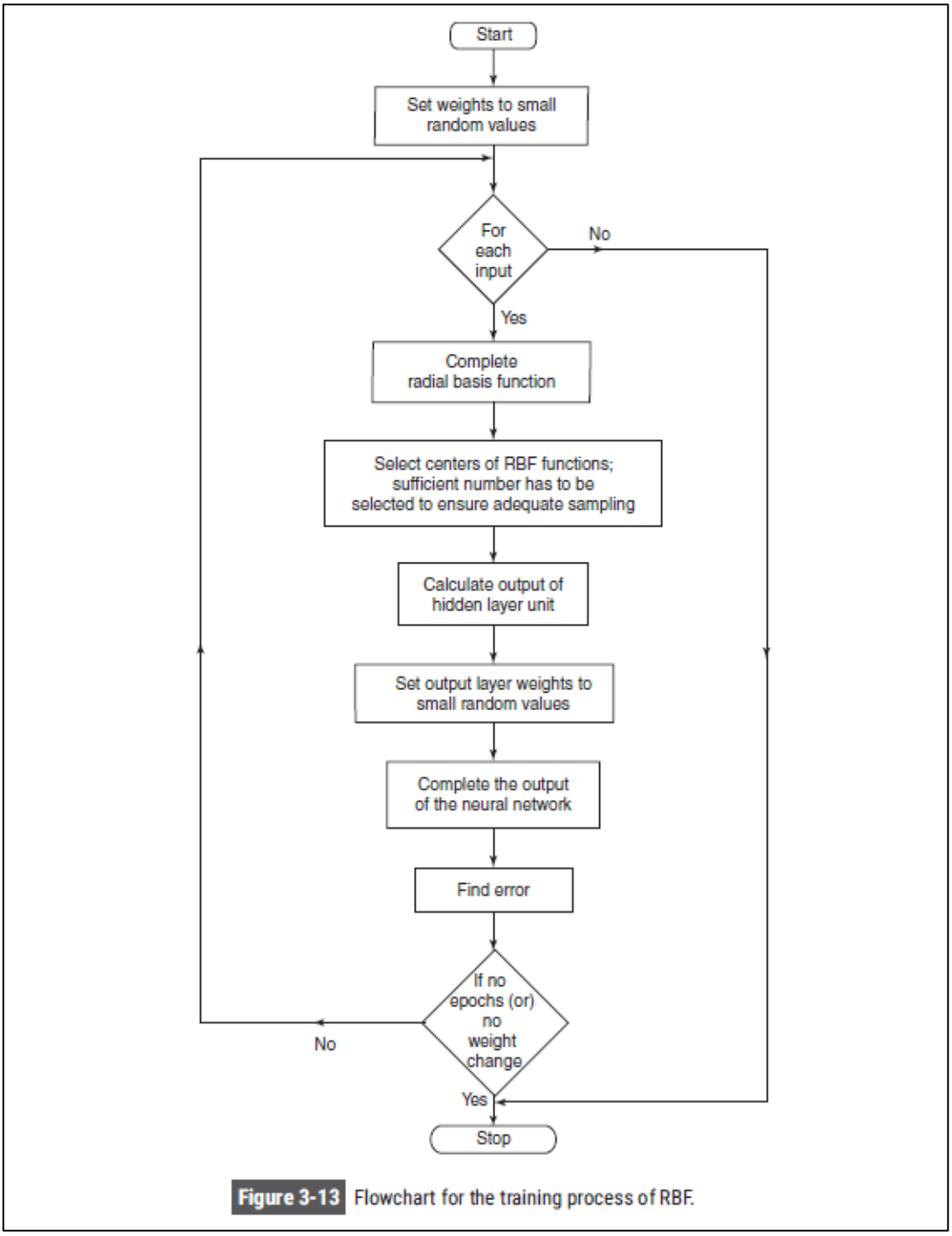


Figure 3-13 Flowchart for the training process of RBF.

Training Algorithm

1. Initialize centers and widths of RBF units.
2. Calculate RBF unit activations based on input pattern's features and unit's center.
3. Solve linear system of equations using RBF unit activations and desired output values.
4. Update weights of the output layer using calculated weights from the linear system.
5. Repeat steps 2 to 4 for all training patterns until convergence criterion is met.
6. Test the trained RBF network on unseen test patterns for evaluation.

UNIT – III

1. ASSOICATIVE MEMORY NETWORK:

These kinds of neural networks work on the basis of pattern association, which means they can store different patterns and at the time of giving an output they can produce one of the stored patterns by matching them with the given input pattern. These types of memories are also called **Content-Addressable Memory** (CAM). Associative memory makes a parallel search with the stored patterns as data files.

Following are the two types of associative memories we can observe −

- Auto Associative Memory
- Hetero Associative memory

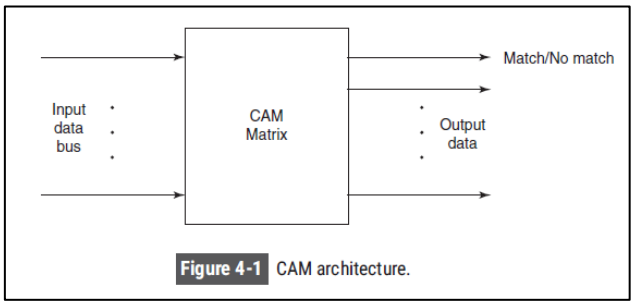
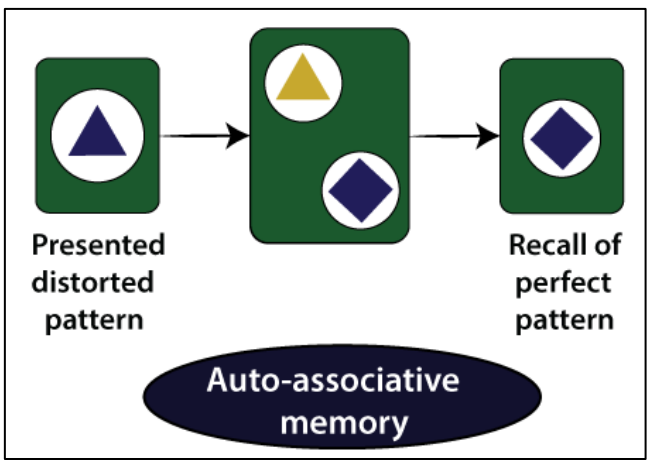


Figure 4-1 CAM architecture.

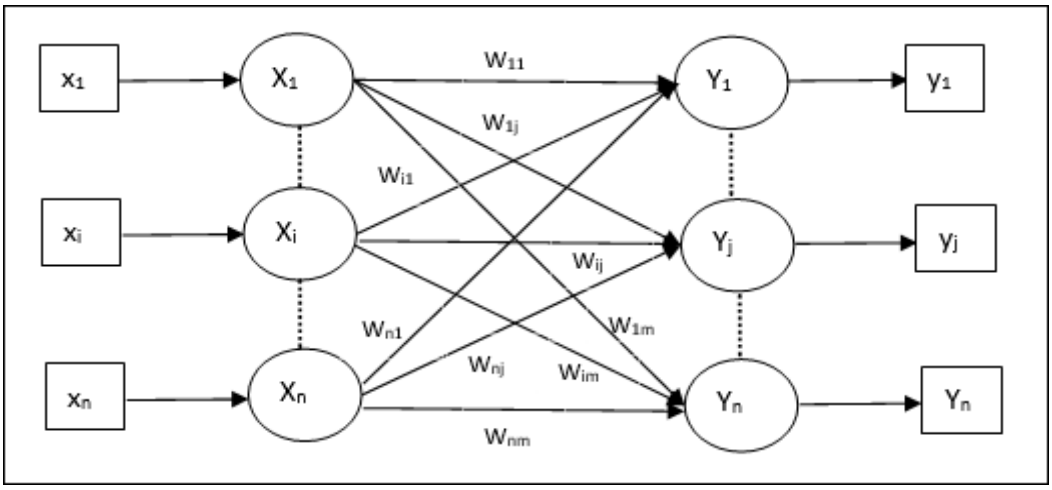
2. AUTOASSOCIATIVE MEMORY NETWORK:

This is a single layer neural network in which the input training vector and the output target vectors are the same. The weights are determined so that the network stores a set of patterns.



Architecture

As shown in the following figure, the architecture of Auto Associative memory network has **'n'** number of input training vectors and similar **'n'** number of output target vectors.



Flow Chart for Training Process:

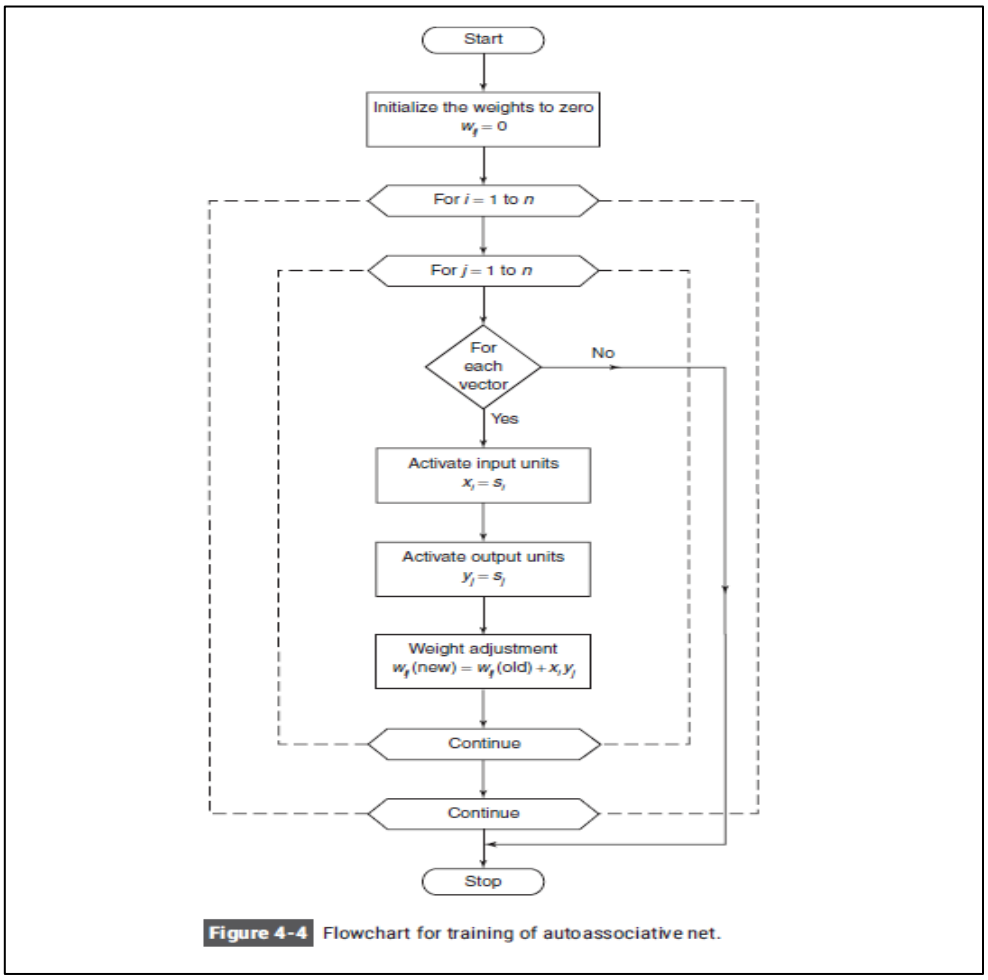


Figure 4-4 Flowchart for training of autoassociative net.

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $w_{ij} = 0, i=1 \text{ to } n, j=1 \text{ to } n$

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows

$$x_i=s_i(i=1 \text{ to } n)$$

Step 4 − Activate each output unit as follows

$$y_j=s_j(j=1 \text{ to } n)$$

Step 5 − Adjust the weights as follows

$$w_{ij}(new)=w_{ij}(old)+x_iy_j$$

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to n**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if\ y_{inj} > 0 \\ -1 & if\ y_{inj} \leqslant 0 \end{cases}$$

3. BIDIRECTIONAL ASSOCIATIVE MEMORY:

Bidirectional Associative Memory (BAM) is a supervised learning model in Artificial Neural Network. This is hetero-associative memory, for an input pattern, it returns another pattern which is potentially of a different size. This phenomenon is very similar to the human brain. Human memory is necessarily associative. It uses a chain of mental associations to recover a lost memory like associations of faces with names, in exam questions with answers, etc.

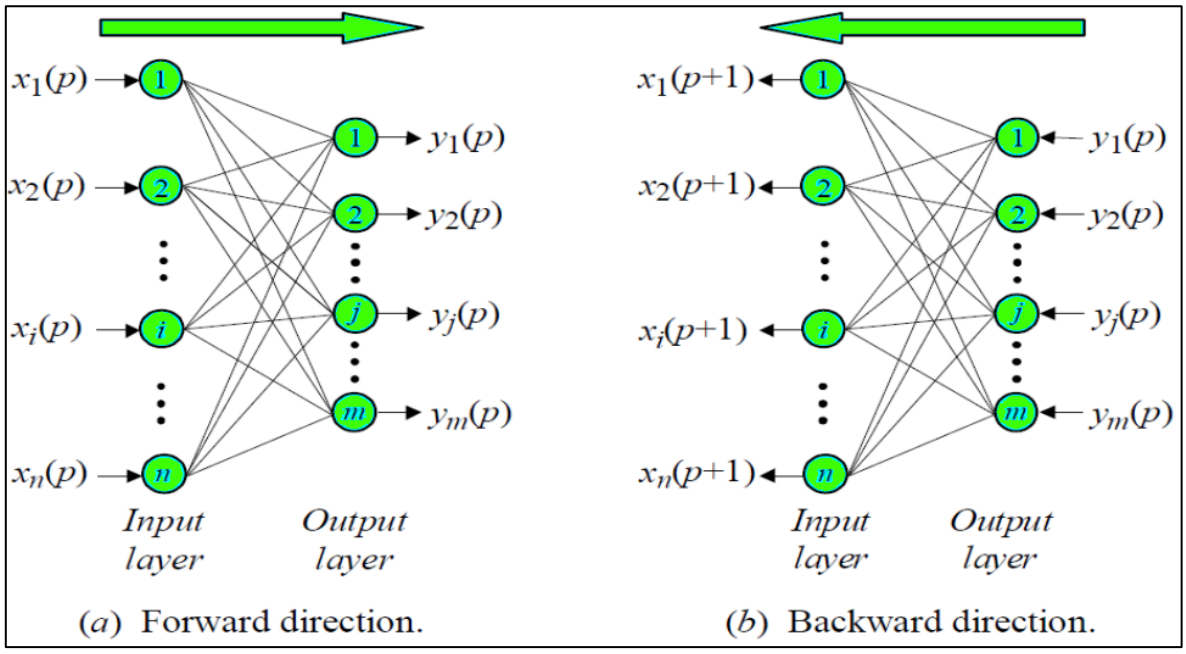
In such memory associations for one type of object with another, a Recurrent Neural Network (RNN) is needed to receive a pattern of one set of neurons as an input and generate a related, but different, output pattern of another set of neurons.

Why BAM is required?

The main objective to introduce such a network model is to store hetero-associative pattern pairs. This is used to retrieve a pattern given a noisy or incomplete pattern.

BAM Architecture:

When BAM accepts an input of n-dimensional vector X from set A then the model recalls m-dimensional vector Y from set B. Similarly, when Y is treated as input, the BAM recalls X.



4. ITERATIVE AUTOASSOCIATIVE MEMORY:

There exists a situation where the net does not respond to the input signal immediately with a stored target pattern but the response may be more like the stored pattern, which suggests using the first response as input to the net again. The iterative autoassociative net should be able to recover an original stored vector when presented with a test vector close to it. These types of networks can also be called as *recurrent autoassociative networks* and Hopfield networks.

Linear Auto associative Memory (LAM):

1. In 1977, James Anderson developed the Linear Auto Associative Memory (LAM) using the Hebbian rule for strengthening connections between neuron-like elements.
2. The LAM uses linear algebra and a non-singular symmetric matrix of size m×m with mutually orthogonal eigenvectors for performance analysis.
3. During training, the LAM uses a set of P orthogonal unit vectors (u1, u2, ..., uP) that can be presented different numbers of times.
4. The weight matrix of the network is determined using the Hebbian learning rule, which may result in some stored vectors being repeated. Each stored vector corresponds to an eigenvector, and the eigenvalues represent the number of times each vector was presented.
5. When an input vector X is presented, the network produces an output response of XW, where W is the weight matrix. The response is highest when X corresponds to the eigenvector with the largest eigenvalue.
6. The recurrent linear autoassociator aims to produce a response that closely resembles the stored vector associated with the input vector. This may require multiple iterations.
7. The input pattern can be represented as a linear combination of vectors. The response of the network to an input vector is a linear combination of its corresponding eigenvalues, and the eigenvector with the largest value is the most similar to the input vector.
8. The network enhances its response for components of the extensively trained input pattern, but the overall output response may grow without bound.

9. To maintain linearity between associative memories, the input and output vector pairs should be mutually orthogonal. Normalizing all input vectors to unit length helps achieve the desired output recall.

5. KOHONEN SELF ORGANIZING FEATURE MAP:

Suppose we have some pattern of arbitrary dimensions; however, we need them in one dimension or two dimensions. Then the process of feature mapping would be very useful to convert the wide pattern space into a typical feature space. Now, the question arises why do we require self-organizing feature map? The reason is, along with the capability to convert the arbitrary dimensions into 1-D or 2-D, it must also have the ability to preserve the neighbor topology.

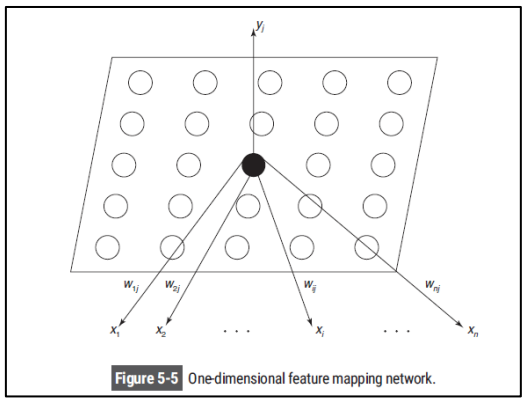


Figure 5-5 One-dimensional feature mapping network.

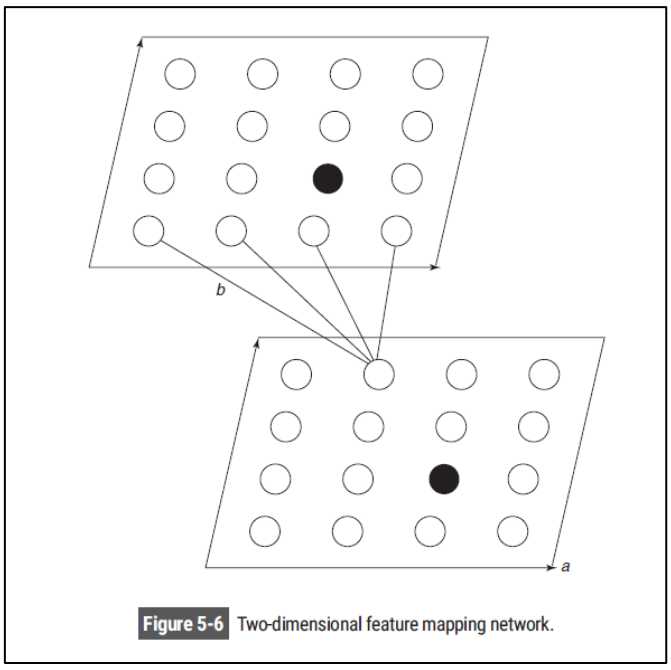


Figure 5-6 Two-dimensional feature mapping network.

During self-organization, the winner unit is chosen based on the closeness of its weight vector to the input pattern. This closeness is often measured using the square of the minimum Euclidean distance.

The weight vectors of the winning unit and its neighboring units are updated. However, the weight vectors of the neighboring units are not close to the input pattern. Additionally, the connective weights do not multiply the signal sent from the input units to the cluster units, unless a dot product measure of similarity is being used.

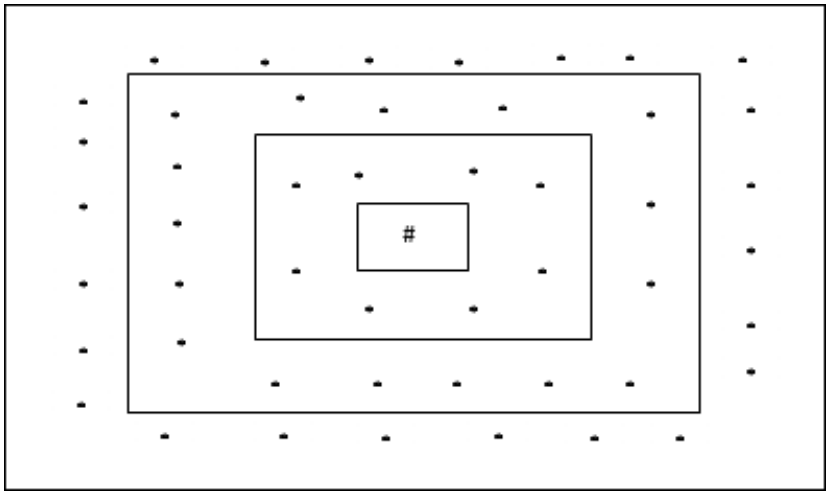
Neighbor Topologies in Kohonen SOM

There can be various topologies, however the following two topologies are used the most

- ***Rectangular Grid Topology***
- ***Hexagonal Grid Topology***

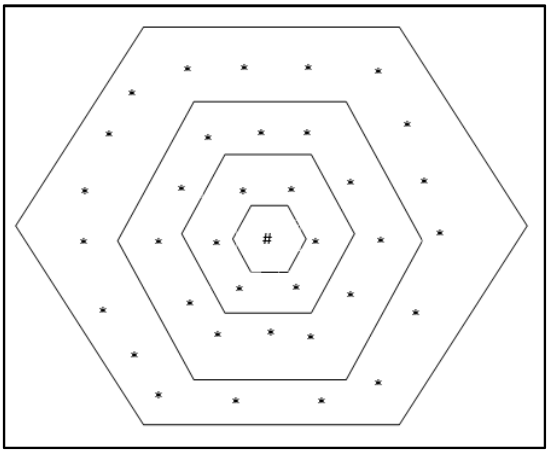
Rectangular Grid Topology

This topology has 24 nodes in the distance-2 grid, 16 nodes in the distance-1 grid, and 8 nodes in the distance-0 grid, which means the difference between each rectangular grid is 8 nodes. The winning unit is indicated by #.



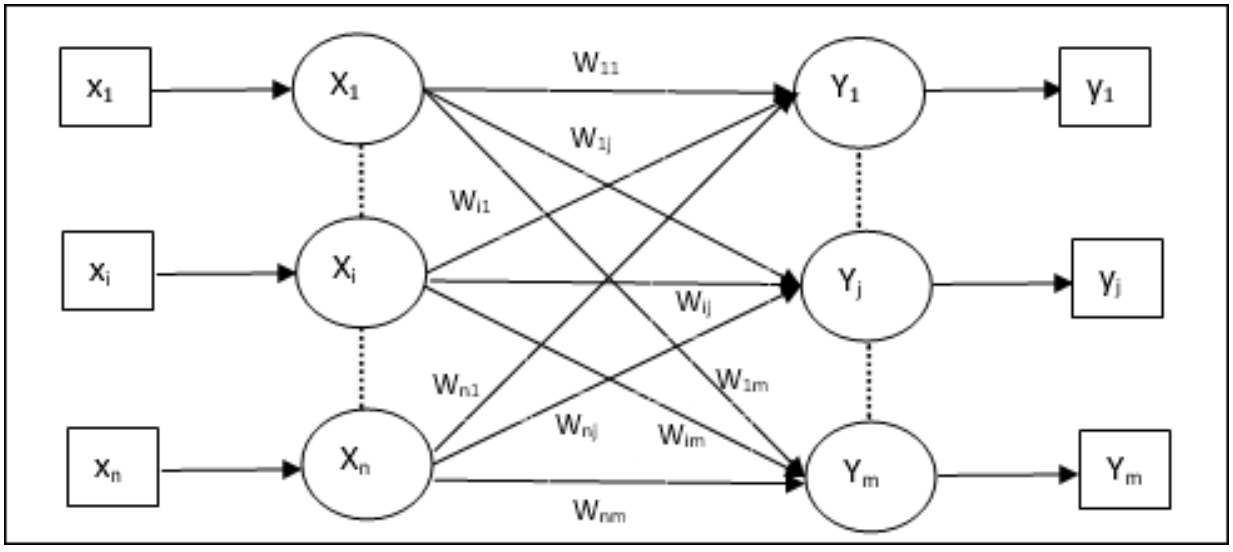
Hexagonal Grid Topology

This topology has 18 nodes in the distance-2 grid, 12 nodes in the distance-1 grid, and 6 nodes in the distance-0 grid, which means the difference between each rectangular grid is 6 nodes. The winning unit is indicated by #.



Architecture

The architecture of KSOM is similar to that of the competitive network. With the help of neighborhood schemes, discussed earlier, the training can take place over the extended region of the network.



Algorithm for training

Step 1 − Initialize the weights, the learning rate **α** and the neighbourhood topological scheme.

Step 2 − Continue step 3-9, when the stopping condition is not true.

Step 3 − Continue step 4-6 for every input vector **x**.

Step 4 − Calculate Square of Euclidean Distance for **j = 1 to m**

$$D(j) = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_i - w_{ij})^2$$

Step 5 − Obtain the winning unit **J** where ***D***j is minimum.

Step 6 − Calculate the new weight of the winning unit by the following relation

$$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)]$$

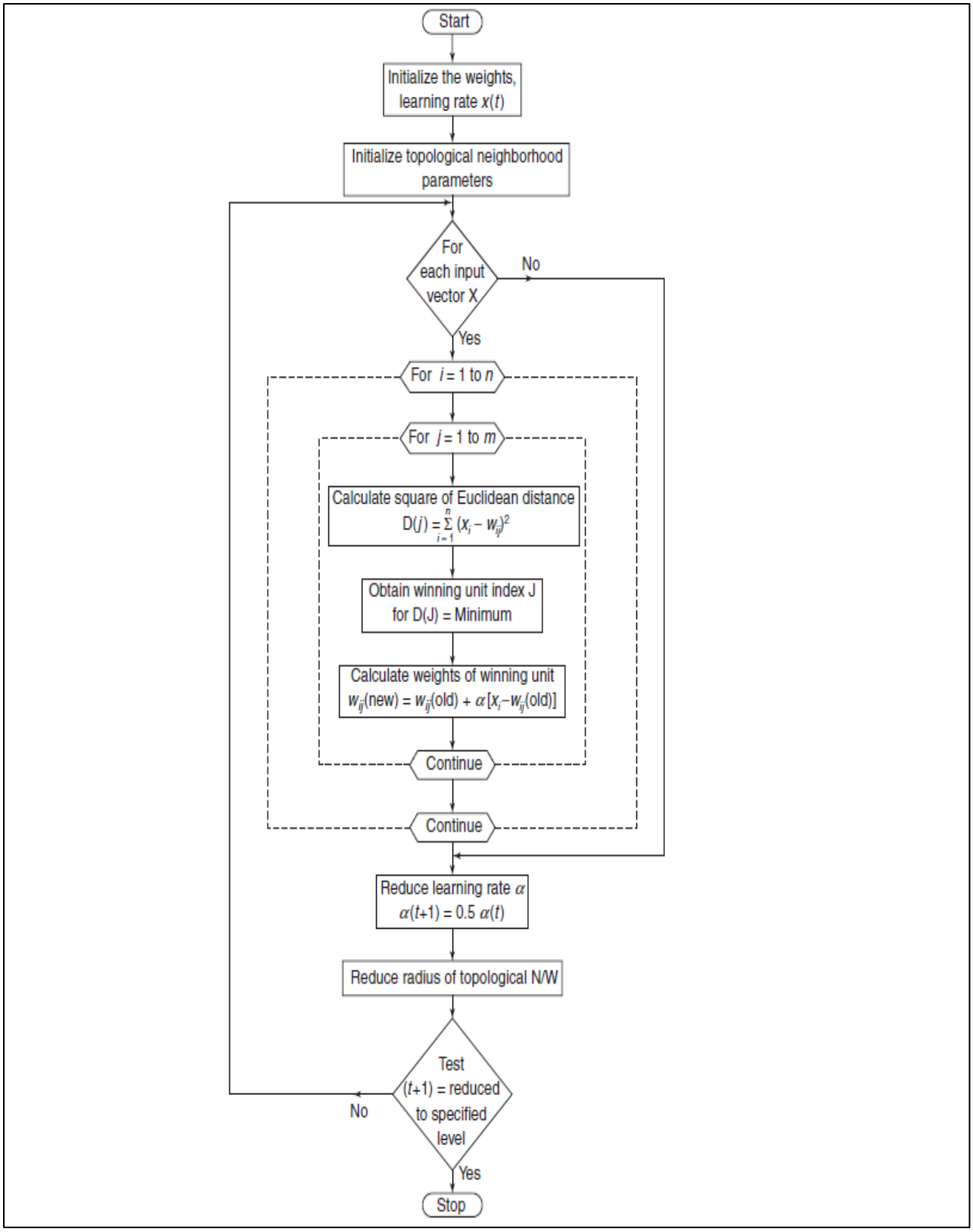
Step 7 − Update the learning rate **α** by the following relation

$$\alpha(t + 1) = 0.5\alpha t$$

Step 8 − Reduce the radius of topological scheme.

Step 9 − Check for the stopping condition for the network.

Flowchart for Training Process:



Kohonen Self-Organizing Motor Map:

The extension of Kohonen feature map for a multilayer network involves the addition of an association layer to the output of the self-organizing feature map layer. The output node is found to associate the desired output values with certain input vectors. This type of architecture is called as Kohonen self-organizing motor map (KSOMM; Ritter, 1992) and layer that is added is called a motor map.

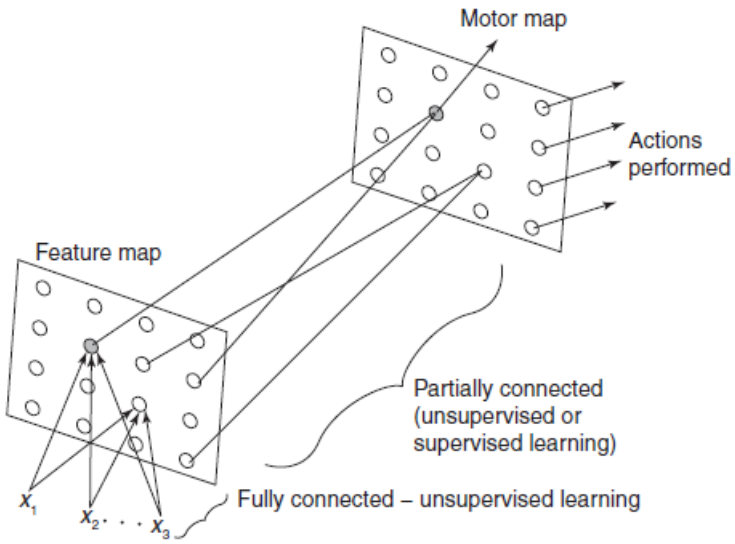


Figure 5-12 Architecture of Kohonen self-organizing motor map.

UNIT – IV

Fuzzy logic:

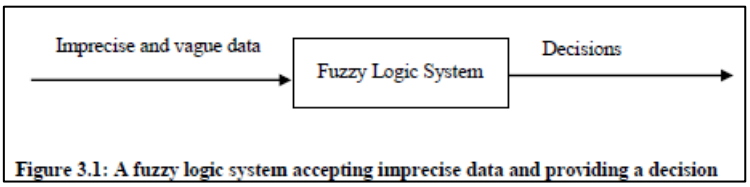


Figure 3.1: A fuzzy logic system accepting imprecise data and providing a decision

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

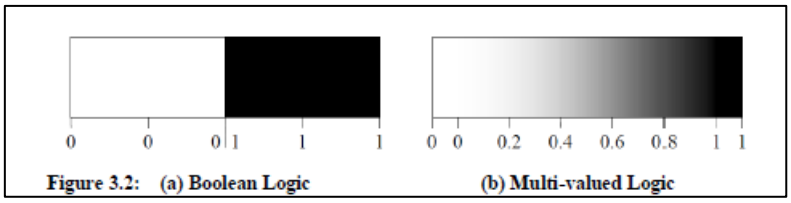


Figure 3.2: (a) Boolean Logic (b) Multi-valued Logic

1. 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 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 is an element belonging to P 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).

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

The set with no elements is defined as an empty set or null set. It is denoted by symbol Ø. The set which consists of all possible subset of a given set A is called power set

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

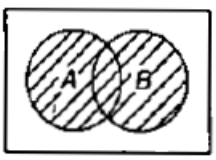
OPERATIONS ON CLASSICAL SETS

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



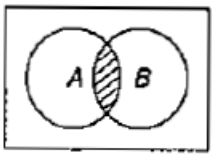
Union of two sets

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



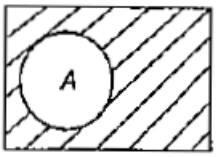
Intersection of two sets

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 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 show below



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.

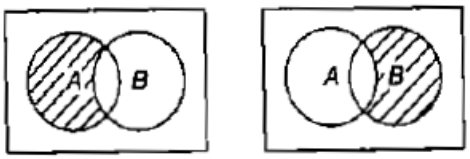
It is denoted by A | B or A- B and is given by

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

The vice versa of it also can be performed

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

The above operations are shown below



(A) **(B)**

(A) Difference A|B or (A-B); (B) Difference B|A or (B-A)

2. FUZZY SETS

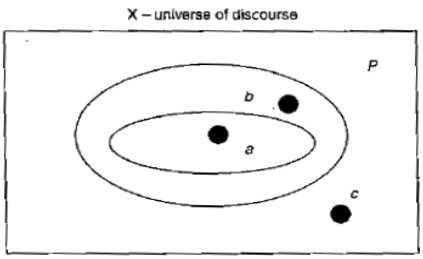
A fuzzy set $A\sim$ in the universe of discourse U can be defined as

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

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

$\mu_{A\sim}(x)$: X → [0, 1], **where $\mu_{A\sim}(x)$ = 1 if x is totally in A;**
$\mu_{A\sim}(x)$ = 0 if x is not in A;
0 < $\mu_{A\sim}(x)$ < 1 if x is partly in A.

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.



Boundary region of a fuzzy set

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 $A\sim$

$$\underset{\sim}{A} \cup \bar{\underset{\sim}{A}} = U; \qquad \underset{\sim}{A} \cap \bar{\underset{\sim}{A}} = \emptyset$$

That is, for fuzzy set $\underset{\sim}{A}$

$$\underset{\sim}{A} \cup \bar{\underset{\sim}{A}} = U; \qquad \underset{\sim}{A} \cap \bar{\underset{\sim}{A}} = \emptyset$$

1. Commutativity

$$\underset{\sim}{A} \cup \underset{\sim}{B} = \underset{\sim}{B} \cup \underset{\sim}{A}; \quad \underset{\sim}{A} \cap \underset{\sim}{B} = \underset{\sim}{B} \cap \underset{\sim}{A}$$

2. Associativity

$$\underset{\sim}{A} \cup \left(\underset{\sim}{B} \cup \underset{\sim}{C}\right) = \left(\underset{\sim}{A} \cup \underset{\sim}{B}\right) \cup \underset{\sim}{C}$$
$$\underset{\sim}{A} \cap \left(\underset{\sim}{B} \cap \underset{\sim}{C}\right) = (\underset{\sim}{A} \cap \underset{\sim}{B}) \cap \underset{\sim}{C}$$

3. Distributivity

$$\underset{\sim}{A} \cup \left(\underset{\sim}{B} \cap \underset{\sim}{C}\right) = \left(\underset{\sim}{A} \cup \underset{\sim}{B}\right) \cap \left(\underset{\sim}{A} \cup \underset{\sim}{C}\right)$$
$$\underset{\sim}{A} \cap \left(\underset{\sim}{B} \cup \underset{\sim}{C}\right) = \left(\underset{\sim}{A} \cap \underset{\sim}{B}\right) \cup \left(\underset{\sim}{A} \cap \underset{\sim}{C}\right)$$

4. Idempotency

$$\underset{\sim}{A} \cup \underset{\sim}{A} = \underset{\sim}{A}; \quad \underset{\sim}{A} \cap \underset{\sim}{A} = \underset{\sim}{A}$$

5. Transitivity

$$If\ \underset{\sim}{A} \subseteq \underset{\sim}{B} \subseteq \underset{\sim}{C}, then\ \underset{\sim}{A} \subseteq \underset{\sim}{C}$$

6. Identity

$$\underset{\sim}{A} \cup \emptyset = \underset{\sim}{A}\ and\ \underset{\sim}{A} \cup U = U$$
$$\underset{\sim}{A} \cap \emptyset = \emptyset\ and\ \underset{\sim}{A} \cap U = \underset{\sim}{A}$$

7. Involution (double negation)

$$\bar{\bar{\underset{\sim}{A}}} = \underset{\sim}{A}$$

8. DeMorgans law

$$\left|\overline{\underset{\sim}{A} \cap \underset{\sim}{B}}\right| = \bar{\underset{\sim}{A}} \cup \bar{\underset{\sim}{B}}; \quad \left|\overline{\underset{\sim}{A} \cup \underset{\sim}{B}}\right| = \bar{\underset{\sim}{A}} \cap \bar{\underset{\sim}{B}};$$

3. FUZZY RELATIONS

A fuzzy relation is a fuzzy set defined on the Cartesian product of classical sets {XI, X2, ... Xn} where tuples (x1, x2, xn) may have varying degrees of membership μ_R (x1,x2, .. , xn) within the relation.

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

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≠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(X2).

Let,

$$X_{\sim}=\{x1,x2,\ldots,xn\} \text{ and } Y_{\sim}=\{y1,y2,\ldots,yn\}$$

Fuzzy relation $R_{\sim}(X_{\sim},Y_{\sim})$ can be expressed by an n × m matrix as follows:

$$R\left(\underset{\sim}{X},\underset{\sim}{Y}\right) = \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \cdots & \mu_R(x_1,y_m) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & \cdots & \mu_R(x_2,y_m) \\ \vdots & \vdots & \vdots & \vdots \\ \mu_R(x_n,y_1) & \mu_R(x_n,y_2) & & \mu_R(x_n,y_m) \end{bmatrix}$$

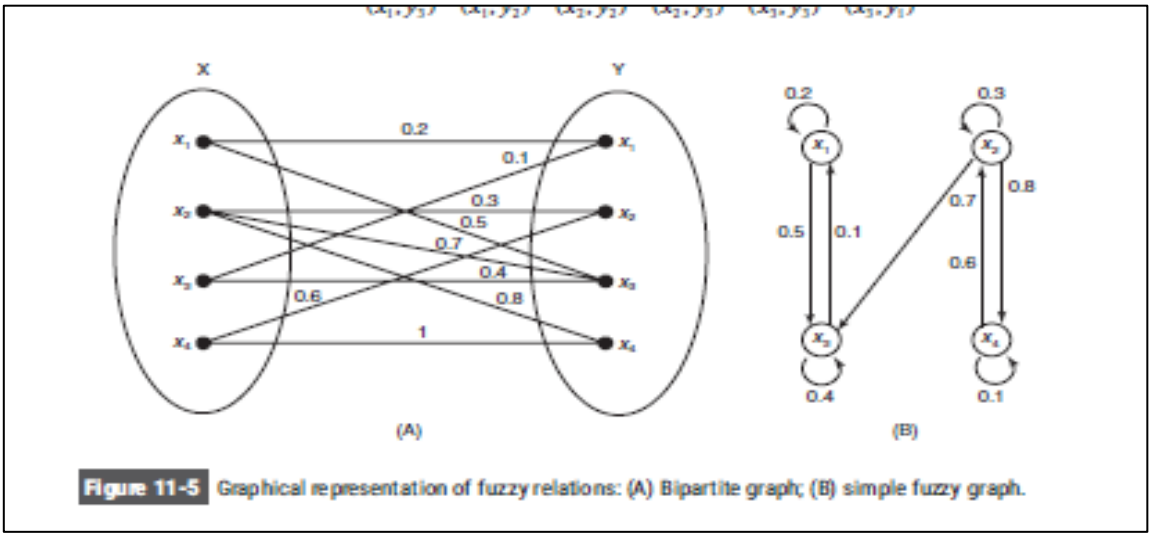


Figure 11-5 Graphical representation of fuzzy relations: (A) Bipartite graph; (B) simple fuzzy graph.

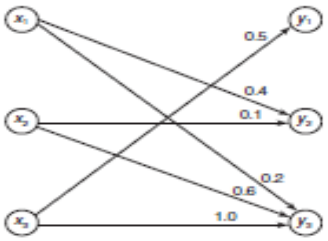


Figure 11-6 Graph of fuzzy relation.

The corresponding fuzzy matrix for relation R is

$$R = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \begin{bmatrix} 0 & 0.4 & 0.2 \\ 0 & 0.1 & 0.6 \\ 0.5 & 0 & 1.0 \end{bmatrix} \end{array}$$

The graph of the above relation $R = X \times Y$ is shown in Figure 11-6.

4. FUZZY MEMBERSHIP FUNCTIONS

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 $A_{\sim} = \{(x, \mu_{A\sim}(x)) \mid x \in X\}$, 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 $A_{\sim}$ can be expressed as

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

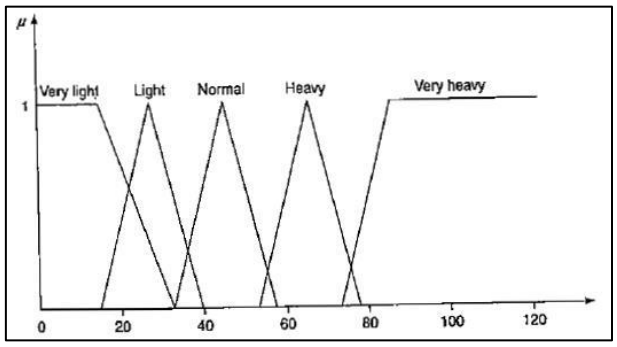
where the symbol ~ means fuzzified. This process of fuzzification is called support fuzzification (s-fuzzification). There is another method of fuzzification called grade fuzzification (g-fuzzification) where x_i is kept constant and μ_i is expressed fuzzy set.

5. METHODS OF MEMBERSHIP VALUE ASSIGNMENTS

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 as to be made. Figure 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



Membership function for the fuzzy variable "weight"

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.
$I_{\sim}$ = isosceles triangle (approximate)
$E_{\sim}$= equilateral triangle (approximate)
$R_{\sim}$ = right-angle triangle (approximate)
$IR_{\sim}$ = isosceles and right-angle triangle (approximate)
$T_{\sim}$= other triangles

The membership can be obtained in equations:

$$\mu_{\underset{\sim}{T}}(X,Y,Z) = min\left\{1-\mu_{\underset{\sim}{I}}(X,Y,Z), 1-\mu_{\underset{\sim}{E}}(X,Y,Z), 1-\mu_{\underset{\sim}{R}}(X,Y,Z)\right\}$$
$$= \frac{1}{180^\circ} min\{3(X-Y), 3(Y-Z), 2|X-90^\circ|, X-Z\}$$

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.

DEFUZZIFICATION:

6. LAMBDA –CUTS FOR FUZZY SETS

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

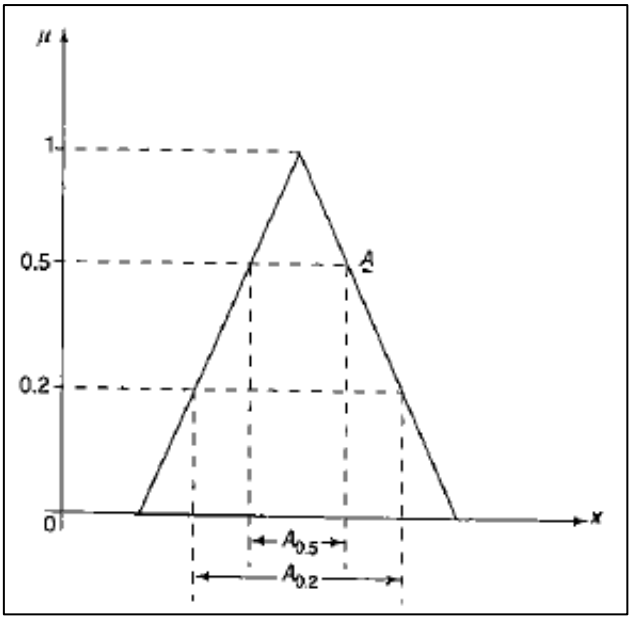
$$A_\lambda = \left\{\left(x, \mu_{\underset{\sim}{A}}(x) \geq \lambda\right)\right\} \lambda \in [0,1]$$

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 consists 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

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

The properties of λ cut are as follows

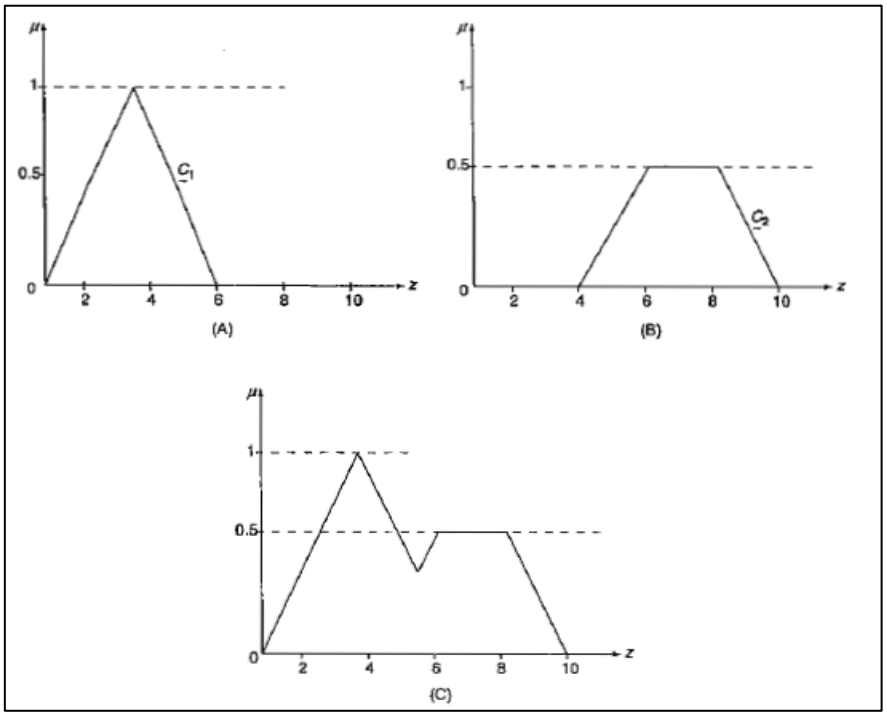
1. $\left(\underset{\sim}{A} \cup \underset{\sim}{B}\right)_\lambda = A_\lambda \cup B_\lambda$
2. $\left(\underset{\sim}{A} \cap \underset{\sim}{B}\right)_\lambda = A_\lambda \cap B_\lambda$
3. $\left(\underset{\sim}{A}\right)_\lambda \neq (\overline{A_\lambda})$ *except when* $\lambda = 0.5$
4. For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



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

7. 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.



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

Defuzzification methods include the following:

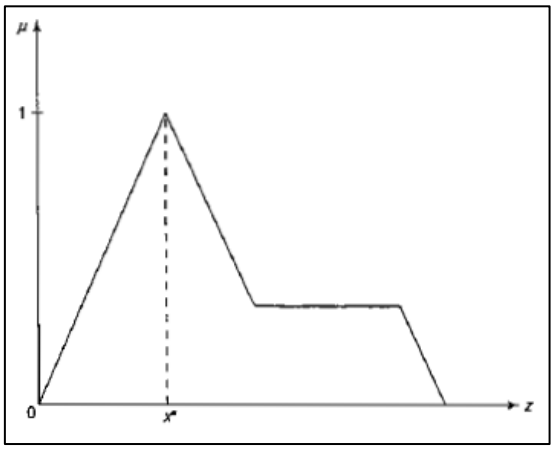
1. Max membership principle
2. Centroid method
3. Weighted average method
4. Mean-max membership
5. Center of sums
6. Center of largest area
7. First of Maxima and last of Maxima

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

$$\mu_{\underset{\sim}{c}}(x^*) \geq \mu_{\underset{\sim}{c}}(x) \; for \; all \; x \in X$$

The method is illustrated in below figure



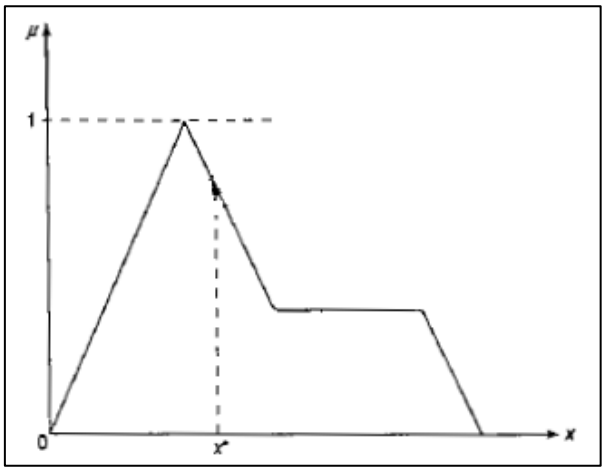
Max-membership defuzzification method

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,

$$x^* = \frac{\int \mu_{\underset{\sim}{c}}(x).\, x dx}{\int \mu_{\underset{\sim}{c}}(x) dx}$$

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



Centroid defuzzification method

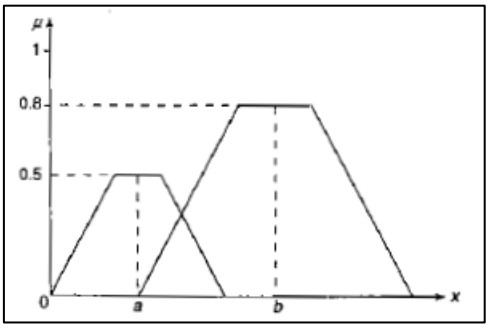
3. Weighted average method

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

$$x^* = \frac{\sum \mu_{\underset{\sim}{c}}(\bar{x}_i).\bar{x}_i}{\sum \mu_{\underset{\sim}{c}}(\bar{x}_i)}$$

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

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



Weighted average defuzzification method (two symmetrical functions)

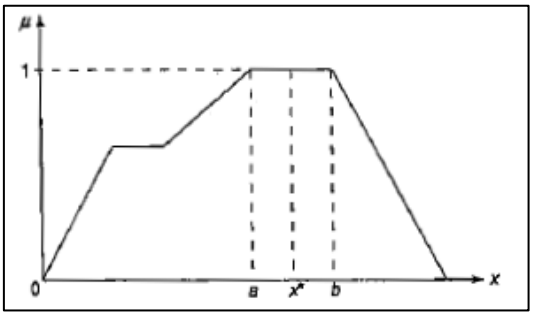
4. Mean-max membership

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, where two fuzzy sets are considered. From the figure the defuzzified output is given by

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



Mean-max defuzzification method

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

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

Figure 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.

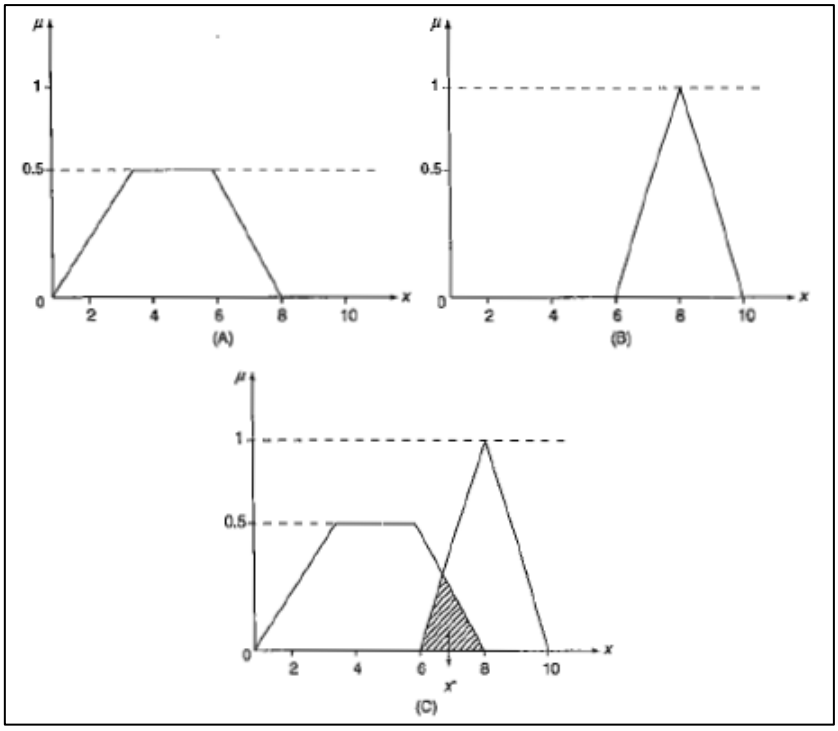


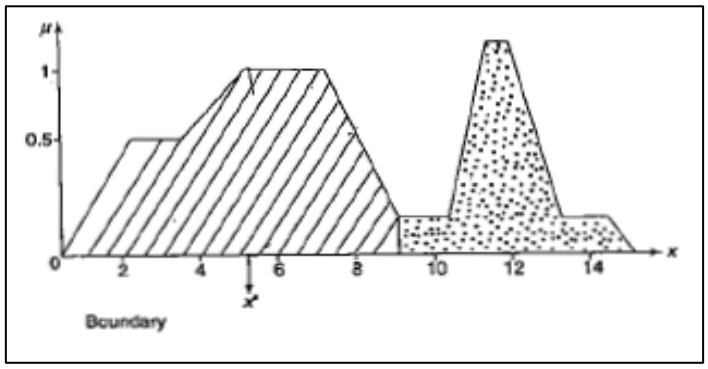
Figure (A) First and (B) second membership functions, (C) defuzzification

6. Center of largest area

This method is adopted when the output consists 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*.

$$x^* = \frac{\int \mu_{\underset{\sim}{ci}}(x).xdx}{\int \mu_{\underset{\sim}{ci}}(x)dx}$$

where $ci_{\sim}$ is the convex subregion that has the largest area making up $ci_{\sim}$. Figure illustrates the center of largest area.



Center of largest area method

7. First of Maxima and last of Maxima:

This method uses the overall output or union of all individual output fuzzy set $ci\sim$ for determining the smallest value of the domain with maximized membership in $cj\sim\cdot$ The steps used for obtaining x* are:

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

$$\text{hgt}(\underset{\sim}{c}_i) = \sup_{x \in X} \mu_{\underset{\sim}{c}_i}(x)$$

where "sup" is supremum, i.e., the least upper bound.

2. Then the first of maxima is found:

$$x^* = \inf_{x \in X} \left\{ x \in X \middle| \mu_{\underset{\sim}{c}_i}(x) = \text{hgt}(\underset{\sim}{c}_i) \right\}$$

where "inf" is the infimum, i.e., the greatest lower bound.

3. After this the last maxima is found:

$$x^* = \sup_{x \in X} \left\{ x \in X \middle| \mu_{\underset{\sim}{c}_i}(x) = \text{hgt}(\underset{\sim}{c}_i) \right\}$$

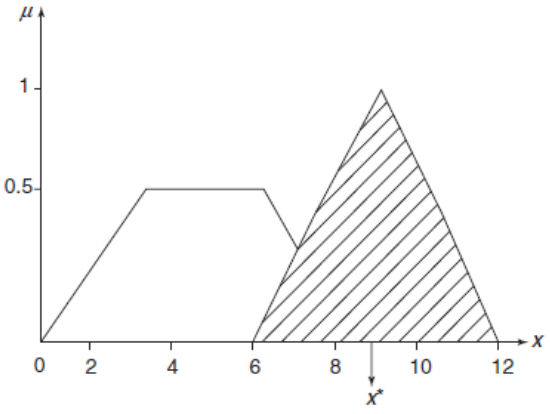


Figure 13-10 First of maxima (last of maxima) method.

8. 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.

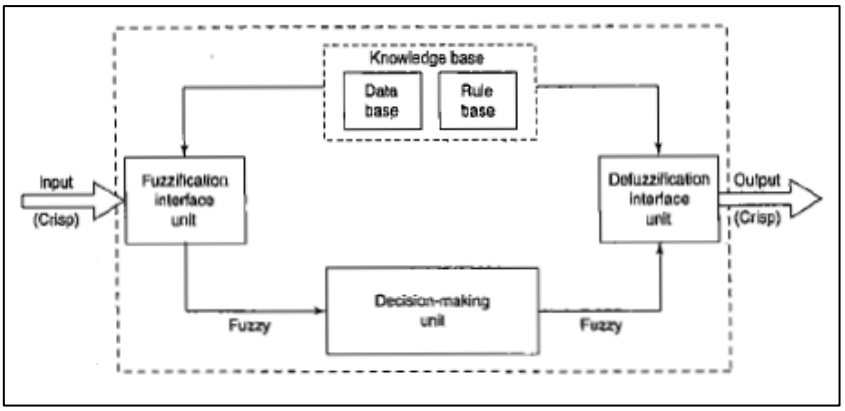
9. FUZZY INFERENCE SYSTEMS

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 FlS may be fuzzy or crisp, but the output from FIS is always a fuzzy set.

Construction and Working Principle 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 convened 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.

Methods of FIS

There are two important types of FIS. They are

1. Mamdani FIS (1975);

2. Sugeno FIS (1985);

1. Mamdani types

Ebsahim 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:

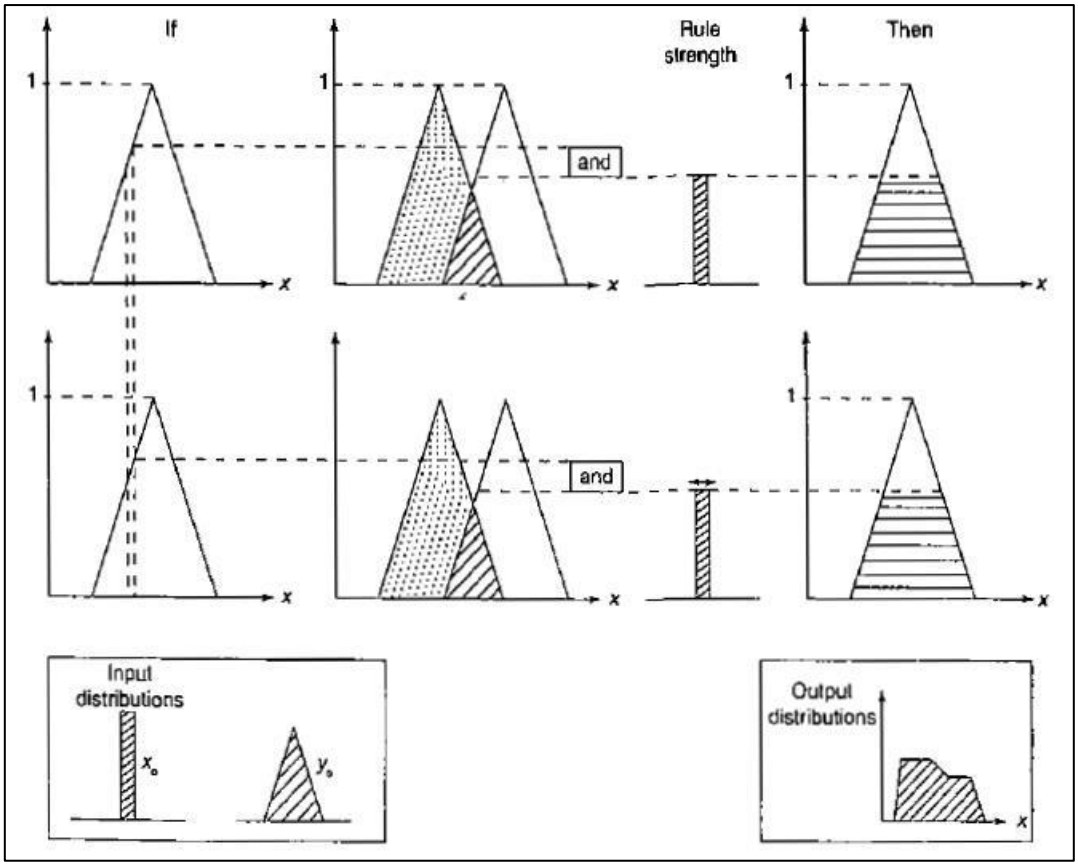
Step 1: Determine a set of fuzzy rules.
Step 2: Make the inputs fuzzy using input membership functions.
Step 3: Combine the inputs according to the fuzzy rules for establishing a rule strength.
Step 4: Determine the consequent of the rule by combining the rule strength and the output membership function.
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



A two-input, two-rule Mamdani FIS with a fuzzy input

2. Sugeno types Takagi-Sugeno Fuzzy Model (TS Method)

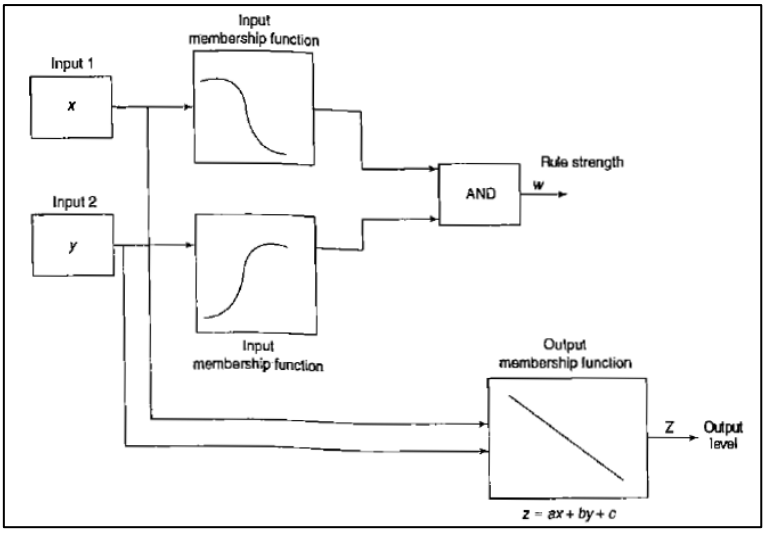
Sugeno fuzzy method was proposed by Takagi, Sugeno and Kang in the year 1985.The format of the fuzzy rule of a Sugeno fuzzy model is given by

IF x is A and y is B THEN z = f(x,y)

where AB 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.**



Sugeno rule

Comparison between Mamdani and Sugeno model

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.

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.

10. FUZZY DECISION MAKING:

It is an activity which includes the steps to be taken for choosing a suitable alternative from those that are needed for realizing a certain goal.

Steps for Decision Making

Let us now discuss the steps involved in the decision-making process

- **Determining the Set of Alternatives** − In this step, the alternatives from which the decision has to be taken must be determined.
- **Evaluating Alternative** − Here, the alternatives must be evaluated so that the decision can be taken about one of the alternatives.
- **Comparison between Alternatives** − In this step, a comparison between the evaluated alternatives is done.

Types of Decision

Making We will now understand the different types of decision making.

1. Individual Decision Making

In this type of decision making, only a single person is responsible for taking decisions. The decision-making model in this kind can be characterized as

- Set of possible actions
- Set of goals Gi(i∈Xn);
- Set of Constraints Cj(j∈Xm)

Now consider a set A. Then, the goal and constraints for this set are given by

$$G_i(a) = \text{composition } [G_i(a)] = G_i^1(G_i(a)) \text{ with } G_i^1$$

$$C_j(a) = \text{composition } [C_j(a)] = C_j^1(C_j(a)) \text{ with } C_j^1 \text{ for } a \in A$$

The fuzzy decision in the above case is given by

$$F_D = min[i \in X_n^{in} fG_i(a), j \in X_m^{in} fC_j(a)]$$

2. Multi-person Decision Making

Decision making in this case includes several persons so that the expert knowledge from various persons is utilized to make decisions.

Calculation for this can be given as follows

Number of persons preferring xi **to** xj = N(xi,xj)

Total number of decision makers = n

$$\text{Then, } SC(x_i, x_j) = \frac{N(x_i, x_j)}{n}$$

3. Multi-objective Decision Making

Multi-objective decision making occurs when there are several objectives to be realized. There are following two issues in this type of decision making

- To acquire proper information related to the satisfaction of the objectives by various alternatives.
- To weigh the relative importance of each objective.

Mathematically we can define a universe of n alternatives as

A= [a1, a2,...,ai,...,an]
And the set of "m" objectives as O= [o1, o2,...,oi,...,on]

4. Multi-attribute Decision Making

Multi-attribute decision making takes place when the evaluation of alternatives can be carried out based on several attributes of the object. The attributes can be numerical data, linguistic data and qualitative data.

Mathematically, the multi-attribute evaluation is carried out on the basis of linear equation as follows

$$Y=A1X1+A2X2+...+AiXi+...+ArXr$$

11. FUZZY LOGIC CONTROL SYSTEM:

Fuzzy logic is applied with great success in various control application. Almost all the consumer products have fuzzy control. Some of the examples include controlling your room temperature with the help of air-conditioner, anti-braking system used in vehicles, control on traffic lights, washing machines, large economic systems, etc.

Why Use Fuzzy Logic in Control Systems

A control system is an arrangement of physical components designed to alter another physical system so that this system exhibits certain desired characteristics. Following are some reasons of using Fuzzy Logic in Control Systems

- While applying traditional control, one needs to know about the model and the objective function formulated in precise terms. This makes it very difficult to apply in many cases.
- By applying fuzzy logic for control, we can utilize the human expertise and experience for designing a controller.
- The fuzzy control rules, basically the IF-THEN rules, can be best utilized in designing a controller.
-

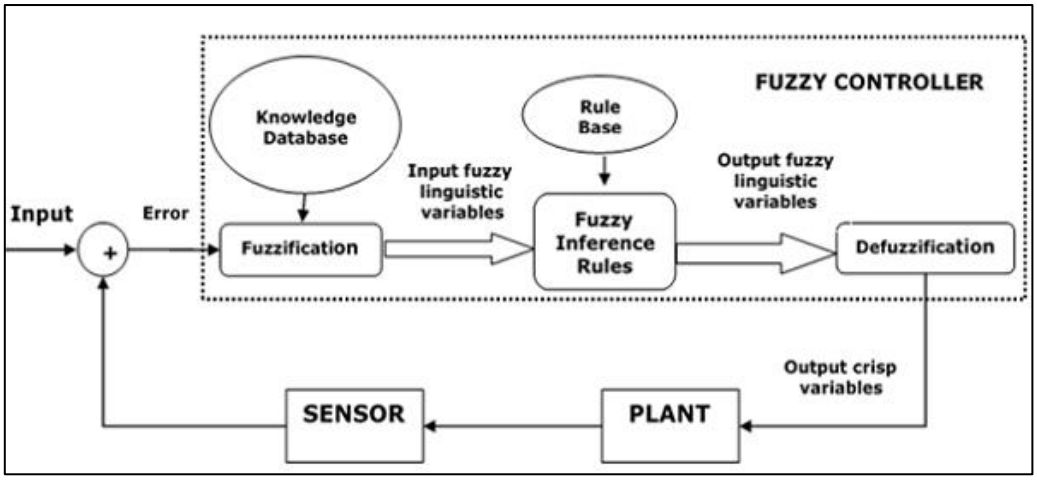
Assumptions in Fuzzy Logic Control (FLC) Design

While designing fuzzy control system, the following six basic assumptions should be made −

- **The plant is observable and controllable** − It must be assumed that the input, output as well as state variables are available for observation and controlling purpose.
- **Existence of a knowledge body** − It must be assumed that there exist a knowledge body having linguistic rules and a set of input-output data set from which rules can be extracted.
- **Existence of solution** − It must be assumed that there exists a solution.
- **'Good enough' solution is enough** − The control engineering must look for 'good enough' solution rather than an optimum one.
- **Range of precision** − Fuzzy logic controller must be designed within an acceptable range of precision.
- **Issues regarding stability and optimality** − The issues of stability and optimality must be open in designing Fuzzy logic controller rather than addressed explicitly.

Architecture of Fuzzy Logic Control

The following diagram shows the architecture of Fuzzy Logic Control (FLC).



Major Components of FLC

Followings are the major components of the FLC as shown in the above figure −

- **Fuzzifier** − The role of fuzzifier is to convert the crisp input values into fuzzy values.
- **Fuzzy Knowledge Base** − It stores the knowledge about all the input-output fuzzy relationships. It also has the membership function which defines the input variables to the fuzzy rule base and the output variables to the plant under control.
- **Fuzzy Rule Base** − It stores the knowledge about the operation of the process of domain.
- **Inference Engine** − It acts as a kernel of any FLC. Basically it simulates human decisions by performing approximate reasoning.
- **Defuzzifier** − The role of defuzzifier is to convert the fuzzy values into crisp values getting from fuzzy inference engine.

Steps in Designing FLC

Following are the steps involved in designing FLC

- Identification of variables
- Fuzzy subset configuration
- Obtaining membership function
- Fuzzy rule base configuration
- Fuzzification
- Combining fuzzy outputs
- Defuzzification

Advantages of Fuzzy Logic Control

Let us now discuss the advantages of Fuzzy Logic Control.

- Cheaper
- Robust
- Customizable
- Emulate human deductive thinking
- Reliability
- Efficiency

Disadvantages of Fuzzy Logic Control

We will now discuss what are the disadvantages of Fuzzy Logic Control.

- Requires lots of data
- Useful in case of moderate historical data
- Needs high human expertise
- Needs regular updating of rules

UNIT-V

1. GENETIC PROGRAMMING:

Genetic programming (GP) is a type of evolutionary algorithm that uses the principles of natural evolution to solve problems, particularly in parameter optimization. The four preconditions for natural selection are entities called individuals that form a population, heredity in reproduction, variety in the course of reproduction, and finite resources that cause individuals to compete. GP is part of a larger field called evolutionary computation, which uses biological reproduction and evolution as a metaphor for computer-based problem-solving.

GP automatically creates a working computer program from a high-level problem statement using the principles of Darwinian natural selection and biologically inspired operations such as reproduction, crossover, mutation, gene duplication, and gene deletion. GP is an excellent problem solver, function approximator, and tool for writing functions to solve specific tasks. However, it does not replace programmers, who still need to specify the fitness function and identify the problem to which GP should be applied.

GP is a domain-independent method that uses evolutionary principles to breed a population of computer programs to solve a problem, and it is a promising technology that will continue to make inroads into automatic programming and machine learning in the future.

CHARACTERISTICS OF GENETIC PROGRAMMING

Genetic programming (GP) is a type of evolutionary algorithm that has several characteristics:

1. **Representation:** GP represents the solution to a problem as a computer program, typically in the form of a tree structure or a linear string of code.
2. **Fitness function:** GP uses a fitness function to evaluate the performance of each program in the population. The fitness function is designed to measure how well the program solves the problem at hand.

3. **Genetic operators**: GP uses genetic operators such as crossover, mutation, reproduction, gene duplication, and gene deletion to create new programs in the population. These operations are patterned after natural biological processes.
4. **Population-based search**: GP maintains a population of programs and searches for the best solution within that population. The search process involves iteratively creating new programs through genetic operators and evaluating them using the fitness function.
5. **Domain-independent:** GP is a domain-independent method, meaning that it can be applied to a wide range of problem domains without requiring domain-specific knowledge or expertise.
6. **Automated program creation:** GP automatically creates programs to solve a problem, starting from a high-level problem statement. This makes it a powerful tool for program synthesis or program induction.
7. **Human involvement:** Despite its ability to automatically create programs, GP still requires human involvement in specifying the fitness function and identifying the problem to which it should be applied.

Overall, GP is a powerful tool for automated program creation and problem-solving that uses evolutionary principles and genetic operators to iteratively create and improve programs in a population.

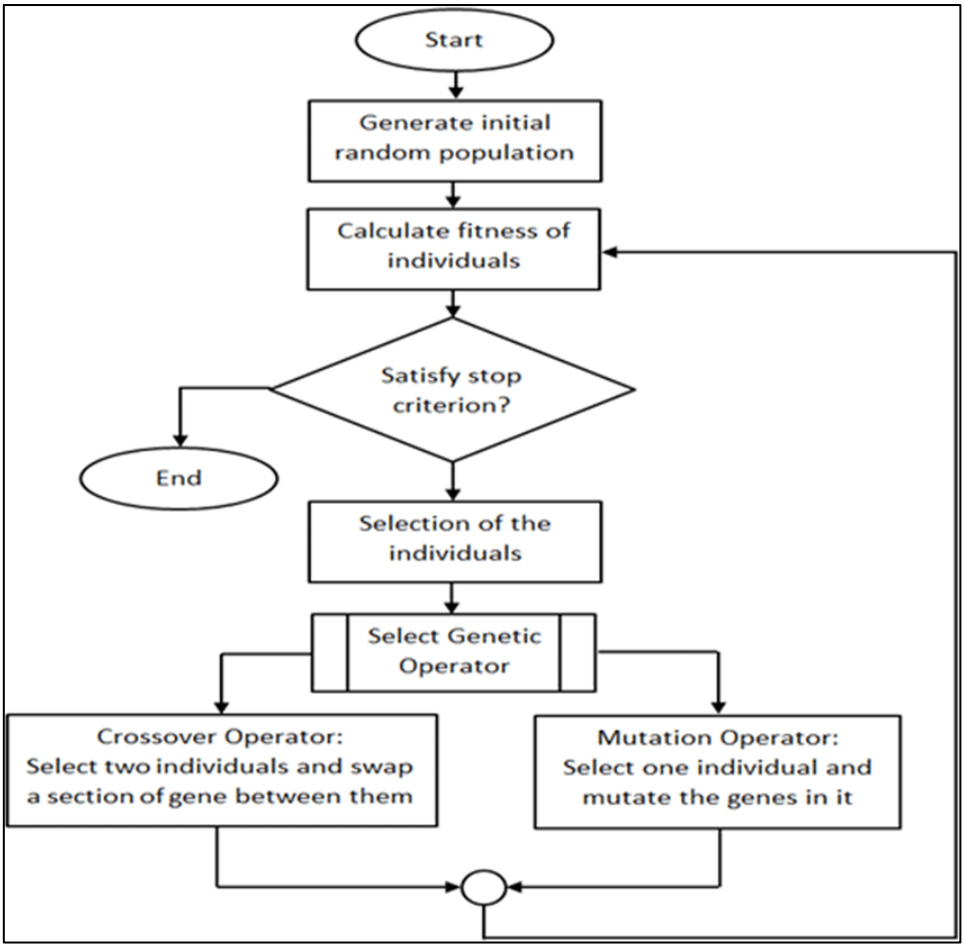
WORKING OF GNETIC PROGMMING

Genetic programming (GP) works by using a population-based search to create and improve computer programs that solve a given problem. The following steps outline the basic working of GP:

1. **Initialization**: A population of random computer programs is generated. Each program is represented as a tree structure or a linear string of code.
2. **Fitness evaluation:** Each program in the population is evaluated using a fitness function that measures how well the program solves the problem at hand.
3. **Selection:** A subset of programs from the population is selected for breeding based on their fitness scores. The selection process is typically based on the principle of "survival of the fittest."

4. **Reproduction:** The selected programs are then used to create new programs through genetic operators such as crossover, mutation, reproduction, gene duplication, and gene deletion.
5. **Evaluation:** The new programs are evaluated using the fitness function, and the best ones are added to the population.
6. **Termination:** The process of selection, reproduction, and evaluation is repeated iteratively until a termination criterion is met. The termination criterion can be a fixed number of generations, a certain level of fitness score, or other measures.
7. **Solution:** The best program in the final population is considered as the solution to the problem.

Overall, GP uses the principles of natural selection and genetic operators to evolve and improve computer programs over multiple generations. By iteratively creating new programs and evaluating them using a fitness function, GP can find solutions to a wide range of problems without requiring domain-specific knowledge or expertise. However, GP still requires human involvement in specifying the fitness function and identifying the problem to which it should be applied.



DATA REPRESENTAITON

Data representation is a crucial aspect of soft computing, which includes techniques such as fuzzy logic, neural networks, genetic algorithms, and other related methods. Each of these techniques uses different methods for representing data, but they share the common goal of handling imprecise or uncertain information.

In fuzzy logic, data is represented using fuzzy sets, which allow for degrees of membership rather than a strict binary true or false value. Fuzzy sets enable the representation of vague or uncertain concepts, such as "tall" or "old," by assigning a degree of membership to each element in a set. For example, a person's height can be represented using a fuzzy set that assigns a degree of membership to each height range, such as "short," "medium," and "tall."

In neural networks, data is represented using a set of input values that are fed into the network. Each input value corresponds to a node in the input layer of the network, and the values are processed through a series of hidden layers to produce an output. The weights and biases of the network are adjusted during training to optimize the network's performance on a given task.

In genetic algorithms, data is represented using a set of genes, which can be thought of as parameters that determine the behaviour of the algorithm. Each gene represents a particular aspect of the solution space, and the algorithm evolves over time by mutating and recombining genes to create new solutions.

Other soft computing techniques may use different methods for data representation, depending on the problem domain and the specific application. The key advantage of soft computing techniques is their ability to handle complex and uncertain data in a flexible and adaptive manner. By representing data in a way that allows for imprecision and uncertainty, soft computing can provide more robust and accurate solutions to a wide range of problems.

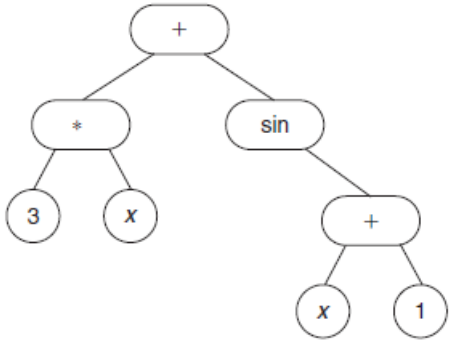


Figure 21-48 The tree representation of $3x + \sin(x + 1)$.