1.1 Introduction to Soft Computing

Two major problem solving techniques are:

- **Hard computing**

  It deals with precise model where accurate solutions are achieved.

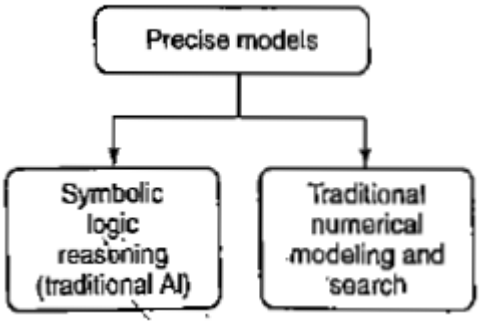

Figure 1.1: Hard Computing

- **Soft computing**

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

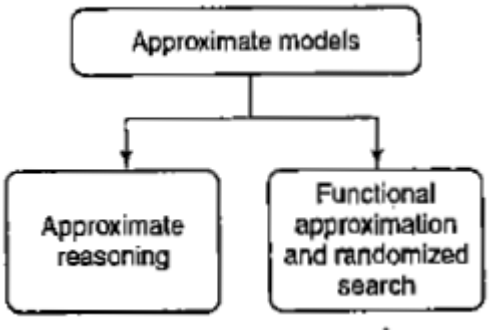

Figure 1.2: Soft Computing

1.2 Biological Neurons

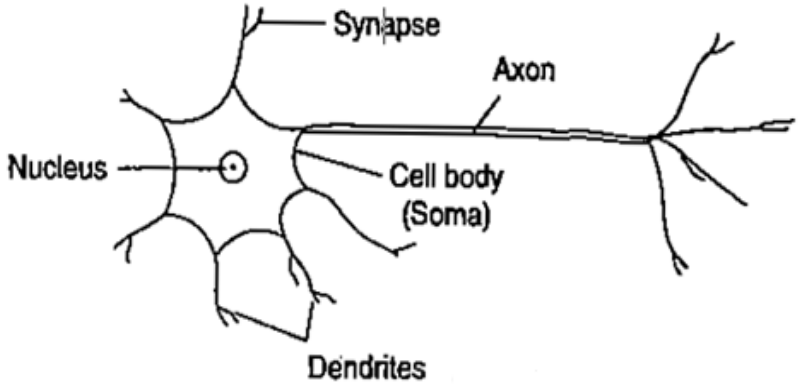

Figure 1.3: Schematic diagram of a biological neuron

The biological neuron consists of main three parts:

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

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

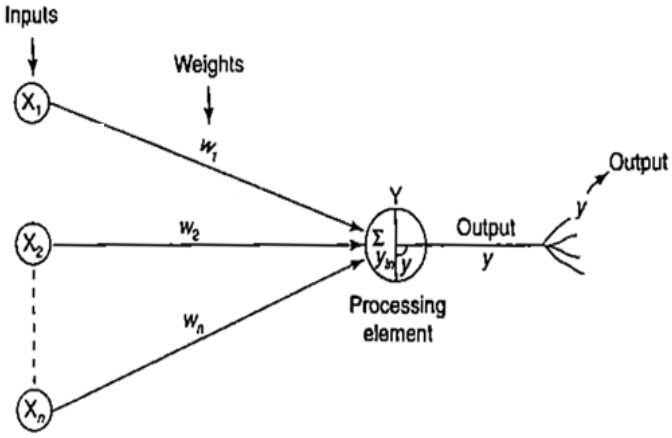

Figure 1.4: Mathematical model of artificial neuron

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

Table 1.1: Terminology relationships between biological and artificial neurons

In this model net input is calculated as

$$y_{in} = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = \sum_{i=1}^{n} x_i w_i$$

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

1.3 Artificial neural networks

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

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

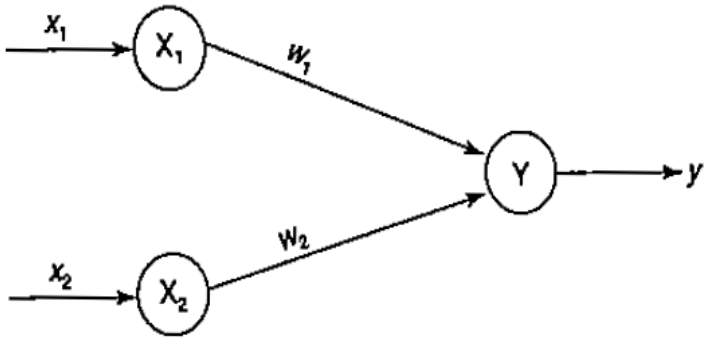

Figure 1.5: Architecture of a simple artificial neuron net

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

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

$$y_{in} = x_1 w_1 + x_2 w_2$$

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

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

Output = Function (net input calculated)

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

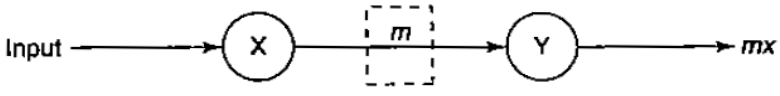

Figure 1.6: Neural net of pure linear equation

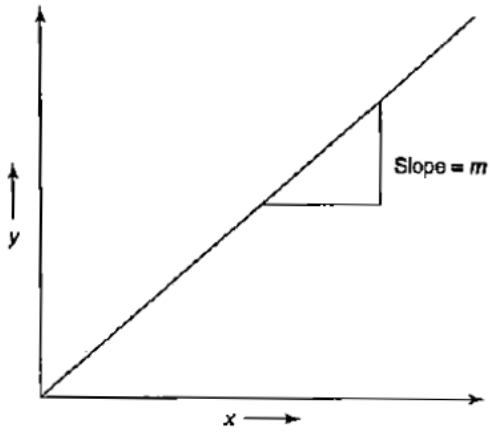

Figure 1.7: Graph for $y = mx$

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

1.4 Comparison between Biological neuron and Artificial neuron

Term	Brain	Computer
Speed	Execution time is few milliseconds	Execution time is few nano seconds
Processing	Perform massive parallel operations simultaneously	Perform several parallel operations simultaneously. It is faster the biological neuron
Size and complexity	Number of Neuron is 10^{11} and number of interconnections is 10^{15}. So complexity of brain is higher than computer	It depends on the chosen application and network designer.
Storage capacity	• Information is stored in interconnections or in synapse strength. • New information is stored without destroying old one. • Sometimes fails to recollect information	• Stored in continuous memory location. • Overloading may destroy older locations. • Can be easily retrieved

Tolerance	• Fault tolerant • Store and retrieve information even interconnections fails • Accept redundancies	• No fault tolerance • Information corrupted if the network connections disconnected. • No redundancies
Control mechanism	Depends on active chemicals and neuron connections are strong or weak	CPU Control mechanism is very simple

Table 1.2: Comparison between Biological neuron and Artificial neuron

Characteristics of ANN:

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

1.5 Evolution of neural networks

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

1960	Adaline	Widrow and Hoff	Here the weights are adjusted to reduce the difference between the net input to the output unit and the desired output.
1972	Kohonen self-organizing feature map	Kohonen	Inputs are clustered to obtain a fired output neuron.
1982, 1984, 1985, 1986, 1987	Hopfield network	John Hopfield and Tank	Based on fixed weights. Can act as associative memory nets
1986	Back propagation network	Rumelhart, Hinton and Williams	• Multilayered • Error propagated backward from output to the hidden units
1988	Counter propagation network	Grossberg	Similar to kohonen network.
1987-1990	Adaptive resonance Theory(ART)	Carpenter and Grossberg	Designed for binary and analog inputs.
1988	Radial basis function network	Broomhead and Lowe	Resemble back propagation network, but activation function used is Gaussian function.
1988	Neo cognitron	Fukushima	For character recognition.

Table 1.3: Evolution of neural networks

1.6 Basic models of artificial neural networks

Models are based on three entities

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

1.6.1 Connections

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

They are:

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

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

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

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

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

1. Single layer feed forward network

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

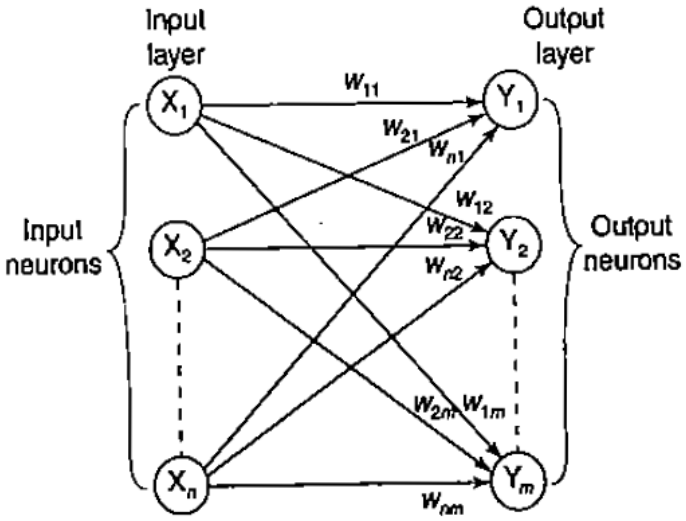

Figure 1.8: Single-layer feed-forward network

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

2. Multilayer feed-forward network

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

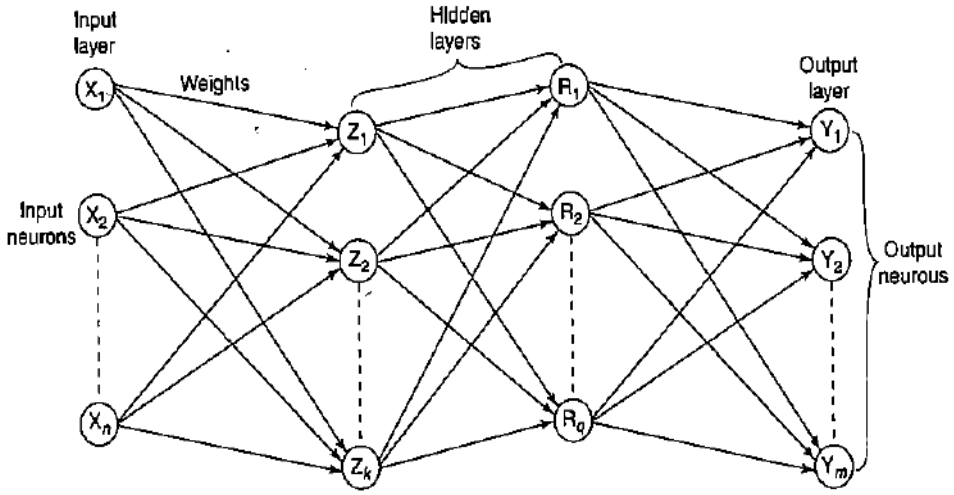

Figure 1.9: Multilayer feed-forward network

3. Single node with its own feedback

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

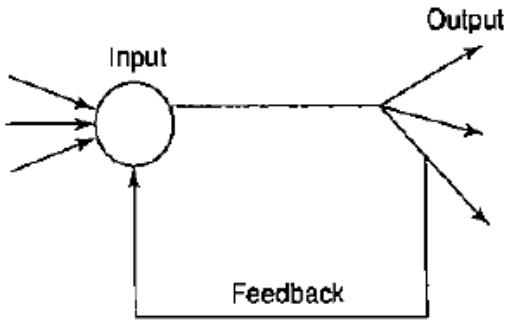

Figure 1.10: Single node with own feedback

4. Single layer recurrent network

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

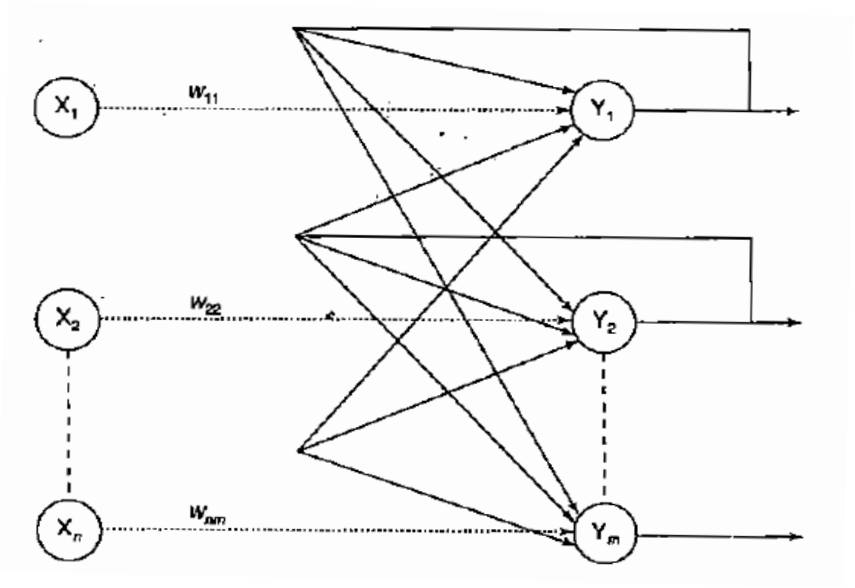

Figure 1.11: Single-layer recurrent network

5. Multilayer recurrent network

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

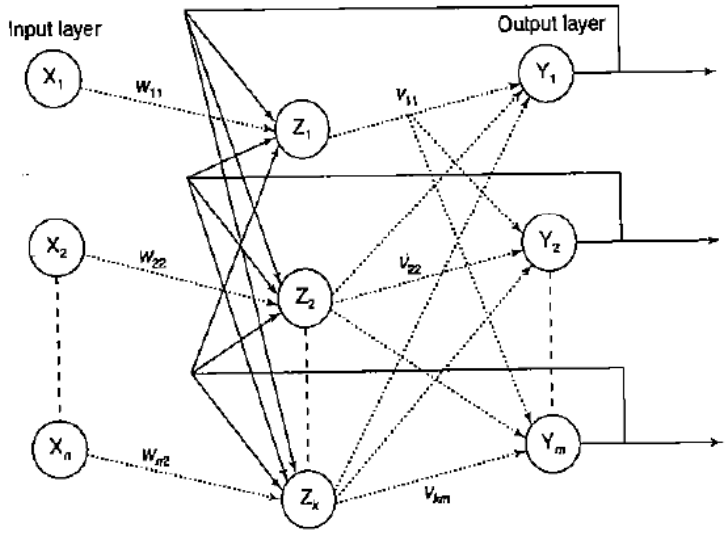

Figure 1.12: Multilayer recurrent network

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

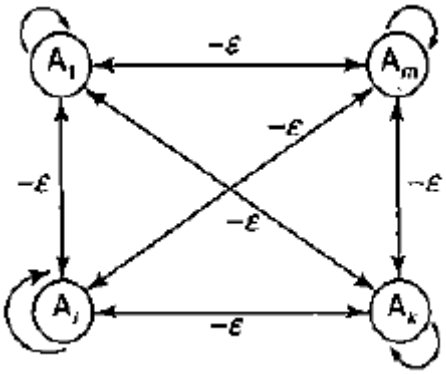

Figure 1.13: Competitive nets

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

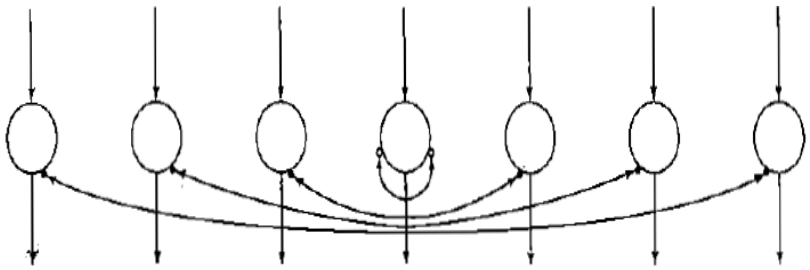

Figure 1.14: Lateral inhibition structure

1.6.2 Learning

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

Two broad kinds of learning in ANNs is:

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

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

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

i) Supervised learning

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

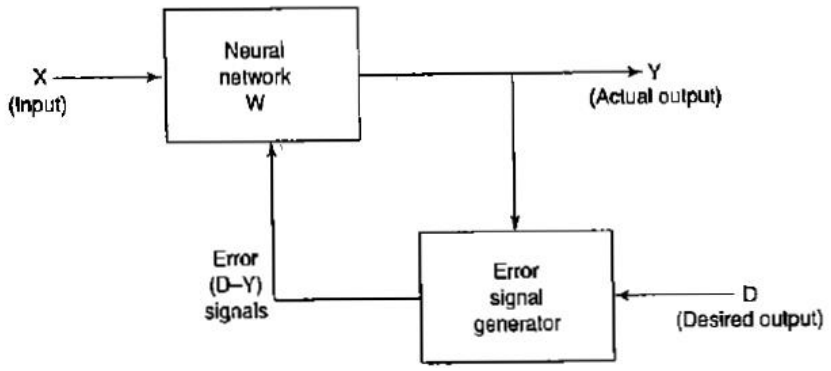

Figure 1.15: Supervised learning

ii) Unsupervised learning

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

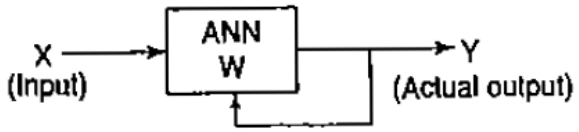

Figure 1.16: Unsupervised learning

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

iii) Reinforcement learning

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

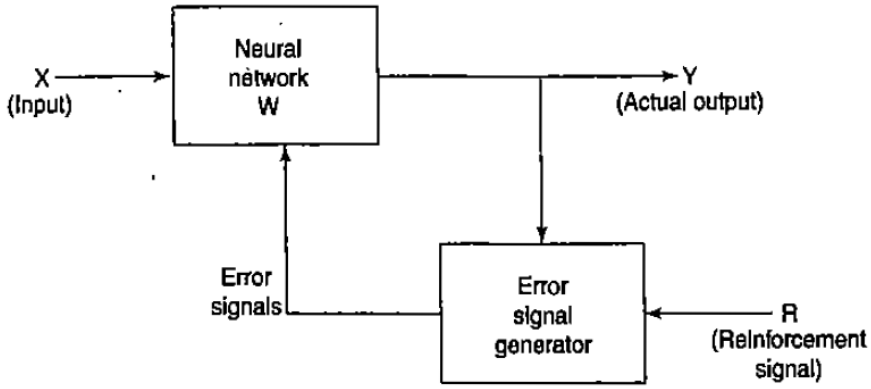

Figure 1.17: Reinforcement learning

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

1.6.3 Activation Functions

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

Several activation functions are there.

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

$$f(x) = x \; for \; all \; x$$

The output is same as the input.

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

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

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

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

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

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

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

Two types:

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

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

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

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

The range of sigmoid function is 0 to 1.

b) Bipolar sigmoid function: This function is defined as

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

Where λ represents steepness parameter and the sigmoid range is between -1 and +1. The derivative of this function can be

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

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

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

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

The derivative of the hyperbolic tangent function is

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

5. Ramp function: The ramp function is defined as

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

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

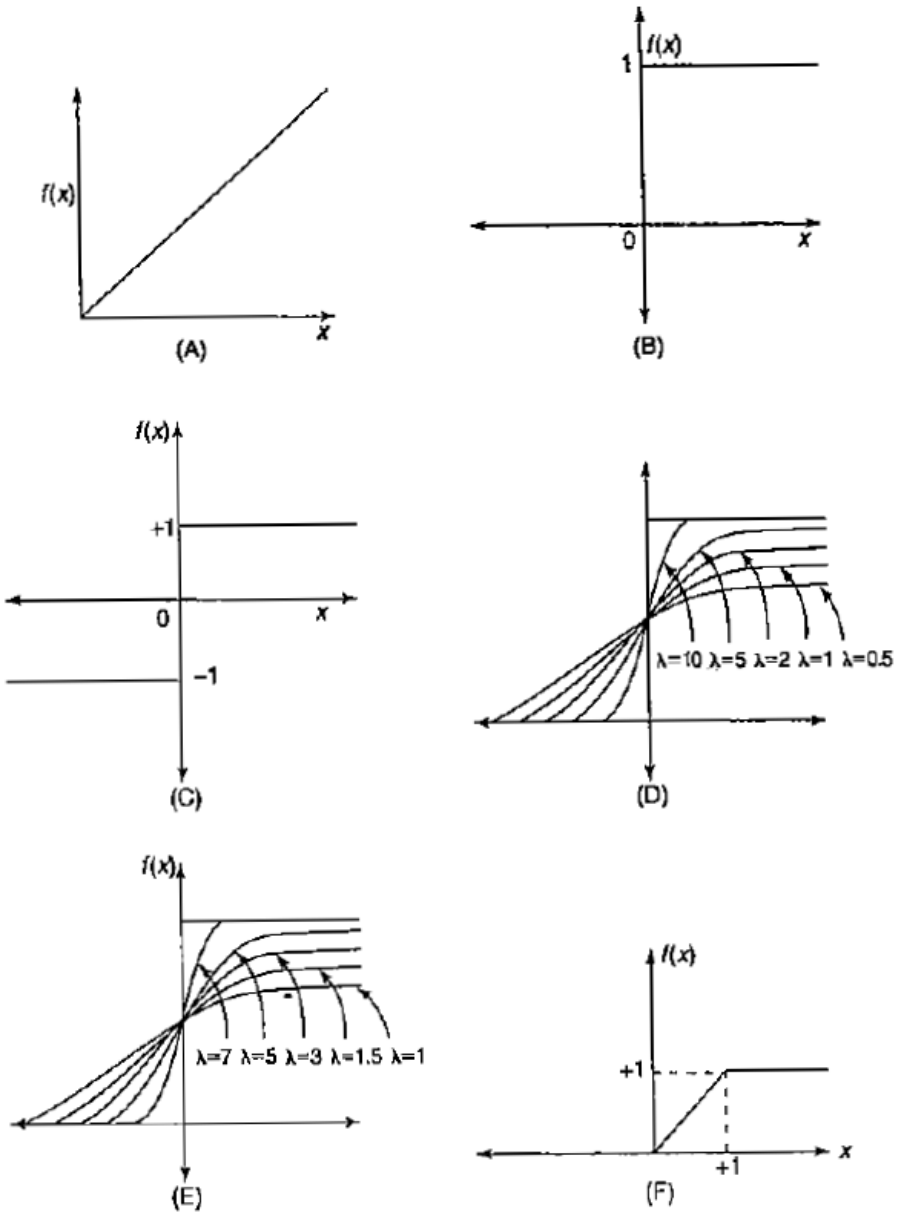

Figure 1.18: Depiction of activation functions: (A) identity function; (B) binary step function; (C) bipolar step function; (D) binary sigmoidal function; (E) bipolar sigmoidal function; (F) ramp function.

1.7 McCulloch and Pitts Neuron

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

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

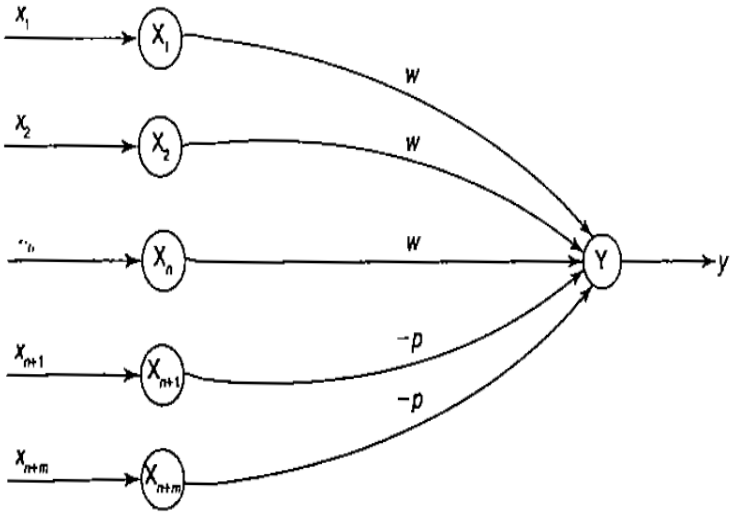

Figure 1.19: McCulloch-Pins neuron model

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

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

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

$$\theta > nw - p$$

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

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

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

1.8 Hebb network

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

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

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

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

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

Training algorithm

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

Step 0: First initialize the weights. Basically in this network they may be set to zero, i.e., *w;* = 0, for i= 1 to *n* where *"n"* may be the total number of input neurons.
Step 1: Steps 2-4 have to be performed for each input training vector and target output pair, s: t.
Step 2: Input units activations are set. Generally, the activation function of input layer is identity function: $x_i = s_i$ *for i=1 to n*
Step 3: Output units activations are set: $y = t$.
Step 4: Weight adjustments and bias adjustments are performed:

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

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

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

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

Hence, Change in weight is expressed as

$$\Delta w = xy$$

As a result,

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

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

Flowchart of Training algorithm

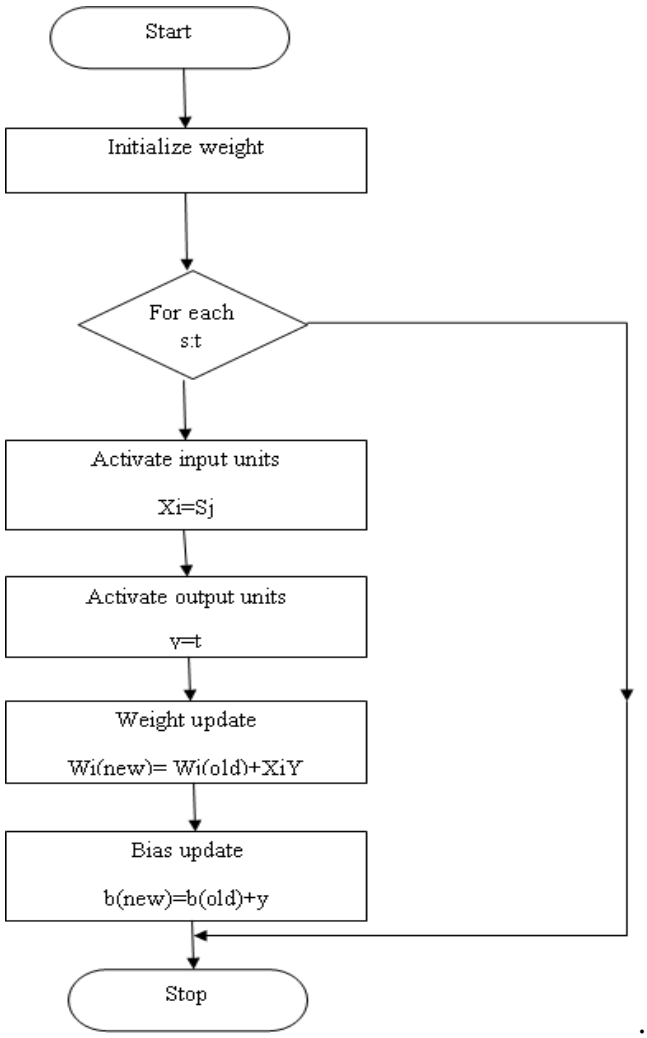

Figure 1.20: Flowchart of Hebb training algorithm