Soft Computing ADALINE (Part-3)

1. ADALINE is a single layer neural network which uses linear activation function (Bipolar activation function) for its input and target outputs.
2. Learning rate is between 0.1 to 1 and bias to 0.
3. ADALINE use delta rule to update the weights between the connections to minimize the difference between output value and target value.
4. Delta rule for change in weight

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

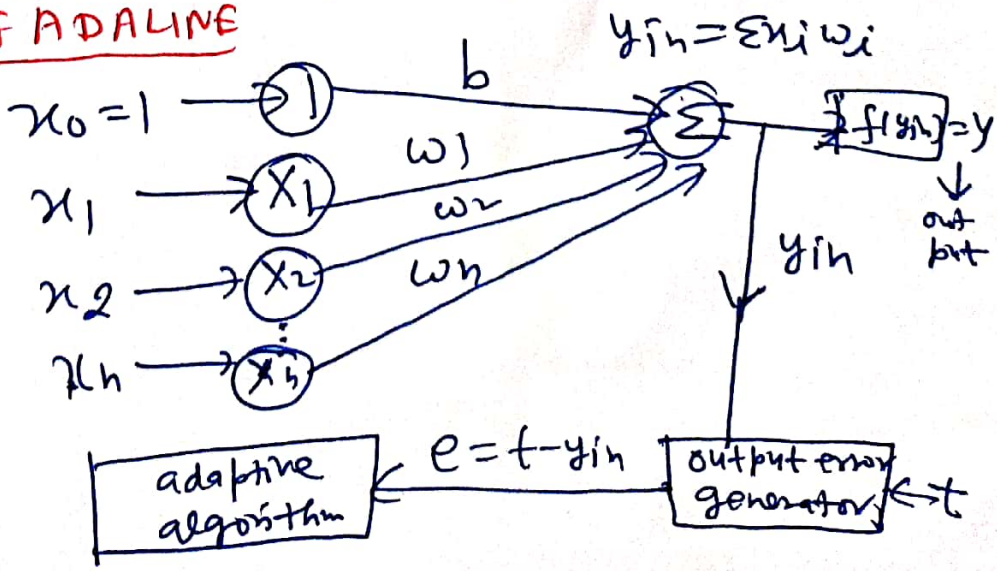
where Δw_i = Change in weight
α = learning rate
x_i = input vector $(-1, +1)$
y_{in} = net input to output unit

$$Y = f(y_{in}) = \Sigma (x_i w_i)$$

t = target output

$$\text{Error } (E_i) = \Sigma (t - y_{in})^2$$

Architecture of ADALINE



ADALINE Training algorithm

(i) weights and bias are set to some random values but not zero. Set the learning rate parameter α.

(ii) Perform steps (iii) to (VII) when stopping condition is false.

(iii) Perform steps (IV) to (VI) for each bipolar training pair s:t

(iv) Set activations for input unit $i = 1$ to n

$$x_i = s_i$$

(v) Calculate the net input to the output unit

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

(vi) Update the weights and bias for $i = 1$ to n

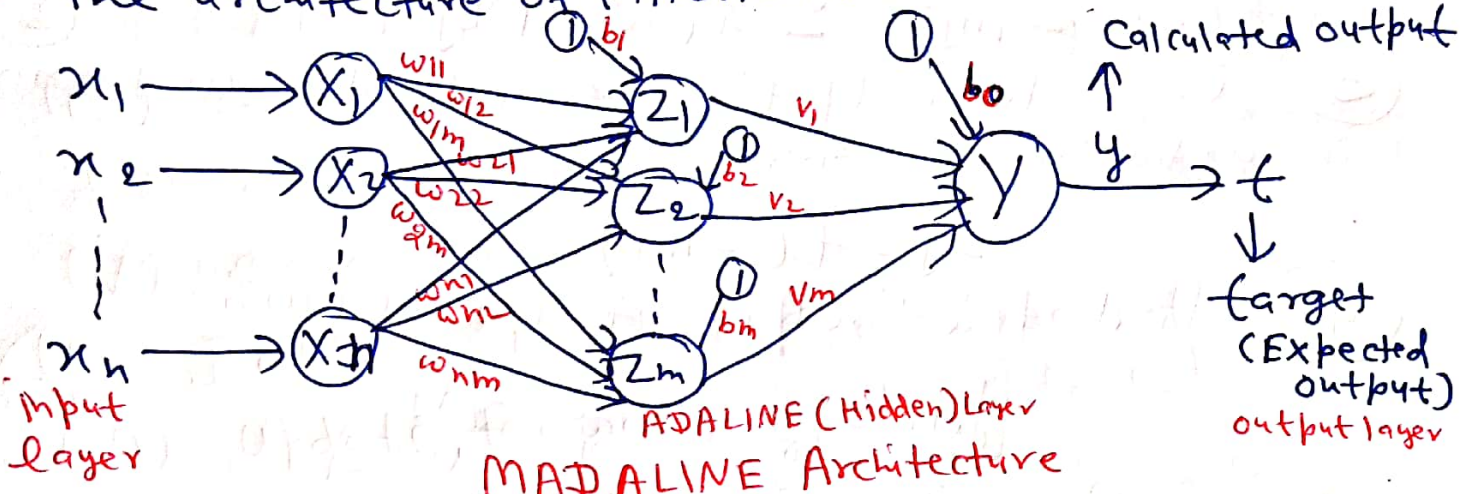
$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$

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

(vii) If the highest weight change that occurred during training is smaller than a specific tolerance then stop the training process else continue

MADALINE :

* This model consists of many ADALINE in parallel with a single output unit whose value is based on certain selection rule.
* The architecture of MADALINE is as given:



MADALINE Architecture

* The weights between input layer and ADALINE layer are adjusted during the training process.
* The weights between ADALINE layer and MADALINE (output) layer are fixed, positive and posses equal value.
* The ADALINE and MADALINE layer neuron have a bias of input (1) connected to them.
* we will initialize the weight between input layer and ADALINE layer. ($w_{11}, w_{12}, w_{21}, w_{22}, \dots w_{n1}, w_{n2}, \dots w_{nm}$) to some random value.
* initiate weight between ADAline and MADALINE layer to some random value ($v_1, v_2 \dots v_m$). Also set learning rate to some fix value.
* Activation function for MADALINE is bipolar step function.

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

Training algorithm for MADALINE

1. Initialize the weight between input layer and hidden layer, hidden layer and output layer to some random value. Also set learning rate to random value.
2. Calculate net input to each hidden ~~layer~~ unit.

$$Z_{inj} = \sum x_i w_{ij} + b_j$$

3. Calculate output of each hidden unit using activation function.

$$Z_j = f(Z_{inj})$$

4. find the output of the network.

$$y_{in} = b_0 + \sum_{j=1}^{m} z_j v_j$$

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

5. (i) If target output = Calculated output then no weight updation is required.
   (ii) If target output (t) ≠ Calculated output (y), then update the weight and calculate new weights.

$$w_{ij}(new) = w_{ij}(old) + \alpha(t - Z_{inj})\, x_i$$
$$b_{ij}(new) = b_{ij}(old) + \alpha(t - Z_{inj})$$

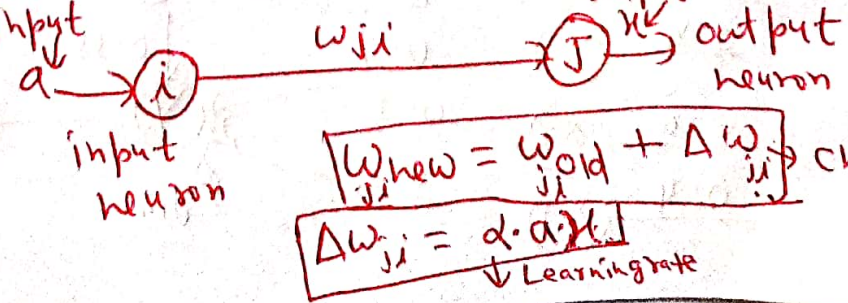
6. Repeat the process till there is no weight change or min. weight change.

Hebb's Network (Hebb's Rule)

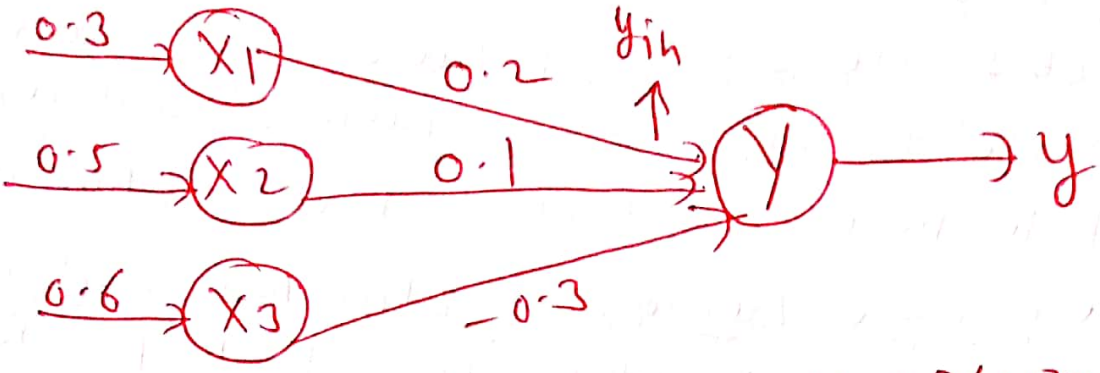
* This rule was proposed by Donald Hebb in 1949.
* Hebb's rule Say " If neuron i is near enough to excite neuron j and repeatedly participate in its activation, the Synaptic connection between these neurons is strengthen and neuron j becomes more sensitive to stimuli from neuron i ".
* Hebb's rule can be represented using two sub rules:
  (i) If two neurons on either side of the connection are activated synchronously, then the weight of that connection is increased.
  (ii) If two neurons on either side of the connection are activated asynchronously, then the weight of that connection is decreased.
* If two neurons have no relationship, then the weight will not change.
* If inputs of both the nodes are either positive or negative, then a positive weight exist between the nodes.
* If the input of a node is either positive or negative for others, a negative weight exist b/w the nodes.

weight update Hebb's formula

input = a
output = x
Learning = α rate



(1) For the given network, calculate the net input to the output neuron. (4)



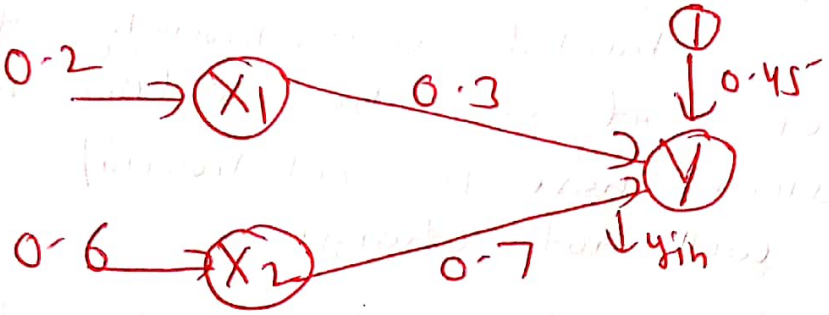
Net input $y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3$

$= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times (-0.3)$

$= \underline{-0.07}$

---

(2) Calculate the net input for the network, with bias included in the network.



Net input $y_{in} = x_1 w_1 + x_2 w_2 + b \times 1$

$= 0.2 \times 0.3 + 0.6 \times 0.7 + 0.45 \times 1$

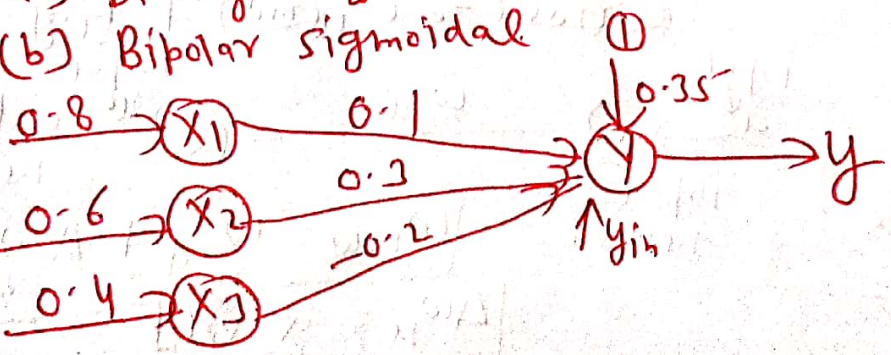
$= \underline{0.93}$

---

(3) Obtain the output of the neuron Y for the given network using activation function as

(a) Binary sigmoidal

(b) Bipolar sigmoidal



(5)

The net Input $y_{in} = b + x_1w_1 + x_2w_2 + x_3w_3$

$y_{in} = x_1w_1 + x_2w_2 + x_3w_3 + bx$

$y_{in} = 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2) + 0.35 \times 1$

$y_{in} = 0.53$

(a) for Binary sigmoidal activation function

$$\text{output}(Y) = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.53}}$$

$$\boxed{Y = 0.625}$$

(b) for Bipolar sigmoidal activation function

$$\text{output}(Y) = f(y_{in}) = \frac{2}{1+e^{-x}} - 1$$

$$\text{output}(Y) = \frac{2}{1+e^{-0.53}} - 1 = 0.259$$

$$\boxed{Y = 0.259}$$

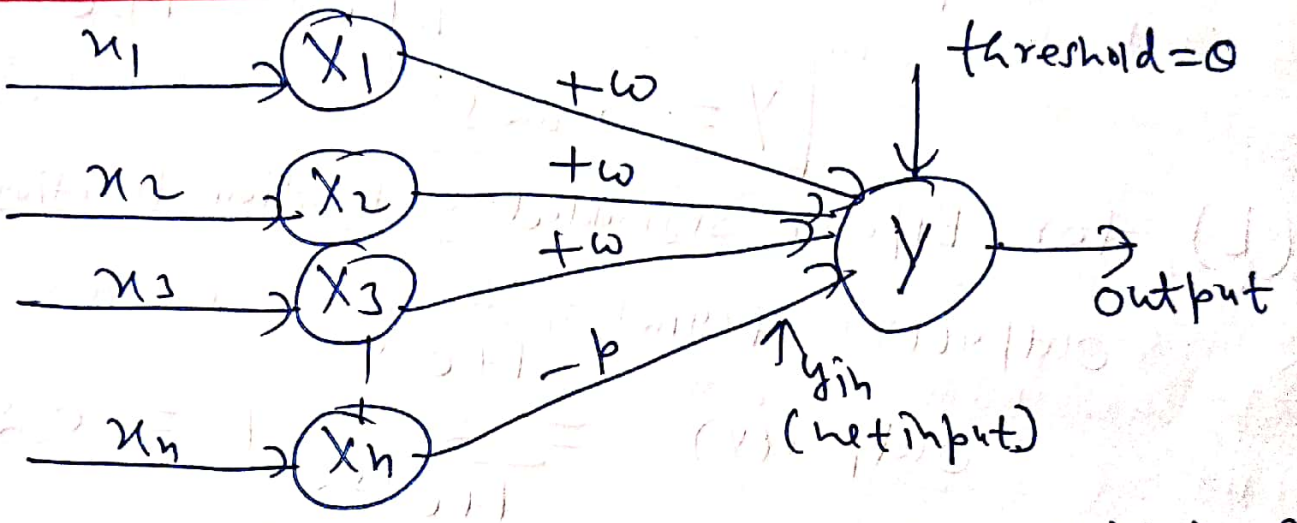
McCulloch Pitts Neuron :

1. It is also called MP- Neuron model.
2. It is the first computational model of a neuron proposed by Warren McCulloch and Walter Pitts in 1943.
3. This model allows binary 0 and 1 states only.
4. These binary neuron are connected by directed weighted path.
5. The connected path can have positive weights (Excitatory) or negative weights (Inhibitory).
6. The neuron is associated with the threshold value.

(7) The neuron activeates (fires), if the total inputs to the neuron is greater than or equal to ($\geq$) to the threshold value.

(8) The M-P neuron model has no particular training algorithm.

Architecture of MP-Neuron Model



Net input (y_{in}) = $x_1 \times w + x_2 \times w + x_3 \times w + \cdots - x_n \times (-p)$

output (Y) = $f(y_{in}) = 1$, if $y_{in} \geq \theta$
$= 0$, if $y_{in} < \theta$

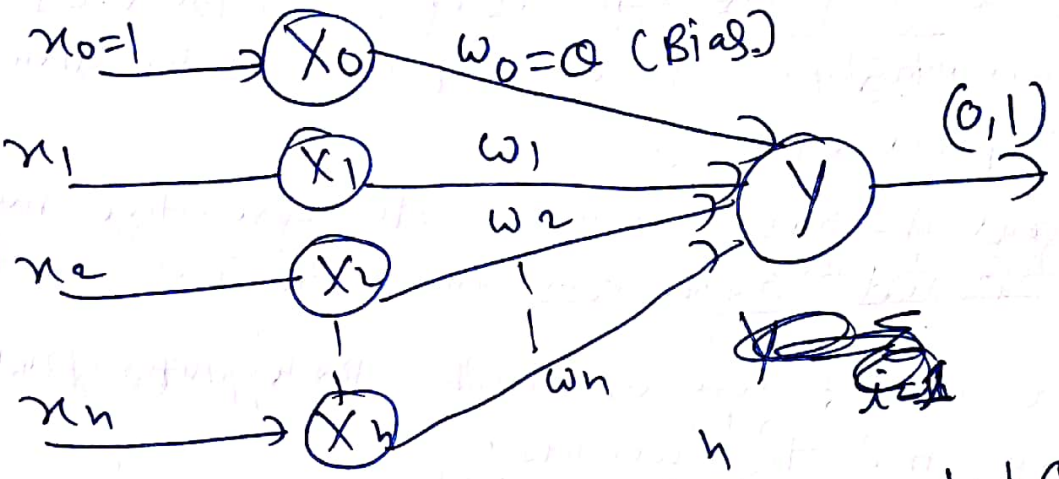
~~Rosenblatt~~ Rosenblatt's Perceptron Model

(1) Rosenblatt's perceptron was designed by Rosenblatt in 1958 to overcome most of the issues of the ~~neu~~ McCulloch-Pitts neuron.
   (a) It can process non-boolean inputs.
   (b) It can assign different weights to each input automatically.
   (c) The threshold θ is computed automatically.

(2) It is a single layer neural network.

(3) Rosenblatt's perceptron can be seen as a set of inputs that are weighted and to which we apply an activation function.

(4) We can represent the perceptron as



$$\text{output } (Y) = \sum_{i=1}^{n} w_i x_i + 1.\theta$$

$$Y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + \theta$$

(5) The inputs can be seen as neurons and will be called the input layer.

(6) These neurons (input layer) and the activation function form a perceptron.

(7) Activation function is used to calculate the output.

$$y = \begin{cases} 1 & , \quad Y \geq 0 \\ 0 & , \quad Y < 0 \end{cases}$$

(8) This model is considered as the first generation of neural network. It has main limitation of not being able to solve non-linear problems.

Auto Associative memory Network

(1) Auto Associative neural network have same value of training input and target output.

(2) The determination of the weights of the associative network is called storing of vectors.

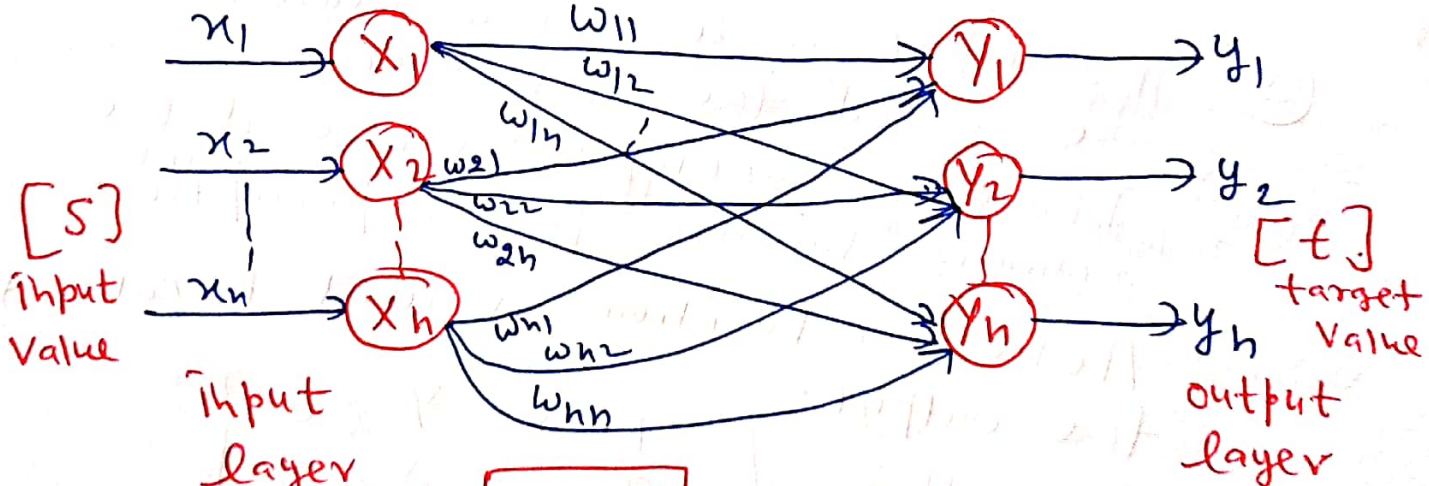
(3) It is capable of retriving a piece of data upon presentation of only partial information from that piece of data.

(4) Neural network using auto associative memory are called auto-associative network.

(5) It makes use of hebb's rule/outer product rule to find the weights.

(6) The input and output layers are connected through weighted connection.

(7) Architecture of Auto associative network



$s = t$

(8) Training algorithm for auto associative network

(a) initialize all weights to zero.

(b) for each of the vector that has to be stored, perform steps c to e.

(c) Activate each of input unit $x_i = s_i$ ($i = 1$ to n)

(d) Activate each of the output unit

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

(e) Adjust the weights

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j$$

or using outer product rule

$$W_{\text{new}} = [s]^T [t]$$

where T = Transpose of matrix

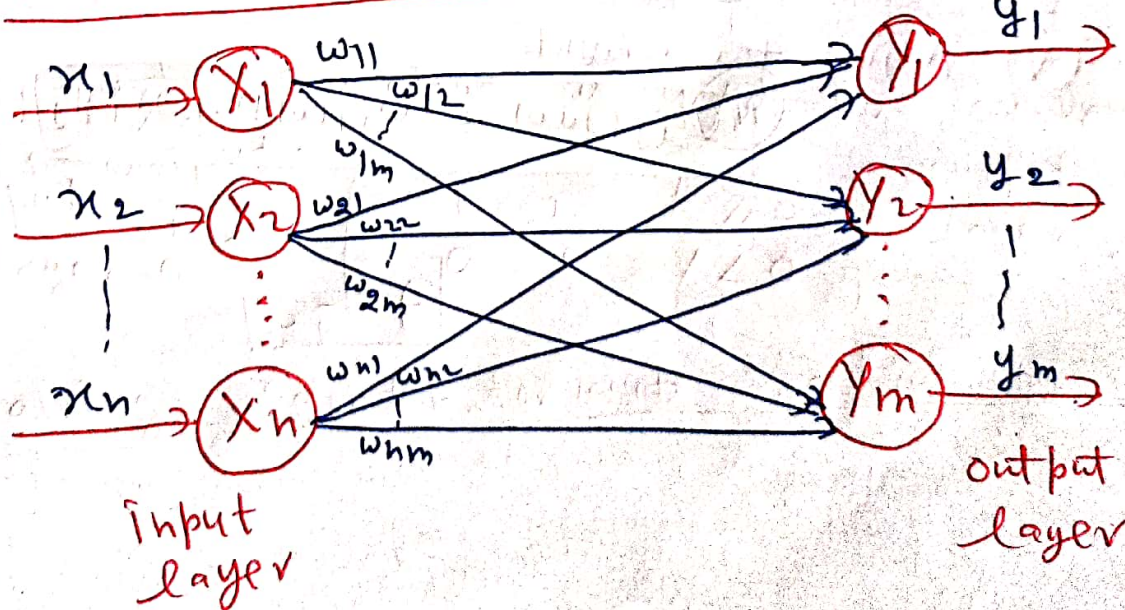
Hetero associative memory network:

(1) Hetero associative memory is capable of retrieving a piece of data of one category upon presentation of data from another category.

(2) Training input vector and target output vector are different.

(3) uses Hebb's rule/delta rule/outer product rule to find the weight matrix.

(4) Architecture of hetero-associative memory network



(5) The net finds an appropriate output vector which corresponds to an input vector x, that may be either one of the stored pattern or a new pattern.

(6) There exist weighted interconnections between the input and output layers.

(7) The input and output layer units are not correlated with each other.

(8) Hebb's ~~Interface~~ training algorithm for hetero-associative neural network.

(i) Set all the initial weights to zero.

$$w_{ij} = 0 \quad (i = 1 \text{ to } n,\ j = 1 \text{ to } m)$$

(ii) for each training target input output pair, perform steps (iii) to (iv)

(iii) Activate the input layer units to current training input

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

(iv) Activate the output layer units to current ~~target~~ output.

$$y_j = t_j \quad (\text{for } j = 1 \text{ to } m)$$

(v) Adjust the weights

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j \quad (\text{for } i = 1 \text{ to } n,\ j = 1 \text{ to } m)$$

Bi-directional Associative Memory (BAM) (11)

(1) It is a hetero associative recurrent neural network consisting of two layers
(i) X-layer (ii) Y-layer.
which are connected by means of directional weighted connection paths.

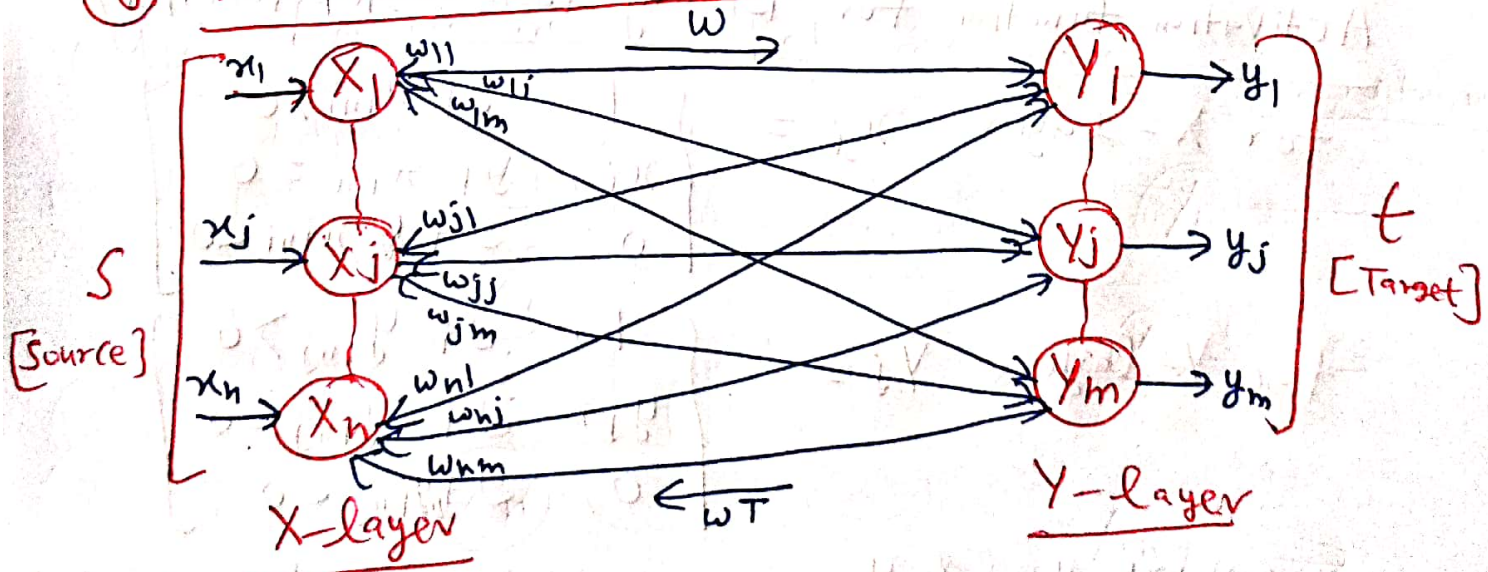
(2) This network iterates by sending a signal back and forth between the two layers until all neurons reach equilibrium.

(3) The network can respond to input on either layer.

(4) The weight matrix is calculated in both directions.

(5) If the weight matrix for signal sent from the X-layer to Y layer is W then weight matrix for signal sent from Y-layer to X-layer is W^T.

(6) Architecture of BAM



Types of BAM

(i) Discrete BAM (ii) Continuous BAM

(i) Discrete BAM

Discrete BAM has the same structure as of BAM. It is available in two forms.

(1) Binary form (0,1)
(2) Bipolar form (-1,1)

for Binary input vector, the weight matrix w_{ij} is

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p)-1][2t_j(p)-1]$$

where $s(p)$ = input vector
$t(p)$ = target vector
and $p = 1, 2, 3, \ldots P$

for Bipolar input vector

$$\text{weight matrix } w_{ij} = \sum_{p=1}^{P} s_i(p)\, t_j(p)$$

Activation function for Binary form (step function)

$$\text{for X-layer } x_i = \begin{cases} 1, & \text{if } x_{ini} > 0 \\ x_i, & \text{if } x_{ini} = 0 \\ 0, & \text{if } x_{ini} < 0 \end{cases}$$

$$\text{for Y-layer } y_j = \begin{cases} 1, & \text{if } y_{inj} > 0 \\ y_j, & \text{if } y_{inj} = 0 \\ 0, & \text{if } y_{inj} < 0 \end{cases}$$

where x_{ini} & y_{inj} are net input at X-layer and net input at Y-layer

for Bipolar inputs (Activation functions)

for x-layer

$$x_i = \begin{cases} 1, & \text{if } x_{ini} > \theta_i \\ x_i, & \text{if } x_{ini} = \theta_i \\ -1, & \text{if } x_{ini} < \theta_i \end{cases}$$

for y-layer

$$y_j = \begin{cases} 1, & \text{if } y_{inj} > \theta_j \\ y_j, & \text{if } y_{inj} = \theta_j \\ -1, & \text{if } y_{inj} < \theta_j \end{cases}$$

(ii) Continuous BAM

(1) A continuous BAM has the capability to transfer the input smoothly and continuously into the respective output.

(2) The continuous BAM uses logistic sigmoid activation function for all units.

(3) The weights are determined using hebb's rule

$$w_{ij} = \sum_{p=1}^{p} [2s_i(p)-1][2t_j(p)-1]$$

(4) Activation function for Binary logistic function

$$f(y_{inj}) = \frac{1}{1+e^{-y_{inj}}}$$

Activation function for Bipolar logistic function

$$f(y_{inj}) = \frac{2}{1+e^{-y_{inj}}} - 1 = \frac{1-e^{-y_{inj}}}{1+e^{-y_{inj}}}$$

where netinput $y_{inj} = \sum x_i w_{ij} + b_i$

using hebb's rule

Hopfield Network

(1)* John J. Hopfield proposed a network in 1982 which consists of a set of neurons with output of each neuron is connected as feedback to all other neurons.

(2)* Hopfield network used in many applications of associative memory and many optimization problem (Travelling Salesman problem).

(3) It is basically of two types
(i) Discrete hopfield network.
(ii) Continuous hopfield network.

(4) The hopfield network is an auto associative fully interconnected single layer feedback network.

(5) It is a symmetrically weighted network

(i) Discrete hopfield network:

(a) Hopfield network when operates in discrete line fashion, then it is called 'Discrete hopfield network'.

(b) This network has two (values inputs
(i) Binary (0,1) (ii) Bipolar (1,-1)

(c) The network has symmetrical weights with no self-connections

$$w_{ij} = w_{ji} \qquad w_{ij} = 0$$

(d) The architecture of discrete hopfield network model consists of two

One inverting and other non-inverting. (15)

(e) The output from each processing element are feedback to the input of the other processing elements but not to itself

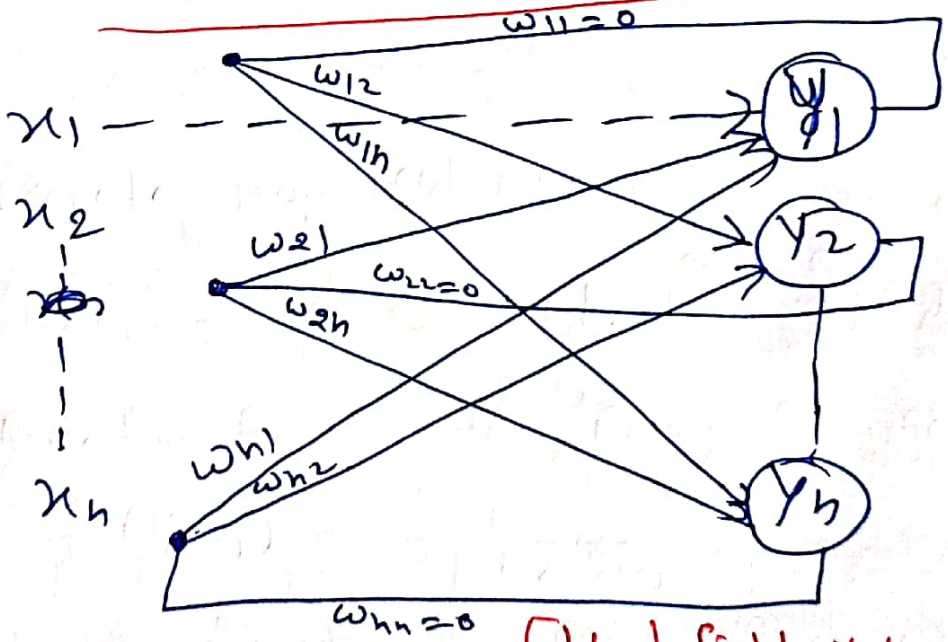
(f) for storing a set of ~~binary~~ binary pattern $S(p)$, where $p=1$ to P, The weight matrix is

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p)-1][2s_j(p)-1], \quad i \neq j$$

(g) for storing a set of bipolar pattern $S(p)$, the weight matrix is

$$w_{ij} = \sum_{p=1}^{P} s_i(p) \cdot s_j(p), \quad i \neq j$$

and weight for self Connection $w_{ij}=0$



[Hopfield Network Architecture]

Binary pattern is like $S(p) = [0\ 1\ 1\ 0\ 1]$ ($s_1\ s_2\ s_3\ s_4\ s_5$)

Bipolar pattern is like $S(p) = [1\ -1\ 1\ 1\ -1]$

(ii) Continuous Hopfield network

(a) A discrete hopfield network can be modified to a continuous model.

(b) The nodes of this network have a continuous output rather than a two state discrete output.

(c) The output value is between 0 to 1.

(d) The continuous hopfield network can be realized as an electronic circuit, which uses non-linear amplifier and resistors.

Self-organizing Maps (SOMs) (17)

(1) This technique was developed by T. Kohonen in 1982.

(2) SOMs are named as "self organizing" because no supervision is required.

(3) Self organizing maps learn on their own through unsupervised competitive learning.

(4) Self-organizing Maps are ~~one~~ neural network that uses unsupervised learning approach and trained its network through a competetive learning algorithm to map multidimensional data into lower-dimensional which allow easy interpretation of complex problems

(5) SOMs have two layers.
(i) input layer (ii) output layer.
There are n units in the input layer and m unit in the output layer.

(6) SOMs Training algorithm?

(i) Initialization: Choose random number for the initial weights w_{ij}

(ii) Sampling: Take a sampling training input vector $x = [x_1, x_2, \dots x_n]$ for the input layer.

(iii) Matching: find the winning neuron from the output layer that has weight vector closest to the input vector. It can be calculated by taking the square of Euclidean distance for each output unit and find the output unit that has minimum Euclidean distance from the input vector.

for each $j = 1$ to m

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

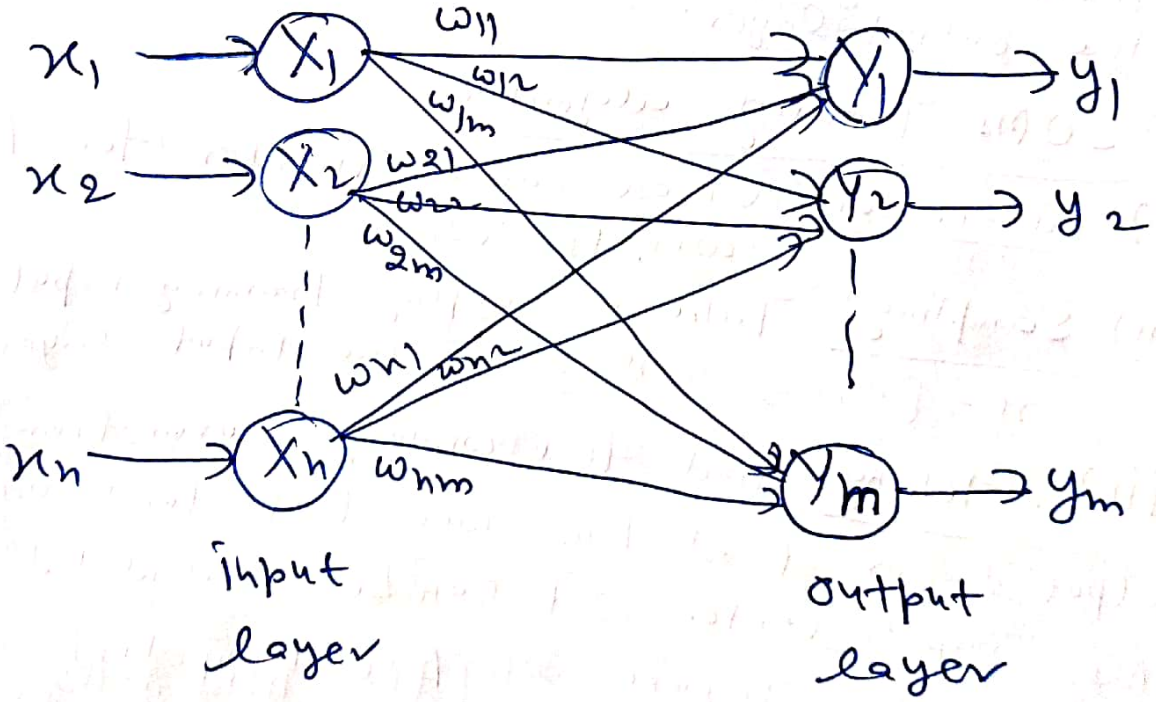
(IV) New weight Calculation find the new weights between input vector sample and winning output unit.

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

where α = learning rate

(V) Continuation: Repeat step-2 (ii) to (iv) until weight updation is negligible (i.e. new weights are similar to old weights) or feature map stop changing.

Architecture of self-organizing map



Supervised learning Vs Unsupervised learning (19)

Parameters	Supervised Learning	Unsupervised Learning
Definition	A Computer uses given label as example to take and sort series of data and thus to predict the future event. In supervised learning, people teach or train the machine using labeled data.	Unsupervised learning sorts data without using predefined labels The unsupervised machine learning algorithms acts without human guidance.
Input data	Uses known and labeled input data	Uses unknown input data
Computational Complexity	More Complex in Computation	Less Complex in Computation
Number of classes	No. of classes is known	No. of classes is not known
Real time	Uses off-line analysis of data	Uses real time analysis of data
Types	Two types of supervised learning (1) Classification (2) Regression	Two types of unsupervised learning (1) Clustering (2) Association

Applications of Neural Network
OR
Applications of Artificial Neural Network (ANN)

Ans (1) **Handwriting Recognition:** Neural networks are used to convert ~~handwritten~~ hand written characters into digital characters that a machine can recognize.

(2) **Stock exchange prediction:** The stock exchange is difficult to track and difficult to understand. Many factors affects the stock market. A neural network can examine a lot of factors and predict the price daily, which would ~~help~~ help stockbrokers

(3) **Image Compression:** The idea behind the data compression neural network is to store, encrypt and recreate the actual image again.

(4) **Speech Recognition:** Artificial Neural network plays a important role in speech recognition. following ANNs are used for speech recognition.

(i) Multilayer networks (ii) Multilayer networks with recurrent connections.
(iii) Kohonen's self-organizing feature map.

(5) **Human face Recognition:** Following neural networks are used for training purposes with preprocessed image

(a) fully-connected multilayer feedforward neural network trained with the help of backpropagation algorithm.

(b) for dimensionality reduction, Principal Component Analysis (PCA) is used.

(6) Air traffic Control Could be automated with the location, altitude, direction and speed using ANNs.

(7) Lake Water Levels: ANNs could be used to predict the lake water levels using precipitation patterns and river/dams flows.

(8) Criminal Sentencing: ANNs could be used to predict the sentencing of a criminal using a large sample of crime details as input and resulting sentences as output.

Delta Rule:

* The delta rule updates the weights between the connections so as to minimize the difference between the net input to the output unit and the target value.
* The main aim is to minimize the error over all training patterns.
* This is done by reducing the error for each pattern, one at a time.
* The delta rule for adjusting the weight of ith pattern (i=1 to n) is

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

for single output

where Δw_i = weight change
α = learning rate
x_i = vector of input unit.
y_{in} = The net input to output unit.

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

Delta rule for several output units

① **Learning Rate (α):** The learning rate is denoted by "α". It is used to control the amount of weight adjustment at each step of training. The learning rate, ranging from 0 to 1, determine the rate of learning at each time step.

② **Weight(s):** The weight contains information about the input signal. Each neuron in ANN is connected to other neurons by directed communication links, and each communication link is associated with weights. The weight information is used by the network to solve a problem. The weights can be represented in terms of matrix called Connection matrix.

③ **Bias:** The bias included in the network has its impact in calculating the net input. It is included by adding a component $x_0 = 1$ to the input vector X.

Bias can be positive or negative. The positive bias helps in increasing the net input of the network and negative bias helps in decreasing the net input of the network.

④ **Threshold:** Threshold is a set value based upon which the final output of the network may be calculated. The threshold value is used in the activation function. A comparison is made between the calculated net input and the threshold to obtain the network output. For each and every application, there is a threshold limit.

$$\text{e.g. } f(net) = \begin{cases} +1, & \text{if } net \geq \theta \\ -1, & \text{if } net < \theta \end{cases}$$

Difference between biological Neuron and Artificial Neuron

Parameter	Biological Neuron	Artificial Neuron
Speed	The execution time of biological neuron is of few millisec.	The execution time of artificial neuron is of few nanosec.
Processing	The biological neuron can perform massive parallel operations simultaneously.	The artificial neuron can also perform several parallel operation simultaneously but, in general, the artificial neuron is faster than brain.
Size and Complexity	The size and complexity of a biological neuron is more than that of artificial neuron.	The size and complexity of a artificial neuron is less than that of biological neuron
Storage Capacity	The biological neuron stores the information in its interconnections or in synapse strength. The biological neuron some times fails to retrieve the stored information.	The artificial neuron store the information in the contiguous memory location. The artificial neuron always retrieve the stored information.
fault tolerance	The biological neuron possesses fault tolerance Capability	The artificial neuron has no fault tolerance.