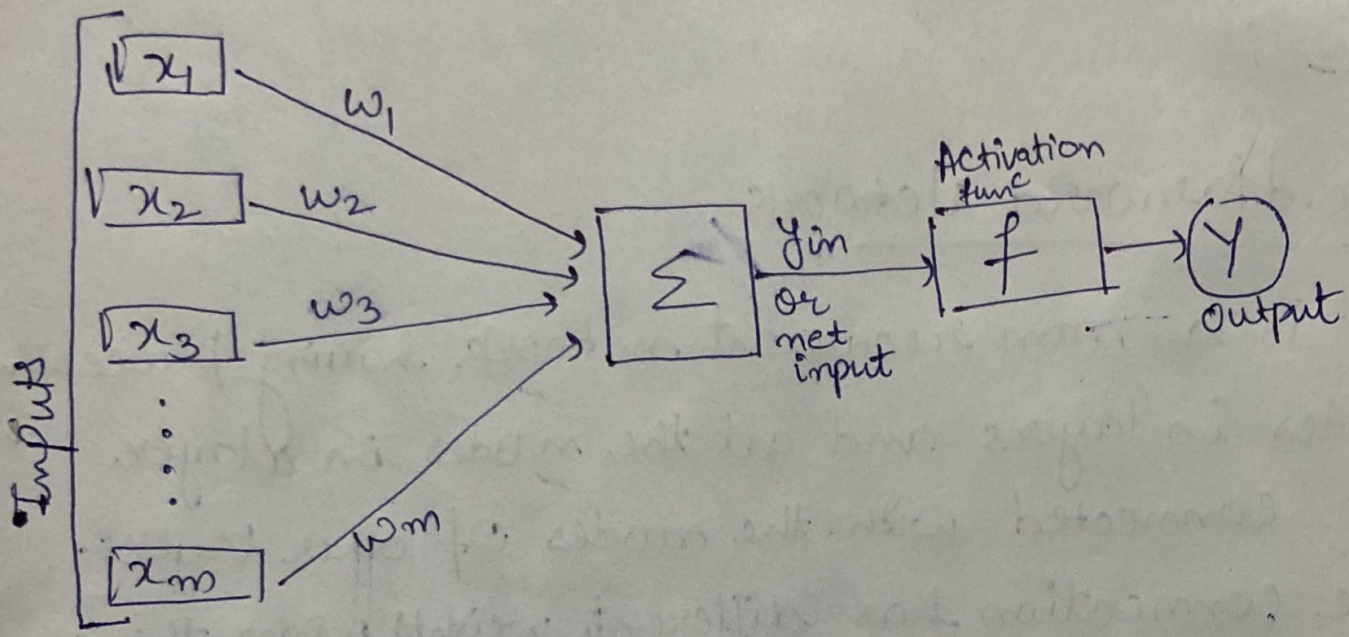
Unit-2

Model of ANN



The net input can be calculated as follows —

$$y_{in}(\text{net input}) = x_1w_1 + x_2w_2 + x_3w_3 + \ldots + x_mw_m$$

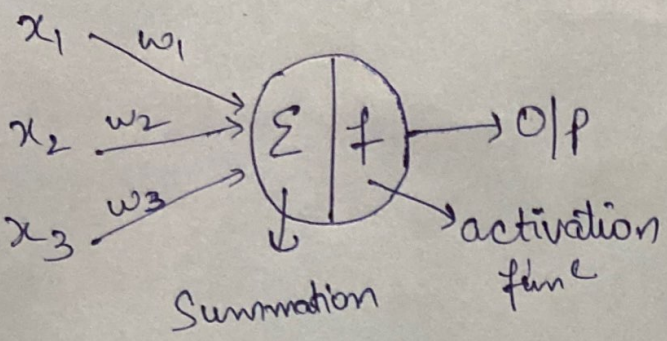
$$\text{i.e net input} = \sum_{i}^{m} x_i w_i$$

The output can be calculated by applying the activation function over the net input.

(actual o/p) $Y = F(y_{in})$

or

$= f(\text{net input})$

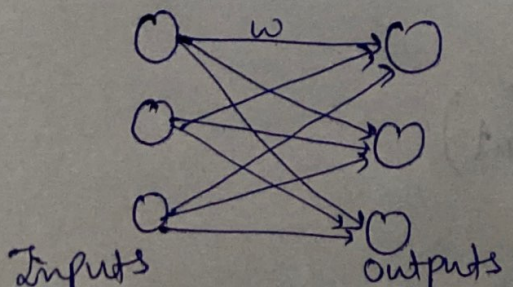


Feedforward Network

It is a non recurrent network having processing nodes in layers and all the nodes in a layer are connected with the nodes of other layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output.

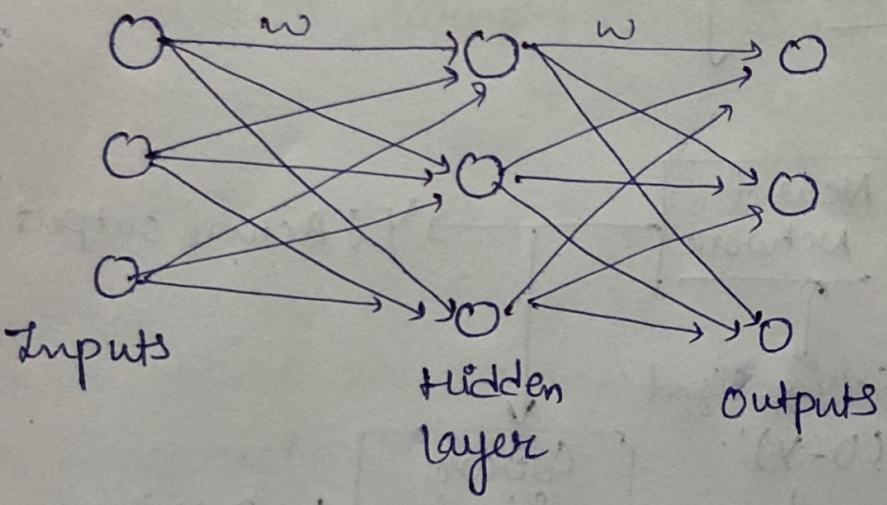
→ Single layer feed forward network

The concept is of feed forward ANN having only one weighted layer. There is only input layer & output layer. Input layer is fully connected to the output layer.



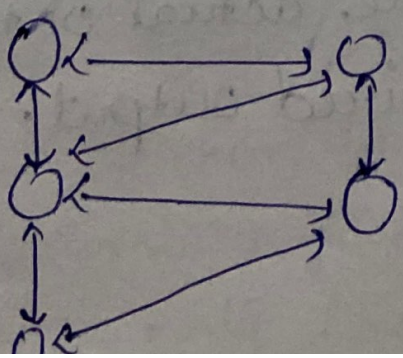
↳ Multilayer feedforward Network

There are more than one weighted layer in multilayer feedforward n/w. This network has one or more layers between the input and the output layer & ~~then~~ that layer is called hidden layer.



feedback Network

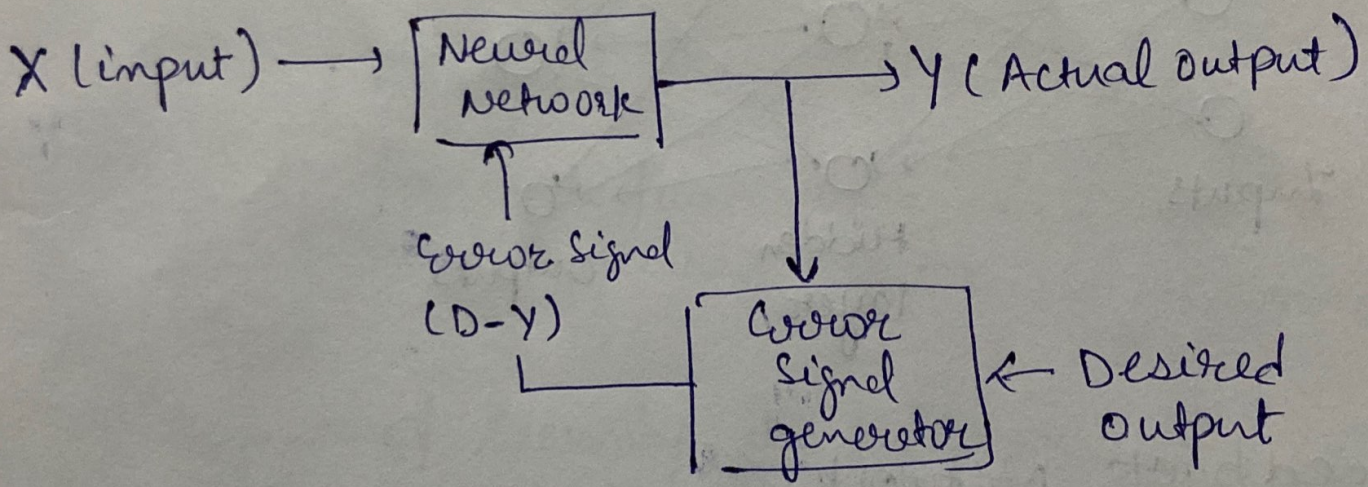
A feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a nonlinear dynamic system, which changes continuously until it reaches a state of equilibrium.



Learning

Learning in ANN, is the method of modifying the weights of connections b/w the neurons of a specified network. Learning can be classified into three categories.

① Supervised Learning

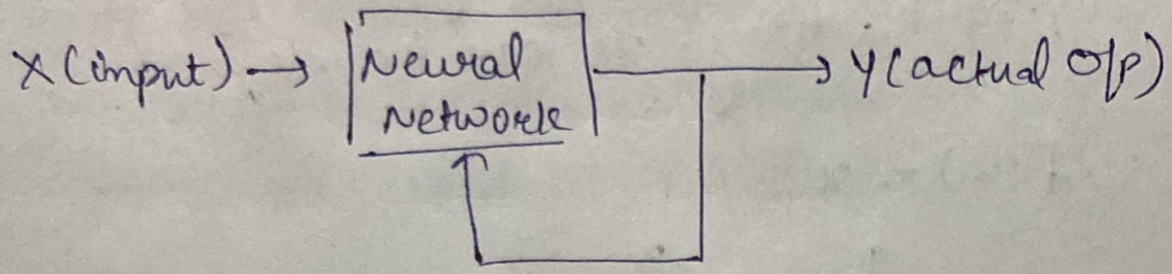


→ This learning is done under the supervision of a teacher.

→ This learning process is dependent.

→ On the basis of error signal, the weights are adjusted until the actual output is matched with the desired output.

Unsupervised learning



→ This type of learning is done without the supervision of a teacher.

→ This learning process is independent.

→ There is no feedback from the environment as to what should be the desired output & if it is correct or incorrect.

→ In this type of learning, the network itself must discover the patterns and features from the input data.

Activation Function

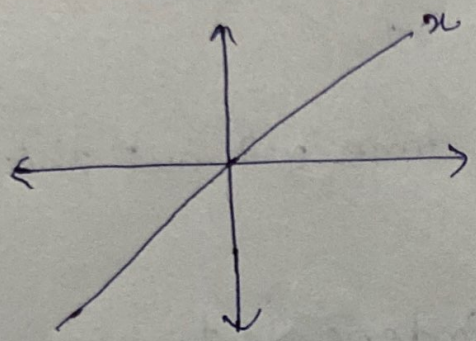
It may be defined as the extra force or effort applied over the input to obtain an exact output.

The purpose of an activation function is to add some kind of non linear property to the function without activation func, the neural network could perform only linear mappings from input X to output Y.

(1) Linear Activation Function

It is also called the identity function as it performs no input editing. It can be defined as -

$$F(x) = x$$

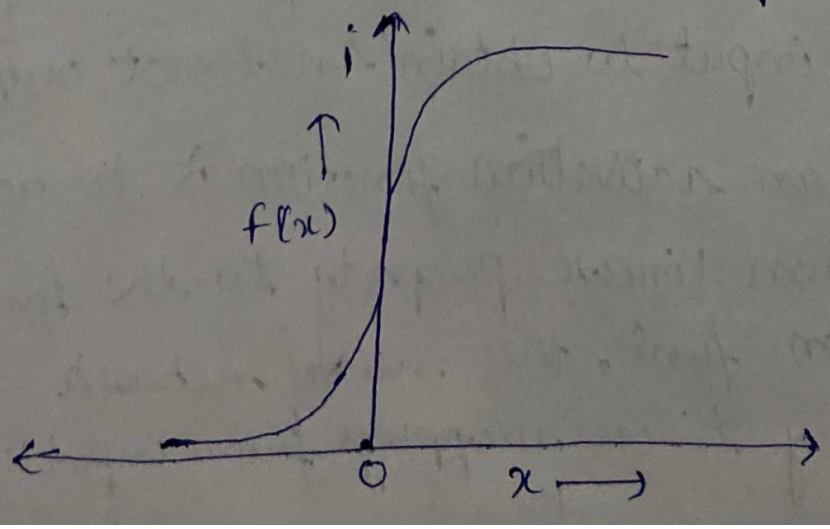


(2) Sigmoid Activation function

→ Binary Sigmoidel fun^c :- This activation fun^c performs input editing b/w 0 and 1.

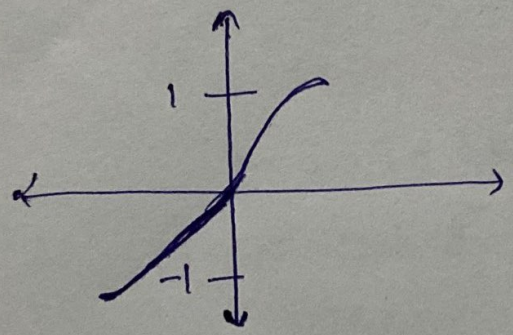
It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1.

$$F(x) = \text{Sigm}(x) = \frac{1}{1+\exp(-x)}$$



→ Bipolar sigmoidal function :- This activation fun^c performs input editing b/w −1 and 1. It can be positive or negative in nature. It is always bounded which means its output cannot be less than −1 and more than 1.

$$F(x) = \text{sign}(x) = \frac{2}{1+\exp^{-x}} - 1$$



→ RELU

→ Softmax

Ques Given a 2 input neuron with following parameters

Bias = 2.5 , weight $W = [3, 6]$

Input $X = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$

Calculate the neuron output for symmetric hard limiting function.

Soln

$$\text{Net input} = W * X + \text{Bias}$$

$$= [3 \quad 6] \begin{bmatrix} -3 \\ 5 \end{bmatrix} + 2.5$$

$$= [-9 + 30] + 2.5$$

$$= 23.5$$

$f(\text{net input}) = \underline{+1}$

because for hard limiting function

$$f(\text{net input}) = \begin{cases} +1 & \text{if net} > 0 \\ -1 & \text{if net} < 0 \end{cases}$$

Learning Rules

→ Hebbian learning rule
→ Perceptron learning rule
→ Delta Learning rule
→ Widrow Hoff learning rule
→ Winner take all learning rule

Hebbian Learning Rule

→ It is unsupervised learning rule

→ It is simple & requires a very little computation.

→ Developed by Donald Hebb in 1949.

→ Donald Hebb wrote: When the axon of cell A is near enough to excite a cell B & repeatedly firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

→ In Hebbian learning, weights between learning nodes are adjusted so that each weight better represents the relationship b/w the nodes. Nodes which tend to be positive or negative at the same time will have strong positive weights while those which tends to be opposite will have strong negative weights.

Steps of Hebb. learning Rule

1. Find net input (net)

$$net' = weight' \times input'(x_i)$$

2. Apply activation function, In Hebb it is

$$Sgn(net')$$

3. update weight

$$w(new) = w(old) + C r x$$

where c & x is learning rule

c = Constant value (given else take 1)

r = sgn(net)

x = input

$$w^2 = w^1 + c * sgn(net^1) * x_1$$

Ques Find the weights using Hebb learning rule

$$w^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

Input vectors are

$$x_1 = \begin{bmatrix} 1 \\ -1 \\ 1.5 \\ 0 \end{bmatrix} \quad x_2 = \begin{bmatrix} 1 \\ -0.5 \\ -1 \\ 0 \end{bmatrix} \quad x_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

Soln **Step 1:**

1. Find net input 1 or net 1

$$net\,1 = w^1 \times x_1$$

$$= \begin{bmatrix} 1 & -1 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1.5 \\ 0 \end{bmatrix}$$

$$= 2$$

2. $sgn(net^1) = sgn(2)$

$$= +1$$

3. update weight i.e w^2

$$w^2 = w^1 + c\,r\,x_1$$

$$= \begin{bmatrix} +1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 1.5 \\ 0.5 \end{bmatrix}$$

Step 2.

$net^2 = w^2 \times x_2$ where T is transpose of a matrix

$$= [2 \quad -2 \quad 1.5 \quad 0.5] \begin{bmatrix} 1 \\ -0.5 \\ -1 \\ 0 \end{bmatrix}$$

$$= [2 + 1 \quad -1.5 + 0] = 1.5$$

$Sgn(net^2) = Sgn(1.5) = +1$

update weight $w^3 = w^2 + c \, r \, x_2$

$$= \begin{bmatrix} 2 \\ -2 \\ 1.5 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ -0.5 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ -2.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

Step 3.

$net^3 = w^3 \times x_3$

$$= [3 \quad -2.5 \quad 0.5 \quad 0.5] \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

$= -2.25$

$Sgn(net^3) = Sgn(-2.25) = -1$

update weight $w^4 = w^3 + c \, r \, x_3$

$$= \begin{bmatrix} 3 \\ -2.5 \\ 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 3 \\ -3.5 \\ 1.5 \\ -1 \end{bmatrix}$$

Perceptron learning rule

→ Perceptron are trained on examples of desired behavior.

→ This rule is for supervised learning mode.

→ The desired behavior can be summarized by a set of input, output pairs & 'd' is the corresponding correct (target) output. The objective is to reduce the error 'e' which is the difference between the actual output and the desired output.

Steps for perceptron learning

1. Find net input (net') = weight X input (x_1)
2. Apply Activation function

   Actual Output (O_1) = Sgn(net$_1$)

3. Check whether Actual output (O_1) is equal to desired Output(d_1), if they are equal don't update the weight (i.e $w_2 = w_1$) or if they are not equal then update the weight (i.e find w_2)

   update weight (new weight w_2) = Old weight + C R X

   where R = d − O

   desired output − actual output

Ques Let $w_0 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$, $d_0 = -1$, $d_1 = -1$, $d_2 = 1$, $c = 0.1$

where d_0, d_1, d_2 are teacher's signal and $x_0 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$, $x_1 = \begin{bmatrix} 0 \\ 1 \\ -0.5 \\ -1.0 \end{bmatrix}$, $x_2 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$. Apply Perceptron learning and find weights.

Soln **Step1.**

1. Find net input $(net^0) = w_0 x_0$

$$= \begin{bmatrix} 1 & -1 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= 2.5$$

2. Find Actual output $(O^0) = Sgn(net^0)$

$$= Sgn(2.5) = +1$$

3. Check whether $O^0 = d_0$ or not, Here

$$O^0 \neq d_0 \quad (1 \neq -1)$$

update weight (i.e w_1) where $r = (-1 - 1)^{d_0 - O^0}$

$$w_1 = w_0 + c\, r\, x_0$$

$$= \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} - 0.2 \begin{bmatrix} 1.0 \\ -2.0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

Step2.

1. $net^1 = w_1 x_1$

$$= \begin{bmatrix} 0.8 & -0.6 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ -0.5 \\ -1 \end{bmatrix} = -1.6$$

2. $o^1 = sgn(net^1)$
$= sgn(-1.6)$
$= -1$

3. Since $o^1 = d_1$ $(-1 = -1)$, no weight update is required.

$\therefore w_2 = w_1 = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$

Step 3.

1. Find $net^2 = w_2 x_2$
$= [0.8 \ -0.6 \ 0 \ 0.7] \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} = -2.1$

2. Find $o^2 = sgn(net^2)$
$= sgn(-2.1) = -1$

3. $o^2 \neq d_2$ $(-1 \neq 1)$

$w_3 = w_2 + 0.1(1+1)x_2$
$= \begin{bmatrix} 0.3 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix} + 0.2 \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$

Delta learning Rule

→ It is only valid for continuous activation function
→ It is based on supervised learning.
→ This rule is derived from LMS (Least Mean Squared)

→ The value of r (learning signal) is.

$$r = [d_i - f(wx)] f'(wx)$$

→ This rule is also called continous perceptron training rule:

Derivation

we know ~~Cost~~ Error (E)

$$E = \frac{1}{2}(d_i - O_i)^2$$ where d_i = desired o/p, O_i = actual o/p

$$E = \frac{1}{2}\left[d_i - \underbrace{f(w_i x)}_{\text{Actual o/p}}\right]^2$$

· Error gradient vector value

$$\nabla E = -\left[d_i - f(w_i x)\right] f'(w_i x)\, x$$

Derivative of E w.r.t w (weight) or component of gradient vector

$$\frac{\partial E}{\partial w_{ij}} = -\left[d_i - f(w_i x)\right] f'(w_i x)\, x$$

Take $\Delta w_i = -\eta \nabla E$ where η is constant

$$\Delta w_i = \eta \left[d_i - f(w_i x)\right] f'(w_i x)\, x$$

$$\Rightarrow \boxed{\Delta w_{ij} = \eta (d_i - o_i) f'(net_i) x_j}$$

Steps of Delta learning rule

1. Find net input $(net') = w^1 x_1$
2. Apply Activation fun^c. (continuous Activation fun^c)

$$o_1 = f(net') = \left[\frac{2}{1 + \exp(-\lambda net)}\right] - 1$$

(o_1 → Actual o/p)

3. update weight (w^2)

$$w^2 = \text{old weight}(w^1) + c r x_1$$
$$= w^1 + c \underbrace{[d_1 - f(net')] f'(net) \cdot x_1}_{r}$$

$$f'(net) = (1 - o^2)/2$$

Numerical

Ques Let $w^1 = [1 \quad -1 \quad 0 \quad 0.5]$ and input vectors $x_1 = \begin{bmatrix} 1 \\ -1 \\ 1.5 \\ 0 \end{bmatrix}$, $x_2 = \begin{bmatrix} 1 \\ -0.5 \\ -1 \\ 0 \end{bmatrix}$, $x_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$, $c = 0.1$

$\lambda = 1$ for bipolar continuous activation function ; use Delta learning rule to find out the final weight adjustment. use $f'(net) = (1-o^2)/2$.
Take $d_1 = -1.0$, $d_2 = -1$, $d_3 = 1$.

Soln 1. $net' = w^1 x_1$

$$= [1, -1, 0, 0.5] \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= 2.5$$

2. $$o_1 = f(net') = \left[\frac{2}{1 + \exp(-\lambda net)}\right] - 1$$

$\lambda = 1$ (given)

$$= \frac{2}{1 + \exp(-1 \times 2.5)} - 1 = 0.85$$

3. update weight $(w^2) = w^1 + c[d_1 - f(net')] f'(net') x_1$

$$f'(net') = (1 - o^2)/2 = \frac{1 - (0.85)^2}{2} = 0.140$$

$$w^2 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + 0.1[-1 - 0.85] \times 0.140 \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.974 \\ -0.948 \\ 0 \\ 0.526 \end{bmatrix}$$

Step2. $net^2 = w^2 x_2$

$$= [0.974 \quad -0.948 \quad 0 \quad 0.526] \begin{bmatrix} 0 \\ 1.5 \\ -1.5 \\ -1 \end{bmatrix}$$

$$= -1.948$$

$$O^2 = f(net^2) = \left[\frac{2}{[1+\exp(-\lambda net)]}\right] - 1$$

$$= -0.75$$

$$f'(net^2) = \frac{(1-O^2)}{2} = \frac{(1-(-0.75)^2)}{2} = 0.218$$

update weight $(w^3) = w^2 + C[d_2 - f(net^2)] f'(net^2) x_2$

$$w^3 = \begin{bmatrix} 0.974 \\ -0.89 \\ 0.102 \\ 0.487 \end{bmatrix}$$