Hebbian Learning

Hebbian Learning Rule, also known as Hebb Learning Rule, was proposed by **Donald O Hebb**. It is one of the first and also easiest learning rules in the neural network. It is used for pattern classification. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n. The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.

Hebbian Learning Rule Algorithm :

1. Set all **weights to zero**, $w_i = 0$ for i=1 to n, and **bias to zero**.
2. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5.
3. Set activations for input units with the input vector $X_i = S_i$ for i = 1 to n.
4. Set the corresponding output value to the output neuron, i.e. y = t.
5. Update weight and bias by applying Hebb rule for all i = 1 to n:

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

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

Implementing AND Gate :

INPUT			TARGET	
x_1	x_2	b		y
-1	-1	1	Y_1	-1
-1	1	1	Y_2	-1
1	-1	1	Y_3	-1
1	1	1	Y_4	1

There are 4 training samples, so there will be 4 iterations. Also, the activation function used here is Bipolar Sigmoidal Function so the range is [-1,1].

Step 1 :

Set weight and bias to zero, $w = [\ 0\ 0\ 0\ ]^T$ and $b = 0$.

Step 2 :

Set input vector $X_i = S_i$ for $i = 1$ to 4.

$X_1 = [\ -1\ -1\ 1\ ]^T$

$X_2 = [\ -1\ 1\ 1\ ]^T$

$X_3 = [\ 1\ -1\ 1\ ]^T$

$X_4 = [\ 1\ 1\ 1\ ]^T$

Step 3 :

Output value is set to $y = t$.

Step 4 :

Modifying weights using Hebbian Rule:

First iteration –

$w(new) = w(old) + x_1y_1 = [\ 0\ 0\ 0\ ]^T + [\ -1\ -1\ 1\ ]^T . [\ -1\ ] = [\ 1\ 1\ -1\ ]^T$

For the second iteration, the final weight of the first one will be used and so on.

Second iteration –

$w(new) = [\ 1\ 1\ -1\ ]^T + [\ -1\ 1\ 1\ ]^T . [\ -1\ ] = [\ 2\ 0\ -2\ ]^T$

Third iteration –

$w(new) = [\ 2\ 0\ -2]^T + [\ 1\ -1\ 1\ ]^T . [\ -1\ ] = [\ 1\ 1\ -3\ ]^T$

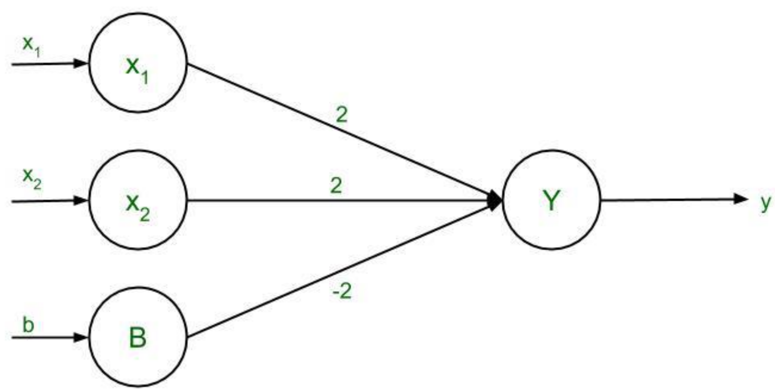
Fourth iteration –

w(new) = [1 1 -3]^T + [1 1 1]^T . [1] = [2 2 -2]^T

So, the final weight matrix is [2 2 -2]^T

Input			Target	Results		
X1	X2	B	Y	W1	W2	B
-1	-1	1	-1	1	1	-1
-1	1	1	-1	2	0	-2
1	-1	1	-1	1	1	-3
1	1	1	1	2	2	-2

Testing the network :



For x_1 = -1, x_2 = -1, b = 1, Y = (-1)(2) + (-1)(2) + (1)(-2) = -6

For x_1 = -1, x_2 = 1, b = 1, Y = (-1)(2) + (1)(2) + (1)(-2) = -2

For x_1 = 1, x_2 = -1, b = 1, Y = (1)(2) + (-1)(2) + (1)(-2) = -2

For x_1 = 1, x_2 = 1, b = 1, Y = (1)(2) + (1)(2) + (1)(-2) = 2

The results are all compatible with the original table