Learning and Adaptation

The most impressive characteristic of the human brain is to learn, hence the same feature is acquired by ANN.

What Is Learning in ANN?

Basically, learning means to do and adapt the change in itself as and when there is a change in environment. ANN is a complex system or more precisely we can say that it is a complex adaptive system, which can change its internal structure based on the information passing through it.

Why Is It important?

Being a complex adaptive system, learning in ANN implies that a processing unit is capable of changing its input/output behavior due to the change in environment. The importance of learning in ANN increases because of the fixed activation function as well as the input/output vector, when a particular network is constructed. Now to change the input/output behavior, we need to adjust the weights.

Neural Network Learning Rules

We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights. Hence, a method is required with the help of which the weights can be modified. These methods are called Learning rules, which are simply algorithms or equations. Following are some learning rules for the neural network −

1- Hebbian Learning Rule
2- Perceptron Learning Rule
3- Delta Learning Rule
4- Competitive Learning Rule
5- Outstar Learning Rule

Hebbian Learning Rule

This rule, one of the oldest and simplest, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949. It is a kind of feed-forward, unsupervised learning

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, wi = 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 Xi = Si 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:

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

$$b\,(\text{new}) = b\,(\text{old}) + y$$

Implementing AND Gate:

Inputs			Target
x1	x2	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

we have used **‘-1'** instead of **‘0’** this is because the Hebb network uses bipolar data and not binary data because the product item in the above equations would give the output as **0** which leads to a wrong calculation.

* $w_1 = 0 \quad w_2 = 0 \quad b = 0$

↳ first i/p $[x_1, x_2, b] = [1, 1, 1]$ and $y = 1$

$w_1 new = w_1 old + x_1 * y$
$= 0 + (1 * 1) = 1$

$w_2 new = w_2 old + x_2 * y$
$= 0 + (1 * 1) = 1$

$b new = b old + y = 0 + 1 = 1$

$\boxed{\therefore w_1 = 1 \quad w_2 = 1 \quad b = 1}$

↳ Second i/p $[x_1, x_2, b] = [1, -1, 1]$ & $y = -1$

$w_1 new = w_1 old + x_1 * y$
$= 1 + (1 * -1) = 0$

$w_2 new = w_2 old + x_2 * y$
$= 1 + (-1 * -1) = 2$

$b new = b old + y = 1 + (-1) = 0$

$\boxed{\therefore w_1 = 0, \ w_2 = 2, \ b = 0}$

↳ 3rd i/p $[x_1, x_2, b] = [-1, 1, 1]$ & $y = -1$

$w_1 new = w_1 old + x_1 * y$
$= 0 + (-1 * -1) = 1$

$w_2 new = w_2 old + x_2 * y$
$= 2 + (1 * -1) = 1$

$b new = b old + y = 0 + (-1) = -1$

$\therefore \boxed{w_1 = 1 \ . \ w_2 = 1 \ . \ b = -1}$

↳ 4th i/p = [x_1, x_2, b] = [-1, -1, 1] & y = -1

$w_1 new = w_1 old + x_1 * y$

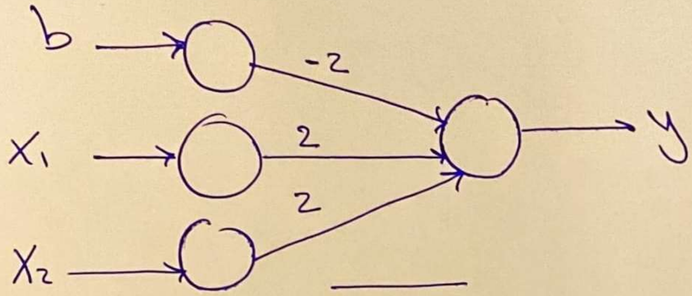
$= 1 + (-1 * -1) = 2$

$w_2 new = w_2 old + x_2 * y$

$= 1 + (-1 * -1) = 2$

$b_{new} = b_{old} + y = -1 + (-1) = -2$

∴ $\boxed{w_1 = 2, \; w_2 = 2, \; b = -2}$



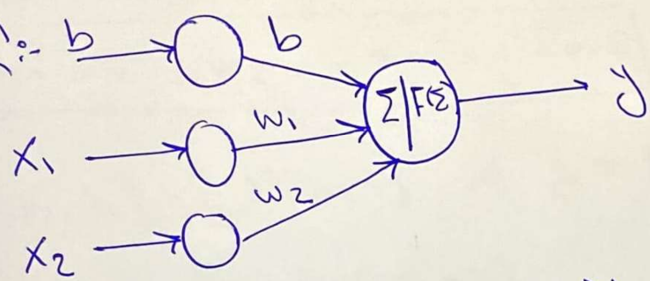
Test the network:-

↳ 2*1 + 2*1 + (-2*1) = 2 → 1
↳ 2*1 + 2*-1 + (-2*1) = -2 → -1
↳ 2*-1 + 2*1 + (-2* 1) = -2 → -1
↳ 2*-1 + 2*-1 + (-2*1) = -6 → -1

} which is the actual o/p of AND gate.

ex:- Using the Hebb network, Solve the OR gate problem, and show the results of the weights and bias, noting that the learning rate $\eta = 0.5$ and the initial weight are: $w_1 = 0$, $w_2 = 0$, $b = -1$. and the activation function $F(\Sigma) = 1$ if $\Sigma \geq 1$ and 0 otherwise.

Solution:-



X_1	X_2	y
1	1	1
1	0	1
0	1	1
0	0	0

* Random initial weights & Bias:-

$$\boxed{w_1 = 0 \quad w_2 = 0 \quad b = -1} \Longrightarrow \underline{\underline{\text{given}}}$$

① $[x_1, x_2, b] = [1, 1, 1]$ & $y = 1$

$w_1 \text{ new} = w_1 \text{ old} + n(x_1 * y)$

$= 0 + 0.5(1 * 1) = 0.5$

$w_2 \text{ new} = w_2 \text{ old} + n(x_2 * y)$

$= 0 + 0.5(1 * 1) = 0.5$

$b\text{new} = b\text{old} + y * n = -1 * (1 * 0.5) = -0.5$

$$\boxed{\therefore w_1 = 0.5, \; w_2 = 0.5 \quad b = -0.5}$$

② $[x_1, x_2, b] = [1, 0, 1]$ & $y = 1$

$w_1 \text{ new} = w_1 \text{old} + n(x_1 * y)$

$= 0.5 + 0.5(1 * 1) = 1$

$w_2 \text{ new} = w_2 \text{ old} + n(x_2 * y)$

$= 0.5 + 0.5(0 * 1) = 0.5$

$b\text{new} = b\text{old} + y * n = -0.5 + (1 * 0.5) = 0$

$$\boxed{\therefore w_1 = 1 \quad w_2 = 0.5 \quad b = 0}$$

③ $[x_1, x_2, b] = [0, 1, 1]$ & $y = 1$

$w_1 new = w_1 old + n(x_1 * y)$

$= 1 + 0.5(0 * 1) = 1$

$w_2 new = w_2 old + n(x_2 * y)$

$= 0.5 + 0.5(1 * 1) = 1$

$b_{new} = b_{old} + y * n = 0 + 1 * 0.5 = 0.5$

$\therefore \boxed{w_1 = 1 \quad w_2 = 1 \quad b = 0.5}$

④ $[x_1, x_2, b] = [0, 0, 1]$ & $y = 0$

$w_1 new = w_1 old + n(x_1 * y)$

$= 1 + 0.5(0 * 0) = 1$

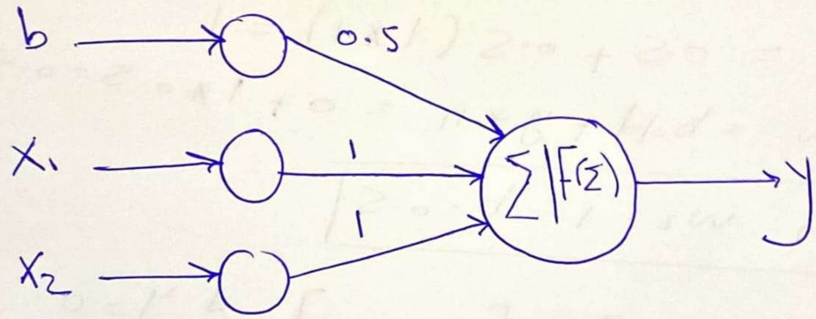
$w_2 new = w_2 old + n(x_2 * y)$

$= 1 + 0.5(0 * 0) = 1$

$b_{new} = b_{old} + y * n = 0.5 + (0 * 0.5) = 0.5$

$\therefore \boxed{w_1 = 1, \quad w_2 = 1 \quad b = 0.5}$

then test the network.



b
0.5
x1
1
1
x2
Σ|F(Σ)
y