Soft Computing

Semester = 6(vi) Paper code :- DSE-3

Module - III (Neural Networks)

Unit - 2 (Learning Methods)

Developed by : Kunal Kumar Mandal
Assistant Professor
Dept. of computer Science.

Hebbian Learning Rule

=> Donald O Hebb (1949) stated that in brain, the learning is performed by the change in the symantic gap.

The function of biological neurons contributed to learning. Theory of Hebb learning in "The organization of behaviour.

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some grouth 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."

* As we know that updation of weight matrix or updation of weights in artificial neural network is know as learning. It is a feed forward and unsupervised learning rule. Hebb's learning rule works on bipolar data. In hebb learning weights are changed according to the principle "Neurons which fire together, wire together."

* It is used in pattern association, pattern recorganization, or pattern classification problem.

Example:- Implement Logic 'AND' operation using Hebbian Learning Mechanism.

Ans:- Binary input and output represented as = 0 & 1, Hebb's Learning rule works on bipolar data. Bipolar input and output represented as = -1 & 1, so replace '0' by '-1' in truth table of AND ~~eg~~ gate.

Truth table of AND gate

Input		Output
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

Bipolar Input			Target output	New w1 (w1)	New w2 (w2)	New bias &
x1	x2	bias, b	y			
-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

Step 1:- Initialize weights and bias with zero,

So, $w_1 = w_2 = b = 0$

Step 2:- Update weights and bias using Hebb learning rule,

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

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

Here, x is input value and y is target output

From truth table, $x_1 = -1$, $x_2 = -1$ and $y = -1$ (case 1)

$\therefore w_1(new) = w_1(old) + (x_1 \cdot y)$

$\therefore w_1(new) = 0 + ((-1).(-1))$

$= 1$

$w_2(new) = w_2(old) + (x_2 \cdot y)$

$= 0 + ((-1)(-1))$

$= 1$

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

$= 0 + (-1)$

$= -1$ (put the 3 value in the table)

Case - 2 from the truth table,

$x_1 = -1$, $x_2 = 1$ and $y = -1$

$w_1(new) = w_1(old) + x_1 \cdot y$

$= 1 + ((-1) \cdot (-1))$

$= 2$

[Now, $w_1(old) = 1$, $w_2(old) = 1$, $b(old) = -1$ See the table first row]

$w_2(new) = w_2(old) + x_2 \cdot y$

$= 1 + (1 \cdot (-1))$

$= 0$

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

$= (-1) + (-1)$

$= -2$

Case - 3

from the truth table

$x_1 = 1$, $x_2 = -1$, and $y = -1$

[now, $w_1(old) = 2$, $w_2(old) = 0$, $b(old) = -2$]

same way,

$w_1 = 2 + (1 \cdot (-1)) = 1$

$w_2 = 0 + ((-1)(-1)) = 1$

$b = -2 + (-1) = -3$

Case - 4 from the truth table,

$x_1 = 1$, $x_2 = 1$ and $y = 1$

[now, $w_1(old) = 1$, $w_2(old) = 1$, $b(old) = -3$]

in the same way,

$w_1 = 1 + 1.1 = 2$

$w_2 = 1 + 1.1 = 2$

$b = (-3) + 1 = -2$

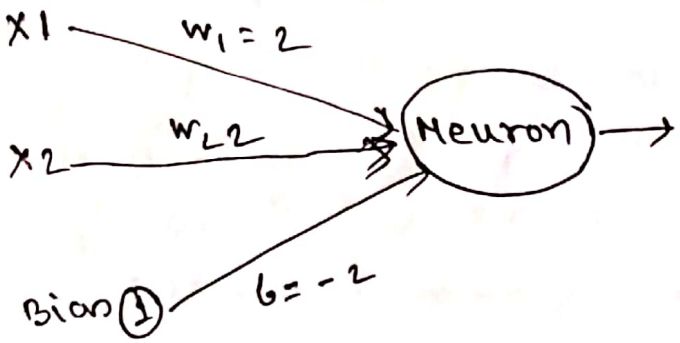
∴ Final value of weights and bias are,

$w_1 = 2$

$w_2 = 2$

$b = -2$

So, draw the AND Gate Neuron :



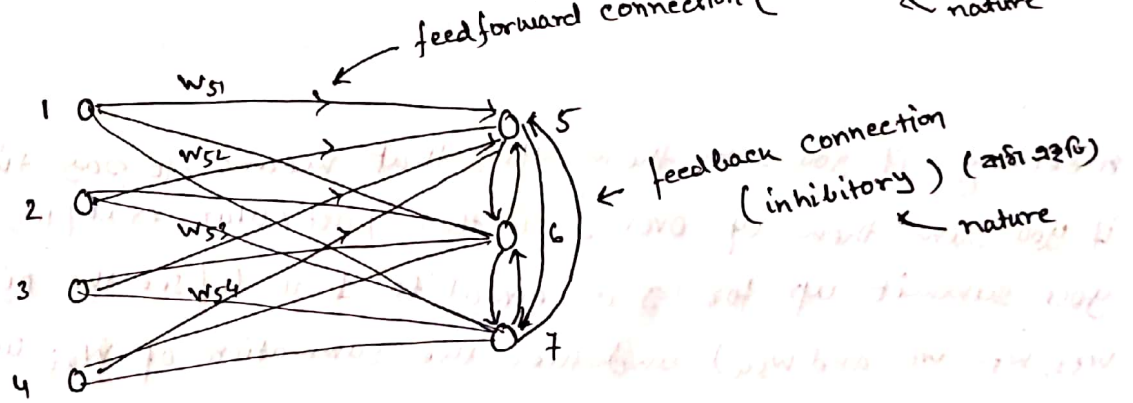
For verification :-

X1	X2	$w_1 = 2$, $w_2 = 2$ and bias $= -2$ $y = \sum w_i x_i + b$	Remark	output(y)
-1	-1	$y = 2.(-1) + 2.(-1) + (-2)$ $= -6$	$-6 < 0$, Neuron does not fire	-1
-1	1	$y = 2(-1) + 2(1) - 2$ $= -2$	$-2 < 0$, Neuron does not fire	-1
1	-1	$y = 2(1) + 2(-1) - 2$ $= -2$	$-2 < 0$, Neuron does not fire.	-1
1	1	$y = 2(1) + 2(1) - 2$ $= 2$	$2 > 0$, Neuron fire	1

∴ In this way, we can prepare Neuron for 'AND' gate using Hebbian Learning.

Competitive Learning

⇒ In the case of Neural Network, what happens is that, if somebody is a topper, then he will be favored, he will be favored in the sense that the synaptic connections will be. so, modified that next time, it will be easy for him to become a topper. Something like an examiner, who will follow the policy of partiality, you know that if I know that a student is already good, he is the topper of the class, then I will try to give him more marks and I will help him in becoming the topper. Something that mechanism, but it happens in the case of competitive learning. That means to say that, in the case of competitive learning, what happens is that, the winner will be favored next time. So, winner will have preference in terms of the synaptic connections, so this is the sprit of the competitive learning.



in competitive learning, they will be fully interconnected, means that all these four inputs will be connected to all the output neurons followed

* all this feedback connections, they are going to be inhibitory in nature, that means the competitors, never support themselves, they only try to weaken the other competitors

To follow a very simple mathematical model,

it says, that y_k, that is the output of the kth unit, the output kth unit will be equal

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \text{ for all } j,\ j \neq k \\ 0 & \text{otherwise.} \end{cases}$$

and another thing is that, a competitive learning network always constraints is that the sum total of the weights to a particular neuron. That is to say that, if we consider the neuron k, then the summation of w_{kj}, meaning what that ~~the~~ for the kth neuron all the inputs that it is

$$\sum w_{kj} = 1 \quad \text{for all } k.$$

receiving, if you sum them up, that means, to say that, if you sum them up over j, in this particular example, if you summit up for j is equal to 1 to 4 (see the picture, w_{51}, w_{52}, w_{53} and w_{54}) and then the summation of w_{kj} will be equal to 1 for all k.

Competitive learning rule :- Every learning mechanism has got learning rule. Competitive learning rule says that

$$\text{change of weight, } \Delta w_{kj} \begin{cases} \eta (x_j - w_{kj}) & \text{if neuron k wins.} \\ 0 & \text{if neuron k ~~losers~~ loses.} \end{cases}$$

* if the neuron is ~~losers~~ losers we don't bothered to readjust it is weight. if supposing (here), 5 is the ~~so~~ losers, ~~if neuron 5~~ in that case, we are going to keep w_{51}, w_{52}, w_{53} and w_{54} as it is, without doing any modification.