Part-C

16, Hebb Training Algorithm with example:

1. Donald Hebb started in 1949 that in the brain, the learning is performed by the change in the Synaptic Gap. Hebb explained it:

"When an axon of cell A is near enough to excite cell B, and repeatedly or permanently takes place in firing it, some growth process or metabolic changes takes place in one or both the cells such that A's efficiency, as one of the cell firing B, is increased"

2. According to the Hebb rule, the weight Vector is found to increase proportionately to the Product of the Input & the learning Signal. Here the Learning Signal is equal to the neuron's Output.

3. In Hebb Learning, if two interconnected neurons are 'on' simultaneously then the weight associated with these neurons can be increased by the modification made by the Synaptic Gap. The updated weight is

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

4. The Hebb rule is more suited for bipolar data than binary data. If binary data is used, the above weight weight updation formula cannot distinguish two conditions namely:

i) A training Pair in which an input unit is "on" & target value is "off"

ii) A training Pair in which both the input unit & the target value are "off"

5. Thus, there are limitation in Hebb rule application over binary data, Hence the representation using bipolar data is advantageous.

Hebb's Training Algorithm:

Training Algorithm is used for the calculation & adjustment of weights.

Step 1: First initialize the weights. Basically in this network they may be set to zero

$$\text{i.e., } w_i = 0 \text{ for } i = 1 \text{ to } n$$

where n - total number of input neurons

Step 2: Step 3-5 have to be performed for each input training vector & target Output pair, s:t.

Step 3: Input Units activations are set. Generally, the activation function of input layer is identity function:

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

Step 4: Output units Activations are set $y = t$

Step 5: weight adjustments & bias adjustments are performed

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

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

The above five steps Complete the algorithm process. In step 5 the weight updation formula can also be

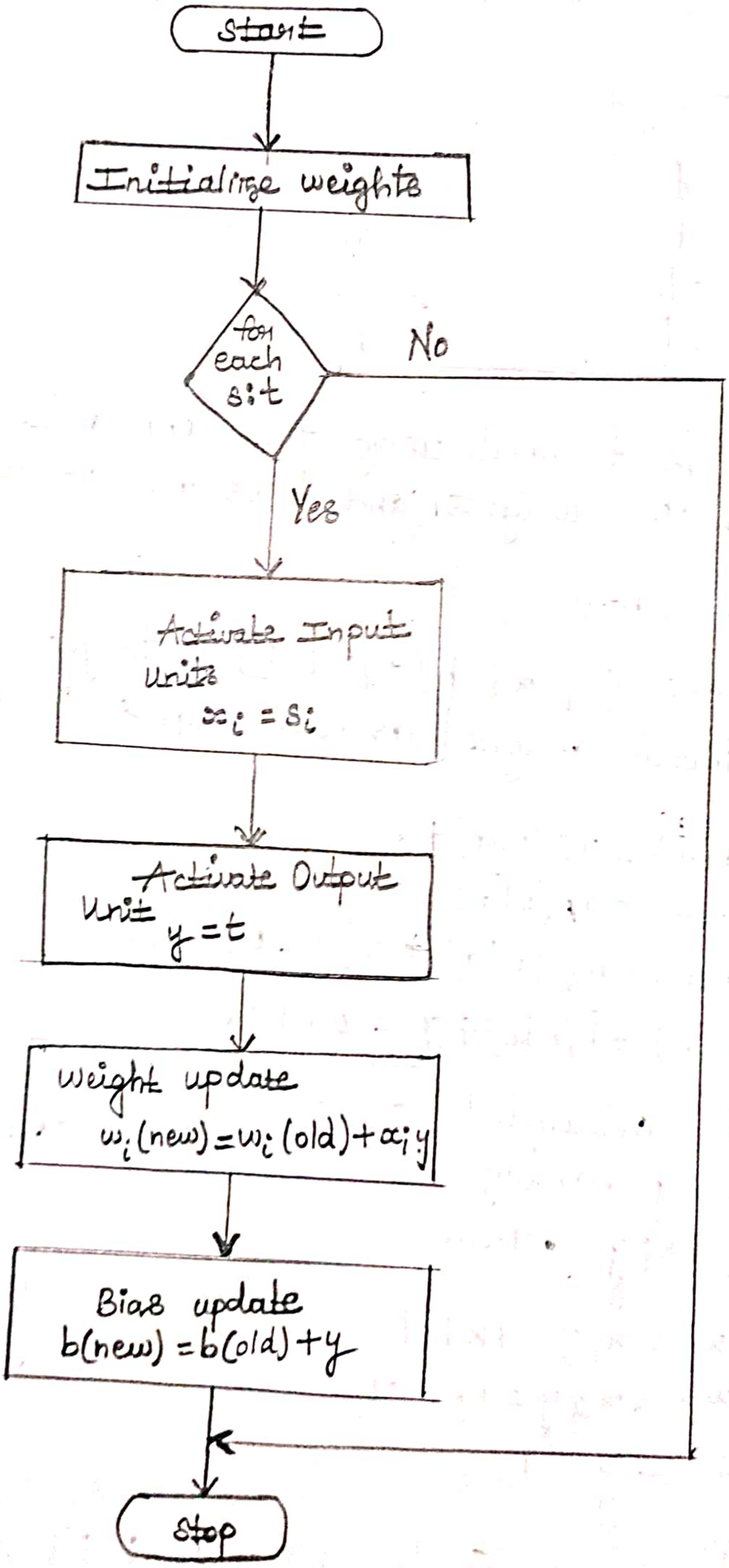
$$w(\text{new}) = w(\text{old}) + xy$$

Here the change in weight can be expressed as

$$\Delta w = xy$$

As a result, $w(\text{new}) = w(\text{old}) + \Delta w$.

Flow chart for Hebb's Training Algorithm:


Start
Initialize weights
for each s:t
No
Yes
Activate Input units $x_i = s_i$
Activate Output unit $y = t$
weight update $w_i(new) = w_i(old) + x_i y$
Bias update $b(new) = b(old) + y$
Stop

Example : Design a Hebb net to Implement logical AND function (use bipolar input & target)

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

The network is trained using the Hebb network training Algorithm. Initially the weights and bias are set to zero

$$w_1 = w_2 = b = 0$$

1. First Input $[x_1 \;\; x_2 \;\; b] = [1 \;\; 1 \;\; 1]$ & target = 1 i.e., $y = 1$ Setting the initial weight as old weights & applying Hebb rule we get

$$w_i\,(\text{new}) = w_i\,(\text{old}) + x_i y$$
$$w_1\,(\text{new}) = w_1\,(\text{old}) + x_1 y = 0 + 1 \times 1 = 1$$
$$w_2\,(\text{new}) = w_2\,(\text{old}) + x_2 y = 0 + 1 \times 1 = 1$$
$$b\,(\text{new}) = b\,(\text{old}) + y = 0 + 1 = 1$$

The weight calculated above are the final weight that are obtained after presenting the first Input. The weight changes here is $\Delta w_i = x_i y$. Hence

$$\Delta w_1 = x_1 y = 1 \times 1 = 1$$
$$\Delta w_2 = x_2 y = 1 \times 1 = 1$$
$$\Delta b = y = 1$$

2. Second Input $[x_1 \;\; x_2 \;\; b] = [1 \;\; -1 \;\; 1]$ and $y = -1$. The initial or old weights here are final (new) weight

$$[w_1 \;\; w_2 \;\; b] = [1 \;\; 1 \;\; 1]$$

$$[w_1 \quad w_2 \quad b] = [1\ 1\ 1]$$

The weight changes here is

$$\Delta w_1 = x_1 y = 1 \times -1 = -1$$
$$\Delta w_2 = x_2 y = -1 \times -1 = 1$$
$$\Delta b = y = -1$$

The new weights here are

$$w_1\,(\text{new}) = w_1\,(\text{old}) + \Delta w_1 = 1 - 1 = 0$$
$$w_2\,(\text{new}) = w_2\,(\text{old}) + \Delta w_2 = 1 + 1 = 2$$
$$b\,(\text{new}) = b(\text{old}) + \Delta b = 1 - 1 = 0$$

III^{ly} by presenting the third & fourth input pattern, the new weights can be calculated.

Inputs				weight changes			weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1 (0	w_2 0	b 0)
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

The separating line equation is given by

$$x_2 = \frac{-w_1}{w_2} x_1 - \frac{b}{w_2}$$

for first Input

$$x_2 = \frac{-1}{1} x_1 - \frac{1}{1} \Rightarrow x_2 = -x_1 - 1$$

IIIly; for Second Input [1 -1 1] , the Separating line is

$$x_2 = \frac{-0}{2} x_1 - \frac{0}{2} \Rightarrow x_2 = 0$$

for third Input [-1 1 1], it is

$$x_2 = \frac{-1}{1} x_1 + \frac{1}{1} \Rightarrow x_2 = -x_1 + 1$$

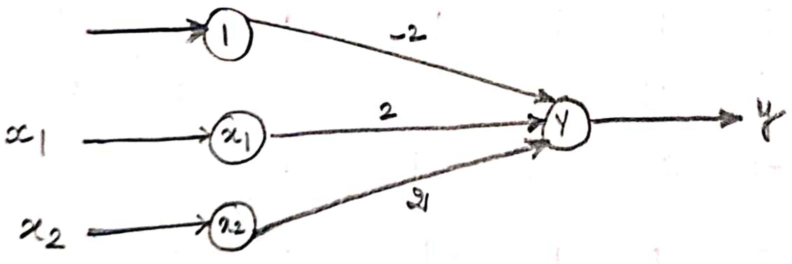
finally fourth Input [-1 -1 1]

$$x_2 = \frac{-2}{2} x_1 + \frac{2}{2} \Rightarrow x_2 = -x + 1$$

By plotting Graph using these separating lines we obtain the decision boundary (It separates the positive response region from the negative response region). Hence the weights obtained from this are the final weight

$$w_1 = 2 \; ; \; w_2 = 2 \text{ and } b = -2$$

The Network can be represented as


1
-2
x_1
x_1
2
Y
y
x_2
x_2
2