Hebb network

- In 1949, Donald Hebb said that learning in the brain occurs due to the change in the synaptic gap
- Hebb proposed one of the key ideas in biological learning, commonly known as **Hebb's Law**.
- Hebb's Law states 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 change takes place in one or both the cells such than A's efficiency, as one of the cells firing B, is increased.

Hebb’s Law can be represented in the form of two rules:

1. **If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.**

2. **If two neurons on either side of a connection are activated asynchronously, then the weight of that connection is decreased.**

Hebb’s Law provides the basis for learning without a teacher-Unsupervised learning (Learning here is a **local phenomenon** occurring without feedback from the environment)

Hebb network

- According to Hebb rule, ***the weight vector is found to increase proportionately to the product of the input and the learning signal. Here learning signal is equal to neuron output.***
- In Hebb learning, two interconnected neurons are 'on' simultaneously. Weight updating associated with these neurons is increased by modification in the synaptic gap (strength)
- The weight update in Hebb rule is given by
  - $\boldsymbol{W_i(\text{new}) = w_i\ (\text{old}) + x_i\, y.}$
- It is suited more for bipolar data.

Hebb Network

Hebb Rule: $w_i(new) = w_i(old) + x_i y$

The Algorithm:

Step 0 : Initialize the weights. Basically in this network they may be set to zero, i.e., $w_i = 0 \; for \; i \; to \; n$

Step 1 : Steps 2 – 4 have to be performed for each input training vector and the target output pair, s:t

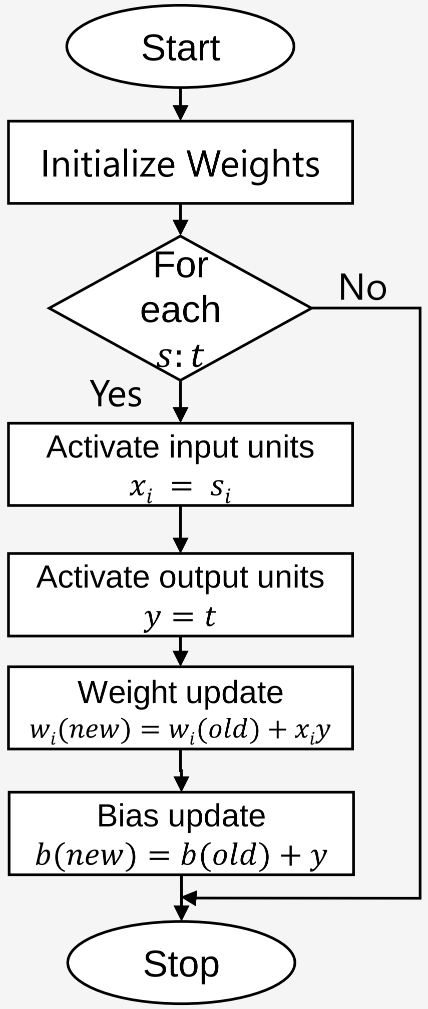
Step 2 : Input units activations are set. Generally, the activation function of the input layer us identify function: $x_i = s_i \; for \; i = 1 \; to \; n$

Step 3 : Output units activations are set: $y = t$

Step 4 : Weight adjustments and bias adjustments are performed:

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

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


Start
Initialize Weights
For each $s:t$
No
Yes
Activate input units $x_i = s_i$
Activate output units $y = t$
Weight update $w_i(new) = w_i(old) + x_i y$
Bias update $b(new) = b(old) + y$
Stop

Design a Hebb net to implement OR function

Use bipolar data in the place of binary data Initially the weights and bias are set to zero w1=w2=b=0

Inputs			Target
X1	X2	B	y
1	1	1	1
1	-1	1	1
-1	1	1	1
-1	-1	1	-1

- **First input [x1 x2 b] = [1 1 1] and target = 1 [i.e. y = 1],** setting the intial weights as

old weights and applying the hebb rule, we get

wi (new) = wi(old) + xi*y

w1(new) = w1(old) + x1*y = 0+1*1 = 1

w2(new) = w2(old) + x2*y = 0+1*1 = 1

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

- **Similarly for second input [x1 x2 b] = [1 -1 1] and y = 1**

w1(new) = w1(old) + x1*y = 1+1*1 = 2

w2(new) = w2(old) + x2*y = 1+(-1)*1 = 0

b(new) = b(old) +y = 1+1 = 2 & Δb = 1

- **Similarly for third and fourth input, output can be calculated.**

Inputs			y	Weights		
X1	X2	b	Y	W1(0)	W2(0)	b(0)
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

Design the Hebb network Numericals

- Design and implement Hebbs Network for OR, AND, NAND function. Use Bipolar inputs and targets
- Design and Implement Hebbs Network for the patter given.