Hebbian Learning Rule

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:

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

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

Design a Hebb Net to implement logical AND function

- Initially the weights and bias are set to zero, i.e.,

$$w_1 = w_2 = b = 0$$

- First input [x1 x2 b] = [111] and target = 1 [i.e., y = 1]:

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

- Setting the initial weights as old weights and applying the Hebb rule, we get

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

$$\Delta w_i = x_i y$$

$$\Delta w_1 = x_1 y = 1 \times 1 = 1$$

$$\Delta w_2 = x_2 y = 1 \times 1 = 1$$

$$\Delta b = y = 1$$

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0 + 1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0 + 1 = 1$$

$$b(\text{new}) = b(\text{old}) + \Delta b = 0 + 1 = 1$$

Design a Hebb Net to implement logical AND function

- Second input [x1 x2 b] = [1 −1 1] and y = −1:
- The weight change here is

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

$$\Delta w_1 = x_1 y = 1 \times -1 = -1$$

$$\Delta w_2 = x_2 y = -1 \times -1 = 1$$

$$\Delta b = y = -1$$

- The new weights here are

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

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 1 + 1 = 2$$

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

Design a Hebb Net to implement logical AND function

- Similarly, by presenting the third and fourth input patterns, the new weights can be calculated.

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

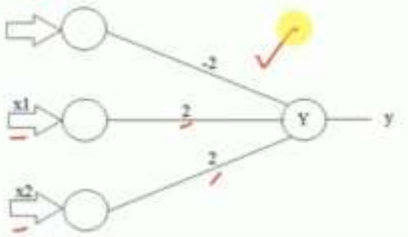
Inputs				Weight changes			Weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1 (0	w_1 0	b 0)
1	1	1	1	1	1	1	1	1	1
1	−1	1	−1	−1	1	−1	0	2	0
−1	1	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−1	1	1	−1	2	2	−2

Design a Hebb Net to implement logical AND function

- Similarly, by presenting the third and fourth input patterns, the new weights can be calculated.

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

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



Hebb Net Solved Numerical Example - 2

+	+	+
	+	
+	+	+

'I'

+	+	+
+		+
+	+	+

'O'

Hebb Net Solved Numerical Example - 2

- Using the Hebb rule, find the weights required to perform the following classifications of the given input patterns shown in Figure.
- The pattern is shown as 3 × 3 matrix form in the squares.
- The "+" symbols represent the value "1" and empty squares indicate "−1".

+	+	+
	+	
+	+	+

'I'

+	+	+
+		+
+	+	+

'O'

Hebb Net Solved Numerical Example - 2

5. Implement AND

- Set the initial weights and bias to zero, i.e.,

$$w_1 = w_2 = w_3 = w_4 = w_5 = w_6 = w_7 = w_8 = w_9 = b = 0$$

- Presenting first input pattern (I), we calculate change in weights:

$$w_i(\text{new}) = w_i(\text{old}) + \Delta w_i \qquad [\Delta w_i = x_i y]$$

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
I	1	1	1	−1	1	−1	1	1	1	1	1
0	1	1	1	1	−1	1	1	1	1	1	−1

$$\Delta w_i = x_i y, \quad i = 1 \text{ to } 9$$
$$\Delta w_1 = x_1 y = 1 \times 1 = 1$$
$$\Delta w_2 = x_2 y = 1 \times 1 = 1$$
$$\Delta w_3 = x_3 y = 1 \times 1 = 1$$
$$\Delta w_4 = x_4 y = -1 \times 1 = -1$$
$$\Delta w_5 = x_5 y = 1 \times 1 = 1$$
$$\Delta w_6 = x_6 y = -1 \times 1 = -1$$

Hebb Net Solved Numerical Example - 2

+	+	+
	+	
+	+	+

'I'

+	+	+
+		+
+	+	+

'O'

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
I	1	1	1	−1	1	−1	1	1	1	1	1
0	1	1	1	1	−1	1	1	1	1	1	−1

Hebb Net Solved Numerical Example - 2

- Setting the old weights as the initial weights here, we obtain

$$w_i(\text{new}) = w_i(\text{old}) + \Delta w_i$$
$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0 + 1 = 1$$
$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0 + 1 = 1$$
$$w_3(\text{new}) = w_3(\text{old}) + \Delta w_3 = 0 + 1 = 1$$
$$w_4(\text{new}) = -1, \quad w_5(\text{new}) = 1, \quad w_6(\text{new}) = -1,$$
$$w_7(\text{new}) = 1, \quad w_8(\text{new}) = 1, \quad w_9(\text{new}) = 1,$$
$$b(\text{new}) = 1$$

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
I	1	1	1	−1	1	−1	1	1	1	1	1
0	1	1	1	1	−1	1	1	1	1	1	−1

$$\Delta w_i = x_i y, \quad i = 1 \text{ to } 9$$
$$\Delta w_1 = x_1 y = 1 \times 1 = 1$$
$$\Delta w_2 = x_2 y = 1 \times 1 = 1$$
$$\Delta w_3 = x_3 y = 1 \times 1 = 1$$
$$\Delta w_4 = x_4 y = -1 \times 1 = -1$$
$$\Delta w_5 = x_5 y = 1 \times 1 = 1$$
$$\Delta w_6 = x_6 y = -1 \times 1 = -1$$
$$\Delta w_7 = x_7 y = 1 \times 1 = 1$$
$$\Delta w_8 = x_8 y = 1 \times 1 = 1$$
$$\Delta w_9 = x_9 y = 1 \times 1 = 1$$
$$\Delta b = y = 1$$

Hebb Net Solved Numerical Example - 2

- Set the initial weights and bias to zero, i.e.,

$$w_1 = w_2 = w_3 = w_4 = w_5$$
$$= w_6 = w_7 = w_8 = w_9 = b = 0$$

- Presenting first input pattern (I), we calculate change in weights:

$$w_i(\text{new}) = w_i(\text{old}) + \Delta w_i \qquad [\Delta w_i = x_i y]$$

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
I	1	1	1	-1	1	-1	1	1	1	1	1
0	1	1	1	1	-1	1	1	1	1	1	-1

$\Delta w_i = x_i y, \quad i = 1 \text{ to } 9$

$\Delta w_1 = x_1 y = 1 \times 1 = 1$

$\Delta w_2 = x_2 y = 1 \times 1 = 1$

$\Delta w_3 = x_3 y = 1 \times 1 = 1$

$\Delta w_4 = x_4 y = -1 \times 1 = -1$

$\Delta w_5 = x_5 y = 1 \times 1 = 1$

$\Delta w_6 = x_6 y = -1 \times 1 = -1$

$\Delta w_7 = x_7 y = 1 \times 1 = 1$

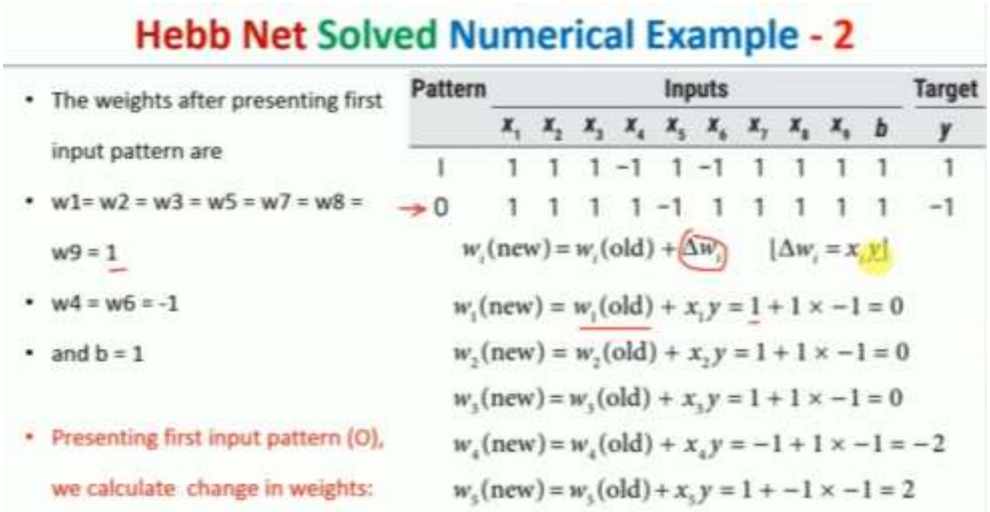
$\Delta w_8 = x_8 y = 1 \times 1 = 1$

$\Delta w_9 = x_9 y = 1 \times 1 = 1$

$\Delta b = y = 1$

Hebb Net Solved Numerical Example - 2

- The weights after presenting first input pattern are
- w1= w2 = w3 = w7 = w8 = w9 = 0
- w5 = 2
- w4 = w6 = -2
- and b = 0



Hebb Net Solved Numerical Example - 2

- The weights after presenting first input pattern are
- w1= w2 = w3 = w5 = w7 = w8 = w9 = 1
- w4 = w6 = -1
- and b = 1
- Presenting first input pattern (O), we calculate change in weights:

Pattern	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	Target y
I	1	1	1	-1	1	-1	1	1	1	1	1
O	1	1	1	1	-1	1	1	1	1	1	-1

$w_i(\text{new}) = w_i(\text{old}) + \Delta w_i \qquad [\Delta w_i = x_i y]$

$w_1(\text{new}) = w_1(\text{old}) + x_1 y = 1 + 1 \times -1 = 0$

$w_2(\text{new}) = w_2(\text{old}) + x_2 y = 1 + 1 \times -1 = 0$

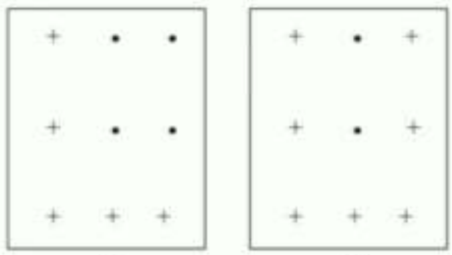
$w_3(\text{new}) = w_3(\text{old}) + x_3 y = 1 + 1 \times -1 = 0$

$w_4(\text{new}) = w_4(\text{old}) + x_4 y = -1 + 1 \times -1 = -2$

$w_5(\text{new}) = w_5(\text{old}) + x_5 y = 1 + -1 \times -1 = 2$

Hebb Net Solved Numerical Example - 3

- Find the weights required to perform the following classifications of given input patterns using the Hebb rule.



'L' 'U'

Hebb or Hebbian learning rule comes under **Artificial Neural Network** (ANN) which is an architecture of a large number of interconnected elements called neurons. These neurons process the input received to give the desired output. The nodes or neurons are linked by **inputs**(x1,x2,x3…xn), **connection weights**(w1,w2,w3…wn), and **activation functions**(a function that defines the output of a node).

In layman's term, a neural network trains itself with known examples and solves various problems that are unknown or difficult to be solved by humans!!

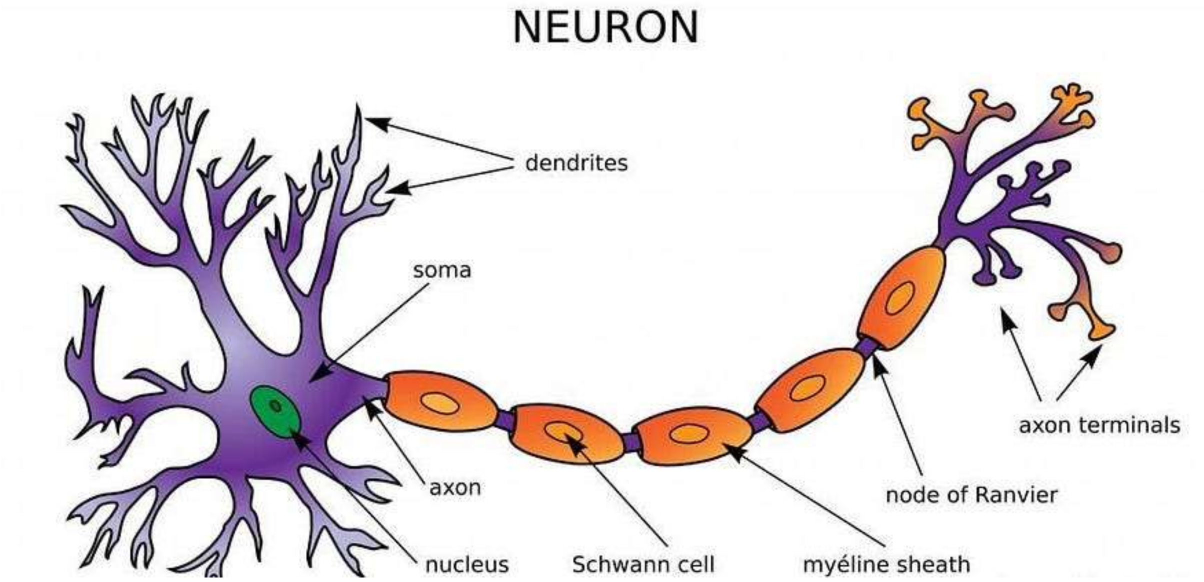


Fig 1. Neuron

Now, coming to the explanation of Hebb network, " 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 cells firing B, is increased."

basically the above explanation is derived from the modus operandi used by the brain where learning is performed by the changes in the ***synaptic gap***

In this, if 2 interconnected neurons are **ON** simultaneously then the weight associated with these neurons can be increased by the modification made in their synaptic gaps(strength). The weight update in the Hebb rule is given by;

ith value of w(new) = ith value of w(old) + (ith value of x * y)

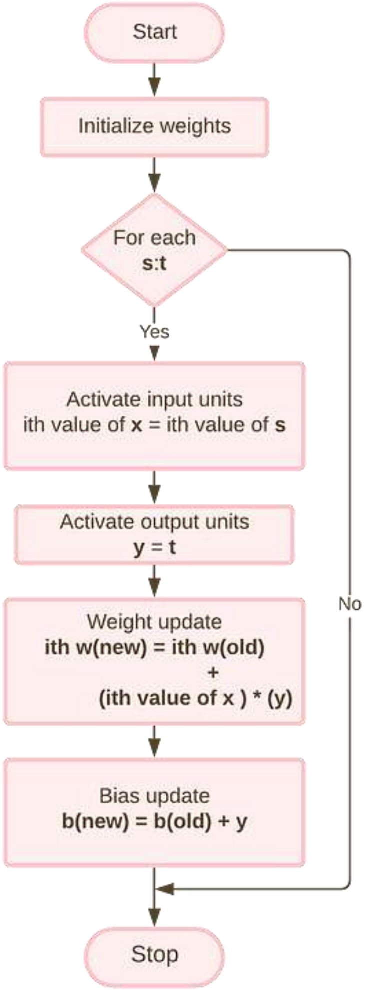


Fig 2. Flowchart of Hebb training algorithm

STEP 1:Initialize the weights and bias to **'0'** i.e w1=0,w2=0, …., wn=0.

STEP 2: 2–4 have to be performed for each input training vector and target output pair **i.e. s:t** (s=training input vector, t=training output vector)

STEP 3: Input units activation are set and in most of the cases is an identity function(one of the types of an activation function) for the input layer;

ith value of x = ith value of s for i=1 to n

Identity Function:*Its a linear function and defined as* **f(x)=x for all x**

STEP 4: Output units activations are set y:t

STEP 5: Weight adjustments and bias adjustments are performed;

1. ***ith value of w(new) = ith value of w(old) + (ith value of x * y)***

2. ***new bias(value) = old bias(value) + y***

Finally the cryptic or to be precise, a bit unintelligible part comes to an end but once you understand the below-solved example, you will definitely understand the above flowchart XD!!

Designing a Hebb network to implement AND function:

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

Fig 3. Training data table

AND function is very simple and mostly known to everyone where the output is **1/SET/ON** if both the inputs are **1/SET/ON.** But in the above example, 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.

Starting with setp1 which is inializing the weights and bias to '0', so we get w1=w2=b=0

A) First input **[x1,x2,b]=[1,1,1]** and **target/y = 1**. Now using the initial weights as old weight and applying the Hebb rule(ith value of w(new) = ith value of w(old) + (ith value of x * y)) as follow;

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

Now the above final weights act as the initial weight when the second input pattern is presented. And remember that weight change here is;

Δith value of w = ith value of x * y

hence weight changes relating to the first input are;

Δw1= x1y = 1*1=1

Δw2 = x2y = 1*1=1

Δb = y = 1

We got our first output and now we start with the second inputs from the table(2nd row)

B) Second input **[x1,x2,b]=[1,-1,1]** and **target/y = -1**.

Note: here that the initial or the old weights are the final(new) weights obtained by performing the first input pattern **i.e [w1,w2,b] = [1,1,1]**

Weight change here is;

Δw1 = x1*y = 1*-1 = -1

Δw2 =x2*y = -1 * -1 = 1

***Δb* = *y* = -1**

The new weights here are;

w1(new) = w1(old) + *$\Delta w1$*= 1–1 = 0

w2(new) = w2(old) + *$\Delta w2$*= 1+1 = 2

b(new) = b(old) + *Δb*= 1–1=0

similarly, using the same process for third and fourth row we get a new table as follows;

Inputs				Weight Changes			Weights		
x1	x2	b	y	Δw1	Δw2	Δb	w1 (0	w2 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

Fig 4. Final output table

Here the final weights we get are w1=2, w2=2, b=-2

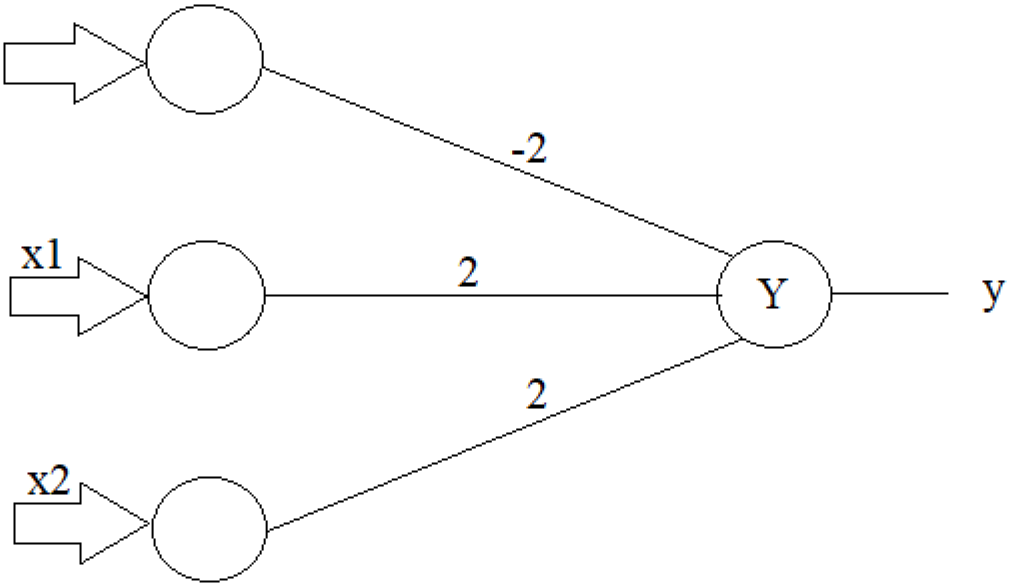


*Fig 5. Hebb network for AND function

Thank you for reading this article till the end, I hope you understood the concept perfectly.