# Hebb Network

Jay shah · Follow

Published in Analytics Vidhya · 4 min read · Oct 21, 2020

138 1

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!!*

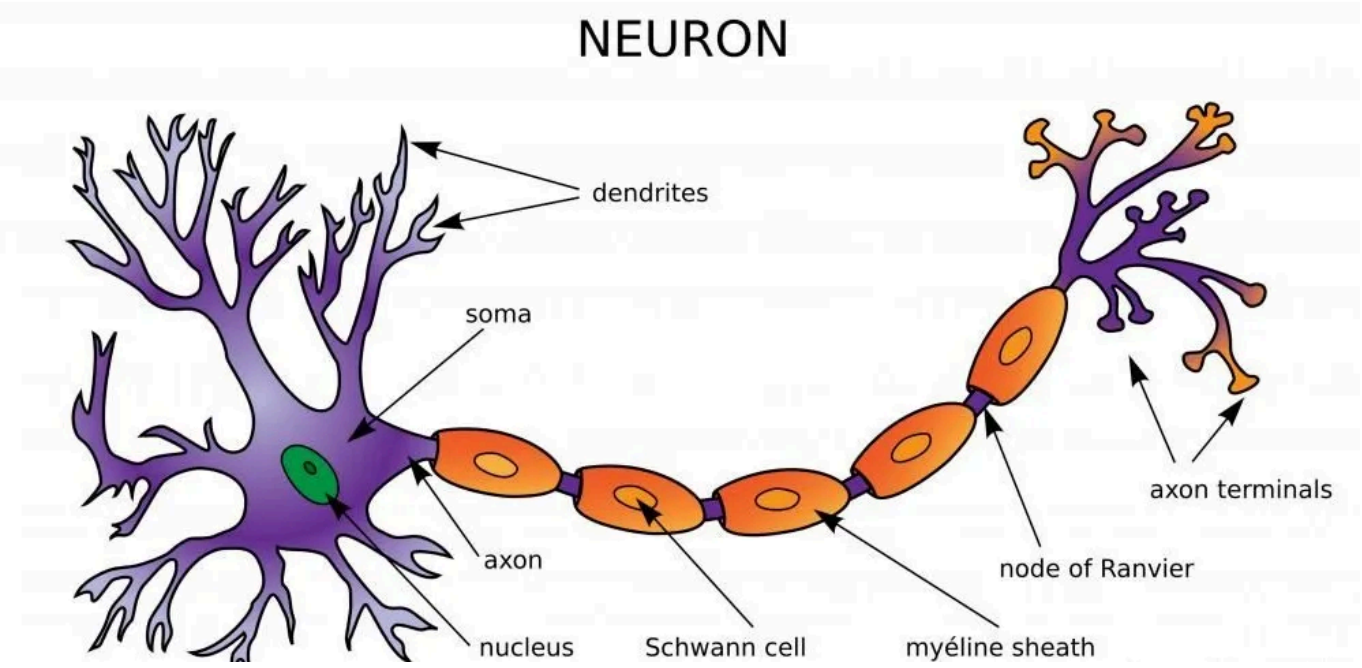


**Fig 1. Neuron**

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*

Open in app ↗ Sign up Sign in

Medium Search Write

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)***

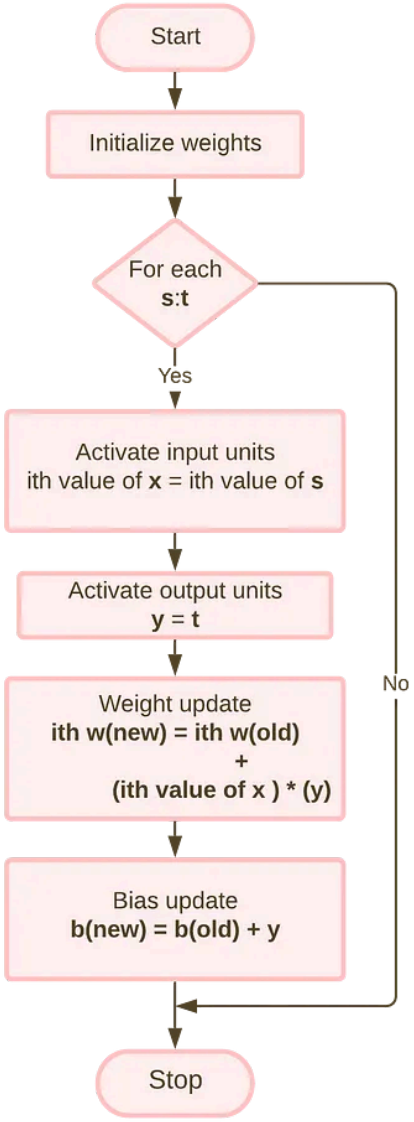


**Fig 2. Flowchart of Hebb training algorithm**

**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*

$\Delta b = y = 1$

> *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;

$\Delta w1 = x1*y = 1*-1 = -1$

$\Delta w2 = x2*y = -1 * -1 = 1$

$\Delta b = y = -1$

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) +** $\Delta b$**= 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**

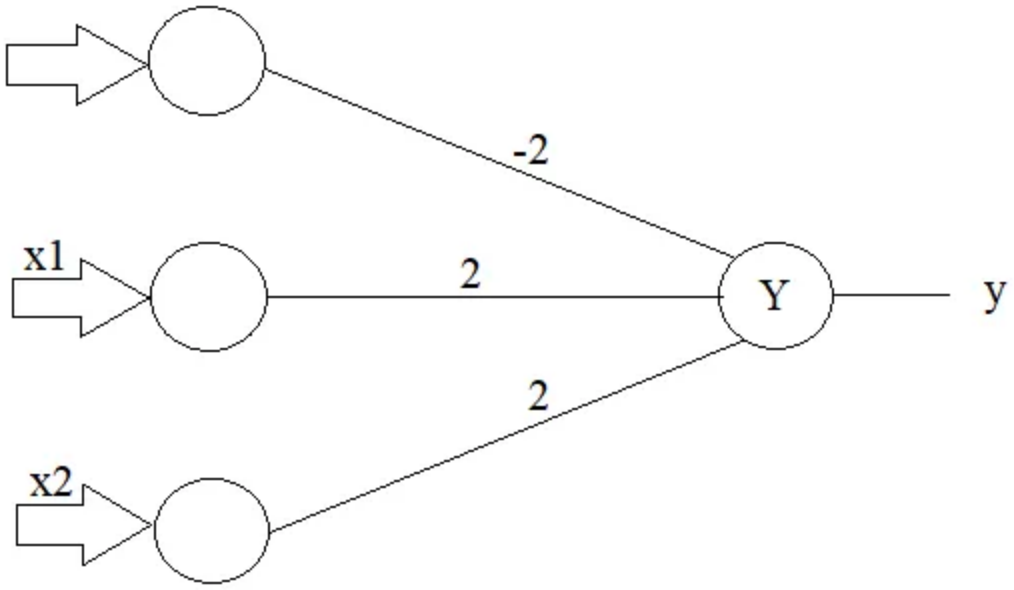


*Fig 5. Hebb network for AND function

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

Hebb Network | Neural Networks | Artificial Neural Network | Hebbian Learning Rule | Soft Computing