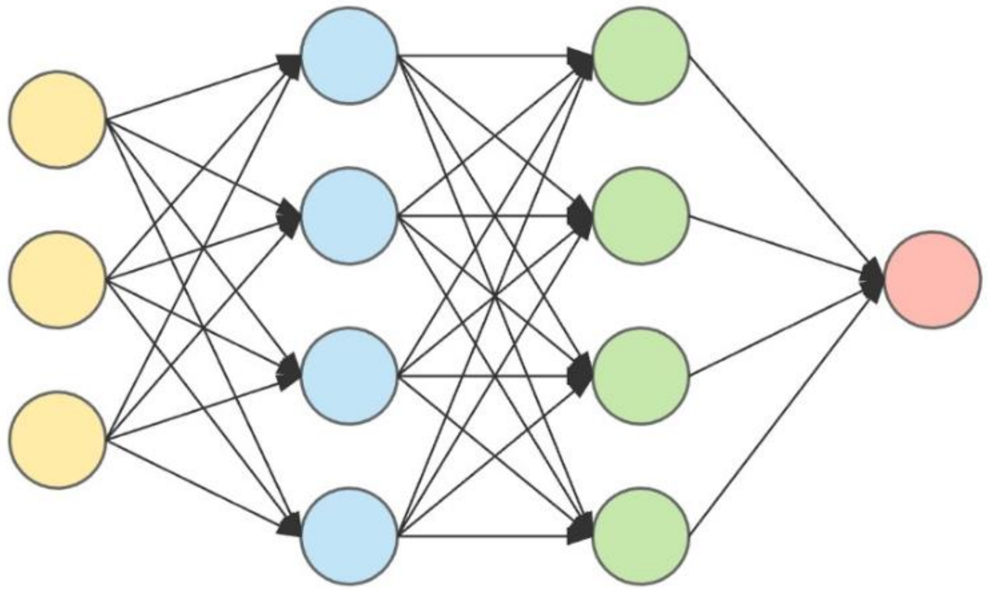
Neural Networks

Neural Networks

- Artificial Neural Networks are normally called Neural Networks (NN).
- Neural networks are in fact multi-layer Perceptrons.
- The perceptron defines the first step into multi-layered neural networks.

The Neural Network Model

- Input data (Yellow) are processed against a hidden layer (Blue) and modified against another hidden layer (Green) to produce the final output (Red).

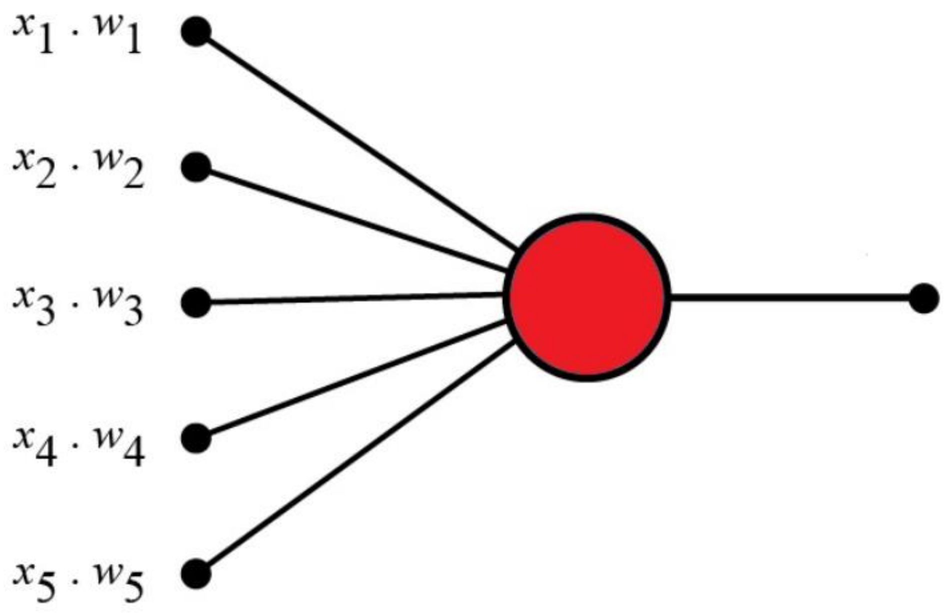


Perceptrons

- A Perceptron is an Artificial Neuron
- It is the simplest possible Neural Network
- Neural Networks are the building blocks of Machine Learning.

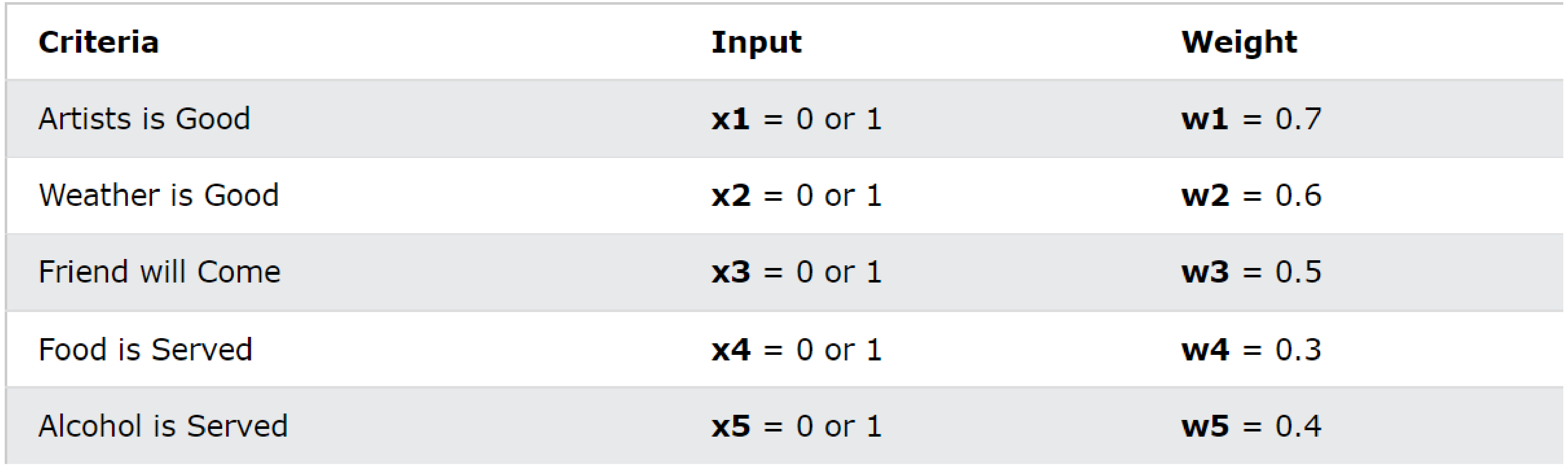
The Perceptron

- The original Perceptron was designed to take a number of binary inputs, and produce one binary output (0 or 1).
- The idea was to use different weights to represent the importance of each input, and that the sum of the values should be greater than a threshold value before making a decision like true or false (0 or 1).



Perceptron Example

- Imagine a perceptron (in your brain).
- The perceptron tries to decide if you should go to a concert.
- Is the artist good? Is the weather good?
- What weights should these facts have?



Criteria	Input	Weight
Artists is Good	**x1** = 0 or 1	**w1** = 0.7
Weather is Good	**x2** = 0 or 1	**w2** = 0.6
Friend will Come	**x3** = 0 or 1	**w3** = 0.5
Food is Served	**x4** = 0 or 1	**w4** = 0.3
Alcohol is Served	**x5** = 0 or 1	**w5** = 0.4

The Perceptron Algorithm

- Frank Rosenblatt suggested this algorithm:

1. Set a threshold value
2. Multiply all inputs with its weights
3. Sum all the results
4. Activate the output

1. Set a threshold value:

- Threshold = 1.5

2. Multiply all inputs with its weights:

- x1 * w1 = 1 * 0.7 = 0.7
- x2 * w2 = 0 * 0.6 = 0
- x3 * w3 = 1 * 0.5 = 0.5
- x4 * w4 = 0 * 0.3 = 0
- x5 * w5 = 1 * 0.4 = 0.4

3. **Sum all the results**:

- 0.7 + 0 + 0.5 + 0 + 0.4 = 1.6 (The Weighted Sum)

4. **Activate the Output**:

- Return true if the sum > 1.5 ("Yes I will go to the Concert")

Perceptron Terminology

- Perceptron Inputs
- Node values
- Node Weights
- Activation Function

Perceptron Inputs

- Perceptron inputs are called nodes.
- The nodes have both a value and a weight.

Node Values

- In the example above, the node values are: 1, 0, 1, 0, 1
- The binary input values (0 or 1) can be interpreted as (no or yes) or (false or true).

Node Weights

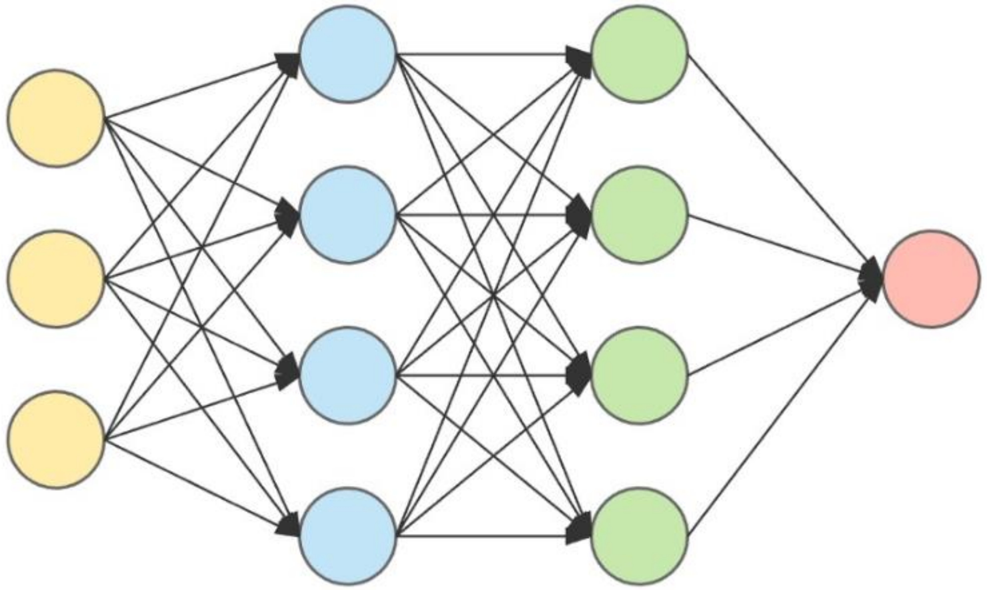
- Weights shows the strength of each node.
- In the example above, the node weights are: 0.7, 0.6, 0.5, 0.3, 0.4

The Activation Function

- The activation functions maps the result (the weighted sum) into a required value like 0 or 1.
- In the example above, the activation function is simple: (sum > 1.5)
- The binary output (1 or 0) can be interpreted as (yes or no) or (true or false).

Neural Networks

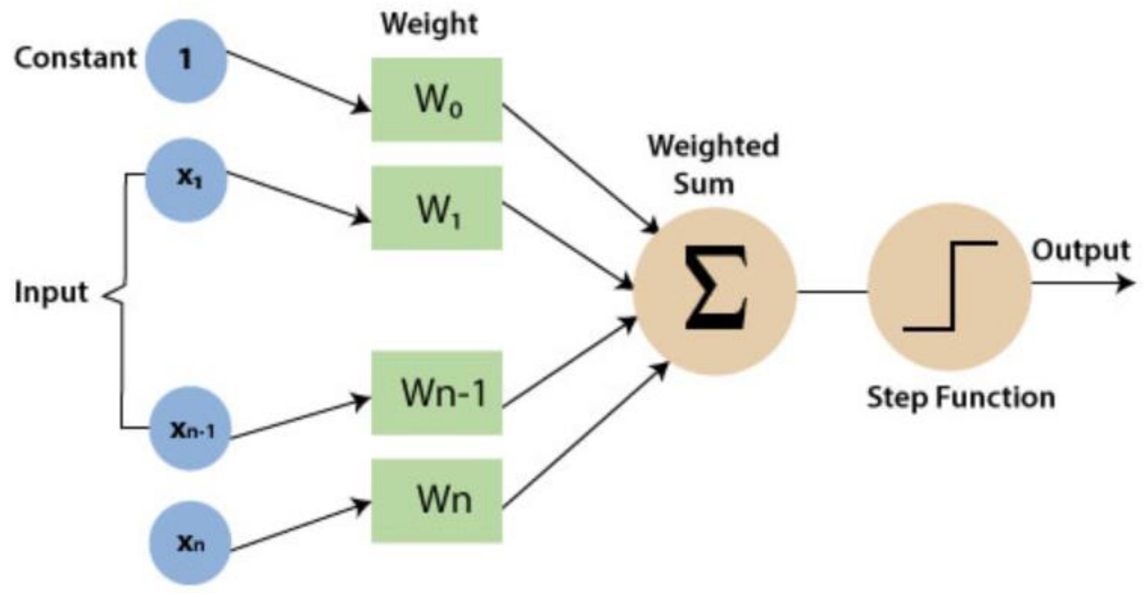
- The Perceptron defines the first step into Neural Networks.
- Multi-Layer Perceptrons can be used for very sophisticated decision making.



- In the Neural Network Model, input data (yellow) are processed against a hidden layer (blue) and modified against more hidden layers (green) to produce the final output (red).
- **The First Layer:**
  The 3 yellow perceptrons are making 3 simple decisions based on the input evidence. Each single decision is sent to the 4 perceptrons in the next layer.
- **The Second Layer**:
  The blue perceptrons are making decisions by weighing the results from the first layer. This layer make more complex decisions at a more abstract level than the first layer.
- **The Third Layer:**
  Even more complex decisions are made by the green perceptons.

- Perceptron consist of four parts and which are required to understand for the implementation of the perceptron model.
- **Input values or one input layer**
  The input layer of a perceptron is made of artificial input neurons and brings the initial data into the system for further processing.
- **Weights and bias**
  Weight represents the strength or dimension of the connection between units. If the weight from node 1 to node 2 has the greater quantity, then neuron 1 has greater influence over neuron 2. How much influence of the input will have on the output, is determined by weight.
  Bias is similar to the intercept added in a linear equation. It is an additional parameter which task is to adjust the output along with the weighted sum of the inputs to the neuron.

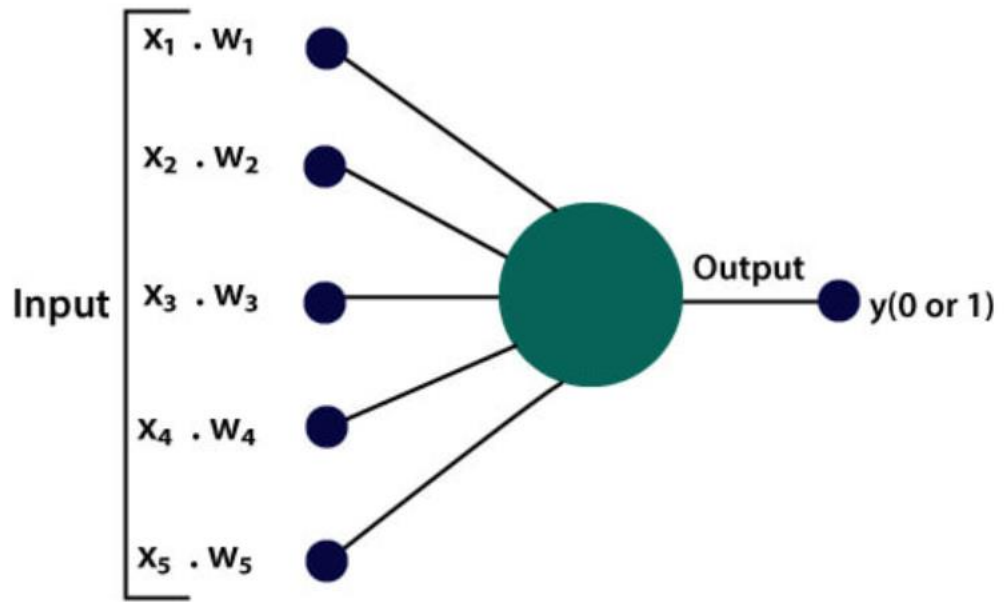
- **Activation Function**
  A neuron should be activated or not, is determined by an activation function. Activation function calculates a weighted sum and further adding bias with it to give the result.



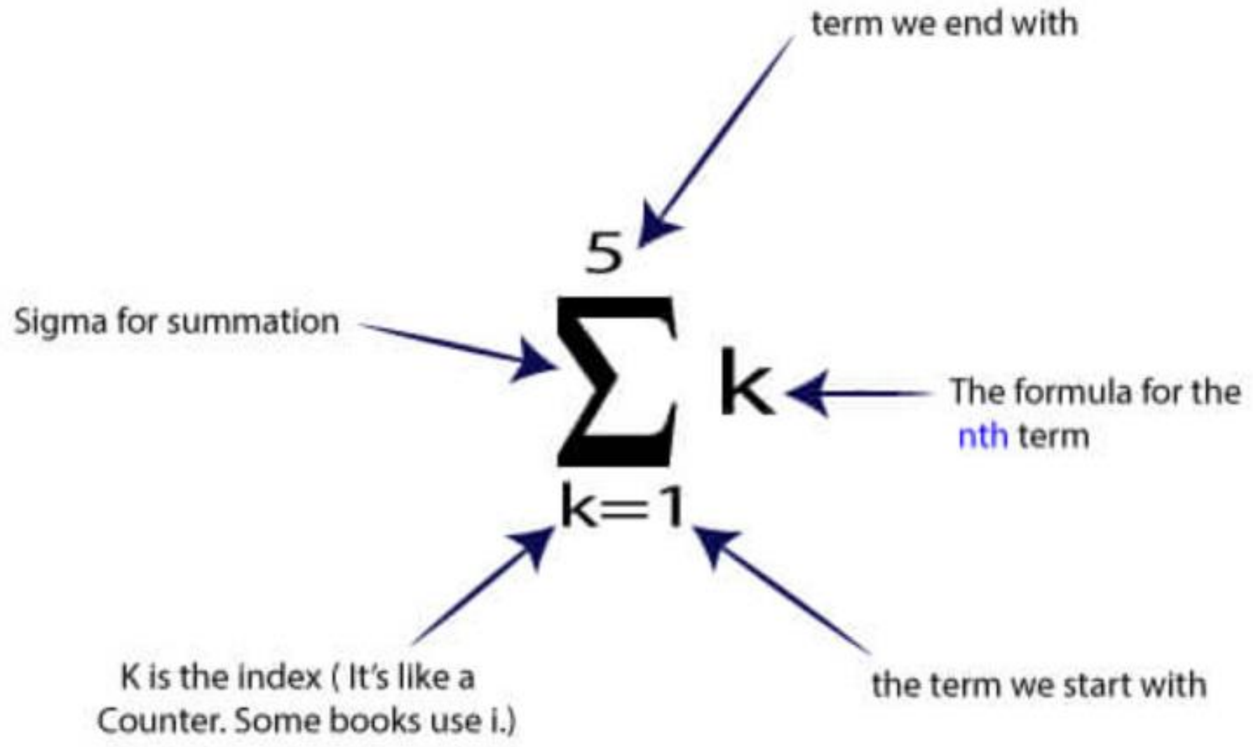
- **Neural Network** is based on the Perceptron, so if we want to know the working of the neural network, learn how perceptron work.

The Perceptron works on three simple steps which are as follows:

- In the first step, all the input x are multiplied with their weights denoted as K. This step is essential because the output of this step will be input for the next step.



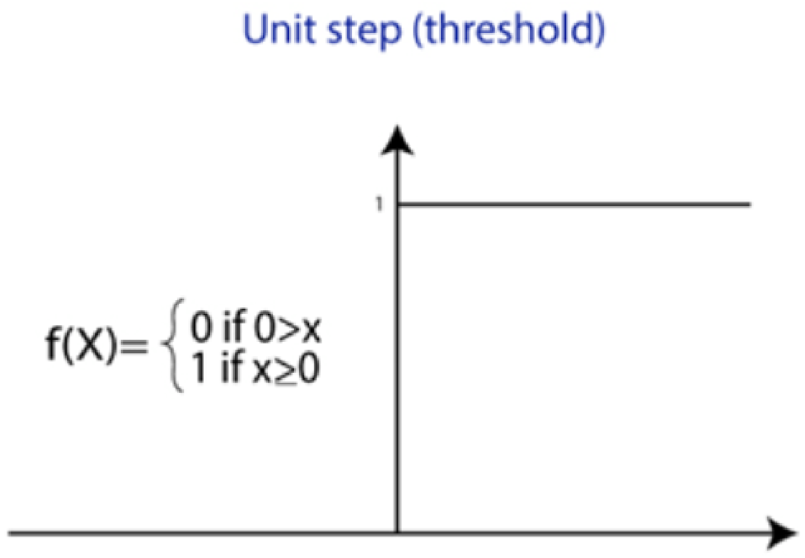
- Next step is to add all the multiplied value from K_1 to K_n. It is known as the weighted sum. This weighted sum will be treated as an input for the next step.


term we end with
Sigma for summation
$\sum_{k=1}^{5} k$
The formula for the nth term
K is the index (It's like a Counter. Some books use i.)
the term we start with

- In the next step, the weighted sum, which is calculated from the previous step, is applied to the correct activation function.

For example

- A unit step activation function

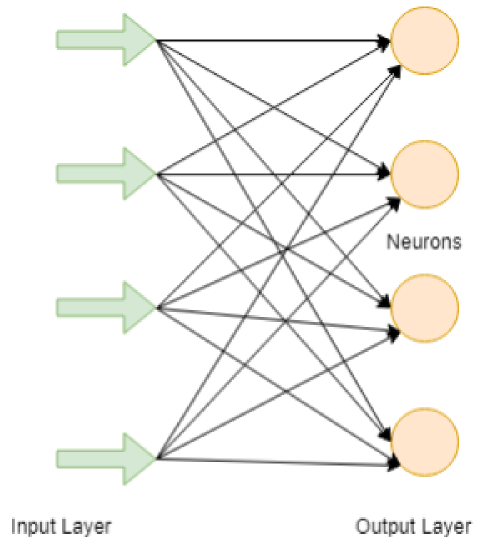


- A bias value allows you to shift the activation function curve up or down.
- The activation functions are used to map the input between the required value like (0, 1) or (-1, 1).
- Perceptron is usually used to classify the data into two parts. Therefore, it is also known as a **Linear Binary Classifier**.

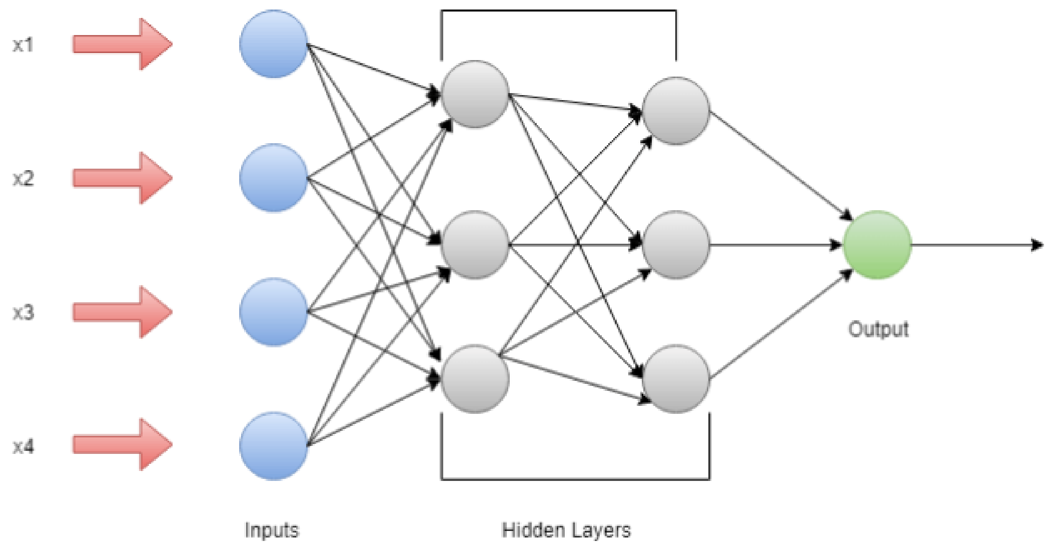
Neural Network and Deep Learning Neural Network

- **Artificial Neural Network** or **Neural Network** was modeled after the human brain.
- Human has a mind to think and to perform the task in a particular situation, but how can a machine do that? For this purpose, an artificial brain was designed, which is known as a Neural Network.
- As the human brain has neurons for passing information, similarly neural network has nodes to perform that task. Nodes are the mathematical functions.
- A Neural Network itself changes or learn based on input and output.
- The information that flows through the network affect the structure of the artificial Neural Network because of its learning and changing property.

- **Deep Learning Neural Network** is an advanced form of neural network.
- Deep Learning Neural Network have more than one hidden layer.
- Deep Learning Neural Network gets the more complex dataset as that your model is able to learn from.



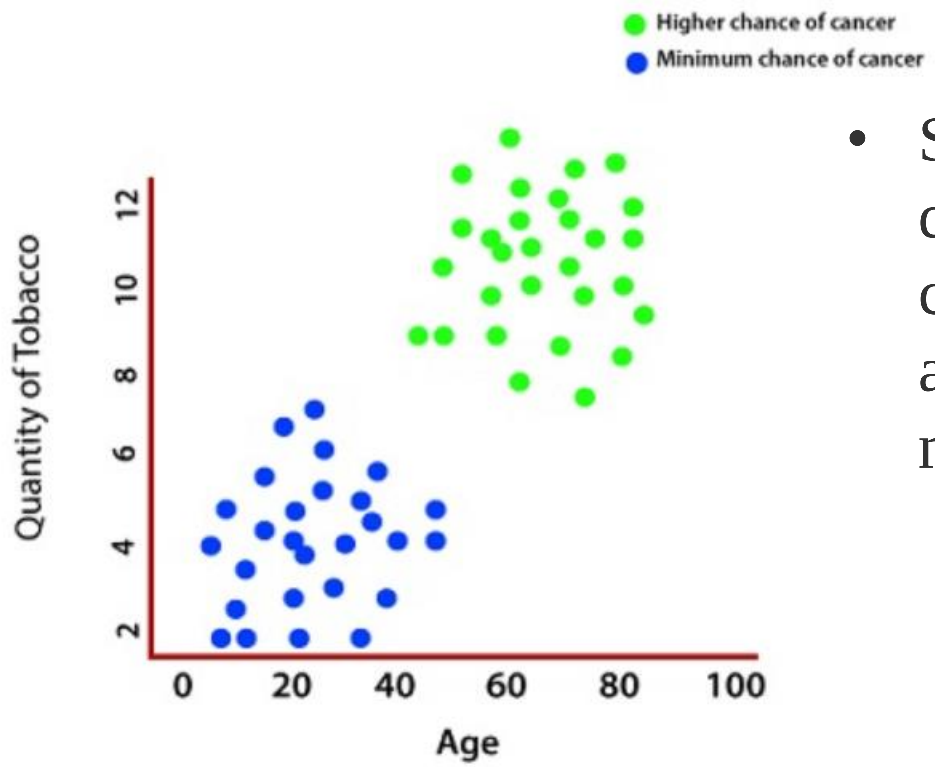
Simple Neural Network



Deep Learning Neural Network

Perceptron Model

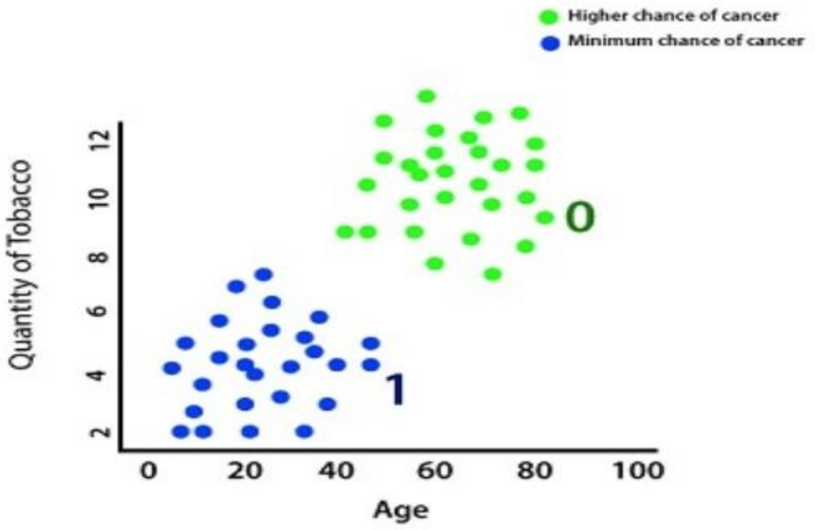
- Let see an example to understand the perceptron model.
- Imagine there is a hospital which annually does the operation of thousands of patients and tells you to create a predictive model which is able to accurately determine whether or not someone is likely to be cancer or not.
- With the help of previously determine data, we predict whether or not someone is likely to be cancer-based on their age, which traverses the x-axis and their quantity of tobacco inhaled which traverse the y-axis.

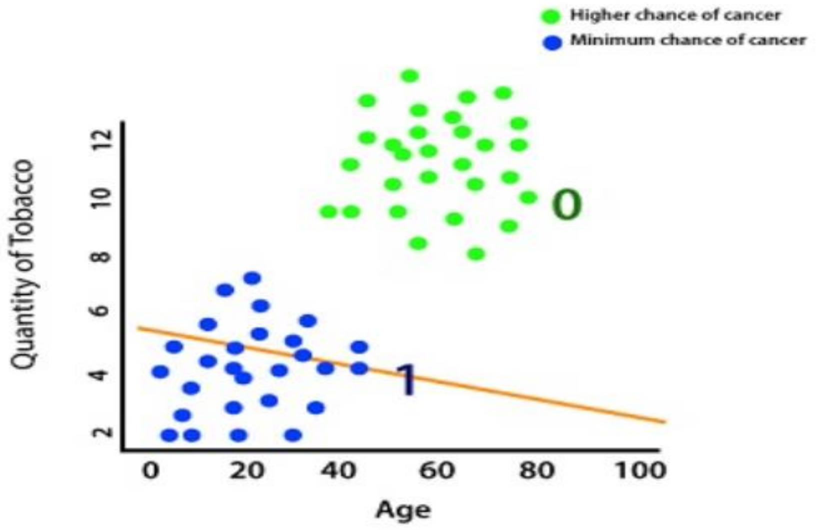


- So that the person who has higher in the age and quantity of tobaccos which he takes have higher chances of cancer and if a person has minimum age and quantity of tobaccos which he takes, have a minimum chance of cancer.

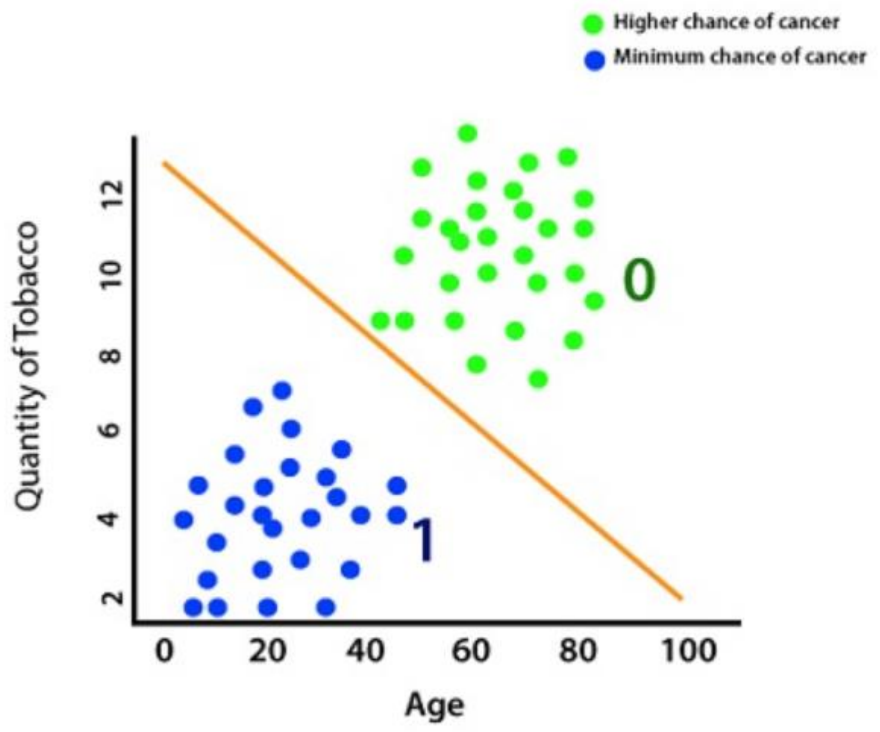
- Every green point, which indicates a higher chance of cancer, was initially assigned a label of zero and every blue point, which indicates a lower chance of cancer, was initially assigned a label of one.

- So, we will start with a random model that will not correctly classify our data, but then the model will be trained through some optimization algorithm.
- The model will be trained through many iterations until it reaches the parameter value, which can classify our data correctly.
- We use previously labeled data everything here is labeled one, and everything here is label zero.





- We use this labeled data to come up with a predictive model which classifies our data into two discrete categories.
- Using that model, we can now make a prediction on newly input data which don't have a label based on whether or not that point lies below the line or above it.



- Now, the biggest question is how the computer know-how to come up with this linear model does.
- For this purpose, we will calculate the error associated with this model and then readjust the parameters of the model to minimize the error and properly classify the data points.
- We will use **cross-entropy ()** function to find an error and **sigmoid gradient descent** to optimize the parameters.
- Let starts to implement the code in which we will see how cross-entropy function and sigmoid gradient descent is used.

Training of Perceptron Model

- Training of the perceptron model is similar to the linear regression model.
- We initialize our neural model, which have two input node in the input layer and a single output node with a sigmoid activation function.
- When we plot our model to the data, we found that it does not fit our data well.
- We need to train this model so that the model has the optimal weight and bias parameters and fit this data.

Step 1

- In the first step, the criterion by which we will compute the error of our model is recall Cross entropy. Our loss function will be measured on the basis of binary cross entropy loss (BCELoss) because we are dealing with only two classes. It is import from NN module.

Step 2

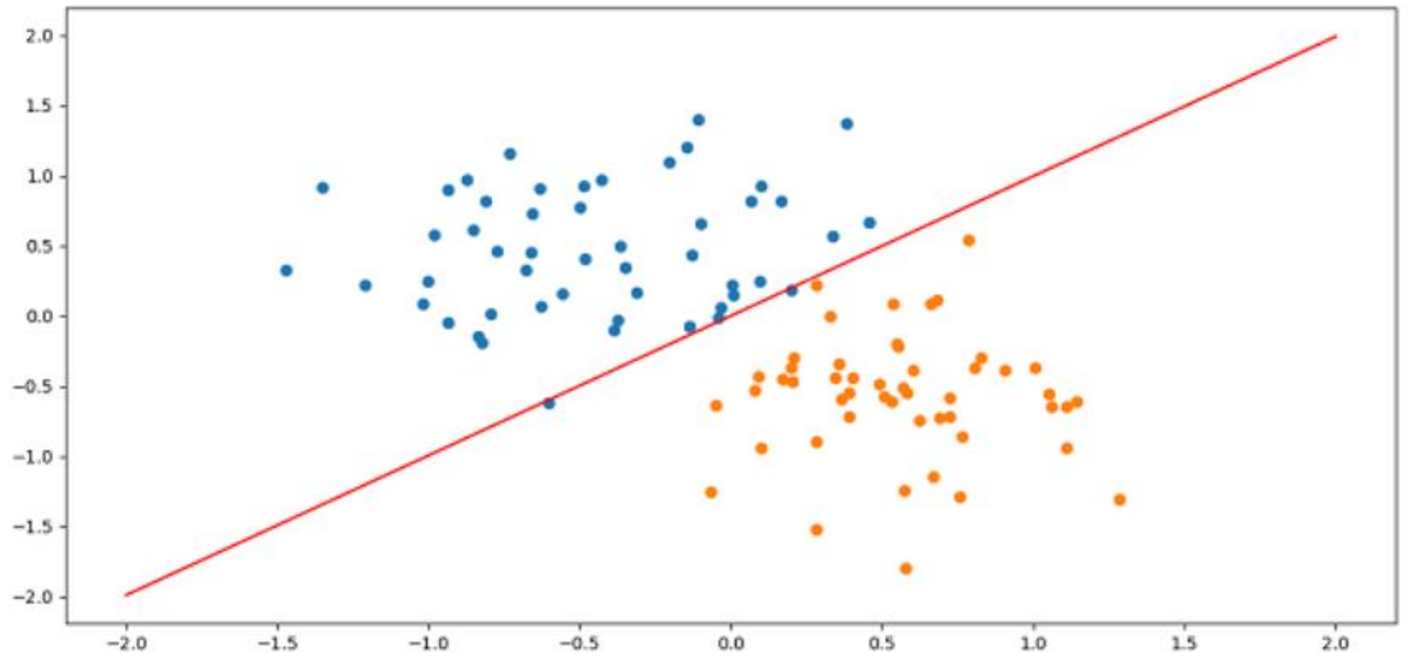
- Now, our next step is to update parameters using optimizer. So we define the optimizer which is used gradient descent algorithm (stochastic gradient descent).

Step 3

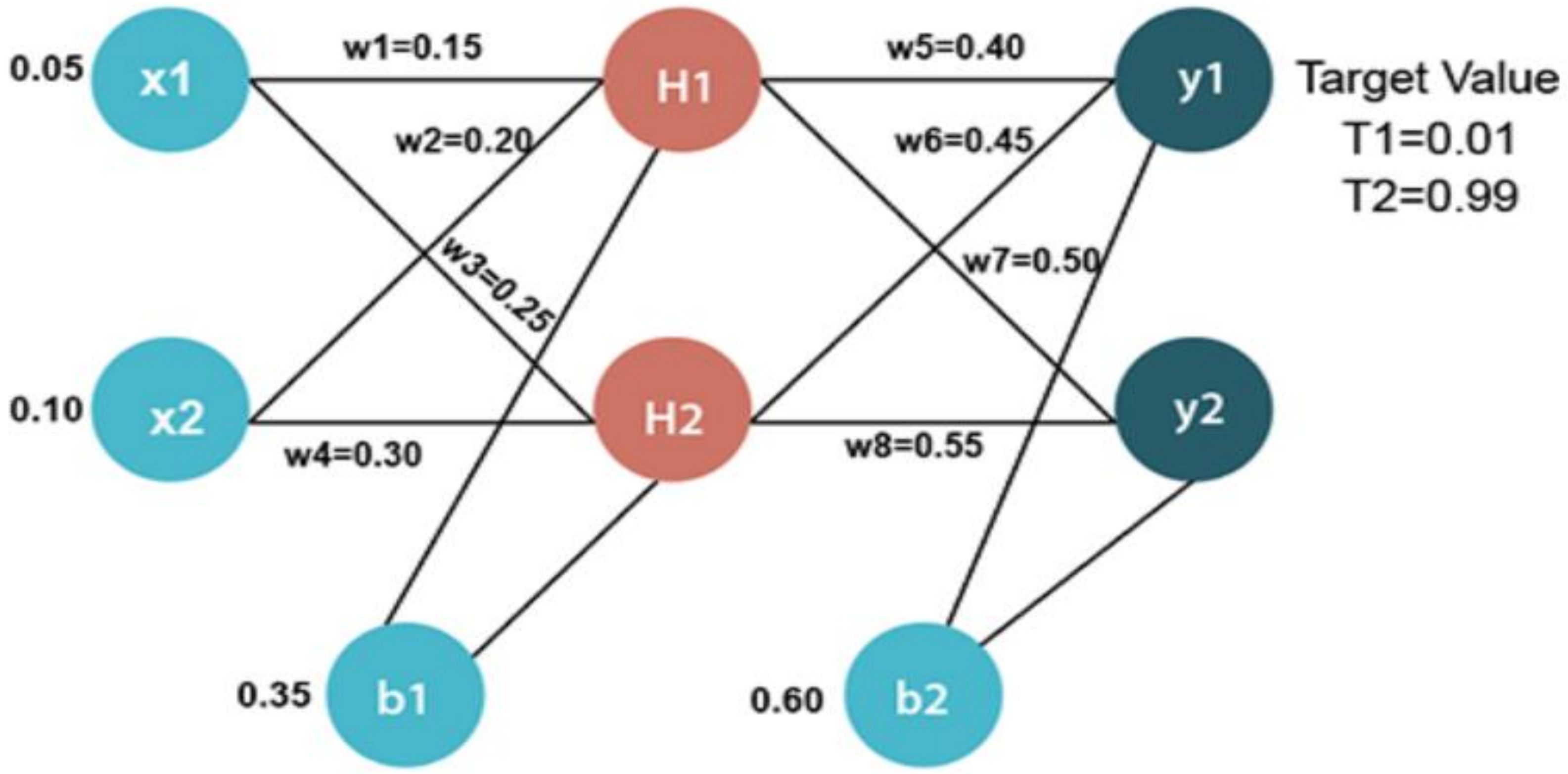
- Now, we will train our model for a specified no of epochs as we have done in linear model.

*Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1

- Now, at last we plot our new linear model by simply calling plotfit() method.



Backpropagation Process in Deep Neural Network



Input values

X1=0.05

X2=0.10

Initial weight

W1=0.15	w5=0.40
W2=0.20	w6=0.45
W3=0.25	w7=0.50
W4=0.30	w8=0.55

Bias Values

b1=0.35 b2=0.60

Target Values

T1=0.01

T2=0.99

Now, we first calculate the values of H1 and H2 by a forward pass.

Forward Pass

To find the value of H1 we first multiply the input value from the weights as

$$H1 = x1 \times w_1 + x2 \times w_2 + b1$$

$$H1 = 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35$$

$$\mathbf{H1 = 0.3775}$$

To calculate the final result of H1, we performed the sigmoid function as

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{H1}}}$$

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{0.3775}}}$$

$$\mathbf{H1_{final} = 0.593269992}$$

We will calculate the value of H2 in the same way as H1

$$H2 = x1 \times w_3 + x2 \times w_4 + b1$$

$$H2 = 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35$$

$$\mathbf{H2 = 0.3925}$$

To calculate the final result of H1, we performed the sigmoid function as

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{H2}}}$$

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{0.3925}}}$$

$$\mathbf{H2_{final} = 0.596884378}$$

Now, we calculate the values of y1 and y2 in the same way as we calculate the H1 and H2.

To find the value of y1, we first multiply the input value i.e., the outcome of H1 and H2 from the weights as

$$y1=H1\times w_5+H2\times w_6+b2$$

$$y1=0.593269992\times 0.40+0.596884378\times 0.45+0.60$$

$$\mathbf{y1=1.10590597}$$

To calculate the final result of y1 we performed the sigmoid function as

$$y1_{\text{final}} = \frac{1}{1 + \frac{1}{e^{y1}}}$$

$$y1_{\text{final}} = \frac{1}{1 + \frac{1}{e^{1.10590597}}}$$

$$\mathbf{y1_{final} = 0.75136507}$$

We will calculate the value of y2 in the same way as y1

$$y2 = H1 \times w_7 + H2 \times w_8 + b2$$

$$y2 = 0.593269992 \times 0.50 + 0.596884378 \times 0.55 + 0.60$$

$$\mathbf{y2 = 1.2249214}$$

To calculate the final result of H1, we performed the sigmoid function as

$$y2_{\text{final}} = \frac{1}{1 + \frac{1}{e^{y2}}}$$

$$y2_{\text{final}} = \frac{1}{1 + \frac{1}{e^{1.2249214}}}$$

$$\mathbf{y2_{final} = 0.772928465}$$

Our target values are 0.01 and 0.99. Our y1 and y2 value is not matched with our target values T1 and T2.

Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs. The total error is calculated as

$$\mathrm{E_{total}} = \Sigma \frac{1}{2}(\mathrm{target} - \mathrm{output})^2$$

So, the total error is

$$= \frac{1}{2}(\mathrm{t1} - \mathrm{y1_{final}})^2 + \frac{1}{2}(\mathrm{T2} - \mathrm{y2_{final}})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2$$

$$= 0.274811084 + 0.0235600257$$

$$\mathbf{E_{total} = 0.29837111}$$

Now, we will backpropagate this error to update the weights using a backward pass.

Backward pass at the output layer

To update the weight, we calculate the error correspond to each weight with the help of a total error. The error on weight w is calculated by differentiating total error with respect to w.

$$\mathrm{Error}_{w} = \frac{\partial E_{total}}{\partial w}$$

We perform backward process so first consider the last weight w5 as

$$\mathrm{Error}_{w5} = \frac{\partial E_{total}}{\partial w5} \dots\dots\dots..(1)$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2 \dots\dots\dots..(2)$$

From equation two, it is clear that we cannot partially differentiate it with respect to w5 because there is no any w5. We split equation one into multiple terms so that we can easily differentiate it with respect to w5 as

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1\_final}{\partial y1} \times \frac{\partial y1}{\partial w5} \dots\dots\dots\dots\ (3)$$

Now, we calculate each term one by one to differentiate E_{total} with respect to w5 as

$$\frac{\partial E_{total}}{\partial y1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial y1_{final}}$$

$$= 2 \times \frac{1}{2} \times (T1 - y1_{final})^{2-1} \times (-1) + 0$$

$$= -(T1 - y1_{final})$$

$$= -(0.01 - 0.75136507)$$

$$\boldsymbol{\frac{\partial E_{total}}{\partial y1_{final}} = 0.74136507 \ldots\ldots\ldots.(4)}$$

$$y1_{final} = \frac{1}{1 + e^{-y1}} \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$\frac{\partial y1_{final}}{\partial y1} = \frac{\partial(\frac{1}{1 + e^{-y1}})}{\partial y1}$$

$$= \frac{e^{-y1}}{(1 + e^{-y1})^2}$$

$$= e^{-y1} \times (y1_{final})^2 \ldots\ldots\ldots\ldots\ldots(6)$$

$$y1_{final} = \frac{1}{1 + e^{-y1}}$$

$$e^{-y1} = \frac{1 - y1_{final}}{y1_{final}} \dots\dots\dots\dots\dots (7)$$

Putting the value of e^{-y} in equation (5)

$$= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2$$

$$= y1_{final} \times (1 - y1_{final})$$

$$= 0.75136507 \times (1 - 0.75136507)$$

$$\frac{\partial \mathbf{y1_{final}}}{\partial \mathbf{y1}} = \mathbf{0.186815602} \dots\dots\dots. \mathbf{(8)}$$

$$y1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \dots\dots\dots\dots\dots\dots\ (9)$$

$$\frac{\partial y1}{\partial w5} = \frac{\partial (H1_{final} \times w5 + H2_{final} \times w6 + b2)}{\partial w5}$$

$$= H1_{final}$$

$$\frac{\partial \mathbf{y1}}{\partial \mathbf{w5}} = \mathbf{0.596884378} \dots\dots\dots \mathbf{(10)}$$

So, we put the values of $\frac{\partial \mathrm{E_{total}}}{\partial \mathrm{y1_{final}}}$, $\frac{\partial \mathrm{y1_{final}}}{\partial \mathrm{y1}}$, and $\frac{\partial \mathrm{y1}}{\partial \mathrm{w5}}$ in equation no (3) to find the final result.

$$\frac{\partial \mathrm{E_{total}}}{\partial \mathrm{w5}} = \frac{\partial \mathrm{E_{total}}}{\partial \mathrm{y1_{final}}} \times \frac{\partial \mathrm{y1_{final}}}{\partial \mathrm{y1}} \times \frac{\partial \mathrm{y1}}{\partial \mathrm{w5}}$$

$$= 0.74136507 \times 0.186815602 \times 0.593269992$$

$$\mathbf{Error_{w5}} = \frac{\boldsymbol{\partial} \mathbf{E_{total}}}{\boldsymbol{\partial} \mathbf{w5}} = \mathbf{0.0821670407} \ldots \ldots \ldots (\mathbf{11})$$

Now, we will calculate the updated weight $\mathrm{w5_{new}}$ with the help of the following formula

$$\mathrm{w5_{new}} = \mathrm{w5} - \eta \times \frac{\partial \mathrm{E_{total}}}{\partial \mathrm{w5}} \text{ Here, } \eta = \text{learning rate} = 0.5$$

$$= 0.4 - 0.5 \times 0.0821670407$$

$$\mathbf{w5_{new}} = \mathbf{0.35891648} \ldots \ldots \ldots (\mathbf{12})$$

In the same way, we calculate $w6_{new}$,$w7_{new}$, and $w8_{new}$ and this will give us the following values

$w5_{new}$=0.35891648

$w6_{new}$=408666186

$w7_{new}$=0.511301270

$w8_{new}$=0.561370121

Backward pass at Hidden layer

Now, we will backpropagate to our hidden layer and update the weight w1, w2, w3, and w4 as we have done with w5, w6, w7, and w8 weights.

We will calculate the error at w1 as

$$\mathrm{Error}_{w1} = \frac{\partial E_{total}}{\partial w1}$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

From equation (2), it is clear that we cannot partially differentiate it with respect to w1 because there is no any w1. We split equation (1) into multiple terms so that we can easily differentiate it with respect to w1 as

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1\_final}{\partial H1} \times \frac{\partial H1}{\partial w1} \dots \dots \dots \dots (13)$$

Now, we calculate each term one by one to differentiate E_{total} with respect to w1 as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial H1} \dots \dots \dots \dots (14)$$

We again split this because there is no any $H1^{final}$ term in E^{toatal} as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}} \dots \dots \dots (15)$$

$\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ will again split because in E1 and E2 there is no H1 term. Splitting is done as

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}} \dots \dots \dots (16)$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final} \dots \dots \dots (17)$$

We again Split both $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ because there is no any y1 and y2 term in E1 and E2. We split it as

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1} \dots \dots \dots (18)$$

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2} \dots \dots \dots (19)$$

Now, we find the value of $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ by putting values in equation (18) and (19) as

From equation (18)

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{\text{fianl}}} \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

$$= \frac{\partial(\frac{1}{2}(T1 - y1_{\text{final}})^2)}{\partial y1_{\text{final}}} \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

$$= 2 \times \frac{1}{2}(T1 - y1_{\text{final}}) \times (-1) \times \frac{\partial y1_{\text{final}}}{\partial y1}$$

From equation (8)

$$= 2 \times \frac{1}{2}(0.01 - 0.75136507) \times (-1) \times \; 0.186815602$$

$$\boldsymbol{\frac{\partial E_1}{\partial y1} = 0.138498562} \; \ldots\ldots\ldots (20)$$

From equation (19)

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{\mathrm{fianl}}} \times \frac{\partial y2_{\mathrm{final}}}{\partial y2}$$

$$= \frac{\partial(\frac{1}{2}(T2 - y2_{\mathrm{final}})^2)}{\partial y2_{\mathrm{final}}} \times \frac{\partial y2_{\mathrm{final}}}{\partial y2}$$

$$= 2 \times \frac{1}{2}(T2 - y2_{\mathrm{final}}) \times (-1) \times \frac{\partial y2_{\mathrm{final}}}{\partial y2} \dots \dots \dots (21)$$

$$y2_{\mathrm{final}} = \frac{1}{1 + e^{-y2}} \dots \dots \dots \dots \dots \dots \; (22)$$

$$\frac{\partial y2_{\mathrm{fianl}}}{\partial y2} = \frac{\partial(\frac{1}{1 + e^{-y2}})}{\partial y2}$$

$$= \frac{e^{-y2}}{(1 + e^{-y2})^2}$$

$$= e^{-y2} \times (y2_{\mathrm{final}})^2 \dots \dots \dots \dots \dots (23)$$

$$y2_{\mathrm{final}} = \frac{1}{1 + e^{-y2}}$$

$$e^{-y2} = \frac{1 - y2_{\mathrm{final}}}{y2_{\mathrm{final}}} \dots \dots \dots \dots \dots (24)$$

Putting the value of e^{-y2} in equation (23)

$$= \frac{1 - y2_{\mathrm{final}}}{y2_{\mathrm{final}}} \times (y2_{\mathrm{final}})^2$$

$$= y2_{\mathrm{final}} \times (1 - y2_{\mathrm{final}})$$

$$= 0.772928465 \times (1 - 0.772928465)$$

$$\frac{\boldsymbol{\partial y2_{\mathrm{fianl}}}}{\boldsymbol{\partial y2}} = \mathbf{0.175510053 \ldots \ldots \ldots (25)}$$

From equation (21)

$$= 2 \times \frac{1}{2}(0.99 - 0.772928465) \times (-1) \times 0.175510053$$

$$\frac{\boldsymbol{\partial E_1}}{\boldsymbol{\partial y1}} = \mathbf{-0.0380982366126414 \ldots \ldots \ldots (26)}$$

Now from equation (16) and (17)

$$\frac{\partial E_1}{\partial H1_{\text{final}}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{\text{final}}}$$

$$= 0.138498562 \times \frac{\partial (H1_{\text{final}} \times w_5 + H2_{\text{final}} \times w_6 + b2)}{\partial H1_{\text{final}}}$$

$$= 0.138498562 \times \frac{\partial (H1_{\text{final}} \times w_5 + H2_{\text{final}} \times w_6 + b2)}{\partial H1_{\text{final}}}$$

$$= 0.138498562 \times w5$$

$$= 0.138498562 \times 0.40$$

$$\mathbf{\frac{\partial E_1}{\partial H1_{\text{final}}} = 0.0553994248 \ldots\ldots\ldots (27)}$$

$$\frac{\partial E_2}{\partial H1_{\text{final}}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_\text{final}}$$

$$= -0.0380982366126414 \times \frac{\partial (H1_{\text{final}} \times w_7 + H2_{\text{final}} \times w_8 + b2)}{\partial H1_{\text{final}}}$$

$$= -0.0380982366126414 \times w7$$

$$= -0.0380982366126414 \times 0.50$$

$$\mathbf{\frac{\partial E_2}{\partial H1_{\text{final}}} = -0.0190491183063207 \ldots\ldots\ldots (28)}$$

Put the value of $\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ in equation (15) as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}}$$

$$= 0.0553994248 + (-0.0190491183063207)$$

$$\boldsymbol{\frac{\partial E_{total}}{\partial H1_{final}} = 0.0364908241736793 \ldots \ldots \ldots (29)}$$

We have $\frac{\partial E_{total}}{\partial H1_{final}}$; we need to figure out $\frac{\partial H1\_final}{\partial H1}$, $\frac{\partial H1}{\partial w1}$ as

$$\frac{\partial H1_{final}}{\partial H1} = \frac{\partial(\frac{1}{1 + e^{-H1}})}{\partial H1}$$

$$= \frac{e^{-H1}}{(1 + e^{-H1})^2}$$

$$e^{-H1} \times (H1_{final})^2 \ldots \ldots \ldots (30)$$

$$H1_{final} = \frac{1}{1 + e^{-H1}}$$

$$e^{-H1} = \frac{1 - H1_{final}}{H1_{final}} \ldots \ldots \ldots \ldots \ldots (31)$$

Putting the value of e^{-H1} in equation (30)

$$= \frac{1 - H1_{final}}{H1_{final}} \times (H1_{final})^2$$

$$= H1_{final} \times (1 - H1_{final})$$

$$= 0.593269992 \times (1 - 0.593269992)$$

$$\boldsymbol{\frac{\partial H1_{final}}{\partial H1} = 0.2413007085923199}$$

We calculate the partial derivative of the total net input to H1 with respect to w1 the same as we did for the output neuron:

$$H1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \ldots \ldots \ldots \ldots \ldots \ldots (32)$$

$$\frac{\partial y1}{\partial w1} = \frac{\partial (x1 \times w1 + x2 \times w3 + b1 \times 1)}{\partial w1}$$

$$= x1$$

$$\frac{\partial \mathbf{H1}}{\partial \mathbf{w1}} = \mathbf{0.05} \ldots\ldots\ldots(\mathbf{33})$$

So, we put the values of $\frac{\partial E_{total}}{\partial H1_{final}}$, $\frac{\partial H1_{final}}{\partial H1}$, and $\frac{\partial H1}{\partial w1}$ in equation (13) to find the final result.

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1_{final}}{\partial H1} \times \frac{\partial H1}{\partial w1}$$

$$= 0.0364908241736793 \times 0.2413007085923199 \times 0.05$$

$$\mathbf{Error_{w1}} = \frac{\partial \mathbf{E_{total}}}{\partial \mathbf{w1}} = \mathbf{0.000438568} \ldots\ldots\ldots.(\mathbf{34})$$

Now, we will calculate the updated weight $w1_{new}$ with the help of the following formula

$$w1_{new} = w1 - \eta \times \frac{\partial E_{total}}{\partial w1} \text{ Here } \eta = \text{learning rate} = 0.5$$

$$= 0.15 - 0.5 \times 0.000438568$$

$$\mathbf{w1_{new} = 0.149780716 \ldots\ldots\ldots (35)}$$

In the same way, we calculate $w2_{new}$,$w3_{new}$, and w4 and this will give us the following values

$w1_{new}$=0.149780716

$w2_{new}$=0.19956143

$w3_{new}$=0.24975114

$w4_{new}$=0.29950229