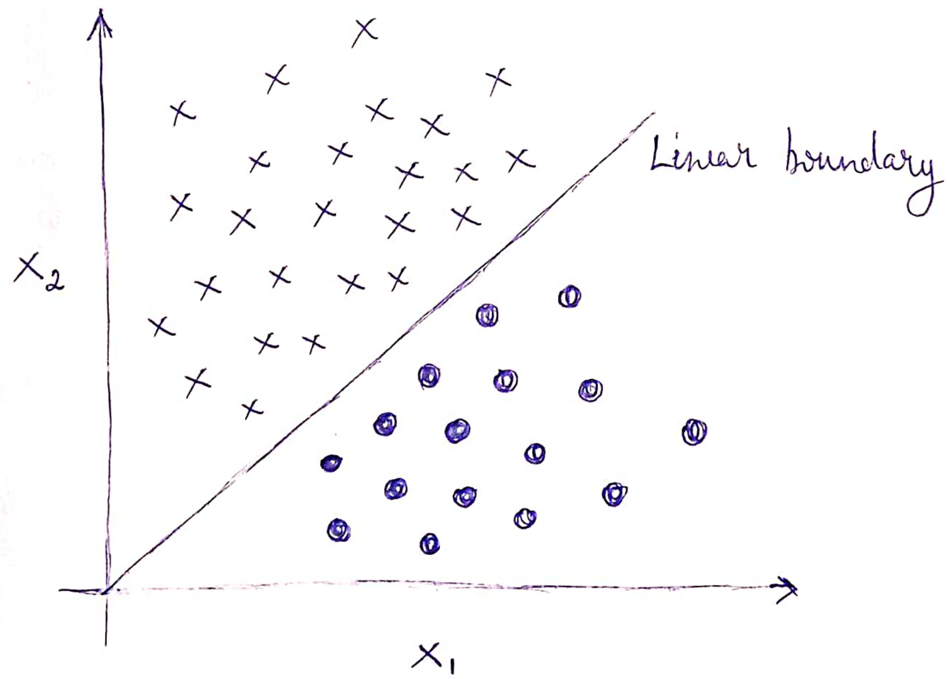
③ The concept of Linear Separability applies to binary classification. Linear separability is a property of two sets of points.

The linear separability of the network is based on the decision-boundary line. If there exist weight for which the training input vectors having a positive (correct) response or lie on one side of the decision boundary and all the other vectors having negative, -1, response lies on the other side of the decision boundary then we can conclude the problem as "Linearly Separable".



Class A (x) and class B (•) are linearly separated from each other.

④ Prior Probability

$$P(\text{class} = +) = \frac{2}{7}$$

$$P(\text{class} = -) = \frac{5}{7}$$

Instances: Feature 1 \ Class	+	−
T	$\frac{1}{2}$	$\frac{3}{5}$
F	$\frac{1}{2}$	$\frac{2}{5}$

Feature 2 \ Class	+	−
T	$\frac{2}{2} = 1$	0
F	0	$\frac{5}{5} = 1$

New instance = { Feature 1 = T, Feature 2 = T }

Now,

$P(\text{Class} = + \mid \text{New instance})$

$\Rightarrow P(+)\,P(\text{Feature 1} = T \mid +)\,P(\text{Feature 2} = T \mid +)$

$\Rightarrow \frac{2}{7} \times \frac{1}{2} \times 1$

$\Rightarrow \frac{1}{7} = 0.1429$

$P(\text{class} = - \mid \text{New instance})$

$\Rightarrow P(-)\, P(\text{Feature } 2 = T \mid -)\, P(\text{Feature } 1 = T \mid -)$

$\Rightarrow \frac{5}{7} \times 0 \times \frac{3}{5}$

$\Rightarrow 0$

Since, $P(\text{class} = + \mid \text{New instance}) > P(\text{class} = - \mid \text{New instance})$

Therefore, the class for instance 8 with Feature 1 = T and Feature 2 = T is + class.

---

(5)

x	y	xy	x^2
0	1	0	0
1	2	2	1
2	2	4	4
3	3	9	9
4	3	12	16
5	4	20	25
15	15	47	55

By the method of least square regression

$$b_1 = \frac{n\Sigma xy - \Sigma x \Sigma y}{n \Sigma x^2 - (\Sigma x)^2}$$

$$= \frac{6 \times 47 - 15 \times 15}{6 \times 55 - 15^2}$$

$$= 0.5429$$

$$b_0 = \frac{1}{n}(\Sigma y - b_1 \Sigma x)$$

$$= \frac{1}{6}(15 - 0.5429 \times 15)$$

$$= 1.14285$$

The regression line

$$\boxed{y = 0.543x + 1.1428}$$

When $x = 15$,

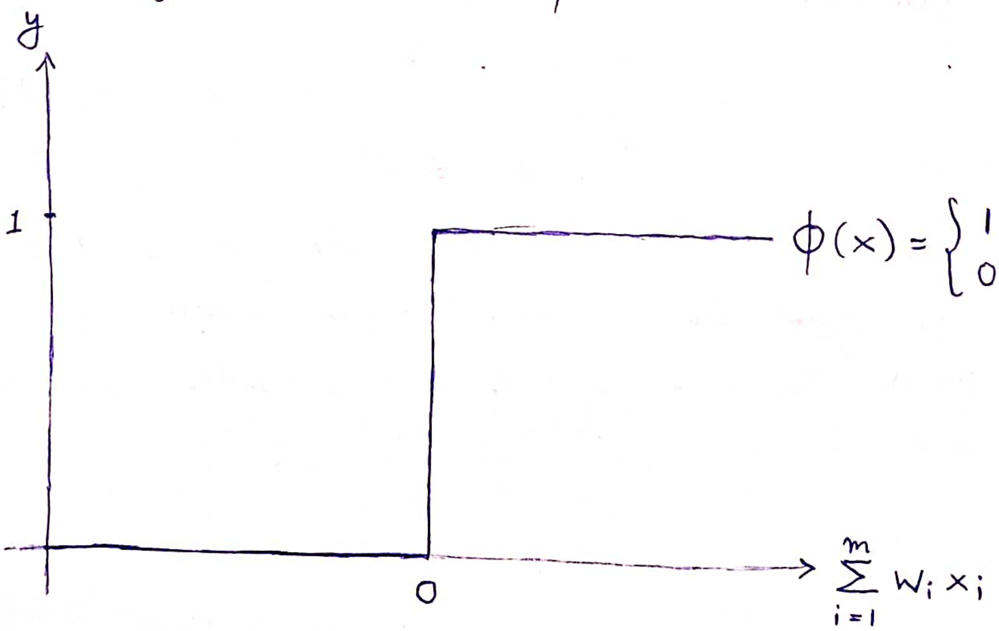
$$y = 0.543 \times 15 - 1.1428$$

$$\boxed{y = 9.2857}$$

(8) In Artificial Neural Network, the value of net input can be anything from -inf to +inf. The neuron doesn't really know how to bound to value and thus is not able to decide the firing pattern. An activation function results in an output signal only when an input signal exceeding a specific threshold value comes as an input. It is similar to the biological neuron which transmits the signal only when the total input signal meets the firing threshold.
Different types of activation functions for firing a neuron are -

1) Threshold / Step Function

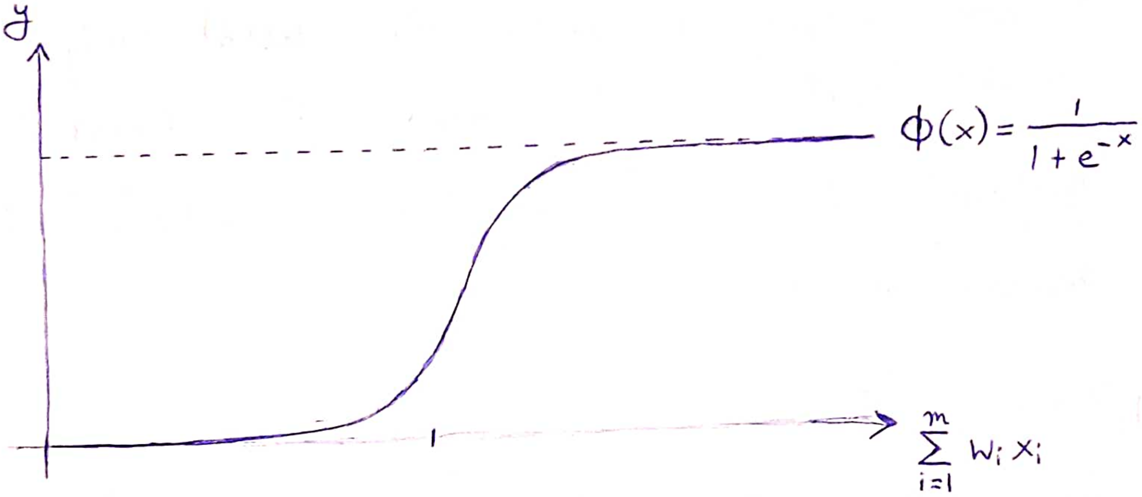
It is a commonly used activation function. It gives 1 as output of the input either 0 or positive. If the input is negative, it gives 0 as output.



$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

2) Sigmoid Function

The need for sigmoid function stems from the fact that many learning algorithms require the activation function to be differentiable and hence continuous. The biggest advantage is that it is non-linear. It can be used when predicting probabilities. The function ranges from 0 to 1 having an S-shape. It is defined as $\frac{1}{1+e^{-x}}$

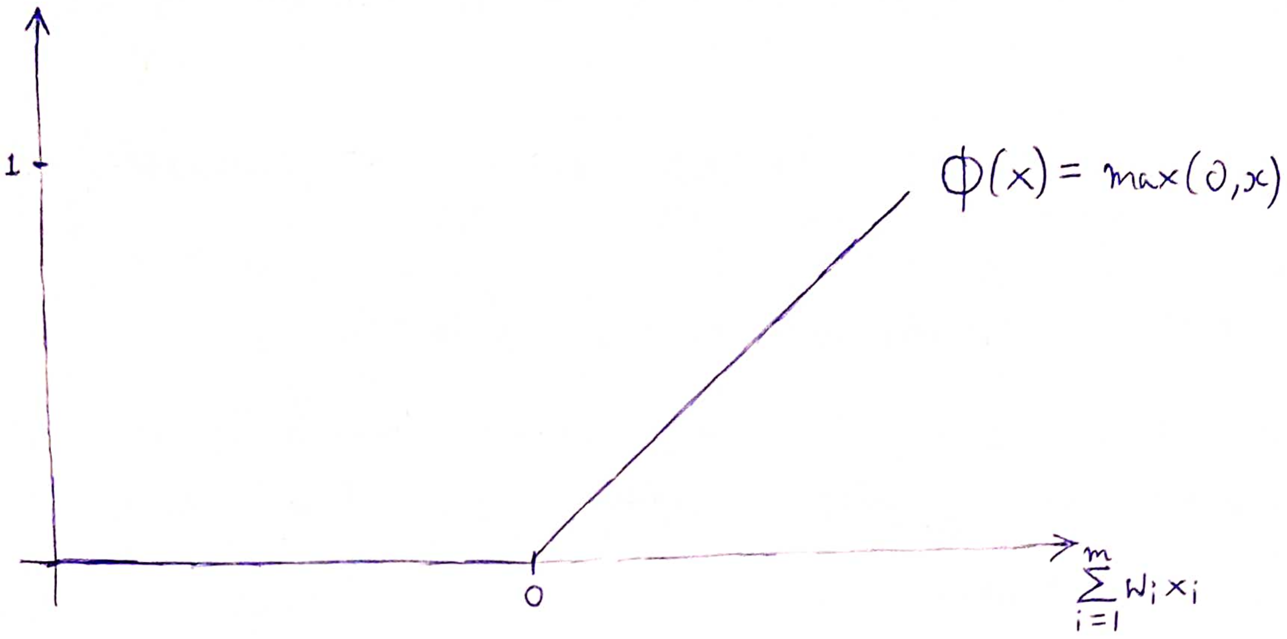


3) ReLu (or Rectifier) Function

ReLu function is the Rectified Linear Unit. It is derivied as

$$f(x) = \max(0, x)$$
$$= \begin{cases} x, & x \geq 0 \\ 0, & x \leq 0 \end{cases}$$

This means that $f(x)$ is zero when x is less than zero and $f(x)$ is equal to x when x is above or equal to zero. The main advantage of using the ReLu function over others is that it does not activate all the neurons at the same time.

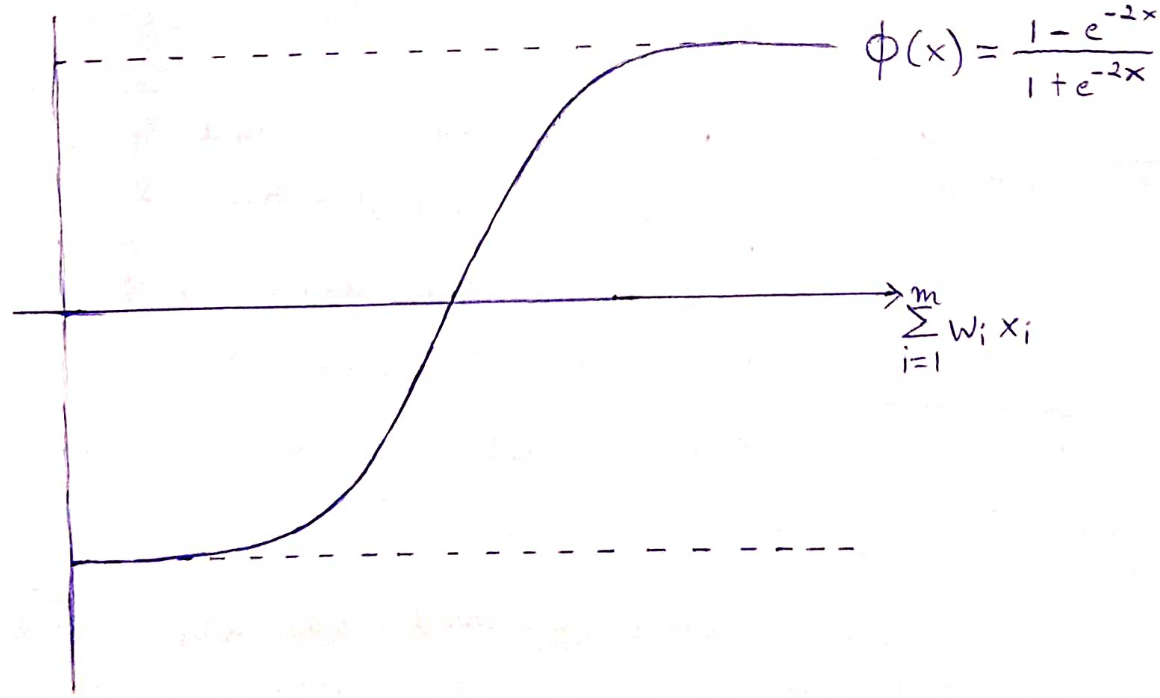


4) Hyperbolic Tangent Function

It is bipolar in nature. It is a widely adopted activation function for a special type of neural network known as Backpropagation Network. It is of the form of

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

It is similar to bipolar sigmoid function.



(9) The Perceptron learning algorithm is inspired by the information processing of a single neutral cell called a neuron.
The perceptron receives input from signals from examples of training data that we weight and combined in a linear equation called the activation.

The activation is then transformed into an output value or prediction using a transfer function, such as the step transfer function.

Perceptron consists of -

1) <u>Input</u> - All the feature becomes the input for a perceptron

$[x_1, x_2, x_3 \ldots\ldots, x_n]$

2) <u>Weights</u> - are the values that are computed over the time of training the model.

$[w_1, w_2, w_3, \ldots\ldots\ldots, w_n]$

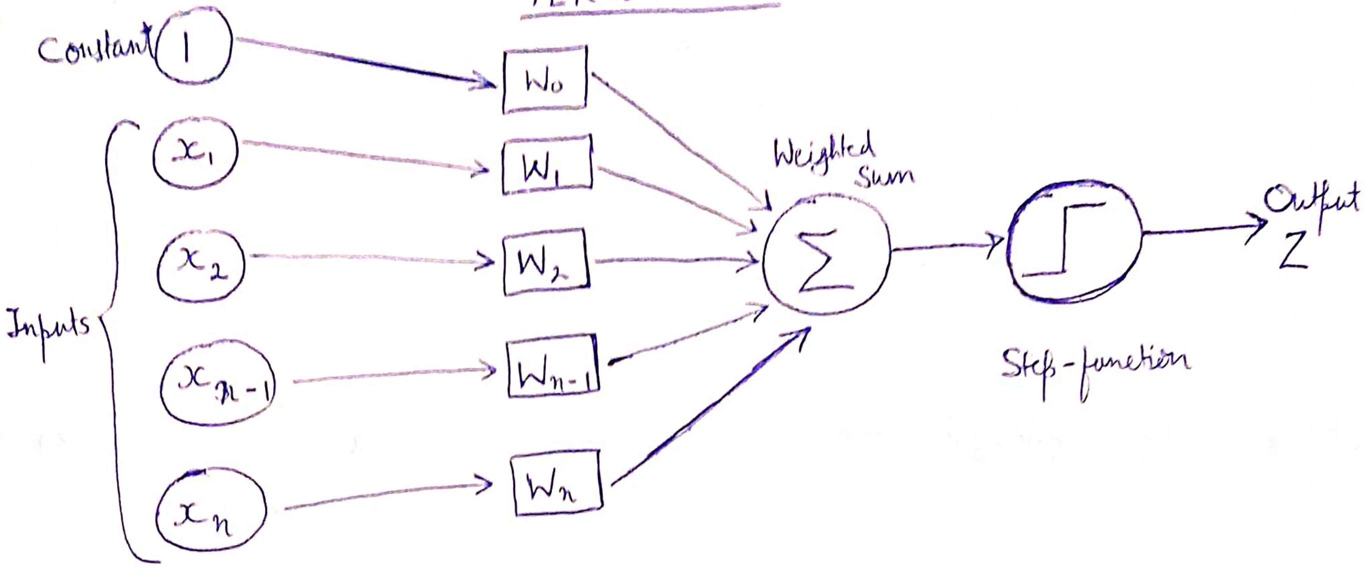
3) <u>Bias</u> - A bias neuron allows a classifier to shift the decision boundary left or right.

4) <u>Weighted Summation</u> - is the sum of value that we get after the multiplication of each weight $[w_n$ associated the each feature value $[x_n]$.

5) <u>Activation Function</u> - the role of activation function is to make neural networks non-linear.

6) <u>Output</u> - The weighted summation is passed to the step / activation function and whatever value we get after compution is the predicted output.

PERCEPTRON



$$Z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} W_i x_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} W_i x_i < \theta \end{cases}$$

Z - output
x - input
W - weights
n - no. of inputs
θ - threshold for step function

The weights of the perceptron algorithm must be estimated from your training data using stochastic gradient descent.

(10) A loss function is a function that compares the target and predicted output values, measures how well the neural network models the training data.
When training, we aim to minimize this loss between the predicted and target outputs.

The 2 major types of loss functions are –

1) Regression Loss Functions – used in regression neural networks
E.g. Mean Squared Error, Mean Absolute Error

2) Classification Loss Function – used in classification neural networks
E.g. Binary cross-Entropy, Categorical Cross-Entropy.

Various Loss functions in neural networks are –

1) Mean Squared Error (MSE)

MSE finds the average of the squared differences between the target and predicted outputs.

$$\boxed{MSE = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2}$$

The difference is squared, which means it does not matter whether the predicted value is above or below the target value; however values with a large error are penalized. MSE is also a convex function with its clearly defined global minimum.
One disadvantage is that it is very sensitive to outliers.

2) Mean Absolute Error (MAE)

MAE finds the average of the absolute differences between the target and the predicted outputs.

$$\boxed{MAE = \frac{1}{n} \sum_{i=1}^{n} \left| y^{(i)} - \hat{y}^{(i)} \right|}$$

MAE is used in cases when the training data has a large number of outliers to mitigate the over-sensitivity to outliers (like in case of MSE).

Its disadvantage is that as the average distance approaches 0, gradient descent optimization will not work, as the function's derivative at 0 is undefined.

3) Binary Cross Entropy / Log loss

It is a loss function in binary classification models.

$$\boxed{CE\ Loss = \frac{1}{n} \sum_{i=1}^{n} -(y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p))}$$

4) Categorical Cross-Entropy Loss

In cases where the number of classes is greater than 2, we utilize categorical cross-entropy.

$$\boxed{CE\ Loss = -\frac{1}{n} \sum_{i=1}^{N} \sum_{j=1}^{N} y_{ij} \cdot \log(p_{ij})}$$