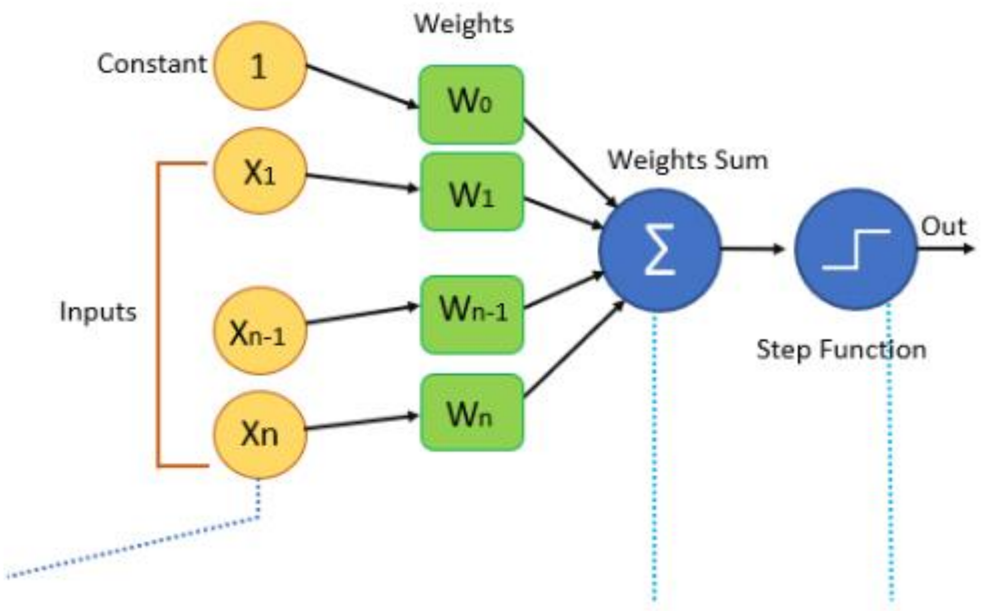
Unit-1

Perceptron Learning Algorithm

Perceptron Algorithm is used in a **supervised machine learning domain** for classification. In classification, there are two types of linear classification and no-linear classification. Linear classification is nothing but if we can classify the data set by drawing a simple straight line then it can be called a linear binary classifier. Whereas if we cannot classify the data set by drawing a simple straight line then it can be called a non-linear binary classifier.

Perceptron Algorithm Block Diagram



Let us see the terminology of the above diagram.

1. Input: All the features of the model we want to train the neural network will be passed as the input to it, Like the set of features [X1, X2, X3…..Xn]. Where n represents the total number of features and X represents the value of the feature.

2. Weights: Initially, we have to pass some random values as values to the weights and these values get automatically updated after each training error that is the values are generated during the training of the model. In some cases, weights can also be called as weight coefficients.

3. Weights Sum: Each input value will be first multiplied with the weight assigned to it and the sum of all the multiplied values is known as a weighted sum.

> Weights sum = ∑Wi * Xi (from i=1 to i=n) + (W0 * 1)

Step or Activation Function

Activation function applies step rule which converts the numerical value to 0 or 1 so that it will be easy for data set to classify. Based on the type of value we need as output we can change the activation function. Sigmoid function, if we want values to be between 0 and 1 we can use a sigmoid function that has a smooth gradient as well.

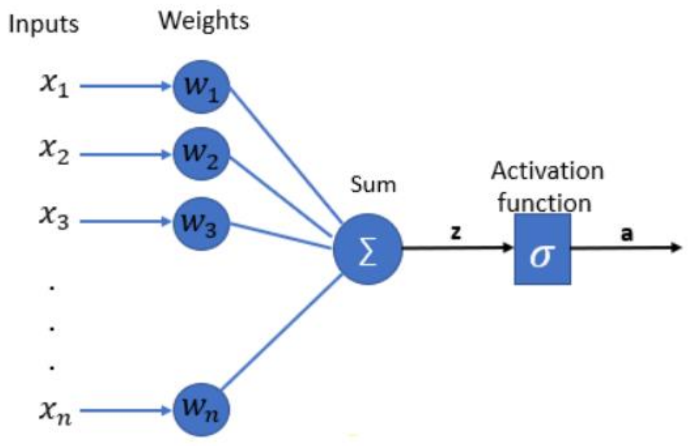
Sign function, if we want values to be +1 and -1 then we can use sign function. The hyperbolic tangent function is a zero centered function making it easy for the multilayer neural networks. Relu function is highly computational but it cannot process input values that approach zero. It is good for the values that are both greater than and less than a Zero.

Bias

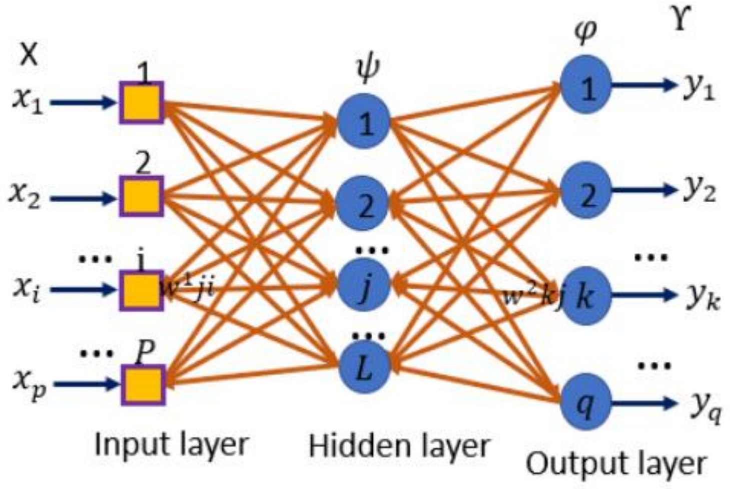
If you notice, we have passed value one as input in the starting and W0 in the weights section W0 is an element that adjusts the boundary away from origin to move the activation function left, right, up or down. since we want this to be independent of the input features, we add constant one in the statement so the features will not get affected by this and this value is known as Bias.

Perceptron algorithms can be divided into two types they are **single layer perceptrons** and multi-layer perceptron's. In single-layer perceptron's neurons are organized in one layer whereas in a multilayer perceptron's a group of neurons will be organized in multiple layers. Every single neuron present in the first layer will take the input signal and send a response to the neurons in the second layer and so on.

Single Layer Perceptron



Multi-Layer Perceptron



Perceptron Learning Steps

1. Features of the model we want to train should be passed as input to the perceptrons in the first layer.
2. These inputs will be multiplied by the weights or weight coefficients and the production values from all perceptrons will be added.
3. Adds the Bias value, to move the output function away from the origin.
4. This computed value will be fed to the activation function (chosen based on the requirement, if a simple perceptron system activation function is step function).

5. The result value from the activation function is the output value.

Features added with perceptron make in deep neural networks. Back Propagation is the most important feature in these.

Back Propagation

After performing the first pass (based on the input and randomly given inputs) error will be calculated and the back propagation algorithm performs an iterative backward pass and try to find the optimal values for weights so that the error value will be minimized. To minimize the error back propagation algorithm will calculate partial derivatives from the error function till each neuron's specific weight, this process will give us complete transparency from total error value to a specific weight that is responsible for the error.