Deep learning with TensorFlow 2.0 Keras basics

Hi, this is Sandeep, working as Software Engineer involved in Python, NLP, Machine Learning, Deep Leaning with TensorFlow 2.0 etc.

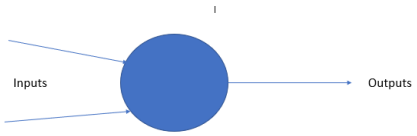
In this article I am exploring into I am exploring into Artificial Neural Networks. Let's starts…

Here are topics I will cover in this document.

1. Simple Perceptron Model
2. Multi-layer Perceptron Model
3. Activation Functions.

Simple Perceptron Model:

Below representation is Simple Perceptron Model:



So we have some function inside the neuron takes in these x values perform something on them and then outputs y so if ever X that function is just the sum then Y is equal to X1 plus X2 and this is a very very simplified perception model.

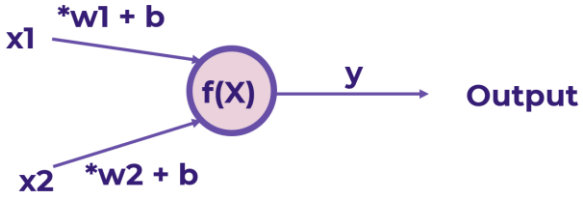
y = W*X + B

W> weight on data point x

X > Actual or Raw data point

B > bias term and

y represents Output values



y= (W1*X1+b1) + (W2*X2 + b2)

In mathematical representation of Simple Perceptron model is…

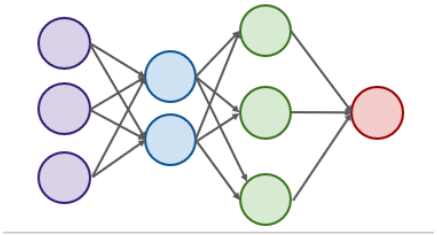
$$\hat{y} = \sum_{i=1}^{n} x_i w_i + b_i$$

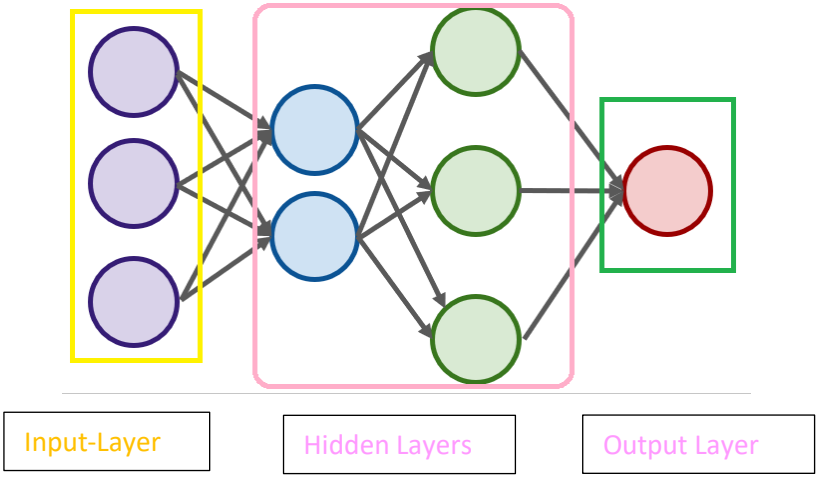
So far, we discussed simple single perceptron model, but this simple perceptron model is not good enough for more complicated systems or model to create.

Multi-Layer Perceptron model.

Here we must introduce Multi-Layer Perceptron model.

The outputs of one perceptron are directly fed into as inputs to another perceptron.



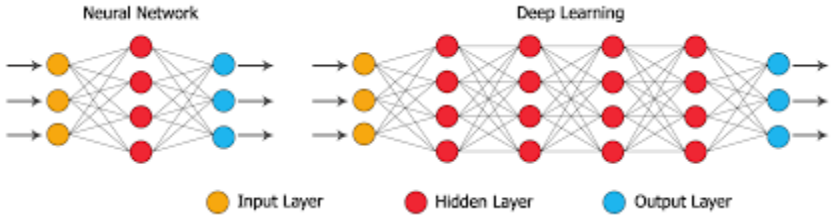


Terminology:

1. Input Layer: First layer that directly accepts real data values
2. Hidden Layer: Any layer between input and output layers

3. Output Layer: The final estimate of the output.

Neural Networks become **"deep neural networks"** if then contain 2 or more hidden layers.



Activation Function:

Activation function decides,

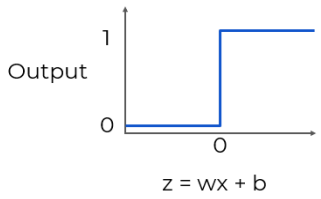
---

whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

The purpose of the activation function is to **introduce non-linearity** into the output of a neuron.

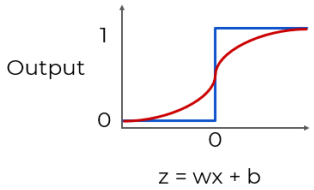
If we don't go with activation functions the it's a simple linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform complex tasks.

The simplest networks rely on a basic **step function** that outputs 0 or 1.



The output values always in between 0 or 1. This is much useful for classification model with binary level (0 or 1). There is just immediate cut off that splits between 0 and 1.

There is another function which **is Sigmoid Functions.**

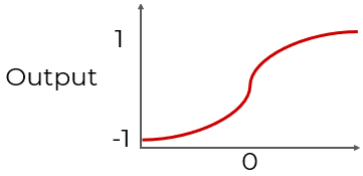


$$f(z) = \frac{1}{1 + e^{(-z)}}$$

This sigmoid function will be more sensitive to small changes.

Next **Hyperbolic Tangent**: tanh(z)

In Tangent function output relys between -1 and 1



$$\cosh x = \frac{\mathbf{e}^{x} + \mathbf{e}^{-x}}{2}$$

$$\sinh x = \frac{\mathbf{e}^{x} - \mathbf{e}^{-x}}{2}$$

$$\tanh x = \frac{\sinh x}{\cosh x}$$

Rectified Linear Unit (ReLu): This is actually a relatively simple function Max(0,z)

This function is very often we use in activation functions. Its gives a good performances while dealing with the issue of **Vanishing Gradient**.

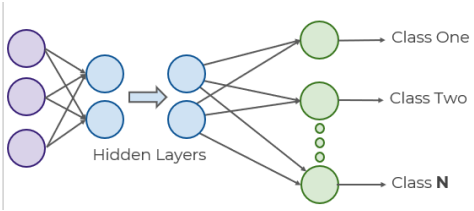
Cool!!, so far we discussed all the activation functions make sense for a single output, either a continuous label or trying to predict a binary classification (either a 0 or 1).

How about multi class classification problems?

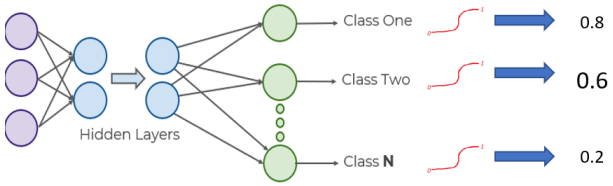
There are 2 types of multi-class situations

1. Non-Exclusive Classes
    a. A data point can have multiple classes/ categories assigned to it.
2. Mutually Exclusive Classes.
    a. Only one class per data point.

Organizing the multiple classes.



Sigmoid Function for Non-Exclusive Multi-classes



Sigmoid function

Keep in mind this allows each neuron to output independent of the other classes, allowing for a single data point fed into the function to have multiple classes assigned to it.

Mutually Exclusive Classes

In Mutually Exclusive Classes we can use **SoftMax function**.

SoftMax function calculates the probabilities distribution of the event over **N** different events.

This function will calculate the probabilities of each target class over all possible target classes.

The range will be 0 to 1, and **the sum of all the probabilities will be equal to one.** The model returns the probabilities of each class and the target class chosen will have the highest probability.

[Yellow, Blue, purple]

[0.1, 0.6, 0.3]