Deep Learning – Artificial Neural Networks

Introduction

Deep Learning (DL), which is a branch of Machine Learning (ML) and Artificial Intelligence (AI). Using Artificial Neural Networks (ANN) for processing data, DL needs a massive data to train a model. ANN is a network that simulates neurons in human brain. As shown in Figure 1, an ANN which has 2 or more layers in the hidden layer can be called as a Deep Neural Network (DNN).

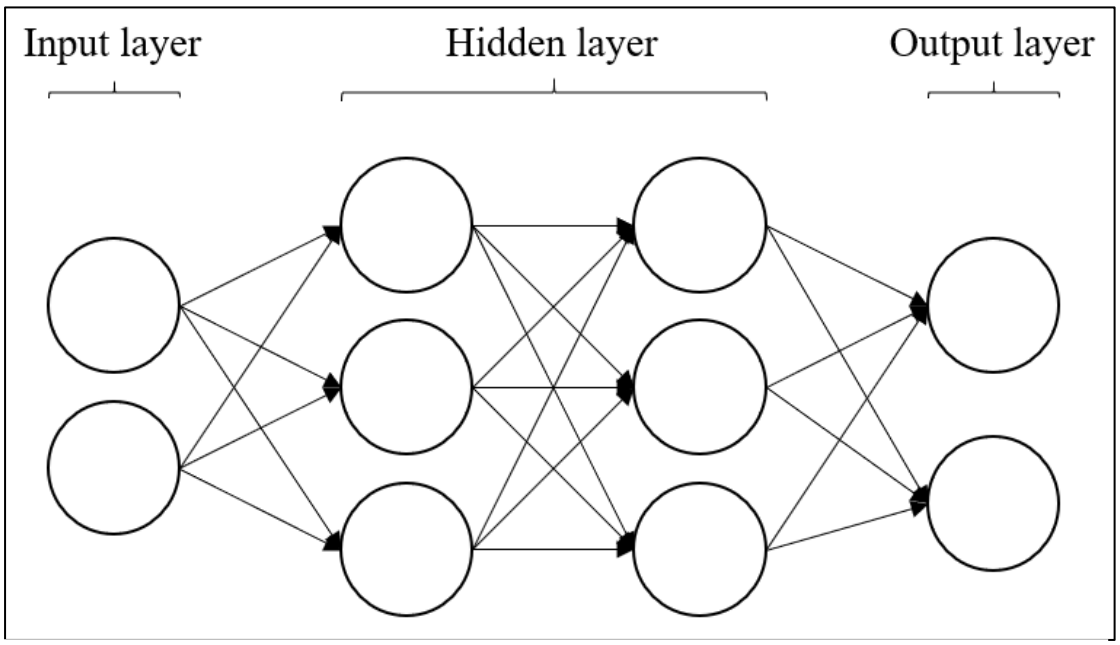

Figure. 1 Deep Neural Network, DNN

Perceptron

Perceptron is the basic component in a Neural Network (NN). As Shown in Figure 2, a perceptron also has an input layer and an output layer. There has an input tensor: $[x_1, x_2, \dots, x_n]$ for the input layer and every input has a corresponded weight w_i in the weights tensor: $[w_1, w_2, \dots, w_n]$, additionally adds a bias b so that we can easier find a solution.

For the output layer we first multiply every input x_i and weight w_i then sum up with the bias b, which comes out an equation: $z = (\sum_{i=1}^{n} x_i w_i) + b$. After we get z from the above equation we send it to the activation function $f()$ and returns the output: $output = f(z) = f((\sum_{i=1}^{n} x_i w_i) + b)$.

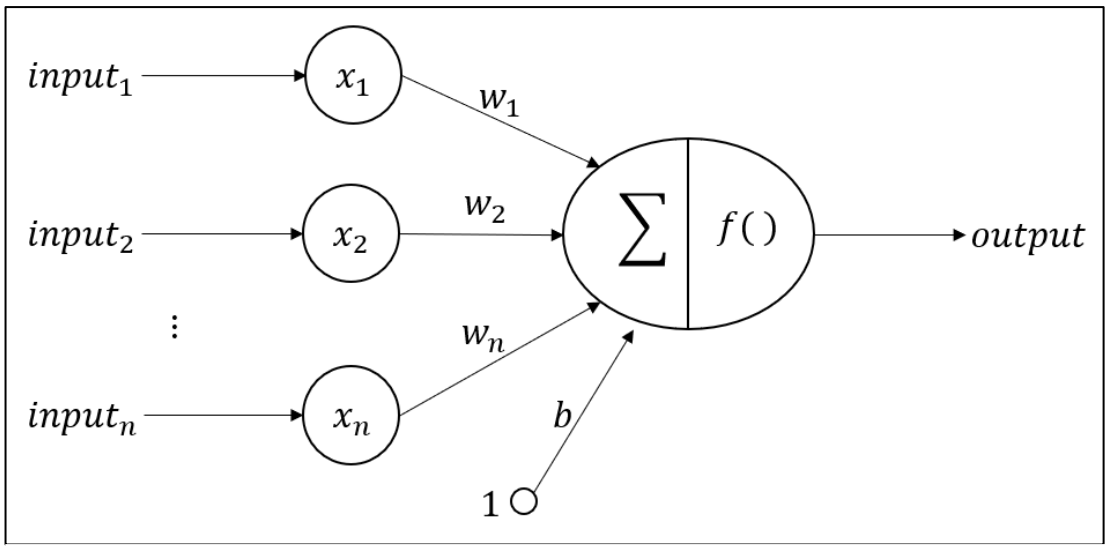

Figure 2. Perceptron

Activation Function

If there is no activation function in a NN no matter how many layers we pass through, it always constructs a linear function which can only deals with linear problems when fitting data. To solve a nonlinear problem, we use activation function to break the linearity which transfer the data into any range like 0 to 1 or -1 to 1 and that makes the NN fits more nonlinear problems. Here list some activation functions:

1. Sigmoid

As shown in Figure 3, a sigmoid function converts any data into a range of 0 to 1, and the equation:

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

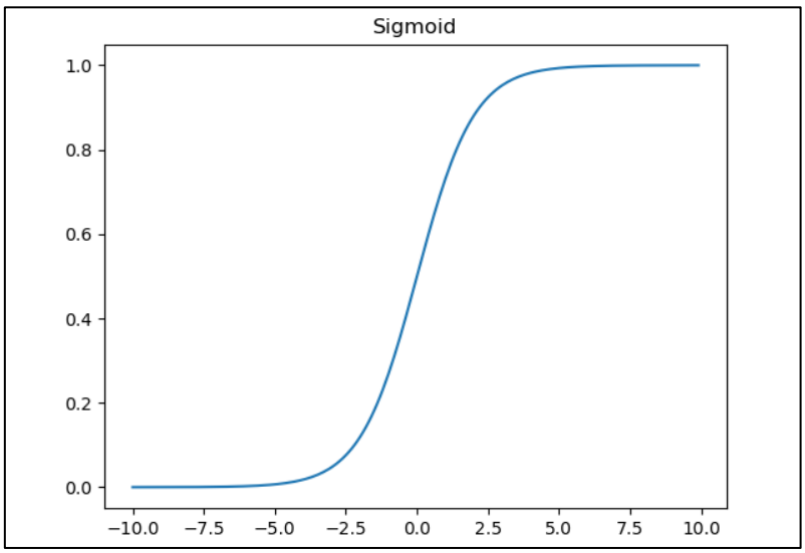

Figure 3. Sigmoid function

2. Rectified Linear Unit (ReLU)

As shown in Figure 4, the maximum value is only 0.25 when derivates the sigmoid function, this will encounter a problem called Vanishing Gradient Problem which happens when we do backpropagation using chain rule to calculate the gradient.

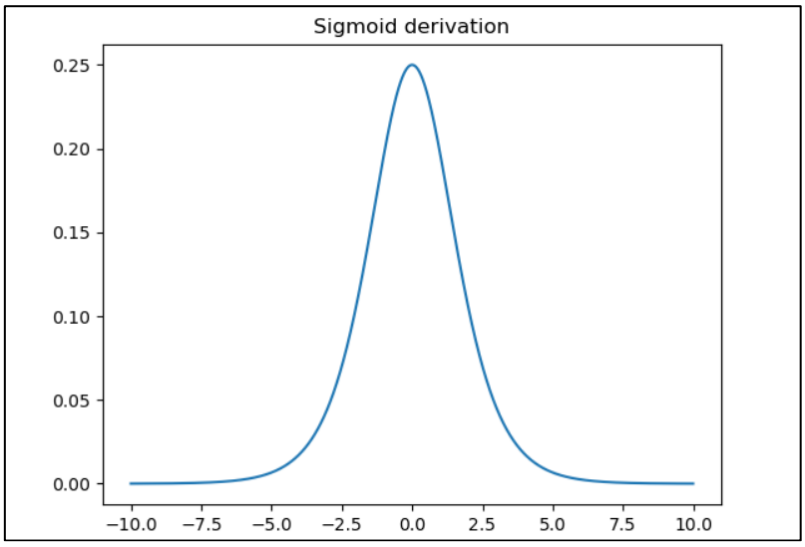

Figure 4. Derivative of Sigmoid function

ReLU function outputs 0 when the input is smaller than 0 else outputs a linear function which is the same as input. We can see that in Figure 5, the ReLU function has a derivation of 1 when it is greater than 0, which avoids the Vanishing Gradient Problem.

The equation:

$$f(x) = \max(x, 0)$$

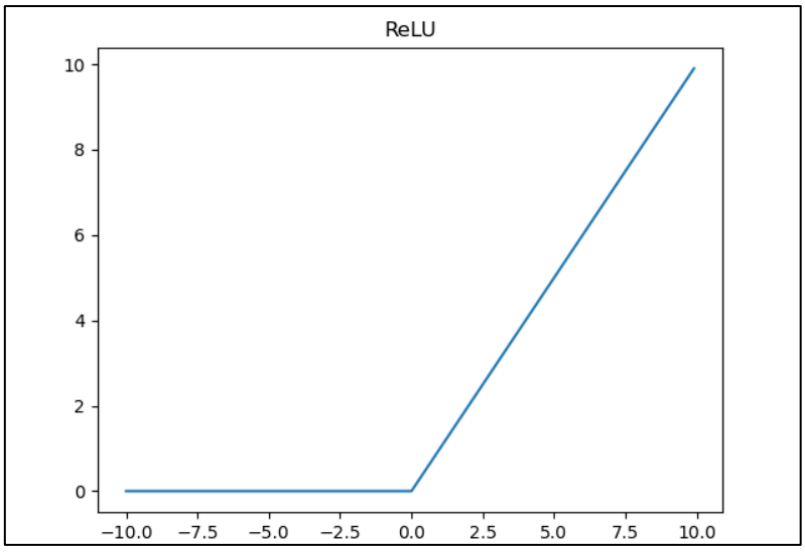

Figure 4. ReLU function

3. Hyperbolic Tangent (Tanh)

Tanh is a trigonometric function, Figure 5 shows that the function has an output in a range of -1 to 1 while Sigmoid and ReLU do not have negative values, the equation:

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

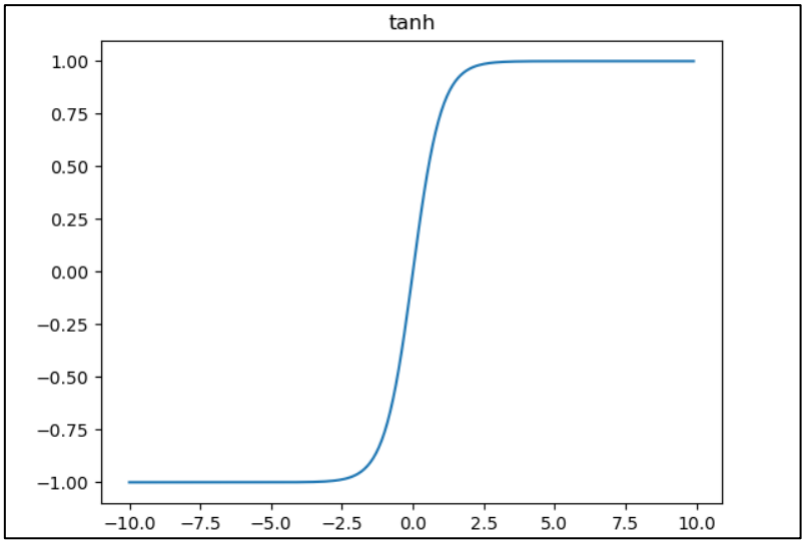

Figure 5. Tanh function

4. Softmax

Softmax function converts the inputs into real numbers between 0 and 1 which presents in a form of probability, the formula is like:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, 2, \dots, K \text{ and } z = (z_1, z_2, \dots, z_K) \in \mathbb{R}^K$$

Multilayer Perceptron (MLP)

With single perceptron we can solve some linearly separable problems, but when we are about to do some problems that is not linearly separable, we need 2 or more layers of perceptron. As described at the beginning of the article, Figure 1 is a MLP, its every node connects to the every nodes in the next layer which is also called Full Connected and this kind of NN layer is called Dense Layer.

Convolutional Neural Network (CNN)