Deep Learning

Jawwad Shamsi

February 8, 2024

1 Fundamentals of Deep Neural Networks

An Artificial Neural Network (ANN) consists of many neurons, which are structured in layers. Each neuron takes one or many inputs, perform some computation, and gives an output. The term Multi-layer Perceptron (MLP) is interchangeably used with ANN.

A neuron performs the following functions:

1. Takes the input signals
2. Computes the weighted sum of the inputs to represent the total strength
3. Applies a step function
4. Checks whether to trigger an output if the threshold exceeds a limit.

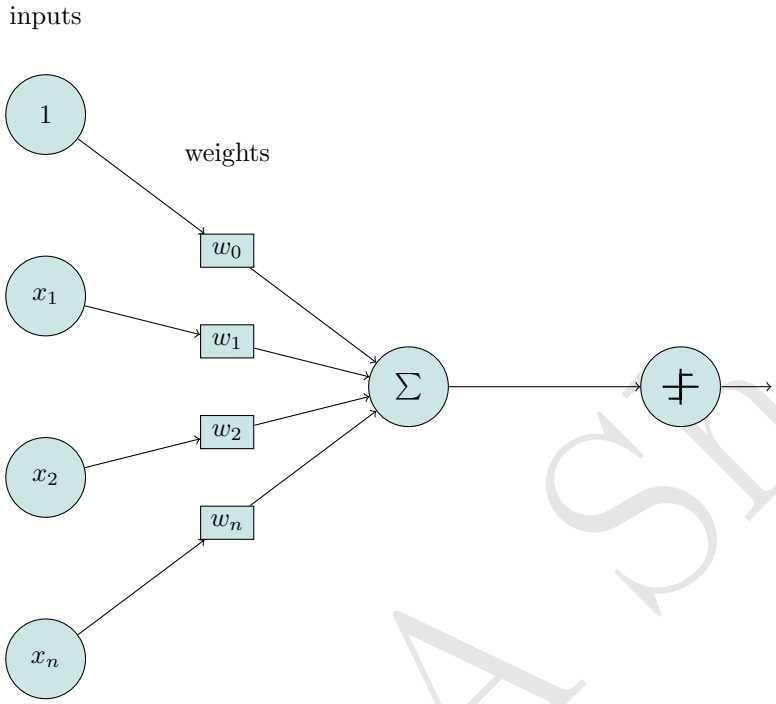
Weights are important. They provide significance of features. Not all features are equally important. During our training phase, the ANN will try to learn the correct value of the weights.

The weighted sum function is the sum of all inputs multiplied by their weights and then add a bias term.

$$z = \sum (X_{-i}.W_{-i}) + b \tag{1}$$

$$z = x_1.w_1 + x_2.w_2 + x_3.w_3 + - - - + x_n.w_n + b \tag{2}$$

where bias is the y-intercept. It allows us to better fit our model Recall: y=mx+b



What is an activation function: It is a function, which triggers the output of a neuron. It takes the input weighted sum and the bias and decides to give an output provided the input computation exceeds a certain threshold.

How does a percetpron learn: A perceptron estimates a value. However, the value may not be correct. It performs an iterative cycle in which it adjusts input weights and the value of bias by comparing the estimates with the predicted value Below is the outline procedure: predicted value.

Error Function: For regression, Mean Square error For categorical values, cross-entropy difference between parameters and hyper parameters

1. Neuron calculates the weighted sum and applies the activation function to make a prediction

$$y_e = activation(\sum(X_{-i}.W_{-i}) + b) \tag{3}$$

2.

$$error = y_e - Yp \tag{4}$$

y_e=estimated value of y; y_p=predicted value of y

3. Weights are updated. If the prediction is too high, weights are adjusted to make a lower prediction and vice versa

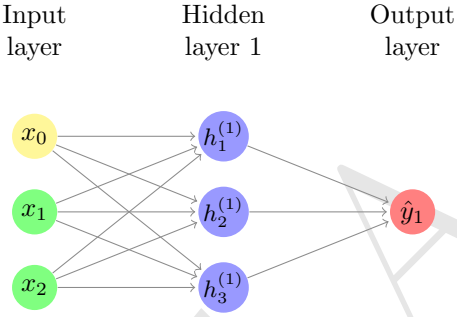
4. Repeat; converge until the error is minimal or within the bounded limits

Why more than one neuron? Perceptron is a linear function [1].

DO we have linear data? Linearly separable data can be separated by a straight line. What if our data is not linearly separable. We will add more lines and more neurons. Each neuron is generating a separate line and now we are moving towards non linearity.

Linear vs non-linear: Linear data is separable by a straight line Non-Linear is not separable by a straight line. Need more than one line. Non-linearity can be introduced through activation function as well as by introducing multiple neurons at a layer.

The example below shows a neural network with three neurons at the hidden layer each of which can generate an independent output. If the output of a single neuron (without activation function) is a straight line then the output of three neurons could be three independent lines. This could be aggregated as a non-linear behavior.



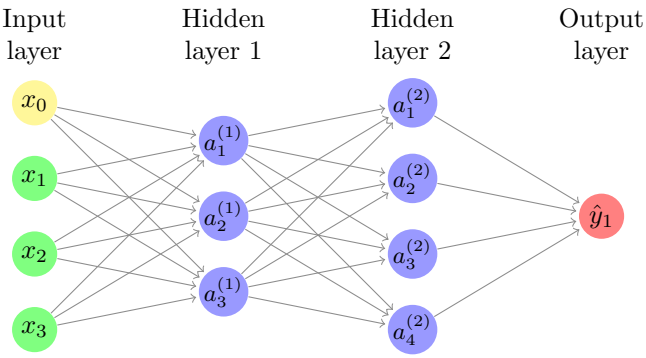
MLP consists of of following:

1. Input vector (or features)
2. Hidden Layers
3. Weight Connections
4. Output Layer

A fully connected Neural Network

Question: Will it help if a non-linear function has added layers?

[1] A linear function is an algebraic equation in which each term is either a constant or the product of a constant and (the first power of) a single variable. A linear function could have multiple variables as well



Why do we need more than one hidden layer? Because at each layer, we can extract different information.

1.1 Activation Functions

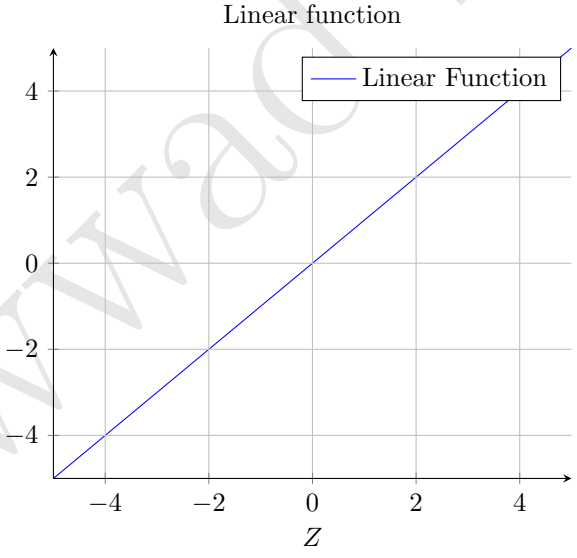
Now we will study different activation functions. Remember that the purpose of the activation function is to trigger an output when the weighted sum of inputs (and the bias) exceed certain threshold. This helps in achieving non-linearity.

1.1.1 Linear

Linear functions are not useful for DNN

example of linear function include

$f(x) = x$



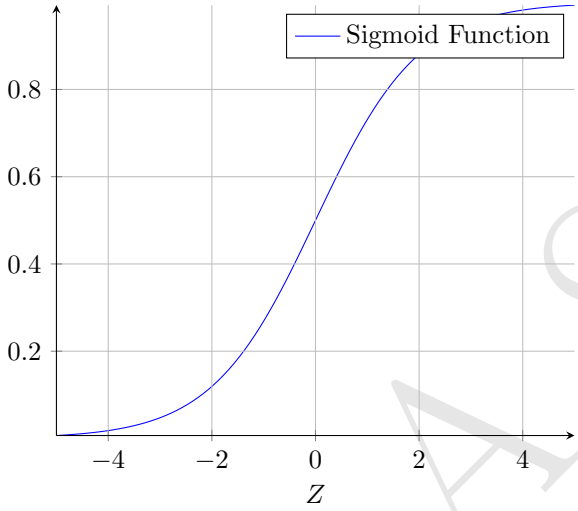
1.1.2 Sigmoid

It is used for binary classification.

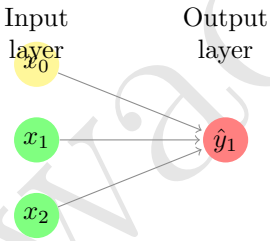
It outputs probabilities. It is a differentiable function, i.e., slope can be found between any two points.

$f(x) = 1/(1 + exp(-x))$

sigmoid function



The sigmoid activation function outputs probabilities for the occurrence of an event. The threshold is set as 0.5. If the output probability is ¡0.5 then the event is not detected, whereas an output probability of ¿ 0.5 implies that the event has been detected.



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Table 1: Truth Table of OR gate

Example 1.1 ***Sigmoid Function***

Consider an OR gate of two inputs x_1 and x_2. The truth table of OR gate is shown above. The equation for the above figure is shown below:

$$y = x_1.w_1 + x_2.w_2 + b \tag{5}$$

Devise weights, i.e. w_1, w_2 *and* b.

Solution *For* $x_1, x_2 = 0$ *the impact of* w_1, w_2 *is negated and* b *should be negative because the input to the sigmoid function should be -ve. This will yield output to be less than 0.5.*

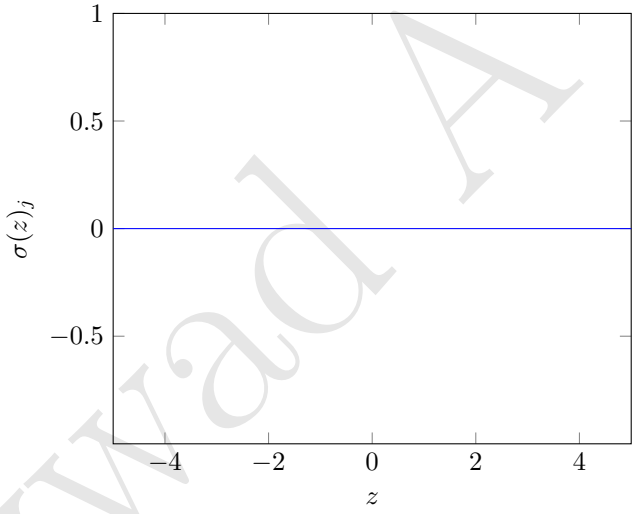
For all other values of x_1, x_2 *the combined effect of* $w_1x_1 + w_2x_2$ *should be greater than* b. *Remember* b *is negative. Some suggested values which will satisfy both of the above conditions are follows:*

$w_1, w_2 = 1, b = -0.5$ *Or* $w_2, w_2 = 2, b = -1$.

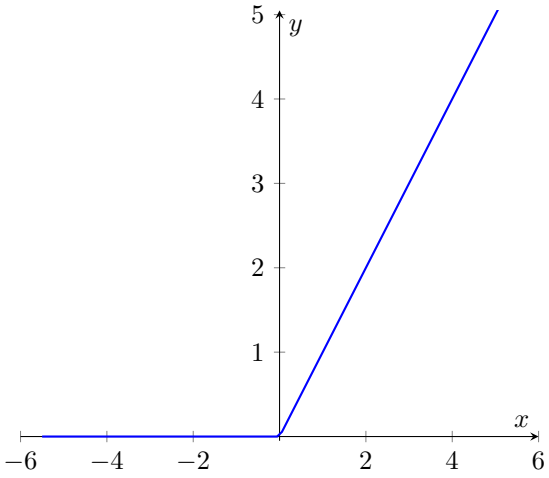
1.1.3 Softmax

The softmax activation function is used for multiple classes. It outputs probabilities whose sum is always 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities.

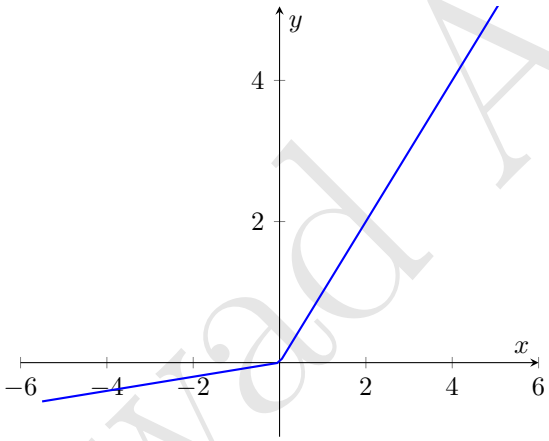
$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$



1.1.4 RELU

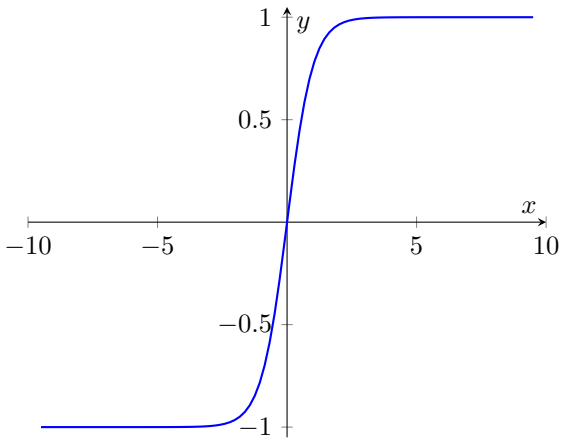


1.1.5 leaky Relu



1.1.6 tanh

$f(x) = (exp(x) - exp(-x))/(exp(x) + exp(-x));$

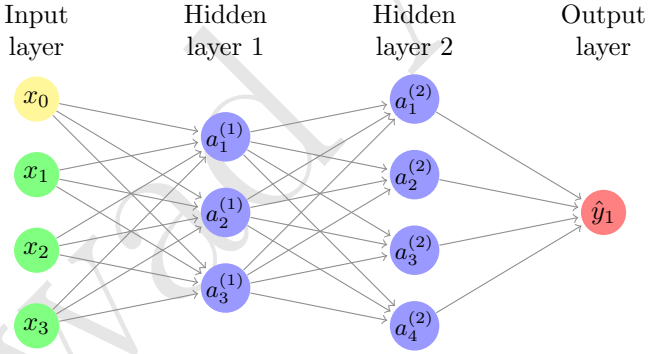


Question Is it possible to have multiple minimums in a output of a DNN?

Yes. because a DNN could have multiple layers and at each layer we have different combinations of weights. So different combinations of weights could lead to the same output.

2 forward Propagation

As we proceed in a DNN, we have different combinations of weights



$a_1^1 = \sigma(w_{11}^1.x_1 + w_{12}^1.x_2 + w_{13}^1.x_3)$

$a_2^1 = \sigma(w_{21}^1.x_1 + w_{22}^1.x_2 + w_{23}^1.x_3)$

$a_3^1 = \sigma(w_{31}^1.x_1 + w_{32}^1.x_2 + w_{33}^1.x_3)$

where w_{jk}^i denotes weight b/w the j$^{\text{th}}$ node at the i$^{\text{th}}$ layer and k$^{\text{th}}$ node at the i-i$^{\text{th}}$ layer

$\hat{y} = \sigma(w_{11}^3 a_1^2 + w_{12}^3 a_2^2 + w_{13}^3 a_3^2 + w_{14}^3 a_4^2)$

We can also solve it using matrix.

$$= \sigma \begin{bmatrix} W_{11}^3 & W_{12}^3 & W_{13}^3 & W_{14}^3 \end{bmatrix} .\sigma \begin{bmatrix} W_{11}^2 & W_{12}^2 & W_{13}^2 \\ W_{21}^2 & W_{22}^2 & W_{23}^2 \\ W_{31}^2 & W_{32}^2 & W_{33}^2 \\ W_{41}^2 & W_{42}^2 & W_{43}^2 \end{bmatrix} .\sigma \begin{bmatrix} W_{11}^1 & W_{12}^1 & W_{13}^1 \\ W_{21}^1 & W_{22}^1 & W_{23}^1 \\ W_{31}^1 & W_{32}^1 & W_{33}^1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

3 Cost Function

Cost function $J(w)$ is a function of weights. It is a non-convex function means that there could be more than one minima

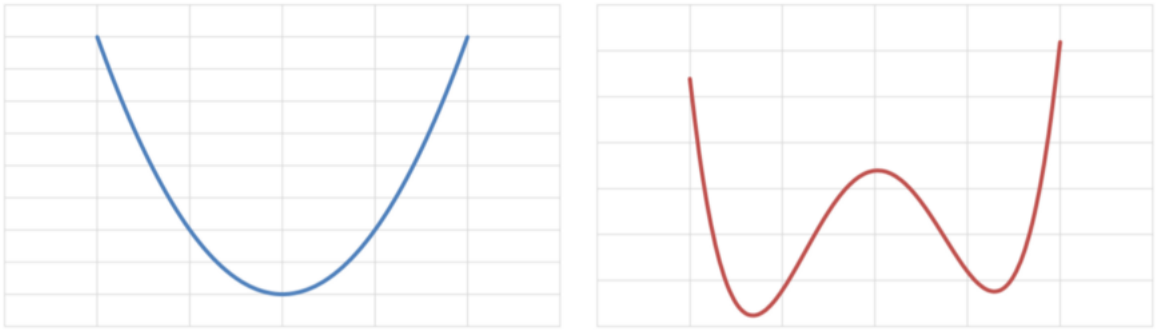


Figure 1: Convex vs Concave

Figure 1 illustrates difference b/w a convex and a non-convex function. A non-convex function may have multiple global minima. Cost function in DNN is a non-convex function b/c of possibilities of multiple layers and different combination of weights Cost function is calculated as follows:

$J(w) = |\hat{y} - y|$ i.e., absolute difference between the estimated y and the calculated y.

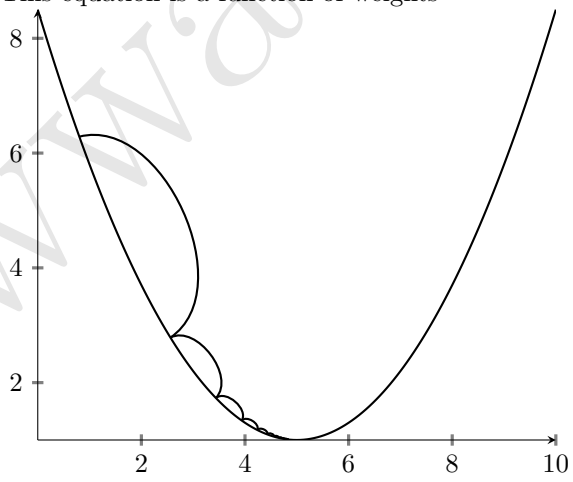
$\hat{y} = w.x$

assuming x=0.4 and y=.7

$\hat{y} = w.0.4$

$J(w) = |w.0.4 - .7|$

This equation is a function of weights



slope is important b/c it shows us the direction of adjustment of weights whereas η denotes the step size