Deep Learning: DL is a technique which basically mimics the human brain.

So, in 1950’s and 60’s our researchers and scientists thought that can we make a machine learn and work like how we human actually learn and you know that we’ll be learning basically from the environment with the help of our brain. Our brain has such capacity that can learn the things very quickly. That is where DL concept came into picture which lead to creation of Neural Networks.

Simplest type of neural network is call perceptron. However there were some drawbacks in the perceptron because the perceptron or the neural network was not able to learn properly because of the concepts that were applied.

So, in 1980’s Jeffrey Hinton invented a concept called Back Propogation. Because of this back propogation ANN, CNN and RNN became so efficient that many companies are now using it. Many people have developed a lot of applications which are efficiently working because of Jeffrey Hinton and his concepts.

Neural Network: When I was a kid and I first saw a dog/cat, at that time those inputs I was not able to distinguish and not body would be able to distinguish, nobody can correctly distinguish it is directly seeing an object for the first time because you will not know whether that is a dog/cat right but some of the information will basically get from someone like suppose if I take that my family members explain me that ok what is the basic difference between dogs and cats by providing the features such as cat has pointy ears,different types of eyes colors and usually small. So this three information and apart from seeing the color of the cat and all those kind of information I basically got interested and remembered in my brain wrt those features. Now I am able distinguish what is a cat and a dog.

So this way only we can also make a NN architecture run by providing those features, ingesting those features for each and every neuron present in the NN and those will learn using that information and give you the output with the help of back propagation to train itself and learn new things.

As I am ingesting the features through my sensory organs I.e., eyes which is my input layer. So first layer of the neural network is the input layer. NN input layer will have nodes which indicates as features present in int.

AFter that information is passed it reaches the hidden layer, which can have multiple number of neuron. All this information from my eyes is passing through all the neurons and there is some pre-processing happing in the neurons as it passes through layers. Each layer can have n-number of layers and for simplifying we will take only 1 layer with 4/5 neurons and can have any number of neurons in the layers.

As the information/features passes through my eyes it goes through all the neurons similarly in NN each and every feature will also pass through all the neurons and some kind of pre-processing happens over here and finally we get the output. We can have any number of outputs. For example if it is a multi-classification problem then we will have that many number of output nodes and if it is a binary classification we can have one o/p node.

Basic Neural Network Architecture:

I/P layer(Nodes) Hidden Layers (NEurons) Output Layer (Nodes)

Feature 1 -->

Feature 2 -->

Feature 3 -->

Working of Neural Networks:

Step 1: Y = W1X1 + W2X2 + W3X3

Z = Act (Y)

I/P layer Neurons O/P Layer

X1 ---------------> W1 W4

X2 ---------------> W2 Z = Z X w4 = Act(Z)

X3 ---------------> W3

Step1:

Sigma(WiXi) for i=1to N

Step 2:

Act(Sigma(WiXi) for i=1to N)

Neuron

As soon as the feature is basically taken from my eyes it passes through the neurons and each neuron does some kind of processing. But before passing through the neurons, the lines passing through the neurons will have some weights (W1,W2,W3) assigned. Information and weights will be passed to my hidden neuron and then 2 types of operations will happen inside the neuron as shown in above Neuron figure.

Step 1: Y = W1X1 + W2X2 + W3X3+Bias

Step2: Z = Act (Y)

Once the product of weight and feature is done and all of them are summarized then apply activation function. However, we will add bias parameter (smaller value) before the activation function.

Lets take an example: if I have kept a hot object in my hand, as soon as it placed on my hand that signal/information will get passed through the neurons, in short these neurons are getting activated. Similarly, Neurons in NN will get activated with the help of higher weights and using a particular activation function.

Take an example of Sigmaoid activation function, this activation function will transform its output from 0 to 1. if the value is less than 0.5 it becomes 0 that means neurons will not get activated, if the value is greater then 0.5 which means 1 where neurons will get activated.

**Sigmoid = 1/(1+e-y)**

I have kept a hot object in my hand, as soon as it placed on my hand that signal/information will get passed through the neurons, then what about the other neurons, these will not get activated however activated neurons will pass the information to the brain. This entire process is called forward propagation. As the output of 1st layer goes to the 2nd layer and hence forth.

Activation Functions: In the neural network architecture we observed that each and every feature will be multiplied to weights and below is the formula:

Step 1: Y = W1X1 + W2X2 + W3X3+Bias(I)

Step2: Z = Act (Y)

We will discuss about 2 activation functions as of now:

1. Sigmaoid Activation Function
2. RELU Activation Function

**Sigmaoid Activation Function:** This will actually transform your y between 0 and 1. This is a sinsodial curver equation. This function is used in Logistic regression also.

**Sigmoid = 1/(1+e-y)**

Y = W1X1 + W2X2 + W3X3+Bias(I)

Once Sigmoid activation function initiated then it will transform y between 0 and 1.

Let y be any value negative/positive/too long/too big values and when we place the y value in sigmaoid function then the value will always be in between 0 and 1.

If the values comes between 0.01 and 0.99 then as per the threshold limit we will convert the values into either 0 or 1. 1 represents neuron is activated and 0 represents neuron is not activated. If the neuron is activated then it means that it is actually transferring the signal which means it is actually helping you to classify the final output.

**Why activation function is required:** I have kept a hot object in my hand, as soon as it placed on my hand that signal/information will get passed through the neurons, then what about the other neurons, these will not get activated however activated neurons will pass the information to the brain and then I will react to hot object

Similarly, this activation function and weights are very very important. If I take an example of other neurons and nothing is happening over here which means no signal is transferred basically and at that time we can say that particular neuron is deactivated.

**RELU Activation Function:**After calculating the Y value and passed into the RELU activation function then a simple formula is basically getting applied. **Max(y,0)** which basically indicates that “Y” is basically the output of weights \* Features +Bias.

Y = W1X1 + W2X2 + W3X3+Bias(I)

RELU AF = Max(y,0)

Suppose if Y is negative then Max(Y,0) ==> Max(-ve,0)==>0. because of max of 0 and negative value is always 0.

Suppose if Y is Positive then Max(Y,0) ==> Max(+ve,0)==>+. because of max of 0 and +ve value is always +ve.

This is the functionality of RELU activation function and much more popular than sigmoid.

**When to use RELU and Sigmoid:** When working on Regression function then RELU AF is used and for Classification problem for intermediate layers we can use RELU AF however in the final layer we need to use Sigmoid AF.

Other Activation Functions: Threshold and Leaku RELU Activation functions

**Neural Network with Back Propagation:**

Dataset: below dataset shows whether the student will pass the exam or not and are indicated by 0 and 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Play | Study | Sleep | olp |
| 2hrs | 4hrs | 8hrs | 1 |

By using the above dataset lets see how the neural network gets trained.

X1= Play

X2 = study

X3 = Sleep

**Forward Propagation:**

I/P layer hidden Neurons O/P Layer (predicted values)

X1 ---------------> W1 W4

X2 ---------------> W2 Z = Z X w4 = Act(Z)

X3 ---------------> W3

Hidden Neuron functionality: AF is sigmoid AF to classify the output

Step1:

Sigma(WiXi) for i=1to N +Bias

Step 2:

Z =Act(y)

Suppose my O/p predicted value as 0 whereas the actual value is 1, we need to compare whether predicted and actual values are same.

So, we try to see the differences and that’s what we call as a **Loss Function.**

**Loss Function = (Actual - Predicted)^2** ==> Square applied to convert negative to positive value

From above example: Loss function = (1-0)^2 ==>1

Now we can see the loss value is higher and by the difference also it is completely wrongly predicted. So, we should try to adjust the weights. These weights adjustments must be done in such a way that for this particular record my predicted output should be 1 and for that we define a loss function.

After defining a loss function, this loss value should be reduced to a minimal value such that predicted should match with Actual value. In this scenario we can see that my loss function is having value as “1”. So during the training phase I should be updating this weights in such a way that the values should be decreasing. In order to achieve this we need to use Optimizers.

There are many optimizers used in DL. Some of the optimizers are Gradient Descent, Stocastic Gradient Descent and many more.

When we apply the optimizers it will try to decrease my loss value. In order to achieve this we need to back propagate now. When do the back propagation which means we need to update the weights.

To do the back propagation what we need to do is that, we need to create new weights which is the subtraction of previous weights - learning rate(n) X derivation of loss function wrt W4.

**W4\_new = W4 - nX(dL/dW4)**

**Note:** Learning rate should be very small value because this will go to the global minima function in the gradient descent whenever we want to actually achieve that particular point. This can be achieved only by using **optimizer function.** Loss function should be small but it should not be very small and it not be very high. If it is very high then we will never come towards the global minima point.

Once W4\_new gets calculated then we need to update the prior weights using the above formula by updating the derivation of loss function wrt the weights.

Once the new weights are calculated then the forward propagation will happen and this will be repeated (epochs) until and unless this loss function value is completely reduced. Once this is reduced then actual and predicted values will be matching finally.

This is all about training the neural network. There are lot of problems that basically happens such as **vanishing gradient, exploding gradient**

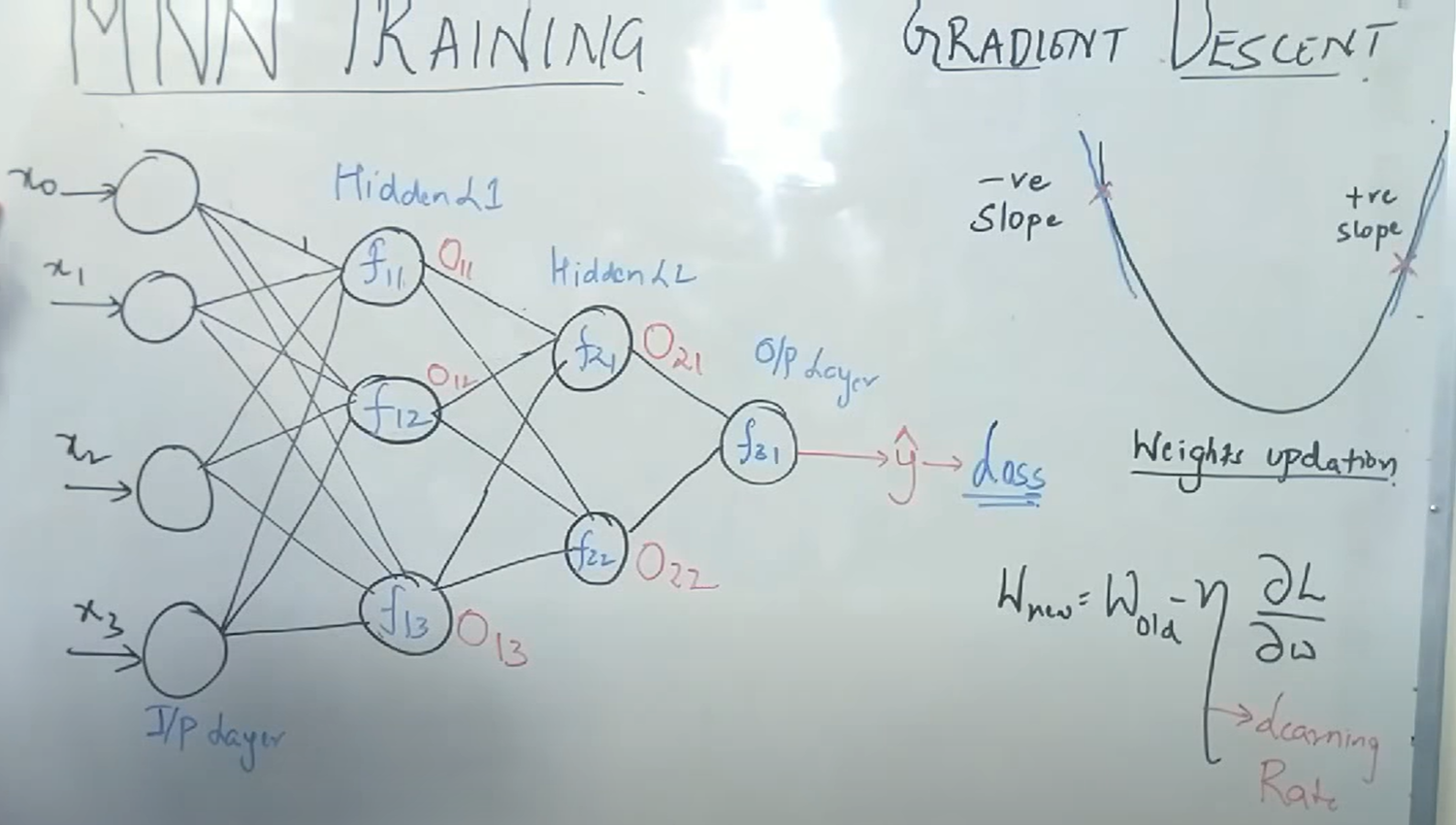
I have taken a single record and hence I created a loss function. If I have bunch of records then I need to create a loss function which will be the sum of all loss values for all records

**Cost function = sigma(I=1 to n) (actual - predicted)^2**

As soon as we get the loss function value, the main aim will be to reduce the loss function value with the help of optimizer. Optimizer will help us to find the derivative and will try to change the weights value wrt older value using the above formula.

**Multi Layer Neural Network (MNN):** Below is the multi layer neural network with 2 hidden layers and 1 output layer.

1st layer is Input layer with 4 features and all these will be passed to hidden layer. Weights will be assigned as they are passing through the hidden layers.



If we have so many inputs and so many neurons, initially we need to understand what kind of matrix will be formed wrt the weights and our next hidden neuron layer. The matrix between input and hidden layer L1 will be **“4X3”** matrix because each input passes through all the neurons of the hidden layer and we have 3 neurons in hidden layer hence it will be “4X3” matrix for the input and L1 layer.

The matrix for L1 and L2 layers will be **“3X2”** matrix as L1 is having 3 neurons and L2 is having 2 neurons which results in “3X2” matrix. For the final output layer, which is L2 and O/p layer will be **“2X1”** as L2 is having 2 neurons and O/p is having 1 neuron which will be “2X1” matrix.

As we know that, Information and weights will be passed to my hidden neuron and then 2 types of operations will happen inside the neuron as shown in above Neuron figure.

Step 1: Y = W1X1 + W2X2 + W3X3+Bias

Step2: Z = Act (Y)

Once the product of weight and feature is done and all of them are summarized then apply activation function. However, we will add bias parameter (smaller value) before the activation function.