Multi-Layer Perceptron

A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non – linear functions.

Figure 1 shows a multi layer perceptron with a single hidden layer. Note that all connections have weights associated with them, but only three weights (w_0, w_1, w_2) are shown in the figure.

Input Layer: The Input layer has three nodes. The Bias node has a value of 1. The other two nodes take X1 and X2 as external inputs (which are numerical values depending upon the input dataset). Here, no computation is performed in the Input layer, so the outputs from nodes in the Input layer are 1, X1 and X2 respectively, which are fed into the Hidden Layer.

Hidden Layer: The Hidden layer also has three nodes with the Bias node having an output of 1. The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges). Figure 1 shows the output calculation for one of the hidden nodes (highlighted). Similarly, the output from other hidden node can be calculated. Remember that f refers to the activation function. These outputs are then fed to the nodes in the Output layer.

Output Layer: The Output layer has two nodes which take inputs from the Hidden layer and perform similar computations as shown for the highlighted hidden node. The values calculated (Y1 and Y2) as a result of these computations act as outputs of the Multi Layer Perceptron.

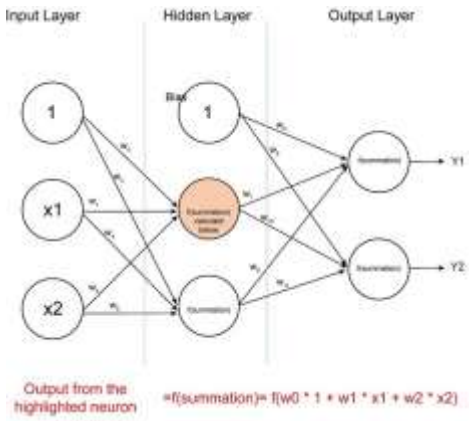

Figure 1: a multi layer perceptron having one hidden layer

Example:

Given a set of features $X = (x1, x2, \ldots)$ and a target y, a Multi Layer Perceptron can learn the relationship between the features and the target, for either classification or regression.

Let's take an example to understand Multi Layer Perceptrons better. Suppose we have the following student-marks dataset:

Hours Studied	Mid term Marks	Final Term Result
35	67	1(Pass)
12	4 75	O(Fail)
16	89	1(Pass)
45	56	1(Pass)
10	90	O(Fail)

The two input columns show the number of hours the student has studied and the mid term marks obtained by the student. The Final Result column can have two values 1 or 0 indicating whether the student passed in the final term. For example, we can see that if the student studied 35 hours and had obtained 67 marks in the mid term, he / she ended up passing the final term.

Now, suppose, we want to predict whether a student studying 25 hours and having 70 marks in the mid term will pass the final term.

Hours Studied	Mid term Marks	Final Term Result
25	70	?

This is a binary classification problem where a multi layer perceptron can learn from the given examples (training data) and make an informed prediction given a new data point. We will see below how a multi layer perceptron learns such relationships.

Training our MLP: The Back-Propagation Algorithm

Back-Propagation Algorithm:

The process by which a Multi Layer Perceptron learns is called the Backpropagation algorithm. Initially all the edge weights are randomly assigned. For every input in the training dataset, the ANN(Artificial Neural Network) is activated and its output is observed. This output is compared with the desired output that we already know, and the error is "propagated" back to the previous layer. This error is noted and the weights are "adjusted" accordingly. This process is repeated until the output error is below a predetermined threshold.

Once the above algorithm terminates, we have a "learned" ANN which, we consider is ready to work with "new" inputs. This ANN is said to have learned from several examples (labeled data) and from its mistakes (error propagation).

Now that we have an idea of how Backpropagation works, lets come back to our student-marks dataset shown above.

The Multi Layer Perceptron shown in Figure 2 has two nodes in the input layer (apart from the Bias node) which take the inputs 'Hours Studied' and 'Mid Term Marks'. It also has a hidden layer with two nodes (apart from the Bias node). The output layer has two nodes as well – the upper node outputs the probability of 'Pass' while the lower node outputs the probability of 'Fail'.

In classification tasks, we generally use a Softmax function as the Activation Function in the output layer of the Multi Layer Perceptron to ensure that the outputs are probabilities and they add up to 1. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one. So, in this case,

$$\text{Probability (Pass)} + \text{Probability (Fail)} = 1$$

Step 1: Forward Propagation

All weights in the network are randomly assigned. Let's consider the hidden layer node marked V in Figure 2 below. Assume the weights of the connections from the inputs to that node are $w_0, w_1, w_2, \ldots, w_{11}$ (as shown).

The network then takes the first training example as input (we know that for inputs 35 and 67, the probability of Pass is 1).

- Input to the network = [35, 67]
- Desired output from the network (target) = [1, 0]

Then output V from the node in consideration can be calculated as below (f is an activation function such as sigmoid):

$$V = f(1 * w_1 + 35 * w_2 + 67 * w_3)$$

Similarly, outputs from the other node in the hidden layer is also calculated. The outputs of the two nodes in the hidden layer act as inputs to the two nodes in the output layer. This enables us to calculate output probabilities from the two nodes in output layer.

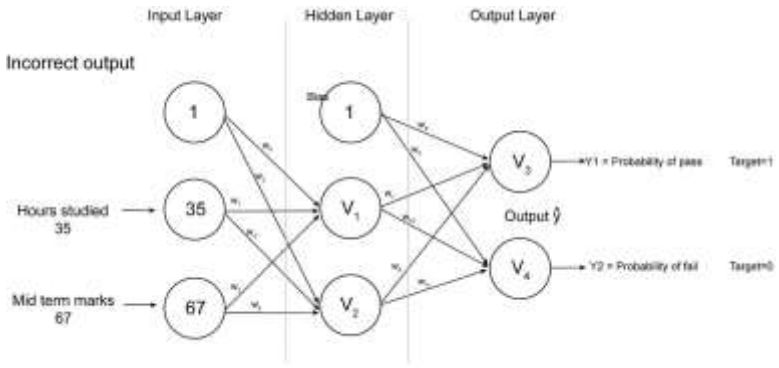

Figure 2: a multi layer perceptron having one hidden layer

Computations for the edges between input(first) layer and hidden(second) layer:

- Now, as we know the input to the network is [1, 35, 67].
- Let's start with random weights for the first layer as $[w_0, w_1, w_2]$ = [-0.1, -0.2 -0.2].
- Since, $V_1 = f(1 * w_1 + 35 * w_2 + 67 * w_3)$ where f is an activation function such as sigmoid.We have the following matrix multiplication as

$$\begin{bmatrix} 1 & 35 & 67 \end{bmatrix} \cdot \begin{bmatrix} -0.1 \\ -0.2 \\ -0.2 \end{bmatrix} = 1 * (-0.1) + 35 * (-0.2) + 67 * (-0.2) = -20.5$$

After applying activation function f we get, $f(-20.5) = 1.25$. Therefore, $V_1 = 1.25$

- Now, let's start with random weights for the remaining weights in first layer as $[w_3, w_4, w_5]$ = [-0.4, -0.1 -0.2].
- Since, $V_2 = f(1 * w_3 + 35 * w_4 + 67 * w_5)$ where f is an activation function such as sigmoid.We have the following matrix multiplication as

$$\begin{bmatrix} 1 & 35 & 67 \end{bmatrix} \cdot \begin{bmatrix} -0.4 \\ -0.1 \\ -0.2 \end{bmatrix} = 1 * (-0.4) + 35 * (-0.1) + 67 * (-0.2) = -17.3$$

After applying activation function f we get, $f(-17.3) = 3.0669$. Therefore, $V_2 = 3.0669$

Computations for the edges between hidden(second) layer and output(third) layer:

- Now, let's start with random weights for the third layer as $[w_6, w_7, w_8]$ = [0.1, -0.2, 0.1].
- Since, $V_3 = f(1 * w_6 + 1.25 * w_7 + 3.0669 * w_8)$ where f is an activation function such as sigmoid. We have the following matrix multiplication as

$$\begin{bmatrix} 1 & 1.25 & 3.0669 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ -0.2 \\ 0.1 \end{bmatrix} = 1 * (0.1) + 1.25 * (-0.2) + 3.0669 * (0.1) = 0.1567$$

After applying activation function f we get, $f(0.1567) = 0.54$. Therefore, $V_3 = 0.54$.

- Now, let's start with random weights for the remaining weights in the third layer as $[w_9, w_{10}, w_{11}]$ = [0.1, 0.2, 0.1].
- Since, $V_4 = f(1 * w_7 + 1.25 * w_8 + 3.0669 * w_9)$ where f is an activation function such as sigmoid. We have the following matrix multiplication as

$$\begin{bmatrix} 1 & 1.25 & 3.0669 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.2 \\ 0.1 \end{bmatrix} = 1 * (0.1) + 1.25 * (0.2) + 3.0669 * (0.1) = 2.906$$

After applying activation function f we get, $f(2.906) = 0.95$. Therefore, $V_4 = 0.95$.

Here,the output probabilities from the two nodes in the output layer are 0.54 and 0.95 respectively (since the weights are randomly assigned, outputs will also be random). We can see that the calculated probabilities (0.54 and 0.95) are very far from the desired probabilities (1 and 0 respectively), hence the network in Figure 2 is said to have an 'Incorrect Output'.

We have Probability of Pass = 0.54 and respective Target = 1. Therefore, Error =(1-0.54)=0.46 and for Probability of fail =0.95 and respective Target = 0. Therefore, Error =(0-0.95)=0.95.

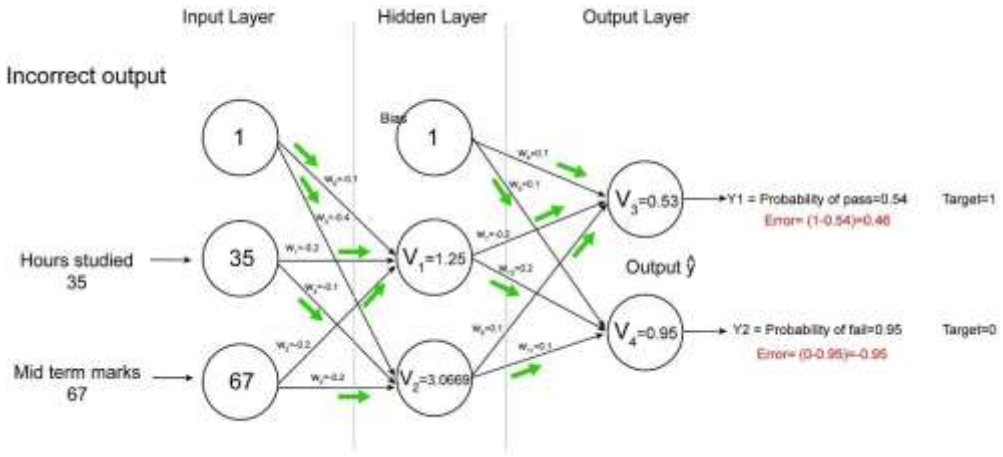

Figure 3: forward propagation step in a multi layer perceptron

Step 2: Back Propagation and Weight Updation

We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation to calculate the gradients. Then we use an optimization method such as Gradient Descent to 'adjust' all weights in the network with an aim of reducing the error at the output layer. This is shown in the Figure 3 in next page (ignore the mathematical equations in the figure for now).

Suppose that the new weights associated with the node in consideration are $w_0^{'}, w_1^{'}, w_2^{'}, ..., w_{11}^{'}$ (after Backpropagation and adjusting weights).

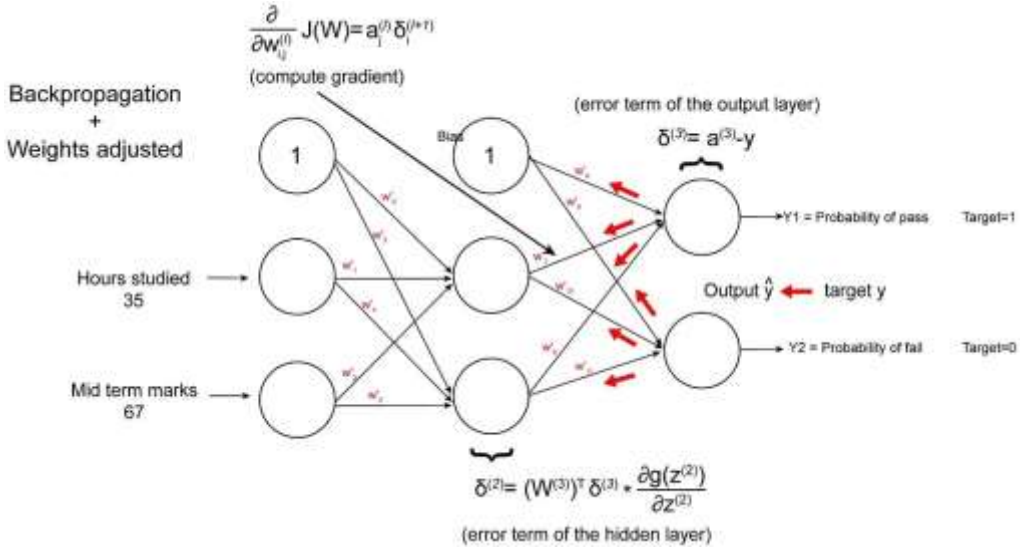

Figure 4: backward propagation and weight updation step in a multi layer perceptron

If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize the error in prediction. As shown in Figure 4, the errors at the output nodes now reduce to [0.31, -0.87] as compared to [0.46, -0.95] earlier. This means that our network has learnt to correctly classify our first training example.

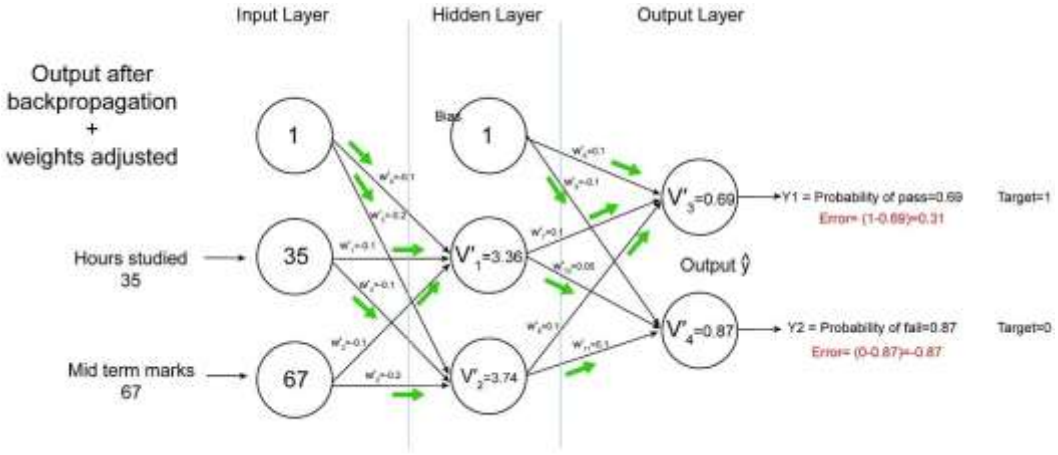

Figure 5: the MLP network now performs better on the same input

We repeat this process with all other training examples in our dataset. Then, our network is said to have learnt those examples.

If we now want to predict whether a student studying 25 hours and having 70 marks in the mid term will pass the final term, we go through the forward propagation step and find the output probabilities for Pass and Fail.

Note: A simple explaination of Maths behind Backpropagation can be found here.

References:

For more details on Multi-Layer Perceptron and Back Propagation can refer below:
http://www.cs.bham.ac.uk/~jxb/NN/l7.pdf
https://www.cse.unsw.edu.au/~cs9417ml/MLP2/BackPropagation.html