Questions

1 -What is a Multi-Layer Perceptron (MLP) and how does it differ from a single-layer perceptron?

2- Explain the architecture of a Multi-Layer Perceptron (MLP) and the role of input, hidden, and output layers.

3- How are weights initialized in a Multi-Layer Perceptron (MLP) and why is it important?

4- What is the purpose of activation functions in a Multi-Layer Perceptron (MLP)? Name some commonly used activation functions.

5- What is backpropagation and how is it used to train a Multi-Layer Perceptron (MLP)

6- How do you choose the number of hidden layers and neurons in each layer of a Multi-Layer Perceptron (MLP)?

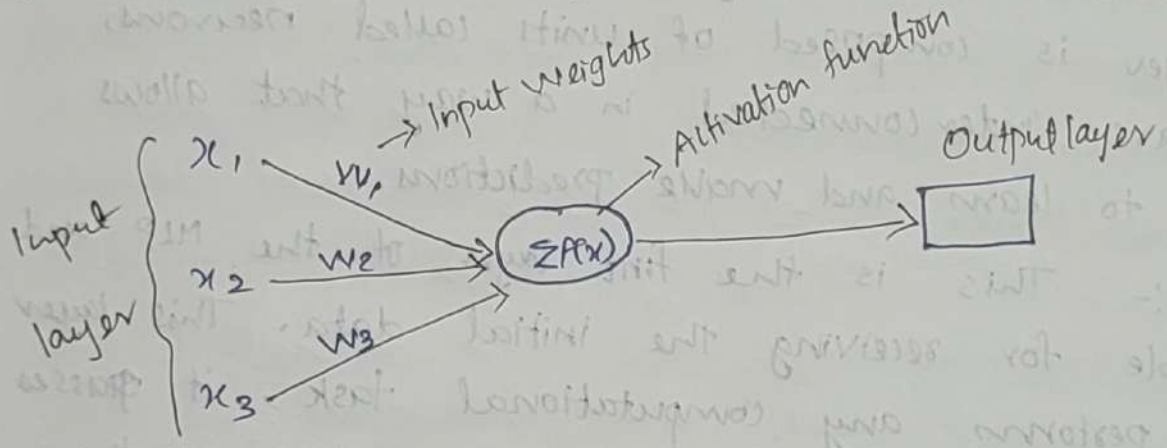
Q1. A multilayer perceptron [MLP] is a type of ANN that consists of multiple layers of neurons, each connected to the next layer. It is the advanced version of the single-layer perceptron, which has only one layer of neurons.

Single-layer Perceptron

→ Consists of a single layer of neurons directly connected to the input layer.

→ Does not have the concept of hidden layers.

→ Suitable for linearly seperable problems.



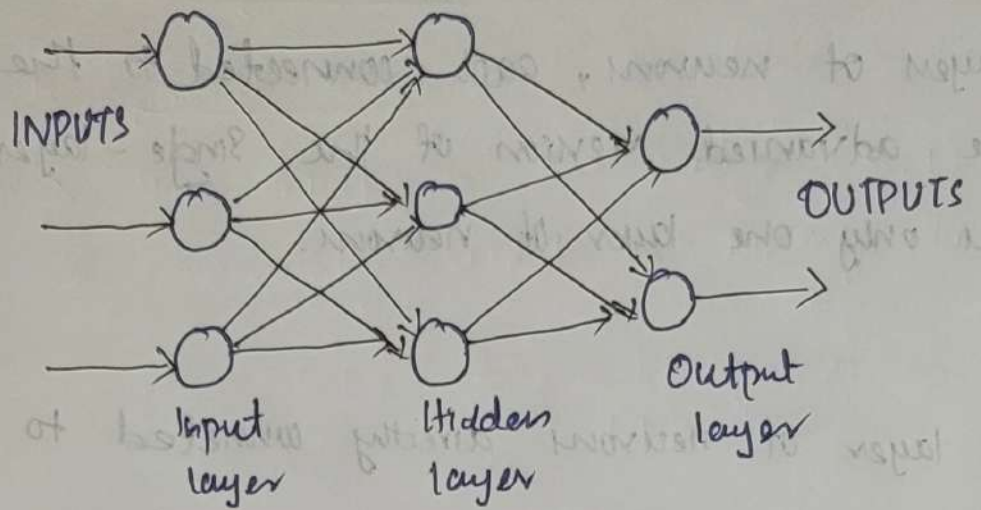
Multi-layer Perceptron

→ Consists of an input layer, one or more hidden layers, and an output layer.

→ Neurons in each layer are fully connected to the neurons in the next layers.

→ Hidden layers allows the network to learn and represent complex, non-linear relationships.

→ Training is more computationally intensive than single-layer perceptrons due to the complexity of the network and the need to calculate gradients for multiple layers.



Q2. The architecture of a Multi-layer perceptron consists of three main types of layers: the input layer, one or more hidden layers and the output layer. Each layer is composed of units called neurons, which are interconnected in a way that allows the MLP to learn and make predictions.

Input layer:- This is the first layer of the MLP and is responsible for receiving the initial data. This layer does not performs any computational task, it passes the input values to the next layer, which is the hidden layer.

Hidden layer:- It is located in between the input and output layer. This layer performs the main computation. This helps the network to learn complex patterns and representations.
The neurons in hidden layers apply a weighted sum to their inputs, add a bias value and then pass through some activation functions.

Output layer:- The output layer is the final layer of the MLP and it produces the final predictions / output values.

Q3. Weights initialization in a MLP is an important step, which can impact the training process and the performance of the neural networks. Proper weight initialization is important to get the desired output.

Importance of Weight Initialization :-

→ If all weights are initialized to the same value, the neurons in each layer will compute the same output and gradient during training, which leads to a situation where all neurons in a layer update identically.

→ Poor initialization can lead to problems like Vanishing gradient or exploding gradients. This can make the training process slow.

→ Properly initalized weights can lead to faster convergence during training. This reduces the computational resources and time required to train the model.

→ Common weight Initialization Techniques are:-

* Normal Initialization
* Uniform Initialization
* Xavier [Glorot] Initialization
* He Initialization.

Q4. The main purpose of activation functions in a MLP is to introduce the non-linearity into the networks. This helps the network to learn and represent complex patterns. Activation functions can map the output of neurons to a specific range such as [0,1] or [-1,1], which is very useful in tasks such as classification.

Commonly Used Activation Functions

1. Sigmoid function :-

→ Maps inputs to the range (0, 1).

→ Commonly used for binary classification problems.

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

2. Tanh function :-

→ Maps input to the range (-1, 1)

→ It is zero-centered, which make training more efficient compared to sigmoid function.

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

3. Rectified Linear Unit [ReLU] :-

→ It introduce non-linearity.

→ It does not suffer from vanishing gradients for positive inputs.

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

4. Softmax :-

→ Used in the output layer for multi-class classification problems.

$$\text{Softmax}(a) = \frac{e^{z}}{\Sigma e^{z}}$$

Q5. Backpropagation is an algorithm used for training the neural network. It consist of two phases :- Forward propagation and Backward Propagation. The main aim of backpropagation is to minimize the error by adjusting the weights and biases of a neural network using gradient descent.

Forward propagation :-

→ The input data is passed through the network layer by layer to compute the output.

→ Each neuron in the hidden layer and output layer computes a weighted sum of inputs, adds a bias value and passes through an activation functions to produce the output.

→ The output of one layer becomes the inputs to the next layer.

Loss Calculation :-

→ The output from the final layer is compared to the actual target values to calculate the loss [error.].

Backward propagation :-

→ Here it propagates the error backward through the network to update the weights and biases.

→ Calculates the gradient of loss function with respect to each weights and bias using the chain rule of calculus.

→ Adjust the weights and biases in the direction that reduces the loss value.

Q6. Choosing the number of hidden layers and the number of neurons in each layer of a MLP is important as it impact the performance of the network.

Choosing the number of hidden layers :-

→ Begin with small number of hidden layers and gradually increase the complexity only if necessary.

→ It is based on the complexity of problems that we are solving. Generally simple tasks like basic classification or regression requires only one hidden layer.

→ Larger ~~dataset~~ and complex datasets with many features require more hidden layers to find the patterns.

Choosing the number of neurons in each layer :-

→ A common way is to start with a number of neurons between the size of the input layer and the size of the output layer.

→ We can use the cross-validation method to choose the one that provides best performance.

→ Avoid overfitting as too many neurons can lead to overfitting issues.

→ We also need to consider the computational resource and time constraints.