**Q1. Explain the concept of forward propagation in a neural network.**

Forward propagation in a neural network refers to the process of passing input data through the network layers to generate an output. Each layer performs calculations using weights, biases, and activation functions to transform the input data. The final output is used to make predictions or classifications, based on the learned parameters of the network.

**Q2. What is the purpose of the activation function in forward propagation?**

The activation function in forward propagation introduces non-linearity into the neural network, allowing it to learn complex patterns. Without it, the network would only be able to model linear relationships, limiting its capability. It helps determine whether a neuron should be activated, influencing the network's ability to make accurate predictions.

**Q3. Describe the steps involved in the backward propagation (backpropagation) algorithm?**

**The steps in the backpropagation algorithm are:**

* Compute the Loss: Calculate the difference between the predicted output and the true output using a loss function.
* Compute Gradients: Use the chain rule to compute gradients of the loss for each weight in the network.
* Update Weights: Adjust the weights by subtracting the gradient multiplied by the learning rate, in order to minimize the loss.
* Repeat: This process is repeated for multiple iterations (epochs) to gradually improve the model's performance.

**Q4. What is the purpose of the chain rule in backpropagation?**

The chain rule in backpropagation is used to compute the gradients of the loss function with respect to each weight in the network. It allows the error to be propagated backward through the layers, layer by layer, by breaking down complex derivatives into simpler ones. This helps in efficiently updating the weights to minimize the loss.

**Q5. Implement the forward propagation process for a simple neural network with one hidden layer using NumPy.**

import numpy as np

X = np.array([[0.5, 0.6]])

W1 = np. random.randn(2, 3)

b1 = np. random.randn(1, 3)

W2 = np. random.randn(3, 1)

b2 = np. random.randn(1, 1)

# Forward propagation

Z1 = np.dot(X, W1) + b1  # Hidden layer input

A1 = np.tanh(Z1)  # Activation function (tanh)

Z2 = np.dot(A1, W2) + b2  # Output layer input

output = 1 / (1 + np.exp(-Z2))  # Sigmoid activation for output

print(output)

**Output:** [[0.28877522]]