Q1. What is the purpose of forward propagation in a neural network? Forward propagation is the process through which inputs are passed through the layers of a neural network to compute the output or predictions. It involves:

Calculating the weighted sum of inputs and biases for each neuron. Applying activation functions to introduce non-linearity. The primary purpose is to make predictions or compute the output of the network for a given input. Q2. How is forward propagation implemented mathematically in a single-layer feedforward neural network? For a single-layer feedforward neural network, the output y y is calculated as follows:

Compute the linear combination of inputs and weights:

z

$\sum i$

$1 n w i x i + b$ or z

$W T X + b$ z= i=1 ∑ nw ix i+borz=W T X+b Where:

W W is the weight vector. X X is the input vector. b b is the bias term. Apply an activation function $f (z)$ f(z) to compute the output:

y

$f (z)$ y=f(z) Q3. How are activation functions used during forward propagation? Activation functions introduce non-linearity into the neural network, enabling it to model complex relationships between inputs and outputs. During forward propagation:

Each neuron's output is passed through an activation function. Examples of activation functions: ReLU: $f (z)$

$\max (0 , z)$ f(z)=max(0,z) for hidden layers. Sigmoid: $f (z)$

$1 1 + e - z$ f(z)= 1+e −z

1for binary outputs. Softmax: Used in the output layer for multi-class classification. Q4. What is the role of weights and biases in forward propagation? Weights: Determine the

importance of each input feature by scaling them. They are learned during training to minimize the error. Biases: Allow the activation function to shift, enabling the model to better fit the data even if all inputs are zero. Together, weights and biases control how inputs are transformed at each layer of the network.

Q5. What is the purpose of applying a softmax function in the output layer during forward propagation? The softmax function is used in the output layer for multi-class classification. It converts raw scores (logits) into probabilities for each class, ensuring:

Probabilities sum to 1. The output represents a likelihood for each class. Mathematically:

$$P(yi)$$

$$ezi\sum jezj \quad \text{P(y i)= } \sum \text{je z j}$$

e z i

Where zi z iis the raw score for class i i.

Q6. What is the purpose of backward propagation in a neural network? Backward propagation (backpropagation) is used to update the weights and biases of a neural network by:

Calculating the gradient of the loss function with respect to weights and biases using the chain rule. Propagating the error backward through the network. Optimizing the weights and biases to minimize the loss function, typically via gradient descent. Q7. How is backward propagation mathematically calculated in a single-layer feedforward neural network? Compute the error:

$$E$$

$$ypred-ytrue \quad \text{E=y pred−y true}$$

Compute the gradient of the loss function with respect to weights:

$$\partial L\partial wi$$

$$E\cdot f'(z)\cdot xi \quad \partial \text{w i}$$

$$\partial \text{L=E}\cdot \text{f}'(\text{z})\cdot \text{x i}$$

Where $f'(z)$ f ′ (z) is the derivative of the activation function.

Update weights and biases:

$w\ i \leftarrow w\ i - \eta \cdot \partial\ L\ \partial\ w\ i$ w i←w i−η· ∂w i

∂L

$b \leftarrow b - \eta \cdot \partial\ L\ \partial\ b$ b←b−η· ∂b ∂L

Here, η η is the learning rate.

Q8. Can you explain the concept of the chain rule and its application in backward propagation? The chain rule is a calculus principle that helps compute the derivative of a composite function. In backpropagation:

The error is propagated backward layer by layer. For each parameter (e.g., weights), the gradient is calculated as: $\partial\ L\ \partial\ w\ i$

$\partial\ L\ \partial\ z \cdot \partial\ z\ \partial\ w\ i$ ∂w i

∂L

∂z ∂L· ∂w i

∂z

This ensures that the gradient of the loss with respect to any parameter is computed efficiently, enabling weight updates. Q9. What are some common challenges or issues that can occur during backward propagation, and how can they be addressed? Vanishing Gradients:

Occurs when gradients become very small, slowing down learning. Solution: Use activation functions like ReLU instead of sigmoid or tanh. Exploding Gradients:

Gradients become excessively large, causing instability. Solution: Apply gradient clipping or use normalized weight initialization. Overfitting:

The model performs well on training data but poorly on unseen data. Solution: Use regularization techniques (L1/L2), dropout, or early stopping. Slow Convergence:

Learning takes too long. Solution: Use adaptive optimizers like Adam or RMSProp. Choosing Hyperparameters:

Selecting an inappropriate learning rate or number of epochs. Solution: Perform hyperparameter tuning using grid search or random search.