Q1. Define Binary cross-entropy as cost function

Ans: Binary cross-entropy is a cost function commonly used in binary classification problems. It measures the difference between the predicted probability distribution and the actual probability distribution of the target variable.

Assuming we have a binary classification problem with two classes, 0 and 1, let's denote the predicted probability of the positive class (class 1) as $\hat{y}$ and the actual label as y . The binary cross-entropy loss function is defined as:

$$-(y * \log \hat{y} + (1 - y) * \log(1 - \hat{y}))$$

In other words, if the actual label is 1, the loss is the negative logarithm of the predicted probability of the positive class. Conversely, if the actual label is 0, the loss is the negative logarithm of the predicted probability of the negative class.

The binary cross-entropy loss function is commonly used as the objective function in training logistic regression models, as well as in deep learning models for binary classification problems such as neural networks with a sigmoid activation function in the output layer. Its gradient is easy to compute and it encourages the model to output high probabilities for the correct class and low probabilities for the incorrect class.

Q2. Difference between Adam optimizer and gradient descent.

Ans: Gradient descent is an iterative optimization algorithm used for minimizing the loss function of a machine learning model during training. Adam (Adaptive Moment Estimation) optimizer is an optimization algorithm used for training machine learning models. Differences between these two are:

1. Updates to model parameters: In gradient descent, the model parameters are updated in the opposite direction of the gradient of the loss function with respect to the parameters, multiplied by a learning rate. In Adam optimizer, the updates are a combination of the first and second moments of the gradients, scaled by a learning rate.
2. Learning rate: In gradient descent, the learning rate is usually constant and the same for all parameters. In Adam optimizer, the learning rate is adaptive, meaning it can vary for each parameter based on the variance and momentum of the gradients.
3. Momentum: In Adam optimizer, the momentum term allows the optimizer to accelerate in the right direction and dampen oscillations, similar to a ball rolling down a hill. In gradient descent, momentum can also be used by adding a fraction of the previous update to the current update.
4. Initialization: In gradient descent, the model parameters are usually initialized randomly and the optimizer starts from there. In Adam optimizer, the moving averages of the gradients are initialized to zero and the optimizer starts from there.
5. Memory requirements: Adam optimizer requires more memory than gradient descent since it needs to store the moving averages of the gradients and the second moments.