Question 1: Define Binary cross entropy as a cost function.

Answer: Binary cross-entropy is a cost function commonly used in binary classification problems, where the goal is to predict one of two possible outcomes. It measures the dissimilarity between the predicted probability distribution and the true distribution. The function calculates the negative log-likelihood of the true labels given a probability prediction of the classifier.

The binary cross-entropy cost function is defined as:

$C = -(1/n) * \sum(y * \log(p) + (1-y) * \log(1-p))$

Here,

- ➔ C is the cost.
- ➔ n is the total number of data in the dataset.
- ➔ y is the true label(0 or 1)
- ➔ p is the predicted probability of the positive class(between 0 and 1)

Example: Suppose we have a binary classification problem where the goal is to predict whether a patient has cancer or not.

The true labels and predicted probabilities for the patients are:

- Patient 1: No cancer ($y = 0$, $p = 0.1$)
- Patient 2: Cancer ($y = 1$, $p = 0.9$)
- Patient 3: No cancer ($y = 0$, $p = 0.2$)
- Patient 4: Cancer ($y = 1$, $p = 0.8$)
- Patient 5: No cancer ($y = 0$, $p = 0.3$)

The cost for each patient can be calculated as:

- Patient 1: -(0 * log(0.1) + 1 * log(1 - 0.1))
  = -log(0.9)
  = 0.152
- Patient 2: -(1 * log(0.9) + 0 * log(1 - 0.9))
  = -log(0.9)
  = 0.152
- Patient 3: -(0 * log(0.2) + 1 * log(1 - 0.2))
  = -log(0.8)
  = 0.322
- Patient 4: -(1 * log(0.8) + 0 * log(1 - 0.8))
  = -log(0.8)
  = 0.322
- Patient 5: -(0 * log(0.3) + 1 * log(1 - 0.3))
  = -log(0.7)
  = 0.514

To find the total cost, we need to sum the cost of each patient and divide it by the number of patients:

C = (0.152 + 0.152 + 0.322 + 0.322 + 0.514) / 5 = 0.2924

So the binary cross-entropy cost for this data is 0.2924.

Question 2: Difference between adam optimizer & gradient descent.

Answer: Adam and gradient descent are both optimization algorithms that are used to update the parameters of a machine learning model during training. However, they differ in how they approach the optimization process.

Gradient descent is a first-order optimization algorithm that iteratively updates the parameters of the model in the direction of the negative gradient of the loss function with respect to the parameters. This means that the algorithm takes the derivative of the loss function with respect to each parameter and adjusts the value of that parameter in the opposite direction of the derivative. The learning rate is a hyperparameter that controls the step size of the update. A smaller learning rate results in smaller steps and a slower convergence, while a larger learning rate results in larger steps and faster convergence.

Adam, on the other hand, is a more advanced optimization algorithm that combines the ideas of gradient descent with momentum and adaptive learning rates(Root mean square propagation). It maintains a running average of the gradient and the squared gradient and uses these averages to adapt the learning rate for each parameter. This allows Adam to make larger updates for parameters with a consistent direction and smaller updates for parameters with a changing direction. This makes Adam more robust to noise and allows it to converge faster than vanilla gradient descent.

In summary, Gradient descent is a simple optimization algorithm that updates the model parameters based on the negative gradient of the loss function with respect to the parameters, while Adam is a more advanced optimization algorithm that adapts the learning rate for each parameter, making it more robust and faster to converge.