Question: Difference between adam optimizer and gradient descent

Answer:

Adam optimizer and gradient descent are both optimization algorithms used to update the weights of a neural network during the training process. However, there are some key differences between them:

Update rule:

Gradient descent computes the gradient of the loss function with respect to the weights and updates the weights by subtracting a multiple of this gradient from the current weights. The learning rate controls the step size of the update. Adam optimizer also computes the gradient of the loss function, but it uses a more complex update rule that incorporates a moving average of past gradients and past squared gradients.

Learning rate:

In gradient descent, the learning rate is fixed and the same for all weight updates. This can lead to slow convergence or oscillations if the learning rate is not set correctly. In contrast, Adam optimizer adapts the learning rate for each weight based on the magnitude of the gradients and the past history of the gradients. This can lead to faster convergence and better performance.

Momentum:

Gradient descent can be improved by adding momentum, which accumulates past gradients to accelerate convergence in directions that have a consistent gradient. Adam optimizer includes a similar concept of momentum in its update rule, which helps to smooth out the updates and prevent oscillations.

Regularization:

Adam optimizer includes L2 regularization by default, which can help prevent overfitting by adding a penalty term to the loss function that discourages large weights. Gradient descent can also be modified to include regularization, but it is not built into the algorithm by default.

Overall, Adam optimizer is a more advanced and adaptable optimization algorithm than simple gradient descent, but it may require more computational resources and can be more sensitive to the choice of hyperparameters.

Question : Define binary cross entropy as cost function

Answer :

Binary cross-entropy is a commonly used cost function in binary classification problems. The binary cross-entropy loss measures the dissimilarity between the predicted probability distribution and the true distribution of the binary target. The binary target is typically represented as a one-hot encoded vector with values 0 or 1. The predicted probabilities can be generated by a model, such as a neural network, that outputs a probability for each class.

The binary cross-entropy is defined as:

$$- (y * \log(p) + (1 - y) * \log(1 - p))$$

Where:

- y is the true binary target
- p is the predicted probability for the positive class

The binary cross-entropy is a measure of how well the predicted probabilities match the true binary targets. The goal of training a model is to minimize the binary cross-entropy, so that the model produces probabilities that are as close as possible to the true binary targets.