#Task-6
Name: Prakanshu Pathak
Registration ID: SIRSS2327

Q1. Calculate/ derive the gradients used to update the parameters in cost function optimization for simple linear regression.

A simple linear regression model tries to explain the relationship between one dependent (output) variable and one independent (predictor) variable using a straight line, which is represented by the following formula:

y = mx +c

where y is the dependent variable, x is the independent variable, m (the slope) is known as the ***weight*** and c (the y-intercept) is known as the ***bias***

A cost function is a measure of the error in prediction committed by an algorithm. It indicates the difference between the predicted and the actual values for a given dataset. Here, we use the Mean Squared Error function as the cost function.

$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_{i\,pred})^2$$

where $y_{i\,pred} = mx_i + c$, so the equation becomes:

$$Cost\ Function(MSE) = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

Closer the predicted value to the actual value, the smaller the difference and lower the value of the cost function We have to minimize the cost function as much as possible in order to find the best fit line. Gradient Descent (GD) is an algorithm that finds the best-fit line for a given training dataset in a smaller number of iterations. Using GD, the gradients are calculated as follows:

The first gradient can be calculated by taking the partial derivative of the cost function with respect to weight m.

$$D_m = \frac{\partial(Cost\ Function)}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial m}\left(\sum_{i=0}^{n}\left(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic\right)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i\left(y_i - (mx_i + c)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}x_i\left(y_i - y_{i\,pred}\right)$$

Again, the second gradient can be calculated by partial derivative of the cost function with respect to bias c.

$$D_c = \frac{\partial(Cost\ Function)}{\partial c} = \frac{\partial}{\partial c}\left(\frac{1}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)^2\right)$$

$$= \frac{1}{n}\frac{\partial}{\partial c}\left(\sum_{i=0}^{n}\left(y_i^2 + m^2x_i^2 + c^2 + 2mx_ic - 2y_imx_i - 2y_ic\right)\right)$$

$$= \frac{-2}{n}\sum_{i=0}^{n}\left(y_i - (mx_i + c)\right)$$

$$\frac{-2}{n}\sum_{i=0}^{n}\left(y_i - y_{i\,pred}\right)$$

These gradients are then used to update the parameters as per the following equations:

$$m = m - LD_m$$

$$c = c - LDc$$

where L is the Learning rate

Q2. What does the sign of gradient say about the relationship between the parameters and cost function?

In a gradient descent, the sign of the gradient refers the minimization of the cost function.

We can show this by using the equation of one of the update steps. For example:

$$m = m - LD_m$$

When the sign of the gradient is positive, the step will decrease, as shown below:

m = m – L*(+ve gradient) m = m – L*|gradient|

When the sign of the gradient is negative, the step will increase.

m = m – L*(–ve gradient) m = m + L*|gradient|

Q3. Why Mean squared error is taken as the cost function for regression problems.

The Mean Squared Error (MSE) is one of the simplest and most common loss functions. It is defined by the equation:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

The MSE tells us how close a regression line is to a set of points. It will never be negative, since the errors are always squared to remove the negative signs. It also gives more weight to larger differences.
The error keeps decreasing as the algorithm gains more and more experience. The smaller the MSE, the closer we are to finding the line of best fit.
We use MSE to ensure that our trained model has no outlier predictions with huge errors, since the MSE puts larger weight on these errors due to the squaring part of the function.

Q4. What is the effect of learning rate on optimization? Discuss all the cases.

Learning rate is used to scale the magnitude of parameter updates during gradient descent. Firstly, the choice of the value for learning rate can impact how fast the algorithm learns. It essentially controls the rate or speed at which the model learns. Given a perfectly configured learning rate, the model will learn to best approximate the function given available resources in a given number of training epochs.
Secondly, learning rate can also impact whether the cost function is minimized or not.
Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.

A learning rate that is too large will result in weight updates that will be too large and the performance of the model will oscillate over training epochs. A learning rate that is too small may never converge or may get stuck on a suboptimal solution.