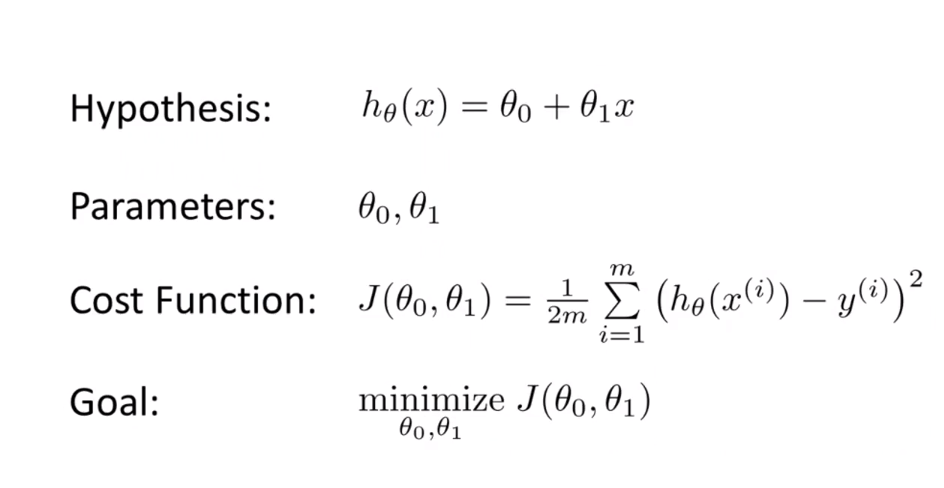
Name : Shubhi Jain

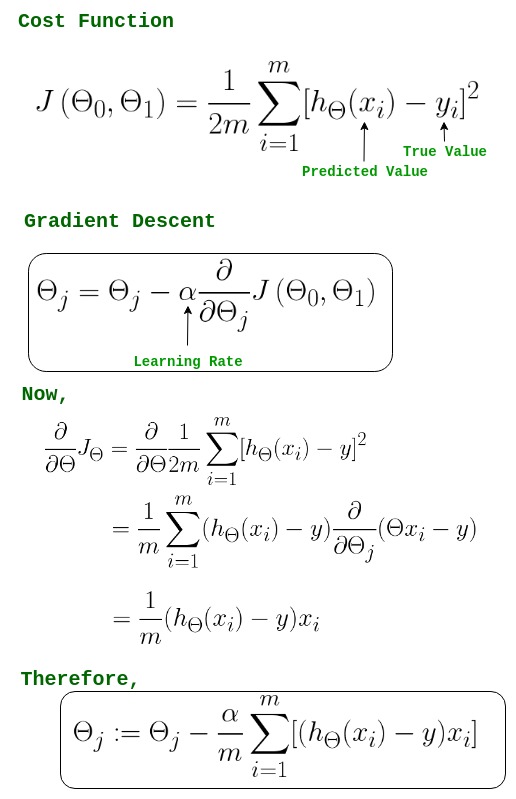
Reg ID : SIRSS1121

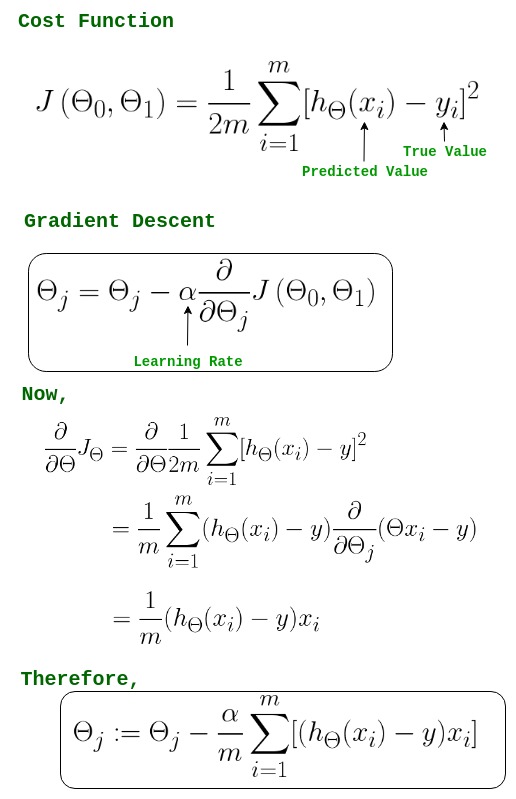
1. **CALCULATE/ DERIVE THE GRADIENTS USED TO UPDATE THE PARAMETERS IN COST FUNCTION OPTIMIZATION FOR SIMPLE LINEAR REGRESSION.**

***Gradient descent*** is an efficient optimization algorithm that attempts to find a local or global minimum of the cost function.

* **A *local minimum*** is a point where our function is lower than all neighboring points. It is not possible to decrease the value of the cost function by making infinitesimal steps.
* **A *global minimum*** is a point that obtains the absolute lowest value of our function, but global minima are difficult to compute in practice.





****

Math behind Gradient Descent :

W = [w0, w1, w2, …., wn]

b = bias

C (W, b) = cost function involving parameters W and b.

*Partial differentiation of Cost Function with respect to weights :*

dW = ∂ / ∂w (W, b)

*Partial differentiation of Cost Function with respect to bias :*

db = ∂ / ∂b (W, b)

Update parameters W and b :

W = W – (a\*dW)

b = b – (a\*db)

**2. WHAT DOES THE SIGN OF GRADIENT SAY ABOUT THE RELATIONSHIP BETWEEN THE PARAMETERS AND COST FUNCTION ?**

The cost function is a function of the parameters.

**CASE I :** When signis **positive,** then step will **decrease.**

*Equation :- ao = aO – [+ve gradient]\*a*

**CASE II :** When sign is **negative,** then step will **increase.**

*Equation :- aO = aO + [gradient]\*a*

**3. WHY MEAN SQUARED ERROR IS TAKEN AS THE COST FUNCTION FOR REGRESSION PROBLEMS.**

Mean Squared Error (MSE) is the sum of the squared differences between the prediction and true value. And the output is a single number representing the **cost**. So the line with the minimum cost function or MSE represents the relationship between X and Y.

**EXAMPLE :**

Let us take an example of actual demand and forecasted demand for a brand of ice cream in a shop in a year.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Month** | **Actual Demand** | **Forecasted Demand** | **Error** | **Squared Error** |
| 1 | 42 | 44 | -2 | 4 |
| 2 | 45 | 46 | -1 | 1 |
| 3 | 49 | 48 | 1 | 1 |
| 4 | 55 | 50 | 5 | 25 |
| 5 | 57 | 55 | 2 | 4 |
| 6 | 60 | 60 | 0 | 0 |
| 7 | 62 | 64 | -2 | 4 |
| 8 | 58 | 60 | -2 | 4 |
| 9 | 54 | 53 | 1 | 1 |
| 10 | 50 | 48 | 2 | 4 |
| 11 | 44 | 42 | 2 | 4 |
| 12 | 40 | 38 | 2 | 4 |
| **SUM** |  |  |  | **56** |

**MSE = 56 / 12 = 4.6667**

**OBSERVATIONS :**

1. As forecasted values can be less than or more than actual values, a simple sum of difference can be zero. This can lead to a false interpretation that forecast is accurate
2. As we take a square, all errors are positive, and mean is positive indicating there is some difference in estimates and actual. Lower mean indicates forecast is closer to actual.
3. All errors in the above example are in the range of 0 to 2 except 1, which is 5. As we square it, the difference between this and other squares increases. And this single high value leads to higher mean. So MSE is influenced by large deviators or outliers.

**CONCLUSION :**

MSE is used to check how close estimates or forecasts are to actual values. Lower the MSE , closer is forecast to actual. This is used as a model evaluation measure for regression models and the lower value indicates a better fit.

1. **WHAT IS THE EFFECT OF LEARNING RATE ON OPTIMIZATION, DISCUSS ALL THE CASES.**

In machine learning, we deal with two types of parameters :

1) machine learnable parameters

2) hyper-parameters.

The ***Machine learnable parameters*** are the one which the algorithms learn/estimate on their own during the training for a given dataset.

The ***Hyper-parameters*** are the one which the machine learning engineers or data scientists will assign specific values to, to control the way the algorithms learn and also to tune the performance of the model.

***Learning rate,*** generally represented by the symbol ‘α’, is a hyper-parameter used to control the rate at which an algorithm updates the parameter estimates or learns the values of the parameters.

When you do deep learning, you want to have the network learn fast and precise as the same time. So therefore, the three different options to find a balance between cautious and impetuous are :

* Oscillate the learning rate between high and low to create a hybrid.
* Adjust the learning rate during training from high to low, too slow down once you get closer to an optimal solution.
* Decide on a learning rate that is neither too low nor too high, i.e. to find the best trade-off.



**CASE I :- *The Hybrid (Stop & Go Learning Rate)***

The main idea is to go from fast to slow, that you should do it more than once during the training. There are two advantages of this one :

1. The jump back to high learning rates helps to avoid local optima.
2. High learning rates are faster in crossing flat areas of the loss function.

The first case is to repeat the second case several times.

**CASE II :- *The Sequence (Lower Learning Rate over time)***

The second case is to start with a high learning rate to harness speed advantages and to switch to a small learning rate, later on to optimize the result.

There are 2 main variations :

1. Firstly, you can adapt the learning rate in response to change in the loss function , i.e. every time the loss function stops to improve, you decrease the learning rate to optimize further.
2. Secondly, you can apply a smoother functional form and adjust learning rate in relation to training time, i.e. the learning rate decreases without direct relation to loss function.

Both approaches and their variations are improvements compared to a fixed learning rate.

**CASE III :- *The Trade-off (Fixed Learning Rate)***

The most basic approach is to stick to the default value and hope for the best. A better implementation of the first case is to test a broad range of possible values, depending on how the loss changes, you go for a higher or lower learning rate.

The aim is to find the fastest rate that still decreases the loss.