Gradient Descent

Gradient Descent is an optimization algorithm for minimizing the Cost Function. The algorithm iteratively adjusts the model's parameters to find the combination of weights that results in the smallest possible value for the Cost Function.In simpler terms, Gradient Descent navigates through the different possible values of the parameters to find those that lead to the least amount of error, as indicated by the Cost Function.

Gradient- A gradient in mathematics is often described as the slope of a curve at a particular point. This slope can vary depending on the direction in which it is measured. In the context of functions, a gradient provides vital information about the rate of change of the function's output with respect to changes in its inputs.

Consider the function $f(x) = 2x^2 + 3x$. To understand the gradient of this function, we look at its first-order derivative with respect to x. The derivative, denoted as $df(x)/dx$, provides the rate at which f(x) changes as x changes.

For our function, the first-order derivative is calculated as follows: $df(x)/dx = 4x + 3$

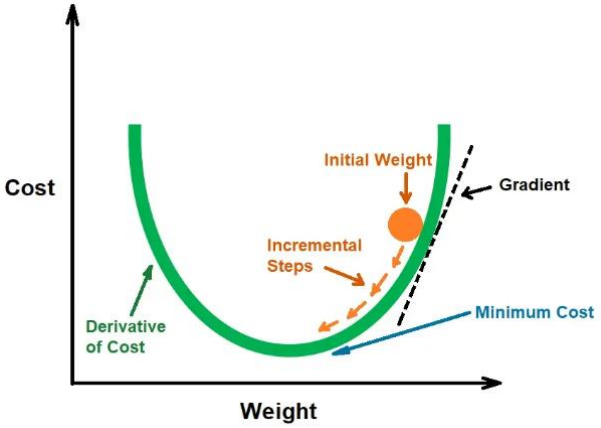
This derivative tells us that for every unit increase in x, the output of the function f(x) changes by an amount equal to 4x + 3. This rate of change is what we refer to as the gradient in the univariate case.

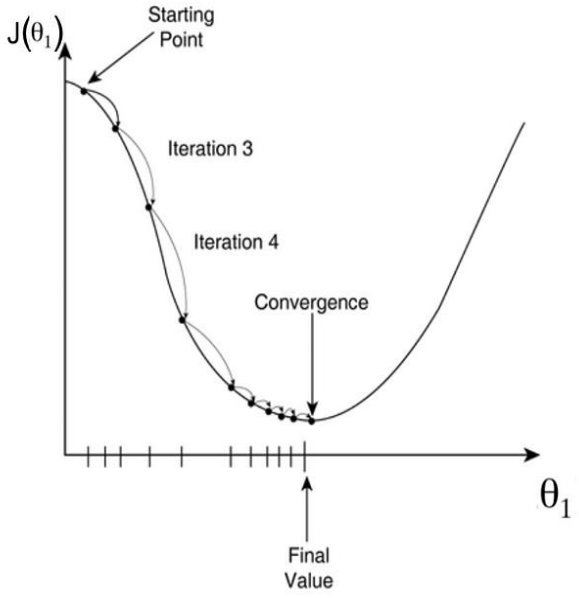
How Does Gradient Descent Work?

1. Starting Point: The process begins with an initial guess for the model's parameters, also known as 'weights'. This starting point is typically random or based on some heuristic.
2. Computing the Gradient: The gradient of the cost function is computed at the current set of parameters. This gradient, which is a vector of partial derivatives, indicates the direction and rate of the steepest increase in cost.
3. Determining the Direction: The sign of each component of the gradient tells us the direction in which the corresponding parameter should be adjusted:
    - A positive gradient for a parameter suggests that increasing that parameter will increase the cost function, and thus, to decrease the cost, we should move in the negative direction.
    - Conversely, a negative gradient implies that decreasing the parameter will increase the cost, so we

should move in the positive direction.

4. Updating Parameters: The parameters are then updated in the opposite direction of the gradient, scaled by a factor known as the 'learning rate'. This step size is crucial as too large a step can overshoot the minimum, while too small a step can lead to a long convergence time.
5. Iterative Process: The steps of computing the gradient and updating the parameters are repeated until the changes in the cost function become negligibly small, indicating that the minimum cost has been approached.





Cost Function – "One Half Mean Squared Error":

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Derivatives:

$$\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

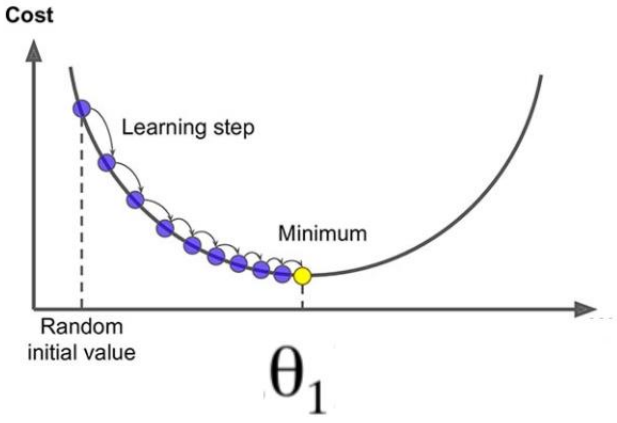
$$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$

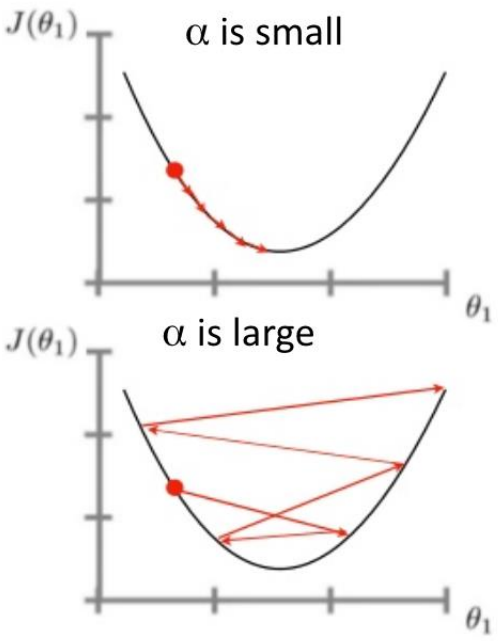
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}





Learning rate (also referred to as step size or the alpha) is the size of the steps that are taken to reach the minimum. This is typically a small value, and it is evaluated and updated based on the behavior of the cost function. High learning rates result in larger steps but risks overshooting the minimum. Conversely, a low learning rate has small step sizes. While it has the advantage of more precision, the number of iterations compromises overall efficiency as this takes more time and computations to reach the minimum.