# Gradient descent

* Gradient Descent is an algorithm used in training a machine learning model.
* The main aim is to minimise the cost function as minimal as possible and also to find the slope and intercept values.
* The equation of this straight line would be **Y = mX + b** where **m** is the slope and **b** is its intercept on the Y-axis.

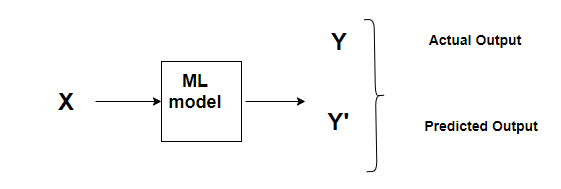
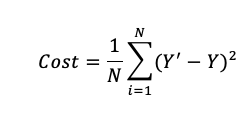


Image for post

## Cost Function

A Cost function basically tells us ‘how good’ our model is at making predictions for a given value of m and b.

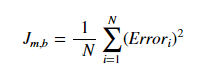


## The Learning rate

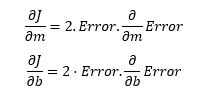
This size of steps taken to reach the minimum or bottom is called **Learning Rate**. We can cover more area with larger steps/higher learning rate but are at the risk of overshooting the minima. On the other hand, small steps/smaller learning rates will consume a lot of time to reach the lowest point.

## Calculating Gradient Descent

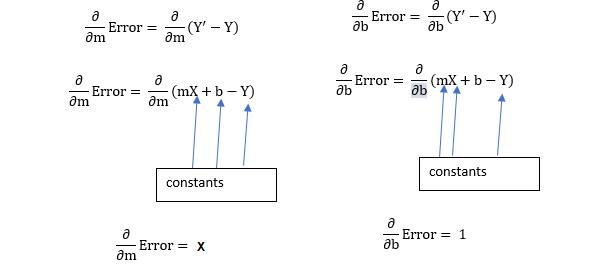
Let us now apply the knowledge of these rules of calculus in our original equation and find the derivative of the Cost Function w.r.t to both **‘m’** and **‘b’**. Revising the Cost Function equation:



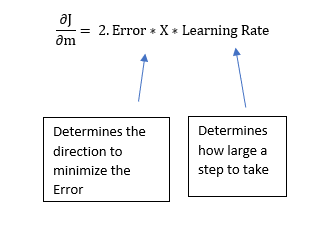
For simplicity, let us get rid of the summation sign. The summation part is important, especially with the concept of **Stochastic gradient descent (SGD )** vs **batch gradient descent**. During the **batch gradient descent**, we look at the error of all the training examples at once while in the **SGD** we look at each error at a time. However, just to keep things simple, we will assume that we are looking at each error one at a time.

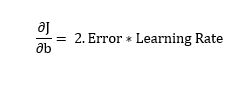


Now let’s calculate the gradient of Error w.r.t to both m and b:

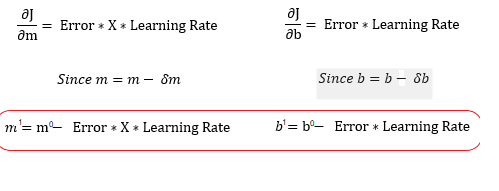


Plugging the values back in the cost function and multiplying it with the learning rate:





Now, this **2** in this equation isn’t that significant since it just says that we have a learning rate twice as big or half as big. So let’s just get rid of it too. So, Ultimately, this entire article boils down to two simple equations which represent the equations for Gradient Descent.



m¹,b¹ = next position parameters; m⁰,b⁰ = current position parameters

## Summarizing the above concept

1. Initially Substitute values to m=1 & b=0 in the linear equation

y=mx+b. Calculate the cost function value

1. Calculate new m value based on previous value of m.

M\_new = m\_old – (step function)

Step function = (d/dm)\*(learning rate)

(d/dm) = 2/N ∑ [-x (y-(mx+b)) ]

1. Calculate new b value based on previous value of b.

B\_new = b\_old – (step function)

Step function = (d/db)\*(learning rate)

(d/db) = 2/N ∑ [-(y - (mx+b)) ]

1. Now assign the new m & b values to old m&b values.
2. Repeat the step 2 to 4 until the error is minimized.