**Chapter 1.2: Gradient Descent with Math.**

**Reference : medium.com // Author: Madhu Sanjeevi**

This story I want to talk about a famous machine learning algorithm called *Gradient Descent* which is used for optimizing the machine leaning algorithms and how it works including the math.

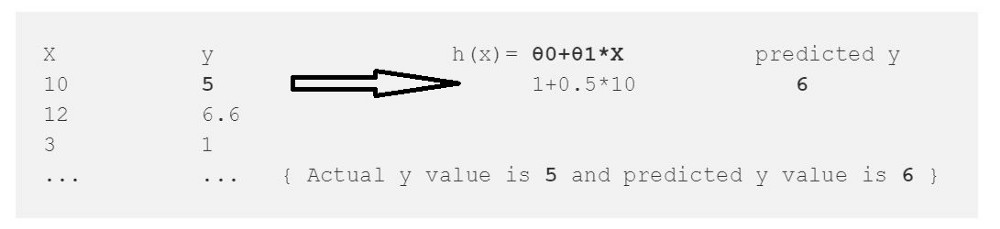
From [chapter 1](https://medium.com/@madhusanjeevi.ai/complete-linear-regression-with-math-edb05500e7ee) we know that we need to update *m* and *b* values, we call them **weights** in machine learning. Lets alias *b* and *m* as −*θ0* and *θ1* (theta 0 and theta 1 ) respectively.

First time we take random values for *θ0* and *θ1*, and we calculate y

**y = θ0+θ1\*X**   
In machine learning we say hypothesis so **h(X) = θ0+θ1\*X**

h(X)=y but this y is not actual value in our data-set, this is predicted y from our hypothesis.

For example let’s say our data-set is something like below and we take random values which are **1**and **0.5** for **θ0**and **θ1**respectively.

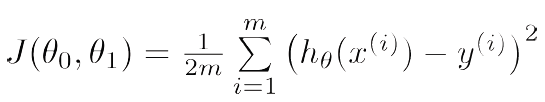


From this we calculate the error which is

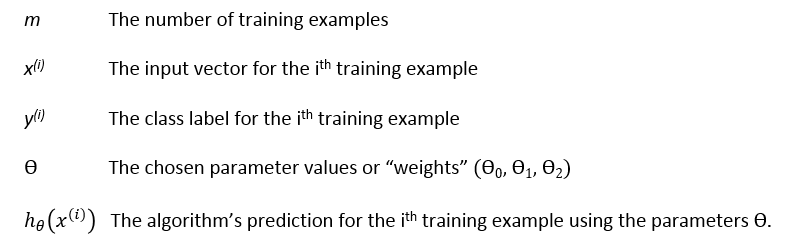
error = (h(x)-y)² --> (Predicted - Actual)²   
error = (6-5)² = 1

² is to get rid of negative values (what if Actual y=6 and Py=5)

we just calculated the error for one data point in our data-set , we need to repeat this for all data points in our data set and sum up the all errors to one error which is called ***Cost Function ‘J(*θ)’**in machine learning.



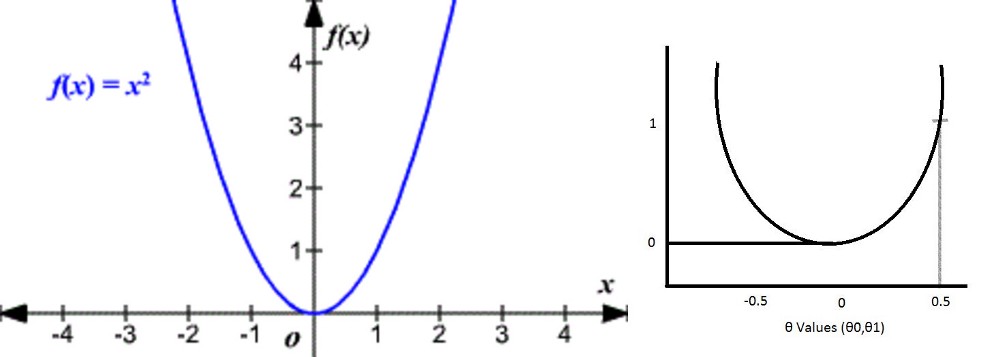
Cost Function.



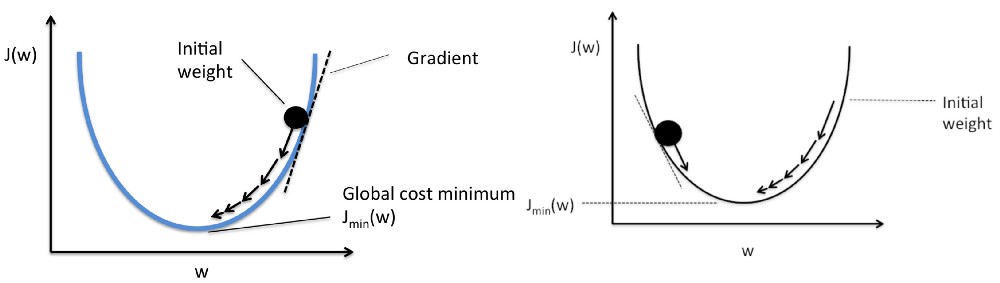
Our goal is to minimize the cost function (error) **we want our error close to zero Period.**

we have the error **1** for first data-point so let’s treat that as whole error and reduce to zero for sake of understanding.

for (h(x)-y)² function we get always positive values and graph will look like this(Left) and lets plot the error graph.



Here is the gradient descent work comes into the picture.



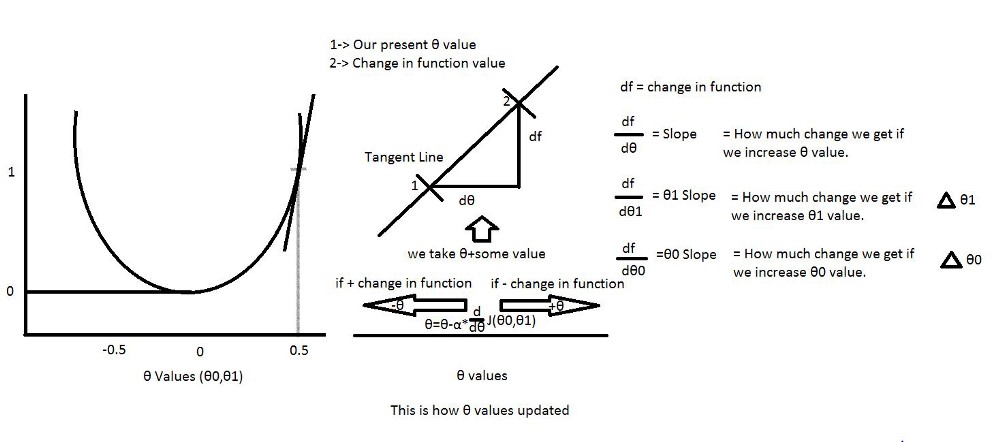
+**θ values (Left), -θ values(Right)**

By taking the little steps down to reach the minimum value (bottom of the curve) and changing the **θ**values in the process.

*How does it know how much value it should go down???*

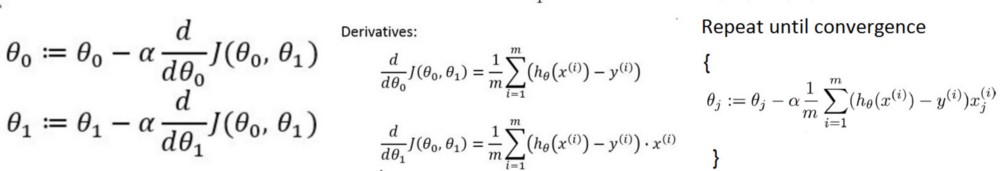
The answer is in Math.

1. It draws the line(Tangent) from the point.
2. It finds the slope of that line.
3. It identifies how much change is required by taking the partial derivative of the function with respective to **θ**
4. The change value will be multiplied with a variable called **alpha**(learning rate) *we provide the value for alpha usually 0.01*
5. It subtracts this change value from the earlier **θ** value to get new **θ**value .



From above picture we can define our **θ0 and θ1.**

And alpha here is a learning rate usually we give 0.01 but it depends, it tells how big the step-size is towards reaching the minimum value.

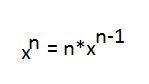


**θ0 and θ1 values(Left),more than two θ’s (Right)**

Again we know our J(**θ0,θ1)**so if we apply this to above equations for **θ0**and **θ1**, we get our new **θ0 and θ1**values**.**

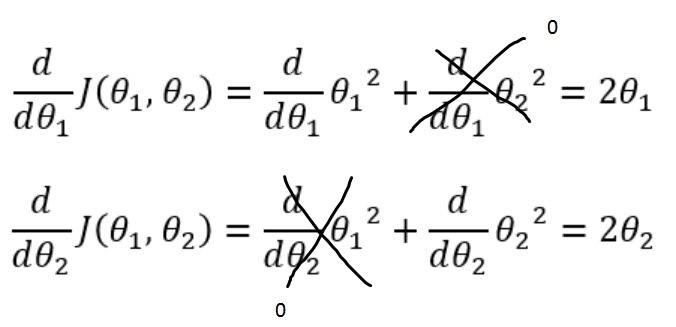
*How to calculate the derivatives???*

For example f(x) =x² → df/dx=2x How ???

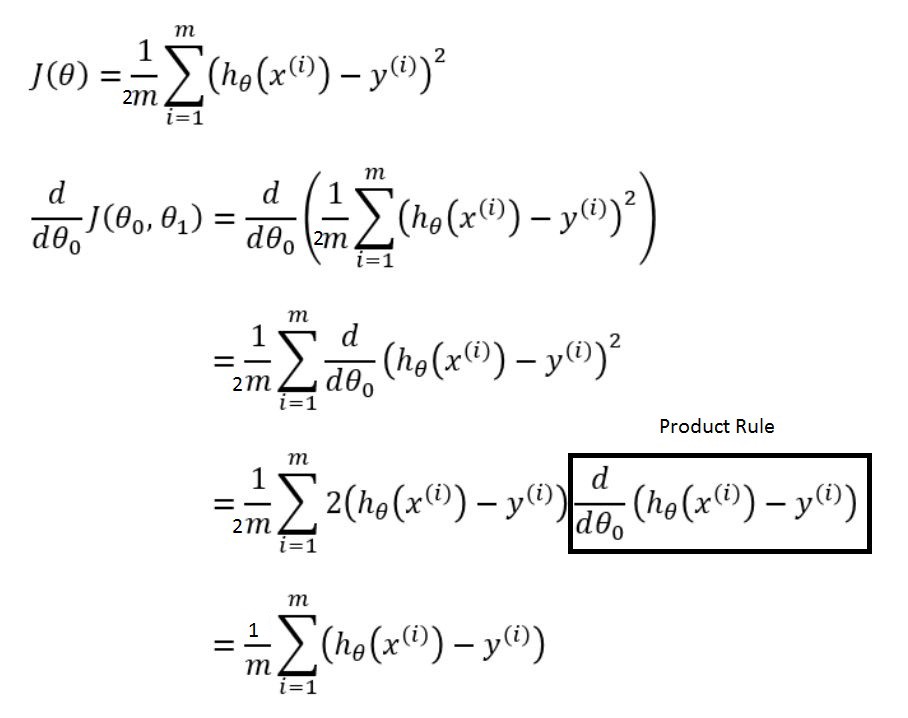


*How to calculate the partial derivatives???*

its same as calculating derivatives but here we calculate the derivative with respective to that value , others are constants (so d/dx(constant)=0)

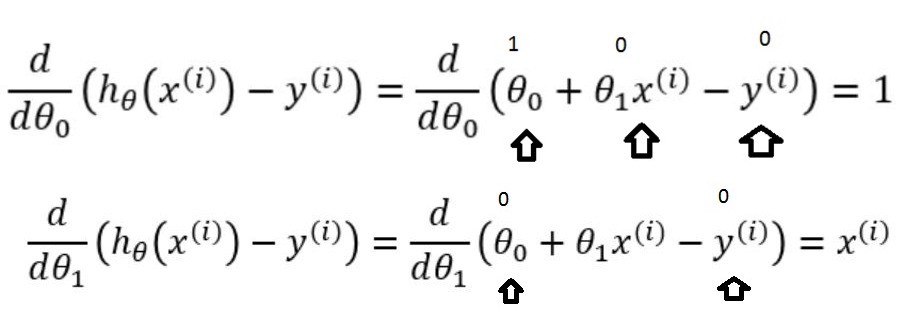


The same thing we can apply for calculating partial derivative with respective to **θ0**and **θ1**.



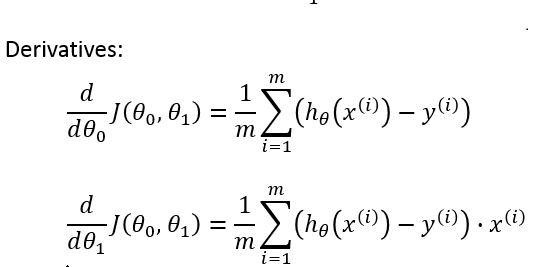
How come that box drawn disappeared in the next step above? just wait and see.

For calculating partial derivative with respective to **θ1**is also same as above except one little part is added



**θ0 box disappeared because value is 1 (Top)**

So Final picture is

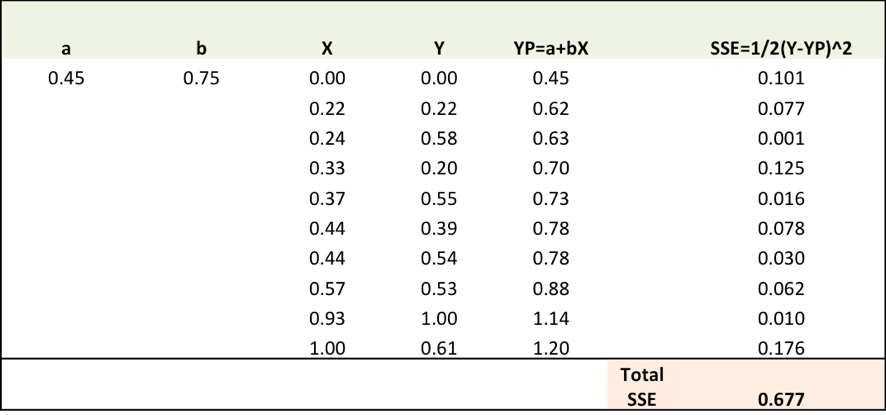


Final **θ0**and **θ1**values

For example:

{

**Step 1:**To fit a line Ypred = a + b X, start off with random values of a and b and calculate prediction error (SSE)



**Step 2:**Calculate the error gradient w.r.t the weights

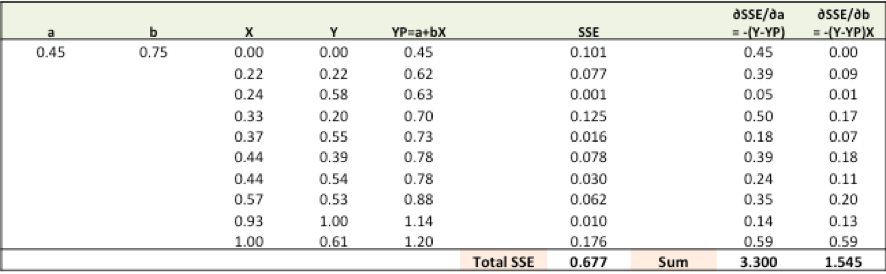
∂SSE/∂a = – (Y-YP)

∂SSE/∂b = – (Y-YP)X

Here, SSE=½ (Y-YP)2 = ½(Y-(a+bX))2

You need to know a bit of calculus, but that’s about it!!

∂SSE/∂a and ∂SSE/∂b are the **gradients** and they give the direction of the movement of a,b w.r.t to SSE.



**Step 3:** Adjust the weights with the gradients to reach the optimal values where SSE is minimized

We need to update the random values of a,b so that we move in the direction of optimal a, b.

Update rules:

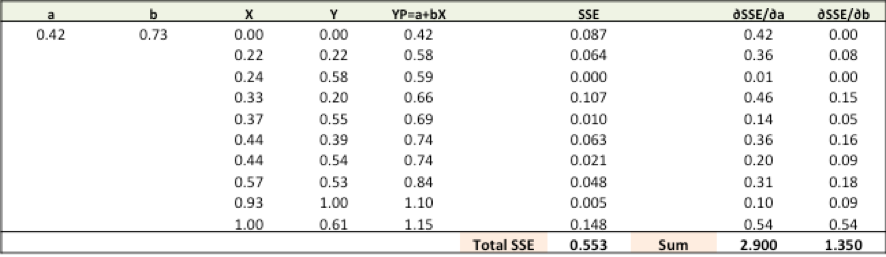
* a – ∂SSE/∂a
* b – ∂SSE/∂b

So, update rules:

1. New a = a – r \* **∂SSE/∂a =**0.45-0.01\*3.300 = 0.42
2. New b = b – r \* **∂SSE/∂b=**0.75-0.01\*1.545 = 0.73

here, r is the learning rate = 0.01, which is the pace of adjustment to the weights.

**Step 4:** Use new a and b for prediction and to calculate new Total SSE



You can see with the new prediction, the total SSE has gone down (0.677 to 0.553). That means prediction accuracy has improved.

**Step 5:**Repeat step 3 and 4 till the time further adjustments to a, b doesn’t significantly reduces the error. At that time, we have arrived at the optimal a,b with the highest prediction accuracy.

This is the Gradient Descent Algorithm. This optimization algorithm and its variants form the core of many machine learning algorithms like Neural Networks and even Deep Learning.

}

Hope it’s not confusing , and I know its little bit hard to grasp in the beginning but I am sure that this will make sense as you go through again and again.

So That’s it for this story , In the next story I will cover another interesting topic in machine learning so See ya!

**LINKS:**

[Code for Gradient Descent and linear regression](https://medium.com/@madhusanjeevi.ai/chapter-1-3-code-for-linear-regression-with-gradient-descent-from-scratch-and-using-libraries-46d07f654cdc)