**Linear Regression**

Each machine learning model is trying to solve a problem with a different objective using a different dataset and hence, it is important to understand the context before choosing a metric. Usually, the answers to the following question help us choose the appropriate metric:

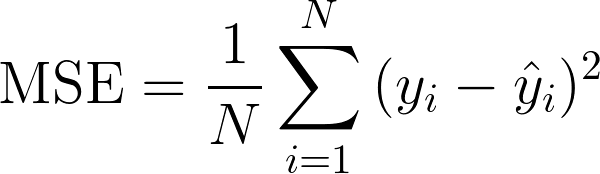
* Type of task: Regression? Classification?
* Business goal?
* What is the distribution of the target variable?

### ****Regression Metrics****

* Mean Squared Error (MSE)
* Root Mean Squared Error (RMSE)
* Mean Absolute Error (MAE)
* R Squared (R²)
* Adjusted R Squared (R²)
* Mean Square Percentage Error (MSPE)
* Mean Absolute Percentage Error (MAPE)
* Root Mean Squared Logarithmic Error (RMSLE

### ****Mean Squared Error (MSE)****

It is perhaps the most simple and common metric for regression evaluation, but also probably the least useful. It is defined by the equation



where *yᵢ* is the actual expected output and *ŷᵢ* is the model’s prediction.

MSE basically measures average squared error of our predictions. For each point, it calculates square difference between the predictions and the target and then average those values.

The higher this value, the worse the model is. It is never negative, since we’re squaring the individual prediction-wise errors before summing them, but would be zero for a perfect model .

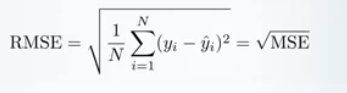
**Advantage:** Useful if we have unexpected values that we should care about. Vey high or low value that we should pay attention.

**Disadvantage:** If we make a single very bad prediction, the squaring will make the error even worse and it may skew the metric towards overestimating the model’s badness. That is a particularly problematic behaviour if we have noisy data (that is, data that for whatever reason is not entirely reliable) — even a “perfect” model may have a high MSE in that situation, so it becomes hard to judge how well the model is performing. On the other hand, if all the errors are small, or rather, smaller than 1, than the opposite effect is felt: we may underestimate the model’s badness.

**Note that** if we want to have a constant prediction the best one will be the **mean value of the target values.** It can be found by setting the derivative of our total error with respect to that constant to zero, and find it from this equation.

### Root Mean Squared Error (RMSE)

RMSE is just the square root of MSE. The square root is introduced to make scale of the errors to be the same as the scale of targets.



Now, it is very important to understand in what sense RMSE is similar to MSE,and what is the difference.

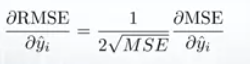
First, they are similar in terms of their minimizers, every minimizer of MSE is also a minimizer for RMSE and vice versa since the square root is an non-decreasing function. For example, if we have two sets of predictions, A and B, and say MSE of A is greater than MSE of B, then we can be sure that RMSE of A is greater RMSE of B.And it also works in the opposite direction.

https://cdn-images-1.medium.com/max/800/1*e9NYGLz3a9wdKpYMuLTI0Q.png

**What does it mean for us?**

It means that, if the target metric is RMSE, we still can compare our models using MSE,since MSE will order the models in the same way as RMSE. Thus we can optimize MSE instead of RMSE.

In fact, MSE is a little bit easier to work with, so everybody uses MSE instead of RMSE. Also a little bit of difference between the two for gradient-based models.



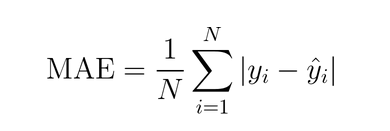
Gradient of RMSE with respect to i-th prediction

It means that travelling along MSE gradient is equivalent to traveling along RMSE gradient but with a different flowing rate and the flowing rate depends on MSE score itself.

So even though RMSE and MSE are really similar in terms of models scoring, they can be not immediately interchangeable for gradient based methods. We will probably need to adjust some parameters like the learning rate.

### Mean Absolute Error (MAE)

In MAE the error is calculated as an average of absolute differences between the target values and the predictions. The MAE is a linear score which means that **all the individual differences are weighted equally** in the average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0. However, same is not true for RMSE. Mathematically, it is calculated using this formula:



What is important about this metric is that it **penalizes huge errors that not as that badly as MSE does.** Thus, it’s not that sensitive to outliers as mean square error.

MAE is widely used in finance, where $10 error is usually exactly two times worse than $5 error. On the other hand, MSE metric thinks that $10 error is four times worse than $5 error. MAE is easier to justify than RMSE.

Because we use the absolute value of the residual, the MAE does not indicate **underperformance** or **overperformance** of the model (whether or not the model under or overshoots actual data). Each residual contributes proportionally to the total amount of error, meaning that larger errors will contribute linearly to the overall error. Like we’ve said above, a small MAE suggests the model is great at prediction, while a large MAE suggests that your model may have trouble in certain areas. A MAE of 0 means that your model is a **perfect** predictor of the outputs (but this will almost never happen).

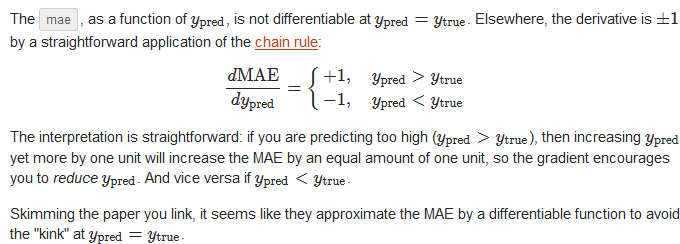
Because we are squaring the difference, the MSE will almost always be bigger than the MAE. For this reason, we cannot directly compare the MAE to the MSE. We can only compare our model’s error metrics to those of a **competing** model. The effect of the square term in the MSE equation is most apparent with the presence of outliers in our data. While each residual in MAE contributes **proportionally** to the total error, the error grows **quadratically** in MSE. This ultimately means that outliers in our data will contribute to much higher total error in the MSE than they would the MAE. Similarly, our model will be penalized more for making predictions that differ greatly from the corresponding actual value.

I would want to use the MSE to ensure that my model takes these outliers into account more.

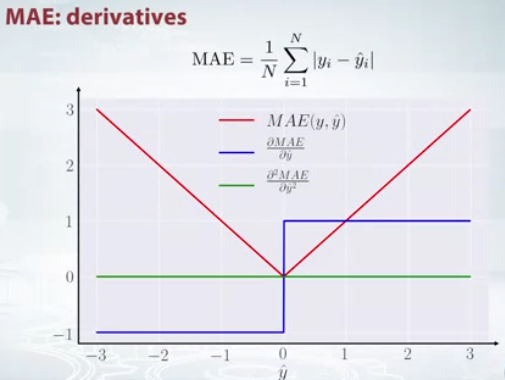
If I wanted to downplay their significance, I would use the MAE since the outlier residuals won’t contribute as much to the total error as MSE. Ultimately, the choice between is MSE and MAE is application-specific and depends on how you want to treat large errors. Both are still viable error metrics, but will describe different nuances about the prediction errors of your model.

MSE has nice mathematical properties which makes it easier to compute the gradient. However, MAE requires more complicated tools such as linear programming to compute the gradient. Because of the square, large errors have relatively greater influence on MSE than do the smaller error. Therefore, MAE is more robust to outliers since it does not make use of square. On the other hand, MSE is more useful if concerning about large errors whose consequences are much bigger than equivalent smaller ones.

**Another important thing about MAE is its gradients with respect to the predictions. The gradient is a step function and it takes -1 when Y\_hat is smaller than the target and +1 when it is larger.**

****

**Now, the gradient is not defined when the prediction is perfect, because when Y\_hat is equal to Y, we can not evaluate gradient. It is not defined**.



So formally, MAE is not differentiable, but in fact, how often your predictions perfectly measure the target. Even if they do, we can write a simple IF condition and return zero when it is the case and through gradient otherwise. Also know that second derivative is zero everywhere and not defined in the point zero.

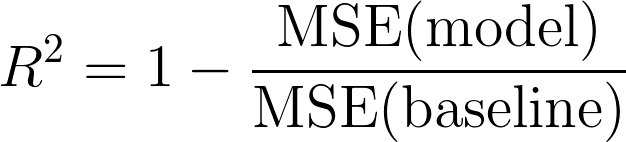
**Note that** if we want to have a constant prediction the best one will be the **median value of the target values.** It can be found by setting the derivative of our total error with respect to that constant to zero, and find it from this equation.

### R Squared (R²)

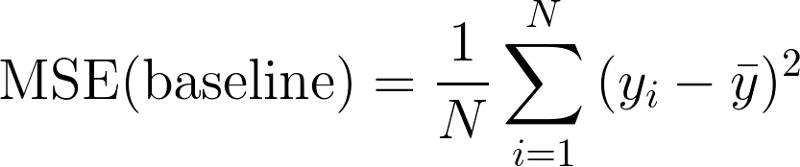
Now, what if I told you that MSE for my models predictions is 32? Should I improve my model or is it good enough?Or what if my MSE was 0.4?Actually, it’s hard to realize if our model is good or not by looking at the absolute values of MSE or RMSE.We would probably want to measure how much our model is better than the constant baseline.

The coefficient of determination, or R² (sometimes read as R-two), is another metric we may use to evaluate a model and it is closely related to MSE, but has the advantage of being **scale-free**— it doesn’t matter if the output values are very large or very small, **the R² is always going to be between -∞ and 1.**

When R² is negative it means that the model is worse than predicting the mean.



The MSE of the model is computed as above, while the MSE of the baseline is defined as:



where the y with a bar is the mean of the observed yᵢ.

To make it more clear, this baseline MSE can be thought of as the MSE that the **simplest possible** model would get. The simplest possible model would be to always predict the average of all samples. A value close to 1 indicates a model with close to zero error, and a value close to zero indicates a model very close to the baseline.

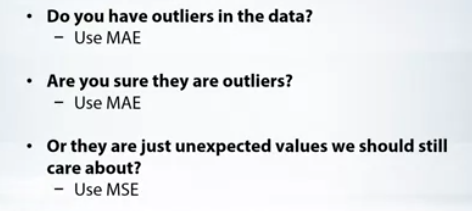
**In conclusion, R² is the ratio between how good our model is vs how good is the naive mean model.**

**Common Misconception:** Alot of articles in the web states that the range of R² lies between 0 and 1 which is not actually true. The maximum value of R² is 1 but minimum can be minus infinity.

For example, consider a really crappy model predicting highly negative value for all the observations even though y\_actual is positive. In this case, R² will be less than 0. This is a highly unlikely scenario but the possibility still exists.

### MAE vs MSE

I stated that MAE is more robust (less sensitive to outliers) than MSE but this doesn’t mean it is always better to use MAE. The following questions help you to decide:

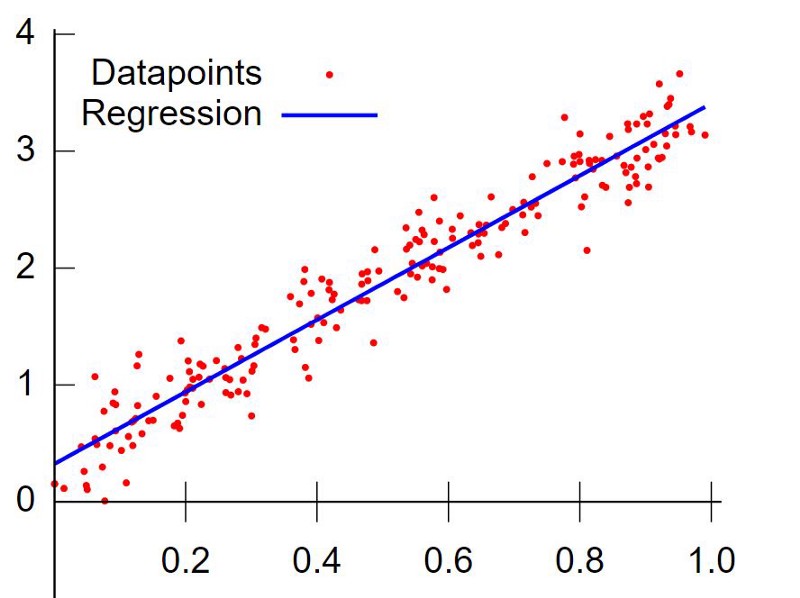


### Take-home message

In this article, we discussed several important regression metrics. We first discussed, Mean Square Error and realized that the best constant for it is the mean targeted value. Root Mean Square Error, and R² are very similar to MSE from optimization perspective. We then discussed Mean Absolute Error and when people prefer to use MAE over MSE.

====================================================================================

Linear regression is a linear model that establishes the relationship between a dependent variable y(Target) and one or more independent variables denoted X(Inputs).



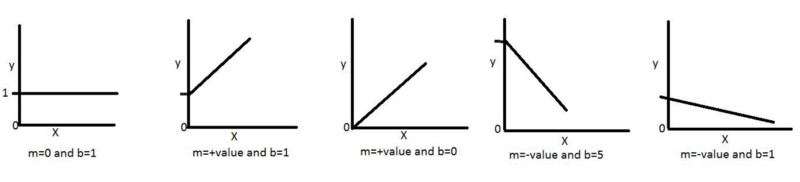
**Goal is to find that blue straight line (which is best fit) to the data.**

Our Training Data consists of X and y values so we can plot them on the graph, that’s damn easy. Now how to find that blue line?

First let’s talk about how to draw a linear line in the graph,

In math we have an equation which is called linear equation

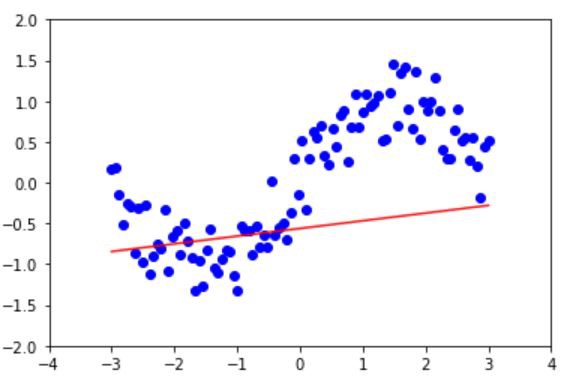
y = mX+b { m->slope , b->Y-intercept }



So we can draw the line if we take any values for m and b

How do we get the m and b values and how do we know exact m and b values for the best fit line??

Let’s take a simple data set (sine wave form -3 to 3) and First time we take random values of m and b values and we draw a line something like this.



Random line for m and b

How we drew the above line?

We take the first X value(x1) from our data set and calculate y value (y1)

y1=m\*x1+b {m,b->random values lets say 0.5,1   
 x1->lets say -3 (first value from our data-set)

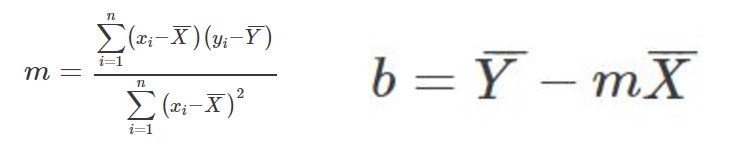
y1=(0.5 \* -3) + 1  
y1=-0.5   
by applying all x values for m and b values we get our first line.

Above picture has its own random variables ( I hope you understand the concept)

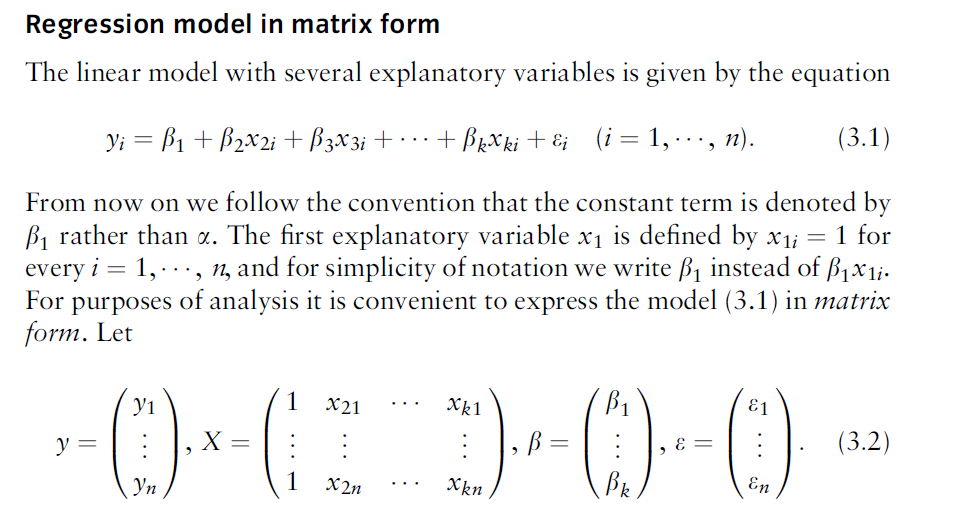
That line is *not* fitting well to the data so we need to change m and b values to get the best fit line.

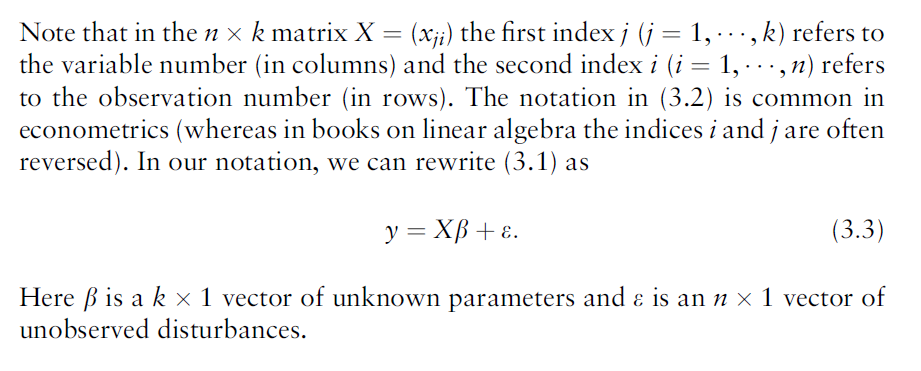
How do we change m and b values for the best fit line??

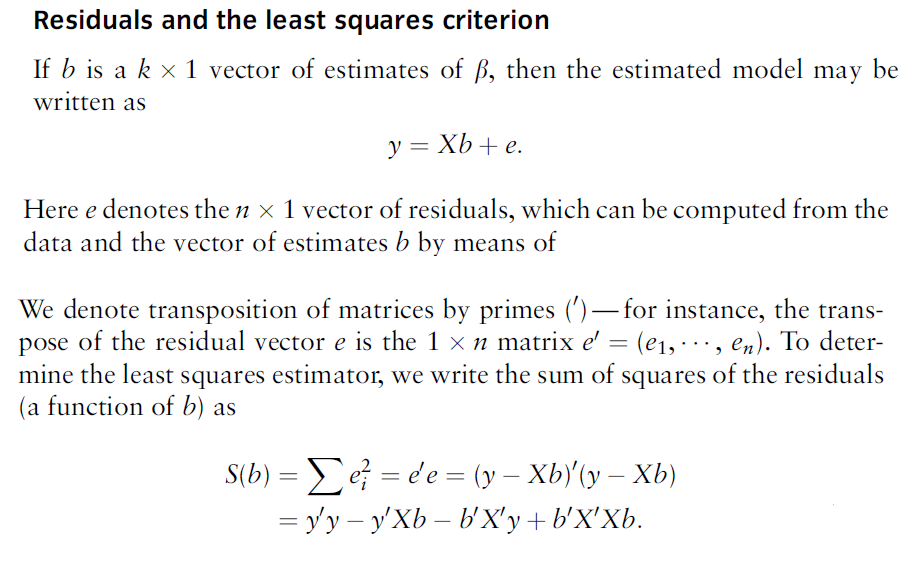
1. Either we can use an awesome algorithm called **Gradient Descent** (Which I will cover in next story with also the math used in there.)
2. We can borrow direct formulas from statistics (they call this **Least Square Method**).

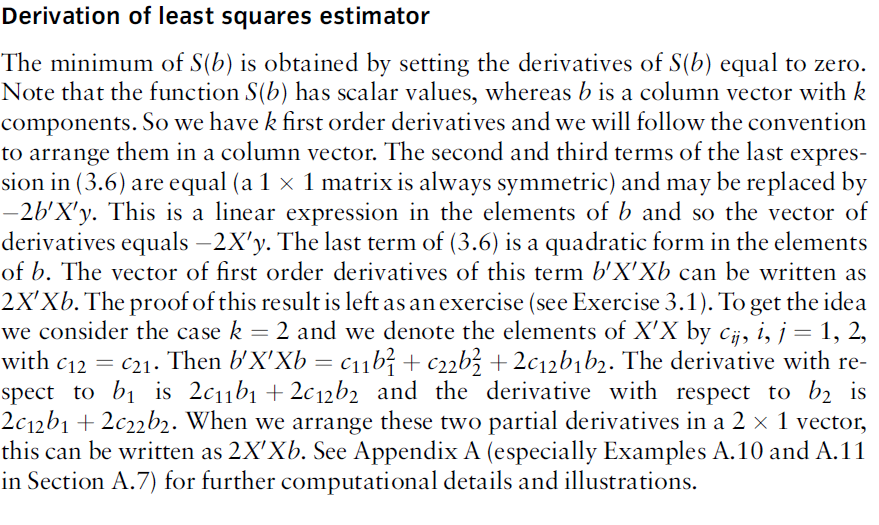


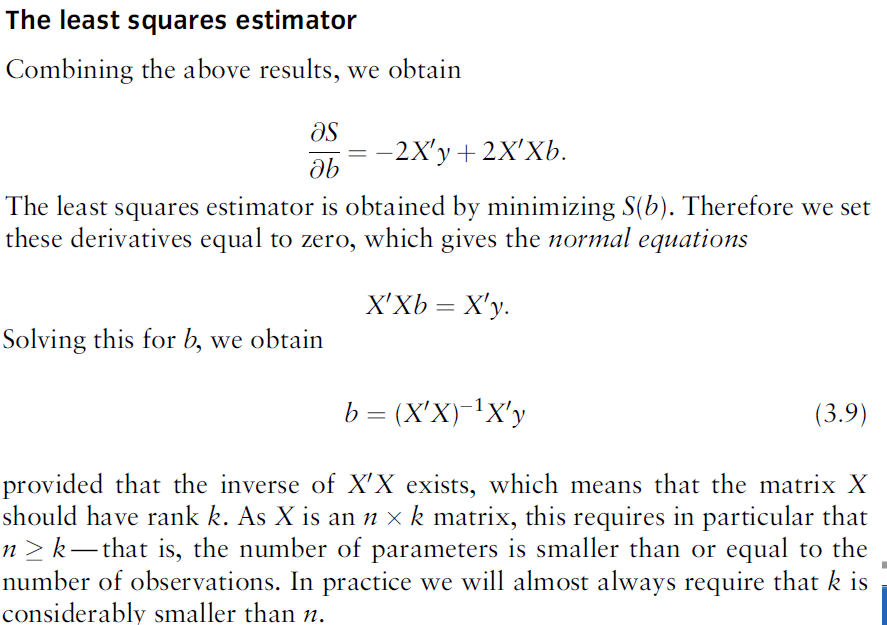
X^ is mean of X values, Y^ mean of y values







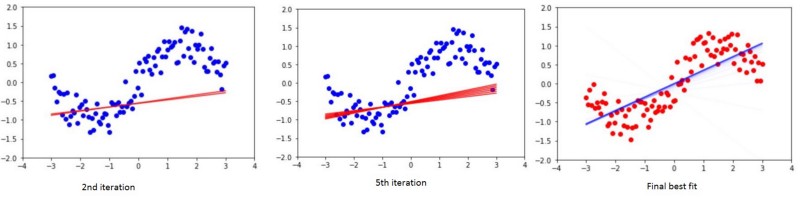




**In this least square method direct calculation of (XX)-1 is the most complex part because if data is input very hugethen this matrix becomes very big and finding inverse of this matrix is very complex and time consuming.**

**So if we have small input data then we can use direct formula above to calculate the value of weights but if input data is huge then we wil go for the Gradient descent methos which world pretty fast on big input data.**

Right now let’s black box, we assume that we are getting the m and b values, Every time when the m and b values change we may get a different line and finally we get the best fit line



So *What’s next???*

*Now we can Predictpng new data, remember so* we give new X values we get the predicted y values how does it work?

Same as above **y= m X +b** , we now know the final m and b values.

This is called **simple linear regression** as we have only one independent X value. Lets say we want predict housing price based the size of house

X= Size (in sqft’s) y= Price (in dollars)

X y  
1000 40  
2000 70  
500 25  
............

**What if we have more independent values of X?**

Let’s say we want predict housing price not only by the size of house but also by no of bedrooms

x1= Size (in sqft’s), x2=N\_rooms and y= Price (in dollar’s)

x1 x2 y  
1000 2 50  
2000 4 90  
500 1 35  
............

The process same as above but the equation changes a bit

Note: Lets alias b and m as − θ0 and θ1 (theta 0 and theta 1 ) respectively.

**y = θ0+θ1\*X → b+mX → Simple LR → Single variable LR**

**y=θ0+θ1\*x1+θ2\*x2+..θn\*xn → Multiple LR → Multi variable LR**

Now we can predict as many things as we wish.

**Maths behind Gradient Descent**

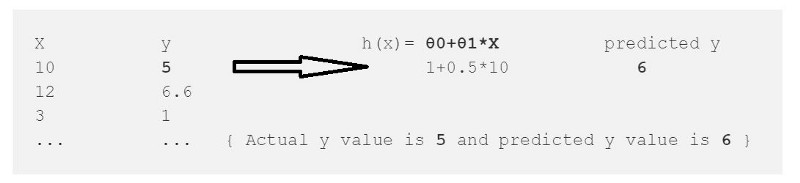
In Linear Regression we need to update *m* and *b* values, we call them **weights** in machine learning. Lets alias *b* and *m* as − *θ0* and *θ1* (theta 0 and theta 1 ) respectively.

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

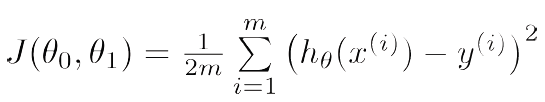


From this we calculate the error which is

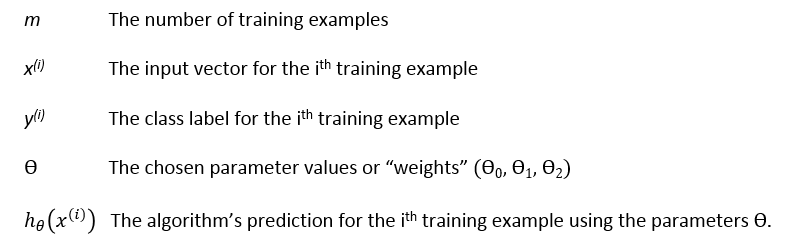
error = (h(x)-y)² --> (Predicted - Actual)²   
error = (6-5)² = 1

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.



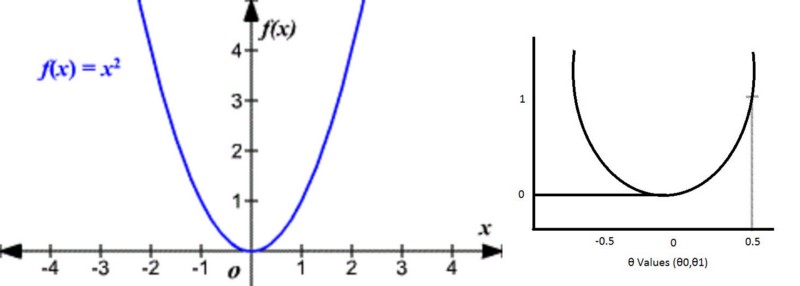
Cost Function.



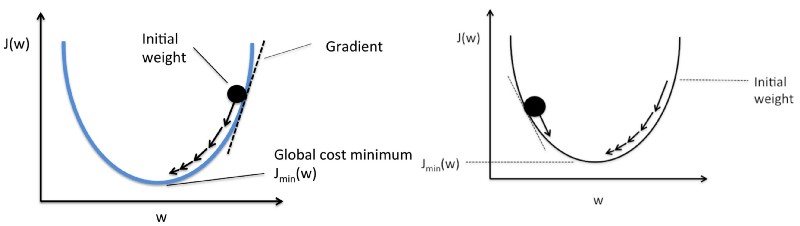
Our goal is to minimize the cost function (error) **we want our error close to zero Period.**

we have the error **1** for first data-point so lets 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.



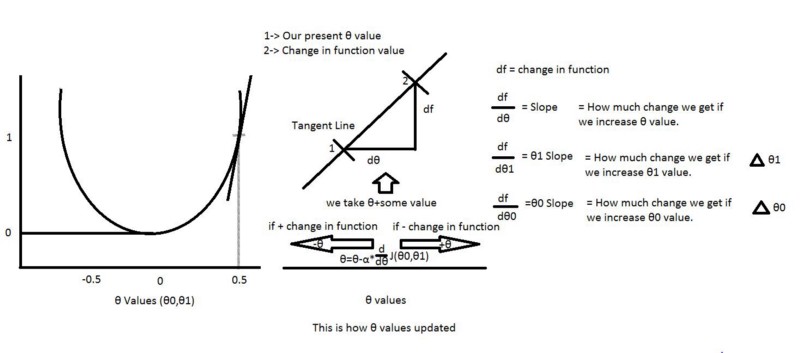
+**θ values (Left), -θ values(Right)**

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???

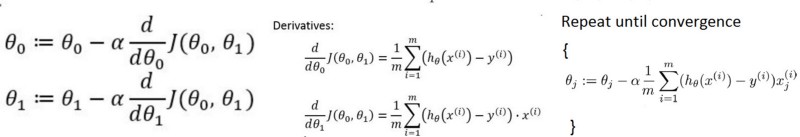
The answer is in Math.

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.

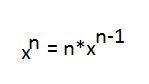


**θ0 and θ1 values(Left),more than two θ’s (Right)**

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**.**

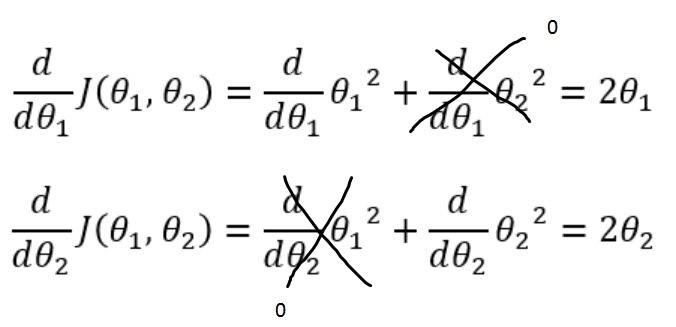
How to calculate the derivatives???

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

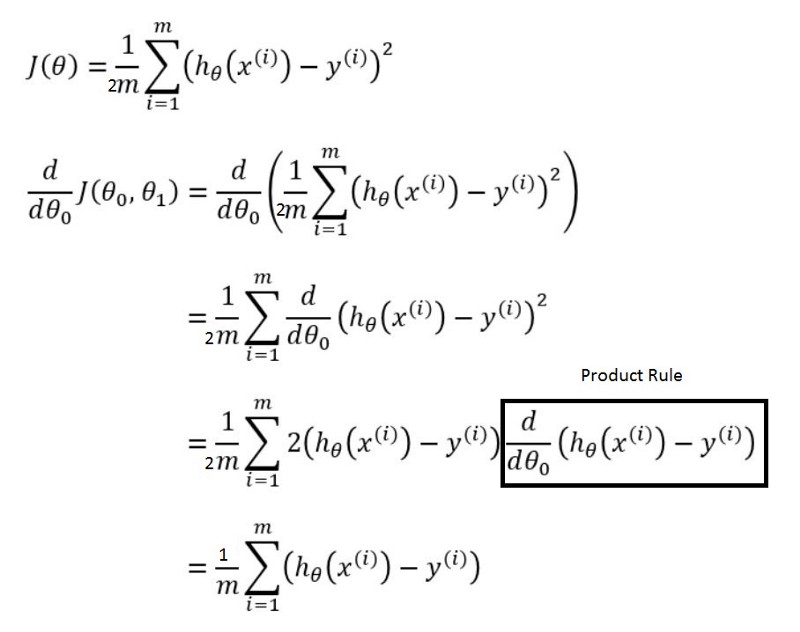


How to calculate the partial derivatives???

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

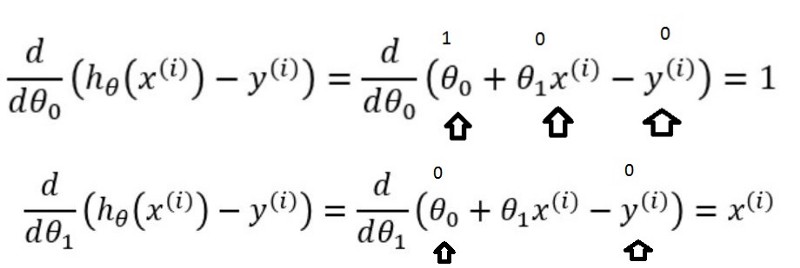


The same thing we can apply for calculating partial derivative with respective to **θ0** and **θ1**.



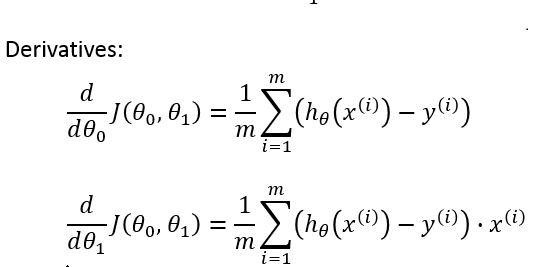
How come that box drawn disappeared in the next step above? Check below

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



These are final **θ0** and **θ1**values

In the same way value for all the weights are calculated using gradient descent for all the input records and then average is taken. This average is subtracted from previous weight wrt learning rate alpha.

This cycle goes on until error is minimized and after that we get the final resultant line or hyper plane that we use for prediction.