Linear Regression and Gradient Descent

Sameer Dhiman

July 16, 2020

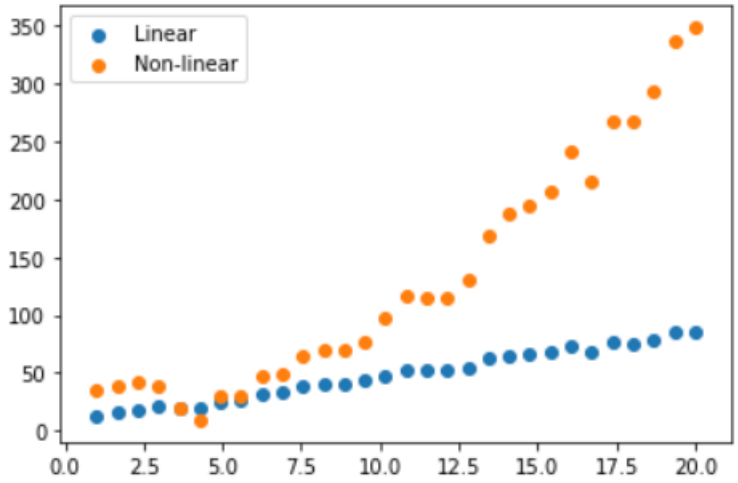
1 Linear Regression

In the previous sessions we saw an introduction to regression. Now it's time to see in it more details.

In linear regression we use the equation

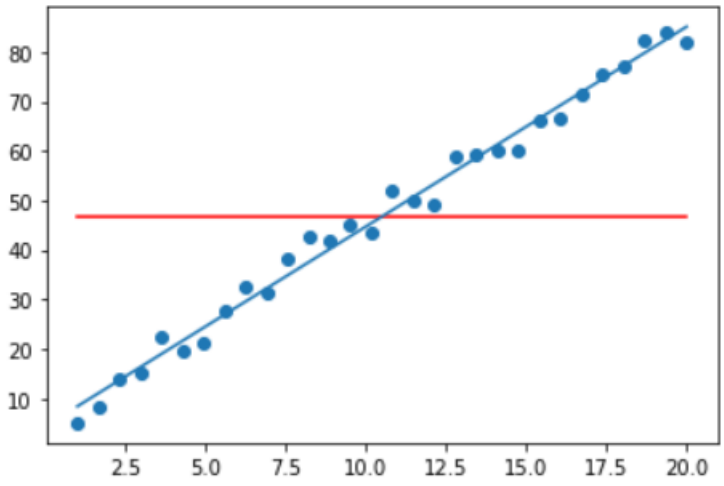
$$f(x) = a * x + b$$

To predict make prediction. This only works if the data also have an linear dependence on the features (Features are the data values depend on for eg. area is a feature for house pricing i.e. Price of house will depend on area). Let's look at two data sets.

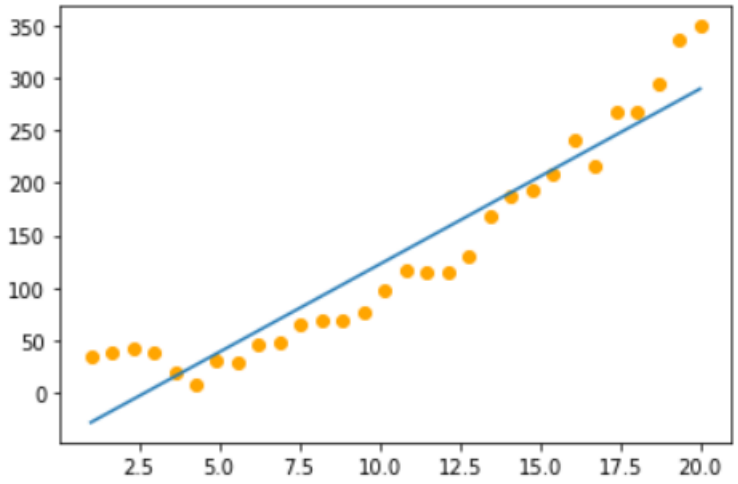


Here Blue one looks to be linearly dependent on x while orange one is not. If we use Linear regression on a data set that is not approximately linearly dependent then we will get large error on the predictions. After running linear regression for both the data sets we get these lines

```
Score =  0.9880399840549782
```




```
Score =  0.8999401506741315
```



As you can see in the blue one line fits much better than in the second one which didn't looked as much linear. Also here score mean R^2 error or coefficient of determination. It is defined as

$$R^2 = 1 - \frac{\sum(y_{true} - y_{pred})^2}{\sum(y_{true} - y_{true.mean})^2}$$

y_{true} is the true value of the prediction, y_{pred} is the value that our model predicted and $y_{true.mean}$ is the mean of all true values. Think of it as comparing our result to the line ($y = y_{true.mean}$). In above example red line is that line.

These were the cases when our predictions depend on only 1 feature. If our data has more than 1 features then nothing will change only thing is a new dimension will be added in the graph. for eg. $y = 10 * x1 + 20 * x2 + 20$.

Now observe that in case of single feature we were getting a straight line, now in case of 2 features we will get a plane. Think about it yourself, try to visualize

it in 3d. After more than 2 features we won't be able to visualize it because graph will have more than 3 dimensions and I have never seen a 4-dimensional graph that I can understand by looking at it.

2 Gradient Descent

Now our main goal was to find the coefficients that minimizes the error. In blue example above we have to find the values of m and b in the equation $mx + b$ so as to be able to plot the line and make predictions.

Last time we found the values using trial and error, this time let's look at a much better way. We have defined the error as

$$MSE = \frac{1}{n}\sum(y_{pred} - y_{true})^2$$

Here n is the total number of data points. Now

$$y_{pred} = m * x + b$$

Here m and b are found by our algorithm and we already know y_{true} from the data that is already given.

$$\implies MSE = \frac{1}{n}\sum(m * x + b - y_{true})^2$$

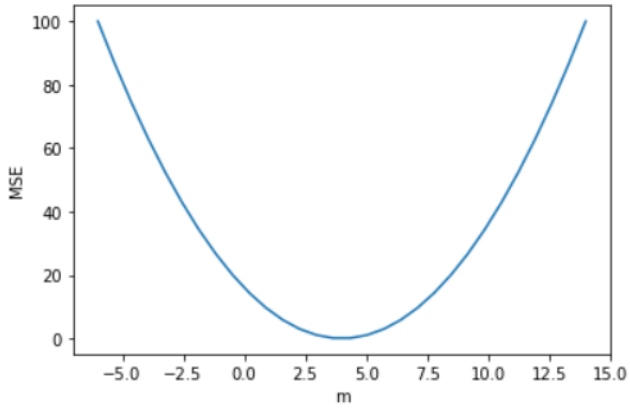
$$\implies \frac{1}{n}\sum(m^2x^2 + b^2 + y_{true}^2 + 2mbx - 2mx * y_{true} - 2 * b * y_{true})$$

Note : Remember that here x and y_{true} values are constant for a given data point. For that data point error depends only on parameters m and b.

It looks like a quadratic equation but in 2 variables. Let's assume b to be constant and plot this error function.

$$MSE = am^2 + bm + c$$

Error will look something like this for different values of m.



Now our task is to minimize error and we can clearly see that error is minimum at local minima ie. where slope is zero. This can be one of the ways

to find the values of m that minimizes MSE but it gets more computationally heavy as number of features increases. Here our prediction depends on only 1 feature if it depends on more than 1 features it will a longer time to find the best values of all the coefficients that will minimize the error. for eg. for $f(x) = ax_1 + bx_2 + cx_3 + d$ we will have to find values of a, b, c, d that best fit the function.

Now lets find a better method. Observe the following in the graph.

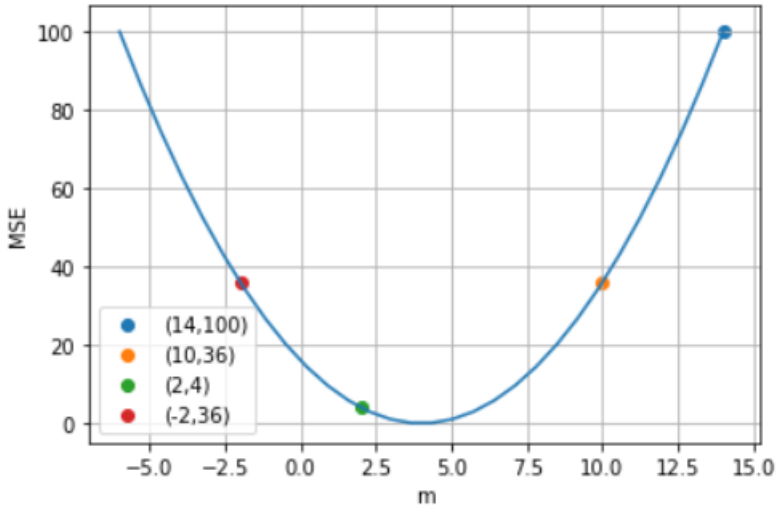
1. First pick a random value of m and find the error. Lets say we picked 14 then error is 100. (Don't think about accuracy of these numerical values, these are only to serve as an example).

2. Now we know by looking at the graph that if we move towards left on x axis error will decrease. ie if we decrease the value of m.

3. Let's subtract 4 from m. Now $m = 14 - 4 = 6$ and error = 36. This looks to be working now what, How long can we keep doing it?

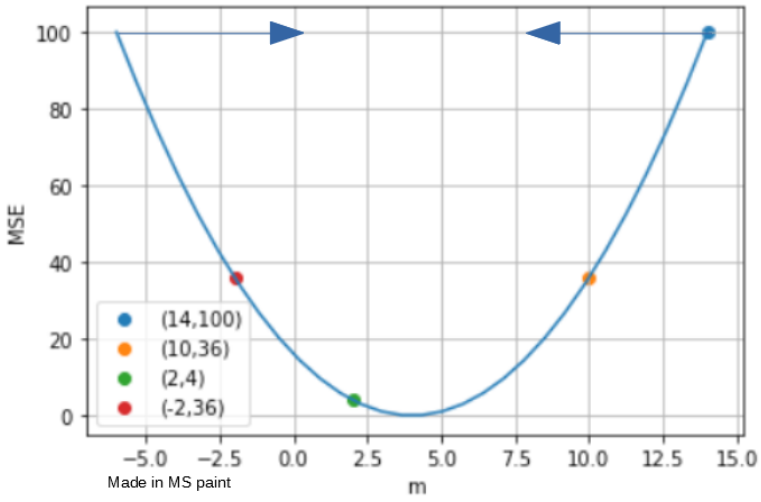
4. We can continue to do this until error starts increasing again which it will after m becomes less than 4. So say if we now if we were to subtract 8 from m then we will completely miss the local minima. since m will be 2 and error = 4. Sure it does look like error has decreased but it's not the minimum values that we want.

5. Then there is the problem of deciding whether to add a value to m or subtract from it. It will depend where local minima lies. for eg. if we picked m = -2 as starting values then we have to add a value to m to decrease the error.



Solution to all these problems is slope of the graph or gradient. Observe that x component of gradient of a function will always face towards decreasing slope ie. towards minima atleast for 2d graphs. We only need the x component of the gradient and we can use it to decide if we have to add a value to m or subtract from it. Also notice that as we get closer to minima the value of magnitude of gradient will decrease and it will be zero

at minima. This means we only need to multiply gradient to the value we want to decrease and it will take care of the rest.



These arrows represent the direction of decreasing slope.

$$m = m - \alpha * \frac{\delta MSE}{\delta m} - (1)$$

Here α will decide how long our steps will be. This equation takes cares of both the problems for us. $\frac{\delta MSE}{\delta m}$ is positive on right of minima and negative on left. This will amount to adding a value to m when we are left of minima and subtracting a value when we are on right of minima.

$$\frac{\delta MSE}{\delta m} = 2 * (m * x + b - y_{true}) * x$$

This will imply that we can write the above eqn (1) as

$$m = m - \alpha * 2 * (m * x + b - y_{true}) * x$$

$$OR$$

$$m = m - \alpha_{new} * (y_{pred} - y_{true}) * x$$

Here $\alpha_{new} = 2 * \alpha$ and $y_{pred} = m * x + b$, similarly for b we can write

$$b = b - \alpha_{new} * (y_{pred} - y_{true})$$

From now on we will refer α_{new} as α only.

***Note*: Notice MSE is defined as mean of all errors squared for each prediction, We skipped the $\frac{1}{n}$ term and also we skipped the $\sum$. We don't need to worry about these, n can be accounted for by choosing a smaller value of α and $\sum$ will be accounted for when we will run this for each prediction.**

What we did today is called Stochastic Gradient descent, It runs the iteration on each prediction independently.

$$m = m - \alpha * \sum(y_{pred} - y_{true})$$

This is called batch Gradient descent, It will first calculate the total error on the entire batch then will change the value of parameter.

Stochastic Gradient Descent provides more accuracy but takes more time than Batch Gradient Descent because we have to run the equation for calculating new value of m every iteration which also changes error every time it runs. There is a mid way solution called Mini-Batch Gradient Descent which makes Batches of sizes smaller than total data set.

I do realize this is going to hard to grasp for some and I am not the best at explaining things properly. If you want we can keep a discussion session and I will try to explain all this in more details to make it more clear.