DATA ANALYTICS

Unit 2:Linear Regression- OLS Vs Gradient Descent

Mamatha.H.R

Department of Computer Science and Engineering

DATA ANALYTICS

Unit 2:Linear Regression -OLS Vs Gradient Descent

Mamatha H R

Department of Computer Science and Engineering

Example:

Years of Experience	Salary in 1000$
2	15
3	28
5	42
13	64
8	50
16	90
11	58
1	8
9	54

Table 1 : Sample Salary Data

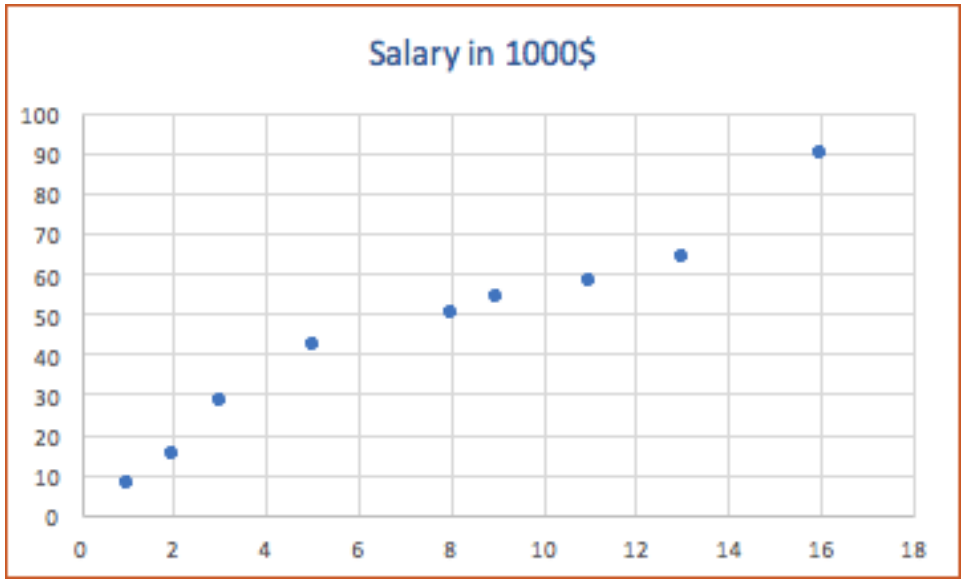


Figure 1: Scatter Plot Diagram

In order to fit the best intercept line between the points in the above scatter plots, we use a metric called "Sum of Squared Errors" (SSE) and compare the lines to find out the best fit by reducing errors. The errors are sum difference between actual value and predicted value.

Sum of Squared Errors (SSE)

To find the errors for each dependent value, we need to use the formula below.

$$SSE = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

y_i = *Dependent Variables (Salary)*

$\bar{y}$ = *Average of Dependent Variables*

Years of Experience	Salary (in 1000$)	Error $y_i - \bar{y}$	Error²
2	15	15 – 45.44 = -30.44	926.59
3	28	28 – 45.44 = -17.44	304.15
5	42	42 – 45.44 = -3.44	11.83
13	64	64 – 45.44 = 18.56	344.47
8	50	50 – 45.44 = 4.56	20.79
16	90	90 – 45.44 = 44.56	1985.59
11	58	58 – 45.44 = 12.56	157.75
1	8	8 – 45.44 = -37.44	1401.75
9	54	54 – 45.44 = 8.56	73.27
	$\bar{y}$ = 45.44		**SSE = 5226.19**

SSE calculations

Sum of Squared Errors (SSE)

- The sum of squared errors SSE output is 5226.19.

- To do the best fit of line intercept, we need to apply a linear regression model to reduce the SSE value at minimum as possible.

- To identify a slope intercept, we use the equation

$$y = mx + b,$$

'm' is the slope
'x' → independent variables
'b' is intercept

Ordinary Least Squares (OLS) Method

We will use Ordinary Least Squares method to find the best line intercept (b) slope (m)

To use OLS method, we apply the below formula to find the equation

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$b = \bar{y} - m * \bar{x}$$

x = independent variables

$\bar{x}$ = average of independent variables

y = dependent variables

$\bar{y}$ = average of dependent variables

Ordinary Least Squares (OLS) Method

We need to calculate slope ‘m’ and line intercept ‘b’.

Years of Experience x_i	Salary (in 1000$) y_i	$(x_i - \bar{x})$	$(y_i - \bar{y})$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
2	15	-5.56	-30.44	169.24	30.91
3	28	-4.56	-17.44	79.53	20.79
5	42	-2.56	-3.44	8.81	6.55
13	64	5.44	18.56	100.97	29.59
8	50	0.44	4.56	2.01	0.19
16	90	8.44	44.56	376.09	71.23
11	58	3.44	12.56	43.21	11.83
1	8	-6.56	-37.44	245.61	43.03
9	54	1.44	8.56	12.33	2.07
$\bar{x}$ =7.56	$\bar{y}$ = 45.44			∑ = 1037.8	∑ = 216.19

OLS method calculations

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$b = \bar{y} - m * \bar{x}$$

m = 1037.8 / 216.19

m = 4.80

b = 45.44 - 4.80 * 7.56 = 9.15

Hence, y = mx + b → 4.80x + 9.15

y = 4.80x + 9.15

Calculate SSE

Let us calculate SSE again by using our output equation.

SSE calculations again after OLS method implementation

Years of Experience x	Salary (in 1000$) y	$\bar{y}$ = mx + b $\bar{y}$ = 4.79x + 9.18	Error $y_i - \bar{y}$	Error²
2	15	18.76	-3.76	14.14
3	28	23.55	4.45	19.80
5	42	33.13	8.87	78.68
13	64	71.45	-7.45	55.50
8	50	47.5	2.5	6.25
16	90	85.82	4.18	17.47
11	58	61.87	-3.87	14.98
1	8	13.97	-5.97	35.64
9	54	52.29	1.71	2.92
				SSE = 245.38

Sum of Squared Error got reduced significantly from 5226.19 to 245.38.

OLS

- Ordinary Least Square method looks simple and computation is easy.
- OLS method will work for both univariate dataset which is single independent variables and single dependent variables and multi-variate dataset containing a single independent variables set and multiple dependent variables sets.
- Ordinary least squares (OLS) is a non-iterative method Ordinary Least Squares solution is the analytical solution and this solution is not scalable.
- Applying this to complex and non-linear algorithms like Support Vector Machine will not be feasible.
- So we will find the numerical approximation of this solution by iterative method — which would be close to (but not exactly equal to) the OLS solution — which gave us the exact solution.

Gradient Descent Algorithm

Gradient descent is one of those “greatest hits” algorithms that can offer a new perspective for solving problems.

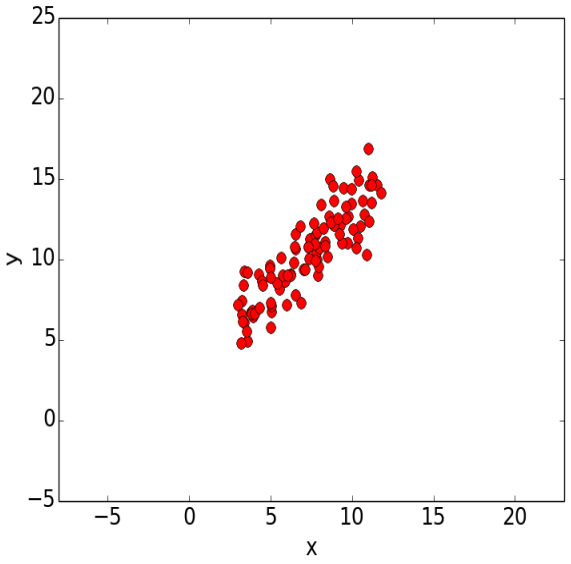
At a theoretical level, gradient descent is an algorithm that minimizes functions.
Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function.

This iterative minimization is achieved using calculus, taking steps in the negative direction of the function gradient.

Gradient Descent Algorithm

Linear Regression Example

Simply stated, the goal of linear regression is to fit a line to a set of points. Consider the following data.



- Let's suppose we want to model with a set of points with a line.
- To do this we'll use the standard y = mx + b line equation where m is the line's slope and b is the line's y-intercept.
- To find the best line for our data, we need to find the best set of slope m and y-intercept values.

DATA ANALYTICS

Gradient Descent Algorithm

Linear Regression Example

A standard approach to solving this type of problem is to define an error function (also called a cost function) that measures how "good" a given line is.

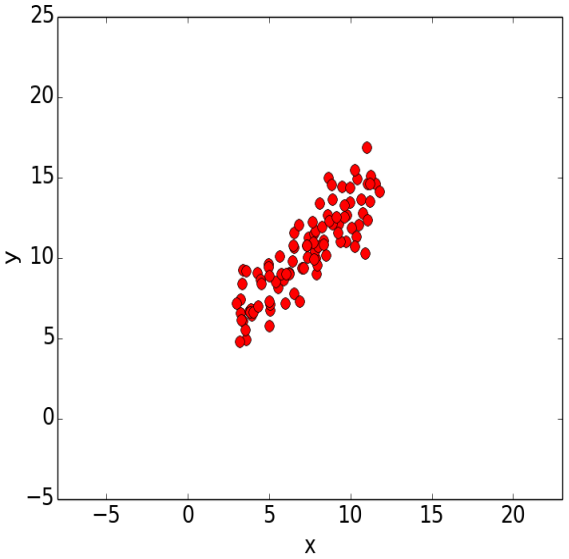
This function will take in a (m,b) pair and return an error value based on how well the line fits our data.

To compute this error for a given line, we'll iterate through each (x,y) point in our data set and sum the square distances between each point's y value and the candidate line's y value (computed at mx + b).

It's conventional to square this distance to ensure that it is positive and to make our error function differentiable. In python, computing the error for a given line will look like:

Formally, this error function looks like:

$$\text{Error}_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$



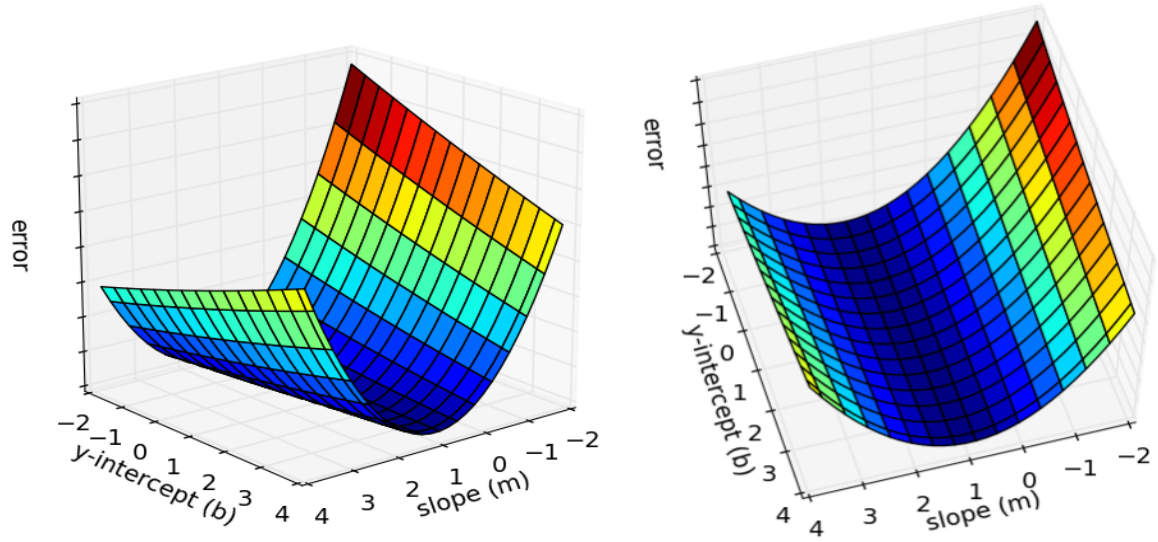
DATA ANALYTICS

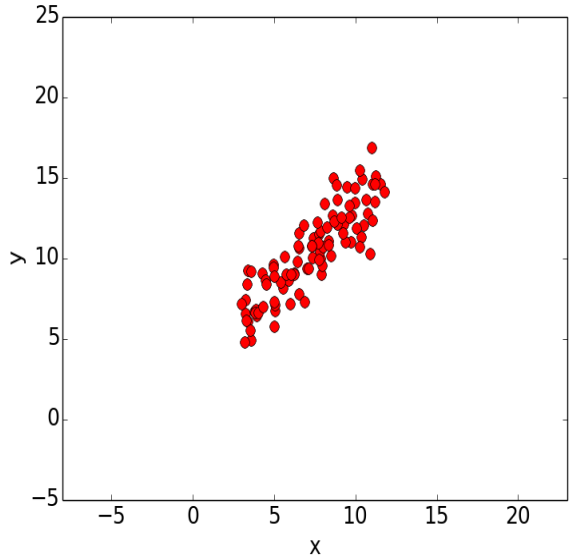
Gradient Descent Algorithm

Formally, this error function looks like: $$\text{Error}_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

Lines that fit our data better (where better is defined by our error function) will result in lower error values.

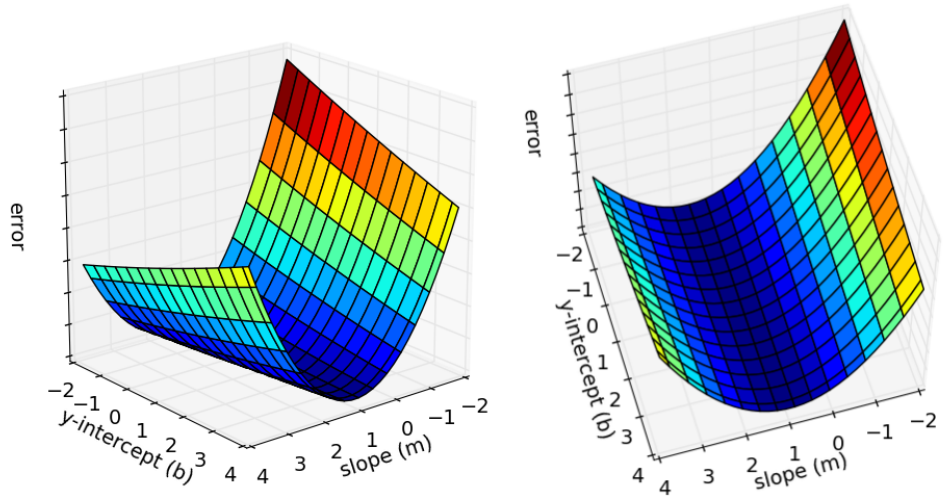
If we minimize this function, we will get the best line for our data. Since our error function consists of two parameters (m and b) we can visualize it as a two-dimensional surface. This is what it looks like for our data set:





Gradient Descent Algorithm

Formally, this error function looks like: $$\text{Error}_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$



Each point in this two-dimensional space represents a line.
The height of the function at each point is the error value for that line.
You can see that some lines yield smaller error values than others (i.e., fit our data better).
When we run gradient descent search, we will start from some location on this surface and move downhill to find the line with the lowest error.

Gradient Descent Algorithm

Analogy:

Imagine a valley and a person with no sense of direction who wants to get to the bottom of the valley. He goes down the slope and takes large steps when the slope is steep and small steps when the slope is less steep. He decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal.

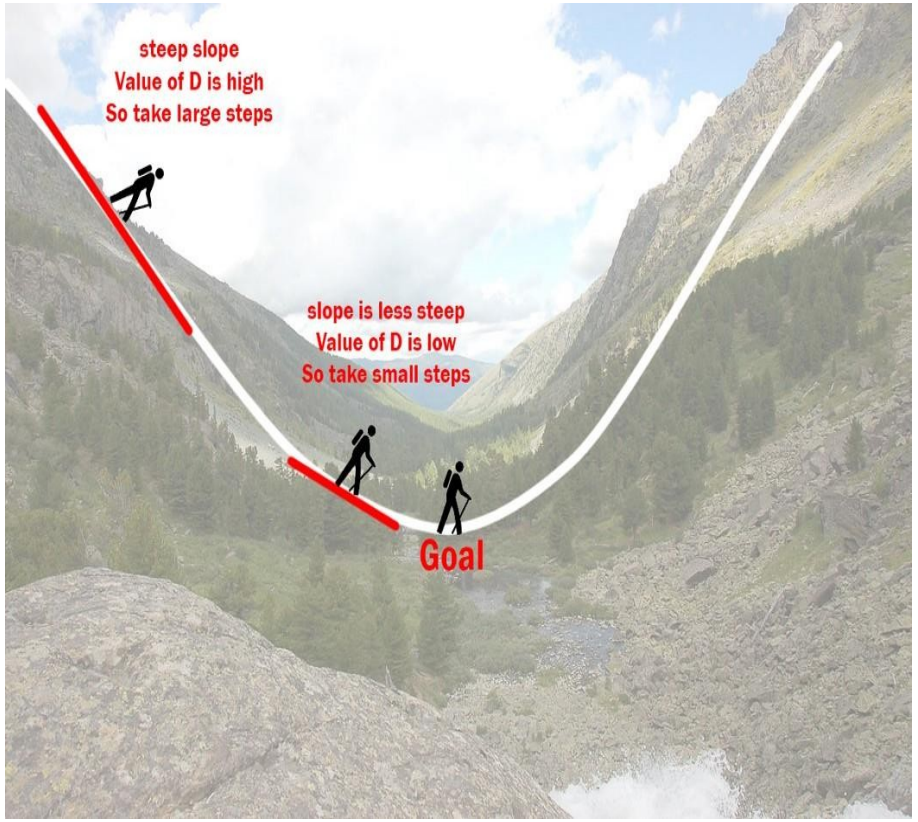


Illustration of how the gradient descent algorithm works

Gradient Descent Algorithm

Let's try applying gradient descent to m and c and approach it step by step:
Initially let m = 0 and c = 0. Let L be our learning rate. This controls how much the value of m changes with each step. L could be a small value like 0.0001 for good accuracy.

Calculate the partial derivative of the loss function with respect to m, and plug in the current values of x, y, m and c in it to obtain the derivative value D.

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

D_m is the value of the partial derivative with respect to m.

Gradient Descent Algorithm

Find the partial derivative with respect to c, Dc :

$$D_c = \frac{-2}{n} \sum_{i=0}^{n} (y_i - \bar{y}_i)$$

Now we update the current value of m and c using the following equation:

$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

We repeat this process until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of m and c that we are left with now will be the optimum values.

Gradient Descent Algorithm

In our analogy,

- m can be considered the current position of the person.
- D is equivalent to the steepness of the slope
- L can be the speed with which he moves.
- Now the new value of m that we calculate using the above equation will be his next position, and L×D will be the size of the steps he will take.
- When the slope is more steep (D is more) he takes longer steps and when it is less steep (D is less), he takes smaller steps.
- Finally he arrives at the bottom of the valley which corresponds to our loss = 0.

Exercise

- Explore more on the Gradient Descent Algorithm- Derivation part

- List the various linear and non-linear algorithms where the Gradient Descent is used.

- Implement both OLS and Gradient descent on a dataset of your choice and list the value of SSE in both the cases.

References

https://towardsdatascience.com/linear-regression-simplified-ordinary-least-square-vs-gradient-descent-48145de2cf76

https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

https://medium.com/meta-design-ideas/linear-regression-by-using-gradient-descent-algorithm-your-first-step-towards-machine-learning

THANK YOU

Dr.Mamatha H R

Professor,Department of Computer Science

mamathahr@pes.edu

+91 80 2672 1983 Extn 834