Mithilesh singh

Linear regression and gradient descent

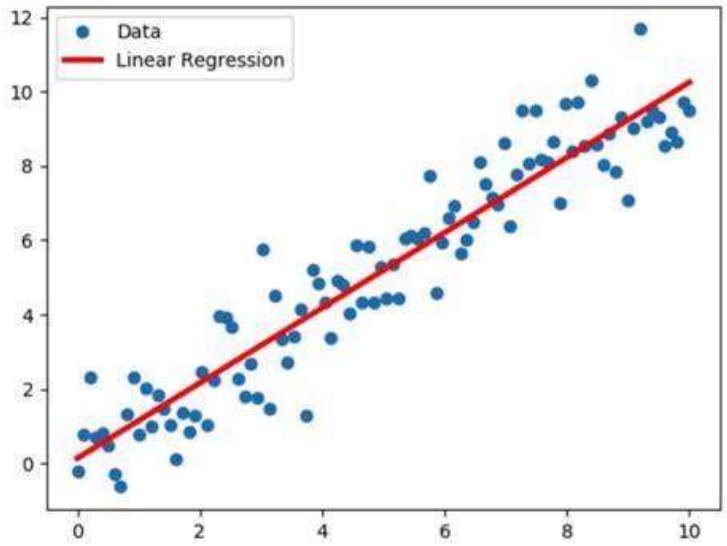
Linear regression is about finding the line of best fit for a dataset. This line can then be used to make predictions.

Statistics way of computing line of best fit:

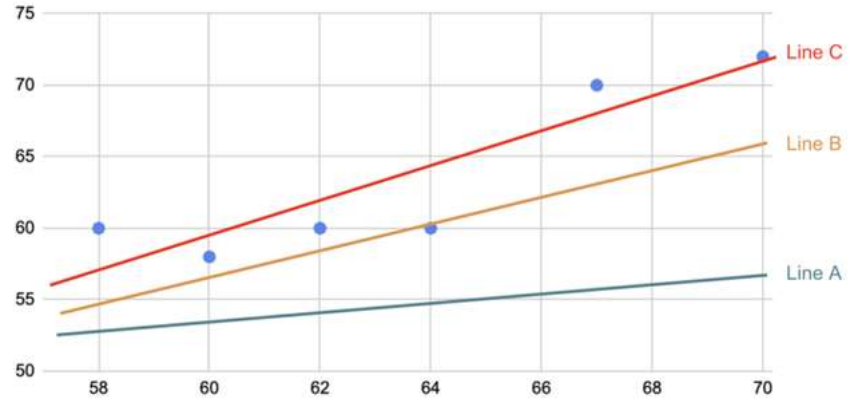
A line can be represented by the formula:

`y = mx + b`.

```
prediction function (y) = (m * x) + b
Here, x is the independent variable
      y is the dependent variable
      m is the weight associated with input variable
      b is the bias or intercept value
```



How do arrived at the line of best fit?

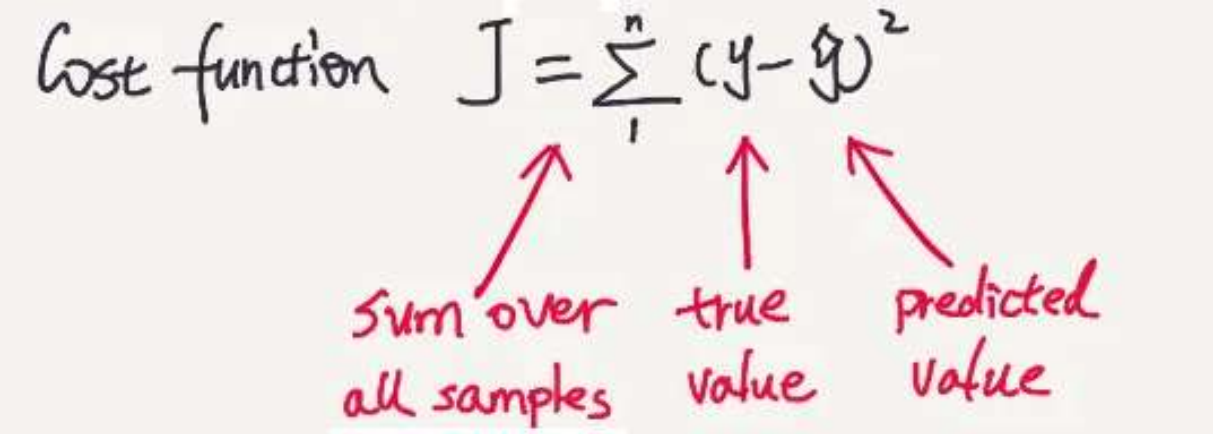


Error is our actual value minus your predicted value. The line of best fit minimizes the sum of the squares of all the errors.

For every line we try — for example - line A, line B, line C,... etc — we calculate the sum of squares of the errors. If line B has a smaller value among all of the lines, then line B is a best fit line.

stochastic gradient descent

In machine learning terminology, the sum of squared error is called the "cost". This cost equation is:



This equation is therefore roughly “sum of squared errors” as it computes the sum of predicted value minus actual value squared.

In gradient descent, we start with a random line. Then we change the parameters of the line (i.e. slope and y-intercept) little by little to arrive at the line of best fit.

we start with a random value of m and b ,Then, we adjust slope m and y-intercept b. Compute the sum of squared errors .

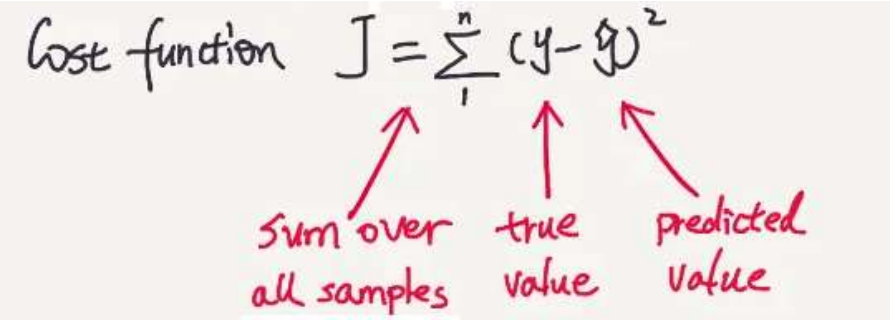
Continue adjusting until we reach a local minimum, where the sum of squared errors is the smallest and additional tweaks does not produce better result.

The basic idea of gradient descent is to start with an **initial set of weights and update them in the direction of the negative gradient of the loss function**.

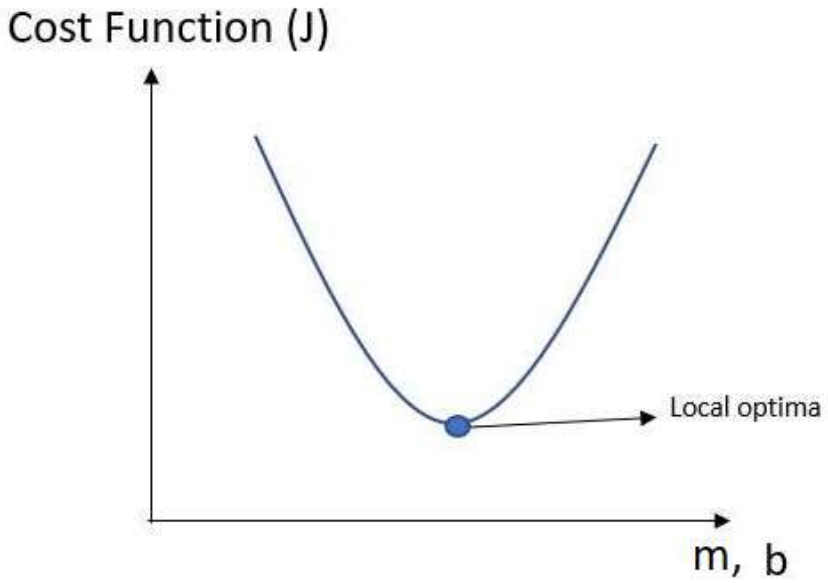
Rate of change of m and b decides the rate of change of lost function.

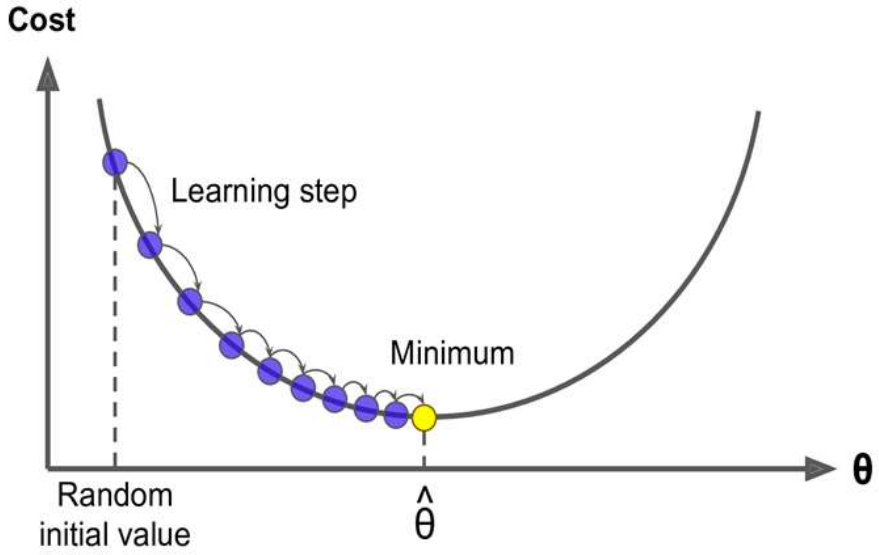
It is the partial derivatives that represents the rate of change of the loss function with respect to the weights.

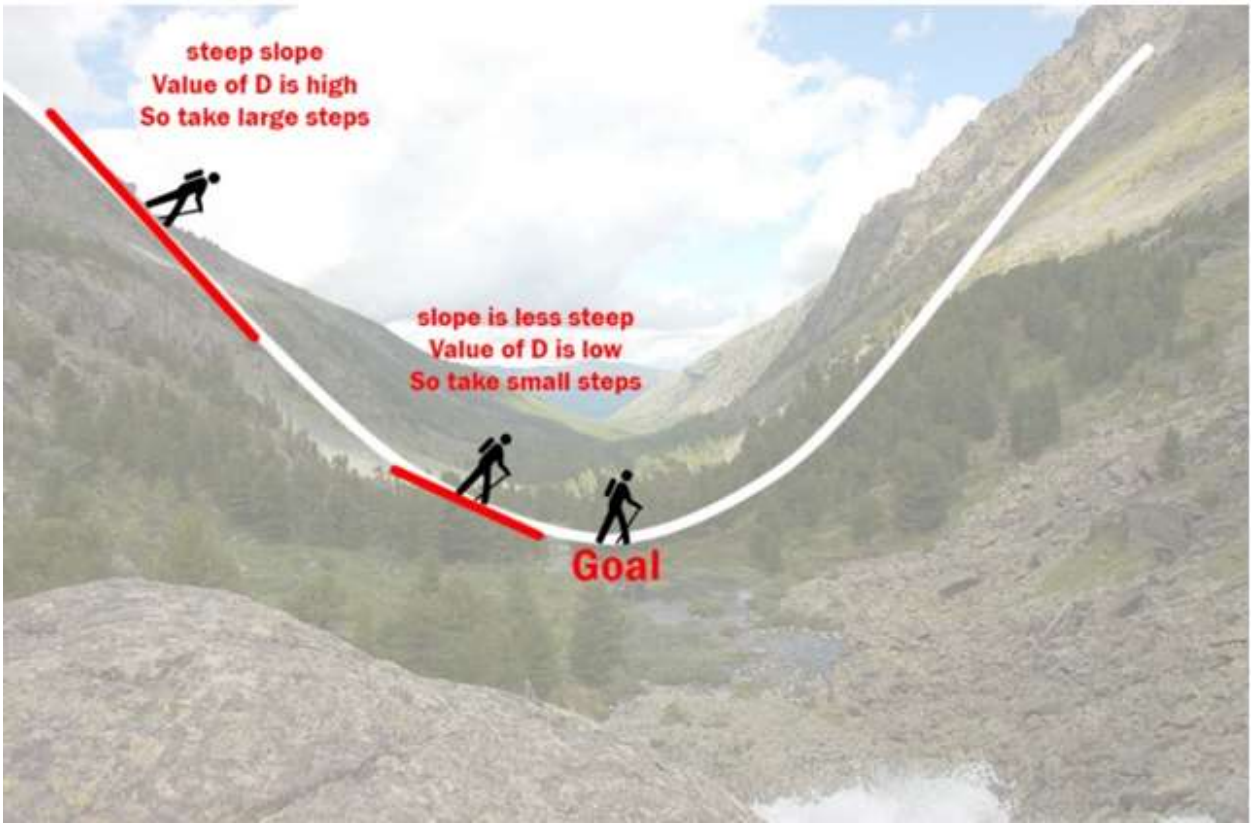
Gradient descent finds local minima **through multiple iterations.**



Goal is to minimize the loss or the loss function
Here the loss or cost function, it is the function of X
Cost value decide by the function of (X square)
If we plot a graph of X square on X-Y plane it will be U shape as below: -







Function J(m,b)= (Error) square

Mathematics of Gradient Descent Algorithm

We change their values according to the gradient descent formula, which comes from taking the partial derivative of the cost function.

```
Updated new b =old b-(learning rate * (dJ/db))
Updated new w = old w - (learning rate * (dJ/dw))
```

By taking the partial derivative, we arrive at the formula:

b new_intercept = old_intercept - $\alpha \cdot (\frac{1}{n}) \sum_{i}^{n} (y - \hat{y})^2$

m new_slope = old_slope - $\alpha \cdot (\frac{1}{n}) \sum_{i}^{n} (y - \hat{y})^2 \cdot x^{(i)}$

This formula computes by how much we change our y theta with each iteration.

The alpha (α) is the learning rate.

The learning rate determines how big the step would be on each iteration. It's critical to have a good learning rate because if it's too large the algorithm will not arrive at the minimum, and if it's too small, the algorithm will take forever to get there.

For example, here we pick the alpha to be 0.001 ,*(to determine the step size while moving towards local minima)*

Example: Size of House and price prediction: -

House size & price Data:

House Size sq.ft (X)	House Price (Y)
2100	199,000
2400	219,000
2425	245,000
2550	255,000
2600	279,000
2700	308,000
2700	312,000
2875	319,000
3350	324,000
3450	405,000

1.Scale the data using standard scaler or Min max scaler.

Normalized data.

House Size sq.ft (X new)	House Price (Y new)
0	0
0.222222222	0.097087379
0.240740741	0.223300971
0.333333333	0.27184466
0.37037037	0.388349515
0.444444444	0.529126214
0.444444444	0.548543689
0.574074074	0.582524272
0.925925926	0.606796117

Predicted value: -

$\hat{y} = b + m*X_{new}$

2. Let's start by initializing the random weights. a= 0.45, m=0.75

Random weight	
b	m
0.45	0.75

3. Compute the ŷ

Consider the random observation from the new scaled data

X new= 0.222222222 ,

Y new= 0.097087379

ŷ = b + m*X new

ŷ = 0.45 + 0.75*0.222222222

=0.61666

4. Compute loss

Residual or Error = ŷ- y new

= 0.6166666–0.0970873

= 0.519579

Objective is to Minimize the error, with the update the best effective value of b & m.

5. Compute partial differentials

Calculate the partial derivative of the total error with respect to weight

The partial differentials are as follows:

The partial derivative of total error w.r.t. b (weight)

= ∂E/∂b

= ∂(ŷ- y new)/∂b

= ∂(ŷ- (b+m*X new))/∂b

= -(ŷ-y new)

=-0.519579

The partial derivative of total error w.r.t. m (weight)
= ∂E/∂m
= ∂(ŷ- y new)/∂m
= ∂(ŷ- (b+m*X new))/∂m
=-(ŷ-y new)*X new
=-0.519579*0.222222222
= -0.115462064

6. Update the weights

Update ‘b’
Use a small learning rate (alpha=0.01)
New b = old b — alpha* ∂E/∂b
= 0.45–0.01 * — 0.519579
= 0.455196
Update ‘m’
New m = old m — alpha*∂E/∂m
= 0.75–0.01 * -0.115462064
= 0.75115
We have just finished our first iteration of stochastic gradient descent and we have updated our weights to be **b = 0.455196 and m= 0.455196.**
Repeat this process for remaining instances from our dataset.

7. Iteration: 5

Below is a list of all the updated values for the coefficients over the 5 iterations.

	House Size (X new)	House Price (Y new)	Random weight b	m	ŷ= b + m *X new	Error ŷ-y new
			Updated Weight			
Iteration-1	0.222222222	0.097087379	0.45	0.75	0.616666667	0.519579
Iteration-2	0.240740741	0.223300971	0.455196	0.751155	0.636029313	0.412728
Iteration-3	0.333333333	0.27184466	0.459323	0.752148	0.710039152	0.438194
Iteration-4	0.37037037	0.388349515	0.463705	0.753609	0.742819419	0.35447
Iteration-5	0.444444444	0.529126214	0.46725	0.754922	0.802770487	0.273644
	0.444444444	0.548543689				
	0.574074074	0.582524272				
	0.925925926	0.606796117				