Linear Regression from scratch

ACM AI SIG: Srinivasa Perisetla

Little bit About me...

Srinivasa Perisetla

2nd Year CSE Major

Hobbies: Basketball, Lifting Weights, Coding

Intro To Linear Regression

Say we have a Graph representing:

X: # of years of Experience

Y: Amount of Salary $

How would a human draw a best fitted line?

How would a human draw a best fitted line?

Best Fit Line

This Line can be represented as

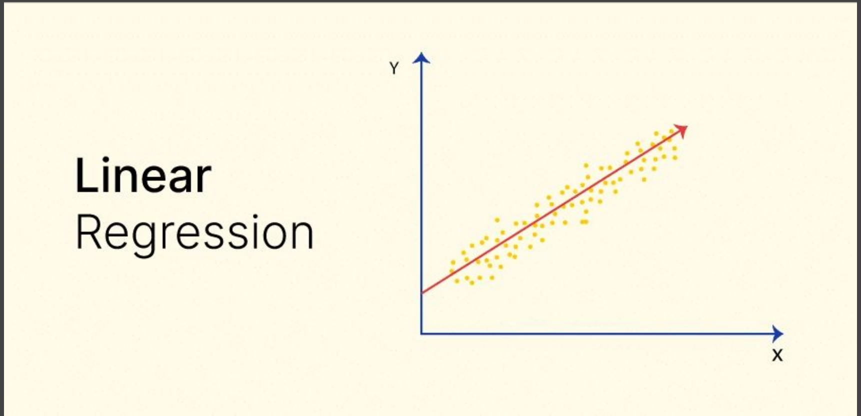
$$Y = mX + b$$

This can be used to predict the salary based off the # years of Experience

How can a computer come up with the best fit line

Y = mX + b

- If we can tune the parameters m and b to reduce the amount of error (difference) between the line and the actual point



How can we measure Error

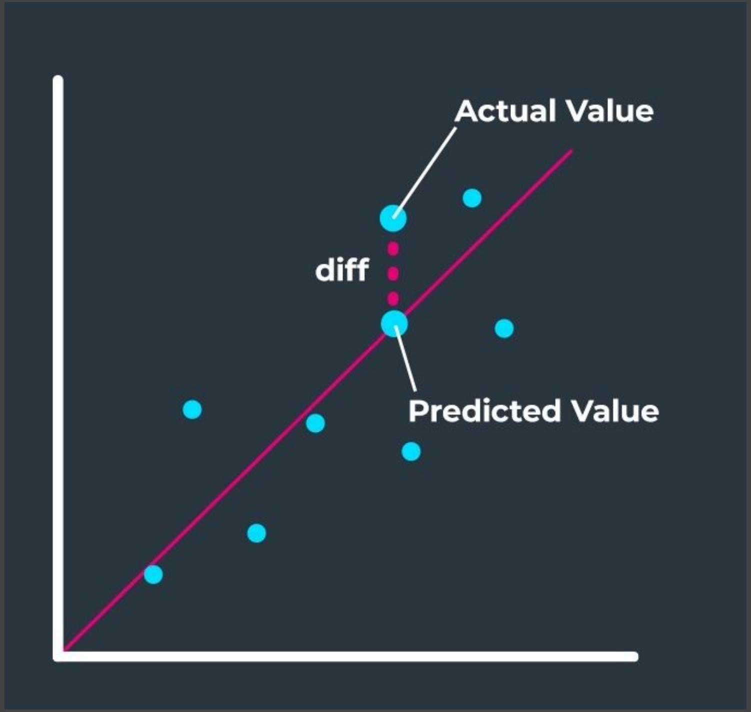
Goal: find m and b in the equation which gives the least amount of error

Error: $(y_i - \hat{y}_i)$

yi = Actual value

y^i = Predicted value from the line

Essentially the difference in distance between the points and the line



Measuring total error

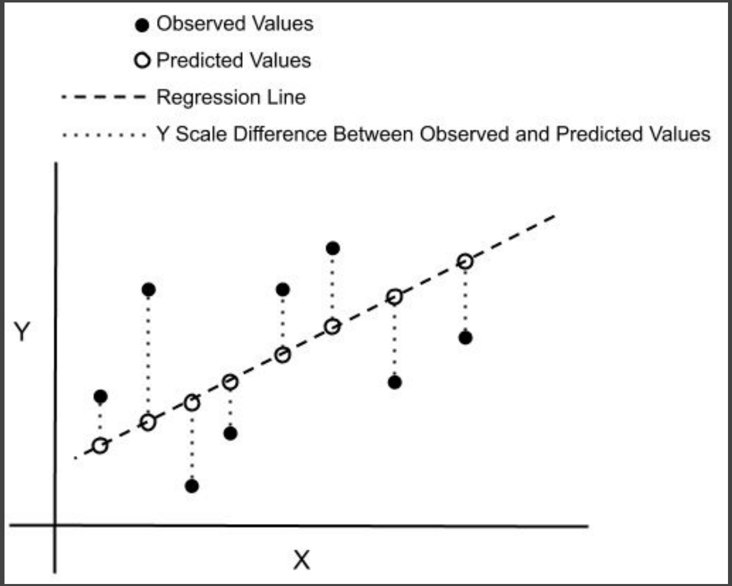
<u>Mean Squared Error</u>

Square the differences:
- Penalize Larger Errors
- Get rid of Negative Error

Sum all of the errors up and take average

This gives the total error!

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Measuring total error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

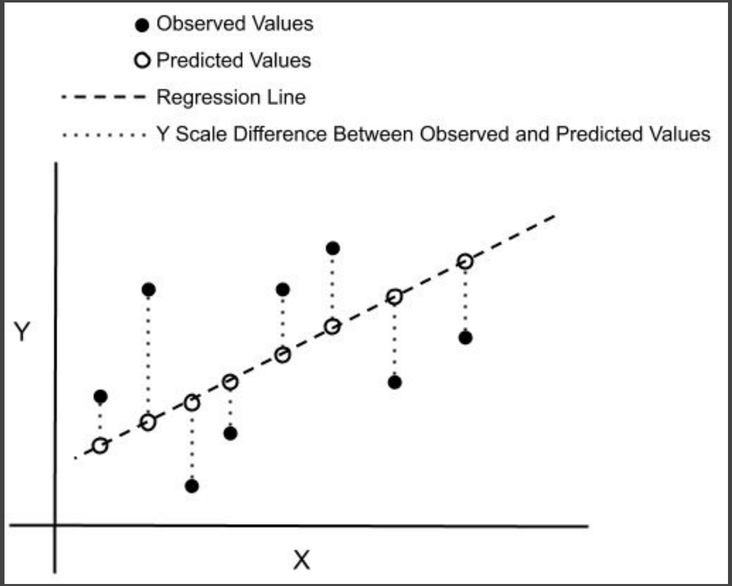
<u>Mean Squared Error</u>

Square the differences:

- Penalize Larger Errors
- Get rid of Negative Error

Sum all of the errors up and take average

This gives the total error!



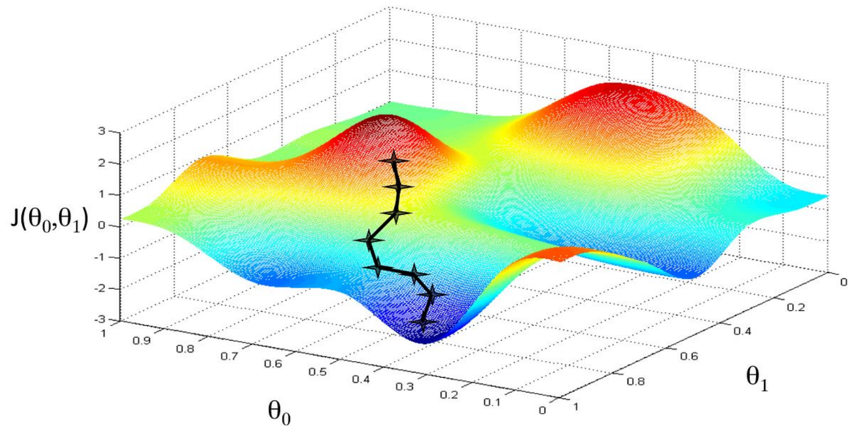
How can we reduce the error?

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Y = mX + b

We have total error (MSE)

Perform Gradient Descent Algorithm



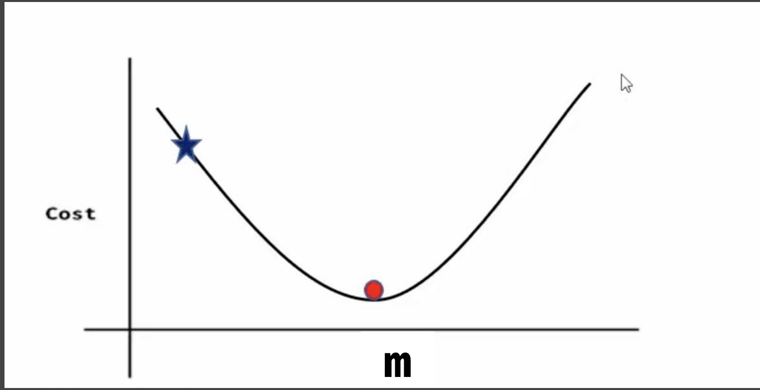
Gradient Descent Algorithm

Our total Error can be graphed as a Function vs m and b (3 dimensions)

Let’s look at Total Error vs m for simplicity (2 Dimensions)

We want to traverse to the global minimum

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Gradient Descent Algorithm

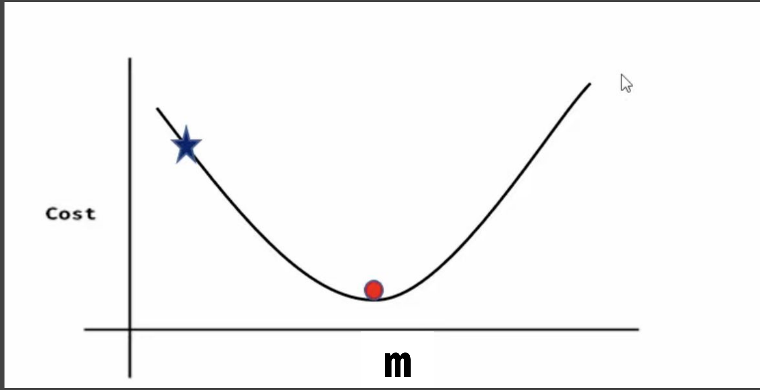
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Find the Derivative ∂E/∂m

- Derivative gives the maximum change in Error per change in variable (steepest ascent)

We need steepest descent (−∂E/∂m)

Then change the current m with the Derivative



Calculating Derivative ∂E/∂m

$$\hat{y} = mx + b$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Calculating Derivative ∂E/∂m

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - mx_i - b)^2$$

$$\frac{\partial E}{\partial m} = \frac{1}{n}\sum_{i=1}^{n} 2\,(y_i - mx_i - b)\cdot(-x_i)$$

Final Calculated Derivatives ∂E/∂m and ∂E/∂b

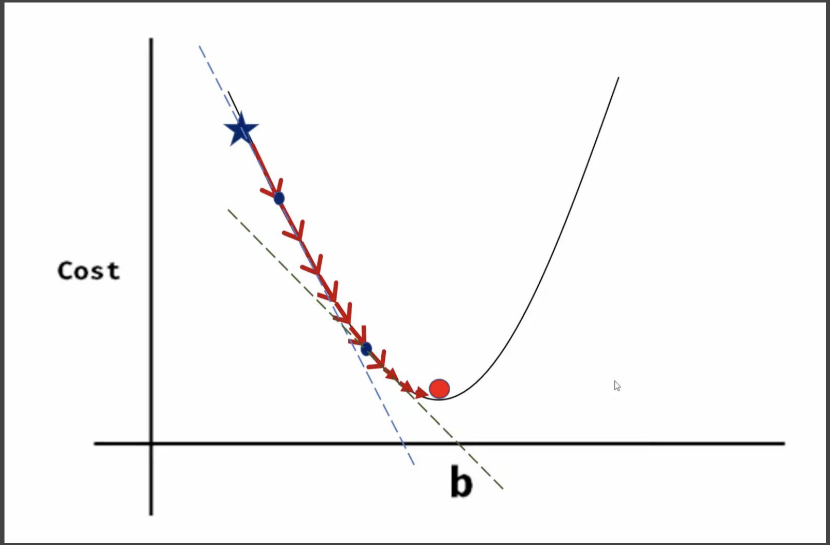
$$\frac{\partial E}{\partial m} = -\frac{2}{n}\sum_{i=1}^{n} x_i\,(y_i - mx_i - b)$$

$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - mx_i - b)$$

Backpropogation

m = m - learning_rate * ∂E/∂m
b = b - learning_rate * ∂E/∂b
Learning_rate = 0.001

This is so the algorithm takes small steps towards the minimum

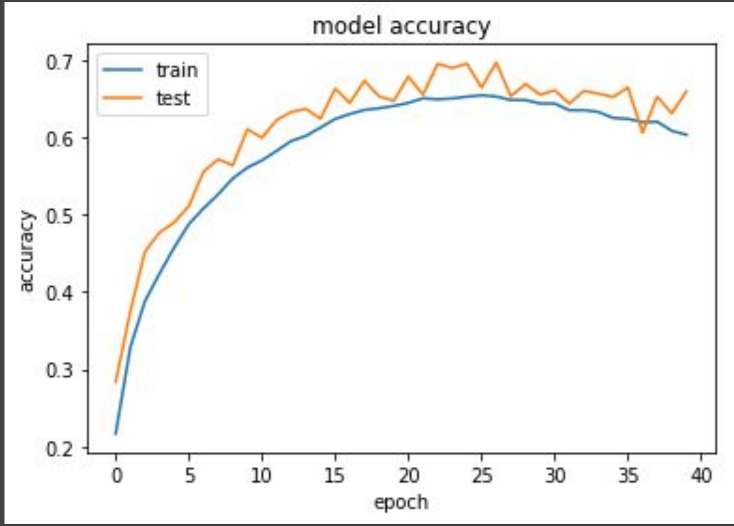


Training / Epochs

Train model for 100 Epochs

Epoch - 1 complete pass through entire training data set

Multiple Epochs are needed for effective training



Finished Model

Y = mX + b

Found values of m and b that reduces the Error (distance between points and line)

Lets Code Linear Regression from Scratch!

Scan the QR Code for Github Link and follow along!