CODE HEROKU

Please turn off your webcam

If you are joining from a mobile phone
be sure to click on
Join via Device Audio

We are waiting for other participants to join
We will begin at 5:30 PM IST

# Gradient Descent For Machine Learning

Mihir Thakkar

Founder and Instructor
hello@codeheroku.com

CODE HEROKU

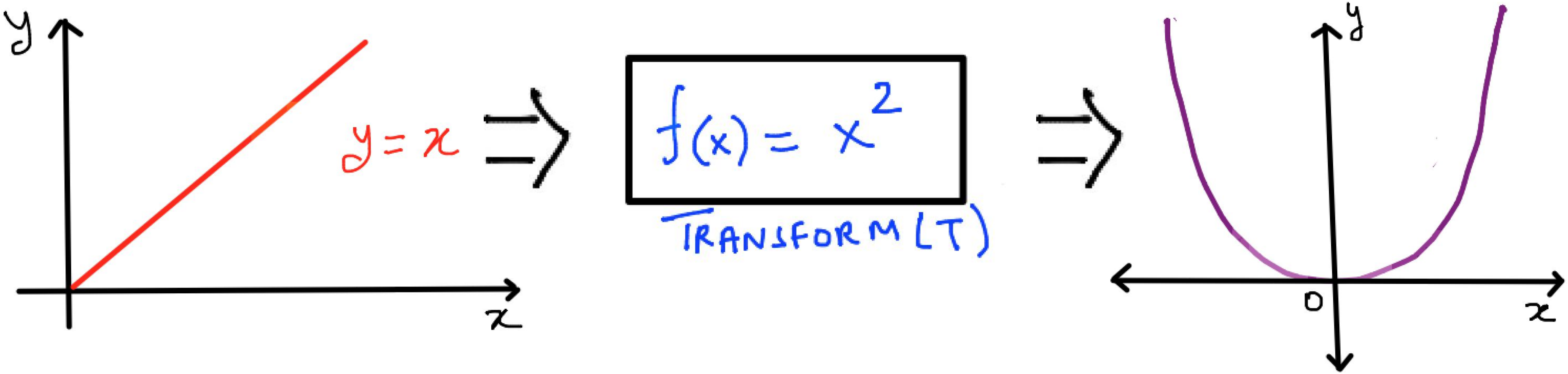
# SESSION OBJECTIVES

- Quick Recap
- Why do we need GD?
- Revise Some Math
- Implement in Python

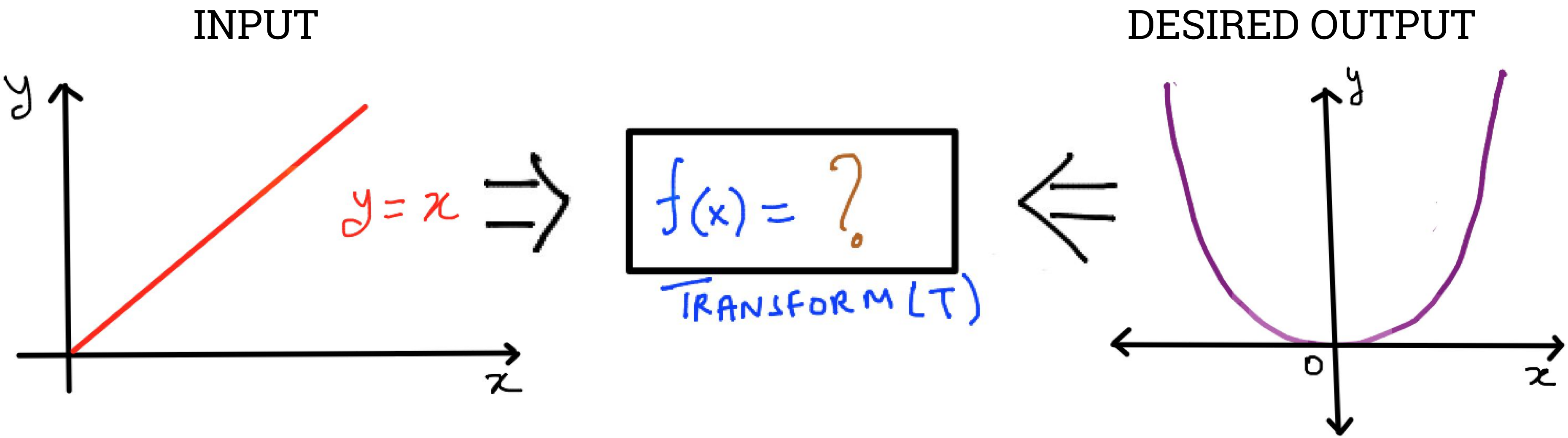
# Highschool Math

INPUT

OUTPUT



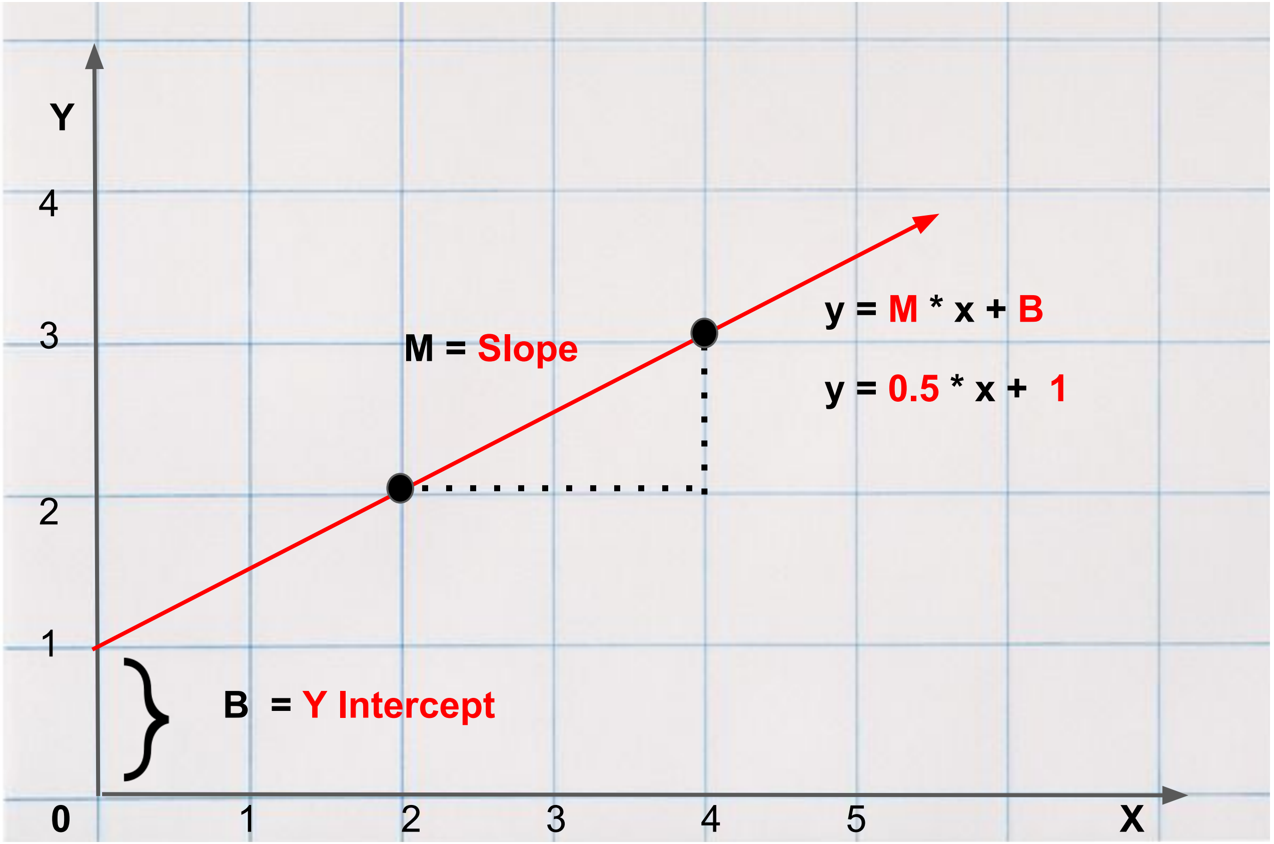
# Machine Learning



INPUT

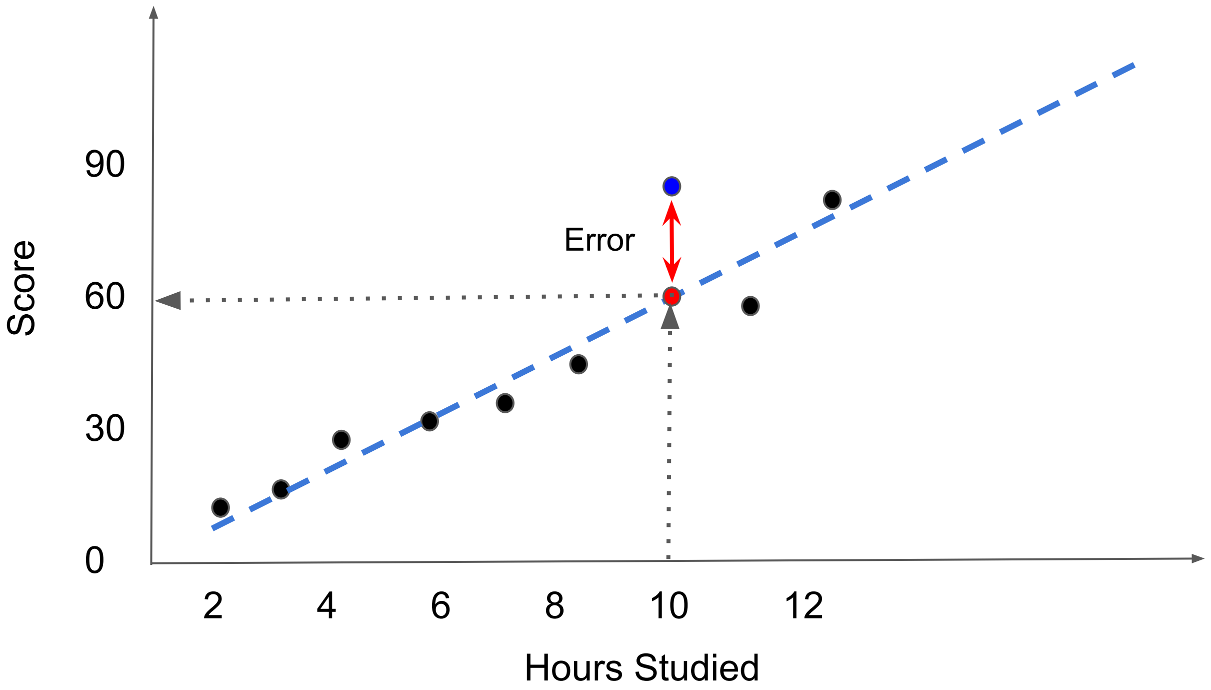
DESIRED OUTPUT

# Let’s Revisit Some Basics



# Linear Regression

E.g. Predict the score of a student based on number of hours studied



① Create a Model

**y = M * x + B**

Predicted Score for
10 hours studied = **60**

# QUIZ

Calculate the Total Error, Mean Error and Mean Squared Error for the following set of predicted and actual results

| Predicted Score | Actual Score |
|---|---|
| 30 | 32 |
| 25 | 25 |
| 22 | 20 |
| 20 | 18 |

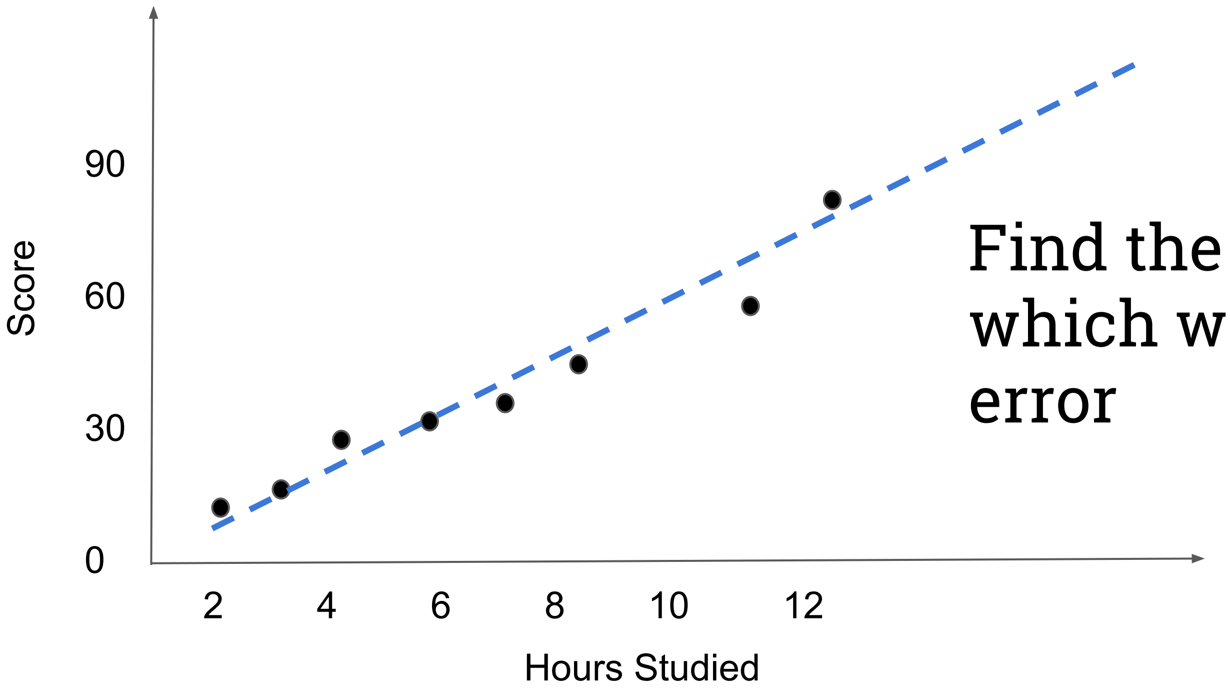
Total Error
= (32 - 30) + (25-25) + (22-20) + (20-18)
= 2 + 0 + 2 + 2
= 6

Mean Error
= 6/4
= 1.5

Mean Squared Error

$$= \frac{(30 - 32)^2 + (25 - 25)^2 + (22-20)^2 + (20-18)^2}{4}$$

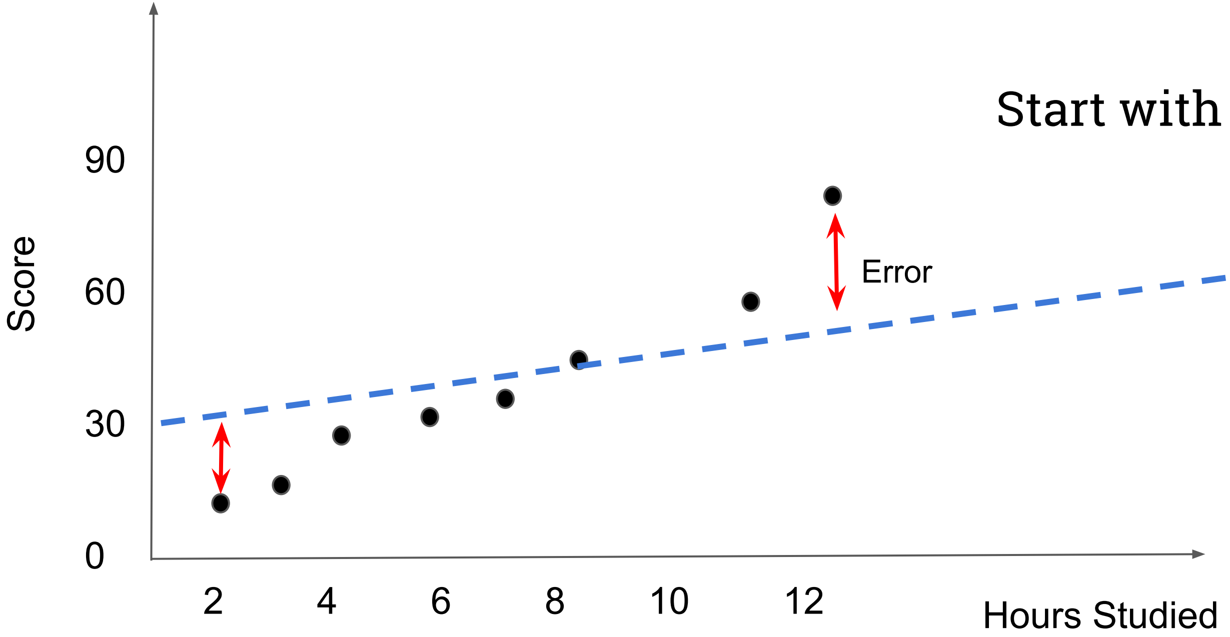
# Our Objective



Find the line (i.e. m & b) which will result in least error

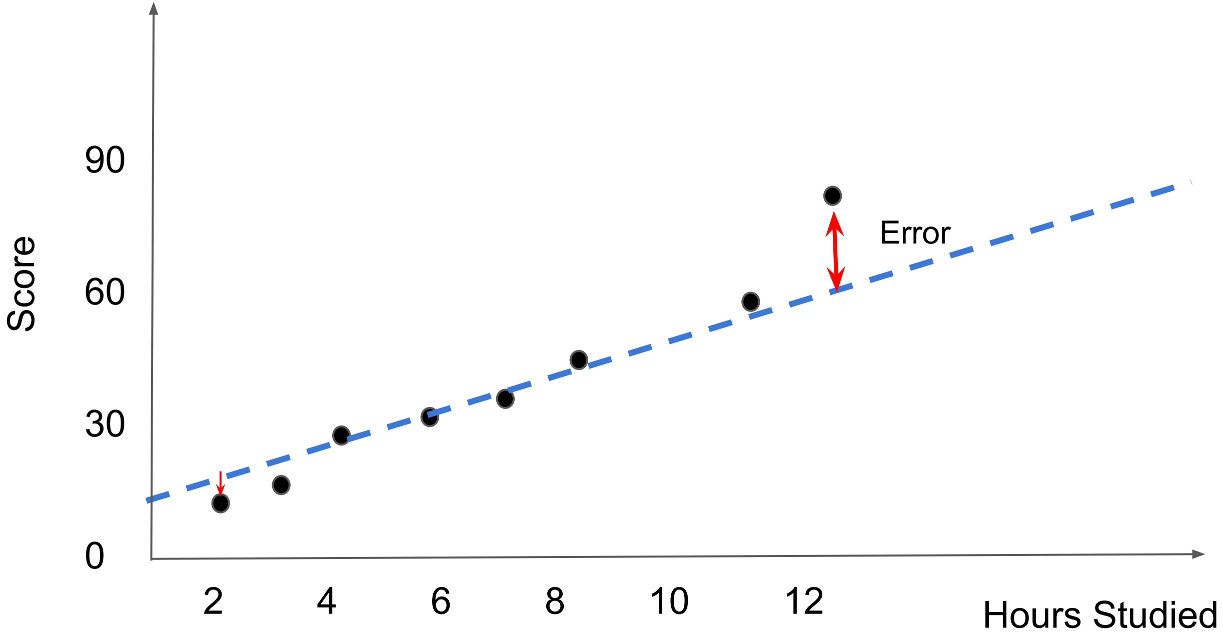
# Solution

# Step 1: Start with a random line

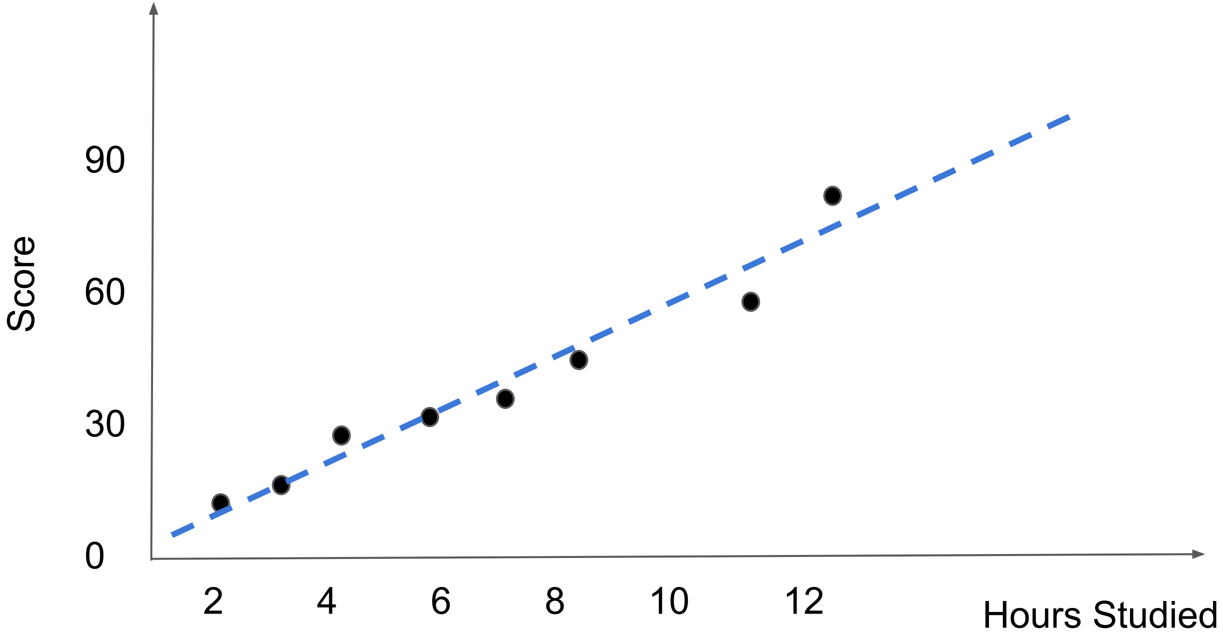


Start with random values of m & b

# Step 2 : Adjust m & b such that error reduces



# Step 3 : Repeat until converge to best approximation



# QUIZ

Which of the following statement is TRUE?

1. Gradient Descent can be applied only to Regression problems
2. Gradient Descent is an optimization algorithm which can be applied to any problem in general

# Let’s Build It

http://www.codeheroku.com/static/workshop/datasets/gd.zip

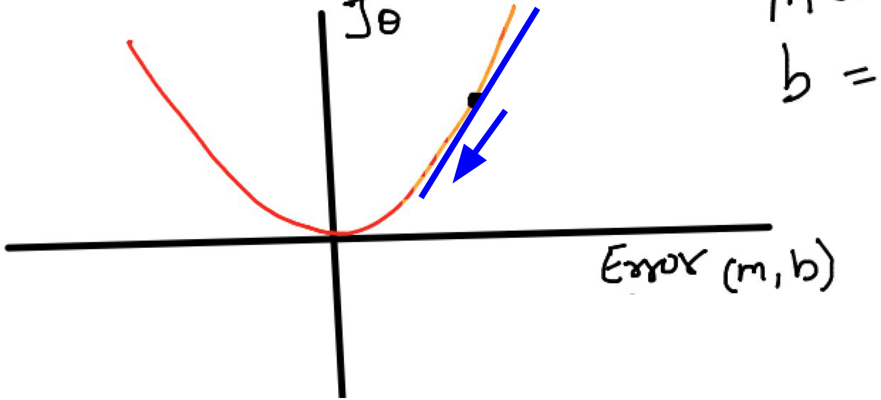
$$MSE = \frac{1}{N} \sum \left( \underbrace{\text{Estimate} - \text{Actual}}_{mx+b-y_i} \right)^2$$

$\Downarrow$

Cost function

$\Downarrow$

$$J_{\theta(m,b)} = \frac{1}{N} \sum \text{Error}^2$$



$$m = m + \Delta m$$

$$b = b + \Delta b$$

$$\Delta m = \frac{\partial}{\partial m} J_{(m,b)}$$

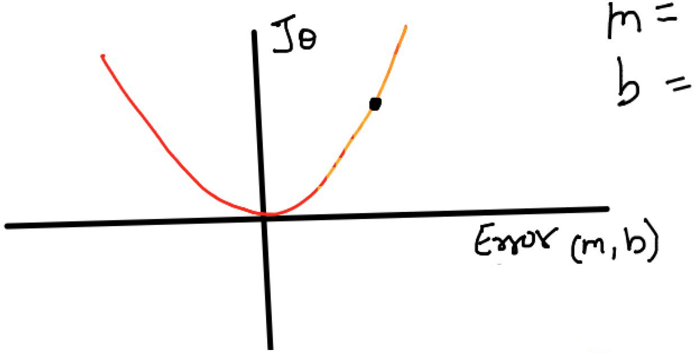
$$= \frac{1}{N} \frac{\partial}{\partial m} \text{Error}^2$$

$$= \frac{2}{N} \cdot \text{Error} \cdot \frac{\partial}{\partial m} (mx+b-y)$$

$$= \underline{\underline{\frac{2}{N} \cdot \text{Error} \cdot x}}$$

$m = m + \Delta m$

$b = b + \Delta b$



$$\frac{d}{dx} x^n = n x^{n-1}$$

## The Chain Rule

$f = f(g)\ ; \quad g = g(x)$

$$\frac{df}{dx} = \frac{df}{dg} \frac{dg}{dx}$$

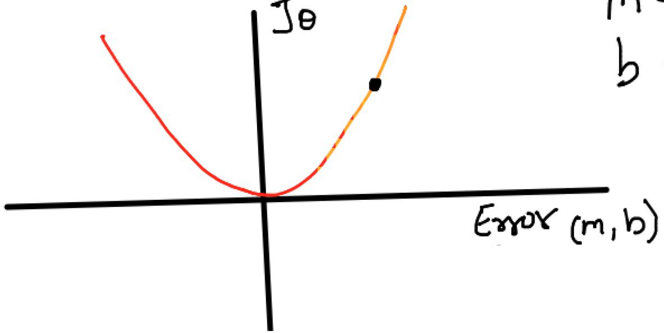
$$\Delta b = \frac{\partial J}{\partial b}$$

$$= \frac{1}{N} \cdot \frac{\partial}{\partial b} \text{Error}^2$$

$$= \frac{2}{N} \cdot \text{Error} \cdot \frac{\partial}{\partial b} \underbrace{mx + b - y}$$

↳ everything else is constant

$$\Delta b = \underline{\underline{\frac{2}{N} \cdot \text{Error} \cdot 1}}$$



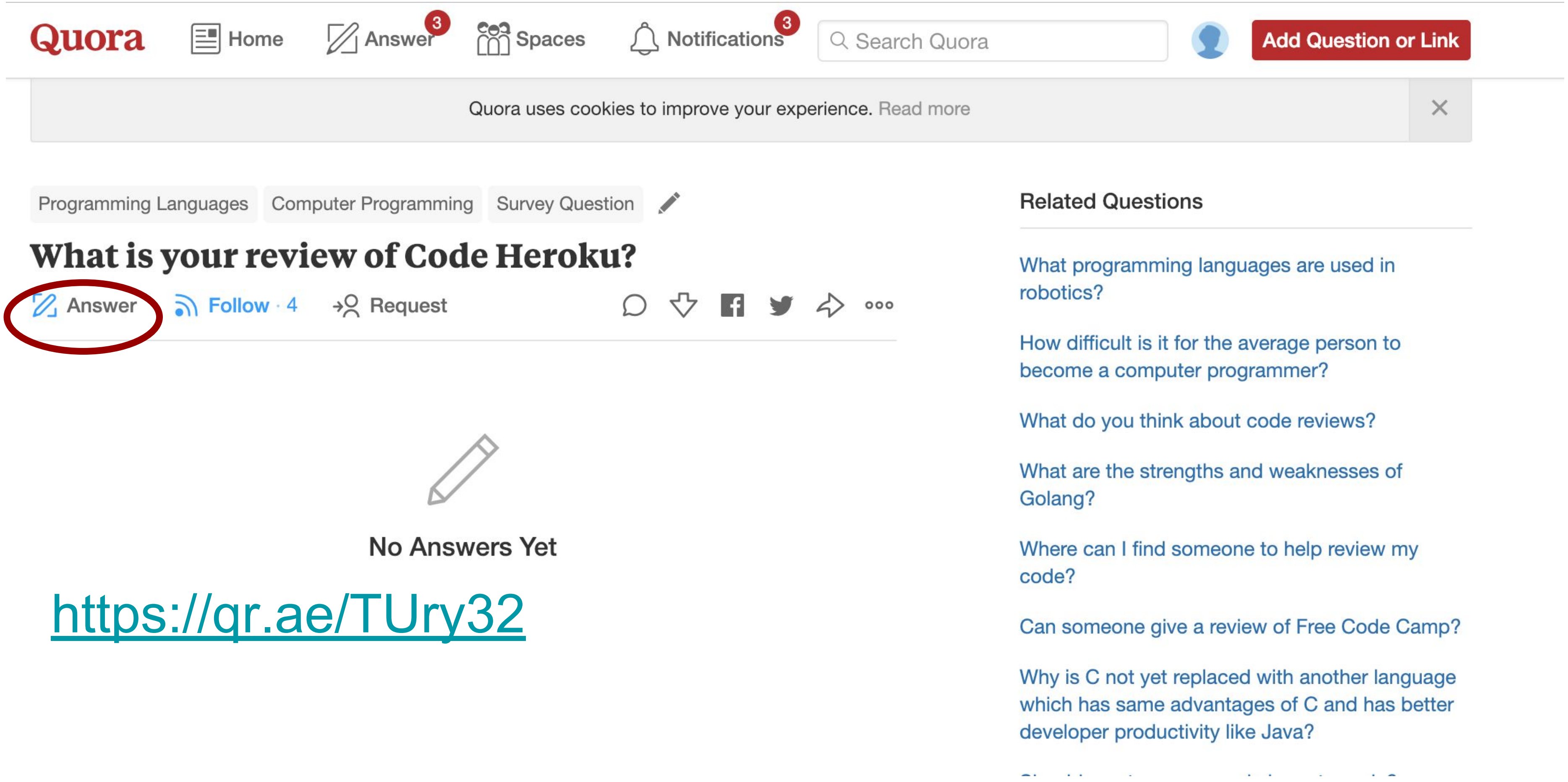
$m = m + \Delta m$
$b = b + \Delta b$

# QUIZ

How do we adjust m & b to reduce error?

1. We take the gradient of Error Function w.r.t. m & b this gives us direction we should adjust( i.e. +ve or -ve)
2. We use a constant learning rate to reduce error
3. We take gradient of all our data points (X,Y) w.r.t. m & b and multiply it by learning rate

# Thank you!



Quora uses cookies to improve your experience. Read more

# What is your review of Code Heroku?

No Answers Yet

Related Questions

- What programming languages are used in robotics?
- How difficult is it for the average person to become a computer programmer?
- What do you think about code reviews?
- What are the strengths and weaknesses of Golang?
- Where can I find someone to help review my code?
- Can someone give a review of Free Code Camp?
- Why is C not yet replaced with another language which has same advantages of C and has better developer productivity like Java?

https://qr.ae/TUry32

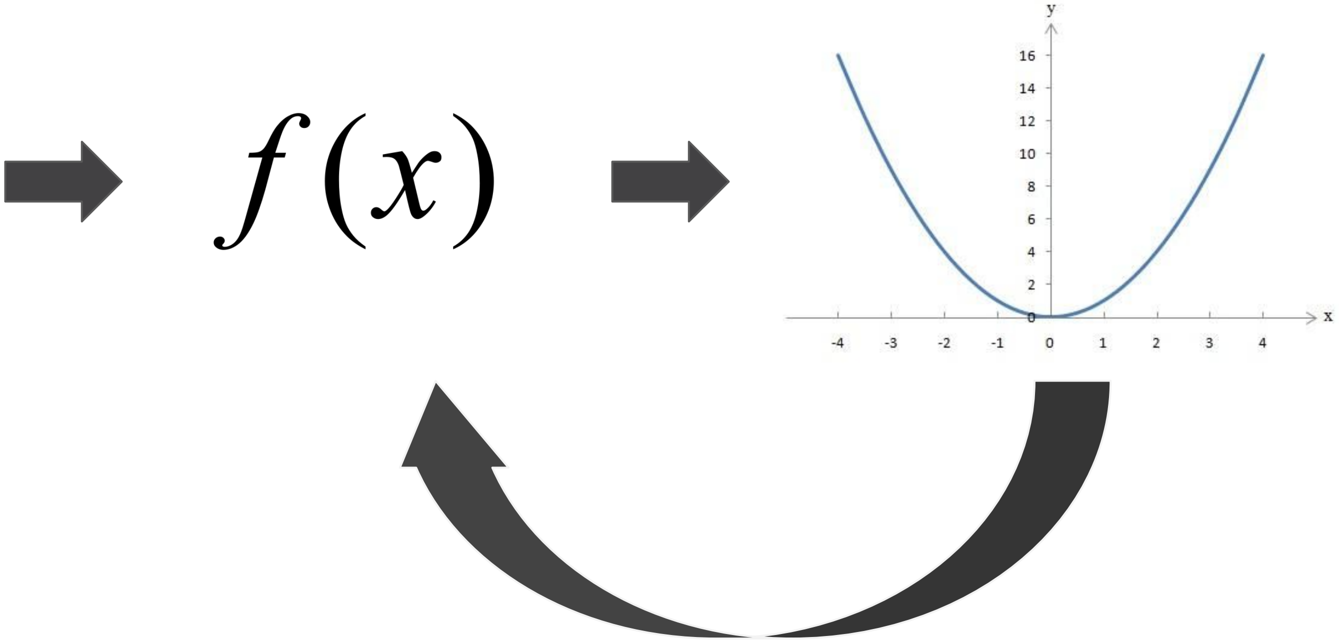
Alternative Links:

DataSet: https://drive.google.com/file/d/17MkZ6vzmZPEq9OTCM6tIgEPF5DYe_1Gm/view?usp=sharing

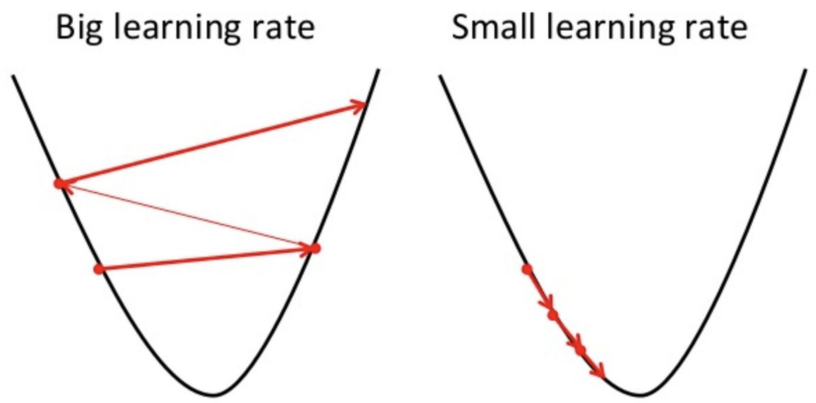
# Machine Learning (ML)

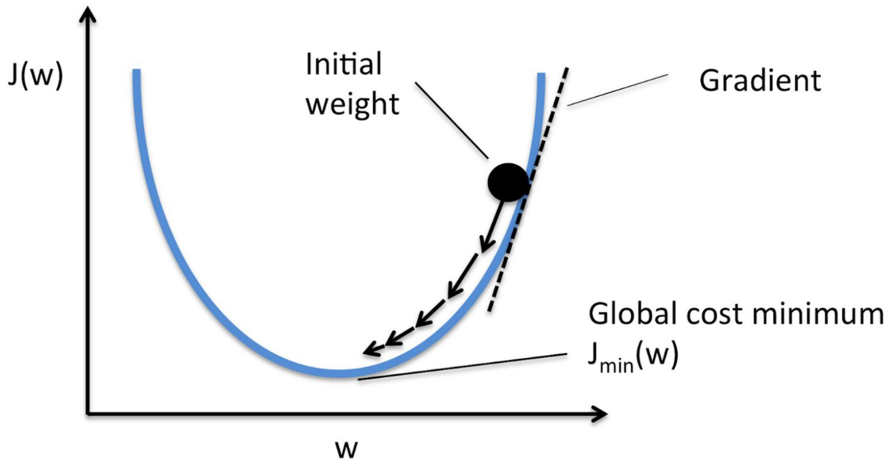
```
01101100 01100101 01110110 01100101 01110010 01100001
01100111 01101001 01101110 01100111 00100000 01100011
01100111 01101001 01100001 01110010 00100000 01100010
01101001 01100111 00100000 01100100 01100001 01110100
01100001 00100000 01100010 01110010 01101001 01101110
01100111 00100000 01100010 01101001 01100111 00100000
01100100 01100001 01110100 01100001 00100000 01110100
01101111 00100000 01100001 01100111 01110010 01101001
01100011 01110101 01101100 01110100 01110101 01110010
01100101 00101100 00100000 01100001 01101110 01100100
00100000 01100001 01100111 01110010 01101001 01100011
01110101 01101100 01110100 01110101 01110010 01100101
00100000 01110100 01101111 00100000 01100010 01101001
01100111 00100000 01100100 01100001 01110100 01100001
```

Data



$$f(x)$$


Big learning rate
Small learning rate


J(w)
Initial weight
Gradient
Global cost minimum
$J_{min}(w)$
w