LECTURE 16

Gradient Descent

Optimization methods to analytically and numerically minimize loss functions.

Sean Kang

If we have data concerning housing sizes and their corresponding prices.
How do we create a model for this?

area = [2600,3000,3200,3600,4000]

price = [550k,565k,610k,680k,725k]

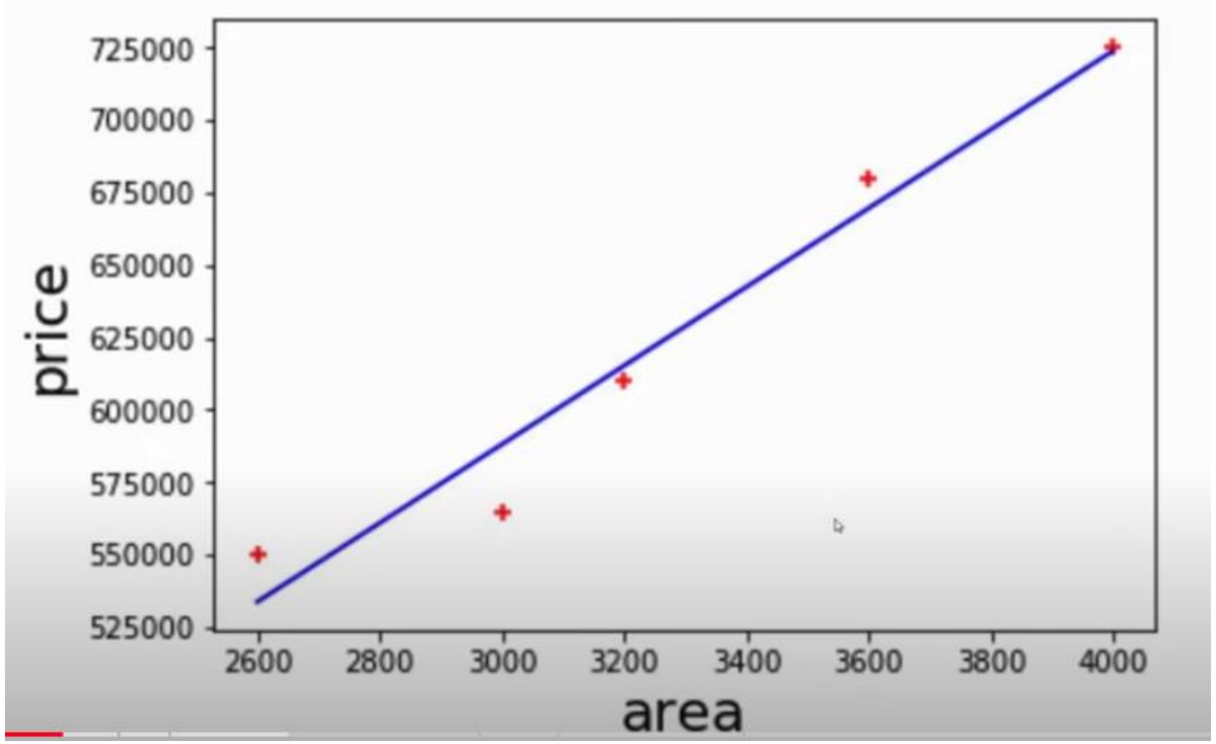
How do we derive some equation that looks like :

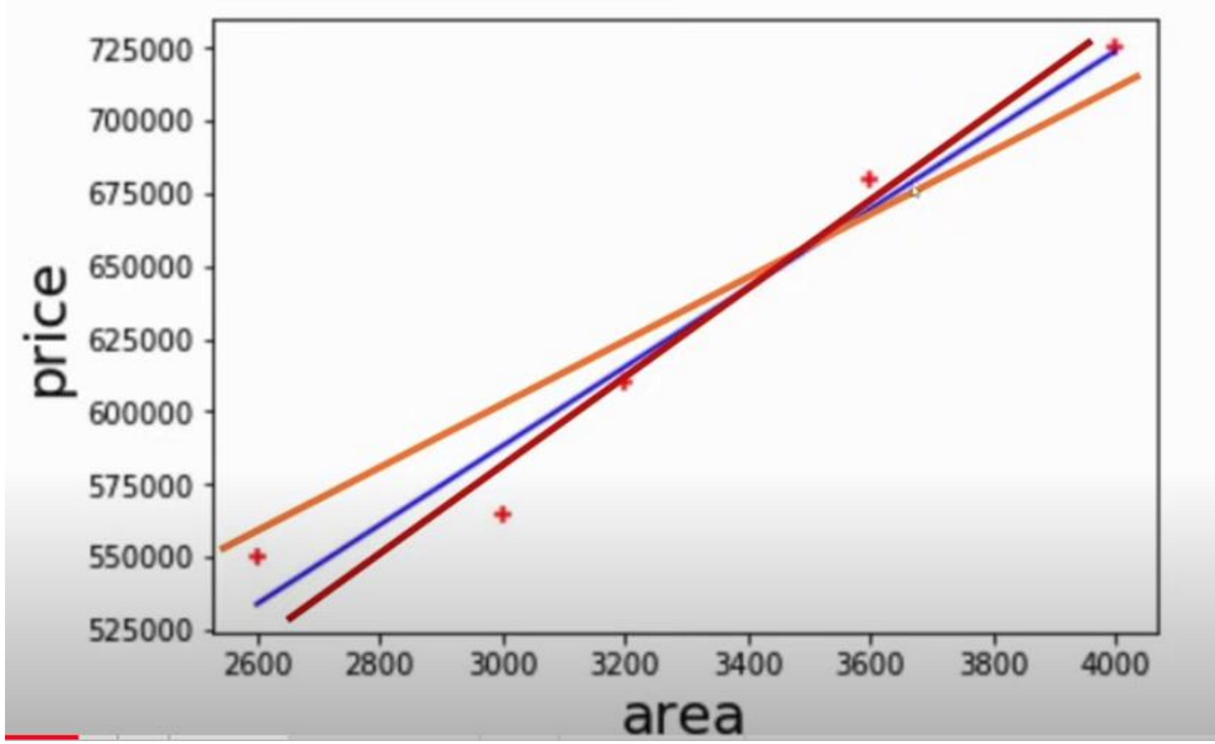
Price = b + m (Area)
(looks like $y = b + mx$). What is the value for b and m?

What is the best fit line?

Left: when there are few points, it might be easy to guess

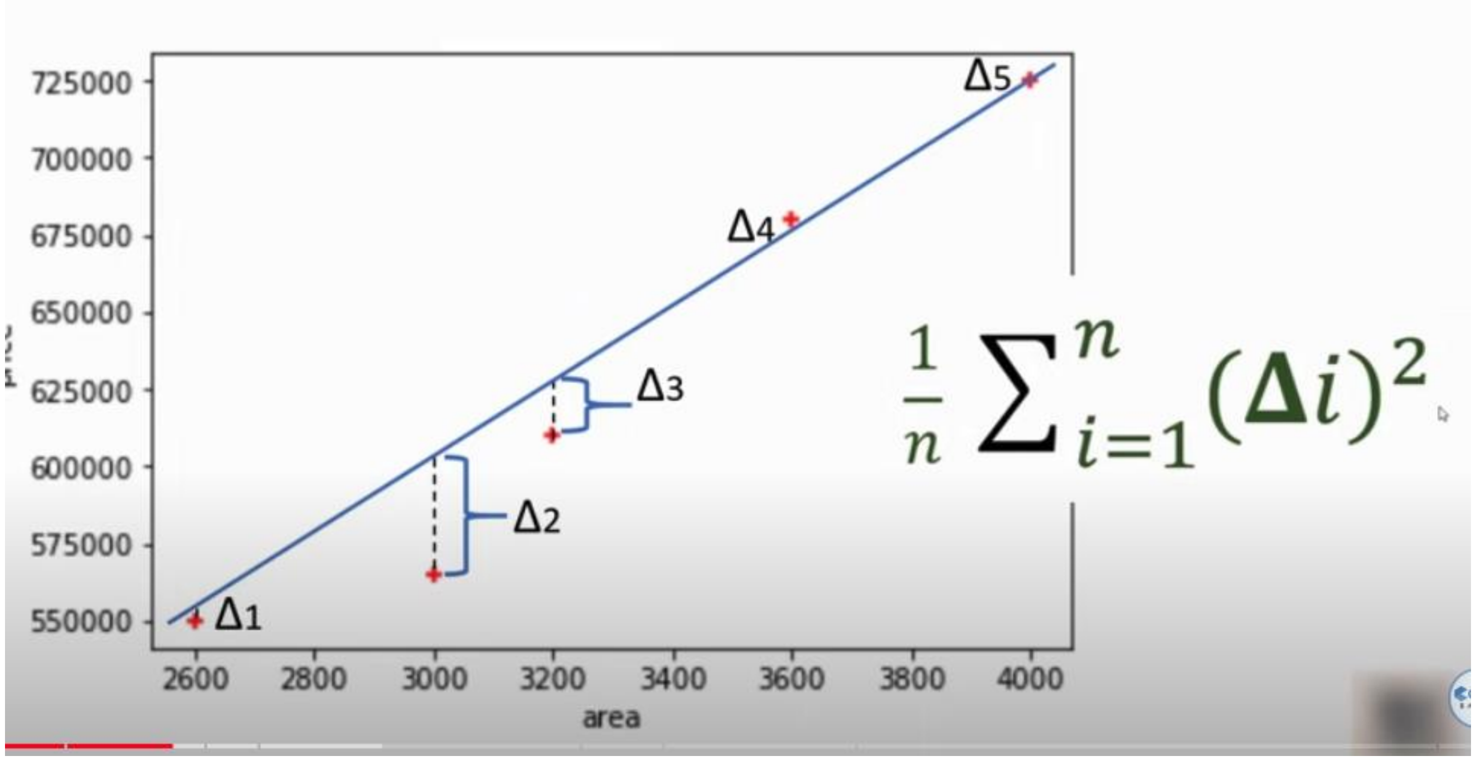
When there are 1000, or 1mil points, then it is harder to create a better fit visually



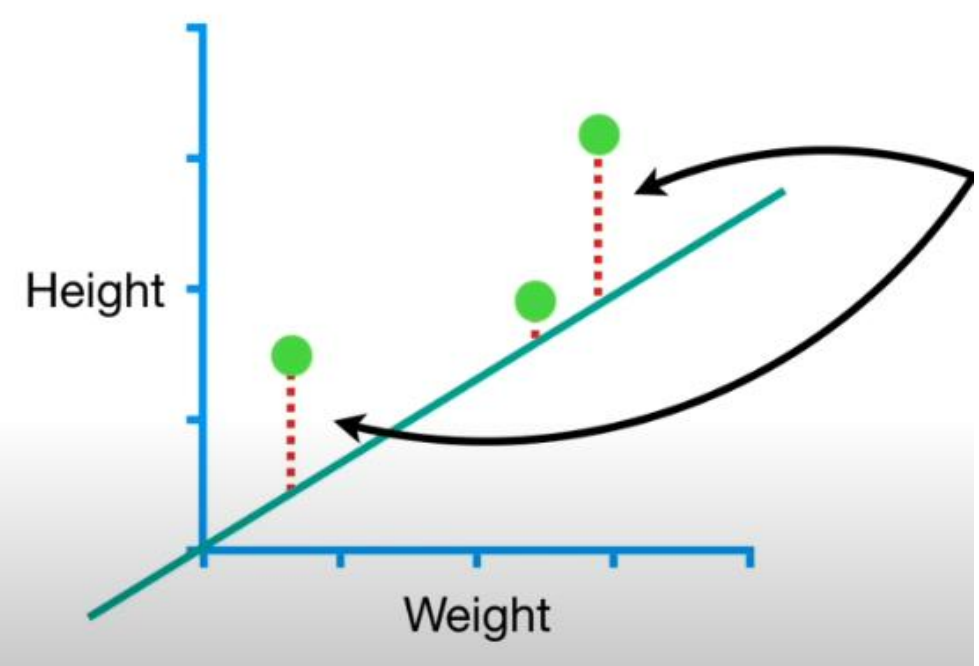


The approach

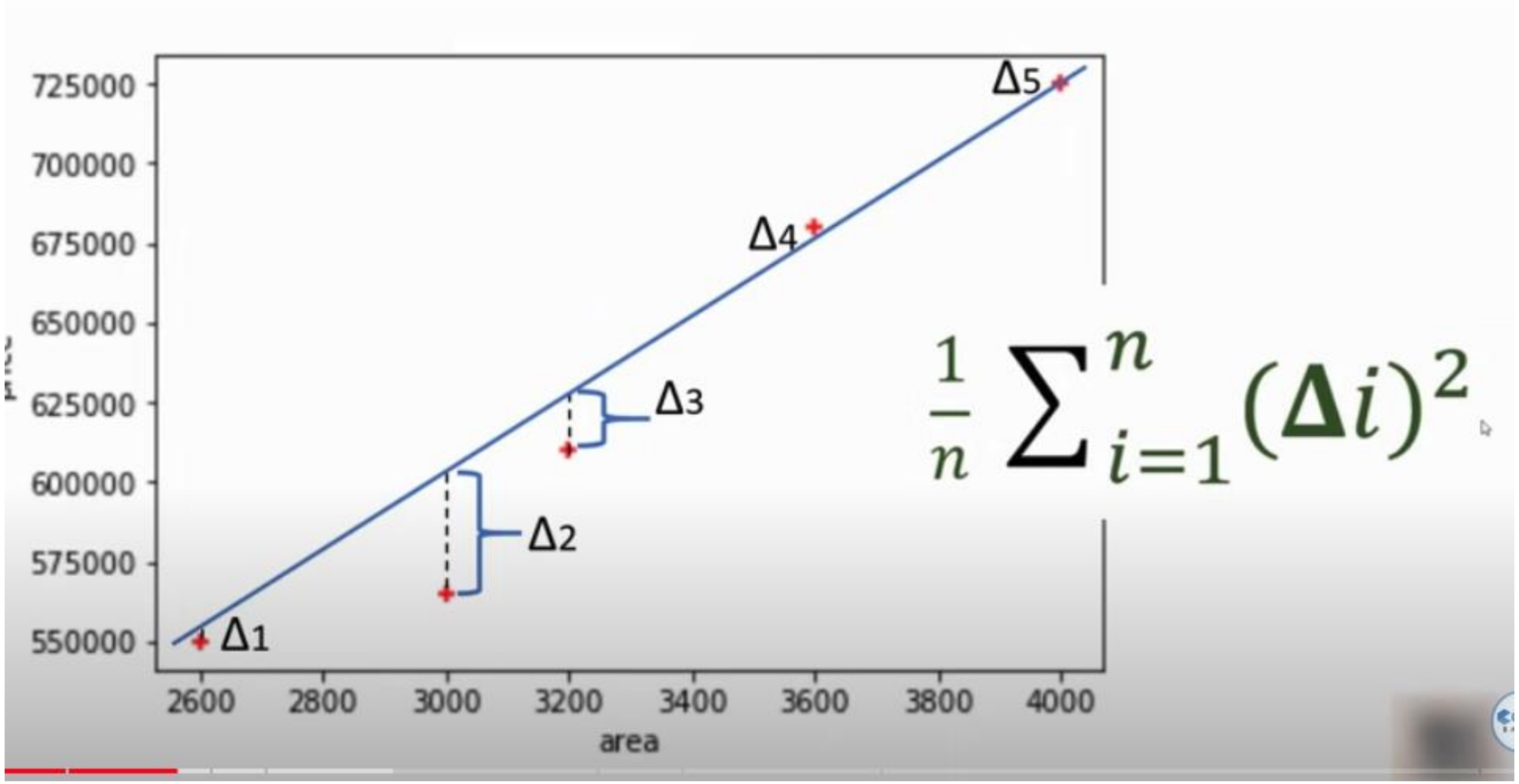
- Draw some random line
- Calculate the distance from actual y from the estimated y
- Square the difference -- squaring amplify the difference and ignore + or – difference
- Sum them up
- Calculate the average



Another example of Height prediction from weight



These differences is our COST function



This should look familiar from our Simple Linear Regression formula: MSE is Mean Square Error, our cost function

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

Looking at MSE and how to minimize it

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

ŷ = mx +b

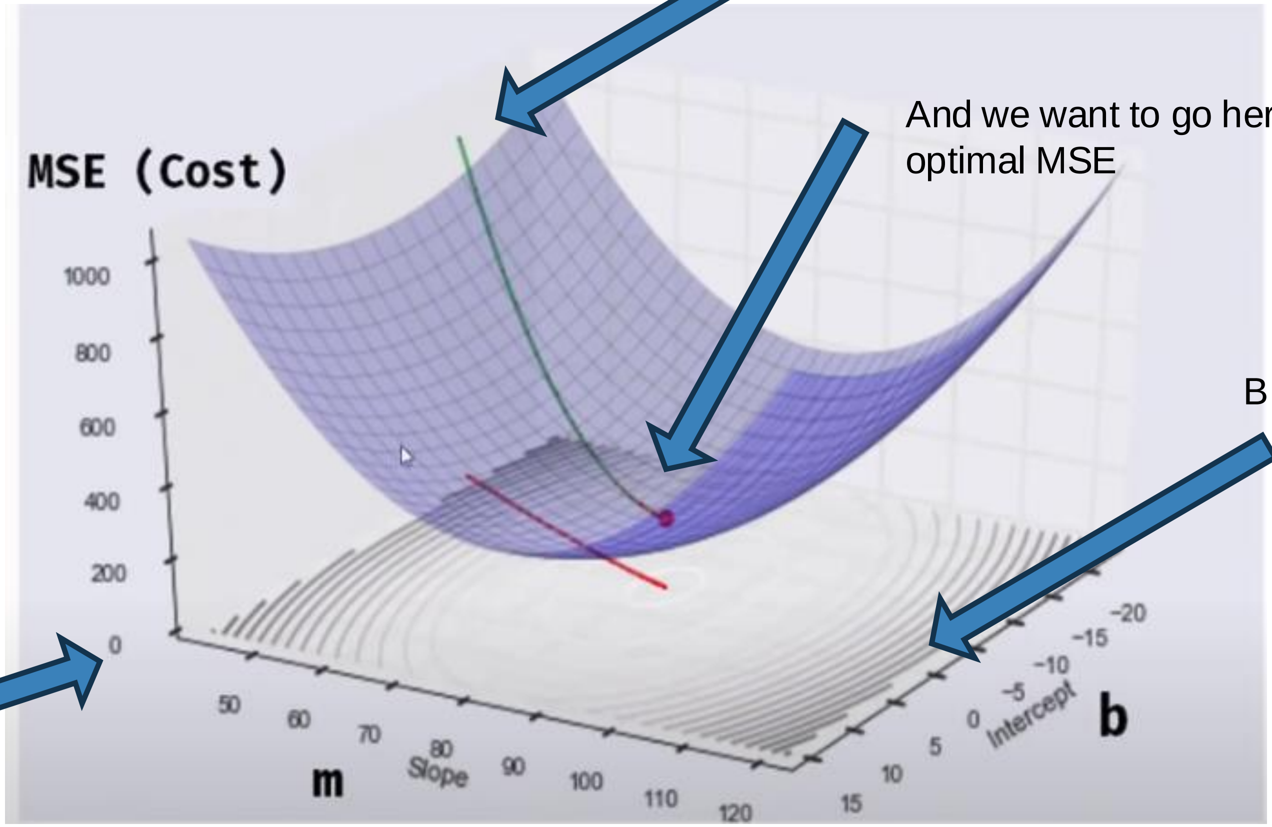
MSE(y, $\hat{y}$) = $\frac{1}{n} \sum_{i=1}^{n} (y - (mx + b))^2$

- Start with m with zero, and b as zero

Maybe we started here

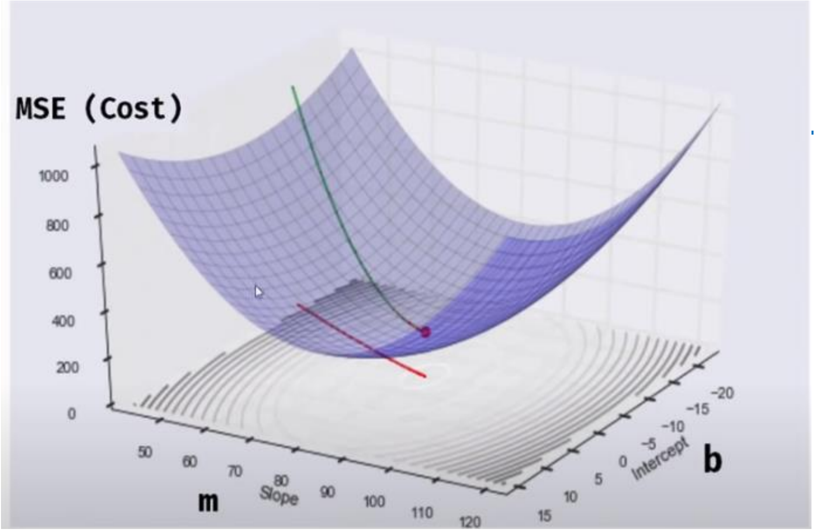
And we want to go here at final optimal MSE

B is zero here

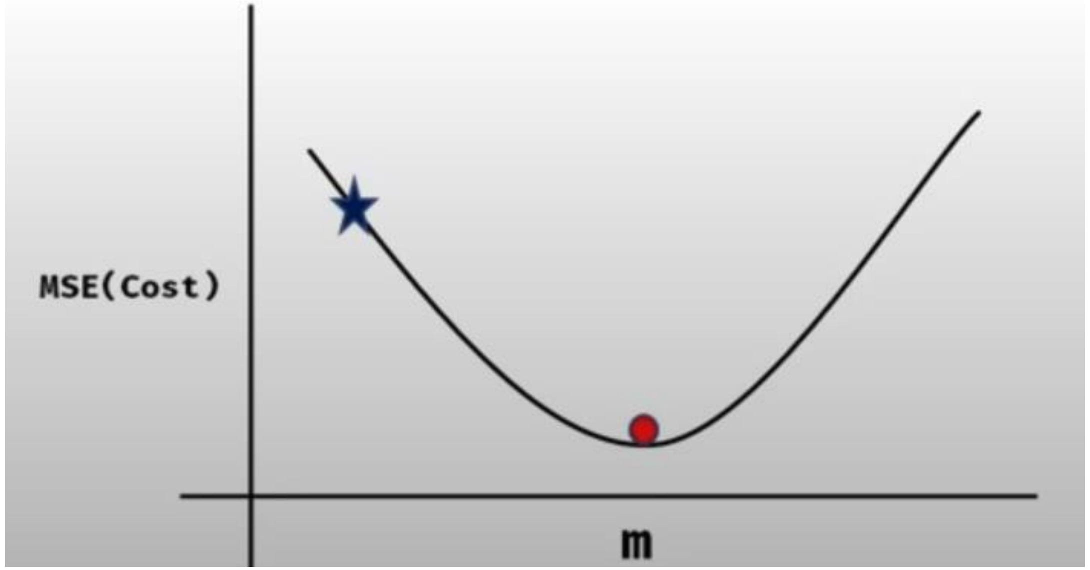


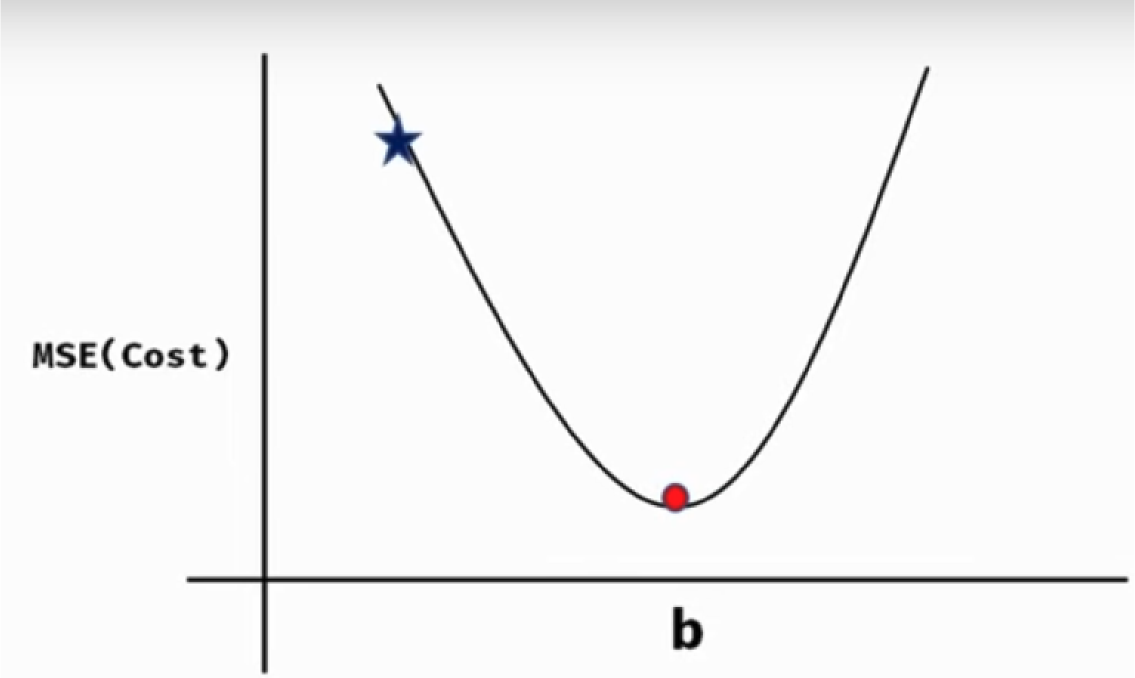
M = 0 here

So, how to find that minimal MSE



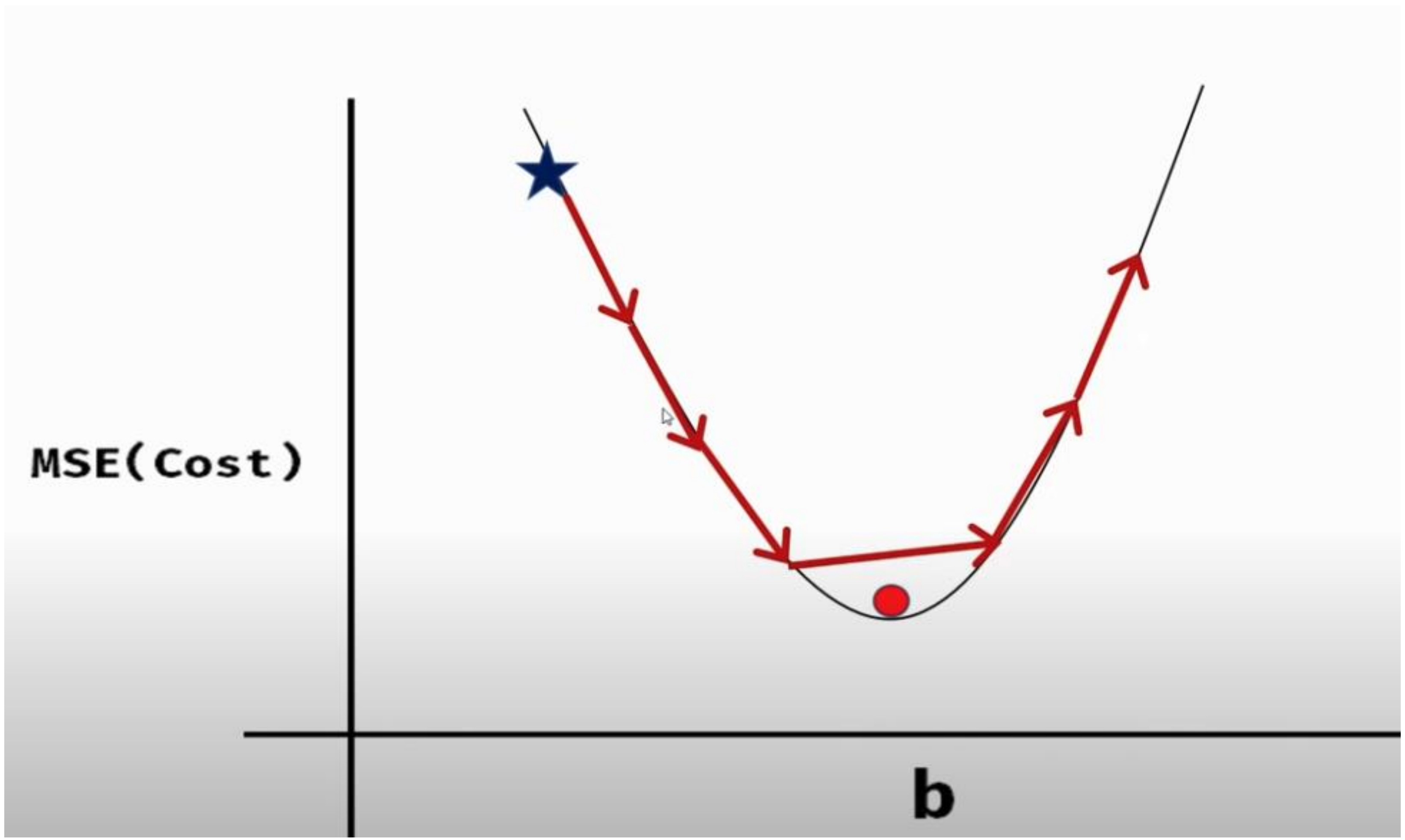
These are the single dimensional view of the 3D plot





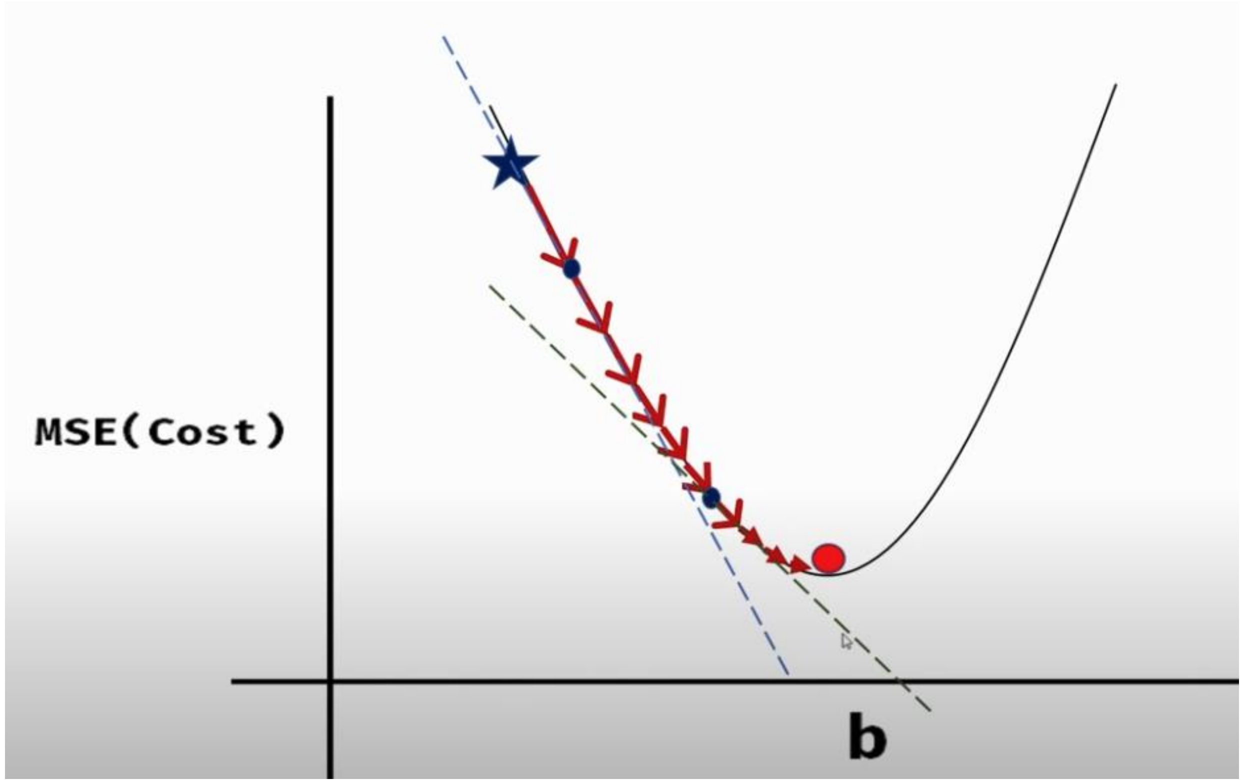
We don't know where the minimal MSE for a given m or b so we need to figure out an efficient way to calculate it

Incremental moving towards a smaller MSE



If we use fixed incremental sizes, we might miss the minimal MSE and realize MSE is increasing again

Finding that minimal MSE



- This approach might work, where the incremental steps are smaller as the MSE reduces…
- The slope is negative here
- We can use derivative of MSE relative to b to see where the slope gets closer to zero.

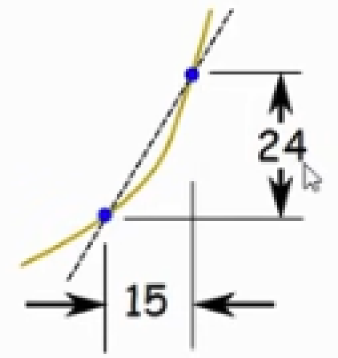
Detour into Derivatives and Partial Derivatives

$$\frac{d}{dx}x^2 = 2x$$

$$f(x, y) = x^3 + y^2$$

$$\partial f/_{\partial x} = 3x^2 + 0 = 3x^2$$

$$\partial f/_{\partial y} = 0 + 2y = 2y$$



What is partial derivative of MSEs

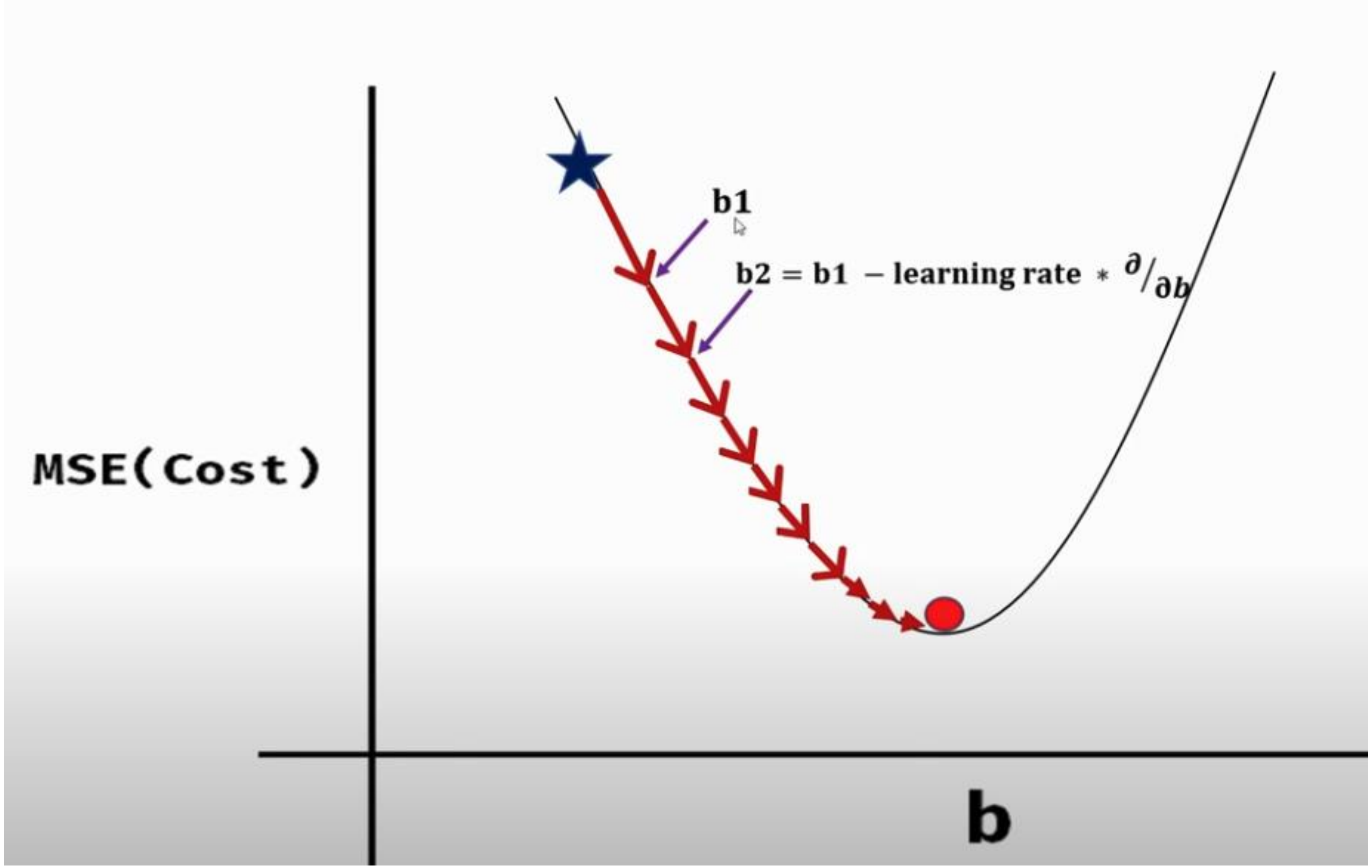
MSE(y, $\hat{y}$) = $\frac{1}{n}\sum_{i=1}^{n}(y - (mx + b))^2$

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

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

$$\partial/_{\partial b} = \frac{2}{\mathrm{n}}\sum_{i=1}^{n} -(y_i - (mx_i + b))$$

Using MSE and Derivatives to find our optimal b and m



Gradient Descent Algorithm

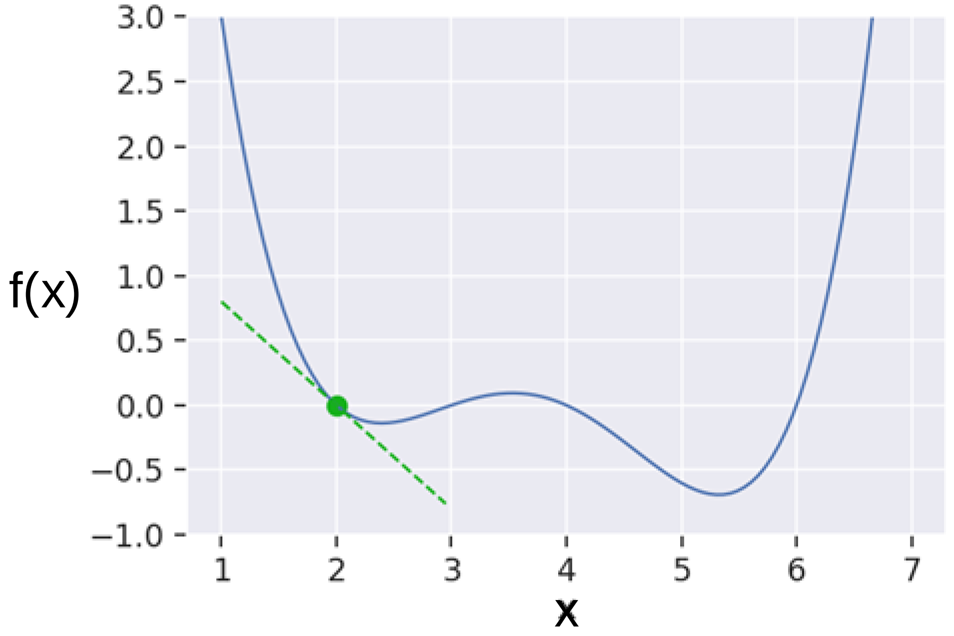
The gradient descent algorithm is shown below:

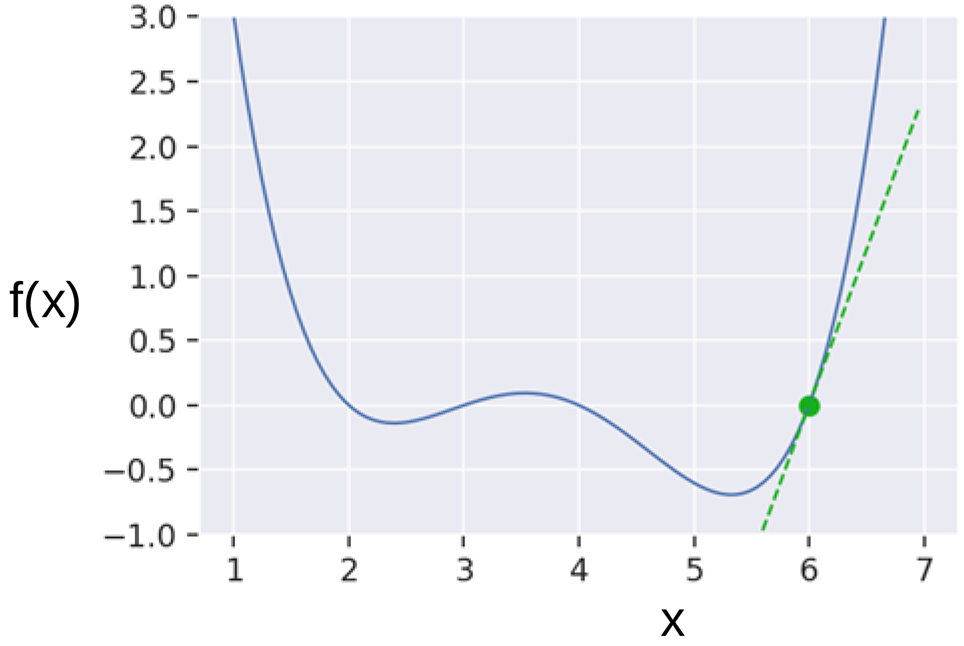
- alpha is known as the “learning rate”.
    - Too large and algorithm fails to converge.
    - Too small and it takes too long to converge.

Gradient Descent Intuition

The intuition behind 1D gradient descent:

- To the left of a minimum, derivative is negative (going down).
- To the right of a minimum, derivative is positive (going up).
- Derivative tells you where and how far to go.





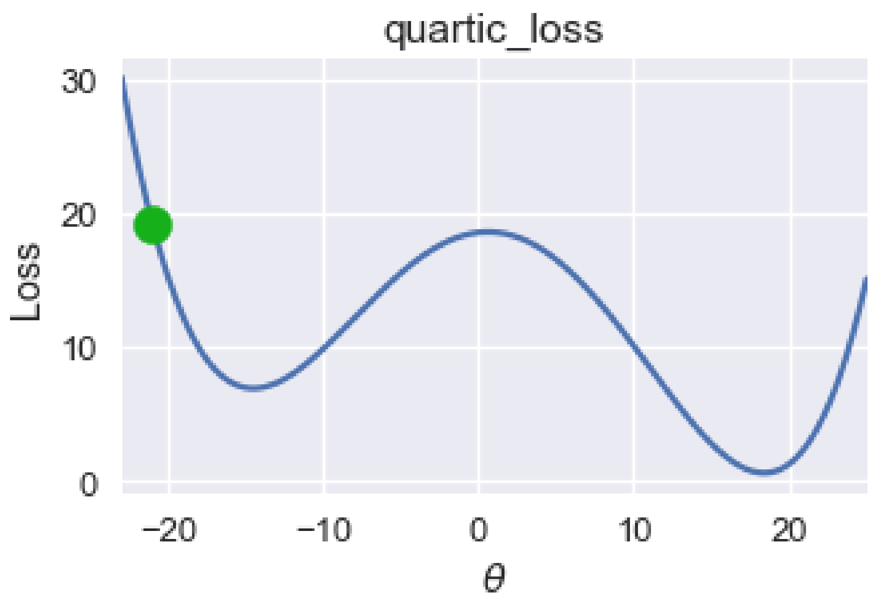
Demo time

Summary of GD Approach

- Step 1: Calc derivative of Loss Function (MSE) for each parameter (ML lingo : get the gradient of the loss function)
- Step 2: Pick random values for the parameters
- Step 3: Use parameters into the derivatives (aka gradient)
- Step 4: Calc the step size
- Step 5: Calculate new parameters
- Step 6: Repeat at Step 3 or exceed the iterations

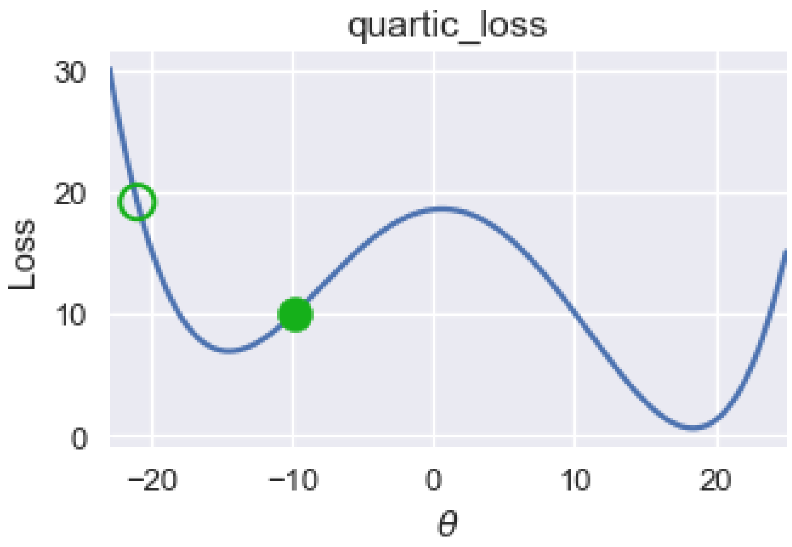
Simple Gradient Descent Approach Only Finds Local Minima

- If loss function has multiple local minima, GD is **not guaranteed** to find global minimum.
- Suppose we have this loss curve (not a convex function)



Gradient Descent Only Finds Local Minima

- Here's how GD runs



- GD can converge at -15 when global minimum is 18 (this happens with quadratic functions, or non-linear functions)

Convexity

- For a **convex** function f, any local minimum is also a global minimum.
    - gradient descent will always find the globally optimal minimizer.
- For concave functions or non-linear functions, GD will find the first local minimum

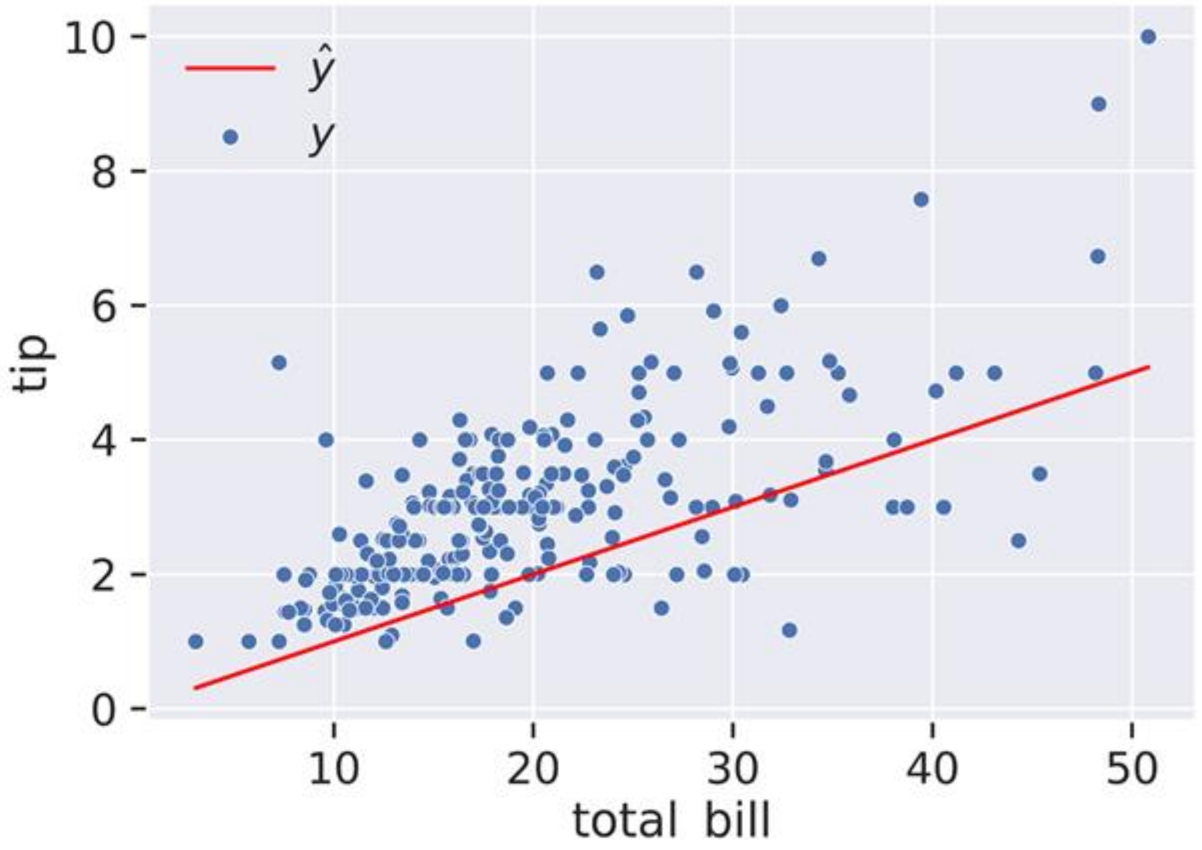
Optimization Goal

Suppose we want to create a model that predicts the tip given the total bill for a table at a restaurant.

For this problem, we'll keep things simple and have only 1 parameter: gamma.

$$\hat{y} = f_{\hat{\gamma}(\vec{x})} = \hat{\gamma}\vec{x}$$

- In other words, we are fitting a line with zero y-intercept.



See Notebook.

Stochastic Gradient Descent

Which Step in This Algorithm is Most Time Consuming?

Gradient Descent Algorithm

- Calculating MSE at each iteration
- Calculating the Derivatives at each iteration

Stochastic Gradient Descent

Draw a smaller simple random sample of data indices

- Often called a **batch** or **mini-batch**
- Choice of **batch size** trade-off **gradient quality** and **speed**

- **Think of the size of the Berkeley call for service data – compared to Los Angels, CA home sales records.** https://maps.assessor.lacounty.gov/m/
  - **Not easy to navigate that web**
  - **But the public free data exists**