

Intro to AI
By Dr. Dongchul Kim

GRADIENT DESCENT ALGORITHM

Optimization

Finding best a and b is an optimization problem.

From now on, we will use w instead of a to represent slope. Our linear model is $y = wx + b$,

Our goal is to estimate w (slope) and b that minimizes the Cost Function (MSE) given a particular data set (x and y)

Let's assume that it's linear model and we have a data set as follows.

x	y
1.0	2.0
2.0	3.0
3.0	4.0

What is optimal w and b ? Can you guess?

Yes! Intuitively, we should be able to guess the answer.

$w = 1$ and $b = 1$

(if you could not guess that w and b should both be 1, try to picture the points on a graph in your mind. As x increases by 1, we can see that y increases by 1 as well so that tells us the slope of the graph. We then know can remember that in our high-school math class to find b , we simply put 0 into x and see what the result is, in this case when $x = 0$, $y = 1$ as expected)

But please let's assume we don't know the answer.

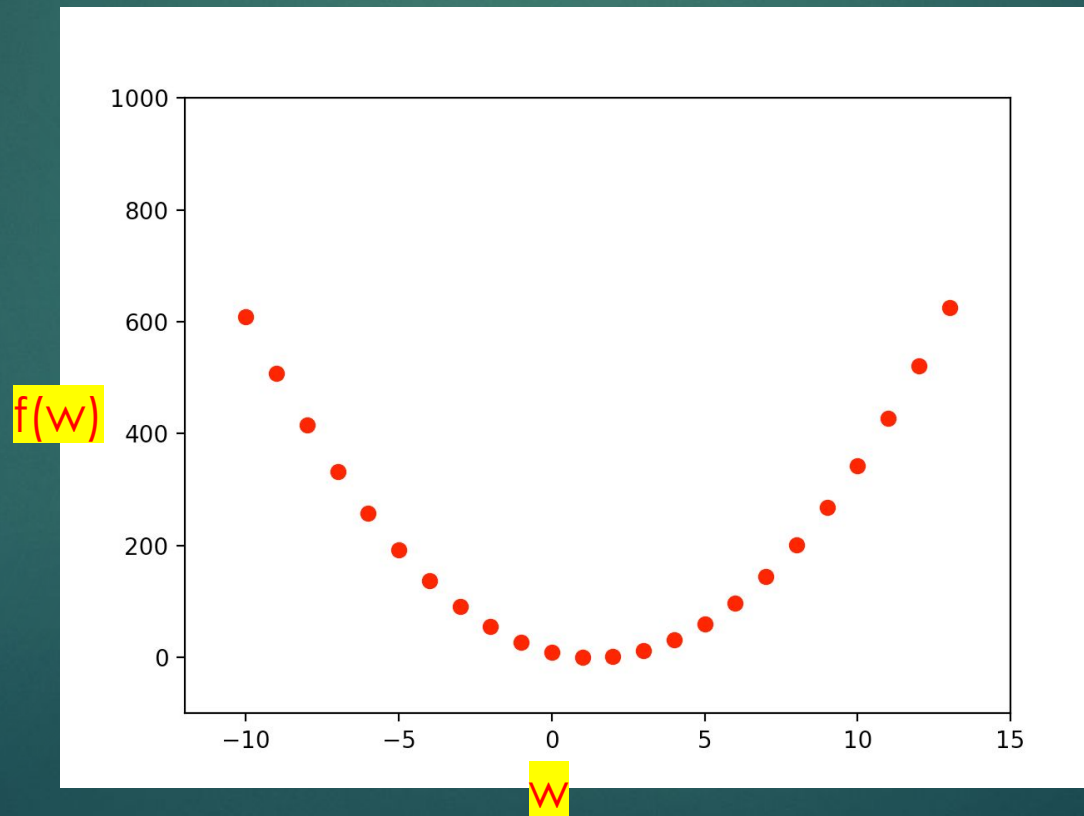
Example

x	y
1.0	2.0
2.0	3.0
3.0	4.0

- ▶ To find optimal w and b , the first step is to set w and b as 0 (zero).
- ▶ Then, estimate w first. (we don't change the value of b)
- ▶ We just calculate the cost (MSE) when w is 0, 1, 2. It is like a simple searching with different values.
- ▶
- ▶ $w = 0, b = 0$
 - ▶ Cost = $((1*0 + 0 - 2)^2 + (2*0 + 0 - 3)^2 + (3*0 + 0 - 4)^2) / 3 = 9.67$
- ▶ $w = 1, b = 0$
 - ▶ Cost = $((1*1 + 0 - 2)^2 + (2*1 - 3)^2 + (3*1 - 4)^2) / 3 = 1$
- ▶ $w = 2, b = 0$
 - ▶ Cost = $((1*2 + 0 - 2)^2 + (2*2 - 3)^2 + (3*2 - 4)^2) / 3 = 1.67$

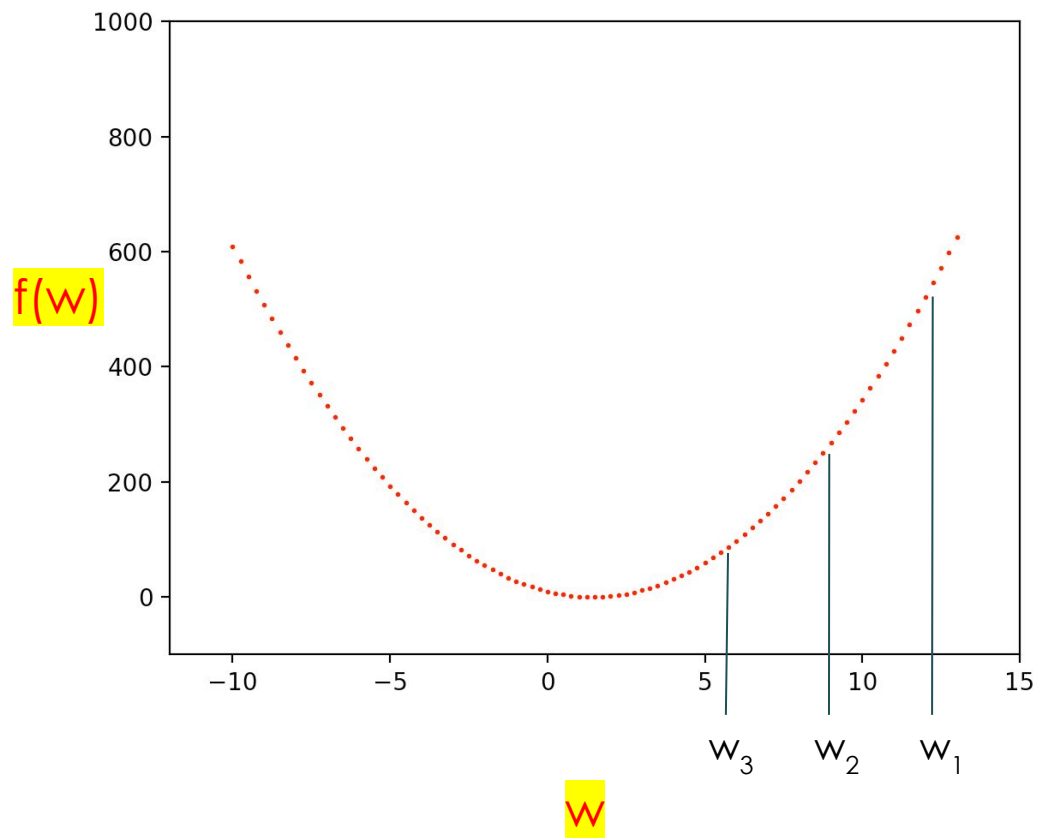
Example

- ▶ Cost function of w (given the data and $b = 0$)



The y-axis represents the error, and the x-axis represents w . The point with the smallest error is the lowermost convex part of the graph. That is, when w is 1, the error is the smallest. However, **since we assume we do not know the answer**, to find the optimal w , we need to calculate the error for a random point w_1 and move w to the side where the error decreases. In other words, the error is smaller for w_2 than for w_1 . The error is smaller for w_3 than for w_2 .

Gradient descent is a method to find w with the smallest error by comparing errors in this way.

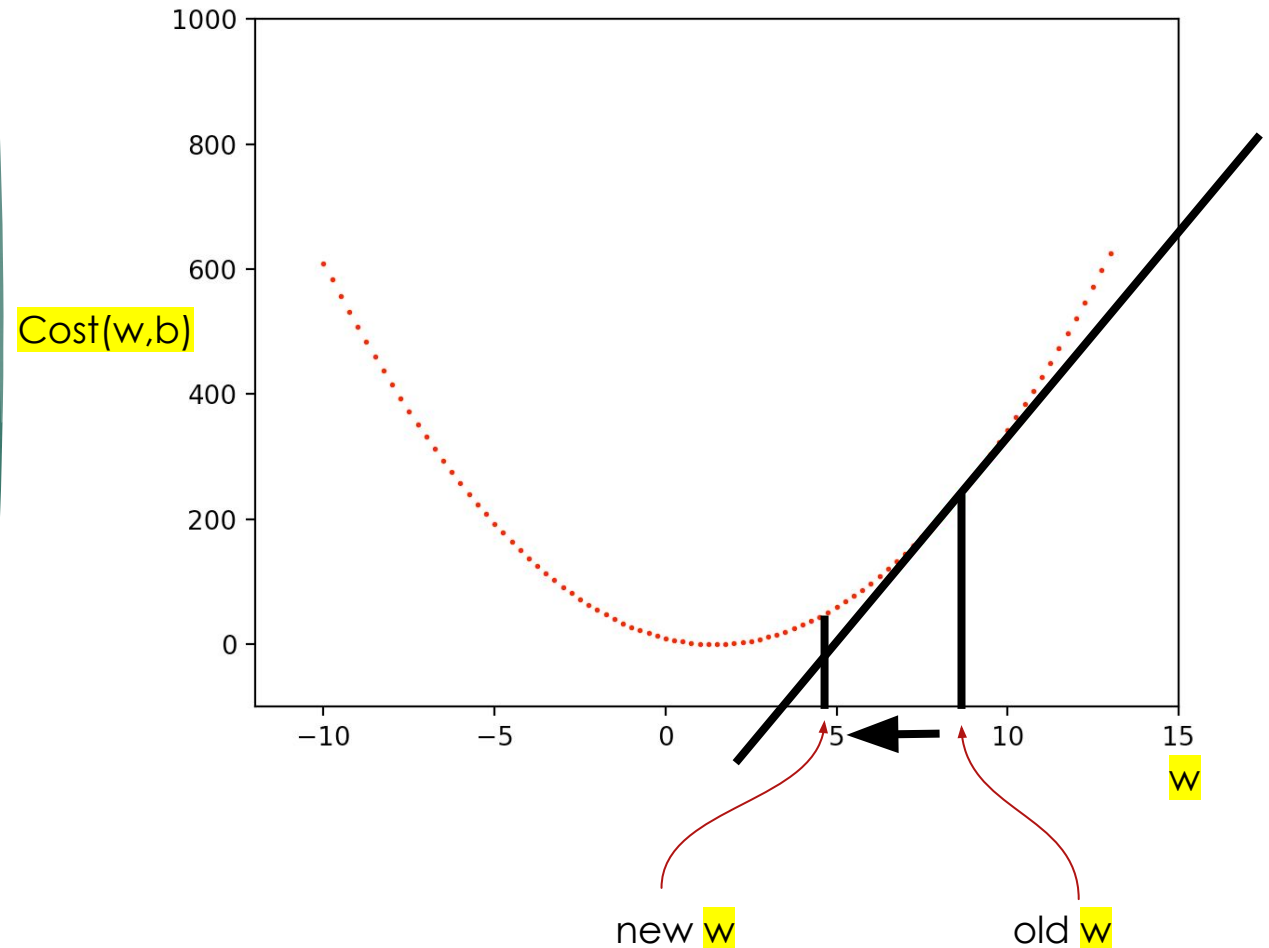


Gradient Descent Algorithm

- ▶ Step1: Initialize to 0
- ▶ Step2: update w and b that reduce cost function
- ▶ Use a gradient of cost function
- ▶ Step3: if w and b converge, stop step2. Otherwise, repeat step2
- ▶ New w

$$w' := w - \alpha \cdot \frac{\partial}{\partial w} \text{Cost}(w, b)$$

- ▶ α is "Learning Rate"

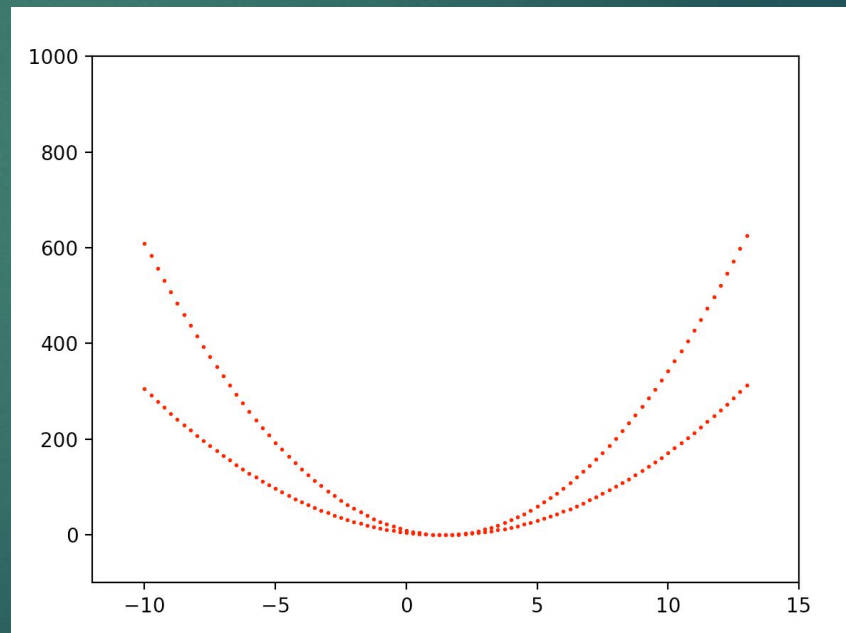


Little modification

- ▶ For optimization, these two cost functions will have same **w** and **b** to minimize each cost function

- ▶ $\min_w \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i)^2$

- ▶ $\min_w \frac{1}{2n} \sum_{i=1}^n (wx_i + b - y_i)^2$



Partial Differential

$$\blacktriangleright w' := w - \alpha \cdot \frac{\partial}{\partial w} Cost(w, b)$$

$$\blacktriangleright w' := w - \alpha \cdot \frac{\partial}{\partial w} \frac{1}{2n} \sum_{i=1}^n (wx_i + b - y_i)^2$$

$$\blacktriangleright w' := w - \alpha \cdot \frac{1}{2n} \sum_{i=1}^n (wx_i + b - y_i) 2x_i$$

$$\blacktriangleright w' := w - \alpha \cdot \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i) x_i$$

← **Update Rule**

Partial Differential

$$b' = b + \alpha \cdot \frac{\partial}{\partial b} \text{Cost}(w, b)$$

$$b' = b + \alpha \cdot \frac{\partial}{\partial b} \frac{1}{2n} \sum_{i=1}^n (wx_i + b - y_i)^2$$

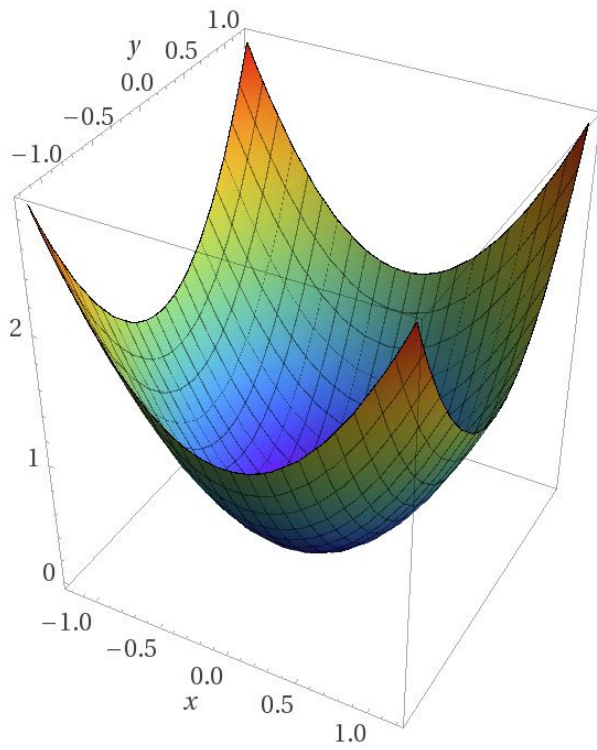
$$b' = b + \alpha \cdot \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i)$$

Gradient Descent Algorithm

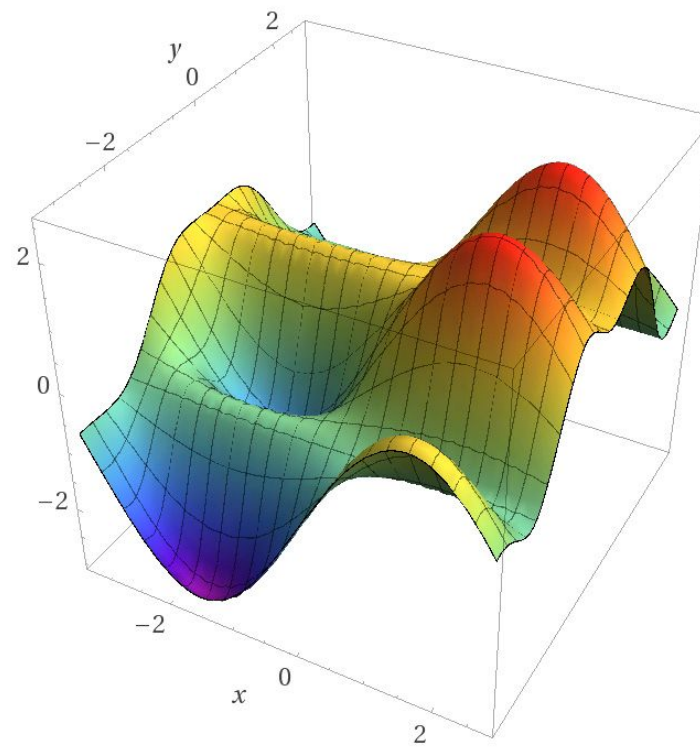
$$w' := w - \alpha \cdot \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i)x_i$$

$$b' = b + \alpha \cdot \frac{1}{n} \sum_{i=1}^n (wx_i + b - y_i)$$

Convex Function



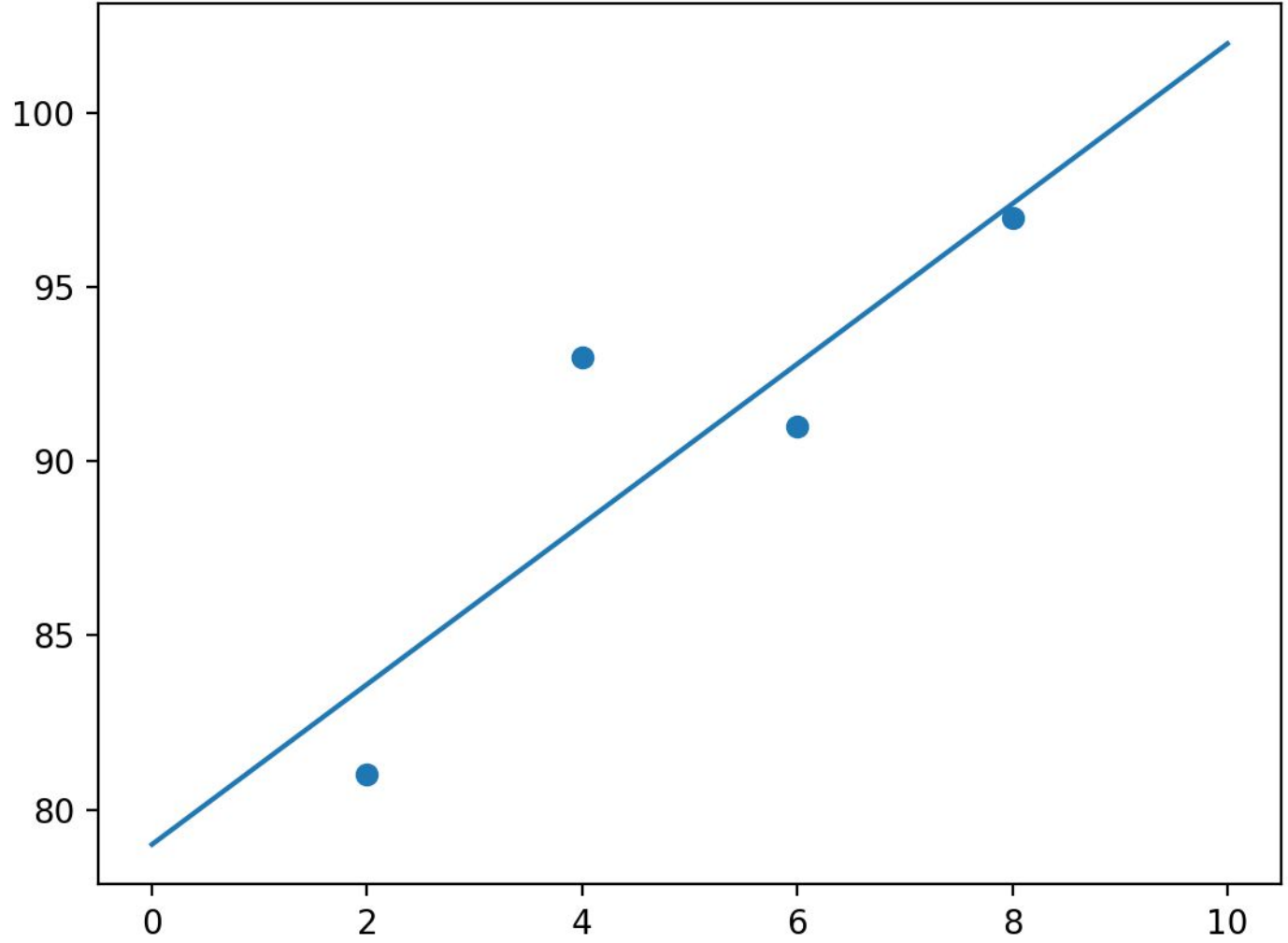
Computed by Wolfram|Alpha



Computed by Wolfram|Alpha

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 data = np.array([[2, 81], [4, 93], [6, 91], [8, 97]])
5 x = data[:, 0]
6 y = data[:, 1]
7
8 # initialization
9 w, b = 0, 0
10 # learning rate
11 alpha = 0.05
12
13 plt.scatter(x, y)
14 xl = np.linspace(0, 10, 100)
15 # GD
16 for i in range(2000):
17     w = w - alpha * (1/len(data)) * sum((w * x + b - y) * x)
18     b = b - alpha * (1/len(data)) * sum((w * x + b - y))
19 print("w = %f, b = %f" % (w, b))
20 plt.plot(xl, w * xl + b)
21 plt.show()
```

$w = 2.300000, b = 79.000000$



Lab 44 - Gradient Descent Algorithm

- ▶ Write a program that estimates optimal w and b by implementing GD algorithm given a data below. (Hint: GD algorithm)
- ▶ Answer is about $w = 2.x$ and $b = 1.x$.

x	y
2.3	6.13
1.2	4.71
4.3	11.13
5.7	14.29
3.5	9.54
8.9	22.43

Next time

Today, we have talked about how to optimize parameters (w and b) given a linear model that has only a single independent variable (x).

What if there are multiple variables?

For example, $y = w_1x_1 + w_2x_2 + w_3x_3 + b$

Next time, we are going to talk about how to estimate the parameters given a multivariable linear model.