Regression 1

Supervised learning

Unsupervised learning

Reinforcement learning

Generative AI

Supervised learning

$$feature(X) \longrightarrow model(f) \xrightarrow{predict} label(y)$$

Supervised learning - examples

application	feature(x)	label(y)
spam detection	email content	spam or not spam
house price prediction	size, location, age	price
online shopping	user behavior	purchase or not purchase
cancer detection	medical images	cancer or not cancer

Regression

predict a number. Many possible outputs.

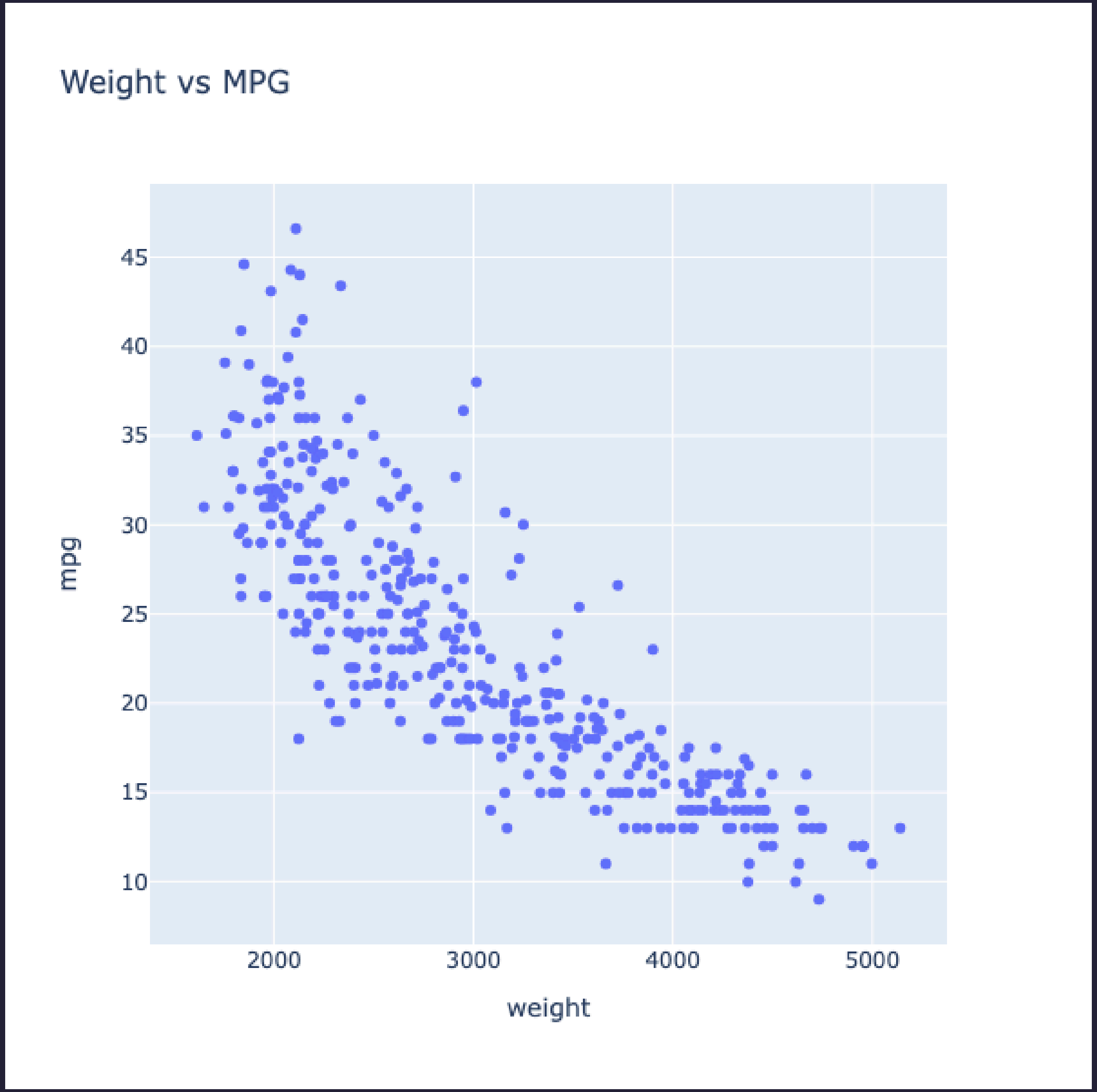
Classification

predict categories. Small number of possible outputs.

Regression

Car fuel efficiency

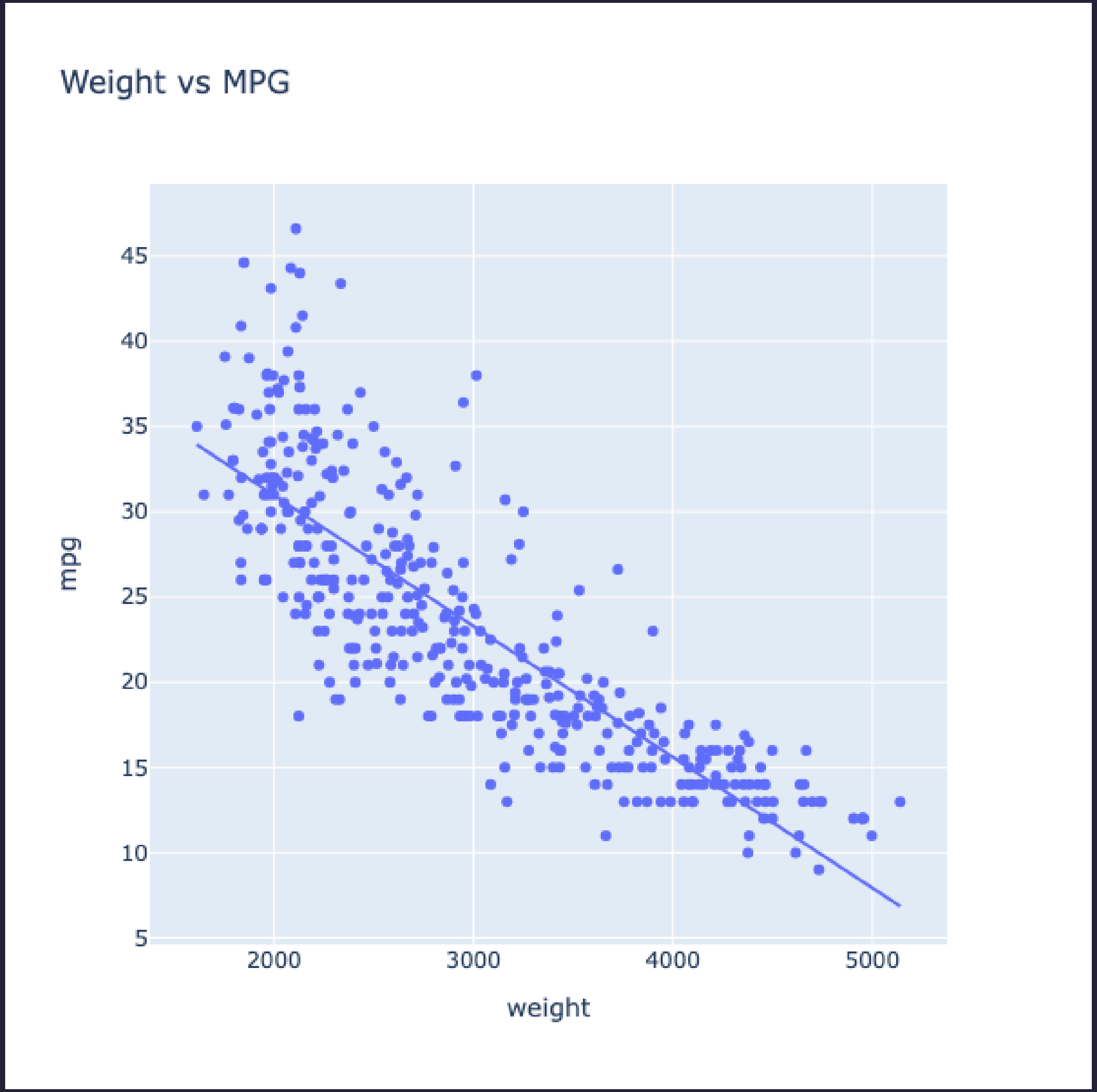
	mpg	weight
0	18.0	3504
1	15.0	3693
2	18.0	3436
3	16.0	3433
4	17.0	3449



A prediction based on weight

weight=3500, mpg=?

weight=1000, mpg=?



Notation

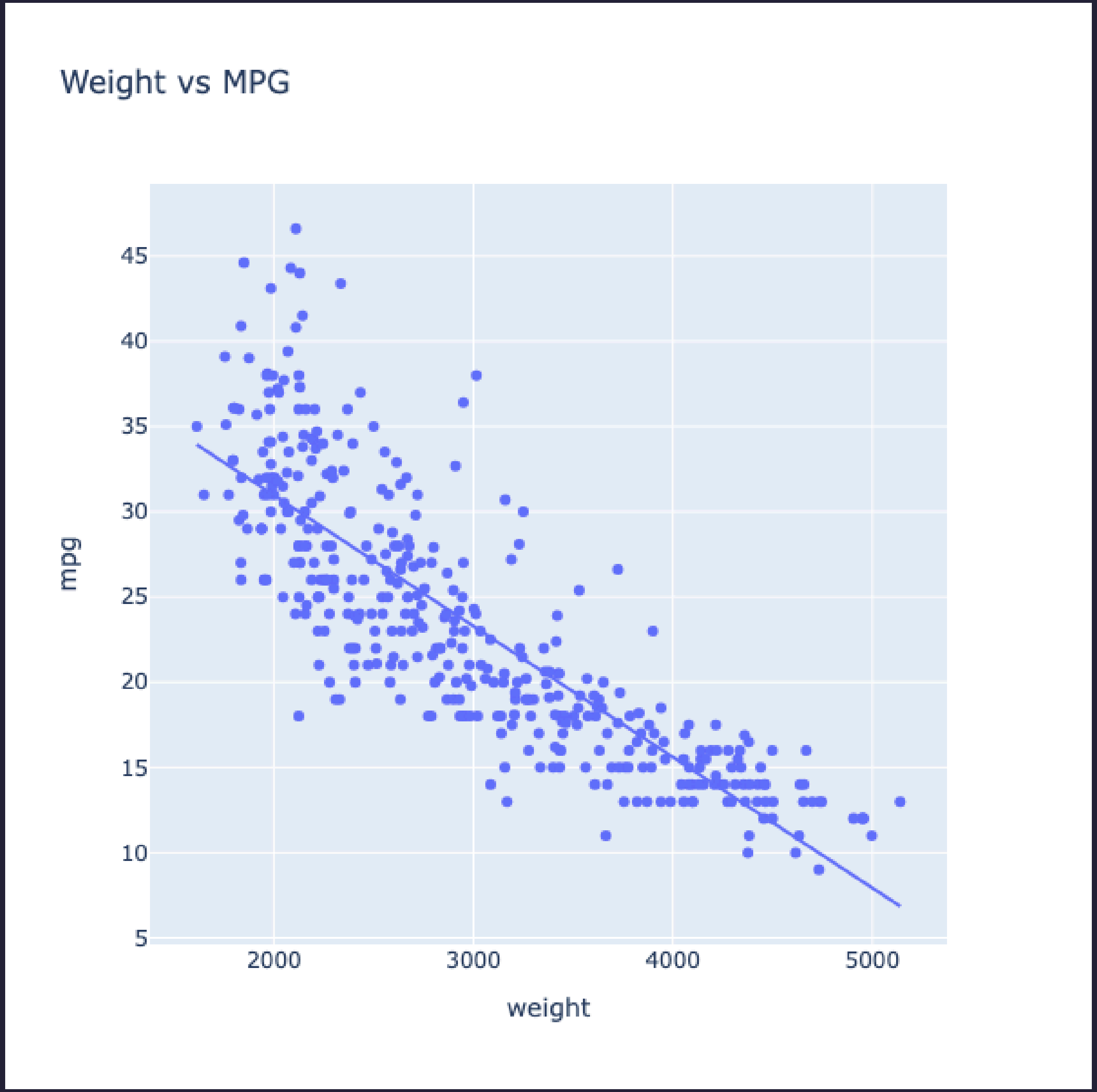
y: label (mpg)

x: feature (weight)

w: weight (slope)

b: bias (intercept)

$\hat{y}$, $f(x)$: model (predicted label given feature)

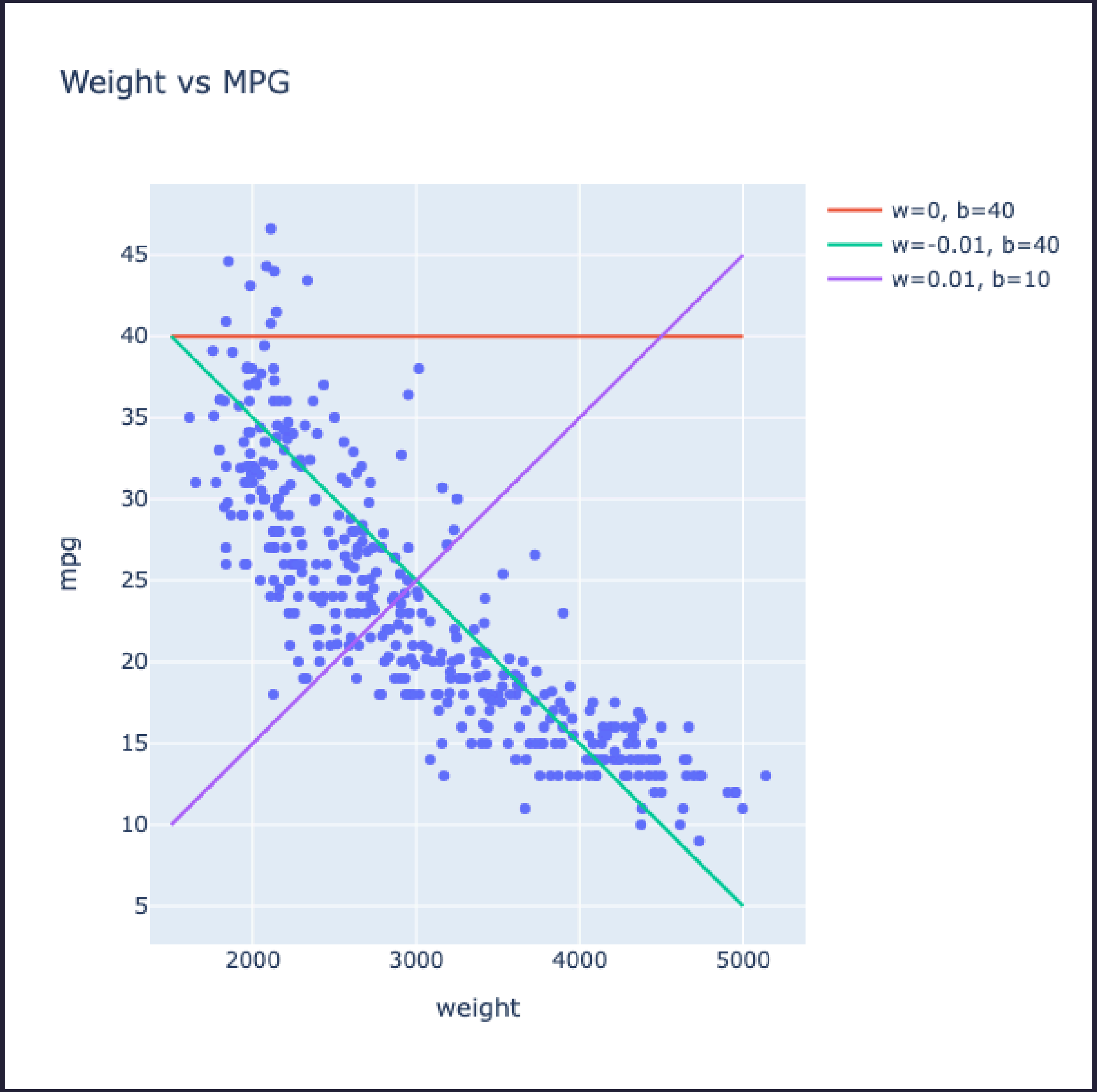


Prediction lines

$$\hat{y} = f(x) = wx + b$$

parameters

- w: weight (slope)
- b: bias (intercept)



💻 Prediction

How to determine the best model?

Loss & cost functions

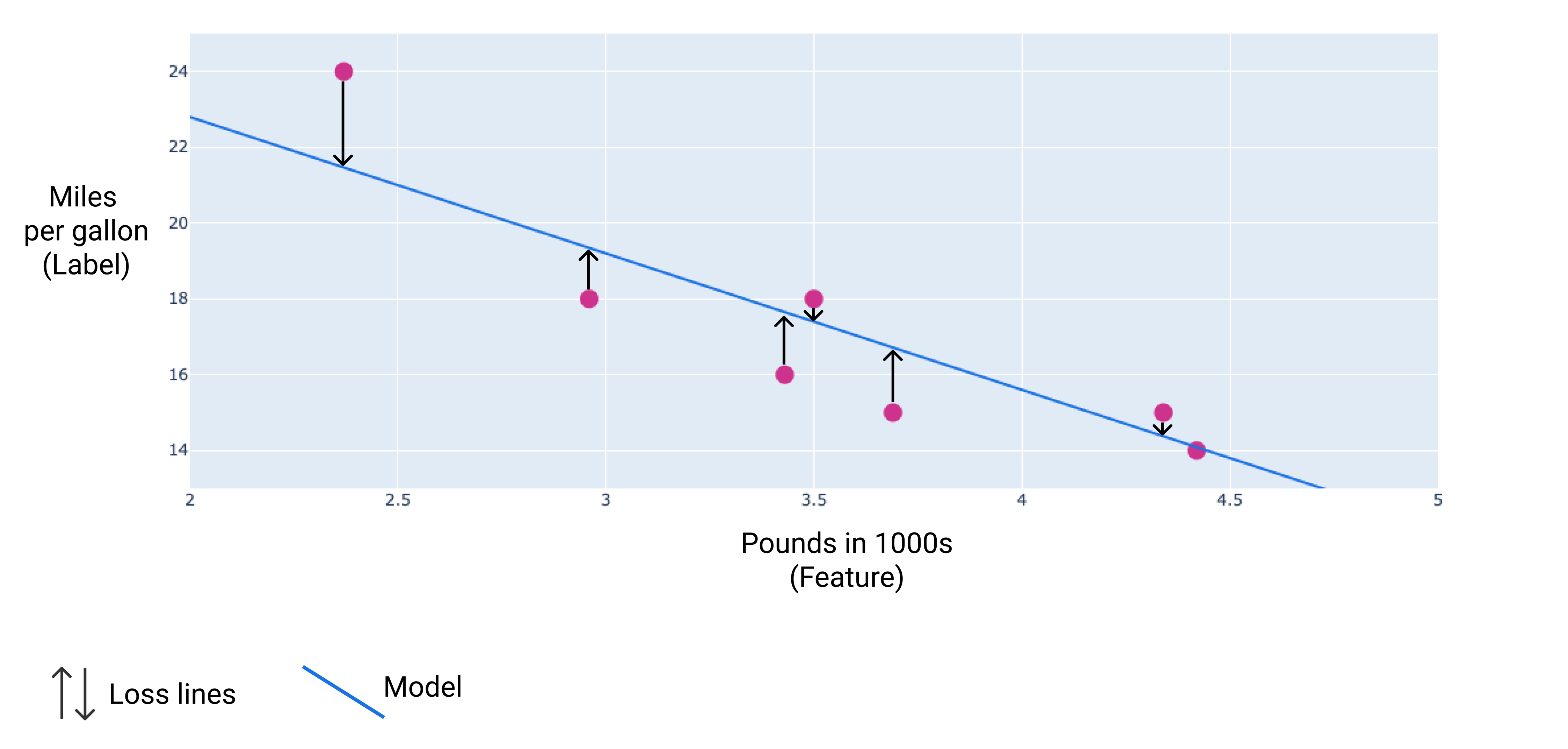
Loss (error):

- how far is the prediction (line) from the actual value?
- how wrong a model's prdictions are.

Cost:

- average of losses over all data points

Loss



Miles
per gallon
(Label)
24
22
20
18
16
14
2
2.5
3
3.5
4
4.5
5
Pounds in 1000s
(Feature)
Loss lines
Model

Types of loss (error) functions

L1 loss: $|y - \hat{y}|$

L2 loss: $(y - \hat{y})^2$

Mean absolute error (MAE): $\frac{1}{m}\sum|y - \hat{y}|$

Mean squared error (MSE): $\frac{1}{m}\sum(y - \hat{y})^2$

Cost functions

Average of losses over all data points

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, f(x_i))$$

e.g., mean squared error (MSE)

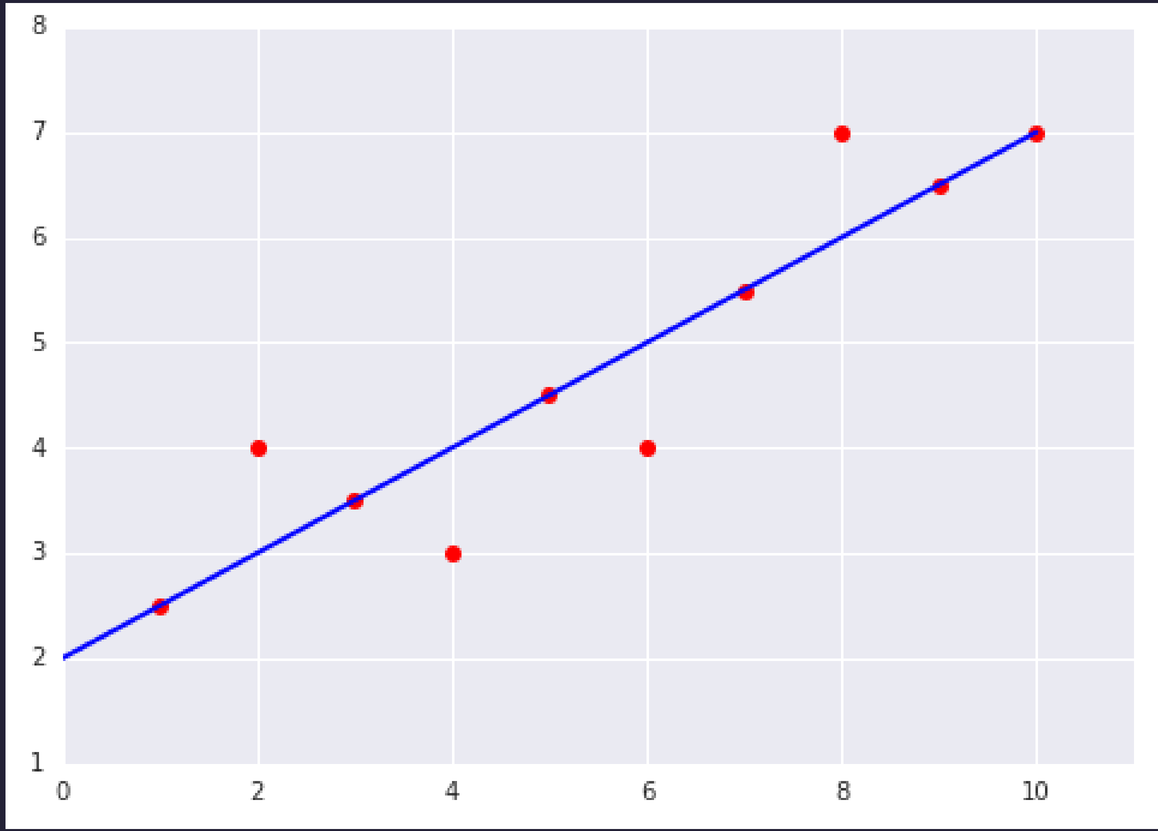
$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} (y_i - f(x_i))^2$$

Calculating cost function (MSE)

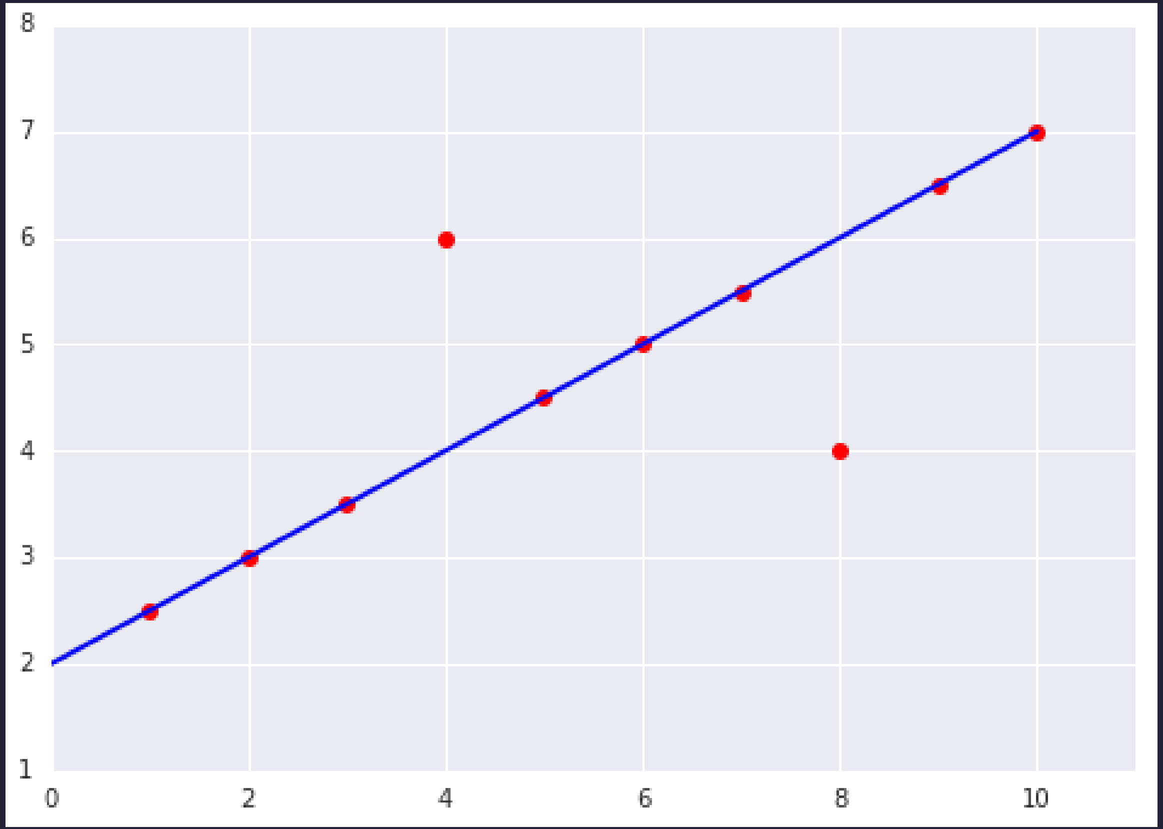
$f(x) = 2x + 1$

$J(w, b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - 2x - 1)^2$

x	y	f(x)	y - f(x)	(y - f(x))^2
1	2	3	-1	1
2	3	5	-2	4
3	4	7	-3	9
sum				14
avg (MSE)				4.67



8
7
6
5
4
3
2
1
0
2
4
6
8
10



8
7
6
5
4
3
2
1
0
2
4
6
8
10

"Linear" regression

model

$$f(x) = wx + b$$

parameters

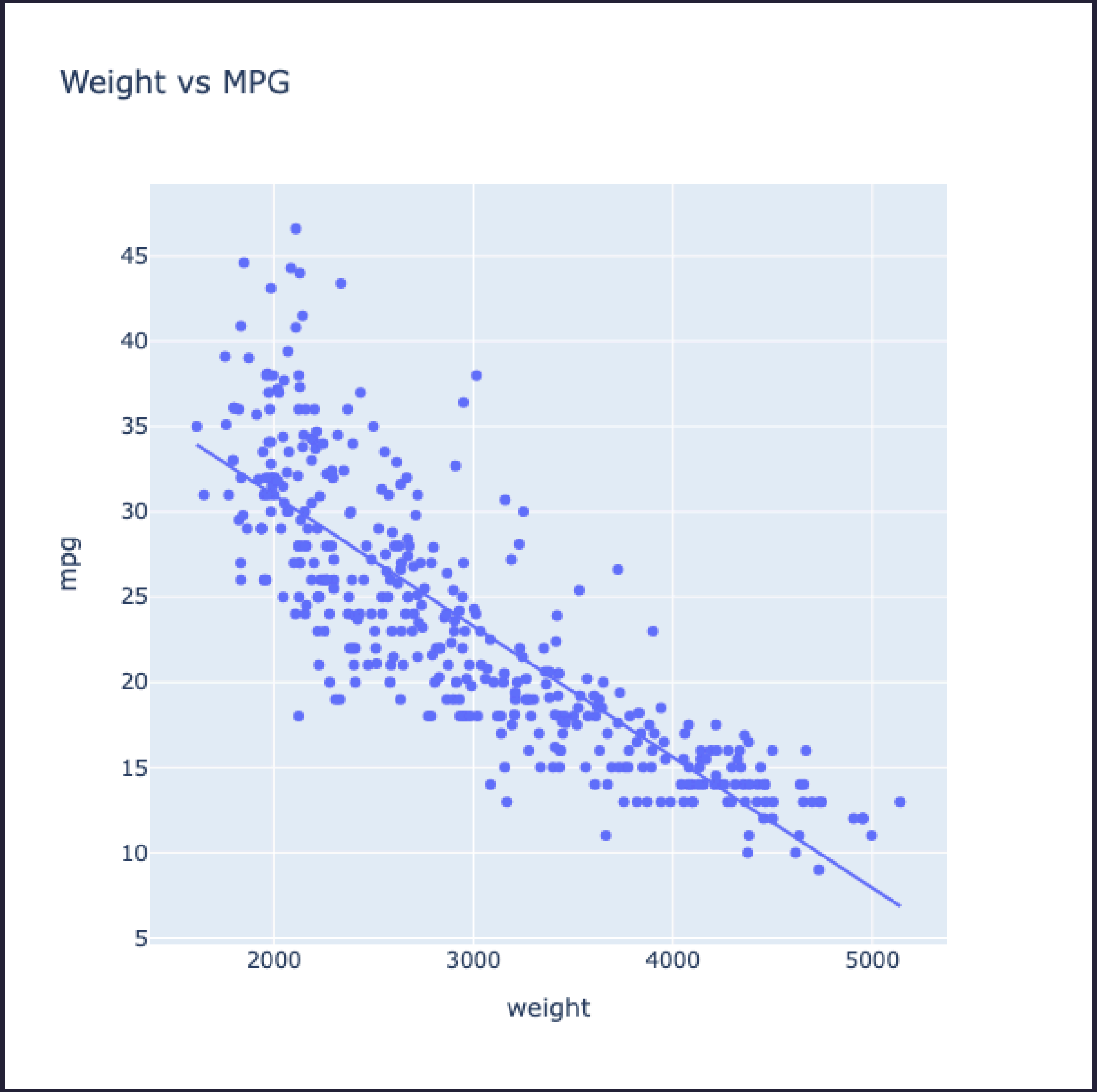
w, b

cost function

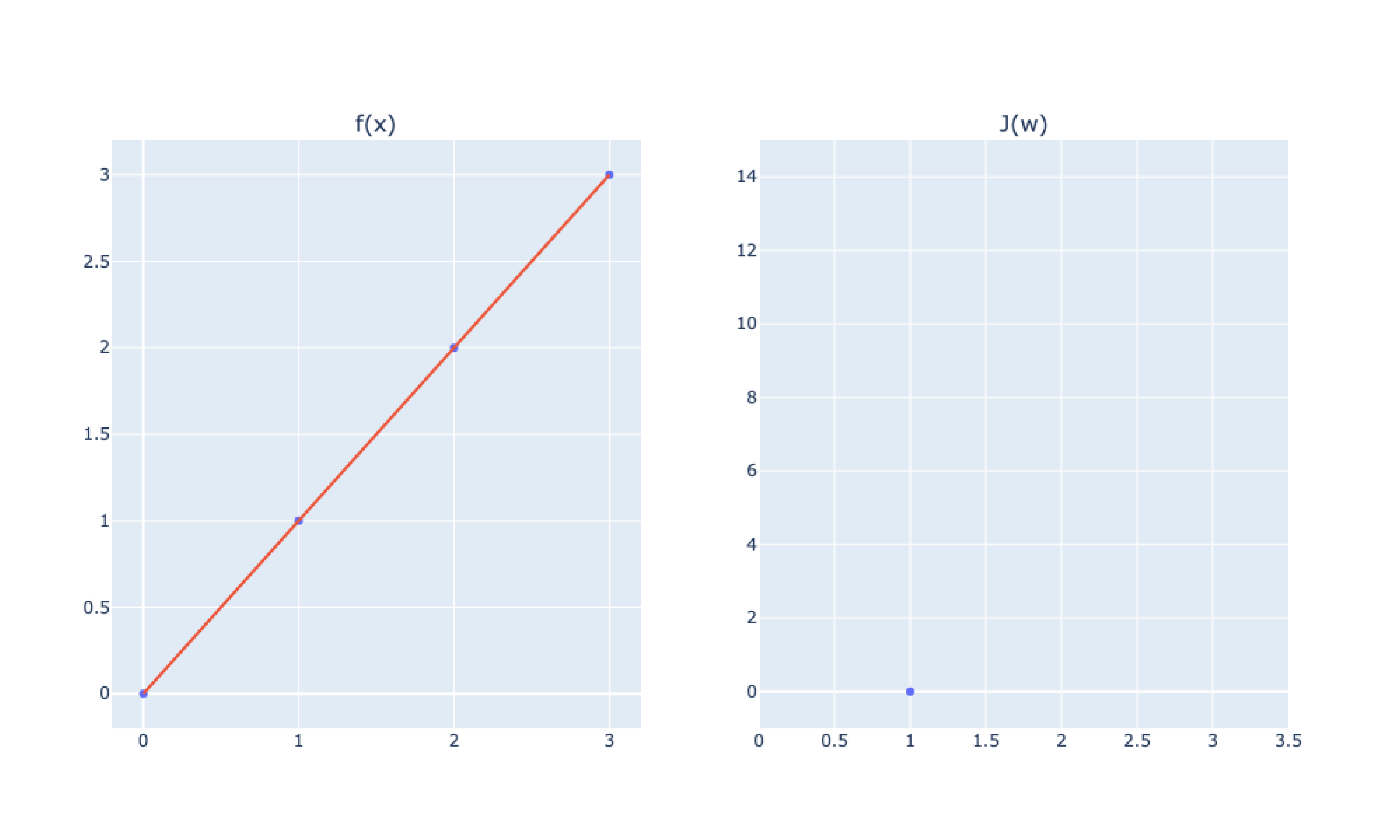
$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} (y_i - f(x_i))^2$$

goal

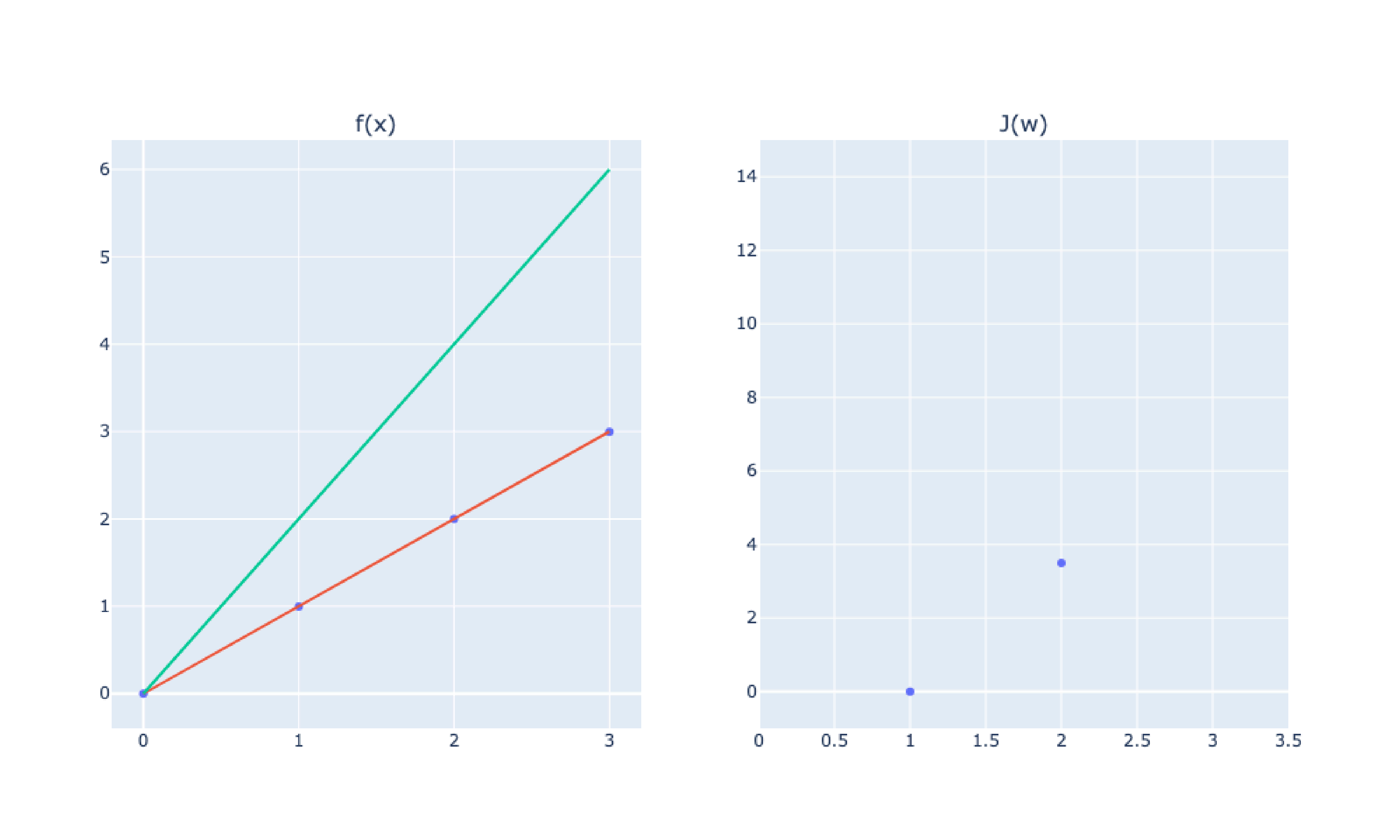
find w and b that minimize $J(w, b)$



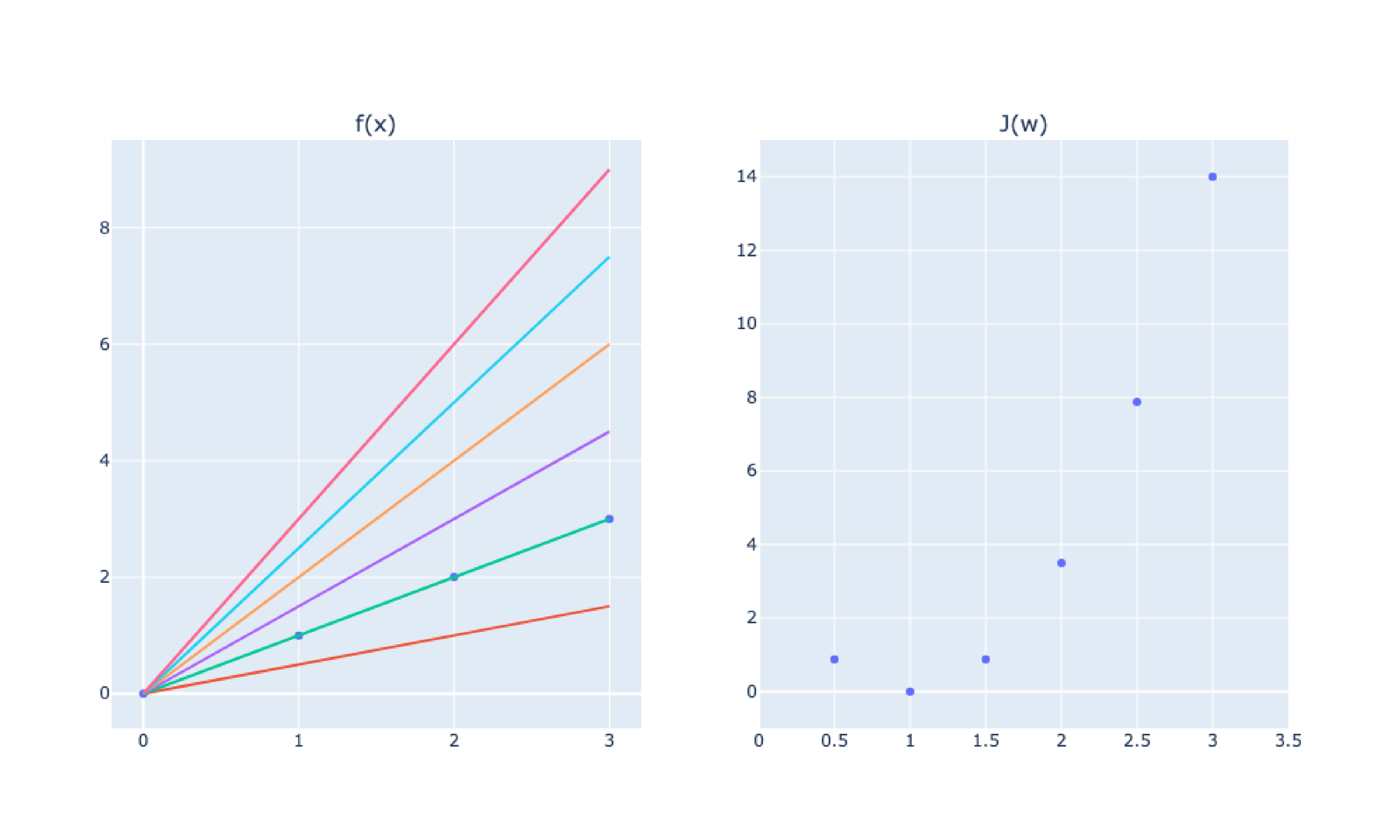
Cost function (w=1, b=0)



Cost function (w=2, b=0)



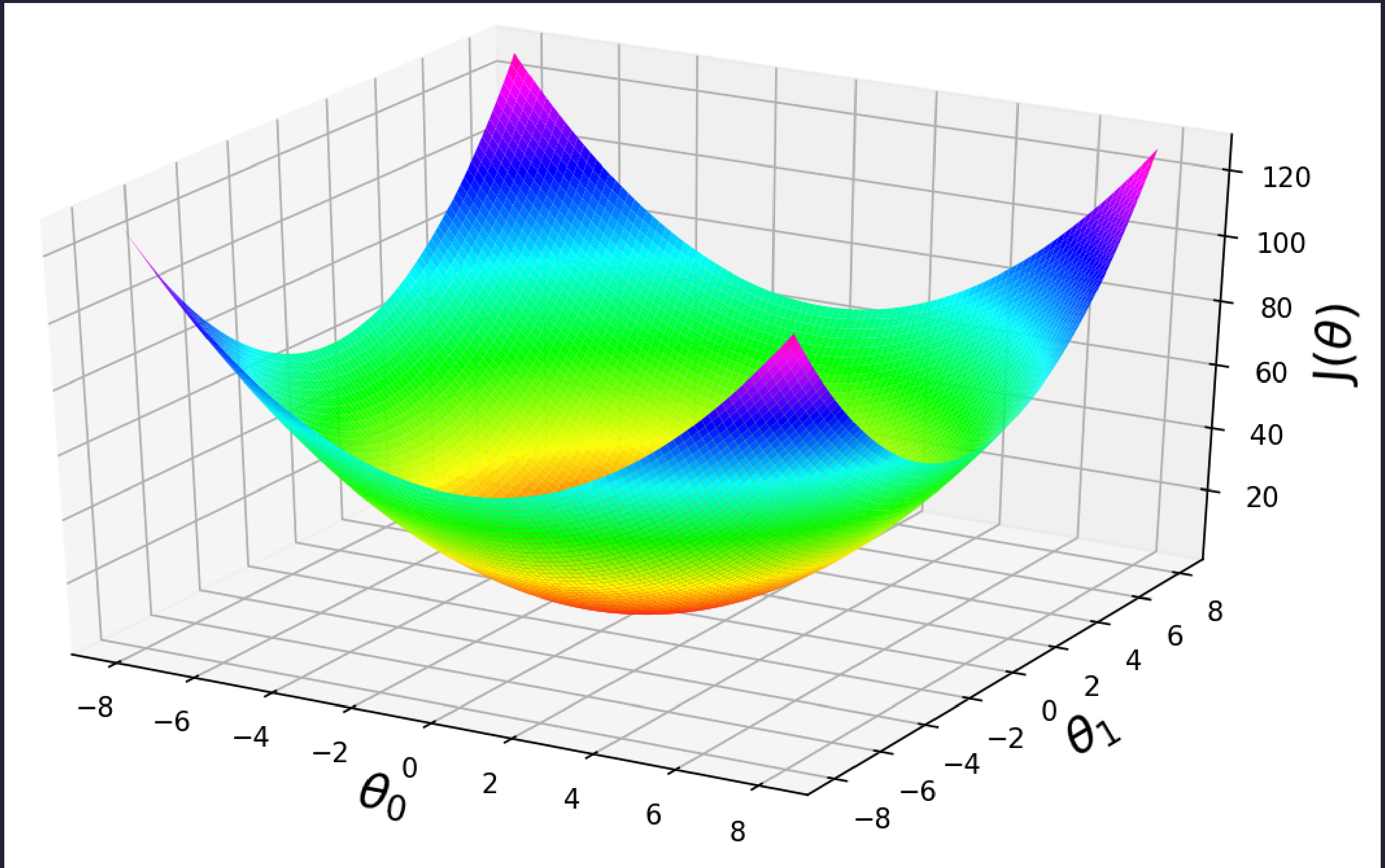
Cost function



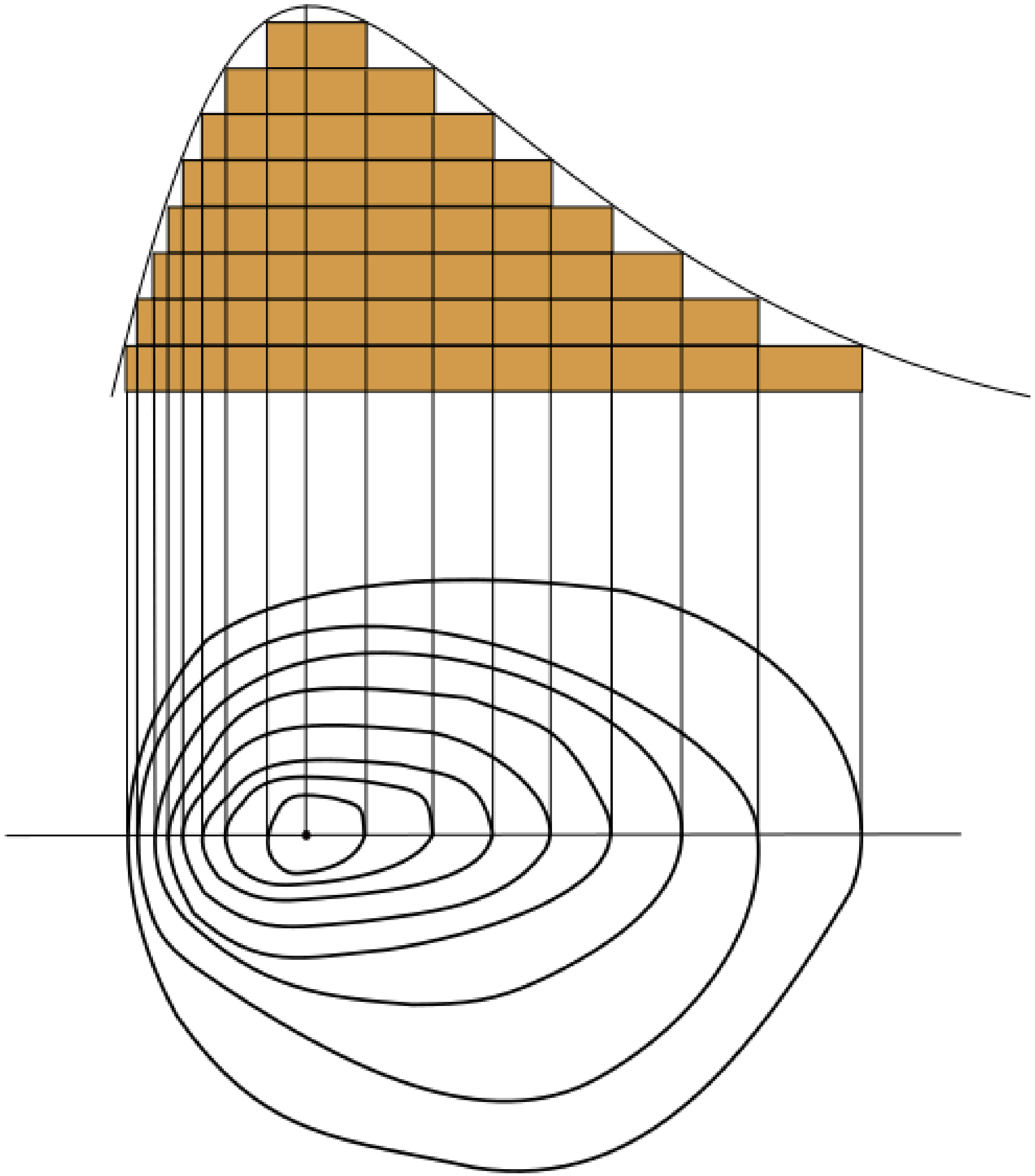
Visualizing cost function

https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/linear-regression/parameters-exercise_3203fed55106e7533d661b3b25a12752a390a085ee793c54c516a1e855787905.frame

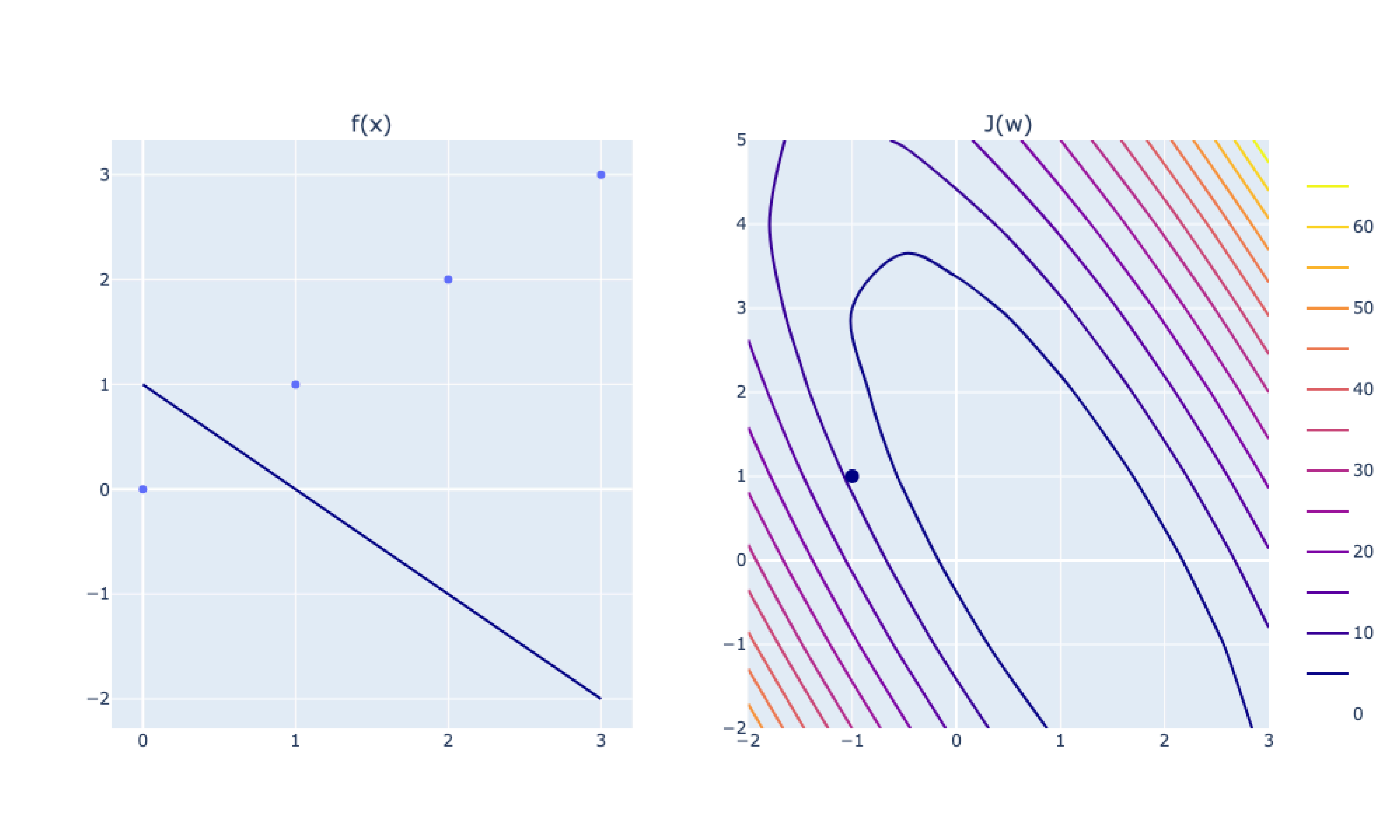
Cost function for two parameters (w,b)



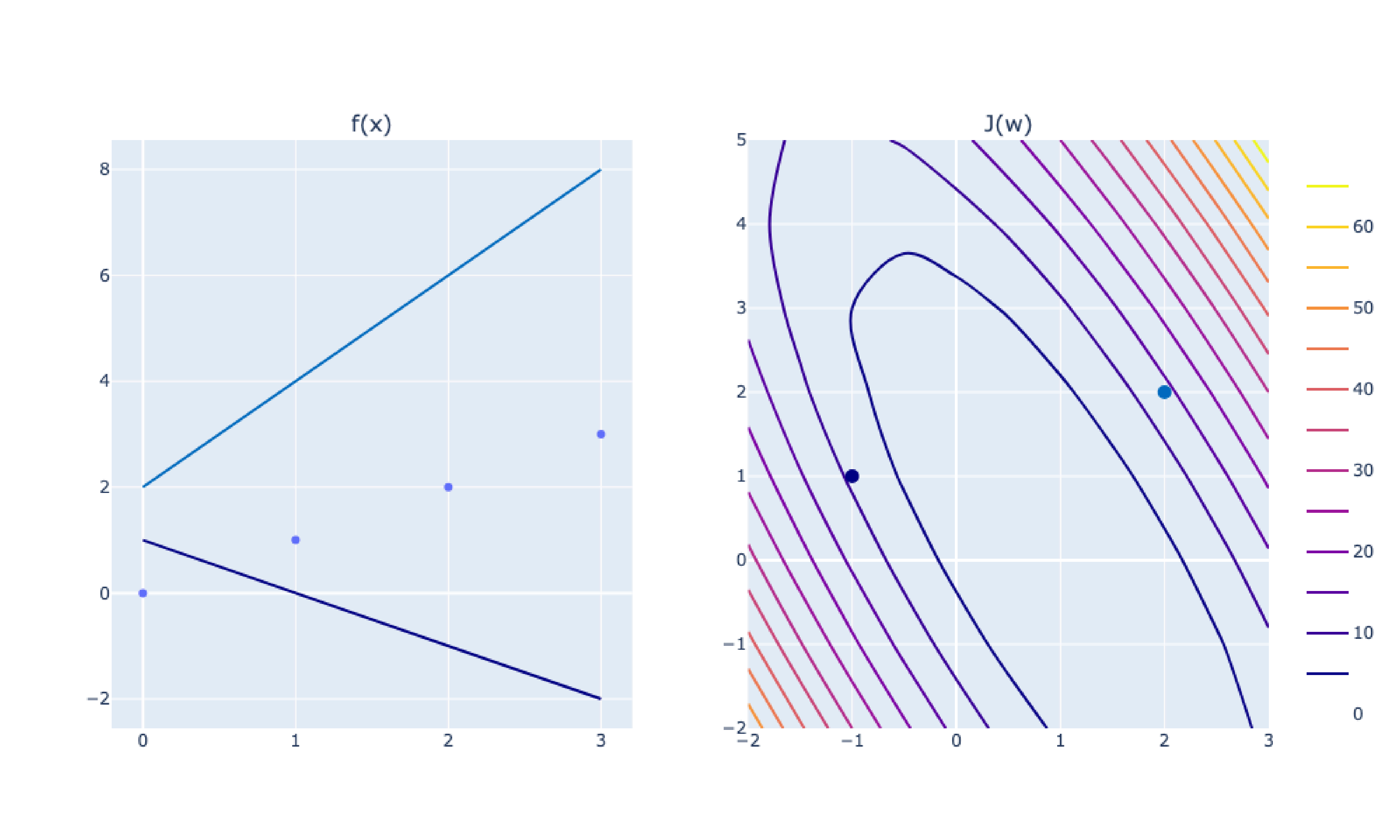
Contour lines



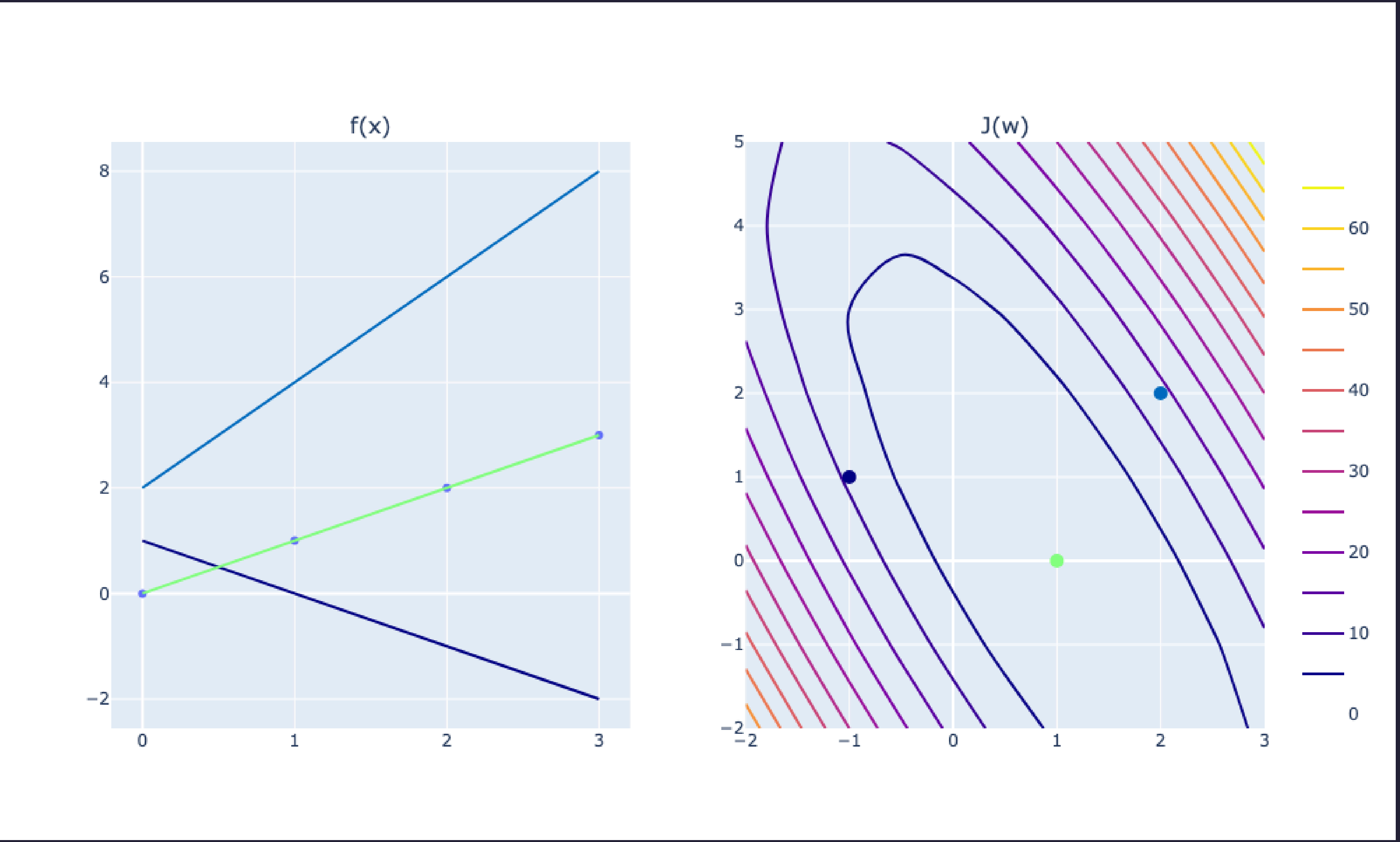
Cost function (w=-1, b=1)



Cost function (w=2, b=2)



Cost function (w=1, b=0)



💻 Cost function

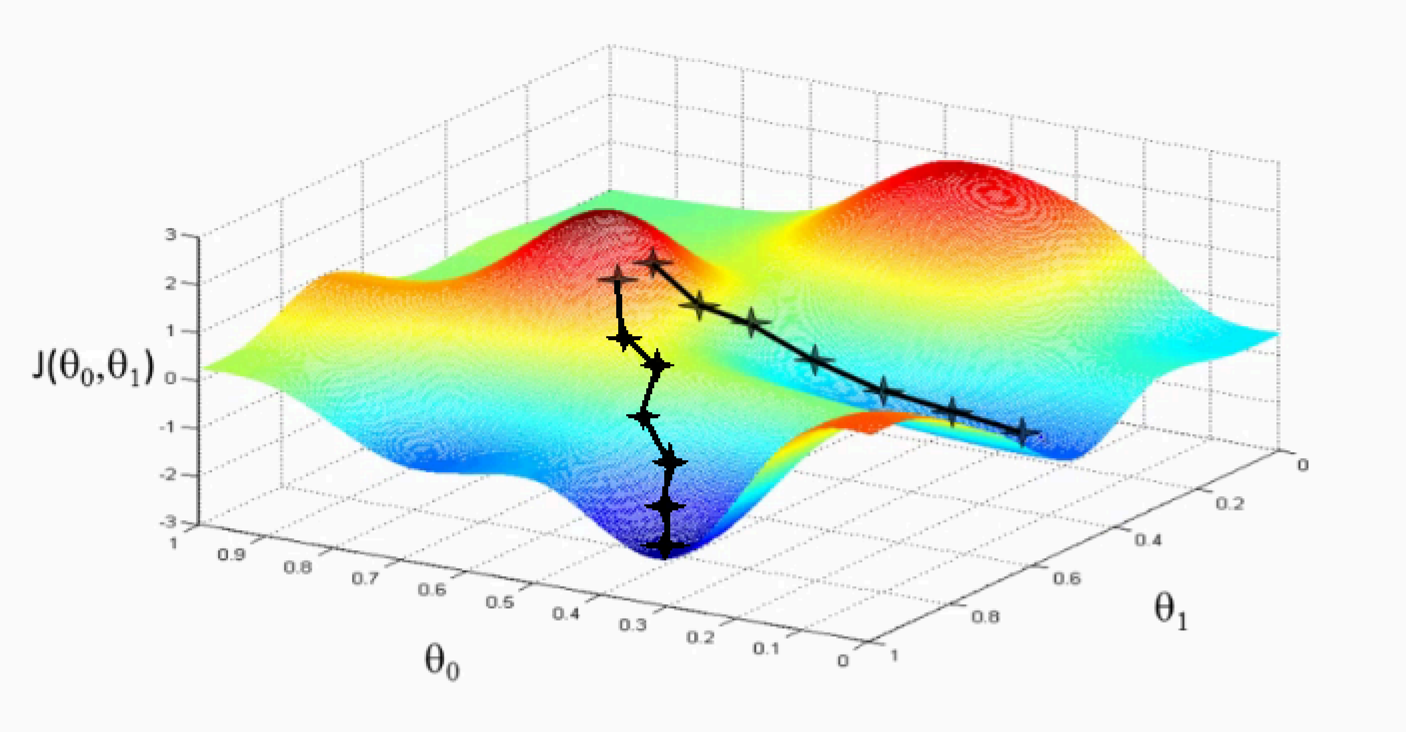
How to find the best model?

Gradient descent

Want to minimize the cost function $J(w, b)$

1. Start with some w, b.
2. Keep chaining w, b to reduce $J(w, b)$.
3. Until we can't reduce $J(w, b)$ any further.

Walking down the hill as quickly as possible



Gradient descent algorithm

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

$$b = b - \alpha \frac{\partial J(w, b)}{\partial b}$$

α: learning rate

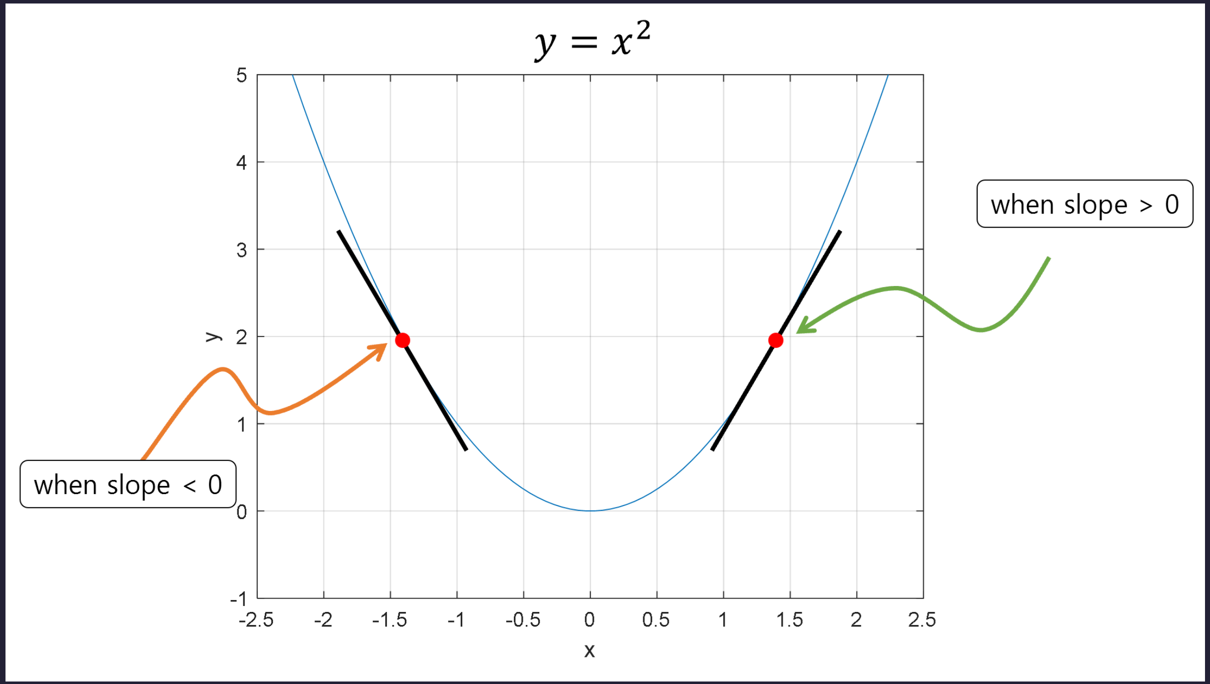
$\frac{\partial J(w,b)}{\partial w}$: partial derivative (gradient) of the cost function with respect to w

$\frac{\partial J(w,b)}{\partial b}$: partial derivative (gradient) of the cost function with respect to b

Gradient "descent" (positive slope)

$$\alpha > 0, \frac{\partial J(w,b)}{\partial w} > 0$$

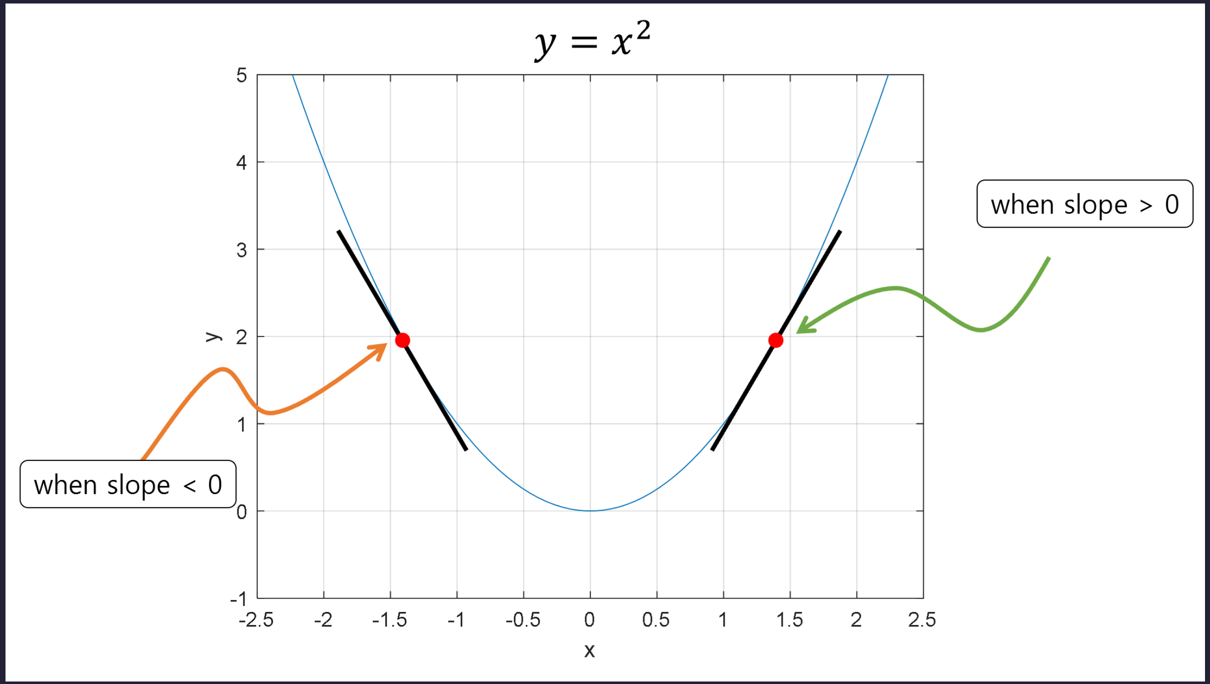
$$w = w - \alpha \frac{\partial J(w,b)}{\partial w} = w - positive\, number$$



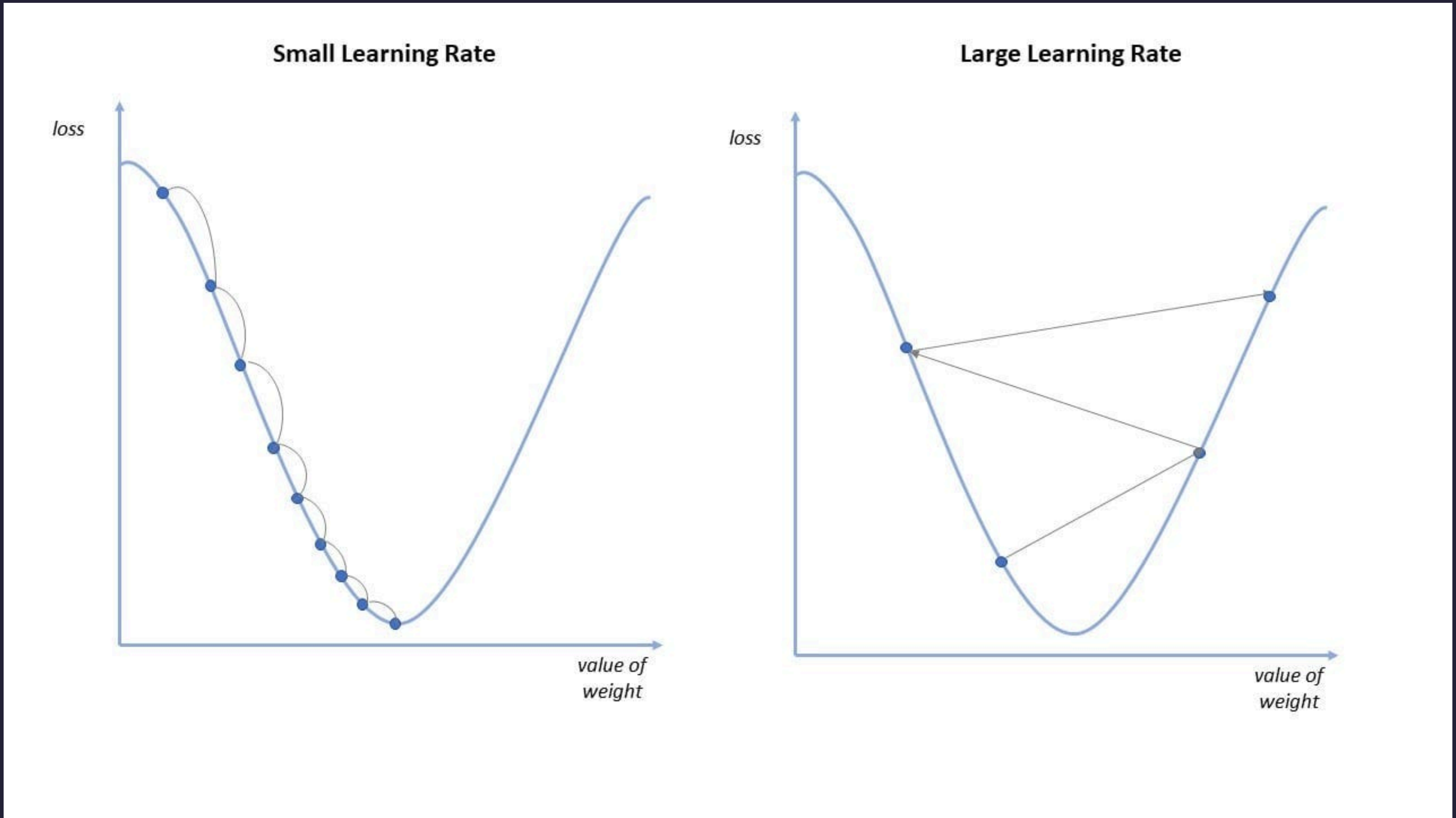
Gradient "descent" (negative slope)

$$\alpha > 0, \frac{\partial J(w,b)}{\partial w} < 0$$

$$w = w - \alpha \frac{\partial J(w,b)}{\partial w} = w - negative\, number$$



Learning rate (α)



Convergence: reaching the minimum

$$w = w - \alpha \frac{\partial J(w, b)}{\partial w}$$

Near a minimum

- gradient is close to zero
- learning rate is small
- no significant change in w (convergence)

Gradient for linear regression

$f(x) = wx + b$

$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2$

Partial derivative of the cost function with respect to w

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))\frac{\partial f(x_i)}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))(-x_i)$$

Partial derivative of the cost function with respect to b

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))\frac{\partial f(x_i)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))(-1)$$

🤔 Calculate gradient

$$X = [1, 2],\ y = [2, 2.5]$$

$$f(x) = wx + b$$

$$w = 0, b = 0, \alpha = 0.01$$

🤔 Calculate gradient

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))(-x_i) = -\frac{2}{2}((2-0)*1 + (2.5-0)*2) = -7$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2(y_i - f(x_i))(-1) = -\frac{2}{2}((2-0) + (2.5-0)) = -4.5$$

$$w = w - \alpha\frac{\partial J(w,b)}{\partial w} = 0 - 0.01*(-7) = 0.07$$

$$b = b - \alpha\frac{\partial J(w,b)}{\partial b} = 0 - 0.01*(-4.5) = 0.045$$

$(0,0) \to (0.07, 0.045)$

Visualizing gradient descent

https://www.benfrederickson.com/numerical-optimization/index.html#gd

💻 Gradient descent

Learning curve: cost function over iterations

For a small learning rate

- the cost function should decrease on every iteration
- but slow

For a large learning rate

- it might not converge
- but if it does, it takes fewer iterations

flat, noisy, or increasing learning curve may indicate a problem

Learning rate demo

https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/linear-regression/gradient-descent-exercise_d1f3f99d99ebad2d51be1d20911fcf707d8537cd2723b99e9dd7e5c78fce85ac.frame

Batch vs. Stochastic vs. Mini-batch gradient descent

- **Batch**: calculate the gradient of the cost function with respect to the parameters for ***the entire training dataset***
- **Stochastic**: calculate the gradient of the cost function with respect to the parameters for ***a single data point***.
- **Mini-batch**: calculate the gradient of the cost function with respect to the parameters for ***a small subset of the training dataset***.