Week 1

# Machine learning

"Field of study that gives computers the ability to learn without being explicitly programmed."

Arthur Samuel (1959)



computers the ability to learn

# Machine learning algorithms

rapid advancements

used most in real-world applications

- Supervised learning ← course 1, 2

- Unsupervised learning ←
- Recommender systems
- Reinforcement learning

course 3

The other thing we're going to spend a lot of

# Supervised learning

Learns from being given "right answers"

**Regression**

Predict a number

infinitely many possible outputs

**Classification**

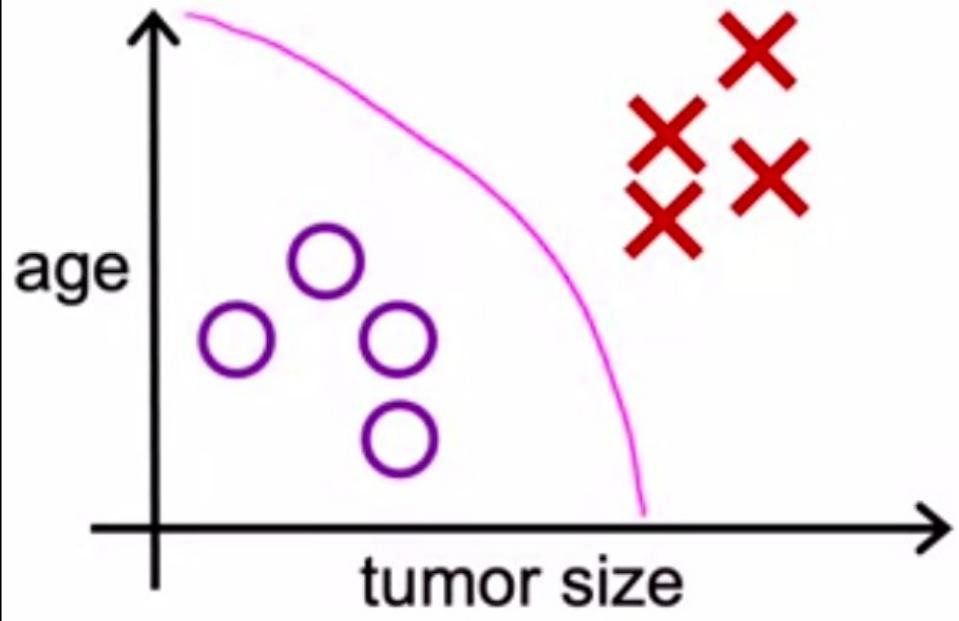
predict categories

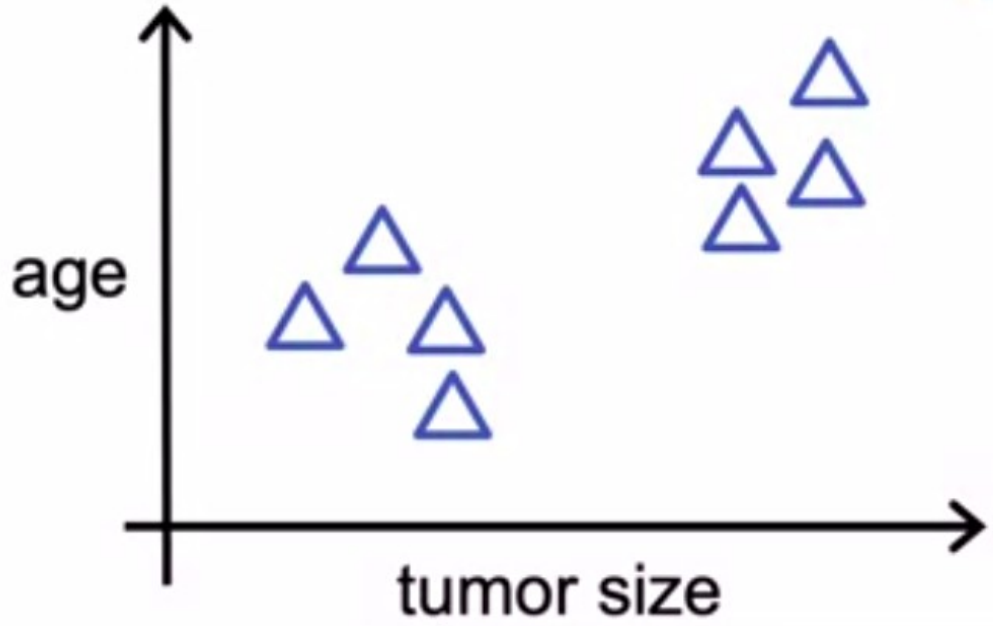
small number of possible outputs

a category,
all of a small set of possible outputs.

Supervised learning
Learn from data labeled with the "right answers"

Unsupervised learning
Find something interesting in unlabeled data.




This is unsupervised learning,

# Unsupervised learning

Data only comes with inputs $x$, but not output labels $y$.
Algorithm has to find structure in the data.

Clustering
Group similar data points together.

Dimensionality reduction
Compress data using fewer numbers.

Anomaly detection
Find unusual data points.

to make too much sense to you yet.

# Practice quiz: Supervised vs unsupervised learning

**Total points** 2

**1.** Which are the two common types of supervised learning? (Choose two) **1 point**

- [x] Regression
- [x] Classification
- [ ] Clustering

**2.** **1 point**

Which of these is a type of unsupervised learning?

- ( ) Regression
- (•) Clustering
- ( ) Classification

# Terminology

Training set: Data used to train the model

| | $x$ size in feet² | $y$ price in $1000's |
|---|---|---|
| (1) | 2104 | 400 |
| (2) | 1416 | 232 |
| (3) | 1534 | 315 |
| (4) | 852 | 178 |
| … | … | … |
| (47) | 3210 | 870 |

$m = 47$

$x^{(1)} = 2104$ $y^{(1)} = 400$

$(x^{(1)}, y^{(1)}) = (2104, 400)$

$x^{(2)} = 1416$ $x^{(2)} \neq x^2$ not exponent

Notation:

$x$ = "input" variable
feature

$y$ = "output" variable
"target" variable

$m$ = number of training examples

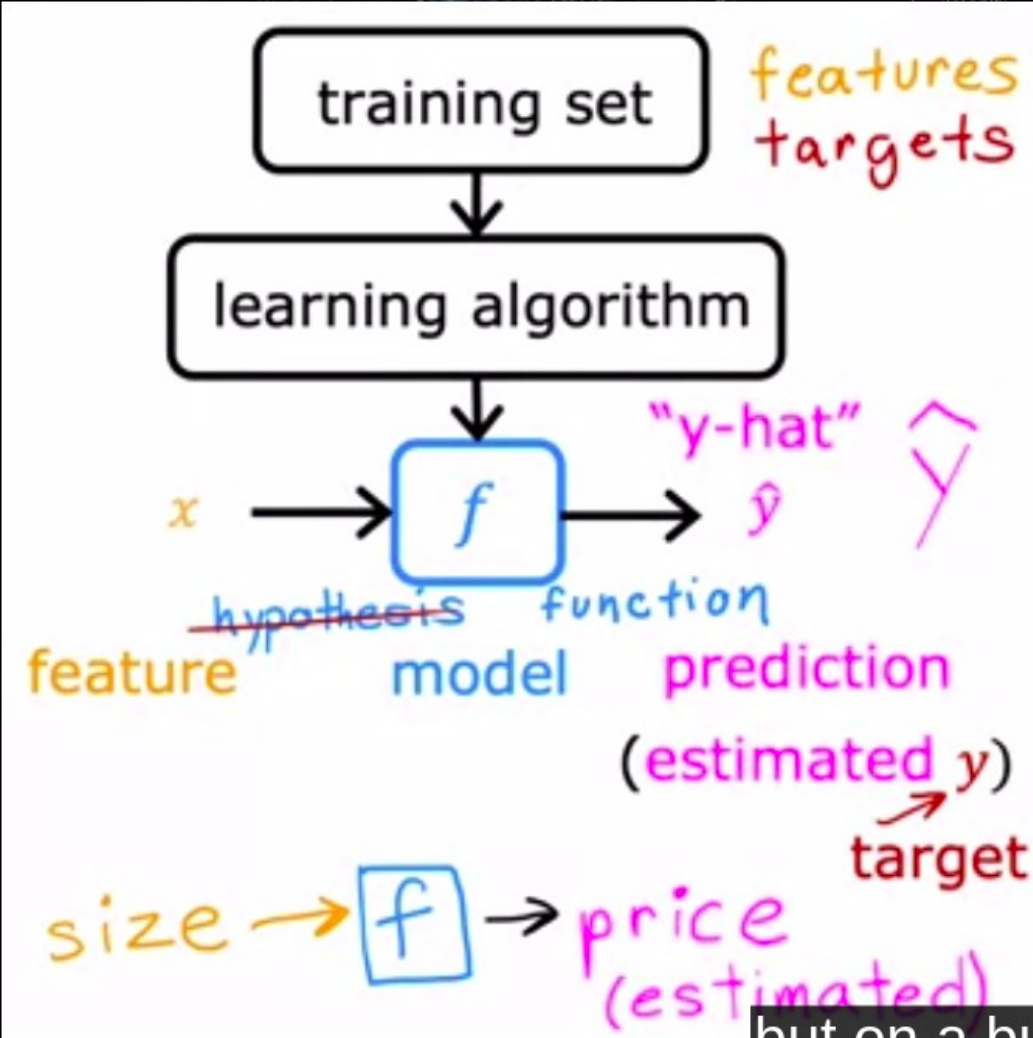
$(x, y)$ = single training example

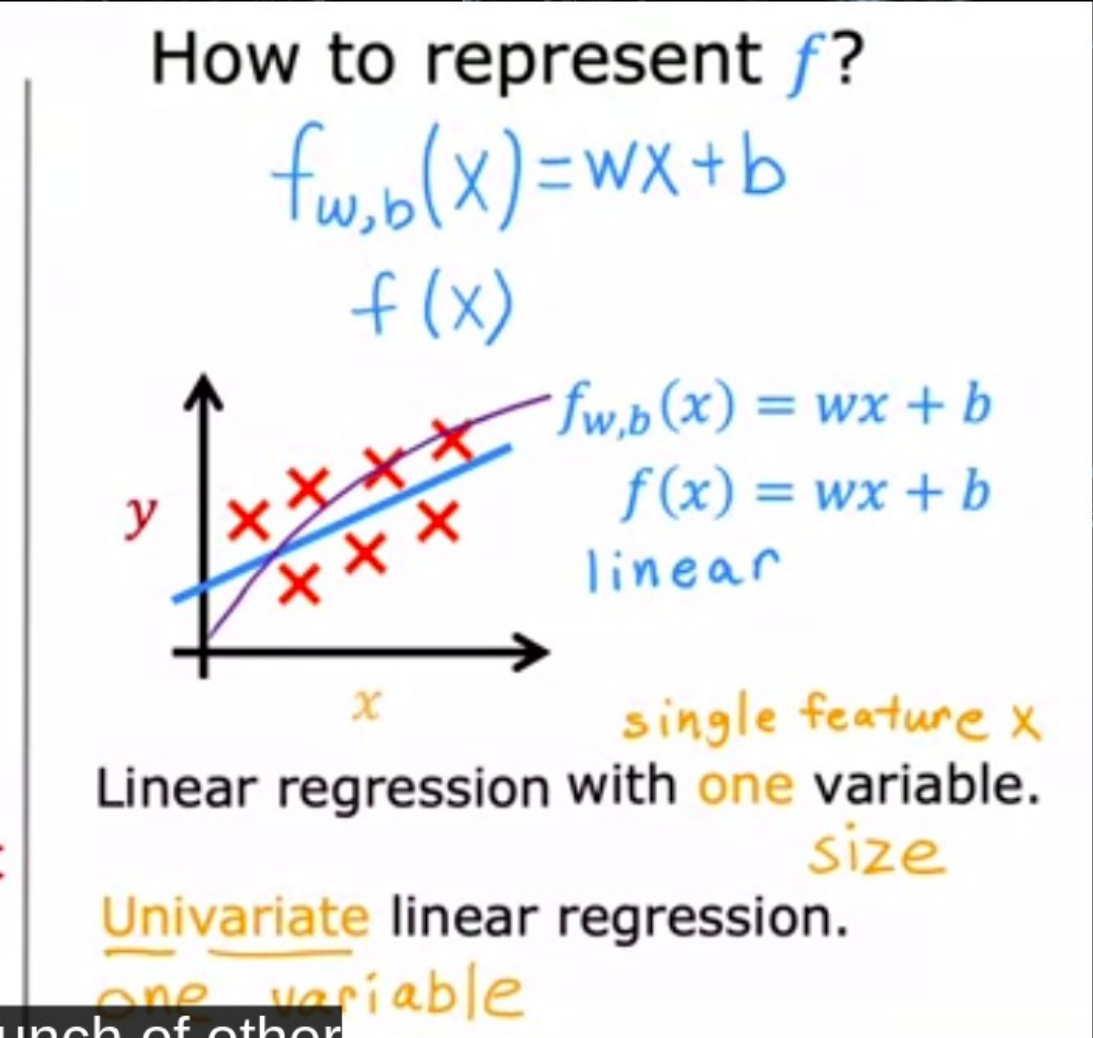
$(x^{(i)}, y^{(i)})$

$(x^{(i)}, y^{(i)})$ = $i^{th}$ training example

index ($1^{st}$, $2^{nd}$, $3^{rd}$ …)

the training set and refers to row i in the table.



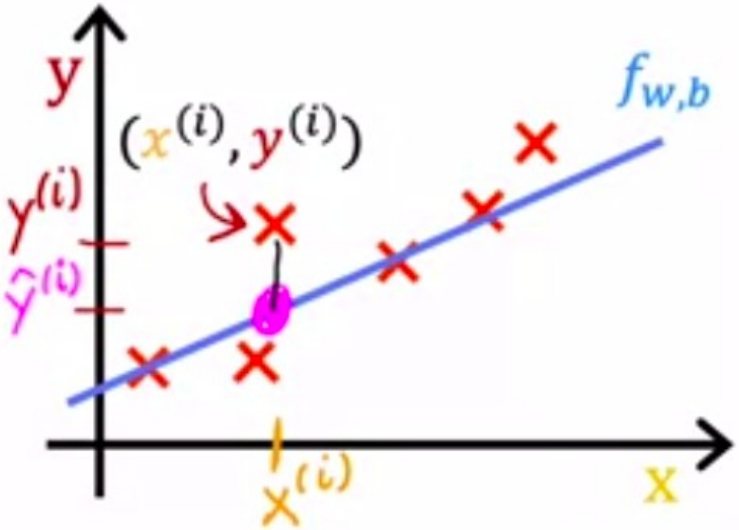

but on a bunch of other
things that you may know

# Notation

Here is a summary of some of the notation you will encounter.

| General Notation | Description | Python (if applicable) |
|---|---|---|
| $a$ | scalar, non bold | |
| $\mathbf{a}$ | vector, bold | |
| **Regression** | | |
| $\mathbf{x}$ | Training Example feature values (in this lab - Size (1000 sqft)) | `x_train` |
| $\mathbf{y}$ | Training Example targets (in this lab Price (1000s of dollars)). | `y_train` |
| $x^{(i)}$, $y^{(i)}$ | $i_{th}$Training Example | `x_i`, `y_i` |
| m | Number of training examples | `m` |
| $w$ | parameter: weight, | `w` |
| $b$ | parameter: bias | `b` |
| $f_{w,b}(x^{(i)})$ | The result of the model evaluation at $x^{(i)}$ parameterized by $w, b$: $f_{w,b}(x^{(i)}) = wx^{(i)} + b$ | `f_wb` |

# Cost function: Squared error cost function




$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$\hat{y}^{(i)}$ ↑ error $y^{(i)}$

m = number of training examples

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

$J(w,b)$ ↑ intuition

Find $w, b$:

$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

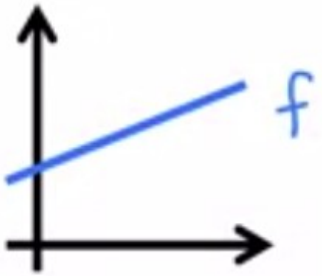
J of wb is really computing.

model:

$f_{w,b}(x) = wx + b$

parameters:

$w, b$




cost function:

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

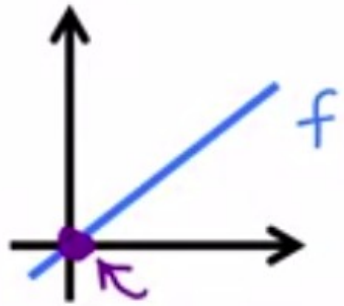
goal:

$$\underset{w,b}{\text{minimize}}\, J(w,b)$$

simplified

$f_w(x) = wx$ $b = \emptyset$

$w$

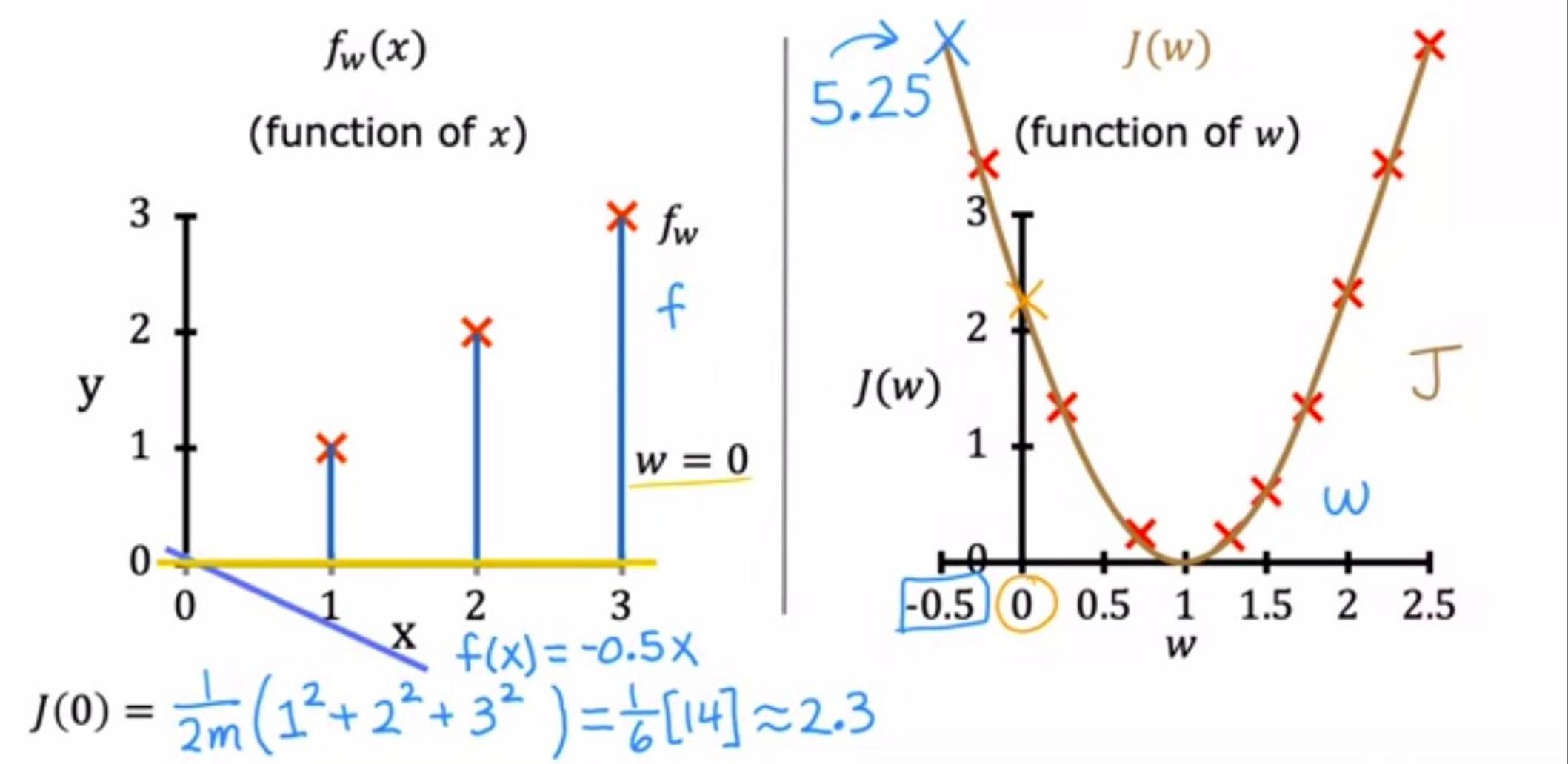



$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$

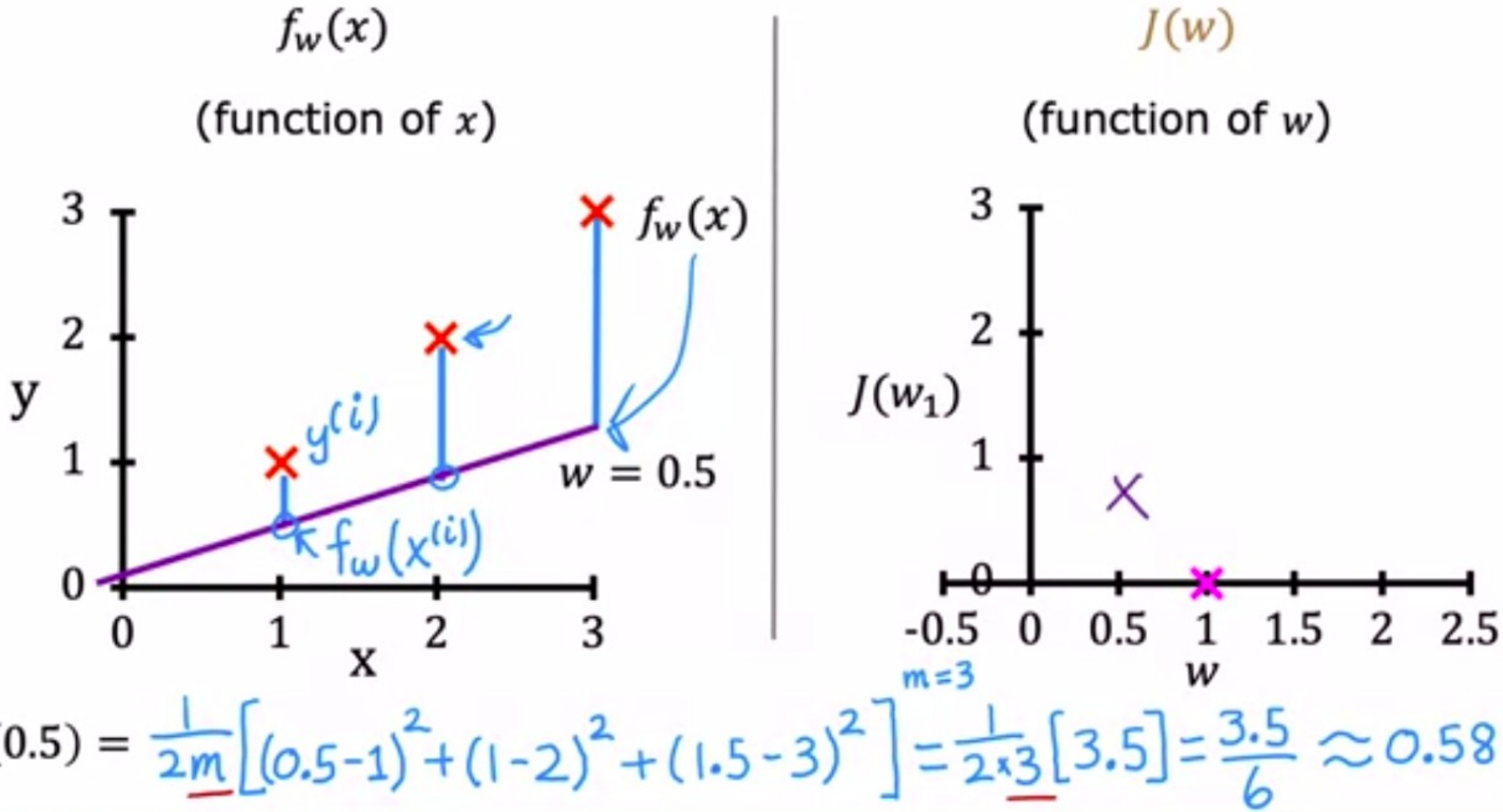
$wx^{(i)}$

$$\underset{w}{\text{minimize}}\, J(w)$$

different values for the parameter w. In particular,

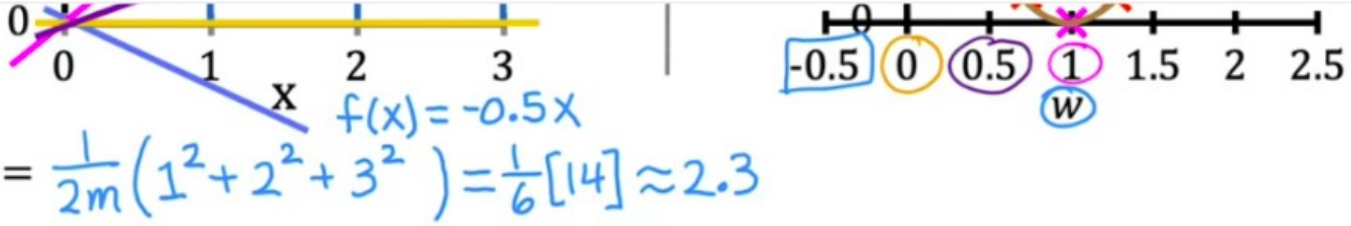



a single point on




w. How about if w equals 0?

## Question



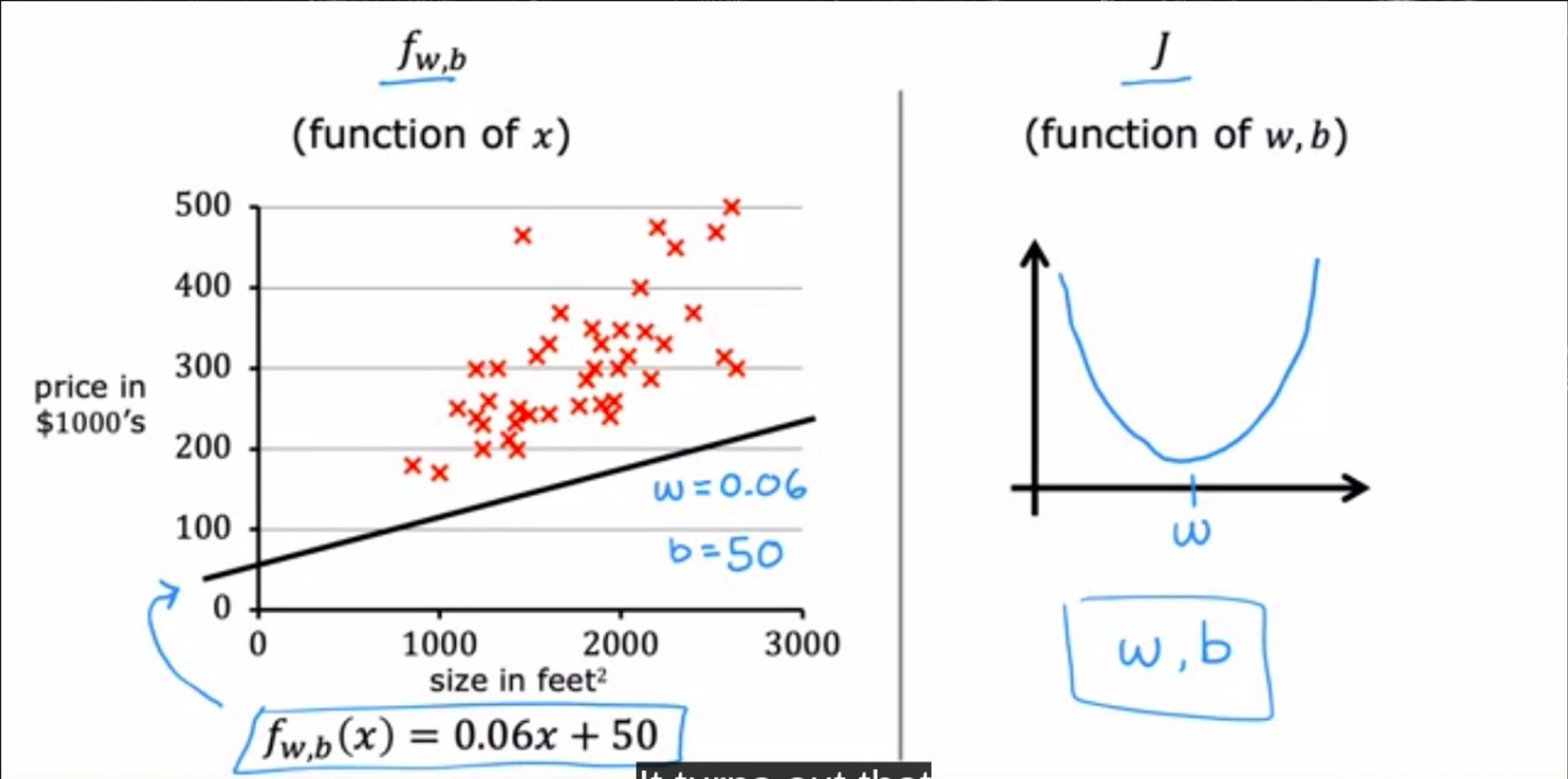

When does the model fit the data relatively well, compared to other choices for parameter w?

- When $x$ is at or near a minimum.
- When w is close to zero.
- When the cost $J$ is at or near a minimum.
- When $f_w(x)$ is at or near a minimum for all the values of x in the training set.

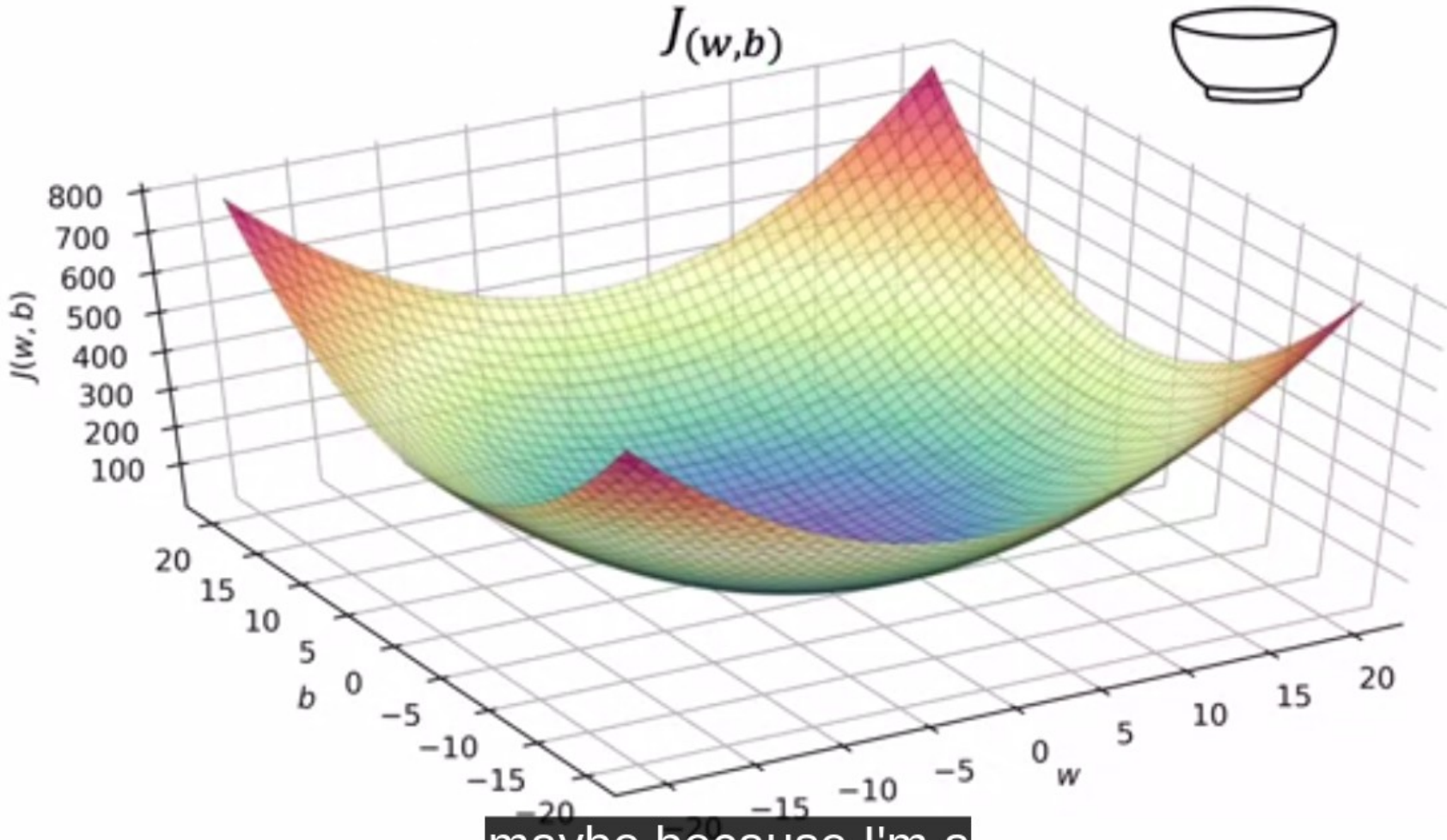
**Correct**

When the cost is relatively small, closer to zero, it means the model fits the data better compared to other choices for w and b.

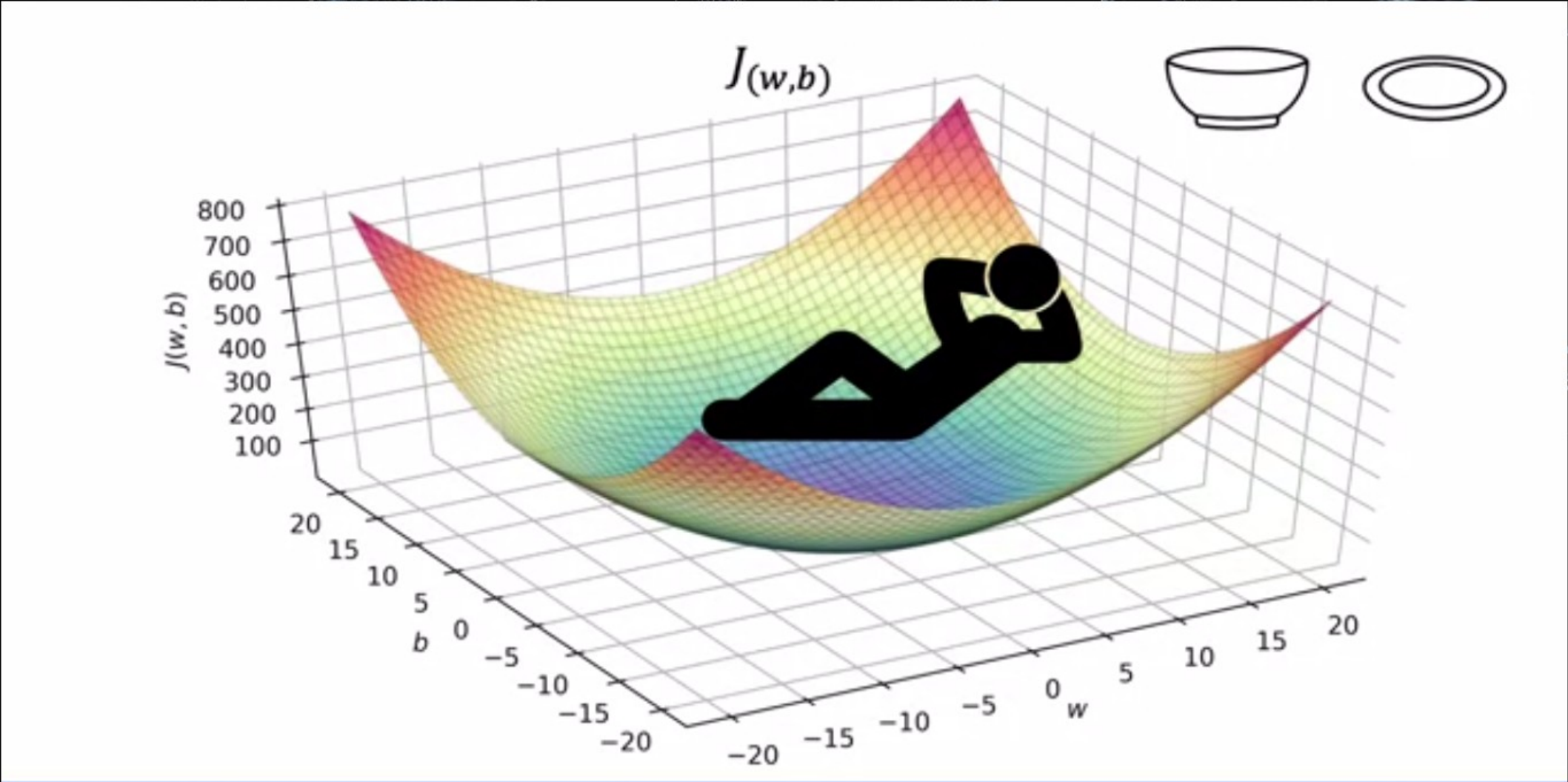
Skip Continue




It turns out that the cost function

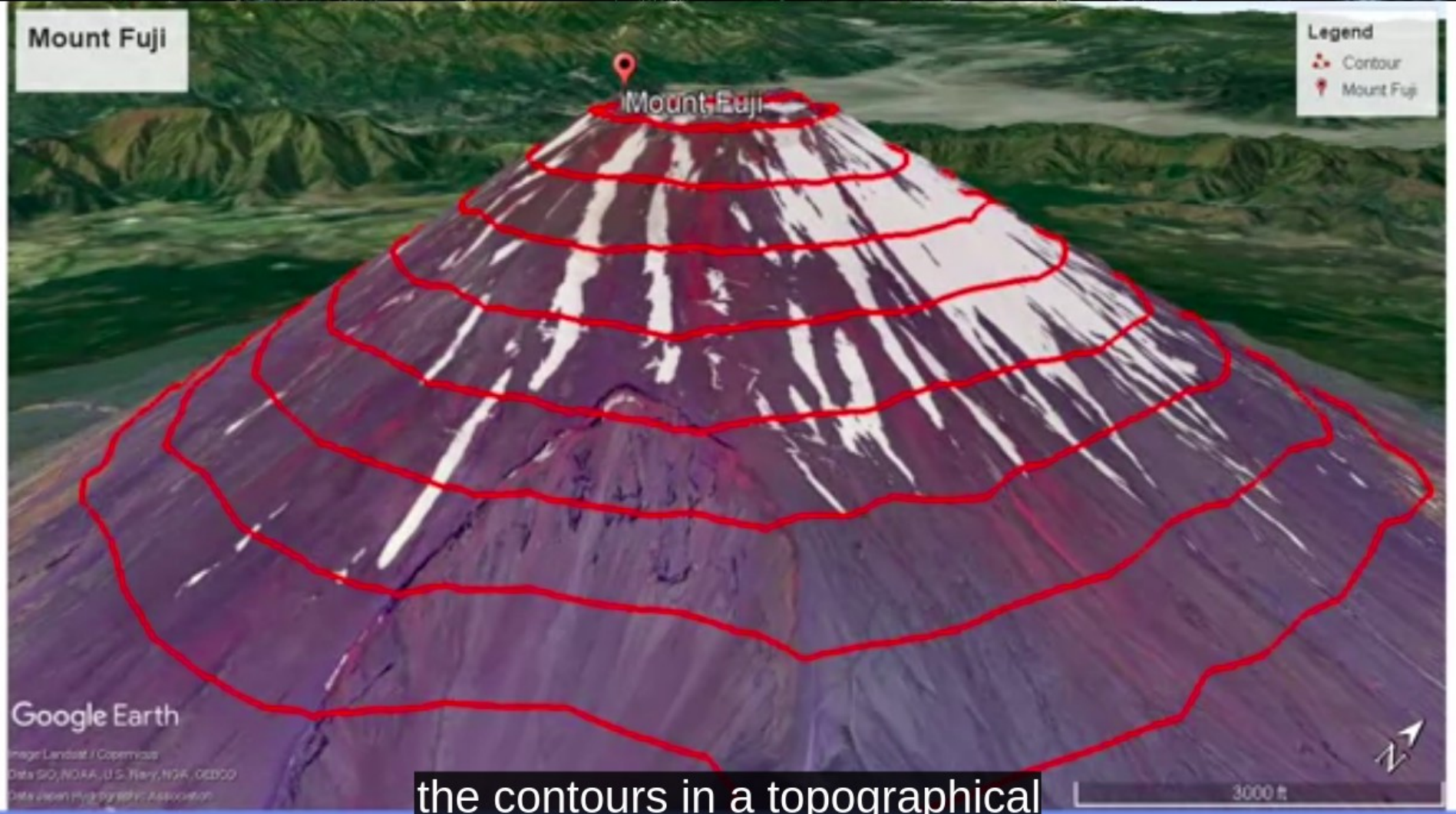



maybe because I'm a
little bit hungry,

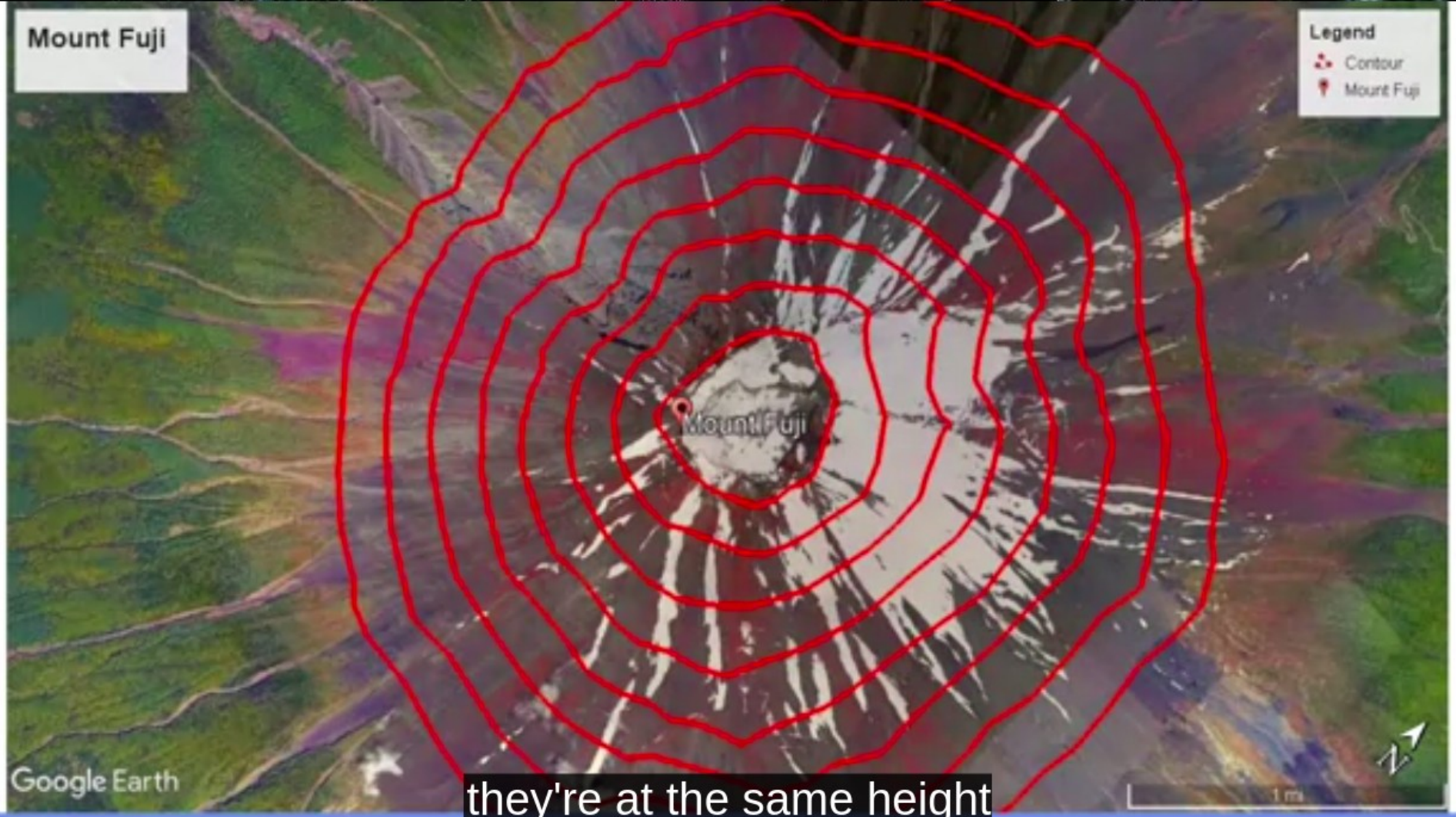




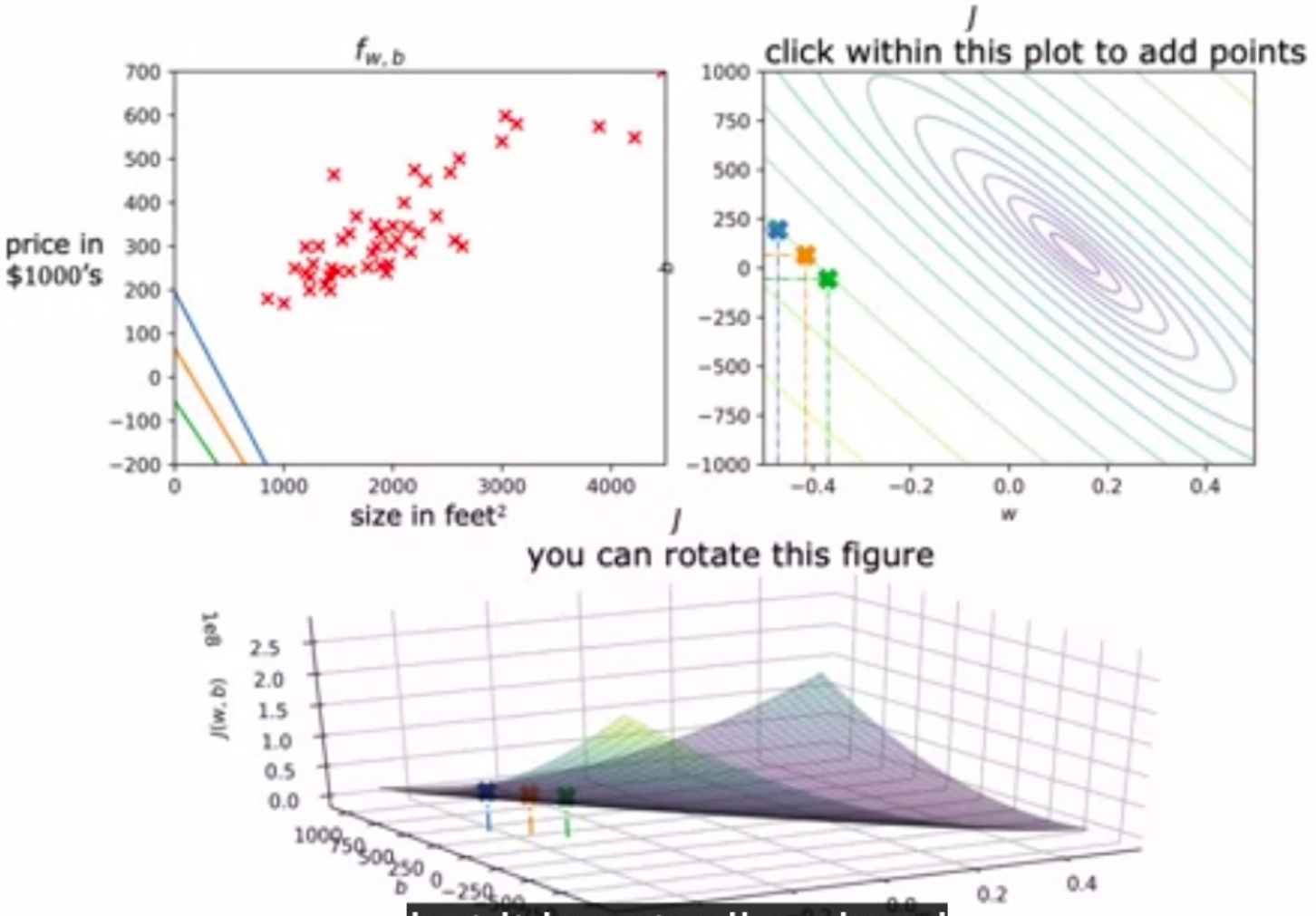

and there's your coconut drink.



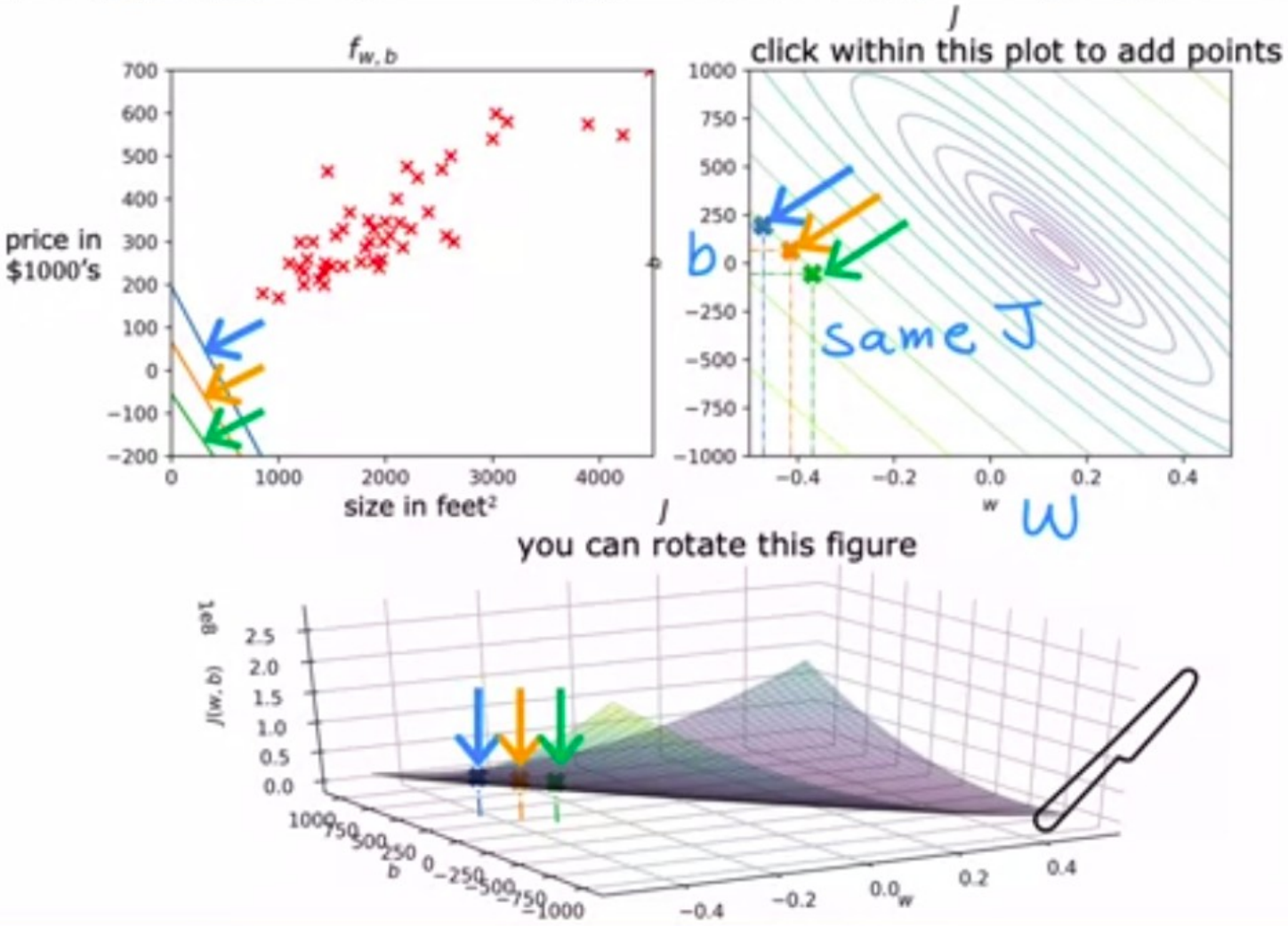

the contours in a topographical map are basically



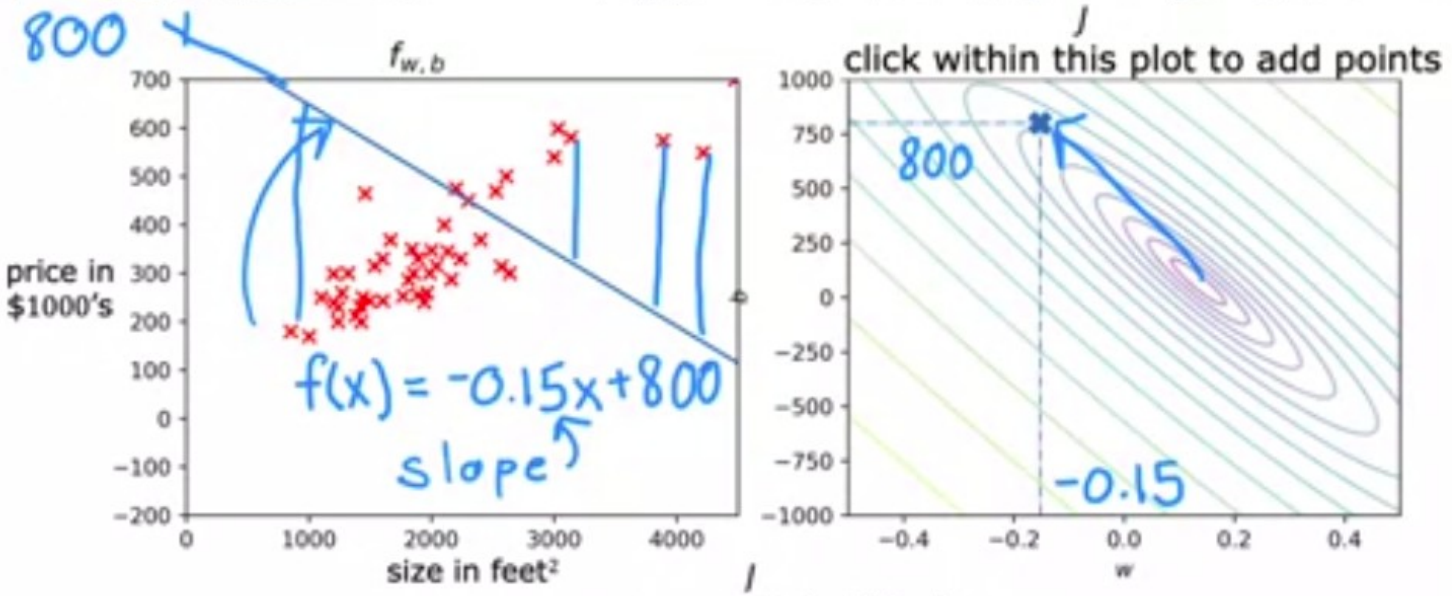

they're at the same height
for different heights.



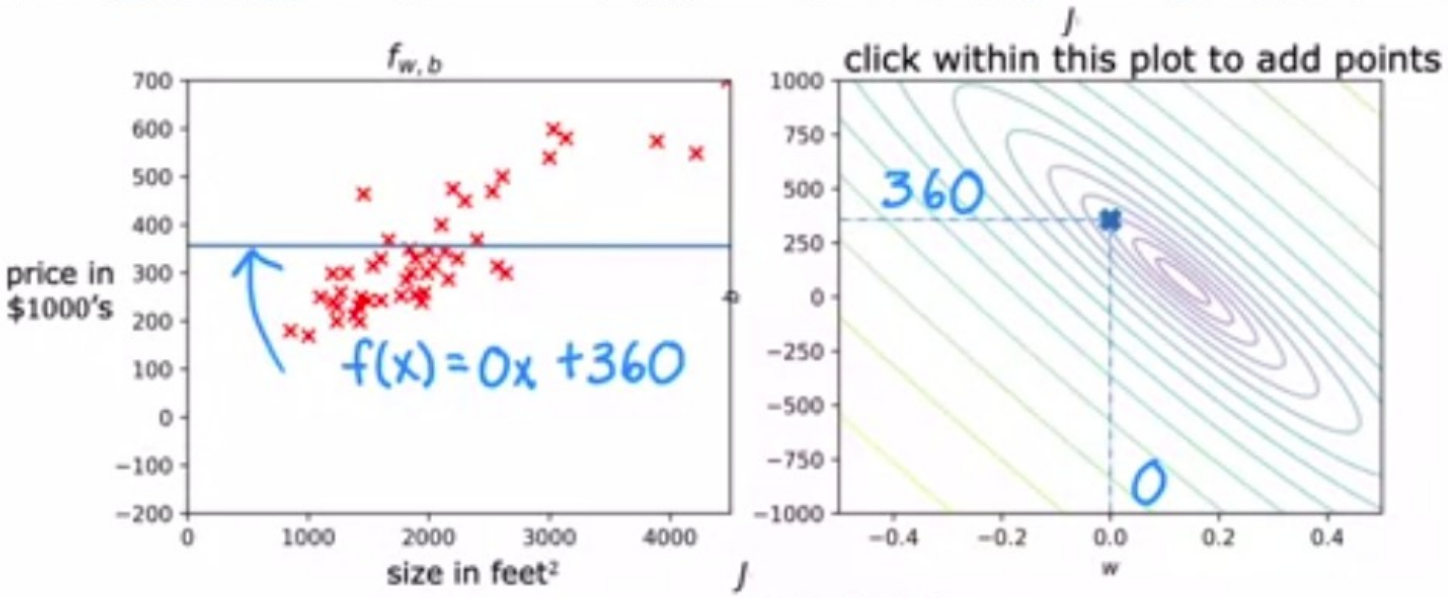

but it is actually a bowl
just very stretched out,



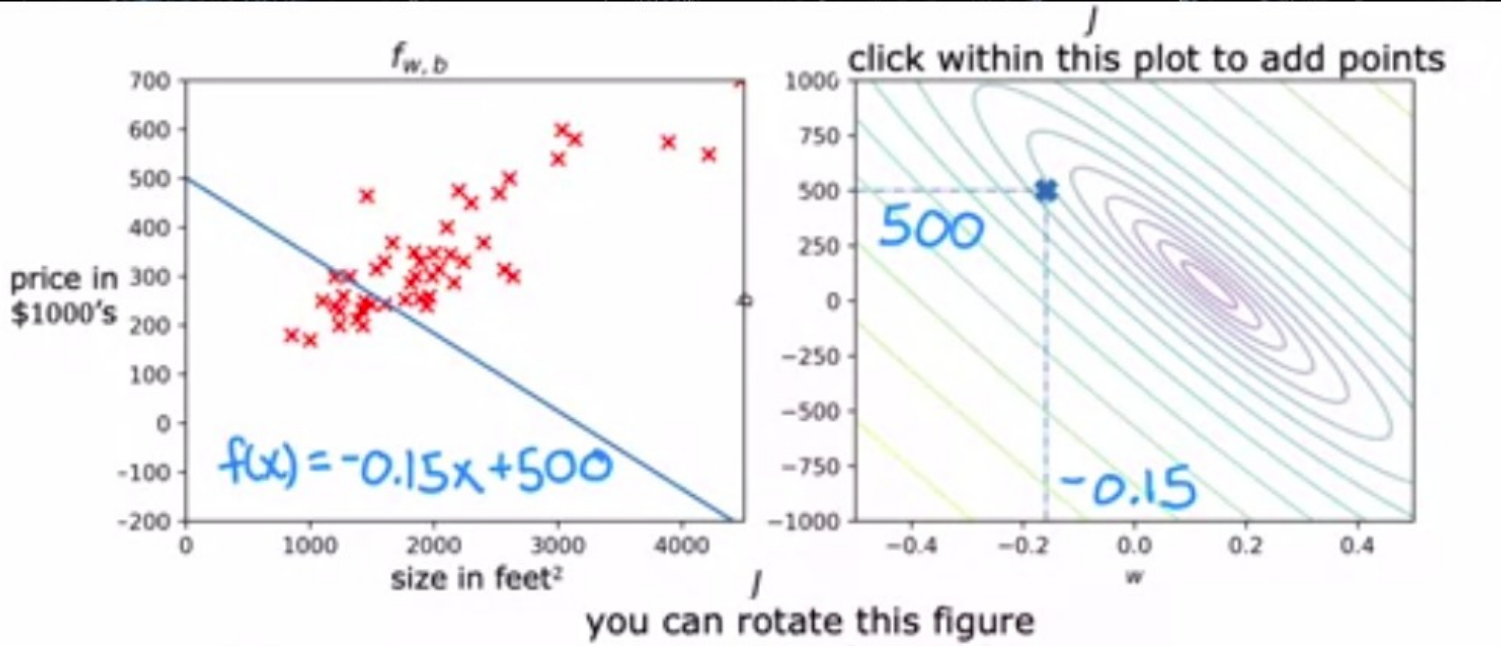

all of these three points have



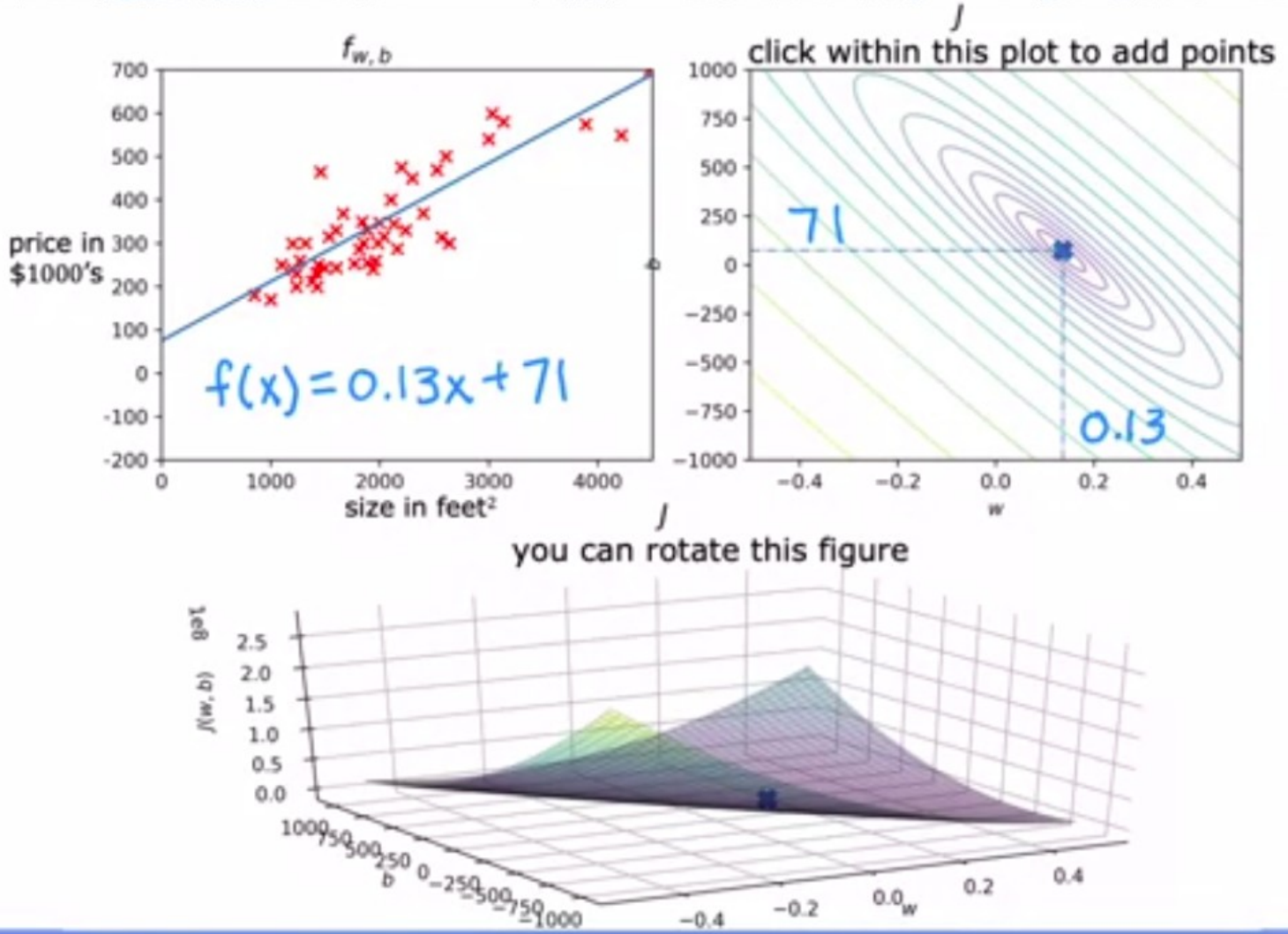

another example with a different choice of w and b.



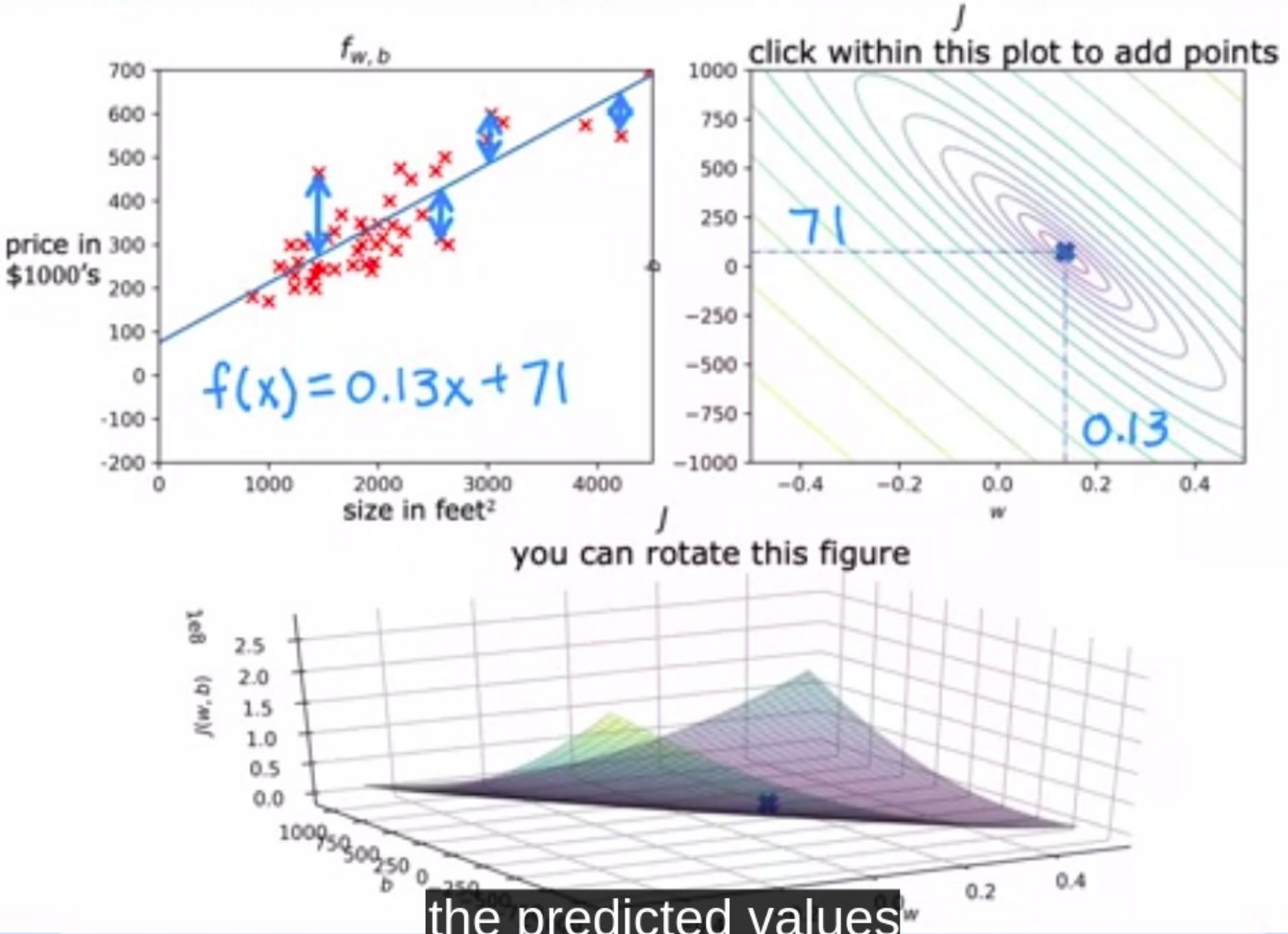

Let's look at yet another example.




Again, not a great fit to the data,




you get to this line, f of x.




the predicted values
on the straight line,

# Practice quiz: Regression

**Total points** 2

**1.** **1 point**

For linear regression, the model is $f_{w,b}(x) = wx + b$.

Which of the following are the inputs, or features, that are fed into the model and with which the model is expected to make a prediction?

- ○ $(x, y)$
- ◉ $x$
- ○ $w$ and $b$.
- ○ $m$

**2.** For linear regression, if you find parameters $w$ and $b$ so that $J(w, b)$ is very close to zero, what can you conclude? **1 point**

- ○ The selected values of the parameters $w$ and $b$ cause the algorithm to fit the training set really poorly.

- ○ $(x, y)$
- ◉ $x$
- ○ $w$ and $b$.
- ○ $m$

**Correct**
The $x$, the input features, are fed into the model to generate a prediction $f_{w,b}(x)$

**2.** For linear regression, if you find parameters $w$ and $b$ so that $J(w, b)$ is very close to zero, what can you conclude? **1 / 1 point**

- ○ The selected values of the parameters $w$ and $b$ cause the algorithm to fit the training set really poorly.
- ○ This is never possible -- there must be a bug in the code.
- ◉ The selected values of the parameters $w$ and $b$ cause the algorithm to fit the training set really well.

**Correct**
When the cost is small, this means that the model fits the training set well.

Have some function $J(w,b)$ for linear regression or any function

Want $\min_{w,b} J(w,b)$ $\min_{w_1,\ldots,w_n,b} J(w_1,w_2,\ldots,w_n,b)$

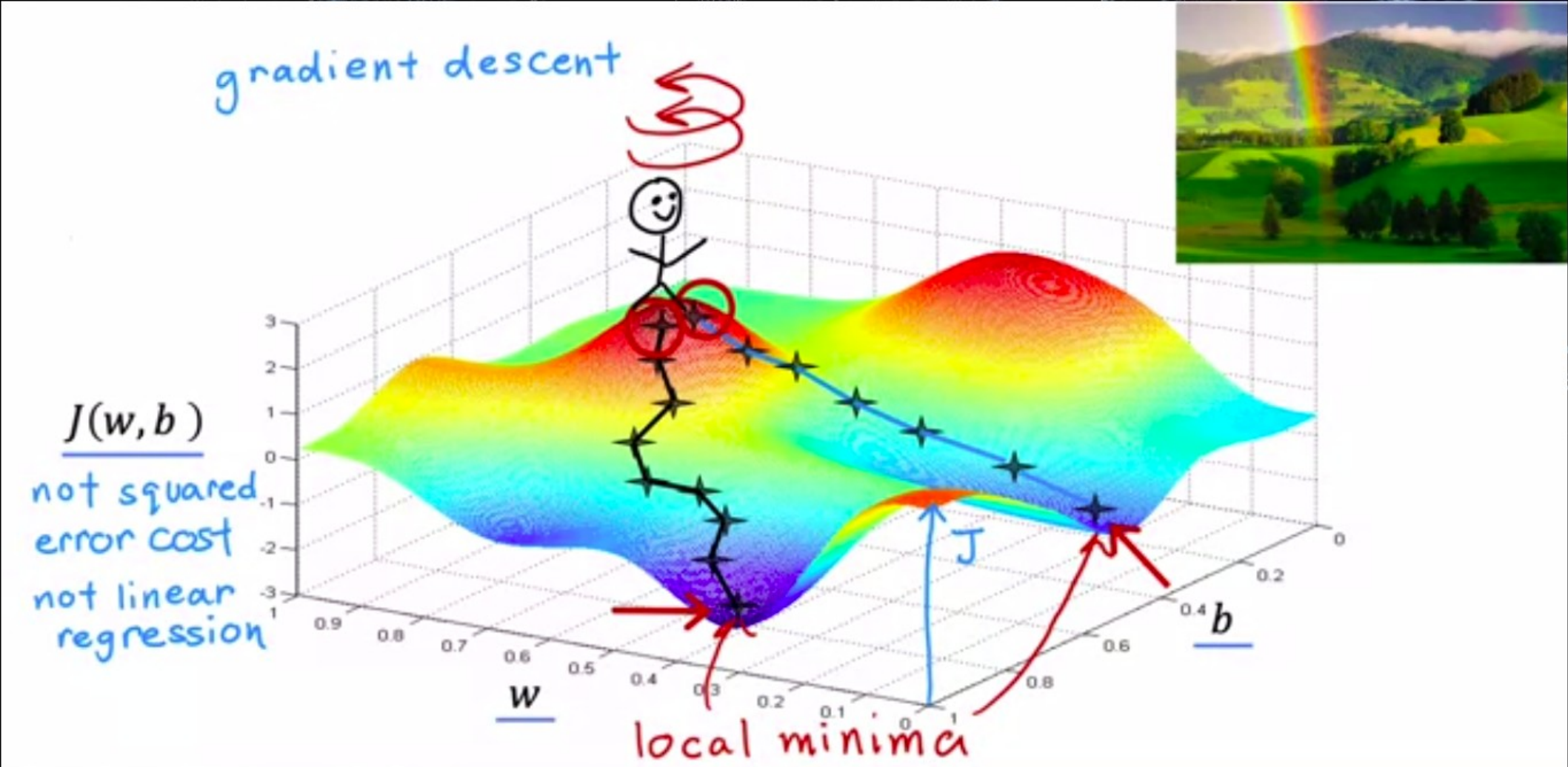
Outline:

- Start with some $w,b$ (set $w=0, b=0$)
- Keep changing $w,b$ to reduce $J(w,b)$
- Until we settle at or near a minimum

J not always

may have >1 minimum

Let's take a look at an example of




This is an interesting property

## Question


Gradient descent is an algorithm for finding values of parameters w and b that minimize the cost function J. What does this update statement do? (Assume $\alpha$ is small.)

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

- (●) Updates parameter $w$ by a small amount
- ( ) Checks whether $w$ is equal to $w - \alpha \frac{\partial J(w,b)}{\partial w}$

✓ **Correct**
This updates the parameter by a small amount, in order to reduce the cost $J$.

Skip | Continue

# Gradient descent algorithm

Repeat until convergence

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

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

Learning rate

Derivative

Simultaneously update w and b

Assignment

$a = c$

$a = a + 1$

Code

Truth assertion

$a = c$

$a = a + 1$

Math

a==c

## Correct: Simultaneous update

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

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

$$w = tmp\_w$$

$$b = tmp\_b$$

## Incorrect

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

$$w = tmp\_w$$

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

$$b = tmp\_b$$

and thus this updated value for

$$w = w - \alpha \frac{d}{dw} J(w)$$

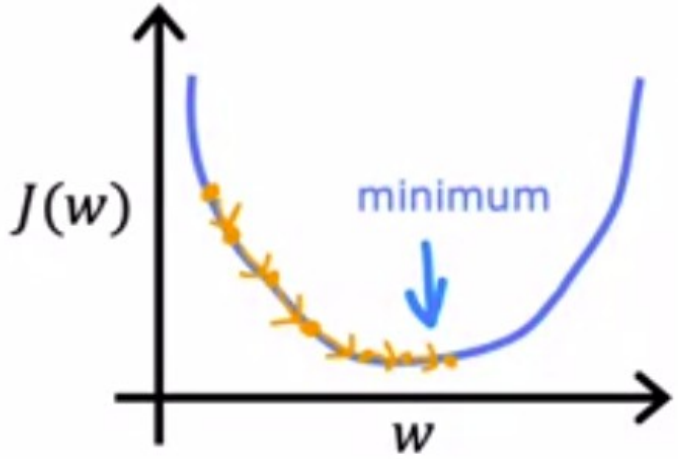
If $\alpha$ is too small...

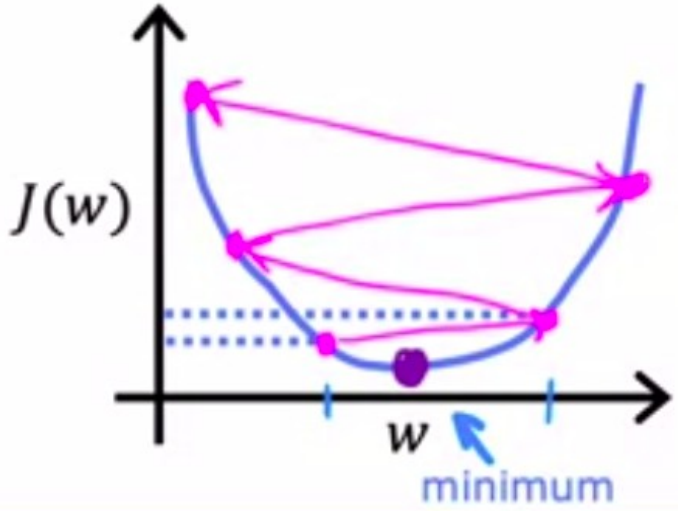
Gradient descent may be slow.

If $\alpha$ is too large...

Gradient descent may:

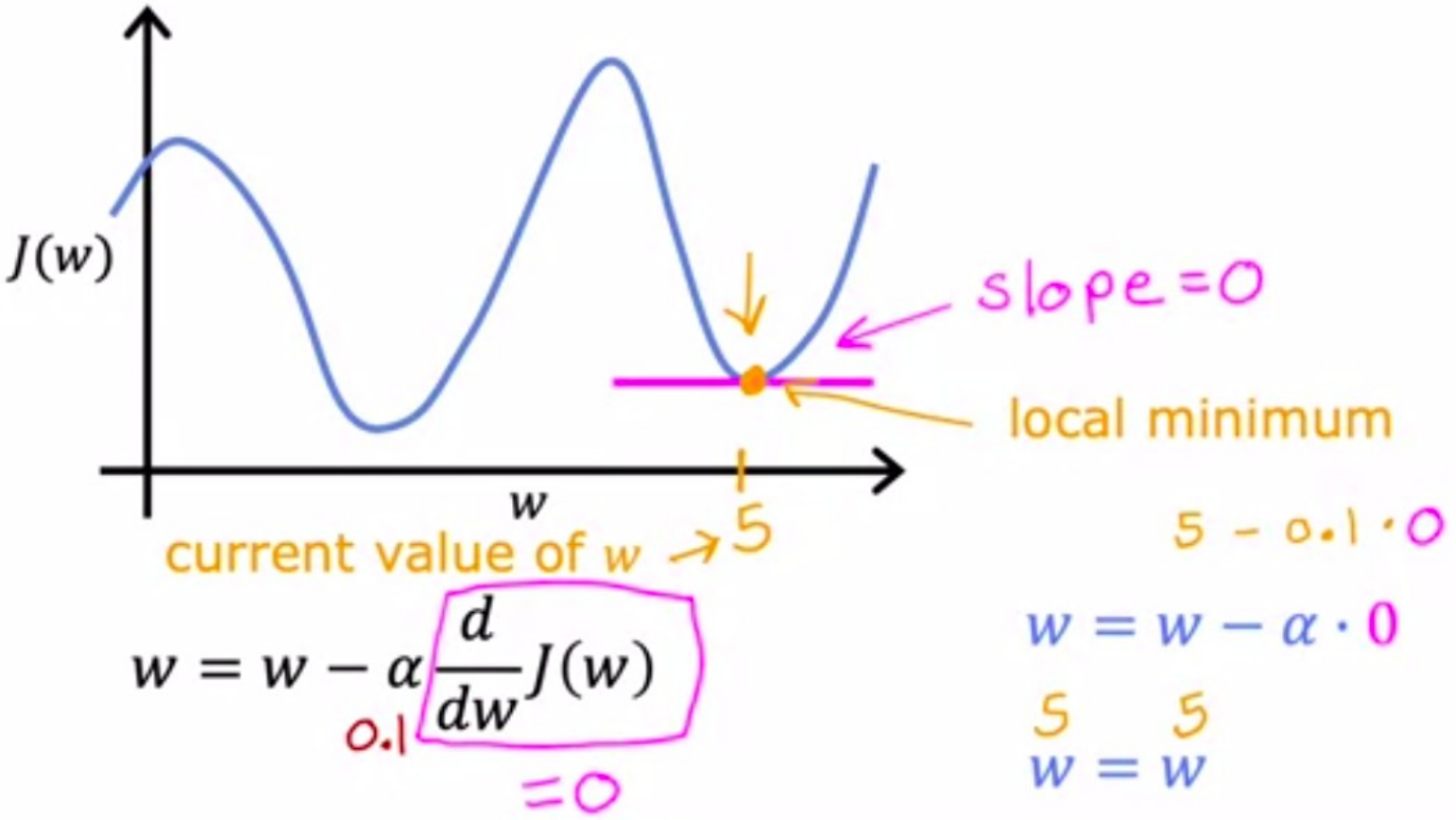
- Overshoot, never reach minimum
- Fail to converge, diverge






may fail to converge and may even diverge.




the solution at that local minimum.

# Can reach local minimum with fixed learning rate

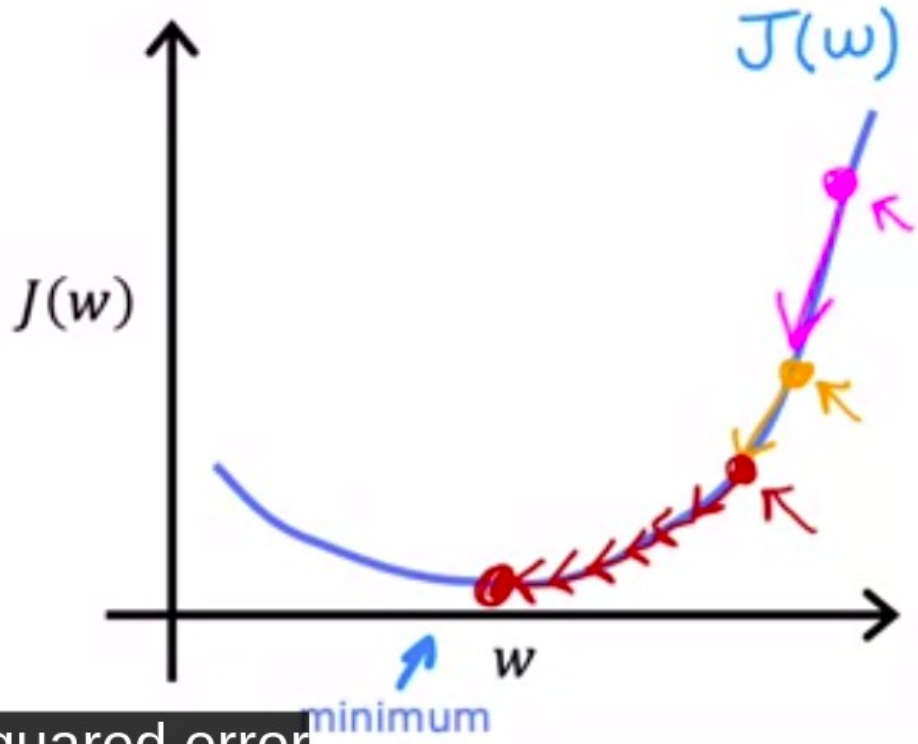
$\alpha$

$$w = w - \alpha \frac{d}{dw} J(w)$$

smaller

not as large

large




Near a local minimum,
- Derivative becomes smaller
- Update steps become smaller

Can reach minimum without decreasing learning rate $\alpha$

Not just the mean squared error cost function that we're using for

## Linear regression model

$$f_{w,b}(x) = wx + b$$

## Cost function

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

## Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

}



these two derivatives and

## Linear regression model

$$f_{w,b}(x) = wx + b$$

## Cost function

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

## Gradient descent algorithm

repeat until convergence {

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

}

implements gradient descent
this way, it will work.

(Optional)

$$\frac{\partial}{\partial w}J(w,b) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)2x^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)x^{(i)}$$

$$\frac{\partial}{\partial b}J(w,b) = \frac{d}{db}\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 = \frac{d}{db}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)2 = \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)$$

no $x^{(i)}$

# Gradient descent algorithm

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

repeat until convergence {

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) \; x^{(i)}$$

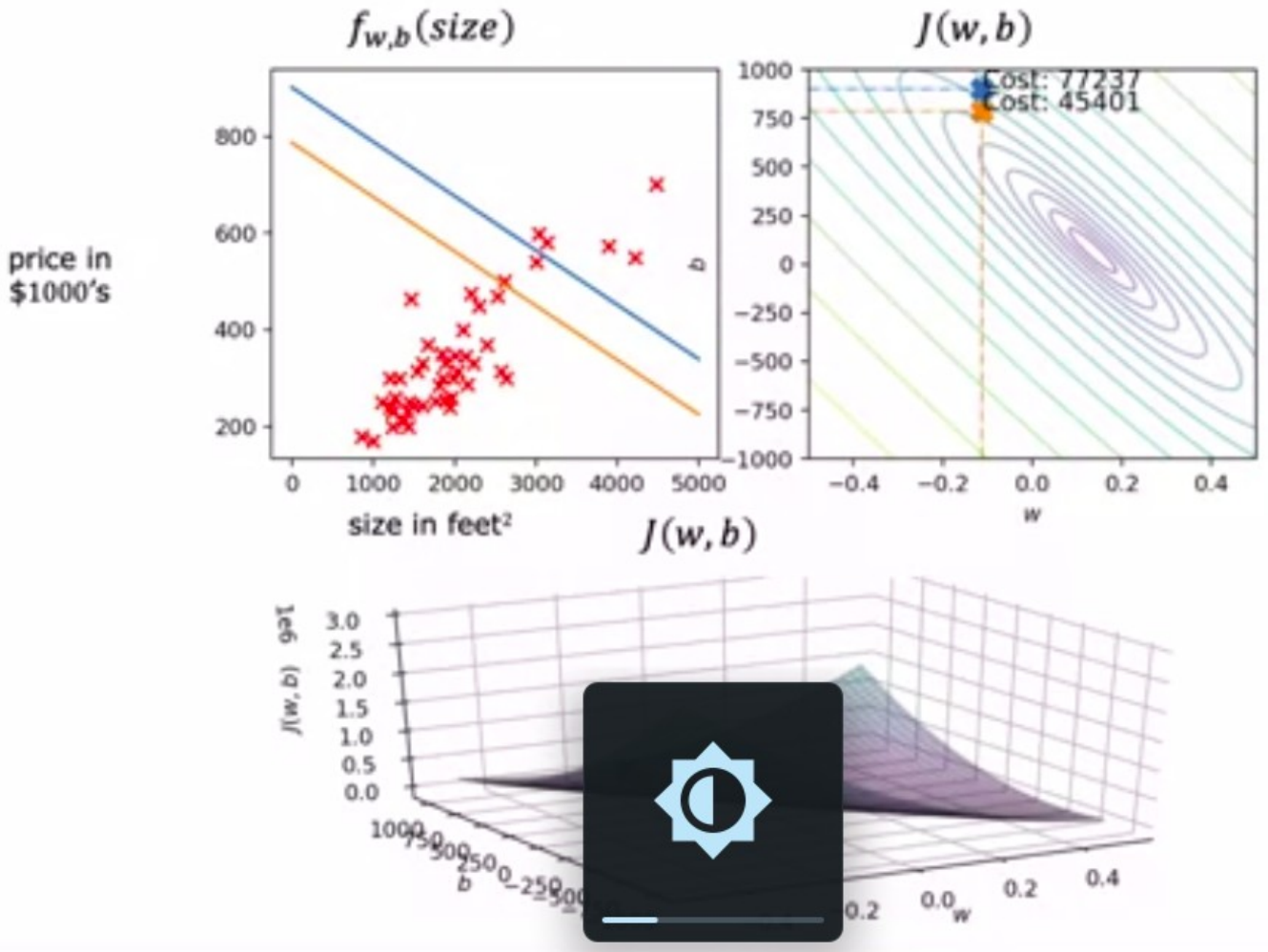
$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

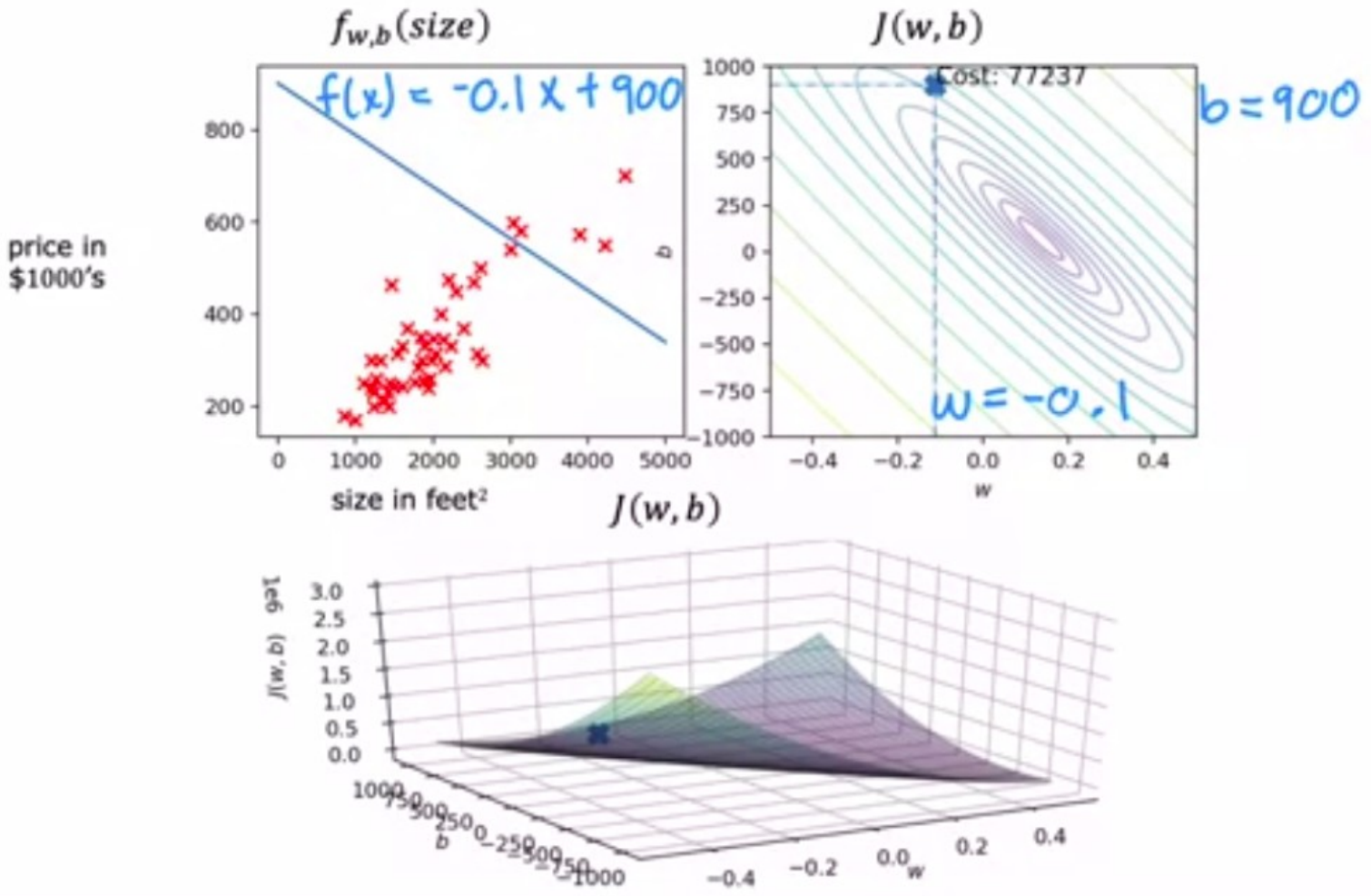
}

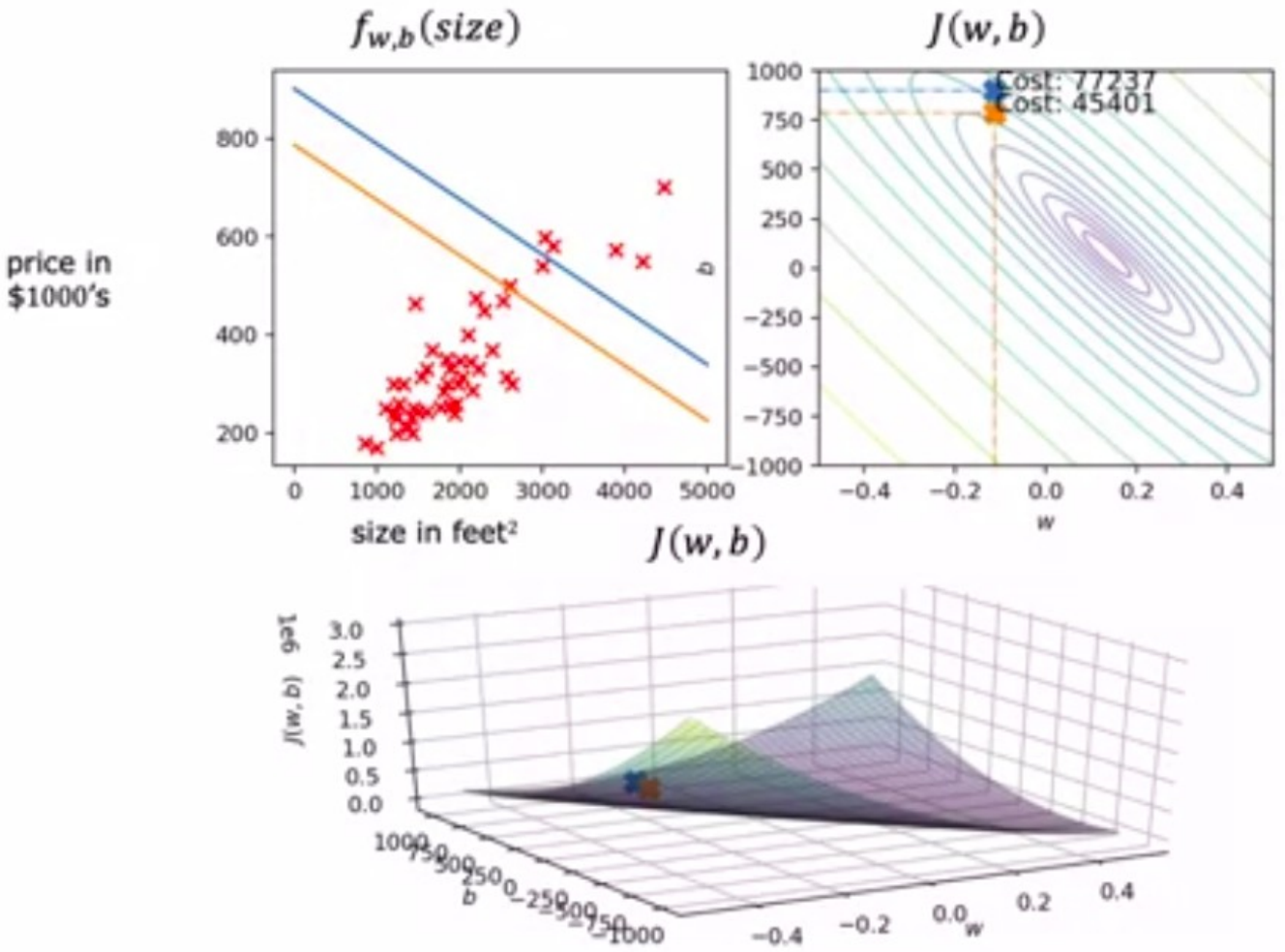
Update $w$ and $b$ simultaneously

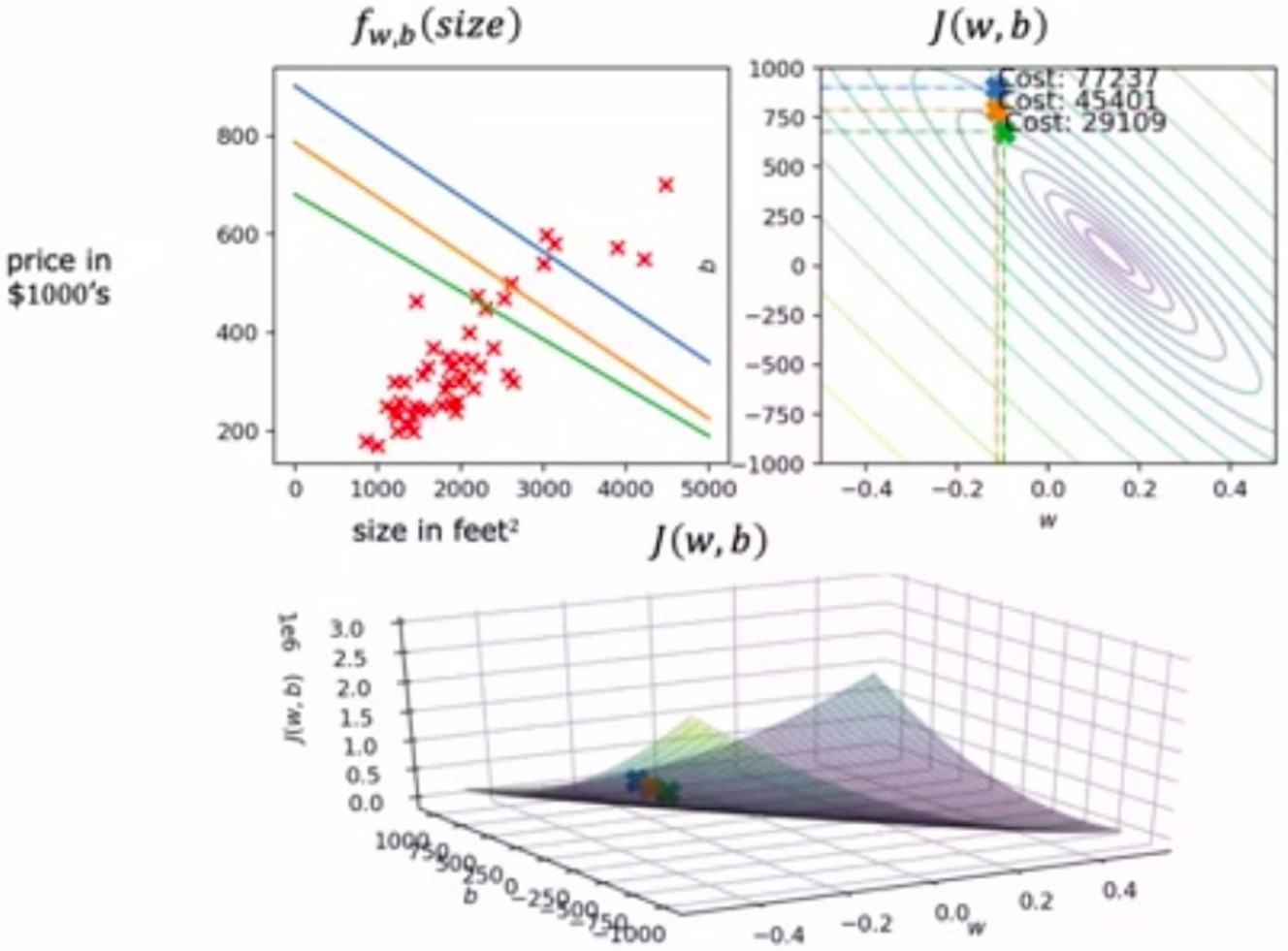
$\frac{\partial}{\partial b} J(w,b)$

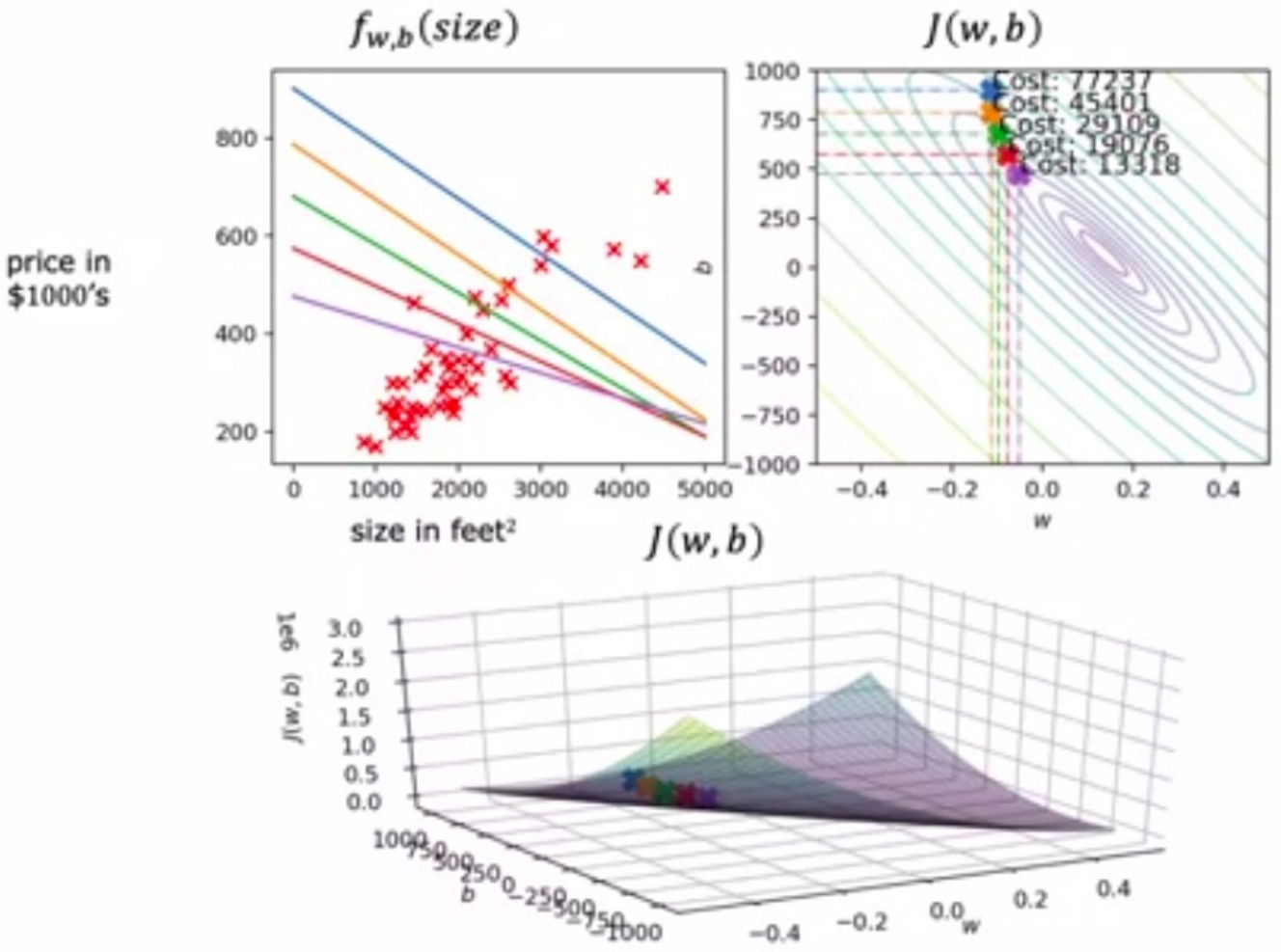
$f_{w,b}(x^{(i)}) = wx^{(i)} + b$

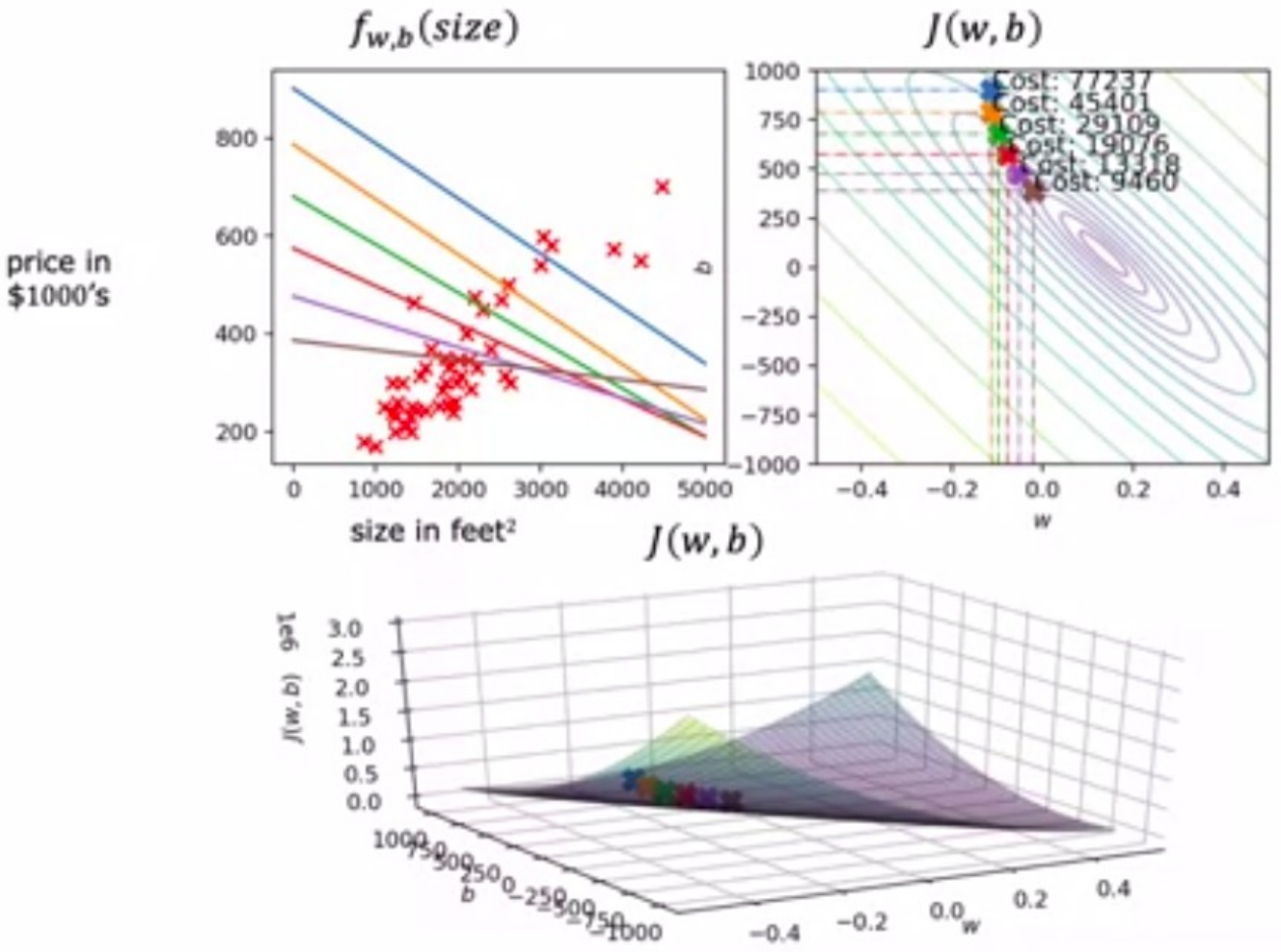

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
w
b
$J(w,b)$
J(w, b)
cost

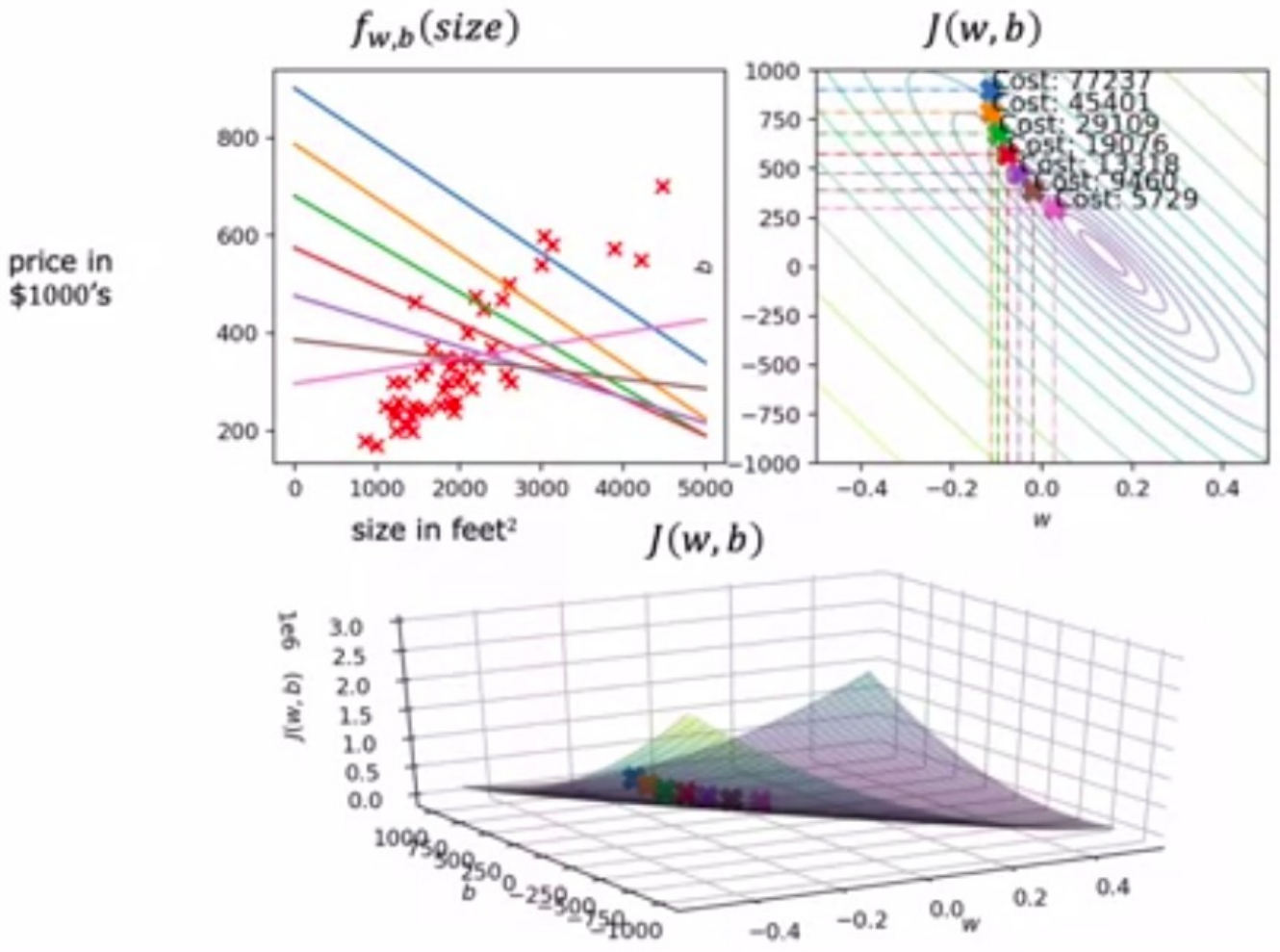

$f_{w,b}(size)$
$f(x) = -0.1x + 900$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
$b = 900$
$w = -0.1$
$J(w,b)$
$J(w,b)$
b
w

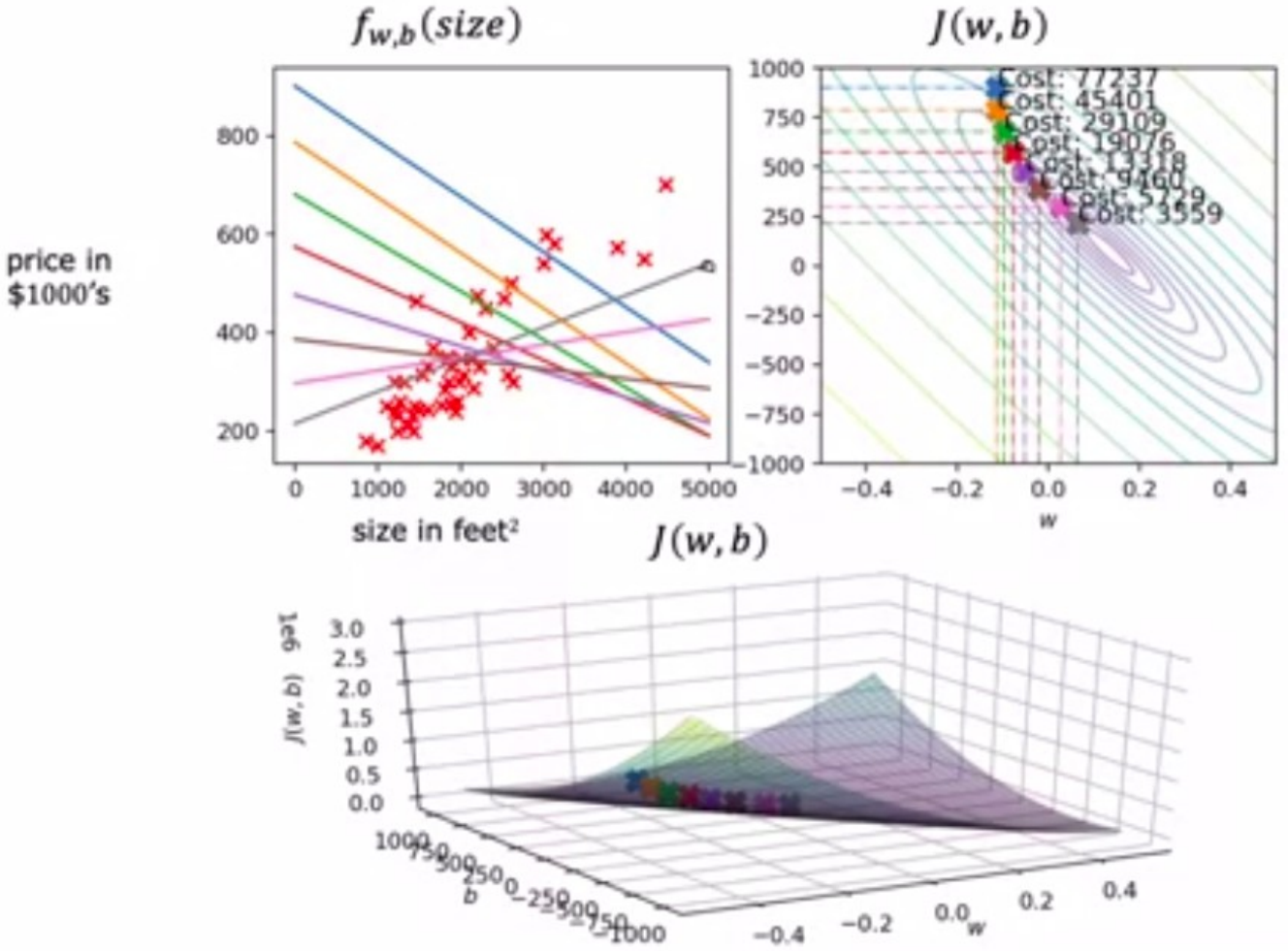

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
w
b
$J(w,b)$
J(w, b)

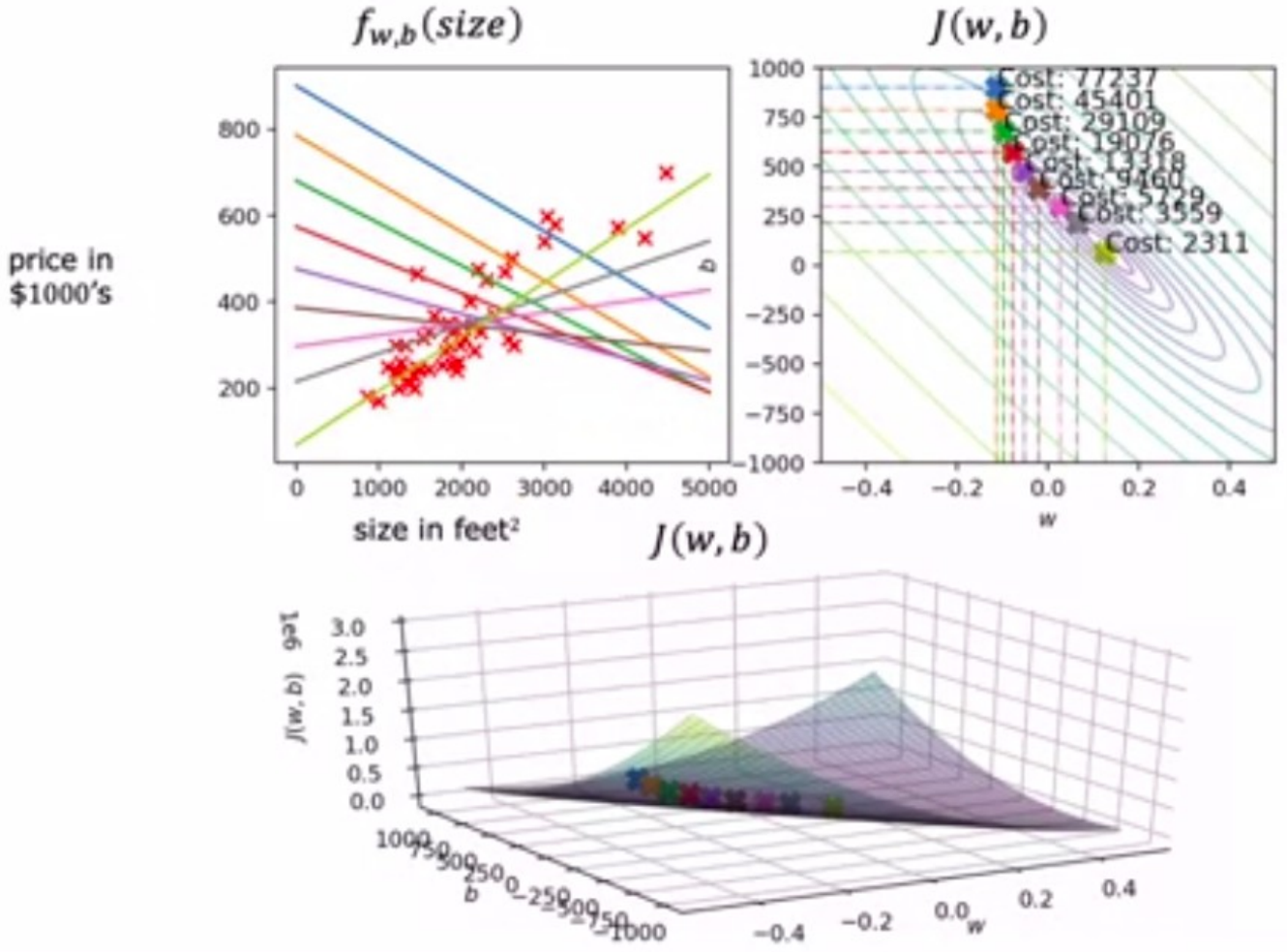

$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
$J(w,b)$
$J(w,b)$


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
w
b
$J(w,b)$
J(w, b)
b
w


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
w
b
$J(w,b)$
J(w, b)
cost


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
$J(w,b)$
J(w, b)
b
w


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
w
$J(w,b)$
J(w, b)
b
w


$f_{w,b}(size)$
price in $1000's
size in feet²
$J(w,b)$
Cost: 77237
Cost: 45401
Cost: 29109
Cost: 19076
Cost: 13318
Cost: 9460
Cost: 5729
Cost: 3559
Cost: 2311
w
b
$J(w,b)$
J(w, b)
b
w

# "Batch" gradient descent

DeepLearning.AI
THE BATCH

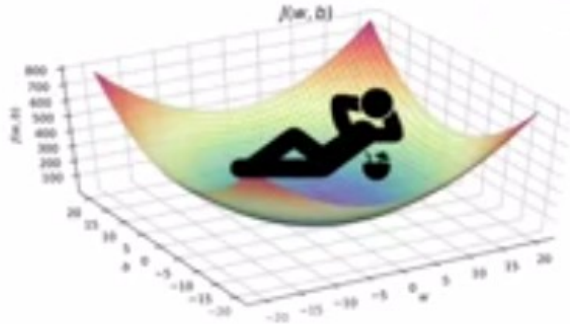
other gradient descent: subsets

"Batch": Each step of gradient descent uses all the training examples.

| | $x$ size in feet² | $y$ price in $1000's |
|---|---|---|
| (1) | 2104 | 400 |
| (2) | 1416 | 232 |
| (3) | 1534 | 315 |
| (4) | 852 | 178 |
| ... | ... | ... |
| (47) | 3210 | 870 |

$m = 47$

$$\sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$



**Practice quiz: Train the model with gradient descent**

Graded Quiz • 10 min

**Due** Sep 4, 11:59 PM IST

**1.** **1 / 1 point**

Gradient descent is an algorithm for finding values of parameters w and b that minimize the cost function J.

repeat until convergence {

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

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

When $\frac{\partial J(w,b)}{\partial w}$ is a negative number (less than zero), what happens to $w$ after one update step?

- ○ $w$ decreases
- ◉ $w$ increases.
- ○ $w$ stays the same
- ○ It is not possible to tell if $w$ will increase or decrease.

- $w$ stays the same
- It is not possible to tell if $w$ will increase or decrease.

**Correct**
The learning rate is always a positive number, so if you take W minus a negative number, you end up with a new value for W that is larger (more positive).

**2.** **1 / 1 point**

For linear regression, what is the update step for parameter b?

- (selected) $b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$
- $b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$

**Correct**
The update step is $b = b - \alpha \frac{\partial J(w,b)}{\partial w}$ where $\frac{\partial J(w,b)}{\partial b}$ can be computed with this expression: $\sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$