Week 2

Multiple features (variables)

Size in feet²	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
x_1	x_2	x_3	x_4	
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$j = 1 \ldots 4$

$n = 4$

$i=2$

$x_j = j^{th}$ feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \;\; 3 \;\; 2 \;\; 40]$

$x_3^{(2)} = 2$

emphasize that this is a
vector and not a number.

Model:

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

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

example

$$f_{w,b}(x) = 0.1 x_1 + 4 x_2 + 10 x_3 + -2 x_4 + 80$$

size #bedrooms #floors years base price

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

$\vec{w} = [w_1 \; w_2 \; w_3 \ldots w_n]$ parameters of the model

b is a number

vector $\vec{x} = [x_1 \; x_2 \; x_3 \ldots x_n]$

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b$$

dot product

multiple linear regression

(not multivariate regression)

as multiple linear regression.

Question

Multiple features (variables)

Size in feet² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price ($) in $1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$j = 1 \ldots 4$

$n = 4$

$i = 2$

x_j = j^{th} feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \ \ 3 \ \ 2 \ \ 40]$

$x_3^{(2)} = 2$

Skip

Continue

Question

In the training set below, what is $x_1^{(4)}$? Please type in the number below (this is an integer such as 123, no decimal points).

Multiple features (variables)

	Size in feet^2	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
	x_1	x_2	x_3	x_4	
	2104	5	1	45	460
$i=2$	1416	3	2	40	232
	1534	3	2	30	315
	852	2	1	36	178
	...	...	...	...	...

$j = 1 \dots 4$

$n = 4$

x_j = j^{th} feature

n = number of features

$\vec{x}^{(i)}$ = features of i^{th} training example

$x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \;\; 3 \;\; 2 \;\; 40]$

$x_3^{(2)} = 2$

Skip

Continue

Question

$i=2$					
	1416	3	2	40	232
	1534	3	2	30	315
	852	2	1	36	178
	...	...	...	...	...

x_j = j^{th} feature
n = number of features
$\vec{x}^{(i)}$ = features of i^{th} training example
$x_j^{(i)}$ = value of feature j in i^{th} training example

$\vec{x}^{(2)} = [1416 \;\; 3 \;\; 2 \;\; 40]$

$x_3^{(2)} = 2$

852

Correct

$x_1^{(4)}$ is the first feature (first column in the table) of the fourth training example (fourth row in the table).

Skip

Continue

Parameters and features

$\vec{w} = [w_1 \quad w_2 \quad w_3]$ $\quad n=3$

b is a number

$\vec{x} = [x_1 \quad x_2 \quad x_3]$

linear algebra: count from 1

NumPy

w[0] w[1] w[2]

```
w = np.array([1.0,2.5,-3.3])
b = 4
x = np.array([10,20,30])
```

x[0] x[1] x[2]

code: count from 0

Without vectorization

$n=100{,}000$

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

Without vectorization

$$f_{\vec{w},b}(\vec{x}) = \left(\sum_{j=1}^{n} w_j x_j\right) + b$$

$\sum_{j=1}^{n} \rightarrow j=1\ldots n$

1,2,3

range(0,n) → j=0...n-1

range(n)

```
f = 0
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
```

Vectorization

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

```
f = np.dot(w,x) + b
```

Question

Which of the following is a vectorized implementation for computing a linear regression model's prediction?

- ( ) f = np.dot(w,x) + b
- ( ) f=0

  For j in range(n):

  f = f + w[j] * x[j]

  f=f + b

✓ **Correct**

This numpy function uses parallel hardware to efficiently calculate the dot product.

Skip

Continue

Without vectorization

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

t_0

`f + w[0] * x[0]`

t_1

`f + w[1] * x[1]`

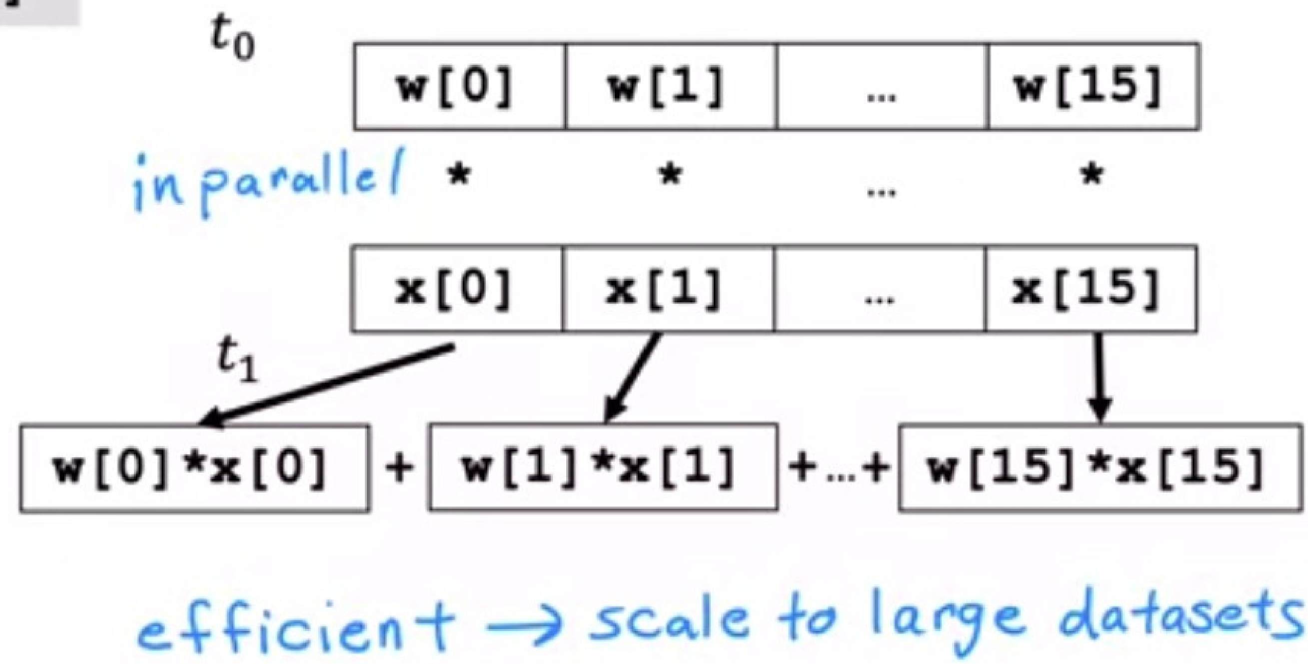
...

t_{15}

`f + w[15] * x[15]`

Vectorization

```
np.dot(w,x)
```



many modern machine
learning algorithms

Gradient descent

$\vec{w} = (w_1 \quad w_2 \quad \cdots \quad w_{16})$ $\cancel{b}$ parameters

derivatives $\vec{d} = (d_1 \quad d_2 \quad \cdots \quad d_{16})$

```
w = np.array([0.5, 1.3, … 3.4])
d = np.array([0.3, 0.2, … 0.4])
```

compute $w_j = w_j - \overbrace{0.1}^{\text{learning rate } \alpha} d_j$ for $j = 1 \ldots 16$

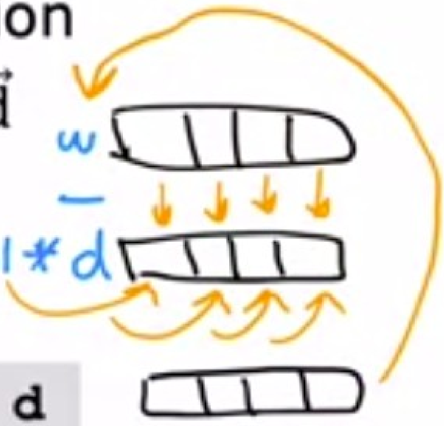
Without vectorization

$$w_1 = w_1 - 0.1 d_1$$
$$w_2 = w_2 - 0.1 d_2$$
$$\vdots$$
$$w_{16} = w_{16} - 0.1 d_{16}$$

```
for j in range(0,16):
    w[j] = w[j] - 0.1 * d[j]
```

With vectorization

$$\vec{w} = \vec{w} - 0.1 \vec{d}$$




```
w = w - 0.1 * d
```

versus taking many hours
to do the same thing.

Question

Which of the following is a vectorized implementation for computing a linear regression model's prediction?

- ( )

```
f = w[0] * x[0] + w[1] * x[1] + w[2] * x[2] + b
```

- ( )

```
f = 0
for j in range(n):
    f = f + w[j] * x[j]
f = f + b
```

- (•)

```
f = np.dot(w,x) + b
```

Skip Continue

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	$\vec{w} = [w_1 \quad \cdots \quad w_n]$ ← vector of length n b still a number
Model	$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \cdots + w_n x_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$ ↑ dot product
Cost function	$J(\underbrace{w_1, \cdots, w_n}, b)$	$J(\vec{w}, b)$

Gradient descent

Previous notation:

$$
\begin{aligned}
&\text{repeat}\{ \\
&\quad w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\underbrace{w_1, \cdots, w_n}, b) \\
&\quad b = b - \alpha \frac{\partial}{\partial b} J(\underbrace{w_1, \cdots, w_n}, b) \\
&\}
\end{aligned}
$$

Vector notation:

$$
\begin{aligned}
&\text{repeat}\{ \\
&\quad w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \\
&\quad b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b) \\
&\}
\end{aligned}
$$

let's take a look at
the derivative term.

Gradient descent

One feature

repeat {

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

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$j=1$

$$w_1 = w_1 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_1^{(i)}}_{\frac{\partial}{\partial w_1}J(\vec{w},b)}$$

$\vdots$

$j=n$

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

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

simultaneously update

w_j (for $j = 1, \cdots, n$) and b

}

That's it for gradient descent for multiple regression.

An alternative to gradient descent

→Normal equation

- Only for linear regression
- Solve for w, b without iterations

Disadvantages

- Doesn't generalize to other learning algorithms.
- Slow when number of features is large (> 10,000)

What you need to know

- Normal equation method may be used in machine learning libraries that implement linear regression.
- Gradient descent is the recommended method for finding parameters w,b

gradient descents offer a
better way to get the job done.

1. In the training set below, what is $x_4^{(3)}$? Please type in the number below (this is an integer such as 123, no decimal points).

1 / 1 point

Size in feet^2	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
x_1	x_2	x_3	x_4	
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

30

2.

1 / 1 point

Which of the following are potential benefits of vectorization? Please choose the best option.

- ○ It makes your code run faster
- ○ It can make your code shorter
- ○ It allows your code to run more easily on parallel compute hardware
- ◉ All of the above

Correct
Correct! All of these are benefits of vectorization!

3. True/False? To make gradient descent converge about twice as fast, a technique that almost always works is to double the learning rate $alpha$.

1 / 1 point

- ◉ False
- ○ True

Correct
Doubling the learning rate may result in a learning rate that is too large, and cause gradient descent to fail to find the optimal values for the parameters w and b.

Feature and parameter values

$\widehat{price} = w_1 x_1 + w_2 x_2 + b$

x_1 ↓ size x_2 ↓ #bedrooms

x_1: size (feet^2)
range: 300 – 2,000
large

x_2: # bedrooms
range: 0 – 5
small

House: $x_1 = 2000$, $x_2 = 5$, $price = \$500k$ one training example

size of the parameters w_1, w_2?

$w_1 = 50, \quad w_2 = 0.1, \quad b = 50$

$\widehat{price} = \underbrace{50 * 2000}_{100{,}000K} + \underbrace{0.1 * 5}_{0.5K} + \underbrace{50}_{50K}$

$\widehat{price} = \$100{,}050.5k = \$100{,}050{,}500$

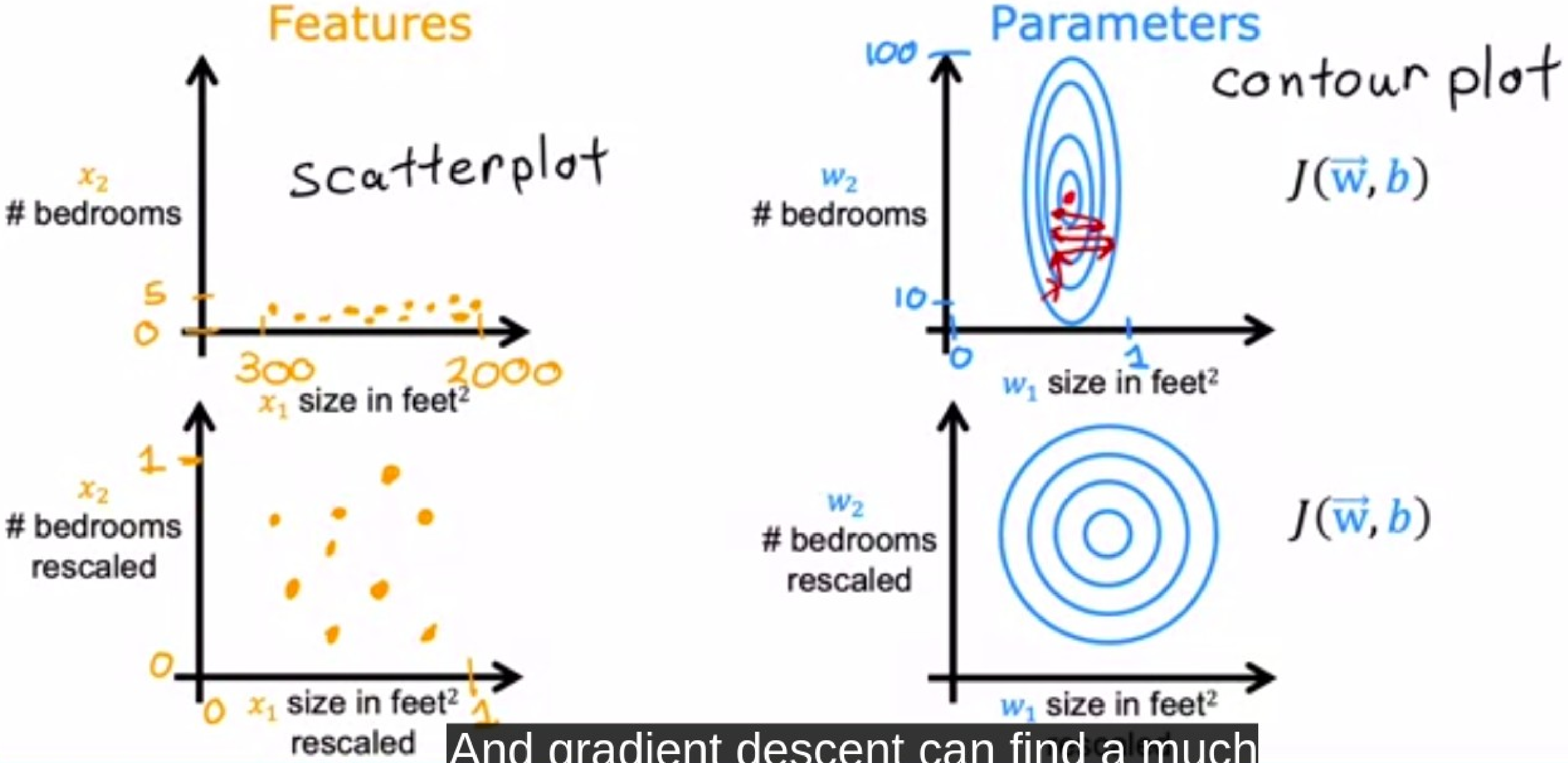
$w_1 = 0.1, \quad w_2 = 50, \quad b = 50$
small large

$\widehat{price} = \underbrace{0.1 * 2000k}_{200K} + \underbrace{50 * 5}_{250K} + \underbrace{50}_{50K}$

$\widehat{price} = \$500k$ more reasonable

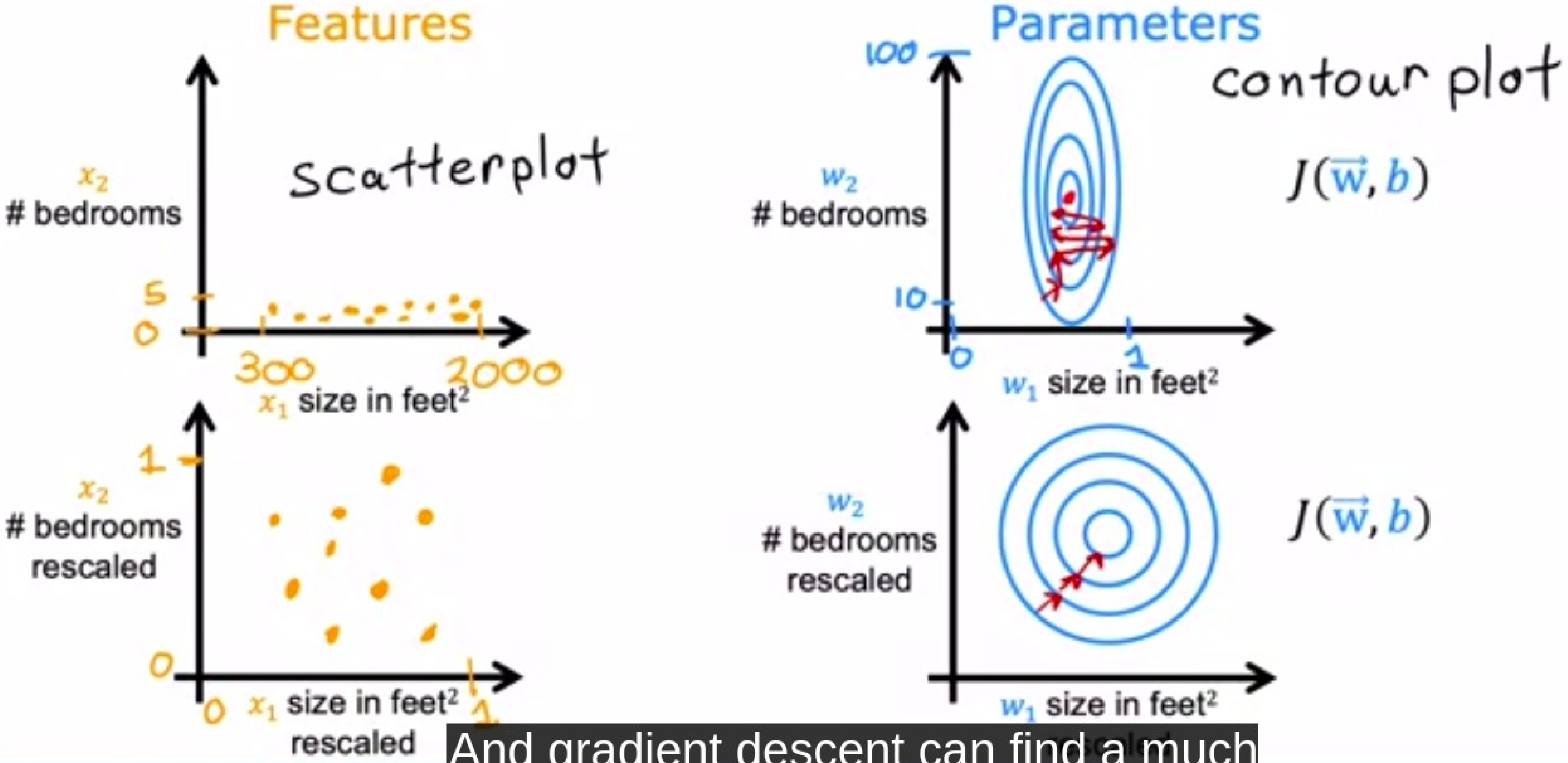
its parameters will be
relatively large like 50.

Feature size and gradient descent



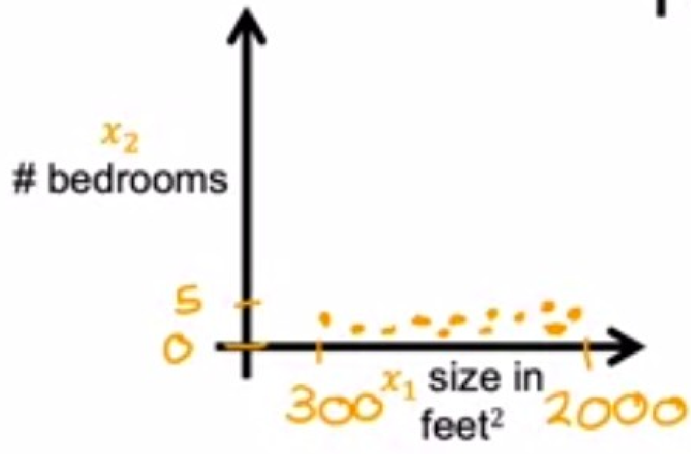
And gradient descent can find a much
more direct path to the global minimum.

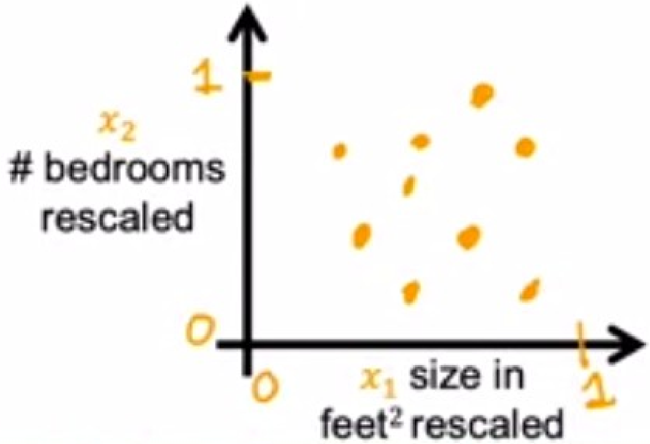
Feature size and gradient descent



And gradient descent can find a much more direct path to the global minimum.

Feature scaling





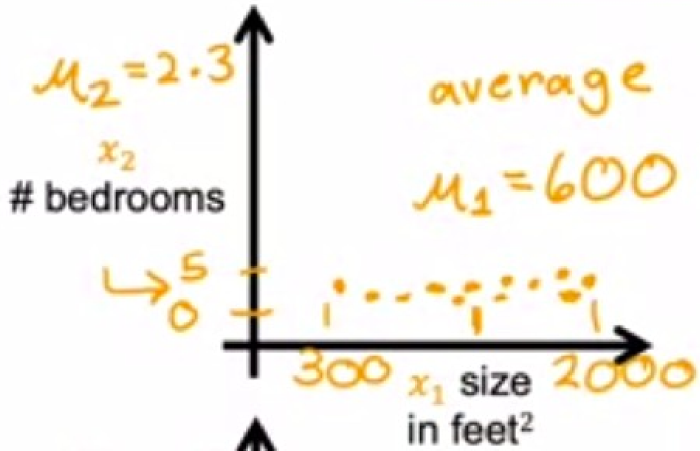
$$300 \le x_1 \le 2000 \qquad 0 \le x_2 \le 5$$

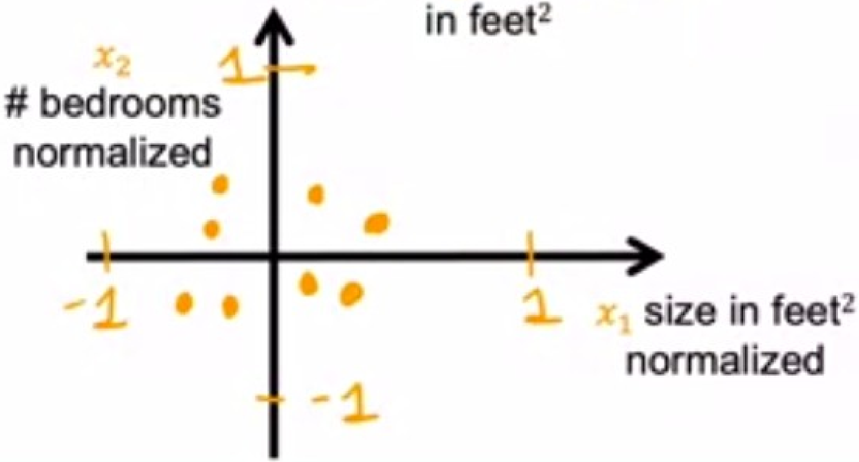
$$x_{1,scaled} = \frac{x_1}{\underset{max}{2000}} \qquad x_{2,scaled} = \frac{x_2}{\underset{max}{5}}$$

$$0.15 \le x_{1,scaled} \le 1 \qquad 0 \le x_{2,scaled} \le 1$$

it might look like this.

Mean normalization





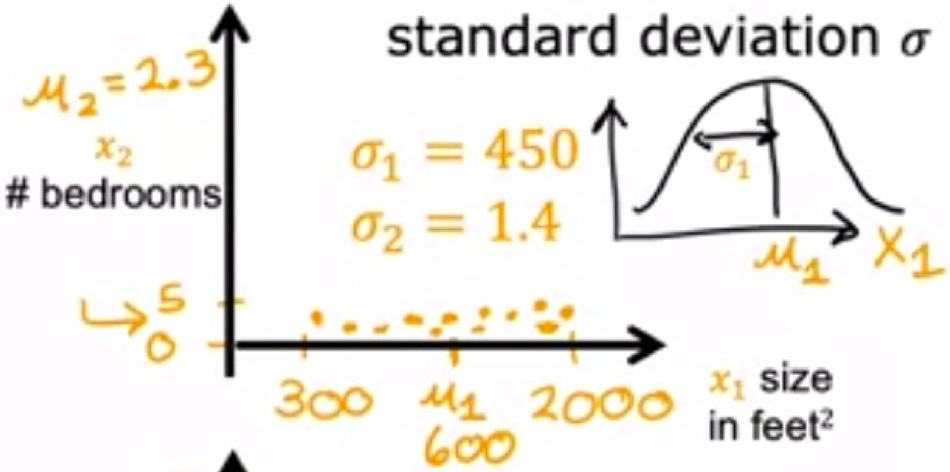
$$300 \le x_1 \le 2000 \qquad 0 \le x_2 \le 5$$

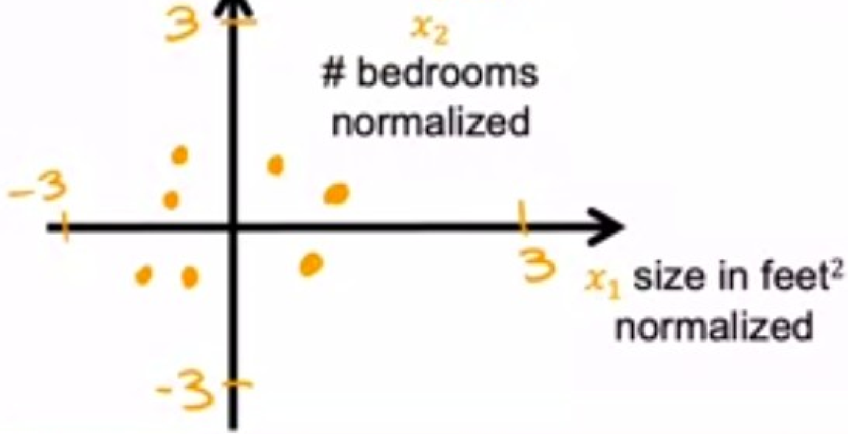
$$x_1 = \frac{x_1 - \mu_1}{2000-300} \qquad x_2 = \frac{x_2 - \mu_2}{5-0}$$

max-min max-min

$$-0.18 \le x_1 \le 0.82 \qquad -0.46 \le x_2 \le 0.54$$

Z-score normalization





$$300 \leq x_1 \leq 2000 \qquad 0 \leq x_2 \leq 5$$

$$x_1 = \frac{x_1 - \mu_1}{\sigma_1} \qquad x_2 = \frac{x_2 - \mu_2}{\sigma_2}$$

$$-0.67 \leq x_1 \leq 3.1 \qquad -1.6 \leq x_2 \leq 1.9$$

it might look like this.

Question

Which of the following is a valid step used during feature scaling?

- ○ Multiply each value by the maximum value for that feature
- ◉ Divide each value by the maximum value for that feature

✓ **Correct**
By dividing all values by the maximum, the new maximum range of the rescaled features is now 1 (and all other rescaled values are less than 1).

Skip Continue

Feature scaling

aim for about $-1 \le x_j \le 1$ for each feature x_j

$$-3 \le x_j \le 3$$
$$-0.3 \le x_j \le 0.3$$

acceptable ranges

$0 \le x_1 \le 3$ okay, no rescaling

$-2 \le x_2 \le 0.5$ okay, no rescaling

$-100 \le x_3 \le 100$ too large → rescale

$-0.001 \le x_4 \le 0.001$ too small → rescale

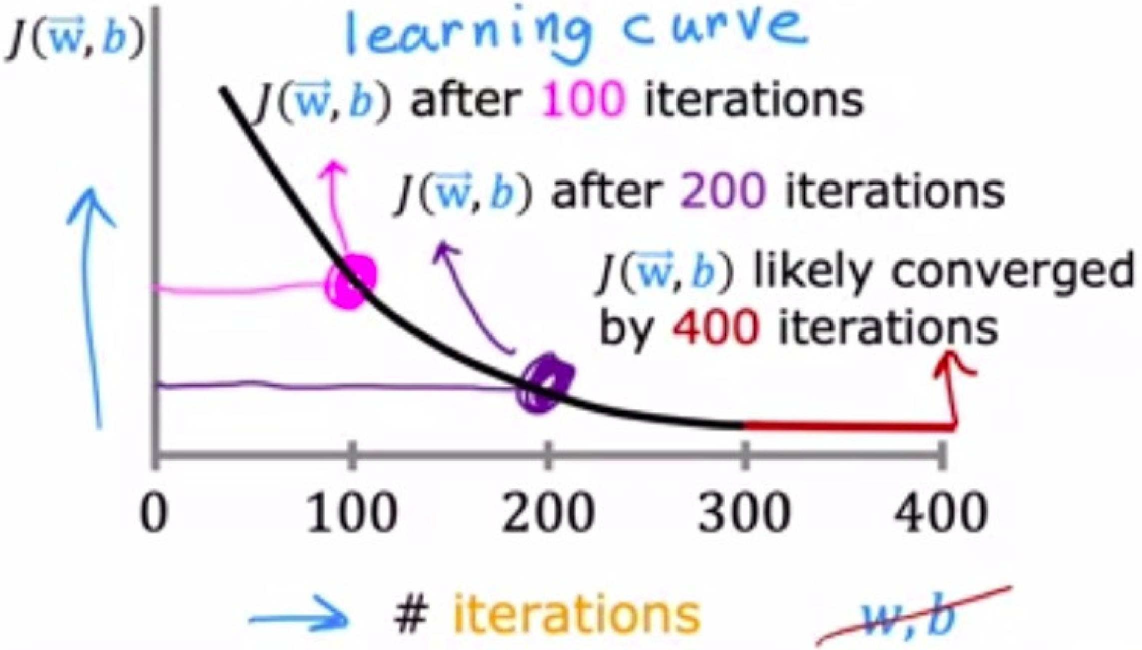
$98.6 \le x_5 \le 105$ too large → rescale

gradient descent to
run much faster.

Make sure gradient descent is working correctly

objective: $\min_{\vec{w},b} J(\vec{w}, b)$

$J(\vec{w}, b)$ should decrease after every iteration



\# iterations needed varies 30 1,000 100,000

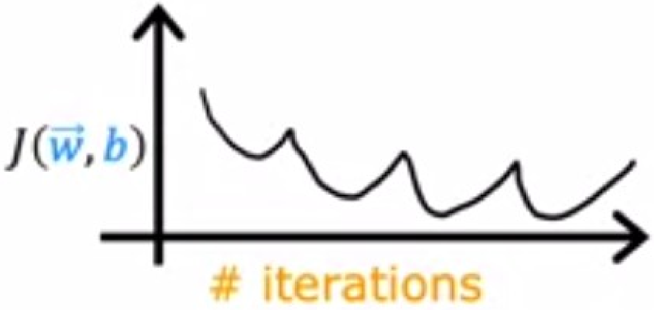
Automatic convergence test

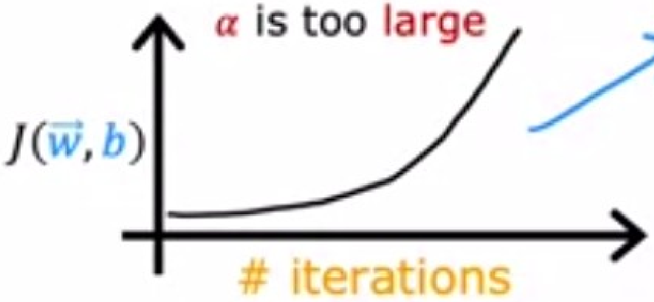
Let ε "epsilon" be 10^{-3}.
0.001

If $J(\vec{w}, b)$ decreases by $\leq \varepsilon$ in one iteration, declare convergence.

(found parameters $\vec{w}, b$ to get close to global minimum)

Identify problem with gradient descent





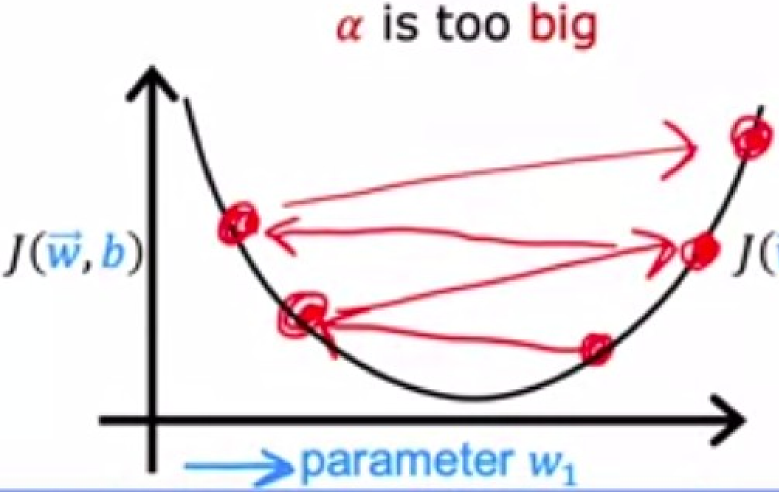
or learning rate is too large

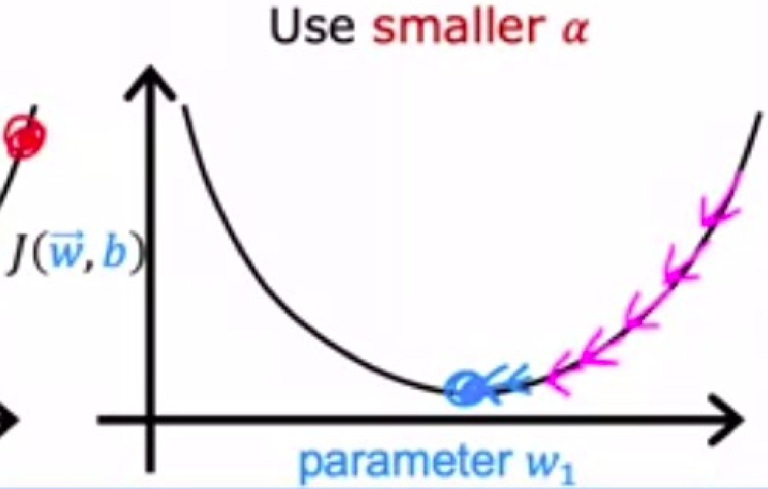
$w_1 = w_1 + \alpha d_1$

use a minus sign

$w_1 = w_1 - \alpha d_1$

Adjust learning rate





With a small enough α, $J(\vec{w}, b)$ should decrease on every iteration

If α is too small, gradient descent takes a lot more iterations to converge

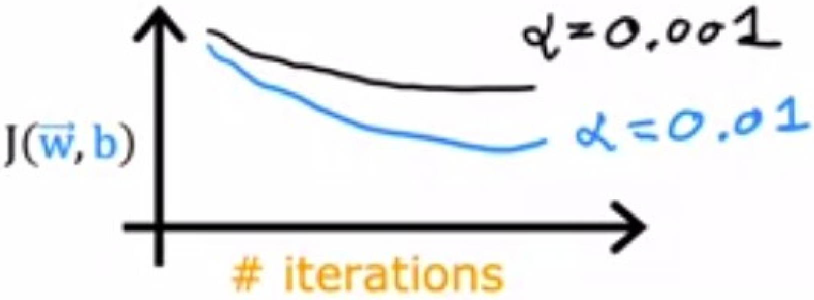
a lot of iterations to converge.

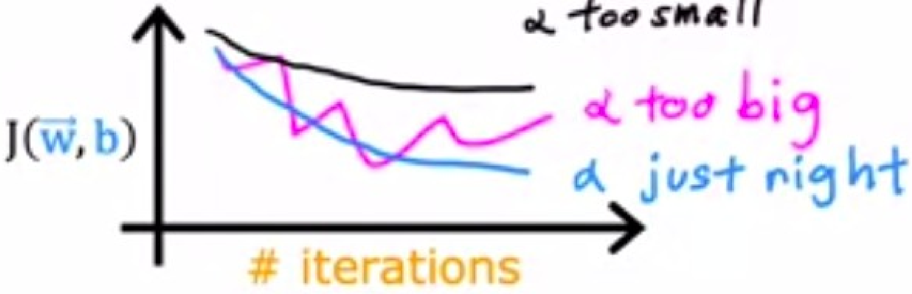
3:24 / 6:06

Values of α to try:

... 0.001 0.003 0.01 0.03 0.1 0.3 1 ...

3X ≈3X 3X ≈3X 3X ≈3X





your implementation
of gradient descent.

Question

You run gradient descent for 15 iterations with $\alpha = 0.3$ and compute $J(w)$ after each iteration. You find that the value of $J(w)$ increases over time. How do you think you should adjust the learning rate α?

- ○ Try a larger value of α (say $\alpha = 1.0$).
- ○ Try running it for only 10 iterations so $J(w)$ doesn't increase as much.
- ○ Keep running it for additional iterations
- ◉ Try a smaller value of α (say $\alpha = 0.1$).

✓ **Correct**
Since the cost function is increasing, we know that gradient descent is diverging, so we need a lower learning rate.

Skip   Continue

Question

$x_3 = x_1 x_2$

new feature

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Feature engineering:
Using intuition to design new features, by transforming or combining original features.

If you have measurements for the dimensions of a swimming pool (length, width, height), which of the following two would be a more useful engineered feature?

- (selected) $length \times width \times height$
- $length + width + height$

Correct
The volume of the swimming pool could be a useful feature to use. This is the more useful engineered feature of the two.

Skip Continue

Feature engineering

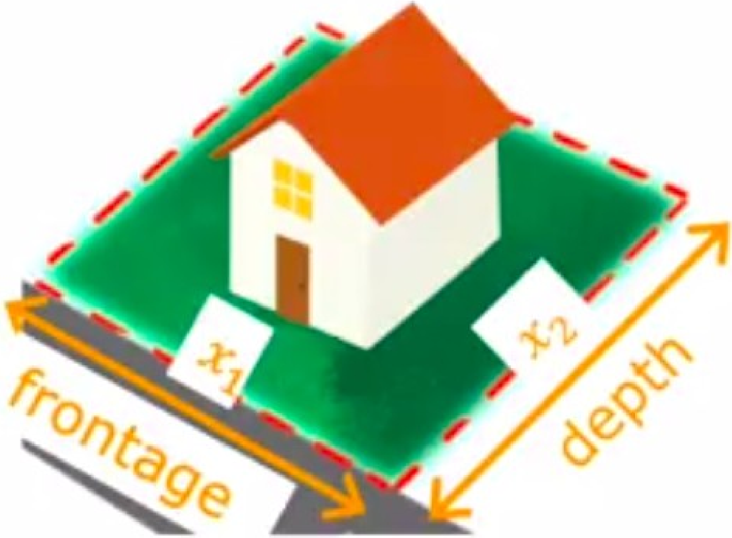
$$f_{\vec{w},b}(\vec{x}) = w_1 \underbrace{x_1}_{\text{frontage}} + w_2 \underbrace{x_2}_{\text{depth}} + b$$

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$

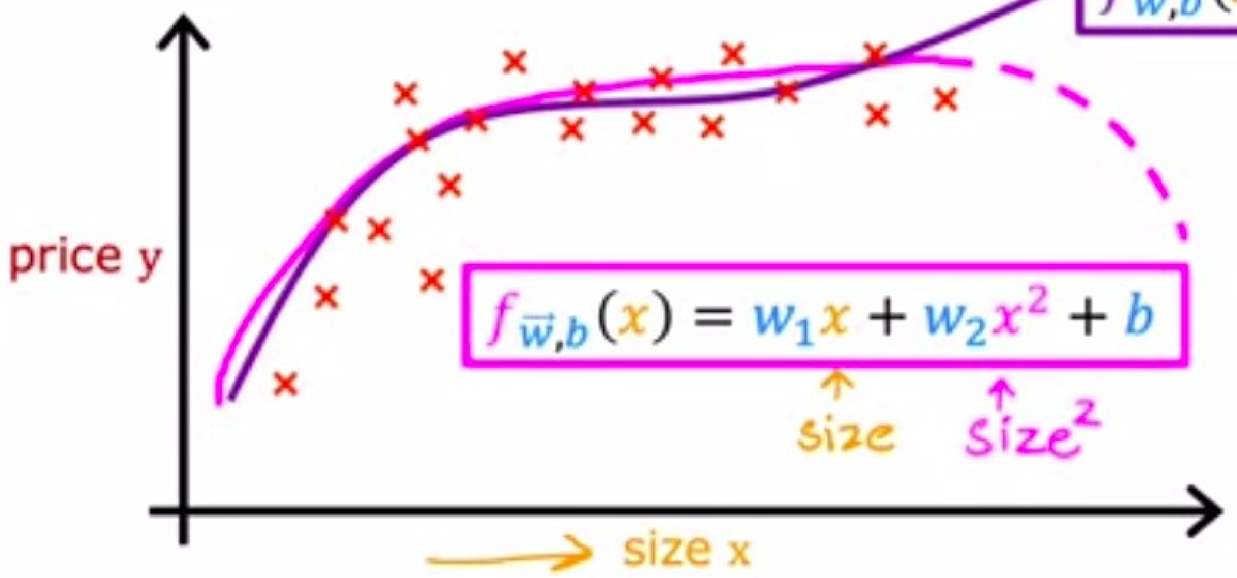
new feature

$$f_{\vec{w},b}(\vec{x}) = \underbrace{w_1} x_1 + \underbrace{w_2} x_2 + \underbrace{w_3} x_3 + b$$



Feature engineering:
Using intuition to design new features, by transforming or combining original features.

Polynomial regression



$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + b$$

size   size^2

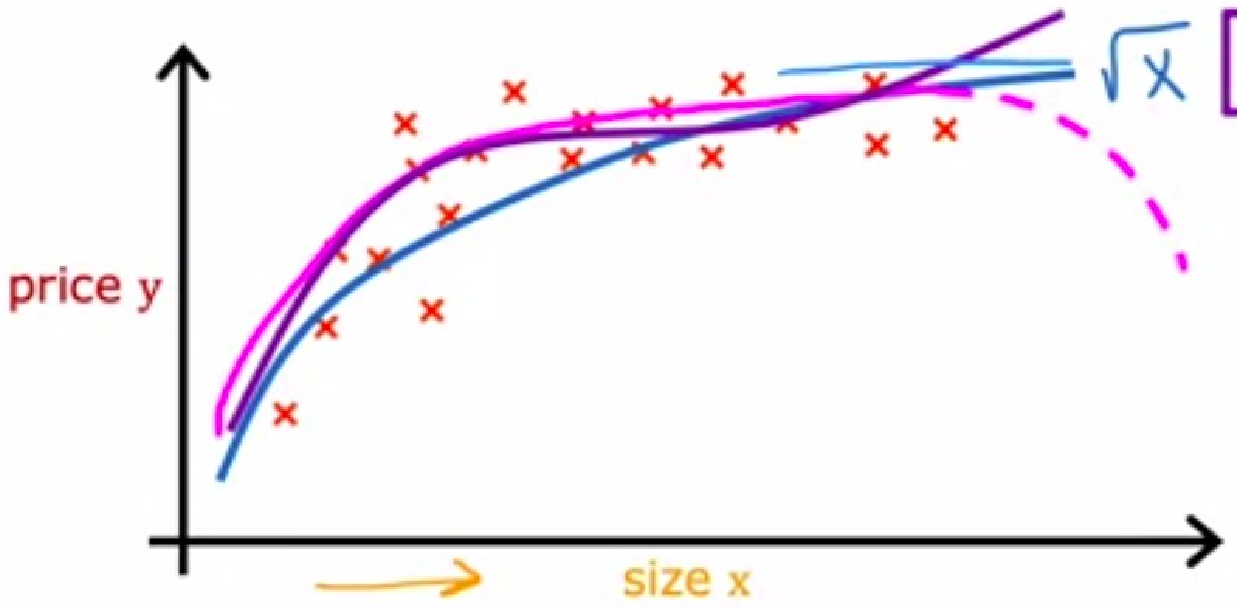
$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + b$$

size	size^2	size^3
$1-10^3$	$1-10^6$	$1-10^9$

feature scaling

your features into
comparable ranges of values.

Choice of features



$$\boxed{f_{\vec{w},b}(x) = w_1 x + w_2\sqrt{x} + b}$$

size $\quad$ $\sqrt{\text{size}}$

what features to use?
↪ course 2

a much better model
for your data.

Back

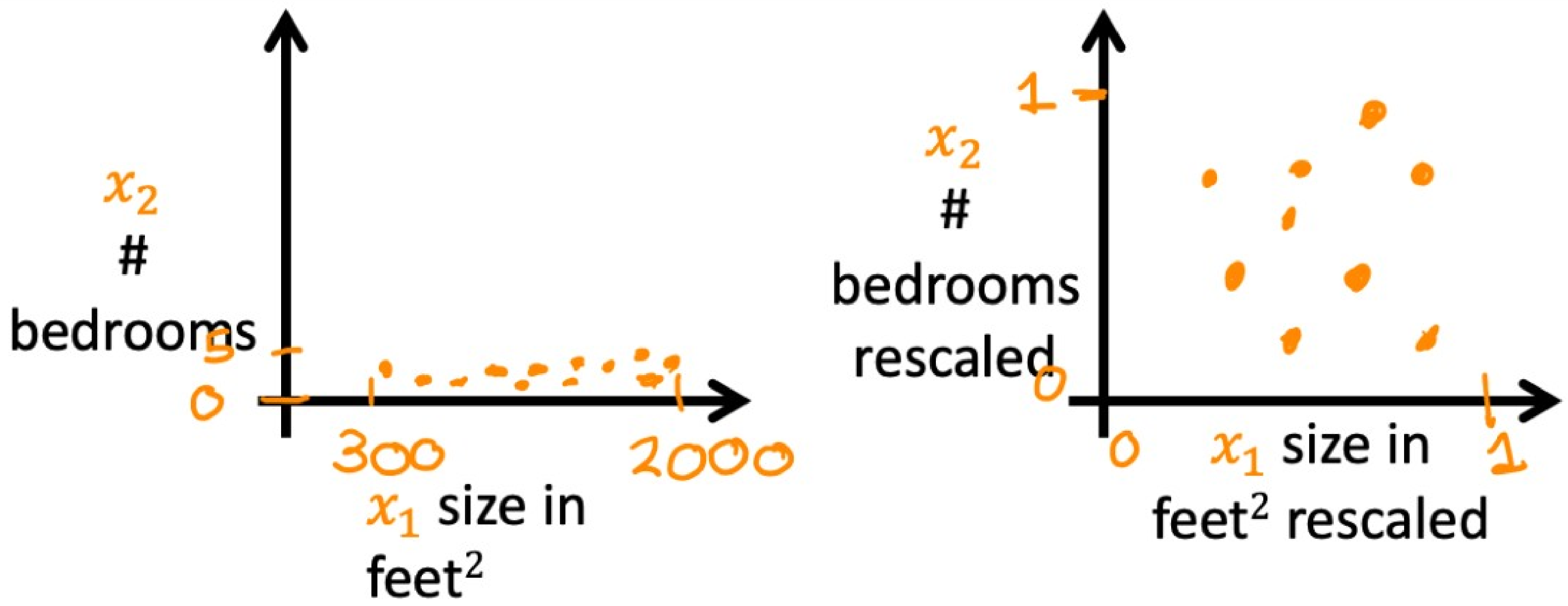
Practice quiz: Gradient descent in practice

Graded Quiz • 30 min

Due Sep 11, 11:59 PM IST

1.

1 / 1 point



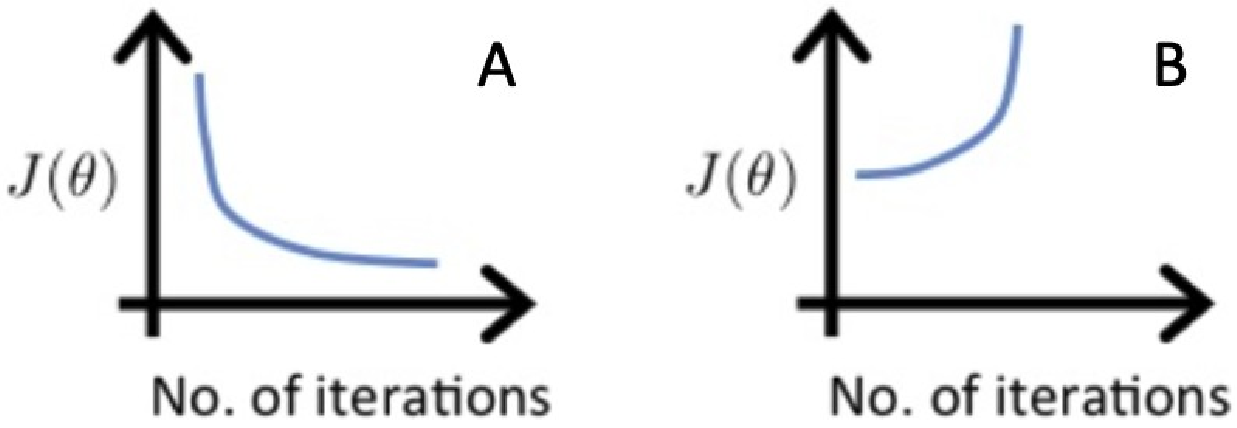
Which of the following is a valid step used during feature scaling?

- (selected) Subtract the mean (average) from each value and then divide by the (max - min).
- Add the mean (average) from each value and and then divide by the (max - min).

Correct

This is called mean normalization.

2. Suppose a friend ran gradient descent three separate times with three choices of the learning rate α and plotted the learning curves for each (cost J for each iteration). **1 / 1 point**



For which case, A or B, was the learning rate α likely too large?

- (●) case B only
- ( ) Both Cases A and B
- ( ) case A only
- ( ) Neither Case A nor B

✓ **Correct**

Correct
The cost is increasing as training continues, which likely indicates that the learning rate alpha is too large.

3. Of the circumstances below, for which one is feature scaling particularly helpful?

1 / 1 point

- ○ Feature scaling is helpful when all the features in the original data (before scaling is applied) range from 0 to 1.
- ◉ Feature scaling is helpful when one feature is much larger (or smaller) than another feature.

Correct
For example, the "house size" in square feet may be as high as 2,000, which is much larger than the feature "number of bedrooms" having a value between 1 and 5 for most houses in the modern era.

4.

1 / 1 point

You are helping a grocery store predict its revenue, and have data on its items sold per week, and price per item. What could be a useful engineered feature?

- ◉ For each product, calculate the number of items sold times price per item.
- ○ For each product, calculate the number of items sold divided by the price per item.

Correct
This feature can be interpreted as the revenue generated for each product.

Correct
For example, the "house size" in square feet may be as high as 2,000, which is much larger than the feature "number of bedrooms" having a value between 1 and 5 for most houses in the modern era.

4. **1 / 1 point**

You are helping a grocery store predict its revenue, and have data on its items sold per week, and price per item. What could be a useful engineered feature?

- (●) For each product, calculate the number of items sold times price per item.
- ( ) For each product, calculate the number of items sold divided by the price per item.

Correct
This feature can be interpreted as the revenue generated for each product.

5. True/False? With polynomial regression, the predicted values f_w,b(x) does not necessarily have to be a straight line (or linear) function of the input feature x. **1 / 1 point**

- ( ) False
- (●) True

Correct
A polynomial function can be non-linear. This can potentially help the model to fit the training data better.