1. Multiple features

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
...	...	...	...	...

$i=2$

$j=1\ldots4$

$n=4$

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

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

$x_3^{(2)} = 2$

Model:

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

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

example

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

size #bedrooms #floors years base price

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

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

$\vec{w} = [w_1 \ \ w_2 \ \ w_3 \ \ldots \ w_n]$ 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)

2. Vectorization part 1 & 2

Parameters and features

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

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]

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

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

code: count from 0

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
```

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
```

Vectorization

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

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

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]
```

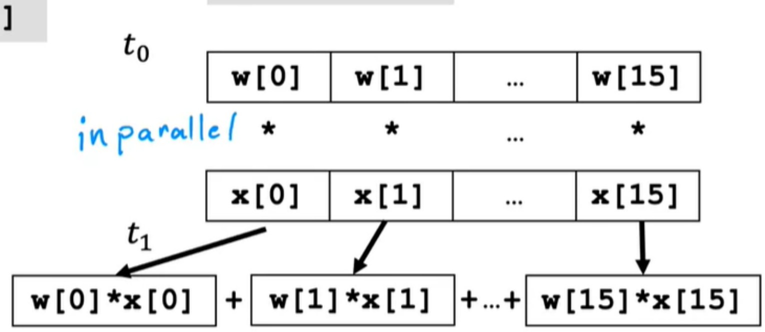
…

t_{15}

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

Vectorization

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



efficient → scale to large datasets

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 $j = 1 \ldots 16$

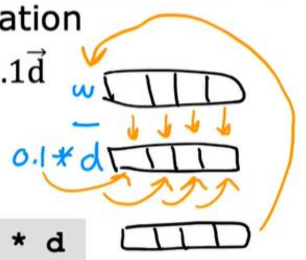
Without vectorization

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

```
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
```

3. Gradient descent for multiple linear regression

	Previous notation	Vector notation
Parameters	$w_1, \cdots, w_n$ b	vector of length n $\vec{w} = [w_1 \ \cdots \ w_n]$ 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	$J(w_1, \cdots, w_n, b)$	$J(\vec{w}, b)$

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:

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

Gradient descent

One feature

$$
\begin{aligned}
&\text{repeat} \{ \\
&\quad w = w - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) x^{(i)}} \\
&\qquad \to \frac{\partial}{\partial w} J(w, b) \\
&\quad b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) \\
&\quad \text{simultaneously update } w, b \\
&\}
\end{aligned}
$$

n features ($n \geq 2$)

$$
\begin{aligned}
&\text{repeat} \{ \\
&j=1 \quad w_1 = w_1 - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_1^{(i)}} \\
&\qquad \to \frac{\partial}{\partial w_1} J(\vec{w}, b) \\
&\quad \vdots \\
&j=n \quad w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) x_n^{(i)} \\
&\quad b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right) \\
&\quad \text{simultaneously update } w_j \text{ (for } j = 1, \cdots, n \text{) and } b \\
&\}
\end{aligned}
$$

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

4. Practice quiz 1

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).

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
...	...	...	...	...

30

Correct
Yes! $x_4^{(3)}$ is the 4th feature (4th column in the table) of the 3rd training example (3rd row in the table).

2.

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 $alpha$.

- ○ True
- ◉ False

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.

5. Feature scaling part 1

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?

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

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

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

$w_1 = 0.1$ (small), $w_2 = 50$ (large), $b = 50$

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

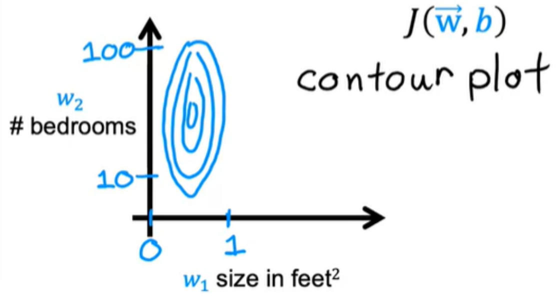
$\widehat{price} = \$500k$ more reasonable

Feature size and parameter size

	size of feature x_j	size of parameter w_j
size in feet^2	⟵⟶	↔
#bedrooms	↔	⟵⟶

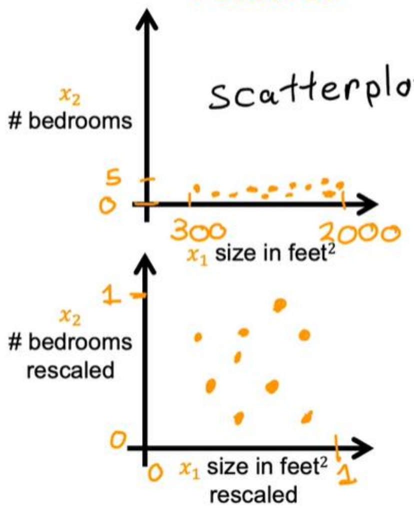
Features

Parameters

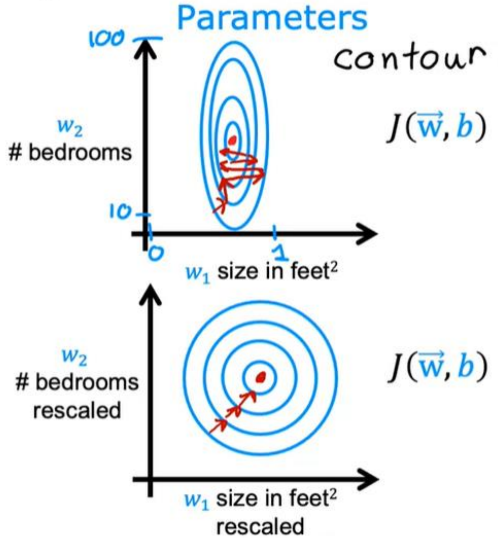


Feature size and gradient descent

Features

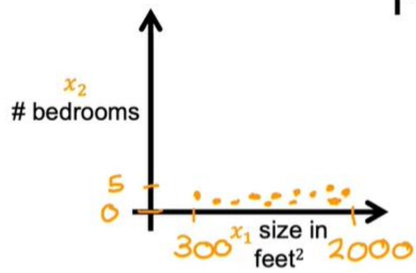


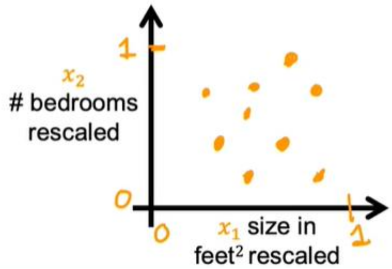
Parameters



6. Feature scaling part 2

Feature scaling



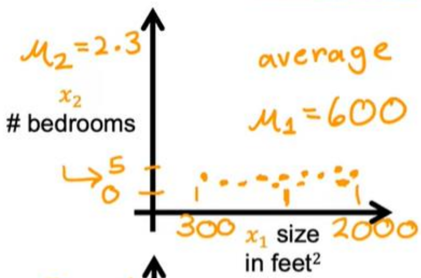


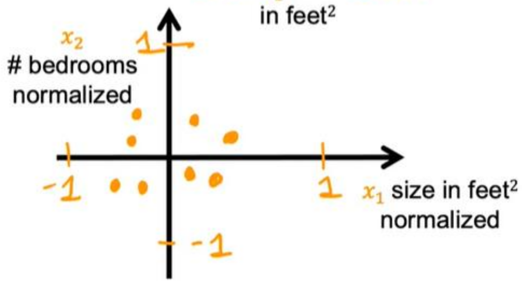
$300 \leq x_1 \leq 2000$ $\quad$ $0 \leq x_2 \leq 5$

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

$0.15 \leq x_{1,scaled} \leq 1$ $\quad$ $0 \leq x_{2,scaled} \leq 1$

Mean normalization



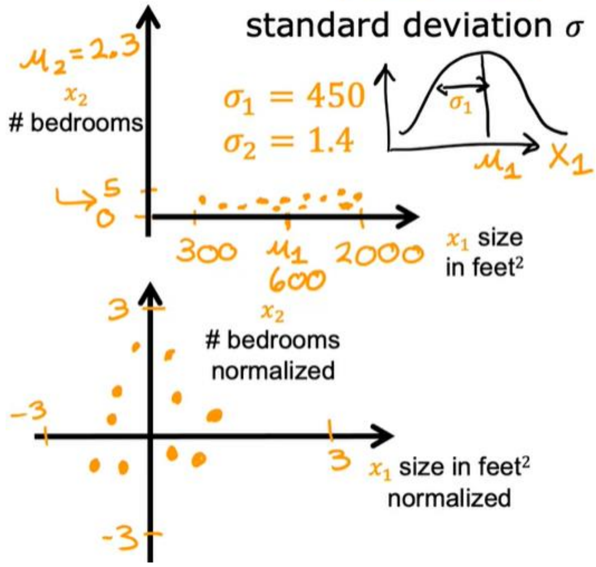


$300 \leq x_1 \leq 2000$ $\quad$ $0 \leq x_2 \leq 5$

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

$-0.18 \leq x_1 \leq 0.82$ $\quad$ $-0.46 \leq x_2 \leq 0.54$

Z-score normalization



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

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

$-0.67 \le x_1 \le 3.1$ $\quad$ $-1.6 \le x_2 \le 1.9$

Feature scaling

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

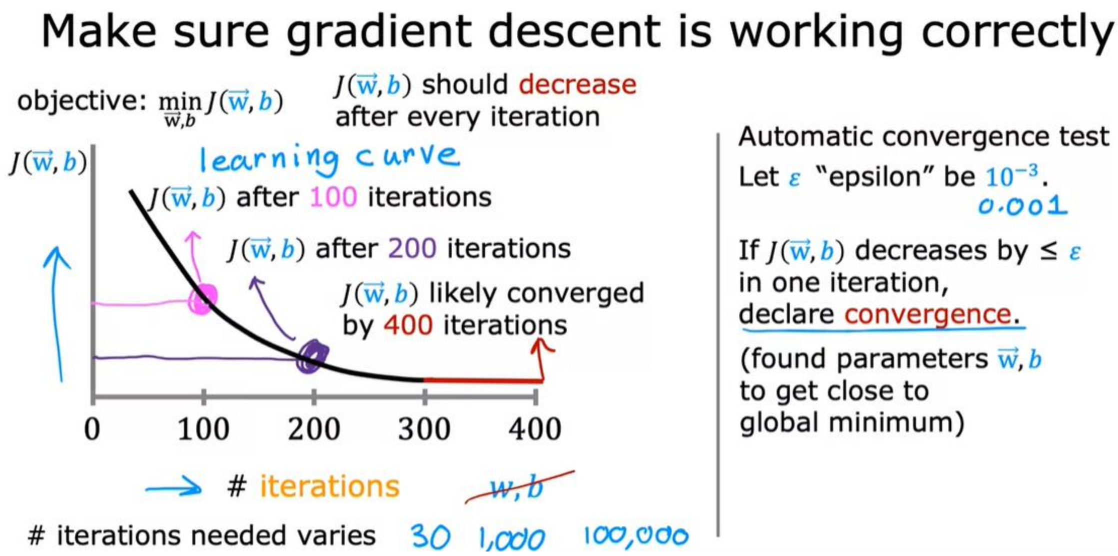
$-3 \le x_j \le 3$

$-0.3 \le x_j \le 0.3$

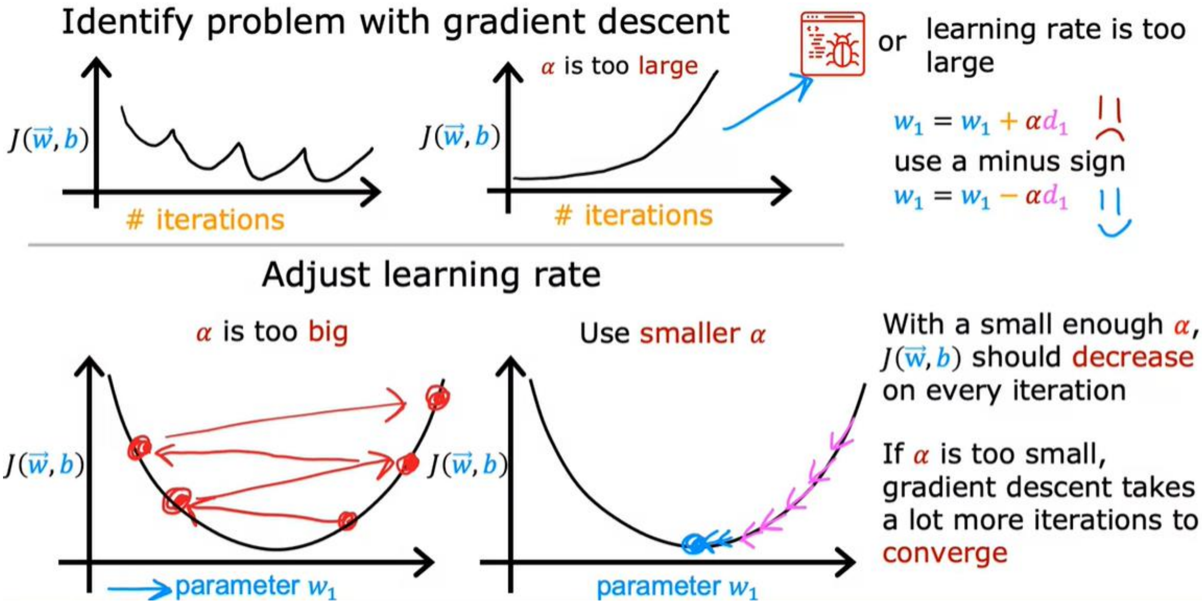
acceptable ranges

$0 \le x_1 \le 3$	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

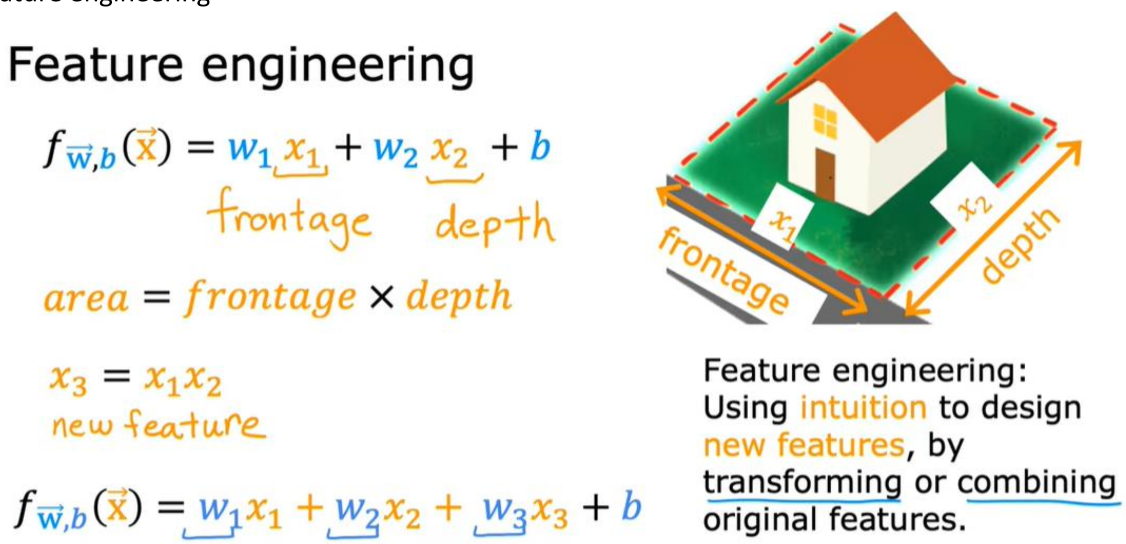
7. Checking gradient descent for convergence



8. Choosing the learning rate

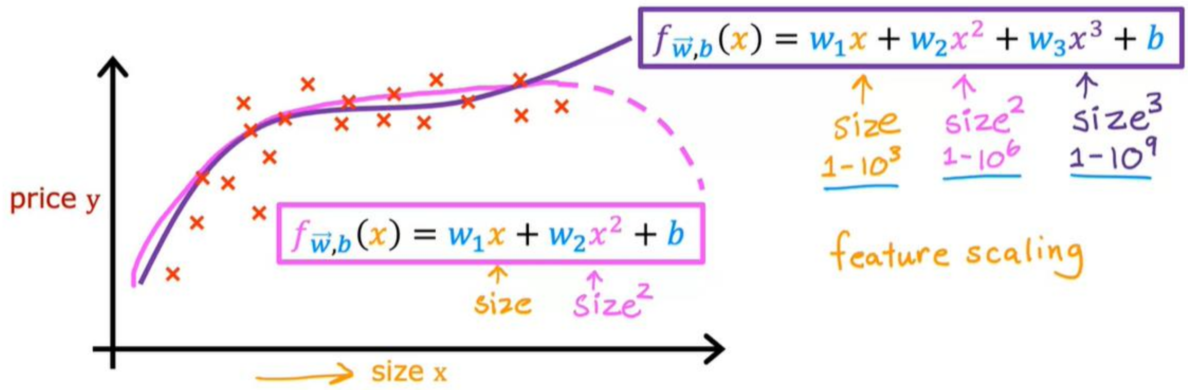


9. Feature engineering

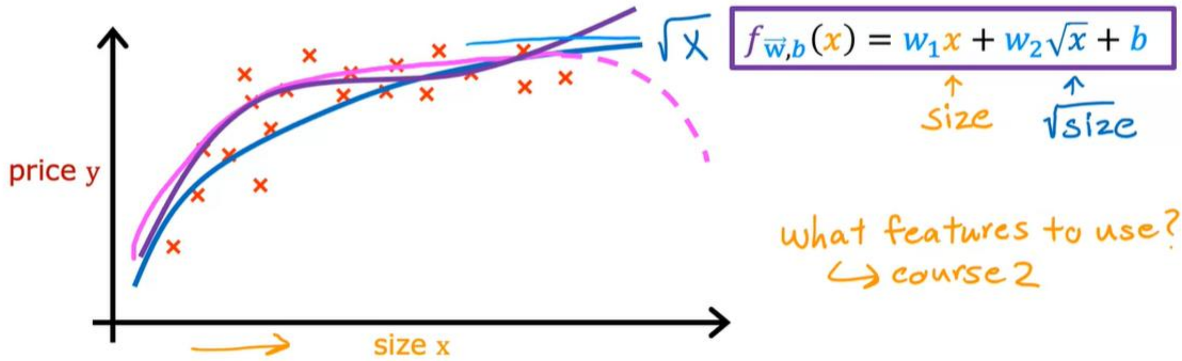


10. Polynomial regression

Polynomial regression



Choice of features



11. Practice quiz

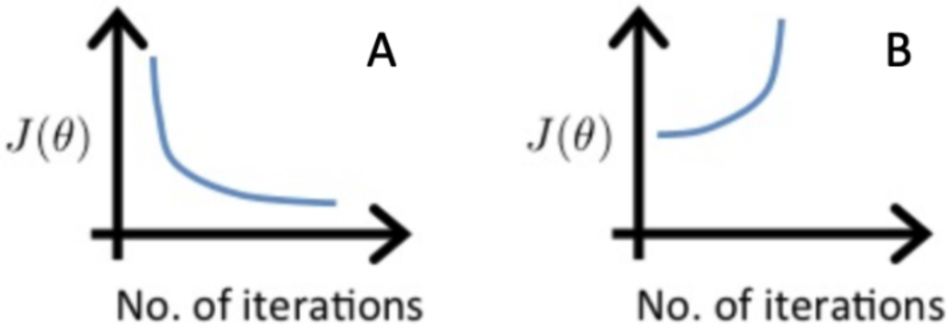
1.

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

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



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

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

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?

- ( ) 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.

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

- ○ False
- ● True

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