Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

Linear Regression with Multiple Variables

Multiple Features

Multiple features (variables)

one feature

Size in feet² (x)	Price (\$) in 1000's (y)
2104	400
1416	232
1534	315
852	178
…	…

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

Multiple features (variables)

Size in feet^2 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

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

$x_3^{(2)} = 2$

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

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

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

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

Linear Regression with Multiple Variables

Vectorization

Part 1

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]

```
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

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

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

Without vectorization

n=100,000

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

```
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]
```

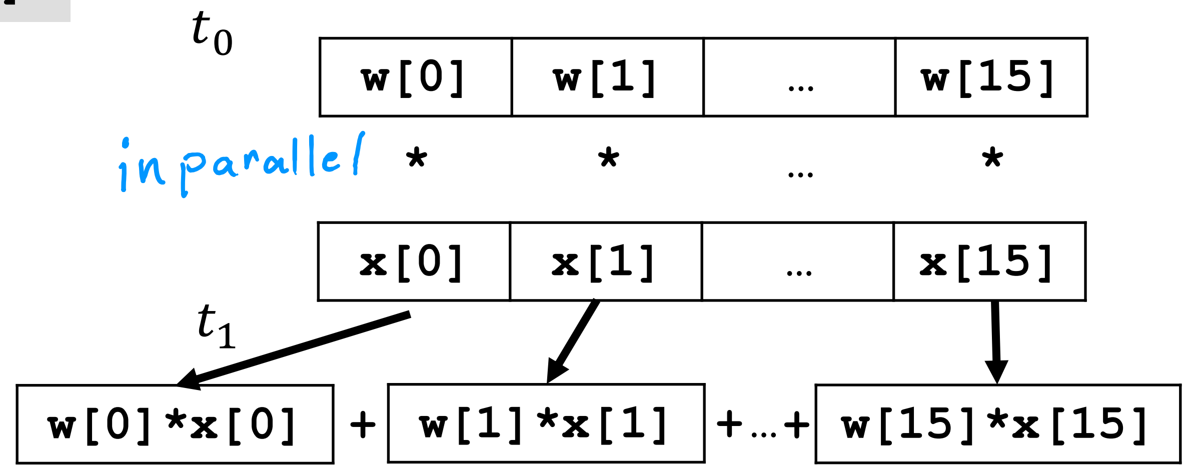
...

t_{15}

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

Vectorization

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



efficient → scale to large datasets

Gradient descent

$\vec{w} = (w_1 \quad w_2 \quad \cdots \quad w_{16})$ 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 $j = 1 \ldots 16$

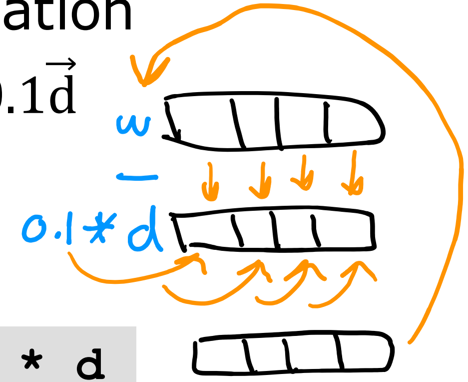
Without vectorization

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

```
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
```

Linear Regression with Multiple Variables

Gradient Descent for Multiple Regression

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

Gradient descent

Previous notation:

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\underbrace{w_1, \cdots, w_n}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\underbrace{w_1, \cdots, w_n}, b)$$

}

Vector notation:

repeat {

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

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

}

Gradient descent

One feature

repeat {

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

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

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

simultaneously update w, b

}

n features ($n \geq 2$)

repeat {

$j=1$

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

$$\to \frac{\partial}{\partial w_1} J(\vec{\mathrm{w}},b)$$

$\vdots$

$j=n$

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

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

simultaneously update

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

}

Practical Tips for Linear Regression

Feature Scaling
Part 1

Feature and parameter values

$\widehat{price} = \underset{\text{size}}{w_1 x_1} + \underset{\text{\# bedrooms}}{w_2 x_2} + b$

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?

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

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

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

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

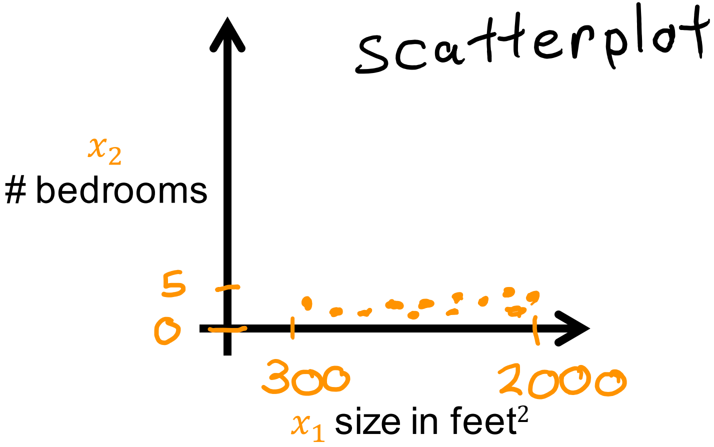
$\widehat{price} = \underbrace{0.1 * 2000k}_{200K} + \underbrace{50 * 5}_{250K} + \underbrace{50}_{50K}$

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

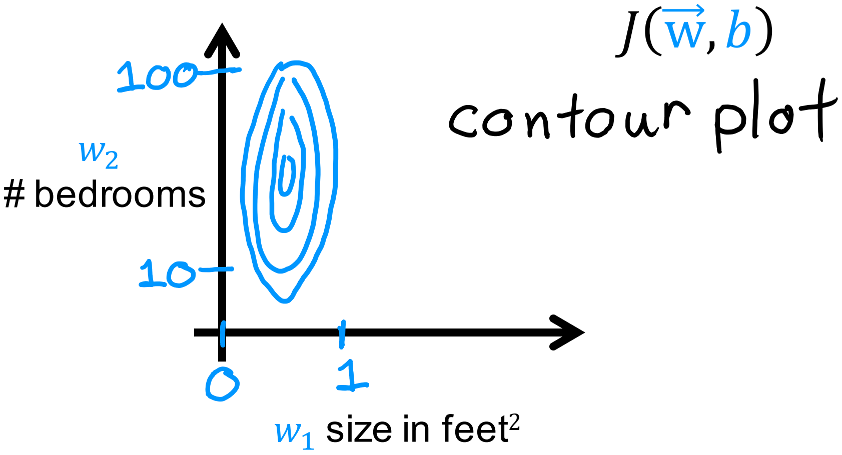
Feature size and parameter size

	size of feature x_j	size of parameter w_j
size in feet^2	⟷ (long)	⟷ (short)
#bedrooms	⟷ (short)	⟷ (long)

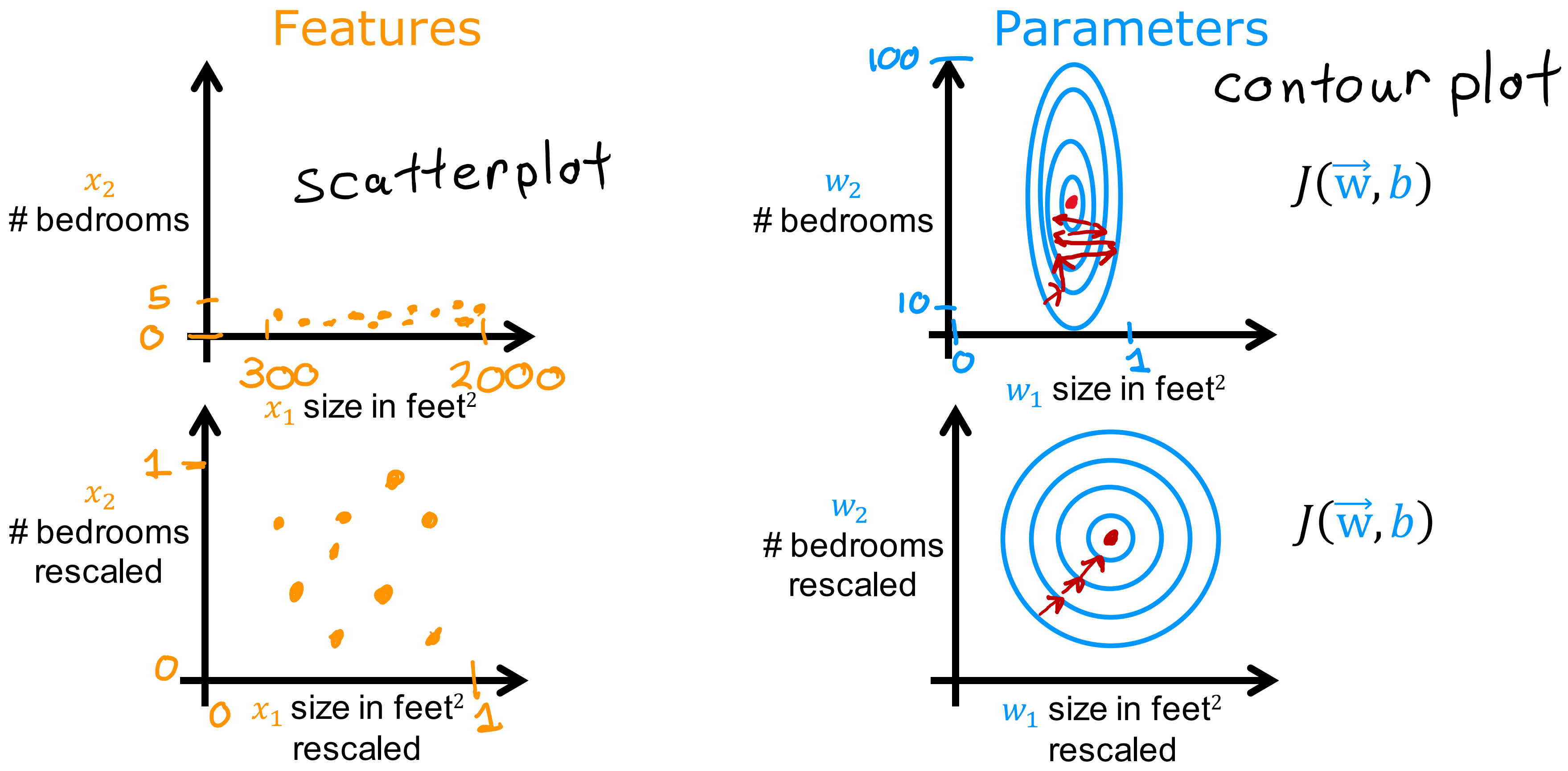
Features



Parameters



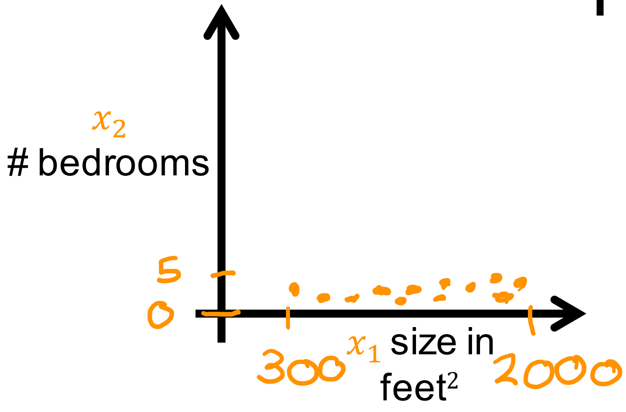
Feature size and gradient descent

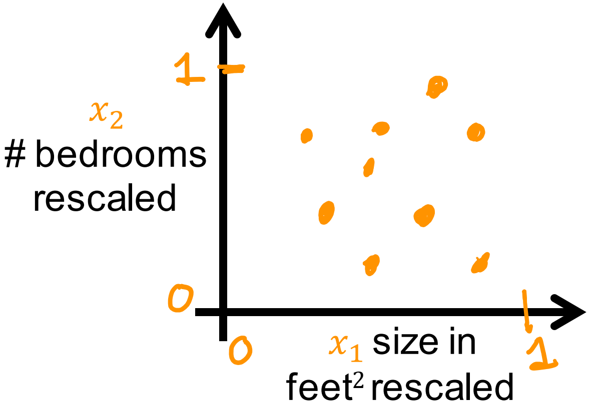


Practical Tips for Linear Regression

Feature Scaling
Part 2

Feature scaling



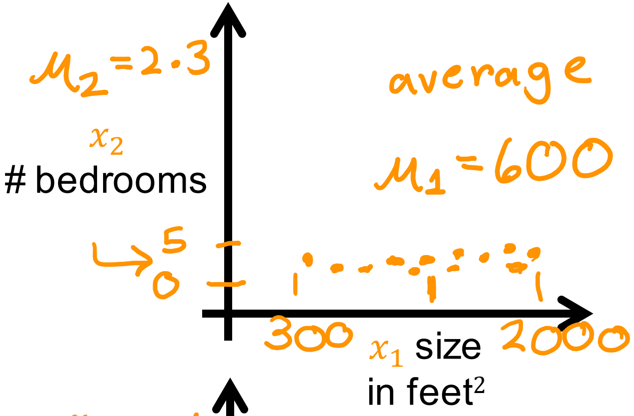


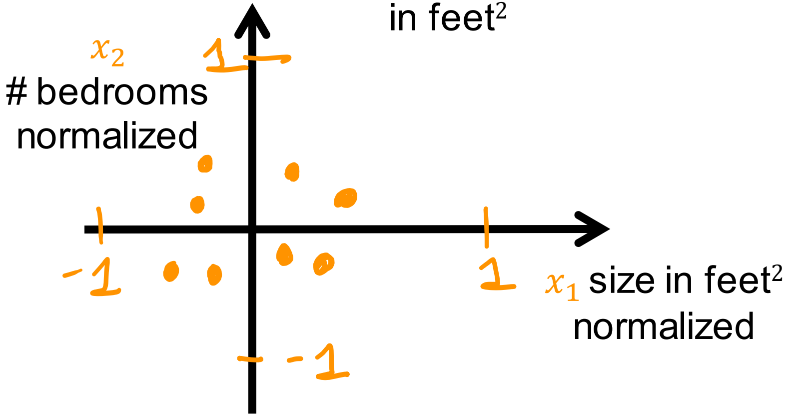
$$300 \le x_1 \le 2000 \qquad 0 \le x_2 \le 5$$

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

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

Mean normalization





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

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

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

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 $\rightarrow$ rescale

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

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

Practical Tips for Linear Regression

Checking Gradient Descent for Convergence

Gradient descent

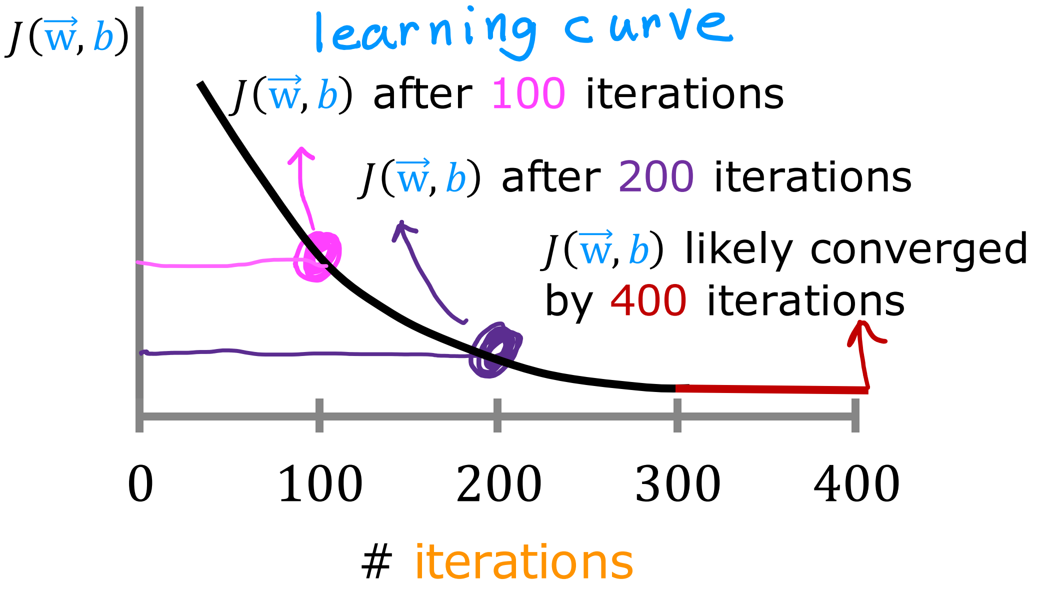
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{\mathrm{w}}, b)$$

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

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

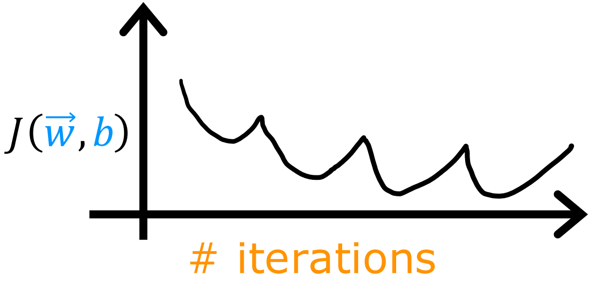
If $J(\vec{w},b)$ decreases by $\leq \varepsilon$ in one iteration, declare convergence.

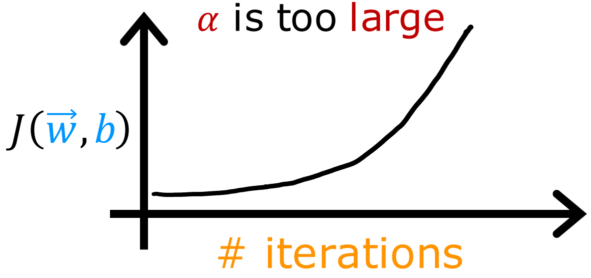
(found parameters $\vec{w},b$ to get close to global minimum)

Practical Tips for Linear Regression

Choosing the Learning Rate

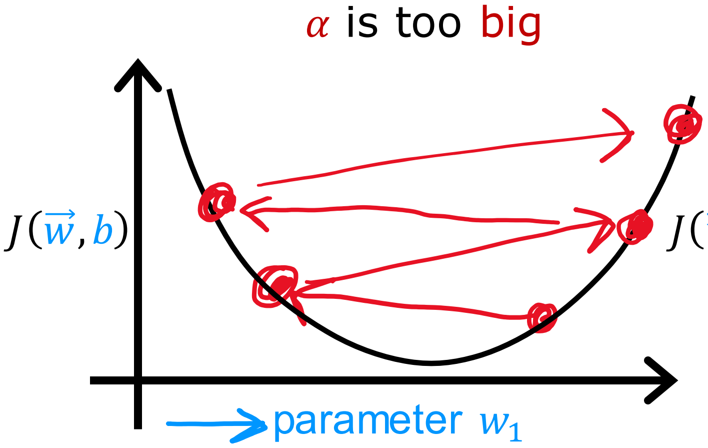
Identify problem with gradient descent

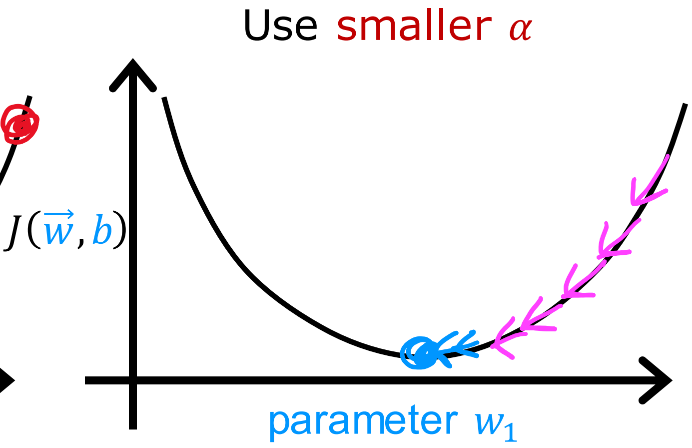




learning rate is too large

Adjust learning rate





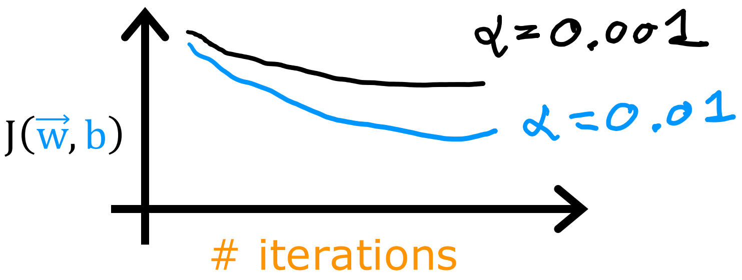
With a small enough α, $J(\vec{w}, b)$ should decrease on every iteration

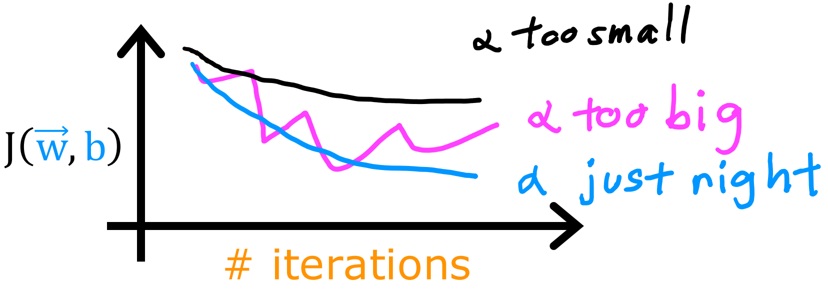
If α is too small, gradient descent takes a lot more iterations to converge

Values of α to try:

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

3X ≈3X 3X ≈3X 3X ≈3X





Practical Tips for Linear Regression

Feature Engineering

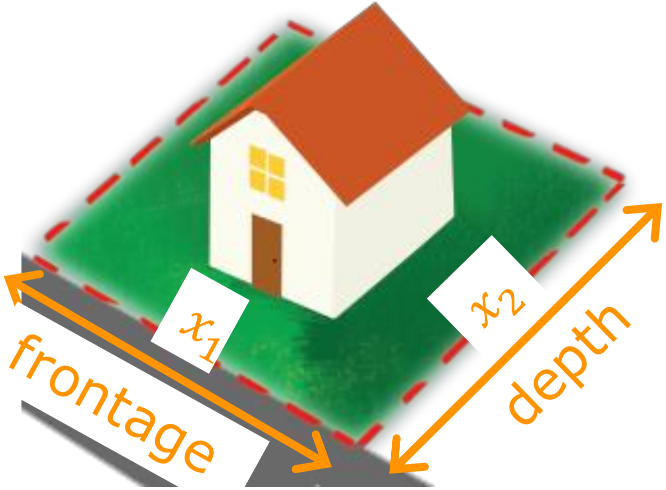
Feature engineering

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

$$area = frontage \times depth$$

$$x_3 = x_1 x_2$$
new feature

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

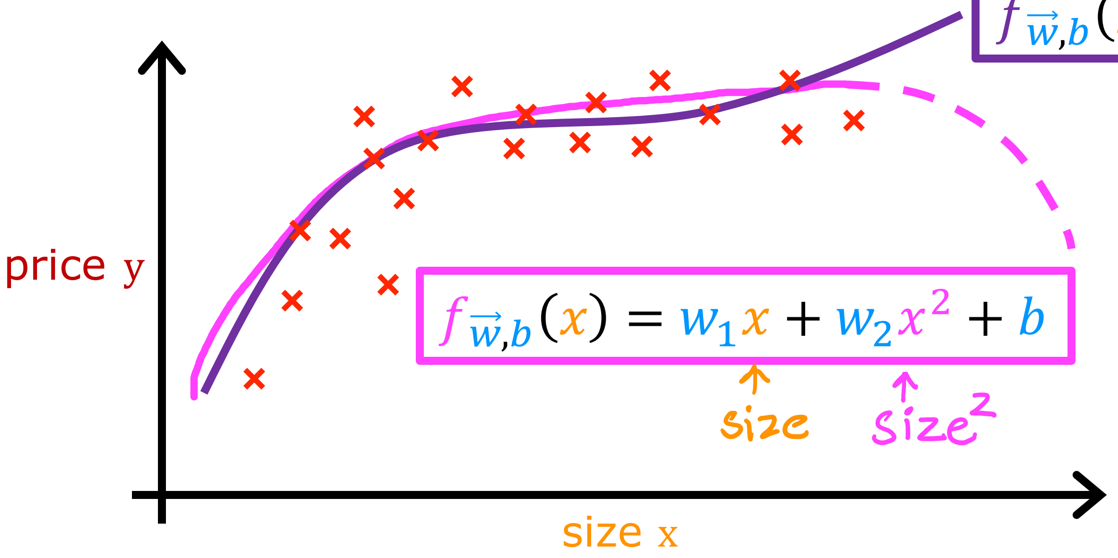


Feature engineering:
Using intuition to design new features, by transforming or combining original features.

Practical Tips for Linear Regression

Polynomial Regression

Polynomial regression



$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$$

size size^2

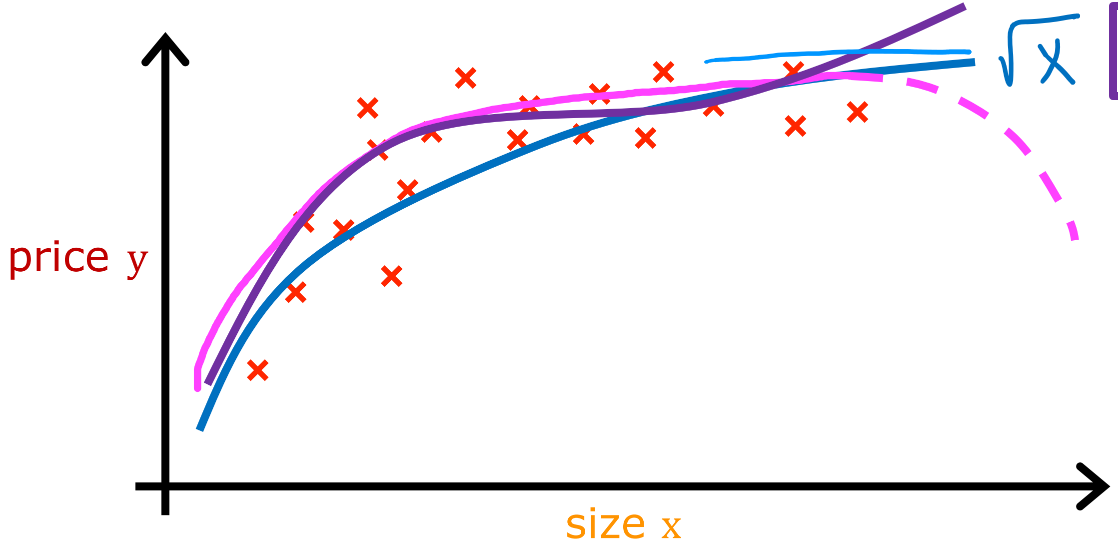
$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + b$$

size size^2 size^3

1-10^3 1-10^6 1-10^9

feature scaling

Choice of features



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

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