

# Copyright Notice

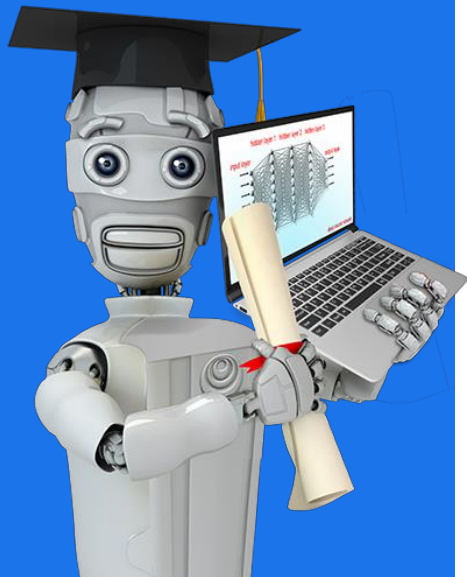
These slides are distributed under the Creative Commons License.

[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>

Stanford  
ONLINE

DeepLearning.AI



---

# Linear Regression with Multiple Variables

## Multiple Features

# Multiple features (variables)

one  
feature



| Size in feet <sup>2</sup> ( $x$ ) | Price (\$) in 1000's ( $y$ ) |
|-----------------------------------|------------------------------|
| 2104                              | 400                          |
| 1416                              | 232                          |
| 1534                              | 315                          |
| 852                               | 178                          |
| ...                               | ...                          |



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

# Multiple features (variables)

|       | Size in<br>feet <sup>2</sup><br>$x_1$ | Number of<br>bedrooms<br>$x_2$ | Number of<br>floors<br>$x_3$ | Age of home<br>in years<br>$x_4$ | Price (\$) in<br>\$1000's |
|-------|---------------------------------------|--------------------------------|------------------------------|----------------------------------|---------------------------|
|       | 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

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

$x_3^{(2)} = 2$

# Model:

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

example

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

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

size # bedrooms # floors years base price

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

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

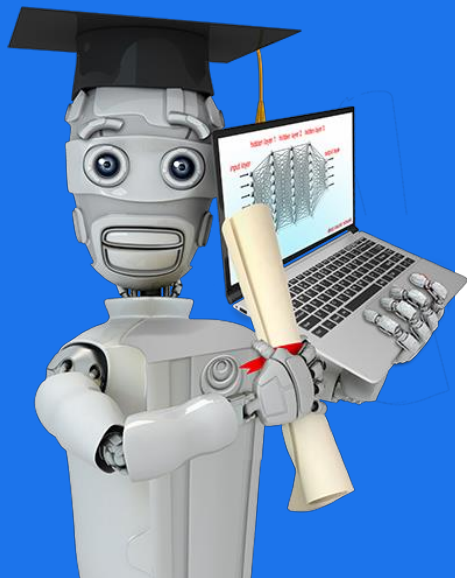
$\vec{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$  parameters of the model  
 $b$  is a number  
 vector  $\vec{x} = [x_1 \ x_2 \ x_3 \ \dots \ x_n]$

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

dot product      multiple linear regression  
 (not multivariate regression)

Stanford  
ONLINE

DeepLearning.AI



# Linear Regression with Multiple Variables

---

## Vectorization Part 1

## 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
```

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

```
x = np.array([10, 20, 30])
```

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

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

$\sum_{j=1}^n \rightarrow j=1 \dots n$   
 $1, 2, 3$

$\text{range}(0, n) \rightarrow j=0 \dots n-1$

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



Vectorization

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

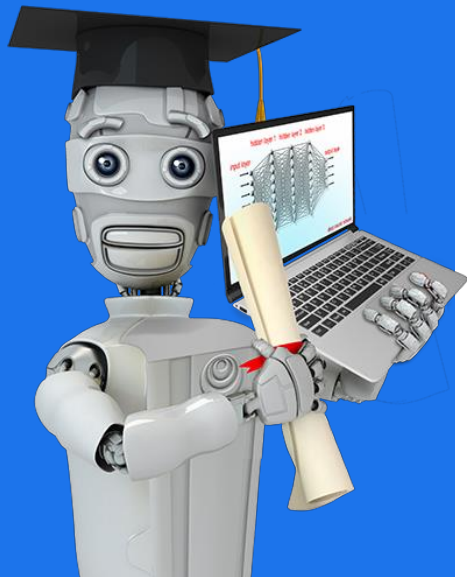
```
f = np.dot(w, x) + b
```





Stanford  
ONLINE

DeepLearning.AI



# Linear Regression with Multiple Variables

---

## Vectorization Part 2

## Without vectorization

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

$t_0$

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

$t_1$

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

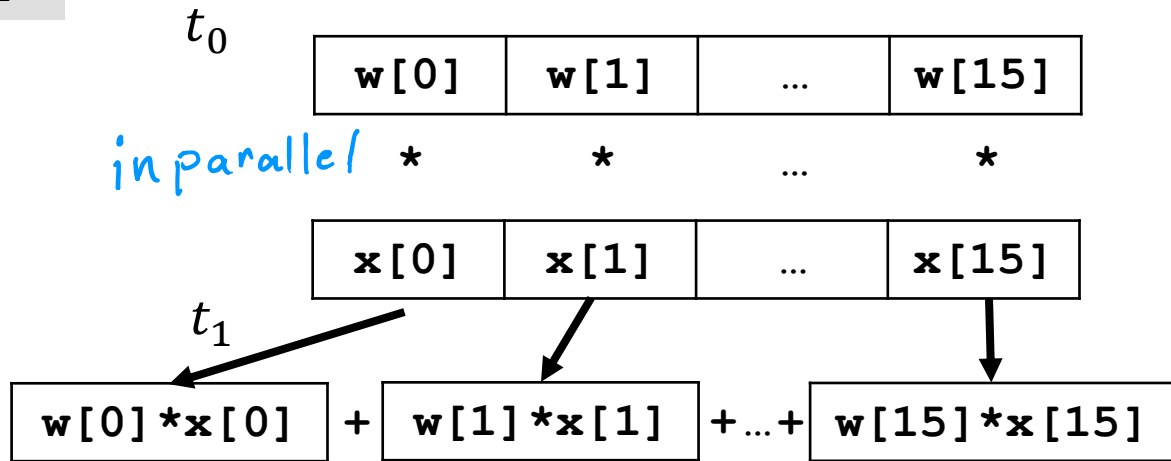
...

$t_{15}$

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

## Vectorization

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



*efficient → scale to large datasets*

Gradient descent  $\vec{w} = (w_1 \quad w_2 \quad \dots \quad w_{16})$  ~~b~~ parameters  
derivatives  $\vec{d} = (d_1 \quad d_2 \quad \dots \quad d_{16})$

```
w = np.array([0.5, 1.3, ... 3.4])
```

```
d = np.array([0.3, 0.2, ... 0.4])
```

compute  $w_j = w_j - \underbrace{0.1}_{\text{learning rate } \alpha} d_j$  for  $j = 1 \dots 16$

Without vectorization

$$w_1 = w_1 - 0.1d_1$$

$$w_2 = w_2 - 0.1d_2$$

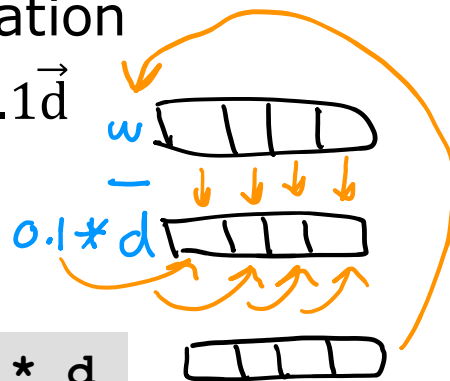
$$\vdots$$

$$w_{16} = w_{16} - 0.1d_{16}$$

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

With vectorization

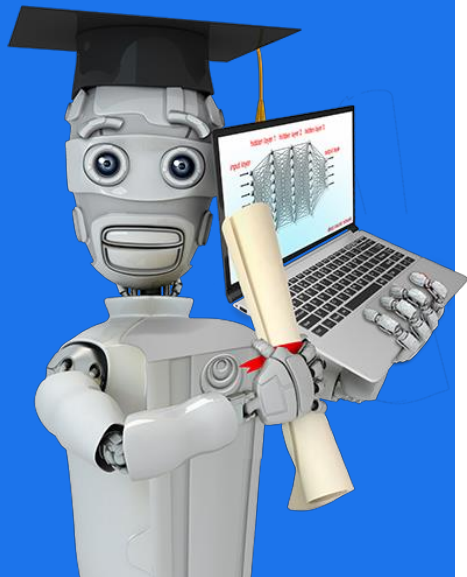
$$\vec{w} = \vec{w} - 0.1\vec{d}$$



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

Stanford  
ONLINE

DeepLearning.AI



# Linear Regression with Multiple Variables

---

## Gradient Descent for Multiple Regression

## Previous notation

Parameters

$$w_1, \dots, w_n$$

$$b$$

Model

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + \dots + w_n x_n + b$$

Cost function

$$J(\underbrace{w_1, \dots, w_n}_{\text{vector}}, b)$$

Gradient descent

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\underbrace{w_1, \dots, w_n}_{\text{vector}}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\underbrace{w_1, \dots, w_n}_{\text{vector}}, b)$$

}

## Vector notation

↖ vector of length n

$$\vec{w} = [w_1 \quad \dots \quad w_n]$$

$b$  still a number

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

↗ dot product

$$J(\underbrace{\vec{w}}_{\text{vector}}, b)$$

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\underbrace{\vec{w}}_{\text{vector}}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\underbrace{\vec{w}}_{\text{vector}}, b)$$

}

# Gradient descent

One feature

repeat {

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

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

simultaneously update  $w, b$

}

$n$  features ( $n \geq 2$ )

repeat {

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

⋮

$j=n$

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

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

simultaneously update

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

}

# An alternative to gradient descent

## → Normal equation

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

### Disadvantages

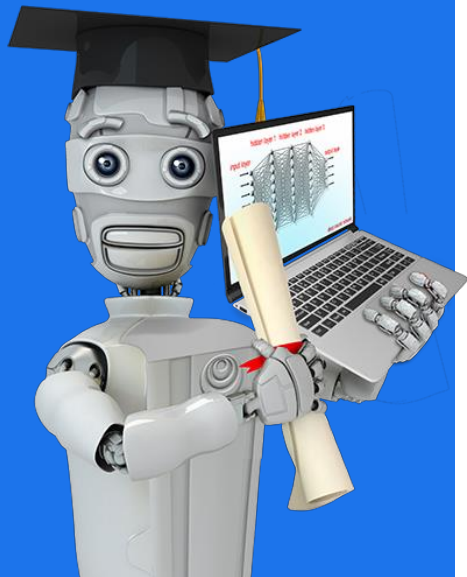
- Doesn't generalize to other learning algorithms.
- Slow when number of features is large ( $> 10,000$ )

### 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  $w, b$

Stanford  
ONLINE

DeepLearning.AI



# Practical Tips for Linear Regression

---

## Feature Scaling Part 1



# Feature and parameter values

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

$x_1$ : size (feet<sup>2</sup>) range: 300 – 2,000       $x_2$ : # bedrooms range: 0 – 5

size      # bedrooms      large      small

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

size of the parameters  $w_1, w_2$ ?

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

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

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

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

small      large

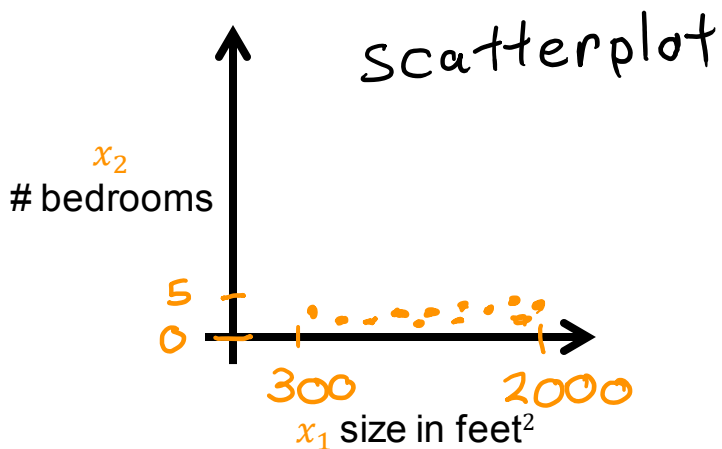
$$\widehat{price} = \underbrace{0.1 * 2000k}_{200k} + \underbrace{50 * 5}_{250k} + \underbrace{50}_{50k}$$

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

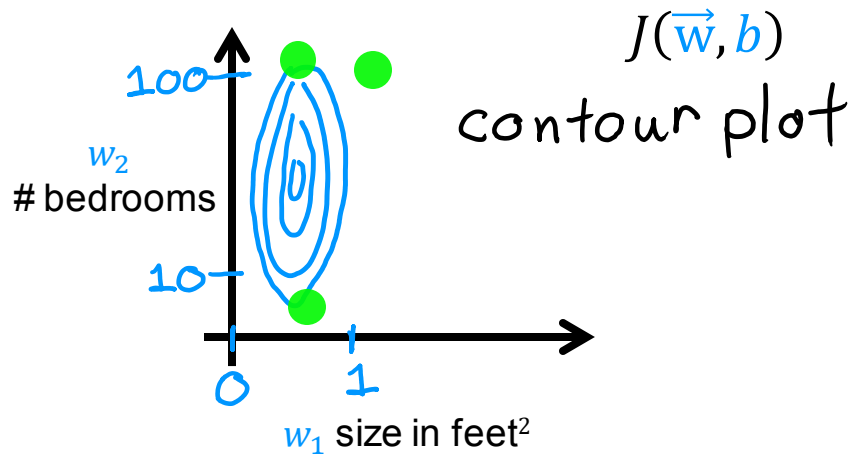
# Feature size and parameter size

|                           | size of feature $x_j$ | size of parameter $w_j$ |
|---------------------------|-----------------------|-------------------------|
| size in feet <sup>2</sup> | ←→                    | ←→                      |
| # bedrooms                | ←→                    | ←→                      |

## Features

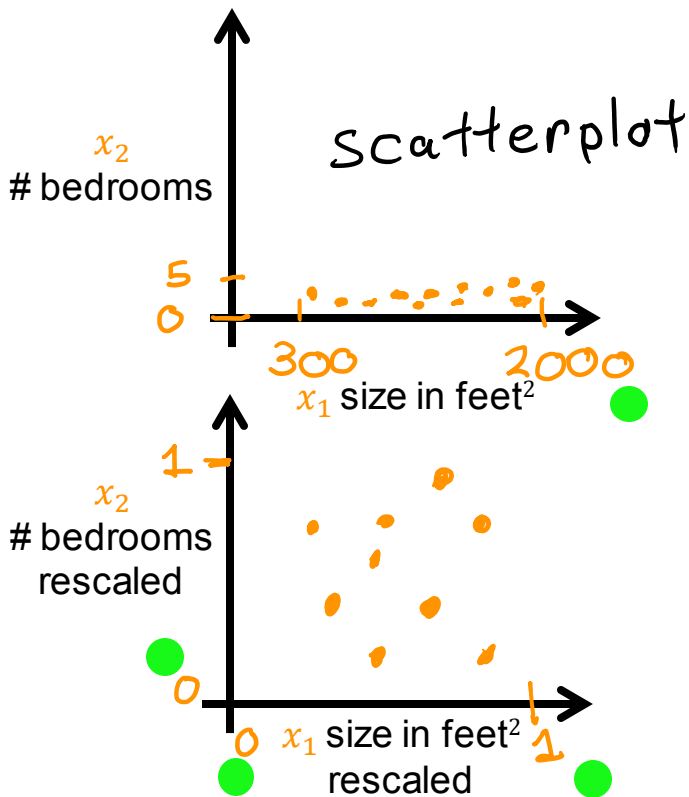


## Parameters

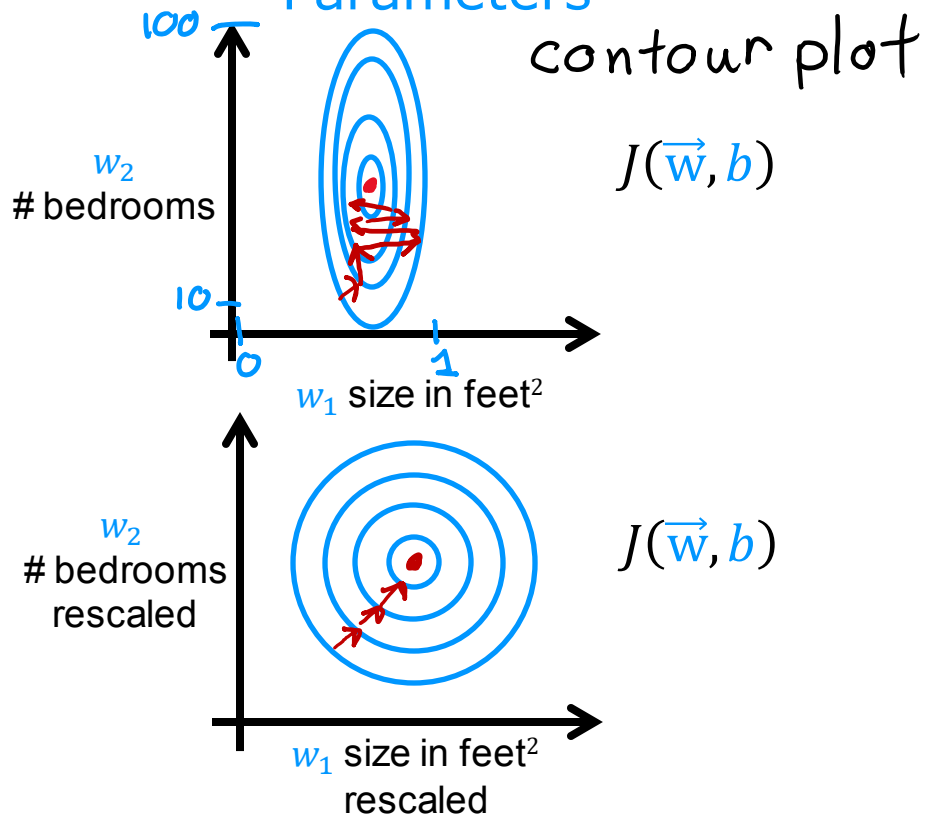


# Feature size and gradient descent

Features

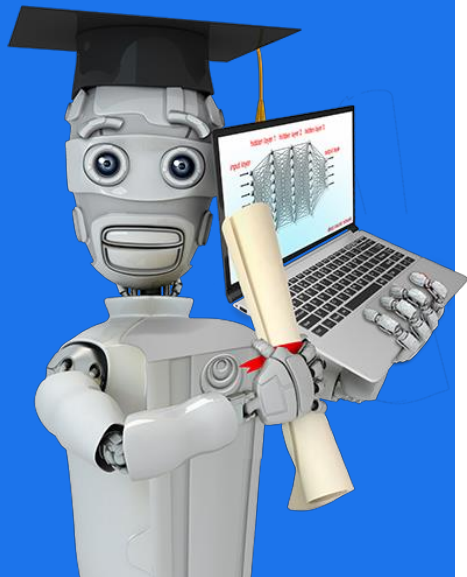


Parameters



Stanford  
ONLINE

DeepLearning.AI

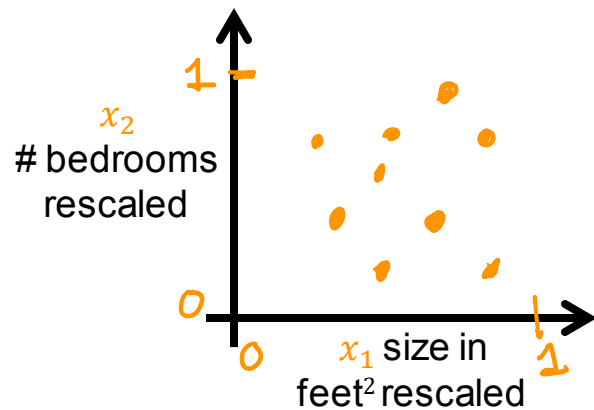
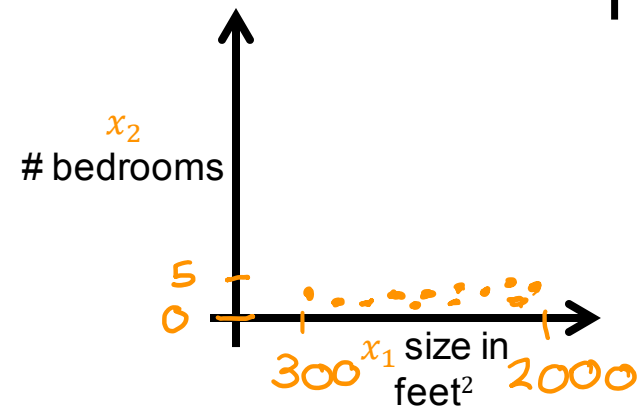


# Practical Tips for Linear Regression

---

## Feature Scaling Part 2

# Feature scaling



$$300 \leq x_1 \leq 2000$$

$$x_{1,scaled} = \frac{x_1}{2000}$$

*max*

$$0.15 \leq x_{1,scaled} \leq 1$$

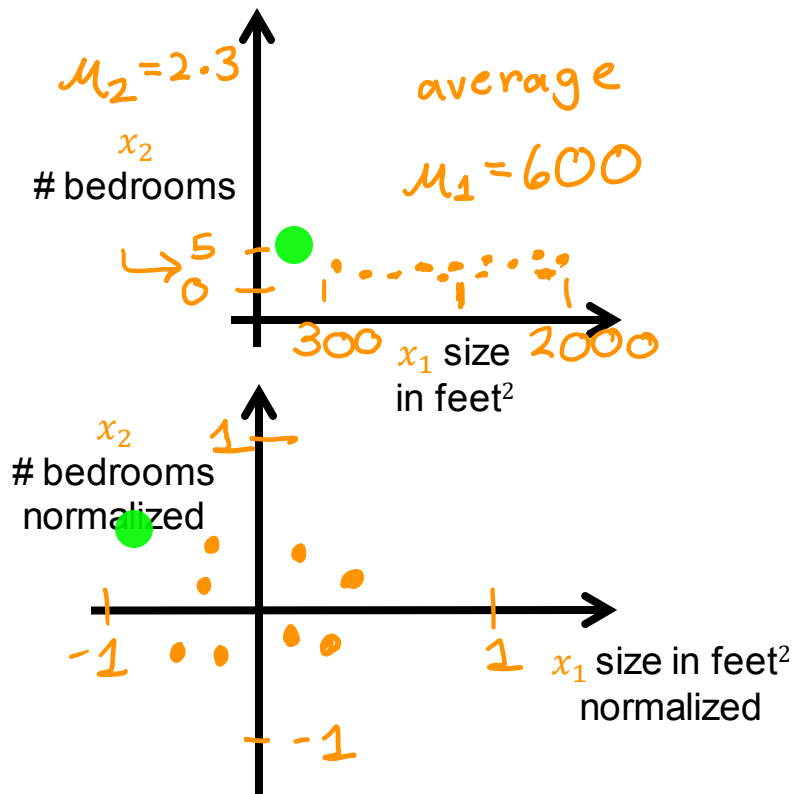
$$0 \leq x_2 \leq 5$$

$$x_{2,scaled} = \frac{x_2}{5}$$

*max*

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

# Mean normalization



$$300 \leq x_1 \leq 2000$$

$$x_1 = \frac{x_1 - \mu_1}{2000 - 300}$$

max-min

$$-0.18 \leq x_1 \leq 0.82$$

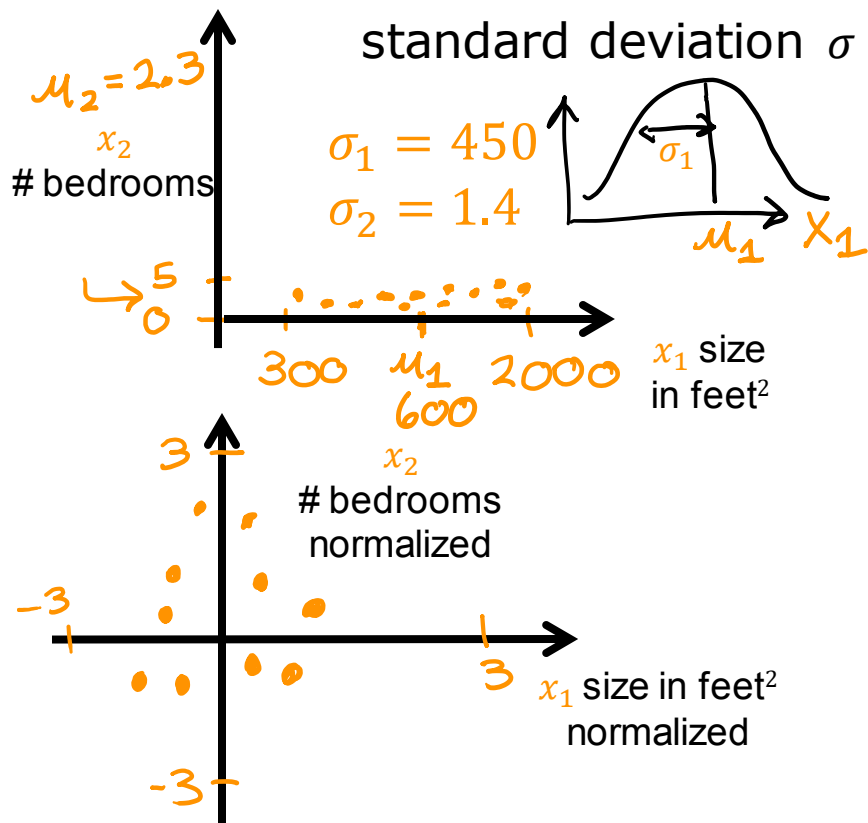
$$0 \leq x_2 \leq 5$$

$$x_2 = \frac{x_2 - \mu_2}{5 - 0}$$

max-min

$$-0.46 \leq x_2 \leq 0.54$$

# Z-score normalization



$$300 \leq x_1 \leq 2000$$

$$0 \leq x_2 \leq 5$$

$$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$$

$$x_2 = \frac{x_2 - \mu_2}{\sigma_2}$$

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

# Feature scaling

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

$-3 \leq x_j \leq 3$   
 $-0.3 \leq x_j \leq 0.3$  } acceptable ranges

$$0 \leq x_1 \leq 3$$

okay, no rescaling

$$-2 \leq x_2 \leq 0.5$$

okay, no rescaling

$$-100 \leq x_3 \leq 100$$

too large  $\rightarrow$  rescale

$$-0.001 \leq x_4 \leq 0.001$$

too small  $\rightarrow$  rescale

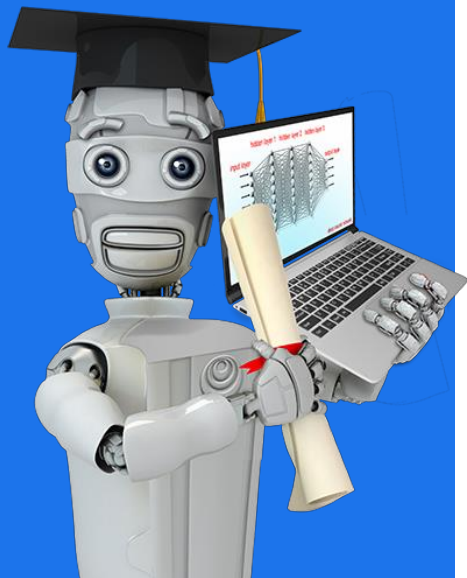
$$98.6 \leq x_5 \leq 105$$

too large  $\rightarrow$  rescale



Stanford  
ONLINE

DeepLearning.AI



# Practical Tips for Linear Regression

---

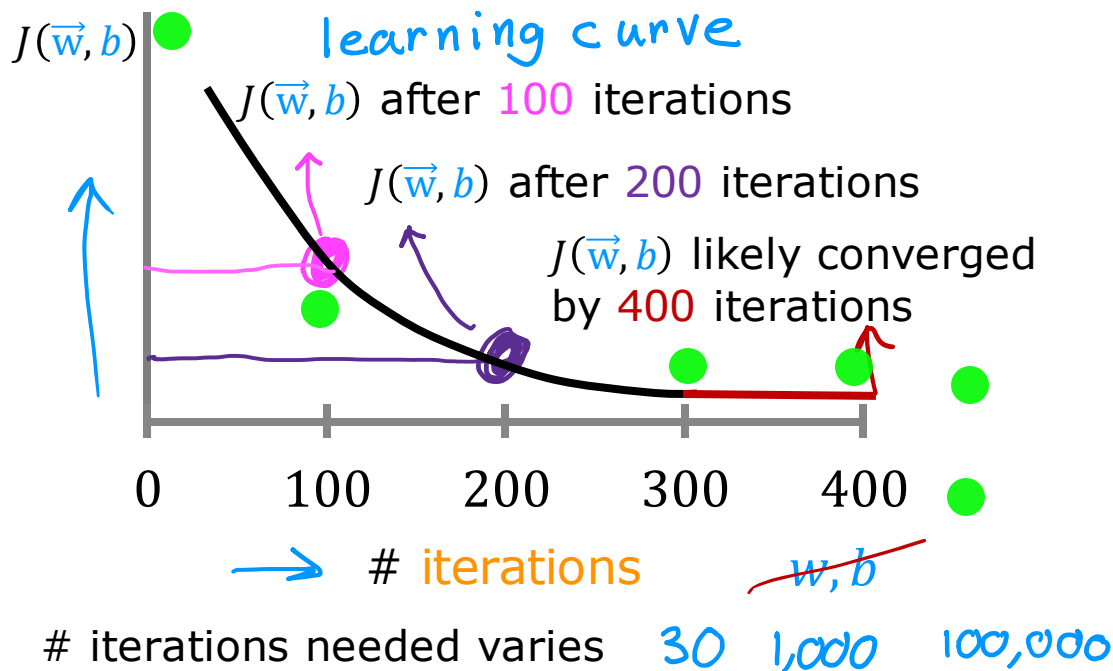
## Checking Gradient Descent for Convergence

# Gradient descent

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

# 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



Automatic convergence test

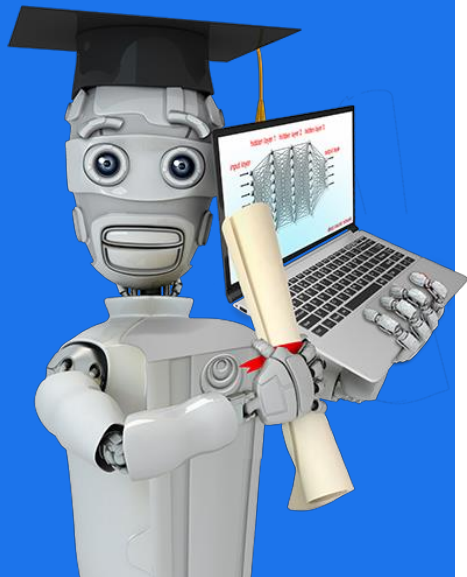
Let  $\epsilon$  "epsilon" be  $10^{-3}$ .  
**0.001**

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

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

Stanford  
ONLINE

DeepLearning.AI

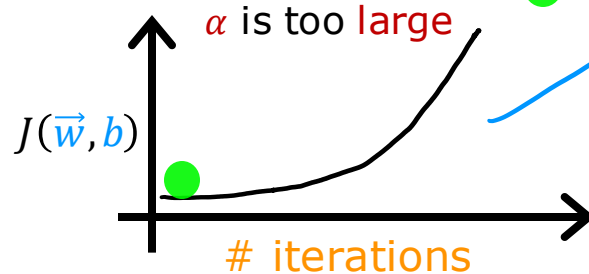
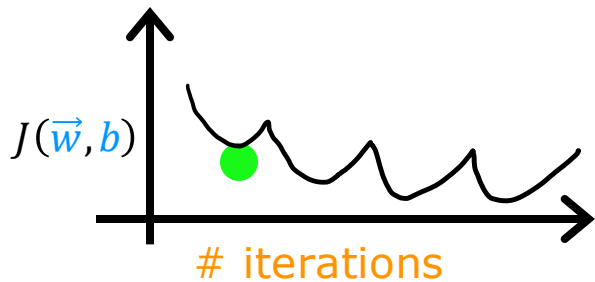


# Practical Tips for Linear Regression

---

## Choosing the Learning Rate

# Identify problem with gradient descent



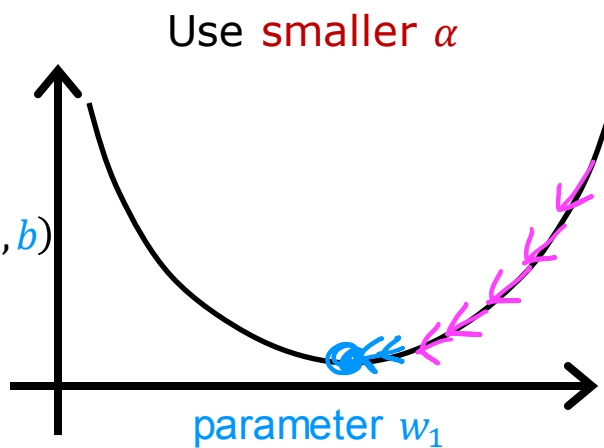
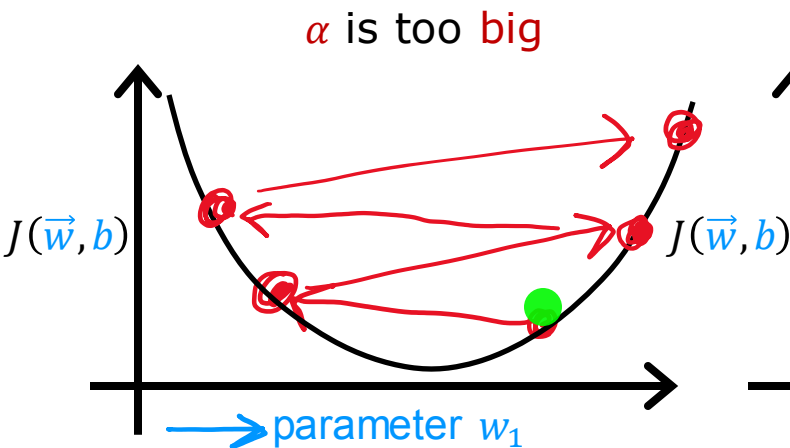
or learning rate is too large

$$w_1 = w_1 + \alpha d_1$$

use a minus sign

$$w_1 = w_1 - \alpha d_1$$

## Adjust learning rate

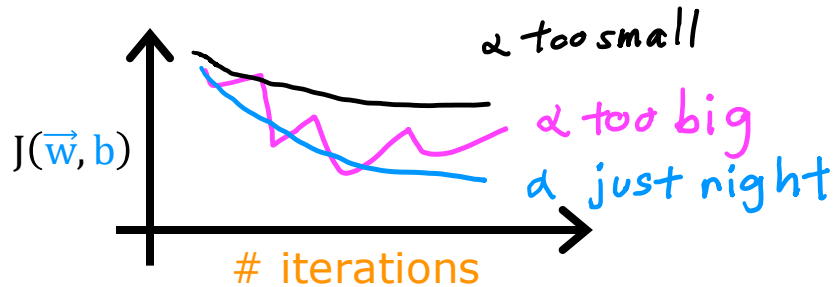
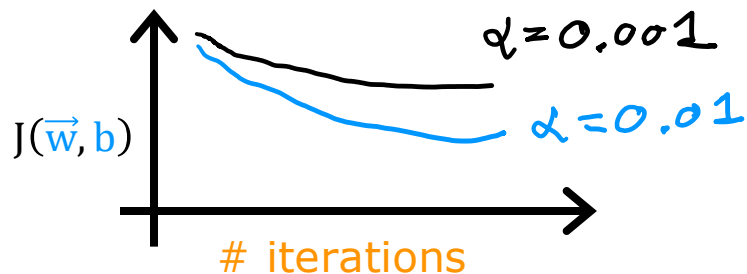


With a small enough  $\alpha$ ,  $J(\vec{w}, b)$  should **decrease** on every iteration

If  $\alpha$  is too small, gradient descent takes a lot more iterations to **converge**

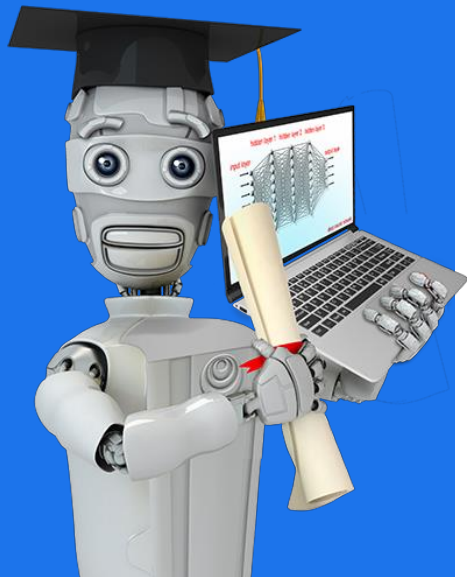
Values of  $\alpha$  to try:

... 0.001 0.003 0.01 0.03 0.1 0.3 1 ...  
           $\nearrow$         $\nearrow$         $\nearrow$         $\nearrow$         $\nearrow$   
           $3\times$         $\approx 3\times$     $3\times$         $\approx 3\times$     $3\times$         $\approx 3\times$



Stanford  
ONLINE

DeepLearning.AI



# Practical Tips for Linear Regression

---

## Feature Engineering

# Feature engineering

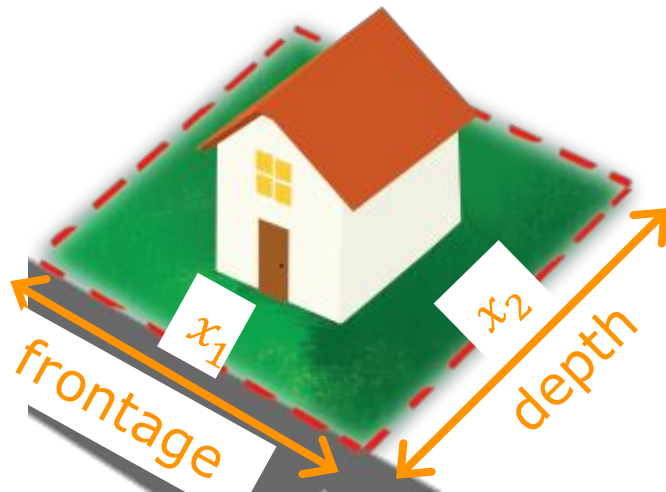
$$f_{\vec{w},b}(\vec{X}) = \underbrace{w_1}_{\text{frontage}} \underbrace{x_1}_{\text{depth}} + \underbrace{w_2}_{\text{depth}} \underbrace{x_2}_{\text{depth}} + b$$

$$\text{area} = \text{frontage} \times \text{depth}$$

$$x_3 = x_1 x_2$$

new feature

$$f_{\vec{w},b}(\vec{X}) = \underbrace{w_1}_{\text{frontage}} x_1 + \underbrace{w_2}_{\text{depth}} x_2 + \underbrace{w_3}_{\text{depth}} x_3 + b$$

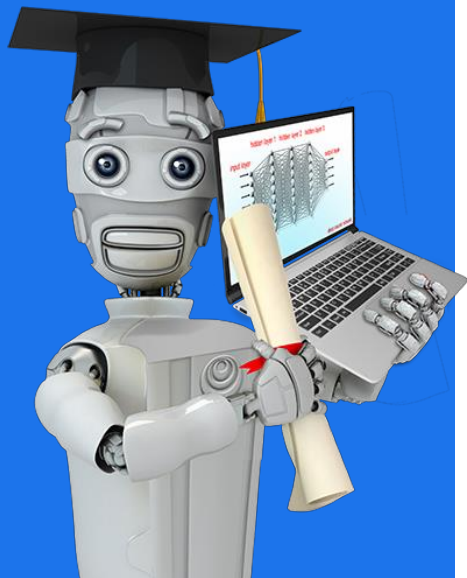


Feature engineering:  
Using **intuition** to design  
**new features**, by  
transforming or combining  
original features.



Stanford  
ONLINE

DeepLearning.AI

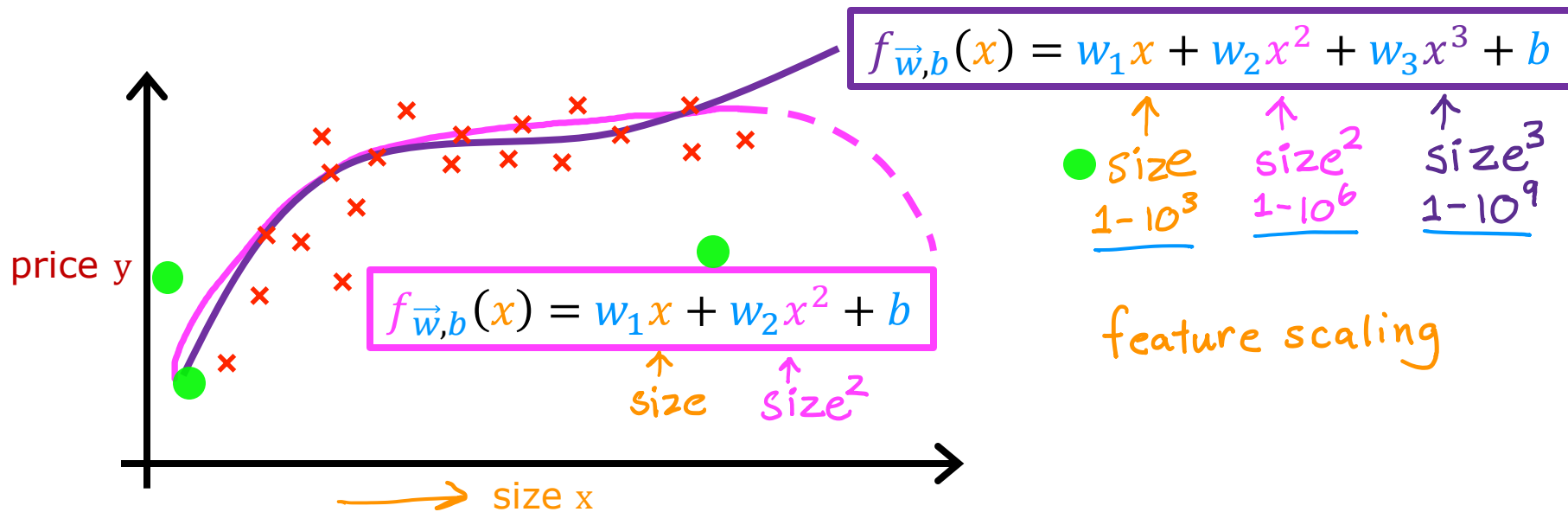


# Practical Tips for Linear Regression

---

## Polynomial Regression

# Polynomial regression



# Choice of features

