

## Multiple Features

適合多樣量、多特徵的線性迴歸表示法

Multiple features list

size $x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home $x_4$
2104	5	1	45
(2) $\rightarrow 1416$	3	2	40
:	:	:	:

$n = \text{number of features}$

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

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

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \quad x_3^{(2)} = 2$$

hypothesis :

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

let  $x_0^{(i)} = 1$

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n \times 1} \in \mathbb{R}^{n+1}$$

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}_{n \times 1} \in \mathbb{R}^{n+1}$$

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

$$= \theta^T x$$

Multiple linear regression

Gradient Descent in Practice

for Multiple variables

hypothesis :  $h_{\theta}(x) = \theta^T x$

parameters :  $\theta = n+1$  dimension vector

cost function :  $J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$

$$= \frac{1}{2m} \sum_{i=1}^m (\theta^T x^{(i)} - y^{(i)})^2$$

$$\text{gradient descent: } \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

simultaneously update  $\theta_j$  for  $j = 0, \dots, n$

$$x_0^{(i)} = 1$$

feature scaling 特徵縮放

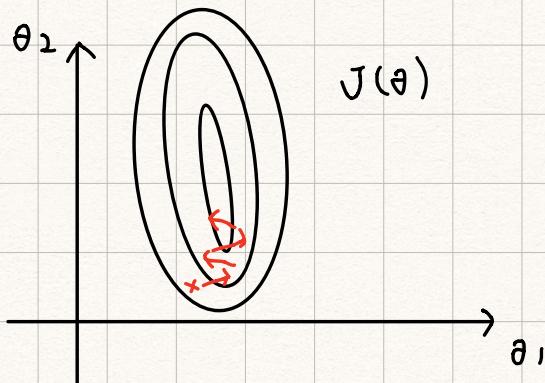
idea: Make sure features are on a similar scale. (如此，會更快收斂)

$$x_1 = \text{size (0~2000)}$$

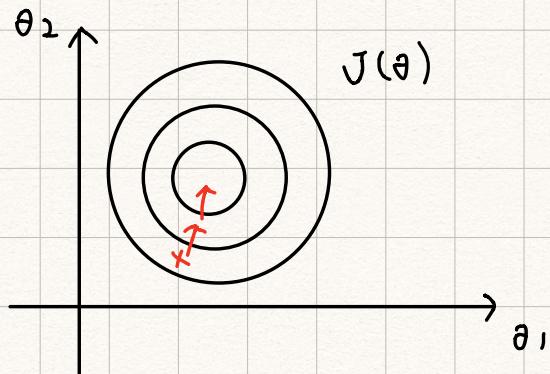
$$x_2 = \text{number of bedrooms (1~5)}$$

$$x_1 = \text{size / 2000}$$

$$x_2 = \text{number of bedrooms / 5}$$



收敛慢



收敛快

Get every feature into approximately a  $-1 \leq x_i \leq 1$  range.

Andrew. Ng suggestion:

$$\begin{array}{ccc} -3 & \text{to} & 3 \\ -\frac{1}{3} & \text{to} & \frac{1}{3} \end{array}$$

### Mean normalization

Replace  $x_i$  with  $\frac{x_i - \bar{x}_i}{s_i}$  to make features have approximately zero mean

$$\text{formula: } \frac{x_i - \bar{x}_i}{s_i}$$

$x_i$ : 原特徵的值

$\bar{x}_i$ : average value of  $x$

$s_i$ :  $\text{Max} - \text{Min}$

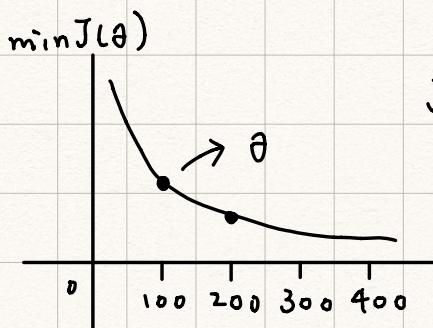
(or 標準差)

特徵縮放不做的太精細至每個值都一樣，這只是為了加快收斂速度而已

### Learning Rate

Making sure gradient descent is working

correctly.



$J(\theta)$  should decrease after every iteration

(1)

plotting the cost function !

(2)

automatic convergence test:

Declare convergence if  $J(\theta)$  decreases by less than  $10^{-3}$  in one iteration.

- For sufficiently small  $\alpha$ ,  $J(\theta)$  should decrease on every iteration
- But if  $\alpha$  is too small, gradient descent can be slow to converge.

$\alpha$  is too small: slow convergence

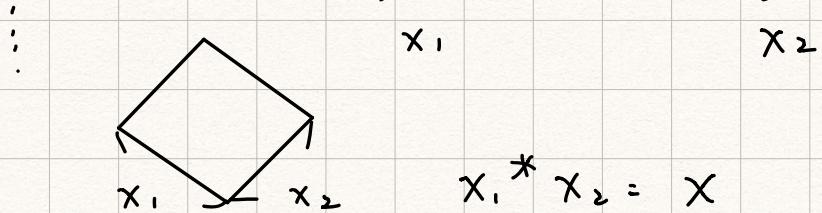
$\alpha$  is too large:  $J(\theta)$  may not decrease on every iteration; may not converge

$\alpha$ 的選擇	0.001	0.003
0.01	0.03	
0.1	0.3	
1	3	....

## Features and Polynomial Regression

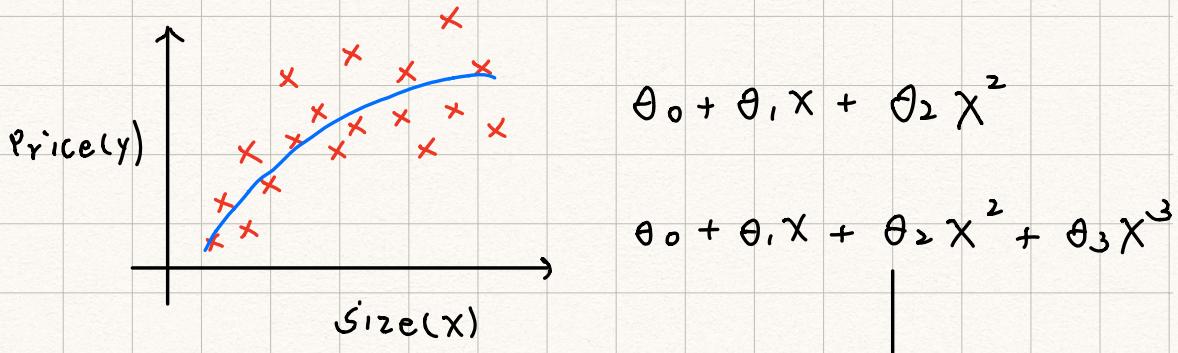
選擇適當的 features，創造新的 features

$$h_{\theta}(x) = \theta_0 + \theta_1 \times \text{frontage} + \theta_2 \times \text{depth}$$



$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

## 多項式迴歸 (Polynomial Regression)



If choose this show way



feature scaling is more important

size: 1 ~ 1000, size<sup>2</sup>: 1 - 1000000, size<sup>3</sup>: 1 - 10<sup>9</sup>

another reasonable hypothesis:

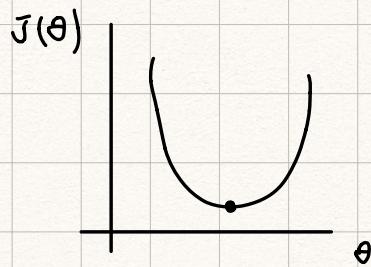
$$h_{\theta}(x) = \theta_0 + \theta_1(size) + \theta_2(\sqrt{size})$$

What features is more useful? There is some algorithm can help people choose.

## Normal Equation

Method to solve for  $\theta$  analytically.

一步就可以得到最像解



$$\frac{\partial J(\theta)}{\partial \theta} = 0, \quad \theta = \theta^*, \quad J(\theta^*) \min$$

$x_0$	$x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home $x_4$
1	2104	5	1	45
1	1416	3	2	40
1	1534	3	2	30
1	852	2	1	36

→ normal equation 的做法

$$X = \begin{pmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{pmatrix} \quad Y = \begin{pmatrix} 460 \\ 232 \\ 315 \\ 178 \end{pmatrix}$$

$m \times (n+1)$      $m \times 1$

$$\theta = (X^T X)^{-1} X^T y$$

$m$  examples  $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$ ;  $n$  features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

design matrix

$$X = \begin{bmatrix} \quad (x^{(1)})^\top \quad \\ \quad (x^{(2)})^\top \quad \\ \vdots \\ \vdots \\ \quad (x^{(m)})^\top \quad \end{bmatrix}$$

example:

$$\text{if } x^{(i)} = \begin{bmatrix} 1 \\ x_i^{(i)} \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_2^{(1)} \\ 1 & x_3^{(1)} \\ 1 & x_4^{(1)} \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \end{bmatrix}$$

$$\theta = (X^\top X)^{-1} X^\top y$$

$\theta$  即為最值  
使得  $J(\theta)$  為最小

(Normal equation 不須用 feature scaling)

Pros and cons

Gradient Descent

Normal Equation

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>• Need to choose <math>\alpha</math></li> <li>• Need many iterations</li> <li>• Works well even when <math>n</math> is large</li> </ul> | <ul style="list-style-type: none"> <li>• No need to choose <math>\alpha</math></li> <li>• Don't need to iteration</li> <li>• Need to compute <math>(X^T X)^{-1}</math></li> <li>• Slow if <math>n</math> is very large</li> </ul> |
|--|---|

$n < 10000$ , use normal equation

### Non-invertibility

$$\theta = (X^T X)^{-1} X^T y$$

Octave Language : `pinv(X'*X)*X'*y`

`pinv` : pseudo-inverse

`inv` : inverse

What if  $X^T X$  is non-invertible?

- Redundant features (linearly dependent)  
like feet<sup>2</sup> or m<sup>2</sup>, very closely related
- Too many features ( $m \leq n$ )

- Delete some features, or use regularization