Multiple Features

適合多變量．多特徵的線性迴歸表示法

Mutiple 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) → 1416	3	2	40
⋮	⋮	⋮	⋮

n = number of features

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

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

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

hypothesis :

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

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

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \in \mathbb{R}^{n+1} \qquad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}_{n\times 1} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \theta^T x$$

Multiple linear regression

Gradient Descent in Practice

for Multiple variables

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

parameters : θ = $n+1$ dimension vector

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

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

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

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

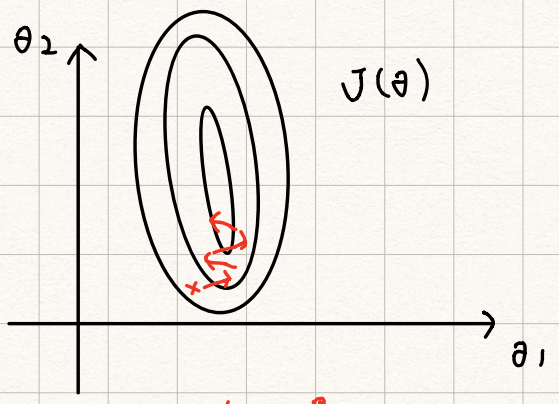
simultaneously update θ_j for $j = 0, \dots, n$

$x_0^{(i)} = 1$

feature scaling (特徵縮放)

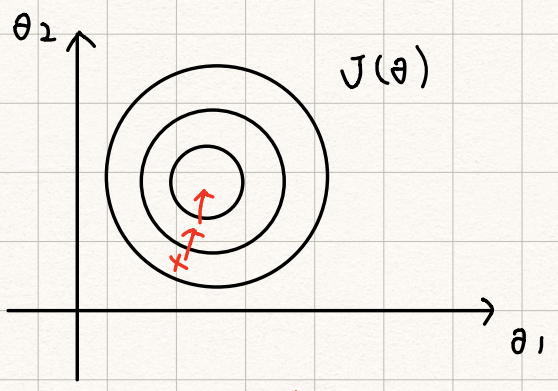
idea: Make sure features are on a similar scale. (如此，會更快收斂，計算速度↑)

x_1 = size (0~2000)
x_2 = number of bedrooms (1~5)



收斂慢

x_1 = size / 2000
x_2 = number of bedrooms / 5



收斂快

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

Andrew. Ng suggestion:

-3	to	3
$-\frac{1}{3}$	to	$\frac{1}{3}$

Mean normalization

Replace x_i with $x_i - M_i$ to make features have approximately zero mean

formula: $\frac{x_i - M_i}{S_i}$

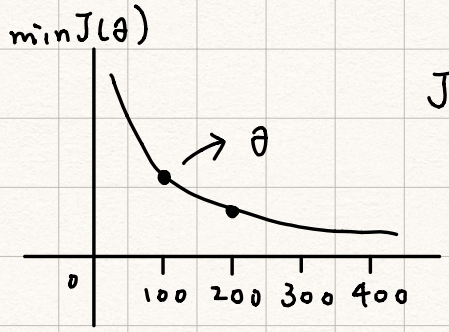
x_i: 原特徵的值
M_i: average value of x
S_i: Max - Min
(or 標準差)

特徵縮放不必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 α, $J(\theta)$ should decrease on every iteration
- But if α is too small, gradient descent can be slow to converge.

- α is too small: slow convergence
- α is too large: $J(\theta)$ may not decrease on every iteration; may not convergence

α 的選擇 0.001 0.003

0.01 0.03

0.1 0.3

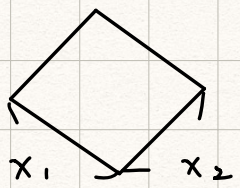
1 3 .....

Features and Polynomial Regression

選擇適當的 features or 創造新的 features

$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$

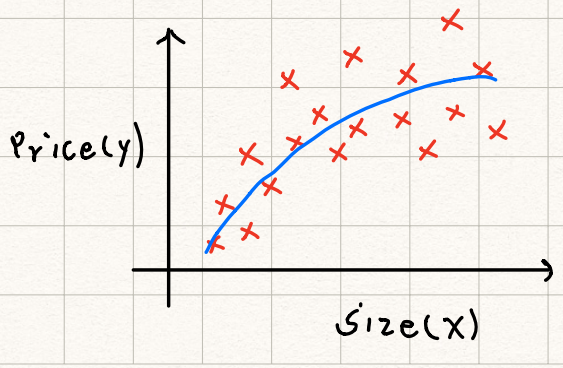
⋮



$x_1 * x_2 = x$

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

多項式迴歸 (Polynomial Regression)



$\theta_0 + \theta_1 x + \theta_2 x^2$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

- if choose this show way ↓

  feanture scaling is more important !

  size: 1 ~ 1000 , $size^2$: 1 - 1000000, $size^3$: 1 - 10^9

- another reasonable hypethsis :

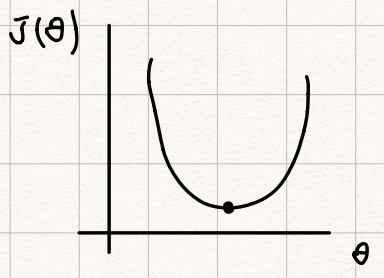
  $h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$

- what features is more useful ? There are some algorithm can help people choose .

Normal Equation

Method to solve for θ analytically.

一步就可以得到最優解



$\frac{dJ(\theta)}{d\theta} = 0$, $\theta = \theta'$ 時, $J(\theta)$ min

x_0	size 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

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

$m \times (n+1)$ $\qquad$ $m \times 1$

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

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

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \qquad X = \text{design matrix} = \begin{bmatrix} \text{——} (x^{(1)})^T \text{——} \\ \text{——} (x^{(2)})^T \text{——} \\ \vdots \\ \text{——} (x^{(m)})^T \text{——} \end{bmatrix}$$

example:

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

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

θ 即為最佳值
使得 J(θ) 為最小

(Normal equation 不須用 feature scaling)

Pros and cons

Gradient Descent	Normal Equation

• Need to choose α • Need many iterations • Works well even when • n is large	• No need to choose α • Don't need to iteration • Need to compute $(X^TX)^{-1}$ • Slow if n is very large

$n < 10000$, use normal equation

Noninvertibility

$$\theta = (X^TX)^{-1}X^Ty$$

- Octave Language : `pinv(x'*x)*x'*y`
  - pinv : pseudo-inverse
  - inv : inverse

- What if X^TX is non-invertible?
  - Redundant features (linearly dependent)
    like $feet^2$ or m^2, very closely related
  - Too many features ($m \leq n$)

- Delete some features, or use regularization