機器學習

Lecture 2　Regression

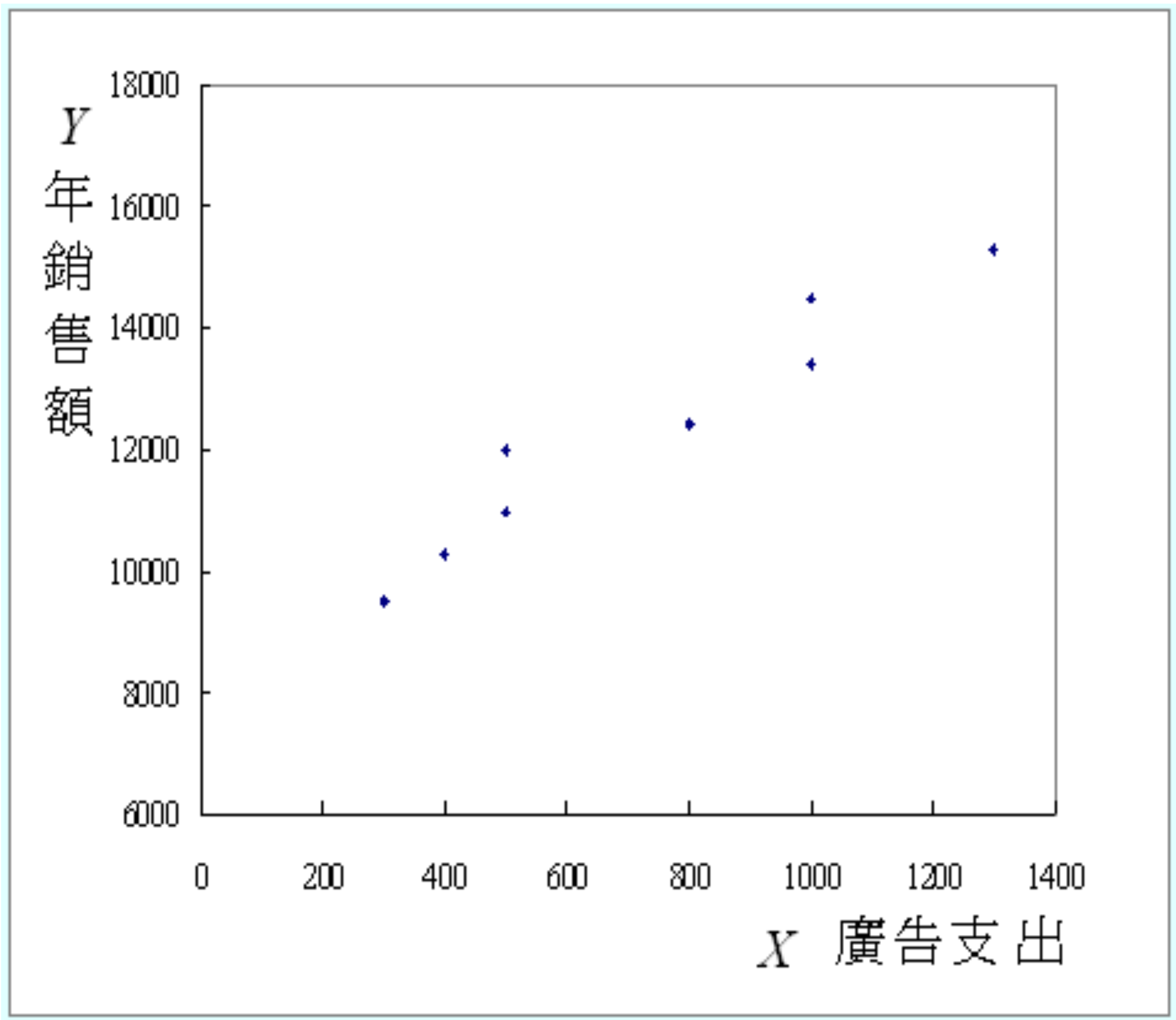
相關關係

- 某汽車公司 8 個分公司汽車銷售額與廣告支出數額的資料。

分公司名稱	廣告支出X	年銷售額Y
A	300	9,500
B	400	10,300
C	500	11,000
D	500	12,000
E	800	12,400
F	1,000	13,400
G	1,000	14,500
H	1,300	15,300

相關關係

- 某汽車公司 8 個分公司汽車銷售額與廣告支出數額的資料。



相關係數

$$r_{xy} = \frac{S_{xy}}{S_x S_y} = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2}\sqrt{\sum (x-\bar{x})^2}} \in [-1,1]$$

其中，$$S_{xy} = \frac{\sum (x-\bar{x})(y-\bar{y})}{n-1}$$

$$S_x = \sqrt{\frac{\sum (x-\bar{x})^2}{n-1}}$$

$$S_y = \sqrt{\frac{\sum (y-\bar{y})^2}{n-1}}$$

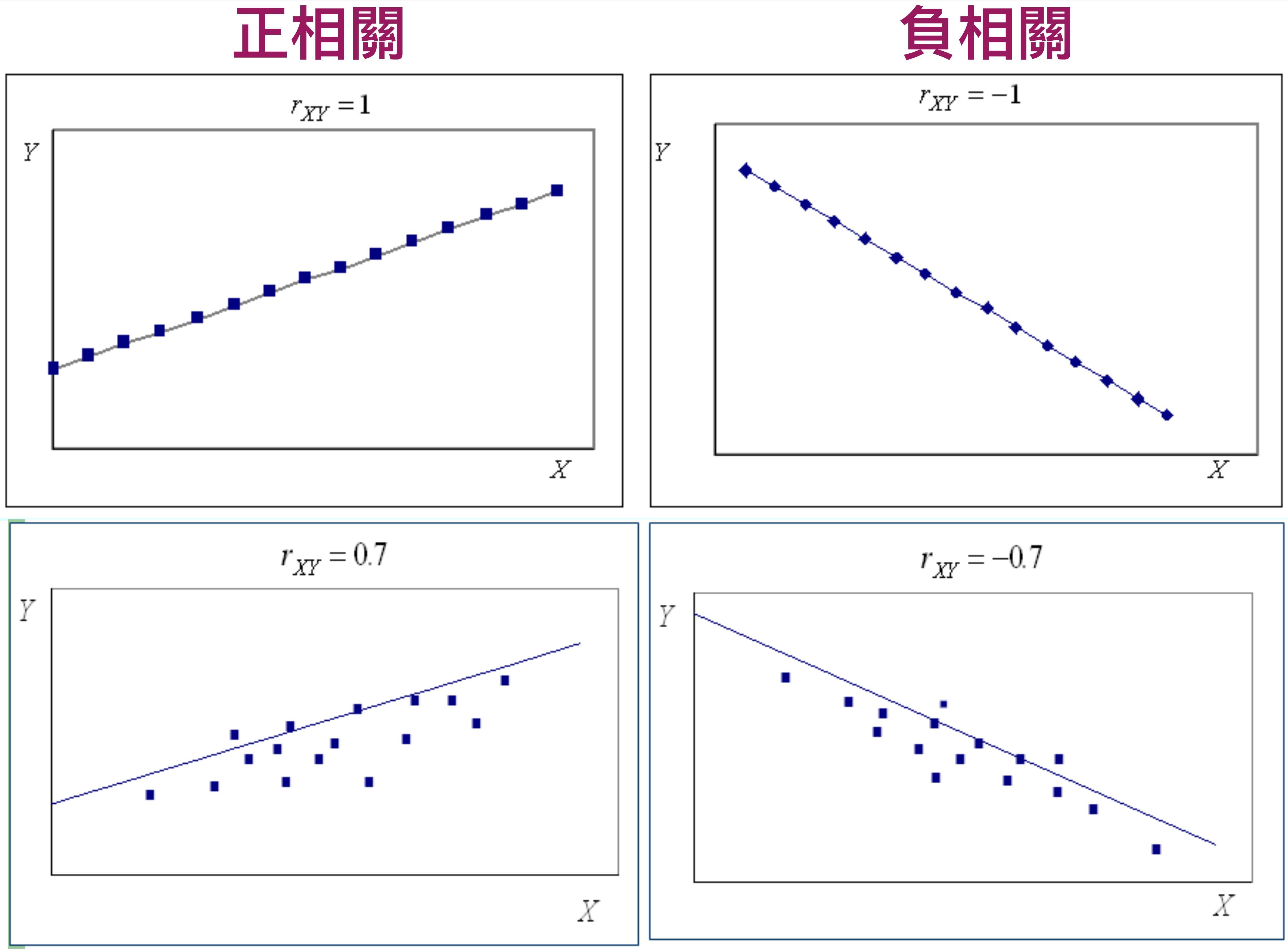
相關係數

- 某汽車公司 8 個分公司汽車銷售額與廣告支出數額的資料。

分公司名稱	廣告支出X	年銷售額Y
A	300	9,500
B	400	10,300
C	500	11,000
D	500	12,000
E	800	12,400
F	1,000	13,400
G	1,000	14,500
H	1,300	15,300

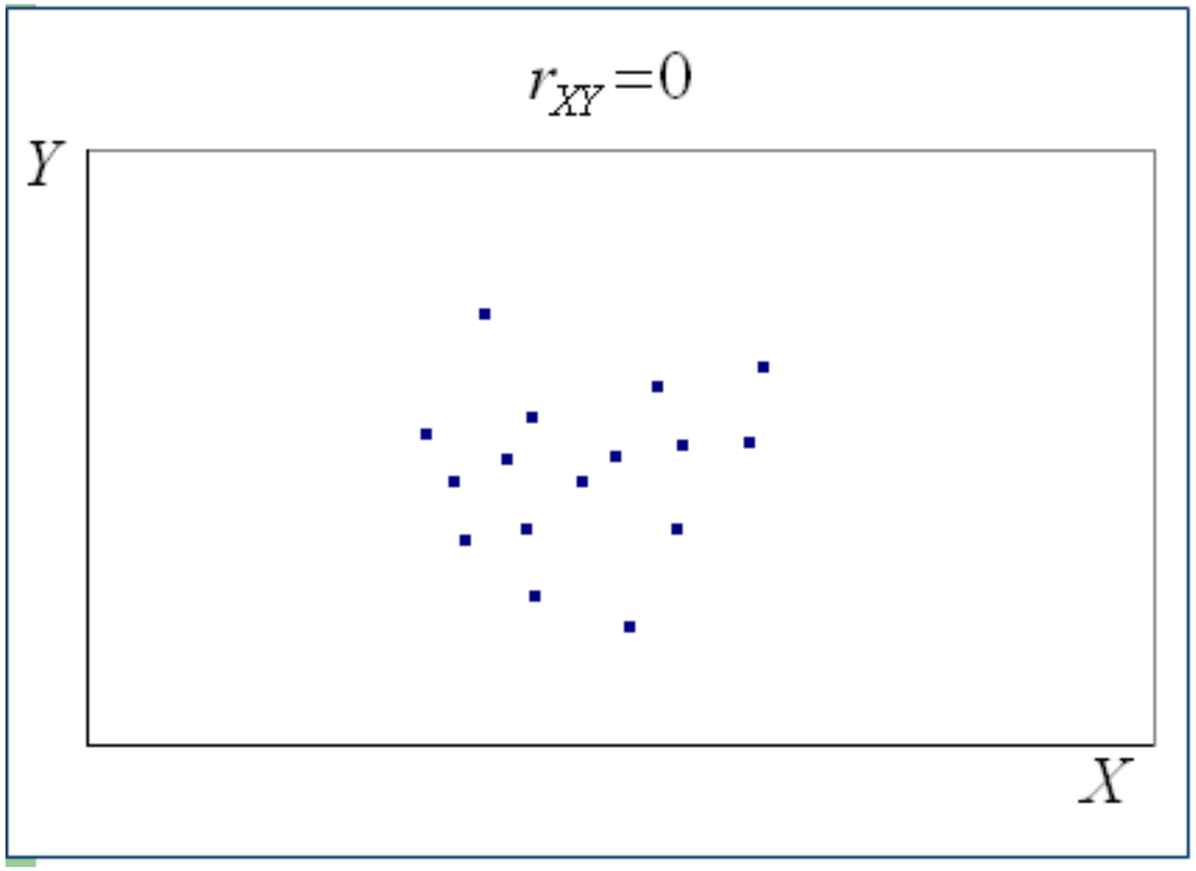
$$r_{XY} = \frac{\sum (X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum (X-\bar{X})^2}\sqrt{\sum (Y-\bar{Y})^2}} = \frac{4{,}840{,}000}{\sqrt{875{,}000}\sqrt{28{,}680{,}000}} = 0.966$$

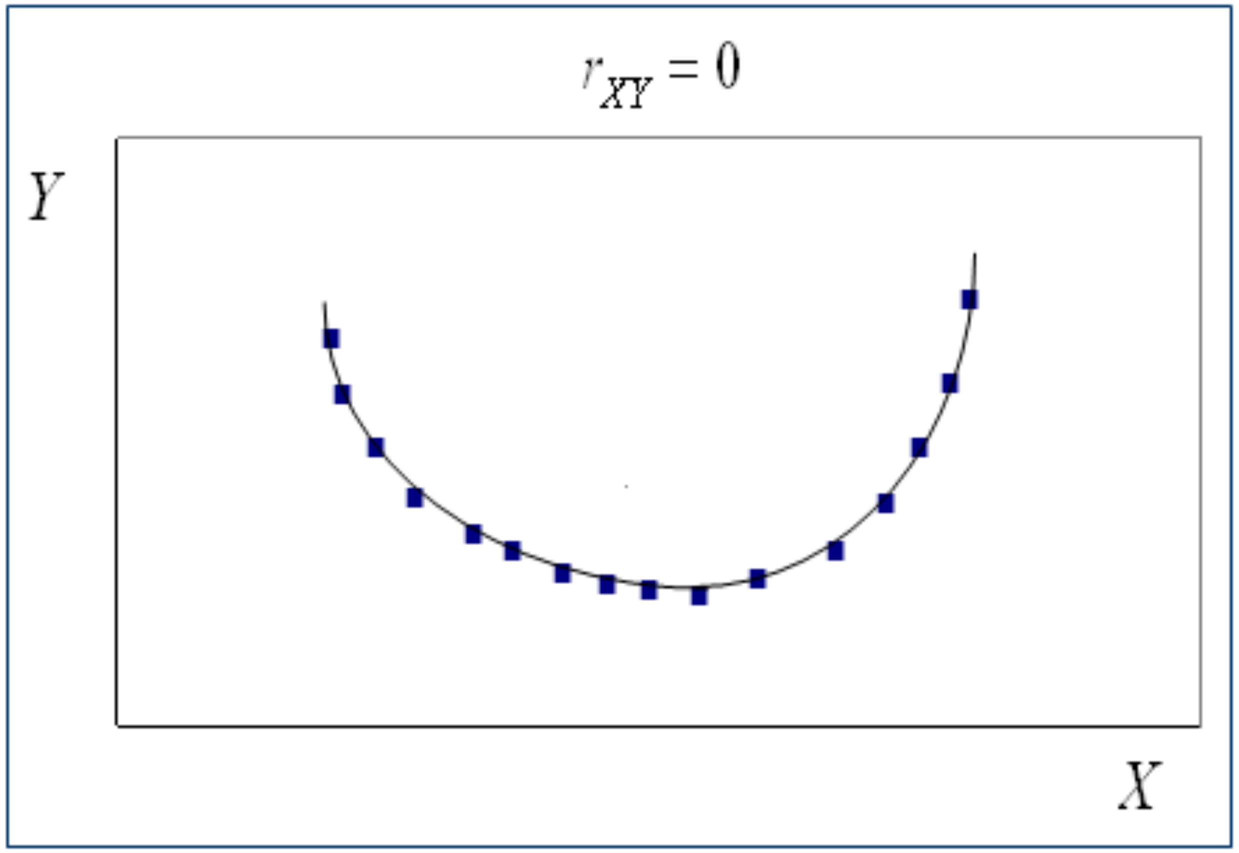
相關係數



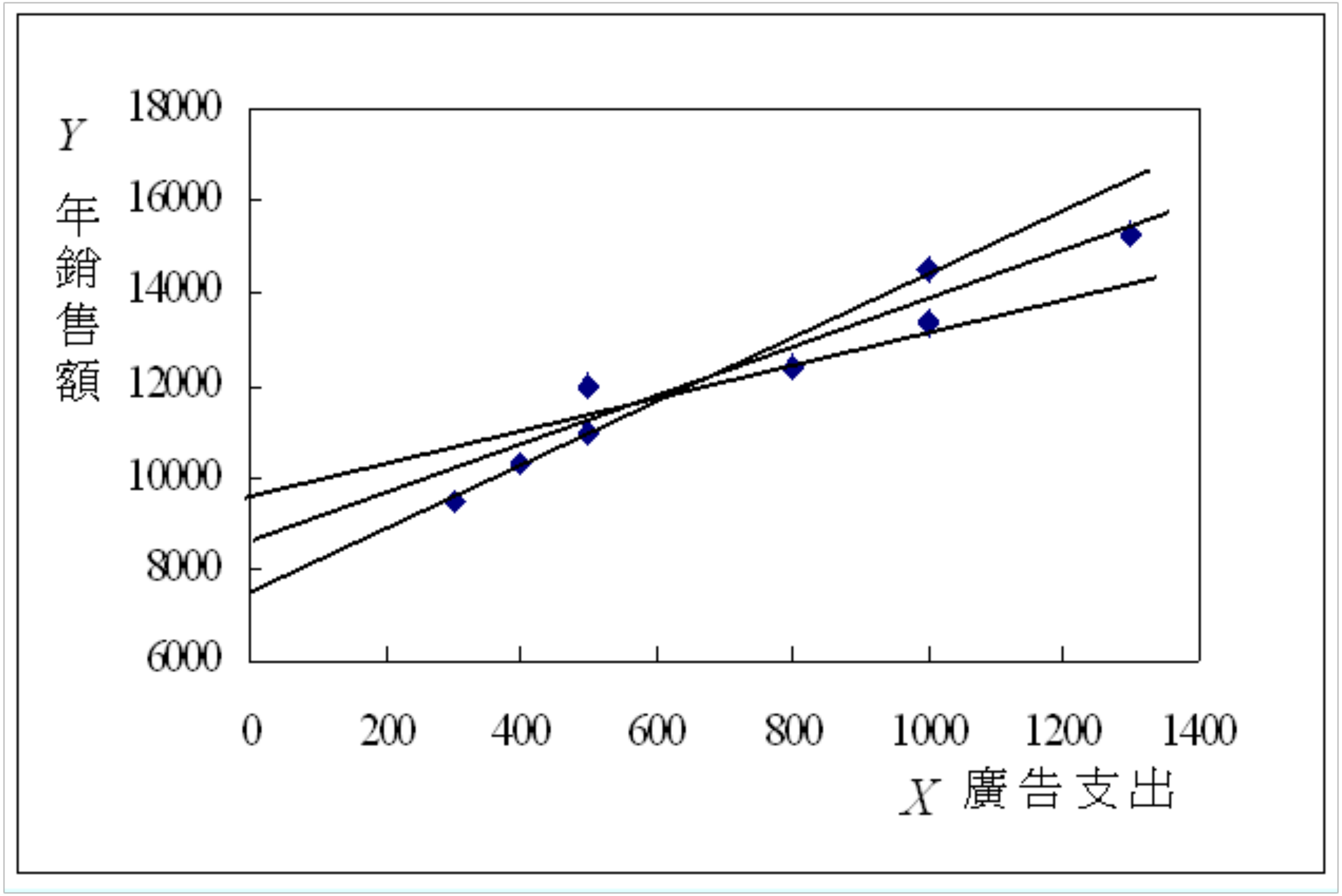
相關係數

不存在線性相關



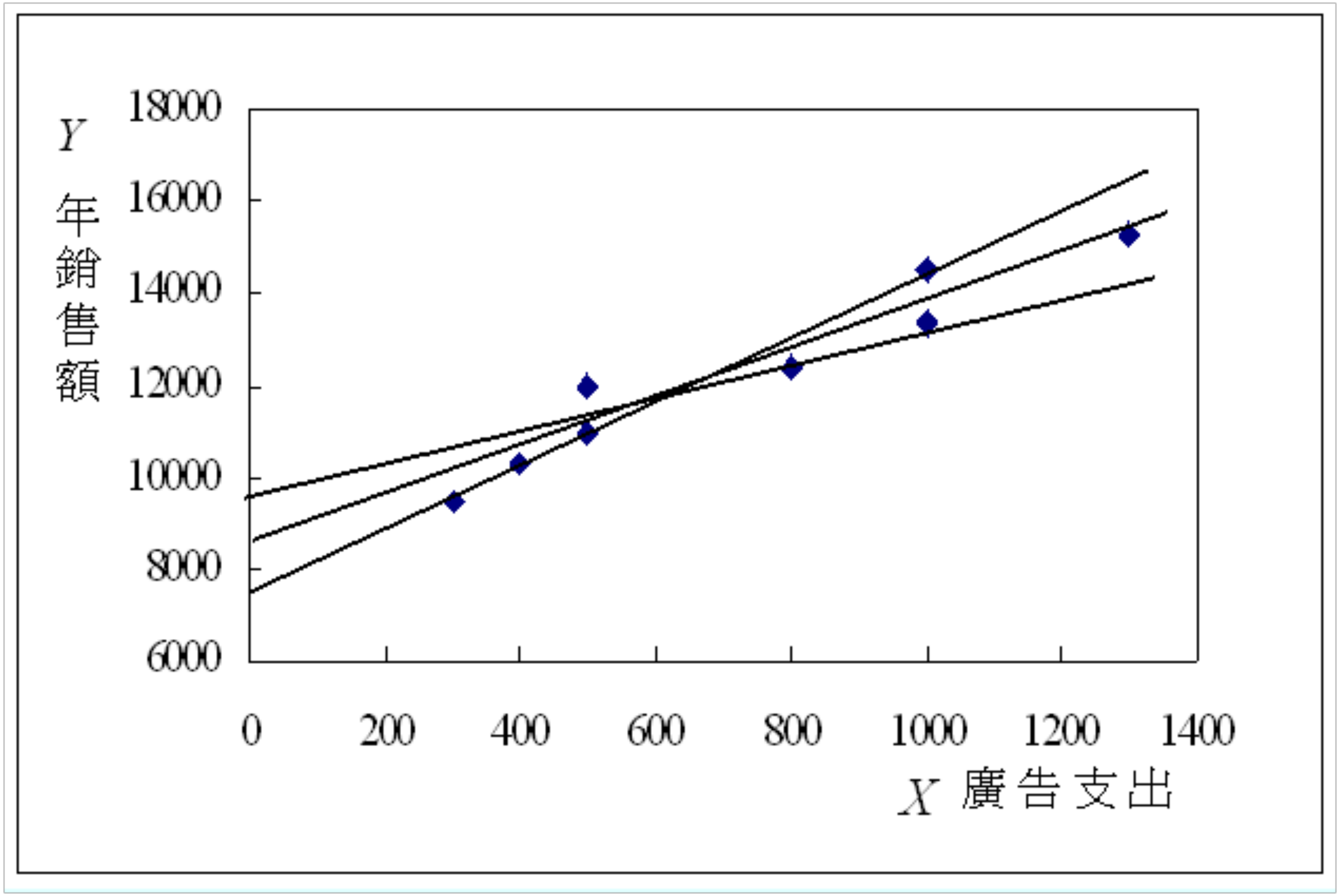


Regression



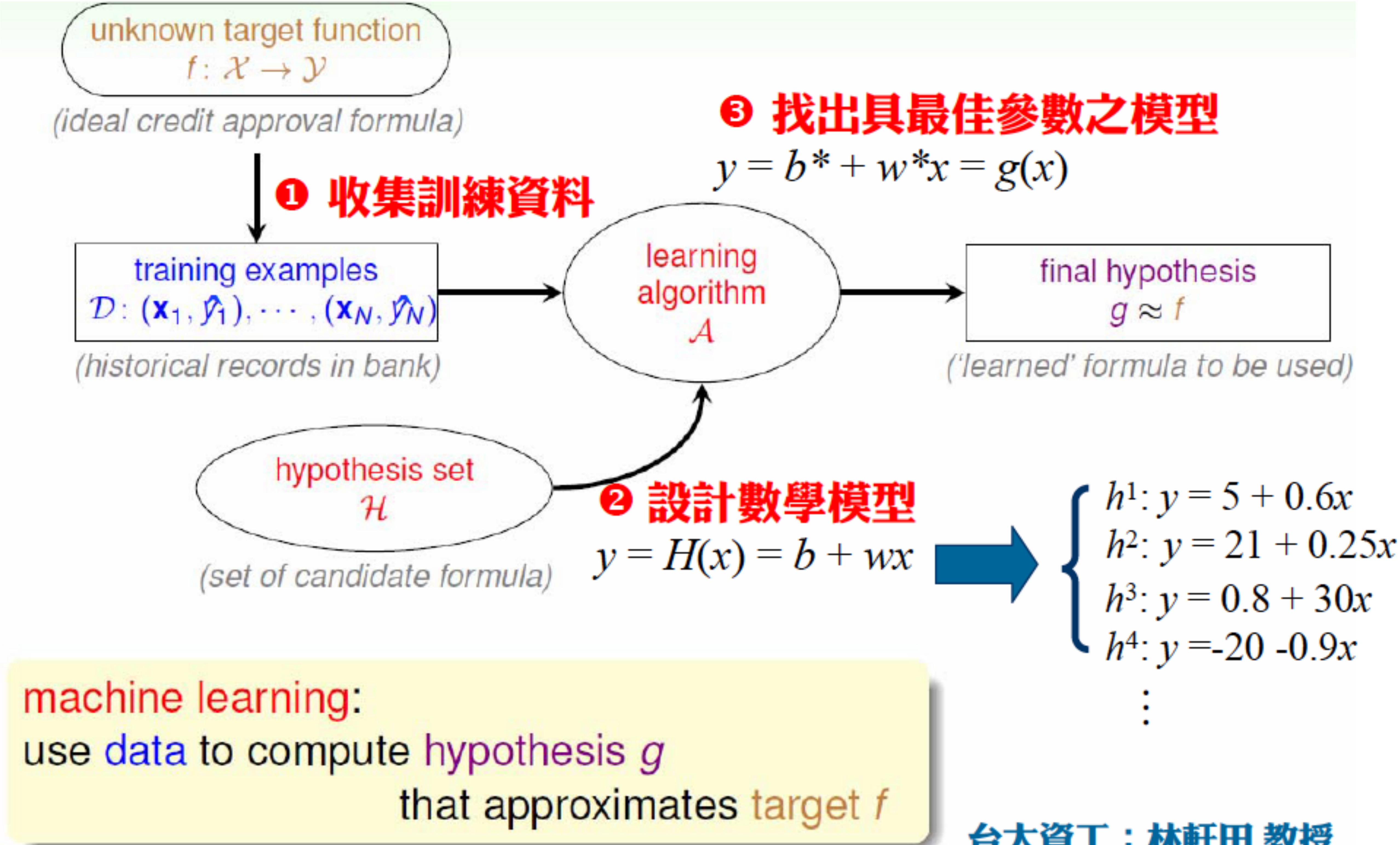
線性模型 $y = ax + b$

Regression



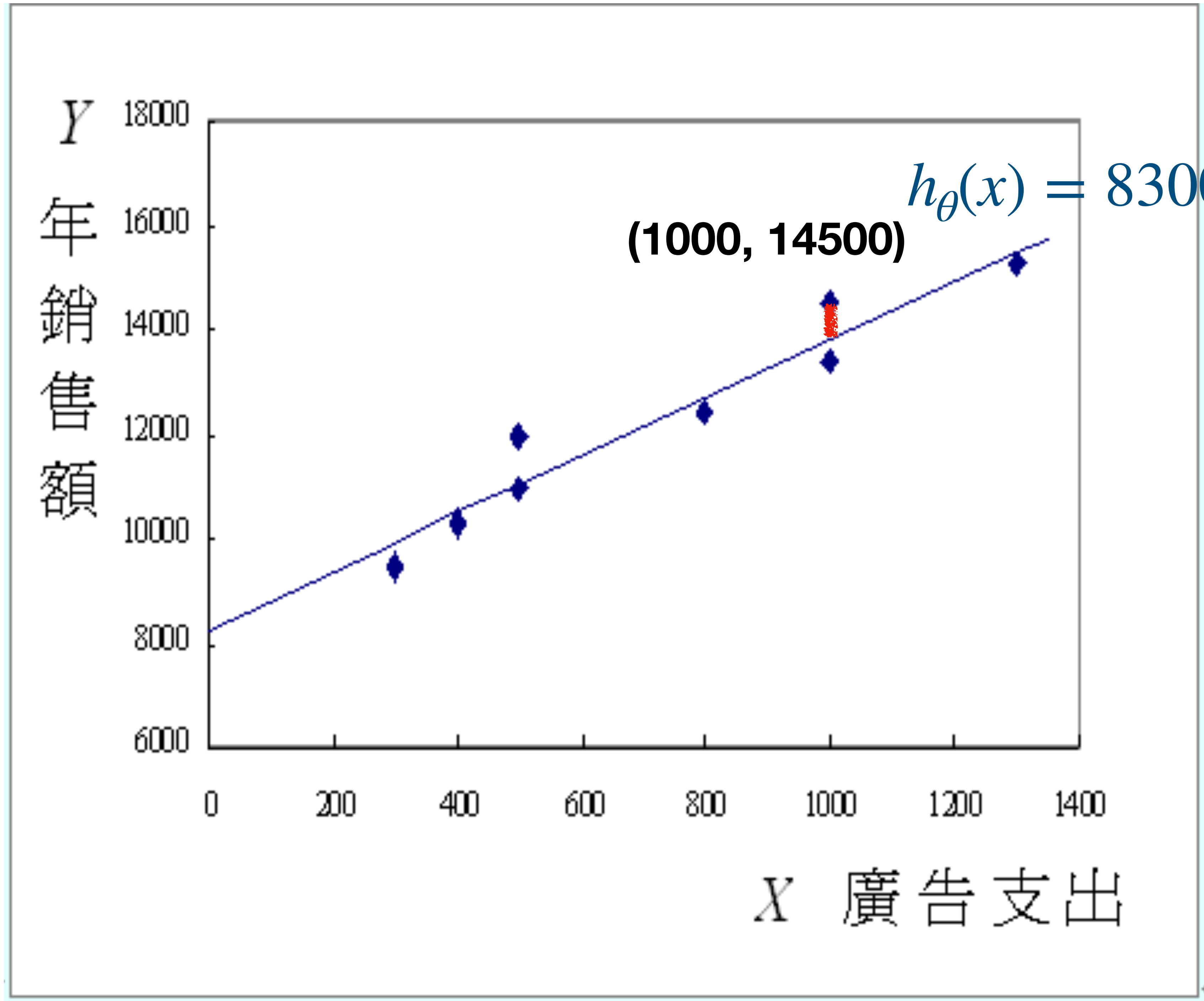
線性模型 $h_\theta(x) = \theta_0 + \theta_1 x$

Regression



台大資工：林軒田 教授

Regression



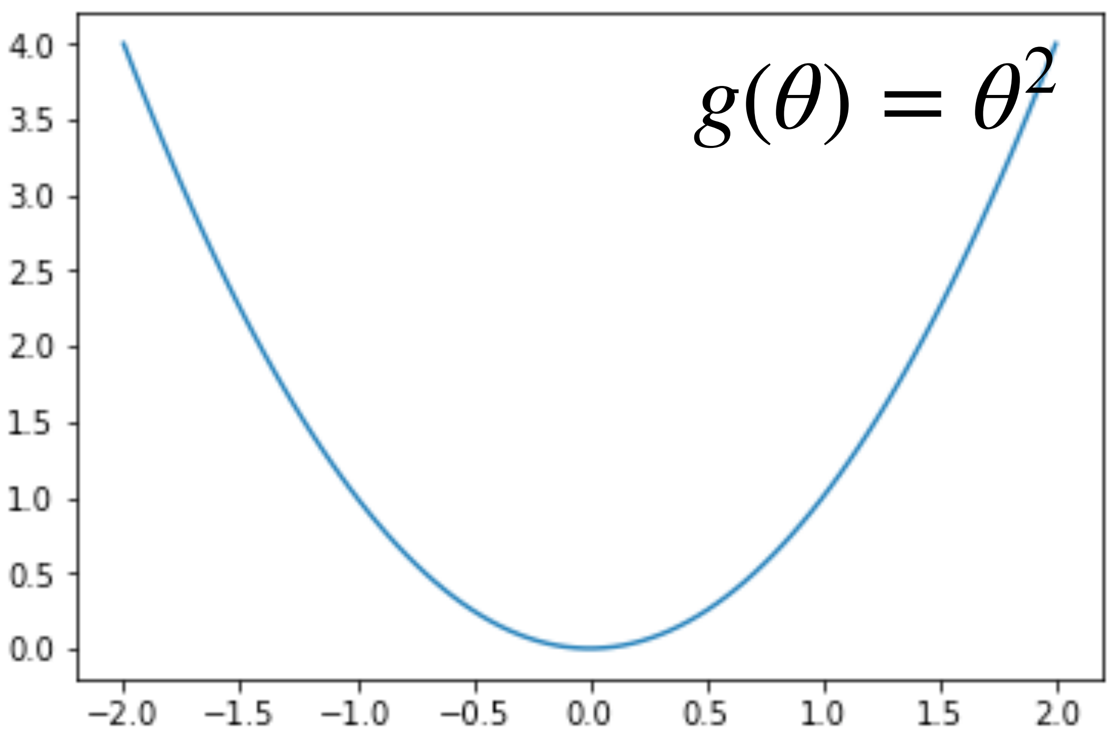
$h_\theta(x) = 8300 + 5.5x$

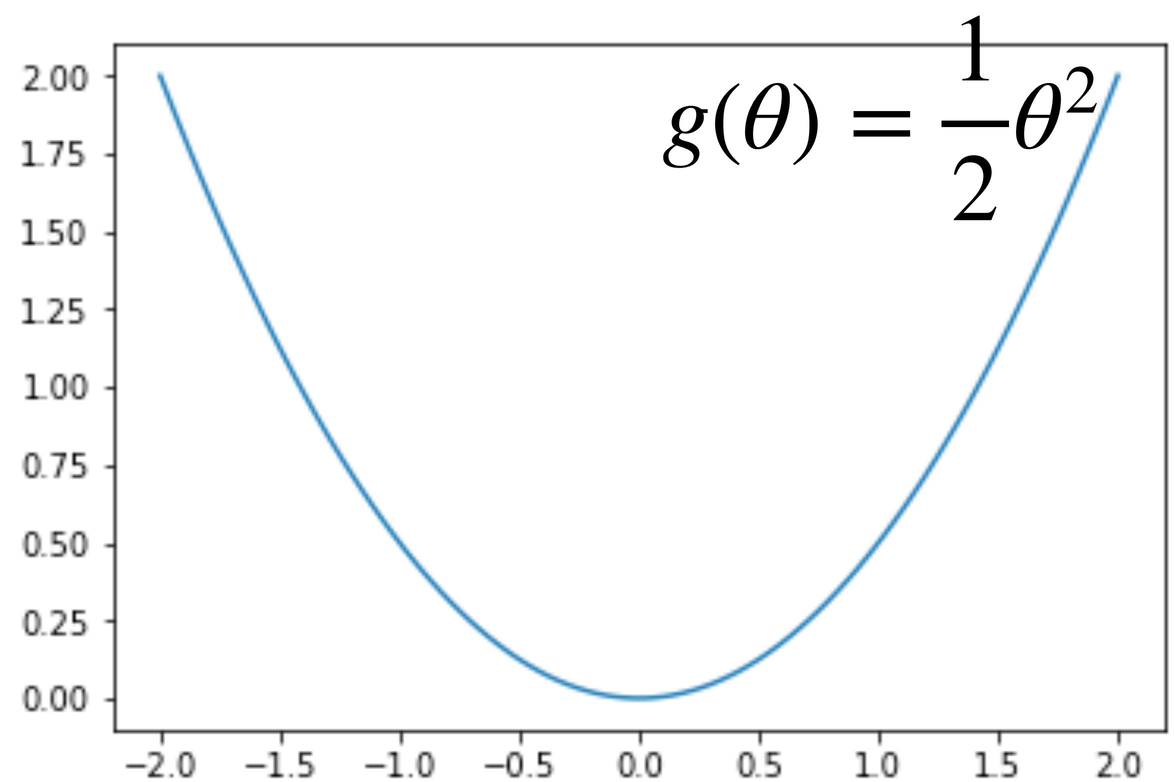
最小平方法 (Least square Method)

- Goal: minimize the cost function

$$min_{\theta_0,\theta_1} \boxed{\frac{1}{2}} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

1/2只是為了微分推導方便，對最小值沒有影響





最小平方法 (Least square Method)

- Cost function

$$E(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- 找一組 θ_0, θ_1 使 $E(\theta_0, \theta_1)$ 最小

$$\frac{\partial E(\theta_0, \theta_1)}{\partial \theta_0} = \sum\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) \equiv 0$$

$$\frac{\partial E(\theta_0, \theta_1)}{\partial \theta_1} = \sum\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) \cdot x^{(i)} \equiv 0$$

最小平方法 (Least square Method)

- 解方程組得

$$\theta_1 = \frac{n\sum x^{(i)}y^{(i)} - \sum x^{(i)}\sum y^{(i)}}{n\sum (x^{(i)})^2 - \left(\sum x^{(i)}\right)^2} = \frac{\sum \left(x^{(i)} - \bar{x}\right)\left(y^{(i)} - \bar{y}\right)}{\sum \left(x^{(i)} - \bar{x}\right)^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

$$\theta_1 = r_{xy} \cdot \frac{S_y}{S_x}$$

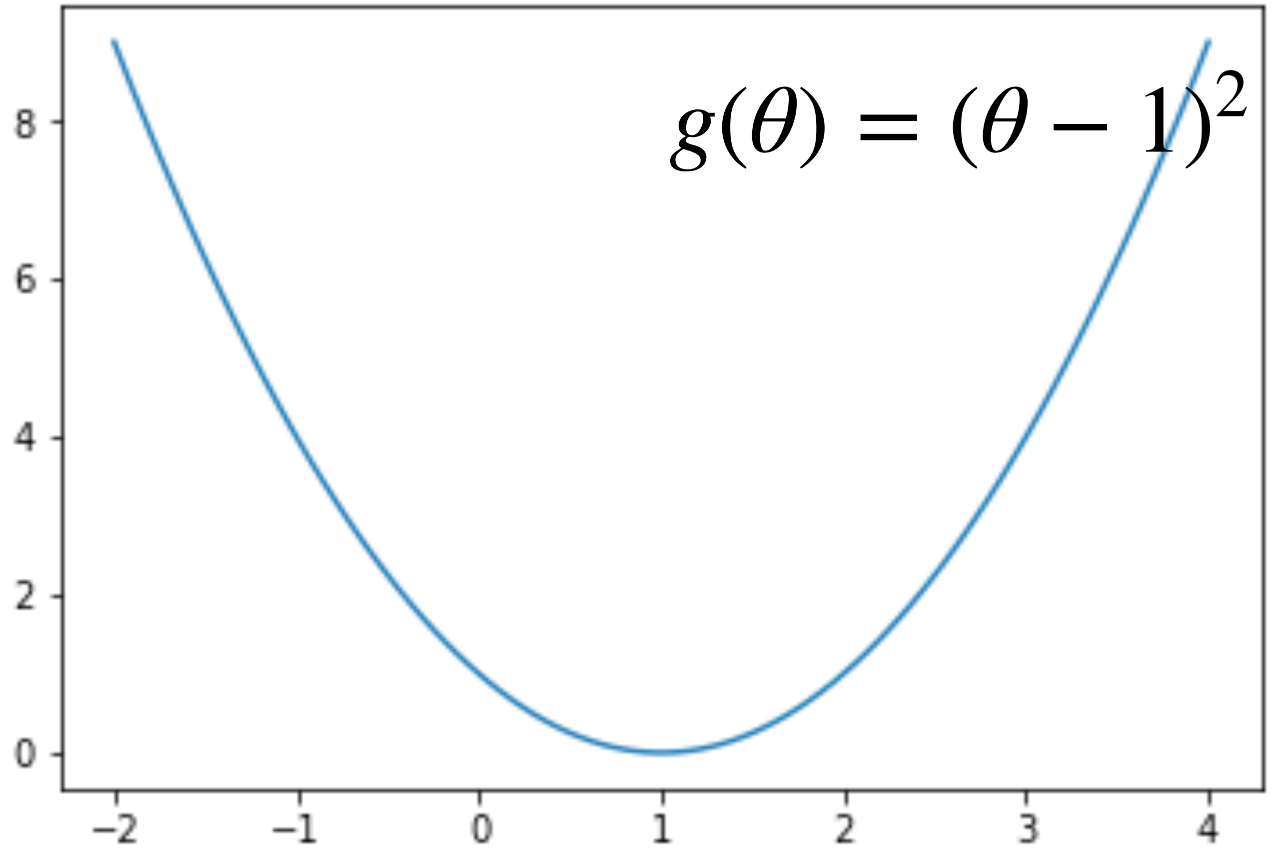
Regression

- Cost function

$$E(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

分公司名稱	廣告支出X	年銷售額Y
A	300	9,500
B	400	10,300
C	500	11,000
D	500	12,000
E	800	12,400
F	1,000	13,400
G	1,000	14,500
H	1,300	15,300

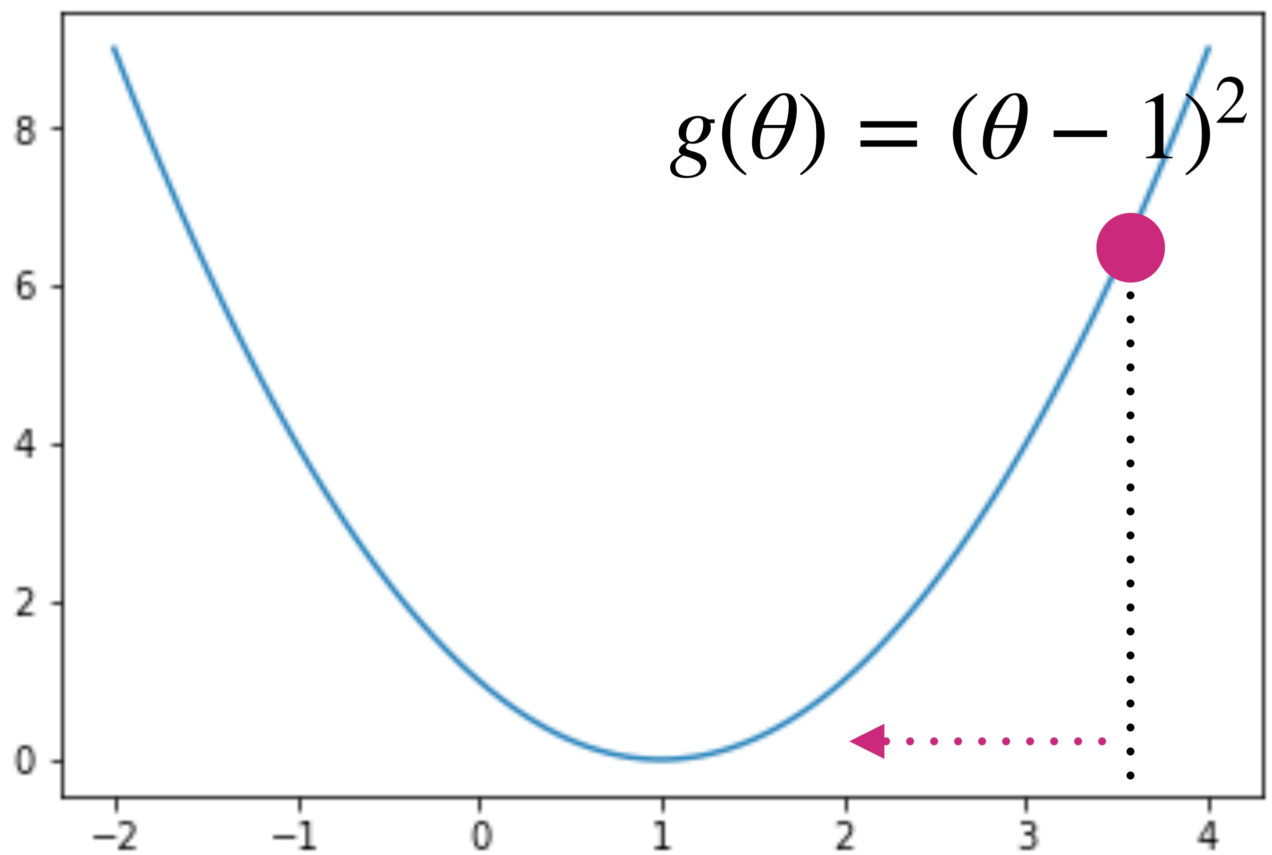
増減表



	$\theta < 1$	$\theta = 1$	$\theta > 1$
$g'(\theta)$	$-$	0	$+$
	$\searrow$	minimum	$\nearrow$

$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

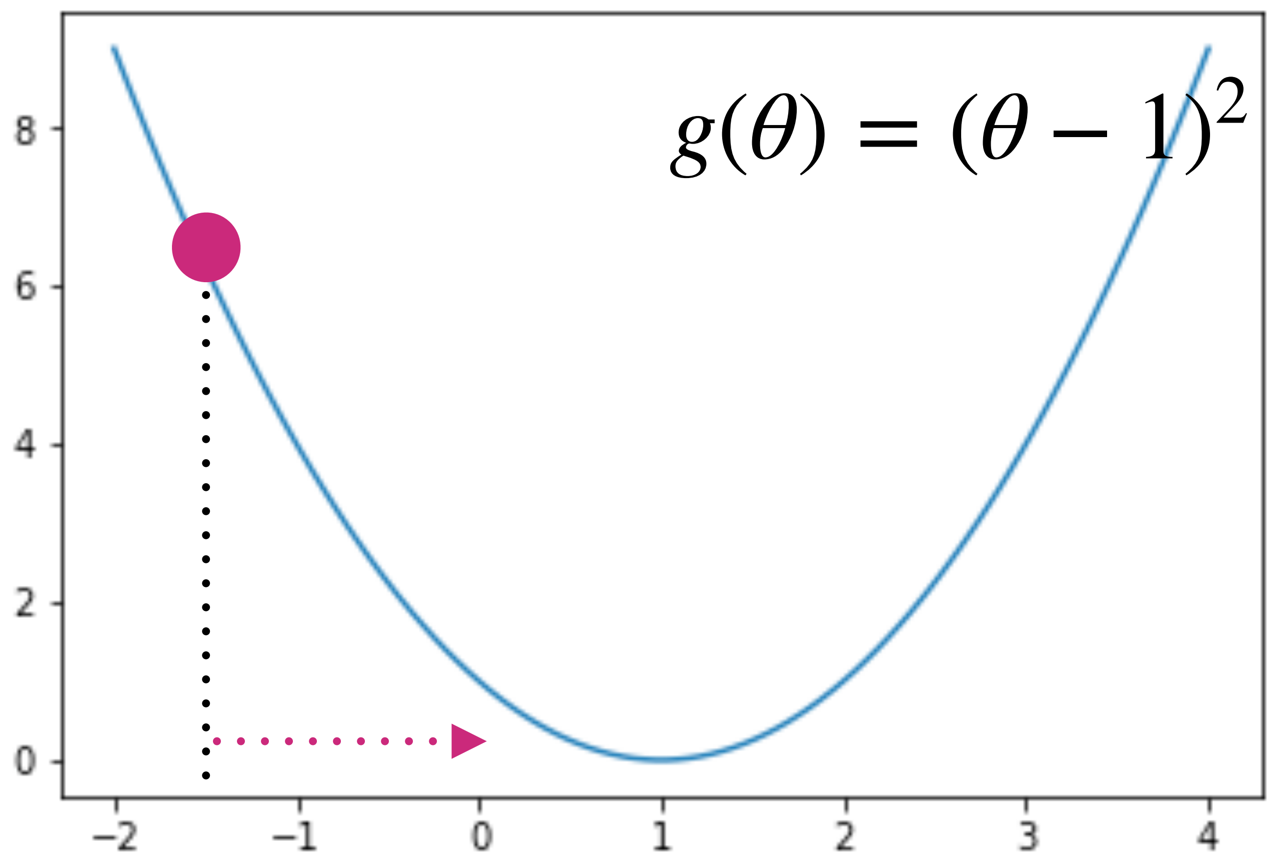
增減表



	$\theta < 1$	$\theta = 1$	$\theta > 1$
$g'(\theta)$	$-$	0	$+$
	$\searrow$	minimum	$\nearrow$

$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

増減表

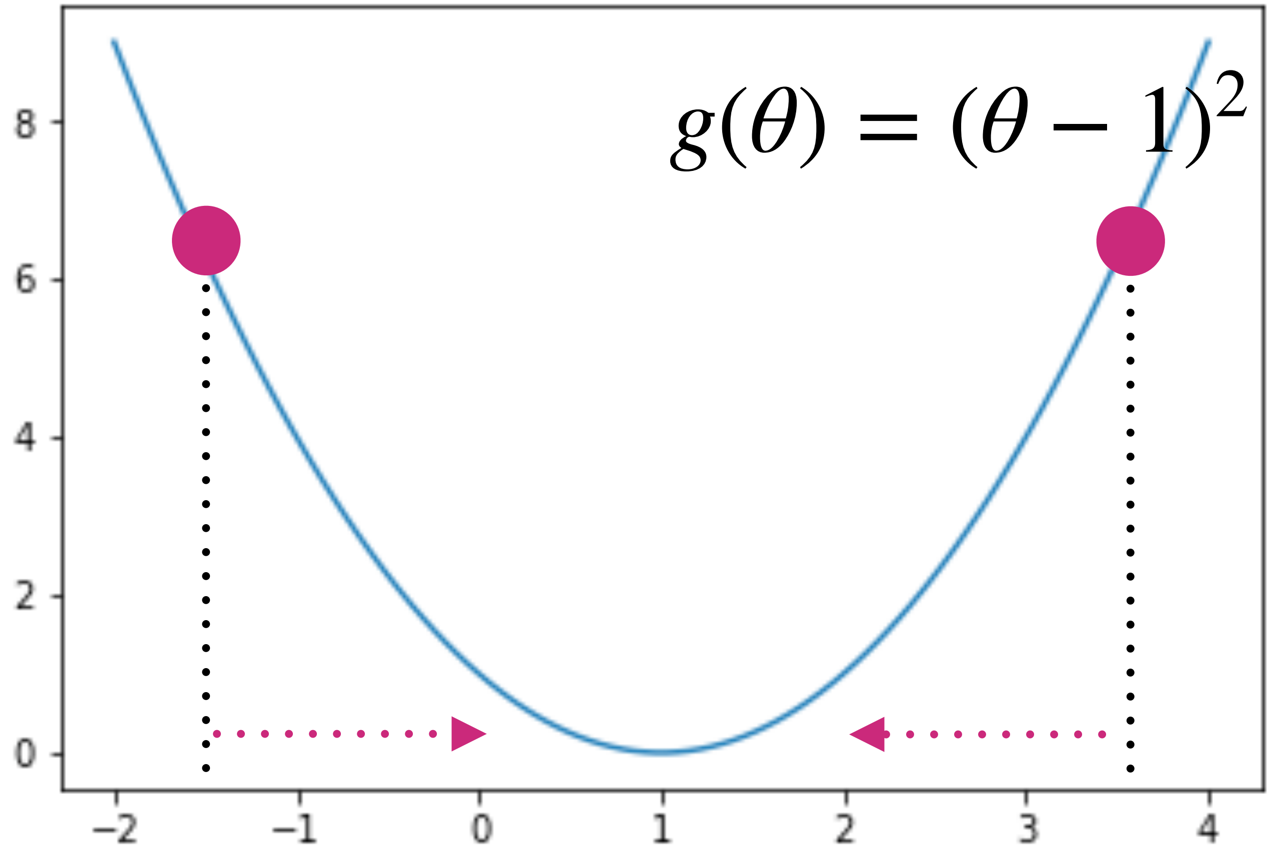


	$\theta < 1$	$\theta = 1$	$\theta > 1$
$g'(\theta)$	$-$	0	$+$
	↘	minimum	↗

$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

Gradient descent（梯度下降法)

往與導函數相反的方向移動，就會往最小值的方向移動



Gradient descent（梯度下降法)

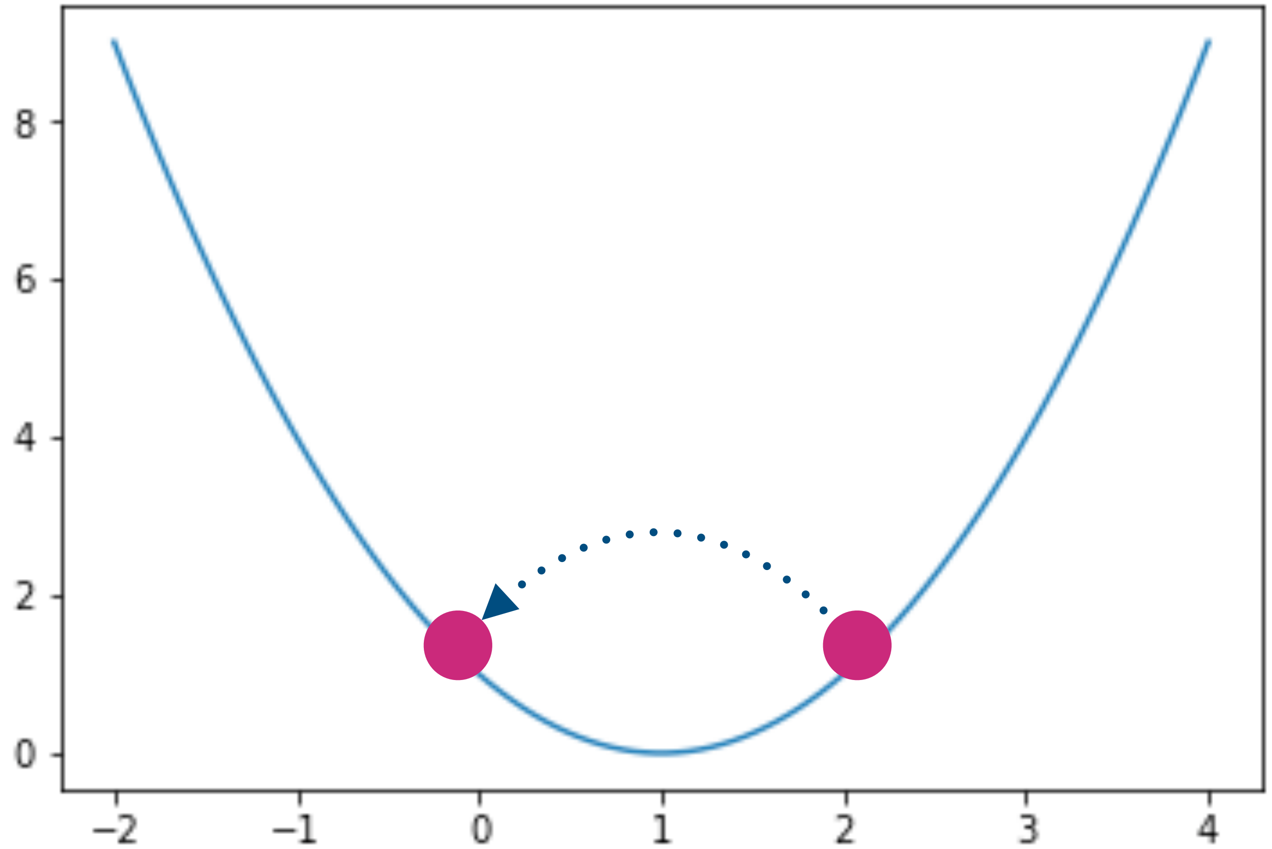
$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

η : learning rate

Gradient descent (梯度下降法)

Suppose $\eta = 1$

$$g(\theta) = (\theta - 1)^2$$



$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

Gradient descent (梯度下降法)

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

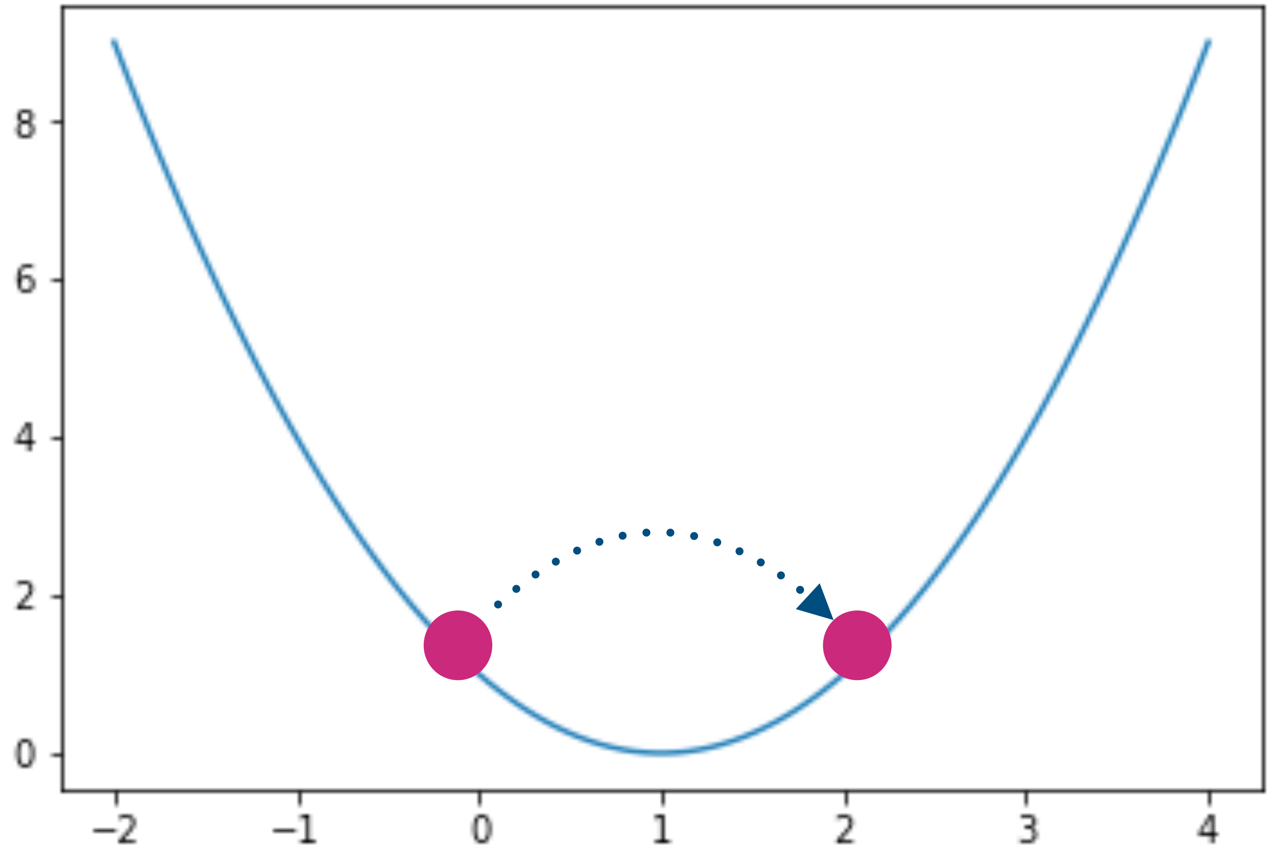
$$= \theta - \eta(2\theta - 2)$$

$$\theta := 2 - 1(2 \cdot 2 - 2) = 2 - 2 = 0$$

Gradient descent（梯度下降法)

Suppose $\eta = 1$

$$g(\theta) = (\theta - 1)^2$$



$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

Gradient descent（梯度下降法)

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

$$= \theta - \eta(2\theta - 2)$$

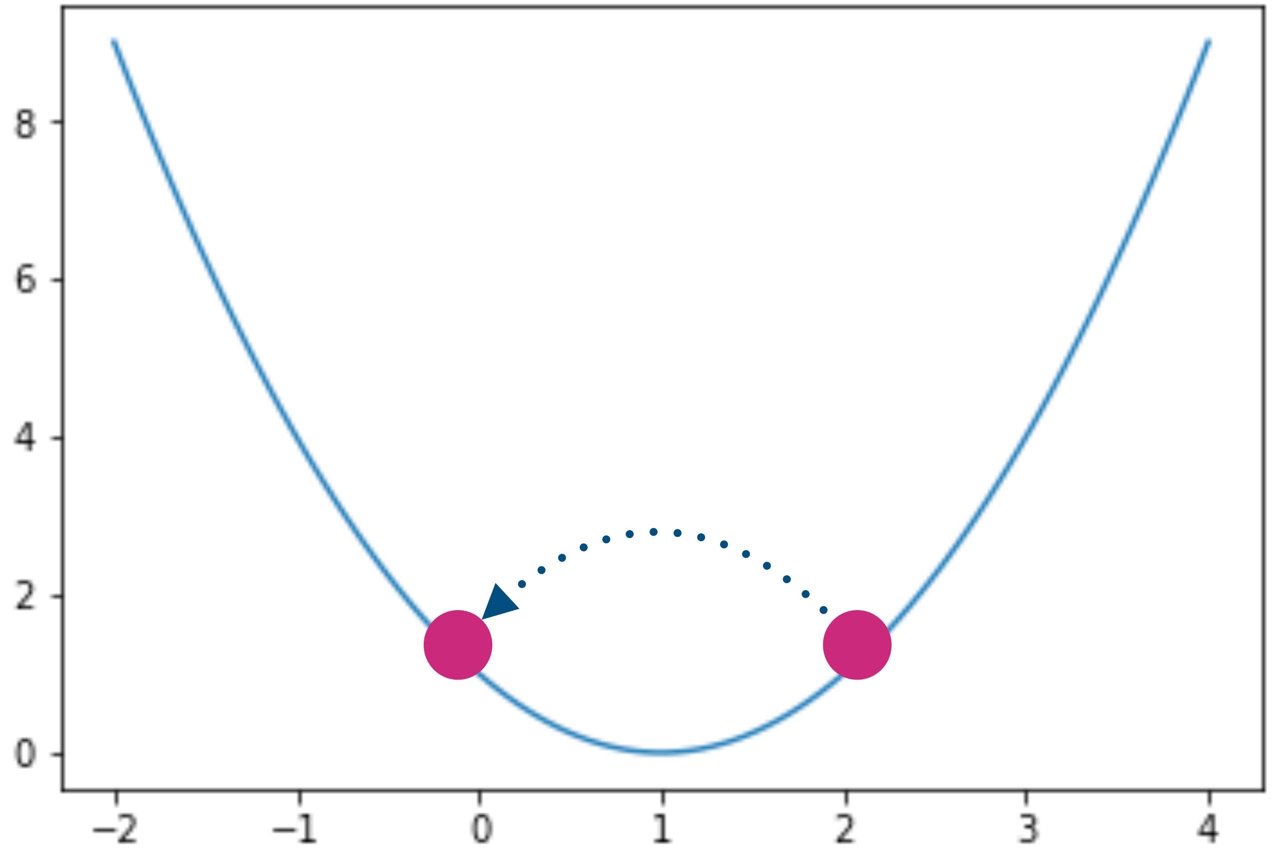
$$\theta := 0 - 1(2 \cdot 0 - 2) = 0 + 2 = 2$$

Gradient descent（梯度下降法）

Suppose $\eta = 1$

$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

$$g(\theta) = (\theta - 1)^2$$



Gradient descent（梯度下降法）

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

$$= \theta - \eta(2\theta - 2)$$

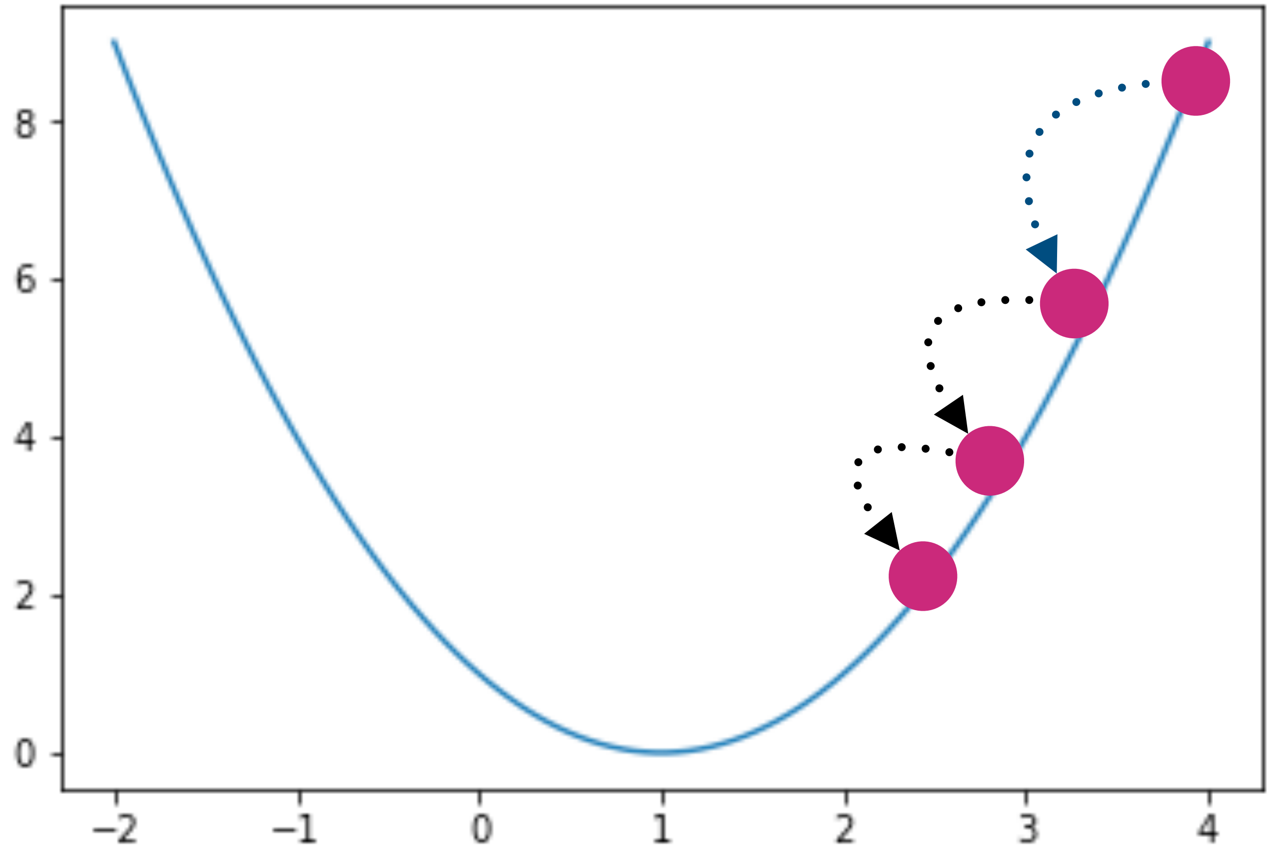
$$\theta := 2 - 1(2 \cdot 2 - 2) = 2 - 2 = 0$$

Gradient descent（梯度下降法)

Suppose $\eta = 0.1$

$$\frac{d}{d\theta}g(\theta) = 2\theta - 2$$

$g(\theta) = (\theta - 1)^2$



Gradient descent（梯度下降法)

$$\theta := \theta - \eta\frac{d}{d\theta}g(\theta)$$

$$= \theta - \eta(2\theta - 2)$$

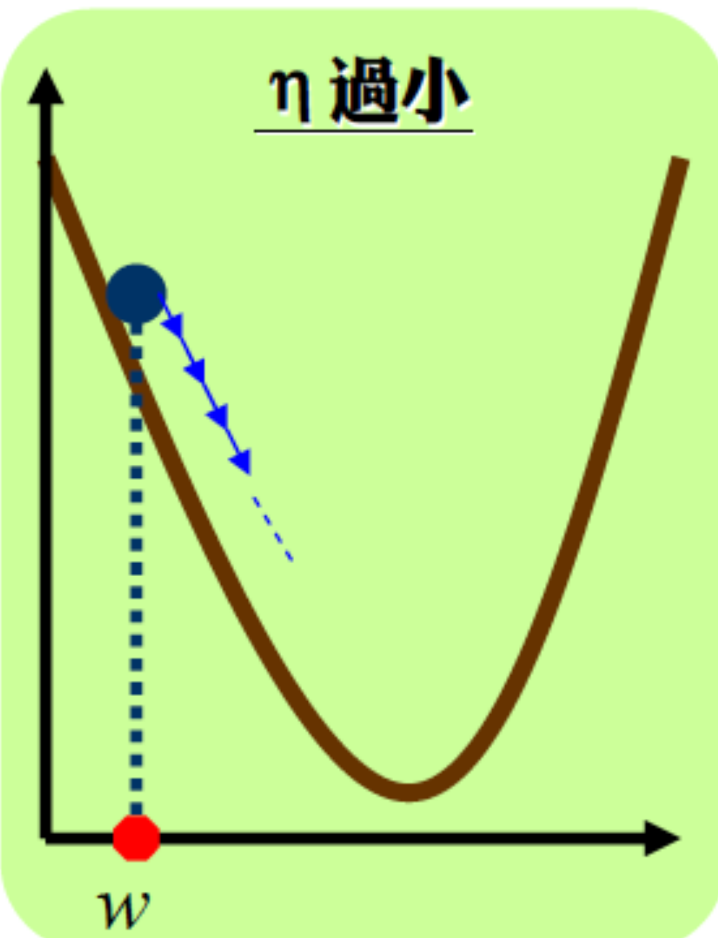
$$\theta := 4 - 0.1(2 \cdot 4 - 2) = 4 - 0.6 = 3.4$$

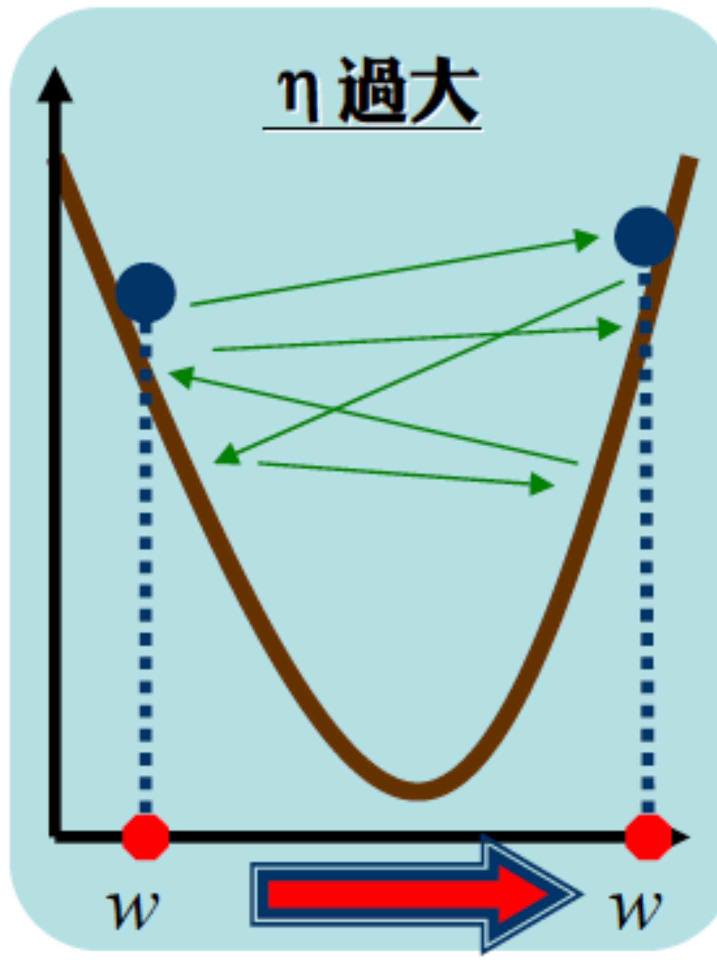
$$\theta := 3.4 - 0.1(2 \cdot 3.4 - 2) = 3.4 - 0.48 = 2.92$$

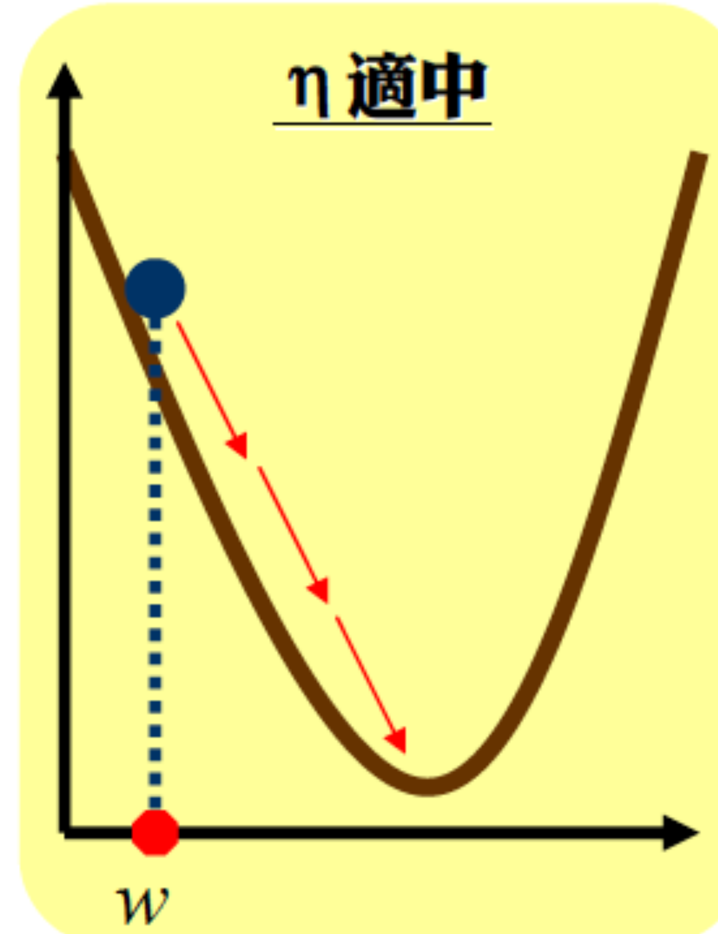
$$\theta := 2.92 - 0.1(2 \cdot 2.92 - 2) = 2.92 - 0.384 = 2.536$$

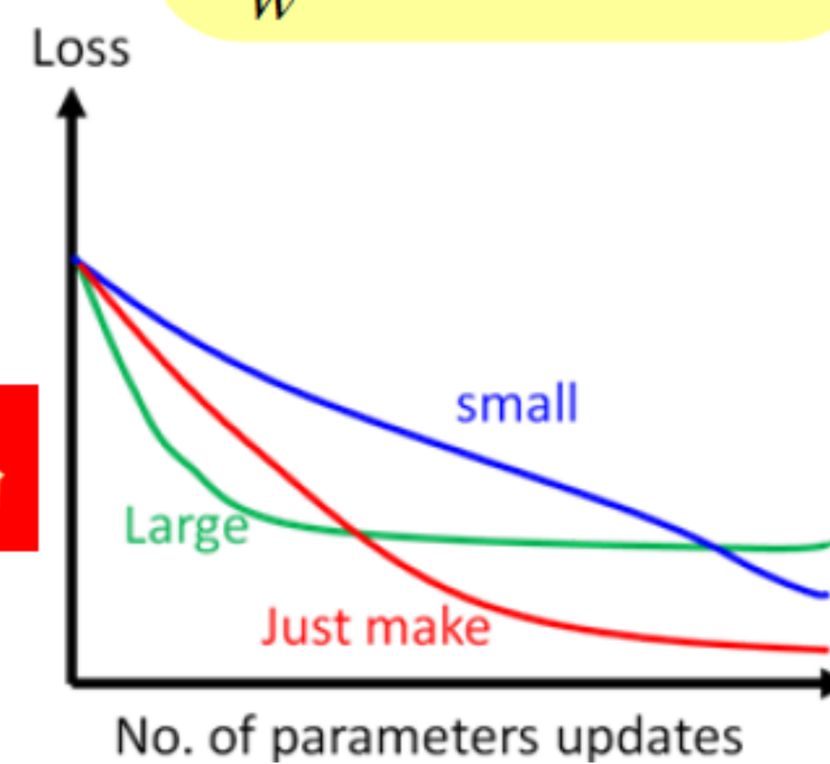
Gradient descent (梯度下降法)

- η的設定，過大過小都不適宜…









要找到一個固定且適合的η不容易

Gradient descent（梯度下降法）

- Cost function

$$E(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent

$$\theta_0 := \theta_0 - \eta\frac{\partial E}{\partial \theta_0}$$

$$\theta_1 := \theta_1 - \eta\frac{\partial E}{\partial \theta_1}$$

Gradient descent（梯度下降法)

- Cost function

$$E(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent

Simultaneous update

$$\theta_0 := \theta_0 - \eta \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \eta \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

Gradient descent (梯度下降法)

- 假設 $\theta_0 = 0, \theta_1 = 0, \eta = 0.5$

x	y
-3	6
-1	4
0	2
1	0
4	-8

Gradient descent (梯度下降法)

- 假設 $\theta_0 = 0, \theta_1 = 0, \eta = 0.5$

x	y	$h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) - y$	$\left(h_\theta(x) - y\right) \cdot x$
-3	6			
-1	4			
0	2			
1	0			
4	-8			

Gradient descent（梯度下降法)

- 假設 $\theta_0 = 0, \theta_1 = 0, \eta = 0.5$

x	y	$h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) - y$	$\left(h_\theta(x) - y\right) \cdot x$
-3	6	0	-6	18
-1	4	0	-4	4
0	2	0	-2	0
1	0	0	0	0
4	-8	0	8	32

$$\theta_0 := \theta_0 - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) = 0 - 0.1 * (-4) = 0.4$$

$$\theta_1 := \theta_1 - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)} = 0 - 0.1 * 54 = -5.4$$

$$h_\theta(x) = \theta_0 + \theta_1 x = 0.4 - 5.4x$$

Gradient descent（梯度下降法)

- Cost function

$$E(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent

$$\theta_0 := \theta_0 - \eta\frac{\partial E}{\partial \theta_0}$$

$$\theta_1 := \theta_1 - \eta\frac{\partial E}{\partial \theta_1}$$

Simultaneous update

Gradient descent（梯度下降法)

- Gradient descent (Simultaneous update)

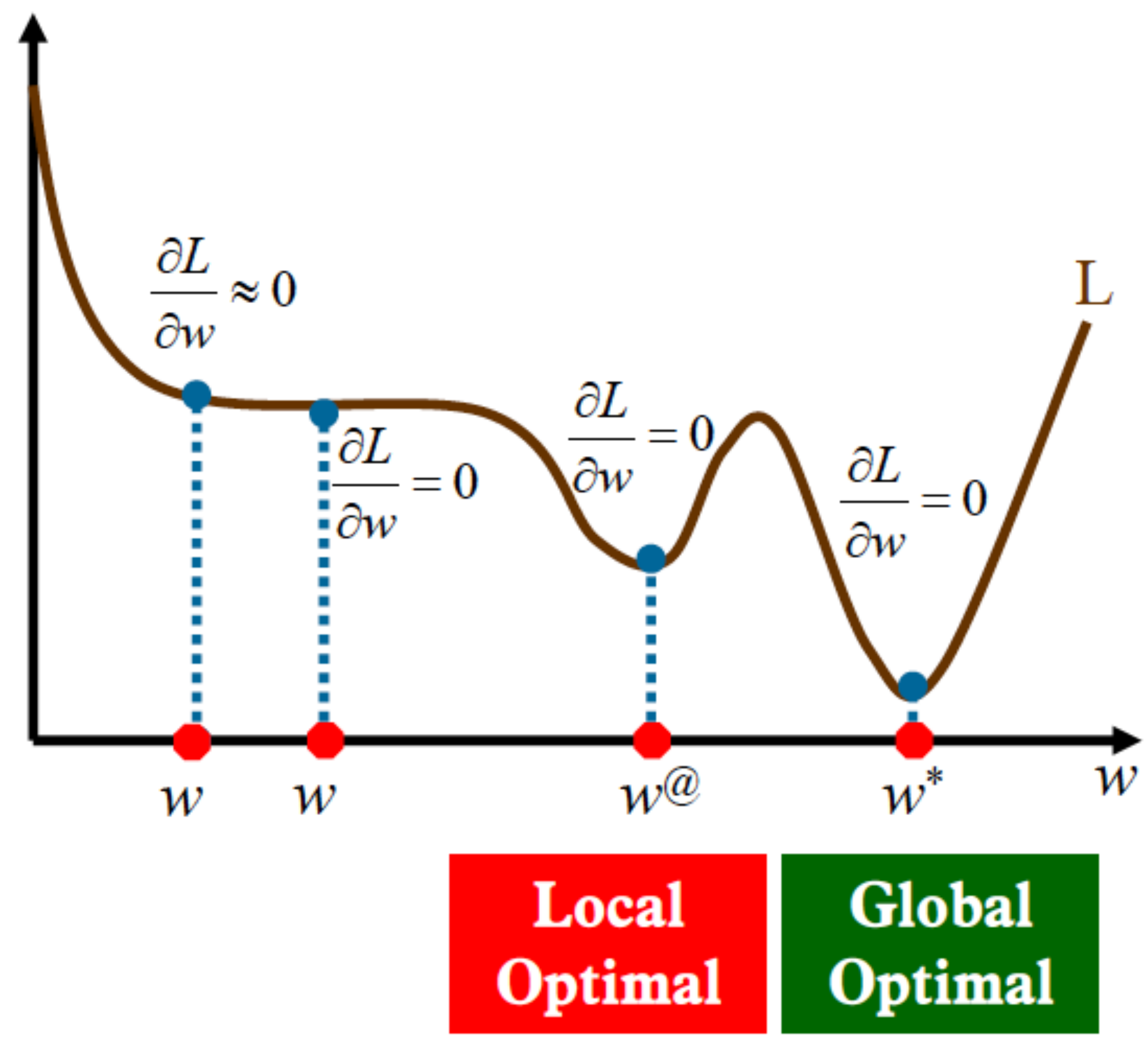
$$temp0 := \theta_0 - \eta \frac{\partial E}{\partial \theta_0}$$

$$temp1 := \theta_1 - \eta \frac{\partial E}{\partial \theta_1}$$

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

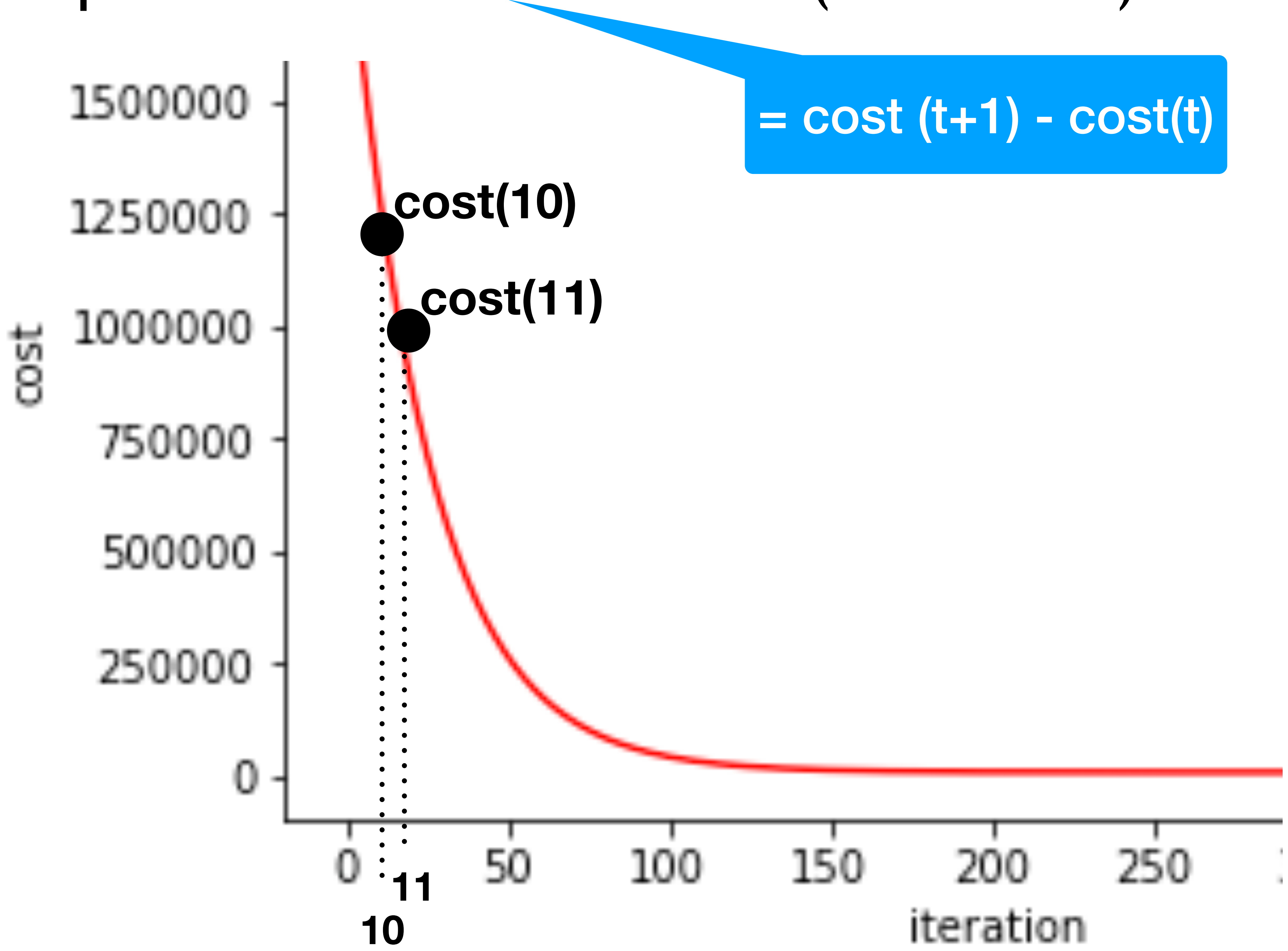
Gradient descent（梯度下降法)



缺點：可能只會收斂到區域最小值，而找不到global optimal

When to stop?

- Stop when the diff is small (ex. <0.01)

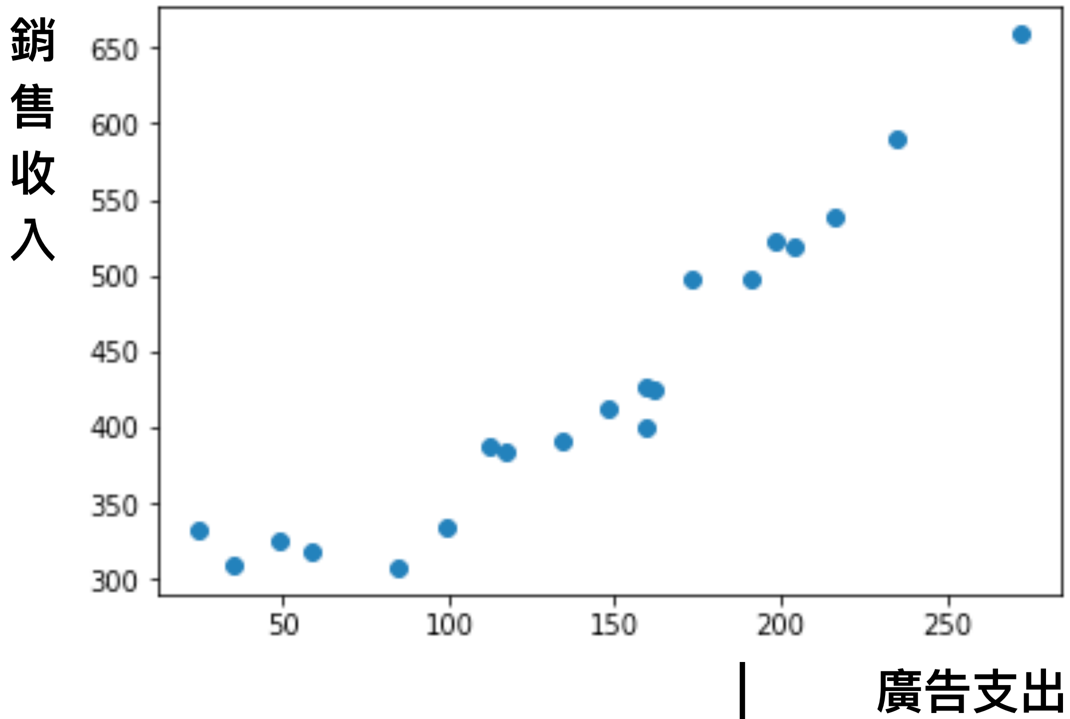


Linear regression
- using sklearn

Linear regression

- Loading data set and normalization

```
# loading libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# loading training data
data = pd.read_csv('regression1.csv')
X = data.iloc[:,0].values
y = data.iloc[:,-1].values

# ========= normalization ===================================
from sklearn.preprocessing import StandardScaler

sc_x = StandardScaler()
X_std = sc_x.fit_transform(X)
```



Data normalization

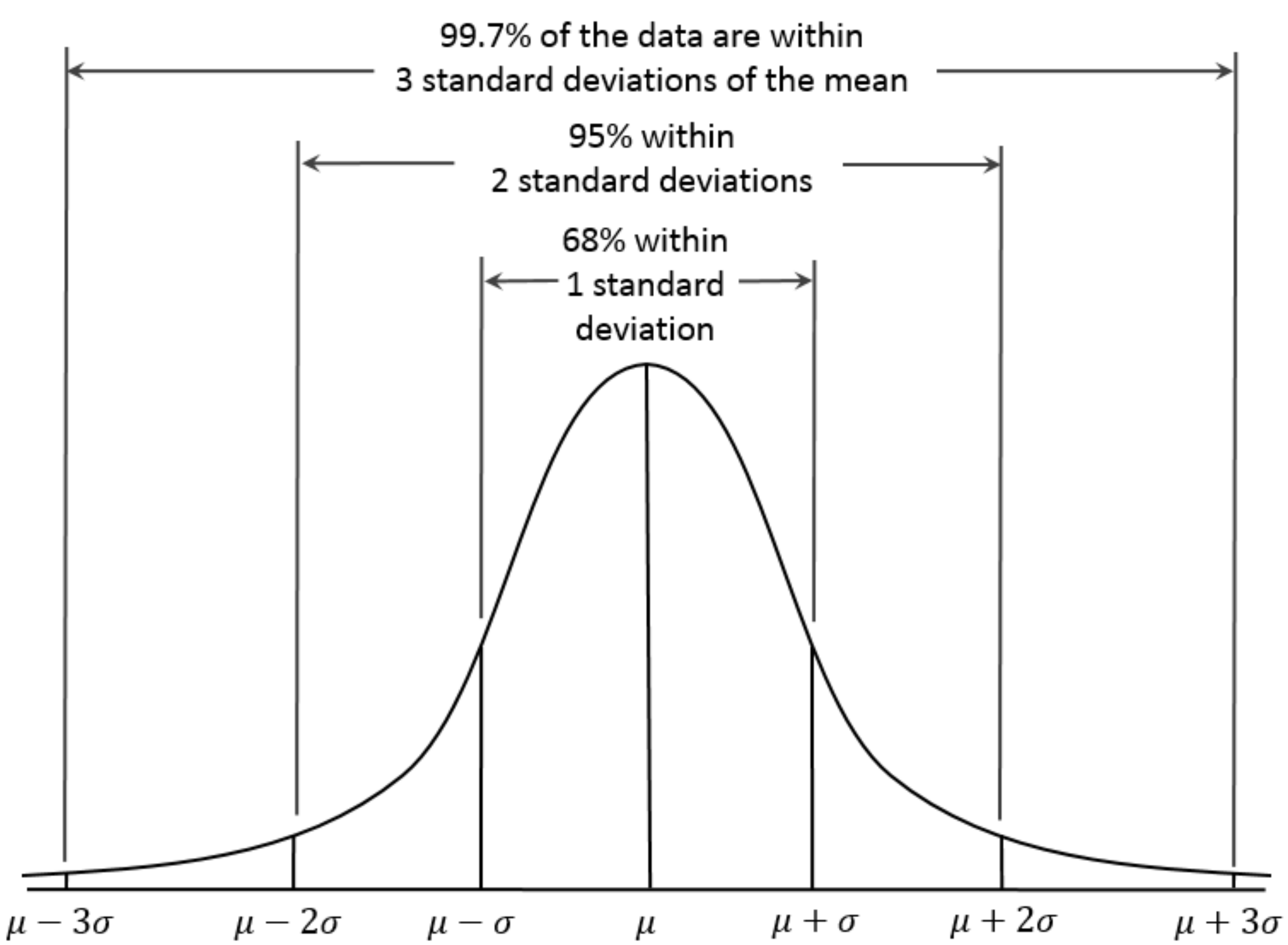
- Z-score標準化 (μ : 均值, σ : 標準差):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - 此標準化做法可使所有標準化後的值，其平均值為0。

- 例. 當 μ = \$54,000, σ = \$16,000. 則透過Z-score標準化，輸入值\$73600會轉換為：

$$\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$$



Linear regression

- 標準化

```
# ========== normalization ==========================================
from sklearn.preprocessing import StandardScaler

sc_x = StandardScaler()
X1 = X.reshape(-1,1)

X_std = sc_x.fit_transform(X1)
```

在scikit-learn中，希望數據要儲存在二維陣列中，而X是一個一維陣列

- 相關係數

```
data.corr()
```

	X	y
X	1.000000	0.949494
y	0.949494	1.000000

Linear regression

- Linear Regression

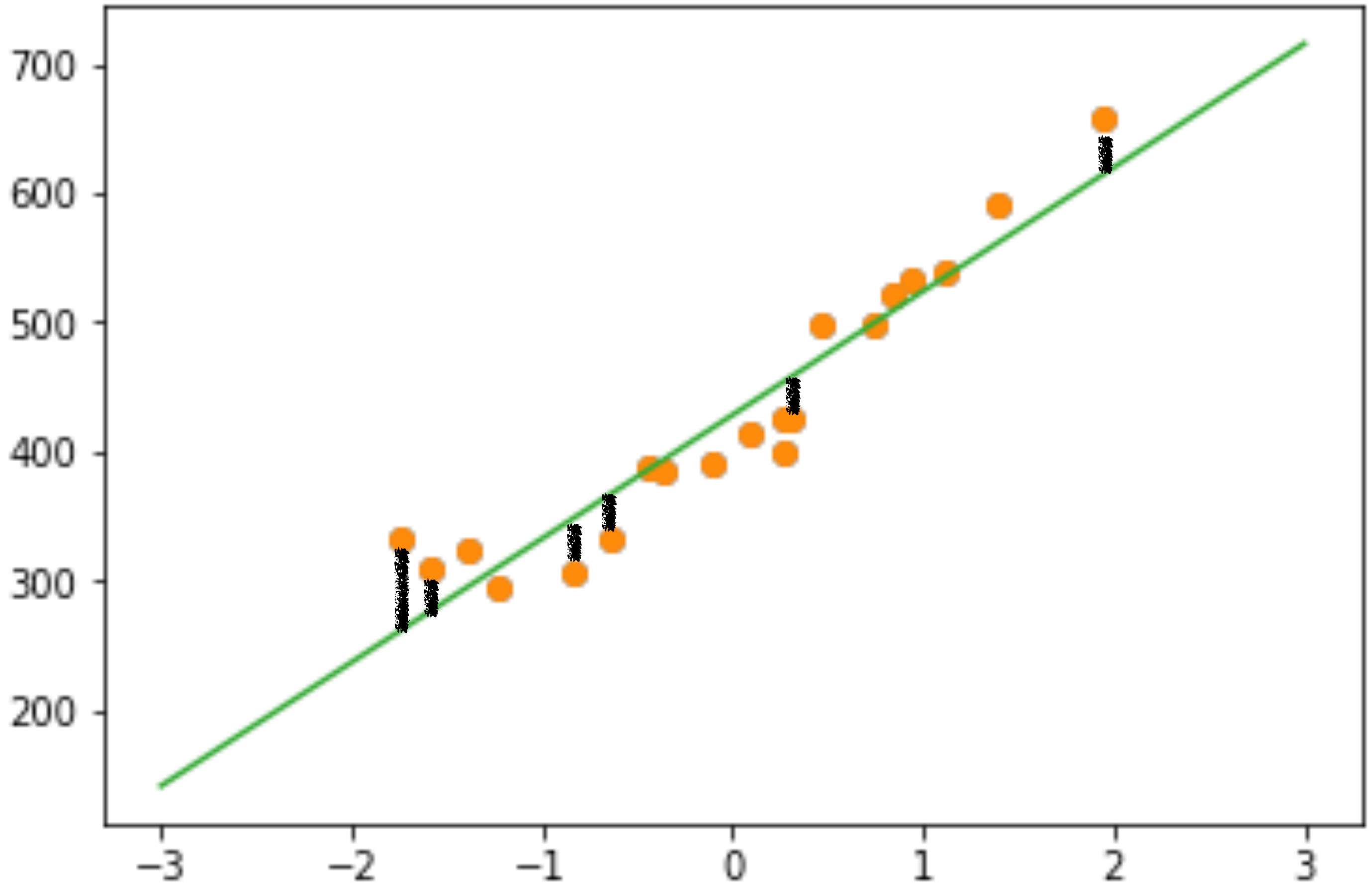
```
# ========= Linear Regression  (normalization) =================

from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_std, y)
y_pred = lr.predict(X_std)
print('Slope: %.3f' % lr.coef_[0])
print('Intercept: %.3f' % lr.intercept_)
```

Linear regression - 評估模型的效能

- MSE (Mean Squared Error, 均方誤差)

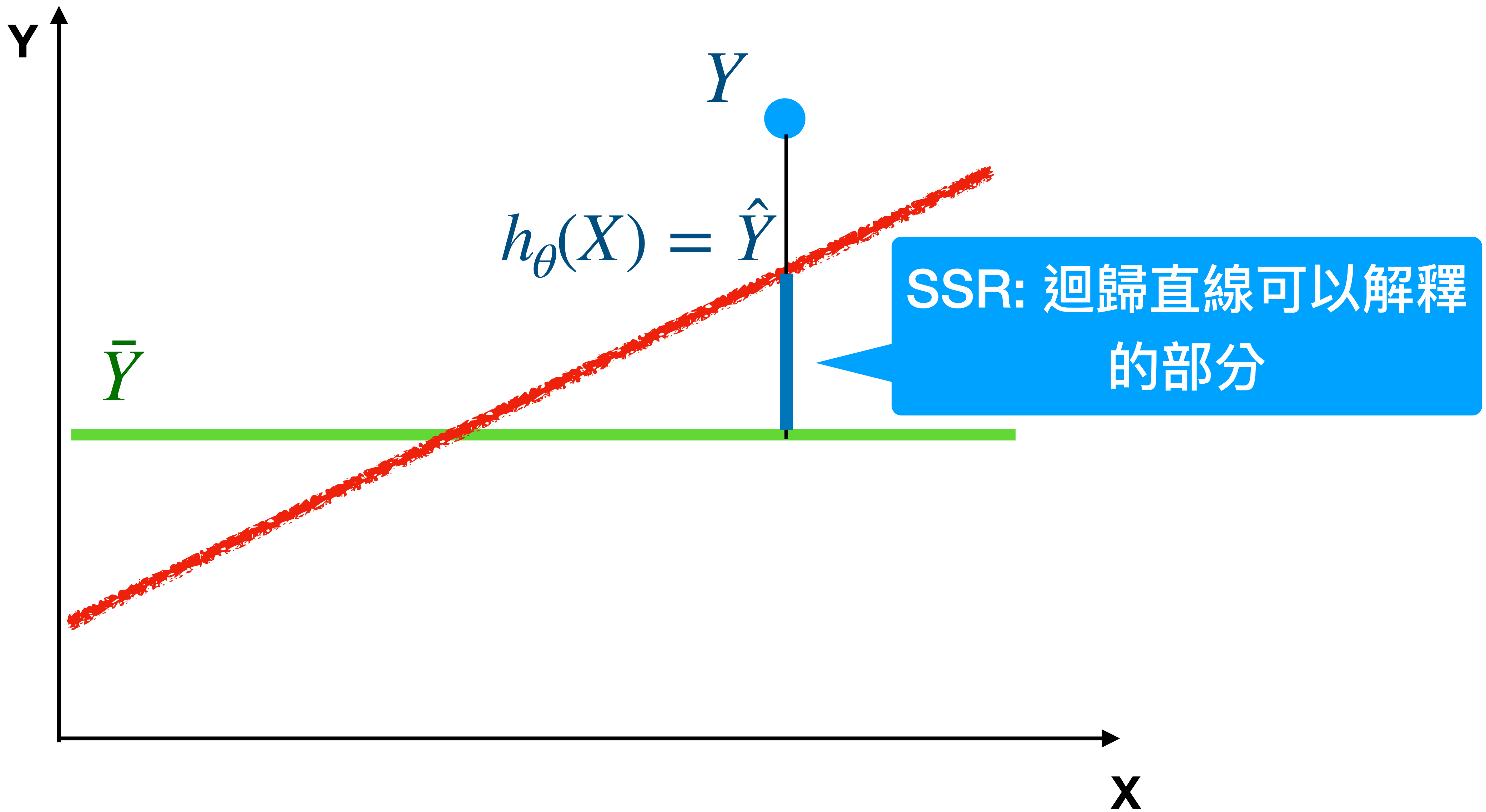
$$\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

除以m是為了消除樣本個數的影響，因為如果樣本很多，只除以2會對結果不利



決定係數(Coefficient of Determination)

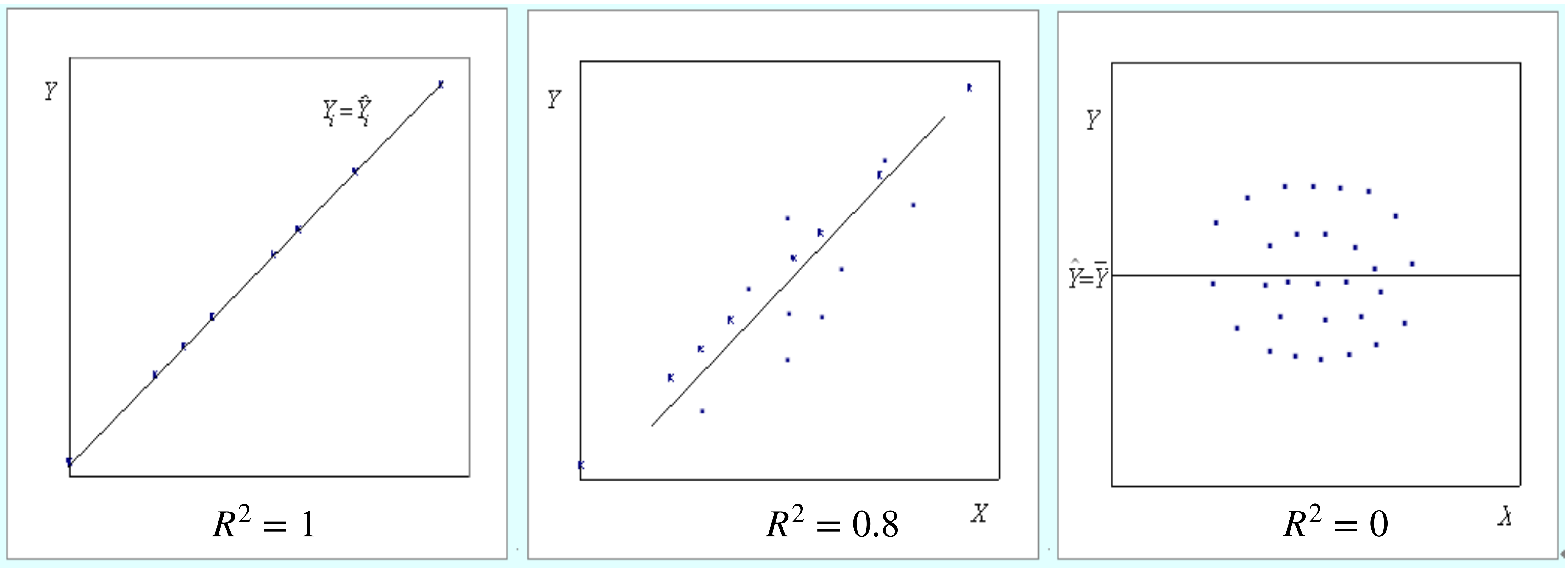
$$\underset{\text{SST}}{\sum_{i=1}^{n}(Y_i-\bar{Y})^2} = \underset{\text{SSR}}{\sum_{i=1}^{n}(\hat{Y}-\bar{Y})^2} + \underset{\text{SSE}}{\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2}$$



決定係數(Coefficient of Determination)

$$R^2 = \frac{\text{SSR}}{\text{SST}} = \frac{\sum(\hat{Y}_i - \overline{Y})^2}{\sum(Y_i - \overline{Y})^2}$$

可以理解為標準化的MSE



Linear regression

- Computing MSE 和 判定係數

```
import sklearn.metrics as sm

print('MSE: %.3f' % sm.mean_squared_error(y, y_pred))
print('R^2: %.3f' % sm.r2_score(y, y_pred))
```

Code (4/6)

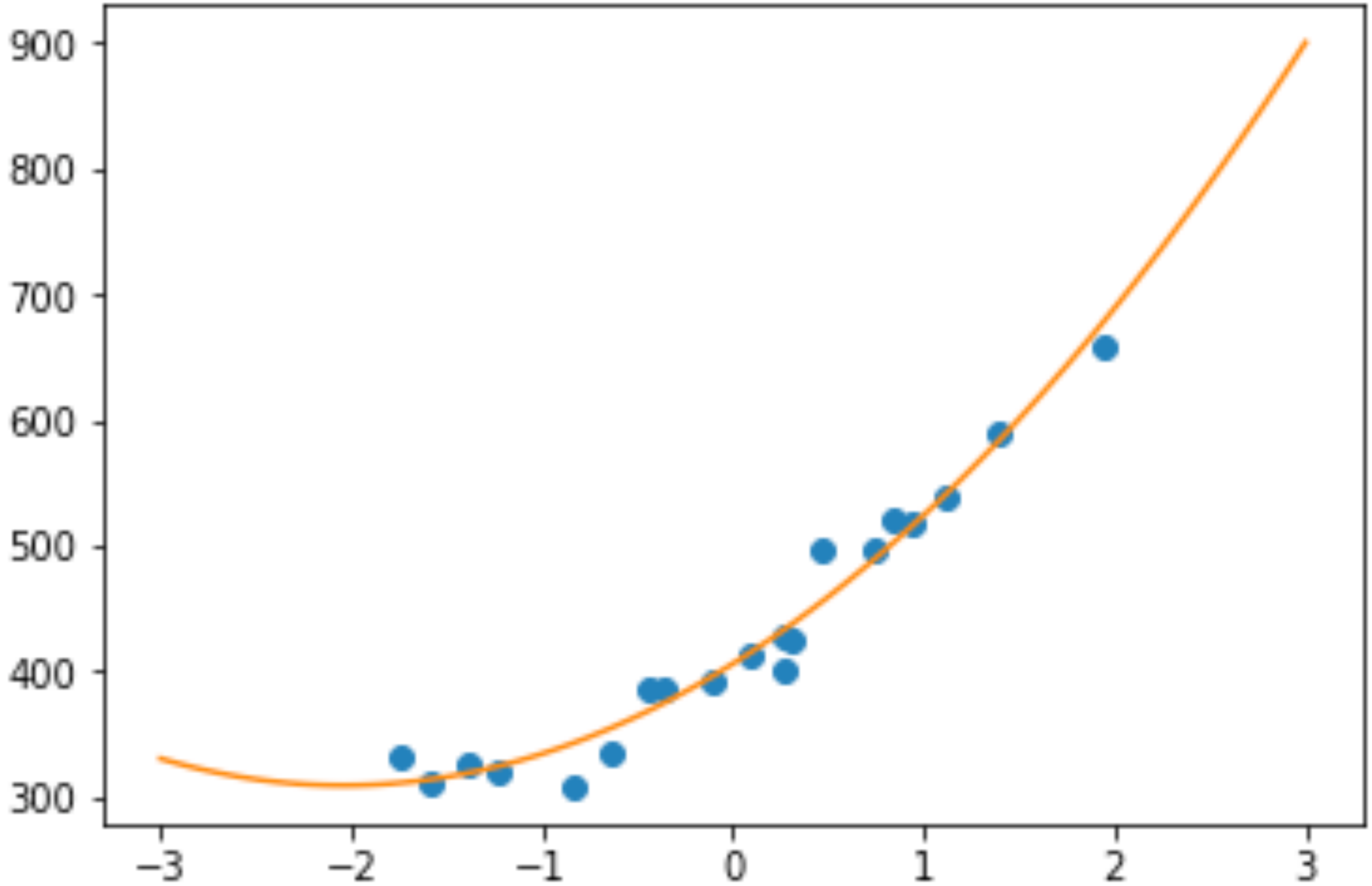
```
MSE: 978.262
R^2: 0.903
```

此公式只能解釋 90.3% 銷售收入變因

Example

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

嘗試用多元曲線擬合看效果

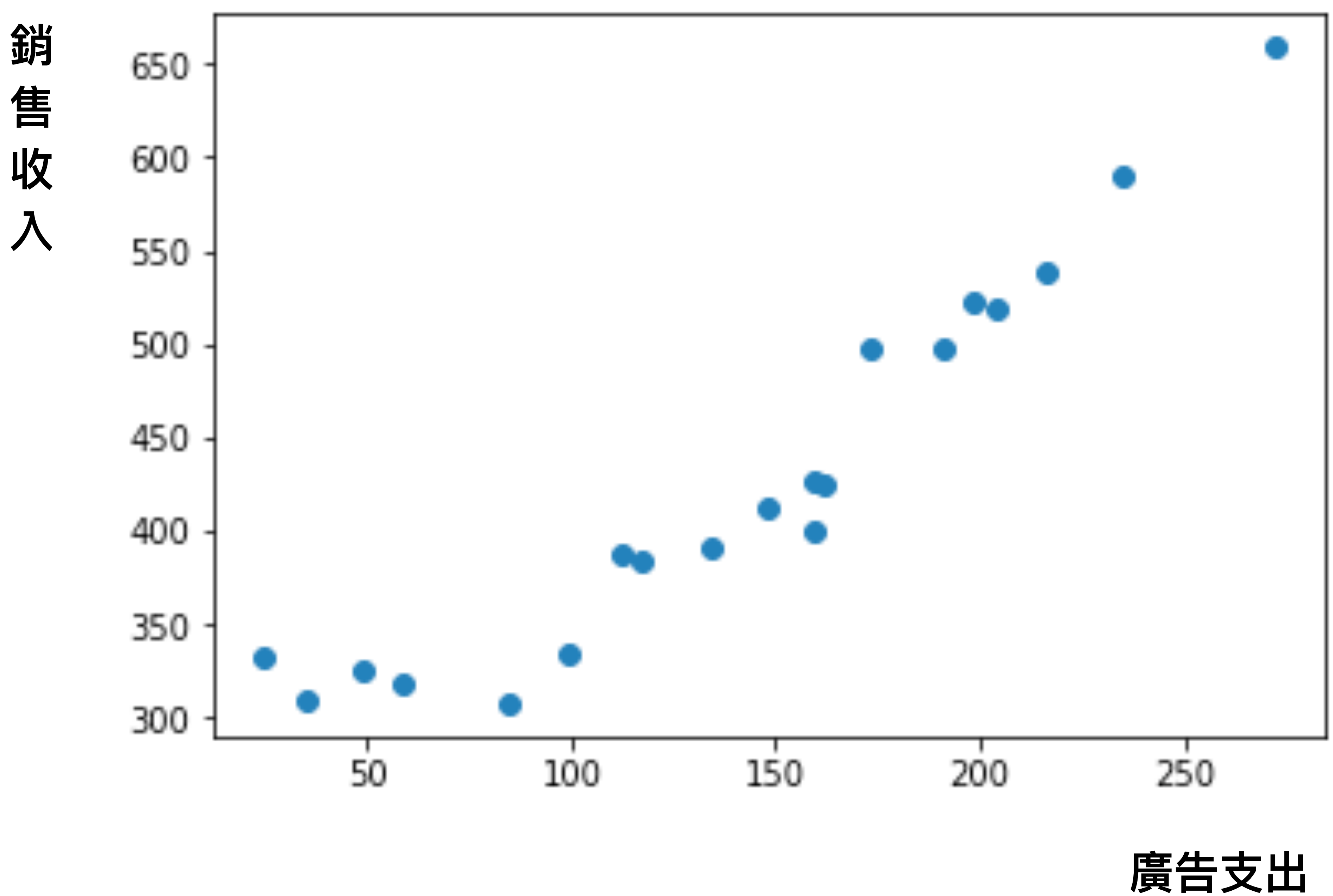


Question

Overfitting

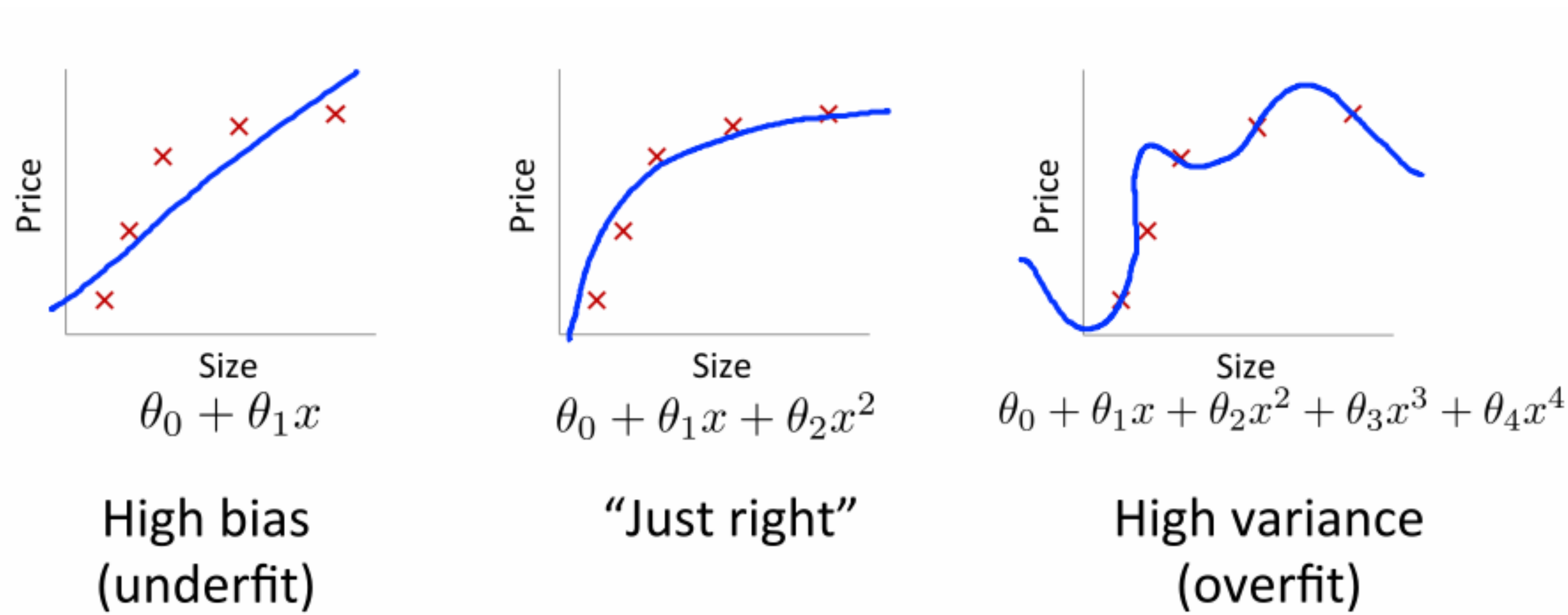
- How about

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n$$



Overfitting v.s. underfitting

Source: http://murphymind.blogspot.com/2017/06/machine-learning-advice-for-applying.html



Gradient descent（梯度下降法)

- Cost function

$$E(\theta_0, \theta_1, \theta_2) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent

$$\theta_0 := \theta_0 - \eta\frac{\partial E}{\partial \theta_0}$$

$$\theta_1 := \theta_1 - \eta\frac{\partial E}{\partial \theta_1}$$

$$\theta_2 := \theta_2 - \eta\frac{\partial E}{\partial \theta_2}$$

Gradient descent（梯度下降法)

repeat until convergence

$$\theta_0 := \theta_0 - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

$$\theta_2 := \theta_2 - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) (x^{(i)})^2$$

Polynomial regression

- Polynomial Regression

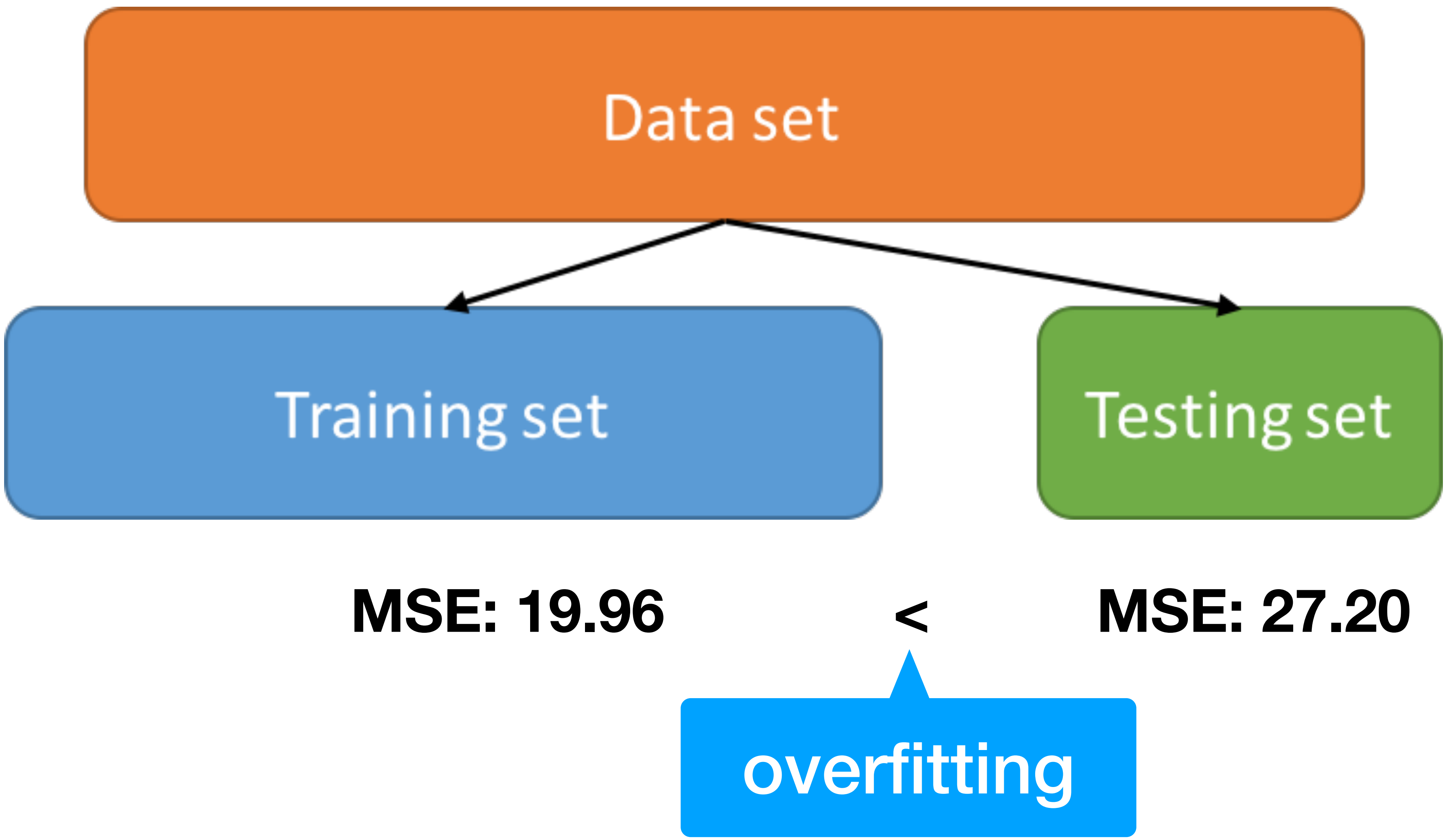
```
from sklearn.preprocessing import PolynomialFeatures

pr = LinearRegression()
quadratic = PolynomialFeatures(degree=2)
X_quad = quadratic.fit_transform(X_std)

# fit linear features
pr.fit(X_quad, y)
y_quad_pred = pr.predict(X_quad)
print('theta1: %.3f' % pr.coef_[1])
print('theta2: %.3f' % pr.coef_[2])
print('Intercept: %.3f' % pr.intercept_)
```

增加一個二次項

訓練回歸模型

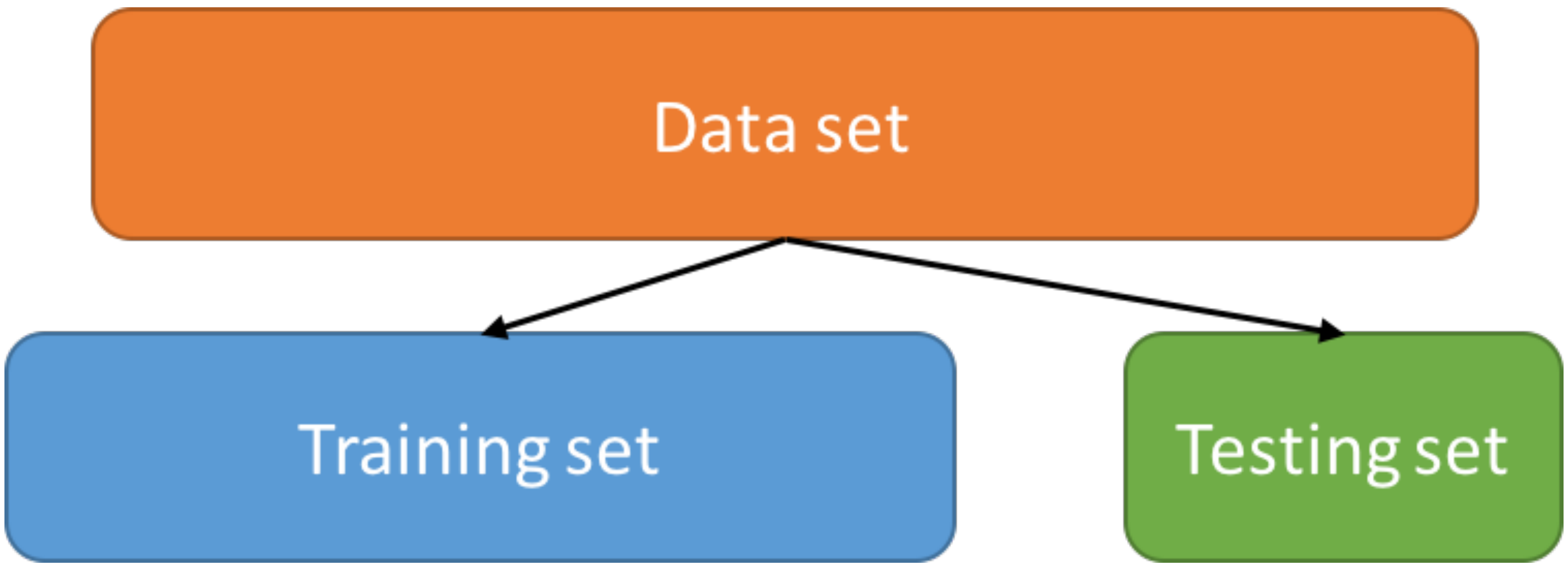
Overfitting v.s. underfitting

Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/



Overfitting v.s. underfitting

Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/

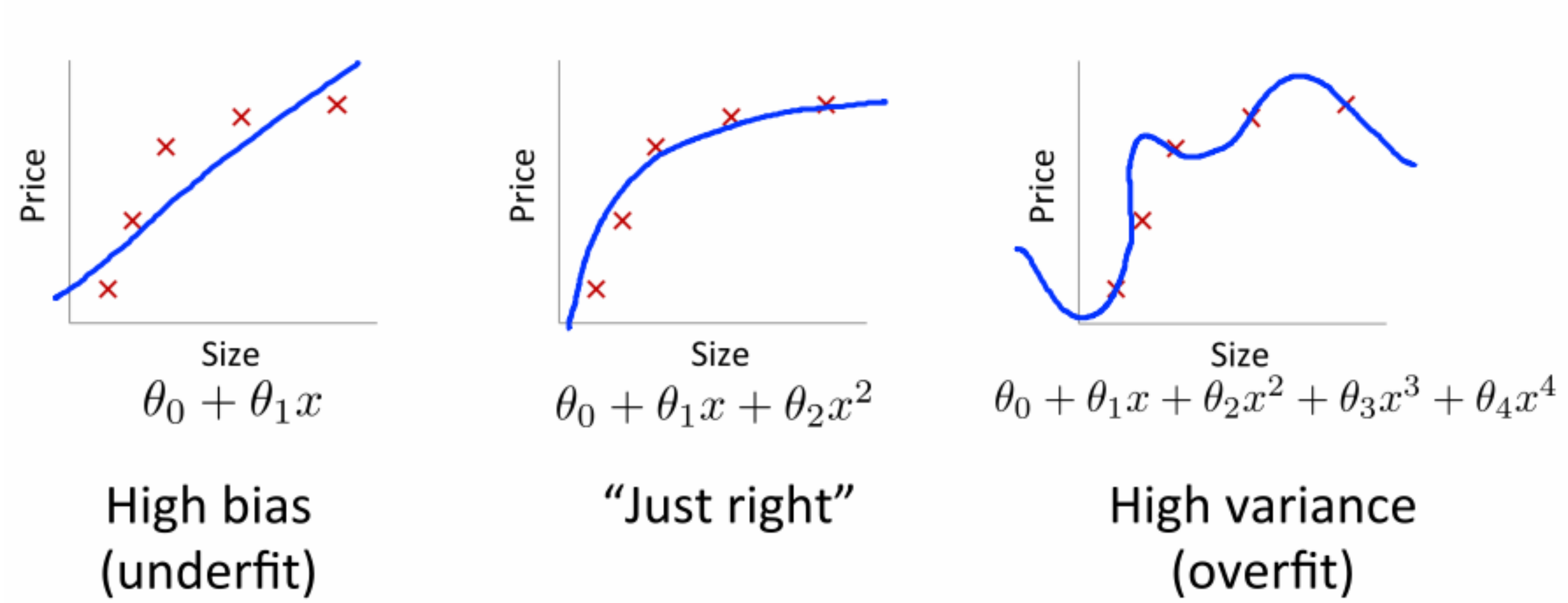


$$R^2 = 0.765 \quad > \quad R^2 = 0.673$$

overfitting

Overfitting v.s. underfitting

Source: http://murphymind.blogspot.com/2017/06/machine-learning-advice-for-applying.html



$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2}$$

糖尿病數據集

- Diabetes dataset
- 10 feature variables （已標準化）
  - Age
  - Sex
  - Body mass index (BMI)
  - Average blood pressure (平均血壓)
  - S1 ~ S6：6 種生理數據
- 目標變數：一年後病情發展的狀況

糖尿病數據集

- 載入數據集

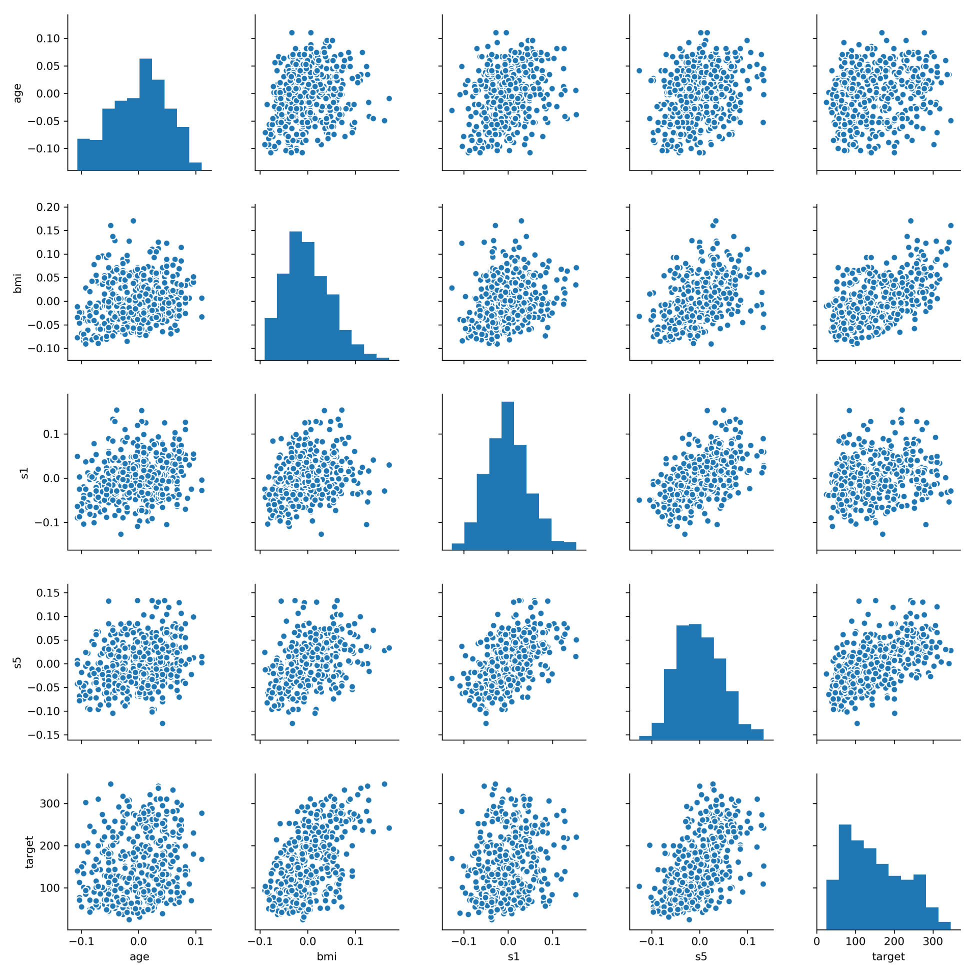
```python
from sklearn.datasets import load_diabetes

data = load_diabetes()
data.keys()


import pandas as pd

feature = pd.DataFrame(data['data'], columns = data['feature_names'])

target = pd.DataFrame(data['target'], columns = ['target'])

df = pd.concat([feature, target], axis = 1)
```

feature.shape

糖尿病數據集

- 畫出散點圖

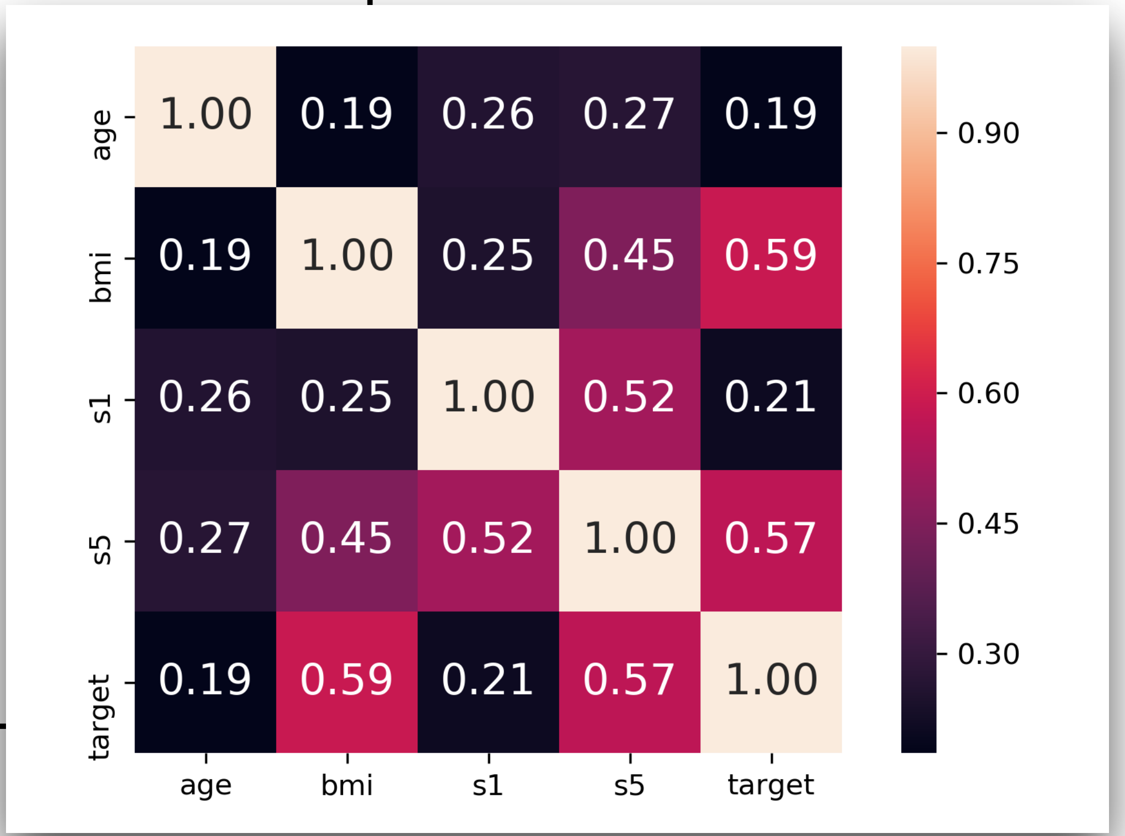
```
import matplotlib.pyplot as plt
import seaborn as sns

cols = ['age', 'bmi', 's1', 's5', 'target']

sns.pairplot(df[cols])
plt.tight_layout()
plt.savefig('scatterplot.png', dpi=300)
plt.show()
```



糖尿病數據集

- 以熱度圖（heat map）畫出相關係數矩陣 (correlation matrix)

```python
import numpy as np

cm = np.corrcoef(df[cols].values.T)
#sns.set(font_scale=1.5)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols,
                 xticklabels=cols)

plt.tight_layout()
plt.savefig('correlation.png', dpi=300)
plt.show()
```



複迴歸

$$h_\theta(x_1, x_2, x_3) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \overrightarrow{\theta} \cdot \overrightarrow{x}$$

where

$$\overrightarrow{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad \overrightarrow{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Normal equation

$$h_\theta(x) = \overrightarrow{\theta} \cdot \overrightarrow{x}$$

$$X = \begin{bmatrix} - - x^{(1)} - - \\ \vdots \\ - - x^{(n)} - - \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- Cost function

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2}\|X\overrightarrow{\theta} - \overrightarrow{y}\|^2$$

$$= \frac{1}{2}\left(\overrightarrow{\theta}^T X^T X \overrightarrow{\theta} - 2\overrightarrow{\theta}^T X^T \overrightarrow{y} + \overrightarrow{y}^T \overrightarrow{y}\right)$$

Normal equation

$$h_\theta(x) = \overrightarrow{\theta} \cdot \overrightarrow{x}$$

- Cost function

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$\Rightarrow \nabla E(\theta) = X^T X \overrightarrow{\theta} - X^T \overrightarrow{y} \equiv \overrightarrow{0}$$

$$\Rightarrow \overrightarrow{\theta} = \left(X^T X\right)^{-1} X^T \overrightarrow{y}$$

Gradient descent（梯度下降法)

- Cost function

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Gradient descent

$$\theta_j := \theta_j - \eta\frac{\partial E}{\partial \theta_j} \qquad \textbf{for j = 0, 1, ..., n}$$

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

Normalization

Why normalize inputs?

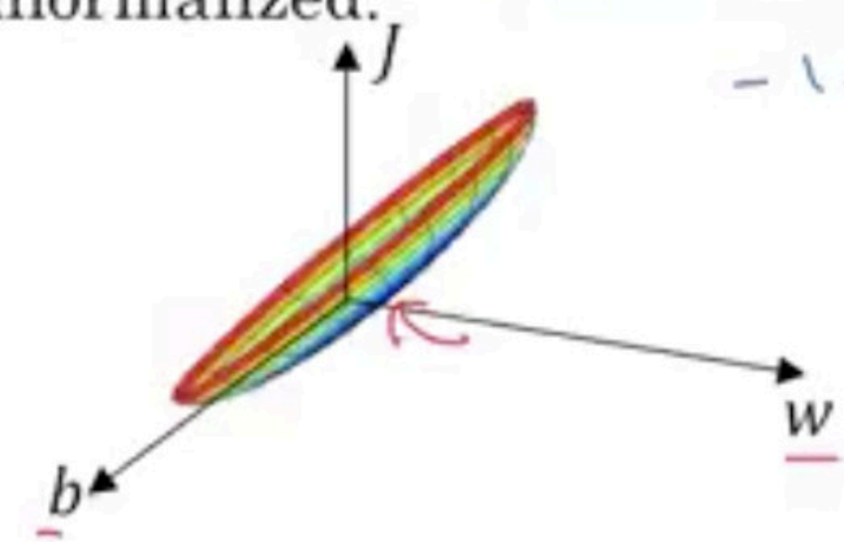
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

w_1 x_1: 1 ... 1000

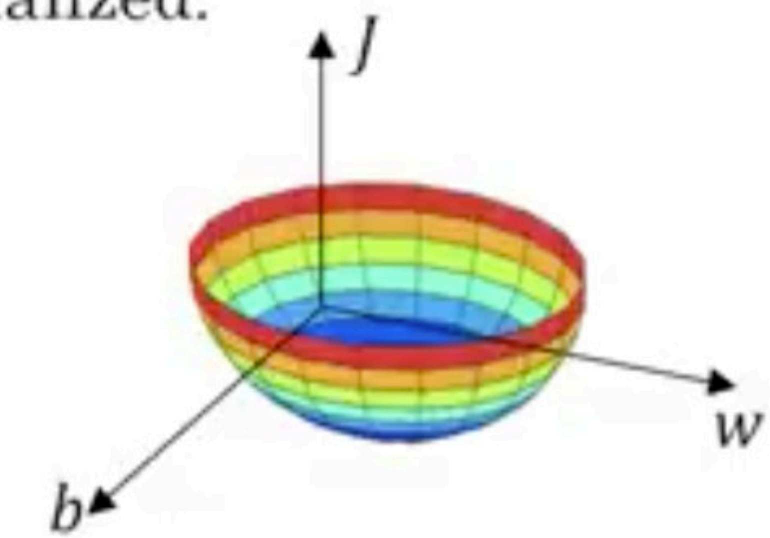
w_2 x_2: 0 ... 1

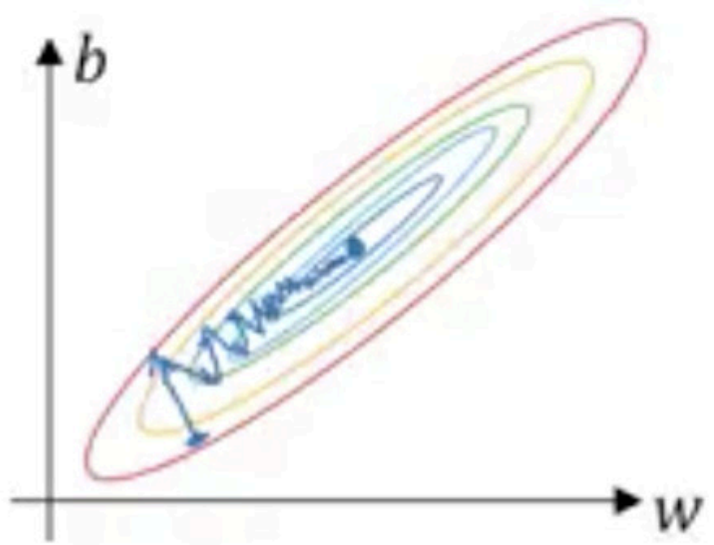
-1 ... 1

Unnormalized:



Normalized:

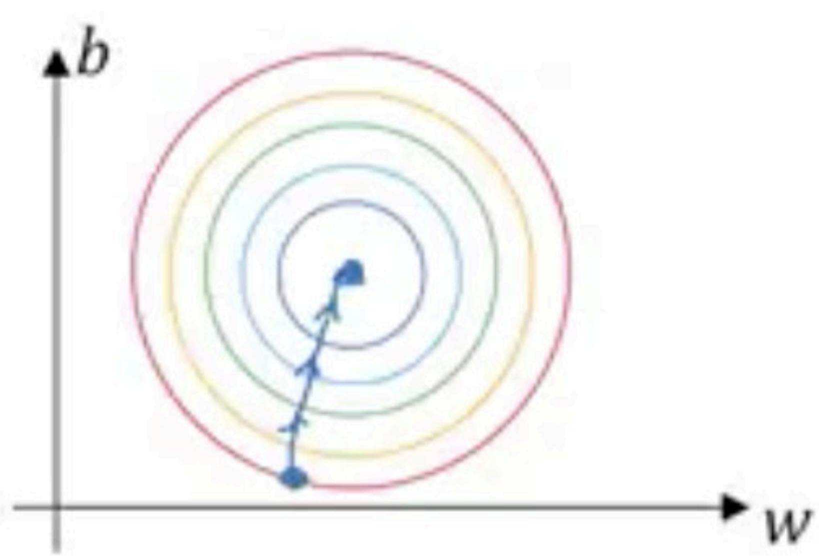




x_1: 0 ... 1

x_2: -1 ... 1

x_3: 1 ... 2



Andrew Ng

糖尿病數據集

- 將糖尿病數據集分成training set 及 test set

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

X,y = load_diabetes().data, load_diabetes().target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=8)
```

糖尿病數據集

- 訓練線性迴歸模型，並計算MSE及 R^2

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

slr = LinearRegression()

slr.fit(X_train, y_train)
print(slr.coef_)
y_train_pred = slr.predict(X_train)
y_test_pred = slr.predict(X_test)

print('MSE train: %.3f, test: %.3f' % (
        mean_squared_error(y_train, y_train_pred),
        mean_squared_error(y_test, y_test_pred)))
print('R^2 train: %.3f, test: %.3f' % (
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

Linear regression 的結果

Age	Sex	BMI	ABP	S1
[11.5106203	-282.51347161	534.20455671	401.73142674	-1043.89718398
634.92464089	186.43262636	204.93373199	762.47149733	91.9460394]
S2	S3	S4	S5	S6

$$\text{Target} = 11.51 * \text{age} + \ldots + 91.94 * \text{S6}$$

迴歸係數

小心共線性

當其他預測因子存在的情況下，該預測因子的強度

Note 1: X 要先標準化 （去除單位的影響）

Note 2: 回歸係數數值越大表示對 Y 的影響力越大

Note 3: 回歸係數為負表示負相關

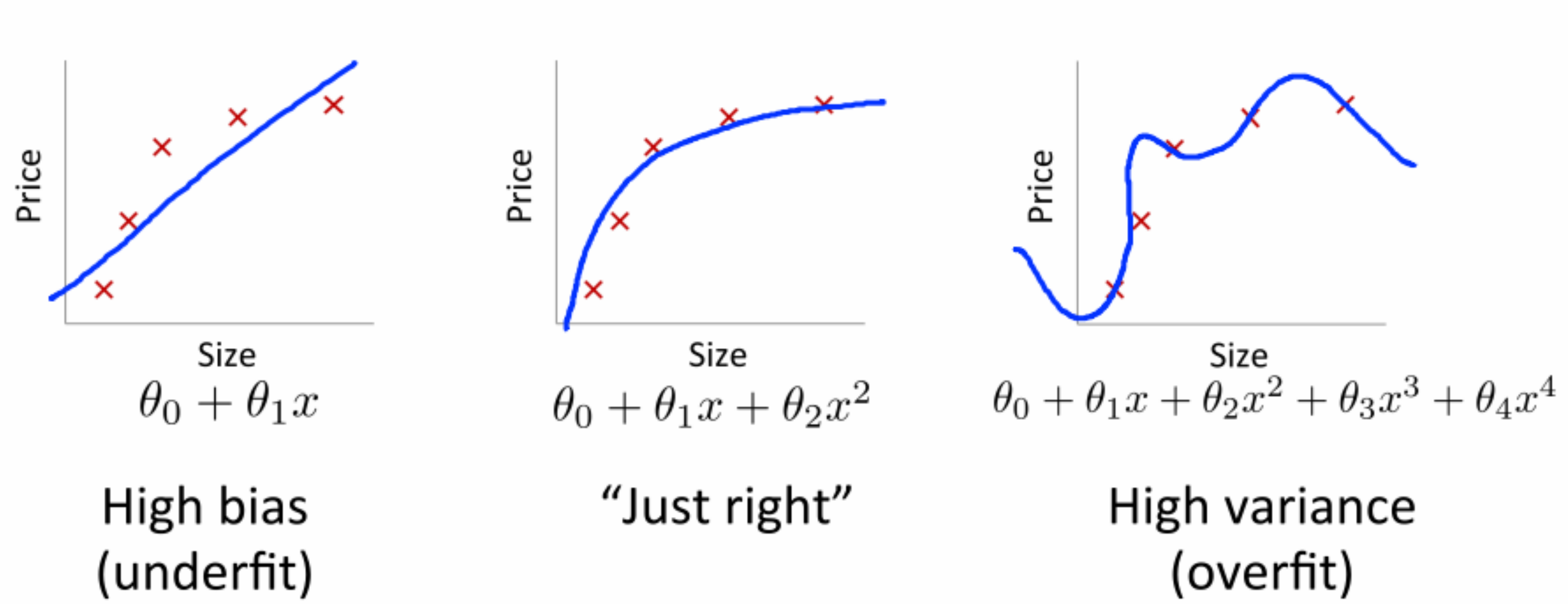
怎麼解決共線性

- 資料預處理
  - 資料轉換
  - 只留下獨立(不相關)的變數
- 脊迴歸（ridge regression）
- 主成分分析 (principal component analysis)

https://taweihuang.hpd.io/2016/09/12/讀者提問：多元迴歸分析的變數選擇/

Overfitting v.s. underfitting

Source: http://murphymind.blogspot.com/2017/06/machine-learning-advice-for-applying.html



$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2}$$

Regularization (正則化)

- 脊迴歸　(Ridge Regression)

不考慮截距項

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \alpha \cdot \sum_{i=1}^{n}\theta_i^2$$

控制正則項的強度

糖尿病數據集

- 訓練Ridge模型，並計算MSE及 R^2

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1).fit(X_train,y_train)  # alpha = 1.0
print(ridge.coef_)

y_train_pred = ridge.predict(X_train)
```

Regularization (正則化)

- 脊迴歸 （Ridge Regression）

不考慮截距項

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \alpha \cdot \sum_{i=1}^{n}\theta_i^2$$

控制正則項的強度

- 最小絕對壓縮挑選機制 （Least Absolute Shrinkage and Selection Operator, LASSO）

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \alpha \cdot \sum_{i=1}^{n}|\theta_i|$$

使某些係數變為 0

糖尿病數據集

- 訓練LASSO模型，並計算MSE及 R^2

```
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=1).fit(X_train,y_train)  # alpha = 1.0
print(lasso.coef_)

y_train_pred = lasso.predict(X_train)
```

Ridge v.s. LASSO

- 實作時，Ridge通常是首選，因為LASSO在移除變數的同時，會犧牲模型的正確性
- 但如果特徵太多，且只有一小部分是真正重要的，那應該選擇LASSO
- 如果須解釋模型，LASSO也更好理解，因為使用較少特徵

Regularization (正則化)

- 彈性網　(Elastic Net)

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + a \cdot \sum_{i=1}^{n} |\theta_i| + b \cdot \sum_{i=1}^{n} \theta_i^2$$

糖尿病數據集

- 訓練Elastic Net模型，並計算MSE及 R^2

```
# ================== Elastic Net ==============================
from sklearn.linear_model import ElasticNet

elanet = ElasticNet(alpha=1, l1_ratio = 0.5).fit(X_train,y_train)
print(elanet.coef_)

y_train_pred = elanet.predict(X_train)
```

l1_ratio = 1 即為 LASSO

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html

Pros & Cons

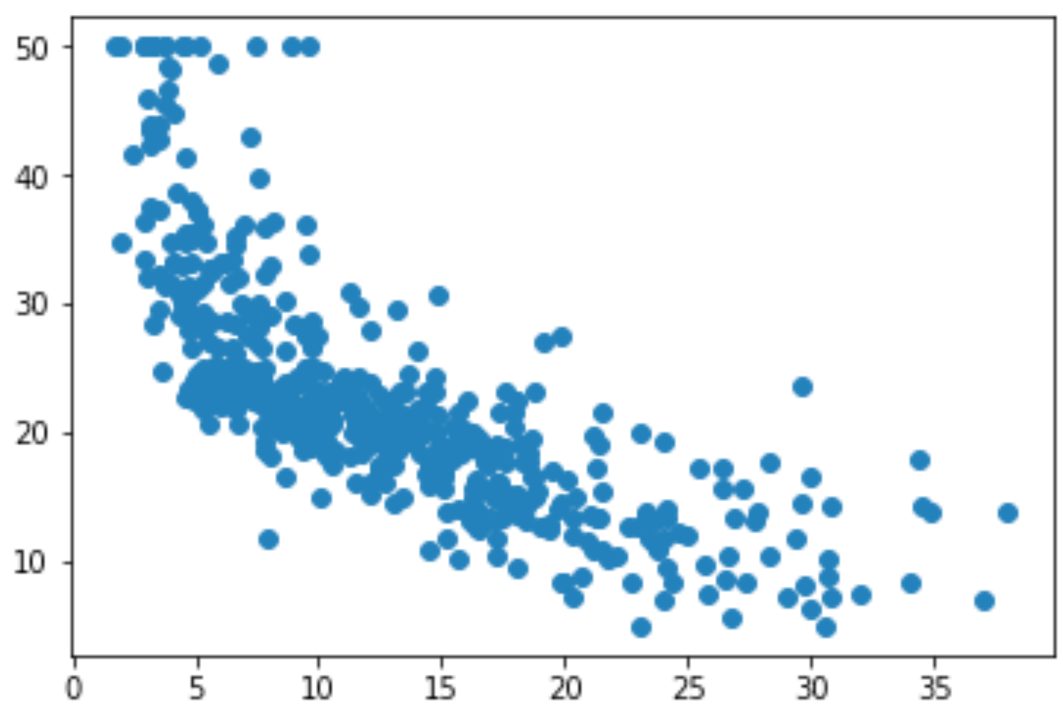
- Pros:
  - 簡單, 直覺, 易於運算
  - 迴歸係數能得到有用的訊息

- Cons:
  - 易受異常值影響
  - 相關預測因子的權重會被扭曲
  - 曲線趨勢

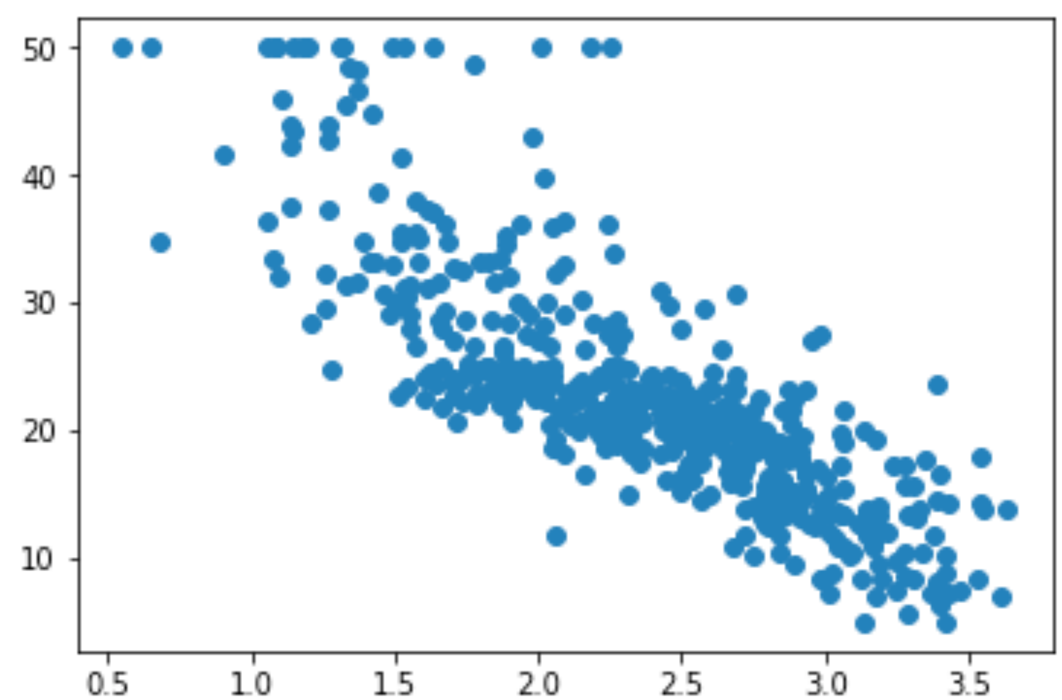
Example: 波士頓房價

- D . Harrison 與 D. L. Rubinfeld 在1978年收集的波士頓郊區的“房價數據集”，其中包含14個特徵，其中 LSTAT(低社經地位的人口比例)

房價中位數，單位：1000美金



LSTAT



log(LSTAT)

Pros & Cons

- Pros:
  - 簡單, 直覺, 易於運算
  - 迴歸係數能得到有用的訊息
- Cons:
  - 易受異常值影響
  - 相關預測因子的權重會被扭曲
  - 曲線趨勢
  - 預測因子和結果並不暗示因果關係