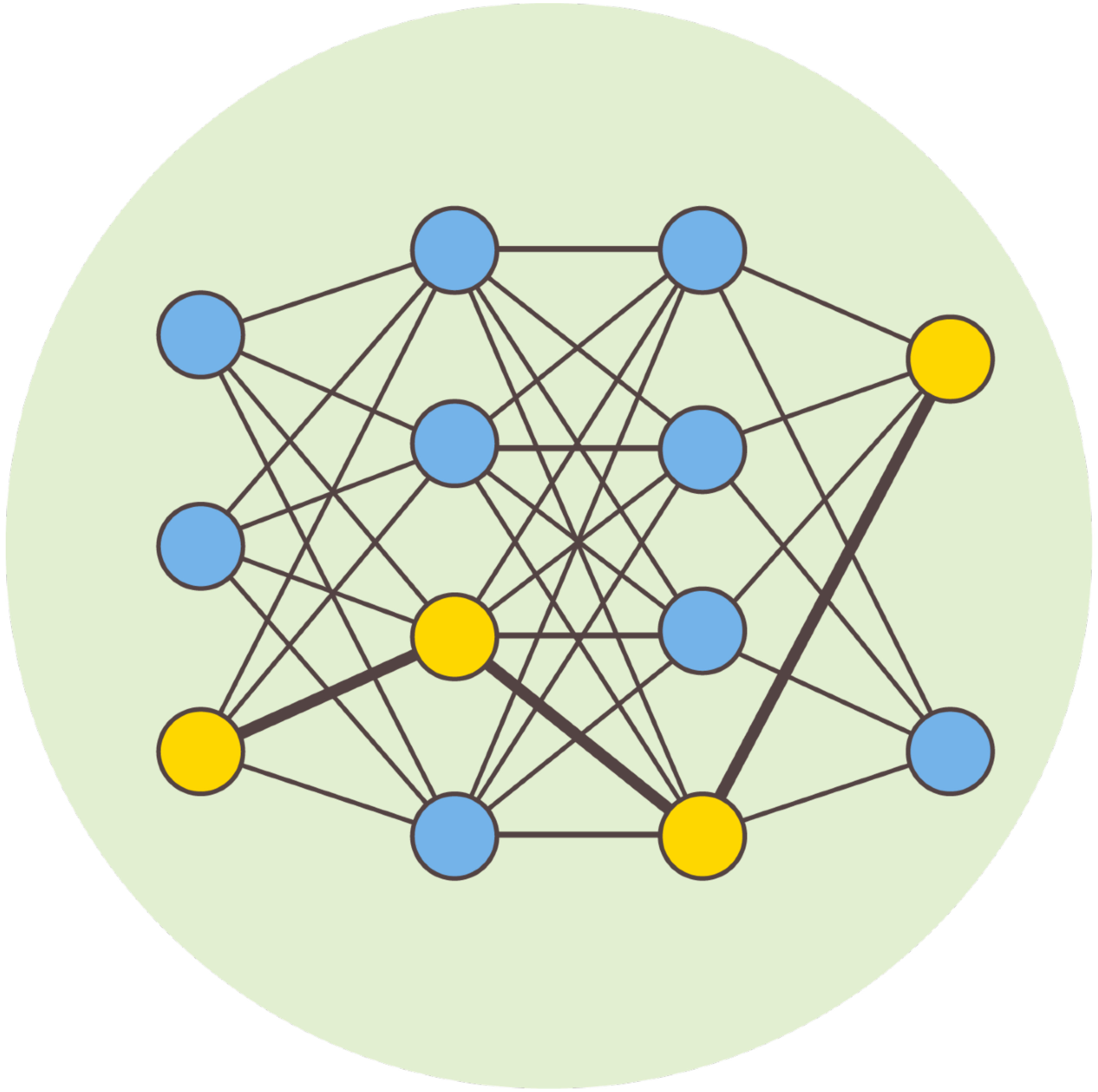
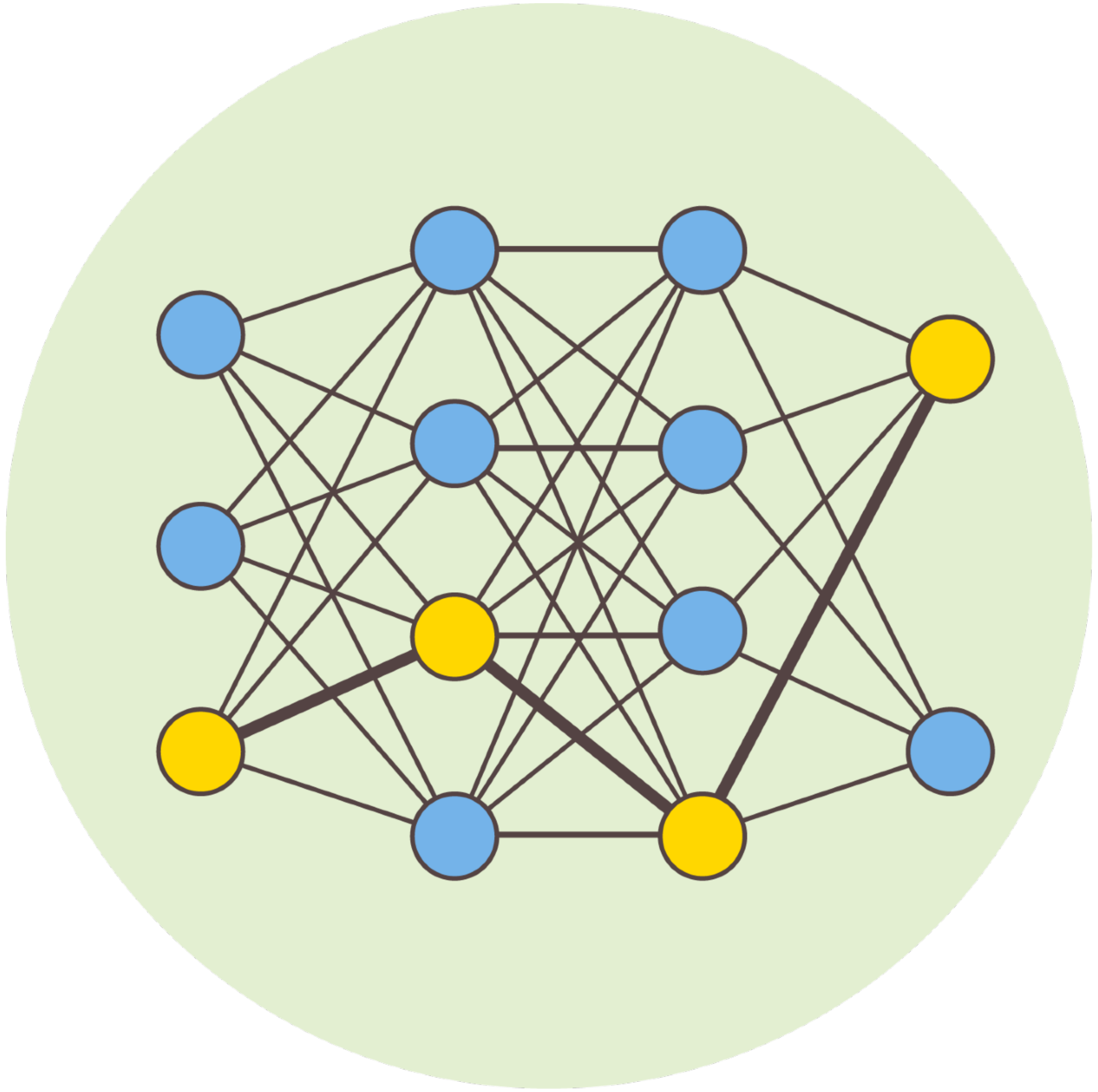


Python 機器學習與深度學習實作

機器學習流程



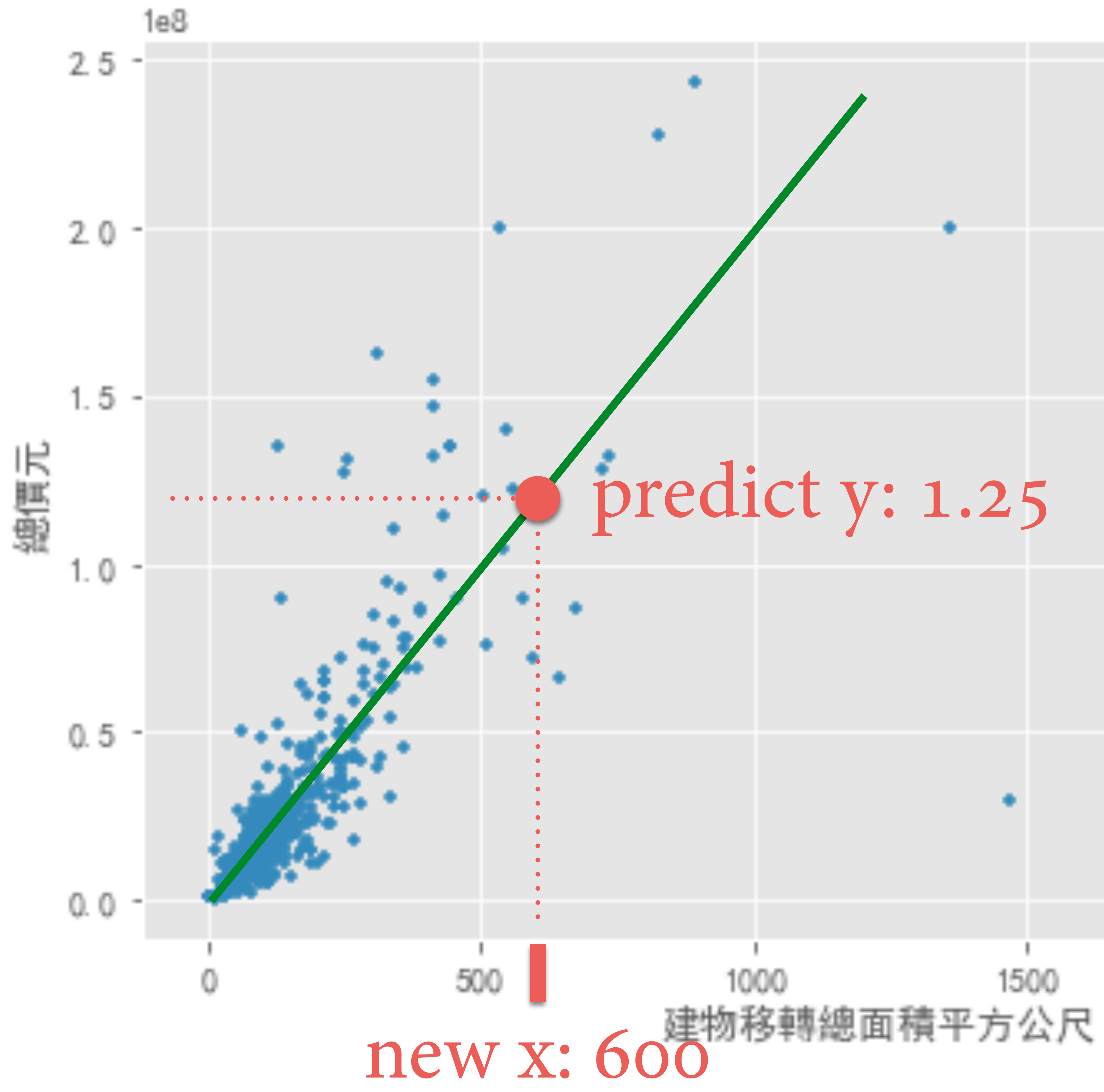
Python 機器學習與深度學習實作

監督式學習介紹

Burgeoning Course

Starting from a real case…

- 預測房價：從坪數(x_1)、幾房幾廳(x_2)、地址(x_3)...預測房價(y)

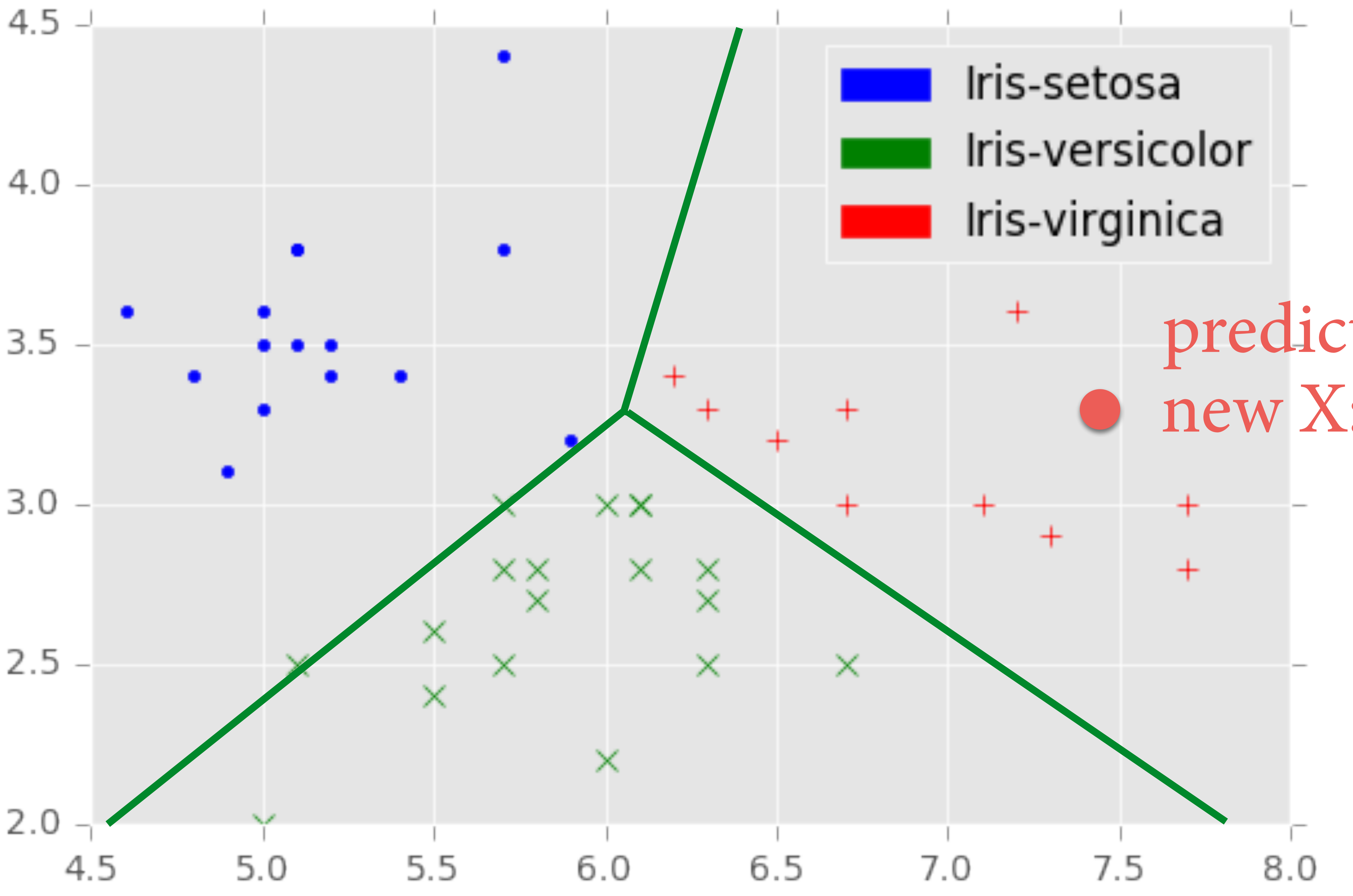


迴歸(Regression):

y為連續型數值

Starting from a real case…

- 預測品種：從花萼寬度、長度判斷鳶尾花品種



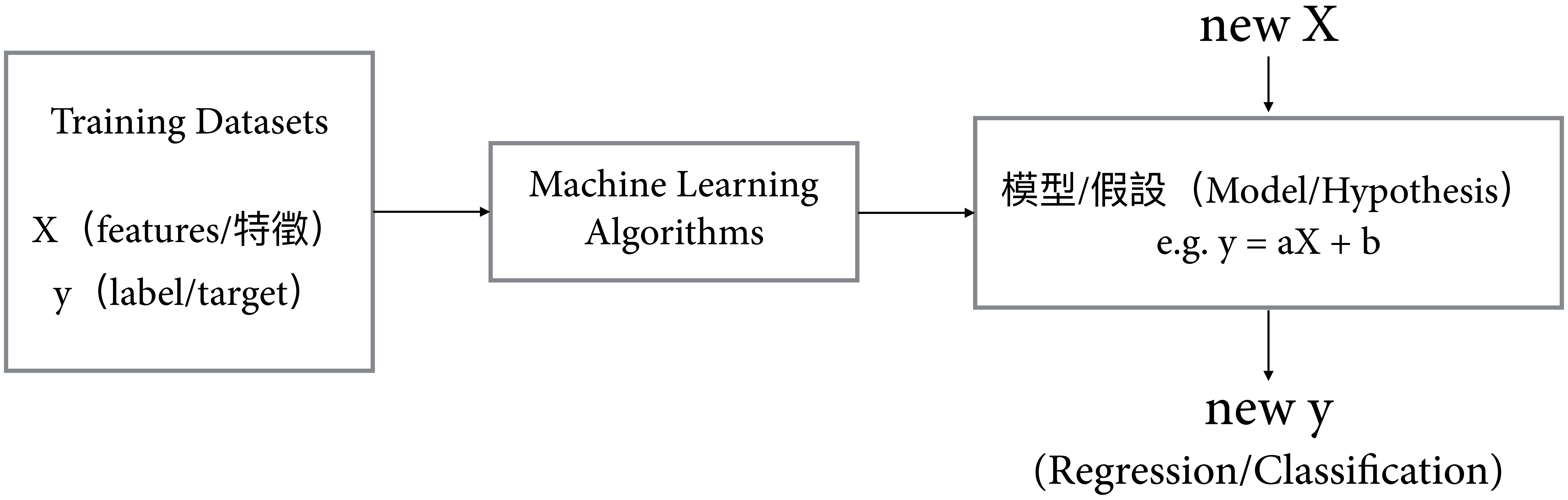
分類(Classification):

y為類別型數值

predict y: Iris-virginica (維吉尼亞鳶尾)

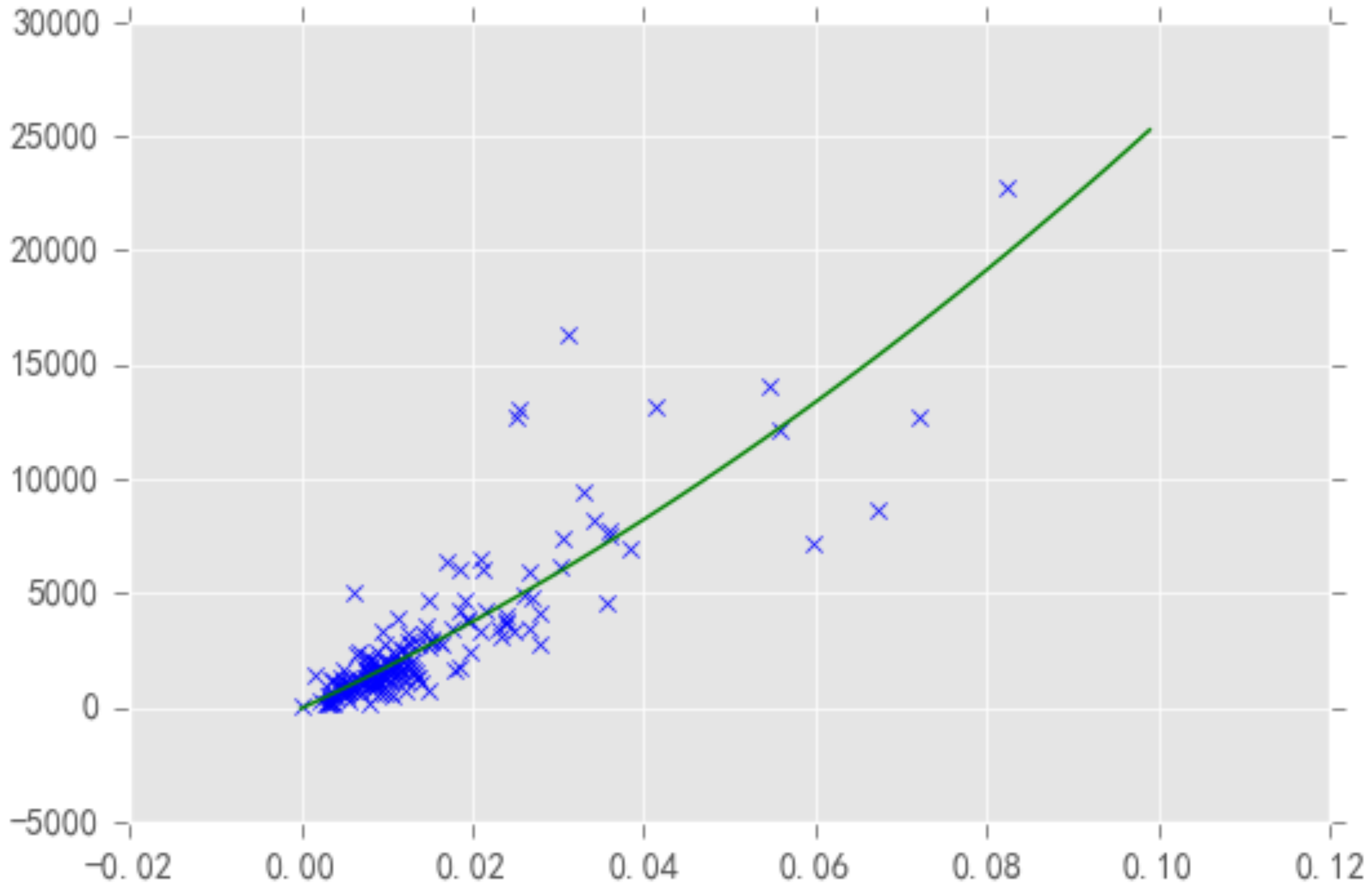
new X: (7.4, 3.3)

學習模式


Training Datasets
X（features/特徵）
y（label/target）
Machine Learning Algorithms
new X
模型/假設（Model/Hypothesis）
e.g. y = aX + b
new y
(Regression/Classification)

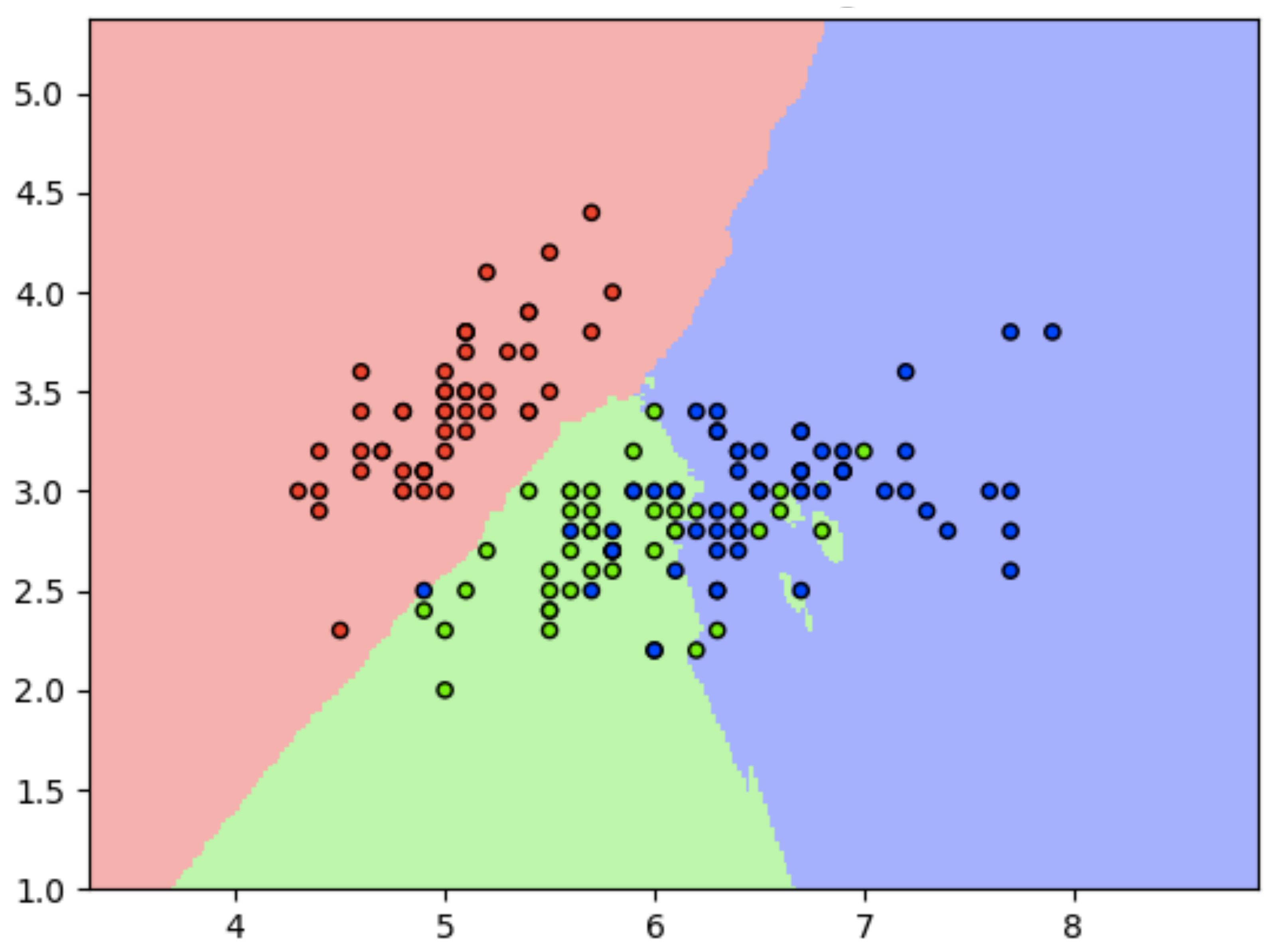
監督式學習

- 迴歸(Regression)

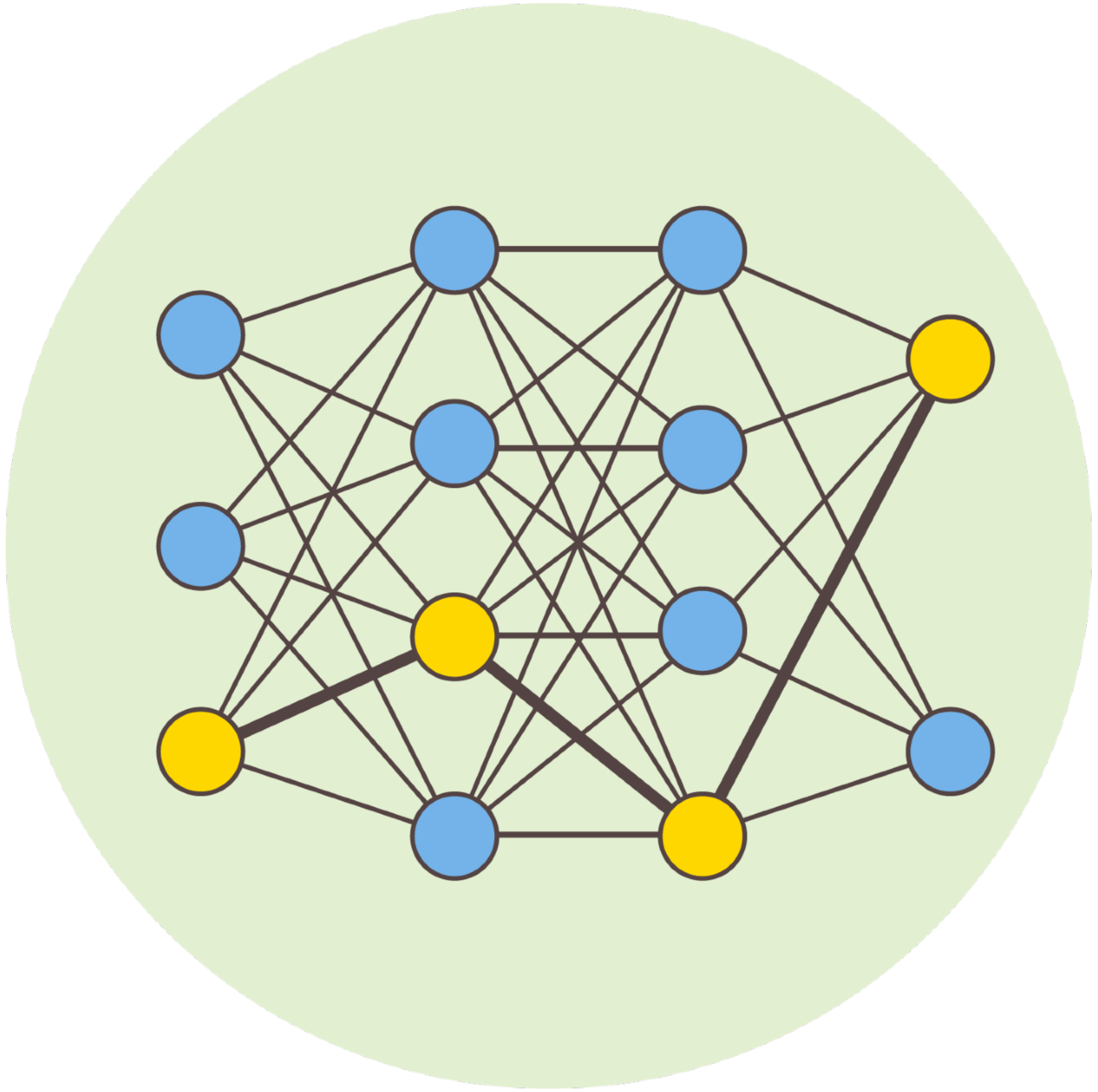


e.g. 房價、股價、成績

- 分類(Classification)



e.g. 是否為垃圾郵件、
是否罹患疾病、生物品種



Python 機器學習與深度學習實作

機器如何學習？
從線性迴歸開始

Burgeoning Course

Roadmap

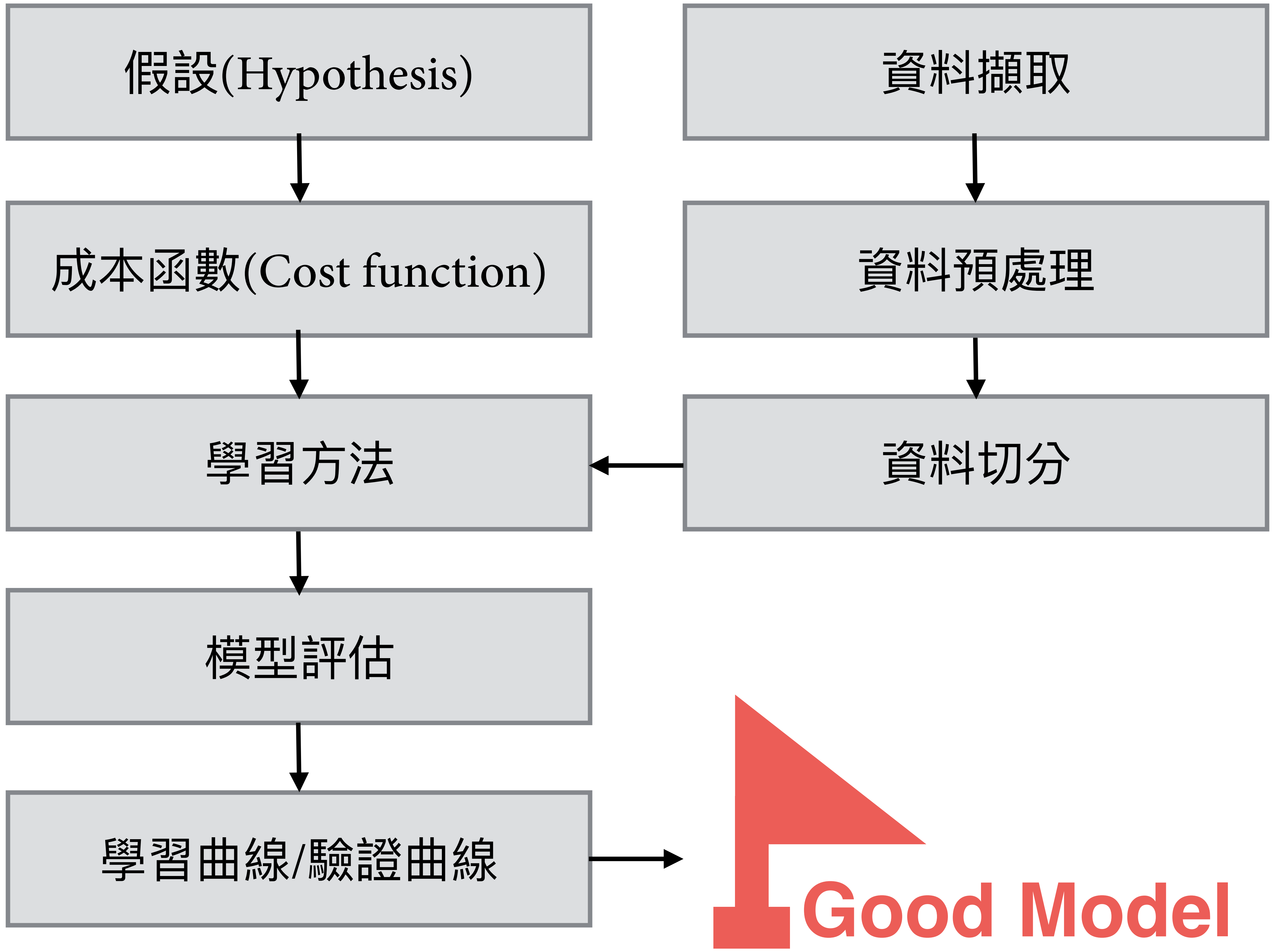
Q: 如何選擇假設函式？

Q: 如何最小化cost？

Q: 如何調節超參數？
Q: 如何選擇學習演算法？

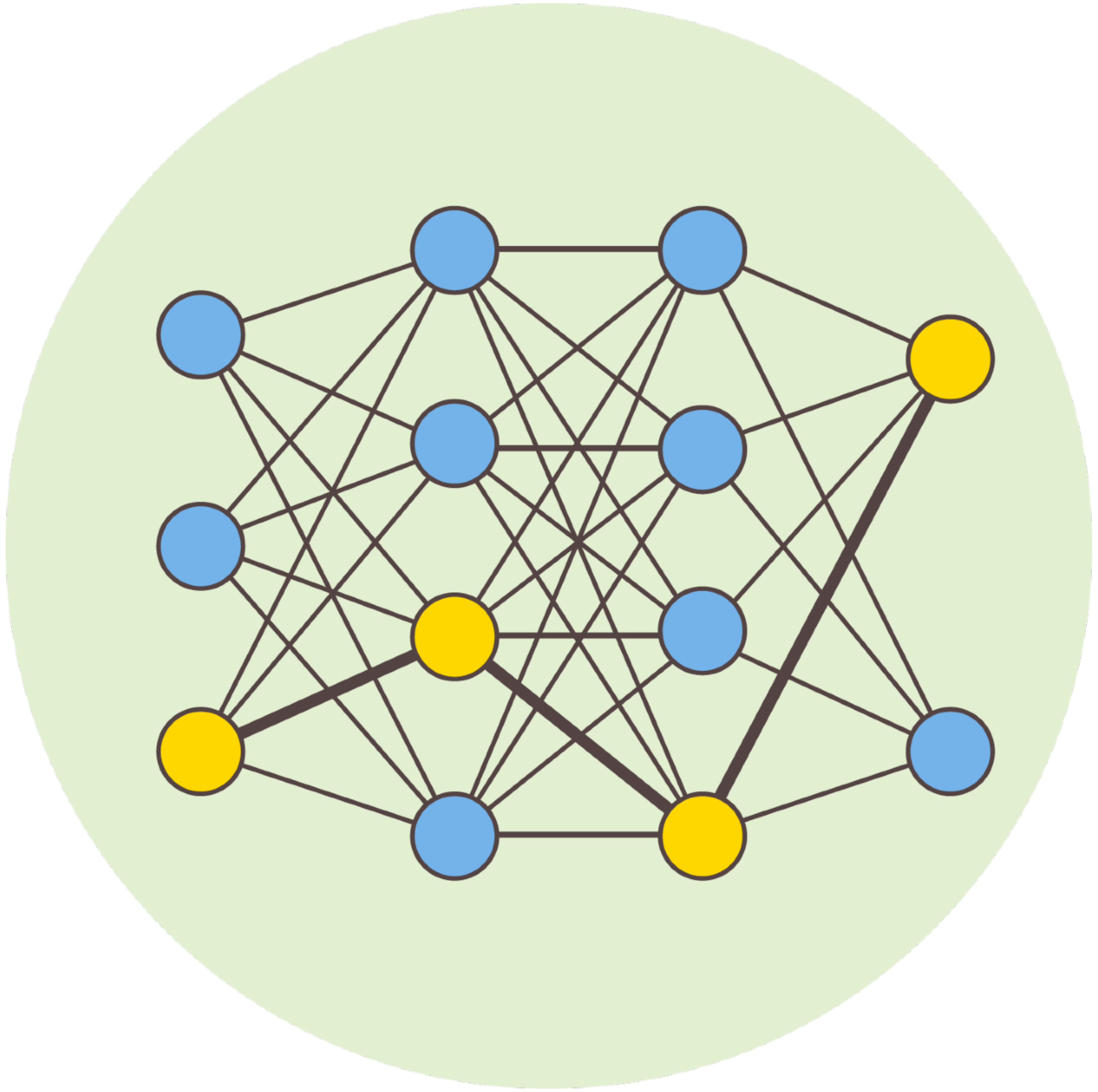
Q: 如何評估模型預測效果？

Q: 模型是否有過適現象？



Q: 如何處理不同種類資料？

Q: 如何切分資料？



Python 機器學習與深度學習實作

線性迴歸方程式

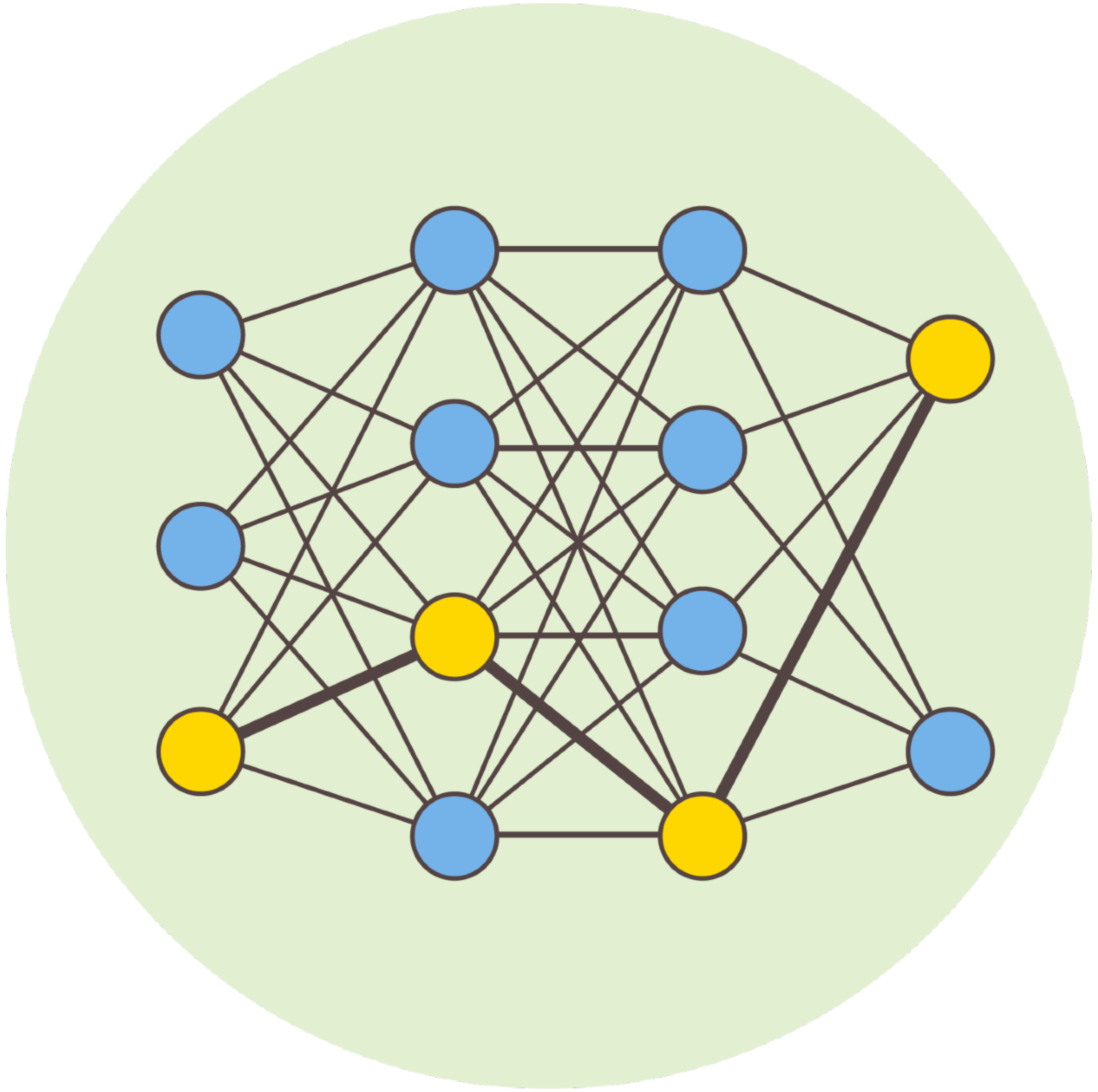
Burgeoning Course

線性迴歸（Linear Regression）

- 簡單線性迴歸（Simple Linear Regression）
  - $y = w^{(0)} + w^{(1)}x$ (e.g. $y = 3 + 2x$)
- 多變項線性迴歸（Multiple Linear Regression）
  - $y = w^{(0)} + w^{(1)}x_1 + \ldots + w^{(n)}x_n$ (e.g. $y = 1 + 2x_1 + 3x_2$)
- 模型假設(Hypothesis)：
  - $y = w^{(0)} + w^{(1)}x_1 + \ldots + w^{(n)}x_n$
  - $y = w^T x$

Notes

- w^T 是矩陣w $(w_1,w_2\ldots w_n)$的轉置矩陣(transpose)

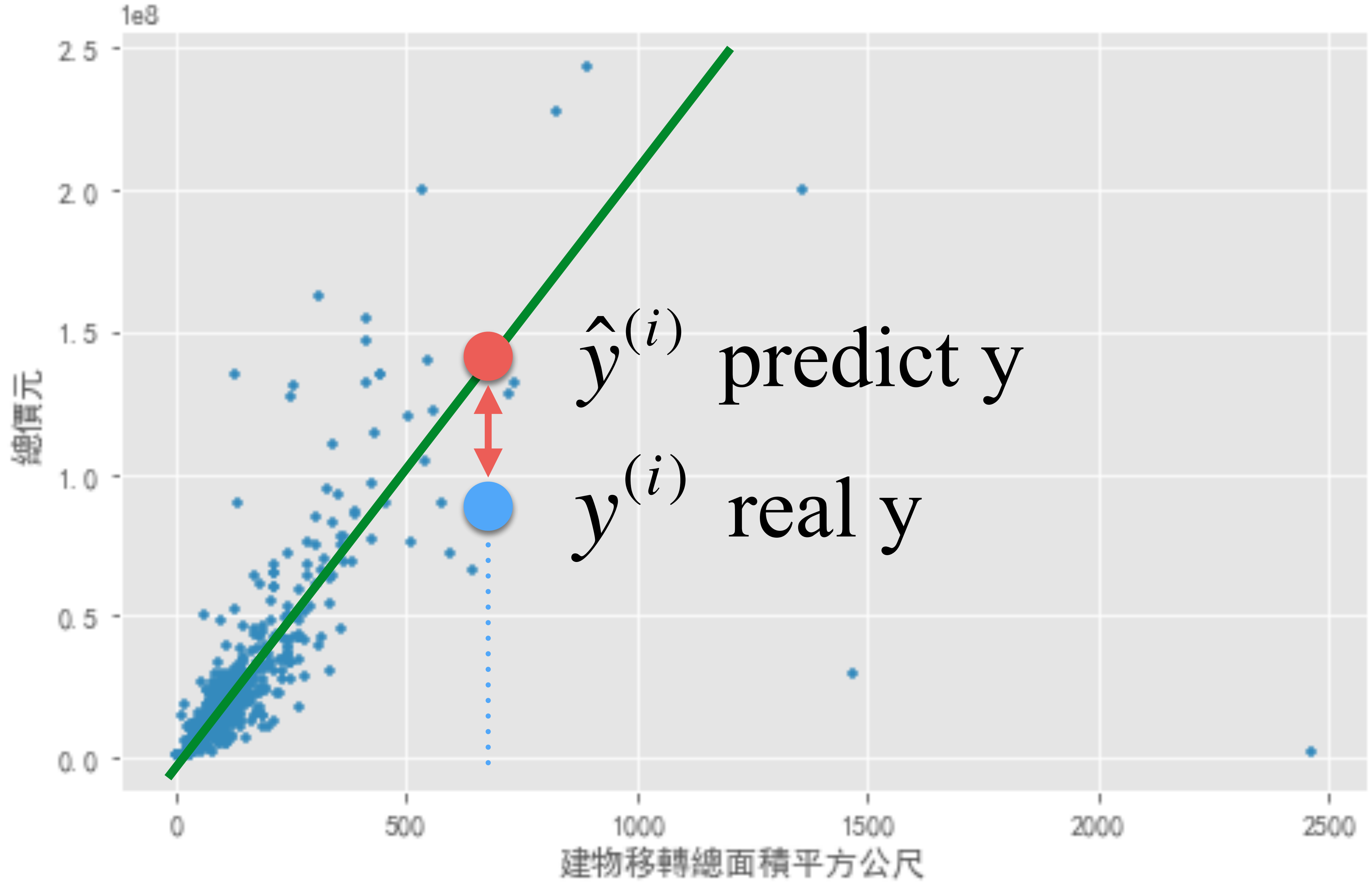


Python 機器學習與深度學習實作

線性迴歸與梯度下降

Cost Function

- 成本函數(Cost function)：均方誤差(Mean Squared Error, MSE)
- $J(w) = \dfrac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$



Notes

- $\hat{y}^{(i)}$ **為預測值，唸作 y hat**
- **m 為資料筆數**

How to minimize cost function

- 梯度下降(Gradient Descent)

- 正規方程(Normal Equation)：公式解

- $(X^T X)^{-1} X^T y$

Notes

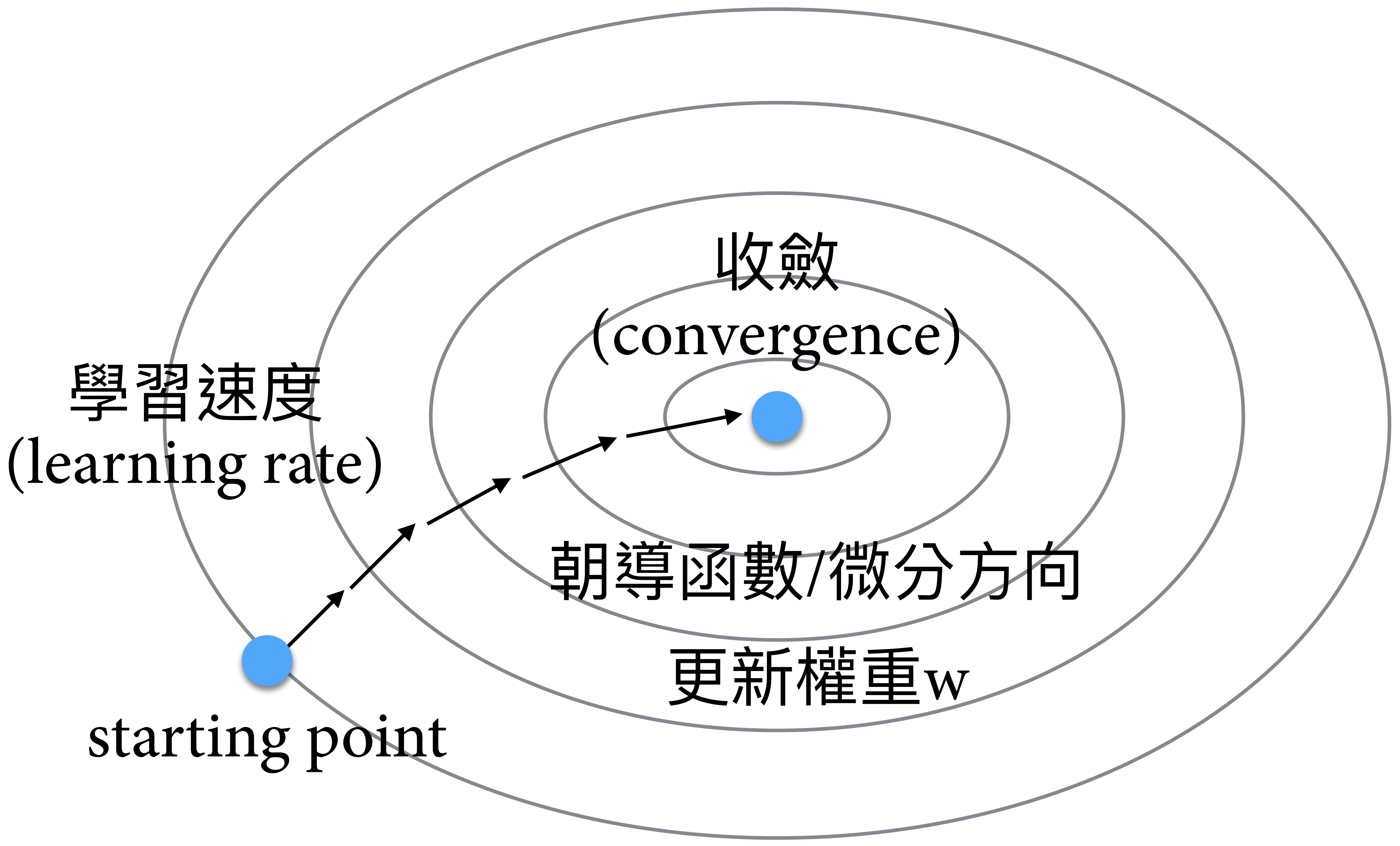
- X^T 是矩陣X的轉置矩陣(transpose)
- X^{-1} 是矩陣X的反矩陣(inverse)
- 梯度下降為機器學習重要解法，適合用於大數據建模
- scikit-learn 解線性迴歸使用正規方程

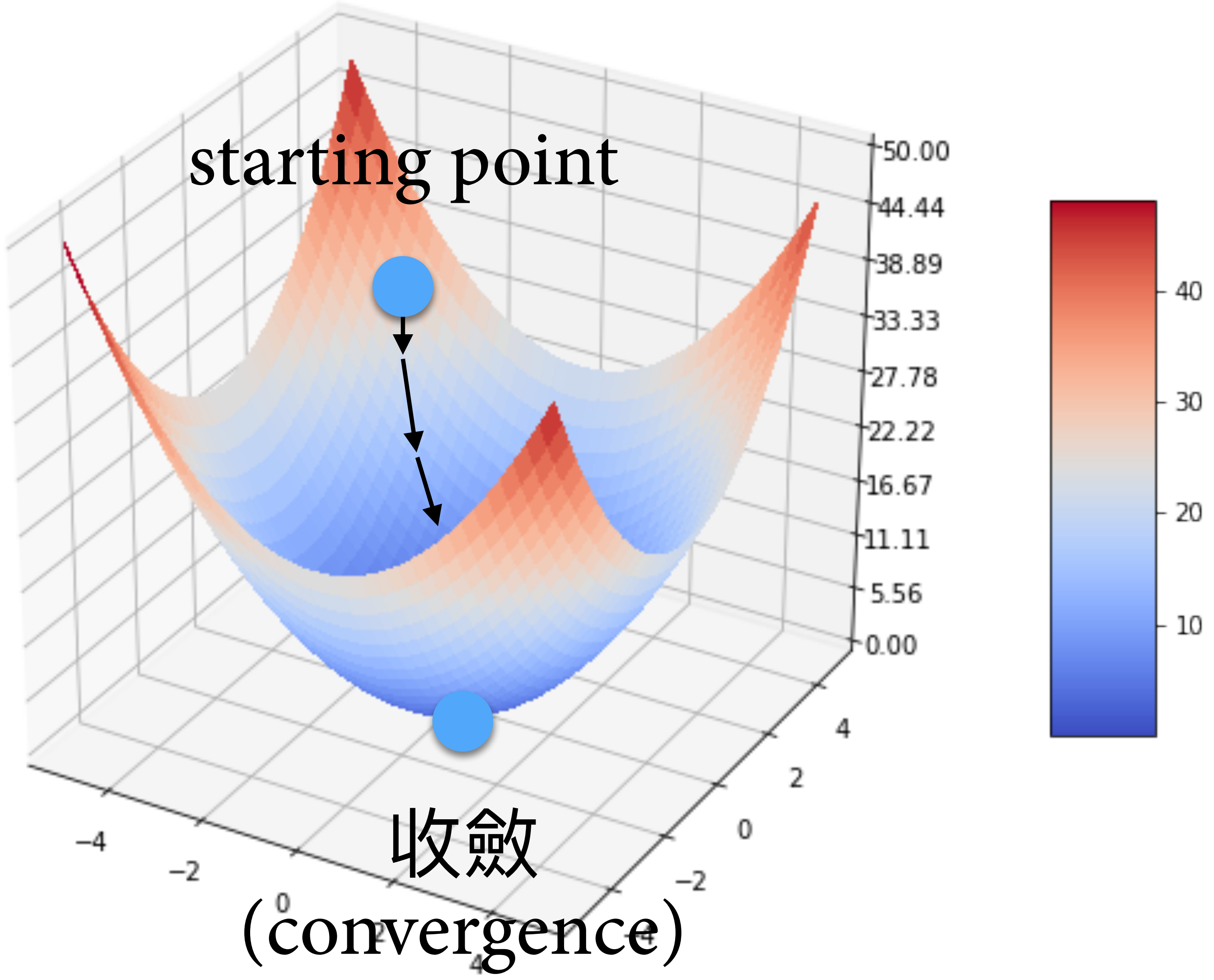
Gradient Descent

- $J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$
- 梯度下降法（Gradient Descent, GD）

Notes

- Cost Function of Linear Regression
- **區域最佳解**(local optimal) = **全域最佳解**(global optimal)





計算梯度

- $\mathrm{J(w)} = \dfrac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \dfrac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)2} - 2\hat{y}^{(i)}y^{(i)} + y^{(i)2})$
- $\hat{y}^{(i)} = w^{(0)} + w^{(1)}x^{(i)}$
- $$\begin{aligned}\frac{\partial}{\partial w_0}J(w) &= \frac{\partial}{\partial w_0}\frac{1}{2m}\sum_{i=1}^{m}(w^{(0)} + w^{(1)}x^{(i)})^2 - 2(w^{(0)} + w^{(1)}x^{(i)})y^{(i)} + y^{(i)2} \\ &= \frac{\partial}{\partial w_0}\frac{1}{2m}\sum_{i=1}^{m}w^{(0)2} + 2(w^{(0)}w^{(1)}x^{(i)}) + (w^{(1)}x^{(i)})^2 - (2w^{(0)} + 2w^{(1)}x^{(i)})y^{(i)} + y^{(i)2} \\ &= \frac{1}{m}\sum_{i=1}^{m}w^{(0)} + w^{(1)}x^{(i)} - y^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m}\hat{y}^{(i)} - y^{(i)}\end{aligned}$$

Notes

- $(a-b)^2 = a^2 - 2ab + b^2$

計算梯度

- $$\begin{aligned}\frac{\partial}{\partial w_1}J(w) &= \frac{\partial}{\partial w_1}\frac{1}{2m}\sum_{i=1}^{m}(w^{(0)}+w^{(1)}x^{(i)})^2-2(w^{(0)}+w^{(1)}x^{(i)})y^{(i)}+y^{(i)2} \\ &= \frac{\partial}{\partial w_1}\frac{1}{2m}\sum_{i=1}^{m}w^{(0)2}+2(w^{(0)}w^{(1)}x^{(i)})+(w^{(1)}x^{(i)})^2-2w^{(0)}-2(w^{(1)}x^{(i)})y^{(i)}+y^{(i)2} \\ &= \frac{1}{m}\sum_{i=1}^{m}w^{(0)}x^{(i)}+x^{(i)2}w^{(1)}-x^{(i)}y^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m}x^{(i)}(w^{(0)}+x^{(1)}w^{(1)}-y^{(i)}) \\ &= \frac{1}{m}\sum_{i=1}^{m}x^{(i)}(\hat{y}^{(i)}-y^{(i)})\end{aligned}$$

計算權重與超參數

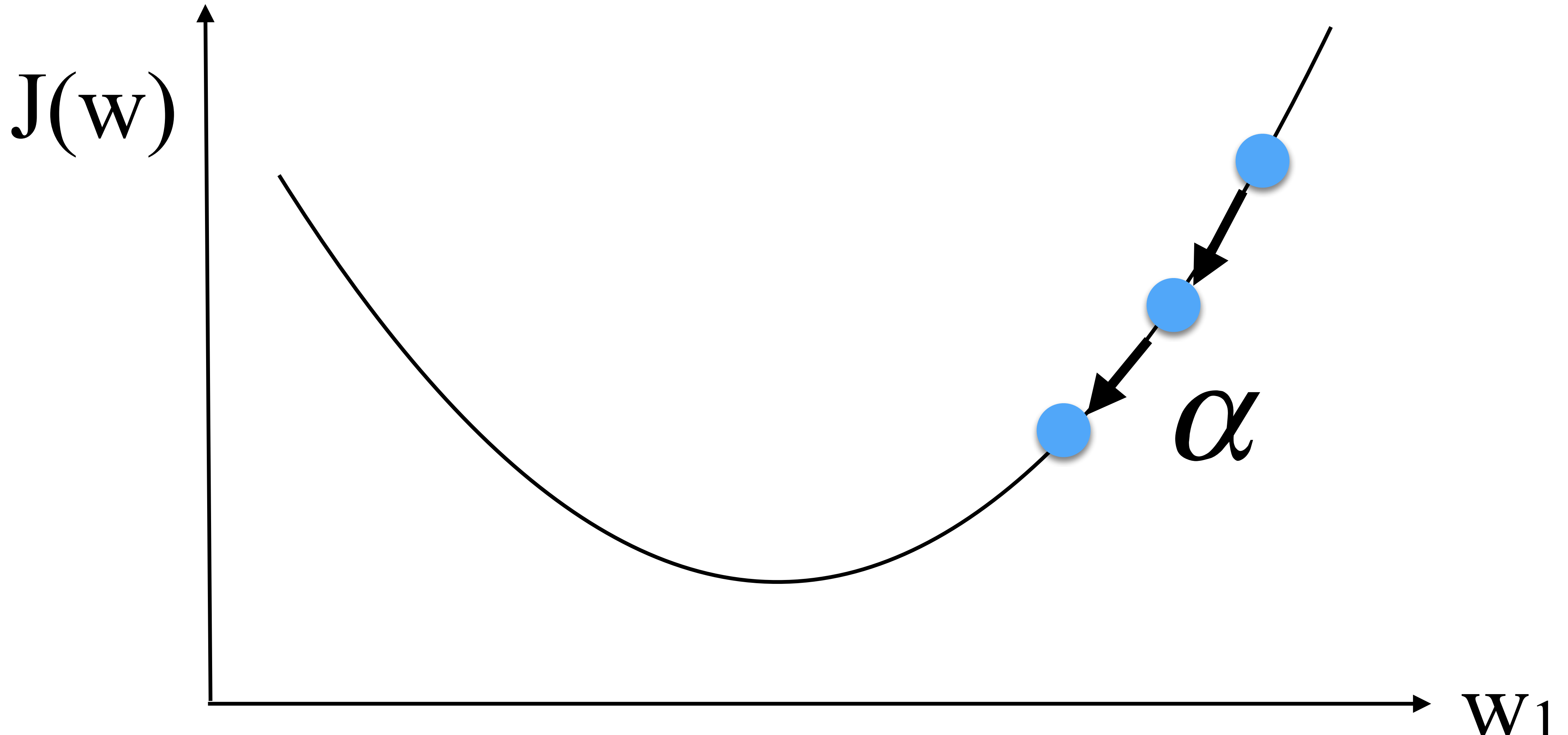
repeat {

$$w^{(0)} = w^{(0)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$$

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

$$\vdots$$

}



$$\hat{y}^{(i)} = w^{(1)} x^{(i)}$$

- 超參數(Hyperparameter)：
  - 步數(step)
  - 學習速率 α

多變項線性迴歸

- 多變項線性迴歸計算權重

repeat {

$$w^{(0)} = w^{(0)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$$

$$w^{(1)} = w^{(1)} - \alpha \frac{1}{m} \sum_{i=1}^{m} x_1^{(i)} (\hat{y}^{(i)} - y^{(i)})$$

$$w^{(2)} = w^{(2)} - \alpha \frac{1}{m} \sum_{i=1}^{m} x_2^{(i)} (\hat{y}^{(i)} - y^{(i)})$$

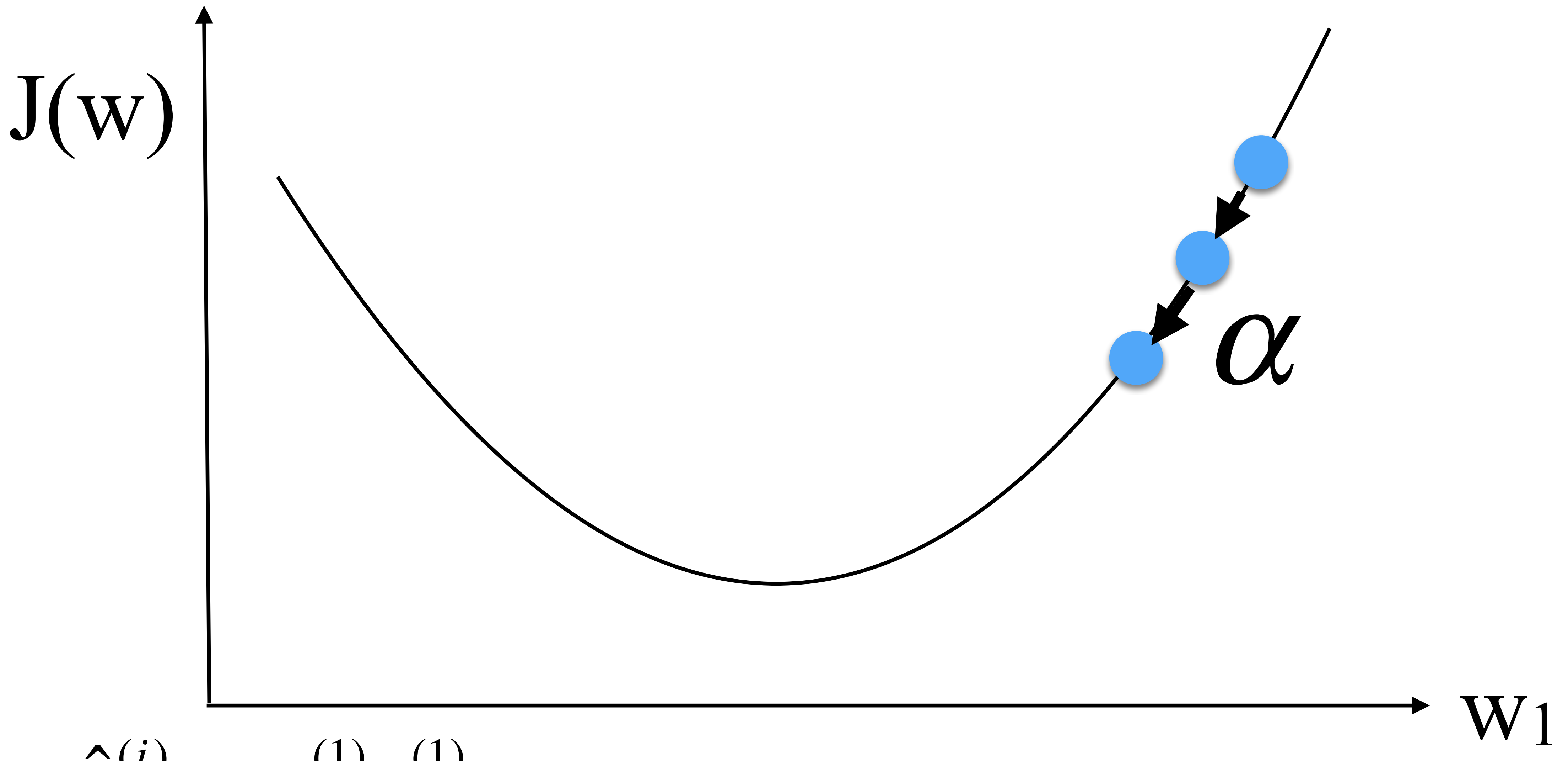
$$w^{(3)} = w^{(3)} - \alpha \frac{1}{m} \sum_{i=1}^{m} x_3^{(i)} (\hat{y}^{(i)} - y^{(i)})$$

$$\vdots$$

}

學習速度

- 如何選擇學習速度？
  - 若學習速度過慢 => 收斂速度慢



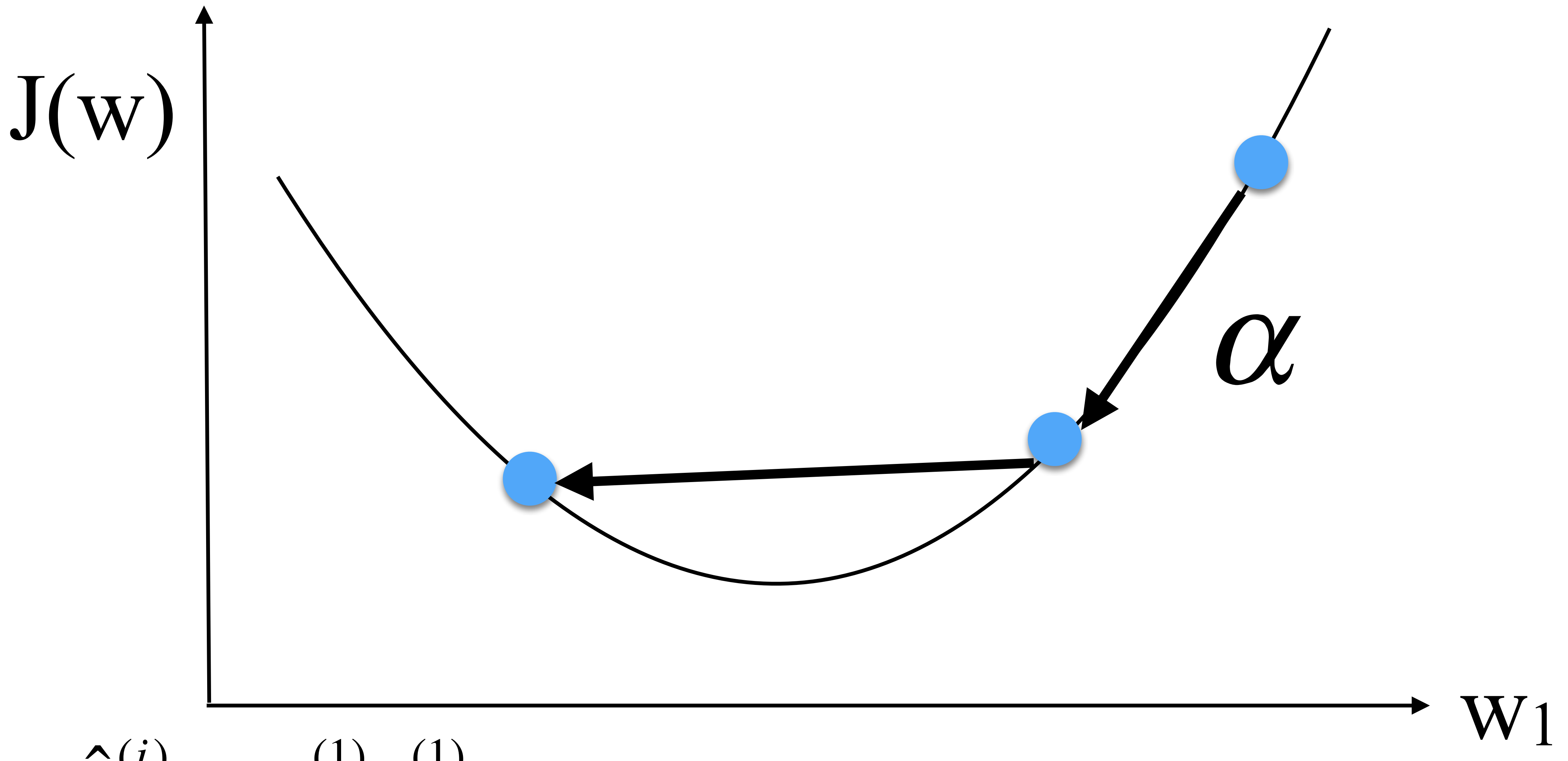
$\hat{y}^{(i)} = w^{(1)}x^{(1)}$

學習速度

- 如何選擇學習速度？
  - 若學習速度過快 => 無法收斂(過頭)

Notes

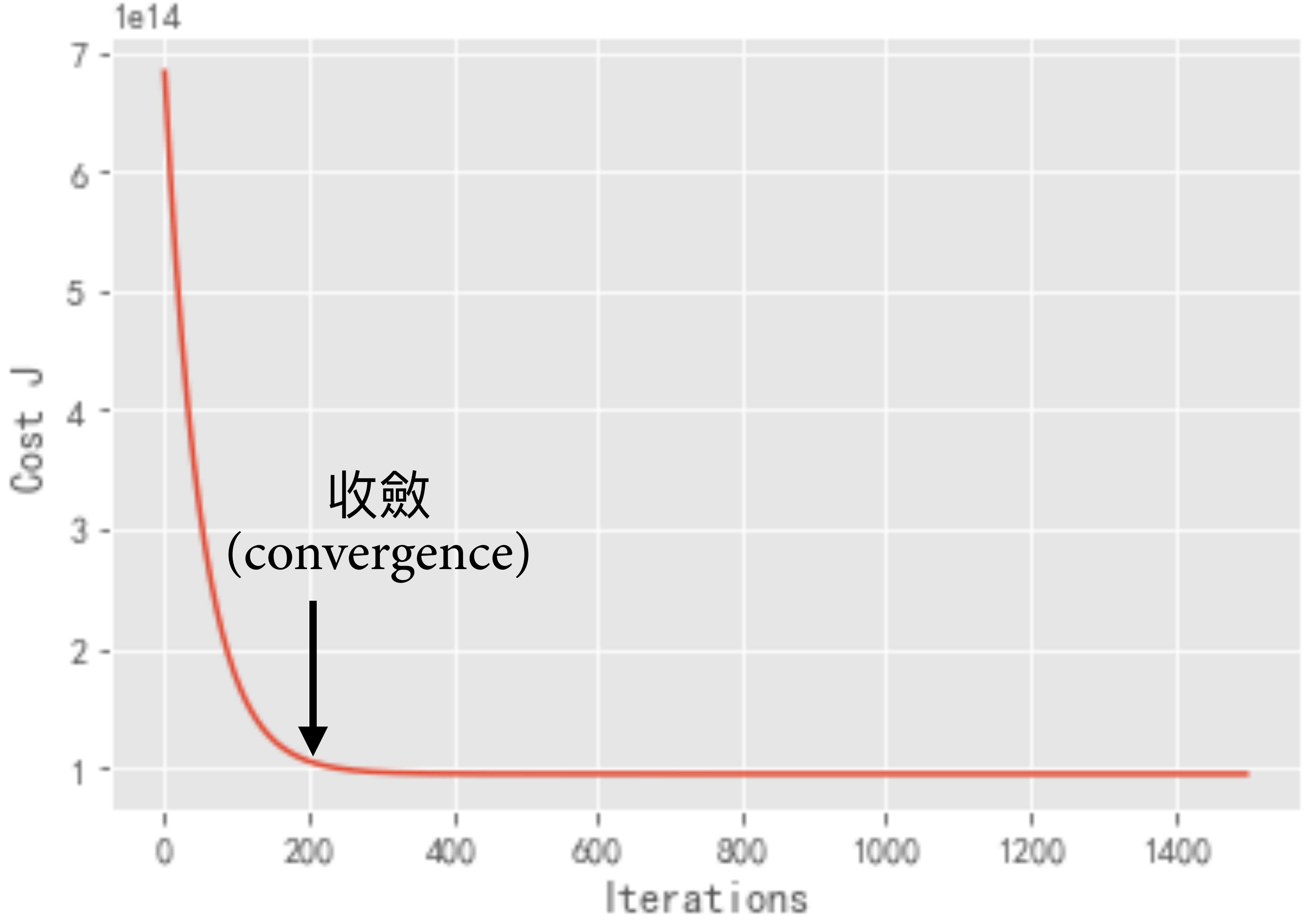
- sklearn.linear_model.SGDRegressor
- 預設α初始值(etao)為0.01，α隨時間縮小為: 1.0 / (α * (t + t_0))
- 後續Deep Learning會再詳細說明學習速度的選擇演算法



$$\hat{y}^{(i)} = w^{(1)}x^{(1)}$$

步數

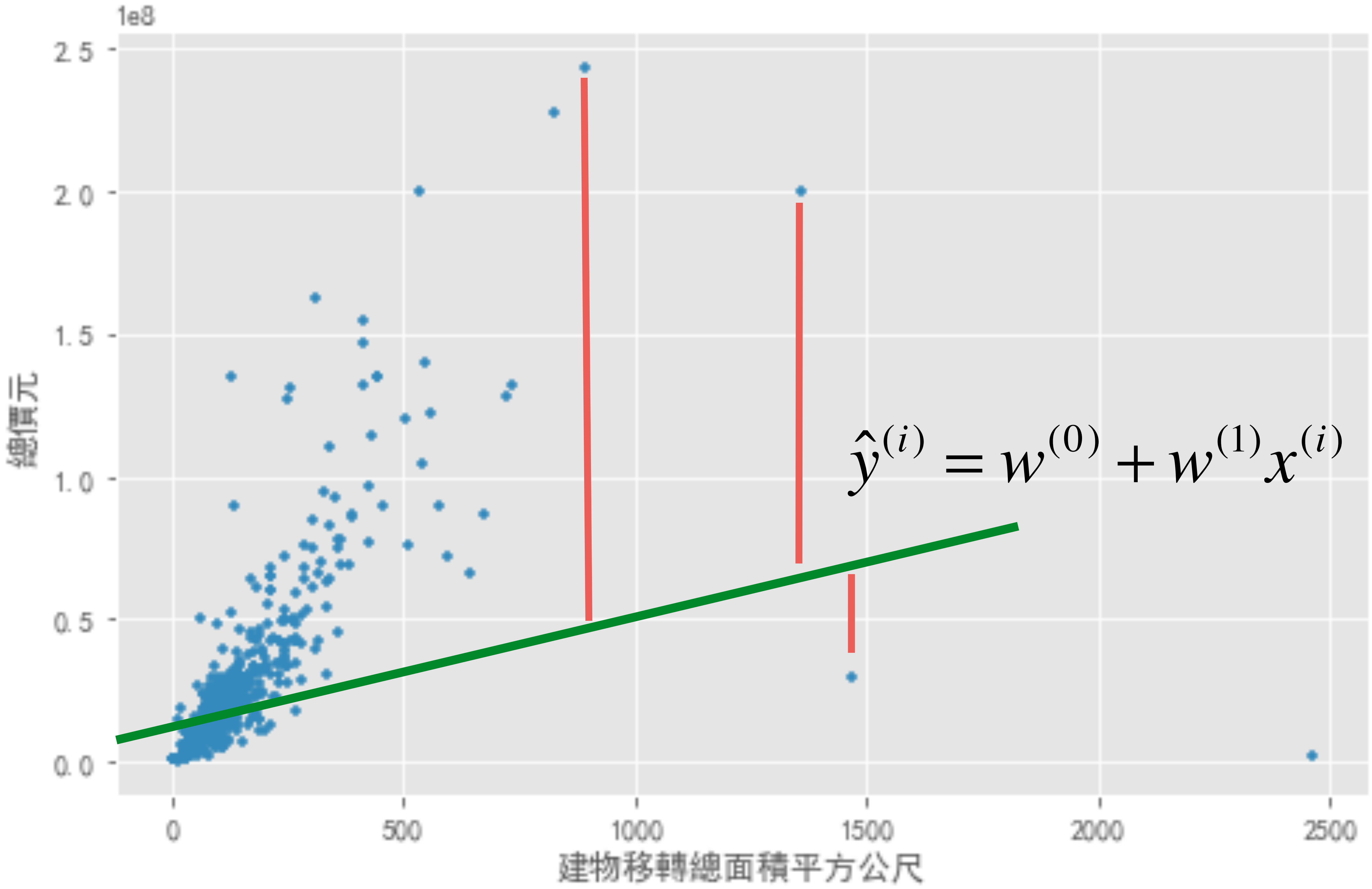
- 如何選擇步數？(iterations)

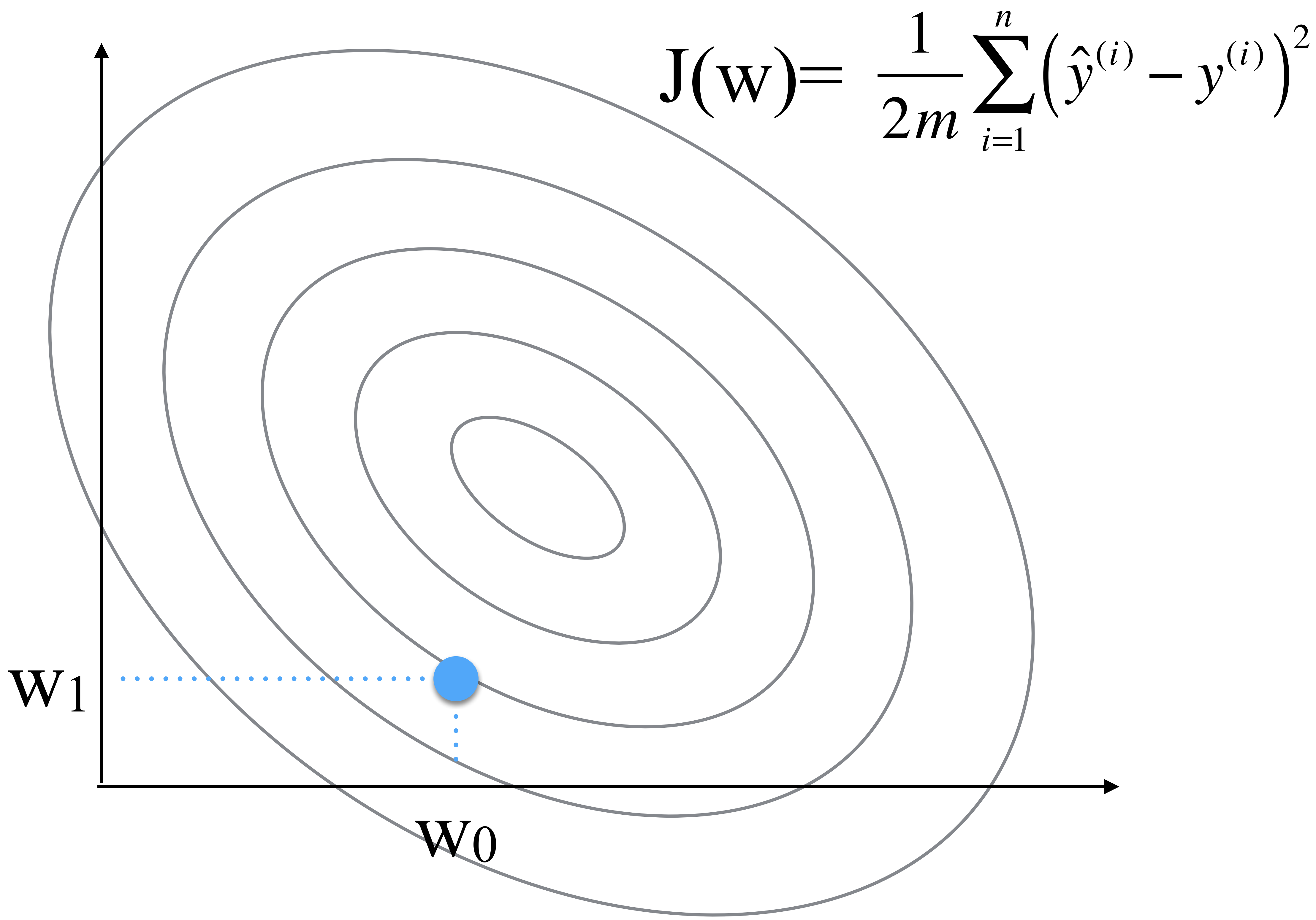


Notes

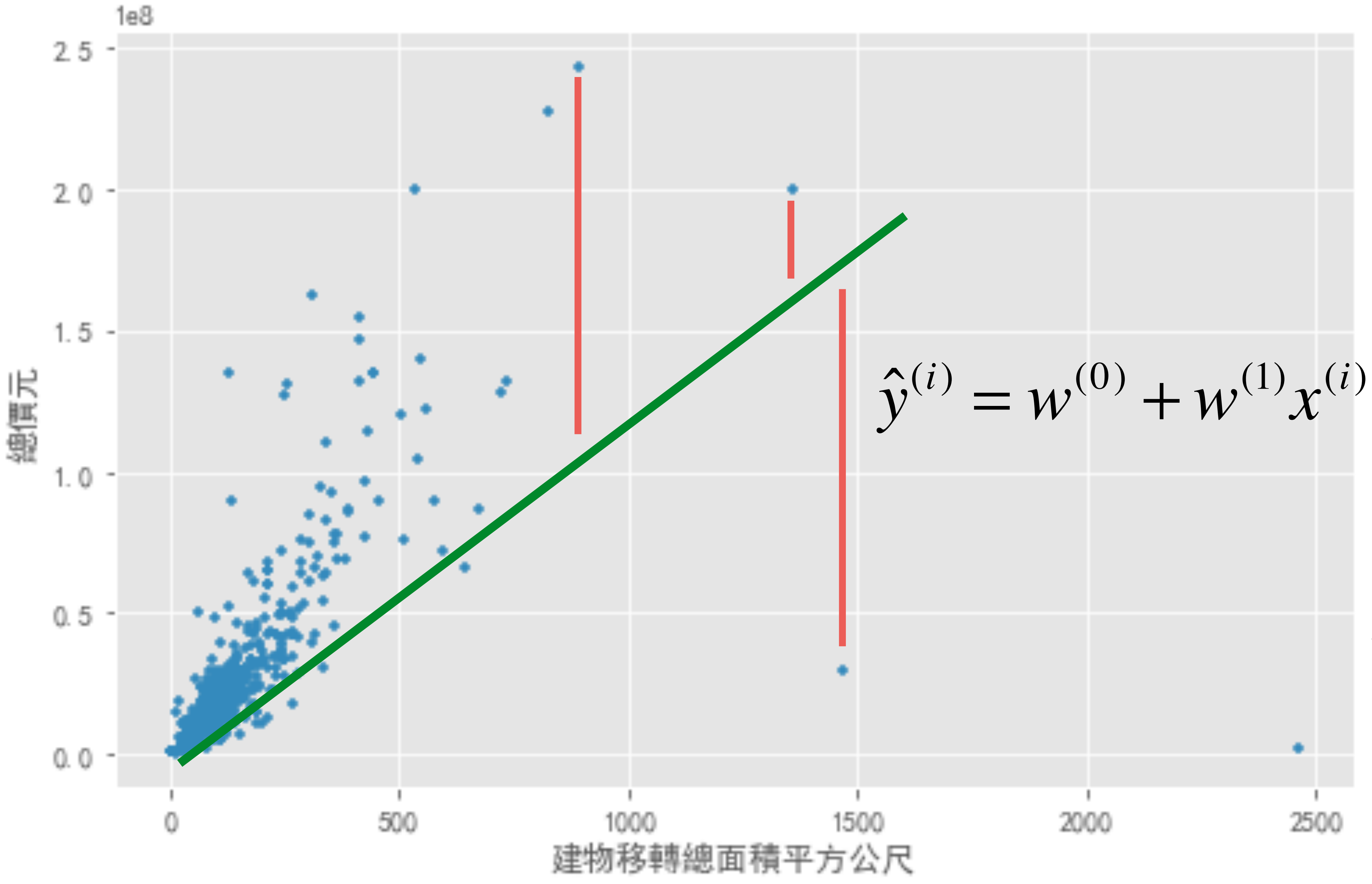
- sklearn.linear_model.SGDRegressor
- **預設步數**(max_iter) **為**1000

Gradient Descent Example



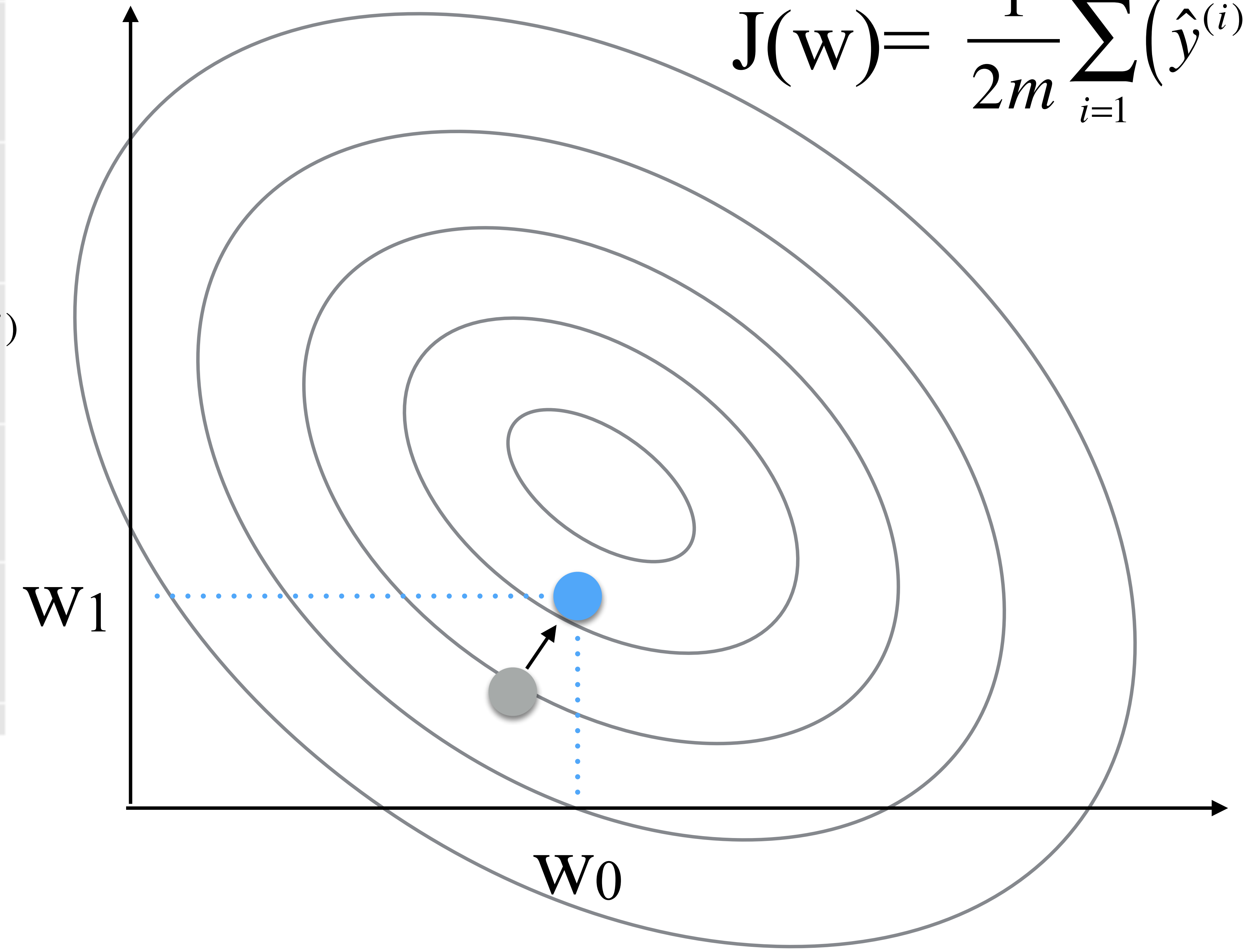


Gradient Descent Example

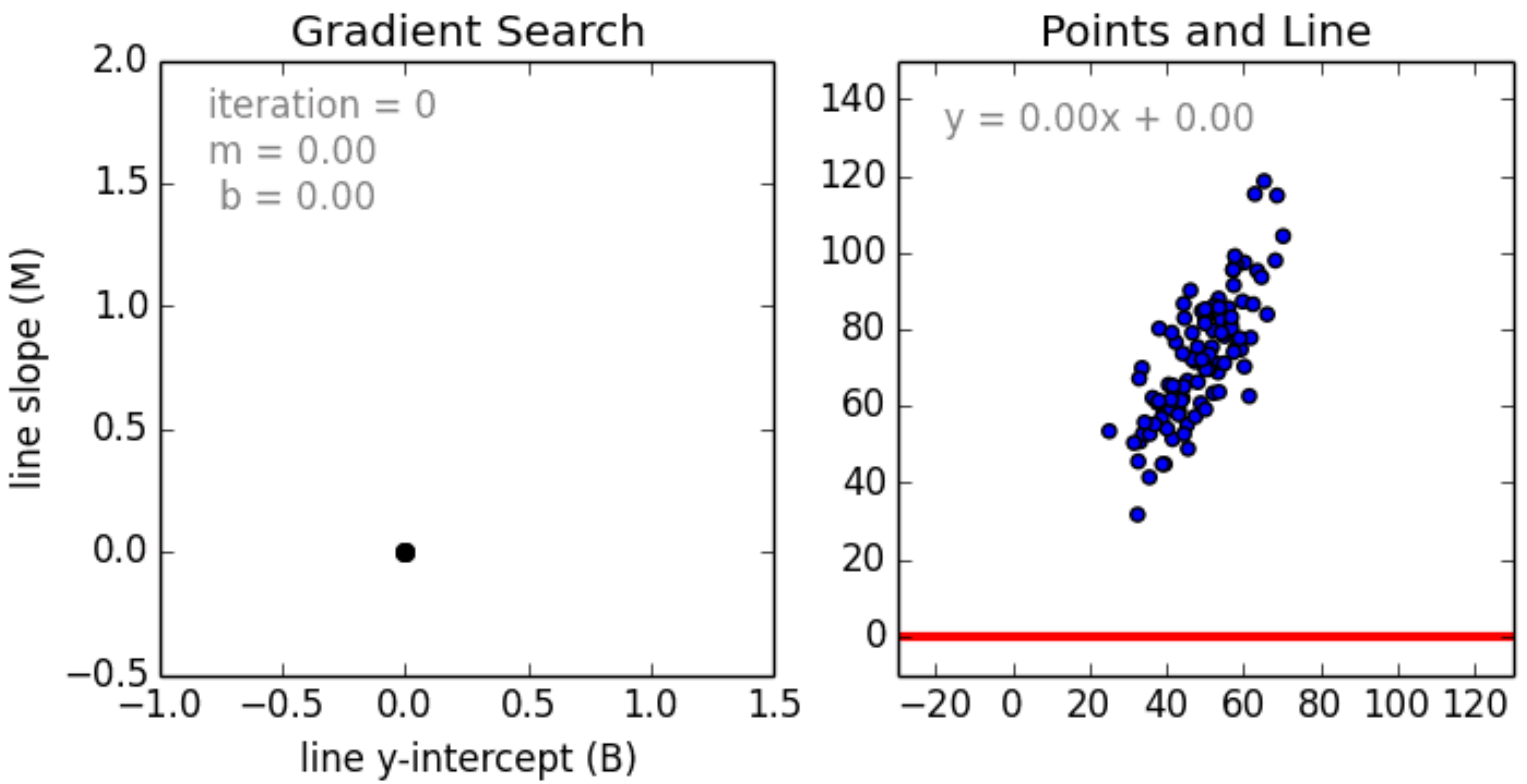


$$\hat{y}^{(i)} = w^{(0)} + w^{(1)}x^{(i)}$$

$$J(w) = \frac{1}{2m}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

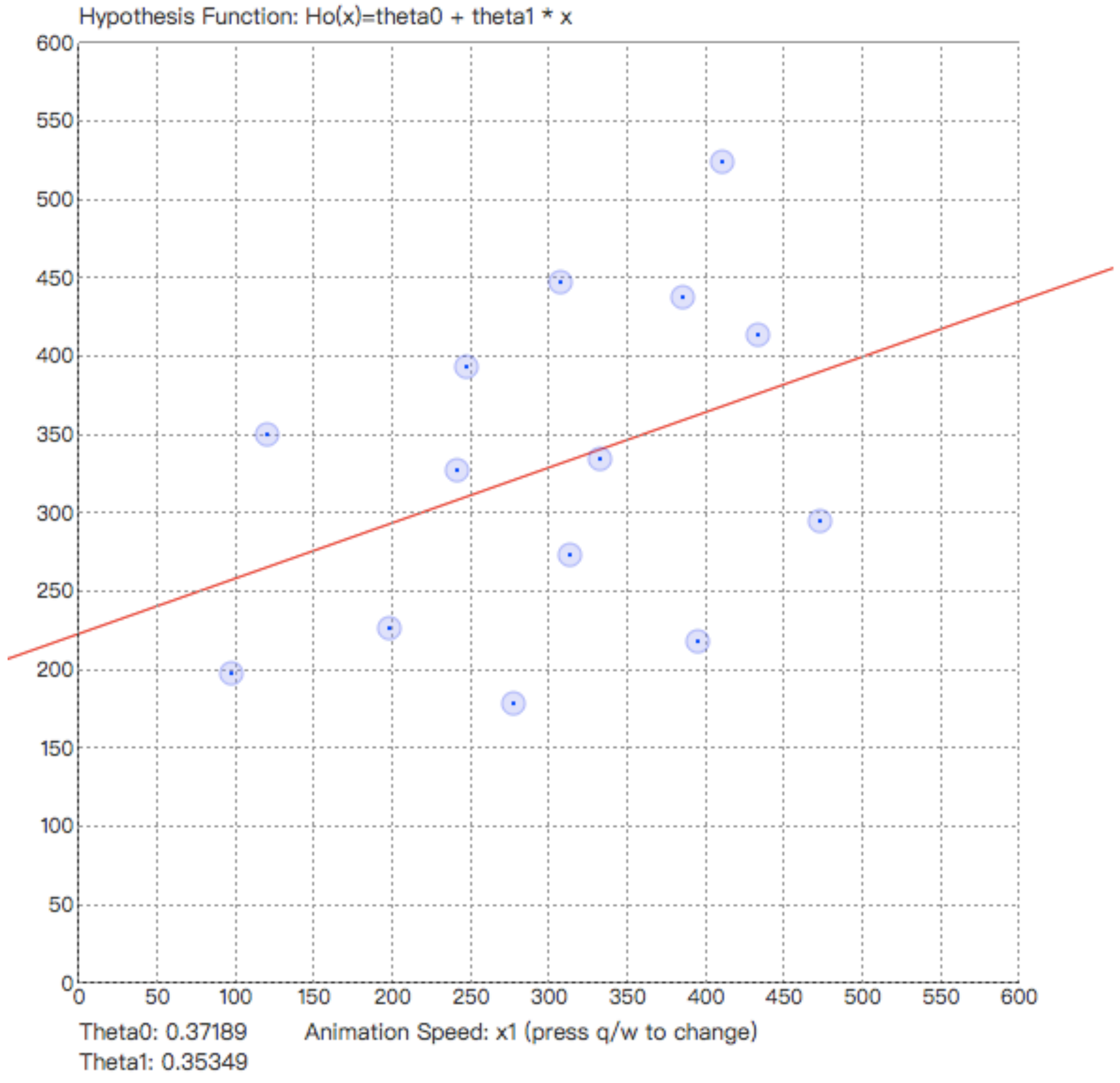


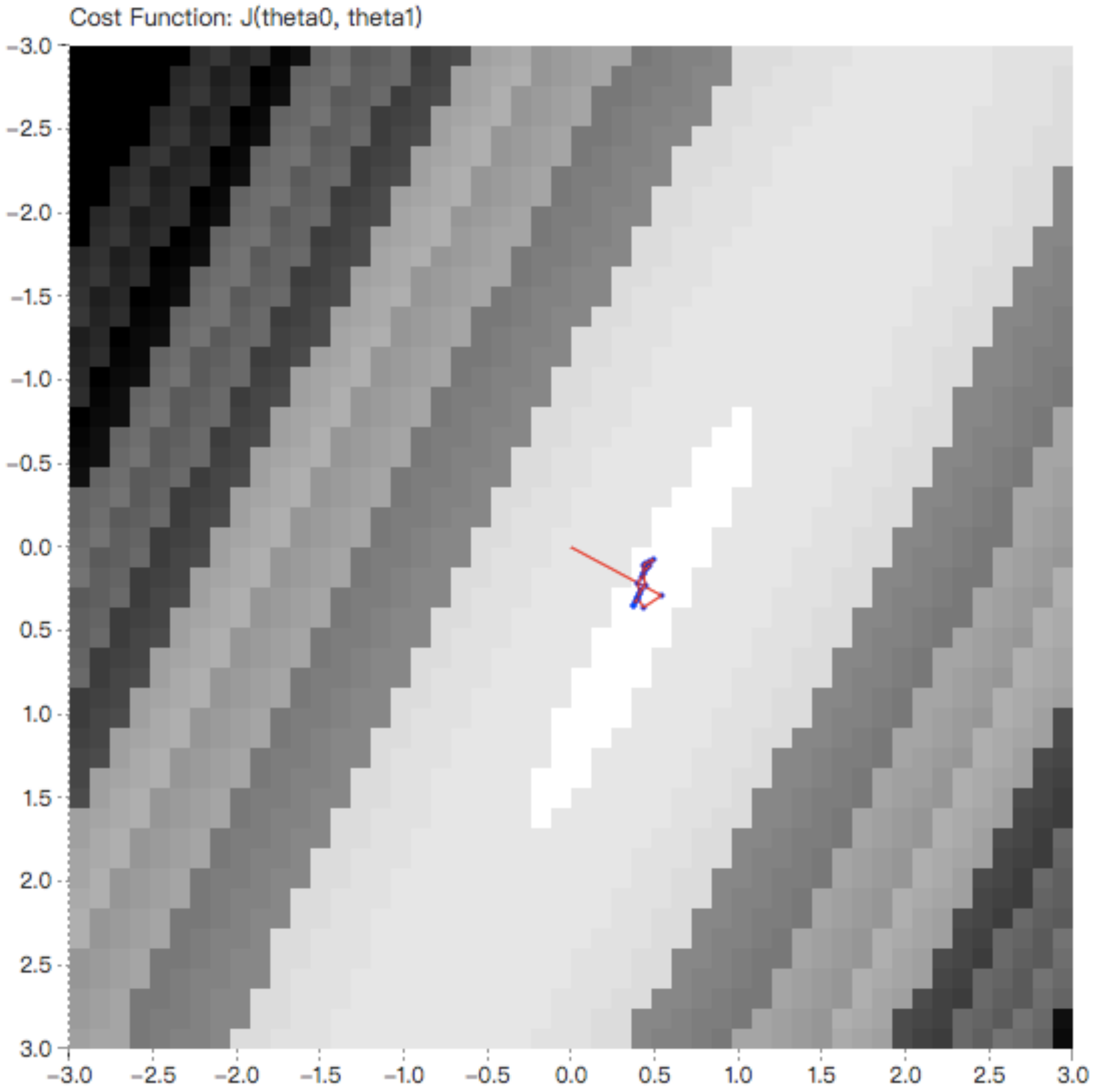
Gradient Descent 動態變化圖



(Source: https://github.com/mattnedrich/GradientDescentExample)

Interactive demonstration of the Gradient Descent algorithm (補充)



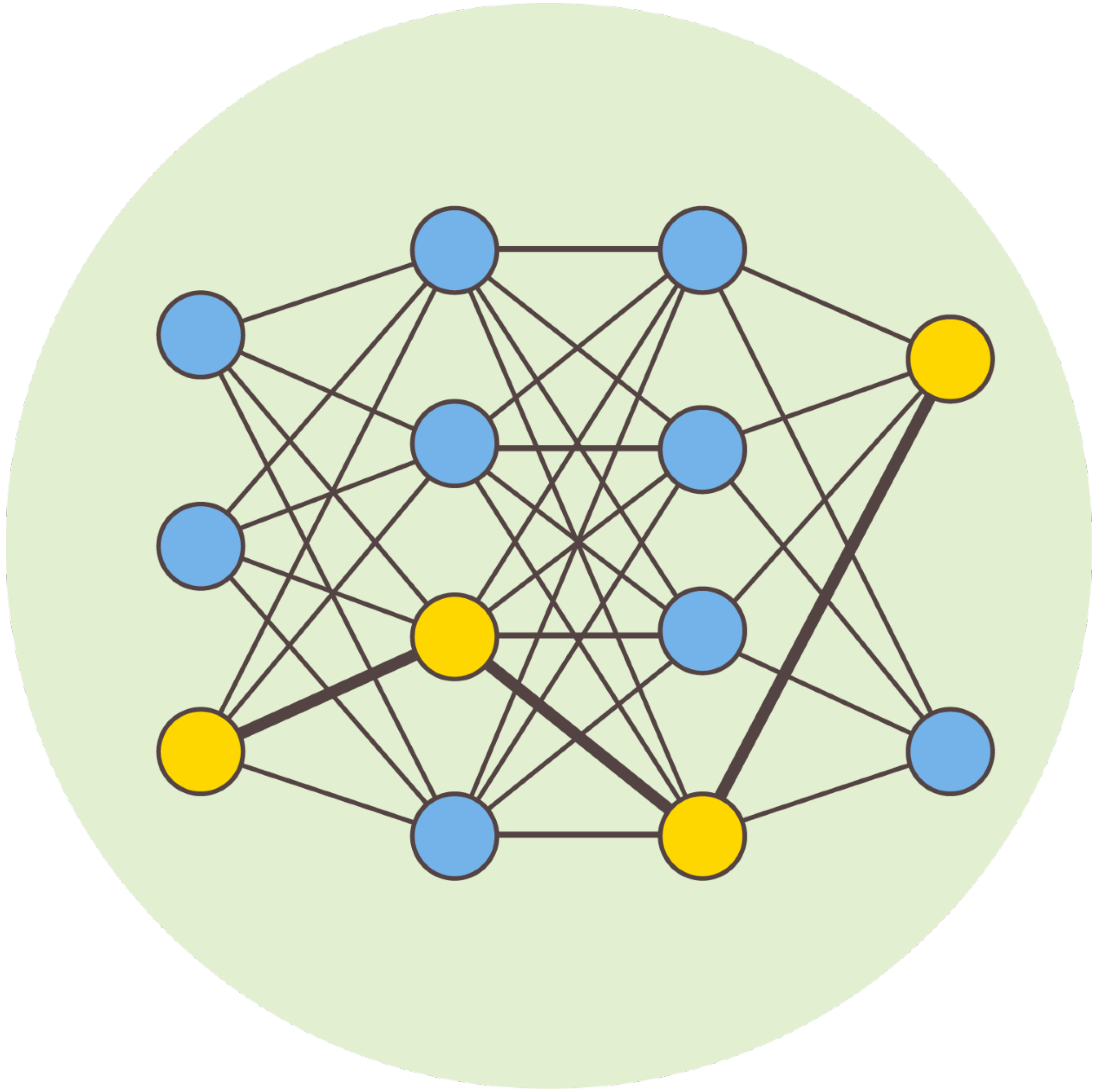


梯度下降法種類

- **批次梯度下降**(Batch Gradient Descent, BGD)：每步使用全部的資料計算，難以用於大數據建模。

- **隨機梯度下降**(Stochastic Gradient Descent, SGD)：每步隨機挑選一個樣本計算(運算速度快，但不一定能往正確方向前進，適用於線上學習)。

- **小批梯度下降**(Mini-batch Gradient Descent, MBGD)：介於BGD和SGD之間，每次隨機選擇m筆資料計算。

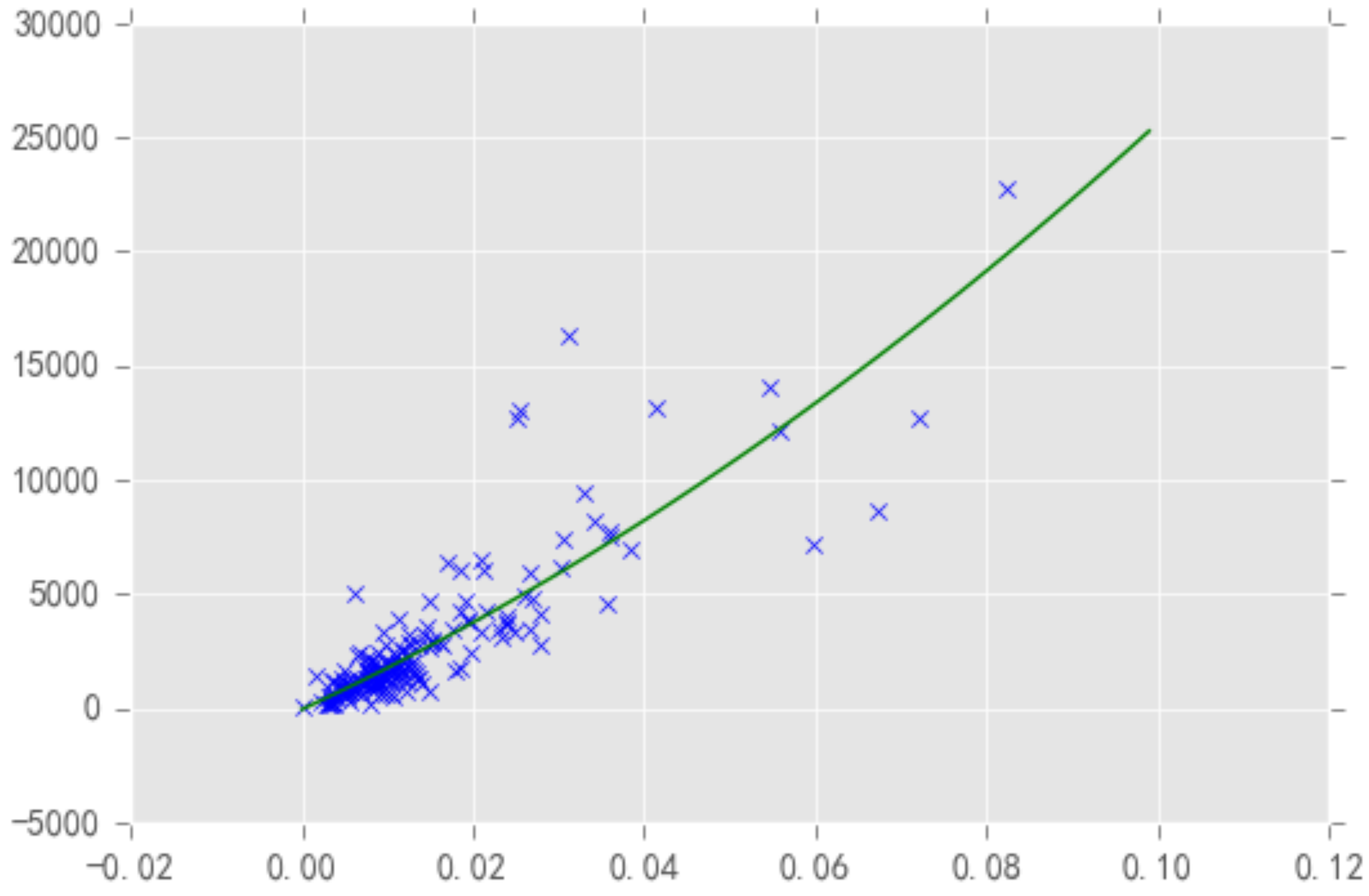


Python 機器學習與深度學習實作

非線性迴歸

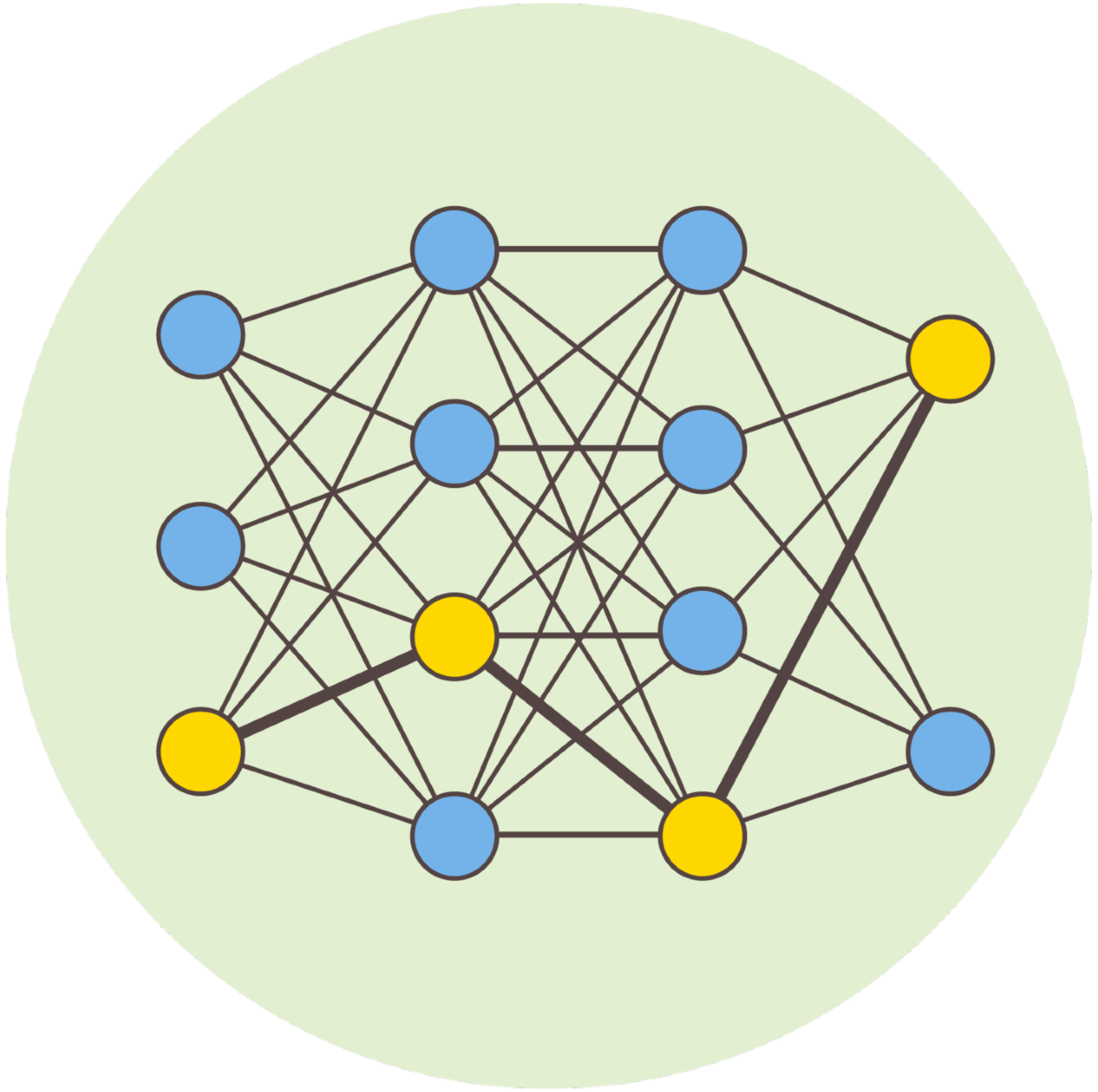
非線性迴歸 (Nonlinear Regression)

- 多項式迴歸(Polynomial Regression)
- $y = w^{(0)} + w^{(1)}x_1 + w^{(2)}x_2^2 + \ldots$ (e.g. $y = 3 + 2x_1 + 3x_2^2$)



Notes

- 一樣可用正規方程解



Python 機器學習與深度學習實作

模型評估

評估模型準確度

- 均方誤差 (Mean Squared Error, MSE)

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

- R平方 (R Square)

$$SSE=\sum_{i=1}^{m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2$$

$$SST=\sum_{i=1}^{m}\left(y^{(i)}-\bar{y}\right)^2$$

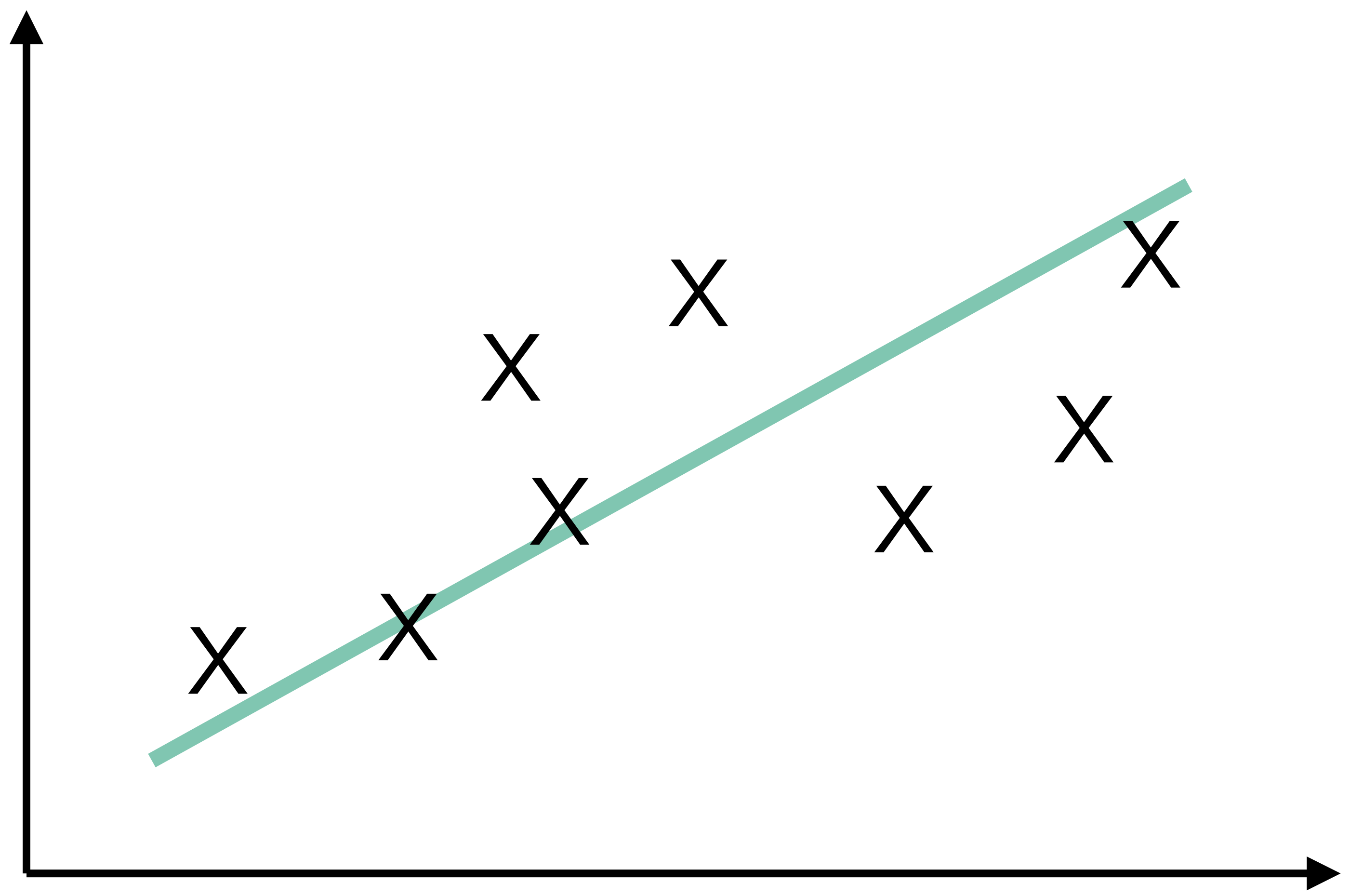
$$R^2=1-\frac{SSE}{SST}$$

Notes

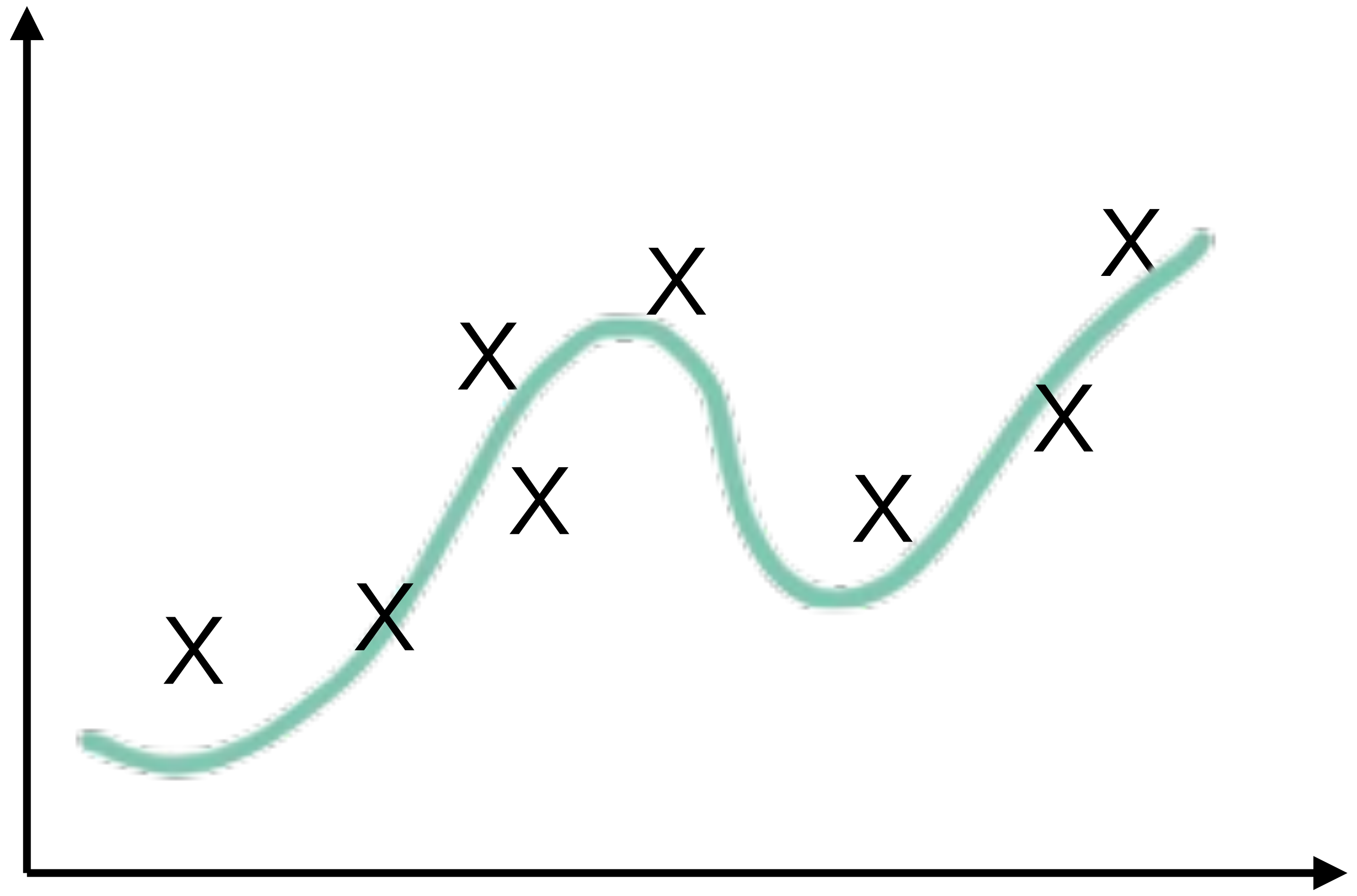
- SSE (Sum of Squared Error): **預測值和實際值的誤差平方和，即迴歸差異**
- SST (Sum of Squared Total): **實際值與平均值的誤差平方和，即內部差異**
- **後續課程會再教大家如何評估分類模型準確度**

Underfitting/Overfitting

- **欠擬合(Underfitting)**



- **過擬合(Overfitting)**



交叉驗證 (Cross-Validation)

原始資料集

訓練資料集 (Training Set) | 測試資料集 (Test Set)

訓練資料集 (Training Set) | 交叉驗證資料集(Cross Validation Set, CV) | 測試資料集 (Test Set)

K折交叉驗證法 (K-fold Cross Validation)：計算平均效能

1/k

Test

Test

⋮

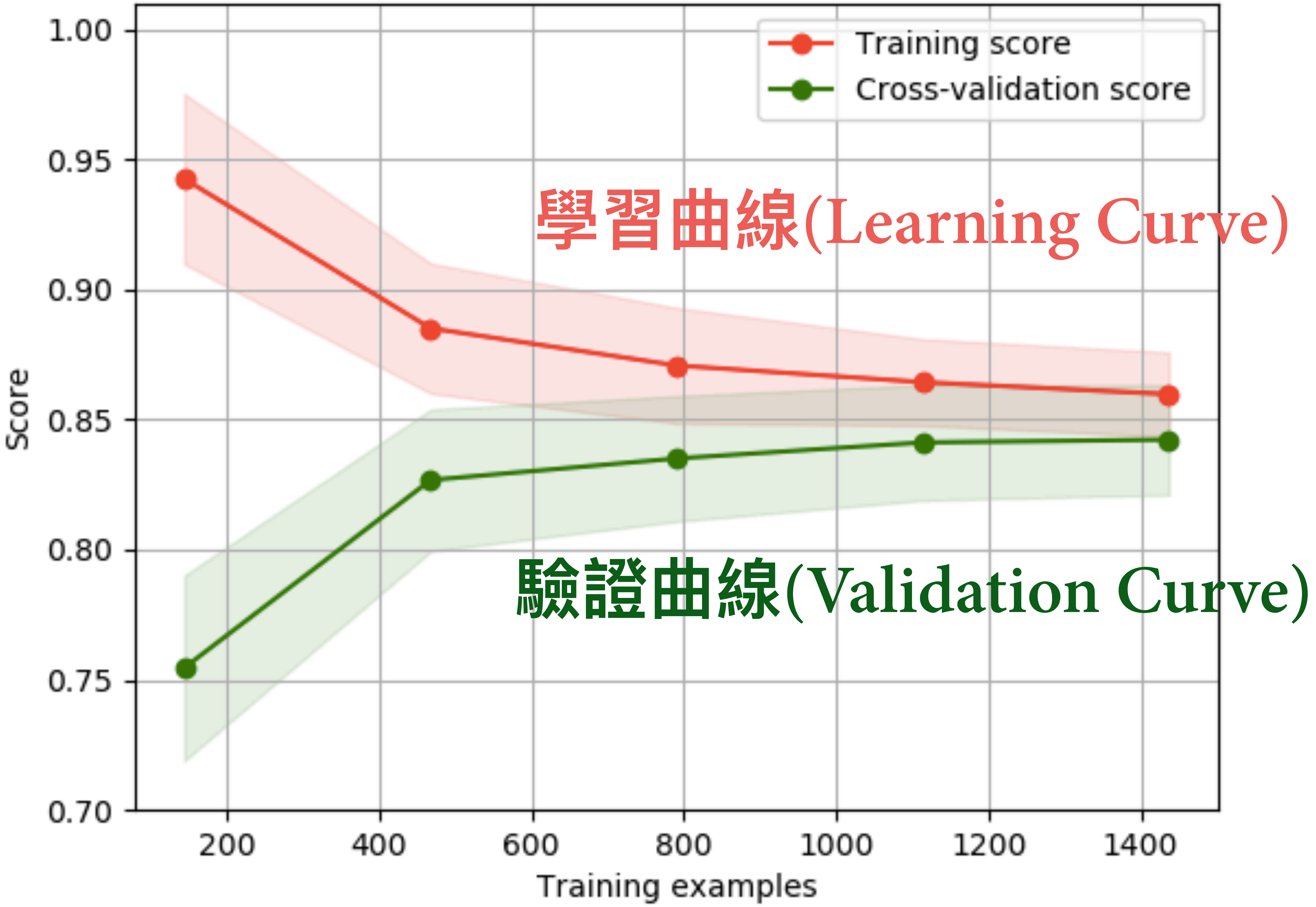
Test

Notes

- 模型泛化(generation)能力：面對新的未知數據仍能保有訓練時的準確度
- 應該一律使用CV作為模型選擇和調整依據，測試資料僅作為最後驗證用

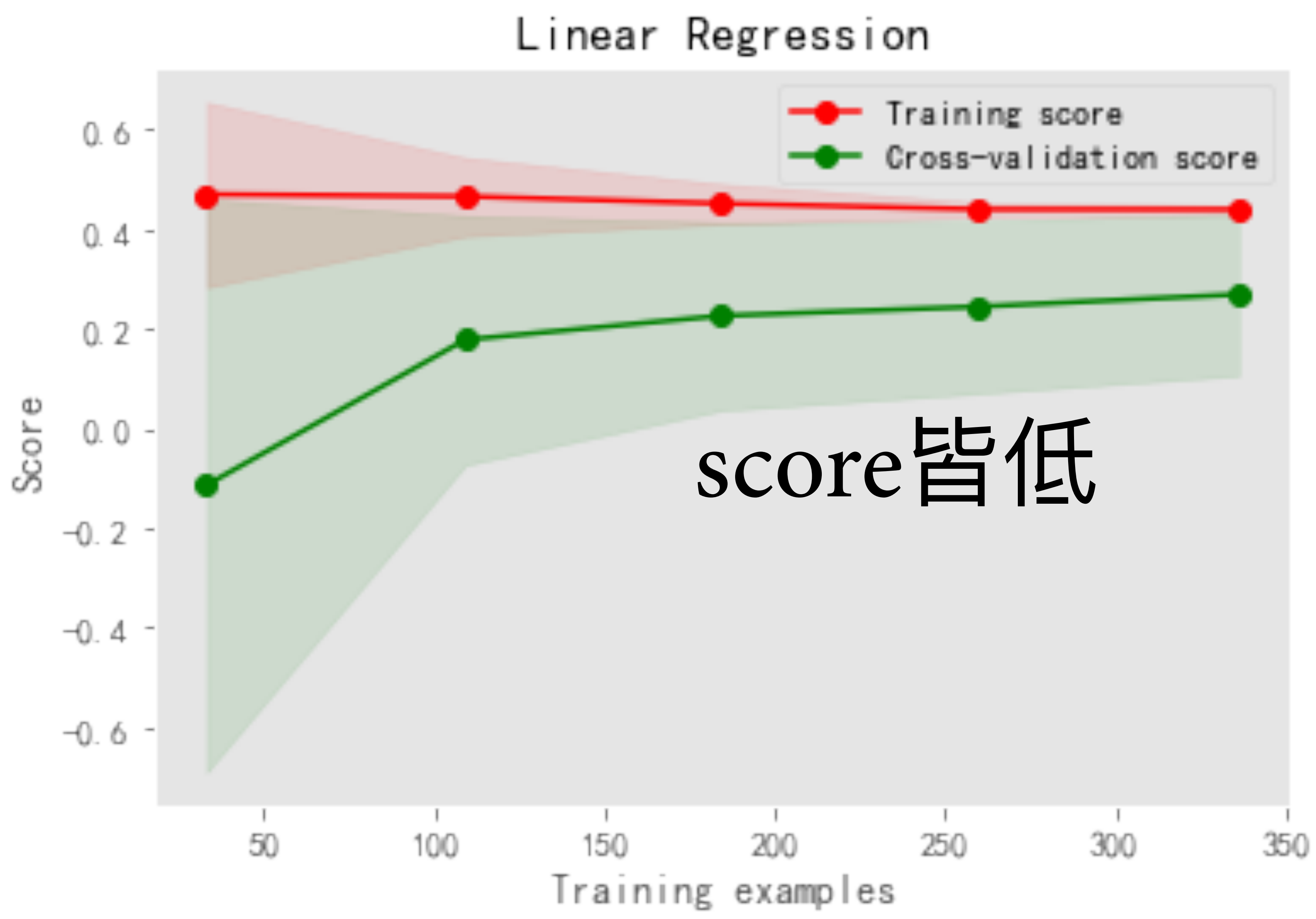
學習曲線與驗證曲線

理想狀態



偏誤 vs. 變異

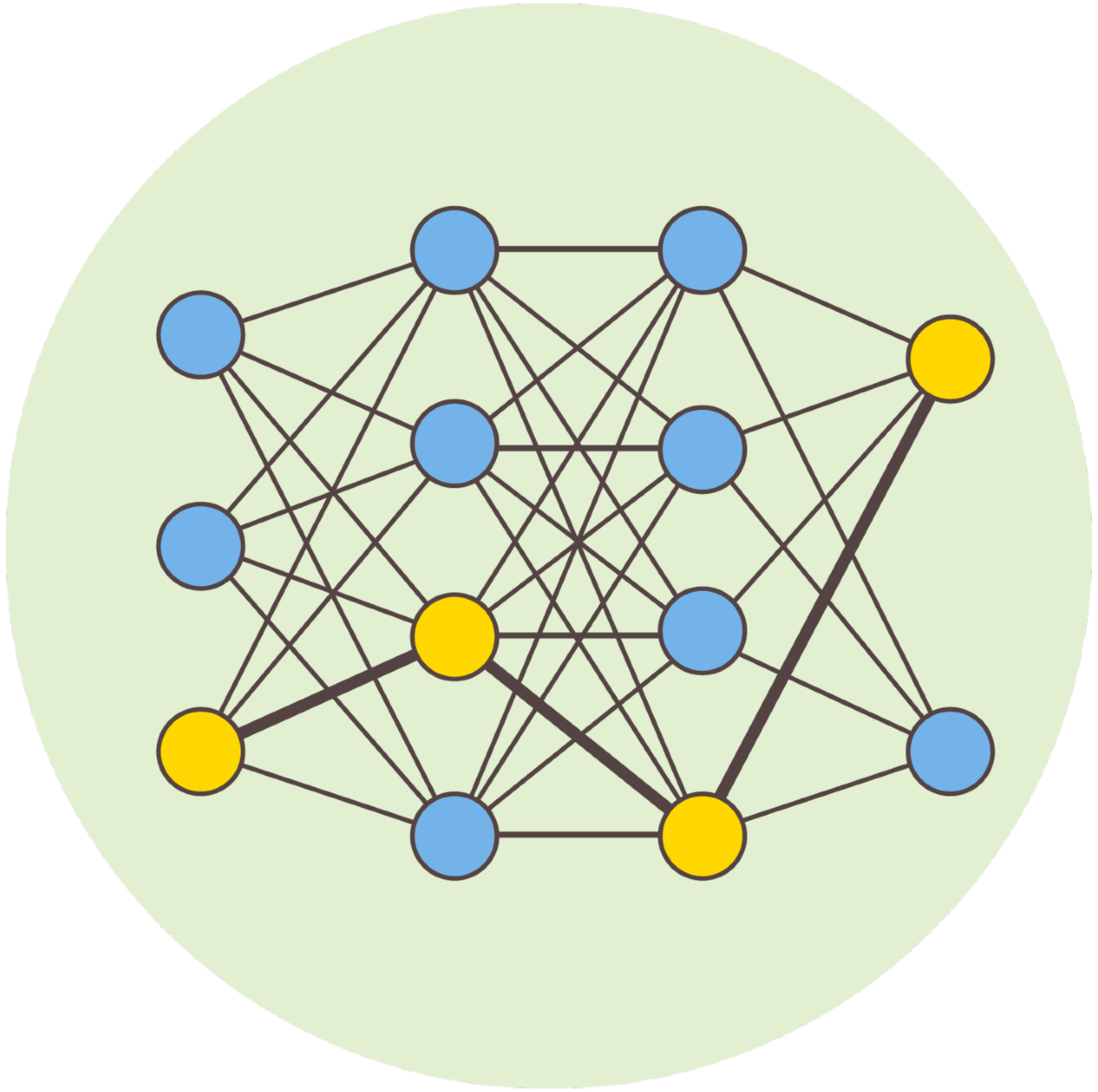
- 高偏誤(High Bias):
  - 欠擬合(Underfitting)



- 高變異(High Variance)
  - 過擬合/過適 (Overfitting)

Solutions to Underfitting/Overfitting

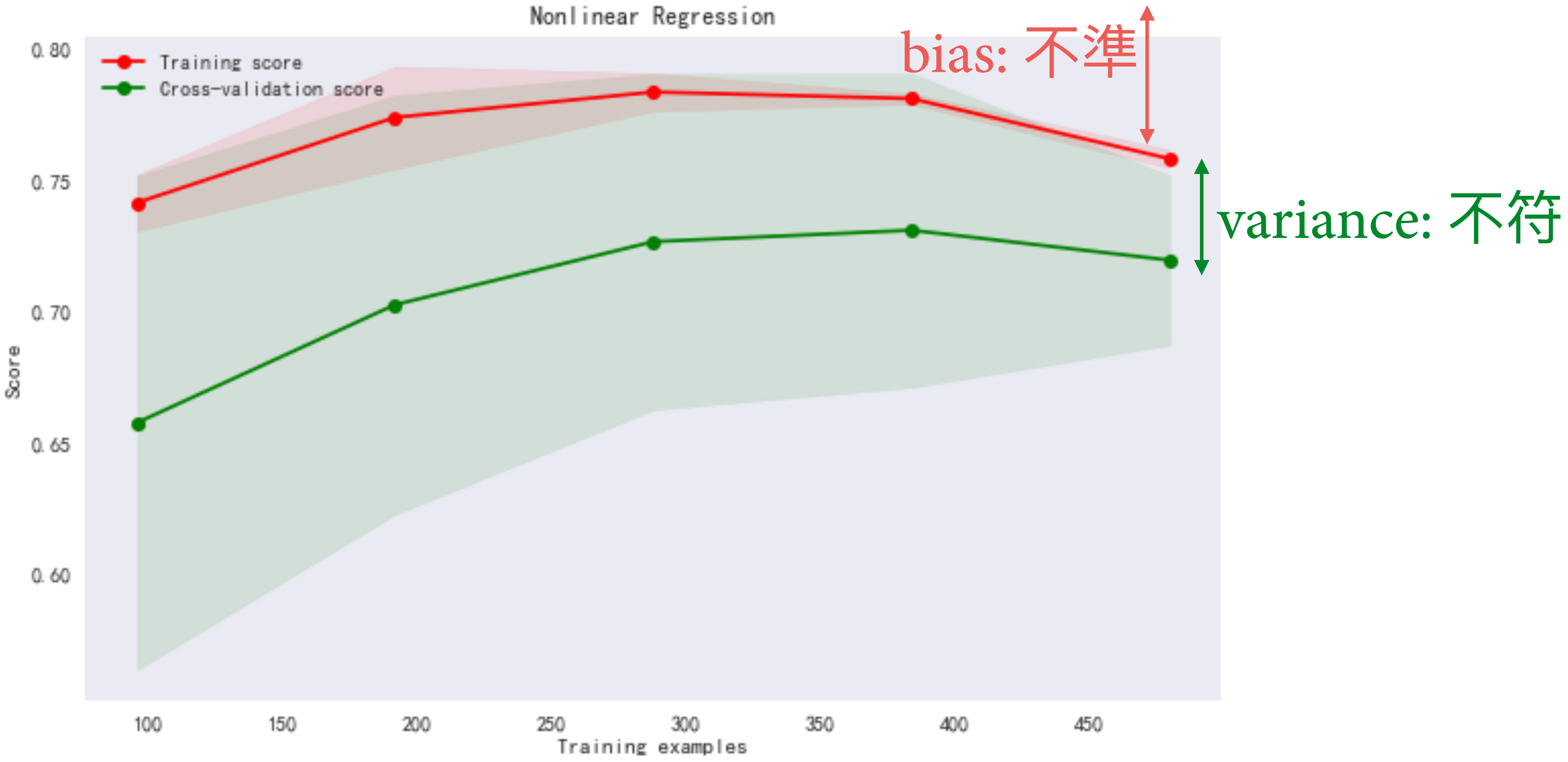
- **欠擬合(Underfitting)**
  - ✓ 增加特徵數量
  - ✓ 增加高次方變項
- **過擬合(Overfitting)**
  - ✓ 減少特徵數量
  - ✓ 增加資料筆數
  - ✓ 正規化(Regularization)



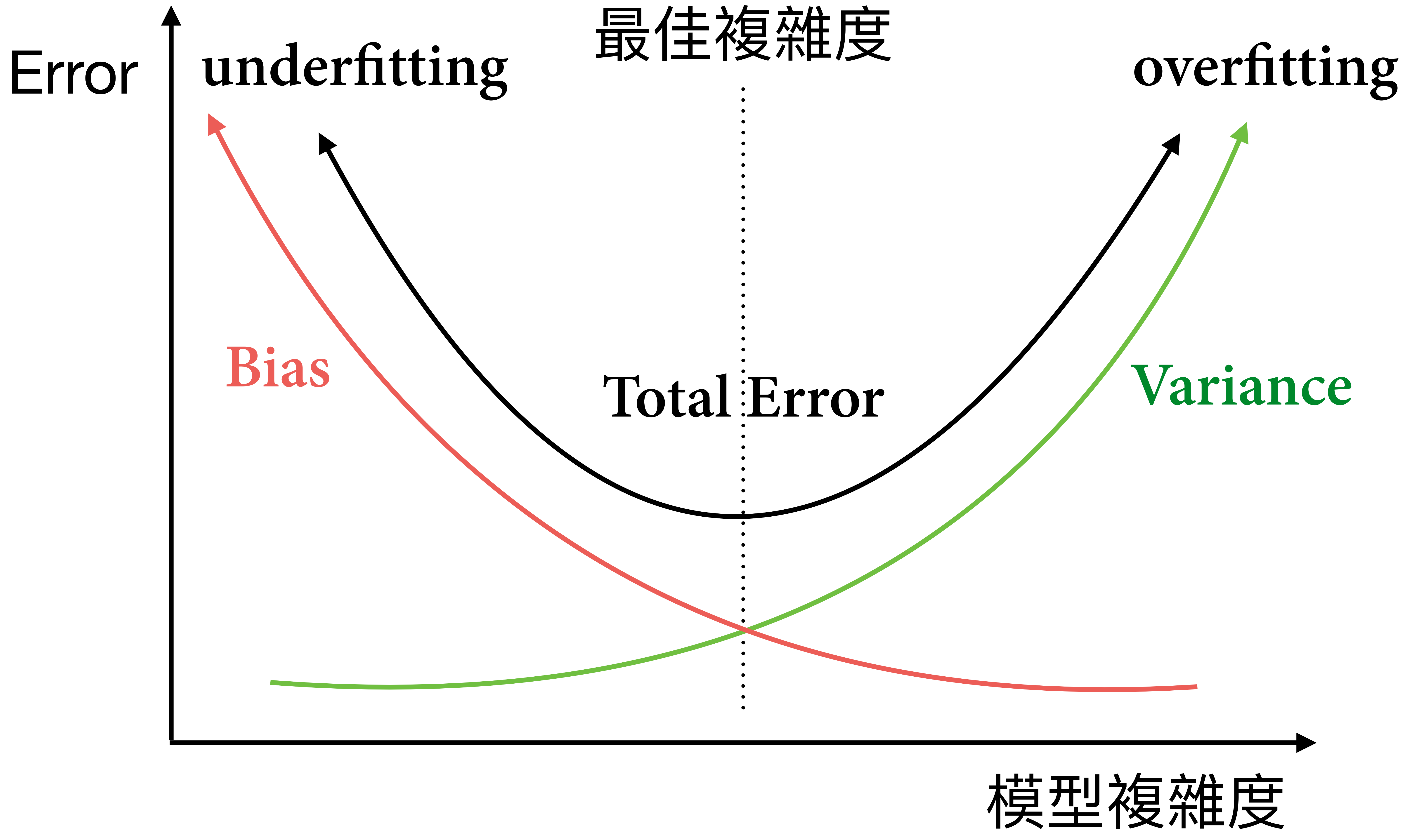
Python 機器學習與深度學習實作

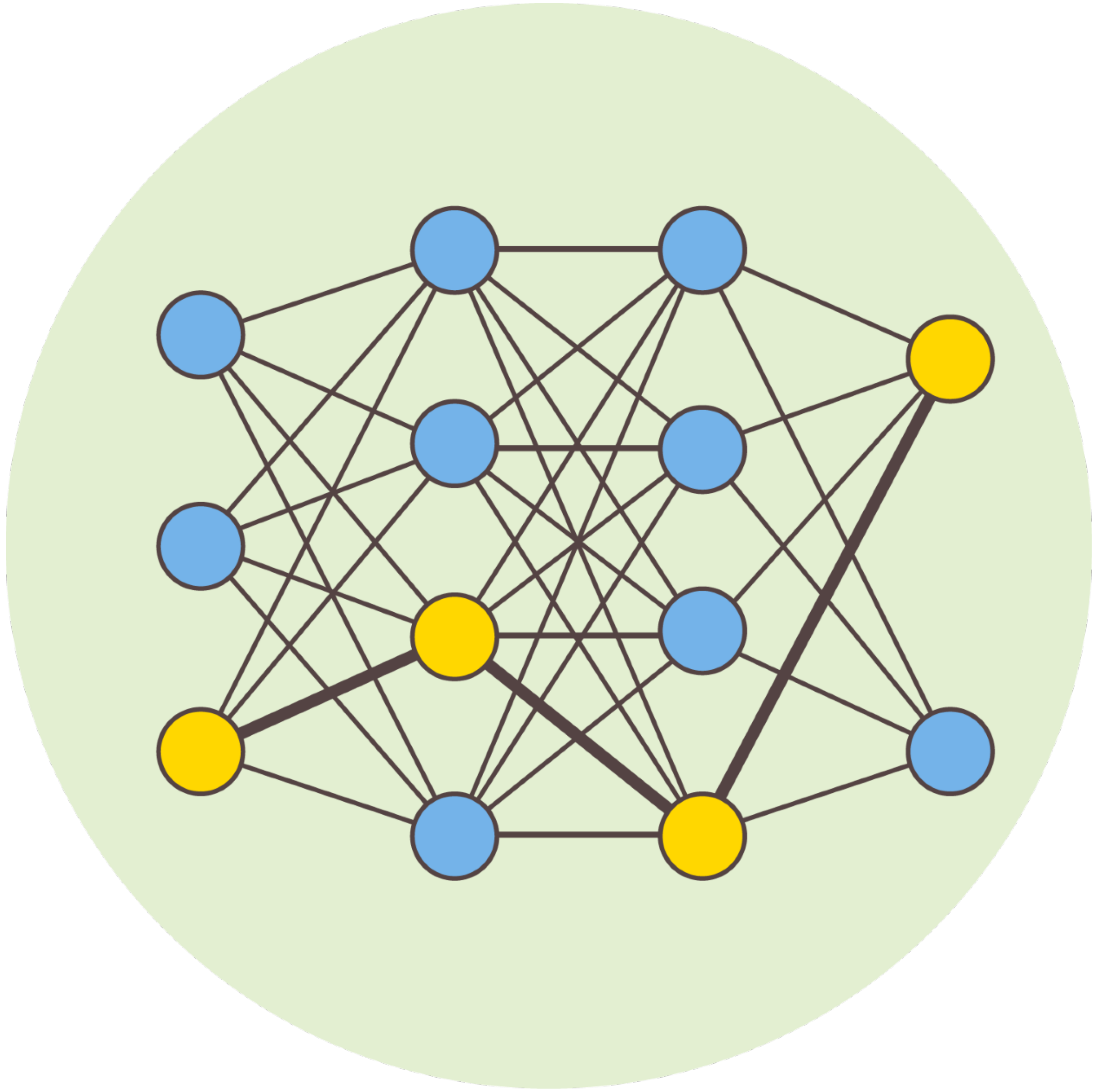
偏誤與變異權衡

Bias-Variance Tradeoff



Bias-Variance Tradeoff





Python 機器學習與深度學習實作

訓練資料預處理

標準化

- 標準化(Standardization)
  - $$x' = \frac{x - \overline{x}}{\sigma}$$
- 標準化後：
  - 平均 = 0
  - 標準差 = 1

Notes

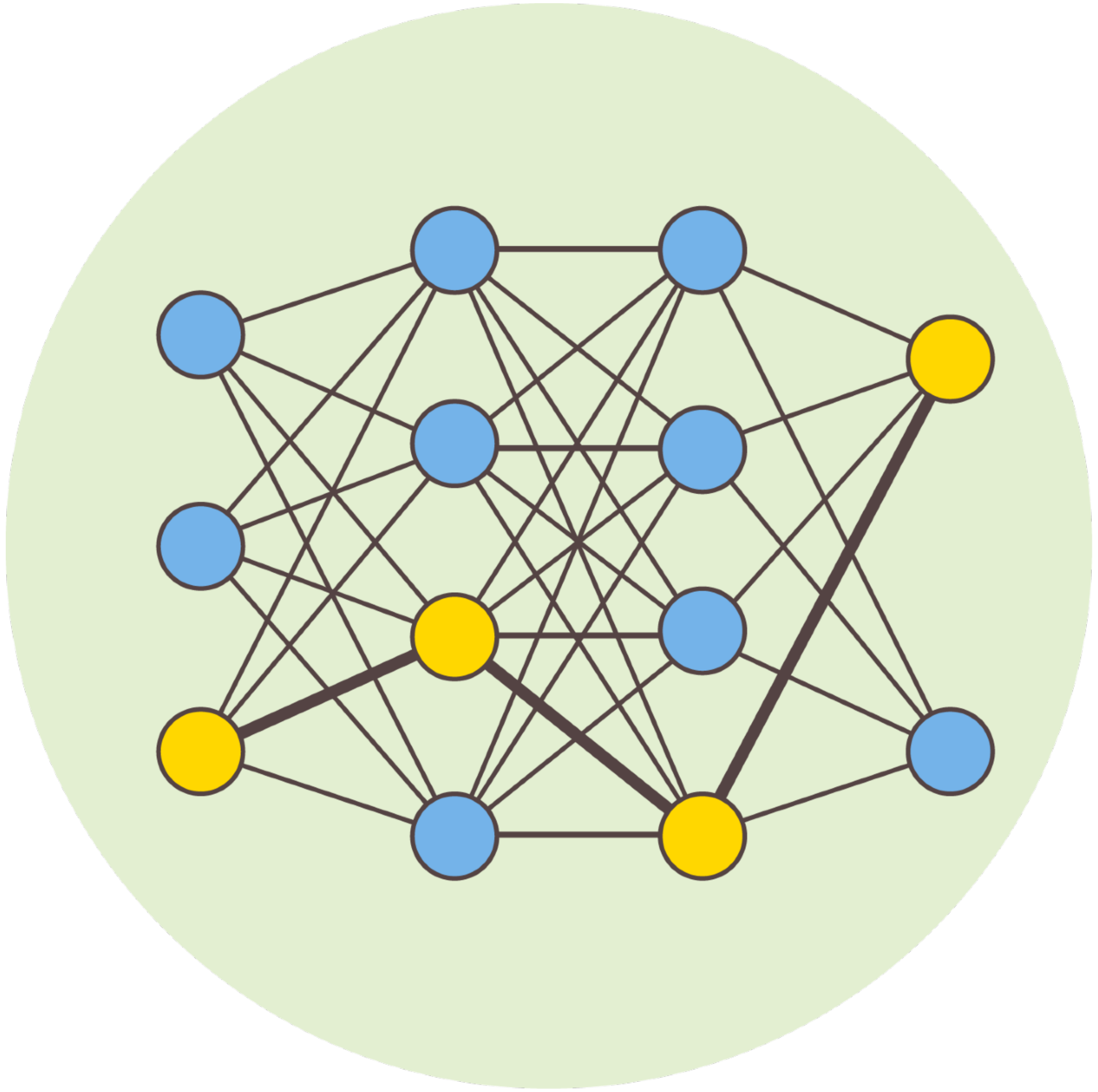
- $\overline{x}$: x 平均
- σ : x 標準差
- 標準差包含資料離散程度資訊，相較於直接把資料限縮於特定範圍內，標準化後對離群值較不敏感。

類別(categorical)資料編碼

- 有大小順序(順序量尺)
  - e.g. S -> 1, M -> 2, L -> 3
- 無大小順序(名義量尺)
  - One-hot Encoding

	顏色
0	紅色
1	藍色
2	綠色

	紅色	藍色	綠色
0	1	0	0
1	0	1	0
2	0	0	1



Python 機器學習與深度學習實作

線性迴歸與正規化

Overfitting 處理

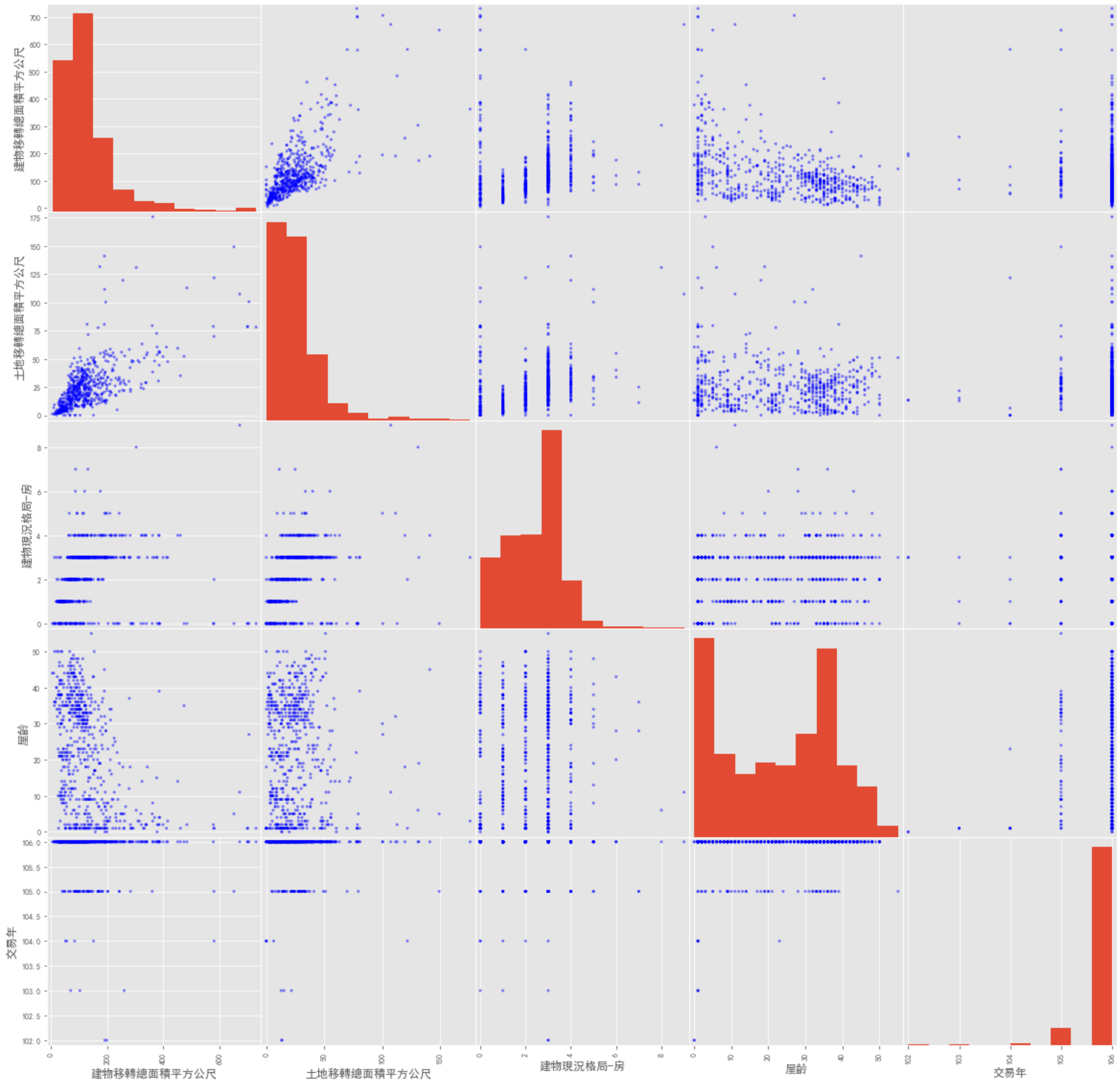
- 減少特徵數量：
  - 手動挑選特徵(利用domain knowledge)
  - 降維、特徵提取(Feature extraction)
  - 特徵重要性計算、特徵選取(Feature Selection)
- 增加資料量
- 正規化：降低權重過高的情況

Notes

- **於決策樹會再教大家如何做特徵選取 (Feature Selection)**
- **於非監督式學習的章節會再教大家如何降維與特徵提取(Feature extraction)**

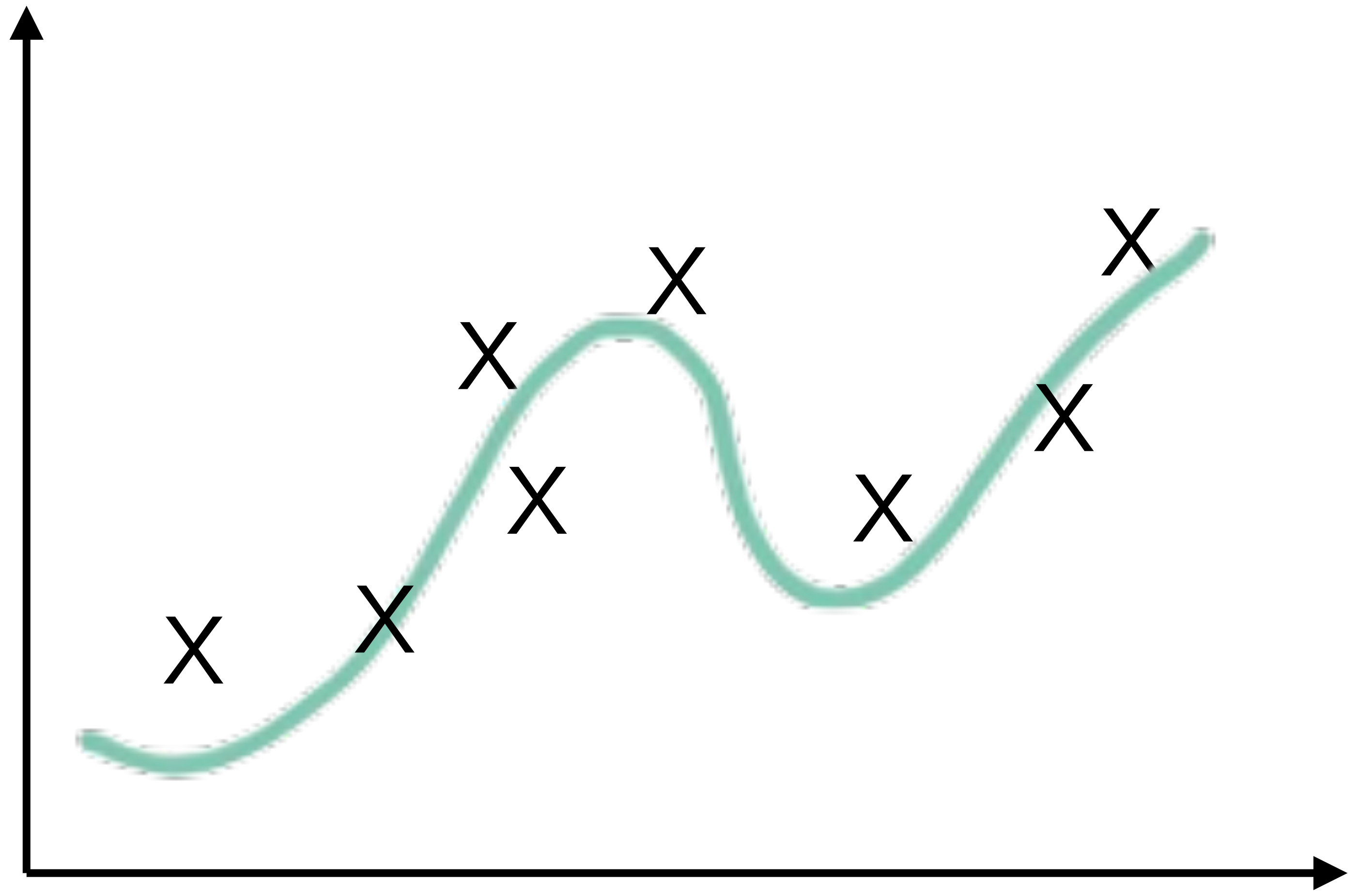
檢查特徵存在的線性關係

- 相關性分析
- 散佈圖



正規化 (Regularization)

- Regularization



$$y = w_0 + w_1 x_1 + w_2 x_1{}^2 + w_3 x_1{}^3 + w_4 x_1{}^4$$

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 + \boxed{1000 w_4 x_1{}^4}$$

$$w_4 \approx 0$$

懲罰項(penalty)

Notes

- 此方法又稱為權重衰減(Weight Decay)
- 限制weight的增長

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 + \alpha\sum_{j=1}^{n} w^2$$

L1, L2 正規化

- L2 Regularization

$$J(w)=\frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2+\alpha\sum_{j=1}^{n}w^2$$

- L1 Regularization

$$J(w)=\frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2+\alpha\sum_{j=1}^{n}|w|$$

Notes

- alpha 越大，正規化懲罰越大，無限大時w=0
- alpha 越小，正規化懲罰越小，alpha=0時，等於無正規化的線性迴歸

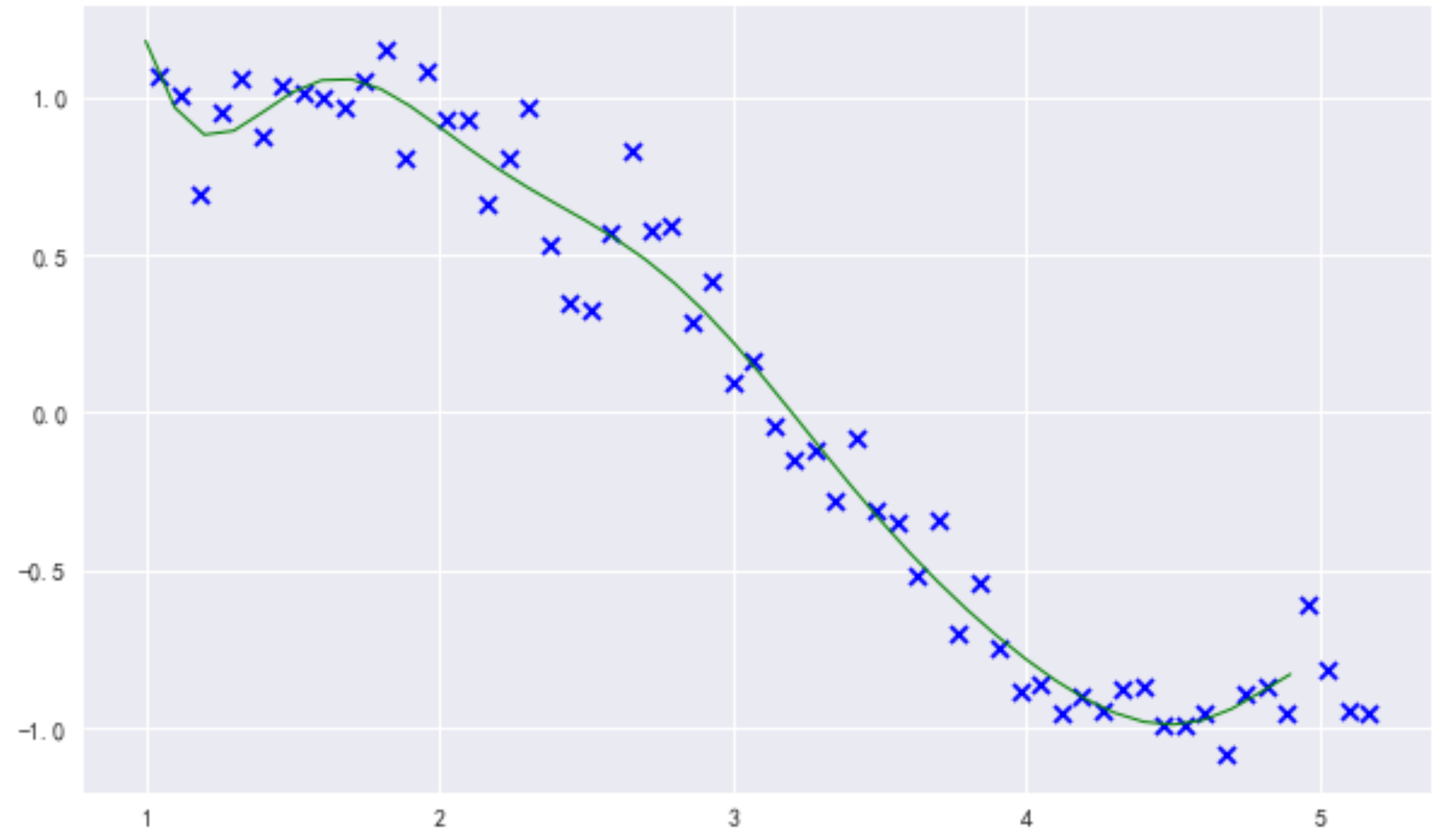
含正規化的迴歸

- Linear Regression with L2 Regularization
  - 脊迴歸 (Ridge Regression)
- Linear Regression with L1 Regularization
  - 最小絕對值收斂和選擇算子、套索算法(least absolute shrinkage and selection operator, LASSO)
- Linear Regression with both
  - 彈性網 (Elastic Net)

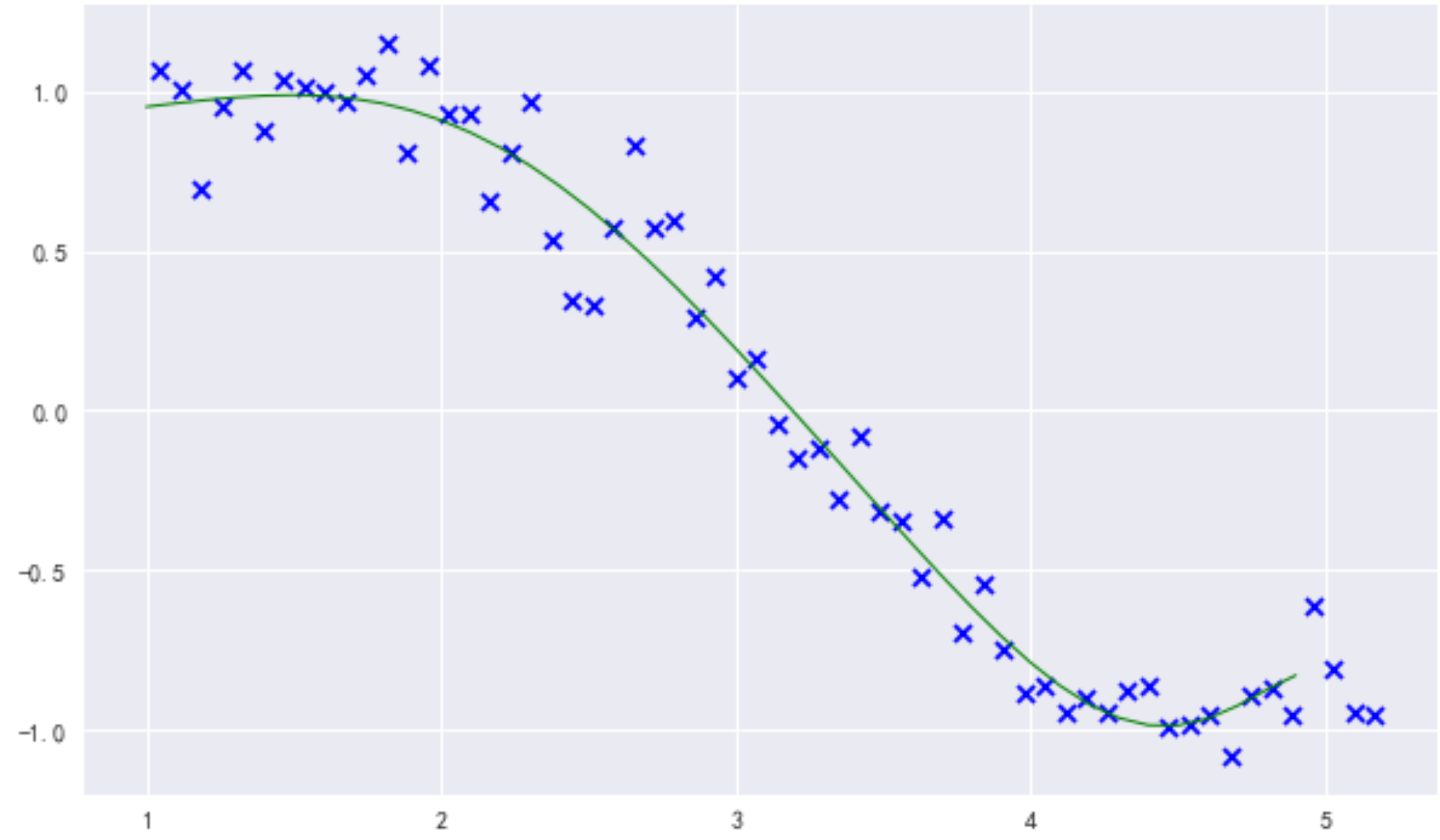
Notes

- Ridge Regression
  - from sklearn.linear_model import Ridge
- LASSO
  - from sklearn.linear_model import Lasso
- ElasticNet
  - from sklearn.linear_model import ElasticNet

Linear Regression with Regularization



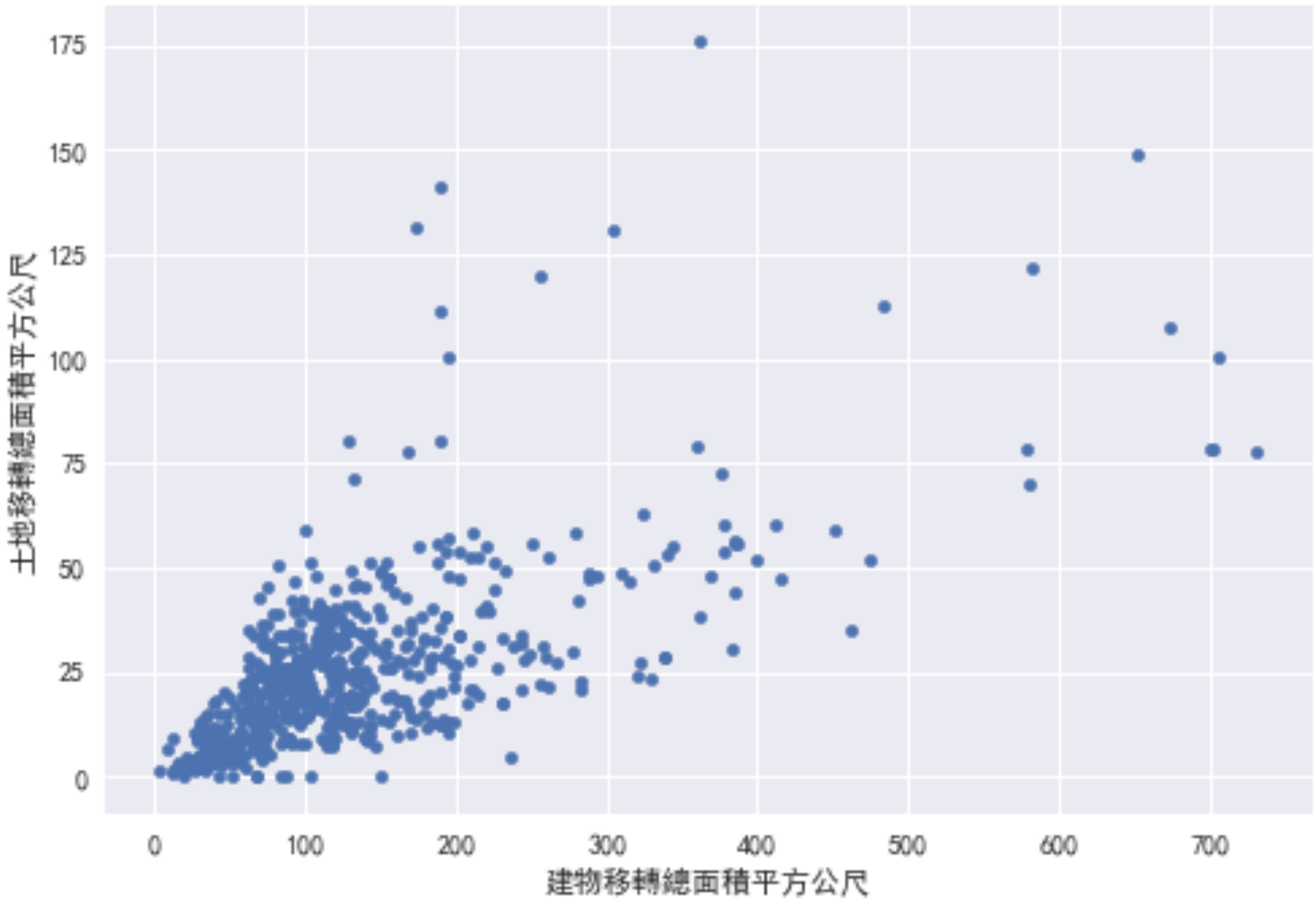
degree=12
Linear Regression



degree=12
Ridge Regression
(alpha=1)

共線性（Collinearity）

- 特徵之間存在線性相關：共線性(Collinearity)




```
Linear Regression weights:
建物移轉面積：2191.4998606
土地移轉面積：-275.7035364
屋鄰：-118.04770571
```

土地移轉面積越大，售價越低？

Handling Collinearity with Ridge

- Experiment Results:

```
Linear Regression weights:
建物移轉面積：2191.4998606
土地移轉面積：-275.7035364
屋鄰：-118.04770571
R Square: 0.727309001534
```

```
Ridge Regression weights: (alpha=100)
建物移轉面積：1506.46178566
土地移轉面積：137.74155002
屋鄰：-308.67028393
R Square: 0.660919727018
```

Notes

- 解決共線性問題，只是使權重值具解釋性，但準確度不一定會提升