

python 機器學習

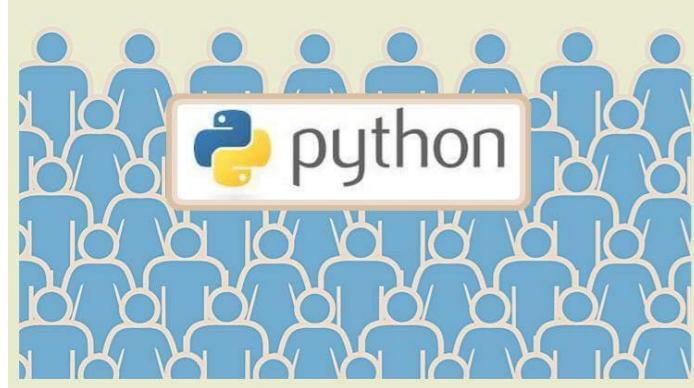




吳老師教學中心

www.facebook.com/chaiyenwu/

Python程式語言,大數據,人工智慧,機器學習,深度學習





Python 機器學習

吳佳諺 老師





Python 機器學習

1. Python 機器學習
2. 認知演算法
3. 加州大學大數據資料
4. 建立模型Perceptron
5. Python SVM萬用分類機
6. Python 類神經網路深度學習



1. Python 機器學習

監督式學習

SVM, SVR,

Perceptron, 類神經

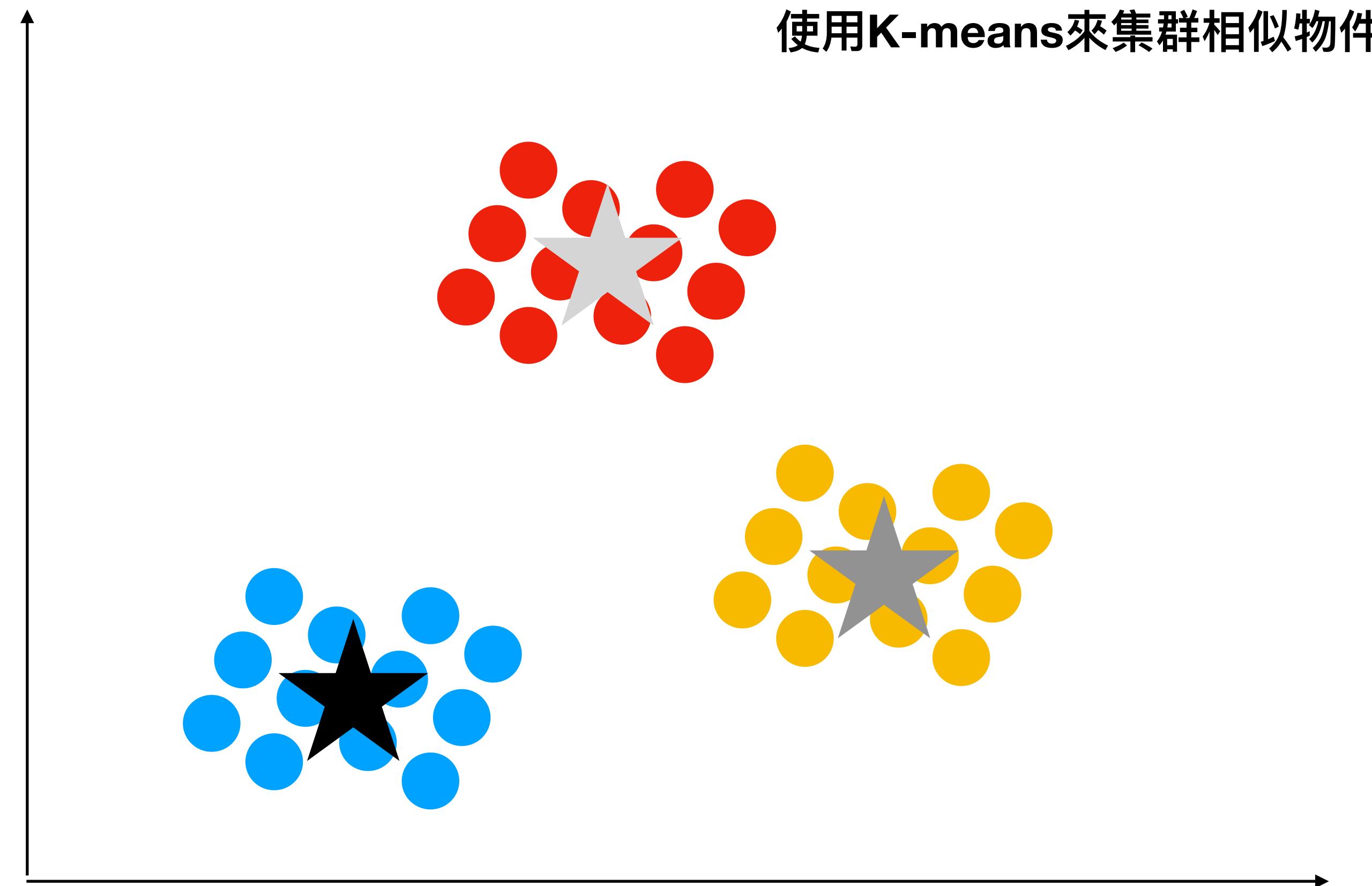
非監督式學習

分群

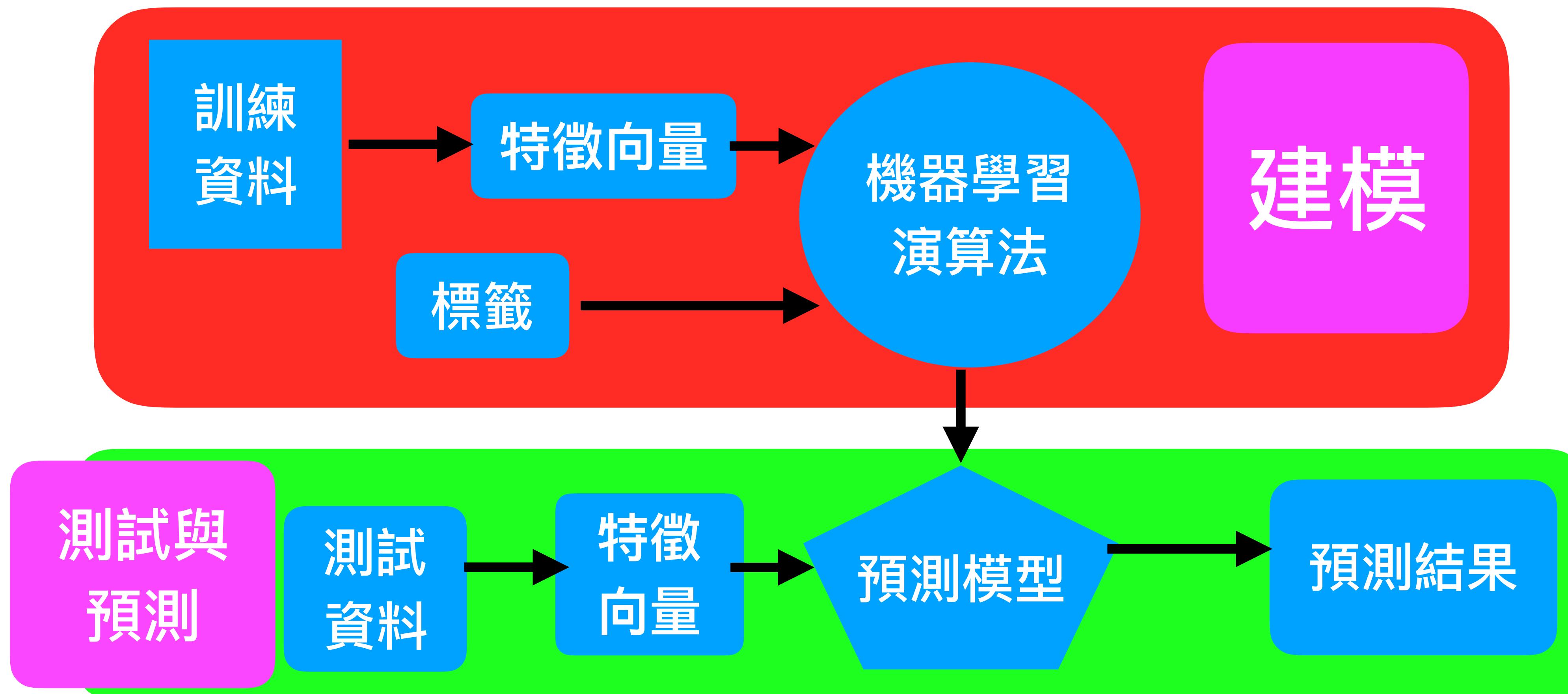
Reinforcement學習

機器學習

靠近星星的距離可分三群



Python 機器學習





2. 認知演算法

```
1: w  $\leftarrow$  0, b  $\leftarrow$  0
2: repeat
3:   errors  $\leftarrow$  0
4:   /* cycle through all training examples */
5:   for i = 1, ..., n do
6:     compute  $f(\mathbf{x}_i) = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ 
7:     if  $f(\mathbf{x}_i) \neq y_i$  then
8:       w  $\leftarrow$  w +  $y_i \mathbf{x}_i$ 
9:       b  $\leftarrow$  b +  $y_i$ 
10:      errors  $\leftarrow$  errors + 1
11:    end if
12:  end for
13: until error = 0
14: output w, b
```





3. 加州大學鳶尾花資料

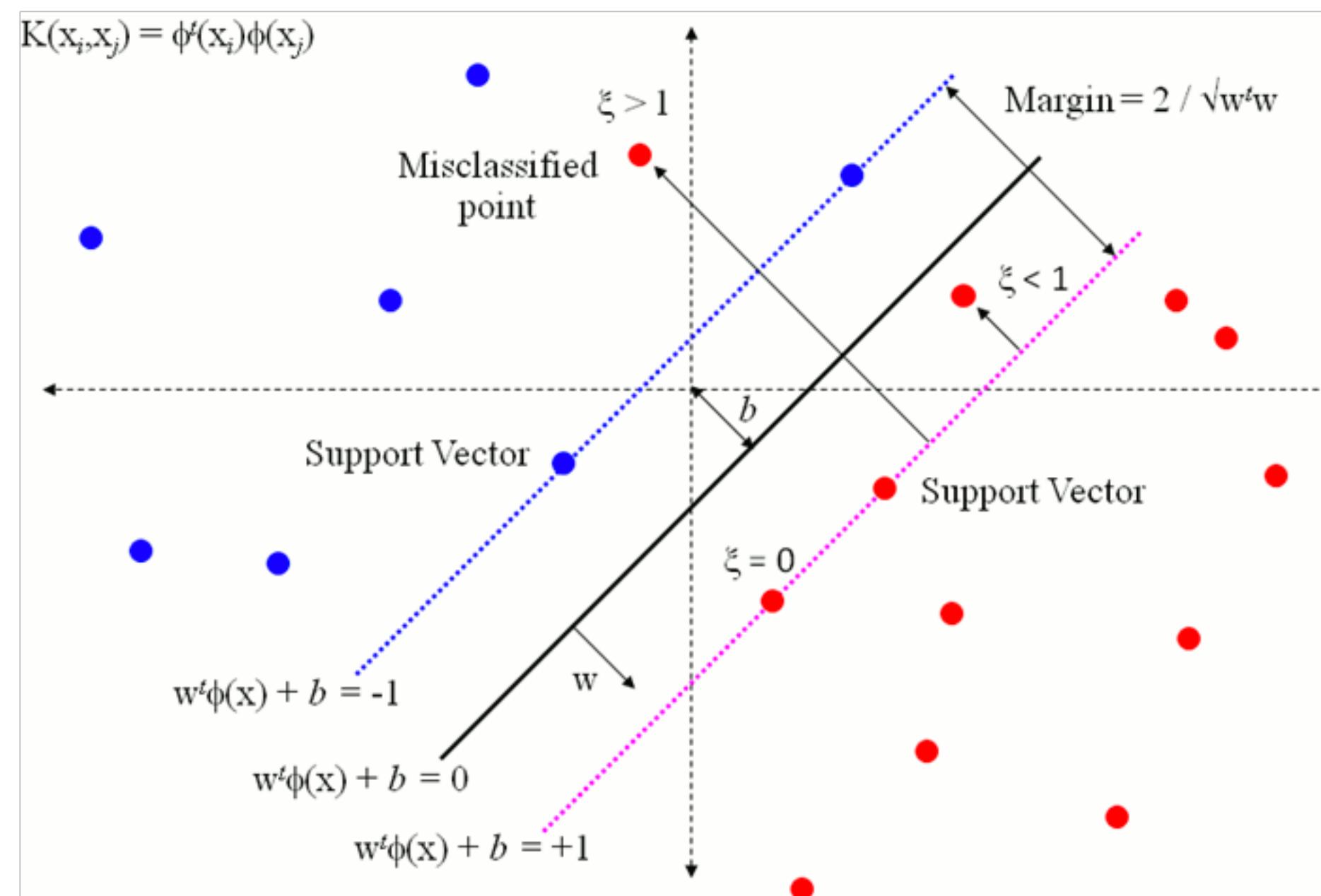
```
34 #####
35 print(50 * '=')
36 print('認知演算法讀取加州大學鳶尾花資料')
37 print(50 * '-')
38
39 df = pd.read_csv('https://archive.ics.uci.edu/ml/'
40                   'machine-learning-databases/iris/iris.data',
41                   header=None)
42 print(df.tail())
```

4. 建立模型Perceptron().fit()

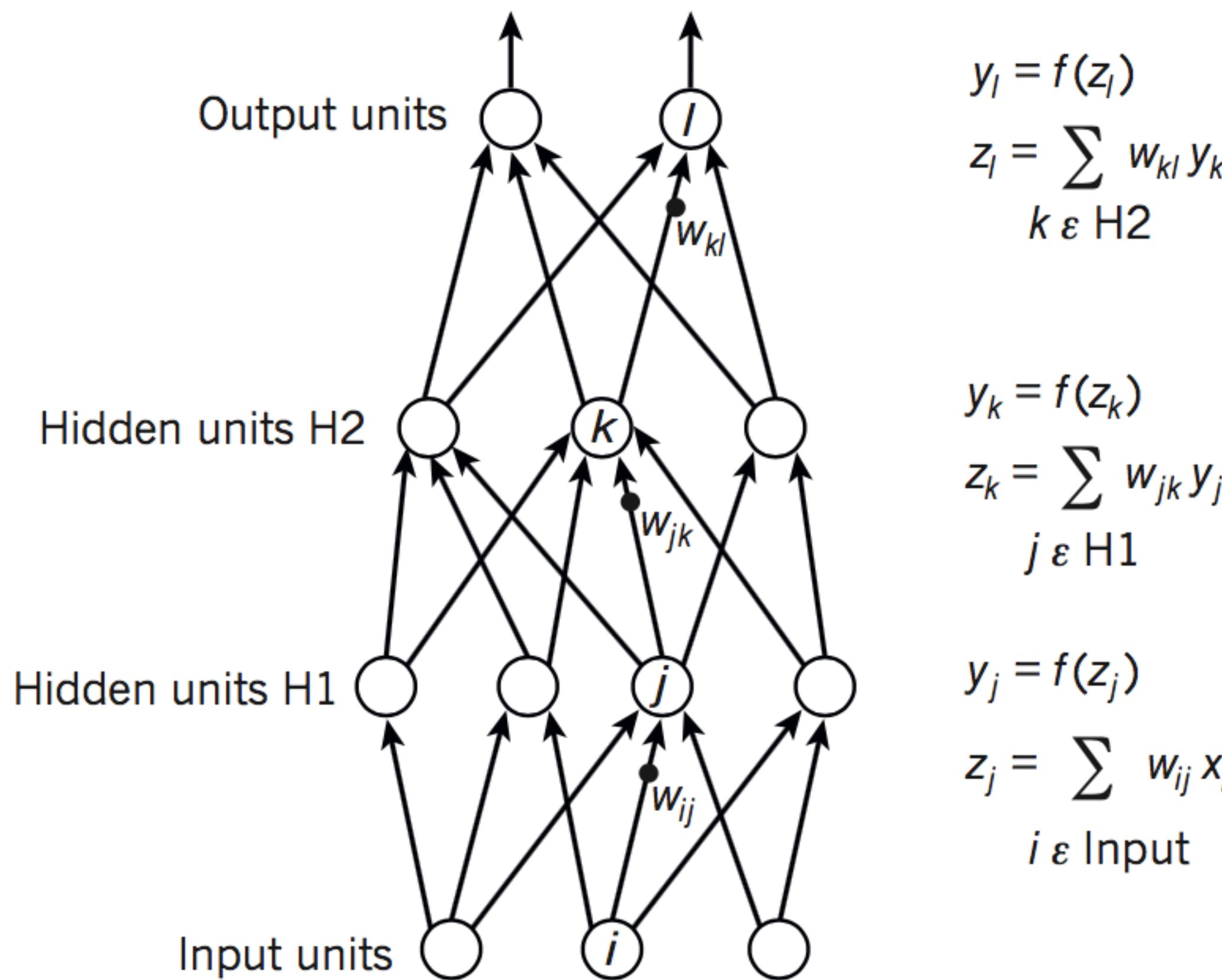
plot_decision_regions()為繪製區域

```
99
100 ppn = Perceptron(eta=0.1, n_iter=10)
101 ppn.fit(X, y)
102 plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
103 plt.xlabel('Epochs')
104 plt.ylabel('Number of misclassifications')
105 plt.show()
106 plt.xlabel('sepal length [cm]')
107 plt.ylabel('petal length [cm]')
108 plt.legend(loc='upper left')
109 plot_decision_regions(X, y, classifier=ppn)
```

5.Python SVM萬用分類機



6. Python 類神經網路深度學習



$$y_l = f(z_l)$$

$$z_l = \sum_{k \in H2} w_{kl} y_k$$

$$y_k = f(z_k)$$

$$z_k = \sum_{j \in H1} w_{jk} y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

- Thanks.

Python 機器學習

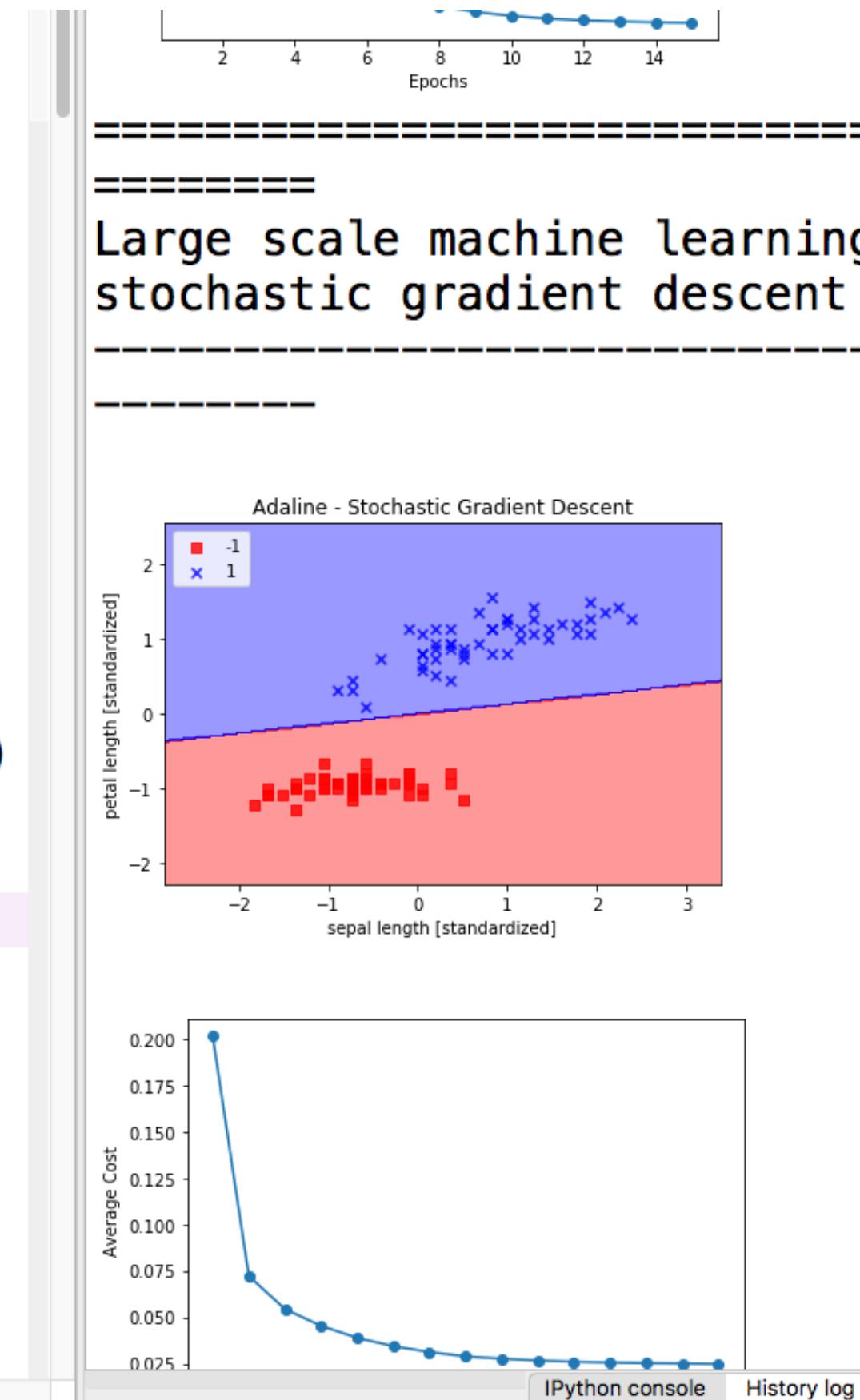
吳佳諺 老師

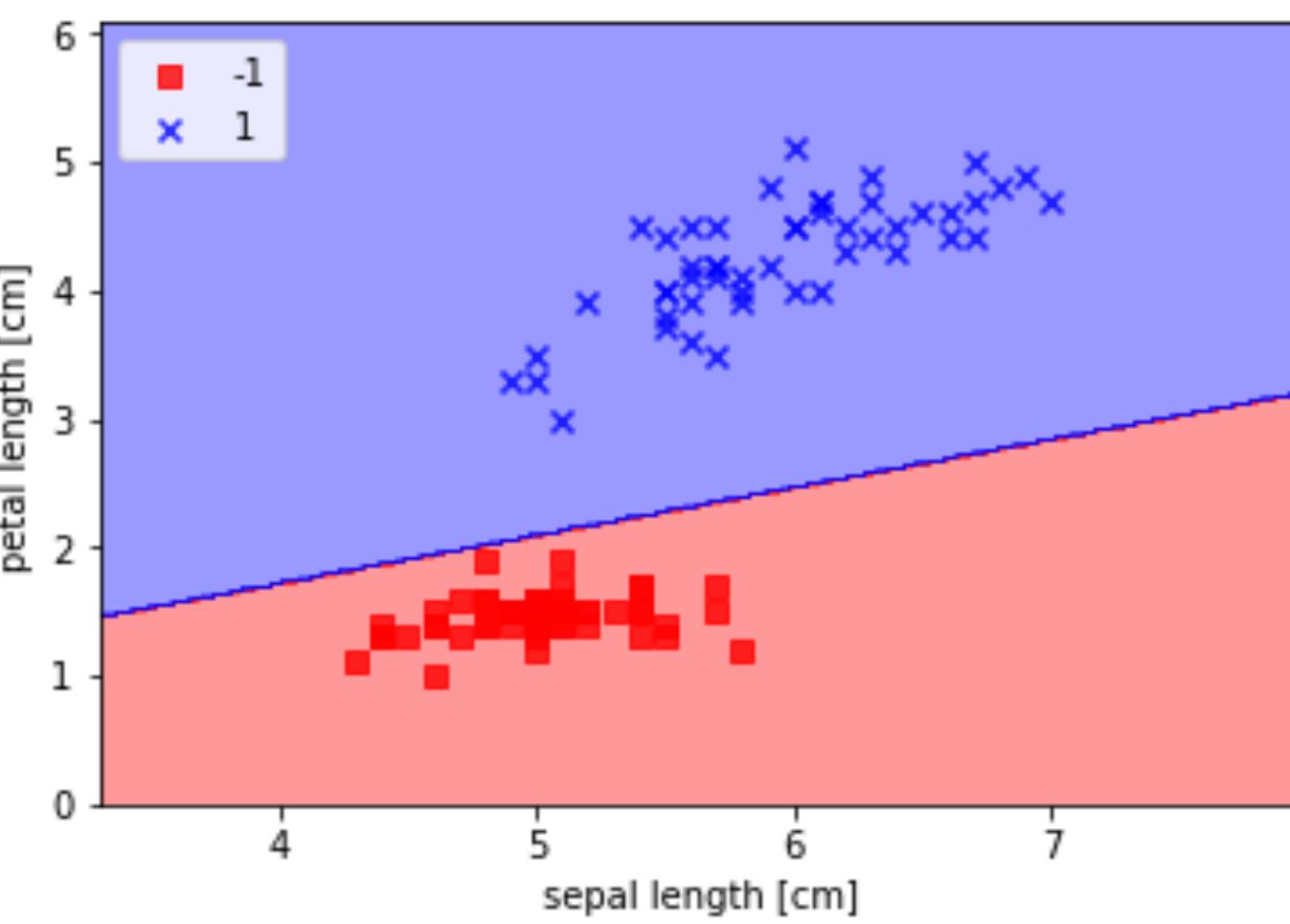
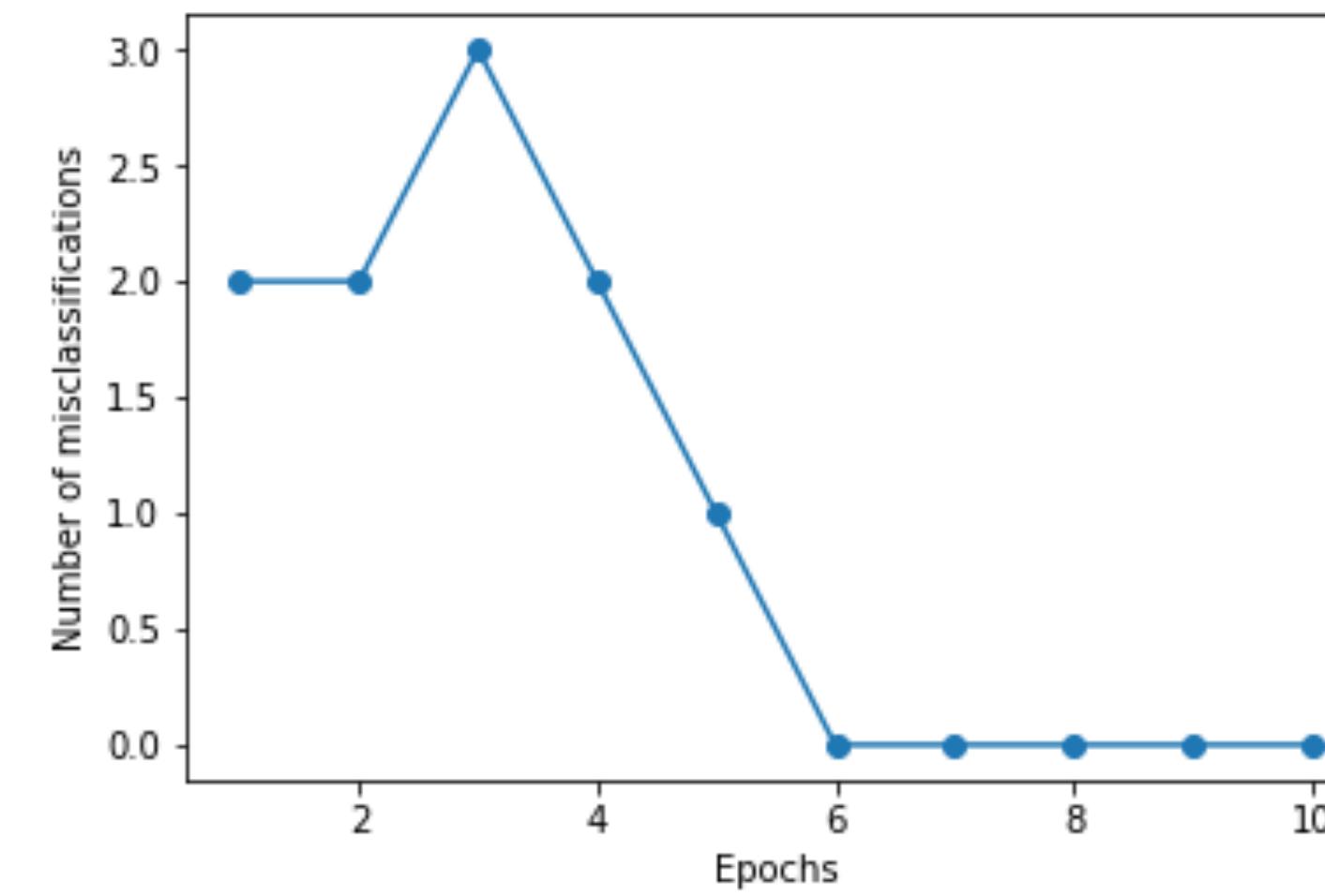
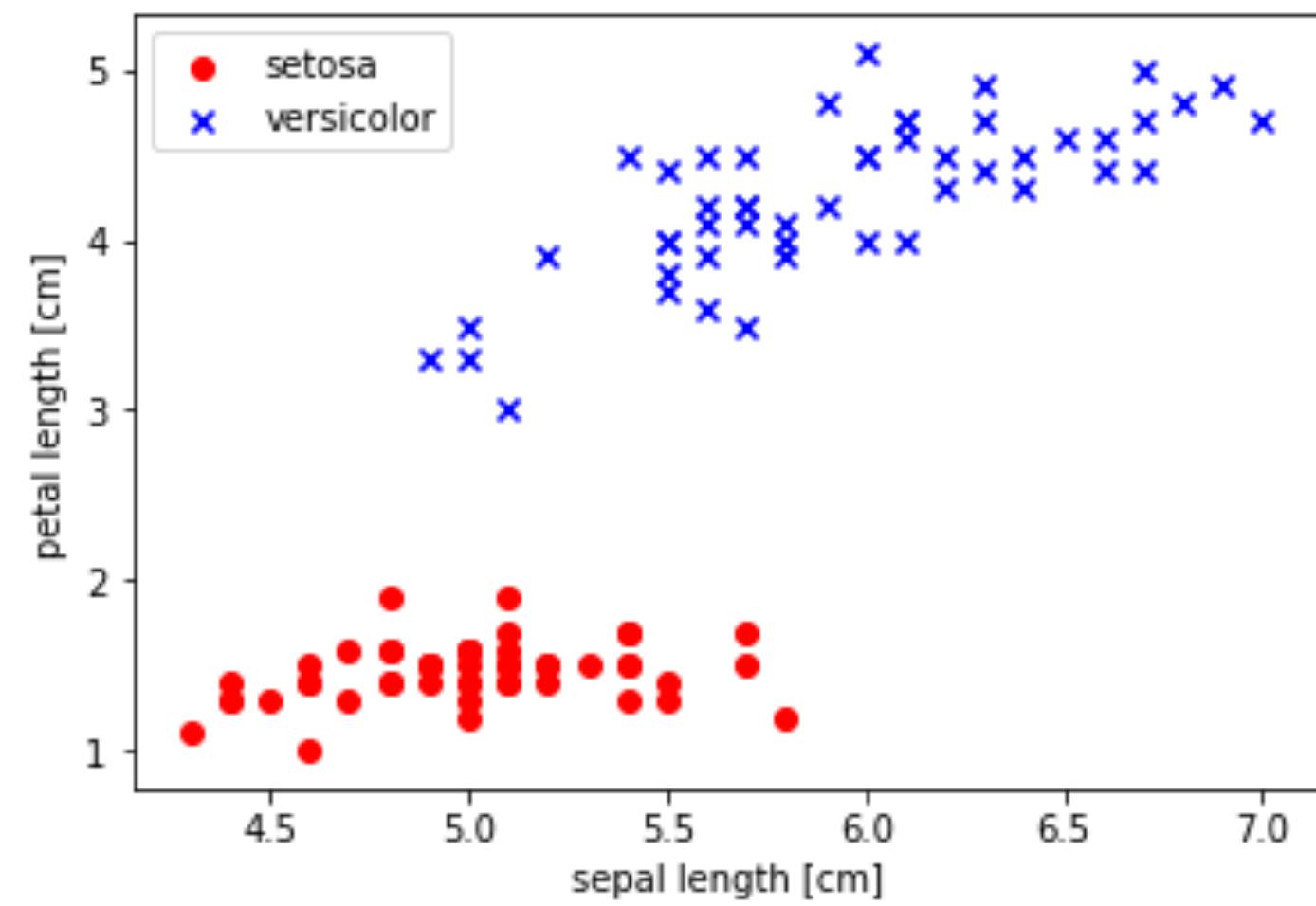
Python 機器學習

1. Python 機器學習
2. 數組tuple和集合set
3. Scipy科學函數庫
4. numpy模組建立矩陣
5. Pandas資料結構
6. Matplotlib畫圖

1. Python 機器學習

```
8 class Perceptron(object):
9
10    def __init__(self, eta=0.01, n_iter=10):
11        self.eta = eta
12        self.n_iter = n_iter
13
14    def fit(self, X, y):
15        self.w_ = np.zeros(1 + X.shape[1])
16        self.errors_ = []
17
18        for _ in range(self.n_iter):
19            errors = 0
20            for xi, target in zip(X, y):
21                update = self.eta * (target - self.predict(xi))
22                self.w_[1:] += update * xi
23                self.w_[0] += update
24                errors += int(update != 0.0)
25            self.errors_.append(errors)
26        return self
27
28    def net_input(self, X):
29        return np.dot(X, self.w_[1:]) + self.w_[0]
30
31    def predict(self, X):
32        return np.where(self.net_input(X) >= 0.0, 1, -1)
```





認知演算法

```
1: w  $\leftarrow$  0, b  $\leftarrow$  0
2: repeat
3:   errors  $\leftarrow$  0
4:   /* cycle through all training examples */
5:   for i = 1, ..., n do
6:     compute  $f(\mathbf{x}_i) = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ 
7:     if  $f(\mathbf{x}_i) \neq y_i$  then
8:       w  $\leftarrow$  w +  $y_i \mathbf{x}_i$ 
9:       b  $\leftarrow$  b +  $y_i$ 
10:      errors  $\leftarrow$  errors + 1
11:    end if
12:  end for
13: until error = 0
14: output w, b
```

2. 數組tuple和集合set

- 可以用數組tuple來儲存固定的元素, 使用小括號()來建立一數組tuple
- 集合的元素放置沒有按照順序, 可以使用{}大括號來建立一集合Set

Tuple數組

- 也可以從字串中建立數組
- `tp5 = tuple('Ivy Lin')`
- 從數組得到串列
- `list1 = list(tp5)`

```
8
9 tp1=()
10 print(tp1)
11 tp2=(1,2,3,4,5,6,7,8)
12 print(tp2)
13 print(sum(tp2))
14 print('-----')
15 print(tp2[2:5])#切削運算子
16 print(tp2[-1])
17 tp3 =tuple([2*x for x in range(1,8)])
18 print(tp3)
19 print('-----')
20 tp4=tuple('Ivy Lin')
21 print(tp4)
22 tp5=("John",'小寶','小文')
23 print(tp5)
24 print(len(tp5))
25 print(tp4+tp5)
26 print('-----')
27 tp6=tuple([1,2,3,4,5,6,7,8,9])
28 print(tp6)
29 print(max(tp6))
30 print(min(tp6))
31
```

```
In [6]: runfile('/Users/justinwu/Desktop/tuple.py', wdir='Desktop')
()
(1, 2, 3, 4, 5, 6, 7, 8)
36
-----
(3, 4, 5)
8
(2, 4, 6, 8, 10, 12, 14)
-----
('I', 'v', 'y', ' ', 'L', 'i', 'n')
('John', '小寶', '小文')
3
('I', 'v', 'y', ' ', 'L', 'i', 'n', 'John', '小寶', '小文')
-----
(1, 2, 3, 4, 5, 6, 7, 8, 9)
9
1

In [7]:
```

```
8  
9 tp6=tuple([66,22,3,46,5,65,7,83,19])  
10 print(tp6)  
11 list1 = list(tp6)  
12 list1.sort()#排序串列  
13 print(list1)  
14 print('-----')  
15 tp8=tuple(list1)  
16 tp9=tuple(list1)  
17 print(tp8)  
18 print(tp8 == tp9)#比較兩個數組tuple  
19  
20  
21
```

```
In [15]: runfile('/Users/justinwu/Desktop')  
(66, 22, 3, 46, 5, 65, 7, 83, 19)  
[3, 5, 7, 19, 22, 46, 65, 66, 83]  
-----  
(3, 5, 7, 19, 22, 46, 65, 66, 83)  
True  
In [16]:
```

Set集合

- 集合(set)用來儲存沒有重複的元素.
- 集合的元素是不可以複製的,元素放置也沒有按照順序
- 可以使用{}大括號來建立一集合Set

Set集合

```
8  
9 st1=set()#建立一個空集合  
10 st2=set([1,2,3,4,5])  
11 print(st2)  
12 st3={'a','b','c','d','e'}  
13 print(st3)  
14 print('-----')  
15 st3.add('f')  
16 print(st3)  
17 st3.remove('d')  
18 print(st3)  
19 print('-----')  
20 print(st3.union(st2))  
21 st5={'a','b','c','d','e'}  
22 print(st3.intersection(st5))  
23 print(st3.difference(st5))
```

```
In [30]: runfile('/Users/justinwu/Desktop')  
{1, 2, 3, 4, 5}  
{'d', 'a', 'e', 'c', 'b'}  
-----  
{'d', 'f', 'a', 'e', 'c', 'b'}  
{'f', 'a', 'e', 'c', 'b'}  
-----  
{1, 2, 3, 4, 5, 'f', 'a', 'e', 'c', 'b'}  
{'b', 'a', 'c', 'e'}  
{'f'}  
  
In [31]:
```

3.Scipy科學函數庫

- Scipy模組提供科學運算及線性代數的運算模組

Python Module Index

s

s

- scipy

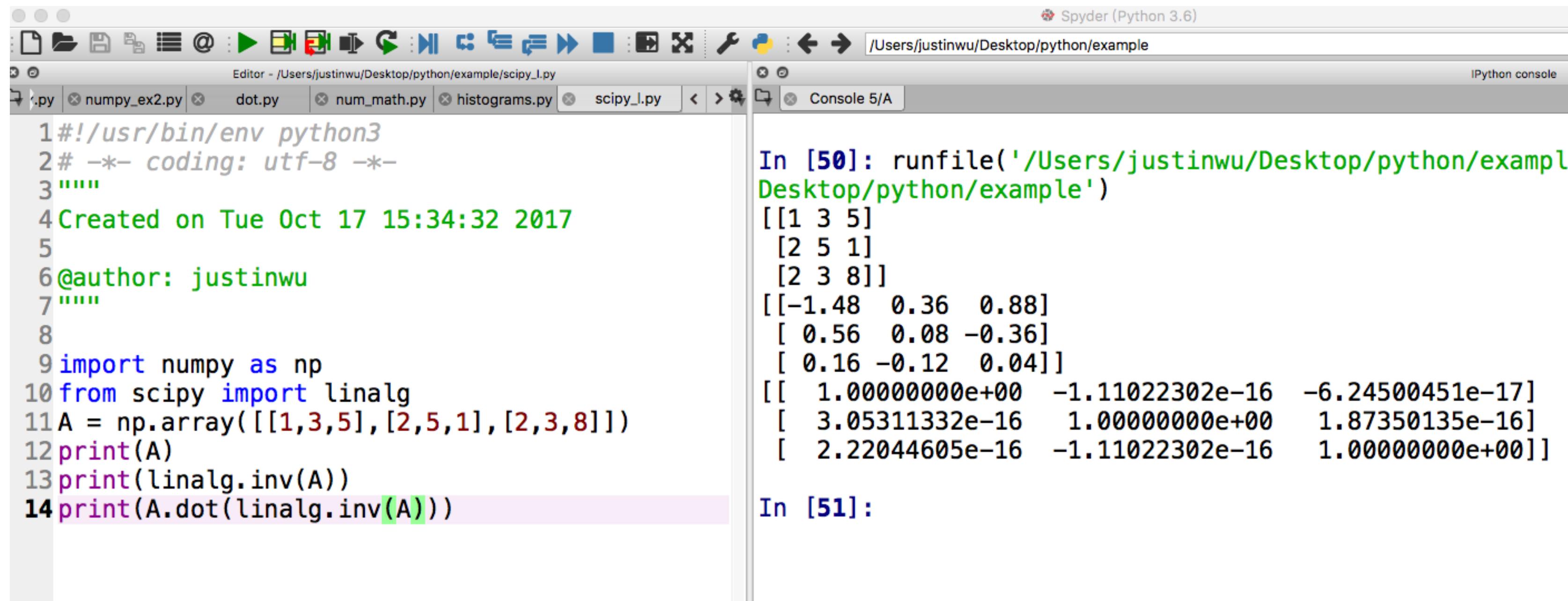
scipy.cluster
scipy.cluster.hierarchy
scipy.cluster.vq
scipy.constants
scipy.fftpack
scipy.fftpack.convolve
scipy.integrate
scipy.interpolate
scipy.io
scipy.io.arff
scipy.io.netcdf
scipy.io.wavfile
scipy.linalg
scipy.linalg.blas
scipy.linalg.cython_blas
scipy.linalg.cython_lapack
scipy.linalg.interpolative
scipy.linalg.lapack
scipy.misc
scipy.ndimage
scipy.odr

使用scipy的linalg作線性代數運算

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{25} \begin{bmatrix} -37 & 9 & 22 \\ 14 & 2 & -9 \\ 4 & -3 & 1 \end{bmatrix} = \begin{bmatrix} -1.48 & 0.36 & 0.88 \\ 0.56 & 0.08 & -0.36 \\ 0.16 & -0.12 & 0.04 \end{bmatrix}.$$

使用scipy的linalg作線性代數運算



The screenshot shows the Spyder IDE interface with two main panes: an Editor and an IPython console.

Editor: The code in the editor is as follows:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 15:34:32 2017
5
6@author: justinwu
7"""
8
9import numpy as np
10from scipy import linalg
11A = np.array([[1,3,5],[2,5,1],[2,3,8]])
12print(A)
13print(linalg.inv(A))
14print(A.dot(linalg.inv(A)))
```

IPython console: The output from the IPython console is:

```
In [50]: runfile('/Users/justinwu/Desktop/python/example/Desktop/python/example')
[[1 3 5]
 [2 5 1]
 [2 3 8]]
[[-1.48  0.36  0.88]
 [ 0.56  0.08 -0.36]
 [ 0.16 -0.12  0.04]]
[[ 1.0000000e+00 -1.11022302e-16 -6.24500451e-17]
 [ 3.05311332e-16  1.0000000e+00  1.87350135e-16]
 [ 2.22044605e-16 -1.11022302e-16  1.0000000e+00]]

In [51]:
```

行列式的值

$$|\mathbf{A}| = \sum_j (-1)^{i+j} a_{ij} M_{ij}.$$

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}$$

$$\begin{aligned} |\mathbf{A}| &= 1 \begin{vmatrix} 5 & 1 \\ 3 & 8 \end{vmatrix} - 3 \begin{vmatrix} 2 & 1 \\ 2 & 8 \end{vmatrix} + 5 \begin{vmatrix} 2 & 5 \\ 2 & 3 \end{vmatrix} \\ &= 1(5 \cdot 8 - 3 \cdot 1) - 3(2 \cdot 8 - 2 \cdot 1) + 5(2 \cdot 3 - 2 \cdot 5) = -25. \end{aligned}$$

行列式的值

The screenshot shows a Jupyter Notebook interface with a code editor and a console. The code editor contains a Python script named `linalg_det.py` with the following content:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 15:56:48 2017
5
6@author: justinwu
7"""

8
9import numpy as np
10from scipy import linalg as la
11A = np.array([[1,2],[3,4]])
12print(A)
13print(la.det(A))
```

The console output shows the execution of two cells:

In [53]:
[[1 2]
 [3 4]]
-2.0

In [54]:

Scipy使用linalg.eig()得到A的 eigenvalues和eigenvalues

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & 2 \\ 2 & 4 & 1 \\ 3 & 6 & 2 \end{bmatrix}.$$

$$\begin{aligned} |\mathbf{A} - \lambda \mathbf{I}| &= (1 - \lambda) [(4 - \lambda)(2 - \lambda) - 6] - \\ &\quad 5 [2(2 - \lambda) - 3] + 2 [12 - 3(4 - \lambda)] \\ &= -\lambda^3 + 7\lambda^2 + 8\lambda - 3. \end{aligned}$$

$$\begin{aligned} \lambda_1 &= 7.9579 \\ \lambda_2 &= -1.2577 \\ \lambda_3 &= 0.2997. \end{aligned}$$

EigenValue和EigenVector

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is an 'Editor' containing Python code for calculating Eigenvalues and Eigenvectors. The right pane is a 'Console' showing the execution of the code and its output.

Code in Editor:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 16:07:30 2017
5
6@author: justinwu
7"""
8
9import numpy as np
10from scipy import linalg as la
11A= np.array([[1,5,2],[2,4,1],[3,6,2]])
12lna,v=linalg.eig(A)
13l1,l2,l3 =lna
14#Eigenvalue
15print(l1,l2,l3)
16print('----')
17#Eigenvector
18print(v)
19print('----')
20print(v[:,0])
21print(v[:,1])
22print(v[:,2])
23v1 = np.array(v[:,0]).T
24print('-----')
25print(v1)
26print(linalg.norm(A.dot(v1)-l1*v1))
27
```

Output in Console:

```
In [63]: runfile('/Users/justinwu/Desktop/python/example/eigenvalues.py')
(7.95791620491+0j) (-1.25766470568+0j) (0.290473770378+0j)
-----
[[-0.5297175 -0.90730751  0.28380519]
 [-0.44941741  0.28662547 -0.39012063]
 [-0.71932146  0.30763439  0.87593408]]
-----
[-0.5297175 -0.44941741 -0.71932146]
[-0.90730751  0.28662547  0.30763439]
[ 0.28380519 -0.39012063  0.87593408]
-----
[-0.5297175 -0.44941741 -0.71932146]
3.233018248352212e-15

In [64]:
```

4.numpy模組建立矩陣

- numpy模組建立矩陣
- 矩陣運算
- reshape()改變陣列的長寬
- np.zeros((10,3))產生 10×3 的矩陣

numpy模組建立矩陣

Numpy 提供了基本的建構區塊，如向量，矩陣和對它們的操作，Scipy使用那些一般的建構區塊來做特定的運算。

numpy模組建立矩陣

- import numpy as np # 建立一個 np array 陣列
- myarr = np.array([[1,2,3],[4,5,6]])
- print(myarr) # 顯示 [[1,2,3],[4,5,6]] 的 2*3 矩陣 # 建立一個 全為 0 的 3*5 矩陣，並指定為
• 整數 int16 格式
- my_zero_arr = np.zeros((3,5), dtype=np.int16)
- print(my_zero_arr)
- # 顯示 [[0 0 0 0 0] [0 0 0 0 0] [0 0 0 0 0]] 的 3*5 矩陣
- # 建立一個長度為 15、數值從 0~14、格式為 int64 的一維陣列
- my_onedim_arr = np.arange(15, dtype=np.int64)
- print(my_onedim_arr) # 顯示 [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

numpy模組建立矩陣

- import numpy as np # 建立一個數值從0~14、格式為int64， 3*5 的矩陣
- my_reshape_arr = np.arange(15, dtype=np.int64).reshape((3,5))
- print(my_reshape_arr) #顯示 [[0 1 2 3 4] [5 6 7 8 9][10 11 12 13 14]] 的 3*5矩陣
- # 建立一個最小值1，最大值100，共分8塊相等間距的矩陣
- my_slice_arr = np.linspace(1,100,8)
- print(my_slice_arr) #顯示[1. 15.14285714 29.28571429 43.42857143
57.57142857
- # 71.71428571 85.85714286 100.] 的 1*8 矩陣

numpy模組建立矩陣

- # 建立一個隨機在0~1之間分布的 2*3 矩陣
- my_random_arr = np.random.random((2, 3))
- print(my_random_arr)
- #顯示 [[0.2687965 0.38765465 0.48728548]
• # [0.33212729 0.92145982 0.87586671]]

矩陣運算

- import numpy as np
- a = np.arange(25) # 一維陣列從0到24
- a = a.reshape((5,5)) # 改成5*5的陣列
- b = np.arange(25)
- b = b.reshape((5,5))
- print(a + b)
- print(a - b)
- print(a * b)

矩陣運算

- `print(a / b)`
- `print(a **2)`
- `print(a < b)`
- `print(a > b)`
- `print(a.dot(b))`#a和b陣列作內積

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 14:32:00 2017
5
6@author: justinwu
7"""
8import numpy as np
9a = np.arange(25)
10a = a.reshape((5,5))
11b = np.arange(25)
12b = b.reshape((5,5))
13print(a + b)
14print(a - b)
15print(a * b)
16print(a / b)
17print(a **2)
18print(a < b)
19print(a > b)
20print(a.dot(b))
21
22
```

```
In [13]: runfile('/Users/justinwu/Desktop/test.py')
[[ 0  2  4  6  8]
 [10 12 14 16 18]
 [20 22 24 26 28]
 [30 32 34 36 38]
 [40 42 44 46 48]]
[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]
[[ 0   1   4   9  16]
 [ 25  36  49  64  81]
 [100 121 144 169 196]
 [225 256 289 324 361]
 [400 441 484 529 576]]
[[ nan   1.   1.   1.   1.]
 [  1.   1.   1.   1.   1.]
 [  1.   1.   1.   1.   1.]
 [  1.   1.   1.   1.   1.]
 [  1.   1.   1.   1.   1.]]
[[ 0   1   4   9  16]
 [ 25  36  49  64  81]
 [100 121 144 169 196]
 [225 256 289 324 361]
 [400 441 484 529 576]]
[[False False False False False]
 [False False False False False]]
[[False False False False False]
 [False False False False False]]
[[False False False False False]
 [False False False False False]]
[[ 150  160  170  180  190]
 [ 400  435  470  505  540]
 [ 650  710  770  830  890]
 [ 900  985 1070 1155 1240]]
```

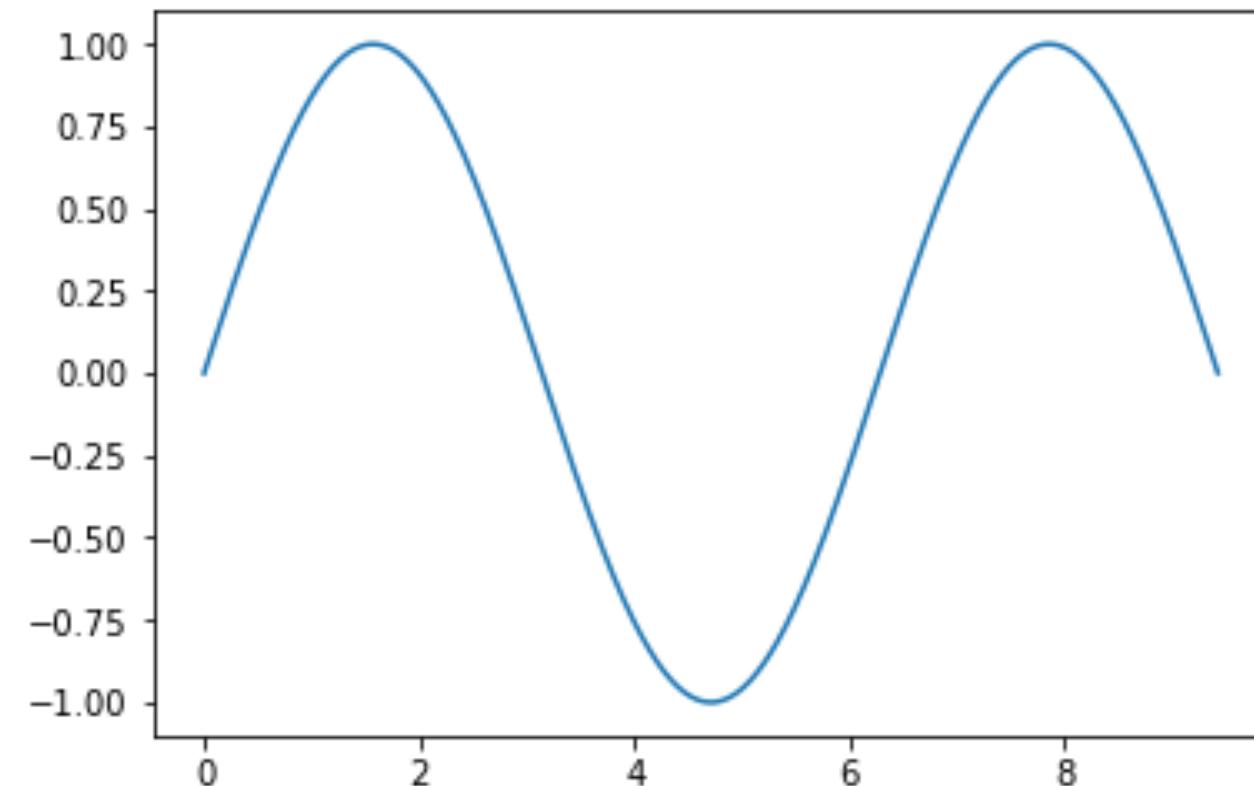
* 印出 no matches ⌂ ⌂ ⌂ Aa ⌂ ⌂
Replace with: 顯示 ⌂ Replace/find next ⌂ Replace selection ⌂ Replace all

numpy模組的sin()函數

- import numpy as np
- import matplotlib.pyplot as plt
- x = np.arange(0, 3 * np.pi, 0.1)#橫坐標為0到3*np.pi,每0.1選一點
- y = np.sin(x)#縱座標為sin()
- # 使用matplotlib畫圖
- plt.plot(x, y)
- plt.show() # 讓圖顯示

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 14:44:56 2017
5
6@author: justinwu
7
8
9import numpy as np
10import matplotlib.pyplot as plt
11
12x = np.arange(0, 3 * np.pi, 0.001)
13y = np.sin(x)
14
15
16plt.plot(x, y)
17plt.show()
18
```

In [41]: runfile('/Users/justinwu/Desktop/test.py')

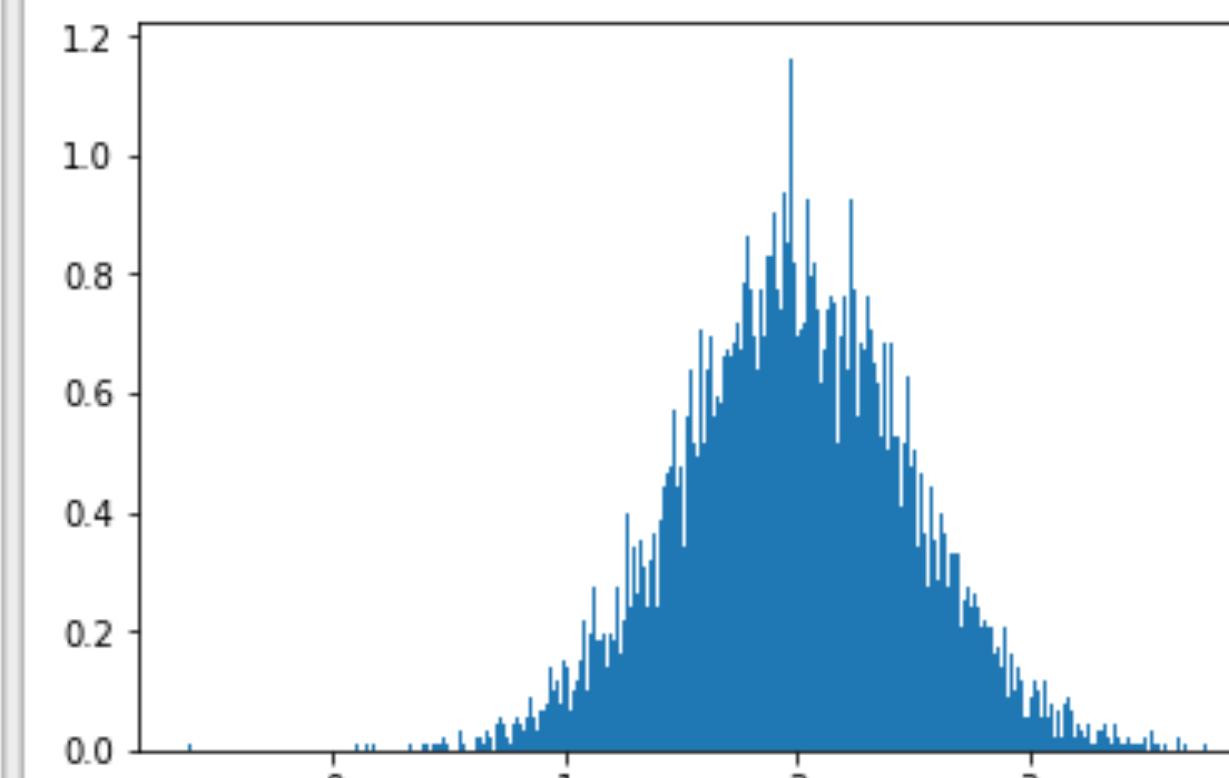


In [42]:

Editor - /Users/justinwu/Desktop/python/example/histograms.py

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 15:22:15 2017
5
6@author: justinwu
7"""
8
9import numpy as np
10import matplotlib.pyplot as plt
11# 建立一個 10000 normal 平均值為 2, 0.5^2 標準偏差
12mu, sigma = 2, 0.5
13v = np.random.normal(mu,sigma,10000)
14# 畫500柱狀histogram圖
15plt.hist(v, bins=500, normed=1)      # matplotlib version 2.0.0
16plt.show()
```

In [47]: runfile('/Users/justinwu/



In [48]:

reshape()改變陣列的長寬

`numpy.reshape(a, newshape, order='C')`

`reshape()`將會將陣列a改成新的維度

參數a為輸入陣列

參數newshape可以是數組或整數,如果回傳整數,則會
將a改為一維陣列

np.zeros((10,3))產生10*3的矩陣

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 10:26:04 2017
5
6@author: justinwu
7"""
8import numpy as np
9
10a = np.zeros((10, 3))
11print(a)
12print('-----')
13b = a.T
14print(b)
15print('-----')
16c = np.reshape(b, (5,6))
17print(d)
```

```
In [51]: runfile('/Users/justinwu/Desktop/p
reshape.py', wdir='/Users/justinwu/Desktop/')

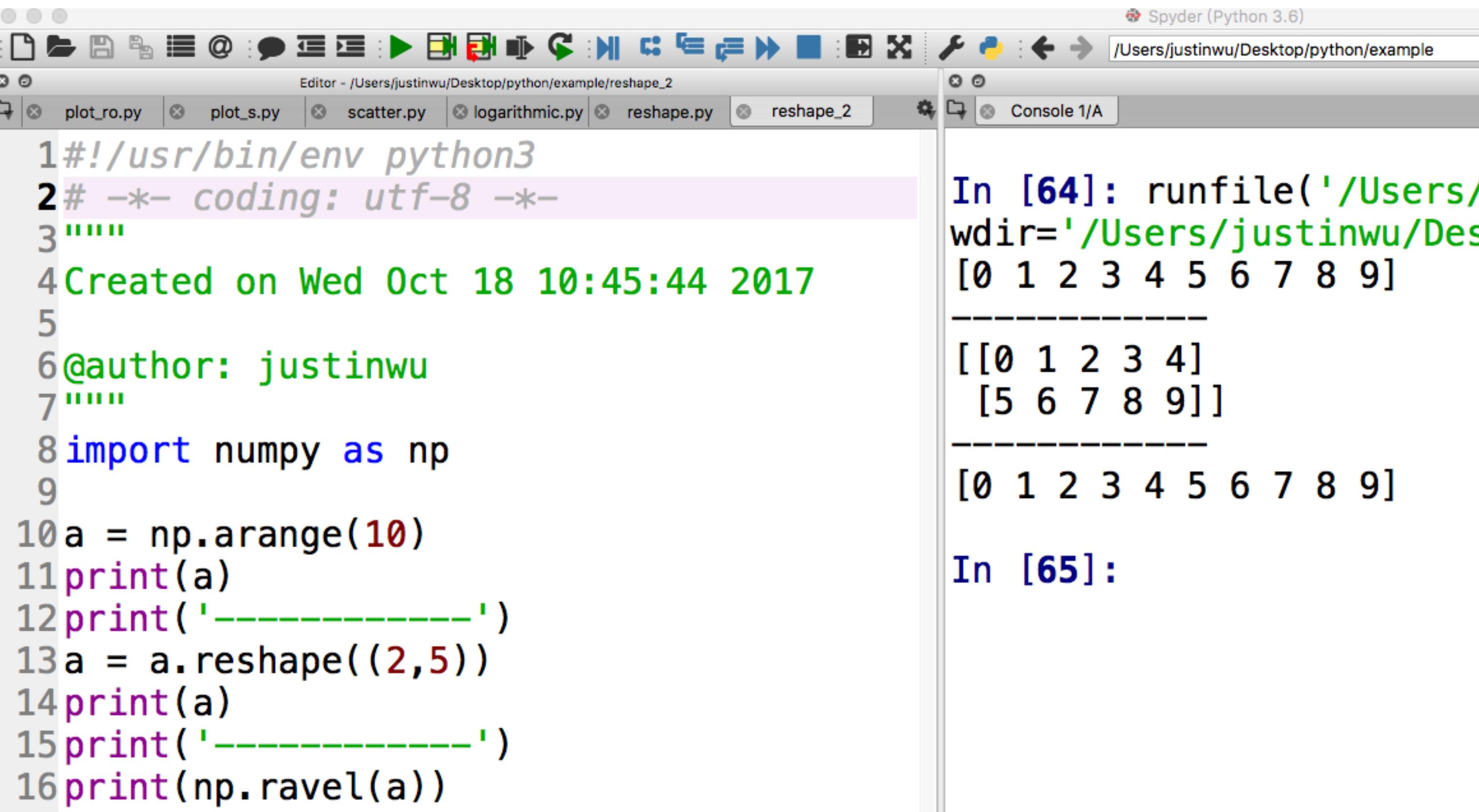
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]

-----
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.]]

-----
[[ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]]
```

np.arange(10)產生一維0-9陣列

np.ravel(a)將傳回a的一維陣列



The screenshot shows the Spyder IDE interface with two panes. The left pane is the Editor, displaying a Python script named 'reshape_2'. The right pane is the Console, showing the output of the code execution.

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 10:45:44 2017
5
6@author: justinwu
7"""
8import numpy as np
9
10a = np.arange(10)
11print(a)
12print('-----')
13a = a.reshape((2,5))
14print(a)
15print('-----')
16print(np.ravel(a))
```

In [64]: runfile('/Users/justinwu/Desktop/python/example/reshape_2', wdir='/Users/justinwu/Desktop/python/example')
[0 1 2 3 4 5 6 7 8 9]

[[0 1 2 3 4]
 [5 6 7 8 9]]

[0 1 2 3 4 5 6 7 8 9]

In [65]:

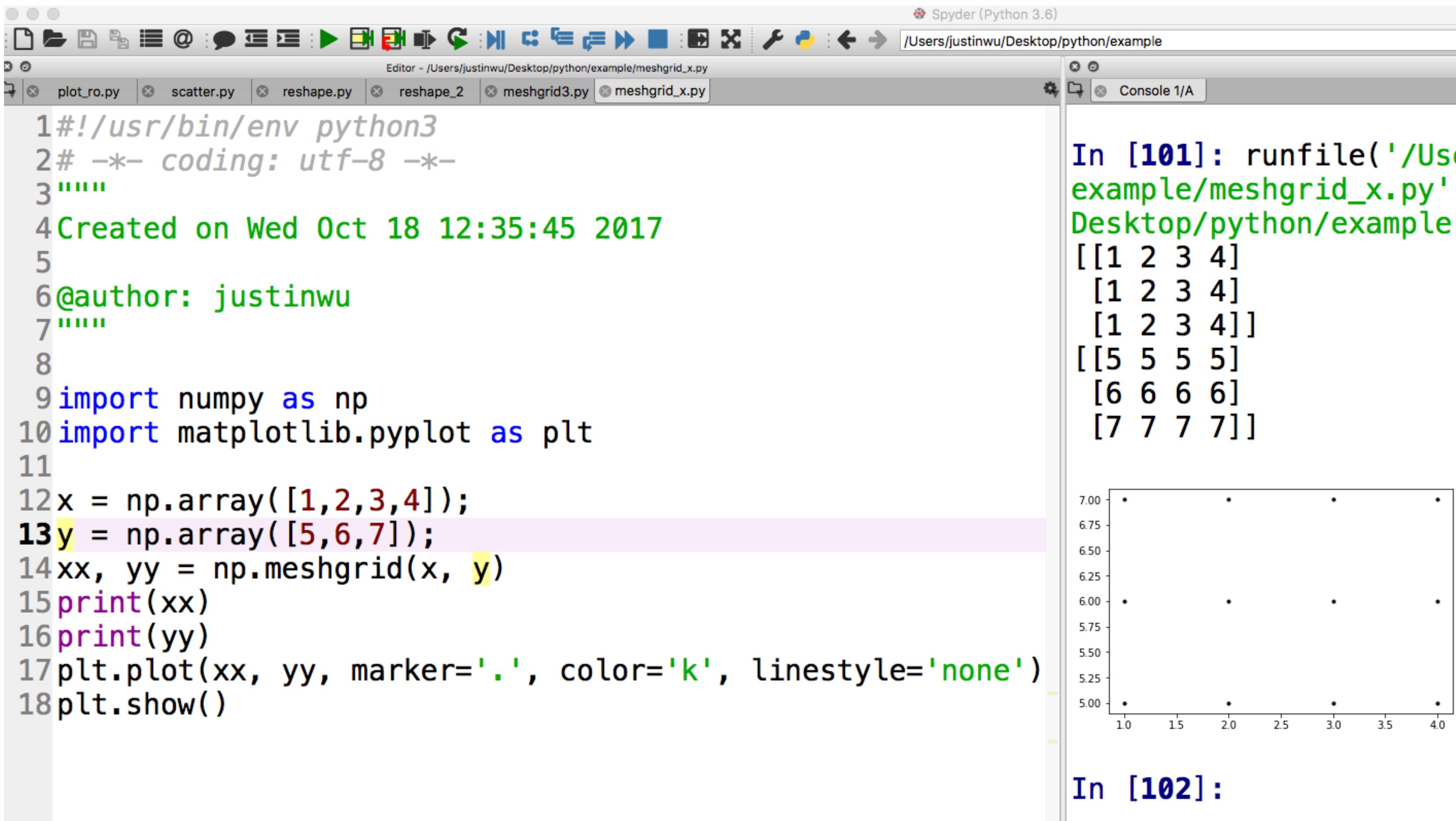
- # meshgrid()函數

- $\text{XX}, \text{YY} = \text{meshgrid}(x, y)$ 函數經常用在網格grid
- #輸入一維矩陣x和一維矩陣y到 $\text{meshgrid}(x, y)$
- #回傳($N_1, N_2, N_3, \dots, N_n$)維度的陣列, $N_i = \text{len}(x_i)$
- 從座標向量得到座標維度,x是X軸座標陣列,y是y軸座標陣列,xx和yy是 $\text{meshgrid}()$ 回傳陣列

```
XX, YY = meshgrid(x, y)

    7          1 7  2 7  3 7  4 7          1 2 3 4  7 7 7 7
y 6 >> 1 6  2 6  3 6  4 6  >> 1 2 3 4  6 6 6 6
    5          1 5  2 5  3 5  4 5          1 2 3 4  5 5 5 5
    1 2 3 4          XX          YY
    x
```

- 從座標向量得到座標維度,x是X軸座標陣列,y是y軸座標陣列,xx和yy是meshgrid()回傳陣列



The screenshot shows the Spyder Python IDE interface. The left pane is the code editor with the file `meshgrid_x.py` open. The right pane is the console output.

```

1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 12:35:45 2017
5
6@author: justinwu
7"""
8
9import numpy as np
10import matplotlib.pyplot as plt
11
12x = np.array([1,2,3,4]);
13y = np.array([5,6,7]);
14xx, yy = np.meshgrid(x, y)
15print(xx)
16print(yy)
17plt.plot(xx, yy, marker='.', color='k', linestyle='none')
18plt.show()

```

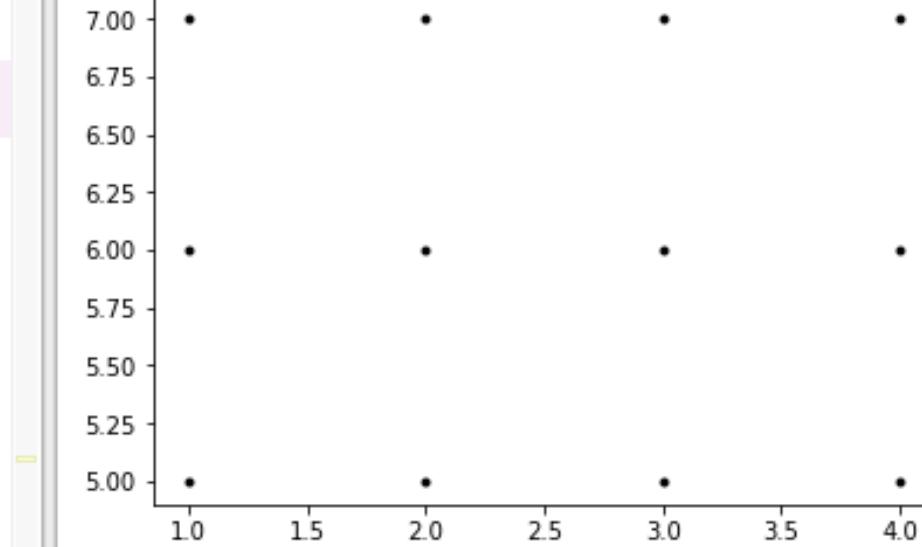
In [101]:

```

[[1 2 3 4]
 [1 2 3 4]
 [1 2 3 4]]
[[5 5 5 5]
 [6 6 6 6]
 [7 7 7 7]]

```

In [102]:



A scatter plot with x-axis from 1.0 to 4.0 and y-axis from 5.00 to 7.00. It displays two sets of points: (1,5), (2,6), (3,7) and their transpose (5,1), (6,2), (7,3).

meshgrid()函數

- meshgrid()函數經常用在網格grid
- #建立0到1三等份的陣列
- `x= np.linspace(0,1,nx)`
- #建立0到1二等份的陣列
- `y= np.linspace(0,1,ny)`
- #輸入一維矩陣x和一維矩陣y到`meshgrid(x,y)`
- #回傳($N_1, N_2, N_3, \dots, N_n$)維度的陣列, $N_i = \text{len}(x_i)$
- `xv,yv = np.meshgrid(x,y)`

Spyder (Python 3.6)

Editor - /Users/justinwu/Desktop/python/example/meshgrid.py

plot_s.py scatter.py logarithmic.py reshape.py reshape_2 meshgrid.py Console 1/A

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 11:16:09 2017
5
6@author: justinwu
7"""
8
9import numpy as np
10
11nx,ny=(3,2)
12#建立0到1三等份的陣列
13x= np.linspace(0,1,nx)
14print(x)
15#建立0到1二等份的陣列
16y= np.linspace(0,1,ny)
17print(y)
18print('-----')
19#輸入一維矩陣x和一為矩陣y到meshgrid(x,y)
20#回傳(N1,N2,N3,...,Nn)維度的陣列, Ni=len(xi)
21xv,yv = np.meshgrid(x,y)
22print(xv)
23print(yv)
```

In [69]: runfile('/Users/justinwu/Desktop/python/example/meshgrid.py', wdir='/Users/justinwu/Desktop/python/example')

[0. 0.5 1.]
[0. 1.]

[[0. 0.5 1.]
 [0. 0.5 1.]]
[[0. 0. 0.]
 [1. 1. 1.]]

In [70]:

matplotlib.pyplot.contourf()是畫等高線圖

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a Python script named `meshgrid2.py` with the following content:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 11:47:23 2017
5
6@author: justinwu
7"""
8import numpy as np
9import matplotlib.pyplot as plt
10x = np.arange(-5, 5, 0.1)
11y = np.arange(-5, 5, 0.1)
12xx, yy = np.meshgrid(x, y, sparse=True)
13z = np.sin(xx**2 + yy**2) / (xx**2 + yy**2)
14h = plt.contourf(x,y,z)
```

On the right, the IPython console window shows the command `In [80]: runfile('/User/meshgrid2.py', wdir='/U')` and the resulting contour plot. The plot displays concentric circular contours centered at the origin (0,0), with colors ranging from dark blue to yellow, indicating varying values of the function $\frac{\sin(r^2)}{r^2}$.

5.Pandas資料結構

- 使用pip install安裝pandas
- Pandas讀取csv
- 讀取MI_INDEX.csv檔,big5編碼
- Series序列資料
- pandas的DataFrame
- 資料選擇與篩選
- lambda可調式參數

Pandas資料結構

Pandas 是 python 的一個數據分析 函式庫，提供高效能、簡易使用的資料格式(Data Frame)讓使用者可以快速操作及分析資料

Pandas

載入至 Pandas 的資料結構物件，可以使用結構化物件所提供的方法，來快速地進行資料的前處理，如資料補值，空值去除或取代等。

使用pip install安裝pandas

- pip install pandas

Pandas讀取csv

- pandas使用read_csv()方法來讀取csv資料,這是讀取加州大學爾灣分校的iris資料
- import pandas as pd
- df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', header=None)
- print(df.tail())

讀取MI_INDEX.csv檔,big5編碼

- import pandas as pd
- #預設utf-8編碼
- df = pd.read_csv('MI_INDEX.csv', encoding='big5')
- print(df)

這是執行的結果



The screenshot shows the Spyder Python 3.6 IDE interface. The top bar displays the title "Spyder (Python 3.6)" and the path "/Users/justinwu/Desktop/python/example". Below the title bar, there are tabs for "Editor - /Users/justinwu/Desktop/python/example/read.py" and "Console 2/A". The main area is divided into two panes. The left pane shows the Python code, and the right pane shows the console output.

```
1#!/usr/bin/
2# -*- coding: utf-8 -*-
3"""
4Created on
5
6@author: justinwu
7"""
8
9
10import pandas as pd
11url='https://www.twse.com.tw/exchangeReport/STOCK_DAY?response=json&date=20230101'
12#預設utf-8編碼
13df = pd.read_json(url)
14print(df)
```

In [8]: runfile('/Users/justinwu/Desktop/python/example/read.py', wdir='/Users/justinwu/Desktop/python/example')

| | 指數名稱 | 最新價 | 變動 | 漲跌 | 漲跌幅 | Unnamed: 5 |
|----|-------------|-----------|---------|------|-------|------------|
| 0 | 寶島股價指數 | 12,374.17 | + 55.24 | 0.45 | 0.45% | NaN |
| 1 | 發行量加權股價指數 | 10,774.21 | + 50.12 | 0.47 | 0.47% | NaN |
| 2 | 臺灣公司治理100指數 | 6,066.00 | + 24.59 | 0.41 | 0.41% | NaN |
| 3 | 臺灣50指數 | 8,227.46 | + 40.92 | 0.50 | 0.50% | NaN |
| 4 | 臺灣中型100指數 | 7,746.33 | + 42.92 | 0.56 | 0.56% | NaN |
| 5 | 臺灣資訊科技指數 | 10,581.84 | + 49.70 | 0.47 | 0.47% | NaN |
| 6 | 臺灣發達指數 | 9,016.34 | + 59.92 | 0.67 | 0.67% | NaN |
| 7 | 臺灣高股息指數 | 5,917.61 | + 35.93 | 0.61 | 0.61% | NaN |
| 8 | 臺灣就業99指數 | 6,240.56 | + 22.47 | 0.36 | 0.36% | NaN |
| 9 | 臺灣高薪100指數 | 5,951.88 | + 36.26 | 0.61 | 0.61% | NaN |
| 10 | 臺灣低波動高股息指數 | 1,524.89 | + 11.05 | 0.73 | 0.73% | NaN |
| 11 | 未含金融保險股指數 | 9,216.93 | + 47.03 | 0.51 | 0.51% | NaN |
| 12 | 未含電子股指數 | 13,943.04 | + 35.55 | 0.26 | 0.26% | NaN |
| 13 | 未含金融電子股指數 | 11,978.63 | + 35.87 | 0.30 | 0.30% | NaN |
| 14 | 小型股300指數 | 6,284.23 | + 20.73 | 0.33 | 0.33% | NaN |

Pandas資料結構

Pandas 提供兩種主要的資料結構，序列Series 和 DataFrame。

序列Series 就是用來處理像時間序列相關的資料，主要為建立索引的一維陣列。

DataFrame 則是用來處理結構化的資料像資料表，有列索引與欄標籤的二維資料集，例如關聯式資料庫、CSV 檔等資料。

Series序列資料

- 序列資料可以是陣列Array,字典dictionary和一般資料
- import pandas as pd
- names = ['小寶','小文','阿呆','John','Lin']
- myarray = pd.Series(names)
- print(myarray)

字典

- import pandas as pd
- SalarysDic ={"John":'50000',
• "Mary":'20000',
• "Ivy":'90000',
• "Steve":'35000',
• "David":'85000'
• }
• myDic = pd.Series(SalarysDic,index=SalarysDic.keys())
• print(myDic[0])
• print(myDic['Ivy'])
• print(myDic[[0,1,3]])
• print(myDic[['John','Ivy','David']])

字典,索引和所對應的值

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 11:40:10 2017
5
6@author: justinwu
7"""
8
9import pandas as pd
10
11SalarysDic = {"John":'50000',
12                 "Mary":'20000',
13                 "Ivy":'90000',
14                 "Steve":'35000',
15                 "David":'85000'
16                 }
17myDic = pd.Series(SalarysDic, index=SalarysDic.keys())
18print(myDic[0])
19print(myDic['Ivy'])
20print(myDic[[0,1,3]])
21print(myDic[['John', 'Ivy', 'David']])
22
```

```
Python 3.6.2 |Ana
Sep 20 2017, 05:4
Type "copyright",
information.

IPython 6.1.0 -- 

In [1]: runfile('
example/dictionar
Desktop/python/ex
50000
90000
John      50000
Mary      20000
Steve     35000
dtype: object
John      50000
Ivy       90000
David     85000
dtype: object

In [2]:
```

pandas的DataFrame

```
class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

data:numpy 陣列,字典或*DataFrame*

index:索引

columns:索引

dtype:資料型態,預設沒有強制資料型態

copy:布林值,預設為*False*,是否從輸入拷貝資料

DataFrame 用來處理結構化二維的資料，有列索引與欄標籤的二維資料集，可以透過 字典 或是陣列 Array 來建立，也可以使用外部的CSV或資料庫的資料來讀取後來建立。

輸入的資料放到DataFrame

- import pandas as pd
- df2 = pd.read_csv('MI_INDEX.csv', encoding='big5')
- df = pd.DataFrame(df2)
- print(df.tail())#顯示後面幾列資料
- print(df.shape)#顯示列數與欄數
- print(df.info())#顯示DataFrame

輸入的資料放到DataFrame

- `print(df.describe())`#顯示統計資訊
- `print(df.head())` # 顯示前面幾列資料
- `print(df.index)`#索引資訊
- `print(df.columns)`#顯示欄位
- `print(df.tail().T)`#`df.T`是將陣列轉置transpose

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Oct 17 12:04:04 2017
5
6@author: justinwu
7"""
8import pandas as pd
9
10df2 = pd.read_csv('MI_INDEX.csv', encoding='big5')
11df = pd.DataFrame(df2)
12print(df.tail())
13print(df.shape)
14print(df.info())
15print(df.describe())
16print(df.head())
17print(df.index)
18print(df.columns)
19print(df.tail().T)
20
```

```
Python 3.6.2 |Anaconda custom (x86_64)| (default, Sep 20 2017, 05:42:37)
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.
    寶島股價指數 12,374.17 + 55.24  0.45 Unnamed: 5
113 航運類報酬指數     85.09 + 0.14  0.16      NaN
114 觀光類報酬指數     149.84 - 0.69 -0.46      NaN
115 金融保險類報酬指數 1,609.13 + 2.33  0.15      NaN
116 貿易百貨類報酬指數 285.27 + 0.55  0.19      NaN
117 油電燃氣類報酬指數 177.13 -- --      NaN
(118, 6)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118 entries, 0 to 117
Data columns (total 6 columns):
寶島股價指數          118 non-null object
12,374.17            118 non-null object
+                   118 non-null object
55.24               118 non-null object
0.45                118 non-null object
Unnamed: 5            0 non-null float64
dtypes: float64(1), object(5)
memory usage: 5.6+ KB
None
    Unnamed: 5
count            0.0
mean             NaN
std              NaN
min              NaN
25%              NaN
50%              NaN
75%              NaN
max              NaN
    寶島股價指數 12,374.17 + 55.24  0.45 Unnamed: 5
0   發行量加權股價指數 10,774.21 + 50.12  0.47      NaN
1   臺灣公司治理100指數 6,066.00 + 24.59  0.41      NaN
2   臺灣50指數     8,227.46 + 40.92  0.50      NaN
3   臺灣中型100指數 7,746.33 + 42.92  0.56      NaN
4   臺灣資訊科技指數 10,581.84 + 49.70  0.47      NaN
RangeIndex(start=0, stop=118, step=1)
Index(['寶島股價指數', '12,374.17', '+', '55.24', '0.45', 'Unnamed: 5'], dtype='object')
    113   114   115   116   117
寶島股價指數  航運類報酬指數  觀光類報酬指數  金融保險類報酬指數  貿易百貨類報酬指數  油電燃氣類報酬指數
12,374.17    85.09   149.84   1,609.13    285.27   177.13
+           +       -           +       + 
55.24        0.14   0.69     2.33     0.55      --
0.45        0.16   -0.46    0.15     0.19      --
Unnamed: 5    NaN    NaN     NaN     NaN     NaN     NaN
```

In [2]:

In [2]:

資料選擇與篩選

- 可以透過下列方法選擇元素
- 中括號[]選擇元素
- 將變數當作屬性選擇
- .loc .iloc 方法選擇
- 使用布林值篩選

使用iloc[]來選取DataFrame()資料

```
1 import pandas as pd
2
3 groups = ["Working", "Dancing", "cooking",
4             "Movies", "Sports", "Fishing"]
5 num = [12, 15, 3, 8, 29, 38]
6
7 dict = {"groups": groups,
8          "num": num
9          }
10
11 mydf = pd.DataFrame(dict)
12 print(mydf.iloc[:, :]) # 所有列和欄
13 print("----")
14 print(mydf.iloc[0, 1]) # 第一列第二欄：組的人數
15 print("----")
16 print(mydf.iloc[0:2, :]) # 第一列和第二列：組的組名與人數
17 print("----")
18 print(mydf.iloc[:, 0]) # 第一欄：各組的人數
19 print("----")
20 print(mydf["num"]) # 各組的人數
21 print("----")
22 print(mydf.num) # 各組的人數
```

| | groups | num |
|---|---------|-----|
| 0 | Working | 12 |
| 1 | Dancing | 15 |
| 2 | cooking | 3 |
| 3 | Movies | 8 |
| 4 | Sports | 29 |
| 5 | Fishing | 38 |

12

| | groups | num |
|---|---------|-----|
| 0 | Working | 12 |
| 1 | Dancing | 15 |

| | Working | Dancing | cooking | Movies | Sports | Fishing |
|---|---------|---------|---------|--------|--------|---------|
| 0 | Working | | | | | |
| 1 | | Dancing | | | | |
| 2 | | | cooking | | | |
| 3 | | | | Movies | | |
| 4 | | | | | Sports | |
| 5 | | | | | | Fishing |

Name: groups, dtype: object

CALLABLE可調式參數lambda

```
In [1]: import pandas as pd  
import numpy as np  
df1 = pd.DataFrame(np.random.randn(6, 4),  
                   index=list('abcdef'), columns=list('ABCD'))
```

columns為欄('ABCD')

```
In [2]: df1
```

```
Out[2]:
```

| | A | B | C | D |
|---|-----------|-----------|-----------|-----------|
| a | 1.282479 | 0.946747 | -1.196755 | -1.568831 |
| b | 0.960616 | 1.148967 | 0.289785 | 0.338350 |
| c | -0.812031 | -0.408394 | 1.036876 | 1.018705 |
| d | -0.076554 | -1.116968 | 0.190509 | 1.032476 |
| e | -0.560559 | -2.004797 | -1.732845 | -0.403581 |
| f | -1.314003 | -0.156767 | -1.405891 | -2.684453 |

lambda可調式參數

df:df.A>0的列數

```
In [3]: df1.loc[lambda df: df.A > 0, :]
```

Out[3]:

| | A | B | C | D |
|---|----------|----------|-----------|-----------|
| a | 1.282479 | 0.946747 | -1.196755 | -1.568831 |
| b | 0.960616 | 1.148967 | 0.289785 | 0.338350 |

CALLABLE可調式參數lambda

```
In [4]: df1.loc[:, lambda df: ['A', 'B']]
```

Out[4]:

| | A | B |
|---|-----------|-----------|
| a | 1.282479 | 0.946747 |
| b | 0.960616 | 1.148967 |
| c | -0.812031 | -0.408394 |
| d | -0.076554 | -1.116968 |
| e | -0.560559 | -2.004797 |
| f | -1.314003 | -0.156767 |

顯示'A'欄和'B'欄

```
In [5]: df1.iloc[:, lambda df: [0, 1]]
```

Out[5]:

| | A | B |
|---|-----------|-----------|
| a | 1.282479 | 0.946747 |
| b | 0.960616 | 1.148967 |
| c | -0.812031 | -0.408394 |
| d | -0.076554 | -1.116968 |
| e | -0.560559 | -2.004797 |
| f | -1.314003 | -0.156767 |

iloc[]股價資訊範例

```
In [14]: import numpy as np
import pandas as pd

In [15]: df2 = pd.read_csv('./data/201711_F3_1_8_2330.csv', encoding='big5',
                      error_bad_lines=False, header=None, names=["日期", "成交股數", "成交金額", "開盤價",
                      , "最高價", "最低價", "收盤價", "漲跌價差", "成交筆數"])
print(df2)
df=pd.DataFrame(df2)
print(df.iloc[:,5])
```

讀進來的資料

| | 日期 | 成交股數 | 成交金額 | 開盤價 | 最高價 | 最低價 | 收盤價 | 漲跌價差 | \ |
|-----------|------------|---------------|-------|-------|-------|-------|------|-------|---|
| 106/11/01 | 20,465,054 | 4,975,624,502 | 243.5 | 245.0 | 241.5 | 242.5 | -0.5 | 6,464 | |
| 106/11/02 | 16,309,342 | 3,940,181,714 | 242.0 | 242.5 | 240.5 | 241.0 | -1.5 | 5,912 | |
| 106/11/03 | 18,327,682 | 4,386,122,470 | 240.5 | 241.0 | 238.5 | 239.0 | -2.0 | 5,621 | |
| 106/11/06 | 21,029,515 | 5,063,670,900 | 243.5 | 244.0 | 239.0 | 239.5 | 0.5 | 6,601 | |
| 106/11/07 | 21,689,261 | 5,266,500,464 | 242.0 | 244.0 | 241.5 | 244.0 | 4.5 | 6,729 | |
| 106/11/08 | 14,579,103 | 3,536,455,029 | 243.0 | 243.5 | 242.0 | 242.5 | -1.5 | 4,610 | |
| 106/11/09 | 21,366,949 | 5,151,628,572 | 240.5 | 243.0 | 240.0 | 241.0 | -1.5 | 6,339 | |
| 106/11/10 | 15,061,296 | 3,629,826,936 | 240.0 | 242.0 | 239.5 | 240.5 | -0.5 | 4,465 | |

| | 成交筆數 |
|-----------|-------|
| 106/11/01 | NaN |
| 106/11/02 | NaN |
| 106/11/03 | NaN |
| 106/11/06 | NaN |
| 106/11/07 | NaN |
| 106/11/08 | NaN |
| 106/11/09 | NaN |
| 106/11/10 | NaN |
| 106/11/01 | 242.5 |
| 106/11/02 | 241.0 |
| 106/11/03 | 239.0 |
| 106/11/06 | 239.5 |
| 106/11/07 | 244.0 |
| 106/11/08 | 242.5 |
| 106/11/09 | 241.0 |
| 106/11/10 | 240.5 |

Name: 最低價, dtype: float64

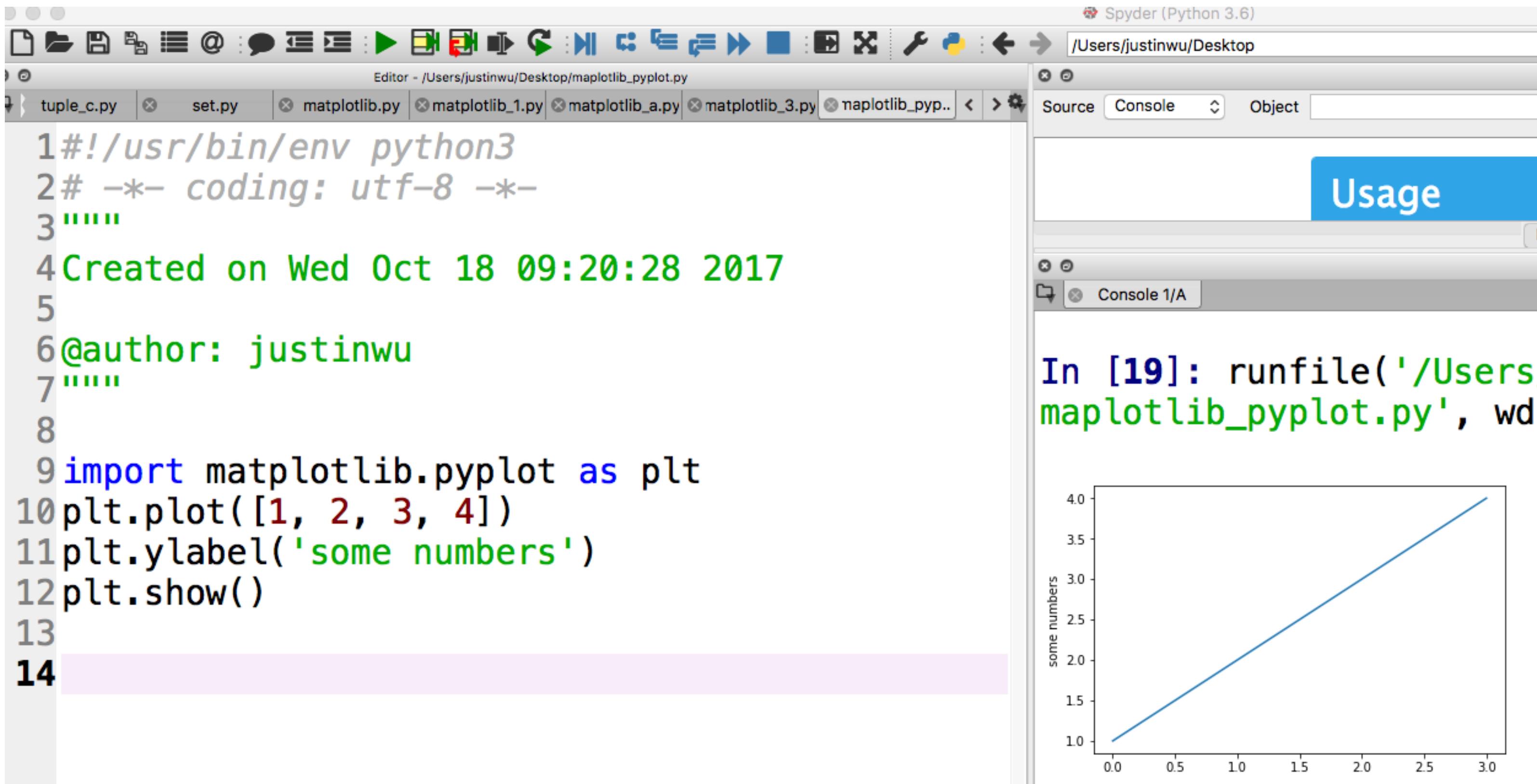
shape維度是二維(8,9)的資料

```
In [16]: print(dfa2.shape)
print(dfa2.info())
(8, 9)
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, 106/11/01 to 106/11/10
Data columns (total 9 columns):
日期      8 non-null object
成交股數    8 non-null object
成交金額    8 non-null float64
開盤價      8 non-null float64
最高價      8 non-null float64
最低價      8 non-null float64
收盤價      8 non-null float64
漲跌價差    8 non-null object
成交筆數    0 non-null float64
dtypes: float64(6), object(3)
memory usage: 640.0+ bytes
None
```

6. Matplotlib 畫圖

- Matplotlib.pyplot是畫圖的命令集合函數.每一個pyplot函數可以建立或修改圖形

`plt.plot([1,2,3,4])`預設是X軸,而Y軸
是我們輸入的資料串列[1,2,3,4].

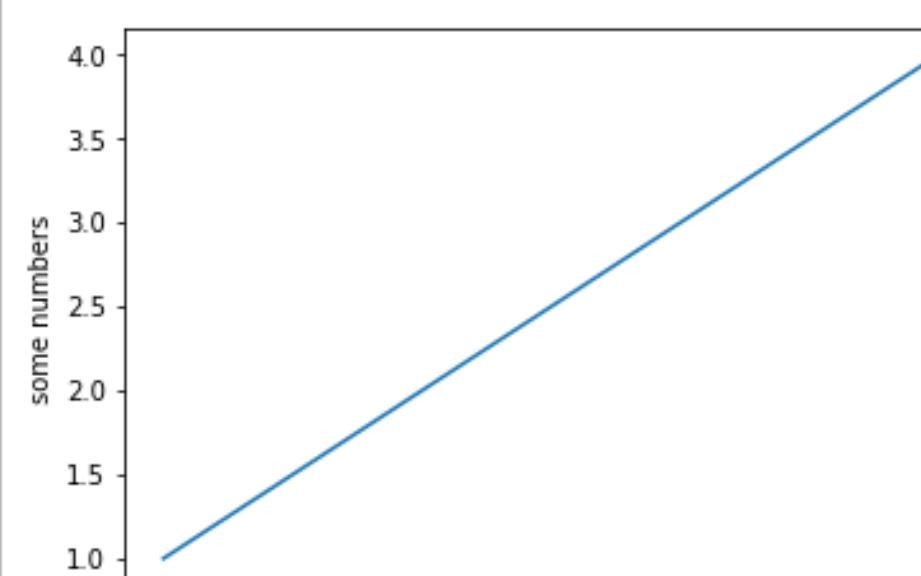


The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a Python script named `matplotlib_pyplot.py`. The code imports `matplotlib.pyplot`, plots the numbers 1, 2, 3, and 4 on the X-axis, and labels the Y-axis as 'some numbers'. The code is annotated with comments and timestamps. On the right, the console window shows the command `In [19]: runfile('/Users/justinwu/Desktop/matplotlib_pyplot.py', wd:)` and the resulting line plot. The plot has a blue line starting at (0, 1) and ending at (3, 4), with the Y-axis labeled 'some numbers' ranging from 1.0 to 4.0 and the X-axis ranging from 0.0 to 3.0.

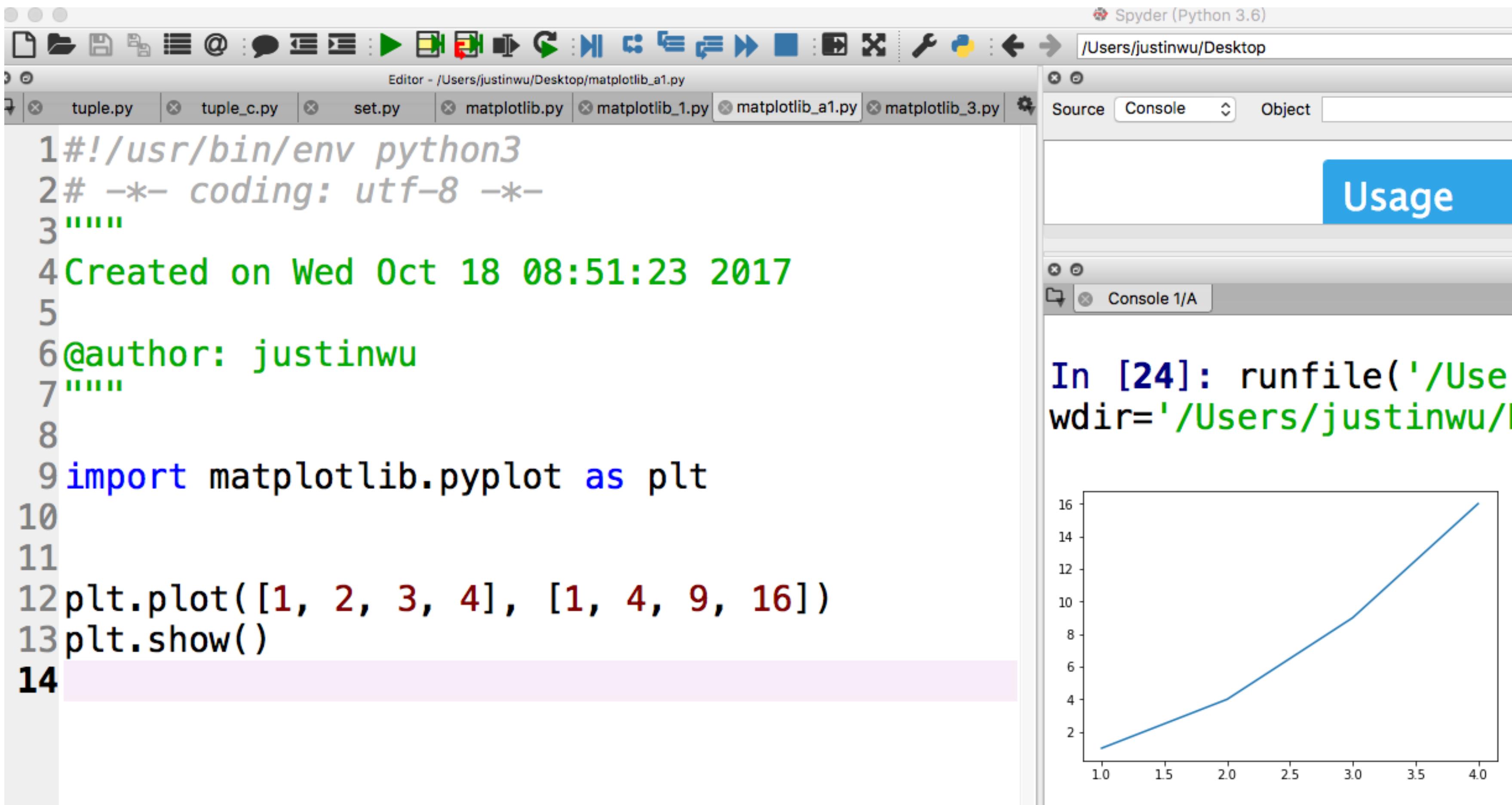
```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 09:20:28 2017
5
6@author: justinwu
7"""
8
9import matplotlib.pyplot as plt
10plt.plot([1, 2, 3, 4])
11plt.ylabel('some numbers')
12plt.show()
13
14
```

Usage

In [19]: runfile('/Users/justinwu/Desktop/matplotlib_pyplot.py', wd:)



第一個[1,2,3,4]參數是X軸,第二個
參數是Y軸



The screenshot shows the Spyder Python IDE interface. The left pane is the code editor with the file `matplotlib_a1.py` open. The code is as follows:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 08:51:23 2017
5
6@author: justinwu
7"""
8
9import matplotlib.pyplot as plt
10
11
12plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
13plt.show()
14
```

The right pane shows the console output and a plot window. The console output includes:

```
In [24]: runfile('/Users/justinwu/Desktop/matplotlib_a1.py', wdir='/Users/justinwu/')
```

The plot window displays a line graph with X-axis values [1, 2, 3, 4] and Y-axis values [1, 4, 9, 16]. The plot area has a light gray background with white grid lines. The X-axis ranges from 1.0 to 4.0 with major ticks every 0.5 units. The Y-axis ranges from 2 to 16 with major ticks every 2 units. A single blue line connects the points (1, 1), (2, 4), (3, 9), and (4, 16).

Matplotlib 畫圖

- import matplotlib.pyplot as plt
- plt.plot([1, 2, 3, 4])
- plt.ylabel('some numbers')
- plt.show()
- print('-----')
- plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
- plt.show()
- print('-----')
- plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
- plt.axis([0, 6, 0, 20])
- plt.show()

plot()第三個參數是格式字串點 plot,'ro'為顯示紅色圓圈

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a file named `plot_ro.py` with the following content:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 09:34:44 2017
5
6@author: justinwu
7"""
8import matplotlib.pyplot as plt
9#第三個參數是格式字串點plot,'ro'為顯示紅色圓圈
10plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
11plt.axis([0, 6, 0, 20])
12plt.show()
```

On the right, the console window shows the output of running the script:

```
In [30]: runfile('/Users/justinwu/Desktop/python/example/plot_ro.py', wdir='/Users/justinwu/Desktop/python/example')
```

A plot window is displayed, showing four red circular markers at the coordinates (1, 1), (2, 4), (3, 9), and (4, 16).

'r--'紅色虛線,'bs'藍色矩形,'g^'綠色三角形

The screenshot shows the Spyder IDE interface with the following details:

- Editor:** The code editor window displays the file `plot_s.py` with the following content:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 09:42:54 2017
5
6@author: justinwu
7"""

8
9import numpy as np
10
11# 0到5每步0.2
12t = np.arange(0., 5., 0.2)
13
14# red dashes, blue squares and green triangles
15# 'r--'紅色虛線,'bs'藍色矩形,'g^'綠色三角形
16plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3,
17plt.show()
```
- Console:** The console window shows the command `In [33]: runfile('/Users/justinwu/Desktop/python/example/plot_s.py', wdir='/Users/justinwu/Desktop/python/example')` and the resulting plot.
- Plot:** The plot shows three data series:
 - A red dashed line (`'r--'`) representing $y = x$.
 - Blue square markers (`'bs'`) representing $y = x^2$.
 - Green triangle markers (`'g^'`) representing $y = x^3$.The x-axis ranges from 0 to 5, and the y-axis ranges from 0 to 100.

scatter()函數#第一個參數是X軸的輸入陣列,第二個
參數是Y軸的輸入陣列
#c參數是設定標記顏色,s參數設定標記大小

```
matplotlib.pyplot.scatter(x, y, s=None,  
c=None, marker=None, cmap=None, norm=None,  
vmin=None, vmax=None, alpha=None,  
linewidths=None, verts=None,  
edgecolors=None, hold=None, data=None,  
**kwargs)
```

numpy.arange(0,50,1)是0到50每步1

The screenshot shows the Spyder Python 3.6 IDE interface. On the left, the code editor displays a script named 'scatter.py' with the following content:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Wed Oct 18 09:49:02 2017
5
6@author: justinwu
7"""

8
9data = {'a': np.arange(50),
10       'c': np.random.randint(0, 50, 50),
11       'd': np.random.randn(50)}
12data['b'] = data['a'] + 10 * np.random.randn(50)
13data['d'] = np.abs(data['d']) * 100
14#第一個參數是X軸的輸入陣列,第二個參數是Y軸的輸入陣列
15#c參數是設定標記顏色,s參數設定標記大小
16#data參數是輸入的資料
17plt.scatter('a', 'b', c='c', s='d', data=data)
18plt.xlabel('entry a')
19plt.ylabel('entry b')
20plt.show()
```

In the center, the console window shows the command runfile and its output:

```
In [35]: runfile('/Users/justinwu/Desktop/python/example/scatter.py', wdir='/Users/justinwu/Desktop/python/example')
```

To the right, a scatter plot titled 'entry a' vs 'entry b' is displayed, showing data points colored by 'c' and sized by 'd'. A tooltip in the top right corner of the IDE window says: "Here you can pressing Cmd on the Editor".

- Thanks.

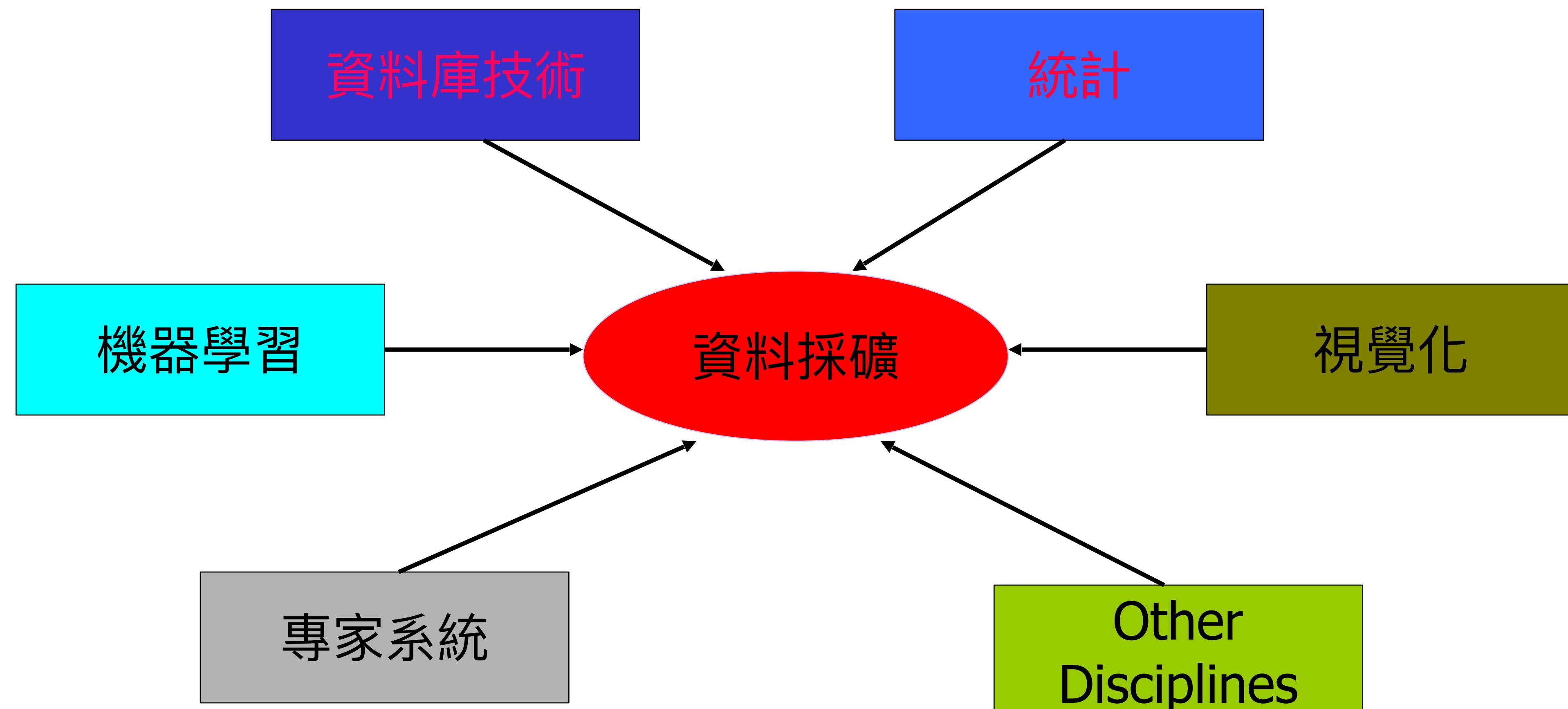


分類與預測



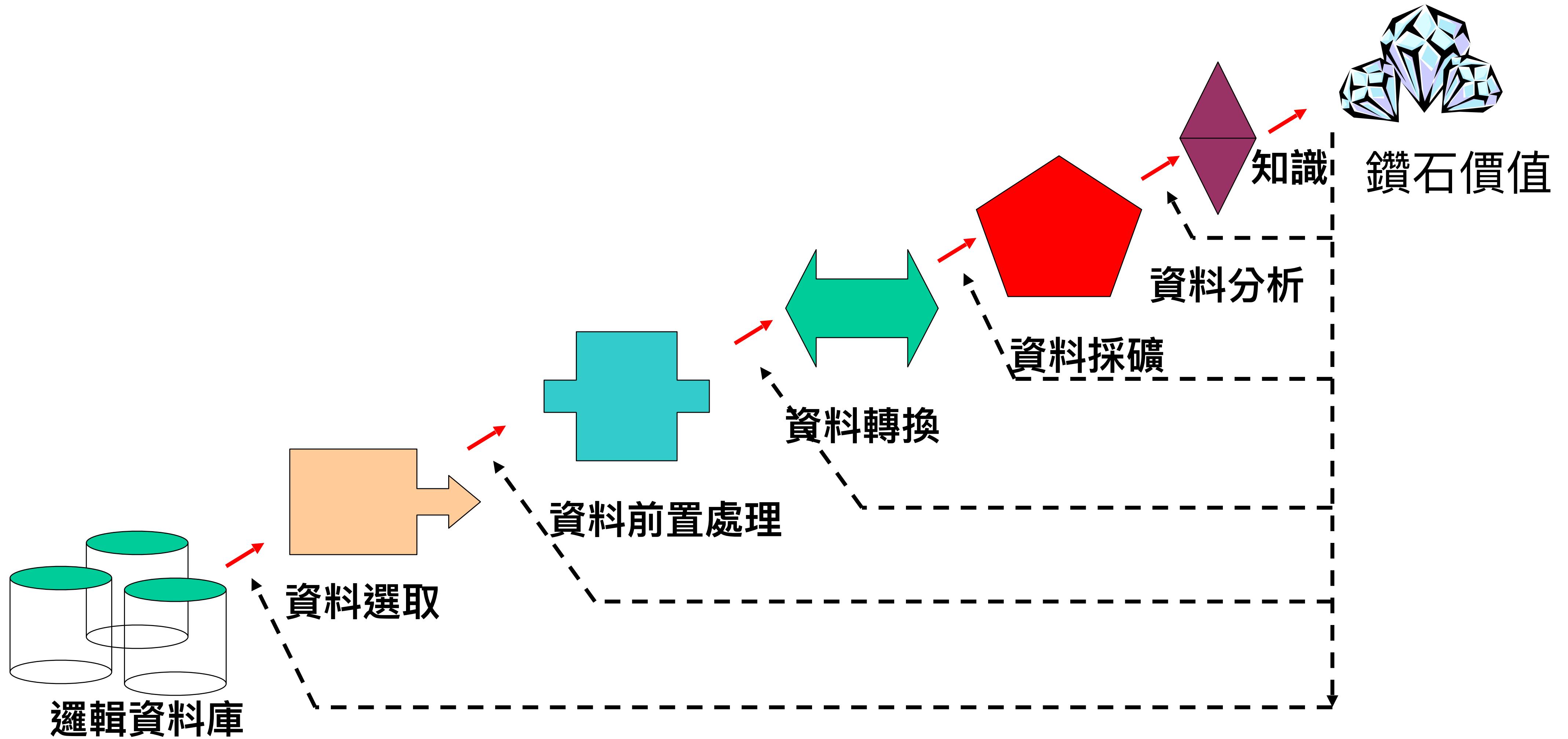
- 1. 資料採礦過程
- 2. 分類與預測
- 3. 決策樹 信用評等範例
- 4. 監督式學習分類
- 5. 天氣決策樹決定是否騎腳踏車
 - 資訊熵Entropy
 - Information Gain資訊增益
 - ID3決策樹演算法
 - 鳶尾花決策樹分類
 - 隨機森林結合強學習
- 6. K-nearest最鄰近分類演算法KNN

1. 資料採礦





資料採礦過程





2. 分類與預測



- **分類:**
 - 預測是哪一類
 - 有標籤的預測分類
- **預測:**
 - 訓練模型後作預測
- **應用**
 - 信用卡評定
 - 目標行銷
 - 醫學診斷

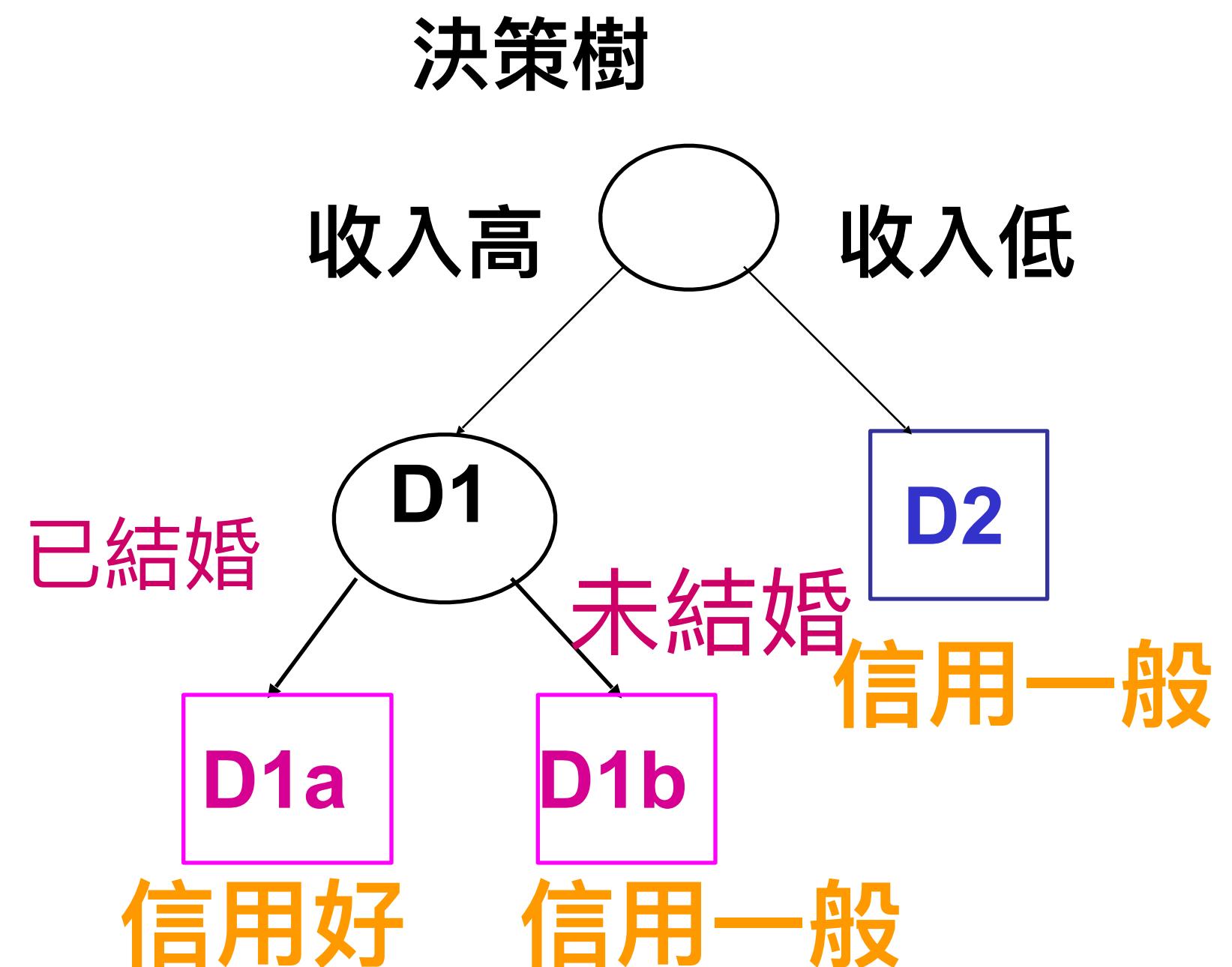


3. 決策樹 信用評等範例

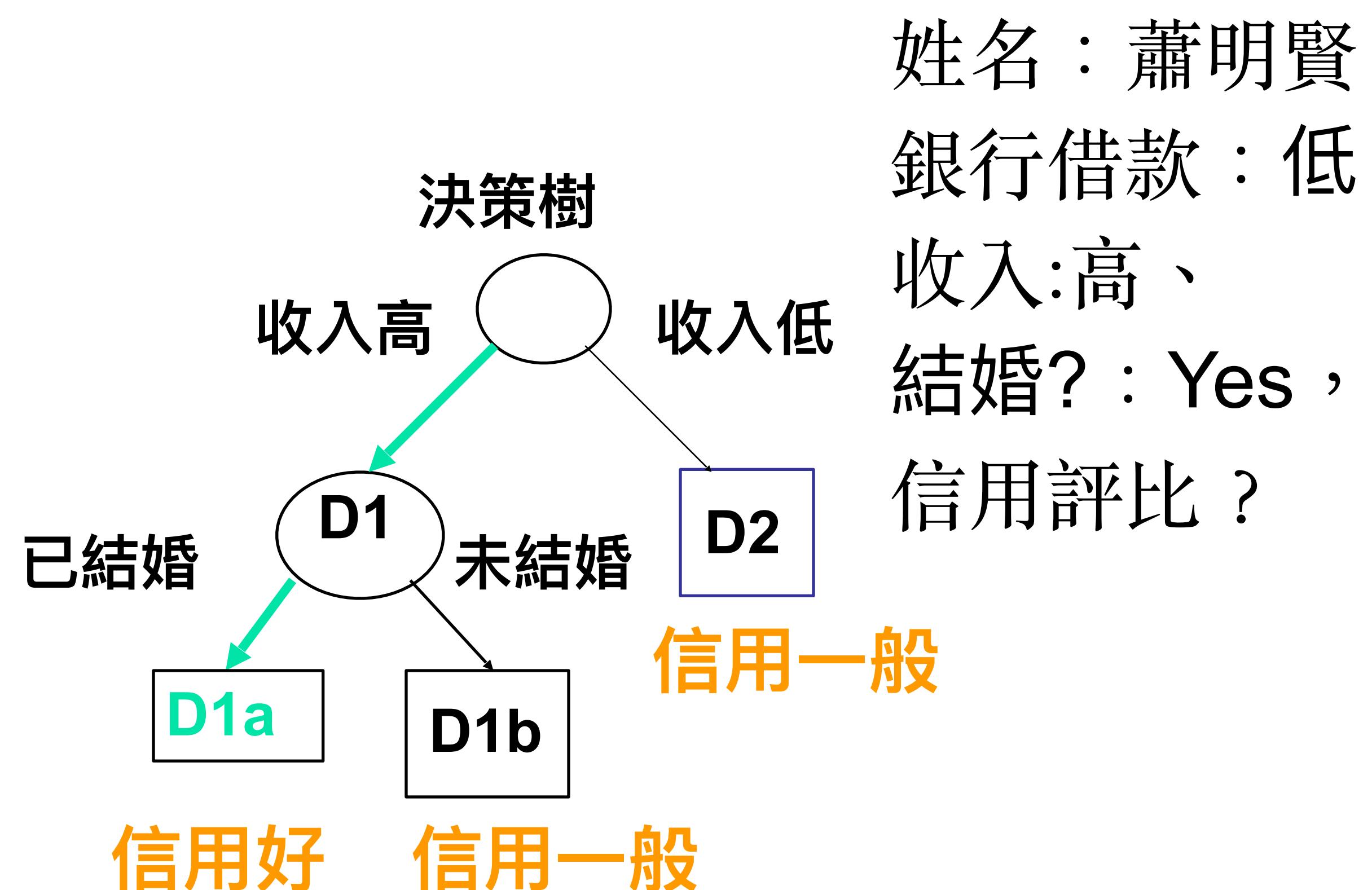


| 姓名 | 銀行借款 | 收入 | 已結婚 | 信用風險 |
|-----|------|----|-----|------|
| 王小明 | 高 | 高 | 是 | 優良 |
| 陳君君 | 低 | 高 | 是 | 優良 |
| 林鈞浩 | 低 | 低 | 否 | 一般 |
| 黃哲平 | 高 | 低 | 否 | 一般 |
| 江雅萍 | 高 | 高 | 否 | 一般 |

3. 決策樹 信用評等



評定信用良好





4. 監督式學習分類



- 監督式學習分類
 - 監督式: 訓練資料已有標籤
 - 新資料分類是根據訓練資料所建立的模型來預測
- 非監督式分群
 - 資料屬於哪一類標籤未知
 - 屬於分群



5. 天氣決策樹 決定是否騎腳踏車



決策樹是一個有向無環圖。
根結點代表所有數據。
分類樹算法可以通過天氣變數，找出最好地解釋非獨立變量騎車。
天氣狀況有晴，雲和雨；氣溫用高低溫度表示；相對濕度用高或一般表示；還有有無風。當然還有是不是真的有騎腳踏車。最終他得到了14行5列的數據表格。



騎腳踏車資料集



| 天氣 | 溫度 | 濕度 | 有無風 | 騎腳踏車嗎 |
|-----|----|----|-----|-------|
| 太陽 | 高溫 | 高 | 無 | 不騎 |
| 太陽 | 高溫 | 高 | 有 | 不騎 |
| 多雲天 | 高溫 | 高 | 無 | 騎 |
| 下雨 | 溫和 | 高 | 無 | 騎 |
| 下雨 | 涼 | 一般 | 無 | 騎 |
| 下雨 | 涼 | 一般 | 有 | 不騎 |
| 多雲天 | 涼 | 一般 | 有 | 騎 |
| 太陽 | 溫和 | 高 | 無 | 不騎 |
| 太陽 | 涼 | 一般 | 無 | 騎 |
| 下雨 | 溫和 | 一般 | 無 | 騎 |
| 太陽 | 溫和 | 一般 | 有 | 騎 |
| 多雲天 | 溫和 | 高 | 有 | 騎 |
| 多雲天 | 高溫 | 一般 | 無 | 騎 |
| 下雨 | 溫和 | 高 | 有 | 不騎 |

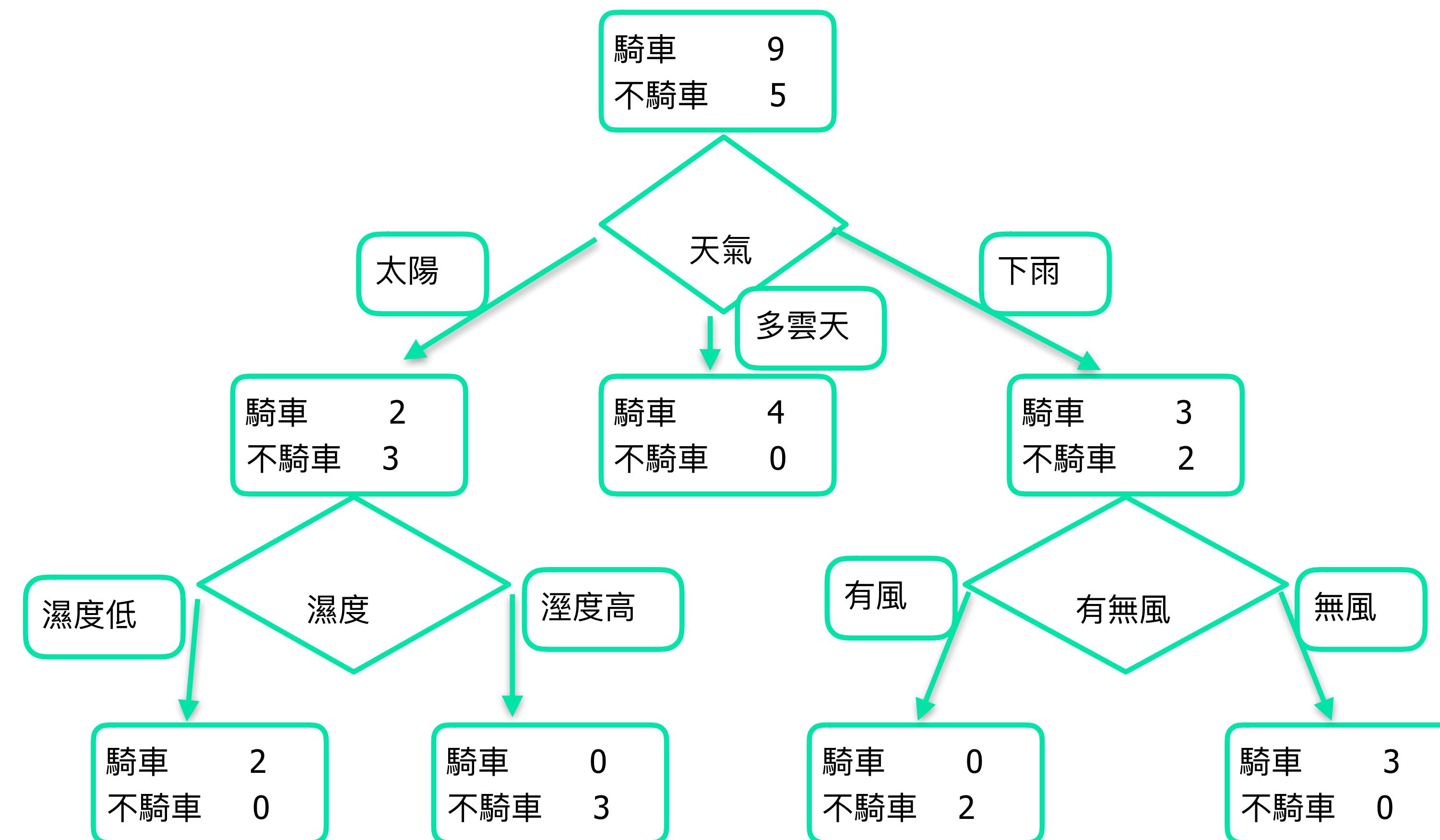


決策樹模型



變數天氣的範疇被劃分為以下三個組：
太陽，多雲天和下雨天。
如果天氣是多雲，人們總是選擇騎腳踏車，而只有少數甚至在下雨天也會騎。
我們把晴天組的分為兩部分，不喜歡濕度高的天氣。
如果雨天還有風的話，就不會有人騎腳踏車。

決策樹模型





指數選取



- 資訊增益 (ID3/C4.5)
 - 所有資訊都假設為類別
 - 能夠像連續值屬性一樣被修改
- Gini 指數 (IBM IntelligentMiner)
 - 所有屬性都假設是連續值
 - 對於每一個屬性都有存在幾個分割的數值
 - 能夠被修改成類別屬性



資訊熵Entropy



ID3演算法可以有下列幾點：

- 1 使用所有沒有使用的屬性並計算與之相關的樣本熵值
- 2 選取其中熵值最小的屬性
- 3 生成包含該屬性的節點

$$I_E(i) = - \sum_j^m f(i,j) \log_2 f(i,j)$$



資訊熵Entropy



- S:目前的資料集
- X:在S中的類別集合
- $p(x)$:在X類別的元素個數對在S資料級的比率

$$H(S) = - \sum_{x \in X} p(x) \log_2 p(x)$$



Information Gain

資訊增益



- $H(S)$:S集合的資訊熵Entropy
- S :目前的資料集
- T :在 S 中的子集合
- $p(t)$:在 t 類別的元素個數對在 S 資料級的比率
- $H(t)$: t 子集的資訊熵

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$



決策樹



決策樹易於理解和實現。

人們在解釋後都有能力去理解決策樹所表達的意義。

對於決策樹，數據的準備是簡單的。其他的技術往往要求先把數據一般化，比如去掉多餘的或者空白的屬性。

能夠同時處理數據型和常規型屬性。其他的技術往往要求數據屬性的單一。

如果給定一個觀察的模型，那麼根據所產生的決策樹很容易推出相應的邏輯表達式。

易於通過靜態測試來對模型進行評測。表示有可能測量該模型的可信度。

在相對短的時間內能夠對大型數據源做出可行且效果良好的結果



決策樹的剪枝



剪枝是決策樹停止分支的方法之一。

剪枝有分預先剪枝和後剪枝兩種。

預先剪枝是在樹的生長過程中設定一個指標，當達到該指標時就停止生長，這樣做容易產生視界局限，一旦停止分支，使得節點N成為葉節點。

後剪枝中樹首要充分生長，直到葉節點都有最小的不純度值為止。



決策樹演算法



- Basic algorithm (a greedy algorithm)
 - 樹的建立是由上而下遞迴的分割和組合
 - 一開始所有訓練集是放在root根
 - 屬性是被分類的
 - 樣本被遞迴式分割,已選取屬性為基礎
 - 測試屬性以統計的方式被選取(資訊增益)
- 停止分割的條件
 - 所有的樣本都屬於同一個類別
 - 已經沒有可以分割的屬性,已經到樹的葉子
 - 已經沒有樣本可分辨了

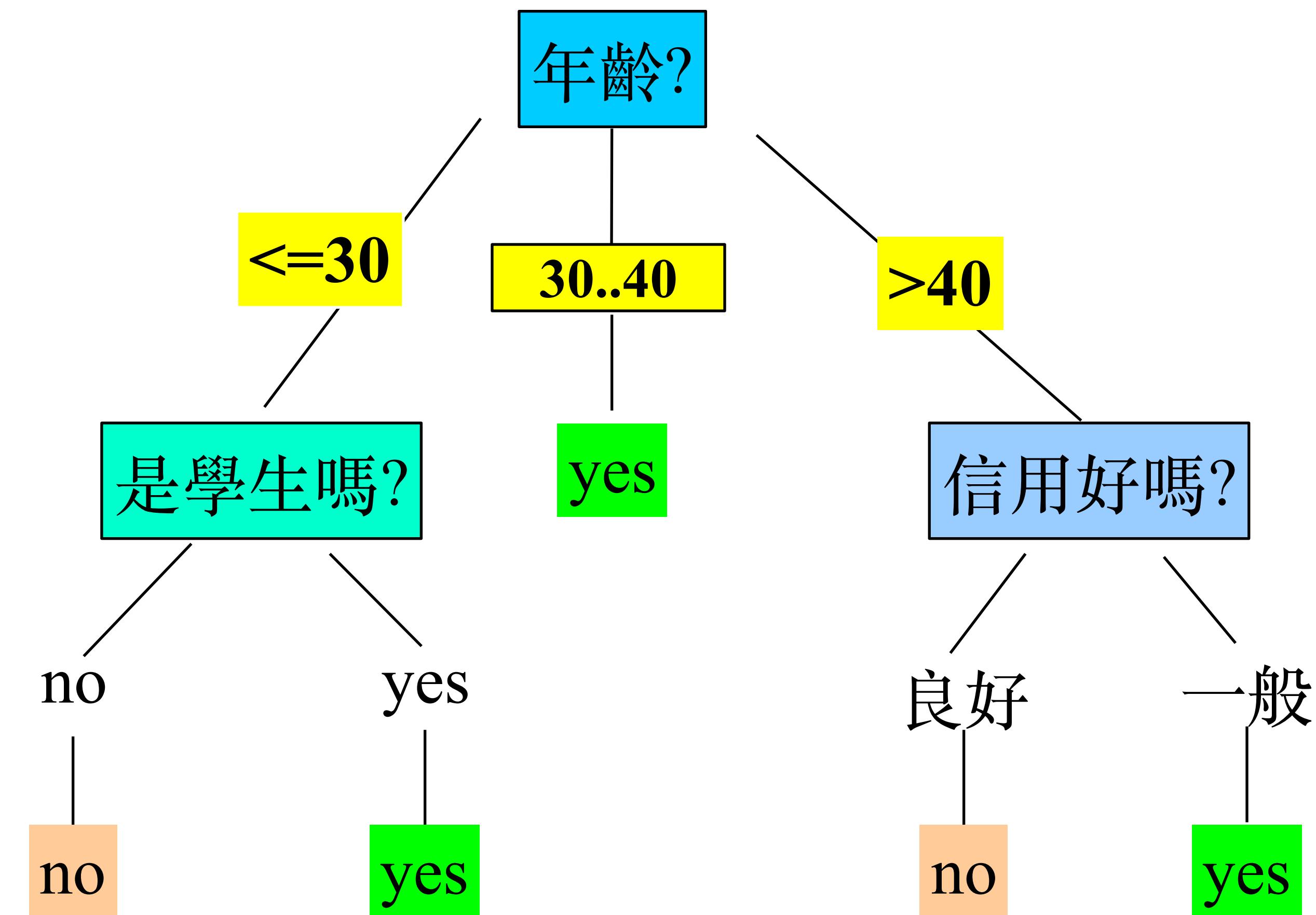


消費者購買電腦訓練資料集



| 年紀 | 收入 | 在學 | 信用評等 | 買電腦 |
|--------|----|----|------|-----|
| <=30 | 高 | 否 | 一般 | no |
| <=30 | 高 | 否 | 良好 | no |
| 30..40 | 高 | 否 | 一般 | yes |
| >40 | 一般 | 否 | 一般 | yes |
| >40 | 低 | 是 | 一般 | yes |
| >40 | 低 | 是 | 良好 | no |
| 30..40 | 低 | 是 | 良好 | yes |
| <=30 | 一般 | 否 | 一般 | no |
| <=30 | 低 | 是 | 一般 | yes |
| >40 | 一般 | 是 | 一般 | yes |
| <=30 | 一般 | 是 | 良好 | yes |
| 30..40 | 一般 | 否 | 良好 | yes |
| 30..40 | 高 | 是 | 一般 | yes |
| >40 | 一般 | 否 | 良好 | no |

買電腦的決策樹





資訊熵

$$I(p, n) = I(9, 5) = 0.940$$



$$I(9,5) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14}$$

$$I(2,3) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$



期望資訊



- 假如 S_i 包含 P 的樣本 p_i 而且包含 N 的樣本 n_i , 在所有子樹 S_i 的期望資訊 $E(A)$

| 年紀 | P | N | I(P,N) |
|--------|---|---|--------|
| <=30 | 2 | 3 | 0.971 |
| 30..40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{P+N} I(p_i, n_i)$$

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$



$$A. I(p, n) = I(9, 5) = 0.940$$

$$B. \text{Gain}(\text{年齡}) = I(9,5) - E(\text{年齡}) = 0.94 - 0.6935 = 0.2465$$

$$\text{Gain}(\text{年齡}) = I(p, n) - E(\text{年齡})$$

$$E(\text{年齡}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.6935$$

$$\text{Gain}(\text{收入}) = 0.029$$

$$\text{Gain}(\text{在學}) = 0.151$$

$$\text{Gain}(\text{信用評比}) = 0.048$$

| 年紀 | P | N | I(P,N) |
|--------|---|---|--------|
| <=30 | 2 | 3 | 0.971 |
| 30..40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |



資訊熵增益Gain計算



- A. 類別 P: 買電腦 = "yes"
- B. 類別 N: 買電腦 = "no"
- C. $I(p, n) = I(9, 5) = 0.940$
- D. 計算年紀的熵:

$$Gain(\text{年齡}) = I(p, n) - E(\text{年齡})$$

| 年紀 | P | N | I(P,N) |
|--------|---|---|--------|
| <=30 | 2 | 3 | 0.971 |
| 30..40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

$$Gain(\text{收入}) = 0.029$$

$$Gain(\text{在學}) = 0.151$$

$$Gain(\text{信用評比}) = 0.048$$



ID3決策樹演算法



- 計算對指定樣本分類所需要的資訊熵
- 計算所有特徵屬性的資訊熵
- 從所有的特徵列中選取資訊增益最大的那個做為
跟節點或內部節點(第一次遞迴選取年齡列
 $G=0.2465$)
- 依據劃分節點的不同來拆分資料極為許多子集合，
在遞迴的執行上面演算法
- 子集中只有一個類別標籤則停止



Gini 指數



- Gini指數, p_j 是在T中的j類別頻率

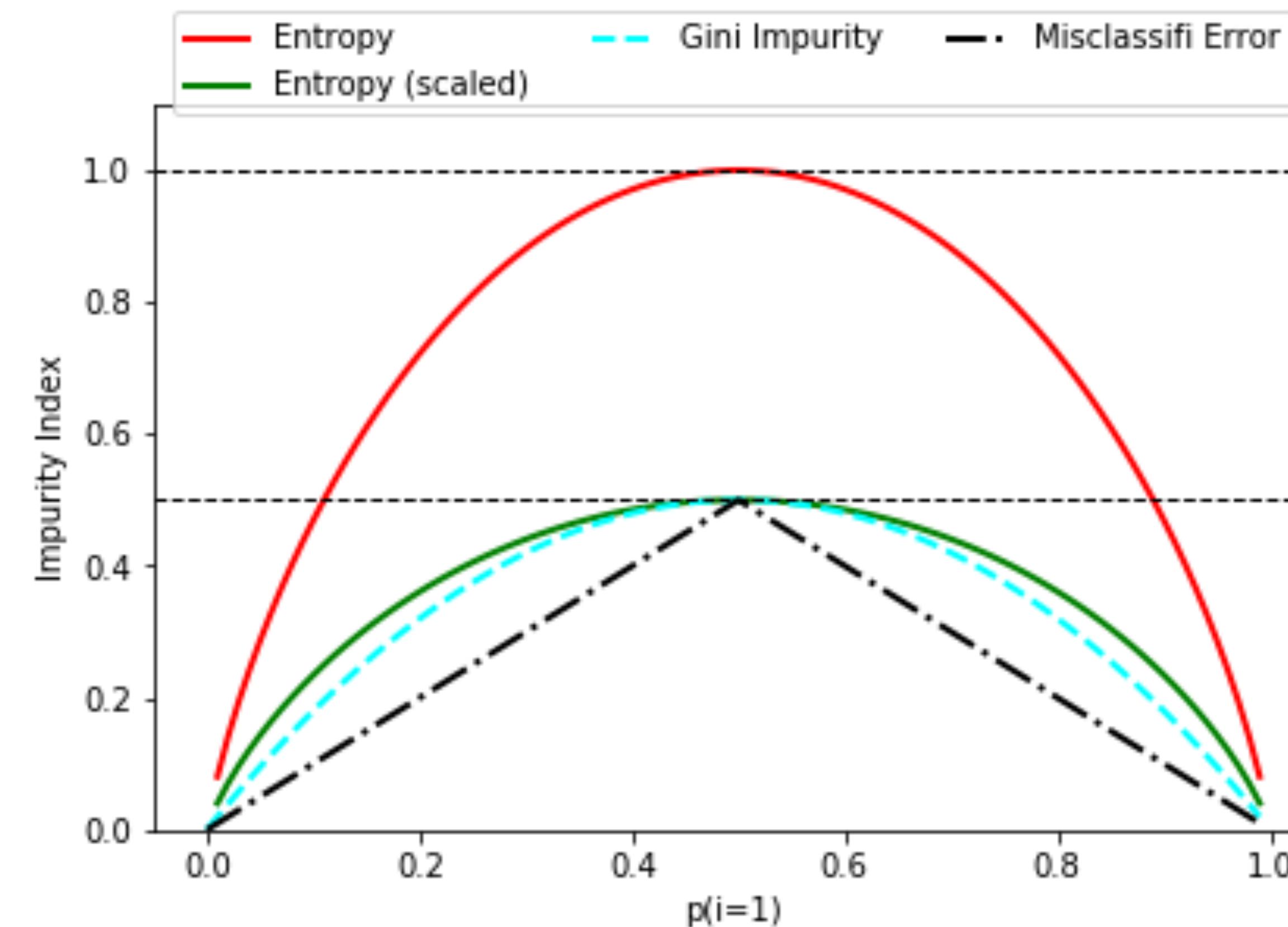
$$gini(T) = 1 - \sum_{j=1}^n p_j^2$$



Entropy 熵



```
8 import numpy as np
9 from sklearn import datasets
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.metrics import accuracy_score
12 from sklearn.tree import DecisionTreeClassifier
13 from matplotlib.colors import ListedColormap
14 import matplotlib.pyplot as plt
15
16
17 #####
18 print(88 * '=')
19 print('Section: Decision tree')
20 print(88 * '-')
21
22 def entropy(p):
23     return - p * np.log2(p) - (1 - p) * np.log2((1 - p))
24
25 def gini(p):
26     return p * (1 - p) + (1 - p) * (1 - (1 - p))
27
```





鳶尾花決策樹分類



```
8 import numpy as np
9 from sklearn import datasets
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.metrics import accuracy_score
12 from sklearn.tree import DecisionTreeClassifier
13 from matplotlib.colors import ListedColormap
14 import matplotlib.pyplot as plt
15 from sklearn.model_selection import train_test_split
16
17 def versiontuple(v):
18     return tuple(map(int, (v.split("."))))
```



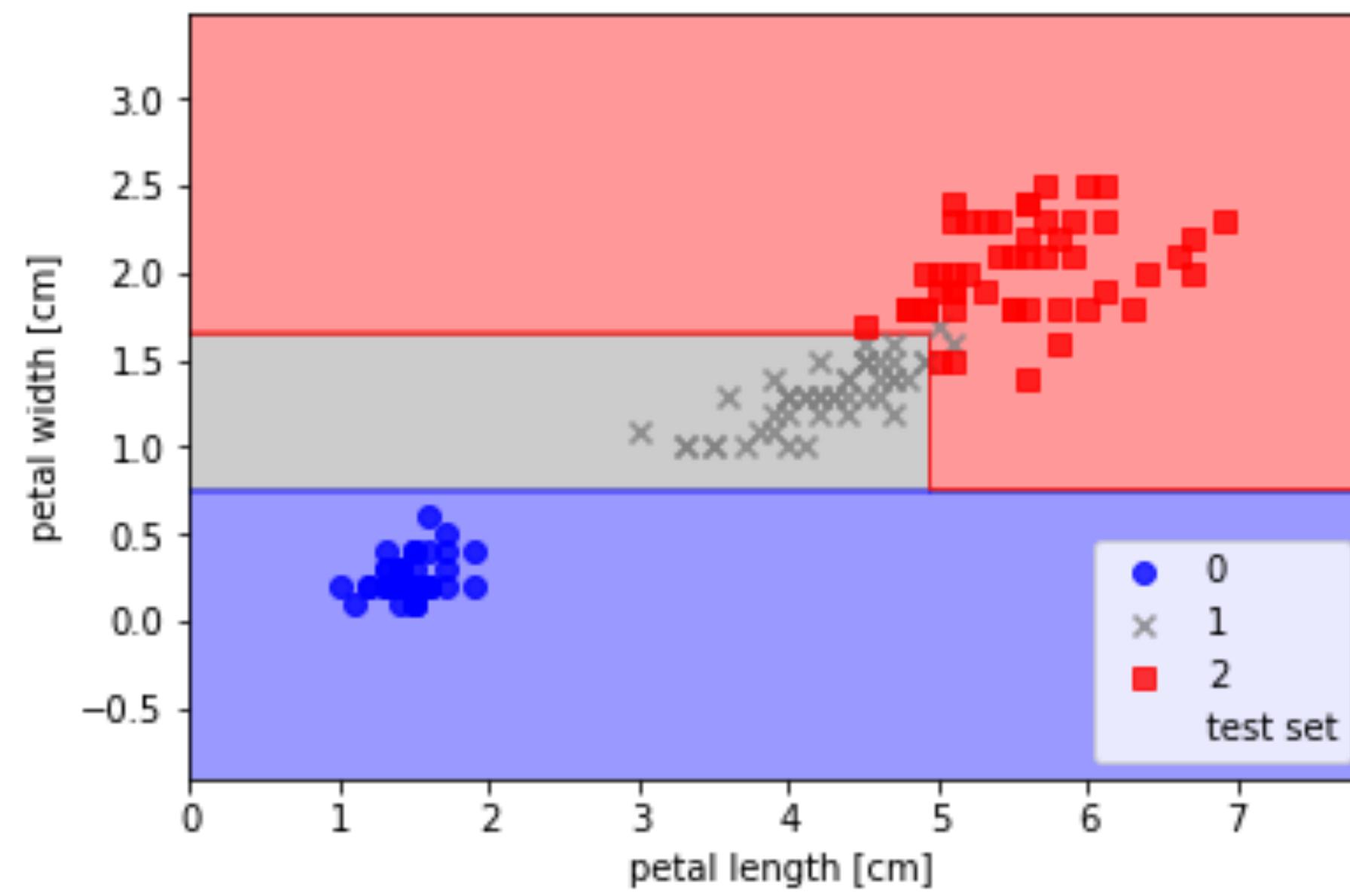
鳶尾花決策樹分類



```
56 print(88 * '=')
57 print('Section: Building a decision tree')
58 print(88 * '-')
59
60 iris = datasets.load_iris()
61 X = iris.data[:, [2, 3]]
62 y = iris.target
63 print('Class labels:', np.unique(y))
64
65 X_train, X_test, y_train, y_test = train_test_split(
66     X, y, test_size=0.3, random_state=0)
67
68 tree = DecisionTreeClassifier(criterion='entropy',
69                               max_depth=3, random_state=0)
70 tree.fit(X_train, y_train)
71
72 X_combined = np.vstack((X_train, X_test))
73 y_combined = np.hstack((y_train, y_test))
74 plot_decision_regions(X_combined, y_combined,
75                       classifier=tree, test_idx=range(105, 150))
76
77 plt.xlabel('petal length [cm]')
78 plt.ylabel('petal width [cm]')
79 plt.legend(loc='lower right')
80
81 plt.show()
```



鳶尾花決策樹分類





- 輸入sklearn.ensemble import RandomForestClassifier

```
8 import numpy as np
9 from sklearn import datasets
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.metrics import accuracy_score
12 from sklearn.ensemble import RandomForestClassifier
13 from matplotlib.colors import ListedColormap
14 import matplotlib.pyplot as plt
15 from sklearn.model_selection import train_test_split
```



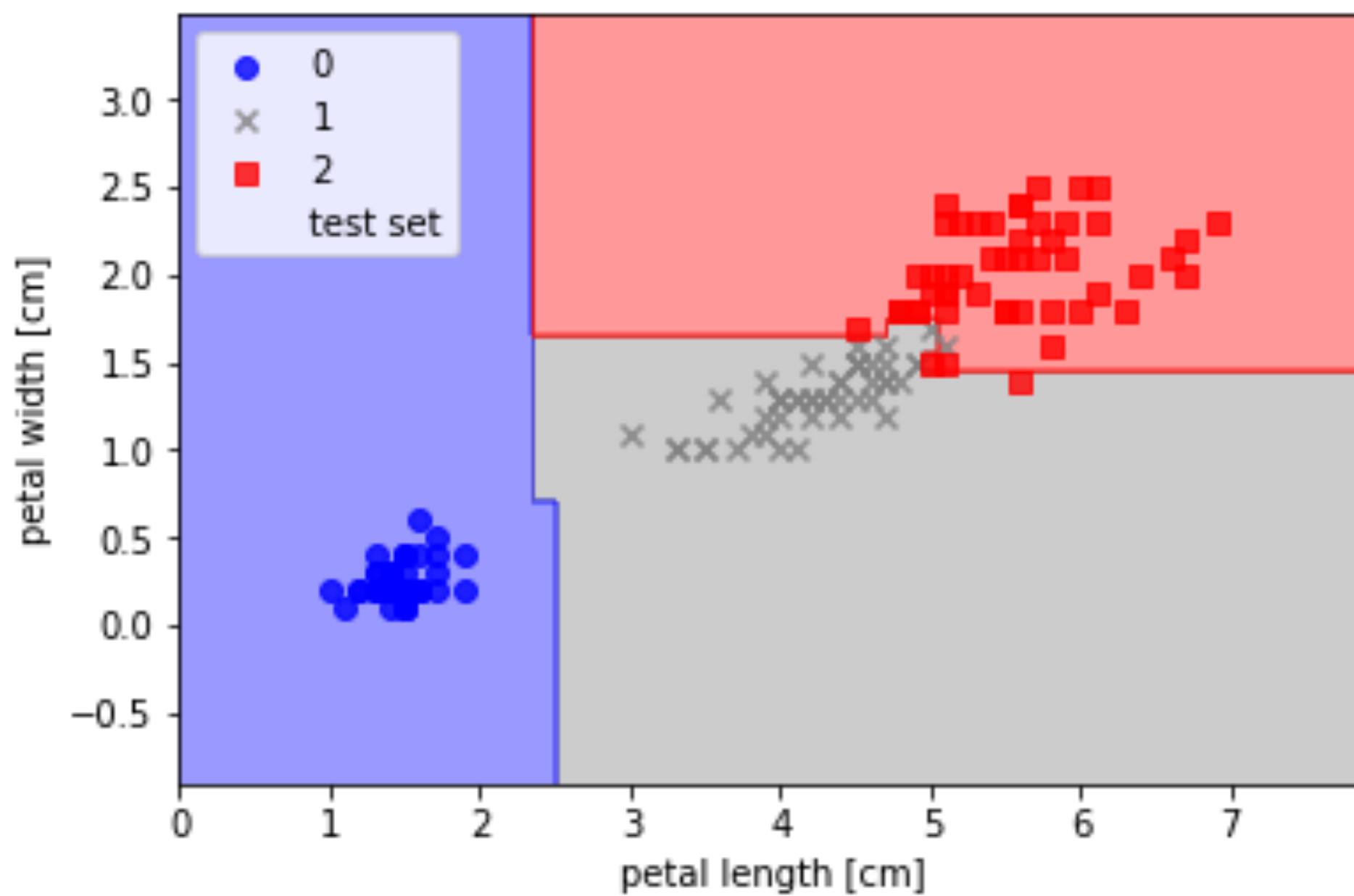
隨機森林結合強學習



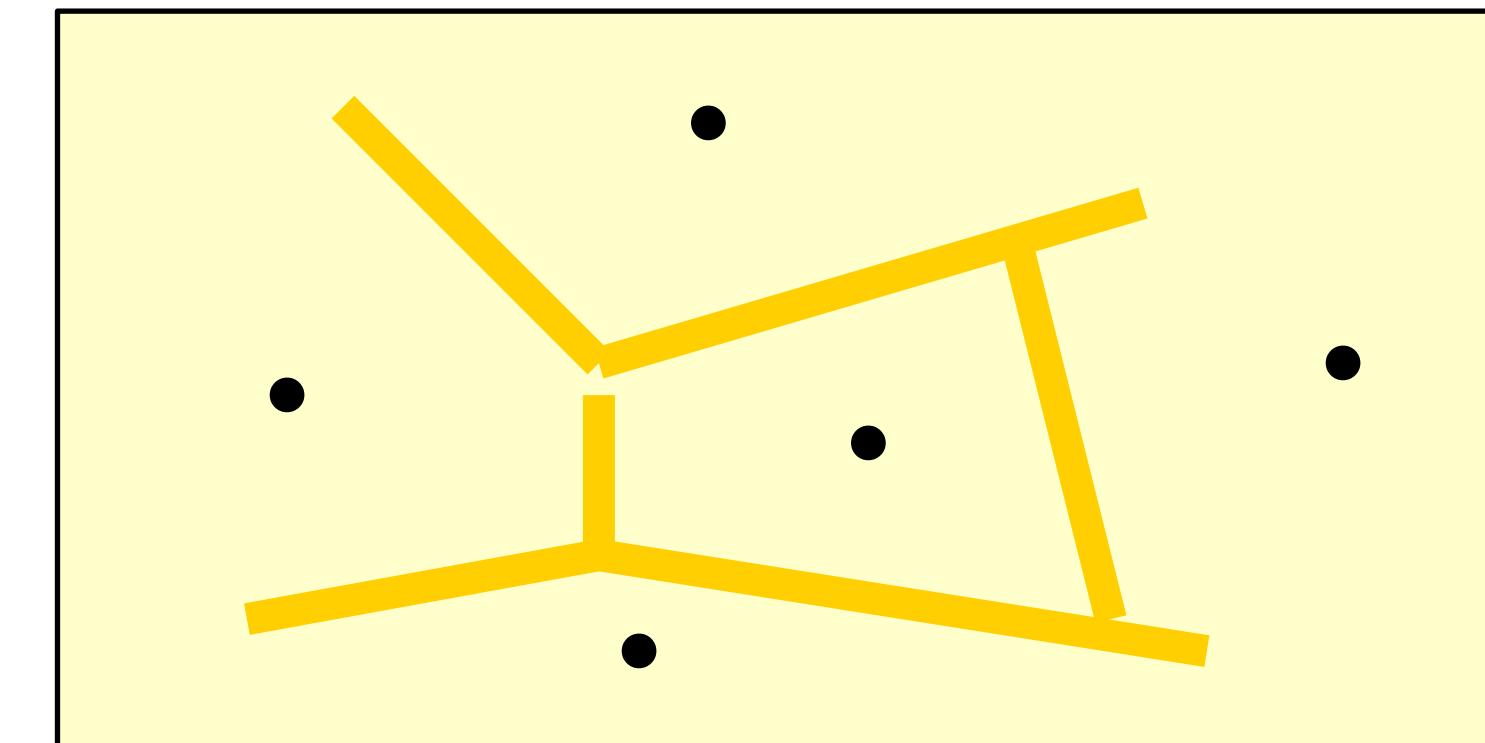
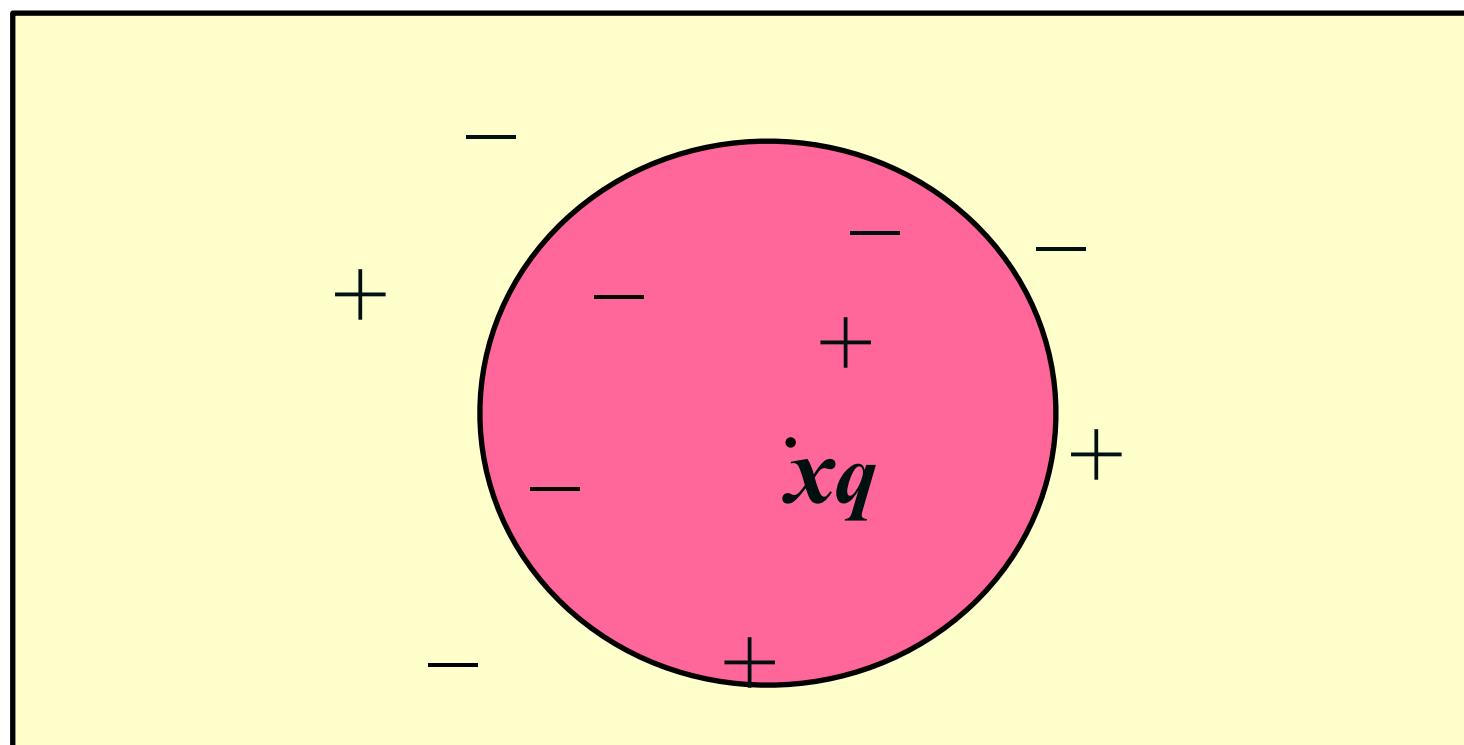
```
60 iris = datasets.load_iris()
61 X = iris.data[:, [2, 3]]
62 y = iris.target
63 print('Class labels:', np.unique(y))
64
65 X_train, X_test, y_train, y_test = train_test_split(
66     X, y, test_size=0.3, random_state=0)
67
68 forest = RandomForestClassifier(criterion='entropy',
69                                 n_estimators=10,
70                                 random_state=1,
71                                 n_jobs=2)
72 forest.fit(X_train, y_train)
73
74 plot_decision_regions(X_combined, y_combined,
75                       classifier=forest, test_idx=range(105, 150))
76
77 plt.xlabel('petal length [cm]')
78 plt.ylabel('petal width [cm]')
79 plt.legend(loc='upper left')
80 plt.show()
```



隨機森林結合強學習



- Voronoi 維諾圖: 它是由一組由連接兩鄰點直線的垂直平分線組成的連續多邊形組成。

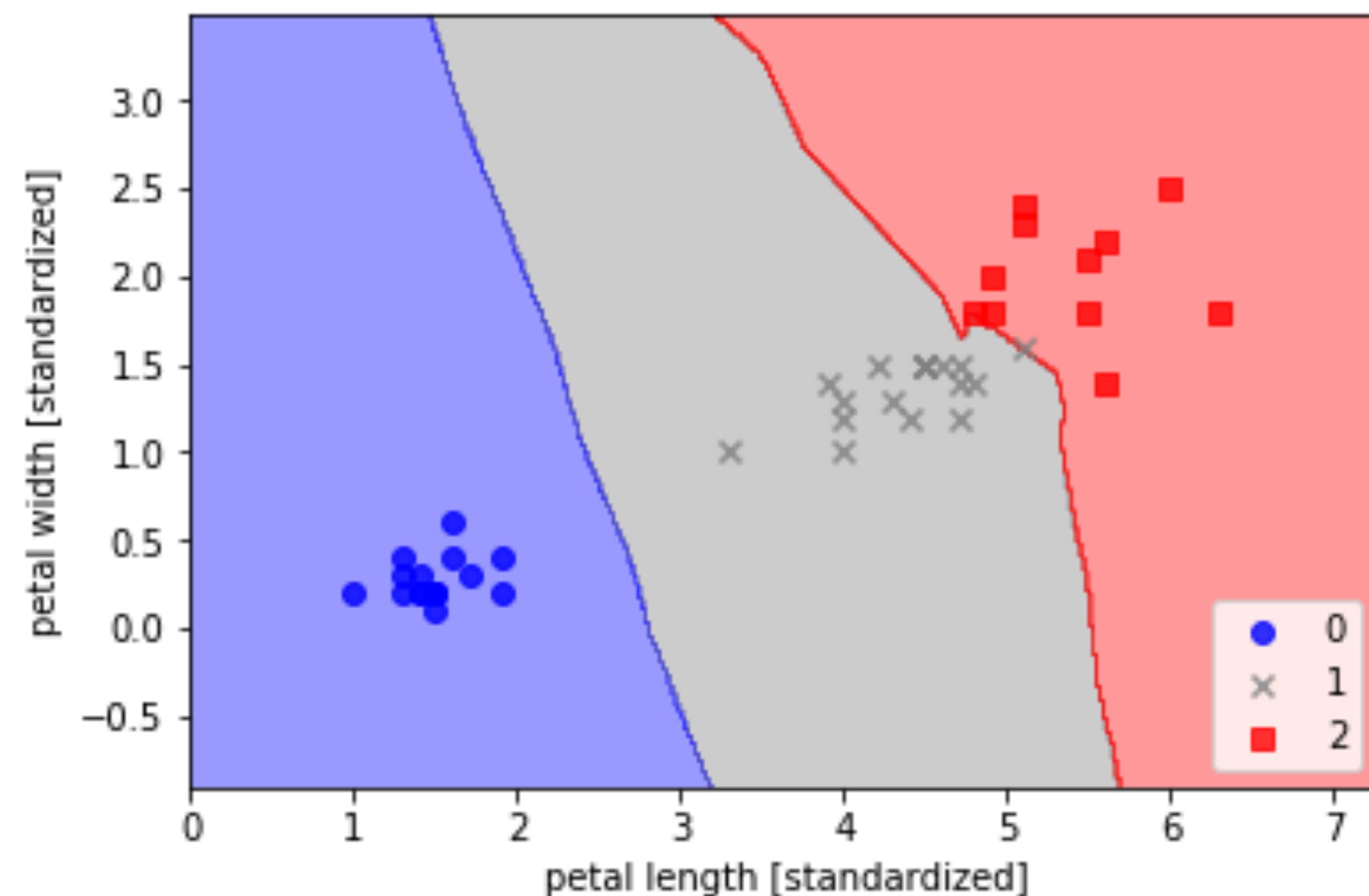




K-nearest最鄰近分類演算法



```
8 import numpy as np
9 from sklearn import datasets
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.metrics import accuracy_score
12 from sklearn.neighbors import KNeighborsClassifier
13 from matplotlib.colors import ListedColormap
14 import matplotlib.pyplot as plt
15 from sklearn.model_selection import train_test_split
16
17 print('Section: K-nearest neighbors')
18 iris = datasets.load_iris()
19 X = iris.data[:, [2, 3]]
20 y = iris.target
21 print('Class labels:', np.unique(y))
22
23 X_train, X_test, y_train, y_test = train_test_split(
24     X, y, test_size=0.3, random_state=0)
25 #歐氏距離
26 knn = KNeighborsClassifier(n_neighbors=5, p=2,
27                             metric='minkowski')
28 knn.fit(X_train, y_train)
29
30 plot_decision_regions(X_test, y_test,
31                       classifier=knn)
```





python



- Thanks



分群



- 分群分析
- 分群的應用
- 資料採礦對分群的需要
- 分群的依據
- 主要的分群方式
 - 群聚演算法k-means
 - 階層式分群
 - 密度為基礎的分群方法



分群分析

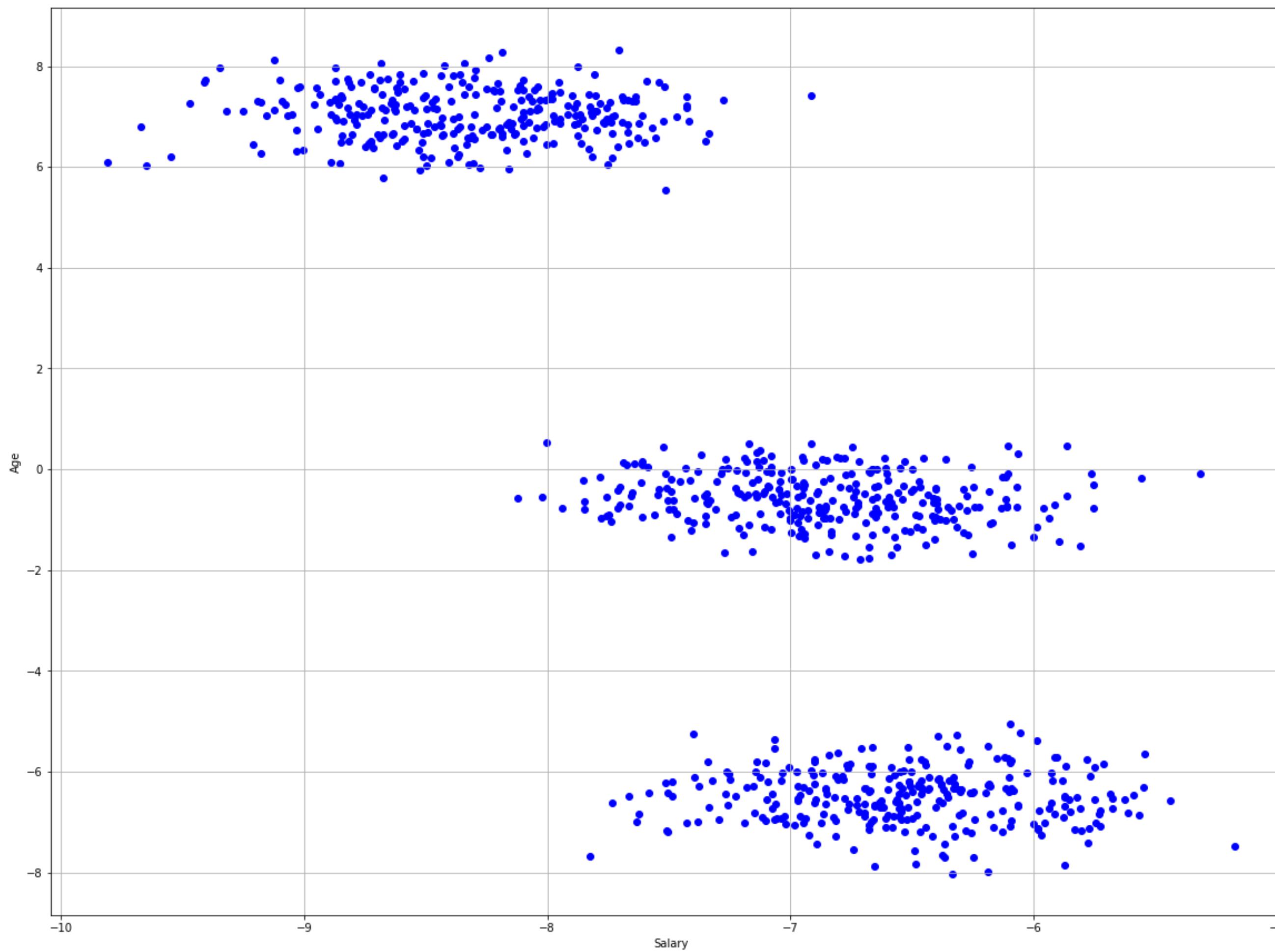


客戶經歷

| Name | Age | 學歷 | 工作 | 收入 |
|------|-----|----|-----|--------|
| 王大偉 | 50 | 學士 | 律師 | 80,000 |
| 林小寶 | 29 | 碩士 | 醫生 | 90,000 |
| 江鈺雯 | 30 | 學士 | 工程師 | 60,000 |
| 陳鴻文 | 48 | 碩士 | 老師 | 55,000 |

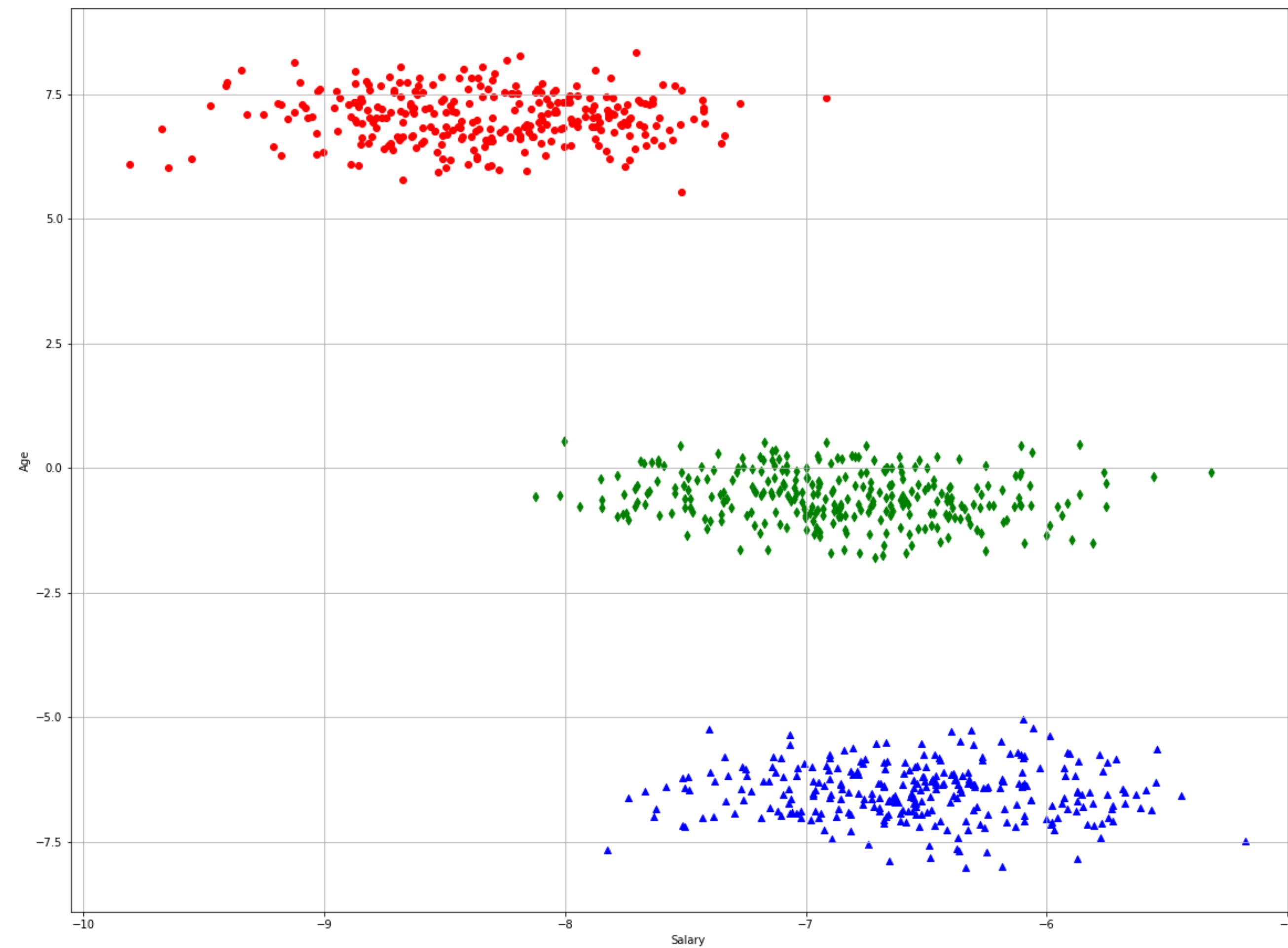


分群分析





分群分析





分群的應用



- 樣本辨識
- GIS空間地圖分析
- 影像辨識
- 經濟財務資料科學
- 消費者市場分析
- 網站爬蟲
 - 網頁網站文件分類
 - 部落格資料群組分群分析



分群應用



- 消費者市場分析：讓業務員區分消費者市場，然後使用此知識發展目標市場行銷，如Facebook臉書廣告和Google廣告。
- 金融保險應用：汽車第三責任險，汽車保險，健康保險保費，癌症保費，火災險，各種金融商品的保費分群 I
- 全民生醫健康計畫：辨識各地區生醫品質如PM2.5的分佈，工業區商業區住宅的分佈對全民健康的影響
- 地震區域風險分析：哪一些是地震高風險區域，附近不可建核電廠
- 工業4.0分析：工業自動化分群應用分析。哪些工作場所是要非常小心的控制。



資料採礦對分群的需要



- 彈性
- 可以處理不一樣型態的屬性
- 資料可以被準確得分群
- 懂該資料的專家或專業人士
- 可以去除不相關的資料雜訊
- 對輸入資料敏感
- 資料屬性高維度



分群的依據



- 在做分類時常常需要計算不同樣本之間的相似性度量(Similarity Measurement)，這時通常採用的方法就是計算樣本間的「距離」。
- 使用什麼樣的方法計算距離，關係到分群的正確與否。



分群的依據,和該群相似, 幾何空間相似



- 幾何空間相似是分群的依據
- 閔可夫斯基距離

$$d(i, j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \dots + |x_{i_p} - x_{j_p}|^q)}$$

$i = (x_{i_1}, x_{i_2}, \dots, x_{i_p})$ 和 $j = (x_{j_1}, x_{j_2}, \dots, x_{j_p})$

兩個p維度的資料 q 是正整數

- 當 $q = 1$,是曼哈頓距離

$$d(i, j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \dots + |x_{i_p} - x_{j_p}|$$



分群的依據,和該群相似, 幾何空間相似



- 當 $q = 2$,歐氏距離(Euclidean Distance)

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)}$$

- 歐氏距離特性
 - $d(i,j) \geq 0$
 - $d(i,i) = 0$
 - $d(i,j) = d(j,i)$
 - $d(i,j) \leq d(i,k) + d(k,j)$



分群的依據,和該群相似,
幾何空間相似



閔氏距離，包括曼哈頓距離、歐氏距離。
二維樣本(身高,體重)，其中身高範圍是152~188，
體重範圍是45~92，有三個樣本：a(182,75)，
b(190,85)，c(174,45)。那麼a與b之間的閔氏距離（無論是曼哈頓距離、歐氏距離）等於a與c之間的閔氏距離，但是身高的1cm為相同於體重的1kg.
閔氏距離的情況：(1)未將各個份量的量綱(scale)，
也就是「單位」加權重。(2)各個份量的分佈（期望，
方差等)可能是不同的。



主要的分群方式



- 群聚演算法k-means
- 建立階層式的資料解構演算法
- 以密度為基礎的空間集群方法DBSCAN演算法



群聚演算法k-means



- 將資料庫的n個物件來建構K個群聚的分群
- k-means:群聚的中心為基礎的分群
 - 遞迴的尋找最小成本
 - 設有k個群聚,0為第0次開始,每次加1,調整群聚中心

$$K^{(0)} = \{\mu_1^0, \mu_2^0, \mu_3^0, \dots, \mu_k^0\}$$

- 計算群聚成本最小化

$$Cost = \sum_t |x_t - \mu_i|^2 \quad \forall i \in (1, k)$$



執行KMeans分群



- 它會遞迴的選擇最靠近最終質心的初始質心
- 設定有三個二維的特徵群聚

```
57     X, _ = make_blobs(n_samples=nb_samples, n_features=2,
58                         centers=3, cluster_std=0.5)
59     # 顯示資料
60     show_dataset(X)
61     # 建立和顯示 K-Means
62     kmCluster = KMeans(n_clusters=3)
63     kmCluster.fit(X)
64     # 顯示中心
65     print(kmCluster.cluster_centers_)
66     # 顯示群集資料集
67     show_clustered_dataset(X, kmCluster)
```



顯示880筆資料



```
19 np.random.seed(880)
20
21 nb_samples = 880
22
23
24 def show_dataset(X):
25     fig, ax = plt.subplots(1, 1, figsize=(20, 15))
26
27     ax.grid()
28     ax.set_xlabel('Salary')
29     ax.set_ylabel('Age')
30
31     ax.scatter(X[:, 0], X[:, 1], marker='o', color='b')
32
33     plt.show()
```



顯示群集資料集



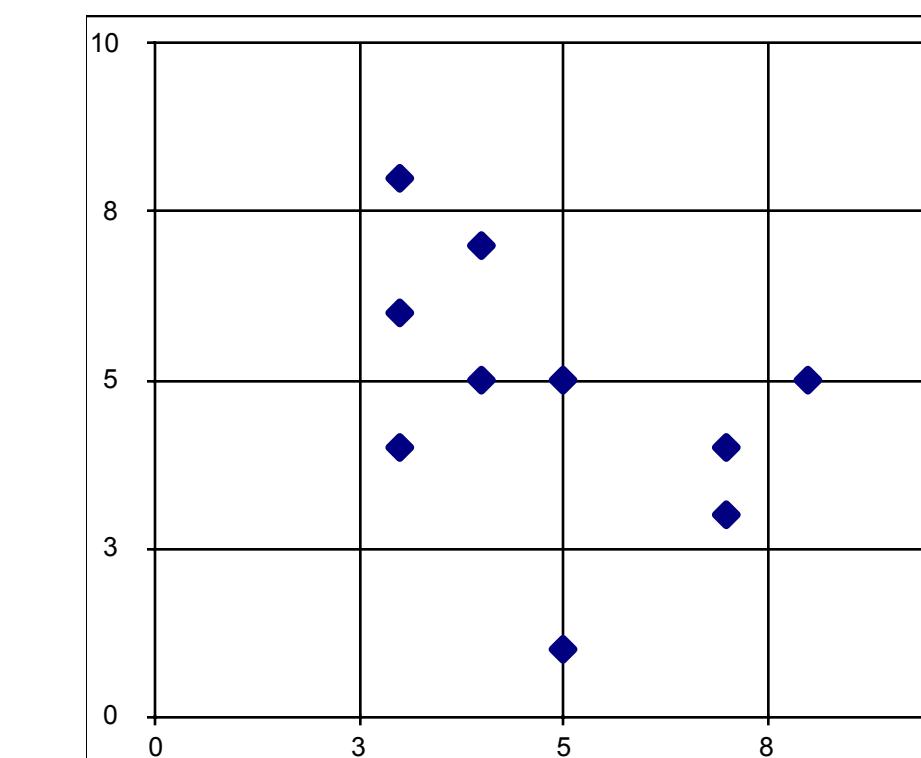
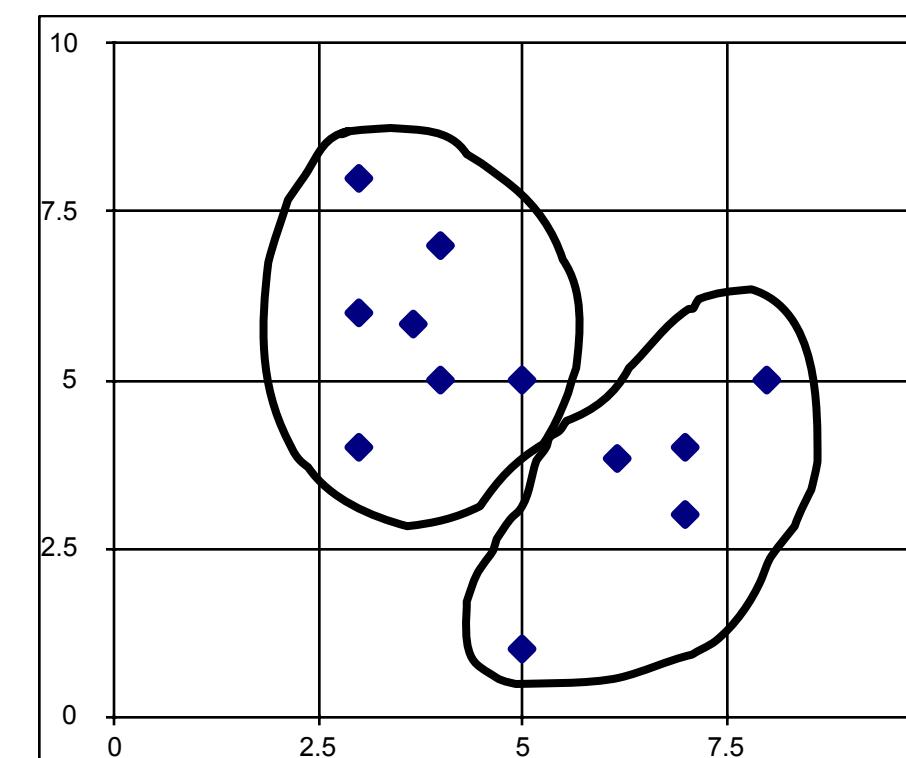
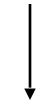
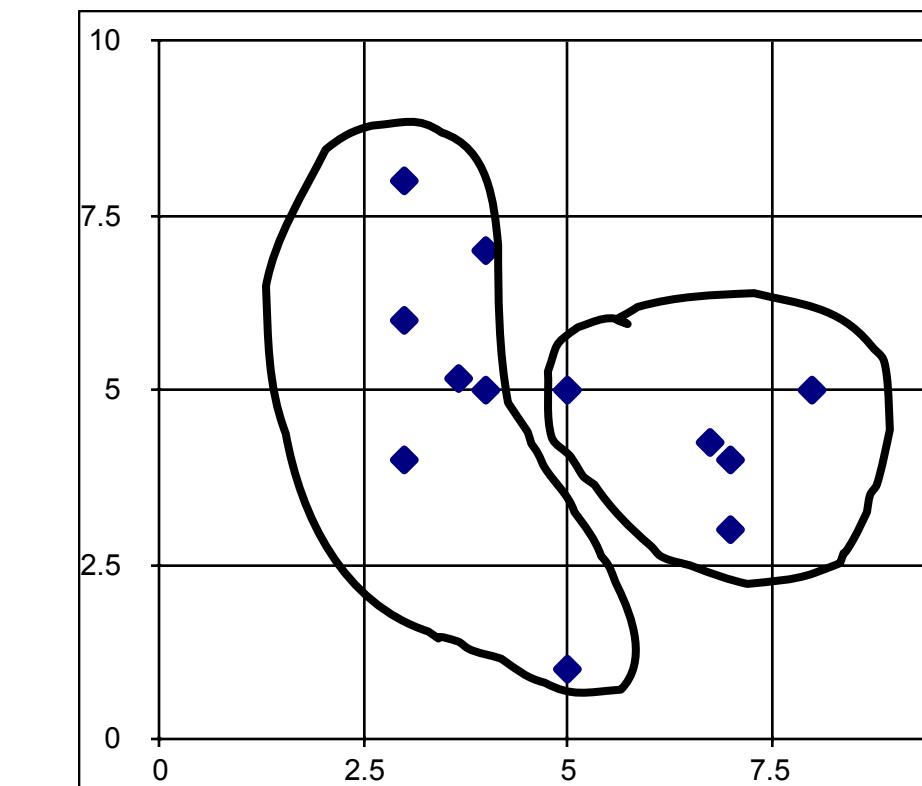
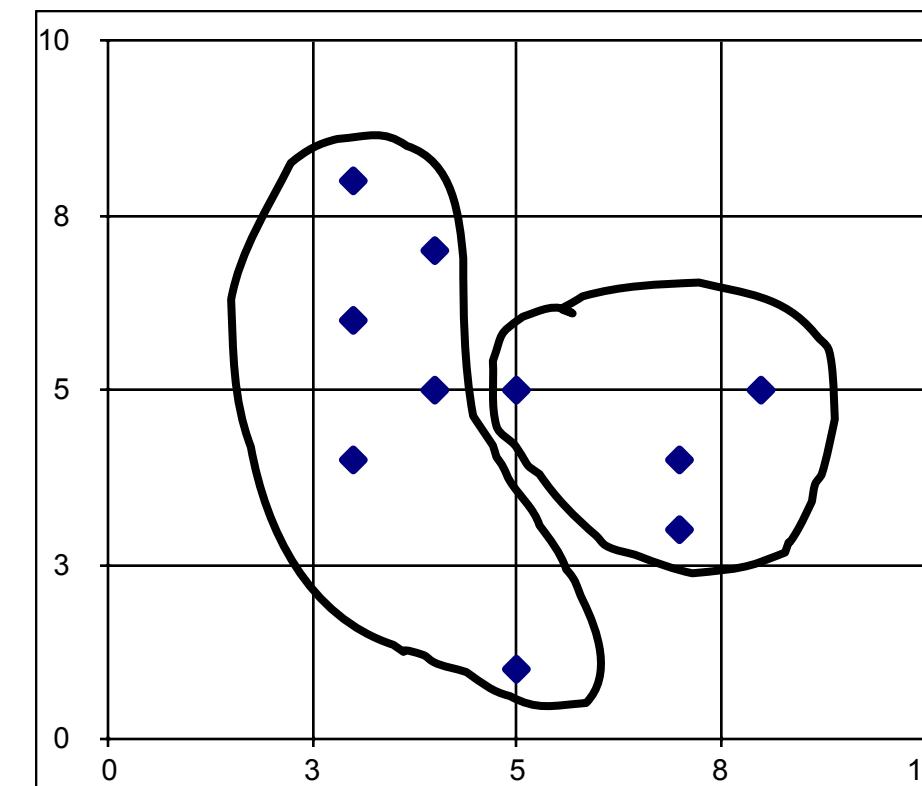
```
36 def show_clustered_dataset(X, kmeansCluster):
37     fig, ax = plt.subplots(1, 1, figsize=(20, 15))
38
39     ax.grid()
40     ax.set_xlabel('Salary')
41     ax.set_ylabel('Age')
42
43     for i in range(nb_samples):
44         c = kmeansCluster.predict(X[i].reshape(1, -1))
45         if c == 0:
46             ax.scatter(X[i, 0], X[i, 1], marker='o', color='r')
47         elif c == 1:
48             ax.scatter(X[i, 0], X[i, 1], marker='^', color='b')
49         else:
50             ax.scatter(X[i, 0], X[i, 1], marker='d', color='g')
51
52     plt.show()
```



K-Means 群聚演算法



- Example

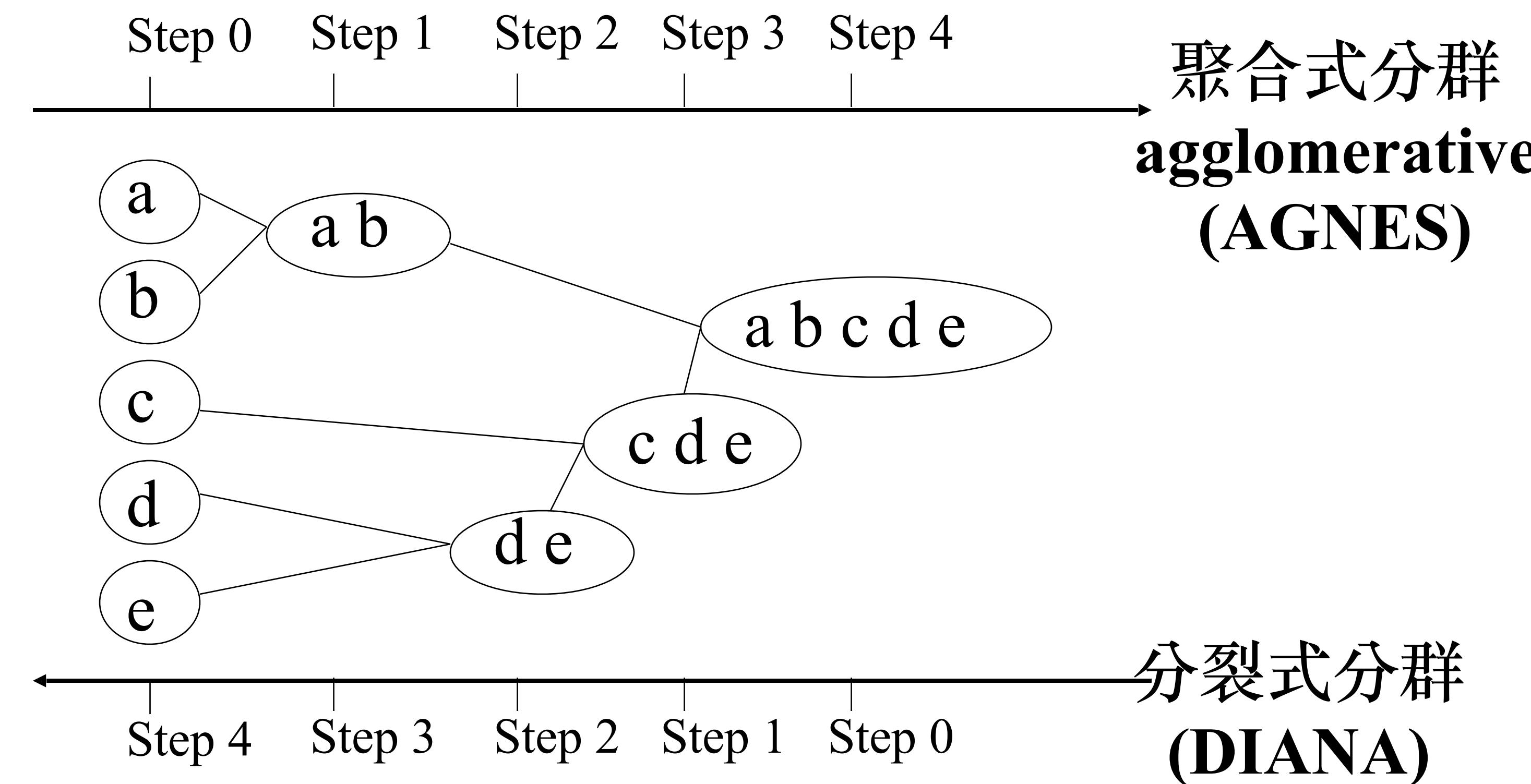




階層式分群



- 使用距離矩陣當作分群的標準

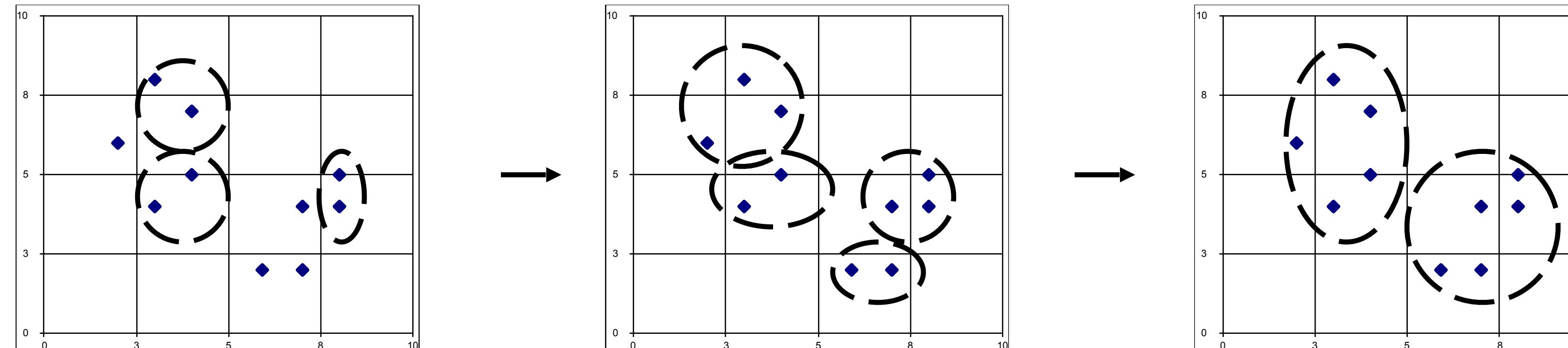




聚合式分群



- 階層式聚合式分群會從下面開始合併群聚,直到滿足停止條件為止。
 - 計算所有樣本的距離矩陣
 - 將每個樣本作為一個單一的集群
 - 合併兩個最近的集群對
 - 更新距離矩陣
 - 重複上面步驟,直到群聚成兩個分群





聚合式分群



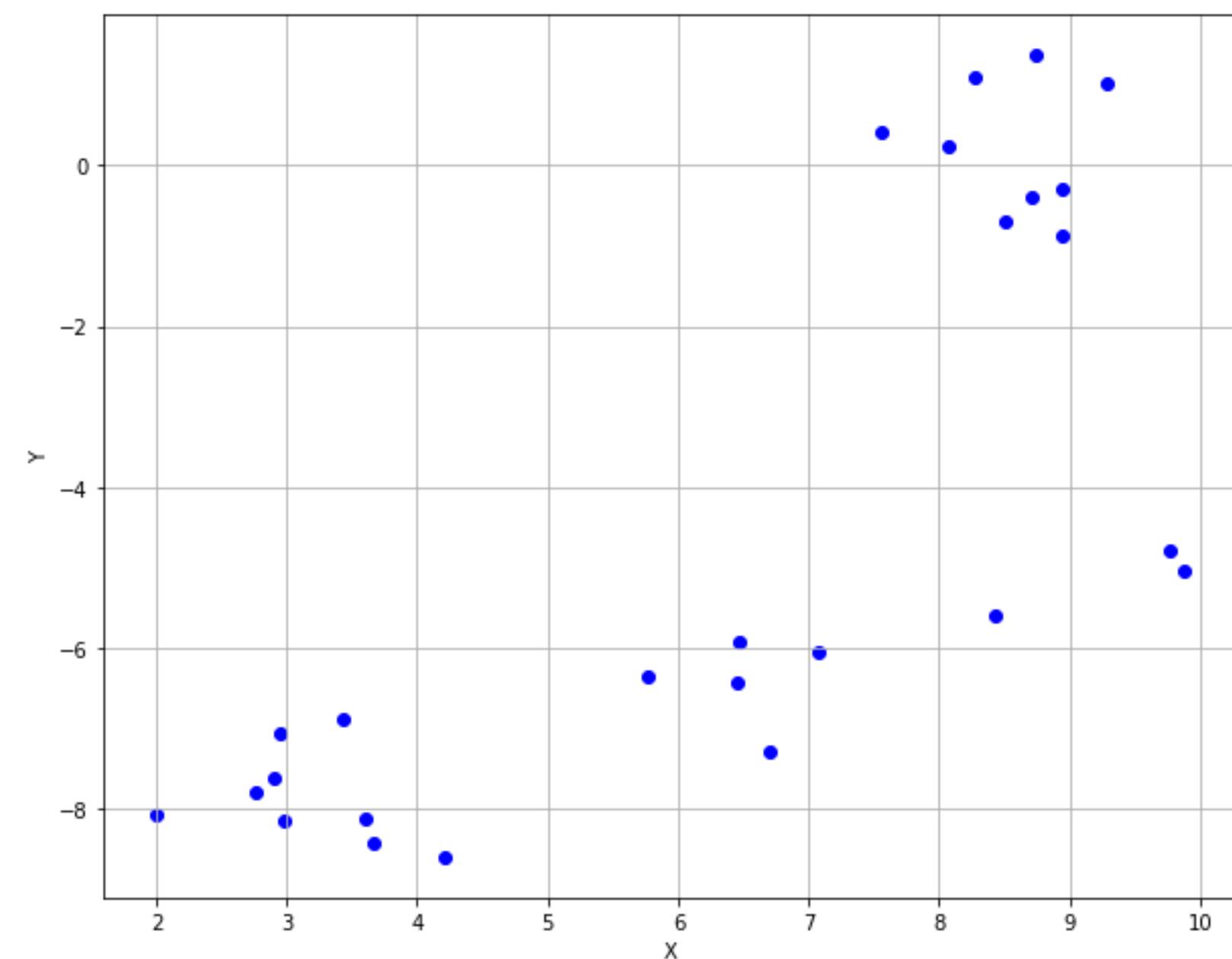
- 階層式聚合式分群會從下面開始合併群聚,直到滿足停止條件為止.
- 使用歐氏距離

$$X = \{ \overline{x_1}, \overline{x_2}, \dots, \overline{x_n} \} \quad \overline{x_i} \in \Re^m$$

$$d_{Euclidean}(\overline{x_1}, \overline{x_2}) = \|\overline{x_1} - \overline{x_2}\|_2$$

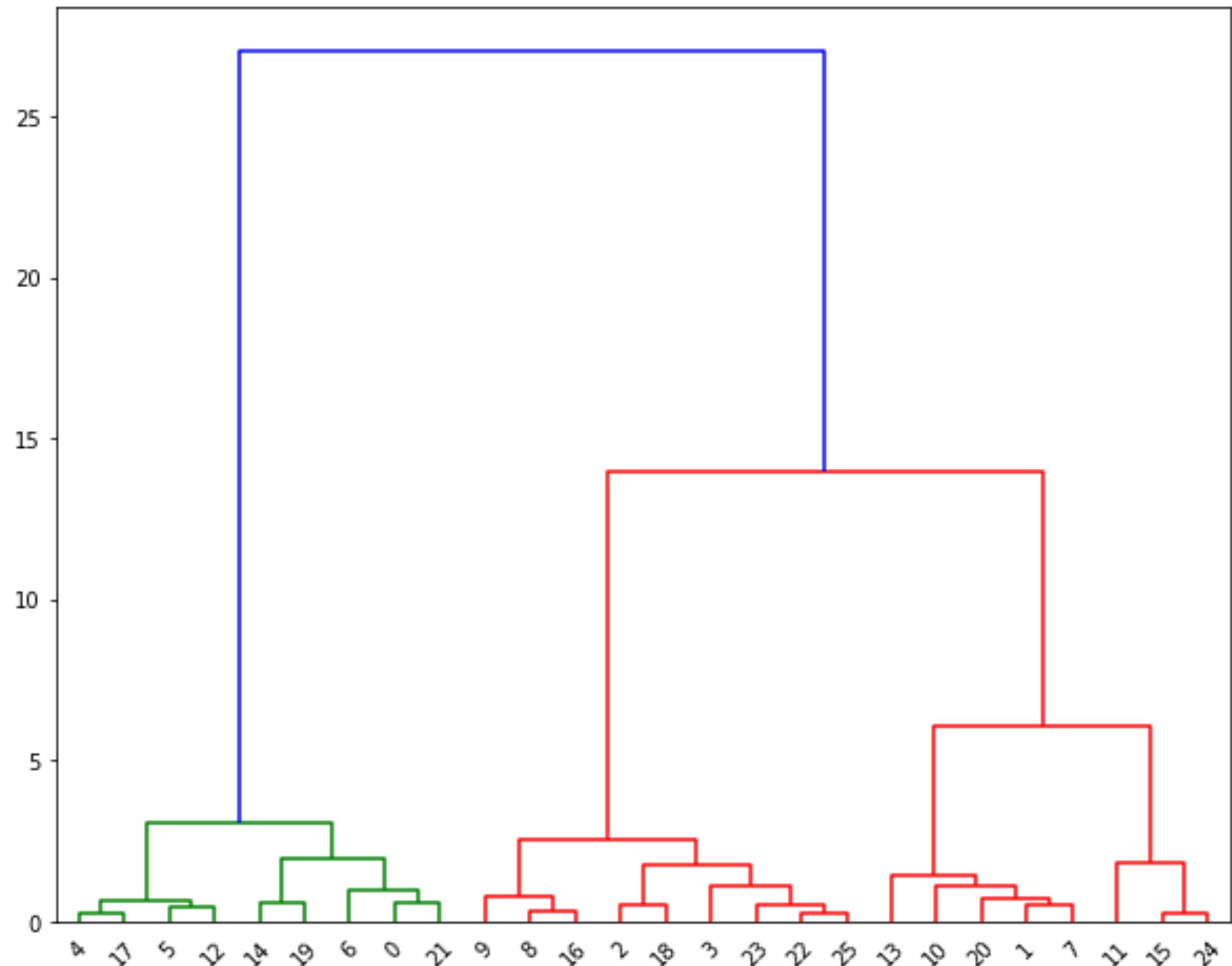


使用樹狀圖來了解群聚





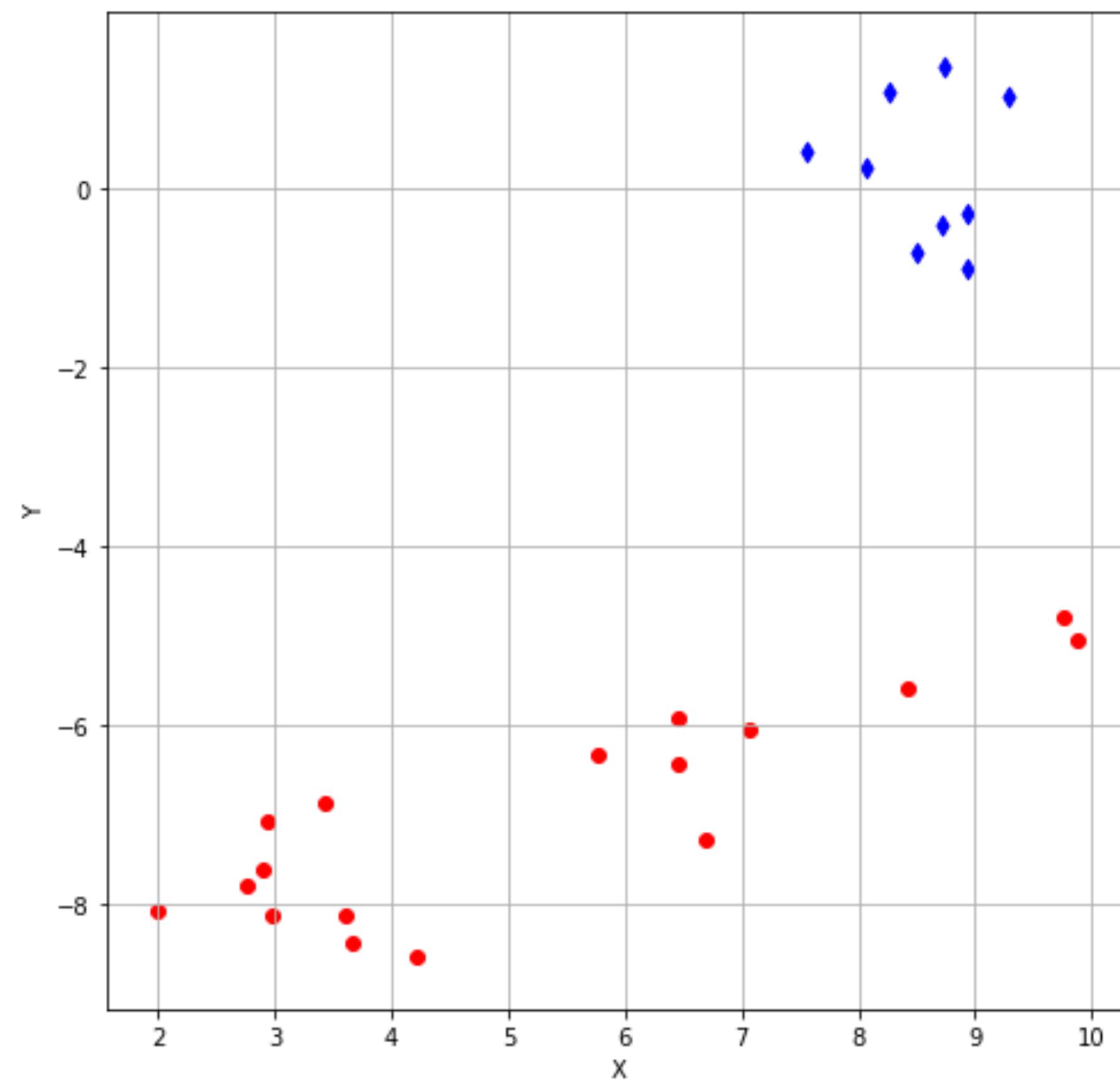
使用樹狀圖來了解群聚



- X軸代表樣本(遞增編號),Y軸代表距離.每一個群聚都代表被演算法合併的群聚.4和17被合併,5和12被合併,接著這兩組又會被群聚合併.
- 我們在距離為10的地方切割圖表,就可以得到兩個群聚



階層式分群





Python程式



```
34 # 建立資料集
35 X, Y = make_blobs(n_samples=nb_samples, n_features=2,
36                     centers=3, cluster_std=0.9)
37
38 # 顯示資料
39 fig, ax = plt.subplots(1, 1, figsize=(10, 8))
40
41 ax.grid()
42 ax.set_xlabel('X')
43 ax.set_ylabel('Y')
44
45 ax.scatter(X[:, 0], X[:, 1], marker='o', color='b')
46 plt.show()
47
48 # 計算距離矩陣
49 Xdist = pdist(X, metric='euclidean')
50
51 # 計算連接
52 Xl = linkage(Xdist, method='ward')
53
54 # 計算和顯示樹狀圖形
55 fig, ax = plt.subplots(1, 1, figsize=(10, 8))
56Xd = dendrogram(Xl)
57 plt.show()
```



Python程式

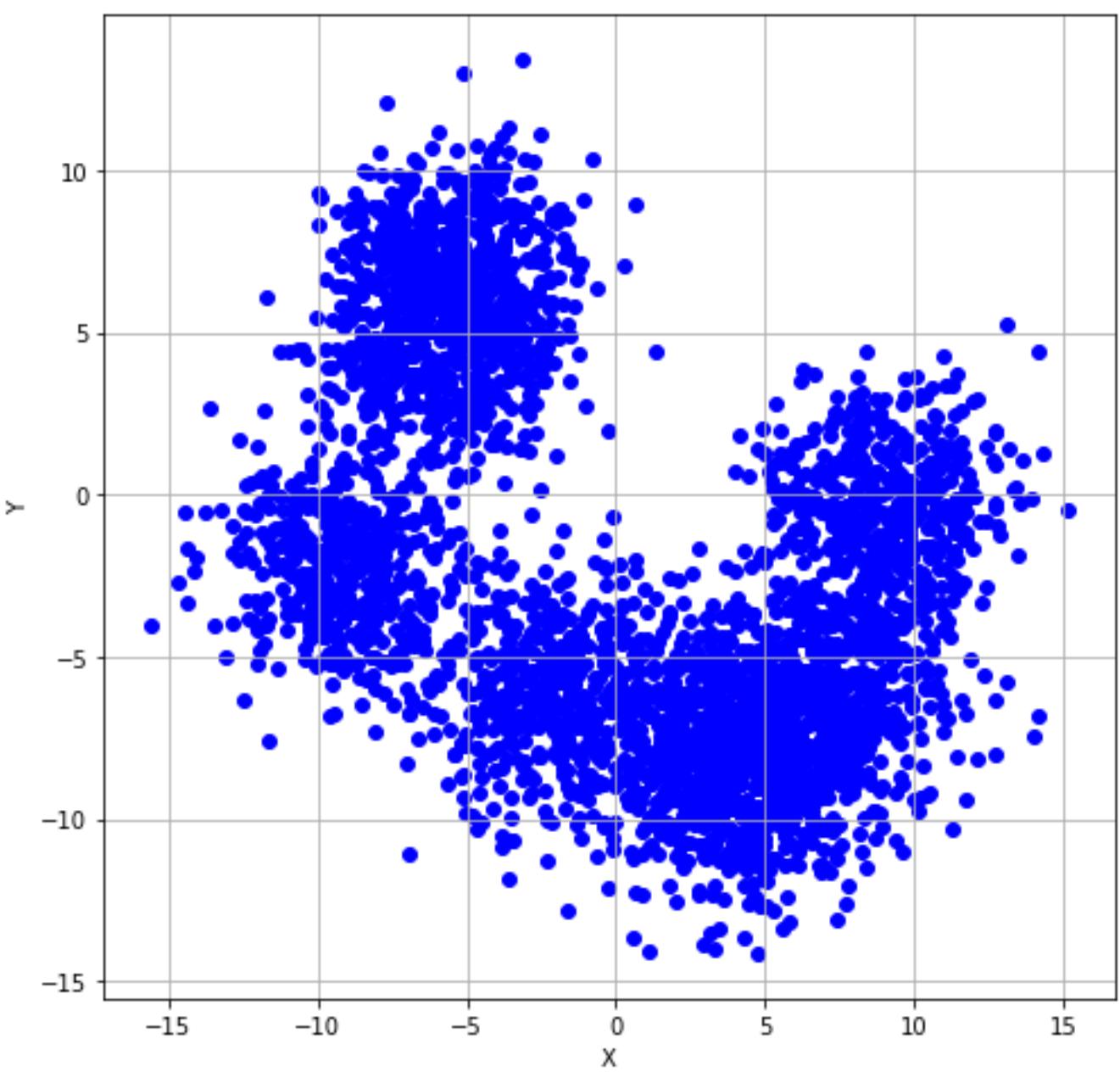


- 分兩群
- 完整complete連接演算法:將每一群聚合併,來將群聚間的最大距離最小化

```
59      # 完成聚合連接
60      print('Complete linkage')
61      ac = AgglomerativeClustering(n_clusters=2, linkage='complete')
62      Y = ac.fit_predict(X)
63
64      # 顯示群聚資料集
65      plot_clustered_dataset(X, Y)
```



建立高斯分佈資料集斑點





完整聚合連接分成八群



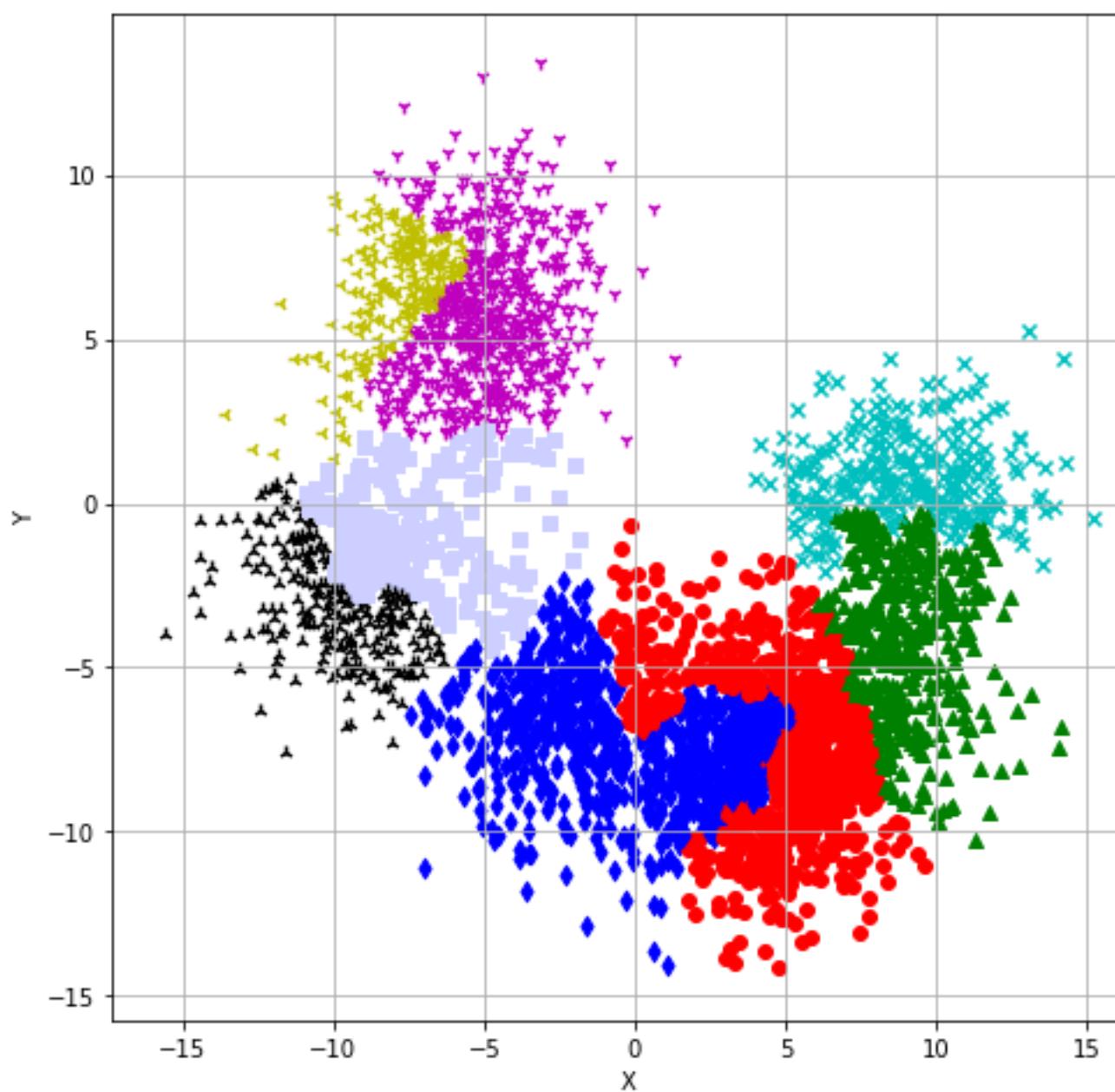
```
32 # 建立高斯分佈資料集斑點
33 X, _ = make_blobs(n_samples=nb_samples, n_features=2,
34                     centers=8, cluster_std=2.0)
35
36 # 顯示資料
37 fig, ax = plt.subplots(1, 1, figsize=(8, 8))
38
39 ax.grid()
40 ax.set_xlabel('X')
41 ax.set_ylabel('Y')
42
43 ax.scatter(X[:, 0], X[:, 1], marker='o', color='b')
44 plt.show()
45
46 # 完成聚合連接
47 print('Complete linkage')
48 ac = AgglomerativeClustering(n_clusters=8, linkage='complete')
49 Y = ac.fit_predict(X)
```



完成聚合連接



- 完整complete連接演算法:將每一群聚合併,來將群聚間的最大距離最小化





平均連接和沃德連接也分成八群



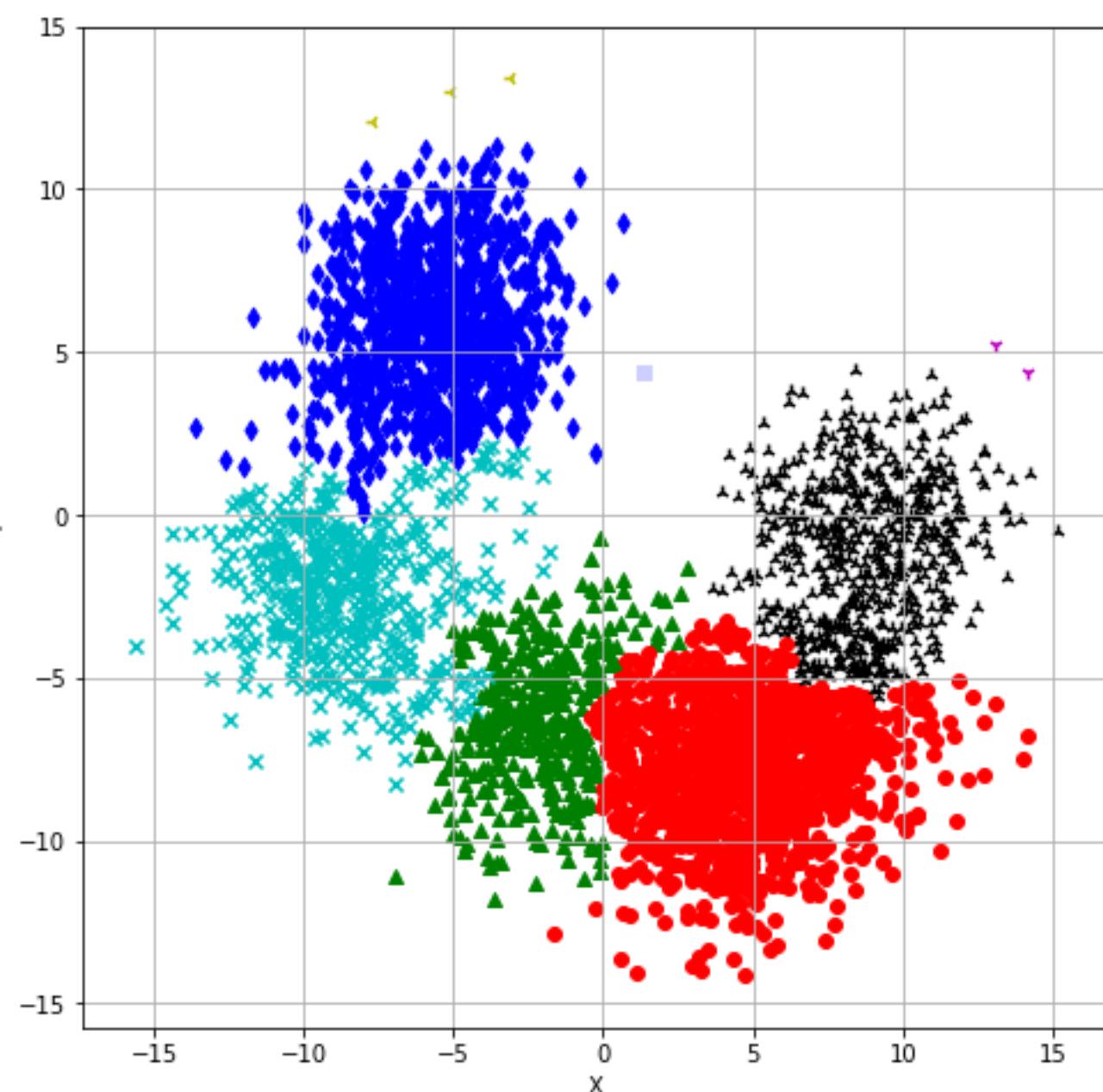
```
54 # 平均連接
55 print('Average linkage')
56 ac = AgglomerativeClustering(n_clusters=8, linkage='average')
57 Y = ac.fit_predict(X)
58
59 # 顯示群聚資料集
60 plot_clustered_dataset(X, Y)
61
62 # 沃德連接
63 print('Ward linkage')
64 ac = AgglomerativeClustering(n_clusters=8)
65 Y = ac.fit_predict(X)
66
67 # 顯示群聚資料集
68 plot_clustered_dataset(X, Y)
```



平均連接



- 平均連接演算法: 將每一群聚合併, 來將群聚間的平均距離最小化

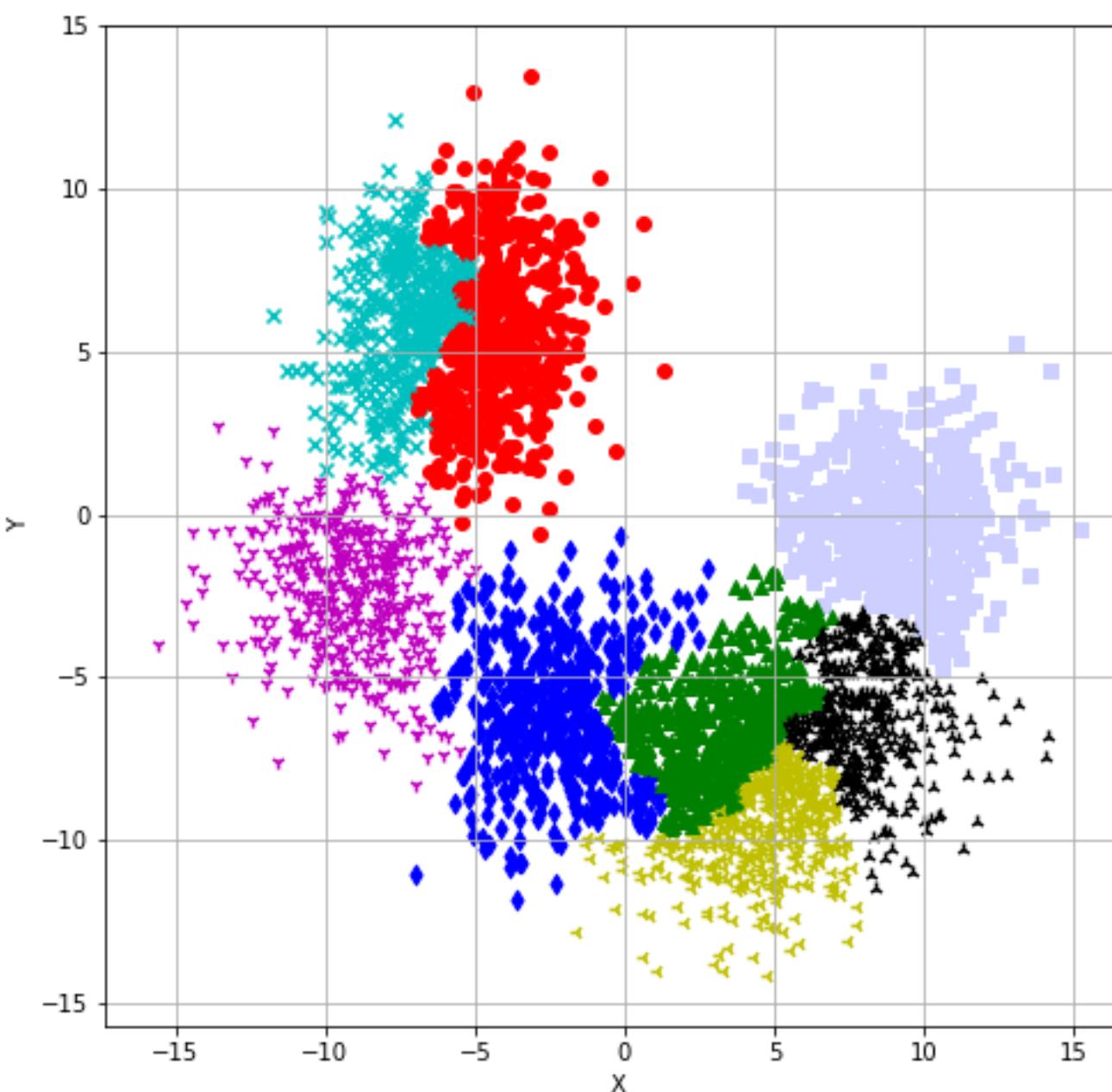




沃德連接



- 沃德連接:考慮所有群聚,計算群聚內的距離平方和,並合併群聚來將這個值最小化





密度為基礎的空間集群方法



- DBSCAN演算法:
 - 以密度為基礎的群聚,如果密度足夠,它就會被視為群聚的一部分,此時會考慮鄰居.如果它們也高密度,就將它們與第一個區域合併,否則決定了分群.
 - 當這個程序掃描完所有區域後,就可確定所有群聚.
- 主要特徵:
 - 發現任意形狀的群聚
 - 可以處理雜訊
 - 密度參數當作終止條件



密度為基礎的分群方法



- scikit-learn參數名詞：
 - 核心點p,q如果在指定半徑Eps之內包含了至少MinPts個相鄰樣本點
 - 邊緣點存在於核心點半徑Eps內,以邊緣點為中心的半徑Eps內包含小於MinPts個相鄰樣本點
 - 雜訊點Noise不是核心點也不是邊緣點



密度為基礎的分群演算法



- 對每個核心點p或是相連接的核心點q,如果兩個核心點它們的距離小於Eps,就當作為連接,而形成一個單獨的集群
- 對每一個邊緣點,指派到其相應的核心點集群

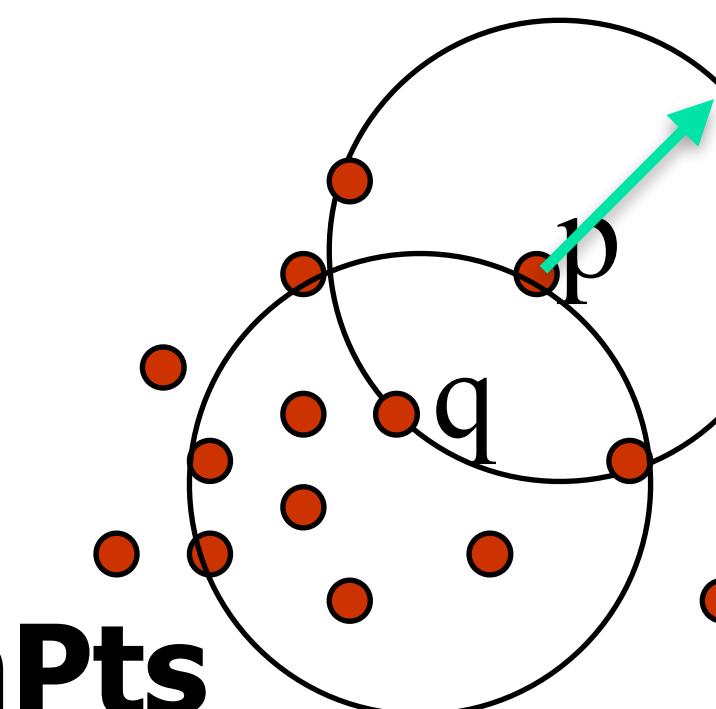


密度為基礎的分群方法



- scikit-learn 使用兩個參數控制群聚：
 - **Eps** 圓半徑：兩個鄰居間的最大距離，較大值會聚合較多點，較小值會建立較多群聚
 - **MinPts**：決定周圍需要多少點才可以定義一個區域
 - $N_{Eps}(p)$: $\{q \text{ 屬於 } D \mid \text{dist}(p,q) \text{ 距離} \leq \text{Eps 圓半徑}\}$
 - 直接密度可到達： p 直接密度可到達 q . **Eps** 半徑, **MinPts** 點數
假如
 - 1) p 屬於 $N_{Eps}(q)$
 - 2) 核心條件：

$$|N_{Eps}(q)| \geq \text{MinPts}$$



$\text{MinPts} = 5$

$\text{Eps} = \text{為圓半徑}$



Python程式,空間群聚DBSCAN

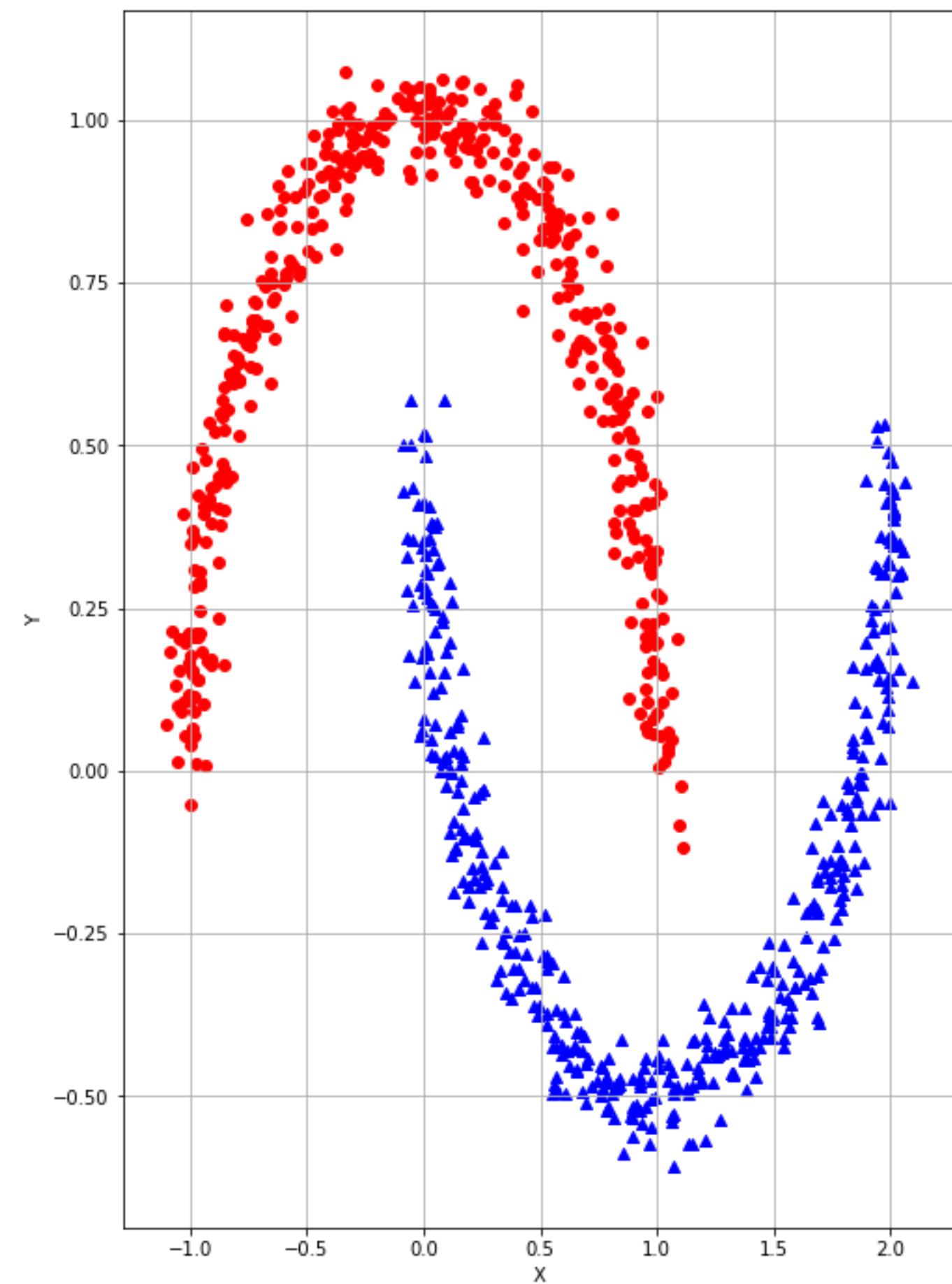


- # 建立和訓練密度為基礎的空間群聚DBSCAN
- #eps為定義兩個鄰居間的最大距離

```
48 if __name__ == '__main__':
49     # 建立資料集半月形資料
50     X, Y = make_moons(n_samples=nb_samples, noise=0.05)
51
52     # 顯示資料
53     show_dataset(X, Y)
54
55     # 建立和訓練密度為基礎的空間群聚DBSCAN
56     #eps為定義兩個鄰居間的最大距離
57     dbs = DBSCAN(eps=0.1)
58     Y = dbs.fit_predict(X)
59
60     # 顯示群聚資料集
61     show_clustered_dataset(X, Y)
```

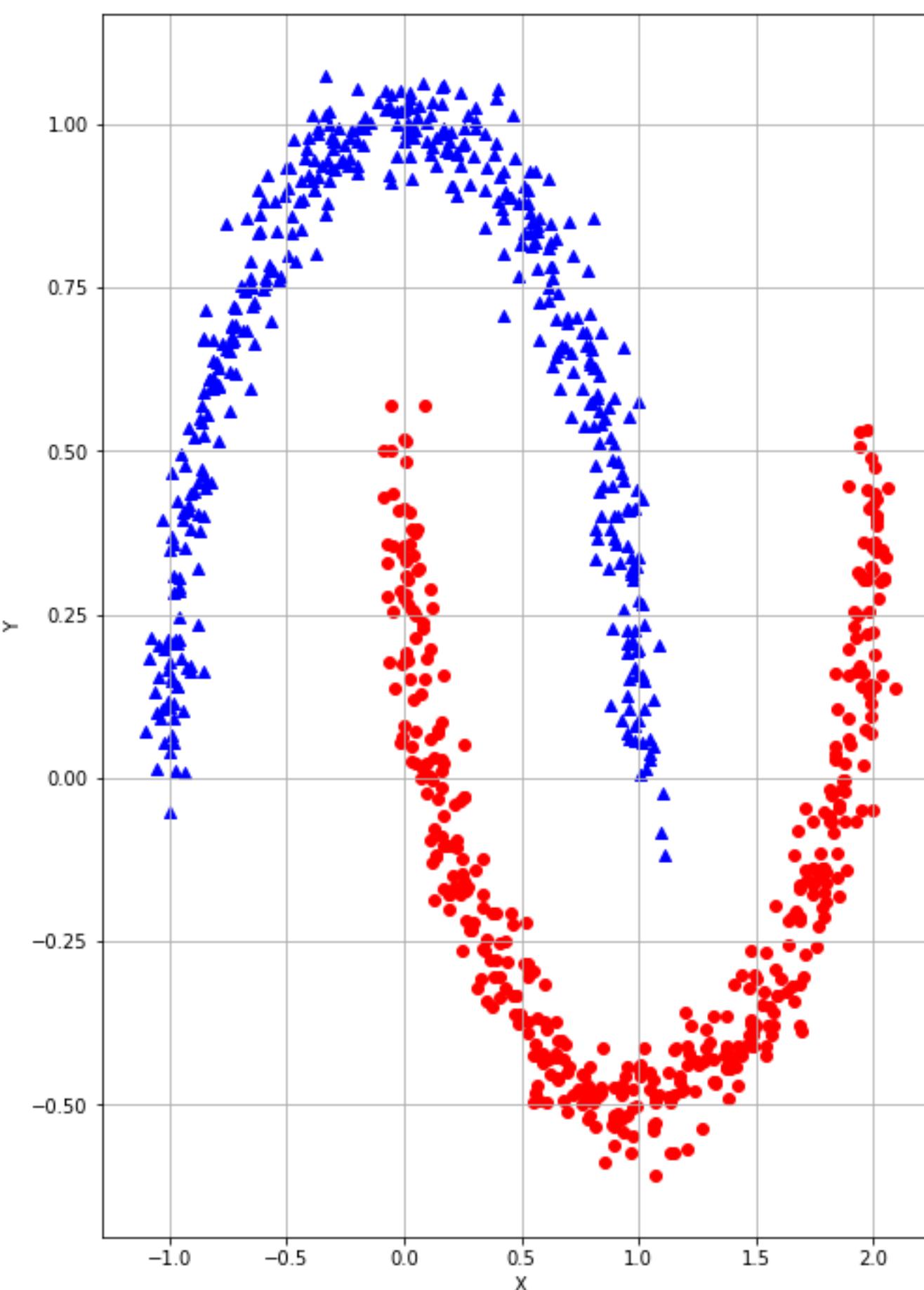


建立資料集半月形資料





預測群聚的結果





python



- Thank You



基本機器學習演算法



- 線性迴歸
- Logistic邏輯迴歸
- 單純貝氏分類器(Naive Bayes Classifiers)
- 分群降維使用主成份分析PCA
- 分類降維使用線性判別分析



線性迴歸



- a,b為未知數,求二元一次方程式

$$\hat{y} = ax + b$$



線性迴歸



- LOSS函數最小化

$$L = \frac{1}{2} \sum_{i=1}^n (ax + b - y_i)^2$$



線性迴歸



- from scipy.optimize import minimize

```
9 import numpy as np
10 import matplotlib.pyplot as plt
11
12 from scipy.optimize import minimize
13
14
15
16 np.random.seed(1000)
17
18
19 nb_samples = 400
20
21
22 def show_dataset(X, Y):
23     fig, ax = plt.subplots(1, 1, figsize=(15, 10))
24
25     ax.scatter(X, Y)
26     ax.set_xlabel('X')
27     ax.set_ylabel('Y')
28     ax.grid()
```



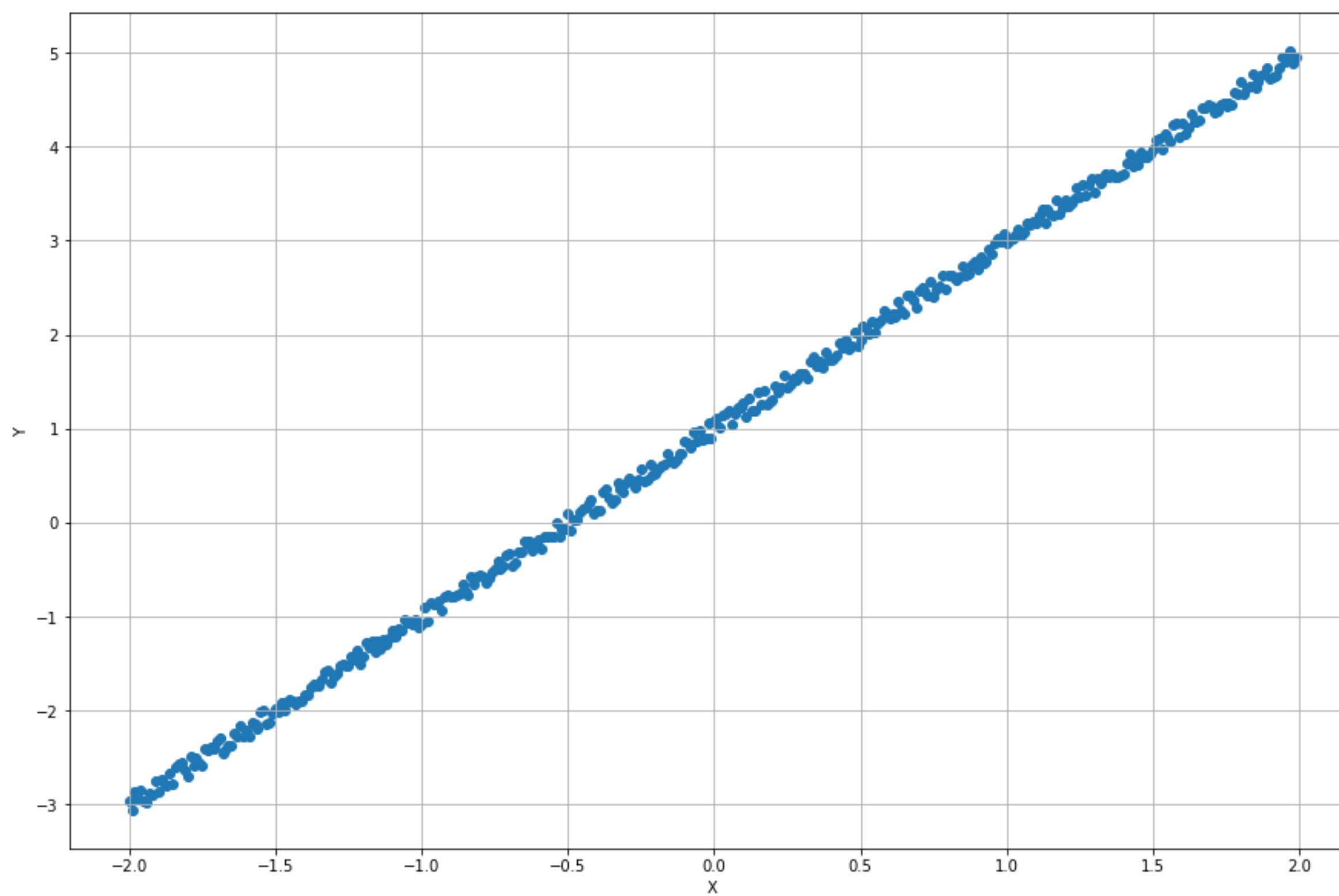
```
33 def gradient(v):
34     gr = np.zeros(shape=2)
35     for i in range(nb_samples):
36         gr[0] += (v[0] + v[1]*X[i] - Y[i])
37         gr[1] += ((v[0] + v[1]*X[i] - Y[i]) * X[i])
38     return gr
39
40
41 def loss(v):
42     err = 0.0
43     for i in range(nb_samples):
44         err += np.square(v[0] + v[1]*X[i] - Y[i])
45     return 0.5 * err
```



```
48 if __name__ == '__main__':
49
50     X = np.arange(-2, 2, 0.01)
51     #y=ax+b, a=2, b=1
52     Y = 2*X+1
53     Y += np.random.uniform(-0.1, 0.1, size=nb_samples)
54     show_dataset(X, Y)
55     # 最小化成本函數
56     result = minimize(fun=loss, x0=np.array([0.0, 0.0]),
57                        jac=gradient, method='L-BFGS-B')
58     print('斜率：')
59     print('y = %.2fx + %.2f' % (result.x[1], result.x[0]))
60     # 計算均方差
61     err = 0.0
62     for i in range(nb_samples):
63         err += np.abs(Y[i] - (result.x[1]*X[i] + result.x[0]))
64
65     print('絕對均方差: %.2f' % err)
```



python





Logistic邏輯迴歸



- 某個樣本屬於某個類別的機率(0~1)

$$y = \sigma(x) = \frac{1}{1 + e^{-x}}$$

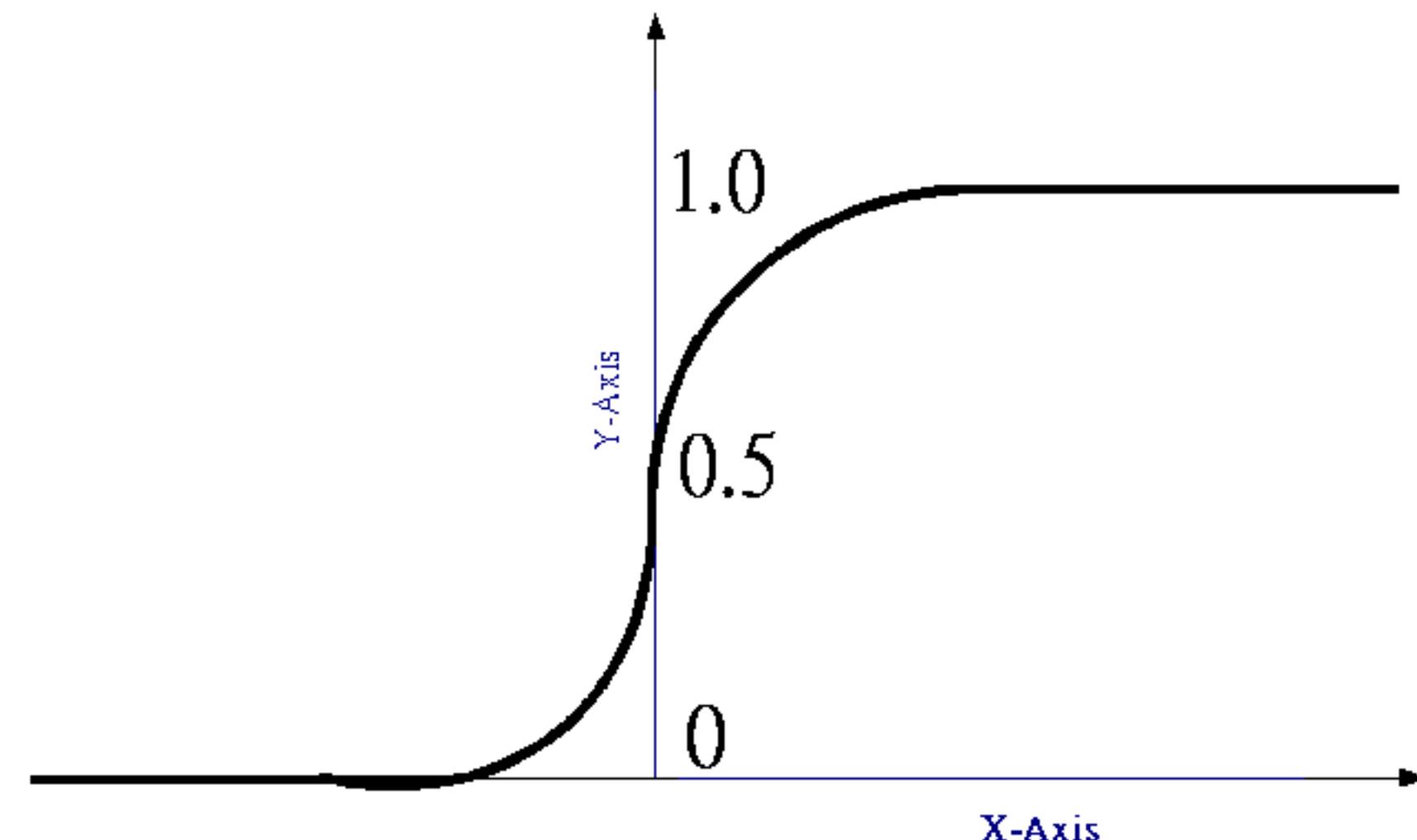


Logistic邏輯迴歸



- 可以用Sigmoid(Logistic)邏輯迴歸
- 當x為0時,y為0.5

$$y = \sigma(x) = \frac{1}{1 + e^{-x}}$$





Logistic邏輯迴歸



- 使用sklearn.linear_model的LogisticRegression
- 500個樣本

```
9 import numpy as np
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.datasets import make_classification
12 from sklearn.model_selection import train_test_split, cross_val_score
13 import matplotlib.pyplot as plt
14
15
16# For reproducibility
17np.random.seed(1000)
18
19nb_samples = 500
20
```



Logistic邏輯迴歸



- 顯示資料

```
22 def show_dataset(X, Y):  
23     fig, ax = plt.subplots(1, 1, figsize=(10, 9))  
24     ax.grid()  
25     ax.set_xlabel('X')  
26     ax.set_ylabel('Y')  
27     for i in range(nb_samples):  
28         if Y[i] == 0:  
29             ax.scatter(X[i, 0], X[i, 1], marker='o', color='r')  
30         else:  
31             ax.scatter(X[i, 0], X[i, 1], marker='^', color='b')  
32     plt.show()
```



Logistic邏輯迴歸



- 顯示分類區域, LogisticRegression().predict()預測資料的類別

```
34 def show_classification_areas(X, Y, LoReg):  
35     x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5  
36     y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5  
37     xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),  
38                           np.arange(y_min, y_max, 0.02))  
39     Z = LoReg.predict(np.c_[xx.ravel(), yy.ravel()])  
40  
41     Z = Z.reshape(xx.shape)  
42     plt.figure(1, figsize=(10, 9))  
43     plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Pastel1)  
44     plt.scatter(X[:, 0], X[:, 1], c=np.abs(Y - 1),  
45                  edgecolors='k', cmap=plt.cm.coolwarm)  
46     plt.xlabel('X')  
47     plt.ylabel('Y')  
48     plt.xlim(xx.min(), xx.max())  
49     plt.ylim(yy.min(), yy.max())  
50     plt.xticks()  
51     plt.yticks()  
52     plt.show()
```



Logistic邏輯迴歸



- 使用LogisticRegression()來建立Logistic邏輯迴歸

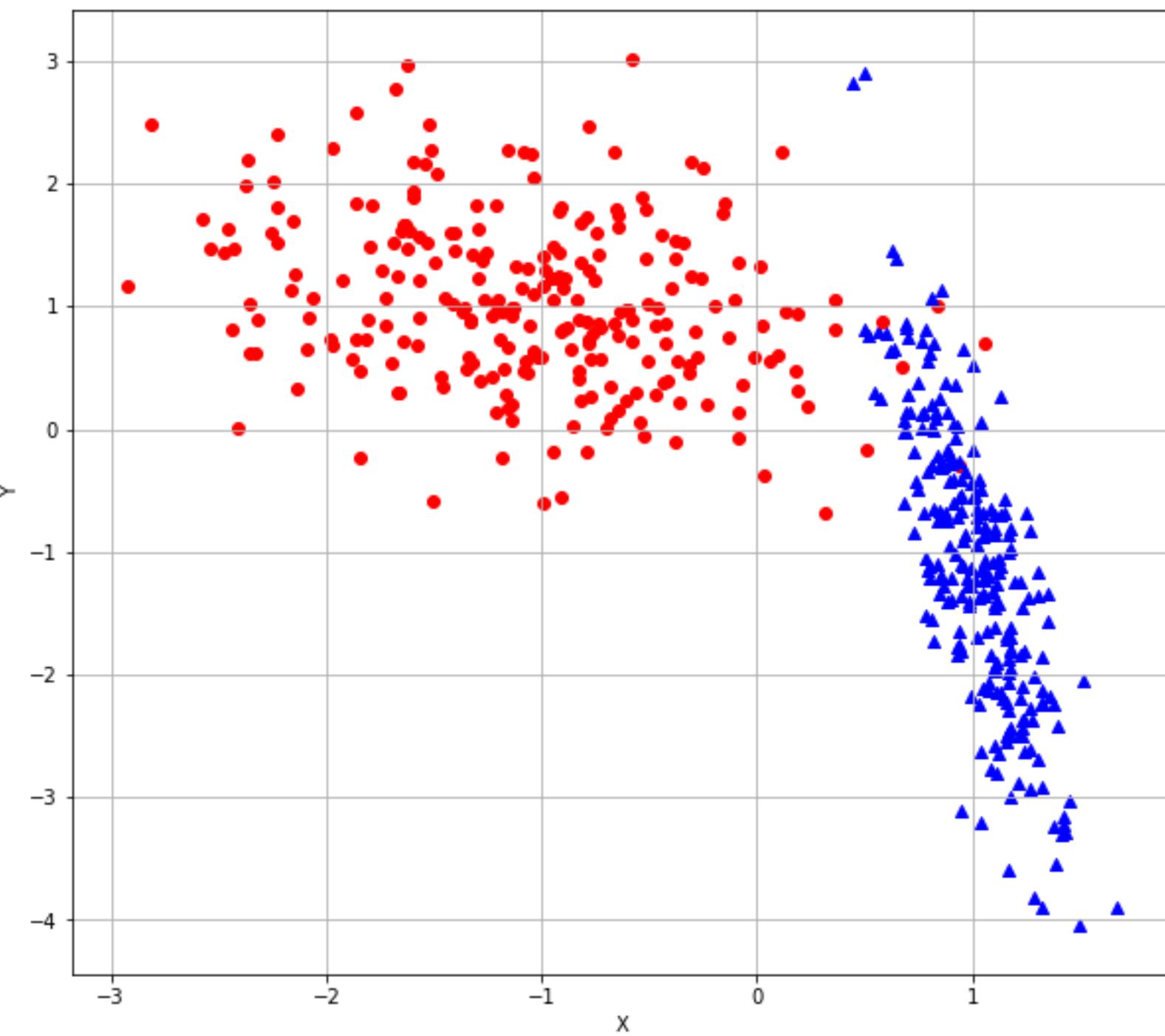
```
58 if __name__ == '__main__':
59     # 建立資料
60     X, Y = make_classification(n_samples=nb_samples, n_features=2,
61                               n_informative=2, n_redundant=0,
62                               n_clusters_per_class=1)
63     show_dataset(X, Y)
64     X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
65                                                       test_size=0.25)
66     # 建立logistic迴歸
67     LoReg = LogisticRegression()
68     LoReg.fit(X_train, Y_train)
69     print('logistic迴歸 成績: %.3f' % LoReg.score(X_test, Y_test))
70     # 計算交叉值成績
71     lr_scores = cross_val_score(LoReg, X, Y, scoring='accuracy',
72                                 cv=10)
73     print('logistic迴歸 交叉值成績: %.3f' % lr_scores.mean())
74     # 顯示分類區域
75     show_classification_areas(X, Y, LoReg)
```



Logistic邏輯迴歸



- 500個樣本

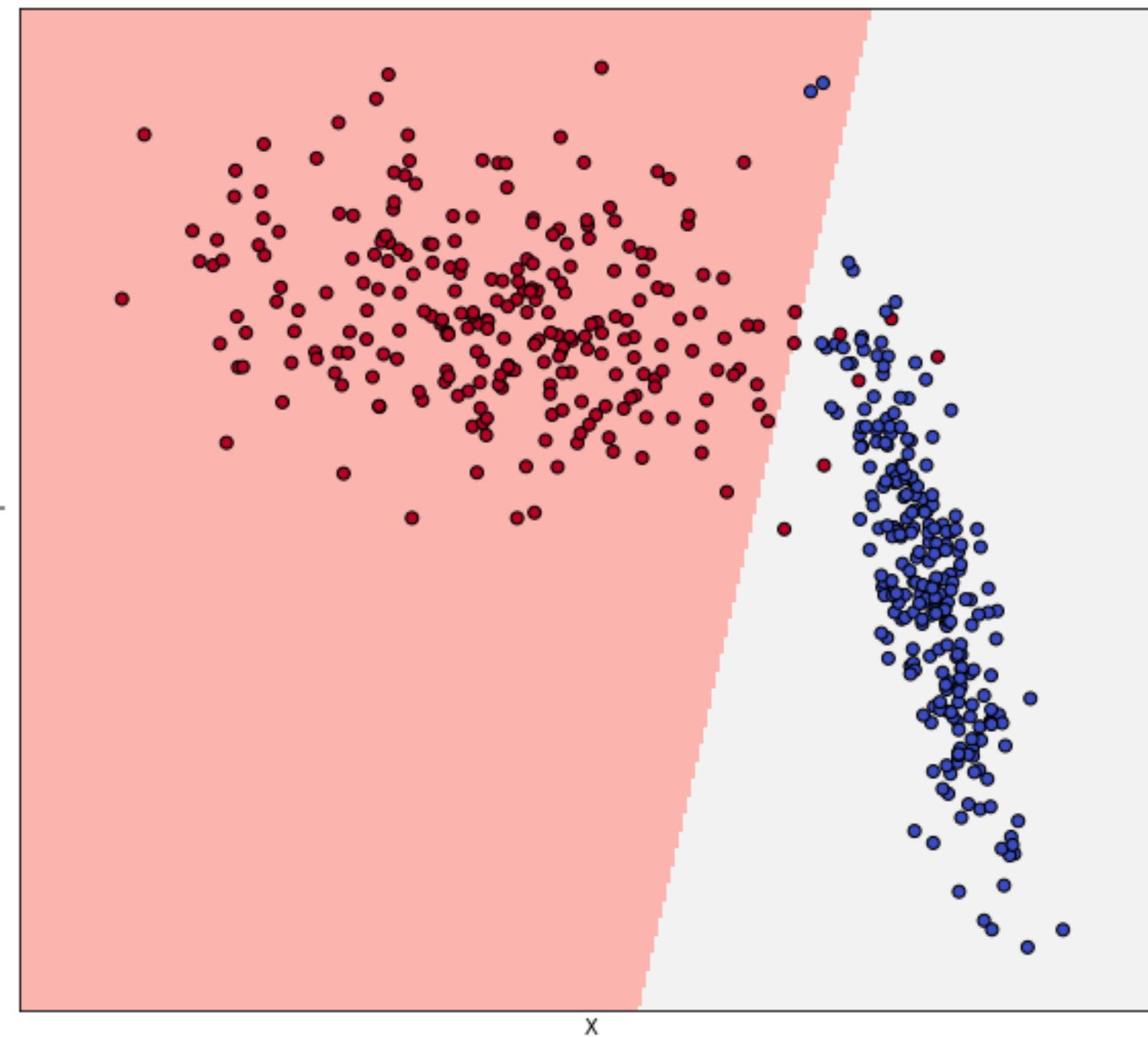




Logistic邏輯迴歸



- 顯示分類區域





單純貝氏分類器

(Naive Bayes Classifiers)



$$P(A \cap B) = P(A|B)P(B)$$

$$P(B \cap A) = P(B|A)P(A)$$

整理與合併這兩個方程式，我們可以得到

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A)$$



單純貝氏分類器

(Naive Bayes Classifiers)



貝氏定理是關於隨機事件A和B的條件機率的一則定理。
其中 $P(A|B)$ 是在B發生的情況下A發生的可能性。

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



單純貝氏分類器

(Naive Bayes Classifiers)



$P(A|B)$ 是已知B發生後A的條件機率。

$P(B|A)$ 是已知A發生後B的條件機率。

$P(A)$ 是A的先驗機率（或邊緣機率）。

之所以稱為"先驗"是因為它不考慮任何B方面的因素。

$P(B)$ 是B的先驗機率或邊緣機率。

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



單純貝氏分類器



單純貝氏分類器是以假設特徵之間（樸素）獨立下運用貝葉斯定理為基礎的簡單機率分類器。
單純貝氏是一種構建分類器的簡單方法。該分類器模型會給問題實例分配用特徵值表示的類標籤。



單純貝氏分類器



- 機率模型分類器是一個條件機率模型。
- 獨立的類別變數 y 有 k 個類別，條件依賴於若干特徵變數 z 。 z 為 m 維度的變數。



單純貝氏分類器



- 機率模型分類器是一個條件機率模型。

$$X = \{ \overline{x_1}, \overline{x_2}, \overline{x_3}, \dots, \overline{x_n} \}, \overline{x_i} \in \Re^m$$



單純貝氏分類器



- 獨立的類別變數 y 有 k 個類別，條件依賴於若干特徵變數 z 。 z 為 m 維度的變數。
- 特徵向量空間有 m 維度

$$\underline{x}_i = \{ \underline{z}_1, \underline{z}_2, \dots, \underline{z}_m \}$$

$$Y = \{y_1, y_2, \dots, y_j\}, y_j \in \{0, 1, 2, \dots k\text{類}\}$$



單純貝氏分類器



- 重複使用鏈式法則，可將該式寫成條件機率的形式
- 則單純貝氏機率模型為

$$P(y|z_1, z_2, \dots, z_m) \propto P(y) \prod_i P(z_i | y)$$



白努力單純貝氏



- 假設只有0和1的值,且它們的機率為

$$P(x) = \begin{cases} q & \text{if } X=0 \\ p & \text{if } X=1 \end{cases}, p=1-q, 0 < p < 1$$



單純貝氏分類器

(Naive Bayes Classifiers)



```
8 import numpy as np
9 from sklearn.datasets import make_classification
10 from sklearn.model_selection import train_test_split
11 from sklearn.model_selection import cross_val_score
12 from sklearn.naive_bayes import BernoulliNB
13 import matplotlib.pyplot as plt
```



單純貝氏分類器

(Naive Bayes Classifiers)



```
21 def show_dataset(X, Y):
22     fig, ax = plt.subplots(1, 1, figsize=(20, 12))
23
24     ax.grid()
25     ax.set_xlabel('X')
26     ax.set_ylabel('Y')
27
28     for i in range(nb_samples):
29         if Y[i] == 0:
30             ax.scatter(X[i, 0], X[i, 1], marker='o', color='r')
31         else:
32             ax.scatter(X[i, 0], X[i, 1], marker='^', color='b')
33
34     plt.show()
```



單純貝氏分類器

(Naive Bayes Classifiers)



```
37 if __name__ == '__main__':
38     # 建立資料
39     X, Y = make_classification(n_samples=nb_samples,
40                               n_features=2, n_informative=2, n_redundant=0)
41     show_dataset(X, Y)
42     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)
43
44     # 建立和訓練白努力單純貝氏Naive Bayes分類
45     bnb = BernoulliNB(binarize=0.0)
46     bnb.fit(X_train, Y_train)
47
48     print('白努力單純貝氏: %.3f' % bnb.score(X_test, Y_test))
49
50     bnb_scores = cross_val_score(bnb, X, Y, scoring='accuracy', cv=10)
51     print('白努力單純貝氏交叉值: %.3f' % bnb_scores.mean())
52
53     # 預測值
54     data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
55     Yp = bnb.predict(data)
56     print(Yp)
```

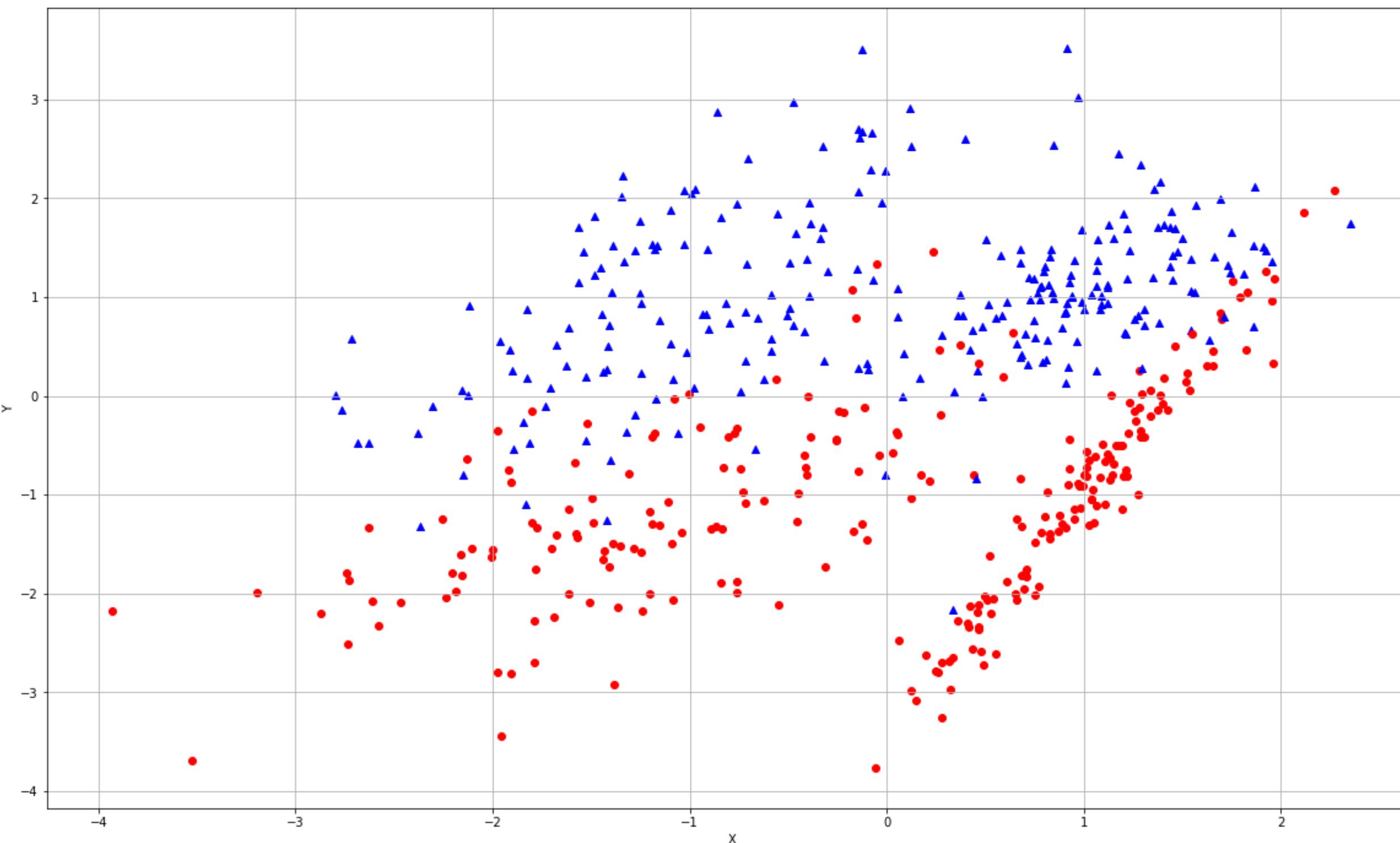


單純貝氏分類器

(Naive Bayes Classifiers)



- 白努力單純貝氏: 0.864
- 白努力單純貝氏交叉值: 0.878
- [0 1 0 1]





降維



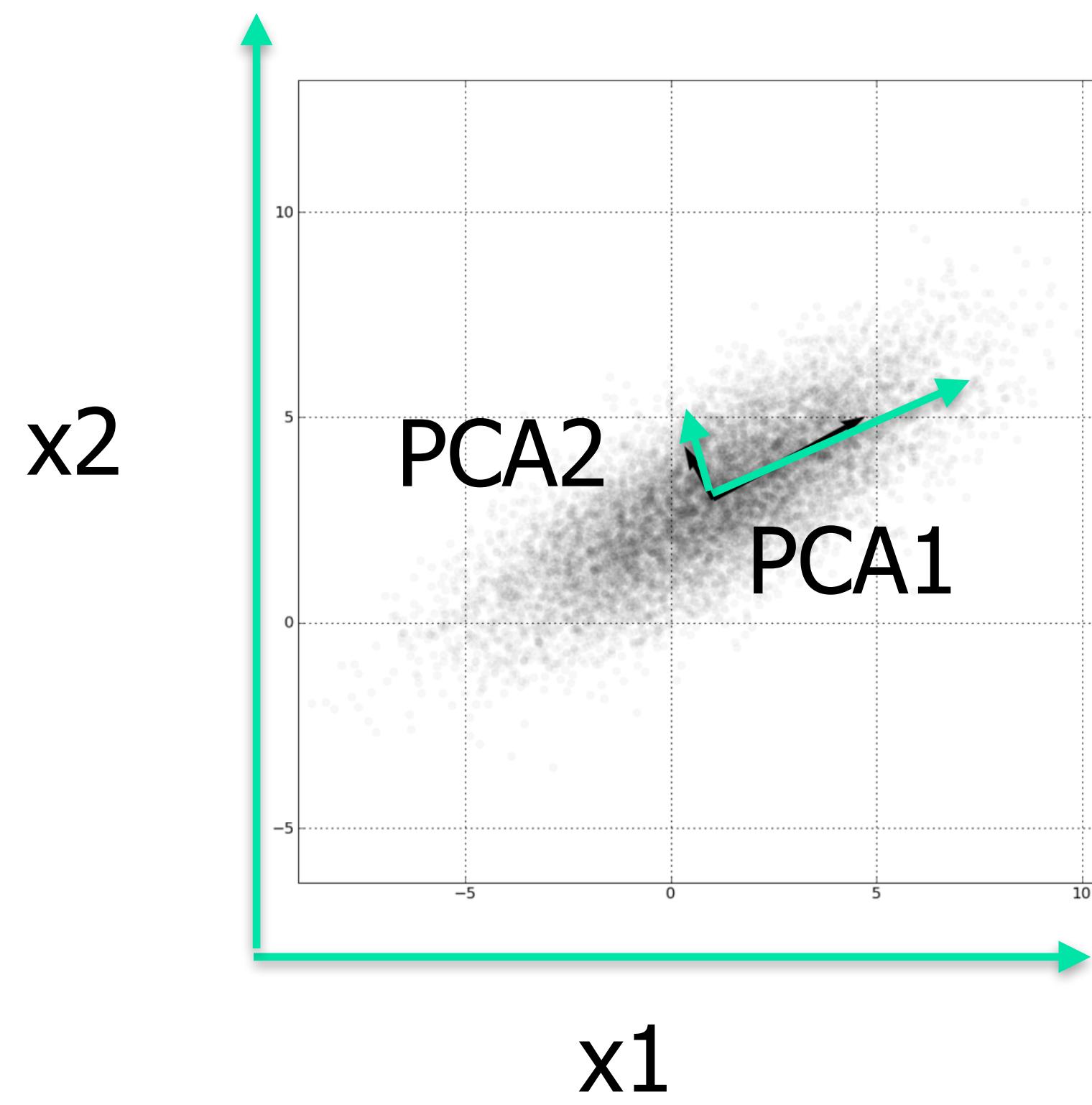
- 分群降維使用主成份分析
- 分類降維使用線性判別分析



分群降維使用主成份分析PCA



主成分分析經常用於減少數據集的維數，同時保持數據集中的對變異數貢獻最大的特徵。





分群降維使用主成份分析PCA



主成分分析PCA是一種分析、簡化數據集的技術。PCA是使用矩陣投影方式將n維度樣本投影到m維度空間使用W轉換矩陣,其中m<<n。

$$x = \{x_1, x_2, \dots, x_n\}, x \in R^n$$

$$\downarrow xW, W \in R^{n*m}$$

$$z = \{z_1, z_2, \dots, z_m\}, z \in R^m$$



分群降維使用主成份分析PCA



- 輸入主成份分析PCA

```
9 import numpy as np
10 from sklearn.datasets import load_digits
11 from sklearn.decomposition import PCA
12 import matplotlib.pyplot as plt
```



PCA主成份分析



```
18 digits = load_digits()
19 selection = np.random.randint(0, 1500, size=100)
20 fig, ax = plt.subplots(10, 10, figsize=(10, 10))
21 samples = [digits.data[x].reshape((8, 8)) for x in selection]
22 for i in range(10):
23     for j in range(10):
24         ax[i, j].set_axis_off()
25         ax[i, j].imshow(samples[(i * 8) + j], cmap='gray')
26 plt.show()
27 pca = PCA(n_components=36, whiten=True)
28 X_pca = pca.fit_transform(digits.data / 255)
29 print('解釋變異數比')
30 print(pca.explained_variance_ratio_)
31 fig, ax = plt.subplots(1, 2, figsize=(12, 6))
32 ax[0].set_xlabel('Component')
33 ax[0].set_ylabel('Variance ratio (%)')
34 ax[0].bar(np.arange(36), pca.explained_variance_ratio_ * 100.0)
35 ax[1].set_xlabel('Component')
36 ax[1].set_ylabel('Cumulative variance (%)')
37 ax[1].bar(np.arange(36), np.cumsum(pca.explained_variance_)[:-1])
38
39 plt.show()
```



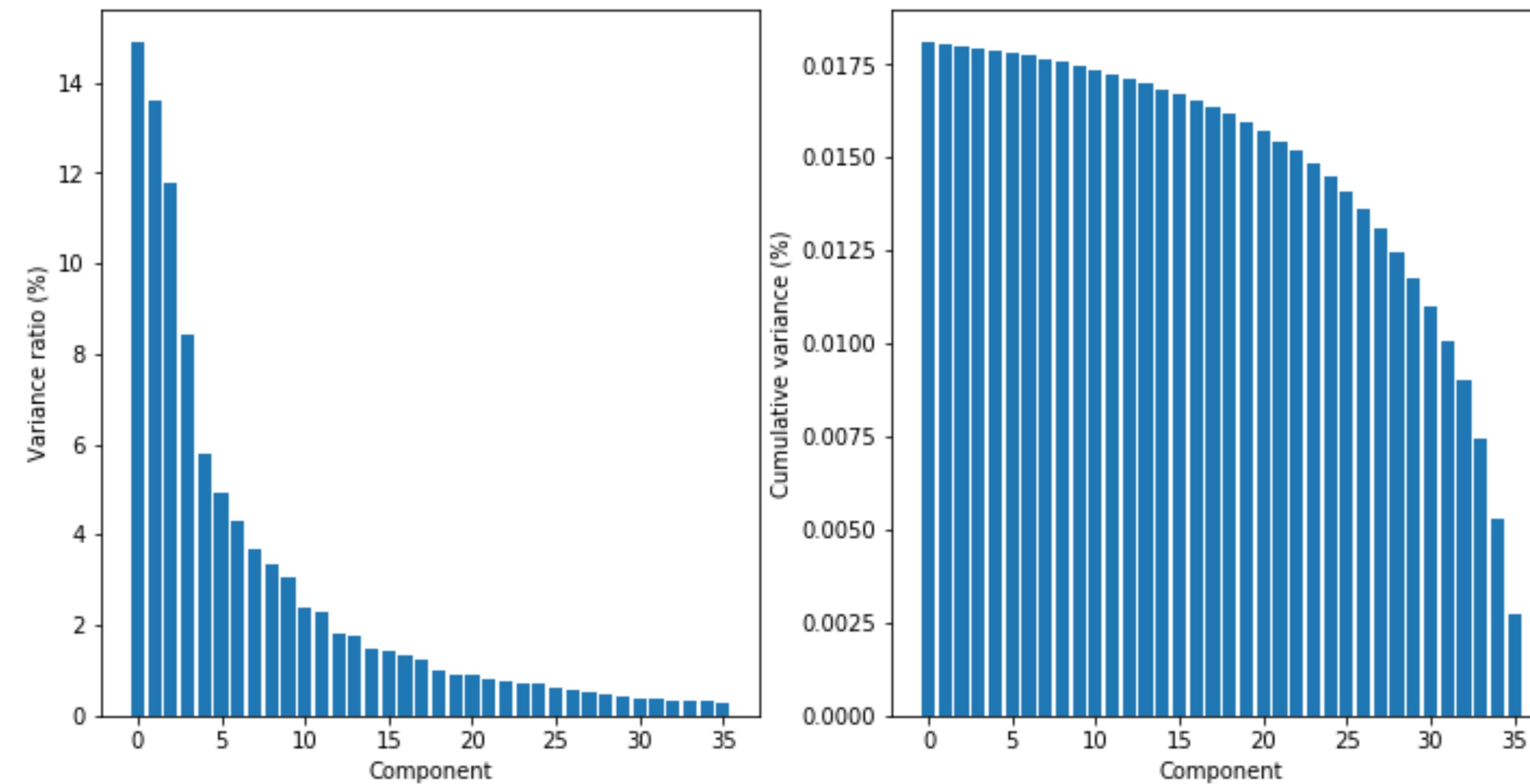
分群降維使用主成份分析PCA



- 手寫辨識使用前36種成分
- 重構圖形

```
41     samples = [pca.inverse_transform(X_pca[x]).reshape((8, 8)) for x in selection]
42
43     for i in range(10):
44         for j in range(10):
45             ax[i, j].set_axis_off()
46             ax[i, j].imshow(samples[(i * 8) + j], cmap='gray')
47
48 plt.show()
```

- 變異數比率及累積變異數





分類降維使用LDA線性判別分析



試圖找到兩類物體或事件的特徵的一個線性組合，以能夠特徵化或區分它們。所得的組合可用來作為一個線性分類器，為後續的分類做降維處理。



用在監督式分類降維的LDA



- from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

```
8 import matplotlib.pyplot as plt
9
10 from sklearn import datasets
11 from sklearn.decomposition import PCA
12 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
13
14 iris = datasets.load_iris()
15
16 X = iris.data
17 y = iris.target
18 target_names = iris.target_names
19
20 pca = PCA(n_components=2)
21 X_r = pca.fit(X).transform(X)
22
23 lda = LinearDiscriminantAnalysis(n_components=2)
24 X_r2 = lda.fit(X, y).transform(X)
```



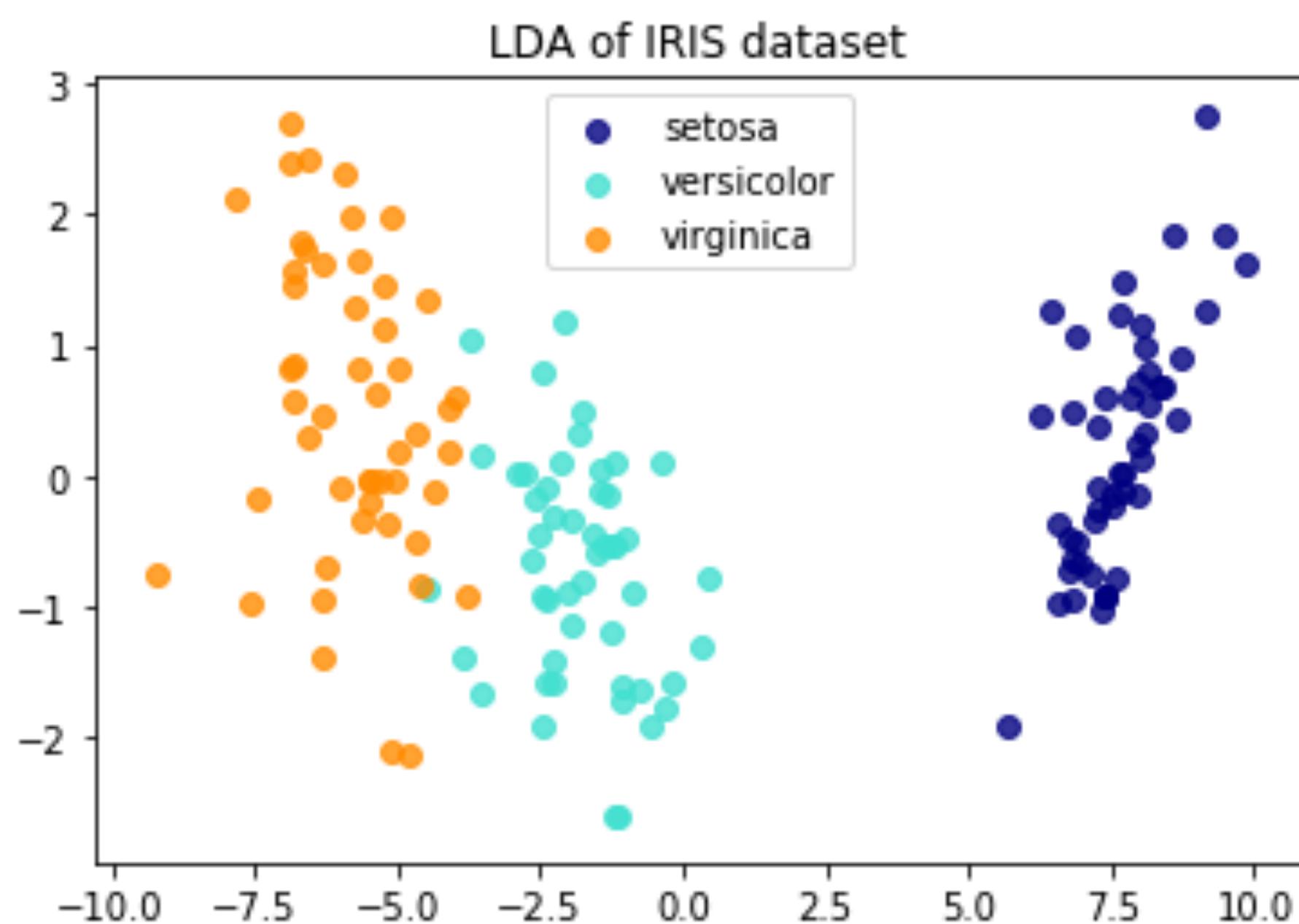
IRIS鳶尾花線性分類降維演算法



```
30 plt.figure()
31 colors = ['navy', 'turquoise', 'darkorange']
32 lw = 2
33
34 for color, i, target_name in zip(colors, [0, 1, 2], target_names):
35     plt.scatter(X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=.8, lw=lw,
36                 label=target_name)
37 plt.legend(loc='best', shadow=False, scatterpoints=1)
38 plt.title('PCA of IRIS dataset')
39
40 plt.figure()
41 for color, i, target_name in zip(colors, [0, 1, 2], target_names):
42     plt.scatter(X_r2[y == i, 0], X_r2[y == i, 1], alpha=.8, color=color,
43                 label=target_name)
44 plt.legend(loc='best', shadow=False, scatterpoints=1)
45 plt.title('LDA of IRIS dataset')
```



IRIS鳶尾花線性分類降維演算法





python



- Thank You



Python 機器學習

吳佳諺 老師





Python 機器學習

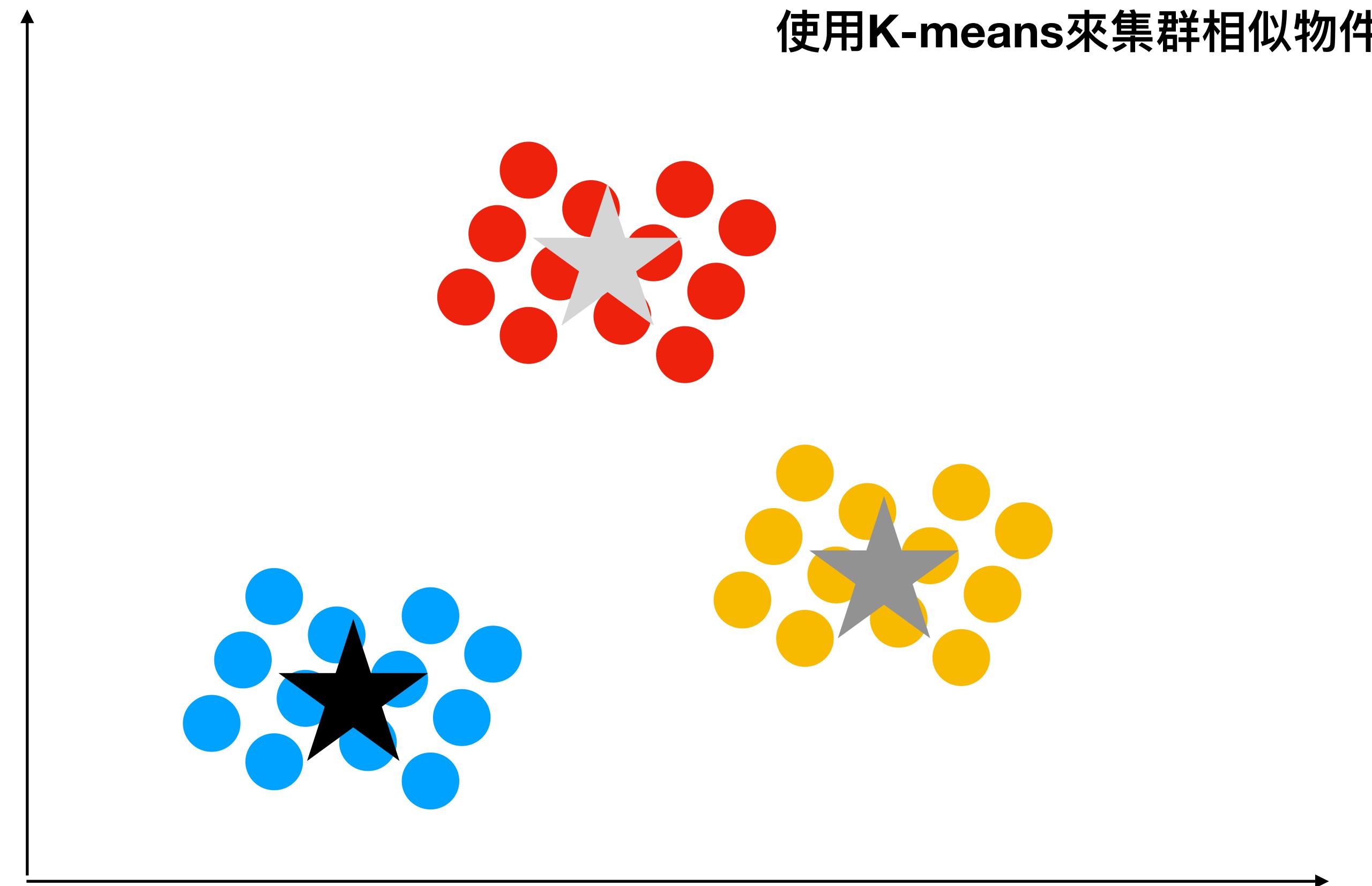
1. Python 機器學習
2. 認知演算法
3. 鳶尾花
4. 加州大學鳶尾花資料
5. 決策區域函數
6. 建立模型 Perceptron(eta=0.1, n_iter=10).fit()



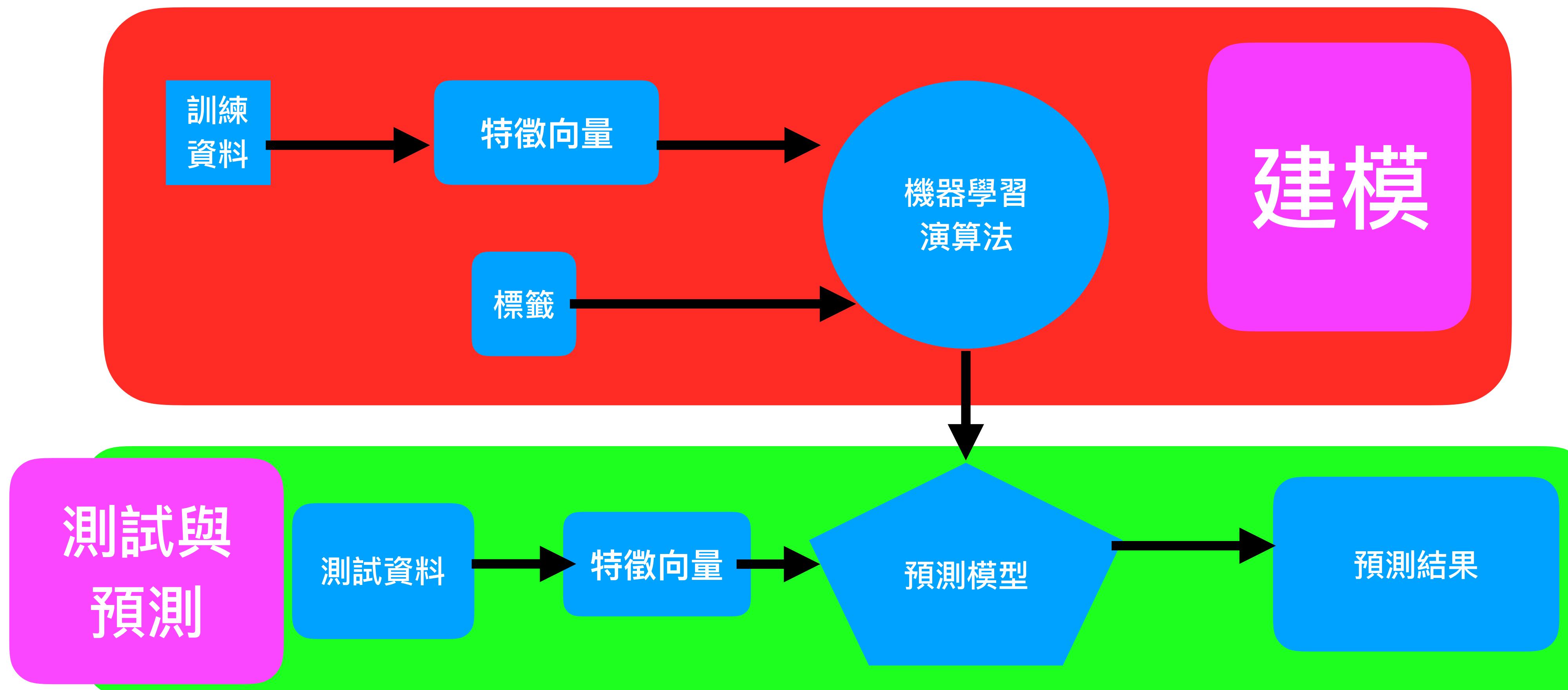
1. Python 機器學習

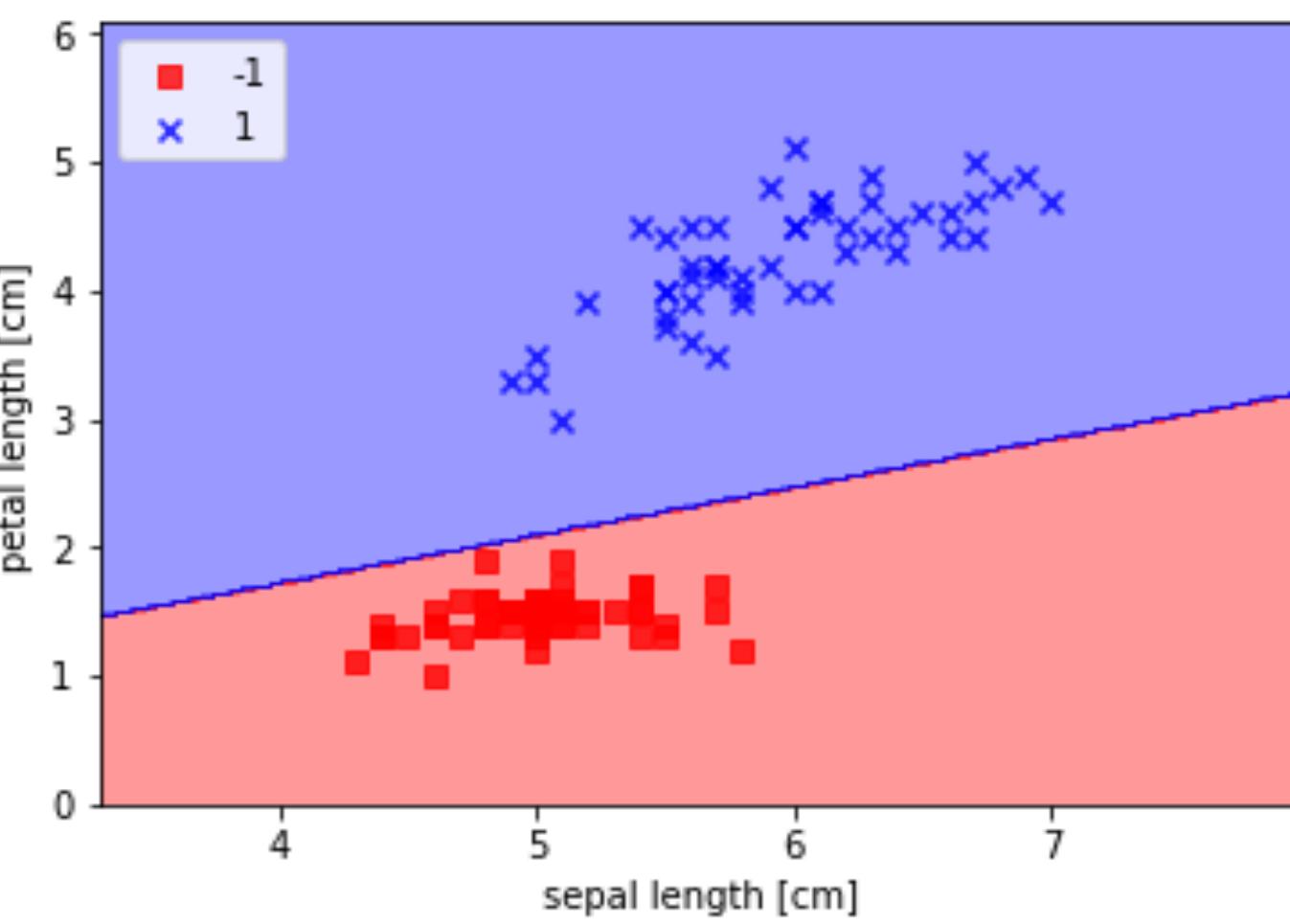
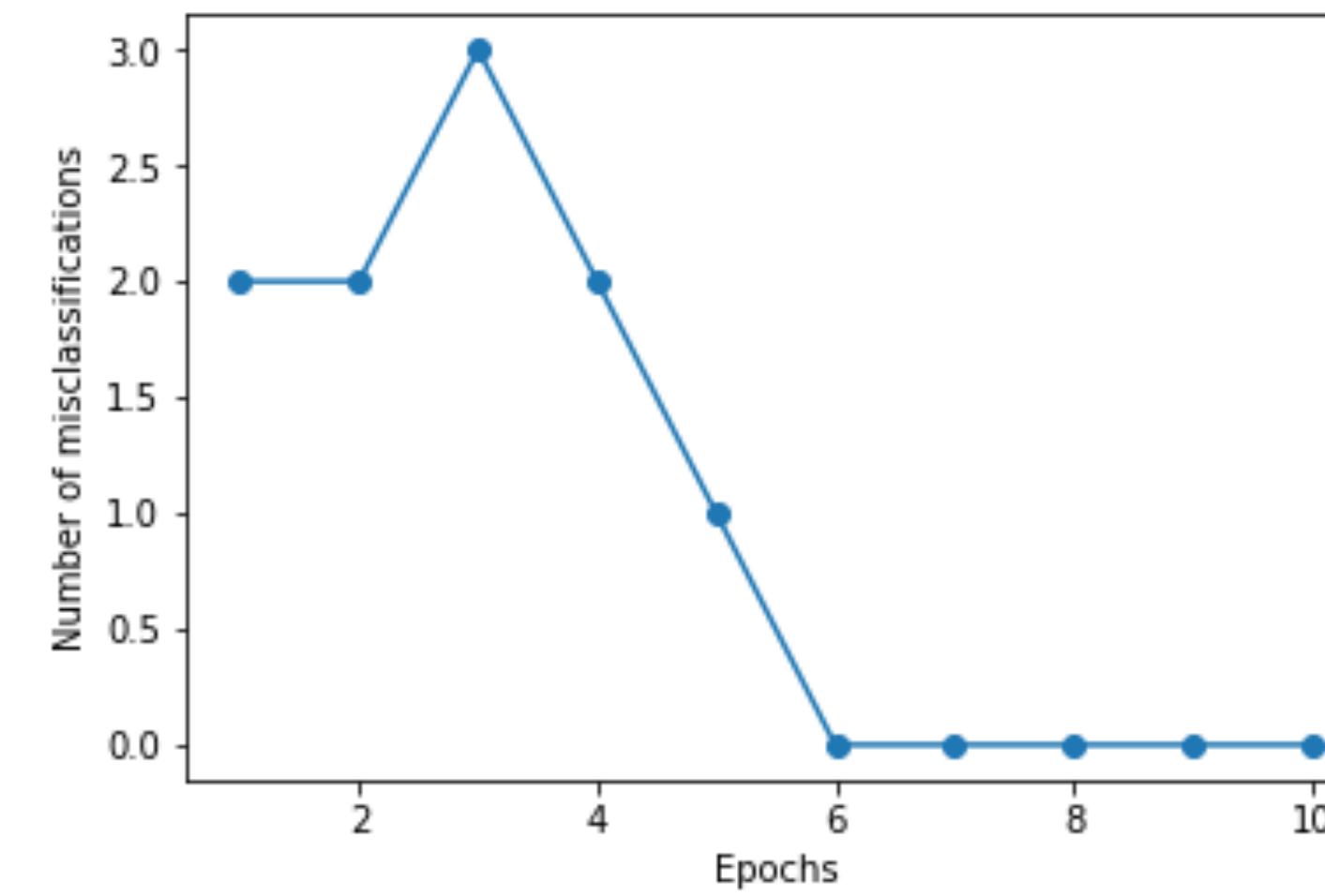
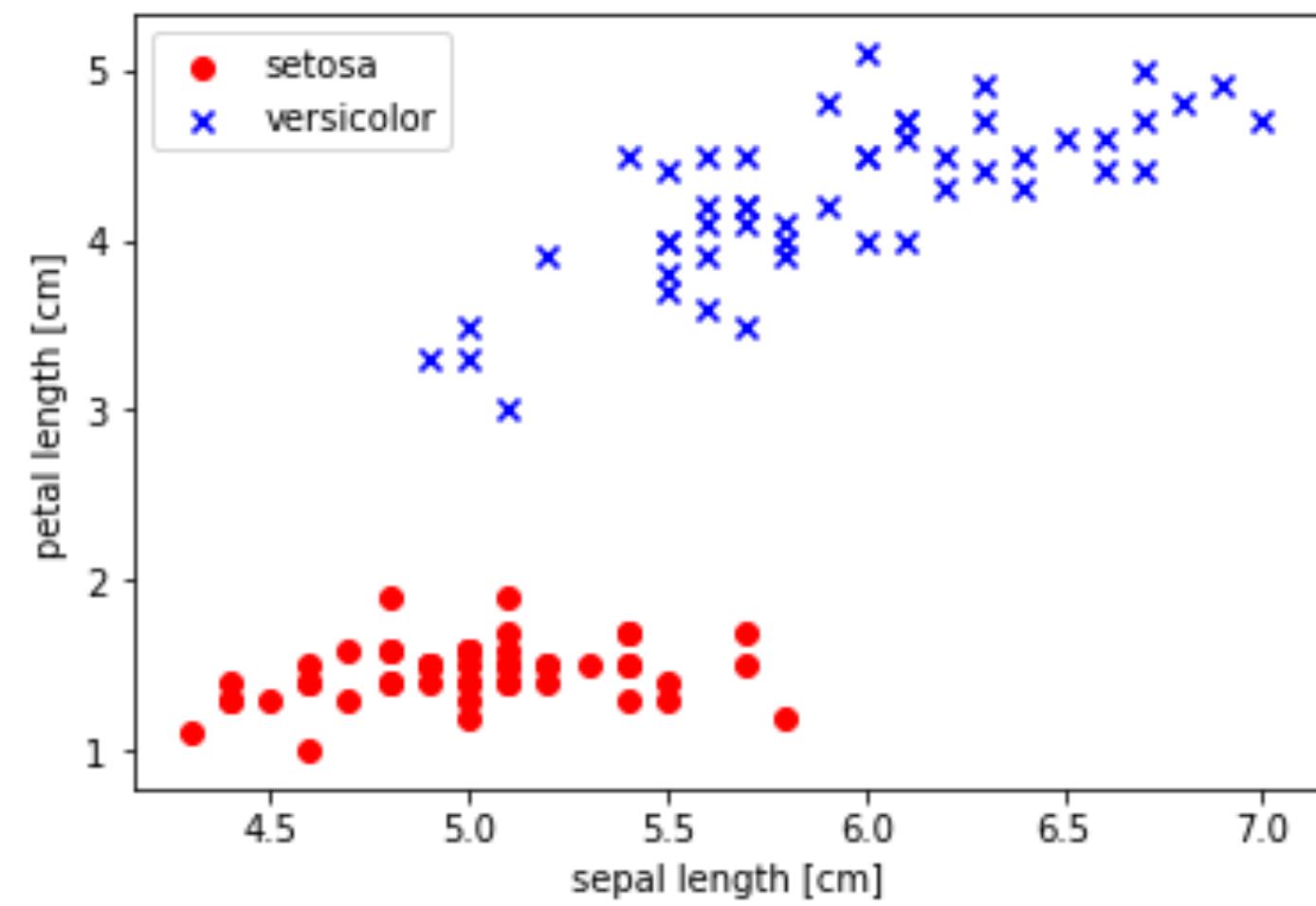


靠近星星的距離可分三群



1. Python 機器學習







1. Python 機器學習

```
8 class Perceptron(object):
9
10    def __init__(self, eta=0.01, n_iter=10):
11        self.eta = eta
12        self.n_iter = n_iter
13
14    def fit(self, X, y):
15        self.w_ = np.zeros(1 + X.shape[1])
16        self.errors_ = []
17
18        for _ in range(self.n_iter):
19            errors = 0
20            for xi, target in zip(X, y):
21                update = self.eta * (target - self.predict(xi))
22                self.w_[1:] += update * xi
23                self.w_[0] += update
24                errors += int(update != 0.0)
25            self.errors_.append(errors)
26
27        return self
```



Python 機器學習

predict()為啟動函數

```
28     def net_input(self, X):  
29         return np.dot(X, self.w_[1:]) + self.w_[0]  
30  
31     def predict(self, X):  
32         return np.where(self.net_input(X) >= 0.0, 1, -1)
```



2. 認知演算法

```
1: w  $\leftarrow$  0, b  $\leftarrow$  0
2: repeat
3:   errors  $\leftarrow$  0
4:   /* cycle through all training examples */
5:   for i = 1, ..., n do
6:     compute  $f(\mathbf{x}_i) = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ 
7:     if  $f(\mathbf{x}_i) \neq y_i$  then
8:       w  $\leftarrow$  w +  $y_i \mathbf{x}_i$ 
9:       b  $\leftarrow$  b +  $y_i$ 
10:      errors  $\leftarrow$  errors + 1
11:    end if
12:  end for
13: until error = 0
14: output w, b
```





3.鳶尾花

| 花萼長 | 花萼寬 | 花瓣長 | 花瓣寬 | 屬種 |
|-----|-----|-----|-----|------------|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 7.0 | 3.2 | 4.7 | 1.4 | versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| 5.9 | 3.0 | 5.1 | 1.8 | virginica |
| 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| 5.9 | 3.0 | 5.1 | 1.8 | virginica |
| 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| 6.2 | 3.4 | 5.4 | 2.3 | virginica |



變色鳶尾,維吉尼亞鳶尾,山鳶尾

變色鳶尾



Versicolor

維吉尼亞鳶尾



Virginica

山鳶尾



Setosa

花瓣長屬性,花萼長屬性



鳶尾花

- 其數據集包含了150個樣本，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾。
- 四個特徵被用作樣本的定量分析，它們分別是花萼和花瓣的長度和寬度。基於這四個特徵的集合，一個線性判別分析以確定其屬種。



4. 加州大學鳶尾花資料

```
34 #####  
35 print(50 * '=')  
36 print('認知演算法讀取加州大學鳶尾花資料')  
37 print(50 * '-')  
38  
39 df = pd.read_csv('https://archive.ics.uci.edu/ml/'  
40                 'machine-learning-databases/iris/iris.data',  
41                 header=None)  
42 print(df.tail())
```



5. 決策區域函數

classifier為我們的分類演算法

```
49 def plot_decision_regions(X, y, classifier, resolution=0.02):
50
51
52     markers = ('s', 'x', 'o', '^', 'v')
53     colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
54     cmap = ListedColormap(colors[:len(np.unique(y))])
55
56
57     x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
58     x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
59     xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
60                           np.arange(x2_min, x2_max, resolution))
61     Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
62     Z = Z.reshape(xx1.shape)
63     plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
```

決策區域函數

```
64 plt.xlim(xx1.min(), xx1.max())
65 plt.ylim(xx2.min(), xx2.max())
66
67 for idx, cl in enumerate(np.unique(y)):
68     plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
69                  alpha=0.8, c=cmap(idx),
70                  marker=markers[idx], label=cl)
```

○山鳶尾紅色setosa,
X變色鳶尾藍色versicolor

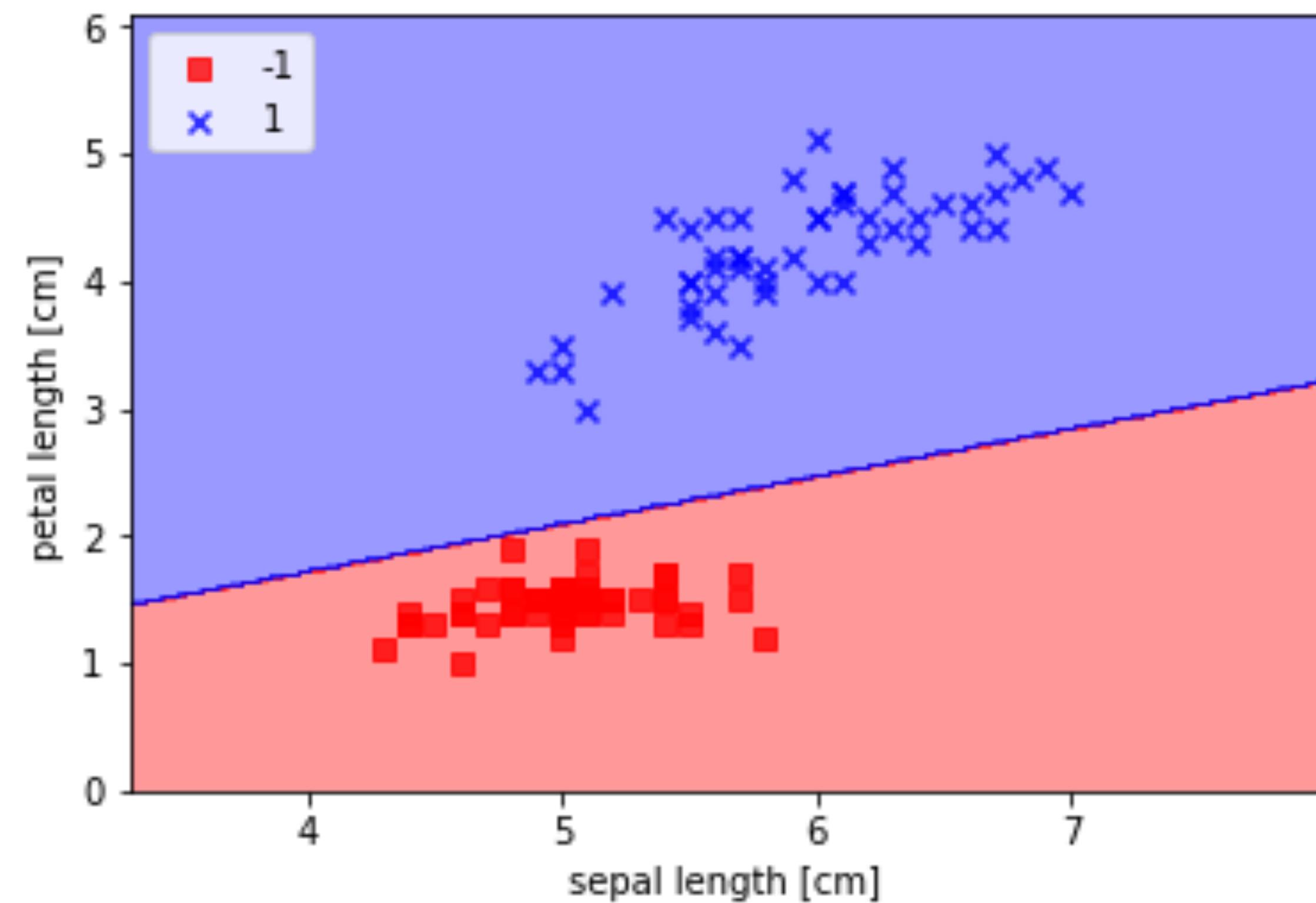
```
79 y = df.iloc[0:100, 4].values
80 y = np.where(y == 'Iris-setosa', -1, 1)
81
82 X = df.iloc[0:100, [0, 2]].values
83
84 plt.scatter(X[:50, 0], X[:50, 1],
85             color='red', marker='o', label='setosa')
86 plt.scatter(X[50:100, 0], X[50:100, 1],
87             color='blue', marker='x', label='versicolor')
88
89 plt.xlabel('sepal length [cm]')
90 plt.ylabel('petal length [cm]')
91 plt.legend(loc='upper left')
92
93 plt.show()
```

6. 建立模型Perceptron().fit()

plot_decision_regions()為繪製區域

```
99
100 ppn = Perceptron(eta=0.1, n_iter=10)
101 ppn.fit(X, y)
102 plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
103 plt.xlabel('Epochs')
104 plt.ylabel('Number of misclassifications')
105 plt.show()
106 plt.xlabel('sepal length [cm]')
107 plt.ylabel('petal length [cm]')
108 plt.legend(loc='upper left')
109 plot_decision_regions(X, y, classifier=ppn)
```

○山鳶尾紅色,
X變色鳶尾藍色



- Thanks.



Python

類神經網路深度學習

吳佳謙 老師





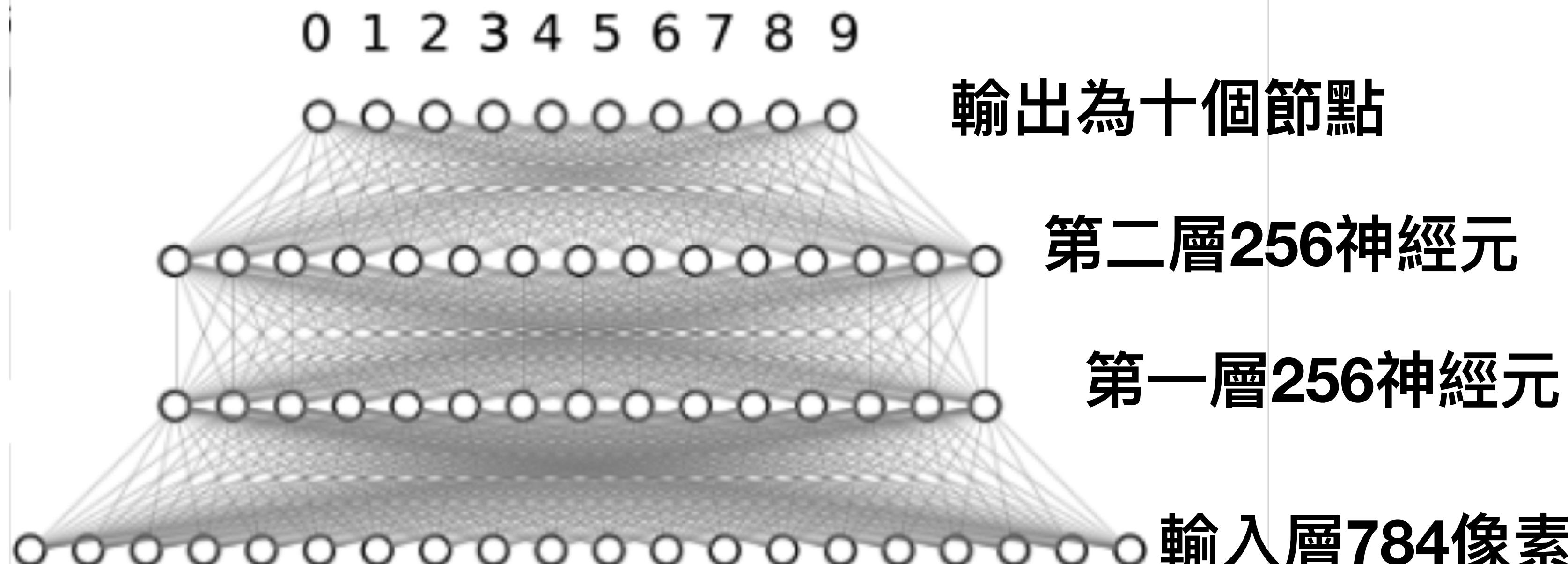
Python

類神經網路深度學習

1. 安裝Tensorflow
2. 安裝Keras
3. 類神經網路圖形辨識MNIST
4. 類神經深度學習
5. 繪製實際和預測結果的手寫辨識
6. 顯示預測圖形



多層的類神經深度學習



1. 安裝Tensorflow

- pip install tensorflow

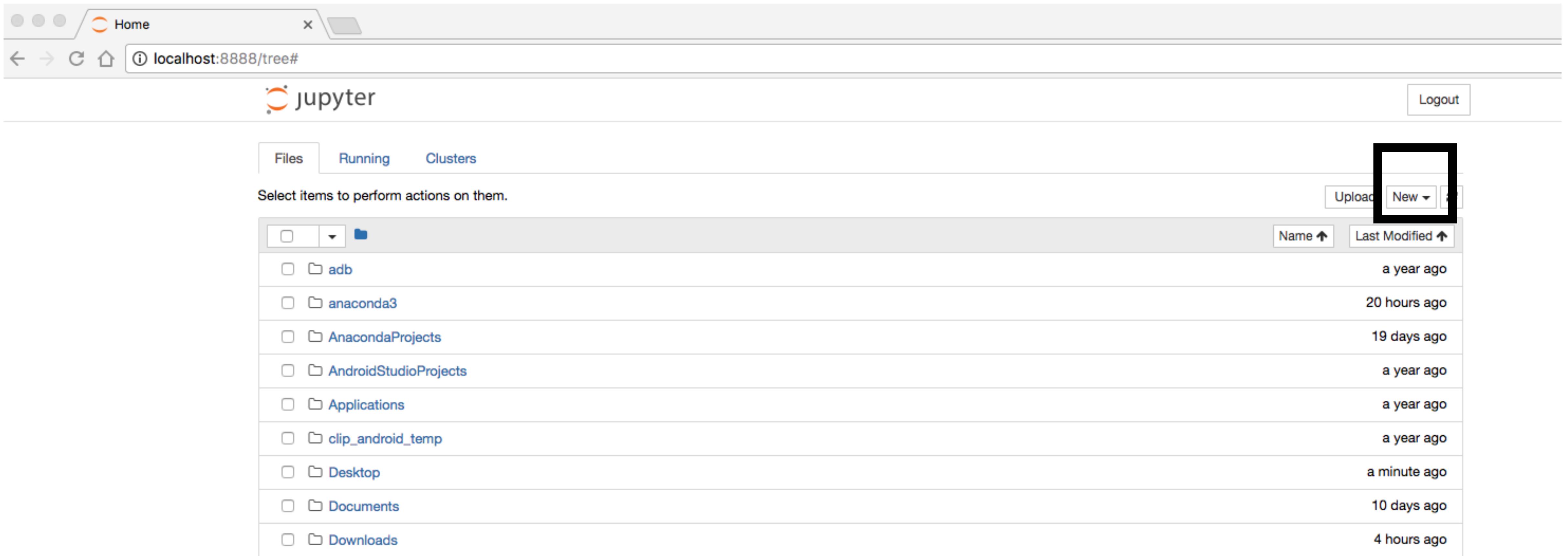
2. 安裝Keras

- pip install keras

啟動jupyter

```
$ jupyter notebook
```

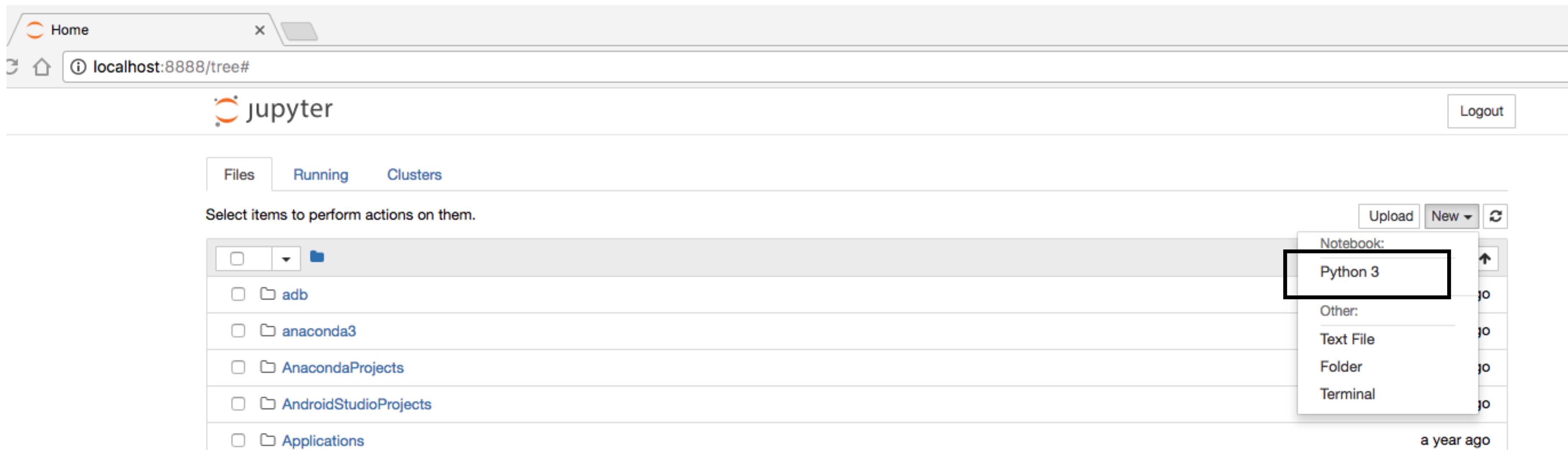
選取New



The screenshot shows a Jupyter Notebook interface running on a local host. The top bar includes navigation icons, a 'Home' button, and a URL bar showing 'localhost:8888/tree#'. Below the header is the Jupyter logo and a 'Logout' button. A navigation menu at the top has tabs for 'Files', 'Running', and 'Clusters', with 'Files' currently selected. A message 'Select items to perform actions on them.' is displayed above the file list. On the right side of the interface, there are buttons for 'Upload' and 'New' (which is highlighted with a black box). The main area displays a file tree with the following entries:

| File/Folder | Last Modified |
|-----------------------|---------------|
| adb | a year ago |
| anaconda3 | 20 hours ago |
| AnacondaProjects | 19 days ago |
| AndroidStudioProjects | a year ago |
| Applications | a year ago |
| clip_android_temp | a year ago |
| Desktop | a minute ago |
| Documents | 10 days ago |
| Downloads | 4 hours ago |

選取Python 3



觀看Tensorflow的版本

- import tensorflow as tf
- tf.__version__

執行程式

The screenshot shows a Jupyter Notebook interface with the following details:

- Toolbar:** Includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Python 3 tab.
- Header:** Shows two tabs: "Home" and "Untitled2".
- Address Bar:** Displays the URL: "localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3".
- Code Cells:** Three cells are visible:
 - In [2]: `import tensorflow as tf`
 - In [3]: `tf.__version__`
 - In []: (empty)
- Output Cell:** The output for In [3] is "Out[3]: '1.3.0'".
- Cell Selection:** The cell containing `tf.__version__` is highlighted with a blue border.
- Execution Counter:** A black square highlights the execution counter icon in the toolbar.

輸入import tensorflow和執行

```
In [2]: import tensorflow as tf
```

```
In [3]: tf.__version__
```

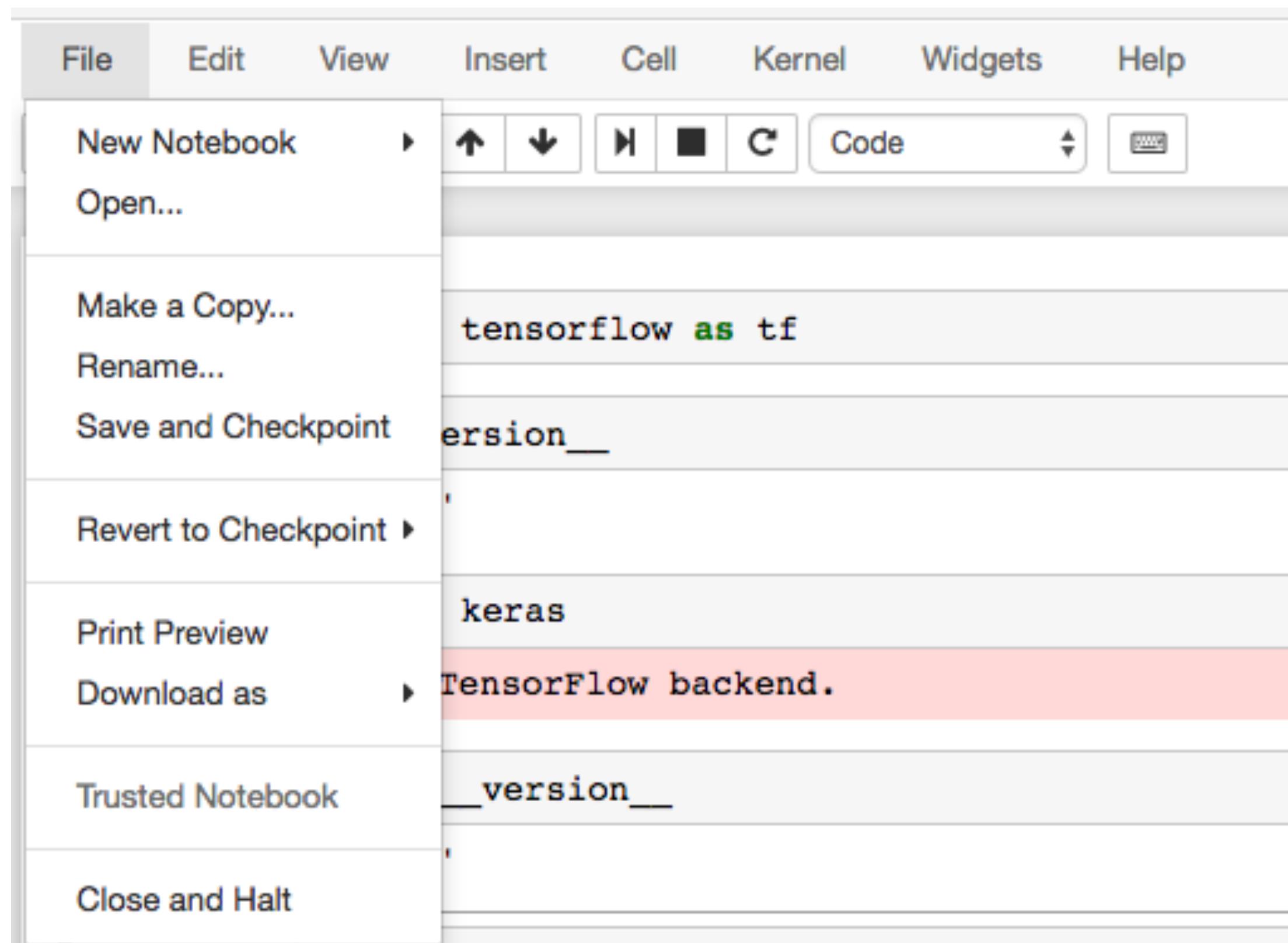
```
Out[3]: '1.3.0'
```

顯示keras版本,使用TensorFlow當後端

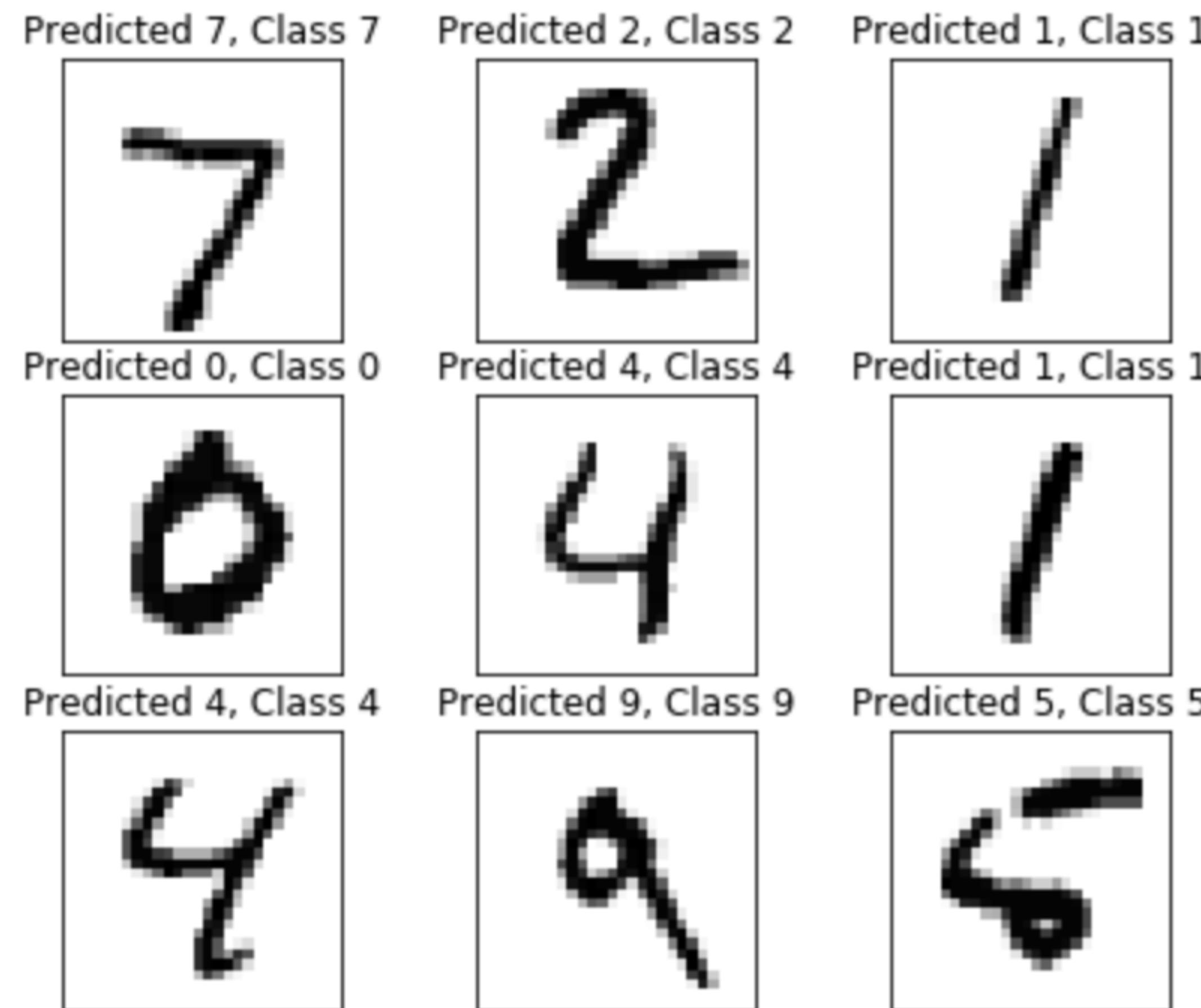
The screenshot shows a Jupyter Notebook interface with the following code execution history:

- In [2]: `import tensorflow as tf`
- In [3]: `tf.__version__`
Out[3]: '1.3.0'
- In [4]: `import keras`
Using TensorFlow backend.
- In [5]: `keras.__version__`
Out[5]: '2.0.8'

Save儲存檔案



3. 類神經網路圖形辨識MNIST



輸入Keras模組

- 輸入keras的資料datasets,mnist手寫數字
- 輸入keras的模型models,sequential循序模型
- 輸入keras類神經核心的Dense,Dropout,和啟動Activation

```
In [42]: import numpy as np
import matplotlib.pyplot as plt

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

載入MNIST資料

- <http://yann.lecun.com/exdb/mnist/>
- 為辨識手寫資料的網址
- mnist.load_data()為載入手寫資料

```
In [189]: nb_classes = 10
          (X_train, y_train), (X_test, y_test) = mnist.load_data()
          print("X_train original shape", X_train.shape)
          print("y_train original shape", y_train.shape)

          X_train original shape (60000, 28, 28)
          y_train original shape (60000,)
```

繪製圖形使用imshow函數

- matplotlib.pyplot.gc()**是得到目前的圖形
- matplotlib.pyplot.imshow**是顯示圖片

```
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(X_train[0], cmap='binary', interpolation='none')
plt.title("Class {}".format(y_train[0]))
plt.xticks([])
plt.yticks([])
plt.subplot(2,1,2)
plt.hist(X_train[0].reshape(784))
plt.title("Pixel Value Distribution")
fig
```

圖片的顯示

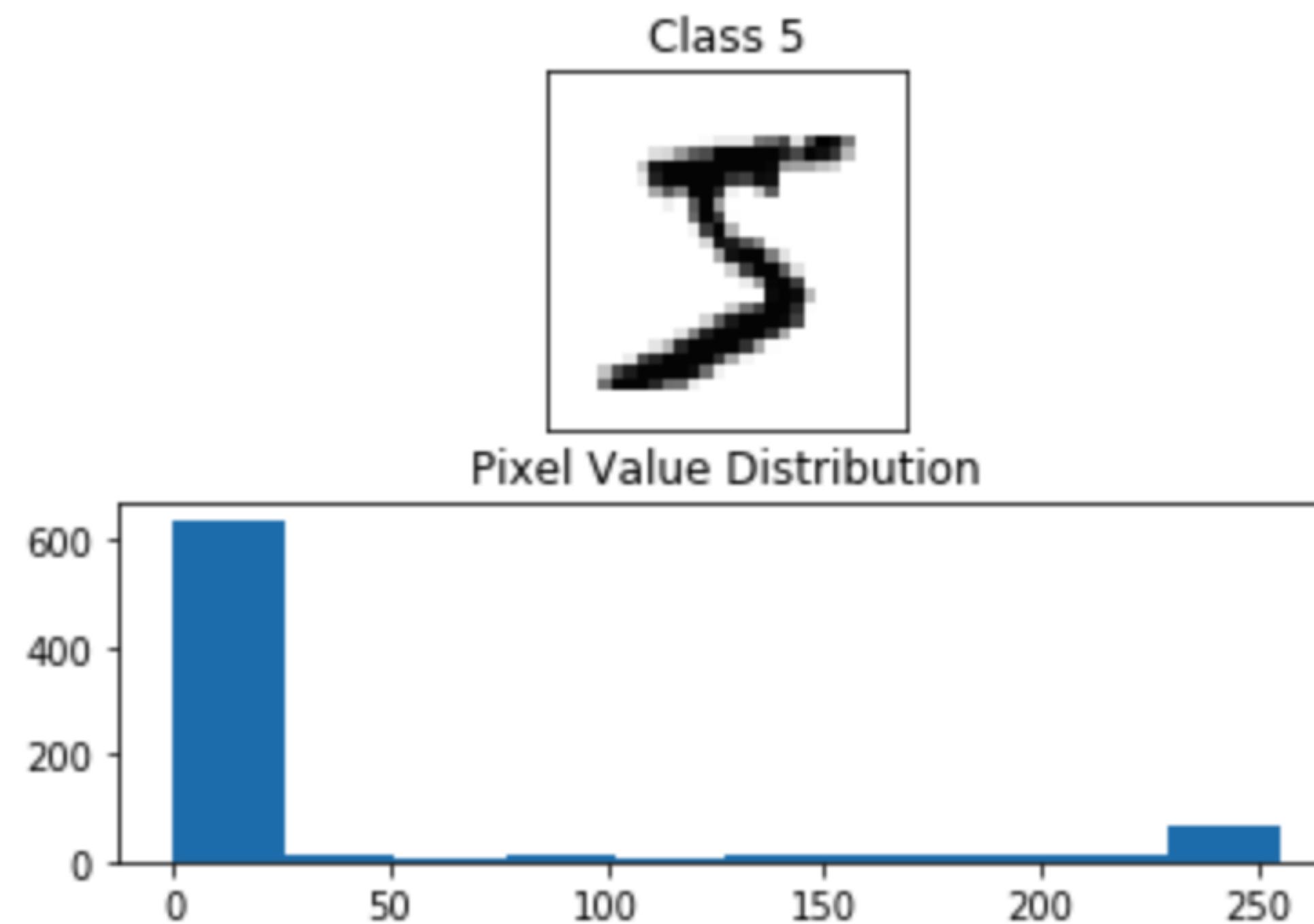
數位影像，是二維圖像用有限數字數值像素的表示。

二值圖像：圖像中每個像素的亮度值(Intensity)僅可以取自0或1的圖像，因此也稱為1-bit圖像。

灰度圖像：也稱為灰階圖像：圖像中每個像素可以由0(黑)到255(白)的亮度值(Intensity)表示。0-255之間表示不同的灰度級。

灰階的分佈

- 顯示手寫圖片第一張,白色0和黑色255,顯示灰階的分佈



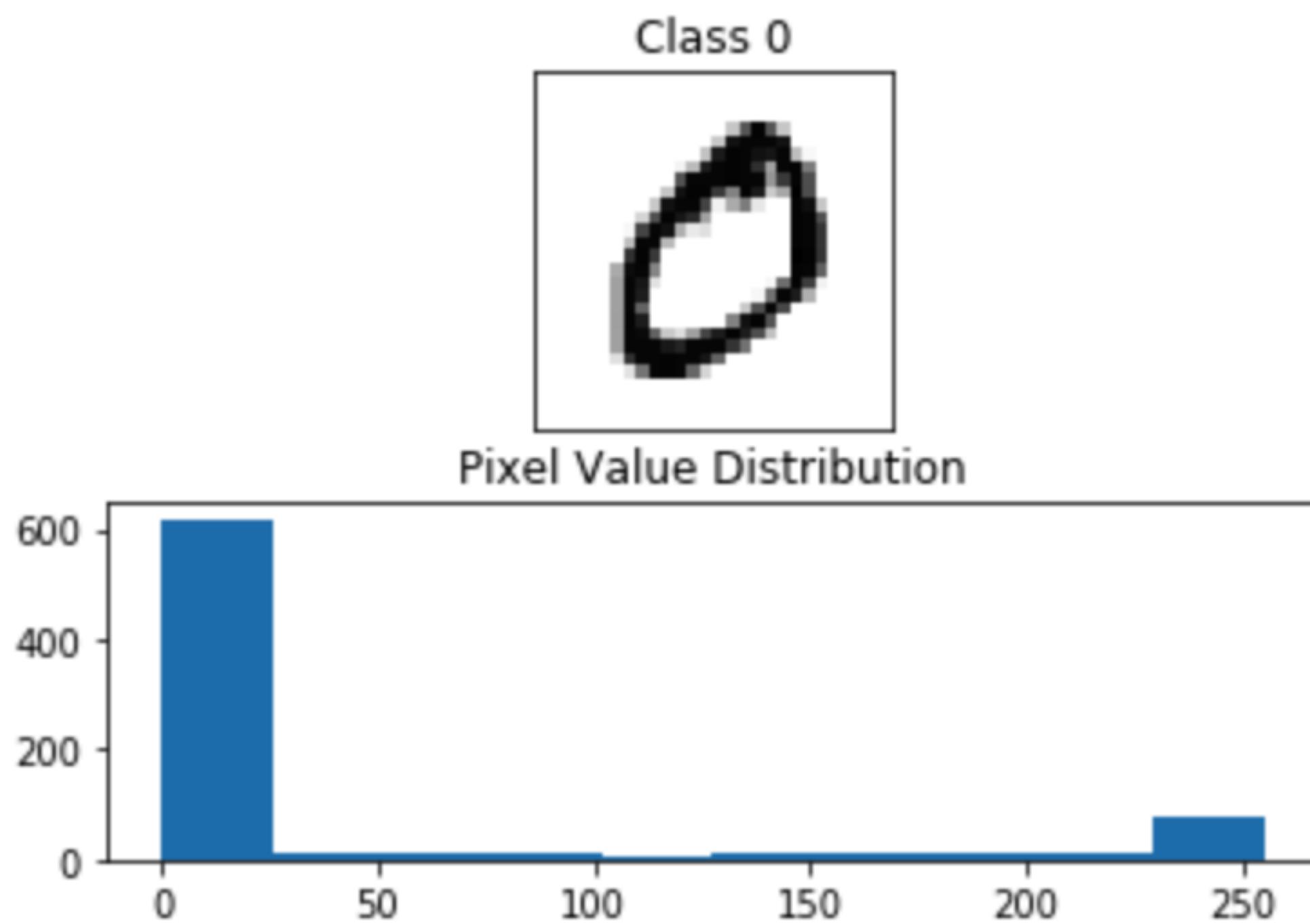
觀看訓練的手寫資料

- X_train[1]第二張圖,cmap為顏色,binary為灰階

```
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(X_train[1], cmap='binary', interpolation='none')
plt.title("Class {}".format(y_train[1]))
plt.xticks([])
plt.yticks([])
plt.subplot(2,1,2)
plt.hist(X_train[1].reshape(784))
plt.title("Pixel Value Distribution")
fig
```

觀看訓練的手寫資料

- $X_{train}[1]$ 第二張圖為數字0



顯示維度

```
# 顯示維度  
print("x_train shape", x_train.shape)  
print("y_train shape", y_train.shape)  
print("x_test shape", x_test.shape)  
print("y_test shape", y_test.shape)
```

```
x_train shape (60000, 28, 28)  
y_train shape (60000, )  
x_test shape (10000, 28, 28)  
y_test shape (10000, )
```

訓練,測試資料的正規化

```
# 從 28x28 pixels建立輸入向量
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

# 正規化資料0到1之間
x_train /= 255
x_test /= 255

# 訓練用最後輸入的維度
print("Train matrix shape", x_train.shape)
print("Test matrix shape", x_test.shape)
```

訓練,測試資料的正規化

- 訓練60000筆資料,784維度,28*28
- 測試10000筆資料,並將資料數值除以255,成為0到1的數值

Train matrix shape (60000 , 784)

Test matrix shape (10000 , 784)

utils.to_categorical將標籤轉換成 0到9的陣列

- 建立10,nb_classes個元素的向量,在第y_train[0]元素值為1,其它向量元素為0.y_train[0]為0到9的數字.

```
n_classes = 10
print("Shape before one-hot encoding: ", y_train.shape)
Y_train = np_utils.to_categorical(y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)
print("Shape after one-hot encoding: ", Y_train.shape)
```

```
Shape before one-hot encoding: (60000,)
Shape after one-hot encoding: (60000, 10)
```

```
Y_train[0]
```

```
array([ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.])
```



4. 類神經深度學習

- Keras的核心為模型，最主要也是最常使用的是 Sequential這個模型，Sequential可以讓我們按照順序將神經網路串起。深度學習為隱藏層有兩層或兩層以上。

```
model = Sequential()
model.add(Dense(256, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.3))

model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.3))

model.add(Dense(10))
model.add(Activation('softmax'))
```

類神經深度學習

- add()我們可以一層一層的將神經網路疊起。在每一層之中我們只需要設定每層的大小(units)與啟動函數(activation function)。
- 第一層輸入向量大小、最後一層為units要等於輸出的向量大小。
- 最後一層的啟動函數(activation function)為softmax。
- softmax()為歸一化指數函數,將向量的值歸一化為0到1之間。

keras編譯模型model.compile()

- 使用 Adam 做為優化器，成本函數使用 categorical_crossentropy。
- `model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])`

訓練和建立模型model.fit()

- 訓練為80%,用來驗證的設為20%,反覆8次執行,每次128筆

```
history = model.fit(X_train, Y_train,
                     batch_size=128, epochs=8,
                     verbose=2,
                     validation_data=(X_test, Y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/8
- 5s - loss: 0.3505 - acc: 0.8939 - val_loss: 0.1270 - val_acc: 0.9607
Epoch 2/8
- 5s - loss: 0.1487 - acc: 0.9556 - val_loss: 0.0938 - val_acc: 0.9699
Epoch 3/8
- 5s - loss: 0.1084 - acc: 0.9671 - val_loss: 0.0844 - val_acc: 0.9732
Epoch 4/8
- 5s - loss: 0.0910 - acc: 0.9716 - val_loss: 0.0762 - val_acc: 0.9754
Epoch 5/8
- 6s - loss: 0.0785 - acc: 0.9754 - val_loss: 0.0716 - val_acc: 0.9774
Epoch 6/8
- 5s - loss: 0.0678 - acc: 0.9784 - val_loss: 0.0717 - val_acc: 0.9780
Epoch 7/8
- 5s - loss: 0.0630 - acc: 0.9803 - val_loss: 0.0668 - val_acc: 0.9806
Epoch 8/8
- 5s - loss: 0.0565 - acc: 0.9820 - val_loss: 0.0642 - val_acc: 0.9803
```

model.fit()傳回History物件

history.history.keys()傳回History物件的字典

loss是成本,acc是準確度

history.history.keys()

```
dict_keys(['val_loss', 'val_acc',
'loss', 'acc'])
```

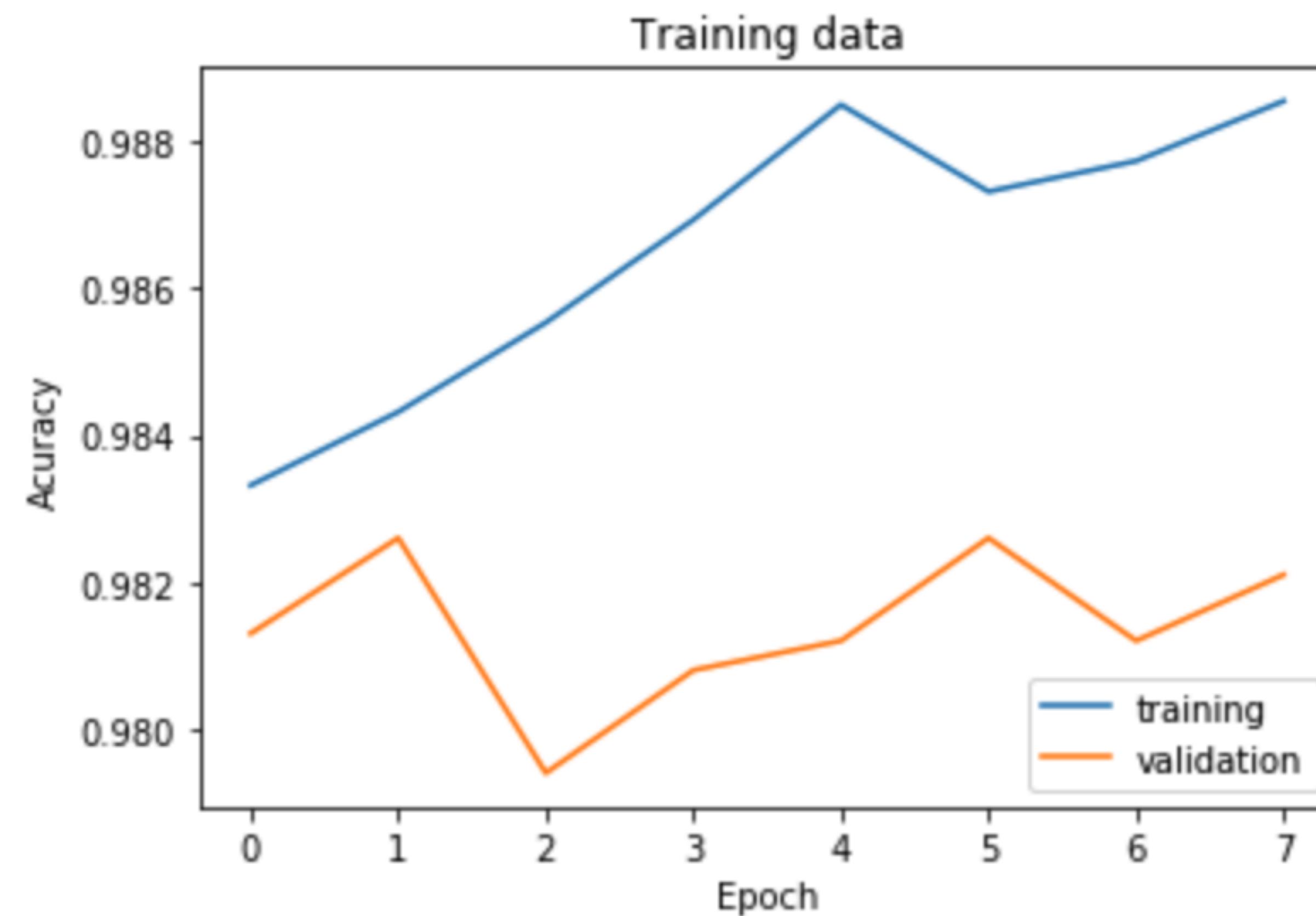
顯示模型資料準確性

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training data')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['training', 'validation'], loc='lower right')
plt.show()
```

訓練和驗證的準確性

- plt.xlabel('Epoch')
- plt.ylabel('Accuracy')
- plt.title('Training data')
- plt.plot(history.history['acc'])
- plt.plot(history.history['val_acc'])
- plt.legend(['training','validation'],loc='lower right')
- plt.show()

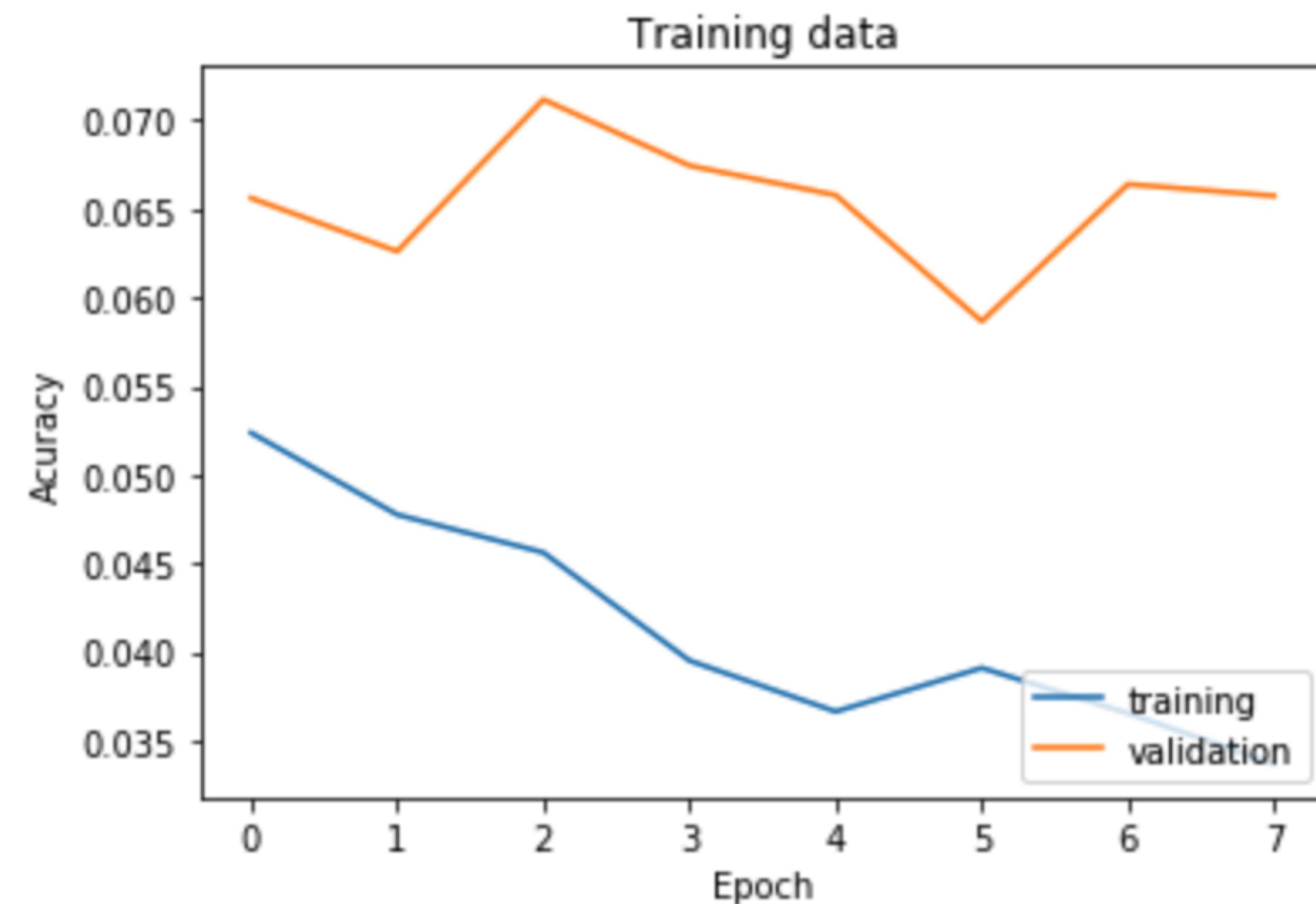
訓練和驗證的準確性



訓練和驗證的準確性

- plt.xlabel('Epoch')
- plt.ylabel('Accuracy')
- plt.title('Training data')
- plt.plot(history.history['loss'])
- plt.plot(history.history['val_loss'])
- plt.legend(['training','validation'],loc='lower right')
- plt.show()

這是訓練和驗證的成本函數



驗證模型準確性使用evaluate()

```
loss_and_metrics = model.evaluate(X_test,  
Y_test, verbose=2)
```

```
print("Test Loss", loss_and_metrics[0])  
print("Test Accuracy", loss_and_metrics[1])
```

Test Loss 0.0657472234725

Test Accuracy 0.9821

model.predict_classes得到驗證 的結果

- correct_indices為預測正確的索引,incorrect_indices為預測錯誤的索引
- predicted_classes = model.predict_classes(X_test)
- correct_indices = np.nonzero(predicted_classes == y_test)[0]
- incorrect_indices = np.nonzero(predicted_classes != y_test)[0]

model.predict_classes得到驗證
的結果,0-9的手寫數值

```
In [166]: predicted_classes  
Out[166]: array([7, 2, 1, ..., 4, 5, 6])
```

matplotlib.pyplot.figure

將回傳圖形實體

rcParams定義預設數值

```
# 設定圖形長12寬7  
plt.rcParams['figure.figsize'] = (7,12)
```

```
figure_evaluation = plt.figure()
```

5.繪製實際和預測結果的手寫辨識

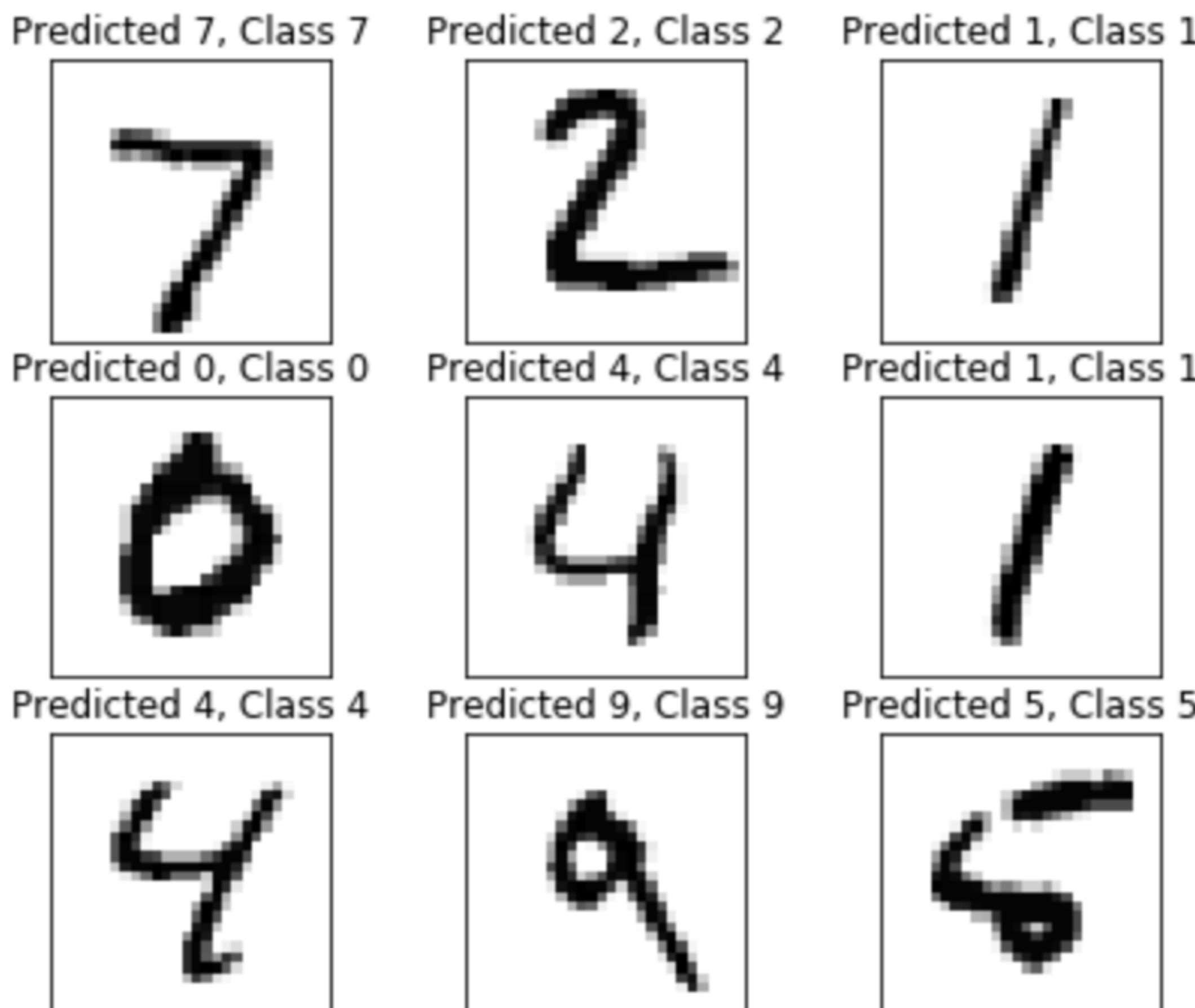
放置9個正確的預測圖形

```
# 放置9個正確的預測圖形
for i, correct in enumerate(correct_indices[:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(X_test[correct].reshape(28,28), cmap='binary')
    plt.title("Predicted {}, Class {}".format(
        predicted_classes[correct], y_test[correct]))
    plt.xticks([])
    plt.yticks([])

figure_evaluation
```



繪製實際和預測結果的手寫辨識



28*28的圖形,正規化成0到1的數值

```
In [11]: X_test[0]
```

| | | | | | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------------|----|---|
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0.06666667, | 0.25882354, | 0.05490196, | 0.26274511, | | | | |
| 0.26274511, | 0.26274511, | 0.23137255, | 0.08235294, | 0.9254902, | | | | | |
| 0.99607843, | 0.41568628, | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0.32549021, | 0.99215686, | 0.81960785, | 0.07058824, | | | | |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0.08627451, | 0.9137255, | | |
| 1. | , | 0.32549021, | 0. | , | 0. | , | 0. | , | |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0. | , | 0. | , | 0. | , | 0. | , |
| 0. | , | 0.50588238, | 0.99607843, | 0.93333334, | 0.17254902, | | | | |

y_test為0到9的數值

```
In [173]: X_test,y_test
Out[173]: (array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]], dtype=float32),
 array([7, 2, 1, ..., 4, 5, 6], dtype=uint8))
```

這是辨識錯誤的圖形編號

```
In [176]: incorrect_indices  
  
Out[176]: array([ 151,  247,  321,  445,  447,  448,  449,  450,  456,  582,  610,  
   619,  646,  659,  684,  691,  720,  883,  890,  938,  944, 1014,  
  1039, 1112, 1156, 1166, 1178, 1182, 1226, 1232, 1242, 1247, 1260,  
 1315, 1319, 1356, 1393, 1438, 1500, 1522, 1530, 1531, 1549, 1554,  
 1600, 1609, 1611, 1621, 1626, 1681, 1751, 1790, 1828, 1901, 1982,  
 1984, 2004, 2024, 2053, 2070, 2098, 2109, 2118, 2129, 2130, 2135,  
 2185, 2189, 2293, 2299, 2326, 2369, 2387, 2406, 2414, 2447, 2455,  
 2488, 2560, 2607, 2618, 2648, 2654, 2713, 2877, 2921, 2927, 2939,  
 2952, 2953, 2995, 3030, 3060, 3062, 3108, 3117, 3251, 3377, 3405,  
 3451, 3475, 3503, 3520, 3533, 3558, 3559, 3597, 3681, 3776, 3780,  
 3796, 3808, 3811, 3893, 3941, 3943, 3976, 3985, 3995, 4027, 4065,  
 4078, 4156, 4176, 4199, 4224, 4248, 4289, 4294, 4317, 4363, 4382,  
 4425, 4497, 4528, 4536, 4567, 4751, 4807, 4823, 4860, 4880, 4966,  
 5068, 5138, 5331, 5457, 5573, 5600, 5634, 5642, 5676, 5734, 5936,  
 5945, 5955, 5973, 6009, 6011, 6023, 6045, 6046, 6071, 6093, 6166,  
 6173, 6426, 6555, 6574, 6576, 6597, 6641, 6651, 6735, 6744, 6783,  
 6817, 7049, 7216, 7459, 8059, 8091, 8094, 8246, 8273, 8277, 8290,  
 8325, 8413, 8504, 8522, 8527, 9009, 9024, 9211, 9253, 9280, 9587,  
 9634, 9664, 9669, 9692, 9700, 9729, 9735, 9745, 9749, 9755, 9768,  
 9770, 9779, 9792, 9832, 9839, 9867, 9904, 9922, 9944, 9982])
```

6.顯示預測圖形

```
figure_evaluation = plt.figure()  
# 放置9個錯誤的預測圖形  
for i, incorrect in enumerate(incorrect_indices[:9]):  
    plt.subplot(3,3,i+1)  
    plt.imshow(X_test[incorrect].reshape(28,28), cmap='binary')  
    plt.title("Predicted {}, Class {}".format(  
        predicted_classes[incorrect], y_test[incorrect]))  
    plt.xticks([])  
    plt.yticks([])  
  
figure_evaluation
```

顯示不正確的預測圖形

Predicted 6, Class 4



Predicted 9, Class 4



Predicted 0, Class 6



Predicted 3, Class 9



Predicted 7, Class 2



Predicted 3, Class 5



Predicted 9, Class 4



Predicted 0, Class 6



Predicted 9, Class 4





- Thanks.





Python 支援向量機 SVM

吳佳諺 老師

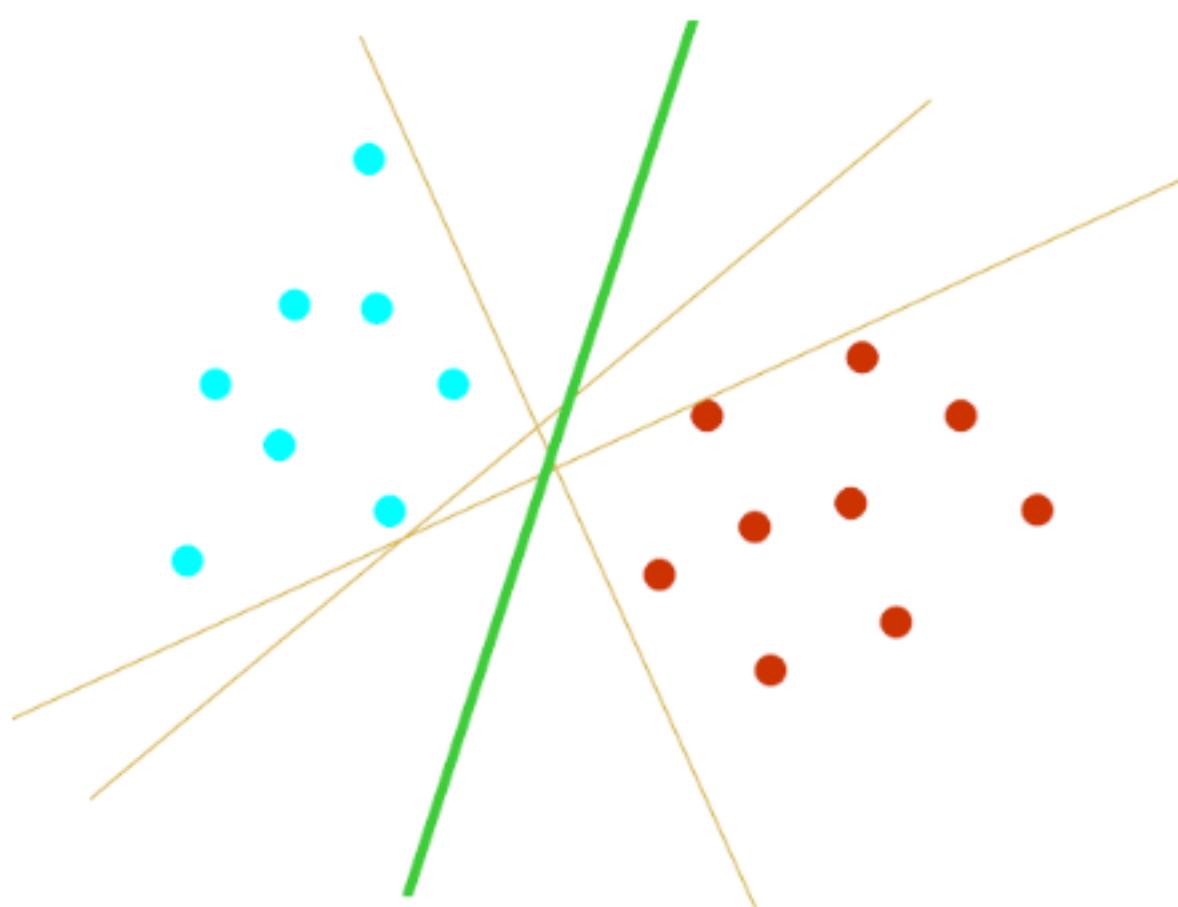




- 1.SVM最佳化分類平面
- 2.解最小化Lagrange乘法數問題
- 3.分錯時要補償的問題
- 4.將輸入資料對應到高維度特徵空間
- 5.Python實作鳶尾花分類
- 6.Support Vector Regression 迴歸



SVM最佳化分類平面



最佳化分類平面

$$\mathcal{D} = \{(x^1, y^1), \dots, (x^l, y^l)\}, \quad x \in \mathbb{R}^n, y \in \{-1, 1\},$$

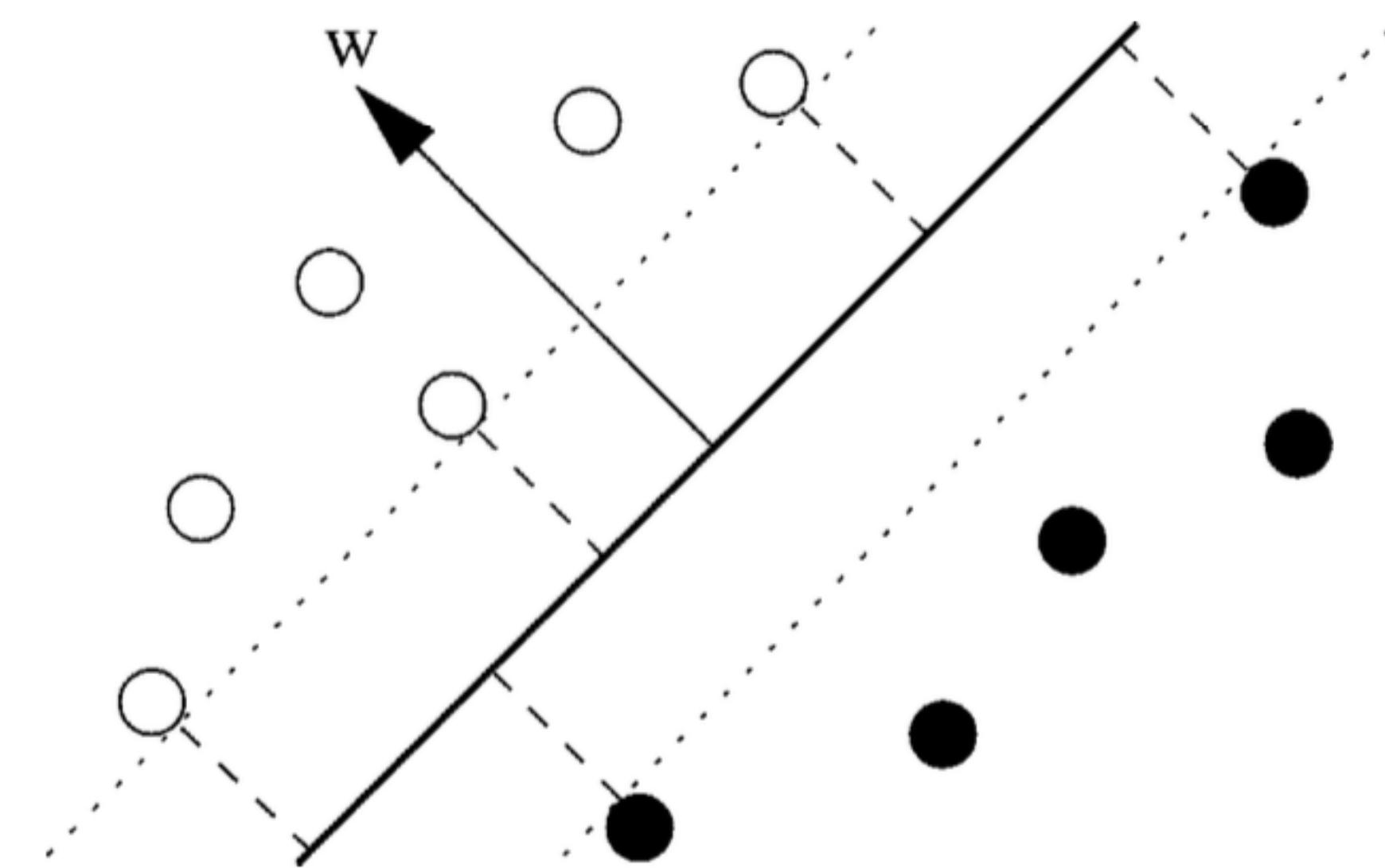
$$\langle w, x \rangle + b = 0.$$

最佳化分類平面

$$\min_i |\langle w, x^i \rangle + b| = 1.$$

$$y^i [\langle w, x^i \rangle + b] \geq 1, \quad i = 1, \dots, l.$$

最佳化分類平面



點到平面距離最大

$$d(w, b; x) = \frac{|\langle w, x^i \rangle + b|}{\|w\|}.$$

邊界距離最大化

$$\begin{aligned}\rho(w, b) &= \min_{x^i: y^i = -1} d(w, b; x^i) + \min_{x^i: y^i = 1} d(w, b; x^i) \\&= \min_{x^i: y^i = -1} \frac{|\langle w, x^i \rangle + b|}{\|w\|} + \min_{x^i: y^i = 1} \frac{|\langle w, x^i \rangle + b|}{\|w\|} \\&= \frac{1}{\|w\|} \left(\min_{x^i: y^i = -1} |\langle w, x^i \rangle + b| + \min_{x^i: y^i = 1} |\langle w, x^i \rangle + b| \right) \\&= \frac{2}{\|w\|}\end{aligned}$$



2. 解最小化Lagrange乘法數問題

$$\Phi(w) = \frac{1}{2} \|w\|^2.$$

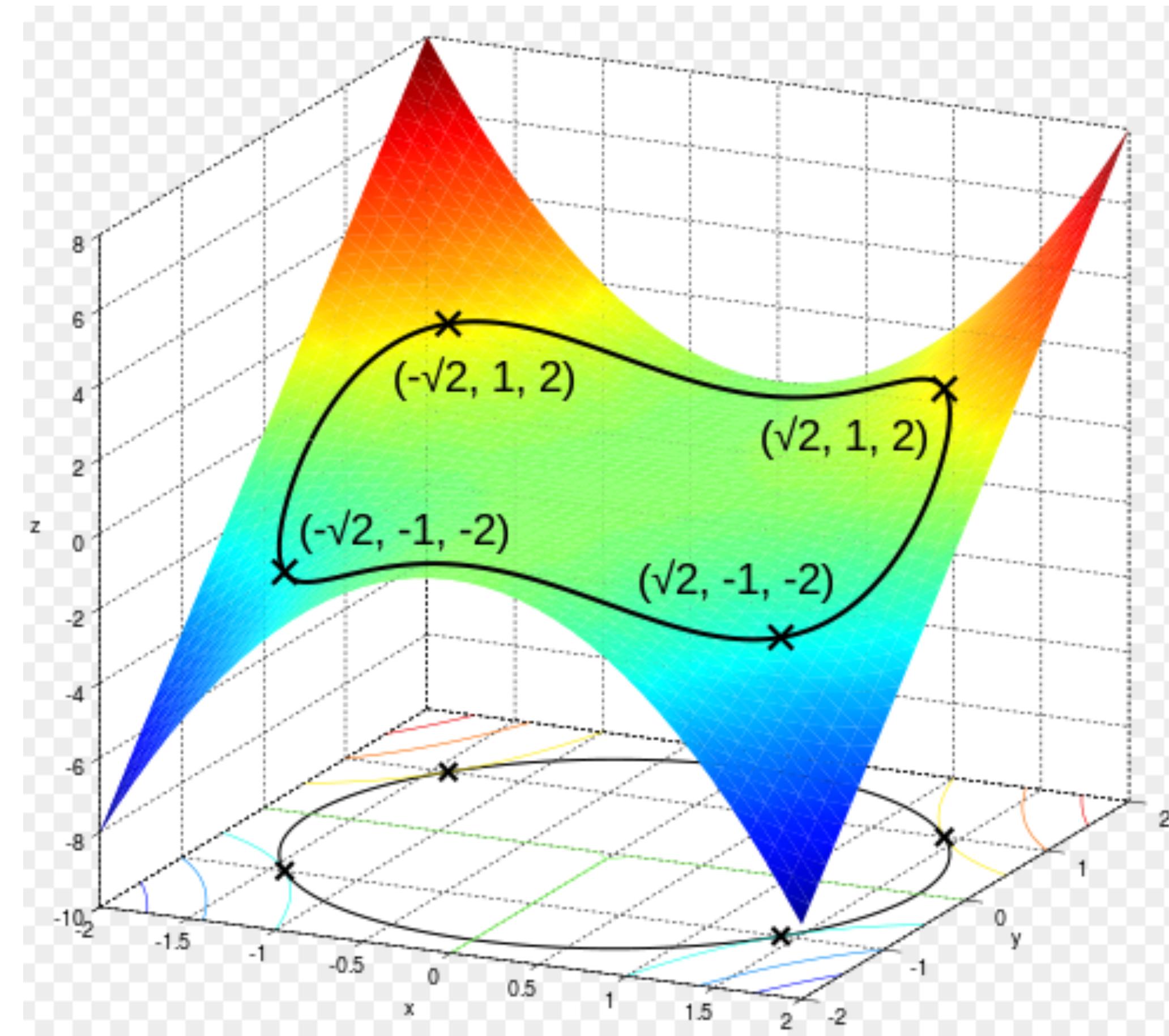
$$\Phi(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i (y^i [\langle w, x^i \rangle + b] - 1),$$

Lagrange求取最佳化問題

maximize $f(x, y)$
subject to $g(x, y) = 0.$

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda \cdot g(x, y),$$

Lagrange求取最佳化問題



$$\max_{\alpha} W(\alpha) = \max_{\alpha} \left(\min_{w,b} \Phi(w, b, \alpha) \right).$$

The minimum with respect to w and b of the Lagrangian, Φ , is given by,

$$\frac{\partial \Phi}{\partial b} = 0 \Rightarrow \sum_{i=1}^l \alpha_i y_i = 0$$

$$\frac{\partial \Phi}{\partial w} = \mathbf{0} \Rightarrow w = \sum_{i=1}^l \alpha_i y_i x_i.$$

the dual problem is,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{k=1}^l \alpha_k,$$

$$\alpha^* = \arg \min_{\alpha} \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^l \alpha_k,$$

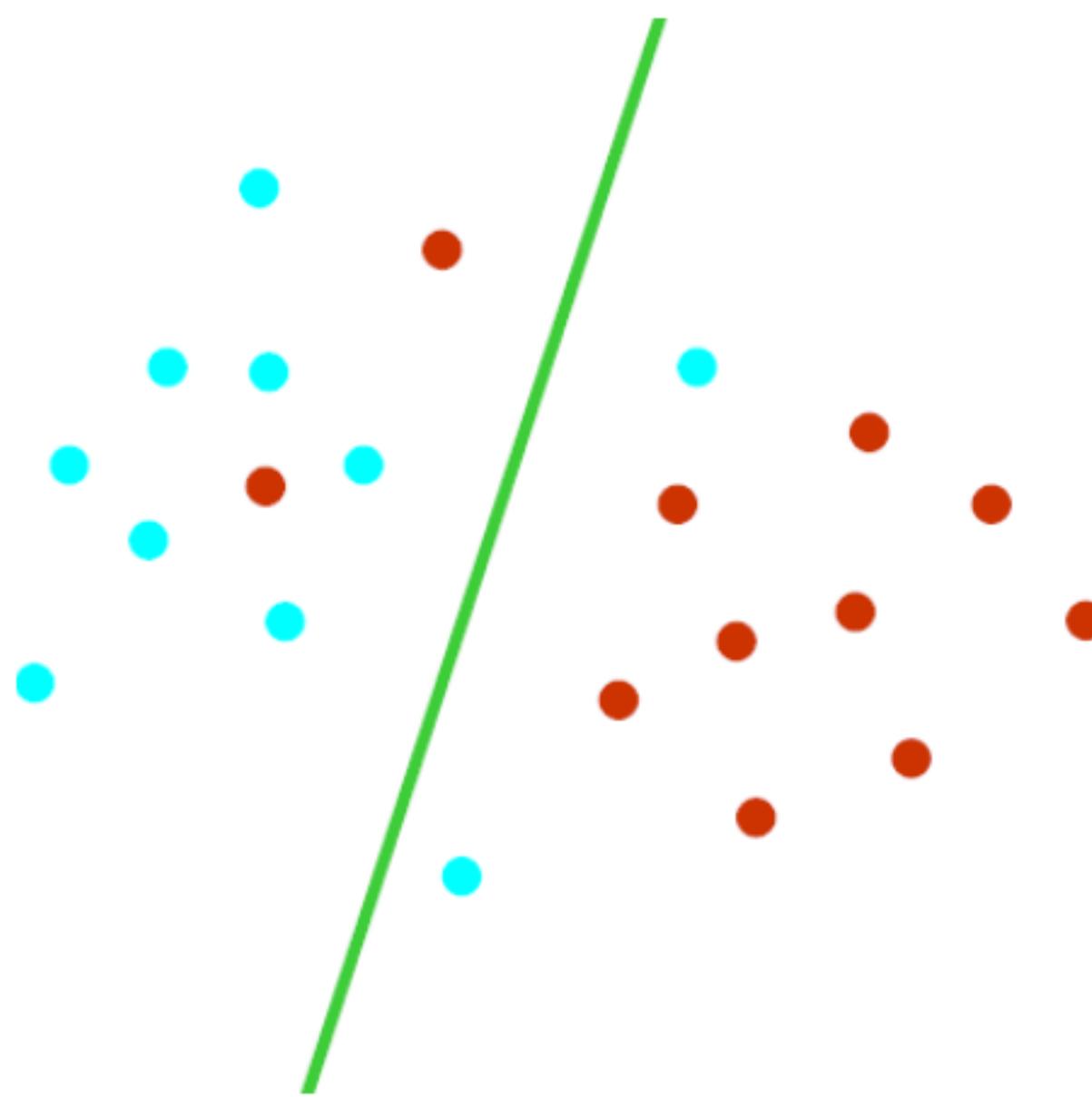
with constraints,

$$\alpha_i \geq 0 \quad i = 1, \dots, l$$

$$\sum_{j=1}^l \alpha_j y_j = 0.$$



3. 分錯時要補償的問題



求取最小解,C為補償參數

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i,$$

$$\alpha^* = \arg \min_{\alpha} \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^l \alpha_k,$$

$$0 \leq \alpha_i \leq C \quad i = 1, \dots, l$$

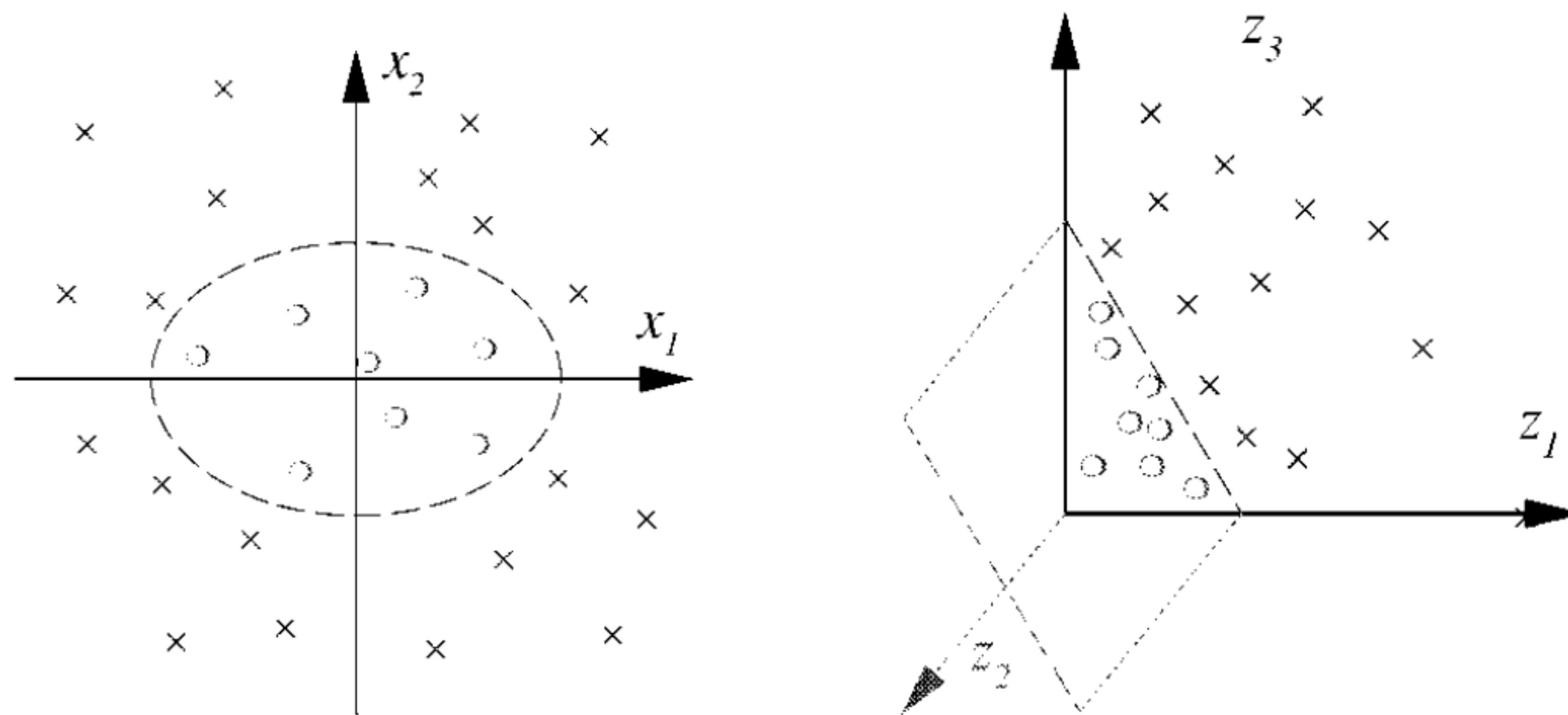
$$\sum_{j=1}^l \alpha_j y_j = 0.$$



4. 將輸入資料對應到高維度特徵空間



特徵空間轉換



核心 $K(x, x')$ 函數做向量空間轉換

$$\alpha^* = \arg \min_{\alpha} \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{k=1}^l \alpha_k,$$

核心函數

$$K(x, x') = \langle \phi(x), \phi(x') \rangle,$$

最佳化分類平面

$$f(x) = \operatorname{sgn}\left(\sum_{i \in SVs} \alpha_i y_i K(x_i, x) + b\right)$$

$$\langle w^*, x \rangle = \sum_{i=1}^l \alpha_i y_i K(x_i, x)$$

特徵空間與核心函數

- 高斯RBF核心

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

多項式核心

$$K(x, x') = \langle x, x' \rangle^d.$$

$$K(x, x') = (\langle x, x' \rangle + 1)^d.$$

指數式RBF函數

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right)$$

Multi-Layer Perceptron函數

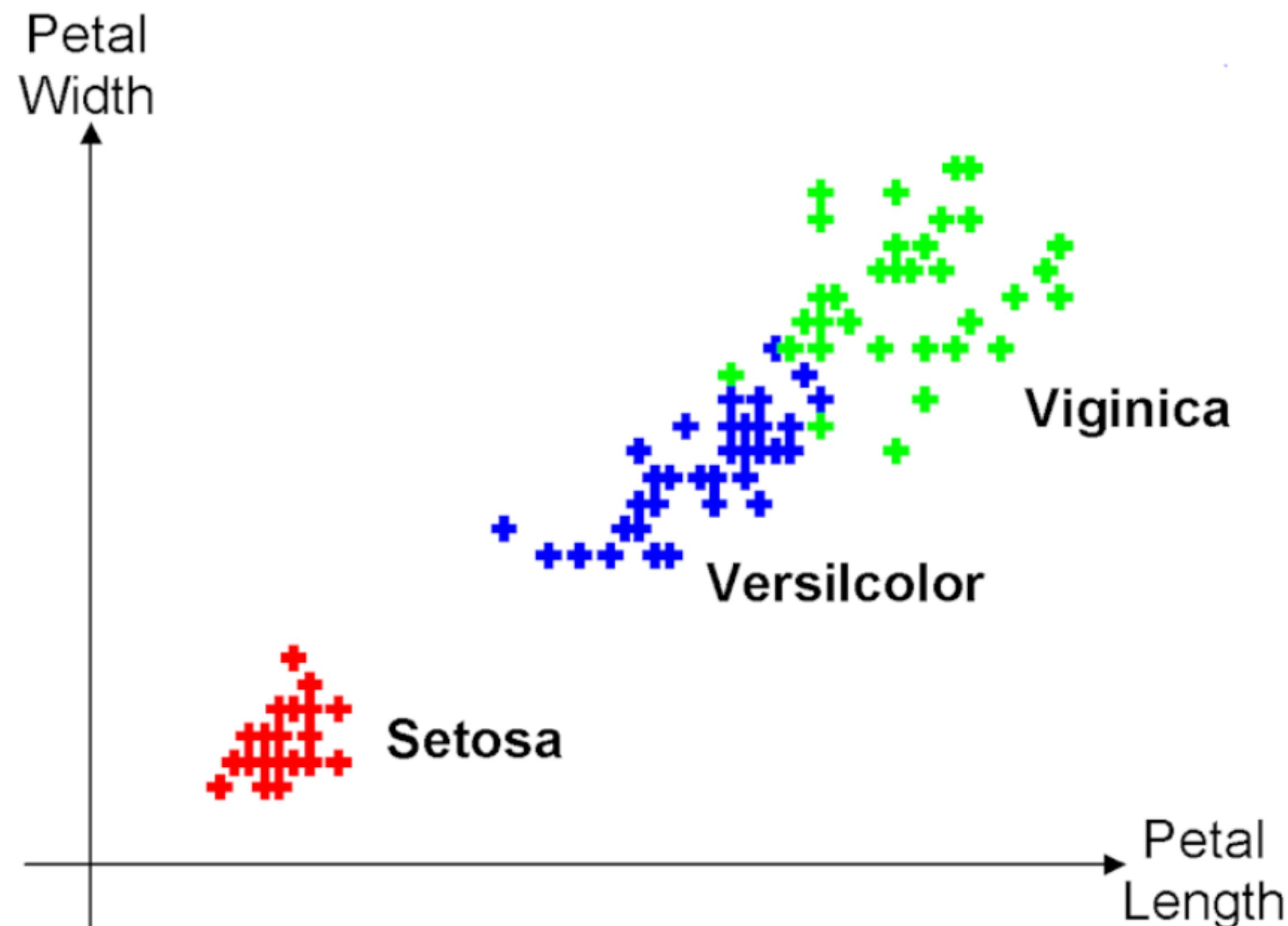
$$K(x, x') = \tanh(\rho \langle x, x' \rangle + \varrho)$$

Fourier Series函數

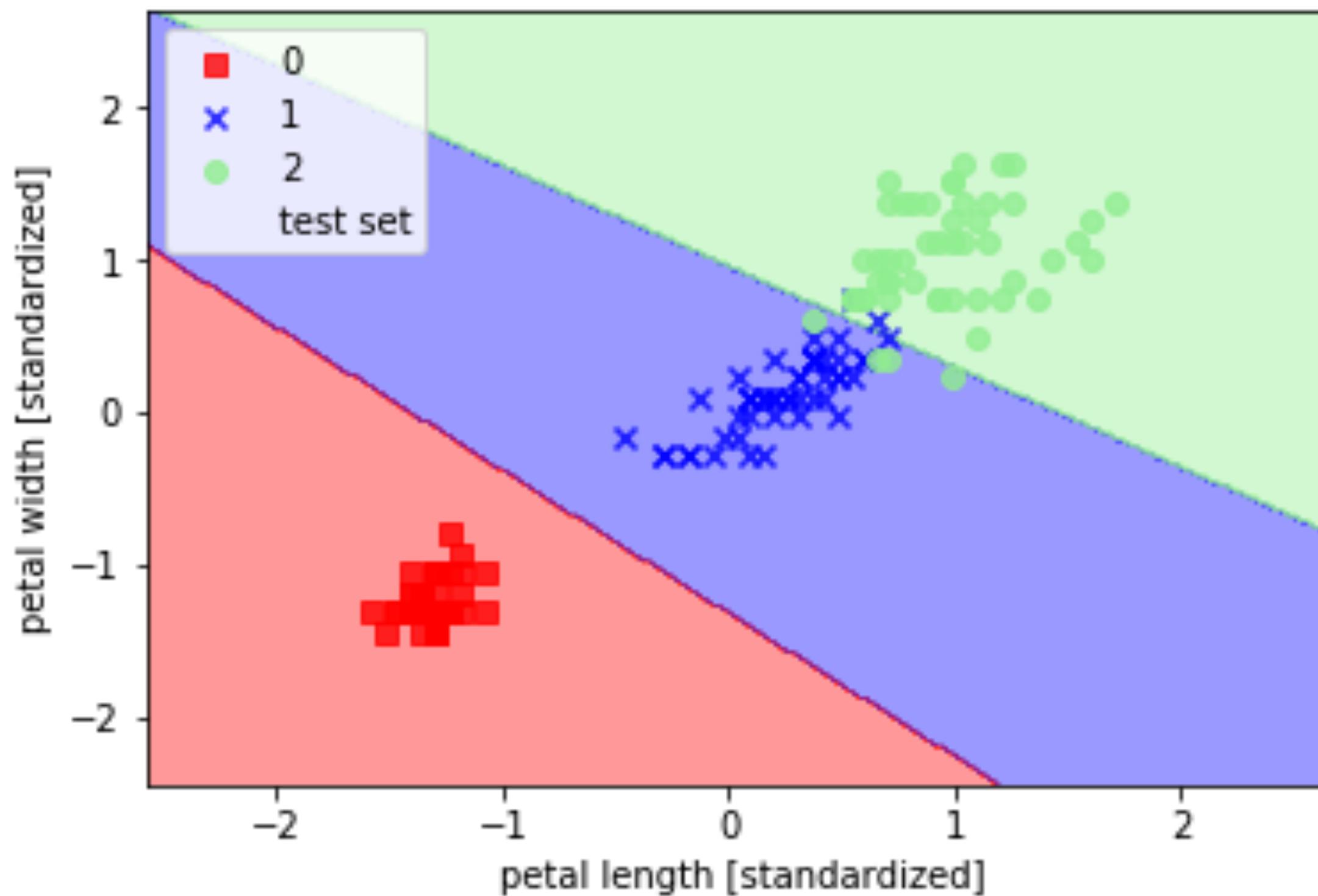
$$K(x, x') = \frac{\sin(N + \frac{1}{2})(x - x')}{\sin(\frac{1}{2}(x - x'))}$$



5. Python 實作鳶尾花分類



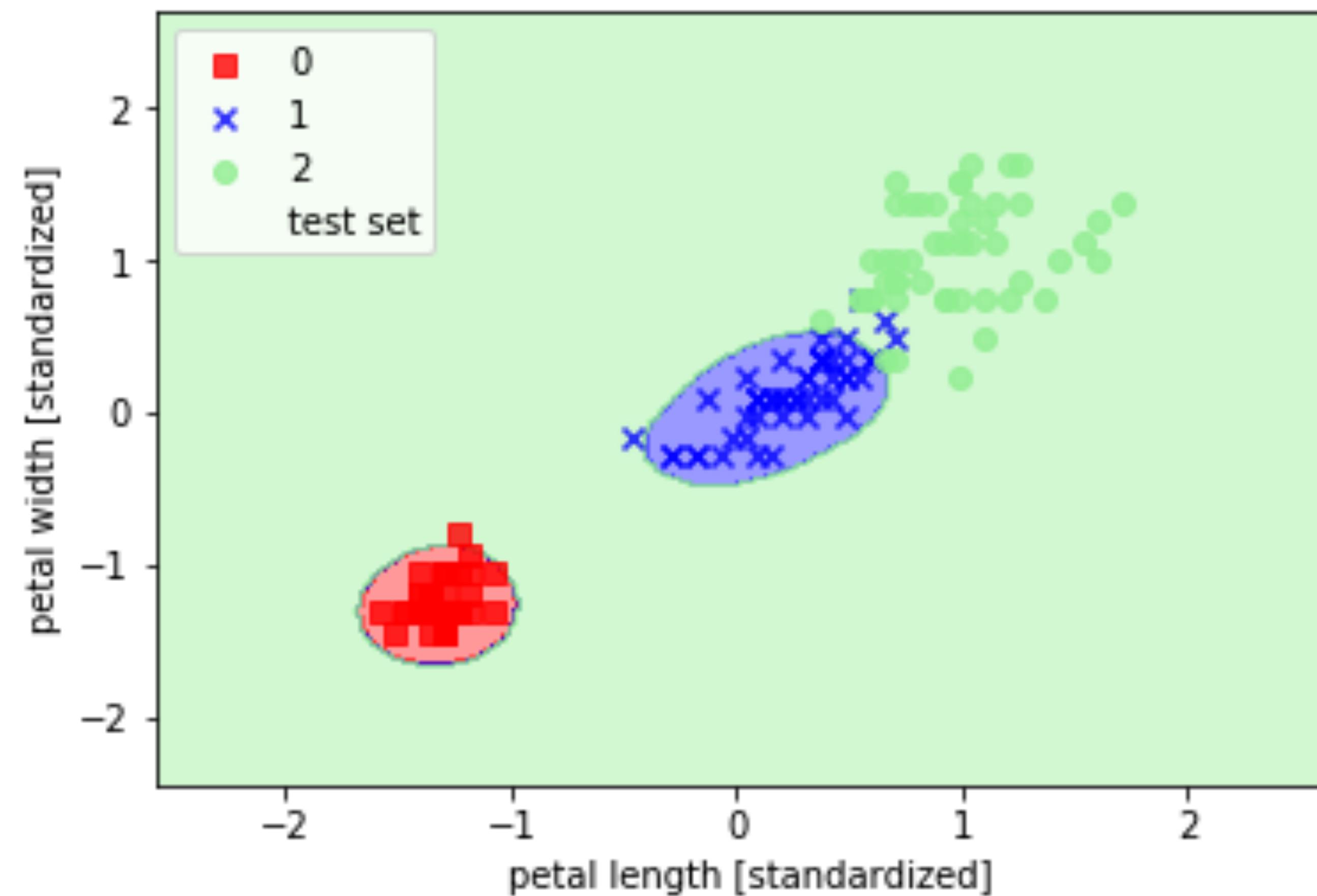
```
svm = SVC(kernel='linear',  
          C=10.0, random_state=0)
```



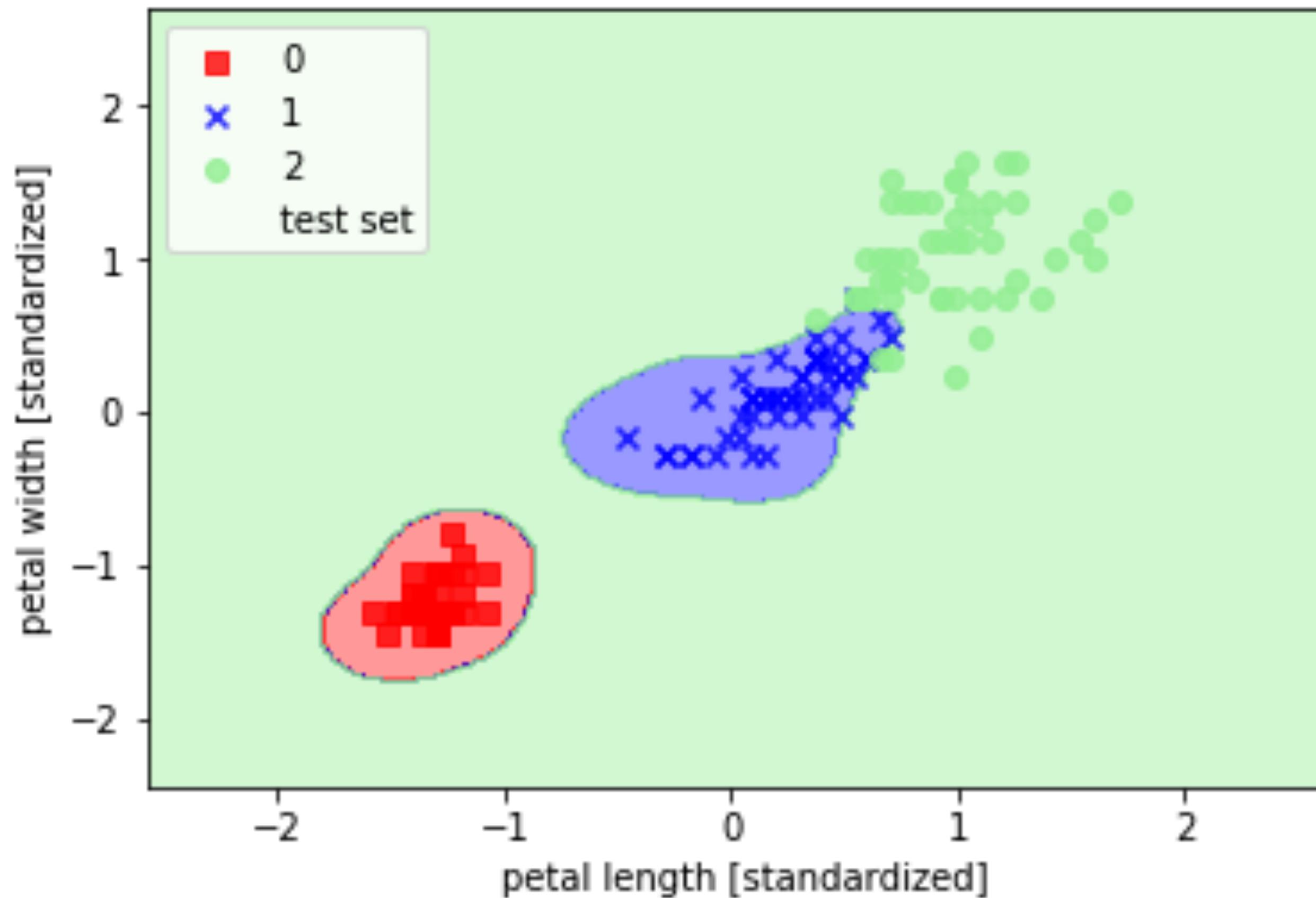
自由參數r,r值大時,決策邊界較緊,
r值小時,決策邊界較鬆

$$\gamma = \frac{1}{2\sigma^2}$$

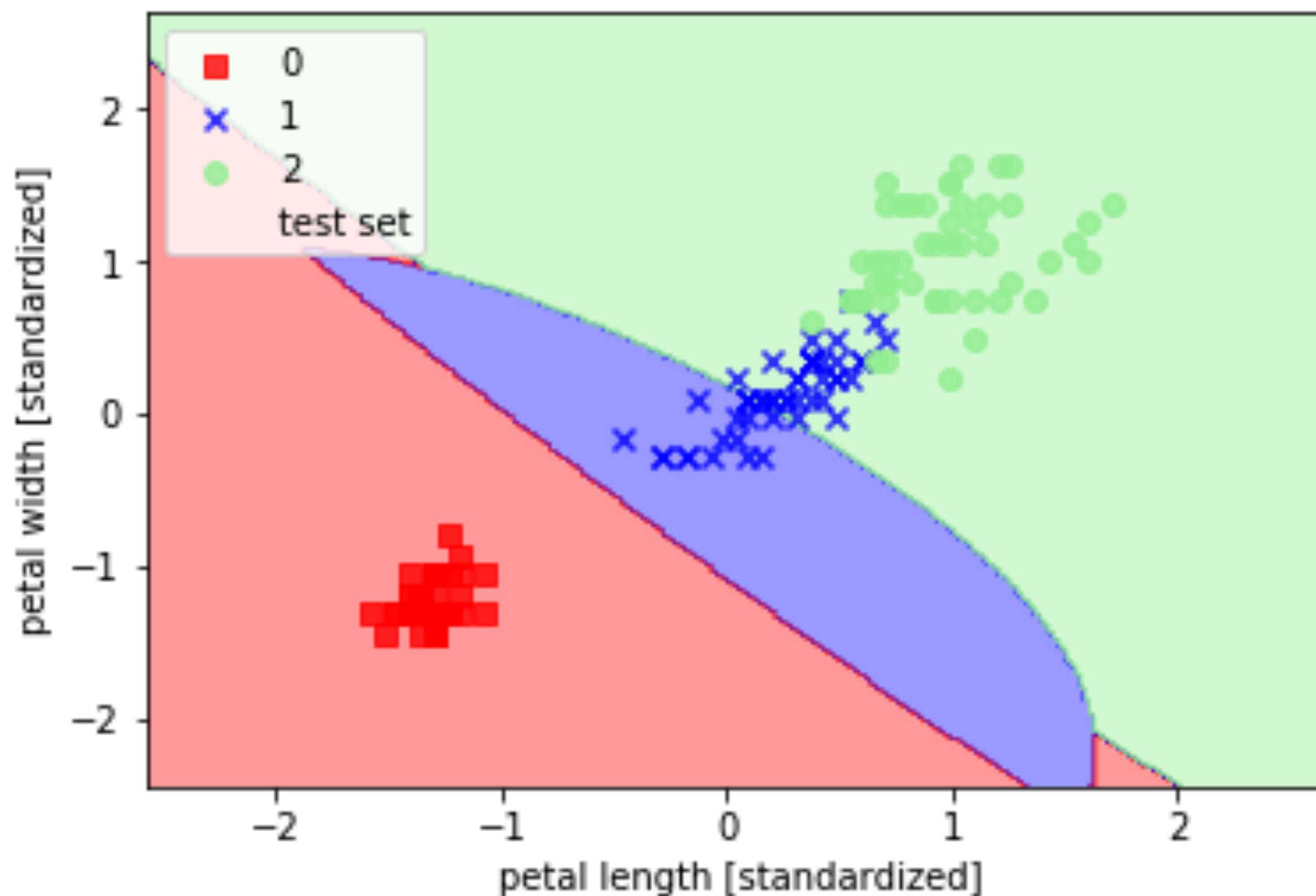
```
svm = SVC(kernel='rbf',  
random_state=0, gamma=10, C=0.10)
```



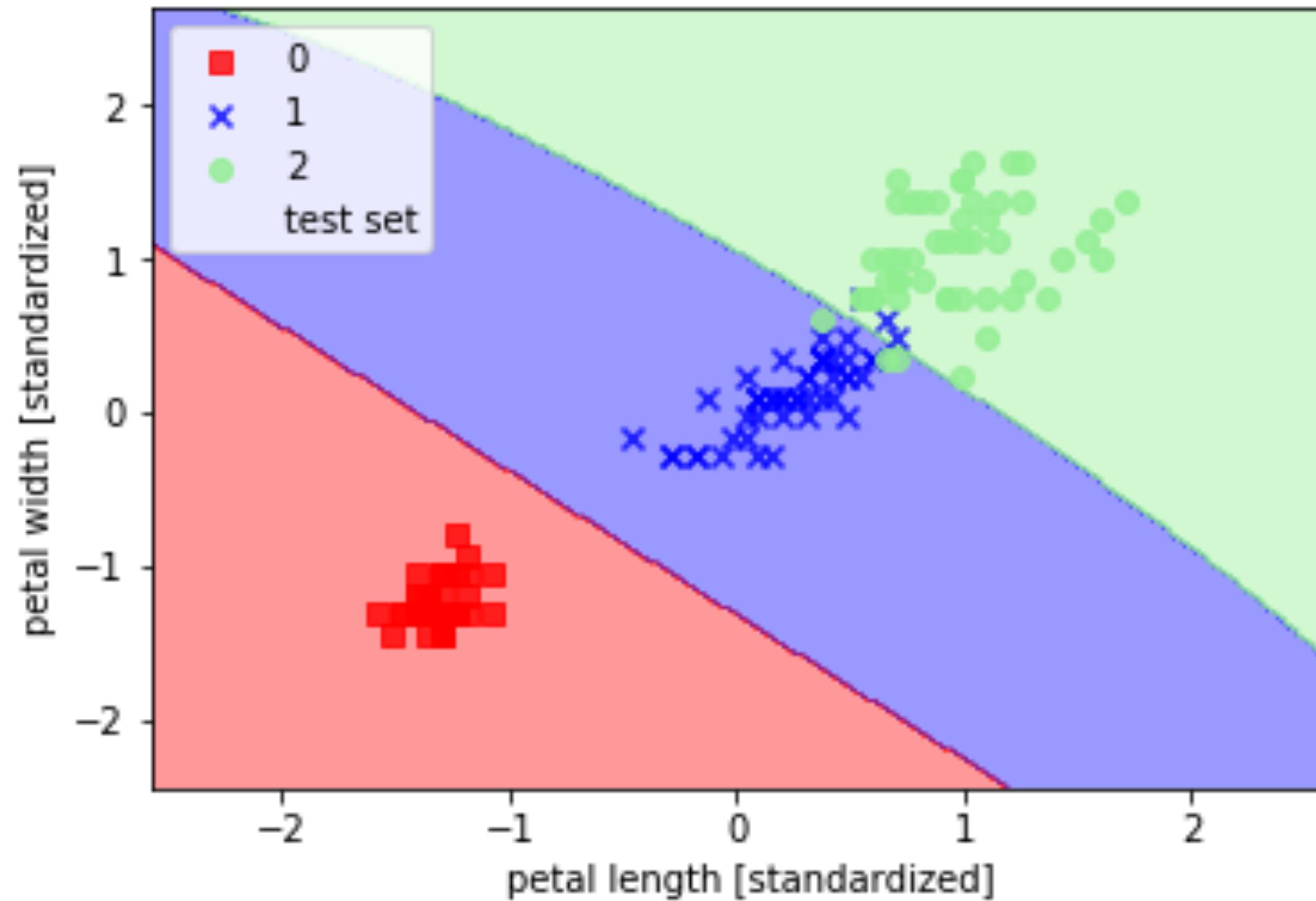
```
svm = SVC(kernel='rbf',  
random_state=0, gamma=10.0, C=10.0)
```



```
svm = SVC(kernel='rbf',  
random_state=0, gamma=0.1, C=0.10)
```



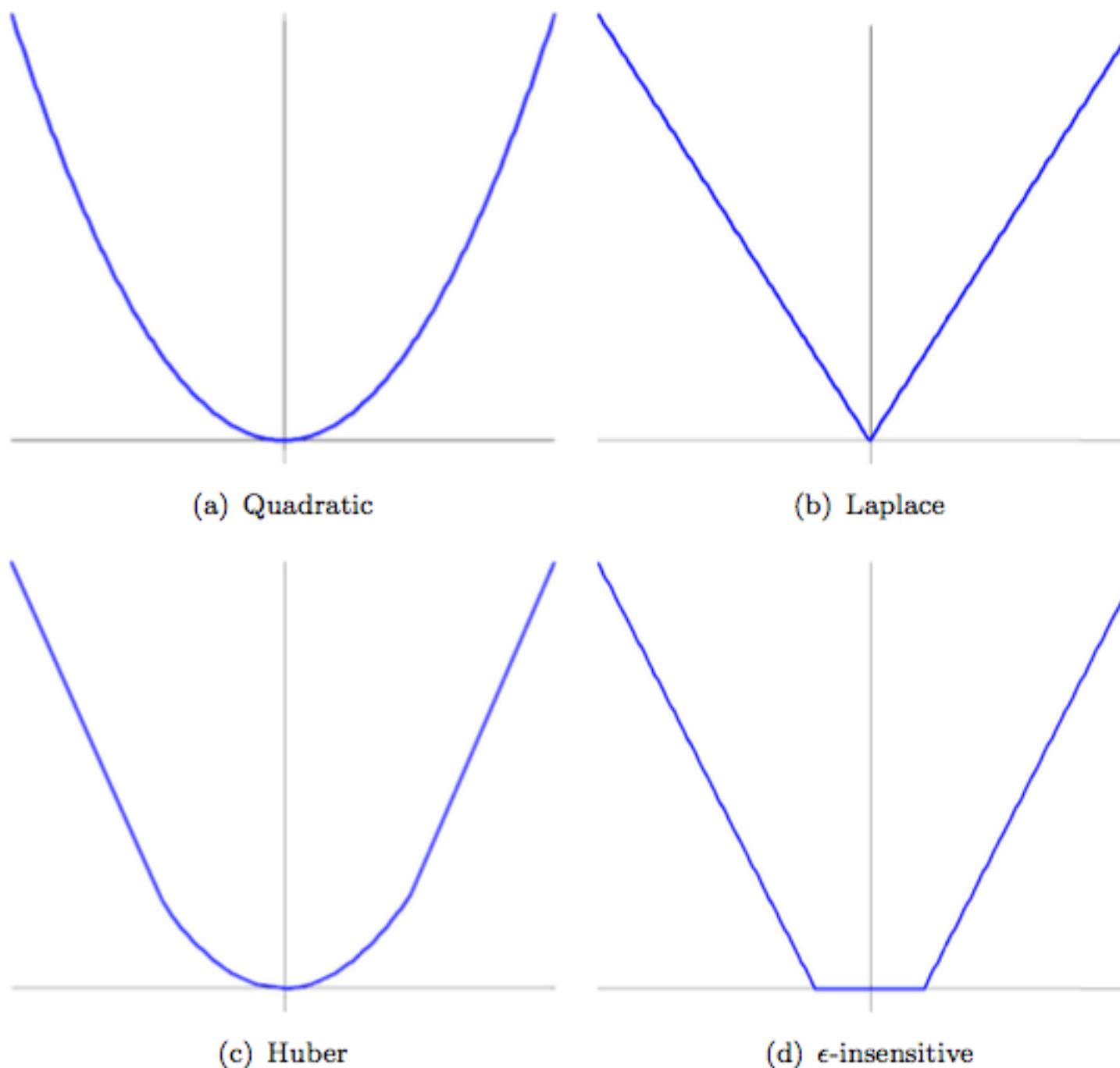
```
svm = SVC(kernel='rbf',  
random_state=0, gamma=0.1, C=10)
```





6. Support Vector Regression 迴歸

- 四種成本函數



6. Support Vector Regression 迴歸

Consider the problem of approximating the set of data,

$$\mathcal{D} = \{(x^1, y^1), \dots, (x^l, y^l)\}, \quad x \in \mathbb{R}^n, y \in \mathbb{R},$$

with a linear function,

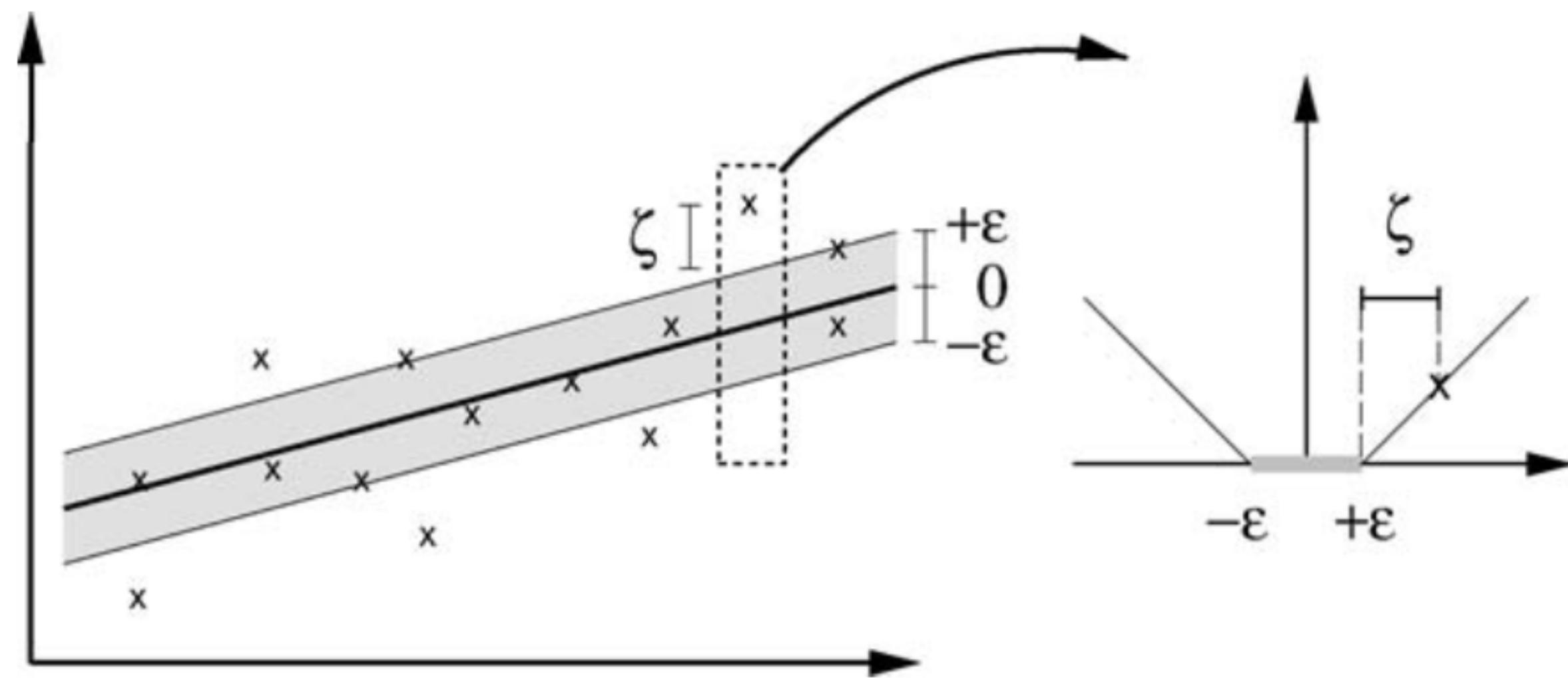
$$f(x) = \langle w, x \rangle + b.$$

the optimal regression function is given by the minimum of the functional,

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i^- + \xi_i^+),$$

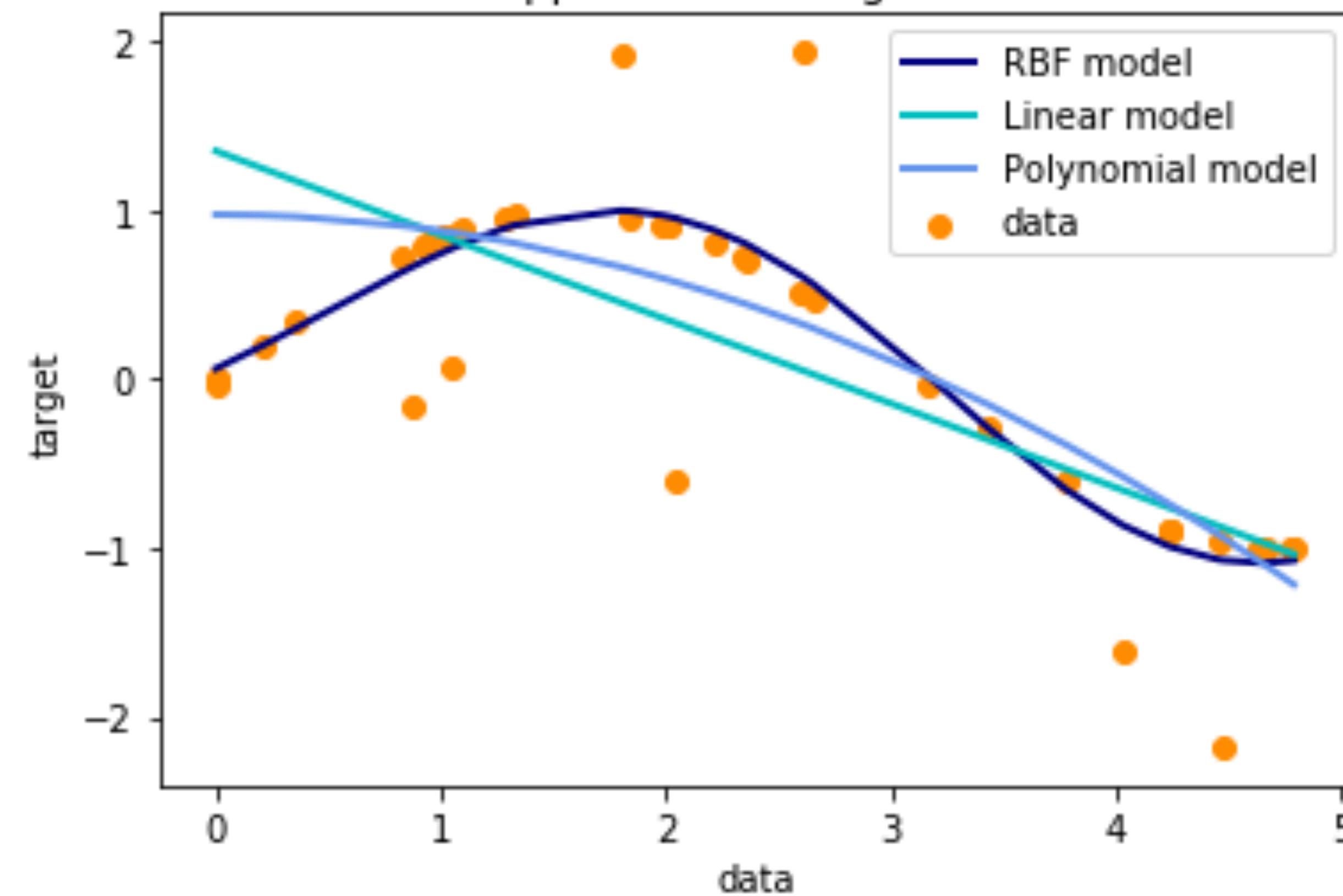
where C is a pre-specified value, and ξ^-, ξ^+ are slack variables representing upper and lower constraints on the outputs of the system.

鬆弛變數



| | Loss function | Density model |
|----------------------------|---|---|
| ε -insensitive | $c(\xi) = \xi _\varepsilon$ | $p(\xi) = \frac{1}{2(1+\varepsilon)} \exp(- \xi _\varepsilon)$ |
| Laplacian | $c(\xi) = \xi $ | $p(\xi) = \frac{1}{2} \exp(- \xi)$ |
| Gaussian | $c(\xi) = \frac{1}{2} \xi^2$ | $p(\xi) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\xi^2}{2}\right)$ |
| Huber's robust loss | $c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \xi \leq \sigma \\ \xi - \frac{\sigma}{2} & \text{otherwise} \end{cases}$ | $p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^2}{2\sigma}\right) & \text{if } \xi \leq \sigma \\ \exp\left(\frac{\sigma}{2} - \xi \right) & \text{otherwise} \end{cases}$ |
| Polynomial | $c(\xi) = \frac{1}{p} \xi ^p$ | $p(\xi) = \frac{p}{2\Gamma(1/p)} \exp(- \xi ^p)$ |
| Piecewise polynomial | $c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \xi \leq \sigma \\ \xi - \sigma \frac{p-1}{p} & \text{otherwise} \end{cases}$ | $p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^p}{p\sigma^{p-1}}\right) & \text{if } \xi \leq \sigma \\ \exp\left(\sigma \frac{p-1}{p} - \xi \right) & \text{otherwise} \end{cases}$ |

Support Vector Regression



```
9 import numpy as np
10 from sklearn.svm import SVR
11 import matplotlib.pyplot as plt
12 # #####
13 # Generate sample data
14 X = np.sort(5 * np.random.rand(40, 1), axis=0)
15 y = np.sin(X).ravel()
16 # #####
17 # Add noise to targets
18 y[::5] += 3 * (0.5 - np.random.rand(8))
19 svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
20 svr_lin = SVR(kernel='linear', C=1e3)
21 svr_poly = SVR(kernel='poly', C=1e3, degree=2)
22 y_rbf = svr_rbf.fit(X, y).predict(X)
23 y_lin = svr_lin.fit(X, y).predict(X)
24 y_poly = svr_poly.fit(X, y).predict(X)
25 lw = 2
26 plt.scatter(X, y, color='darkorange', label='data')
27 plt.plot(X, y_rbf, color='navy', lw=lw, label='RBF model')
28 plt.plot(X, y_lin, color='c', lw=lw, label='Linear model')
29 plt.plot(X, y_poly, color='cornflowerblue', lw=lw, label='Polynomial model')
30 plt.xlabel('data')
31 plt.ylabel('target')
32 plt.title('Support Vector Regression')
33 plt.legend()
34 plt.show()
```



- Thanks.





Python 分類演算法

適應性神經元

吳佳謙 老師



Python 分類演算法

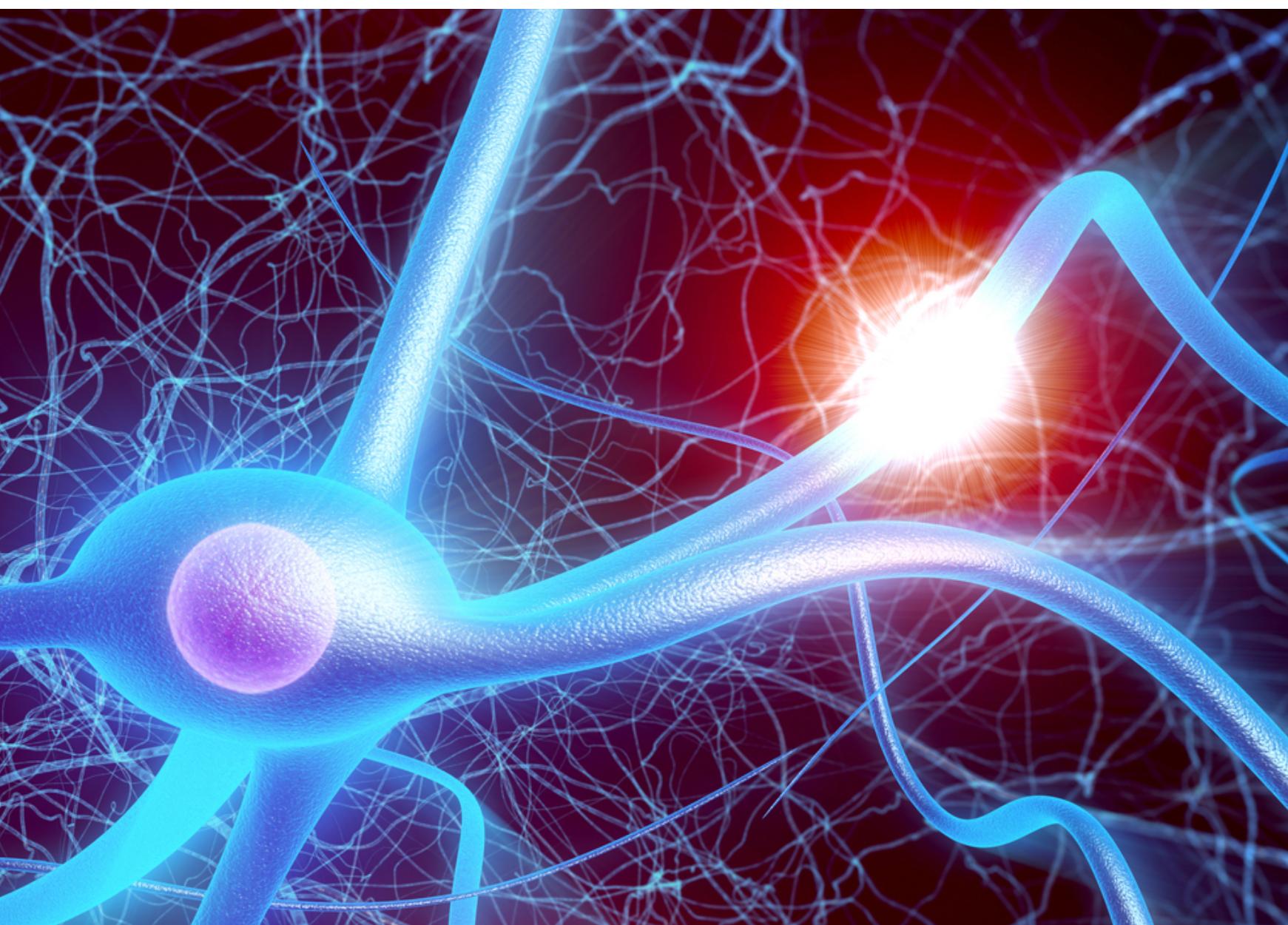
適應性神經元

1. 適應性神經元
2. 過大的學習速率, 無法收斂
3. 最小化成本函數
4. 遞減梯度
5. 以Python實作適應性神經元
6. 標準化的特徵縮放



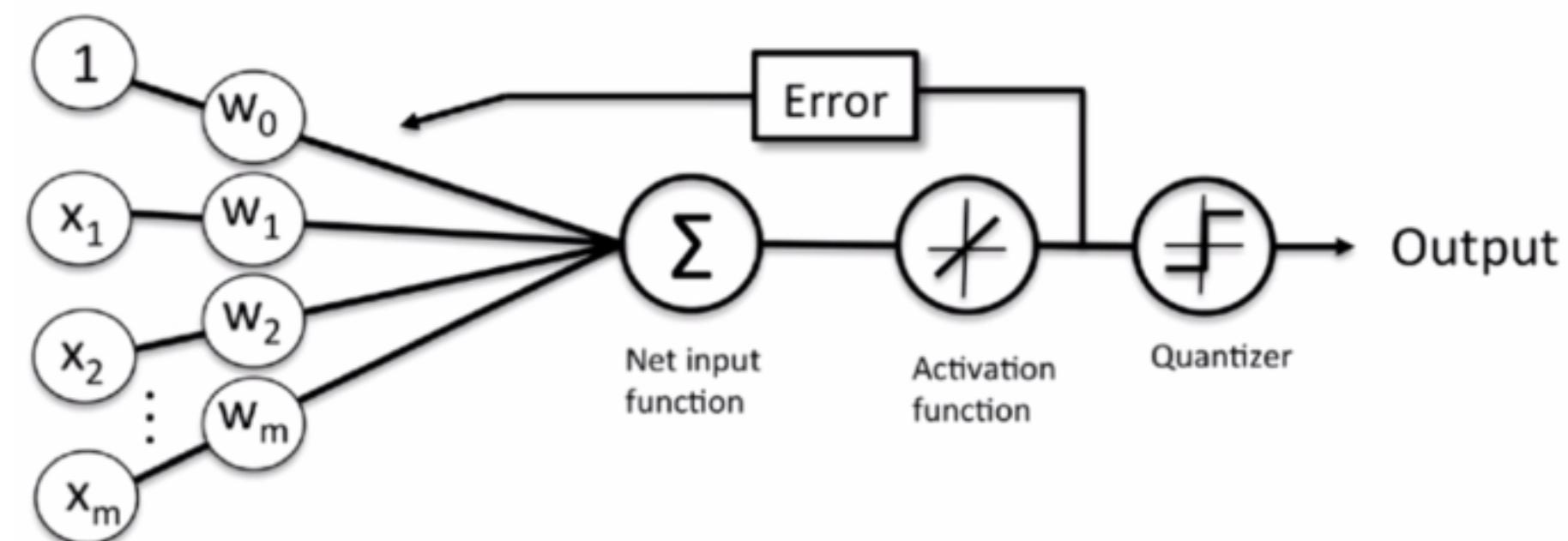


1.適應性神經元



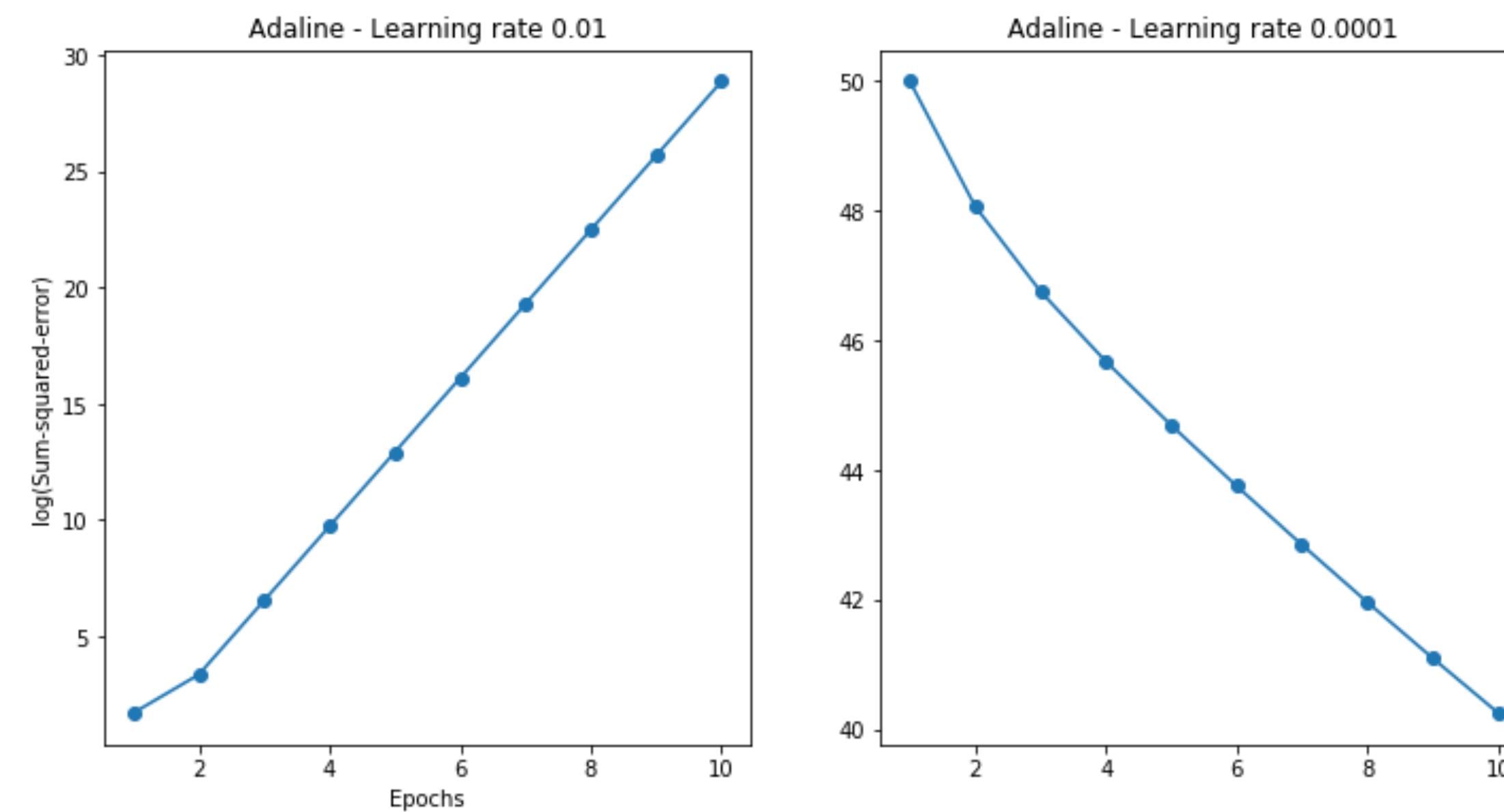
1.適應性神經元

Adaline使用輸出是連續值的線性啟動函數
來計算模型誤差,並且更新權重W





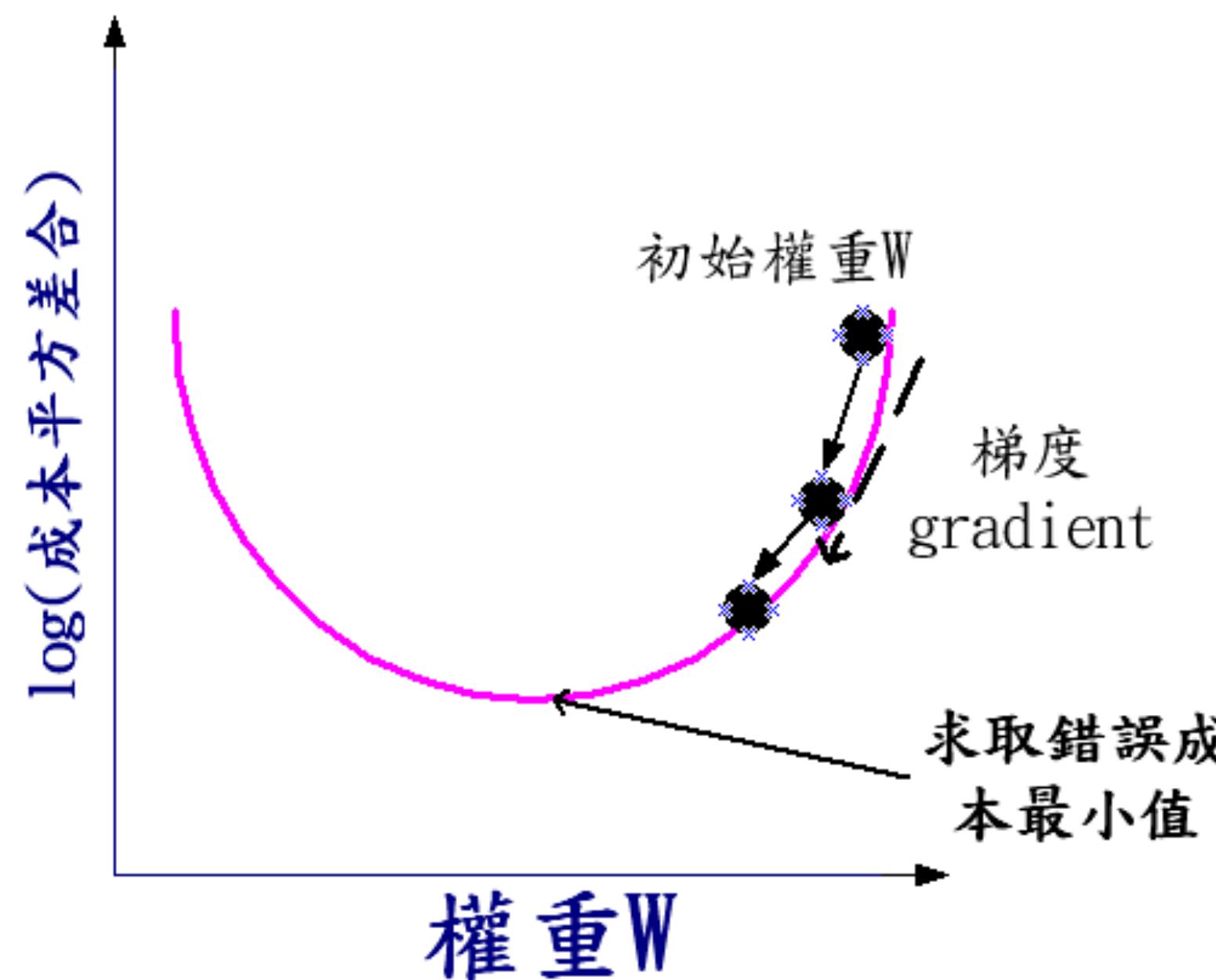
2. 過大的學習速率, 無法收斂





3. 最小化成本函數

使用成本函數的梯度來更新權重





4. 遞減梯度

$$J(w) = \frac{1}{2} * \left(\sum_i (y^i - \phi(z^i))^2 \right)$$

J(w)成本函數

$$\frac{\partial J}{\partial w_j} = - \sum_i (y^i - \phi(z^i))x_j^i$$

遞減梯度

$$w = w + \Delta w$$

Δw 是更新權重

負的表示相反方向

$$\Delta w = -\eta \nabla J(w)$$

$\nabla J(w)$ 是梯度

η 是學習速率



5.以Python實作適應性神經元

- `output = self.net_input(X)`
- `errors = (y - output)`
- `self.w_[1:] += self.eta * X.T.dot(errors)`
- `self.w_[0] += self.eta * errors.sum()`
- `cost = (errors**2).sum() / 2.0`
- `self.cost_.append(cost)`

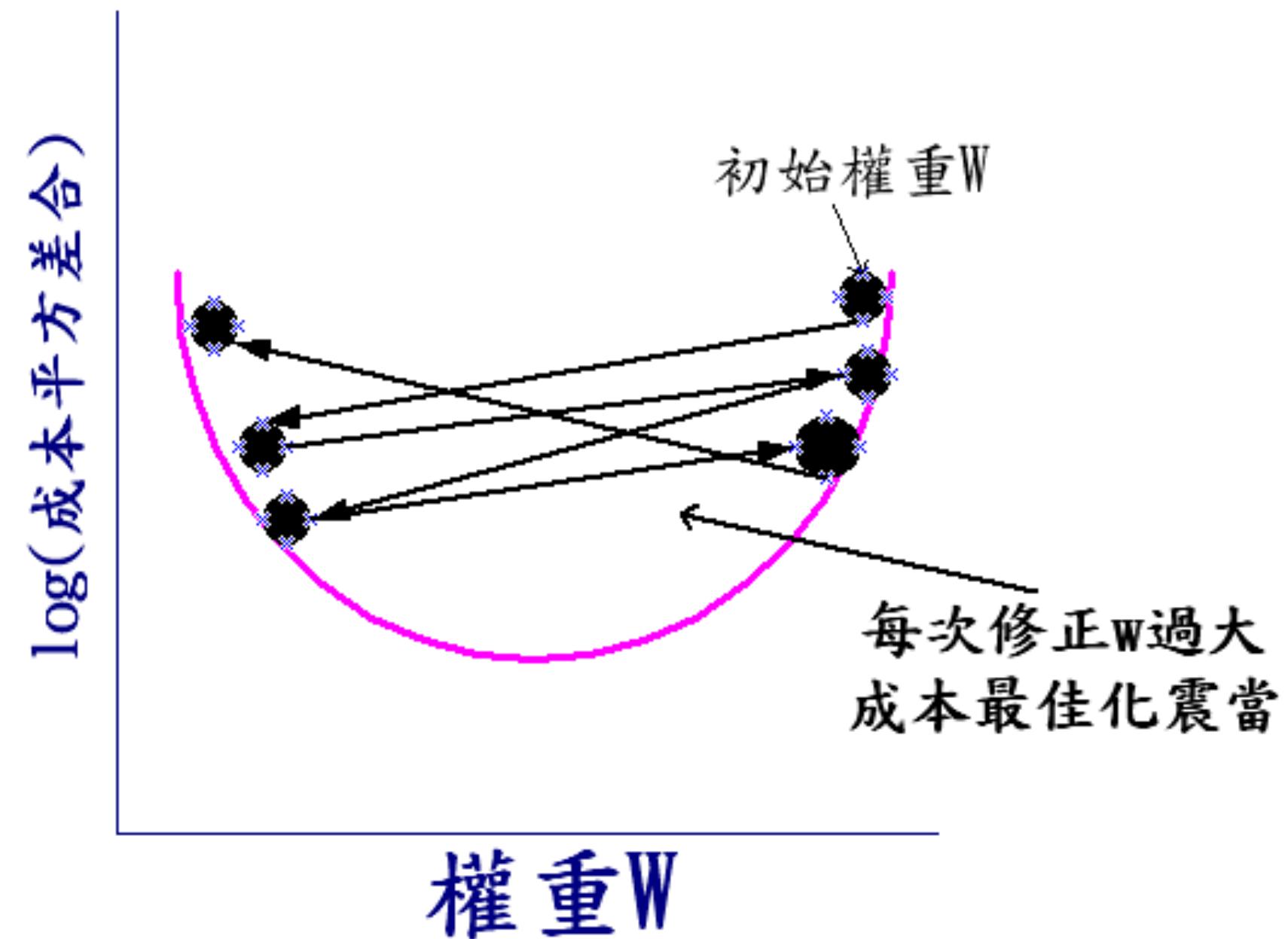
以Python實作適應性神經元

```
56 class AdalineGD(object):
57
58     def __init__(self, eta=0.01, n_iter=50):
59         self.eta = eta
60         self.n_iter = n_iter
61
62     def fit(self, X, y):
63         self.w_ = np.zeros(1 + X.shape[1])
64         self.cost_ = []
65
66         for i in range(self.n_iter):
67             output = self.net_input(X)
68             errors = (y - output)
69             self.w_[1:] += self.eta * X.T.dot(errors)
70             self.w_[0] += self.eta * errors.sum()
71             cost = (errors**2).sum() / 2.0
72             self.cost_.append(cost)
73
74         return self
```

predict()啟動函數

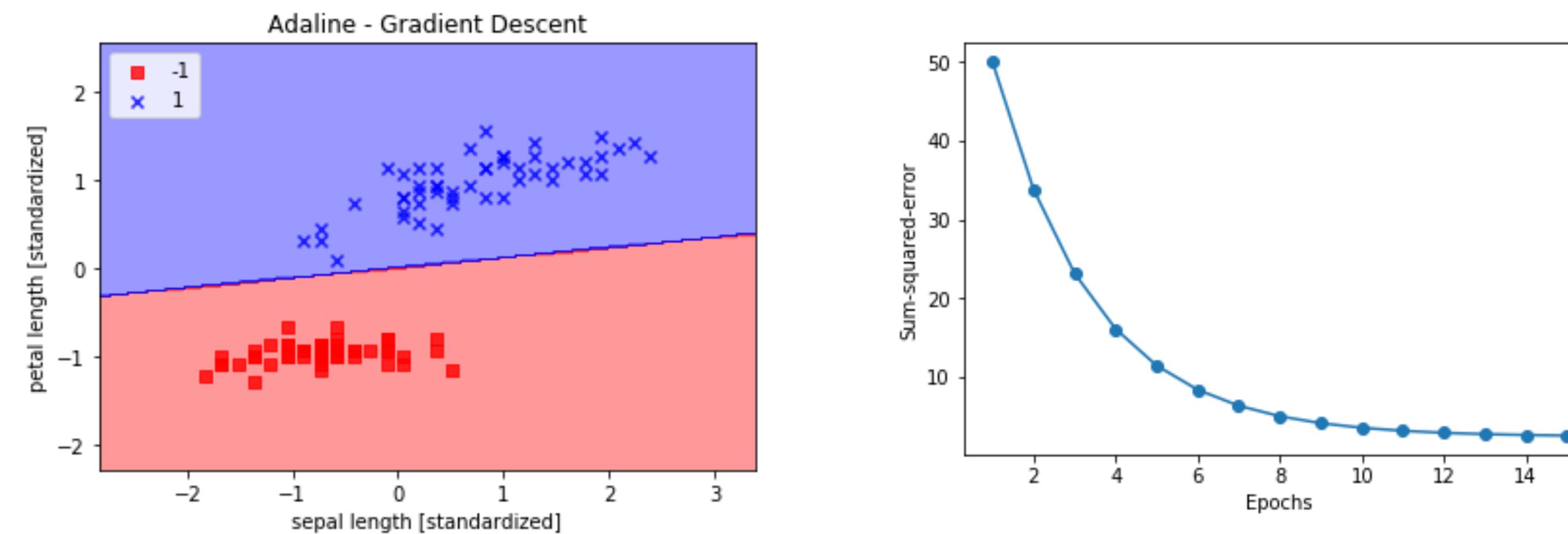
```
75     def net_input(self, X):  
76         return np.dot(X, self.w_[1:]) + self.w_[0]  
77  
78     def predict(self, X):  
79         return np.where(self.net_input(X) >= 0.0, 1, -1)
```

選擇過大的學習速率，會衝過 全域最小值





6. 標準化的特徵縮放



標準化的特徵縮放

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

μ_j 是平均值

σ_j 是標準差

- $X_{\text{std}} = \text{np.copy}(X)$
- $X_{\text{std}}[:, 0] = (X[:, 0] - X[:, 0].mean()) / X[:, 0].std()$
- $X_{\text{std}}[:, 1] = (X[:, 1] - X[:, 1].mean()) / X[:, 1].std()$

- Thanks.



Python 分類演算法

適應性神經元

吳佳謙 老師



Python 分類演算法

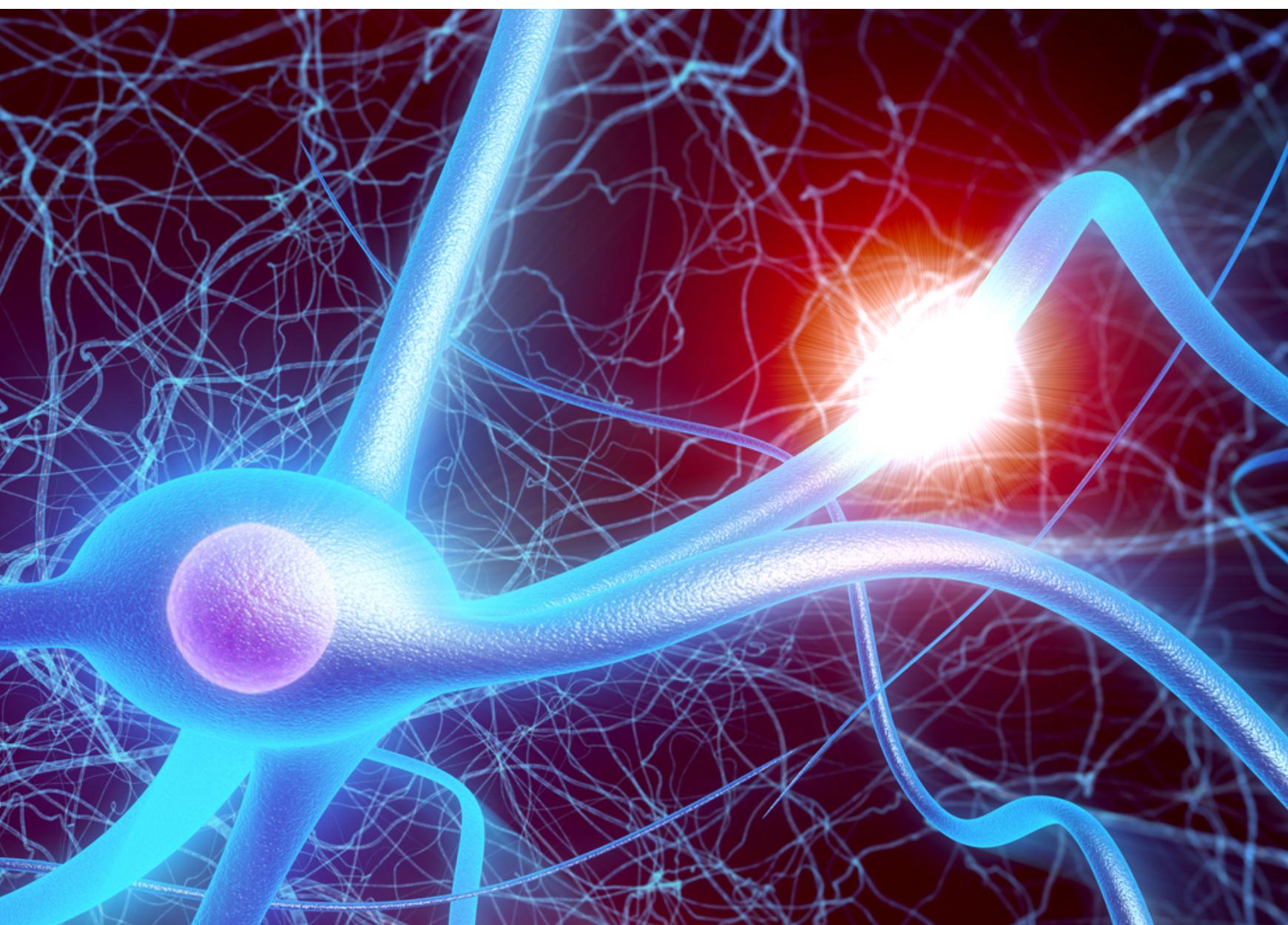
適應性神經元

1. 適應性神經元
2. 過大的學習速率, 無法收斂
3. 最小化成本函數
4. 遞減梯度
5. 以Python實作適應性神經元
6. 標準化的特徵縮放



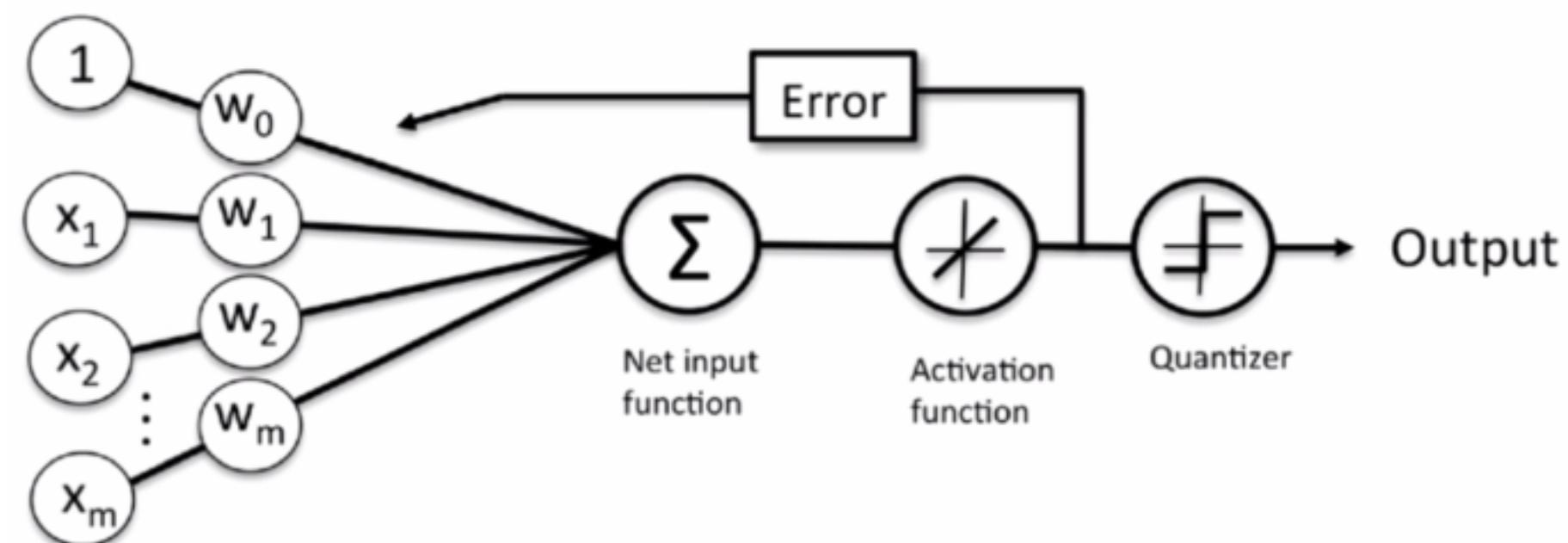


1.適應性神經元



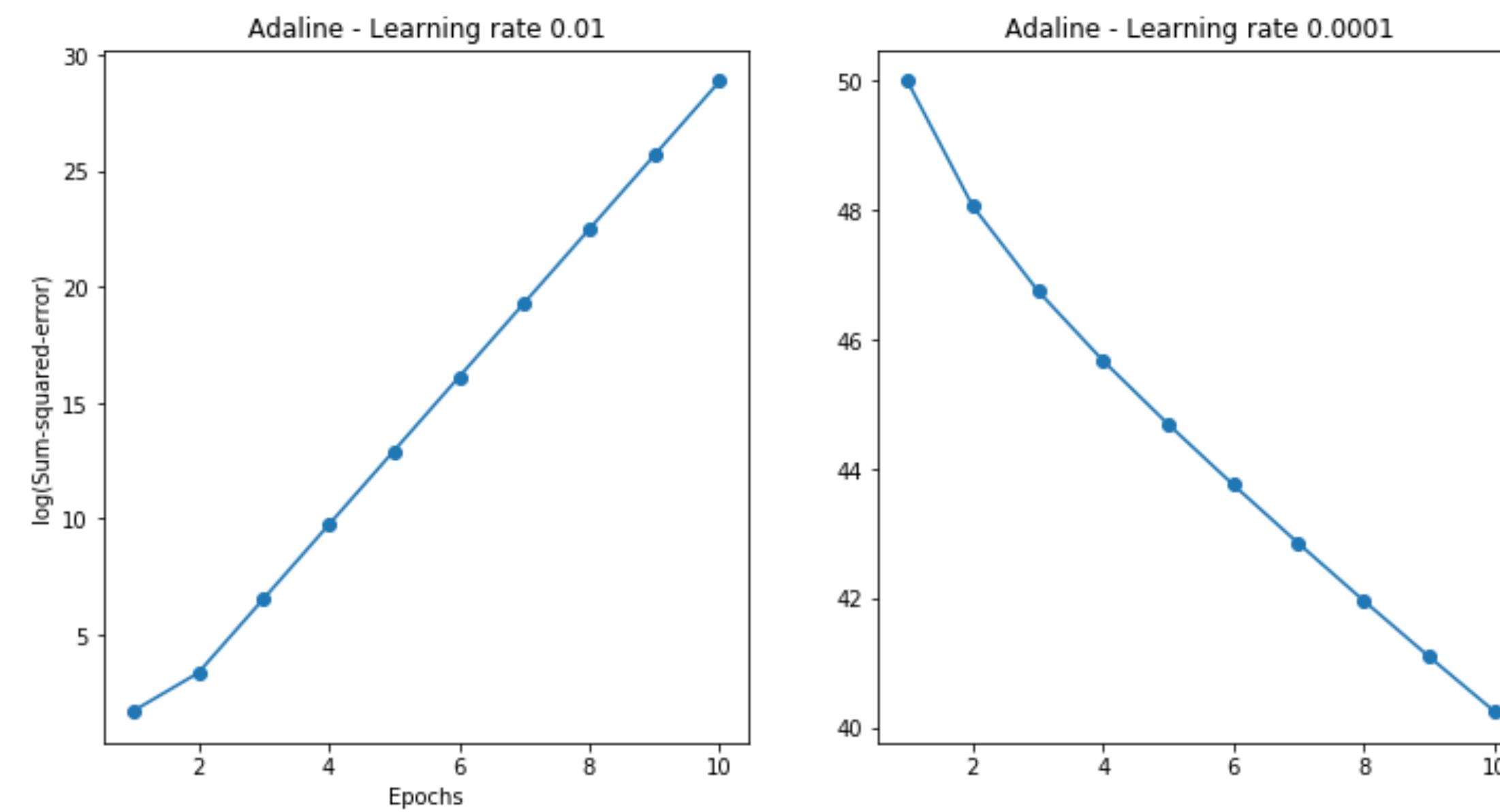
1.適應性神經元

Adaline使用輸出是連續值的線性啟動函數
來計算模型誤差,並且更新權重W





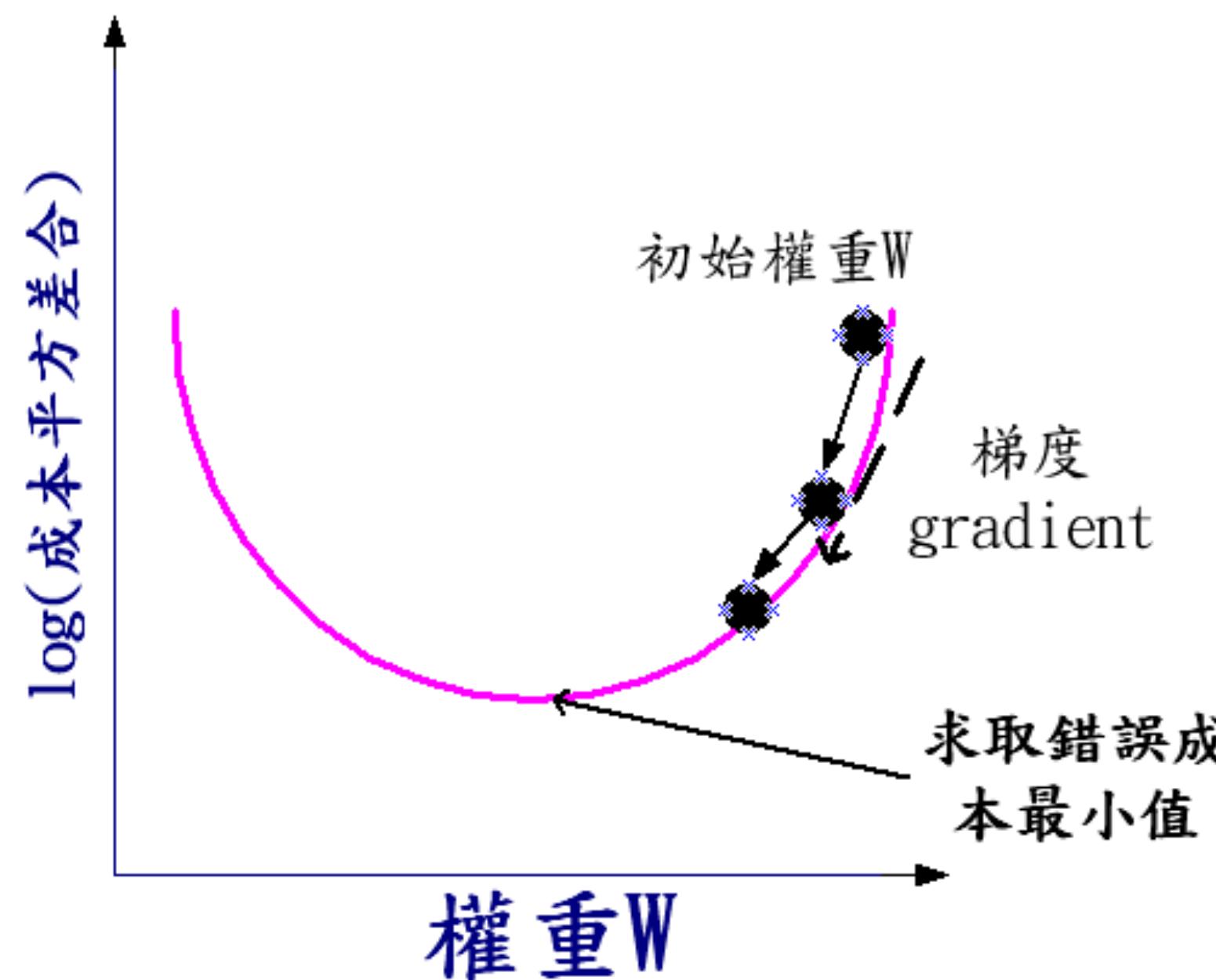
2. 過大的學習速率, 無法收斂





3. 最小化成本函數

使用成本函數的梯度來更新權重





4. 遞減梯度

$$J(w) = \frac{1}{2} * \left(\sum_i (y^i - \phi(z^i))^2 \right)$$

J(w)成本函數

$$\frac{\partial J}{\partial w_j} = - \sum_i (y^i - \phi(z^i))x_j^i$$

遞減梯度

$$w = w + \Delta w$$

Δw 是更新權重

負的表示相反方向

$$\Delta w = -\eta \nabla J(w)$$

$\nabla J(w)$ 是梯度

η 是學習速率



5.以Python實作適應性神經元

- `output = self.net_input(X)`
- `errors = (y - output)`
- `self.w_[1:] += self.eta * X.T.dot(errors)`
- `self.w_[0] += self.eta * errors.sum()`
- `cost = (errors**2).sum() / 2.0`
- `self.cost_.append(cost)`

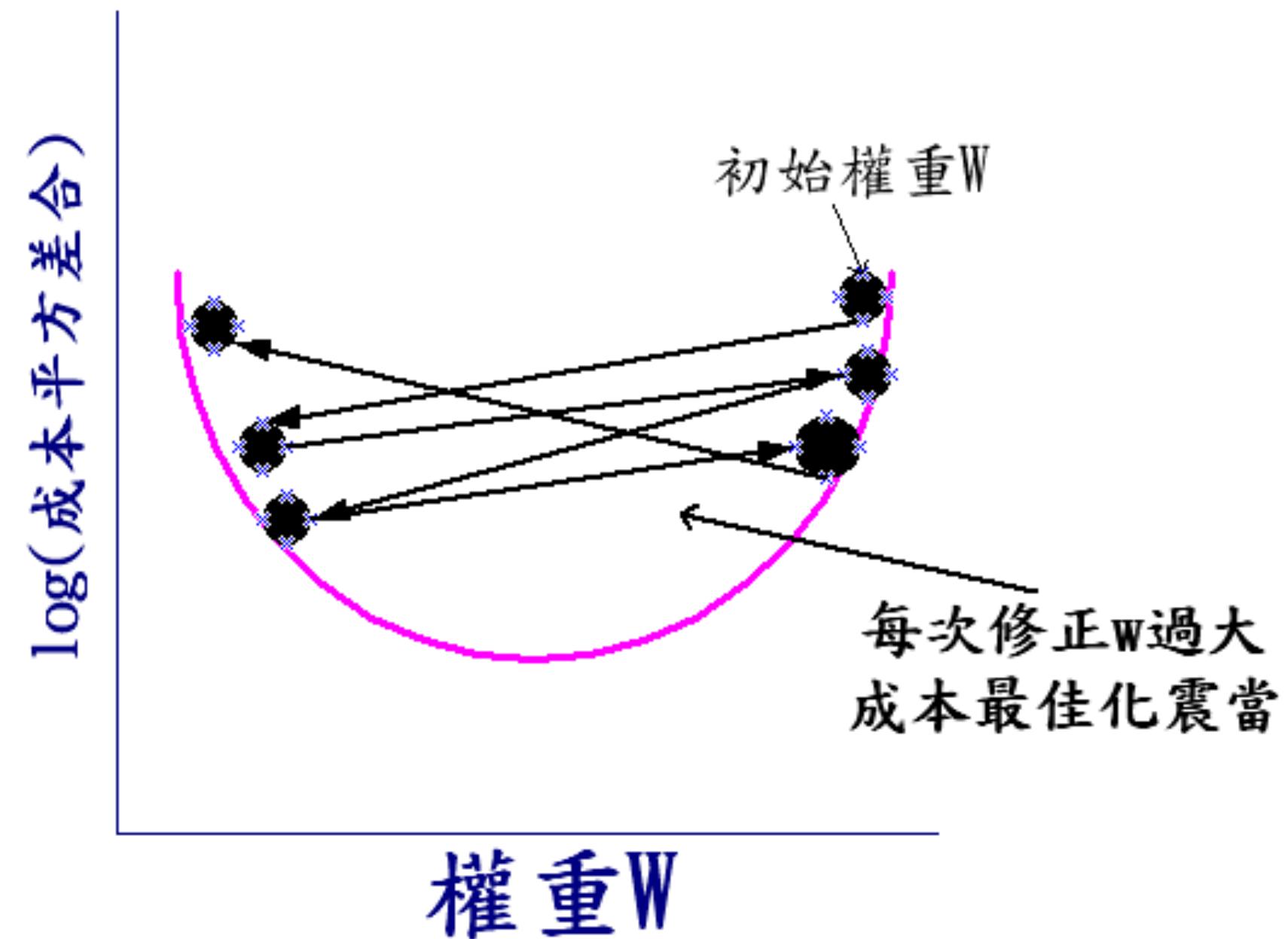
以Python實作適應性神經元

```
56 class AdalineGD(object):
57
58     def __init__(self, eta=0.01, n_iter=50):
59         self.eta = eta
60         self.n_iter = n_iter
61
62     def fit(self, X, y):
63         self.w_ = np.zeros(1 + X.shape[1])
64         self.cost_ = []
65
66         for i in range(self.n_iter):
67             output = self.net_input(X)
68             errors = (y - output)
69             self.w_[1:] += self.eta * X.T.dot(errors)
70             self.w_[0] += self.eta * errors.sum()
71             cost = (errors**2).sum() / 2.0
72             self.cost_.append(cost)
73
74         return self
```

predict()啟動函數

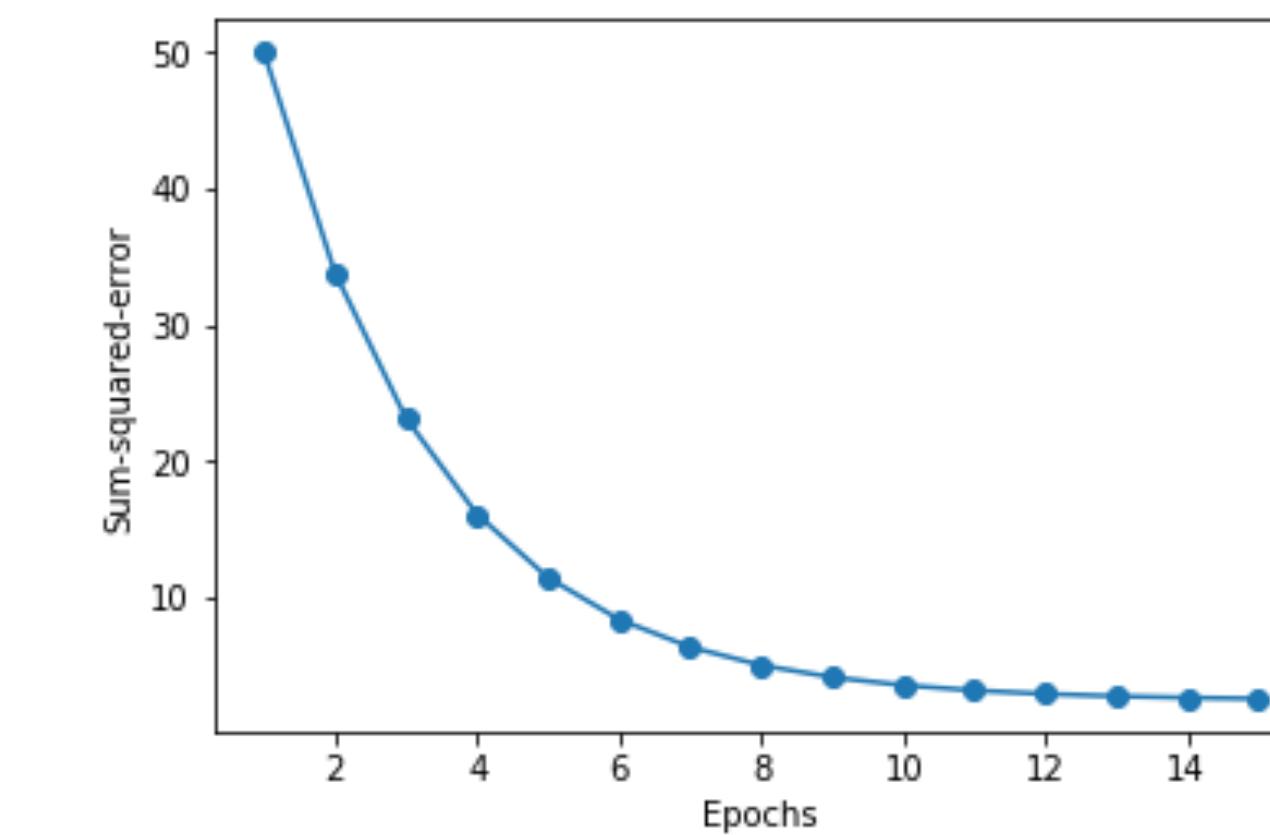
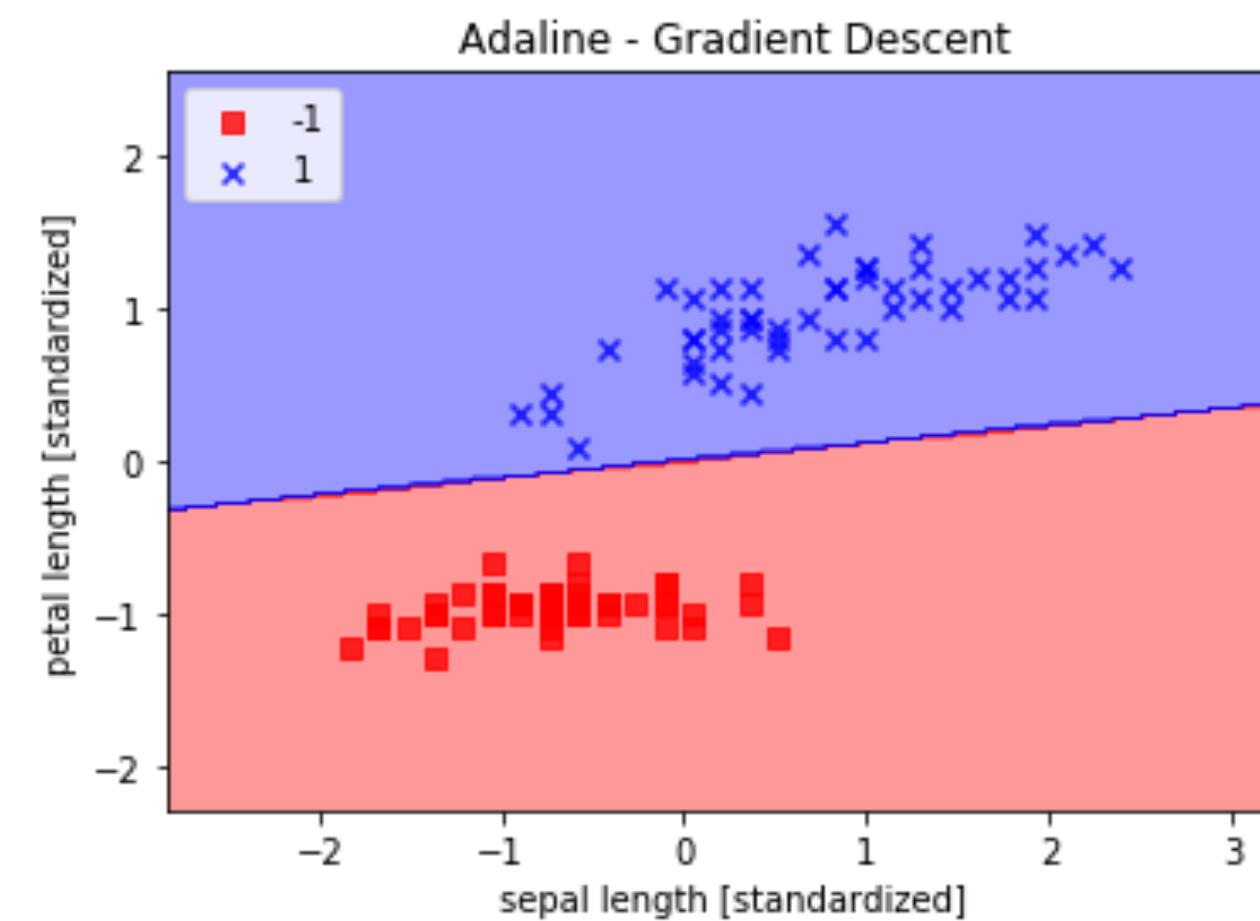
```
75     def net_input(self, X):  
76         return np.dot(X, self.w_[1:]) + self.w_[0]  
77  
78     def predict(self, X):  
79         return np.where(self.net_input(X) >= 0.0, 1, -1)
```

選擇過大的學習速率，會衝過全域最小值





6. 標準化的特徵縮放



標準化的特徵縮放

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

μ_j 是平均值

σ_j 是標準差

- $X_{\text{std}} = \text{np.copy}(X)$
- $X_{\text{std}}[:, 0] = (X[:, 0] - X[:, 0].mean()) / X[:, 0].std()$
- $X_{\text{std}}[:, 1] = (X[:, 1] - X[:, 1].mean()) / X[:, 1].std()$

- Thanks.

Python連接MySQL

吳佳謙 老師

1. Mac環境,在Mac 安裝brew
2. 在Windows或Mac直接使用pip安裝PyMySQL
3. 在Python3.x中安裝PyMySQL函式庫
4. 建立temp資料庫
5. 使用pip list觀看Python的已安裝模組是否有
PyMySQL
6. import pymysql
7. 建立資料表

1.Mac環境,在Mac 安裝brew

OSX 套件管理軟體，常常在安裝不同版本的軟體時，可能會造成不同軟體上的衝突，而 Homebrew 可以幫助你管理這些不同軟體之間的相依性問題。

在安裝完Xcode後，開啟Xcode同意授權，等Xcode授權完畢且開啟完成後，開啟 terminal 之後，執行:

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

2.在Windows或Mac直接使用pip

安裝PyMySQL

```
pip install PyMySQL
```

3.在Python3.x中安裝PyMySQL函式庫

PyMySQL

PyMySQL 是在 Python3.x 版本中用於連接 MySQL的函式庫，Python2中則使用 mysqldb 。

PyMySQL 遵循 Python 資料庫 API v2.0規範，並包含了 pure-Python MySQL 使用者端函式庫 。

4. 建立temp資料庫

- create database temp;

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| company        |
| mysql          |
| performance_schema |
| sakila         |
| sys            |
| world          |
+-----+
7 rows in set (0.05 sec)

mysql> create database temp;
Query OK, 1 row affected (0.00 sec)
```

5. 使用pip list 觀看Python的已安裝模組是否有PyMySQL

- pip list
PyMySQL (0.7.11)

6.import pymyql

```
import pymysql

#打開資料庫連接,設定連接帳號root,密碼322739aa,資料庫temp
db = pymysql.connect("localhost","root","322739aa","temp" )

# 使用cursor()方法得到操作指標
cursor = db.cursor()

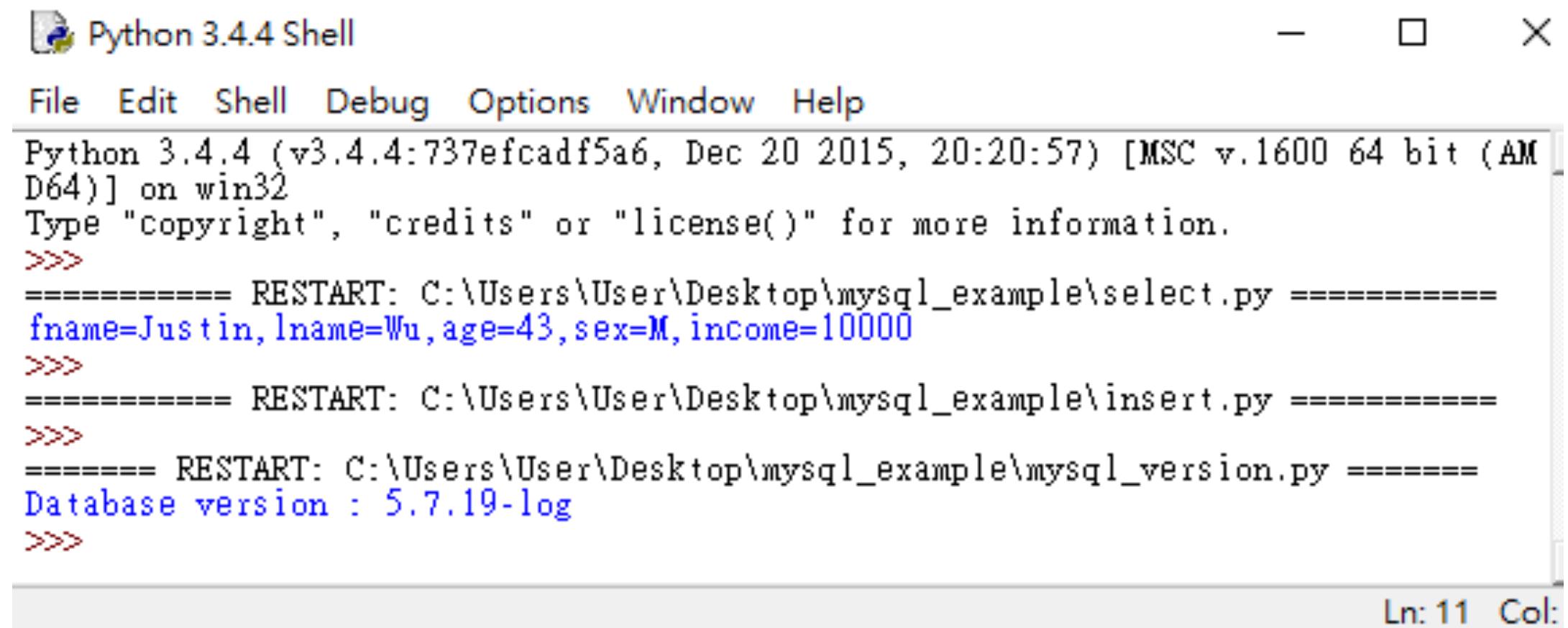
# 執行SQL語句
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法得到資料
data = cursor.fetchone()

print ("Database version : %s " % data)

# 關閉資料庫連接
db.close()
```

执行程式



The screenshot shows the Python 3.4.4 Shell window. The title bar says "Python 3.4.4 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\User\Desktop\mysql_example\select.py ======  
fname=Justin, lname=Wu, age=43, sex=M, income=10000  
>>>  
===== RESTART: C:\Users\User\Desktop\mysql_example\insert.py ======  
>>>  
===== RESTART: C:\Users\User\Desktop\mysql_example\mysql_version.py ======  
Database version : 5.7.19-log  
>>>
```

At the bottom right of the shell window, it says "Ln: 11 Col: 1".

```
$ python mysql_version.py  
Database version : 5.7.19
```

7. 建立資料表

- `#!/usr/bin/python3`
- `import pymysql`
- `#打開資料庫連接`
- `db = pymysql.connect("localhost","root","322739aa","temp")`
- `# 使用cursor()方法得到操作指標`
- `cursor = db.cursor()`
- `# 使用 execute() 方法執行 SQL，如果資料表存在則刪除`
- `cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")`
- `# SQL 語法建立資料表EMPLOYEE`
- `sql = """CREATE TABLE EMPLOYEE (FIRST_NAME CHAR(20) NOT NULL, LAST_NAME CHAR(20),
AGE INT, SEX CHAR(1), INCOME FLOAT)"""`
- `cursor.execute(sql)`
- `# 關閉資料庫連接`
- `db.close()`

建立資料表

```
$ python create_mysql.py
```

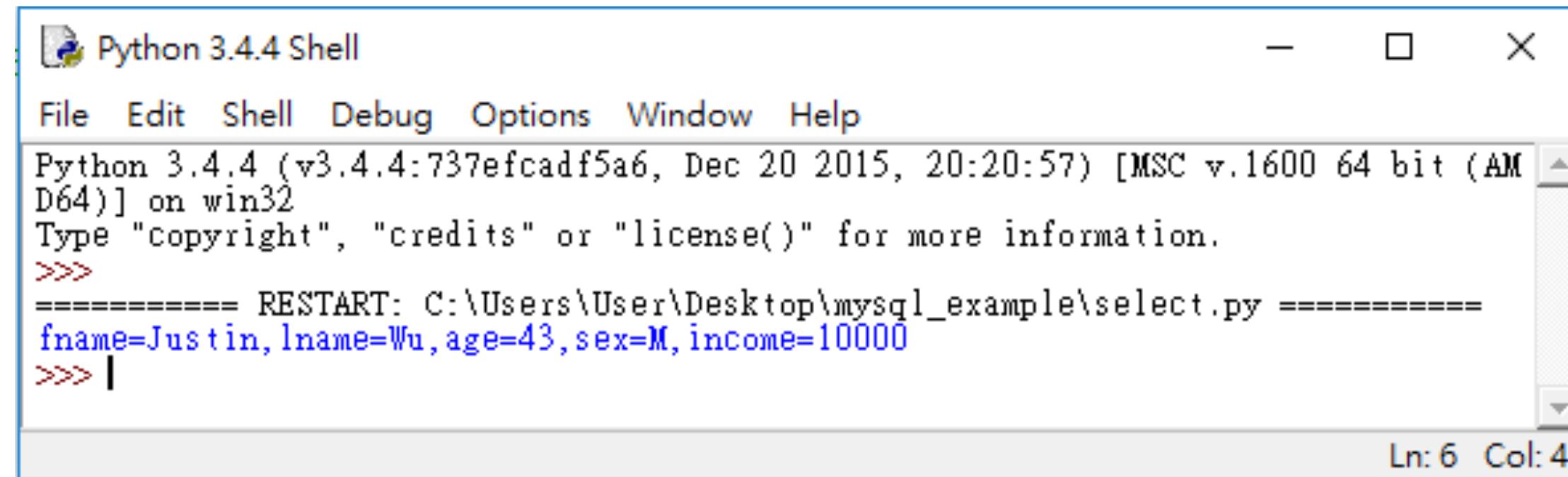
- import pymysql
- #打開資料庫連接
- db = pymysql.connect("localhost","root","322739aa","temp")
- # 使用cursor()方法得到操作指標
- cursor = db.cursor()
- # SQL 語法刪除資料
- sql = """INSERT INTO EMPLOYEE(FIRST_NAME, LAST_NAME, AGE, SEX, INCOME) VALUES ('Justin', 'Wu', 43, 'M', 10000)"""
- try:
- # 執行SQL語句
- cursor.execute(sql)
- # 提交到資料庫系統執行
- db.commit()
- print("insert a record into temp")
- except:
- # 發生異常錯誤時回復
- db.rollback()
- # 關閉資料庫連接
- db.close()

執行插入

```
$ python insert.py  
insert a record into temp
```

- import pymysql
- db = pymysql.connect("localhost","root","322739aa","temp") #打開資料庫連接
- cursor = db.cursor() # 使用cursor()方法得到操作指標
- sql = "SELECT * FROM EMPLOYEE WHERE INCOME > '%d'" % (1000) # SQL 語法
- try:
 - cursor.execute(sql) # 執行SQL語句
 - results = cursor.fetchall() # 使用 fetchall()得到所有資料
 - for row in results:
 - fname = row[0]
 - lname = row[1]
 - age = row[2]
 - sex = row[3]
 - income = row[4]
 - #列印資料
 - print ("fname=%s,lname=%s,age=%d,sex=%s,income=%d" %(fname, lname, age, sex, income))
 - except:
 - print ("Error: unable to fetch data")
 - db.close() # 關閉資料庫連接

執行查詢



The screenshot shows the Python 3.4.4 Shell window. The title bar reads "Python 3.4.4 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the Python interpreter's welcome message and a command-line session. The command `fname=Justin, lname=Wu, age=43, sex=M, income=10000` is entered and executed, resulting in the output `fname=Justin, lname=Wu, age=43, sex=M, income=10000`. The status bar at the bottom right indicates "Ln: 6 Col: 4".

```
$ python select.py
fname=Justin, lname=Wu, age=43, sex=M, income=1
0000
```

更新update資料表

```
• #!/usr/bin/python3  
• import pymysql  
• db = pymysql.connect("localhost","root","322739aa","temp" ) #打開資料庫連接  
• cursor = db.cursor() # 使用cursor()方法得到操作指標  
• # SQL 更新語句  
• sql = "UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = '%c'" % ('M')  
• try:  
•     # 執行SQL語句  
•     cursor.execute(sql)  
•     # 提交到資料庫系統執行  
•     db.commit()  
• except:  
•     # 發生異常錯誤時回復  
•     db.rollback()  
• 
```

- `#!/usr/bin/python3`
- `import pymysql`
- `#打開資料庫連接`
- `db = pymysql.connect("localhost","root","322739aa","temp")`
- `# 使用cursor()方法得到操作指標`
- `cursor = db.cursor()`
- `# SQL 語法刪除資料`
- `sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)`
- `try:`
- `# 執行SQL語句`
- `cursor.execute(sql)`
- `# 提交到資料庫系統執行`
- `db.commit()`
- `except:`
- `# 發生異常錯誤時回復`
- `db.rollback()`
- `# 關閉資料庫連接`
- `db.close()`

執行刪除

```
$ python delete.py
```

- Thanks.