



Python程式語言起步走~

使用 Python 來做機器學習初探

洪暉鈞

臺北醫學大學 大數據科技及管理研究所

hch@tmu.edu.tw

課程大綱

Python 預備備~

- 手把手開發環境jupyter notebook介紹、實用小秘訣

Python 起步走~

- 零基礎Python程式快速入門
- numpy, pandas

Python GO !

- Scikit-learn入門與資料預處理
- 使用有標記的數據集-監督式學習

Python FIGHT !

- Scikit-learn機器學習實作初探
- 使用未標記的數據-集群分析Clustering





Python 預備備~ 手把手開發環境介紹

Why Python?

手把手開發環境jupyter notebook介紹

實用小秘訣



Why Python

Why Python?

The classic "Hello, world!" program illustrates the relative verbosity of Java.

Java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello, world!");
    }
}
```

Python

```
print "Hello, world!"
```

```
print("Hello, world!") # Python version 3
```



Life is SHORT,



You Need PYTHON.

Python簡介

◎起源

- 發明人是Guido van Rossum
- 第一個公開發行的版本在1991/2/20
- 2000年10月16日發佈2.0版
- 2008年12月3日發佈3.0版

◎優點

- 好上手
- 擁有豐富的函式庫
- 社群完整



Python 強大社群及完整套件庫

科學計算

◎ NumPy (Numerical Python)

- 高階大量的維度陣列與矩陣運算。

◎ Pandas

- 基於 NumPy 再增加了 Series 和 DataFrame 兩種資料結構。

◎ SciPy

- 科學計算的工具。



Python 強大社群及完整套件庫

資料視覺化

◎ Matplotlib

- python 基礎的視覺的工具。

◎ Seaborn 及 ggplot

- 基於 matplotlib 的繪圖函式庫。



Python 強大社群及完整套件庫

機器學習

◎ Scikit-learn 。

○ 完整的處理資料處理、探勘、機器學習流程

- Preprocessing
- Dimensionality reduction
- Model selection
- Mining/Learning
- Experiment



Python的版本

◎Python 2和Python 3的差別

- 可參考網址：

<https://docs.python.org/3/whatsnew/3.0.html>

- Python 2最新版是2.7.x

- Python 3是以後發展的主軸，持續更新

◎主要差異

- Print

- 整數的除法

- Unicode的支持

...



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

馬上動手寫Python!!

- ◎大部份的Linux作業系統
- ◎MacOS作業系統
- ◎線上直接執行Python
 - python3
 - <https://repl.it/languages/python3>
 - python2
 - <https://repl.it/languages/python>



軟體安裝：ANACONDA

◎python 新手的好幫手!

- 包含了熱門眾多Python 套件
 - 科學、數學、工程、數據分析....
- 開源和免費
- 支持Linux、Windows、Mac
- 支持 Python 2、3
- 內帶spyder 編譯器、iunvter notebook 環境



安裝步驟

請依照自己電腦選擇下載版本(Python 3以上)

<https://www.continuum.io/downloads>

DOWNLOAD ANACONDA NOW

Download for   



Download for Your Preferred Platform

 Windows |  macOS |  Linux

Anaconda 4.4.0 For Windows Graphical Installer

Python 3.6 version *
64-Bit (437 MB) ⓘ

↓ DOWNLOAD

Download 32-bit (362 MB)

Python 2.7 version *
64-Bit (430 MB) ⓘ

↓ DOWNLOAD

Download 32-bit (354 MB)

Behind a firewall?

* How to get Python 3.5 or other Python versions

How to Install ANACONDA

安裝步驟

◎說明：

- Window 安裝過程如果有一頁"Advanced Options"
兩個都要打勾
 - ☒ Add Anaconda to my PATH environment variable
 - ☒ Register Anaconda as my default Python 3.5



安裝步驟

◎說明：

○安裝擴充套件方法

● windows

- 進入命令提示字元cmd透過conda安裝套件
- 如：conda install matplotlib

● mac

- 進入終端機terminal(mac)透過pip3指令安裝套件
- 如：pip3 install matplotlib



Jupyter Notebook

◎Windows

- windows鍵+R→叫出執行
- 輸入cmd→打開命令提示字元
- C:\Users\使用者名稱>
 - 輸入 jupyter notebook
 - 執行之後會打開預設瀏覽器
 - 對應的資料夾即為C:\Users\使用者名稱

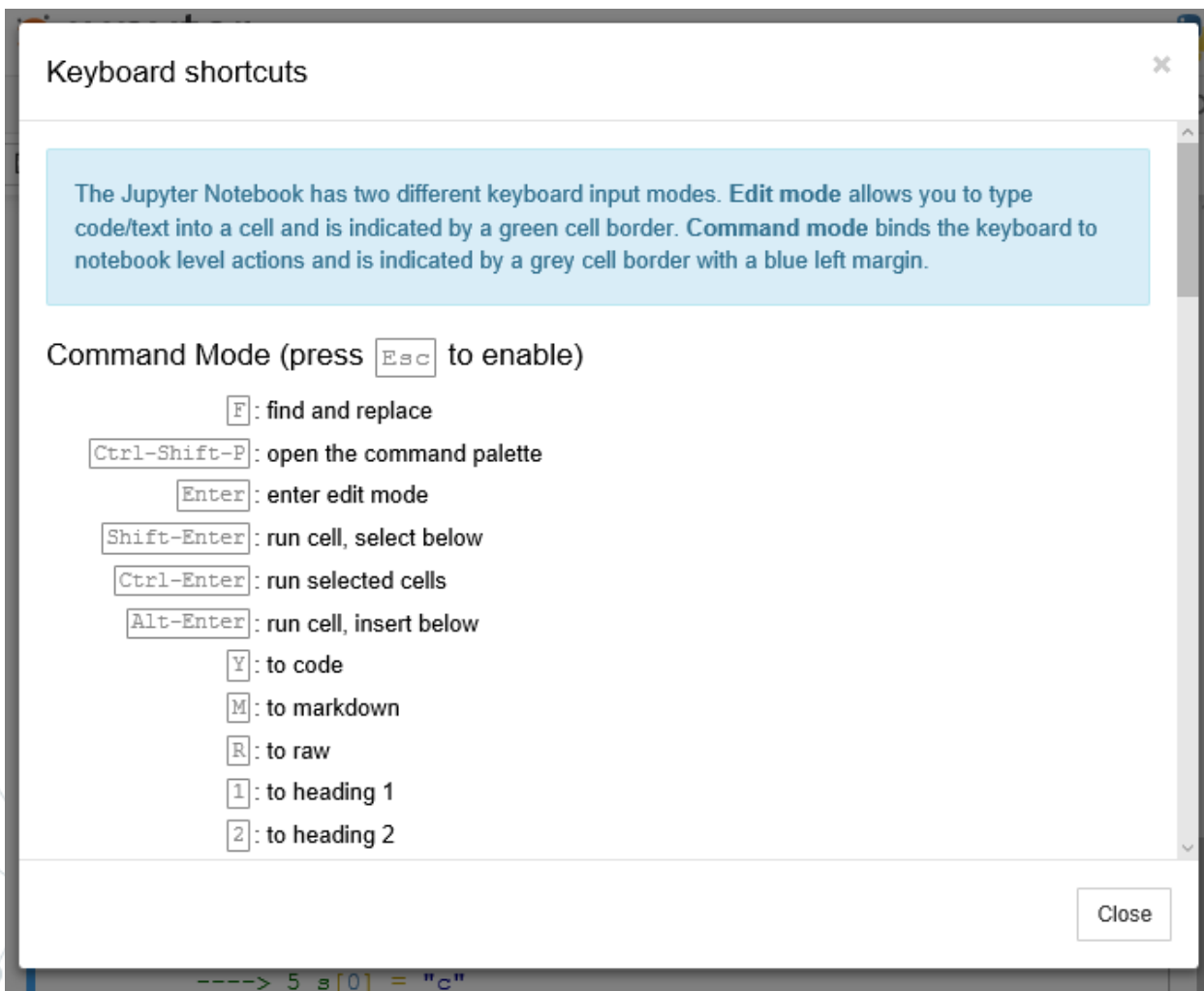
◎Mac

- 打開terminal
- 輸入jupyter notebook



Jupyter Notebook

◎ H鍵會跳出快捷鍵說明



The screenshot shows the 'Keyboard shortcuts' dialog box in Jupyter Notebook. It has a title bar with a close button. Inside, there is a light blue informational box at the top explaining the two keyboard input modes: Edit mode (green border) and Command mode (grey border with blue left margin). Below this, the text 'Command Mode (press `Esc` to enable)' is displayed. A list of keyboard shortcuts follows, each with the key combination in a box and its function. At the bottom right is a 'Close' button. The background of the slide shows a network diagram with nodes and connecting lines.

Keyboard shortcuts

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code/text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level actions and is indicated by a grey cell border with a blue left margin.

Command Mode (press `Esc` to enable)

- `F`: find and replace
- `Ctrl-Shift-P`: open the command palette
- `Enter`: enter edit mode
- `Shift-Enter`: run cell, select below
- `Ctrl-Enter`: run selected cells
- `Alt-Enter`: run cell, insert below
- `Y`: to code
- `M`: to markdown
- `R`: to raw
- `1`: to heading 1
- `2`: to heading 2

Close



Jupyter Notebook

- ◎ 注意檔案儲存位置與格式
- ◎ 快速作筆記大絕招
- ◎ 執行程式碼 & 快捷鍵
- ◎ 結束
 - Save & Quit 以及命令提示字元 Ctrl + c





Python 起步走~

零基礎Python程式快速入門

帶著 Python 踏上資料科學之路
快速入門小教室

Python快速入門小教室-1

Ten ways to say "Hello World"

◎ #這是Python的HELLO WORLD

◎ print("Hello World! ")

◎ print('Hello World! ')

◎ #用單引號或雙引號來表示字串格式

◎ a = "Hello World!"

◎ print (a)



Python快速入門小教室-1

Ten ways to say "Hello World"

◎ #變數與字串

◎ a = "Hello"

◎ b = "World"

◎ print (a, b)

◎ print (a+b)

◎ #靈活的變數指定a,b = "Hello", "World"



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-2

What's your phone?

- ◎ #字串、變數命名方式、*input*
- ◎ M_phone = "ZenFone 3"
- ◎ U_phone = input("請輸入你的手機品牌")
- ◎ print("My phone is " + M_phone + "\n" + "Your phone is " + U_phone)

- ◎ #不須先定義變數型態
- ◎ type(M_phone)



Python快速入門小教室-2

What's your phone?

◎ #變數命名方式

- 開頭不能是數字
- 大小寫字母有區別
- 可使用
 - ◎ 大小寫字母a-z A-Z
 - ◎ 數字0-9
 - ◎ 底線_



Python快速入門小教室-2

What's your phone?

◎ #變數命名方式

○ 不可與內建關鍵字(保留字)同名

- ◉ *True, False, class, finally, is, return, None, continue,*
- ◉ *for, while, if, as, elif, if, or, yield, with, not, and,*
- ◉ *lambda, try, def, from, nonlocal, del, global, assert, else,*
- ◉ *import, pass, break, except, in, raise....*

○ Python3 加入Unicode(中文也可！不建議)



Python快速入門小教室-3

Apple Pen

◎ #跳脫字元 \

◎ #如何print出「Jim's Apple Pen」

◎ c='Jim's Apple Pen'

◎ print (c)

◎ #試試看 \n 跟 \t



Python快速入門小教室-4

Not_a_List

◎ #字串?陣列?

◎ S = 'Jim\'s Apple Pen 吉姆的蘋果筆!'

◎ print (S)

◎ B = S[0:5]

◎ print (B)

◎ # len()

◎ # lower()

◎ # upper()



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-4

Not_a_List

◎ # 跳脫字元不占位置 空白占位置

◎ S = 'Jim\'s Apple Pen 吉姆的蘋果筆!'

◎ pos = 0

◎ for i in S:

◎ print (pos, i)

◎ pos += 1



Python快速入門小教室-5

Variable

- ◎ *#python 變數型態*
- ◎ 布林 (Boolean)
 - True / False
 - 1/0
- ◎ 整數 (Integer)
 - 2,4,6,8,10 ...
- ◎ 浮點數 (Float)
 - 3.14159 ...
- ◎ 字串 (String)
 - 'hello world' ...



Python快速入門小教室-5

Variable

◎ *# python* 不需要先指定變項型態

◎ `my_i = 7`

◎ `my_f = 1.23`

◎ `my_b = True`

◎ `my_s = "OK"`

◎ `print (my_i * my_f)`

◎ `print (my_i, my_f, sep='\n')`

◎ *#type()* 與變形



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-6

+ - * /

◎ # 算術運算式 + - * / % // **

◎ print(10+3)

◎ print(10-3)

◎ print(10*3)

◎ print(10/3)

◎ print(10%3)

◎ print(10**3)

◎ print(10//3)

◎ # 運算的優先順序以及注意事項

◎ 6*7**2//2



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-7

> = <

- ◎ #關係運算式 < <= > >= == !=
 - ◎ #Equal to (==) Not equal to (!=)
 - ◎ #Less than (<) Less than or equal to (<=)
 - ◎ #Greater than (>) Greater than or equal to (>=)
 - ◎ '#關係運算式會吐出True or False
-
- ◎ `a = 5 > 4`
 - ◎ `print (a)`
-
- ◎ `b = 5 == 4`
 - ◎ `print (b)`



Python快速入門小教室-8

TrueFalseTrue

- ◎ # 邏輯運算式
- ◎ # *and* 兩者皆真為真
- ◎ # *or* 兩者有一真為真
- ◎ # *not* 否定
- ◎ a = True and True
- ◎ b = True and False
- ◎ c = False and False
- ◎ d = True or True
- ◎ e = True or False
- ◎ f = False or False
- ◎ g = not False
- ◎ print (a, b, c, d, e, f, g)



Python快速入門小教室-9

while it's *True*

◎ #while condition:

◎ condition = 0

◎ while condition < 10:

◎ print(condition)

◎ condition = condition + 1



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-10

if you are *True*

◎ *# if, < = >, !=, ==*

◎ `x, y, z = 3, 9, 6`

◎ `if x < y > z:`

◎ `print("OK")`



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-10

if you are *True*

◎ *# if, elif and else*

◎ `a = -1`

◎ `if a > 3:`

◎ `print ("a > 3")`

◎ `elif a == 3:`

◎ `print ("a = 3")`

◎ `else:`

◎ `print ("a < 3")`



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-11

for you are *True*

◎ *# for item in sequence:*

◎ example_list =
[1,2,3,4,5,6,7,12,543,876,12,3,2,5]

◎ for i in example_list:

◎ print(i)

◎ for i in range(1, 10):
◎ print(i)



Python快速入門小教室-12

List

- ◎ numbers = [1,"2",3,4,"a","b",3.2]
- ◎ print ("數列長度: ",len(numbers))
- ◎ print (numbers[0])
- ◎ print (numbers[-1])
- ◎ print (numbers[2:5])
- ◎ print (numbers[3] + numbers[-1])
- ◎ print (2 in numbers)



Python快速入門小教室-12

List

- ◎ #其他應用
- ◎ numbers[3] = 100
- ◎ numbers.append(10)
- ◎ numbers.insert(2,100)
- ◎ numbers.remove(100)
- ◎ numbers[4:6]=[]
- ◎ ##只能移除第一個碰到的元素
- ◎ numbers.sort()
- ◎ len(numbers)



Python快速入門小教室-12

List

- ◎ *## 三倍長度*
- ◎ `print (3 * numbers)`

- ◎ *## 分別操作list中的每個元素*
- ◎ `for i in numbers:`
- ◎ `print (i)`
- ◎
- ◎ `for i in numbers:`
- ◎ `print (i * 3)`



Python快速入門小教室-13

List VS String

◎ `s = "string"`

◎ `s_list = list("string")`

◎ *#那一個可以?*

◎ `s[0] = "c"`

◎ `s_list[0] = "c"`

◎ `s_list[-2:] = "bc"`



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-13

List VS String

◎ *# String to List*

◎ `s = "I love pokemon"`

◎ `A_list = list(s)`

◎ `B_list = s.split()`

◎ `print(A_list)`

◎ `print(B_list)`



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

Python快速入門小教室-14

List in List

- ◎ `a = [1,2,3,4,5]` # 一行五列
- ◎ `multi_dim_a = [[1,2,3],`
- ◎ `[2,3,4],`
- ◎ `[3,4,5]]` # 三行三列
- ◎ `print(a[1])`
- ◎ `print(multi_dim_a[0][1])`
- ◎ `# 2`



Python快速入門小教室-15

tuple

- ◎ # t 是tuple
- ◎ t = (1,2,3)
- ◎ another_t = 1,2,3

- ◎ #l 是list
- ◎ l = [1,2,3]

- ◎ #有什麼不同?!?!



Python快速入門小教室-15

tuple

#練習

- ◎ `t = ("妙蛙種子","小火龍","傑尼龜","胖丁","皮皮")`
- ◎ `for i in range(len(t)):`
- ◎ `print (t[i])`



Python快速入門小教室-15

tuple

◎ `format_A = "Hello %s %s"`

◎ `values_B = ("AA","BB")`

◎ `print (format_A % values_B)`



Python快速入門小教室-15

tuple

◎ *#input使用*

◎ dataA = input("Hi: 請輸入資料 ")

◎ print (dataA)

◎ *#綜合input跟tuple*

◎ format_A = "你是 %s 歲的 %s"

◎ dataA = input("Hi: 請輸入你的年齡 ")

◎ dataB = input("Hi: 請輸入你的職業 ")

◎ values_B = (dataA, dataB)

◎ print (format_A % values_B)



Python快速入門小教室-16

dictionary

- ◎ *# Dictionary 就是key and value*
- ◎ Week = {"Monday" : "星期一", "Tuesday" : "星期二", "Wednesday" : "星期三"}
- ◎ print (Week["Monday"])
- ◎ print (Week["Tuesday"])
- ◎ print (Week["Wednesday"])

- ◎ *#建新增、修改、移除、確認資訊*
- ◎ Week["Thursday"]="星期四"
- ◎ Week["Friday"]="星期五"
- ◎ del Week["Friday"]
- ◎ "Friday" in Week



Python快速入門小教室-16

dictionary

◎ #從空白新增

◎ Week = {}

◎ Week["Monday"] = "星期一"

◎ Week["Tuesday"] = "星期二"

◎ Week["Wednesday"] = "星期三"



Python快速入門小教室-16

dictionary

- ◎ # 用for迴圈把dict內容print出來
- ◎ for i in Week:
- ◎ print (i, Week[i])



Python快速入門小教室-16

dictionary

- ◎ *#Dictionary 内部的value可以放的東西*
- ◎ `inventory = {`
- ◎ `'coin' : 500,`
- ◎ `'wallet' : ['student ID', 'key'],`
- ◎ `'backpack' : {'iphone':2,'NB':1, 'book':3, 'apple':2}`
- ◎ `}`

- ◎ `inventory["wallet"].append("earphone")`
- ◎ `inventory['wallet'].sort()`
- ◎ `inventory["backpack"]["book"]=5`
- ◎ `inventory["coin"]+=50`
- ◎ `print(inventory)`



Python快速入門小教室-16

dictionary

◎ *#清空dictionary 使用*

◎ `d.clear()`

◎ `d = {}`

◎ *#使用d2的內容去更新d相同的鍵值*

◎ `Week = {"Monday": "星期一", "Tuesday": "星期二",
"Wednesday": "星期三"}`

◎ `Week_new = {"Wednesday": "週三", "Thursday": "週四"}`

◎ `Week.update(Week_new)`



Python快速入門小教室-17

set

- ◎ 集合set和字典dict的差異
- ◎ 都是使用{}，但是設定的方法不同
 - Dict是以(key, value)的型式儲存
 - 而Set只有儲存值
 - Dict可以有重複的資料項(value)，但是Set每一個資料只能有一份



Python快速入門小教室-18

def the function

- ◎ *# def* 一個函式，將功能抽象成一個函數以方便其他模塊使用。
- ◎ *# Python*透過":"和縮排來管理語法
- ◎ *#*函數內沒有接受參數的情形
- ◎ **def spam():**
- ◎ eggs = 12
- ◎ **return** eggs, eggs*2, str(eggs)*2
- ◎ **spam()**



Python快速入門小教室-18

Define the function

◎ *#PYTHON*在設定參數時，不用指定參數型態

◎ `def super_print(x):`

◎ `for i in x:`

◎ `print (i)`

◎ `super_print("cool")`

◎ `#super_print(13+5)`

◎ `super_print([1,2,3,4,5])`



Python快速入門小教室-19

牛刀小試

◎班上的成績表

◎Tom: 100, Mary: 95, Gary: 88, Tim: 80, Eva: 95

◎實作一個方法 `check_dic(name)`

```
check_dic("Tom")
```

```
Tom's score is 100
```

```
check_dic("Nick")
```

```
He/She is not in this class
```


Python快速入門小教室-19

牛刀小試

◎ 提示：

```
my_dict = {'Tom' : 100, 'Mary' : 95, 'Gary' : 88, 'Tim' : 80, 'Eva' :
```

```
def check_dic(name):  
    for n in my_dict:
```

```
        ???
```

```
        ???
```

```
            break
```

```
    else:
```

```
        ???
```

```
check_dic("Tom")
```

Tom 的分數是 100 分

```
check_dic("Jim")
```

班上沒有這位同學喔

Python快速入門小教室-19

牛刀小試

- ◎ #程式碼
- ◎ my_dict = {'Tom' : 100, 'Mary' : 95, 'Gary' : 88, 'Tim' : 80, 'Eva' : 95}
- ◎ def check_dic(name):
- ◎ for n in my_dict:
- ◎ _____
- ◎ _____
- ◎ break
- ◎ else:
- ◎ _____





帶著 Python 踏上資料科學之路！_科學運算 numpy & pandas

科學運算當中兩個重要的模塊，
一個是 numpy, 一個是 pandas。

為什麼使用 numpy & pandas

◎ 運算速度快

- numpy 和 pandas 都是採用 C 語言編寫, pandas 又是基於 numpy, 是 numpy 的升級版本。

◎ 消耗資源少

- 採用的是矩陣運算，會比 python 自帶的字典或者列表快好多



A decorative network graph pattern in the top-left corner, featuring a complex web of interconnected nodes and edges. Some nodes are highlighted with blue circles, and some edges are solid blue lines, while others are gray.

numpy

A decorative network graph pattern in the bottom-right corner, similar to the one in the top-left, with a web of nodes and edges, some highlighted in blue.

Numpy array

- ◎是固定大小的
 - 不像 Python List 可以動態增減
- ◎Array 之中的每一個元素都必須是相同型態
- ◎NumPy 會讓你程式碼更有效率
 - NumPy 所有元件都需要是相同大小的，因此在記憶體有相同的 Size。更適合用於數學運算與資料較龐大時的運算。



Numpy

屬性

- ◎ `import numpy as np` *#使用numpy 採用np簡寫*
- ◎ `array = np.array([[1,2,3],[4,5,6]])` *#二維的array*
- ◎ `print(array)`
- ◎ `print('維度:',array.ndim)`
- ◎ `print('行列:',array.shape)`
- ◎ `print('個數:',array.size)`



Numpy array

- ◎ `import numpy as np`
- ◎ `a=np.array([3,6,9,12,15,18])`
- ◎ `b=np.arange(6)`
- ◎ `c = np.zeros(6)` # 數據全為0，6個
- ◎ `d = np.ones(6, dtype = np.int)` # 數據為1，6個，整數
- ◎ `e = np.random.random(6)` # 亂數，6個
- ◎ `f=b-a`

- ◎ `print(a,b,c,d,e,f, sep="\n")`
- ◎ `print(f<-9)`
- ◎ `print(f== -9)`



Numpy array

- ◎ `import numpy as np`
- ◎ `a=np.array([[3,6,9],[12,15,18]])`
- ◎ `b=np.arange(6).reshape(2,3)`
- ◎ `c = np.zeros((2,3))` # 數據全為0
- ◎ `d = np.ones((2,3), dtype = np.int)` # 數據為1，6個
- ◎ `e = np.random.random((2,3))` # 亂數，6個

- ◎ `print(a,b,c,d,e, sep="\n")`
- ◎ `print(np.sum(a))`
- ◎ `print(np.sum(a, axis=1))`



Numpy array

#加減乘除法 (*非矩陣乘法*)

- ◎ import numpy as np
- ◎ a=np.array([[2,4,6],[8,10,12]])
- ◎ b=np.arange(6).reshape((2,3))

- ◎ print(a)
- ◎ print(b)
- ◎ print(a-b)
- ◎ print(a*b)



Numpy

矩陣乘法

矩陣乘法

- ◎ `import numpy as np`
- ◎ `a=np.array([[2,4,6],[8,10,12]])`
- ◎ `b=np.arange(6).reshape((2,3))`
- ◎ `print(a)`
- ◎ `print(b.T) # transpose`
- ◎ `print(np.dot(a,b.T))`

			0	3
			1	4
			2	5
2	4	6		
8	10	12		



Numpy array 合併

- ◎ `import numpy as np`
- ◎ `A = np.array([1,1,1])`
- ◎ `B = np.array([2,2,2])`
- ◎
- ◎ `print(np.vstack((A,B))) #vertical`
- ◎ `print(np.hstack((A,B))) #horizontal`
- ◎



Numpy array 分割

- ◎ `import numpy as np`
- ◎ `a = np.arange(12).reshape((3,4))`
- ◎ `print(a)`
- ◎ `print(np.vsplit(a,3))`
- ◎ `print(np.hsplit(a,2))`



A decorative network graph pattern in the top-left corner, featuring a complex web of interconnected nodes and edges. Some nodes are highlighted with blue circles, and some edges are solid blue lines, while others are gray.

Pandas

A decorative network graph pattern in the bottom-right corner, similar to the one in the top-left, with a web of nodes and edges, some highlighted in blue.

Pandas 基本介紹

◎ Numpy 和 Pandas 有什麼不同？

- Numpy 是array形式的，沒有欄位名稱或標籤
- Pandas 基於Numpy構建的，有列表的標籤。
 - 主要兩個數據結構：Series和DataFrame。



Series

- ◎ `import pandas as pd`
- ◎ `import numpy as np`
- ◎ `s = pd.Series([1,"abc","6",np.nan,44,1])`
- ◎ `print(s)`
- ◎ *#如果是numpy的array呢?*



DataFrame

生成方式一：用array的矩陣(1)

◎ `df = pd.DataFrame(np.random.randn(7,3))`

◎ `print(df)`



Hui-Chun Hung

Python程式語言起步走~
使用 Python 來做機器學習初探

DataFrame

生成方式一：用array的矩陣(2)

- ◎ `eat = np.random.randint(10,size=(7,3))*5+50`
- ◎ `dates = pd.date_range('20170812',periods=7)`
- ◎ `df0 = pd.DataFrame(eat)`
- ◎ #加上欄位
- ◎ `df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])`
- ◎ `print(df1)`



DataFrame

#生成方式二・字典方式

- ◎ `df2 = pd.DataFrame({'小數': pd.Series(1,index=list(range(4)),dtype='float32'),
'整數': np.array([3] * 4,dtype='int32'),
'時間': pd.Timestamp('20170812'),
'類別資料': pd.Categorical(["test","train","test","train"]),`
- ◎ `})`
- ◎ #字典的key代表甚麼?
- ◎ #dtype指定資料格式



DataFrame

#DataFrame的屬性

- ◎ `print(df2)`
- ◎ `print(df2.dtypes)`
- ◎ `print(df2.index)` #`print(df2.columns)`, `print(df2.values)`
- ◎ `print(df2.describe())`

- ◎ `print(df2.T)`
- ◎ `print(df2.sort_index(axis=1, ascending=False))`
- ◎ `print(df2.sort_values(by='類別資料'))`



Pandas 選擇數據

- ◎ `print(df1)`
- ◎ `print(df1['午餐'])`
- ◎ `print(df1[0:3])`



Pandas 選擇數據

- ◎ `# select by label`

- ◎ `print(df1.loc['20170812'])`

- ◎ `print(df1.loc[:,['早餐','晚餐']])`

- ◎ `#select by position`

- ◎ `print(df1.iloc[3,1])`

- ◎ `print(df1.iloc[3:5,1:3])`

- ◎ `#select by ix`

- ◎ `print(df1.ix[:3,['午餐','晚餐']])`



Pandas 選擇數據

◎ # 判斷

◎ `print(df1[df1.午餐 > 80])`



Pandas 輸入值

- ◎ `eat = np.random.randint(10,size=(7,3))*5+50`
- ◎ `dates = pd.date_range('20170812',periods=7)`
- ◎ `df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])`

- ◎ `df1.iloc[2,2] = 95`
- ◎ `df1.loc['20170818','晚餐'] = 60`
- ◎ `df1.晚餐[df1.早餐>80] = 40`
- ◎ `df1.loc['20170814','午餐'] = np.nan`

- ◎ `print(df1)`

處理遺漏值na

- ◎ #判斷是否有缺失數據 NaN, 為 True 表示缺失數據:
- ◎ `df1.isnull()`
- ◎ `df1.isnull().sum()`



處理遺漏值na

- ◎ #處理方法一：是將 NaN 的值用其他值代替, 比如代替成 0
- ◎ `df1.fillna(value=0)`



處理遺漏值na

- ◎ #處理方法二：將 NaN 值刪除
- ◎ df1.dropna(
 - ◎ axis=0, # 0: 對資料列進行操作; 1: 對該欄位進行操作
 - ◎ how='any' # 'any': 存在 NaN 就 drop 掉; 'all': 全是 NaN 才 drop
 - ◎)
- ◎ #其他用法
- ◎ df1.dropna(thresh=3)
- ◎ df1.dropna(subset=['晚餐'])



處理遺漏值na

◎ #處理方法三：差補法 (使用sklearn)

◎ from sklearn.preprocessing import Imputer

◎ imr = Imputer(missing_values='NaN', strategy='mean', axis=0)

◎ imr = imr.fit(df1)

◎ imputed_data = imr.transform(df1.values)

◎ imputed_data

◎ # strategy='median', 'most_frequent'



Pandas 檔案讀取

- ◎ `import pandas as pd` #加載模塊
- ◎ `#讀取csv`
- ◎ `data = pd.read_csv('XXX.csv')`
- ◎ `#打印出data`
- ◎ `print(data)`

- ◎ `#存成csv`
- ◎ `data.to_csv('student.csv')`
- ◎ `# 說明 http://pandas.pydata.org/pandas-docs/stable/io.html`



Pandas 合併dataframe

- ◎ `#concat`
- ◎ `import pandas as pd`
- ◎ `import numpy as np`

- ◎ `#定義資料集`
- ◎ `df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'])`
- ◎ `df2 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])`
- ◎ `df3 = pd.DataFrame(np.ones((3,4))*2, columns=['a','b','c','d'])`

- ◎ `#concat縱向合併`
- ◎ `res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)`



Pandas 合併dataframe

- ◎ `#concat`
- ◎ `#join, ['inner','outer']`
- ◎ `df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'], index=[1,2,3])`
- ◎ `df2 = pd.DataFrame(np.ones((3,4))*1, columns=['b','c','d','e'], index=[2,3,4])`
- ◎ `res = pd.concat([df1, df2], axis=0, join='outer', ignore_index=True)`
- ◎ `#append`
- ◎ `#res = df1.append(df2, ignore_index=True)`
- ◎ `#用append要注意甚麼呢??`



Pandas 合併dataframe

- ◎ # merge
- ◎ import pandas as pd
- ◎ left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
 'A': ['A0', 'A1', 'A2', 'A3'],
 'B': ['B0', 'B1', 'B2', 'B3']})
- ◎ right = pd.DataFrame({'key': ['K1', 'K2', 'K3', 'K4'],
 'C': ['C0', 'C1', 'C2', 'C3'],
 'D': ['D0', 'D1', 'D2', 'D3']})
- ◎ res = pd.merge(left, right, on='key')



Pandas 合併dataframe

- ◎ # merge
- ◎ import pandas as pd
- ◎ #定義資料集並打印出
- ◎ left = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
- ◎ 'key2': ['K0', 'K1', 'K0', 'K1'],
- ◎ 'A': ['A0', 'A1', 'A2', 'A3'], 'B': ['B0', 'B1', 'B2', 'B3']})
- ◎ right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
- ◎ 'key2': ['K0', 'K0', 'K0', 'K0'],
- ◎ 'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']})
- ◎ #依key1與key2 進行合併，並印出四種結果['left', 'right', 'outer', 'inner']
- ◎ res = pd.merge(left, right, on=['key1', 'key2'], how='inner')



pandas & matplotlib.pyplot

- ◎ `#Series`
- ◎ `import pandas as pd`
- ◎ `import numpy as np`
- ◎ `import matplotlib.pyplot as plt`
- ◎ `# 隨機生成100個數據`
- ◎ `data = pd.Series(np.random.randn(100),index=np.arange(100))`
- ◎
- ◎ `# pandas 數據可以直接觀看其可視化形式`
- ◎ `data.plot()`
- ◎ `plt.show()`



pandas & matplotlib.pyplot

- ◎ #Dataframe
- ◎ data = pd.DataFrame(
 - ◎ np.random.randn(10,4),
 - ◎ index=np.arange(10),
 - ◎ columns=list("ABCD"))
- ◎ # bar, hist, box, scatter, pie,.....
- ◎ data.plot()
- ◎ #ax = data.plot.scatter(x='A', y='B', label='class1', color='lightgreen')
- ◎ #data.plot.scatter(x='C', y='D', label='class1', color='darkred', ax=ax)
- ◎ plt.show()





Python GO !

Scikit-learn入門與資料預處理

甚麼是機器學習？

Scikit-learn入門

資料預處理



什麼是機器學習？

Giving Computers the Ability to Learn from
Data

從人的學習轉換到機器學習

◎人和電腦其實沒甚麼差別?!

◎人學習是為了習得一種技能，

○從觀察中累積經驗而學會辨認貓或狗

●觀察 -> 累積經驗、學習 -> 習得技能

◎機器學習呢？

○電腦可以從觀察資料及計算累積模型

●資料 -> 計算、學習出模型 -> 習得技能。

●重新組織已有的知識結構不斷改進自我。



機器學習

◎機器學習之應用已遍及各種領域

- 自然語言處理、影像辨識、推薦系統、人工智慧、深度學習.....等領域。

◎機器學習不只是機器學習

- 也是對自我意識與心靈等哲學問題的探索。
- 融合了統計學、神經科學、資訊理論、演算法等學科知識。

◎課堂討論：

機器學習的條件？

身邊常見的機器學習？



機器學習就在你身邊

◎深入日常生活應用

- 「AlphaGo」擊敗了圍棋冠軍
- 博客來的好書推薦
- 手機內的語音助理應用Siri
- Google 即時資訊卡片



機器學習方法

◎機器學習包含了哪些方法

○監督學習 supervised learning

- ◉有資料、有標籤

○非監督學習 unsupervised learning

- ◉有資料、沒有標籤

○半監督學習 semi-supervised learning

○強化學習 reinforcement learning

- ◉從經驗中總結提升

○遺傳算法 genetic algorithm

- ◉適者生存



範例：下雨預測

範例	溫度(度)	濕度(%)	是否下雨
1	18	70	是(1)
2	30	30	否(0)
3	20	60	是(1)

範例：房價

範例	面積(坪)	學區(1-10)	房價
1	50	9	20000K
2	50	7	15000K
3	30	7	8000K

概念定義

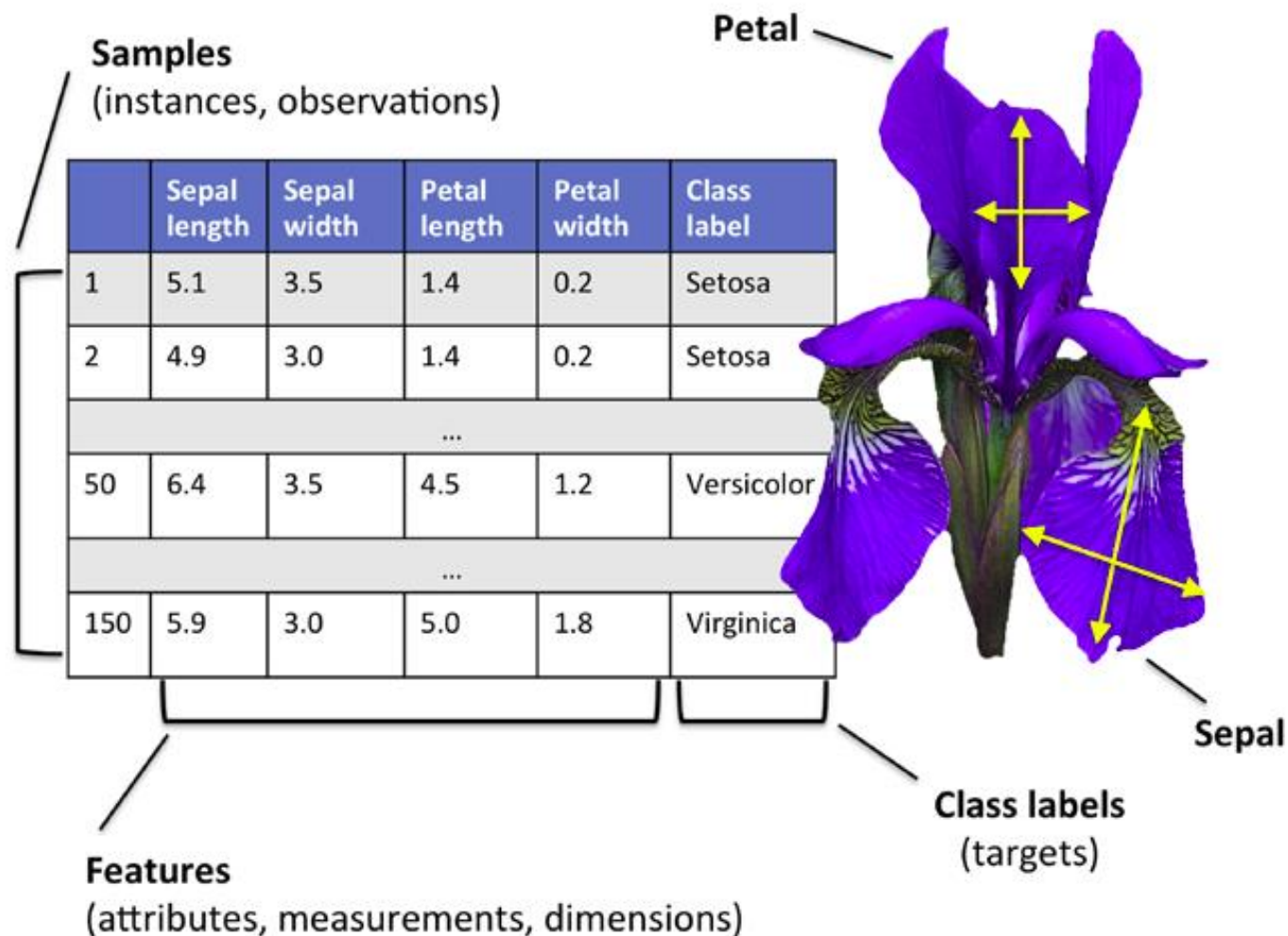
- ◎ 範例：
 - 濕度、溫度來預測是否下雨
- ◎ 在實際例子(instance)集合中，這個集合為 X 。
 - X ：所有日子的濕度、溫度
 - x ：每一個實例
- ◎ 待學習的概念或目標函數(target concept)為 c
 - $c(x)=1$ ，下雨
 - $c(x)=0$ ，沒下雨
 - $c(x)=y$
- 學習目標： $f: X \rightarrow Y$

相關名詞

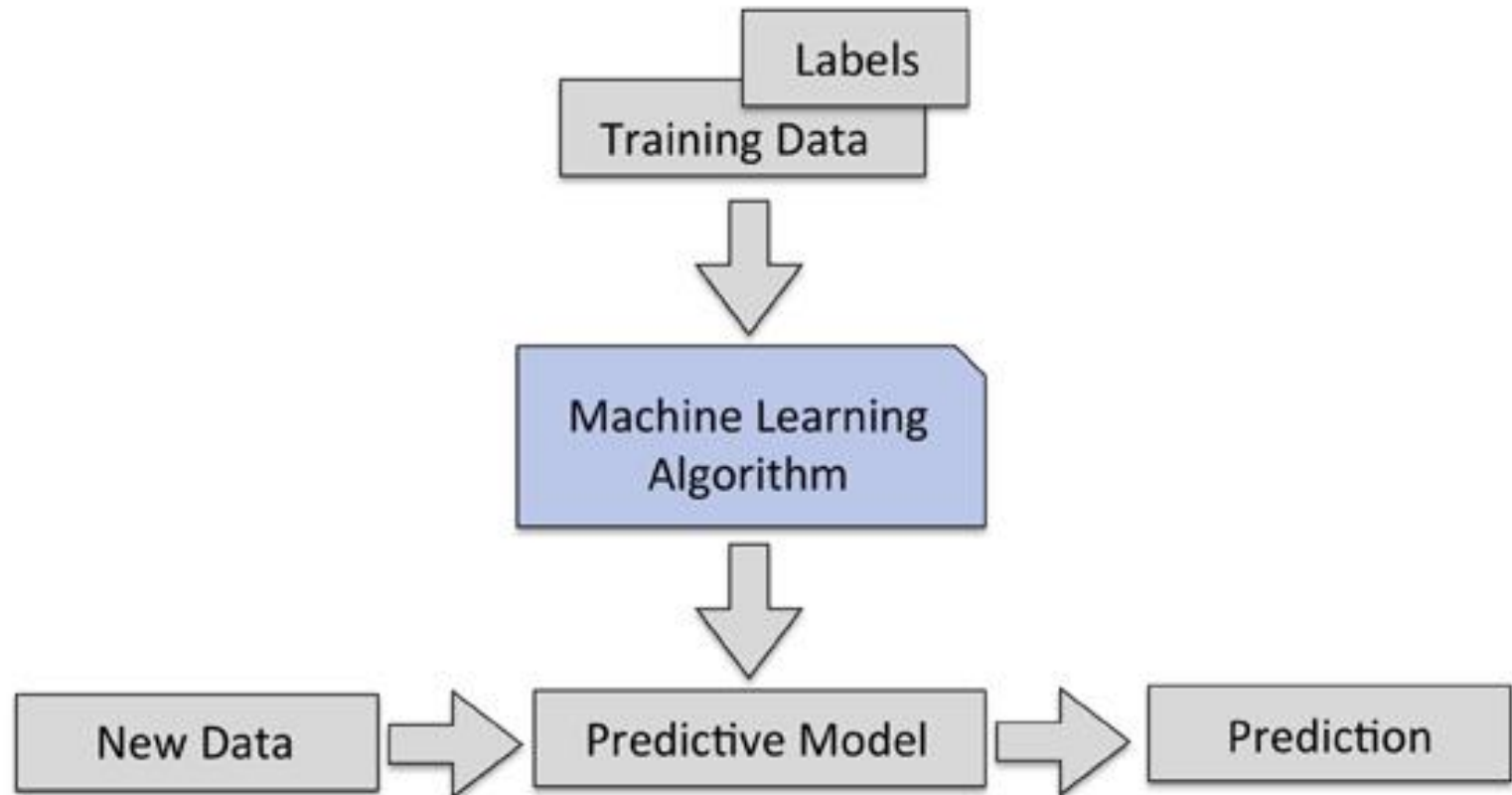
- ◎ 訓練集 training set/data
- ◎ 測試集 testing set/data
- ◎ 特徵向量 features / feature vector
- ◎ 標籤 label



相關名詞



Making predictions about the future with supervised learning



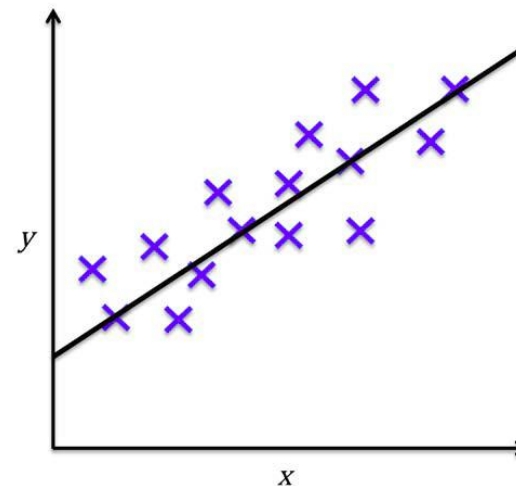
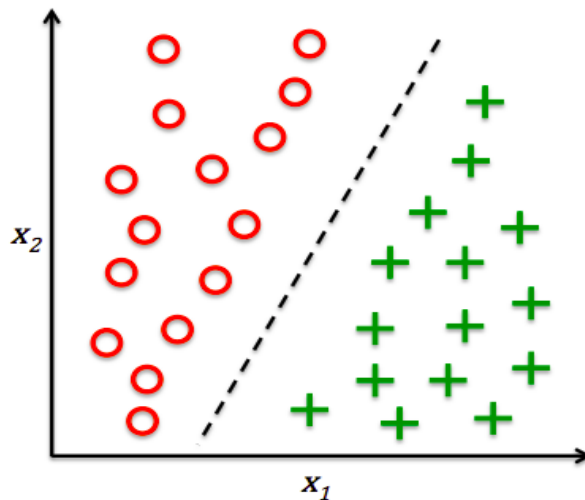
Making predictions about the future with supervised learning

◎分類：

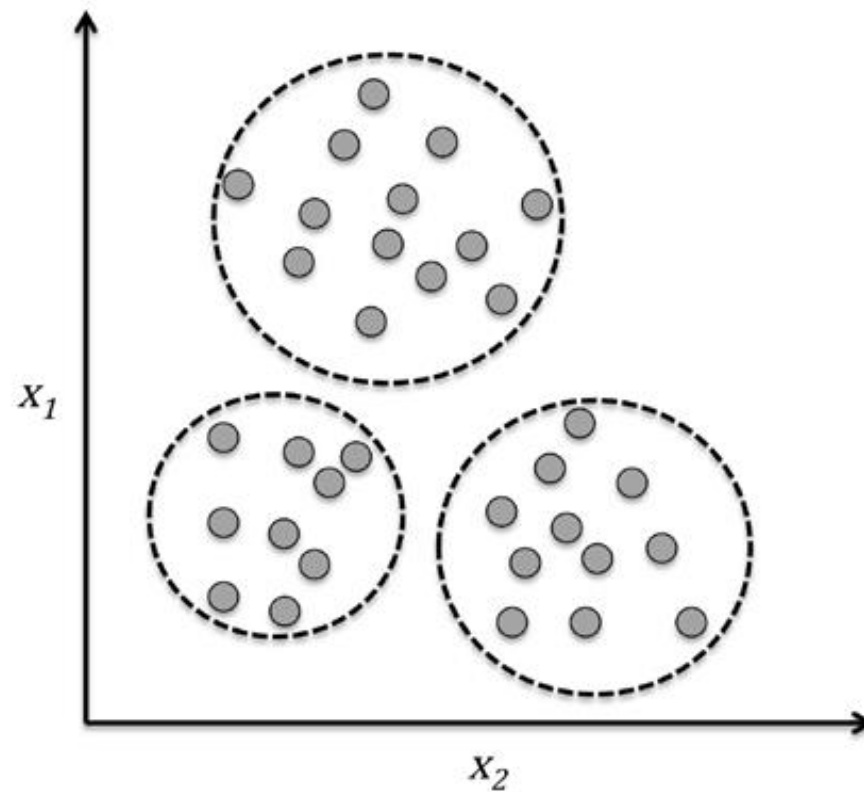
○目標標籤為類別category

◎迴歸：

○目標標籤為連續性數值 continuous numeric value



Discovering hidden structures with unsupervised learning



scikit-learn

第一次就上手GO!

使用有標記的數據集-監督式學習
分類方法實作



程式實作環境介紹

◎官方安裝 [教程](#)

◎Windows 使用 [Anaconda](#) 安裝

◎首先確認自己電腦中有安裝

○Python (≥ 3.3 版本)

○Numpy ($\geq 1.6.1$)

○Scipy (≥ 0.9)

查詢已安裝的package

- `pip list`



Mac/Windows 安裝方式

Scikit-learn requires:

- Python (≥ 2.6 or ≥ 3.3),
- NumPy ($\geq 1.6.1$),
- SciPy (≥ 0.9).

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using `pip`

```
pip install -U scikit-learn
```

Or `conda`:

```
conda install scikit-learn
```

Mac使用pip3

- `pip3 install scikit-learn`

Anaconda使用者

- `conda install scikit-learn`



Mac/Windows 安裝方式

Anaconda offers scikit-learn as part of its free distribution.

Warning: To upgrade or uninstall scikit-learn installed with Anaconda or `conda` you **should not use the pip command**. Instead:

To upgrade `scikit-learn`:

```
conda update scikit-learn
```

To uninstall `scikit-learn`:

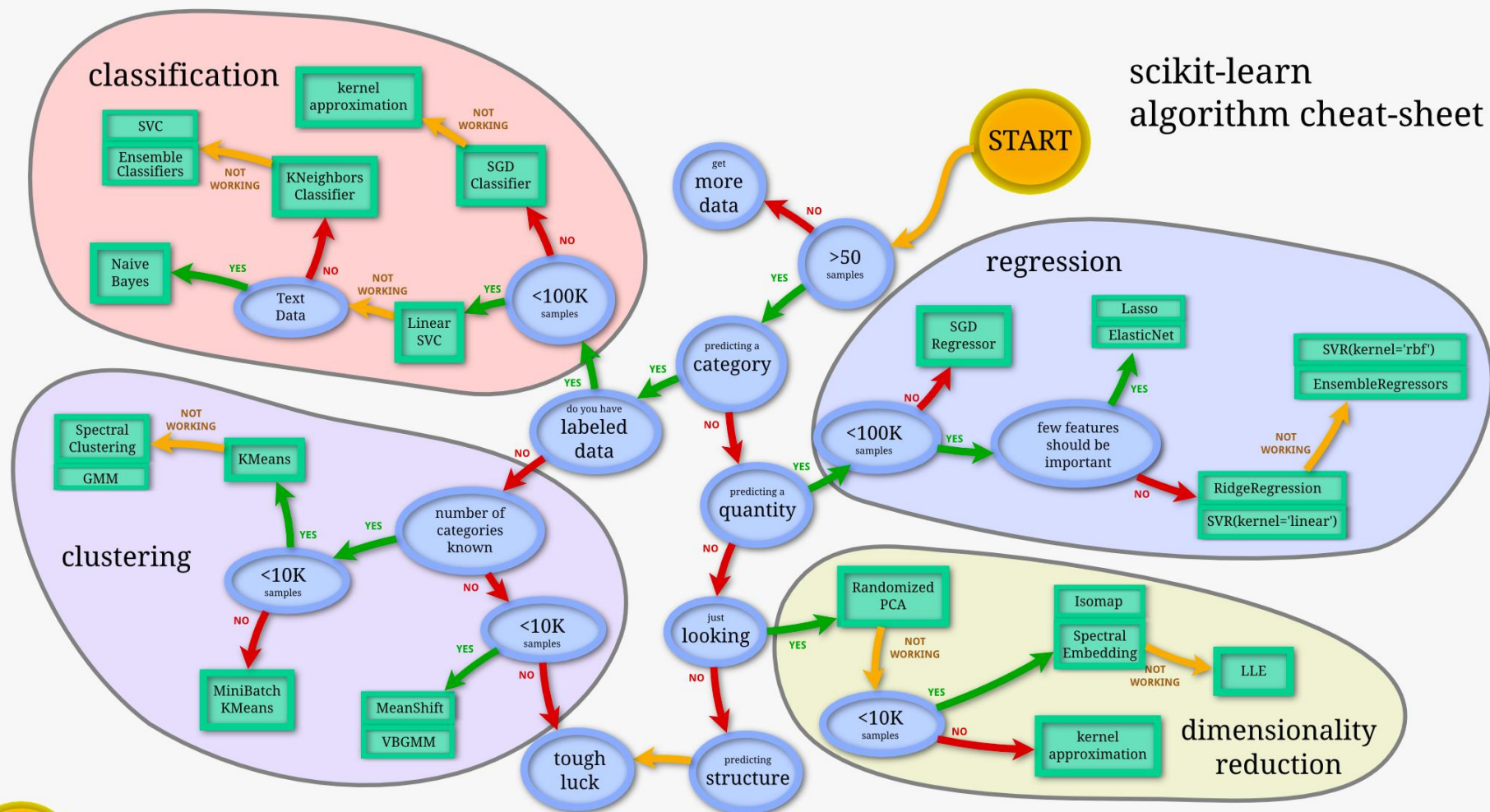
```
conda remove scikit-learn
```

Upgrading with `pip install -U scikit-learn` or uninstalling `pip uninstall scikit-learn` is likely fail to properly remove files installed by the `conda` command.

`pip` upgrade and uninstall operations only work on packages installed via `pip install`.



選擇學習方法



使用scikit-learn的流程

載入模組

- import 需使用的模組
- 規劃要使用之機器學習方法
- EX：datasets、train_test_split、KNeighborsClassifier....



創建數據

- 建立或匯入資料
- 資料整理、預處理、標準化
- 分屬性、標籤
- 把數據集分為【訓練集】和【測試集】



建立模型 - 訓練 - 預測

- 訓練fit
- 預測predict



視覺化

- 結果呈現
- 評估與修正



scikit-learn data sets 介紹

◎強大的數據集

- <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets>

◎有適用於各種情形的現成數據可以練習

- 房價、Iris...

◎也可以自己產生模擬數據集



分類器實作

◎常見的scikit-learn支援的分類器有非常多種類....

- `from sklearn.linear_model import Perceptron`
- `from sklearn.linear_model import LogisticRegression`
- `from sklearn.svm import SVC`
- `from sklearn.tree import DecisionTreeClassifier`
- `from sklearn.ensemble import RandomForestClassifier`
- `from sklearn.neighbors import KNeighborsClassifier`

.....



分類器實作 (KNN classifier)

◎使用iris資料集

- 四個屬性，花瓣長、花瓣寬，花萼長、花萼寬
- 根據這些屬性把花分為三類(標籤)。

◎使用模型的步驟：

○載入模組

- datasets、train_test_split、KNeighborsClassifier

○創建數據

- 屬性、標籤
- 把數據集分為訓練集和測試集

○建立模型 - 訓練 - 預測

- 訓練fit、預測predict

視覺化



分類器實作

- ◎ `import numpy as np`
- ◎ `from sklearn import datasets`
- ◎ `from sklearn.cross_validation import train_test_split`
- ◎ `from sklearn.neighbors import KNeighborsClassifier`
- ◎ `import matplotlib.pyplot as plt`

- ◎ `iris = datasets.load_iris()`
- ◎ `iris_X = iris.data`
- ◎ `iris_y = iris.target`

- ◎ `print(iris_X[:2, :])`
- ◎ `print(iris_y)`



分類器實作

- ◎ `X_train, X_test, y_train, y_test = train_test_split(iris_X, iris_y, test_size=0.3)`
- ◎ `#print(X_train)`

- ◎ `sc = StandardScaler()`
- ◎ `sc.fit(X_train)`
- ◎ `X_train_std = sc.transform(X_train)`
- ◎ `X_test_std = sc.transform(X_test)`
- ◎ `#print(X_test_std)`

分類器實作

- ◎ `knn = KNeighborsClassifier()`
- ◎ `knn.fit(X_train_std, y_train)`
- ◎ `y_predict = knn.predict(X_test_std)`

- ◎ `print(y_predict)`
- ◎ `print(y_test)`
- ◎ `print('Misclassified samples: %d' % (y_test != y_predict).sum())`



分類器實作

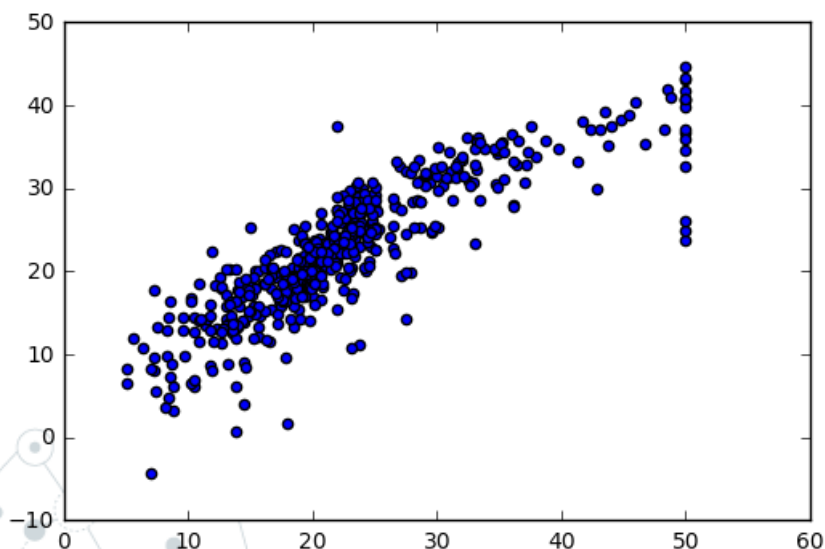
- ◎ `plt.scatter(y_predict,y_test,alpha=0.2)`
- ◎ `plt.show()`
- ◎ *#想一想：視覺化之後點都疊在一起要如何改善*
- ◎ `from sklearn.metrics import accuracy_score`
- ◎ `print('Accuracy: %.2f' % accuracy_score(y_test, y_predict))`



別忘了...

- ◎使用有標記的數據集的監督式學習
- ◎除了分類，還有回歸也可以做做看喔
- ◎可以試試看波士頓房價的datasets

http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html#sklearn.datasets.load_boston





Python FIGHT!

Scikit-learn機器學習實作

使用未標記的數據-集群分析Clustering

非監督式學習-集群分析

- ◎ 在不知道正確答案的情況下
 - 分析挖掘資料數據的隱藏結構。
- ◎ 集群的目的
 - 找到一種自然的分群，
 - 分群中樣本之間的相似度，比分群之外的樣本來高
- ◎ 群組內相似度高，群組間相異度高！



使用k-means來集群相似物件

◎物以類聚

- 在同群中的樣本比不在此群中的樣本，彼此更「相關」

◎實務上應用

- 使用者(顧客、學生)的分群
- 評論的分群
- 音樂的分群
- 電影的分群.....

◎K-means為原型基礎(prototype-based)的集群

- 每一個集群是由一個原型(中心點)來表示。



「K means」的用處

- ◎ 「K means」是一種聚類 (Cluster) 的方式
- ◎ 依照著「**物以類聚**」的方式在進行
 - 相似的東西有著相似的特徵
- ◎ 給予一組資料，將之分為**k類**
 - k由使用者設定



k-means屬「原型為基礎集群」

- ◎ k-means在確認數據的「球型集群」來說，是非常好的演算法！
- ◎ 缺點：必須先指定k
 - 要先設定要分幾群



k-means演算法4步驟：

1. 從樣本裡隨機挑選k個質心(centroid)
2. 指定每個樣本到他最近的質心
3. 移動質心到被分配的樣本點的中心
4. 重複2~3，直到群集分配不在改變，或是達到使用者定義的「可容許差誤」或「最大迭代次數」



隨機選取初始質心的k-means演算法

◎如果初始質心選取不良時，會導致

- 計算結果很差！
- 收斂過程太緩慢！

◎解決方法：

- 對數據集**多次**執行k-means演算法，選擇SSE表現最好的模型
- 使用**k-means++**演算法，選取初始質心時，盡可能彼此遠離=>更一致結果！
 - ◎scikit-learn實作，將Kmeans物件的init參數值，由random 改成 k-means++

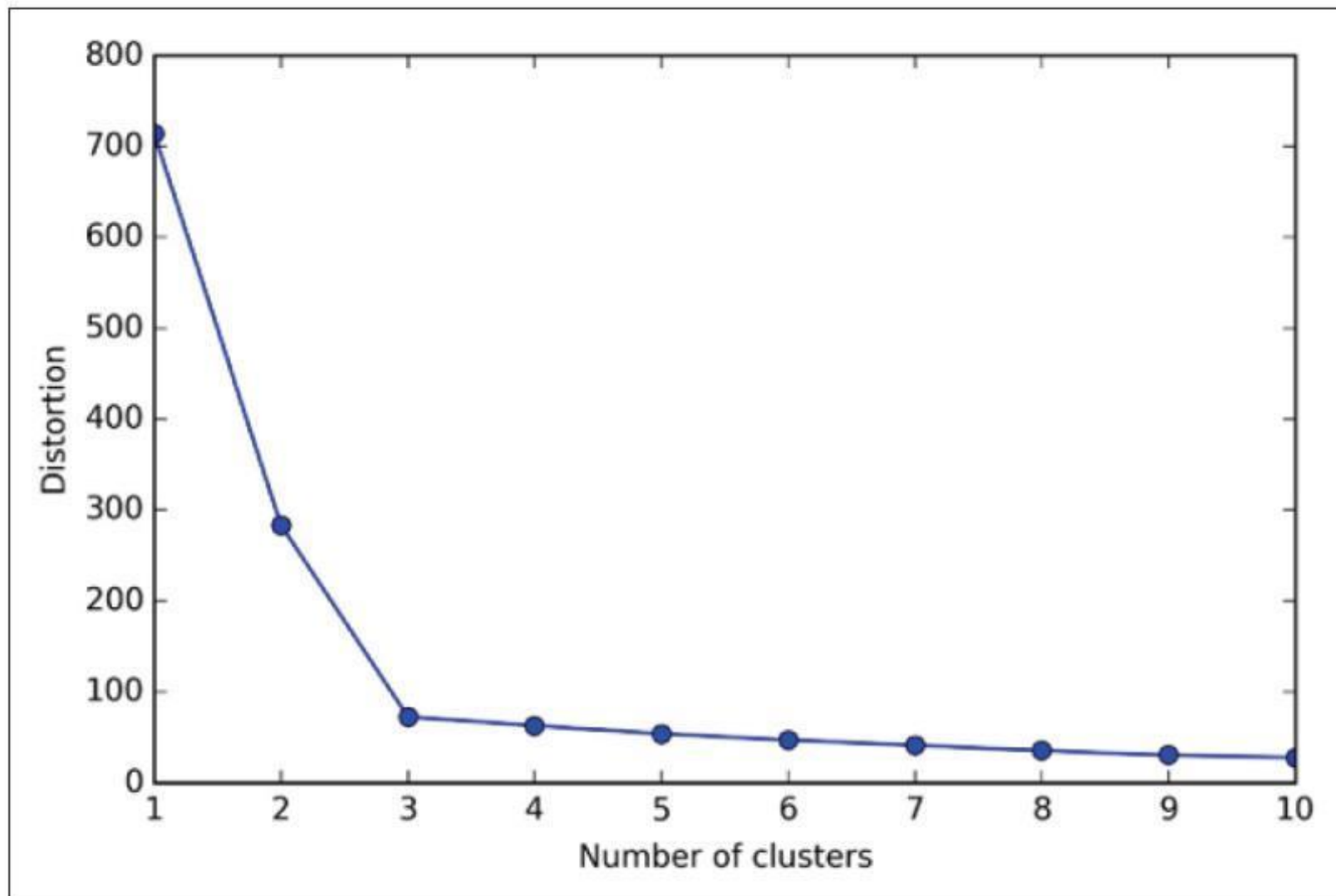


找出最佳的集群數目

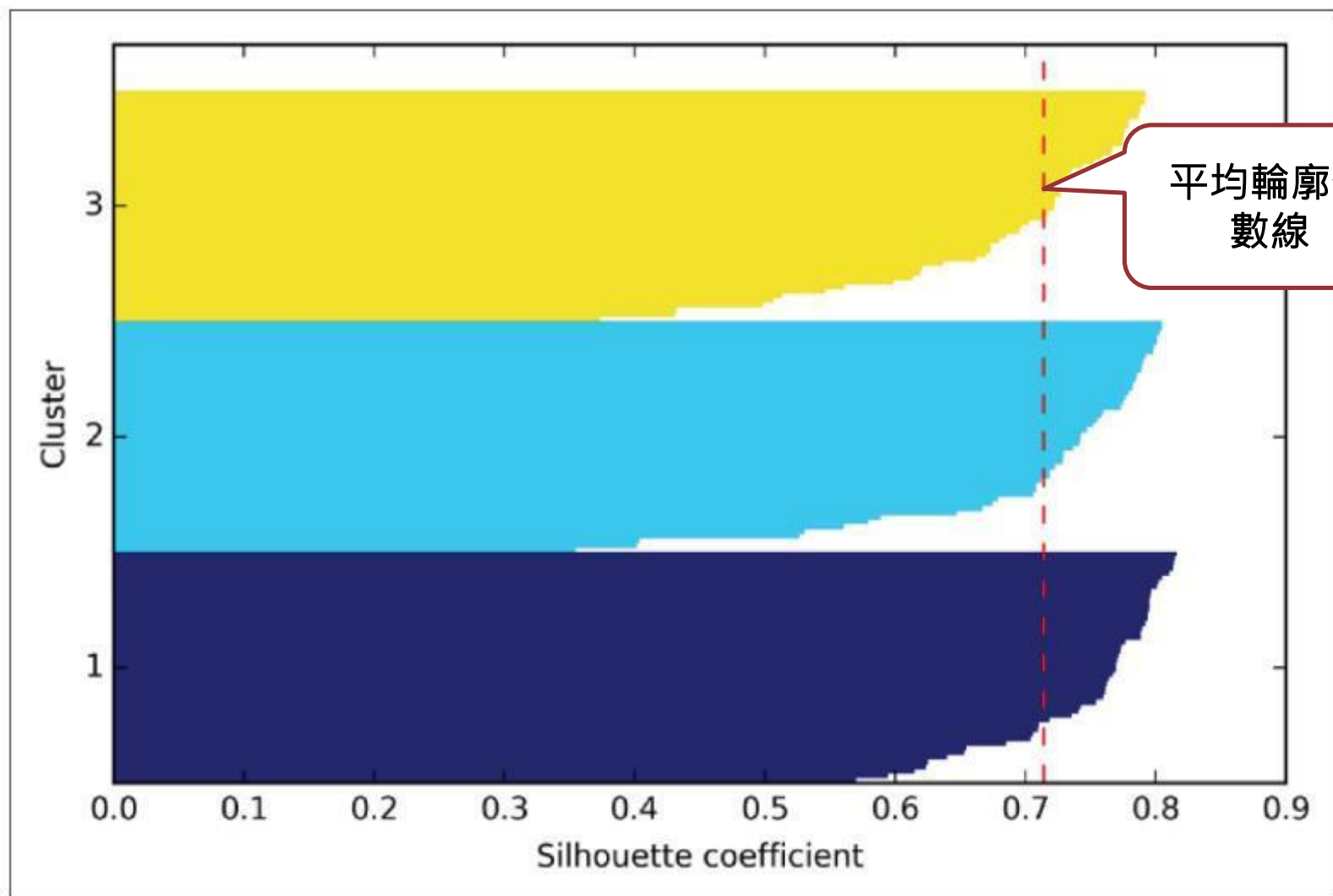
- ◎ 使用轉折判斷法找出最佳集群數
- ◎ 利用輪廓圖量化集群品質



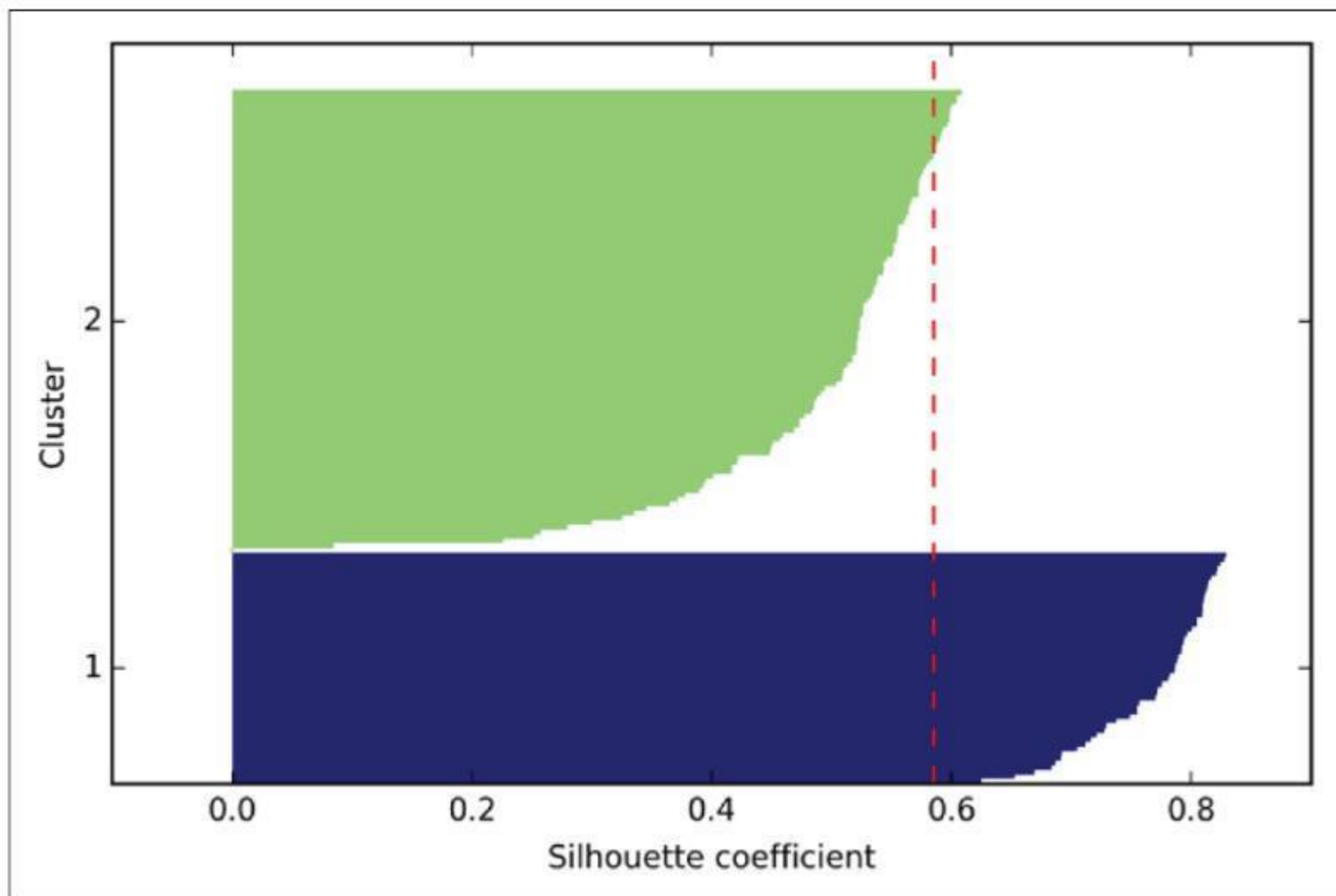
使用轉折判斷法找出最佳集群數



輪廓係數圖



2個集群「輪廓係數圖」長寬差異大，非理想集群方式

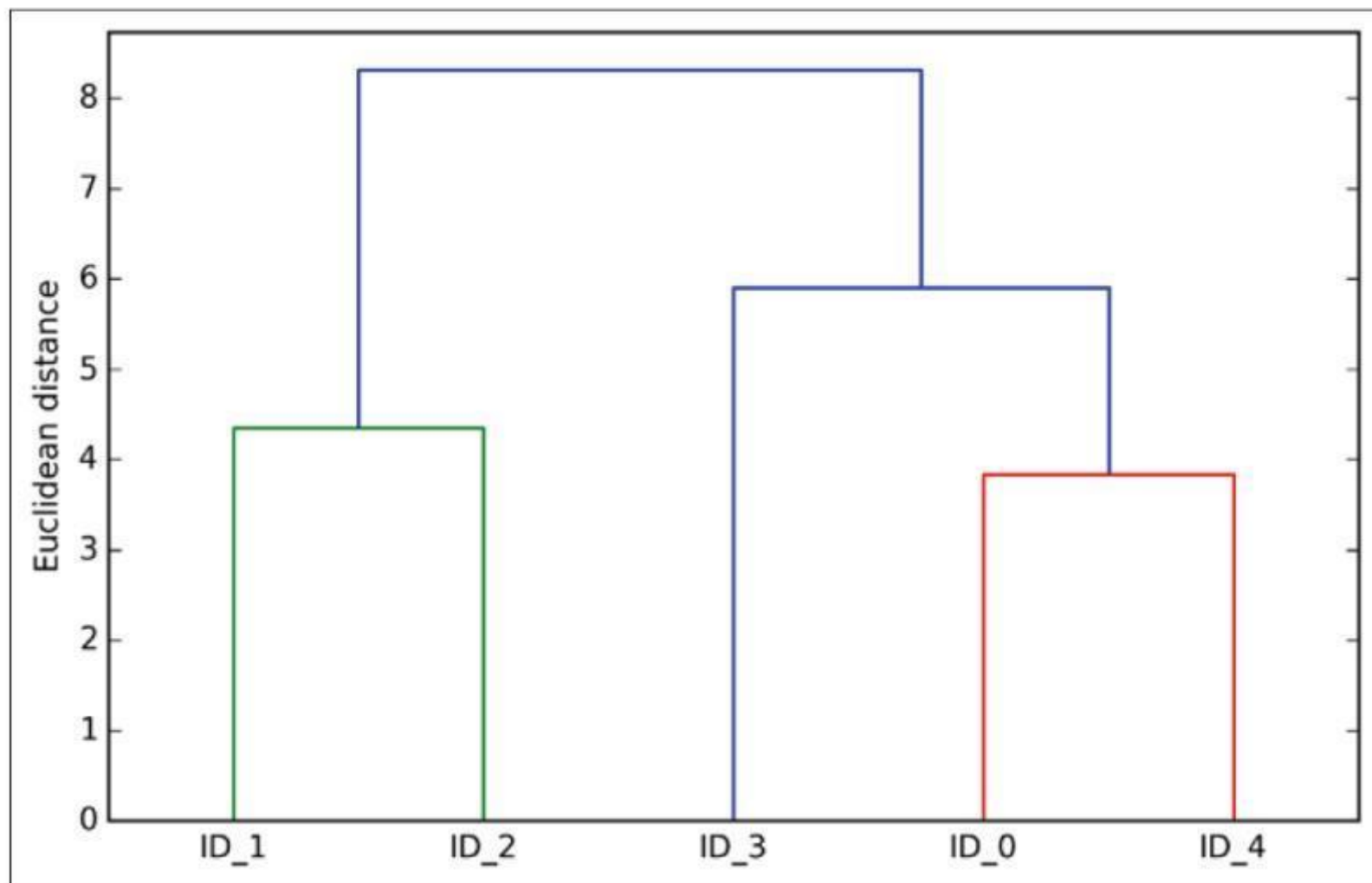


以階層樹方式組織集群 (hierarchical clustering)

- ◎ 「階層集群演算法」優點：
- ◎ 可繪製「樹狀圖」(dentrograms，一種視覺化的二階層分群工具)，用以建立有意義的分類，解釋結果。
- ◎ 不需要事先指定集群的個數。
- ◎ 新聞文類→運動類裡面的NBA、MLB



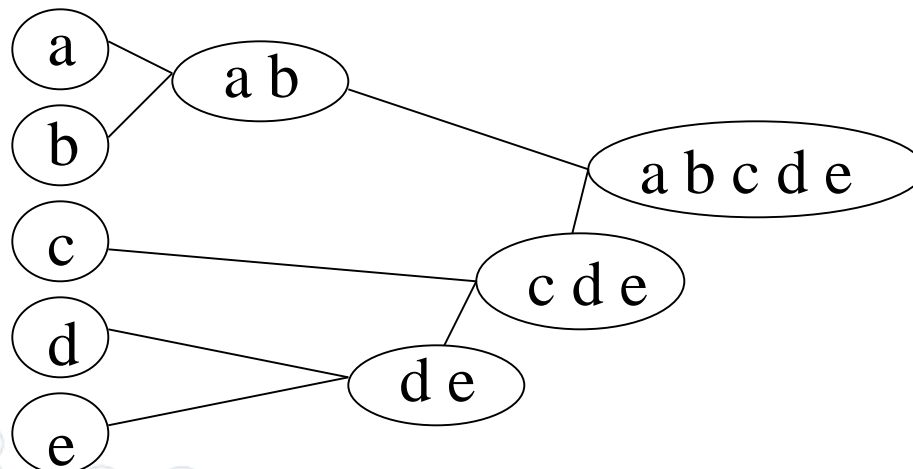
以「樹狀圖」，來呈現結果



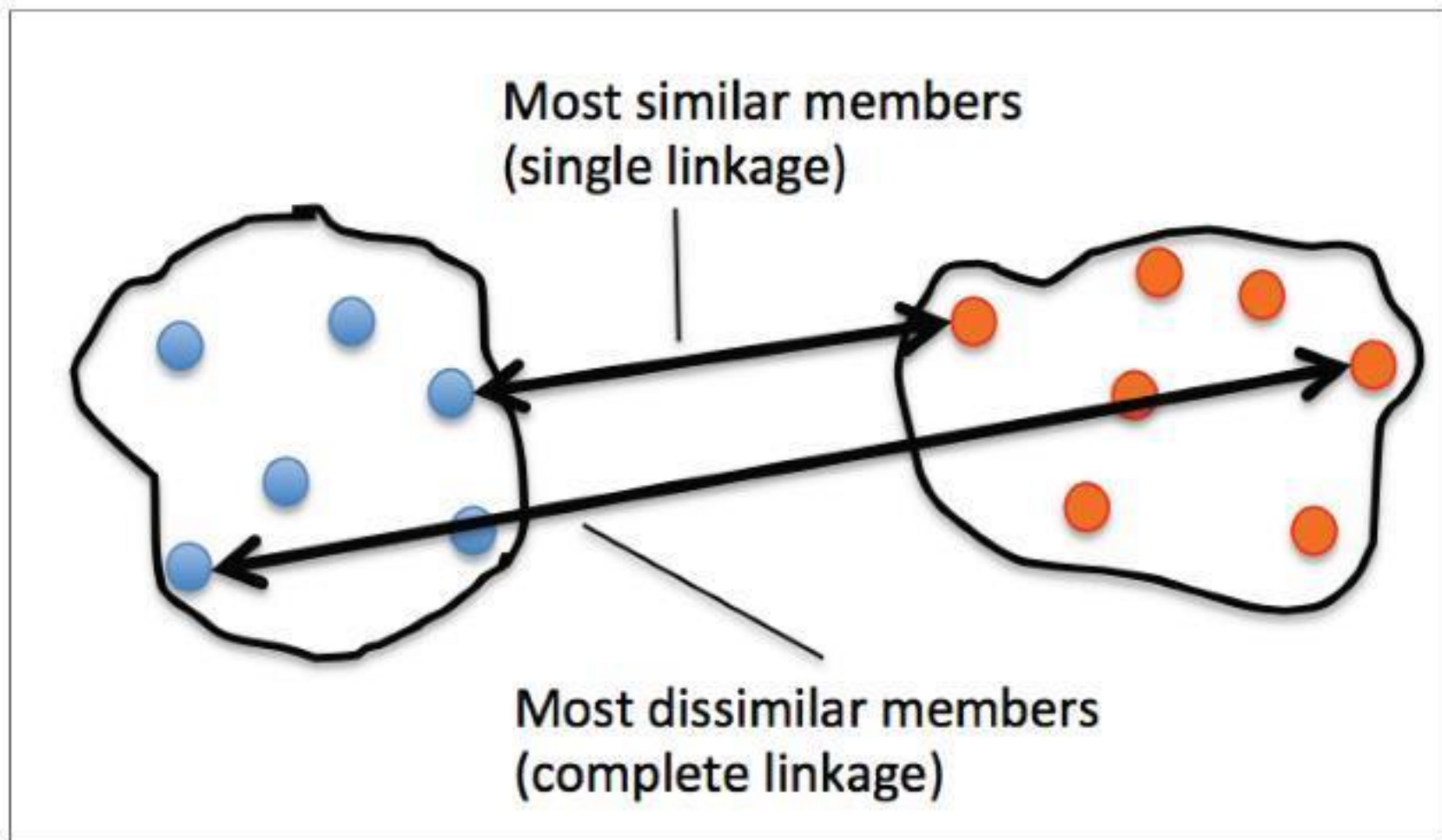
Hierarchical Clustering

- ◎ Use distance matrix as clustering criteria.
 - This method does **not** require the number of clusters k as an input, but needs a termination condition

Step 0 Step 1 Step 2 Step 3 Step 4 **agglomerative (AGNES)**

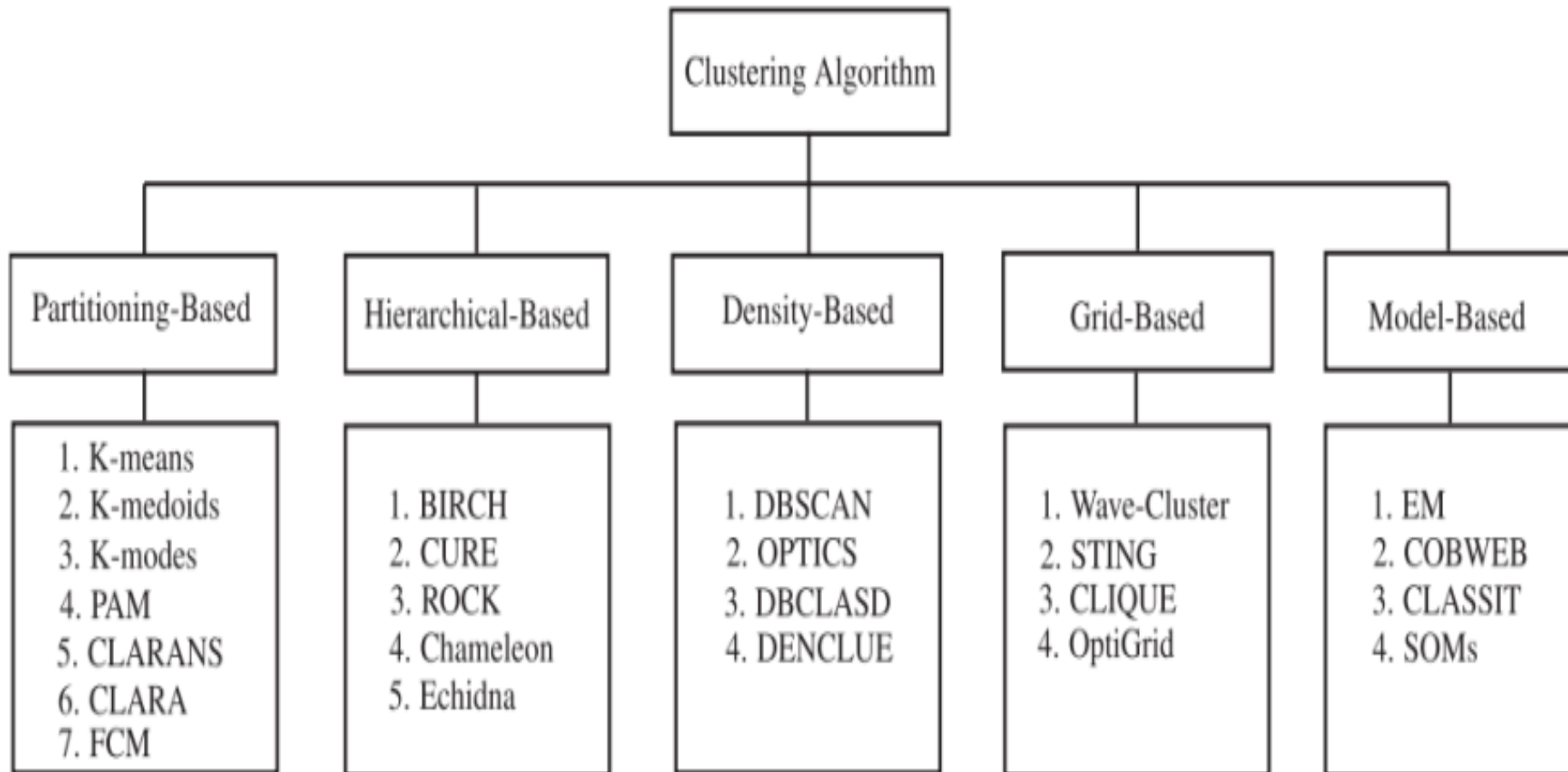


Step 4 Step 3 Step 2 Step 1 Step 0 **divisive (DIANA)**



其他相關分群方法

Taxonomy of Clustering





課堂實作練習

2016-17 NBA球員群集分析

2016-17 NBA球員群集分析

◎資料來源及介紹

- 資料來源為NBA官網之數據統計：stats.nba.com
- 共486名球員資料，資料內容包含：
 - ◎PLAYER：球員姓名
 - ◎Conference：所屬球隊為東/西區
 - ◎MIN：出場時間
 - ◎PTS：得分數
 - ◎FGM：投籃進球數
 - ◎FGA：投籃出手數
 - ◎FG%：投籃進球率
 - ◎3PM：三分進球數
 - ◎3PA：三分出手數
 - ◎3P%：三分進球率
 - ◎FTM：罰球進球數
 - ◎FTA：罰球出手數
 - ◎FT%：罰球進球率
 - ◎OREB：進攻籃板數
 - ◎DREB：防守籃板數
 - ◎REB：籃板數
 - ◎AST：助攻數
 - ◎TOV：失誤數
 - ◎STL：抄截數
 - ◎BLK：阻攻數
 - ◎PF：個人犯規數
 - ◎DD2：兩項數據達雙位數 (Double-double)，不含失誤、犯規
 - ◎TD3：三項數據達雙位數 (Triple-double)，不含失誤、犯規
 - ◎+/-：該球員之所有出場時間內，球隊之分差



參考資料

- ◎ <https://www.python.org/>
- ◎ <http://scikit-learn.org/stable/>
- ◎ Raschka, S. (2015). Python Machine Learning. Birmingham, UK: Packt Publishing.
- ◎ 何敏煌(2016)Python程式設計實務：從初學到活用Python開發技巧的16堂課，博碩文化。

