AI & ML

張家瑋 博士

助理教授

國立臺中科技大學資訊工程系

分類
CLASSIFICATION

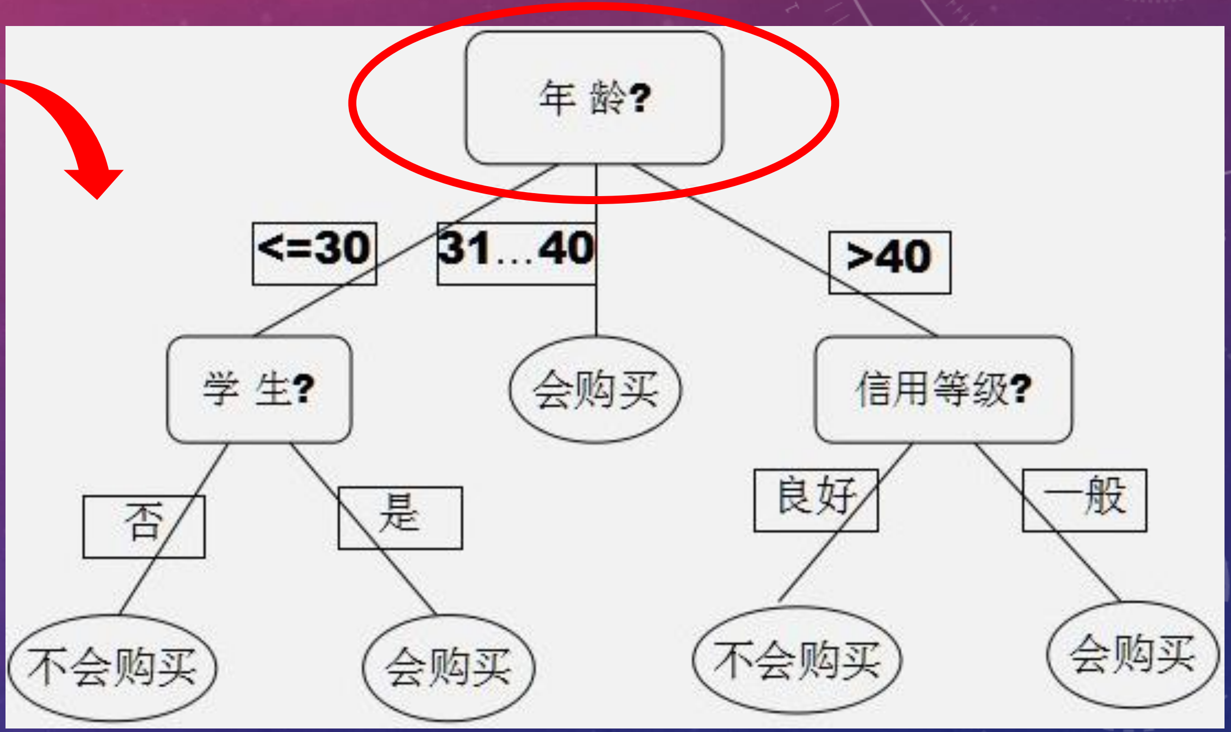
決策樹
DECISION TREE

概念

- 該決策樹方法先根據訓練集數據形成決策樹，如果該樹不能對所有對象給出正確的分類，那麼選擇一些例外加入到訓練集數據中，重複該過程一直到形成正確的決策集。

ID3 EXAMPLE

编号	年龄	收入	学生	信用等级	类别：购买电脑
1	<=30	高	否	一般	不会购买
2	<=30	高	否	良好	不会购买
3	31…40	高	否	一般	会购买
4	>40	中等	否	一般	会购买
5	>40	低	是	一般	会购买
6	>40	低	是	良好	不会购买
7	31…40	低	是	良好	会购买
8	<=30	中等	否	一般	不会购买
9	<=30	低	是	一般	会购买
10	>40	中等	是	一般	会购买
11	<=30	中等	是	良好	会购买
12	31…40	中等	否	良好	会购买
13	31…40	高	是	一般	会购买
14	>40	中等	否	良好	不会购买



三個年紀區間

9個買，5個不買

資訊熵 & 資訊量增益

$$H(p_1, \cdots, p_n) = -K \sum_{i=1}^{n} p_i \log p_i$$

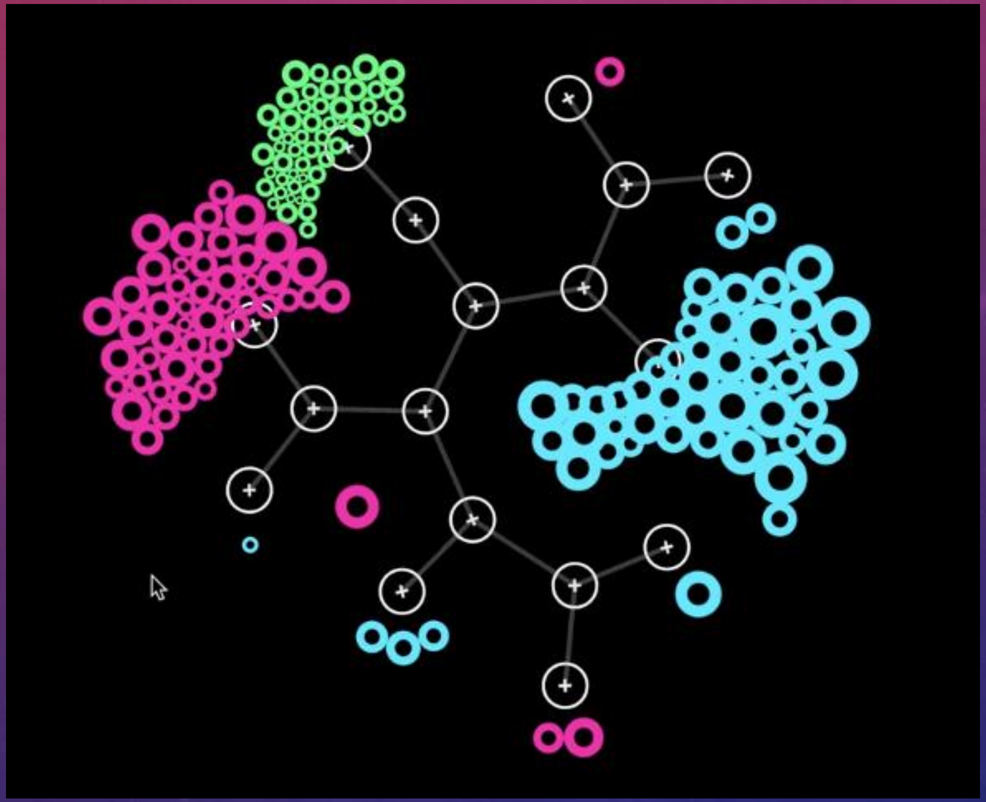
$$H(D) = -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14} = 0.94$$ 不買與買的資訊熵

$$H_{age}(D_{youth}) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.971$$ 年輕人、不買與買的資訊熵

$$H_{age}(D) = \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.694$$ 年紀、不買與買的總資訊熵

Gain(age) = 0.94 – 0.694 = 0.246
Gain(student) = 0.94 – 0.789 = 0.151
Gain(credit_rating) = 0.94 – 0.892 = 0.048
Gain(income) = 0.94 – 0.911 = 0.029

視覺化決策樹



https://vimeo.com/249969247

參考來源

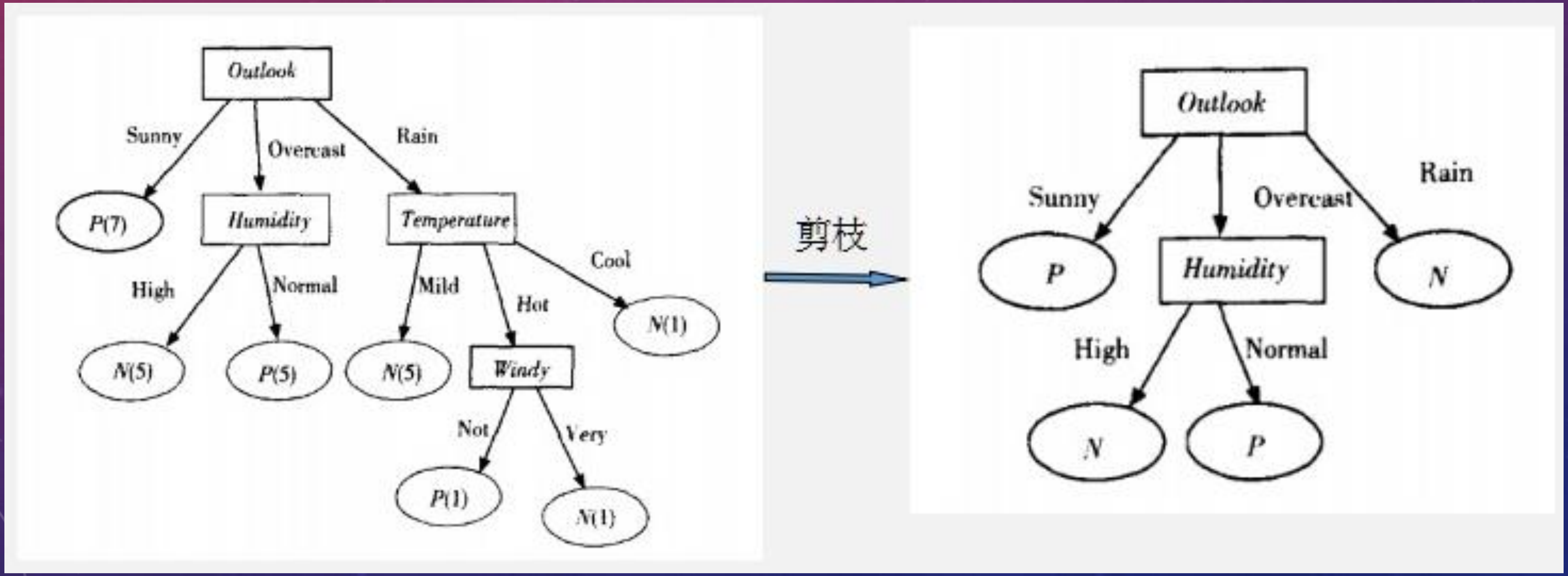
1. Visualizing a Decision Tree - Machine Learning Recipes #2
2. Decision Analysis 3: Decision Trees
3. C4.5決策樹算法

THINKING

重點？

重點

- Overfitting 過度擬合
- Pruning 剪枝



實作範例-Wine Dataset

[1.207e+01, 2.160e+00, 2.170e+00, 2.100e+01, 8.500e+01, 2.600e+00, 2.650e+00, 3.700e-01, 1.350e+00, 2.760e+00, 8.600e-01, 3.280e+00, 3.780e+02]

(1) Alcohol → 1.207e+01

(2) Malic acid → 2.160e+00

(3) Ash → 2.170e+00

(4) Alcalinity of ash → 2.100e+01

(5) Magnesium → 8.500e+01

(6) Total phenols → 2.600e+00

(7) Flavanoids → 2.650e+00

(8) Nonflavanoid phenols → 3.700e-01

(9) Proanthocyanins → 1.350e+00

(10)Color intensity → 2.760e+00

(11)Hue → 8.600e-01

(12)OD280/OD315 of diluted wines → 3.280e+00

(13)Proline → 3.780e+02

延伸閱讀

1. C4.5 決策樹 (GAINRATIO)
2. 隨機森林 RANDOM FOREST
3. K NEAREST NEIGHBOR (KNN)

LINEAR REGRESSION

數值型輸出

概念

某次實驗得到了四個數據點 (x,y)：$(1,6)$、$(2,5)$、$(3,7)$、$(4,10)$（右圖中紅色的點）。我們希望找出一條和這四個點最匹配的直線 $y=\beta_1+\beta_2 x$，即找出在某種「最佳情況」下能夠大致符合如下超定線性方程組的 β_1 和 β_2：

$$\begin{aligned}\beta_1+1\beta_2&=6\\ \beta_1+2\beta_2&=5\\ \beta_1+3\beta_2&=7\\ \beta_1+4\beta_2&=10\end{aligned}$$

最小平方法採用的手段是儘量使得等號兩邊的方差最小，也就是找出這個函數的最小值：

$$\begin{aligned}S(\beta_1,\beta_2)=&[6-(\beta_1+1\beta_2)]^2+[5-(\beta_1+2\beta_2)]^2\\ &+[7-(\beta_1+3\beta_2)]^2+[10-(\beta_1+4\beta_2)]^2.\end{aligned}$$

最小值可以通過對 $S(\beta_1,\beta_2)$ 分別求 β_1 和 β_2 的偏導數，然後使它們等於零得到。

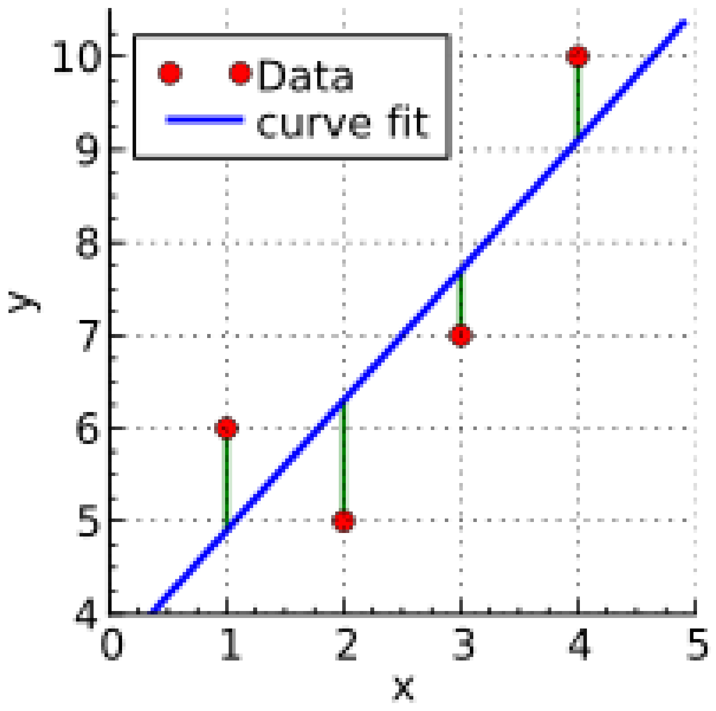
$$\frac{\partial S}{\partial\beta_1}=0=8\beta_1+20\beta_2-56$$

$$\frac{\partial S}{\partial\beta_2}=0=20\beta_1+60\beta_2-154.$$

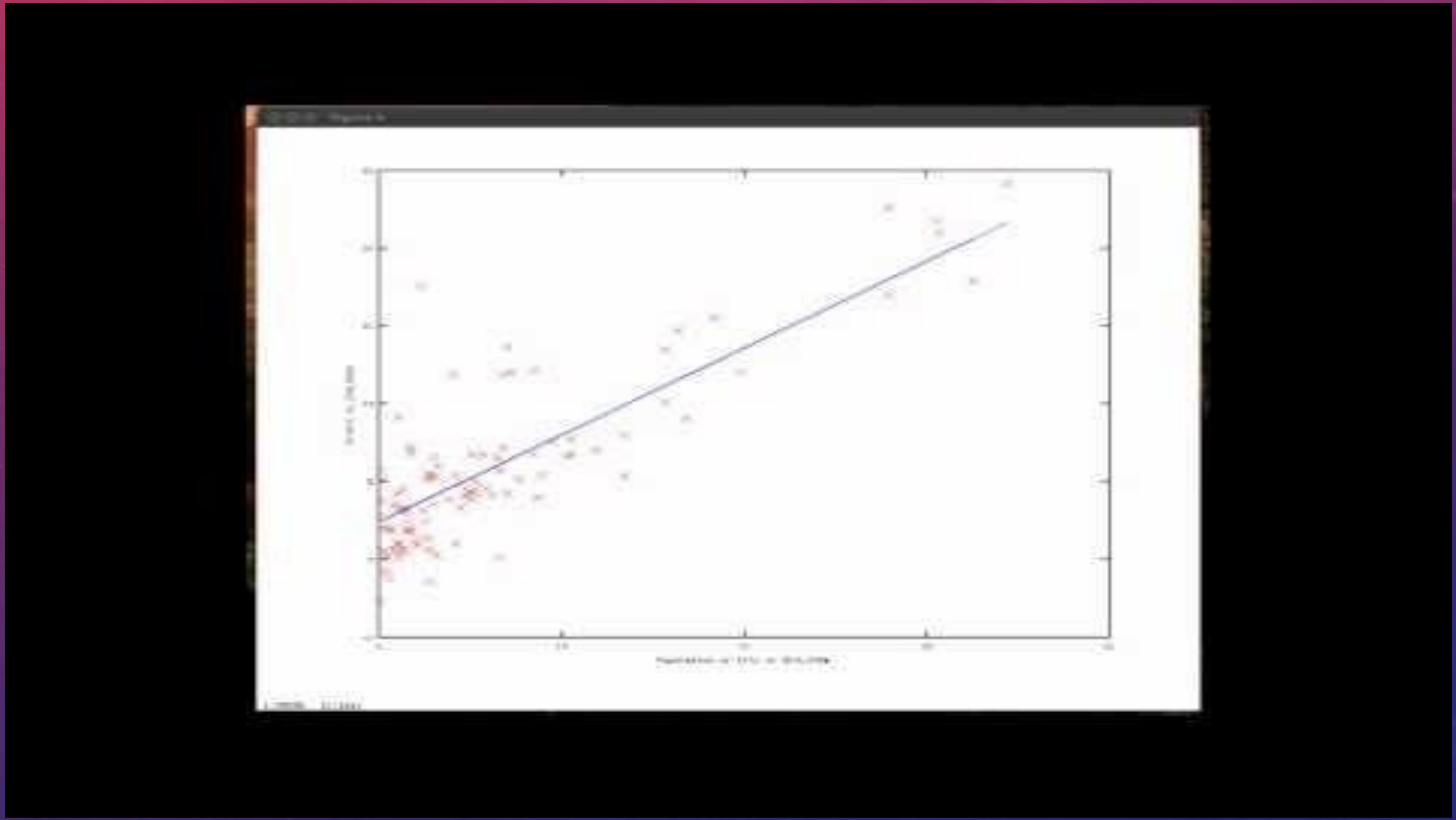
如此就得到了一個只有兩個未知數的方程組，很容易就可以解出：

$$\begin{aligned}\beta_1&=3.5\\ \beta_2&=1.4\end{aligned}$$

也就是說直線 $y=3.5+1.4x$ 是最佳的。

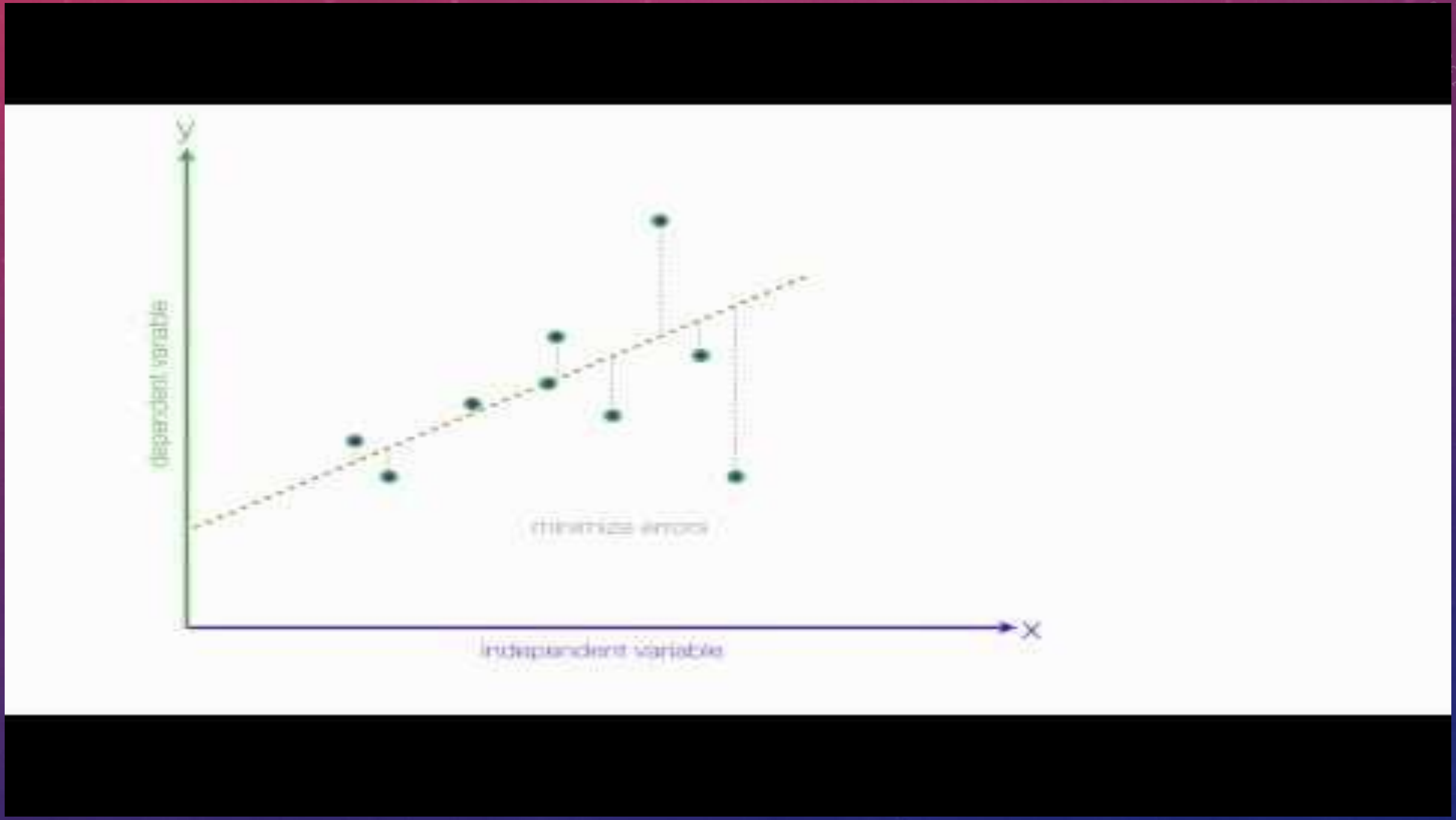


視覺化線性迴歸



https://www.youtube.com/watch?v=1hGsKphwC-A

EXAMPLE



https://youtu.be/zPG4NjIkCjc

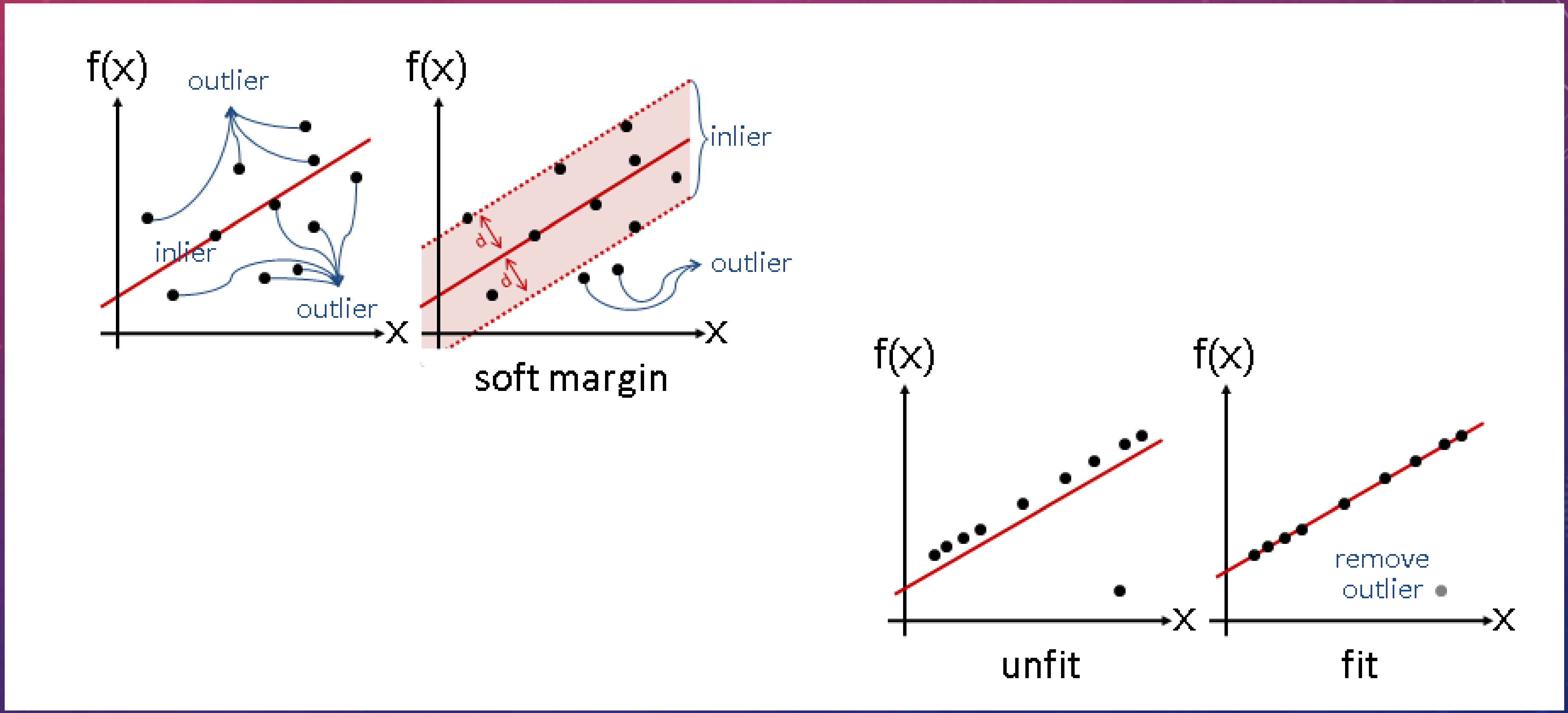
參考來源

1. An Introduction to Linear Regression Analysis
2. 最小平方法
3. http://www.csie.ntnu.edu.tw/~u91029/Regression.html

THINKING

重點？

重點


f(x)
outlier
inlier
outlier
x
f(x)
inlier
d
d
outlier
x
soft margin
f(x)
x
unfit
f(x)
remove
outlier
x
fit

延伸閱讀

1. LOGISTIC REGRESSION
2. SUPPORT VECTOR REGRESSION

實作參考

WEKA

1. Tutorial on K Means Clustering using Weka
2. algoritma c4 5 in weka
3. Linear Regression Example in Weka : Weka Tutorias # 4

PYTHON

1. 莫煩- sklearn常用屬性與功能 (Linear Regression 範例)
2. Scikit-Learn 教學：Python 與機器學習

分類器：KNN 與 決策樹
CLASSIFIER: KNN AND DECISION TREE

實作篇

使用SKLEARN及SKLEARN資料集實做

KNN的曼哈頓、歐幾里得距離及決策樹分類器

題目敘述

1. 使用SKLEARN中的預設的WINE資料集進行作業
2. WINE資料集中美筆資料都含有13種特徵
3. 使用KNN的曼哈頓、歐幾里得及決策樹分類器將13種特徵進行演算並且分類

事前要件：安裝SKLEARN模組

PIP3 INSTALL -U SCIKIT-LEARN

載入SKLEARN預設資料集

 # ---導入模塊---

```
FROM SKLEARN IMPORT DATASETS
FROM SKLEARN.CROSS_VALIDATION IMPORT TRAIN_TEST_SPLIT
IMPORT PANDAS AS PD
```

 # ---資料處理---

```
WINE = DATASETS.LOAD_WINE()
PRINT(WINE)
```

 # 載入SKLEARN內建資料集

PRINT(WINE) *#將資料集內容打印出來*

```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

← data為酒的特徵

PRINT(WINE) *#將資料集內容打印出來*

```
'target': 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, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2])
```

← target為上頁各項特徵
所對應到的酒種類
類別分為0,1,2三種標籤

```
WINE_DATA =  WINE.DATA
# 定義資料特徵
WINE_TARGET = WINE.TARGET
# 定義資料標籤
# PRINT(PD.DATAFRAME(WINE.DATA))
# 印出資料特徵查看
# PRINT(PD.DATAFRAME(WINE.TARGET))
# 印出資料標籤查看
X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = TRAIN_TEST_SPLIT(WINE_DATA, WINE_TARGET, TEST_SIZE = 0.2)
# 使用"TRAIN_TEST_SPIT"將數據分成訓練和測試兩類,TEST_SIZE = 0.2, 代表測試數據佔20%
```

將data打印出一列，來查看一下特徵有哪些

[1.207e+01, 2.160e+00, 2.170e+00, 2.100e+01, 8.500e+01, 2.600e+00, 2.650e+00, 3.700e-01, 1.350e+00, 2.760e+00, 8.600e-01, 3.280e+00, 3.780e+02]

(1) Alcohol → 1.207e+01

(2) Malic acid → 2.160e+00

(3) Ash → 2.170e+00

(4) Alcalinity of ash → 2.100e+01

(5) Magnesium → 8.500e+01

(6) Total phenols → 2.600e+00

(7) Flavanoids → 2.650e+00

(8) Nonflavanoid phenols → 3.700e-01

(9) Proanthocyanins → 1.350e+00

(10)Color intensity → 2.760e+00

(11)Hue → 8.600e-01

(12)OD280/OD315 of diluted wines → 3.280e+00

(13)Proline → 3.780e+02

查看訓練及測試資料集數據

```
print('x_test:測試用特徵')
print(x_test)
print('------------------------------------------------------------')
print('x_train:訓練用特徵')
print(x_train)
print('------------------------------------------------------------')
print('y_test:測試用標籤')
print(y_test)
print('------------------------------------------------------------')
print('y_train:訓練用標籤')
print(y_train)
```

```
x_test:測試用特徵
[[1.207e+01 2.160e+00 2.170e+00 2.100e+01 8.500e+01 2.600e+00 2.650e+00
  3.700e-01 1.350e+00 2.760e+00 8.600e-01 3.280e+00 3.780e+02]
 [1.382e+01 1.750e+00 2.420e+00 1.400e+01 1.110e+02 3.880e+00 3.740e+00
  3.200e-01 1.870e+00 7.050e+00 1.010e+00 3.260e+00 1.190e+03]
 [1.369e+01 3.260e+00 2.540e+00 2.000e+01 1.070e+02 1.830e+00 5.600e-01
  5.000e-01 8.000e-01 5.880e+00 9.600e-01 1.820e+00 6.800e+02]
 [1.141e+01 7.400e-01 2.500e+00 2.100e+01 8.800e+01 2.480e+00 2.010e+00
  4.200e-01 1.440e+00 3.080e+00 1.100e+00 2.310e+00 4.340e+02]
 [1.182e+01 1.720e+00 1.880e+00 1.950e+01 8.600e+01 2.500e+00 1.640e+00
  3.700e-01 1.420e+00 2.060e+00 9.400e-01 2.440e+00 4.150e+02]]
----------------------------------------------------------
x_train:訓練用特徵
[[1.358e+01 1.660e+00 2.360e+00 ... 1.090e+00 2.880e+00 1.515e+03]
 [1.406e+01 2.150e+00 2.610e+00 ... 1.060e+00 3.580e+00 1.295e+03]
 [1.243e+01 1.530e+00 2.290e+00 ... 6.900e-01 2.840e+00 3.520e+02]
 ...
 [1.216e+01 1.610e+00 2.310e+00 ... 1.330e+00 2.260e+00 4.950e+02]
 [1.200e+01 3.430e+00 2.000e+00 ... 9.300e-01 3.050e+00 5.640e+02]
 [1.182e+01 1.470e+00 1.990e+00 ... 9.500e-01 3.330e+00 4.950e+02]]
----------------------------------------------------------
y_test:測試用標籤
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
----------------------------------------------------------
y_train:訓練用標籤
[0 0 1 0 1 2 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 2 2 2 2 2 0 2 0 1 1 2 1 0 0 2
 1 1 0 1 2 0 2 0 2 2 1 0 1 1 2 1 0 1 0 1 1 0 0 1 0 2 2 2 1 1 2 1 2 0 1 1 1
 1 0 0 1 0 0 1 1 2 1 2 0 2 1 0 2 2 1 1 1 1 2 0 0 2 1 2 1 2 1 0 0 1 1 1 0 1
 1 1 0 0 0 1 2 0 0 1 2 2 0 0 2 1 2 0 2 1 0 2 1 0 0 2 0 2 1 1 1]
```

← 20%特徵
(因數據過多只打印出5組)

← 80%特徵

← 20%標籤

← 80%標籤

KNN-曼哈頓距離分類器

```
# ---最短距離---
```

```
knn = KNeighborsClassifier(p = 1)
# 定義模塊,設定p值為1,p值為Minkowski metric參數,p=1使用曼哈頓距離
knn.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(knn.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 0 1 1 1 2 2 1 0 1 2 0 2 0 0 1 0 0 1 2 1 0 0 2 1 1 2 2 0 1 1 1]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

KNN-歐幾里得距離分類器

```
# ---KNN分類---
```

```
from sklearn.neighbors import KNeighborsClassifier
# 導入模塊
```

```
knn = KNeighborsClassifier(p = 2)
# 定義模塊,設定p值為2,p值為Minkowski metric參數,p=2使用歐幾里得距離
knn.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(knn.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 2 2 1 1 2 2 2 0 1 2 0 2 0 1 1 0 0 1 2 1 0 0 2 1 1 2 1 0 1 1 2]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

決策樹分類器

```
# ---決策樹---
```

```
from sklearn.tree import DecisionTreeClassifier
# 導入模塊
```

```
tree = DecisionTreeClassifier()
# 定義模塊
tree.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(tree.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 1 2 0 1 1 0 2 2 1 1 1 1 2 1 2 2 0 1 1 1]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

參考資料

SKLEARN官網:

HTTPS://SCIKIT-LEARN.ORG/STABLE/INDEX.HTML

莫煩PYTHON:

HTTPS://MORVANZHOU.GITHUB.IO/TUTORIALS/MACHINE-LEARNING/SKLEARN/

完整程式碼參考: HTTPS://GITHUB.COM/ANUISE/PYTHONPRACTICE-/BLOB/MASTER/SORT/SKLEARN%20SORT.IPYNB

THANK YOU