

程式設計 (二)

機器學習辨認鳶尾花

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Spring Semester, 2022

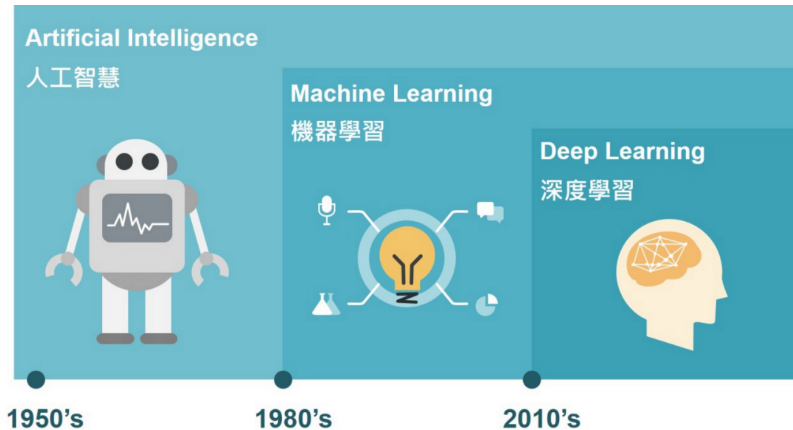
本章目錄

1. 機器學習簡介
2. 資料集與方法
3. 程式實作

機器學習簡介

什麼是機器學習？

機器學習為人工智慧的分支，目的在於使電腦具有自動學習的能力，能從資料中尋找出規律，進而做出最佳的決策與預測。



機器學習分為哪些？

- 監督式學習 (Supervised learning)
- 非監督式學習 (unsupervised learning)
- 強化式學習 (Reinforcement learning)
- ...etc

具有標準答案 (Label)。
於訓練資料中學習、建立一個 learning model，
並藉由 learning model 去推測未知資料樣本。
常用於分類與迴歸問題。

常見演算法：

- 決策樹 (Decision tree)
- 支持向量機 (Support Vector Machine)
- 邏輯斯迴歸 (Logistic regression)
- 線性迴歸 (linear regression)
- ...etc

監督式學習



(a) Label: 猴子



(b) Label: 猴子跑山

訓練資料內的資料樣本並無標準答案，
於訓練的根據資料樣本的相關性、相似度等進行建立模型的依據。
常用於分群 (cluster analysis)、關聯規則分析 (association rule analysis)
等等。

常見演算法：

- k-平均演算法 (k-means clustering)
- Apriori
- FP-Growth
- ...etc

非監督式學習



(a) Label: 猴子



(b) Label: 猴子跑山

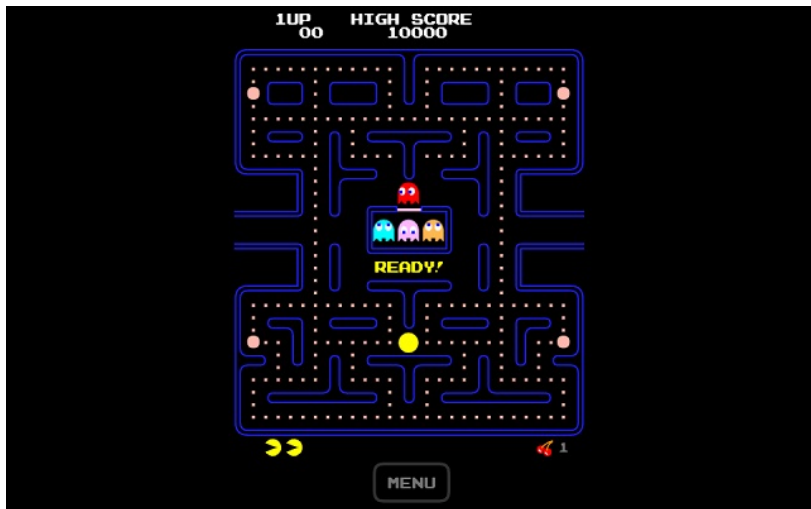
與上述兩種學習方式不同，強化式學習強調基於環境行動，因此將有一個代理人 (Agent) 於環境中不斷嘗試各種行為 (Action)，並根據每個行為所帶來的回饋 (Reward) 不斷調整，以取得最大化的利益。

強化式學習

常見演算法：

- Q-learning
- SARSA
- ...etc

強化式學習



此資料共包含了 150 個資料樣本，每個樣本皆有 4 個特徵：

- 花萼長度
- 花萼寬度
- 花瓣長度
- 花瓣寬度

而其屬種（亦即 Label）共分為三類：

- 山鳶尾 (setosa)
- 變色鳶尾 (versicolor)
- 維吉尼亞鳶尾 (virginica)

鳶尾花資料集

費雪鳶尾花卉資料集

| 花萼長度 ◆ | 花萼寬度 ◆ | 花瓣長度 ◆ | 花瓣寬度 ◆ | 屬種 ◆ |
|--------|--------|--------|--------|---------------|
| 5.1 | 3.5 | 1.4 | 0.2 | <i>setosa</i> |
| 4.9 | 3.0 | 1.4 | 0.2 | <i>setosa</i> |
| 4.7 | 3.2 | 1.3 | 0.2 | <i>setosa</i> |
| 4.6 | 3.1 | 1.5 | 0.2 | <i>setosa</i> |
| 5.0 | 3.6 | 1.4 | 0.2 | <i>setosa</i> |
| 5.4 | 3.9 | 1.7 | 0.4 | <i>setosa</i> |
| 4.6 | 3.4 | 1.4 | 0.3 | <i>setosa</i> |
| 5.0 | 3.4 | 1.5 | 0.2 | <i>setosa</i> |
| 4.4 | 2.9 | 1.4 | 0.2 | <i>setosa</i> |

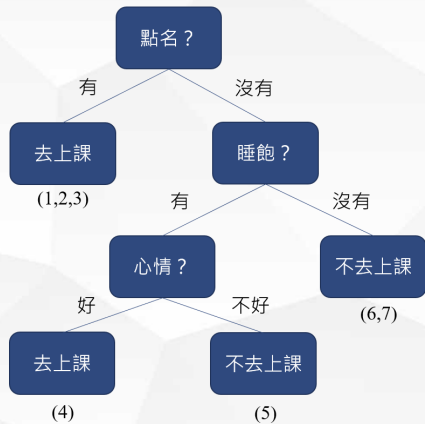
決策樹是一棵由多個規則建構而成的樹狀結構，除了葉節點 (leaf node) 以外，每個節點都代表一個規則，而選用規則的基準是根據當下能將樣本分的最乾淨的條件，作為該節點的規則。

選用規則的方法：

- 熵 (Entropy)
- Gini 不純度 (Gini Impurity)

決策樹

| 編號 | 點名 | 睡飽 | 心情 | 上課 |
|----|----|----|----|----|
| 1 | 有 | 有 | 好 | 去 |
| 2 | 有 | 有 | 好 | 去 |
| 3 | 有 | 無 | 不好 | 去 |
| 4 | 無 | 有 | 好 | 去 |
| 5 | 無 | 有 | 不好 | 不去 |
| 6 | 無 | 無 | 好 | 不去 |
| 7 | 無 | 無 | 好 | 不去 |



決策樹

優點：

- 具有高度可解釋性
- 模型容易理解
- 計算時間複雜度低

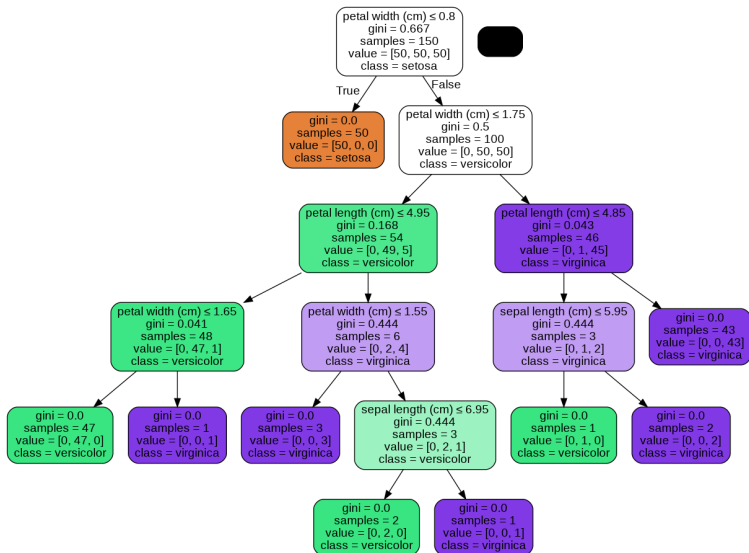
缺點：

- 容易 Overfitting（過度擬合）
- Label 越多，樹狀結構越複雜

程式實作

```
1 from sklearn import tree
2 from sklearn.datasets import load_iris
3
4 # load data
5
6 iris = load_iris()
7 clf = tree.DecisionTreeClassifier()
8 clf = clf.fit(iris.data, iris.target)
9
10 # get decision tree
11 import pydotplus
12 from IPython.display import Image
13 dot_data = tree.export_graphviz(clf, out_file=None,
14                                 feature_names=iris.feature_names,
15                                 class_names=iris.target_names,
16                                 filled=True, rounded=True,
17                                 special_characters=True)
18 graph = pydotplus.graph_from_dot_data(dot_data)
19 Image(graph.create_png())
```

程式實作




程式實作



```
1 from sklearn import datasets
2 from sklearn.model_selection import train_test_split
3 import numpy as np
4 from sklearn.tree import DecisionTreeClassifier
5
6 # prepare data
7 iris = datasets.load_iris()
8
9 iris_data = iris.data
10 iris_label = iris.target
11
12 train_data, test_data, train_label, test_label = train_test_split(iris_data, iris_label, test_size=0.2)
13
14 # Decision tree classifier with depth 3 #
15 clf1 = DecisionTreeClassifier(max_depth=3)
16 clf1.fit(train_data, train_label)
17 score1 = clf1.score(train_data, train_label)
18 score2 = clf1.score(test_data, test_label)
19 print('Results of decision tree with depth 3:')
20 print('Training acc and test acc are', score1, score2)
```

程式實作



```
1 # Decision tree classifier with depth 5 #
2 clf2 = DecisionTreeClassifier(max_depth=5)
3 clf2.fit(train_data, train_label)
4 score1 = clf2.score(train_data, train_label)
5 score2 = clf2.score(test_data, test_label)
6 print('Results of decision tree with depth 5:')
7 print('Training acc and test acc are', score1, score2)
8
9 # Decision tree classifier with depth 10 #
10 clf3 = DecisionTreeClassifier(max_depth=10)
11 clf3.fit(train_data, train_label)
12 score1 = clf3.score(train_data, train_label)
13 score2 = clf3.score(test_data, test_label)
14 print('Results of decision tree with depth 10:')
15 print('Training acc and test acc are', score1, score2)
```

程式實作



```
1 # Print test label and true label #
2 lab1 = clf1.predict(test_data)
3 lab2 = clf2.predict(test_data)
4 lab3 = clf3.predict(test_data)
5
6 for n in range(len(test_label)):
7     print(n, ': ', test_label[n], lab1[n], lab2[n], lab3[n])
```

程式實作



```
1 Results of decision tree with depth 3:  
2 Training acc and test acc are 0.9666666666666667 0.9666666666666667  
3 Results of decision tree with depth 5:  
4 Training acc and test acc are 1.0 0.9666666666666667  
5 Results of decision tree with depth 10:  
6 Training acc and test acc are 1.0 0.9666666666666667
```

程式實作



| | | | | | | |
|----|---|---|---|---|---|---|
| 1 | 0 | : | 1 | 1 | 2 | 2 |
| 2 | 1 | : | 2 | 2 | 2 | 2 |
| 3 | 2 | : | 0 | 0 | 0 | 0 |
| 4 | 3 | : | 0 | 0 | 0 | 0 |
| 5 | 4 | : | 2 | 1 | 2 | 2 |
| 6 | 5 | : | 1 | 1 | 1 | 1 |
| 7 | 6 | : | 1 | 1 | 1 | 1 |
| 8 | 7 | : | 0 | 0 | 0 | 0 |
| 9 | 8 | : | 2 | 2 | 2 | 2 |
| 10 | 9 | : | 2 | 2 | 2 | 2 |



| | | | | | | |
|----|----|---|---|---|---|---|
| 1 | 10 | : | 1 | 1 | 1 | 1 |
| 2 | 11 | : | 2 | 2 | 2 | 2 |
| 3 | 12 | : | 0 | 0 | 0 | 0 |
| 4 | 13 | : | 2 | 2 | 2 | 2 |
| 5 | 14 | : | 1 | 1 | 1 | 1 |
| 6 | 15 | : | 1 | 1 | 1 | 1 |
| 7 | 16 | : | 1 | 1 | 1 | 1 |
| 8 | 17 | : | 2 | 2 | 2 | 2 |
| 9 | 18 | : | 2 | 2 | 2 | 2 |
| 10 | 19 | : | 1 | 1 | 1 | 1 |



| | | | | | | |
|----|----|---|---|---|---|---|
| 1 | 20 | : | 2 | 2 | 2 | 2 |
| 2 | 21 | : | 0 | 0 | 0 | 0 |
| 3 | 22 | : | 2 | 2 | 2 | 2 |
| 4 | 23 | : | 1 | 1 | 1 | 1 |
| 5 | 24 | : | 0 | 0 | 0 | 0 |
| 6 | 25 | : | 1 | 1 | 1 | 1 |
| 7 | 26 | : | 2 | 2 | 2 | 2 |
| 8 | 27 | : | 0 | 0 | 0 | 0 |
| 9 | 28 | : | 1 | 1 | 1 | 1 |
| 10 | 29 | : | 1 | 1 | 1 | 1 |

END