D19 K-近鄰演算法

重要知識點

- 深入了解 KNN 原理與優缺點

資料來源：K-Nearest Neighbors and Bias–Variance Tradeoff

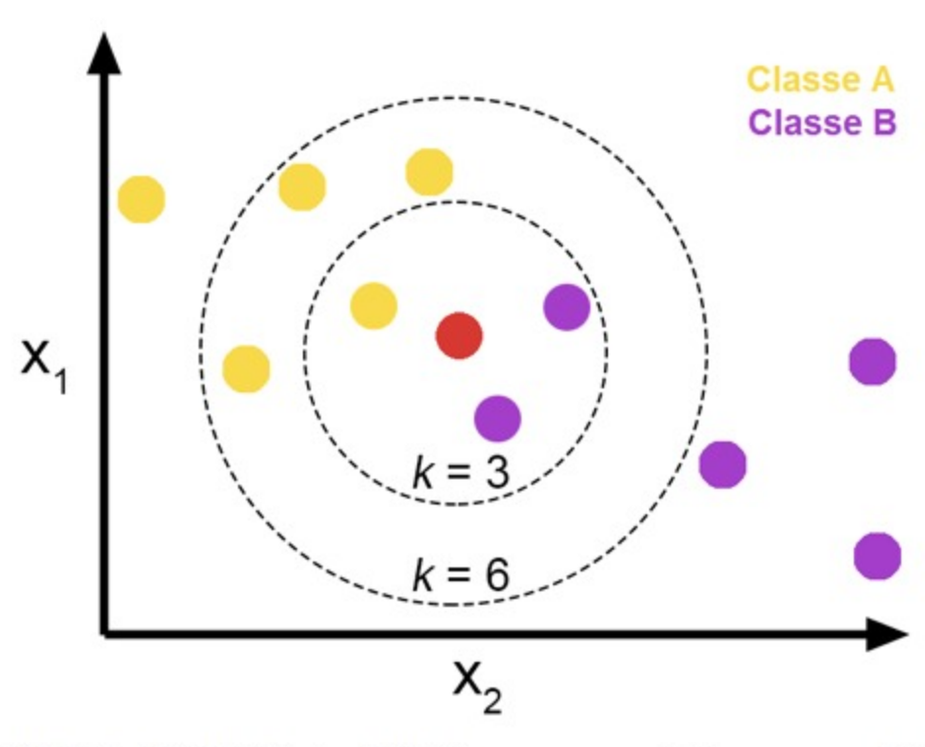
K 值的選擇

KNN 中 K 值的選擇沒有標準的作法，但我們可以遵循一些規則針對不同的資料集選擇適當的 K 值。

不合適 K 值帶來的問題：

過小的 K 值：

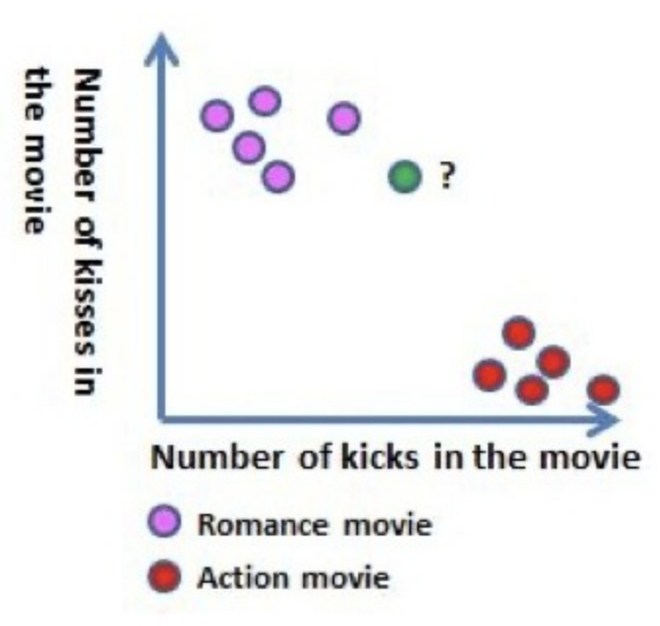
- 通常過小的 K 值會導致『Overfitting training data』，也就是模型的泛化能力不足。
- 綜觀來看紅點更接近 ClasseA 類別，然而當 K 值過小時，紅點就很容易受到一些 Outliers 影響。



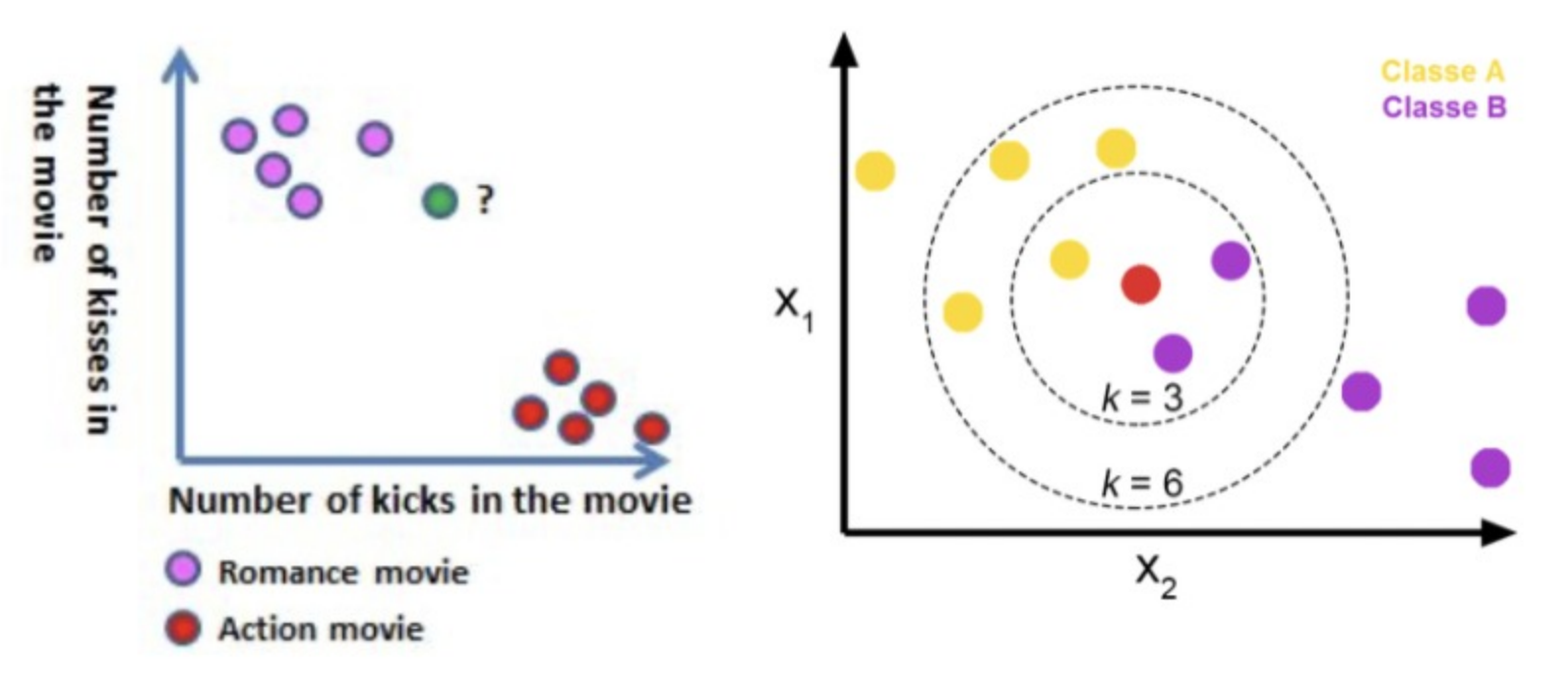
資料來源：機器學習中的K最近鄰(k-Nearest Neighbor，KNN)分類算法簡介

過大的 K 值：

- 過大的K值往往會導致『Underfitting training data』，也就是模型缺乏了鑑別力。
- 由上圖可知，當 K 值設置過大時，模型就失去了粉紅色群體對綠色點較為重要的資訊。
- 過大K值另外一個壞處是需要更久的運算時間。



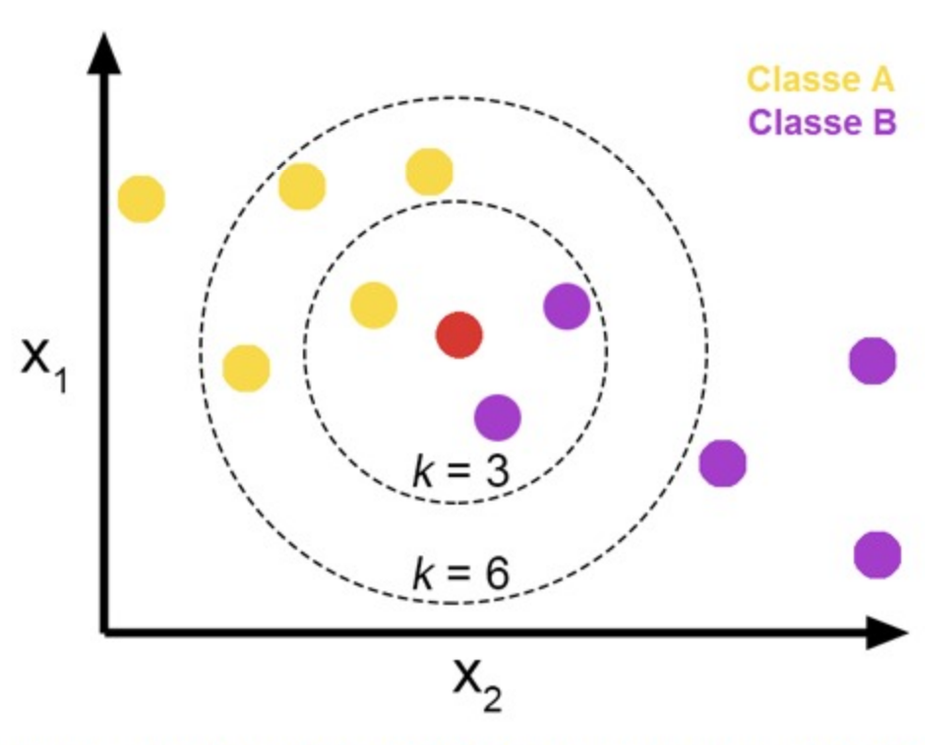
資料來源：Classifying with k-Nearest Neighbors



上圖為了視覺化方便，以二維抽象空間解釋多維空間的資料，實際應用 KNN 時，維度應該為訓練資料集的特徵數量。

適當的 K 值選擇

- K 值應為**奇數**，以避免無法判斷的情況發生。
- 以下圖為例，當我們設置 K 值為 4 時，可能就會造成 ClasseA/B 各有兩個樣本。



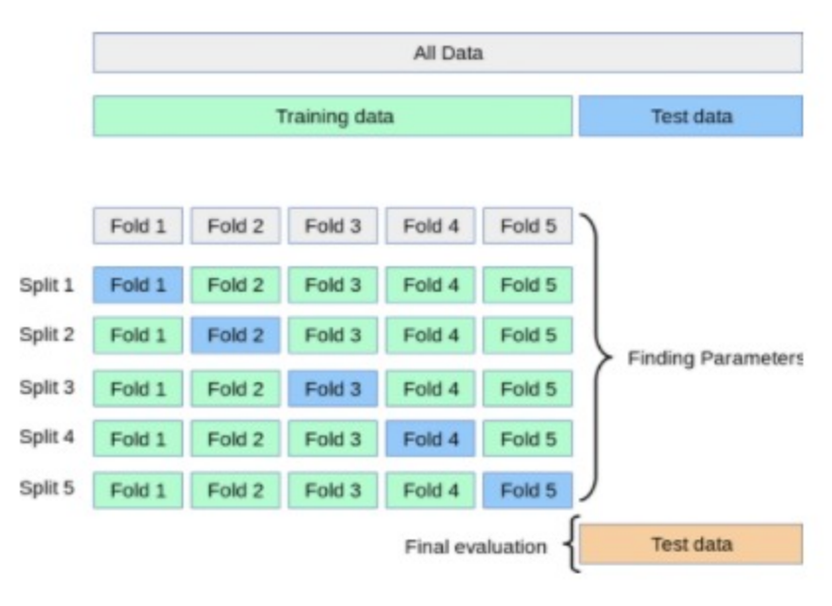
資料來源：機器學習中的K最近鄰(k-Nearest Neighbor，KNN)分類算法簡介

- 常見的作法是設置 K = sqrt(N)，其中N為訓練集的資料量。
- 因此當我們訓練集擁有 10000 個樣本時，K 值就可以設為 100。
- 儘管設置 K = sqrt(N) 相當方便，但往往並不是最適合的值。
- 另外一個常見的方式是採用 “cross-validation” (交叉驗證)

補充資料：Cross-validation: evaluating estimator performance

K-fold

- K-fold 為 cross-validation 中常見的作法。
- cross-validation 可以用來做『Hyperparameters tuning』。



資料來源：Cross-validation: evaluating estimator performance

- 我們將訓練集拆成 K 份，每次取出 1/K 份當驗證集，(K-1)/K 當訓練集，因此我們最後可以獲得 K 個驗證集的結果。
- 這裡的 K 為 K-fold 中的K，並不是 KNN 中的 K 值，也就是要將訓練資料拆成幾分的意思。

- 當今天我們有 100 筆資料，我們設置切為 5 份，代表每此訓練都會選用 80 筆資料當訓練集，剩下 20 筆資料當驗證集。
- 我們持續上述做法，直到所有資料都當過驗證集，因此以上述例子，我們切成五份代表要訓練五次。
- 這裡要切記，每次的訓練都獨立完成，彼此之間不分享資訊。
- 最後我們可以將 5 組驗證集得到的結果平均，得到最終結果。

下方示範 K-fold 用法，可以看到 K-fold 會切割原本的訓練集，主要是透過紀錄資料 index 的方式來返回每次訓練/驗證集。

```
import numpy as np
from sklearn.model_selection import KFold
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4])
kf = KFold(n_splits=2)
kf.get_n_splits(X)

print(kf)

for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

KFold(n_splits=2, random_state=None, shuffle=False)
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
```

資料來源：Cross-validation: evaluating estimator performance

K-fold 好處

- 現實中我們難以收集到大量的訓練集資料，透過 K-fold 我們可以在有限的訓練集內多次測試模型的泛化能力。
- 除此之外，K-fold 得到的結果往往更有代表性，因為我們是透過多個驗證集獲得平均值。

適當的 K 值選擇

K-fold 運用在 KNN 中 K 值的選擇上：

- 我們可以**預先選定幾個 K 值**，如 10、50、100，運用 K-fold 方式得到不同 K 值在驗證集上得到的平均準確度，藉此選擇最適合的 K 值。

KNN 回顧

KNN 雖然被歸類為監督式學習，但當我們了解後可以發現 KNN 是透過記住訓練集資料的位置，在預測時透過比對預測資料與訓練集樣本的距離決定最終預測結果，所以**並沒有真正『訓練的過程』**。

參考資料

Cross-validation 的概念先當常見且重要，有興趣的讀者們可以參考下方文章：

- 交叉驗證介紹(Cross-validation)

交叉驗證（交叉驗證，CV）

湯米·黃 跟隨
2018年3月29日 · 5 分鐘閱讀

Python代碼

交叉驗證在機器學習上通常是用來驗證“你設計出來模型”的好壞。

預期：

1.數據庫（database）沒有先切割好「訓練資料（Training data）」和「測試資料（Testing data）」

或是

2.你要從「訓練資料（訓練資料）」找到一組最合適的參數出來，一些SVM的懲罰參數（懲罰參數），就可以從訓練資料（訓練資料）做交叉驗證找出來，而不是從「測試資料（Testing data）」得到參數。

> *機器學習最忌諱把「測試資料（Testing data）」偷偷拿進到模型內訓練或找參數。在做模型性能評估時，要記住一件事情「測試資料（Testing data）」絕對不能進到模型內訓練或是找參數。*

重點

PS：但主動學習和某些半監督學習方法會額外抽出一些資料，但是不會跟模型說這些資料的答案（Ground Truth），出來讓模型更好。

下一步：閱讀範例與完成作業