**AI人工智慧無監督學習：聚類**

**AI人工智慧 聚類概述**

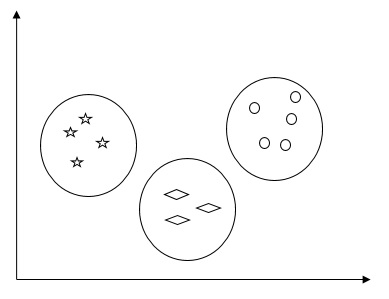
2020-09-24 11:17 更新

無監督機器學習演算法沒有任何監督者提供任何指導。 這就是為什麼它們與真正的人工智慧緊密結合的原因。

在無人監督的學習中，沒有正確的答案，也沒有監督者指導。 演算法需要發現用於學習的有趣資料模式。

基本上，它是一種無監督學習方法，也是用於許多領域的統計資料分析的常用技術。 聚類主要是將觀測集合劃分為子集(稱為聚類)的任務，以同一聚類中的觀測在一種意義上相似並且與其他聚類中的觀測不相似的方式。 簡而言之，可以說聚類的主要目標是根據相似性和不相似性對資料進行分組。

例如，下圖顯示了不同群集中的類似資料 -



# AI人工智慧 資料聚類演算法

2020-09-23 16:25 更新

以下是資料聚類的幾種常用演算法 -

**K-Means 演算法** K均值聚類演算法是眾所周知的資料聚類演算法之一。 我們需要假設簇的數量已經是已知的。 這也被稱為平面聚類。 它是一種反覆運算聚類演算法。 該演算法需要遵循以下步驟 -

**第1步** - 需要指定所需的K個子組的數量。 **第2步** - 修復群集數量並將每個數據點隨機分配到群集。 換句話說，我們需要根據群集數量對資料進行分類。

在這一步中，計算聚類質心。

由於這是一種反覆運算演算法，因此需要在每次反覆運算中更新 K 個質心的位置，直到找到全域最優值或換句話說質心到達其最佳位置。

以下代碼將有助於在 Python 中實現 K-means 聚類演算法。 我們將使用 Scikit-learn 模組。

導入必需的套裝軟體 -

1. import matplotlib.pyplot as plt
2. import seaborn as sns; sns.set()
3. import numpy as np
4. from sklearn.cluster import KMeans

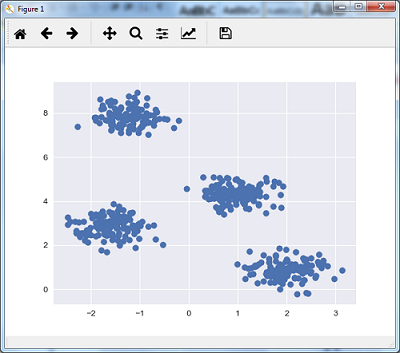
以下代碼行將通過使用 sklearn.dataset包中的 make\_blob 來生成包含四個 blob 的二維資料集。

1. from sklearn.datasets.samples\_generator import make\_blobs
2. X, y\_true = make\_blobs(n\_samples = 500, centers = 4,
3. cluster\_std = 0.40, random\_state = 0)

可以使用下面的代碼視覺化資料集 -

1. plt.scatter(X[:, 0], X[:, 1], s = 50);
2. plt.show()

得到以下結果 -



在這裡，將 kmeans 初始化為 KMeans 演算法，以及多少個群集 (n\_clusters) 所需的參數。

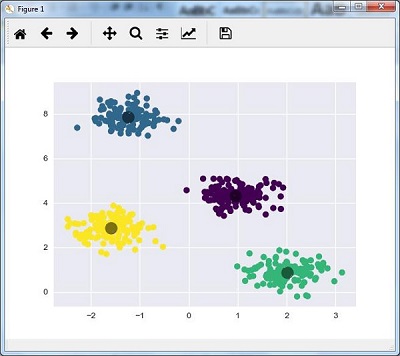
1. kmeans = KMeans(n\_clusters = 4)

需要用輸入資料訓練 K-means 模型。

1. kmeans.fit(X)
2. y\_kmeans = kmeans.predict(X)
3. plt.scatter(X[:, 0], X[:, 1], c = y\_kmeans, s = 50, cmap = 'viridis')
4. centers = kmeans.cluster\_centers\_

下面給出的代碼將根據資料繪製和視覺化機器的發現，並根據要找到的聚類數量進行擬合。

1. plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 200, alpha = 0.5);
2. plt.show()

得到以下結果 - 

**均值偏移演算法**

它是另一種在無監督學習中使用的流行和強大的聚類演算法。 它不做任何假設，因此它是非參數演算法。 它也被稱為分層聚類或均值聚類分析。 以下將是該演算法的基本步驟 -

* 首先，需要從分配給它們自己的集群的資料點開始。
* 現在，它計算質心並更新新質心的位置。
* 通過重複這個過程，向簇的頂點靠近，即朝向更高密度的區域移動。
* 該演算法停止在質心不再移動的階段。

在下面的代碼的幫助下，在 Python 中實現了 Mean Shift 聚類演算法。使用 Scikit-learn 模組。

導入必要的套裝軟體 -

1. import numpy as np
2. from sklearn.cluster import MeanShift
3. import matplotlib.pyplot as plt
4. from matplotlib import style
5. style.use("ggplot")

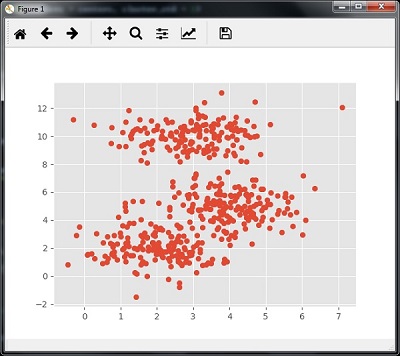
以下代碼將通過使用 sklearn.dataset 包中的 make\_blob 來生成包含四個 blob 的二維資料集。

1. from sklearn.datasets.samples\_generator import make\_blobs

可以用下面的代碼視覺化資料集 -

1. centers = [[2,2],[4,5],[3,10]]
2. X, \_ = make\_blobs(n\_samples = 500, centers = centers, cluster\_std = 1)
3. plt.scatter(X[:,0],X[:,1])
4. plt.show()

執行上面示例代碼，得到以下結果 -



現在，我們需要用輸入資料來訓練 Mean Shift 聚類模型。

1. ms = MeanShift()
2. ms.fit(X)
3. labels = ms.labels\_
4. cluster\_centers = ms.cluster\_centers\_

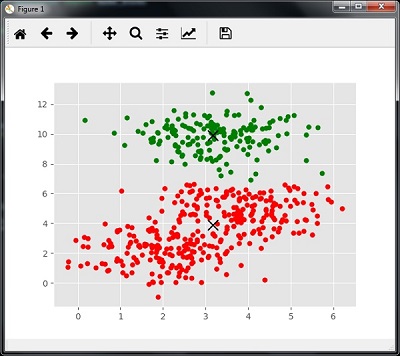
以下代碼將按照輸入資料列印聚類中心和預期的聚類數量 -

1. print(cluster\_centers)
2. n\_clusters\_ = len(np.unique(labels))
3. print("Estimated clusters:", n\_clusters\_)
4. [[ 3.23005036 3.84771893]
5. [ 3.02057451 9.88928991]]
6. Estimated clusters: 2

下面給出的代碼將有助於根據資料繪製和視覺化機器的發現，並根據要找到的聚類數量進行裝配。

1. colors = 10\*['r.','g.','b.','c.','k.','y.','m.']
2. for i in range(len(X)):
3. plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)
4. plt.scatter(cluster\_centers[:,0],cluster\_centers[:,1],
5. marker = "x",color = 'k', s = 150, linewidths = 5, zorder = 10)
6. plt.show()

執行上面示例代碼，得到以下結果 -



## 測量群集性能

現實世界的資料不是自然地組織成許多獨特的群集。 由於這個原因，要想像和推斷推理並不容易。 這就是為什麼需要測量聚類性能及其品質。 它可以在輪廓分析的幫助下完成。

**輪廓分析**

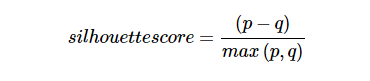
該方法可用於通過測量群集之間的距離來檢查聚類的品質。 基本上，它提供了一種通過給出輪廓分數來評估像集群數量這樣的參數的方法。 此分數是衡量一個群集中每個點與相鄰群集中的點的距離的度量。

**分析輪廓分數** 得分範圍為[-1，1]。 以下是對這個分數的分析 -

* 得分為 +1分 - 得分接近 +1表示樣本距離相鄰集群很遠。
* 得分為 0分 - 得分 0 表示樣本與兩個相鄰群集之間的決策邊界處於或非常接近。
* 得分為 -1分 - 得分為負分數表示樣本已分配到錯誤的群集。

## 計算輪廓分數

在本節中，我們將學習如何計算輪廓分數。

輪廓分數可以通過使用以下公式來計算 - 

以上內容是