## 实验 4 聚类学习

1. **实验目的**

学习理解 KMeans 算法为代表的聚类算法的原理。

1. **实验步骤**
   1. 下载 sklearn 中 DBSCAN 的样例代码；
   2. 运行代码理解聚类算法的基本作用；
   3. 运行 min\_samples 参数在不同取值下的结果，理解参数含义；
   4. 将 DBSCAN 算法更换为 KMeans 算法，观察区别；
   5. 总结原型聚类方法 KMeans 和密度聚类方法 DBSCAN 的区别；
2. **实验结果展示**

**3.1 DBSCAN 算法**

**3.1.1 定义**

a) 目标对象

当前的研究对象，可以是核心对象，也可以是普通的样本对象。

b) 领域

以目标对象为中心，可以形成一个指定半径的圆，圆的区域称为领域，用以考察样 本点之间的关系。

c) 核心对象

若某个目标对象领域内的样本点数量达到了指定值（min\_samples），则称此目 标对象为核心对象。

d) 密度直达

若某个样本点 A 在某个核心对象 B 的领域内，则称 A 由 B 密度直达，即 A 受到 B 的领域控制。

e) 密度可达

若核心对象 T1 与 T2 在相互的领域内，则称此为密度可达，密度可达存在传递 性，这个性质可以连接多个原本“无关”的领域空间。

**3.1.2 特殊情况**

a) 噪音点

某些样本点既不是核心对象，也不在任意一个核心对象的领域内，则称这些点为噪 音点。

b) ambi-point

存在一种特殊情况，两个核心对象相互之间密度不直达，也不可达，但是某个目标 对象可以分别由这两个核心对象密度直达。至于到底需要将此目标对象分到两个核 心对象代表的两个簇中的哪一个，可能的算法是“谁先到，谁说了算”。

**3.1.3 算法流程**

a） 对于一个样本集 D ，给出领域半径 r ，以及核心对象领域内的样本点数量 n ，样本距离计算方法 F。

b) 初始化。令聚类簇的数量为 Cn = 0

c) 对 D 集合计算获得核心对象集 Dcore ；

d) 从 Dcore 中随机取出一个核心对象，找出所有与其密度可达的核心对象集（包 括之前的核心对象） Dcore\_radius 及其领域集，将领域集中的所有样本（包 括核心对象） Ditem 归为一类。同时，Dcore 集合减去 Dcore\_radius 集 合， D 集合减去 Ditem 集合，避免重复计算。

e) 重复 d 步骤，直到 Dcore 为空集。

f) 将 D 中剩余的样本设置为噪音点。

**3.1.3 算法区别**

a) DBSCAN 不需要提前告知聚类簇的数量，K-means 需要提前设定。

b) DBSCAN 可以发现任意形状的聚类簇，K-means 一般适用于凸函数数据集。

c) DBSCAN 在聚类过程存在噪音点的概念，可能不能将所有点聚类。

**3.2 更换算法**

**3.2.1 代码**

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

centers = [[1, 1], [-1, -1], [1, -1]]

X, labels\_true = make\_blobs(n\_samples=750, centers=centers,

cluster\_std=0.4, random\_state=0)

X = StandardScaler().fit\_transform(X)

random\_state = 100

y\_pred = KMeans(n\_clusters=3, random\_state=random\_state).fit\_predict(X)

""" 画画

"""

unique\_labels = set(y\_pred)

colors = [plt.cm.Spectral(each)

for each in np.linspace(0, 1, len(unique\_labels))]

colors = ListedColormap(colors)

plt.figure()

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap=colors) # 补充训练数据点

plt.xlim(X[:, 0].min(), X[:, 0].max())

plt.ylim(X[:, 1].min(), X[:, 1].max())

plt.title("K-means 3 classification")

plt.show()

**3.2.2 运行结果**



1. 实验总结与收获

通过本次实验，我学习了解了 DESCAN 算法，同时对比了其他的聚类算法。