《机器学习基础》

崔志勇

交通科学与工程学院

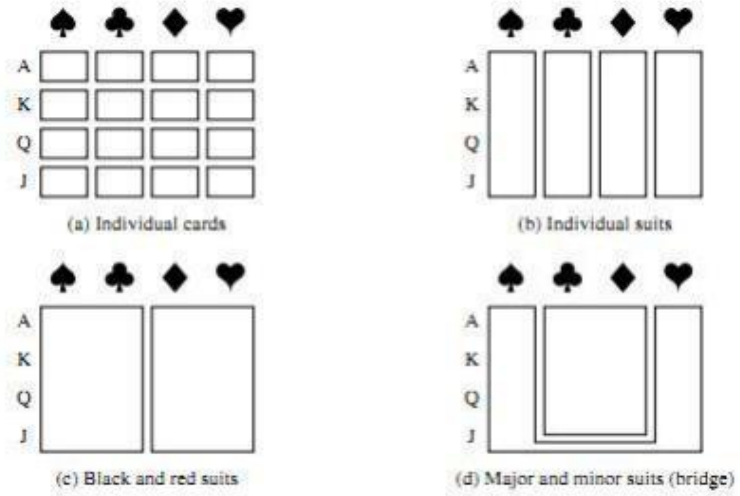
2024年5月

聚类

聚类概述

聚类分析

- 聚类（Cluster）分析是由若干模式（Pattern）组成的，通常，模式是一个度量的向量，或者是多维空间中的一个点。
- 聚类分析以相似性为基础，在一个聚类中的模式之间比不在同一聚类中的模式之间具有更多的相似性。
  - 目标：发现项目（或变量） 的自然分组方法
  - 探索性方法理解复杂多元关系
  - 依赖于相似性定义和度量
  - 通用算法寻找较好分组（未必最好， 排列组合）



聚类方法

传统聚类算法：

①划分法　②层次方法　③基于密度方法　④基于网络方法　⑤基于模型方法

聚类算法分类：

- **分层聚类方法**

分层聚类方法是开始时把每个样品作为一类，然后把最靠近的样品（即距离最小的群品）首先聚为小类，再将已聚合的小类按其类间距离再合并，不断继续下去，最后把一切子类都聚合到一个大类。

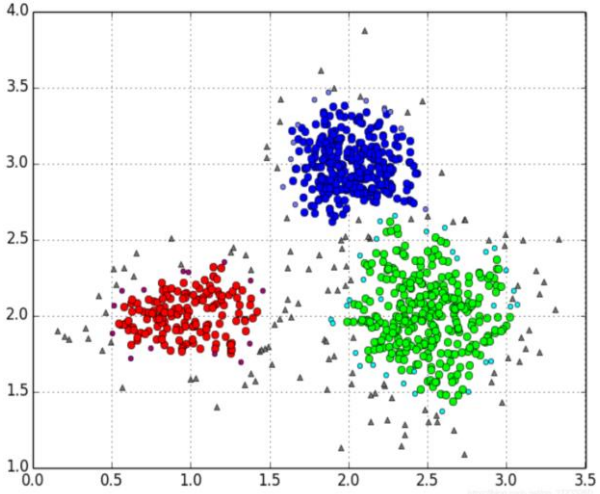
- **分割分层法**

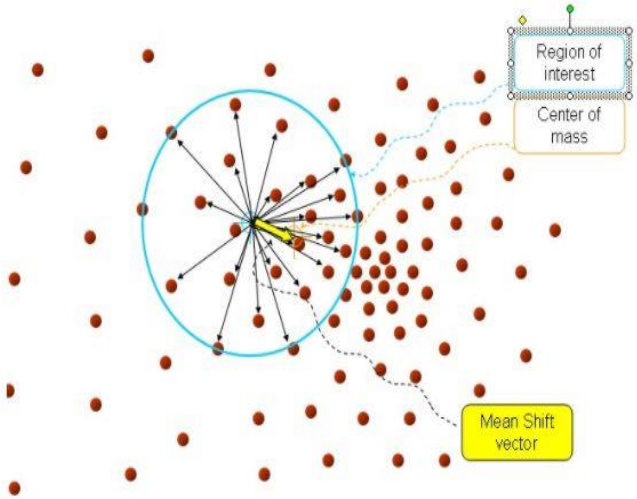
最初由所有对象组成的一个组开始，将它分割成两个子组，使一个子组的对象“远离”另一子组的对象。然后将这两个组进一步 分割成不相似的组。这一过程一直进行到每个对象单独成为一组时为止。

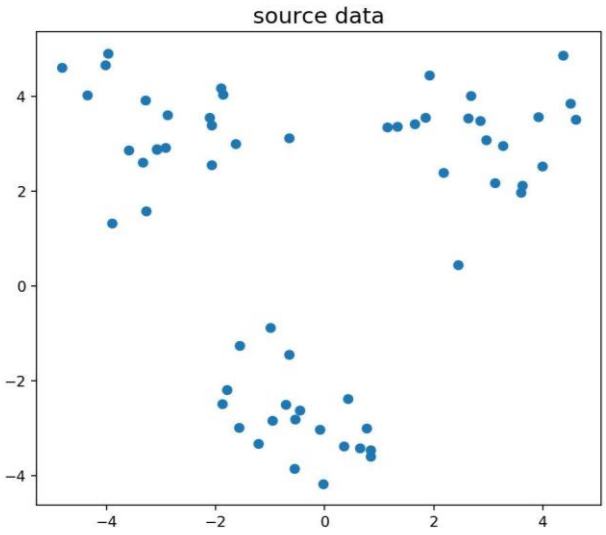
聚类方法

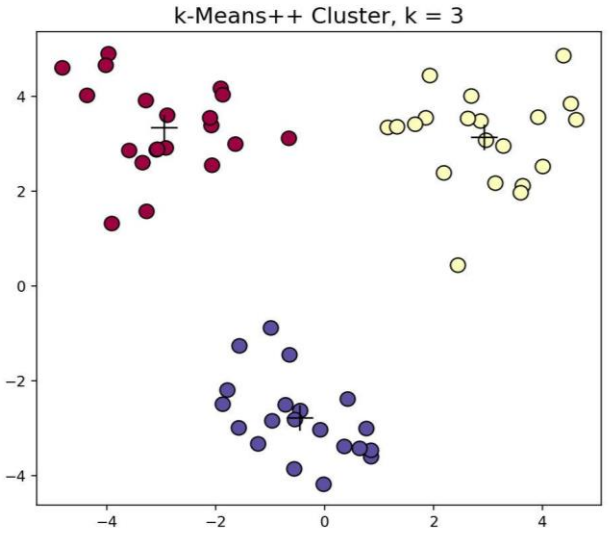
具体方法

- K-MEANS
- K-MEDOIDS
- Clara
- Clarans
- DBSCAN
- OPTICS









K-means聚类算法

K均值法

基本思想是将每一个项目分给具有最近中心K（均值）的聚类。

步骤:

1. 初始化：随机选择 **K 个**数据点作为**初始簇中心**。
2. 分配：将每个数据点**分配**给**最近的簇中心**。
3. 更新：**重新计算**每个**簇的中心**。
4. 迭代：重复步骤 2 和 3 **直到簇中心不再发生变化**或**达到预设的迭代次数**

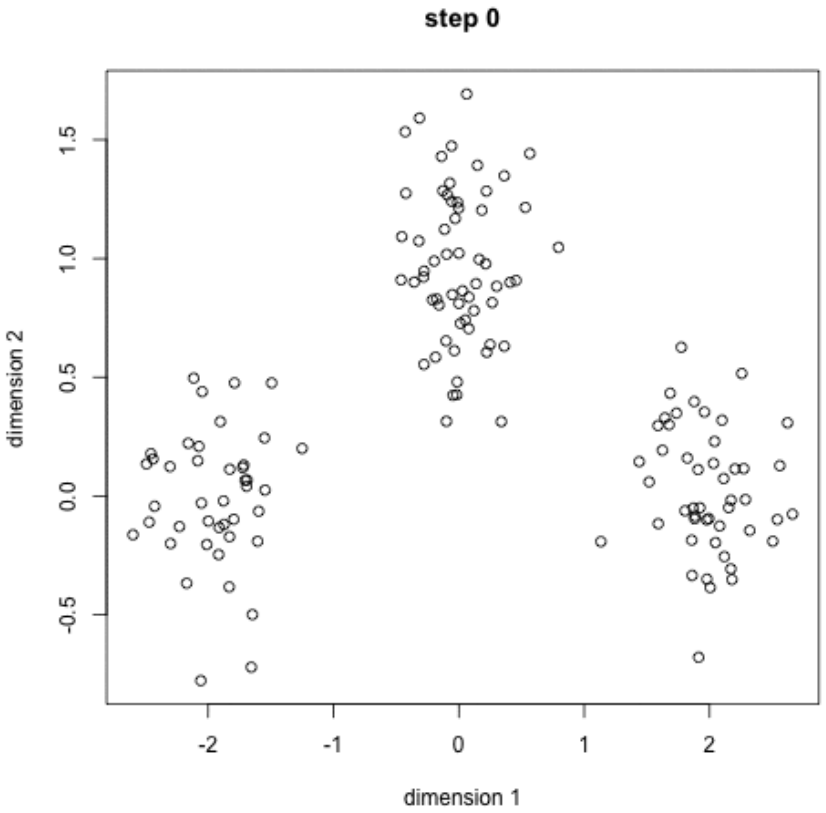
注：1) 上述过程中，步骤1也可以不从分割出K个初始聚类开始， 而从规定K个初始中心（中心点） 开始，然后进入步骤2。

2) 最终的分类结果在某种程度上依赖于初始分组或初始中心点的选择。经验显示， 聚类过程中的绝大多数重要变化均发生在第一次再分配中。

K-means聚类算法

K-means目标函数

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(y_i-x_i)^2}$$



Python 实现

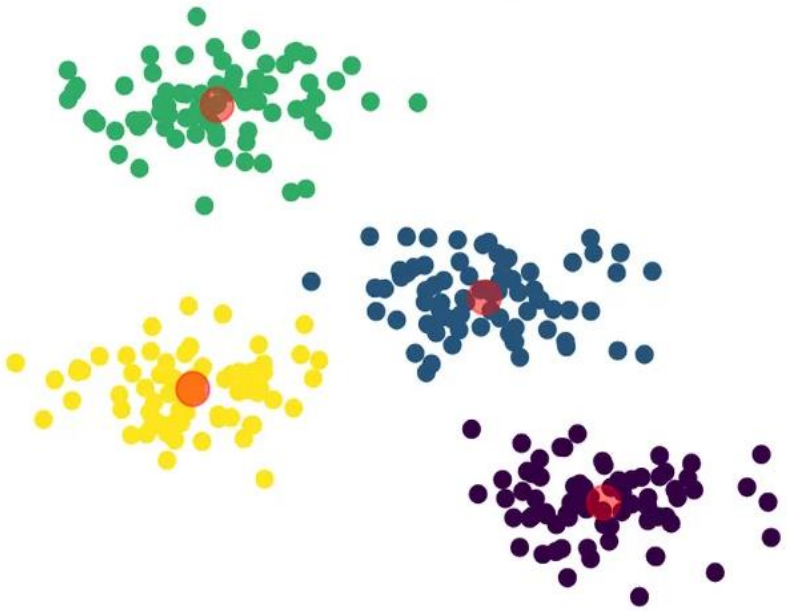
使用 Python 的 sklearn 库来实现 K-means 算法

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# 生成模拟数据
np.random.seed(0)
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# 应用K-means算法
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# 绘制数据点和聚类中心
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5)
plt.title("K-means Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

K-means聚类算法

K-means聚类优点：

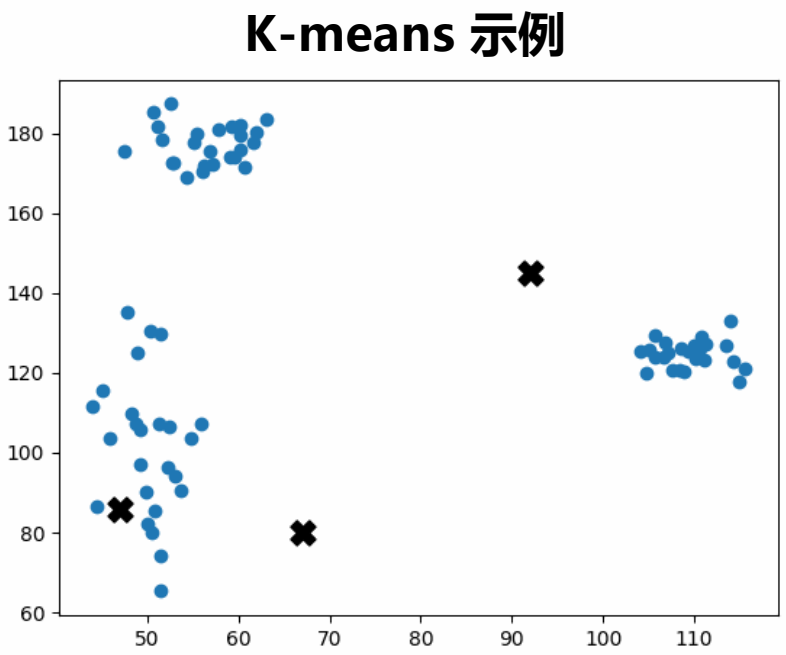
- K-mean算法简单高效
- K-mean算法具有广泛的适用性
- K-mean算法计算迭代的速度加快



K-means聚类缺点：

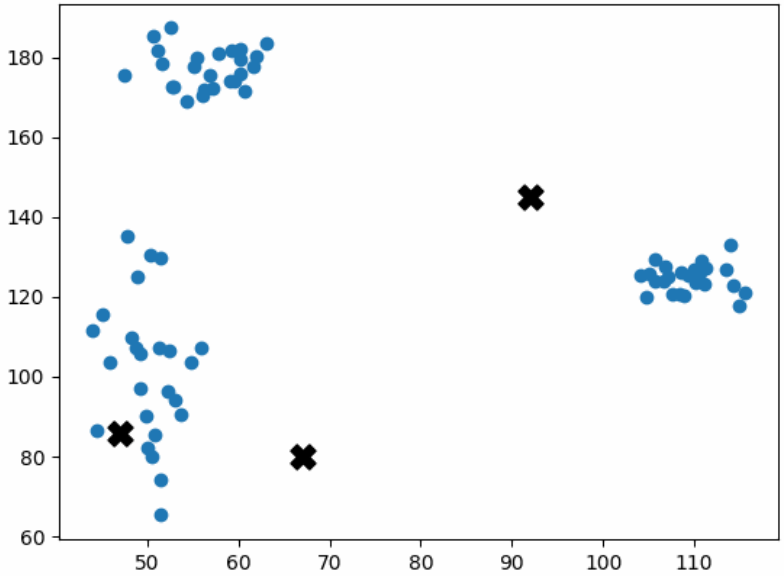
- K-mean算法**必须先确定分类数量**
- K-mean算法**迭代速度**和**初始聚类中心的选择**有很大关系
- K-mean算法在不同的算法运行中**可能产生不同的聚类结果**，**结果不可重复**
- K-mean算法可能会**陷入局部最优**，且假设簇是凸形的，**对于复杂形状的数据可能不适用**

K-means聚类算法

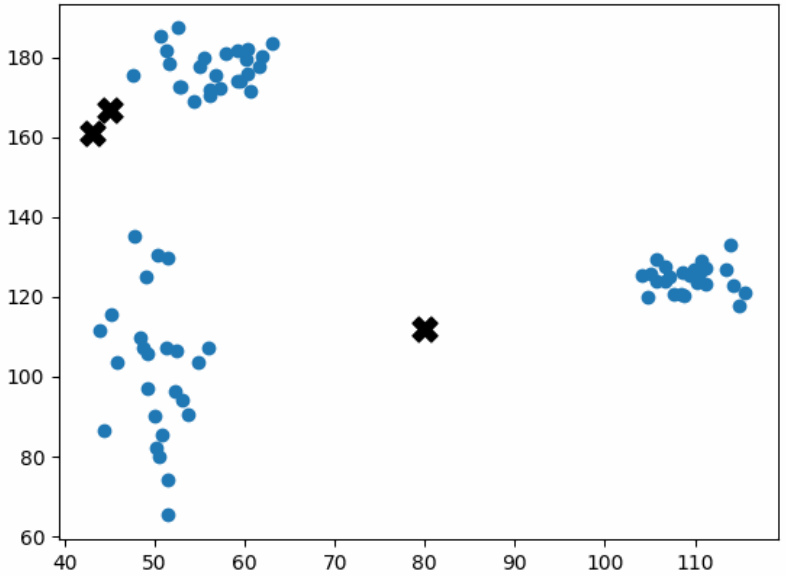


K-means聚类算法

K-means 示例

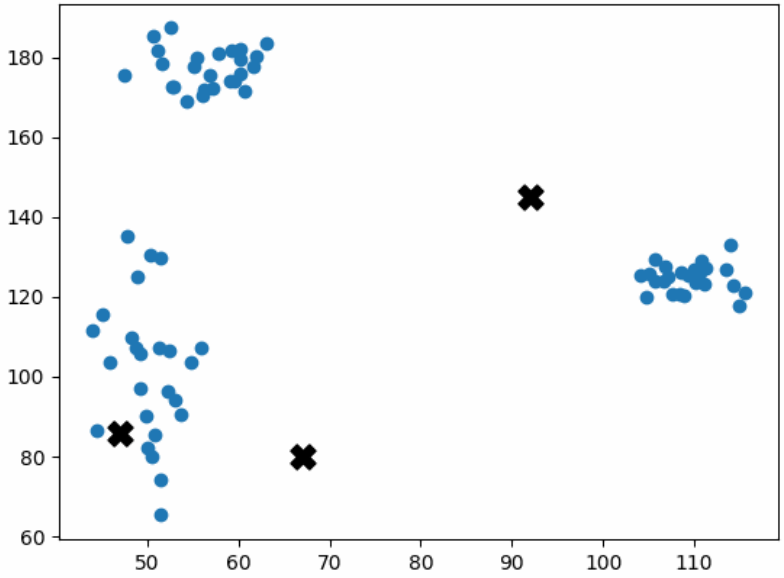


糟糕的起始点

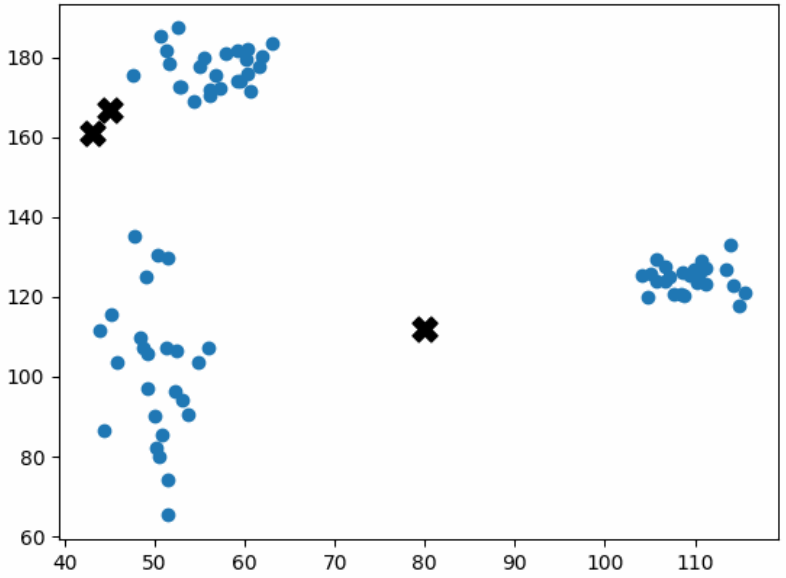


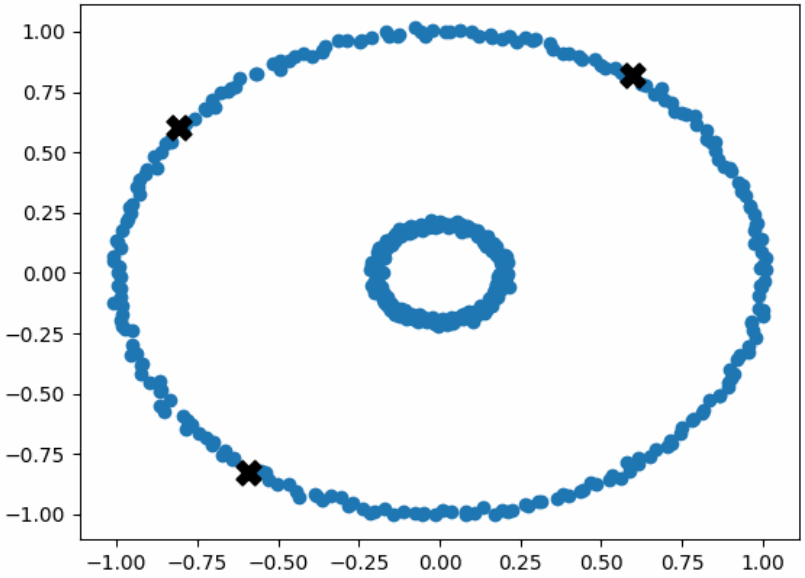
K-means聚类算法

K-means 示例



糟糕的起始点





不适合的场景

DBSCAN聚类算法

DBSCAN聚类算法

Density-Based Spatial Clustering of Applications with Noise

核心概念：

（1）核心点：如果一个对象在其半径Eps 内含有超过MPts 数目的点，则该对象为核心点

（2）边界点：如果一个对象在其半径 Eps 内有点的数量小于 MinPts，但是该对象落在核心点的邻域内，则该对象为边界点

（3）噪音点：如果一个对象既不是核心点也不是边界点，则该对象为噪音点

DBSCAN的参数：

（1）Eps：定义密度时的邻域半径

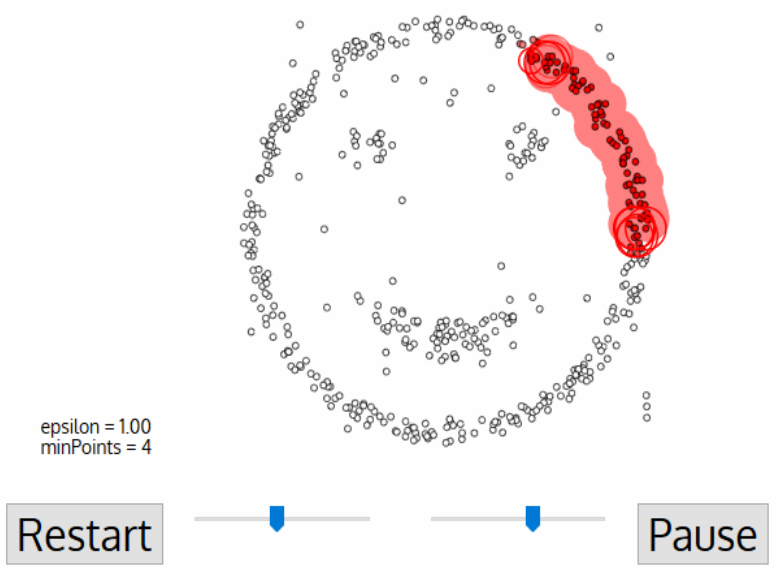
（2）MinPts：定义核心点时的阈值，即形成密集区域所需的最小点数

DBSCAN聚类算法

DBSCAN聚类算法步骤

DBScan需要**二个参数**： **扫描半径 (eps)**和**最小包含点数(minPts)**。

1. 任选一个未被访问(unvisited)的点开始，找出与其距离在eps之内(包括eps)的所有附近点。
2. 如果 **附近点的数量 ≥ minPts**，则当前点与其附近点**形成一个簇**，并且**出发点**被标记为**已访问**(visited)。 然后递归，以相同的方法处理该簇内所有未被标记为已访问(visited)的点，从而对簇进行扩展。
3. 如果 **附近点的数量 < minPts**，则该点暂时被标记作为**噪声点**。
4. 如果簇充分地被扩展，即簇内的所有点被标记为已访问，然后用同样的算法去处理未被访问的点。



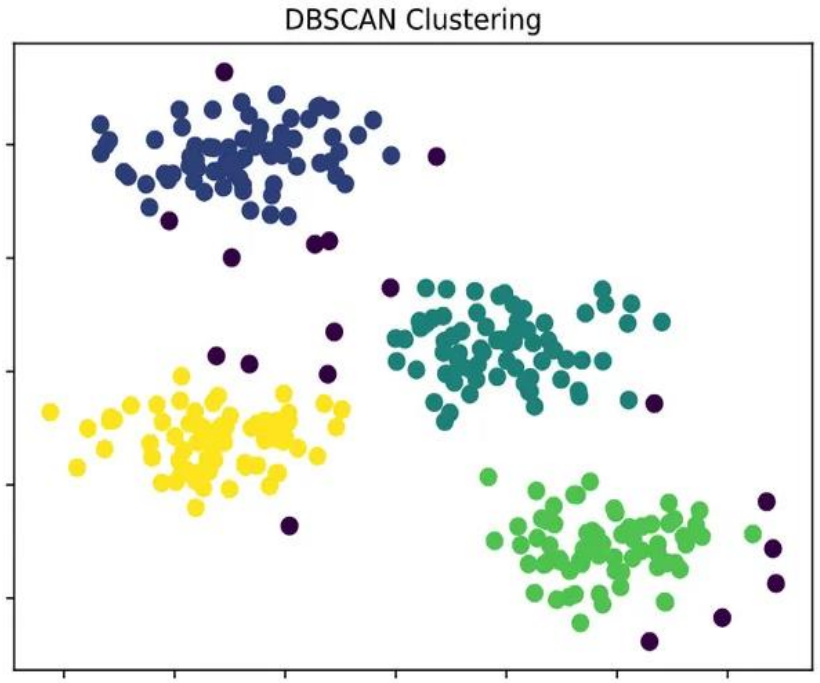
DBSCAN聚类算法

DBSCAN聚类算法python实现

使用 Python 的 sklearn 库中的 DBSCAN 类：

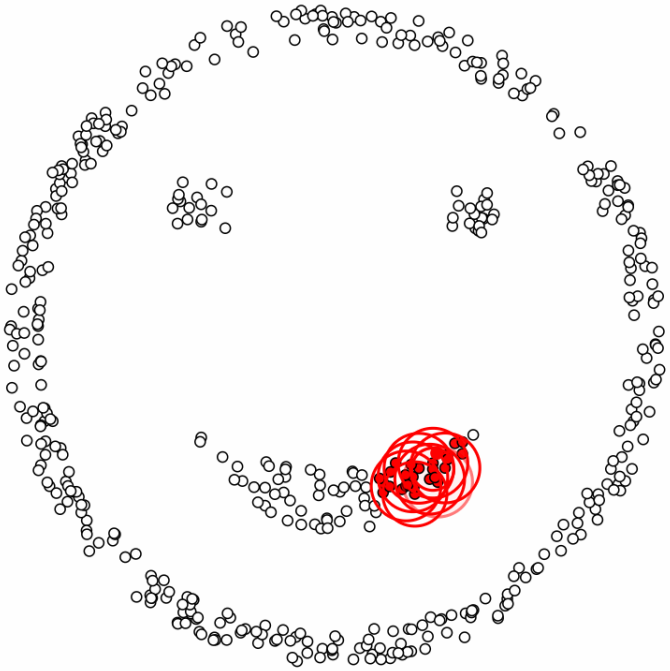
```
from sklearn.cluster import DBSCAN
import numpy as np
import matplotlib.pyplot as plt

# 再次使用之前的模拟数据
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# 应用DBSCAN算法
dbscan = DBSCAN(eps=0.5, min_samples=5)
clusters = dbscan.fit_predict(X)

# 绘制聚类结果
plt.scatter(X[:, 0], X[:, 1], c=clusters, cmap='viridis', marker='o', s=50)
plt.title("DBSCAN Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

DBSCAN 算法对模拟数据进行的聚类结果如下：

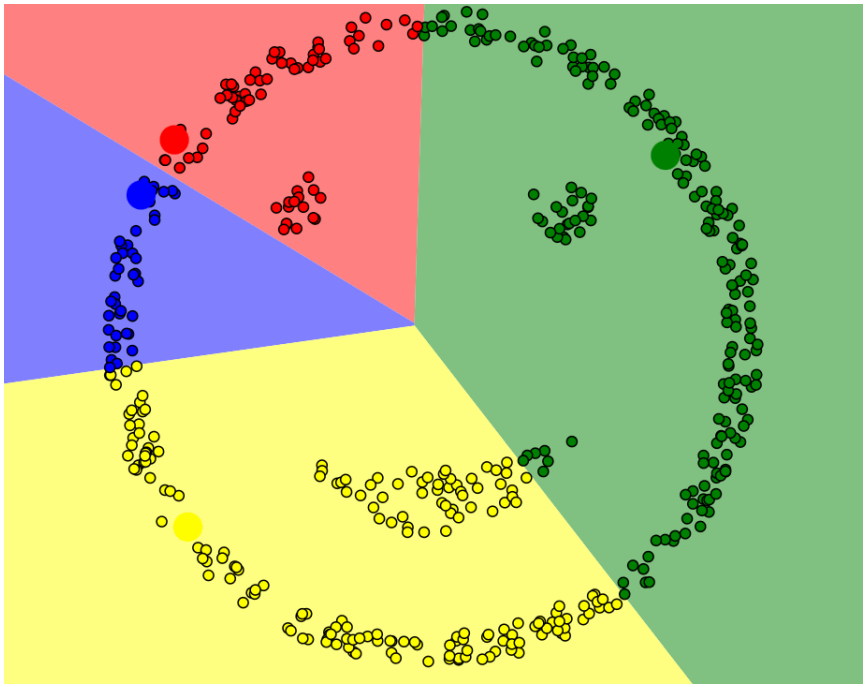


同种颜色的点属于同一个簇

DBSCAN聚类算法



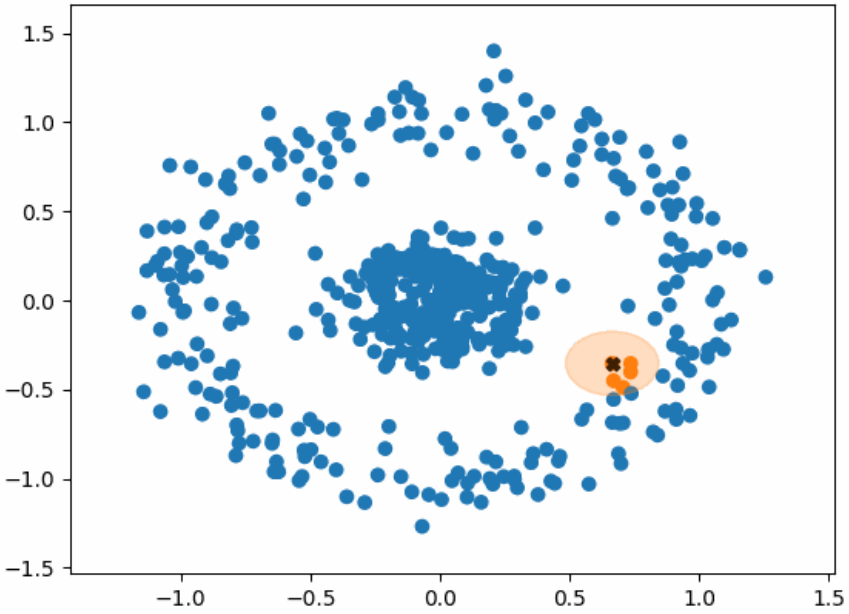
DBSCAN



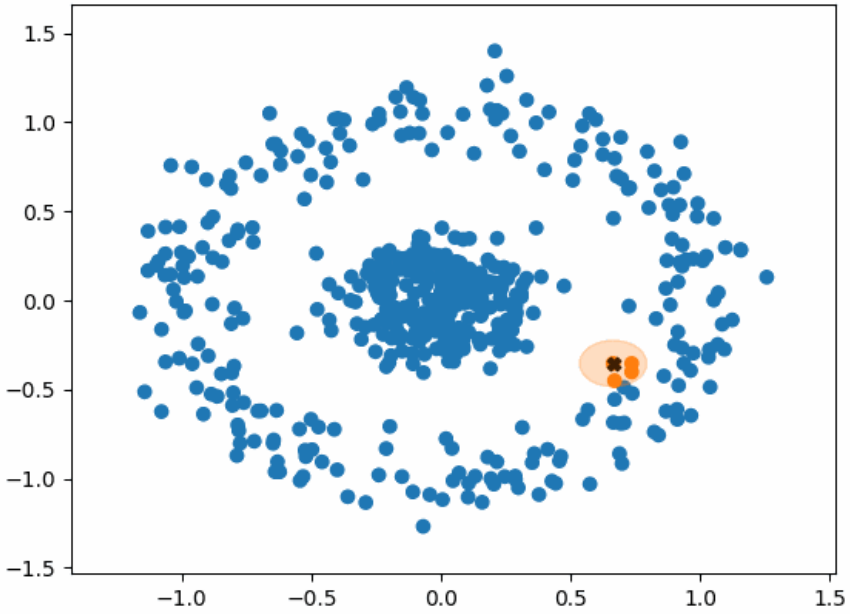
K-MEANS

DBSCAN聚类算法

DBSCAN 示例



过小的半径Eps



DBSCAN聚类算法

DBSCAN聚类算法优缺点总结

优点：

- DBSCAN比其他聚类算法，**不需要一个预设定的聚类数量**。它还将异常值识别为噪声，即使数据点非常不同，它也会将它们放入一个聚类中。
- DBSCAN能很好地找到**任意大小**和**任意形状**的聚类，对噪声数据具有良好的鲁棒性。
- DBSCAN结果对数据集样本的**随机抽样顺序不敏感**

缺点：

- 当**聚类具有不同的密度**时，它的性能不像其他聚类算法那样好。
- DBSCAN聚类算法在非常**高维的数据**中会因为**距离阈值ε**变得难以估计而效果差。

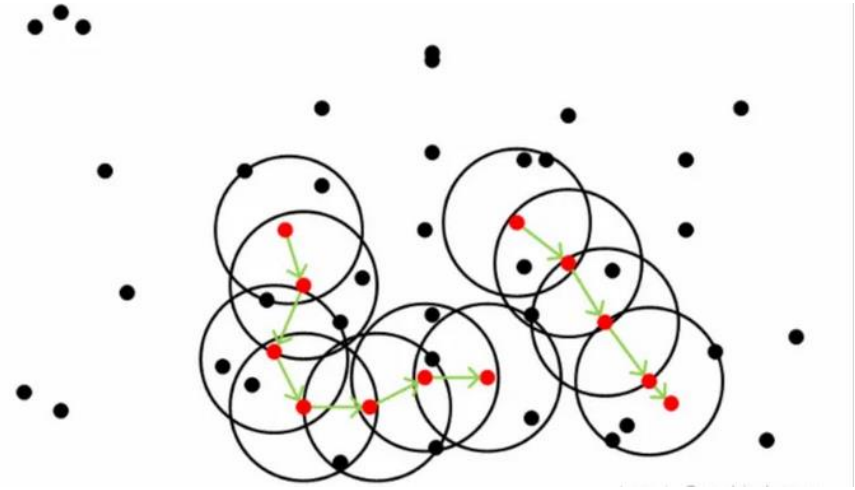
OPTICS聚类算法

相关概念：

OPTICS算法也是一种基于密度的聚类算法

OPTICS算法是DBSCAN算法的一种改进，核心对象、邻域等概念相同

与DBSCAN算法相比，在处理可变密度的数据集时更为有效



核心思想：

OPTICS算法通过分析数据点的**密度-可达性**来识别聚类结构。

OPTICS聚类算法

算法流程

1.初始化核心对象集合Ω=Ø

2.遍历X的元素，如果是核心对象，则将其加入到核心对象集合 Ω 中。

3.如果核心对象集合Ω中元素都已经被处理，则算法结束，否则转入步骤4。

4.在核心对象集合Ω中，随机选择一个未处理的核心对象o，首先将o标记为已处理，同时将o压入到有序列表 p 中，根据可达距离的大小 ，依次存放到种子集合 seeds 中。

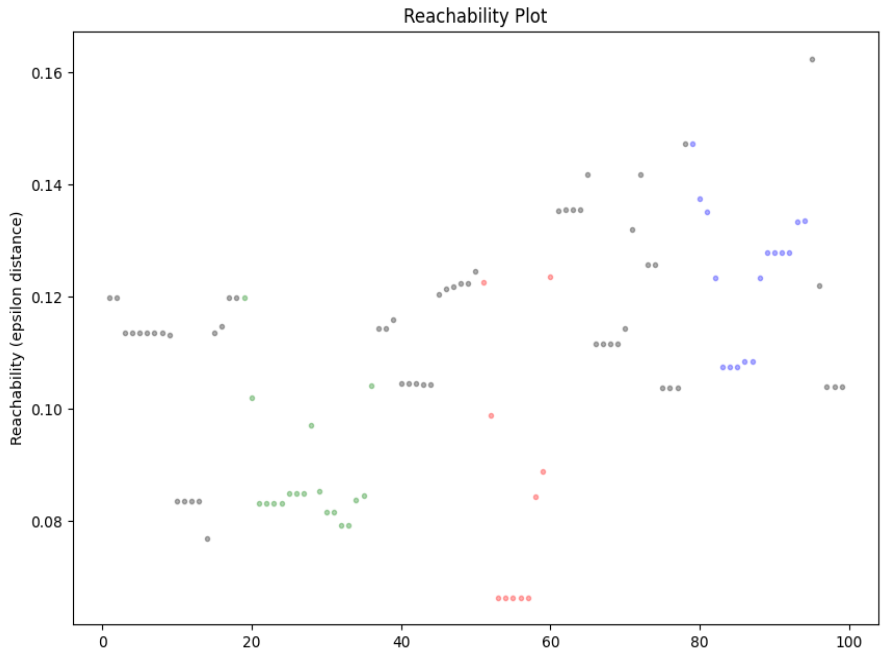
5.如果种子集合 seeds =Ø，跳转到3，否则，从种子集合 seeds 中挑选可达距离最近的种子点 seed，将其标记为已访问，将 seed 标记为已处理，同时将 seed 压入到有列表p中，判断 seed 是否为核心对象，将 seed 中未访问的邻居点加入到种子集合中重新计算可达距离。

OPTICS聚类算法

Python代码示例

使用sklearn库中的OPTICS类来实现OPTICS算法

```python
from sklearn.cluster import OPTICS
import matplotlib.pyplot as plt
import numpy as np
# 示例数据
X = np.random.rand(100, 2)
# OPTICS模型
optics_model = OPTICS(min_samples=5, xi=0.05, min_cluster_size=0.1)
optics_model.fit(X)
# 可视化
space = np.arange(len(X))
reachability = optics_model.reachability_[optics_model.ordering_]
labels = optics_model.labels_[optics_model.ordering_]

plt.figure(figsize=(10, 7))
colors = ['g.', 'r.', 'b.', 'y.', 'c.']
for klass, color in zip(range(0, 5), colors):
    Xk = space[labels == klass]
    Rk = reachability[labels == klass]
    plt.plot(Xk, Rk, color, alpha=0.3)
plt.plot(space[labels == -1], reachability[labels == -1], 'k.', alpha=0.3)
plt.ylabel('Reachability (epsilon distance)')
plt.title('Reachability Plot')
plt.show()
```



OPTICS聚类算法