密度聚类算法实现

一、问题描述：

实现DBSCAN算法。

1. 假设EPS邻域内样本个数最小值MinPts = 4，邻域半径EPS自行确定参数。
2. 自行设计生成数据集，要求有两个半月形的簇组成，两个簇之间有间隙，每个半月100个数据点，共200个数据点，要求结果不同的簇用不同的颜色表示。
3. 算法思路
4. 首先生成月牙形的数据集，并且将数据集以图的形式展示在二维的坐标轴上。
5. 然后每个点按照K近邻的距离进行排序，以折线的形式显示在二维坐标轴上，根据图确定EPS。
6. 实现DBSCAN算法将第二部第一步所产生的数据集进行分类，并将分类的结果以图的形式现实在二位坐标轴上，每个分类簇都对应不同的颜色。
7. 算法实现

算法采用python程序设计语言是实现。

1.月牙数据集的生成采用的是机器学习sklearn库下datasets数据集模块的make\_moons()方法来实现：

X,y = make\_moons(n\_samples=200, noise=0.09)

第一个参数n\_numbers代表生成样本数量，第二个参数noise代表加到数据里面的高斯噪声的标准差，返回值是样本数据X和样本分类结果x；产生数据集以后通过基于python的绘图库matplotlib下的pyplot包函数将数据集以图的形式展示在二维的坐标轴上。

1. 在计算一个点的k近邻距离时，首先要计算该点到其他所有点的欧氏距离，按照距离升序排序后，选择第k小的距离作为k-距离的值，再将所有保存的k-距离的值进行排序，给排序后的数值标上标签，按照横坐标为k近邻排序点的个数，纵坐标为k近邻的距离显示在展示在二维坐标轴上，根据坐标轴中去曲线，找出曲线的拐点即为对应的EPS。

输入：数据集dataset，k

输出：k-距离曲线图

For i in dataset：

构建dataset每个点的k距离矩阵dist\_Matrix

For j in dist\_Matrix：

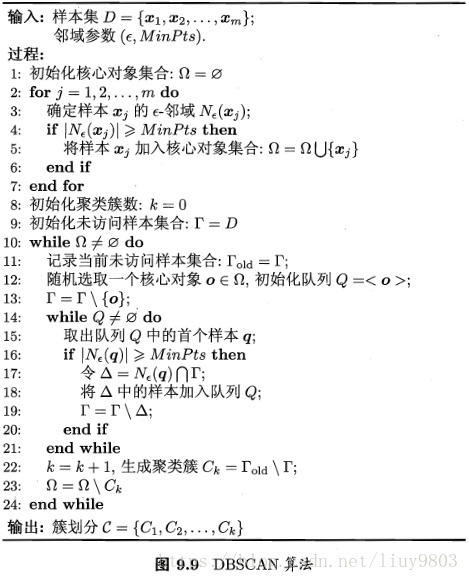
j.sort() #对每个点的k距离排序

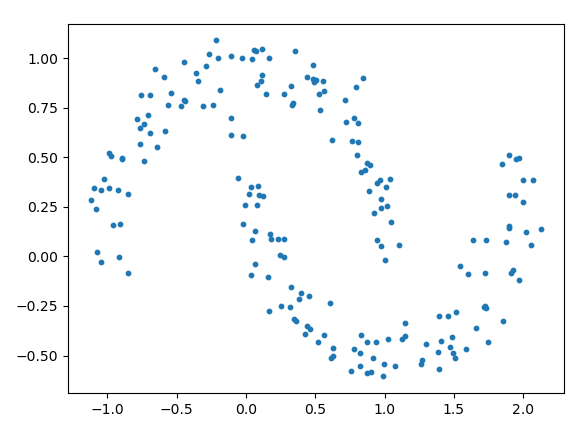
选择第k小的距离作为k-距离的值存储在dist\_list

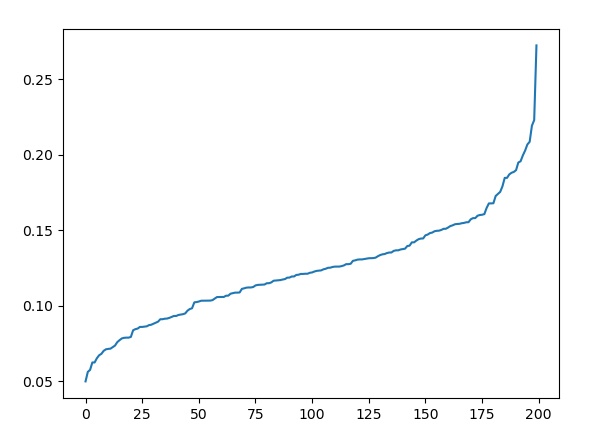
dist\_list.sort() #对k-距离的值进行排序

plt.show() #展示曲线图

1. DBSCAN算法的完成需要使用到第一步产生的数据集和第二部找到的EPS以及实现假定的MinPts，DBSCAN算法先任选数据集中的一个核心对象作为种子，创建一个簇并找出它所有的核心对象，寻找合并核心对象密度可达的对象，直到所有核心对象均被访问过为止。 DBSCAN的簇中可以少包含一个核心对象：如果只有一个核心对象，则其他非核心对象都落在核心对象的EPS-邻域内；如果有多个核心对象，则任意一个核心对象的EPS-邻域内至少有一个其他核心对象，否则这两个核心对象无法密度可达；包含过少对象的簇可以被认为是噪音。算法过程如下：



1. 结果展示
2. 生成的数据集展示
3. 生成的k近邻距离曲线图展示 



3.算法分类解雇展示

