第四次大作业 - 密度聚类

# 一、基于密度的聚类

聚类分析又称群分析，它是研究分类问题的一种统计分析方法，同时也是数据挖掘的一个重要算法。

基于密度聚类 (Density-Based Clustering)的主要特点:

* 发现任意形状的聚类
* 处理噪音
* 一遍扫描
* 需要密度参数作为终止条件

两个参数:

* Eps: 邻域的最大半径
* MinPts: 在 Eps-邻域中的最少点数

直接密度可达的(Directly density reachable, DDR): 给定对象集合D, 如果p是在q的–邻域内, 而q是核心对象, 我们说对象p是从对象q直接密度可达的(如果q是一个核心对象，p属于q的邻域，那么称p直接密度可达q。)

密度可达的(density reachable): 存在 一个从p到q的DDR对象链(如果存在一条链<p1,p2,…..,pi>，满足p1=p，pi=q，pi直接密度可达pi+1，则称p密度可达q)

密度可达: 点 p 关于Eps, MinPts 是从 q密度可达的, 如果 存在一个节点链 p1, …, pn, p1 = q, pn = p 使得 pi+1 是从pi直接密度可达的

密度相连的:点 p关于 Eps, MinPts 与点 q是密度相连的, 如果 存在点 o 使得, p 和 q 都是关于Eps, MinPts 是从 o 密度可达的(如果存在o，o密度可达q和p，则称p和q是密度连通的)

核心点条件: **|*NEps (q)*| >= *MinPts***

# 二、DBSCAN

DBSCAN(Density-Based Spatial Clustering of Application with Noise)是一种典型的基于密度的聚类算法，在DBSCAN算法中将数据点分为一下三类：

* 核心点：在半径Eps内含有超过MinPts数目的点
* 边界点：在半径Eps内点的数量小于MinPts，但落在核心点的邻域内
* 噪音点：既不是核心点也不是边界点的点

相比其他的聚类方法，基于密度的聚类方法可以在有噪音的数据中发现各种形状和各种大小的簇。DBSCAN（Ester, 1996）是该类方法中最典型的代表算法之一。其核心思想就是先发现密度较高的点，然后把相近的高密度点逐步都连成一片，进而生成各种簇。算法实现上就是，对每个数据点为圆心，以eps为半径画个圈（称为邻域eps-neigbourhood），然后数有多少个点在这个圈内，这个数就是该点密度值。然后我们可以选取一个密度阈值MinPts，如圈内点数小于MinPts的圆心点为低密度的点，而大于或等于MinPts的圆心点高密度的点（称为核心点Core point）。如果有一个高密度的点在另一个高密度的点的圈内，我们就把这两个点连接起来，这样我们可以把好多点不断地串联出来。之后，如果有低密度的点也在高密度的点的圈内，把它也连到最近的高密度点上，称之为边界点。这样所有能连到一起的点就成一了个簇，而不在任何高密度点的圈内的低密度点就是异常点。

# 三、算法流程

输入：

1. epsilon半径
2. MinPts：给定点在epsilon邻域内成为核心对象的最小领域点数
3. D：集合

输出：目标类簇集合

方法： repeat

1. 判断输入点是否为核心对象
2. 找出核心对象的epsilon邻域中的所有直接密度可达点，直到所有输入点都判断完毕
   1. Repeat
   2. 针对所有核心对象的epsilon邻域所有直接密度可达点找到最大密度相连对象集合，中间涉及到一些密度可达对象的合并。
   3. 直到所有核心对象的epsilon邻域都遍历完毕

DBSCAN的复杂度：采用空间索引, 复杂度为O(nlog n), 否则为O(n2)

DBSCAN的缺点:对用户定义的参数是敏感的, 参数难以确定(特别是对于高维数据), 设置的细微不同可能导致差别很大的聚类。

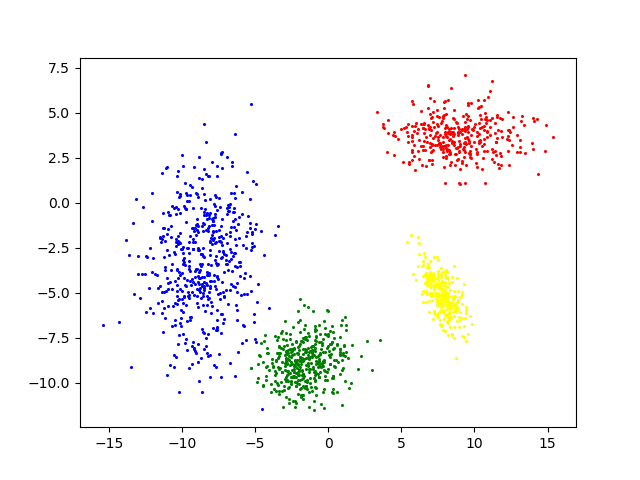
# 四、实验结果

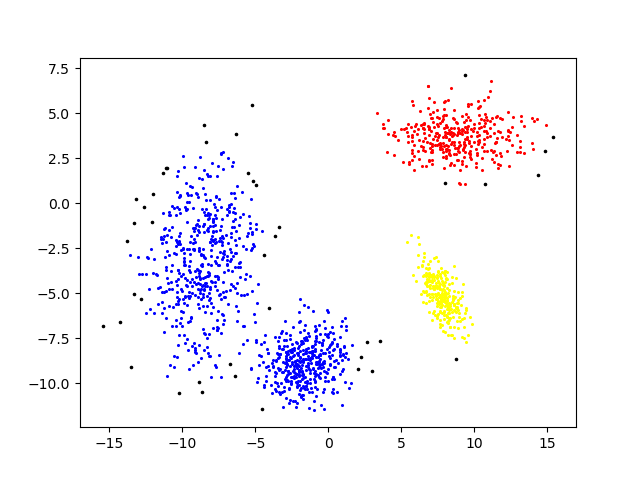
实验环境：Python3

总共测试了8个数据集。对于每个数据集，给出两幅图：第一幅图展示原始数据分布，用不同的颜色表示不同的类别；第二幅展示划分之后的效果图，同时在图下给出以下参数值：

* Epsilon:
* MinPit:
* 总样本数:
* 核节点数:
* 划分类别数:
* 孤立节点数： (黑点表示)
* 耗时:

## 1.数据a





Epsilon: 0.8

MinPit: 8

总样本数: 1572

核节点数: 1347.0

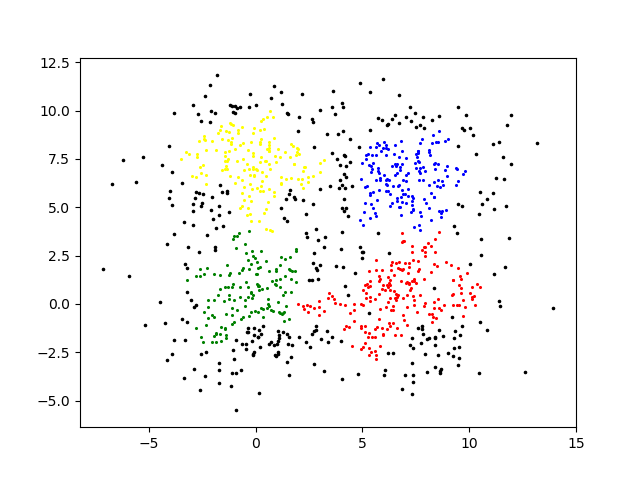
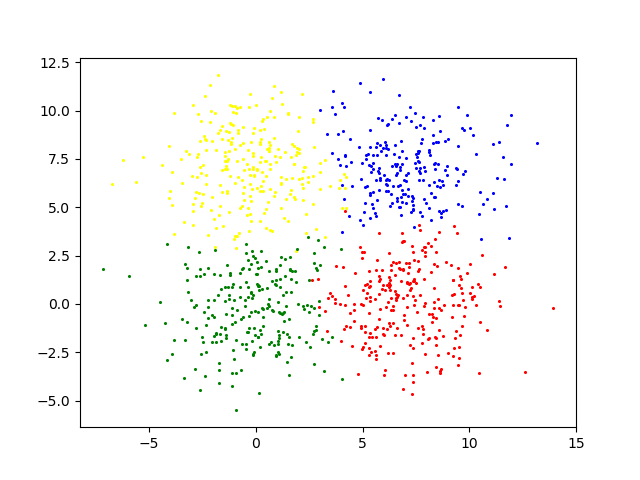
划分类别数: 3

孤立节点数： 43 (黑点表示)

正确率: 0.7061068702290076

耗时: 166.53283858299255 s

## 2.数据b



Epsilon: 0.5

MinPit: 10

总样本数: 1000

核节点数: 56.0

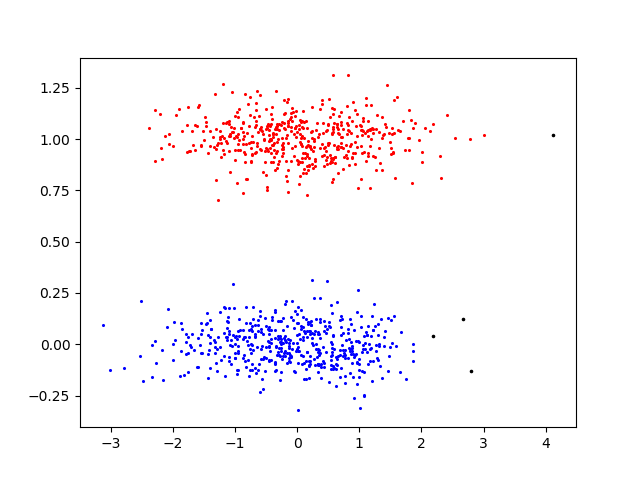
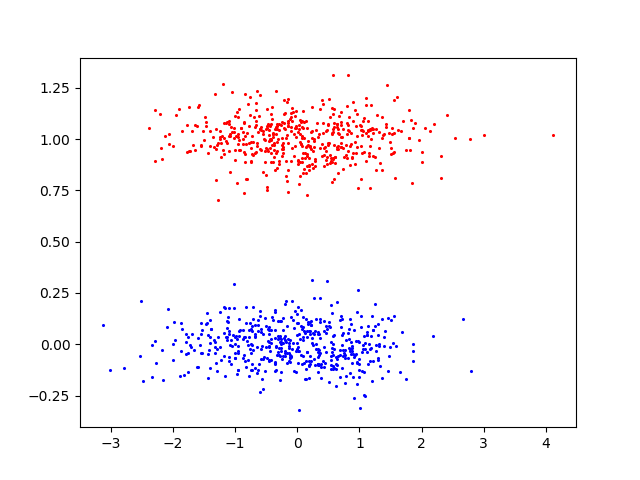
划分类别数: 4

孤立节点数： 339 (黑点表示)

正确率: 0.643

耗时: 9.575685024261475 s

## 3. 数据long1



Epsilon: 0.3

MinPit: 8

总样本数: 1000

核节点数: 980.0

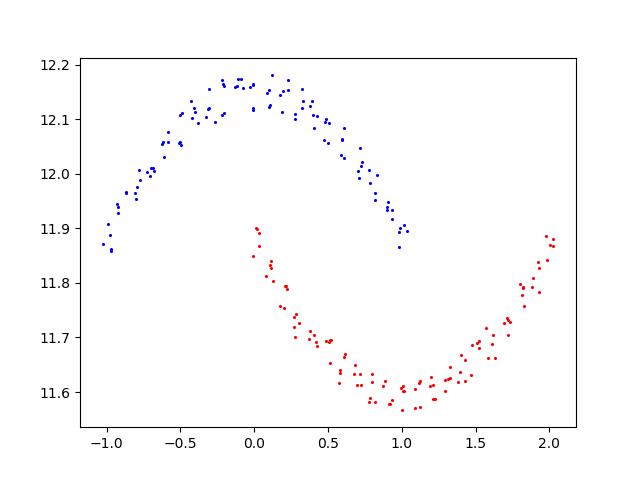
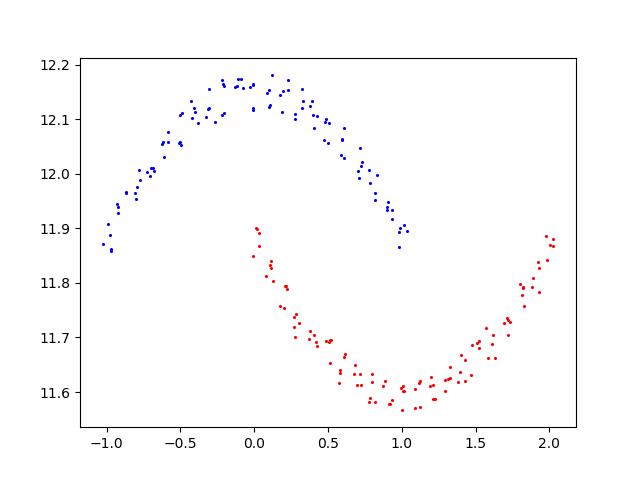
划分类别数: 2

孤立节点数： 4 (黑点表示)

正确率: 0.996

耗时: 213.63904762268066 s

## 4.数据moon



Epsilon: 0.15

MinPit: 4

总样本数: 210

核节点数: 210.0

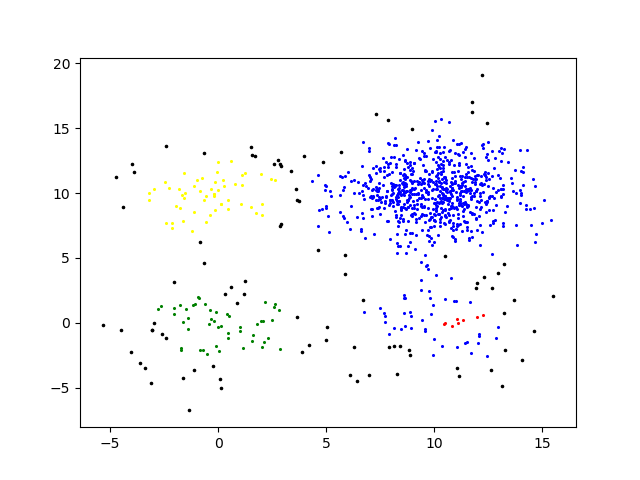
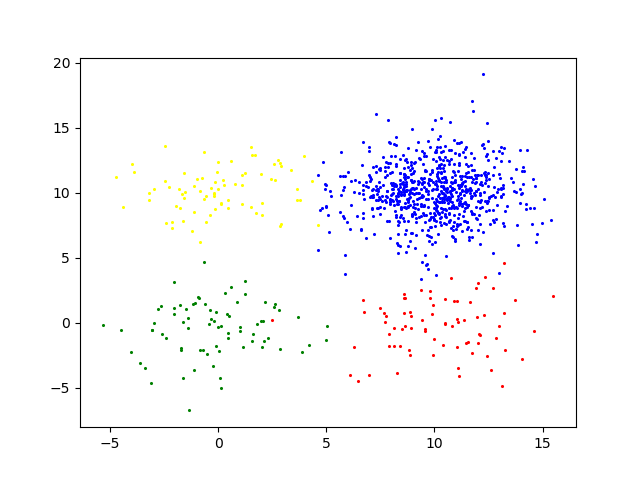
划分类别数: 2

孤立节点数： 0 (黑点表示)

正确率: 1.0

耗时: 1.620103120803833 s

## 5.数据sizes5



Epsilon: 0.91

MinPit: 5

总样本数: 1000

核节点数: 779.0

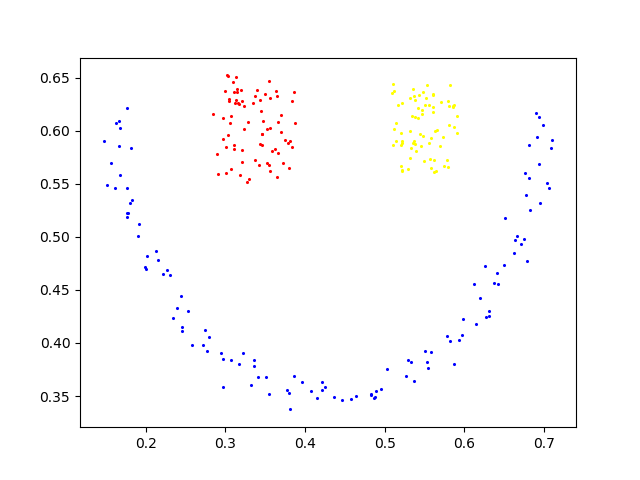
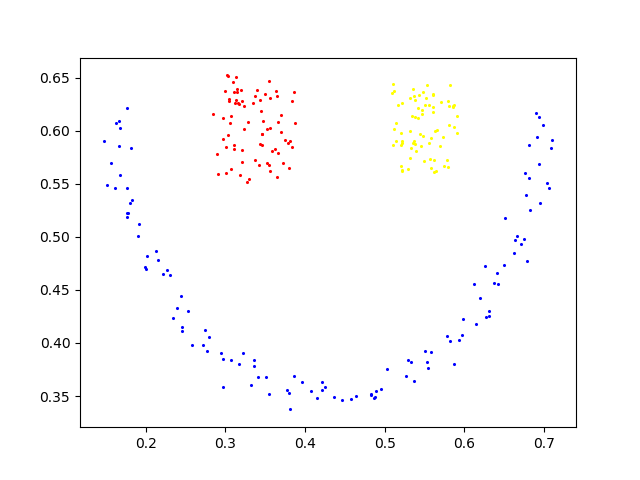
划分类别数: 4

孤立节点数： 90 (黑点表示)

正确率: 0.865

耗时: 68.85463666915894 s

## 6.数据smile



Epsilon: 0.05

MinPit: 3

总样本数: 266

核节点数: 266.0

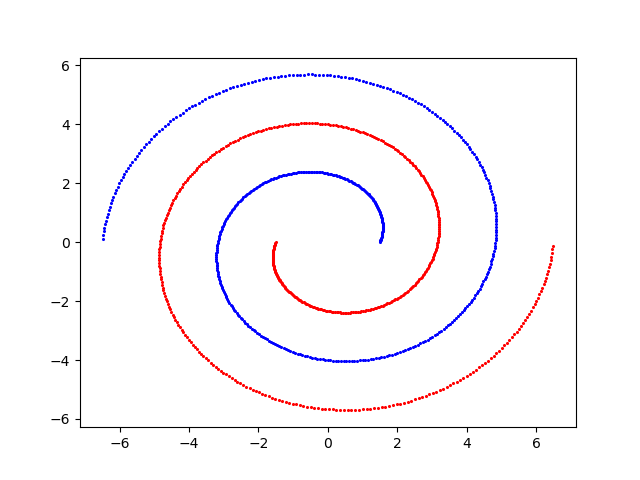
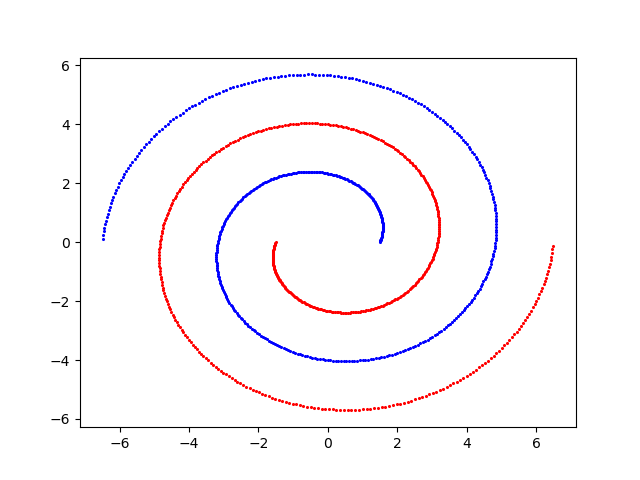
划分类别数: 3

孤立节点数： 0 (黑点表示)

正确率: 1.0

耗时: 5.500326871871948 s

## 7.数据spiral



Epsilon: 0.5

MinPit: 10

总样本数: 1000

核节点数: 752.0

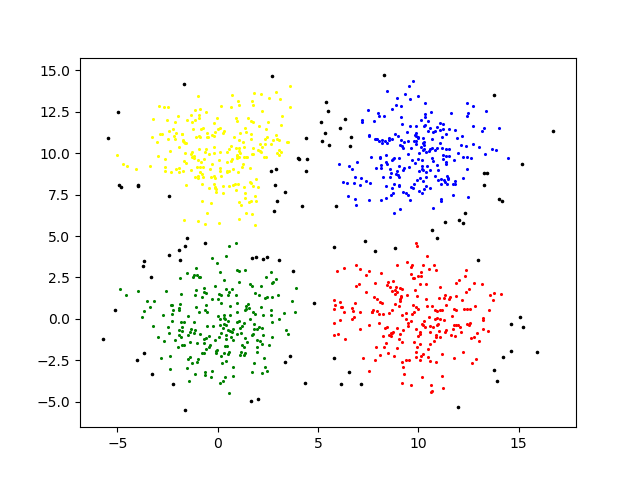
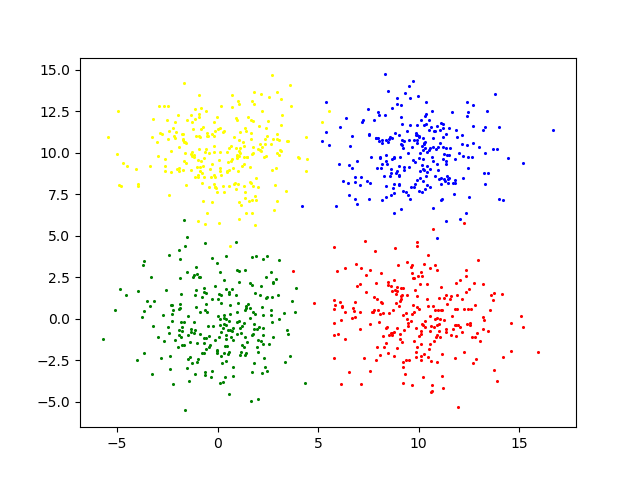
划分类别数: 2

孤立节点数： 0 (黑点表示)

正确率: 1.0

耗时: 27.96483278274536 s

## 8.数据square1



Epsilon: 0.8

MinPit: 5

总样本数: 1000

核节点数: 752.0

划分类别数: 4

孤立节点数： 93 (黑点表示)

正确率: 0.905

耗时: 26.139179944992065 s