**哈尔滨工业大学**

**数据挖掘理论与算法实验报告**

**(2015年度秋季学期)**

**课程编码** S1300019C

**授课教师 邹兆年**

**学生姓名 白辰甲**

**学 号 15S003002**

**学 院 计算机科学与技术**

**一、实验内容**

**1.问题：**使用DBscan算法聚类。

**2.目标：**使用具有一定形状的具有密度特征的簇，使用Dbscan算法根据用户设置的Eps和MinPts参数完成聚类，可以标记噪声点，并把核心点和边界点聚类。

**3.算法：**

|  |  |
| --- | --- |
|  | DBscan算法 |
| 1 | 将所有的点标记为核心点、边界点或噪声点 |
| 2 | 删除噪声点 |
| 3 | 为距离在Eps之内的所有核心点之间赋一条边。 |
| 4 | 每组连通的核心点形成一个簇。 |
| 5 | 将每个边界点指派到一个与之关联的核心点的簇中。 |

**二、实验设计**

**1.程序流程图**



**2.数据结构**

存储由数据点和与数据点间距在Eps之内的邻近点构成的图，使用字典模拟邻接表。

其中字典的键为点的标号，值记录与该点距离在eps范围内的点的标号。 首先初始化邻接表中每个节点的邻近节点为空，随后遍历测试数据集，如果两个点之间的距离在Eps之内，在邻接表中这两个点对应的邻近节点序列添加该点。

构造该图的代码如下：

|  |
| --- |
| dis\_eps = {}  for i in range(n):  dis\_eps[i] = []  for i in range(n):  for j in range(i+1, n):  distance = sqrt(pow((points[i][0] - points[j][0]), 2) + pow((points[i][1] - points[j][1]), 2))  if distance < eps:  dis\_eps[i].append(j)  dis\_eps[j].append(i) |

**3.关键函数及说明**

（1）构造图

代码如上，首先初始化邻接表中每个节点的邻近节点为空，随后遍历测试数据集，如果两个点之间的距离在Eps之内，在邻接表中这两个点对应的邻近节点序列添加该点。

（2）将所有点分成核心点、边界点和噪声点。

如果某个点的邻近节点的个数超过minPts，则该点为核心点。如果该点不是核心点，且该点的邻近节点内含有核心点，则该点是边界点。则该点是噪声点。

|  |
| --- |
| #记录该点是 1：核心点 2：边界点 3：噪声点  core\_point, border\_point, noise\_point = [], [], []  for i in range(n):  if len(dis\_eps[i]) >= minpts:  core\_point.append(i) # 核心点  for i in range(n):  if i not in core\_point:  if len(list(set(dis\_eps[i]).intersection(set(core\_point)))) != 0: # 如果该点的邻近点有核心点  border\_point.append(i) # 边界点  else:  noise\_point.append(i) # 噪声点 |

（3）广度优先遍历，将核心点分割成簇。

首先初始化队列和标记数组，循环，如果该点未被访问则进入队列。对于队列中的每个点，出队列表示该点被访问，查找该点邻近的未被访问的核心点，加入队列中。直到队列为空则停止。表示一个联通分支已被访问完毕。该联通分支中的点则构成了一个簇。

|  |
| --- |
| for p in core\_point: # 将核心点分割成簇  newlist = []  if flag\_list[p] == 0:  queue.append(p)  while len(queue) != 0:  p1 = queue.pop(0)  flag\_list[p1] = 1  newlist.append(p1)  for p2 in dis\_eps[p1]:  if (flag\_list[p2] == 0) and (p2 not in queue) and (p2 in core\_point):  queue.append(p2)  if len(newlist) != 0:  clusters.append(newlist)  print "簇的个数 ", len(clusters) |

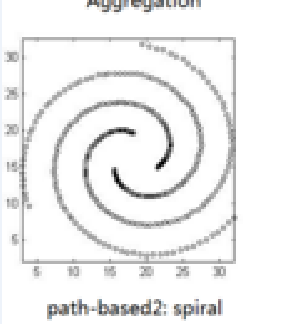
**三、测试数据**

**1.数据集描述**

实验数据集1如图所示：



实验数据集2如图所示：



**2.数据样例（共8000个点）**

样例如表所示：

|  |
| --- |
| 68.601997 102.491997 |
| 454.665985 264.808990 |
| 101.283997 169.285995 |
| 372.614990 263.140991 |
| 300.989014 46.555000 |

**3.预处理过程**

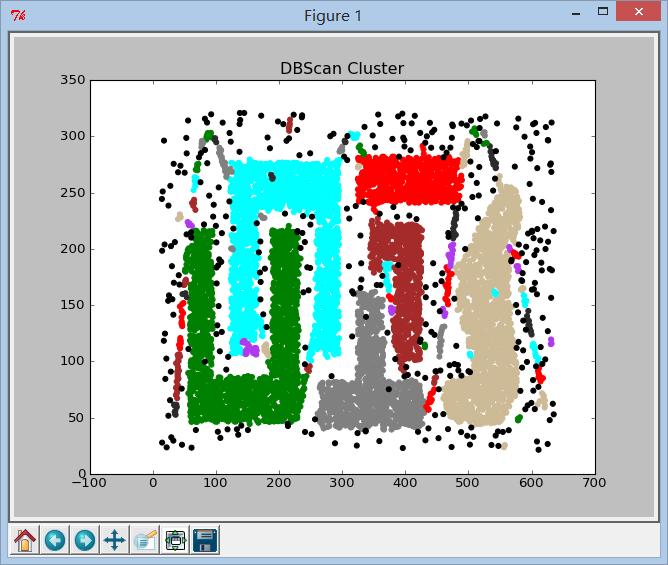
数据质量良好，只需要将两个方向的坐标分割即可。

|  |
| --- |
| for line in fp.readlines():  line = line.strip()  point = re.split(r'\s+', line)  #print point[:]  points.append((float(point[0]), float(point[1]))) |

**四、实验结果**

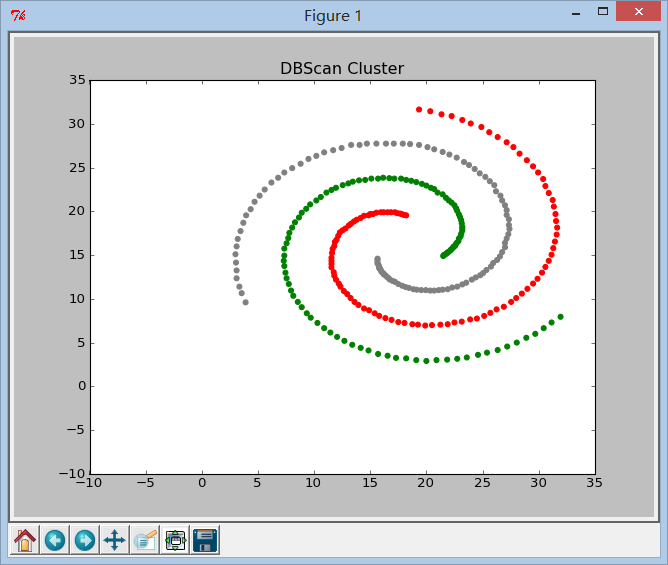
1.实验数据集1

聚类效果如下图所示，其中黑色点为噪声点。



2.实验数据集2

聚类效果如下图所示，没有噪声点。



**五、遇到的困难及解决方法、心得体会**

Dbscan在基于密度的簇的聚类中表现很好，而且能够处理噪声。但是Dbscan需要计算所有点之间的临近度，因而开销较大。同时，对用户指定的参数Eps和minPts较为敏感。