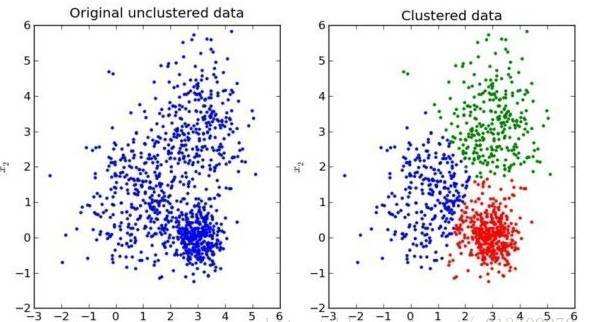
作业一：DBSCAN算法的实现

一、作业内容：

在二维空间内随机创建**300**个数据对象构建数据集D，每个维度的取值范围为[0,10]，设置**三组**不同的的(Eps，Minpts)值，使用DBSCAN算法对数据集D进行聚类分析。

作业要求：输出原始数据对象分布和聚类分布。可参考下图，不同聚类可用不同符号标注，如+，○，□，△等。

**在文档P3-P5处填写相应内容，交打印版报告。**



二、DBSCAN算法

**1.主要概念**

**Ε邻域**：给定对象半径为Ε内的区域称为该对象的Ε邻域；

**核心对象**：如果给定对象Ε领域内的样本点数大于等于MinPts，则称该对象为核心对象；

**直接密度可达**：对于样本集合D，如果样本点q在p的Ε领域内，并且p为核心对象，那么对象q从对象p直接密度可达。

**密度可达**：对于样本集合D，给定一串样本点p1,p2….pn，p= p1,q= pn,假如对象pi从pi-1直接密度可达，那么对象q从对象p密度可达。

**密度相连**：存在样本集合D中的一点o，如果对象o到对象p和对象q都是密度可达的，那么p和q密度相联。

**2.算法执行步骤**

（1）检测数据库中尚未检查过的对象***p***，如果***p***未被处理(归为某个簇或者标记为噪声)，则检查其邻域，若包含的对象数不小于minPts，建立新簇***C***，将其中的所有点加入候选集***N***；

（2）对候选集***N***中所有尚未被处理的对象***q***，检查其邻域，若至少包含minPts个对象，则将这些对象加入N；如果***q***未归入任何一个簇，则将***q***加入***C***；

（3）重复步骤2)，继续检查***N***中未处理的对象，当前候选集***N为空***；

（4）重复步骤1)~3)，直到所有对象都归入了某个簇或标记为噪声。

3.[**伪代码**](https://baike.baidu.com/item/%E4%BC%AA%E4%BB%A3%E7%A0%81)**描述如下：**

输入：[数据对象](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AF%B9%E8%B1%A1)集合D，半径Eps，密度阈值MinPts

输出：聚类C

DBSCAN（D, Eps, MinPts）

Begin

init C=0; //初始化簇的个数为0

for each unvisited point p in D

mark p as visited; //将p标记为已访问

N = getNeighbours (p, Eps);

if sizeOf(N) < MinPts then

mark p as Noise; //如果满足sizeOf(N) < MinPts，则将p标记为噪声

else

***C***= next cluster; //建立新簇***C***

ExpandCluster (p, N, C, Eps, MinPts);

end if

end for

End

**其中ExpandCluster算法伪码如下：**

ExpandCluster(p, N, C, Eps, MinPts)

add p to cluster C; //首先将核心点加入***C***

for each point p’ in N

mark p' as visited;

N’ = getNeighbours (p’, Eps); //对N邻域内的所有点在进行半径检查

if sizeOf(N’) >= MinPts then

N = N+N’; //如果大于MinPts，就扩展N的数目

end if

if p’ is not member of any cluster

add p’ to cluster C; //将p' 加入簇***C***

end if

end for

End ExpandCluster

**《数据分析方法》**

**作业一：DBSCAN算法的实现**

**学号：** 1514010316

**班级：** 软件15-3

**姓名：** 王炳炎

作业一：DBSCAN算法的实现

**一、程序运行结果**

1.原始数据集数据分布如图1所示。

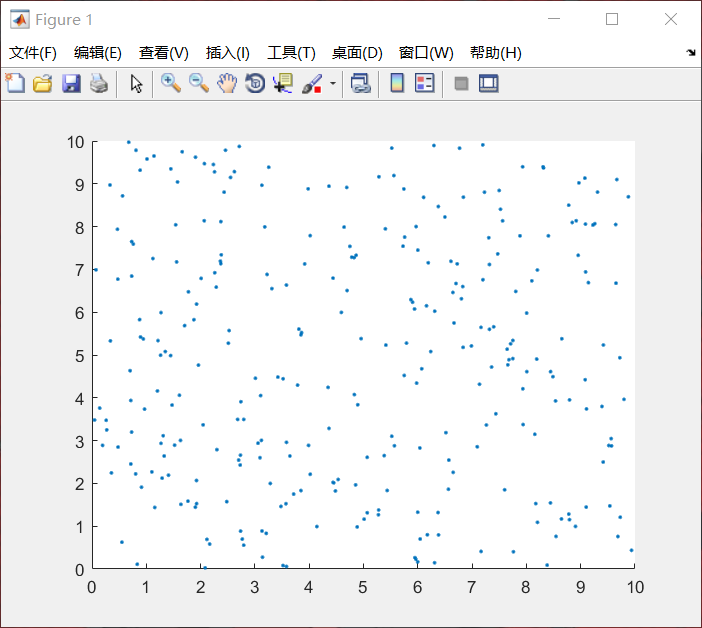


图1 原始数据集数据分布图

原始数据集共有300个数据点，随机分布在 10\*10的二维空间中。

2.三组不同Eps，Minpts值下聚类结果。

（1）Eps=1.0，Minpts=10时，聚类结果如图2所示。

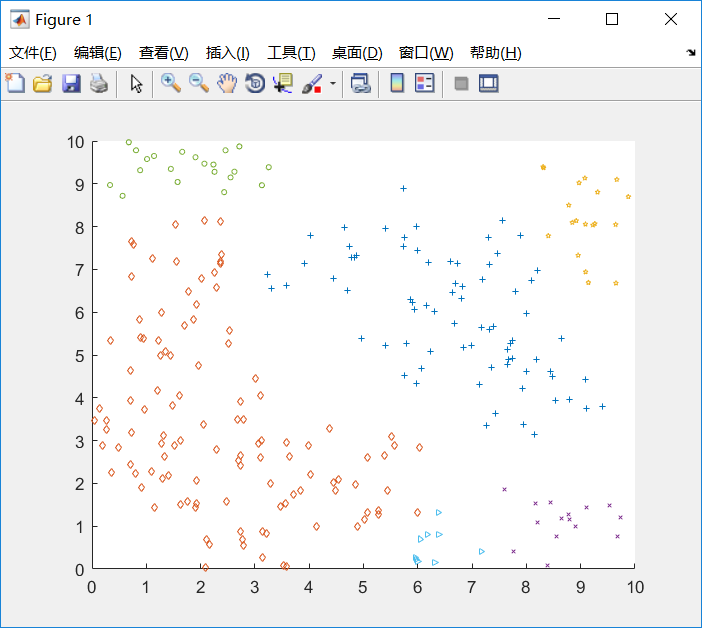


图2 Eps=1.0，Minpts=10时，聚类结果图

当Eps=1.0，Minpts=10时，共生成6个簇，其中各个簇的散点个数分别为76，114，19，15，21，10。还包括45个离群点。

（2）Eps=1.1，Minpts=10时，聚类结果如3所示。

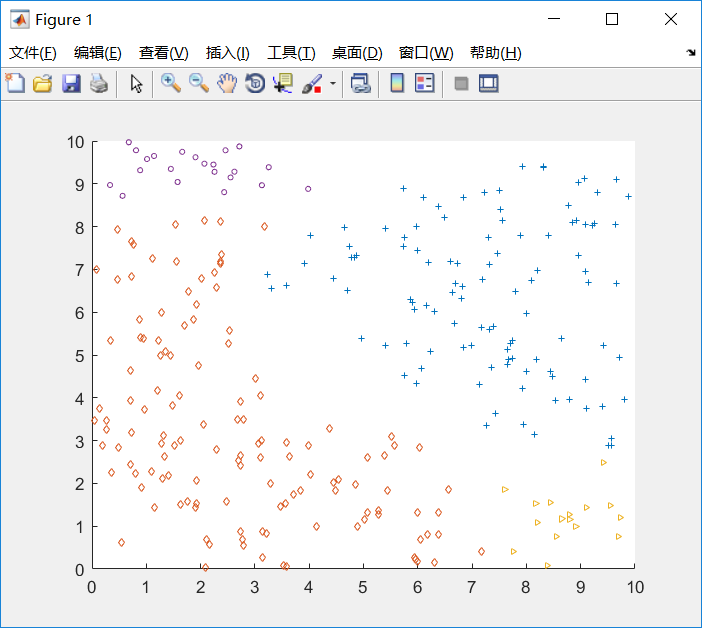


图3 Eps=1.1，Minpts=10时，聚类结果图

当Eps=1.1，Minpts=10时，共生成4个簇，其中各个簇的散点个数分别为109，129，16，22。还包括24个离散点。

（3）Eps=1.0，Minpts=13时，聚类结果如图4所示。

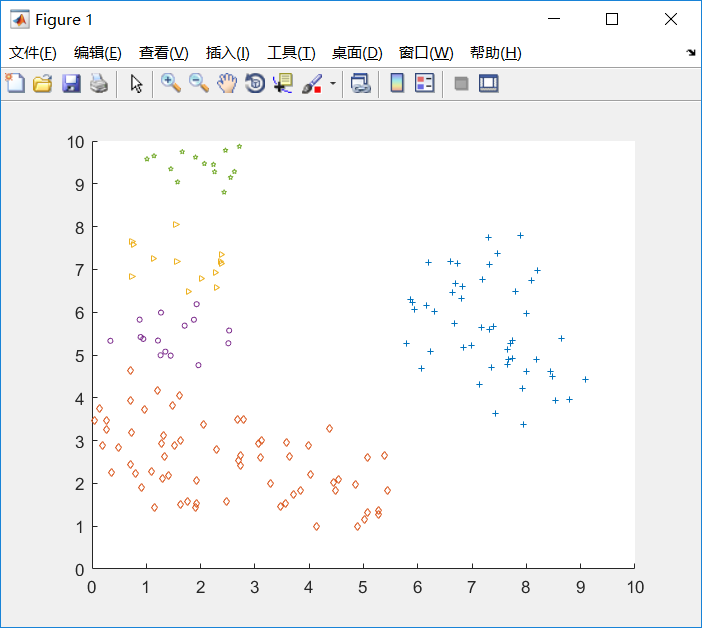


图4 Eps=1.0，Minpts=13 时，聚类结果图

当Eps=1.0，Minpts=13时，共生成4个簇，其中各个簇的散点个数分别为109，129，16，22。还包括24个离散点。

**二、聚类结果分析**

1.随Eps值增大时，簇的个数减少，每个簇包含的数据元素的个数越多。

2.随Minpts值增大时，簇的个数减少，数据集在高密度集数据时，每个簇的元素个数会增加，在低密度集数据时，每个簇的元素个数会减少。

**三、程序主要代码**

**Main.m (主程序代码)**

1. function [ output\_args ] = main( )
2. %UNTITLED1 Summary of **this** function goes here
3. %  Detailed explanation goes here
4. global cluster;
5. global len;
6. len = 300;
7. Eps = 1.0;
8. global MinPts;
9. MinPts = 16;
10. count = 1;
11. % data1 = rand(1,len)
12. % data2 = rand(1,len)
13. % save x.txt -ascii data1
14. % save y.txt -ascii data2
15. data1=load ('x.txt')\*10;
16. data2=load ('y.txt')\*10;
17. %计算距离矩阵
18. dis\_mat=zeros(len,len);
19. **for** i=1:len
20. **for** j=i:len
21. **if** i==j
22. dis\_mat(i,j)=0;
23. **else**
24. dis\_mat(i,j)=sqrt((data1(1,i)-data1(1,j)).^2+(data2(1,i)-data2(1,j)).^2);
25. dis\_mat(j,i)=dis\_mat(i,j);
26. end
27. end
28. end
29. %---------在cluster中标记每个对象eps邻域内的数据对象
30. num=zeros(1,len);
31. cluster=zeros(len,len);
32. **for** i=1:len
33. **for** j=1:len
34. **if** dis\_mat(i,j)<Eps
35. cluster(i,j)=j;
36. end
37. end
38. end
39. %------------------------------%
40. global state;      %状态标记，0：未标记，>1：类别
41. state=zeros(1,len);
42. **for** i=1:len
43. num\_exp=zeros(1,len);
44. **if** state(1,i)==0     %找到未标记的数据对象
45. num=find\_cluster(i,cluster)   %找到点i的直接邻域
46. **if** size(num,2)<MinPts
47. % i=i+1;
48. **else**
49. num\_exp=find\_expand(num,cluster);         %找到点i密度可达的对象
50. **for** j=1:len
51. **if** num\_exp(1,j)==1
52. state(1,j)=count;
53. end
54. end
55. count=count+1;
56. end
57. end
58. end
59. draw(data1,data2,state,count)

**find\_cluster.m (查找簇代码)**

1. function [ sum ] = find\_cluster( p,cluster )
2. %UNTITLED1 Summary of **this** function goes here
3. %  %找到点i的直接邻域
4. %global cluster;
5. global len;
6. sum=zeros(1,len);
7. **for** i=1:len
8. **if** cluster(p,i)~=0 & sum(1,i)==0
9. sum(1,i)=i;
10. end
11. end
12. sum=sum(sum~=0);

**find\_expand.m（扩展簇代码）**

1. function [ label] = find\_expand( num,cluster )
2. %UNTITLED1 Summary of **this** function goes here
3. %  %找到点i密度可达的对象，采用堆栈的思想，弹出最后一个对象，计算eps邻域，再将未标记的对象压入
4. %  直到栈为空为止
5. global cluster;
6. global state;
7. global MinPts;
8. global len;
9. label=zeros(1,len);       %类别标签
10. **while**( ~isempty(num))
11. m=size(num,2);
12. **for** i=1:m        %找到邻域内未被标记的对象
13. **if** state(1,num(1,i))==0
14. label(num(1,i))=1;
15. end
16. end
17. n=num(1,m);
18. sum=find\_cluster(n,cluster);     %获取末位元素的eps邻域对象
19. **if**  size(sum,2)>MinPts
20. num(m)=[];  %弹出栈中末位元素并将它邻域内的数据对象压入栈
21. sum\_t=[];   %sum\_t统计未被分类的末位元素的eps邻域对象
22. **for** i=1:size(sum,2)
23. **if** label(sum(1,i))==0
24. sum\_t=**union**(sum\_t,sum(1,i));
25. end
26. end
27. num=**union**(sum\_t,num);
28. **if**(isempty(num))
29. **break**;
30. end
31. **else**
32. num(m)=[];        %末位元素不是核心点，直接弹出
33. end
34. end

**draw.m（图像实现代码）**

1. %画出个簇
2. %
3. function draw(data1,data2,state,count)
4. type = ['+','d','>','o','p','x','v','<'];
5. **for** i=1:count - 1
6. c = find(state == i);
7. plan1 = zeros(1,size(c,2));
8. plan2 = zeros(1,size(c,2));
9. b = size(c, 2)
10. **for** j= 1:size(c, 2)
11. plan1(:,j) = data1(:,c(:,j));
12. plan2(:,j) = data2(:,c(:,j));
13. end
14. scatter(plan1,plan2,10,type(i));
15. hold on
16. end