上次课回顾

1、聚类的基本定义：cluster

2、如何评判聚类的好坏：距离函数

3、聚类的技术与方法

划分法：k均值、PAM

层次法：凝聚层次聚类、分裂层次聚类

基于密度的方法：Density-based approach

基于模型的方法：Model-based approach

4、聚类的应用

k-means例子

样本数据

序号	属性 1	属性 2
1	1	1
2	2	1
3	1	2
4	2	2
5	4	3
6	5	3
7	4	4
8	5	4

根据所给的数据通过对其实施*k*-means (设*n*=8，*k*=2),，其主要执行执行步骤：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇{1，2}和{3，4，5，6，7，8}。

对于产生的簇分别计算平均值，得到平均值点。

对于{1，2}，平均值点为（1.5，1）（这里的平均值是简单的相加出2）；对于{3，4，5，6，7，8}，平均值点为（3.5，3）。

第二次迭代：通过平均值调整对象的所在的簇，重新聚类，即将所有点按离平均值点（1.5，1）、（3.5，1）最近的原则重新分配。得到两个新的簇：{1，2，3，4}和{5，6，7，8}。重新计算簇平均值点，得到新的平均值点为（1.5，1.5）和（4.5，3.5）。

第三次迭代：将所有点按离平均值点（1.5，1.5）和（4.5，3.5）最近的原则重新分配，调整对象，簇仍然为{1，2，3，4}和{5，6，7，8}，发现没有出现重新分配，而且准则函数收敛，程序结束。

迭代次数	平均值（簇1）	平均值（簇2）	产生的新簇	新平均值（簇1）	新平均值（簇2）
1	（1，1）	（1，2）	{1，2}，{3，4，5，6，7，8}	（1.5，1）	（3.5，3）
2	（1.5，1）	（3.5，3）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）
3	（1.5，1.5）	（4.5，3.5）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）

第五章 聚类分析

1、聚类的基本定义：cluster

2、如何评判聚类的好坏：距离函数

3、聚类的技术与方法

划分法：k均值、PAM

层次法：凝聚层次聚类、分裂层次聚类

基于密度的方法：Density-based approach

基于模型的方法：Model-based approach

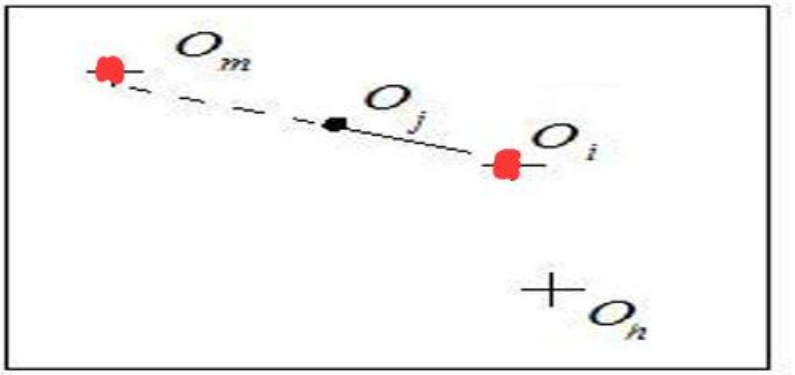
4、聚类的应用

PAM 方法

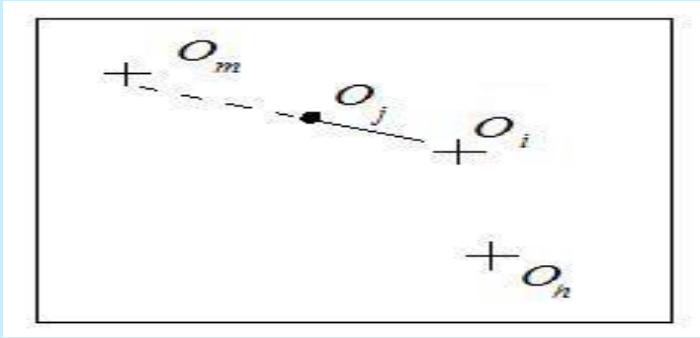
- PAM：Partitioning Around Medoid，围绕中心点的聚类划分，选用簇中位置最中心的对象作为代表对象，试图对n个对象给出k个划分。
  - 代表对象也被称为是中心点，其他对象则被称为非代表对象。
  - 最初随机选择k个对象作为中心点，该算法反复地用非代表对象来代替代表对象，试图找出更好的中心点，以改进聚类的质量。
  - 在每次迭代中，所有可能的对象对被分析，每个对中的一个对象是中心点，而另一个是非代表对象。
  - 对可能的各种组合，估算聚类结果的质量。一个对象Oi被可以产生最大平方-误差值减少的对象代替。在一次迭代中产生的最佳对象集合成为下次迭代的中心点。

PAM 方法

- O_i 和 O_m 是两个原中心点，O_h 是替换 O_i 作为新的中心点



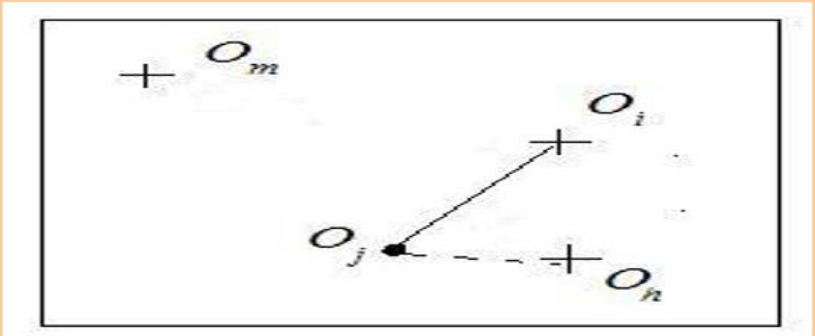
第一种情况



O_j 被重新分配给 O_m,

$C_{jih} = d(j, m) - d(j, i)$

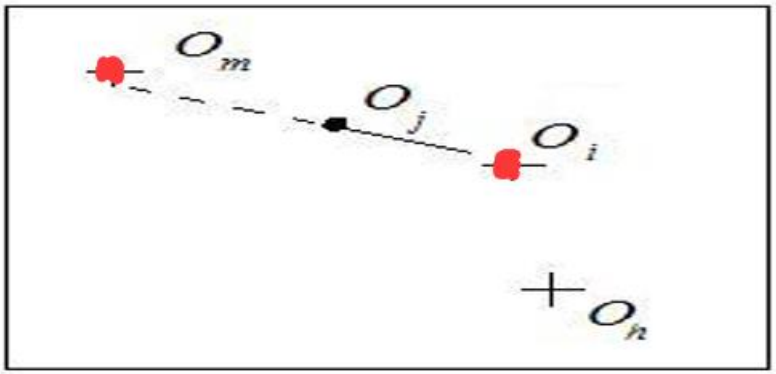
第二种情况



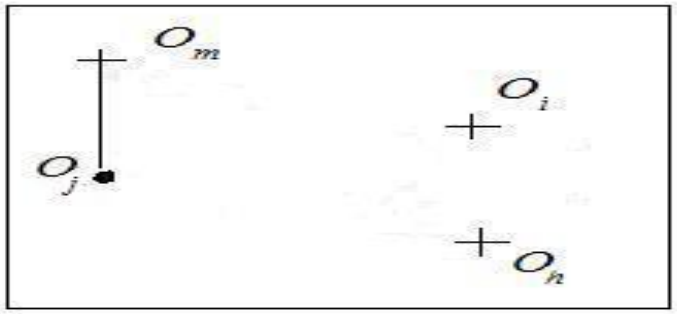
O_j被重新分配给O_h,

$C_{jih} = d(j, h) - d(j, i)$

- O_i 和 O_m 是两个原中心点，O_h 是替换 O_i 作为新的中心点

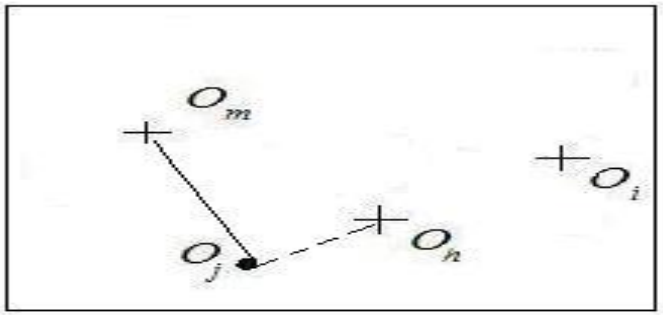


第三种情况



O_j 的隶属不发生变化，
$C_{jih} = 0$

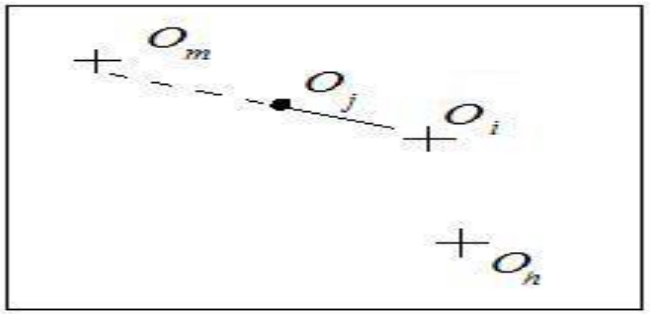
第四种情况



O_i被重新分配给O_h，
$C_{jih} = d(j, h) - d(j, m)$

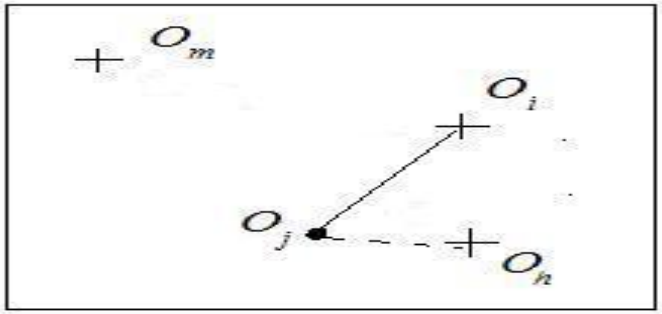
PAM算法代价函数的四种情况

第一种情况



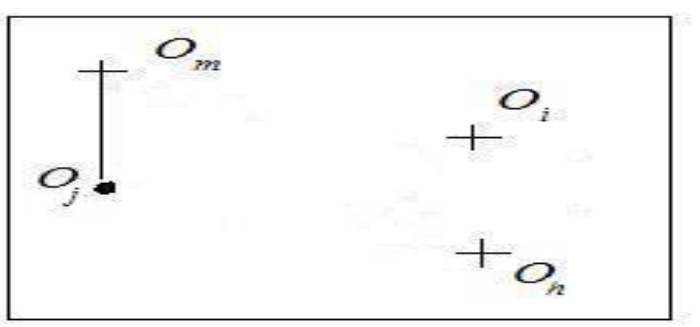
O_j被重新分配给O_m,
$C_{jih} = d(j, m) - d(j, i)$

第二种情况



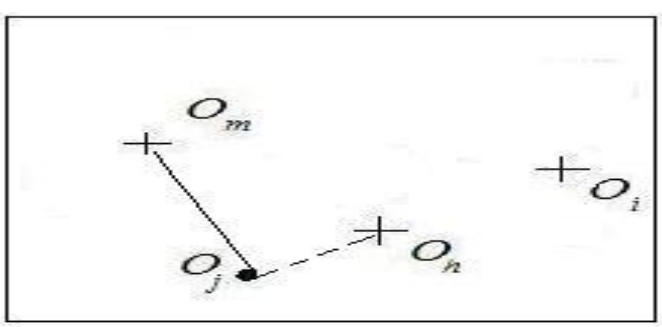
O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, i)$

第三种情况



O_j的隶属不发生变化,
$C_{jih} = 0$

第四种情况



O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, m)$

PAM算法基本思想(续)

- 为了判定一个非代表对象O_h是否是当前一个代表对象O_i的好的替代，对于每一个非中心点对象O_j，下面的四种情况被考虑：
  - **第一种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h所代替作为中心点，且O_j离一个O_m最近，$i \neq m$，那么O_j被重新分配给O_m。
  - **第二种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_j被重新分配给O_h。
  - **第三种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，而O_j依然离O_m最近，那么对象的隶属不发生变化。
  - **第四种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_i被重新分配给O_h。

PAM算法基本思想(续)

- 每当重新分配发生时，平方-误差E所产生的差别对代价函数有影响。因此，如果一个当前的中心点被非中心点所代替，代价函数计算平方-误差值所产生的差别。替换的总代价是所有非中心点对象所产生的代价之和。
  - 总代价定义如下：

$$TC_{ih} = \sum_{j=1}^{n} C_{jih}$$

其中，C_{jih}表示O_j在O_i被O_h代替后产生的代价。下面我们将介绍上面所述的四种情况中代价函数的计算公式，其中所引用的符号有：O_i和O_m是两个原中心点，O_h将替换O_i作为新的中心点。

  - 如果总代价是负的，那么实际的平方-误差将会减小，O_i可以被O_h替代。
  - 如果总代价是正的，则当前的中心点O_i被认为是可接受的，在本次迭代中没有变化。

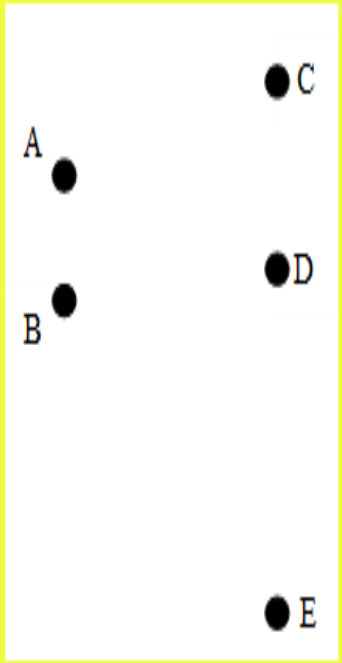
PAM算法

在PAM算法中，可以把过程分为两个步骤：

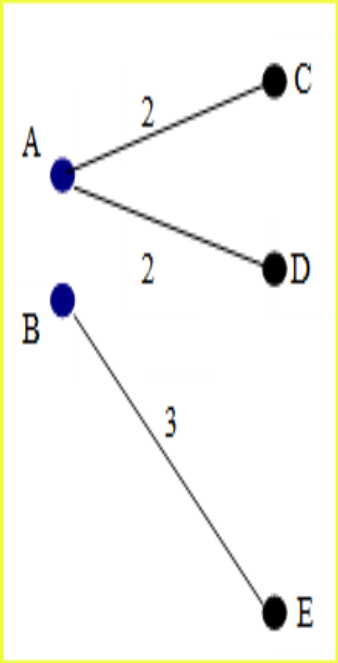
- 建立：随机寻找k个中心点作为初始簇的中心点
- 交换：对于所有可能的对象对进行分析，找到交换后可以使平方-误差减少的对象，代替原中心点

假如空间中的五个点{A、B、C、D、E}如图1所示，各点之间的距离关系如表1所示，根据所给的数据对其运行PAM算法实现划分聚类（设k=2）。样本点间距离如下表所示:

样本点	A	B	C	D	E
A	0	1	2	2	4
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



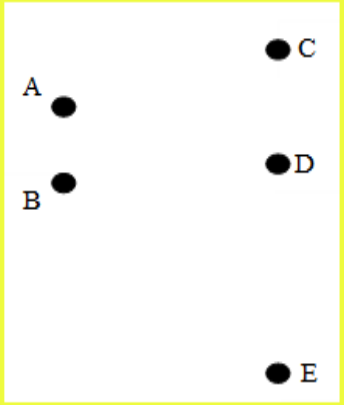
样本点



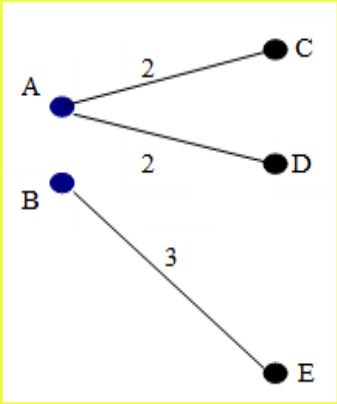
起始中心点为A, B

第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A，B}，则样本被划分为{A、C、D}和{B、E}，如图5-3所示。

样本点	A	B	C	D	E
A	0	1	2	2	4
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



样本点



起始中心点为A,B

第二步　交换阶段：假定中心点A、B分别被非中心点{C、D、E}替换，根据PAM算法需要计算下列代价TC_{AC}、 TC_{AD}、 TC_{AE}、TC_{BC}、TC_{BD}、 TC_{BE}。以TC_{AC}为例说明计算过程：

a）当A被C替换以后，A不再是一个中心点，因为A离B比A离C近，A被分配到B中心点代表的簇，$C_{AAC}=d(A,B)-d(A,A)=1$。

b）B是一个中心点，当A被C替换以后，B不受影响，$C_{BAC}=0$。

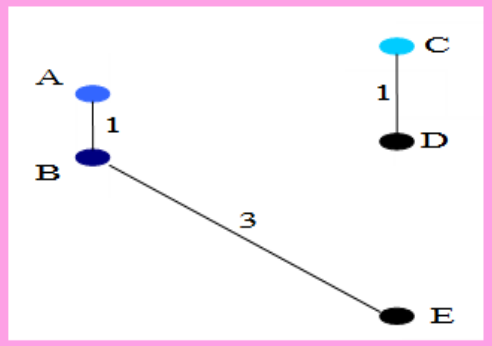
c）C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，符合PAM算法代价函数的第二种情况$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$。

d）D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，根据PAM算法代价函数的第二种情况$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$。

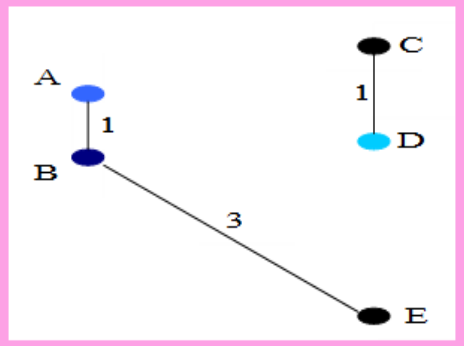
e）E原先属于B中心点所在的簇，当A被C替换以后，离E最近的中心仍然是 B，根据PAM算法代价函数的第三种情况$C_{EAC}=0$。

因此，$TC_{AC}=CA_{AC}+CB_{AC}+CB_{AC}+CD_{AC}+CE_{AC}=1+0-2-1+0=-2$。

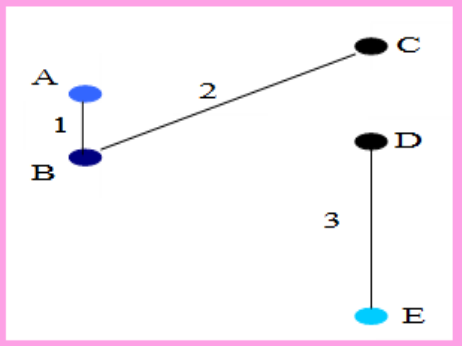
在上述代价计算完毕后，我们要选取一个最小的代价，显然有多种替换可以选择，我们选择第一个最小代价的替换（也就是C替换A），根据图5-4（a）所示，样本点被划分为{ B、A、E}和{C、D}两个簇。图5-4（b）和图5-4（c）分别表示了D替换A，E替换A的情况和相应的代价



(a) C替换A, $TC_{AC} = -2$



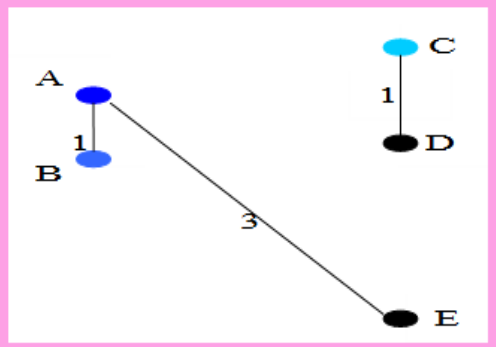
(b) D替换A, $TC_{AD} = -2$



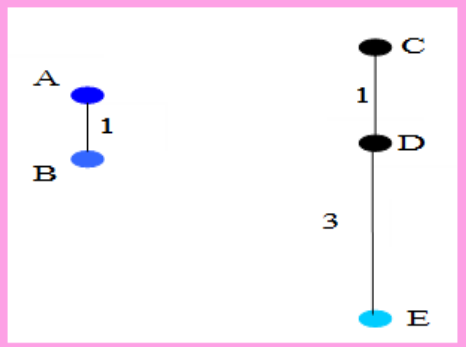
(c) E替换A, $TC_{AE} = -1$

图5-4 替换中心点A

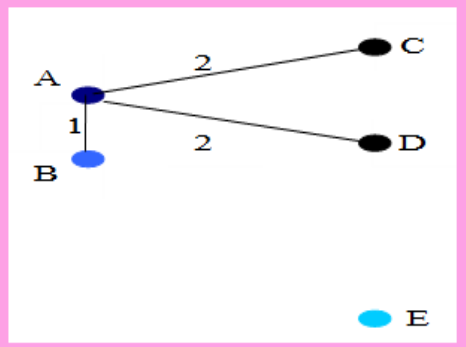
图5-5（a）、（b）、（c）分别表示了用C、D、E替换B的情况和相应的代价。



(a) C替换B, $TC_{BC} = -2$



(b) D替换B, $TC_{BD} = -2$



(c) E替换B, $TC_{BE} = -2$

图5-5 替换中心点B

通过上述计算，已经完成了PAM算法的第一次迭代。在下一迭代中，将用其他的非中心点{A、D、E}替换中心点{B、C}，找出具有最小代价的替换。一直重复上述过程，直到代价不再减小为止。

PAM算法基本思想(续)

算法5-2 PAM（围绕中心点的划分）

输入：簇的数目k和包含n个对象的数据库。

输出：k个簇，使得所有对象与其最近中心点的相异度总和最小。

（1） 任意选择k个对象作为初始的簇中心点；

（2） REPEAT

（3） 指派每个剩余的对象给离它最近的中心点所代表的簇；

（4） REPEAT

（5） 选择一个未被选择的中心点O_i；

（6） REPEAT

（7） 选择一个未被选择过的非中心点对象O_h；

（8） 计算用O_h代替O_i的总代价并记录在S中；

（9） UNTIL 所有的非中心点都被选择过；

（10） UNTIL 所有的中心点都被选择过；

（11） IF 在S中的所有非中心点代替所有中心点后的计算出的总代价有小于0的存在 THEN 找出S中的用非中心点替代中心点后代价最小的一个，并用该非中心点替代对应的中心点，形成一个新的k个中心点的集合；

（12）UNTIL 没有再发生簇的重新分配，即所有的S都大于0.

PAM算法性能分析

- (1) 消除k-平均算法对于孤立点的敏感性。
- (2) K-中心点方法比k-平均算法的代价要高
- (3) 必须指定k
- (4) PAM对小的数据集非常有效，对大数据集效率不高。特别是n和k都很大的时候。

第五章 聚类分析

1、聚类的基本定义：cluster

2、如何评判聚类的好坏：距离函数

3、聚类的技术与方法

划分法：k均值、PAM、

层次法：凝聚层次聚类、分裂层次聚类

基于密度的方法：Density-based approach

基于模型的方法：Model-based approach

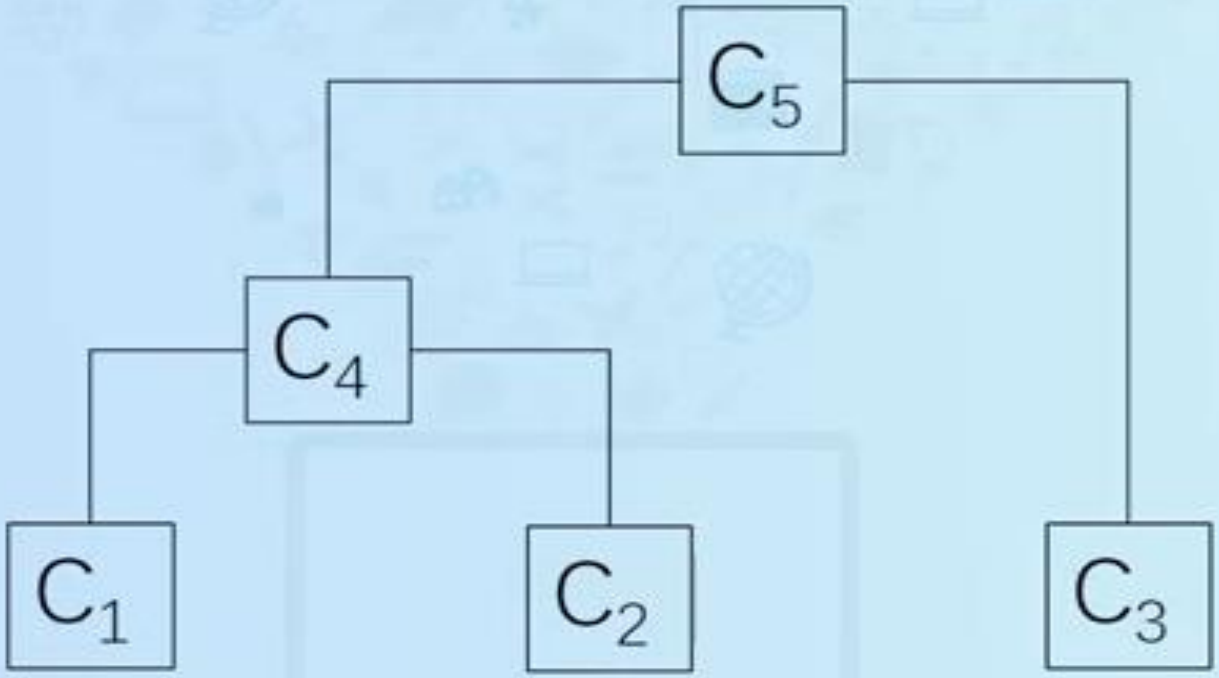
4、聚类的应用

第五章 聚类方法

内容提要

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 其它聚类方法

层次聚类方法概述



层次型聚类示意

层次聚类法分为:凝聚层次聚类、分裂层次聚类

层次聚类方法概述

- **层次聚类方法对给定的数据集进行层次的分解，直到某种条件满足为止。具体又可分为：**
  - 凝聚的层次聚类：一种自底向上的策略，首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到某个终结条件被满足，如AGNES算法。
  - 分裂的层次聚类：采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到达到了某个终结条件，如DIANA算法。

AGNES算法

- AGNES (AGglomerative NESting)：自底向上凝聚算法，先将每个对象作为一个簇，然后这些簇根据某些准则被一步步地合并。两个簇间的相似度由这两个不同簇中距离最近的数据点对的相似度来确定。聚类的合并过程反复进行直到所有的对象最终满足

算法5-3 AGNES（自底向上凝聚算法）

输入：包含n个对象的数据库，终止条件簇的数目k。

输出：k个簇，达到终止条件规定簇数目。

(1) 将每个对象当成一个初始簇；

(2) REPEAT

(3) 根据两个簇中最近的数据点找到最近的两个簇；

(4) 合并两个簇，生成新的簇的集合；

(5) UNTIL 达到定义的簇的数目；

AGNES算法

序号	属性 1	属性 2
①	1	1
②	1	2
③	2	1
④	2	2
⑤	3	4
⑥	3	5
⑦	4	4
⑧	4	5

AGNES算法例子

序号	属性 1	属性 2
①	1	1
②	1	2
③	2	1
④	2	2
⑤	3	4
⑥	3	5
⑦	4	4
⑧	4	5

第1步：根据初始簇计算每个簇之间的距离，随机找出距离最小的两个簇，进行合并，最小距离为1，合并后1，2点合并为一个簇。

第2步：，对上一次合并后的簇计算簇间距离，找出距离最近的两个簇进行合并，合并后3，4点成为一簇。

第3步：重复第2步的工作，5，6点成为一簇。

第4步：重复第2步的工作，7，8点成为一簇。

第5步：合并{1，2}，{3，4}成为一个包含四个点的簇。

第6步：合并{5，6}，{7，8}，由于合并后的簇的数目已经达到了用户输入的终止条件程序结束。

步骤	最近的簇距离	最近的两个簇	合并后的新簇
1	1	{1}，{2}	{1，2}，{3}，{4}，{5}，{6}，{7}，{8}
2	1	{3}，{4}	{1，2}，{3，4}，{5}，{6}，{7}，{8}
3	1	{5}，{6}	{1，2}，{3，4}，{5，6}，{7}，{8}
4	1	{7}，{8}	{1，2}，{3，4}，{5，6}，{7，8}
5	1	{1，2}，{3，4}	{1，2，3，4}，{5，6}，{7，8}
6	1	{5，6}，{7，8}	{1，2，3，4}，{5，6，7，8}结束

AGNES性能分析

- AGNES算法比较简单，但经常会遇到合并点选择的困难。假如一旦一组对象被合并，下一步的处理将在新生成的簇上进行。已做处理不能撤消，聚类之间也不能交换对象。如果在某一步没有很好的选择合并的决定，可能会导致低质量的聚类结果。

- 这种聚类方法不具有很好的可伸缩性，因为合并的决定需要检查和估算大量的对象或簇。

- 假定在开始的时候有n个簇，在结束的时候有1个簇，因此在主循环中有n次迭代，在第i次迭代中，我们必须在$n-i+1$个簇中找到最靠近的两个聚类。另外算法必须计算所有对象两两之间的距离，因此这个算法的复杂度为$O(n^2)$，该算法对于n很大的情况是不适用的。

DIANA算法

- DIANA (Divisive ANAlysis)算法是典型的分裂聚类方法。
- 用户能定义希望得到的簇数目作为一个结束条件。同时，它使用下面两种测度方法：
  - 簇的直径：在一个簇中的任意两个数据点的距离中的最大值。
  - 平均相异度（平均距离）：

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} |x - y|$$

算法5-4 DIANA（自顶向下分裂算法）

输入：包含n个对象的数据库，终止条件簇的数目k。

输出：k个簇，达到终止条件规定簇数目。

（1）将所有对象整个当成一个初始簇；

（2） FOR （i=1; i≠k; i++) DO BEGIN

（3） 在所有簇中挑出具有最大直径的簇C；

（4） 找出C中与其它点平均相异度最大的一个点p并把p放入splinter group，剩余的放在old party中；

（5）. REPEAT

（6） 在old party里找出到最近的splinter group中的点的距离不大于到old party中最近点的距离的点，并将该点加入splinter group。

（7） UNTIL 没有新的old party的点被分配给splinter group；

（8） splinter group和old party为被选中的簇分裂成的两个簇，与其它簇一起组成新的簇集合。

（9） END.

DIANA算法

序号	属性 1	属性 2
①	1	1
②	1	2
③	2	1
④	2	2
⑤	3	4
⑥	3	5
⑦	4	4
⑧	4	5

序号	属性 1	属性 2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

第1步，找到具有最大直径的簇，对簇中的每个点计算平均相异度（如欧式距离）

1的平均距离：（1+1+1.414+3.6+4.24+4.47+5）/7=2.96

2的平均距离为2.526;

3的平均距离为2.68;

4的平均距离为2.18;

.....

挑出平均相异度最大的点1放到splinter group中，剩余点在old party中。

第2步，在old party里找出到最近的splinter group中的点的距离不大于到old party中最近的点的距离的点，将该点放入splinter group中，该点是2。

第3步，重复第2步的工作，splinter group中放入点3。

第4步，重复第2步的工作，splinter group中放入点4。

第5步，没有在old party中的点放入了splinter group中且达到终止条件（k-2），程序终止。如果没有到终止条件，因该从分裂好的簇中选一个直径最大的簇继续分裂。

步骤	具有最大直径的簇	splinter group	Old party
1	{1，2，3，4，5，6，7，8}	{1}	{2，3，4，5，6，7，8}
2	{1，2，3，4，5，6，7，8}	{1，2}	{3，4，5，6，7，8}
3	{1，2，3，4，5，6，7，8}	{1，2，3}	{4，5，6，7，8}
4	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8}
5	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8} 终止

第五章 聚类方法

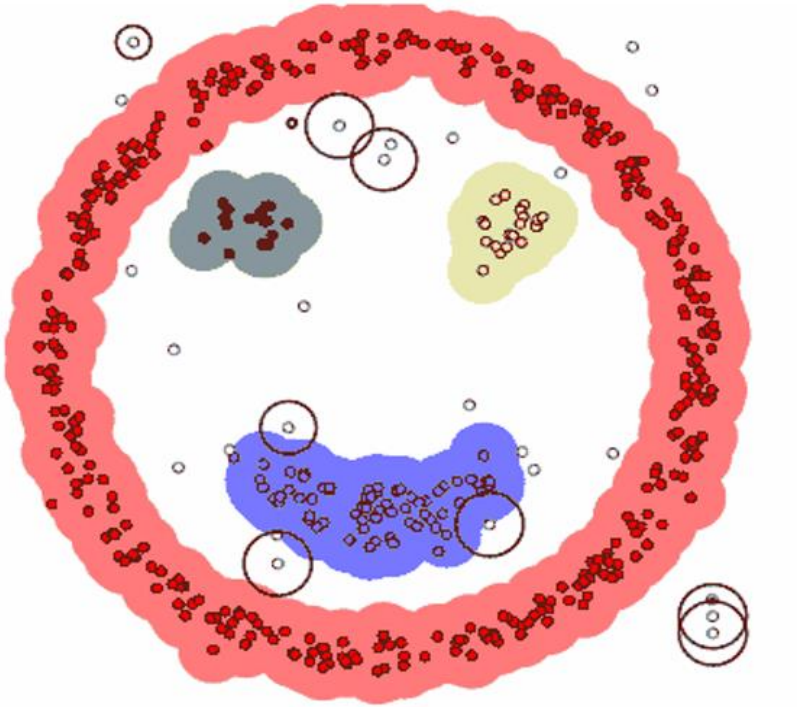
内容提要

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 其它聚类方法

第5.4章 密度聚类方法

- 密度聚类方法的指导思想是，只要一个区域中，点的密度大于某个阈值，就把它加到与之相连的簇中去。

区域：以q为中心点 ε 为半径，



阈值：包含多于MinPts个对象

第5.4章 密度聚类方法

- 1. 密度聚类方法的概念
- 2. 密度聚类典型算法
- 3. 密度聚类算法描述
- 4. 密度聚类算法的性能分析
- 5. 密度聚类在数据挖掘中的应用

第5.4章 密度聚类方法

- 密度聚类的典型算法：

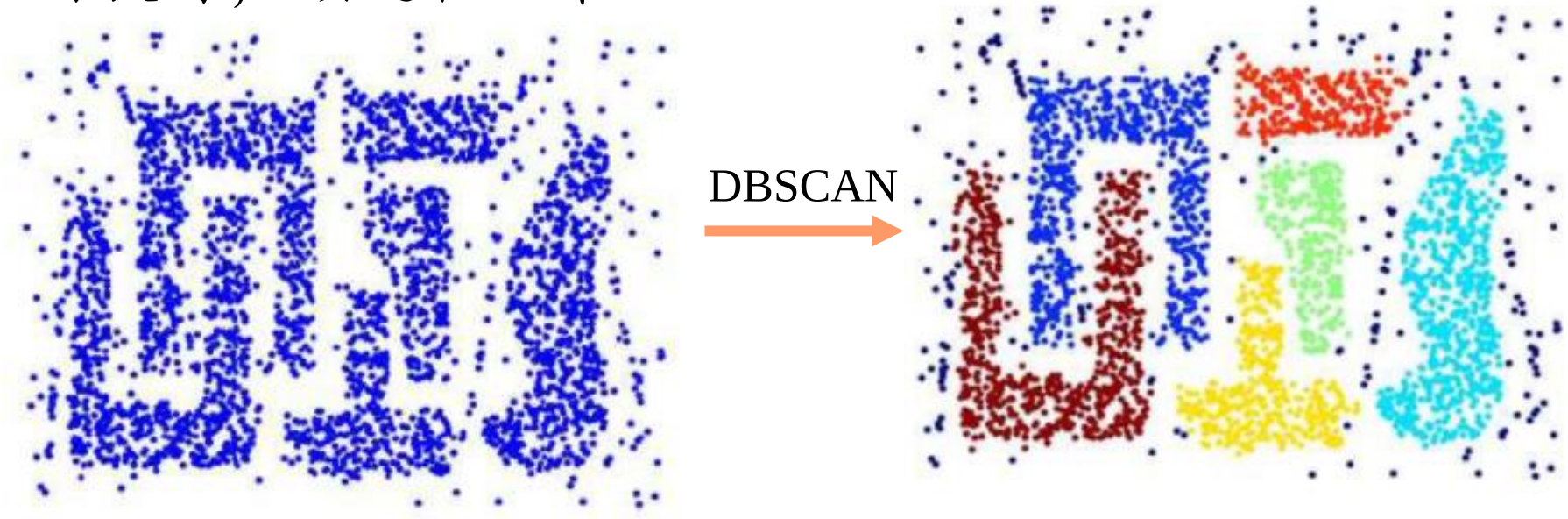
(1) DBSCAN：Density-Based Spatial Clustering of Applications with Noise，噪声环境下的密度聚类算法

(2) OPTICS：Ordering Points To Identify the Clustering Structure，基于不同密度的聚类算法

(3) EDNCLUE：Density Clustering，基于一组密度分布函数的聚类算法

第5.4章 密度聚类方法

- DBSCAN算法：如果一个点q的区域内包含多于MinPts个对象，则创建一个q作为核心对象的簇。然后，反复地寻找从这些核心对象直接密度可达的对象，把一些密度可达簇进行合并。当没有新的点可以被添加到任何簇时，该过程结束。

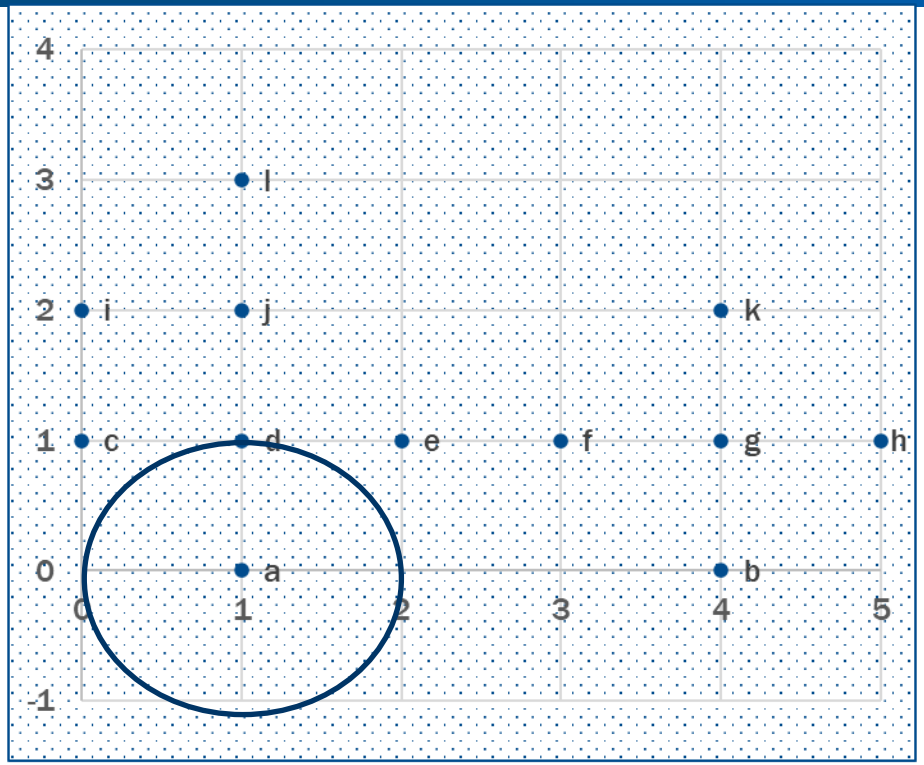


原始数据集　　DBSCAN聚类结果

第5.4章 密度聚类方法

样本	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*
属性x	1	4	0	1	2	3	4	5	0	1	4	1
属性y	0	0	1	1	1	1	1	1	2	2	2	3

2、密度聚类方法



n=12，

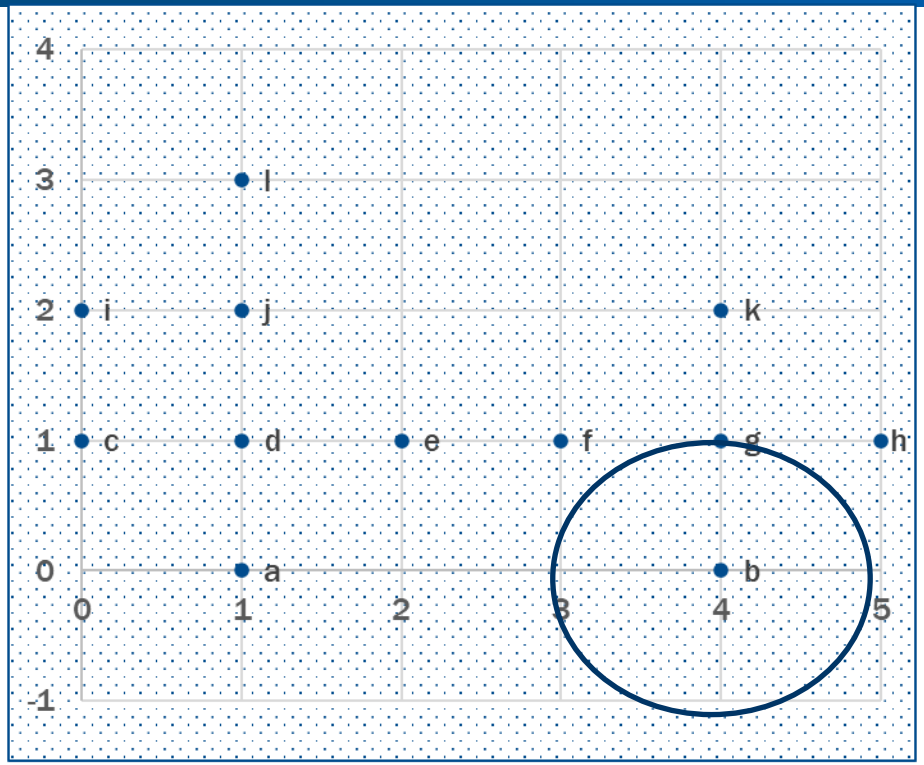
ε=1，

MinPts=4。

核心对象？

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
①	a	2	无

2、密度聚类方法



n=12，

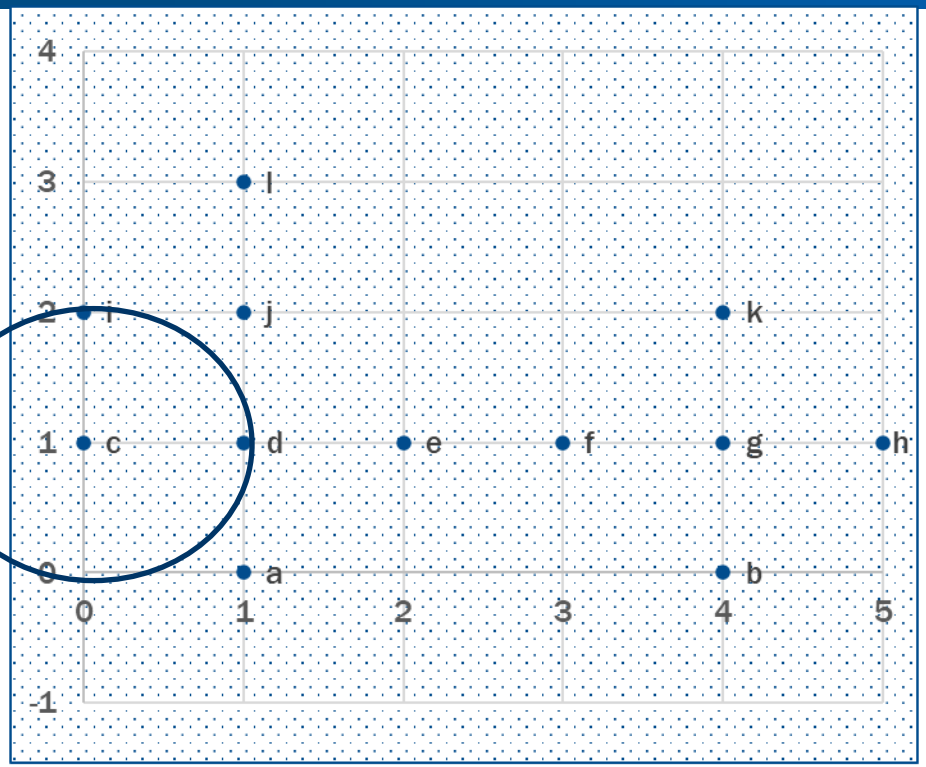
ε=1，

MinPts=4。

核心对象？

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
②	b	2	无

2、密度聚类方法

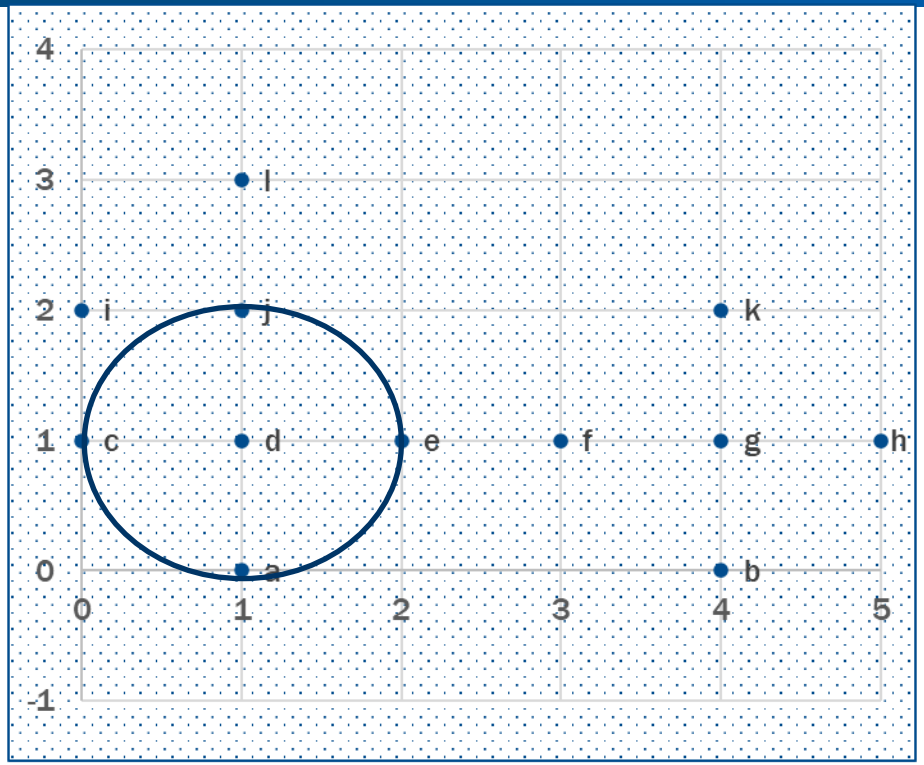


n=12，ε=1，MinPts=4。

核心对象？

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
③	c	3	无

2、密度聚类方法



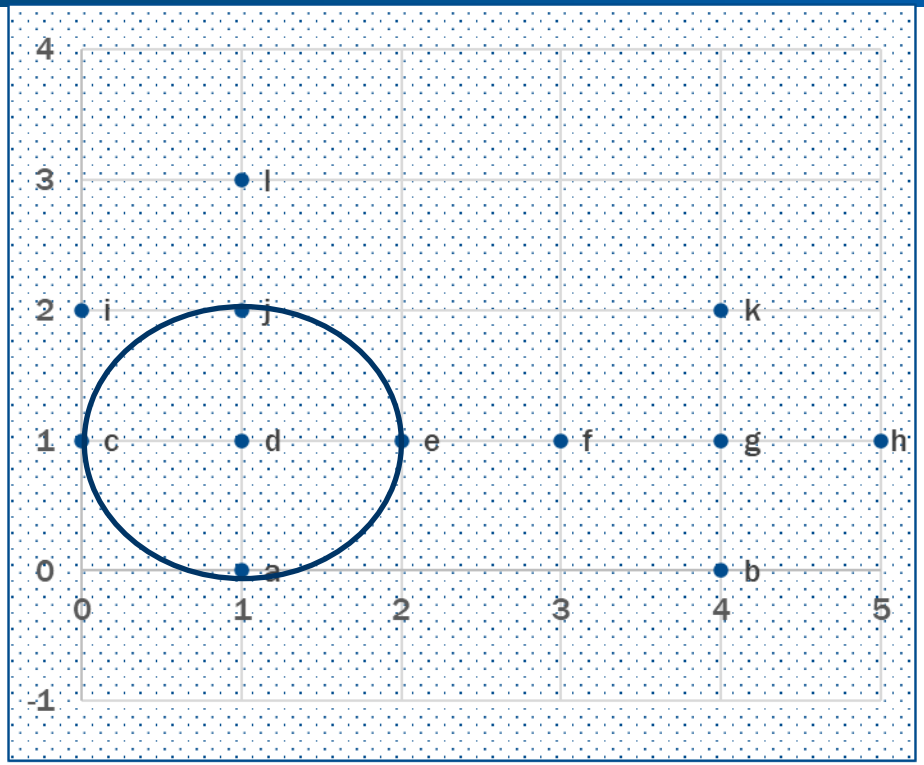
n=12，ε=1，
MinPts=4。

核心对象？

这簇还有其他的点吗？

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
④	d	5	簇C1：{ d }

2、密度聚类方法



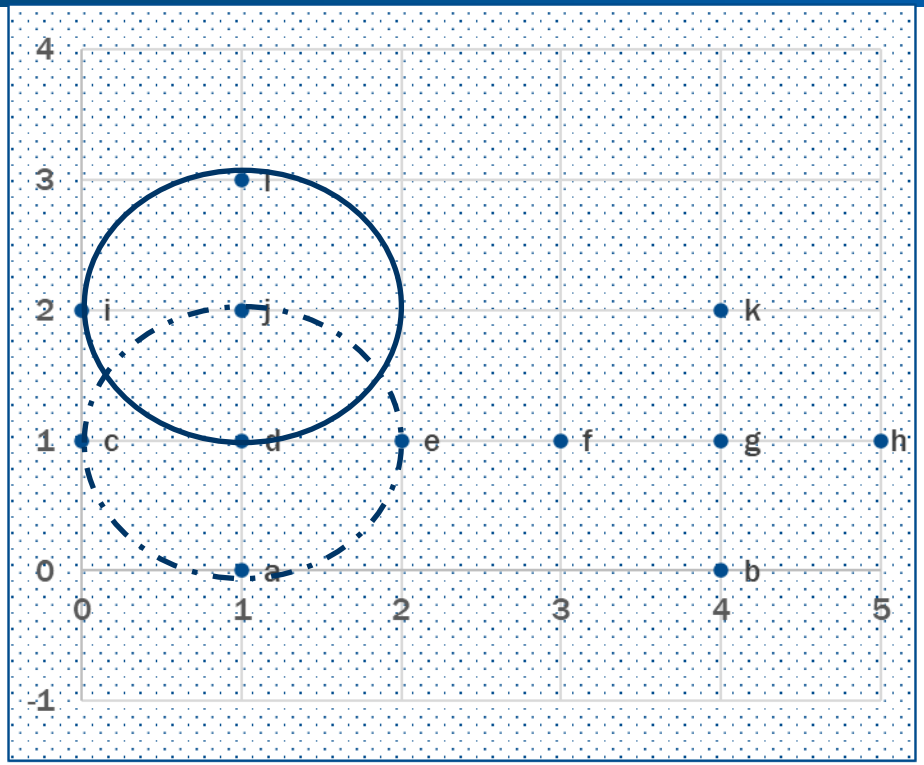
n=12，ε=1，MinPts=4。

直接密度可达的点

这次迭代结束了吗？

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
④	*d*	5	簇C1：{*a*，*c*，*d*，*e*，*j*}

2、密度聚类方法

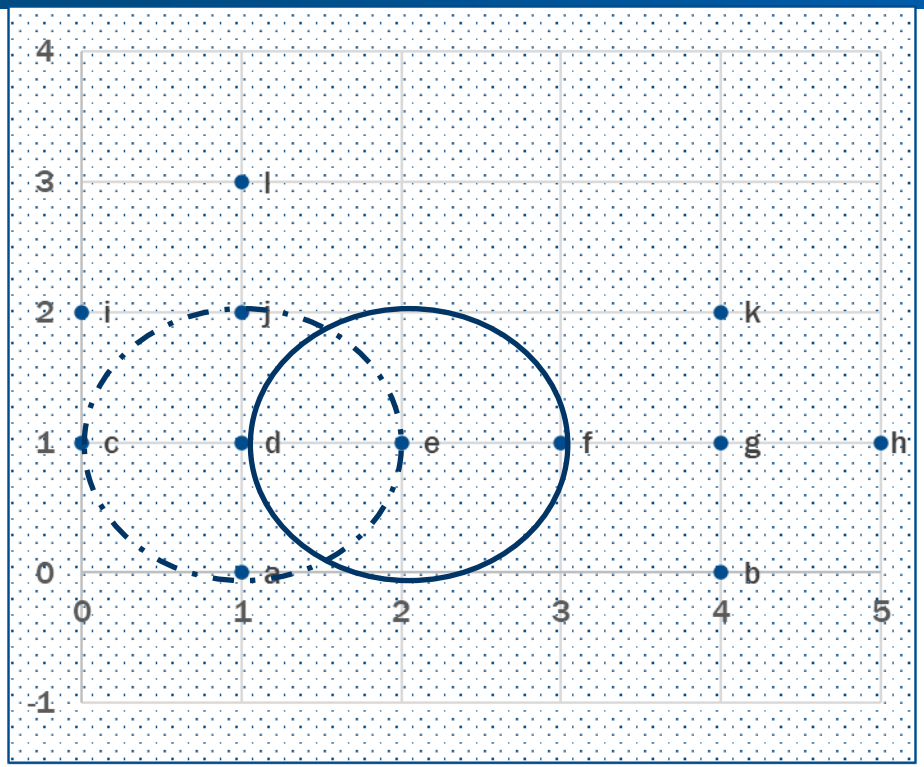


n=12，ε=1，MinPts=4。

密度可达

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
④	*d*	5	簇C1：{*a*，*c*，*d*，*e*，***i***，*j*，***l***}

2、密度聚类方法

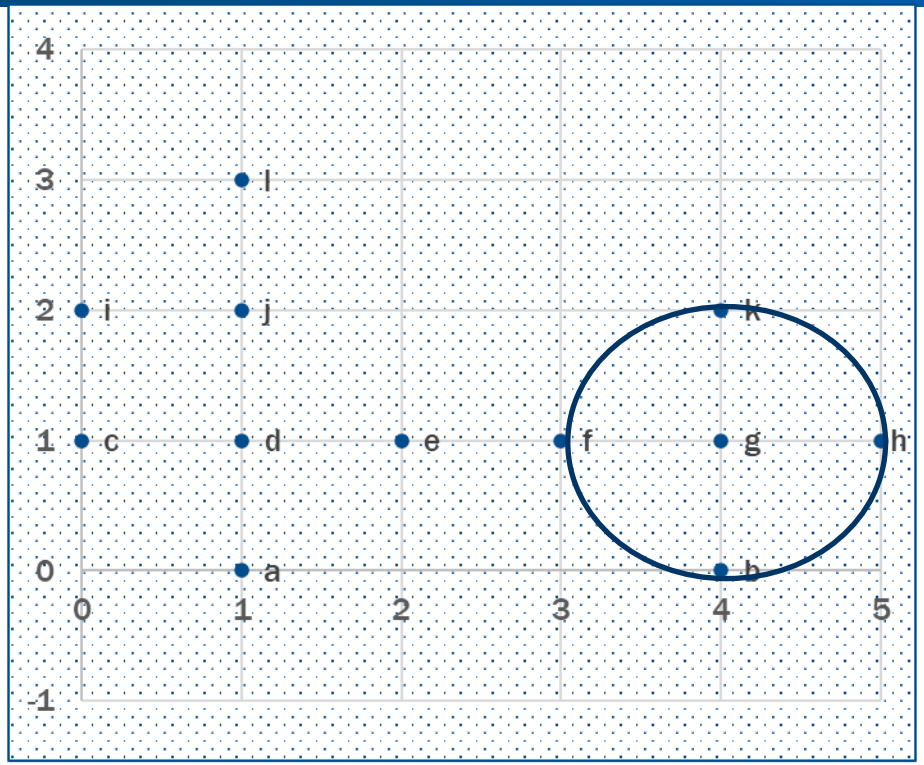


n=12，ε=1，MinPts=4。

密度可达

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
④	*d*	5	簇C1：{*a*，*c*，*d*，*e*，*i*，*j*，*l*}

2、密度聚类方法



n=12，ε=1，MinPts=4。

1、核心对象

2、直接密度可达

3、密度可达

4、密度相连

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
⑦	g	5	簇C2：{ b，f，g，h，k }

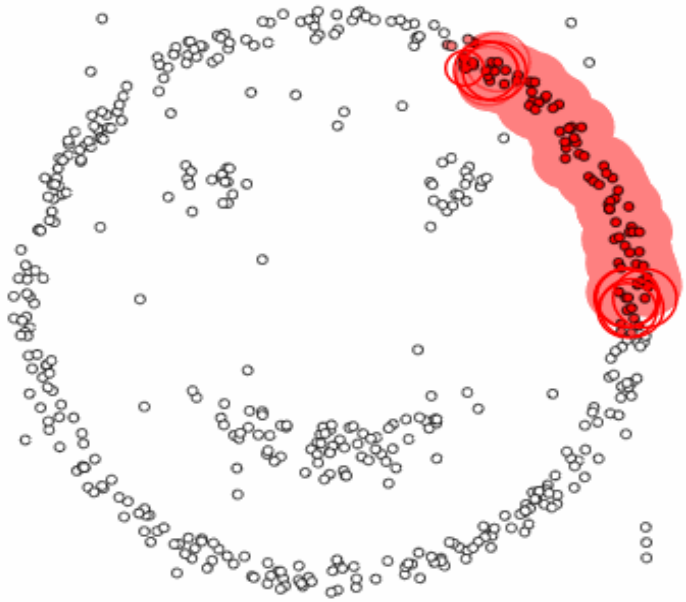
2、密度聚类方法

n=12， ε =1，MinPts=4。

迭代	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
⑦	*g*	5	簇C2：{ *b*，*f*，*g*，*h*，*k* }
⑧	*h*	2	已在一个簇C2中
⑨	*i*	3	已在一个簇C1中
⑩	*j*	4	已在一个簇C1中
⑪	*k*	2	已在一个簇C2中
⑫	*l*	2	已在一个簇C1中

第5.4章 密度聚类方法

- 1. 密度聚类方法的概念
  - （1）核心对象
  - （2）直接密度可达
  - （3）密度可达
  - （4）密度相连



第5.4章 密度聚类方法

- 定义 5-3 对象的ε-临域：给定对象在半径ε内的区域。

  定义 5-4 核心对象：如果一个对象的ε-临域至少包含最小数目MinPts个对象，则称该对象为核心对象。

  定义 5-5 直接密度可达：给定一个对象集合D，如果p是在q的ε-邻域内，而q是一个核心对象，我们说对象p从对象q出发是直接密度可达的。

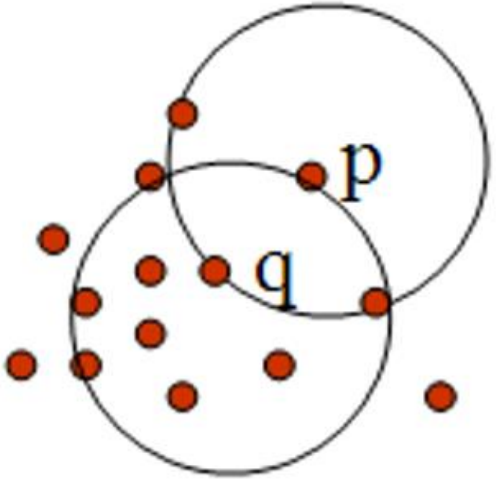


图5-6直接密度可达

第5.4章 密度聚类方法

定义 5-6 密度可达：如果存在一个对象链p_1，p_2,…，p_n，$p_1=q$，$p_n=p$，对$p_i \in D$，（$1<=i<=n$），$pi+1$是从p_i关于ε和MitPts直接密度可达的，则对象p是从对象q关于ε和MinPts间接密度可达。

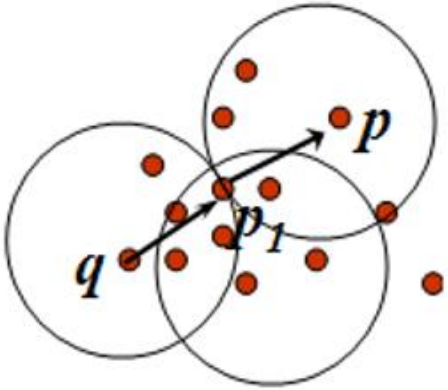


图5-7 密度可达

定义 5-7 密度相连：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和MinPts间接密度可达，那么对象p和q是关于ε和MinPts密度相连。

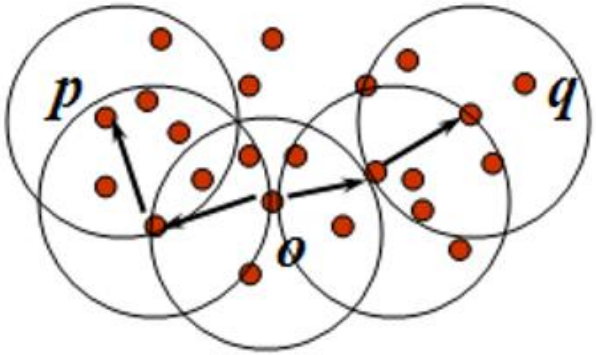


图5-8密度相连

第5.4章 密度聚类方法

定义 5-8 噪声：一个基于密度的簇是基于密度可达性的最大的密度相连对象的集合。不包含在任何簇中的对象被认为是“噪声”。

补充 边界点：落在某个核心点的邻域内，是一个稠密区域边缘上的点。

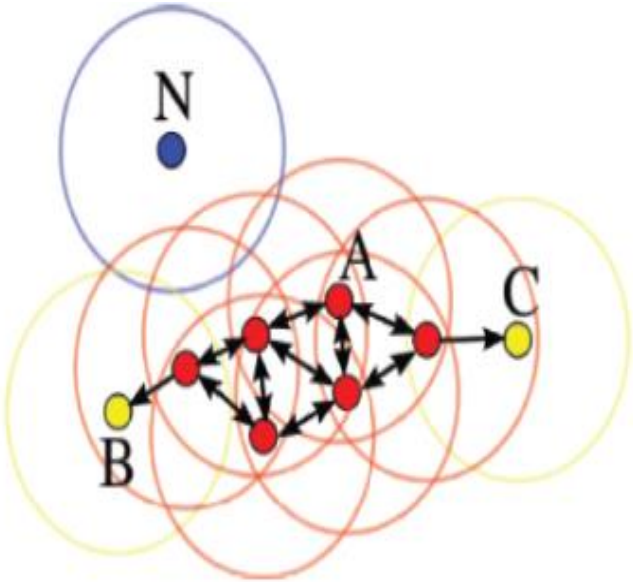


图5-9 噪声与边界点

第5.4章 密度聚类方法

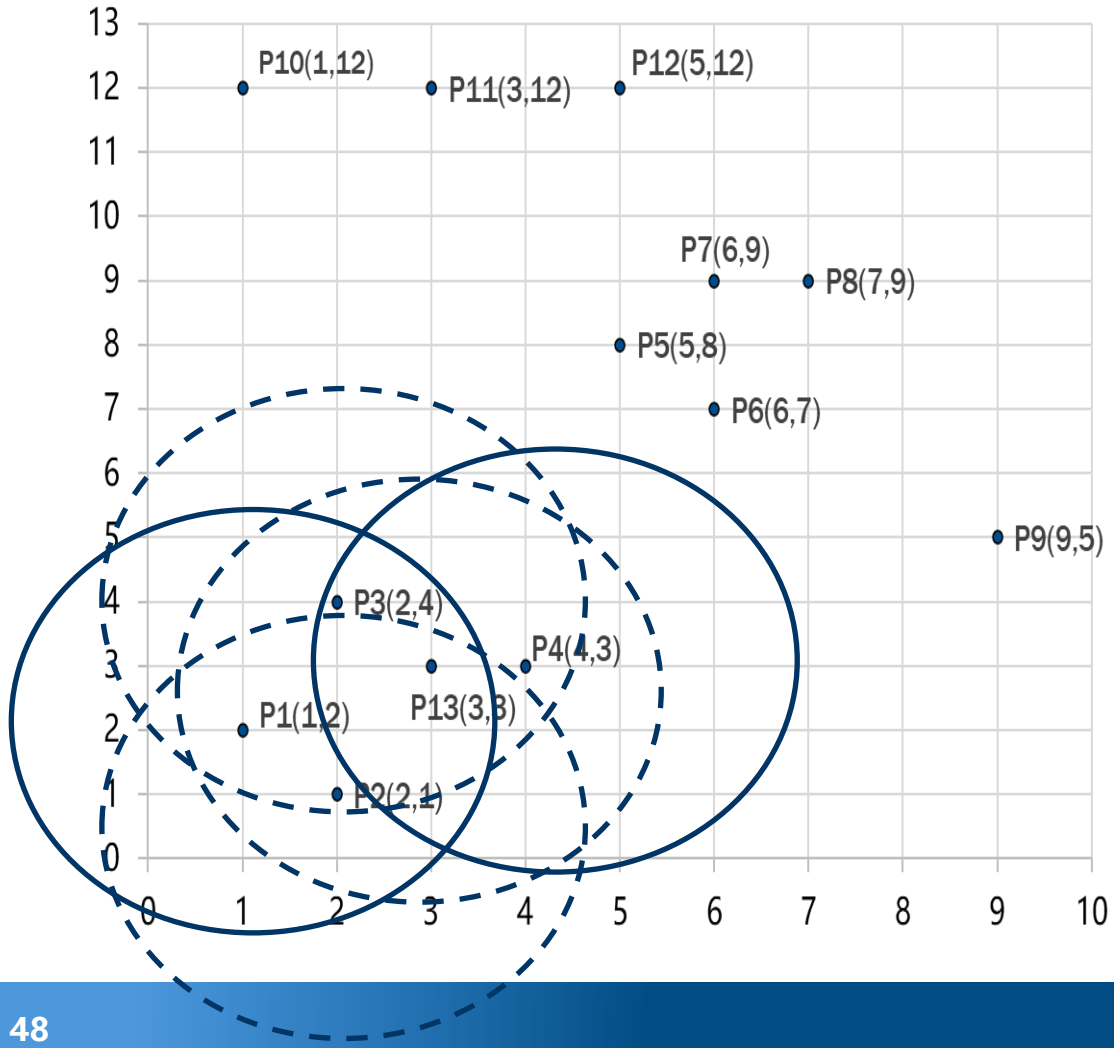
- DBSCAN聚类

举例：这有如下二维数据集，共13个样本点，取ε=3，minpts=3，请使用DBSCAN算法对其聚类（使用曼哈顿距离）。

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13
1	2	2	4	5	6	6	7	9	1	3	5	3
2	1	4	3	8	7	9	9	5	12	12	12	3

第5.4章 密度聚类方法

- DBSCAN算法三个关键的参数： n=12，ε=3，minpts=3

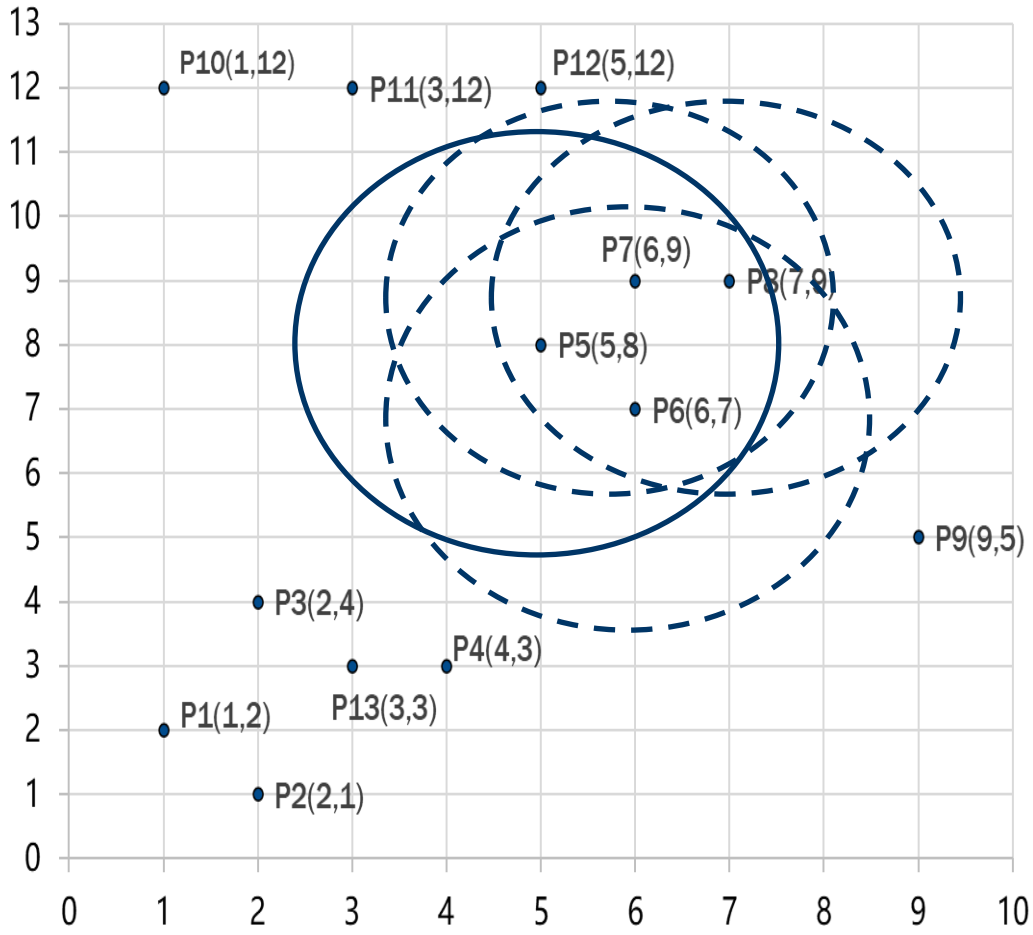


顺序扫描样本点

P1	1	2
P2	2	1
P3	2	4
P4	4	3
P5	5	8
P6	6	7
P7	6	9
P8	7	9
P9	9	5
P10	1	12
P11	3	12
P12	5	12
P13	3	3

第5.4章 密度聚类方法

- DBSCAN算法三个关键的参数： n=12，ε=3，minpts=3

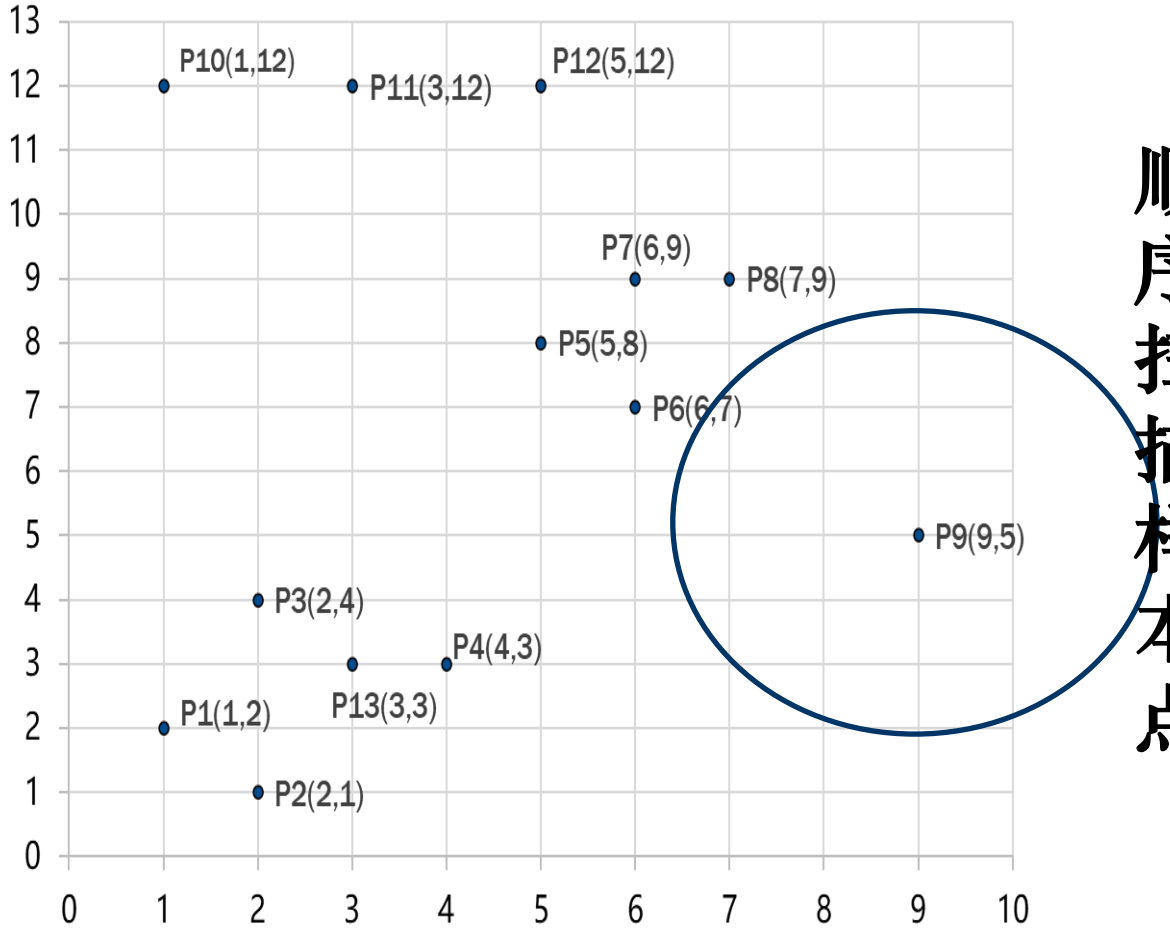


顺序扫描样本点

P1	1	2
P2	2	1
P3	2	4
P4	4	3
P5	**5**	**8**
P6	**6**	**7**
P7	**6**	**9**
P8	**7**	**9**
P9	**9**	**5**
P10	**1**	**12**
P11	**3**	**12**
P12	**5**	**12**
P13	3	3

第5.4章 密度聚类方法

- DBSCAN算法三个关键的参数： n=12，ε=3，minpts=3

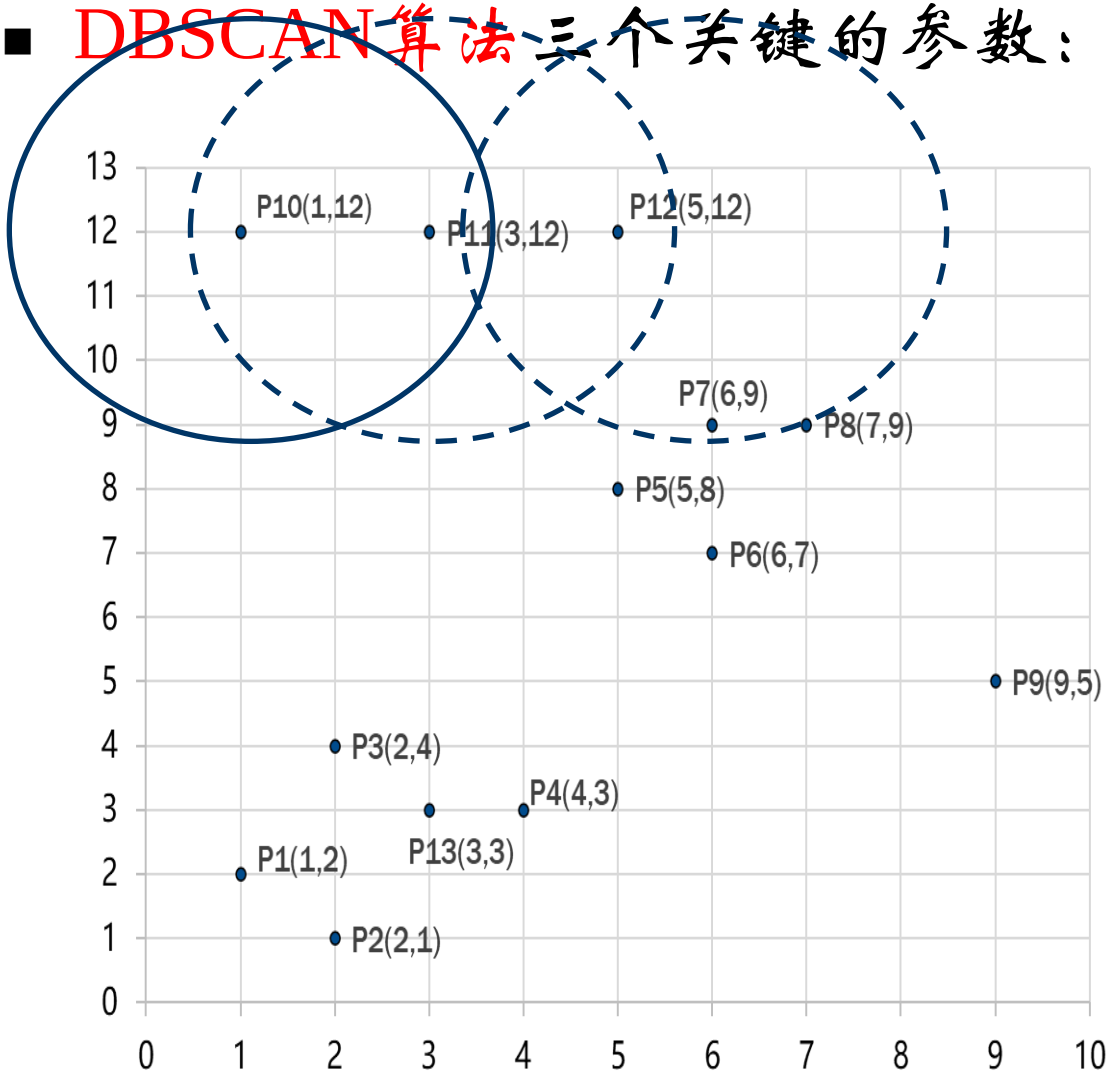


顺序扫描样本点

P1	1	2
P2	2	1
P3	2	4
P4	4	3
P5	5	8
P6	6	7
P7	6	9
P8	7	9
P9	**9**	**5**
P10	**1**	**12**
P11	**3**	**12**
P12	**5**	**12**
P13	3	3

第5.4章 密度聚类方法

- DBSCAN算法三个关键的参数： n=12，ε=3，minpts=3

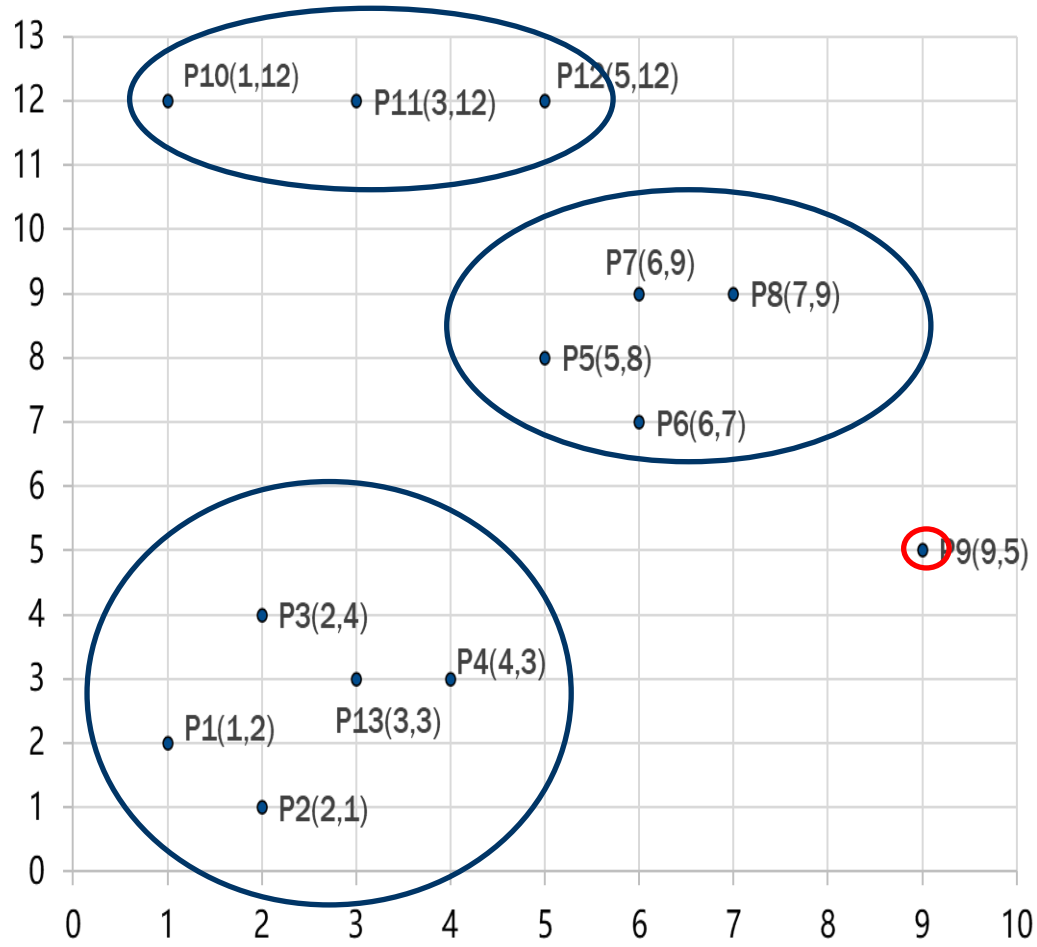


顺序扫描样本点

P1	1	2
P2	2	1
P3	2	4
P4	4	3
P5	5	8
P6	6	7
P7	6	9
P8	7	9
P9	9	5
P10	**1**	**12**
P11	**3**	**12**
P12	**5**	**12**
P13	3	3

第5.4章 密度聚类方法

- DBSCAN算法三个关键的参数： n=12，ε=3，minpts=3



顺序扫描样本点

P1	1	2
P2	2	1
P3	2	4
P4	4	3
P5	5	8
P6	6	7
P7	6	9
P8	7	9
P9	9	5
P10	1	12
P11	3	12
P12	5	12
P13	3	3

第5.4章 密度聚类方法

- 1. 密度聚类方法的概念
- 2. 密度聚类典型算法
- 3. 密度聚类算法描述
- 4. 密度聚类算法的性能分析
- 5. 密度聚类在数据挖掘中的应用

DBSCAN 算法描述：

输入：包含n个对象的数据库，半径 ε，最少数目MinPts。

输出：所有生成的簇，达到密度要求。

（1）DBSCAN通过检查数据集中每点的Eps邻域来搜索簇，如果点p的Eps邻域包含的点多于MinPts个，则创建一个以p为核心对象的簇；

（2）然后，DBSCAN迭代地聚集从这些核心对象直接密度可达的对象，这个过程可能涉及一些密度可达簇的合并；

（3）当没有新的点添加到任何簇时，该过程结束。

第5.4章 密度聚类方法

(1) 首先将数据集D中的所有对象标记为未处理状态
(2) for（数据集D中每个对象p） do
(3)　if （p已经归入某个簇或标记为噪声） then
(4)　　continue;
(5)　else
(6)　　检查对象p的Eps邻域 NEps(p)；
(7)　　if (NEps(p)包含的对象数小于MinPts) then
(8)　　　标记对象p为边界点或噪声点;
(9)　　else
(10)　　　标记对象p为核心点，并建立新簇C, 并将p邻域内所有点加入C
(11)　　　for (NEps(p)中所有尚未被处理的对象q) do
(12)　　　　检查其Eps邻域NEps(q)，若NEps(q)包含至少MinPts个对象，则将NEps(q)中未归入任何一个簇的对象加入C;
(13)　　　end for(14)　　end if(15)　end if(16) end for

第5.4章 密度聚类方法

DBSCAN 算法在educoder平台中运行：

第4关：DBScan算法组合实现

任务要求 参考答案 评论

- 任务描述
- 相关知识
  - DB-Scan算法原理
  - DB-Scan算法流程
  - DB-Scan算法优缺点
- 编程要求
- 测试说明

任务描述

本关任务：综合前面两个关卡的算法，实现 DB-Scan 聚类功能。

相关知识

为了完成本关任务，你需要掌握：

1. DB-Scan 算法原理；
2. DB-Scan 算法的整体流程；
3. DB-Scan 算法优点。

DB-Scan算法原理

基于密度的聚类算法通过寻找被低密度区域分离的高密度区域，并将高密度区域作

```
def LoadDataSet(fileName, splitChar='\t'):
    """
    输入：文件名
    输出：数据集
    描述：从文件读入数据集
    """
    #    请在此添加实现代码      #
    #********** Begin *********#
    #从txt文件中读取数据
    #********** End **********#

def DBScan(data, eps, minPts):
    """
    输入：数据集，半径大小，最小点个数
```

测试集1 消耗内存23.61MB 代码执行时长：2.05秒

预期输出

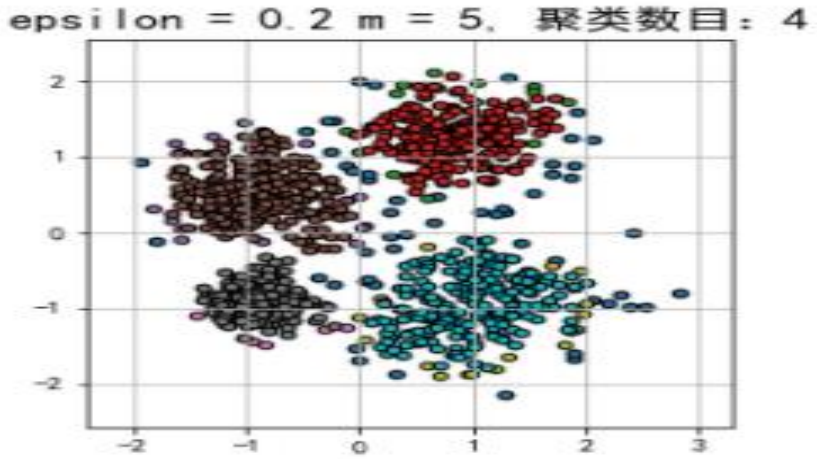
```
cluster Numbers = 7
[3, 167, 34, 273, 103, 129, 45, 34]
```

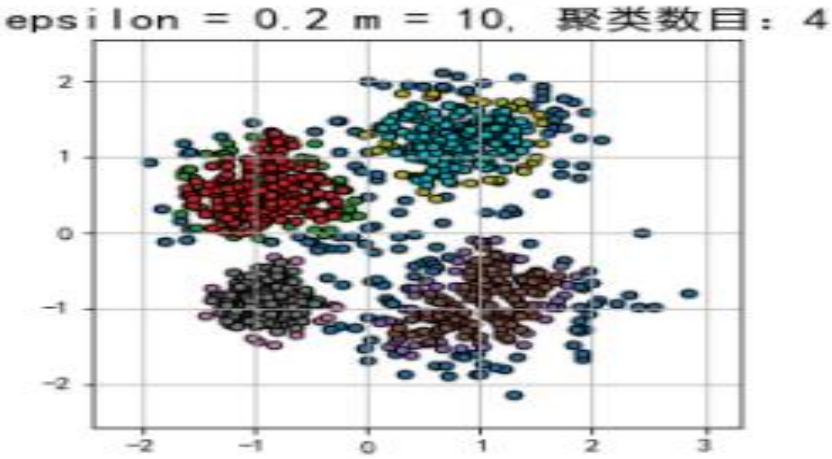
实际输出

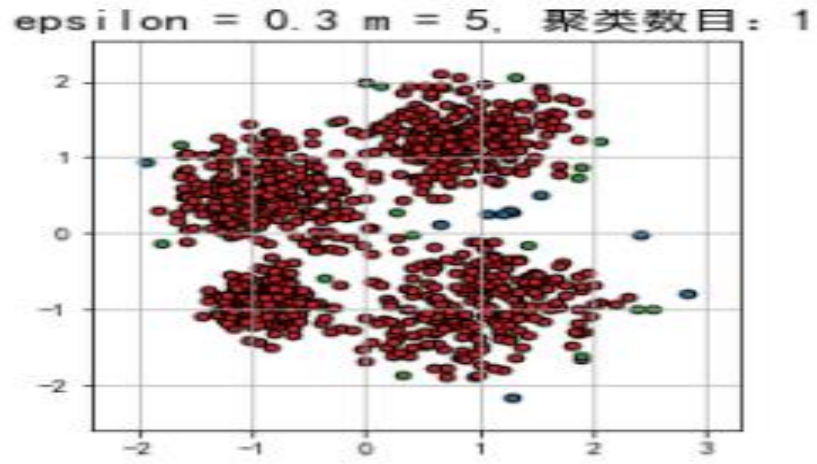
```
Traceback (most recent call last):
  File "DBScan/DBScanTest.py", line 40, in <module>
    main()
```

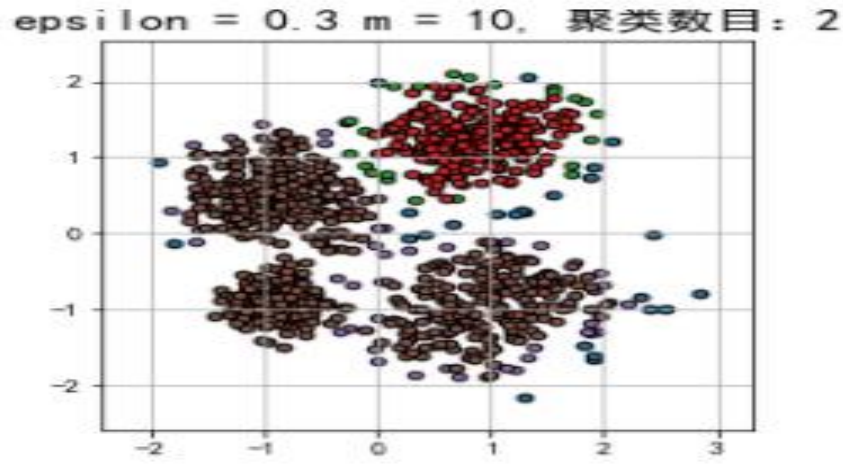
第5.4章 密度聚类方法

- 不同的ε，不同的*minpts*，聚类结果不一样







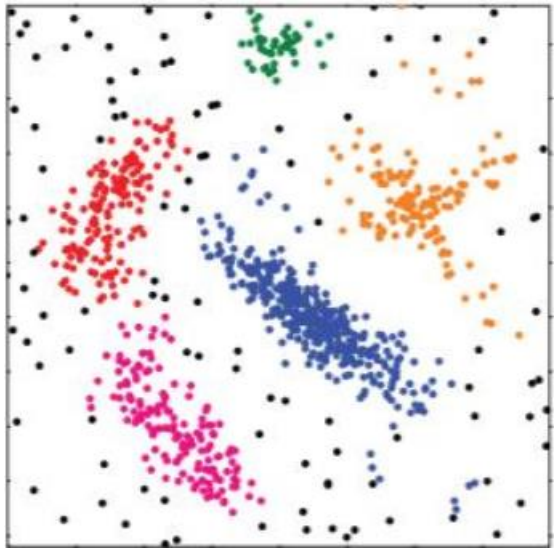


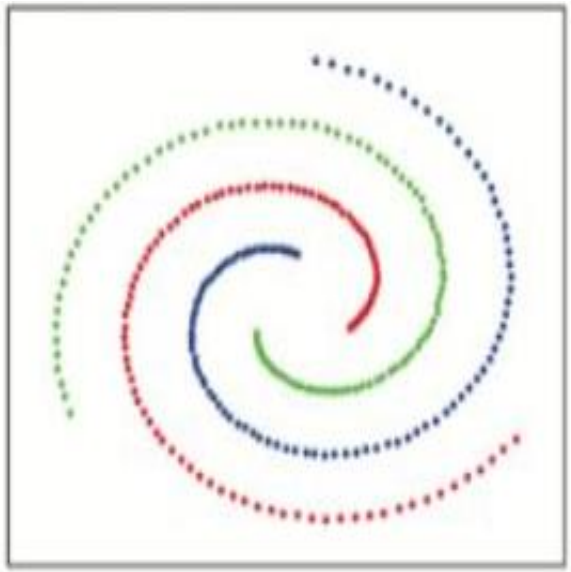
第5.4章 密度聚类方法

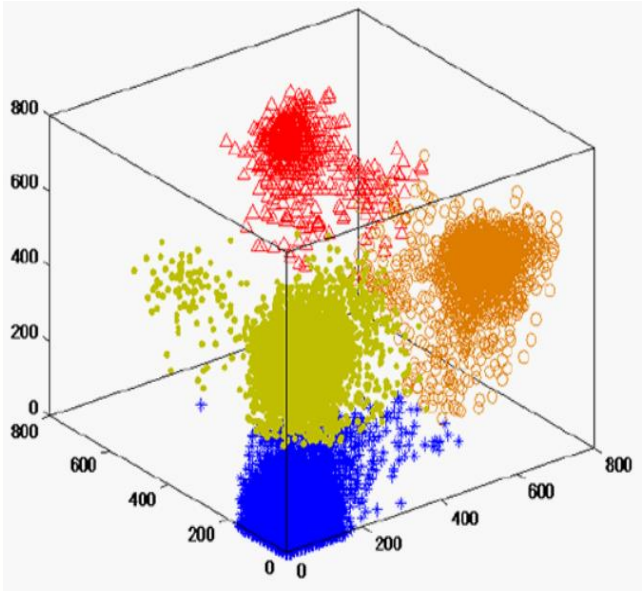
- 1. 密度聚类方法的概念
- 2. 密度聚类典型算法
- 3. 密度聚类算法描述
- 4. 密度聚类算法的性能分析
- 5. 密度聚类在数据挖掘中的应用

第5.4章 密度聚类方法

- DBSCAN算法， Density-Based Spatial Clustering of Applications with Noise，噪声环境下的密度聚类算法，将密度相连的点的最大集合聚成簇，并可在有“噪声”的空间数据库中发现任意形状的聚类。







- DBSCAN算法的性能分析：

如果采用空间索引，DBSCAN的计算复杂度是O($n \log n$)，这里n是数据库中对象的数目。否则，计算复杂度是O(n^2)

时间复杂度	一次邻居点的查询	DBSCAN
无索引	0(n)	0(n)
有索引	log n	n log n

第5.4章 密度聚类方法

- DBSCAN算法的优点：

（1）聚类速度快且能够有效处理噪声点和发现任意形状的空间聚类；

（2）与K-MEANS比较起来，不需要输入要划分的聚类个数；

（3）聚类簇的形状没有偏倚；

（4）对噪声数据不敏感。

第5.4章 密度聚类方法

- DBSCAN算法的缺点：

（1）当数据量增大时，要求较大的内存支持I/O消耗也很大；

（2）当空间聚类的密度不均匀、聚类间距差相差很大时，聚类质量较差，因为这种情况下参数MinPts和Eps选取困难。

（3）算法聚类效果依赖与距离公式选取，实际应用中常用欧式距离，对于高维数据，存在“维数灾难”。

第五章 聚类方法

内容提要

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 其它聚类方法

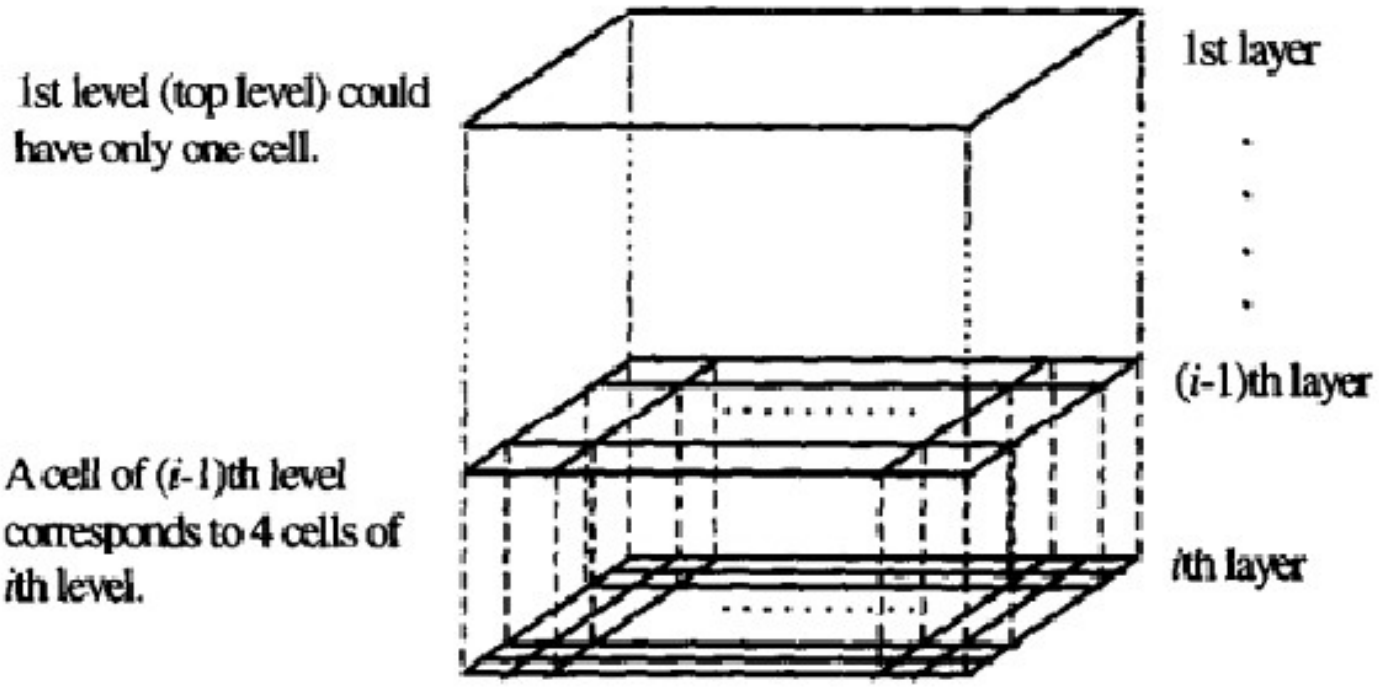
STING

- STING(Statistaical Information Grid_based method)是一种基于网格的多分辨率聚类技术，它将空间区域划分为矩形单元。针对不同级别的分辨率，通常存在多个级别的巨型单元，这些单元形成了一个层次结构：高层的每个单元被划分为多个第一层的单元。高层单元的统计参数可以很容易的从底层单元的计算得到。这些参数包括属性无关的参数*count*、属性相关的参数*m*（平均值）、*s*(标准偏差)、*min*(最小值)、*max*(最大值)以及该单元中属性值遵循的分布类型。

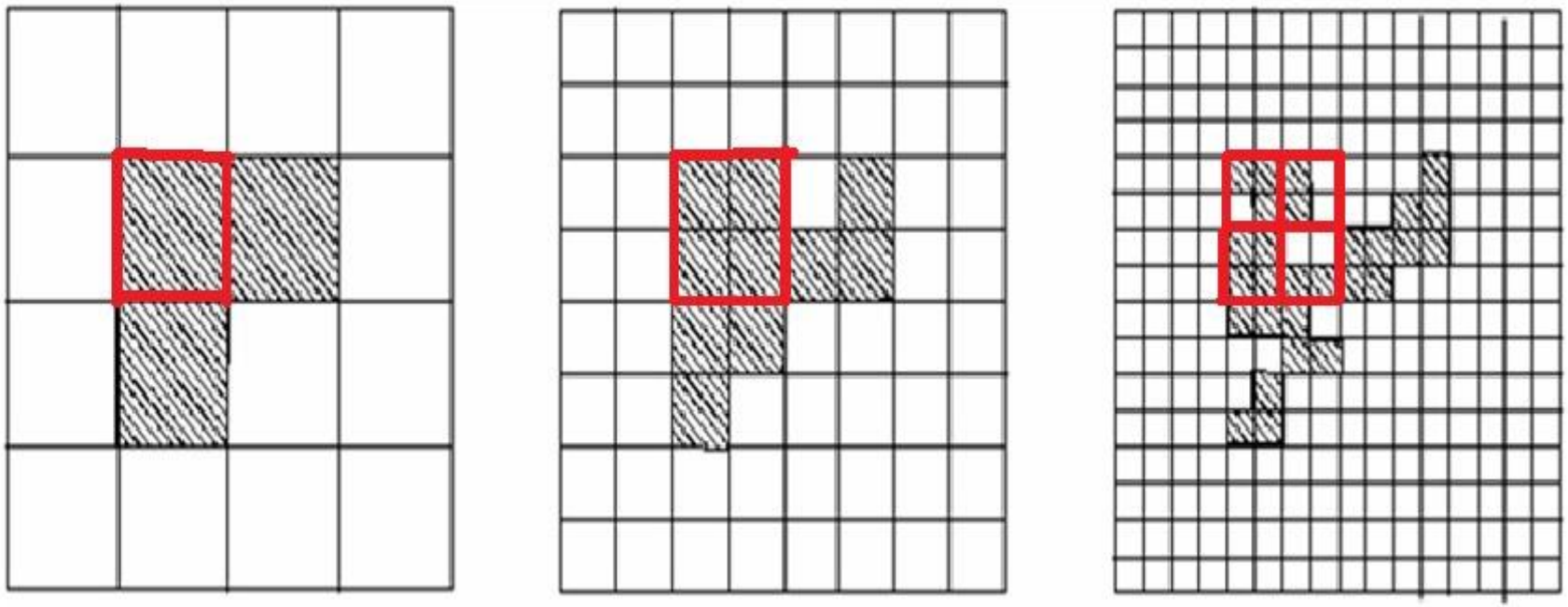
- STING算法的主要优点是效率高，通过对数据集的一次扫描来计算单元的统计信息，因此产生聚类的时间复杂度是$O(n)$。在建立层次结构以后，查询的时间复杂度是$O(g)$，g远小于n。STING算法采用网格结构，有利于并行处理和增量更新。

STING: 统计信息网格

STING聚类的层次结构



STING: 统计信息网格



level i　　level i+1　　level i+2

a cell of (i-1)th level corresponds to 4 cells of (i)th level

假设当前层的属性x的统计信息记为n, m, s, min, max, dist, 而n_i, m_i, s_i, min_i, max_i是相对于当前层来说，对应于更低一层的统计参数。那么n, m, s, min, max, dist可以用以下方法计算:

$$n = \sum_i n_i$$

$$m = \frac{\sum_i m_i n_i}{n}$$

$$s = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) n_i}{n} - m^2}$$

$$min = \min_i (min_i)$$

$$max = \max_i (max_i)$$

dist ?

The determination of *dist* for a parent cell is a bit more complicated. First, we set *dist* as the distribution type followed by most points in this cell. This can be done by examining $dist_i$ and n_i. Then, we estimate the number of points, say *confl*, that conflict with the distribution determined by *dist*, *m*, and *s* according to the following rule:

1. If $dist_i \neq dist$, $m_i \approx m$ and $s_i \approx s$, then *confl* is increased by an amount of n_i;
2. If $dist_i \neq dist$, but either $m_i \approx m$ or $s_i \approx s$ is not satisfied, then set *confl* to *n* (This enforces *dist* will be set to NONE later);
3. If $dist_i = dist$, $m_i \approx m$ and $s_i \approx s$, then *confl* is not changed;
4. If $dist_i = dist$, but either $m_i \approx m$ or $s_i \approx s$ is not satisfied, then *confl* is set to *n*.

Finally, if $\frac{confl}{n}$ is greater than a threshold *t* (This threshold is a small constant, say 0.05, which is set before the hierarchical structure is built), then we set *dist* as NONE; otherwise, we keep the original type. For example, the parameters of lower level cells are as follows.

STING: 统计信息网格

Table 1: Parameters of Children Cells

i	1	2	3	4
n_i	100	50	60	10
m_i	20.1	19.7	21.0	20.5
s_i	2.3	2.2	2.4	2.1
min_i	4.5	5.5	3.8	7
max_i	36	34	37	40
$dist_i$	NORMAL	NORMAL	NORMAL	NONE

Then the parameters of current cell will be

$$n = 220$$
$$m = 20.27$$
$$s = 2.37$$
$$min = 3.8$$
$$max = 40$$
$$dist = \text{NORMAL}$$

The distribution type is still NORMAL based on the following: Since there are 210 points whose distribution type is NORMAL, *dist* is first set to NORMAL. After examining $dist_i$, m_i, and s_i of each lower level cell, we find out $confl = 10$. So, *dist* is kept as NORMAL ($\frac{confl}{n} = 0.045 < 0.05$).

- SOM神经网络是一种基于模型的聚类方法。SOM神经网络由输入层和竞争层组成。
  - 输入层由N个输入神经元组成，竞争层由$m \times m = M$个输出神经元组成，且形成一个二维平面阵列。
  - 输入层各神经元与竞争层各神经元之间实现全互连接。
- 该网络根据其学习规则，通过对输入模式的反复学习，捕捉住各个输入模式中所含的模式特征，并对其进行自组织，在竞争层将聚类结果表现出来，进行自动聚类。竞争层的任何一个神经元都可以代表聚类结果。

SOM神经网络(续)

- 图1给出了SOM神经网络基本结构，图2给出了结构中各输入神经元与竞争层神经元j的连接情况。

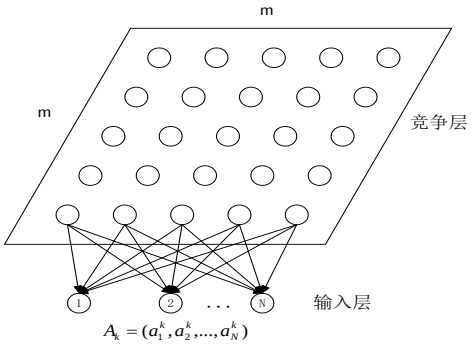


图1SOM网络基本结构

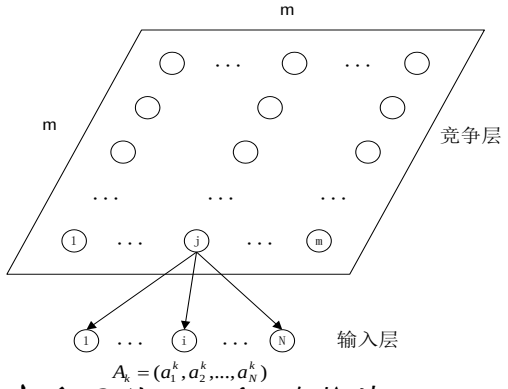


图2输入神经元与竞争层神经元j的连接情况

- 设网络的输入模式为 $k=1,2,\cdots,p$；竞争层神经元向量为$B_j=(b_{j1},b_{j2},\cdots,b_{jm})$，$j=1,2,\cdots,m$；其中$A_k$为连续值，$B_j$为数字量。网络的连接权为$\{w_{ij}\}$ $i=1,2,\cdots,N$；$j=1,2,\cdots,M$。
- SOM网络寻找与输入模式Ak最接近的连接权向量$W_g=(w_{g1},w_{g2},\cdots,w_{gN})$，将该连接权向量$W_g$进一步朝与输入模式$Ak$接近的方向调整，而且还调整邻域内的各个连接权向量$W_j$，$j\in N_g(t)$。随着学习次数的增加，邻域逐渐缩小。最终得到聚类结果。
- SOM类似于大脑的信息处理过程，对二维或三维数据的可视是非常有效的。SOM网络的最大局限性是，当学习模式较少时，网络的聚类效果取决于输入模式的先后顺序；且网络连接权向量的初始状态对网络的收敛性能有很大影响。

聚类分析在数据挖掘中的应用分析

- **聚类在数据挖掘中的典型应用有：**
  - **可以作为其它算法的预处理步骤：**利用聚类进行数据预处理，可以获得数据的基本概况，在此基础上进行特征抽取或分类就可以提高精确度和挖掘效率。也可将聚类结果用于进一步关联分析，以获得进一步的有用信息。
  - **可以作为一个独立的工具来获得数据的分布情况：**聚类分析是获得数据分布情况的有效方法。通过观察聚类得到的每个簇的特点，可以集中对特定的某些簇作进一步分析。这在诸如市场细分、目标顾客定位、业绩估评、生物种群划分等方面具有广阔的应用前景。
  - **可以完成孤立点挖掘：**许多数据挖掘算法试图使孤立点影响最小化，或者排除它们。然而孤立点本身可能是非常有用的。如在欺诈探测中，孤立点可能预示着欺诈行为的存在。

聚类分析在数据挖掘中的应用分析

- **聚类在数据挖掘中的典型应用有：**
  - **可以作为其它算法的预处理步骤：**利用聚类进行数据预处理，可以获得数据的基本概况，在此基础上进行特征抽取或分类就可以提高精确度和挖掘效率。也可将聚类结果用于进一步关联分析，以获得进一步的有用信息。
  - **可以作为一个独立的工具来获得数据的分布情况：**聚类分析是获得数据分布情况的有效方法。通过观察聚类得到的每个簇的特点，可以集中对特定的某些簇作进一步分析。这在诸如市场细分、目标顾客定位、业绩估评、生物种群划分等方面具有广阔的应用前景。
  - **可以完成孤立点挖掘：**许多数据挖掘算法试图使孤立点影响最小化，或者排除它们。然而孤立点本身可能是非常有用的。如在欺诈探测中，孤立点可能预示着欺诈行为的存在。