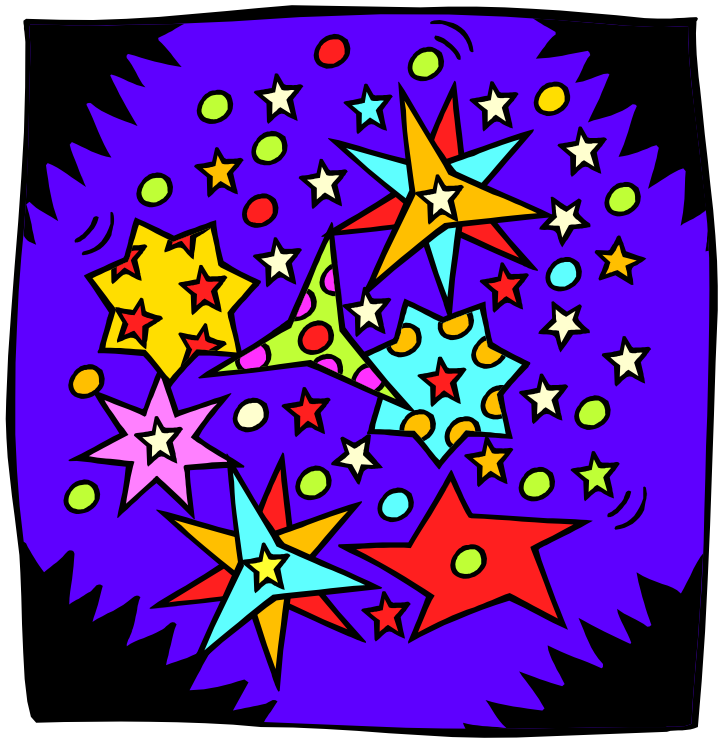
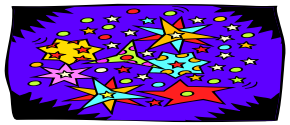
第五章 聚类方法

内容提要

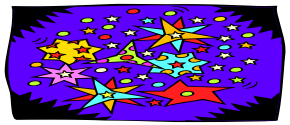
- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘





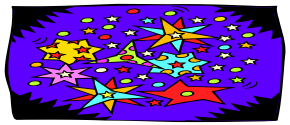
聚类分析研究概述

- 什么是聚类？
- 簇：数据对象的集合
  - 一个簇内的数据尽量相似（high intra-class similarity）
  - 不同簇的数据尽量不相似（low inter-class similarity）
- 聚类分析
  - 把大型数据划分成不同的簇
- 聚类是无监督学习：没有事先定义好的类别
- 典型应用
  - 作为一个独立工具深入了解数据分布情况
  - 作为其它算法的预处理步骤
  - 聚类分析可以完成孤立点挖掘



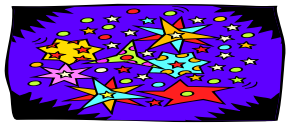
聚类的通用应用场景

- 空间数据分析
  - 通过对特征空间聚类创建主题地图
  - 在空间数据挖掘中发现并解释空间簇
- 模式识别
- 经济学（尤其是市场研究）
- 互联网
  - 文本分类
  - 聚类博客数据，发现相似群组



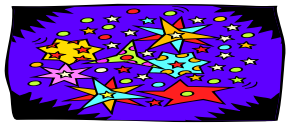
聚类应用的示例

- 市场营销：帮助营销人员帮他们发现顾客中独特的群组，然后利用他们的知识发展目标营销项目
- 土地利用：在土地观测数据库中发现相似的区域
- 保险：识别平均索赔额度较高的机动车辆保险客户群组
- 城市规划：通过房屋的类型、价值、地理位置识别相近的住房
- 地震研究：沿着大陆断层聚类地震的震中



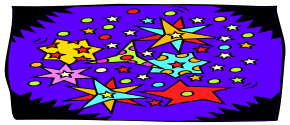
好的聚类方法

- 好的聚类方法需要产生高质量的聚类结果，这些簇必须满足：
  - 高的内部相似度
  - 低的外部相似度
- 聚类算法结果的质量取决于方法中所用的相似性度量，以及实现细节
- 聚类算法结果的质量同样可以通过其发现部分或者全部隐藏模式的能力来评估



聚类的性能指标

- 可伸缩性
- 处理不同类型属性的能力
- 发现任意形状的聚类
- 对于决定输入参数的领域知识需求最小
- 处理带噪声数据的能力
- 增量聚类和对输入记录的次序不敏感
- 高维性
- 基于约束的聚类
- 可解释性和可用性



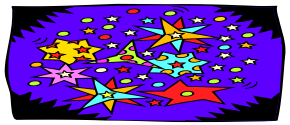
聚类分析的数据结构

- **数据矩阵**
  - 被聚类的数据的一种表示方式
  - 一共n个数据，每个数据有m维
- **相异度矩阵**
  - 存放n个对象之间的距离
- **聚类分析中的数据类型**
  - 区间标度变量（如165cm）
  - 二元变量（0或1）
  - 分类变量、序数变量

  和比例标度变量
  - 混合类型变量

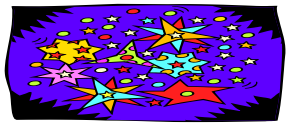
$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$



聚类分析的数据规范化

- 规范化确保变量的变化范围相等
- 最大值归一化
  - 每一维属性除以该属性的绝对值最大值，-1~1
- 总和规范化
  - 数据对象各分量除以全体在这个分量上的总和
- 均值标准差规范化
  - 适合正态分布的数据
- 极差规范化
  - 数据的最大值为1，最小值为0



聚类分析方法的分类

- 按照聚类聚类分析算法的主要思路，可以被归纳为如下几种。
  - 划分法（Partitioning Methods）：

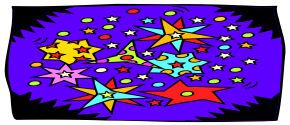
    基于一定标准构建数据的划分。属于该类的聚类方法有：k-means、k-modes、k-prototypes、k-medoids、PAM、CLARA、CLARANS等。
  - 层次法（Hierarchical Methods）：

    对给定数据对象集合进行层次的分解。
  - 密度法（density-based Methods）：

    基于数据对象的相连密度评价。
  - 网格法（Grid-based Methods）：

    将数据空间划分成为有限个单元（Cell）的网格结构，基于网格结构进行聚类。
  - 模型法（Model-Based Methods）：

    给每一个簇假定一个模型，然后去寻找能够很好的满足这个模型的数据集。



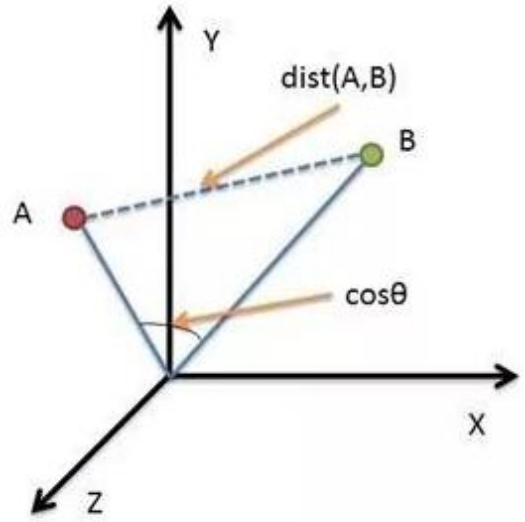
评估聚类的质量

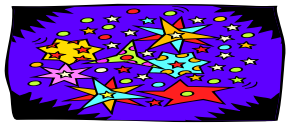
- 相异性/相似性 度量：相似性通常用一个距离函数来表示，通常为：　　$d(i, j)$
- 有一个单独的“质量”函数用于度量聚类的好坏.
- 对于区间标度，布尔，分类，序数以及比率等变量，距离函数的定义通常是不同的.
- 不同的应用以及数据语义，通常给变量赋与不同的权重.
- 很难定义“足够相似”或者“足够好”
  - 答案往往带有典型的主观性.

- 通常用距离来衡量两种数据对象之间的相似性与相异性
- 一些常用的包括:明可夫斯基距离（*Minkowski distance*）：

$$d(i,j)=\sqrt[q]{(|x_{i_1}-x_{j_1}|^q+|x_{i_2}-x_{j_2}|^q+...+|x_{i_p}-x_{j_p}|^q)}$$

其中 $i=(x_{i1}, x_{i2}, \cdots, x_{ip})$，$j=(x_{j1}, x_{j2}, \cdots, x_{jp})$ 是两个P维数据对象，q是一个正整数





区间标度变量的距离与相似性

- 如果 $q = 1$, d 是曼哈顿距离

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ip} - x_{jp}|$$

- 如果$q = 2$, d 是欧氏距离:

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2)}$$

  - 属性
    - ***d(i,j)*** ≥ **0**
    - ***d(i,i)*** = **0**
    - ***d(i,j)*** = ***d(j,i)***
    - ***d(i,j)*** ≤ ***d(i,k)*** + ***d(k,j)***

- 同样，还可以使用加权距离，皮尔逊积差相关系数，或者 其它相异度度量

- 一个二元变量的相依表

		Object *j*		
		1	0	*sum*
Object *i*	1	a	b	$a+b$
	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- 简单匹配系数(对称的二元变量，例如性别男和女，同等权重)：

$$d(i,j)=\frac{b+c}{a+b+c+d}$$

- Jaccard 系数 (不对称的二元变量，比如某疾病检测阳性和阴性)

$$d(i,j)=\frac{b+c}{a+b+c}$$

- 示例

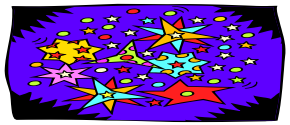
Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- 性别是对称属性
- 其它属性都是非对称属性
- 假设Y和P编码为1，N为0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

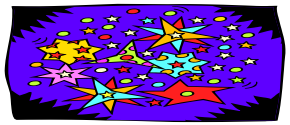


标称变量的距离与相似性

- 分类变量是二元变量的推广，可以取对于两个状态值，如颜色变量（红、橙、黄、绿、蓝等）
- Method 1: 简单匹配
  - *m*: 匹配的数目，（即对象 i 和 j 取值相同的属性数）
  - *p*: 变量总数，（即刻画对象的属性总数）

$$d(i,j)=\frac{p-m}{p}$$

  - *取值不同的同位属性数/单个元素的全部属性数*
- Method 2:使用大量的二元变量
  - 为M的每个状态创建一个新的二元变量



序数变量的距离与相似性

- 序数变量可以是离散的或者连续的
- 顺序很重要,如冠军、亚军和季军
- 对于序数变量，一般为每个值分配一个数，叫做这个值的秩，然后以秩代替原值当做标量属性计算相异度。
- 可以当成区间标度变量处理
  - 通过 x_{if} 的秩替换它 $r_{if} \in \{1,...,M_f\}$
  - 将每个变量的值映射到[0, 1]， 通过替换第f个变量的第i个对象的秩

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - 利用区间标度变量的相异度来计算

比例标度变量的距离与相似性

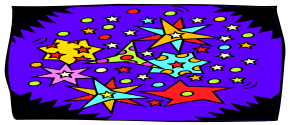
- 比例标度变量:在非线性刻度取正的度量值，近似的遵循指数标度，如下公式：
  - Ae^{Bt} or Ae^{-Bt}

A和B为正的常数，例如细菌增长的数据描述，放射元素的衰减

- 方法：
  - 首先采用与区间标度变量相同的方法（why?—刻度可能扭曲了）
  - 再使用对数变换

$$y_{if} = log(x_{if})$$

  - 将其看成连续的序数数据，将其秩作为区间值来处理



混合类型变量的距离度量

- 一个数据库可能包含所有这六种类型的变量
  - 对称二元的，非对称二元的，分类的，序数的，区间标度的，以及比例标度的
- 可以使用一个带权重的公式将这些变量的影响整合起来

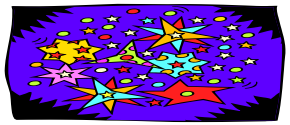
$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}}$$

  - f 是二元或者分类的：
    如果$x_{if} = x_{jf}$，$d_{ij}^{(f)} = 0$，否则 $d_{ij}^{(f)} = 1$
  - f 是区间标度的：使用区间标度距离
  - f 是序数或者比例标度的
    - **计算秩 r_{if}**
    - **将 z_{if} 当成区间标度处理**

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
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘



划分聚类算法的主要思想

- 划分方法：给定n个对象的数据集D，以及要生成的簇的数目k，划分方法将对象组织为k个划分
- 给定一个k，找到一个k个簇的划分，使得划分的准则最优
  - 全局最优：枚举所有的划分
  - 启发式方法：*k-means* 和 *k-medoids* 算法
  - *k-means*（k-均值）：每个簇都用簇中心表示
  - *k-medoids*（K-中心点）or PAM（围绕中心的划分）：每个簇用簇中的一个对象表示

- 给定k，生成K个非空子集
  - *k-means*算法包含4个步骤:

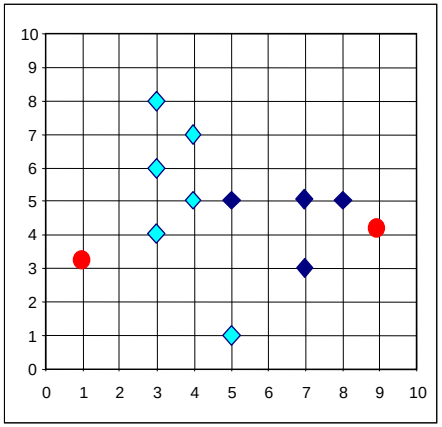
1）随机指定k个对象作为中心点，进行聚类

2）得到k个簇，计算簇平均值

3）根据簇中对象的均值，将每个对象重新分配到最相似的簇 更新簇均值

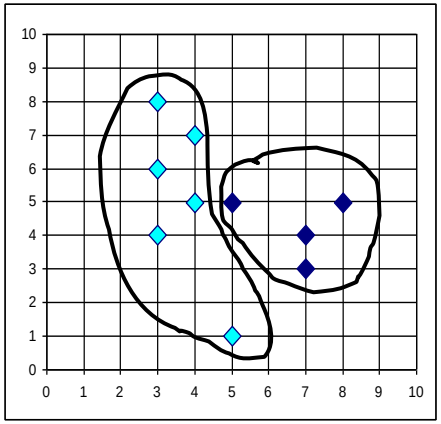
4）重复第3步，直到不再发生变化

■ 示例

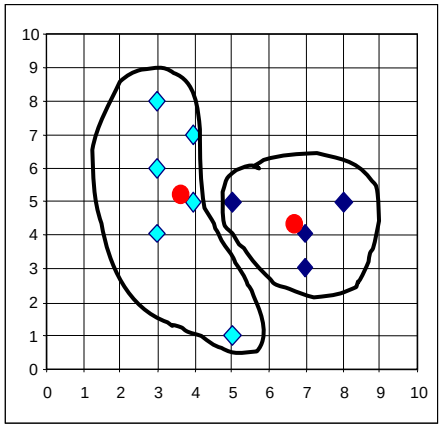


K=2
Arbitrarily choose K object as initial cluster center

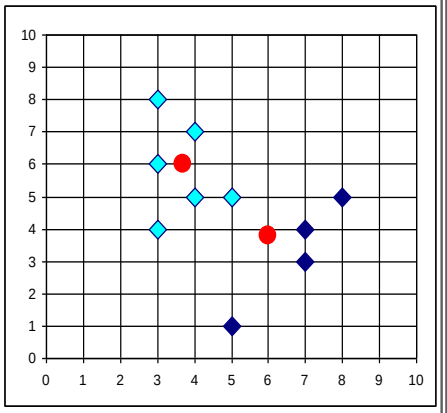
Assign each objects to most similar center



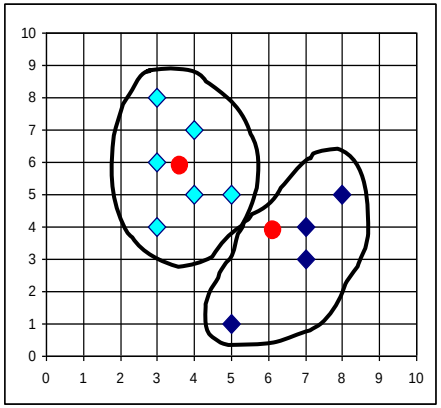
Update the cluster means



reassign



Update the cluster means



reassign

- 根据所给的数据通过对其进行k-平均算法（设n=8，k=2），进行分类。（指定1、3为初值中心点）

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

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。

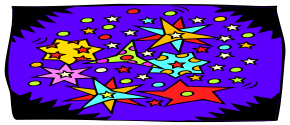
对于{1，2}，平均值点为（1.5，1）（这里的平均值是简单的相加出2）；

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）



k-means算法的性能分析

- **主要优点：**
  - 效率相对较高:复杂度为 O(迭代数*k*对象数)
  - 是解决聚类问题的一种经典算法，简单、快速。
  - 对处理大数据集，该算法是相对可伸缩和高效率的。
  - 当结果簇是密集的，它的效果较好。
- **主要缺点**
  - 在簇的平均值被定义的情况下才能使用，可能不适用于某些应用。比如如何处理分类属性数据?
  - 必须事先给出k（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸面形状的簇或者大小差别很大的簇。
  - 它对于“噪声”和孤立点数据是敏感的。

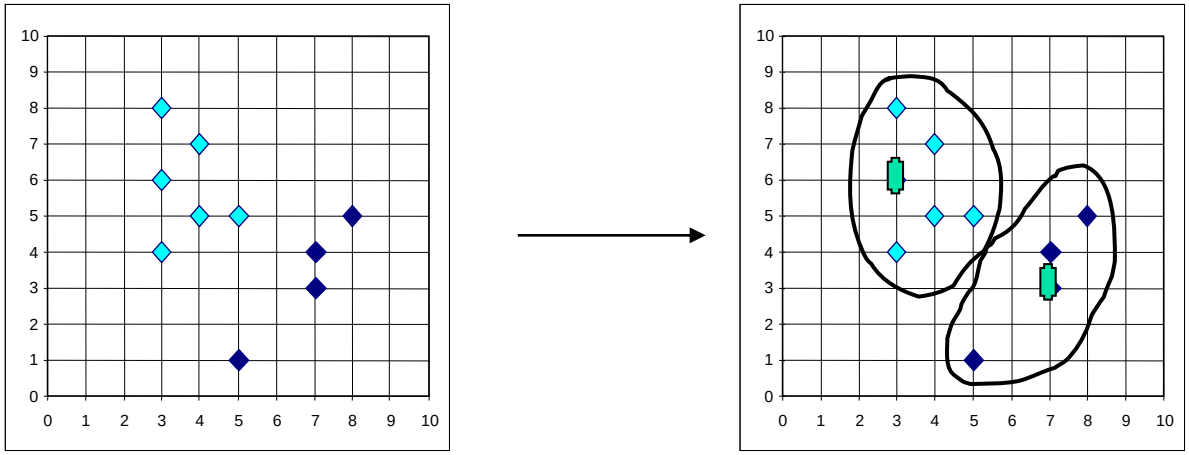
k-means算法作业

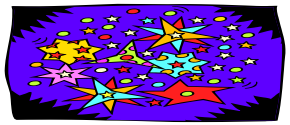
- 根据所给的数据通过对其进行k-平均算法（设n=8，k=3），进行分类。（指定1、4、7为初值中心点）

序号	属性1	属性2
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

{1,4,8}、{3,5,6}和{2,7}

- k-means算法对离群点比较敏感!
  - 一个对象有一个特别大的极端值可能对数据分布产生显著的扭曲.
- k-中心点算法，不采用簇中的平均值作为参照点，可以选用簇中位置最中心的对象，即中心点作为参照点。这样划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的。



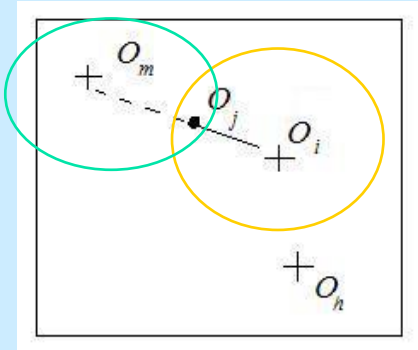


PAM算法基本思想

- PAM作为最早提出的k-中心点算法之一，它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。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。

- 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。

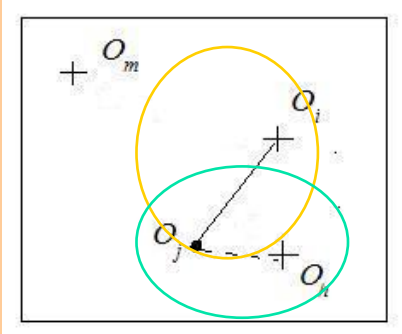
第一种情况



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

- 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。

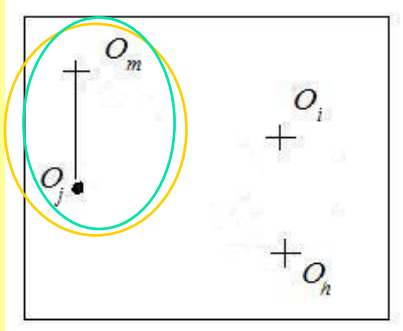
第二种情况



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

- 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。

第三种情况

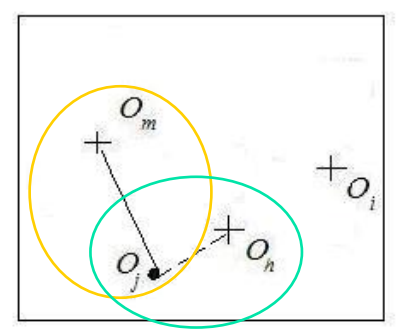


O_j的隶属不发生变化，

$C_{jih} = 0$

- 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。

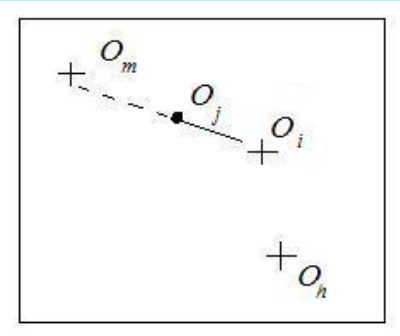
第四种情况



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

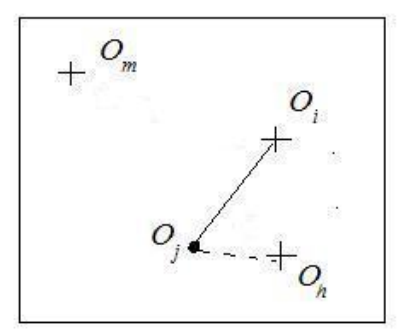
PAM算法代价函数的四种情况

第一种情况



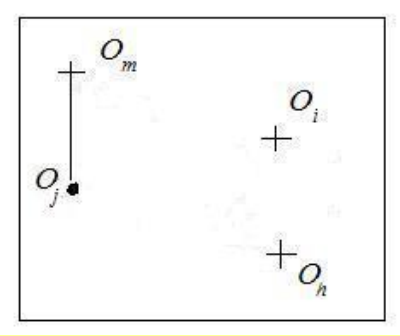
O_j被重新分配给O_m,
$C_{jih} = d(j, m) - d(j, i)$

第二种情况



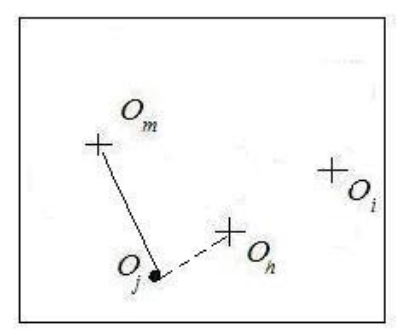
O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, i)$

第三种情况

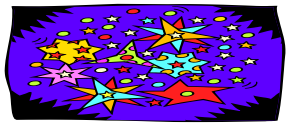


O_j的隶属不发生变化，
$C_{jih} = 0$

第四种情况



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



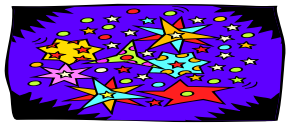
PAM算法基本思想(续)

- 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对象被非中心点对象所代替，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。
  - 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。

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。前面我们已经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。



PAM算法基本思想(续)

算法5-2 PAM（*k*-中心点算法）

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；

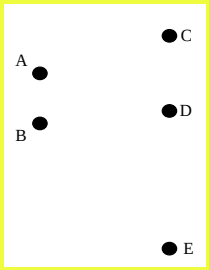
（11） IF 在*S*中的所有非中心点代替所有中心点后的计算出的总代价有小于0的存在 THEN 找出*S*中的用非中心点替代中心点后代价最小的一个，并用该非中心点替代对应的中心点，形成一个新的*k*个中心点的集合；

（12）UNTIL 没有再发生簇的重新分配，即所有的*S*都大于0.

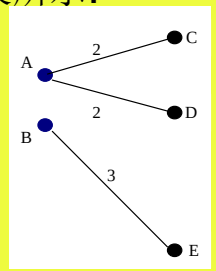
PAM算法基本思想(续)

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	3
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。

第二步 交换阶段：假定中心点A、B分别被非中心点}和{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}。

我们以$TCAC$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$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$。

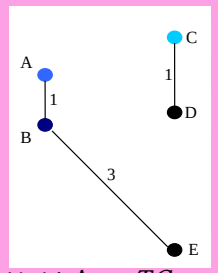
d）D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，根据PAM算法代价函数的第二种情况$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$。

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$C_{EAC}=0$。

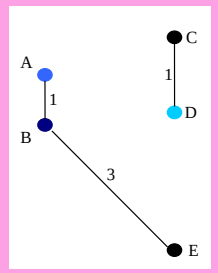
因此，$TC_{AC}=CA_{AC}+CB_{AC}+CB_{AC}+CD_{AC}+CE_{AC}=1+0-2-1+0=-2$。

PAM算法基本思想(续)

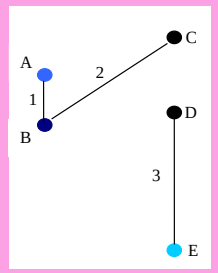
在上述代价计算完毕后，我们要选取一个最小的代价，显然有多种替换可以选择，我们选择第一个最小代价的替换（也就是C替换A），根据图5-4（a）所示，样本点被划分为{ B、A、E}和{C、D}两个簇。图5-4（b）和图5-4（c）分别表示了D替换A，E替换A的情况和相应的代价



(a) C替换A, TC_{AC}=-2



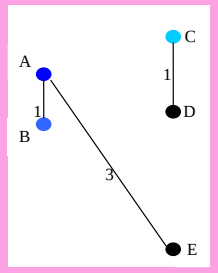
(b) D替换A, TC_{AD}=-2



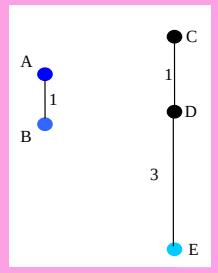
(c) E替换A, TC_{AE}=-1

图5-4 替换中心点A

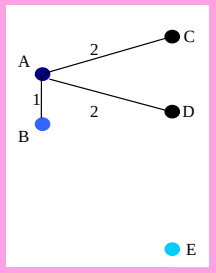
图5-5（a）、（b）、（c）分别表示了用C、D、E替换B的情况和相应的代价。



(a) C替换B, TC_{BC}=-2



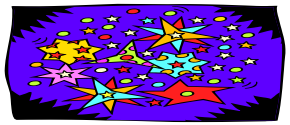
(b) D替换B, TC_{BD}=-2



(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的一些问题?

- 当存在噪声和离群点时，PAM比k-means方法更鲁棒，因为中心点不像均值那样容易受到离群点或者其它极值的影响
- PAM在小数据集上执行效率高，但是在大数据集上的伸缩性不好，每轮迭代$O(k(n-k)^2)$
- 基于抽样的算法
  - CLARA和CLARANS
  - 抽取数据集合的多个样本，将*PAM* 应用到每个样本,将最好的聚类结果作为输出，比*PAM*能处理更大规模的数据集
  - 不足:

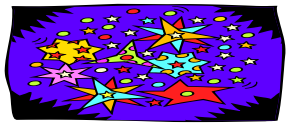
--有效性取决于样本大小

--如果样本是有偏的，基于抽样的好的聚类不一定代表整个数据集的好的聚类

第五章 聚类方法

内容提要

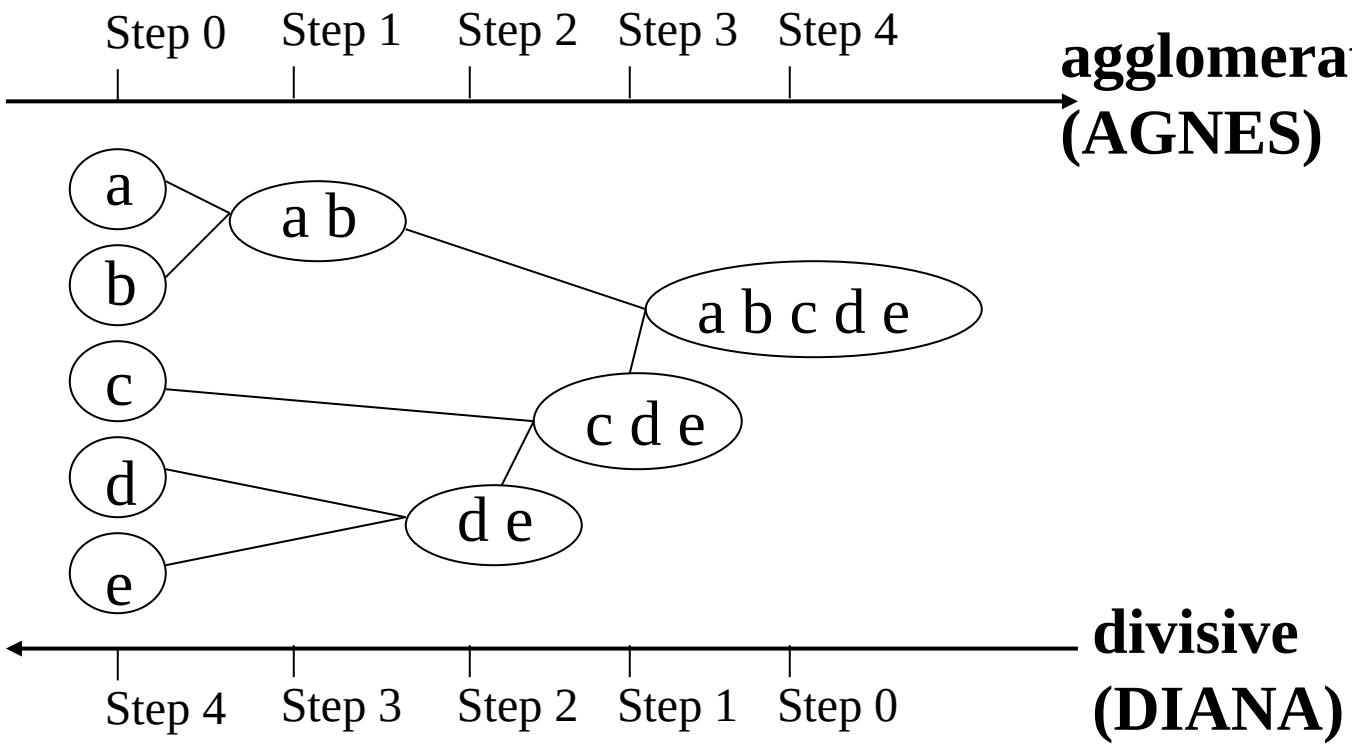
- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘

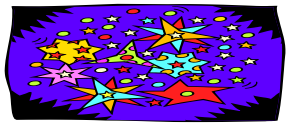


层次聚类方法概述

- 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。
  - 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。
- 层次凝聚的代表是AGNES算法。层次分裂的代表是DIANA算法。

- 使用距离矩阵作为聚类度量准则，需要一个终止条件(可以是k个簇)





层次聚类类间距离

- 距离函数都是关于两个样本的距离刻画，然而在聚类应用中，最基本的方法是计算类间（簇间）的距离。
- 类间距离的度量主要有：
  - 最短距离法：定义两个类中最靠近的两个元素间的距离为类间距离。
  - 最长距离法：定义两个类中最远的两个元素间的距离为类间距离。
  - 中心法：定义两类的两个中心间的距离为类间距离。
  - 类平均法：它计算两个类中任意两个元素间的距离，并且综合他们为类间距离

- AGNES (AGglomerative NESting)算法最初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簇数目。

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；

- 在所给数据集上运行AGNES算法，设n=8，用户输入终止条件为两个簇。

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。

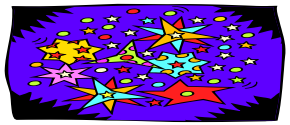
第3步：重复第2步的工作，5，6点成为一簇。

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。

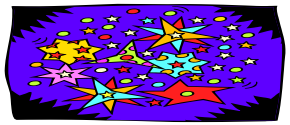
第6步：合并{5，6}，{7，8}，由于合并后的簇的数目已经达到了用户输入的终止条件程序结束。

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。
- 在聚类中，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.

- 在所给数据集上运行DIANA算法，设n=8，用户输入终止条件为两个簇。

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

DIANA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，找到具有最大直径的簇，对簇中的每个点计算平均相异度（假定采用是欧式距离）。

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

类似地，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；5的平均距离为2.18；6的平均距离为2.68；7的平均距离为2.526；8的平均距离为2.96。

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。

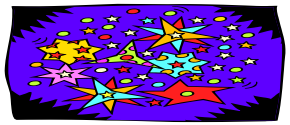
步骤	具有最大直径的簇	splinter group	Old party
1	{1，2，3，4，5，6，7，8}	{1}	{2，3，4，5，6，7，8}
2	{1，2，3，4，5，6，7，8}	{1，2}	{3，4，5，6，7，8}
3	{1，2，3，4，5，6，7，8}	{1，2，3}	{4，5，6，7，8}
4	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8}
5	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8} 终止

AGNES算法&DIANA算法作业

- 在所给数据集上分别运行AGNES算法以及DIANA算法，假定算法的终止条件为3个簇。

序号	属性1	属性2
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

- 思考：两种算法的结果是否一致
- {4，8，1}，{3，5，6}，{2，7}
- {2，7}，{1}，{3，4，5，6，8}



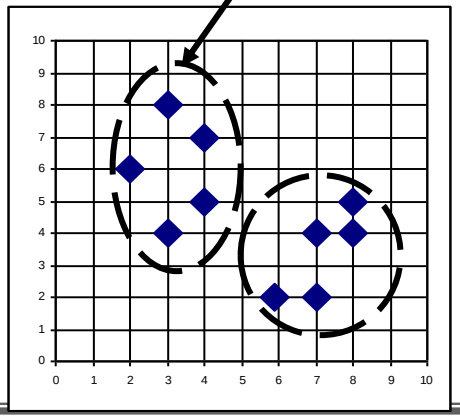
层次聚类方法的改进

- 层次聚类方法尽管简单，但经常会遇到合并或分裂点的选择的困难。
- 可伸缩性不足：时间复杂度至少 $O(n^2)$，其中 n 是对象总数量
- 不能回溯
- 改进层次方法的聚类质量的一个有希望的方向是将层次聚类和其他聚类技术进行集成，形成多阶段聚类。
- 下面介绍两个改进的层次聚类方法CURE和BIRCH。

- BIRCH（利用层次方法的平衡迭代归约和聚类）是一个综合的层次聚类方法，它用聚类特征和聚类特征树（CF）来概括聚类描述。
- 聚类特征CF是一个三元组：（N，LS，SS）其中
  - N：这个CF中拥有的样本点的数量
  - LS：这个CF中拥有的样本点各特征维度的和向量
  - SS：这个CF中拥有的样本点各特征维度的平方和

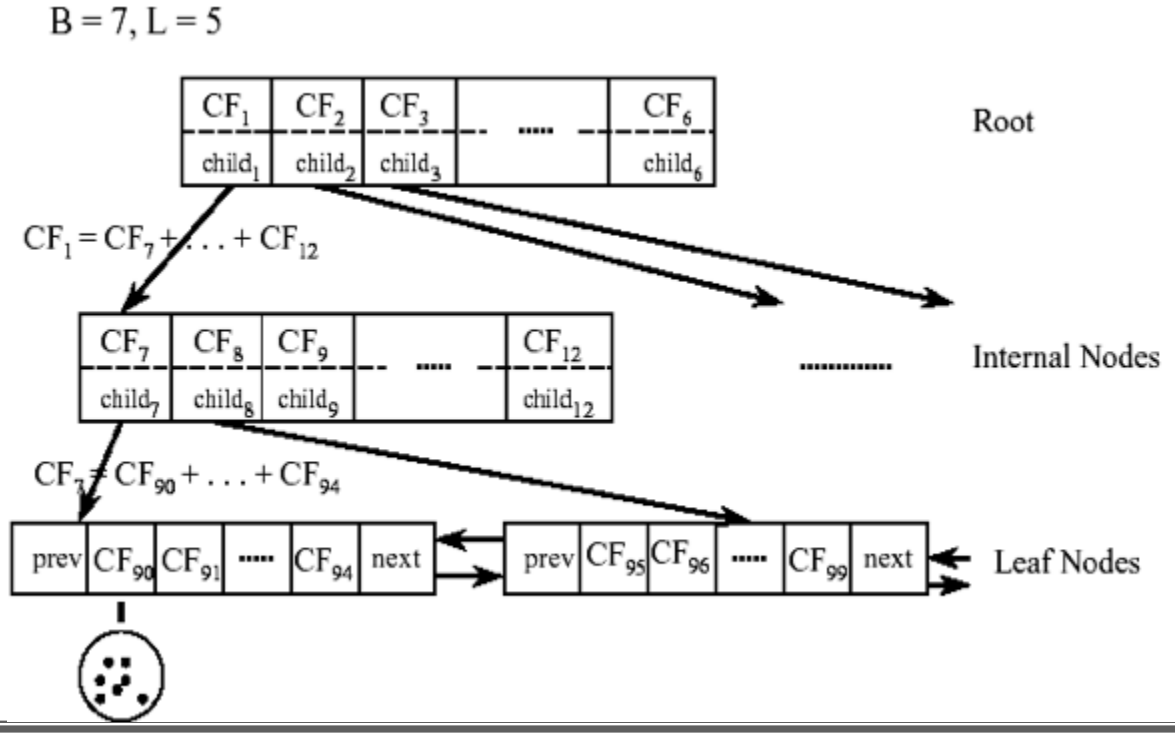
$$CF1 + CF2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

CF = (5, (16,30),(54,190))

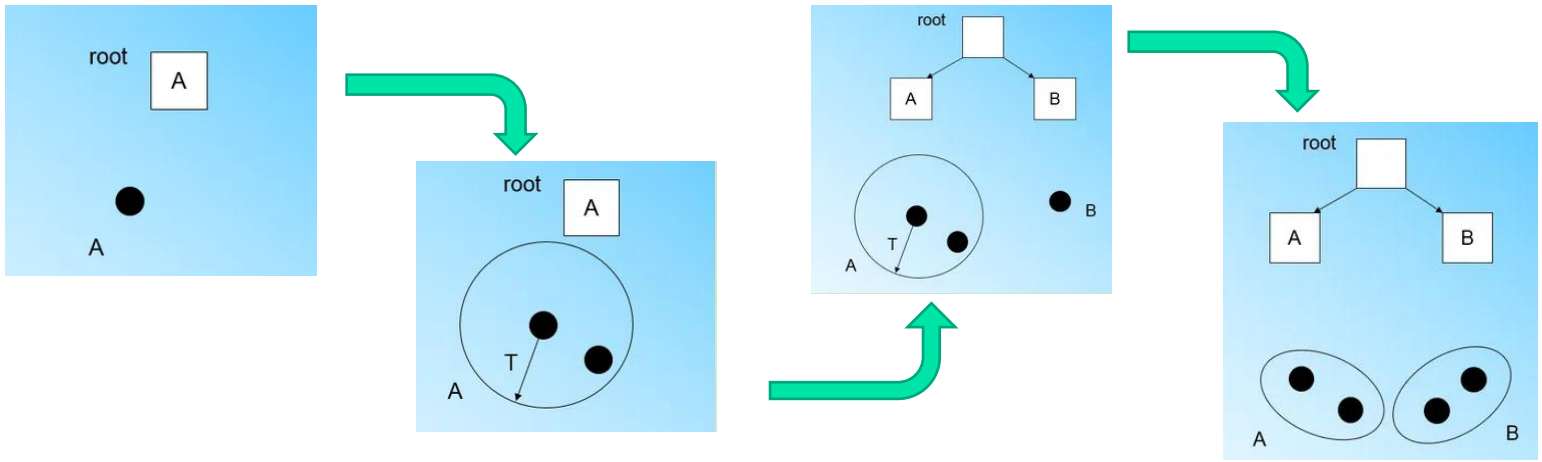


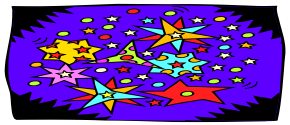
(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

- CF树具有三个参数：内部节点平衡因子B，叶节点平衡因子L和簇半径阈值T。
  - 内部节点平衡因子B定义了每个非叶节点孩子的最大数目
  - 叶节点平衡因子L定义了每个叶子节点的最大CF数
  - 阈值T给出了存储在树的叶子节点中每个CF（子聚类）的最大直径



- BIRCH算法的工作过程包括两个阶段：
  - 阶段一：BIRCH扫描数据库，建立一个初始存放于内存的CF树，它可以被看作数据的多层压缩，试图保留数据内在的聚类结构。随着对象的插入，CF树被动态地构造，不要求所有的数据读入内存，而可在外存上逐个读入数据项。因此，BIRCH方法对增量或动态聚类也非常有效。
  - 阶段二：BIRCH采用某个聚类算法对CF树的叶结点进行聚类。在这个阶段可以执行任何聚类算法，例如典型的划分方法。
- BIRCH算法试图利用可用的资源来生成最好的聚类结果。通过一次扫描就可以进行较好的聚类，故该算法的计算复杂度是$O(n)$，n是对象的数目。

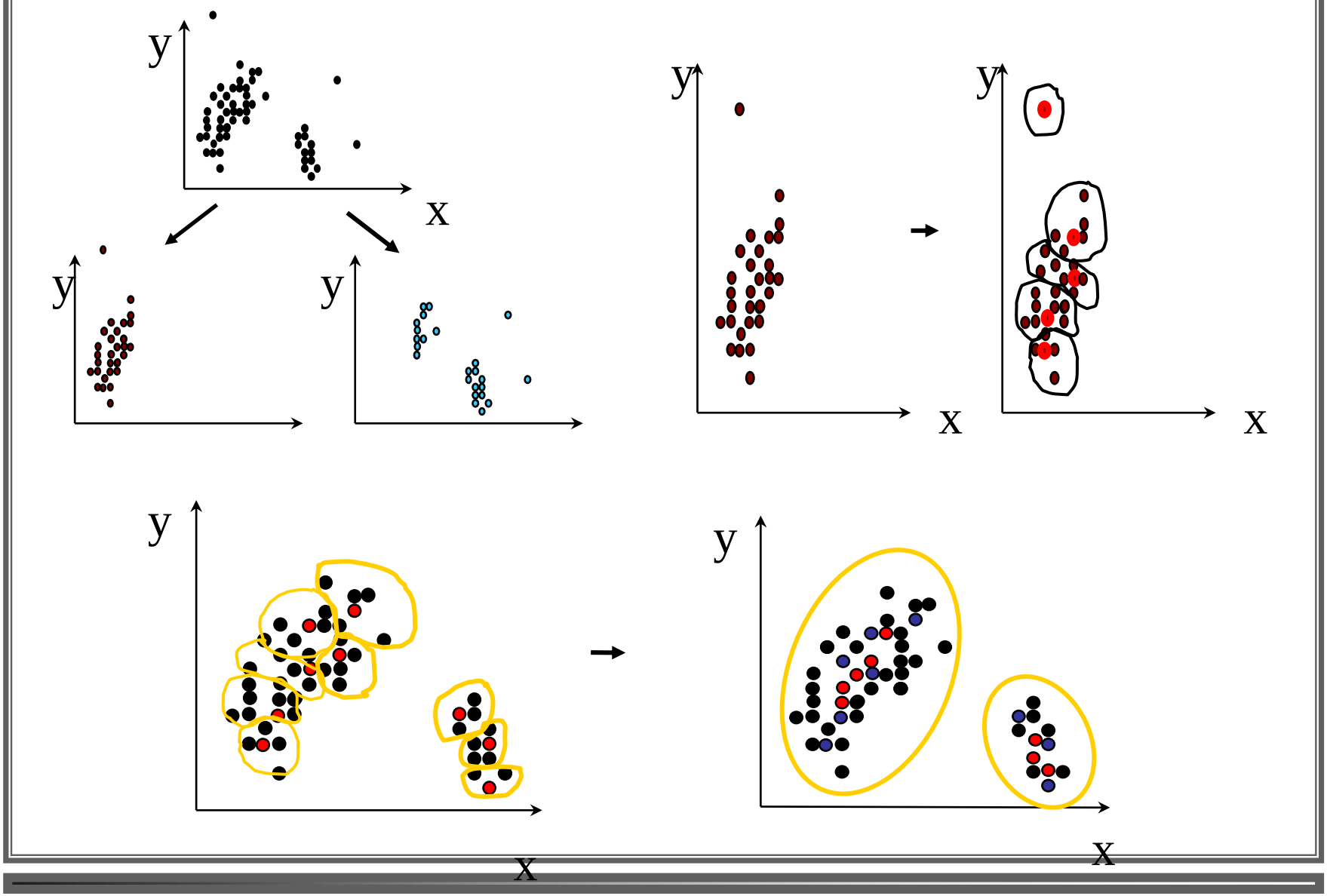




层次聚类方法的改进--CURE

- 很多聚类算法只擅长处理球形或相似大小的聚类，另外有些聚类算法对孤立点比较敏感。CURE算法解决了上述两方面的问题。
- CURE算法采用随机取样和划分两种方法的组合，具体步骤如下：
  - 从源数据集中抽取一个随机样本。
  - 为了加速聚类，把样本划分成p份，每份大小相等。
  - 对每个划分局部地聚类。
  - 根据局部聚类结果，对随机取样进行孤立点剔除。主要有两种措施：如果一个簇增长得太慢，就去掉它。在聚类结束的时候，非常小的类被剔除。
  - 对上一步中产生的局部的簇进一步聚类。落在每个新形成的簇中的代表点根据用户定义的一个收缩因子α收缩或向簇中心移动。这些点代表和捕捉到了簇的形状。
  - 用相应的簇标签来标记数据。
- CURE的复杂度是$O(n)$，n是对象的数目，所以该算法适合大型数据的聚类。

层次聚类方法的改进--CURE



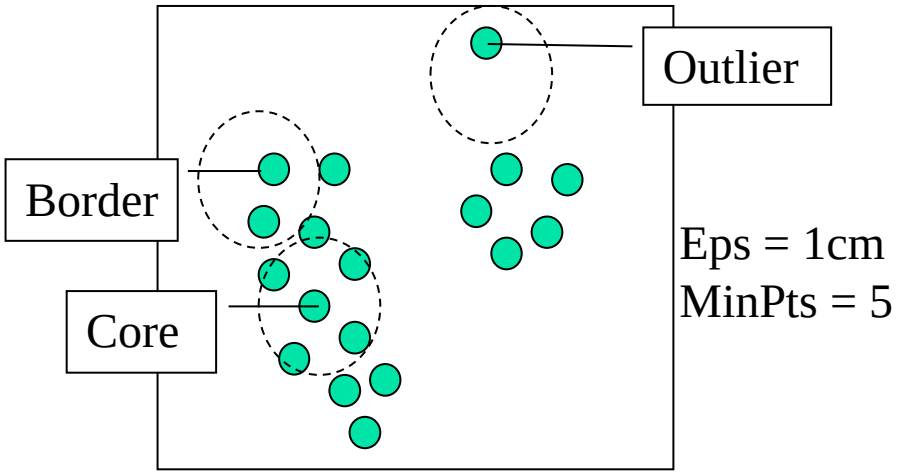
第五章 聚类方法

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
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘

- 基于划分的聚类—球形簇
- 基于层次的聚类—链状簇
- 密度聚类方法的指导思想是，只要一个区域中的点的密度大于某个域值，就把它加到与之相近的聚类中去。这类算法能发现任意形状的聚类，且对噪声数据不敏感。
- 算法：DBSCAN、OPTICS、DENCLUE算法等。
- DBSCAN（Density-Based Spatial Clustering of Applications with Noise）一个比较有代表性的基于密度的聚类算法。

它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在有“噪声”的空间数据库中发现任意形状的聚类。



- 对象的 ε/ep'silon/ -临域：给定对象在半径 ε 内的区域。
- 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。
- 例如，在图5-6中， ε=1cm，MinPts=5，q是一个核心对象。
- 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。
- 例如，在图5-6中， ε=1cm，MinPts=5，q是一个核心对象，对象p从对象q出发是直接密度可达的。

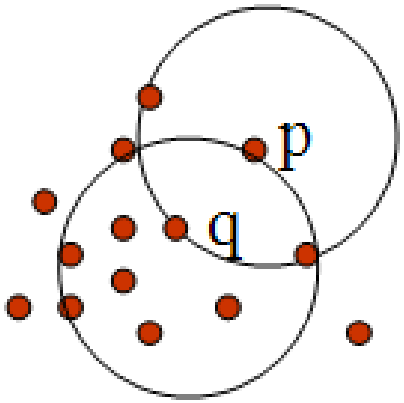


图5-6直接密度可达

- **密度可达的**：如果存在一个对象链p_1，p_2，…，p_n，$p_1=q$，$pn=p$，对$pi \in D$，（$1<=i<=n$），$pi+1$是从pi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密度可达的。
- 例如，在图5-6中， ε=1cm，MinPts=5，q是一个核心对象，$p1$是从q关于 ε和MitPts直接密度可达，p是从p_1关于 ε和MitPts直接密度可达，则对象p从对象q关于 ε和MinPts密度可达的。
- **密度相连的**：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密度可达的，那么对象p和q是关于 ε和MinPts密度相连的。
- 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”。

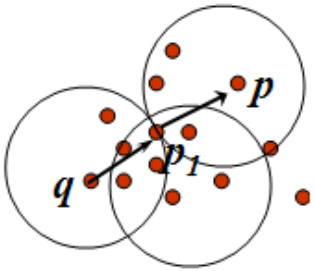


图5-6密度可达

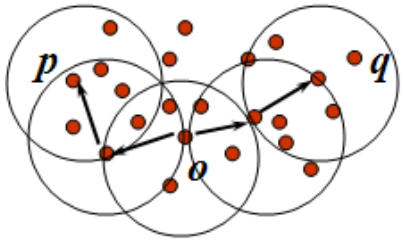


图5-8密度相连

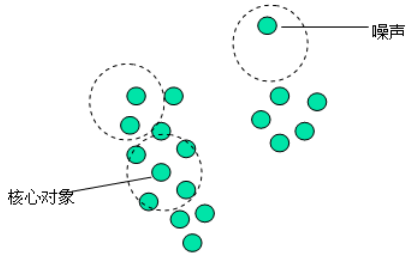
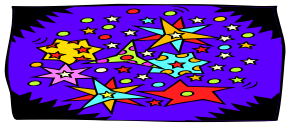


图5-9 噪声



DBSCAN算法描述

- DBSCAN通过检查数据集中每个对象的 ε-邻域来寻找聚类。如果一个点p的 ε-邻域包含多于MinPts个对象，则创建一个p作为核心对象的新簇。然后，DBSCAN反复地寻找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。当没有新的点可以被添加到任何簇时，该过程结束。具体如下：

DBSCAN算法描述
算法5-5 DBSCAN
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. REPEAT
2. 从数据库中抽取一个未处理过的点；
3. IF 抽出的点是核心点 THEN找出所有从该点密度可达的对象，形成一个簇
4. ELSE 抽出的点是边缘点(非核心对象)，跳出本次循环，寻找下一点；
5. UNTIL 所有点都被处理；

DBSCAN算法举例

下面给出一个样本事务数据库（见左表），对它实施DBSCAN算法。
根据所给的数据通过对其进行DBSCAN算法，以下为算法的步骤（设n=12，用户输入 ε=1，MinPts=4）

样本事务数据库

序号	属性 1	属性 2
1	1	0
2	4	0
3	0	1
4	1	1
5	2	1
6	3	1
7	4	1
8	5	1
9	0	2
10	1	2
11	4	2
12	1	3

DBSCAN算法执行过程示意

步骤	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	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10，12}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中，
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

聚出的类为{1，3，4，5，9，11，12}，{2，6，7，8，10}。

DBSCAN算法举例（续）

算法执行过程：

步骤	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	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10，12}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中，
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

第1步，在数据库中选择一点1，由于在以它为圆心的，以1为半径的圆内包含2个点（小于4），因此它不是核心点，选择下一个点。

第2步，在数据库中选择一点2，由于在以它为圆心的，以1为半径的圆内包含2个点，因此它不是核心点，选择下一个点。

第3步，在数据库中选择一点3，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第4步，在数据库中选择一点4，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点（直接可达4个，间接可达3个），聚出的新类{1，3，4，5，9，10，12}，选择下一个点。

第5步，在数据库中选择一点5，已经在簇1中，选择下一个点。

第6步，在数据库中选择一点6，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第7步，在数据库中选择一点7，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点，聚出的新类{2，6，7，8，11}，选择下一个点。

第8步，在数据库中选择一点8，已经在簇2中，选择下一个点。

第9步，在数据库中选择一点9，已经在簇1中，选择下一个点。

第10步，在数据库中选择一点10，已经在簇1中，选择下一个点。

第11步，在数据库中选择一点11，已经在簇2中，选择下一个点。

第12步，选择12点，已经在簇1中，由于这已经是最后一点所有点都以处理，程序终止。

- 设有12个样本点（样本点的具体坐标可以根据图中数据的位置获得，设横、纵坐标的单位间隔为1）。指定DBSCAN聚类的参数中：r=1，minPoints=4。假设有数据集如下图所示，请进行聚类。

	A	B	C	D	E	F	G	H	I	J	K	L
x	1	1	2	2	2	2	3	4	5	5	5	6
y	3	2	4	3	2	1	2	2	3	2	1	2

第五章 聚类方法

内容提要

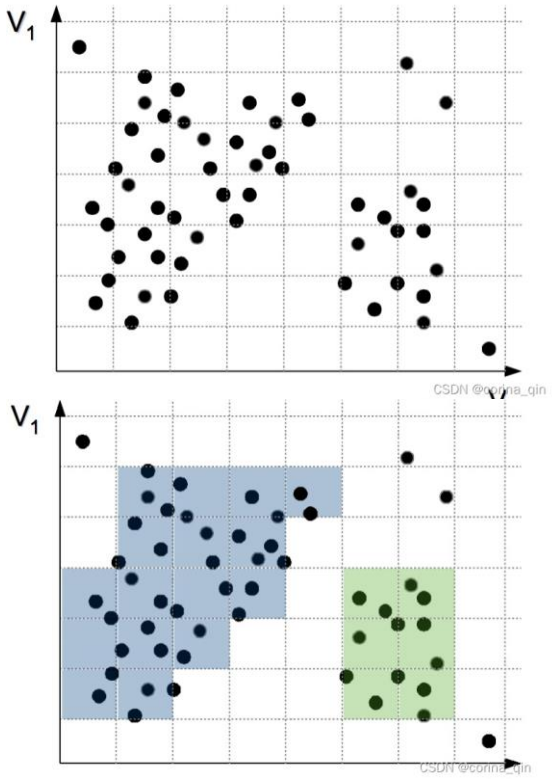
- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘

- 使用多分辨率网格数据结构
- 将空间划分为有限数据的单元，作为聚类分析的基础
- 网格方法可以有效减少算法的计算复杂度，且同样对密度参数敏感
- 主要的算法：

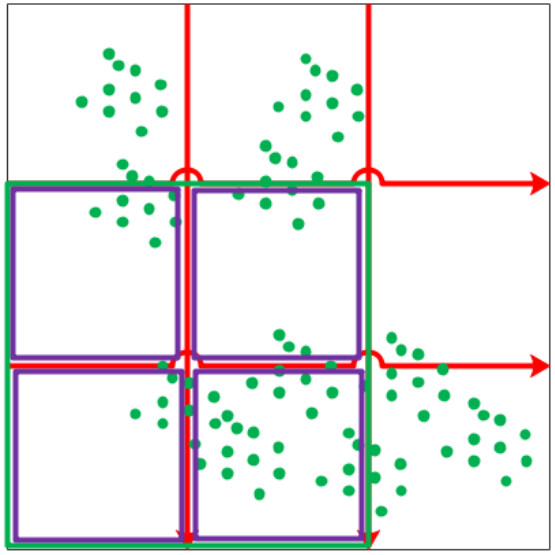
STING ：基于网格多分辨率，将空间划分为方形单元，对应不同分辨率

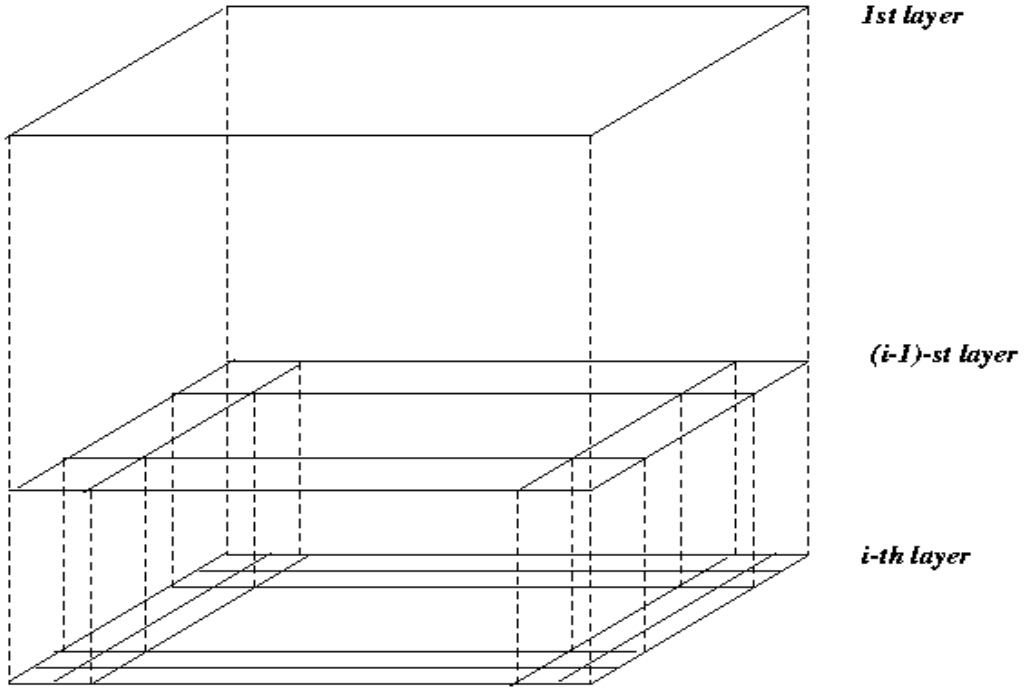
WaveCluster　用小波分析使簇的边界变得更加清晰

CLIQUE：结合网格和密度聚类的思想，子空间聚类处理大规模高维度数据



- Wang, Yang and Muntz (1997)
- 空间区域被划分为矩形单元，不同层次的矩形单元代表不同的分辨率，形成了一个层次结构





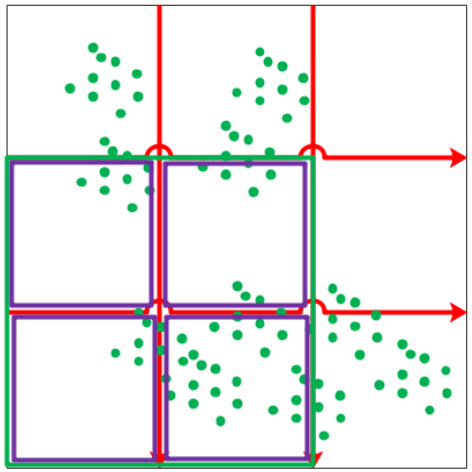
- 统计信息：每个单元都有数据统计信息，如单元所有样本的平均值，最大值，最小值，数据分布等数据；
- 预先计算：统计信息需要预先计算出来，供之后的聚类操作使用；高层单元的统计可以用低层计算得到
- 聚类分组：根据每个数据单元的统计信息，为数据单元进行聚类分组；

优势：

查询独立，易实现，增量更新

劣势：

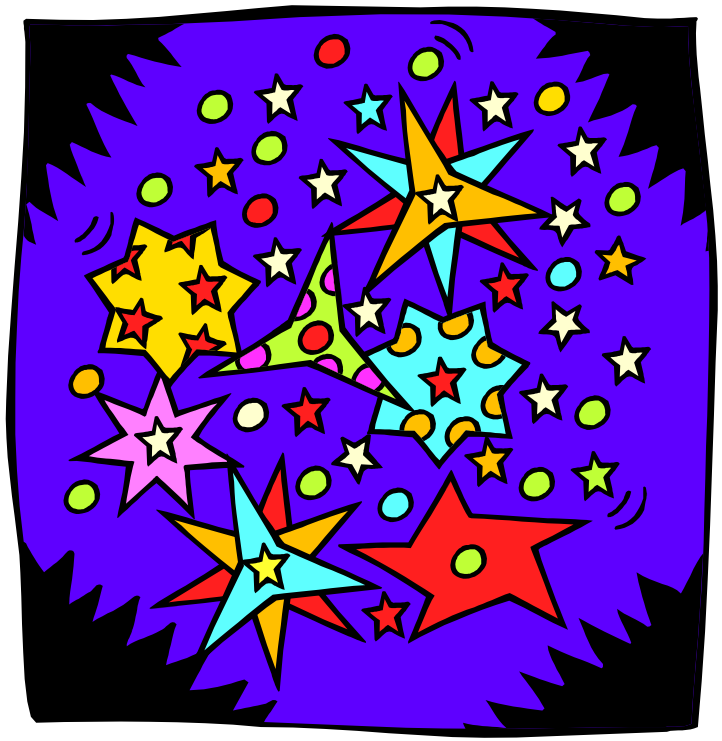
簇边界都是水平或者垂直的，没有斜的分界线

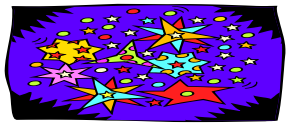


第五章 聚类方法

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
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘

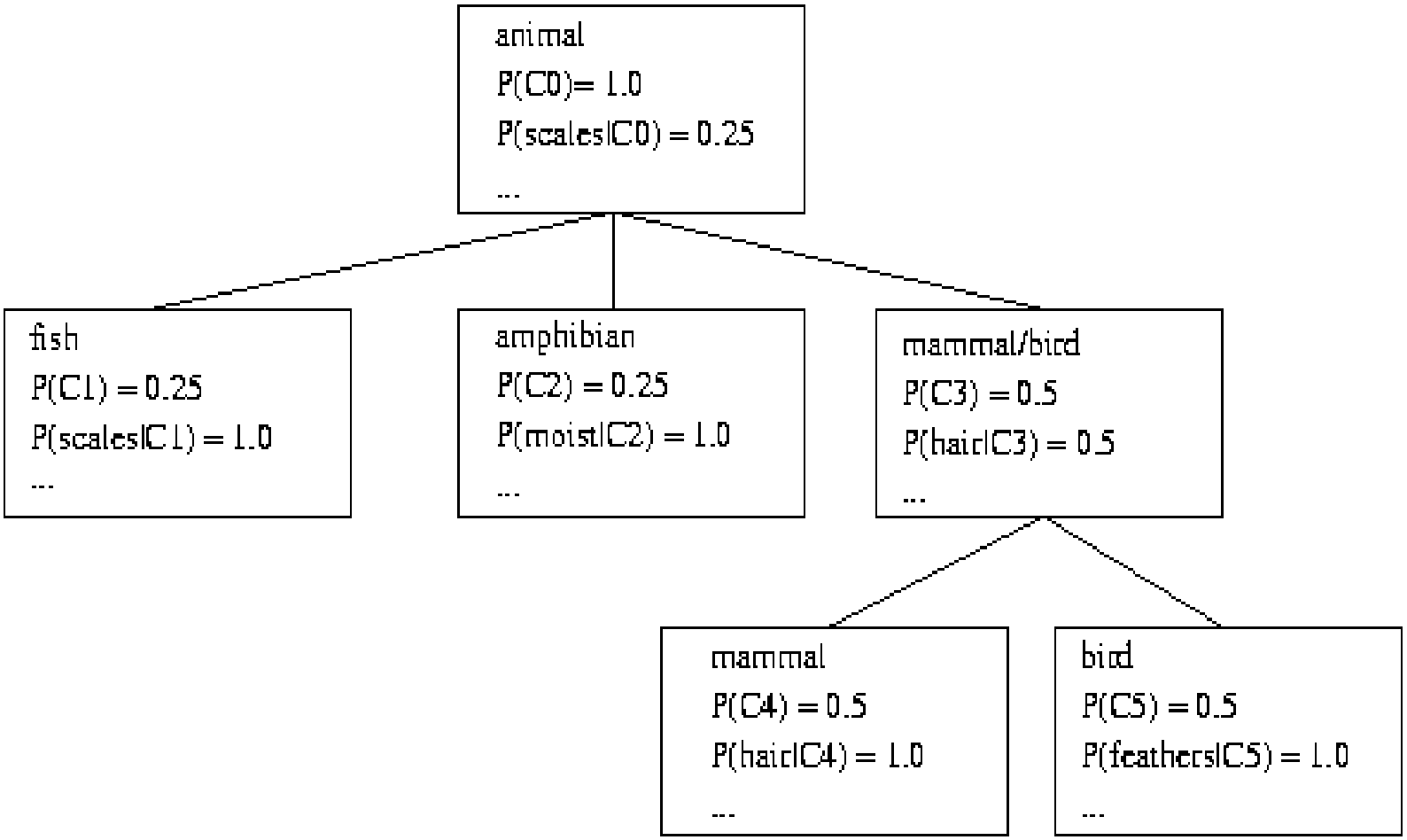


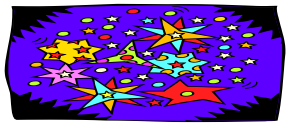


基于模型的聚类

- 把聚类问题看作是数据拟合某一种分布的优化问题
- 基于统计和人工智能的方法
  - 概念聚类
    - 机器学习中一种聚类的模式
    - 为没有标签的对象数据集构建一个分类的策略
    - 为每个概念（类）找出特征描述
  - COBWEB算法
    - 一个常用的简单的增量概念学习
    - 利用分类树的形式创建层次聚类
    - 每个节点对应一个概念，并且包含概念的概率描述

- 分类树


animal
P(C0)= 1.0
P(scales|C0) = 0.25
...
fish
P(C1) = 0.25
P(scales|C1) = 1.0
...
amphibian
P(C2) = 0.25
P(moist|C2) = 1.0
...
mammal/bird
P(C3) = 0.5
P(hair|C3) = 0.5
...
mammal
P(C4) = 0.5
P(hair|C4) = 1.0
...
bird
P(C5) = 0.5
P(feathers|C5) = 1.0
...

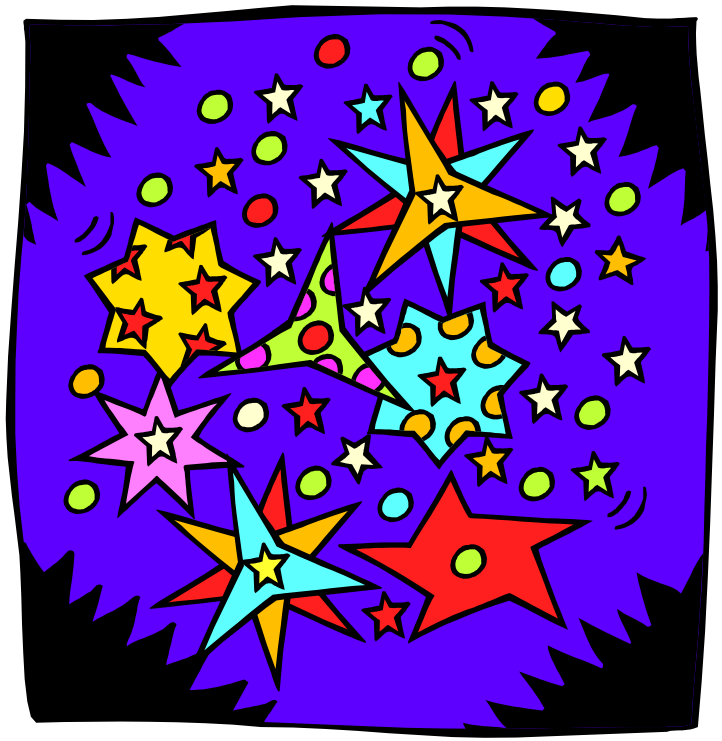


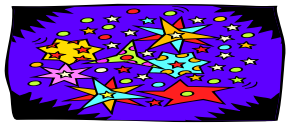
- COBWEB的一些局限性
  - 它基于这样一个假设：各个属性的概率分布是彼此统计上独立的，然后，由于属性之间经常存在相关，这个假设并不总是成立
  - 不适合大数据集聚类 – 簇的概率分布使得更新和存储簇相当昂贵
- CLASSIT
  - COBWEB的扩展，用于处理连续性数据的增量聚类
  - 与COBWEB 一样存在类似的问题
- AutoClass (Cheeseman and Stutz, 1996)
  - 使用贝叶斯统计分析来估计簇的数量
  - 在工业界比较受欢迎

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
- 网格聚类方法
- 模型聚类方法
- 离群点挖掘



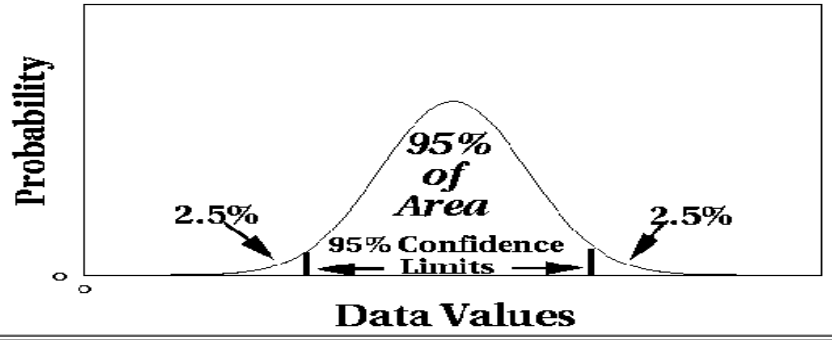


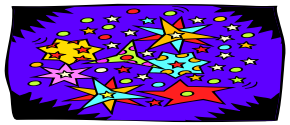
离群点挖掘

- 什么是离群点?
  - 经常存在一些数据对象，与数据的一般行为或模型不一致，这样的数据对象称为离群点
- 问题
  - 发现top n个离群点
- 应用:
  - 信用卡欺诈检测
  - 电信服务欺诈检测
  - 顾客分析
  - 医学分析

对给定的数据集假设了一个分布或概率模型 (e.g. 正态分布)

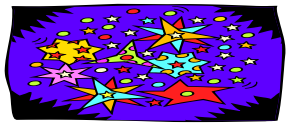
- 采用不和谐检验识别离群点
  - 第一步，判断数据分布，也可假设
  - 第二步，求得分布参数（e.g., mean, variance)
  - 第三步，检测期望的离群点数量
- 弊端
  - 大多数检验都仅仅针对单一属性
  - 在许多情况下，数据的分布是未知的





离群点挖掘-基于距离的策略

- 为了解决统计学方法带来的限制，引入了基于距离的离群点的概念
  - 我们需要在未知数据分布的情况下进行多维分析.
- 基于距离的离群点：如果数据集合D中对象至少有pct部分与对象o的距离大于dmin，则称对象o是以pct和dmin为参数的基于距离的离群点
- 基于距离的离群点检测算法
  - 基于索引的分析算法
  - 嵌套-循环的分析算法
  - 基于单元格的分析算法



离群点挖掘-基于偏差的策略

- 通过检查一组对象的主要特征来识别离群点
- “背离”这种描述的对象被认为是离群点
- 顺序异常技术
  - 模仿人类从一系列推测类似的对象中识别异常对象的方式
- OLAP数据立方体技术
  - 使用数据立方体识别大型多维数据中的异常区域

Thank you !!!