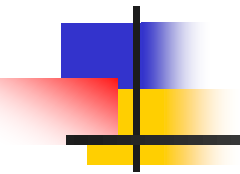


Chapter 9.聚类分析

第9章 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）
- 小结

什么是聚类分析?

- 聚类: 数据对象的集合/簇 (cluster)
  - 同一簇中的对象彼此相似
  - 不同簇中的对象彼此相异
- 聚类分析
  - 将数据对象分组成为多个类或簇
- 聚类是无指导的分类: 没有预先定义的类
- 典型应用
  - 作为洞察数据内部分布的工具
    - 模式识别、万维网(web文档/日志) 、市场营销、保险业
  - 作为其它算法(如特征化、分类)的预处理步骤

划分方法

聚类的一般应用

- 模式识别
- 空间数据分析
  - 聚类产生GIS(地理信息系统)的专题地图thematic maps
  - 在空间数据挖掘中检测空间聚类并解释它们
- 图象处理
- 经济科学 (特别是市场研究)
- WWW
  - 文本分类
  - Web日志数据聚类，发现类似访问模式群

聚类应用的例子

- 市场营销:

  帮助市场营销者发现他们的基本顾客的不同组群，然后利用这一知识制定有针对性的营销计划

- 国土利用

  在地球观测数据库中识别类似的国土使用区域

- 保险

  对汽车保险持有者的分组

- 城市规划

  根据房子的类型，价值，和地理位置对一个城市中房屋的分组

什么是好的聚类方法?

- 一个好的聚类方法应当产生高质量的聚类
  - 类内相似性高
  - 类间相似性低
- 聚类结果的质量依赖于方法所使用的相似性度量和它的实现.
- 聚类方法的质量也用它发现某些或全部隐藏的模式的能力来度量

数据挖掘对聚类的要求

- 可伸缩性
  - 有的算法当数据对象少于200时处理很好, 但对大量数据对象偏差较大
  - 大型数据库包含数百万个对象
- 处理不同属性类型的能力
  - 许多算法专门用于数值类型的数据
  - 实际应用涉及不同的数据类型,i.e. 混合了数值和分类数据
- 发现任意形状的聚类

数据挖掘对聚类的要求(续)

- 用于决定输入参数的领域知识最小化
  - 许多聚类算法要求用户输入一定的参数, 如希望产生的簇的数目.聚类结果对于输入参数十分敏感
  - 参数难以确定, 增加了用户的负担, 使聚类质量难以控制
- 处理噪声数据和孤立点的能力
  - 一些聚类算法对于噪音数据敏感, 可能导致低质量的聚类结果
  - 现实世界中的数据库大都包含了孤立点, 空缺, 或者错误的数据

数据挖掘对聚类的要求(续)

- 高维性（high dimensionality）
  - 许多聚类算法擅长处理低维的数据, 可能只涉及两到三维
  - 数据库或者数据仓库可能包含若干维或者属性, 数据可能非常稀疏, 而且高度偏斜
- 整合用户指定的约束
  - 现实世界的应用可能需要在各种约束条件下进行聚类
  - 要找到既满足特定的约束, 又具有良好聚类特性的数据分组是一项具有挑战性的任务
- 可解释性和可用性
  - 用户希望聚类结果是可解释的, 可理解的, 和可用的

第7章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）

数据结构

- 数据矩阵
  - (two modes)

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

- 相异度矩阵

  (Dissimilarity matrix)
  - (one mode)

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

评估聚类的质量

- 有一个单独的“质量”函数, 它度量聚类的“好坏”.
- 很难定义“足够类似”或“足够好”
  - 对此问题是相当主观的.
- 相异度/相似度矩阵
  - 相似性用距离函数表示, 通常记作 $d(i, j)$
- 对于区间标度变量, 二元变量, 标称变量, 序数和比例标度变量, 距离函数的定义通常是很不相同的.

聚类分析的数据类型

- 区间标度变量(Interval-scaled variables)
- 二元变量(Binary variables)
- 标称(名词性), 序数, 和比例标度变量(Nominal, ordinal, and ratio variables)
- 混合类型变量(Variables of mixed types)

区间标度变量

- 区间标度变量：一种粗略线形标度的连续度量
- 为了避免度量单位的影响，数据标准化
  - (1)计算平均绝对偏差：$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$

    $$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$$

    其中
  - (2)计算标准化的度量值 (*z-score*)

    $$z_{if} = \frac{x_{if} - m_f}{s_f}$$

对象之间的相似性/相异性

- 通常, 使用距离来度量两个数据对象之间的相似性/相异性
- 常用的距离包括:闵可夫斯基(*Minkowski*) 距离:

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

其中 $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$和 $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ 是两个 p-维数据对象(q 正整数)

- 如果$q = 1$, d是曼哈坦 (Manhattan)距离

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+\ldots+|x_{i_p}-x_{j_p}|$$

对象之间的相似性/相异性

- 如果 ***q* = 2, *d*是欧几里德(Euclidean)距离 :**

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

- 距离的性质
  - 非负性: ***d(i,j)*** ≥ **0**
  - 自身到自身的距离为**0: *d(i,i)* = 0**
  - 对称性: ***d(i,j)* = *d(j,i)***
  - 三角不等式: ***d(i,j)*** ≤ ***d(i,k)* + *d(k,j)***
- 也可以使用加权的距离, 如加权的欧几里德距离

$$d(i,j)=\sqrt{w_1(|x_{i1}-x_{j1}|^2+w_2|x_{i2}-x_{j2}|^2+\ldots+w_p|x_{ip}-x_{jp}|^2)}$$

二元变量

- 二元变量(binary variable)只有两个状态0或1. 0表示该变量为空, 1表示该变量存在
  - 例如, 描述病人的变量*smoker*, 1表示病人抽烟, 而0表示病人不抽烟
- 计算二元变量的相似度
  - 假定所有二元变量具有相同的权重, 则得到一个两行两列的可能性表(contingency table)

		对象 j		
		1	0	sum
对象 i	1	q	r	$q+r$
	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

二元变量

- 对称的: 二元变量的两个状态具有同等价值,并具有相同的权重
  - 例: 性别是对称的二元变量
- 恒定的相似度: 基于对称的二元变量的相似度, 当一些或全部二元变量编码改变时, 计算结果不会发生变化
- 对称的二元变量的相异度计算 $d(i,j) = \frac{r+s}{q+r+s+t}$ 简单匹配系数

- 不对称的: 二元变量的两个状态的输出不是同样重要
  - 例: 疾病检查结果的*肯定*和*否定*.

二元变量(续)

- 根据惯例, 比较重要的输出结果是出现几率较小的
  - 例: HIV阳性是比较重要的结果,出现几率较小, 而HIV阴性(正常情况)出现几率较大
- 通常, 比较重要的输出结果编码为1, 另一种结果编码为0
- 两个都取1的情况(正匹配)比两个都取0的情况(负匹配)更有意义. ----非恒定的相似度
- 对于非对称的相似度, 负匹配数目t被忽略.
  - 采用Jaccard系数

$$d(i,j)=\frac{r+s}{q+r+s}=1-\frac{q}{q+r+s}=1-Jaccard(i,j)$$

二元变量之间的相异度

- 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

- **gender 是对称的**
- **其余都不是对称的**
- **Y和P的值设置为1, 而 N的值设置为 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-分类变量，名义变量

- 标称变量(Nominal variable)是二元变量的拓广, 它可以取多于两种状态值, 如, red, yellow, blue, green
- 方法1: 简单匹配
  - *m*:状态取值匹配的变量数目, *p*: 变量总数

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

- 方法 2：（可以用非对称的二元变量对标称变量编码）使用大量二元变量，
  - 对M个标称状态的每一个, 创建一个新的二元变量. 对于

序数型变量

- 序数型变量(ordinal variable)可以是离散的, 也可以是连续的
  - 离散的序数型变量类似于标称变量, 但序数型变量的M个状态是以有意义的序列排序
  - 连续的序数型变量看起来像一个未知刻度的连续数据的集合.
    - 值的相对顺序是必要的, 而其实际的大小则不重要
  - 将区间标度变量的值域划分为有限个区间, 从而将其值离散化, 也可以得到序数型变量
  - 序数型变量的值可以映射为秩(rank).

序数型变量(续)

- 相异度计算可以用类似于区间标度变量的方法处理
  - 设第i 个对象f 的值为 x_{if}, 用对应的秩r_{if} 替代x_{if}, 其中 $r_{if} \in \{1,\ldots,M_f\}$
  - 将每个变量的值域映射到[0, 1]区间, 以便每个变量都具有相同的权重: 用下式替换r_{if}

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

  - 使用区间标度变量计算距离的方法计算相异度, z_{if}作为第i 个对象f 的值

比例标度变量

- 比例标度变量(Ratio-scaled variable)非线性的刻度上取正的度量值，例如指数标度，近似地遵循如下的公式

  Ae^{Bt} 或Ae^{-Bt}

- 相异度计算:
  - 采用与处理区间标度变量同样的方法 —*不是好的选择!*
    (为什么?—标度可能被扭曲)
  - 进行对数变换

    $y_{if} = log(x_{if})$
  - 将x_{if}看作连续的序数型数据, 将其秩作为区间标度值

混合类型变量

- 数据库可能包含所有六种类型
  - 对称的二元变量, 不对称的二元变量, 标称的, 序数的, 区间的, 比例标度的

- 如何计算混合类型变量描述的对象的相异度?
  - 方法1: 将变量按类型分组, 对每种类型的变量单独进行聚类分析
    - 如果这些分析得到兼容的结果, 这种方法是可行的
    - 在实际的应用中, 这种方法行不通

混合类型变量(续)

- 假设数据集包含p个不同类型的变量，对象i 和j 之间的相异度$d(i,j)$定义为

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

其中, 如果x_{if}或x_{jf}缺失(即对象i 或对象j 没有变量f 的度量值)或者$x_{if}=x_{jf}=0$, 且变量f 是不对称的二元变量, 则指示项 $\delta_{ij}^{(f)}=0$; 否则, 指示项 $\delta_{ij}^{(f)}=1$

- 变量f 对i 和j 之间相异度的计算方式与其具体类型有关
  - 如果f 是二元或标称变量:

    如果 $x_{if}=x_{jf}$, $d_{ij}^{(f)}=0$; 否则 $d_{ij}^{(f)}=1$

混合类型变量(续)

- 如果f 是区间标度变量, 使用规格化的距离

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$$

- 如果f 是序数型或比例标度型变量
  - 计算秩r_{if} 和 $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
  - 将 z_{if} 作为区间标度变量对待

向量对象

- 向量x和y
  - 余弦度量

$$\cos(x,y)=\frac{x^t\cdot y}{\|x\|\cdot\|y\|}=\frac{\sum_{i=1}^{p}x_iy_i}{\sqrt{\sum_{i=1}^{p}x_i^{\,2}\cdot\sum_{i=1}^{p}y_i^{\,2}}}$$

  - Tanimoto系数

$$s(x,y)=\frac{x^t\cdot y}{x^t\cdot x+y^t\cdot y-x^t\cdot y}$$

第8章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）
- 小结

主要聚类方法的分类

1. 划分方法(Partitioning method): 构造*n*个对象数据库*D*的划分, 将其划分成*k*个聚类满足如下的要求：
   - 每个组至少包含一个对象
   - 每个对象必须属于且只属于一个组
   - 在某些模糊划分技术中, 第二个要求可以放宽
- 基本方法: 首先创建一个初始划分. 然后采用一种迭代的重定位技术, 尝试通过在划分间移动对象来改进划分
- 好的划分的一般准则: 在同一个类中的对象之间尽可能“接近”或相关, 而不同类中的对象之间尽可能“远离”或不同

主要聚类方法的分类(续)

- 全局最优: 穷举所有可能的划分
- 启发式方法: *k*-平均值(*k- means*)和 *k*-中心点(*k- medoids*)算法
  - *k*-平均值(MacQueen'67): 每个簇用该簇中对象的平均值来表示
  - *k*-中心点或 PAM (Partition around medoids) (Kaufman & Rousseeuw'87): 每个簇用接近聚类中心的一个对象来表示
- 这些启发式算法适合发现中小规模数据库中的球状聚类

主要聚类方法的分类(续)

2. 层次方法(Hierarchy method): 对给定数据对象集合进行层次的分解

- 两种层次方法
  - 凝聚方法, 也称为自底向上的方法: 开始将每个对象作为单独的一个组; 然后继续地合并相近的对象或组, 直到所有的组合并为一个(层次的最上层), 或者达到一个终止条件
  - 分裂方法, 也称为自顶向下的方法: 开始将所有的对象置于一个簇; 在迭代的每一步, 一个簇被分裂为更小的簇, 直到最终每个对象在单独的一个簇中, 或者达到一

主要聚类方法的分类(续)

- 层次方法的缺点: 一个步骤一旦完成便不能被撤消.
  - 该规定可以避免考虑选择不同的组合, 减少计算代价
  - 问题: 不能更正错误的决定
- 改进层次聚类结果的措施
  - 在每层划分中, 仔细分析对象间的“联接”, 例如**CURE**和**Chameleon**中的做法
  - 综合层次凝聚和迭代的重定位方法。首先用自底向上的层次算法，然后用迭代的重定位来改进结果。例如在**BIRCH**中的方法

主要聚类方法的分类(续)

3. 基于密度的方法(Density-based method): 基于密度函数

- 基本思想: 只要临近区域的密度(对象或数据点的数目)超过某个阀值, 就继续聚类. 也就是说, 对给定类中的每个数据点, 在一个给定范围的区域中必须至少包含一定数目的点
- 该方法可以用来过滤“噪音”数据，发现任意形状的簇
- DBSCAN是一个有代表性的基于密度的方法, 它根据一个密度阀值来控制簇的增长
- OPTICS是另一个基于密度的方法, 它为自动的, 交互的聚类分析计算一个聚类顺序

主要聚类方法的分类(续)

4. 基于网格的方法(Grid-based method): 把对象空间量化为有限数目的单元, 形成了一个网格结构. 所有的聚类操作都在这个网格结构(即量化的空间)上进行

- 这种方法的主要优点是它的处理速度很快, 其处理时间独立于数据对象的数目，只与量化空间中每一维的单元数目有关
- STING是基于网格方法的一个典型例子
- CLIQUE和WaveCluster这两种算法既是基于网格的, 又是基于密度的

主要聚类方法的分类(续)

5. 基于模型的方法(Model-based Method): 基于模型的方法为每个簇假定了一个模型, 寻找数据对给定模型的最佳拟合

第9章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）
- 小结

划分方法

- 划分方法: 构造*n*个对象数据库*D*的划分, 将其划分成*k*个聚类
- 给定 *k*, 找*k* 个*clusters* 对于选定的划分标准它是最优的
  - 全局最优(Global optimal): 枚举所有的划分
  - 启发式方法(Heuristic methods): *k*-平均(*k-means*)和*k*-中心点(*k-medoids*)算法
    - *k*-平均(MacQueen'67): 每个簇用簇的重心(簇的平均值) 表示
    - *k*-中心点或PAM (Partition around medoids) (Kaufman & Rousseeuw'87): 每个簇用接近聚类中心的一个对象来表示

k-平均聚类算法

- 算法： *k*-平均

  (1) 任意选择k个对象作为初始的簇中心；

  (2) repeat

  (3)　根据簇中对象的平均值, 将每个对象(重新)赋给最类似的簇；

  (4)　更新簇的平均值, 即重新计算每个簇中对象的平均值；

  (5) until 不再发生变化

- 通常, 采用平方误差准则作为收敛函数, 其定义如下

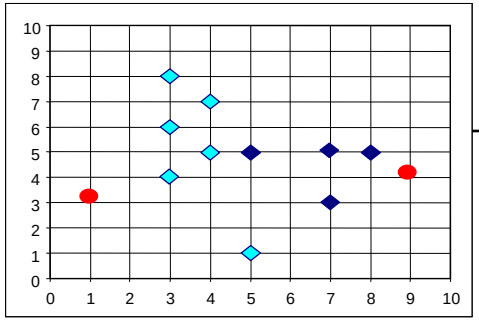
$$E = \sum_{i=1}^{k} \sum_{p \in C_i} | p - m_i |^2$$

  其中, m_i是簇C_i的平均值

  该准则试图使生成的结果簇尽可能紧凑, 独立

k-平均聚类算法(续)

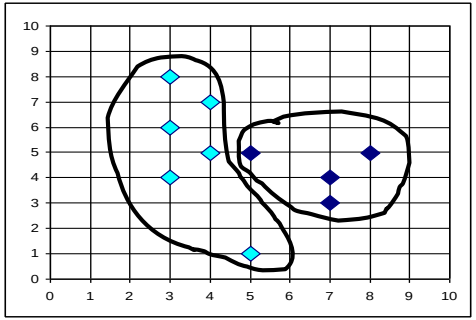
- 例



K=2

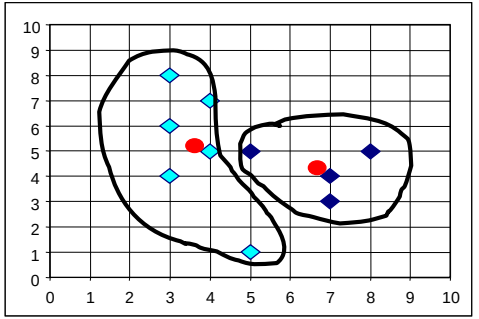
任意选择 K个对象作为初始聚类中心

将每个对象赋给最类似的中心

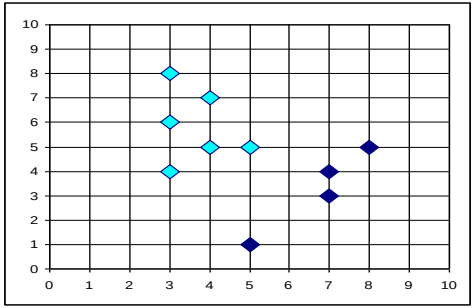


重新赋值

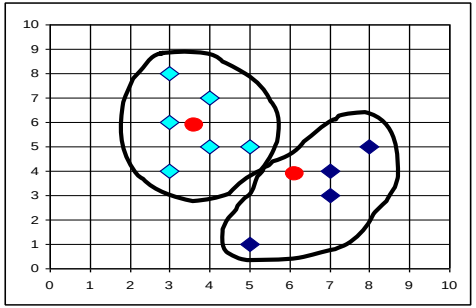
更新簇的平均值



重新赋值



更新簇的平均值



k-平均聚类算法(续)

- 优点: 相对有效性: ***O(tkn)***,

  其中 ***n*** 是对象数目, ***k*** 是簇数目, ***t*** 是迭代次数; 通常,

  k, t << n.

  - 比较: **PAM: $O(k(n-k)^2)$**
- 当结果簇是密集的，而簇与簇之间区别明显时，它的效果较好
- **Comment:** 常常终止于*局部最优*.
- *全局最优* 可以使用诸如确定性的退火(***deterministic annealing)***和遗传算法(***genetic algorithms***)等技术得到

k-平均聚类算法(续)

- 弱点
  - 只有在簇的平均值(*mean*)被定义的情况下才能使用.可能不适用于某些应用, 例如涉及有分类属性的数据
  - 需要预先指顶簇的数目*k*
  - 不能处理噪音数据和孤立点(*outliers*)
  - 不适合用来发现具有非凸形状(*non-convex shapes*)的簇

k-平均方法的变种

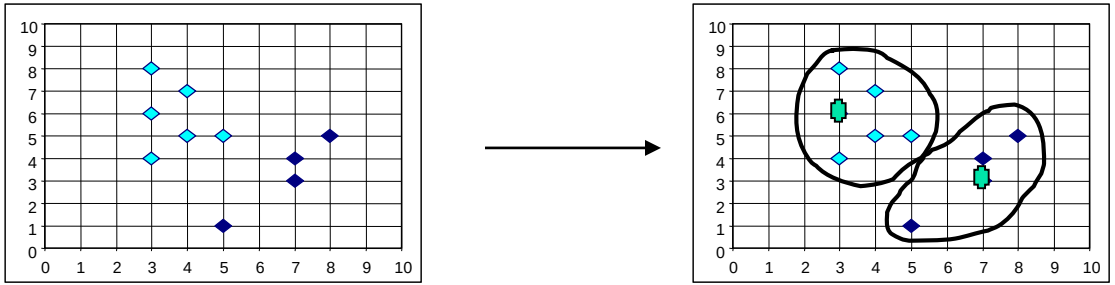
- k-平均方法的变种, 它们在以下方面有所不同
  - 初始k个平均值的选择
  - 相异度的计算
  - 计算聚类平均值的策略
- 较好的策略: 先用层次凝聚算法决定簇的数目, 并产生初始聚类, 然后用迭代重定位改进聚类结果
- 处理分类属性: *k*- 模(*k-modes*) 方法(Huang'98)
  - 用模(modes众数)替代聚类的平均值
  - 使用新的相异性度量方法来处理分类对象
  - 用基于频率的方法来修改聚类的模
- *k*-原型(*k-prototype*)方法: *k*-平均和*k*-模的结合, 处理具有数值和分类属性的数据

k-平均方法的变种(续)

- **EM(Expectation Maximization, 期望最大)算法**
  - 以另一种方式对*k*-means方法进行了扩展: 不把对象分配给一个确定的簇, 而是根据对象与簇之间隶属关系发生的概率来分派对象
- 怎样增强*k-means*算法的可扩展性？
  - 数据分成三种区域:
  1. 可废弃的: 一个对象与某个簇的隶属关系是确定的
  2. 可压缩的: 一个对象不是可废弃的,但属于某个紧密的子簇
  3. 必须在主存: 既不是可废弃的, 又不是可压缩的
  - 迭代的算法只包含可压缩的对象和必须在主存中的对

k-中心点聚类方法

- ***k*-平均值算法对孤立点很敏感!**
  - 因为具有特别大的值的对象可能显著地影响数据的分布.
- ***k*-中心点(*k*-Medoids): 不采用簇中对象的平均值作为参照点, 而是选用簇中位置最中心的对象, 即中心点(medoid)作为参照点.**



k-中心点聚类方法(续)

- 找聚类中的代表对象(中心点)
- ***PAM* (Partitioning Around Medoids, 1987)**
  - 首先为每个簇随意选择选择一个代表对象, 剩余的对象根据其与代表对象的距离分配给最近的一个簇; 然后反复地用非代表对象来替代代表对象，以改进聚类的质量
  - *PAM* 对于较小的数据集非常有效, 但不能很好地扩展到大型数据集
- ***CLARA* (Kaufmann & Rousseeuw, 1990)抽样**
- ***CLARANS* (Ng & Han, 1994): 随机选样**

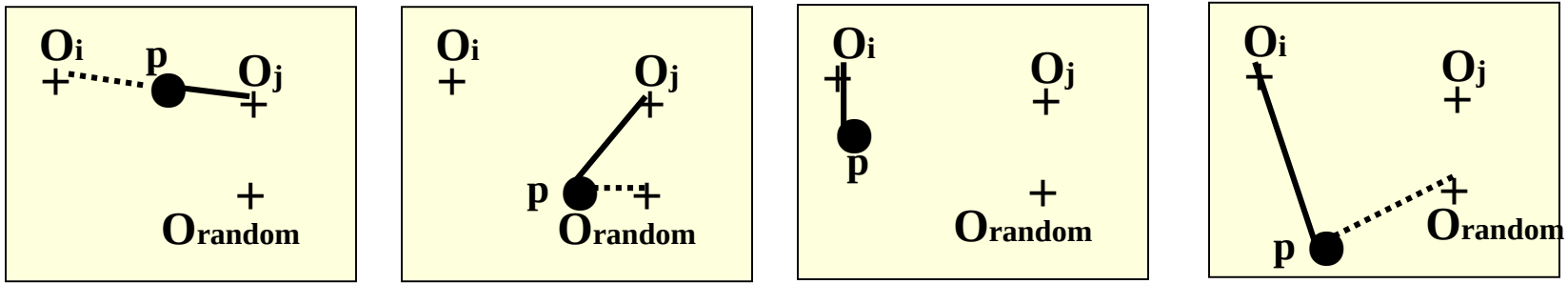
k-中心点聚类方法(续)

- 基本思想：
  - 首先为每个簇随意选择一个代表对象；剩余的对象根据其与代表对象的距离分配给最近的一个簇
  - 然后反复地用非代表对象来替代代表对象，以改进聚类的质量
  - 聚类结果的质量用一个代价函数来估算，该函数评估了对象与其参照对象之间的平均相异度

k-中心点聚类方法(续)

- 为了判定一个非代表对象O_{random}是否是当前一个代表对象O_j的好的替代, 对于每一个非代表对象p,考虑下面的四种情况：
  - 第一种情况：p当前隶属于代表对象 O_j. 如果O_j被O_{random}所代替, 且p离O_i最近, $i \neq j$, 那么p被重新分配给O_i
  - 第二种情况：p当前隶属于代表对象 O_j. 如果O_j 被O_{random}代替, 且p离O_{random}最近, 那么p被重新分配给O_{random}
  - 第三种情况：p当前隶属于O_i，$i \neq j$。如果O_j被O_{random}代替，而p仍然离O_i最近，那么对象的隶属不发生变化
  - 第四种情况：p当前隶属于O_i，$i \neq j$。如果O_j被O_{random}代替，且p离O_{random}最近，那么p被重新分配给O_{random}

k-中心点聚类方法(续)



重新分配给O_i　　2. 重新分配给O_{random}　　3. 不发生变化　　4.重新分配给O_{random}

● 数据对象　　+ 簇中心　　—— 替代前　　..... 替代后

图8-3　k-中心点聚类代价函数的四种情况

k-中心点聚类方法(续)

- 算法: k-中心点

(1) 随机选择k个对象作为初始的代表对象；

(2) repeat

(3) 指派每个剩余的对象给离它最近的代表对象所代表的簇；

(4) 随意地选择一个非代表对象O_{random}；

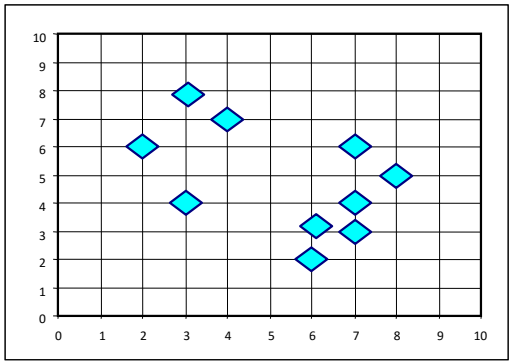
(5) 计算用O_{random}代替O_j的总代价S；

(6) 如果S<0，则用O_{random}替换O_j，形成新的k个代表对象的集合；

PAM

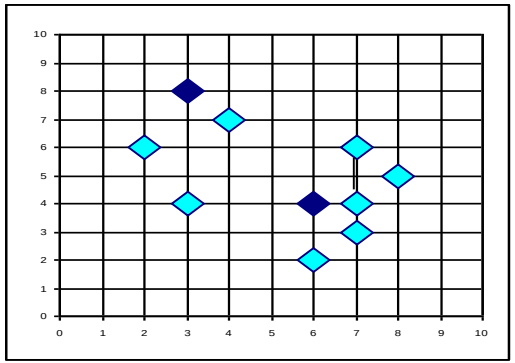
- **PAM (Partitioning Around Medoids) (Kaufman and Rousseeuw, 1987)**
  - 是最早提出的*k*-中心点聚类算法
  - 基本思想:
    - 随机选择*k*个代表对象
    - 反复地试图找出更好的代表对象：分析所有可能的对象对，每个对中的一个对象被看作是代表对象, 而另一个不是. 对可能的各种组合, 估算聚类结果的质量

PAM(续)



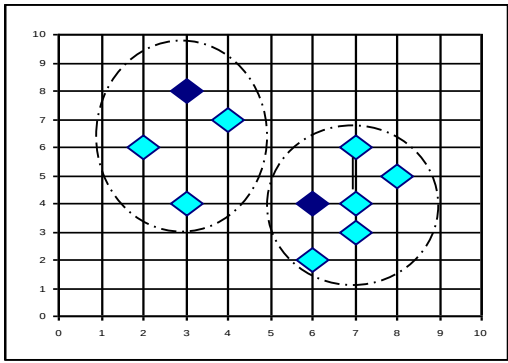
K=2

Arbitrary choose k object as initial medoids

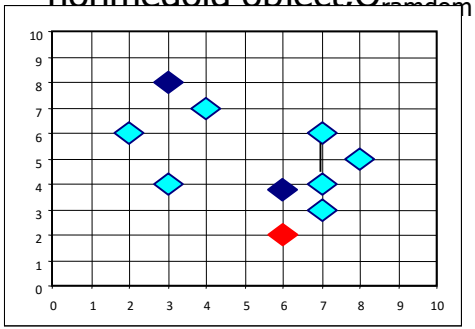


Assign each remaining object to nearest medoids

Total Cost = 20

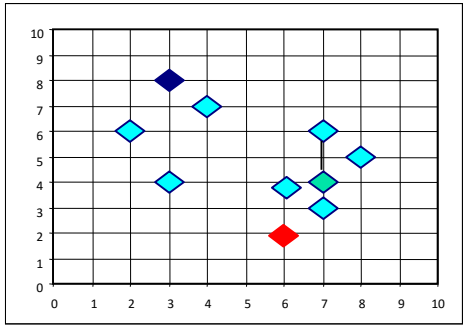


Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

PAM(续)

- 当存在噪音和孤立点时, PAM 比 *k*-平均方法更健壮. 这是因为中心点不象平均值那么容易被极端数据影响
- PAM对于小数据集工作得很好, 但不能很好地用于大数据集
  - 每次迭代$O(k(n-k)^2)$

其中 *n* 是数据对象数目, *k* 是聚类数

➔基于抽样的方法,

CLARA(Clustering LARge Applications)

CLARA (Clustering Large Applications) (1990)

- ***CLARA* (Kaufmann and Rousseeuw in 1990)**
  - 不考虑整个数据集, 而是选择数据的一小部分作为样本
- 它从数据集中抽取多个样本集, 对每个样本集使用***PAM***, 并以最好的聚类作为输出
- 优点: 可以处理的数据集比 ***PAM***大
- 缺点:
  - 有效性依赖于样本集的大小
  - 基于样本的好的聚类并不一定是 整个数据集的好的聚类, 样本可能发生倾斜

CLARANS ("Randomized" CLARA) *(1994)*

- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
- CLARANS将采样技术和PAM结合起来
  - CLARA在搜索的每个阶段有一个固定的样本
  - CLARANS任何时候都不局限于固定样本, 而是在搜索的每一步带一定随机性地抽取一个样本
- 聚类过程可以被描述为对一个图的搜索, 图中的每个节点是一个潜在的解, 也就是说 *k* medoids
  - 相邻节点：代表的集合只有一个对象不同

CLARANS(续)

- 如果一个更好的邻居被发现, CLARANS移到该邻居节点, 处理过程重新开始, 否则当前的聚类达到了一个局部最优
- 如果找到了一个局部最优, CLARANS从随机选择的节点开始寻找新的局部最优
- 实验显示CLARANS比PAM和CLARA更有效
- CLARANS能够探测孤立点
- 聚焦技术和空间存取结构可以进一步改进它的性能 (Ester et al.'95)

第7章. 聚类分析

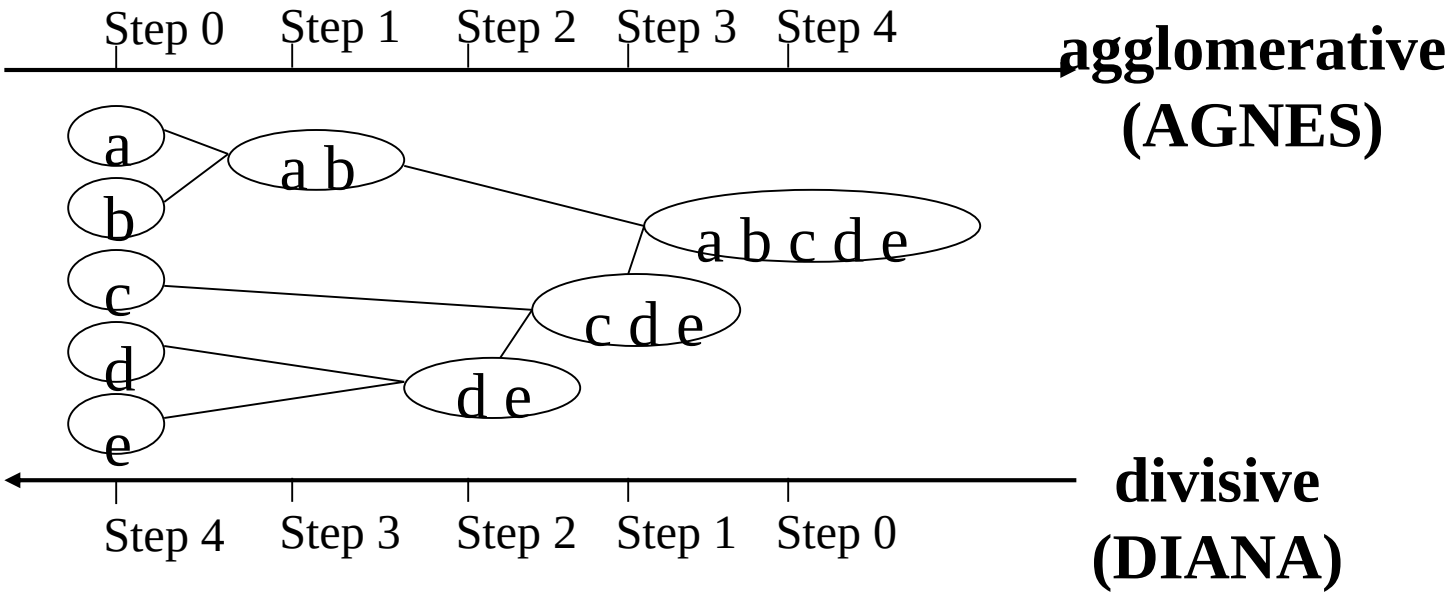
- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）

层次方法

- 层次的聚类方法将数据对象组成一棵聚类的树
- 根据层次分解是自底向上, 还是自顶向下形成, 层次的聚类方法可以进一步分为凝聚的(agglomerative)和分裂的(divisive)层次聚类
- 纯粹的层次聚类方法的聚类质量受限于如下特点：一旦一个合并或分裂被执行，就不能修正
- 最近的研究集中于凝聚层次聚类和迭代重定位方法的集成
- 使用距离矩阵作为聚类标准. 该方法不需要输入聚类数目 ***k***, 但需要终止条件

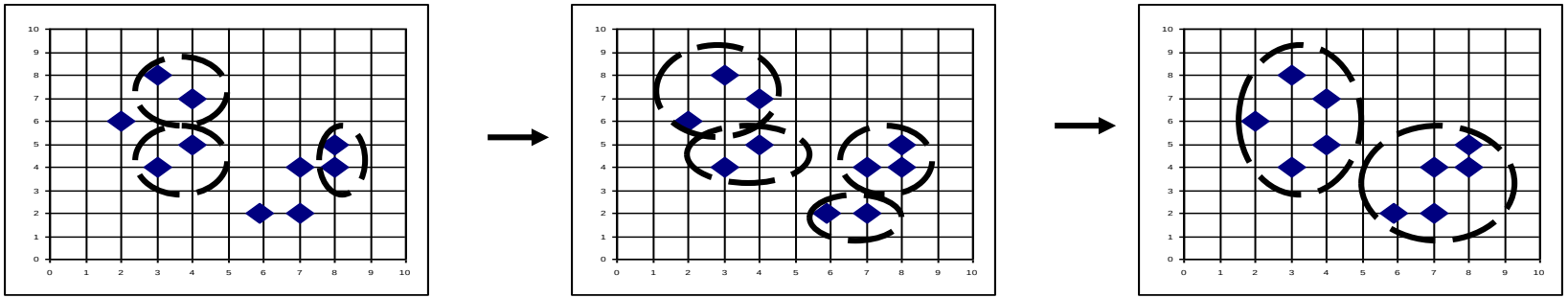
层次方法(续)

- 凝聚的(agglomerative)和分裂的(divisive)层次聚类图示



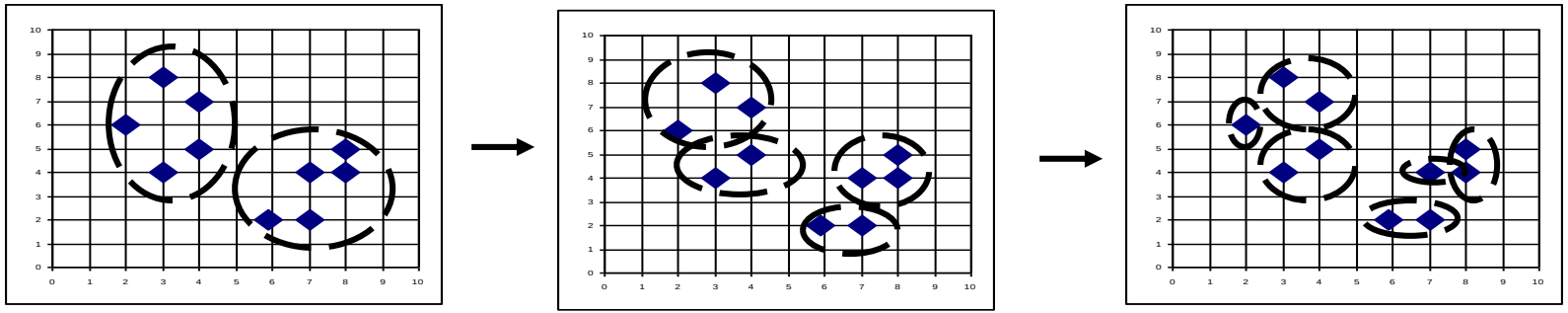
AGNES (Agglomerative Nesting)

- 由 Kaufmann和Rousseeuw提出(1990)
- 已在一些统计分析软件包中实现 . 如 Splus
- 使用单链接(Single-Link)方法和相异度矩阵
- 合并具有最小相异度的节点
- 以非递减的方式继续
- 最终所有的节点属于同一个簇



DIANA (Divisive Analysis)

- 由 Kaufmann和Rousseeuw提出 (1990)
- 已在一些统计分析软件包中实现 . 如 Splus
- 是 AGNES的逆
- 最终每个节点自己形成一个簇



层次方法(续)

- 四个广泛采用的簇间距离度量方法
  - 最小距离：$d_{min}(C_i,C_j) = min_{\ p\in Ci,\ p'\in Cj} |p-p'|$
  - 最大距离：$d_{max}(C_i,C_j) = max_{\ p\in Ci,\ p'\in Cj} |p-p'|$
  - 平均值的距离：$d_{mean}(C_i,C_j) = | m_i - m_j |$
  - 平均距离：$d_{avg}(C_i,C_j) = \sum_{p\in Ci} \sum_{p'\in Cj} |p-p'| / ninj$

其中，$|p-p'|$是两个对象p和p'之间的距离

m_i是簇C_i 的平均值，n_i是簇Ci中对象的数目

层次方法(续)

- 层次聚类的主要缺点
  - 不具有很好的可伸缩性: 时间复杂性至少是 $O(n^2)$, 其中 n 对象总数
  - 合并或分裂的决定需要检查和估算大量的对象或簇
  - 不能撤消已做的处理, 聚类之间不能交换对象. 如果某一步没有很好地选择合并或分裂的决定, 可能会导致低质量的聚类结果

层次方法(续)

- 改进层次方法的聚类质量的方法：将层次聚类和其他的聚类技术进行集成，形成多阶段聚类
  - **BIRCH (1996)**：使用 CF-tree对对象进行层次划分，然后采用其他的聚类算法对聚类结果进行求精
  - **ROCK1999**：基于簇间的互联性进行合并
  - **CHAMELEON (1999)**：使用动态模型进行层次聚类
  - **CURE (1998)**：采用固定数目的代表对象来表示每个簇，然后依据一个指定的收缩因子向着聚类中心对它们进行收缩

BIRCH (1996)

- Birch (Balanced Iterative Reducing and Clustering using Hierarchies): 利用层次方法的平衡迭代归约和聚类由 Zhang, Ramakrishnan和Livny 提出(SIGMOD'96)
- 两个重要概念
  - 聚类特征(Clustering Feature, CF)
  - 聚类特征树(Clustering Feature Tree, CF树)
- 聚类特征
  - 聚类特征(CF)是一个三元组，给出对象子类的信息的汇总描述
  - 设某个子类中有N个d维的点或对象$\{o_i\}$，则该子类的CF定义如下

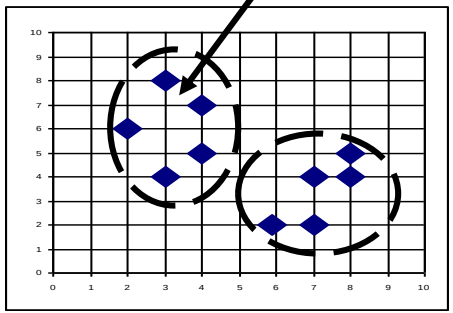
聚类特征

Clustering Feature: $CF = (N, \overrightarrow{LS}, SS)$

N: 数据点数目

LS: $\sum_{i=1}^{N} \overrightarrow{X_i}$

SS: $\sum_{i=1}^{N} \overrightarrow{X_i^2}$



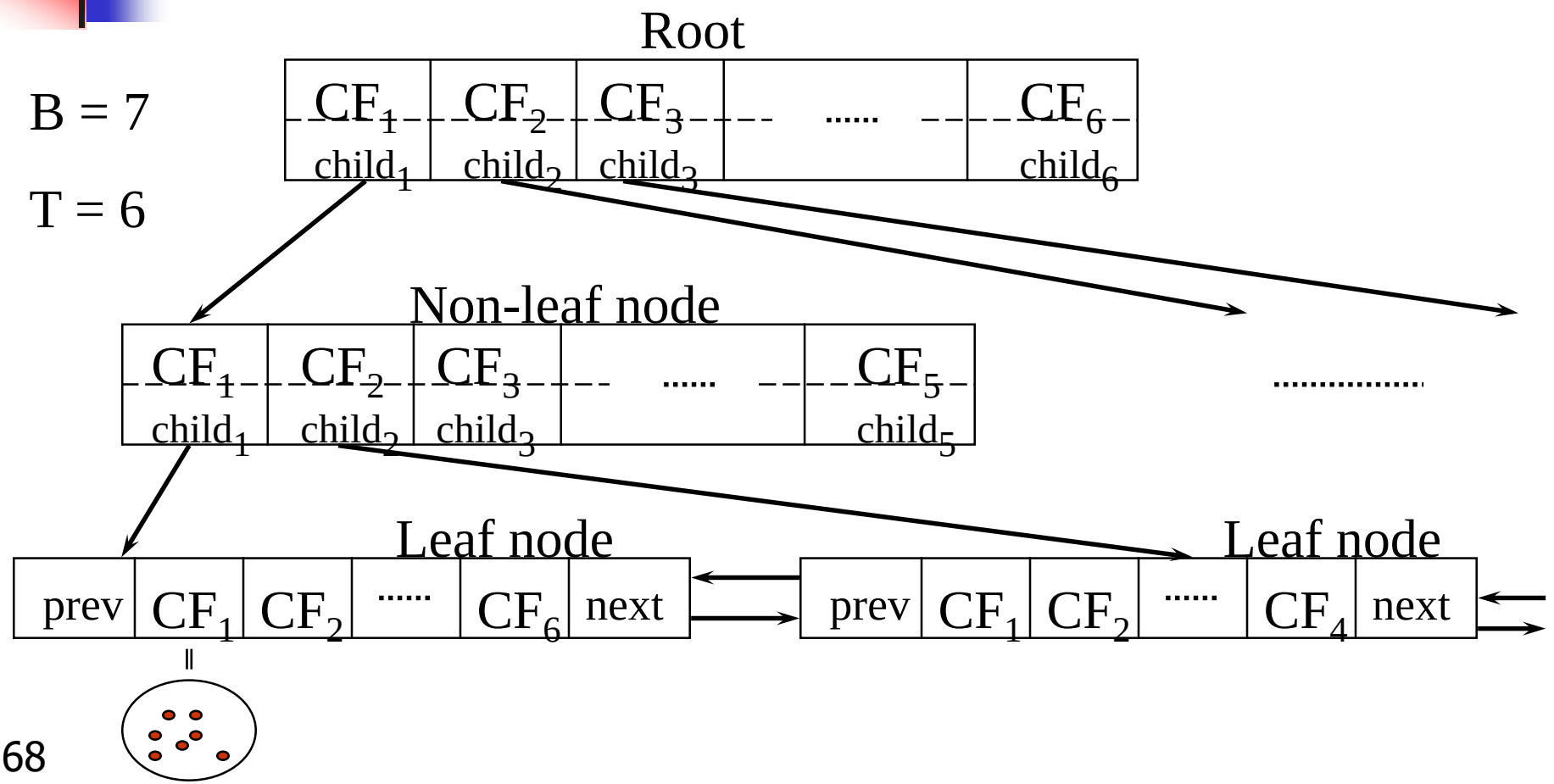
CF = (5, (16,30),(54,190))

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

BIRCH的CF树

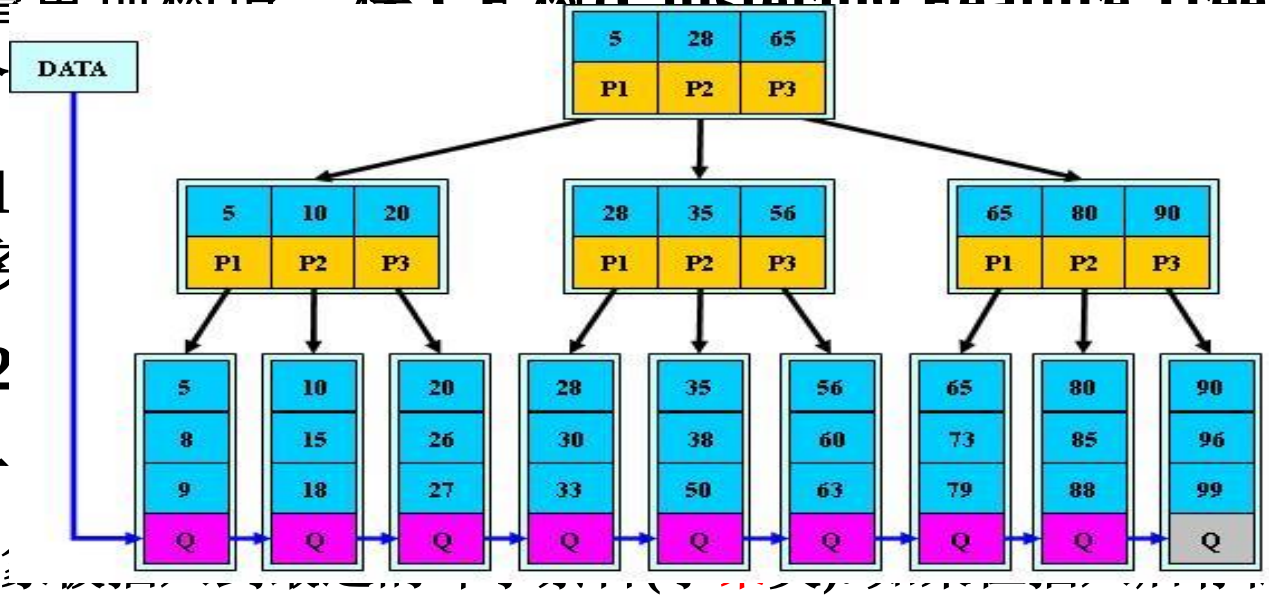
- 聚类特征
  - 从统计学的观点来看，聚类特征是对给定子类统计汇总：子聚类的0阶, 1阶和 2阶矩(moments)
  - 记录了计算聚类和有效利用存储的关键度量, 并有效地利用了存储,因为它汇总了关于子类的信息，而不是存储所有的对象
- CF 树是高度平衡的树，它存储了层次聚类的聚类特征
  - 树中的非叶节点有后代或“孩子”
  - 非叶节点存储了其孩子的CF的总和，即汇总了关于其孩子的聚类信息
- CF树有两个参数 ----影响CF树的大小

CF Tree


Root
B = 7
T = 6
CF1 child1
CF2 child2
CF3 child3
.....
CF6 child6
Non-leaf node
CF1 child1
CF2 child2
CF3 child3
.....
CF5 child5
Leaf node
prev
CF1
CF2
.....
CF6
next
Leaf node
prev
CF1
CF2
.....
CF4
next

BIRCH (续)

- BIRCH增量地构造一棵 CF 树(Clustering Feature Tree) , CF 树是一个



  - 阶段 1 树(数据的多
  - 阶段 2
- 在阶段一
  - 一个对 在叶子节点中的子类的直径大于阀值, 那么该叶子节点(可能还有其他节点)被分裂. 新对象插入后. 关于该对象的信息向着树根传递----类似于B+树构建中的插入和节点分裂

BIRCH (续)

- 重建过程从旧树的叶子节点建造一个新树。这样，重建树的过程不需要重读所有的对象 ----建树只需读一次数据
- 在阶段二被采用任何聚类算法，例如典型的划分方法
- **BIRCH的性能**
  - 支持增量聚类
  - 线性可伸缩性: 计算复杂性O(n), 单遍扫描, 附加的扫描可以改善聚类质量
  - 较好的聚类质量
- 缺点
  - 只能处理数值数据
  - 对数据的输入次序敏感

CURE(1998)

- CURE (Clustering Using REpresentatives) : 由 Guha, Rastogi 和 Shim提出(1998)
- 绝大多数聚类算法或者擅长处理球形和相似大小的聚类，或者在存在孤立点时变得比较脆弱
- CURE解决了偏好球形的问题，在处理孤立点上也更加健壮
- CURE采用了一种新的层次聚类算法
  - 选择基于质心和基于代表对象方法之间的中间策略. 它不用单个质心或对象来代表一个簇, 而是选择了数据空间中固定数目的具有代表性的点
  - 首先选择簇中分散的对象, 然后根据一个特定的收缩因

CURE(续)

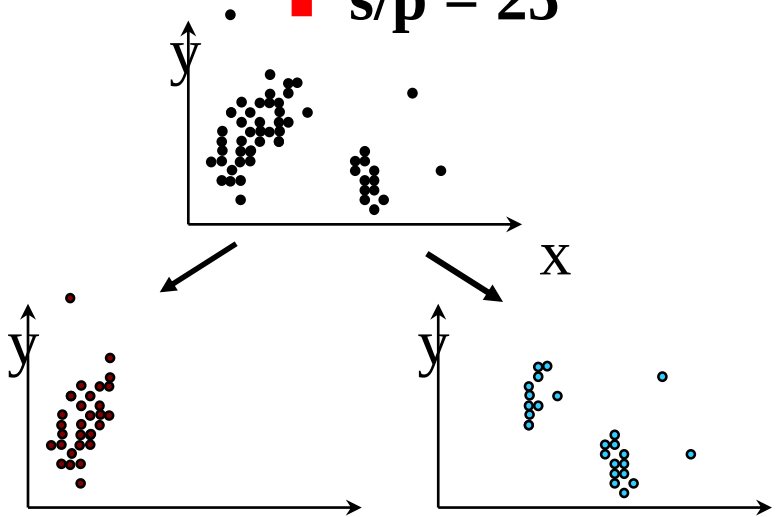
- 每个簇有多于一个的代表点使得CURE可以适应非球形的任意形状的聚类
- 簇的收缩或凝聚可以有助于控制孤立点的影响
- CURE的优点
  - CURE对孤立点的处理更加健壮
  - 能够识别非球形和大小变化较大的簇
  - 对于大规模数据库, 它也具有良好的伸缩性, 而且没有牺牲聚类质量
- 针对大型数据库, CURE采用了随机取样和划分两种方法的组合
  - 首先划分一个随机样本,每个划分被部分聚类

Cure(续)

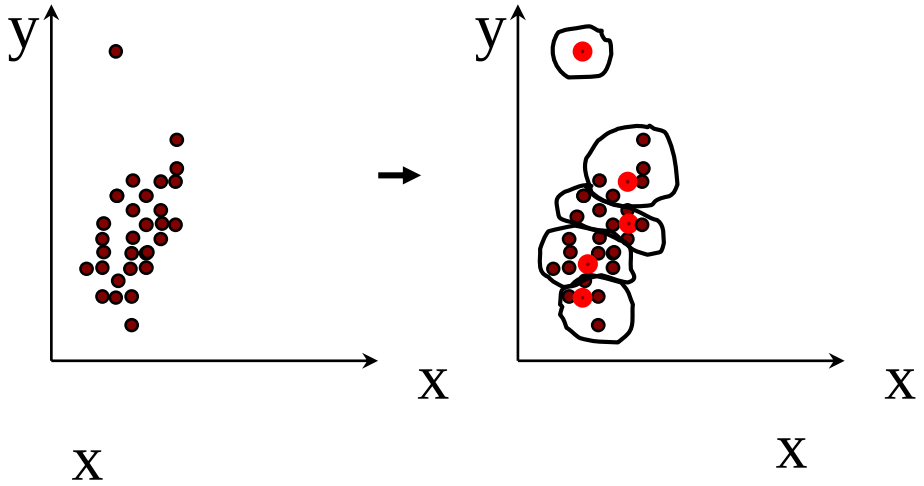
- **CURE算法核心:**
  - 从源数据对象中抽取一个随机样本S .
  - 将样本S分割为*p*个划分, 每个的大小为 *s/p*
  - 将每个划分局部地聚类成 *s/pq* 个簇
  - 删除孤立点
    - 通过随机选样
    - 如果一个簇增长太慢, 就删除它.
  - 对局部聚类进行聚类.
  - 用相应的簇标签来标记数据

CURE: 例

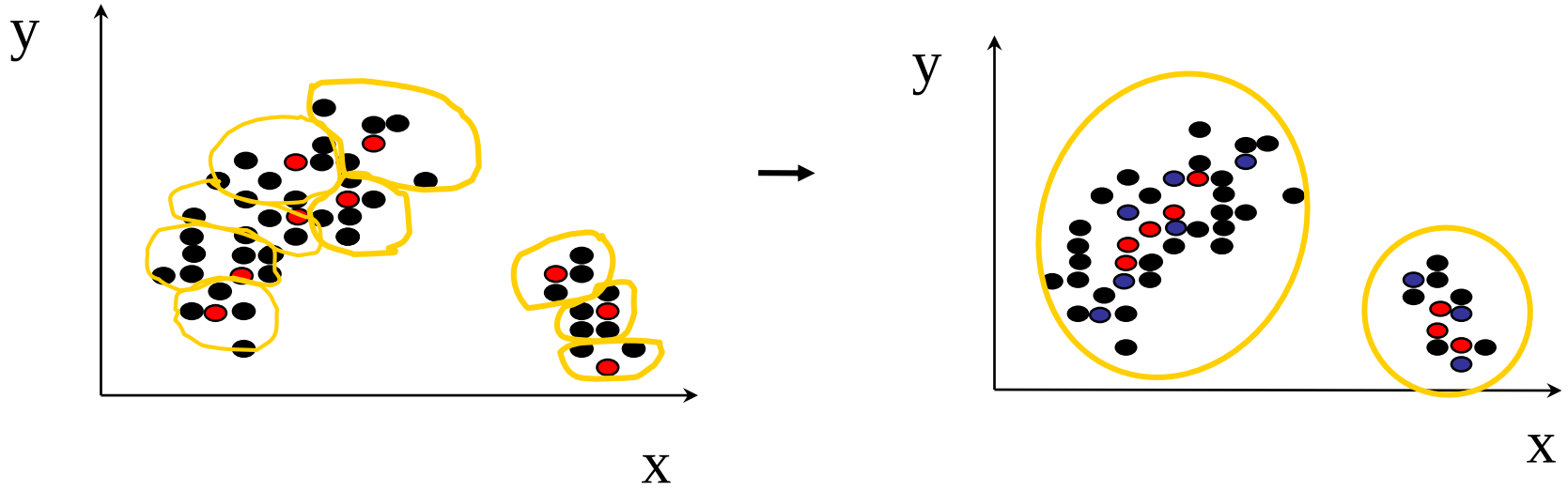
- s = 50
- p = 2
- s/p = 25



- s/pq = 5



CURE：例(续)



- 多个代表点向重心 以因子α移动, 进行收缩或凝聚
- 多个代表点描述了每个簇的形状

对分类数据聚类: ROCK

- **ROCK(RObust Clustering using linKs) 由S. Guha, R. Rastogi, K. Shim提出 (ICDE'99).**
  - 使用链接(link)度量相似性/接近性
    - 链接：两个对象间共同的近邻的数目
  - 不是基于距离的
  - 计算复杂性: $O(n^2 + nm_m m_a + n^2 \log n)$
- 基本思想:
  - 相似性函数:Jaccard系数

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

$$Sim(T1, T2) = \frac{|\{3\}|}{|\{1,2,3,4,5\}|} = \frac{1}{5} = 0.2$$

设 T_1 = {1,2,3}, T_2={3,4,5}

Rock(续)

- 两个点pi和p_j是近邻，如果sim(pi,p_j)>=用户指定阈值
- link(p_i,p_j)是两个点pi和p_j共同的近邻的数目
- 两个簇Ci和Cj的互连性被定义为两个簇间交叉链（cross link）的数目 $\sum_{p_q \in C_i, p_r \in C_j} link(p_q, p_r)$

- ROCK首先根据相似度阀值和共享近邻的概念，从给定的

we define the *goodness measure* $g(C_i, C_j)$ for merging clusters C_i, C_j

$$g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

goodness measure is maximum is the best pair of clusters to be merged at any given step.

CHAMELEON

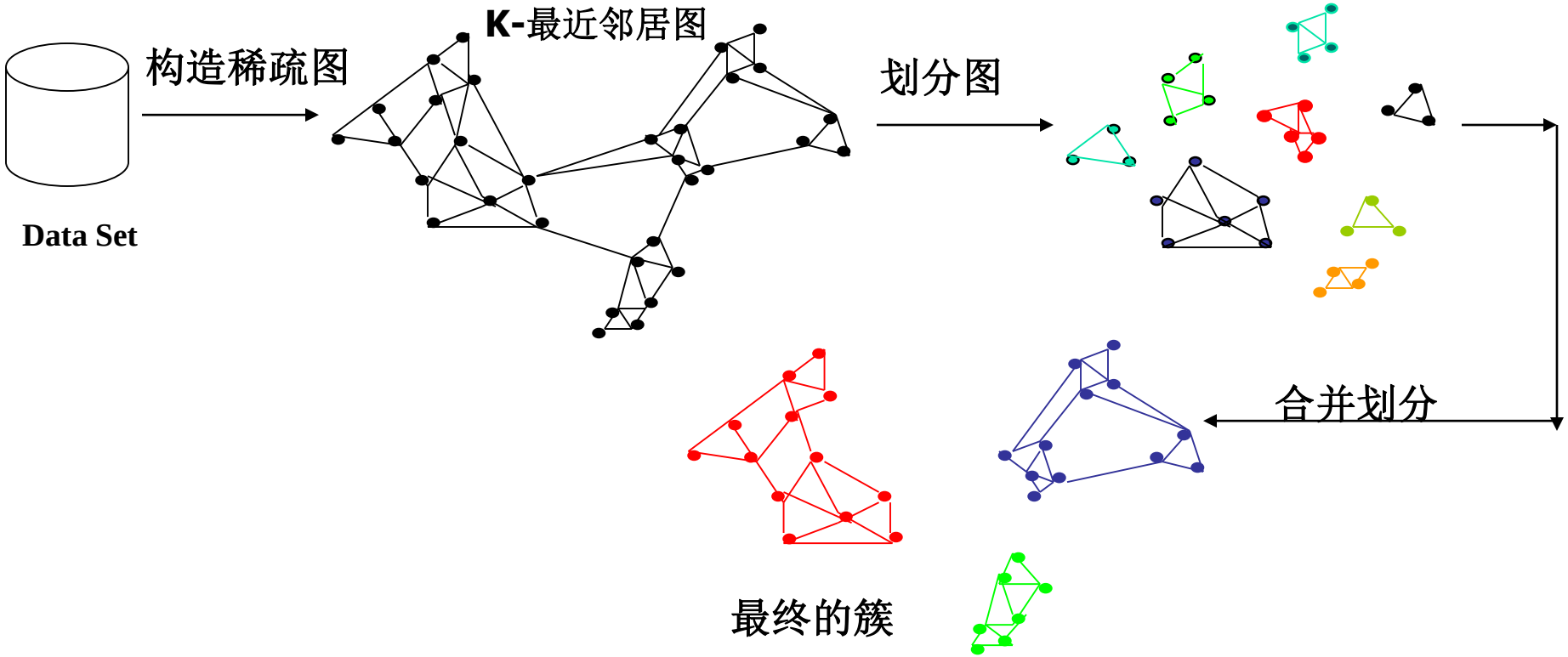
- **CHAMELEON :一个利用动态模型的层次聚类算法 (Hierarchical clustering using dynamic modeling) 由G. Karypis, E.H. Han, and V. Kumar’99 提出**
- **对CURE和ROCK缺点的观察:**
  - **Cure忽略了关于两个不同簇中对象的聚集互连性的信息**
  - **Rock强调对象间互连性, 却忽略了关于对象间近似度的信息**
- **CHAMELEON基于动态模型度量相似性**
  - **如果两个簇间的互连性和近似度与簇内部对象间的互连性和近似度高度相关, 则合并这两个簇**

CHAMELEON(续)

- 两阶段算法
  1. 使用图划分算法: 将数据对象聚类为大量相对较小的子类

     逐步用图划分算法把k近邻图分成 相对较小de子簇，最小化割边。
  2. 使用凝聚的层次聚类算法：通过反复地合并子类来找到真正的结果簇
- 既考虑互连性, 又考虑簇间的近似度, 特别是簇内部的特征, 来确定最相似的子类.
- 这样，它不依赖于静态的用户提供的模型，能够自动地适应被合并的簇的内部特征
  - 割边最小化——簇c划分为两个子簇C_i和C_j时需要割断的

CHAMELEON图示



CHAMELEON(续)

- k-最近邻图G_k: 图中的每个点代表一个数据对象, 如果一个对象是另一个对象的k个最类似的对象之一, 在这两个点之间存在一条边

- k-最近邻图G_k动态地捕捉邻域的概念：一个对象的邻域半径由对象所在区域的密度所决定
  - 在一个密集区域, 邻域的定义范围相对狭窄; 在一个稀疏区域, 它的定义范围相对较宽

- 区域的密度作为边的权重被记录下来

CHAMELEON(续)

- Chameleon通过两个簇的相对互连性RI(C_i, C_j)和相对接近度RC(C_i, C_j)来决定簇间的相似度
  - RI(C_i, C_j)定义为C_i和C_j之间的绝对互联性关于两个簇的内部互连性的规范化

$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}$$

其中, $EC_{\{Ci,\ Cj\}}$是包含C_i和C_j的簇分裂为C_i和C_j的割边, EC_{Ci}（或EC_{Cj}）是它的最小截断等分线的大小（即将图划分为两个大致相等的部分需要切断的边的加权和）

CHAMELEON(续)

- RC(C_i, C_j)定义为Ci和Cj之间的绝对接近度关于两个簇的内部接近度的规范化

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

$\overline{S}_{EC(C_i, C_j)}$

$\overline{S}_{EC_{C_i}}$

其中，　　是连接C_i和C_j顶点的边的平均权重

是C_i的最小二等分的边的平均权重

第7章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）

基于密度的方法

- 基于密度聚类 (Density-Based Clustering)
- 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
- 一些有趣的研究:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

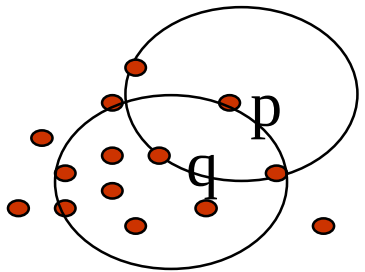
密度概念

- ε-邻域：给定对象半径ε内的领域
- 核心对象 (Core object): 一个对象的ε–邻域至少包含最小数目MinPts个对象
- 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ε–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
- 密度可达的(density reachable): 存在 一个从p到q的DDR对象链

基于密度的聚类: 背景I

- 两个参数:
  - *Eps*: 邻域的最大半径
  - *MinPts*: 在 Eps-邻域中的最少点数
  - $N_{Eps}(p)$: *{q belongs to D | dist(p,q) <= Eps}*
- 直接密度可达的: 点 *p* 关于*Eps, MinPts* 是从点*q*直接密度可达的, 如果
  - 1) *p* 属于 $N_{Eps}(q)$
  - 2) 核心点条件:

$$|N_{Eps}(q)| >= MinPts$$

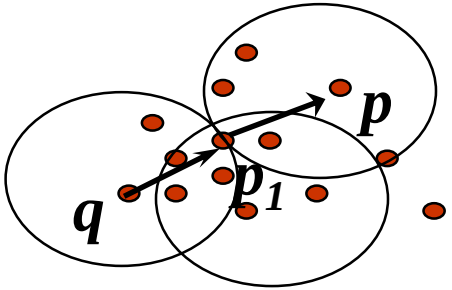


MinPts = 5

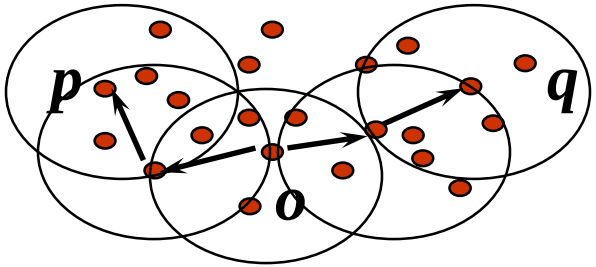
Eps = 1 cm

基于密度的聚类: 背景II

- 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 $p_1, \ldots, p_n, p_1 = q, p_n = p$ 使得 p_{i+1} 是从p_i直接密度可达的

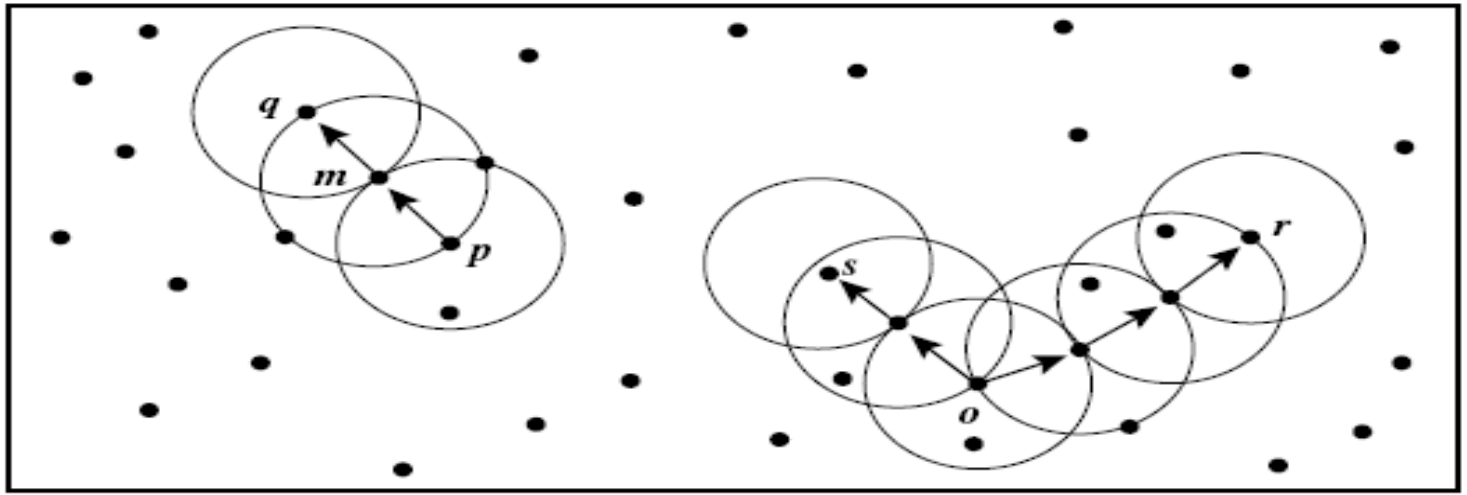


- 密度相连的:
  - 点 *p*关于 *Eps*, *MinPts* 与点 *q*是密度相连的, 如果 存在点 *o* 使得, *p* 和 *q* 都是关于*Eps*, *MinPts* 是从 *o* 密度可达的



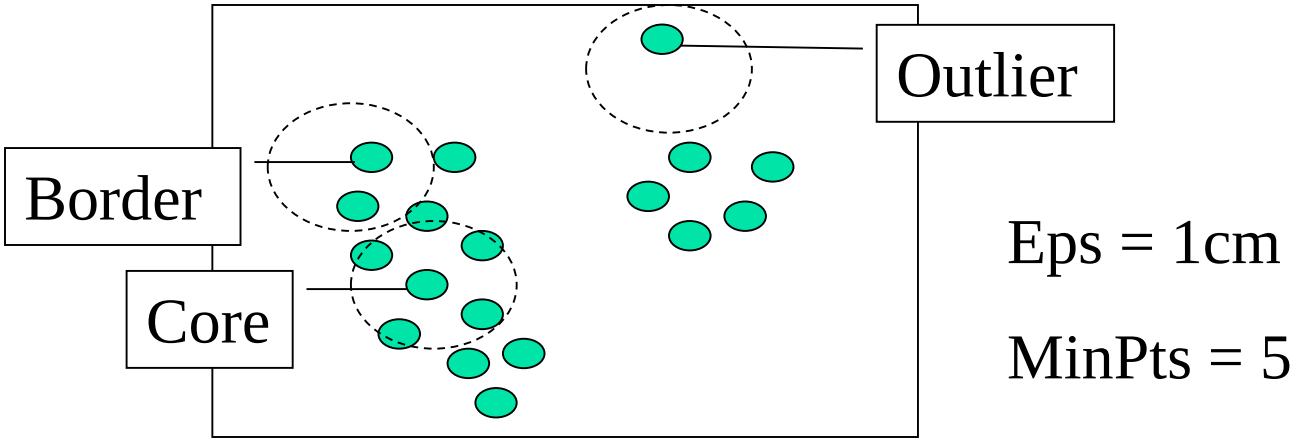
例子

- **MinPts=3**
  - **q是从p密度可达； p不是从q密度可达（q非核心）**
  - **S和r从o密度可达； o从r密度可达；**
  - **r, s, o密度相连**



DBSCAN(1996)

- **DBSCAN(Density Based Spatial Clustering of Applications with Noise) 一个基于密度的聚类算法**
- **可以在带有“噪音”的空间数据库中发现任意形状的聚类**



DBSCAN(续)

- 算法
  - 任意选取一个点 *p*
  - 得到所有从*p* 关于 *Eps* 和 *MinPts*密度可达的点.
  - 如果*p* 是一个核心点, 则找到一个聚类.
  - 如果 *p* 是一个边界点, 没有从*p* 密度可达的点, DBSCAN 将访问数据库中的下一个点.
  - 继续这一过程, 直到数据库中的所有点都被处理.
- DBSCAN的复杂度
  - 采用空间索引, 复杂度为O(nlog n), 否则为$O(n^2)$
- DBSCAN的缺点:

OPTICS (1999)

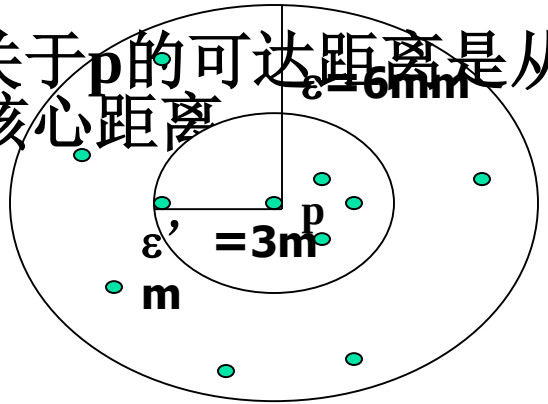
- OPTICS(Ordering Points To Identify the Clustering Structure)
  - Ankerst, Breunig, Kriegel, 和 Sander 提出(SIGMOD'99)
  - 为自动和交互的聚类分析计算一个簇次序(cluster ordering).
  - 这个次序代表了数据的基于密度的聚类结构。它包含了信息, 等同于从一个广域的参数设置所获得的基于密度的聚类
  - 可用于自动和交互聚类分析, 包括发现内在聚类结构

OPTICS(续)

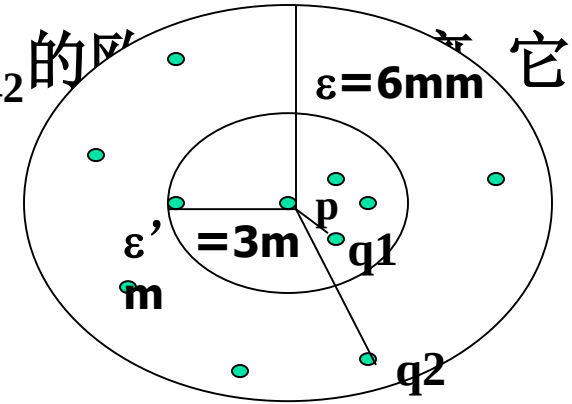
- 考虑DBSCAN, 对一个恒定的MinPts值, 关于高密度的(即较小的ε值)的聚类结果被完全包含在根据较低密度所获得的密度相连的集合中
- 扩展DBSCAN算法来同时处理一组距离参数值
- 为了同时构建不同的聚类, 应当以特定的顺序来处理对象. 优先选择最小的ε值密度可达的对象, 以便高密度的聚类能被首先完成
- 每个对象需要存储两个值
  - 对象p的核心距离(core-distance)是使得p成为核心对象的最小ε。如果p不是核心对象, p的核心距离没有定义
  - 对象q关于另一个对象p的可达距离(reachability-distance)

OPTICS(续)

- 例: 设ε=6(mm), MinPts=5.
  - p的核心距离是p与第四个最近的数据对象之间的距离ε’.
  - q_1关于p的可达距离是p的核心距离(即ε’=3mm), 因为它比从p到q_1的欧几里得距离要大.
  - q_2关于p的可达距离是从p到q_2的欧[illegible]离, 它大于p的核心距离



P的核心距离



可达距离 (p,q1)=ε’=3mm

可达距离 (p,q2)=d(p,q2)

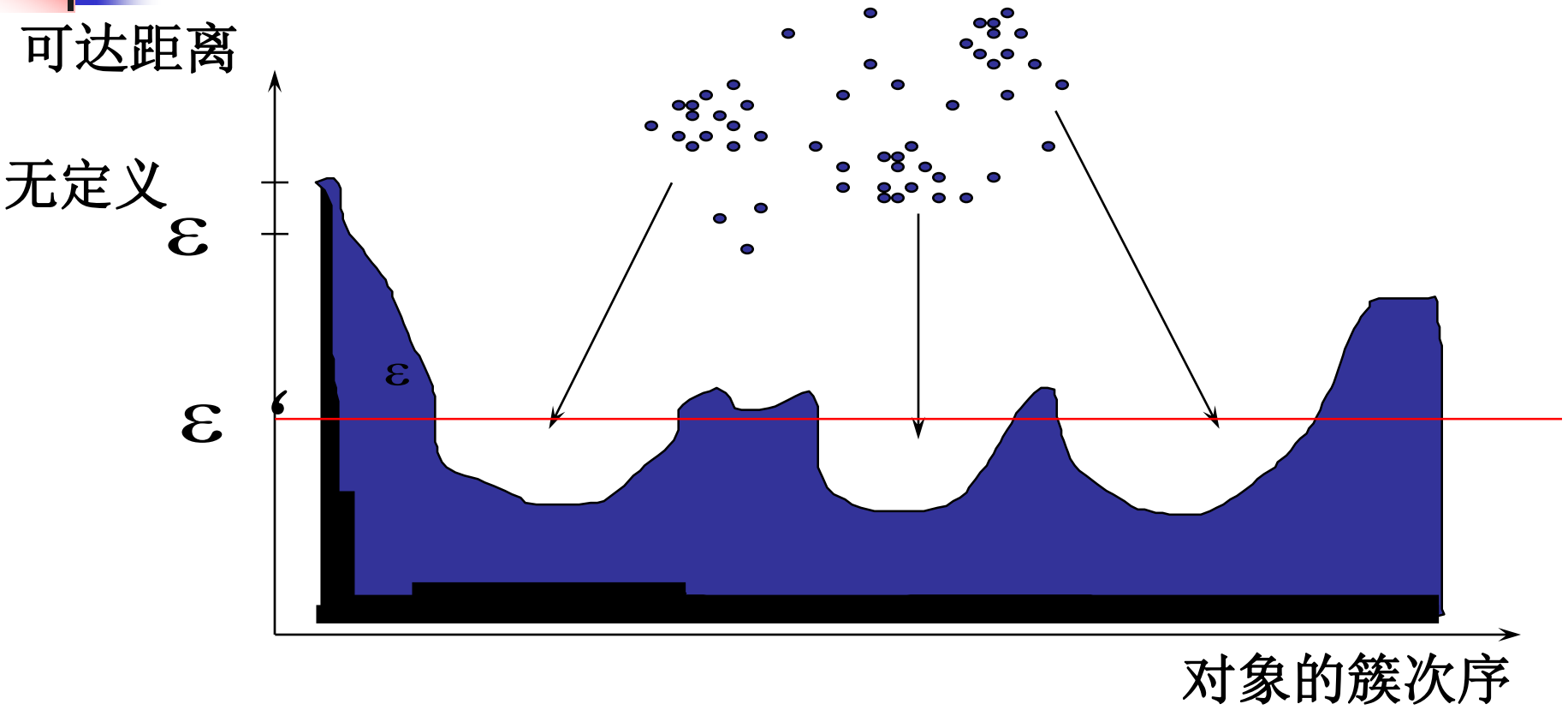
OPTICS(续)

Definition 7: (results of the OPTICS algorithm)

- Let DB be a database containing n points. The OPTICS algorithm generates an ordering of the points $o: \{1..n\} \rightarrow DB$ and corresponding reachability-values $r: \{1..n\} \rightarrow \boldsymbol{R}_{\geq 0}$.

  对每个对象的核心距离和一个适当的可达距离

  - 已经提出了一种算法, 基于OPTICS产生的次序信息来抽取聚类. 对于小于在生成该次序中采用的距离ε的任何距离ε’, 为提取所有基于密度的聚类, 这些信息是足够的
- 一个数据集合的聚类次序可以被图形化地描述，以助于理解
-
- 由于OPTICS算法与DBSCAN在结构上的等价性, 它具有和DBSCAN相同的时间复杂度, 即当使用空间索引时, 复杂度


可达距离
无定义
ε
ε
ε
对象的簇次序

DENCLUE(1998)

- DENCLUE(DENsity-based CLUstEring) 由Hinneburg 和 Keim (KDD'98)提出, 是基于密度分布函数的聚类方法
- 主要特点
  - 坚实的数学基础, 概括了其他的聚类方法, 包括基于划分的, 层次的, 及基于位置的方法
  - 适用于具有大量噪音的数据集
  - 可用于高维数据集任意形状的聚类, 它给出了简洁的数学描述
  - 明显快于现有算法 (比 DBSCAN 快 45倍)
  - 但是, 需要大量参数,要求对密度参数σ和噪音阀值ξ进行

Denclue: 技术要点

- 使用栅格单元, 但只保存实际存放数据点的栅格单元信息, 并且在一个基于树的存取结构中管理这些单元.
- 影响函数(Influence function): 描述数据点在其邻域的影响.
- 数据空间的整体密度可以被模拟为所有数据点的影响函数的总和
- 聚类可以通过确定密度吸引点 (density attractor)来得到.
- 密度吸引点是全局密度函数的局部最大值.

DENCLUE(续)

- 设x和y是d维特征空间F^d中的对象. 数据对象y对x的**影响函数**是一个函数f^y_B: $F^d \rightarrow R^+_0$, 它是根据一个基本的影响函数 f_B来定义的

$$f^y_B(x) = f_B(x, y)$$

- 原则上, 影响函数可以是一个任意的函数, 它由某个邻域内的两个对象之间的距离来决定
- 例如欧几里得距离函数, 用来计算一个方波影响函数(square wave influence function):

$$f^y_{Square}(x) = \begin{cases} 0 & \text{如果} d(x,y) > \sigma \\ 1 & \text{其它} \end{cases}$$

DENCLUE(续)

- 高斯影响函数

$$f_{Gauss}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

- 一个对象$x \in F^d$的密度函数被定义为所有数据点的影响函数的和. 给定n个对象, $D=\{x1,\ldots,xn\} \subset F^d$, 在$x$上的密度函数定义如下 $f_D(x) = \sum_{i=1}^{n} f_B^{x_i}(x)$

DENCLUE(续)

- 例如, 根据高斯影响函数得出的密度函数是

$$f_{Gauss}^{D}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

- 根据密度函数, 我们能够定义该函数的梯度和密度吸引点(全局密度函数的局部最大)
- 一个点x是被一个**密度吸引点** x*密度吸引的, 如果存在一组点$x_0, x_1, \ldots, x_k$, $x_0=x$, $x_k=x^*$, 对$0<i<k$，x_{i-1}的梯度是在x_i的方向上
- 对一个连续的, 可微的影响函数, 用梯度指导的爬山算法

密度吸引点

Def. 2 (Gradient)
The gradient of a function $f_B^D(x)$ is defined as

$$\nabla f_B^D(x) = \sum_{i=1}^{N} (x_i - x) \cdot f_B^{x_i}(x).$$

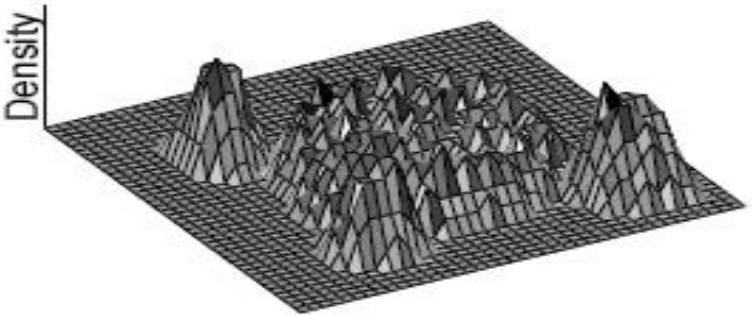
In case of the Gaussian influence function, the gradient is defined as:

$$\nabla f_{Gauss}^D(x) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}.$$

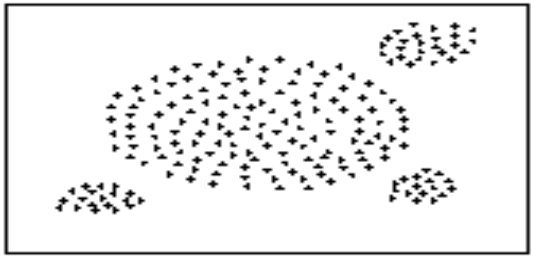
Def. 3 (Density-Attractor)
A point $x^* \in F^d$ is called a *density-attractor* for a given influence function, iff x^* is a local maxium of the density-function f_B^D.
A point $x \in F^d$ is *density-attracted* to a density-attractor x^*, iff $\exists k \in N : d(x^k, x^*) \leq \epsilon$ with

$$x^0 = x, \; x^i = x^{i-1} + \delta \cdot \frac{\nabla f_B^D(x^{i-1})}{\|\nabla f_B^D(x^{i-1})\|}.$$

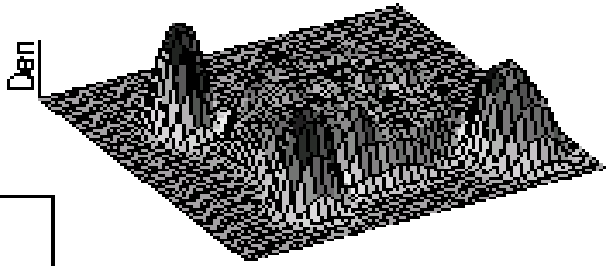
密度吸引点



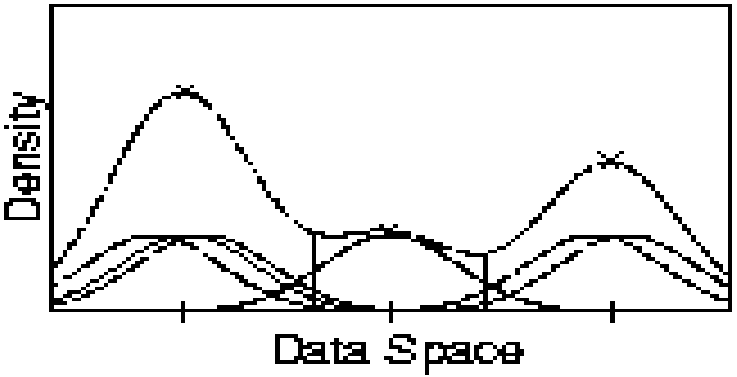
(b) Square Wave



(a) Data Set



(c) Gaussian

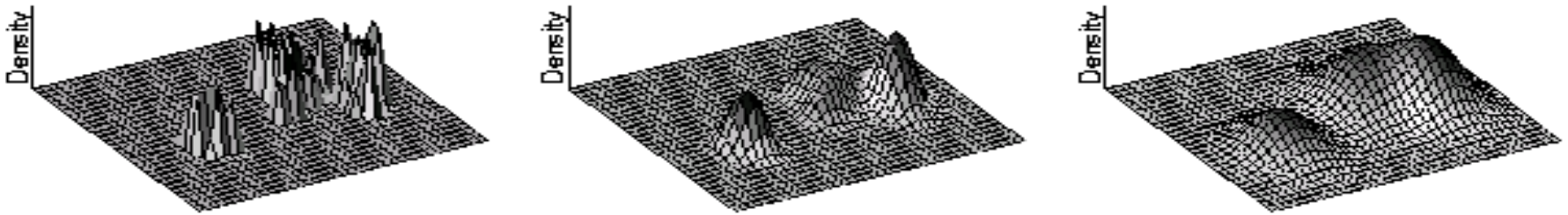


中心定义的簇和任意形状的簇

- 密度吸引点 x^*的中心定义的簇(center-defined cluster)是一个被x^*密度吸引的子集C, 在x^*的密度函数不小于一个阀值ξ; 否则(即如果它的密度函数值小于ξ), 它被认为是孤立点

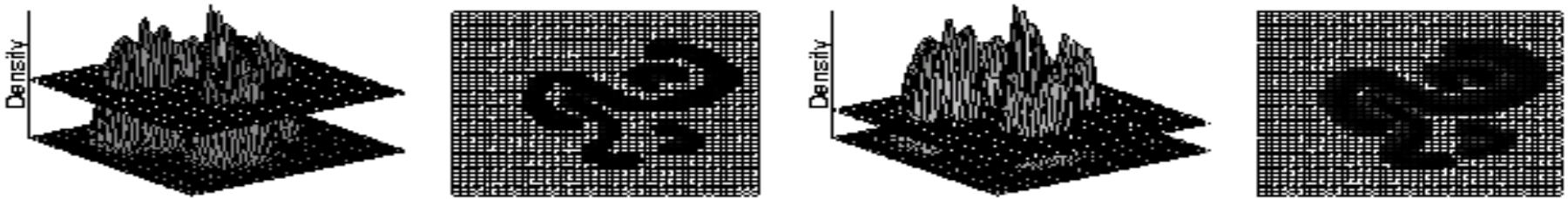
- 一个任意形状的簇(arbitrary-shape cluster)是子集C的集合, 每一个是各自密度吸引子密度吸引的,有不小于阀值ξ的密度函数值, 从每个区域到另一个都存在一条路径P, 该路径上每个点的密度函数值都不小于ξ

中心定义的簇和任意形状的簇



(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

Figure 3: Example of Center-Defined Clusters for different σ



(a) $\xi = 2$ (b) $\xi = 2$ (c) $\xi = 1$ (d) $\xi = 1$

Figure 4: Example of Arbitray-Shape Clusters for different ξ

第9章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）

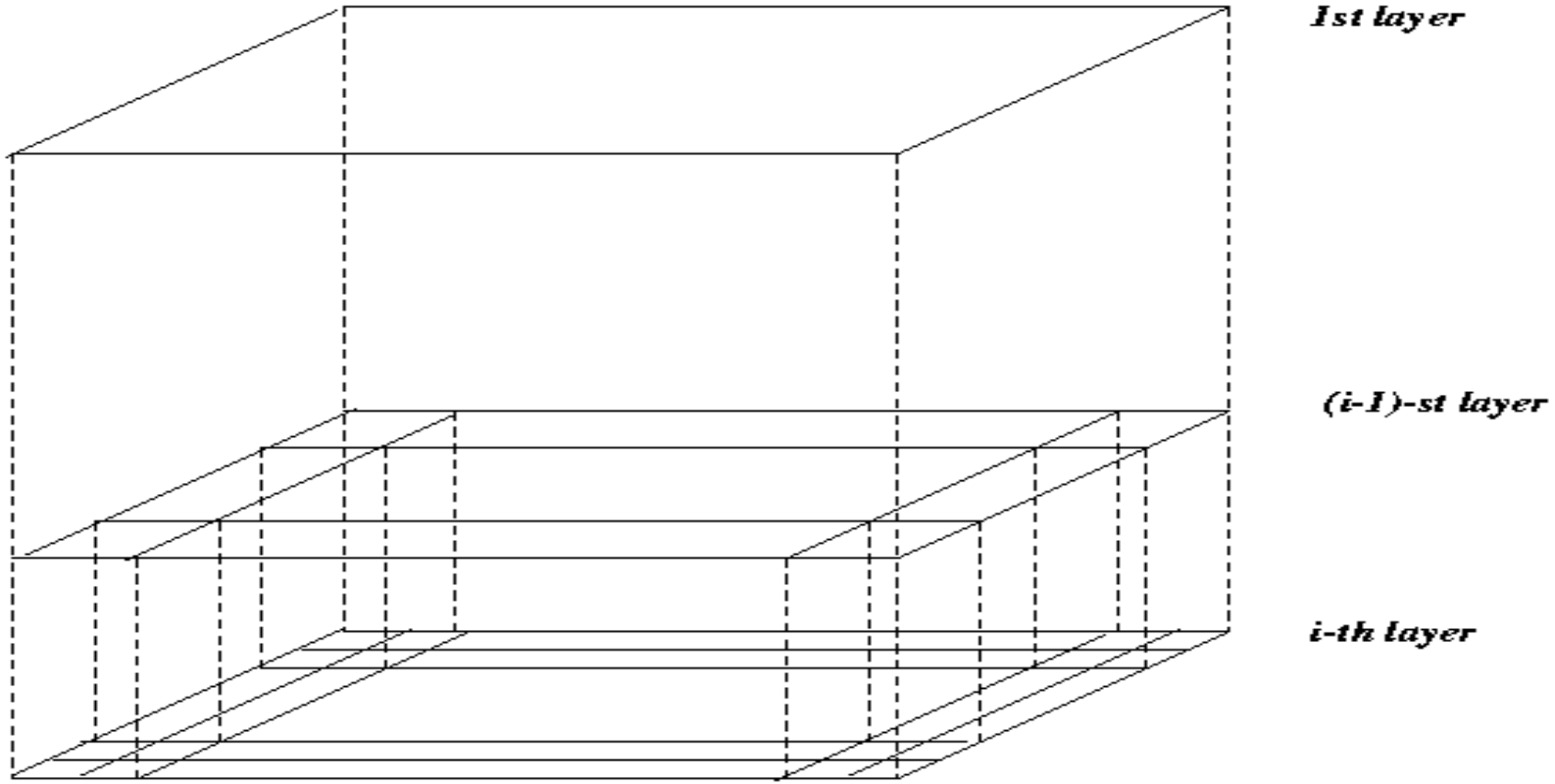
基于网格的聚类方法

- 使用多分辨率的网格数据结构
- 一些有趣的方法
  - **STING (a STatistical INformation Grid approach)** 由**Wang, Yang 和 Muntz 提出(1997)**
  - **WaveCluster 由 Sheikholeslami, Chatterjee, 和 Zhang 提出(VLDB'98)**
    - 采用小波方法的多分辨率的聚类方法
  - **CLIQUE: Agrawal, et al. 提出(SIGMOD'98)**

STING: 统计信息网格

- **STING(STatistical INformation Grid)是一个基于网格的多分辨率聚类技术, 由Wang, Yang 和 Muntz 提出 (VLDB'97)**
- 空间区域划分为矩形单元
- 多个级别的矩形单元, 对应不同级别的分辨率. 这些单元形成了一个层次结构: 每个高层单元被划分为多个低一层的单元
- 预先计算和存储关于每个网格单元属性的统计信息 (如平均值, 最大值, 和最小值), 用于回答查询

STING(续)



STING(续)

- 高层单元的统计参数可以很容易地从低层单元的计算得到. 这些统计参数包括：
  - 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)
  - 该单元中属性值遵循的分布类型：正态的, 一致的, 指数的, 无(分布未知)
- 分布的值可以由用户指定, 也可以通过假设检验(如χ^2检验)来获得
- 最底层单元的参数count, m, s, min, 和max直接进行计算

STING(续)

- 使用自顶向下的方法回答空间数据查询
  - 在层次结构中选定一层作为查询处理的开始点——通常，该层包含少量的单元
  - 对当前层次的每个单元，计算置信度区间(或者估算其概率)，用以反映该单元与给定查询的关联程度
  - 删除不相关的单元，进一步处理不考虑它们
  - 结束当前层的考查后，就处理下一层
  - 重复这一过程，直到最低层

STING(续)

- 优点:
  - 基于网格的计算是独立于查询的：存储在每个单元中的统计信息是不依赖于查询的汇总信息
  - 网格结构有利于并行处理和增量更新
  - 效率很高：STING扫描数据库一次来计算单元的统计信息, 因此产生聚类的时间复杂度是$O(n)$, 其中, n是对象的数目

    层次结构建立后, 查询处理时间是 $O(K)$, K是最底层网格单元的数目, 通常远远小于n
- 缺点:

WaveCluster (1998)

- 由Sheikholeslami, Chatterjee, 和Zhang (VLDB'98) 提出
- 采用小波变换聚类: 是一种多分辨率的聚类算法, 对特征空间采用小波变换(wavelet transform)
  - 小波变换是一种信号处理技术, 它将信号压缩到 不同频率的子波段.
- 既是基于网格的方法, 又是基于密度的方法 grid-based and density-based
- 输入参数:
  - 每维单元的数目
  - 小波和应用小波变换的次数

WaveCluster (1998)

- 如何使用小波变换找出聚类
  - 首先通过在数据空间上强加一个多维网格结构来汇总数据
  - 用n-维向量空间表示这些多维空间数据对象
  - 对特征空间施加小波变换, 找出特征空间中的稠密区域
  - 使用小波变换多次, 得到由细到粗不同尺度的聚类
- 小波变换
  - 将信号分解到不同频率的子波段(可以用于n-维信号)
  - 变换后的数据在不同的分辨率下保留对象之间的相对距离.

WaveCluster

- 为什么小波变换对聚类是有用的
  - 提供了无指导的聚类

    它采用了帽形(hat-shape)过滤, 强调点密集的区域, 而忽视在密集区域外的较弱的信息----特征空间中的密集区域成为了附近点的吸引点(attractor), 距离较远的点成为抑制点(inhibitor). 这意味着数据的聚类自动地显示出来, 并“清理”了周围的区域
  - 能够自动地排除孤立点
  - 多分辨率
    - 图8.16显示了不同分辨率的小波变换结果, 从细的尺度到粗的尺度. 在每一个层次, 显示了原始数据分解得到的四个子波段. 左上

Quantization

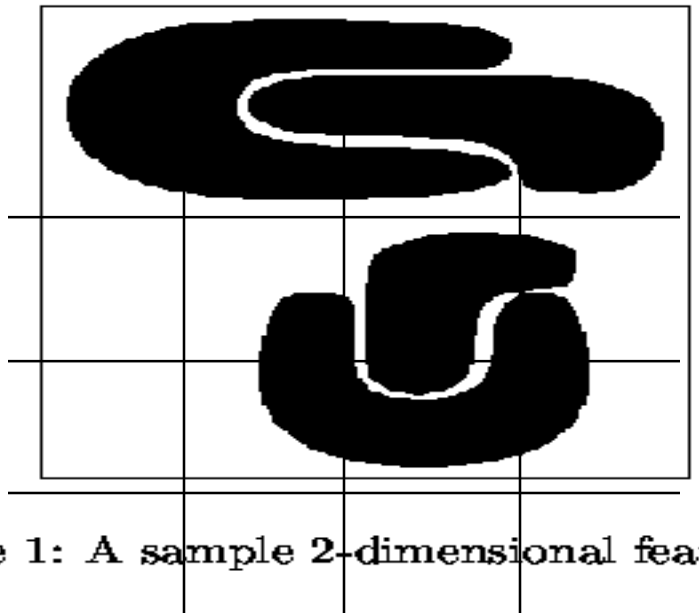


Figure 1: A sample 2-dimensional feature space.

Transformation

- 量化数据m-D grid structure, then wavelet transform
  - a) scale 1: high resolution
  - b) scale 2: medium resolution
  - c) scale 3: low resolution

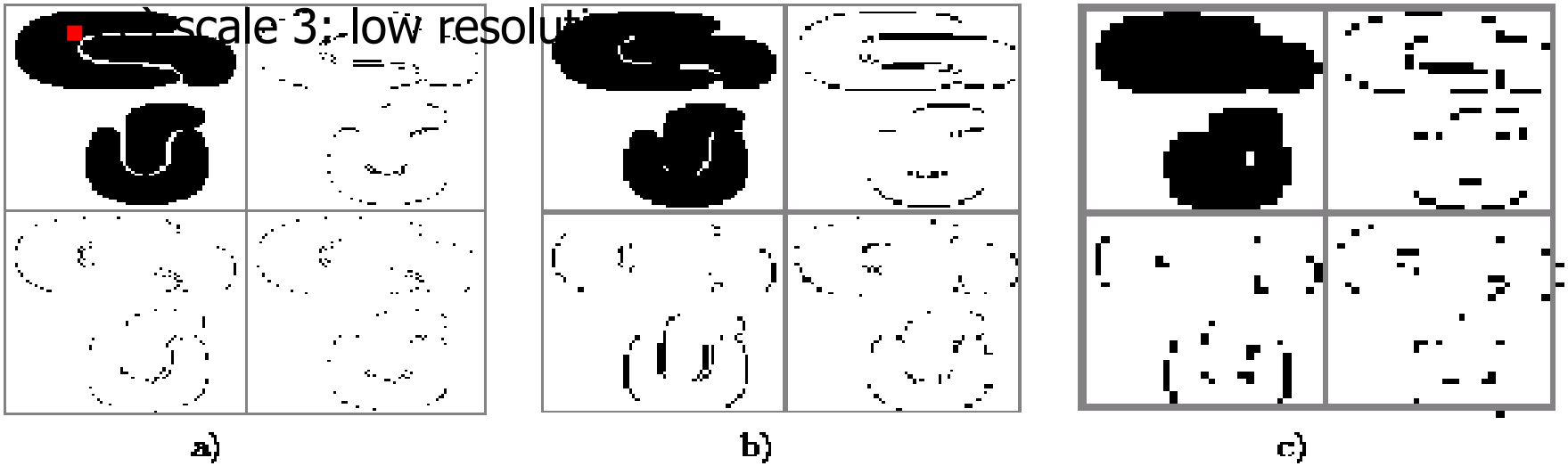


图7-16 特征空间的多分辨率的结果

WaveCluster

- 主要特点:
  - 复杂性 O(N)
  - 检测不同的尺度下的任意形状聚类
  - 对噪音不敏感, 对孤立点不敏感
  - 通常只能处理低维数据

第7章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）

基于模型的聚类方法

- 试图优化给定的数据和某些数学模型之间的拟合

- 三类基于模型的方法:
  - 统计学方法
    - 期望最大化方法
  - 概念聚类
  - 神经网络方法

EM 方法

- 每个分布代表一个簇，k个概率分布的混合模型表示整个数据，估计分布参数拟合数据
- EM — 一种流行的迭代求精算法，k-means方法的一种扩展
  - 根据权重 (prob. distribution) 把对象指派到簇cluster
  - 基于权重度量计算新的均值
- 基本想法
  - 对参数向量进行初始估计/猜测
  - 反复地根据参数向量产生的混合密度对每个对象重新打分
  - 重新打分后的对象又用来更新参数向量

The EM Algorithm

- 初始, 随机选择k 聚类中心
- 根据如下两步迭代求精
  - 期望步: 用以下概率将数据点 x_i 指派到cluster C_i

$$P(X_i \in C_k) = p(C_k|X_i) = \frac{p(C_k)p(X_i|C_k)}{p(X_i)},$$

  - 最大
    - I $m_k = \frac{1}{N}\sum_{i=1}^{N}\frac{X_i P(X_i \in C_k)}{\sum_j P(X_i \in C_j)}$.

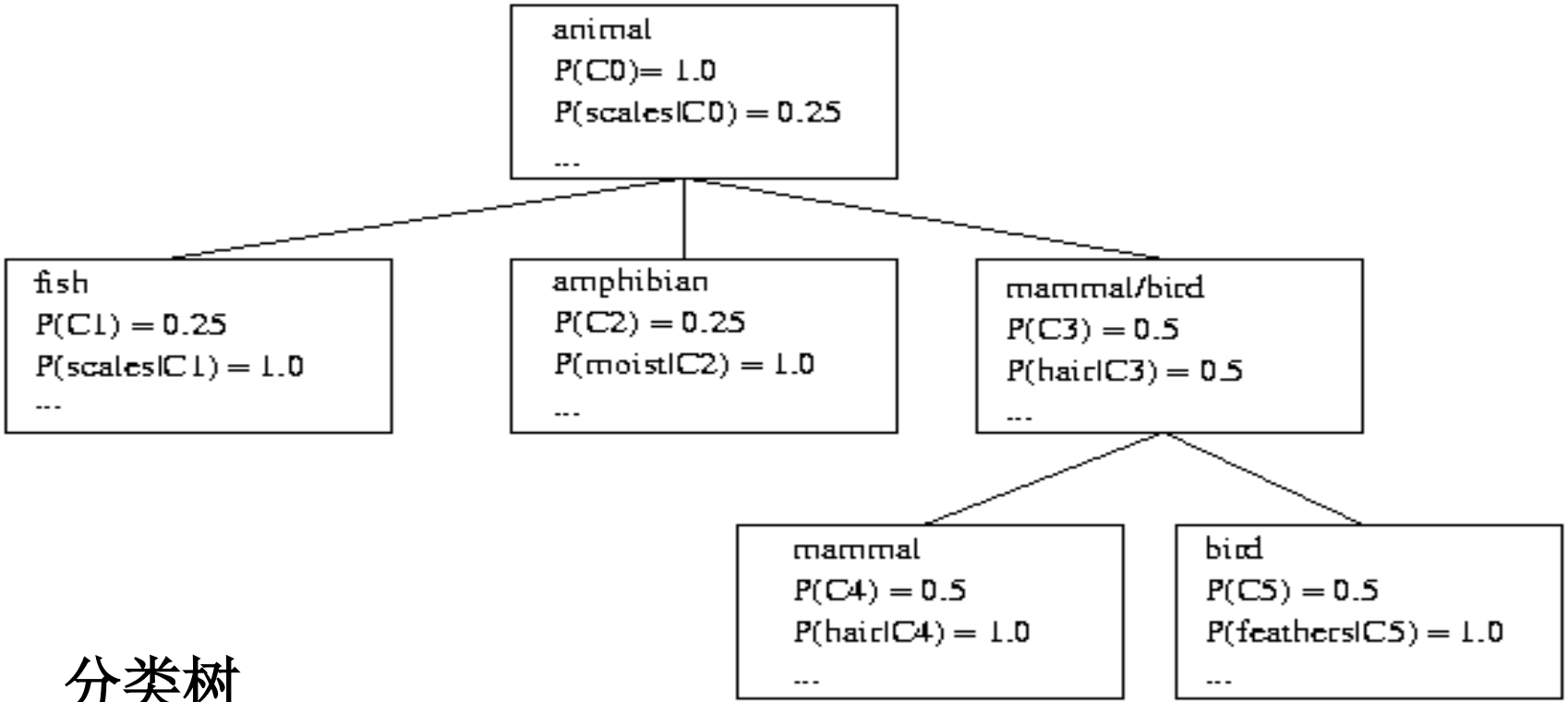
概念聚类

- 概念聚类
  - 一种机器学习聚类方法
  - 给出一组未标记的对象，它产生一个分类模式
  - 为每组对象找出特征描述
- **COBWEB (Fisher’87)**
  - 一种简单的、流行的增量概念聚类算法
  - 以一个分类树的形式创建层次聚类

COBWEB

- 分类树
  - 分类树中的每个节点对应一个概念, 包含该概念的一个概率描述, 概述了被分在该节点下的对象
  - **概率描述**包括概念的概率和形如$P(A_i = V_{ij} | C_k)$的条件概率, 这里$A_i = V_{ij}$是一对属性和值, C_k是概念类
  - 为了用分类树对一个对象进行分类, 采用了一个部分匹配函数来沿着最佳匹配节点的路径在树中向下移动
- 分类树VS判定树
  - 判定树标记分支, 而非节点, 而且采用逻辑描述符, 而不是概率描述符

COBWEB 聚类方法



分类树

COBWEB

- **COBWEB采用了一个启发式估算度量——分类效用(Category Utility,CU)来指导树的构建, 分类效用定义如下**

$$\frac{\sum_{i=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2]}{n}$$

  - **n是在树的某个层次上形成一个划分{C1, C2, …, Cn}的节点, 概念或“种类”的数目**
  - **概率$P(A_i = V_{ij} \mid C_k)$ 表示类内相似性. 该值越大, 共享该属性-值的类成员比例越大, 该属性-值对类成员的预见性就越大**
  - **$P(C_k \mid A_i = V_{ij})$ 表示类间相异性. 该值越大, 共享该属性-值**

COBWEB

- COBWEB将对象增量地加入到分类树中
  - 给定一个新的对象, COBWEB沿着一条适当的路径向下, 修改计数, 寻找可以分类该对象的最好节点
    - 基于将对象临时置于每个节点, 计算结果划分的分类效用.
    - COBWEB也计算为给定对象创建一个新的节点所产生的分类效用.
  - 与基于现存节点的结果相比较, 根据产生最高分类效用的划分, 对象被置于一个已存在的类, 或者为它创建一个新类
  - 要注意COBWEB可以自动修正划分中类的数目, 它不需要用户提供这样的输入参数

COBWEB

- COBWEB的限制
  - 属性上的概率分布是彼此独立的 假定太强, 因为相关性可能存在
  - 不适合对大型数据库中的数据进行聚类: 倾斜的树, 计算概率分布的代价太高

- COBWEB的一个扩展CLASSIT可以用于连续数据的聚类, 但也有同样的问题
- AutoClass (Cheeseman and Stutz, 1996)
  - 使用Bayesian 统计分析估计聚类的个数
  - 在产业界很流行

神经网络方法

- 神经网络方法将每个簇描述为一个标本(exemplar)
- 标本作为聚类的“原型”, 不一定对应一个特定的数据实例或对象
- 根据某些距离函数, 新的对象可以被分配给标与其本最相似的簇. 被分配给一个簇的对象的属性可以根据该簇的标本的属性来预测
- 两个比较著名的方法
  - 竞争学习(competitive learning)
  - 自组织特征映射(Self-organizing feature maps)

这两种方法都涉及有竞争的神经单元

神经网络方法

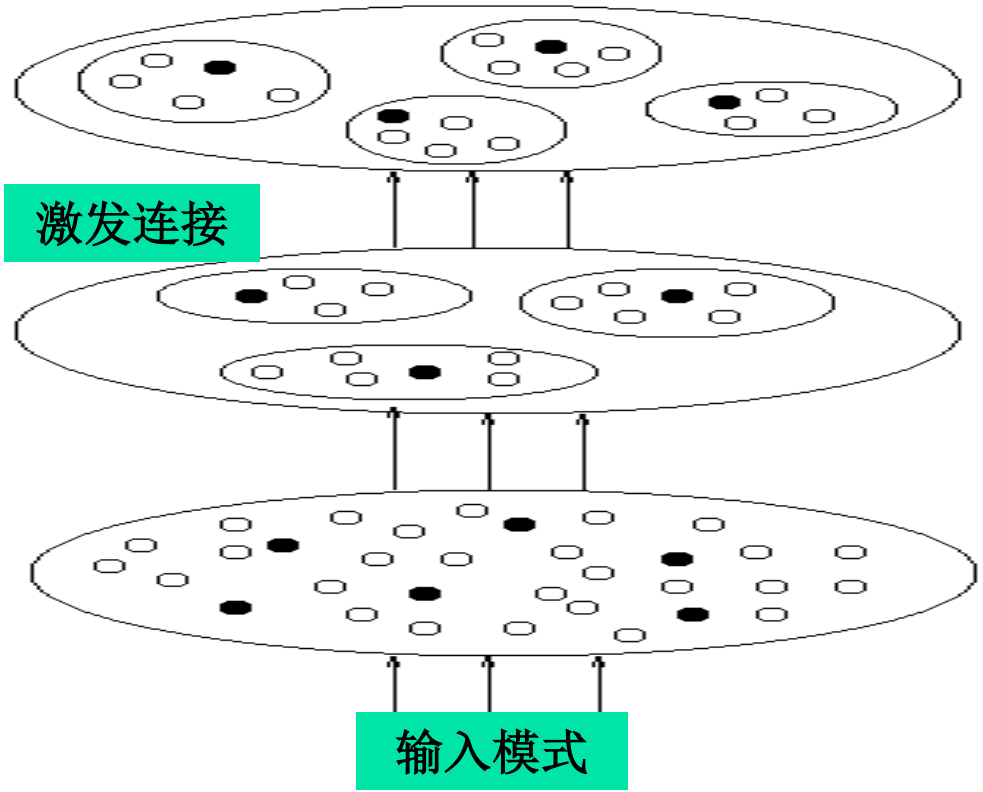
- 竞争学习
  - 涉及若干个单元(“神经元” neurons)的层次结构
  - 它们以一种“胜者全取(winner-take-all)”的方式对系统当前处理的对象进行竞争
  - 下图显示了一个竞争学习系统的例子
    - 每个圆圈代表一个单元, 在一个簇中获胜的单元成为活跃的(以实心圆点表示), 而其它是不活跃的(以空心圆点表示)
    - 各层之间的连接是激发(excitatory)——在某个给定层次中的单元可以接收来自低一层次所有单元的输入
    - 在某个给定层次中, 一个簇中的单元彼此竞争, 对低一

竞争学习结构

层3
抑制簇

层2
抑制簇

层1
输入单元



自组织特征映射

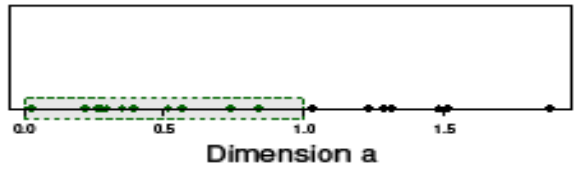
- 自组织特征映射 (Self-organizing feature maps,SOM)
  - 也是通过若干个单元竞争当前对象来进行聚类
  - 权重向量最接近当前对象的单元成为获胜的或活跃的单元
  - 为了更接近输入对象, 获胜单元及其最近的邻居的权重进行调整
  - SOM被认为类似于大脑的处理过程
  - 对在二或三维空间中直观化高维数据是有用的

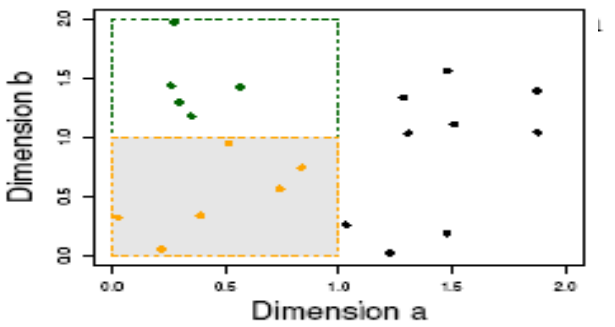
聚类高维数据

- 聚类高维数据，应用广泛: text documents, DNA micro-array data；重要挑战:
  - 多个不相关的维度掩盖聚类
  - 距离函数变得没有意义—由于 equi-distance(高维空间，数据变稀疏)
  - 聚类可能存在于某些子空间中
- 特征变换: 仅当大部分维度与聚类相关时有效
  - PCA & SVD有效，当特征高度相关/冗余
- 特征选择: 缠绕wrapper 或 过滤方法

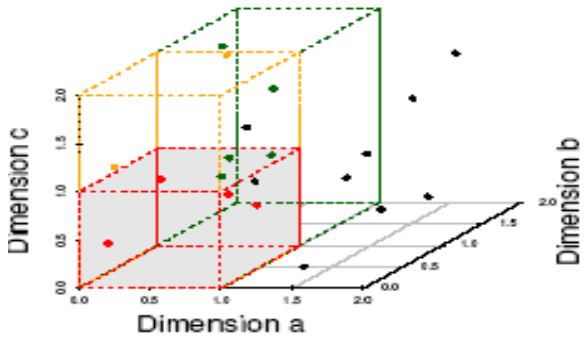
维数灾难

- (graphs adapted from Parsons et al. KDD Explorations 2004)
  - 一维的数据相对压缩的
  - 增加一个维度将沿此维“伸展” 数据点，使得数据更分散
  - 增加更多的维度将使得数据更稀疏—高维数据非常稀疏
  - 距离变得没有意义—**due to equi-distance**



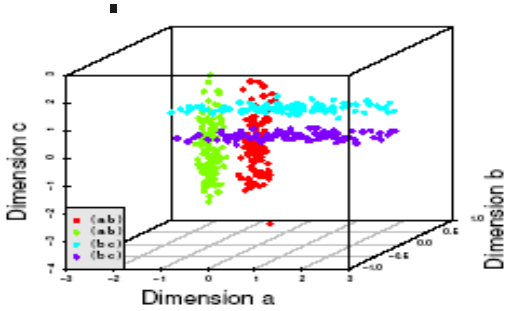


(b) 6 Objects in One Unit Bin

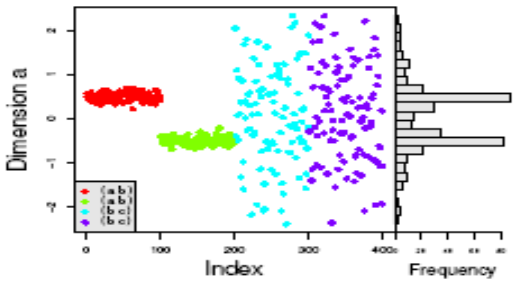


(c) 4 Objects in One Unit Bin

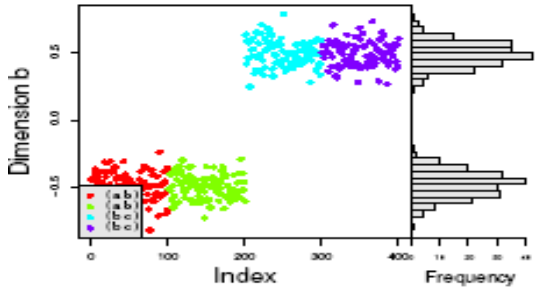
子空间聚类，为什么？



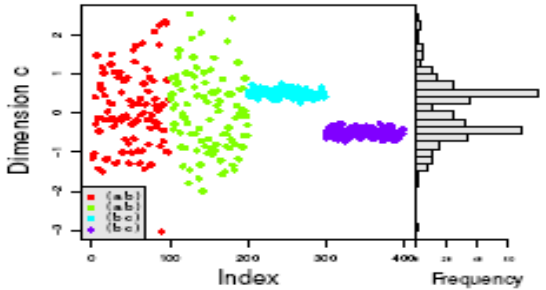
- 聚类可能只存在于某些子空间
- 子空间聚类: **find clusters in all the subspaces**



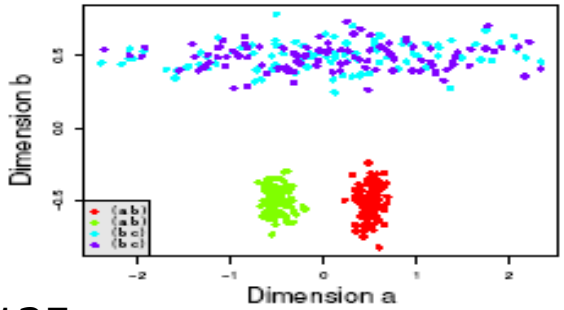
(a) Dimension a



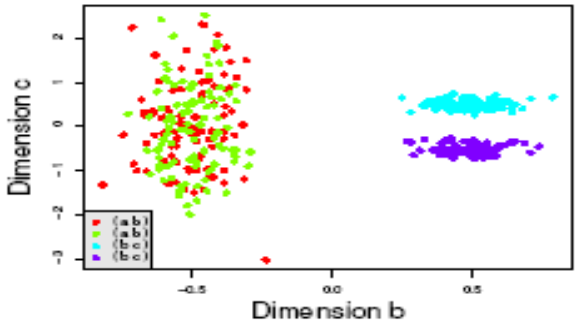
(b) Dimension b



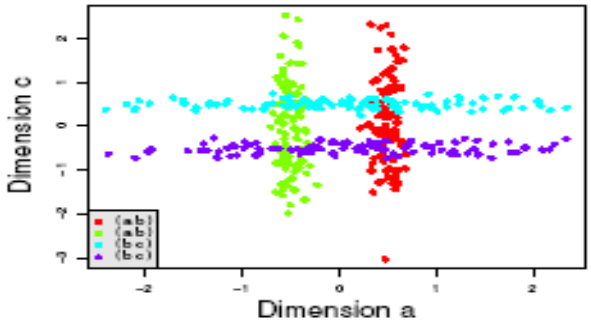
(c) Dimension c



(a) Dims a & b



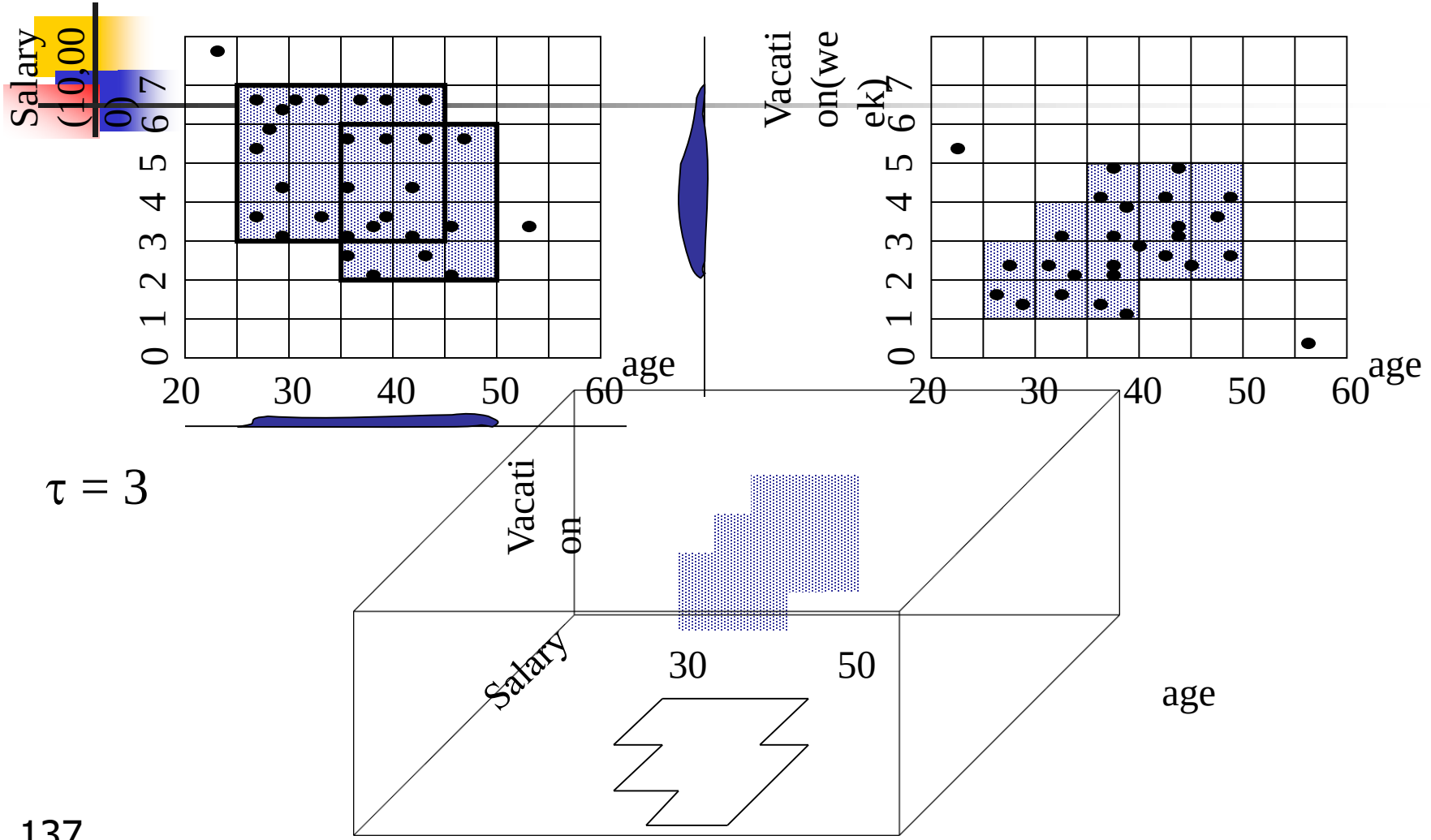
(b) Dims b & c



(c) Dims a & c

CLIQUE

- CLIQUE(Clustering In QUEst)综合了基于密度和基于网格的聚类方法. 由Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)提出, 对于大型数据库中的高维数据的聚类非常有效 .
- 自动识别高维数据空间的子空间, 在子空间上聚类比在原空间上更好
- 基本思想: CLIQUE是基于密度和基于网格的聚类方法
  - 它将每个维划分成相同个数的等长区间
  - 它将m-维数据空间划分成不重叠的长方形单元
  - 一个单元是稠密的, 如果包含在该单元中的数据点占全部数据点的比例超过输入的模型参数


Salary (10,000)
0 1 2 3 4 5 6 7
20 30 40 50 60
age
Vacation(week)
0 1 2 3 4 5 6 7
20 30 40 50 60
age
τ = 3
Vacation
Salary
30 50
age

CLIQUE

- CLIQUE所采用的先验性质(Apriori property)如下
  - 如果一个k维单元是密集的, 那么它在k-1维空间上的投影也是密集的
  - 也就是说, 给定一个k维的候选密集单元, 如果我们检查它的k-1维投影单元, 发现任何一个不是密集的, 那么我们知道第k维的单元也不可能是密集的
  - 可以从k-1维空间中发现的密集单元来推断k维空间中潜在的或候选的密集单元. 通常, 最终的结果空间比初始空间要小很多

CLIQUE：主要步骤

- 划分数据空间,并找出划分的每个单元中 的点数.
- 使用Apriori性质, 识别包含聚类的子空间
- 识别聚类：
  - 确定所有感兴趣的子空间中的稠密单元
  - 确定所有感兴趣的子空间中的连接的稠密单元.
- 产生聚类的最小描述
  - 对每个簇, 确定覆盖相连的密集单元的最大区域
  - 然后确定最小的覆盖

CLIQUE的优缺点

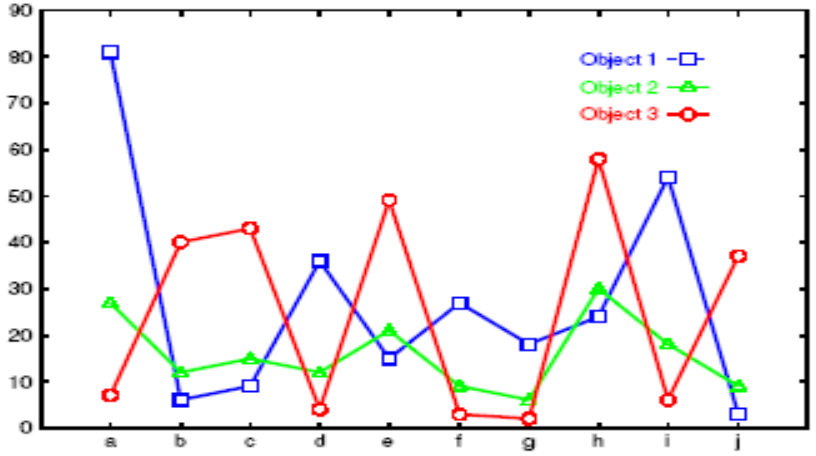
- 优点
  - 它自动地找出高维的子空间, 高密度的聚类存在于在这些子空间中
  - 对元组的输入顺序不敏感, 无需假设任何规范的数据分布
  - 它随输入数据的大小线性地扩展, 当数据的维数增加时具有良好的可扩展性
- 缺点
  - 由于方法大大简化, 聚类结果的精确性可能会降低

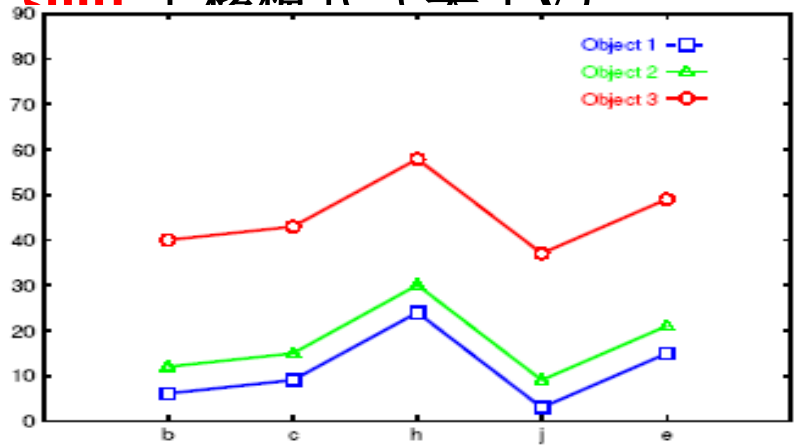
基于频繁模式的方法

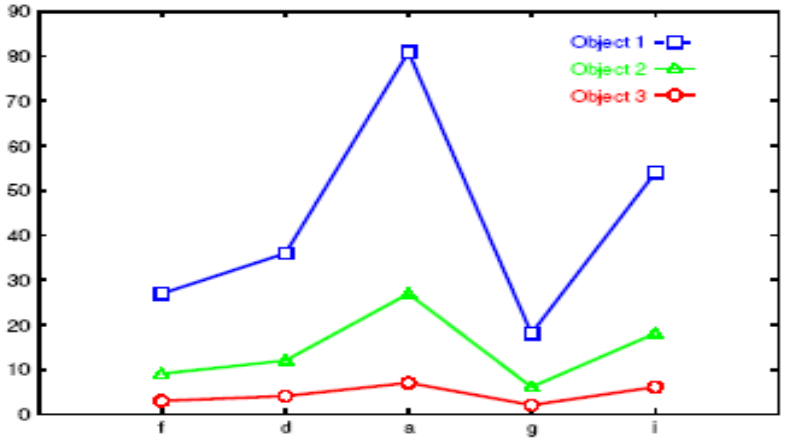
- 基本思想：发现的频繁模式也可能预示簇
- 典型的方法
  - 基于频繁相的文本聚类
    - 提取相，每个文档用项的集合表示
      - 项由单个或多个词组成
    - 挖掘频繁项集
    - 从频繁项集的集合中精选出的子集可以看成聚类
      - 精选子集覆盖所有的文档
      - 不同子集覆盖的部分间的重叠小

基于模式相似性聚类

- 原始的微阵列数据包含3个基因在多个条件（维度）下的表达值
  - 难以发现规律
- 特定的维度子集形成了有趣的模式
  - shift 平移模式（关于y）
  -







Why *p*-Clustering?

- 微阵列数据分析需要
  - 针对数千个attributes 聚类
  - 发现 shift 和 scaling 的模式
- 用Euclidean 距离来聚类 — 不能发现平移模式
- $H(IJ) = \frac{1}{|I||J|}\Sigma_{i\in I,j\in J}(d_{ij} - d_{iJ} - d_{Ij} + d_{IJ})^2$引入N(N-1) 维

$$d_{ij} = \frac{1}{|J|}\sum_{j\in J} d_{ij}$$

$$d_{Ij} = \frac{1}{|I|}\sum_{i\in I} d_{ij}$$

$$d_{IJ} = \frac{1}{|I||J|}\sum_{i\in I, j\in J} d_{ij}$$

- 2）提出的Bi-cluster使用均方残差得分h(I, J)
  - 子矩阵是δ-cluster，如果H(I, J) ≤ δ 对某个δ > 0
  - 应用随机算法发现簇

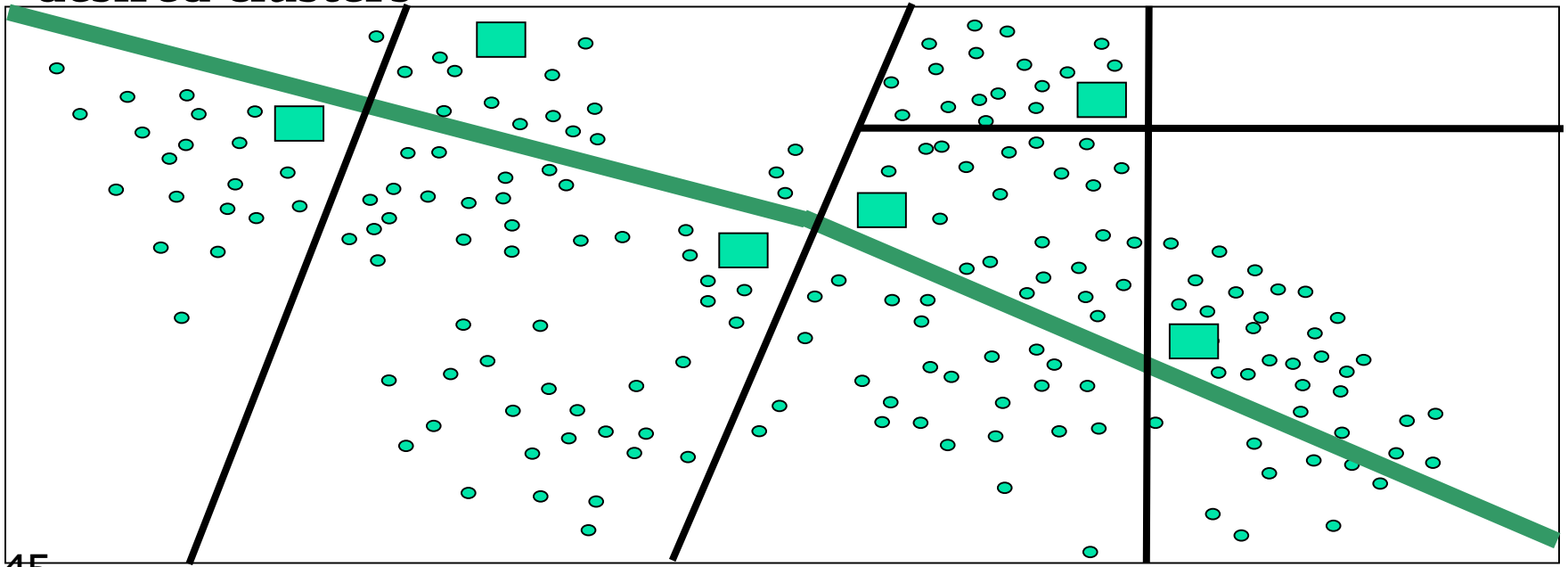
(*p*-Clustering)

$$pScore\left(\begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix}\right) = |(d_{xa} - d_{xb}) - (d_{ya} - d_{yb})|$$

- 给定集合O中对象x, y 和T中特征a, b, pScore是基于2x2 matrix
- (O, T)是δ-pCluster，如果(O, T)的任何2x2 矩阵X,pScore(X) ≤ δ(δ>0)
- δ-pCluster的特点
  - 向下闭包特性
  - 得到的簇更同源，相比于bi-cluster (要求两两组合满足条件) $\frac{d_{xa}/d_{ya}}{d_{xb}/d_{yb}} \delta$
- 已经提出模式增长的算法来挖据

基于约束的聚类分析?

- 需要用户反馈: 用户对应用需求有更清楚的认识
- 更少参数但更多的用户约束, e.g., ATM 分配问题: obstacle & desired clusters

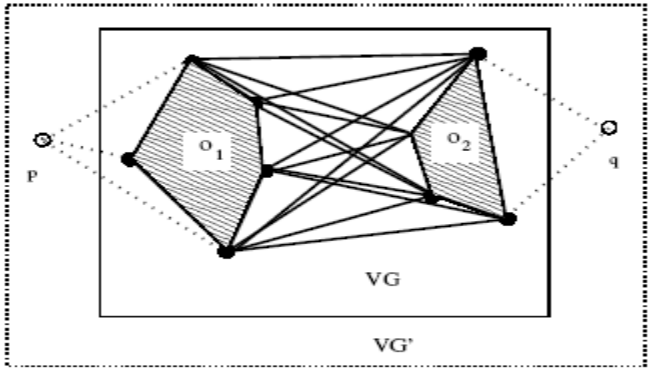


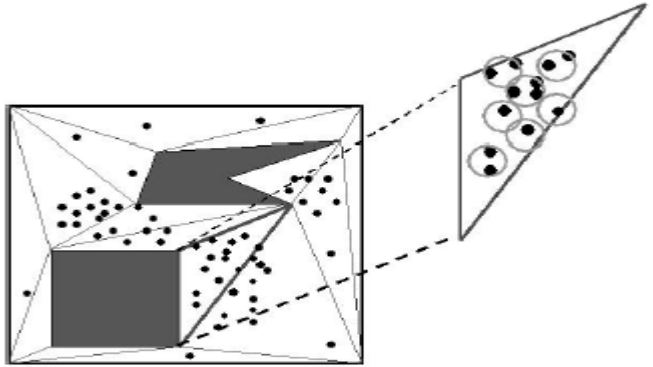
约束聚类的分类

- 根据约束的种类:
  - 个体对象的约束 (对数据对象进行选择)
    - 房地产应用：价值$300K以上的房子聚类
  - 聚类参数的约束
    - # of clusters, MinPts, etc.
  - 距离或相似度函数的约束
    - Weighted functions, obstacles (e.g., rivers, lakes)
  - 用户指定的约束
    - 服务站设置：每个簇包含至少500高价值客户和5000普通客户
  - 半监督聚类:

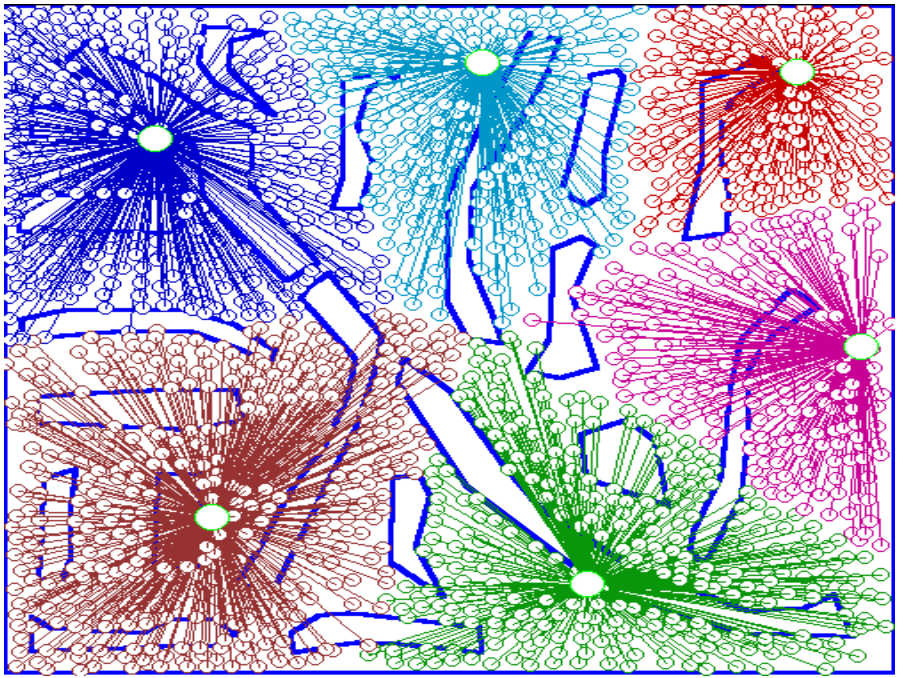
含障碍对象的聚类

- K-中心点方法更合适，k-means某个质心（分配ATM ）可能是湖中心
- 构造可见图（最短路径 ）
- 三角划分，形成微簇
- 两类连接索引 (基于最短路径）
  - VV index: 任意一对障碍物定点
  - MV index: 任意一对微簇和障碍物定点

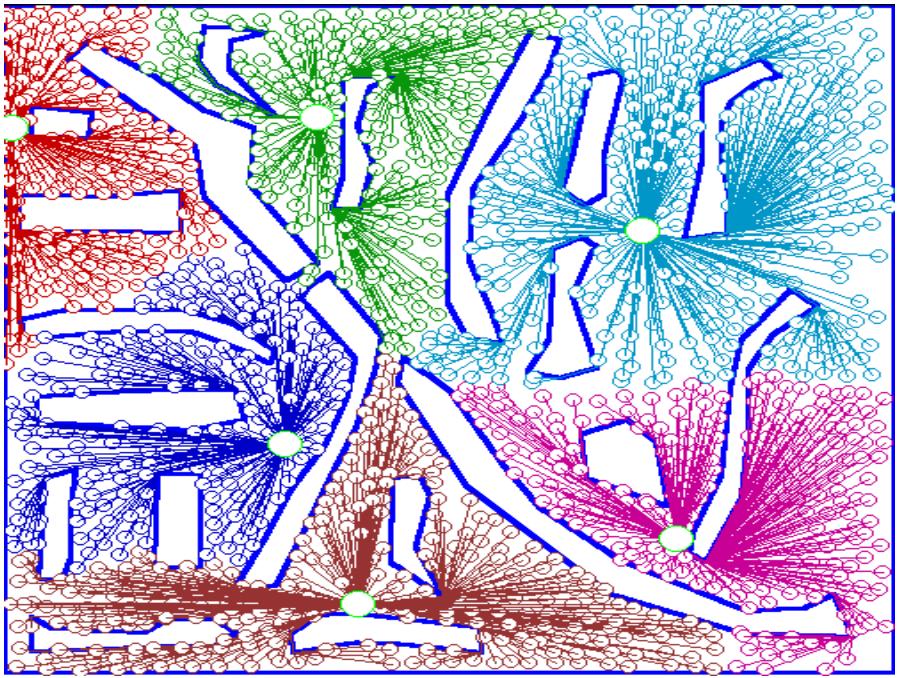




An Example: Clustering With Obstacle Objects



Not Taking obstacles into account



Taking obstacles into account

用户约束的聚类

- 例: 确定k个投递中心
- 方法
  - 划分数据k组，寻找一个满足约束的初始解
  - 迭代地调整微簇来改进结果 (e.g., 移动δ μ-clusters 从C_i到C_j)
  - 处理“死锁deadlock” (需要时分裂微簇)
- 对数据预处理形成微簇，可以提高效率

第9章. 聚类分析

- 什么是聚类（Clustering）分析?
- 聚类分析中的数据类型
- 主要聚类方法分类
- 划分方法（Partitioning Methods）
- 层次方法（Hierarchical Methods）
- 基于密度的方法（Density-Based Methods）
- 基于网格的方法（Grid-Based Methods）
- 基于模型的聚类方法（Model-Based Clustering Methods）
- 孤立点分析（Outlier Analysis）

孤立点分析

- 什么是孤立点?
  - 对象的集合, 它们与数据的其它部分不一致
  - 孤立点可能是度量或执行错误所导致的
  - 孤立点也可能是固有的数据变异性的结果
- 问题
  - 给定一个n个数据点或对象的集合, 及预期的孤立点的数目k, 发现与剩余的数据相比是相异的, 例外的, 或不一致的前k个对象
- 两个子问题:
  - 定义在给定的数据集合中什么样的数据可以被认为是不一致的

孤立点分析

- 应用:
  - 信用卡欺诈检测
  - 电信欺诈检测
  - 顾客分割: 确定极低或极高收入的客户的消费行为
  - 医疗分析: 发现对多种治疗方式的不寻常的反应
- 孤立点的定义是非平凡的
  - 如果采用一个回归模型, 余量的分析可以给出对数据“极端”的很好的估计
  - 当在时间序列数据中寻找孤立点时, 它们可能隐藏在趋势的, 周期性的, 或者其他循环变化中, 这项任务非常棘手

孤立点分析

- 采用数据可视化方法来进行孤立点探测如何？
  - 不适用于包含周期性曲线的数据
  - 对于探测有很多分类属性的数据，或高维数据中的孤立点效率很低
- 方法
  - 统计学方法
  - 基于距离的方法
  - 基于偏差的方法
  - 基于密度的方法

基于统计学的孤立点检测

- 对给定的数据集合假设了一个分布或概率模型(例如, 正态分布), 然后根据模型采用不一致性检验(discordancy test)来确定孤立点
- 检验要求的参数
  - 数据集参数: 例如, 假设的数据分布
  - 分布参数: 例如平均值和方差
  - 和预期的孤立点的数目
- 统计学的不一致性检验需要检查的两个假设
  - 工作假设(working hypothesis)
  - 替代假设(alternative hypothesis)

基于统计学的孤立点检测

- 工作假设H是一个命题：n个对象的整个数据集合来自一个初始的分布模型F,
  - 即 H：$O_i \in F$，$i = 1, 2, \ldots, n$
- 不一致性检验验证一个对象O_i关于分布F是否显著地大(或者小)
- 依据关于数据的可用知识, 已提出不同的统计量用于不一致性检验
- 假设某个统计量被选择用于不一致性检验, 对象O_i的该统计量的值为V_i, 则构建分布T
  - 估算显著性概率$SP(V_i)=Prob(T>V_i)$

基于统计学的孤立点检测

- 结果非常依赖于模型*F*的选择
  - O_i可能在一个模型下是孤立点, 在另一个模型下是非常有效的值
- 替代分布在决定检验的能力上是非常重要的
- 不同的替代分布
  - 固有的替代分布(inherent alternative distribution):所有对象来自分布*F*的工作假设被拒绝, 而所有对象来自另一个分布*G*的替代假设被接受
  - 混合替代分布(mixture alternative distribution)：不一致的值不是*F*分布中的孤立点, 而是来自其他分布的污染物

基于统计学的孤立点检测

- 检测孤立点有两类基本的过程
  - 批(block)过程：或者所有被怀疑的对象都被作为孤立点对待, 或者都被作为一致数据而接受
  - 连续的过程：

    该过程的一个例子是内部出局(inside-out)过程
    - 主要思想
      - 首先检验最不可能是孤立点的对象. 如果它是孤立点, 那么所有更极端的值都被认为是孤立点;否则, 检验下一个极端的对象, 依次类推
    - 该过程往往比批过程更为有效

基于统计学的孤立点检测

- 缺点
  - 绝大多数检验是针对单个属性的, 而许多数据挖掘问题要求在多维空间中发现孤立点
  - 统计学方法要求关于数据集合参数的知识(如, 数据分布), 但是在许多情况下, 数据分布可能是未知的
  - 当没有特定的检验时, 统计学方法不能确保所有的孤立点被发现; 或者观察到的分布不能恰当地被任何标准的分布来模拟

基于距离的孤立点检测

- 为了解决统计学方法带来的一些限制，引入了基于距离的孤立点的概念
- 基于距离的孤立点:
  - ***DB(p, d)*-孤立点是数据集*T*中的一个对象*o*, 使得 *T*中的对象至少有*p*部分与*o*的距离大于*d***
- 将基于距离的孤立点看作是那些没有“足够多”邻居的对象. 这里的邻居是基于距给定对象的距离来定义的
- 对许多不一致性检验来说, 如果一个对象 *o*根据给定的检验是一个孤立点, 那么对恰当定义的*p*和*d*，*o*也是一个*DB(p,d)*孤立点

基于距离的孤立点挖掘算法

- 基于索引的算法
  - 采用多维索引结构, *R*树或*k-d*树, 来查找每个对象*o*在半径*d*范围内的邻居
  - 设*M*是一个孤立点的d-邻域内的最大对象数目. 一旦对象*o*的*M*+1个邻居被发现, *o*就不是 孤立点
  - 最坏情况下的复杂度为$O(kn^2)$, 这里*k*是维数, *n*是数据集合中对象的数目
  - 建造索引的任务是计算密集的

- 嵌套循环算法
  - 嵌套-循环算法和基于索引的算法有相同的计算复杂度, 但它避免了索引结构的构建, 试图最小化I/O的次数
  - 它把内存的缓冲空间分为两半, 把数据集合分为若干个逻

基于距离的孤立点挖掘算法

- 基于单元(cell-based)的算法
  - 为了避免O(n^2)的计算复杂度，为驻留内存的数据集合开发了基于单元的算法. 它的复杂度是O(c^k+n)，这里c是依赖于单元数目的常数, k是维数
- 方法
  - 数据空间被划分为单元, 单元的边长等于 $d/2\sqrt{k}$
  - 每个单元有两层围绕着. 第一层的厚度是一个单元, 而第二层的厚度是 $[2\sqrt{k}-1]$
  - 算法逐个单元地对孤立点计算，而不是逐个对象地进行计算. 对一个给定的单元, 它累计三个计数——单元中对象的数目cell_count , 单元和第一层中对象的数目

基于距离的孤立点挖掘算法

- 确定孤立点

  设M是一个孤立点的d-邻域中可能存在的孤立点的最大数目

  - 如果cell_+_1_layer_count大于M, 那么该单元中所有的对象可以从进一步的考察中移走, 因为它们不可能是孤立点
  - 如果cell_+_2_layers_count 小于或等于M，那么单元中所有的对象被认为是孤立点
  - 否则, 对单元中的每个对象*o*, 检查*o*的第二层中的对象. 只有那些*d*-邻域内的对象不超过M个的点是孤立点

基于偏离的孤立点检测

- 通过检查一组对象的主要特征来确定孤立点
- 与给出的描述偏离的对象被认为是孤立点
- 序列异常技术(sequential exception technique)
  - 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
- 术语
  - 异常集(exception set): 它是偏离或孤立点的集合, 被定义为某类对象的最小子集, 这些对象的去除会导致剩余集合的相异度的最大减少
  - 相异度函数(dissimilarity function)：是满足如下条件的任意函数：当给定一组对象时，如果对象间相似，返值

基于偏离的孤立点检测

例: 给定n个对象的子集合$\{x_1,\ldots,x_n\}$, 一个可能的相异度函数是集合中对象的方差

- 基数函数(cardinality function):
  - 一般是给定的集合中对象的数目
- 平滑因子(smoothing factor):
  - 一个为序列中的每个子集计算的函数.
  - 它估算从原始的数据集合中移走子集合可以带来的相异度的降低程度.
  - 平滑因子值最大的子集是异常集

基于偏离的孤立点检测

- 一个顺序的方法在计算上是可行的，能够用一个线性的算法实现
  - 不考虑估算当前子集关于其补集的相异度，该算法从集合中选择了一个子集合的序列来分析
  - 对每个子集合，它确定其与序列中前一个子集合的相异度差异
  - 为了减轻输入顺序对结果的任何可能的影响，以上的处理过程可以被重复若干次，每一次采用子集合的一个不同的随机顺序
  - 在所有的迭代中有最大平滑因子值的子集合成为异常集

基于偏离的孤立点检测

- **OLAP 数据方技术**
  - 使用数据方识别大型多维数据中的异常区域

Summary

- **Cluster analysis groups objects based on their similarity and has wide applications**
- **Measure of similarity can be computed for various types of data**
- **Clustering algorithms can be categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods**
- **Outlier detection and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches**

References (1)

- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD'98
- M. R. Anderberg. Cluster Analysis for Applications. Academic Press, 1973.
- M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure, SIGMOD'99.
- P. Arabie, L. J. Hubert, and G. De Soete. Clustering and Classification. World Scietific, 1996
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. KDD'96.
- M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. SSD'95.
- D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139-172, 1987.
- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based

References (2)

- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. VLDB'98.
- G. J. McLachlan and K.E. Bkasford. Mixture Models: Inference and Applications to Clustering. John Wiley and Sons, 1988.
- P. Michaud. Clustering techniques. Future Generation Computer systems, 13, 1997.
- R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. VLDB'94.
- E. Schikuta. Grid clustering: An efficient hierarchical clustering method for very large data sets. Proc. 1996 Int. Conf. on Pattern Recognition, 101-105.
- G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. VLDB'98.