中国科学技术大学
University of Science and Technology of China

数据挖掘与数据仓库

第九讲 聚类

November 15, 2017

ustc

Outline

- 概述
- 划分方法
- 层次方法
- 基于密度的方法
- 基于网格的方法
- 聚类评估
- 其它问题

问题提出

- 给定数据集DS，没有任何元组的类标签已知，要求给所有元组添加类标签。具体要求：
  - 同类元组相似，类间不相似。相似即距离
- 概念
  - 同类元组构成 cluster/簇
  - 所有簇的集合成为聚类
- 数据分割/无监督学习

问题提出

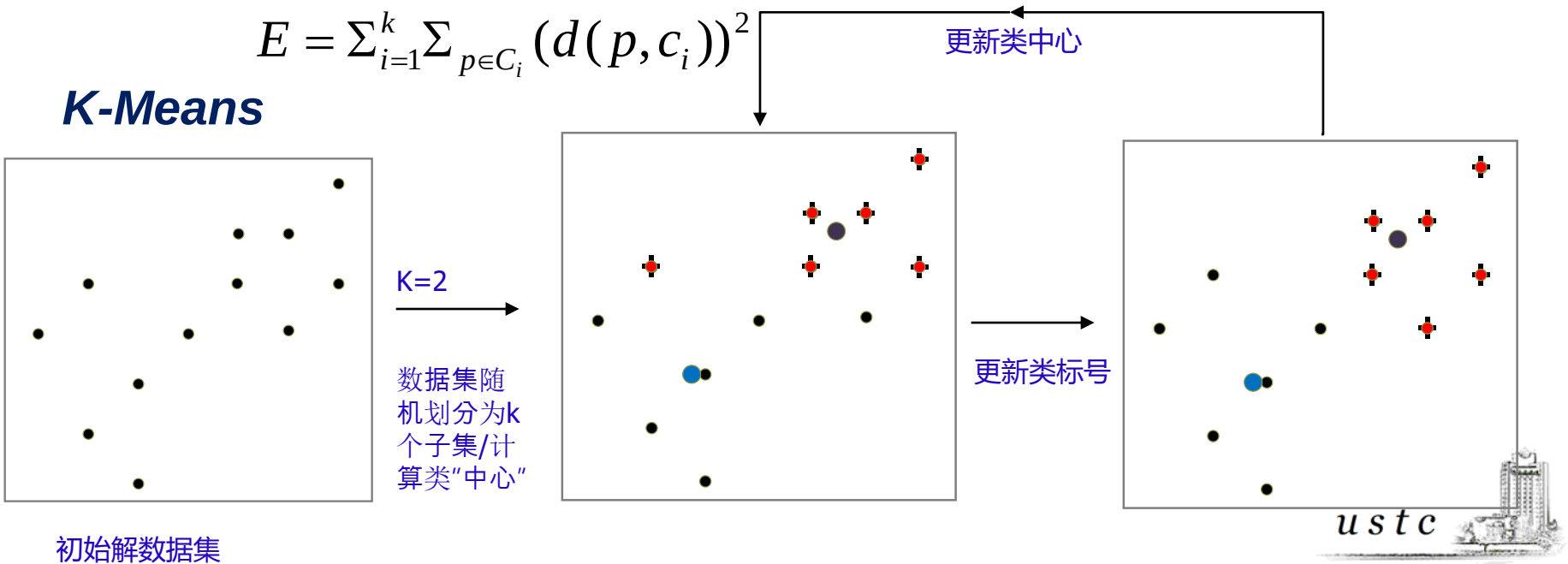
- 给定数据集DS，没有任何元组的类标签已知，要求给所有元组添加类标签。具体要求：
  - 同类元组相似，类间不相似。相似即距离
- 概念
  - 同类元组构成 cluster/簇
  - 所有簇的集合成为聚类
- 数据分割/无监督学习

划分方法

划分方法

- 同类元组相似，类间不相似。相似即距离小
- 思想：每个元组到所属类的中心的距离和最小；每个类用聚类中心来“代表”
  - 问题：类的个数k=?距离如何定义？中心是什么？

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$$



划分方法 评述

- 划分方法的全局最优性？
  - 穷举所有的元组划分方案
- 可行的变型：启发式方法来划分
  - K-means/K-均值：cluster中心是由每个属性的均值构成，虚拟的“中心/重心”代表整个cluster
  - K-medoids/K-中心点：cluster中心为某个元组，该元组最靠近簇的中心，由它代表整个cluster。

K-中心算法需要先利用K-均值算法求出虚拟中心，然后在cluster的点中找到与虚拟中心最近的点作为该cluster的中心

K-means 评述

- 最优性？
  - 常陷入局部最优
- 时间复杂度
  - O(元组数 * cluster数 * 迭代次数)
- 应用限制：
  - 连续型n-D空间，标称属性可以中位数代替
  - 需要事先指定k的值
  - 对噪声和离群点敏感 K-中心点对噪声和离群点不敏感，鲁棒性强
  - 非凸形状的cluster没办法发现
  - 簇大小比较接近

K-means 变型

- 从三个角度考虑改进/变型
  - 初始时刻k个中心的选择
  - 距离计算方法
  - 计算簇中心的策略
- 标称数据处理
  - 均值用众数代替
  - 改进距离计算方法用于标称数据
  - 用基于频率的方法更新簇的众数
  - 标称数据和数值数据混合：k-prototype

K-medoids/K中心点

- 一定程度上降低离群点的影响！
  - 寻找实际元组为簇代表，NP-Hard问题
  - 实践方法：PAM，比K-means费时
- PAM：围绕中心点划分方法
  - 随机选择k个种子/簇代表
  - 交换一个种子和非种子，检查是否能提高聚类质量：所有元组到各自代表/种子的距离和
  - 启发式方法选择交换的种子和非种子
  - 易陷入局部最优

大型数据集的划分

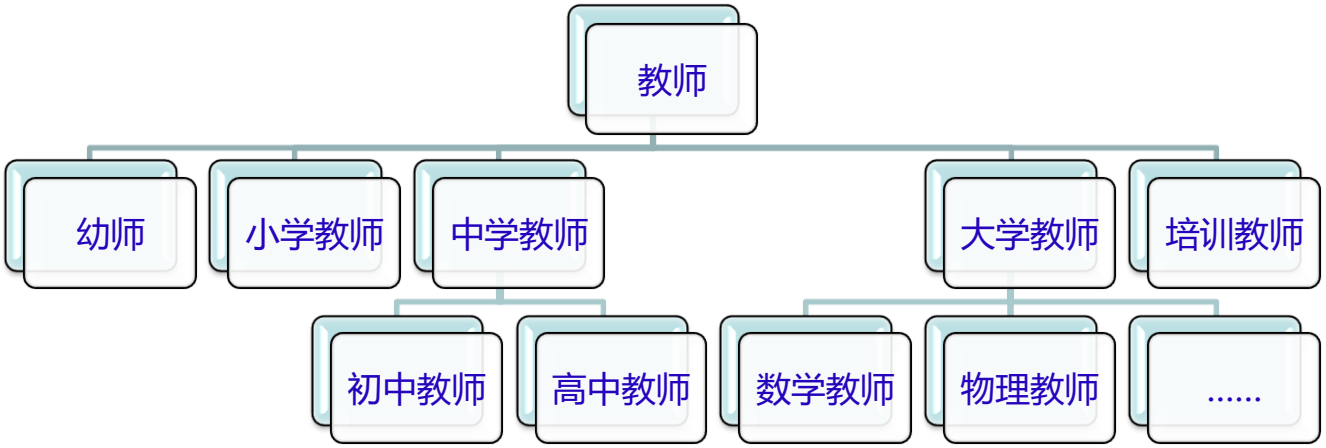
- 计算k个代表和数据集中所有元组的距离
  - 聚类的基本操作，重复进行多次O(nkm)
  - 无法一次将数据集载入内存？
- 划分方法的可扩展性　数据集太多，在数据集的子集上操作
  - 对数据集中的元组进行采样
  - 在数据集采样得到的子集上找中心点: CLARA
  - 除了在采样子集上进行计算，中心点可以在数据集全部元组中寻找：CLARANS

中心点在整个数据集上找，在子集上计算距离

层次方法

层次方法

- 很多数据对象，语义蕴含着“层次”概念
  - 在数据蕴含的层次概念不明时，能否自动构建？
  - 数据在不同层次上进行汇总？
  - 层次聚类：在不同层次上各自聚类，形成一棵树/ Dendrogram



层次方法：基本思想

凝聚

- 自底向上
- 每个数据对象是一个簇，然后合并最“相似”的簇，形成更大的簇

类似于构建Huffman树，树中的一层就是一个层次

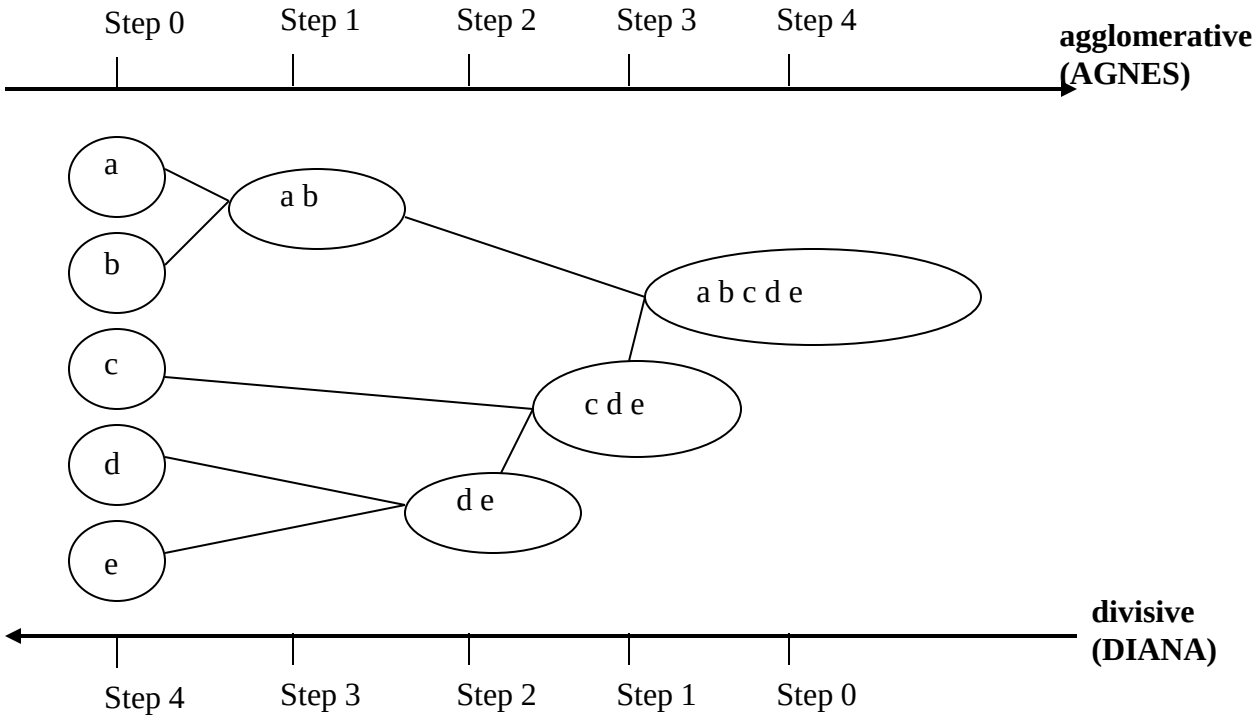
可以通过指定cluster/簇的目标数目k来停止

分裂

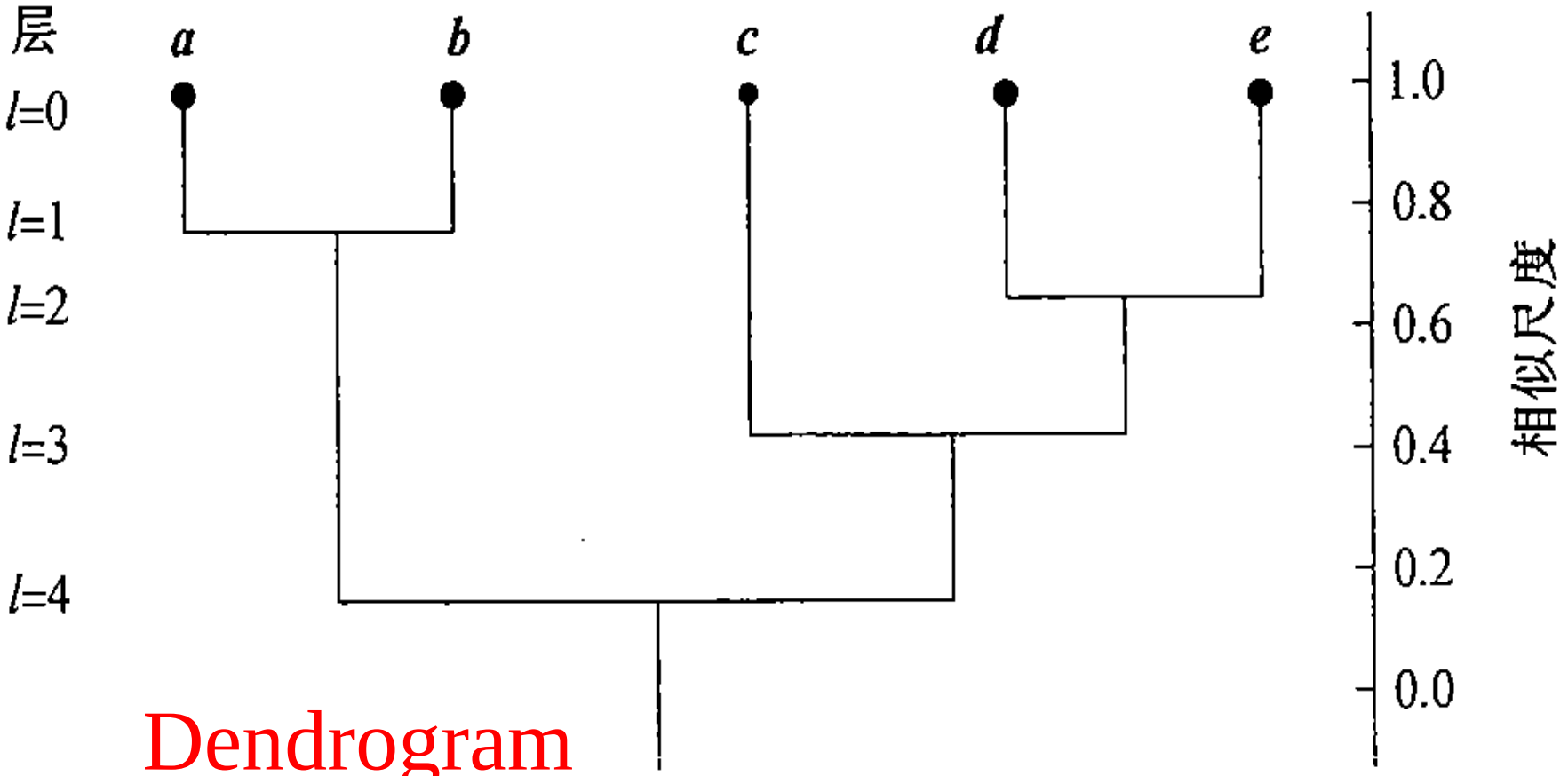
- 自顶向下
- 初始所有数据对象在一个簇中，把不满足“凝聚”条件的簇分裂成更小的，具有更好凝聚力的小簇
- 指数种分裂方式

层次方法：例子

- 自底向上：AGNES，相似性=两个簇中最相似两个数据的距离
- 自顶向下：DIANA，两个簇中最近两个数据的最大距离



层次方法：例子

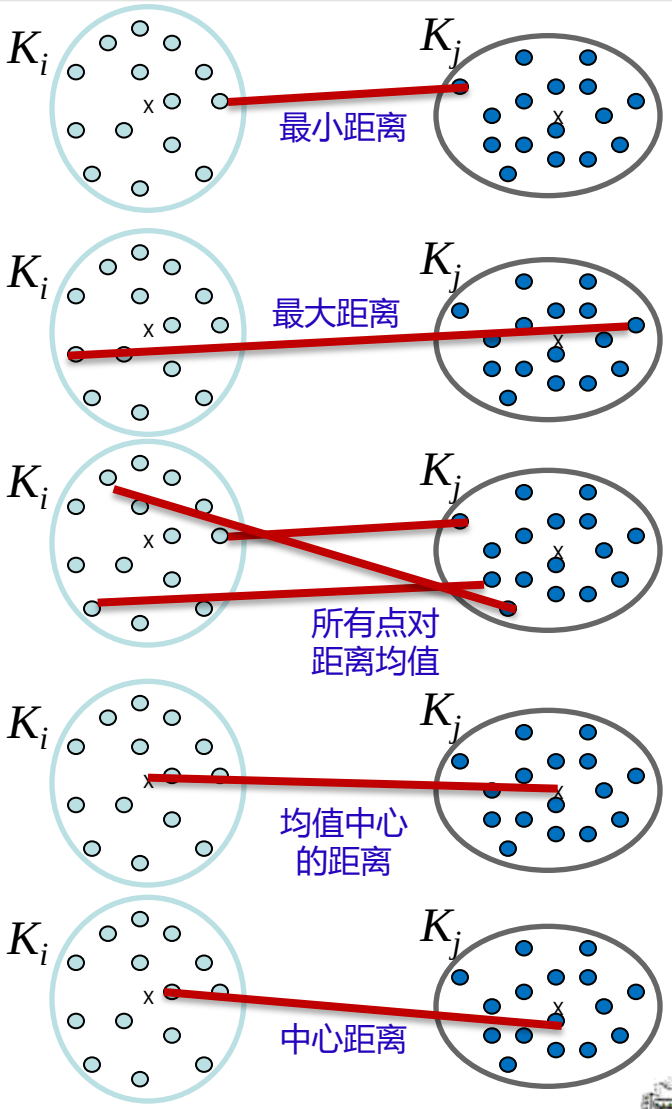


Dendrogram

层次方法：核心问题

度量两个簇的距离

- **最小距离**：两个簇之间最相似两个数据之间的距离
- **最大距离**：两个簇之间最不相似两个数据之距离
- **平均距离**：簇之间所有点对距离之均值
- **虚拟中心距离**：两个簇的虚拟“中心”代表之距离
- **Medoid距离**：两个簇的“中心”代表之距离



簇特征/CF

- 随机变量的矩/moments：描述随机变量的数字特征
  - k阶原点矩：$M_k = \sum_{i=1}^{N} X_i^k$, k=0,1,2
- 定义一个数据集的3-D簇特征 $CF = (M_0, M_1, M_2)$

$$M_0 = \sum_{i=1}^{N} X_i^0 = N$$

$$M_1 = \sum_{i=1}^{N} X_i$$

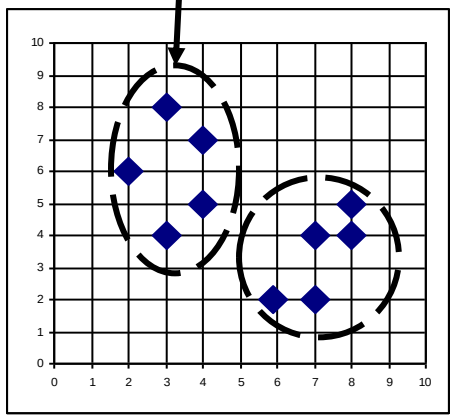
$$M_2 = \sum_{i=1}^{N} X_i^2$$

CF: 数据集/簇的汇总统计信息

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

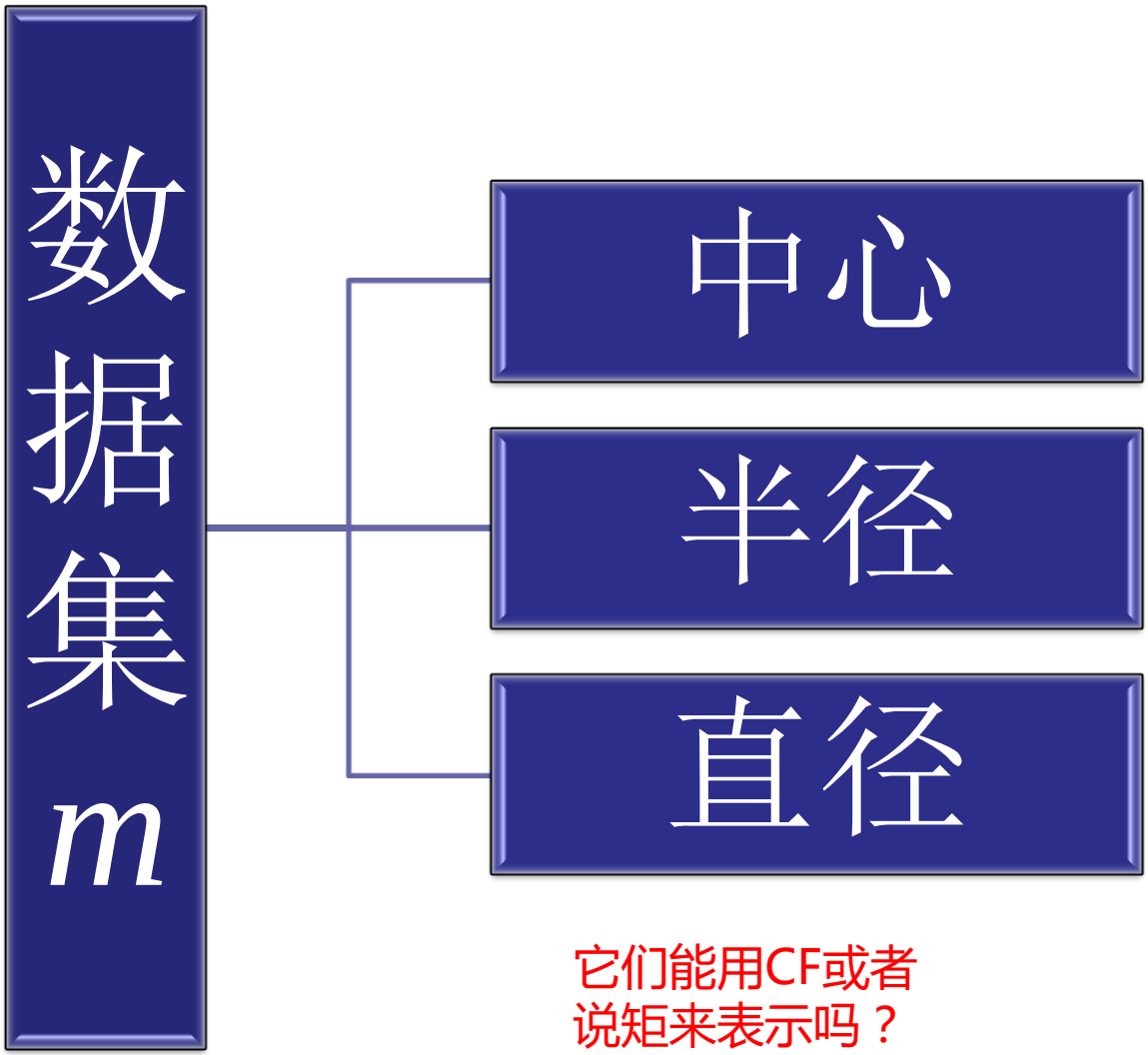
3+2+4+4+3=16

3^2+2^2+4^2+4^2+3^2=54



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

数据集的几个特征



$$c_m = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$R_m = \sqrt{\frac{\sum_{i=1}^{N} (x_i - c_m)^2}{N}}$$

$$D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (x_i - x_j)^2}{N(N-1)}}$$

它们能用CF或者
说矩来表示吗？

两个簇及其并集的CF

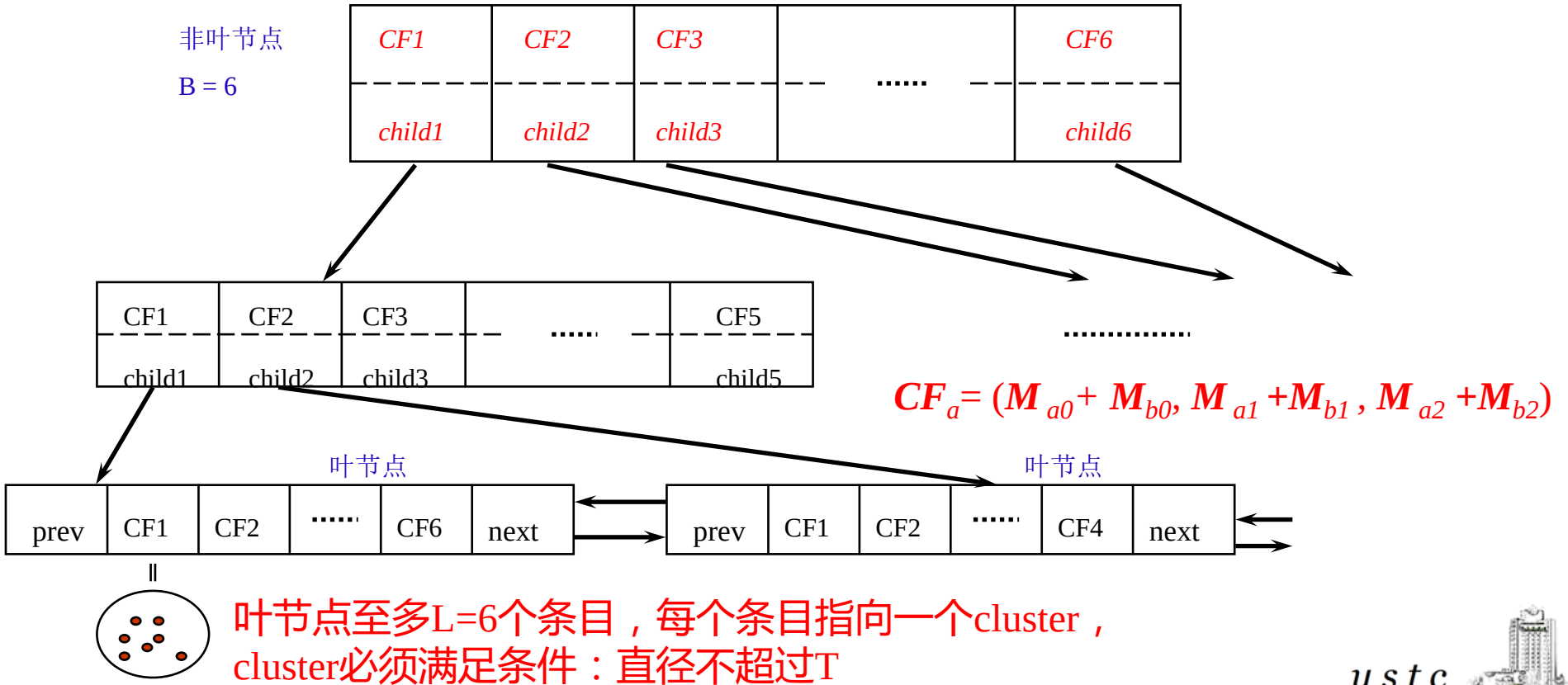
- 设有两个簇a和b：
  - $CF_a = (M_{a0}, M_{a1}, M_{a2})$
  - $CF_b = (M_{b0}, M_{b1}, M_{b2})$
- 这两个簇合并得到的新簇，其
  - $CF_a = (M_{a0} + M_{b0}, M_{a1} + M_{b1}, M_{a2} + M_{b2})$
- 请证之。
- 优点：CF计算简单快捷

层次方法：可扩展性

- 数据集非常大，无法一次载入内存，如何处理？
- 时间复杂度？每一层的时间代价？

BIRCH：思想

- 非叶节点是长度为B_i的结构体数组，一个结构体包括{一个CF向量，一个指针指向一个孩子节点}
  - B:子树中结构体数组长度的最大值，称为分支因子
  - T: 子树相关叶节点的直径最大值



$$CF_a = (M_{a0} + M_{b0}, M_{a1} + M_{b1}, M_{a2} + M_{b2})$$

叶节点至多L=6个条目，每个条目指向一个cluster，cluster必须满足条件：直径不超过T

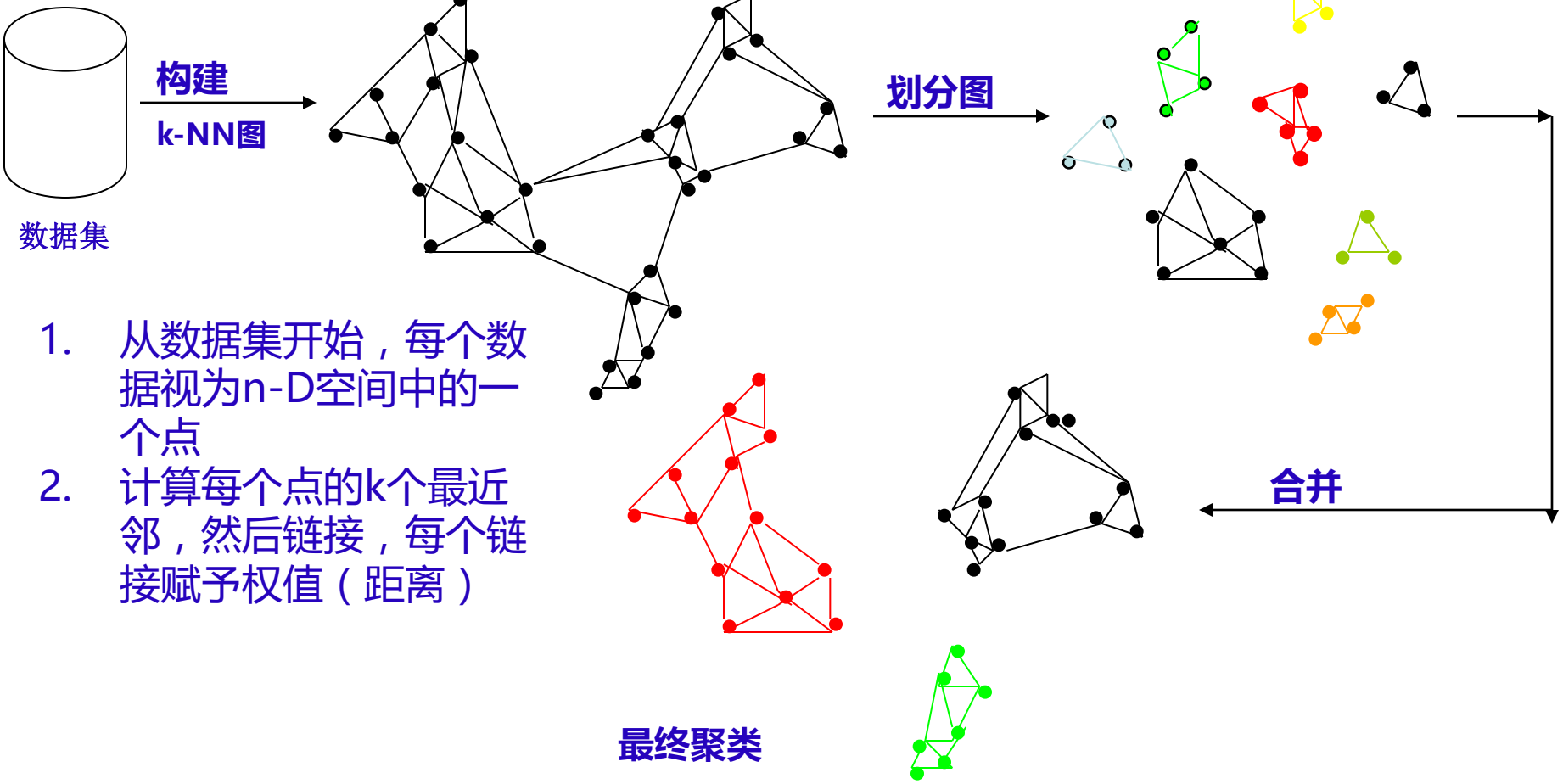
BIRCH：算法与评述

L:叶子结点的最多条目数

- 初始化算法参数B、 L和T
- 扫描数据集，把数据对象Di插入CF-tree
  - 找到离Di最近的叶节点条目，加入Di，并重新计算CF
  - 若新条目直径>T，就分裂叶节点/对应父节点

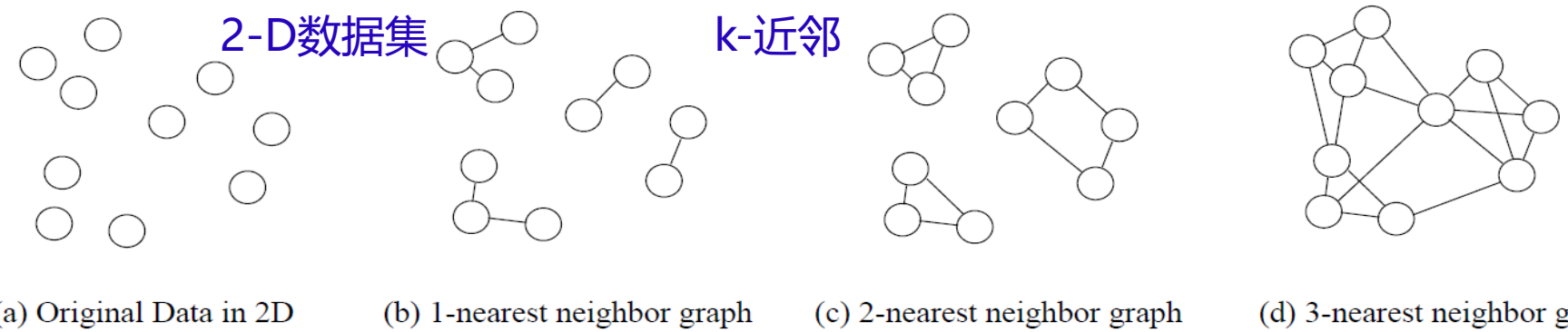
- 评述：
  - B，T和L的确定没有标准和经验，但它们显著影响性能
  - 对数据对象的输入次序敏感/次序相关的
  - 时间复杂度O(n),只需要装入一次数据集，适用于大型数据集
  - 重点在CF和CF-tree的设计和利用很巧妙，压缩存储数据集
  - 动态调整参数，使得其和机器的内存配置相适应
  - 对叶条目对应的cluster可以继续处理：采用其它技术（比如聚类），删除小的cluster（离群点），合并稠密的cluster
  - T的使用，使得cluster形状受约束

Chamleon：思想



1. 从数据集开始，每个数据视为n-D空间中的一个点
2. 计算每个点的k个最近邻，然后链接，每个链接赋予权值（距离）

Chamleon：基础概念



(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

- 两个子集/簇Ci,Cj之间的绝对互联性： $EC_{\{Ci,Cj\}}$ 簇之间连边权值之和
- 簇Ci内的互联性： EC_{Ci} 定义为将簇划分为近似相等两个子簇的“分割边”的权值之和

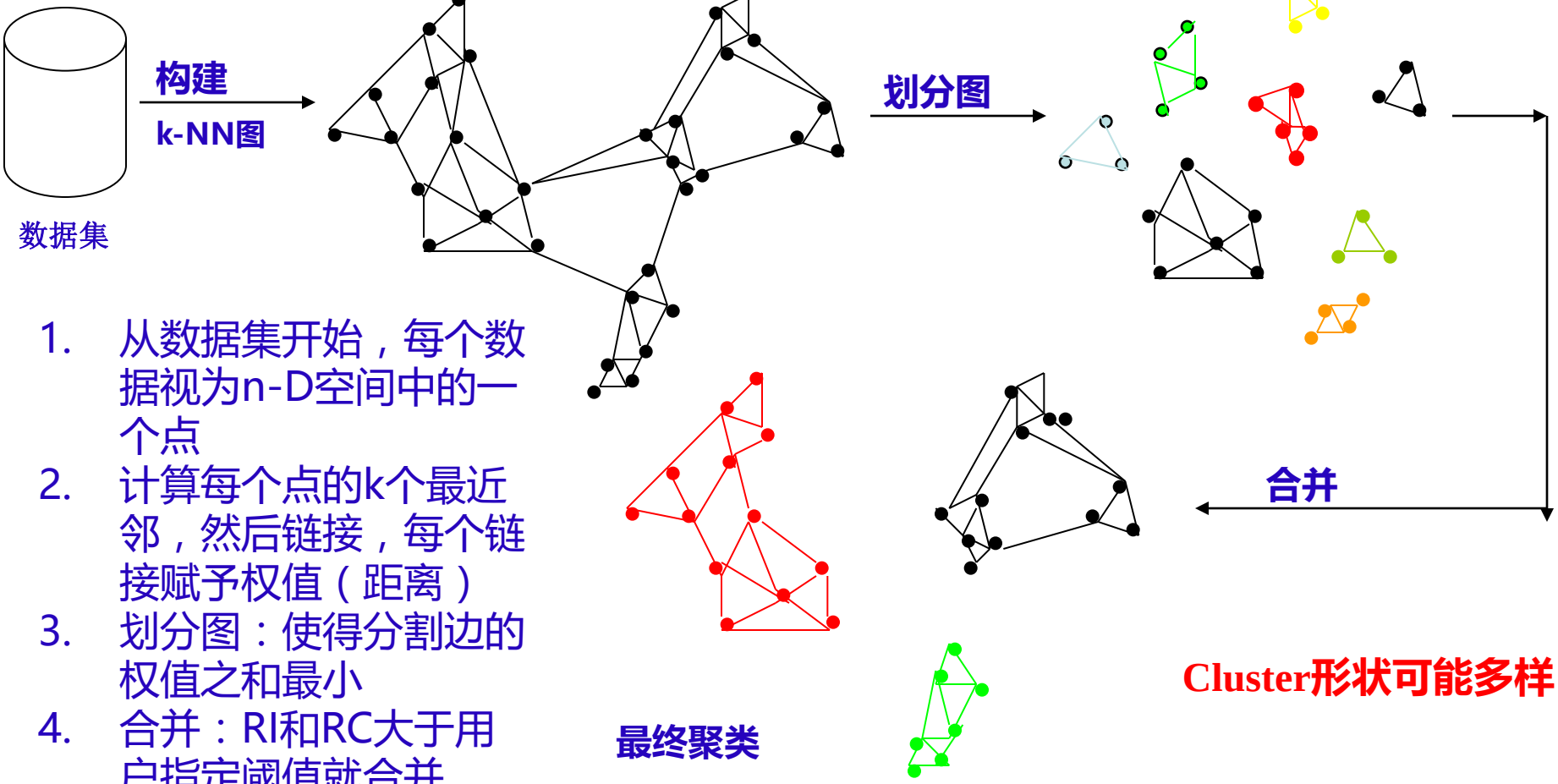
两个子集/簇Ci,Cj之间的相对互联性

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

两个子集/簇Ci,Cj之间的相对接近性

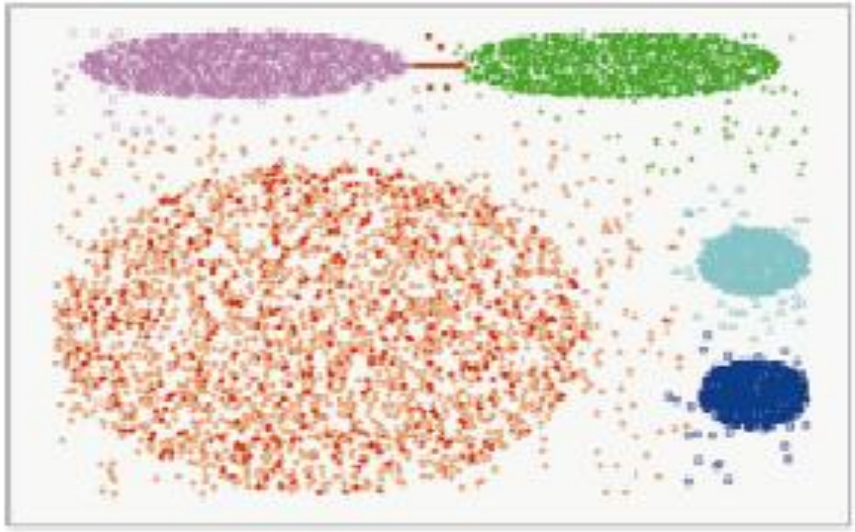
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

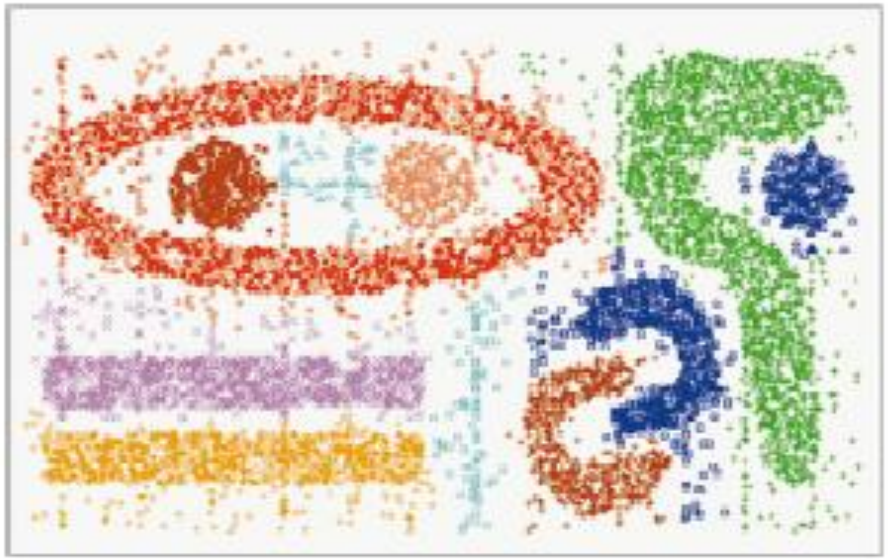
Chamleon：思想

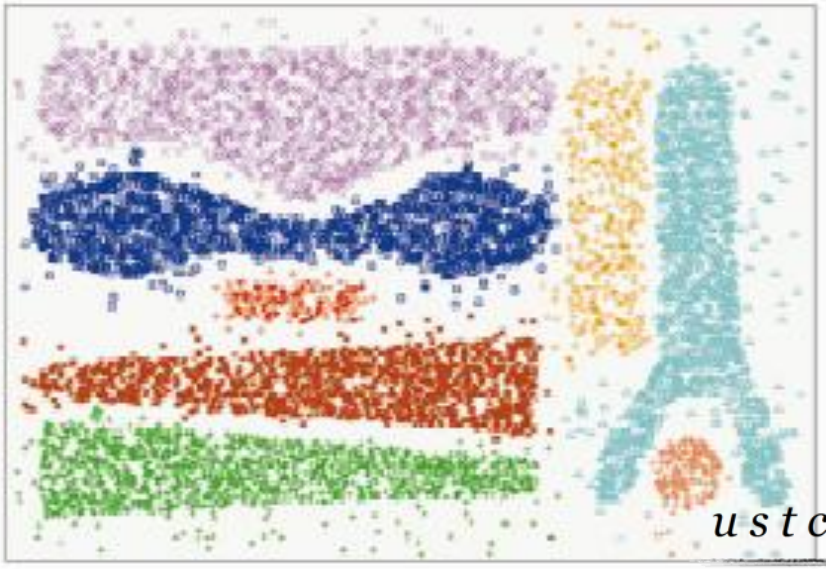


1. 从数据集开始，每个数据视为n-D空间中的一个点
2. 计算每个点的k个最近邻，然后链接，每个链接赋予权值（距离）
3. 划分图：使得分割边的权值之和最小
4. 合并：RI和RC大于用户指定阈值就合并

Chamleon：结果例子







概率层次方法：动机

层次方法的不足

- 距离度量方法的确定存在困难
- 数据对象存在缺失值会导致距离计算困难
- 划分/合并操作是采用的启发式方法，难以保证最优性

概率层次方法：思想

概率层次聚类

假定存在一个“产生模型”/高斯分布/伯努利分布等，数据集是该模型的样本集	缺失值：用该属性的已有值来构建一个值的分布函数	用概率模型来度量距离

产生模型：简单例子

- 给定一个1-D样本集 $X=\{x1,x2,\ldots,xn\}$,为做聚类分析，我们假定产生模型是高斯分布

$$\mathcal{N}(\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- xi 产生的概率为：

$$P(x_i|\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- 整个数据集/样本集产生的概率，即似然函数：

$$L(\mathcal{N}(\mu,\sigma^2):X)=P(X|\mu,\sigma^2)=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- 寻找产生模型，即 寻找高斯分布参数μ和σ^2 ，满足

$$\mathcal{N}(\mu_0,\sigma_0^2)=\arg\max\{L(\mathcal{N}(\mu,\sigma^2):X)\}$$

最大似然

概率层次聚类：算法

- 若数据集被划分为m个clusters C1,C2,...,Cm，计算划分质量：

$$Q(\{C_1,\ldots,C_m\}) = \prod_{i=1}^{m} P(C_i)$$

其中$P()$是最大似然

- 若C1和C2合并为一个cluster C1 U C2，则上述质量改变为：

$$\begin{aligned} & Q((\{C_1,\ldots,C_m\} - \{C_{j_1}, C_{j_2}\}) \cup \{C_{j_1} \cup C_{j_2}\}) - Q(\{C_1,\ldots,C_m\}) \\ = & \frac{\prod_{i=1}^{m} P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - \prod_{i=1}^{m} P(C_i) \\ = & \prod_{i=1}^{m} P(C_i)\left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - 1\right) \end{aligned}$$

$$dist(C_i, C_j) = -\log \frac{P(C_1 \cup C_2)}{P(C_1)P(C_2)}$$

左式/距离小于0则合并两个簇

初始时刻，每个数据对象是一个簇

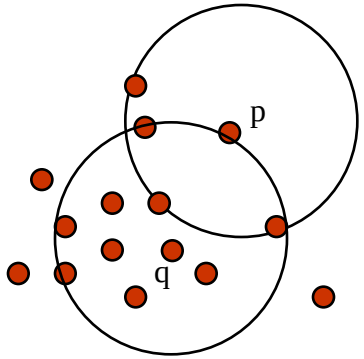
基于密度的方法

cluster的密度比较大/稠密

基础：密度方法

- 给定数据集D，讨论其中任一数据对象o的密度，定义
  - o的ε-领域：$N_\varepsilon(o)=\{p|dist(p,o)< \varepsilon, p \in D\}$
  - o的密度：$|N_\varepsilon(o)|$，即领域中数据的个数
- 参数Minpts，表示领域最小值/阈值，若对象o的密度大于Minpts，则称对象o为“核心对象”
- 核心对象的意义：核心对象及其邻域构成稠密区
- 直接密度可达：对象p从核心对象q直接密度可达，当且仅当p在$N_\varepsilon(q)$，且$|N_\varepsilon(q)|$>Minpts

非对称的



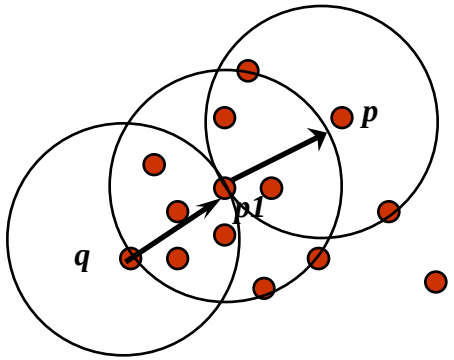
MinPts = 5

ε = 1 cm

基础：密度方法

密度可达

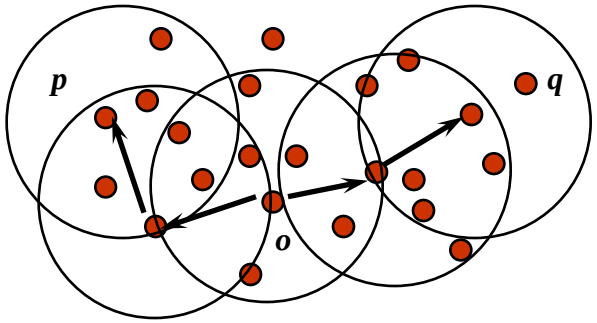
1. 存在一条链，前一个直接密度可达后一个数据
2. 链上前n-1个数据都是核心对象



都是给定ε和Minpts

密度连通

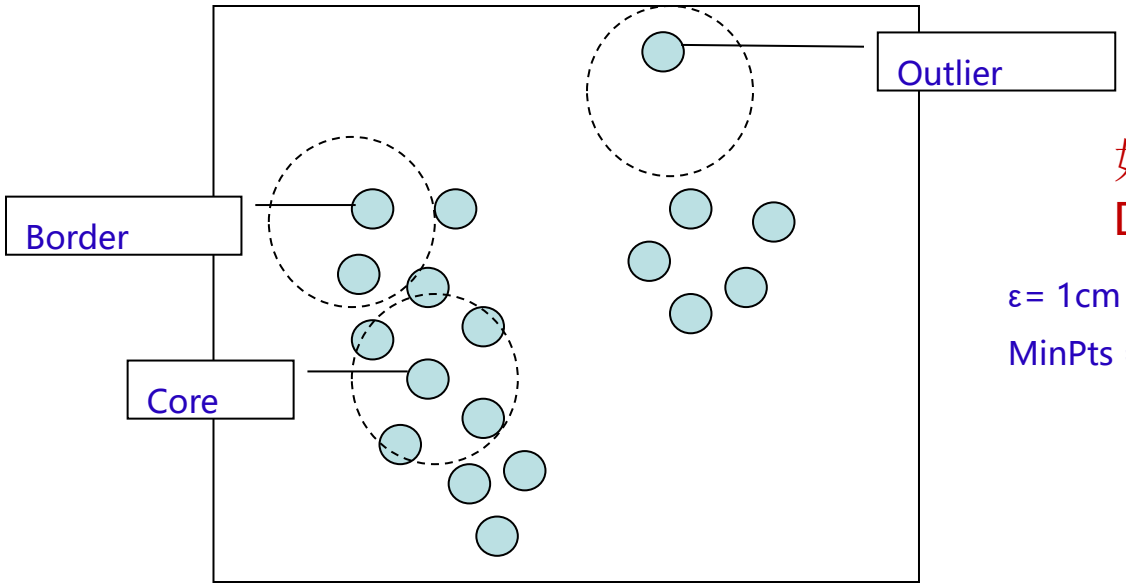
1. 存在一个数据o，它分别密度可达p和q
2. 则称p和q密度连通



Cluster： 密度连通的最大点集，如何寻找？

- 初始，所有点都是unvisited，随机选择一个，记为p，并标注visited，检查是否是核心对象，否则记为噪声/非核心点
- 若p是核心对象，则创建新簇C，将$N_\varepsilon(p)$ 添加到集合N；集合N是待检查点集合
- 检查N中的点，假设为q，加入C，检查q的标记，若是unvisited，则改为visited，并检查其邻域大小$|N_\varepsilon(q)|$,若q为核心对象，则$N_\varepsilon(q)$都加入N
- N为空时，找到一个簇C

第二个Cluster重复上述过程



好的数据结构设计，可使得
DBSCAN: O(nlgn)

ε= 1cm

MinPts = 5

DBSCAN:参数敏感性

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

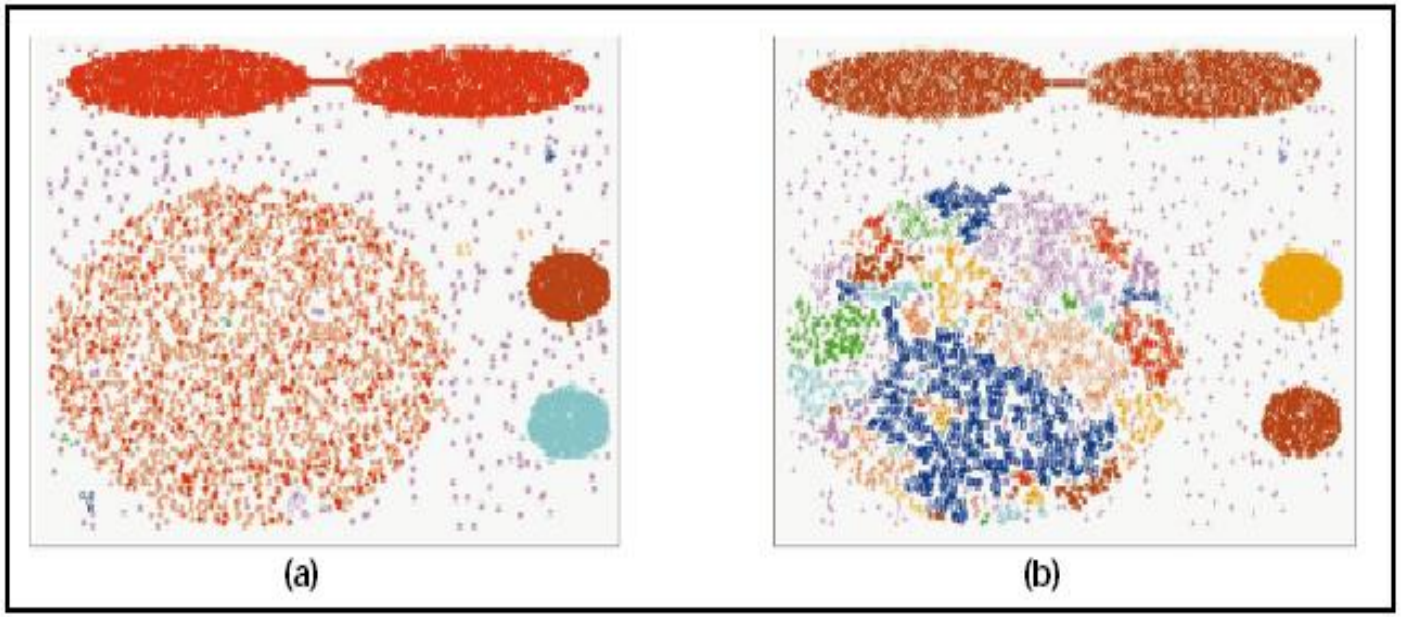
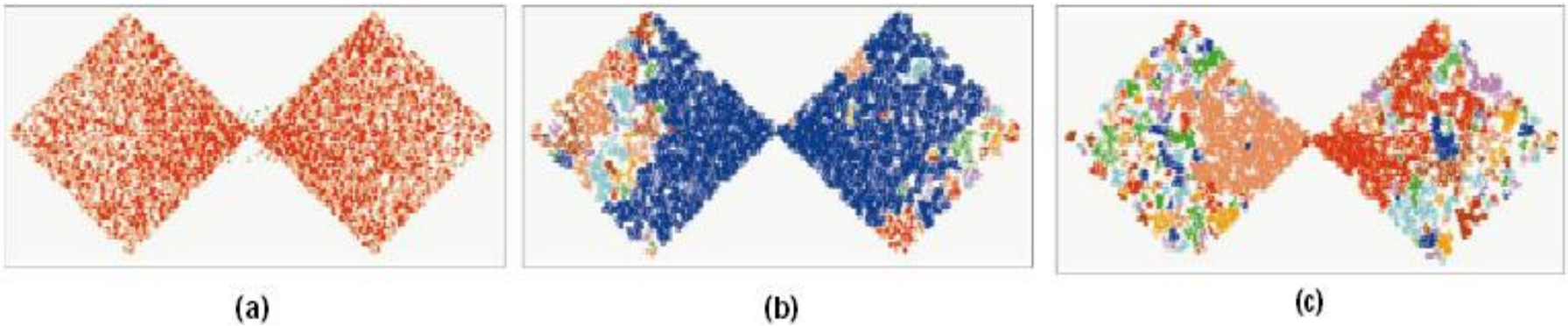


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



ε，Minpts

DBSCAN的改进

给定 ε，Minpts，判断点是否是核心对象 DBSCAN
给定Minpts，点是核心对象， ε至少是多少？ OPTICS

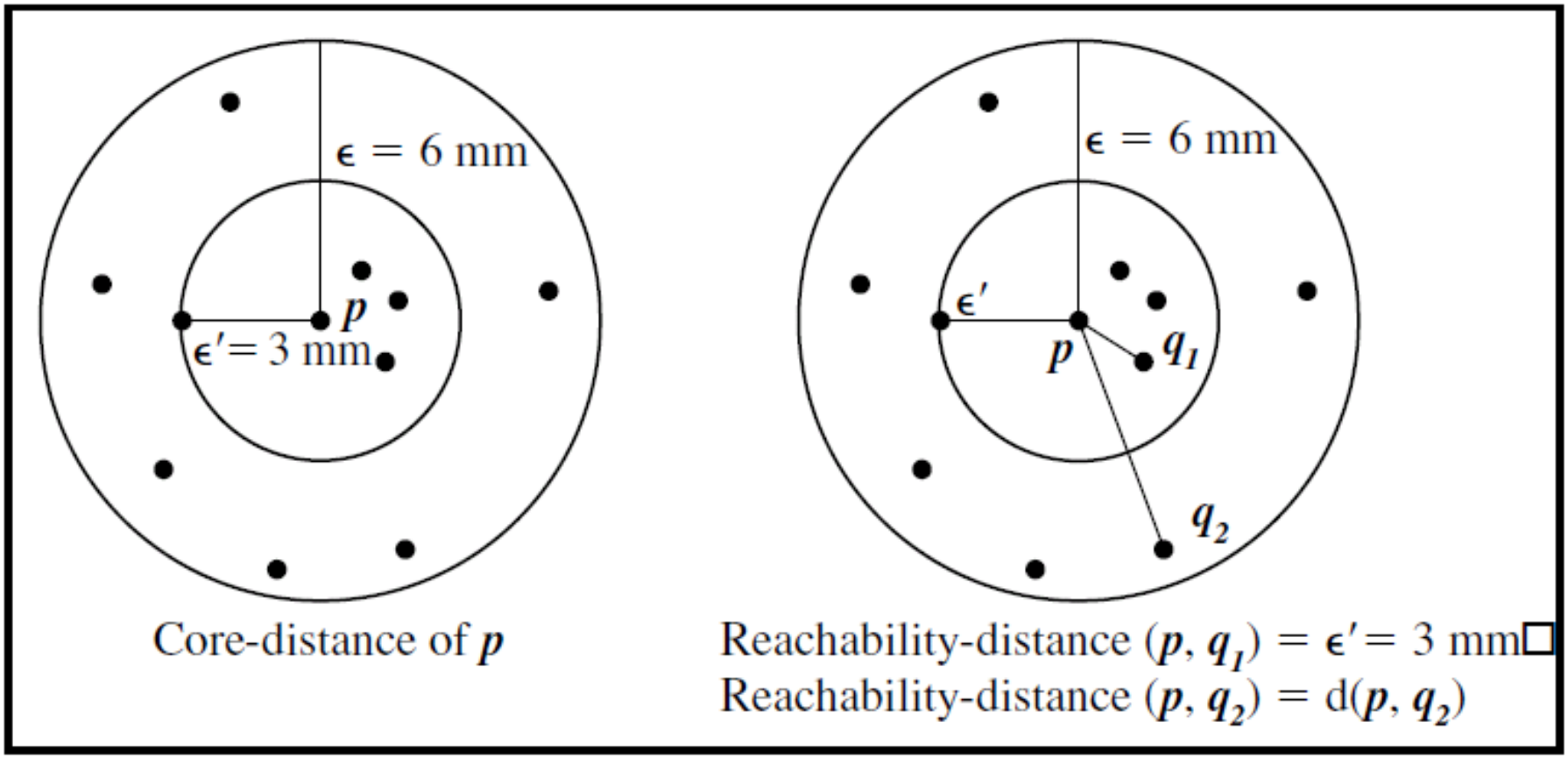


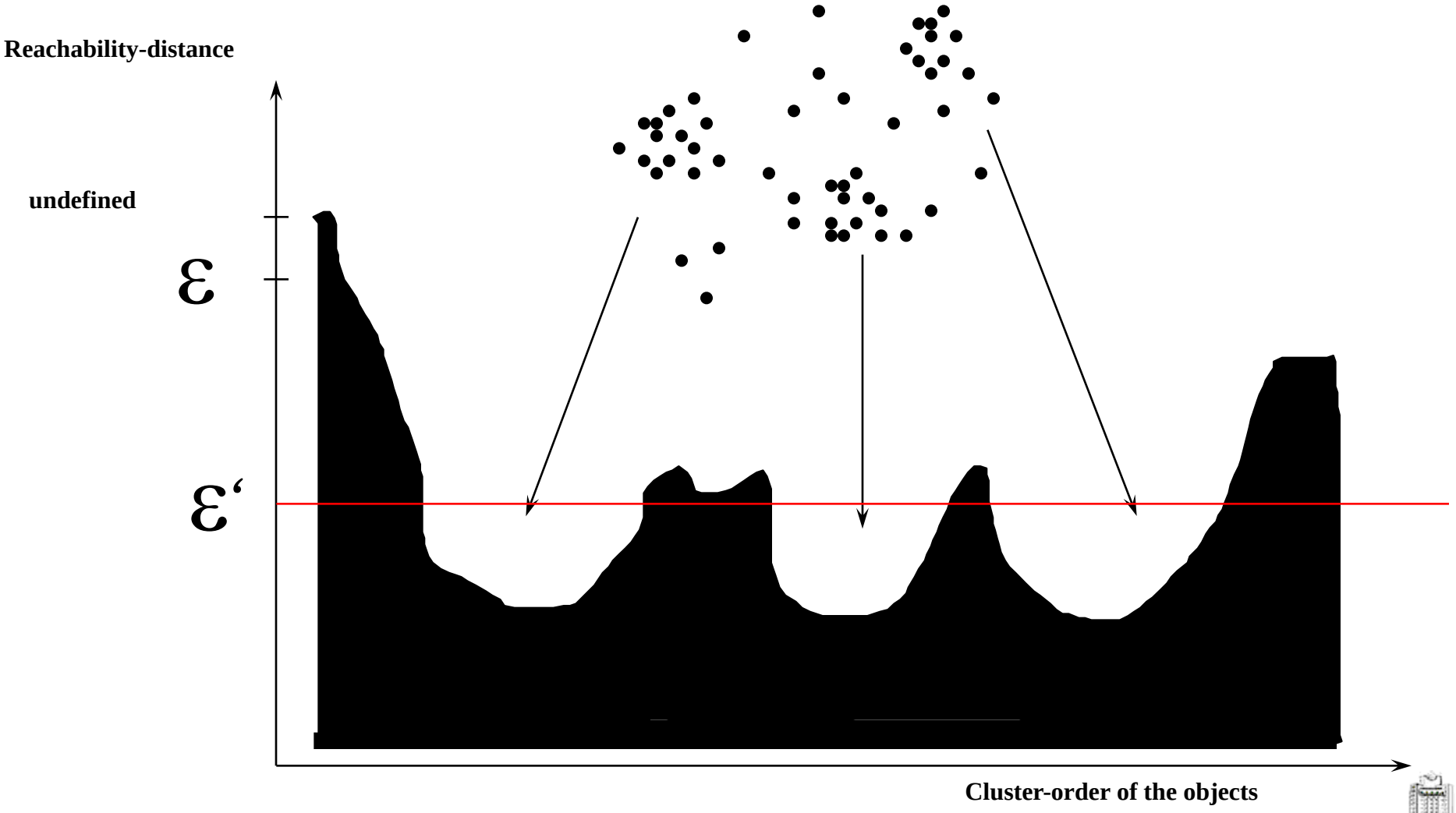
Figure 10.16: OPTICS terminology. Based on [ABKS99].

OPTICS: 算法 略

1. 初始，给定ε ,Minpts
2. 任选数据p，检查邻域，确定核心距离，设置可达距离为“未定义”
3. 输出p，并检查p是否是核心对象
   3.1 若p是核心对象（$\varepsilon' < \varepsilon$），对$N_\varepsilon(p)$ 的数据q，计算p到q的可达距离，更新p到q的可达距离；若q是 未处理过（检查过是否是核心对象），将q添加到OrderSeeds表中
   3.2 若p不是核心对象，则继续下一步
   3.3 选择OrderSeeds表中的一个数据对象，执行步骤3，直到OrderSeeds表空
4. 数据集中若还有其他数据未处理，跳到步骤2

上述算法OrderSeeds表中的数据按照核心对象达到的最小距离排序
可以想象：数据集中某些数据对象，都有其它核心对象到它的可达距离，这些可达距离表述了核心对象到该数据的远近，每个数据对象采用最小的可达距离来描述自己的特点，用于排序。
注意：还有些数据，没有可达距离（未定义）

OPTICS: 解释


Reachability-distance
undefined
ε
ε‘
Cluster-order of the objects

评述：密度方法

- 核心问题：密度估计
- 概率论中，密度估计是指从样本/观测数据中获得随机变量的概率密度函数。
- DBSCAN和OPTICS中，估计每个样本的密度，方法是邻域中对象的个数，参数ε影响很大，也就是说密度对ε的值非常敏感
- 改进方法：核密度估计/kernel density estimation
- 非参数密度估计方法
- 一个样本出现了，那么它的出现概率较大，其附近区域的点出现的概率也较大，故可以假定任何一个点的概率密度依赖该点到样本的距离
- 假设*x1，x2,…,xn*是随机变量f的独立同分布的*n*个样本，概率密度函数的近似核密度函数：

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

DENCLUE

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \qquad K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$$

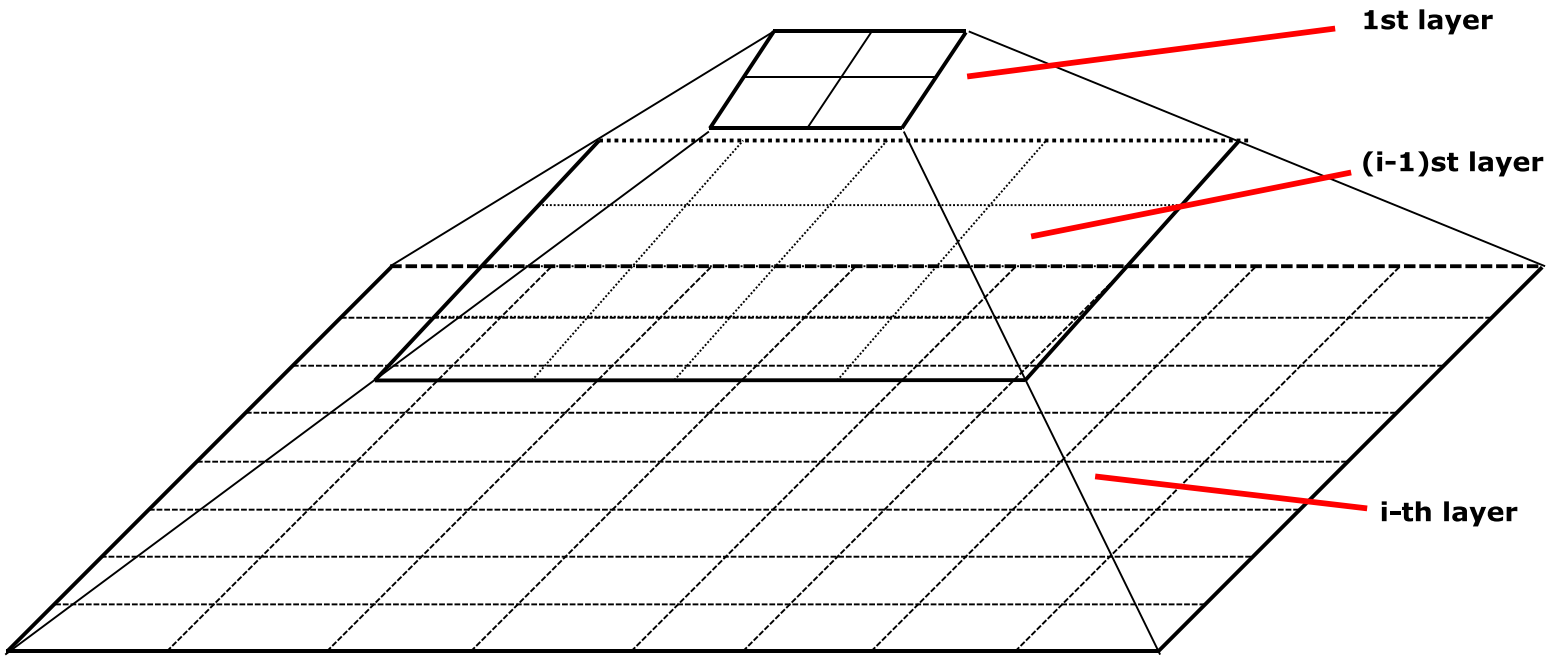
- 由上述函数，可以得到任何一个点x的概率
- 密度吸引子：某个点x的概率比周围所有点的密度都要大（通常要求密度吸引子的概率要大于某个阈值ξ）
- 密度吸引子构成簇的中心
- 密度吸引子的发现：Local search算法/爬山法

- 基于密度吸引子的簇的构造：将密度吸引子归并到不同的簇，同簇的密度吸引子之间具有概率大于ξ的路径

任意形状的簇，DBSCAN的泛化，抗噪，速度快

基于网格的方法 略

STING/统计信息网格



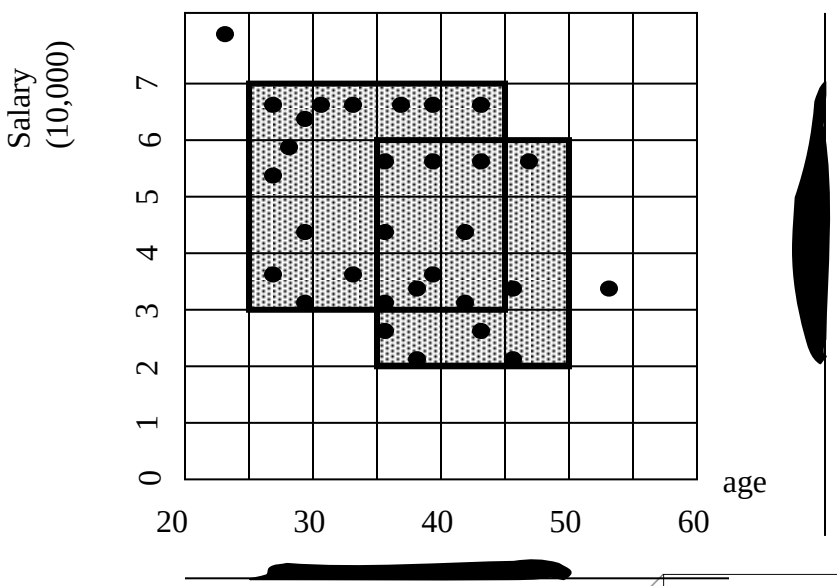
最底层的格子对应着cluster

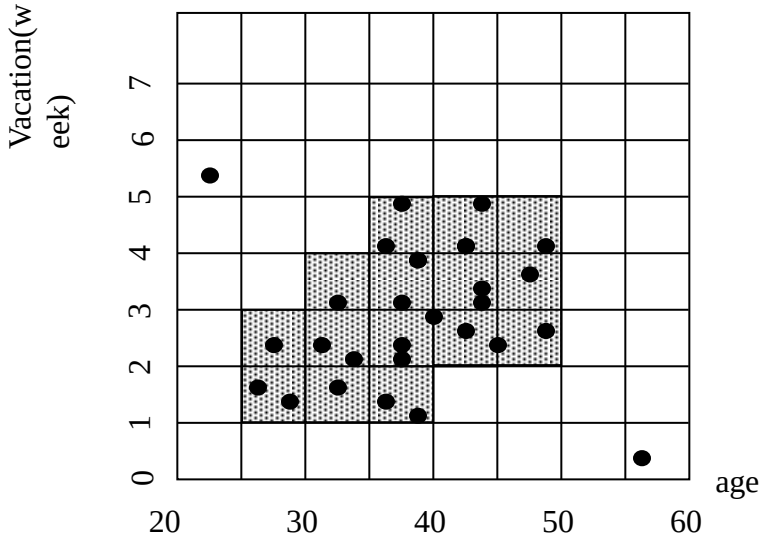
- 空间被划分为多个超立方体的小格子
- 小格子具有不同的层次/分辨率
- 高层小格子被划分为多个低层的小格子
- 每个小格子有统计信息（计数/最大值/最小值/均值/标准差/服从分布等），高层格子的统计信息可由底层的来计算

CLIQUE

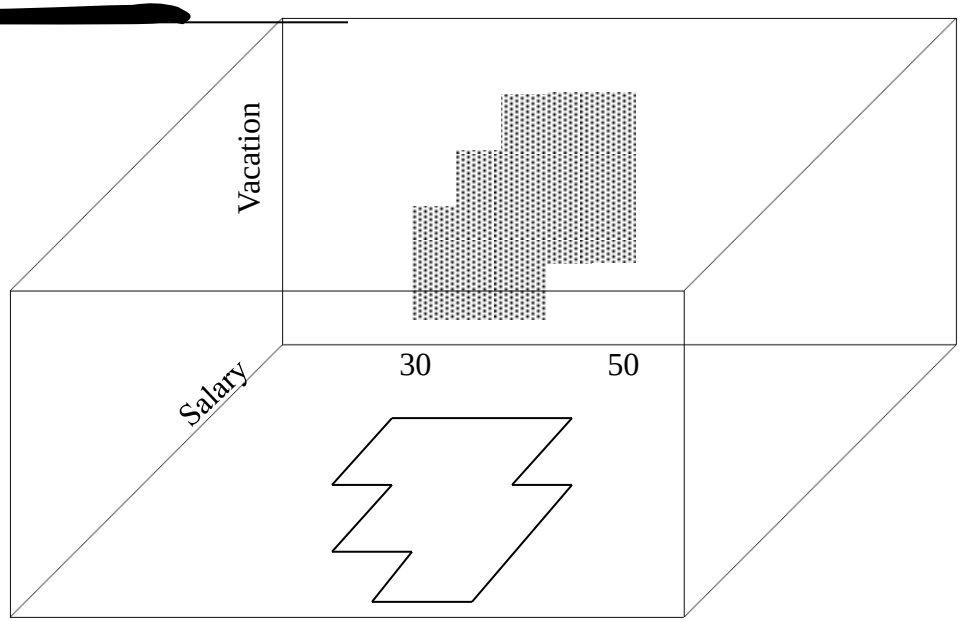
- 每一维被等分成若干区间
- 一个m-D的数据空间就被分割为若干不相交的小格子
- 如果一个小个子里的数据个数超过了预定义的阈值l，则称该小格子为“稠密单元”
- 相邻稠密单元的并集构成了簇
- 问题：如何找到稠密单元？
- 问题：如何求不同维/子空间的稠密单元的并集？

CLIQUE





如果在3-D空间中是稠密的，那么在其任何投影空间中都应该是稠密的



类似Apriori思想，寻找高维空间的稠密单元

基于网格方法的评述

- STING
  - 易并行化，增量式
  - 时间复杂度低O(n)读入数据，分析O(k)，k是最小粒度格子数
  - 不足：分类边界都是水平或垂直的线

- CLIQUE
  - 自动识别高维数据空间中数据稠密的子空间
  - 不需要对数据分布做假设，数据输入次序无关/不敏感
  - 输入数据的多少和维度，对算法的时间复杂度影响是线性的
  - 不足：聚类精度可能会降低

如何评估不同聚类算法得到的结果？

趋势/簇数/质量

估计聚类趋势

- 比较数据集和均匀分布的数据集之间的差异
- Hopkins统计量
  - 数据集D视为某个随机变量o的样本集
  - 在数据集D形成的空间中均匀随机采样n个点p1,p2,...,pn
  - 对每个pi，找一个xi=v, v∈D且满足min {dist(pi,v)}，即每个均匀采样得到的样本都找一个离自己最近的数据集中的点
  - 均匀随机从D中采样n个点q1,q2,...,qn,并找出D中离qi最近（不等于qi）的点，记为yi
  - 计算Hopkins统计量：

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

包含簇时，$\sum y_i$ 会显著小于$\sum x_i$，H接近0
不包含明显的“自然簇”时，H约为0.5

尝试回答问题：数据集中存在簇（结构）吗？

确定簇数

- 经验方法： √(n/2)个簇，每个簇√2n个数据，总数据n个
- 肘方法：
  - 给定k，采用一种聚类方法对数据集聚类，计算每个簇的簇内方差之和S
  - 增加k，S会降低，因为簇多了，簇小了，簇内差异变小
  - 检查S随k降低的速度，或者说找第一个“拐点”
- 交叉验证法：
  - 将数据集划分为近似相等的m分
  - 用其中m-1份构建聚类模型，剩下一份检验聚类质量（比如，检验数据到最近“簇心”的距离平方和）
  - 对给定的k，重复上述过程m次
  - 比较不同的k

度量聚类质量

- 外在方法： 给定一个标准，标准看成是“类标号”，故是有监督的
  - Bcubed精度和召回率
- 内在方法：没有给定标准,无监督的
  - 簇间分离状况，簇内紧凑状况
  - 数据对象o的轮廓系数：
  - $a(o)$是数据o到其所属簇的其它数据的距离均值，表示簇内紧凑性
  - $b(o)$是数据o到其它数据对象的距离均值，表示o和其它簇的分离性
  - $s(o)$取值在[-1,1]，接近1时，o所在簇紧凑且远离其它簇；为负数时，表示o离其它簇对象的距离比自己所在簇的对象更近
  - 聚类方法获得的聚类质量：所有对象的轮廓系数之和/均值

$$s(o)=\frac{b(o)-a(o)}{\max\{a(o),b(o)\}}$$

度量聚类质量

- 给定数据集D={o1,o2,...,on}, C是D的一个聚类结果，L(oi)是聚类标准给出的oi的类别，C(oi)是C给出的oi的类别；
- 考虑两个对象oi，oj，定义正确性

$$\text{Correctness}(o_i,o_j) = \begin{cases} 1 & \text{如果 } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{其他} \end{cases}$$

一个对象的精度指C给出的同一个簇中，多少个等于L给定的

$$\text{Precision BCubed} = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_{o_j: i\neq j, C(o_i)=C(o_j)} \text{Correctness}(o_i,o_j)}{\| \{o_j \mid i \neq j, C(o_i) = C(o_j)\} \|}$$

一个对象的召回率指L给出的同一个簇的对象多少被C正确给定了

$$\text{Recall BCubed} = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_{o_j: i\neq j, L(o_i)=L(o_j)} \text{Correctness}(o_i,o_j)}{\| \{o_j \mid i \neq j, L(o_i) = L(o_j)\} \|}$$