WDS

DataScience@Web

密度聚类 层次聚类

赵满

2018.08.17

内容概要

- 密度聚类基本概念
- 密度聚类的常见算法
  - DBSCAN: Ester, et al. (1996)
  - OPTICS: Ankerst, et al (1999).
  - DENCLUE: Hinneburg & D. Keim (1998)
- 层次聚类基本概念
- 层次聚类常见算法
  - AGNES & DIANA
  - BIRCH：Zhang, Ramakrishnan(1996)
  - ROCK:S. Guha, R. Rastogi & K. Shim(1999).
  - Chameleon：G. Karypis, E.H. Han, and V. Kumar(1999)

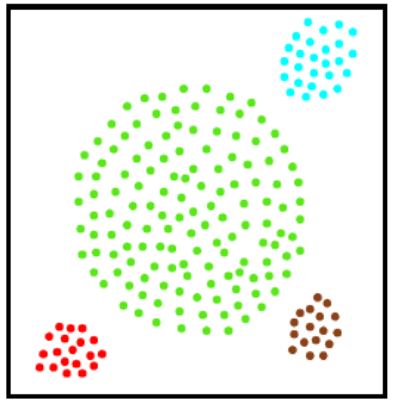
基于密度的聚类 Density-based clustering

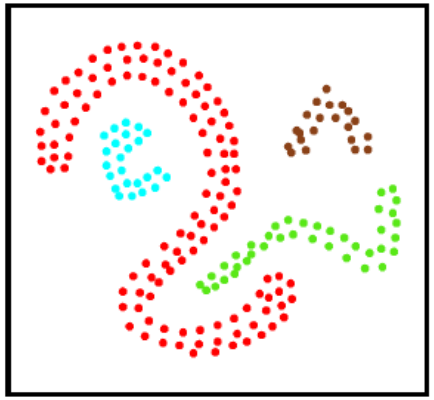
聚类结构能通过样本分布的紧密程度确定。

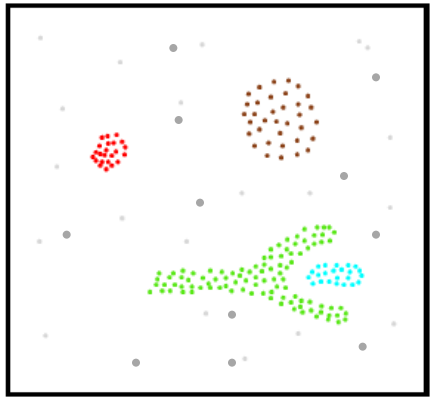
核心思想：先发现密度较高的点，再把相近的高密度点连成一片，进而生成各种簇。

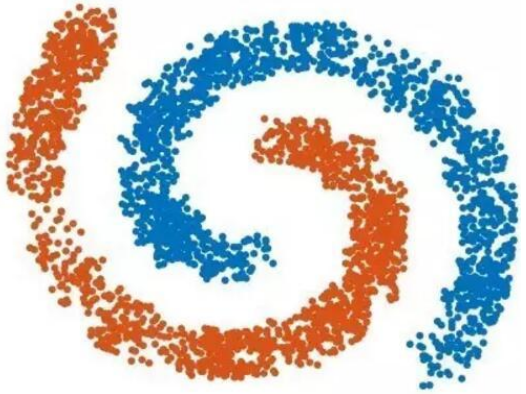
主要特点:

- 发现任意形状的聚类
- 处理噪音





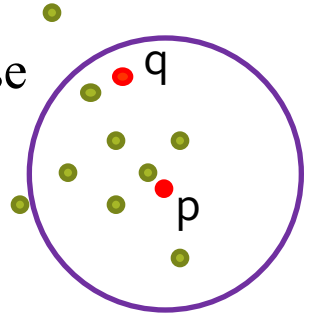




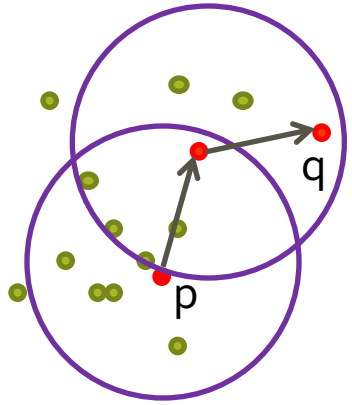
基于密度的聚类方法数据中可以发现各种形状和各种大小的簇

DBSCAN

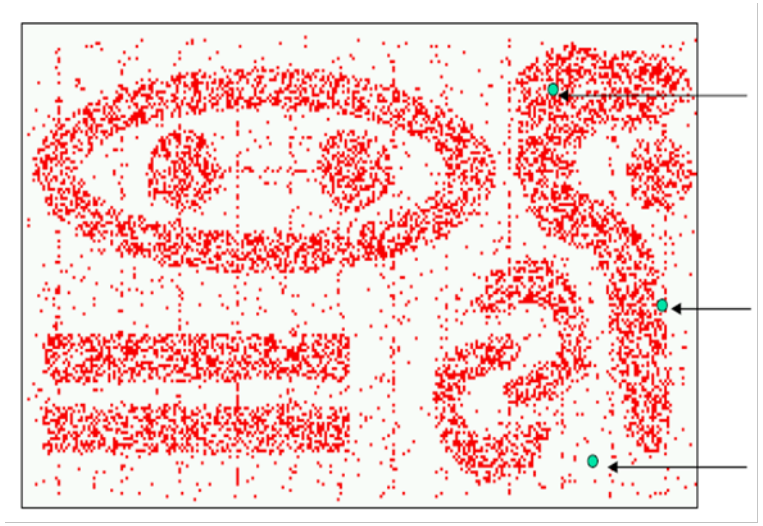
Density-Based Spatial Clustering of Applications with Noise

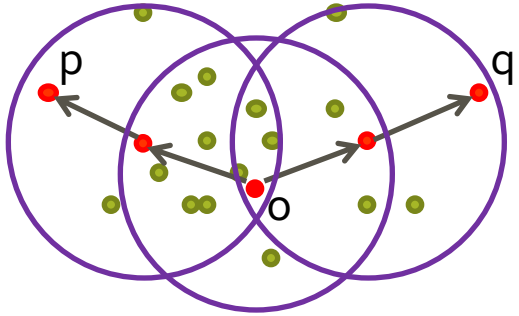


directly density reachable



density reachable





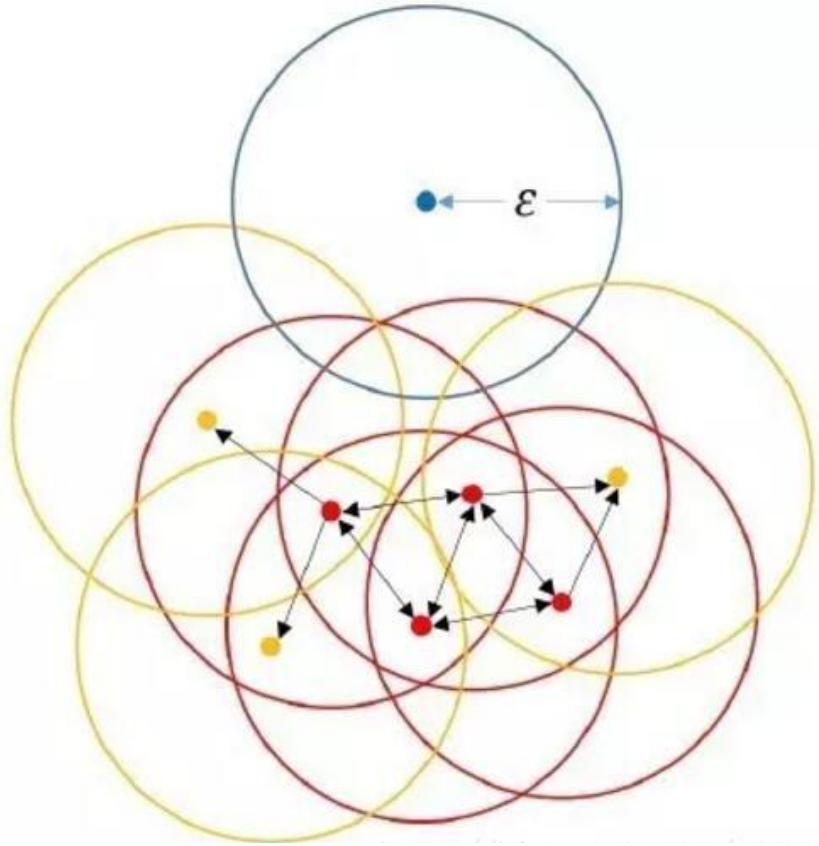
density connected

DBSCAN

Input ：dataset，邻域大小eps，密度阈值MinPts
Output ：簇集合

- 以数据点为圆心，以eps为半径画圆，有多少个点在这个圈内，这个数就是该点密度值。
- 小于MinPts的圆心点为低密度点，大于或等于MinPts的圆心点为高密度点（核心对象）。
- 如果高密度点在另一个高密度的点的圈内，这两个点就可以连接起来。（直接密度可达）
- 如果有低密度的点也在高密度的点的圈内，把它也连到最近的高密度点上，称之为边界点（边界对象）。
- 这样所有能连到一起的点（密度可达）就成了一个簇，而不在任何高密度点的圈内的低密度点就是异常点。

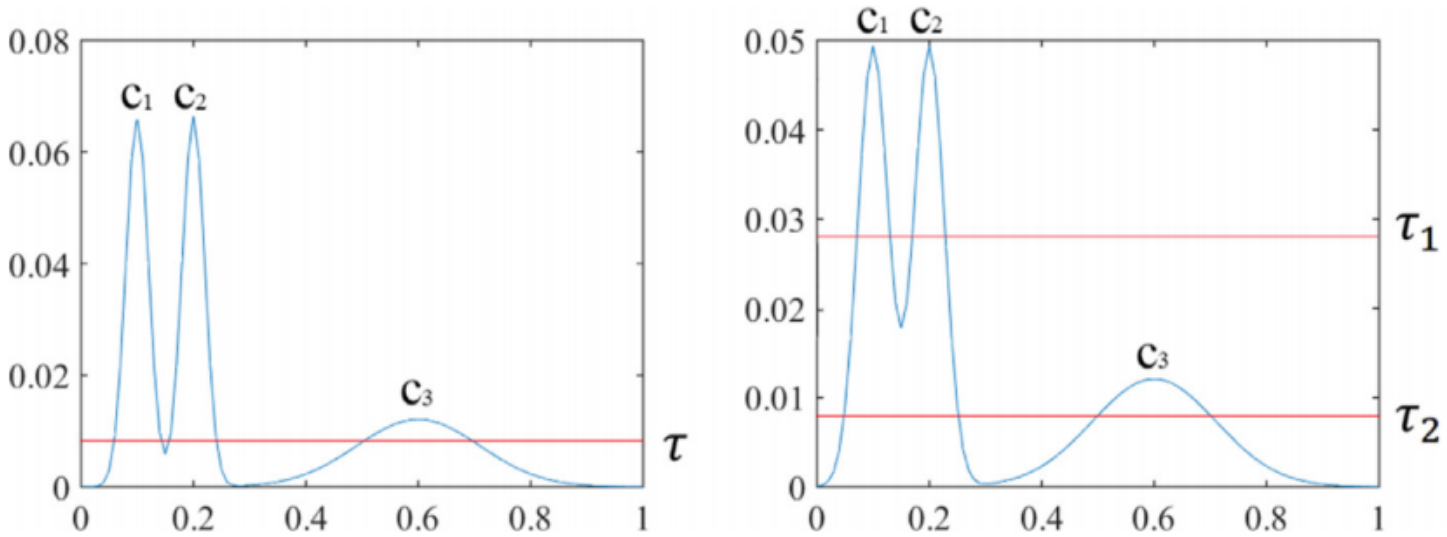
➢ **由核心对象和其密度可达的所有对象构成一个聚类**



MinPts=4
红点为高密度点，核心对象
蓝点为异常点，噪音
黄点为边界点，边界对象
红黄点成了一个簇

DBSCAN算法局限

DBSCAN使用的是全局的密度阈值MinPts, 只能发现密度不少于MinPts的点组成的簇，即很难发现不同密度的簇。



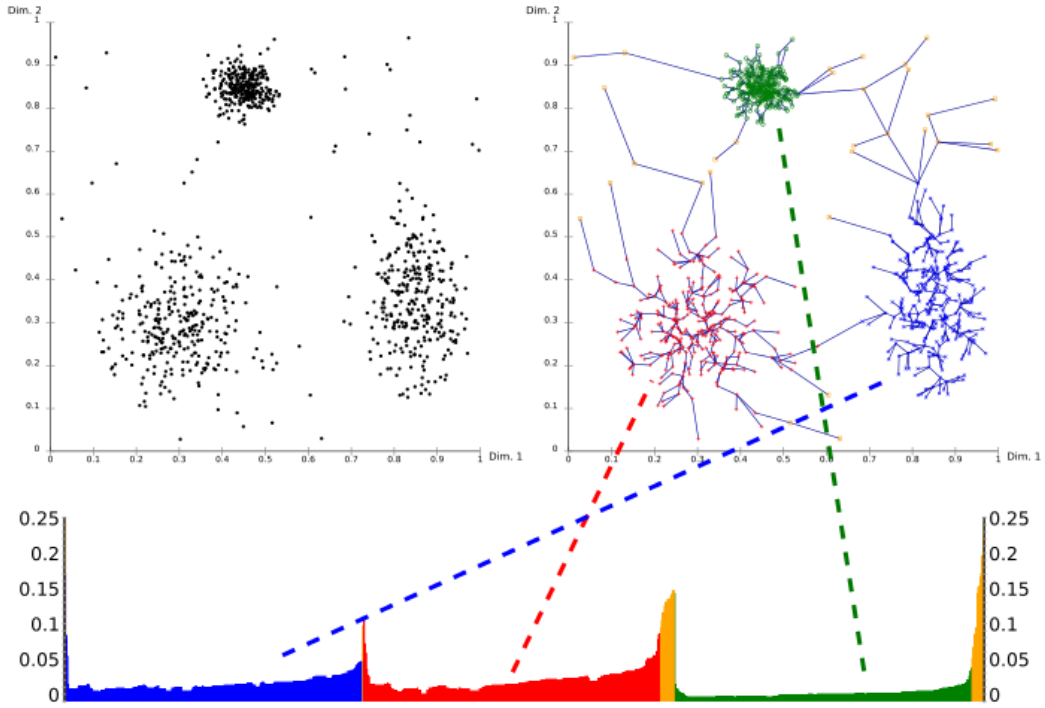
对用户定义的参数是敏感, 参数难以确定(特别是对于高维数据), 设置的细微不同可能导致差别很大的聚类.

大多数需要初始化参数聚类算法的弊端！

OPTICS

Ordering points to identify the clustering structure

OPTICS并不显示的产生结果类簇，而是将邻域点按照密度大小进行排序，为聚类分析生成一个簇排序，再用可视化的方法来发现不同密度的簇。



这个排序包含的信息等价于从一个广泛的参数设置所获得的基于密度的聚类，换句话说，从这个排序中可以得到基于任何参数eps和minPts的DBSCAN算法的聚类结果。

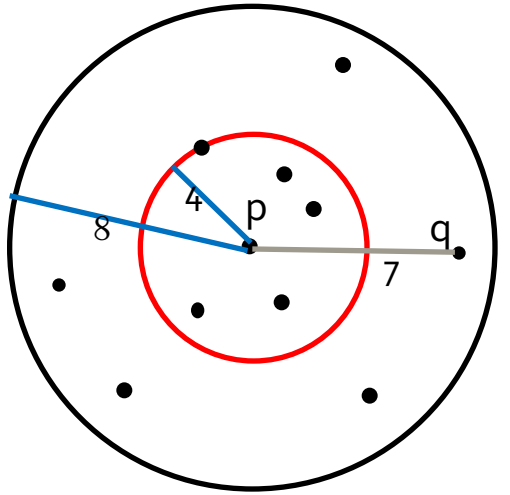
OPTICS

Ordering points to identify the clustering structure

核心距离：假定P是核心对象，然后计算关于 P 点满足阈值MinPts的最小的半径。如果p不是核心对象, p的核心距离没有意义 。

可达距离：q到p的可达距离是指p的核心距离和p与q之间欧几里得距离之间的较大值。如果p不是核心对象，p和q之间的可达距离没有意义。

一个点有多个可达距离，通常选最小可达距离，代表离该点最近的一个簇的距离。



阈值MinPts=5，P是核心对象，P的核心距离为4 ， 因为在半径4内，有5个近邻点, 满足阈值minpts>=5。
q到p的可达距离为max（4,7）=7

OPTICS算法过程(1)

输入：数据集D，初始化所有点的可达距离和核心距离为MAX，半径ε，和最少点数MinPts。
输出：结果序列

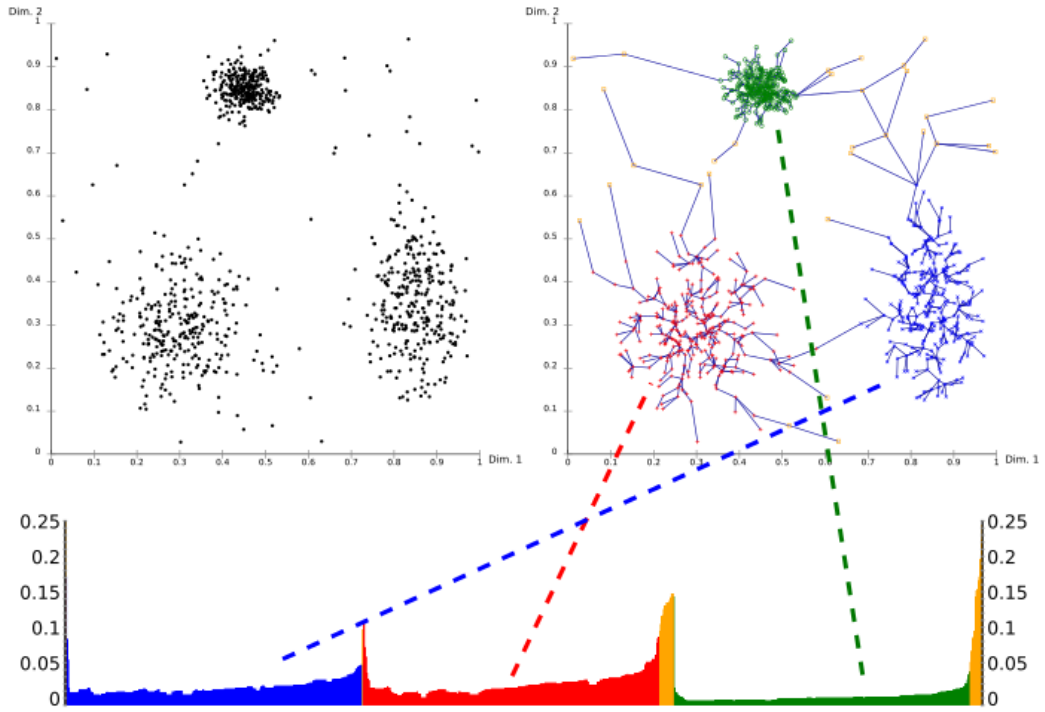
1、建立两个队列，有序队列（核心点及该核心点的直接密度可达点），结果队列（存储样本输出及处理次序）
2、如果D中数据全部处理完，则算法结束，否则从D中选择一个未处理且为核心对象的点，将该核心点放入结果队列，该核心点的直接密度可达点放入有序队列，直接密度可达点并按可达距离升序排列；
3、如果有序序列为空，则回到步骤2，否则从有序队列中取出第一个点；
   3.1 判断该点是否为核心点，不是则回到步骤3，是核心点且不在结果队列则将该点存入结果队列；
   3.2 该点是核心点的话，找到其所有直接密度可达点，并将这些点放入有序队列，且将有序队列中的点按照可达距离重新排序，如果该点已经在有序队列中且新的可达距离较小，则更新该点的可达距离。
   3.3 重复步骤3，直至有序队列为空。
4、算法结束。

OPTICS算法过程(2)

根据结果队列，给定ε，就可以输出所有的聚类。
1、从结果队列中按顺序取出点，如果该点的可达距离不大于ε，则该点属于当前类别，否则至步骤2；
2、如果该点的核心距离大于ε，则该点为噪声，可以忽略，否则该点属于新的聚类，跳至步骤1；
3、结果队列遍历结束，则算法结束。

广泛的参数设置获得基于密度的聚类结果

ε和密度阈值MinPts只起到算法辅助作用，也就是说ε和MinPts的细微变化并不会影响到样本点的相对输出顺序，这对我们分析聚类结果是没有任何影响。



OPTICS优缺

- **优点：**
  - 排序反映出数据空间基于密度的簇结构信息。基于这些信息可以容易地确定合适的ε值，并随之发现各个簇，较好地解决了DBSCAN算法的对输入参数ε敏感的问题。
  - 一个数据集合的聚类次序可以被图形化地描述，以助于直观理解。

- **缺点：**
  - 采用了复杂的处理方法，使得OPTICS实际运行的速度远远慢于DBSCAN。

DENCLUE(Density-based clustering)

由Hinneburg 和Keim (1998)提出, 是基于密度分布函数的聚类方法

该算法的原理是：

- 每个数据点的影响可以用一个数学函数来形式化地模拟，它描述了一个数据点在邻域内的影响，被称为影响函数。
- 数据空间的整体密度(全局密度函数)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，这里的密度吸引点是全局密度函数的局部最大值。

DENCLUE

数据对象y对x的**影响函数**是一个函数f^{y}_{B}，它是根据一个基本的影响函数 f_B来定义的

$$f^{y}_{B}(x)=f_B(\mathrm{x},\mathrm{y})$$

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，高斯影响函数 以及相应的全局密度函数

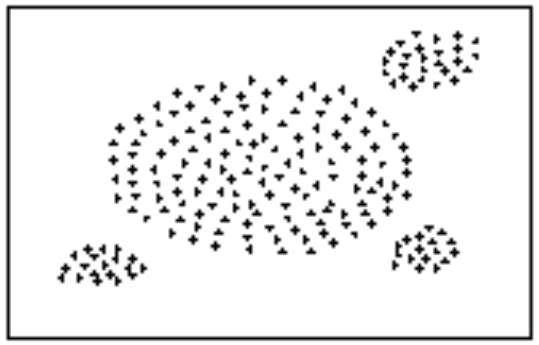
$$f_{Gauss}(x,y)=e^{\frac{d(x,y)^2}{2\sigma^2}} \qquad f^{D}_{Gauss}(x)=\sum_{i=1}^{N} e^{\frac{d(x,x_i)^2}{2\sigma^2}}$$

- 例如 欧几里得距离作为影响函数，相应的方波影响函数(square wave influence function)：

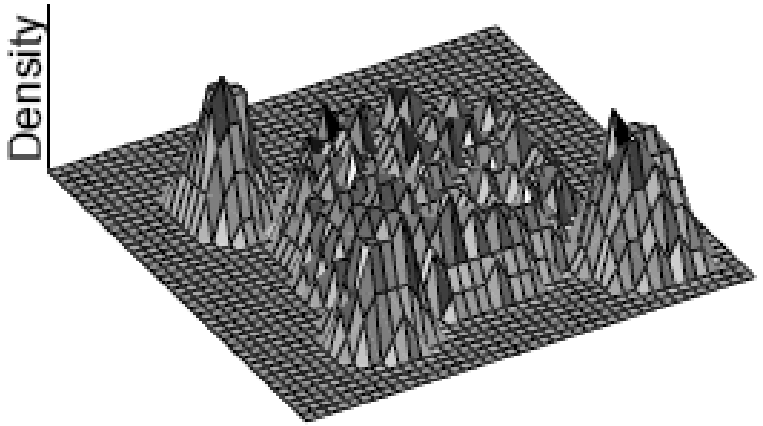
$$f_B(\mathrm{x},\mathrm{y})=\mathrm{d}(\mathrm{x},\mathrm{y}) \qquad f_{Square}(x,y)=\begin{cases}0 & \text{如果} d(x,y)>\sigma \\ 1 & \text{其它}\end{cases}$$

DENCLUE(Density-based clustering)

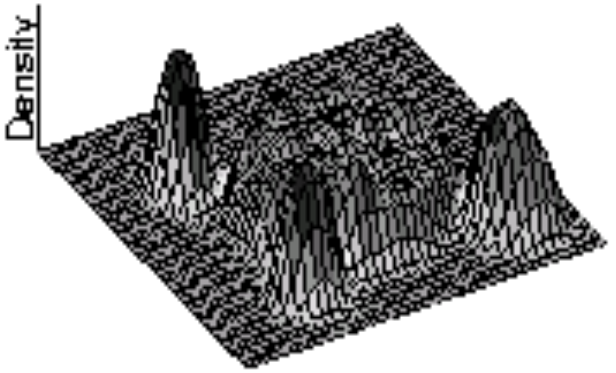
根据密度函数，能够求得该函数的梯度、密度吸引点(全局密度函数的局部最大)
对一个连续的，可微的影响函数，用梯度指导的爬山算法能用来计算一组数据点的密度吸引点。



(a) Data Set



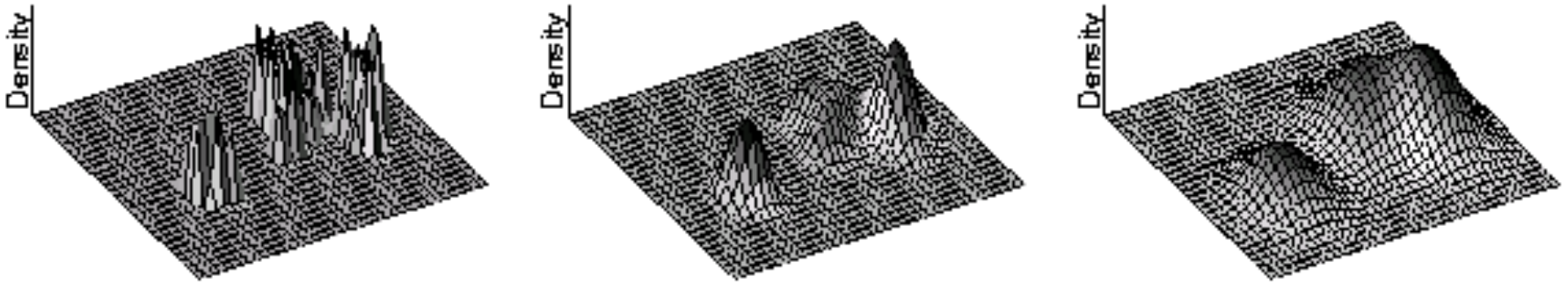
(b) Square Wave



(c) Gaussian

DENCLUE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σ; 否则(即如果它的密度函数值小于σ), 它被认为是孤立点。

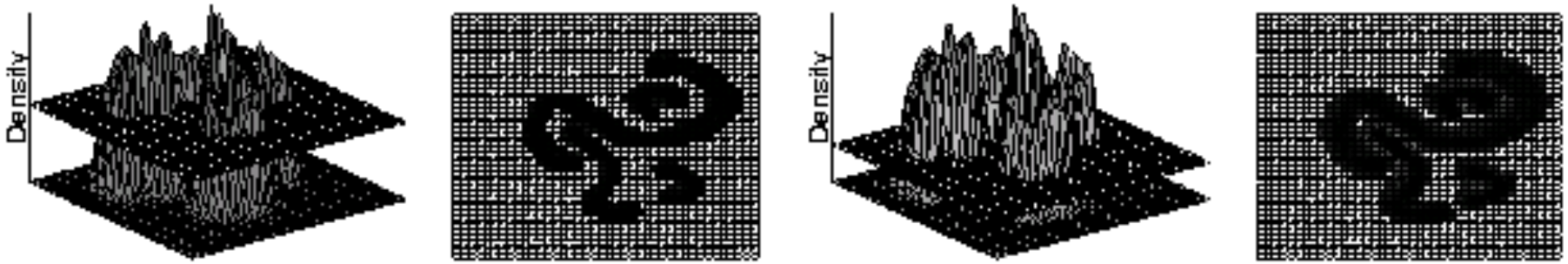


(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

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

DENCLUE中心定义的簇和任意形状的簇

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



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

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

DENCLUE优缺点

- 优点
  - 坚实的数学基础支持
  - 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)

- 缺点
  - 需要大量参数,要求对密度参数σ和噪音阀值ξ进行仔细的选择

密度聚类总结

基于密度的聚类方法可以用来过滤噪声孤立点数据，发现任意形状的簇。

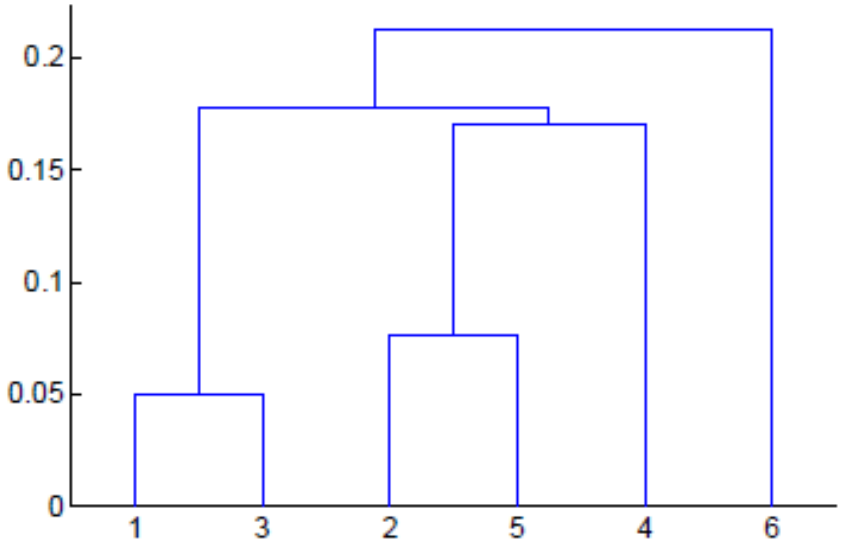
- DBSCAN：基于高密度连通区域聚类
- OPTICS：通过点排序识别聚类结构
- DENCLUE: 基于密度分布函数的聚类

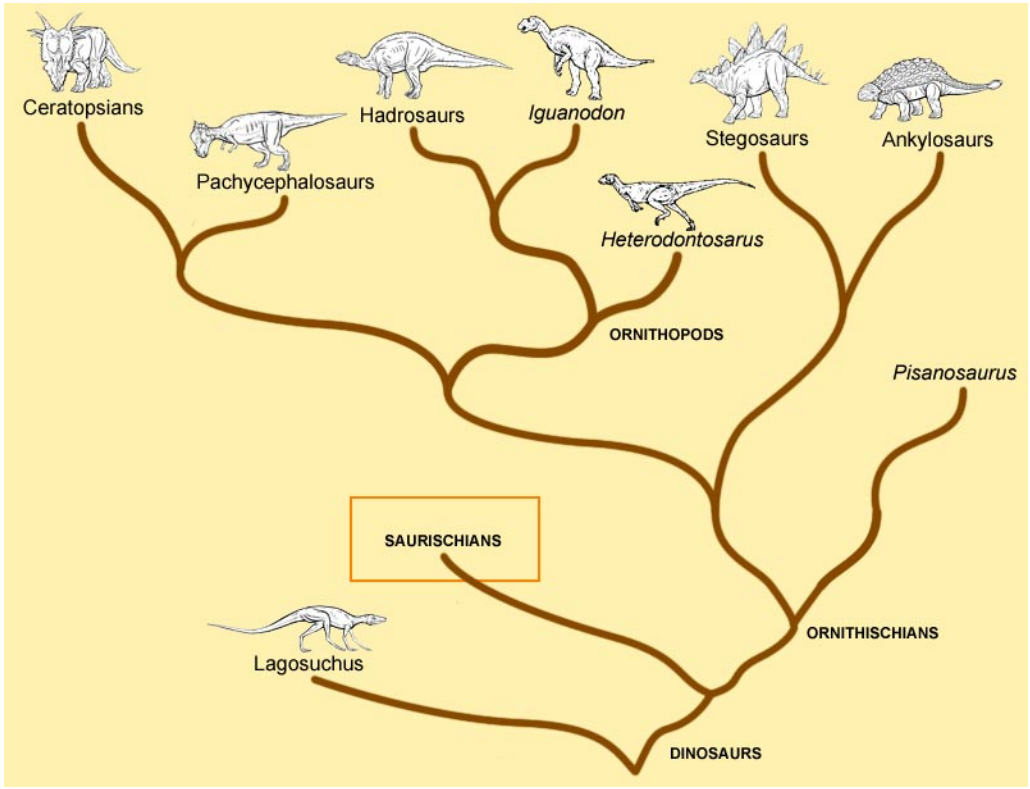
主要特点：

- 发现任意形状聚类
- 处理噪音
- 一般只需要一遍扫描
- 需要密度相关参数作为终止条件

基于层次的聚类

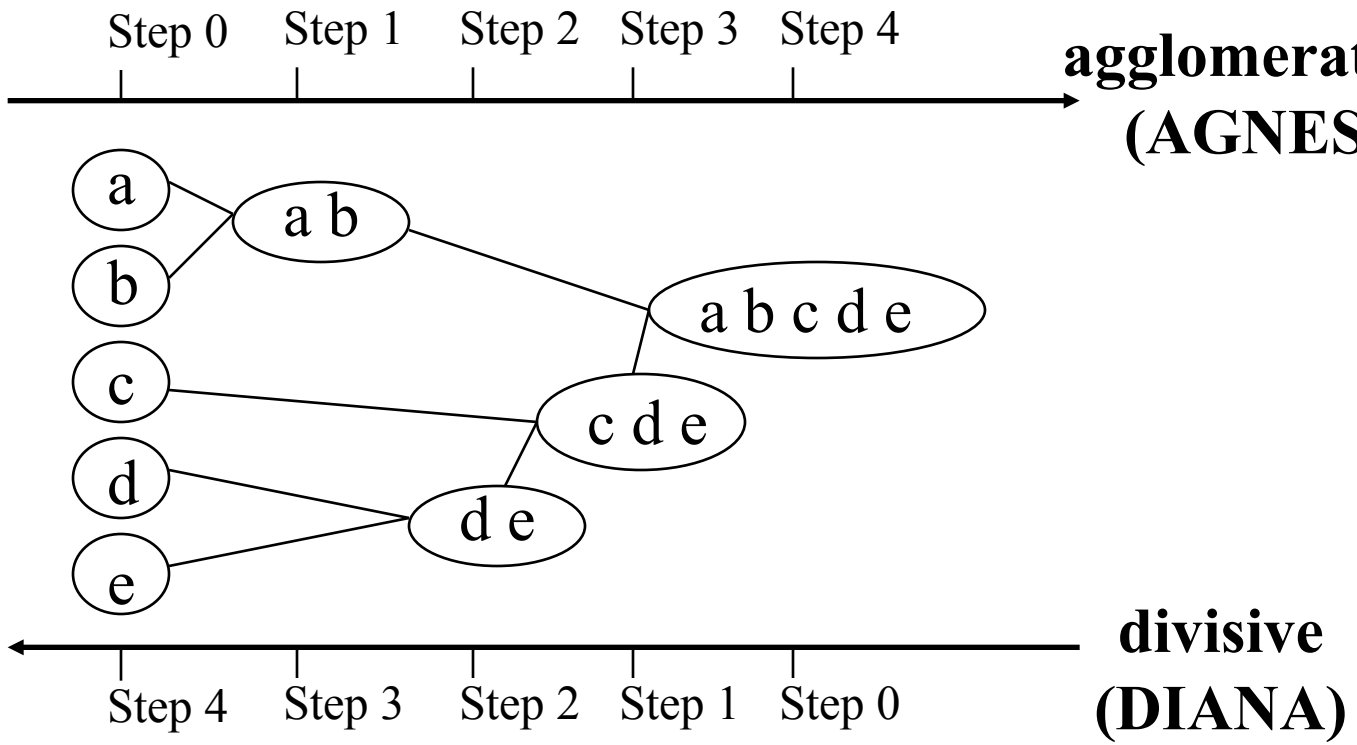
层次的聚类方法将数据对象组成一棵聚类树，通过在期望的水平上切割来完成聚类。





基于层次的聚类

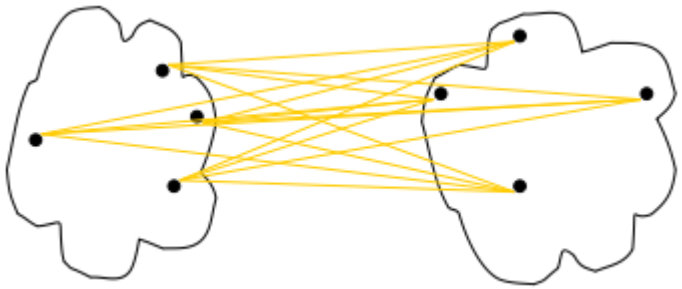
根据层次分解是自底向上, 还是自顶向下形成,
层次的聚类方法可以进一步分为凝聚的(agglomerative)和分裂的(divisive)层次聚类 。



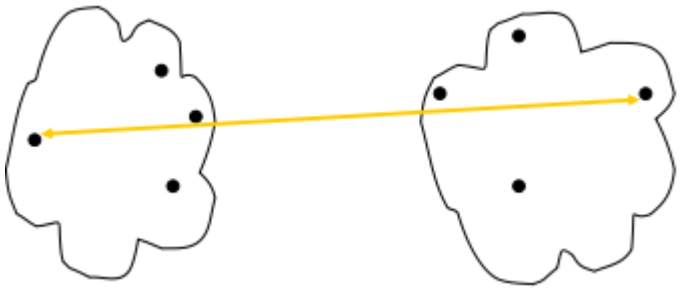
层次凝聚的代表是AGNES算法。

层次分裂的代表是DIANA算法。

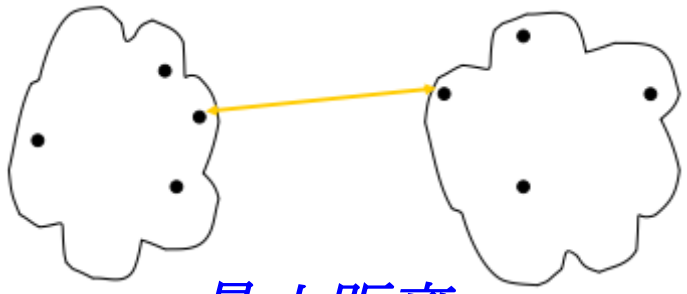
簇间距



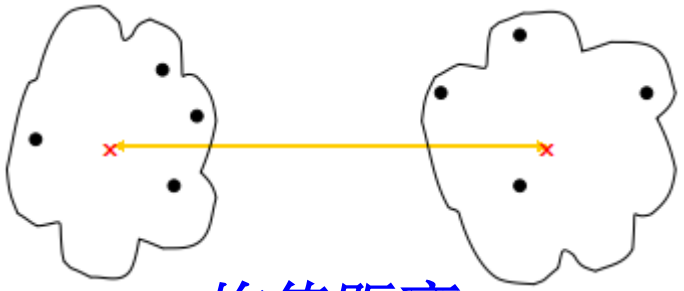
平均距离



最大距离



最小距离



均值距离

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：$d_{max}(C_i,C_j) = max_{\ p\in Ci,\ p'\in Cj} |p-p'|$

均值的距离：$d_{mean}(C_i,C_j) = | m_i - m_j |$

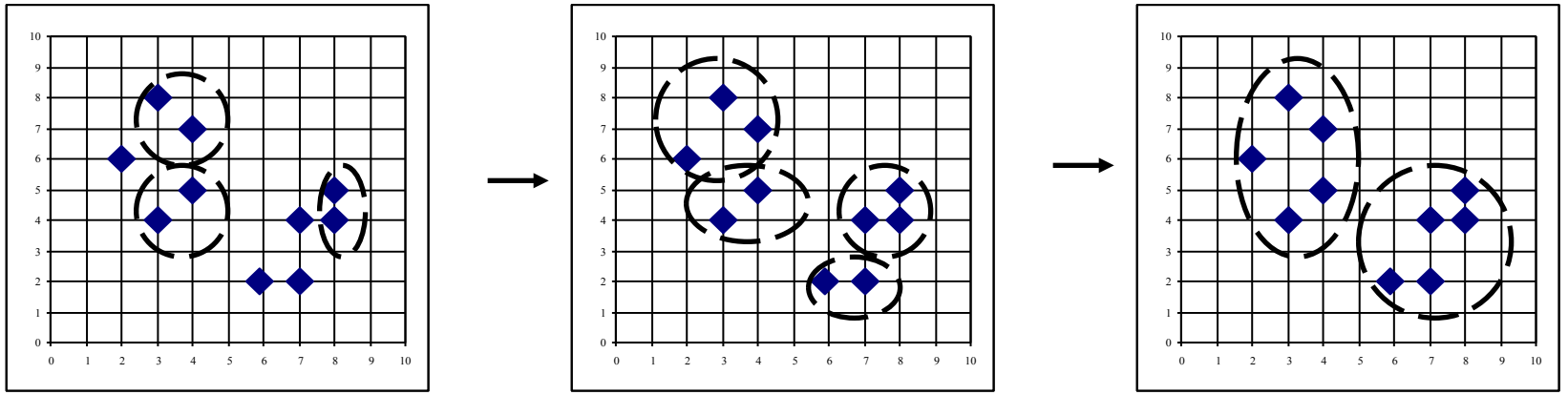
平均距离：$d_{avg}(C_i,C_j) = \sum_{p\in Ci} \sum_{p'\in Cj} |p-p'| / ninj$

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

AGNES (Agglomerative Nesting)

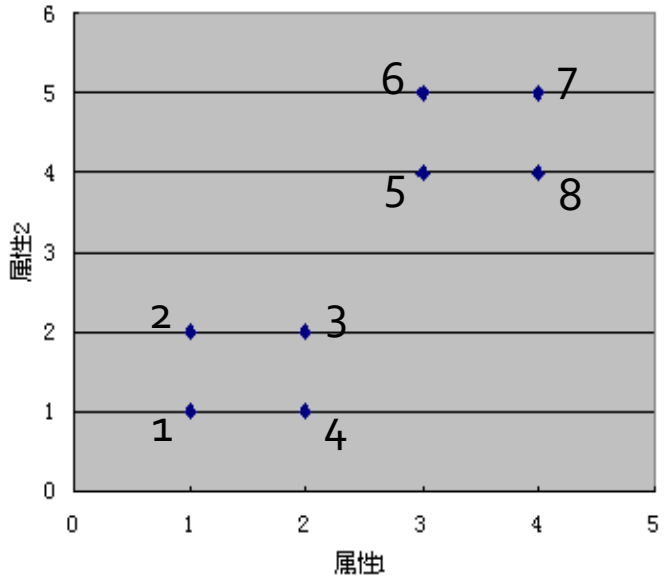
- 由 Kaufmann和Rousseeuw提出(1990)
- 两个簇间的相似度由这两个不同簇中**距离最近**的数据点对的相似度来确定。
- 聚类的合并过程反复进行直到所有的对象最终满足簇数目。



按层次逐渐凝聚

AGNES

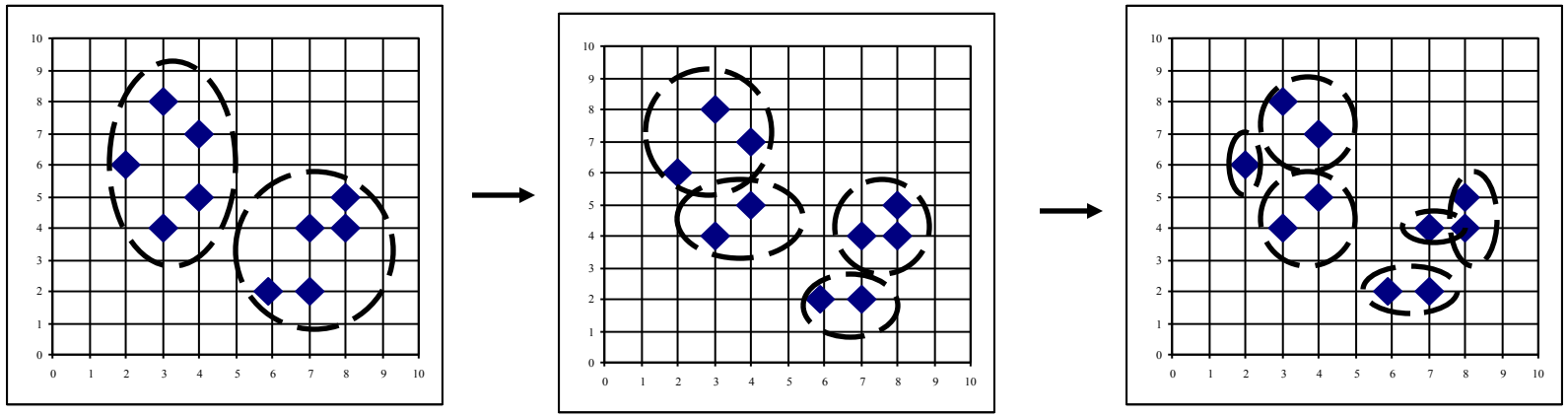
- 输入：n个对象，终止条件簇的数目k。
- 输出：k个簇，达到终止条件规定簇数目。



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

DIANA (Divisive Analysis)

- 由 Kaufmann和Rousseeuw提出 (1990)
- 是 AGNES的逆过程
- 最终每个节点自己形成一个簇

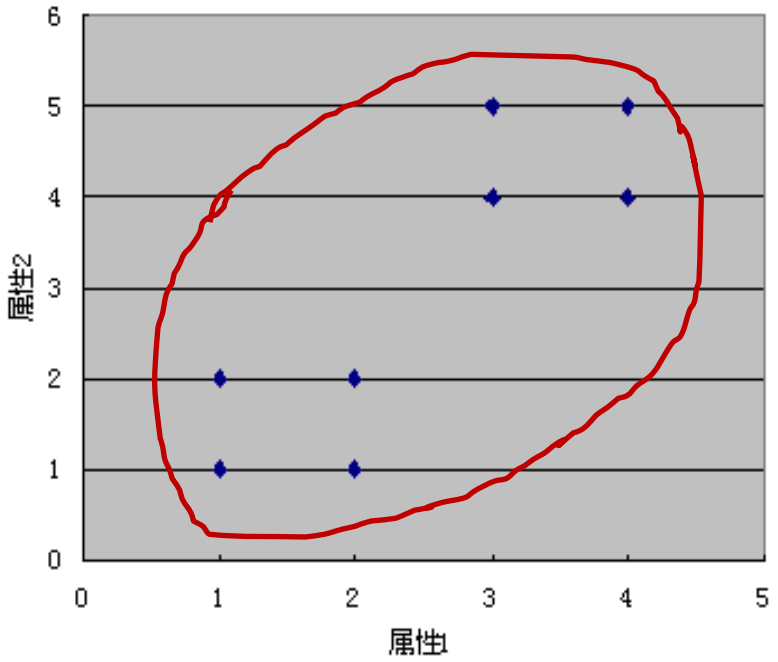


按层次逐渐分裂

DIANA

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）。

簇的直径：簇中任意两点之间的最大距离



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

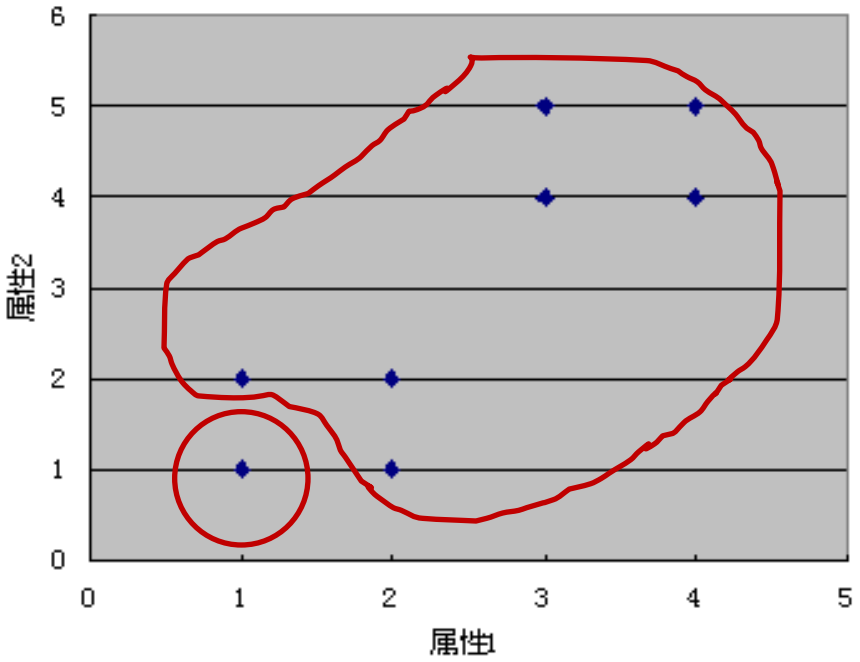
类似地，2的平均距离为2.526；3的平均距离为2.68；4的平均距离为2.18；5的平均距离为2.18；6的平均距离为2.68；7的平均距离为2.526；8的平均距离为2.96。

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

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

DIANA

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。

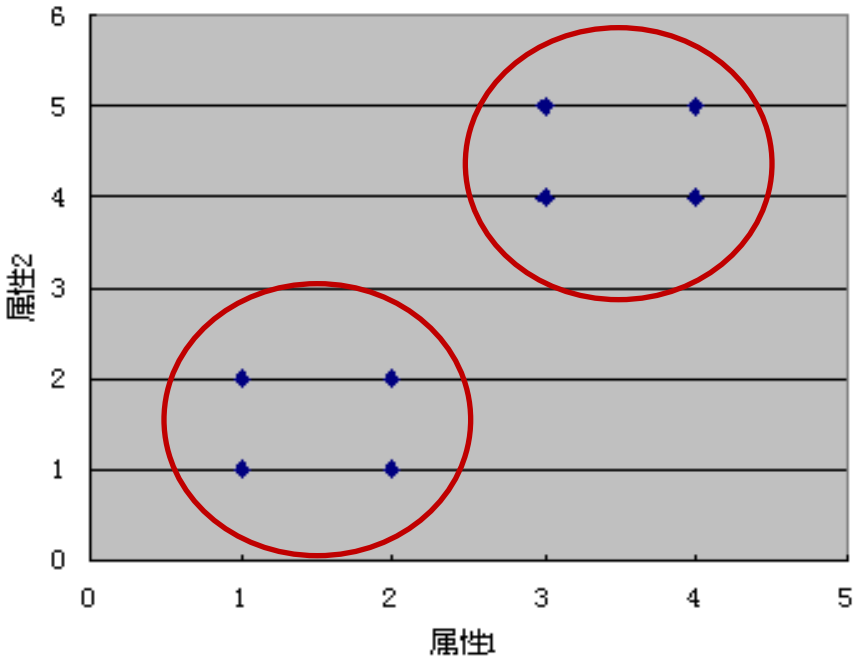


第3步，重复第2步的工作，splinter group中放入点3。
第4步，重复第2步的工作，splinter group中放入点4。

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

DIANA

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，应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 & DIANA的缺点

- 算法比较简单，但经常会遇到合并或者分裂点选择的困难。
- 合并或分裂的决定需要检查和估算大量的对象或簇，终止条件K需要设定
- 假如一旦一组对象被合并or分裂，下一步的处理将在新生成的簇上进行。已做处理不能撤销，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。

BIRCH (1996）利用层次结构的平衡迭代归约聚类
Balanced Iterative Reducing and Clustering using Hierarchies

- 利用层次方法的平衡迭代归约和聚类，该算法的特点是能利用有限的内存资源完成对大数据集的高质量的聚类，同时通过单遍扫描数据集能最小化I/O代价。
- 两个重要概念
  - 聚类特征(Clustering Feature, CF)
  - 聚类特征树(Clustering Feature Tree, CF树)

BIRCH · 聚类特征(Clustering Feature, CF)

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

聚类特征(CF)是一个三元组，给出聚类的信息的汇总描述

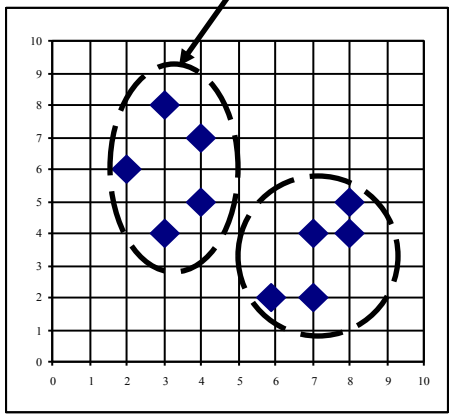
N: 数据点数目

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

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

N：数据点数目；LS：线性和； SS：平方和

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



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

BIRCH · 簇的质心C　簇的半径R

- 假如一个簇中包含n个数据点：{Xi}，i=1,2,3...n.，则质心C和半径R计算公式如下：
- C=(X1+X2+...+Xn)/n （这里X1+X2+...+Xn是向量加）
- R=(|X1-C|^2+|X2-C|^2+...+|Xn-C|^2)/n

  簇半径表示簇中所有点到簇质心的平均距离
- CF中存储的是**簇中所有数据点**的特性的统计和，所以当我们把一个数据点加入某个簇的时候，那么这个数据点的特征就可以丢弃了。由于此，BIRCH聚类可以在很大程度上对数据集进行压缩。

BIRCH · 簇的性质

- 簇半径、簇直径以及两簇之间的距离 都可以由***CF = (N, LS, SS)***直接计算，比如

- 簇直径 $D_2 = \sqrt{\frac{2N * SS - 2LS^2}{N(N-1)}}$

- 簇间距离 $D = \sqrt{\frac{SS_1}{N_1} + \frac{SS_2}{N_2} - \frac{2LS_1 LS_2}{N_1 N_2}}$

  这里的N，LS和SS是指两簇合并后大簇的N，LS和SS。

- 两簇合并只需要两个对应的CF相加即可

BIRCH ·聚类特征树(Clustering Feature Tree, CF树)

CF 树是高度平衡的树，它存储了层次聚类的聚类特征 。

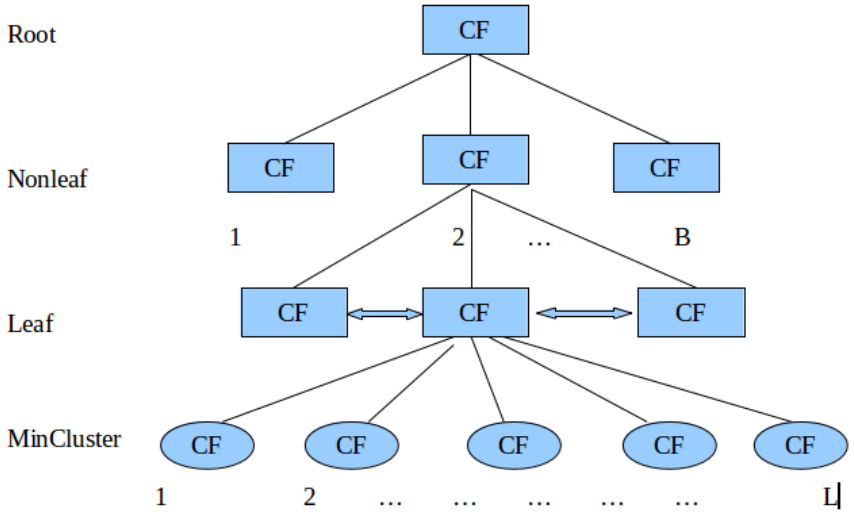
CF tree的结构类似于一棵B树，它有3个参数：

- **内部节点平衡因子B，**
- **叶节点平衡因子L，**
- **簇直径阈值T。**

非叶节点最多包含B个子节点

叶节点最多只能有L个MinCluster（初始划分子簇）

MinCluster的直径不能超过T。



B为6，L为5的一棵CF树

BIRCH

BIRCH是一种多阶段可伸缩聚类技术：数据集的单遍扫描产生一个基本的好聚类，一或多遍的额外扫描可以用来进一步（优化）改进聚类质量。它主要包括两个阶段：

- 阶段一：BIRCH扫描数据库，建立一棵存放于内存的初始CF树，它可以看作数据的多层压缩，试图保留数据的内在的聚类结构。当一个对象被插入，CF树被动态地构造，这样支持增量聚类。

- 阶段二：BIRCH采用某个（选定的）聚类算法对CF树的叶节点进行聚类，把稀疏的簇当作离群点删除，而把稠密的簇合并为更大的簇。一个对象被插入到最近的叶。如果在插入后，存储在叶节点中的子簇的直径大于阈值，则该叶节点和可能的其他节点被分裂。

BIRCH优缺点

- 优点
  - 支持增量聚类
  - 线性可伸缩性， 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
  - 簇非球形时效果不好(使用半径或直径的概念来控制簇的边界)

ROCK(Robust Clustering using links)

- 由S. Guha, R. Rastogi, K. Shim提出 (1999).
  - 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，可以处理对于聚类包含布尔或分类属性的数据。
- 基本思想:如果两个相似的点同时具有相似的邻域，那么这两个点可能属于同一个簇而合并。
  - 相似性函数:Jaccard系数 $Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$

设 $T_1 = \{1,2,3\}$, $T_2 = \{3,4,5\}$

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

ROCK(Robust Clustering using links)

- 包含分类属性数据的一个很好的例子就是购物数据。
  - 这种数据由事务数据库组成，其中每个事务都是商品的集合
  - 事务看作具有布尔属性的记录，每个属性对应于一个单独的商品，如面包或奶酪。
  - 如果一个事务包含某个商品，那么该事务的记录中对应于此商品的属性值就为真；否则为假。

ROCK(Robust Clustering using links)

- 假定一个购物数据库包含关于商品a, b, . . . , g。考虑这些事务（数据点）的两个簇C_1和C_2。

  - C_1涉及商品{a, b, c, d, e}，包含事务{a, b, c}，{a, b, d}，{a, b, e}，{a, c, d}，{a, c, e}，{a, d, e}，{b, c, d}，{b, c, e}，{b, d, e}，{c, d, e}
  - C_2涉及商品{a, b, f, g}，包含事务{a, b, f}，{a, b, g}，{a, f, g}，{b, f, g}

  - 假设我们首先只考虑点间的相似度而忽略邻域信息。C_1中事务{a, b, c}和{b, d, e}之间的Jaccard系数为1/5=0.2。
  - 事实上，C_1中任意一对事务之间的Jaccard系数都在0.2和0.5之间，而属于不同簇的两个事务之间的Jaccard系数也可能达到0.5。
  - 很明显，仅仅使用Jaccard系数本身，无法得到所期望的簇。

ROCK(Robust Clustering using links)

- ROCK基于链接的方法可以成功地把这些事务划分到恰当的簇中。
- 事实证明，对于每一个事务，与之链接最多的那个事务总是和它处于同一个簇中。例如，
  - 令 相似度阈值 θ = 0.5，则C_2中的事务{a, b, f}与同样来自同一簇中的事务{a, b, g}之间的链接数为5（因为它们有共同的近邻{a, b, c}，{a, b, d}，{a, b, e}，{a, f, g}和{b, f, g}）
  - 然而，C_2中的事务{b, f, g}与C_1中的事务{a, b, c}之间的链接数仅为3（其共同的邻居为{a, b, d}，{a, b, e}，{a, b, g}）
  - 类似地，C_2中的事务{a, f, g}与C_2中其他每个事务之间的链接数均为2，而与C_1中所有事务的链接数都为0。因此，这种基于链接的方法能够正确地区分出两个不同的事务簇，因为它除了考虑对象间的相似度之外还考虑邻域信息。

CHAMELEON (Hierarchical clustering using dynamic modeling)

- 由G. Karypis, E.H. Han, and V. Kumar 1999
- Chameleon基于动态模型度量相似性

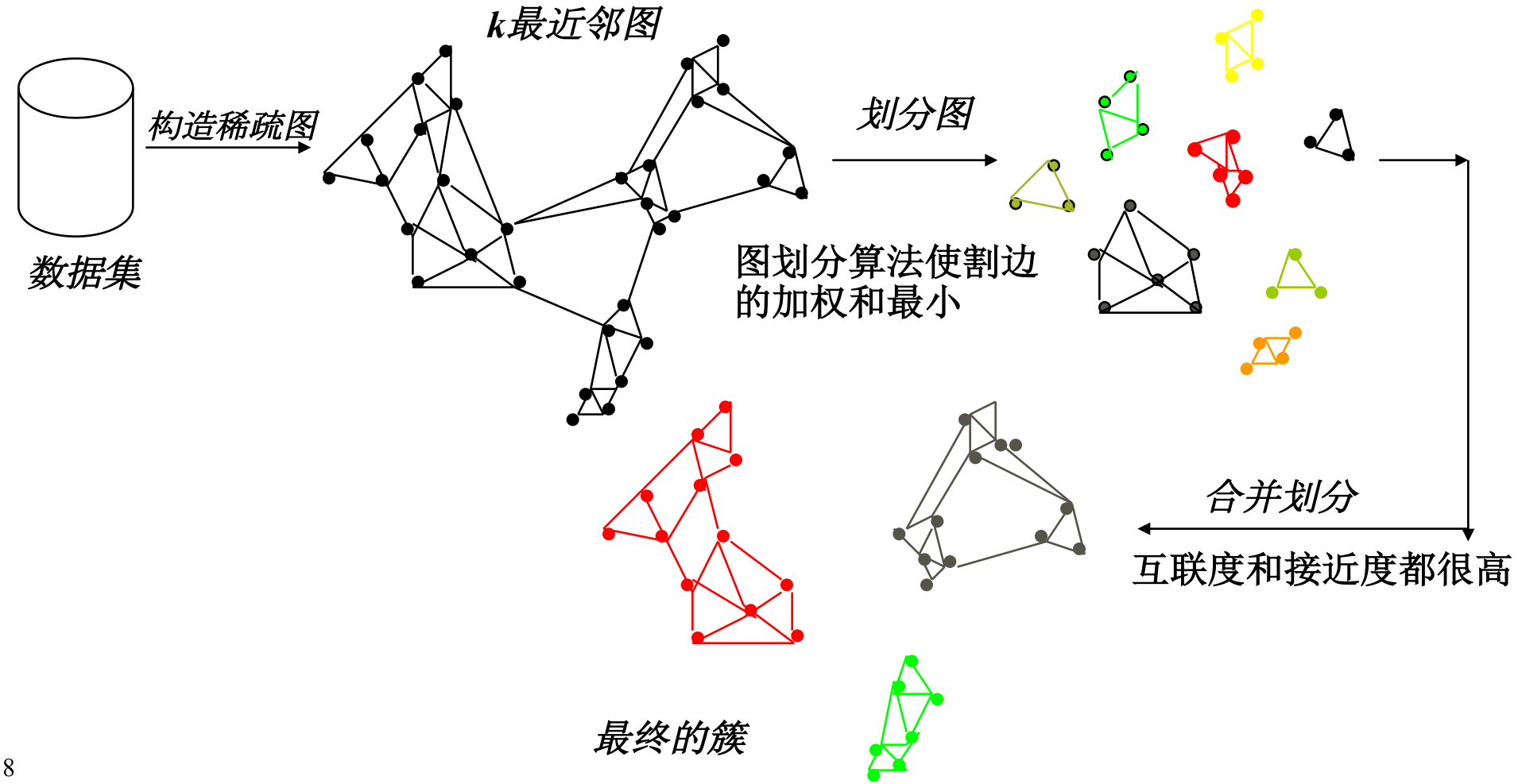
在Chameleon中，簇的相似度依据如下两点评估：

- 簇中对象的连接情况
- 簇的邻近性

也就是说，如果两个簇的互连性都很高并且它们又靠的很近就将其合并。

Chameleon算法的思想是：

- 首先使用一种图划分算法将k最近邻图划分成大量相对较小的子簇。
- 然后使用凝聚层次聚类算法，基于子簇的相似度反复地合并子簇。



Chameleon · 相对互连

- Chameleon根据每对簇C_i和C_j的相对互连度RI（C_i，C_j）和相对接近度RC（C_i，C_j）来决定它们之间的相似度：
  - 两个簇C_i和C_j之间的相对互连度RI（C_i，C_j）定义为C_i和C_j之间的绝对互连度关于两个簇C_i和C_j的内部互连度的规范化，即

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

其中 $EC_{\{Ci,Cj\}}$ 是包含C_i和C_j的簇的割边，如上面所定义。类似地，EC_{Ci}(或EC_{Cj})是将C_i(或C_j)划分成大致相等的两部分的割边的最小和。

Chameleon ·相对互连

- 两个簇C_i和C_j的的相对接近度RC（C_i，C_j）定义为C_i和C_j之间的绝对接近度关于两个簇C_i和C_j的内部接近度的规范化，定义如下：

$$RC(C_i, C_j) = \frac{\bar{S}_{EC\{C_i,C_j\}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{ECC_i} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{ECC_j}}$$

其中$\bar{S}_{EC\{C_i,C_j\}}$是连接C_i中顶点和C_j中顶点的边的平均权重，$\bar{S}_{ECC_i}$ (或$\bar{S}_{ECC_j}$)是最小二分簇C_i(或C_j)的边的平均权重。

Chameleon优缺点

- 优点
  - 与一些著名的算法（如BIRCH和基于密度的DBSCAN）相比，Chameleon在发现高质量的任意形状的簇方面具有很强的能力
- 缺点
  - 在最坏的情况下，高维数据的处理代价可能对n个对象需要 $O(n^2)$ 的时间。

层次聚类总结

基于层次的聚类方法可以用来过滤噪声孤立点数据，发现任意形状的簇。

- AGENES：凝聚层次聚类算法
- DIANA：分裂层次聚类算法
- BIRCH: 聚类特征，聚类特征树
- ROCK：近邻链接数，处理布尔型数据
- Chameleon：相对互连，相对接近

主要特点：

- 将聚类对象组成聚类树，发现聚类的层次关系
- 有凝聚和分裂，较多算法为凝聚

实践

- 密度聚类
  - DBSCAN
    https://blog.csdn.net/haiyang_duan/article/details/77978932
  - OPTICS
    https://blog.csdn.net/ann_hp/article/details/51155845
  - DENCLUE
    http://www.pudn.com/Download/item/id/3288980.html
- 层次聚类
  - AGNES & DIANA
    https://blog.csdn.net/u014028027/article/details/72211395
    http://www.pudn.com/Download/item/id/1113279.html
  - BIRCH
    https://blog.csdn.net/luanpeng825485697/article/details/79822524
  - ROCK
    https://blog.csdn.net/neuqxzy/article/details/44452901
  - Chameleon
    https://www.cnblogs.com/zhangchaoyang/articles/2182752.html

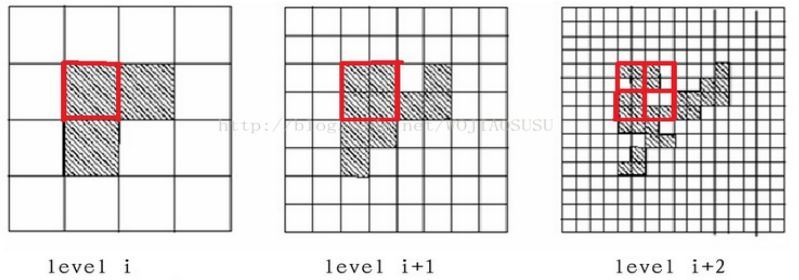
了解内容

其他聚类方法

基于网格的方法（grid-based method）

基于网格的方法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。

基于网格的聚类算法主要有STING, WaveCluster, CLIQUE等。



算法	描述
STING：	利用网格单元保存数据统计信息，从而实现多分辨率的聚类
WaveCluster：	在聚类分析中引入了小波变换的原理，主要应用于信号处理领域。
CLIQUE：	是一种结合了网格和密度的聚类算法

基于模型的方法（model-based method）

每个簇假定了一个模型，寻找数据对给定模型的最佳拟合。基于模型聚类方法主要有两种：统计学方法和神经网络方法。

算法	描述
COBWeb：	一个通用的概念聚类方法，它用分类树的形式表现层次聚类
AutoClass：	是以概率混合模型为基础，利用属性的概率分布来描述聚类，该方法能够处理混合型的数据，但要求各属性相互独立
算法	**描述**
自组织神经网络SOM：	该方法的基本思想是由外界输入不同的样本到人工的自组织映射网络中，一开始时，输入样本引起输出兴奋细胞的位置各不相同，但自组织后会形成一些细胞群，它们分别代表了输入样本，反映了输入样本的特征

算法名称	可伸缩性	适合的数据类型	高维性	异常数据的抗干扰性	聚类形状	算法效率
WaveCluster	很高	数值型	很高	较高	任意形状	很高
ROCK	很高	混合型	很高	很高	任意形状	一般
BIRCH	较高	数值型	较低	较低	球形	很高
CURE	较高	数值型	一般	很高	任意形状	较高
K-Prototypes	一般	混合型	较低	较低	任意形状	一般
DENCLUE	较低	数值型	较高	一般	任意形状	较高
OptiGrid	一般	数值型	较高	一般	任意形状	一般
CLIQUE	较高	数值型	较高	较高	任意形状	较低
DBSCAN	一般	数值型	较低	较高	任意形状	一般
CLARANS	较低	数值型	较低	较高	球形	较低

谢 谢