信息检索导论

An Introduction to Information Retrieval

第15讲 层次聚类

Hierarchical Clustering

授课人：古晓艳

中国科学院信息工程研究所/国科大网络空间安全学院

*改编自"An introduction to Information retrieval"网上公开的课件，地址 http://nlp.stanford.edu/IR-book/

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

聚类(Clustering)的定义

- (文档)聚类是将一系列文档按照相似性聚团成子集或者簇(cluster)的过程
- 簇内文档之间应该彼此相似
- 簇间文档之间相似度不大
- 聚类是一种最常见的无监督学习(unsupervised learning)方法
- 无监督意味着没有已标注好的数据集

扁平算法

- 扁平算法将N篇文档划分成K个簇
- **输入**：给定一个文档集合及聚类结果簇的个数K
- **输出**：寻找一个划分将这个文档集合分成K个簇，该结果满足某个最优划分准则
- 全局优化：穷举所有的划分结果，从中选择最优的那个划分结果
  - 无法处理
- 高效的启发式方法: K-均值聚类算法

K- 均值算法

K-MEANS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS$(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)$
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

***O*(IKNM)**

***O*(KNM)**

***O*(NM)**

K-均值聚类算法的初始化

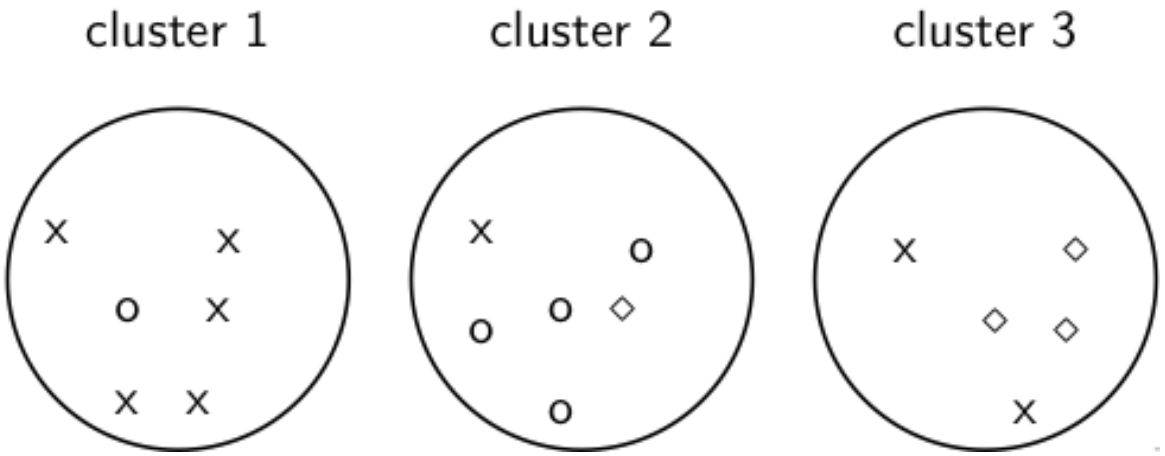
- 种子的随机选择只是K-均值聚类算法的一种初始化方法之一
- 随机选择不太鲁棒：可能会获得一个次优的聚类结果
- 一些确定初始质心向量的更好办法：
  - 非随机地采用某些启发式方法来选择种子(比如，过滤掉一些离群点，或者寻找具有较好文档空间覆盖度的种子集合)
  - 采用层级聚类算法寻找好的种子
  - 选择 i (比如 $i = 10$) 次不同的随机种子集合，对每次产生的随机种子集合运行K-均值聚类算法，最后选择具有最小RSS值的聚类结果
  - 为每个簇选出i个随机向量，将它们的质心向量作为该簇的种子向量

聚类评价的外部准则: 纯度

$$\text{purity}(\Omega, C) = \frac{1}{N}\sum_{k}\max_{j}|\omega_k \cap c_j|$$

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ 是簇的集合
- $C = \{c_1, c_2, \ldots, c_J\}$ 是类别的集合
- 对每个簇 ω_k : 找到一个类别c_j，该类别包含ω_k中的元素最多，为n_{kj} 个，也就是说ω_k的元素最多分布在c_j中
- 将所有 n_{kj} 求和，然后除以所有的文档数目

纯度计算的例子



为计算纯度

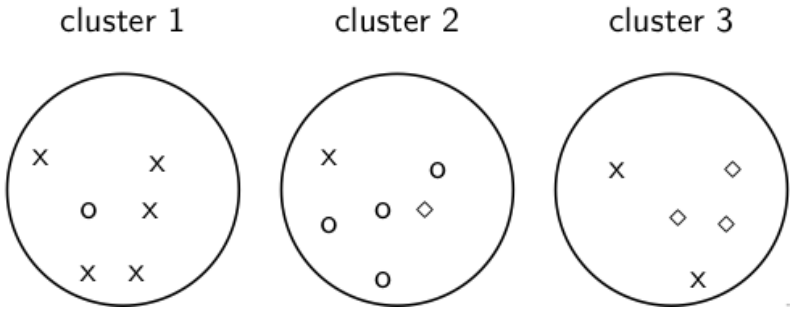
$\max_j |\omega_1 \cap c_j| = 5$ (class x, cluster 1);

$\max_j |\omega_2 \cap c_j| = 4$ (class o, cluster 2);

$\max_j |\omega_3 \cap c_j| = 3$ (class ◇, cluster 3)

纯度为 $(1/17) \times (5 + 4 + 3) \approx 0.71$.

兰迪指数(Rand index)



- 定义: $\mathrm{RI} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$
- 考虑所有两个文档之间(文档对)的关系，可以得到 2x2 的列联表：

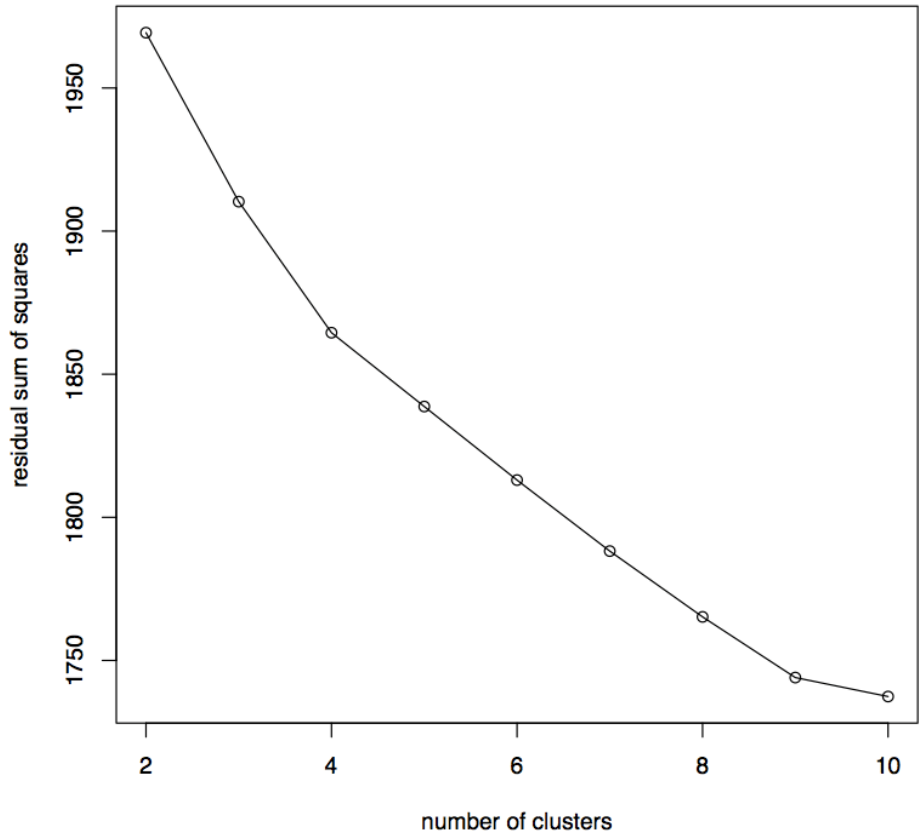
	same cluster	different clusters
same class	true positives (TP)	false negatives (FN)
different classes	false positives (FP)	true negatives (TN)

- 总的文档对数目为TP+FN+FP+TN
- 对于N篇文档，总共有$\binom{N}{2}$个文档对

回到上例，$\mathrm{TP}+\mathrm{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$

$$\mathrm{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

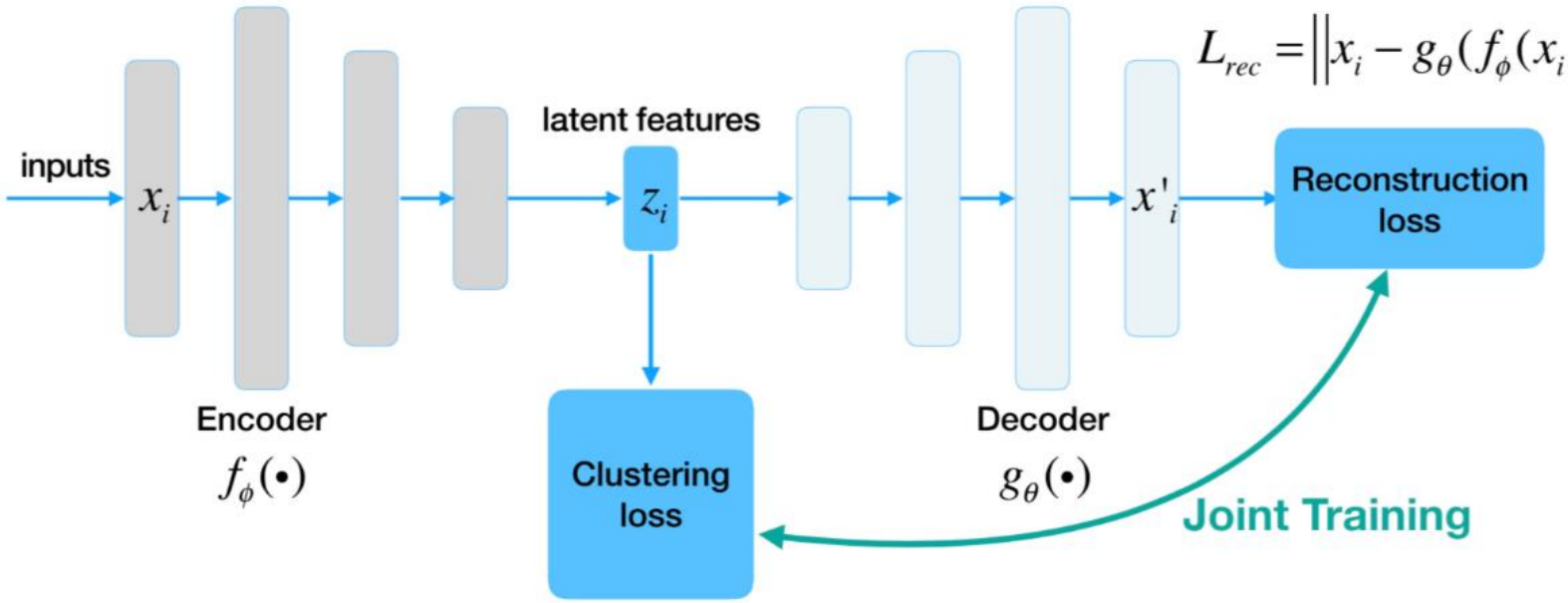
簇个数的确定



拐点的结果具有一定的代表性

本图中两个拐点：4 和 9

基于AE的深度聚类



- 代表性算法：
- Deep Clustering Network (DCN), ICML 2017;
- Deep Embedding Network (DEN), ICPR 2014;
- Deep Subspace Clustering Networks (DSC-Nets),NIPS 2017;
- Deep Multi-Manifold Clustering (DMC), AAAI 2017;
- Deep Embedded Regularized Clustering (DEPICT), ICCV 2017;
- Deep Continuous Clustering (DCC), arxiv 2018

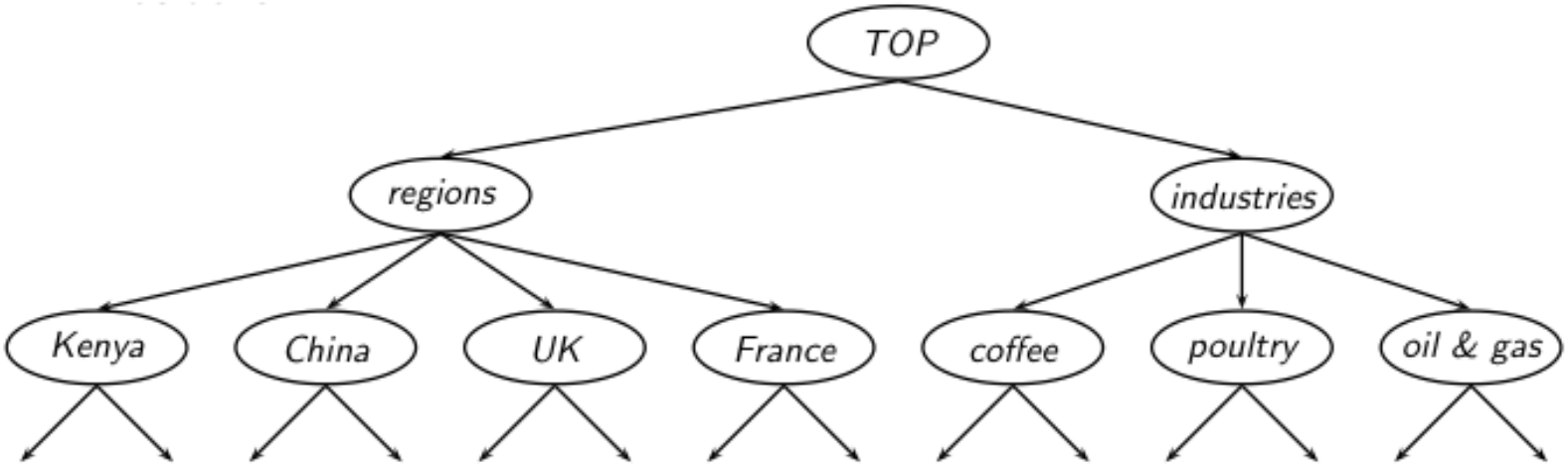
$$\min_{\substack{W,Z,\\ M,\{s_i\}}} \sum_{i=1}^{N} \left(\ell\left(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i\|_2^2 \right)$$

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

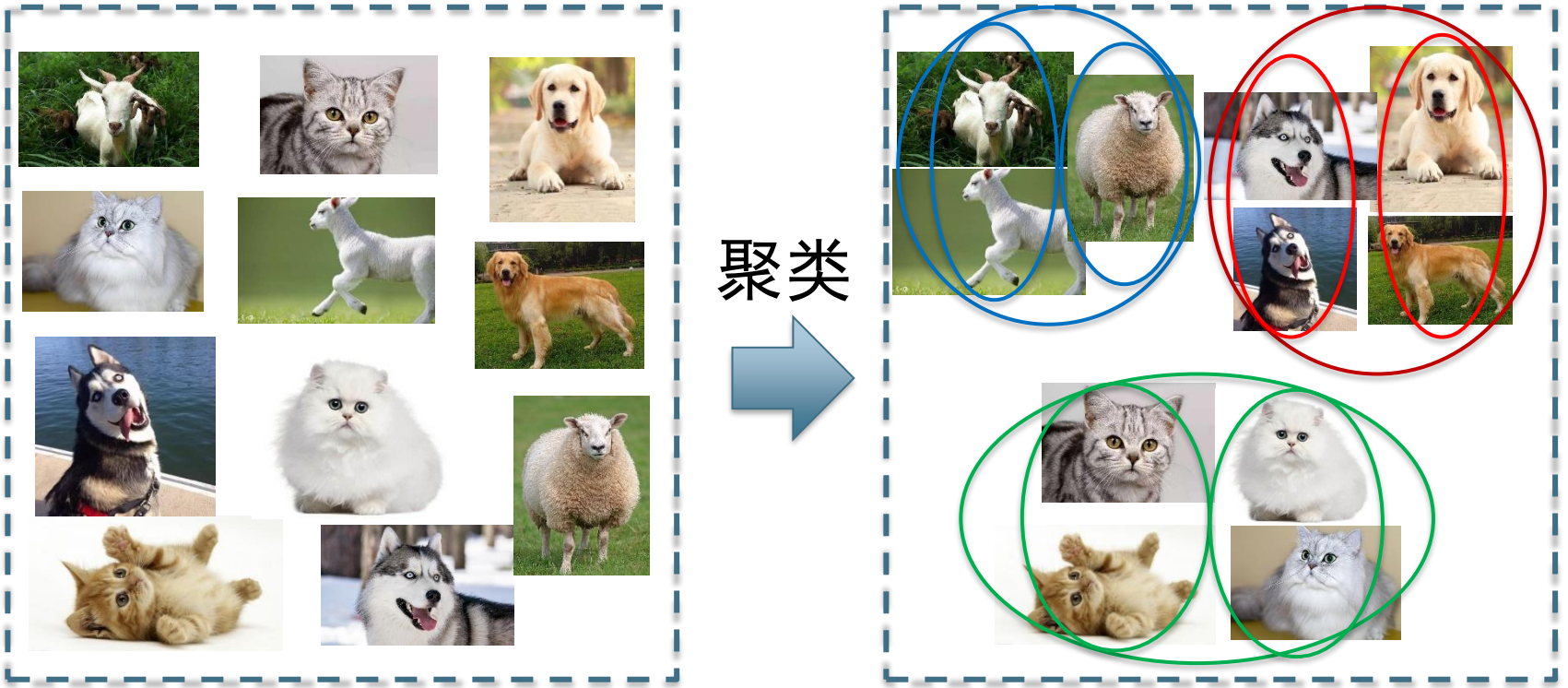
层次聚类

层次聚类的目标是生成类似于前面提到的Reuters目录的一个层次结构：



这个层次结构是自动创建的，可以通过自顶向下或自底向上的方法来实现。

层次聚类的作用



- 可得到聚类的层次结构
- 不需要事先定义簇的个数

按照层次结构浏览

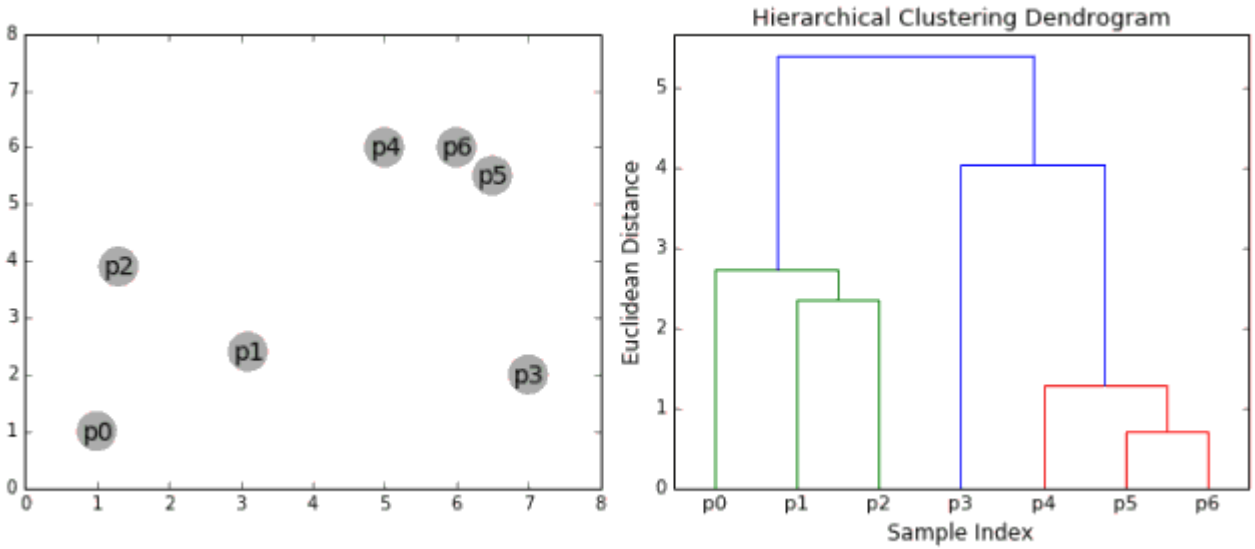

京东
火锅超级品类日 满299减150
二手手机节 家具9.9 车主福利 冬季防护 电脑自营 满99减10 二手手机
秒杀 优惠券 PLUS会员 品牌闪购 拍卖 京东家电 京东超市 京东生鲜
家用电器
手机/运营商/数码
电脑/办公
家居/家具/家装/厨具
男装/女装/童装/内衣
美妆/个护清洁/宠物
女鞋/箱包/钟表/珠宝
男鞋/运动/户外
房产/汽车/汽车用品
母婴/玩具乐器
食品/酒类/生鲜/特产
艺术/礼品鲜花/农资绿植
医药保健/计生情趣
图书/文娱/教育/电子书
机票/酒店/旅游/生活
理财/众筹/白条/保险
安装/维修/清洗/二手
家电馆 > 家电专卖店 > 家电服务 > 企业采购 > 商用电器 > 高价回收 >
电视 > 超薄电视 全面屏电视 智能电视 教育电视 OLED电视 智慧屏 4K超清电视 55英寸 65英寸 电视配件
空调 > 空调挂机 空调柜机 中央空调 变频空调 一级能效 移动空调 以旧换新
洗衣机 > 滚筒洗衣机 洗烘一体机 波轮洗衣机 迷你洗衣机 烘干机 洗衣机配件
冰箱 > 多门 对开门 三门 双门 冷柜/冰吧 酒柜 冰箱配件
厨卫大电 > 油烟机 燃气灶 烟灶套装 集成灶 消毒柜 洗碗机 电热水器 燃气热水器 空气能热水器 太阳能热水器 嵌入式厨电 烟机灶具配件
厨房小电 > 破壁机 电烤箱 电饭煲 电压力锅 电炖锅 豆浆机 料理机 咖啡机 电饼铛 榨汁机/原汁机 电水壶/热水瓶 微波炉 电热饭盒 电火锅 养生壶 电磁炉 面包机 空气炸锅 面条机 电陶炉 煮蛋器 电烧烤炉
生活电器 > 取暖器 空气净化器 吸尘器 除螨仪 扫地机器人 除湿机 干衣机 蒸汽拖把/拖地机 挂烫机/熨斗 电话机 饮水机 净水器 电风扇 冷风扇 加湿器 毛球修剪器 生活电器配件
个护健康 > 剃须刀 电动牙刷 电吹风 美容仪 洁面仪 按摩器 健康秤 卷/直发器 剃/脱毛器

层次凝聚式聚类 (HAC)

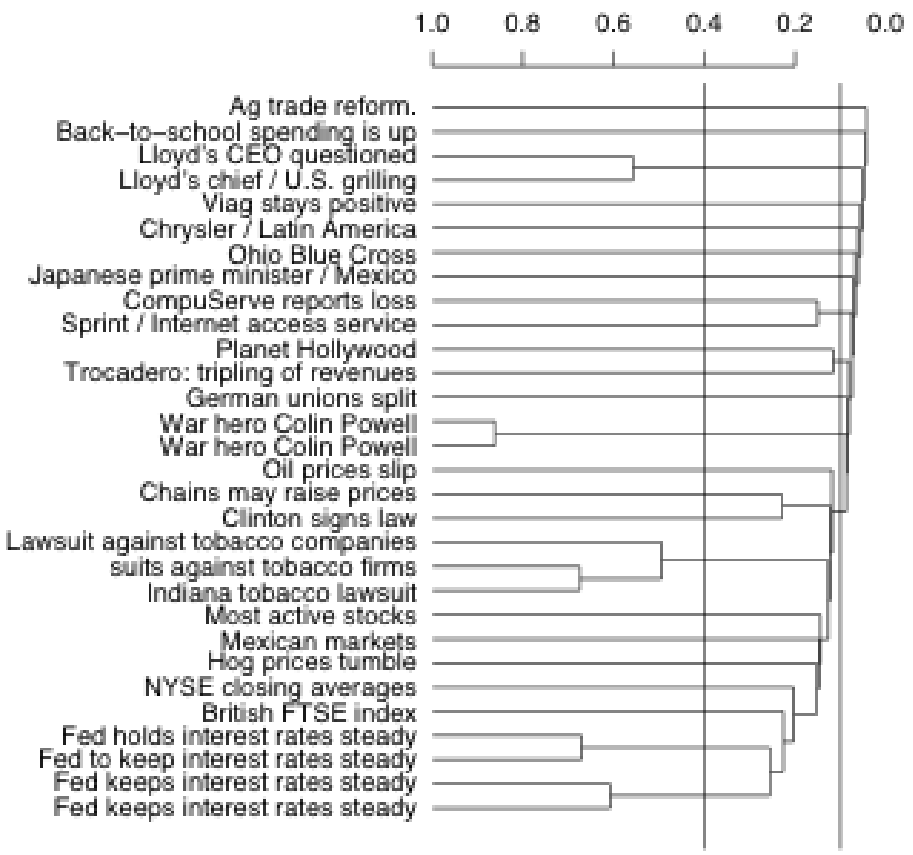
- 最著名的自底向上的方法是层次凝聚式聚类(hierarchical agglomerative clustering，HAC)。
- HAC会生成一棵二叉树形式的类别层次结构
- 到目前为止，我们的相似度都定义在文档之间，现在我们假设相似度定义在两个簇之间
- 接下来我们考察不同的簇相似度计算方法

层次凝聚式聚类 (HAC)

- 一开始每篇文档作为一个独立的簇
- 然后，将其中**最相似**的两个簇进行合并
- 重复上一步直至仅剩一个簇
- 整个合并的历史构成一个二叉树
- 一个标准的描述层次聚类合并历史的方法是采用树状图



树状图



24个 12个

- 合并的历史可以从底往上生成
- 水平线上给出的是每次合并的相似度
- 我们可以在特定点截断 (比如 0.1 或 0.4) 来获得一个扁平的聚类结果

分裂式聚类

- 分裂式聚类是从顶往下
- HAC (自底往上) 的一种替代形式
- 分裂式聚类：
  - 一开始所有文档聚成一类
  - 然后，不断迭代进行类别分割
  - 最终，每个节点构成一个类
- → Bisecting *K*-均值算法
- 接下来介绍 HAC

原始的HAC算法

SIMPLEHAC($d_1, \ldots, d_N$)

```
1  for n ← 1 to N
2  do for i ← 1 to N
3       do C[n][i] ← SIM(d_n, d_i)
4       I[n] ← 1  (keeps track of active clusters)
5  A ← []  (collects clustering as a sequence of merges)
6  for k ← 1 to N − 1
7  do ⟨i, m⟩ ← arg max_{⟨i,m⟩: i≠m ∧ I[i]=1 ∧ I[m]=1} C[i][m]
8       A.APPEND(⟨i, m⟩)  (store merge)
9       for j ← 1 to N
10      do  (use i as representative for < i, m >)
11           C[i][j] ← SIM(< i, m >, j)
12           C[j][i] ← SIM(< i, m >, j)
13      I[m] ← 0  (deactivate cluster)
14 return A
```

计算 $N \times N$ 个文档对之间的相似度

扫描$O(N \times N)$个相似度来寻找最大相似度

计算合并后的簇和先前其它簇之间的相似度

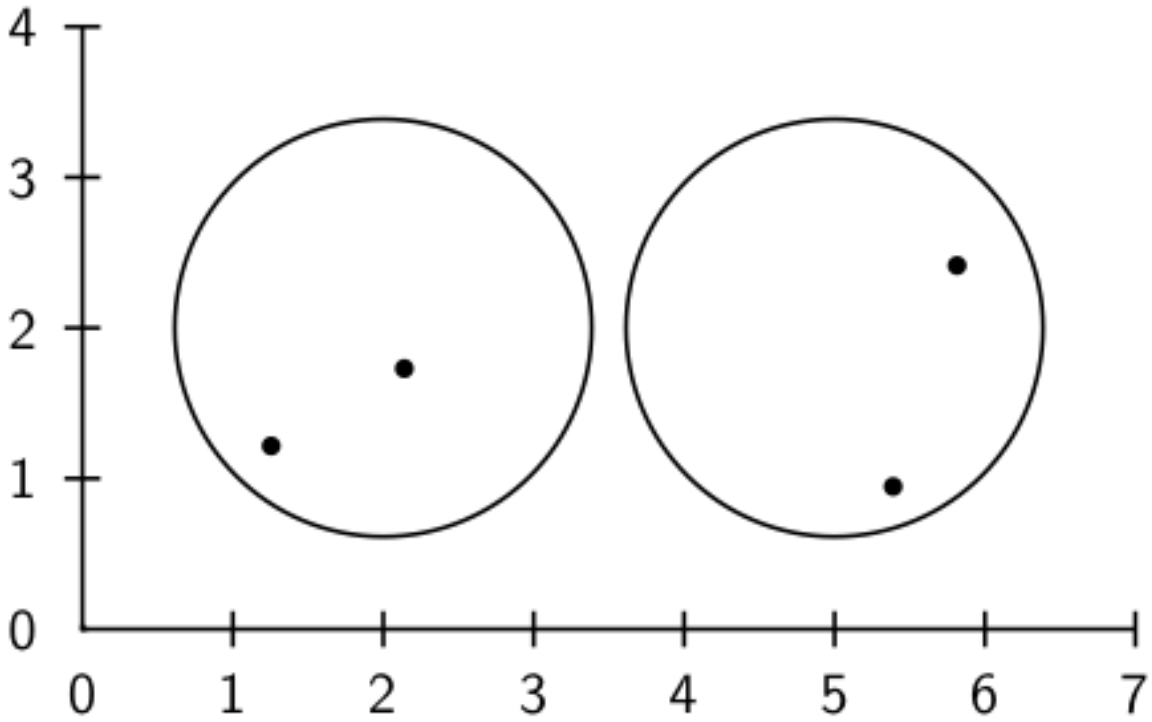
原始算法的计算复杂度

- 首先，我们需要计算 $N \times N$ 个文档对之间的相似度
- 其次，在N次迭代的每一次迭代中:
  - 扫描$O(N \times N)$ 个相似度来寻找最大相似度
  - 合并具有最大相似度的两个簇
  - 计算合并后的簇和先前其它簇之间的相似度
- 迭代次数是 $O(N)$，每次迭代需要 $O(N \times N)$ “扫描” 操作
- 整体复杂度为 $O(N^3)$.
- 后面我们将考察更高效的算法

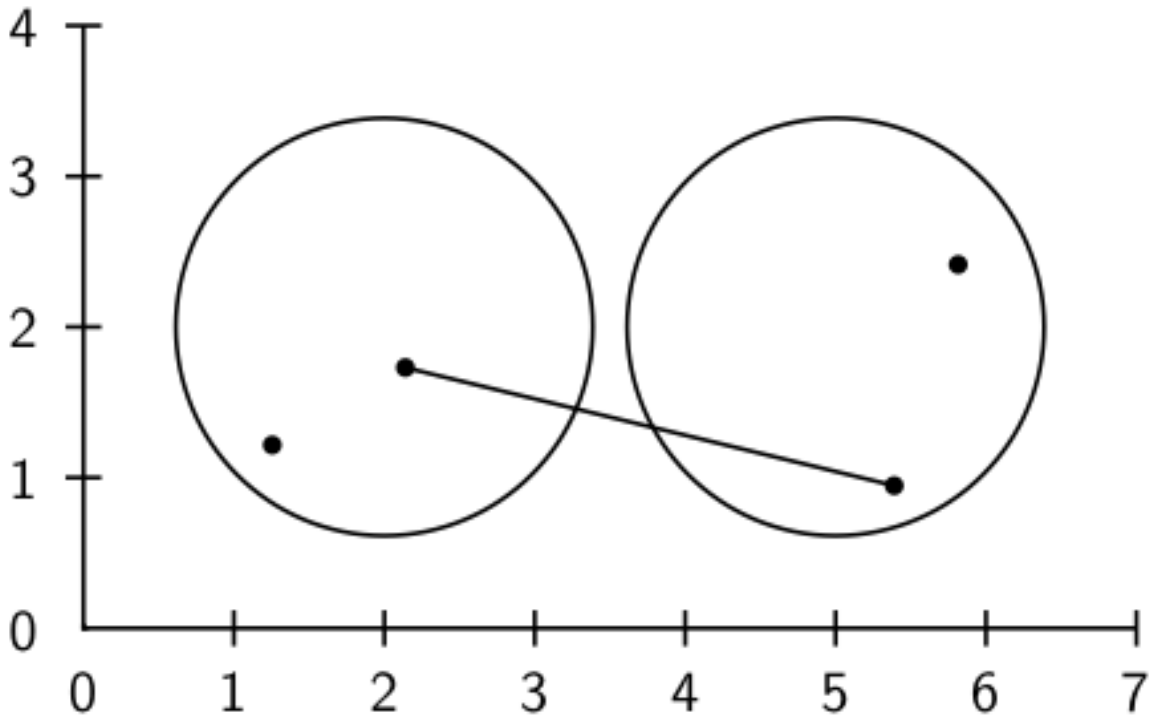
关键问题：如何定义簇相似度

- 单连接(Single-link): 最大相似度
  - 计算簇间任意两篇文档之间的相似度，取其中的最大值
- 全连接(Complete-link): 最小相似度
  - 计算簇间任意两篇文档之间的相似度，取其中的最小值
- 质心法: 平均的类间相似度
  - 所有的簇间文档对之间相似度的平均值 (不包括同一个簇内的文档之间的相似度)
  - 这等价于两个簇质心之间的相似度
- 组平均(Group-average): 平均的类内和类间相似度
  - 所有的簇间文档对之间相似度的平均值 (包括同一个簇内的文档之间的相似度)

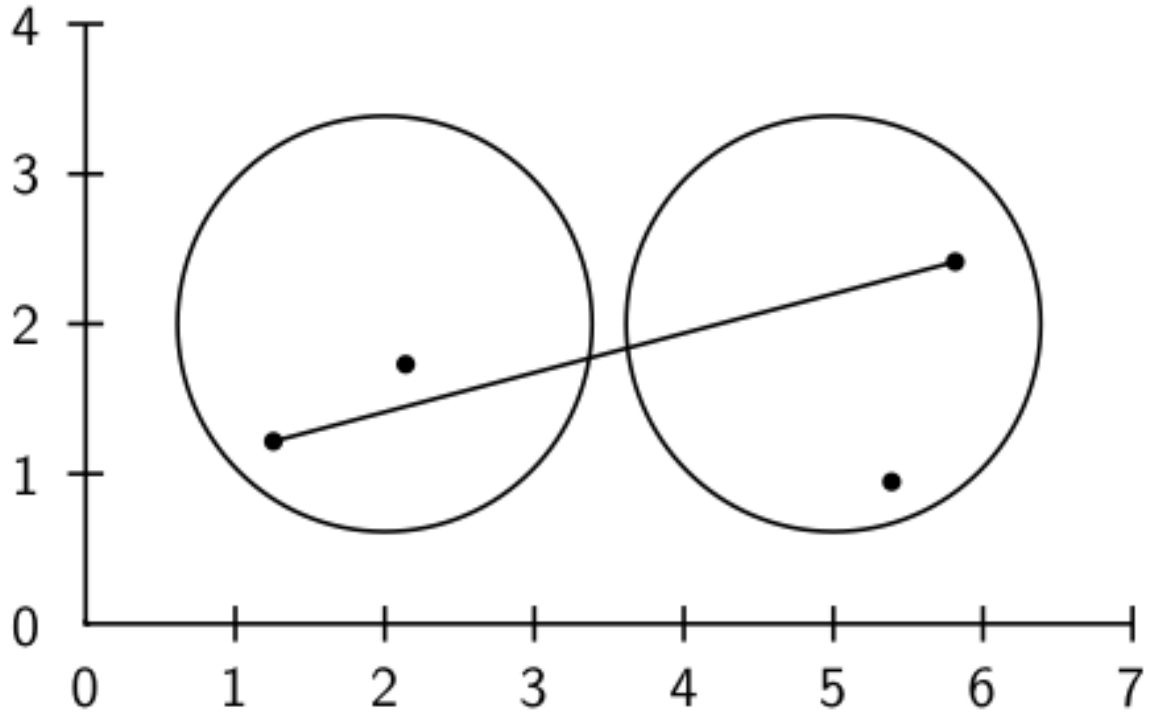
簇间相似度: 例子



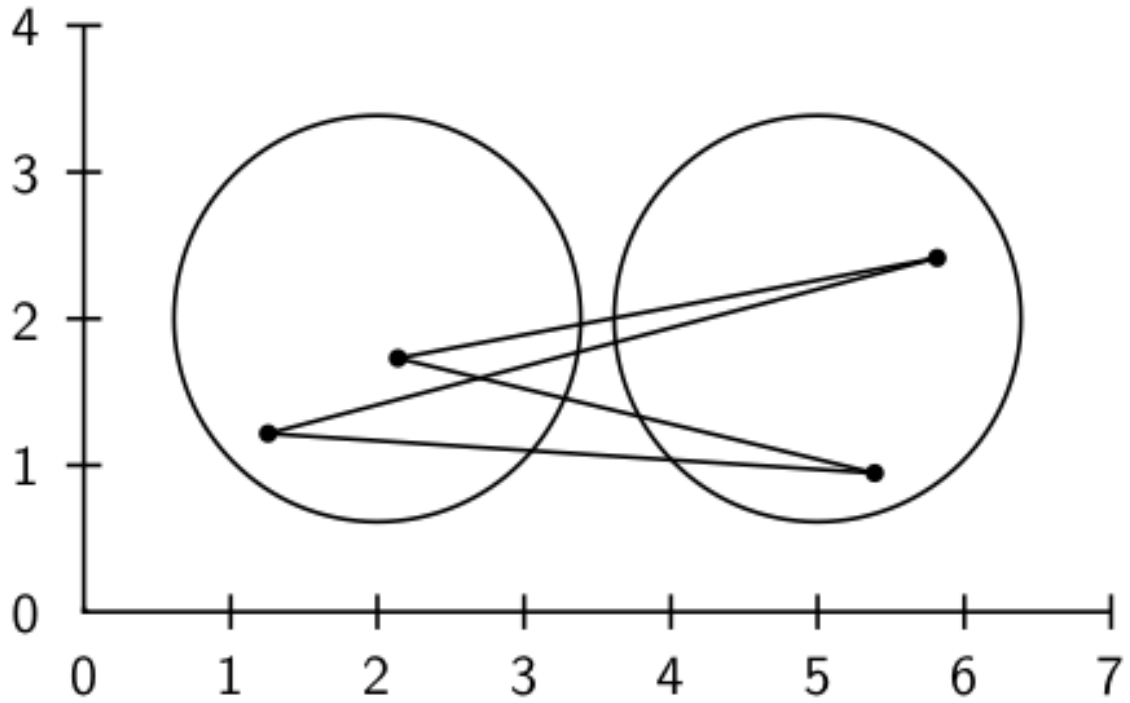
单连接: 最大相似度(最短距离)



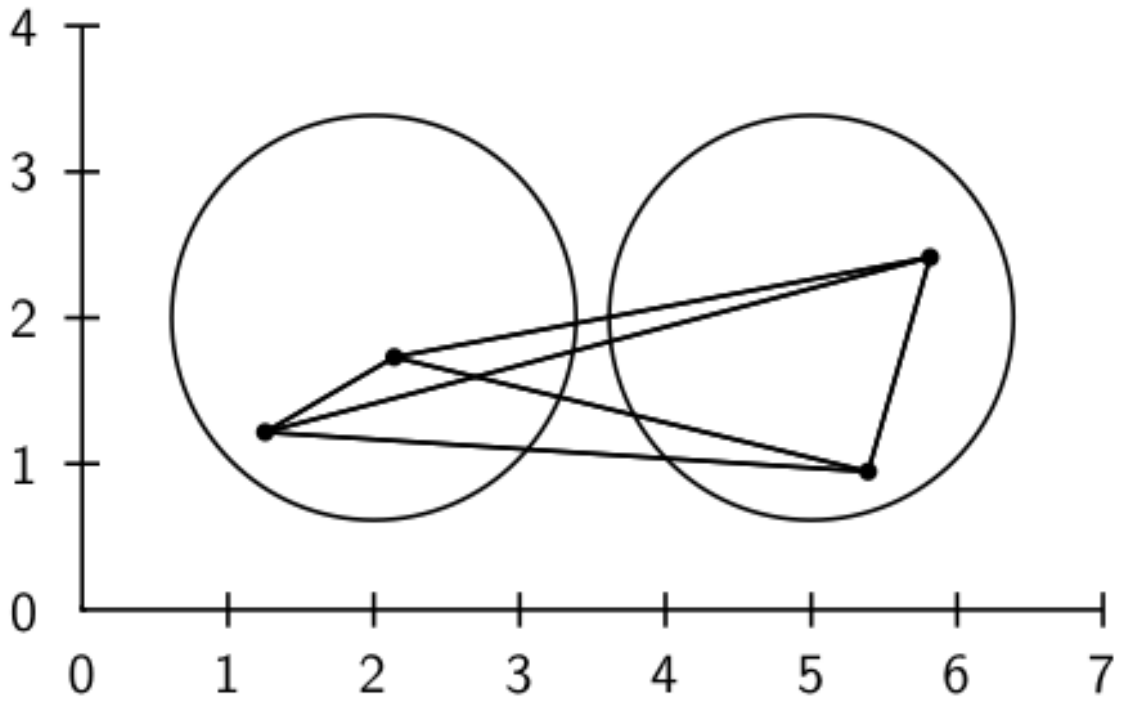
全连接: 最小相似度



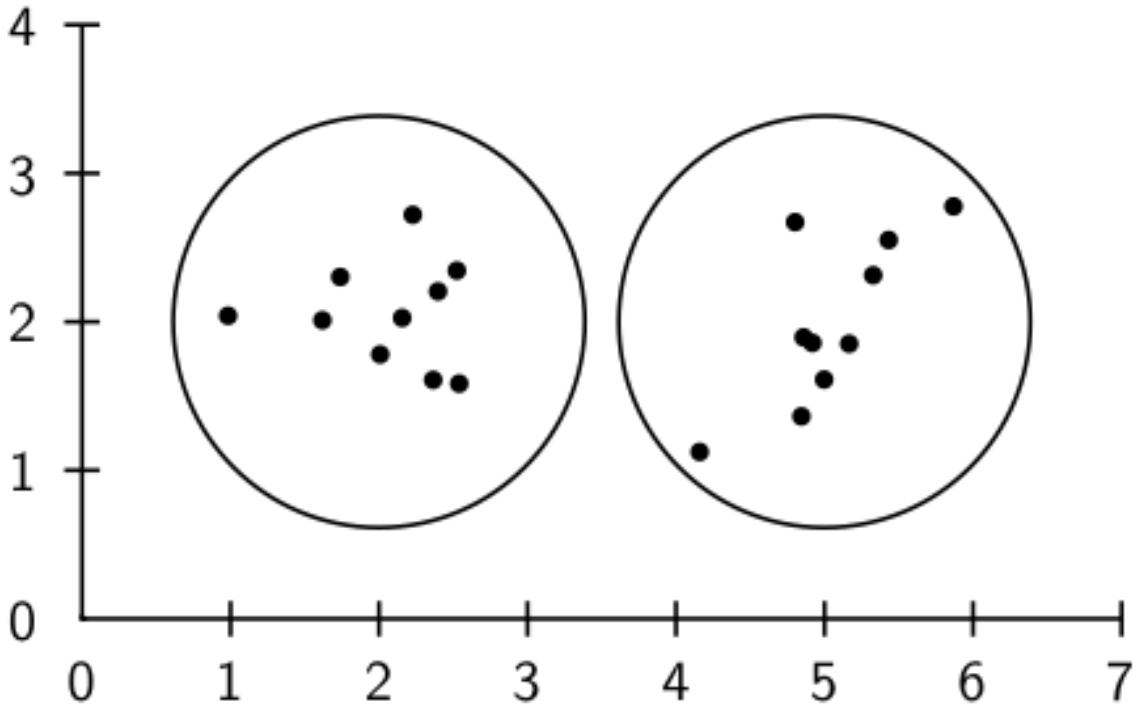
质心法



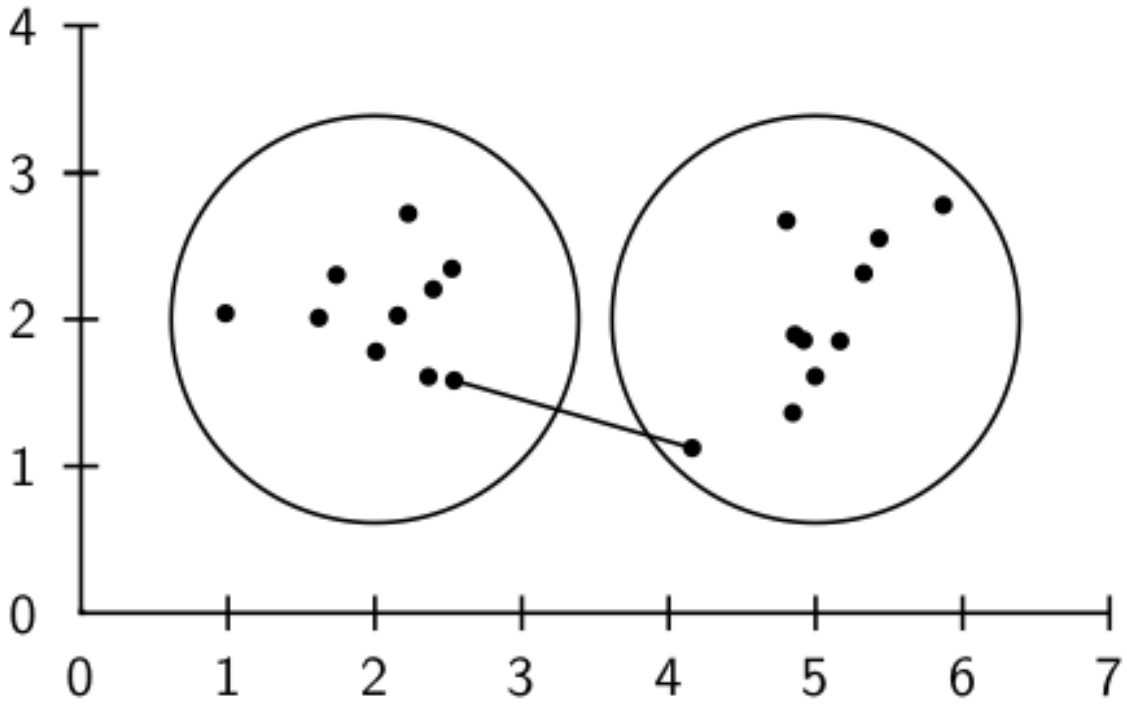
组平均



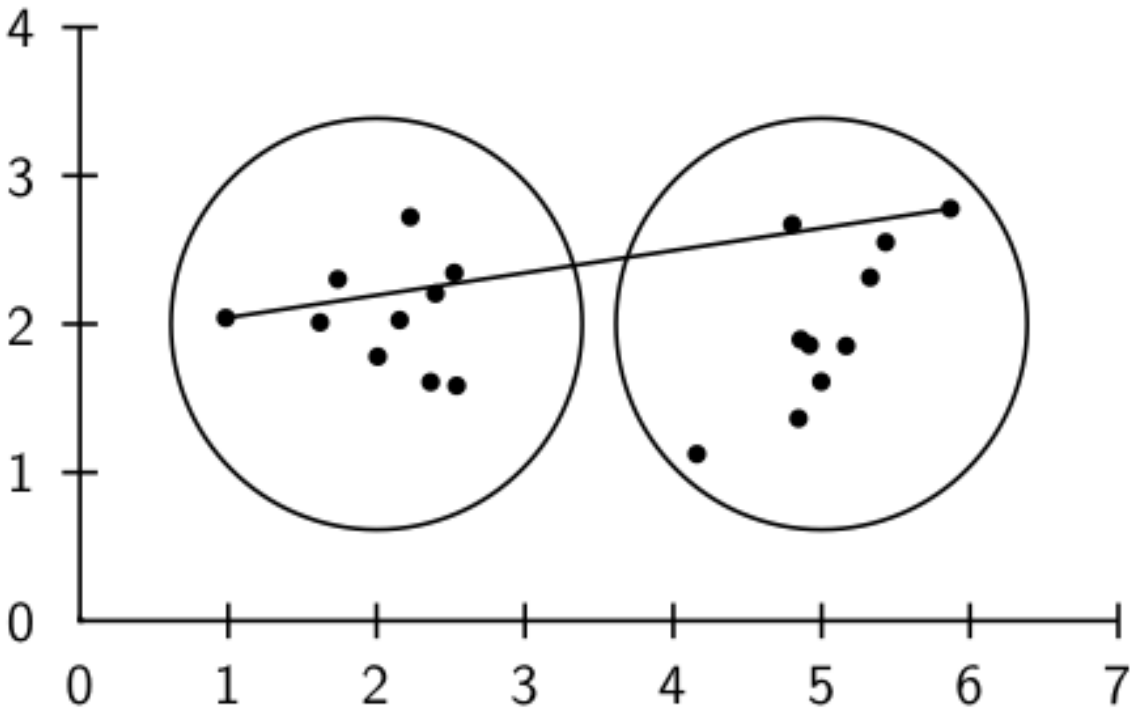
簇间相似度：一个更大的例子



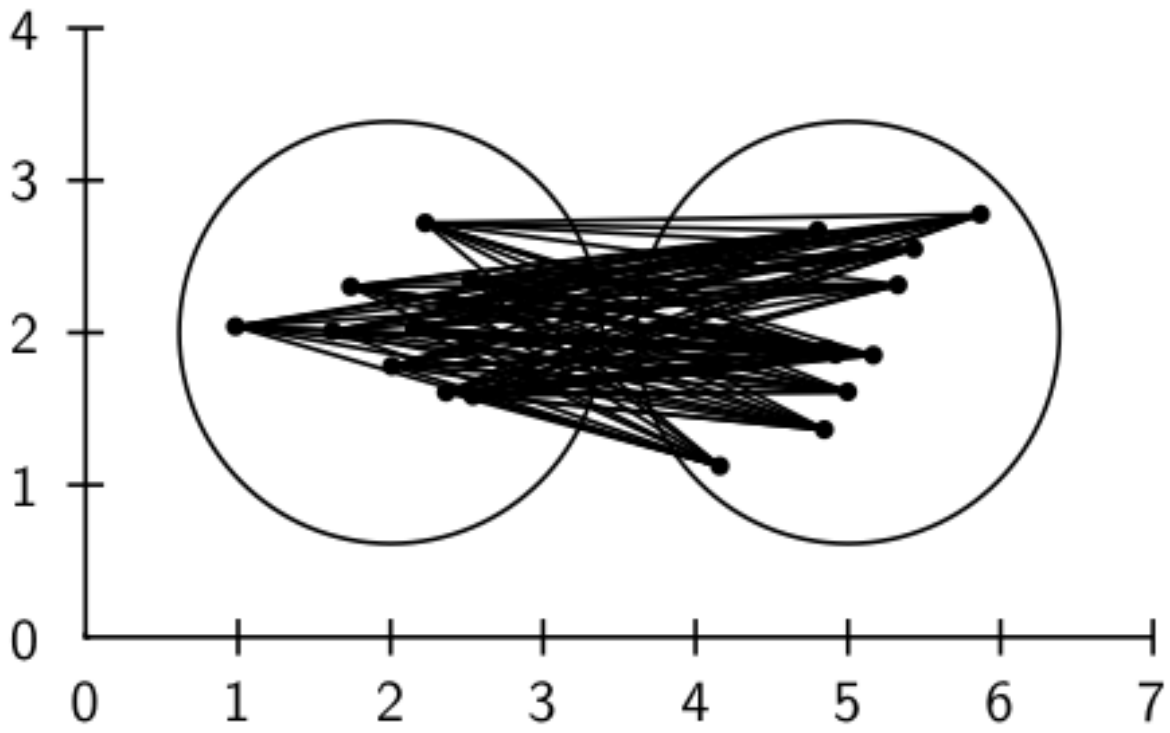
单连接法



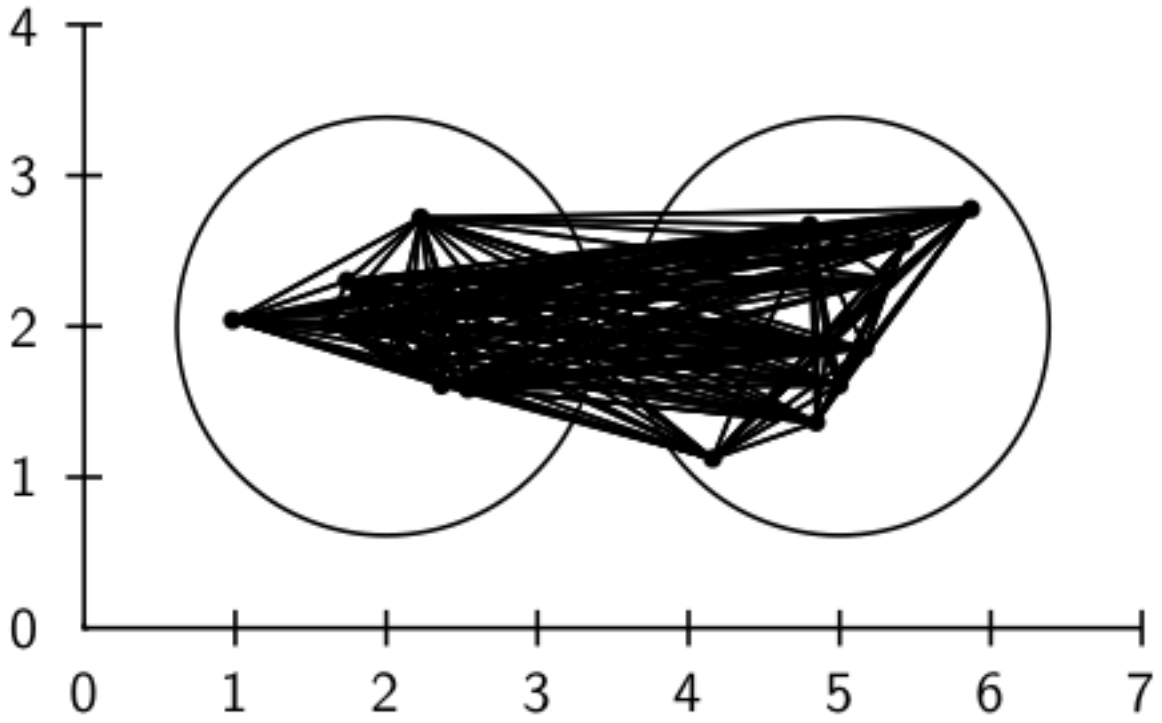
全连接法



质心法



组平均法



提纲

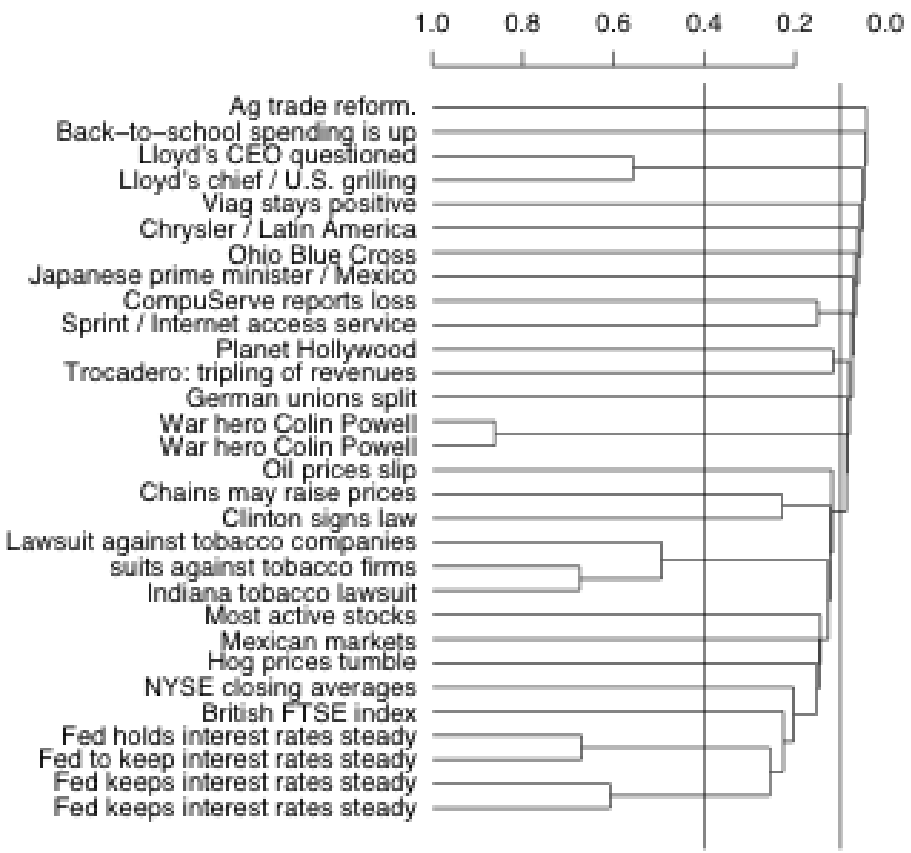
1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

单连接HAC

- 两个簇的相似度等于最大的类间相似度
- 一旦将两个簇合并，如何更新相似度矩阵？
- 对于单连接法来说，非常简单:

$$\text{SIM}(\omega_i, (\omega_{k1} \cup \omega_{k2})) = \max(\text{SIM}(\omega_i, \omega_{k1}), \text{SIM}(\omega_i, \omega_{k2}))$$

单连接算法产生的树状图



- 注意：很多很小的簇 (1 或 2 个成员) 加入到一个大的主簇上面去
- 不存在2个簇或者3个簇的非常均衡的结果

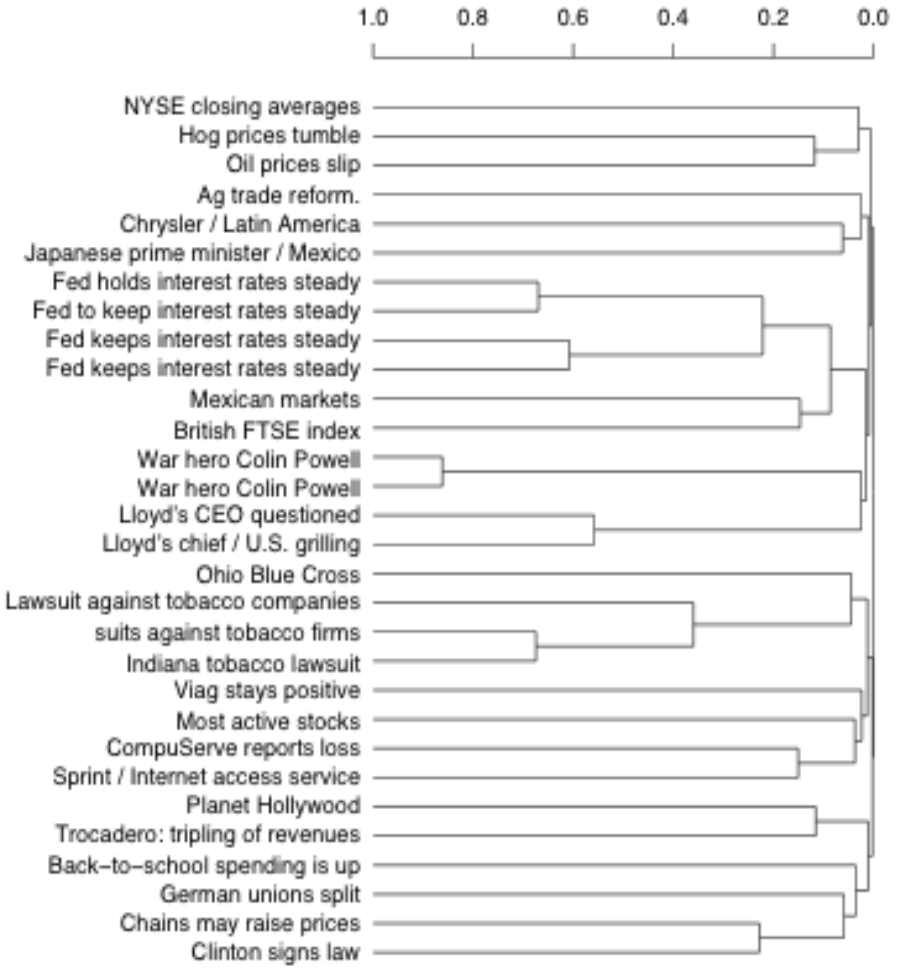
全连接HAC

- 两个簇的相似度等于最小的类间相似度
- 一旦将两个簇合并，如何更新相似度矩阵？
- 对于全连接法来说，也非常简单:

$$\text{SIM}(\omega_i, (\omega_{k1} \cup \omega_{k2})) = \min(\text{SIM}(\omega_i, \omega_{k1}), \text{SIM}(\omega_i, \omega_{k2}))$$

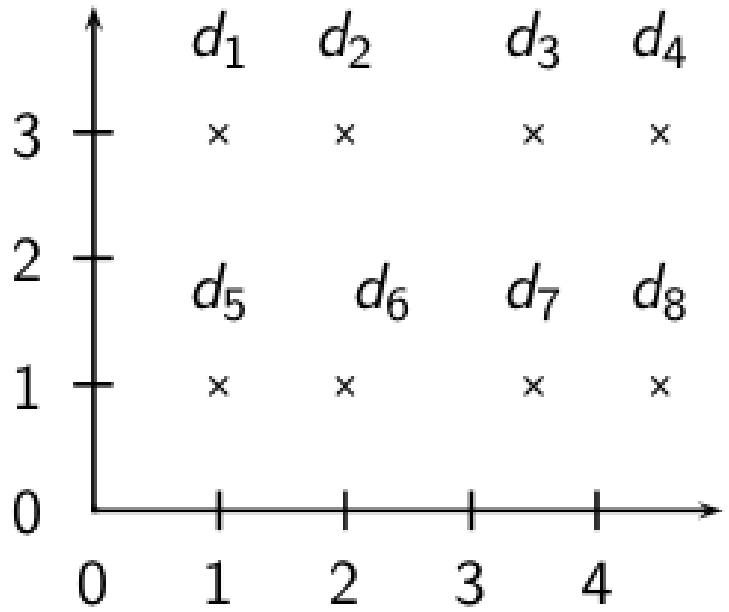
- 计算两个簇的相似度相当于计算合并后的簇的直径

全连接聚类的树状图

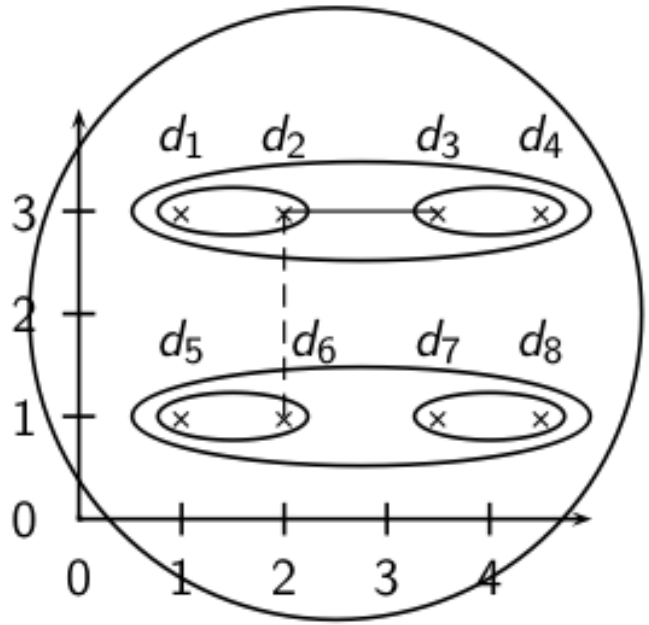


- 我们注意到该图比刚才单连接算法产生的树状图均衡得多
- 我们可以生成一个2个簇的结果，每个簇大小基本相当

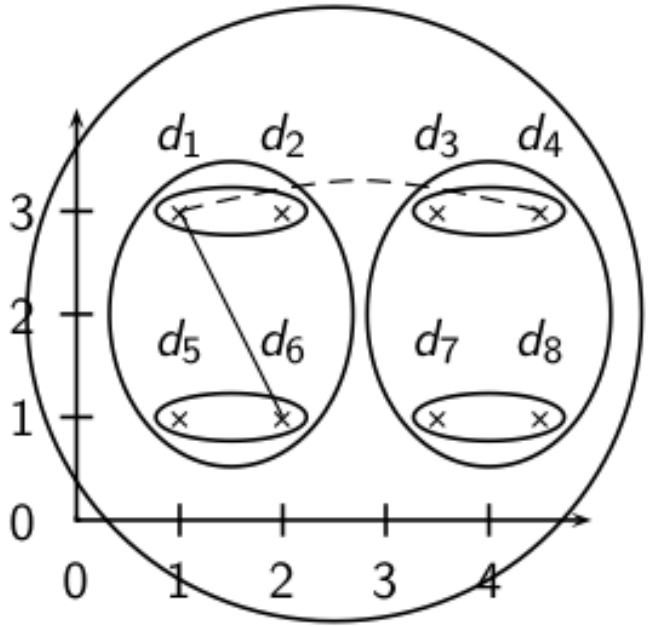
课堂练习： 采用单连接和全连接方法进行聚类



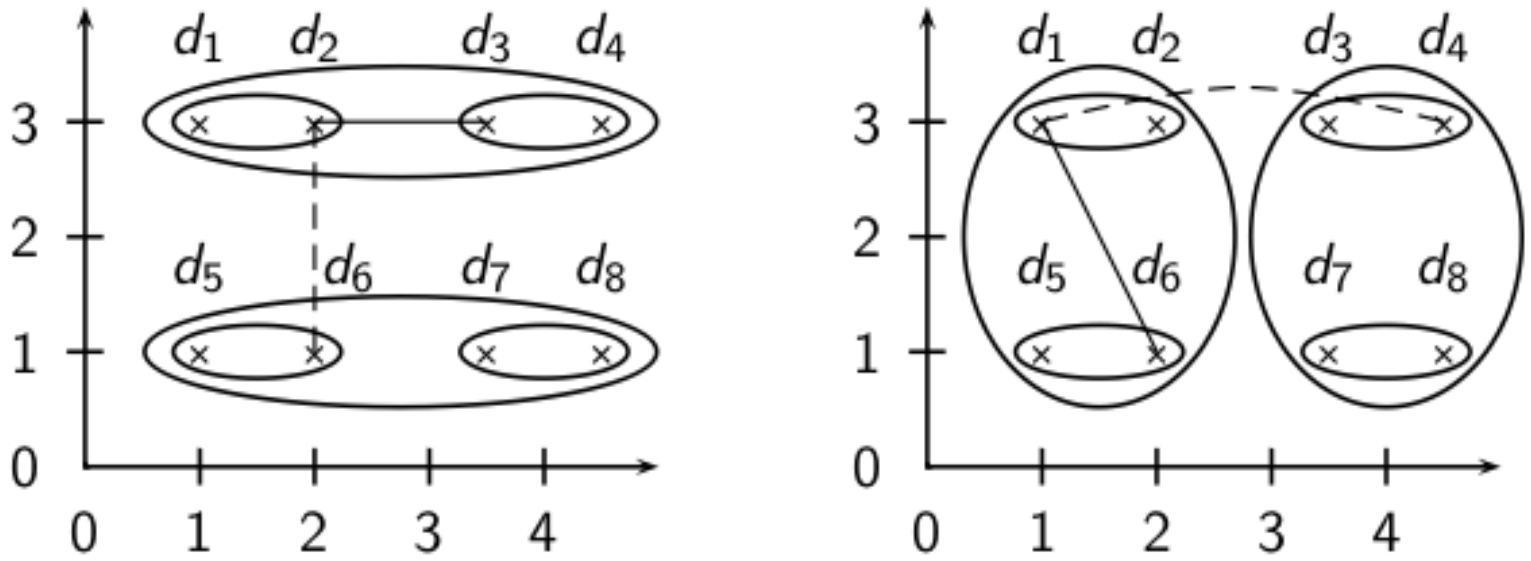
单连接聚类



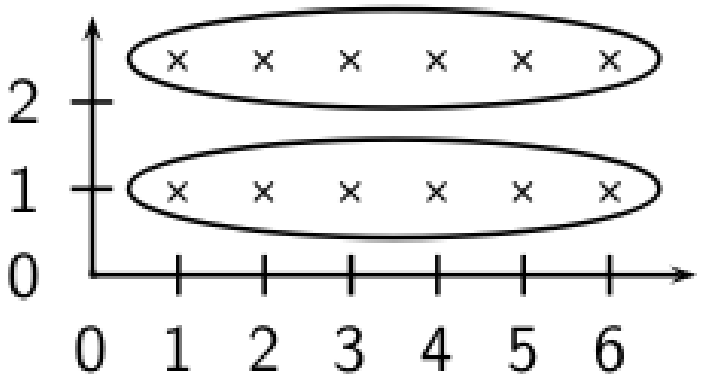
全连接聚类



单连接 vs. 全连接聚类

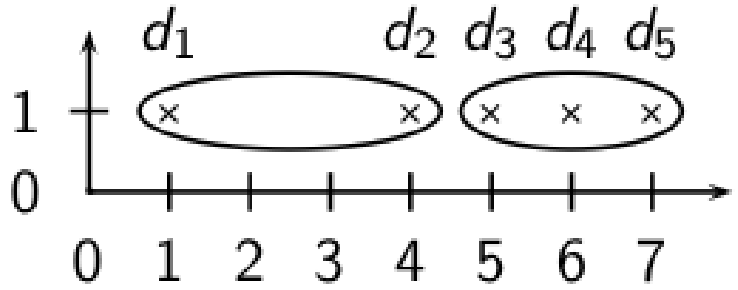

d_1 d_2 d_3 d_4
d_5 d_6 d_7 d_8
0 1 2 3
0 1 2 3 4
d_1 d_2 d_3 d_4
d_5 d_6 d_7 d_8
0 1 2 3
0 1 2 3 4

单连接方法的链化(Chaining)现象



单连接聚类算法往往产生长的、凌乱的簇结构。对大部分应用来说，这些簇结构并不是所期望的。

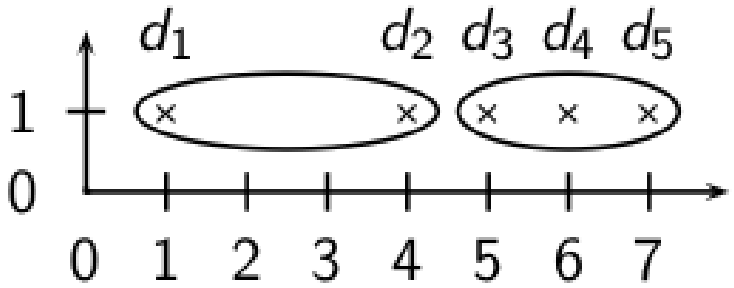
全连接算法会产生怎样的2个簇的结果？



横坐标：

$$1 + 2 \times \epsilon, 4, 5 + 2 \times \epsilon, 6, 7 - \epsilon$$

全连接法: 对离群点非常敏感



- 全连接聚类将d_2和它的正确邻居分开----这显然不是我们所需要的
- 出现上述结果的最主要原因是存在离群点 d_1.
- 这也表明单个离群点的存在会对全连接聚类的结果起负面影响
- 单连接聚类能够较好地处理这种情况

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

质心法HAC

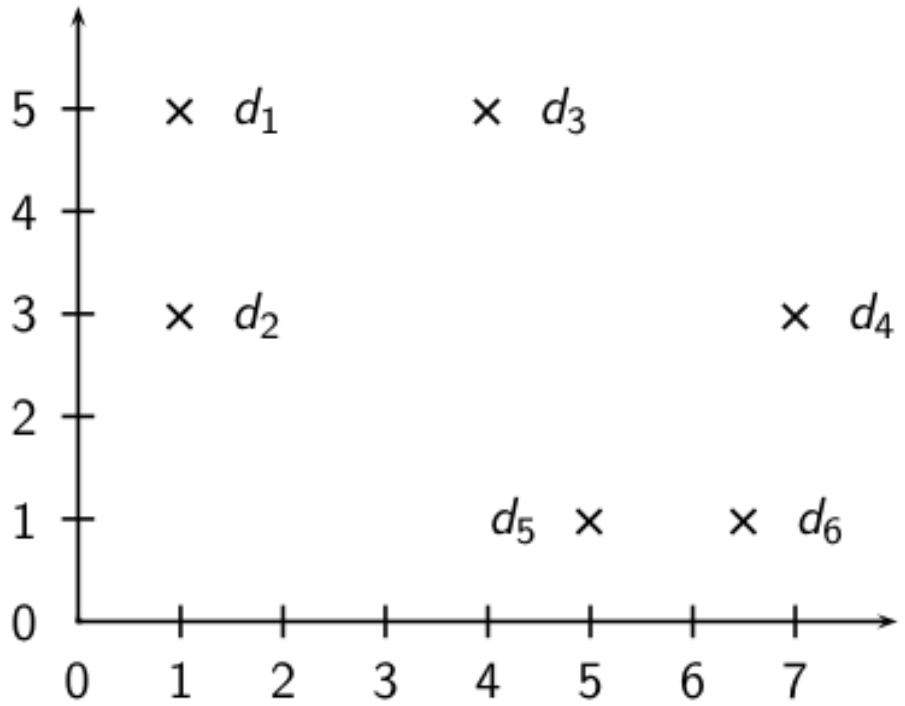
- 簇相似度为所有簇间文档对之间相似度的平均值
- 一个原始的粗糙实现方法效率不高 ($O(N^2)$)，但是上述定义相当于计算两个簇质心之间的相似度：

$$\text{SIM-CENT}(\omega_i, \omega_j) = \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)$$

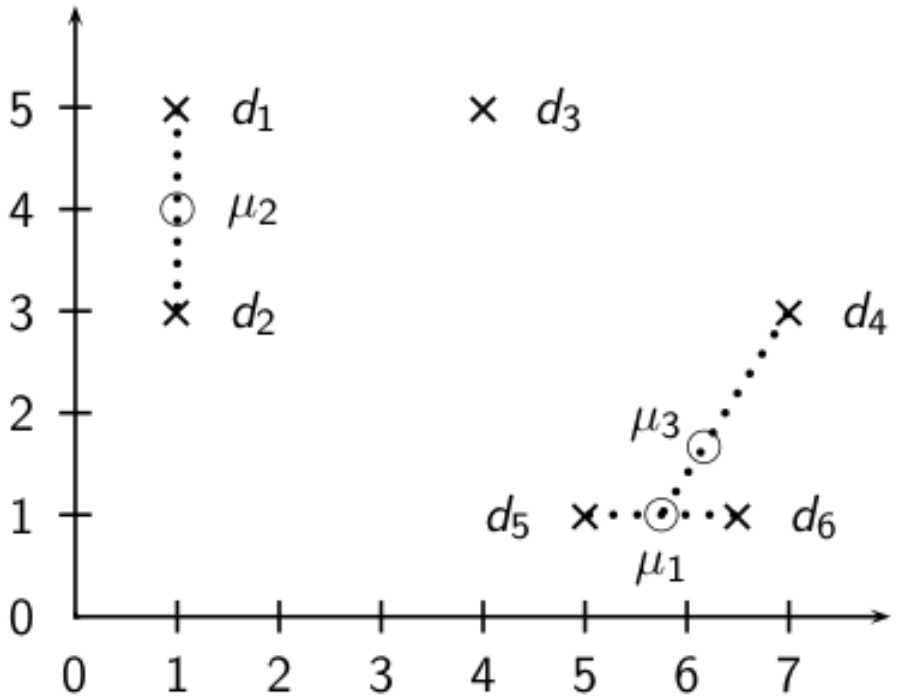
- 这也是质心HAC名称的由来
- 注意：这里是内积计算，而非余弦相似度

$$\begin{aligned}\text{SIM-CENT}(\omega_i, \omega_j) &= \frac{1}{N_i N_j} \sum_{d_m \in \omega_i} \sum_{d_n \in \omega_j} \vec{d}_m \cdot \vec{d}_n \\ &= \left(\frac{1}{N_i} \sum_{d_m \in \omega_i} \vec{d}_m\right) \cdot \left(\frac{1}{N_j} \sum_{d_n \in \omega_j} \vec{d}_n\right) \\ &= \vec{\mu}(\omega_i) \cdot \vec{\mu}(\omega_j)\end{aligned}$$

课堂练习：采用质心法进行聚类

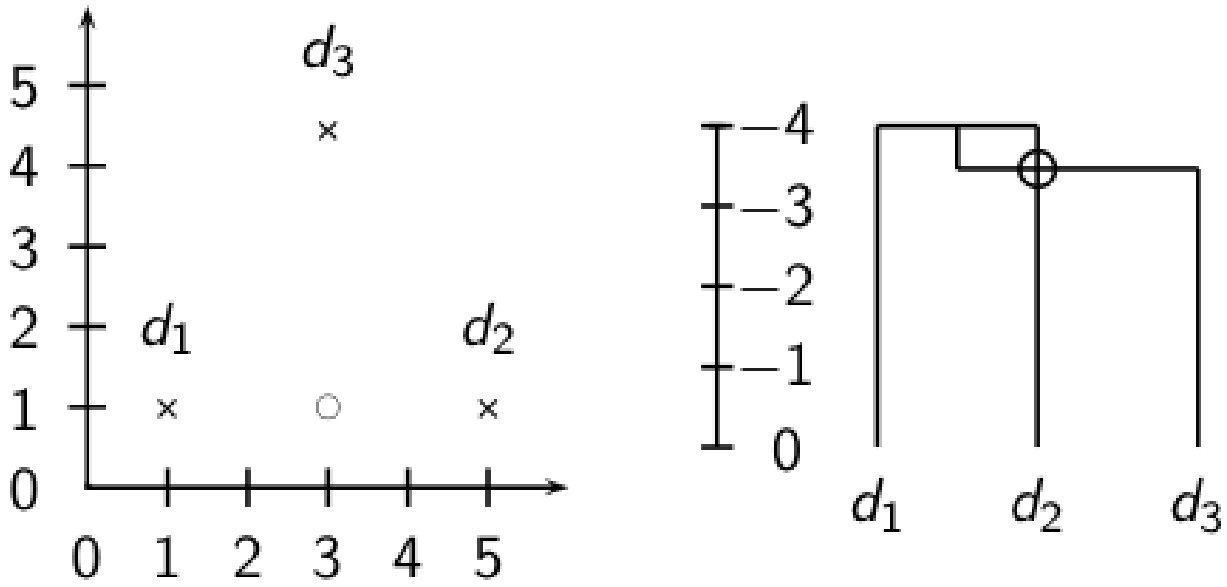


质心法聚类



质心法聚类过程中的相似度颠倒(Inversion)现象

- 在相似度颠倒过程中，合并过程中相似度会增加，导致“颠倒”的树状图
- 下图中，第一次合并 $(d_1 \cup d_2)$ 的相似度是-4.0，第二次合并的相似度 $((d_1 \cup d_2) \cup d_3) \approx -3.5$.



关于相似度颠倒现象

- 允许层次聚类的算法相对较差.
- 层次聚类的基本原理就是在任何给定的点，我们会找到给定大小的最连贯(具有凝聚性) 的结果
- 直观上看：小簇应该比大簇更连贯
- 相似度颠倒现象与此直觉相矛盾：我们产生了一个大簇，但是其连贯性超过其两个子簇

组平均凝聚式算法(GAAC)

- Group-average Agglomerative Clustering
- GAAC 也称为平均相似度准则，但是这个算法中不存在相似度颠倒现象
- 簇之间的相似度是所有文档对之间相似度的平均值(包括来自同一簇的算法)
- 但是不考虑文档自身的自相似度

GAAC凝聚式聚类算法

- 同样，采用原始的简单实现算法复杂度会很高，达到$O(N^2)$，但是也存在一个等价的基于质心定义的高效算法：

$$\text{SIM-GA}(\omega_i,\omega_j)=\frac{1}{(N_i+N_j)(N_i+N_j-1)}\sum_{d_m\in\omega_i\cup\omega_j}\sum_{d_n\in\omega_i\cup\omega_j,d_n\neq d_m}\vec{d}_m\cdot\vec{d}_n$$

$$\text{SIM-GA}(\omega_i,\omega_j)=$$

$$\frac{1}{(N_i+N_j)(N_i+N_j-1)}[(\sum_{d_m\in\omega_i\cup\omega_j}\vec{d}_m)^2-(N_i+N_j)]$$

- 同样需要指出，这里也是内积计算，而非余弦相似度

到底使用哪一个HAC聚类算法？

- 由于存在相似度颠倒，不使用质心法
- 由于GAAC不会受限于链化，并且对离群点不敏感，所以大部分情况下，GAAC都是最佳选择
- 然而，GAAC只能基于向量表示来计算
- 对于其他文档表示方法(或者如果仅仅提供了文档对之间的相似度)时，使用全连接方法
- 有些应用中适合用单链算法 (比如，Web搜索中的重复性检测，判断一组文档重复并不受那些离它们较远的文档所影响)

扁平聚类还是层次聚类?

- 为了达到高效性，使用扁平算法 (或者或许可以采用二分K-均值算法)
- 对于确定性结果: HAC
- 当需要层次结构时：层次算法
- HAC也能在不确定K的情况下使用

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

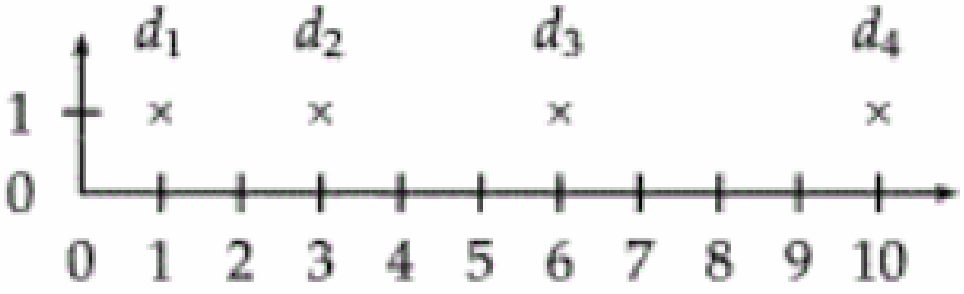
高效的单连接聚类算法

SINGLELINKCLUSTERING$(d_1, \ldots, d_N)$

1 **for** $n \leftarrow 1$ **to** N
2 **do for** $i \leftarrow 1$ **to** N
3 **do** $C[n][i].\text{sim} \leftarrow \text{SIM}(d_n, d_i)$
4 $C[n][i].\text{index} \leftarrow i$
5 $I[n] \leftarrow n$
6 $NBM[n] \leftarrow \arg\max_{X \in \{C[n][i]: n \neq i\}} X.\text{sim}$
7 $A \leftarrow []$
8 **for** $n \leftarrow 1$ **to** $N - 1$
9 **do** $i_1 \leftarrow \arg\max_{\{i: I[i]=i\}} NBM[i].\text{sim}$
10 $i_2 \leftarrow I[NBM[i_1].\text{index}]$
11 $A.\text{APPEND}(\langle i_1, i_2 \rangle)$
12 **for** $i \leftarrow 1$ **to** N
13 **do if** $I[i] = i \wedge i \neq i_1 \wedge i \neq i_2$
14 **then** $C[i_1][i].\text{sim} \leftarrow C[i][i_1].\text{sim} \leftarrow \max(C[i_1][i].\text{sim}, C[i_2][i].\text{sim})$
15 **if** $I[i] = i_2$
16 **then** $I[i] \leftarrow i_1$
17 $NBM[i_1] \leftarrow \arg\max_{X \in \{C[i_1][i]: I[i]=i \wedge i \neq i_1\}} X.\text{sim}$
18 **return** A

全连接聚类不具最佳合并持续性

- 例子



- 在聚类开始前，d_3的最佳合并簇是?
- d_1和d_2合并后，d_3的最佳合并簇是?

HAC的时间复杂度

- 刚才看到的单连接算法的复杂度是$O(N^2)$.
- 比前面提到的$O(N^3)$ 快很多！
- 目前还没有听说有 $O(N^2)$的全连接HAC、质心HAC和GAAC算法
- 后三种方法的最优时间复杂度为$O(N^2 \log N)$，参考教材相关部分。
- 实际当中，$O(N^2 \log N)$ 和$O(N^2)$的差别不大。

四种算法的组合相似度(Combination similarity)

clustering algorithm	$\text{sim}(\ell, k_1, k_2)$
single-link	$\max(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
complete-link	$\min(\text{sim}(\ell, k_1), \text{sim}(\ell, k_2))$
centroid	$(\frac{1}{N_m}\vec{v}_m) \cdot (\frac{1}{N_\ell}\vec{v}_\ell)$
group-average	$\frac{1}{(N_m+N_\ell)(N_m+N_\ell-1)}[(\vec{v}_m + \vec{v}_\ell)^2 - (N_m + N_\ell)]$

四种HAC算法的比较

方　　法	结合相似度	时间复杂度	是否最优？	注　　释
单连接	簇间文档的最大相似度	$\Theta(N^2)$	yes	链化效应
全连接	簇间文档的最小相似度	$\Theta(N^2 \log N)$	no	对离群点敏感
质心法	所有簇间相似度的平均值	$\Theta(N^2 \log N)$	no	相似度颠倒
组平均	所有文档相似度的平均值	$\Theta(N^2 \log N)$	no	大部分应用中的最佳选择

如何使用层次结构?

- 按其结构使用 (比如，在Yahoo层次结构中浏览)
- 在某个预定阈值出截断
- 在某个预定的簇个数K处截断
  - 忽略截断线上下的层次结构

二分(Bisecting) *K*-均值: 一个自顶向下的算法

- 一开始所有的文档在一个簇中
- 使用K-均值算法将该簇分裂成2个簇
- 在已产生的簇当中，选择其中一个进行分裂操作 (比如，选择最大的那个簇)
- 重复上述操作直至达到期望的簇的数目

二分K-均值(Bisecting K-means)

BISECTINGKMEANS($d_1, \ldots, d_N$)
1 $\omega_0 \leftarrow \{\vec{d}_1, \ldots, \vec{d}_N\}$
2 *leaves* $\leftarrow \{\omega_0\}$
3 **for** $k \leftarrow 1$ **to** $K - 1$
4 **do** $\omega_k \leftarrow$ PICKCLUSTERFROM(*leaves*)
5 $\{\omega_i, \omega_j\} \leftarrow$ KMEANS(ω_k, 2)
6 *leaves* $\leftarrow$ *leaves* $\setminus \{\omega_k\} \cup \{\omega_i, \omega_j\}$
7 **return** *leaves*

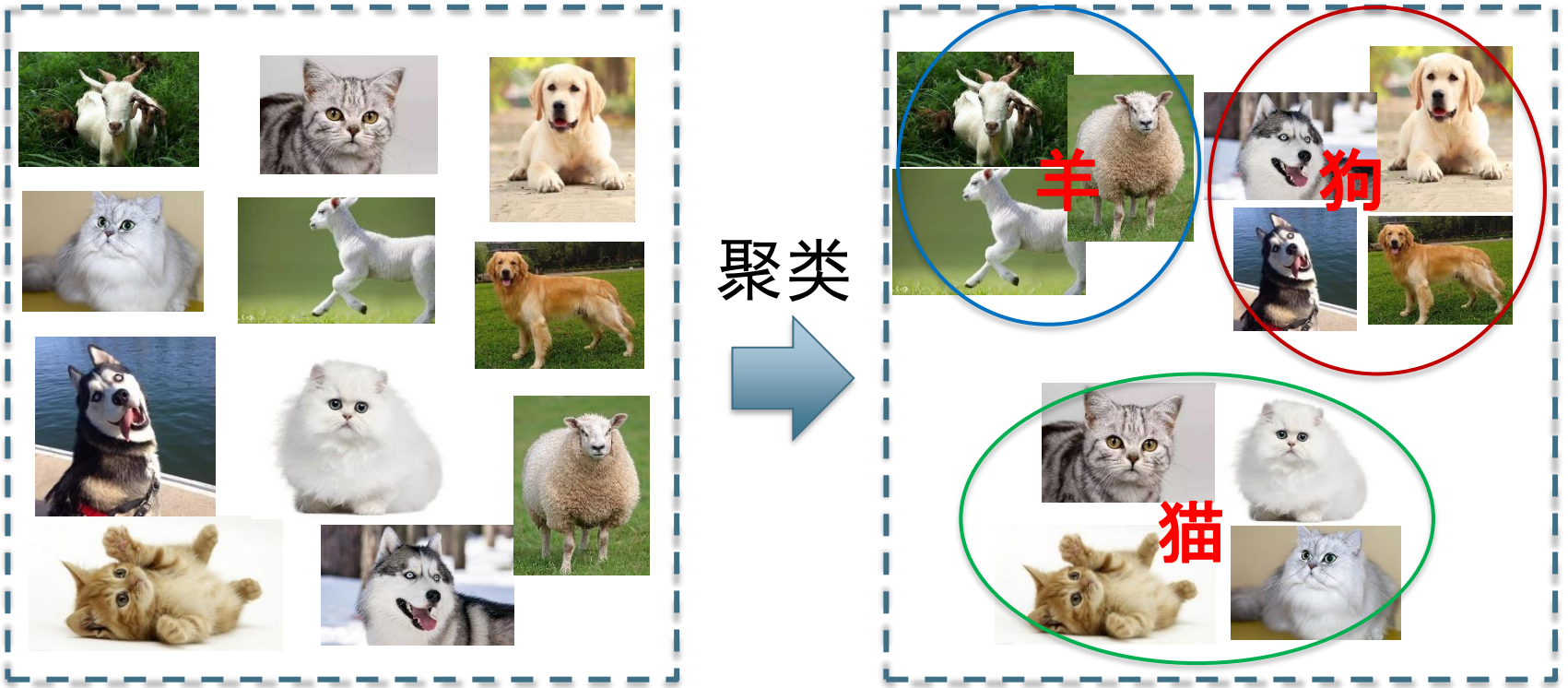
二分K-均值算法

- 如果不期望产生一个完整的层次结构，那么采用诸如二分K-均值算法的自顶向下的方法会比HAC算法高效很多
- 但是，二分K-均值算法的结果不是确定性的
- 存在确定性的二分K-均值算法版本 (参考讲义最后的参考资料部分)，但是这些算法的效率要差一些

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

簇标签生成



- 如何给聚类出的簇打标签？

簇标签生成(clustering labeling)中的主要问题

- 聚类算法结束之后输出一系列簇，那么这些簇怎样才能对用户有用？
- 我们必须对每个簇给出一个简洁精炼的标签
- 例如，在 “jaguar”的搜索结果中，可以采用 “animal”、“car”和“operating system”等不同标签
- 本节主题：如何能够自动找到好的簇标签？

课堂练习

- 给出一个簇标签生成算法
- 输入： 被分成K个簇的一系列文档
- 输出：每个簇给出一个标签
- 练习中要思考，我们应该考虑哪种标签类型？词语？

差别式簇标签生成方法

- 为了对簇ω生成标签，将 ω 和其它簇进行比较
- 找到那些将 ω 和其它簇区别开的词项或者短语
- 我们可以使用前面在文本分类中提到的满足差别式要求的任意特征选择方法：互信息、χ^2 或者词频
- (但是，最后一种实际上不是差别式方法)

非差别式簇标签生成方法

- 只基于簇本身的信息来选择词项或者短语
- 比如，质心向量中高权重的词项 (假如我们正在使用一个向量空间模型)
- 非差别式方法有时会选出并不能区分簇的高频词项
- 如，新闻文本中的MONDAY, TUESDAY, . . .

在簇标签生成中使用标题

- 不论是词项还是短语，都很难完整表达整个簇的含义
- 另一种办法是使用标题
- 例如，最接近质心的2到3篇文档的标题
- 标题比一系列短语的可读性更强

簇标签生成的例子

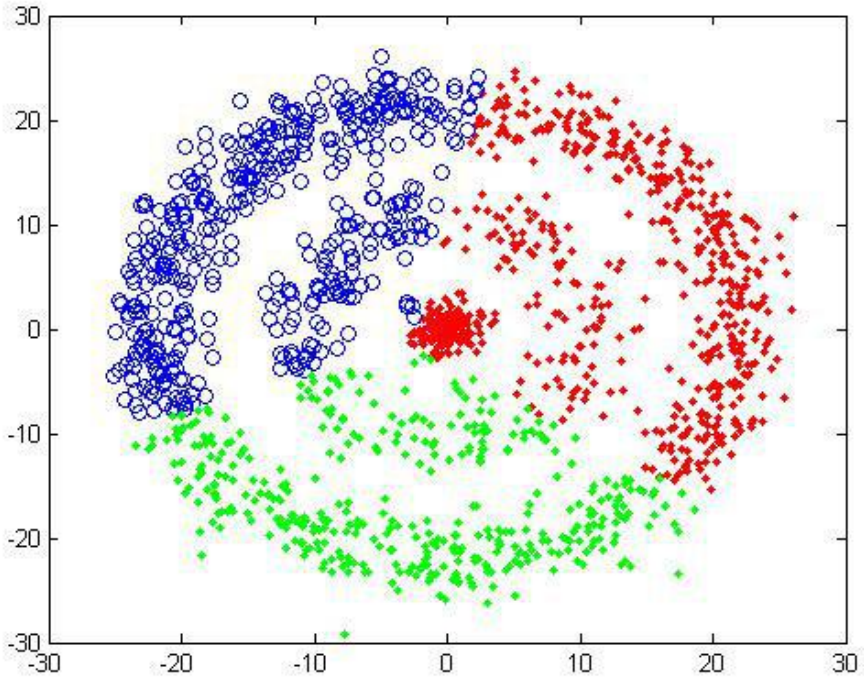
	文档数目	簇标签生成方法		
		质心	互信息	标题
4	622	oil plant mexico production crude **power 000 refinery gas** bpd	plant oil production **barrels** crude bpd mexico **dolly capacity petroleum**	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security **russian** people military peace killed told **grozny court**	police killed military security peace told **troops forces rebels** people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices **cents september** tonne	**delivery** traders futures tonne tonnes **desk** wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

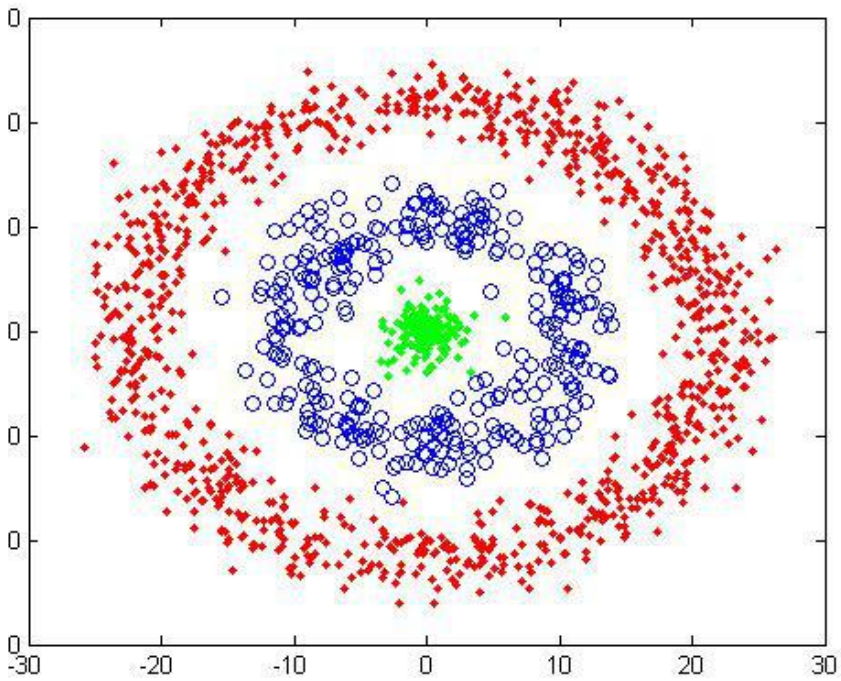
- 三种方法：选择质心向量中的突出词项，使用MI的差别式标签，使用离质心最近的文档的标题
- 三种方法的结果都不错

提纲

1. 上一讲回顾
2. 层次聚类介绍
3. 单连接/全连接算法
4. 质心/GAAC算法
5. 其他实现变种
6. 簇标签生成
7. 其他聚类算法

非凸数据集的聚类





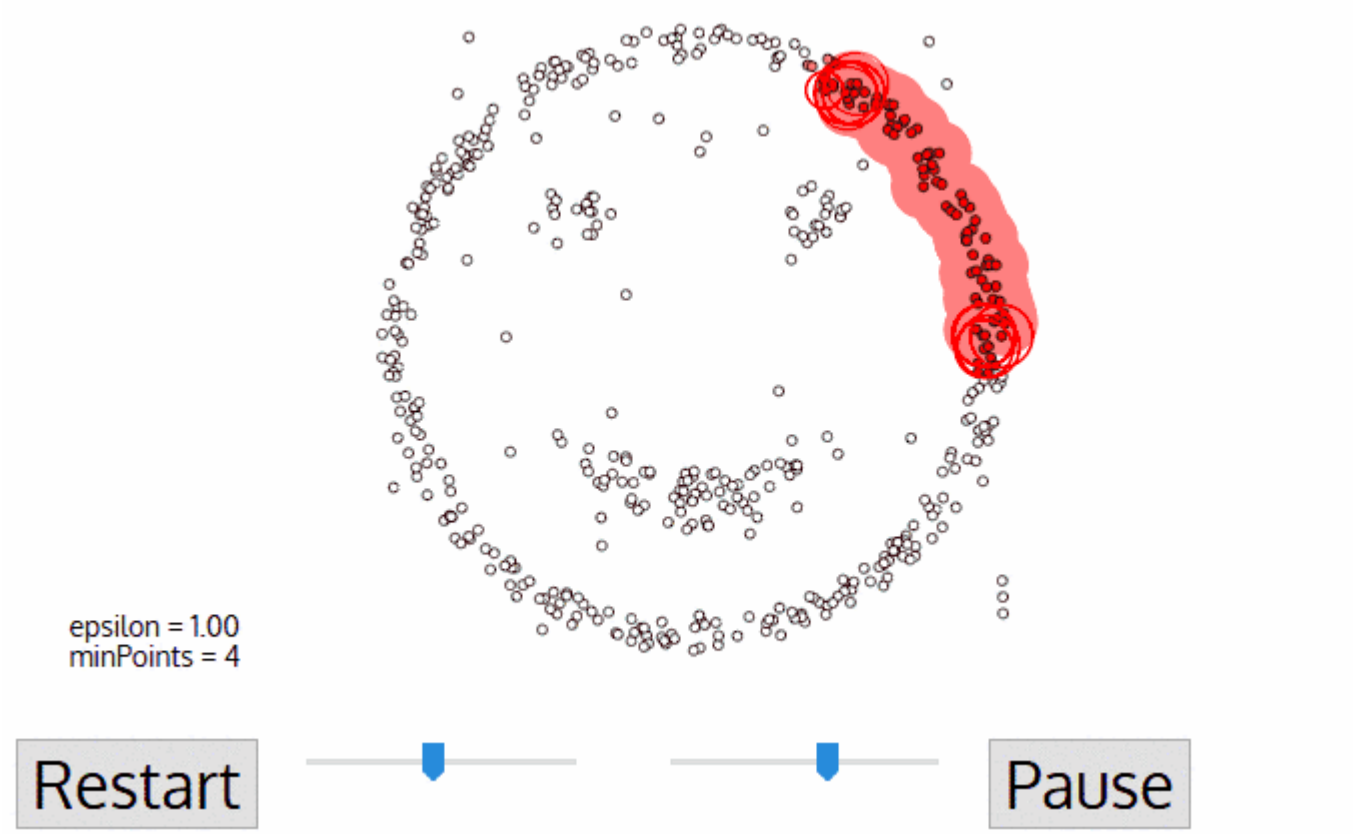
- 哪种聚类结果好？

DBSCAN

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类)算法是一种常见的基于密度的聚类方法
- 大致过程如下：首先把所有的样本标记为核心点、边界点或噪声点。其中一个样本是核心点，满足在该样本的邻域（由距离函数和用户指定的参数R确定）内的样本的个数大于给定的阈值Min。边界点是位于某核心样本邻域的非核心样本，而噪声点指既非核心样本又不是边界样本的样本。然后对每个样本，做如下处理：删除噪声点，而足够靠近的核心点（它们的距离小于R）聚集在同一簇中，与核心点足够靠近（它们的距离小于R）的边界点也聚集在与核心点相同的簇中。
- DBSCAN算法可以有效地发现数据库中任意形状的簇，自动确定聚类的簇个数，但也存在一定的局限性，例如参数R和Min仍然需要用户依靠经验设置。

A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, KDD 1996

DBSCAN



Science 2014年的聚类算法

- 对每个点定义两个值：
- 局部密度：

$$\rho_i = \sum_{j \in I_S \setminus \{i\}} \chi(d_{ij} - d_c)$$

其中

$$\chi(x) = \begin{cases} 1, & x < 0; \\ 0, & x \geq 0, \end{cases}$$

- 距离：

$$\delta_{q_i} = \begin{cases} \min\limits_{\substack{q_j \\ j<i}} \{d_{q_i q_j}\}, & i \geq 2; \\ \max\limits_{j \geq 2} \{\delta_{q_j}\}, & i = 1. \end{cases}$$

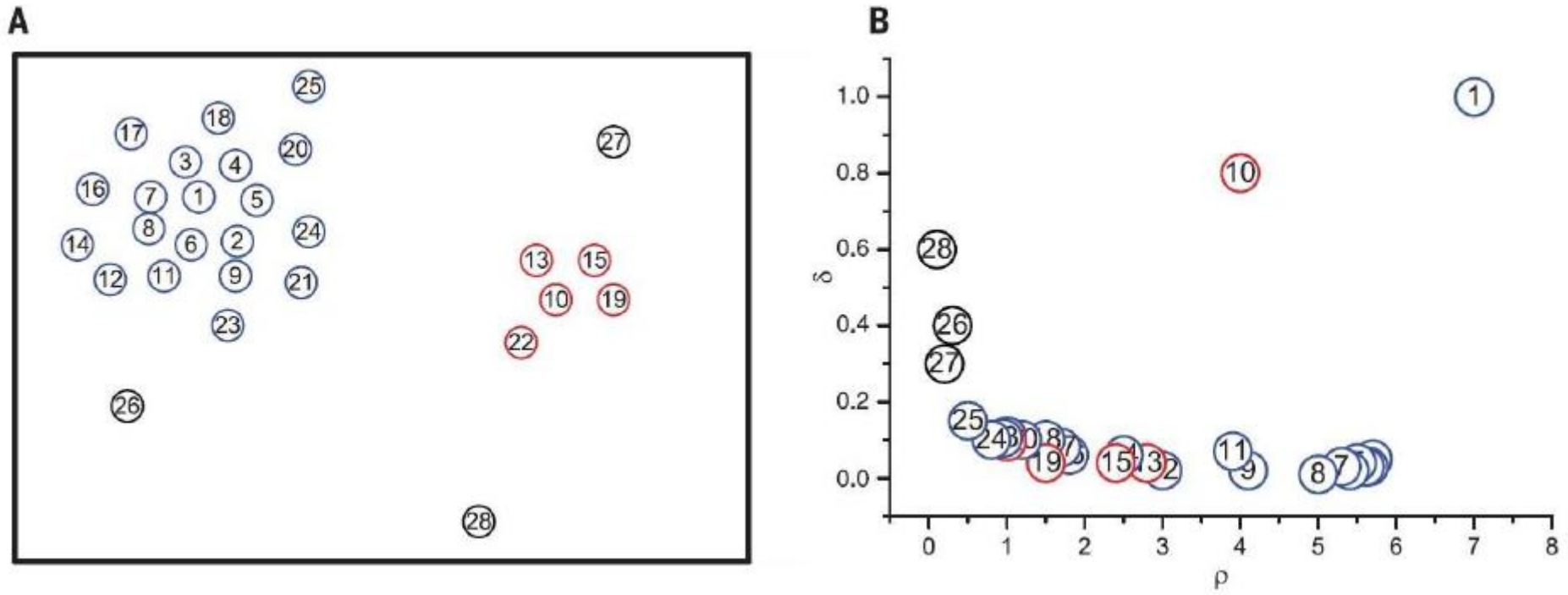
其中

$$\rho_{q_1} \geq \rho_{q_2} \geq \cdots \geq \rho_{q_N}$$

- 选择两者乘积最大的那些点作为聚类中心点。

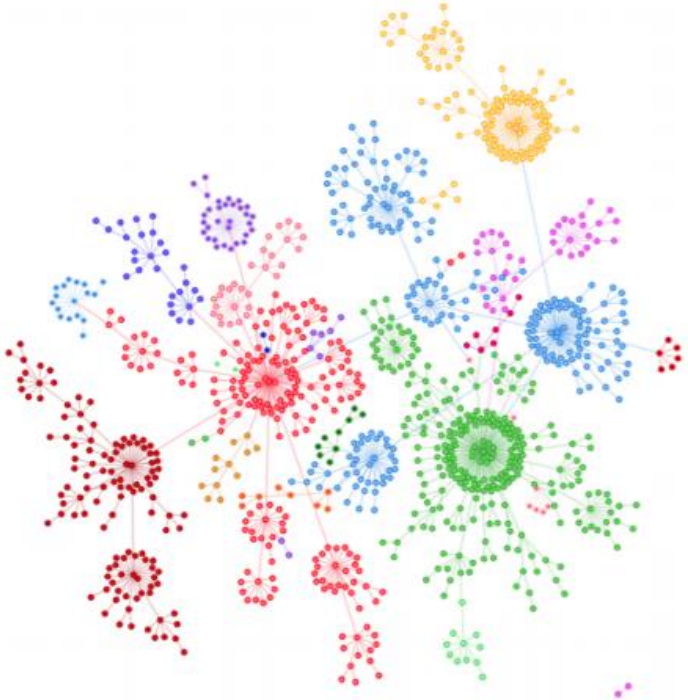
Alex Rodriguez, Alessandro Laio, Clustering by fast search and find of density peaks,Science 344, 1492, April 2014

Science 2014年的聚类算法



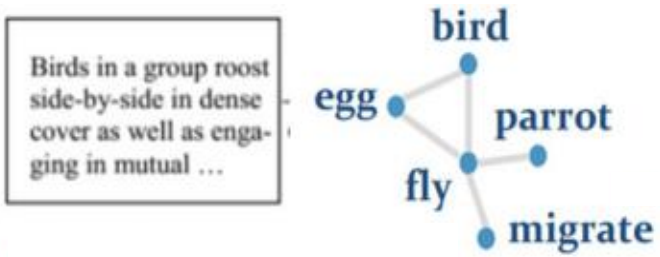
谱聚类

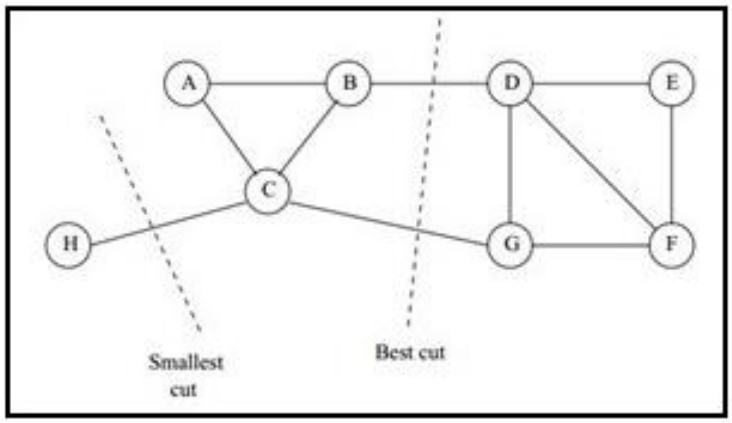
- 谱聚类(Spectral Clustering, SC)是一种基于图论的聚类方法。文本可以表示为图的形式，比如可以把整个文档当做顶点，文档和文档间的相似度作为连边的权重，也可以把文档中的关键词作为顶点，关键词间的关联关系作为连边。然后用基于图的聚类方法。



谱聚类

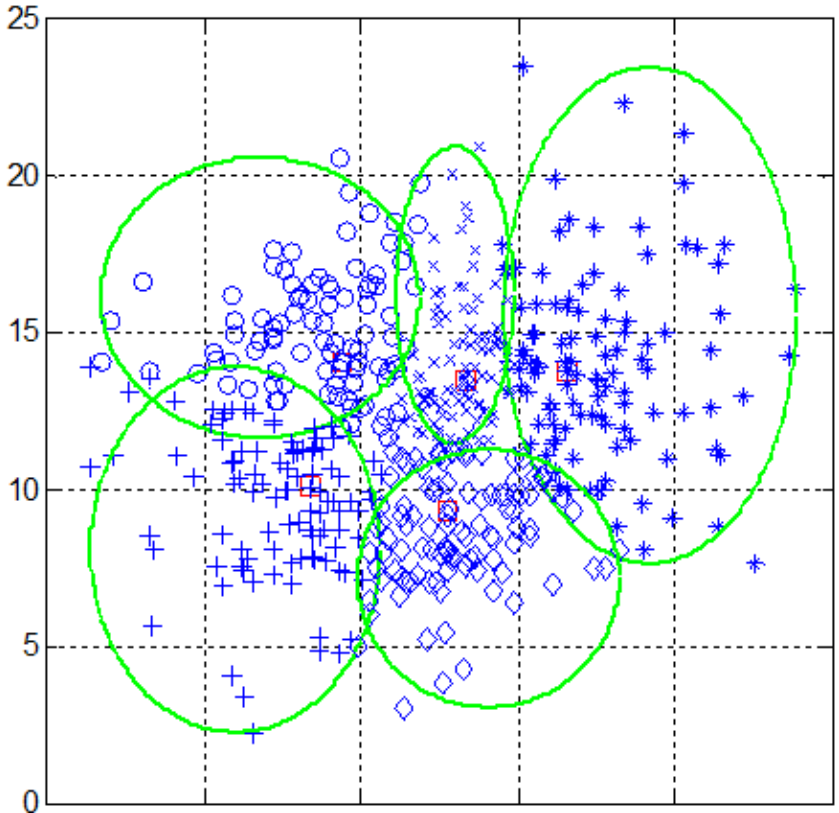
- 将带权无向图划分为两个或两个以上的最优子图，使子图内部尽量相似，而子图间距离尽量距离较远，以达到常见的聚类的目的。其中的最优指最优目标函数，可以有不同的最优目标函数，比如可以是割边最小分割， 也可以是分割规模差不多且割边最小的分割。





模糊聚类

- 模糊C均值（Fuzzy C-means）算法简称FCM算法，是一种基于目标函数的模糊聚类算法，属于扁平的软聚类算法。



模糊聚类

$X=\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$

划分X为c类

$X=X_1\cup X_2\cup\cdots\cup X_c$

隶属度矩阵定义：j个样品属于第i个中心的隶属度u_{ij}

$$\sum_{i=1}^{c}u_{ij}=1,u_{ij}\geq 0.\quad j=1,2,\ldots,n,i=1,2,\ldots,c$$

$\mathbf{U}=(u_{ij})_{c\times n}$ ---隶属度矩阵

$$\mathbf{U}=(u_{ij})_{2\times 6}=\begin{pmatrix}1 & 0.8 & 0.6 & 0.4 & 0.2 & 0\\ 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1\end{pmatrix}$$

模糊聚类

定义目标函数：

$$J_m(\mathbf{U},\mathbf{V})=\sum_{j=1}^{n}\sum_{i=1}^{c}u_{ij}^{m}d^2(x_j,v_i)$$

其中 $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\cdots,\mathbf{v}_c\}\subset R^p$ 是c个类的聚类中心

$m>1$ 是加权指数，m 的取值能够影响聚类的效果.

最优划分：选择 $\mathbf{U},\mathbf{V},m,\ J_m(\mathbf{U},\mathbf{V})$最小

模糊聚类

(1) 预先给定分类数c、加权指标数m、初始化隶属度矩阵

$$\mathbf{U}=(u_{ij})_{c\times n} \qquad \sum_{i=1}^{c} u_{ij}=1;$$

(2) 计算聚类中心 $v_i, i=1,2,\ldots,c;$

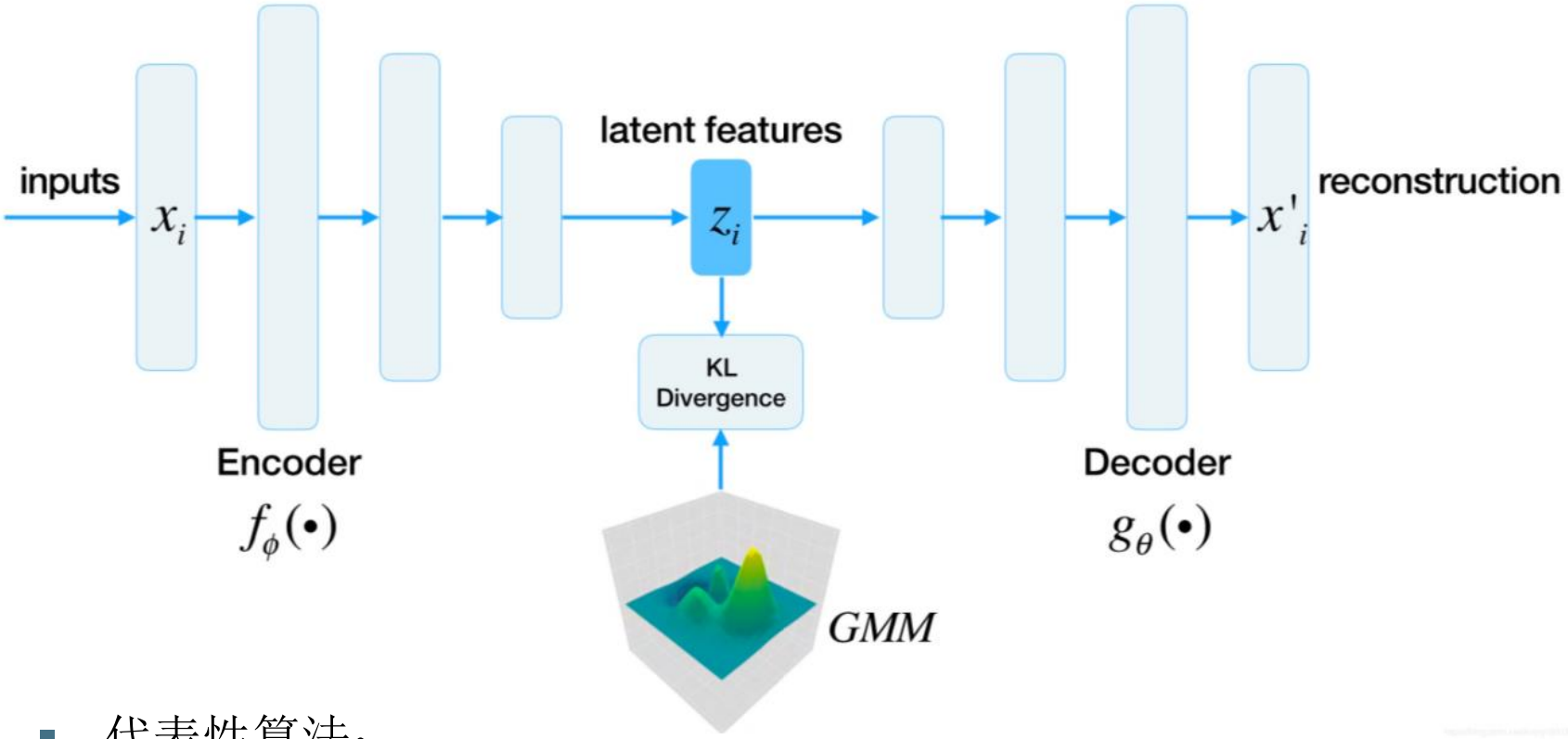
$$\mathbf{v}_i=\sum_{j=1}^{n}(u_{ij})^m\mathbf{x}_j \Big/ \sum_{j=1}^{n}(u_{ij})^m$$

(3) 计算新的隶属度矩阵；$\mathbf{U}=(u_{ij})_{c\times n}$

$$u_{ij}=\left[\sum_{k=1}^{c}\left(\frac{\|\mathbf{x}_j-\mathbf{v}_i\|^2}{\|\mathbf{x}_j-\mathbf{v}_k\|^2}\right)^{1/(m-1)}\right]^{-1} \qquad 1\le i\le c,\ 1\le j\le n$$

基于VAE的深度聚类

- 变分自编码器（Variational Autoencoder）



- 代表性算法：
  - Variational Deep Embedding (VaDE)
  - Gaussian Mixture VAE (GMVAE)

聚类研究趋势

- 大规模环境下的聚类算法
- 深度聚类模型
- 聚类的评价
- 聚类的应用
  - 事件的发现与跟踪
  - 话题的提取
  - 文档主题
  - ……

参考资料

- 《信息检索导论》第17章
- http://ifnlp.org/ir
  - Columbia Newsblaster (Google News的先驱者): McKeown et al. (2002)
  - 二分 *K*-均值聚类: Steinbach et al. (2000)
  - PDDP (与二分K-均值方法类似，但是是确定性算法，效率稍差): Saravesi and Boley (2004)
- A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture 2018

课后练习

- 习题17-3

对于固定的N篇文档，单连接和全连接聚类中，簇之间相似度的计算均退化为两篇文档之间的相似度计算，因此，簇相似度最多有N(N-1)/2个。那么对于GAAC和质心聚类算法，不同的簇相似度数目最多是多少？

- 习题17-10

考虑对如下直线上的N个点进行单连接聚类。

试证明我们总共只需要计算N次相似度。另外，请给出对直线上点进行单连接聚类的总时间复杂度。