人工智能
——层次聚类

Yanghui Rao
Assistant Prof., Ph.D
School of Data and Computer Science,
Sun Yat-sen University
raoyangh@mail.sysu.edu.cn

层次聚类

- 层次聚类是树状组织的一系列嵌套簇。

- 层次聚类有两种基本方法：
  - Agglomerative (凝聚式) -> bottom-up
  - Divisive (分裂式) -> top-down

凝聚式层次聚类

- ***凝聚式***层次聚类（Hierarchical Agglomerative Clustering，简称HAC），一开始先把每个样本视为个体簇。
- 每一步，合并距离最近的一对簇。
- 这需要定义簇间距离的概念。 (如何计算簇与簇之间的距离?)

凝聚式层次聚类

凝聚式层次聚类

凝聚式层次聚类

凝聚式层次聚类

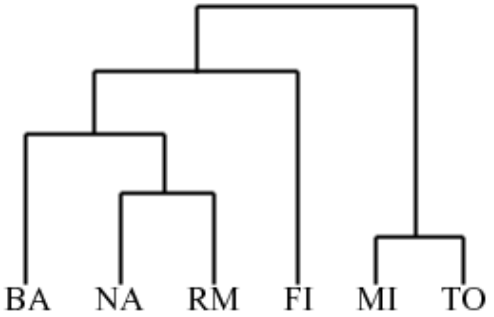
凝聚式层次聚类

分裂式层次聚类

- ***分裂式***层次聚类，一开始把所有样本视作一个大簇。步骤如下：
  - 每一步分割一个簇。
  - 持续这个过程，直到每个簇只包含一个样本/对象
- 此情况下我们需要决定
  - 在每一步分割哪个簇，以及
  - 如何分割

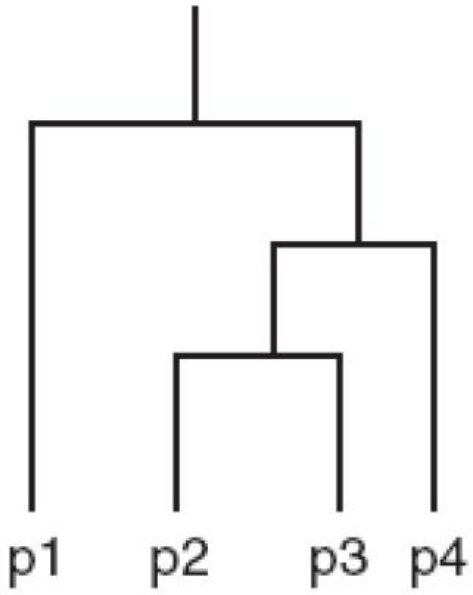
层次聚类的结果展示

- 层次聚类通常用一个类似树的图案表示，称为树状图（dendrogram）
- 树状图展示了
  - 簇和子簇的关系
  - 簇凝聚或者分裂的顺序

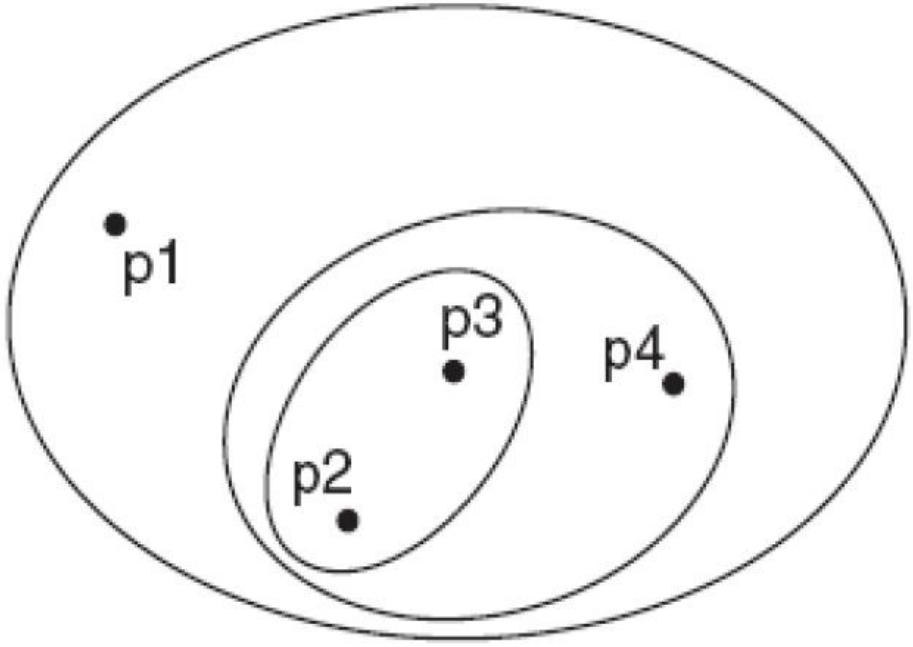


- 对二维样本，层次聚类也可以用嵌套簇图表示

层次聚类的结果展示



(a) Dendrogram.



(b) Nested cluster diagram.

凝聚式层次聚类

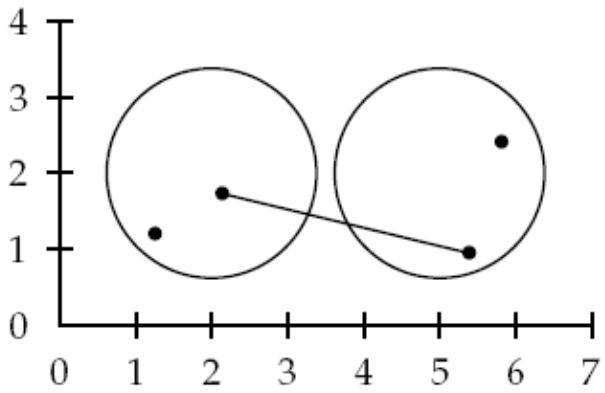
- 凝聚式层次聚类的步骤大致总结如下：
  - 计算距离矩阵
  - 迭代
    - 合并距离最近的两个簇
    - 更新距离矩阵，从而反映新簇和原簇的距离
  - 直到只剩下一个簇

凝聚式层次聚类

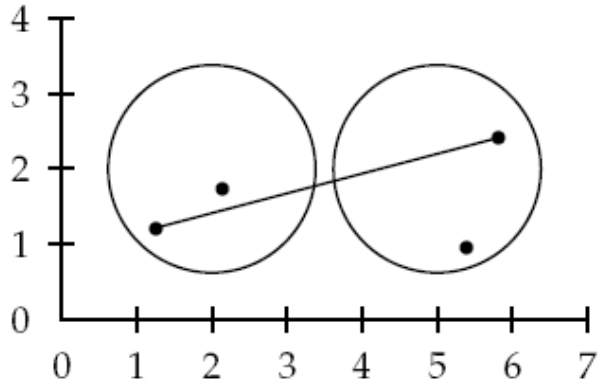
- 对簇间距离的不同定义会导致凝聚式层次聚类的不同版本。

- 这些版本包括
  - Single link (单连接) or MIN
  - Complete link (全连接) or MAX
  - Group average (组平均)
  - Centroid Similarity

凝聚式层次聚类



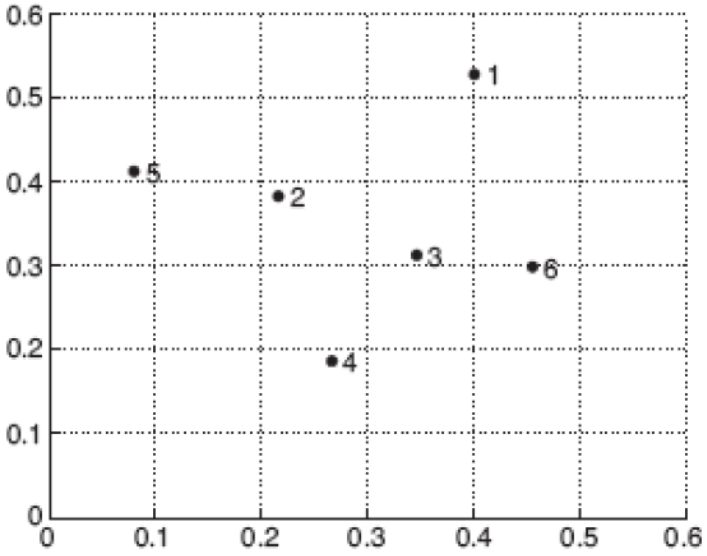
(a) single link: maximum similarity



(b) complete link: minimum similarity

凝聚式层次聚类

- 考虑如下点集。
- 这些数据点的欧氏距离矩阵在下一页PPT。



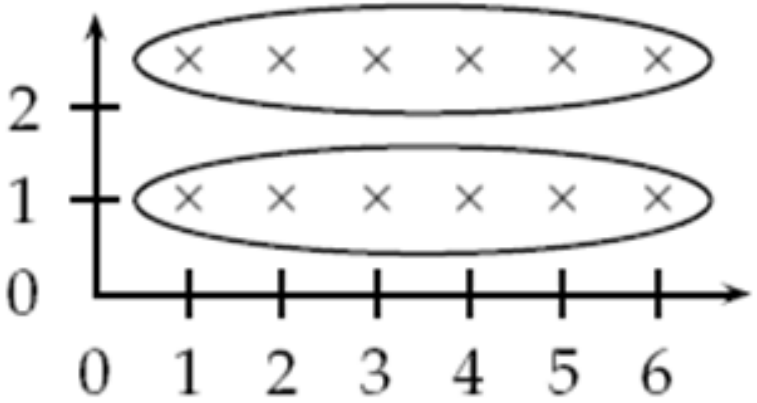
Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

凝聚式层次聚类

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

单连接

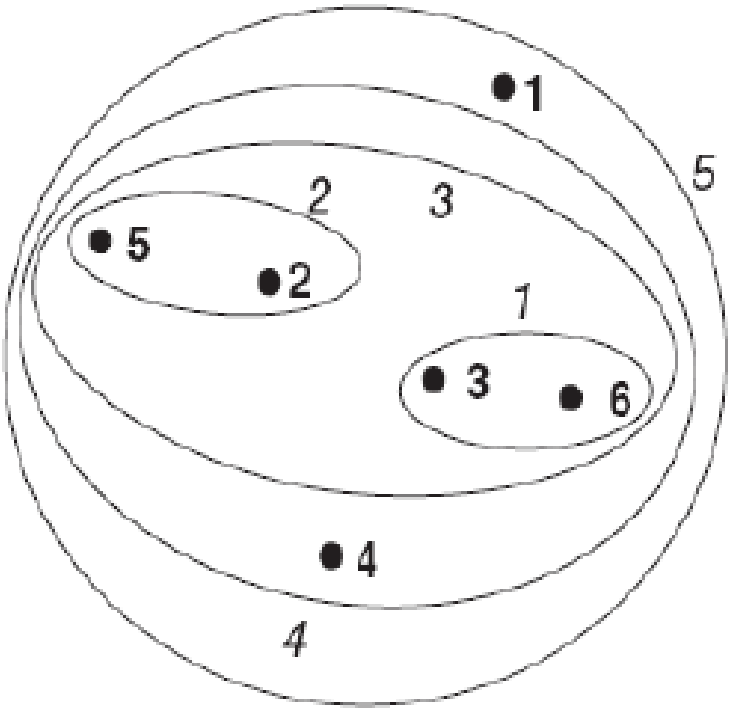
- 考虑层次聚类的单连接版本（MIN版本）
- 该情况下，两个簇的距离定义为位于两个簇中的任意两个点的最小距离。
- 这种方法可以较好地处理非球状（non-elliptical）样本集。



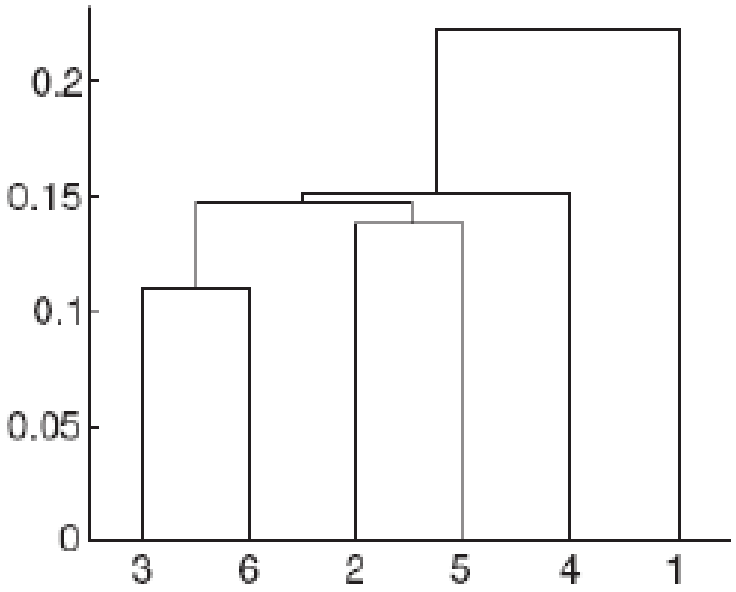
单连接

- 下图展示了在样例数据上应用单连接的结果。
- 左图以嵌套球的序列展示了嵌套簇。
- 球上的数字意为聚类顺序。
- 右图用树状图展示了相同信息。
- 树状图的高度即距离值。

单连接



(a) Single link clustering.



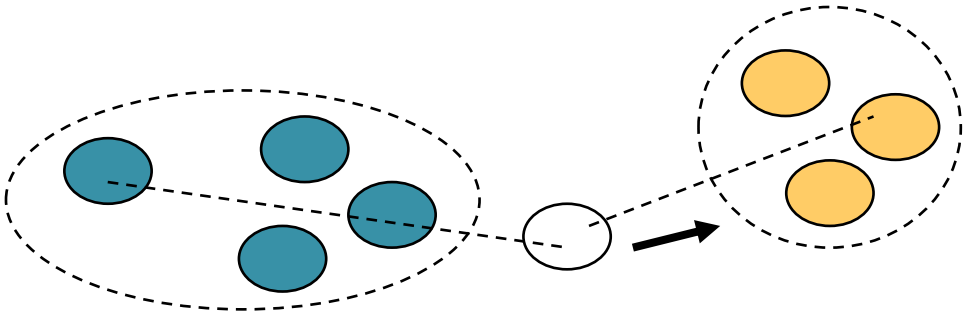
(b) Single link dendrogram.

单连接

- 比如，点 3 和点 6 的距离为 0.11.
- 这就是当它们在树状图中聚成一个簇时所在的高度
- 那么，为什么簇 {3,6} 和 {2,5} 的距离是 0.15?

全连接

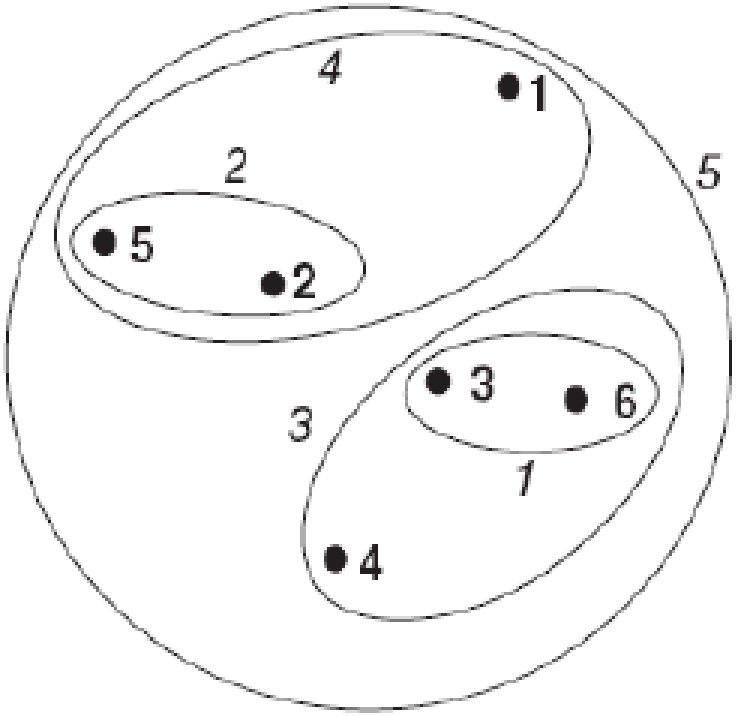
- 现在考虑层次聚类的全连接版本（MAX版本）
- 该情况下，两个簇的距离定义为位于两个簇中的任意两个点的最大距离。
- 全连接倾向于形成球状簇。



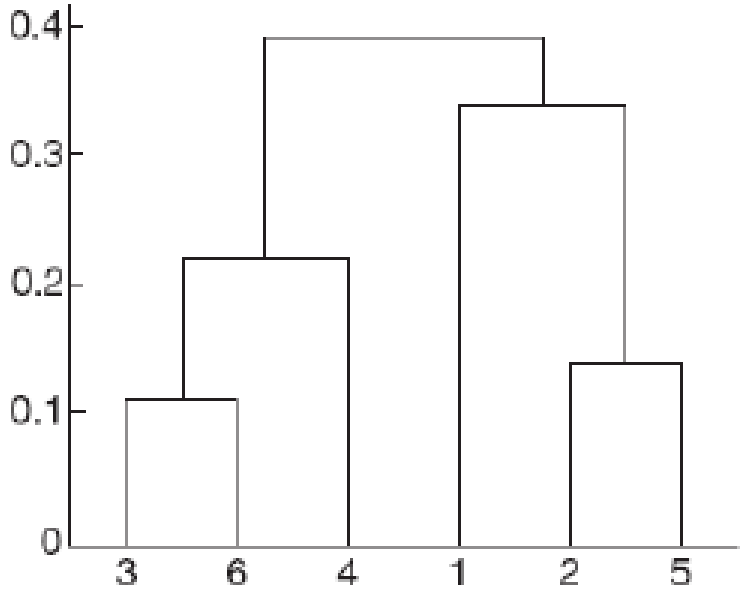
全连接

- 下图展示了在样本数据点上应用全连接后的结果。
- 和单连接一样，点3和6先被合并。
- 然后点 2 和 5 被合并。
- 之后 {3,6} 和 {4}合并

全连接

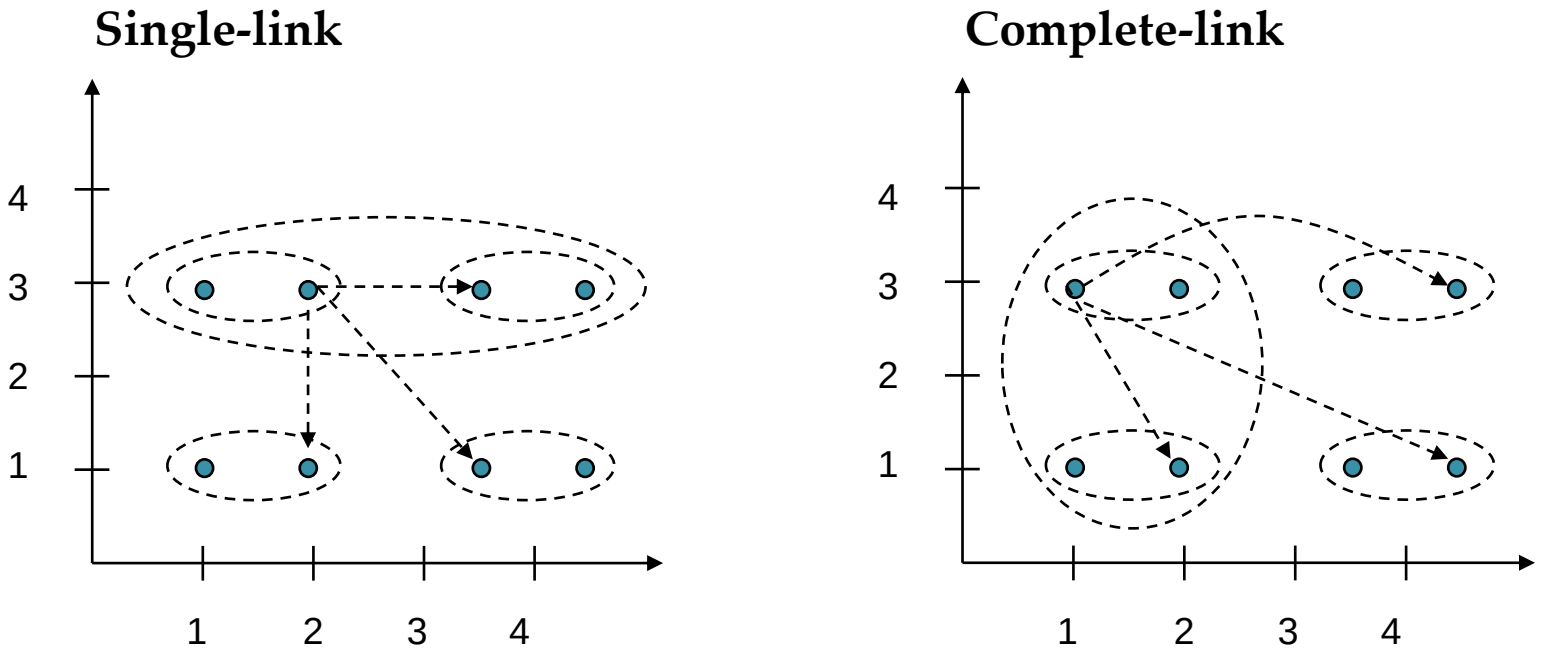


(a) Complete link clustering.



(b) Complete link dendrogram.

单连接 v.s. 全连接



组平均

- 现在考虑层次聚类的组平均版本。
- 此情况下，两个簇的距离定义为两个簇形成的所有不同点对的距离平均值。
  - 属于同一个簇的不同点对的距离，也计算入内

$$dis-ga(C_i,C_j)=\frac{1}{(N_i+N_j)(N_i+N_j-1)}\sum_{d_m\in C_i\cup C_j}\sum_{d_n\in C_i\cup C_j,d_n\neq d_m}dis\left(\vec{d}_m,\vec{d}_n\right)$$

the **normalized** vector of d_m

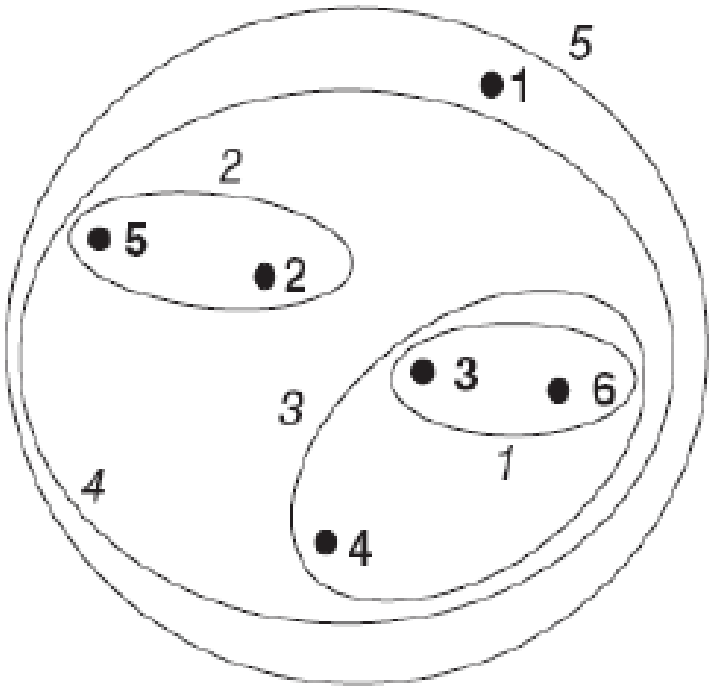
clusters i and j

the number of documents in cluster i

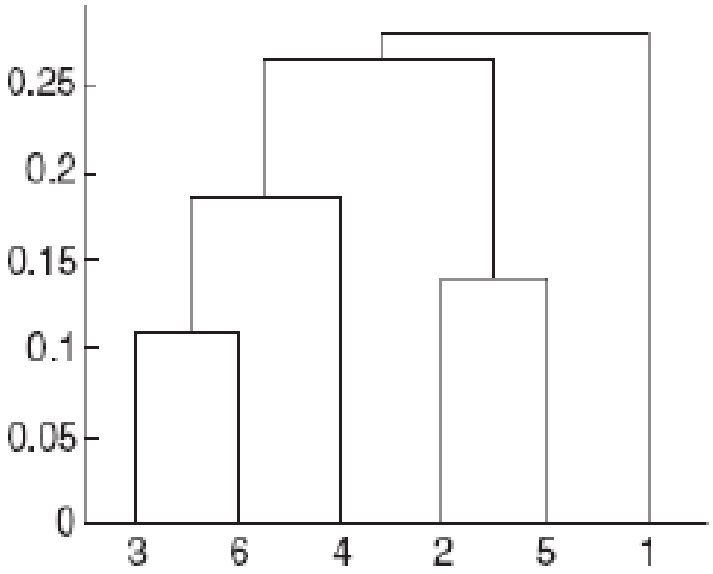
- 这是全连接和单连接的折中版本。

组平均

- 下图展示了样例数据上应用组平均的结果。



(a) Group average clustering.



(b) Group average dendrogram.

关键要点

- 当需要多层结构时，层次聚类比较有效。
- 但是，计算耗时比较久，存储开销比较大。
- 另外，一旦决定了合并两个簇，之后难以撤销。

分裂式层次聚类

- 简要过程：
  1. 初始所有样本都在一个簇中；
  2. 采用非层次聚类算法（如*K*-means）对每个簇进行分裂（如划分为*k*个子簇）；
  3. 重复步骤2，直到每个簇中只有1个样本，或者最近的两个簇之间的距离大于某个人工给定的阈值.

- Since we need a second, flat clustering algorithm as a "subroutine", top-down clustering is conceptually more complex than bottom-up clustering.
  - But … if we do not need a complete hierarchy (all the way down to leaves), top-down clustering with a fixed number of levels can be very efficient, using an effective flat algorithm (e.g., *K*-means).

分裂式层次聚类

- 分裂式层级聚类的每一次分割需要关注两个方面：一是选哪一个簇来分割；二是如何分割。关于簇的选取，通常采用一些衡量松散程度的度量值来比较，例如簇中距离最远的两个数据点之间的距离，或者簇中所有节点相互距离的平均值等，直接选取最“松散”的一个簇来进行分割。而分割的方法也有多种，比如采用普通的K-means来进行二类聚类或者采用如下方法：
  - 待分割的簇记为G，在 G 中取出一个到其他点的平均距离最远的点 x ，构成新簇H；
  - 在 G 中选取这样的点 x' ，x' 到 G 中其他点的平均距离减去 x' 到 H 中所有点的平均距离这个差值最大，将其归入 H 中；
  - 重复上一个步骤，直到差值为负。

基于频繁项集的层次聚类

#	Document name	Feature vector (flow, form, layer, patient, result, treatment)
1	cisi.1	(0 1 0 0 0 0)
2	cran.1	(1 1 1 0 0 0)
3	cran.2	(2 0 1 0 0 0)
4	cran.3	(2 1 2 0 3 0)
5	cran.4	(2 0 3 0 0 0)
6	cran.5	(1 0 2 0 0 0)
7	med.1	(0 0 0 8 1 2)
8	med.2	(0 1 0 4 3 1)
9	med.3	(0 0 0 3 0 2)
10	med.4	(0 0 0 6 3 3)
11	med.5	(0 1 0 4 0 0)
12	med.6	(0 0 0 9 1 1)

Benjamin Chin Ming Fung. Hierarchical Document Clustering Using Frequent Itemsets. 西蒙弗雷泽大学. 硕士学位论文, 2002

基于频繁项集的层次聚类

- 词的最小支持度阈值为：35%

Global frequent itemset	Global support
{flow}	42%
{form}	42%
{layer}	42%
{patient}	50%
{result}	42%
{treatment}	42%
{flow, layer}	42%
{patient, treatment}	42%

基于频繁项集的层次聚类

- 簇的最小支持度阈值为：70%

Cluster	Documents in cluster	Cluster frequent items & their cluster supports (CS)
C(flow)	cran.1, cran.2, cran.3, cran.4, cran.5	{flow, CS=100%}, {layer, CS=100%}
C(form)	cisi.1, cran.1, cran.3, med.2, med.5	{form, CS=100%}
C(layer)	cran.1, cran.2, cran.3, cran.4, cran.5	{layer, CS=100%}, {flow, CS=100%}
C(patient)	med.1, med.2, med.3, med.4, med.5, med.6	{patient, CS=100%}, {treatment, CS=83%}
C(result)	cran.3, med.1, med.2, med.4, med.6	{result, CS=100%}, {patient, CS=80%}, {treatment, CS=80%}
C(treatment)	med.1, med.2, med.3, med.4, med.6	{treatment, CS=100%}, {patient, CS=100%}, {result, CS=80%}
C(flow, layer)	cran.1, cran.2, cran.3, cran.4, cran.5	{flow, CS=100%}, {layer, CS=100%}
C(patient, treatment)	med.1, med.2, med.3, med.4, med.6	{patient, CS=100%}, {treatment, CS=100%}, {result, CS=80%}

基于频繁项集的层次聚类

$$Score(C_i \leftarrow doc_j) = [\sum_x n(x) * cluster\_support(x)] - [\sum_{x'} n(x') * global\_support(x')]$$

where x represents a global frequent item in doc_j and the item is also cluster frequent in C_i, x' represents a global frequent item in doc_j that is not cluster frequent in C_i, $n(x)$ is the weighted frequency of x in the feature vector of doc_j, and $n(x')$ is the weighted frequency of x' in the feature vector of doc_j.

Let us explain the rationale behind the score function. The first term of the function rewards cluster C_i if a global frequent item x in doc_j is cluster frequent in C_i. In order to capture the importance (weight) of item x in different clusters, we multiply the frequency of x in doc_j by its cluster support in C_i. The second term of the function penalizes cluster C_i if a global frequent item x' in doc_j is not cluster frequent in C_i. The frequency of x' is multiplied by its global support which can be viewed as the importance of x' in the entire document set or as the weight of the penalty on this item. This part encapsulates the concept of dissimilarity into the score.

基于频繁项集的层次聚类

$$Score(C(patient) \leftarrow med.6) = 9 * 1 + 1 * 0.83 - 1 * 0.42 = 9.41$$

$$Score(C(result) \leftarrow med.6) = 10.6$$

$$Score(C(treatment) \leftarrow med.6) = 10.8$$

$$Score(C(patient, treatment) \leftarrow med.6) = 10.8$$

Cluster	Documents in cluster	Cluster frequent items & their cluster supports (CS)
C(flow)	cran.1, cran.2, cran.3, cran.4, cran.5	{flow, CS=100%}, {layer, CS=100%}
C(form)	cisi.1	{form, CS=100%}
C(layer)		none
C(patient)	med.5	{patient, CS=100%}, {treatment, CS=83%}
C(result)		none
C(treatment)		{treatment, CS=100%}, {patient, CS=100%}, {result, CS=80%}
C(flow, layer)		none
C(patient, treatment)	med.1, med.2, med.3, med.4, med.6	{patient, CS=100%}, {treatment, CS=100%}, {result, CS=80%}

基于频繁项集的层次聚类

The criterion for selecting the best parent is similar to choosing the best cluster for a document. We first merge all the documents in the subtree of C_i into a single conceptual document doc(C_i), and then compute the score of doc(C_i) against each potential parent. The one which has the highest score would become the parent of C_i.