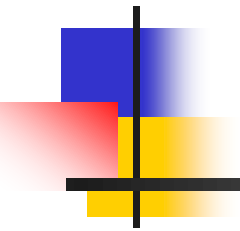
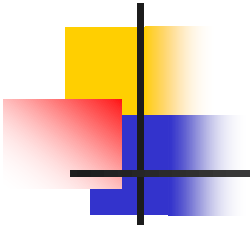


专题二　聚 类

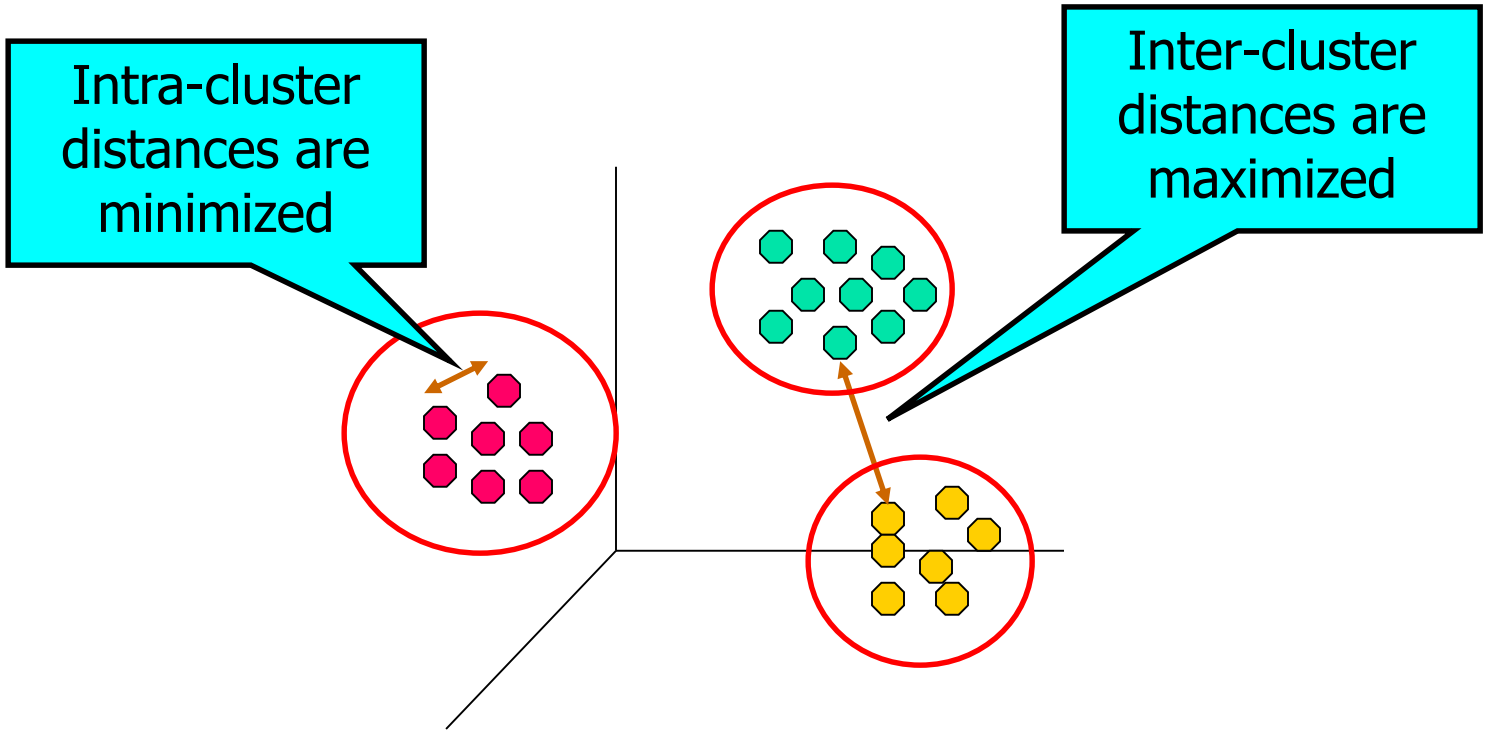


目录

- 聚类的定义
- 聚类的类型
- 两种常见的聚类方法
  - K-means
  - 层次聚类

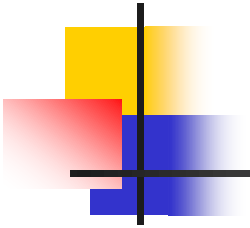
1. 聚类的定义

- 从对象集中发现对象组，使得处于同一组中的对象相互之间是相似的（或相关的），而不同组中的对象是不同的（不相关的）.



1. 聚类的定义——应用

- **聚类的部分应用场景**
  - 浏览相关文档的组,
  - 具有类似功能的基因和蛋白组
  - 有类似价格浮动的股票组.



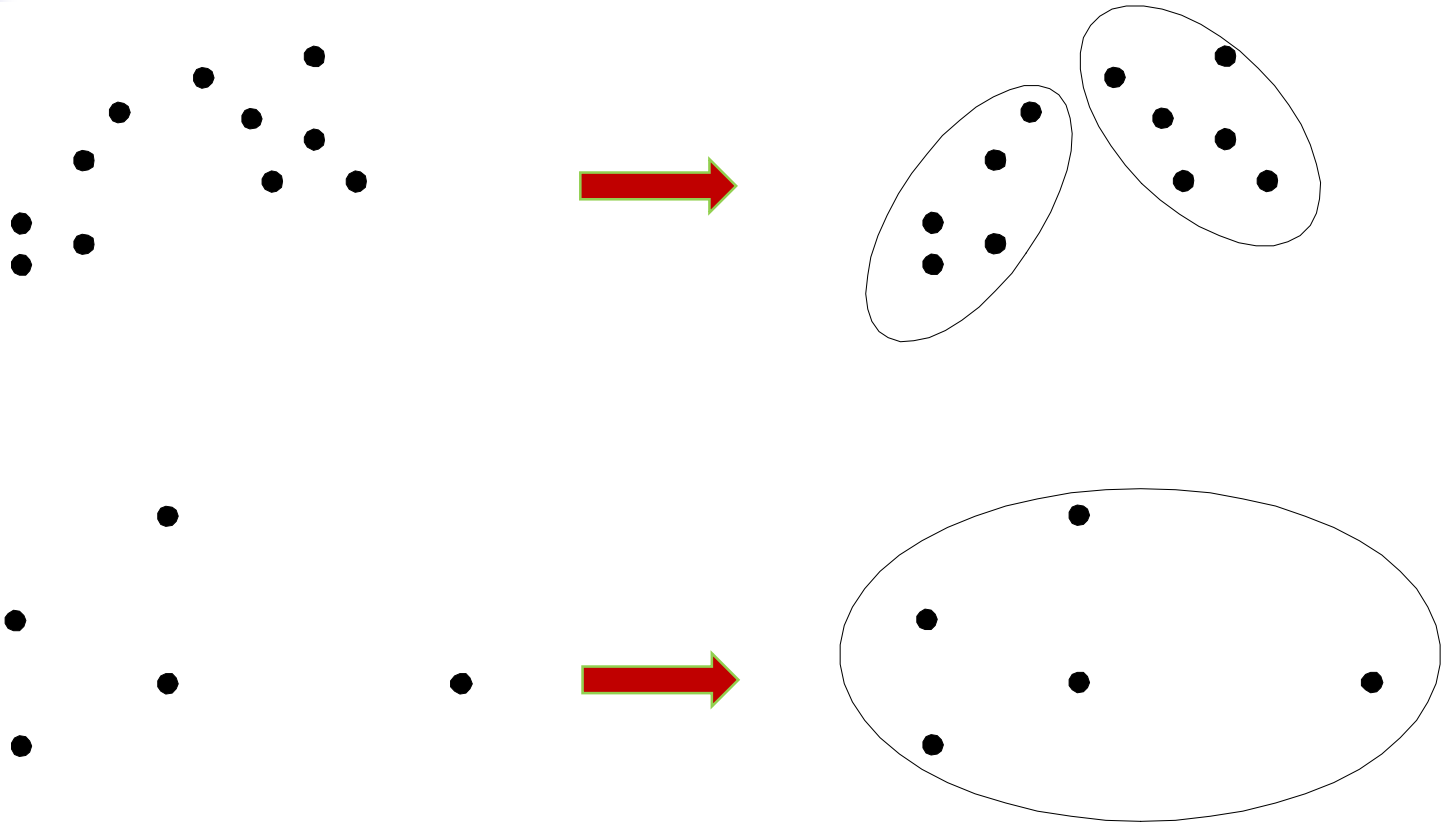
目录

- 聚类的定义
- 聚类的类型
- 两种常见的聚类方法
  - K-means
  - 层次聚类

2. 聚类的类型

- 整个簇集合通常称为**聚类**.
- **不同类型的聚类之间最常讨论的差别是**：簇的集合是层次的(嵌套的)，还是划分的(非嵌套的).
- 划分的**聚类**:
  - 简单地将数据对象集划分成不重叠的子集(簇)，使得每个数据对象恰在一个子集中，如上页中的例子。
- 层次的**聚类**:
  - 是嵌套簇的集族(允许簇具有子簇)，组织成一棵树。

2. 聚类的类型——划分的聚类

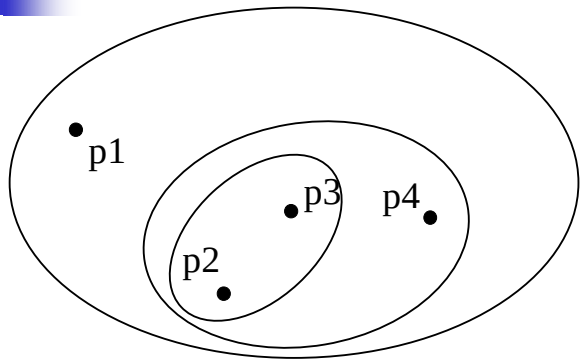


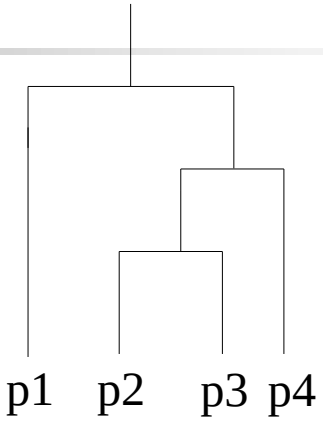
Original Points

A Partitional Clustering

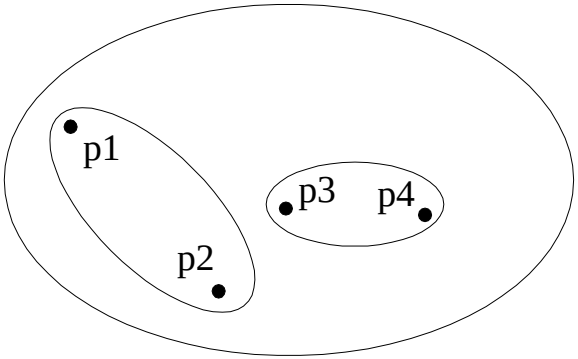
每个数据对象属于且仅属于一个簇

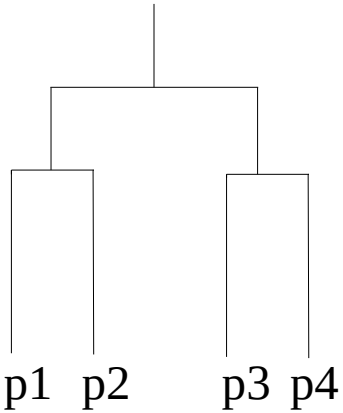
2. 聚类的类型——层次的聚类



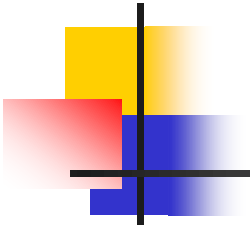


系统树图（一种表示亲缘相似关系的树状图解）





除叶节点外，树中每一个节点（簇）都是其子女（子簇）的并，而树根是包含所有对象的簇。



目录

- 聚类的定义
- 聚类的类型
- 两种常见的聚类方法
  - K-means
  - 层次聚类

3.1 K-均值聚类——一般过程

- 一种划分聚类方法.
- 每个簇被指定一个质心（中心点），代表相应的簇.
- 然后，每个点被**指派**到最近的质心，而指派到同一个质心的点集形成一个簇. 之后，根据指派到簇的点，**更新**每个簇的质心. 重复**指派和更新过程**，直到质心不发生变化.
- 簇的数量 K，必须被用户指定.
- 基本算法非常简单.

3.1 K-均值聚类——一般过程

- **基本K-均值算法**

选择K个点作为初始质心

repeat

将每个点指派到最近的质心，形成K个簇。

重新计算每个簇的质心。

Until 质心不再发生变化。

3.1 K-均值聚类——细节

- 初始化质心通常是任意被选择的.
  - 通常是任意选择的，如随机初始化.
  - 使用其它方法初始化（层次聚类）。
- 指派数据点到最近的质心
  - 典型地,质心是簇中所有点的均值.
  - ‘最近’ 可用欧几里德距离、余弦相似度、相关性等度量.
  - 指派最近质心所代表的簇中.

3.1 K-均值聚类——细节

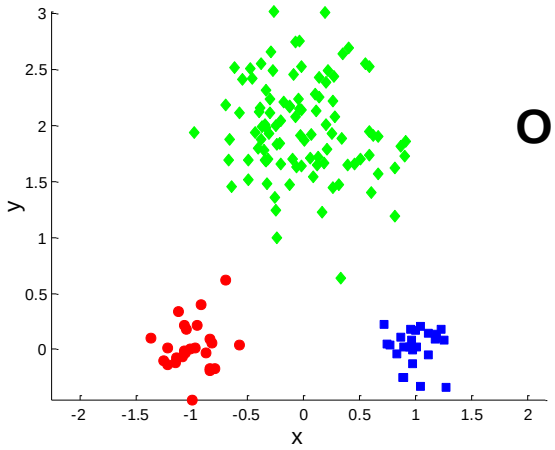
- 质心和目标函数
  - 质心是由簇中的数据点计算而得到的. 当数据点重新指派后， 簇的质心可能发生变化，需要重新计算.
  - 聚类的目标通常用一个目标函数表示。 目标函数的形式依赖于邻近性度量. 如 基于欧几里得距离的误差平方和.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

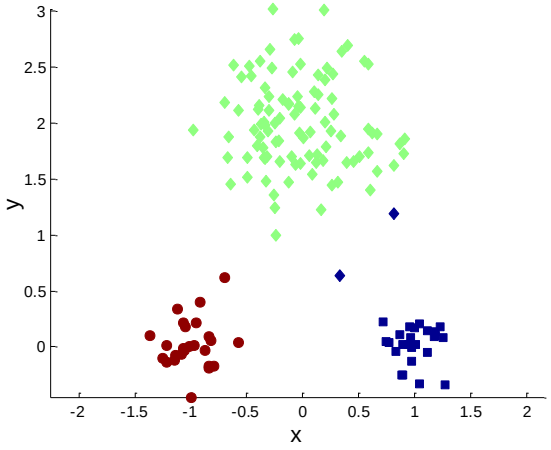
3.1 K-均值聚类——细节

- 基于上述相似性度量，K-均值算法随着迭代的进行而逐渐收敛.
- 大部分收敛发生在最初的几次迭代中.
  - 通常停止条件被改变为 ‘直到几乎没有点改变其分簇’.

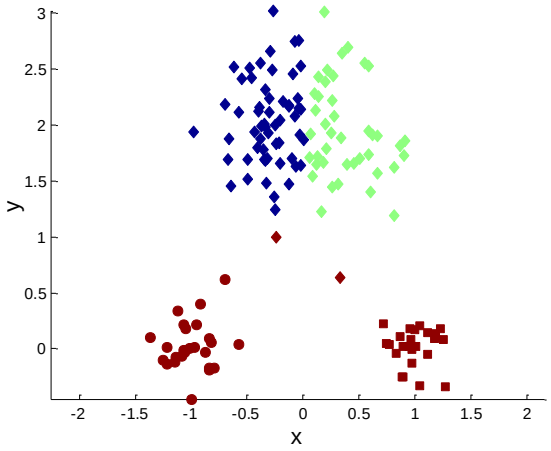
3.1 K-均值聚类——示例



Original Points



Optimal Clustering



Sub-optimal Clustering

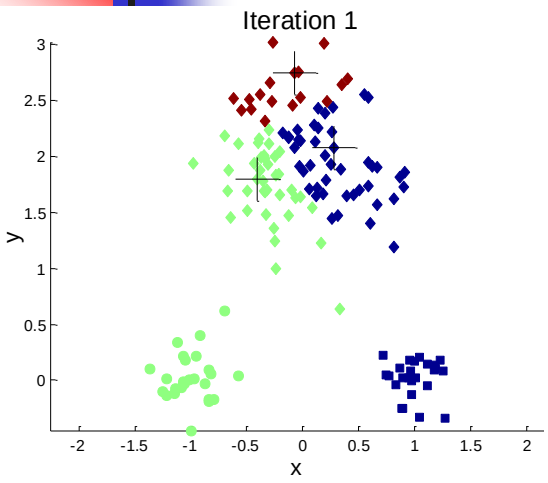
3.1 K-均值聚类——评价

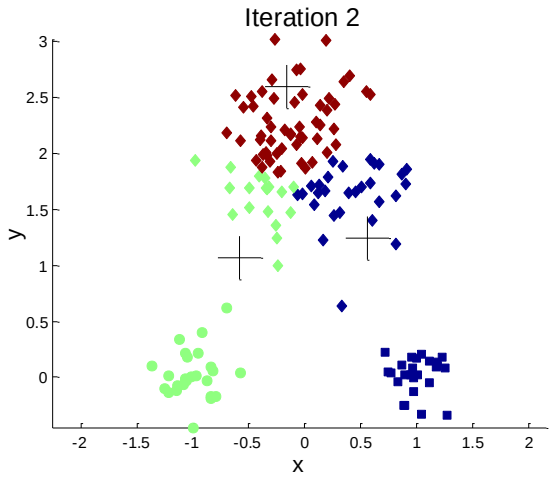
- 最常用的度量是误差的平方和（SSE）
  - 对每个点，我们求它到最近簇的质心的距离（误差）.
  - 为得到 SSE，我们求它们的平方和.

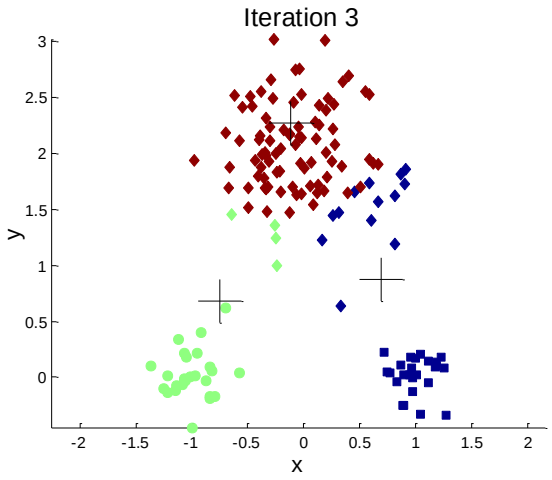
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

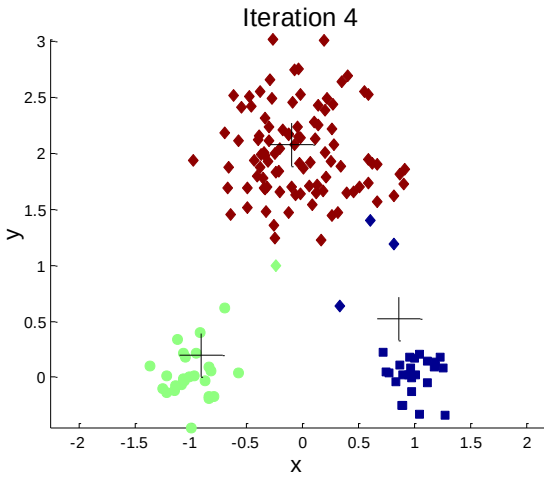
  - x 是簇 C_i 中的数据点而 m_i 簇 C_i 的代表点.
    - m_i 可对应为簇的中心（均值）.
  - 给定由2次运行K-均值产生的2个不同的簇集，我们更喜欢误差的平方和最小的那个.

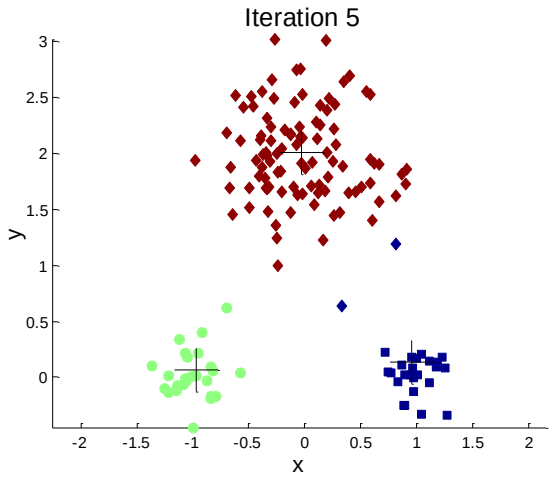
3.1 K-均值聚类——初始质心

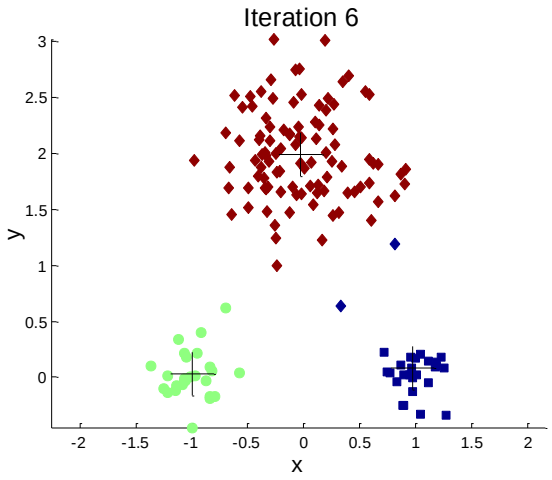




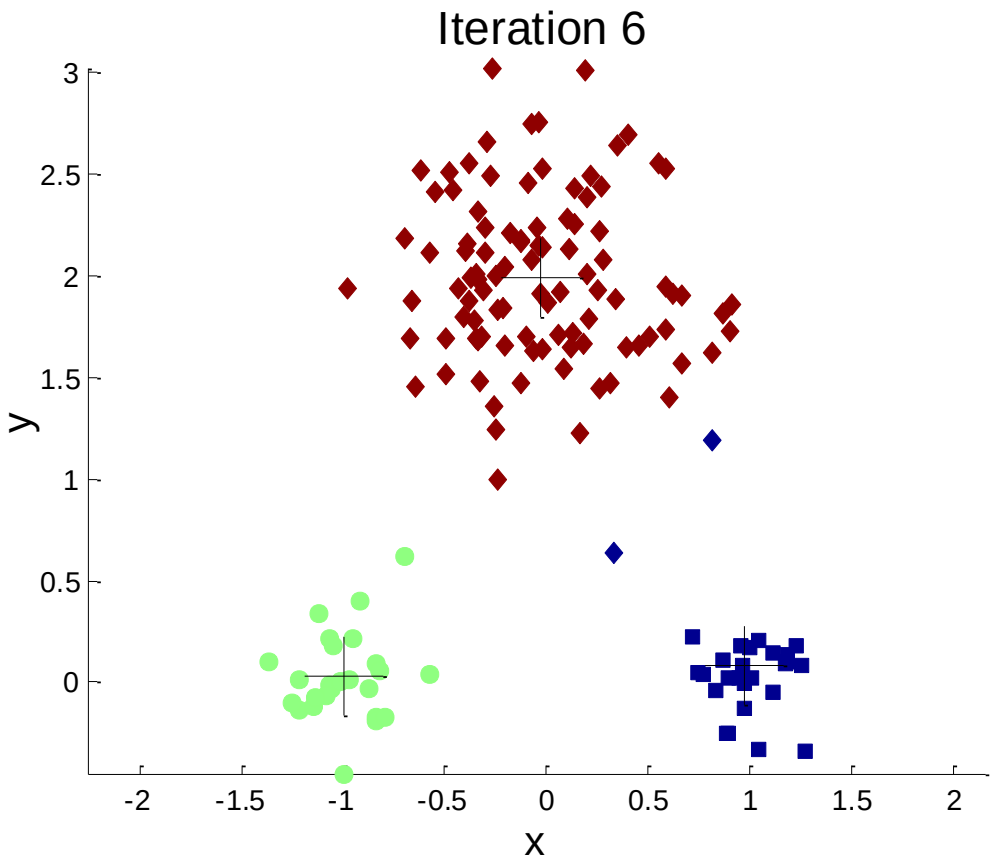




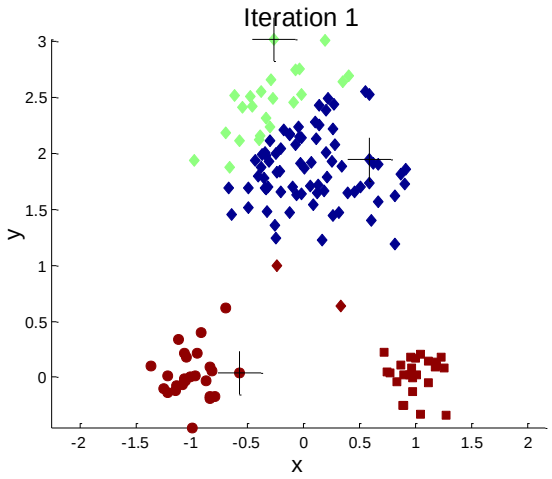


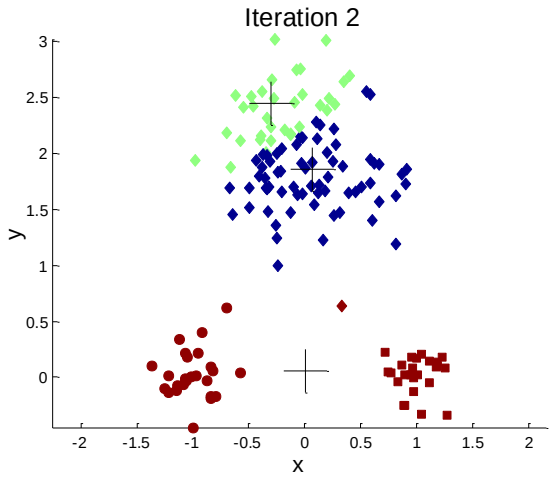


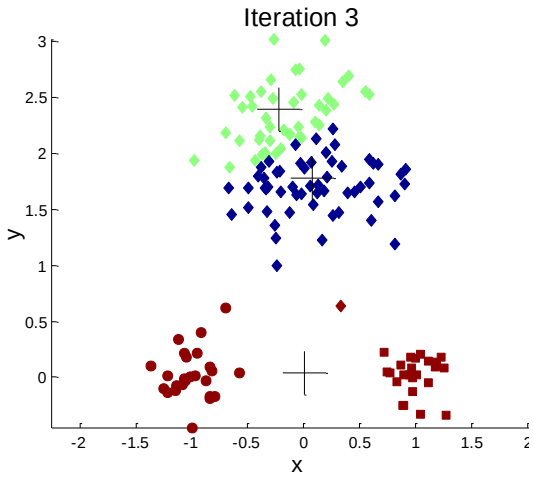
3.1 K-均值聚类——初始质心

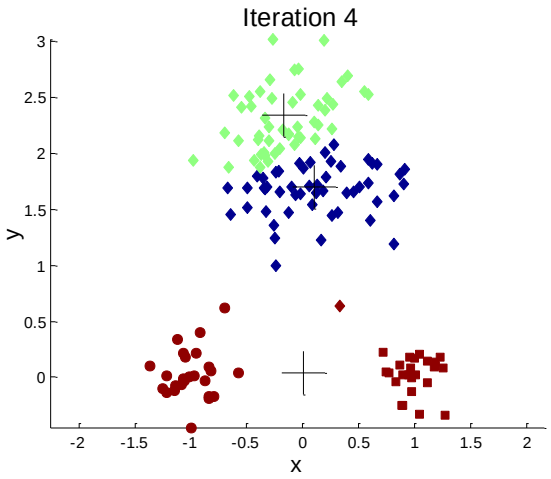


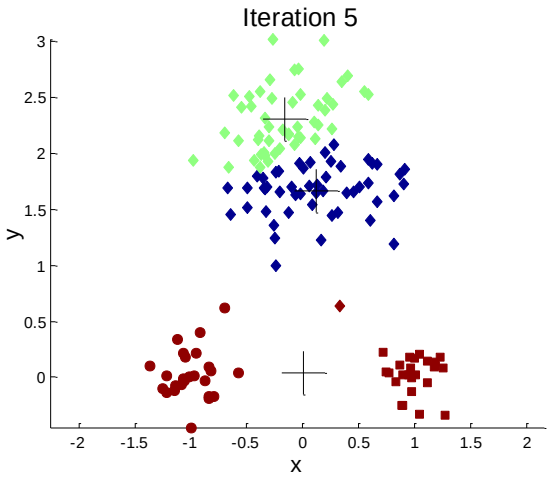
3.1 K-均值聚类——初始质心



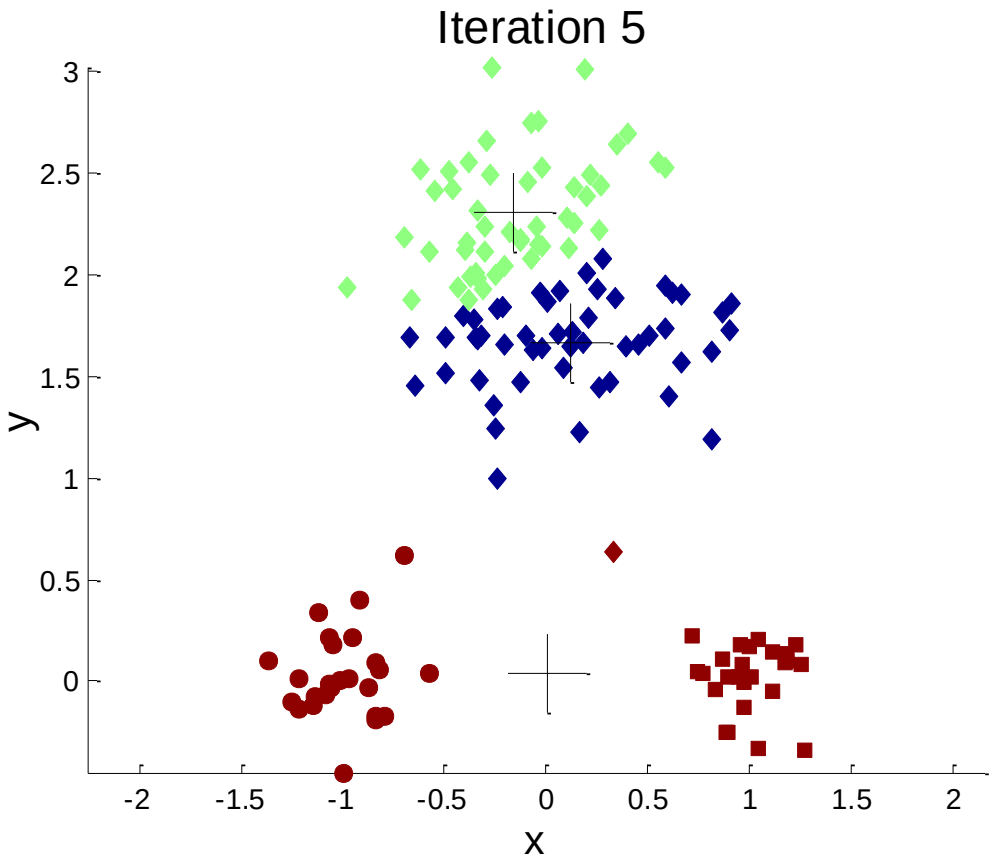








3.1 K-均值聚类——初始质心



3.1 K-均值聚类——初始质心

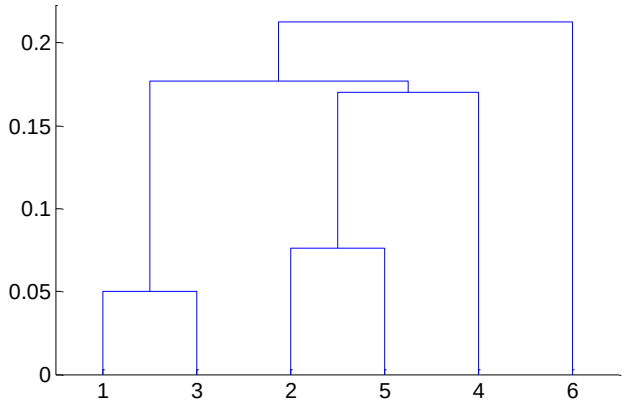
- 多次运行
  - 有帮助，但概率不会总在你这边.
- 采样并使用层次聚类来决定初始质心.
- 选择多于 k 的初始质心，然后从这些初始质心中再进行选择.
  - 选择原则：进行相差最大的分割.
- 使用后处理来“修补”所产生的簇集.
- 二分 (Bisecting) K-均值
  - 不太受初始化问题的困扰.

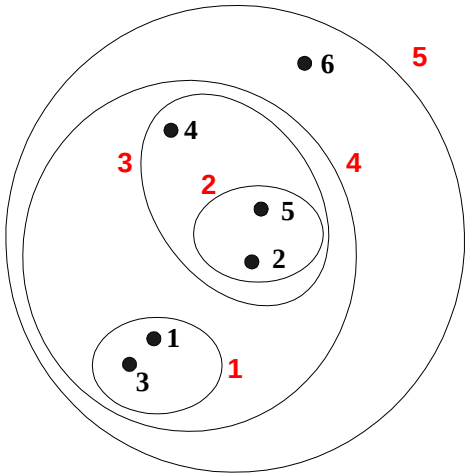
3.1 K-均值聚类——局限性

- 当簇具有下列不同特性（K-均值并不适合所有的数据类型），K-均值很难找到”自然的”簇.
  - 大小(不同尺寸)
  - 密度(不同密度)
  - 非球形簇
- 当数据包含离群点，K-均值也存在问题，需要进行离群点检测和删除.
- K-均值仅限于具有中心(质心)概念的数据.

3.2 层次聚类——概述

- 层次聚类可以产生一组能组织成一个层次树的嵌套簇图，是第2种重要的聚类方法.
- 能被可视化为一个树状图.
  - 该图记录了簇-子簇联系和簇合并（凝聚）或分裂的次序.





3.2 层次聚类——优势

- 不必假设具体的簇数量.
  - 在适当等级截断树状图，便获得任何理想数量的簇.
- 它们可能符合意味深长的分类法.
  - 例如，在生物科学中（如.，动物界，语系发展，...）
- 能够反映聚类的过程

3.2 层次聚类——类型

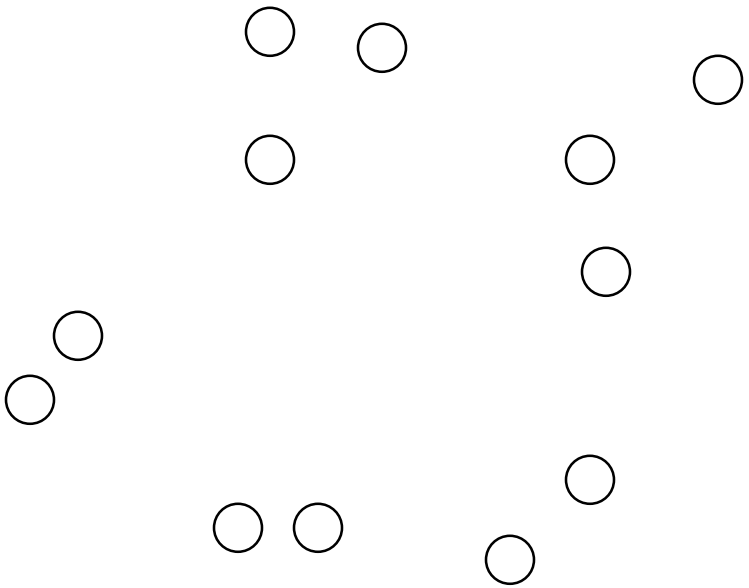
- 2种主要类型：
  - 凝聚的层次聚类：
    - 从点作为个体簇开始.
    - 每一步合并2个最接近的簇，直到仅剩1个（or k个）簇. 需要定义邻近性概念.
  - 分裂的层次聚类：
    - 从包含所有点的一个簇开始，
    - 每一步分裂一个簇，直到仅剩单点簇（or 存在 k个簇）. 需要确定每一步分裂哪个簇，以及如何分裂.
- 传统的层次聚类算法使用一种相似性或距离度量.
  - 每次合并或分裂一个簇.

3.2.1 凝聚的聚类算法——过程

- 较流行的层次聚类技术.
- 基本算法的表达很直接了当.
  1. 如果需要,计算邻近度矩阵.
  2. 令每个数据点为一个簇;
  3. Repeat
  4. 合并最接近的2个簇
  5. 更新邻近度矩阵,以反映新簇与原来其它簇之间的
  6. 邻近性.
  7. Until 仅剩1个簇.
- 关键操作是2个簇的邻近度计算.
  - 簇之间距离的不同定义方法导致了不同的算法.

3.2.1 凝聚的聚类算法——示例

- 开始时，每个数据点为一个簇，存在1个初始的邻近度矩阵.

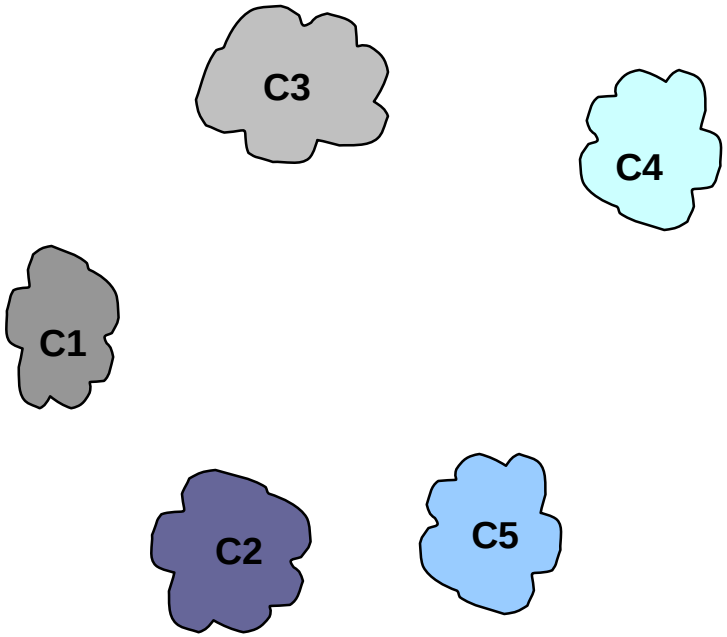


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

邻近度矩阵
(Proximity Matrix)

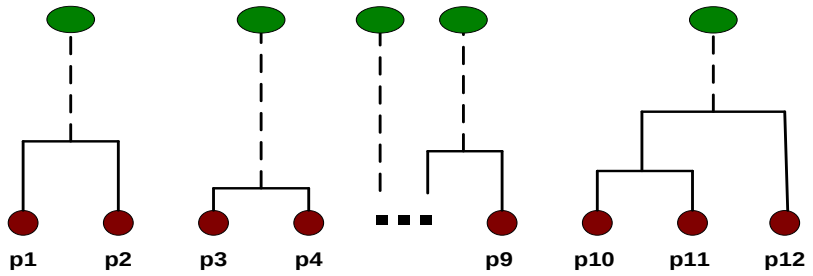
3.2.1 凝聚的聚类算法——示例

- 在经过一些合并后，我们得到一些簇.



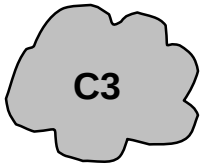
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

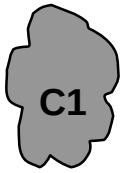
Proximity Matrix

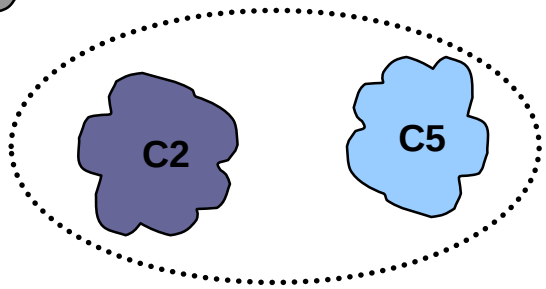


3.2.1 凝聚的聚类算法——示例

- 我们想合并2个最近的簇 (C2和C5) 并且更新邻近度矩阵.

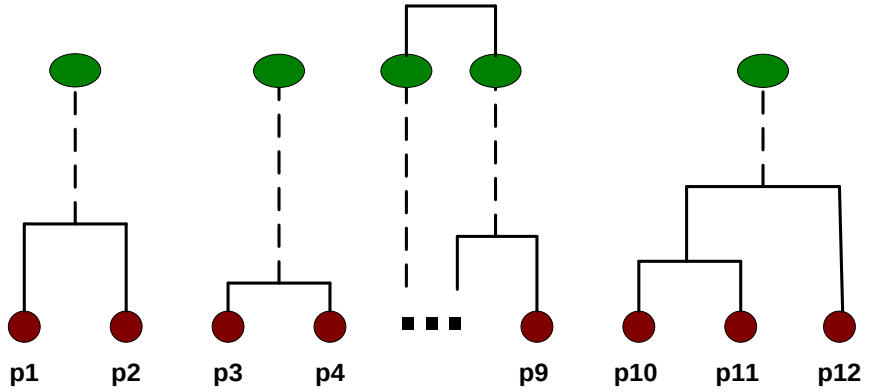






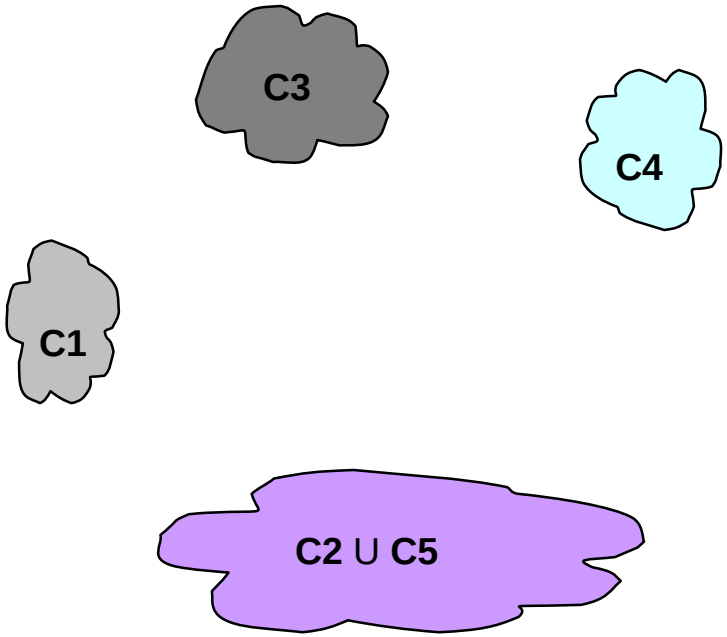
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



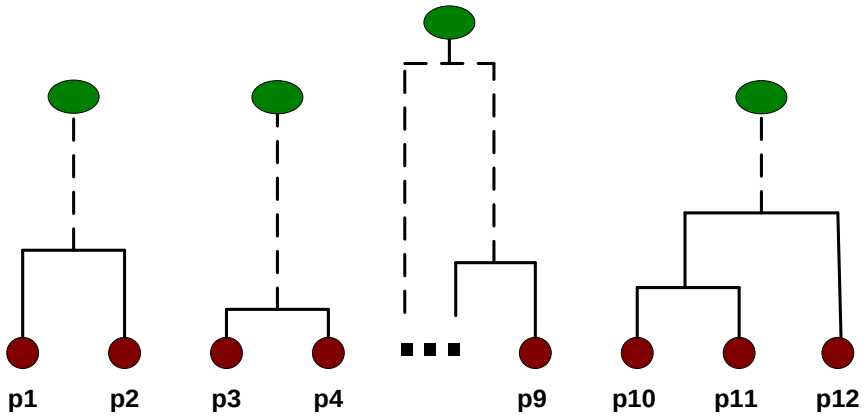
3.2.1 凝聚的聚类算法——示例

- 问题是 “我们如何更新邻近度矩阵?”

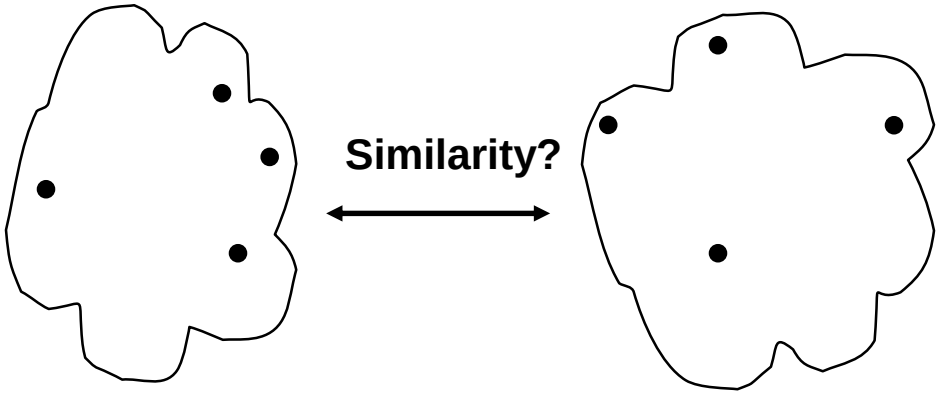


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



3.2.1 凝聚的聚类算法——邻近性



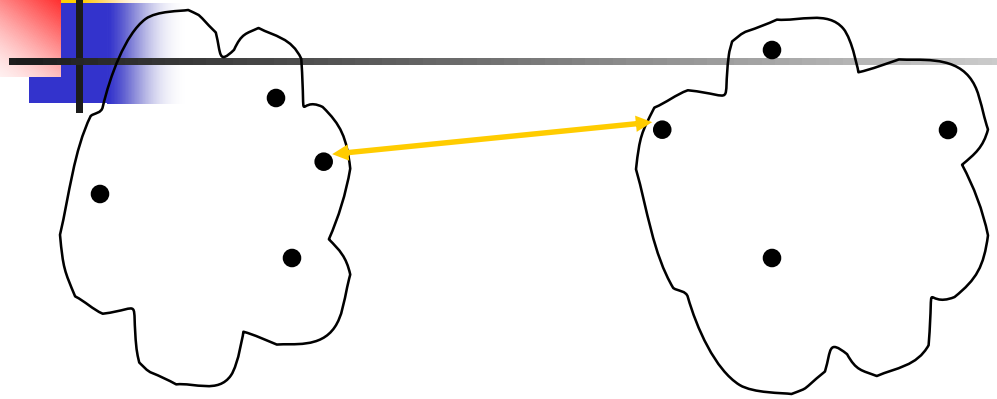
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

. .

Proximity Matrix

- 基于图的观点
  - MIN
  - MAX
  - 组平均 (Group Average)

3.2.1 凝聚的聚类算法——邻近性



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

. .

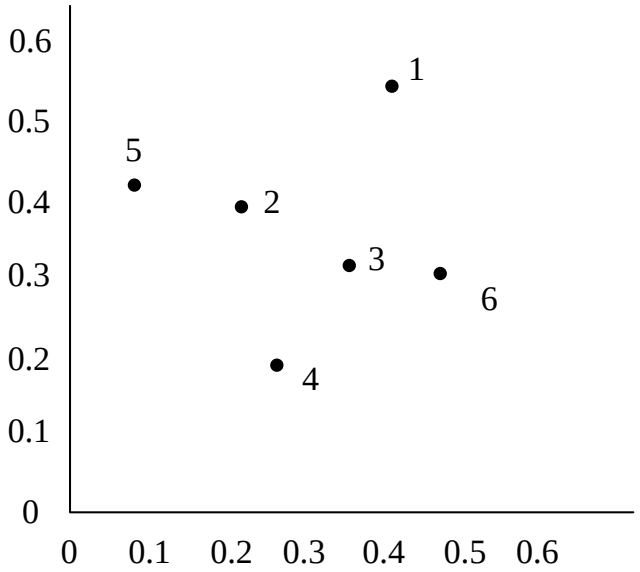
Proximity Matrix

- MIN（单链）：定义簇的邻近度为不同簇的2个最近的点之间的邻近度. 或不同的节点子集中2个节点之间的最短边.

示例：样本数据

表　6个点的x y坐标

点	x 坐标	y 坐标
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022



6个点的欧几里德距离矩阵

	p1	p2	p3	P4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

聚类邻近度——MIN（单链）

- 2个簇的邻近度定义为2个不同簇中任意2点之间的最短距离（最大邻近度）. 基于2个不同簇中最大相似的点(最短距离), 只需要加一条链.
  - 从所有点作为单点簇开始, 每次在点之间加上一条链, 最短的链先加, 则这些链将点合并成簇.

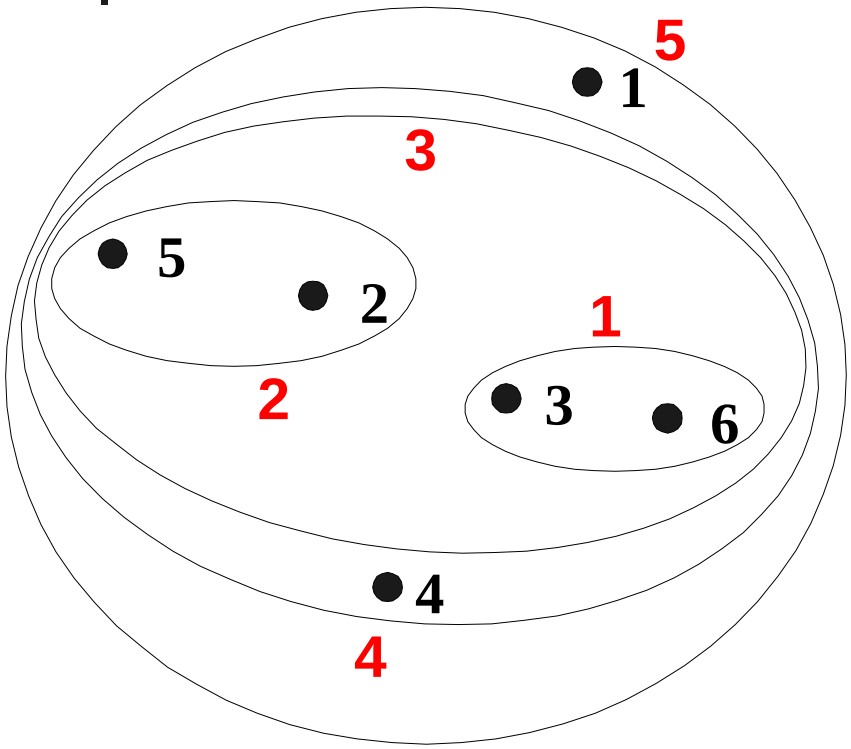
1. $dist(3,6) = 0.11;$

3. $dist(\{3,6\},\{2,5\}) = \min(dist(3,2), dist(6,2), dist(3,5), dist(6,5))$

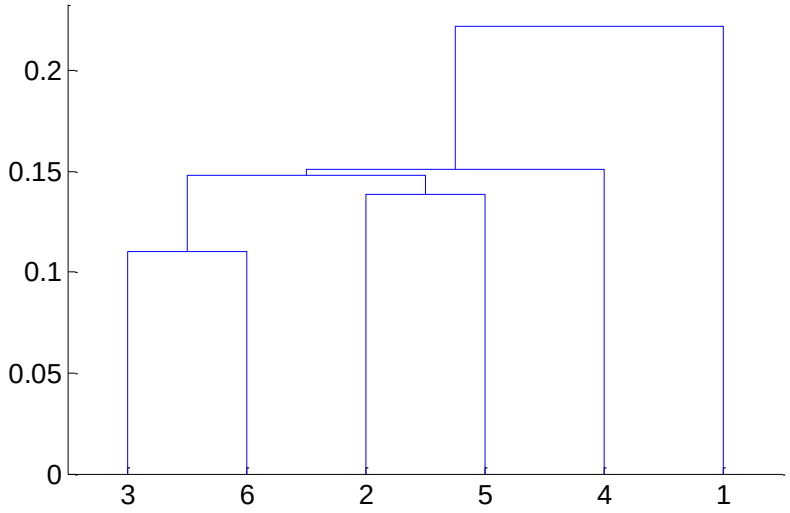
$= \min(0.15, 0.25, 0.28, 0.39) = 0.15$

.....

层次聚类—— MIN

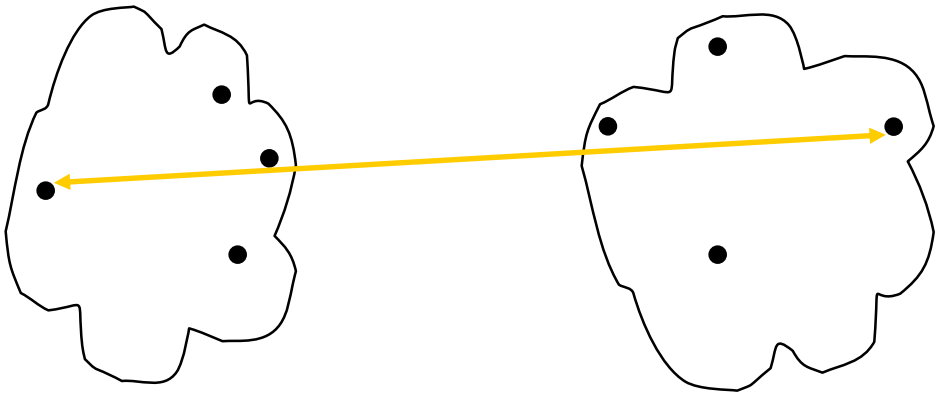


Nested Clusters



Dendrogram

3.2.1 凝聚的聚类算法——邻近性



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

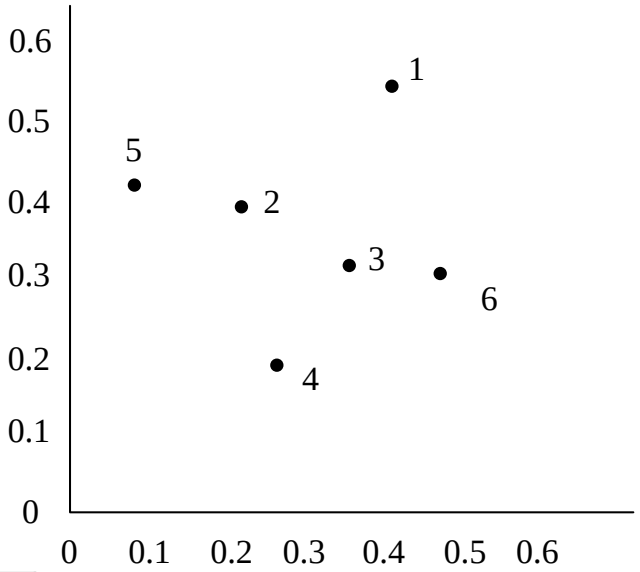
Proximity Matrix

- MAX（全链）：定义簇的邻近度为不同簇的2个最远的点之间的邻近度。或不同的节点子集中2个节点之间的最长边.

示例：样本数据

表　6个点的x y坐标

点	x 坐标	y 坐标
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022



6个点的欧几里德距离矩阵

	p1	p2	p3	P4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

聚类邻近度—— MAX

- 2个簇的邻近度定义为2个不同簇中任意2点之间的最长距离（最小邻近度）. 基于2个不同簇中最小相似的点(最长距离),需要加所有的链.
- 从所有点作为单点簇开始,每次在点之间加上一条链,最短的链先加,则一组点直到其中所有的点都完全被连接(即形成团)才形成一个簇.
- （例 第3步）

$$dist(\{3,6\},\{4\}) = \max(dist(3,4).dist(6,4)) = \max(0.15,0.22) = 0.22.$$

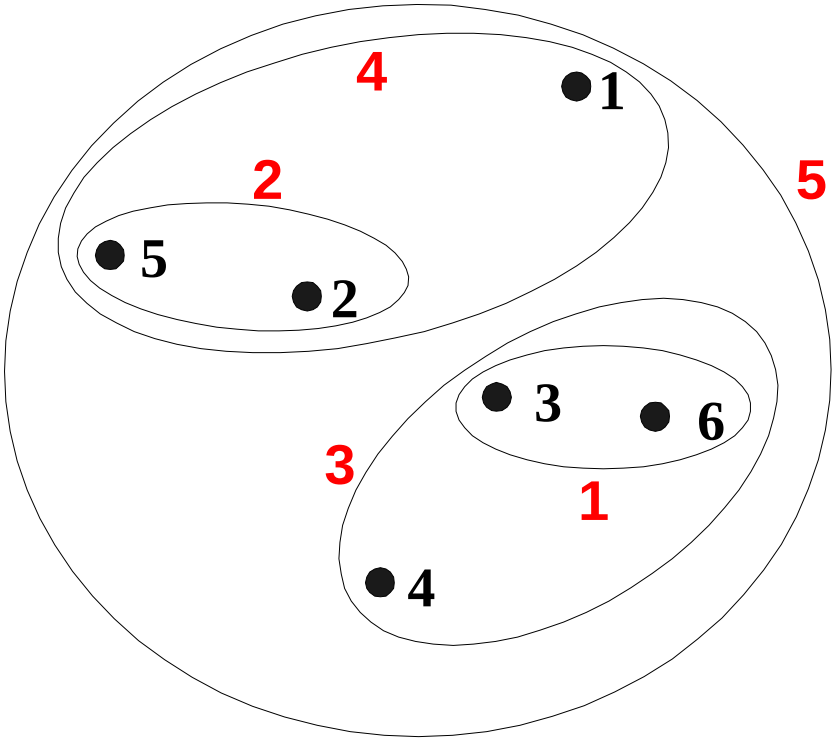
$$dist(\{3,6\},\{2,5\}) = \max(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \quad \surd$$

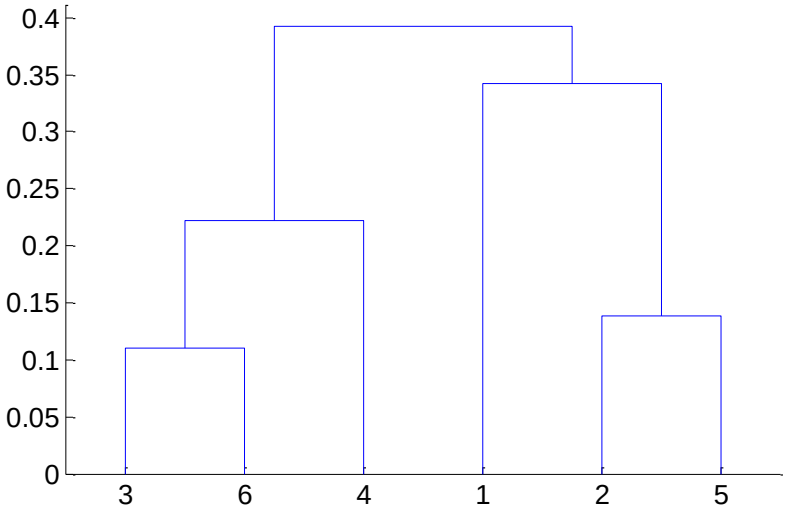
$$= \max(0.15,0.25,0.28,0.39) = 0.39.$$

$$dist(\{3,6\},\{1\}) = \max(dist(3,1).dist(6,1)) = \max(0.22,0.23) = 0.23.$$

.....

层次聚类—— MAX（全链）

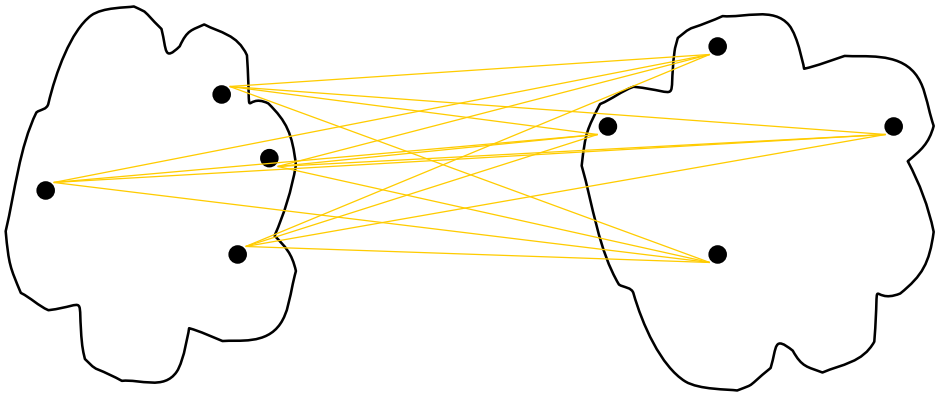




Nested Clusters

Dendrogram

3.2.1 凝聚的聚类算法——邻近性



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

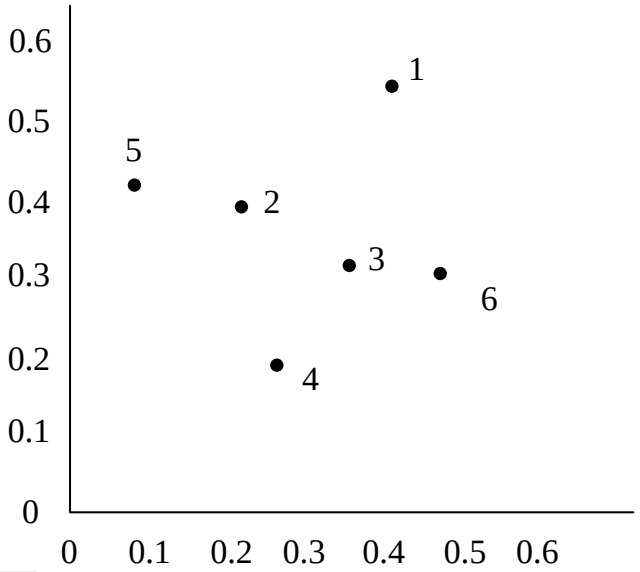
Proximity Matrix

- 组平均：定义簇的邻近度为取自不同簇的所有点对的逐对邻近度和的平均.

示例：样本数据

表　6个点的x y坐标

点	x 坐标	y 坐标
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022



6个点的欧几里德距离矩阵

	p1	p2	p3	P4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

聚类邻近度——组平均

- 2个簇的邻近度定义为：不同簇的所有点对的平均邻近度.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- 下图显示组平均用于6个点数据例子的结果.（例 第4步）

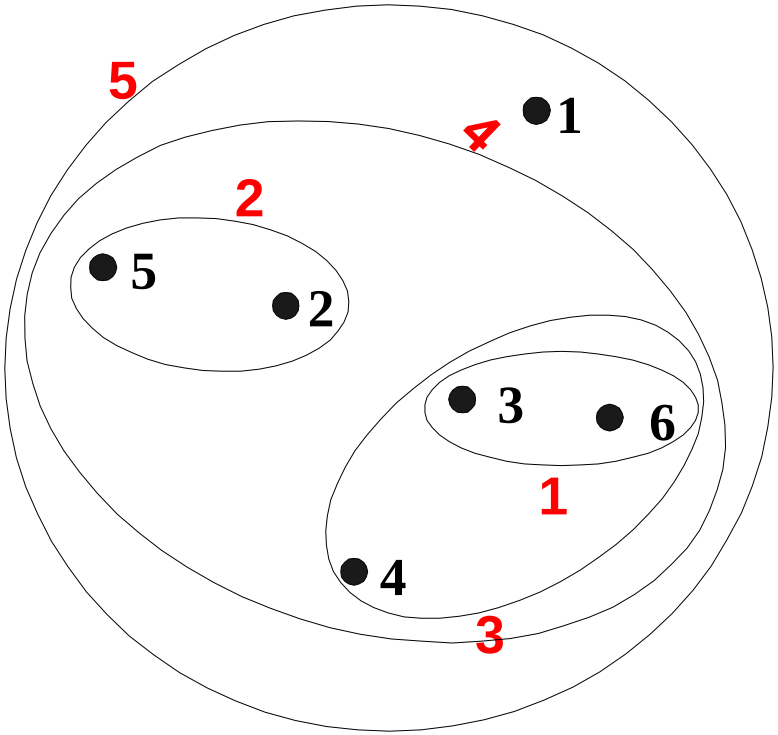
$$dist(\{3,6,4\},\{1\}) = (0.22 + 0.37 + 0.23)/(3*1) = 0.28.$$

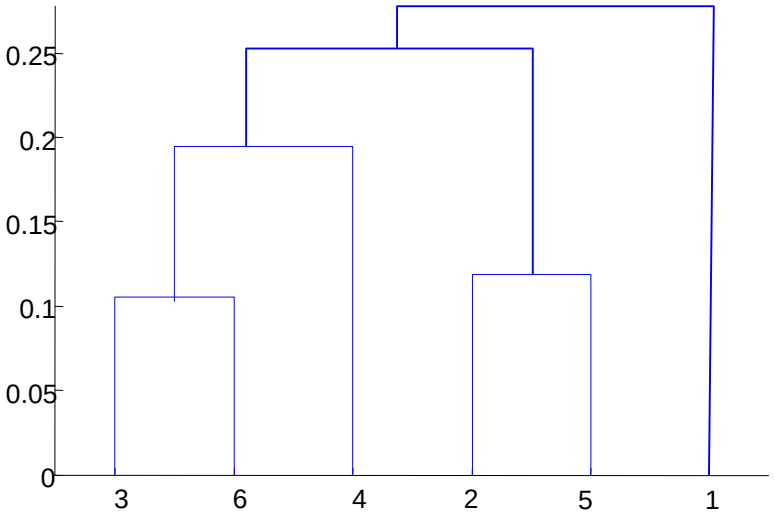
$$dist(\{2,5\},\{1\}) = (0.2357 + 0.3421)/(2*1) = 0.2889.$$

$$dist(\{3,6,4\},\{2,5\}) = (0.15 + 0.28 + 0.25 + 0.39 + 0.2 + 0.29)/(3*2) = 0.26.$$

√

层次聚类——组平均

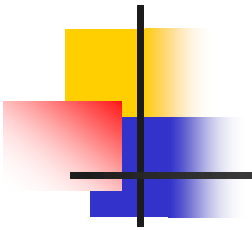




Nested Clusters **Dendrogram**

Min、Max和组平均的优势与局限性

- Min的优势与局限性
  - 优势：能处理非椭圆形状的簇
  - 局限性：对噪声和离群点很敏感
- Max的优势与局限性
  - 优势：对噪声和离群点不太敏感。
  - 局限性：可能使大的簇破裂；偏好球形。
- 组平均的优势与局限性
  - 优势：对噪声和离群点不太敏感。
  - 局限性：偏好球形。



谢谢大家！