课程回顾

- 第1章 绪论
- 第2章 知识发现过程与应用结构
- 第3章 关联规则挖掘理论和算法
- 第4章 分类方法
- 第5章 聚类方法
- 第6章 时间序列和序列模式挖掘
- 第7章 Web挖掘技术
- 第8章 空间挖掘

数据挖掘十大算法

一、 C4.5：分类决策树算法,其核心算法是ID3 算法

二、k-means：k-平均算法是解决聚类问题

三、SVM：支持向量机

四、Apriori：挖掘布尔关联规则频繁项集的算法

五、EM ：最大期望算法

六、PageRank：网页排名、搜索引擎

七、AdaBoost：自适应增强

八、KNN：k-近邻算法

九、Naive Baye：朴素贝叶斯分类器

十、CART：分类回归树

数据分析技术

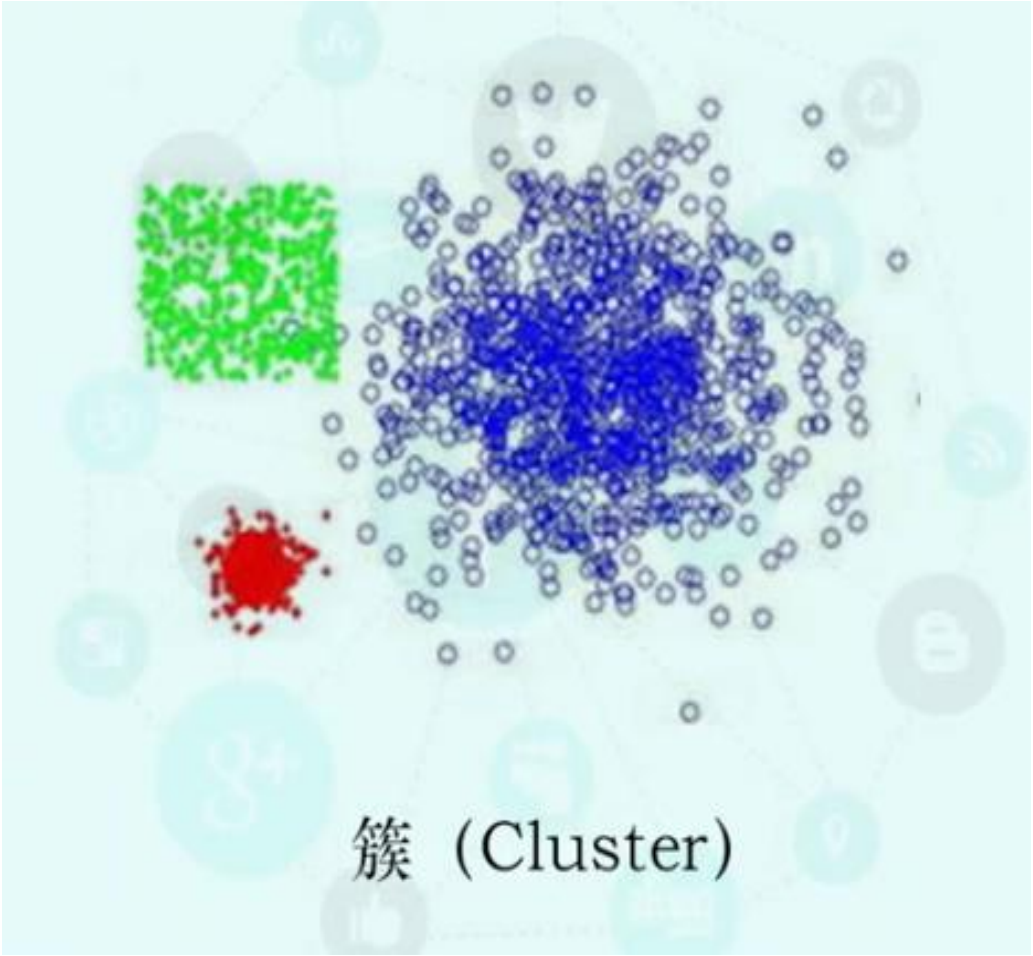
第5章 聚类方法

聚类分析

授课对象：计算机专业学生

聚类分析

- 聚类分析：是指根据给定一组对象的描述信息，发现由具有共同特性的对象构成簇的过程。



聚类方法概述

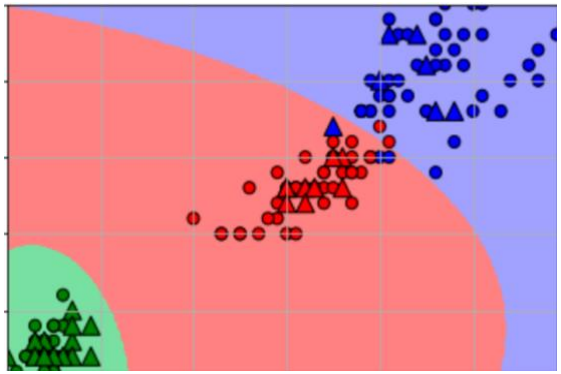
聚类的基本概念

- 在没有训练的条件下，把样本划分为若干类
- 无类别标签的样本
- 无监督学习

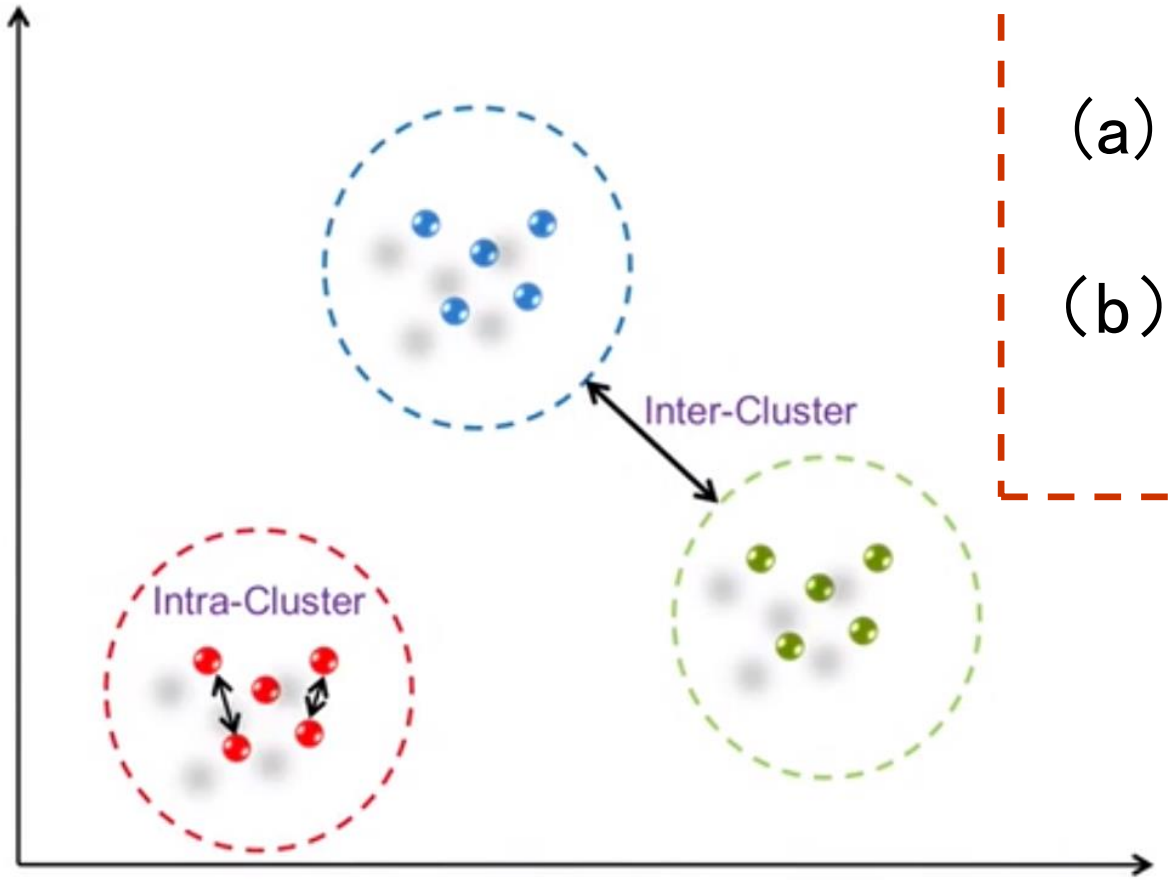
分类的基本概念

- 已知存在哪些类，将每一条记录分别属于哪一类标记出来。
- 有类别标签的样本
- 有监督学习

区别



聚类方法概述



如何评判聚类的好坏：

（a） 类间距离：大

（b）类内距离：小

基于距离的相似度

明可夫斯基距离(Minkowski Distance)是欧式距离和曼哈顿距离的概括，如式(3)所示：

$$d(o_i, o_j) = \sqrt[p]{|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \ldots + |x_{im} - x_{jm}|^p} \quad (3)$$

明可夫斯基距离又称为L_p范式。因此，p=1 时对应曼哈顿距离，又称L_1范式；P=2 时对应欧式距离，又称L_2范式

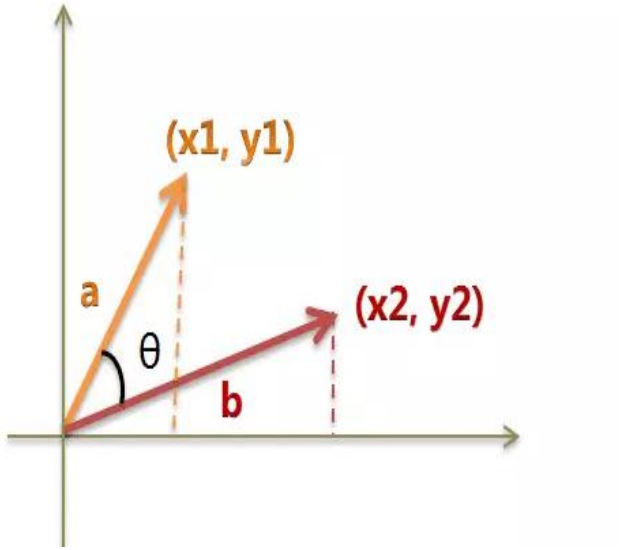
p=正无穷时称为切比雪夫距离，相应公式如式(4) 所示：

$$d(o_i, o_j) = \lim_{p \to \infty} \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}} = \max_{1 \le k \le m} |x_{ik} - x_{jk}| \quad (4)$$

余弦距离

- 用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。

$$sim(X,Y) = cos\theta = \frac{\vec{x} \cdot \vec{y}}{\|x\| \cdot \|y\|}$$



多种属性相似度的综合度量

- 什么时候用余弦距离什么时候用欧式距离呢？
- 欧氏距离体现数值上的绝对差异，而余弦距离体现方向上的相对差异。
- 例如，统计两部剧的用户观看行为，用户A的观看向量为(0,1)，用户B为(1,0)；此时二者的余弦距很大，而欧氏距离很小；分析两个用户对于不同视频的偏好，更关注相对差异，应当使用余弦距离。
- 而当我们分析用户活跃度，以登陆次数(单位：次)和平均观看时长(单：分钟)作为特征时，余弦距离会认为(1,10)、(10,100)两个用户距离很近；但显然这两个用户活跃度是有着极大差异的，此时我们更关注数值绝对差异，应当使用欧氏距离。

余弦距离

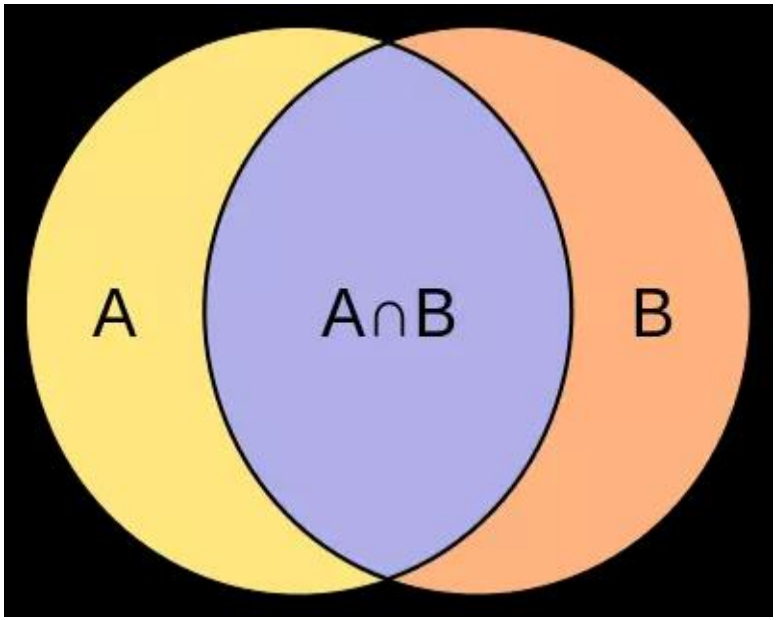
- 余弦相似度特点：
  - 相似度忽略了各个属性值的绝对大小
  - 两个向量中，只要有一个对象在某维度的取值为0，则该维度相当于被忽略

余弦相似度是信息检索中用于衡量文档相似度的主要方法

Jaccard距离

- Jaccard系数：判断两个集合的相似度， jaccard similarity coefficient。非对称二元属性的相似。性给定两个集合A,B jaccard 系数定义为A与B交集的大小与并集大小的比值，jaccard值越大说明相似度越高。

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|}$$



当A和B都为空时，jaccard(A,B)=1；

Jaccard Similarity

Set A = 

Set B = 

Jaccard Similarity J (A,B) = | Intersection (A,B) | / | Union (A,B) |

首先计算出A

= 2 / 7

= 0.286

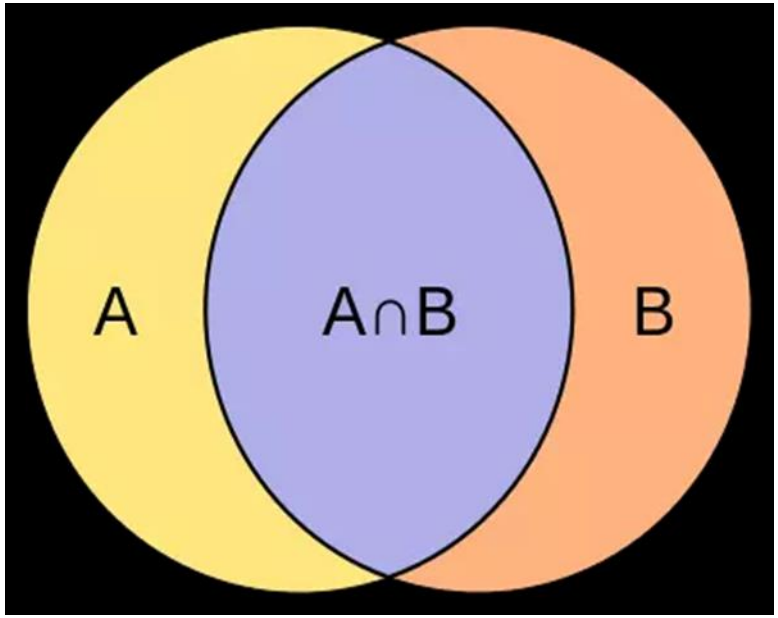
Union 

Intersection (A,B) = 

| Union (A,B) | = 7 | Intersection (A,B)| = 2

- jaccard距离：用于描述不相似度

$$d_j(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} = \frac{A \Delta B}{|A \cup B|}$$



常见的距离函数

- 常用的距离函数有如下几种：
  1. 欧式距离
  2. 曼哈顿距离
  3. 明可夫斯基距离
  4. 余弦距离
  5. Jaccard距离
  6. ……

按照距离公理，在定义距离测度时需要满足距离公理的三个条件非负性、对称性、三角不等性

自定义距离

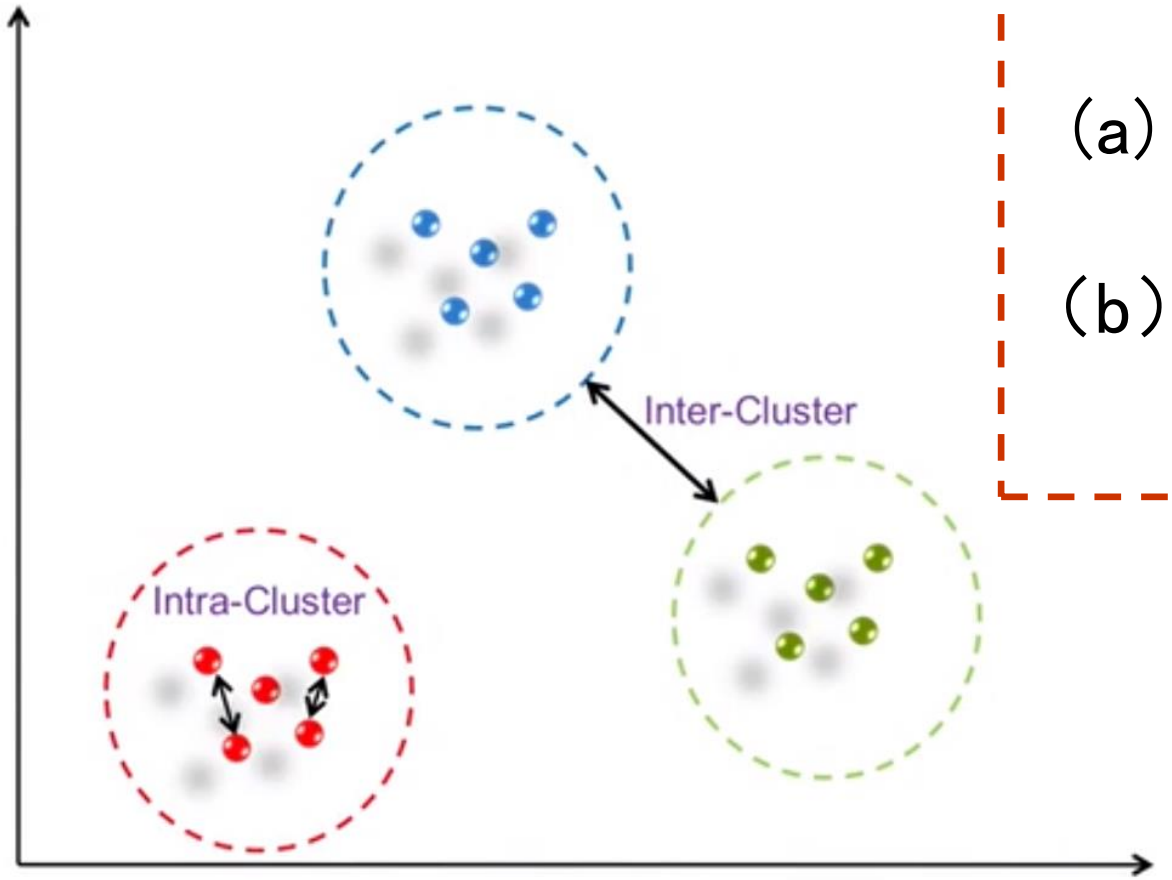
设两个n维向量$\vec{x}=(x_1,x_2,\cdots,x_n)^T$和$\vec{y}=(y_1,y_2,\cdots,y_n)^T$为两个观测，其所定义的距离一般需要满足三个条件：

1. 非负性：$d(\vec{x},\vec{y})\geq 0$，$d(\vec{x},\vec{y})=0$当且仅当$\vec{x}=\vec{y}$

2. 对称性：$d(\vec{x},\vec{y})=d(\vec{y},\vec{x})$

3. 三角不等式：假设存在另一个n维向量$\vec{z}$，$d(\vec{x},\vec{y})\leq d(\vec{x},\vec{z})+d(\vec{z},\vec{y})$

聚类方法概述



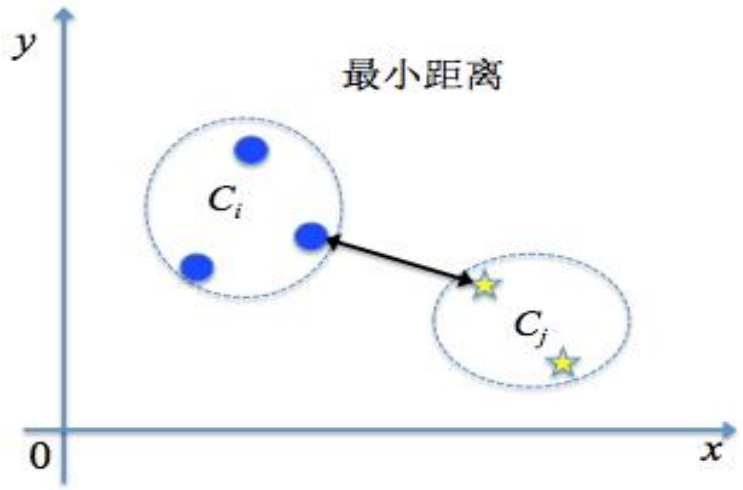
如何评判聚类的好坏：

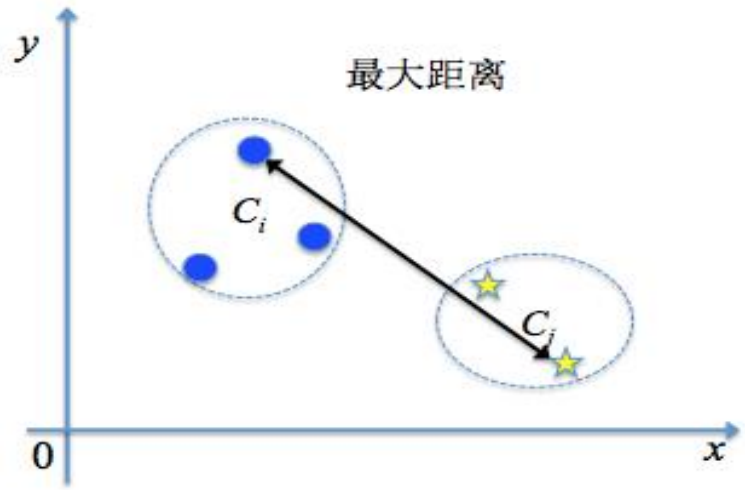
（a） 类间距离：大

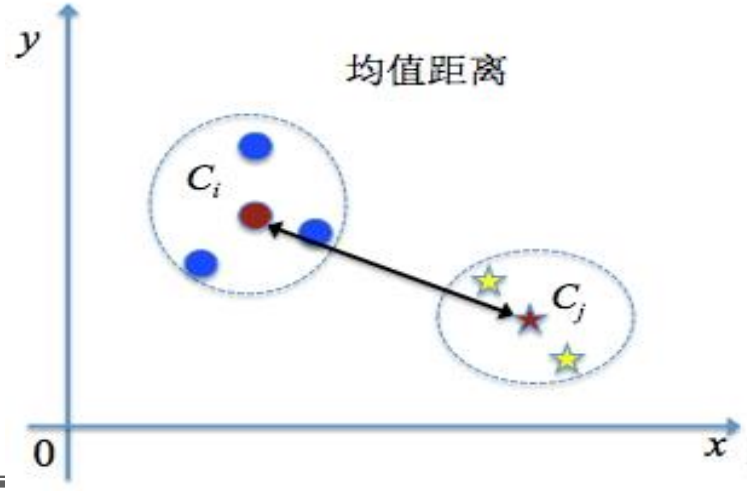
（b）类内距离：小

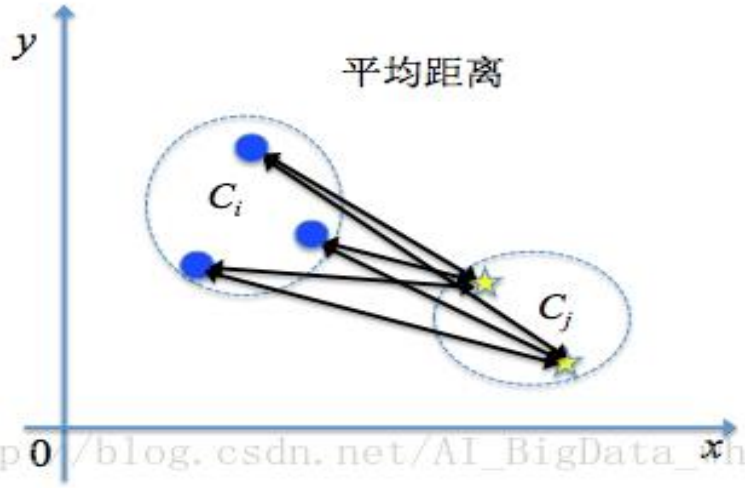
类间距离

- 关键在于如何计算聚类簇之间的距离？









类间距离

- 距离函数都是关于两个样本的距离刻画，然而在聚类应用中，最基本的方法是计算类间的距离。

- 设有两个类C_a和C_b，它们分别有m和h个元素，它们的中心分别为γ_a和γ_b。设元素$x \in C_a$，$y \in C_b$，这两个元素间的距离通常通过类间距离来刻画，记为$D(C_a, C_b)$。

- 类间距离的度量主要有：
  - 最短距离法：
  - 最长距离法：
  - 中心法：
  - 类平均法：
  - 离差平方和

- 类间距离的度量主要有：
  - 最短距离法：定义两个类中最靠近的两个元素间的距离为类间距离。

$$D_{kl} = \min_{i,j}\left[d_{ij}\right]$$

  - 最长距离法：定义两个类中最远的两个元素间的距离为类间距离。

$$D_{kl} = \max_{i,j}\left[d_{ij}\right]$$

  - 中心法：定义两类的两个中心间的距离为类间距离。

$$\overline{x_i} = \frac{1}{n_i}\sum_{x \in C_i} x \qquad D_C(C_a, C_b) = d(r_a, r_b)$$

  - 类平均法：它计算两个类中任意两个元素间的距离，并且综合他们为类间距离：
  - 离差平方和：

$$D_G(C_a, C_b) = \frac{1}{mh}\sum_{x \in C_a}\sum_{y \in C_b} d(x, y)$$

离差平方和

- 离差平方和用到了类直径的概念：
  - 类的直径反映了类中各元素间的差异，可定义为类中各元素至类中心的欧氏距离之和，其量纲为距离的平方：

$$r_a = \sum_{i=1}^{m} (x_i - \overline{x_a})^T (x_i - \overline{x_b})$$

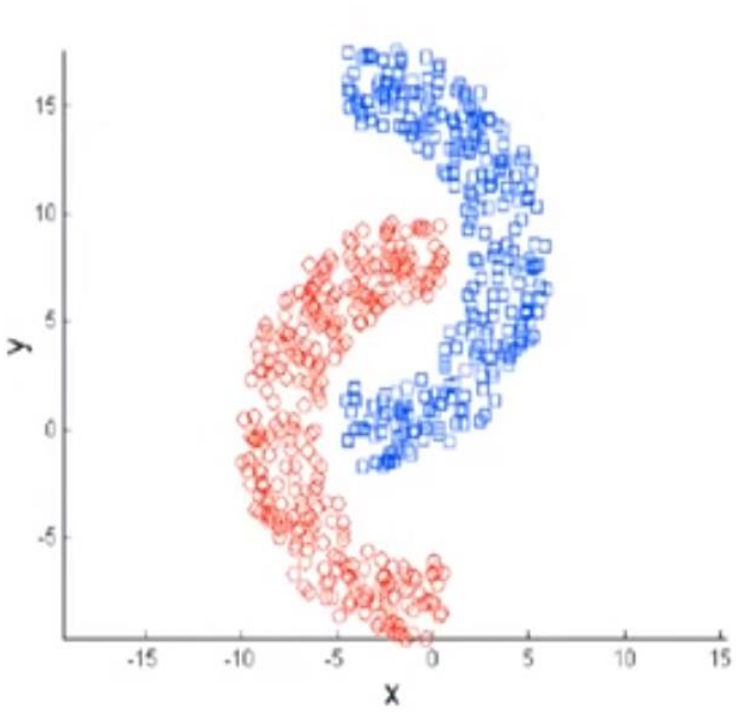
- 根据上式得到两类C_a和C_b的直径分别为γ_a和γ_b，类$C_{a+b} = C_a \cup C_b$的直径为γ_{a+b}，则可定义类间距离的平方为：

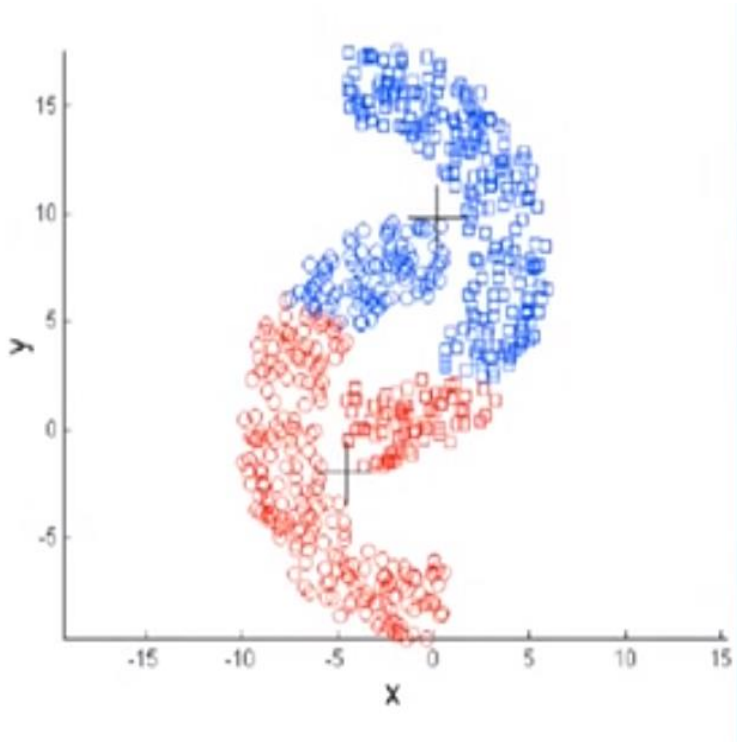
$$D_W^2(C_a, C_b) = r_{a+b} - r_a - r_b$$

聚类方法概述

评判聚类好坏：

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \|x - m_i\|^2, \qquad m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$





聚类的原则

- 基本原则：
  - 高内聚、低耦合
- 衡量聚类效果的标准
  - 簇内相似度越高、簇间相似度越低，聚类效果越好

第五章 聚类分析

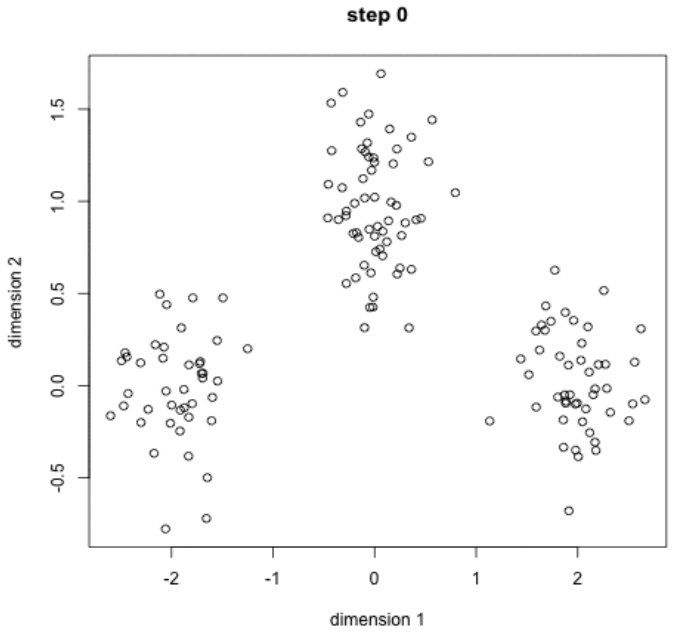
1、聚类的基本定义

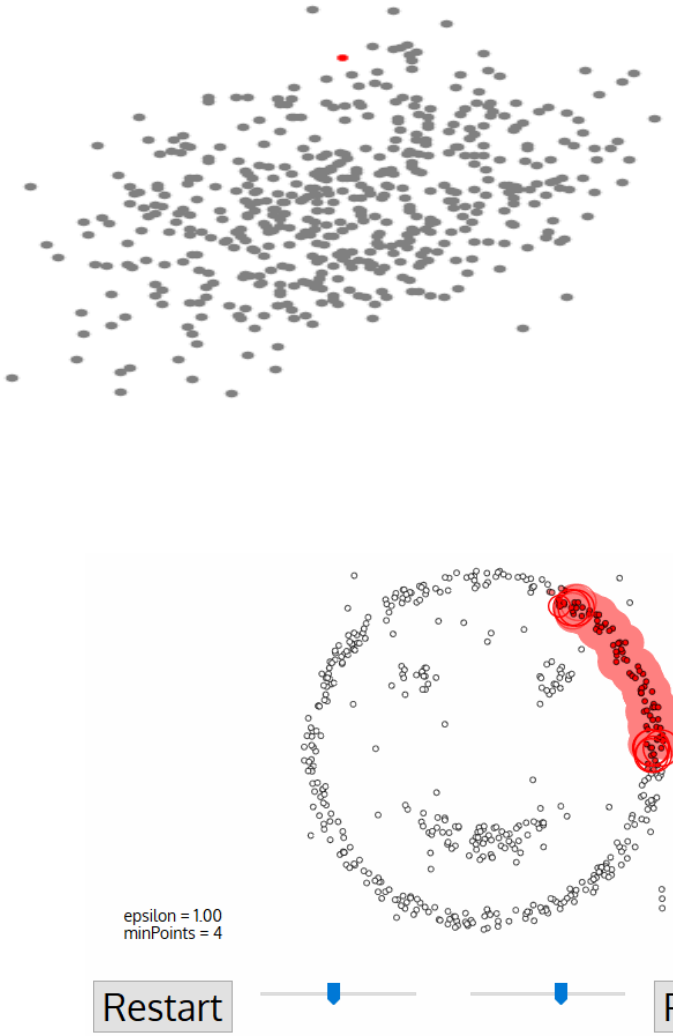
2、如何评判聚类的好坏：距离函数

3、聚类的技术与方法

4、聚类的应用

聚类方法概述


step 0
dimension 2
dimension 1
-0.5
0.0
0.5
1.0
1.5
-2
-1
0
1
2


epsilon = 1.00
minPoints = 4
Restart
Pause

聚类方法分类

- 聚类技术：
  - 划分法：k均值、k中心点
  - 层次法：凝聚层次聚类、分裂层次聚类
  - 基于密度的方法：Density-based approach
  - 基于模型的方法：Model-based approach

聚类方法-kmeans算法

k-means（K均值算法）

(a) 给定集合D，有n个样本点

(b) 随机指定k个点，作为k个子集的质心

(c) 根据样本点与k个质心的距离远近，将每个样本点划归最近质心所在的子集

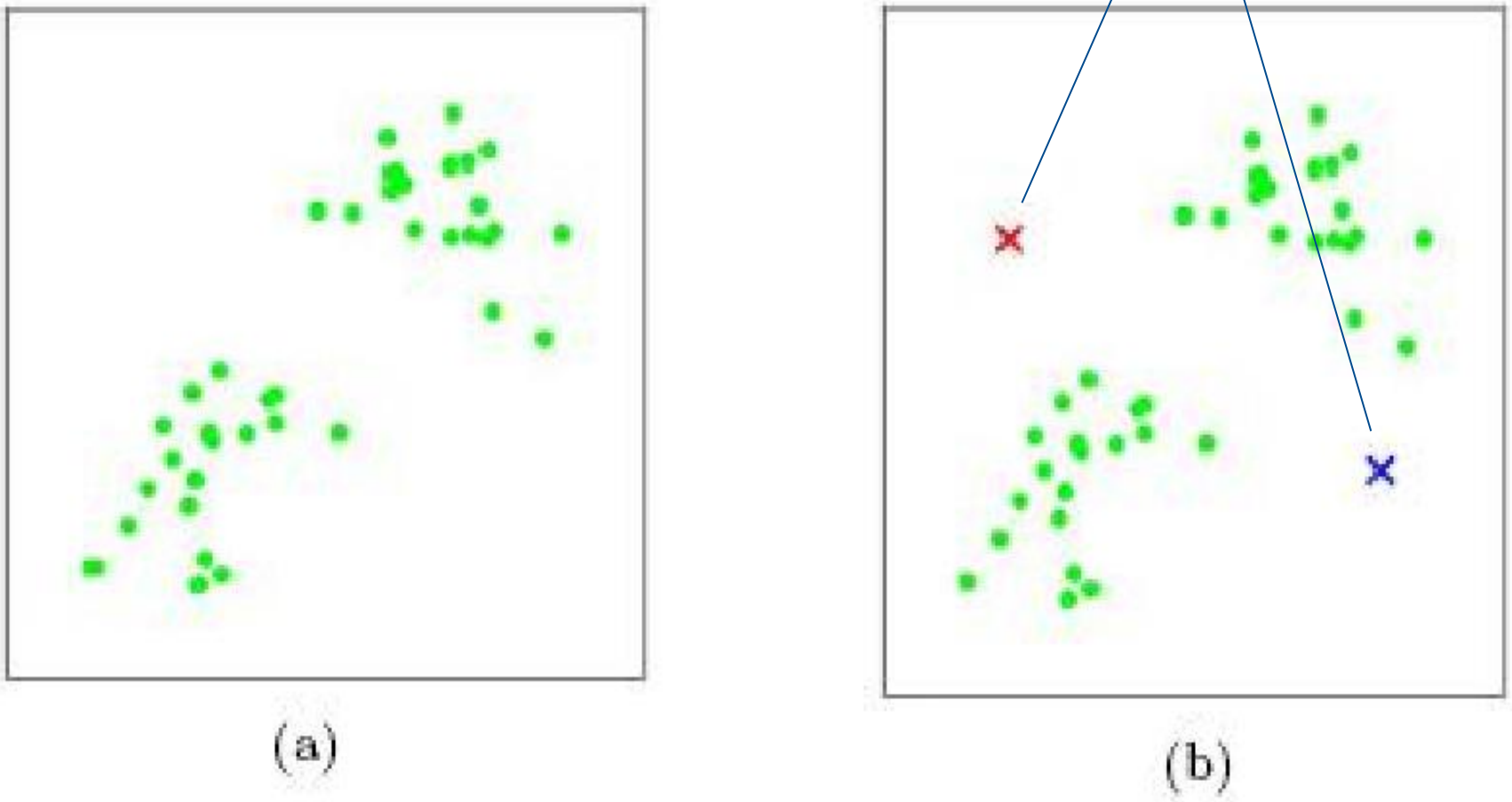
(d) 对k个子集重新计算质心

(e) 根据新的质心，重复操作(c)

(f) 重复操作(d)和(e)，直至结果足够收敛或者不再变化

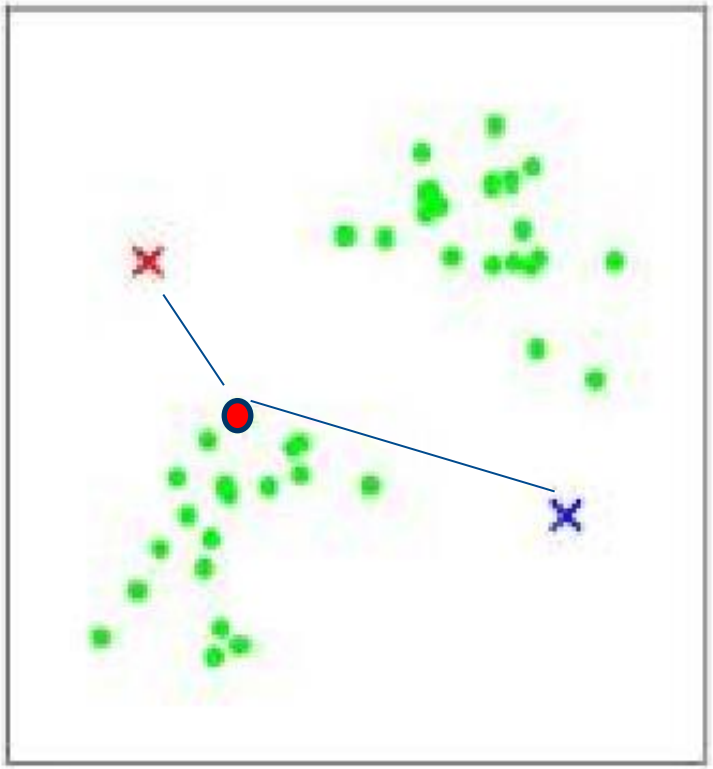
聚类方法-kmeans算法

a、随机指定两个点，作为两个子集的质心

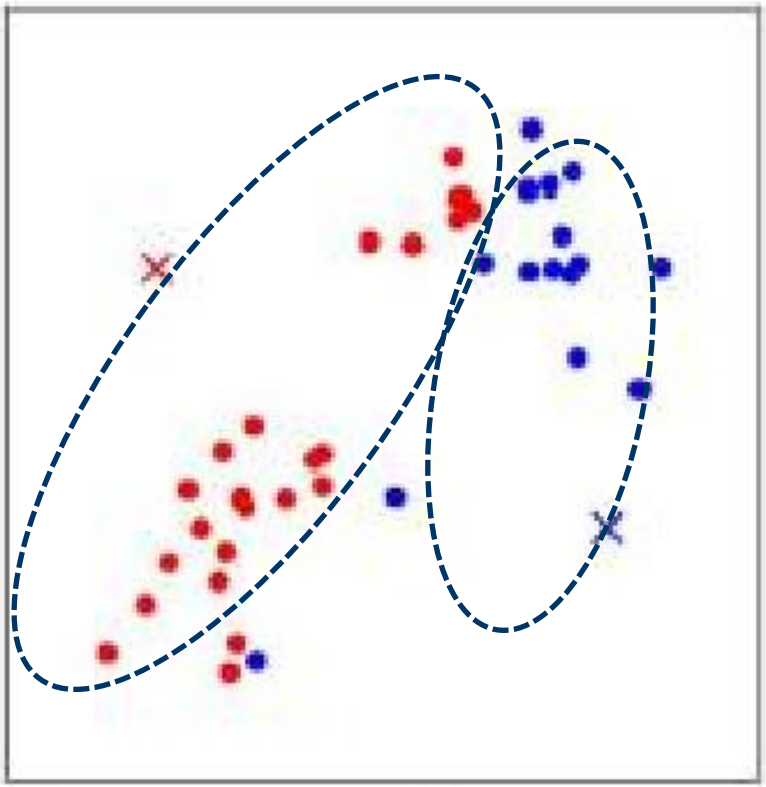


聚类方法-kmeans算法

b、第一次迭代



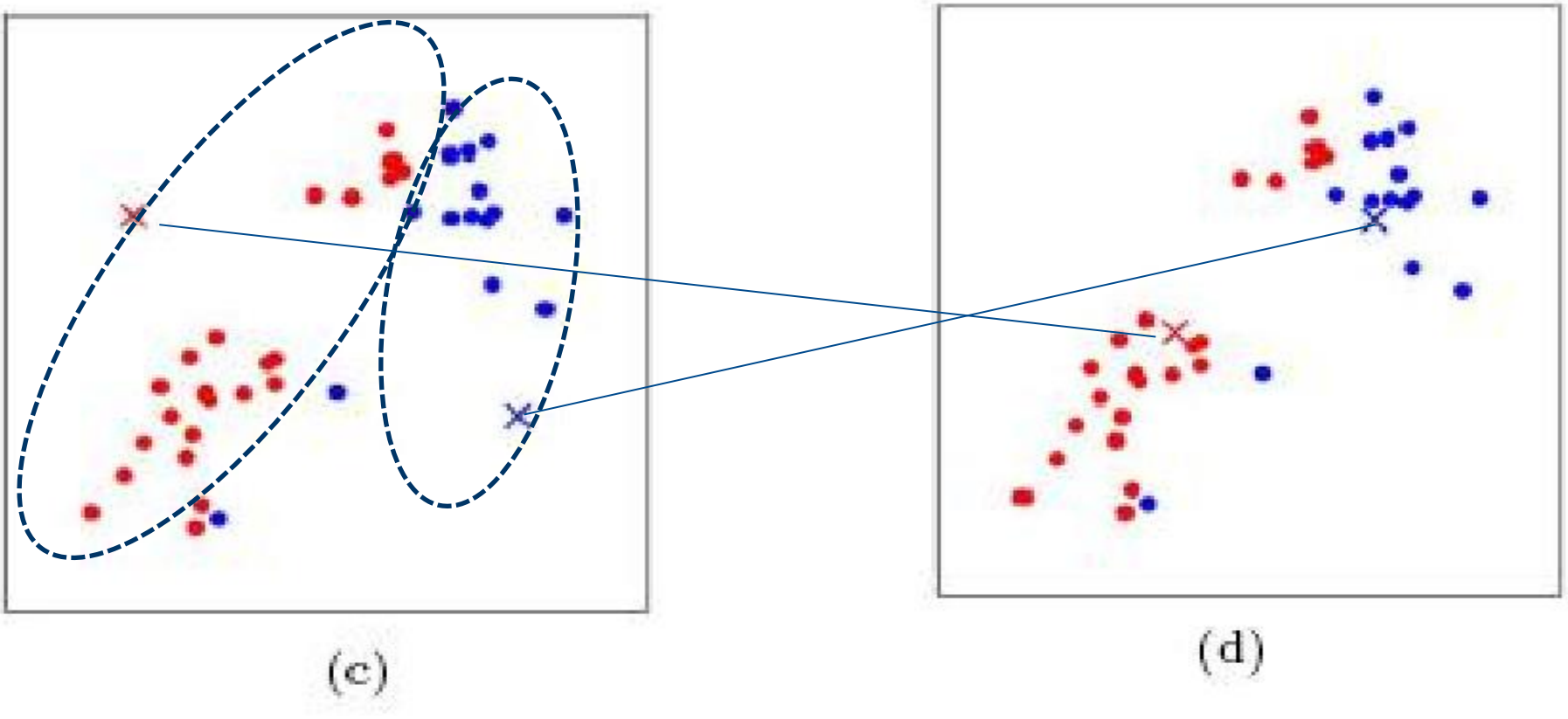
(b)



(c)

聚类方法-kmeans算法

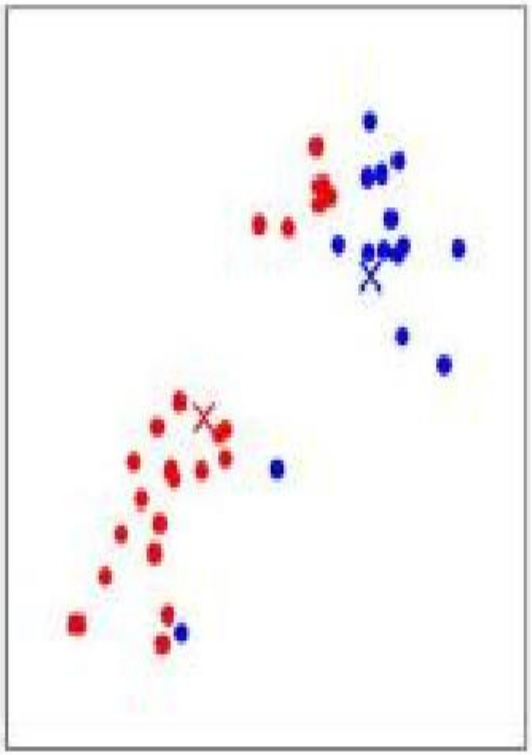
c、重新计算质心：$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$



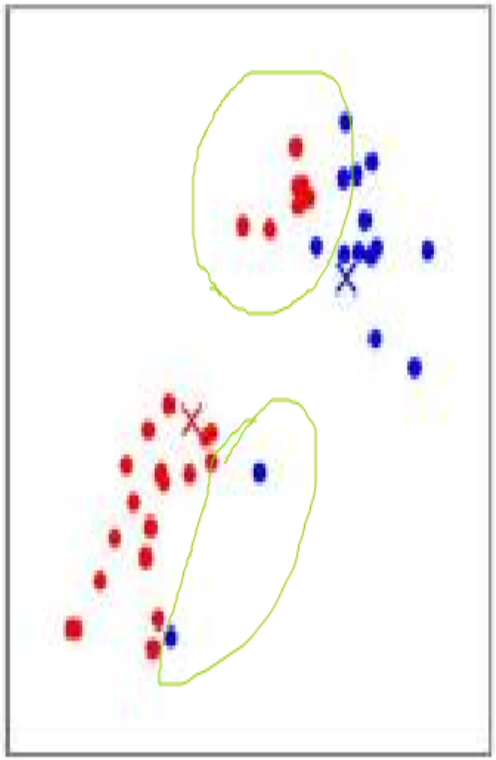
(c)　　(d)

聚类方法-kmeans算法

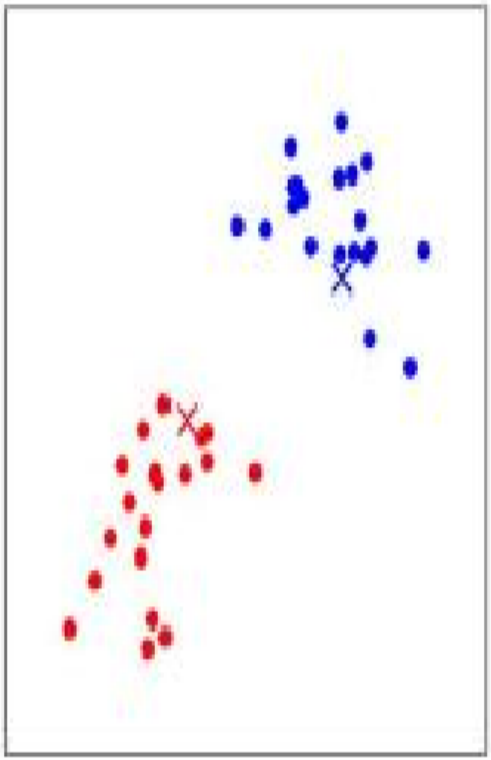
d、第二次迭代，重新计算质心



(d)



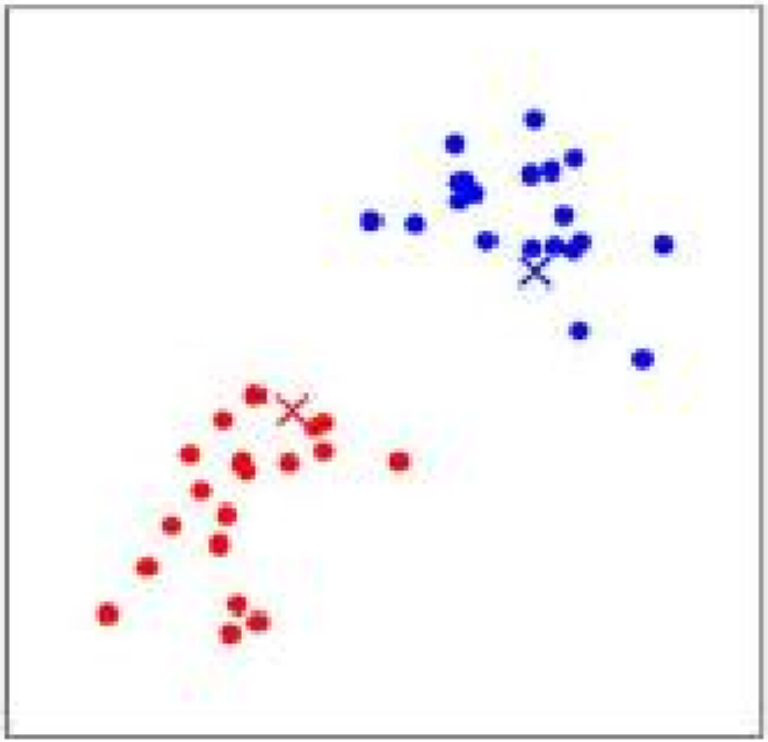
(d)



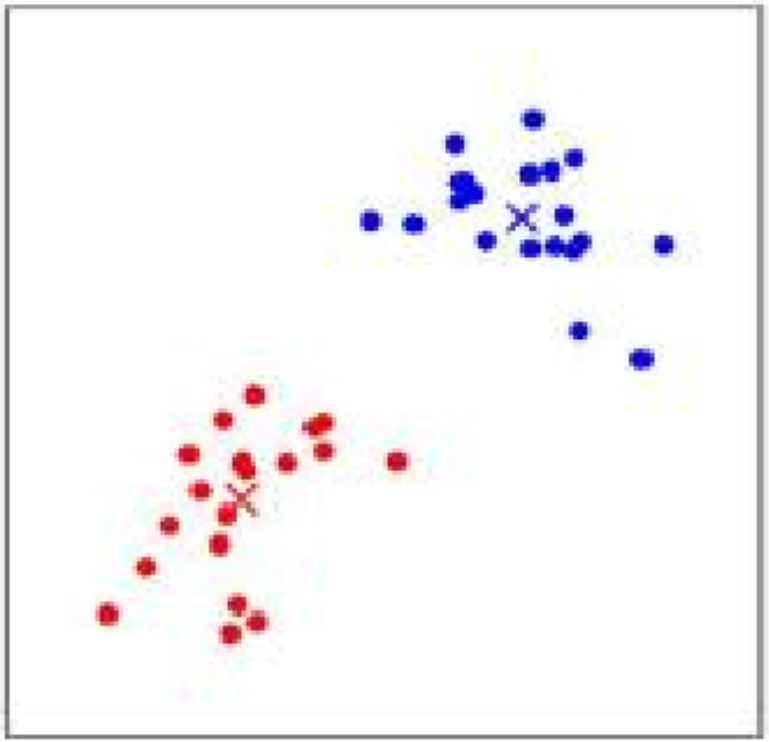
(e)

聚类方法-kmeans算法

e、重复迭代，计算质心，直到质心足够收敛或不再变化



(e)



(f)

样本数据

序号	属性 1	属性 2
①	1	1
②	2	1
③	1	2
④	2	2
⑤	4	3
⑥	5	3
⑦	4	4
⑧	5	4

k-means (设*n*=8，*k*=2)：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇

k-means例子

样本数据

序号	属性 1	属性 2
1	1	1
2	2	1
3	1	2
4	2	2
5	4	3
6	5	3
7	4	4
8	5	4

根据所给的数据通过对其实施*k*-means (设n=8，k=2),，其主要执行执行步骤：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇{1，2}和{3，4，5，6，7，8}。

对于产生的簇分别计算平均值，得到平均值点。

对于{1，2}，平均值点为（1.5，1）（这里的平均值是简单的相加出2）；对于{3，4，5，6，7，8}，平均值点为（3.5，3）。

k-means例子

样本数据		
序号	属性 1	属性 2
①	1	1
②	2	1
③	1	2
④	2	2
⑤	4	3
⑥	5	3
⑦	4	4
⑧	5	4

根据所给的数据通过对其实施*k*-means (设*n*=8，*k*=2),，其主要执行执行步骤：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇{①，②和{③，④，⑤，⑥，⑦，⑧}。对于产生的簇分别计算平均值，得到平均值点。

对于{①，②}，平均值点为（1.5，1）（这里的平均值是简单的相加出2）；
对于{③，④，⑤，⑥，⑦，⑧}，平均值点为（3.5，3）。

第二次迭代：通过平均值调整对象的所在的簇，重新聚类，即将所有点按离平均值点（1.5，1）、（3.5，1）最近的原则重新分配。得到两个新的簇：{①，②，③，④}和{⑤，⑥，⑦，⑧}。重新计算簇平均值点，得到新的平均值点为（1.5，1.5）和（4.5，3.5）。

第三次迭代：将所有点按离平均值点（1.5，1.5）和（4.5，3.5）最近的原则重新分配，调整对象，簇仍然为{1，2，3，4}和{5，6，7，8}，发现没有出现重新分配，而且准则函数收敛，程序结束。

迭代次数	平均值（簇1）	平均值（簇2）	产生的新簇	新平均值（簇1）	新平均值（簇2）
1	（1，1）	（1，2）	{1，2}，{3，4，5，6，7，8}	（1.5，1）	（3.5，3）
2	（1.5，1）	（3.5，3）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）
3	（1.5，1.5）	（4.5，3.5）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）

K-means算法主要思想

算法5-1 *k*-means算法

输入：簇的数目*k*和包含*n*个对象的数据库。

输出：*k*个簇，使平方误差准则最小。

(1)assign initial value for means; /*任意选择*k*个对象作为初始的簇中心；*/

(2) REPEAT

(3) FOR *j*=1 to *n* DO assign each X_j to the closest clusters;

(4) FOR *i*=1 to *k* DO / *更新簇平均值*/

(5) Compute /*计算准则函数*E*/

(6) UNTIL *E*不再明显地发生变化。

k-means算法

- K值：要得到的簇的个数
- 质心：每个簇的均值向量，即向量各维取平均
- 距离量度：

欧式距离： $d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

曼哈顿距离： $d_{12} = |x_1 - x_2| + |y_1 - y_2|$

切比雪夫距离： $d_{12} = \max(|x_1 - x_2|, |y_1 - y_2|)$

余弦距离： $\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}$

Jaccard相似系数： $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$

相关系数： $\rho_{XY} = \frac{Cov(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}} = \frac{E((X - EX)(Y - EY))}{\sqrt{D(X)}\sqrt{D(Y)}}$

k-means算法的性能分析

- **主要优点：**
  - 是解决聚类问题的一种经典算法，简单、快速。
  - 对处理大数据集，该算法是相对可伸缩和高效率的。
  - 当结果簇是密集的，它的效果较好。
- **主要缺点**
  - 在簇的平均值被定义的情况下才能使用，可能不适用于某些应用。
  - 必须事先给出k（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸面形状的簇或者大小差别很大的簇。而且，它对于躁声和孤立点数据是敏感的。

第五章 聚类分析

1、聚类的基本定义

2、如何评判聚类的好坏：距离函数

3、聚类的技术与方法：k-means、PAM

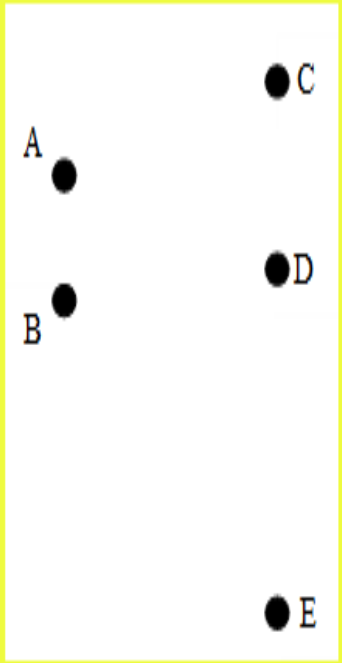
4、聚类的应用

PAM 方法

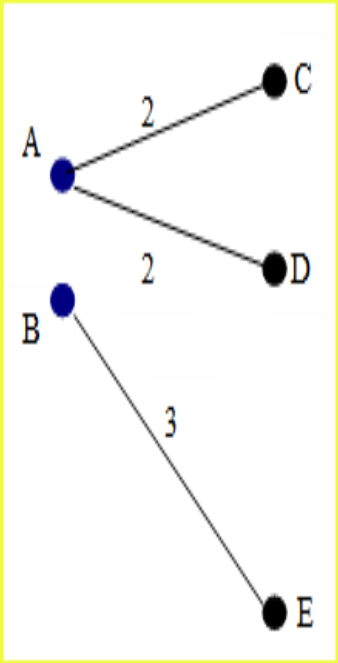
- PAM：Partitioning Around Medoid，围绕中心点的划分，选用簇中位置最中心的对象作为代表对象，试图对n个对象给出k个划分。
  - 代表对象也被称为是中心点，其他对象则被称为非代表对象。
  - 最初随机选择k个对象作为中心点，该算法反复地用非代表对象来代替代表对象，试图找出更好的中心点，以改进聚类的质量。
  - 在每次迭代中，所有可能的对象对被分析，每个对中的一个对象是中心点，而另一个是非代表对象。
  - 对可能的各种组合，估算聚类结果的质量。一个对象Oi被可以产生最大平方-误差值减少的对象代替。在一次迭代中产生的最佳对象集合成为下次迭代的中心点。

假如空间中的五个点{A、B、C、D、E}如图1所示，各点之间的距离关系如表1所示，根据所给的数据对其运行PAM算法实现划分聚类（设k=2）。样本点间距离如下表所示:

样本点	A	B	C	D	E
A	0	1	2	2	4
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



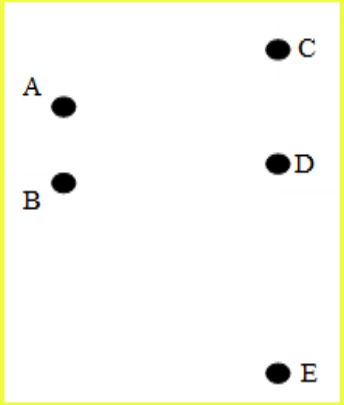
样本点



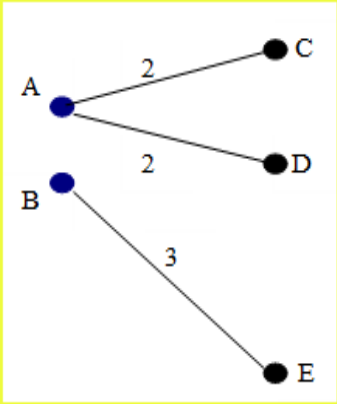
起始中心点为A, B

第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A，B}，则样本被划分为{A、C、D}和{B、E}，如图5-3所示。

样本点	A	B	C	D	E
A	0	1	2	2	4
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



样本点



起始中心点为A,B

第二步　交换阶段：假定中心点A、B分别被非中心点{C、D、E}替换，根据PAM算法需要计算下列代价TC_{AC}、 TC_{AD}、 TC_{AE}、TC_{BC}、TC_{BD}、 TC_{BE}。以TC_{AC}为例说明计算过程：

a）当A被C替换以后，A不再是一个中心点，因为A离B比A离C近，A被分配到B中心点代表的簇，$C_{AAC}=d(A,B)-d(A,A)=1$。

b）B是一个中心点，当A被C替换以后，B不受影响，$C_{BAC}=0$。

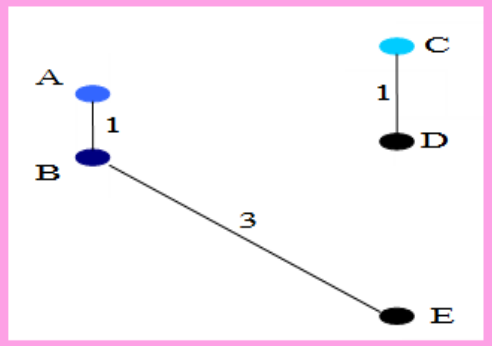
c）C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，符合PAM算法代价函数的第二种情况$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$。

d）D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，根据PAM算法代价函数的第二种情况$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$。

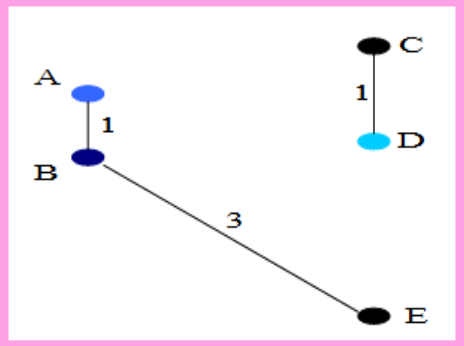
e）E原先属于B中心点所在的簇，当A被C替换以后，离E最近的中心仍然是 B，根据PAM算法代价函数的第三种情况$C_{EAC}=0$。

因此，$TC_{AC}=CA_{AC}+CB_{AC}+CB_{AC}+CD_{AC}+CE_{AC}=1+0-2-1+0=-2$。

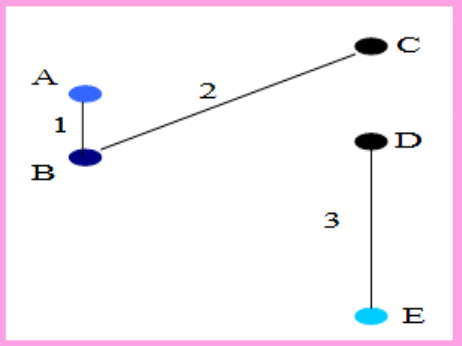
在上述代价计算完毕后，我们要选取一个最小的代价，显然有多种替换可以选择，我们选择第一个最小代价的替换（也就是C替换A），根据图5-4（a）所示，样本点被划分为{ B、A、E}和{C、D}两个簇。图5-4（b）和图5-4（c）分别表示了D替换A，E替换A的情况和相应的代价



(a) C替换A, $TC_{AC} = -2$



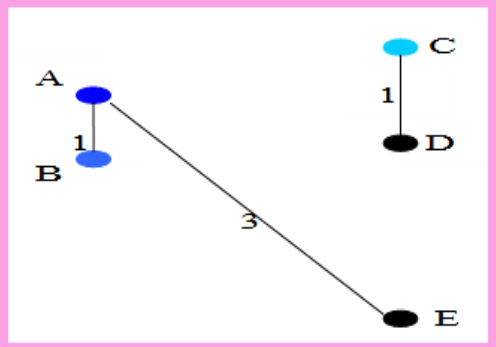
(b) D替换A, $TC_{AD} = -2$



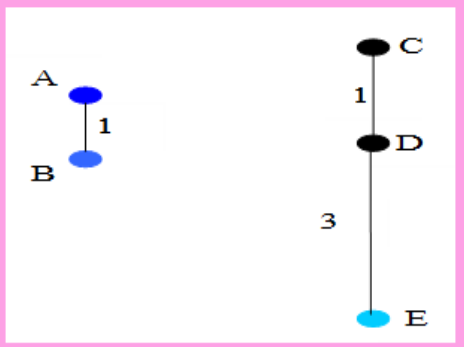
(c) E替换A, $TC_{AE} = -1$

图5-4 替换中心点A

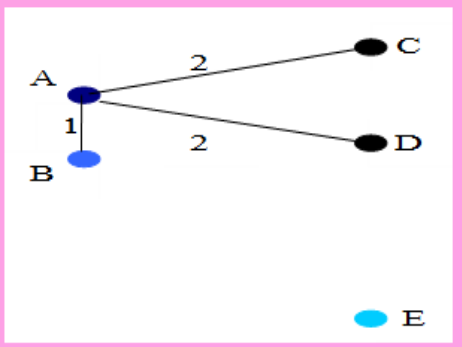
图5-5（a）、（b）、（c）分别表示了用C、D、E替换B的情况和相应的代价。



(a) C替换B, $TC_{BC} = -2$



(b) D替换B, $TC_{BD} = -2$



(c) E替换B, $TC_{BE} = -2$

图5-5 替换中心点B

通过上述计算，已经完成了PAM算法的第一次迭代。在下一迭代中，将用其他的非中心点{A、D、E}替换中心点{B、C}，找出具有最小代价的替换。一直重复上述过程，直到代价不再减小为止。

PAM算法基本思想(续)

算法5-2 PAM（围绕中心点的划分）

输入：簇的数目k和包含n个对象的数据库。

输出：k个簇，使得所有对象与其最近中心点的相异度总和最小。

（1） 任意选择k个对象作为初始的簇中心点；

（2） REPEAT

（3） 指派每个剩余的对象给离它最近的中心点所代表的簇；

（4） REPEAT

（5） 选择一个未被选择的中心点O_i；

（6） REPEAT

（7） 选择一个未被选择过的非中心点对象O_h；

（8） 计算用O_h代替O_i的总代价并记录在S中；

（9） UNTIL 所有的非中心点都被选择过；

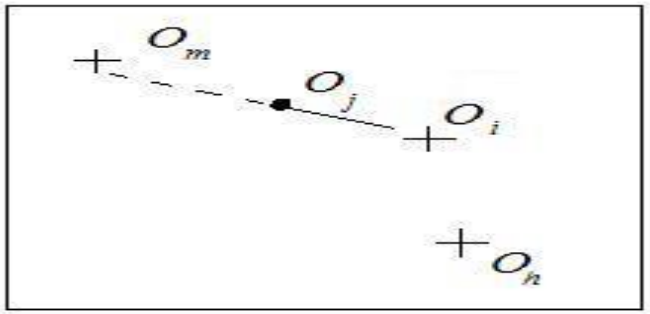
（10） UNTIL 所有的中心点都被选择过；

（11） IF 在S中的所有非中心点代替所有中心点后的计算出的总代价有小于0的存在 THEN 找出S中的用非中心点替代中心点后代价最小的一个，并用该非中心点替代对应的中心点，形成一个新的k个中心点的集合；

（12）UNTIL 没有再发生簇的重新分配，即所有的S都大于0.

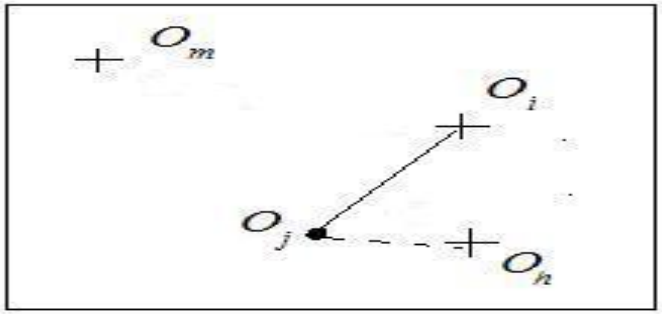
PAM算法代价函数的四种情况

第一种情况



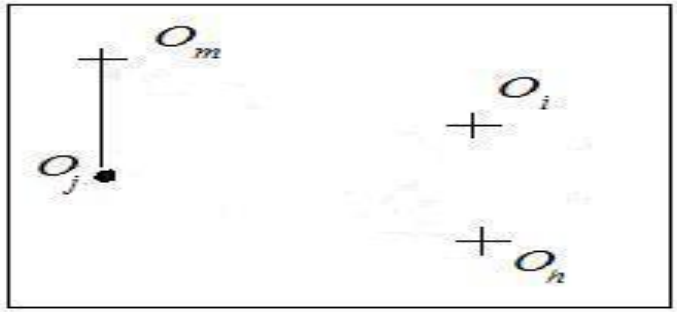
O_j被重新分配给O_m,
$C_{jih} = d(j, m) - d(j, i)$

第二种情况



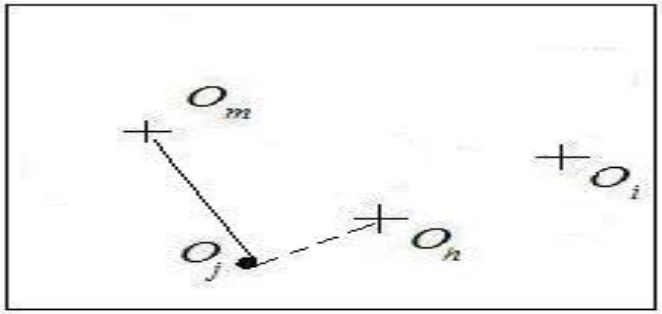
O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, i)$

第三种情况



O_j的隶属不发生变化,
$C_{jih} = 0$

第四种情况



O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, m)$

PAM算法基本思想(续)

- 为了判定一个非代表对象O_h是否是当前一个代表对象O_i的好的替代，对于每一个非中心点对象O_j，下面的四种情况被考虑：
  - **第一种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h所代替作为中心点，且O_j离一个O_m最近，$i \neq m$，那么O_j被重新分配给O_m。
  - **第二种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_j被重新分配给O_h。
  - **第三种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，而O_j依然离O_m最近，那么对象的隶属不发生变化。
  - **第四种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_i被重新分配给O_h。

PAM算法基本思想(续)

- 每当重新分配发生时，平方-误差E所产生的差别对代价函数有影响。因此，如果一个当前的中心点对象被非中心点对象所代替，代价函数计算平方-误差值所产生的差别。替换的总代价是所有非中心点对象所产生的代价之和。
  - 总代价定义如下：

$$TC_{ih} = \sum_{j=1}^{n} C_{jih}$$

其中，C_{jih}表示O_j在O_i被O_h代替后产生的代价。下面我们将介绍上面所述的四种情况中代价函数的计算公式，其中所引用的符号有：O_i和O_m是两个原中心点，O_h将替换O_i作为新的中心点。

  - 如果总代价是负的，那么实际的平方-误差将会减小，O_i可以被O_h替代。
  - 如果总代价是正的，则当前的中心点O_i被认为是可接受的，在本次迭代中没有变化。

PAM算法性能分析

- (1) 消除k-平均算法对于孤立点的敏感性。
- (2) K-中心点方法比k-平均算法的代价要高
- (3) 必须指定k
- (4) PAM对小的数据集非常有效，对大数据集效率不高。特别是n和k都很大的时候。

k-means的几种改进方法

- k--medoids算法：K中心点算法中，每次迭代后的质点都是从聚类的样本点中选取，k中心点算法不采用簇中对象的平均值作为簇中心，而选用簇中离平均值最近的对象作为簇中心。

- k-mode 算法：实现对离散数据的快速聚类，保留了k-means算法的效率同时将k-means的应用范围扩大到离散数据。

- k-prototype算法：可以对离散与数值属性两种混合的数据进行聚类，在k-prototype中定义了一个对数值与离散属性都计算的相异性度量标准。

K-medoids 方法

- k-中心点（K-medoids）：算法k-means算法对于孤立点是敏感的。为了解决这个问题，不采用簇中的平均值作为参照点，可以选用簇中位置最中心的对象，即中心点作为参照点。这样划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的。

- k中心点聚类方法：对下列表中的10个数据聚类，每个数据的维度都为2， k=2。 c1=（3，4），c2=（7，4）.那么将所有点到这两点的距离计算出来

X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6

K-medoids 方法

- 随机挑选k=2个中心点：c1=（3，4），c2=（7，4）.那么将所有点到这两点的距离计算出来

Data object		Distance to	
i	X_i	$c_1 = (3,4)$	$c_2 = (7,4)$
1	(2, 6)	3	7
2	(3, 4)	0	4
3	(3, 8)	4	8
4	(4, 7)	4	6
5	(6, 2)	5	3
6	(6, 4)	3	1
7	(7, 3)	5	1
8	(7, 4)	4	0
9	(8, 5)	6	2
10	(7, 6)	6	2
Cost		11	9

- 随机挑选k=2个中心点：c1=（3，4），c2=（7，4）.那么将所有点到这两点的距离计算出来

Data object		Distance to	
i	X_i	$c_1 = (3, 4)$	$c_2 = (7, 4)$
1	(2, 6)	3	7
2	(3, 4)	0	4
3	(3, 8)	4	8
4	(4, 7)	4	6
5	(6, 2)	5	3
6	(6, 4)	3	1
7	(7, 3)	5	1
8	(7, 4)	4	0
9	(8, 5)	6	2
10	(7, 6)	6	2
Cost		11	9

$$\underbrace{3+0+4+4}_{\text{objects in cluster 1}} + \underbrace{3+1+1+0+2+2}_{\text{objects in cluster 2}} = 20$$

- 挑选一个非中心点O’，比如X7，O‘=（7，3）。那么此时这两个中心点暂时变成了c1(3,4) and O’(7,3)；total cost = 3+4+4+2+2+1+3+3 = 22

i	O′		Data objects (X_i)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	**2**
6	7	3	6	4	**2**
8	7	3	7	4	**1**
9	7	3	8	5	**3**
10	7	3	7	6	**3**

K-medoids 方法

- K-modes算法可以看做是k-means算法在非数值属性集合上的版本。
- 具体算法步骤如下：
- 1.随机确定k个聚类中心
- 2.对于样本xj，分别比较其与k个中心之间的距离（这里的距离为不同属性值的个数）
- 3.将xj划分到距离最小的簇，在全部的样本都被划分完毕之后，重新确定簇中心，向量Ci中的每一个分量都更新为簇i中的众数
- 重复步骤2和3，直到总距离（各个簇中样本与各自簇中心距离之和）不再降低，返回最后的聚类结果

聚类的应用

聚类在数据挖掘中的典型应用有：

1、聚类分析可以作为其它算法的预处理步骤

2、聚类分析可以作为一个独立的工具来获得数据的分布情况

3、聚类分析可以完成孤立点挖掘