

# 机器学习

## 第2讲 无监督学习

湖南大学 陈湘涛

# 第 2 讲

## 无监督学习

1

概述

2

相似性计算方法

3

划分方法

4

层次方法

## 2.1 概述

### ■ 无监督学习的定义

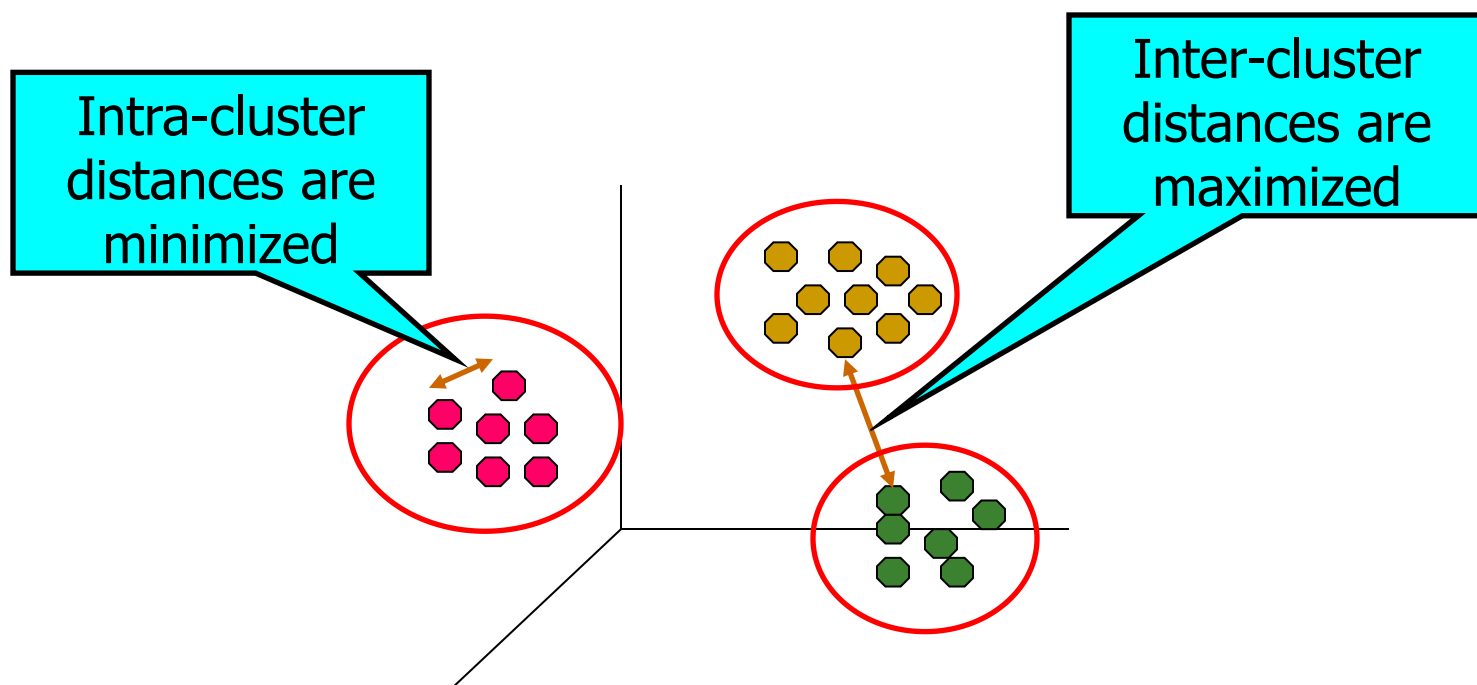
- 无监督学习，又称为聚类分析(Cluster Analysis)，是一个将数据集中的所有数据，按照相似性划分为多个类别（Cluster, 簇）的过程；

- 簇是相似数据的集合。

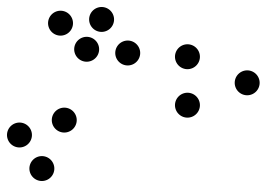
- 是一种无监督(Unsupervised Learning)分类方法：数据集中的数据没有预定义的类别标号（无训练集和训练的过程）。
- 要求：尽可能保证类别相同的数据之间具有较高的相似性，而类别不同的数据之间具有较低的相似性。

# 无监督学习

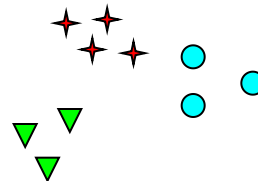
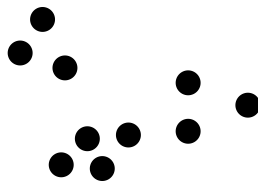
- 发现对象簇(Cluster), 使得同一个簇内的对象尽量相似, 不同簇间的对象尽量不同。



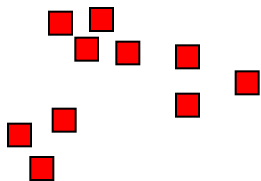
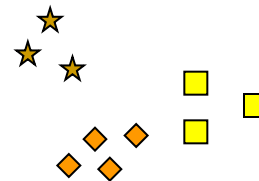
# 簇的概念可能会模糊



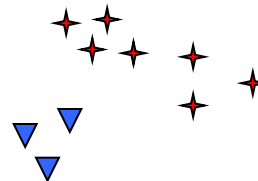
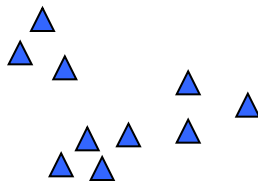
How many clusters?



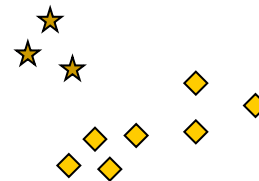
Six Clusters



Two Clusters



Four Clusters



## 2.1 概述

- **无监督学习**在机器学习中的作用：
  - 作为一个独立的工具来获得数据集中数据的分布情况；
    - 首先，对数据集执行聚类，获得所有簇；
    - 然后，根据每个簇中样本的数目获得数据集中每类数据的大体分布情况。
  - 作为其他机器学习算法的预处理步骤。
    - 首先，对数据进行聚类——**粗分类**；
    - 然后，分别对每个簇进行特征提取和**细分类**，可以有效提高分类精度。

## 2.1 概述

### ■ 无监督学习的典型应用：

#### □ 空间数据分析

- 图像处理——灰度图像的二值化（对灰度像素进行聚类）。

#### □ 万维网

- 对WEB日志数据进行聚类，以发现类似的用户访问模式。

#### □ 金融领域

- 用户交易数据的聚类分析，以获得奇异点（异常交易）。

#### □ .....

## 2.1 概述

- 无监督学习（聚类）质量的评价
- ◆ 高质量的聚类：
  - 高簇内相似性(high intra-class similarity)
  - 低簇间相似性(low inter-class similarity)
- ◆ 聚类的质量不但依赖于所使用的方法，而且也依赖于实现方式。
- ◆ 聚类质量最主要的评价标准还是用户的满意程度。



## 2.1 概述

### ■ 无监督学习（聚类）质量的度量

#### ◆ 相似性度量:

- 一般通过距离函数来描述:  $d(i, j)$
- 针对不同数据，如区间值数据、布尔数据、类别数据、顺序数据等，会有不同的距离函数
- 根据不同应用和数据的语义，变量会被赋予不同的权重。

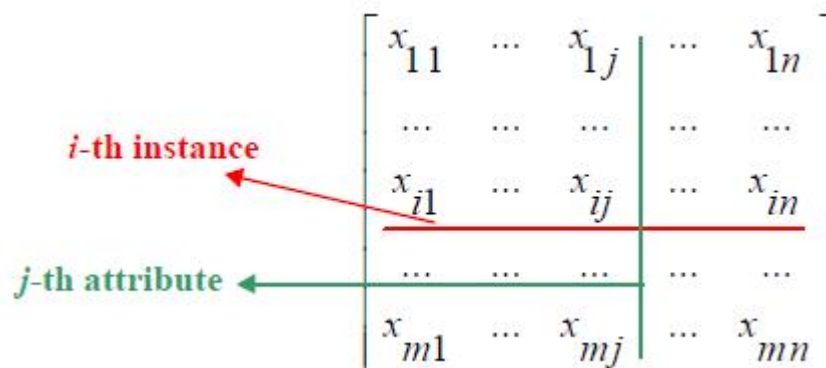
#### ◆ 聚类的质量:

- 通常会有明确的质量函数来度量聚类质量的好坏。
- 很难定义“足够好”
  - 这类问题的答案往往具有明显的主观色彩。

## 2.1 概述

### ■ 聚类中常用的数据结构

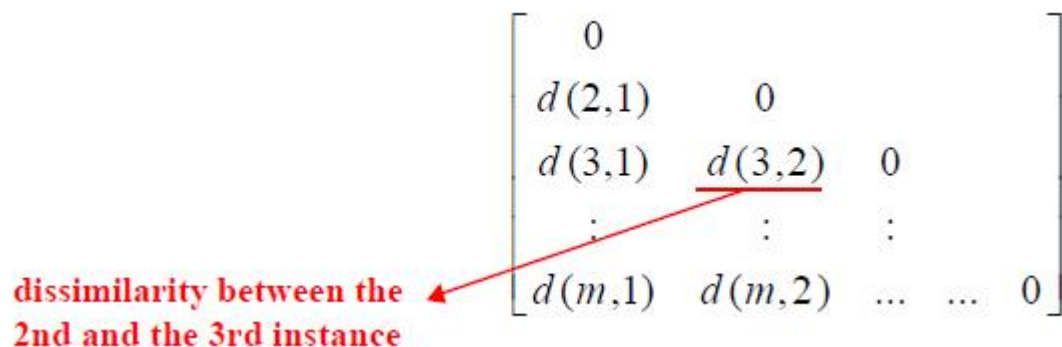
#### ■ 数据矩阵



The diagram shows a data matrix with  $m$  rows and  $n$  columns. A red arrow points from the text " $i$ -th instance" to the  $i$ -th row. A green arrow points from the text " $j$ -th attribute" to the  $j$ -th column. The matrix is represented as follows:

$$\begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{bmatrix}$$

#### ■ 相异矩阵



The diagram shows a dissimilarity matrix, which is a square matrix where the diagonal elements are 0. A red arrow points from the text "dissimilarity between the 2nd and the 3rd instance" to the element  $d(3,2)$  in the matrix. The matrix is represented as follows:

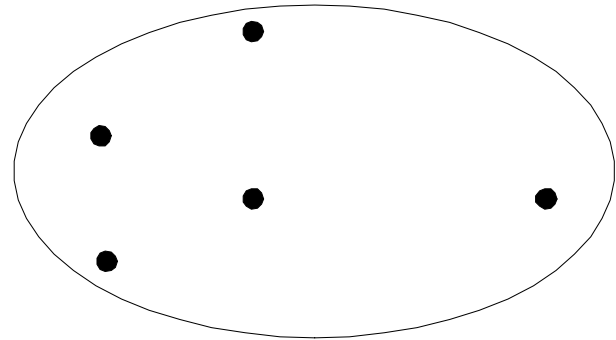
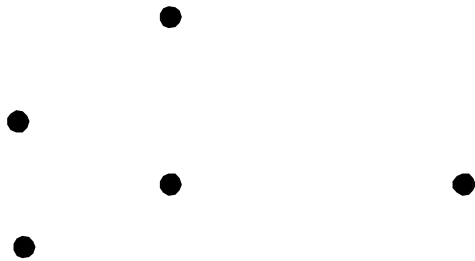
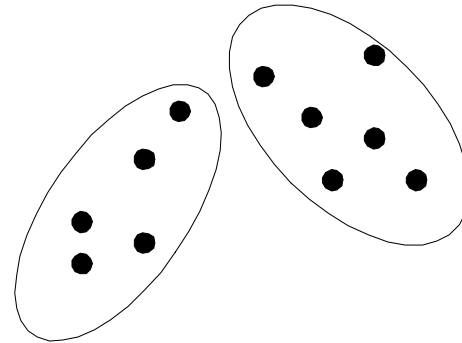
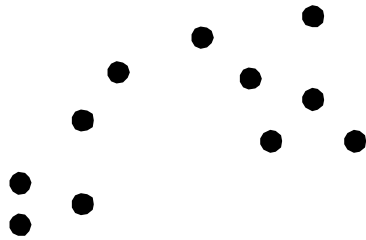
$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & \underline{d(3,2)} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(m,1) & d(m,2) & \dots & \dots & 0 \end{bmatrix}$$

## 2.1 概述

### ■ 常用的无监督学习方法：

- **划分法**（Partitioning Methods）：以距离作为数据集中不同数据间的相似性度量，将数据集划分成多个簇。(分割聚类)
  - 属于这样的聚类方法有：**k-means**、**k-medoids**等。
- **层次法**（Hierarchical Methods）：对给定的数据集进行层次分解，形成一个树形的聚类结果。(层次聚类)
  - 属于这样的聚类方法有：**自顶向下法**、**自底向上法**。

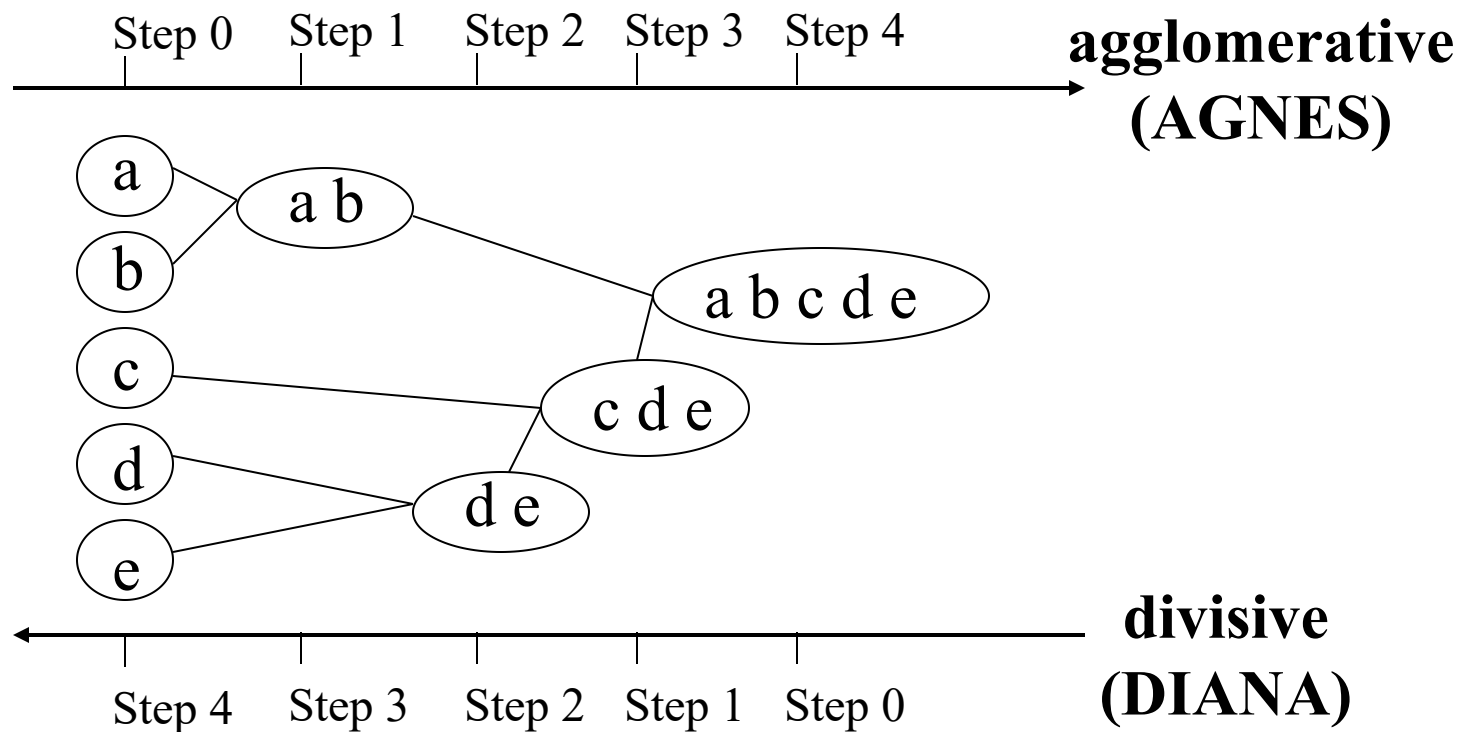
# 分割聚类



**Original Points**

**A Partitional Clustering**

# 层次聚类



# 聚类的类型（1）

## ◆ 排它的与非排它的

- 非排它聚类中，对象可能属于多个簇

## ◆ 模糊与非模糊的

- 模糊聚类中，每个点与每个簇都有一个0到1之间的隶属度
- 这些隶属度的和必须是1

## ◆ 部分与完全的

- 在某些情况下，可能只对一部分数据进行聚类

## ◆ 同构与异构的

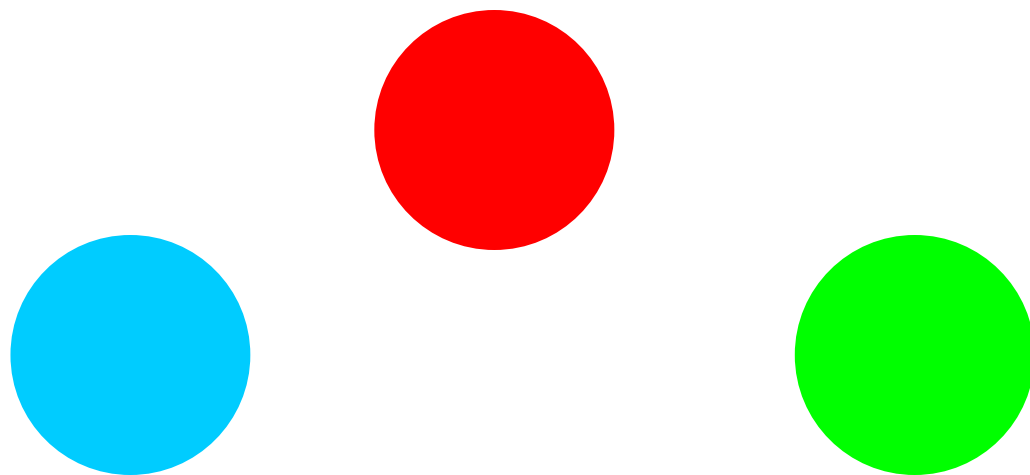
- 可能对不同类型的对象进行聚类

## 聚类的类型 (2)

- ◆ 良分割聚类(Well-separated clusters)
- ◆ 基于中心的聚类(Center-based clusters)
- ◆ 邻近聚类(Contiguous clusters)
- ◆ 基于密度的聚类(Density-based clusters)
- ◆ 基于概念的聚类(Concept-based clusters)
- ◆ 基于目标函数的聚类(Objective Function clusters)

# 良分割聚类

- 每个点到簇内其它点的距离都小于到簇外其它点的距离。

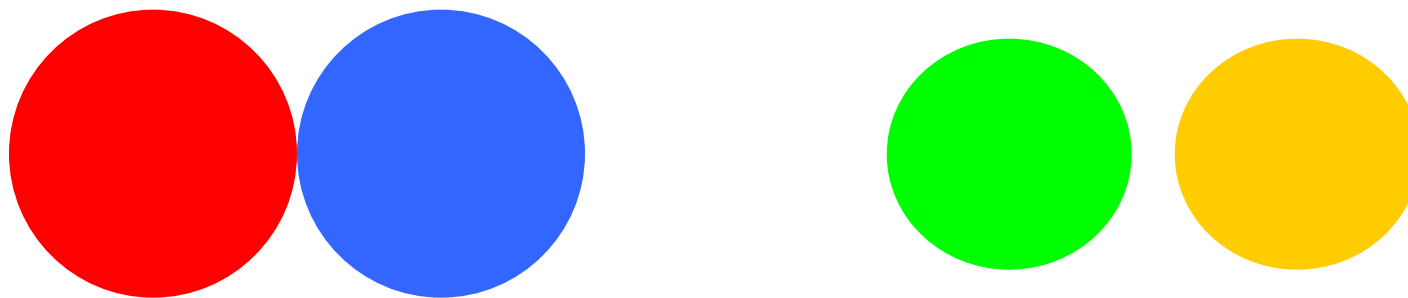


**3 well-separated clusters**



# 基于中心的聚类

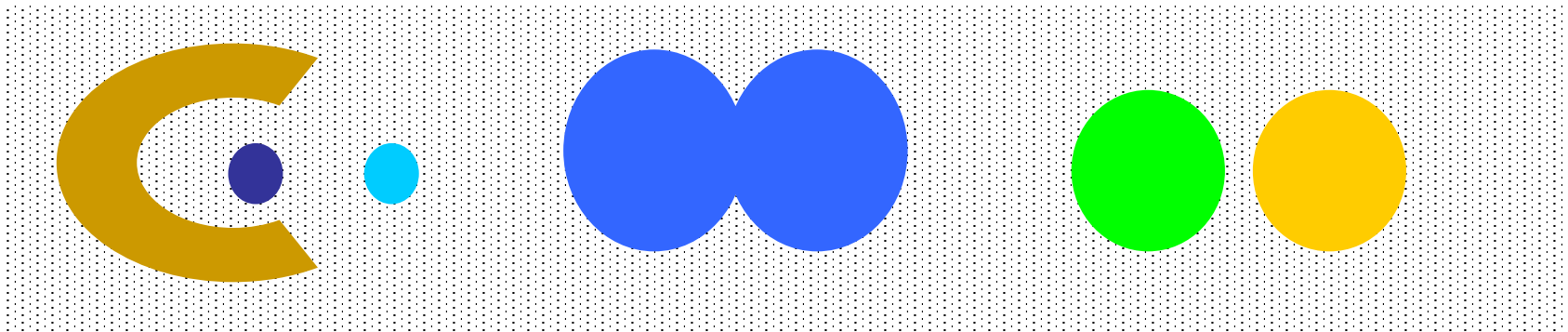
- ◆ 每个点与其所在簇中心的距离都小于与其它簇中心的距离。
- ◆ 簇的中心可以是质心(**centroid**), 即簇内所有点的均值; 或是**medoid**, 即簇内最有代表性的点。(它到其他所有(当前cluster中的)点的距离之和最小)



**4 center-based clusters**

# 基于密度的聚类

- 簇是密度超过一定阈值的点的集合，簇与簇之间被一些低密度区域分开。
- 适用于簇不规则(irregular)、或相互纠缠(intertwined), 及存在噪声和孤立点(outlier) 的情况。



**7 density-based clusters**

# 基于目标函数的聚类（1）

- 用目标函数来定义簇

- 构建簇，使目标函数最大或最小。

- 枚举所有可能的把对象点构成簇的方式，并使用目标函数对这些簇进行评价，评价结果最好的作为最终的聚类方式。(NP Hard)

- 可以有全局和局部目标

- 层次聚类通常是局部目标

- 分割聚类通常是全局目标

## 基于目标函数的聚类（2）

- 可将聚类问题映射到不同的域，并在这些域内求解问题：
  - 相关矩阵(**Proximity matrix**)定义了一个加权图，其中结点是用于聚类的点，加权的边代表结点间的相关性。
  - 聚类等价于将图分割为互连的子图，每个子图代表一个簇。
  - 目标是使簇间的权重和最小，簇内的权重和最大。

# 第 2 讲

## 无监督学习



1

概述

2

相似性计算方法

3

划分方法

4

层次方法

## 2.2 相似性计算方法

- 在无监督学习中，样本之间的相似性通常采用样本之间的距离来表示。
  - 两个样本之间的距离越大，表示两个样本越不相相似性，差异性越大；
  - 两个样本之间的距离越小，表示两个样本越相似性，差异性越小。
  - 特例：当两个样本之间的距离为零时，表示两个样本完全一样，无差异。

## 2.2 相似性计算方法

- 在无监督学习中，样本之间的相似性通常采用样本之间的距离来表示。
  - 样本之间的距离是在样本的描述属性（特征）上计算的。
  - 在不同应用领域，样本的描述属性的类型可能不同，因此相似性的计算方法也不尽相同。
    - 连续型属性(如：重量、高度、年龄等)
    - 二值离散型属性(如：性别、考试是否通过等)
    - 多值离散型属性(如：收入分为高、中、低等)
    - 混合类型属性(上述类型的属性至少同时存在两种)

# 1.连续型属性的相似性计算方法

- 假设两个样本 $X_i$ 和 $X_j$ 分别表示成如下形式：
  - $X_i=(x_{i1}, x_{i2}, \dots, x_{id})$
  - $X_j=(x_{j1}, x_{j2}, \dots, x_{jd})$
  - 它们都是 $d$ 维的特征向量，并且每维特征都是一个连续型数值。
- 对于连续型属性，样本之间的相似性通常采用如下三种距离公式进行计算。



# 1.连续型属性的相似性计算方法

## ■ 欧氏距离 (Euclidean distance)

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2} \quad \leftarrow q=2$$

## ■ 曼哈顿距离 (Manhattan distance)

$$d(x_i, x_j) = \sum_{k=1}^d |x_{ik} - x_{jk}| \quad \leftarrow q=1$$

## ■ 闵可夫斯基距离 (Minkowski distance)

$$d(x_i, x_j) = \left( \sum_{k=1}^d |x_{ik} - x_{jk}|^q \right)^{1/q}$$

# 1.连续型属性的相似性计算方法

## ■ *Euclidean*距离和*Manhattan*距离的性质:

- $d(i,j) \geq 0$

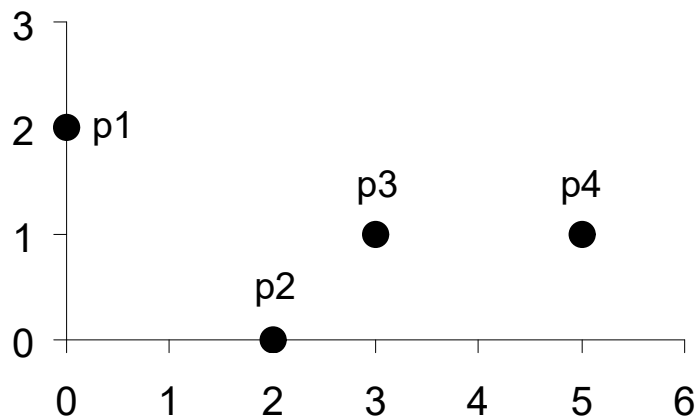
- $d(i,i) = 0$

- $d(i,j) = d(j,i)$

- $d(i,j) \leq d(i,k) + d(k,j)$

# 1.连续型属性的相似性计算方法

## ■ 欧式距离的示例



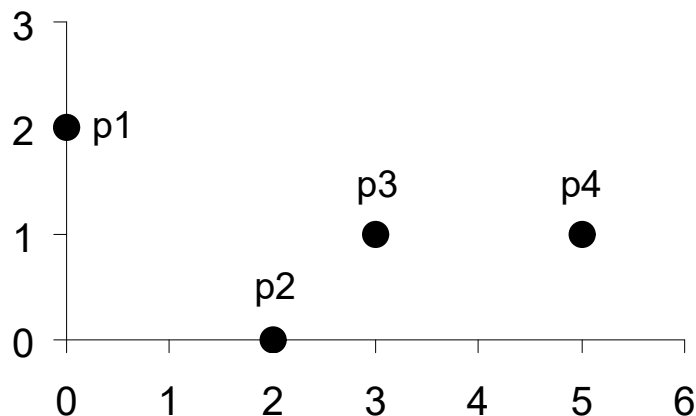
point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

**Distance Matrix**

# 1.连续型属性的相似性计算方法

## ■ 曼哈顿距离的示例



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

**Distance Matrix**

## 2.二值离散型属性的相似性计算方法

- 二值离散型属性只有0和1两个取值。
  - 其中：0表示该属性为空，1表示该属性存在。
  - 例如：描述病人是否抽烟的属性(*smoker*)，取值为1表示病人抽烟，取值0表示病人不抽烟。
- 假设两个样本 $X_i$ 和 $X_j$ 分别表示成如下形式：
  - $X_i=(x_{i1}, x_{i2}, \dots, x_{ip})$
  - $X_j=(x_{j1}, x_{j2}, \dots, x_{jp})$
  - 它们都是 $p$ 维的特征向量，并且每维特征都是一个二值离散型数值。

## 2. 二值离散型属性的相似性计算方法

- 假设二值离散型属性的两个取值具有相同的权重，则可以得到一个两行两列的**可能性矩阵**。

		$X_j$		
		1	0	<i>sum</i>
$X_i$	1	$a$	$b$	$a + b$
	0	$c$	$d$	$c + d$
<i>sum</i>		$a + c$	$b + d$	$p$

- $a$  = the number of attributes where  $X_i$  was 1 and  $X_j$  was 1;  
 $b$  = the number of attributes where  $X_i$  was 1 and  $X_j$  was 0;  
 $c$  = the number of attributes where  $X_i$  was 0 and  $X_j$  was 1;  
 $d$  = the number of attributes where  $X_i$  was 0 and  $X_j$  was 0.

## 2.二值离散型属性的相似性计算方法

- 如果样本的属性都是**对称的**二值离散型属性，则样本间的距离可用**简单匹配系数**(Simple Matching Coefficients, SMC)计算：

$$SMC = (b + c) / (a + b + c + d)$$

- 其中：**对称的**二值离散型属性是指属性取值为1或者0同等重要。
- 例如：性别就是一个**对称的**二值离散型属性，即：用1表示男性，用0表示女性；或者用0表示男性，用1表示女性是等价的，属性的两个取值没有主次之分。

## 2.二值离散型属性的相似性计算方法

- 如果样本的属性都是**不对称的**二值离散型属性，则样本间的距离可用**Jaccard系数**计算 (Jaccard Coefficients, JC):

$$JC = (b + c) / (a + b + c)$$

- 其中：**不对称的**二值离散型属性是指属性取值为1或者0不是同等重要。
- 例如：血液的检查结果是**不对称的**二值离散型属性，阳性结果的重要程度高于阴性结果，因此通常用1来表示阳性结果，而用0来表示阴性结果。



## 2.二值离散型属性的相似性计算方法

■ 例：已知两个样本 $p=[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ ,

■  $q=[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$

□  $a=?$

□  $b=?$

□  $c=?$

□  $d=?$

## 2.二值离散型属性的相似性计算方法

- $SMC = ?$
- $JC = ?$

### 3.多值离散型属性的相似性计算方法

- 多值离散型属性是指取值个数大于2的离散型属性。
  - 例如：成绩可以分为优、良、中、差。
- 假设一个多值离散型属性的取值个数为 $N$ ，给定数据集 $X=\{x_i \mid i=1,2,\dots,\text{total}\}$ 。
  - 其中：每个样本 $x_i$ 可用一个 $d$ 维特征向量描述，并且每维特征都是一个多值离散型属性，即：
  - $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 。

每维特征都是一个多值离散型属性。

### 3.多值离散型属性的相似性计算方法

- **问题：**给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？
  - 方法一：简单匹配方法。
  - 方法二：先将多值离散型属性转换成多个二值离散型属性，然后再使用**Jaccard系数**计算样本之间的距离。

### 3.多值离散型属性的相似性计算方法

- 问题：给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？

- 方法一：简单匹配方法。

- 距离计算公式如下：

$$d(x_i, x_j) = \frac{d - u}{d}$$

- 其中：  $d$ 为数据集中的属性个数，  $u$ 为样本 $x_i$ 和 $x_j$ 取值相同的属性个数。

### 3.多值离散型属性的相似性计算方法

■ 问题：给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？

□ 方法一：简单匹配方法。

□  $d(x_1, x_2) = ?$

样本序号	年龄段	学历	收入
$x_1$	青年	研究生	高
$x_2$	青年	本科	低
$x_3$	老年	本科以下	中
$x_4$	中年	研究生	高

### 3.多值离散型属性的相似性计算方法

■ 问题：给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？

□ 方法一：简单匹配方法。

□  $d(x_1, x_3) = ?$

样本序号	年龄段	学历	收入
$x_1$	青年	研究生	高
$x_2$	青年	本科	低
$x_3$	老年	本科以下	中
$x_4$	中年	研究生	高

### 3.多值离散型属性的相似性计算方法

■ 问题：给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？

□ 方法一：简单匹配方法。

□  $d(x_1, x_4) = ?$

样本序号	年龄段	学历	收入
$x_1$	青年	研究生	高
$x_2$	青年	本科	低
$x_3$	老年	本科以下	中
$x_4$	中年	研究生	高



### 3.多值离散型属性的相似性计算方法

- **问题：**给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？
  - 方法二：先将多值离散型属性转换成多个二值离散型属性，然后再使用**Jaccard系数**计算样本之间的距离。
    - 对有**N个**取值的多值离散型属性，可依据该属性的**每种取值**分别创建一个新的二值离散型属性，这样可将多值离散型属性转换成多个二值离散型属性。

### 3.多值离散型属性的相似性计算方法

- **问题：**给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？
  - 方法二：先将多值离散型属性转换成多个二值离散型属性，然后再使用**Jaccard系数**计算样本之间的距离。

样本序号	年龄段	学历	收入
$x_1$	青年	研究生	高
$x_2$	青年	本科	低
$x_3$	老年	本科以下	中
$x_4$	中年	研究生	高

### 3.多值离散型属性的相似性计算方法

- **问题：**给定两个样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ 和 $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，如何计算它们之间的距离？
  - 方法二：先将多值离散型属性转换成多个二值离散型属性，然后再使用**Jaccard系数**计算样本之间的距离。

样本序号	青年	中年	老年	本科以下	本科	研究生	高	中	低
$x_1$	0	0	1	0	0	1	1	0	0
$x_2$	0	0	1	0	1	0	0	0	1
$x_3$	1	0	0	1	0	0	0	1	0
$x_4$	0	1	0	0	0	1	1	0	0

## 4.混合类型属性的相似性计算方法

- 在实际中，数据集中数据的**描述属性**通常不只一种类型，而是各种类型的混合体。
  - 连续型属性
  - 二值离散型属性
  - 多值离散型属性

## 4.混合类型属性的相似性计算方法

- 问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？
- 方法：将混合类型属性放在一起处理。
  - 假设：给定的数据集 $X=\{x_i \mid i=1,2,\dots,\text{total}\}$ ，每个样本用 $d$ 个描述属性 $A_1, A_2, \dots, A_d$ 来表示，属性 $A_j(1 \leq j \leq d)$ 包含多种类型。

## 4.混合类型属性的相似性计算方法

- **问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？**
- **方法：将混合类型属性放在一起处理。**
  - 在聚类之前，对样本的属性值进行预处理：
    - 对**连续型属性**，将其各种取值进行规范化处理，使得属性值规范化到区间[0.0, 1.0]；
    - 对**多值离散型属性**，根据属性的每种取值将其转换成多个二值离散型属性。
    - 预处理之后，样本中**只包含**连续型属性和二值离散型属性。

## 4.混合类型属性的相似性计算方法

- 问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？

- 给定两个样本  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})$  和  $\mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jd})$ ，它们之间的距离为：

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^d \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^d \delta_{ij}^{(k)}}$$

- 其中： $d_{ij}^{(k)}$ 表示 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 在第 $k$ 个属性上的距离。  
 $\delta_{ij}^{(k)}$ 表示第 $k$ 个属性对计算 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 距离的影响。

## 4.混合类型属性的相似性计算方法

- 问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？

- $d_{ij}^{(k)}$ 表示 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 在第 $k$ 个属性上的距离。

- 当第 $k$ 个属性为连续型时，使用如下公式来计算 $d_{ij}^{(k)}$ ：

$$d_{ij}^{(k)} = | \mathbf{x}_{ik} - \mathbf{x}_{jk} |$$

- 当第 $k$ 个属性为二值离散型时，如果 $\mathbf{x}_{ik} = \mathbf{x}_{jk}$ ，则 $d_{ij}^{(k)} = 0$ ；否则， $d_{ij}^{(k)} = 1$ 。



## 4.混合类型属性的相似性计算方法

- 问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？
  - $\delta_{ij}^{(k)}$ 表示第 $k$ 个属性对计算 $x_i$ 和 $x_j$ 距离的影响。
    - （1）如果 $x_{ik}$ 或 $x_{jk}$ 缺失（即：样本 $x_i$ 或样本 $x_j$ 没有第 $k$ 个属性的度量值），则： $\delta_{ij}^{(k)} = 0$ 。
    - （2）如果 $x_{ik}=x_{jk}=0$ ，且第 $k$ 个属性是不对称的二值离散型，则： $\delta_{ij}^{(k)} = 0$ 。
    - （3）除了上述（1）和（2）之外的其他情况下，则： $\delta_{ij}^{(k)} = 1$ 。

# 第 2 讲

## 无监督学习



1

概述

2

相似性计算方法

3

划分方法

4

层次方法

## 2.3 划分方法

- 给定 $n$ 个样本的数据集，以及要生成的簇的数目 $k$ ，**划分方法**将样本组织为 $k$ 个划分（ $k \leq n$ ），每个划分代表一个簇。
  - 划分准则：同一个簇中的样本尽可能接近或相似，不同簇中的样本尽可能远离或不相似。
  - 以样本间的距离作为相似性度量。
- 典型的划分方法：
  - k-means（k-均值）
    - 由簇中样本的平均值来代表整个簇。
  - k-medoids（k-中心点）
    - 由处于簇中心区域的某个样本代表整个簇。

## 2.3.1 K-means算法

- 每个簇都有一个中心点(centroid)
- 每个点最终都归属于一个与其距离最近的中心点所决定的簇。
- 簇的数目 $K$ 需要事先给定
- 基本的算法非常简单

## 2.3.1 k-means算法

### ■ k均值算法流程

- ✓ 1.随机选择 $k$ 个对象，每个对象代表一个簇的初始均值或中心
- ✓ 2.对剩余的每个对象，根据它与簇均值的距离，将他指派到最相似的簇
- ✓ 3.计算每个簇的新均值
- ✓ 4.回到步骤2，循环，直到准则函数收敛

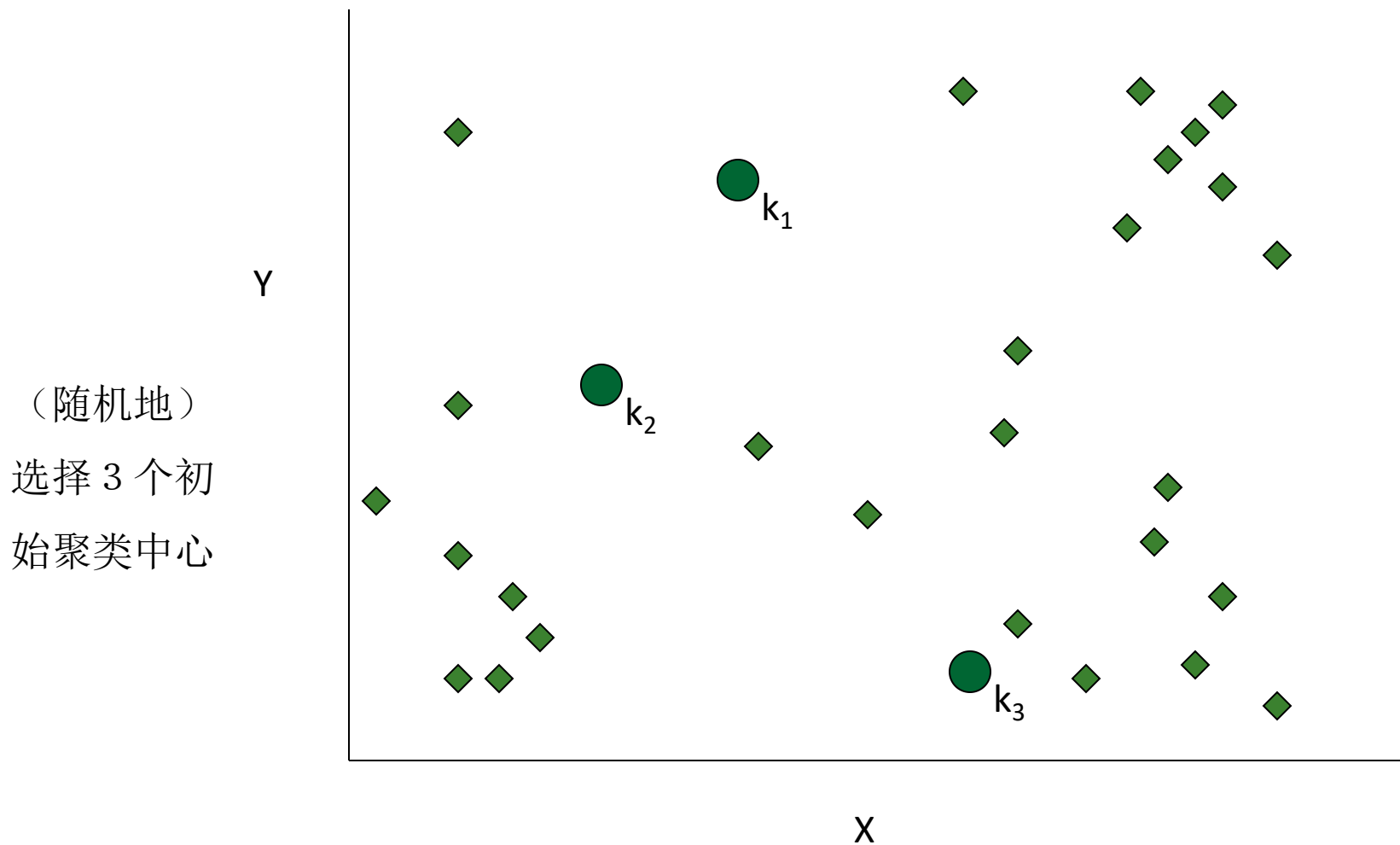
### □ 常用准则函数：平方误差准则

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2 \quad (p \text{ 是空间中的点, } m_i \text{ 是簇 } C_i \text{ 的均值})$$

## 2.3.1 K-means算法的细节

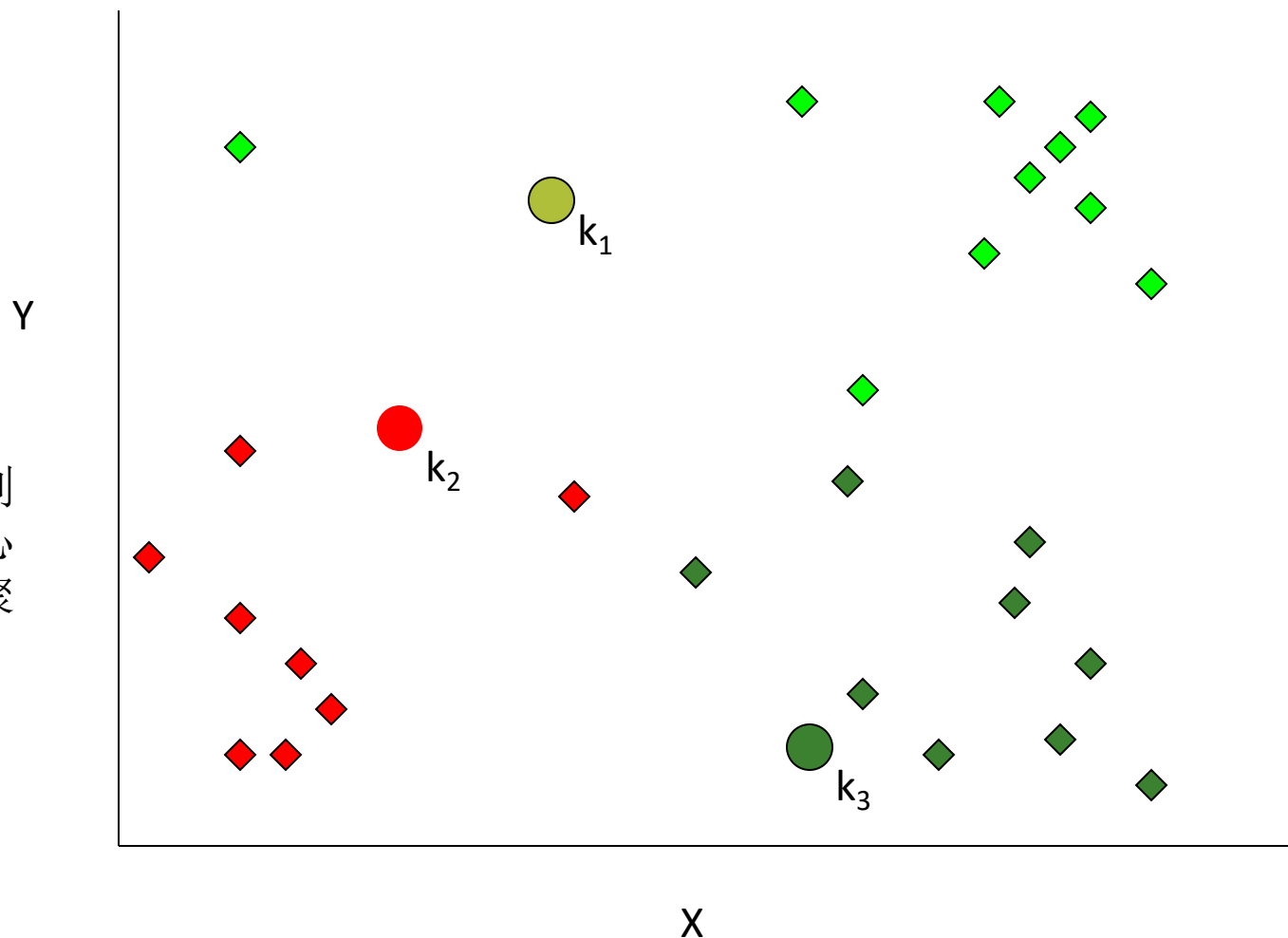
- 初始中心点是随机选择的
  - 每次迭代之后簇往往会发生变化.
- 中心点一般是该簇的均值.
- “相似性”一般通过Euclidean距离, cosine 相似性等来度量.
- 在以上这些相似性度量标准下, K-平均聚类一般都会收敛.

## 2.3.1 k-means算法——例1



# K-means 聚类算法

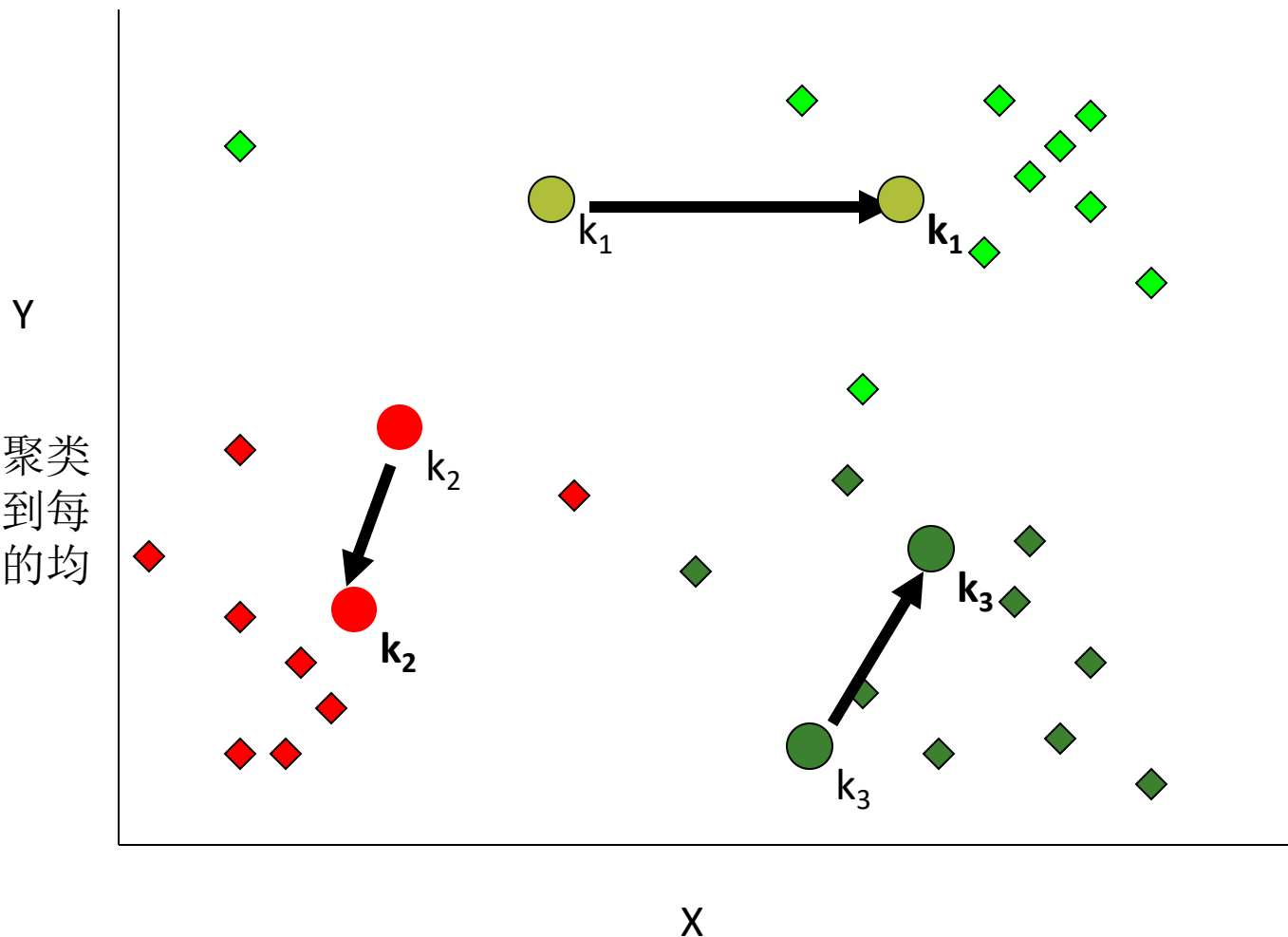
每一个点被划分到离某中心最近的那个聚类





# K-means 聚类算法

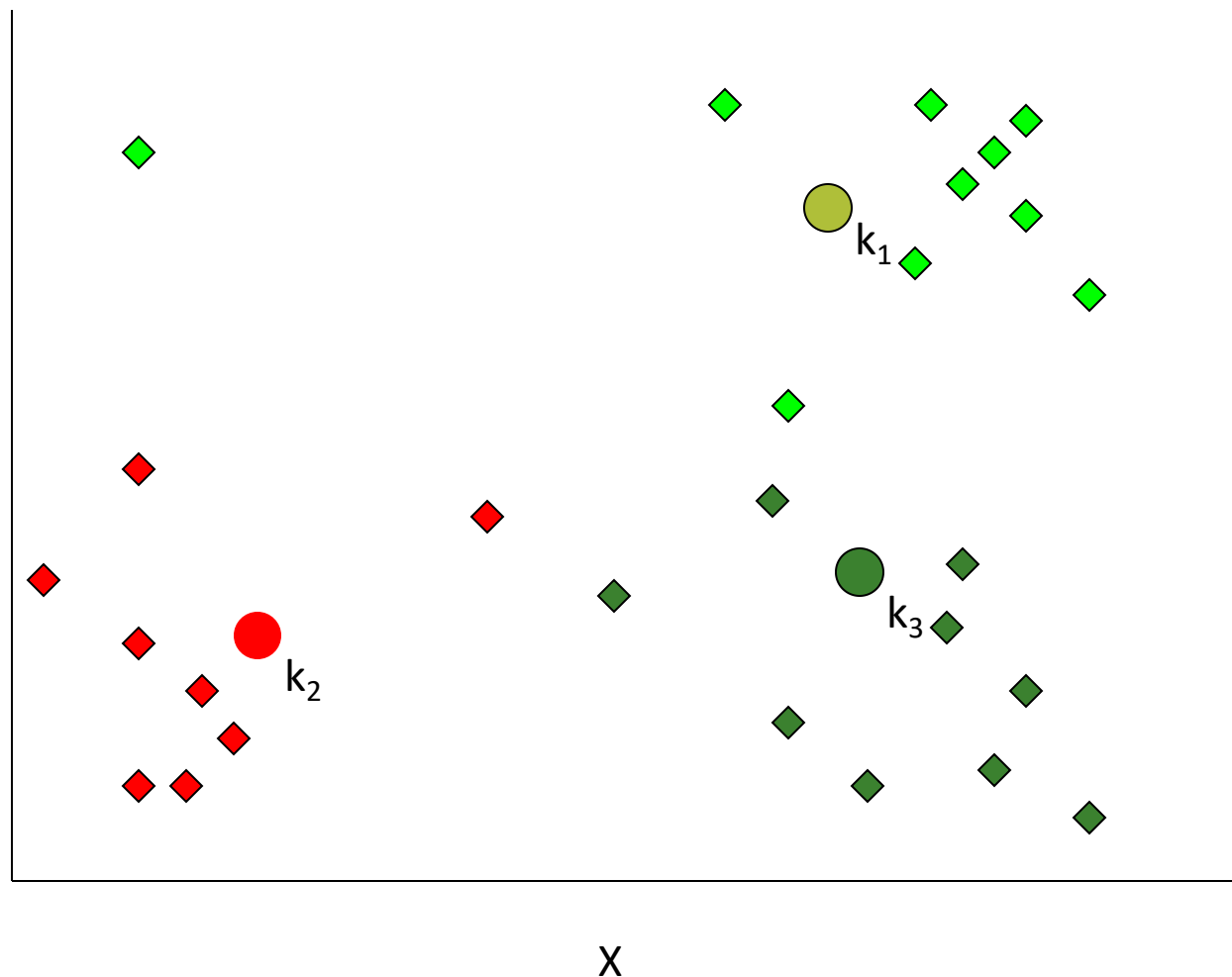
把每一个聚类  
中心移动到每  
一个聚类的均  
值上



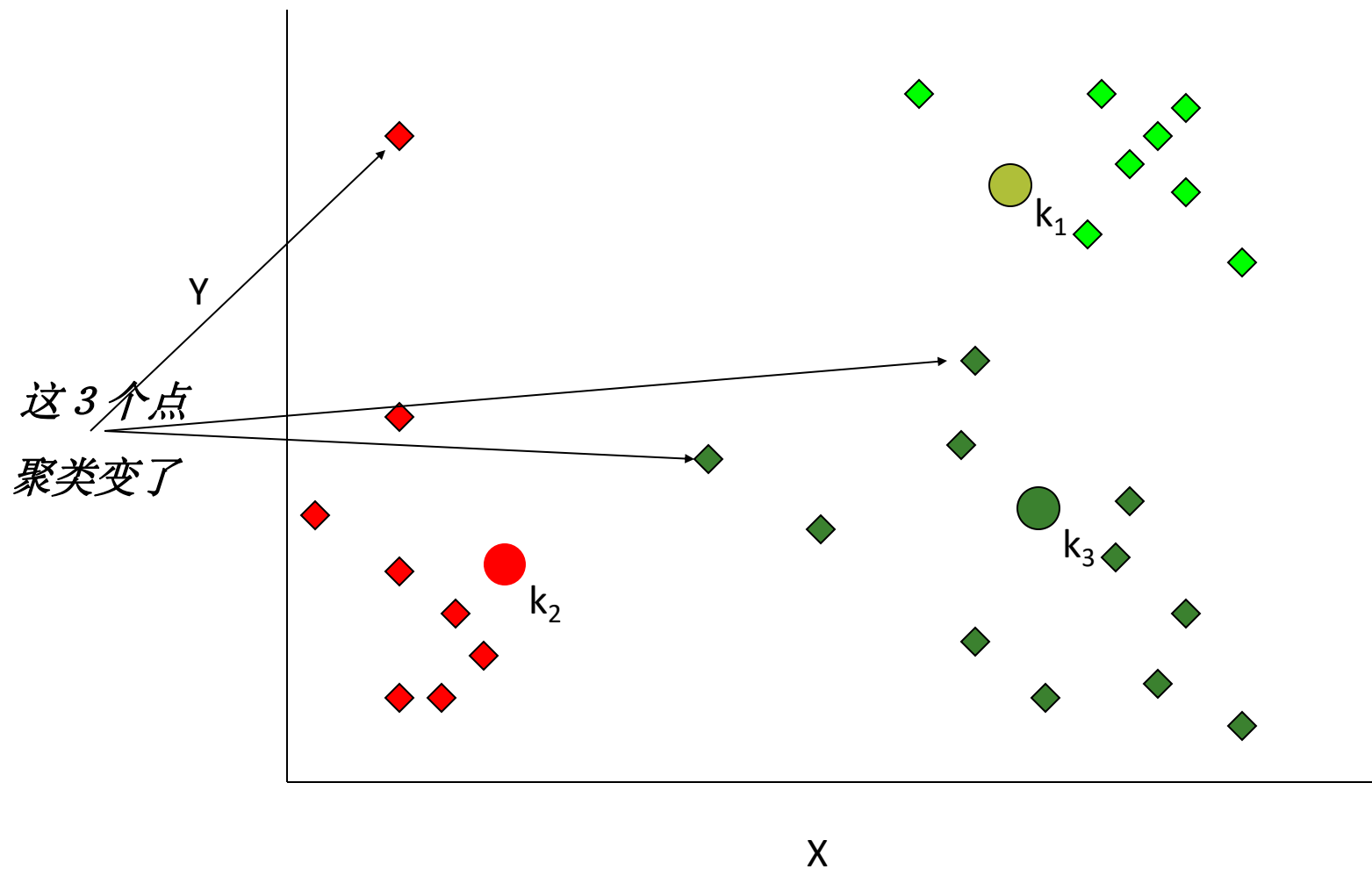
# K-means 聚类算法

重新把这些点  
安排到中心最  
近的聚类

哪些点的聚类  
变了?

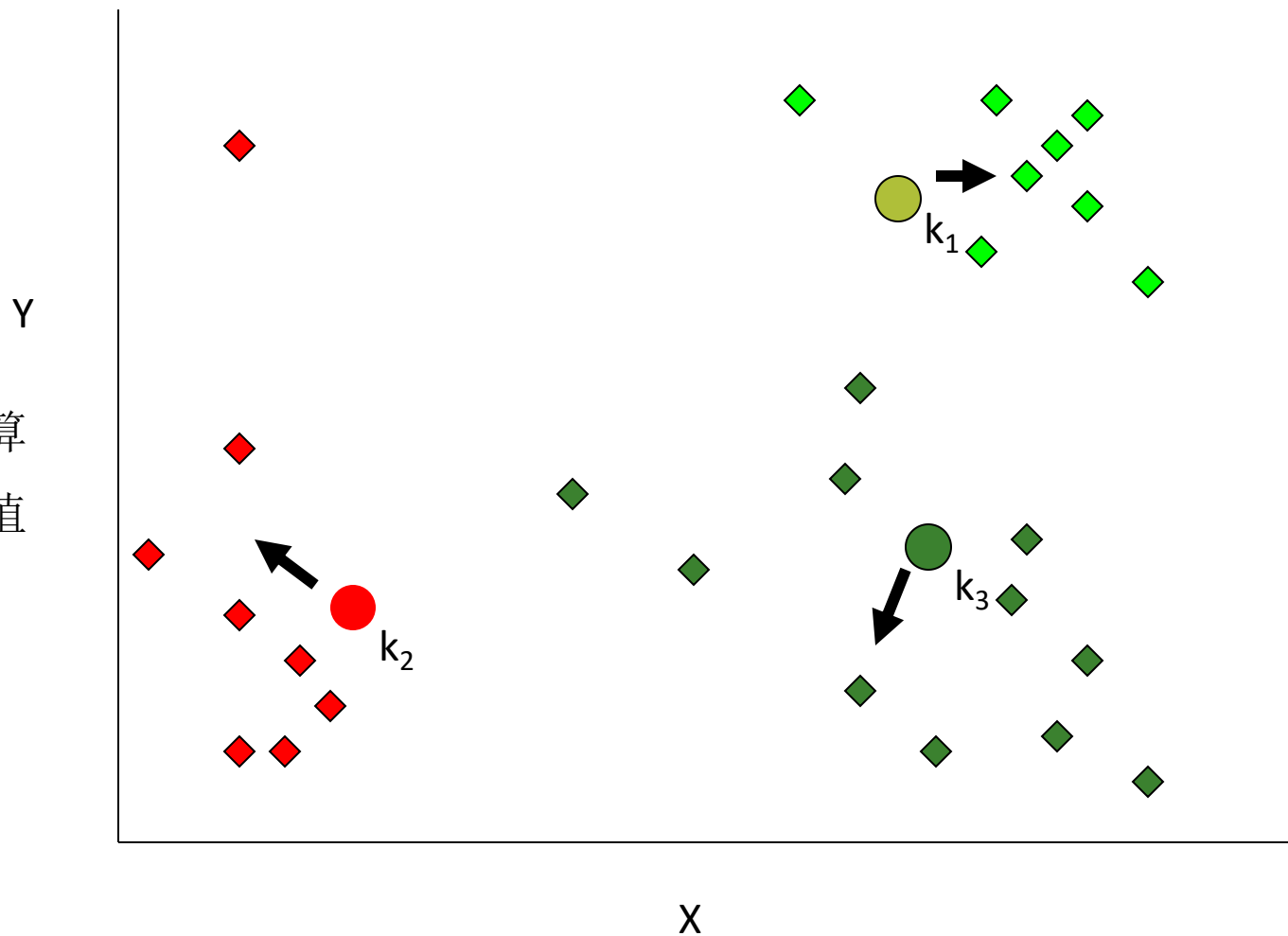


# K-means 聚类算法

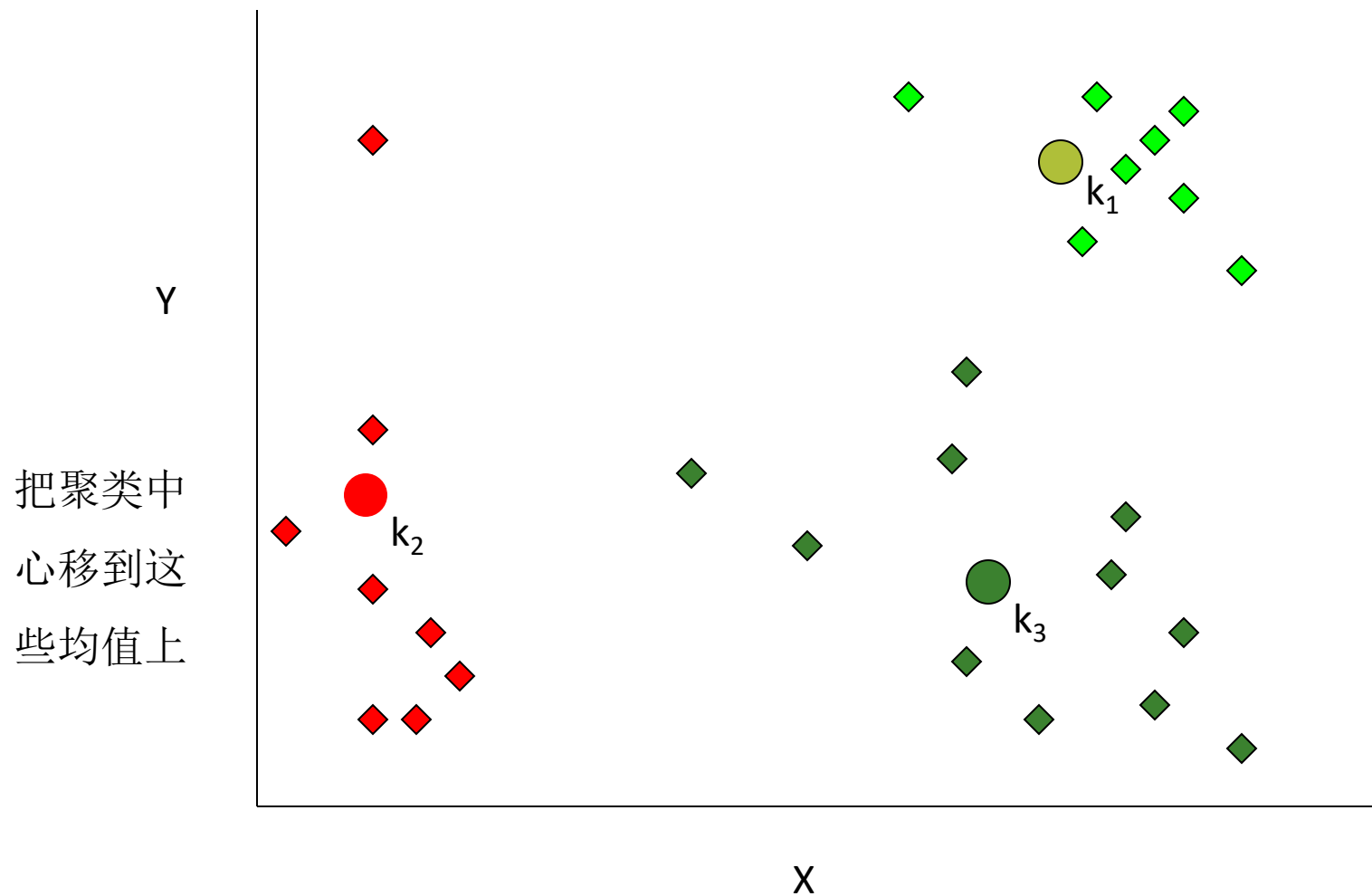


# K-means 聚类算法

重新计算  
聚类均值



# K-means 聚类算法



## 2.3.1 k-means算法

- k-means算法的评价准则：误差平方和准则

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2$$

- 误差平方和达到最优（小）时，可以使各聚类的类内尽可能紧凑，而使各聚类之间尽可能分开。
- 对于同一个数据集，由于k-means算法对初始选取的聚类中心敏感，因此可用该准则评价聚类结果的优劣。
- 通常，对于任意一个数据集，k-means算法无法达到全局最优，只能达到局部最优。

## 2.3.1 k-Means聚类

### ■ k-Means算法实现过程：

- ❑ （1）导入数据：收集并准备数值型矩阵作为原始数据集；
- ❑ （2）创建初始质心：指定聚类数 $k$ ，从原始数据集中随机选取 $k$ 个对象作为初始质心；
- ❑ （3）分配：计算数据集中每个对象到所有质心的距离，并将数据点分配到距离最接近的质心，从而形成簇分配矩阵；
- ❑ （4）重新计算质心：计算簇中所有点的均值，并将均值作为新的质心。
- ❑ （5）反复执行（3）和（4），直至质心不再移动。

## 2.3.1 k-Means聚类

- 数据导入：从文本文件中读入数据到矩阵
- 计算欧氏距离
- 构建 $k$ 个随机质心
- $k$ Means算法

### kMeans的编码实现

```
In [48]: import numpy as np
import matplotlib.pyplot as plt

In [49]: def loadFile(path):
dataList = []
#打开文件: 以二进制读模式、utf-8格式的编码方式打开
fr = open(path, 'r', encoding='UTF-8')
record = fr.read()
fr.close
#按照行转换为二维表即包含各行作为元素的列表, 分隔符有'lr', 'lrln', 'ln'
recordList = record.splitlines()
#逐行遍历: 行内字段按'\t'分隔符分隔, 转换为列表
for line in recordList:
    if line.strip():
        dataList.append(list(map(float, line.split('\t'))))
#返回转换后的矩阵
recordmat = np.mat(dataList)
return recordmat

In [50]: def distEclud(vecA, vecB):
return np.linalg.norm(vecA - vecB, ord=2)

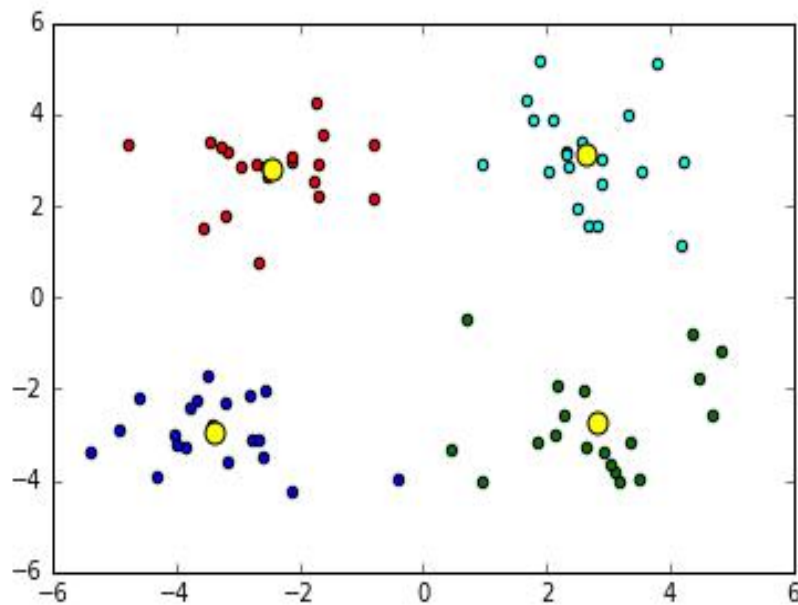
In [51]: def randCentrs(dataSet, k):
n = np.shape(dataSet)[1]
cents = np.mat(np.zeros((k, n)))
for j in range(n):
    #质心必须在数据集范围内, 也就是在min到max之间
    minCol = min(dataSet[:, j])
    maxCol = max(dataSet[:, j])
    #利用随机函数生成0到1.0之间的随机数
    cents[:, j] = np.mat(minCol + float(maxCol - minCol) * np.random.rand(k, 1))
return cents

In [52]: def kMeans(dataset, k):
m = np.shape(dataset)[0]
ClustDist = np.mat(np.zeros((m, 2)))
cents = randCentrs(dataset, k)
clusterChanged = True
# 循环迭代, 得到最近的聚类中心
while clusterChanged:
    clusterChanged = False
    for i in range(m):
```



## 2.3.1 k-Means聚类

- 经过三次迭代之后 $k$ -Means算法收敛，形成四个聚类中心。运行结果：



## 2.3.1 k-means算法

### ■ 优点:

- 可扩展性较好，算法复杂度为 $O(nkt)$ 。

- 其中： $n$ 为样本个数， $k$ 是簇的个数， $t$ 是迭代次数。

### ■ 缺点:

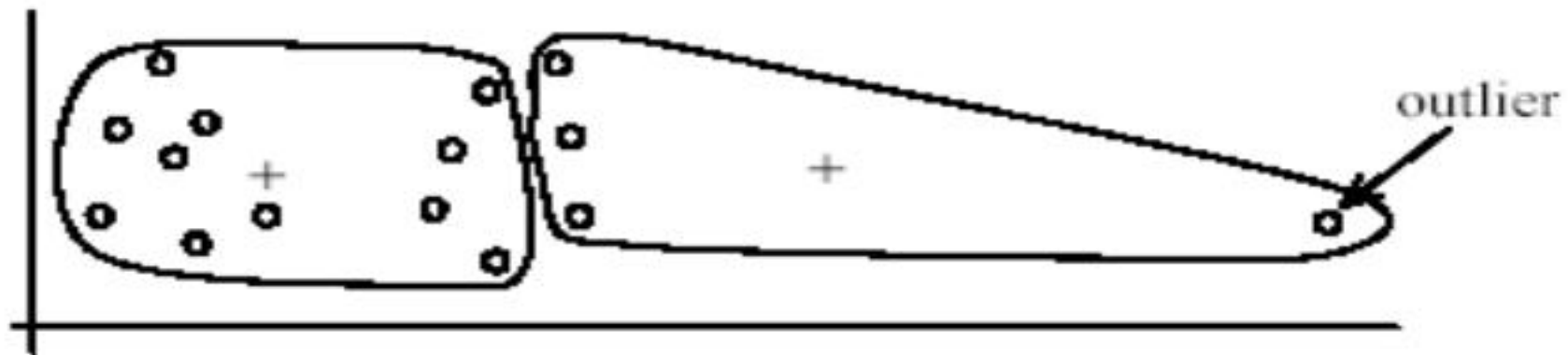
- 簇数目 $k$ 需要事先给定，但非常难以选定；

- 初始聚类中心的选择对聚类结果有较大的影响；

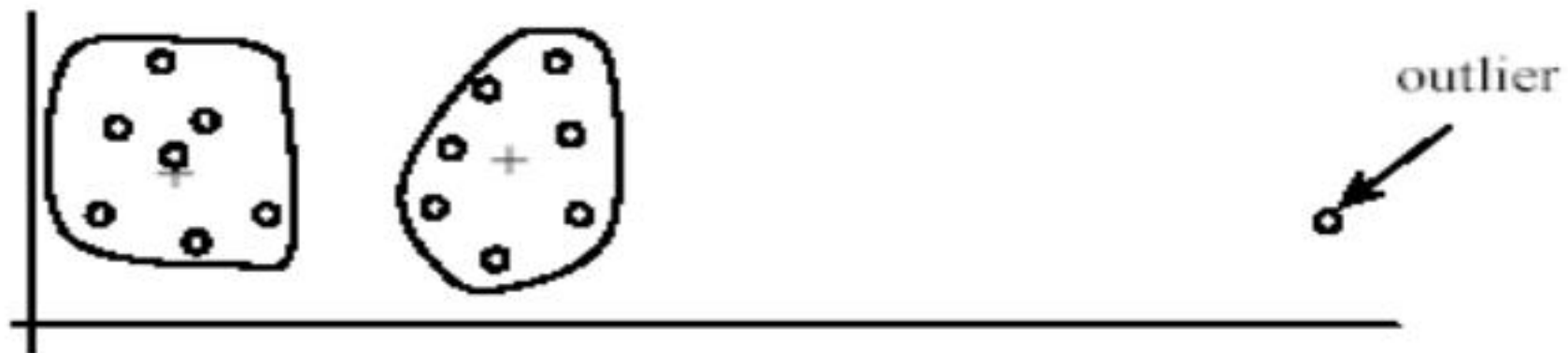
- 不适合于发现非球状簇，或者大小差别很大的簇；

- 对噪声和离群点数据敏感。

## 2.3.1 k-means

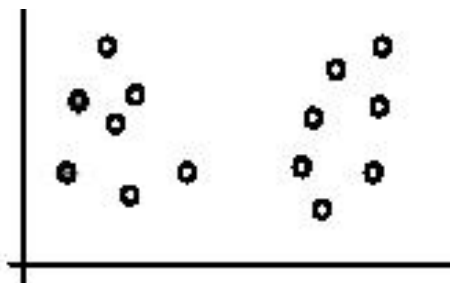


(A): Undesirable clusters

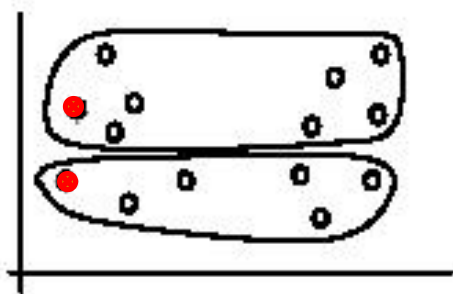


(B): Ideal clusters

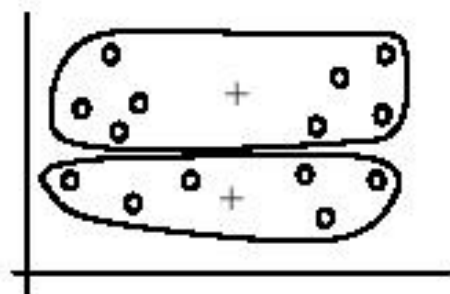
## 2.3.1 k-means



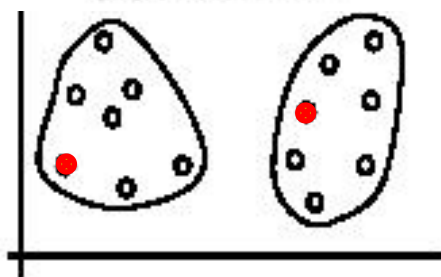
(A). Random selection of seeds (centroids)



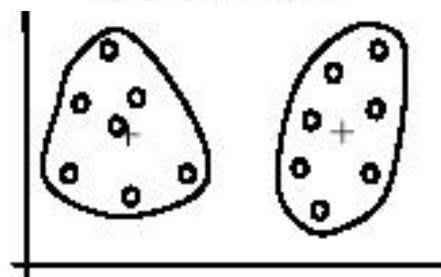
(B). Iteration 1



(C). Iteration 2

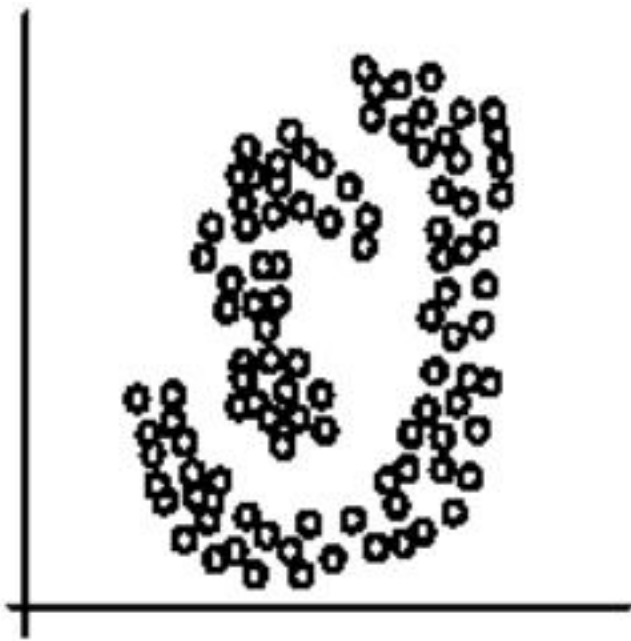


(B). Iteration 1

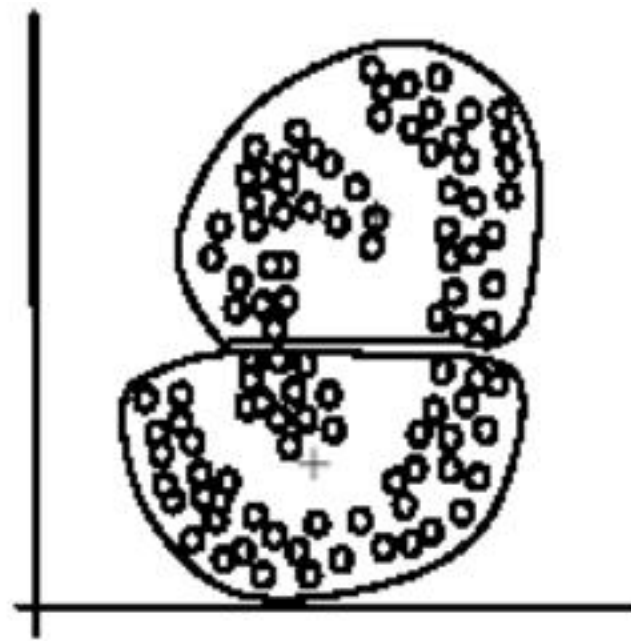


(C). Iteration 2

## 2.3.1 k-means



(A): Two natural clusters



(B):  $k$ -means clusters

## 2.3.1 k-means

### ■ k-means方法的改进策略

- ✓ 初始 $k$ 个均值的选择
- ✓ 相异度计算
- ✓ 计算簇均值的策略

# 初始点的选取

初始聚类中心的选取决定着计算的迭代次数，甚至决定着最终的解是否为全局最优。

首先选择第一个样本点作为第一个聚类中心。

然后选取距离第一个点最远的点作为第二个聚类中心。

.....

第 $j$ 个聚类中心要远离第 $1 \sim j-1$ 个聚类中心。

或者可以多次随机取初始聚类中心，得到聚类结果，取平均值。

## 初始点的选取

- 若有 $K$ 个实际的簇，则从每个簇中选择一个初始点的概率是非常小的。
- 多次执行
  - 可缓解，但效果不一定好
- 抽样并执行层次聚类来确定初始中心点
- 设置多于 $K$ 个的初始中心点，然后从中选取。
  - 选择 $K$ 个分割效果最好的中心点



## 2.3.2 k-medoids算法

- k-medoids算法基本思想：
  - 选取有代表性的样本（而不是均值）来表示整个簇，即：选取最靠近中心点(medoid)的那个样本来代表整个簇。
  - 以降低聚类算法对离群点的敏感度。
  - PAM (Partitioning Around Medoids, 围绕中心点的划分)算法，于1987年提出。

## 103.2 k-medoids算法

■如果**代表样本**能被**非代表样本**所替代，则替代产生的**总代价** $S$ 是**所有样本**产生的代价之和。

■总代价的定义如下：

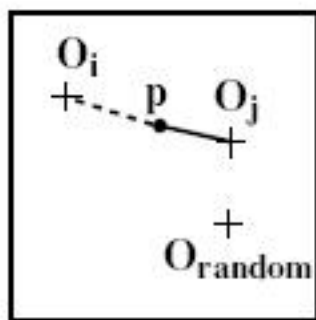
$$TC_{jh} = \sum_{p=1}^n C_{pjh}$$

■其中： $n$ 是数据集中样本的个数； $C_{pjh}$ 表示中心点 $O_j$ 被非中心点 $O_h$ 替代后，样本点 $p$ 的代价。

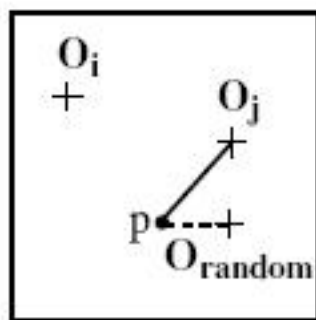
■**问题**：如何计算每个样本点 $p$ 产生的代价 $C_{pjh}$ ？

## 2.3.2 k-medoids算法

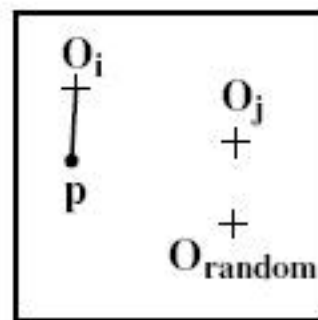
- 当非代表样本  $O_{random}$  替代代表样本  $O_j$  后，对于数据集中的每一个样本  $p$ ，它所属的簇的类别将有以下四种可能的变化：



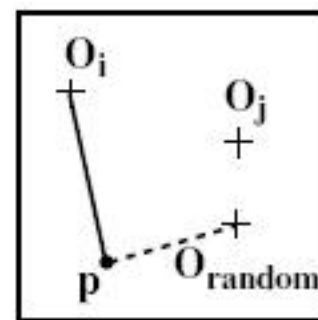
1. Reassigned to  $O_i$



2. Reassigned to  $O_{random}$



3. No change

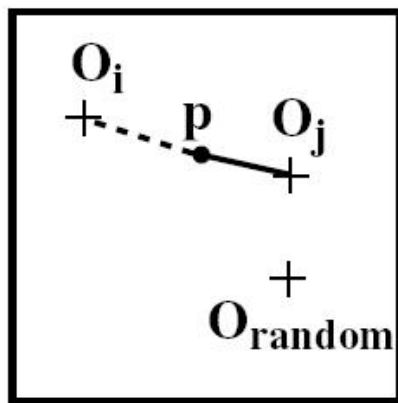


4. Reassigned to  $O_{random}$

- data object
- + cluster center
- before swapping
- after swapping

## 2.3.2 k-medoids算法

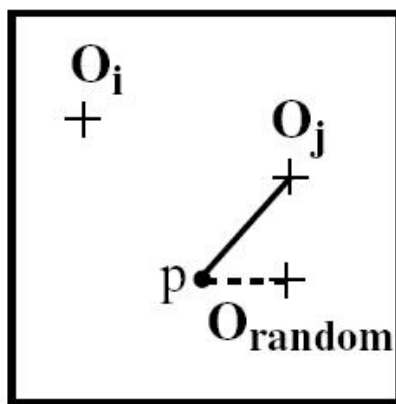
- 当非代表样本  $O_{random}$  替代代表样本  $O_j$  后，对于数据集中的每一个样本  $p$ ，它所属的簇的类别将有以下四种可能的变化：
  - 情况1：样本  $p$  属于代表样本  $O_j$ 。
  - 如果  $O_j$  被  $O_{random}$  替代，则此时样本  $p$  最接近另外一个代表样本  $O_i$ ，因此  $p$  被分配为  $O_i$  ( $i \neq j$ )。



1. Reassigned to  $O_i$

## 2.3.2 k-medoids算法

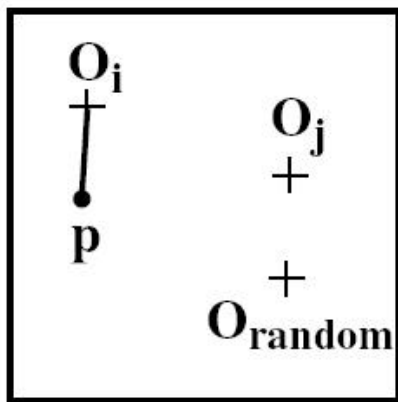
- 当非代表样本  $O_{random}$  替代代表样本  $O_j$  后，对于数据集中的每一个样本  $p$ ，它所属的簇的类别将有以下四种可能的变化：
  - 情况2：样本  $p$  属于代表样本  $O_j$ 。
  - 如果  $O_j$  被  $O_{random}$  替代，则此时样本  $p$  最接近代表样本  $O_{random}$ ，因此  $p$  被分配为  $O_{random}$ 。



2. Reassigned to  
 $O_{random}$

## 2.3.2 k-medoids算法

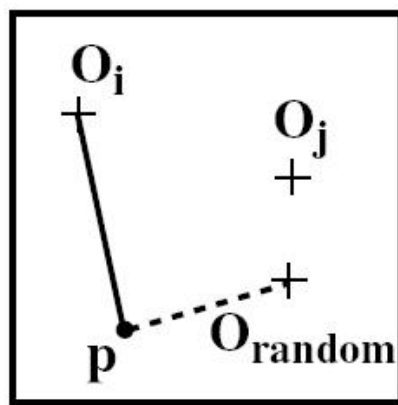
- 当非代表样本  $O_{random}$  替代代表样本  $O_j$  后，对于数据集中的每一个样本  $p$ ，它所属的簇的类别将有以下四种可能的变化：
  - 情况3：样本  $p$  属于代表样本  $O_i$  ( $i \neq j$ )。
  - 如果  $O_j$  被  $O_{random}$  替代，则此时样本  $p$  任然最接近代表样本  $O_i$ ，因此  $p$  无变化。



3. No change

## 2.3.2 k-medoids算法

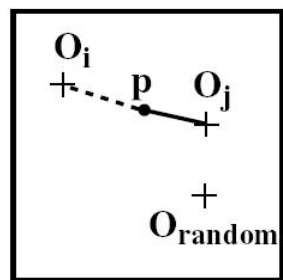
- 当非代表样本  $O_{random}$  替代代表样本  $O_j$  后，对于数据集中的每一个样本  $p$ ，它所属的簇的类别将有以下四种可能的变化：
  - 情况4：样本  $p$  属于代表样本  $O_i$  ( $i \neq j$ )。
  - 如果  $O_j$  被  $O_{random}$  替代，则此时样本  $p$  最接近代表样本  $O_{random}$ ，因此  $p$  被分配为  $O_{random}$ 。



4. Reassigned to  
 $O_{random}$

## 2.3.2 k-medoids算法

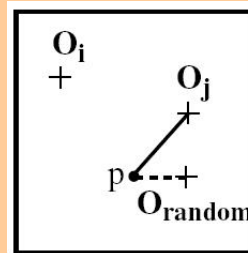
第一种情况



1. Reassigned to  $O_i$

$p$ 被重新分配给 $O_i$ ,  
 $C_{pjh} = d(p, i) - d(p, j)$

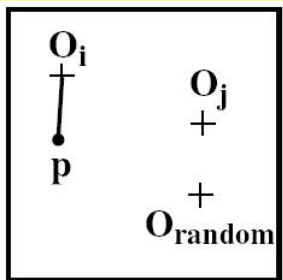
第二种情况



2. Reassigned to  
 $O_{\text{random}}$

$p$ 被重新分配给 $O_h$ ,  
 $C_{pjh} = d(p, h) - d(p, j)$

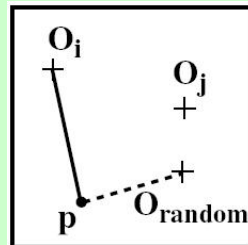
第三种情况



3. No change

$p$ 的隶属不发生变化,  
 $C_{pjh} = 0$

第四种情况



4. Reassigned to  
 $O_{\text{random}}$

$p$ 被重新分配给 $O_h$ ,  
 $C_{pjh} = d(p, h) - d(p, i)$



## 2.3.2 k-medoids算法

### ■ PAM算法

① 随机选择  $k$  个对象作为初始的代表对象；

② repeat

指派每个剩余的对象给离它最近的代表对象所代表的簇；

随意地选择一个非代表对象  $O_{random}$ ；

计算用  $O_{random}$  代替  $O_j$  的总代价  $S$ ；

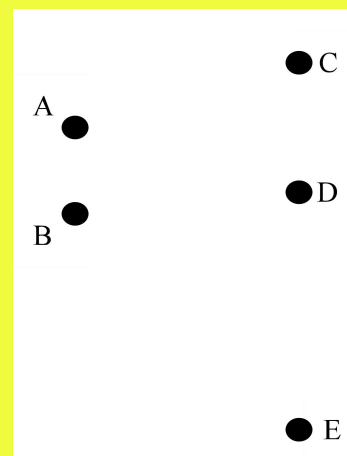
如果  $S < 0$ ，用  $O_{random}$  替换  $O_j$ ，形成新  $k$  个代表对象的集合；

③ until 不发生变化

## 2.3.2 k-medoids算法--例2

假设空间中的五个点{A、B、C、D、E}，如下图所示。各点之间的距离关系如下表所示，根据所给的数据对其运行k-medoids算法实现划分聚类（设 $k=2$ ）。

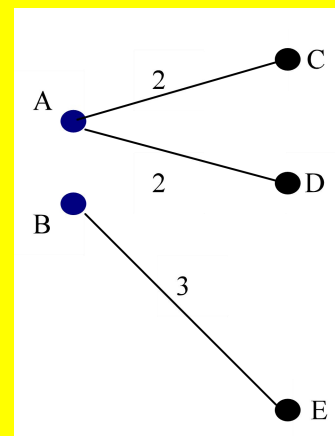
样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



## 2.3.2 k-medoids算法--例2

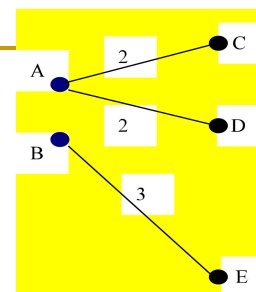
- 第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A, B}，则样本被划分为{A、C、D}和{B、E}，如图所示。

- 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  
 $TC_{AC}$ 、 $TC_{AD}$ 、 $TC_{AE}$ 、 $TC_{BC}$ 、 $TC_{BD}$ 、 $TC_{BE}$ 。



起始中心点为A,B

## 2.3.2 k-medoids算法--例2



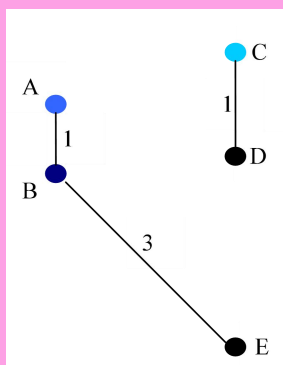
■ 以  $TC_{AC}$  为例说明计算的过程:

- 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更近, A被分配到B中心点代表的簇,  $C_{AAC}=d(A,B)-d(A,A)=1$ 。
  - B是一个中心点, 当A被C替换以后, B不受影响,  $C_{BAC}=0$ 。
  - 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,  $TC_{AC}=CA_{AC}+CB_{AC}+CB_{AC}+CD_{AC}+CE_{AC}=1+0-2-1+0=-2$ 。

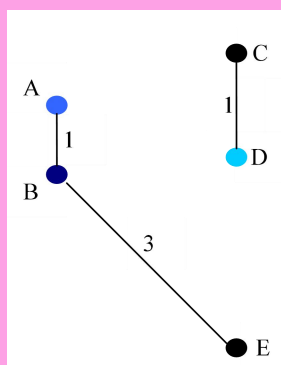
可按上述步骤依次计算代价  $TC_{AD}$ 、 $TC_{AE}$  以及  $TC_{BC}$ 、 $TC_{BD}$ 、 $TC_{BE}$ 。

## 2.3.2 k-medoids算法--例2

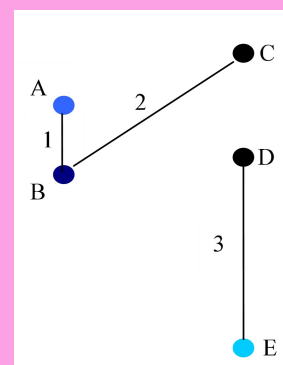
- 在上述代价计算完毕后，要选取一个代价最小的替换。
- 图（a）、（b）、（c）分别表示了C替换A， D替换A， E替换A的情况和相应的代价。



(a) C替换A,  $TC_{AC} = -2$



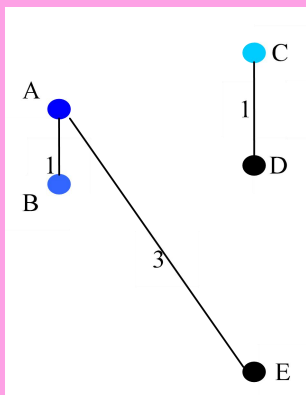
(b) D替换A,  $TC_{AD} = -2$



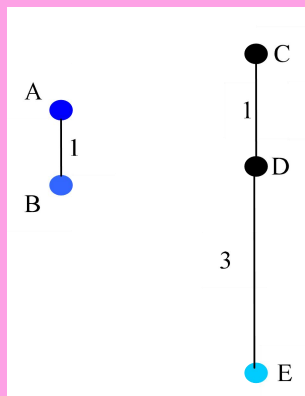
(c) E替换A,  $TC_{AE} = -1$

## 2.3.2 k-medoids算法--例2

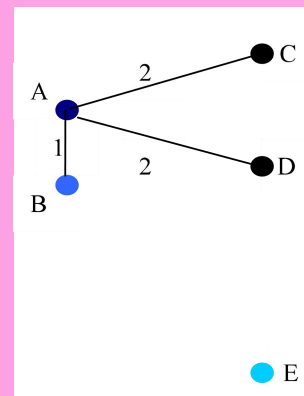
■ 图（d）、（e）、（f）分别表示了用C、D、E替换B的情况和相应的代价。



(d) C替换B,  $TC_{BC} = -2$



(e) D替换B,  $TC_{BD} = -2$



(f) E替换B,  $TC_{BE} = -2$

- 通过上述计算，选择代价最小的替换（如：C替换A）。
- 这样就完成了PAM算法的第一次迭代。在下一迭代中，将用其新的非中心点{A、D、E}来替换中心点{B、C}，从中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。

# k-medoids与k-means的比较

- 当存在噪声和离群点时，k-medoids算法比k-means算法更加鲁棒(稳定)。
  - 这是因为中心点不像均值那样易被极端数据(噪声或者离群点)影响。
- k-medoids算法的执行代价比k-means算法要高。
  - k-means算法:  $O(nkt)$
  - k-medoids算法:  $O(k(n-k)^2)$ 
    - 当n与k较大时，k-medoids算法的执行代价很高。
- 两种方法都需要事先指定簇的数目k。

# 第 2 讲

## 无监督学习

1

概述

2

相似性计算方法

3

划分方法

4

层次方法

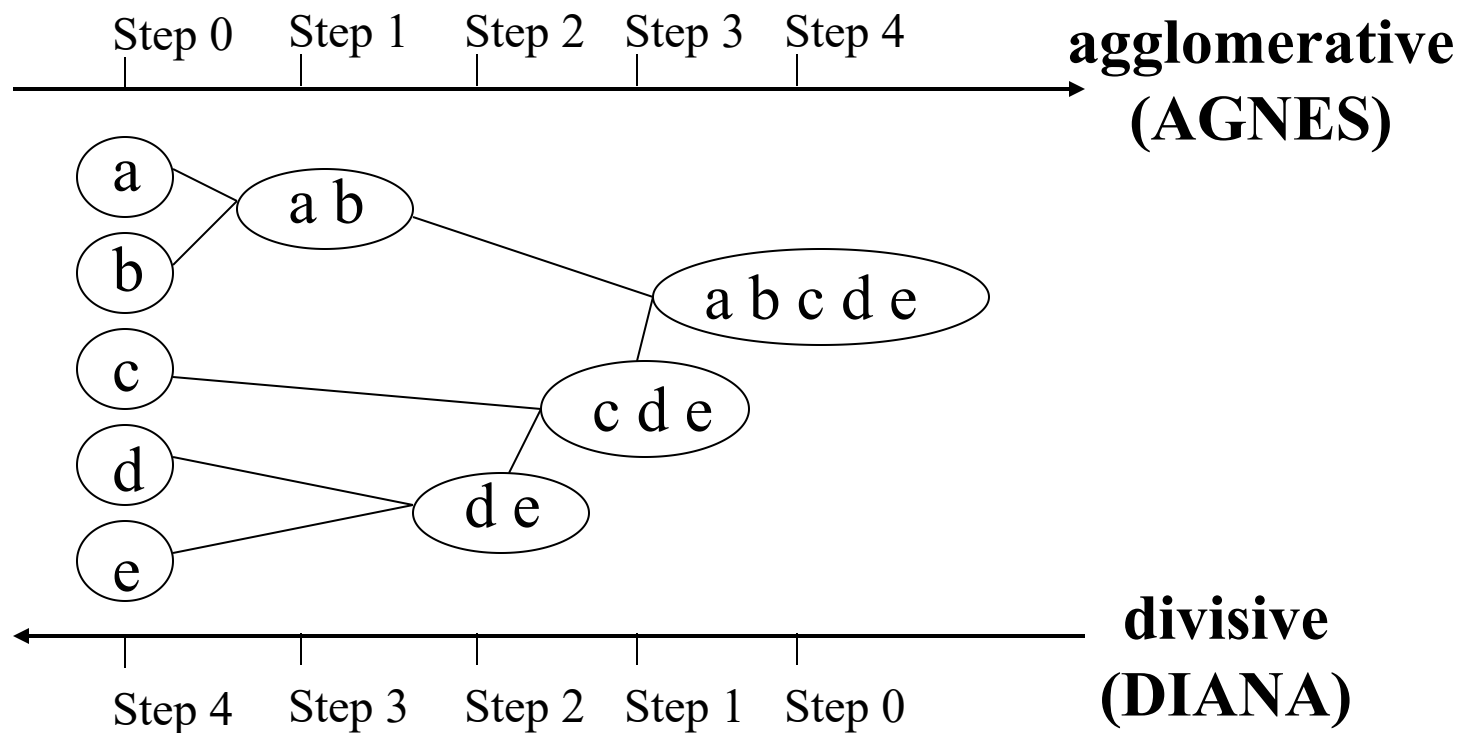


## 2.4 层次方法

- 对给定的数据集进行层次分解：
  - **自底向上方法**（合并）：开始时，将每个样本作为单独的一个组；然后，依次合并相近的样本或组，直至所有样本或组被合并为一个组或者达到终止条件为止。
    - 代表算法：**AGNES算法**
  - **自顶向下方法**（分裂）：开始时，将所有样本置于一个簇中；然后，执行迭代，在迭代的每一步中，一个簇被分裂为多个更小的簇，直至每个样本分别在一个单独的簇中或者达到终止条件为止。
    - 代表算法：**DIANA算法**

## 2.4 层次方法

### ■ 层次方法的示例：



## 2.4.1 AGNES算法

- AGNES (**A**gglomerative **N**esting)算法
  - 首先，将数据集中的每个样本作为一个簇；
  - 然后，根据**某些准则**将这些簇逐步合并；
  - 合并的过程反复进行，直至不能再合并或者达到结束条件为止。
- **合并准则**：每次找到**距离最近**的两个簇进行合并。
  - **两个簇之间的距离**由这两个簇中距离最近的样本点之间的距离来表示。

## 2.4.1 AGNES算法

### AGNES算法（自底向上合并算法）

输入：包含 $n$ 个样本的数据集，终止条件簇的数目 $k$ 。

输出： $k$ 个簇，达到终止条件规定的簇的数目。

(1) 初始时，将每个样本当成一个簇；

(2) REPEAT

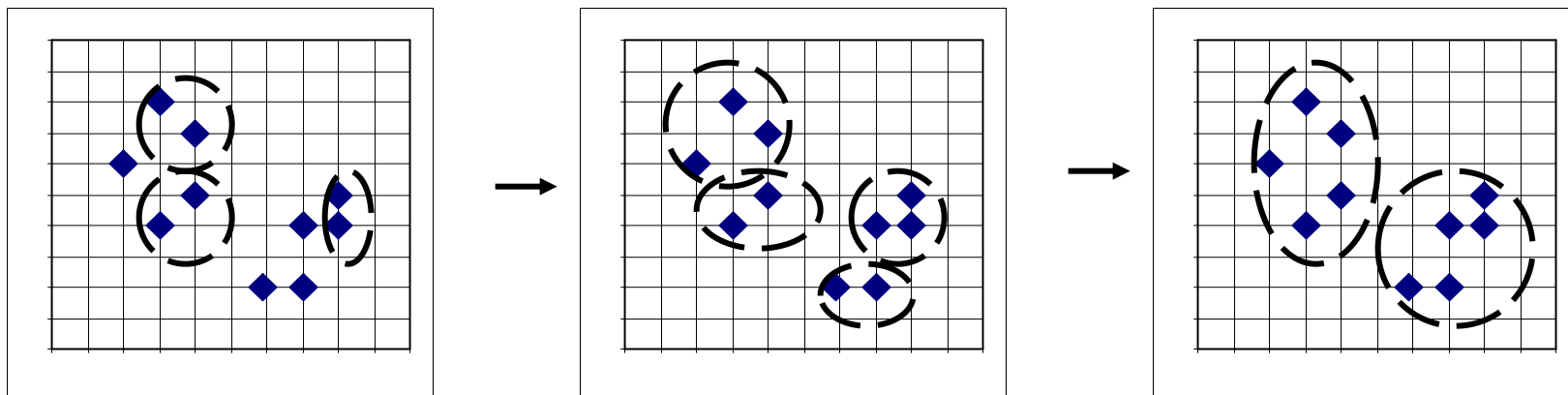
    根据不同簇中最近样本间的距离找到最近的两个簇；  
    合并这两个簇，生成新的簇的集合；

(3) UNTIL 达到定义的簇的数目。

## 2.4.1 AGNES算法

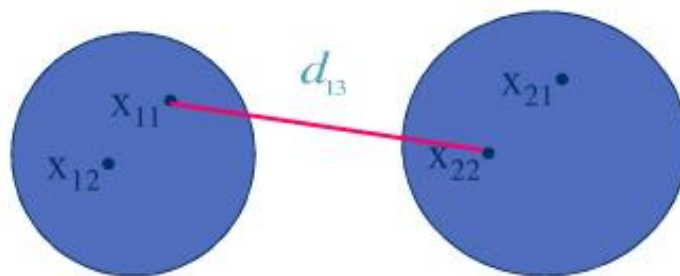
■ **AGNES (Agglomerative Nesting)**: 由 Kaufmann和 Rousseeuw提出(1990)

- ✓ 使用单链接(Single-Link)方法和相异度矩阵
- ✓ 合并具有最小相异度的节点
- ✓ 以非递减的方式继续
- ✓ 最终所有的节点属于同一个簇



## 2.4.1 AGNES算法

- 基本思路：单链接(Single-Link)方法和相异度矩阵。
  - 单链接方法用于确定任意两个簇之间的距离；



类 $G_p$ 与类 $G_q$ 之间的距离 $D_{pq}$  ( $d(x_i, x_j)$ 表示点 $x_i \in G_p$ 和 $x_j \in G_q$ 之间的距离)

$$D_{pq} = \min d(x_i, x_j)$$

## 2.4.1 AGNES算法

- 基本思路：单链接(Single-Link)方法和相异度矩阵。
  - 单链接方法用于确定任意两个簇之间的距离；
  - 相异度矩阵用于记录任意两个簇之间的距离（它是一个下三角矩阵，即：主对角线及其上方元素全部为零）。

表1 数据集

省份	x1	x2	x3	x4	x5	x6	x7	x8
辽宁	7.90	39.77	8.49	12.94	19.27	11.05	2.04	13.29
浙江	7.68	50.37	11.35	13.30	19.25	14.59	2.75	14.87
河南	9.42	27.93	8.20	8.14	16.17	9.42	1.55	9.76
甘肃	9.16	27.98	9.01	9.32	15.99	9.10	1.82	11.35
青海	10.06	28.64	10.52	10.05	16.18	8.39	1.96	10.81

## 2.4.1 AGNES算法

- 基本思路：单链接(Single-Link)方法和相异度矩阵。
  - 单链接方法用于确定任意两个簇之间的距离；
  - 相异度矩阵用于记录任意两个簇之间的距离（它是一个下三角矩阵，即：主对角线及其上方元素全部为零）。

5个样本之间的  
相异度矩阵



$$D_1 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{pmatrix} 0 & & & & \\ 11.67 & 0 & & & \\ 13.80 & 24.63 & 0 & & \\ 13.12 & 24.06 & 2.20 & 0 & \\ 12.80 & 23.54 & 3.51 & 2.21 & 0 \end{pmatrix} \end{matrix}$$



## 2.4.1 AGNES算法

- AGNES算法的优、缺点：
  - 算法简单，但有可能遇到合并点选择困难的情况；
  - 一旦不同的簇被合并，就不能被撤销；
  - 算法的时间复杂度为 $O(n^2)$ ，因此不适用处理 $n$ 很大的数据集。

## 2.4.2 DIANA算法

### ■ DIANA (**D**ivisive **A**nalysis)算法

- 在该种层次聚类算法中，也是以希望得到的簇的数目作为聚类的结束条件。
- 同时，使用下面两种测度方法：
  - 簇的直径：在一个簇中，任意两个样本间距离的最大值。
  - 平均相异度（平均距离）：

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} |x - y|$$

## 2.4.2 DIANA算法

### DIANA算法（自顶向下分裂算法）

输入：包含 $n$ 个样本的数据集，终止条件簇的数目 $k$ 。

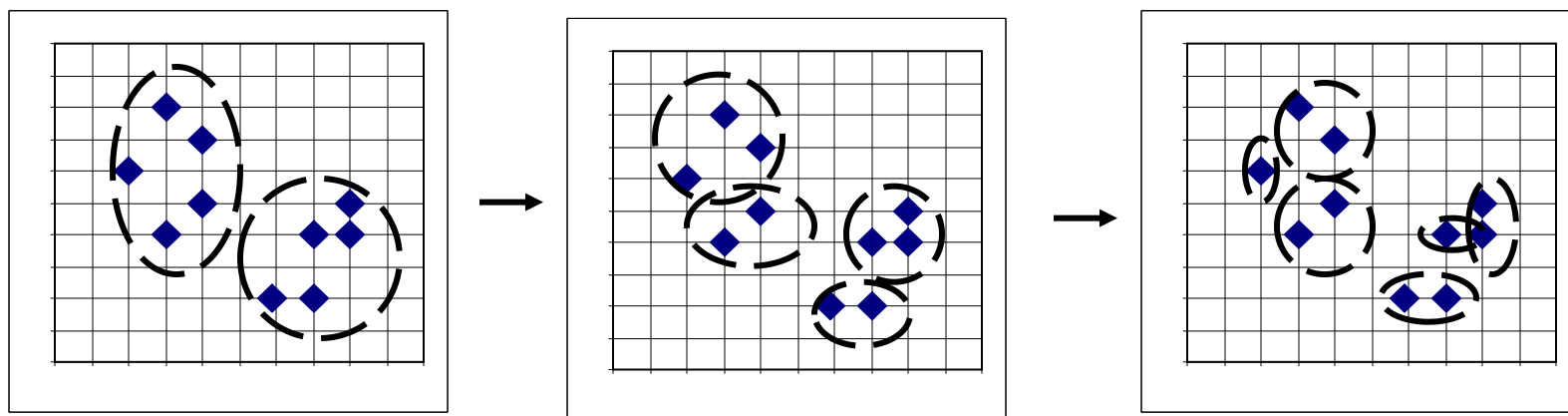
输出： $k$ 个簇，达到终止条件规定的簇的数目。

- (1) 初始时，将所有样本当成一个簇；
- (2) **FOR** ( $i=1$ ;  $i \neq k$ ;  $i++$ ) **DO BEGIN**
- (3)     在所有簇中挑出具有**最大直径**的簇 $C$ ；
- (4)     找出 $C$ 中与其它点**平均相异度**最大的一个点 $p$ ，并把 $p$ 放入splinter group，剩余的放在old party中；
- (5)     **REPEAT**
- (6)         在old party里找出到最近的splinter group中的点的距离不大于到old party中最近点的距离的点，并将该点加入splinter group。
- (7)     **UNTIL** 没有新的old party的点被分配给splinter group；
- (8)     splinter group和old party为被选中的簇分裂成的两个簇，与其它簇一起组成新的簇集合。
- (9) **END**

## 2.4.2 DIANA算法

■ **DIANA (Divisive Analysis):** 由 Kaufmann和Rousseeuw 提出(1990)

- ✓ 是 AGNES的逆
- ✓ 最终每个节点自己形成一个簇



## 2.4 层次方法

### ■ 层次聚类的主要缺点

- ✓ 不具有很好的可扩展性: 时间复杂性至少是  $O(n^2)$ , 其中  $n$  对象总数
- ✓ 合并或分裂的决定需要检查和估算大量的对象或簇
- ✓ 不能撤消已做的处理, 聚类之间不能交换对象. 如果某一步没有很好地选择合并或分裂的决定, 可能会导致低质量的聚类结果

## 2.4 层次方法

- **改进层次聚类质量的方法:** 将层次聚类和其他的聚类技术进行集成, 形成多阶段聚类
  - ✓ **BIRCH (1996):** 使用 CF-tree 对对象进行层次划分, 然后采用其他的聚类算法对聚类结果进行求精
  - ✓ **ROCK(1999):** 基于簇间的互联性进行合并
  - ✓ **CHAMELEON (1999):** 使用动态模型进行层次聚类
  - ✓ **CURE (1998):** 采用固定数目的代表对象来表示每个簇, 然后依据一个指定的收缩因子向着聚类中心对它们进行收缩

# 总结

- **Cluster analysis** groups objects based on their **similarity** and has wide applications
- Measure of similarity can be computed for **various types of data**
- Clustering algorithms can be **categorized** into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- **K-means and K-medoids** algorithms are popular partitioning-based clustering algorithms
- **AGNES** and **DIANA** are interesting hierarchical clustering algorithms
- Quality of clustering results can be evaluated in various ways