聚类分析
Cluster Analysis

无监督学习

监督学习和无监督学习的区别

监督学习

在一个典型的监督学习中，训练集有标签y，我们的目标是找到能够区分正样本和负样本的决策边界，需要据此拟合一个假设函数。

无监督学习

与此不同的是，在无监督学习中，我们的数据没有附带任何标签y，无监督学习主要分为聚类、降维、关联规则、推荐系统等方面。

聚类分析

- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

1 聚类分析概述

- **聚类分析**的定义
  - **聚类分析**(or ***clustering***, ***data segmentation***, …)是一个将数据集的所有数据，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。
  - **聚类分析**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：聚类分析之后，应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**。

1 聚类分析概述

- 聚类分析在数据挖掘中的作用：
  - 作为一个独立的工具来获得数据集中数据的分布情况；
  - 作为其他数据挖掘算法的预处理步骤。

1 聚类分析概述

- 聚类分析在数据挖掘中的作用：
  - 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。
  - 作为其他数据挖掘算法的预处理步骤。

1 聚类分析概述

- 聚类分析在数据挖掘中的作用：
  - 作为一个独立的工具来获得数据集中数据的分布情况；
  - 作为其他数据挖掘算法的预处理步骤。
    - 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。

1 聚类分析概述：要求和挑战

- 可扩展性(Scalability)
  - 大多数来自于机器学习和统计学领域的聚类算法在处理数百条数据时能表现出高效率。Clustering all the data instead of only on samples。
- 处理不同数据类型的能力
  - 数字型；二元类型，分类型/标称型，序数型,比例标度型等等
- 发现任意形状的能力
  - 基于距离的聚类算法往往发现的是球形的聚类，其实现的聚类是任意形状的
- 用于决定输入参数的领域知识最小化
  - 对于高维数据，参数很难决定，聚类的质量也很难控制
- 处理噪声数据的能力
  - 对空缺值、孤立点、数据噪声不敏感

1 聚类分析概述：要求和挑战

- 对于输入数据的顺序不敏感
  - 同一个数据集合，以不同的次序提交给同一个算法，应该产生相似的结果
- 高维度
  - 高维度的数据可能比较稀疏，而且高度倾斜
- 基于约束的聚类
  - 找到既满足约束条件，又具有良好聚类特性的数据分组
- 可解释性和可用性
  - 聚类要与特定的语义解释和应用相联系

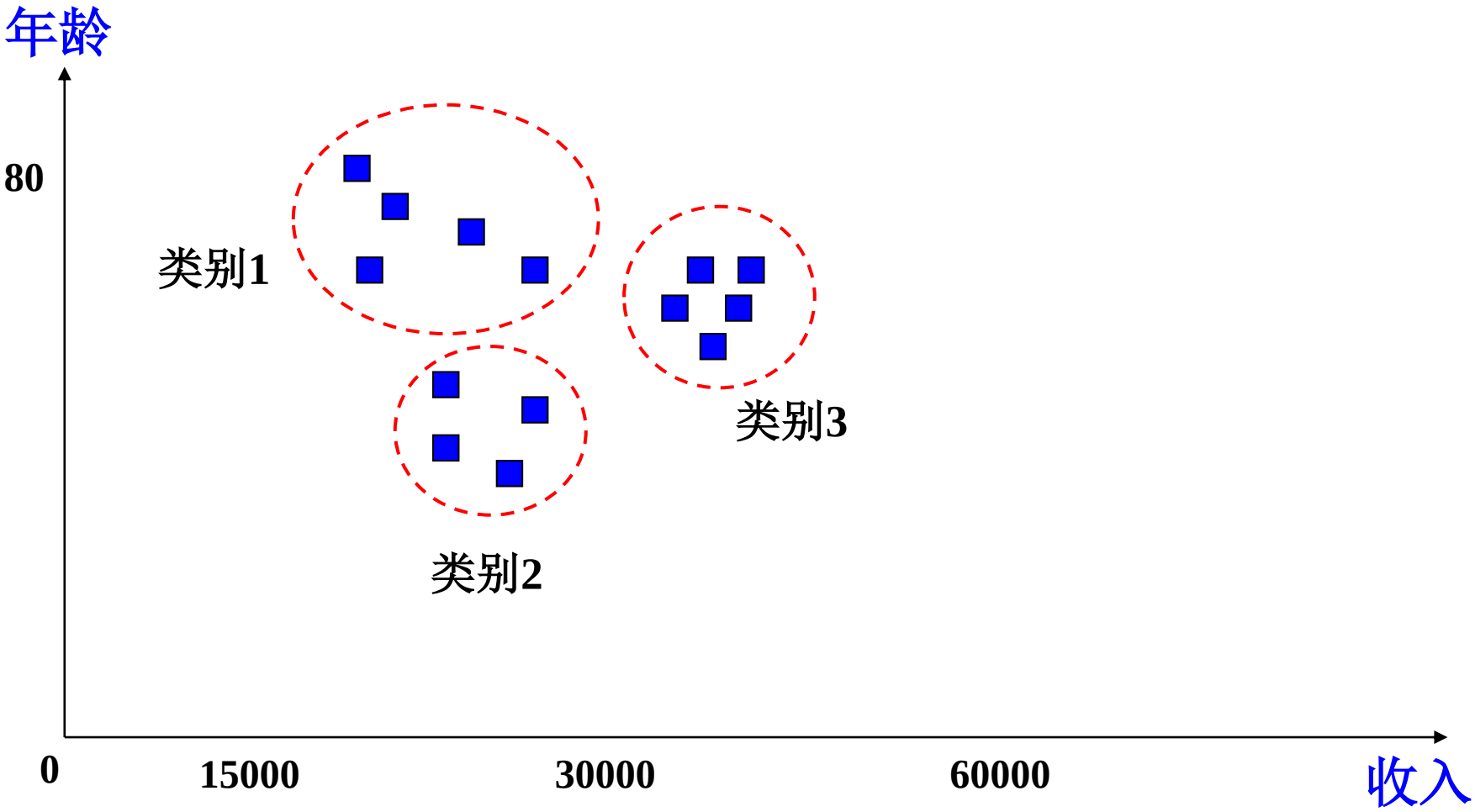
1 聚类分析概述

- 聚类分析的典型应用
  - 市场细分、文档聚类、图像分割、图像压缩、聚类分析、特征学习或者词典学习、确定犯罪易发地区、保险欺诈检测、公共交通数据分析、IT 资产集群、客户细分、识别癌症数据、搜索引擎应用、医疗应用、药物活性预测
  - 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）。
  - ……

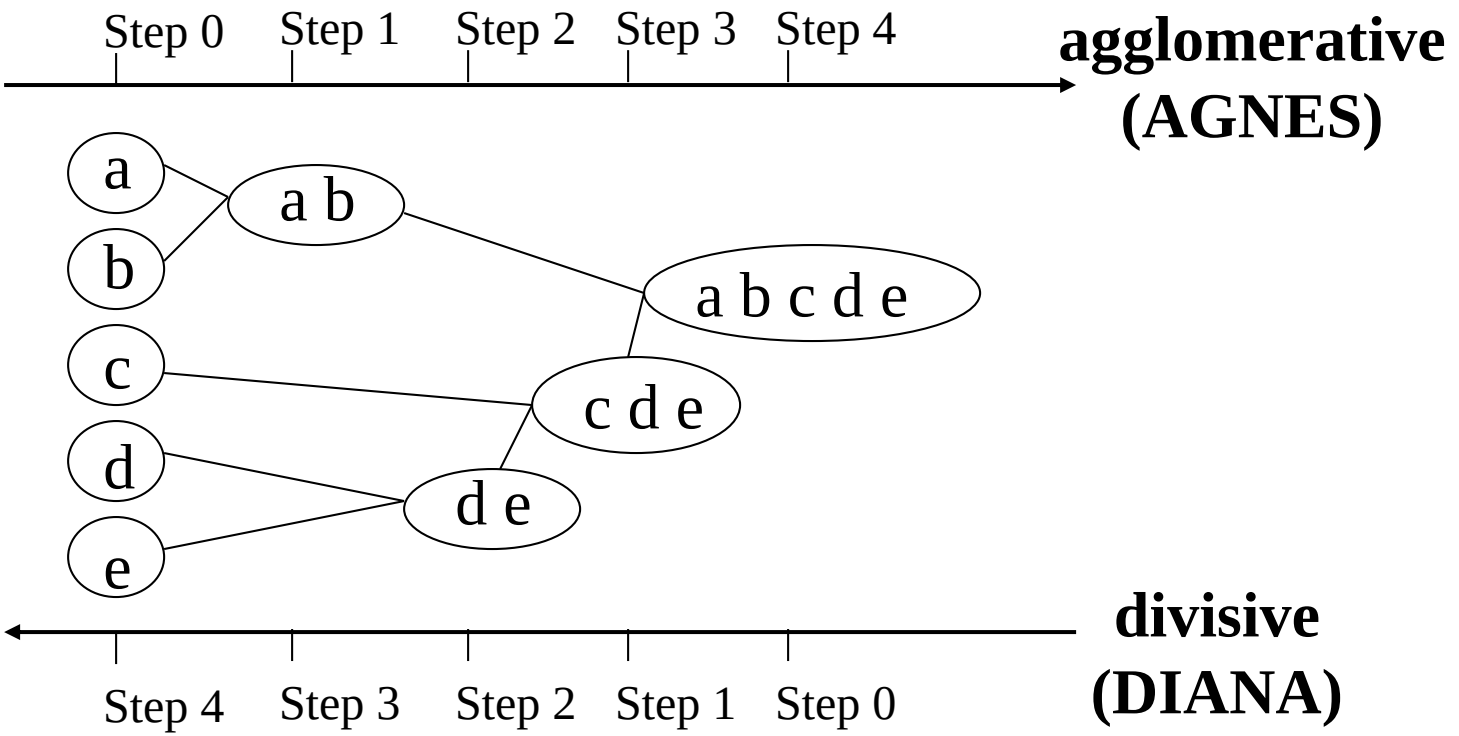
1 聚类分析概述

- 常用的聚类分析方法：
  - **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。
    - 经典方法有：*k*-means及其变体k-means++、bi-kmeans、kernel k-means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。
    - 聚类方法有：自顶向下法、自底向上法。
  - **基于密度的方法**（Density-Based Methods）：假定类别可以通过样本分布的紧密程度决定，同一类别的样本，他们之间是紧密相连的，也就是说，在该类别任意样本周围不远处一定有同类别的样本存在
    - 属于这样的聚类方法有：DBSCAN等。

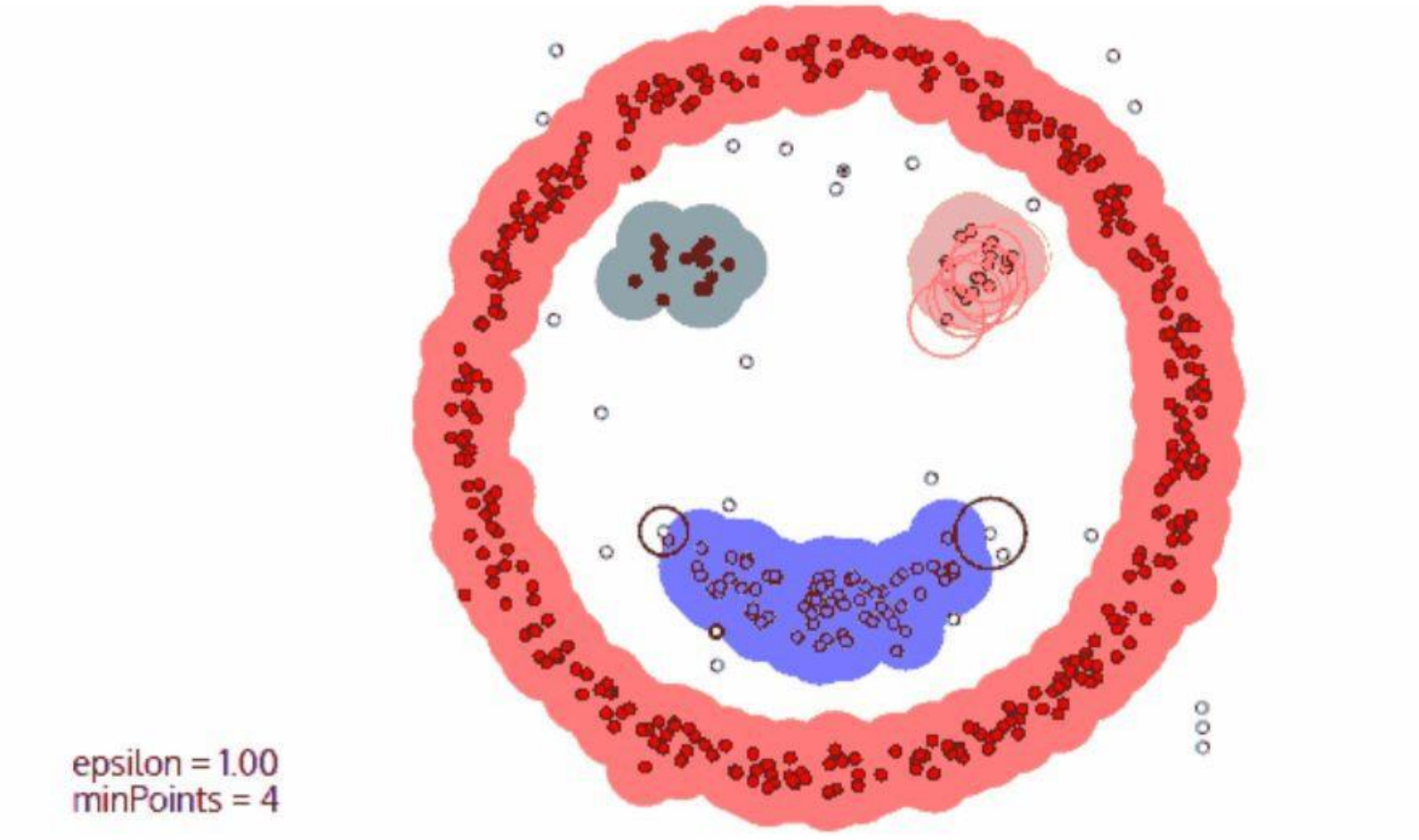
1 聚类分析概述：划分法示例



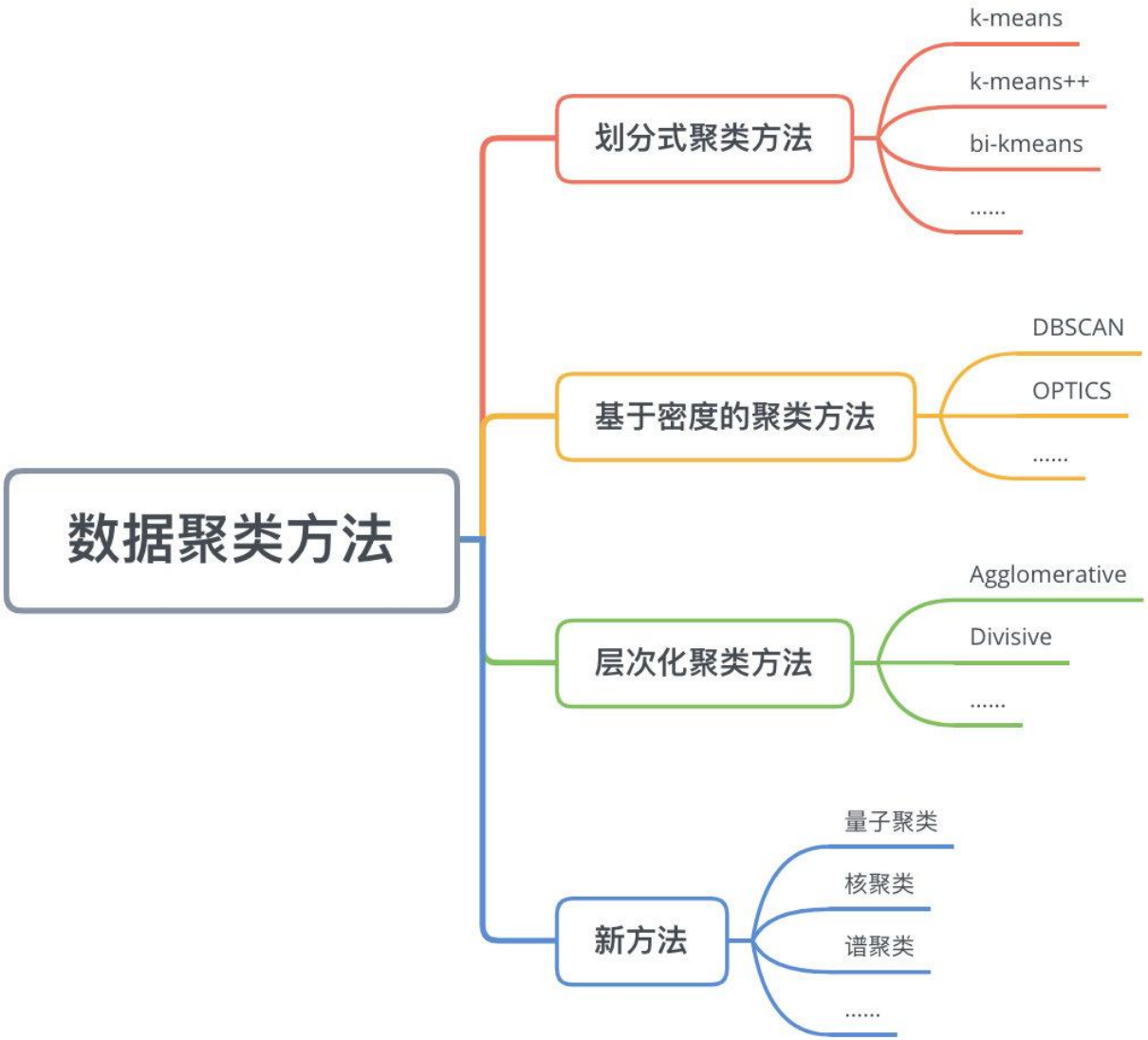
1 聚类分析概述：层次法示例



1 聚类分析概述：密度法示例



1 聚类分析概述：层次法示例



聚类分析

- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

2 相似性计算方法

- 在聚类分析中，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 相似性计算方法

- 在聚类分析中，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）上进行计算的。
  - 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)

2 相似性计算方法

- 2.1 连续型属性的相似性计算方法
- 2.2 二值离散型属性的相似性计算方法
- 2.3 多值离散型属性的相似性计算方法
- 2.4 混合类型属性的相似性计算方法

2 相似性计算方法

- 2.1 连续型属性的相似性计算方法
- 2.2 二值离散型属性的相似性计算方法
- 2.3 多值离散型属性的相似性计算方法
- 2.4 混合类型属性的相似性计算方法

2.1 连续型属性的相似性计算方法

- 假设两个样本$\boldsymbol{X_i}$和$\boldsymbol{X_j}$分别表示成如下形式：
  - $X_i=(x_{i1}, x_{i2}, \ldots, x_{id})$
  - $X_j=(x_{j1}, x_{j2}, \ldots, x_{jd})$
  - 它们都是$\boldsymbol{d}$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。

2.1 连续型属性的相似性计算方法

- 欧氏距离（Euclidean distance）

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

q=2

- 曼哈顿距离（Manhattan distance）

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

q=1

- 闵可夫斯基距离（Minkowski distance）

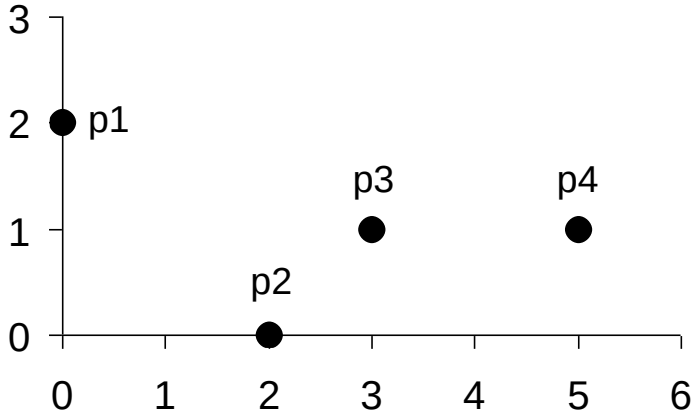
$$d(x_i, x_j) = \left(\sum_{k=1}^{d} \left| x_{ik} - x_{jk} \right|^q \right)^{1/q}$$

2.1 连续型属性的相似性计算方法

- 距离函数的性质
  - $d(i, j) \geq 0$
  - $d(i, i) = 0$
  - $d(i, j) = d(j, i)$
  - $d(i, j) \leq d(i, k) + d(k, j)$
- *Minkowski*距离是*Euclidean*距离和*Manhattan*距离的推广
  - 如果$q = 1$则表示*Manhattan*距离，如果$q = 2$则表示*Euclidean*距离

2.1 连续型属性的相似性计算方法

- 欧式距离的示例



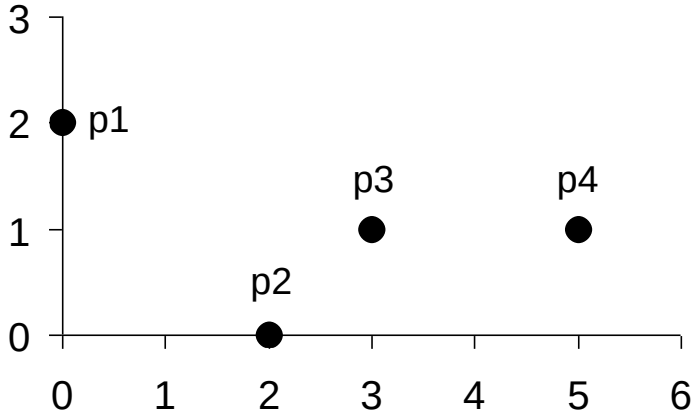
point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	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

2.1 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 相似性计算方法

- 2.1 连续型属性的相似性计算方法
- 2.2 二值离散型属性的相似性计算方法
- 2.3 多值离散型属性的相似性计算方法
- 2.4 混合类型属性的相似性计算方法

2.2 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：描述病人的是否抽烟的属性(*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$\boldsymbol{X_i}$和$\boldsymbol{X_j}$分别表示成如下形式：
  - $X_i=(x_{i1}, x_{i2}, \ldots, x_{ip})$
  - $X_j=(x_{j1}, x_{j2}, \ldots, x_{jp})$
  - 它们都是$\boldsymbol{p}$维的特征向量，并且每维特征都是一个二值离散型数值。

2.2 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	***sum***
X_i	1	***a***	***b***	***a+b***
	0	***c***	***d***	***c+d***
	sum	***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.

2.2 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。

2.2 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。

2.2 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]和q= [0 0 0 0 0 0 1 0 0 1]
  - a = 0 (the number of attributes where p was 1 and q was 1)
  - b = 1 (the number of attributes where p was 1 and q was 0)
  - c = 2 (the number of attributes where p was 0 and q was 1)
  - d = 7 (the number of attributes where p was 0 and q was 0)

2.2 二值离散型属性的相似性计算方法

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

  $= (1+2) / (0+1+2+7)$

  $= 0.3$

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

  $= (1+2) / (0+1+2)$

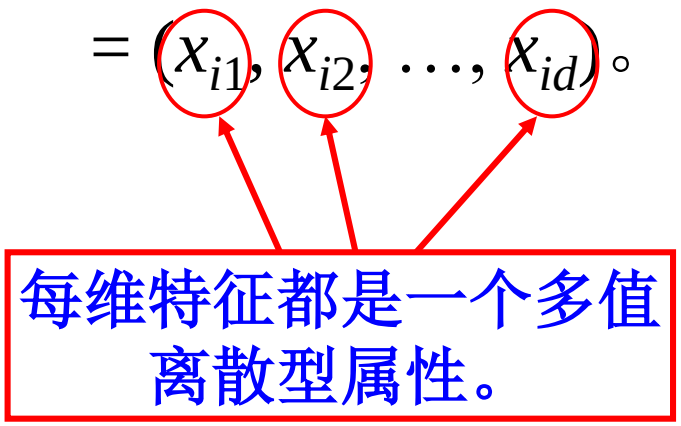
  $= 1$

2 相似性计算方法

- 2.1 连续型属性的相似性计算方法
- 2.2 二值离散型属性的相似性计算方法
- 2.3 多值离散型属性的相似性计算方法
- 2.4 混合类型属性的相似性计算方法

2.3 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$X=\{x_i \mid i=1,2,\ldots,total\}$。
  - 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：$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$。



2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，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。

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，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。

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，如何计算它们之间的距离？**
  - 方法一：简单匹配方法。

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

  - $d(x_1, x_2) = (3\text{-}1)/3 \approx 0.667$

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，如何计算它们之间的距离？**
  - 方法一：简单匹配方法。

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

  - $d(x_1, x_3) = (3-0)/3 = 1$

2.3 多值离散型属性的相似性计算方法

- 问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，如何计算它们之间的距离？
  - 方法一：简单匹配方法。

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

  - $d(x_1, x_4) = (3\text{-}2)/3 \approx 0.333$

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，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。

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，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

2.3 多值离散型属性的相似性计算方法

- **问题：给定两个样本$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，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

2 相似性计算方法

- 2.1 连续型属性的相似性计算方法
- 2.2 二值离散型属性的相似性计算方法
- 2.3 多值离散型属性的相似性计算方法
- 2.4 混合类型属性的相似性计算方法

2.4 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

2.4 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，进行一次聚类分析。
  - 假设：给定的数据集$X=\{x_i \mid i=1,2,\ldots,total\}$，每个样本用$\boldsymbol{d}$个描述属性$\boldsymbol{A_1}, \boldsymbol{A_2}, \ldots, \boldsymbol{A_d}$来表示，属性$\boldsymbol{A_j}(1\le j\le d)$包含多种类型。

2.4 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，进行一次聚类分析。
  - 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。

2.4 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$x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$和$x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，它们之间的距离为：

$$d(x_i, x_j) = \frac{\sum_{k=1}^{d} \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^{d} \delta_{ij}^{(k)}}$$

  - 其中：$d_{ij}^{(k)}$表示x_i和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x_i和x_j距离的影 响。

2.4 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x_i和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$d_{ij}^{(k)}$：

$$d_{ij}^{(k)} = | x_{ik} - x_{jk} |$$

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

2.4 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：$\delta_{ij}^{(k)}=1$ 。

聚类分析

- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

3.1 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*≤*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。

k-means算法

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

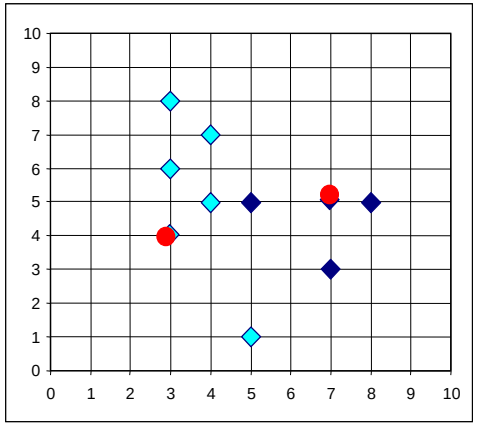
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

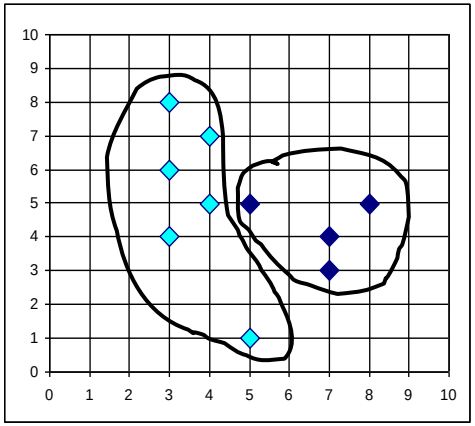
k-means算法：示例



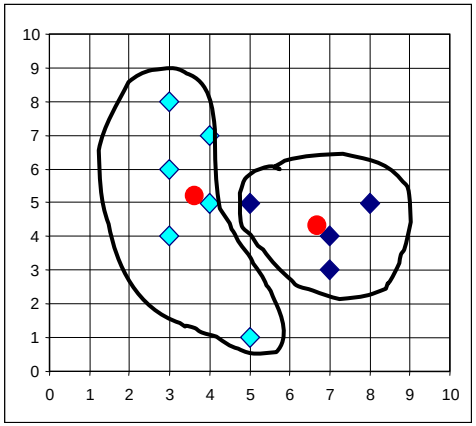
Arbitrarily choose *k* object as initial cluster center

k = 2

Assign each objects to most similar center

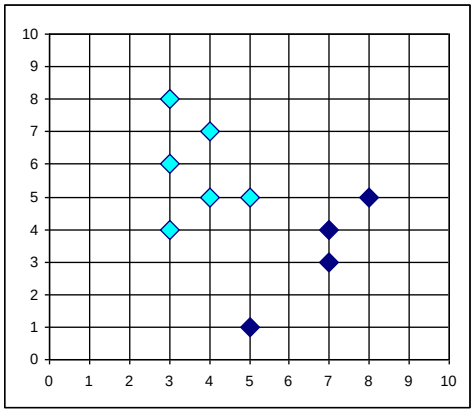


Update the cluster means



Update the cluster means

reassign



实例

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

给定数据集合，算法*k*-means (*n*=8，*k*=2)的主要执行步骤：

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）；

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）、（3.5，3）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）

实例

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

- 假设对于相同的样本数据，若随机选择的两个初始点为序号4和7。
- 问题1：结果与前面会一样吗？
- 问题2：有何结论?

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。

k是簇的个数　p是簇C_i中的样本　m_i是簇C_i的均值

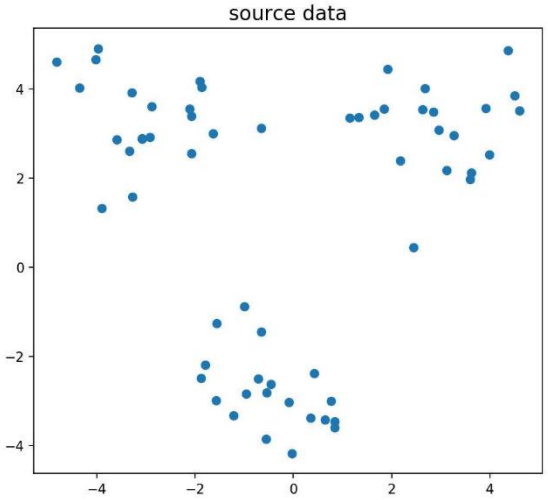
- 对于同一个数据集，由于*k*-means算法对初始选取的聚类中心敏感，因此可用该准则评价聚类结果的优劣。
- 通常，对于任意一个数据集，*k*-means算法无法达到全局最优，只能达到局部最优。（损失函数是非凸优化函数，会收敛于局部最优解）

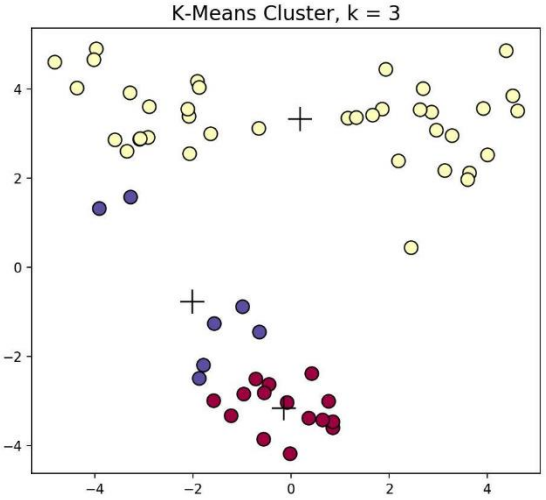
k-means算法

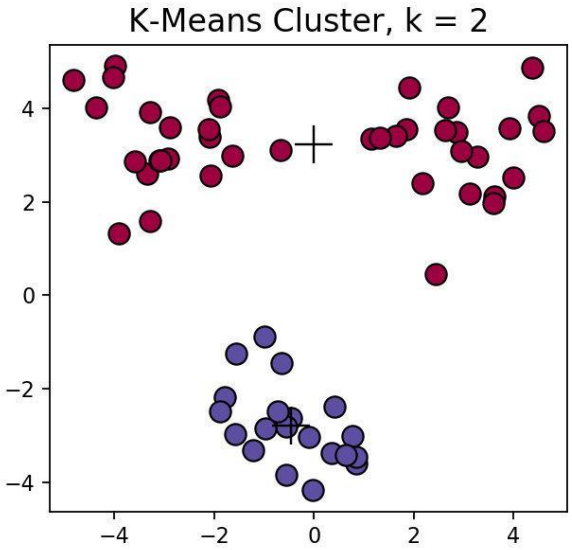
- 优点：
  - 原理比较简单，实现也是很容易，收敛速度快。
  - 聚类效果较优
  - 可扩展性较好，算法复杂度为$O(nkt)$。
    - 其中：n为样本个数，k是簇的个数，t是迭代次数。
- 缺点：
  - 簇数目k需要事先给定，但非常难以选定；
  - 初始聚类中心的选择对聚类结果有较大的影响；
  - 不适合高度重叠的数据、非线性数据集；
  - 对噪声和离群点数据敏感。
  - 不适用于过大的数据集

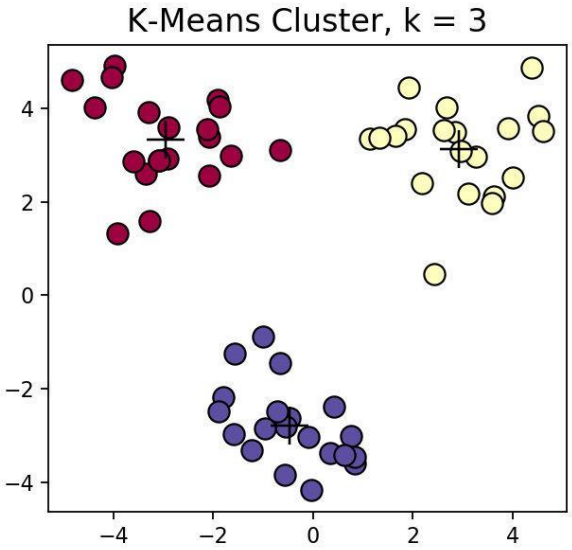
k-means算法

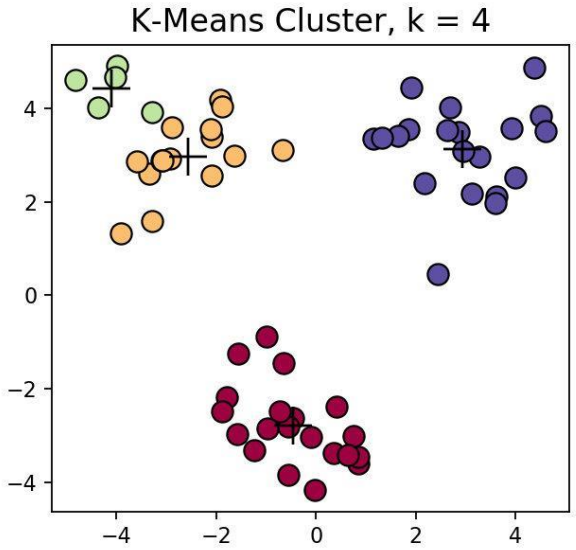
由于初始质心点选取的不合理造成的误分







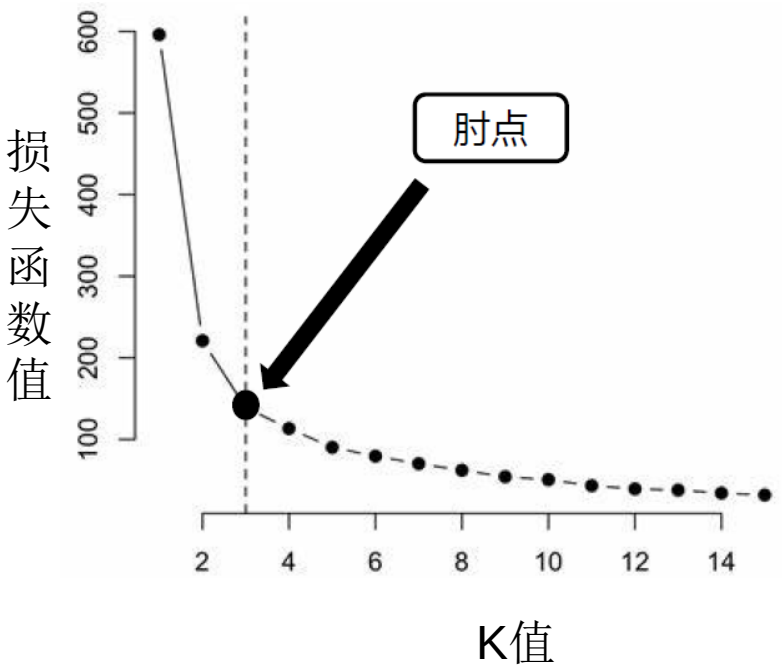




K值选取不合适造成的影响

k-means算法

- 肘部法则：
  - 通常通过“肘部法则”确定最佳的k值。我们可能会得到一条类似于人的肘部的曲线。右图中，代价函数的值会迅速下降，在K=3的时候达到一个肘点。在此之后，代价函数的值会就下降得非常慢，所以，我们选择K=33。这个方法叫 肘部法则。
  - 为避免局部最优，我们通常需要多次运行K 均值算法，每一次都重新进行随机初始化，比较多次运行K 均值的结果，选择代价函数最小的结果



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。
  - 可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。

首先随机选择k个对象作为中心，把每个对象分配给离它最近的中心。
然后**随机地**选择一个非中心对象替换中心对象，计算分配后的距离改进量
如果总的损失减少，则交换中心对象和非中心对象；
如果总的损失增加，则不进行交换

k-medoids算法

Algorithm: *k*-**medoids.** PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

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}？

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算法的鲁棒性更好(稳定)。
  - 因为中心点不像均值那样易被极端数据(噪声或者离群点)扭曲。
- *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*。

CLARA 算法

- *CLARA* 算法(Clustering Large Applications，Kaufmann & Rousseeuw) (1990)
- 该算法首先获得数据集的多个采样，然后在每个采样上使用PAM算法，最后返回最好的聚类结果作为输出。
- 优点: 能够处理大数据集。
- 缺点
  - 效率依赖于采样的大小；
  - 如果样本发生偏斜，基于样本的一个好的聚类不一定代表得了整个数据集合的一个好的聚类。

Review

- 聚类的目标和要求
- 相似性的度量方式
- K-means算法核心思想
- 如何确定合适的K
- K-means算法优缺点
- *k*-medoids算法思想

聚类分析

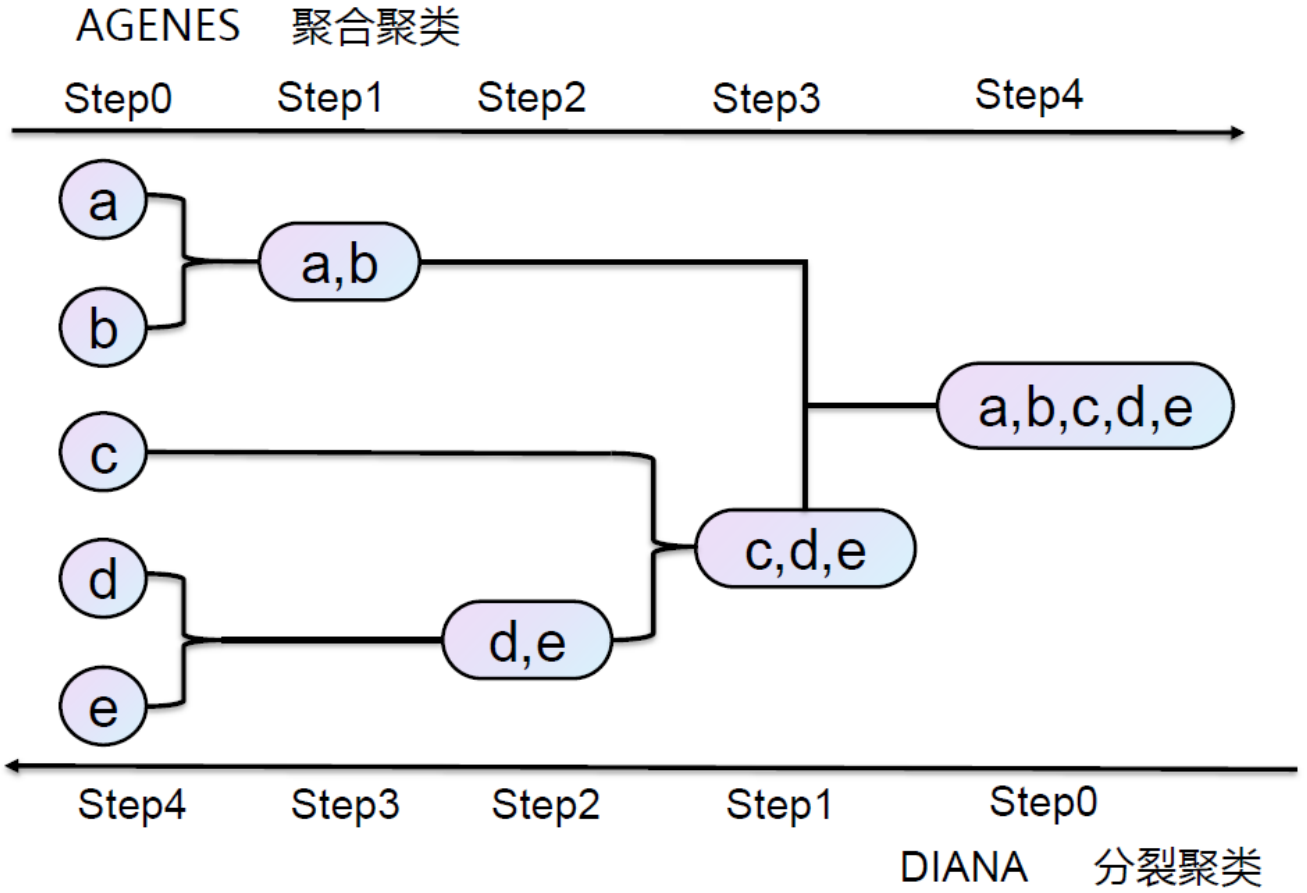
- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

3.2 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**

3.2 层次方法

- 层次方法的示例：



层次聚类算法是一种贪心算法（greedy algorithm），因其每一次合并或划分都是基于某种局部最优的选择

3.2 层次方法

- cluster之间的相似度度量：
  - 除了需要衡量对象之间的距离之外，层次聚类还需要衡量cluster之间的距离。假设C_i和C_j为两个 cluster，则四种常用的C_i和C_j之间的距离如下表所示。

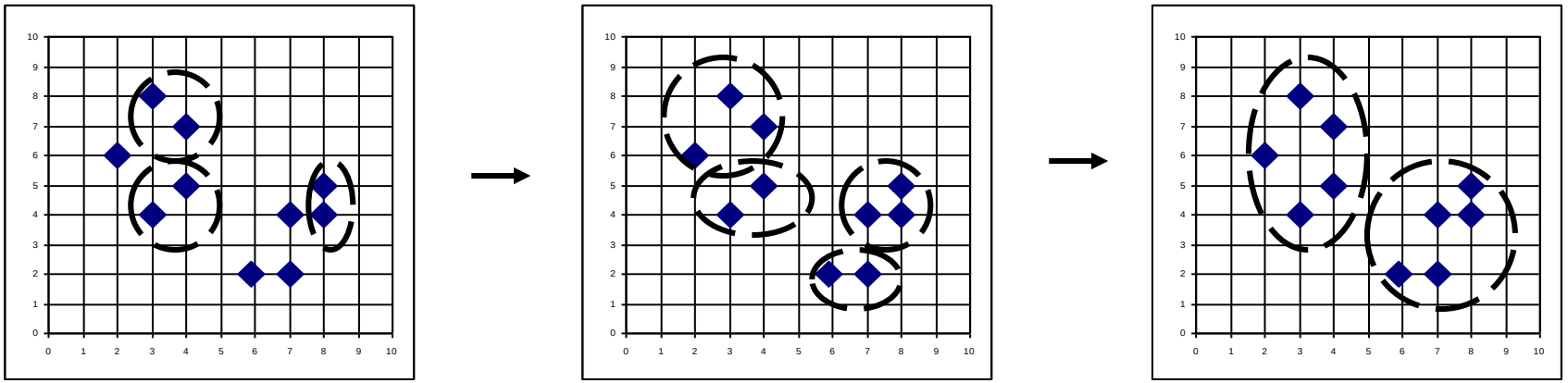
相似度度量准则	相似度度量函数
Single-link	$D(C_i, C_j) = \min_{x \subseteq C_i, y \subseteq C_j} d(x, y)$
Complete-link	$D(C_i, C_j) = \max_{x \subseteq C_i, y \subseteq C_j} d(x, y)$
UPGMA	$D(C_i, C_j) = \frac{1}{\|C_i\| \|C_j\|} \sum_{x \subseteq C_i, y \subseteq C_j} d(x, y)$
WPGMA	-

3.2 层次方法

- cluster之间的相似度度量：
  - Single-link定义两个cluster之间的距离为两个cluster之间距离最近的两个点之间的距离，这种方法会在聚类的过程中产生链式效应，即有可能会出现非常大的cluster
  - Complete-link定义的是两个cluster之间的距离为两个cluster之间距离最远的两个点之间的距离，这种方法可以避免链式效应,对异常样本点却非常敏感，容易产生不合理的聚类
  - UPGMA正好是Single-link和Complete-link方法的折中，他定义两个cluster之间的距离为两个cluster之间所有点距离的平均值
  - 最后一种WPGMA方法计算的是两个 cluster 之间两个对象之间的距离的加权平均值，加权的目的是为了使两个 cluster 对距离的计算的影响在同一层次上，而不受 cluster 大小的影响，具体公式和采用的权重方案有关。

AGNES算法

- 凝聚：最初将每个对象作为一个簇，然后这些簇根据某些准则被逐步合并。两个簇间的相似度由这两个不同簇中距离最近的数据点对的相似度来确定。聚类的合并过程反复进行直到所有对象最终属于同一个簇或达到一个终止条件。
  - Use the **single-link** method and the dissimilarity matrix
  - Merge nodes that have the least dissimilarity
  - Go on in a non-descending fashion
  - Eventually all nodes belong to the same cluster



AGNES算法

- AGNES (Agglomerative Nesting)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。

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

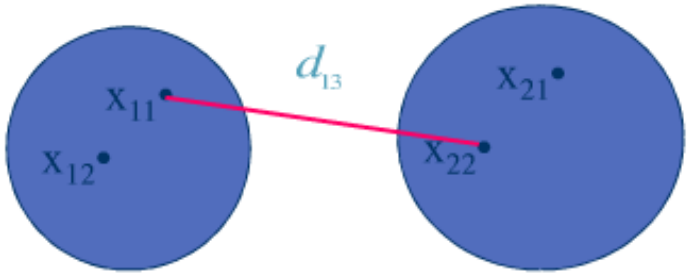
根据不同簇中最近样本间的距离找到最近的两个簇；

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。

AGNES算法

- 在这个算法中，需要使用**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）

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

AGNES算法

- 在这个算法中，需要使用**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

AGNES算法

- 在这个算法中，需要使用**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 →

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

AGNES算法：示例

- 例：为了研究辽宁省等五省区某年度城镇居民生活消费的分布规律，对如下调查数据进行聚类。

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

AGNES算法：示例

G1={辽宁}，G2={浙江}，G3={河南}，G4={甘肃}，G5={青海}

采用欧氏距离：

$d_{12} = 11.67$

$d_{13}=13.80$ $d_{14}=13.12$ $d_{15}=12.80$ $d_{23}=24.63$ $d_{24}=24.06$ $d_{25}=23.54$ $d_{34}=2.2$
$d_{35}=3.51$ $d_{45}=2.21$

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

河南与甘肃的距离最近，先将二者（3和4）合为一类G6={G3, G4}

AGNES算法：示例

G1={辽宁}，G2={浙江}，G3={河南}，G4={甘肃}，G5={青海}

采用欧氏距离：

$d_{61}=d_{(3,4)1}=\min\{d_{13},d_{14}\}=13.12$　$d_{62}=d_{(3,4)2}=\min\{d_{23},d_{24}\}=24.06$

$d_{65}=d_{(3,4)5}=\min\{d_{35},d_{45}\}=2.21$

$$D_2=\begin{array}{c} \\ 6 \\ 1 \\ 2 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccc} 6 & 1 & 2 & 5 \end{array} \\ \begin{pmatrix} 0 & & & \\ 13.12 & 0 & & \\ 24.06 & 11.67 & 0 & \\ 2.21 & 12.80 & 23.54 & 0 \end{pmatrix} \end{array}$$

河南、甘肃与青海并为一新类

G7={G6，G5}={G3，G4，G5}

AGNES算法：示例

G1={辽宁}，G2={浙江}，G3={河南}，G4={甘肃}，G5={青海}

采用欧氏距离：

$d_{71}=d_{(3,4,5)1}=\min\{d_{13},d_{14},d_{15}\}=12.80$

$d_{72}=d_{(3,4,5)2}=\min\{d_{23},d_{24},d_{25}\}=23.54$

$$D_3=\begin{array}{c} \\ 7 \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccc} 7 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & & \\ 12.80 & 0 & \\ 23.54 & 11.67 & 0 \end{pmatrix} \end{array}$$

G8={G1,G2}

AGNES算法：示例

G1={辽宁}，G2={浙江}，G3={河南}，G4={甘肃}，G5={青海}

采用欧氏距离：

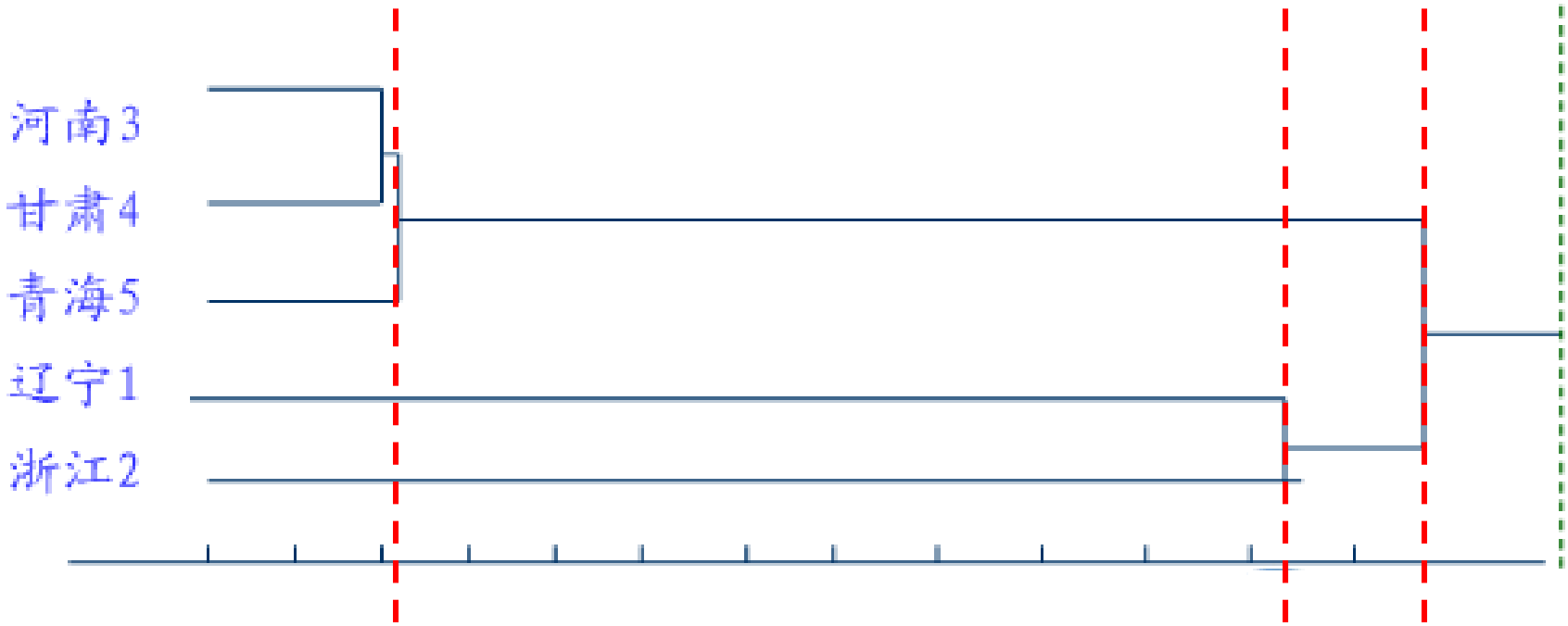
$d_{78}=\min\{d_{71},d_{72}\}=12.80$

$$D_4=\begin{array}{c} \\ 7 \\ 8 \end{array}\begin{array}{c} \begin{array}{cc} 7 & 8 \end{array} \\ \begin{bmatrix} 0 & \\ 12.8 & 0 \end{bmatrix} \end{array}$$

→ G9={G7, G8}

AGNES算法：示例

G1={辽宁}，G2={浙江}，G3={河南}，G4={甘肃}，G5={青海}



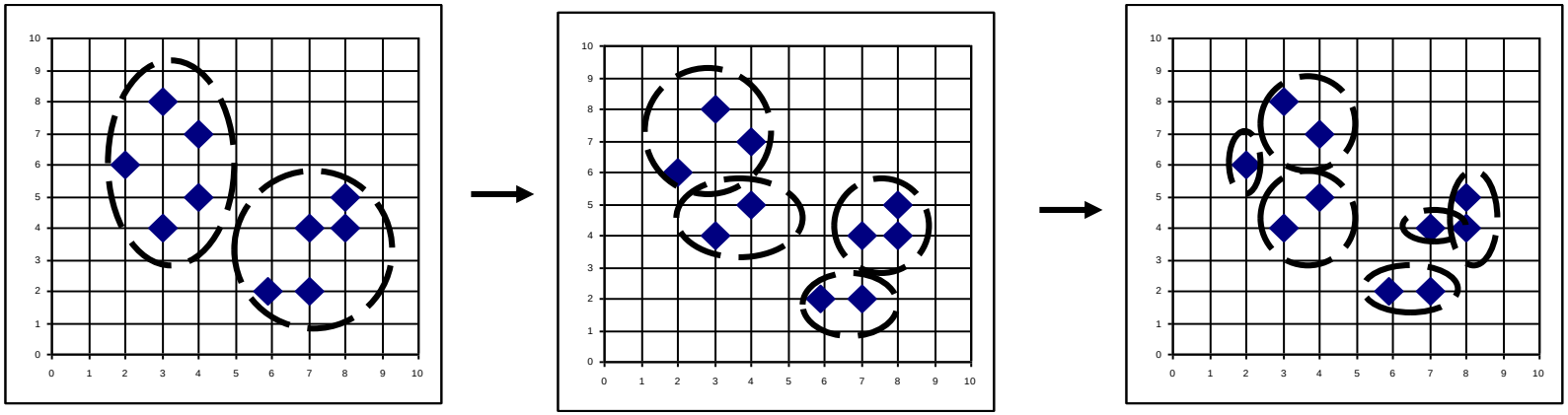
K=4　　K=3　　K=2

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。

DIANA算法

- 分裂：开始将所有的对象置于一个簇中，在迭代的每一步，一个簇被分裂为多个更小的簇，直到最终每个对象在一个单独的簇中，或达到一个终止条件
- AGNES算法的逆过程



DIANA算法

- **DIANA (Divisive Analysis)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）：

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

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*≠*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

DIANA算法：示例

- 例：有如下表所示的数据集，使用**DIANA算法**对该数据集进行分裂层次聚类。

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

DIANA算法：示例

- 第1步，首先找到具有最大直径的簇，然后计算该簇中每个样本的平均相异度（假定采用是欧式距离）。

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

$$\begin{bmatrix} 0 & & & & & & & \\ 1 & 0 & & & & & & \\ 1 & 1.4 & 0 & & & & & \\ 1.4 & 1 & 1 & 0 & & & & \\ 3.6 & 2.8 & 3.2 & 2.2 & 0 & & & \\ 4.5 & 3.6 & 4.1 & 3.2 & 1 & 0 & & \\ 4.2 & 3.6 & 3.6 & 2.8 & 1 & 1.4 & 0 & \\ 5 & 4.2 & 4.5 & 3.6 & 1.4 & 1 & 1 & 0 \end{bmatrix}$$

DIANA算法：示例

- 第1步，首先找到具有最大直径的簇，然后计算该簇中每个样本的平均相异度（假定采用是欧式距离）。
  - 样本1的平均距离为：(1+1+1.4+3.6+4.5+4.2+5)/7=**2.96**
  - 样本2的平均距离为：2.53
  - 样本3的平均距离为：2.68
  - 样本4的平均距离为：2.18
  - 样本5的平均距离为：2.18
  - 样本6的平均距离为：2.68
  - 样本7的平均距离为：2.53
  - 样本8的平均距离为：**2.96**
  - 从上述值中挑出具有最大平均相异度的样本1(或者样本8)，将其放到splinter group中，剩余点在old party中。

DIANA算法：示例

- 第2步，在old party中，找出到splinter group距离最近的样本，并且该样本到splinter group的距离不大于该样本与old party中其他样本之间的最小距离，则将该样本放入splinter group中；此时找到满足上述条件的是**样本2**。

$$\begin{bmatrix}
0 & & & & & & & \\
1 & 0 & & & & & & \\
1 & 1.4 & 0 & & & & & \\
1.4 & 1 & 1 & 0 & & & & \\
3.6 & 2.8 & 3.2 & 2.2 & 0 & & & \\
4.5 & 3.6 & 4.1 & 3.2 & 1 & 0 & & \\
4.2 & 3.6 & 3.6 & 2.8 & 1 & 1.4 & 0 & \\
5 & 4.2 & 4.5 & 3.6 & 1.4 & 1 & 1 & 0
\end{bmatrix}$$

DIANA算法：示例

- 第2步，在old party中，找出到splinter group距离最近的样本，并且该样本到splinter group的距离不大于该样本与old party中其他样本之间的最小距离，则将该样本放入splinter group中；此时找到满足上述条件的是**样本2**。
- 第3步，重复第2步，splinter group中放入**样本3**。
- 第4步，重复第2步，splinter group中放入**样本4**。
- 第5步，在old party中，已找不到可以放入splinter group中的样本，并且此时达到算法的终止条件（k=2），算法结束。（如果没有达到算法的终止条件，应该从分裂出来的簇中再挑选一个具有直径最大的簇继续分裂。）

DIANA算法：示例

步骤	具有最大直径的簇	splinter group	Old party
1	{1，2，3，4，5，6，7，8}	{1}	{2，3，4，5，6，7，8}
2	{1，2，3，4，5，6，7，8}	{1，2}	{3，4，5，6，7，8}
3	{1，2，3，4，5，6，7，8}	{1，2，3}	{4，5，6，7，8}
4	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8}
5	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8} 终止

DIANA算法：练习

- 对于表中给定的样本数据，用DIANA算法将它们聚为三个簇(*k*=3)。

序号	属性 1	属性 2
1	1	0
2	4	0
3	0	1
4	1	1
5	3	1
6	5	1
7	4	2
8	1	3

3.2 层次方法

- 层次方法的主要问题
  - 可伸缩性不好，合并或分裂需要很大的计算开销，时间复杂度至少是$O(n^2)$
  - 一旦一个合并或分裂被执行，就不能撤消
- 多阶段聚类：将分层聚类和其他聚类技术集成，提高层次方法的聚类质量。
  - *CURE*
  - *BIRCH*
  - *ROCK*
  - …….

聚类分析

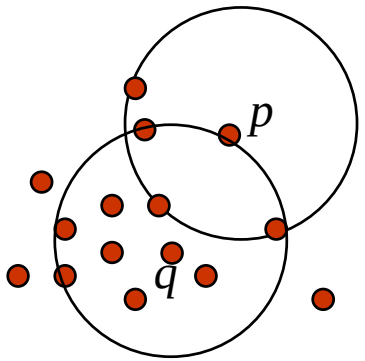
- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

3.3 基于密度的方法

- 将簇看作是数据空间中被低密度区域分割开的高密度区域，只要一个区域中的点的密度大于某个域值，就把它加到与之相近的簇中去。
- 优点
  - 可发现任意形状的分簇
  - 对噪声数据不敏感
- 缺点
  - 计算密度单元的计算复杂度大，需要建立空间索引来降低计算量；
  - 对数据维数的伸缩性较差；
  - 需要扫描整个数据库。

3.3 基于密度的方法

- DBSCAN(Density-Based Spatial Clustering of Applications with Noise)：基于高密度连接区域的密度聚类方法
- 对象的ε-邻域：给定对象在半径ε内的区域，$N_{\varepsilon}(p) = \{q \in D \mid dist(p, q) \leq \varepsilon\}$
- 核心对象：如果一个对象的ε-邻域至少包含最小数目*MinPts*个对象，则称该对象为核心对象
- 直接密度可达：给定一个对象集合D，如果p是在q的ε-邻域内，而q是一个核心对象，则对象p从对象q出发是直接密度可达的

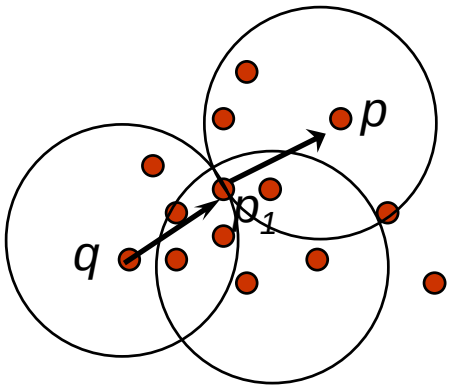


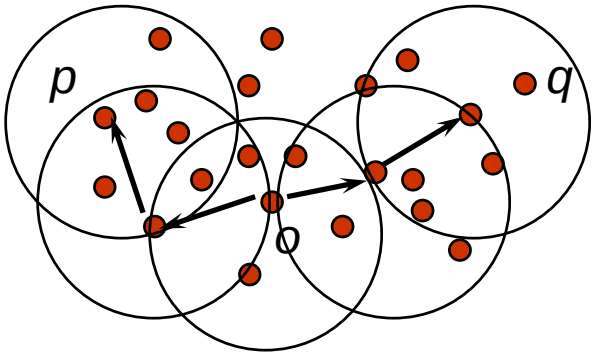
MinPts = 5

ε = 1 cm

5.3.3 基于密度的方法

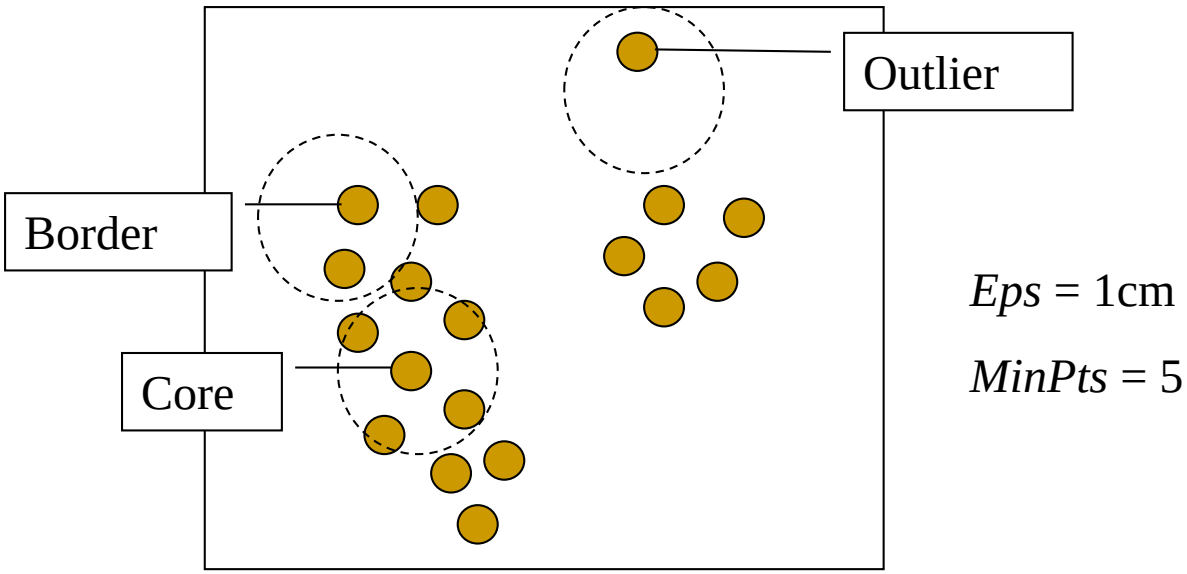
- 密度可达的：如果存在一个对象链p_1，p_2，…，p_n，$p_1=q$，$p_n=p$，对$p_i \in D$，（$1<=i<=n$），p_{i+1}是从p_i关于ε和*MitPts*直接密度可达的，则对象p是从对象q关于ε和*MinPts*密度可达的
- 密度相连的：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和*MinPts*密度可达的，那么对象p和q是关于ε和*MinPts*密度相连的





3.3 基于密度的方法

- 噪声: 一个基于密度的簇是基于密度可达性的最大的密度相连对象的集合。不包含在任何簇中的对象被认为是“噪声”
- 边界对象：不是核心对象，但在簇中，即至少从一个核心对象直接可达
- 簇：基于密度可达的最大的密度相连对象的集合



3.3 基于密度的方法

- DBSCAN算法
  - 1. 任意选择没有加簇标记的点 p
  - 2. 找到从p关于ε和 $MinPts$ 密度可达的所有点
  - 3. 如果$|N_{\varepsilon}(p)| \geq MinPts$，则$p$是核心对象，形成一个新的簇，给簇内所有的对象点加簇标记
  - 4. 如果p是边界点, 则处理数据库的下一点
  - 5. 重复上述过程，直到所有的点处理完毕

3.3 基于密度的方法

DBSCAN算法的执行步骤（设n=12，用户输入ε=1，*MinPts*=4）

样本数据库

序号	属性 1	属性 2
1	1	0
2	4	0
3	0	1
4	1	1
5	2	1
6	3	1
7	4	1
8	5	1
9	0	2
10	1	2
11	4	2
12	1	3

DBSCAN算法执行过程示意

步骤	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
1	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10，12}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中，
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

聚出的类为{1，3，4，5，9，11，12}，{2，6，7，8，10}。

3.3 基于密度的方法

算法执行过程：

步骤	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
1	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10，12}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中，
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中

第1步，在数据库中选择一点1，由于在以它为圆心的，以1为半径的圆内包含2个点（包括点1自己，小于阈值4），因此它不是核心点，选择下一个点。

第2步，在数据库中选择一点2，由于在以它为圆心的，以1为半径的圆内包含2个点，因此它不是核心点，选择下一个点。

第3步，在数据库中选择一点3，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第4步，在数据库中选择一点4，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点（直接可达5个，间接可达2个），聚出的新类{1，3，4，5，9，10，12}，选择下一个点。

第5步，在数据库中选择一点5，已经在簇1中，选择下一个点。

第6步，在数据库中选择一点6，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第7步，在数据库中选择一点7，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点，聚出的新类{2，6，7，8，11}，选择下一个点。

第8步，在数据库中选择一点8，已经在簇2中，选择下一个点。

第9步，在数据库中选择一点9，已经在簇1中，选择下一个点。

第10步，在数据库中选择一点10，已经在簇1中，选择下一个点。

第11步，在数据库中选择一点11，已经在簇2中，选择下一个点。

第12步，选择12点，已经在簇1中，由于这已经是最后一点所有点都以处理，程序终止。

3.3 基于密度的方法

- Comments：
  - 只能发现密度相仿的簇，对密度不均的数据聚合效果不好
  - 需要指定参数(ε and *MinPts*)，且对定义的参数敏感
  - 对初值(起始点)选取敏感，对噪声不敏感
  - 不需要提前设置聚类的个数，自动生成
  - 计算复杂度为$O(n^2)$
  - 可采用R-树等空间索引技术来降低计算复杂度($O(nlogn)$)

聚类分析

- 1. 聚类分析概述
- 2. 相似性计算方法
- 3. 常用聚类方法
  - 3.1 划分方法
  - 3.2 层次方法
  - 3.3 基于密度的方法
- 4. 聚类的常用评价指标

4 聚类的常用评价指标

- 由于聚类是一种无监督学习方法，因此没有可以比较聚类结果的基础真值标签
- **确定“正确”簇数量或“最佳”簇通常是一个主观的决定**
- 对聚类结果，可以通过某种性能度量来一定程度上评估其好坏，若明确了最终将要使用的性能度量，则可直接将其作为聚类过程的优化目标
- 一般使用两种类型的聚类评估度量:
  - 内部指标：完全基于数据和聚类结果
  - 外部指标：将聚类结果与真值标签或其他模型的结果进行比较（需要从外部引入真值或其他模型的结果）

4 聚类的常用评价指标

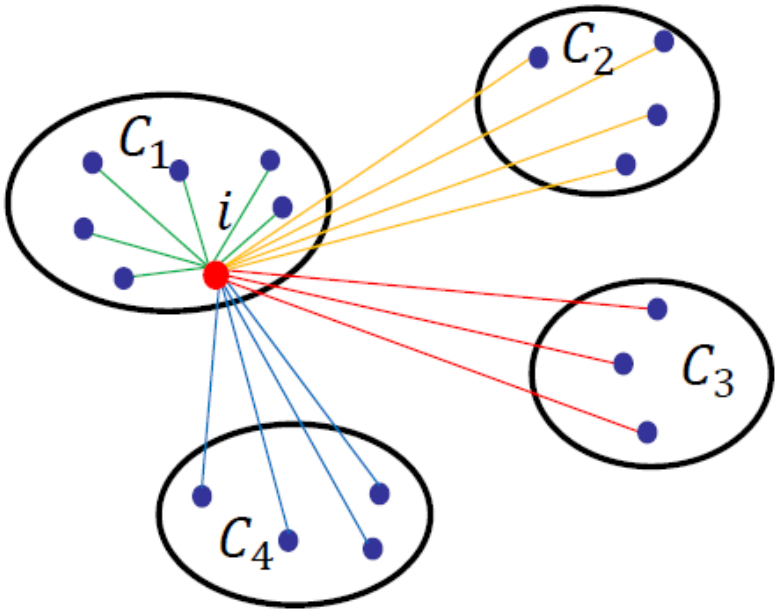
- **轮廓系数(Silhouette Coefficient)：**
  - 轮廓系数使用同一聚类中的点之间的距离，以及下一个临近聚类中的点与所有其他点之间的距离来评估模型的表现。首先定义数据点x_i的轮廓系数为

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}}$$

  - 簇内不相似度：a 为x_i到同簇其它数据点的平均距离，应尽可能小
  - 簇间不相似度：b 为x_i到最近的其它簇的所有数据点的平均距离，应尽可能大

4 聚类的常用评价指标

- **数据点的轮廓系数：**
  - 假设数据集被拆分为4个簇，样本i 对应的$a(i)$ 值就是所有C_1 中其他样本点与样本i 的距离平均值
  - 样本对应的$b(i)$ 值分两步计算，首先计算该点分别到C_2、C_3 和C_4 中样本点的平均距离，然后将三个平均值中的最小值作为$b(i)$ 的度量



4 聚类的常用评价指标

- **轮廓系数(Silhouette Coefficient)：**
  - 轮廓系数s 取值范围[-1,1]，越接近1表示对x_i的聚类越合理；越接近-1，表示样本x_i应该分类到另外的簇中；近似为0，则表示样本x_i应该在边界上
  - 所有样本x_i的轮廓系数的均值被定义为聚类结果的轮廓系数

$$SI = \frac{1}{n}\sum_{i=1}^{n} s(\mathbf{x}_i)$$

  - n为样本的总数

4 聚类的常用评价指标

- **方差比准则（Calinski-Harabasz Index）：**
  - CHI是簇内和簇间方差的比值，所以又将其称之为方差比准则，反映了不同簇之间的差异性和簇内部的一致性

$$CHI = \frac{BCSS/(k-1)}{WCSS/(n-k)}$$

  - *k*是簇的数量，*n*是数据点的总数
  - BCSS 是每个聚类质心(mean)与整体数据质心之间欧氏距离的加权平方和
  - WCSS 是每个簇内数据点与其各自聚类质心之间的欧氏距离的平方和
  - [0, +∞) 指数越大，聚类效果越好

4 聚类的常用评价指标

- **Davies-Bouldin指数(DB值)：**
  - DBI衡量每个聚类与其最相似的聚类之间的平均相似度，其中相似度定义为聚类内距离(聚类中点到聚类中心的距离)与聚类间距离(聚类中心之间的距离)之比

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{S_i + S_j}{d(\mathbf{c}_i, \mathbf{c}_j)}\right)$$

  - k是簇的数量，S_i是簇i中所有点到簇的质心c_i的平均距离
  - [0, +∞)，如果簇与簇之间的相似度越高（DB指数偏高），也就说明簇与簇之间的距离越小（直观上理解只有相距越近的事物才会约相似），那么此时聚类结果就越差，反之亦然

4 聚类的常用评价指标

- **以上的内部评价指标局限：**
  - 倾向于支持凸簇，而非凸或不规则形状的簇可能表现不佳
  - 不适合评估像DBSCAN这样基于密度的聚类结果
  - SI计算量大，因为它需要计算所有$O(n^2)$个点之间的成对距离。这可能会使评估过程比聚类本身更昂贵(例如，当使用k-means时)
  - 对噪声和异常值敏感，因为它依赖于可能受异常值影响的平均距离

4 聚类的常用评价指标

- **均一性：p**
  - 类似于精确率，一个簇中只包含一个类别的样本，则满足均一性。其实也可以认为就是正确率(每个聚簇中正确分类的样本数占该聚簇总样本数的比例和)

$$p = \frac{1}{k}\sum_{i=1}^{k}\frac{N(C_i == K_i))}{N(K_i)}$$

4 聚类的常用评价指标

- **完整性：r**
  - 类似于召回率，同类别样本被归类到相同簇中，则满足完整性(每个聚簇中正确分类的样本数占该类型的总样本数比例的和)

$$r = \frac{1}{k}\sum_{i=1}^{k}\frac{N(C_i == K_i))}{N(C_i)}$$

- **V-measure：均一性和完整性的加权平均（β 默认为1）**

$$V = \frac{(1+\beta^2) * pr}{\beta^2 * p + r}$$

4 聚类的常用评价指标

- **兰德指数(Rand Index)：**
  - Rand Index 衡量的是聚类算法将数据点分配到聚类中的准确程度。 Rand Index 的范围为 [0,1]，如果 Rand Index 为 1 表示两个聚类完全相同，接近 0 表示两个聚类有很大的不同

$$RI = \frac{a+b}{\binom{n}{2}}$$

  - a：在真实标签中处于同一簇中的样本对数，在预测聚类中处于同一簇中的样本对数
  - b：真实聚类和预测聚类中处于不同聚类的样本对的数目

4 聚类的常用评价指标

- **调整兰德指数(Adjusted Rand)：**
  - 它是 Rand Index 的一种调整形式，可以用于评估将样本点分为多个簇的聚类算法。 它考虑了机会的概率，取值范围为 [−1,1]，其中值越接近 1 表示聚类结果越准确， 值越接近 0 表示聚类结果与随机结果相当，值越接近 -1 表示聚类结果与真实类别完全相反。从广义的角度来讲，ARI 衡量的是两个数据分布的吻合程度

$$Adj\_RI = \frac{(RI - ExpectedRI)}{max(RI) - ExpectedRI}$$

  - 注意，Rand Index只能用于评估将样本点分成两个簇的聚类算法。对于将样本点分成多个簇的聚类算法，需要使用其他的指标来评估其性能

4 聚类的常用评价指标

- **Fowlkes-Mallows Index（FMI）：**
  - 定义为对精度(分组点对的准确性)和召回率(正确分组在一起的对的完整性)的几何平均值

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$$

  - TP(True Positive)是具有相同类标签并属于同一簇的点对的数量。FP (False Positive)是具有不同类标签但被分配到同一聚类的点对的数量。FN(False Negative)是具有相同类标签但分配给不同簇的点对的数量
  - (0,1)，其中0表示聚类结果与真实标签不相关，1表示完全相关

4 聚类的常用评价指标

- **以上外部评价指标的优缺点：**
  - 没有对簇结构做任何假设，支持任意形式的簇
  - 同时考虑准确率和召回率，提供一个平衡的聚类性能视图
  - 对噪声和异常值不敏感
  - 但是需要有真实的标签来确定结果

4 聚类的常用评价指标

- 评价指标对比：

Measure	Type	Range	Adjusted for Chance	Assumptions on Clusters
Silhouette Index (SI)	Internal	[-1, 1] higher is better	No	Spherical
Calinski-Harabasz Index (CHI)	Internal	[0, ∞] higher is better	No	Spherical, similar sized
Davies-Bouldin Index (DBI)	Internal	[0, ∞] lower is better	No	Spherical, similar sized
Adjusted Rand Index (ARI)	External	[-1, 1] higher is better	Yes	None
V-Measure	External	[0, 1] higher is better	No	None
Fowlkes-Mallows Index (FMI)	External	[0, 1] higher is better	Yes	None