CHAPTER 5

# 第5章 聚类方法

“物以类聚，人以群分”，聚类是人类的一项最基本的认识活动。聚类的用途是非常广泛的。在生物学中，聚类可以辅助动、植物分类方面的研究，以及通过对基因数据的聚类，找出功能相似的基因；在地理信息系统中，聚类可以找出具有相似用途的区域，辅助石油开采；在商业上，聚类可以帮助市场分析人员对消费者的消费记录进行分析，从而概括出每一类消费者的消费模式，实现消费群体的区分。

聚类就是将数据对象分组成为多个类或簇(Cluster)，划分的原则是在同一个簇中的对象之间具有较高的相似度，而不同簇中的对象差别较大。与分类不同的是，聚类操作中要划分的类是事先未知的，类的形成完全是数据驱动的，属于一种无指导的学习方法。

本章内容安排如下：首先对聚类方法进行一个简要、全面的概述，包括对聚类的概念、算法的分类方法、相似性度量等；然后详细介绍几种典型的聚类方法，包括划分方法 $k$-平均($k$-means)和 $k$-中心点($k$-medoids)，层次聚类方法 AGNES 和 DIANA，密度聚类方法 DBSCAN；最后进行一个简单的小结。

## 5.1 概述

聚类分析源于许多研究领域，包括数据挖掘、统计学、机器学习、模式识别等。它是数据挖掘中的一个功能，但也能作为一个独立的工具来获得数据分布的情况，概括出每个簇的特点，或者集中注意力对特定的某些簇作进一步的分析。此外，聚类分析也可以作为其他分析算法(如关联规则、分类等)的预处理步骤，这些算法在生成的簇上进行处理。

数据挖掘技术的一个突出的特点是处理巨大的、复杂的数据集，这对聚类分析技术提出了特殊的挑战，要求算法具有可伸缩性、处理不同类型属性的能力、发现任意形状的类的能力、处理高维数据的能力等。根据潜在的各项应用，数据挖掘对聚类分析方法提出了不同要求。典型要求可以通过以下几个方面来刻画。

**1. 可伸缩性**

可伸缩性是指聚类算法不论对于小数据集还是对于大数据集,都应是有效的。在很多聚类算法当中,数据对象小于几百个的小数据集合上鲁棒性很好,而对于包含上万个数据对象的大规模数据库进行聚类时,将会导致不同的偏差结果。研究大容量数据集的高效聚类方法是数据挖掘必须面对的挑战。

**2. 具有处理不同类型属性的能力**

既可处理数值型数据,又可处理非数值型数据;既可以处理离散数据,又可以处理连续域内的数据,如布尔型,序数型,枚举型或这些数据类型的混合。

**3. 能够发现任意形状的聚类**

许多聚类算法经常使用欧几里得距离来作为相似性度量方法,但基于这样的距离度量的算法趋向于发现具有相近密度和尺寸的球状簇。但对于一个簇可能是任意形状的情况,提出能发现任意形状簇的算法是很重要的。

**4. 输入参数对领域知识的弱依赖性**

在聚类分析当中,许多聚类算法要求用户输入一定的参数,如希望得到的簇的数目。聚类结果对于输入的参数很敏感,通常参数较难确定,尤其是对于含有高维对象的数据集更是如此。要求用人工输入参数不但加重了用户的负担,也使得聚类质量难以控制。一个好的聚类算法应该对这个问题给出一个好的解决方法。

**5. 对于输入记录顺序不敏感**

一些聚类算法对于输入数据的顺序是敏感的。例如,对于同一个数据集合,以不同的顺序提交给同一个算法时,可能产生差别很大的聚类结果。研究和开发对数据输入顺序不敏感的算法具有重要的意义。

**6. 挖掘算法应具有处理高维数据的能力**

既可处理属性较少的数据,又能处理属性较多的数据。很多聚类算法擅长处理低维数据,一般只涉及两到三维,人类对两三维数据的聚类结果很容易直观地判断聚类的质量。但是,高维数据聚类结果的判断就不是那样直观了。数据对象在高维空间的聚类是非常具有挑战性的,尤其是考虑到这样的数据可能高度偏斜并且非常稀疏。

**7. 处理噪声数据的能力**

在现实应用中绝大多数的数据都包含了孤立点、空缺、未知数据或者错误的数据。如果聚类算法对于这样的数据敏感,将会导致质量较低的聚类结果。

**8. 基于约束的聚类**

在实际应用当中可能需要在各种约束条件下进行聚类。既要找到满足特定的约束,

又要具有良好聚类特性的数据分组是一项具有挑战性的任务。

**9. 挖掘出来的信息是可理解的和可用的**

这一点是容易理解的，但在实际挖掘中有时往往不能令人满意。

## 5.1.1　聚类分析在数据挖掘中的应用

聚类分析在数据挖掘中的应用主要有以下几个方面：

**1. 聚类分析可以作为其他算法的预处理步骤**

利用聚类进行数据预处理，可以获得数据的基本概况，在此基础上进行特征抽取或分类就可以提高精确度和挖掘效率。也可将聚类结果用于进一步关联分析，以进一步获得有用的信息。

**2. 可以作为一个独立的工具来获得数据的分布情况**

聚类分析是获得数据分布情况的有效方法。例如，在商业上，聚类分析可以帮助市场分析人员从客户基本库当中发现不同的客户群，并且用购买模式来刻画不同的客户群的特征。通过观察聚类得到的每个簇的特点，可以集中对特定的某些簇作进一步分析。这在诸如市场细分、目标顾客定位、业绩估评、生物种群划分等方面具有广阔的应用前景。

**3. 聚类分析可以完成孤立点挖掘**

许多数据挖掘算法试图使孤立点影响最小化，或者排除它们。然而孤立点本身可能是非常有用的。如在欺诈探测中，孤立点可能预示着欺诈行为的存在。

## 5.1.2　聚类分析算法的概念与基本分类

**1. 聚类概念**

**定义 5-1**　聚类分析的输入可以用一组有序对$(X,s)$或$(X,d)$表示，这里 $X$ 表示一组样本，$s$ 和 $d$ 分别是度量样本间相似度或相异度(距离)的标准。聚类系统的输出是对数据的区分结果，即 $C=\{C_1,C_2,\cdots,C_k\}$，其中 $C_i(i=1,2,\cdots,k)$是 $X$ 的子集，且满足如下条件：

(1)　$$C_1 \cup C_2 \cup \cdots \cup C_k = X$$

(2)　$$C_i \cap C_j = \varnothing, \quad i \neq j$$

$C$ 中的成员 $C_1,C_2,\cdots,C_k$ 称为类或者簇。每一个类可以通过一些特征来描述。通常有如下几种表示方式：

- 通过类的中心或类的边界点表示一个类。
- 使用聚类树中的结点图形化地表示一个类。
- 使用样本属性的逻辑表达式表示类。

用类的中心表示一个类是最常见的方式，当类是紧密的或各向分布同性时用这种方法非常好，然而，当类是伸长的或各向分布异性时，这种方式就不能正确地表示它们了。

**2. 聚类分析方法的分类**

聚类分析是一个活跃的研究领域,已经有大量的、经典的和流行的算法涌现,例如$k$-平均、$k$-中心点、PAM、CLARANS、BIRTH、CURE、OPTICS、DBSCAN、STING、CLIQUE、Wave Cluster等。采用不同的聚类方法,对于相同的数据集可能有不同的划分结果。很多文献从不同角度对聚类分析方法进行了分类,概括来讲,有如下几种分类方法。

(1) 按聚类的标准划分

按照聚类的标准,聚类方法可分为以下两种:

① 统计聚类方法。

统计聚类方法基于对象之间的几何距离。统计聚类分析包括系统聚类法、分解法、加入法、动态聚类法、有序样品聚类、有重叠聚类和模糊聚类等。这种聚类方法是一种基于全局比较的聚类,它需要考察所有的个体才能决定类的划分。因此,它要求所有的数据必须预先给定,而不能动态地增加新的数据对象。

② 概念聚类方法。

概念聚类方法基于对象具有的概念进行聚类。这里的距离不再是传统方法中的几何距离,而是根据概念的描述来确定的。典型的概念聚类或形成方法有COBWEB、OLOC和基于列联表的方法。

(2) 按聚类算法所处理的数据类型划分

按照聚类算法所处理的数据类型,聚类方法可分为以下三种:

① 数值型数据聚类方法。

数值型数据聚类方法所分析的数据的属性为数值数据,因此可对所处理的数据直接比较大小。目前,大多数的聚类算法都是基于数值型数据的。

② 离散型数据聚类方法。

由于数据挖掘的内容经常含有非数值的离散数据,近年来人们在离散型数据聚类方法方面做了许多研究,提出了一些基于此类数据的聚类算法,如$k$-模($k$-mode)、ROCK、CACTUS、STIRR。

③ 混合型数据聚类方法。

混合型数据聚类方法是能同时处理数值数据和离散数据的聚类方法,这类聚类方法通常功能强大,但性能往往不尽如人意。混合型数据聚类方法的典型算法有$k$-原型($k$-prototypes)算法。

(3) 按聚类的尺度划分

按照聚类的尺度,聚类方法可被分为以下三种:

① 基于距离的聚类算法。

距离是聚类分析常用的分类统计量。常用的距离定义有欧氏距离和马氏距离。许多聚类算法都是用各式各样的距离来衡量数据对象之间的相似度,如$k$-平均、$k$-中心点、BIRCH、CURE等算法。算法通常需要给定聚类数目$k$,或区分两个类的最小距离。基于距离的算法聚类标准易于确定、容易理解,对数据维度具有伸缩性,但只适用于欧几里得

空间和曼哈坦空间，对孤立点敏感，只能发现圆形类。为克服这些缺点，提高算法性能，$k$-中心点、BIRCH、CURE 等算法采取了一些特殊的措施。如 CURE 算法使用固定数目的多个数据点作为类代表，这样可提高算法处理不规则聚类的能力，降低对孤立点的敏感度。

② 基于密度的聚类算法。

从广义上说，基于密度和基于网格的算法都可算作基于密度的算法。此类算法通常需要规定最小密度门限值。算法同样适用于欧几里得空间和曼哈坦空间，对噪声数据不敏感，可以发现不规则的类，但当类或子类的粒度小于密度计算单位时，会被遗漏。

③ 基于互连性的聚类算法。

基于互连性(Linkage-Based)的聚类算法通常基于图或超图模型。它们通常将数据集映像为图或超图，满足连接条件的数据对象之间画一条边，高度连通的数据聚为一类。属于此类的算法有 ROCK、CHAMELEON、ARHP、STIRR、CACTUS 等。此类算法可适用于任意形状的度量空间，聚类的质量取决于链或边的定义，不适合处理太大的数据集。当数据量大时，通常忽略权重小的边，使图变稀疏，以提高效率，但会影响聚类质量。

(4) 按聚类算法的思路划分

按照聚类分析算法的主要思路，它可以归纳为以下几种：

① 划分法(Partitioning Methods)。

给定一个 $n$ 个对象或者元组的数据库，划分方法构建数据的 $k$ 个划分，每个划分表示一个簇，并且 $k \leqslant n$。也就是说，它将数据划分为 $k$ 个组，同时满足如下的要求：每个组至少包含一个对象；每个对象必须属于且只属于一个组。

属于该类的聚类方法有$k$-平均、$k$-模、$k$-原型、$k$-中心点、PAM、CLARA、CLARANS 等。

② 层次法(Hierarchical Methods)。

层次的方法对给定数据对象集合进行层次的分解。根据层次的分解如何形成，层次的方法又可以分为凝聚的和分裂的。

分裂的方法，也称为自顶向下的方法，一开始将所有的对象置于一个簇中。在迭代的每一步中，一个簇被分裂成更小的簇，直到每个对象在一个单独的簇中，或者达到一个终止条件。如 DIANA 算法属于此类。

凝聚的方法，也称为自底向上的方法，一开始就将每个对象作为单独的一个簇，然后相继地合并相近的对象或簇，直到所有的簇合并为一个，或者达到终止条件。如 AGNES 算法即属于此类。

③ 基于密度的方法(Density-based Methods)。

基于密度的方法与其他方法的一个根本区别是：它不是用各式各样的距离作为分类统计量，而是看数据对象是否属于相连的密度域。属于相连密度域的数据对象归为一类。如 DBSCAN 即属于密度聚类方法。

④ 基于网格的方法(Grid-based Methods)。

这种方法首先将数据空间划分成为有限个单元(Cell)的网格结构，所有的处理都是以单个单元为对象的。这样处理的一个突出优点是处理速度快，通常与目标数据库中记

录的个数无关,只与把数据空间分为多少个单元有关。但处理方法较粗放,往往影响聚类质量。代表算法有 STING、CLIQUE、WaveCluster、DBCLASD、OptiGrid 算法。

⑤ 基于模型的方法(Model-Based Methods)。

基于模型的方法给每一个簇假定一个模型,然后去寻找能够很好地满足这个模型的数据集。这样一个模型可能是数据点在空间中的密度分布函数或者其他函数。它的一个潜在的假定是:目标数据集是由一系列的概率分布所决定的。通常有两种尝试方案:统计的方案和神经网络的方案。基于统计学模型的方法有 COBWEB、Autoclass;基于神经网络模型的是 SOM。

### 5.1.3 距离与相似性的度量

一个聚类分析过程的质量取决于对度量标准的选择,因此必须仔细选择度量标准。

为了度量对象之间的接近或相似程度,需要定义一些相似性度量标准。本章我们用 $s(x,y)$ 表示样本 $x$ 和样本 $y$ 的相似度。当 $x$ 和 $y$ 相似时,$s(x,y)$ 的取值是很大的;当 $x$ 和 $y$ 不相似时,$s(x,y)$ 的取值是很小的。相似性的度量具有自反性 $s(x,y)=s(y,x)$。对于大多数聚类方法来说,相似性度量标准被标准化为 $0\leqslant s(x,y)\leqslant 1$。

但是在通常情况下,聚类算法不是计算两个样本间的相似度,而是用特征空间中的距离作为度量标准来计算两个样本间的相异度。对于某个样本空间来说,距离的度量标准可以是度量的或半度量的,以便用来量化样本的相异度。相异度的度量用 $d(x,y)$ 来表示,通常称相异度为距离。当 $x$ 和 $y$ 相似时,距离 $d(x,y)$ 的取值很小;当 $x$ 和 $y$ 不相似时,$d(x,y)$ 就很大。

下面对这些度量标准进行简要介绍。

#### 1. 距离函数

按照距离公理,在定义距离测度时需要满足距离公理的四个条件:自相似性、最小性、对称性以及三角不等性。常用的距离函数有如下几种:

(1) 明可夫斯基距离(Minkowski)

假定,$x,y$ 是相应的特征,$n$ 是特征的维数。$x$ 和 $y$ 的明可夫斯基距离度量的形式如下:

$$d(x,y)=\left[\sum_{i=1}^{n}|x_i-y_i|^r\right]^{\frac{1}{r}}$$

当 $r$ 取不同的值时,上述距离度量公式演化为一些特殊的距离测度。

- 当 $r=1$ 时,明可夫斯基距离演变为绝对值距离:

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

- 当 $r=2$ 时,明可夫斯基距离演变为欧氏距离:

$$d(x,y)=\left[\sum_{i=1}^{n}|x_i-y_i|^2\right]^{\frac{1}{2}}$$

(2) 二次型距离(Quadratic)

二次型距离测度的形式如下：

$$d(x,y)=((x-y)^{\mathrm{T}}A(x-y))^{\frac{1}{2}}$$

其中 $A$ 是非负定矩阵。

当 $A$ 取不同的值时，上述距离度量公式演化为一些特殊的距离测度：

- 当 $A$ 为单位矩阵时，二次型距离演变为欧氏距离。
- 当 $A$ 为对角阵时，二次型距离演变为加权欧氏距离：

$$d(x,y)=\left[\sum_{i=1}^{n}a_{ii}\mid x_i-y_i\mid^2\right]^{\frac{1}{2}}$$

- 当 $A$ 为协方差矩阵时，二次型距离演变为马氏距离。

(3) 余弦距离

余弦距离的度量形式如下：

$$d(x,y)=\frac{\sum_{i=1}^{n}x_iy_i}{\sqrt{\sum_{i=1}^{n}x_i^2\sum_{i=1}^{n}y_i^2}}$$

(4) 二元特征样本的距离度量

前面我们所阐述的几种距离度量对于包含连续特征的样本是很有效的。但对于包含一些或全部不连续特征的样本，计算样本间的距离是比较困难的。因为不同类型的特征是不可比的，只用一个标准作为度量标准是不合适的。下面我们介绍几种二元类型数据的距离度量标准。假定 $x$ 和 $y$ 分别是 $n$ 维特征，$x_i$ 和 $y_i$ 分别表示每维特征，且 $x_i$ 和 $y_i$ 的取值为二元类型数值{0,1}。则 $x$ 和 $y$ 的距离定义的常规方法是先求出如下几个参数，然后采用 SMC、Jaccard 系数或 Rao 系数。

假设：

- $a$ 是样本 $x$ 和 $y$ 中满足 $x_i=y_i=1$ 的二元类型属性的数量。
- $b$ 是样本 $x$ 和 $y$ 中满足 $x_i=1,y_i=0$ 的二元类型属性的数量。
- $c$ 是样本 $x$ 和 $y$ 中满足 $x_i=0,y_i=1$ 的二元类型属性的数量。
- $d$ 是样本 $x$ 和 $y$ 中满足 $x_i=y_i=0$ 的二元类型属性的数量。

则：

- 简单匹配系数 SMC(Simple Match Coefficient)定义为：

$$S_{smc}(x,y)=\frac{a+b}{a+b+c+d}$$

- Jaccard 系数定义为：

$$S_{jc}(x,y)=\frac{a}{a+b+c}$$

- Rao 系数定义为：

$$S_{rc}(x,y)=\frac{a}{a+b+c+d}$$

上面所给出的距离函数,都是关于两个样本的距离的,为考察聚类的质量,有时需要计算类间的距离。下面介绍几种常用的类间距离计算方法。

**2. 类间距离**

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 $r_a$ 和 $r_b$。设元素 $x\in C_a$,$y\in C_b$,这两个元素间的距离记为 $d(x,y)$。可以采用不同的策略来定义类间距离,记为 $D(C_a,C_b)$。

(1) 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:

$$D_S(C_a,C_b)=\min\{d(x,y)\mid x\in C_a,y\in C_b\}$$

(2) 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:

$$D_L(C_a,C_b)=\max\{d(x,y)\mid x\in C_a,y\in C_b\}$$

(3) 中心法

定义两类的两个中心间的距离为类间距离。

下面给出了类中心和类间距离的描述。

假如 $C_i$ 是一个聚类,使用 $C_i$ 的所有数据点 $x$,可以定义 $C_i$ 的类中心 $\overline{x_i}$ 如下:

$$\overline{x_i}=\frac{1}{n_i}\sum_{x\in C_i}x$$

其中 $n_i$ 是第 $i$ 个聚类中的点数。基于类中心,可以进一步定义两个类 $C_a$ 和 $C_b$ 的类间距离为:

$$D_C(C_a,C_b)=d(r_a,r_b)$$

(4) 类平均法

它将两个类中任意两个元素间的距离的平均值定义为类间距离,即:

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

其中 $m$ 和 $h$ 是两个类 $C_a$ 和 $C_b$ 的元素个数。

(5) 离差平方和

离差平方和用到了类直径的概念,我们首先介绍一下类直径。

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。或者说,其量纲为距离的平方,即类 $C_a$ 的直径表示为:

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

其中 $\overline{x_a}$ 为 $C_a$ 的类中心。

假设类 $C_a$ 和 $C_b$ 的直径分别为 $r_a$ 和 $r_b$,类 $C_{a+b}=C_a\cup C_b$ 直径为 $r_{a+b}$,则可定义类间距离的平方为:

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

## 5.2　划分聚类方法

划分聚类方法是最基本的聚类方法。像 $k$-平均、$k$-模、$k$-原型、$k$-中心点、PAM、CLARA 以及 CLARANS 等都属于划分聚类方法。

本节首先介绍划分聚类方法的主要思想，然后介绍经典的划分聚类方法 $k$-平均和 PAM 算法，最后介绍其他聚类方法。

**1. 主要思想**

**定义 5-2**　给定一个有 $n$ 个对象的数据集，划分聚类技术将构造数据 $k$ 个划分，每一个划分就代表一个簇，$k \leqslant n$。也就是说，它将数据划分为 $k$ 个簇，而且这 $k$ 个划分满足下列条件：

- 每一个簇至少包含一个对象。
- 每一个对象属于且仅属于一个簇。

对于给定的 $k$，算法首先给出一个初始的划分方法，以后通过反复迭代的方法改变划分，使得每一次改进之后的划分方案都较前一次更好。所谓好的标准就是：同一簇中的对象越近越好，而不同簇中的对象越远越好。目标是最小化所有对象与其参照点之间的相异度之和。这里的远近或者相异度/相似度实际上是聚类的评价函数。

**2. 评价函数**

大多数为聚类设计的评价函数都着重考虑两个方面：每个簇应该是紧凑的，各个簇间的距离应该尽量远。实现这种概念的一种直接方法就是观察聚类 $C$ 的类内差异(Within cluster variation)$w(C)$和类间差异(Between cluster variation)$b(C)$。类内差异衡量类内的紧凑性，类间差异衡量不同类之间的距离。

类内差异可以用多种距离函数来定义，最简单的就是计算类内的每一个点到它所属类中心的距离的平方和：

$$w(C) = \sum_{i=1}^{k} w(C_i) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \overline{x_i})^2$$

类间差异定义为类中心间的距离：

$$b(C) = \sum_{1 \leqslant j < i \leqslant k} d(\overline{x_j}, \overline{x_i})^2$$

聚类 $C$ 的总体质量可被定义为 $w(C)$和 $b(C)$的一个单调组合，比如 $b(C)/w(C)$。

针对上面的类内差异和类间差异计算方法，$w(C)$和 $b(C)$的复杂度是比较容易计算的。

下面要讨论的 $k$-平均算法就是用类内的均值作为聚类中心、用欧氏距离定义 $d$，并使上述 $w(C)$最小化。

### 5.2.1　k-平均算法

$k$-平均($k$-Means)，也被称为 $k$-均值，是一种得到最广泛使用的聚类算法。$k$-平均算法以 $k$ 为参数，把 $n$ 个对象分为 $k$ 个簇，以使簇内具有较高的相似度。相似度的计算根据

一个簇中对象的平均值来进行。

算法首先随机地选择 $k$ 个对象,每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离,将它赋给最近的簇。然后重新计算每个簇的平均值。这个过程不断重复,直到准则函数收敛。

$k$-Means 算法的准则函数定义为:

$$E=\sum_{i=1}^{k}\sum_{x\in C_i}|x-\overline{x_i}|^2$$

即 $E$ 是数据库所有对象的平方误差的总和。其中 $x$ 是空间中的点,表示给定的数据对象,$\overline{x_i}$是簇 $C_i$ 的平均值。这个准则可以保证生成的结果簇尽可能的紧凑和独立。

### 1. 算法描述

**算法 5-1** $k$-平均算法

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 xj to the cluster which has the closest mean;
//根据簇中对象的平均值,将每个对象赋给最类似的簇
(4) FOR i=1 to k DO $\overline{x_i}=\sum_{x\in C_i}x/|C_i|$;
//更新簇的平均值,即计算每个对象簇中对象的平均值
(5) Compute E; //计算准则函数 E
(6) UNTIL E 不再明显地发生变化;

### 2. 算法执行例子

**例子 5-1** 下面给出一个样本事务数据库(见表 5-1),并对它实施 $k$-平均算法。

**表 5-1 样本事务数据库**

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

根据所给的数据通过对其进行 $k$-平均算法(设 $n=8,k=2$),以下为算法的执行步骤。

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。

表 5-2 给出了整个过程中平均值计算和簇生成的过程和结果。

**表 5-2　样本事务数据库**

| 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) |

**3. 算法的性能分析**

(1) 优点

- $k$-平均算法是解决聚类问题的一种经典算法,这种算法简单、快速。
- 对处理大数据集,该算法是相对可伸缩的和高效率的,因为它的复杂度大约是 $O(n \cdot k \cdot t)$,其中 $n$ 是所有对象的数目,$k$ 是簇的数目,$t$ 是迭代的次数。通常地,$k \ll n$,且 $t \ll n$。这个算法经常以局部最优结束。
- 算法尝试找出使平方误差函数值最小的 $k$ 个划分。当结果簇是密集的,而簇与簇之间区别明显时,它的效果较好。

(2) 缺点

- $k$-平均方法只有在簇的平均值被定义的情况下才能使用。这可能不适用于某些应用,例如涉及有分类属性的数据。
- 要求用户必须事先给出 $k$(要生成的簇的数目)可以算是该方法的一个缺点。而且 $k$ 值的选择对聚类的质量和效果影响很大。
- $k$-平均方法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“噪声”和孤立点数据是敏感的,少量的该类数据能够对平均值产生极大影响。

(3) 改进措施

为了实现对离散数据的快速聚类,$k$-模算法被提出,它保留了 $k$-平均算法的效率同时将 $k$-平均的应用范围扩大到离散数据。$k$-原型可以对离散与数值属性两种混合的数据进行聚类,在 $k$-原型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。

$k$-平均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靠近中心的对象,即中心点作为参照点。$k$-中心点算法的基本

思路是：首先为每个簇任意选择一个代表对象；剩余的对象根据其与代表对象的距离分配给最近的一个簇。然后反复地用非代表对象来代替代表对象，以改进聚类的质量。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。

### 5.2.2 PAM

PAM(Partitioning Around Medoid，围绕中心点的划分)是最早提出的 $k$-中心点算法之一，它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 $O_i$ 可以被使最大平方-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。

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：假设 $O_i$ 被 $O_h$ 代替作为新的中心点，$O_j$ 当前隶属于中心点对象 $O_i$。如果 $O_j$ 离某个中心点 $O_m$ 最近，$i \neq m$，那么 $O_j$ 被重新分配给 $O_m$。
- 第二种情况：假设 $O_i$ 被 $O_h$ 代替作为新的中心点，$O_j$ 当前隶属于中心点对象 $O_i$。如果 $O_j$ 离这个新的中心点 $O_h$ 最近，那么 $O_j$ 被分配给 $O_h$。
- 第三种情况：假设 $O_i$ 被 $O_h$ 代替作为新的中心点，但是 $O_j$ 当前隶属于另一个中心点对象 $O_m$，$m \neq i$。如果 $O_j$ 依然离 $O_m$ 最近，那么对象的隶属不发生变化。
- 第四种情况：假设 $O_i$ 被 $O_h$ 代替作为新的中心点，但是 $O_j$ 当前隶属于另一个中心点对象 $O_m$，$m \neq i$。如果 $O_j$ 离这个新的中心点 $O_h$ 最近，那么 $O_i$ 被重新分配给 $O_h$。

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。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。

总代价定义如下：

$$\mathrm{TC}_{ih} = \sum_{j=1}^{n} C_{jih}$$

其中 $C_{jih}$ 表示 $O_j$ 在 $O_i$ 被 $O_h$ 代替后产生的代价。

接下来将介绍上面所述的四种情况中代价函数的计算公式，所引用的符号中，$O_i$ 和 $O_m$ 是两个原中心点，$O_h$ 将替换 $O_i$ 作为新的中心点。

- 第一种情况：$O_j$ 当前隶属于 $O_i$，但 $O_h$ 替换 $O_i$ 后 $O_j$ 被重新分配给 $O_m$，则代价函数为：

$$C_{jih} = d(j,m) - d(j,i)$$

- 第二种情况：$O_j$ 当前隶属于 $O_i$，但 $O_h$ 替换 $O_i$ 后 $O_j$ 被重新分配给 $O_h$，则代价函数为：

$$C_{jih} = d(j,h) - d(j,i)$$

- 第三种情况：$O_j$ 当前隶属于另一个中心点对象 $O_m(m \neq i)$，但 $O_h$ 替换 $O_i$ 后 $O_j$ 的隶属不发生变化，则代价函数为：

$$C_{jih} = 0$$

- 第四种情况：$O_j$ 当前隶属于另一个中心点对象 $O_m(m \neq i)$，但 $O_h$ 替换 $O_i$ 后 $O_j$ 被重新分配给 $O_h$，则代价函数为：

$$C_{jih} = d(j,h) - d(j,m)$$

图 5-1 的(a)、(b)、(c)、(d)分别表示上述情况。

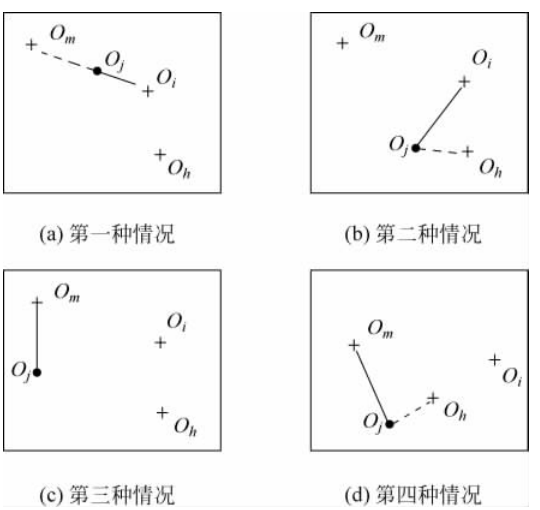



图 5-1　PAM 算法代价函数的四种情况

在 PAM 算法中，可以把过程分为两个步骤。

- 建立：随机寻找 $k$ 个中心点作为初始的簇中心点。
- 交换：对于所有可能的对象对进行分析，找到交换后可以使平方-误差减少的对象，代替原中心点。

### 1. 算法描述

**算法 5-2**　PAM($k$-中心点算法)

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

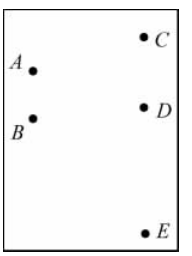


图 5-2 样本点

**2. 算法执行例子**

**例子 5-2** 假如空间中的五个点$\{A,B,C,D,E\}$,如图 5-2 所示。各点之间的距离关系如表 5-3 所示,根据所给的数据对其运行 PAM 算法实现划分聚类(设 $k=2$)。

**表 5-3 样本点间距离**

| 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 |

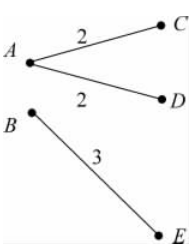


图 5-3 起始中心点为 $A$,$B$

算法执行步骤如下。

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。

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。

- 当 $A$ 被 $C$ 替换以后,看 $A$ 是否发生变化:$A$ 不再是一个中心点,$C$ 成为新的中心点,属于上面的第一种情况。因为 $A$ 离 $B$ 比 $A$ 离 $C$ 近,$A$ 被分配到 $B$ 中心点代表的簇:

$$C_{AAC} = d(A,B) - d(A,A) = 1$$

- 当 $A$ 被 $C$ 替换以后,看 $B$ 是否发生变化:属于上面的第三种情况。当 $A$ 被 $C$ 替换以后,$B$ 不受影响:

$$C_{BAC} = 0$$

当 $A$ 被 $C$ 替换以后,看 $C$ 是否发生变化:$C$ 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,符合图 5-1 所示的第二种情况:

$$C_{CAC} = d(C,C) - d(C,A) = 0 - 2 = -2$$

- 当 $A$ 被 $C$ 替换以后，看 $D$ 是否发生变化：$D$ 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 $C$，根据图 5-1 所示的第二种情况：

$$C_{DAC} = d(D,C) - d(D,A) = 1 - 2 = -1$$

- 当 $A$ 被 $C$ 替换以后，看 $E$ 是否发生变化：$E$ 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 $B$，根据图 5-1 所示的第三种情况：

$$C_{EAC} = 0$$

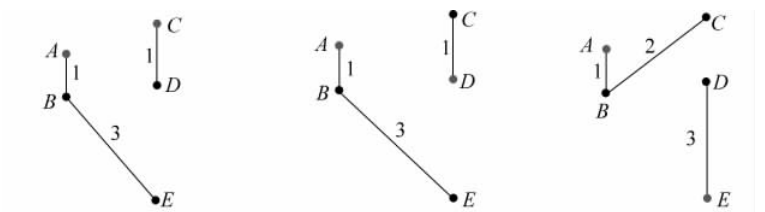
因此，$TC_{AC} = C_{AAC} + C_{BAC} + C_{CAC} + C_{DAC} + C_{EAC} = 1 + 0 - 2 - 1 + 0 = -2$。

同理，可以计算出 $TC_{AD} = -2$，$TC_{AE} = -1$，$TC_{BC} = -2$，$TC_{BD} = -2$，$TC_{BE} = -2$。

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，选择第一个最小代价的替换（也就是 $C$ 替换 $A$），这样，样本点被重新划分为 $\{A,B,E\}$ 和 $\{C,D\}$ 两个簇，如图 5-4(a)所示。

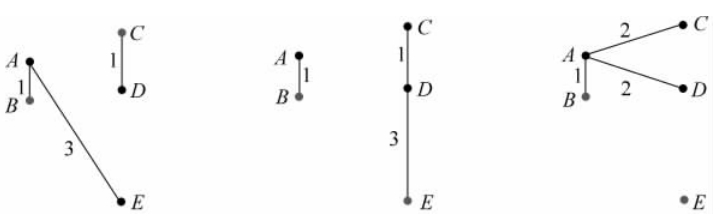
图 5-4(b)和图 5-4(c)分别表示了 $D$ 替换 $A$，$E$ 替换 $A$ 的情况和相应的代价。图 5-5(a)、(b)、(c)分别表示了用 $C$、$D$、$E$ 替换 $B$ 的情况和相应的代价。

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。关于后几次迭代过程，本章不再赘述，请读者自己完成。




(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 $A$




(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 $B$

### 3. 算法的性能分析

- $k$-中心点算法消除了 $k$-平均算法对于孤立点的敏感性。

- 当存在“噪声”和孤立点数据时,$k$-中心点方法比$k$-平均方法更健壮,这是因为中心点不像平均值那么容易被极端数据影响,但是,$k$-中心点方法的执行代价比$k$-平均方法高。
- 算法必须指定聚类的数目$k$,$k$的取值对聚类质量有重大影响。
- PAM算法对于小的数据集非常有效(例如100个对象聚成5类),但对于大数据集其效率不高。因为在替换中心点时每个点的替换代价都可能计算,因此,当$n$和$k$的值较大时,这样的计算代价相当高。

### 5.2.3 其他方法

PAM算法对小数据集非常有效,但对大的数据集合没有良好的可伸缩性。CLARA(Cluster LARger Application)是基于$k$-中心点类型的算法,能处理更大的数据集合。与PAM算法不同,CLARA不是直接在给定的数据集合中寻找最佳的$k$个中心点,而是先抽取数据集合的多个样本,然后用PAM方法在抽取的样本中寻找最佳的$k$个中心点,返回最好的聚类结果作为输出。如果样本是以非常随机的方式选取的,则足以代表原来的数据集合,这样从样本中选出的中心点很可能和整个数据集合中选出的非常近似。如果任何取样得到的中心点不属于最佳的中心点,CLARA就不能得到最佳的聚类结果。也就是说,如果样本发生偏斜,基于样本的一个好的聚类不一定代表了整个数据集合的一个好的聚类。所以对CLARA来说,最初的取样过程非常重要。但CLARA能处理比PAM更大的数据集合。每步迭代的复杂度是$O(ks^2+k(n-k))$,$s$是样本的大小,$k$是簇的数目,而$n$是所有对象的总数。可见CLARA的有效性取决于样本的大小。

CLARANS(Cluster Larger Application based upon RANdomized Search,随机搜索聚类算法)是另外一种$k$-中心点方法,它将采样技术和PAM结合起来,对CLARA的聚类质量和可伸缩性进行了改进。不像CLARA那样每个阶段选取一个固定样本,CLARANS在搜索的每一步都带一定随机性地选取一个样本。聚类过程可以描述为一个图的搜索,图中的每个结点都是一个潜在的解。在替换了一个中心点后得到的聚类结果称为当前聚类结果的邻居。如果找到一个比它更好的邻居,也就是说它有更小的平方-误差值,CLARANS移到该邻居结点,处理过程重新开始。否则,当前的聚类达到了一个局部最优。如果找到了一个局部最优,CLARANS再随机选择一个点来寻找另一个局部最优。该算法的计算复杂度大约是$O(n^2)$,$n$是对象的数目。CLARANS聚类的质量取决于所用的抽样方法。此外,该算法能检测到孤立点。

## 5.3 层次聚类方法

层次聚类方法对给定的数据集进行层次的分解,直到某种条件满足为止。具体又可分为凝聚的、分裂的两种方案。

- 凝聚的层次聚类是一种自底向上的策略,首先将每个对象作为一个簇,然后合并

这些原子簇为越来越大的簇，直到所有的对象都在一个簇中，或者某个终结条件被满足，绝大多数层次聚类方法属于这一类，它们只是在簇间相似度的定义上有所不同。

- 分裂的层次聚类与凝聚的层次聚类相反，采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到每个对象自成一簇，或者达到了某个终结条件。

层次凝聚的代表是 AGNES 算法。层次分裂的代表是 DIANA 算法。

## 5.3.1　AGNES 算法

AGNES(AGglomerative NESting)算法是凝聚的层次聚类方法。AGNES 算法最初将每个对象作为一个簇，然后这些簇根据某些准则被一步步地合并。例如，如果簇 $C_1$ 中的一个对象和簇 $C_2$ 中的一个对象之间的距离是所有属于不同簇的对象间欧氏距离中最小的，$C_1$ 和 $C_2$ 可能被合并。这是一种单链接方法，其每个簇可以被簇中所有对象代表，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的对象最终合并形成一个簇。在聚类中，用户能定义希望得到的簇数目作为一个结束条件。

**1. 算法描述**

**算法 5-3**　AGNES(自底向上凝聚算法)

输入：包含 $n$ 个对象的数据库，终止条件簇的数目 $k$。

输出：$k$ 个簇，达到终止条件规定簇数目。

(1) 将每个对象当成一个初始簇；
(2) REPEAT
(3)　根据两个簇中最近的数据点找到最近的两个簇；
(4)　合并两个簇，生成新的簇的集合；
(5) UNTIL 达到定义的簇的数目；

**2. 算法执行例子**

**例子 5-3**　下面给出一个样本事务数据库(见表 5-4)，并对它实施 AGNES 算法。

**表 5-4　样本事务数据库**

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

在所给的数据集上运行 AGNES 算法，表 5-5 为算法的步骤(设 $n=8$，用户输入的终止条件为两个簇)。初始簇{1}，{2}，{3}，{4}，{5}，{6}，{7}，{8}。

表 5-5 执行过程

| 步骤 | 最近的簇距离 | 最近的两个簇 | 合并后的新簇 |
|---|---|---|---|
| 1 | 1 | {1},{2} | {1,2},{3},{4},{5},{6},{7},{8} |
| 2 | 1 | {3},{4} | {1,2},{3,4},{5},{6},{7},{8} |
| 3 | 1 | {5},{6} | {1,2},{3,4},{5,6},{7},{8} |
| 4 | 1 | {7},{8} | {1,2},{3,4},{5,6},{7,8} |
| 5 | 1 | {1,2},{3,4} | {1,2,3,4},{5,6},{7,8} |
| 6 | 1 | {5,6},{7,8} | {1,2,3,4},{5,6,7,8}结束 |

在第 1 步中,根据初始簇计算每个簇之间的距离,随机找出距离最小的两个簇,进行合并,最小距离为 1,合并后 1、2 点合并为一个簇。

在第 2 步中,对上一次合并后的簇计算簇间距离,找出距离最近的两个簇进行合并,合并后 3、4 点成为一簇。

在第 3 步中,重复第 2 步的工作,5、6 点成为一簇。

在第 4 步中,重复第 2 步的工作,7、8 点成为一簇。

在第 5 步中,合并{1,2},{3,4}成为一个包含四个点的簇。

在第 6 步中,合并{5,6},{7,8},由于合并后的簇的数目已经达到了用户输入的终止条件,程序结束。

**3. 算法的性能分析**

AGNES 算法比较简单,但经常会遇到合并点选择的困难。这样的决定是非常关键的,因为一旦一组对象被合并,下一步的处理将在新生成的簇上进行。已做的处理不能撤销,聚类之间也不能交换对象。如果在某一步没有很好地合并选择,可能会导致低质量的聚类结果。而且,这种聚类方法不具有很好的可伸缩性,因为合并的决定需要检查和估算大量的对象或簇。

这种算法的复杂度到底是多大呢?假定在开始的时候有 $n$ 个簇,在结束的时候有 1 个簇,那么在主循环中有 $n$ 次迭代,在第 $i$ 次迭代中,我们必须在 $n-i+1$ 个簇中找到最靠近的两个聚类。另外,算法必须计算所有对象两两之间的距离,因此这个算法的复杂度为 $O(n^2)$,该算法对于 $n$ 很大的情况是不适用的。

## 5.3.2 DIANA 算法

DIANA(Divisive ANAlysis)算法属于分裂的层次聚类。与凝聚的层次聚类相反,它采用一种自顶向下的策略,它首先将所有对象置于一个簇中,然后逐渐细分为越来越小的簇,直到每个对象自成一簇,或者达到了某个终结条件,例如达到了某个希望的簇数目,或者两个最近簇之间的距离超过了某个阈值。

在 DIANA 方法的处理过程中,所有的对象初始都放在一个簇中。根据一些原则(如簇中最临近对象的最大欧氏距离),将该簇分裂。簇的分裂过程反复进行,直到最终每个新的簇只包含一个对象。

在聚类中,用户能定义希望得到的簇数目作为一个结束条件。同时,它使用下面两种测度方法。

- 簇的直径：在一个簇中的任意两个数据点都有一个欧氏距离，这些距离中的最大值是簇的直径。
- 平均相异度（平均距离）：

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

### 1. 算法描述

**算法 5-4**　DIANA（自顶向下分裂算法）

输入：包含 $n$ 个对象的数据库，终止条件簇的数目 $k$。

输出：$k$ 个簇，达到终止条件规定簇数目。

(1) 将所有对象整个当成一个初始簇；
(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；
(4)　找出所挑中簇里与其他点平均相异度最大的一个点放入 splinter group，剩余的放入 old party 中；
(5)　REPEAT
(6)　在 old party 里找出到 splinter group 中点的最近距离不大于到 old party 中点的最近距离的点，并将该点加入 splinter group；
(7)　UNTIL 没有新的 old party 的点分配给 splinter group；
(8)　splinter group 和 old party 为被选中的簇分裂成的两个簇，与其他簇一起组成新的簇集合。
(9) END;

### 2. 算法执行例子

**例子 5-4**　下面给出一个样本事务数据库（见表 5-6），并对它实施 AGNES 算法。

**表 5-6　样本事务数据库**

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

对所给的数据进行 DIANA 算法，表 5-7 为算法的步骤（设 $n=8$，用户输入的终止条件为两个簇）。初始簇为{1,2,3,4,5,6,7,8}。

**表 5-7　执行过程**

| 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 |

第1步,找到具有最大直径的簇,对簇中的每个点计算平均相异度(假定采用的是欧式距离)。

- 1的平均距离:(1+1+1.414+3.6+4.24+4.47+5)/7=2.96。
- 2的平均距离:(1+1.414+1+2.828+3.6+3.6+4.24)/7=2.526。
- 3的平均距离:(1+1.414+1+3.16+4.12+3.6+4.47)/7=2.68。
- 4的平均距离:(1.414+1+1+2.24+3.16+2.828+3.6)/7=2.18。
- 5的平均距离:2.18。
- 6的平均距离:2.68。
- 7的平均距离:2.526。
- 8的平均距离:2.96。

这时挑出平均相异度最大的点1放到splinter group中,剩余点在old party中。

第2步,在old party里找出到最近的splinter group中的点的距离不大于到old party中最近的点的距离的点,将该点放入splinter group中,该点是2。

第3步,重复第2步的工作,在splinter group中放入点3。

第4步,重复第2步的工作,在splinter group中放入点4。

第5步,没有新的old party中的点分配给splinter group,此时分裂的簇数为2,达到终止条件。如果没有到终止条件,下一阶段还会从分裂好的簇中选一个直径最大的簇按刚才的分裂方法继续分裂。

**3. 算法的性能分析**

层次聚类方法的缺点是已做的分裂操作不能撤销,类之间不能交换对象。如果在某步没有选择好分裂点,可能会导致低质量的聚类结果。而且,这种聚类方法不具有很好的可伸缩性,因为分裂的决定需要检查和估算大量的对象或簇。

### 5.3.3 其他聚类方法

层次聚类方法尽管简单,但经常会遇到合并或分裂点的选择的困难。改进层次方法的聚类质量的一个有希望的方向是将层次聚类和其他聚类技术进行集成,形成多阶段聚类。下面我们介绍两个改进的层次聚类方法BIRCH和CURE。

**1. BIRCH**

BIRCH(利用层次方法的平衡迭代归约和聚类)是一个综合的层次聚类方法,它用聚类特征和聚类特征树(CF)来概括聚类描述。该算法通过聚类特征可以方便地进行中心、半径、直径及类内、类间距离的运算。CF树是一个具有两个参数分支因子B和阈值T的高度平衡树,它存储了层次聚类的聚类特征。分支因子定义了每个非叶结点的最大数目,而阈值给出了存储在树的叶子结点中的子聚类的最大直径。

BIRCH算法的工作过程包括两个阶段。

- 阶段一:BIRCH扫描数据库,建立一个初始存放于内存的CF树,它可以被看作数据的多层压缩,试图保留数据内在的聚类结构。随着对象的插入,CF树被动态

地构造，不要求所有的数据读入内存，而可在外存上逐个读入数据项。因此，BIRCH 方法对增量或动态聚类也非常有效。

- 阶段二：BIRCH 采用某个聚类算法对 CF 树的叶结点进行聚类。在这个阶段可以执行任何聚类算法，例如典型的划分方法。

BIRCH 算法试图利用可用的资源来生成最好的聚类结果。通过一次扫描就可以进行较好的聚类，故该算法的计算复杂度是 $O(n)$，$n$ 是对象的数目。BIRCH 算法在大型数据库中可以取得高的速度和伸缩性。

**2. CURE**

很多聚类算法只擅长处理球形或相似大小的聚类，另外有些聚类算法对孤立点比较敏感。CURE 算法解决了上述两方面的问题，选择基于质心和基于代表对象方法之间的中间策略，即选择空间中固定数目的具有代表性的点，而不是用单个中心或对象来代表一个簇。该算法首先把每个数据点看成一簇，然后再以一个特定的收缩因子向簇中心“收缩”它们，即合并两个距离最近的代表点的簇。

CURE 算法采用随机取样和划分两种方法的组合，具体步骤如下：

- 从源数据集中抽取一个随机样本。
- 为了加速聚类，把样本划分成 $p$ 份，每份大小相等。
- 对每个划分局部地聚类。
- 根据局部聚类结果，对随机取样进行孤立点剔除。主要有两种措施：如果一个簇增长得太慢，就去掉它。在聚类结束的时候，非常小的类被剔除。
- 对上一步中产生的局部的簇进一步聚类。落在每个新形成的簇中的代表点根据用户定义的一个收缩因子 $\alpha$ 收缩或向簇中心移动。这些点代表和捕捉到了簇的形状。
- 用相应的簇标签来标记数据。

由于它回避了用所有点或单个质心来表示一个簇的传统方法，将一个簇用多个代表点来表示，使 CURE 可以适应非球形的几何形状。另外，收缩因子降低了噪声对聚类的影响，从而使 CURE 对孤立点的处理更加健壮，而且能识别非球形和大小变化比较大的簇。CURE 的复杂度是 $O(n)$，$n$ 是对象的数目，所以该算法适合大型数据的聚类。

## 5.4　密度聚类方法

密度聚类方法的指导思想是，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聚类中去。这类算法能克服基于距离的算法只能发现“类圆形”聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。但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，且对数据维数的伸缩性较差。这类方法需要扫描整个数据库，每个数据对象都可能引起一次查询，因此当数据量大时会造成频繁的 I/O 操作。代表算法有 DBSCAN、OPTICS、DENCLUE 算法等。

## DBSCAN算法

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同,它将簇定义为密度相连的点的最大集合,能够把具有足够高密度的区域划分为簇,并可在有"噪声"的空间数据库中发现任意形状的聚类。

下面首先介绍关于密度聚类涉及的一些定义。

**定义 5-3** 对象的ε-邻域:给定对象在半径ε内的区域。

**定义 5-4** 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。

例如,在图 5-6 中,ε=1cm,MinPts=5,$q$ 是一个核心对象。

**定义 5-5** 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,我们说对象 $p$ 从对象 $q$ 出发是直接密度可达的。

例如,在图 5-6 中,ε=1cm,MinPts=5,$q$ 是一个核心对象,对象 $p$ 从对象 $q$ 出发是直接密度可达的。

**定义 5-6** 密度可达的:如果存在一个对象链 $p_1, p_2, \cdots, p_n, p_1=q, p_n=p$,对 $p_i \in D$,$1 \leqslant i \leqslant n$,$p_i+1$ 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。

例如,在图 5-7 中,ε=1cm,MinPts=5,$q$ 是一个核心对象,$p_1$ 是从 $q$ 关于ε和 MitPts 直接密度可达;由于 $p$ 是从 $p_1$ 关于ε和 MitPts 直接密度可达,所以对象 $p$ 是从对象 $q$ 关于ε和 MinPts 密度可达的。

**定义 5-7** 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,如图 5-8 所示。

**定义 5-8** 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",如图 5-9 所示。

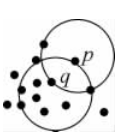



图 5-6 直接密度可达

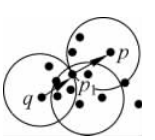



图 5-7 密度可达

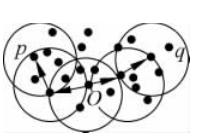



图 5-8 密度相连

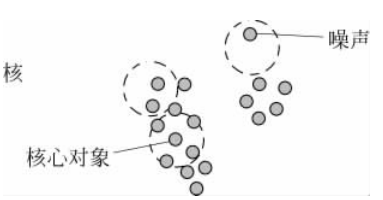



图 5-9 噪声

DBSCAN 通过检查数据集中每个对象的ε-邻域来寻找聚类。如果一个点 $p$ 的ε-邻域包含多于 MinPts 个对象,则创建一个 $p$ 作为核心对象的新簇。然后,DBSCAN 反复地

寻找从这些核心对象直接密度可达的对象，这个过程可能涉及一些密度可达簇的合并。当没有新的点可以被添加到任何簇时，该过程结束。

### 1. DBSCAN 算法描述

**算法 5-5**　DBSCAN

输入：包含 $n$ 个对象的数据库，半径 $\varepsilon$，最少数目 MinPts。

输出：所有生成的簇，达到密度要求。

(1) REPEAT
(2)　从数据库中抽取一个未处理过的点；
(3)　IF 抽出的点是核心点 THEN 找出所有从该点密度可达的对象，形成一个簇；
(4)　ELSE 抽出的点是边缘点(非核心对象)，跳出本次循环，寻找下一点；
(5) UNTIL 所有点都被处理；

### 2. 算法执行例子

**例子 5-5**　下面给出一个样本事务数据库(见表 5-8)，并对它实施 DBSCAN 算法。

**表 5-8　样本事务数据库**

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

对所给的数据进行 DBSCAN 算法，表 5-9 给出了算法执行过程示意(设 $n=12$，用户输入 $\varepsilon=1$，MinPts=4)。

**表 5-9　DBSCAN 算法执行过程示意**

| 步骤 | 选择的点 | 在 $\varepsilon$ 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{1,3,4,5,9,10,12},{2,6,7,8,11}。具体过程如下:

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(小于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4个,间接可达3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(所有点都已处理),程序终止。

### 3. 算法的性能分析

DBSCAN需要对数据集中的每个对象进行考察,通过检查每个点的ε-邻域来寻找聚类,如果某个点 $p$ 为核心对象,则创建一个以该点 $p$ 为核心对象的新簇,然后寻找从核心对象直接密度可达的对象。如表5-10所示,如果采用空间索引,DBSCAN的计算复杂度是 $O(n\log n)$,这里 $n$ 是数据库中对象的数目。否则,计算复杂度是 $O(n^2)$。

**表5-10 DBSCAN的性能**

| 时间复杂度 | 一次邻居点的查询 | DBSCAN |
|---|---|---|
| 无索引 | $O(n)$ | $O(n^2)$ |
| 有索引 | $\log n$ | $n\log n$ |

DBSCAN算法将具有足够高密度的区域划分为簇,并可以在带有“噪声”的空间数据库中发现任意形状的聚类。

但是,该算法对用户定义的参数是敏感的,ε、MinPts的设置将影响聚类的效果。设置的细微不同,会导致聚类结果的很大差别。为了解决上述问题,OPTICS(Ordering Points To Identify the Clustering Structure)被提出,它通过引入核心距离和可达距离,使得聚类算法对输入的参数不敏感。

## 5.5　其他聚类方法

前面主要介绍了常用的几种聚类方法，包括划分聚类方法、层次聚类方法和密度聚类方法。

划分聚类方法通过评价函数把数据集分割成 $k$ 个部分，主要有两种类型：$k$-平均和 $k$-中心点。$k$-平均在处理大数据集方面非常有效，特别是对于数值属性。$k$-中心点算法消除了 $k$-平均算法对于孤立点的敏感性。PAM 是最早提出的 $k$-中心点算法之一，其计算复杂度 $O(n(n-k)^2)$。PAM 算法对小数据集非常有效，但对大的数据集合没有良好的可伸缩性。CLARA 能处理更大的数据集合，复杂度是 $O(ks^2+k(n-k))$，$s$ 是样本的大小，$k$ 是簇的数目，而 $n$ 是所有对象的总数。CLASNS 采用了随机搜索改进了 $k$-中心点算法，对 CLARA 的聚类质量和可伸缩性进行了改进，计算复杂度大约是 $O(n^2)$。

层次的方法对给定数据对象集合进行层次的分界。根据层次分界的形成方式，层次的方法又可以分为凝聚的和分裂的。5.3 节介绍了两种简单的层次聚类方法 AGNES 和 DIANA 算法。这两种算法的缺点是已做的合并或分裂操作不能撤销，类之间不能交换对象，如果在某步没有选择好合并或分裂点，可能会导致低质量的聚类结果。而且，这两种聚类方法不具有很好的可伸缩性，因为合并或分裂的决定需要检查和估算大量的对象或簇。一个有望改进层次方法的聚类质量的方向是将层次聚类和其他聚类技术进行集成，形成多阶段聚类。接着介绍两个改进的层次聚类方法 CURE 和 BIRCH。BIRCH 用聚类特征和聚类特征树(CF)来概括聚类描述，通过一次扫描就可以进行较好的聚类，计算复杂度是 $O(n)$，在大型数据库中取得了高的速度和伸缩性。CURE 对孤立点的处理更加健壮，而且能识别非球形和大小变化比较大的簇。CURE 的复杂度是 $O(n)$。

密度聚类方法利用数据密度函数进行聚类，克服了基于距离的算法只能发现“类圆形”的聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。DBSCAN 是一个比较有代表性的基于密度的聚类算法，当采用空间索引时，该算法的复杂度为 $O(n\log n)$。

下面我们简单介绍一些其他聚类方法例如网格聚类方法 STING、基于模型的聚类方法 SOM 和 COBWEB、概念聚类方法和模糊聚类方法。

### 5.5.1　STING 算法

STING(Statistical Information Grid_based Method)是一种基于网格的多分辨率聚类技术，它将空间区域划分为矩形单元。针对不同级别的分辨率，通常存在多个级别的矩形单元，这些单元形成了一个层次结构：高层的每个单元被划分为多个第一层的单元。高层单元的统计参数可以很容易地从底层单元的计算得到。这些参数包括属性无关的参数 count、属性相关的参数 m(平均值)、s(标准偏差)、min(最小值)、max(最大值)以及该单元中属性值遵循的分布类型。STING 算法中由于存储在每个单元中的统计信息提供了单元中的数据不依赖于查询的汇总信息，因而计算是独立于查询的。

STING 算法采用了一种多分辨率的方法来进行聚类分析，该聚类算法的质量取决于网格结构最底层的粒度。如果粒度比较细，处理的代价会显著增加；但如果粒度较粗，则

聚类质量会受到影响。

STING算法的主要优点是效率高,通过对数据集的一次扫描来计算单元的统计信息,因此产生聚类的时间复杂度是$O(n)$。在建立层次结构以后,查询的时间复杂度是$O(g)$,$g$远小于$n$。此外,STING算法并行采用网格结构,有利于并行处理和增量更新。

## 5.5.2 SOM算法

SOM神经网络是一种基于模型的聚类方法。SOM神经网络由输入层和竞争层组成。输入层由$N$个输入神经元组成,竞争层由$m\times m=M$个输出神经元组成,且形成一个二维平面阵列。输入层各神经元与竞争层各神经元之间实现全互连接。该网络根据其学习规则,通过对输入模式的反复学习,捕捉住各个输入模式中所含的模式特征,并对其进行自组织,在竞争层将聚类结果表现出来,进行自动聚类。竞争层的任何一个神经元都可以代表聚类结果。如图5-10给出了SOM神经网络基本结构,图5-11给出了结构中各输入神经元与竞争层神经元$j$的连接情况。

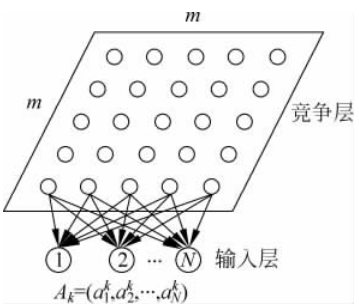



图5-10 SOM网络基本结构

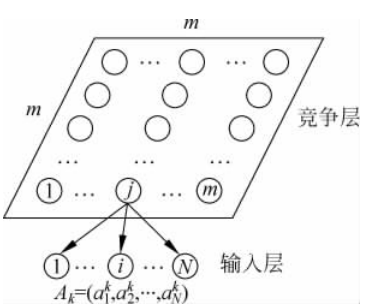



图5-11 输入神经元与竞争层神经元$j$的连接情况

设网络的输入模式为$A_k=(a_1^k,a_2^k,\cdots,a_N^k)$,$k=1,2,\cdots,p$;竞争层神经元向量为$\boldsymbol{B}_j=(b_{j1},b_{j2},\cdots,b_{jm})$,$j=1,2,\cdots,m$;其中$A_k$为连续值,$\boldsymbol{B}_j$为数字量。网络的连接权为$\{w_{ij}\}$,$i=1,2,\cdots,N$;$j=1,2,\cdots,M$。

SOM网络寻找与输入模式$A_k$最接近的连接权向量$\boldsymbol{W}_g=(w_{g1},w_{g2},\cdots,w_{gN})$,将该连接权向量$\boldsymbol{W}_g$进一步朝与输入模式$A_k$接近的方向调整,而且还调整邻域内的各个连接权向量$\boldsymbol{W}_j$,$j\in N_g(t)$。随着学习次数的增加,邻域逐渐缩小,最终得到聚类结果。

SOM类似于大脑的信息处理过程,对二维或三维数据的可视是非常有效的。SOM网络的最大局限性是,当学习模式较少时,网络的聚类效果取决于输入模式的先后顺序,且网络连接权向量的初始状态对网络的收敛性能有很大影响。

## 5.5.3 COBWEB算法

概念聚类是机器学习中的一种聚类方法,大多数概念聚类方法采用了统计学的途径,在决定概念或聚类时使用概率度量。COBWEB(简单增量概念聚类算法)以一个分类树的形式创建层次聚类,它的输入对象用分类属性-值对来描述。

分类树和判定树不同。分类树中的每个结点对应一个概念，包含该概念的一个概率描述，概述被分在该结点下的对象。概率描述包括概念的概率和形如 $P(A_i=V_{ij}|C_k)$ 条件概率，这里 $A_i=V_{ij}$ 是属性-值对，$C_k$ 是概念类。在分类树某层次上的兄弟结点形成了一个划分。

COBWEB 采用了一个启发式估算度量——分类效用来指导树的构建。分类效用定义如下：

$$C_f=\frac{\sum_{k=1}^{n}P(C_k)\left[\sum_i\sum_j P(A_i=V_{ij}\mid C_k)^2-\sum_i\sum_j P(A_i=V_{ij})^2\right]}{n}$$

其中 $n$ 是在树的某个层次上形成一个划分$\{C_1,C_2,\cdots,C_n\}$的结点、概念或“种类”的数目。

分类效用回报类内相似性和类间相异性进行评定：

- 概率 $P(A_i=V_{ij}|C_k)$表示类内相似性。该值越大，共享该“属性-值”对的类成员比例就越大，就更能预见该“属性-值”对是类成员。
- 概率 $P(C_k|A_i=V_{ij})$表示类间相异性。该值越大，在对照类中的对象共享该“属性-值”对就越少，便更能预见该“属性-值”对是类成员。

### 5.5.4　模糊聚类算法 FCM

前面介绍的几种聚类算法可以导出确定的聚类，也就是说，一个数据点或者属于一个类，或者不属于一个类，而不存在重叠的情况。可以称这些聚类方法为“确定性分类”。在一些没有确定支持的情况中，聚类可以引入模糊逻辑概念。对于模糊集来说，一个数据点都是以一定程度属于某个类，也可以同时以不同的程度属于几个类。常用的模糊聚类算法是模糊 $C$ 平均值 FCM(Fuzzy $C$-Means)算法。该算法是在传统 $C$ 均值算法中应用了模糊技术。

## 5.6　本章小结和文献注释

聚类在数据挖掘中是一项非常重要的任务，是本书重点介绍内容之一。本章对聚类简单概括，然后重点介绍了划分聚类方法、层次聚类方法和密度聚类方法，最后对其他方法进行了简要介绍。

本章的主要内容及相关的文献引用情况如下所述。

**1. 概述**

关于聚类介绍方面的文献很多，[KR90]、[JMF99]对聚类作了全面的综述。[HK00]对聚类分析的概念、聚类中的数据类型以及主要的聚类方法进行了简要概述。[邵峰晶 03] 对聚类中采用的各种距离函数进行了全面的介绍。另外，[NH94]对空间数据挖掘中的聚类方法进行了综述。

5.1.1 节和 5.1.2 节主要引用了文献[HK00]和[张云涛 04]。5.1.3 节主要参考了文献[邵峰晶 03]。

**2. 划分聚类方法**

划分聚类方法方面的文献比较多。[ARS98]和[BF98b]探讨了$k$-平均方法,[BF98b]对$k$-平均初始点的确定进行了详细讨论。[Hua98]提出了$k$-平均的改进算法$k$-模和$k$-原型。

文献[KR90]提出了$k$-中心点算法PAM,文献[NH94]比较了CLARANS与CLARA及PAM算法,对这些算法感兴趣的读者可参阅此文献。

本章在划分聚类方法的定义方面主要引用了文献[HK00],在评价函数方面引用了文献[张银奎03],在PAM算法、CLARA以及CLARANS算法介绍中引用了文献[HK00]和[Dun03]。

**3. 层次聚类方法**

文献[KR90]提出了凝聚的层次聚类算法AGNES和分裂的层次聚类算法。

文献[ZRL96]提出的BIRCH算法,文献[GRS98]提出的CURE算法。

文献[Guh99]提出了ROCK算法,文献[KHK99]提出了CHAMELEON算法等。

另外,文献[HK00]对上述层次聚类算法做了详尽的介绍。本章在上述算法的介绍中主要引用了文献[KR90]和[HK00]。

**4. 密度聚类方法**

文献[EK+96]最先提出了密度聚类方法DBSCAN,为密度聚类的研究奠定了基础。但是,算法对用户定义的参数是敏感的。文献[AB+99]提出的OPTICS通过引入核心距离和可达距离,解决了DBSCAN存在的问题。

本章在上述算法的介绍中主要引用了文献[HK00]。

**5. 其他聚类方法**

针对不同的应用需求许多其他聚类方法被提出,例如基于模型的聚类方法和基于网格的聚类方法。

基于网格的聚类方法STING由文献[WYM97]提出,它是一种基于网格的多分辨率聚类方法,它将空间区域划分为矩形,从不同的层次实现数据分析。文献[SCZ98]提出了一种密度与网格方法相结合的多分辨率聚类方法WaveCluster,它采用小波实现空间的转换,在变换后的空间中找到高密度的区域。CLIQUE是另外一种综合密度与网格的聚类方法,可以实现高维空间的聚类,感兴趣的读者请参阅文献[AG+98]。

基于模型的聚类方法包括统计方法和神经网络方法。统计的模型聚类方法有[Fis87]提出的概念聚类方法COBWEB,文献[CS96a]提出的Autoclass。神经网络方法的典型代表是文献[Koh82]提出的SOM。

此外,文献[KR90]介绍了模糊聚类方法,并对模糊聚类与一般聚类方法进行了比较。

# 习　题　5

1. 简单地描述下列英文短语的含义。

(1) Partitioning method

(2) Hierarchical method

(3) Density-based method

(4) Grid-based method

2. 简单地描述下列英文缩写的含义。

(1) PAM

(2) STING

(3) DBSCAN

3. 简述聚类的基本概念。

4. “物以类聚，人以群分”，请举例说明聚类的基本概念。

5. 聚类分析具有重要的作用，简述聚类分析在数据挖掘中的应用。

6. 举例说明聚类分析的用途。

7. 你认为一个好的聚类算法应该具备哪些特性？

8. 简述基于距离的聚类算法的主要特点。

9. 在对数据进行聚类的时候，会遇到二元特征样本，简述对二元特征样本进行距离度量的主要方法。

10. 哪种聚类算法对噪声数据不明显，可以发现不规则的类？

11. 给定两个对象，分别用元组(22,1,42,10),(20,0,36,8)表示。(a)计算两个对象之间的欧氏距离。(b)计算两个对象之间的绝对距离。

12. 请说出在聚类分析中常用的距离度量方法。

13. 简述划分聚类方法的主要思想。

14. 请说出划分聚类与层次聚类的主要特点。

15. 请用 $k$-平均算法把表 A5-1 中的 8 个点聚为 3 个簇，假设第一次迭代选择序号 1、序号 4 和序号 7 当作初始点，请给出第一次执行后的 3 个聚类中心以及最后的 3 个簇。

**表 A5-1　样本数据 1**

| 序号 | 属性 1 | 属性 2 | 序号 | 属性 1 | 属性 2 |
|---|---|---|---|---|---|
| 1 | 2 | 10 | 5 | 7 | 5 |
| 2 | 2 | 5 | 6 | 6 | 4 |
| 3 | 8 | 4 | 7 | 1 | 2 |
| 4 | 5 | 8 | 8 | 4 | 9 |

16. 举例说明 $k$-平均算法的主要思想。

17. 请说出 $k$-平均算法的优点和缺点。

18. 试比较 $k$-平均算法与 $k$-中心点算法的特点。

19. 简述 $k$-中心点算法的主要思路。

20. 简述 PAM 算法的主要步骤。

21. 简述凝聚的层次聚类方法的主要思路。

22. 在表 A5-2 中给定的样本上运行 AGNES 算法,假定算法的终止条件为 3 个簇,初始簇{1},{2},{3},{4},{5},{6},{7},{8}。

**表 A5-2　样本数据 2**

| 序号 | 属性 1 | 属性 2 | 序号 | 属性 1 | 属性 2 |
|---|---|---|---|---|---|
| 1 | 2 | 10 | 5 | 7 | 5 |
| 2 | 2 | 5 | 6 | 6 | 4 |
| 3 | 8 | 4 | 7 | 1 | 2 |
| 4 | 5 | 8 | 8 | 4 | 9 |

23. 在表 A5-3 中给定的样本上运行 DIANA 算法,假定算法的终止条件为 3 个簇,初始簇{1,2,3,4,5,6,7,8}。

**表 A5-3　样本数据 3**

| 序号 | 属性 1 | 属性 2 | 序号 | 属性 1 | 属性 2 |
|---|---|---|---|---|---|
| 1 | 2 | 10 | 5 | 7 | 5 |
| 2 | 2 | 5 | 6 | 6 | 4 |
| 3 | 8 | 4 | 7 | 1 | 2 |
| 4 | 5 | 8 | 8 | 4 | 9 |

24. 请分析 DIANA 和 AGNES 算法的特点。

25. 简述密度聚类方法的主要思路。

26. 请举例说明 DBSCAN 算法的主要思想。

27. 简述 STING 算法的主要特点。

28. 简述聚类挖掘与分类挖掘的联系与区别。