第六章　聚类分析

聚类（clustering）是一个将数据集划分为若干组（class）或类（cluster）的过程，并使得同一个组内的数据对象具有较高的相似度；而不同组中的数据对象是不相似的。相似或不相似的描述是基于数据描述属性的取值来确定的。通常就是利用（各对象间）距离来进行表示的。许多领域，包括数据挖掘、统计学和机器学习都有聚类研究和应用。

本章将要介绍对大量数据进行聚类分析的有关方法；同时也还将介绍如何根据数据对象的属性来计算各数据对象之间的距离（不同）。有关的聚类方法（类型）主要有：划分类方法、分层类方法、基于密度类方法、基于网格类方法和基于模型类方法。此外本章的最后将要介绍利用聚类方法进行异常数据（outlier）检测的有关内容。

6.1　聚类分析概念

将一组（set）物理的或抽象的对象，根据它们之间的相似程度，分为若干组（group）；其中相似的对象构成一组，这一过程就称为聚类过程（clustering）。一个聚类（cluster）就是由彼此相似的一组对象所构成的集合；不同聚类中对象是不相似的。就是从给定的数据集中搜索数据项（items）之间所存在的有价值联系。在许多应用，一个聚类中所有对象常常被当作一个对象来进行处理或分析等操作。

聚类分析是人类活动中的一个重要内容。早在儿童时期，一个人就是通过不断完善潜意识中的分类模式，来学会识别不同物体，如：狗和猫，或动物和植物等。聚类分析已被应用到许多领域，其中包括：模式识别、数据分析、图像处理和市场分析等。通过聚类，人可以辨认出空旷和拥挤的区域，进而发现整个的分布模式，以及数据属性之间所存在有价值的相关关系。

聚类分析的典型应用主要包括，在商业方面，聚类分析可以帮助市场人员发现顾客群中所存在的不同特征的组群；并可以利用购买模式来描述这些不同特征的顾客组群。在生物方面，聚类分析可以用来获取动物或植物所存在的层次结构（taxonomies），以及根据基因功能对其进行分类以获得对人群中所固有的结构更深入的了解。聚类还可以从地球观测数据库中帮助识别具有相似的土地使用情况的区域。此外还可以帮助分类识别互联网上的文档以便进行信息发现。作为数据挖掘的一项功能，聚类分析还可以作为一个单独使用的工具，来帮助分析数据的

分布、了解各数据类的特征、确定所感兴趣的数据类以便作进一步分析。当然聚类分析也可以作为其它算法（诸如：分类和定性归纳算法）的预处理步骤。

数据聚类分析是一个正在蓬勃发展的领域。聚类分析所涉及的领域包括：数据挖掘、统计学、机器学习、空间数据库技术、生物学和市场学等。由于各应用数据库所包含的数据量越来越大，聚类分析已成为数据挖掘研究中一个非常活跃的研究课题。

作为统计学的一个分支，聚类分析已有多年的研究历史，这些研究主要集中在基于距离的聚类分析方面。许多统计软件包，诸如：S-Plus、SPSS 和 SAS，都包含基于 *k*-均值、*k*-中心等其它许多聚类分析工具。

在机器学习中，聚类分析属于一种无（教师）监督的学习方法。与分类学习不同，无（教师）监督学习不依靠事先确定的数据类别，以及标有数据类别的学习训练样本集合。正因为如此，聚类分析又是一种通过观察学习方法（learning by observation），而不是示例学习（learning by example）。在概念聚类方法中，仅当一组对象可以由一个概念所描述时，这些对象方才能构成一个类。这与基于几何距离表示相似程度并进行聚类的传统聚类方法有所不同。概念聚类方法主要包含两部分内容：（1）发现适当的类；（2）根据每个类形成相应的特征描述，与在分类学习中的方法类似。无论如何最大程度地实现类中对象相似度最大，类间对象相似度最小是聚类分析的基本指导思想。

在数据挖掘中，大多数工作都集中在发现能够有效、高效地对大数据库进行聚类分析的方法上。相关的研究课题包括：聚类方法的可扩展性、复杂形状和复杂数据类型的聚类分析的有效高效性、高维聚类技术，以及混合数值属性与符号属性数据库中的聚类分析方法等。

聚类分析是一个富有挑战的研究领域，有关每一个应用都提出了一个自己独特的要求。以下就是对数据挖掘中的聚类分析的一些典型要求。

（1）可扩展性。许多聚类算法在小数据集（少于 200 个数据对象）时可以工作很好；但一个大数据库可能会包含数以百万的对象。利用采样方法进行聚类分析可能得到一个有偏差的结果，这时就需要可扩展的聚类分析算法。

（2）处理不同类型属性的能力。许多算法是针对基于区间的数值属性而设计的。但是有些应用需要对其它类型数据，如：二值类型、符号类型、顺序类型，或这些数据类型的组合。

（3）发现任意形状的聚类。许多聚类算法是根据欧氏距离和 Manhattan 距离来进行聚类的。基于这类距离的聚类方法一般只能发现具有类似大小和密度的圆形或球状聚类。而实际上一个聚类是可以具有任意形状的，因此设计出能够发现任意形状类集的聚类算法是非常重要的。

（4）需要（由用户）决定的输入参数最少。许多聚类算法需要用户输入聚类分析中所需要的一些参数（如：期望所获聚类的个数）。而聚类结果通常都与输入参数密切相关；而这些参数常常也很难决定，特别是包含高维对象的数据集。这不仅构成了用户的负担；也使得聚类质量难以控制。

（5）处理噪声数据的能力。大多数现实世界的数据库均包含异常数据、不明数据、数据丢失和噪声数据，有些聚类算法对这样的数据非常敏感并会导致获得质量较差的数据。

（6）对输入记录顺序不敏感。一些聚类算法对输入数据的顺序敏感，也就是不同的数据输入顺序会导致获得非常不同的结果。因此设计对输入数据顺序不敏感的聚类算法也是非常重要的。

（7）高维问题。一个数据库或一个数据仓库或许包含若干维或属性。许多聚类算法在处理低维数据时（仅包含二到三个维）时表现很好。人的视觉也可以帮助判断多至三维的数据聚类分析质量。然而设计对高维空间中的数据对象，特别是对高维空间稀疏和怪异分布的数据对象，能进行较好聚类分析的聚类算法已成为聚类研究中的一项挑战。

（8）基于约束的聚类。现实世界中的应用可能需要在各种约束之下进行聚类分析。假设需要在一个城市中确定一些新加油站的位置，就需要考虑诸如：城市中的河流、高速路，以及每个区域的客户需求等约束情况下居民住地的聚类分析。设计能够发现满足特定约束条件且具有较好聚类质量的聚类算法也是一个重要聚类研究任务。

（9）可解释性和可用。用户往往希望聚类结果是可理解的、可解释的，以及可用的。这就需要聚类分析要与特定的解释和应用联系在一起。因此研究一个应用的目标是如何影响聚类方法选择也是非常重要的。

了解上述的需求后，下面按照聚类分析的工作过程进行介绍。首先不同数据类型对聚类方法的影响；然后就介绍聚类分析的常用分类；并详细讲解其中的每一个聚类方法，包括：划分方法、层次方法、基于密度方法、基于网格方法和基于模型方法。此外还要介绍在高维空间和异常数据分析中的相关聚类算法。

6.2 聚类分析中的数据类型

本节将主要介绍聚类分析中常见的数据类型，以及在聚类分析之前时如何对它们进行预处理的。假设一个要进行聚类分析的数据集包含 n 个对象，这些对象可以是人、房屋、文件等。基于内存的聚类算法通常都采用以下两种数据结构：

（1）数据矩阵

数据矩阵是一个对象-属性结构。它是由 n 个对象组成，如：人；这些对象是利用 p 个属性来进行描述的，如：年龄、高度、重量等。数据矩阵采用关系表形式或 $n \times p$ 矩阵来表示，如式（6.1）所示。

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \tag{6.1}$$

（2）差异矩阵

差异矩阵是一个对象-对象结构。它存放所有 n 个对象彼此之间所形成的差异。它一般采用 $n \times n$ 矩阵来表示，如式（6.2）所示。

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \cdots & \cdots & \cdots & \cdots \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \tag{6.2}$$

其中 $d(i,j)$ 表示对象 i 和对象 j 之间的差异（或不相似程度）。通常 $d(i,j)$ 为一个非负数；当对象 i 和对象 j 非常相似或彼此“接近”时，该数值接近 0；该数值越大，就表示对象 i 和对象 j 越不相似。由于有 $d(i,j)=d(j,i)$ 且 $d(i,i)=0$，因此就有式（6.2）所示矩阵。本节都是基于差异计算进行讨论的。

数据矩阵通常又称为是双模式矩阵；而差异矩阵则称为是单模式矩阵。因为前者行和列分别表示不同的实体；而后者行和列则表示的是同一实体。许多聚类算法都是基于差异矩阵进行聚类分析的。如果数据是以数据矩阵形式给出的，那么就首先需要转换为差异矩阵，方可利用聚类算法进行处理。

以下将要讨论如何对采用间隔数值（interval-scaled）属性、二值属性、符号属性、顺序属性和比例属性（ratio-scaled），或者这些属性的组合进行处理，以计算出对象之间的差异值。利用数据差异值就可以对对象进行聚类分析了。

6.2.1 间隔数值属性

本小节将要介绍间隔数值属性和它的标准化过程。然后介绍根据这一属性计算对象之间差异值（不相似程度）的具体计算方法。这些计算方法包括：欧氏距离计算方法、Manhattan 距离计算方法和 Minkowski 距离计算方法。

间隔数值属性就是基本呈直线比例的连续测量值。典型的间隔数值有：重量、高度和温度等。

所采用的测量单位可能会对聚类分析产生影响。例如：将测量单位（对于高度属性）从米变为英尺，或（对于重量属性）从公斤变为英磅，都会导致不同的聚类结构。通常采用一个较小的单位表示一个属性会使得属性的取值范围变大，因此对聚类结构就有较大的影响。为帮助避免对属性测量单位的依赖，就需要对数据进行标准化。所谓标准化测量就是给所有属性相同的权值。这一做法在没有任何背景知识情况下是非常有用的。而在一些应用中，用户会有意识地赋予某些属性更大权值以突出其重要性。例如：在对候选篮球选手进行聚类分析时，可能就会给身高属性赋予更大的权值。

为了实现标准化测量，一种方法就是将初始测量值转换为无单位变量。给定一个属性（变量）f，可以利用以下计算公式对其进行标准化：

（1）计算绝对偏差均值s_f

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|) \tag{6.3}$$

其中$x_{1f}, x_{2f}, \ldots, x_{nf}$是变量$f$的$n$个测量值；$m_f$为变量$f$的均值，也就是$m_f = (x_{1f} + x_{2f} + \cdots + x_{nf})/2$。

（2）计算标准化测量（$z-$分值）

$$z_{if} = \frac{x_{if} - m_f}{s_f} \tag{6.4}$$

其中绝对偏差均值s_f要比标准偏差σ_f更为鲁棒（对含有噪声数据而言）。在计算绝对偏差均值时，对均值的偏差$|x_{if} - m_f|$没有进行平方运算，因此异常数据的作用被降低；还有一些关于针对分散数据更鲁棒的处理方法，如：中间值绝对偏差方法。但是利用绝对偏差均值的好处就是：异常数据（outlier）的$z-$分值不会变得太小，从而使得异常数据仍是可识别的。

在一些特定应用中，标准化方法或许有用，但不一定有用，因此只能由用户决定是否或如何使用标准化方法。标准化方法在第二章预处理方法中的规格化处理方法也有详细介绍。

在标准化之后，或在无需标准化的特定应用中，由间隔数值所描述对象之间的差异（或相似）程度可以通过计算相应两个对象之间距离来确定。最常用的距离计算公式就是欧氏距离（Euclidean distance），具体公式内容如下：

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2)} \tag{6.5}$$

其中$i = (x_{i1}, x_{i2}, \ldots, x_{ip})$；$j = (x_{j1}, x_{j2}, \ldots, x_{jp})$；它们分别表示一个$p-$维数据对象。

另一个常用的距离计算方法就是Manhattan距离，它的具体计算公式定义如

下：

$$d(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\cdots+|x_{ip}-x_{jp}| \quad (6.6)$$

欧氏距离和 Manhattan 距离均满足距离函数的有关数学性质（要求）：

- $d(i,j)\geq 0$，这表示对象之间距离为非负数的一个数值；
- $d(i,i)=0$；这表示对象自身之间距离为零；
- $d(i,j)=d(j,i)$；这表示对象之间距离是对称函数；
- $d(i,j)\leq d(i,h)+d(h,j)$；这表示对象自身之间距离满足“两边之和不小于第三边”的性质；若将两个对象之间距离用一条边来表示的话；其中 h 为第三个对象。

Minkowski 距离是欧式距离和 Manhattan 距离的一个推广，它的计算公式定义如下：

$$d(i,j)=(|x_{i1}-x_{j1}|^q+|x_{i2}-x_{j2}|^q+\cdots+|x_{ip}-x_{jp}|^q)^{1/q} \quad (6.7)$$

其中 q 为一个正整数；当 $q=1$ 时，它代表 Manhattan 距离计算公式；而当 $q=2$ 时，它代表欧氏距离计算公式。

若每个变量均可被赋予一个权值，以表示其所代表属性的重要性。那么带权的欧氏距离计算公式就是：

$$d(i,j)=\sqrt{w_1|x_{i1}-x_{j1}|^2+w_2|x_{i2}-x_{j2}|^2+\cdots+w_p|x_{ip}-x_{jp}|^2)} \quad (6.8)$$

同样，Manhattan 距离和 Minkowski 距离也可以引入权值进行计算。

6.2.2 二值属性

本节将要介绍如何计算采用对称或非对称二值属性（量）描述对象之间的差异（程度）。

一个二值变量仅取 0 或 1 值；其中 0 代表（变量所表示的）状态不存在；而 1 则代表相应的状态存在。给定变量 *smoker*，它描述了一个病人是否吸烟情况。如： *smoker* 为 1 就表示病人吸烟；而若 *smoker* 为 0，就表示病人不吸烟。如果按照间隔数值变量对二值变量进行处理，常常会导致错误的聚类分析结果产生。因此采用特定方法计算二值变量所描述对象间的差异（程度）是非常必要的。

一种差异计算方法就是根据二值数据计算差异矩阵。如果认为所有的二值变量的权值均相同，那么就能得到一个 2×2 条件表，如图-6.1 所示；表中 q 表示在对象 i 和对象 j 中均取 1 的二值变量个数； r 表示在对象 i 取 1 但在对象 j 中取 0 的二值变量个数； s 表示在对象 i 中取 0 而在对象 j 中取 1 的二值变量个数； t 则表示在对象 i 和对象 j 中均取 0 的二值变量个数。.二值变量的总个数为 p，那么

就有： $p = q + r + s + t$。

		对象j		
对象i		1	0	合计
	1	q	r	$q+r$
	0	s	t	$s+t$
	合计	$q+s$	$r+t$	p

图-6.1 二值属性条件表

如果一个二值变量取 0 或 1 所表示的内容同样重要，那么该二值变量就是对称的；如 *smoker* 就是对称变量，因为它究竟是用 0 还是用 1 来（编码）表示一个病人的确吸烟（状态）并不重要。同样的基于对称二值变量所计算相应的相似（或差异）性就称为是不变相似性（invariant similarity）；因为无论如何对相应二值变量进行编码并不影响到它们相似（或差异）性的计算结果。对于不变相似性（计算），最常用的描述对象 i 和对象 j 之间差异（程度）参数就是简单匹配相关系数，它的具体定义描述如公式（6.9）所示。

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{6.9}$$

如果一个二值变量取 0 或 1 所表示内容的重要性是不一样的，那么该二值变量就是非对称的；如一个疾病 *disease* 的测试结果可描述为 *positive* 或 *negative*。显然这两个测试（输出）结果的重要性是不一样的。通常将少见的情况用 1 来表示（如：HIV *positive*）；而将其它情况用 0 来表示（HIV *negative*）。给定两个非对称二值变量，如果它们认为取 1 值比取 0 值所表示情况更重要，那么这样的二值变量就可称为是单性的（好象只有一个状态）。而这种这种变量的相似性就称为是非变相似性（nonvariant similarity）。对于非变相似性（计算），最常用的描述对象 i 和对象 j 之间差异（程度）参数就是 Jaccard 相关系数，它的具体定义描述如公式（6.10）所示。

$$d(i,j) = \frac{r+s}{q+r+s} \tag{6.10}$$

若一个数据集中既包含对称二值变量，又包含非对称二值变量，那么就可以利用 6.2.4 小节所要介绍的计算公式进行处理。

示例 6.1：二值变量的差异性。假设一个病人记录表如表-6.1 所示；表中所描述的属性（变量）分别为 *name*、*gender*、*fever*、*cough*、*test-1*、*test-2*、*test-3* 和 *test-4*；其中 *name* 作为（病人）对象的标识； *gender*（性别）是一个对称二值变量。其它变量则均为非对称变量。

name	***gender***	***fever***	***cough***	***test-1***	***test-2***	***test-3***	***test-4***
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

表-6.1 一个包含许多二值属性的关系数据表示意描述

对于非对称属性（变量）值，可将其 Y 和 P 设为 1；N 设为 0。根据非对称变量计算不同对象（病人）间的距离（差异性），就可以利用 Jaccard 相关系数计算公式（6.10）进行，具体计算结果如下：

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33;$$

$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67;$$

$$d(Jim, Mary) = \frac{1+2}{1+1+2} = 0.75;$$

上述计算值表明：Jim 和 Mary，由于他们之间距离值（差异性）三个中最大，因此不太可能得的是相似的病；而 Jack 和 Mary，由于他们之间距离值（差异性）三个中是最小，因此可能得的就是相似的病。 ■

6.2.3 符号、顺序和比例数值属性

本节将要介绍如何计算采用符号、顺序和比例数值属性（变量）所描述对象之间的差异（程度）。

（1）符号变量

符号变量是二值变量的一个推广。符号变量可以对两个以上的状态进行描述。例如：地图颜色 *map_color* 变量就是一个符号变量；它可以表示五种状态，即红、绿、篮、粉红和黄色。

设一个符号变量所取状态个数为 M；其中的状态可以用字母、符号，或一个整数集合来表示，如 $1,2,\ldots,M$。这里的整数仅仅是为了方便数据处理而采用

的，并不表示任何顺序关系。

对于符号变量，最常用的计算对象 i 和对象 j 之间差异（程度）的方法就是简单匹配方法。它的具体定义描述如公式（6.11）所示。

$$d(i,j)=\frac{p-m}{p} \qquad (6.11)$$

其中 m 表示对象 i 和对象 j 中取同样状态的符号变量个数（匹配数）；p 为所有的符号变量个数。

为增强 m 的作用，可以给它赋予一定的权值；而对于拥有许多状态的符号变量，也可以相应赋予更大的权值。

通过为符号变量的每个状态创建一个新二值变量，能够将符号变量表示为非对称的二值变量。对于具有给定状态的一个对象，代表一个状态的二值变量置为 1；而其它的二值变量置为 0。例如：要用二值变量表示地图颜色 *map_color* 符号变量，就需要上面所介绍的五种颜色分别创建一个二值变量。而对一个颜色为黄色的对象，就要将代表黄色状态的二值变量设为 1；而将其它二值变量设为 0。采用这种（二值变量）表达方式的对象间差异（程度）就可以利用 6.2.2 小节所介绍的计算方法进行计算了。

（2）顺序变量

一个离散顺序变量与一个符号变量相似，不同的是（对应 M 个状态的）的 M 个顺序值是具有按照一定顺序含义的。顺序变量在描述无法用客观方法表示的主观质量评估时是非常有用的。例如：专业等级（描述）就是一个顺序变量；它是按照助教、讲师、副教授和教授的顺序进行排列的。一个连续顺序变量看上去就象一组未知范围的连续数据；但它的相对位置要比它的实际数值有意义的多。例如在足球比赛中，一个球队排列名次常常要比它的实际得分更为重要。顺序变量的数值常常是通过对间隔数值（变量）的离散化而获得的，也就是通过将取值范围分为有限个组而得到的。一个顺序变量可以映射到一个等级（rank）集合上。如：若一个顺序变量 f 包含 M_f 个状态，那么这些有序的状态就映射为 1,2,…, M_f 的等级。

在计算对象间差异程度时，顺序变量的处理方法与间隔数值变量的处理方法类似。假设变量 f 为一组描述 n 个对象顺序变量中的一个。涉及变量 f 的差异程度计算方法描述如下：

- 第 i 个对象的 f 变量值标记为 x_{if}，变量 f 有 M_f 个有序状态，可以利用等级 1,2,…, M_f 分别替换相应的 x_{if}，得到相应的 r_{if}，$r_{if}\in\{1,2,\ldots,M_f\}$；
- 由于每个顺序变量的状态个数可能不同。因此有必要将每个顺序变量的

取值范围映射到$[0-1]$区间，以便使每个变量的权值相同。可以通过将第i个对象中的第f个变量的r_{if}用以下所计算得到的值来替换:

$$z_{if}=\frac{r_{if}-1}{M_f-1} \tag{6.12}$$

- 这时可以利用 6.2.1 小节所介绍有关间隔数值变量的任一个距离计算公式，来计算用顺序变量描述的对象间距离；其中用z_{if}来替换第i个对象中的变量f值。

（3）比例数值变量

一个比例数值变量就在非线性尺度上所获得的正测量值，如：指数比例，就可以用以下公式近似描述:

$$Ae^{Bt} \text{ 或 } Ae^{-Bt} \tag{6.13}$$

其中A和B为正的常数。典型例子包括：细菌繁殖增长的数目描述，或放射元素的衰减。

在计算比例数值变量所描述对象间距离时，有三种方法处理比例数值变量的方法。它们是:

- 将比例数值变量当作间隔数值变量来进行计算处理；但这不是一个好方法，因为比例尺度时非线性的。
- 利用对数转换方法（$y_{if}=\log(x_{if})$）来处理第i个对象中取x_{if}变量f；然后将y_{if}当作间隔数值变量并根据 6.2.1 小节所介绍有关间隔数值变量的任一个距离计算公来进行计算处理。需要说明的是，对于某些比例数值变量还可以根据具体定义和应用要求，采用$\log-\log$或其它转换方法对其进行变换。
- 最后就是将x_{if}当作连续顺序数据，即将其顺序值作为间隔数值来进行相应的计算处理。

后两个方法是最有效的；尽管选择所使用的方法或许与相应的应用相关。

6.2.4 混合类型属性

从6.2.1小节到6.2.3小节讨论了计算利用相同类型变量所描述对象间的距离方法；这些变量类型包括：间隔数值类型、对称二值类型、非对称二值类型、符号类型、顺序类型和比例数值类型。但在实际数据库中，数据对象往往是用复合数据类型来描述；而且常常它们（同时）包含上述六种数据类型。

一种将每种类型的变量分别组织在一起，并根据每种类型的变量完成相应的聚类分析。如果这样做可以获得满意的结果，那这种方法就是可行的。但在实际

应用中，根据每种类型的变量（单独）进行聚类分析不可能获得满意的结果。

一个更好的方法就是将所有类型的变量放在一起进行处理，一次（性）完成聚类分析。这就需要将不同类型变量（值）组合到一个差异矩阵中，并将它们所有有意义的值全部映射到$[0-1]$区间内。

假设一个数据集包含p个组合类型变量。对象i和对象j之间距离$d(i,j)$就可以定义为:

$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}} \tag{6.14}$$

其中如果（1）x_{if}或x_{jf}数据不存在（对象i或对象j的变量f无测量值）;或（2）$x_{if}=x_{jf}=0$且变量f为非对称二值变量，则标记$\delta_{ij}^{(f)}=0$；否则$\delta_{ij}^{(f)}=1$。而变量f对对象i和对象j之间差异程度（距离）的贡献，$d_{ij}^{(f)}$可以根据其具体变量类型进行相应计算:

（1）若变量f为二值变量或符号变量，则如果$x_{if}=x_{jf}$，那么$d_{ij}^{(f)}=0$；否则$d_{ij}^{(f)}=1$。

（2）若变量f为间隔数值变量，则$d_{ij}^{(f)}=\frac{|x_{if}-x_{jf}|}{\max_h x_{hf}-\min_h x_{hf}}$；其中$h$为变量$f$所有可能的对象。

（3）若变量f为顺序变量或比例数值变量，则计算顺序r_{if}和$z_{if}=\frac{r_{if}-1}{M_f-1}$，并将$z_{if}$当作间隔数值变量来进行计算处理。

综上所述，即使在对象是由不同类型变量（一起）描述时，也能够计算相应每两个对象间的距离。

6.3 主要聚类方法

在研究论文中有许多聚类算法。需要根据应用所涉及的数据类型、聚类的目的以及具体应用要求来选择合适的聚类算法。如果利用聚类分析作为描述性或探索性的工具，那么就可以使用若干聚类算法对同一个数据集进行处理以观察可能获得的有关（数据特征）描述。

通常聚类分析算法可以划分为以下几大类:

（1）划分方法

给定一个包含 n 个对象或数据行，划分方法将数据集划分为 k 个子集（划分）。其中每个子集均代表一个聚类（$k \le n$）。也就是说将数据分为 k 组，这些组满足以下要求：（a）每组至少应包含一个对象；且（b）每个对象必须只能属于某一组。需要注意的是后一个要求在一些模糊划分方法中可以放宽。有关这类方法将参考书后参考文献。

给定需要划分的个数 k，一个划分方法创建一个初始划分；然后利用循环再定位技术，即通过移动不同划分（组）中的对象来改变划分内容。一个好的划分衡量标准通常就是同一个组中的对象“相近”或彼此相关；而不同组中的对象“较远”或彼此不同。当然还有许多其它判断划分质量的衡量标准。

为获得基于划分聚类分析的全局最优结果就需要穷举所有可能的对象划分。为此大多数应用采用一至二种常用启发方法：（a）*k–means* 算法，该算法中的每一个聚类均用相应聚类中对象的均值来表示；和（b）*k–medoids* 算法，该算法中的每一个聚类均用相应聚类中离聚类中心最近的对象来表示。这些启发聚类方法在分析中小规模数据集以发现圆形或球状聚类时工作的很好。但为了使划分算法能够分析处理大规模数据集或复杂数据类型，就需要对其进行扩展。6.4 小节将要详细介绍基于划分的聚类方法。

（2）层次方法

层次方法就是通过分解所给定的数据对象集来创建一个层次。根据层次分解形成的方式，可以将层次方法分为自下而上和自上而下两种类型。自下而上的层次方法从每个对象均为一个（单独的）组开始；逐步将这些（对象）组进行合并，直到组合并在层次顶端或满足终止条件为止。自上而下层次方法从所有均属于一个组开始；每一次循环将其（组）分解为更小的组；直到每个对象构成一组或满足终止条件为止。

层次方法存在缺陷就是在进行（组）分解或合并之后，无法回溯。这一特点也是有用的，因为在进行分解或合并时无须考虑不同选择所造成的组合爆炸问题。但这一特点也使得这种方法无法纠正自己的错误决策。

将循环再定位与层次方法结合起来使用常常是有效的，即首先通过利用自下而上层次方法；然后再利用循环再定位技术对结果进行调整。一些具有可扩展性的聚类算法，如：BIRCH 和 CURE，就是基于这种组合方法设计的。6.5 小节将要详细介绍基于层次聚类方法。

（3）基于密度方法

大多数划分方法是基于对象间距离进行聚类的。这类方法仅能发现圆形或球状的聚类而在较难发现具有任何形状的聚类。而基于密度概念的聚类方法实际上

就是不断增长所获得的聚类直到“邻近”(数据对象或点)密度超过一定阈值(如:一个聚类中的点数，或一个给定半径内必须包含至少的点数）为止。这种方法可以用于消除数据中的噪声（异常数据)，以及帮助发现任意形状的聚类。

DBSCAN 就是一个典型的基于密度方法，该方法根据密度阈值不断增长聚类。OPTICS 也是一个基于密度方法，该方法提供聚类增长顺序以便进行自动或交互式数据分析。基于密度方法将在 6.6 小节作详细介绍。

（4）基于网格方法

基于网格方法将对象空间划分为有限数目的单元以形成网格结构。所有聚类操作均是在这一网格结构上进行的。这种方法主要优点就是处理时间由于与数据对象个数无关而仅与划分对象空间的网格数相关，从而显得相对较快。

STING 就是一个典型的基于网格的方法。CLIQUE 和 Wave-Cluster 是两个基于网格和基于密度的聚类方法。基于网格方法将在 6.7 小节进行讨论。

（5）基于模型方法

基于模型方法就是为每个聚类假设一个模型，然后再去发现符合相应模型的数据对象。一个基于模型的算法可以通过构造一个描述数据点空间分布的密度函数来确定具体聚类。它根据标准统计方法并考虑到“噪声”或异常数据，可以自动确定聚类个数；因而它可以产生很鲁棒的聚类方法。有关基于模型方法的情况将在 6.8 小节进行介绍。

一些聚类算法将若干聚类方法的思想结合在一起，因此有时很难明确界定一个聚类算法究竟属于哪一个聚类方法类别。此外一些应用也需要将多个聚类技术结合起来方可实现其应用目标。

以下各小节中，将要陆续介绍以上五种聚类方法；同时还将介绍将多个聚类思想结合在一起的聚类算法；此外在 6.9 小节还将讨论应用到聚类的异常数据分析的有关情况。

6.4 划分方法

给定包含 n 个数据对象的数据库和所要形成的聚类个数 k，划分算法将对象集合划分为 k 份（$k \le n$），其中每个划分均代表一个聚类。所形成的聚类将使得一个客观划分标准（常称为相似函数，如：距离）最优化；从而使得一个聚类中的对象是“相似”的；而不同聚类中的对象是“不相似”的。

6.4.1 传统划分方法

最常用也是最知名的划分方法就是 *k-means* 算法和 *k-medoids* 算法，以及它们的变化（版本)。

（1）*k-means* 算法

算法 6.1：根据聚类中的均值进行聚类划分的 *k-means* 算法。

输入：聚类个数 k，以及包含 n 个数据对象的数据库。

输出：满足方差最小标准的 k 个聚类。

处理流程：

（1） 从 n 个数据对象任意选择 k 个对象作为初始聚类中心；

（2） 循环（3）到（4）直到每个聚类不再发生变化为止

（3） 根据每个聚类对象的均值（中心对象），计算每个对象与这些中心对象的距离；并根据最小距离重新对相应对象进行划分；

（4） 重新计算每个（有变化）聚类的均值（中心对象）

如算法 6.1 所示，*k-means* 算法接受输入量 k；然后将 n 个数据对象划分为 k 个聚类以便使得所获得的聚类满足：同一聚类中的对象相似度较高；而不同聚类中的对象相似度较小。聚类相似度是利用各聚类中对象的均值所获得一个“中心对象”（引力中心）来进行计算的。

k-means 算法的工作过程说明如下：首先从 n 个数据对象任意选择 k 个对象作为初始聚类中心；而对于所剩下其它对象，则根据它们与这些聚类中心的相似度（距离），分别将它们分配给与其最相似的（聚类中心所代表的）聚类；然后再计算每个所获新聚类的聚类中心（该聚类中所有对象的均值）；不断重复这一过程直到标准测度函数开始收敛为止。一般都采用均方差作为标准测度函数，具体定义如下：

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} | p - m_i |^2 \qquad (6.15)$$

其中 E 为数据库中所有对象的均方差之和；p 为代表对象的空间中的一个点；m_i 为聚类 C_i 的均值（p 和 m_i 均是多维的）。公式（6.15）所示聚类标准旨在使所获得的 k 个聚类具有以下特点：各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。*k-means* 算法的计算复杂度为 $O(nkt)$，因而它在处理大数据库时也是相对有效的（具有可扩展性）；这里 n 为对象个数；k 为聚类个数；而 t 为循环次数。通常有 $k << n$ 和 $t << n$。*k-means* 算法常常终止于局部最优。

但是 *k-means* 算法只适用于聚类均值有意义的情况。因此在某些应用中，诸如：数据集包含符号属性时，直接应用 *k-means* 算法就有困难了。*k-means* 算法一个缺点就是用户还必须事先指定聚类个数 k。*k-means* 算法还不适合用于发现非凸形状的聚类，或具有各种不同大小的聚类。此外 *k-means* 算法还对噪声和异常数据也很敏感，因为这类数据可能会影响到各聚类的均值（计算结果）。

示例 6.2：假设空间数据对象分布如图-6.2（a）所示，设 $k=3$，也就是需要将数据集划分为三份（聚类）。

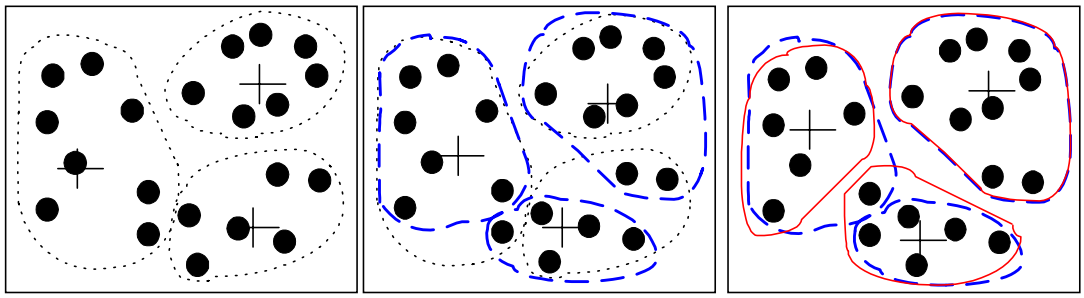


图-6.2 *k-means* 算法聚类过程示意描述

根据算法 6.1，从数据集中任意选择三个对象作为初始聚类中心（图-6.2（a）中这些对象被标上了“+”）；其余对象则根据与这三个聚类中心（对象）的距离，根据最近距离原则，逐个分别聚类到这三个聚类中心所代表的（三个）聚类中；由此获得了如图-6.2（a）所示的三个聚类（以虚线圈出）。

在完成第一轮聚类之后，各聚类中心发生了变化；继而更新三个聚类的聚类中心（图-6.2（b）中这些对象被标上了“+”）；也就是分别根据各聚类中的对象计算相应聚类的（对象）均值。根据所获得的三个新聚类中心，以及各对象与这三个聚类中心的距离，（根据最近距离原则）对所有对象进行重新归类。有关变化情况如图-6.2（b）所示（已用粗虚线圈出）。

再次重复上述过程就可获得如图-6.2(c)所示的聚类结果(已用实线圈出)。，这时由于各聚类中的对象（归属）已不再变化，整个聚类操作结束。 ■

k-means 算法还有一些变化（版本）。它们主要在初始 *k* 个聚类中心的选择、差异程度计算和聚类均值的计算方法等方面有所不同。一个常常有助于获得好的结果的策略就是首先应用自下而上层次算法来获得聚类数目，并发现初始分类；然后再应用循环再定位（聚类方法）来帮助改进分类结果。

另一个 *k-means* 算法的变化版本就是 *k-modes* 算法。该算法通过用模来替换聚类均值、采用新差异性计算方法来处理符号量，以及利用基于频率对各聚类模进行更新方法，从而将 *k-means* 算法的应用范围从数值量扩展到符号量。将 *k-means* 算法和 *k-modes* 算法结合到一起，就可以对采用数值量和符号量描述对象进行聚类分析，从而构成了 *k-prototypes* 算法。

而 EM（期望最大化）算法又从多个方面对 *k-means* 算法进行了扩展。其中包括：它根据描述聚类所属程度的概率权值，将每个对象归类为一个聚类，不是将一个对象仅归类为一个聚类（所拥有）；也就是说在各聚类之间的边界并不是

非常严格。因此可以根据概率权值计算相应的聚类均值。

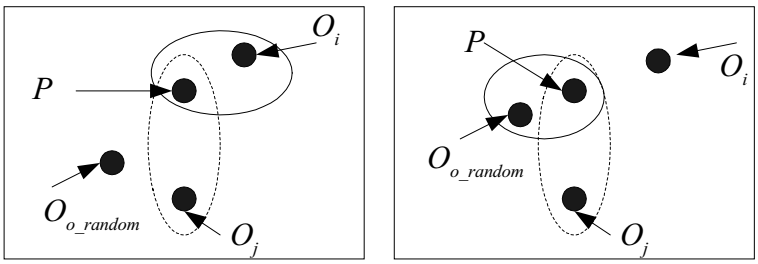
此外通过识别数据中所存在的三种类型区域，即可压缩区域、必须存入内存区域和可以丢弃区域，来改善 *k-means* 算法的可扩展性。若一个对象归属某个聚类的隶属值是不确定的，那它就是可丢弃的；若一个对象不是可丢弃的且属于一个更紧密的子聚类，那么它就是可压缩的。利用一个被称为是聚类特征的数据结构来对所压缩或所丢弃数据进行综合（summarize），若一个对象既不是可以丢弃的，也不是可以压缩的，那它就需要保持在内存里（在聚类过程中）。为实现可扩展性，循环聚类算法仅需对可压缩和可丢弃数据的聚类特征，以及须保持在内存中的对象进行分析处理即可。

（2）*k-medoids* 算法

由于一个异常数据的取值可能会很大，从而会影响对数据分布的估计（*k-means* 算法中的各聚类均值计算），因此 *k-means* 算法对异常数据很敏感。

为此就设想利用 *medoid* 来作为一个参考点代替 *k-means* 算法中的各聚类的均值（作为聚类中心）。从而可以根据各对象与各参考点之间的距离（差异性）之和最小化的原则，继续应用划分方法。这就构成了 *k-medoids* 算法。

k-medoids 聚类算法的基本策略就是通过首先任意为每个聚类找到一个代表对象（medoid）而首先确定 n 个数据对象的 k 个聚类；（也需要循环进行）其它对象则根据它们与这些聚类代表的距离分别将它们归属到各相应聚类中（仍然是最小距离原则）。而如果替换一个聚类代表能够改善所获聚类质量的话，那么就可以用一个新对象替换老聚类对象。这里将利用一个基于各对象与其聚类代表间距离的成本函数来对聚类质量进行评估。为了确定任一个非聚类代表对象 o_{random} 是否可以替换当前一个聚类代表（medoid）o_j，需要根据以下四种情况对各非聚类代表对象 p 进行检查。



(1) (2)

图-6.3 *k-medoids* 算法聚类过程示意描述（一）

（1） 若对象 p 当前属于 o_j（所代表的聚类），且如果用 o_{random} 替换 o_j 作为新聚类代表，而 p 就更接近其它 o_i（$i \neq j$），那么就将 p 归类到 o_i（所代表的

聚类）中；

（2） 若对象 p 当前属于 o_j（所代表的聚类），且如果用 o_{random} 替换 o_j 作为新聚类代表，而 p 最更接近 o_{random}，那么就将 p 归类到 o_{random}（所代表的聚类）中；

（3） 若对象 p 当前属于 o_i（所代表的聚类）（$i \neq j$），且如果用 o_{random} 替换 o_j 作为新聚类代表，而 p 仍然最接近 o_i，那么 p 归类不发生变化；

（4） 若对象 p 当前属于 o_i（所代表的聚类）（$i \neq j$），且如果用 o_{random} 替换 o_j 作为新聚类代表，而 p 最更接近 o_{random}，那么就将 p 归类到 o_{random}（所代表的聚类）中；

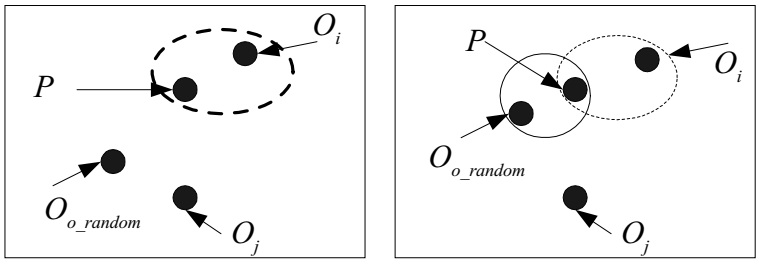


图-6.4 *k-medoids* 算法聚类过程示意描述（二）

图-6.3 和图-6.4 分别示意描述了上述 *k-medoids* 聚类算法的四种主要处理情况。每次对对象进行重新归类，都会使得构成成本函数的方差 E 发生变化。因此成本函数能够计算出聚类代表替换前后的方差变化。通过替换不合适的代表来而使距离方差发生变化的累计就构成了成本函数的输出。若整个输出成本为负值，那么就用 o_{random} 替换 o_j，以便能够减少实际的方差 E。若整个输出成本为正值，那么就认为当前的 o_j 是可接受的，本次循环就无需变动。一个基本的 *k-medoids* 聚类算法如算法 6.2 所示。

算法 6.2：根据聚类的中心对象（聚类代表）进行聚类划分的 *k-medoids* 算法。

输入：聚类个数 k，以及包含 n 个数据对象的数据库。

输出：满足基于各聚类中心对象的方差最小标准的 k 个聚类。

处理流程：

（1） 从 n 个数据对象任意选择 k 个对象作为初始聚类（中心）代表；

（2） 循环（3）到（5）直到每个聚类不再发生变化为止

（3） 依据每个聚类的中心代表对象，以及各对象与这些中心对象间距离；并根据最小距离重新对相应对象进行划分；

（4） 任意选择一个非中心对象o_{random}；计算其与中心对象o_j交换的整个成本S。

（5） 若S为负值则交换o_{random}与o_j以构成新聚类的k个中心对象

PAM(围绕中心对象进行划分)方法是最初提出的*k-medoids*聚类算法之一。它在初始选择k个聚类中心对象之后，不断循环对每两个对象（一个为非中心对象，一个为中心对象）进行分析，以便选择出更好的聚类中心代表对象。并根据每组对象分析计算所获得的聚类质量。若一个中心对象o_j被替换后导致方差迅速减少，那么就进行替换。对于较大的n与k值这样的计算开销也非常大。

*k-medoids*聚类算法比*k-means*聚类算法在处理异常数据和噪声数据方面更为鲁棒；因为与聚类均值相比，一个聚类中心的代表对象要较少受到异常数据或极端数据的影响。但是前者的处理时间要比后者更大。两个算法都需要用户事先指定所需聚类个数k。

6.4.2 大数据库的划分方法

像PAM方法这样典型的*k-medoids*聚类算法，在小数据集上可以工作的很好；但是对于大数据库则处理效果并不理想。可以利用一个基于采样的聚类方法，称为CLARA（Clustering LARge Application），来有效处理大规模数据。

CLARA算法的基本思想就是：无需考虑整个数据集，而只要取其中一小部分数据作为其代表；然后利用PAM方法从这个样本集中选出中心对象。如果样本数据是随机选择的，那么它就应该近似代表原来的数据集。从这种样本集所选择出来的聚类中心对象可能就很接近从整个数据集种所选择出来聚类中心（对象）。CLARA算法分别取若干的样本集，然后对每个样本数据集应用PAM方法，然后将其中最好的聚类（结果）输出。CLARA算法能够处理大规模数据集，而它的每次循环（计算）复杂度为$O(ks^2+k(n-k))$；其中s为样本集合大小；k为聚类个数；n为对象总数。

CLARA算法的有效性依赖其所选择的样本集合大小；PAM方法从给定的数据集中搜索最好的k个聚类中心（对象）；而CLARA算法则从所采样的数据样本集中搜索最好的k个聚类中心（对象）。如果样本集中的聚类中心不是（整个数据集中）最好的k个聚类中心，那么CLARA算法就无法发现最好的聚类结果。例如：若一个对象o_i是一个最好的聚类中心（对象），但在样本集聚类中没有被选中，那CLARA算法就无法找到（整个数据集中）最好的聚类。这也就是对效率和精度的折衷。如果采样有偏差（bias），那么一个基于采样的好聚类算法常常就无法找出（整个数据集中）最好的聚类。

另一个*k-medoids*聚类算法类型的聚类方法，称为CLARANS（Clustering Large Application based upon RANdomized Search），将采样方法与PAM方法结合

起来。但 CLARANS 方法与 CLARA 算法不同，CLARANS 方法并不总是仅对样本数据集进行分析处理; CLARANS 方法在搜索的每一步都以某种随机方式进行采样（而 CLARA 算法搜索每一步所处理的数据样本是固定的）。CLARANS 方法的搜索过程可以描述成一个图，图中每个结点度代表潜在的解决方案（一组聚类中心代表），替换一个中心对象所获得新聚类就称为当前聚类的邻居。随机产生的聚类邻居数由用户所设置的参数所限制。若发现一个更好的邻居（具有较低的方差），CLARANS 方法移动到这一邻居结点然后再开始搜索。否则当前结点就形成了一个局部最优。若发现局部最优，CLARANS 方法则随机选择一个结点以便重新开始搜索（一个新的局部最优）。

CLARANS 方法的实验结果表明它比 CLARA 方法和 PAM 方法更为有效。利用（聚类）轮廓相关系数（描述一个对象所代表聚类可以真正拥有多少对象的性质），CLARANS 方法能够发现最“自然”的聚类个数。CLARANS 方法也可以用于检测异常数据。但是 CLARANS 方法的计算复杂度为 $O(n^2)$，其中 n 为对象总数。CLARANS 方法的聚类质量与所使用的采样方法无关。通过采用诸如 $R*-$ 树或其它技术可以帮助改进 CLARANS 方法的处理性能。

6.5 层次方法

层次聚类方法是通过将数据组织为若干组并形成一个组的树来进行聚类的。层次聚类方法又可以分为自顶而下和自下而上层次聚类两种。一个完全层次聚类的质量由于无法对已经做的合并或分解进行调整而受到影响。目前的研究都强调将自下而上层次聚类与循环再定位方法相结合。

6.5.1 两种基本层次聚类方法

一般有两种基本层次聚类方法，它们分别是:

（1） 自下而上聚合层次聚类方法。这种自下而上策略就是最初将每个对象（自身）作为一个聚类; 然后将这些原子聚类进行聚合以构造越来越大的聚类，直到所有对象均聚合为一个聚类，或满足一定终止条件为止。大多数层次聚类方法都属于这类方法，但它们在聚类内部对象间距离定义描述方面有所不同。

（2） 自顶而下分解层次聚类方法。这种自顶而下策略的作法与自下而上策略做法相反。它首先将所有对象看成一个聚类的内容; 将其不断分解以使其变成越来越小但个数越来越多的小聚类，直到所有对象均独自构成一个聚类，或满足一定终止条件（如：一个聚类数阈值，或两个最近聚类的最短距离阈值）为止。

示例 6.3：如图-6.5 所示，就分别是一个自下而上聚合层次聚类方法 AGNES（AGglomerative NESting）和一个自顶而下分解层次聚类方法 DIANA（DIvsia ANAlysia）的应用示例。其中数据集为 $\{a,b,c,d,e\}$，共有 5 个对象。开始 AGNES 方法将每个对象构成一个单独聚类；然后根据一定标准不断进行聚合。如：对于聚类 C_1 和 C_2 来讲，若 C_1 中对象与 C_2 中对象间欧式距离为不同聚类中任两个对象间的最小距离，则聚类 C_1 和 C_2 就可以进行聚合。两个聚类之间相似程度是利用相应两个聚类中每个对象间的最小距离来加以描述的。AGNES 方法不断进行聚合操作，直到所有聚类最终聚合为一个聚类为止。

而在 DIANA 方法中，首先所有的对象在一起构成了一个聚类。然后根据一定原则，如：聚类中最近对象间的最大欧式距离，对其进行不断分解，直到每个聚类均只包含一个对象为止。 ■

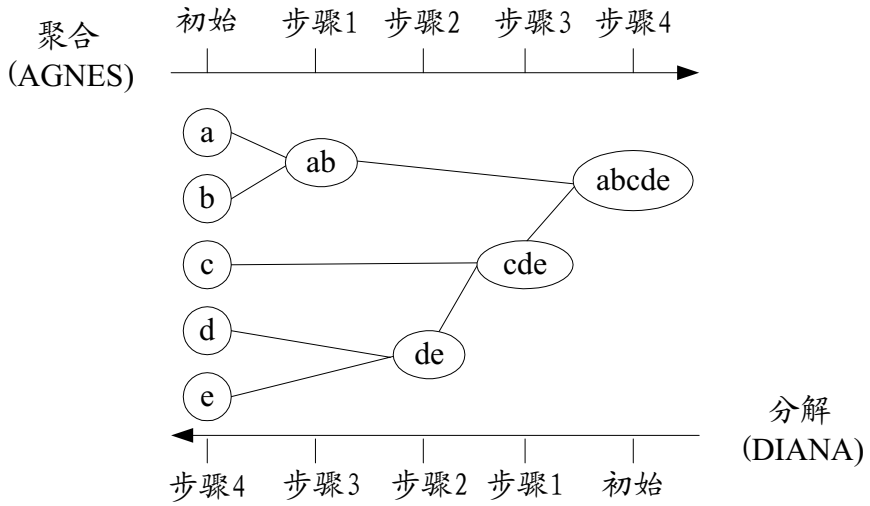


图-6.5 聚合和分解层次聚类方法示意描述

自下而上聚合层次聚类方法和自顶而下分解层次聚类方法中，用户均需要指定所期望的聚类个数作为聚类过程的终止条件。

四个常用的计算聚类间距离的公式说明如下：

- **最小距离**：$d_{\min}(C_i,C_j)=\min_{p\in C_i,p'\in C_j}|p-p'|$
- **最大距离**：$d_{\max}(C_i,C_j)=\max_{p\in C_i,p'\in C_j}|p-p'|$
- **距离均值**：$d_{mean}(C_i,C_j)=|m_i-m_j|$

♦ **平均距离**：$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$

其中 m_i 为聚类 C_i 的均值；n_i 为 C_i 中的对象数；$|p - p'|$ 为两个数据对象或点 p 和 p' 之间的距离。

层次聚类方法尽管简单，但经常会遇到如何选择合并或分解点的问题。这种决策非常关键，因为在对一组对象进行合并或分解之后，聚类进程将在此基础上继续进行合并或分解，这样就既无法回到先前的（聚类）状态；也不能进行聚类间的对象交换。因此如果所做出的合并或分解决策（在某一点上）不合适，就会导致聚类结果质量较差。此外由于在作出合并或分解决策前需要对许多对象或聚类进行分析评估，因此使得该类方法的可扩展性也较差。

改进层次方法聚类质量的可行方法就是将层次方法与其它聚类技术相结合以进行多阶段的聚类。在以下各小节中将要介绍一些有关的具体结合（所得）方法。第一个是 BIRCH 方法，它首先利用树的结构对对象集进行划分；然后再利用其它聚类方法对这些聚类进行优化。第二个是 CURE 方法，它利用固定数目代表对象来表示相应聚类；然后对各聚类按照指定量（向聚类中心）进行收缩。第三个是 ROCK 方法，它利用聚类间的连接进行聚类合并。最后一个 CHAMELEON，它则是在层次聚类时构造动态模型。

6.5.2 两种层次聚类方法

BIRCH（Balanced Iterative Reducing and Clustering using Hierarchies）方法是一个集成的层次聚类方法。它包含两个重要概念：聚类特征（简称 CF）和聚类特征树（CF tree）。这两个概念用于对聚类描述进行概要总结。相应的有关数据结构将帮助聚类方法获得较好的聚类速度和可对大数据库进行处理的可扩展性。此外 BIRCH 方法在进行增量和动态聚类时也是很有效的。

现在介绍 BIRCH 方法所采用的主要数据结构。聚类特征（CF）是有关对象子集概要信息的一个三元组。设一个子聚类（subcluster）包含 N 个 d – 维数据或对象 o_i，那么这个子聚类的 CF 就定义为：

$$CF = (N, \vec{LS}, SS) \qquad (6.16)$$

其中 N 为该子聚类所含对象的个数；$\vec{LS}$ 为这 N 个点的和，即 $\sum_{i=1}^{N} \vec{\sigma}_i$；$SS$ 为数据点的平方和，即 $\sum_{i=1}^{N} \vec{\sigma}_i^{\,2}$。

聚类特征基本上就是对给定子聚类统计信息的总结。它包含了聚类计算和空间存储利用所需要的关键信息。

CF 树是一个高度平衡树，它存有用于层次聚类的聚类特征。图-6.6 所示就是一个 CF 树示意描述。根据定义 CF 树中非叶结点存放其子女结点的 CF 值。一个 CF 树有两个主要参数：分支系数 *B* 和阈值 *T*。分支系数 *B* 指定了每个非叶结点的最大子女数；而阈值 *T* 则指定了存放在叶节点中子聚类的最大直径。这两个参数影响所获 CF 树的大小。

BIRCH 方法工作主要包括两个阶段：

- 第一阶段：BIRCH 方法扫描数据库以建立一个初始基于内存的 CF 树，该树可以看成是对数据的压缩且还保留着数据中所包含的有关聚类结构的内涵。
- 第二阶段：BIRCH 方法应用一个（所选择）的聚类算法对 CF 树的叶结点进行聚类。

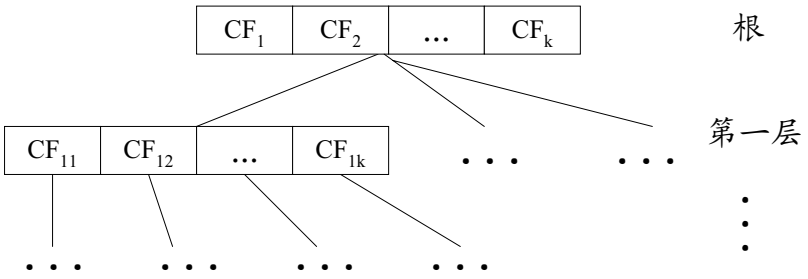


图-6.6 CF 树示意描述

在第一阶段，CF 树是根据不断插入的对象而动态建立的。因此 BIRCH 方法是增量式聚类。一个对象被插入到与它最接近的叶节点中。若一个叶结点的子聚类直径在插入一个新对象后大于阈值 *T*，那该叶结点和其它结点就要进行分解；而在插入一个新对象之后，有关（它）的信息将上传到根结点。通过修改阈值 *T* 可以改变 CF 树的大小。如果存放 CF 树所需要的内存大于现有内存，那就要指定较小的阈值 *T* 并重建 CF 树；重建工作将从原来 CF 树的叶结点开始。所以 CF 树重建工作将无须重新读入所有的数据。这一点与构造 B+树时的插入与分解过程类似。因此构造 CF 树时，数据只需要读一遍；而利用一些启发式方法则可以通过重复读入数据来帮助处理异常数据以改善 CF 树质量。

在构造完 CF 树后，（第二阶段）可利用任何聚类算法，主要是划分聚类方法，对所获得的 CF 树进行聚类分析。

BIRCH 方法努力在现有资源条件下产生最好的聚类。面对有限的内存，一个重要考虑就是如何使得 I/O 时间最小。BIRCH 方法利用多阶段处理方式：现扫描一遍数据获得一个基本理想的聚类；再次扫描一遍数据以帮助改善（所获）

聚类的质量。BIRCH 的计算复杂度为 $O(n)$，其中 n 为带聚类的对象数。

有关的实验结果表明：基于对象数目和聚类的质量， BIRCH 算法表现出线性可扩展性；然而由于大小的限制，CF 树中的每个结点仅能容纳有限的入口，因此一个 CF 树结点并不总能对应用户所认为的一个自然聚类；此外如果聚类不是圆状的，则会由于 BIRCH 算法是利用半径来控制一个聚类半径的，从而导致算法的性能变差。

6.5.3 层次聚类方法：CURE

大多聚类算法偏向发现具有相似大小和圆形形状的聚类，或在处理异常数据时表现很差。CURE 方法将层次方法与划分方法结合到了一起。它克服了偏向发现相似大小和圆形形状聚类的问题；同时在处理异常数据时也表现得更加鲁棒。

CURE（Clustering Using REpresentatives）利用一个新的层次聚类算法，该算法属于（自下而上）聚合方法与（自上而下）分解的中间做法。它不是仅用一个聚类中心或对象来描述一个聚类；而是选用固定数目有代表性的空间点来表示一个聚类。表示聚类的代表性点则是首先通过选择分布较好的聚类对象来产生；然后根据指定的速率（收缩因子）将它们“收缩”或移向聚类的中心。算法的每一步，就是对拥有分别来自两个不同聚类两个最近（代表性）点所涉及的两个聚类进行合并。

每个聚类包含多于一个的代表性点将有助于 CURE 方法调整好自己的非圆状边界。聚类的收缩或压缩将有助于帮助压制异常数据。因此 CURE 方法对异常数据表现得更加鲁棒；同时它也能识别具有非圆形状和不同大小的聚类。此外 CURE 方法在不牺牲聚类质量的情况下，对大数据库的处理也具有较好的可扩展性。

为处理好大数据库，CURE 方法利用了随机采样和划分方法。即首先对随机采样（集合）进行划分，每个划分都是部分聚类；然后这些部分聚类在第二遍扫描中进行聚类以获得所期望的最终聚类结果。

CURE 算法的主要处理步骤说明如下：

（1） 进行随机采样并获得这样集合 S，它包含 s 个对象；

（2） 将采样集合 S 划分为 p 个划分，每个划分大小为 s/p；

（3） 将各划分部分聚类成 s/pq 个聚类，其中 $q>1$；

（4） 通过随机采样消除异常数据，即若一个聚类增长太慢，就除去它；

（5） 对部分聚类进行聚类，落在每个新获得的聚类中的代表性点，则根据收缩因子 α，“收缩”或移向聚类的中心。这些点将要用于代表并描绘出聚类的边界；

（6） 对聚类中的数据标记上相应聚类号。

以下就是一个描述 CURE 算法主要处理步骤的具体示例。

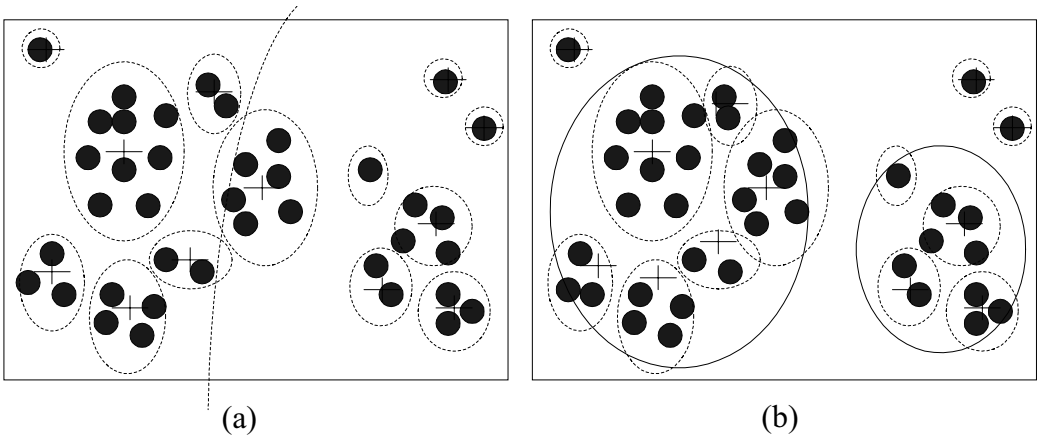


图-6.7 CURE 算法主要处理步骤示意描述

示例 6.4: 在一个矩形区域内分布着一组点或对象。假设需要将这些对象划分为两个聚类，即 $p=2$。

首先随机采样 $s=52$ 个对象，如图-6.7（a）所示，这些对象被分为两个划分，每个划分包含 $52/2=26$ 个点。设 $q=2$，那根据最小距离均值，将这些划分归并为 $52/(2\times2)=13$ 个部分聚类；如图-6.7（a）所示，其中部分聚类由虚线标出.每个聚类代表用“+”标出。然后再对部分聚类作进一步的聚类，并获得如图-6.7（b）所示的用实线标出的两个聚类。每个新获得的聚类中的代表性点，则根据收缩因子 α，“收缩”或移向聚类的中心。这些点将代表并帮助描绘出相应聚类的边界。因此最初的数据对象最终被聚合为两个聚类，且有关的异常数据被排除在外。

CURE 算法在对含有异常数据对象进行分析时，也能够获得较高的聚类质量。此外它还容许聚类具有复杂的形状和不同的大小。该算法只需要对整个数据库进行一遍扫描。因此给定 n 个对象，CURE 算法的复杂度为 $O(n)$。相应的敏感性分析结果表明：尽管某些参数变化不会影响其聚类质量，但是参数的设置的确会对最终结果产生较大的影响。

ROCK 也是一个聚合层次聚类算法。与 CURE 算法不同，ROCK 算法适合处理符号属性。它通过将两个聚类间连接的累计与用户所指定的静态连接模型相比较，计算出两个聚类间的相似性。而所谓两个聚类（C_1 和 C_2）间连接就是指两个聚类间的连接数目。而 $link(p_1,p_2)$ 就是指两个点 p_1 和 p_2 之间共同邻居的数目。也就是聚类间的相似程度是利用不同聚类中点所具有的共同邻居数来确定的。

ROCK 算法首先根据所给数据的相似矩阵和相似阈值，构造出一个松散图；然后在这一松散图上应用一个层次聚类算法。

6.5.4 层次聚类方法：CHAMALEON

CHEMALEON 是一个探索层次聚类中动态模型的聚类算法。在其聚类过程中，如果两个聚类间的连接度和相似度与聚类内部的连接度和相似度密切相关，那么就合并这两个聚类。基于动态模型的合并过程将有助于发现自然和同质的聚类；并在定义了有关相似函数的情况下，适用于任何的数据类型。

CHEMALEON 是针对 CURE 和 ROCK 这两个层次聚类算法所存在的不足而提出的。CURER 忽略了两个不同聚类间的连接累计信息；而 ROCK 则在强调聚类间连接信息却忽略了有关两个聚类间相接近的信息。

CHEMALEON 首先利用一个图划分算法将数据对象聚合成许多相对较小的子聚类；然后再利用聚合层次聚类方法，并通过不断合并这些子聚类来发现真正的聚类。为确定哪两个子聚类最相似，该算法不仅考虑了聚类间的连接度，而且也考虑了聚类间的接近度，特别是聚类本身的内部特征。由于算法并不依赖一个静态用户指定的模型，因此它能够自动适应要合并的聚类内部特征。

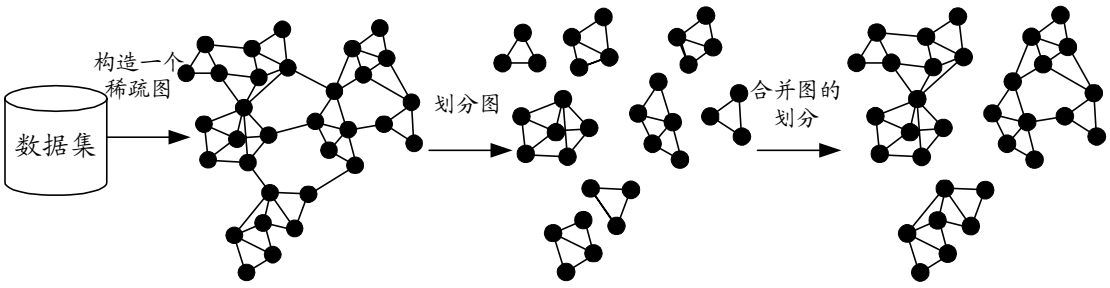


图-6.8 CHEMALEON 算法主要处理步骤示意描述

如图-6.8 所示，CHEMALEON 算法利用常用 $k-$最近邻图方法来表示相应的对象。$k-$最近邻图中的每个顶点代表一个数据对象；若一个对象为另一个对象 $k-$最近邻（对象）之一，则这两个对象之间就存在一条边。$k-$最近邻图 C_k 动态描述了相邻的概念。一个对象近邻半径是由该对象所处区域的密度来决定的。在一个密集区域中，其近邻就很狭小；而在稀疏区域中，其近邻就较广。这样就可能得到比较自然的聚类。此外一个区域密度就定义为其中的边数。这样一个密集的区域所含的边数显然要多于一个稀疏区域所含的。

CHEMALEON 算法根据两个聚类间的相对连接度 $RI(C_i,C_j)$ 和相对接近度 $RC(C_i,C_j)$，来确定两个聚类 C_i 和 C_j 间的相似度。

两个聚类C_i和C_j间的相对连接度$RI(C_i,C_j)$定义为两个聚类C_i和C_j间的绝对连接度除以两个聚类C_i和C_j内的连接度。也就是:

$$RI(C_i,C_j)=\frac{|EC_{\{C_i,C_j\}}|}{(|EC_{C_i}|+|EC_{C_j}|)/2} \tag{6.17}$$

其中$EC_{\{C_i,C_j\}}$为包含C_i和C_j聚类的切边(edge-cut),以便该聚类可分解为C_i和C_j;类似的EC_{C_i}或EC_{C_j}就是最小二分的切边数(即将图切为基本相同的两半带权边的合计)。

两个聚类C_i和C_j间的相对接近度$RC(C_i,C_j)$,就是两个聚类C_i和C_j间的绝对接近度$RC(C_i,C_j)$除以两个聚类C_i和C_j内的接近度,也就是:

$$RC(C_i,C_j)=\frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}}+\frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}} \tag{6.18}$$

其中$\overline{S}_{E_{\{C_i,C_j\}}}$为$C_i$中顶点与$C_j$中顶点之间边的平均权值;$\overline{S}_{E_{C_i}}$或$\overline{S}_{E_{C_j}}$为二分$C_i$或$C_j$的最小边切的平均边权值。

有关研究表明,与 CURE 和 RDBSCAN 方法相比,CHEMALEON 算法在发现具有高质量任意形状聚类方面能力更强;但在最坏情况下,它处理高维数据还可能需要$O(n^2)$时间。

6.6 基于密度方法

基于密度方法能够帮助发现具有任意形状的聚类。一般在一个数据空间中,高密度的对象区域被低密度(稀疏)的对象区域(通常就认为是噪声数据)所分割。

6.5.1 基于密度方法:DBSCAN

DBSCAN(Density-based Spatial Clustering of Application with Noise)是一个基于密度的聚类算法。该算法通过不断生长足够高密度区域来进行聚类;它能从含有噪声的空间数据库中发现任意形状的聚类。DBSCAN 方法将一个聚类定义为一组“密度连接”的点集。

为了讲解清楚基于密度聚类方法的基本思想,以下首先介绍该方法思想所包含一些概念;然后再给出一个示例来加以说明。

（1） 一个给定对象的ε半径内的近邻就称为该对象的ε–近邻;

（2） 若一个对象的ε–近邻至少包含一定数目（*MinPts*）的对象，该对象就称为核对象;

（3） 给定一组对象集D，若对象p为另一个对象q的ε–近邻且q为核对象，那么就说p是从q可以“直接密度可达”;

（4） 对于一个ε而言，一个对象p是从对象q可“密度可达”；一组对象集D有*MinPts*个对象；若有一系列对象$p_1, p_2, \ldots, p_n$，其中$p_1 = q$且$p_n = p$，从而使得（对于ε和*MinPts*来讲）p_{i+1}是从p_i可“直接密度可达”。其中有$p_i \in D$，$1 \le i \le n$。

（5） 对于ε和*MinPts*来讲，若存在一个对象o（$o \in D$），使得从o可“密度可达”对象p和对象q，对象p是“密度连接”对象q。

密度可达是密度连接的一个传递闭包。这种关系是非对称的。仅有核对象是相互“密度可达”。而密度连接是对称的。

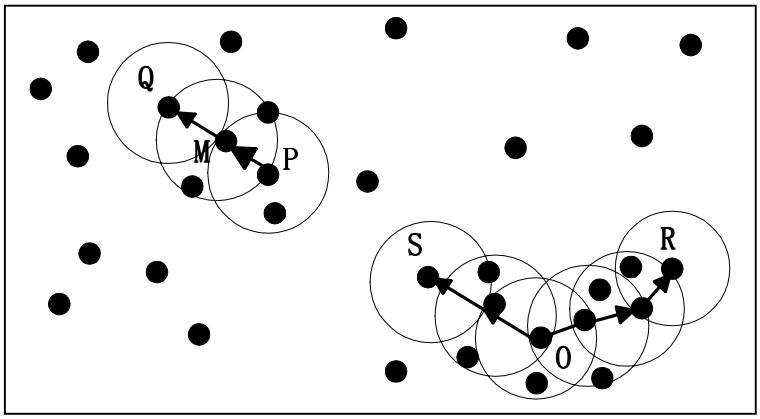


图-6.9 “直接密度可达”和“密度可达”概念示意描述

示例 6.5: 如图-6.9 所示，ε用一个相应的半径表示，设*MinPts*=3。根据以上概念就有:

- 由于有标记的各点M、P、O和R的ε–近邻均包含 3 个以上的点，因此它们都是核对象;
- M是从P可“直接密度可达”；而Q则是从M可“直接密度可达”;
- 基于上述结果，Q是从P可“密度可达”；但P从Q无法“密度可达”；（非对称）。类似的，S和R从O是“密度可达”的;
- O、R和S均是“密度连接”的。 ■

基于密度聚类就是一组“密度连接”的对象，以实现最大化的“密度可达”。不包含在任何聚类中的对象就为噪声数据。

DBSCAN 检查数据库中每个点的ε–近邻。若一个对象 p 的ε–近邻包含多于 *MinPts*，就要创建包含 p 的新聚类。然后 DBSCAN 根据这些核对象，循环收集“直接密度可达”的对象，其中可能涉及进行若干“密度可达”聚类的合并。当各聚类再无新点（对象）加入时聚类进程结束。

DBSCAN 的计算复杂度为 $O(n\log n)$，其中 n 为数据库中对象数。DBSCAN 算法对用户所要设置的参数敏感。在下一小节还将涉及 DBSCAN 的比较。

6.6.2 基于密度方法：OPTICS

虽然上一小节所介绍的 DBSCAN 可以在给定输入参数ε和 *MinPts* 时进行聚类操作。但它仍然需要用户负责设置可帮助发现有效聚类的参数。而实际上这是一个许多聚类算法都存在的问题。这些参数常常是根据经验而定，尤其在多维数据集中一般都较难确定。而许多算法对参数的设置都较为敏感。参数稍微的改变都会引起聚类结果的巨大不同。而且多维数据集中数据分布经常是怪异的。有时甚至不存在一个全局的参数设置以使得聚类算法获得能准确描述聚类内在结构的结果。

为帮助克服这一问题，人们提出了一个称为 OPTICS（Ordering Points To Identify the Clustering Structure）的聚类顺序方法。OPTICS 方法并不明确产生一个聚类，而是为自动交互的聚类分析计算出一个增强聚类顺序。这一顺序表达了基于密度的数据聚类结构。它包括与基于许多参数设置所获基于密度聚类相当的信息。

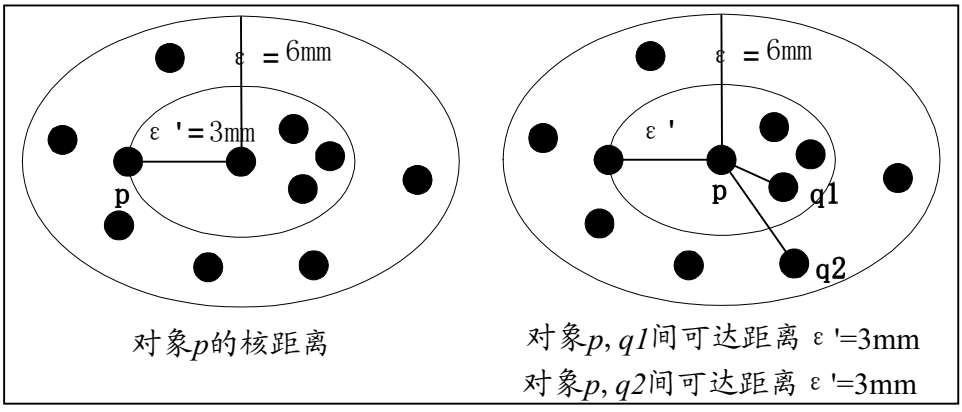


图-6.10 OPTICS 方法中“核距离”和“可达距离”概念描述

仔细研究一下 DBSCAN，就会发现：对于一个 *MinPts* 常数，具有较高密度的密度聚类（ε值较小）包含在具有密度较低的密度聚类中。而参数ε为一个距离（近邻半径），因此为获得一组密度聚类顺序，就要提供一系列距离参数值。

为了同时构造不同聚类，应该按照一个特定的顺序处理对象。这个顺序是选择(对于低ε)“密度可达”的对象以便将高密度聚类排在前列。基于这个思路，每个对象需要保存两个值：核距离(core-distance)和可达距离(reachability-distance)。

- 一个对象p的核距离就是使p成为核对象的最小ε'。若p不是一个核对象，p的核距离就是未定义。
- 一个对象p和另一个对象q间的可达距离是p的核距离和p、q间的欧氏距离中较大的。若p不是一个核对象，p和q间的可达距离就是未定义。

示例 6.6：如图-6.10所示，就是对核距离和可达距离概念的示意描述。假设ε=6mm，*MinPts*=5。对象p的核距离就是p和第四个最近数据对象的距离。

■

OPTICS算法对数据库中对象建立一个对象顺序，并保存（每个对象）核距离和一个合适的可达距离。这些信息足以帮助根据任何小于产生聚类顺序(所用)距离ε的距离ε'，产生所有的密度聚类。

由于OPTICS算法的基本结构与DBSCAN类似，因此OPTICS算法的计算复杂度同DBSCAN相同，即也为$O(n\log n)$。此外还可以利用空间索引结构以帮助改善OPTICS算法的性能。

6.7 基于网格方法

基于网格聚类方法利用多维网格数据结构。它将空间划分为有限数目的单元，以构成一个可以进行聚类分析的网格结构。这种方法的主要特点就是处理时间与数据对象数目无关，但与每维空间所划分的单元数相关，因此基于网格聚类方法处理时间很短。

6.7.1 基于网格方法：STING

STING（STatistical INformation Grid）是一个基于网格多分辨率的聚类方法。它将空间划分为方形单元。不同层次的方形单元对应不同层次的分辨率。这些单元构成了一个层次结构：高层次单元被分解形成一组低层次单元。有关各网格单元属性的统计信息（如：均值、最大、最小）可以事先运算和存储。这些信息将在（稍候介绍的）查询处理用到。

如图-6.11所示，就是一个STING使用的层次结构。高层次单元的统计信息可以通过低层次单元很容易地计算处理。这些参数包括：与属性无关的参数，计数*count*和与属性有关的参数，均值*m*、标准方差*s*、最小值*min*与最大值*max*，以及单元中属性值的分布类型，如：均匀分布、随机分布、指数分布或未知。当

数据存入数据库时，首先根据数据计算最底层单元的参数 *count*、*m*、*s*、*min*、*max*，而数据分布可以由用户指定（如果分布事先已知的话），也可以用 χ^2 – 测试来进行假设测试以获得数据分布（类型）。高层次单元中的数据分布将根据低层次单元中占多数的数据分布类型以及过滤阈值来确定。若低层次单元中的数聚分布彼此不同且阈值测试失败，那么高层次单元中的数据分布设为未知。

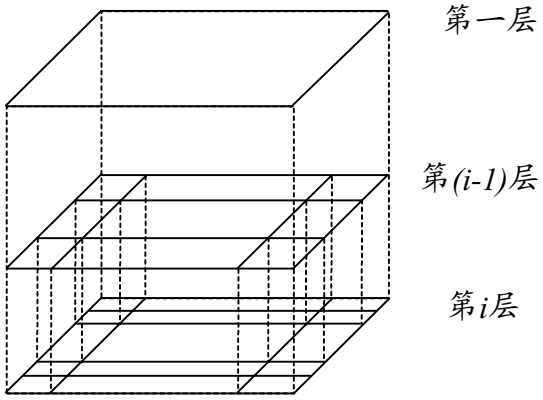


图-6.11 STING 方法使用的层次结构示意描述

一个自上而下基于网格方法处理查询的操作步骤说明如下：首先根据查询内容确定层次结构的开始层次。通常这一层次包含较少的单元。对于当前层次中的每个单元，计算信任度差（或估计概率范围）以反映当前单元与查询要求的相关程度。消除无关单元以便仅考虑相关单元。不断重复这一过程直到到达最底层。这时若满足查询要求，返回满足要求的相关单元区域。否则取出相关区域单元中的数据，对它们作进一步处理直到满足查询要求。

与其它聚类方法相比，STING 方法有以下几个优点：（1）基于网格计算由于描述网格单元数据统计信息是存储在相应单元中，因此它与查询要求无关；（2）网格结构有助于实现并行运算和增量更新；（3）STING 方法仅扫描一遍数据库以获得各单元的统计信息，因此它产生聚类的时间复杂度为 $O(n)$；其中 n 为所有对象数。在产生聚类后进行查询的实际复杂度为 $O(g)$；其中 g 为在最底层的所有网格数，它通常比 n 要小许多。

由于 STING 方法是利用多分辨率来完成聚类分析的，STING 方法的聚类质量就依赖于网格结构的最低层细度。若细度非常高，那处理开销将会增加许多；然而若网格结构的最低层太粗，那就会降低聚类分析的质量。此外 STING 方法没有考虑子女与其父单元相邻单元在空间中的相互关系。

因此所获得的聚类形状是直方的，也就是所有聚类的边界是水平的或是垂直的；而没有对角边界。尽管处理速度很快，但会降低聚类的质量和准确性。

6.7.2 基于网格方法：CLIQUE

CLIQUE（Clustering In QUEst）聚类方法，将基于密度方法与基于网格方法结合在一起。它对处理大数据库中的高维数据比较有效。

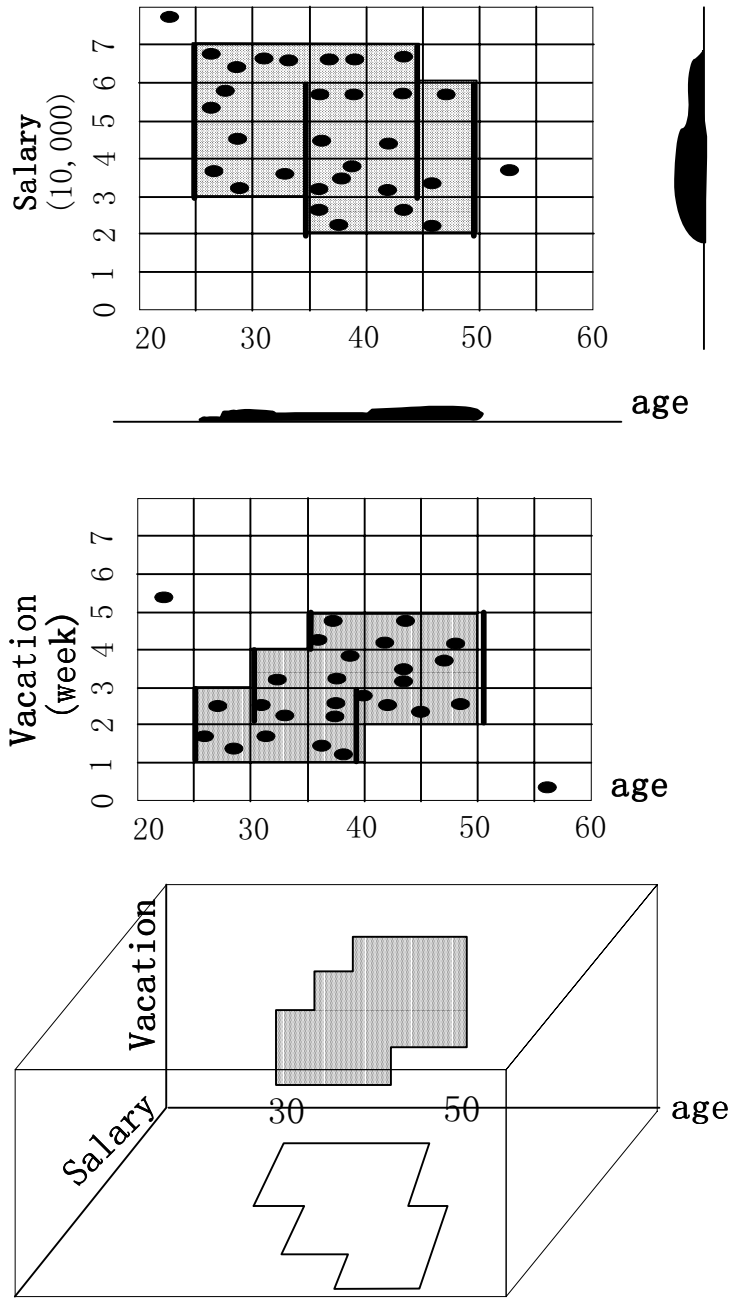


图-6.12 根据 Salary 和 Vacation 所发现 Age 密度描述

CLIQUE 方法的基本内容说明如下:

- 给定一个大规模多维数据点，数据空间中的数据点通常并不是均匀分布的。CLIQUE 聚类识别稀疏和“拥挤”空间区域（unit），以便发现数据集的整个分布;
- 若一个 unit 所包含数据点中的一部分超过了输入模型参数，那这个 unit 就是密集的。CLIQUE 方法中，一个聚类被定义为连接的密集 unit 的最大集合。

CLIQUE 方法的操作主要包含两个步骤:

（1）首先，CLIQUE 方法将 $n-$维数据空间划分为不重叠的矩形 unit; 再从中对每一维识别出其中的密集 units。如图-6.12 所示，其中矩形 unit 就是根据 Salary 和 Vacation 所发现 Age 密度。代表这些密集 units 的次空间交叉形成了搜索空间的候选，从中就可以发现高维的密集 units;

从搜索空间候选中识别出真正的密集 units，利用了关联规则中的 Apriori 性质。一般利用有关搜索空间项的先验知识将帮助删除部分搜索空间。CLIQUE 方法所利用的性质就是: 若一个 $k-$维 unit 是密集的，那它在 $(k-1)-$维的投影 unit 也是密集的。这样给定一个 $k-$维候选密集 unit，若它的 $(k-1)-$维的投影 unit 中有不密集的，那么这样一个 $k-$维候选就不会是密集的 unit。因此可以利用所发现的 $(k-1)-$维的密集 unit 来产生 $k-$维的密集 unit 候选。这样所获得的搜索空间会比原来空间小许多。最后依次检查密集 unit 以确定最终的聚类。

（2）CLIQUE 为所获每个聚类产生一个最小描述。具体做法就是: 对每个聚类，确定覆盖连接密集 units 聚类的最大区域; 然后再确定每个聚类的最小覆盖。

CLIQUE 方法能自动发现最高维中所存在的密集聚类。它对输入数据元组顺序不敏感; 也不需要假设（数据集中存在）任何特定的数据分布。它与输入数据大小呈线性关系; 并当数据维数增加时具有较好的可扩展性。但是在追求方法简单化的同时往往就会降低聚类的准确性。

6.8 基于模型聚类方法

基于模型的聚类方法就是试图对给定数据与某个数学模型达成最佳拟合。这类方法经常是基于数据都是有一个内在的混合概率分布假设来进行的。基于模型聚类方法主要有两种: 统计方法和神经网络方法。以下就将介绍这两种方法。

6.8.1 统计方法

机器学习中的概念聚类就是一种形式的聚类分析。给定一组无标记数据对

象，它根据这些对象产生一个分类模式。与传统聚类不同，后者主要识别相似的对象；而概念聚类则更进一步，它发现每组的特征描述；其中每一组均代表一个概念或类，因此概念聚类过程主要有两个步骤：首先完成聚类；然后进行特征描述。因此它的聚类质量不再仅仅是一个对象的函数；而且还包涵了其它因素，如所获特征描述的普遍性和简单性。

大多概念聚类都采用了统计方法，也就是利用概率参数来帮助确定概念或聚类。每个所获得的聚类通常都是由概率描述来加以表示。

COBWEB 是一个常用的且简单的增量式概念聚类方法。它的输入对象是采用符号量（属性-值）对来加以描述的。COBWEB 方法采用分类树的形式来创建一个层次聚类。

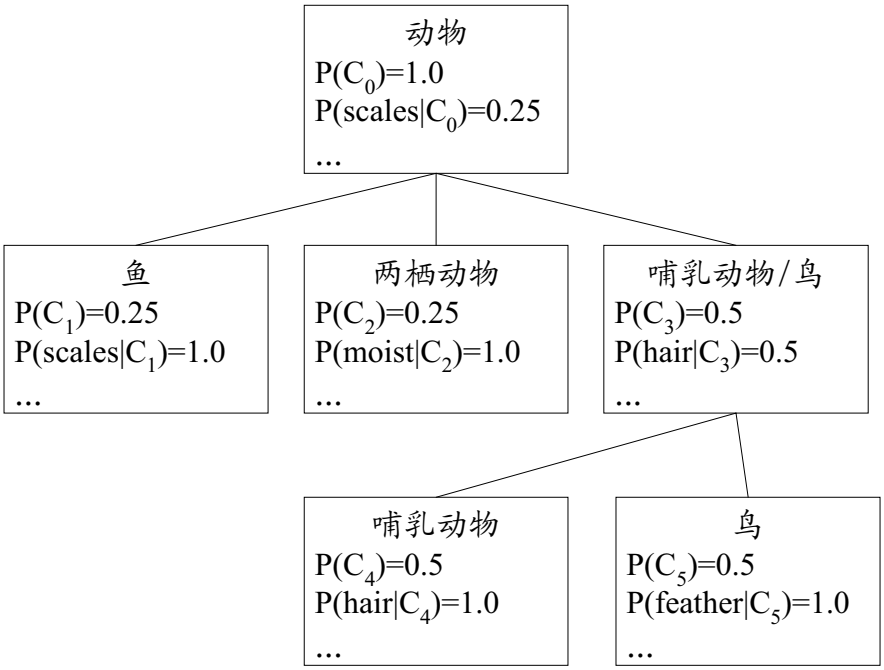


图-6.13 一棵动物分类树示意描述

如图-6.13 所示，就是动物数据的一棵分类树。它与决策树有所不同，前者每个结点均代表一个概念；并包含对（相应结点分类）数据总结概念的一个概率描述。这一概率描述包括：概念的概率和 $P(A_i = V_{ij} \mid C_k)$ 形式的条件概率；这里 $A_i = V_{ij}$ 就是一个属性-值对，而 C_k 就是一个概念类。每个结点都保存累计值以便计算相应的概率。而决策树只是对每个分支所代表的逻辑值（以便进行分类测试）而不是每个结点存放概率描述。一个分类树中的一层兄弟结点形成了一个划分。为利用分类树来对对象进行分类，需要利用一个部分匹配函数沿树“最合适”的

路径走下来。

COBWEB 利用一个启发式评估方法（称为分类能力）来帮助进行树的构造。分类能力（CU）定义如下:

$$\frac{\sum_{k=1}^{n} P(C_k)[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij}^2)]}{n} \qquad (6.18)$$

这里 n 为结点、概念和（在给定层次）构成划分的类别个数。换句话说，给定一个划分（所期望的数值由公式（6.18）中的第一项所表示），以及与没有其它知识（公式中的第二项）时所正确猜出的属性值数目相比，分类能力就是能够猜出的属性值数目的增加值。分类能力表达了聚类中相似性和聚类间不同。

- 聚类间相似性，就是概率 $P(C_k \mid A_i = V_{ij})$。这个值越大，在其它聚类中同样具有该属性-值对的对象就越少。该属性-值对对相应聚类的预测能力就越强;
- 聚类内的相似性，就是概率 $P(A_i = V_{ij} \mid C_k)$。这个值越大，同一聚类具有该属性-值对的对象就越多。该属性-值对对相应聚类内部对象的预测能力就越强。

现在介绍一下 COBWEB 是如何进行处理的。COBWEB 不断将对象插入到分类树中。

COBWEB 沿分类树的一个合适的路径下来，在搜索能够对当前对象进行分类的“最合适”结点时，更新沿途累计值。根据将对象临时存放各结点；计算所获得划分的分类能力来帮助做出决策。最后所放位置应是分类能力最强的。

事实上，COBWEB 也要计算（为这对象）产生一个新结点所获得划分的分类能力，并将其与现存结点的相比较。根据划分所获得的最高分类能力值，当前待插入对象要么放入现成的一个聚类中；要么新建一个聚类。注意 COBWEB 有能力自动调整一个划分中的聚类个数。

以上提到的两个操作对输入对象的顺序非常敏感。COBWEB 又提供了两个附加操作来帮助缓解这一敏感问题。这两个操作是合并和分解，当一个对象被合并时，两个最好的类合并为一个类。COBWEB 还将在现有的聚类中进行分解。这两个操作都是基于分类能力的。合并与分解操作使得 COBWEB 能够进行双向搜索，即合并可以恢复所做的分解操作。

COBWEB 的局限性有以下几点：首先它是基于各属性的概率分布均是相互独立的假设。由于属性间经常存在相互关联，因此这种假设并不总是成立。同时聚类的概率分布表示使得它较难更新和存储聚类。特别是在属性取值非常多的情况下。其原因就是计算时间和空间复杂度并不仅依赖属性数目，而且也与属性取

值的个数相关。此外对于分布异常的数据，所产生的分类树并不一定是平衡的，同时也会导致时间和空间复杂度急剧增大。

CLASSIT 是 COBWEB 的另一个版本。它可以对连续取值属性进行增量式聚类。它为每个结点中的每个属性保存相应的连续正态分布（均值与方差）；并利用一个改进的分类能力描述方法，即不象 COBWEB 那样计算离散属性（取值）和而是对连续属性求积分。但是 CLASSIT 方法也存在与 COBWEB 类似的问题。因此它们都不适合对大数据库进行聚类处理。要想在数据挖掘中应用概念聚类方法还需要进行更多的研究。

6.8.2 神经网络方法

神经网络聚类方法是将每个聚类描述成一个例证（*exemplar*）。每个例证作为聚类的一个“典型”；它不必与一个示例或对象相对应。可以根据新对象与哪个例证最相似（基于某种距离计算方法）而将它分派到相应的聚类中。可以通过聚类的例证来预测分派到该聚类的一个对象的属性。

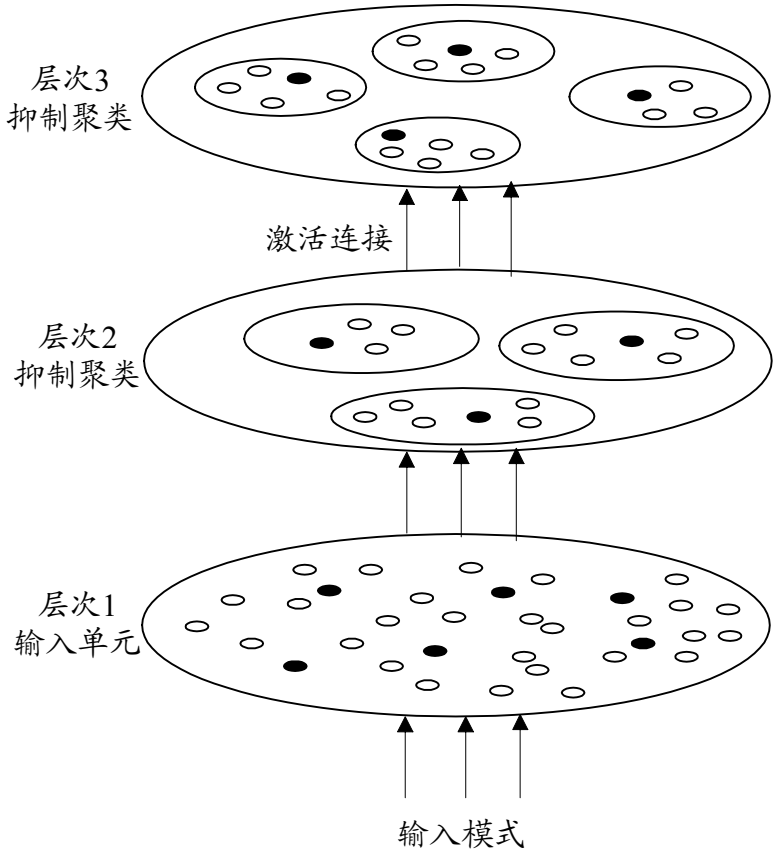


图-6.14 竞争学习结构示意描述

本小节就要讨论神经网络聚类的两种主要方法。第一个方法就是竞争学习方法（*competitive learning*）；第二种就是自组织特征图方法（*self-organizing feature maps*）。两种方法都涉及神经单元的竞争。

竞争学习方法包含一个有若干单元组成的层次结构。这些单元以一种“赢者通吃”方式对所提供给系统的对象进行竞争。如图-6.14 所示，就是一个竞争学习的示例。每个园代表一个单元。一个聚类中取胜的单元被激活（用实心圆表示）；而其它单元仍处于非激活状态（用空心圆表示）。层与层之间的连接是有刺激的，即一个给定层上的单元接受来自低一层所有单元的输入。一个层上激活单元的配置就构成了对高一层的输入模式。在一个给定层上的聚类中单元相互竞争，以响应来自低一层输出的模式。层内的联接是受抑制的以使得一个特定聚类只有一个单元可被激活。获胜的单元调整与同一聚类中其它单元的连接以使得之后可以对类似对象反应更强烈。如果将权值定义为一个例证，那么新对象就被赋给最近的例证。输入参数为聚类个数和每个聚类的单元个数

在聚类结束时，每个聚类能够被认为是一个新的特征，它可以检测出对象中的规律。因此所获得的聚类可以看成是从低层特征到高层特征的一个映射。

在自组织特征图方法中（SOMs），聚类过程也是通过若干单元对当前对象的竞争来完成。与当前对象权值向量最接近的单元成为赢家或激活单元。为变得与输入对象更接近，获胜单元以及最近的邻居调整它们的权值。SOMs 方法假设在输入对象中有一些布局和次序，SOMs 方法将最终利用这些空间中的结构。单元的组织就形成了一个特征图。SOMs 方法被认为是与人脑中的处理过程类似。

神经网络聚类方法与脑处理具有较强的理论联系。但由于存在较长处理时间和复杂数据中复杂关系问题，还需要做更多研究才能使这类方法适合处理大数据库。

6.9 异常数据分析

常常存在与数据模型或数据一般规律不符合的数据对象，这类与其它数据不一致或非常不同的数据对象就称为异常数据（outliers）。

异常数据可能由于测量误差、输入错误或运行错误而造成的。例如：一个人的年龄为-999 就可能是由于程序在处理遗漏数据所设置的缺省值所造成的；或者异常数据也可能是由于数据内在特性而造成的；如：一个公司的首席执行官工资就可能构成一个异常数据（在与其他公司雇员工资相比时）。

许多数据挖掘算法都试图降低异常数据的影响，或全部消除它们。然而这就会导致丢失重要的信息，因为“由于一个人的噪声可能就是一个人的信号”，换

句话说，就是异常数据有时也可能是具有特殊意义的数据。如：在欺诈检测中，异常数据可能就意味着诈骗行为的发生。因此异常数据检测和分析是一个有意义的数据挖掘任务，这一挖掘工作就称为异常挖掘（outlier mining）。

异常挖掘用途很广，如上面所提到的，它可以用于欺诈检测，即检测信用卡使用或电信服务中的异常活动行为；还有通过分析花费较小或较高顾客的消费行为，提出有针对性的营销策略，或在医疗分析中发现多种医疗方案所产生的不同寻常的反应等。

异常挖掘可以描述为：给定 n 个数据对象（或点）和所预期的异常数据个数 k 发现明显不同、意外，或与其它数据不一致的头 k 个对象。异常挖掘问题可以看成是两个子问题：（1）定义在一个数据集中什么样的数据是不一致；（2）找出一个能够挖掘出所定义的异常数据的有效方法。

定义异常挖掘问题是一个较大工作。如果利用一个回归模型来构造相应的数据模型，分析其余数则可以帮助估计数据中的“极端”（情况）。当要从时序数据中发现异常时，由于异常数据可能隐含在趋势、季节性变化或其它周期变化中，从而导致异常挖掘变得更为复杂。在分析多维数据时，可能不是一个而是一组维的取值都较异常。对于非数值（如符号量），这时都要求要认真考虑异常数据的定义。

人的眼睛可以非常有效迅速地发现数据中存在的不一致情况；但由于数据中存在可能周期性（变化）情况，因此一些看起来明显是异常数据的值而在实际情况里可能就是非常正常的数据；而且数据可视化方法在检测符号属性的异常数据，或高维数据时就显得明显不足，其道理很简单，人的眼睛只能有效识别二至三维的数值数据。

本节将要介绍利用计算机检测出异常数据的一些方法。它们可以分为三种：统计类方法、基于距离方法和基于偏差方法。以下将分别介绍这三种方法。值得一提的是，聚类算法将异常数据当作噪声而将其丢弃，因此这里可将聚类算法包括到异常数据检测（其副产品）中。通常用户还需要检查这些方法所发现的每个异常数据以确定它们是否就是异常数据。

6.9.1 基于统计的异常检测方法

基于统计的异常检测方法假设所给定的数据集存在一个分布或概率模型（如一个正态分布）；然后根据相应模型并通过不一致性测试来发现异常数据。应用这种测试需要了解数据集参数的有关知识（如数据分布情况）、分布参数知识（如均值和方差），以及所预期的异常数据个数。

一个统计不一致测试检查两个假设，即一个正面假设和反面假设。一个正面

假设H就是一个描述n个数据对象来自一个分布F，即:

$$H: o_i \in F，\ 其中\ i = 1,2,\ldots,n$$

如果没有重大的统计证据来反驳H，那么H就成立。一个不一致测试就是验证一个对象o_i与分布F关系是否非常大(或小)。根据所获得的不同数据知识，可以有相应不同的不一致测试具体方法。假设选择统计T作为不一致测试方法，对象o_i的统计值为v_i；构造分布T并对重要概率$SP(v_i) = \Pr ob(T > v_i)$进行评估。若有些$SP(v_i)$足够小，那$o_i$就是不一致的，从而拒绝正面假设。一个反面假设$\overline{H}$，它描述$o_i$来自另一个分布$G$，这个结果依赖如何选择$F$模型，因为$o_i$在一个模型为异常数据而在另一个模型中可能就是有效数据。

在决定(不一致)测试效能时，反面模型也是非常重要的。也就是当o_i的确为异常数据时正面假设被拒绝的概率。有几种类型的反面分布:

(1) 内在反面分布。这种情况下，正面假设认为所有来自分布F的对象均被拒绝；而反面假设则描述所有对象来自分布G却成立:

$$\overline{H}: o_i \in G，\ 其中\ i = 1,2,\ldots,n$$

F和G可能是不同的分布，或同一个分布而具有不同的参数。对G的分布形式有所规定以使其有能力产生异常数据。如:它可以有不同的均值或分布。

(2) 混合反面分布。混合反面假设认为不一致的值在分布F中不是异常数据，但被来自其它分布的数据所污染，这种情况下，反面假设就是:

$$\overline{H}: o_i \in (1-\lambda)F + \lambda G，\ 其中\ i = 1,2,\ldots,n$$

(3) 滑动反面分布。这种反面假设认为所有(除了较少部分外)对象独立来自初始分布F(具有参数μ，σ^2)；剩余对象独立来自修改后的分布F(其中的参数有所变化)。

有两种检测异常数据的基本过程:

(1) 块过程，这种情况下，要么所有被怀疑的对象均作为异常数据；要么所有对象均作为一致的;

(2) 序列过程，这种过程的一个例子就是内翻过程。其主要思想就是首先检测最不可能的对象，如果发现其为异常数据；那其它所有更可能的对象均可认为是异常数据；否则在对次不可能的对象(进行检测)，如此下去等等。这一过程比块过程更为有效。

利用统计方法检测异常数据的一个主要不足就是:大多数测试都是针对单个属性的；而由于许多数据挖掘问题需要发现多维空间中的异常数据。此外统计方

法还需要数据集参数的有关知识，如：数据分布（情况）；但在许多情况下，数据分布是未知的。统计方法也不能保证能够发现所有的异常数据，尤其在不采用特别的测试方法，或数据不能被任何标准分布所描述时。

6.9.2 基于距离的异常检测方法

针对统计方法所存在各种问题，人们提出了基于距离的异常检测方法。一个数据集 S 中的一个对象 o 是一个基于距离的异常数据（相对参数 p 和 d ），记为 $DB(p,d)$，它表示：若 S 中至少有 p 部分对象落在距离对象 o 大于 d 的位置；换句话说这里不依赖统计测试；而是将没有足够邻居的对象看成基于距离（检测）的异常数据。这里的邻居则是根据指定对象而定义的；与基于统计方法相比，基于距离异常检测推广或综合了（根据标准分布）不一致测试。所以基于距离的异常数据也称为综合异常。基于距离的异常检测避免了（由于拟合标准分布和选择不一致检测方法所引起的）过度计算。

许多不一致测试表明：若根据一个特定测试，一个对象 o 是一个异常数据，那么对象 o 也是一个 $DB(p,d)$ 异常数据（对于合适的 p 和 d ）。如：若（假设一个正态分布）对象距离均值偏离 3 倍或更多的偏差，那就可以认为是一个异常数据，而这个定义也可以描述为：$DB(0.9988,0.13\sigma)-$异常数据。

目前已提出了一些挖掘基于距离异常数据的有效算法，有关情况介绍如下：

（1） 基于索引的算法。给定一个数据集，基于索引的算法利用多维索引结构，如：$R-tree$，或 $k-d$ 数来帮助搜索每个对象 o 在半径 d 内的近邻。设 M 为一个异常数据 $d-$近邻中的最大对象数。因此一但发现对象 o 近邻数为 $M+1$，就可断定对象 o 不会是异常数据。该算法的最糟情况下的时间复杂度为 $O(kn^2)$，其中 k 为维数；n 为数据集中的对象数。基于索引的算法在 k 维数增加时，也具有较好的可扩展性。然而这个复杂性评估仅考虑了搜索时间，而即使建立索引任务本身也是计算量很大的。

（2） 嵌套循环算法。嵌套循环算法的复杂度与基于索引的算法复杂度相同。但它没有建立索引的时间开销。为了减少 I/O 数量，它将内存分为两部分；又将数据集分为若干逻辑块，通过仔细选择各数据块读入（一半）内存的顺序，来获得较好的 I/O 效率。

（3） 基于单元算法。为避免 $O(n^2)$ 的计算复杂度，又提出了一个基于单元的算法以实现基于内存的数据集处理。它的复杂度为 $O(c^k+n)$，其中 c 为依赖单元数的常数；k 为维数。在该方法中，数据空间被划分为边长为 $d/(2\sqrt{k})$ 的单元。每个单元都有两层面围绕着它。第一层面有一个单元高；第二层面有 $2\sqrt{k}$ 高（近似取最近整数）。算法逐个单元累计异常数据，而不是逐

个对象（进行）。对于一个给定单元，它获得三个累计数：单元中的对象数、单元和第一层单元对象数累计、单元和两层单元数累计。这些累计数分别记为：*cell_count*、*cell_+_1_layer*、*cell_2_layers_count*。

设 M 为一个异常数据 d –近邻中存在的最大异常数据值，确定异常数据具体做法如下：

- 当且仅当 *cell_+_1_layer* 小于等于 M 时，当前单元中的一个对象 o 被认为是异常数据；若这个条件不成立，那么这个单元中的所有对象均被移去（无须再作进一步分析）；
- 若 *cell_2_layers_count* 小于等于 M 时，当前单元中所有对象均认为是异常数据；否则若 *cell_2_layers_count* 大于 M 时，当前单元中的一部分对象可能是异常数据。为检测出这些异常数据，需要采用逐个对象检查方式对当前单元中和第二层中的对象进行检查。只有 d –近邻中对象数小于 M 的对象（包括第一层和第二层）才认为是异常数据。

该算法对于对象数 n 来讲是线性的，并能保证对整个数据集的扫描不超过三次。因此它可用于对大规模数据集的处理；但它对维数的可扩展性并不好。

基于距离的异常检测需要用户设置 p、d 参数。而要发现合适的参数（设置）又涉及了更多的尝试与失败（过程）。

6.9.3 基于偏差的异常检测方法

基于偏差的异常检测没有利用统计测试，或基于距离的方法来识别意外对象；相反它是通过对一组对象特征进行检查来识别异常数据的。偏离（所获）特征描述的对象就认为是异常数据。因此这种方法中的“偏差”就是异常。本小节将要介绍基于偏差异常数据检测的两种方法。第一种是对一组对象进行依次比较；而另一种则利用了 OLAP 数据立方的方法。

（1）顺序意外方法

顺序意外方法同人从一系列假定相似的对象中识别出不寻常的对象方式类似。该方法利用了潜在的数据冗余。给定包含 n 个对象的数据集 S，构造一系列子集 $\{S_1, S_2, \ldots, S_m\}$，其中 $2 \le m \le n$，并有：$S_{j-1} \subset S_j \subseteq S$。

依次对对象间的差异进行评估，其中涉及以下概念：

- 意外集合。该集合就是偏差或异常数据集合。它是根据将其移去所剩（对象构成）集合的差异的最大减少，而得到的最小移去（子）集合。
- 差异函数。差异函数无须计算对象间的距离。它可以为任何函数，只要给定一组对象集，在对象彼此相似时能够返回较小值即可。对象间差异

性越大，函数返回的值就应越大。一个子集的差异性是根据前一个子集的计算结果（增量）计算所获得的。给定一组对象$\{x_1, x_2, \ldots, x_n\}$，一个差异函数可以是集合中数目的变化，也就是:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad (6.19)$$

这里$\bar{x}$为集合中n个数值的均值。对于字符串来讲，差异函数可以采用模式串形式来描述，以概括至今所见过的所有模式。当模式包括S_{j-1}中所有对象却不包括S_j中任何不在S_{j-1}出现的串，这时差异性就要增加。

- 集合的势函数，这是典型累计一个给定集合中对象数目的方法。
- 平滑因子，这是一个依次计算每个子集的函数。它对从当前集合中移去一个子集所减少的差异性进行评估。所得到的值可以利用集合的势进行缩放。那些平滑因子最大的子集就是意外集合。

发现一个意外集合的普通任务就是一个 NP 问题。顺序方法是一个计算可行的方法，它可以实现线性处理时间。

为避免对根据其补集来评估当前子集的差异性，算法选择从集合中选择出一些系列子集进行分析。对于每个子集，它根据序列前一个子集来确定其差异性。

为帮助减轻输入顺序对结果的影响，上述过程需要重复若干次，每次都随机产生一个子集的顺序。在所有循环中，具有最大平滑因子的子集就成为意外集合。

上述方法依赖所使用的差异性计算函数。然而定义差异性计算函数，将由于事先无法知道意外的规律而变得较为困难。根据实际数据库应用已经排除了寻找统一差异性计算函数的可能。在循环次数不多的情况下，算法的时间复杂度为$O(n)$，n为输入对象的数目。这一复杂度是建立在差异性计算是增量进行的，即是根据（序列中）前一子集计算获得后一个（给定）子集的差异性。

（2）OLAP 数据立方的方法

利用 OLAP 数据立方进行异常数据检测的方法就是利用数据立方来识别大型多维数据集中的异常区域。为提高效率，偏差检测过程与立方计算相交叉。因此这种方法又是一种发现驱动的探索形式。其中可利用指示数据意外的计算结果来帮助用户在所有累计层次进行数据分析。如果根据统计模型，立方中一个单元的值与所期望值明显不同，那么就认为该单元就是一个意外单元。如果一个单元涉及概念层次树中的维，那么所期望的值还依赖概念层次树上的祖先。可以利用可视化暗示（如背景颜色）来反映每个单元的意外程度。

例如: 有一个销售数据立方并观察其中每月的销售情况。在可视化暗示的帮助下，可以注意到与其它月份相比，12 月份的销售增加情况。这似乎可以看成

时间维的意外。然而沿着月份进行 drill-down 操作，就可以获得 12 月份每个商品的销售情况（细化）。这时或许还可以观察到在 12 月份其它商品销售也在增长，因此 12 月份销售的增长就不能是一个意外（在考虑商品维之后）。由于搜索空间很大，单靠手工检测是较难发现这样的意外的。特别是在维数较多而又涉及概念层次树（具有多层）的情况下。

6.10 本章小结

- 一个聚类就是一组数据对象的集合；集合内各对象彼此相似；各集合间的对象彼此相差较大。将一组物理或抽象对象中类似的对象组织成若干组的过程就称为聚类过程。
- 聚类分析具有广泛的应用，其中包括：市场营销、顾客分类、模式识别、生物研究、空间分析、Web 文档分类等。聚类分析既可以用于数据挖掘工具以获得数据分布内在规律；或作为其它数据挖掘的预处理步骤。
- 聚类质量可以根据对象差异性计算结果进行评估。可以对不同数据类型进行计算，其中包括：间隔数值属性、二值属性、符号属性、顺序属性和比例数值属性，或是这些类型的组合。
- 聚类分析是数据挖掘中的一个很活跃的研究领域，并提出了许多聚类算法。这些算法可以被分为划分方法、层次方法、基于密度方法、基于网格方法和基于模型方法。
- 划分方法，首先创建 k 个划分，k 为要创建的划分个数；然后利用一个循环定位技术通过将对象从一个划分移到另一个划分来帮助改善划分质量。典型的划分方法包括：$k-means$、$k-medoids$、CLARANS 和它们的改进版本。
- 层次方法，创建一个层次以分解给定的数据集。该方法可以分为自上而下（分解）和自下而上（合并）两种操作方式。为弥补分解与合并的不足，层次合并经常要与其它聚类方法相结合，如循环定位。典型的这类方法包括：BIRCH、CURE 和 CHEMALOEN。
- 基于密度方法，根据密度完成对象的聚类。它根据对象周围的密度（如 DBSCAN）不断增长聚类。典型的基于密度方法包括：DBSCAN 和 OPTICS。
- 基于网格方法，首先将对象空间划分为有限个单元以构成网格结构；然后利用网格结构完成聚类。STING 就是一个利用网格单元保存的统计信息进行基于网格聚类的方法。而 CLIQUE 则是一个将基于网格与基于密度相结合的方法。
- 基于模型方法，它假设每个聚类的模型并发现适合相应模型的数据。典型的

基于模型方法包括：统计方法 COBWEB 和神经网络方法 SOM。

- 异常数据检测与分析在欺诈检测、营销定制、医疗分析等诸多领域有着广泛的用途。利用计算机进行异常数据分析方法主要包括：基于统计的方法、基于距离的方法和基于偏差的方法。

参考文献

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data, pages 94~105,Seattle, Washington, June 1998.

[2] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data, pages 49{60, Philadelphia, PA, June 1999.

[3] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In Proc.1996 Int. Conf. Data Mining and Knowledge Discovery (KDD'96), pages 164~169, Portland, Oregon, August 1996.

[4] V. Barnett and T. Lewis. Outliers in Statistical Data. John Wiley & Sons, 1994.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In Proc. 1990 ACM-SIGMOD Int. Conf. Management of Data, pages 322~331, AtlanticCity, NJ, June 1990.

[6] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U.M. Fayyad,G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 153~180. AAAI/MIT Press, 1996.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD'96), pages 226~231, Portland,Oregon, August 1996.

[8] M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In Proc. 4th Int. Symp. Large Spatial Databases (SSD'95), pages 67~82, Portland,Maine, August 1995.

[9] D. Fisher. Optimization and simplification of hierarchical clusterings. In Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95), pages 118{123, Montreal, Canada, Aug. 1995.

[10] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data, pages 73~84, Seattle, Washington, June 1998.

[11] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In Proc.1999 Int. Conf. Data Engineering, pages 512~521, Sydney, Australia,

March 1999.

[12] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98), pages 58~65, New York, NY, August 1998.

[13] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 2:283~304, 1998.

[14] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Printice Hall, 1988.

[15] G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. COMPUTER, 32:68~75, 1999.

[16] E. Knorr and R. Ng. A unified notion of outliers: Properties and computation. In Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97), pages 219~222, Newport Beach, California, August 1997.

[17] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 392~403, New York, NY, August 1998.

[18] T. Kohonen. Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59~69,1982.

[19] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. Computational Statistics and Data Analysis, 19:191~201, 1995.

[20] M. Li and P. Vitanyi. Applied Multivariate Statistical Analysis. New York: Springer Verlag, 1991.

[21] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In Proc. 1994 Int. Conf. Very Large Data Bases, pages 144~155, Santiago, Chile, September 1994.

[22] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In Proc. Int.Conf. of Extending Database Technology (EDBT'98), pages 168~182, Valencia, Spain, March 1998.

[23] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 428~439, New York, NY, August 1998.

[24] W. Wang, J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. In Proc. 1997 Int. Conf. Very Large Data Bases, pages 186{195, Athens, Greece, Aug. 1997.

[25] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data, pages 103~114, Montreal, Canada, June 1996.