第五章 聚类分析

聚类是将物理或抽象对象的集合划分成为有类似的对象组成的多个属类的过程。聚类分析按照一定的算法规则，将判定为较近和相似的对象，或具有相互依赖和关联关系的数据聚集为自相似的组群，构成不同的簇。从实际应用的角度看，聚类分析是数据挖掘的主要任务之一，在工业数据的分析方面有着广泛的应用。例如，在估算城市电力消费量、流程工业故障研究、工业负荷预测等方面具有较高的实际应用价值。

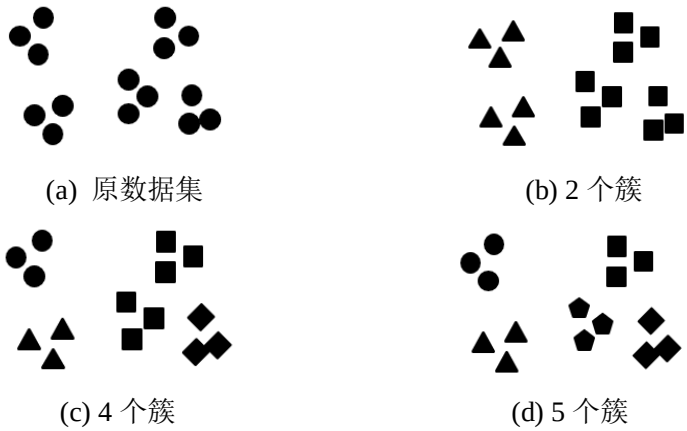
本章首先介绍聚类的基本理论，阐述对聚类算法性能的要求。然后从基于划分、层次和密度三个角度对聚类分析的方法进行介绍和实例解析。最后介绍聚类结果性能评估方法和指标。

5.1 聚类分析的基本概念

聚类是一个把数据集划分成多个组或者簇的过程，使得同一个簇中的对象具有较高的相似性，但不同簇的对象却很不相似。聚类与分类不同，聚类的类别取决于数据本身，而分类的类别有数据分析人员预先定义好的。使用聚类算法的用户不但需要深刻地了解所用的特殊技术，而且还要知道数据收集过程的细节。

5.1.1 聚类分析的定义

聚类就是将物理或抽象对象的集合分成由类似的对象组成的多个类或者簇的过程。通常，利用对象之间的相似度或距离作为度量方法。由聚类所生成的簇是一组数据对象的集合，这些对象与同一簇中对象相似度较高或距离较近，与其他簇中的对象相似度较低或距离较远。这种分析事物聚类的过程称为聚类分析，又称群分析。图 5.1 显示了一个数据集对象划分成簇的 3 种不同方法。



(a) 原数据集　　(b) 2 个簇

(c) 4 个簇　　(d) 5 个簇

图 5.1 相同数据集的不同聚类方法

聚类分析方法可以应用在数据挖掘的各个过程中，例如在数据预处理中，可

以针对数据需求，对于数据结构简单或者与运量分析有单属性和较少属性关联的数据可以在经过数据清理等预处理后直接整合如数据仓库，而对于复杂结构的多维数据可以通过聚类的方法将数据聚集后构造出逻辑库，使复杂结构数据标准化，为某些数据挖掘方法提供预处理。

在实际工业数据处理中，聚类方法的应用也很广泛。如对工业数据进行工况识别、异常检测，使得工业生产能够顺利执行，同时也被应用于故障诊断、故障预测、负荷预测等方面。图 5.2 显示了通过聚类方法检测风电功率数据异常点的过程。

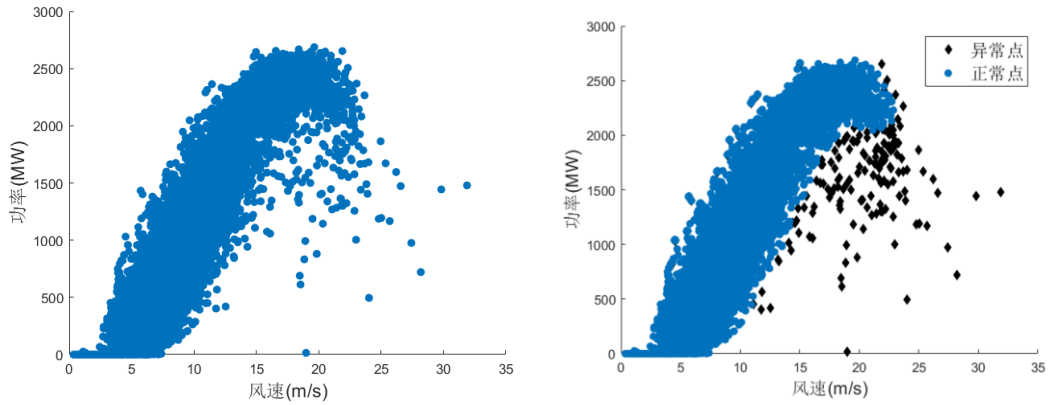


图 5.2 风电功率数据异常点检测

典型的聚类过程首先需要选择合适特征类型的某种距离函数或者构造新的距离函数进行相似度的计算，而后执行聚类。在实际应用中，有许多算法可以实现聚类任务，例如划分聚类、层次聚类、密度聚类等，将分别在 5.2-5.4 小节中详细介绍。

5.1.2 聚类算法的性能要求

不同的聚类算法有不同的应用背景，有的适合于大数据集，可以发现任意形状的簇；有的算法思想简单，适用于小数据集。总的来说，它的一些潜在的应用对分析算法提出了特别的要求，包括：

（1）伸缩性：许多聚类算法在小于几百个数据对象的小数据集上运行良好，但是在包含数百万甚至数十亿的大型数据库运行时可能会导致有偏差的结果。因此，要求聚类算法具有高度的可伸缩性，使得聚类结果准确度不会因为数据量的大小而变化。

（2）处理不同字段类型的能力：实际上我们处理的数据类型是多样的，处理常见的数值型，还有二元的、对称的、序数的，甚至是图、序列、文档等。这就要求聚类算法不仅要处理数值型的字段，还要有处理其他类型字段的能力。

（3）发现具有任意形状的聚类的能力：许多聚类方法基于欧几里得或曼哈

顿距离度量来量化对象的相似度。基于这些距离度量的算法往往只能发现相似尺寸和密度的球状簇或者凸型簇。然而，一个簇可能是任意形状的，故要求算法有发现任意形状的聚类的能力。

（4）能够处理异常数据：大部分数据集常常包含异常数据，例如离群点、缺失值、未知或错误的数据。一些聚类算法对这样的异常数据敏感，会导致低质量的聚类结果。因此，要求聚类算法能够处理异常数据。

（5）增量聚类和对输入次序的不敏感：一些聚类算法不能将新加入的数据快速插入到已有的聚类结果中，或是针对不用次序的数据输入，产生的聚类结果差异很大。因此，要求聚类算法对增量聚类和对输入次序不敏感。

（6）处理高维数据的能力：数据集可能包含大量的维或属性。一些聚类分析算法只对处理维数较少的数据集时效果较好，但是对于高维数据的处理能力很弱，高维空间中的数据十分稀疏，且高度倾斜，即聚类结果的形状极其不规则。

（7）初始化参数的需求最小化：很多算法需要用户提供一定个数的初始参数，比如期望的簇个数，簇初始中心点的设定。聚类的结果对这些参数十分敏感，调参数需要大量的人力负担，也非常影响聚类结果的准确性。故而，我们希望能将初始化参数的需求最小化。

（8）可解释性和可用性：聚类的结果最后都是要面向用户的，所以结果应该是容易解释和理解的，并且是可应用的。故而要求聚类算法必须与一定的语义环境及语义解释相关联。

基于以上的要求，对聚类算法进行比较和研究时，应考察以下几方面的问题：

（1）算法是否适用于大数据量，算法的效率是否满足大数据量、高复杂性的要求。

（2）是否能够应对不同的数据类型，能否处理符号属性。

（3）是否能发现不同类型的聚类。

（4）是否能处理脏数据或异常数据。

（5）是否对数据的输入顺序不敏感。

聚类算法是否能满足上述要求，处理效能如何，可以通过对簇的评估来进行评价。聚类分析性能评估方法将在 5.5 小节详细介绍。

聚类算法的选择取决于数据的类型、聚类的目的。可以对同样的数据尝试多种聚类算法并比较结果，以发现数据可能揭示的结果。主要聚类算法有：基于划分的方法、基于层次的方法和基于密度的方法等。下面对这三种聚类方法进行详细介绍。

5.2 划分聚类方法

聚类分析最简单、最基本的版本是划分，它把对象组织成多个互斥的组或簇。给定一个含有N个对象的数据集，以及要生成的簇的数目K。每一个分组就代表

一个聚类，K < N。这K个分组至少包含一个数据记录，每一个数据记录有且仅有一个所属分组。对于给定的K，算法首先的任务是将数据构建成K个划分，以后通过反复迭代从而改变分组的重定位技术，使得每一次改进之后的分组方案都较前一次好。将对象在不同的划分移动，直至满足一定的准则。一个好的划分一般准则是：在同一个簇中的对象尽可能“相似”，同簇中的对象尽可“相异”。划分聚类的含义就是简单地将数据对象集划分成不重叠的子集，使得每个数据对象恰在一个子集，如图 5.3 所示。

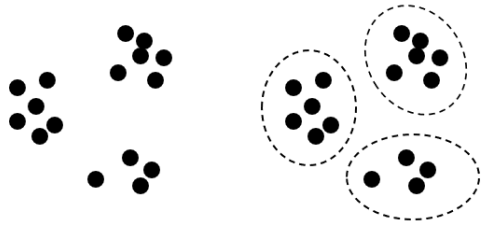


图 5.3 简单的划分聚类

在划分聚类算法中，最经典的就是 K-means 算法和 K-medoids 算法，在这两个算法的基础上衍生出了很多改进方法，适应不同蕨类分析任务、数据类型等的需求。

5.2.1 K-means 算法

K-means 算法（K-means clustering algorithm），也称为 K-平均或 K-均值算法，由 J. B MacQueen 于 1967 年提出，是最常用也最为经典的一种基于划分的聚类算法，在图像分析、市场研究、生物信息学、医学信息学等领域得到了广泛的应用。

该算法的主要思想是：随机从数据集中选取K个点作为初始聚类中心，然后计算各个样本到聚类中心的距离，通过距离大小把样本归到离它最近的聚类中心所在的类。计算新形成的每个聚类的数据对象的平均值来得到新的聚类中心，调整到相邻两次的聚类中心没有任何变化为止。

该算法的特点是在每次迭代中都要考察每次样本的分类是否正确。若不正确，就要调整，在全部样本调整完后，再修改聚类中心，进入下一次迭代。如果在一次迭代算法中，所有的样本被正确分类，则不会有调整，聚类中心也不会有任何变化，算法结束。

K-means 算法的伪代码如下：

输入：簇的个数 K 和数据集 D
输出：K 个簇的集合
1. 从 D 中随机选择 K 个点作为初始簇中心

2. repeat
3. 根据簇中对象的均值，将每个点分配到最近的簇中，形成 K 个簇
4. 更新均值，重新计算每个簇的质心
5. until 质心不发生变化

K-means 算法是典型的基于距离的聚类算法，通常采用欧式距离衡量数据对象与聚类中心之间的相似度。根据应用场合的不同，也可以选择其他的相似性度量方法，如对于文本，采用余弦相似度或者 Jaccard 系数的效果更好。

在 K-means 算法中，每一轮迭代完成后，都需要判断聚类结果是否收敛。为此，通常会定义一个准则函数（也称为目标函数），其中最常用的误差平方和函数（Sum of the Squared Error）,定义如下：

$$SSE=\sum_{i=1}^{k}\sum_{x\in C_i} d(x,c_i)^2 \tag{5.1}$$

其中，x是集合D的对象，C_i代表第i个簇，c_i是C_i的中心，$c_i=\frac{1}{m_i}\sum_{x\in C_i} x$，$m_i$是$C_i$中心数据对象的个数。K-means 算法迭代执行过程中，SSE 的值会不断减小。当前后两轮迭代所得到的 SSE 保持不变，或者二者之间的差异小于某个预设的门限值ε，就可以认为算法已收敛。

例 5.1 在工业生产过程中，生产加工后铣床的刀具的磨损程度与电机电流、主轴振动等操作条件有关。聚类方法是根据电机电流和主轴振动分析铣床刀具磨损程度的常用方法。如表 5.1，属性 1 为铣床电流传感器的采集数据，属性 2 为铣床振动传感器的采集数据。试用 K-means 算法聚类将这些对象划分成 3 个簇。

表 5.1 样本数据点

点	属性 1	属性 2	点	属性 1	属性 2	点	属性 1	属性 2
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.800	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

在聚类之前，先绘制数据点的散点图，直观地展示出这些散点的分布情况。图 5.4 为整个聚类过程示意图，每个虚线框中的散点表示在同一个簇中。

（1）根据 K-means 算法步骤，首先任意选择 3 个对象（点 1、点 2、点 3）

作为初始的簇中心，其中簇中心用“×”标记。根据与簇中心的距离（假设采用欧式距离），每个对象被分配到最近的一个簇。

（2）更新簇中心。即根据簇中的当前对象，重新计算每个簇的均值。使用这些新的簇中心，把对象重新分布到里簇中心最近的簇中。

重复这一过程，直到对象的重新分配不再发生，处理结果结束，输出聚类结果。

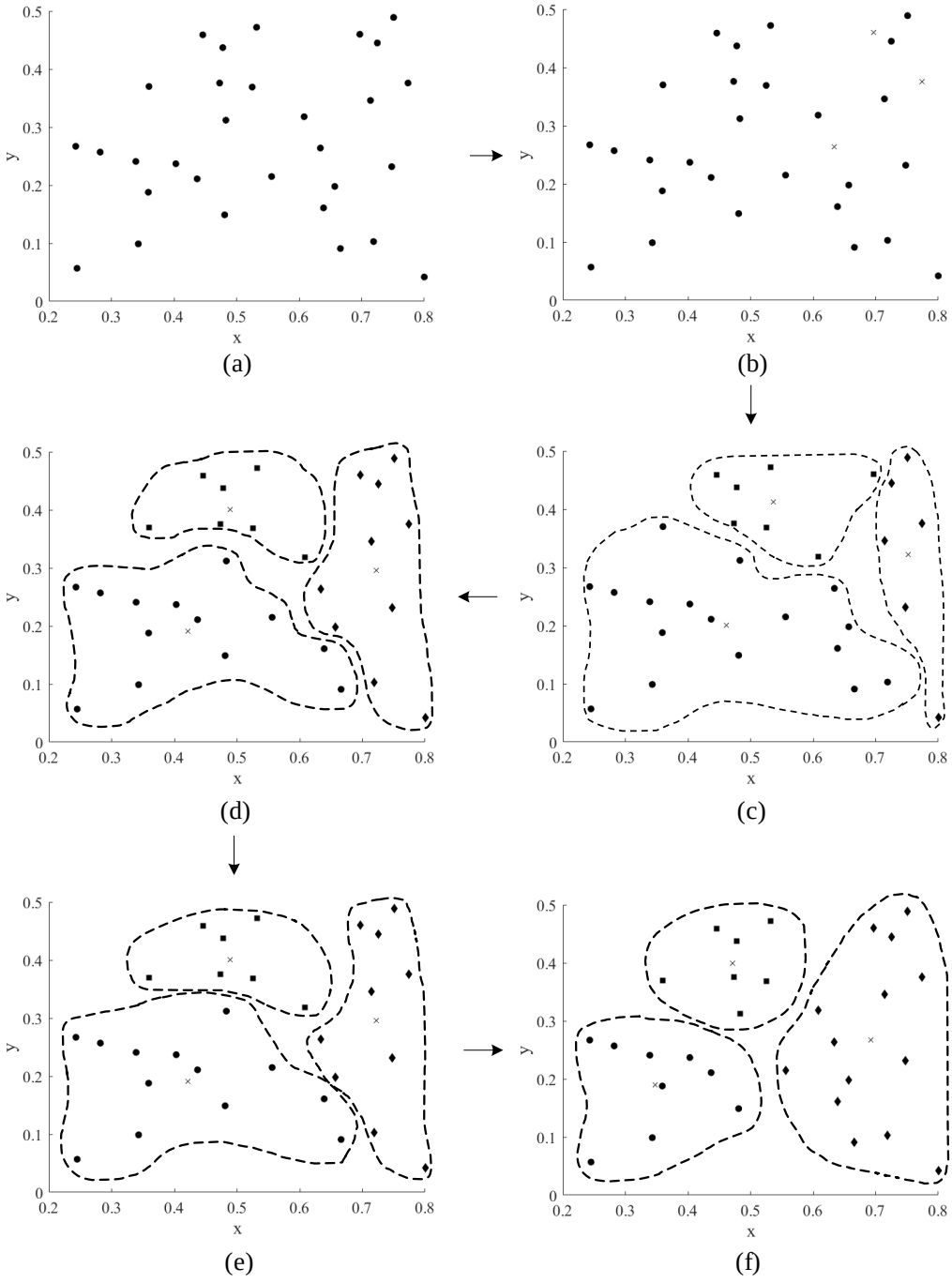


图 5.4 K-means 聚类过程

K-means 算法的优点是计算简单、快速，算法的时间复杂度和空间复杂度都相对较低，能处理大型数据集，结果簇相当紧凑，并且簇和簇之间明显分离。其中，空间复杂度为$O((m+k)d)$，这里m是数据对象的个数，d是数据的维度，k是分簇个数；时间复杂度为$O(tkmd)$，t是迭代次数。通常有$k \ll m, d \ll m, t \ll m$，即算法执行时间基本与数据集的规模呈线性关系，因此处理大数据集时的效率较高。

但 K-means 算法也存在不足，概括如下：

（1）算法要求用户实现给出分簇的数目K，如果用户没有关于数据集的先验信息，K值的估计将非常困难。

（2）算法对初始聚类中心的选取较为敏感，不同的初始值会造成不同的聚类结果。当初始聚类中心的选择不合适时，会导致产生局部最优解，无法获得全局最优解和较小的 SSE。

（3）K-means 算法只有在簇的均值被定义的情况下才能使用，而无法直接处理离散型数据。

（4）K-means 算法采用欧式距离度量数据对象之间的相似性，以误差平方和最小化为优化目标，导致该算法对球状簇有较好的聚类效果，却不适合非球状的簇。

（5）对于离群点敏感。由于 K-means 算法是以均值作为聚类的中心，在计算时容易受到离群点的影响造成中心偏移，产生聚类分布上的误差。

对于最佳聚类簇 K 的选取方法，常用肘部法判断。肘部法利用观察 K 与 SSE 的变化获得，SSE 随着 K 的增大而减小，并且在达到第一个临界转折点处，SSE 减小速度变缓，这个临界转折点就可以考虑为最佳聚类性能的点，该点对应的 K 就是最佳聚类数。如图 5.5 所示，考虑最佳聚类数为 2。

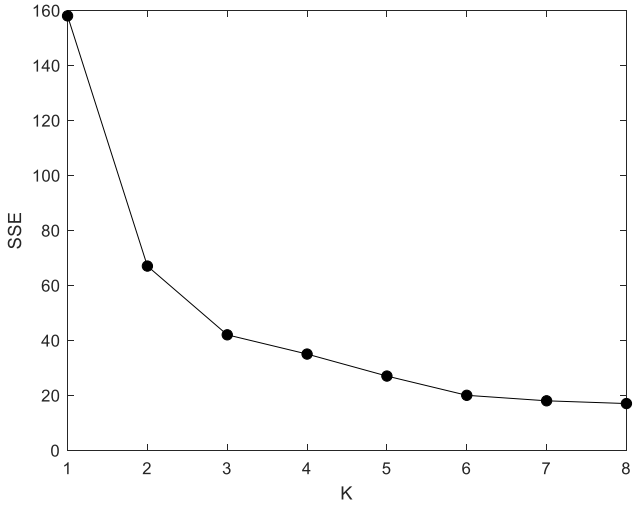


图 5.5 肘部法选取最佳聚类数

5.2.2 K-medoids 算法

K-medoids 算法也称为 K-中心点算法，是基于“代表对象”的聚类方法，它在 K-means 算法基础上，利用簇的中心点代替均值点作为聚类中心，然后根据各数据对象与这些聚类中心之间的距离之和最小化的原则，进行簇的划分。K-medoids 算法实质上是对 K-means 算法的改进和优化，具有对离群点不敏感、可以处理离散型数据的优势。

K-medoids 算法的伪代码如下：

输入：簇的个数 K 和数据集 D
输出：K 个簇的集合

1. 从 D 中随机选择 K 个点作为初始代表对象
2. Repeat
3. 根据与中心点最近的原则，将剩余点分配到当前最佳的中心点代表的簇中
4. 在每一个簇中，计算每个样本点对应的准则函数，选取准则函数最小时对应的点作为新的中心点
5. Until 中心点不发生变化或达到设定的最大迭代次数

PAM（Partitioning Around Medoid）是聚类分析算法中划分法的一个聚类方法，是最早提出的 K-medoids 之一。该算法的基本思想为：选用簇中位置最中心的对象，试图对n个对象给出K个划分。其中，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，当前某个对象被一个非代表对象替代后，数据集中所有其他的数据对象需要重新归类，并计算交换前后各数据点与所在簇的中心点之间的距离差，以此作为该点因簇的代表变化而产生的代价。交换的总代价是所有对象的代价之和。如果总代价为负值，则表明交换后的类内聚合度更好，原来的代表对象将被替代；如果总代价为正值，则表明原来的代表对象更好，不应被取代。为了确定任意一个非代表对象c_{random}是否可以替换当前的某个代表对象c_j，需要对数据集中的所有非代表对象x进行检查，以确定x是否要重新归类。根据x位置的不同，分为 4 种情况，如图 5.6 所示。

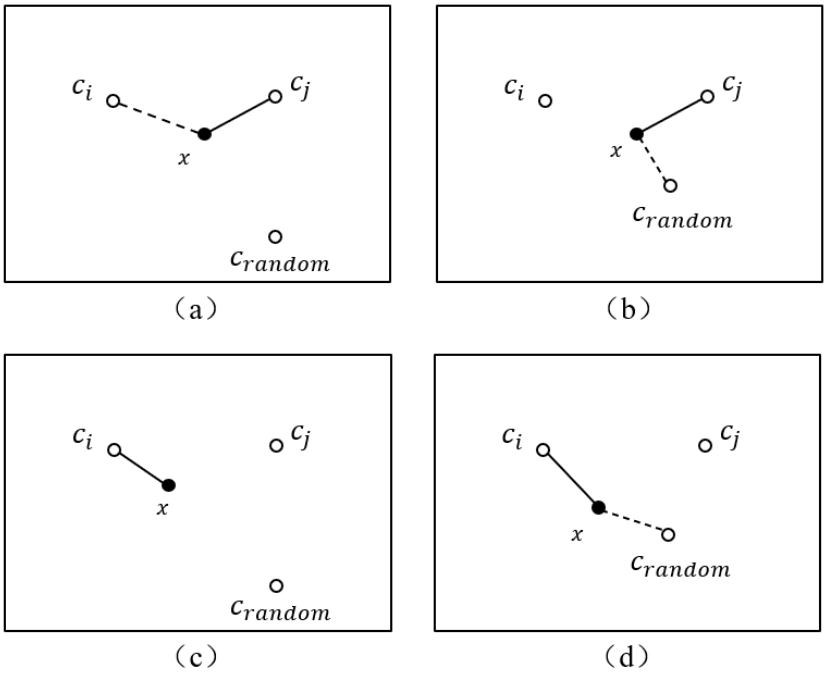


图 5.6 交换后数据对象重新归类的四种情况

情况 1(见图 5.6a)：对象x当前属于以c_j，为中心点的簇，如果用c_{random}替换c_j代表对象，x将更接近其他簇的中心点c_i，那么就将x归类到以c_i为中心点的簇中。

情况 2(见图 5.6b)：对象x当前属于以c_j为中心点的簇，如果用c_{random}替换c_j作为新的代表对象，x将更接近c_{random}，那么就将x归类到以c_{random}为中心点的簇中。

情况 3(见图 5.6c)：对象x当前属于以c_i为中心点的簇，如果用c_{random}替换c_j作为新的代表对象，x仍然最接近c_i，那么x的归类将保持不变。

情况 4(见图 5.6d)：对象x当前属于以c_i为中心点的簇，如果用c_{random}替换c_j作为新的代表对象，x将更接近c_{random}，那么就将x归类到以c_{random}为中心点的簇中。

PAM 算法在每次迭代中都需要交换每个簇的代表对象与该簇中的所有非代表对象，从而得到可以改善聚类质量的候选对象集，然后再将这些候选对象作为下一次迭代的代表对象。每次迭代的复杂度为$O(K(n-K)^2)$，其中n是数据对象的总数，K是分簇数量。显然，当数据对象与簇的数目增大时，PAM 算法的执行代价很高，因此有许多算法针对 PAM 进行修改以适用于大数据集，如 CLARA。

CLARA(Clustering LARge Applications)是 PAM 算法的扩展，它不考虑整个数据集，而是利用抽样的方法，选择数据的一小部分作为样本，能够有效处理大规模数据所带来的计算开销和 RAM 存储问题。该算法的思想就是通过在大数据中进行随机抽样，然后对每个抽样的样本使用 PAM 算法，最后在每个样本聚类

出的最佳中心点中寻找一个代价最小的聚类中心作为当前的大数据样本的最佳聚类。

CLARA 算法的有效性依赖于样本的大小、分布及抽样质量。如果样本包含的数据点过少，可能会没有把全部的最佳聚类中心都选中，这样就不能找到最好的聚类结果。也就是说，如果样本发生偏斜，基于样本的聚类效果好，不一定代表整个数据集的聚类效果好。反之，如果样本的数据量很大，虽然能够提升聚类质量，但是算法效率会明显下降，因为 CLARA 算法每一轮迭代的计算复杂度是$O(ks^2 + k(n-k))$，其中s是样本中的对象个数。

例 5.2 在例 5.1 的对象集合加入离群点 31，坐标为（1,1）。利用 K-medoids 算法与 K-means 算法聚类，并比较聚类结果。

根据 K-medoids 算法思想，聚类过程如图 5.7 所示，聚类步骤如下：

（1）首先任意选择 3 个对象作为初始的簇中心，用“×”标记。根据与中心点距离最近的原则，将剩余点分配到当前最佳的中心点代表的类中，并计算当前代价。

（2）更新簇中心。随机选择一个非簇中心对象作为新的簇中心，并计算用新簇中心的总代价。如果新簇中心的总代价小于旧簇中心的总代价，则用新的簇中心替换旧的簇中心。

重复这一过程，直到对象的重新分配不再发生，处理结果结束，输出聚类结果。聚类结果如表 5.2 所示。

表 5.2 K-medoids 算法聚类结果

簇	簇中心	簇中对象
1	1	{1,2,22,24,26,27,29,30,31}
2	6	{6,7,8,10,11,12,15,18,19,20,23,25,28}
3	14	{3,4,5,9,13,14,16,17,21}

根据例 5.1 中 K-means 算法对该对象集合聚类，可得到聚类结果如表 5.3 所示。

表 5.3 K-means 算法聚类结果

簇	簇中心	簇中对象
1	(0.7410,0.5732)	{1,26,27,29,31}
2	(0.6832,0.2133)	{2,3,4,5,9,13,14,16,17,21,22}
3	(0.3931,0.2686)	{6,7,8,10,11,12,15,18,19,20,23,24,25,28,30}

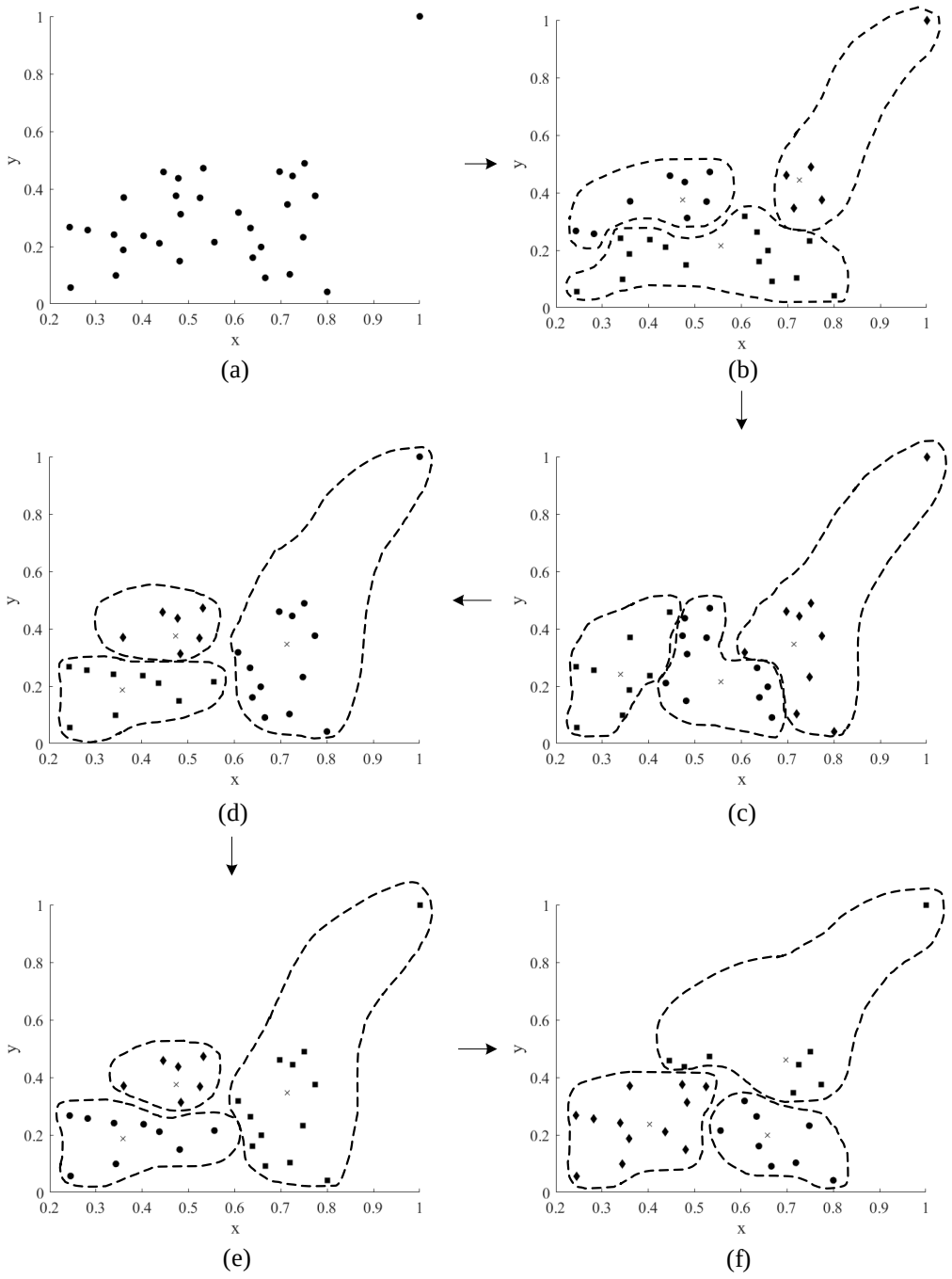


图 5.7 K-medoids 聚类过程

根据式(5.1)分别求得 K-medoids 算法聚类后的 SSE=4.0014，K-means 算法聚类后的 SSE=4.3207。从两种聚类结果来看，例如点 30，由于离群点 31 的影响 K-means 将该点划分到了簇 3 中，使得簇中心向离群点偏移而导致簇中心显著偏离簇中成员且 K-means 算法聚类后 SSE 较大。由此可知，对于含有离群点的数

据集，使用 K-medoids 算法聚类可以减少 K-means 算法聚类受到离群点影响而导致聚类效果变差的不良影响。

5.2.3 K-means++算法

K-means++（K 均值++）算法是 K-means 的改进，其主要区别在于初始化确定初始聚类中心。K-means 算法是随机确定初始聚类中心，而 K-means++则是根据当前样本点到簇中心的距离计算该样本点成为下一个聚类中心的概率，公式如下：

$$P(x)=\frac{\sum_{c_i\in C} d(x,c_i)^2}{\sum_{c_i\in C}\sum_{i\in X} d(i,c_i)^2} \tag{5.2}$$

其中，x为当前样本点，c_i代表第i个簇中心，C代表已确定的簇中心集合，X为样本点总数。

距离越大则概率越大，然后根据概率大小抽取下一个聚类中心，不断重复直至抽取 K 个聚类中心。选出初始聚类中心后，继续使用标准的 K-means 算法进行聚类。

K-means++的伪代码如下：

输入：簇的个数 K 和数据集 D
输出：K 个簇的集合

1. 从 D 中随机选择一个样本作为簇中心c_0
2. 计算并逐个选出所有簇中心
3. repeat
4. 根据簇中均值，分配对象到最相似的簇中更新均值，重新计算每个簇的质心
5. 更新簇均值，即重新计算每个簇中均值
6. until 质心不发生变化

K-means++算法由于其选择初始聚类中心的方法，可以解决 K-means 算法对于初值敏感的问题。

例 5.3 根据例 5.1 的数据集，试利用 K-means++算法把数据集聚成 3 类，并与 K-means 聚类结果进行比较。

K-means++算法聚类过程于图 5.8 显示，聚类步骤如下：

（1）随机选择一个簇中心（点 1）。利用公式(5.2)计算出其他样本点成为下一个簇中心的概率，计算结果如表 5.4 所示。

表 5.4 各点选为第二个簇中心的概率

点	1	2	3	4	5	6	7	8	9	10
概率	0	0.013	0.024	0.019	0.033	0.043	0.044	0.042	0.043	0.057
点	11	12	13	14	15	16	17	18	19	20
概率	0.070	0.058	0.035	0.031	0.040	0.050	0.041	0.050	0.049	0.053
点	21	22	23	24	25	26	27	28	29	30
概率	0.027	0.013	0.030	0.025	0.023	0.007	0.019	0.028	0.004	0.029

根据表 5.4 结果显示，概率最大点为点 11，故选择点 11 为第二个簇中心点。同理，通过计算，得出第三个簇中心点为点 16。

初始化簇中心分别为点 1、点 11、点 16，簇中心用“×”表示。根据与簇中心的距离（假设采用欧式距离），其他对象被分配到最近的一个簇。

（2）更新簇中心。即根据簇中的当前对象，重新计算每个簇的均值。使用这些新的簇中心，把对象重新分布到里簇中心最近的簇中。

重复这一过程，直到对象的重新分配不再发生，处理结果结束，输出聚类结果。聚类结果如表 5.5 所示。

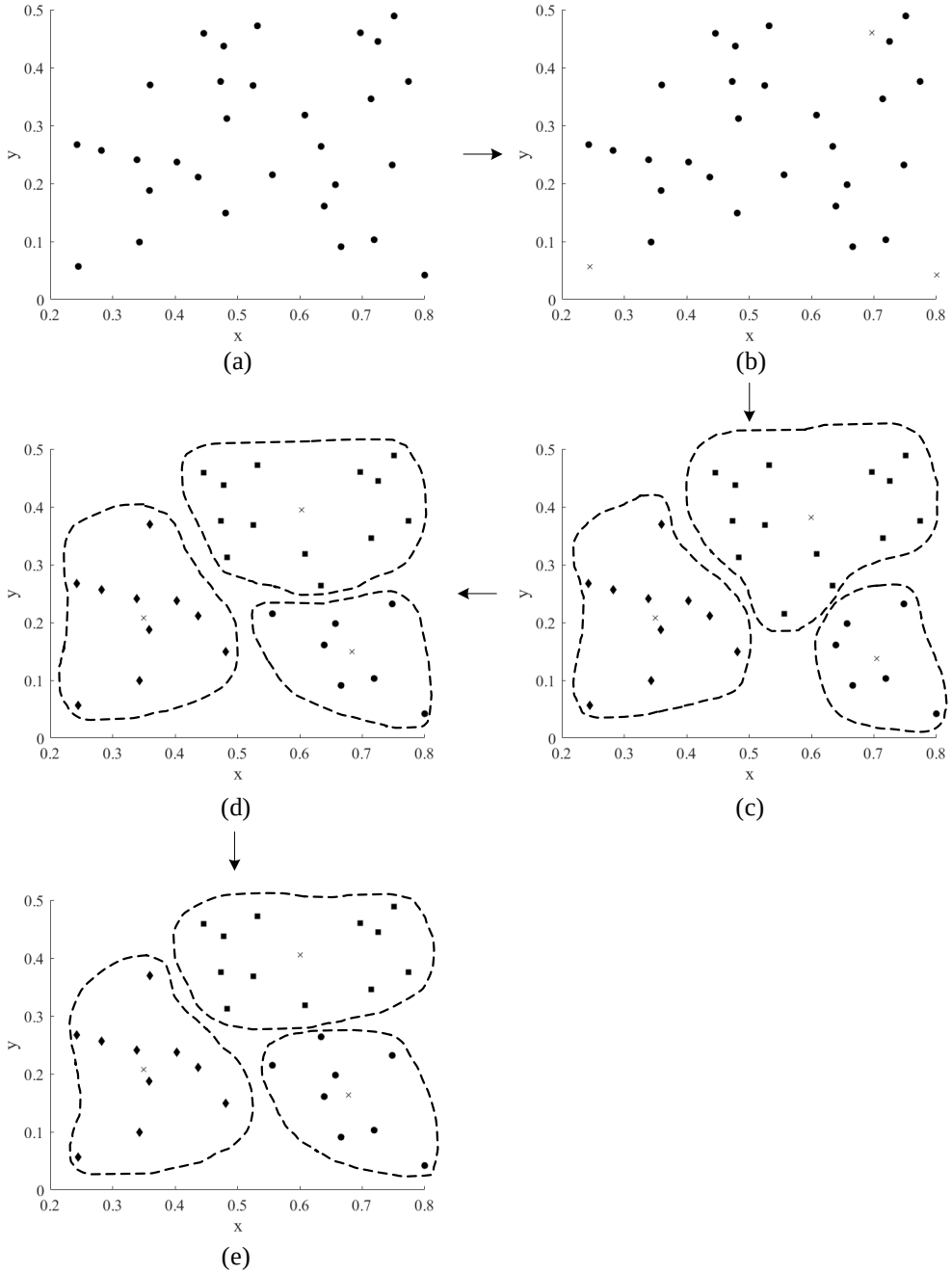


图 5.8 K-means++聚类过程

表 5.5 K-means++算法聚类结果

簇	簇中心	簇中对象
1	(0.6005,0.4049)	{1,2,4,22,23,24,25,26,27,28,29,30}
2	(0.3492,0.2076)	{6,7,8,10,11,12,15,18,19,20}
3	(0.6774,0,1633)	{3,5,9,13,14,16,17,21}

5.3 层次聚类方法

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底向上的凝聚聚类和自顶向下的分类聚类。凝聚聚类方法最初将每个对象作为一个簇，然后通过逐步合并相近的簇从而形成越来越大的簇，直至所有的对象都在一个簇里，或者满足一定的终止条件为止。分裂聚类方法恰好相反，初始时是将所有的对象置于一个簇中，然后逐步将其细分为较小的簇，直到每个对象自成一个簇，或者满足一定的终止条件为止。

层次聚类的步骤可总结如下：

（1）将每个对象归为一类，共得到N类，每类仅包含一个对象。类与类之间的距离就是它们包含的对象之间的距离。

（2）找到最接近的两个类合并成一类，于是总的类数减少一个。

（3）重新计算新类与所有旧类之间的距离。

（4）重复（2）、（3），直到最后合并成一个类为止（此类包含了N个对象）。

层次聚类是聚类算法常常用树状图表示，图 5.9 所示是一个简单的例子。图中最顶部的根节点表示整个数据集，中间节点代表有若干个对象构成的子簇，最底部的叶子节点代表数据集中的对象，子簇还能进一步划分为更小的簇，在图中对应为该节点的子节点。

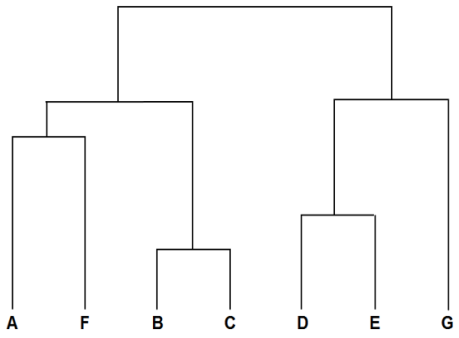


图 5.9 层次聚类的树状图

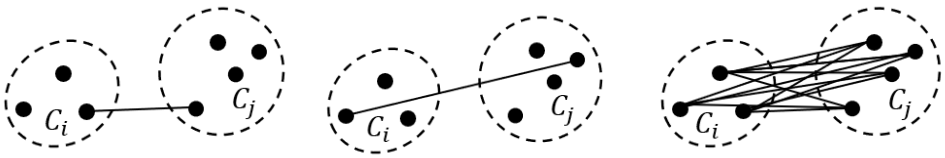
5.3.1 算法方法的距离度量

无论是使用凝聚方法还是使用分裂方法，聚类过程中都需要计算两个聚类簇之间距离。度量簇之间的距离常用的方法有：最小距离、最大距离、平均距离等。如图 5.10 所示，给定聚类簇C_i和C_j，可以通过下面的公式来计算距离。

最小距离： $$d_{\min} = (C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z) \quad (5.3)$$

最大距离： $d_{\max} = (C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$ (5.4)

平均距离： $d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{C_i} \sum_{C_j} dist(x, z)$ (5.5)



(a)最小距离　　(b)最大距离　　(c)平均距离

图 5.10 距离示意图

当算法使用最小距离来衡量簇间距离时，这种情况可称为最近邻聚类算法。此外，如果最近的两个簇之间的距离超过用户给定的阈值时聚类过程就会终止，则称其为**单连接算法**。如果把数据点看做图的结点，图中的边构成簇内结点间的路径，那么两个簇C_i和C_j合并就对应于在C_i和C_j的最近的一对结点之间添加一条边。由于连接簇的边总是从一个簇通向另一个簇，结果图将形成一棵树。因此，使用最小距离度量的凝聚层次聚类算法也被称为最小生成数算法，其中图的生成数是一颗连接所有结点的树，而最小生成数是具有最小边权重和的生成树。

当一个算法使用最大距离来度量簇间距离时，这种情况可称为最远邻聚类算法。如果当最近的两个簇之间的最大距离超过用户给定的阈值时聚类过程便终止，则称其为**全连接算法**。通过把数据点看做图的结点，用边来连接结点，可以把每个簇看成是一个完全子图，即簇中所有结点都有边来连接。两个簇间的距离由两个簇中距离最远的结点间的距离确定。最远邻算法试图在每次迭代中尽可能少地增加簇的直径。如果真实的簇较为紧凑并且大小近似相等，则这种方法将会产生高质量的簇，否则产生的簇可能毫无意义。

以上最小和最大距离度量代表了簇间距离度量的两个极端。它们趋向对离群点或噪声数据过分敏感。使用平均距离是对最小和最大距离之间的一种折中方法，并且可以克服离群点的敏感性问题。平均距离既能处理数值数据又能处理分类数据。

5.3.2 凝聚的与分裂的层次聚类

层次聚类是在不同层次上对数据进行划分，从而形成树状的聚类结构。对给定的数据集进行层次的分解，直到满足某种条件或者达到最大迭代次数，具体又

可分为凝聚的和分类的层次聚类，如图 5.11 所示。

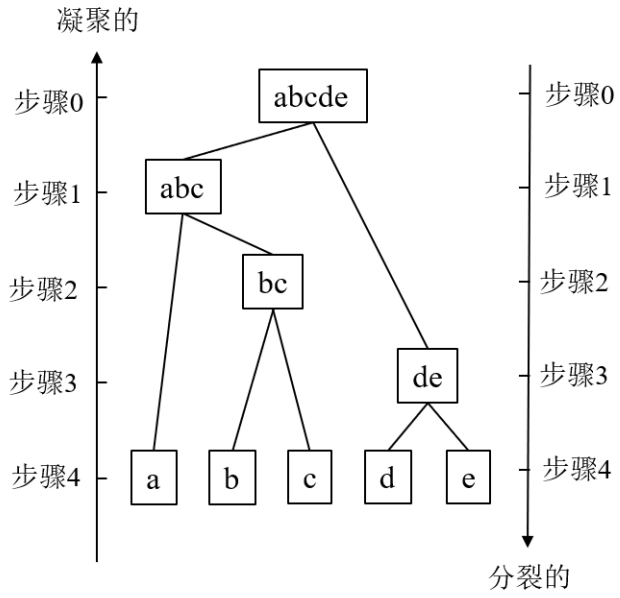


图 5.11 层次聚类算法

层次聚类算法是凝聚的还是分裂的取决于层次分解是以自底向上（合并）还是以自顶向下（分裂）方式形成。

凝聚的层次聚类是一种自底向上的策略，它的典型代表是 AGNES 算法（AGglomerative NESting），它的处理步骤是：①将数据集中的每个对象作为一个簇；②每次找到距离最近的两个簇进行合并；③合并过程反复进行，直至不能再合并或者达到结束条件为止。

AGNES 算法的伪代码如下：

输入：数据集$X = (x_1, x_2, \ldots, x_m)^T$
输出：树状图C

1. 初始化：将每个数据对象当成一个初始簇，分别计算每对簇之间的距离，结果存入距离矩阵$D = (d_{ij})$
2. repeat
3. 寻找距离最近的一对簇C_i和C_j
4. 构造一个新簇$C_k = C_i \cup C_j$，这一过程相当于在树状图中新增一个节点，并将该节点分别连接簇C_i和簇C_j的节点
5. 更新距离矩阵，即计算簇C_k与其他簇（除了C_i和C_j）之间的距离
6. 从矩阵D中删除对应C_i和C_j的行和列，增加对应C_k的行和列
7. until 所有数据对象都在同一个簇中

分裂的层次聚类是一种自顶向下的策略，它的典型代表是 DIANA 算法（Divisive Analysis），它的处理步骤是：①将数据集中的所有集合作为一个簇；②根据某种准则，每次将当前最大的簇分裂成两个子簇；③分裂过程反复进行，直至每个簇只包含一个对象或者达到结束条件为止。

为了确定当前的最大簇，DIANA 算法定义了簇的直径这一概念，它是簇中任意两个数据对象之间距离的最大值。DIANA 算法还定义了平均相异度作为进行簇分裂的依据，它的计算公式如下：

$$d_{avg}(x,C)=\begin{cases}\dfrac{1}{|C|-1}\displaystyle\sum_{y\in C,y\neq x}d(x,y), x\in C\\ \dfrac{1}{|C|}\displaystyle\sum_{y\in C}d(x,y), x\notin C\end{cases} \tag{5.6}$$

其中，$\mathrm{d_{avg}(x,C)}$表示对象x与簇C的平均相异度，即x与C中所有对象（不包括x）之间距离的平均值。$|\mathrm{C}|$表示簇C中对象的个数，$\mathrm{d(x,y)}$是对象x与对象y之间的距离。$\mathrm{d_{avg}(x,C)}$越大，表明x与C中其他对象的差异性越大，进行簇的分裂时，应当优先选择将x分离出去。

DIANA 算法的伪代码如下：

输入：数据集$\mathrm{X}=(\mathrm{x_1},\mathrm{x_2},\ldots,\mathrm{x_m})^{\mathrm{T}}$

输出：树状图C

---

1. 初始化：将所有数据对象作为一个初始簇$\mathrm{C_0}$，并初始化簇的集合$\mathrm{C}=\{\mathrm{C_0}\}$
2. repeat
3. 　将对象集合$\mathrm{U_{new}}$与$\mathrm{U_{old}}$置空
4. 　从C中选择出具有最大直径的簇$\mathrm{C_k}$，找出$\mathrm{C_k}$中平均相异度最大的点p，将p放入$\mathrm{U_{new}}$，将$\mathrm{C_k}$中剩余的对象放入$\mathrm{U_{old}}$
5. 　在$\mathrm{U_{old}}$中找出平均相异度之差满足$\mathrm{d_{avg}(x,U_{old})-d_{avg}(x,U_{new})>0}$的对象，选择其中差值最大的点$\mathrm{q}$加入$\mathrm{U_{new}}$，并将$\mathrm{q}$从$\mathrm{U_{old}}$中删除
6. 　创建两个新簇$\mathrm{C_i}$和$\mathrm{C_j}$，$\mathrm{C_i=U_{new}}$，$\mathrm{C_j=U_{old}}$，将$\mathrm{C_i}$和$\mathrm{C_j}$加入集合C中，将$\mathrm{C_k}$从集合C中删除
7. until 所有数据对象都单独构成一个簇

例 5.3 风速与风向都是影响风力发电的重要指标之一。利用风速、风向余弦量数据聚类是风电功率预测工程中数据预处理部分的常用手段，目的是为了通过聚类寻找相似样本，进而对各类别样本的预测误差情况进行分析，最终通过分析各类别的误差情况构造不同的预测模型有效提高预测精度。表 5.6 给出了某风电场某天连续 8 个时间点的风速值（属性 1）与风向余弦值（属性 2）的采样值，试使用 AGNES 算法将这 8 种气象情况分成两类。

表 5.6 风速风向采样数据集

序号	属性 1	属性 2	序号	属性 1	属性 2
1	4.05	0.25	5	5.65	0.24
2	4.56	0.21	6	5.12	0.37
3	4.87	0.16	7	4.97	0.48
4	4.89	0.31	8	5.94	0.34

在所给的数据集上运行 AGNES 算法，算法的执行过程如表 5.8 所示，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}。（采用最小距离计算）

具体步骤如下：

（1）先根据最小距离计算公式，将两两样本点的距离计算出来，如表 5.7（距离结果保留小数点后两位）。找到距离最小的两个簇合并，故而将 1、7 合并。

表 5.7 两两样本点之间的距离

样本点	1	2	3	4	5	6	7	8
1	0	0.512	0.825	0.842	1.600	1.077	0.948	1.892
2	0.512	0	0.314	0.345	1.090	0.582	0.491	1.386
3	0.825	0.314	0	0.151	0.784	0.326	0.335	1.085
4	0.842	0.345	0.151	0	0.763	0.238	0.188	1.050
5	1.600	1.090	0.784	0.763	0	0.546	0.721	0.307
6	1.077	0.582	0.326	0.238	0.546	0	0.186	0.821
7	0.948	0.491	0.335	0.188	0.721	0.186	0	0.980
8	1.892	1.386	1.085	1.050	0.307	0.821	0.980	0

（2）对上一次合并后的簇进行簇间计算，找出距离最近的两个簇进行合并，合并后 6、7 合并成为一簇。

（3）重复第（2）步的工作，5、8 成为一簇。

（4）重复第（2）步的工作，1、2 成为一簇。

（5）合并{3,4}，{6,7}成为一簇。

（6）合并{3,4,6,7}，{1,2}成为一簇，合并后的簇的数目达到终止条件，计算完毕。

表 5.8 AGNES 算法执行过程

步骤	最近的簇距离	最近的两个簇	合并后的新簇
1	0.151	{3},{4}	{1},{2},{3,4},{5},{6},{7},{8}
2	0.186	{6},{7}	{1},{2},{3,4},{5},{6,7},{8}
3	0.307	{5},{8}	{1},{2},{3,4},{5,8},{6,7}
4	0.512	{1},{2}	{1,2},{3,4},{5,8},{6,7}
5	0.188	{3,4},{6,7}	{1,2},{3,4,6,7} {5,8}
6	0.491	{3,4,6,7},{1,2}	{1,2,3,4,6,7},{5,8}结束

例 5.4 针对表 5.6 的样本事物数据库，实施 DIANA 算法。对所给的数据进行 DIANA 算法，算法的执行过程如表 5.9 所示，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}。

具体步骤如下：

（1）找到具有最大直径的簇，根据表 5.9 对簇中的每个点计算平均相异度。可求出各点的平均相异度如表 5.10 所示。挑出平均相异度最大的点放到分裂的簇中，即点 1，剩余点在原始簇。

表 5.10 各样本点平均相异度

序号	1	2	3	4	5	6	7	8
平均相异度	1.099	0.674	0.546	0.511	0.830	0.539	0.550	1.074

（2）在原始簇里找出到分裂的簇中的最近的点的距离不大于到原始簇中最近的点的距离的点，将该点放入分裂的簇中。根据表 5.7 可知，该点是 2。

（3）重复（2）的工作，在分裂的簇中放入点 3。

（4）重复（2）的工作，在分裂的簇中放入点 4。

（5）没有新的原始簇中的点分配给分裂的簇，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照上边所说的分裂方法继续分裂。

根据两种聚类方法对该数据集的聚类结果显示，两种聚类算法都可以将该数据集根据风速大小

表 5.9 DIANA 算法执行过程

步骤	具有最大直径的簇	分裂的簇	原始簇
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

5.3.3 BIRCH 算法

BIRCH 算法（Balanced Iterative Reducing and Clustering using Hierarchies），由 Tian Zhang 等人于 1996 年提出。该算法是一种适用于大规模数据集的综合性层次聚类算法，首先利用层次聚类方法对数据集进行划分，然后利用其他聚类方法进行优化，以改善聚类质量。

BIRCH 算法引入聚类特征（Clustering Feature, CF）和聚类特征树（CF 树）的概念来描述簇的整体特征。聚类特征是一个三元组，概括了子簇的统计信息。假定一个子簇包含n个d维的数据对象x_t，那么这个子簇的 CF 被定义为

$$CF = (N, LS, SS) \tag{5.7}$$

其中，n是子簇中对象的数目，LS 是n个对象的线性和（即$LS = \sum_{i=1}^{n} x_i$），SS 是对象的平方和（即$SS = \sum_{i=1}^{n} x_i^2$）。从统计学的观点来看，n、LS 和 SS 分别是簇的零阶矩、一阶矩和二阶矩，利用它们可以方便地计算簇的中心、簇的范围（半径或直径），以及簇内或者簇间的距离。

簇的中心、半径、直径和两个簇之间距离的计算公式分别为

簇中心： $x_o = \frac{1}{n}\sum_{i=1}^{n} x_i$ (5.8)

簇半径： $R = [\frac{1}{n}\sum_{i=1}^{n}(x_i - x_o)^2]^{1/2}$ (5.9)

簇直径： $D = [\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2]^{1/2}$ (5.10)

两簇之间的距离： $D' = [\frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}(x_i - x_j)^2}{n_1 n_2}]^{1/2}$ (5.11)

其中，半径R是数据对象到簇中心的平均距离，直径D是簇内两两对象间距离的平均值，它们都反映了簇的紧凑程度。D′是两个簇C_1和C_2的平均距离，反映这两个簇的接近程度。n_1和n_2分别代表簇C_1和C_2中对象的个数。

式(5.8)~式(5.11)进行数学演算后，可以通过 CF 加以表示，具体为

簇中心： $x_o = \frac{LS}{n}$ (5.12)

簇半径： $R = [\frac{SS - (LS)^2/n}{n}]^2$ (5.13)

簇直径： $D = \sqrt{\frac{2SS}{n-1} - \frac{2LS^2}{n(n-1)}}$ (5.14)

两簇之间的距离： $D' = \sqrt{\frac{SS_1}{n_1} + \frac{SS_2}{n_2} - \frac{2LS_1 LS_2}{n_1 n_2}}$ (5.15)

可见，聚类特征足以用来计算簇的距离，并且由于它概括了子簇的信息，所以需要的存储空间远小于簇内所有数据对象占用的空间，因而是一种有效的信息

存储方法。

5.4 基于密度的聚类方法

基于划分的聚类算法和基于层次的聚类算法往往只能发现球状聚类簇，而且多数容易受到离群点的影响。为了更好地发现各种形状的聚类簇，人们提出了基于密度的聚类算法。基于密度的聚类方法与其他方法的根本区别在于：它不是基于距离的，而是基于密度的，这样就能克服基于距离的算法只能发现球状聚类，对发现任意形状的聚类则显得不足的缺点。根据密度定义的不同，有 DBSCAN、OPTICS 和 DENCLUE 等典型算法。

5.4.1 DBSCAN 聚类算法

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的空间聚类算法。该算法的基本思想是每个簇的内部点的密度比簇的外部点的密度要高很多，簇是密度相连的点的最大集合，聚类是在数据空间中不断寻找最大集合的过程。

基于上述原理，DBSCAN 度量密度需要两个参数，即领域半径（ε）和最小包含点数（MinPts），该算法一般任意选择一个未被访问的点开始，找出与其距离在ε半径之内（包括ε的所有近邻点，如果近邻点的数目大于最小包含数，则将该点标记为核心点；如果近邻点数目小于最小包含点，则又分为两种情况，如果该点位于某个核心点的领域半径范围内，就标记该点为边界点，否则，标记为噪声点。如图 5.12 所示。

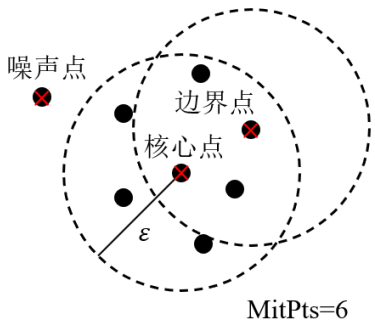


图 5.12 核心点、边界点、噪声点的概念

下面介绍 DBSCAN 中定义的几个基本概念。

（1）ε-领域：对于给定对象p，以p为中心，以ε为半径的区域称为对象p的ε-领域。

（2）核心对象：如果给定对象ε-领域内的样本点数大于 MinPts，则称该对象**为核心对象。**

（3）直接密度可达：对于样本集合D，如果样本点q在p的ε-领域内，并且p为核心对象，则称对象q从对象p出发是直接密度可达的。

（4）密度可达：对于样本集合D，如果存在一个对象序列$p_1, p_2, \ldots, p_n$，$p = p_1, q = p_n$，其中任意对象p_i都是从p_{i-1}直接密度可达的，则称对象q从对象p出发是密度可达的。即多个方向相同的直接密度可达连接在一起称为密度可达。

（5）密度相连：假设样本集合D中存在一个对象o，如果对象o到对象p和到对象q都是密度可达的，那么称p和q密度相连。

需要指出，直接密度可达是具有方向性的，因此密度可达作为直接密度可达的传递闭包是非对称的，而密度相连不具有方向性，是对象关系。DBSCAN 算法的目标是找到密度相连对象的最大集合。并将密度相连的最大对象集合作为簇，不包含任何簇中的对象被称为“噪声点”，图 5.13 展示了上述概念。

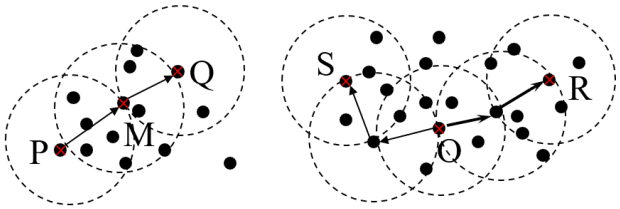


图 5.13 直接密度可达、密度可达、密度相连示意图

在图 5.13 中，Eps 用一个相应圆的半径表示，设 MinPts=3，下面将分析图 5.13 中Q、M、P、S、O、R这 6 个样本点之间的关系。

由于M、P、O这 3 个对象的ε近邻内包含 3 个以上的点，因此它们都是核心对象；M是从P“直接密度可达”的，而Q则是从M“直接密度可达”的；基于以上结果，Q是从P“密度可达”的；但是P从Q无法“密度可达”（非对称性）。同理，S和R是从O“密度可达”的；因此，O、R、S均是“密度相连”的。

这样，DBSCAN 算法就从样本集中找到两个簇，Q、M、P是一个簇，S、O、R是另一个簇。

DBSCAN 聚类算法的步骤分为两步。

（1）寻找核心点形成临时聚类簇

扫描全部样本点，如果某个样本点ε-领域内点的数目≥MinPts，则将其纳入核心点列表，并将其密度可达的点形成对应的临时聚类簇。

（2）合并临时聚类簇得到聚类簇

对于每一个临时聚类簇，检查其中的点是否为核心点，如果是，将该点对应的临时聚类簇和当前临时聚类簇合并，得到新的临时聚类簇。

重复此操作，直到当前临时聚类簇中的每一个点要么不在核心点列表，要么其密度直达的点都已经在该临时聚类簇，该临时聚类簇升级成为聚类簇。

继续对剩余的临时聚类簇进行相同的合并操作，直到全部临时聚类簇被处理。

1）检测样本集中尚未检查过的对象p，如果p未被处理(未归入某个簇或者标记为噪声点)，则检查其邻域，若包含的对象数不小于 MinPts，建立新簇 C,将其领域中的所有点加入候选集N。

2）对候选集N中所有尚未被处理的对象q，检查其邻域，若至少包含 MinPts 个对象，则将这些对象加入N；如果q未归入任何一个簇，则将q加入C。

3）重复步骤 2)，继续检查 N 中未处理的对象，直到当前候选集 N 为空。

4）重复步骤 1）~3)，直到所有对象都归入了某个簇或标记为噪声。

DBSCAN 算法伪代码如下。

输入：数据集 D；领域半径ε；给定点在ε邻域内成为核心对象的最小邻域点数 MinPts；

输出：簇集合C

```
1. repeat
2.    判断输入点是否为核心对象
3.    找出核心对相关的ε邻域中的所有直接密度可达点
4. until 所有输入点都判断完毕
5. repeat
6.    针对所有核心对象的ε邻域中的所有直接密度可达点
7.    找到最大密度相连对象集合，中间涉及到一些密度可达对象的合并
8. until 所有核心对象的ε邻域都遍历完毕
```

例 5.5 如表 5.11 所示，已知某风电场某天连续 12 次采集的风速、功率数据样本集D，试用 DBSCAN 算法对其进行密度聚类，取ε = 2、MinPts=3、n=12。

表 5.11 风速、功率样本集

序号	风速（m/s）	功率(MW)	序号	风速（m/s）	功率(MW)
1	7.85	29.75	7	8.85	33.14
2	7.95	29.5	8	8.97	38.28
3	7.89	29.85	9	9.20	38.43
4	8.54	35.69	10	9.24	37.81
5	8.58	34.51	11	8.86	33.82
6	8.62	32.38	12	8.51	34.2

原始数据散点图如图 5.14 所示，对所给的数据进行 DBSCAN 算法。算法执行过程如表 5.12。

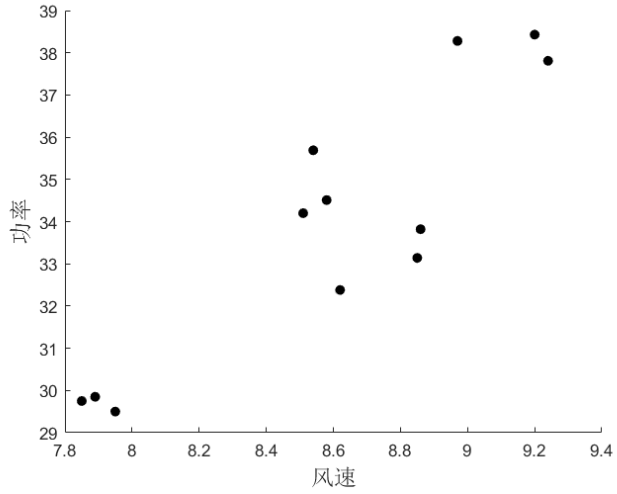


图 5.14 风速-功率散点图

聚出的簇为{1,2,3}，{4,5,6,7,11,12}，{8,9,10}具体步骤如下：

（1）在数据库中选择一点 1，由于在以它为圆心的，以 2 为半径的圆内包含 3 个点，因此它是核心点，寻找从它出发可达的点（直接密度可达的点：2、3），得出新类为{1,2,3}，选择下一个点。

（2）在数据库中选择一点 2，已经在簇 1 中，选择下一个点。

（3）在数据库中选择一点 3，已经在簇 1 中，选择下一个点。

（4）在数据库中选择一点 4，由于在以它为圆心的，以 2 为半径的圆内包含 4 个点，因此它是核心点，寻找从它出发可达的点（直接密度可达的点：5、11、12），得出新类为{4,5,11,12}。

（5）在数据库中选择一点 5，已经在簇 2 中，选择下一个点。

（6）在数据库中选择一点 6，由于在以它为圆心的，以 2 为半径的圆内包含 4 个点，因此它是核心点，寻找从它出发可达的点（直接密度可达的点：7，密度可达：11、12），故而加入簇 2 中。

（7）在数据库中选择一点 7，已经在簇 2 中，选择下一个点。

（8）在数据库中选择一点 8，由于在以它为圆心的，以 2 为半径的圆内包含 3 个点，因此它是核心点，寻找从它出发可达的点（直接密度可达的点：9、10），得出新类为{8,9,10}。

（9）在数据库中选择一点 9，已经在簇 3 中，选择下一个点。

（10）在数据库中选择一点 10，已经在簇 3 中，选择下一个点。

（11）在数据库中选择一点 11，已经在簇 2 中，选择下一个点。

（12）在数据库中选择一点 12，已经在簇 2 中，由于这已经是最后一点（所有点部已处理），计算完毕。

聚类结果散点图如图 5.15 所示，不同形状的散点表示不同的簇。通过聚类结果显示，DBSCAN 算法将风速、功率数据集分成了三种不同发电情况。在实

际的风力发电工况构建工程中，可以利用此方法将风速、功率数据分成低速工况、中速工况、高速工况三种情况，为风力发电工况识别做好数据准备。

表 5.12 DBSCAN 算法执行过程

步骤 1	选择的点	在ε中点的个数	簇
1	1	3	簇C_1:{1,2,3}
2	2	3	已在簇C_1中
3	3	3	已在簇C_1中
4	4	4	簇C_2:{4,5,11,12}
5	5	5	已在簇C_2中
6	6	4	簇C_2:{4,5,6,11,12}
7	7	5	簇C_2:{4,5,6,7,11,12}
8	8	3	簇C_3:{8,9,10}
9	9	3	已在簇C_3中
10	10	3	已在簇C_3中
11	11	6	已在簇C_2中
12	12	6	已在簇C_2中

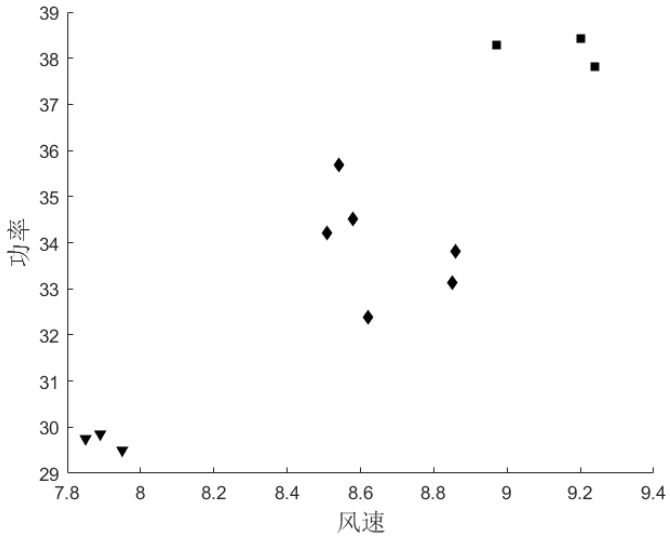


图 5.15 DBSCAN 聚类结果

DBSCAN 算法的目的在于过滤低密度区域，发现稠密样本区域。与传统的基于层次或划分的聚类只能发现凸形簇不同，DBSCAN 算法可以发现任意形状的聚类簇，它具有如下优点。

（1）与 K-means 等算法相比，DBSCAN 算法不需要预先指定簇的数目。

（2）聚类簇的形状没有偏倚。

（3）对噪声点不敏感，并可以在需要时输入过滤噪音的参数。

但由于 DBSCAN 算法直接对整个数据集进行操作，且使用了一个全局性的

表征密度的参数，因此它具有两个较明显的弱点。

（1）当数据量增大时，要求较大的内存支持，I/O 消耗也很大，DBSCAN 的基本时间复杂度是O（N*找出 Eps 领域中的点所需要的时间），N是样本点的个数，因此最坏情况下的时间复杂度是$O(N^2)$。

（2）当遇到密度变化的数据集（即密度分布不均匀）时，聚类间距相差很大时，聚类质量较差。

（3）DBSCAN 算法不能很好反映高维数据。

5.4.2 OPTICS 算法

尽管 DBSCAN 算法能够过滤噪声和离群点数据，并且可以处理任意形状和大小的簇，但是它对参数（ε和 MinPts）的取值较为敏感，而参数的设置通常依靠用户的经验，往往难以确定。而且，真实的高维数据经常具有差异很大的密度分布，因此单一的参数往往不能刻画各个局部的聚类结构。为此，Miaed Ankel 等人提出一种基于簇排序的聚类分析方法——OPTICS（Orderig Points To Idenity the Clustering Structure）。

OPTICS 以 DBSCAN 算法为基础，但是它并不显式地产生一个数据集的聚类，而是基于密度建立起一种簇排序。该序列蕴涵了数据集的内在聚类结构，它包含的信息等同于从一系列参数设置所获得的基于密度的聚类。

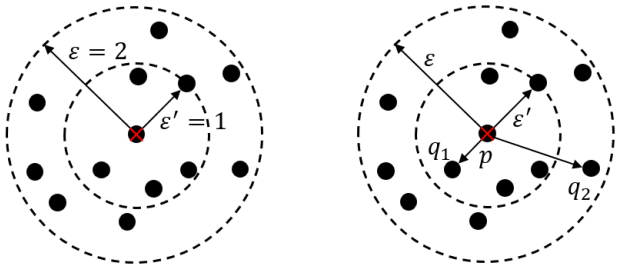
在 DBSCAN 算法中，参数 ε 和 MinPts 是由用户输人的常数。对于给定的 MinPts，当邻域半径ε取值较小时，聚类所得到的簇的密度较高，簇的数量也较多；反之，簇的密度和数量都会减小，但是聚类结果可能会过于粗糙。因此，OPTICS 算法对ε赋予一系列不同的取值，以获得一组密度聚类结果，从而适应不同的密度分布。考虑到在 MinPts 一定时，具有较高密度的簇包含在具有较低密度的簇中，OPTICS 在处理数据集时，按照邻域半径参数ε从小到大的顺序来进行，以便高密度的聚类能够先被执行。基于这个思路，OPTICS 引入了两个参数：核心距离和可达距离。

（1）核心距离：对于一个给定的 MinPts，对象p的核心距离是使得p成为核心对象的最小邻域半径，记作ε'。如果p不可能成为核心对象，则p的核心距离没有定义。

（2）可达距离：一个对象p和另一个对象q的可达距离是p的核心距离与p、q之间的欧氏距离中的较大者。如果p不是核心对象，则p与q之间的可达距离没有定义。

图 5.16 演示了核心距离和可达距离的概念，假设$\varepsilon = 2$，MinPts=5。在 OPTICS 算法中，ε称为生成距离，它确定了领域半径取值的上限。对象p的核心距离为p的第 5 个最近点与p的距离，即$\varepsilon' = 1$，如图 5.16a 所示。在图 5.16b 中，对象q_1与p之间的可达距离等于p的核心距离，即$\varepsilon' = 3$，因为p的核心距离大于p与q_1之间的欧

式距离。对象q_2与p之间的可达距离等于二者的欧式距离，因为欧式距离大于p的核心距离。



（a）　　（b）

图 5.16 核心距离与可达距离的概念

OPTICS 算法执行时，需要对数据集中的对象进行排序，同时计算并存储每个对象的核心距离与可达距离，这些信息足以使用户获得邻域半径小于或等于 ε 范围内的所有聚类结果。数据集的簇排序可以用图形描述，有助于可视化和理解数据集中聚类结构。例如，图 5.17 是一个简单的二维数据集的可达性图，它给出了如何对数据结构化和聚类的一般观察。数据对象连同它们各自的可达距离（纵轴）按簇排序（横轴）绘出。图中箭头所指的三个低谷，表明数据集中存在三个簇。

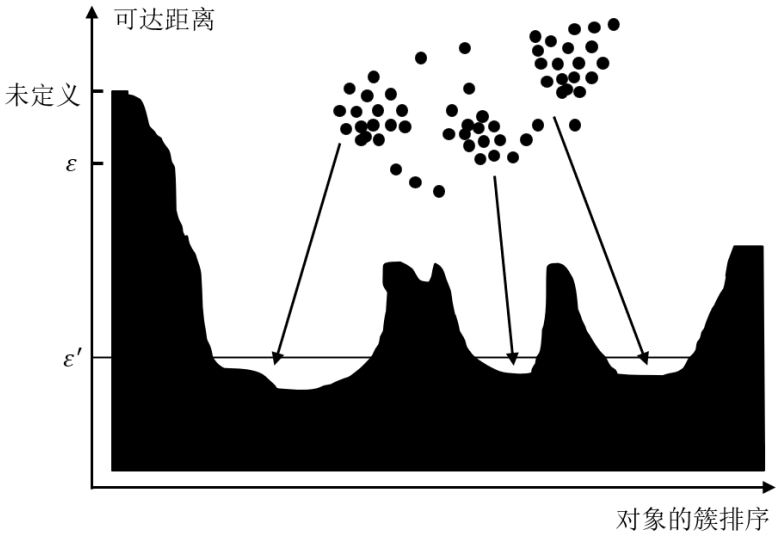


图 5.17 OPTICS 算法的簇排序

5.5 聚类分析性能评估

聚类算法的目标是发现数据集的隐含结构。通常，找到数据集合的最佳隐含结构是一个难题，实用的聚类算法只能得到其近似解，因此需要验证聚类结果的

有效性，即考察聚类结果与数据真实的最佳隐含结构有多大差别。

聚类质量的度量方法大致可以分为外部准则、内部准则和相对准则三大类。

1.外部准则法

外部准则法属于有监督的度量方法，它假设数据集中各对象的真实类别已知，通过比较聚类结果与已知类别的匹配程度来判断聚类质量的优劣。现有外部评价指标分为基于匹配度的指标和基于信息熵的指标。

（1）基于集合匹配度的指标

该类度量方法认为待评测的聚类集C与真实的分类集L之间具有一一对应的关系，并利用信息检索中的精确率和召回率等概念来构造不同的评价指标，如纯度（Purity）、逆纯度（Inverse Purity、F测度（F-measure）等。

对于给定的类别L_j，簇C_i的精确率和召回率分别定义为

$$P(C_i, L_j) = \frac{|C_i \cap L_j|}{|C_i|} \tag{5.16}$$

$$R(C_i, L_j) = \frac{|C_i \cap L_j|}{|L_j|} \tag{5.17}$$

在此基础上，定义聚类的纯度和逆纯度。前者是最大精确率的加权平均，后者是最大召回率的加权平均，二者计算公式如下：

$$Pur = \sum_{C_i \in C} \frac{|C_i|}{N} \max_{L_j \in L} P(C_i, L_j) \tag{5.18}$$

$$InvP = \sum_{L_j \in L} \frac{|L_j|}{N} \max_{C_i \in C} R(C_i, L_j) \tag{5.19}$$

其中，N为样本个数。

F测度将精确率和召回率的概念相结合，从而更好地评估聚类结果。它们的定义如下：

$$F = \sum_{L_j \in L} \frac{|L_j|}{N} \max_{C_i \in C} \frac{2 \times P(C_i, L_j) \times R(C_i, L_j)}{P(C_i, L_j) + R(C_i, L_j)} \tag{5.20}$$

（2）基于信息熵的指标

该类度量方法利用信息论中信息熵和互信息的概念来评价聚类质量。聚类熵被定义为聚类结果中所有簇的熵的平均值，它的计算公式为

$$Ent = -\sum_i \frac{|C_i|}{N} \sum_j P(i, j) \times \log_2 P(i, j) \tag{5.21}$$

其中，求和式$-\sum_j P(i,j) \times \log_2 P(i,j)$表示簇$C_i$的熵，它反映了所有类在该簇内的分布情况；$P(i,j)$表示属于$L_j$类的对象出现在簇$C_i$中的概率。聚类熵的取值范围是[0,1]，0 表示每个簇都是由单个类组成的纯净簇，1 表示每个簇内的类是均匀分布的，显然熵值越小，聚类效果越好。

互信息 MI（Mutual Information）用于度量聚类结果C与数据真实分布L之

间的相似程度，它表示在已知对象真实类别的条件下，可以多大程度地减少对象随机选簇的不确定性。MI 的定义为

$$MI(C,L)=\sum_{i=1}^{|C|}\sum_{j=1}^{|L|}P(C_i,L_j)\log_2\frac{P(C_i,L_j)}{P(C_i)P(L_j)} \tag{5.22}$$

其中，概率$P(C_i)$、$P(L_i)$和$P(C_i,L_i)$分别为

$$P(C_i)=\frac{|C_i|}{N} \tag{5.23}$$

$$P(L_j)=\frac{|L_j|}{N} \tag{5.24}$$

$$P(C_i,L_j)=\frac{|C_i\cap L_j|}{N} \tag{5.25}$$

在互信息的基础上，还定义有标准化互信息 NMI（Normalized Mutual Information）和调整互信息 AMI（Adjusted Mutual Information），它们的计算公式分别为

$$NMI(C,L)=\frac{MI(C,L)}{\sqrt{H(C)H(L)}} \tag{5.26}$$

$$AMI(C,L)=\frac{MI-E(MI)}{\max(H(C),H(L))-E(MI)} \tag{5.27}$$

其中，E(MI)是互信息 MI 的数学期望，H(C)和H(L)分别代表C和L这两种分布的熵。

MI 和 NMI 的取值范围都是[0,1]，AMI 的取值范围是[-1,1]。它们的值越接近 1，表明聚类效果越好。

2.内部准则法

内部准则法是非监督的度量方法，它没有可参考的外部信息，只能依赖数据集自身的特征和量值对聚类结果进行评价。在这种情况下，需要从聚类的内在需求出发，考察簇的紧密征和量值对聚类结果进行评价。在这种情况下，需要从聚类的内在需求出发，考察簇的紧密度、分离度以及簇表示的复杂度。紧密度(Cohesion)反映簇内成员的凝聚程度；分离度(Separation)表示簇与簇之间的相异程度，理想的聚类效果应该具有较高的簇内紧密度和较大的簇间分离度。大多数评价聚类质量的方法都是基于这两个原则，如轮廓系数（Silhouette Coefficient）和 CH 指标（Calinski-Harabasz）。

（1）轮廓系数

轮廓系数的基本思想是将数据集中的任一对象与本簇中其他对象的相似性以及该对象的相似性以及该对象与其他簇中对象的相似性进行量化，并将量化后的两种相似性以某种形式组合，以获得聚类优劣的评价标准。

对于数据集D中的任意对象x_i，假设聚类算法将x_i划分到簇C，则该对象的轮廓系数s_i定义为

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{5.28}$$

其中，a_i是x_i与本簇中其他对象之间的平均距离，它反映了x_i所属簇的紧密程度，该值越小，簇的紧密程度越好。b_i是x_i与最近簇的对象之间的平均距离，它表示x_i与其他簇的分离程度，该值越大，分离度越高。

轮廓系数s_i的值在[-1,1]之间，可用于评价对象x_i是否适合所属的簇。若s_i接近 1，说明包含x_i的簇是紧凑的，并且x_i远离其他簇，因此对x_i所采取的分配方式是合理的。若s_i为负值，则意味着x_i距离其他簇的对象比距离本簇的对象更近，当前对x_i的分簇是不合理的，将其分配到最近邻的簇会获得更好的效果。

为了评价聚类方案的有效性，需要对D中所有对象的轮廓系数求平均数，得到平均轮廓系数s，公式为

$$s = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{5.29}$$

其中，n是D中对象的个数。同样地，s的值越大，表明聚类质量越好。

（2）CH 指标

CH 指标通过类内协方差矩阵描述紧密度，类间协方差描述分离度，它的定义为

$$CH(k) = \frac{trB(k/(k-1))}{trW(k)/(n-k)} \tag{5.30}$$

其中，n为数据中对象的个数，k为簇的个数，trB(k)和trW(k)分别是类间协方差矩阵B(k)和类内协方差矩阵W(k)的迹。

$$trW(k) = \frac{1}{2}\sum_{i=1}^{k}(|C_i|-1)\bar{d}_i^{\,2} \tag{5.31}$$

$$trB(k) = \frac{1}{2}[(k-1)\bar{d}_i^{\,2} + (n-k)A_k] \tag{5.32}$$

其中，$|C_i|$表示簇C_i中对象的个数，$\bar{d}_i^2$是C_i中对象的平均距离，$\bar{d}^2$是数据集中所有对象间的平均距离，且有

$$A_k = \frac{1}{n-k}\sum_{i=1}^{k}(|C_i|-1)(\bar{d}^{\,2} - \bar{d}_i^{\,2}) \tag{5.33}$$

CH 指标值越大表示聚类结果的性能越好。

3.相对准则法

相对准则法用于比较不同聚类的结果，通常是针对同一个聚类算法的不同参数设置进行算法测试(如不同的分簇个数)，最终选择最优的参数设置和聚类模式。相对准则法在进行聚类比较时，直接使用外部准则法或者内部准则法定义的评价指标，因而它实际上并不是一种单独的聚类质量度量方法，而是前两类度量方法的一种具体应用。例如，可以利用 CH 指标确定 K-means 算法的最佳分簇数目 K，具体过程是：首先给定 K 的取值范围$[k_{min}, k_{max}]$，然后使用同一聚类算法、不

同的 K 值对数据集进行聚类，得到一系列聚类结果，再分别计算每个聚类结果的有效性指标，最后比较各个指标值，对应最佳指标值的 K 就是最佳分簇数目。

5.6 实例

在高炉炼钢行业中，高炉煤气利用率的大小被确立为用来反映高炉冶炼效果的指标之一。高炉煤气利用率能够实时反映煤气利用程度、煤气流分布状态、能源消耗程度以及铁水的质量和产量，是反映高炉整体状态的重要参数。在炼钢过程中，高炉煤气利用率的大小又受各种操作参数的影响，如冷风流量、冷风压力、矿焦比 3 等的影响。在进行高炉煤气利用率预测时，可以利用聚类方法对各操作参数做分类分析，为后续高炉煤气利用率预测建模做准备。附表 1 为 50 组高炉炼钢过程中采集到的冷风流量、冷风压力、矿焦比 3 的数据集。使用 K-means 和 DBSCAN 两种聚类方法对其进行聚类，并进行聚类性能评估。

由于不同属性的量纲不同，需先对数据进行归一化，归一化过程参考本书 3.4 部分。归一化结果如附表 2 所示。

通过散点图将数据集可视化，如图 5.18 所示。从样本集的散点分布来看，样本集偏向于球形分布并存在部分较为“稀疏”的样本点。在处理该种样本集聚类时，DBSCAN 算法可能无法避免噪声点的存在，相比之下，K-means 算法可能会更加合适。下面将对该数据集进行聚类处理，证明：对于该数据集，K-means 算法的聚类效果比 DBSCAN 算法的聚类效果要好。

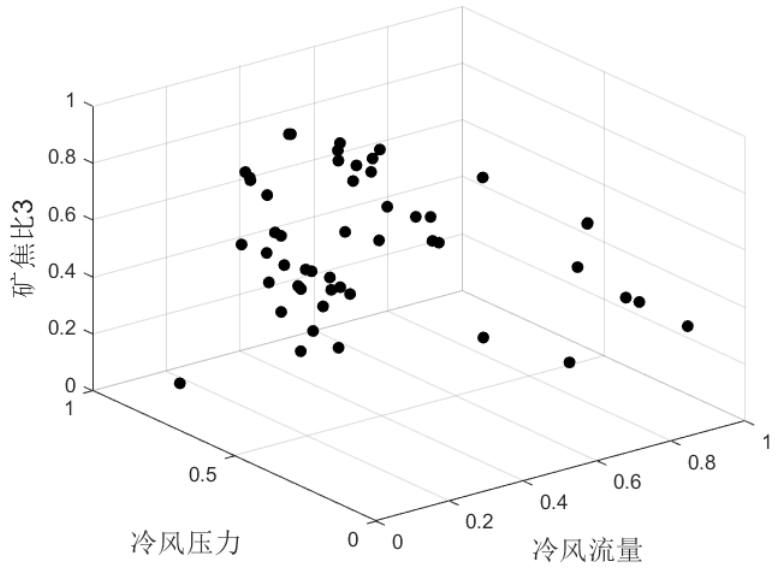


图 5.18 数据集散点图

1.K-means 聚类方法聚类

利用肘部法确定最佳聚类数。如图 5.19 所示，最佳聚类数为 4。根据 K-means 聚类算法进行聚类，初始聚类中心选择点 1、点 2、点 3，K 取 4。聚类结果如图

5.20 所示，各类别信息如表 5.13 所示。

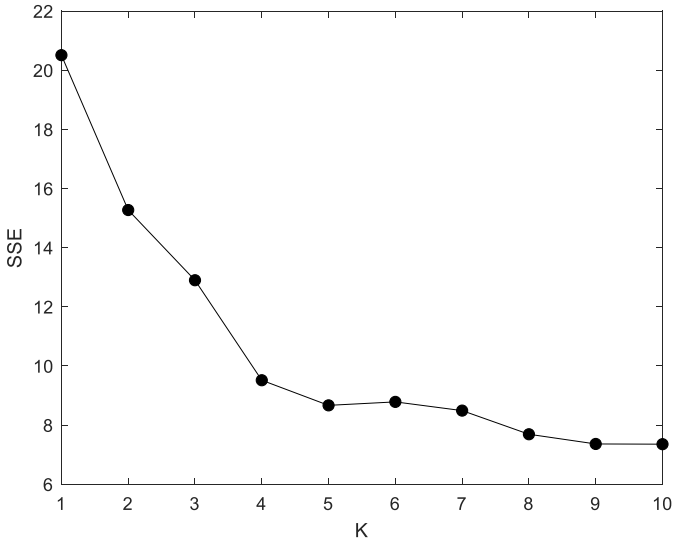


图 5.19 肘部法取 K 值折线图

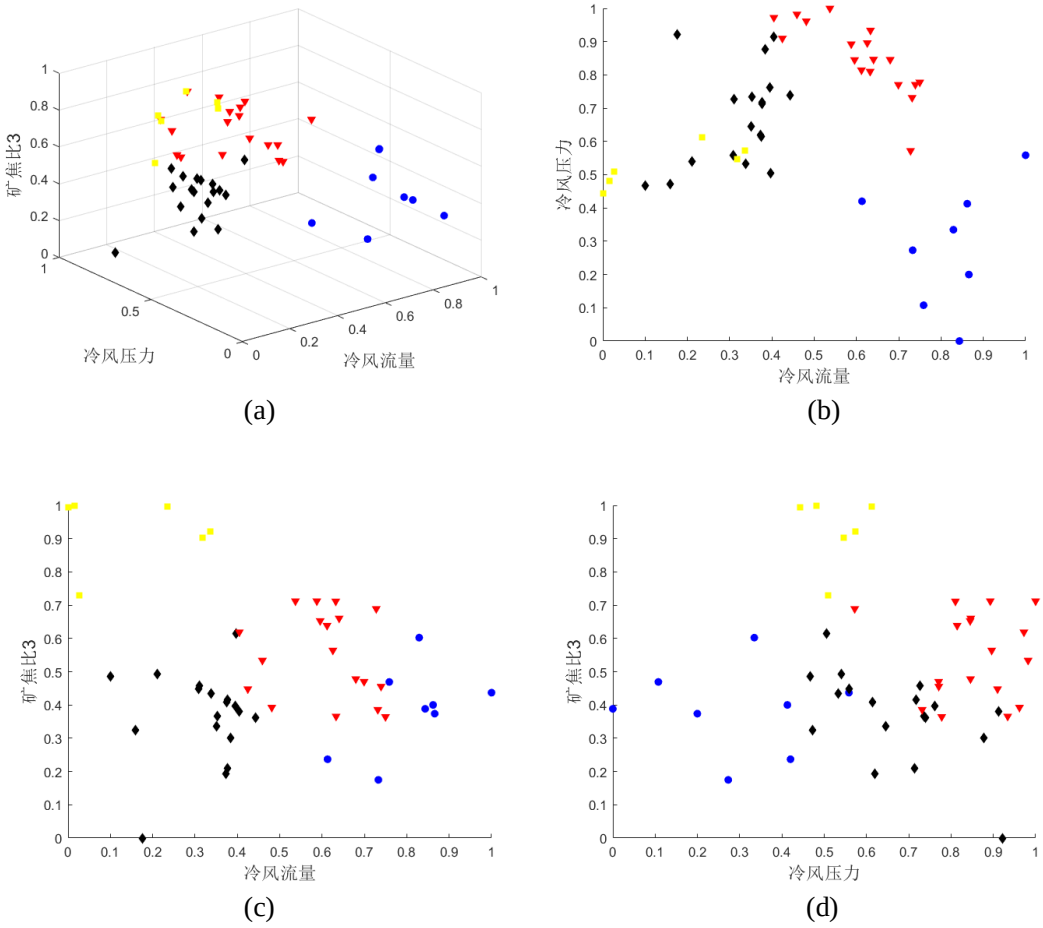


图 5.20 K-means 聚类结果散点图

表 5.13 K-means 聚类分簇结果

簇	聚类中心	聚类结果
1	(0.8131,0.2883,0.3854)	{21,22,23,24,25,31,32,33}
2	(0.3240,0.6702,0.3683)	{1,2,3,10,11,12,13,14,15,16,17,18,19,20,28,48,49,50}
3	(0.6088,0.8520,0.5476)	{26,27,29,30,34,35,36,37,38,39,40,41,42,43,44,45,46,47}
4	(0.1553,0.5278,0.9242)	{4,5,6,7,8,9}

聚类性能评估：利用轮廓系数对本次聚类结果进行评估，根据式(5.28)计算得 s=0.7729。

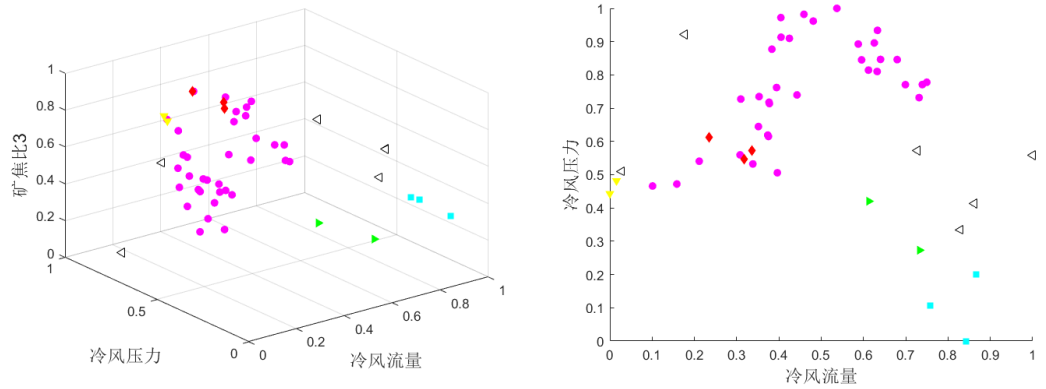
2.DBSCAN 聚类方法聚类

设置不同参数，利用轮廓系数评估聚类质量，评估结果如表 5.14。

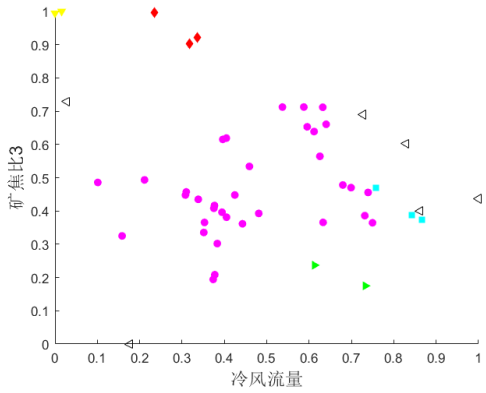
表 5.14 轮廓系数评估结果

ε \ MinPts	1	2	3	4	5
0.1	0.4634	0.4758	0.2729	0.2815	0.0081
0.2	0.5432	0.6539	0.5430	0.4270	0.4270
0.3	0.3857	0.3857	0.3857	0.3857	0.5523
0.4	-1	-1	-1	-1	-1

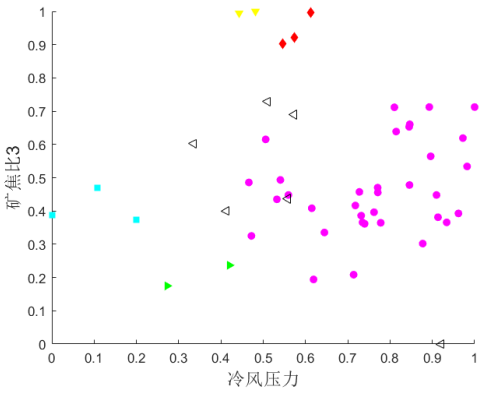
根据表 5.14 所示的轮廓系数评估结果，当ε = 0.2、MinPts=2 时，轮廓系数 s=0.6539，聚类结果较好。聚类结果如图 5.21 所示，各簇信息如表 5.15 所示，空心散点代表噪声点。



(a) (b)



(c)



(d)

图 5.21 DBSCAN 聚类结果散点图

表 5.15　DBSCAN 聚类分簇结果

簇	聚类结果
1	{1,2,3,10,11,12,13,14,15,16,17,19,20,21,26,27,28,29,10,35,36,37,38,39,40,41,42, 43,44,45,46,47,48,49,50}
2	{4,5,6}
3	{7,8}
4	{22,23,32}
5	{24,25}
噪声点	{9,18,31,33,34}

通过以上两种聚类算法对该数据集的聚类结果可知：使用 DBSCAN 聚类算法后轮廓系数较小，且结果存在明显噪声点。因此，对于该数据集，使用 K-means 聚类算法效果比 DBSCAN 聚类算法效果要好些。由此证明，对于密度分布不均、维数较高的数据集，使用 K-means 聚类相对更合适。

在高炉炼钢过程中，由于炉况波动导致参数变化过大，直接构建预测模型往往预测精度不高。而通过工况的分类构建不同的预测模型往往能获得更好的预测效果。根据本次 K-means 聚类结果可知，冷风流量、冷风压力、矿焦比 3 参数数据聚成 4 类，即将炼钢过程分成 4 种工况。在高炉煤气利用率预测工程中，我们需要根据不同工况构建不同的预测模型，从而分析本次工况分类是否有效。

5.7 本章小结

本章介绍了关于聚类的基本概念以及按照不同方面进行的类别划分，对在不同的聚类类别中具有代表性的三种聚类算法（划分聚类算法、层次聚类算法以及基于密度的聚类算法）进行了详细介绍，并通过实例深入地展示了其聚类计算和建立聚类结果的过程。

基本概念：介绍了聚类分析的定义，指出聚类分析可以应用在数据挖掘的各

个过程，并详细阐述了聚类算法的性能要求，包括伸缩性、处理不同字段类型的能力等。

划分聚类方法：介绍了划分聚类算法的定义，并详细介绍 K-means、K-medoids 和 K-means++算法的聚类思想和聚类过程。

层次聚类方法：介绍了层次聚类算法中常用的距离度量，通过层次分解的不同分别介绍了凝聚的与分裂的层次聚类算法的聚类思想和聚类过程，并扩展介绍了 BIRCH 算法的基本聚类思想。

基于密度的聚类方法：简单介绍了基于密度的聚类算法与前两种聚类方法的区别，接着详细介绍了 DBSCAN 算法的聚类思想、所涉及参数的基本概念和聚类过程，最后扩展介绍了 OPTICS 算法的基本思想。

聚类分析性能评估：通过外部准则法、内部准则法和相对准则法三个方面介绍了常用的聚类结果性能评估方法和指标。

练习

1、简述聚类分析的基本思想。

2、简述 K-means 算法的工作流程以及优缺点。

3、简述 K-medoids 算法的优缺点，并说明其与 K-means 算法的相同点与不同点。

4、分别取 K=2 和 3，随机选择初始聚类中心，利用 K-means 聚类算法对以下的点聚类：（2,1），（1,2），（2,2），（3,2），（2,3），（3,3），（2,4），（3,5），（4,4），（5,3），求聚类结果的 SSE 值并讨论 K 值以及初始聚类中心对聚类结果的影响。

5、假设数据挖掘的任务是将如下的 8 个点聚类为三个类：（2,10），（2,5），（8,4），（5,8），（7,5），（6,4），（1,2），（4,9）。距离函数选用欧式距离。用 K-medoids 算法得到：

（1）初始聚类中心；

（2）最后的三个簇。

6、试用 DIANA 算法对下表进行聚类，要求采用欧式距离，终止条件为两个簇。

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

7、对题 5 的数据样本集实施 AGNES 算法，要求采用欧式距离。

8、根据下表所示的数据集距离矩阵，要求分别采用最近距离法和最远距离法进行凝聚聚类。

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.57	0.90	0.32	0.65	0.46
x_2	0.57	0	0.34	0.04	0.15	0.56
x_3	0.90	0.34	0	0.11	0.03	0.52
x_4	0.32	0.04	0.11	0	0.19	0.26
x_5	0.65	0.15	0.03	0.19	0	0.24
x_6	0.46	0.56	0.52	0.26	0.24	0

9、检验某类产品的重量，抽了六个样品，每个样品只测了一个指标，分别是 1，2，3，6，9，11。试用最短距离法进行聚类分析。

10、尝试利用 DBSCAN 算法对如下 13 个样本点进行聚类，取ε=3，MinPts=3。

	1	2	3	4	5	6	7	8	9	10	11	12	13
X	1	2	2	4	5	6	6	7	9	1	3	5	3
Y	2	1	4	3	8	7	9	9	5	12	12	12	3

11、简述评价聚类结果质量的主要方法。

12、分别利用 K-means 算法（取 K=3）和 DBSCAN 算法（取ε=1，MinPts=4）对如下某工业数据集进行聚类，并利用轮廓系数评估聚类效果，比较得出针对该数据集合适的聚类方法。

	1	2	3	4	5	6	7	8	9	10	11	12
特征 1	4.7	4.9	0.9	5.2	5.0	4.6	5.5	4.7	4.3	3.7	4.6	4.1
特征 2	0.43	0.43	0.48	0.73	0.77	0.28	0.40	0.42	0.43	0.40	0.44	0.46

参考文献

[1] 唐四薪, 赵辉煌, 唐琼. 大数据分析使用教程：基于 Python 实现[M]. 北京：机械工业出版社，2021

[2] 陈燕. 数据挖掘技术与应用[M]. 北京：清华大学出版社，2011

[3] 王振武. 大数据挖掘与应用[M]. 北京：清华大学出版社，2017

[4] 赵志升, 梁俊花, 李静, 刘洋. 大数据挖掘[M]. 北京：清华大学出版社，2019

[5] 施苑英. 大数据技术及应用[M]. 北京：机械工业出版社，2021
[6] Hartigan J A, Wong M A. Algorithm AS 136: A k-means clustering algorithm[J]. Journal of the royal statistical society. series c (applied statistics), 1979, 28(1): 100-108.
[7] Chan T F, Xu J, Zikatanov L. An agglomeration multigrid method for unstructured grids[J]. Contemporary Mathematics, 1998, 218: 67-81.
[8] Hill M O, Bunce R G H, Shaw M W. Indicator species analysis, a divisive polythetic method of classification, and its application to a survey of native pinewoods in Scotland[J]. The Journal of Ecology, 1975: 597-613.
[9] Birant D, Kut A. ST-DBSCAN: An algorithm for clustering spatial–temporal data[J]. Data & knowledge engineering, 2007, 60(1): 208-221.

附表 1 操作参数数据集

序号	冷风流量	冷风压力	矿焦比 3
1	5472.86	420.80	6.664021
2	5466.69	421.42	6.673884
3	5484.70	420.20	6.801036
4	5468.69	421.13	7.020458
5	5472.46	421.73	7.033815
6	5451.58	422.63	7.090986
7	5406.93	419.64	7.093571
8	5403.72	418.77	7.089507
9	5409.36	420.30	6.887106
10	5424.39	419.29	6.702608
11	5436.11	419.43	6.580399
12	5446.94	420.99	6.708367
13	5475.54	423.36	6.588259
14	5480.85	424.94	6.491762
15	5484.33	426.03	6.634532
16	5480.70	425.03	6.649927
17	5475.88	425.41	6.611236
18	5439.82	429.67	6.333272
19	5482.06	428.66	6.562865
20	5494.21	425.53	6.608088
21	5579.91	418.08	6.637492
22	5576.10	408.68	6.62861
23	5558.79	411.13	6.690026
24	5553.52	414.89	6.466219
25	5529.01	418.25	6.513356
26	5415.49	418.50	6.925911
27	5393.56	419.11	6.926398

28	5394.93	418.57	7.058929
29	5394.63	418.74	6.793927
30	5407.50	418.06	6.677193
31	5421.07	419.04	6.799025
32	5439.69	419.35	6.913349
33	5449.55	421.06	6.746479
34	5461.98	421.48	6.802104
35	5461.16	421.39	6.68525
36	5465.13	421.95	6.990408
37	5475.02	421.95	6.787754
38	5468.35	424.23	6.785635
39	5467.73	426.21	6.93793
40	5469.26	425.75	6.908029
41	5485.10	425.96	6.849778
42	5516.94	424.76	6.923503
43	5556.61	410.34	6.881509
44	5540.06	410.75	6.810779
45	5520.46	413.64	6.801903
46	5494.84	417.40	6.7054
47	5493.57	418.16	6.665369
48	5466.69	422.25	6.478297
49	5460.37	424.30	6.522306
50	5481.91	421.75	6.232704

附表 2 操作参数数据集归一化

序号	冷风流量	冷风压力	矿焦比 3
1	0.426	0.578	0.501
2	0.392	0.607	0.512
3	0.489	0.549	0.660
4	0.403	0.593	0.915
5	0.423	0.622	0.931
6	0.311	0.665	0.997
7	0.072	0.522	1.000
8	0.055	0.481	0.995
9	0.085	0.553	0.760
10	0.165	0.506	0.546
11	0.228	0.512	0.404
12	0.286	0.587	0.553
13	0.440	0.700	0.413
14	0.468	0.775	0.301
15	0.487	0.827	0.467
16	0.468	0.779	0.485
17	0.442	0.797	0.440

18	0.248	1.000	0.117
19	0.475	0.952	0.384
20	0.540	0.803	0.436
21	1.000	0.448	0.470
22	0.980	0.000	0.460
23	0.887	0.117	0.531
24	0.858	0.296	0.271
25	0.727	0.456	0.326
26	0.118	0.468	0.805
27	0.000	0.497	0.806
28	0.007	0.471	0.960
29	0.006	0.479	0.652
30	0.075	0.447	0.516
31	0.148	0.494	0.658
32	0.248	0.508	0.791
33	0.300	0.590	0.597
34	0.367	0.610	0.661
35	0.363	0.605	0.526
36	0.384	0.632	0.880
37	0.437	0.632	0.645
38	0.401	0.741	0.642
39	0.398	0.835	0.819
40	0.406	0.813	0.784
41	0.491	0.823	0.717
42	0.662	0.766	0.802
43	0.875	0.079	0.754
44	0.786	0.099	0.672
45	0.681	0.237	0.661
46	0.544	0.416	0.549
47	0.537	0.452	0.503
48	0.392	0.647	0.285
49	0.359	0.744	0.336
50	0.474	0.623	0.000