第 3 讲 聚类分析

聚类分析，相当于“物以类聚”，事先并不知道类别的个数与结构，根据对象之间的相似性或相异性（距离度量）对研究对象进行分类.

聚类分析分为：

样本聚类（Q 型）：其统计指标是类与类之间距离，把每一个样本看成空间中的一个点，用某种原则规定类与类之间的距离，将距离近的点聚合成一类，距离远的点聚合成另一类.

变量聚类（R 型）：其统计指标是变量间的相似系数，将比较相似的变量归为一类，而把不怎么相似的变量归为另一类，用它可以把变量的亲疏关系直观地表示出来.

聚类分析内容非常丰富，有层次聚类法、有序样品聚类法、动态聚类法、模糊 聚类法、图论聚类法等. 本讲主要介绍常用的层次聚类、K 均值聚类.

3.1 数据变换

设有 n 个样品, 每个样品测得 p 项指标 (变量), 原始数据阵为

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{bmatrix}.$$

其中 $a_{ij}\ (i=1,\cdots,n;\ j=1,\cdots,p)$ 为第 i 个样品 ω_i 的第 j 个指标的观测数据.

由于样本数据矩阵由多个指标组成, 不同指标一般有不同的量纲, 为消除量纲的影响, 通常需要进行数据变换处理. 常用的数据变换方法有以下两种.

1) 规格化变换

规格化变换是从数据矩阵的每一个变量值中找出其最大值和最小值, 这两者之差称为极差, 然后从每个变量值的原始数据中减去该变量值的最小值, 再除以极差, 就得到规格化数据, 即有

$$b_{ij}=\frac{a_{ij}-\min\limits_{1\leqslant i\leqslant n}(a_{ij})}{\max\limits_{1\leqslant i\leqslant n}(a_{ij})-\min\limits_{1\leqslant i\leqslant n}(a_{ij})}\quad(i=1,\cdots,n;j=1,\cdots,p).$$

2) 标准化变换

首先对每个变量进行中心化变换, 然后用该变量的标准差进行标准化, 即有

$$b_{ij} = \frac{a_{ij} - \mu_j}{s_j} \quad (i = 1, \cdots, n; j = 1, \cdots, p),$$

其中 $\mu_j = \dfrac{\sum_{i=1}^{n} a_{ij}}{n}$, $s_j = \sqrt{\dfrac{1}{n-1}\sum_{i=1}^{n}(a_{ij} - \mu_j)^2}$.

记变换处理后的数据矩阵为

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}.$$

3.2 样品间亲疏程度的测度计算

研究样品的亲疏程度或相似程度的数量指标通常有两种: 一种是相似系数，性质越接近的样品，其取值越接近于 1 或 -1，而彼此无关的样品相似系数则接近于 0，相似的归为一类，不相似的归为不同类. 另一种是距离，它将每个样品看成 p 维 空间的一个点，n 个样品组成 p 维空间的 n 个点. 用各点之间的距离来衡量各样品 之间的相似程度. 距离近的点归为一类，距离远的点属于不同的类.

3.2.1 常用距离的计算

常见的计算样本间的距离有以下几种：

（1）Minkowski（闵可夫斯基）距离

$$d_{ij} = \left(\sum_{k=1}^{p} \left| b_{ik} - b_{jk} \right|^{q} \right)^{\frac{1}{q}}, 1 \le i, j \le n .$$

当 $q = 1$ 时为**绝对值距离**，也称为 **Manhattan（曼哈顿）距离**；当 $q = 2$ 时为**欧氏距离**，当 $q \to \infty$ 时，为 **Chebyshev（切比雪夫）距离**.

当 $q = 1$ 时,

$$d_{ij}(1) = \sum_{k=1}^{p} |b_{ik} - b_{jk}|, \quad \text{即绝对值距离.}$$

当 $q=2$ 时,

$$d_{ij}(2)=\left(\sum_{k=1}^{p}(b_{ik}-b_{jk})^2\right)^{1/2},\quad \text{即欧氏距离.}$$

当 $q=\infty$ 时,

$$d_{ij}(\infty)=\max_{1\leqslant k\leqslant p}|b_{ik}-b_{jk}|,\quad \text{即切比雪夫距离.}$$

在 Minkowski 距离中，最常用的是欧氏距离，它的主要优点是当坐标轴进行正交旋转时，欧氏距离是保持不变的. 因此，如果对原坐标系进行平移和旋转变换，则变换后样本点间的距离和变换前完全相同.

值得注意的是在采用 Minkowski 距离时，一定要采用相同量纲的变量. 如果变量的量纲不同，测量值变异范围相差悬殊时，建议首先进行数据的标准化处理，然后再计算距离. 在采用 Minkowski 距离时，还应尽可能地避免变量的多重相关性. 多重相关性所造成的信息重叠，会片面强调某些变量的重要性.

由于 Minkowski 距离的这些缺点，一种改进的距离就是马氏距离，定义如下

（2）Mahalanobis（马哈拉诺比斯）距离：

$$d_{ij}=\sqrt{\left(X_i-X_j\right)^T S^{-1}\left(X_i-X_j\right)}.$$

其中 X_i, X_j 为来自 p 维总体 Z 的样本观测值，S 为 Z 的协方差矩阵，实际中 S 往往是不知道的，常常需要用样本协方差来估计. 马氏距离对一切线性变换是不变的，故不受量纲的影响.

（3）推广的 Lance（兰氏）距离

$$d_{ij}=\sum_{k=1}^{q}\frac{\left|b_{ik}-b_{jk}\right|}{\left|b_{ik}+b_{jk}\right|}.$$

（4）定性变量样本间的距离

例如，如下两个样本（表 2），按定性变量的各水平值，是=1，否=0.

表 2 定性变量样本示例

样本	性别		外语				专业			职业	
	男	女	英	日	德	俄	统计	会计	金融	教师	工程师
X_1	1	0	1	0	0	0	0	0	1	0	1
X_2	0	1	1	0	0	0	1	0	0	1	0

记 m_0 为两样本 0-0 配对的数目，m_1 为 1-1 配对的数目，m_2 为不配对的数目，定义两样本之间的距离：

$$d_{ij}=\frac{m_2}{m_1+m_2}.$$

本例中 $m_0=4, m_1=1, m_2=6$，故距离 $d_{12}=\frac{6}{7}$.

注：距离选择的基本原则是要考虑所选择的距离公式在实际应用中有明确的意义，如欧氏距离就有非常明确的空间距离概念，马氏距离有消除量纲影响的作用；应根据研究对象的特点不同做出具体分析，或测试几种距离从中选择聚类结果最好的.

3.2.2 相似系数的计算

研究样品之间的关系, 除了用距离表示外, 还有相似系数. 相似系数是描述样品之间相似程度的一个统计量, 常用的相似系数有以下几种.

1) 夹角余弦

将任何两个样品 ω_i 与 ω_j 看成 p 维空间的两个向量, 这两个向量的夹角余弦用 $\cos\theta_{ij}$ 表示, 则

$$\cos\theta_{ij}=\frac{\sum_{k=1}^{p}b_{ik}b_{jk}}{\sqrt{\sum_{k=1}^{p}b_{ik}^2}\cdot\sqrt{\sum_{k=1}^{p}b_{jk}^2}},\quad i,j=1,2,\cdots,n.$$

当 $\cos\theta_{ij}=1$ 时, 说明两个样品 ω_i 与 ω_j 完全相似; $\cos\theta_{ij}$ 接近 1 时, 说明 ω_i 与 ω_j 相似密切; $\cos\theta_{ij}=0$ 时, 说明 ω_i 与 ω_j 完全不一样; $\cos\theta_{ij}$ 接近 0 时, 说明 ω_i 与 ω_j 差别大. 把所有两两样品的相似系数都计算出来, 可排成相似系数矩阵

$$\Theta=\begin{bmatrix}\cos\theta_{11} & \cos\theta_{12} & \cdots & \cos\theta_{1n}\\ \cos\theta_{21} & \cos\theta_{22} & \cdots & \cos\theta_{2n}\\ \vdots & \vdots & & \vdots\\ \cos\theta_{n1} & \cos\theta_{n2} & \cdots & \cos\theta_{nn}\end{bmatrix},$$

其中 $\cos\theta_{11}=\cdots=\cos\theta_{nn}=1$. 根据 Θ 可对 n 个样品进行分类, 把比较相似的样品归为一类, 不怎么相似的样品归为不同的类.

2) 皮尔逊相关系数

第 i 个样品与第 j 个样品之间的相关系数定义为

$$r_{ij}=\frac{\sum_{k=1}^{p}(b_{ik}-\bar{\mu}_i)(b_{jk}-\bar{\mu}_j)}{\sqrt{\sum_{k=1}^{p}(b_{ik}-\bar{\mu}_i)^2\cdot\sum_{k=1}^{p}(b_{jk}-\bar{\mu}_j)^2}},\quad i,j=1,2,\cdots,n,$$

其中, $\bar{\mu}_i=\dfrac{\sum_{k=1}^{p}b_{ik}}{p}$.

实际上, r_{ij} 就是两个向量 $\boldsymbol{B}_i-\bar{\boldsymbol{B}}_i$ 与 $\boldsymbol{B}_j-\bar{\boldsymbol{B}}_j$ 的夹角余弦, 其中 $\bar{\boldsymbol{B}}_i=\bar{\mu}_i[1,2,\cdots,1]$. 若将原始数据标准化, 满足 $\bar{\boldsymbol{B}}_i=\bar{\boldsymbol{B}}_j=0$, 这时 $r_{ij}=\cos\theta_{ij}$.

$$\boldsymbol{R}=(r_{ij})_{n\times n}=\begin{bmatrix}r_{11}&r_{12}&\cdots&r_{1n}\\r_{21}&r_{22}&\cdots&r_{2n}\\\vdots&\vdots&&\vdots\\r_{n1}&r_{n2}&\cdots&r_{nn}\end{bmatrix},$$

其中 $r_{11}=\cdots=r_{nn}=1$, 可根据 $\boldsymbol{R}$ 对 n 个样品进行分类.

3.3 scipy.cluster.hierarchy 模块的层次聚类

scipy.cluster.hierarchy 模块的层次聚类函数介绍如下.

1. distance.pdist

B = pdist(A, `metric='euclidean'`) 用 `metric` 指定的方法计算 $n\times p$ 矩阵 A (看作 n 个 p 维行向量, 每行是一个对象的数据) 中两两对象间的距离, `metric` 可取表 11.9 中的特征字符串. 输出 B 是包含距离信息的长度为 $(n-1)\cdot n/2$ 的向量. 可用 distance.squareform 函数将此向量转换为方阵, 这样可使矩阵中的 (i,j) 元素对应原始数据集中对象 i 和 j 间的距离.

表 11.9 常用的 'metric' 取值及含义

字符串	含义
`'euclidean'`	欧氏距离 (缺省值)
`'cityblock'`	绝对值距离
`'minkowski'`	Minkowski 距离
`'chebychev'`	Chebychev 距离
`'mahalanobis'`	Mahalanobis 距离

2. linkage

Z = linkage(B, 'method') 使用由 'method' 指定的算法计算生成聚类树, 输入矩阵 B 为 pdist 函数输出的 $n\cdot(n-1)/2$ 维距离行向量, 'method' 可取表 11.10 中特征字符串值.

表 11.10 'metric' 取值及含义

字符串	含义
'single'	最短距离 (缺省值)
'average'	无权平均距离
'centroid'	重心距离
'complete'	最大距离
'ward'	离差平方和方法 (Ward 方法)

输出 Z 为包含聚类树信息的 $(n-1)\times 4$ 矩阵. 聚类树上的叶节点为原始数据集中的对象, 其编号由 0 到 $n-1$, 它们是单元素的类, 级别更高的类都由它们生成. 对应于 Z 中第 j 行每个新生成的类, 其索引为 $n+j$, 其中 n 为初始叶节点的数量.

Z 的第 1 列和第 2 列, 即 Z [:, :2] 包含了被两两连接生成一个新类的所有对象的索引. Z [j, :2] 生成的新类索引为 $n+j$. 共有 $n-1$ 个级别更高的类, 它们对应于聚类树中的内部节点.

Z 的第 3 列 Z [:, 2] 包含了相应的在类中的两两对象间的连接距离. Z 的第 4 列 Z [:, 3] 表示当前类中原始对象的个数.

3. fcluster

T = fcluster(Z,t) 从 linkage 的输出 Z, 根据给定的阈值 t 创建聚类.

4. H = dendrogram(Z,p)

由 linkage 产生的数据矩阵 Z 画聚类树状图. p 是结点数, 默认值是 30.

3.4 基于类间距离的层次聚类

层次聚类法是聚类分析方法中使用最多的方法. 其基本思想是: 距离相近的样品 (或变量) 先聚为一类, 距离远的后聚成类, 此过程一直进行下去, 每个样品总能聚到合适的类中. 它包括如下步骤:

(1) 将每个样品独自聚成一类, 构造 n 个类.

(2) 根据所确定的样品距离公式, 计算 n 个样品 (或变量) 两两间的距离, 构造距离矩阵, 记为 $\boldsymbol{D}_{(0)}$.

(3) 把距离最近的两类归为一新类, 其他样品仍各自聚为一类, 共聚成 $n-1$ 类.

(4) 计算新类与当前各类的距离, 将距离最近的两个类进一步聚成一类, 共聚成 $n-2$ 类. 以上步骤一直进行下去, 最后将所有的样品聚成一类.

(5) 画聚类谱系图.

(6) 决定类的个数及各类包含的样品数, 并对类做出解释.

正如样品之间的距离可以有不同的定义方法一样, 类与类之间的距离也有各种定义. 例如, 可以定义类与类之间的距离为两类之间最近样品的距离, 或者定义为两类之间最远样品的距离, 也可以定义为两类重心之间的距离等. 类与类之间用不同的方法定义距离, 就产生了不同的层次聚类方法. 常用的层次聚类方法有, 最短距离法、最长距离法、中间距离法、重心法、类平均法、可变类平均法、可变法和离差平方和法.

下面介绍两种常用的层次聚类法.

1) 最短距离法

最短距离法定义类 G_i 与 G_j 之间的距离为两类间最邻近的两样品之距离, 即 G_i 与 G_j 两类间的距离 D_{ij} 定义为

$$D_{ij} = \min_{\boldsymbol{\omega}_s \in G_i, \boldsymbol{\omega}_t \in G_j} d_{st}.$$

设类 G_p 与 G_q 合并成一个新类记为 G_r, 则任一类 G_k 与 G_r 的距离是

$$D_{kr} = \min_{\boldsymbol{\omega}_i \in G_k, \boldsymbol{\omega}_j \in G_r} d_{ij} = \min\{\min_{\boldsymbol{\omega}_i \in G_k, \boldsymbol{\omega}_j \in G_p} d_{ij}, \min_{\boldsymbol{\omega}_i \in G_k, \boldsymbol{\omega}_j \in G_q} d_{ij}\} = \min\{D_{kp}, D_{kq}\}.$$

最短距离法聚类的步骤如下:

(1) 定义样品之间的距离: 计算样品两两间的距离, 得一距离矩阵记为 $\boldsymbol{D}_{(0)} = (d_{ij})_{n\times n}$, 开始每个样品自成一类, 显然这时 $D_{ij} = d_{ij}$.

(2) 找出 $\boldsymbol{D}_{(0)}$ 的非对角线最小元素, 设为 d_{pq}, 则将 G_p 和 G_q 合并成一个新类, 记为 G_r, 即 $G_r = \{G_p, G_q\}$.

(3) 给出计算新类与其他类的距离公式:

$$D_{kr} = \min\{D_{kp}, D_{kq}\}.$$

将 $\boldsymbol{D}_{(0)}$ 中第 p, q 行及 p, q 列, 用上面公式合并成一个新行新列, 新行新列对应 G_r, 所得到的矩阵记为 $\boldsymbol{D}_{(1)}$.

(4) 对 $\boldsymbol{D}_{(1)}$ 重复上述类似 $\boldsymbol{D}_{(0)}$ 的 (2), (3) 两步得到 $\boldsymbol{D}_{(2)}$. 如此下去, 直到所有的元素并成一类为止.

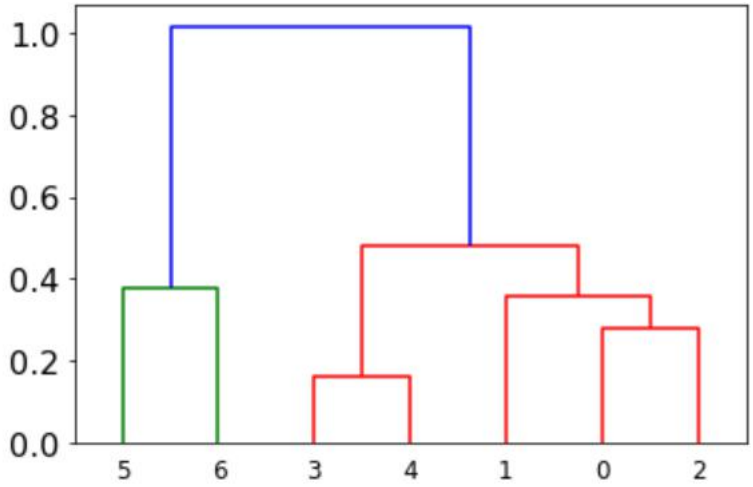
如果某一步 $\boldsymbol{D}_{(k)}$ 中非对角线最小的元素不止一个, 则对应这些最小元素的类可以同时合并.

例 11.11 在某地区有 7 个矽卡岩体, 对 7 个岩体的三种元素 Cu, W, Mo 作分析的原始数据见表 11.11, 对这 7 个样品进行聚类.

表 11.11 7 个矽卡岩体数据

	1	2	3	4	5	6	7
Cu	2.9909	3.2044	2.8392	2.5315	2.5897	2.9600	3.1184
W	0.3111	0.5348	0.5696	0.4528	0.3010	3.0480	2.8395
Mo	0.5324	0.7718	0.7614	0.4893	0.2735	1.4997	1.9850

数学原理及聚类过程就不赘述了. 按照最短距离聚类时, 所画的聚类图如图 11.2 所示. 如果取阈值 $d = 0.5$, 则可把这些岩体划分成两类, 6, 7 为一类, 1, 2, $\cdots$, 5 为另一类.



Python 代码

```
import numpy as np
from sklearn import preprocessing as pp
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
a=np.loadtxt("Pdata11_11.txt")
b=pp.minmax_scale(a.T)     #数据规格化
d = sch.distance.pdist(b)    #求对象之间的两两距离向量
dd = sch.distance.squareform(d)    #转换为矩阵格式
z=sch.linkage(d); print(z) #进行聚类并显示
s=[str(i+1) for i in range(7)]; plt.rc('font',size=16)
sch.dendrogram(z,labels=s); plt.show()    #画聚类图
```

2) 最长距离法

定义类 G_i 与类 G_j 之间的距离为两类最远样品的距离, 即

$$D_{ij} = \max_{\omega_s \in G_i, \omega_t \in G_j} d_{st}.$$

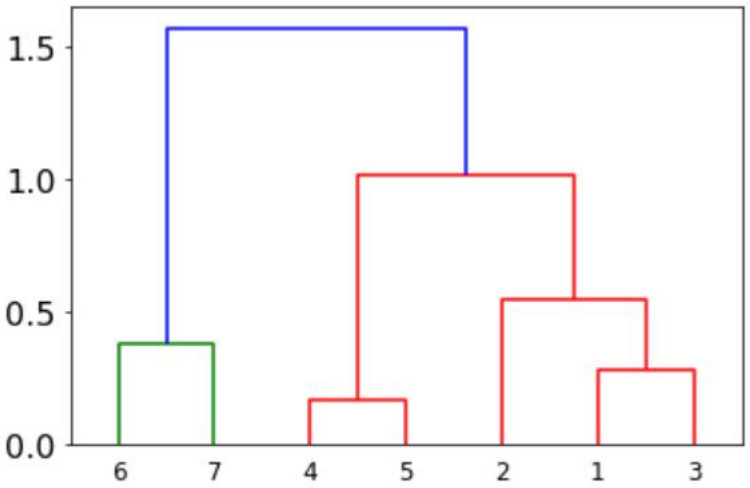
最长距离法与最短距离法的合并步骤完全一样, 也是将各样品先自成一类, 然后将非对角线上最小元素对应的两类合并. 设某一步将类 G_p 与 G_q 合并为 G_r, 则任一类 G_k 与 G_r 的最长距离公式为

$$D_{kr} = \max_{\omega_i \in G_k, \omega_j \in G_r} d_{ij} = \max\{\max_{\omega_i \in G_k, \omega_j \in G_p} d_{ij}, \max_{\omega_i \in G_k, \omega_j \in G_q} d_{ij}\} = \max\{D_{kp}, D_{kq}\}.$$

再找非对角线最小元素对应的两类并类, 直至所有的样品全归为一类为止.

可见, 最长距离法与最短距离法只有两点不同, 一是类与类之间的距离定义不同; 二是计算新类与其他类的距离所用的公式不同.

例 11.12 (续例 11.11)　用最长距离法对 7 个矽卡岩体进行聚类. 所画的聚类图如图 11.3 所示. 聚类结果和例 11.11 是一样的.



Python 代码

```
import numpy as np
from sklearn import preprocessing as pp
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
a=np.loadtxt("Pdata11_11.txt")
b=pp.minmax_scale(a.T)    #数据规格化
d = sch.distance.pdist(b)   #求对象之间的两两距离向量
z=sch.linkage(d,'complete'); print(z)   #进行聚类并显示
s=[str(i+1) for i in range(7)]; plt.rc('font',size=16)
sch.dendrogram(z,labels=s); plt.show()   #画聚类图
```

3.5 K 均值聚类

用层次聚类法聚类时, 随着聚类样本对象的增多, 计算量会迅速增加, 而且聚类结果——谱系图会十分复杂, 不便于分析. 特别是样品的个数很大 (如 $n \geqslant 100$) 时, 层次聚类法的计算量非常大, 将占据大量的计算机内存空间和较多的计算时间. 为了改进上述缺点, 一个自然的想法是先粗略地分一下类, 然后按某种最优原则进行修正, 直到将类分得比较合理为止. 基于这种思想就产生了动态聚类法, 也称逐步聚类法.

动态聚类法适用于大型数据. 动态聚类法有许多种方法, 这里介绍一种比较流行的动态聚类法——K 均值法. 它是一种快速聚类法, 该方法得到的结果简单易懂, 对计算机的性能要求不高, 因而应用广泛. 该方法由麦克奎因 (Macqueen) 于 1967 年提出.

算法的思想是假定样本集中的全体样本可分为 C 类, 并选定 C 个初始聚类中心, 然后根据最小距离原则将每个样本分配到某一类中, 之后不断迭代计算各类的聚类中心, 并依据新的聚类中心调整聚类情况, 直到迭代收敛或聚类中心不再改变.

K 均值聚类算法最后将总样本集 G 划分为 C 个子集：$G_1, G_2, \cdots, G_C$，它们满足下面条件：

(1) $G_1 \cup G_2 \cup \cdots \cup G_C = G$;

(2) $G_i \cap G_j = \varnothing$ $(1 \leqslant i < j \leqslant C)$;

(3) $G_i \neq \varnothing$, $G_i \neq G$ $(1 \leqslant i \leqslant C)$.

设 $\boldsymbol{m}_i$ $(i = 1, \cdots, C)$ 为 C 个聚类中心，记

$$J_e = \sum_{i=1}^{C} \sum_{\boldsymbol{\omega} \in G_i} \|\boldsymbol{\omega} - \boldsymbol{m}_i\|^2,$$

使 J_e 最小的聚类是误差平方和准则下的最优结果.

K 均值聚类算法描述如下：

(1) 初始化. 设总样本集 $G = \{\boldsymbol{\omega}_j, j = 1, 2, \cdots, n\}$ 是 n 个样品组成的集合，聚类数为 C $(2 \leqslant C \leqslant n)$，将样本集 G 任意划分为 C 类，记为 $G_1, G_2, \cdots, G_C$，计算对应的 C 个初始聚类中心，记为 $\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_C$，并计算 J_e.

(2) $G_i = \varnothing$ $(i = 1, 2, \cdots, C)$，按最小距离原则将样品 $\boldsymbol{\omega}_j$ $(j = 1, 2, \cdots, n)$ 进行聚类，即若 $d(\boldsymbol{\omega}_j, G_k) = \min\limits_{1 \leqslant i \leqslant C} d(\boldsymbol{\omega}_j, \boldsymbol{m}_i)$，则 $\boldsymbol{\omega}_j \in G_k$，$G_k = G_k \cup \{\boldsymbol{\omega}_j\}$，$j = 1, 2, \cdots, n$. 重新计算聚类中心

$$\boldsymbol{m}_i = \frac{1}{n_i} \sum_{\boldsymbol{\omega}_j \in G_i} \boldsymbol{\omega}_j, \quad i = 1, 2, \cdots, C,$$

式中，n_i 为当前 G_i 类中的样本数目，并重新计算 J_e.

(3) 若连续两次迭代的 J_e 不变，则算法终止，否则算法转 (2).

注 11.2 实际计算时，可以不计算 J_e，只要聚类中心不发生变化，算法即可终止.

例 11.13 已知聚类的指标变量为 x_1, x_2，四个样本点的数据分别为

$$\boldsymbol{\omega}_1 = (1, 3), \quad \boldsymbol{\omega}_2 = (1.5, 3.2), \quad \boldsymbol{\omega}_3 = (1.3, 2.8), \quad \boldsymbol{\omega}_4 = (3, 1).$$

试用 K 均值聚类法把样本点分成两类.

解 现要分为两类 G_1 和 G_2，设初始聚类为 $G_1 = \{\boldsymbol{\omega}_1\}$，$G_2 = \{\boldsymbol{\omega}_2, \boldsymbol{\omega}_3, \boldsymbol{\omega}_4\}$，则初始聚类中心为

G_1 类：$\boldsymbol{\omega}_1$ 值，即 $\boldsymbol{m}_1 = (1, 3)$;

G_2 类：$\boldsymbol{m}_2 = \left(\dfrac{1.5 + 1.3 + 3}{3}, \dfrac{3.2 + 2.8 + 1}{3}\right) = (1.9333, 2.3333)$.

计算每个样本点到 G_1, G_2 聚类中心的距离

$$d_{11} = \|\boldsymbol{\omega}_1 - \boldsymbol{m}_1\| = \sqrt{(1-1)^2 + (3-3)^2} = 0, \quad d_{12} = \|\boldsymbol{\omega}_1 - \boldsymbol{m}_2\| = 1.1470;$$
$$d_{21} = \|\boldsymbol{\omega}_2 - \boldsymbol{m}_1\| = 0.5385, \quad d_{22} = \|\boldsymbol{\omega}_2 - \boldsymbol{m}_2\| = 0.9690;$$
$$d_{31} = \|\boldsymbol{\omega}_3 - \boldsymbol{m}_1\| = 0.3606, \quad d_{32} = \|\boldsymbol{\omega}_3 - \boldsymbol{m}_2\| = 0.7867;$$
$$d_{41} = \|\boldsymbol{\omega}_4 - \boldsymbol{m}_1\| = 2.8284, \quad d_{42} = \|\boldsymbol{\omega}_4 - \boldsymbol{m}_2\| = 1.7075.$$

得到新的划分为 $G_1 = \{\omega_1, \omega_2, \omega_3\}$, $G_2 = \{\omega_4\}$, 新的聚类中心为

$$G_1 \text{ 类: } \boldsymbol{m}_1 = \left(\frac{1+1.5+1.3}{3}, \frac{3+3.2+2.8}{3}\right) = (1.2667, 3.0);$$

G_2 类: ω_4 值, 即 $\boldsymbol{m}_2 = (3, 1)$.

重新计算每个样本点到 G_1, G_2 聚类中心的距离

$$d_{11} = \|\boldsymbol{\omega}_1 - \boldsymbol{m}_1\| = 0.2667, \quad d_{12} = \|\boldsymbol{\omega}_1 - \boldsymbol{m}_2\| = 2.8284;$$
$$d_{21} = \|\boldsymbol{\omega}_2 - \boldsymbol{m}_1\| = 0.3073, \quad d_{22} = \|\boldsymbol{\omega}_2 - \boldsymbol{m}_2\| = 2.6627;$$
$$d_{31} = \|\boldsymbol{\omega}_3 - \boldsymbol{m}_1\| = 0.2028, \quad d_{32} = \|\boldsymbol{\omega}_3 - \boldsymbol{m}_2\| = 2.4759;$$
$$d_{41} = \|\boldsymbol{\omega}_4 - \boldsymbol{m}_1\| = 2.6466, \quad d_{42} = \|\boldsymbol{\omega}_4 - \boldsymbol{m}_2\| = 0.$$

所以, 得新的划分为: $G_1 = \{\omega_1, \omega_2, \omega_3\}$, $G_2 = \{\omega_4\}$.

可见, 新的划分与前面的划分相同, 聚类中心没有改变, 聚类结束.

Python 代码

```
import numpy as np
from sklearn.cluster import KMeans
a = np.array([[1, 3], [1.5, 3.2], [1.3, 2.8], [3, 1]])
md = KMeans(n_clusters=2)  # 构建模型
md.fit(a)  # 求解模型
labels = 1 + md.labels_  # 提取聚类标签
centers = md.cluster_centers_  # 提取聚类中心,每一行是一个聚类中心
print(labels, '\n----------\n', centers)
```

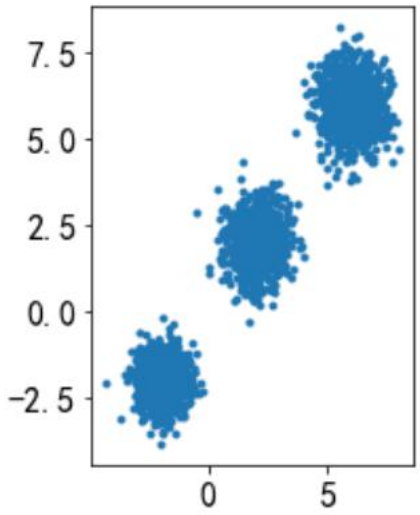
3.6 K 均值聚类法最佳簇数 k 值的确定

对于 K 均值聚类来说, 如何确定簇数 k 值是一个至关重要的问题, 为了解决这个问题, 通常会选用探索法, 即给定不同的 k 值, 对比某些评估指标的变动情况, 进而选择一个比较合理的 k 值. 本节将介绍非常实用的两种评估方法, 即簇内离差平方和拐点法与轮廓系数法.

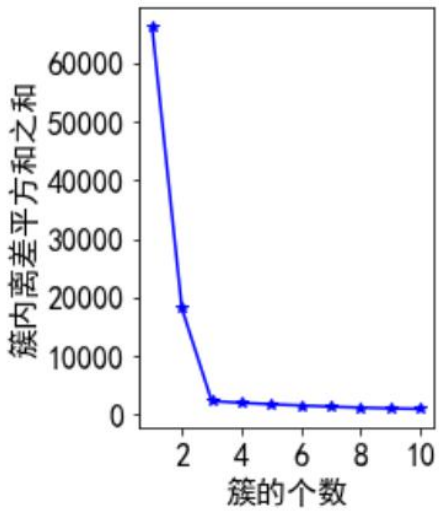
1. 簇内离差平方和拐点法

簇内离差平方和拐点法的思想很简单, 就是在不同的 k 值下计算簇内离差平方和, 然后通过可视化的方法找到 "拐点" 所对应的 k 值. 重点关注的是斜率的变化, 当斜率由大突然变小时, 并且之后的斜率变化缓慢, 则认为突然变换的点就是寻找的目标点, 因为继续随着簇数 k 的增加, 聚类效果不再有大的变化.

为了验证这个方法的直观性, 这里随机生成三组二维正态分布数据, 首先基于该数据绘制散点图如图 11.4(a) 所示, 模拟的数据呈现三个簇. 接下来基于这个模拟数据, 使用拐点法, 绘制簇的个数与总的簇内离差平方和之间的折线图如图 11.4(b) 所示.



(a) 生成三个簇的样本点



(b)拐点法选择合理的 k 值

从图 11.4(b) 可以看出, 当簇的个数为 3 时形成了一个明显的拐点, 3 之后的簇对应的簇内离差平方和的变动都很小, 合理的 k 值应该为 3, 与模拟的三个簇数据是吻合的.

Python 代码

```
import numpy as np
import matplotlib.pyplot as plt; from sklearn.cluster import KMeans
mean=np.array([[-2, -2],[2, 2], [6,6]])
cov=np.array([[[0.3, 0], [0, 0.3]],[[0.4, 0], [0, 0.4]],[[0.5, 0], [0, 0.5]]])
x0=[]; y0=[];
for i in range(3):
      x,y=np.random.multivariate_normal(mean[i], cov[i],1000).T
      x0=np.hstack([x0,x]); y0=np.hstack([y0,y])
plt.rc('font',size=16); plt.rc('font',family='SimHei')
plt.rc('axes',unicode_minus=False); plt.subplot(121)
plt.scatter(x0,y0,marker='.')    #画模拟数据散点图
X=np.vstack([x0,y0]).T
np.save("Pzdata11_1.npy",X)    #保存数据供下面使用
TSSE=[]; K=10
for k in range(1,K+1):
      SSE = []
      md = KMeans(n_clusters=k); md.fit(X)
      labels = md.labels_; centers = md.cluster_centers_
      for label in set(labels):
            SSE.append(np.sum((X[labels == label,:]-centers[label,:])**2))
      TSSE.append(np.sum(SSE))
plt.subplot(122); plt.style.use('ggplot')
plt.plot(range(1,K+1), TSSE, 'b*-')
plt.xlabel('簇的个数'); plt.ylabel('簇内离差平方和之和'); plt.show()
```

2. 轮廓系数法

该方法综合考虑了簇的密集性与分散性两个信息，如果数据集被分割为理想的 k 个簇，那么对应的簇内样本会很密集，而簇间样本会很分散.

如图 11.5 所示，假设数据集被拆分为三个簇 G_1, G_2, G_3，样本点 i 对应的 a_i 值为所有 G_1 中其他样本点与样本点 i 的距离平均值；样本点 i 对应的 b_i 值分两步计算，首先计算该点分别到 G_2 和 G_3 中样本点的平均距离，然后将两个平均值中的最小值作为 b_i 的度量.

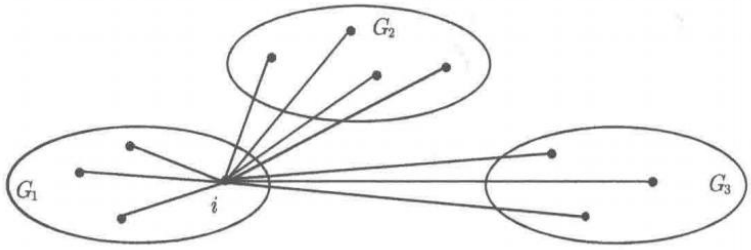


图 11.5　轮廓系数计算示意图

定义样本点 i 的轮廓系数

$$S_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \tag{11.14}$$

k 个簇的总轮廓系数定义为所有样本点轮廓系数的平均值.

当总轮廓系数小于 0 时, 说明聚类效果不佳; 当总轮廓系数接近于 1 时, 说明簇内样本的平均距离非常小, 而簇间的最近距离非常大, 进而表示聚类效果非常理想.

上面的计算思想虽然简单, 但是计算量是很大的, 当样本量比较多时, 运行时间会比较长. 有关轮廓系数的计算, 可以直接调用 sklearn.metrics 中的函数 silhouette_score. 需要注意的是, 该函数接受的聚类簇数必须大于等于 2.

利用上面同样的模拟数据, 画出的簇数与轮廓系数对应关系图如图 11.6 所示, 当 k 等于 3 时, 轮廓系数最大, 且比较接近于 1, 说明应该把模拟数据聚为 3 类比较合理, 同样与模拟数据的三个簇是吻合的.

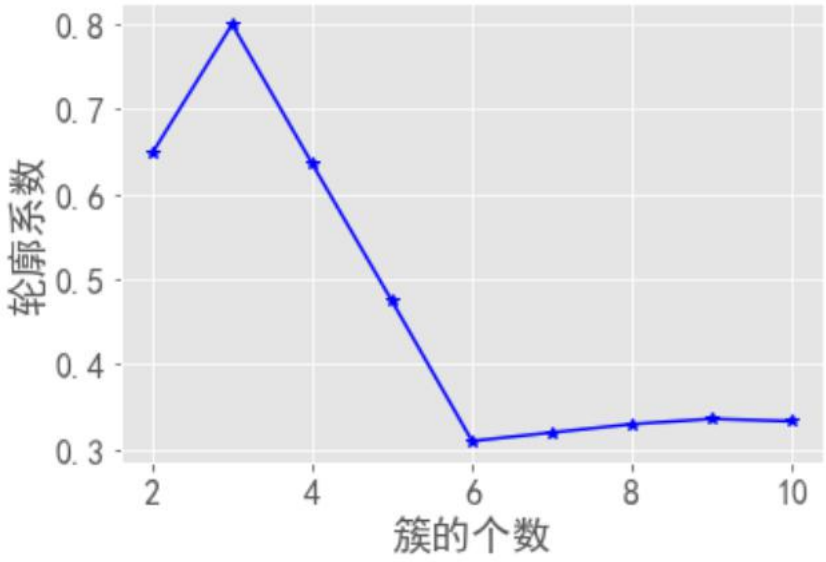


图 11.6 轮廓系数法选择合理的 k 值

Python 代码

```
import numpy as np; import matplotlib.pyplot as plt
from sklearn.cluster import KMeans; from sklearn import metrics
X=np.load("Pzdata11_1.npy")
S=[]; K=10
for k in range(2,K+1):
      md = KMeans(k); md.fit(X)
      labels = md.labels_;
      S.append(metrics.silhouette_score(X,labels,metric='euclidean'))    #计算轮廓系数
plt.rc('font',size=16); plt.rc('font',family='SimHei')
plt.plot(range(2,K+1), S, 'b*-')
plt.xlabel('簇的个数'); plt.ylabel('轮廓系数'); plt.show()
```

3.7 K 均值聚类的应用

在做 K 均值聚类时需要注意两点, 一个是聚类前必须指定具体的簇数 k 值, 如果 k 值是已知的, 可以直接调用 cluster 子模块中的 KMeans 函数, 对数据集进行分割; 如果 k 值是未知的, 可以根据行业经验或前面介绍的两种方法确定合理的 k 值.

另一个是对原始数据集做必要的标准化处理. 由于 K 均值的思想是基于点之间的距离实现 "物以类聚" 的, 所以如果原始数据集存在量纲上的差异, 就必须对其进行标准化的预处理. 数据集的标准化预处理可以借助 sklearn 子模块 preprocessing 中的 scale 函数或 minmax_scale 函数. scale 函数的标准化公式为

$$x^* = \frac{x-\mu}{\sigma}, \tag{11.15}$$

minmax_scale 函数的标准化公式为

$$x^* = \frac{x-x_{\min}}{x_{\max}-x_{\min}}, \tag{11.16}$$

其中, $\mu, \sigma, x_{\min}, x_{\max}$ 分别为 x 取值的均值、标准差、最小值和最大值.

Iris 数据集是常用的分类实验数据集, 下面用该数据集来验证 K 均值聚类的效果.

例 11.14 Iris 数据集由 Fisher 于 1936 年收集整理. Iris 也称鸢尾花卉数据, 是一类多重变量分析的数据集. 数据集包含 150 个数据, 分为 3 类, 每类 50 个数据, 每个数据包含 4 个属性, 数据格式如表 11.12 所示. 可通过花萼长度、花萼宽度、花瓣长度、花瓣宽度 4 个属性预测鸢尾花卉属于 (setosa, versicolour, virginica) 三个种类中的哪一类.

表 11.12 Iris 数据集数据

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
⋮	⋮	⋮	⋮	⋮	⋮
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3	5.1	1.8	virginica

如表 11.12 所示, 数据集的前 4 个变量分别为花萼的长度、宽度及花瓣的长度、宽度, 它们之间没有量纲上的差异, 故无需对其做标准化预处理, 最后一个变量为鸢尾花所属的种类. 如果将其聚为 3 类, 所得结果为各簇样本量分别为 60, 50, 38. 为了直观验证聚类效果, 对比建模后的 3 类与原始数据 3 类的差异, 绘制花瓣长度与宽度的散点图如图 11.7 所示.

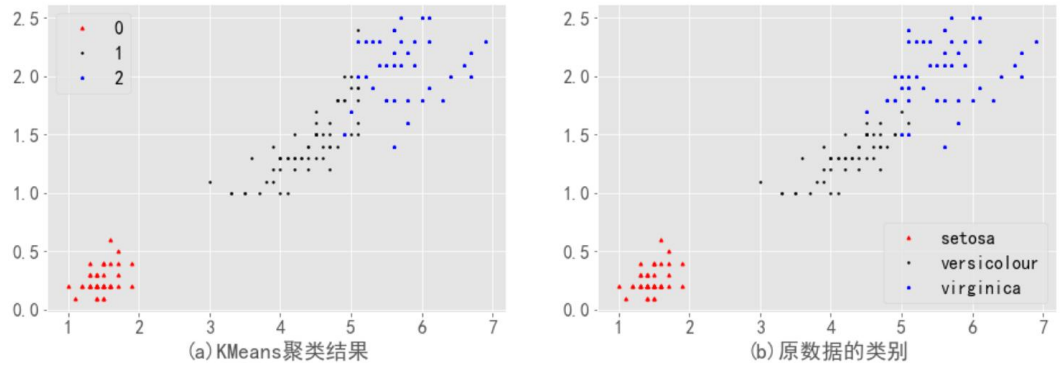


图 11.7　K 均值聚类效果与原始数据类别的对比

Python 代码

```
import numpy as np; import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
a=pd.read_csv("iris.csv")
b=a.iloc[:,:-1]
md=KMeans(3); md.fit(b)    #构建模型并求解模型
labels=md.labels_ ;    centers=md.cluster_centers_
b['cluster']=labels    #数据框b添加一个列变量cluster
c=b.cluster.value_counts()    #各类频数统计
plt.rc('font',family='SimHei'); plt.rc('font',size=16)
str1=['^r','.k','*b']; plt.subplot(121)
for i in range(len(centers)):
      plt.plot(b['Petal_Length'][labels==i],b['Petal_Width']
                  [labels==i], str1[i],markersize=3,label=str(i))
      plt.legend(); plt.xlabel("(a)KMeans聚类结果")
plt.subplot(122); str2=['setosa','versicolour','virginica']
ind=np.hstack([np.zeros(50),np.ones(50),2*np.ones(50)])
for i in range(3):
      plt.plot(b['Petal_Length'][ind==i],b['Petal_Width'][ind==i],
                        str1[i],markersize=3,label=str2[i])
      plt.legend(loc='lower right'); plt.xlabel("(b)原数据的类别")
plt.show()
```