

# 15

## Hierarchical Clustering

# 层次聚类

基于数据之间距离，自下而上聚合，或自上而下分裂



如果不能简单地解释某个理论，说明你并没有真正理解它。

***If you can't explain it simply, you don't understand it well enough.***

—— 阿尔伯特·爱因斯坦 (Albert Einstein) | 理论物理学家 | 1879 ~ 1955



```
◀ numpy.triu() 提取上三角矩阵
◀ scipy.cluster.hierarchy.dendrogram() 绘制树形图
◀ scipy.cluster.hierarchy.linkage() 计算簇间距离
◀ seaborn.clustermap() 绘制树形图和热图
◀ seaborn.heatmap() 绘制热图
◀ sklearn.cluster.AgglomerativeClustering() 层次聚类函数
◀ sklearn.metrics.pairwise.rbf_kernel() 计算 RBF 核成对亲密度矩阵
```

## 15.1 层次聚类

**层次聚类** (hierarchical clustering) 算法是一种聚类分析算法。层次聚类依据数据之间的距离远近，或者亲近度大小，将样本数据划分为簇。层次聚类可以通过**自下而上** (agglomerative) 合并，或者**自上而下** (divisive) 分割来构造分层结构聚类。

图 1 所示为根据鸢尾花样本数据前两个特征——花萼长度和宽度——获得的层次聚类**树形图** (dendrogram)。

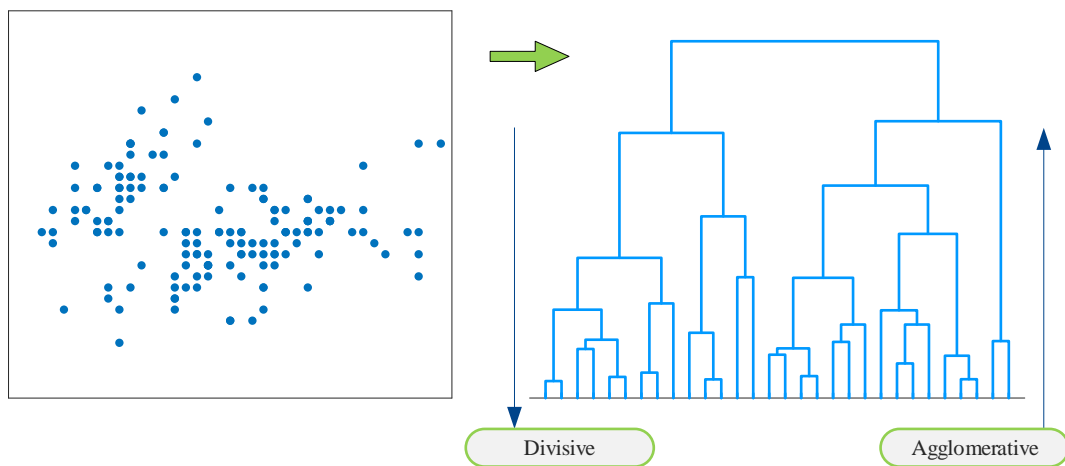


图 1. 区分“自上而下”和“自下而上”层次聚类

### 自下而上合并

图 2 所示为自下而上合并原理。整个过程有点像“搭积木”，首先以每个数据点本身作为一簇，每次迭代合并“距离”较近或亲近度大的类别，直到最后只剩一簇为止。这个过程可以使用的距离度量或亲和度也是多种多样的。请大家回顾本书第 3 章有关距离度量和亲近度内容。

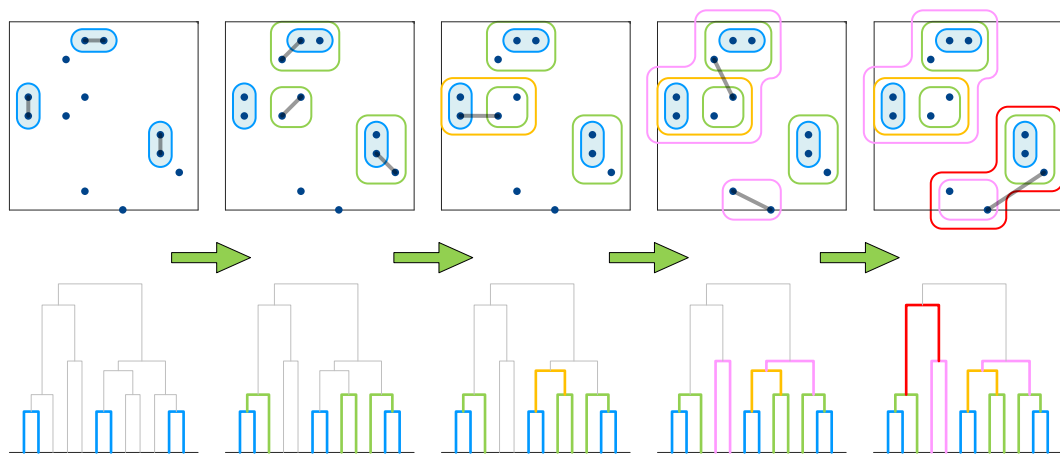


图 2. 层次聚类原理

本章下面首先介绍如何一步步通过自下而上合并获得树形图。大家可能已经注意到，图 2 中不仅仅要考虑，“点”与“点”之间距离，还需要考虑“簇”与“簇”之间的距离。“簇”与“簇”之间的距离，也是本章要探讨的核心内容之一。

## 15.2 树形图

图 3 给出 12 个样本数据在平面上的位置。相信大家还记得成对距离矩阵 (pairwise distance matrix) 这个概念。图 4 所示 12 个样本数据成对欧氏距离矩阵的热图。

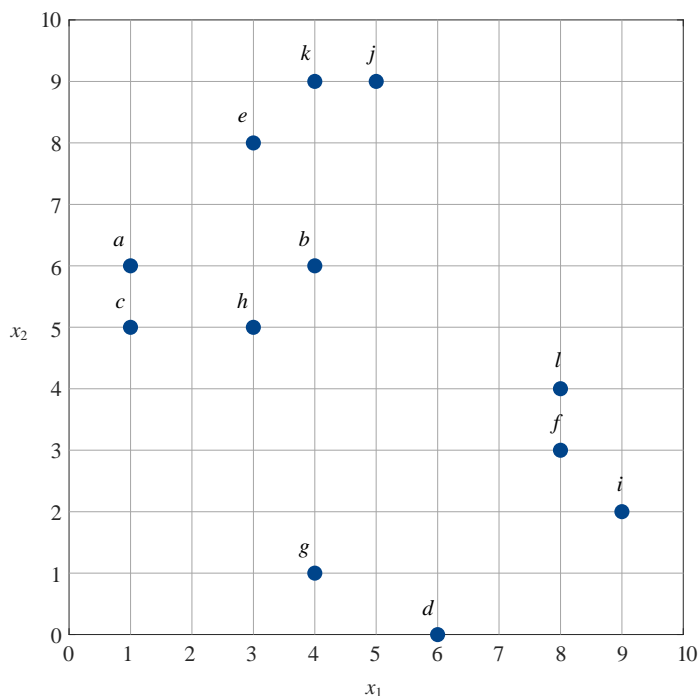


图 3. 12 个样本数据

有了图 4 所示成对欧氏距离矩阵，便可以得到如图 5 所示树形图。树形图横轴对应样本数据编号，纵轴对应数据点间距离和簇间欧氏距离。

观察图 5 树形图，在距离值 2.5 处切一刀，可以将图 3 所示数据分成 3 簇；如果在距离值为 4 处切一刀，可以将图 3 所示数据分成 2 簇。下面，我们一步步介绍如何自下而上构造如图 5 所示树形图。

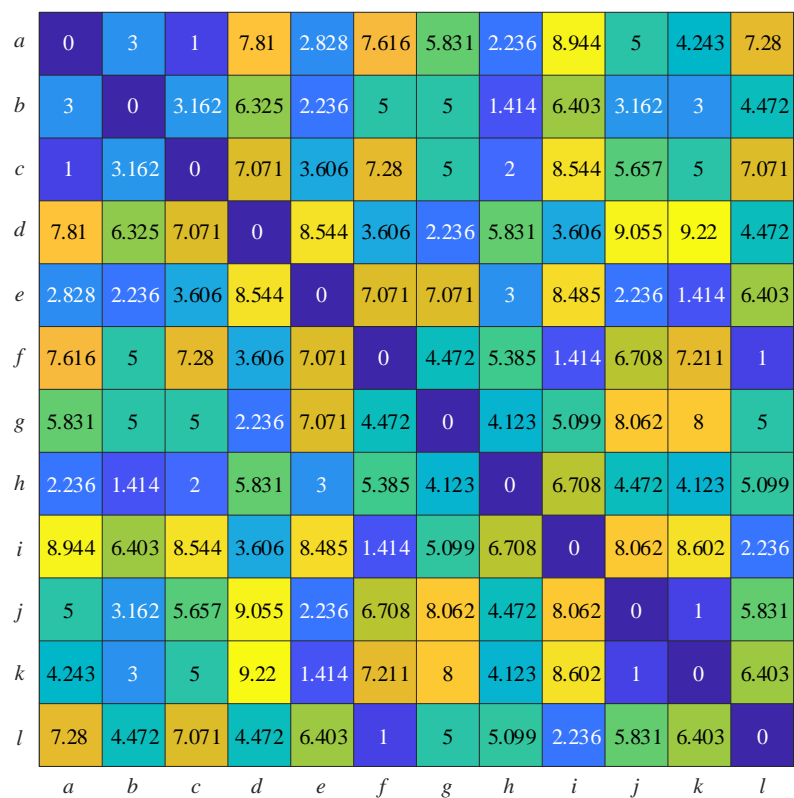


图 4. 12 个样本数据成对距离构成的方阵热图

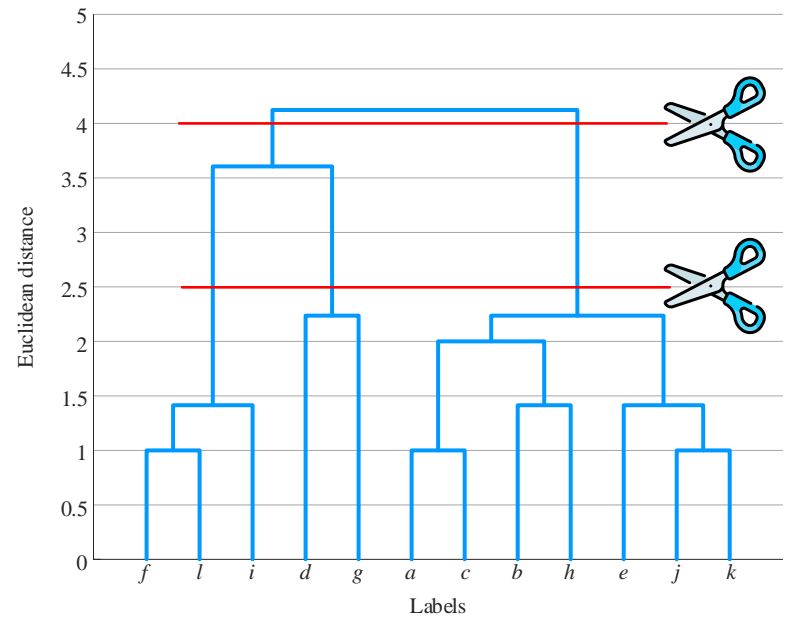


图 5. 数据树形图

第一层

图 6 所示，首先发现  $a$  和  $c$ 、 $k$  和  $j$ 、 $f$  和  $l$  成对距离最短，均为 1；这样我们便构造树形图最底层。这三个成对距离在热图位置如图 8 (a) 所示。

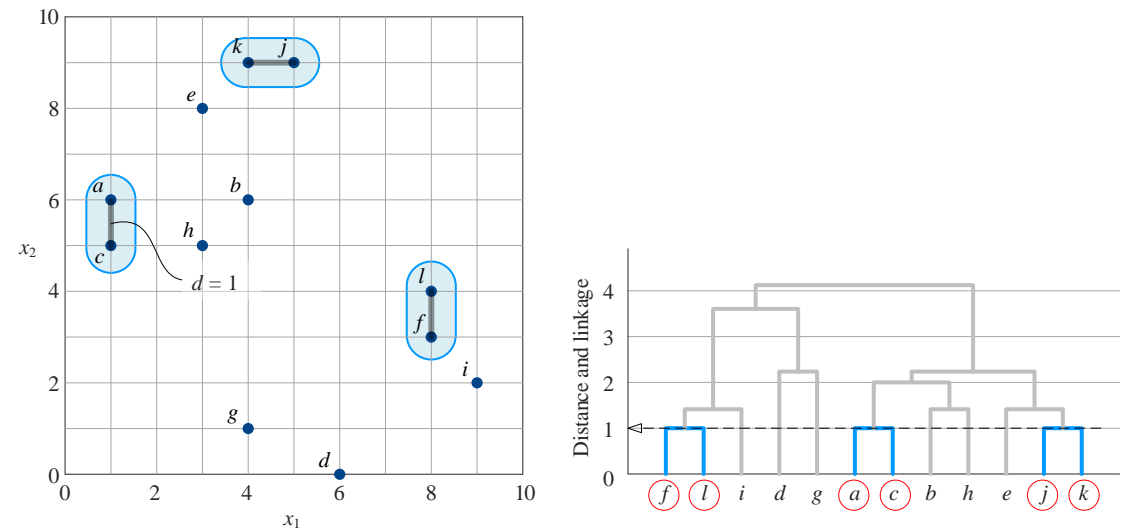


图 6. 构建树形图，第一层

第二层

构造树形图第二层时，遇到一个麻烦——簇间距离如何定义。这里，我们首先采用最简单的最近点距离 (single linkage 或 nearest neighbor)。最近点距离指的是两个簇样本数据成对距离最近值。

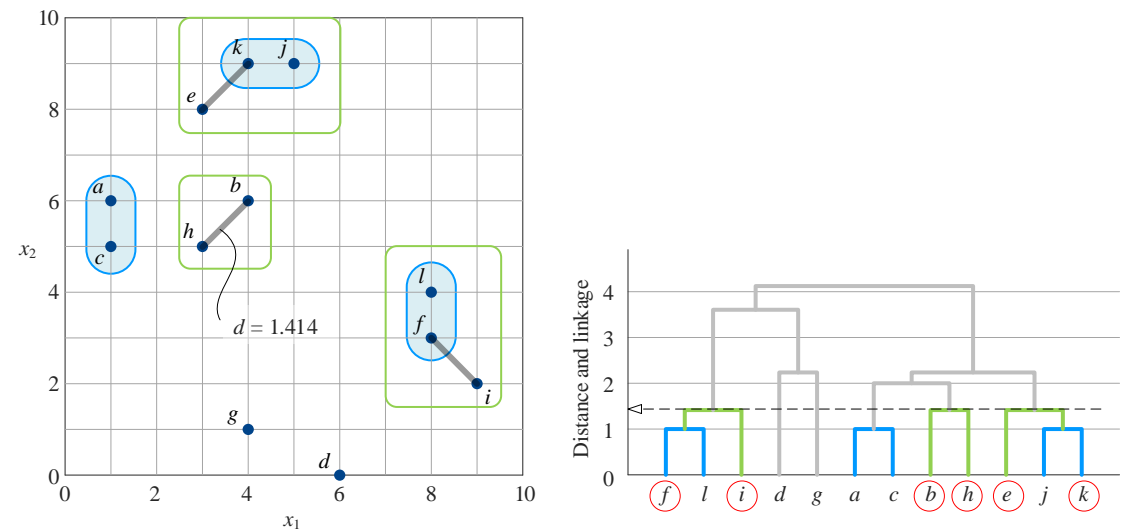


图 7. 构建树形图，第二层

$k$  和  $j$ 、 $f$  和  $l$  已经分别“成团”； $e$  距离  $k$  更近，而  $i$  距离  $f$  更近。因此树形图第二层的距离值定为  $\sqrt{2}$ ，也就是约 1.414。同样， $b$  和  $h$  的距离也是 1.414。这样我们便构造得到了如图 7 所示的树形图第二层。这三个“簇间”/“点间”距离在热图位置如图 8 (b) 所示。

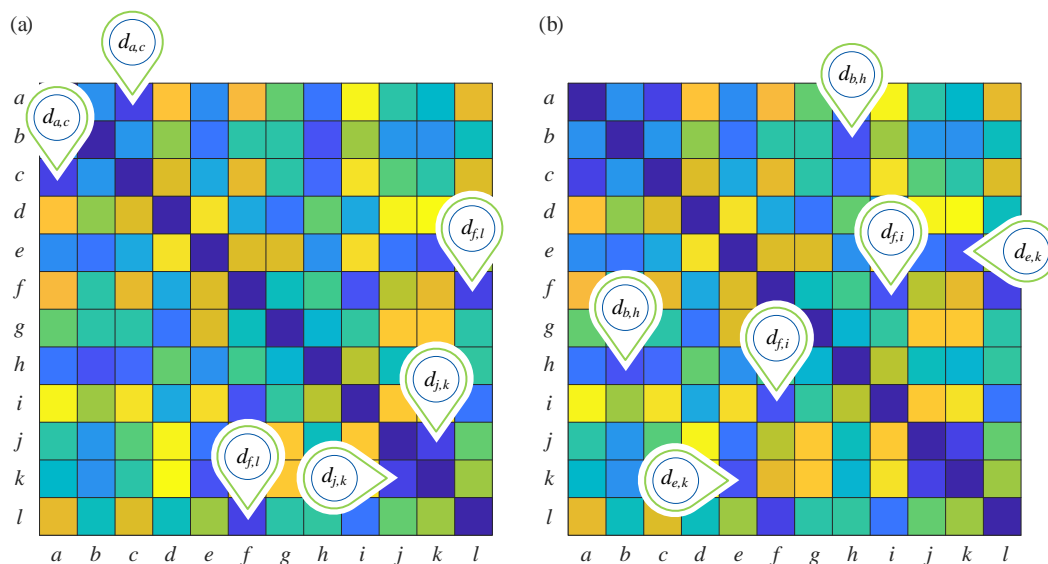


图 8. 成对距离矩阵热图，第一层和第二层距离位置

### 第三层

再向上一层，利用簇  $a$  和  $c$  (第一层)、簇  $b$  和  $h$  (第二层) 之间簇间距离 2，从而得到树形图第三层。这个距离在热图上的位置如图 11 (a) 所示。

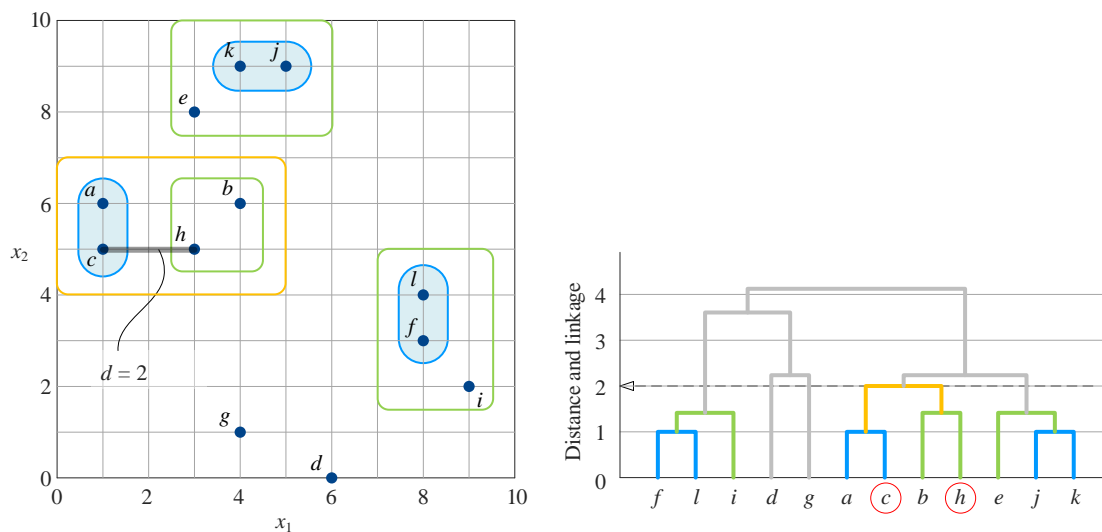


图 9. 构建树形图，第三步

第四层

树形图的第四层采用的距离值为  $\sqrt{5}$ ，约 2.236。图 10 所示为树形图第四层位置。可以发现此时，所有的数据点均参与聚类，形成 3 簇。

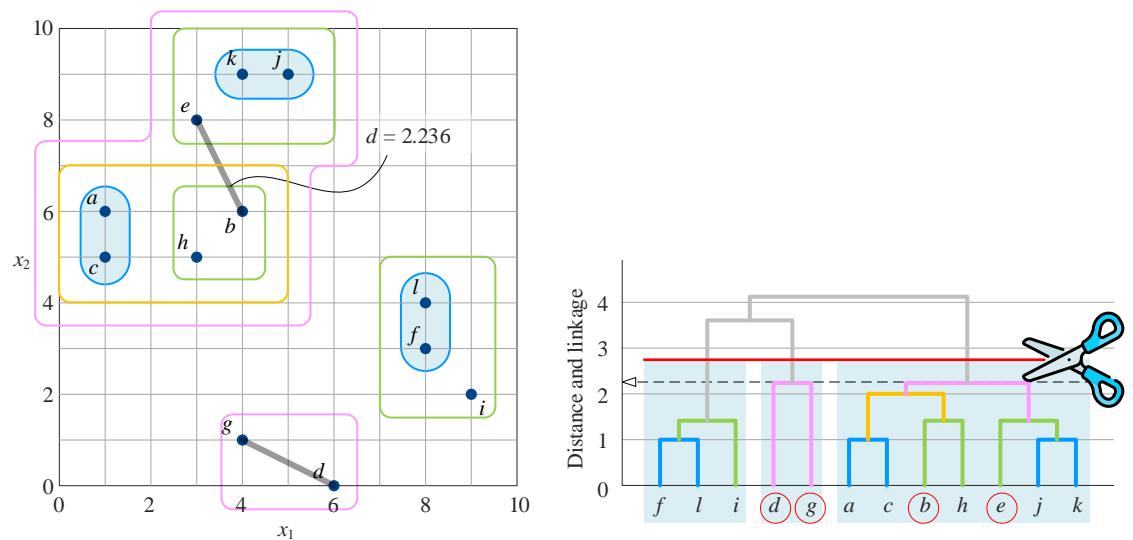


图 10. 构建树形图，第四层

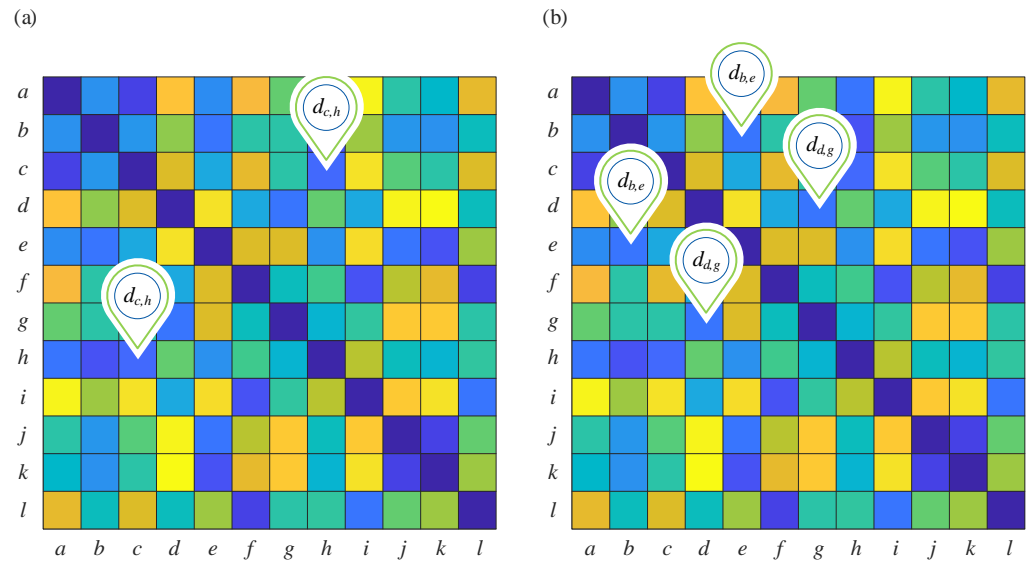


图 11. 成对距离矩阵热图，第三层和第四层距离位置

第五层

图 12 所示为树形图第五层位置。在第五层，样本数据被划分为两簇；再加一层，整个树形图便封顶。

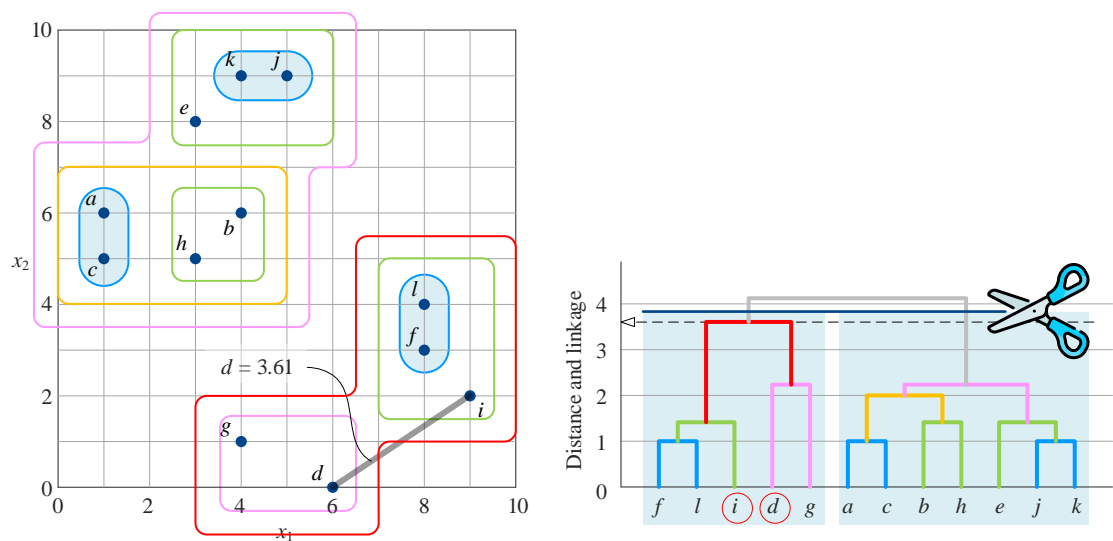


图 12. 构建树形图，第五层

重新排序

树形结构把数据序号重新排列。根据这个顺序，可以得到一个全新的热图，如图 13 所示。根据颜色，图 13 所示热图很容易分为两个区域，对应数据划分为两簇。这便是层次聚类的思路。

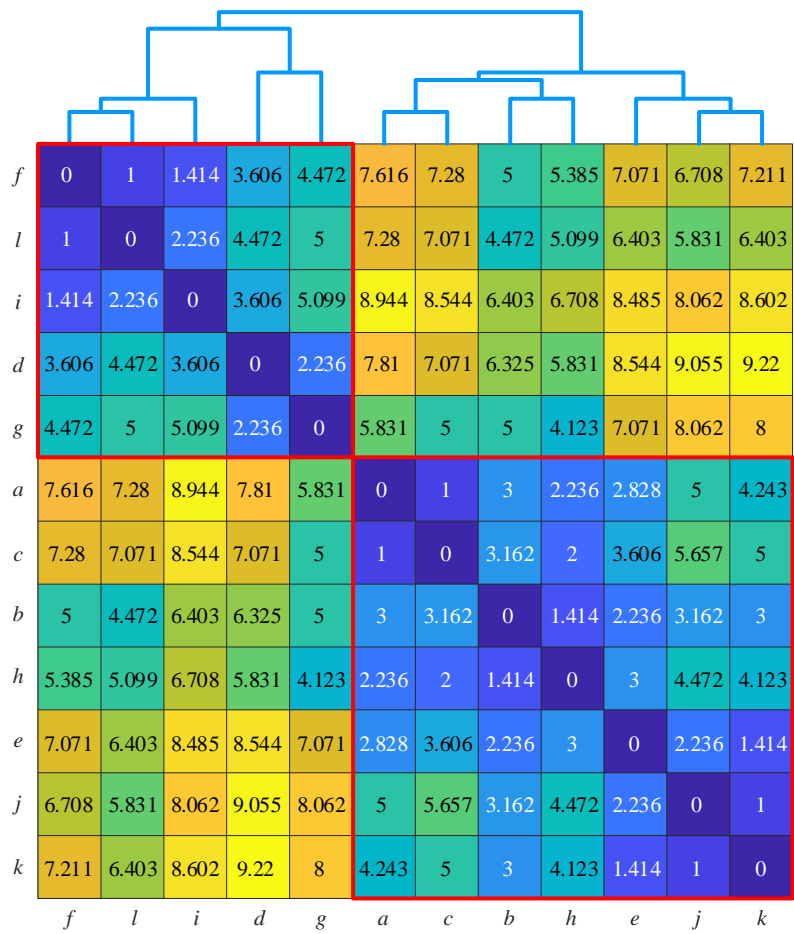




图 13. 按树形图重组数据树形图

## 15.3 簇间距离

上一节提到，两簇之间的距离可以采用最近点距离；当然，簇间距离也有其他定义。本节介绍常用的几种簇间距离。

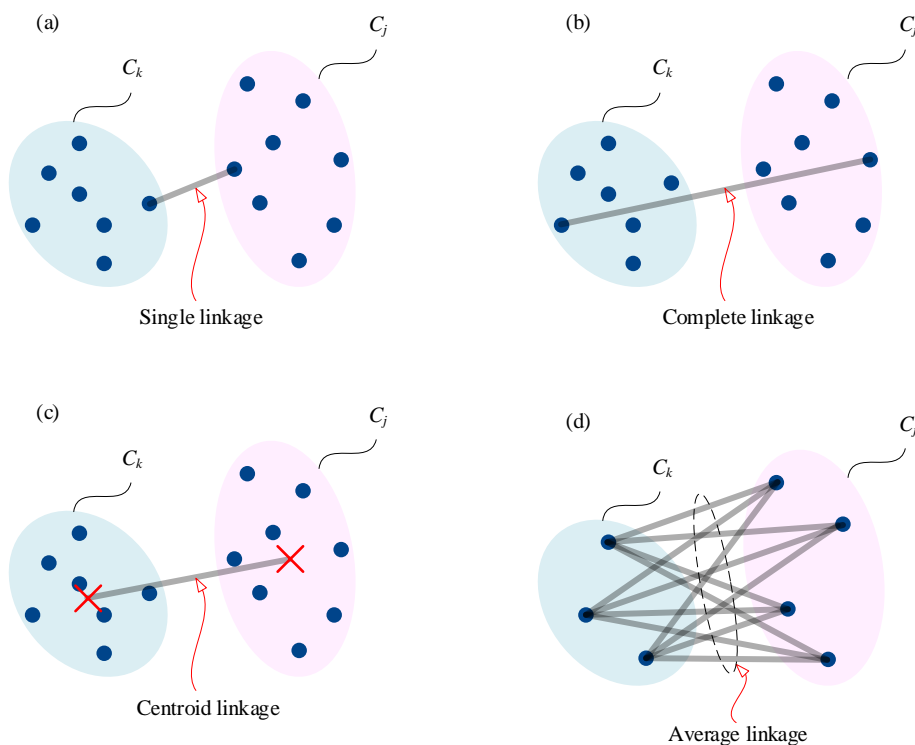


图 14. 簇间距离定义

### 最近点距离

簇间距离，也叫做**距离值** (linkage distance 或者 linkage)。如图 14 (a) 所示，**最近点距离** (single linkage 或 nearest neighbor)，代号为 'single'，指的是两个簇样本数据成对距离最近值：

$$d(C_k, C_j) = \min_{x \in C_k, z \in C_j} (\text{dist}(x, z)) \quad (1)$$

图 15 所示为，采用 'single' 层次聚类得到的树形图和鸢尾花数据聚类结果。可以发现，树形图分支并不均衡，聚类结果不理想。

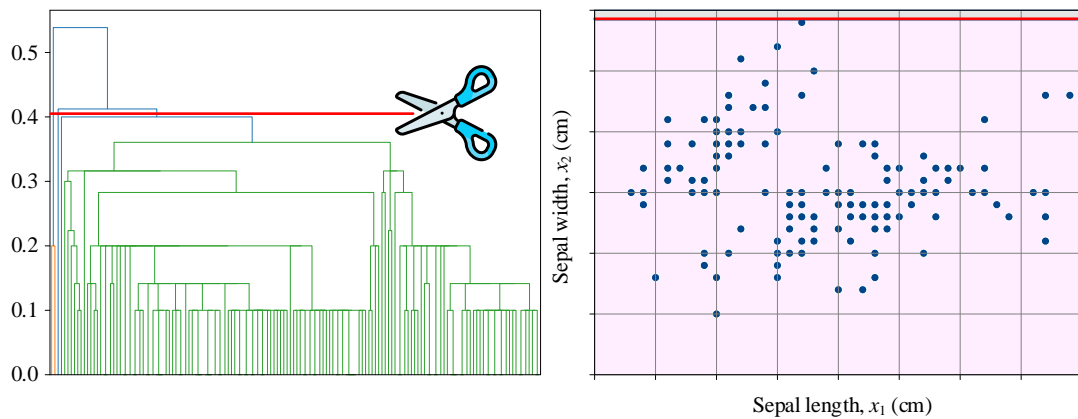


图 15. 鸢尾花聚类结果, 'single' 层次聚类

### 最远点距离

如图 14 (b) 所示, **最远点距离** (complete linkage 或 farthest neighbor) 定义为, 两簇样本数据成对距离最远值:

$$d(C_k, C_j) = \max_{x \in C_k, z \in C_j} (\text{dist}(x, z)) \quad (2)$$

最远点距离代号为 **'complete'**。图 16 所示为, 采用 **'complete'** 层次聚类得到的树形图和鸢尾花数据聚类结果。最远点距离对于离群点/噪音点敏感。

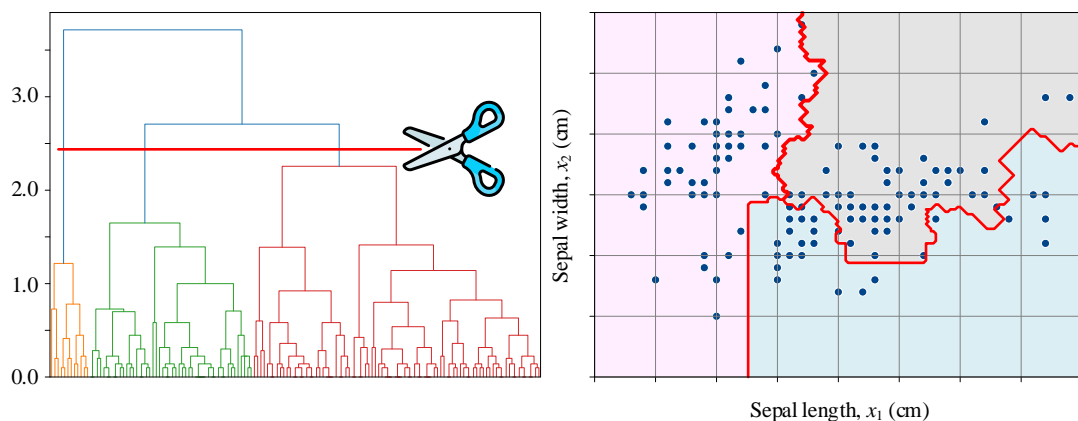


图 16. 鸢尾花聚类结果, 'complete' 层次聚类

### 均值点距离

如图 14 (c) 所示, **均值点距离** (centroid linkage) 采用两个簇样本数据均值点之间的距离:

$$d(C_i, C_j) = d(\mu_i, \mu_j) \quad (3)$$

其中,  $\mu_i$  和  $\mu_j$  分别为  $C_i$  和  $C_j$  的质心点。目前 Scikit-learn 中的层次聚类函数并不支持均值点距离; 但是 `scipy.cluster.hierarchy.linkage` 支持均值点距离, 代号为 `'centroid'`。

### 平均距离

如图 14 (d) 所示, **平均距离** (average linkage) 采用两簇样本数据成对点之间距离取平均值:

$$d(C_k, C_j) = \text{mean}_{x \in C_k, z \in C_j} (\text{dist}(x, z)) = \frac{\sum_{x \in C_k, z \in C_j} \text{dist}(x, z)}{\text{count}(C_k) \cdot \text{count}(C_j)} \quad (4)$$

平均距离代号为 `'average'`。图 17 所示为, 采用 `'average'` 层次聚类得到的树形图和鸢尾花数据聚类结果。

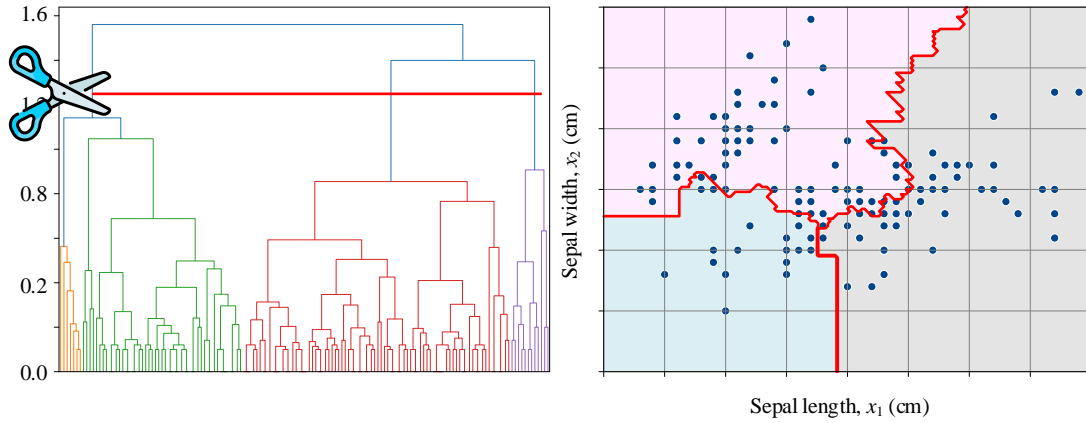


图 17. 鸢尾花聚类结果, 'average' 层次聚类

### 平均距离

Ward's 簇间距离的定义如下:

$$\begin{aligned} d(C_k, C_j) &= \sqrt{2 \times \left( \underbrace{\sum_{x \in C_k \cup C_j} \text{dist}(x, \mu_{C_k \cup C_j})^2}_{\text{After merge}} - \underbrace{\left( \sum_{x \in C_k} \text{dist}(x, \mu_{C_k})^2 + \sum_{x \in C_j} \text{dist}(x, \mu_{C_j})^2 \right)}_{\text{Before merge}} \right)} \\ &= \sqrt{\frac{2 \cdot \text{count}(C_k) \cdot \text{count}(C_j)}{\text{count}(C_k) + \text{count}(C_j)}} \cdot \text{dist}(\mu_{C_k}, \mu_{C_j}) \end{aligned} \quad (5)$$

观察 (5), 可以发现它等价于:

$$d(C_k, C_j) = \sqrt{2 \times \left( \underbrace{\text{SST}(C_k \cup C_j)}_{\text{After merge}} - \underbrace{(\text{SST}(C_k) + \text{SST}(C_j))}_{\text{Before merge}} \right)} \quad (6)$$

其中，SSE 为丛书前文介绍的**总离差平方和** (Sum of Squares for Total, SST)。

SSE 便是本书前文介绍的“**簇惯性** (cluster inertia)”，也就是：

$$\left\{ \begin{aligned} \text{SST}(C_k \cup C_j) &= \sum_{x \in C_k \cup C_j} \text{dist}(x, \mu_{C_k \cup C_j})^2 = \sum_{x \in C_k \cup C_j} \|x - \mu_{C_k \cup C_j}\|^2 \\ \text{SST}(C_j) &= \sum_{x \in C_j} \text{dist}(x, \mu_{C_j})^2 = \sum_{x \in C_j} \|x - \mu_{C_j}\|^2 \\ \text{SST}(C_k) &= \sum_{x \in C_k} \text{dist}(x, \mu_{C_k})^2 = \sum_{x \in C_k} \|x - \mu_{C_k}\|^2 \end{aligned} \right. \quad (7)$$

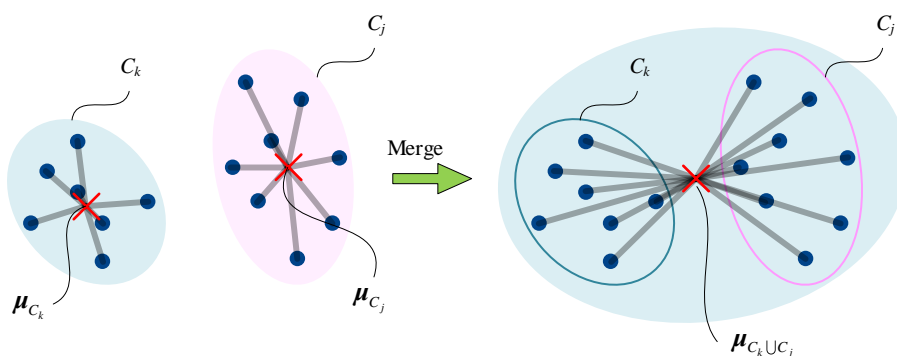


图 18. 鸢尾花聚类结果，Ward's 簇间距离

Ward's 簇间距离定义看着复杂，实际上背后的思想很简单——计算**合并后** (after merge)、**合并前** (before merge) 残差平方和 SSE 的差值。原理如图 18 所示。这个差值，也就是一种簇数据“合并”的“代价”。

平均距离代号为 **'ward'**。 **'ward'** 为 Scikit-learn 默认簇间距离。图 19 所示为，采用 **'ward'** 层次聚类得到的树形图和鸢尾花数据聚类结果。

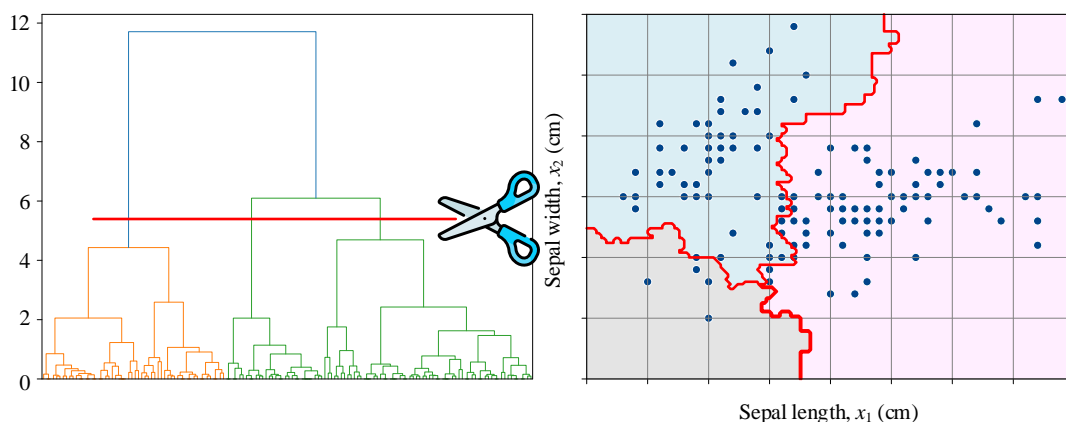


图 19. 鸢尾花聚类结果，'ward' 层次聚类



代码 Bk7\_Ch14\_01.py 可以绘制图 15、图 16、图 17 和图 19。

## 15.4 亲密度层次聚类

本章前文介绍的是采用欧氏距离构造树形图，以便进行层次聚类；其实，亲密度也可以用来构造树形图，从而聚类。回顾高斯核 (Gaussian kernel) 亲密度定义：

$$\kappa(\mathbf{x}, \mathbf{q}) = \exp(-\gamma \|\mathbf{x} - \mathbf{q}\|^2) \quad (8)$$

图 20 左图是鸢尾花数据高斯核亲密度成对矩阵热图。利用 `seaborn.clustermap()` 函数可以绘制基于亲密度矩阵的树形图，以及相应热图，具体如图 20 右图所示。

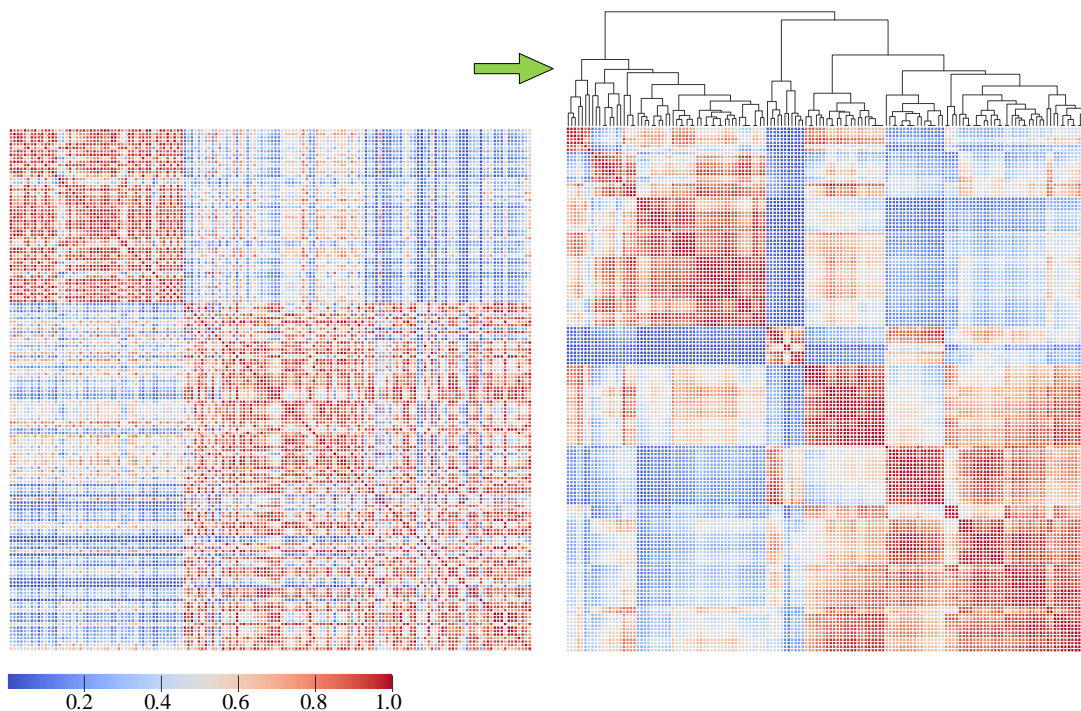


图 20. 鸢尾花花萼两个特征构造的亲密度矩阵，以及树形结构和重排后的亲密度矩阵



代码 Bk7\_Ch14\_02.py 可以绘制图 20 两幅子图。