报告

0. 概述

根据事先资料收集的结果，可以发现层次聚类是一种基于距离度量的聚类方法，其基本思想是将所有样本点自底向上地合并为若干个聚类，或自顶向下地将若干个聚类分解为若干个子聚类，以达到预设的聚类数目或满足某种终止条件。

层次聚类可以分为两大类：

- **凝聚层次聚类 (agglomerative)**：自底向上，从小到大合并
- **分裂层次聚类 (divisive)**：自顶向下，从大到小分裂

凝聚层次聚类

凝聚层次聚类算法的基本步骤如下：

1. 初始化：将每个样本点视为一个单独的聚类。
2. 计算所有聚类对之间的距离。
3. 合并距离最小的两个聚类。
4. 重复步骤 2 和 3，直到达到预设的聚类数目或满足某种终止条件。

分裂层次聚类

分裂层次聚类算法的基本步骤如下：

1. 初始化：将所有样本点合并为一个聚类。
2. 选择一个聚类进行分裂。
3. 将选定的聚类分裂成两个子聚类。
4. 重复步骤 2 和 3，直到达到预设的聚类数目或满足某种终止条件。

相同点

- 都是基于距离度量的聚类方法
- 都会生成一个层次聚类树
- 都可以用于不同类型的数据

不同点

- 合并顺序不同：凝聚层次聚类从小到大合并，分裂层次聚类从大到小分裂
- 计算复杂度不同：凝聚层次聚类的计算复杂度通常低于分裂层次聚类
- 适用场景不同：凝聚层次聚类适用于数据点之间距离较小的场景，分裂层次聚类适用于数据点之间距离较大的场景

本次实验采用实现较简单，并且更符合一般思维的凝聚层次聚类。

1. 距离计算准则

在凝聚层次聚类中，有几种不同的链接准则，它们定义了如何计算两个聚类之间的距离：

单链接 (Single Linkage)

- 两个聚类之间的距离定义为这两个聚类中**最近的两个点之间的距离**。
- 优点：计算简单，易于理解。
- 缺点：容易受到噪声和异常值的影响，倾向于产生长链状的聚类，而不是球状的聚类。
- 适用场景：数据非球状，存在噪声或异常值。

完全链接 (Complete Linkage)

- 两个聚类之间的距离定义为这两个聚类中**最远的两个点之间的距离**。
- 优点：可以产生紧凑的球状聚类。
- 缺点：计算复杂度较高，对异常值敏感。
- 适用场景：数据球状，没有噪声或异常值。

平均链接 (Average Linkage)

- 两个聚类之间的距离定义为这两个聚类中**所有点对之间的距离的平均值**。
- 优点：兼顾了单链接和完全链接的优点，对噪声和异常值有一定的鲁棒性。
- 缺点：计算复杂度较高。
- 适用场景：数据分布均匀，没有明显异常值。

质心链接 (Centroid Linkage)

- 两个聚类之间的距离定义为这两个聚类的**质心之间的距离**。
- 优点：计算简单，易于理解。
- 缺点：对数据分布的形状敏感，容易受到噪声和异常值的影响。
- 适用场景：数据分布均匀，没有明显异常值。

Ward 链接 (Ward's Linkage)

- 两个聚类之间的距离定义为**合并这两个聚类后总体方差增加的量**。
- 优点：可以产生大小相等的聚类，对数据分布的形状不敏感。
- 缺点：计算复杂度较高。
- 适用场景：数据分布不均匀，需要聚类成大小相等的簇。

总结:

- 五种连接准则各有特点，选择哪种连接准则取决于具体的应用场景和数据特性。
- 在实际应用中，可以尝试不同的连接准则，并根据聚类结果进行选择。

连接准则	距离定义	优点	缺点	适用场景
单链接	最近点距离	计算简单	容易受噪声影响，易形成长链	非球状数据，存在噪声
完全链接	最远点距离	形成紧凑聚类	计算复杂，对异常值敏感	球状数据，无噪声
平均链接	点对平均距离	兼顾单/完全链接	计算复杂	数据分布均匀，无明显异常
质心距离	质心间距离	计算简单	对数据形状敏感	数据分布均匀，无明显异常
Ward 链接	方差增加量	形成大小相等聚类	计算复杂	数据分布不均匀

> **在凝聚层次聚类中，除了欧氏距离，还可以使用以下几种距离度量方法：**

马氏距离 (Mahalanobis Distance)

- 由印度统计学家马哈拉诺比斯提出的，表示数据的协方差距离。
- 适用于**具有不同尺度**的簇，例如不同年龄段的人的身高。
- 缺点是：需要计算协方差矩阵，计算复杂度较高。

余弦相似度 (Cosine Similarity)

- 通过测量两个向量的夹角的余弦值来度量它们之间的相似性。
- 适用于**具有方向性**的场景，例如文本语义分析。
- 缺点是：对数据量敏感，容易受到数据规模的影响。

欧氏距离 (Euclidean Distance)：

- 衡量多维空间中两点之间的真实距离。
- 适用于数据具有相似尺度的情况。

- 优点：直观易懂，计算简单。
- 缺点：对噪声敏感，不适用于数据具有不同尺度的情况。

距离度量方法	适用场景	优点	缺点
欧氏距离	数据具有相似尺度	直观易懂，计算简单	对噪声敏感，不适用于数据具有不同尺度的情况
马氏距离	数据具有不同尺度	适用于不同尺度的特征	计算复杂度较高
余弦相似度	具有方向性的场景	适用于文本语义分析等场景	对数据量敏感

2. 实验设置

2.1 实验目标

本实验旨在对比凝聚层次聚类算法中三种常用距离计算准则（SINGLE、COMPLETE、WARD）的性能和特点。

2.2 实验内容

1. 选择一个典型数据集：Iris（鸢尾花数据集）
2. 对这个数据集的四个特征，使用单链接、完全链接和Ward 链接三种距离计算准则进行凝聚层次聚类。
3. 对比聚类结果的优劣，观察的是三种计算准则的特点，实验说明不同距离计算准则的倾向、优缺点和适用场景等。

2.3 实验方法

本实验采用多种指标来评估聚类算法的性能，包括内部指标和外部指标。

内部指标

内部指标用于评估聚类结果的紧凑性和分离度，**无需**真实标签。

1. 轮廓系数 (Silhouette Coefficient)

轮廓系数衡量每个样本与其所属聚类的相似度与与其他聚类的相似度之间的差距。取值范围为[-1, 1]：

- 值越大，表示样本与其所属聚类越相似，与其他聚类越不相似，聚类效果越好。
- 值为0，表示样本与所属聚类和与其他聚类一样相似，聚类效果一般。
- 值越小，表示样本与其所属聚类越不相似，与其他聚类越相似，聚类效果越差。

2. Davies-Bouldin Index (DBI)

DBI 指数衡量聚类簇之间的平均分离度和簇内部的平均紧凑度。取值越小，表示聚类效果越好。

外部指标

外部指标用于评估聚类结果与真实标签的一致性，**需要**真实标签。

1. 调整兰德指数 (Adjusted Rand Index, ARI)

ARI 指数衡量聚类结果与真实标签之间的相似度。取值范围为[-1, 1]：

- 值为1，表示聚类结果与真实标签完全一致。
- 值为0，表示聚类结果与随机分配一样。
- 值为-1，表示聚类结果与真实标签完全相反。

2. 互信息 (Mutual Information, MI)

MI 指数衡量聚类结果与真实标签之间的相互信息量。取值越大，表示聚类结果与真实标签的一致性越高。

运行效率

为了评估不同聚类算法的运行效率，本实验记录了它们在同一台机器上的运行时间。

参数设置

- 聚类数：根据数据集的实际情况设置，Iris数据集的情况为3

- 距离计算方法：单链接、完全链接、Ward 链接

2.4 实验算法设计

单链接准则

算法过程

1. 计算距离矩阵：计算所有数据点之间的成对距离，并存储在一个距离矩阵中。
2. 初始化：初始化一个聚类树，每个数据点作为树中的一个叶子节点。
3. 迭代合并：
   - 找到距离矩阵中最小距离所对应的两个数据点。
   - 将这两个数据点合并为一个新的簇，并将其加入到聚类树中。
   - 更新距离矩阵，删除与已合并数据点相关的行和列。
4. 循环终止：当距离矩阵中只剩下一个元素时，迭代终止。

实现技术

1. **距离计算**：计算所有数据点对之间的欧几里得距离，形成一个距离数组。
2. **压缩索引**：将完整矩阵的索引转换为压缩矩阵的索引，这在存储上三角矩阵时非常有用，可以压缩存储空间。
3. **并查集：**并查集数据结构用于快速合并集合并查找元素的根节点，这在聚类时标记和合并集合非常有效。
4. **最小生成树**：使用了类似于最小生成树（MST）的算法来实现最小链接法则。它迭代地找到距离最小的未合并节点对，并将它们合并到同一个聚类中。
5. **排序和标记**：对聚类结果进行排序和正确标记，确保聚类的顺序与它们在树状图中的位置相对应。

时空复杂度

1. 计算初始两辆距离矩阵时
   - 时间复杂度：该函数的时间复杂度取决于两个嵌套的循环，时间复杂度为O(n^2)。
   - 空间复杂度：创建了一个长度为n^2/2的上三角矩阵的压缩数组，因此空间复杂度为O(n^2)。
2. 并查集：
   - 初始化时的时间复杂度为O(n)，其中n是样本数量。它创建了一个长度为2n-1的父节点数组和一个长度为2n-1的大小数组，因此空间复杂度为O(n)。
   - 集合合并方法的时间复杂度为O(1)。
   - 查找根节点方法的时间复杂度取决于树的高度。但由于并查集按秩合并和路径压缩优化，因此时间复杂度为O(1)。
3. 最小生成树迭代时：
   - 时间复杂度：该函数的时间复杂度取决于两个嵌套的循环。外部循环迭代(n-1)次，内部循环在每次迭代中需要线性地搜索剩余的未合并节点，因此迭代次数为(n-1) + (n-2) + ... + 1，即(n-1)*n/2次。在内部循环中，计算两个样本之间的距离需要O(1)的时间复杂度。因此，总体时间复杂度为O(n^2)。
   - 空间复杂度：创建了一个长度为(n-1)的链接矩阵，因此空间复杂度为O(n)。

综上所述，时间复杂度为O(n^2)，空间复杂度为O(n^2)

完全连接计算法则

算法过程

相比于单链接，在一次合并后还要更新距离矩阵（每个簇中点的最大距离作为簇的新距离）然后再继续迭代。

实现技术

距离计算、压缩索引、并查集、排序与标记都与单链接法则相同
1.
链：用于存储相邻簇的序列，以找到最近的邻居。

具体来说，链的作用如下：

- **初始化链**：在算法开始时，链是空的。首先找到一个活跃的簇（即还没有被合并的簇）作为链的起点。
- **构建链**：然后，算法通过比较距离，找到与链尾部簇最近的簇，并将其添加到链的末尾。

- **检测循环**：如果链中的最后两个簇互为最近邻居，那么它们就会被合并，这个过程会从链中移除这两个簇。
- **重复过程**：这个过程会一直重复，直到所有的簇都被合并成一个单一的簇。

时空复杂度

空间复杂度与单链接类似。

时间复杂度上，主循环迭代运行了 (n - 1) 次，因为每次迭代都会合并两个簇，直到只剩下一个簇。在主循环内部，存在一个循环，它通过链来寻找最近的邻居。在最坏的情况下，这个循环可能会遍历所有的簇，但由于链的特性，实际上它通常会更快地找到最近的邻居。更新距离矩阵的在每次合并后都会执行，这也是 (O(n)) 的操作。

因此整个算法的时间复杂度接近于 O(n^2) ，空间复杂度为O(n^2)

Ward 链接计算法则

算法过程

与完全连接计算基本一致，只是簇与簇的距离被定义为了方差。

实现技术

与完全连接计算基本一致

时空复杂度

与完全连接计算基本一致

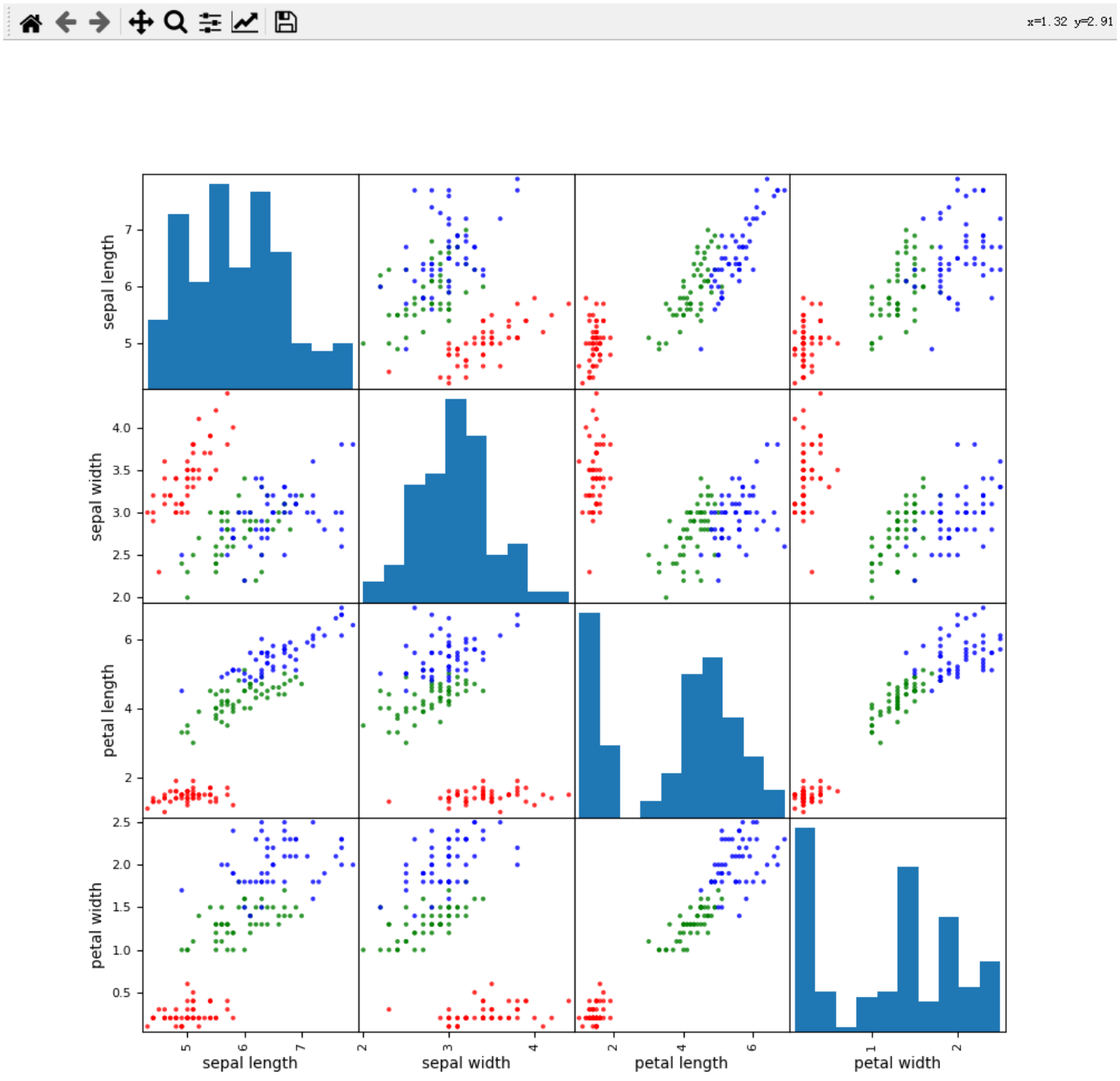
因此整个算法的时间复杂度接近于 O(n^2) ，空间复杂度为O(n^2)

2.5 实验意义

- 比较不同距离计算准则的聚类效果：通过使用单链接、完全链接和Ward链接三种距离计算准则，我们可以观察它们对于Iris数据集的聚类结果的优劣。这有助于我们了解不同准则对聚类结果的影响，并选择最适合特定数据集的准则。
- 理解不同距离计算准则的特点：每种距离计算准则都有其独特的特点和计算方式。通过比较它们的聚类结果，我们可以深入了解它们之间的差异。例如，单链接倾向于形成长而不稳定的聚类链，完全链接倾向于形成紧凑的球状聚类，而Ward链接倾向于形成方差较小的聚类。
- 评估聚类算法的性能：实验中采用了多种内部指标和外部指标来评估聚类算法的性能。通过这些指标，我们可以客观地评价不同距离计算准则的聚类结果。例如，轮廓系数和DBI指数可以评估聚类结果的紧凑性和分离度，而ARI和MI指数可以评估聚类结果与真实标签的一致性。
- 运行效率比较：实验还记录了不同聚类算法在同一台机器上的运行时间。这样可以比较它们的运行效率，从而在实际应用中选择合适的算法。

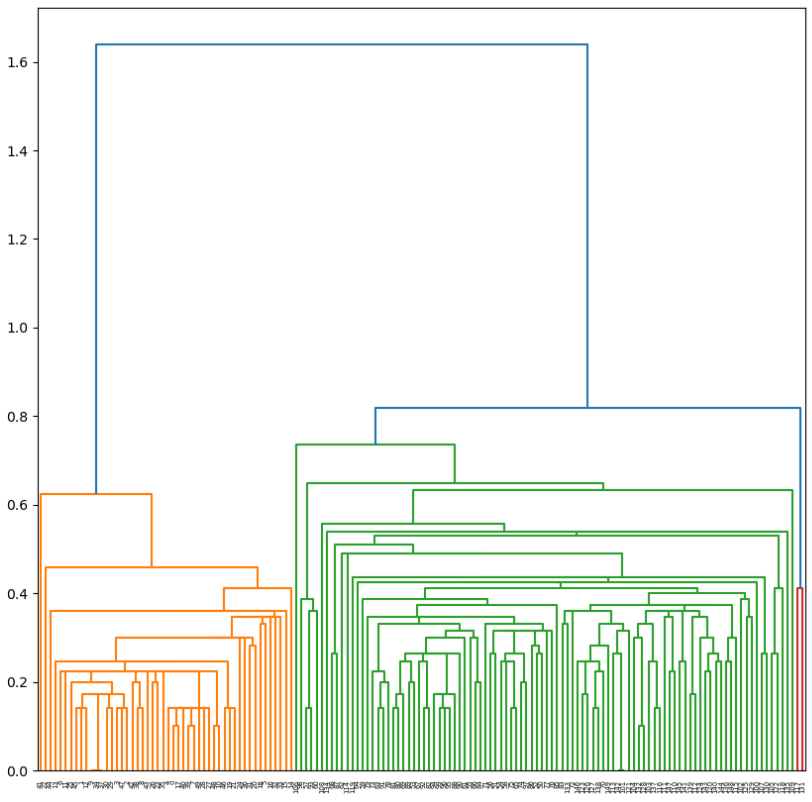
3. 实验结果

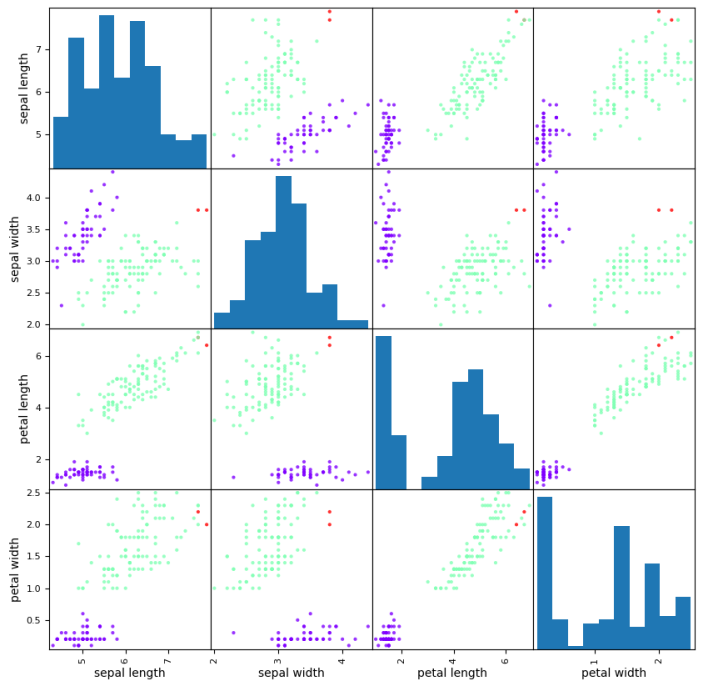
首先观察源数据集的聚类分布特征。



单链接计算准则

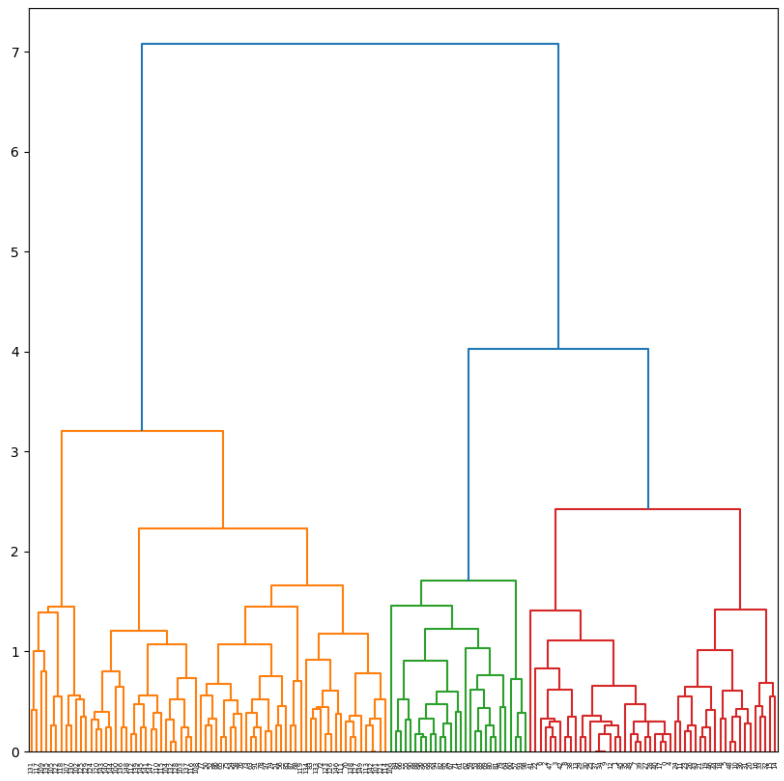
```
Elapsed time: 0.08774948120117188 seconds
Silhouette Coefficient: 0.5118387098922373
Davies-Bouldin Index: 0.44743843419896284
Adjusted Rand Index: 0.5637510205230709
Adjusted Mutual Information: 0.7125764811325073
```

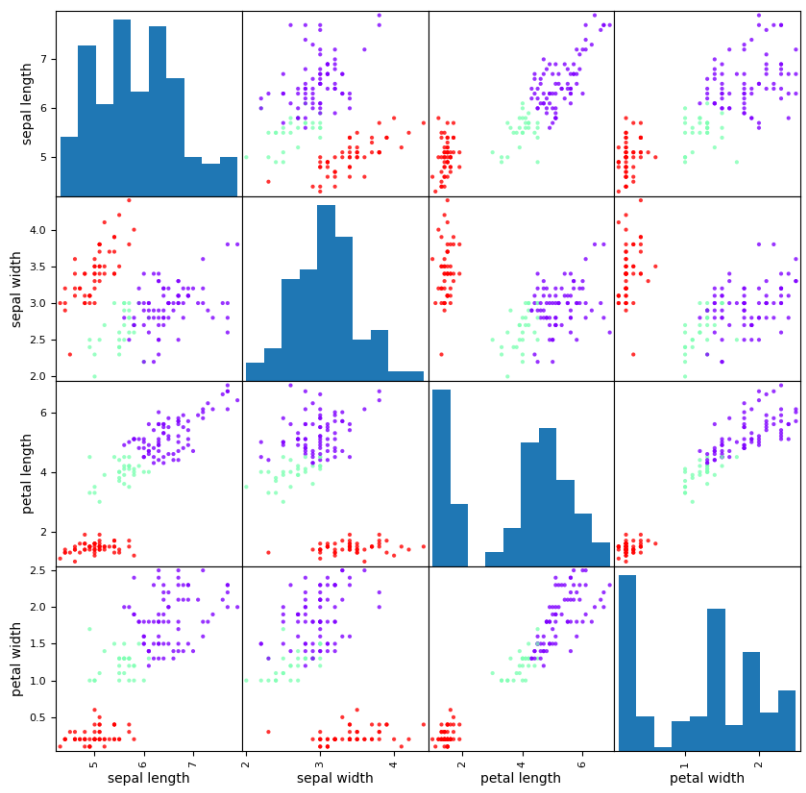




完全链接计算准则

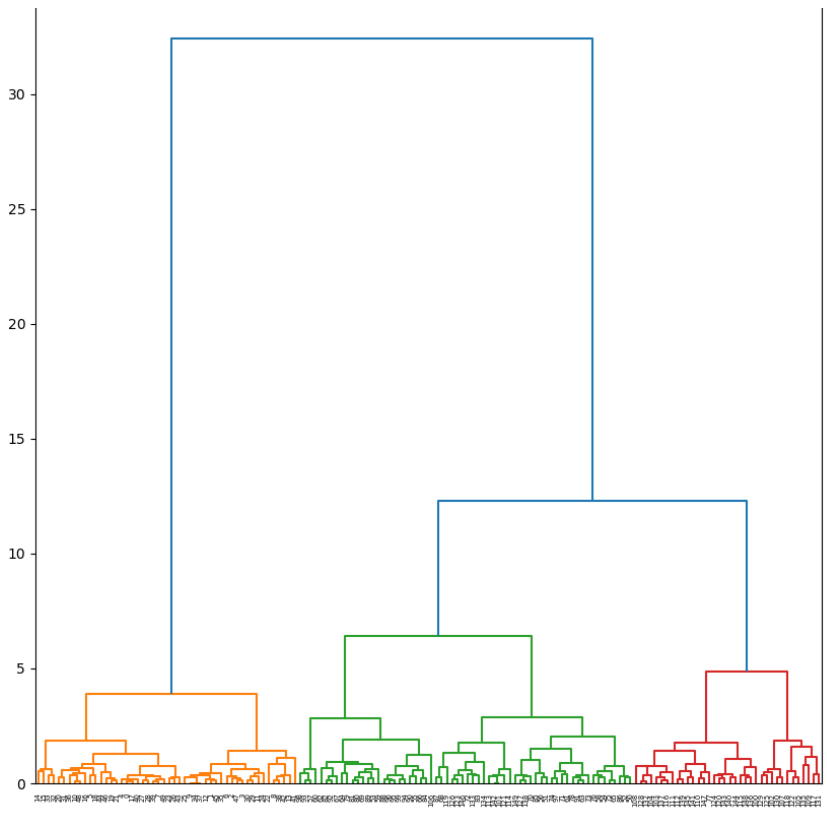
```
Elapsed time: 0.1453062675476074 seconds
Silhouette Coefficient: 0.5133502348448076
Davies-Bouldin Index: 0.6337780002446376
Adjusted Rand Index: 0.6422512518362898
Adjusted Mutual Information: 0.7184641371994781
```

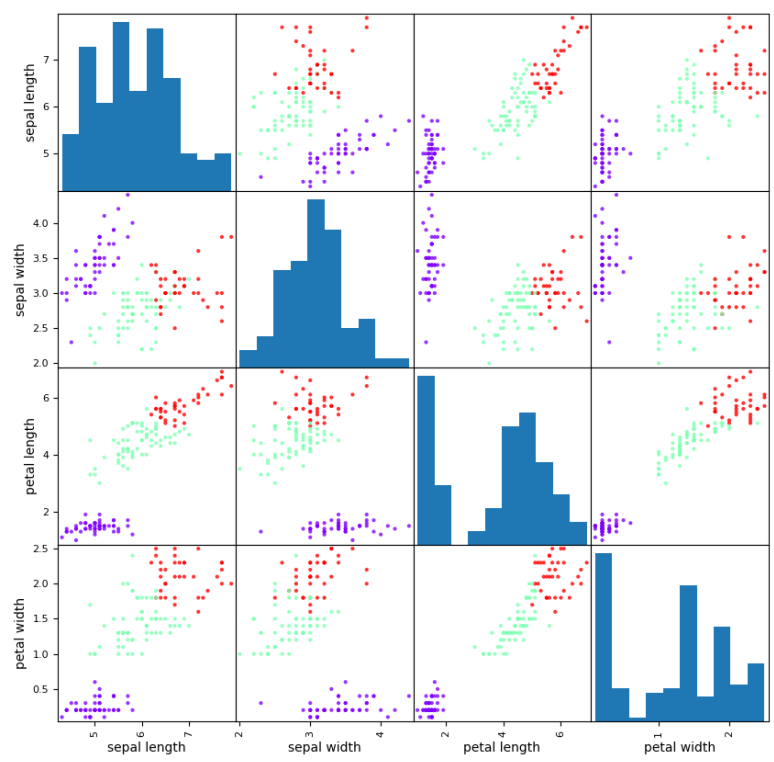




WARD链接计算准则

```
Elapsed time: 0.18940186500549316 seconds
Silhouette Coefficient: 0.5540972908150551
Davies-Bouldin Index: 0.6566044124178404
Adjusted Rand Index: 0.7311985567707746
Adjusted Mutual Information: 0.7671669615713111
```





4. 实验分析

指标	SINGLE	COMPLELE	WARD
轮廓系数 (Silhouette Coefficient)	0.5118	0.5133	0.5541
戴维斯-鲍兰德指数 (Davies-Bouldin Index)	0.4474	0.6338	0.6566

指标	SINGLE	COMPLELE	WARD
调整兰德指数 (Adjusted Rand Index)	0.5638	0.6423	0.7312
调整互信息 (Adjusted Mutual Information)	0.7126	0.7185	0.7672
运行时间 (Elapsed time)	0.0952 秒	0.1453 秒	0.1894 秒

1. **不同距离计算准则的聚类效果比较**:
   - **轮廓系数 (Silhouette Coefficient)**: WARD 方法的轮廓系数最高（**0.5541**），表明其聚类质量最佳，具有较好的内聚度和分离度。
   - **戴维斯-鲍兰德指数 (Davies-Bouldin Index)**: SINGLE 方法的DBI最低（**0.4474**），意味着其聚类的分离度较好，但考虑到其他指标，SINGLE 方法的整体聚类效果可能不如WARD。
   - **调整兰德指数 (Adjusted Rand Index)** 和 **调整互信息 (Adjusted Mutual Information)**: 这两个指标均以WARD方法得分最高，分别为 **0.7312** 和 **0.7672**，显示出与真实聚类结果的高度一致性。
2. **理解不同距离计算准则的特点**:
   - SINGLE 方法可能形成长链状的聚类，这在某些情况下可能导致聚类结果不稳定，**比如说在这个鸢尾花数据集上就对两个离群点特别敏感，导致把单独的两个点作为一个聚类**。
   - COMPLETE 方法倾向于形成紧凑的球状聚类，适合于数据集中存在明显分隔的情况。
   - WARD 方法通过最小化聚类内方差来形成更加均匀的聚类，适用于各类大小相似的情况。
3. **聚类算法的性能评估**:
   - 轮廓系数和DBI指数主要评估聚类的紧凑性和分离度，而ARI和AMI指数则评估聚类结果与真实标签的一致性。在这些指标上，WARD方法表现最佳，说明其聚类效果最接近真实情况。
4. **运行效率比较**:
   - 虽然SINGLE方法的运行时间最短（**0.0952 秒**），但从聚类质量的角度来看，WARD方法可能是更好的选择，尽管其运行时间稍长（**0.1894 秒**），虽然三种方法的时间复杂度一致，但由于WARD计算方差要更多额外的计算，导致时间偏长，而SINGLE只需要计算欧氏距离，耗时较短。

综上所述，WARD方法在多个重要指标上均表现最佳，尽管其运行时间略长。这表明在选择聚类方法时，应综合考虑聚类质量和运行效率，以便在特定应用场景中做出最佳选择。

5. 实验小结

在本次实验中，我选择了凝聚层次聚类（agglomerative hierarchical clustering）算法来对鸢尾花数据集进行分析。我实现了单链接（SINGLE）、完全链接（COMPLETE）和Ward链接（WARD）三种不同的距离计算准则，并对比了它们在聚类效果上的差异。

算法实现: 我采用了凝聚层次聚类算法，该算法从每个数据点作为单独的簇开始，逐步合并距离最近的簇，直到达到预定的簇数量。在实现过程中，我特别注意了算法的时序复杂度，通过优化数据结构和合并步骤来提高效率。

实验设置: 我使用了加州大学欧文分校机器学习资料库提供的鸢尾花数据集进行测试。该数据集包含150个样本，每个样本有4个特征。我分别应用了三种不同的距离计算准则，并记录了算法的运行时间。

距离计算讨论:

- 单链接（SINGLE）: 此准则下，簇间的距离由簇中最近的两个点决定。它倾向于产生长而薄的簇，可能导致链式效应。
- 完全链接（COMPLETE）: 此准则下，簇间的距离由簇中最远的两个点决定。它倾向于产生紧凑的簇，适合于球状或者均匀分布的数据。
- Ward链接（WARD）: 此准则通过最小化合并后簇内的方差增量来选择合并的簇。它倾向于产生大小相似的簇。

性能评估: 通过轮廓系数、戴维斯-鲍兰德指数、调整兰德指数和调整互信息等指标，我评估了聚类的质量。Ward链接在多数指标上表现最佳，但运行时间略长于其他两种准则。

运行时间: 实验记录了三种准则的运行时间，其中单链接准则的运行时间最短，但其聚类质量不如Ward链接。

结论: 综合考虑聚类质量和运行效率，Ward链接是对鸢尾花数据集进行层次聚类的最佳选择。然而，如果对运行时间有严格要求，单链接准则可能是一个更合适的选择。

未来工作: 未来可以探索更多的距离计算准则，并在更大的数据集上测试算法的可扩展性和效率。此外，可以考虑实现分裂层次聚类算法，并与凝聚层次聚类算法进行比较。