南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

机器学习实验报告

Lab5 聚类分析

马溟怡

学号：2311061

专业：计算机科学与技术

2025 年 12 月 4 日

目录

一、 实验原理

(一) 层次聚类的基本原理

层次聚类是一种无监督学习方法，用于将数据集中的样本点组织成一个树状结构。它不要求事先指定聚类簇的数量 k，而是通过迭代地合并或分裂簇来形成层次结构。

本次实验采用**凝聚**方法，其基本步骤如下：

1. **初始化：**将每个样本点视为一个独立的簇。
2. **迭代合并：**重复以下步骤，直到所有样本点合并成一个大簇（或达到预设的簇数 k）：
   - 计算所有当前簇对之间的距离。
   - 将距离最近的两个簇合并成一个新的簇。
3. **最终划分：**根据用户设定的目标簇数 k 或在谱系图上“剪枝”来获得最终的聚类结果。

(二) 簇间距离度量

层次聚类算法的核心在于定义两个簇 C_i 和 C_j 之间的距离 $d(C_i, C_j)$。本实验实现了三种主要的距离度量方式：

1. Single-linkage（单链接 / 最小距离）

- **定义：**两个簇之间的距离定义为**两个簇中最接近的样本点**之间的距离。
- **公式：**
$$d(C_i, C_j) = \min\{d(a, b) \mid a \in C_i, b \in C_j\}$$
- **特点：**倾向于产生长链状的聚类。它对噪声和离群点高度敏感，容易产生链式效应。

2. Complete-linkage（全链接 / 最大距离）

- **定义：**两个簇之间的距离定义为**两个簇中最远的样本点**之间的距离。
- **公式：**
$$d(C_i, C_j) = \max\{d(a, b) \mid a \in C_i, b \in C_j\}$$
- **特点：**倾向于产生紧凑的球形聚类。它对离群点具有较好的鲁棒性。

3. Average-linkage（平均链接 / 平均距离）

- **定义：**两个簇之间的距离定义为**两个簇中所有样本点对之间距离的平均值**。
- **公式：**
$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{a \in C_i} \sum_{b \in C_j} d(a, b)$$
- **特点：**该方法综合了 Single-linkage 和 Complete-linkage 的特点，通常比前两者更稳健。

二、 实验内容

(一) 基础任务：Single-linkage 与 Complete-linkage 聚类

1. 代码

基础任务

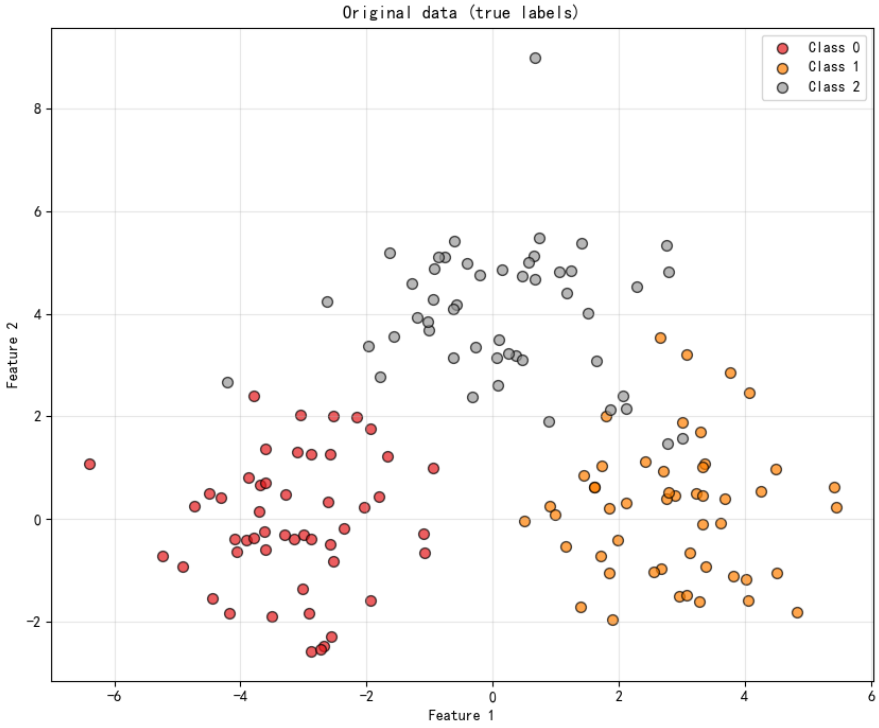
```
    def _single_linkage_distance(self, cluster1, cluster2, dist_matrix):
        """
        Single-linkage: 两个簇之间的【最小】距离

        公式: d(C1, C2) = min{d(a,b) | a C1, b C2}

        输入:
            cluster1: 第一个簇的样本索引列表, 如 [0, 3, 5]
            cluster2: 第二个簇的样本索引列表, 如 [1, 2]
            dist_matrix: 距离矩阵, shape=(n_samples, n_samples)

        输出:
            两个簇之间的最小距离

        TODO: 请完成single-linkage距离计算
        提示:
        1. 遍历cluster1中的每个样本i
        2. 遍历cluster2中的每个样本j
        3. 使用dist_matrix[i, j]获取i和j的距离
        4. 返回所有距离中的最小值
        """
        min_dist = np.inf

        # TODO: 遍历两个簇中的所有样本对, 找最小距离
        for i in cluster1:
            for j in cluster2:
                # TODO: 比较并更新最小距离
                current_dist = dist_matrix[i, j]
                if current_dist < min_dist:
                    min_dist = current_dist

        return min_dist

    def _complete_linkage_distance(self, cluster1, cluster2, dist_matrix):
        """
        Complete-linkage: 两个簇之间的【最大】距离

        公式: d(C1, C2) = max{d(a,b) | a C1, b C2}

        输入:
            cluster1: 第一个簇的样本索引列表
```

```
        cluster2: 第二个簇的样本索引列表
        dist_matrix: 距离矩阵

    输出:
        两个簇之间的最大距离
    TODO: 请完成complete-linkage距离计算
    提示: 与single-linkage类似, 但找最大值
    """
    max_dist = 0

    # TODO: 遍历两个簇中的所有样本对, 找最大距离
    for i in cluster1:
        for j in cluster2:
            # TODO: 比较并更新最大距离
            current_dist = dist_matrix[i, j]
            if current_dist > max_dist:
                max_dist = current_dist

    return max_dist
```

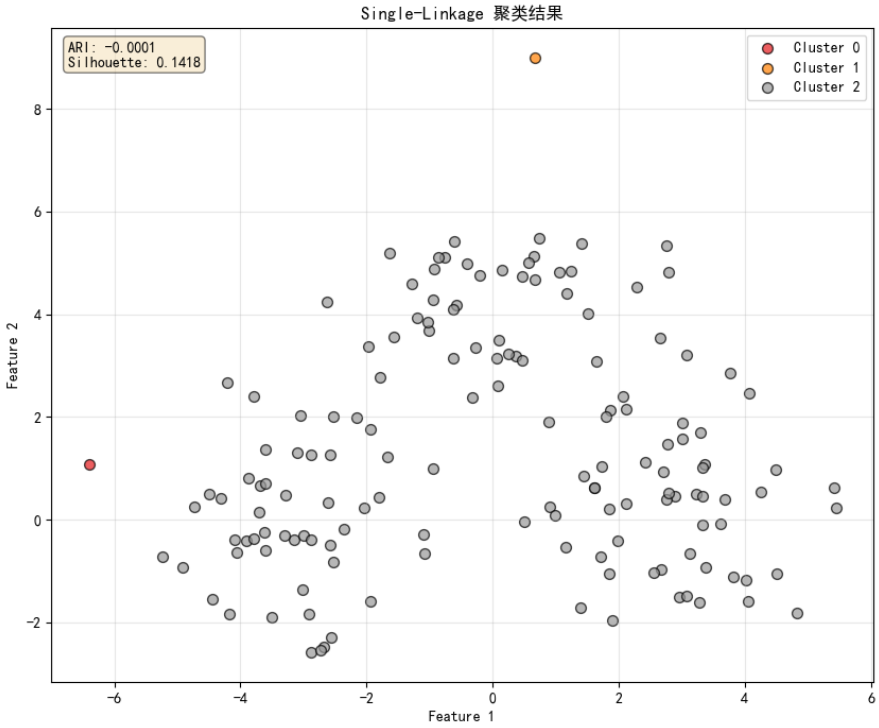
2. 控制台输出结果

```
[步骤2] 基本要求: Single-linkage 和 Complete-linkage

Single-linkage聚类...
  ARI: -0.0001
图片已保存: out_exp5\single_linkage.png
Complete-linkage聚类...
  ARI: 0.7481
图片已保存: out_exp5\complete_linkage.png
```

3. 结果分析

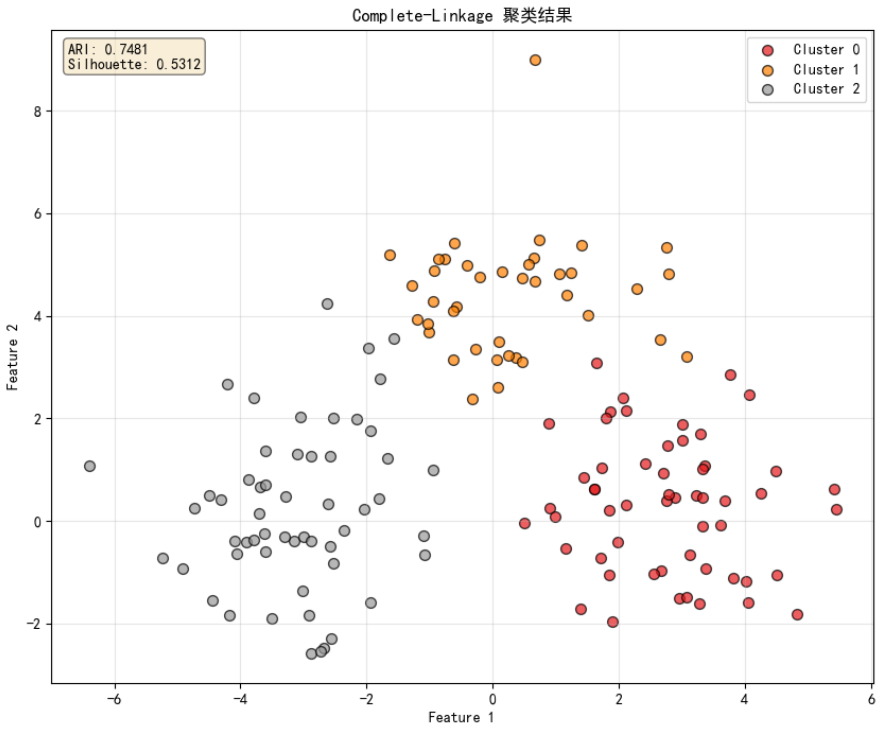


这是原始数据分布。可以看到数据集中有三个大致呈球形的簇（红色 Class 0、橙色 Class 1、灰色 Class 2），且簇之间有一定间隔。



ARI: −0.0001，Silhouette: 0.1418

可以证明这是个失败的聚类。图中显示大部分样本（灰色 Cluster 2）被合并成了一个巨大的、链状的簇，而另外两个簇（红色 Cluster 0 和橙色 Cluster 1）各只有一两个样本。这说明了 Single-linkage 的“链式效应”：由于它只关注簇间最近距离，只要两个点足够近，整个簇就会被连接，导致它无法正确分割出球形、密度均匀的簇。性能指标极差，证明它不适合这类数据。



ARI: 0.7481，Silhouette: 0.5312

可以证明这是个较好的聚类。Complete-linkage 倾向于产生紧凑的球形簇。图中它成功地将三个原始簇大致分离，划分的簇边界清晰。但与真实标签相比，它在簇的边界（例如红色和灰色簇的重叠区域）上有一些误分类，因此 ARI 略低于最优值。

(二) 中级任务：Average-linkage 聚类

1. 代码

中级任务

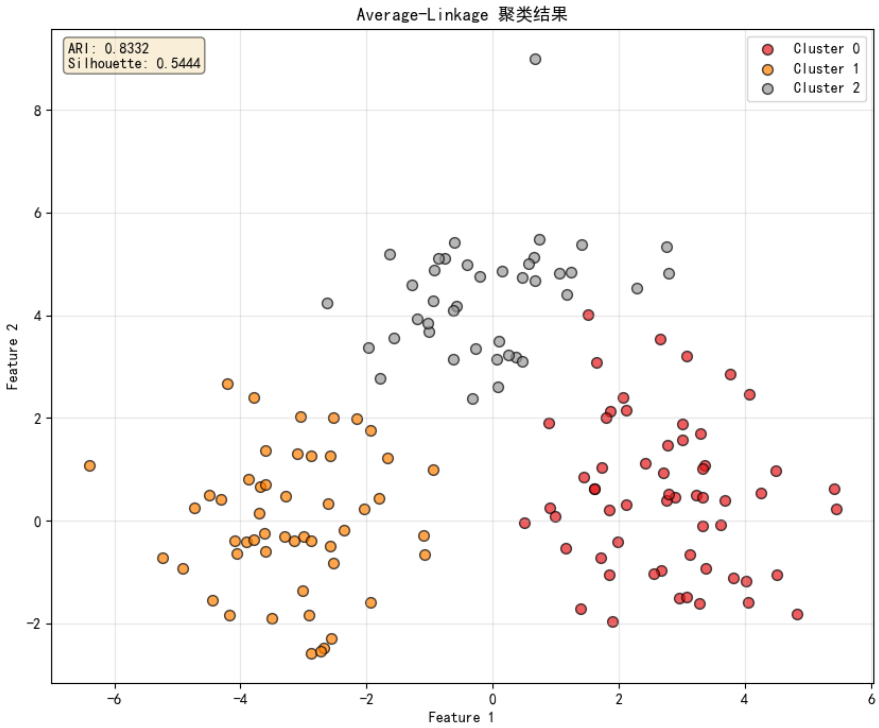
```
def _average_linkage_distance(self, cluster1, cluster2, dist_matrix):
    """
    Average-linkage: 两个簇之间的【平均】距离

    公式: d(C1, C2) = (1/(|C1|×|C2|)) × Σ Σ d(a,b)

    输入:
        cluster1: 第一个簇的样本索引列表
        cluster2: 第二个簇的样本索引列表
        dist_matrix: 距离矩阵

    输出:
        两个簇之间的平均距离
    TODO: 请完成average-linkage距离计算
    提示:
    1. 累加所有样本对的距离
    2. 除以样本对的数量 (|C1| × |C2|)
    """
```

```
        total_dist = 0
        count = 0

        # TODO: 遍历两个簇中的所有样本对，累加距离
        for i in cluster1:
            for j in cluster2:
                # TODO: 累加距离并计数
                total_dist += dist_matrix[i, j]
                count += 1

        # TODO: 返回平均距离
        if count == 0:
            return 0  # 避免除以零
        return total_dist / count
```

2. 控制台输出结果

```
[步骤3] 中级要求: Average-linkage
    ARI: 0.8332
  图片已保存: out_exp5\average_linkage.png
```

3. 结果分析



ARI: 0.8332，Silhouette: 0.5444

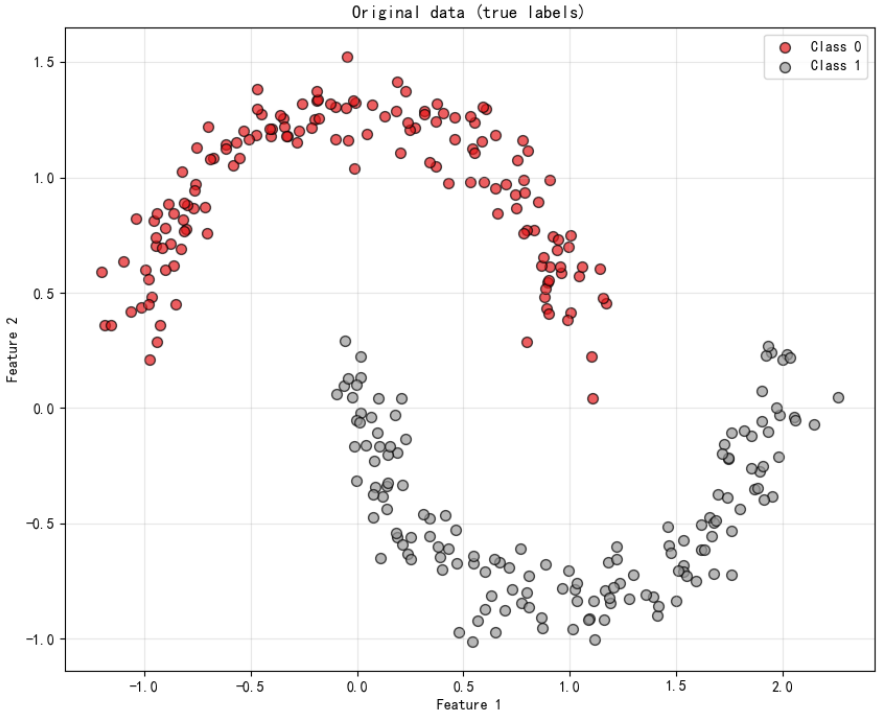
相比基础任务的两个聚类，这是最佳聚类。Average-linkage 表现最好，其 ARI 和 Silhouette Score 均是三种方法中最高的。这表明在所有样本对的平均距离基础上进行合并，能够提供最稳健、最准确的划分。图中簇的划分比 Complete-linkage 更接近真实边界。

(三) 提高任务：算法对比

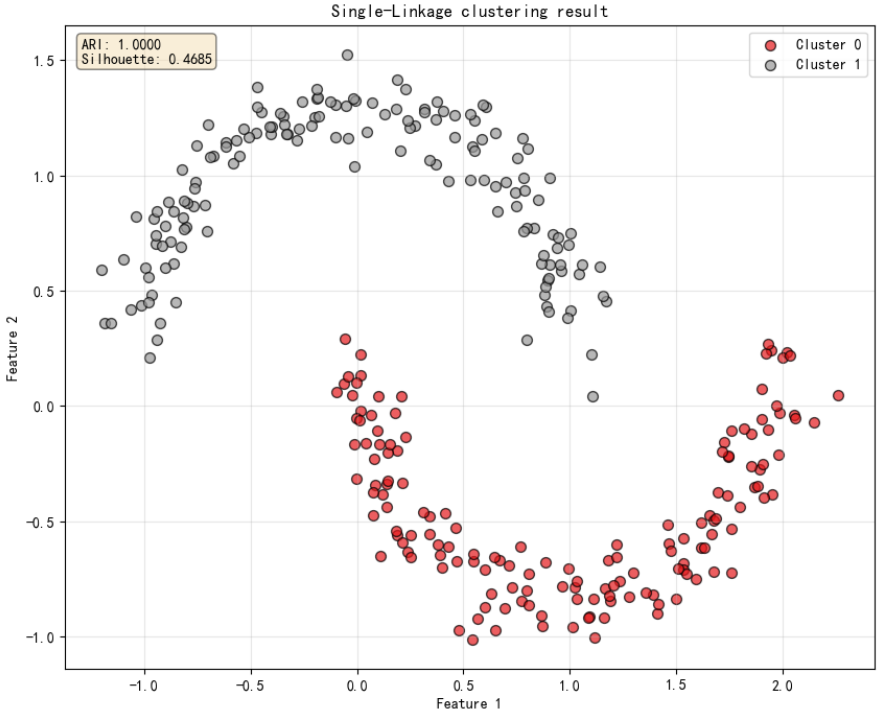
1. 控制台输出结果

```
[步骤4] 提高要求: 算法对比 (make_blobs 数据集)

[算法对比]
------------------------------------------------------------
linkage          ARI          Silhouette
------------------------------------------------------------
single           -0.0001      0.1418
complete         0.7481       0.5312
average          0.8332       0.5444
------------------------------------------------------------

结论 (make_blobs 数据集):
  - 在当前数据集上, average-linkage 表现最好
  - Single-linkage: 容易产生链式效应
  - Complete-linkage: 倾向于产生紧凑的簇
  - Average-linkage: 两者的折中

============================================================
[moons-2] 三种链接方式算法对比 (k=2, make_moons 数据集)

[算法对比]

------------------------------------------------------------

linkage          ARI          Silhouette

------------------------------------------------------------

single           1.0000       0.4685

  图片已保存: out_exp5_moons\clustering_single.png

complete         0.6823       0.4887

  图片已保存: out_exp5_moons\clustering_complete.png

average          0.5073       0.4760

  图片已保存: out_exp5_moons\clustering_average.png
```

2. 结果分析

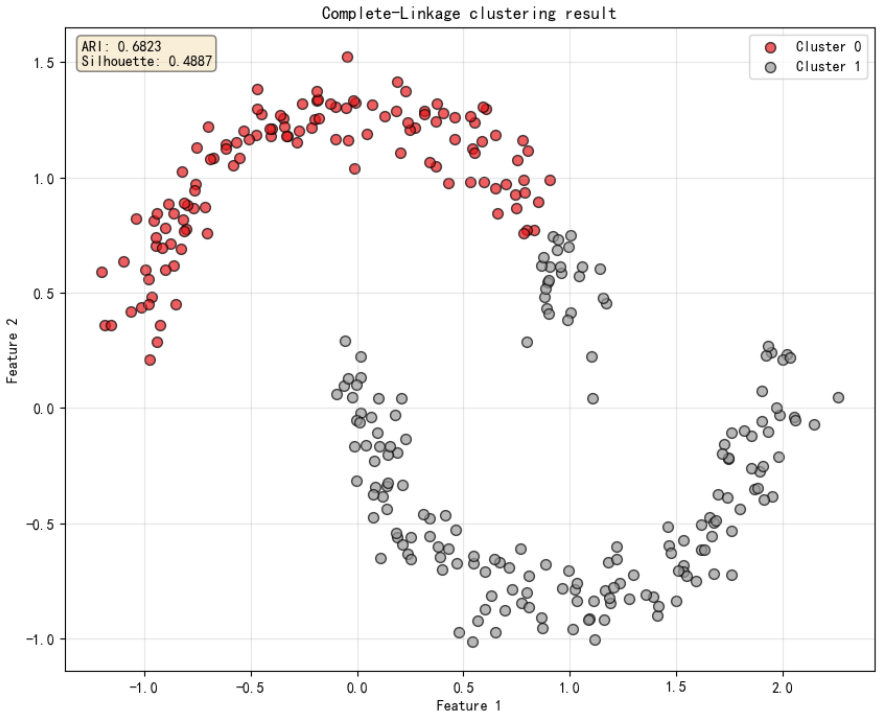


这是原始数据分布。可以看到数据集中包含两个稍微嵌入在一起的弯月形簇（红色 Class 0 和灰色 Class 1），这是典型的非凸形状数据。



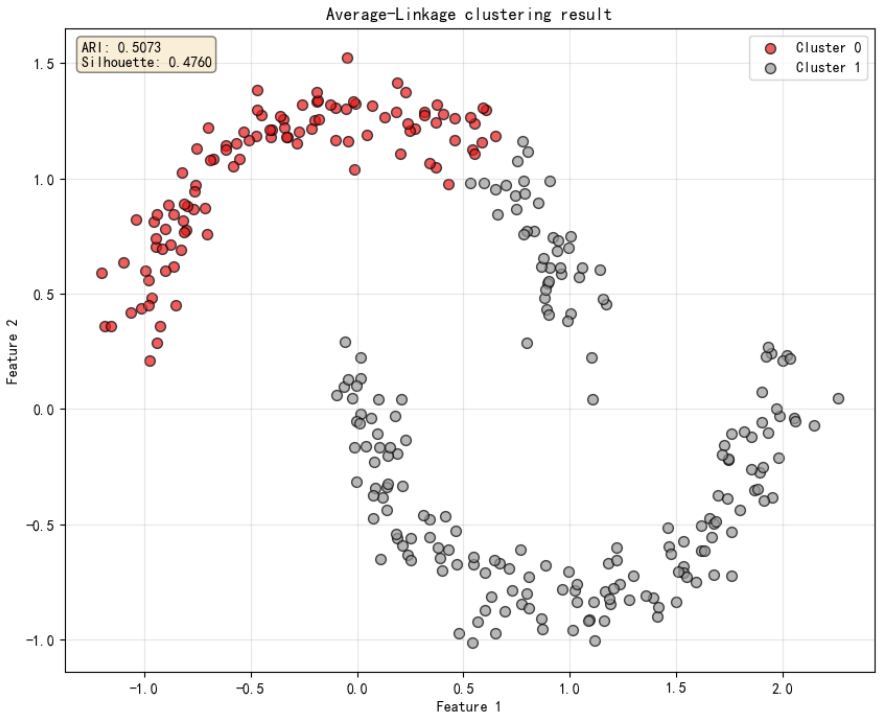
ARI: 1.0000，Silhouette: 0.4685

这是个满分的划分。Single-linkage 达到了 ARI = 1.0000，这说明聚类结果与真实标签完全匹配。对于非凸形状数据，Single-linkage 效果最佳，因为它只关注簇间最近距离，能够沿着弯月形的曲线进行合并，正确地将两个交织的簇分离。



ARI: 0.6823，Silhouette: 0.4887

这个划分表现一般。Complete-linkage 倾向于发现紧凑的球形簇，因此它在弯月形簇的中间部分进行了“垂直切断”，而不是沿着形状边界划分。结果是，两个簇都包含了来自原始两个弯月形簇的样本，准确度显著下降。



ARI: 0.5073，Silhouette: 0.4760 Average-linkage 的性能比 Complete-linkage 更差一些，结果接近随机划分（即 ARI 接近 0.5）。它处理非凸形状表现较差，错误地将靠近中心的样本混合，表明它更适合球形或凸形状簇。

（四） 拓展任务：变换聚类簇数 k

1. 控制台输出结果

```
[步骤5] 拓展要求: 测试不同聚类数k (make_blobs 数据集)

[不同聚类数k的性能测试]

k=2:

  single     : ARI=0.0000, Silhouette=0.3928

  complete   : ARI=0.5229, Silhouette=0.4568

  average    : ARI=0.5584, Silhouette=0.4593

k=3:

  single     : ARI=-0.0001, Silhouette=0.1418

  complete   : ARI=0.7481, Silhouette=0.5312

  average    : ARI=0.8332, Silhouette=0.5444

k=4:

  single     : ARI=0.0000, Silhouette=-0.0537

  complete   : ARI=0.6273, Silhouette=0.4166

  average    : ARI=0.8246, Silhouette=0.5290

k=5:

  single     : ARI=0.0012, Silhouette=-0.2742

  complete   : ARI=0.6188, Silhouette=0.4060

  average    : ARI=0.8143, Silhouette=0.4577

k=6:

  single     : ARI=0.0008, Silhouette=-0.3719

  complete   : ARI=0.5086, Silhouette=0.3142

  average    : ARI=0.8164, Silhouette=0.3588

==========
```

[moons−3] 拓展：测试不同聚类数k（make_moons 数据集）
[不同聚类数k的性能测试]

k=2:

single : ARI=1.0000, Silhouette=0.4685

complete : ARI=0.6823, Silhouette=0.4887

average : ARI=0.5073, Silhouette=0.4760

k=3:

single : ARI=0.9868, Silhouette=0.1394

complete : ARI=0.5108, Silhouette=0.4447

average : ARI=0.4057, Silhouette=0.4352

k=4:

single : ARI=0.9802, Silhouette=−0.1148

complete : ARI=0.3453, Silhouette=0.4515

average : ARI=0.5437, Silhouette=0.4860

k=5:

single : ARI=0.9802, Silhouette=−0.1545

complete : ARI=0.3352, Silhouette=0.4589

average : ARI=0.4264, Silhouette=0.5080

k=6:

single : ARI=0.9736, Silhouette=−0.1877

complete : ARI=0.3700, Silhouette=0.4747

average : ARI=0.3619, Silhouette=0.4976

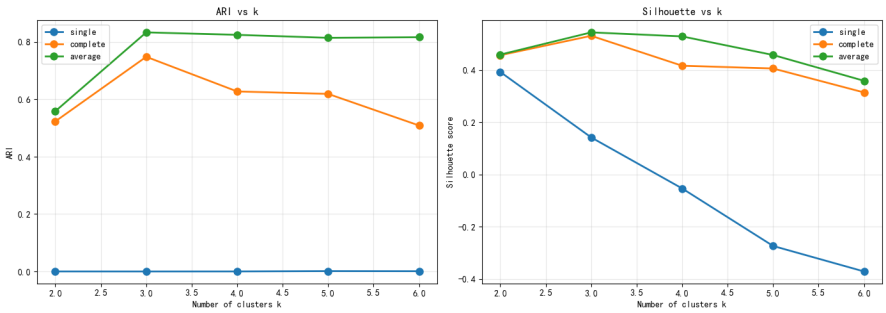
分析（make_moons 数据集）:

– make_moons 数据集为非凸形状簇，更考验聚类算法的鲁棒性

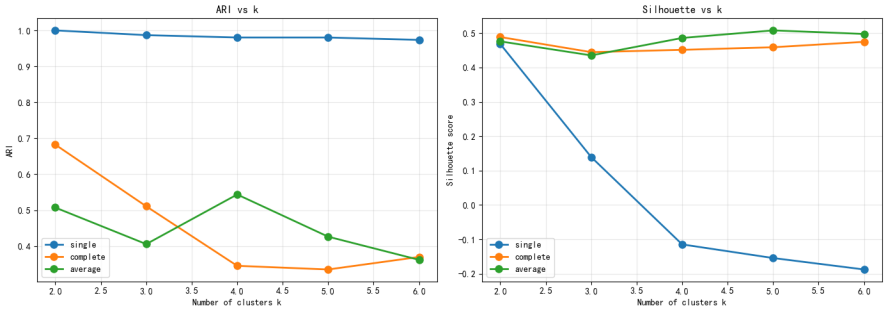
- 在 make_moons 数据集上，single-linkage 的 ARI 最高

- 可以观察不同链接方式在非球形簇上的优劣

2. 结果分析



上图是三种链接方式在 $k = 2$ 到 $k = 6$ 范围内的 ARI 和 Silhouette Score 曲线对比。所有算法的 ARI 都在 $k = 3$ 处达到峰值，这证明了数据集中真实的簇数量是 3。当 $k \neq 3$ 时，性能显著下降（尤其是 Complete-linkage），表明强行合并（$k = 2$）或强行拆分（$k > 3$）会导致聚类准确度降低。



这是三种链接方式在 $k = 2$ 到 $k = 6$ 范围内的 ARI 和 Silhouette Score 曲线对比（针对 make_moons）。Single-linkage 的 ARI 始终保持在 1.0 附近，再次确认其对该数据集的优秀适应性。Complete 和 Average-linkage 这两种方法的 ARI 随 k 的增加而剧烈波动和下降，表明它们对非凸形状和不正确的 k 值非常敏感。Single-linkage 的 Silhouette Score 随着 k 的增大急剧下降，并在 $k \geq 4$ 之后变为负值。这表明虽然其 ARI 仍高，因为链状结构容易在 k 增大时保持整体连通性，但强行拆分非凸形状会导致簇内样本距离过大，划分质量差。