实验报告：人工智能问题求解与实践——基于 kmeans 算法实现聚类

姓名：余俊洁

学号：523030910244

一、实验设计：

1、题目分析与算法选择：

首先我们初步分析四个聚类的题目，第一题给定了 1000 个一维且 Shape 为(2)的向量，并要求将它们聚类成两类。第二题也是给定了 1000 个一维且 Shape 为(2)的向量，需要将它们聚类成 N 类，类别 ID 从 0 至 N-1，但 N 值需要我们自己确定。第三题给定的 1000 个一维且 Shape 为(128)的向量,需要我们将它们聚类成 3 类,类别 ID 从 0 至 2。第四题则给定的是 10000 个一维且 Shape 为(128)的向量,需要将它们聚类成 N 类,类别 ID 从 0 至 N-1，同第二题一样，N 值也需要我们自己确定。

针对于聚类的问题，我第一时间想到的，可以通过 **kmeans 算法**，**EM 算法**或者 **GMM** 来实现聚类，但鉴于自身对 kmeans 算法更为熟悉，且 kmeans 在计算速度方面具有优良的性质，故最终选择采用 **kmeans 算法**实现四个聚类问题。

选择 kmeans 之后遇到的第一个问题就是如何选取合适的 k 值，是通过不停的调参数进行尝试？这条路显然行不通，在调参过程中首先面临的问题就是大量的时间成本，其次，尽管在第二个问题中，我们可以通过绘图的方法直观的看出所有的数据点该分为几类，但是当我们面临第四个问题时，就不能再通过绘制图像的方法直观的看出需要分为几类了，因为即使已经聚好类的 128 维数据投影在 2 维平面上也是杂乱无章的。因此，我们准备采用**手肘法**绘制图像，在图像的“手肘”附近的 k 值中进行进一步的筛选，以确定最终的 k 值的选择。

那么，对于聚类的效果，我们应该怎么去评估呢？在图像上显示是一种很好的方法，但是对于 128 维的数据，我们则需要先对数据进行**降维**，而对于降维，我选择了 **PCA 算法**，然后再在图像中绘制出来，这样就可以像二维数据一样，直观地看出聚类效果了。

在此基础上，为了进一步检验数据的聚类效果，我同时也选择引入**轮廓系数**，对每个题的聚类效果进行评估。

2、 算法简介：

（1）kmeans：一种常用的聚类算法，用于将数据集分成预先指定的 k 个簇（簇的数量由用户指定）。其主要思想是通过迭代的方式，将数据点分配到 k 个簇中的某一个，并将每个簇的中心调整为该簇所有数据点的平均值。kmeans 的简要步骤为：

[1] 初始化：随机选择 k 个数据点作为初始的簇中心（质心）。

[2] 簇分配：将每个数据点分配到距离其最近的簇中心所对应的簇中。

[3] 更新簇中心：对于每个簇，计算该簇所有数据点的平均值，将该平均值作为新的簇中心。

[4] 迭代：重复步骤 2 和步骤 3，直到簇中心不再发生显著变化，或者达到预定的迭代次数。

[5] 收敛：当簇中心不再发生显著变化时，算法收敛，停止迭代。

Kmeans 算法的优点包括简单易实现、计算速度快，适用于大规模数据集。

然而，kmeans 算法在对初始簇中心的选择敏感，可能会收敛到局部最优解，而不是全

局最优解。

（2）手肘法：一种用于帮助确定 kmeans 聚类算法中最佳 k 值的常用技术。通过绘制簇内平方和（SSE）与簇数 k 的关系图，来帮助找到合适的簇数。其基本思想是，随着簇数 k 的增加，SSE 会逐渐减小，因为更多的簇意味着更小的簇内距离。然而，当 k 值增加到某一点时，SSE 的下降速度会明显减缓，形成一个类似手肘的形状。这个“肘部”对应着最佳的 k 值，即在这一点上增加的簇数不再显著降低 SSE。手肘法的简要步骤为：

[1] 将 k 值取一系列可能的值（如从 1 到预设的最大值）。

[2] 对于每个 k 值，运行 k 均值算法，并计算对应的 SSE。

[3] 绘制 k 值与对应的 SSE 的折线图。

[4] 观察图形中是否存在明显的"肘部"，该点对应的 k 值就是最佳的簇数。

手肘法的优点在于简单易行，并且通常能够给出合理的结果。

然而，有时候手肘法并不能清晰地确定最佳的 K 值，特别是当数据没有明显的手肘点时。在这种情况下，可能需要对代码进行多次运行或者结合其他方法来确定最佳的 k 值。

（3）PCA：一种常用的降维技术，发现数据中的主要特征并将数据投影到一个低维空间中。PCA 的目标是通过线性变换将原始数据投影到一个新的坐标系中，使得投影后的数据具有最大的方差。通过保留尽可能多的方差，PCA 能够保留数据中的关键信息，同时减少数据的维度。PCA 的主要步骤为：

[1] 数据标准化：对原始数据进行标准化处理，使得每个特征的均值为 0，方差为 1。

[2] 计算协方差矩阵：计算标准化后数据的协方差矩阵，该矩阵描述了数据特征之间的关系。

[3] 计算特征值和特征向量：对协方差矩阵进行特征值分解，得到特征值和对应的特征向量。

[4] 选择主成分：根据特征值的大小，选择最重要的特征向量作为主成分，通常选择方差最大的前几个特征向量作为主成分。

[5] 投影数据：将原始数据投影到选定的主成分上，得到降维后的数据集。

PCA 的优点包括简单易实现、计算高效，并且能够提供数据的最重要的结构信息，被广泛应用于数据预处理、特征提取和数据可视化等领域。

然而，PCA 假设数据是线性可分的，对非线性结构的数据表现可能不佳。在这种情况下，可以考虑使用核 PCA 等方法来处理非线性数据。

（4）轮廓系数：一种用于评估聚类结果质量的指标，它结合了簇内的紧密度和簇间的分离度，对聚类效果进行综合评价。轮廓系数的取值范围在[-1, 1]之间，如果轮廓系数接近于 1，表示样本点与其所属簇内的其他点距离相近，与其他簇的点距离较远，说明聚类结果合理。如果轮廓系数接近于 0，表示样本点与其所属簇内的其他点距离相近，与其他簇的点距离相近，说明样本点处于两个簇的边界上。如果轮廓系数接近于-1，表示样本点与其所属簇内的其他点距离较远，与其他簇的点距离较近，说明聚类结果可能存在错误。轮廓系数的计算方法为：

[1] 对于每个样本点，计算它与同簇内其他点的平均距离（簇内紧密度，记为 a）。

[2] 对于每个样本点，计算它与其最近的其他簇的所有样本点的平均距离，选择其中最小值（簇间分离度，记为 b）。

[3] 计算每个样本点的轮廓系数：轮廓系数 = (b - a) / max(a, b)

[4] 对所有样本点的轮廓系数求平均，得到聚类结果的整体轮廓系数。

轮廓系数越大，表示聚类结果越合理。在实际应用中，通常会选择具有最大轮廓系数的聚类数作为最优的聚类数。

3、 代码实现：

伪代码（辅助函数部分）：

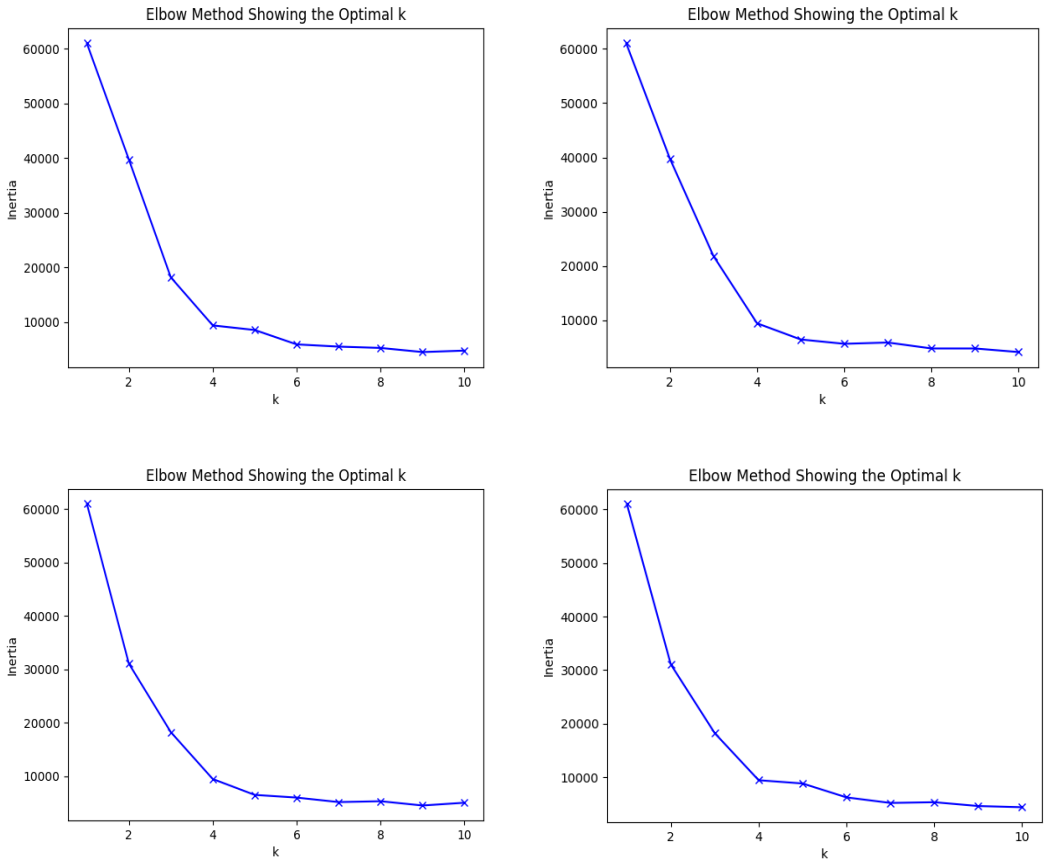
```
euclidean_distance (a, b):
    Calculate the Euclidean distance between two points a and b.

kmeans (data, k, max_iters=100):
    Initialize centroids by randomly selecting k data points.
    Iterate for a maximum of max_iters times:
        Assign each data point to the closest centroid.
        Calculate new centroids based on the mean of assigned data points.
        If centroids do not change, break the loop.
    Return the final labels and centroids.

compute_inertia (data, labels, centroids):
    Calculate and return the inertia (sum of squared distances of samples to their closest
    cluster center).

elbow_method (data, max_k=10):
    Calculate inertia for each value of k from 1 to max_k.
    Plot the inertia values to visualize the elbow method.

silhouette_score (data, labels):
    Calculate the silhouette score for the given data and labels.

pca (data, n_components=2):
    Perform PCA to reduce data dimensions to n_components.
    Return the transformed data.

plot_clusters (data, labels, title="Cluster Visualization"):
    Plot the data points with different colors for each cluster based on the labels.
```

二、参数选择：

对于 task1 和 task3 而言，k 值已经给定，故我接下来仅对 task2 和 task4 的 k 参数的选取进行讨论。

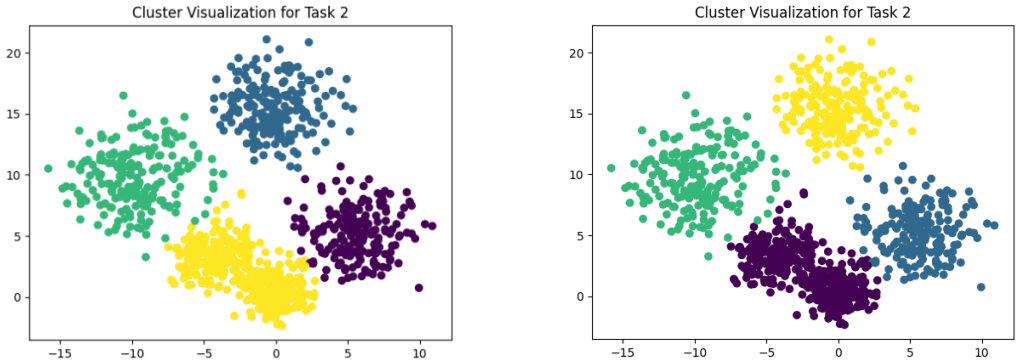
首先是对于 task2 参数选择的讨论：

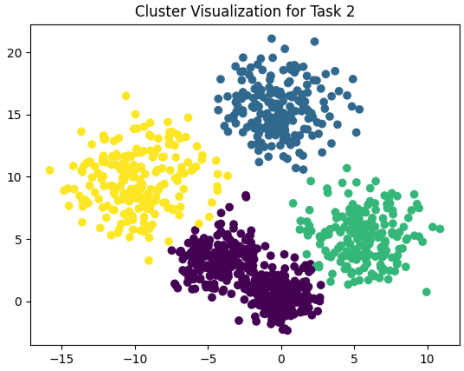
如上所言，我选择用手肘法对 k 值进行判断与选择。但鉴于在某些情况下的不稳定性，我通过多次运行程序，绘制了 4 张手肘图，如下所见：

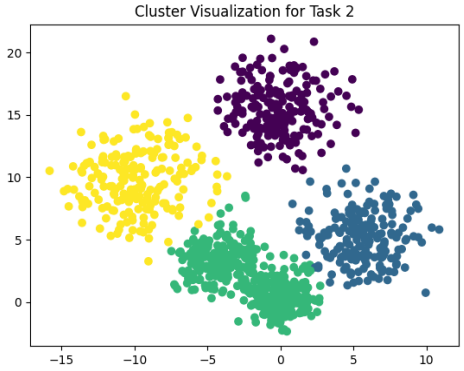


平均看来，图像在 k 值为 4 或 5 附近出现拐点，我们可以初步确定当 k = 4 或 5 时，会获得较好的聚类效果。为了更好地研究 k 值是取 4 还是 5，我选择对两种情况下的聚类结果进行比较，并通过图像的形式直观地显示出来，进而确定最佳的 k 值。

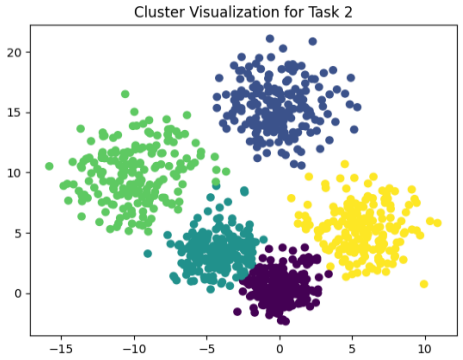
首先是 k = 4 时的聚类情况：

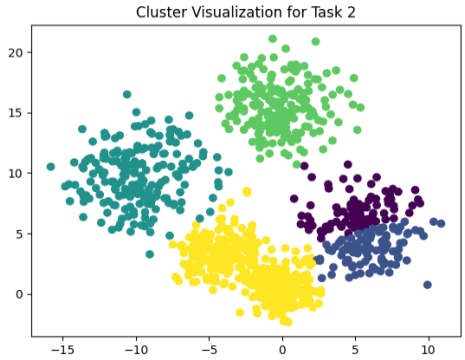


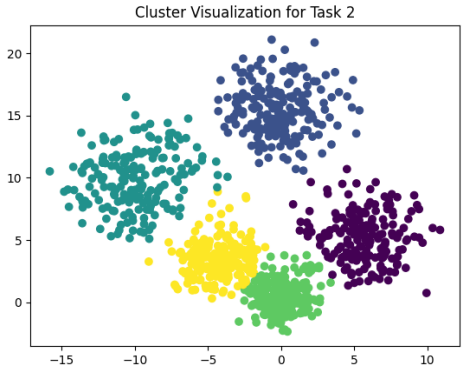


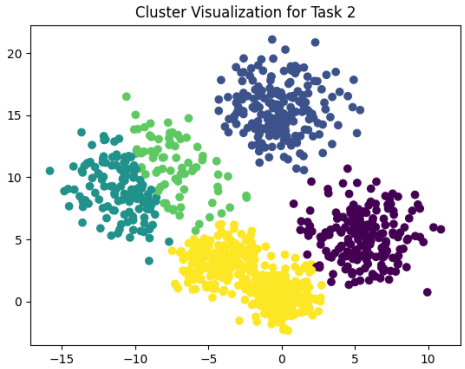


其次是 k = 5 时的聚类情况：





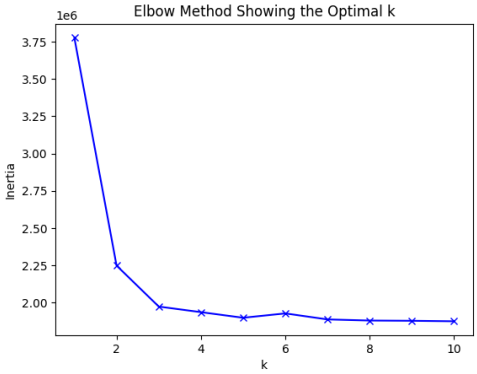


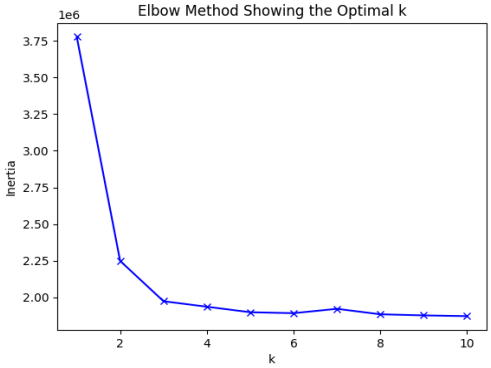


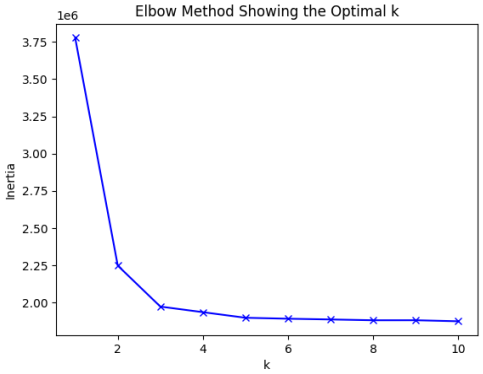
可见，当 k = 4 时聚类效果很稳定，且有很明显的聚类效果。而反观当 k = 5 时，则出现了不稳定的现象，注意到在中下方的一组类似“哑铃型”的数据点分布，在该“哑铃型”数据簇中间部位，有断开的趋势，同时左上方的数据簇的聚合性也不是很紧密，这就导致可能会出现类似图四的不稳定的聚类效果，同时，由于这两个数据簇都没有明显的分离的趋势，因此，4 更可能是 k 的最优值。

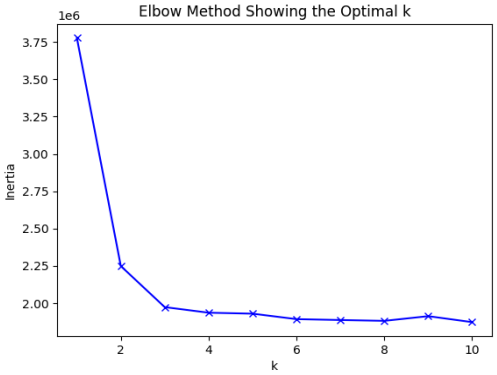
其次，是对于 task4 参数选择的讨论：

同样的，为了数据的稳定性，我首先绘制了 4 张手肘图，如下所见：



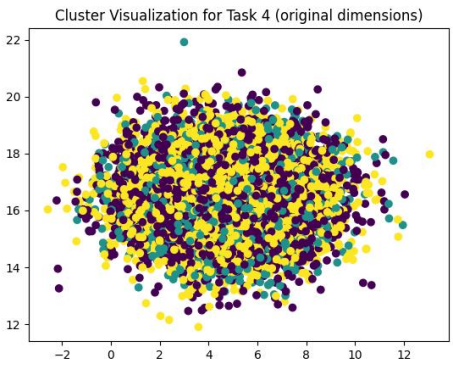


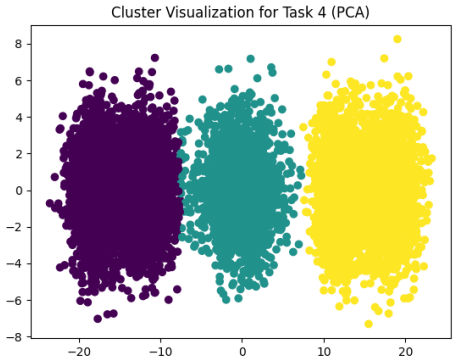


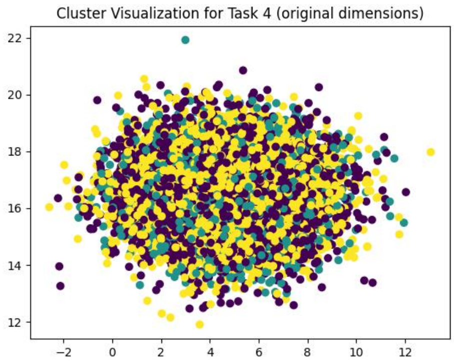


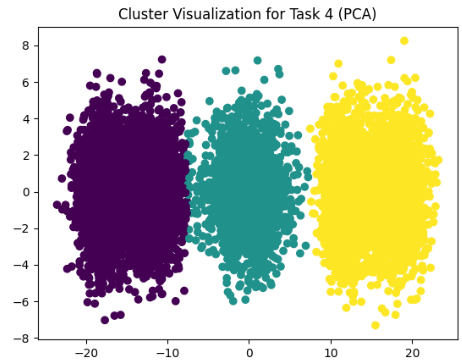
平均看来，图像在 k 值为 3 或 4 附近出现拐点，我们可以初步确定当 k = 3 或 4 时，会获得较好的聚类效果。同样的，为了更好地研究 k 值是取 3 还是 4，我选择对两种情况下的聚类结果进行比较，并通过图像的形式展现，但由于数据的维度为 128 维，在图像上并不能直观地反映出聚类的效果到底如何，故我们先对聚类的数据进行 PCA 降维，降到 2 维后再在图像上加以呈现，进而确定最佳的 k 值。

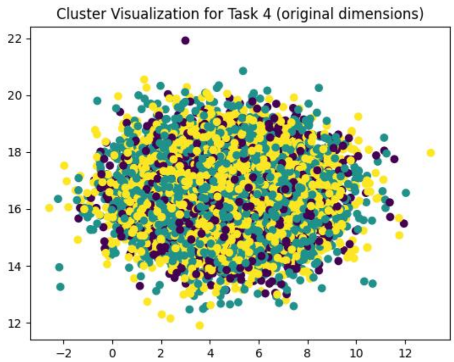
首先是 k = 3 时的聚类情况：

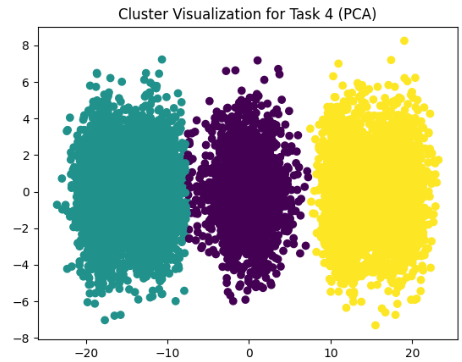


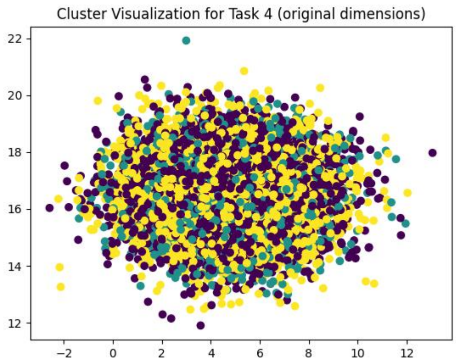


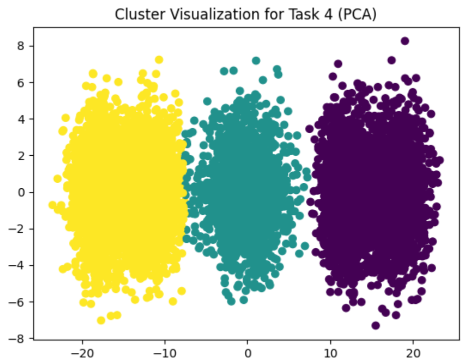




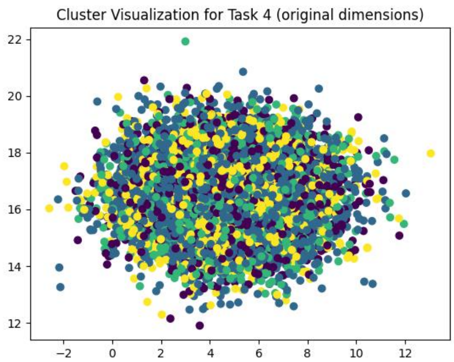


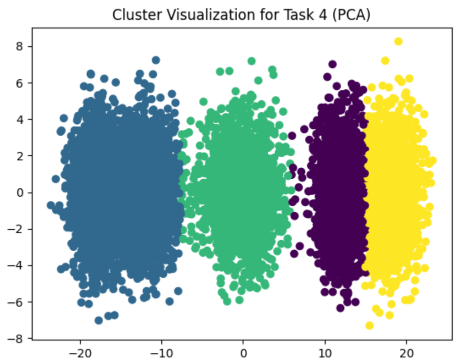


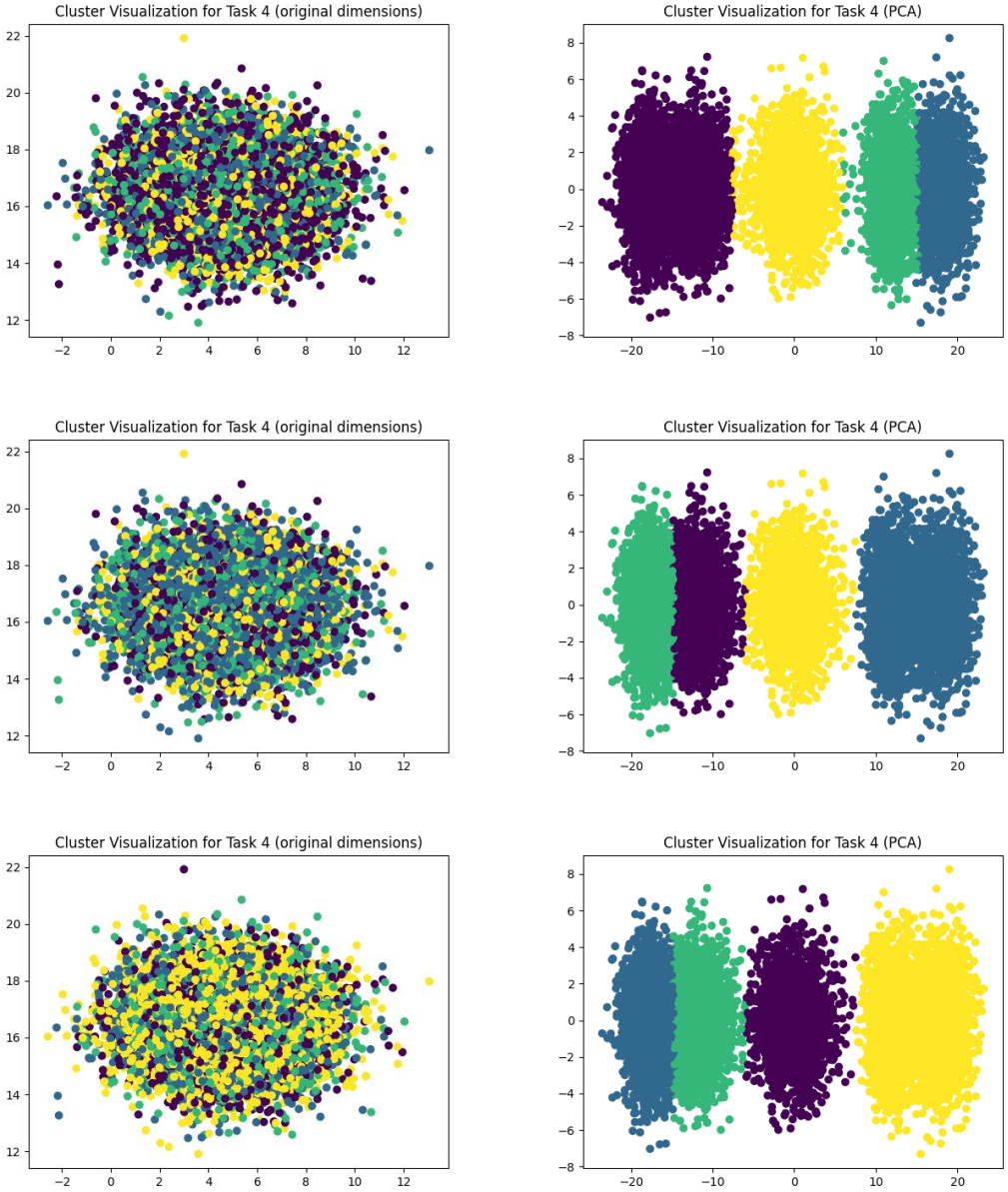




其次是 k = 4 时的聚类情况：







同样可见，当 k = 3 时聚类效果很稳定，且有很明显的聚类效果。而反观当 k = 4 时，则出现了不稳定的现象，注意到当我们把聚类的数据降维到二维之后，然后投影到平面图上时，发现数据的分布主要为三大块，虽然左右两边的数据簇的中间部分的耦合会显得不是很紧，但是已经能够很好地达到我们的要求了，故最终选择 k 的值为 3.

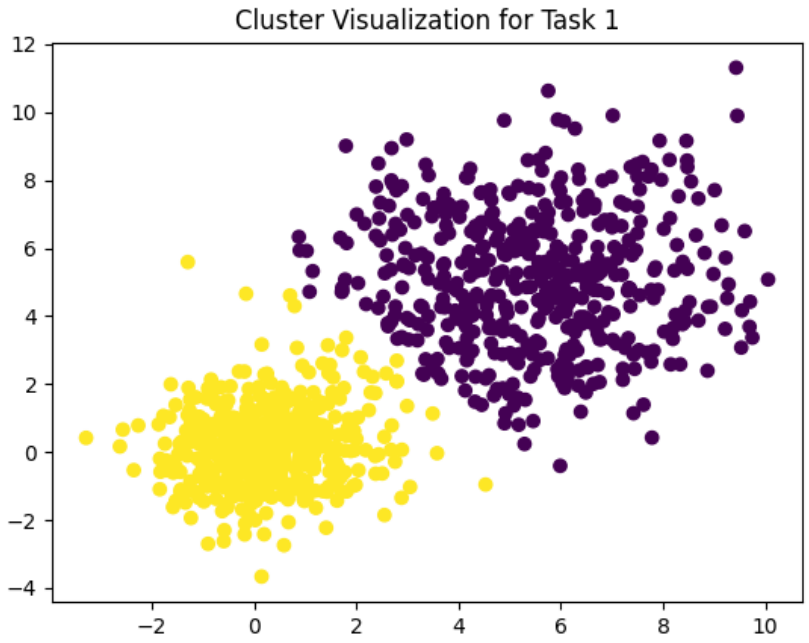
三、结果与分析：

1、 实验结果：

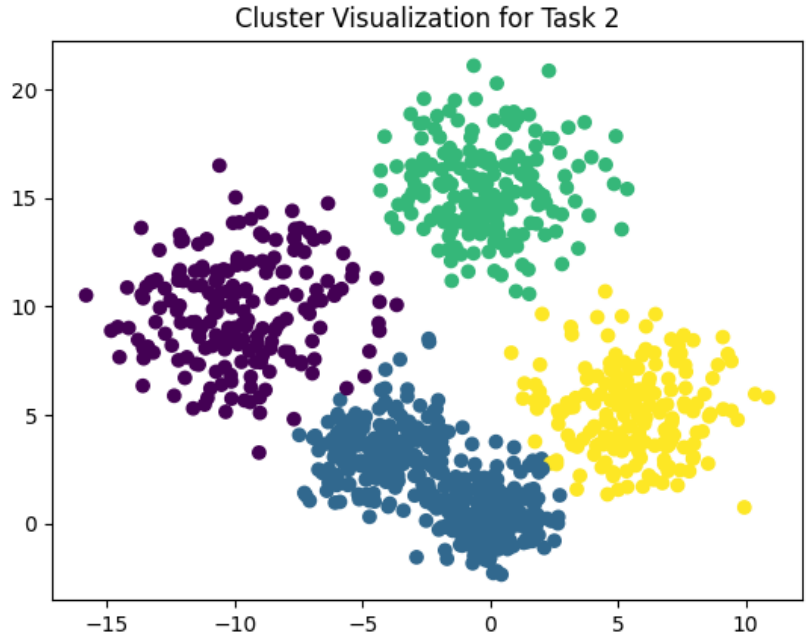
对于每组的聚类之后的数据标签，已经存放在了 4 个 label 的.npy 文件中了。

在此，对四个题目的聚类结果进行可视化呈现。

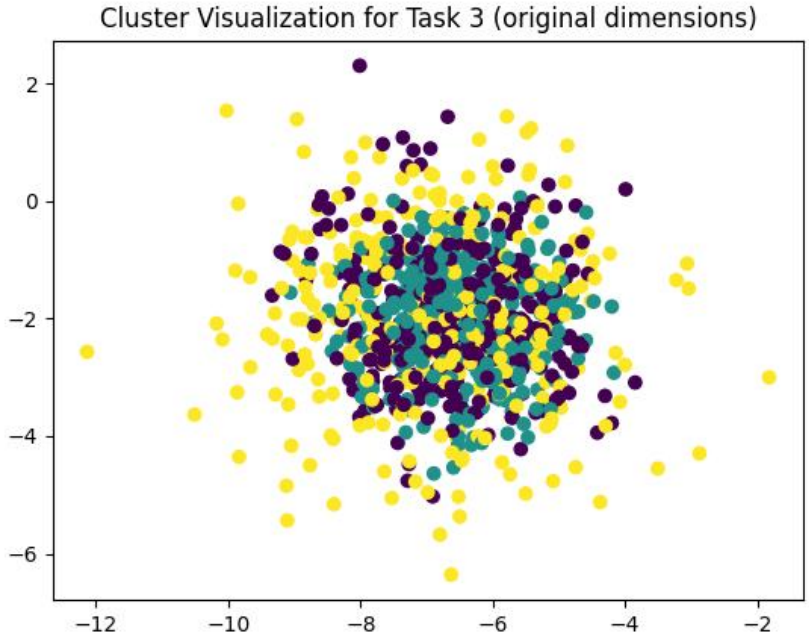
Task_1:

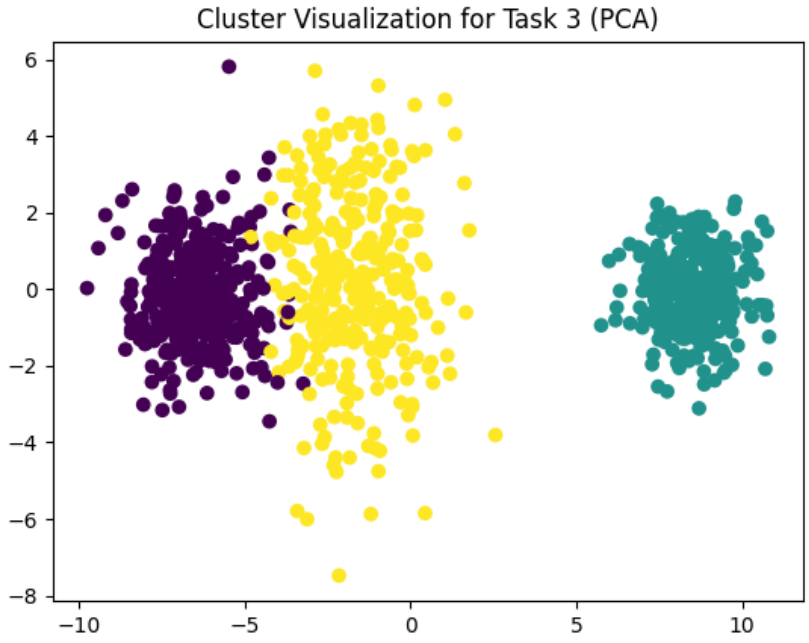


Task_2:

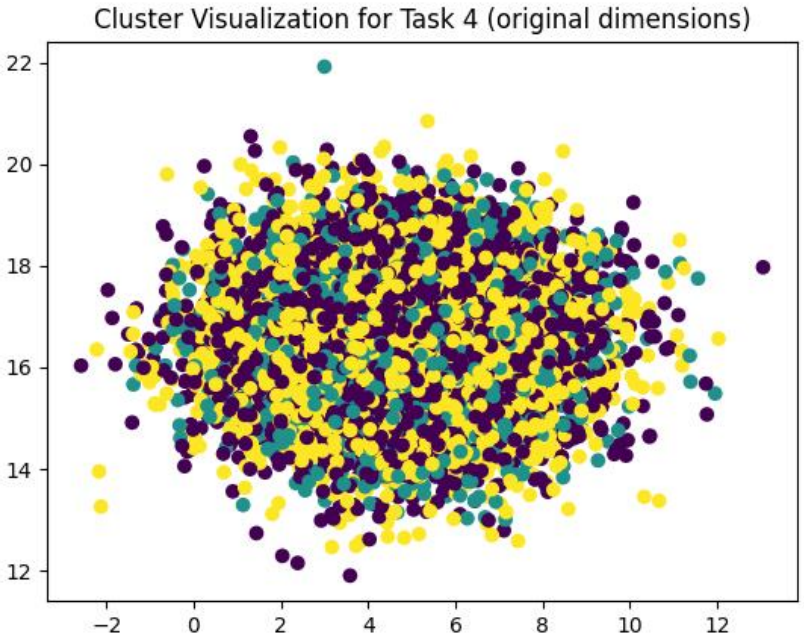


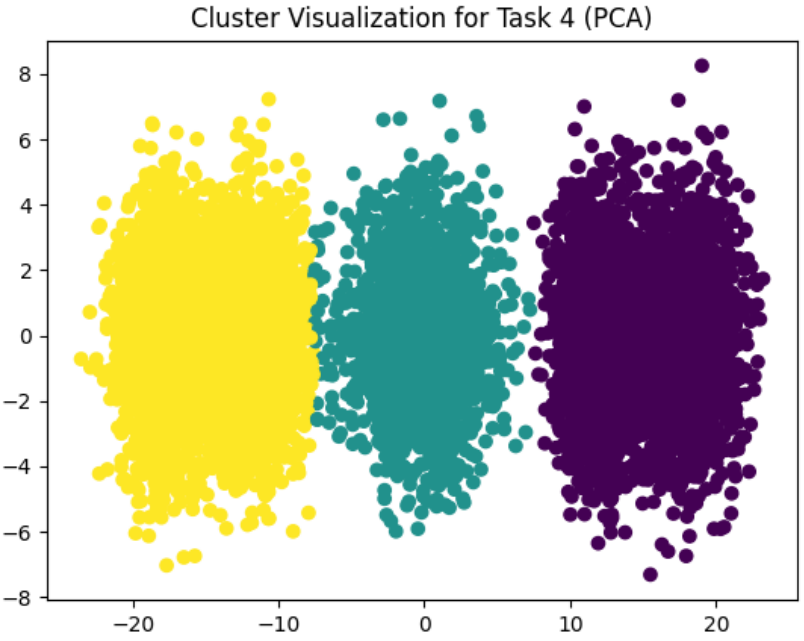
Task_3:


Cluster Visualization for Task 3 (original dimensions)
2
0
−2
−4
−6
−12
−10
−8
−6
−4
−2


Cluster Visualization for Task 3 (PCA)
6
4
2
0
−2
−4
−6
−8
−10
−5
0
5
10

Task_4:





2、 实验结果评估：

（1） 直观图像呈现：从上面所呈现出来的图像看来，每一组的聚类效果还是很明显的。

（2） 轮廓系数评估：由于 kmeans 聚类的效果并不是强稳定的，所以为了减小误差我在多次运行程序之后得到了以下几组数据：

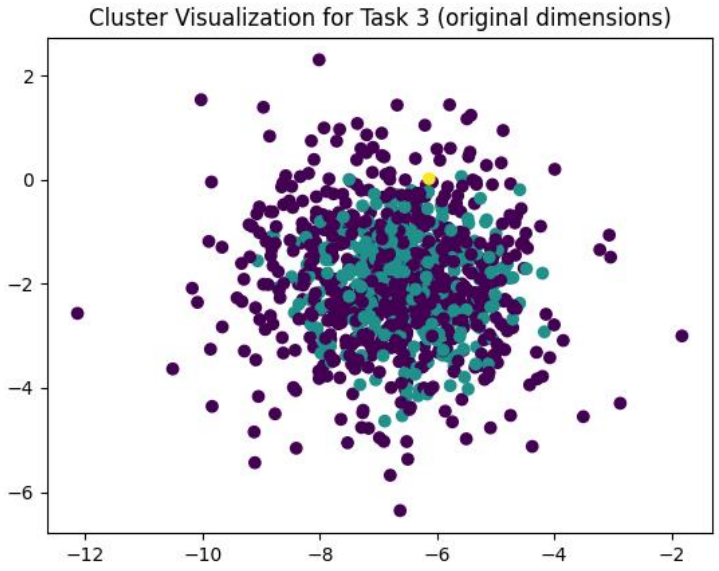
Task_1	Task_2	Task_3	Task_4
0.629787	0.678108	0.164062	0.324627
0.629787	0.727351	0.163873	0.323787

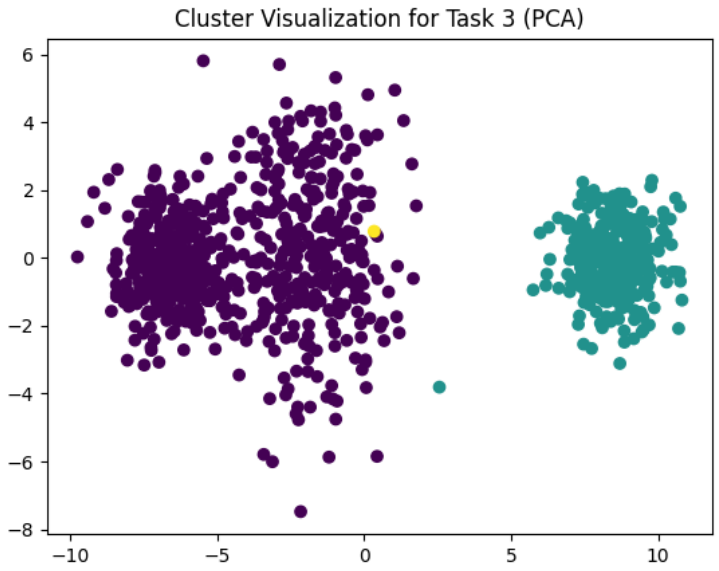
0.629787	0.687035	0.163608	0.324627
0.629787	0.727351	0.164062	0.324627
0.629787	0.678108	0.163876	0.341918

由表中的数据可见，task_1 和 task_2 的轮廓系数都处于 0.6 以上，可见其聚类效果还是挺不错的，而对于 task_3 和 task_4 两组数据的聚类效果，其轮廓系数就不算很高，仅仅是处于可接受的范围，可见 kmeans 算法在处理较高维的数据时显得不是那么有效，可见在算法的设计方面可以进一步优化。

3、 实验结果不足：

当我们在反复运行程序的过程中，发现在 task_3 的聚类任务中，会有 30%的概率出现一种极端情况，如下所示：





由于 kmeans 算法在对初始簇中心的选择敏感的特性，在初始随机选择质心的时候，可能会选取到某些较为极端的点，而使得最终得到的结果收敛到局部最优解，而不是全局最优解。如上图所示的聚类效果就可能是因为在聚类的初始选取质心的步骤出了问题，使得最终某一类只有一个数据点，而没有达到聚类的效果，问题根源在于初始选取质心的随机性，可通过概率选择的方式使得聚类效果更为准确和稳定。

四、实验总结

这次实现聚类的实验很好地检验了我们对于聚类算法的掌握程度以及实验修养，同时对于我这样的对于 python 代码十分不熟练的同学是一个很大的挑战，在接近 1 周的时间里，不断地 debug，不断地优化，全面自己的程序，不断地调整各个参数，在这个过程中，收获真的很大，非常感谢课程组的老师的设计与安排！同时也非常感谢互相交流学习的同学！

聚类在人工智能领域是一个非常基础也是非常重要的基本问题，学会了聚类的一般算法与代码实现，并实际上地完成了挺有挑战性的任务，我感觉真的很有成就感，也坚定了我在这条路上继续走下去的信念！在这个过程中，我自身的 python 能力，阅读资料文献的能力也得到了很大的提高！总之，真心非常感谢这样一个大作业，这样一段经历！