聚类方法实验报告

Boxuan Hu

Xi'an Jiaotong University

摘要 本文探讨了传统机器学习算法在聚类任务中的应用，重点研究了 K-Means (自编实现) 和 Agglomerative Clustering 两种算法。在同一数据集 Cifar10 上实现并比较了这两种算法的性能，通过参数调优，优化模型在准确率、召回率和错误率等方面的表现，并评估了模型的训练和执行时间。实验结果表明，这两种方法在聚类任务中都表现有效，但各自具有不同的性能特征。本实验全部代码已开源发布至 `https://github.com/root-hbx/ml-exp/tree/main/cluster-exp`

Keywords: K-Means · Agglomerative Clustering

1 介绍

本文旨在探索传统机器学习算法在聚类任务中的应用，并通过实验比较不同算法的性能。在本研究中，我选择了两种具有代表性的传统机器学习算法：K-Means (自编实现) 和 Agglomerative Clustering，将其应用于同一聚类任务。通过调整算法的关键参数，寻求最优模型配置以优化聚类性能。此外，还比较了两种算法在准确率、召回率、错误率以及训练和执行时间等方面的表现，以全面评估其性能。

实验中，首先介绍了用于聚类任务的数据集及其预处理过程，以确保数据的质量和适用性。随后，详细描述了模型训练的实现过程，包括自实现的算法、参数设置和模型训练。为找到最优参数，比较了不同参数组合下的模型表现。

实验结果显示，这两种方法在聚类任务中均表现良好，但在准确率、召回率和错误率等方面存在一定差异。此外，不同算法在训练和执行时间上也有显著差异。通过对比实验结果，可以得出有价值的结论，并为类似聚类任务中选择合适的机器学习算法提供参考。

本文不仅展示了两种方法在聚类任务中的应用，还通过对比实验揭示了它们各自的优缺点。研究结果对于理解和应用传统机器学习算法具有一定的

理论和实践意义，同时也为进一步探索机器学习在聚类任务中的应用提供了有价值的参考。

2 数据集

CIFAR-10 数据集是聚类任务中常用的基准数据集，包含 60,000 张 32x32 的彩色图像，分为 50,000 张训练样本和 10,000 张测试样本。每张图片属于 10 个物体类别之一：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船和卡车。CIFAR-10 为研究人员提供了一个具有挑战性但可控的数据集，适用于各种计算机视觉技术的实验。

2.1 数据预处理

在输入模型前，进行预处理步骤以确保数据格式和范围适合模型：

1. 像素缩放：将图像像素值从 0-255 缩放到 0-1。
2. 归一化：利用预先计算的均值 (0.4914, 0.4822, 0.4465) 和标准差 (0.2023, 0.1994, 0.2010) 对图像进行标准化，加速训练并提升模型表现。
3. 通道转换：将图像通道从 HWC（高、宽、通道）格式转换为 CHW（通道、高、宽）格式，符合深度学习模型的输入要求。

2.2 数据集划分

为了评估模型性能，进一步将训练集划分为训练集和验证集，使用 PyTorch 框架加载全部训练数据并进行预处理。具体操作如下：

1. 将数据集按 10,000 为一批进行分批。
2. 仅使用第一批作为训练和测试集，并通过 sklearn 的 train_test_split 函数进行划分。
3. 最终得到训练集（X_train, Y_train），约占第一批训练数据的 80%。

由于本任务为聚类任务，未采用传统的测试集划分。

3 机器学习模型

本文采用了 K-Means(自编实现) 和 Agglomerative Clustering 两种机器学习方法进行分析和预测。每种方法具有独特的优势和适用场景，选择它们以全面评估不同算法在给定数据集上的表现。

3.1 模型简介

K-Means：广泛使用的迭代聚类算法，旨在将数据集划分为 K 个预定义的聚类。通过初始化 K 个质心，将每个数据点分配到最近的质心，然后根据分配重新计算质心。K-Means 简单高效，适合大规模数据集，且常能生成高类内相似度、低类间相似度的聚类结果。

Agglomerative Clustering：自底向上的层次聚类方法，最初将每个数据点视为单独的聚类，然后逐步合并最近的聚类，直到达到预定的聚类数。常用的距离度量包括单链接、全链接和平均链接。该方法灵活，可生成不同形状和大小的聚类，并能可视化聚类层次结构。

3.2 K-Means 模型实现

模型简介 K-Means 是一种常用的无监督聚类算法，通过迭代将数据划分为 K 个簇。其核心思想是最小化样本到各自簇中心的距离。算法首先随机选取 K 个初始中心点，然后不断分配样本到最近的中心，并更新中心位置，直到收敛。K-Means 具有计算效率高、实现简单等优点，广泛应用于图像分割、市场分析等领域。

模型实现 详见图 1。实现阶段创建字典 “Func” 存储分类器引用，定义函数 “Deal” 接收分类器名称、训练集和测试集，执行训练和预测。函数从字典中获取相应分类器类，实例化并传入参数，使用训练集训练模型，测试集进行预测。调用同学实现的 visualize 函数可视化预测结果，打印各项评估指标。

3.3 Agglomerative Clustering

模型简介 凝聚层次聚类（Agglomerative Clustering）是一种基于层次结构的聚类方法，属于自底向上的聚类策略。它的基本思想是：将每个样本初始化为一个独立的簇，然后通过计算簇与簇之间的距离，逐步将距离最近的两个簇合并，直到满足预设的终止条件（如簇的数量达到指定值）。该算法常用于揭示数据的嵌套层次结构，适合处理具有树状结构或层次关系的数据集。

模型原理 Agglomerative Clustering 的核心在于定义簇之间的距离（也称为链接准则）以及合并过程的控制。算法流程如下：

1. 初始化：将每个数据点看作一个独立的簇，共有 n 个簇。

```python
class MyKMeans:
    def __init__(self, n_clusters=8, max_iter=300, tol=1e-4, random_state=None):
        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.tol = tol
        self.random_state = random_state
        self.labels_ = None
        self.cluster_centers_ = None
        self.inertia_ = None
        self.n_iter_ = 0

    def _init_centroids(self, X):
        np.random.seed(self.random_state)
        idx = np.random.choice(X.shape[0], self.n_clusters, replace=False)
        return X[idx]

    def fit(self, X):
        n_samples, n_features = X.shape

        self.cluster_centers_ = self._init_centroids(X)

        prev_inertia = float('inf')
        for i in range(self.max_iter):
            distances = np.zeros((n_samples, self.n_clusters))
            for j in range(self.n_clusters):
                distances[:, j] = np.sum((X - self.cluster_centers_[j])**2, axis=1)

            self.labels_ = np.argmin(distances, axis=1)
            new_centroids = np.zeros((self.n_clusters, n_features))
            for j in range(self.n_clusters):
                if np.sum(self.labels_ == j) > 0:
                    new_centroids[j] = np.mean(X[self.labels_ == j], axis=0)
                else:
                    new_centroids[j] = X[np.random.choice(n_samples)]

            shift = np.sum((new_centroids - self.cluster_centers_)**2)
            self.cluster_centers_ = new_centroids

            inertia = 0
            for j in range(self.n_clusters):
                cluster_points = X[self.labels_ == j]
                if len(cluster_points) > 0:
                    inertia += np.sum((cluster_points - self.cluster_centers_[j])**2)

            self.inertia_ = inertia
            self.n_iter_ += 1

            if abs(prev_inertia - inertia) < self.tol:
                break

            prev_inertia = inertia

        return self
```

图 1. 自我实现的 KMeans 模型

2. 计算所有簇对之间的距离，选出距离最小的两个簇。
3. 合并这两个簇，簇的总数减一。
4. 重复步骤 2 和 3，直到达到终止条件（如簇数为 k）。

连接准则 常见的链接准则包括：

- 单链接（Single Linkage）：两个簇之间最近样本点的距离。
- 全链接（Complete Linkage）：两个簇之间最远样本点的距离。
- 平均链接（Average Linkage）：两个簇之间所有点对的平均距离。
- Ward 链接：以最小化簇内平方误差（Within-cluster Sum of Squares）为目标的合并策略。

该算法最终可以用树状图表示样本之间的层次结构，用户可以通过选择不同的截断高度来获得不同数量的聚类结果。

4 实验设计

4.1 评估指标

- 训练时间：模型训练所用时间；

- 执行时间：模型执行所用时间；
- 准确率：分类正确样本占比；
- 错误率：分类错误样本占比；
- 精确率：预测为正样本中真实为正的比例，计算宏平均、微平均、加权平均及各类别精确率；
- 召回率：真实正样本中被正确预测为正的比例，计算宏平均、微平均、加权平均及各类别召回率；
- 混淆矩阵：展示各类别预测结果，包括真阳性（TP）、假阳性（FP）、真阴性（TN）、假阴性（FN）。

4.2 K-Means

在 KMeans 模型中，研究了参数 n_components 对结果的影响，分别设置为 1、2、3。最佳模型（以准确率为准）对应的 n_components 为 2。

实验结果见图 2，图 3 和图 4。

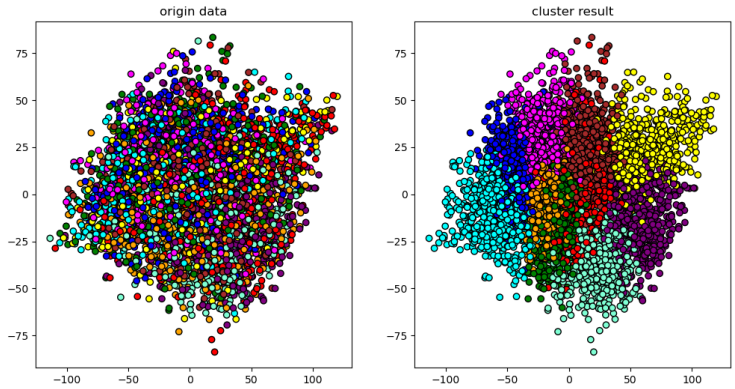

图 2. 模型输出 (n_components=3)

4.3 Agglomerative Clustering

我们跟上面一样，研究了 n_components 参数，最佳模型同样为 2。

实验结果见图 5，图 6 和图 7。

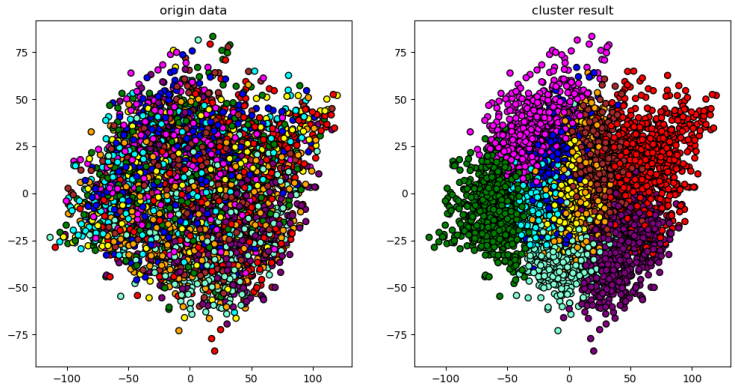

图 3. 模型输出 (n_components=10)

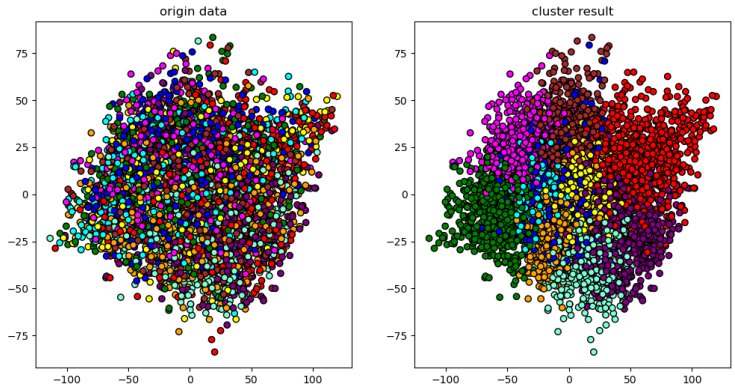

图 4. 模型输出 (n_components=100)

4.4 实验结论

总体来看，Agglomerative Clustering 模型在本次数据集上表现最佳。

5 结论

本文对 K-Means 和 Agglomerative Clustering 两种传统机器学习聚类算法进行了对比分析。通过在 Cifar10 数据集上的实验，发现两种算法在聚类任务中均有效，但在准确率、召回率和错误率等方面存在差异，同时训练和执行时间也不同。参数优化对于提升聚类性能至关重要。

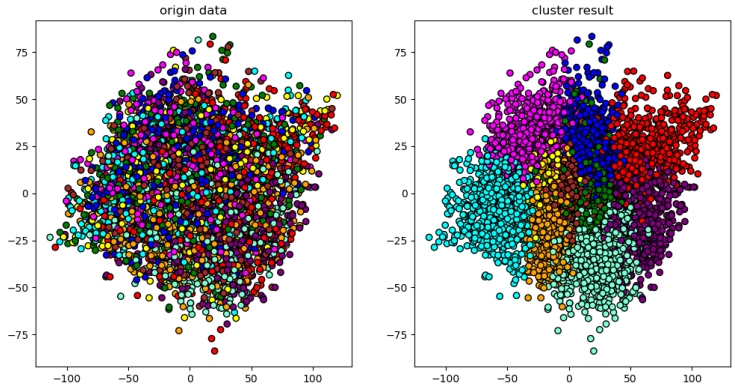

图 5. 模型输出 (n_components=1)

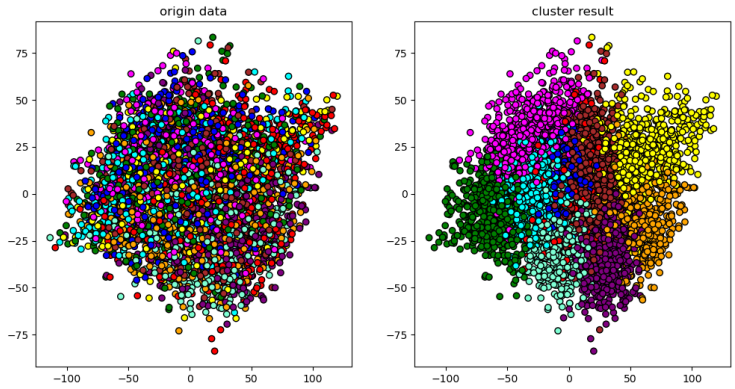

图 6. 模型输出 (n_components=2)

K-Means 在初始质心选择和聚类数设定上影响较大，Agglomerative Clustering 则受链接准则和停止条件影响显著。

总体而言，K-Means 速度较快，适合大规模数据集，Agglomerative Clustering 则在聚类形状和链接准则上更具灵活性。选择算法时需根据具体任务权衡模型准确性和效率。未来可进一步将这些算法应用于更复杂、多样的数据集，并结合更多特征和技术提升聚类表现。

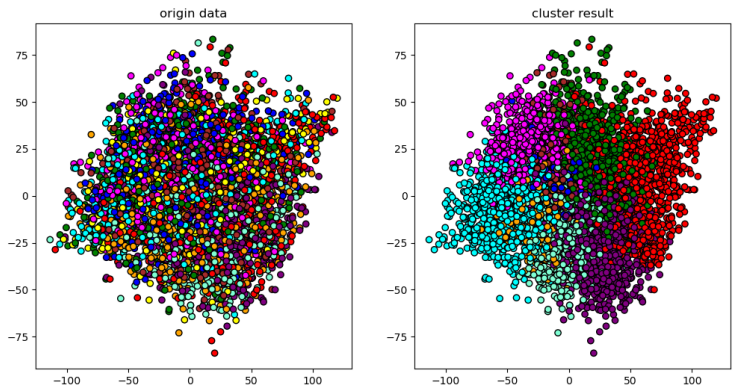

图 7. 模型输出 (n_components=3)