Kmeans 算法实验报告

2023200440

2024 年 10 月 23 日

1 实验目的

本次实验是通过实现 kmeans 聚类算法，掌握 kmeans 算法的原理和实现方法，了解 kmeans 的局限和缺陷。通过对 kmeans 算法的学习，对无监督学习有初步的认识，并且掌握一定的聚类算法的基础。

2 实验环境

机器：windows11
解释器：python3.12
编辑器：pycharm

3 实验方法

3.1 实验步骤

1. 生成数据 (1) 随机生成数据: 生成符合高斯分布的随机二维数据

```
def init_random_data(n, sigma, mu):
    data = sigma * np.random.randn(n, 2) + mu
    return data
```

(2) 获取数据集 AI Cat and Dog Images by DALL · E Mini, 调用文澜 API 获取图像编码，并以 numpy_ndarray 数据格式保存在本地，从本地加载数据时进行降维

```
def load_images_deocde(num):
......
def save_data(data):
......
def load_data(path: str):
```

2. 随机化质心方法 (1) 随机选择 k 的质心

```
indexs = np.random.choice(data.shape[0], k, replace=False)
return data[indexs]
```

(2)Kmeans++ 初始化质心随机选择第一个质心，计算所有点与该质心之间的距离，并将距离归一化得到每个数据点被选为质心的概率，计算概率累计和，依次遍历，选取第一个累积概率大于等于随机数 r 的点作为下一个质心

```
def init_centroids_plusplus(data, k):
"""
kmeans++随机初始质心
"""
centroids = [data[np.random.randint(data.shape[0])]]
for _ in range(1, k):
    distances = np.array([min([np.inner(c-x, c-x) for c in centroids]) f
    probabilities = distances / distances.sum()
    cumulative_probabilities = np.cumsum(probabilities)
    r = np.random.rand()
    for j, p in enumerate(cumulative_probabilities):
        if r < p:
            centroids.append(data[j])
            break
return np.array(centroids)
```

3. Kmeans 算法实现 (1) 用数据集数据或随机数据选取随机 k 个质心 (2) 计算每个点到每个质心的欧式距离，分配到最近质心所属的簇中 (3) 重新计算质心，并计算新旧质心的余弦相似度，检查是否收敛 (质心基本变化或者变化小于阈值) 或深度是否超过最大设定深度 (4) 若收敛，结束迭代，否则继续迭代 (5) 返回聚类结果和聚类中心 (6) 将聚类结果进行可视化，观

察聚类效果

4 代码细节

1. Kmeans 算法

```
def Kmeans(centroids, k, maxdepth=10000, epsilon=0.98):
    """
    maxdepth: 最大迭代深度
    epsilon : 前后两次质心向量余弦相似度的阈值
    """
    depth = 0
    clusters = [[] for i in range(k)]    # k个簇
    while depth < maxdepth:
        # 分配点到每个簇
        for point in data:
            lable = cal_distance(centroids, point)
            clusters[lable].append(point)
        # 重新计算质心
        new_centroids = update_centroids(clusters)
        # 检查收敛
        if cal_consine_similiraty(centroids, new_centroids) > epsilon:
# 如果质心基本不在变化，结束迭代
            break
        # 更新质心
        centroids = new_centroids
        depth += 1
    return clusters, centroids
```

---

2. 质心的更新

```
def update_centroids(clusters):
    """
```

```
    计算每一簇的平均值更新质心
    """
    new_centroids = []
    for cluster in clusters:
        cluster = np.array(cluster)
        new_centroids.append(cluster.mean(axis=0))
    return np.array(new_centroids)
```

5 实验结果

5.1 结果展示

实验结果如下:

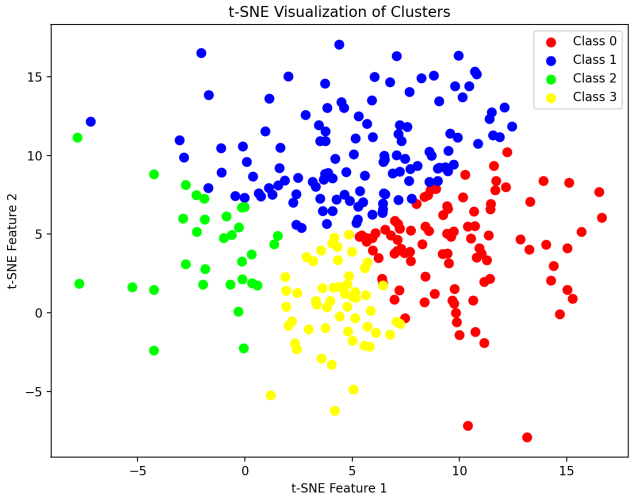

图 1: 随机数据聚类结果

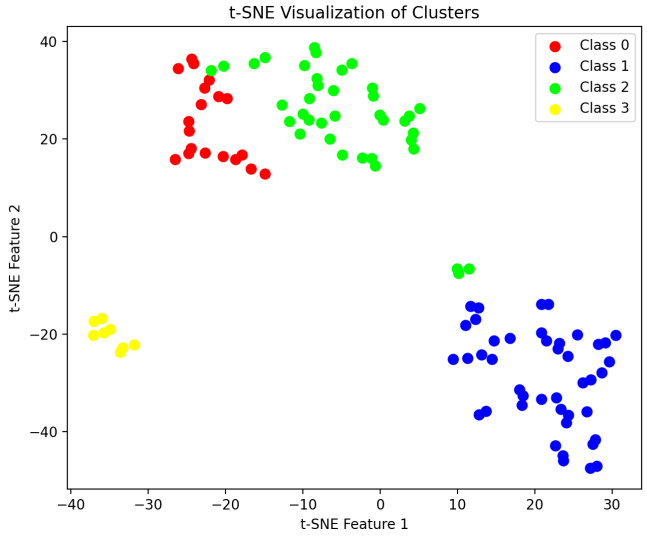
t-SNE Visualization of Clusters
Class 0
Class 1
Class 2
Class 3
t-SNE Feature 1
t-SNE Feature 2
40
20
0
−20
−40
−40
−30
−20
−10
0
10
20
30

图 2: 数据集聚类结果