K-Means聚类算法

田田 经12计18 2021011048

一、实现K-Means聚类算法

（一）任务分析

本次作业使用 K-Means 聚类算法实现对 MNIST 数据集的训练集部分的分类。本次作业只在训练集上进行聚类和分析，不在测试集上测试结果具体如何。

（二）算法整体逻辑设计

给定聚类数 15 ，使用之前利用 K-Means++ 方式求得的聚类中心坐标作为起始状态，根据样本和聚类中心的欧氏距离，将样本划分到离它最近的聚类中心上，完成一次分类。然后对于每个聚类，利用所有属于这一聚类的样本，重新计算聚类中心的位置（所有数据点的平均值）。重复这一循环操作，直到最终满足终止条件。

具体聚类数量如何确定，聚类中心如何初始化，如何使用特征向量表示图片，如何衡量样本距离和如何设置终止条件，将在下一部分详细讨论。

（三）算法细节处理设计

本部分将具体介绍如何设置算法处理的细节。

1、如何用特征向量表示图片?

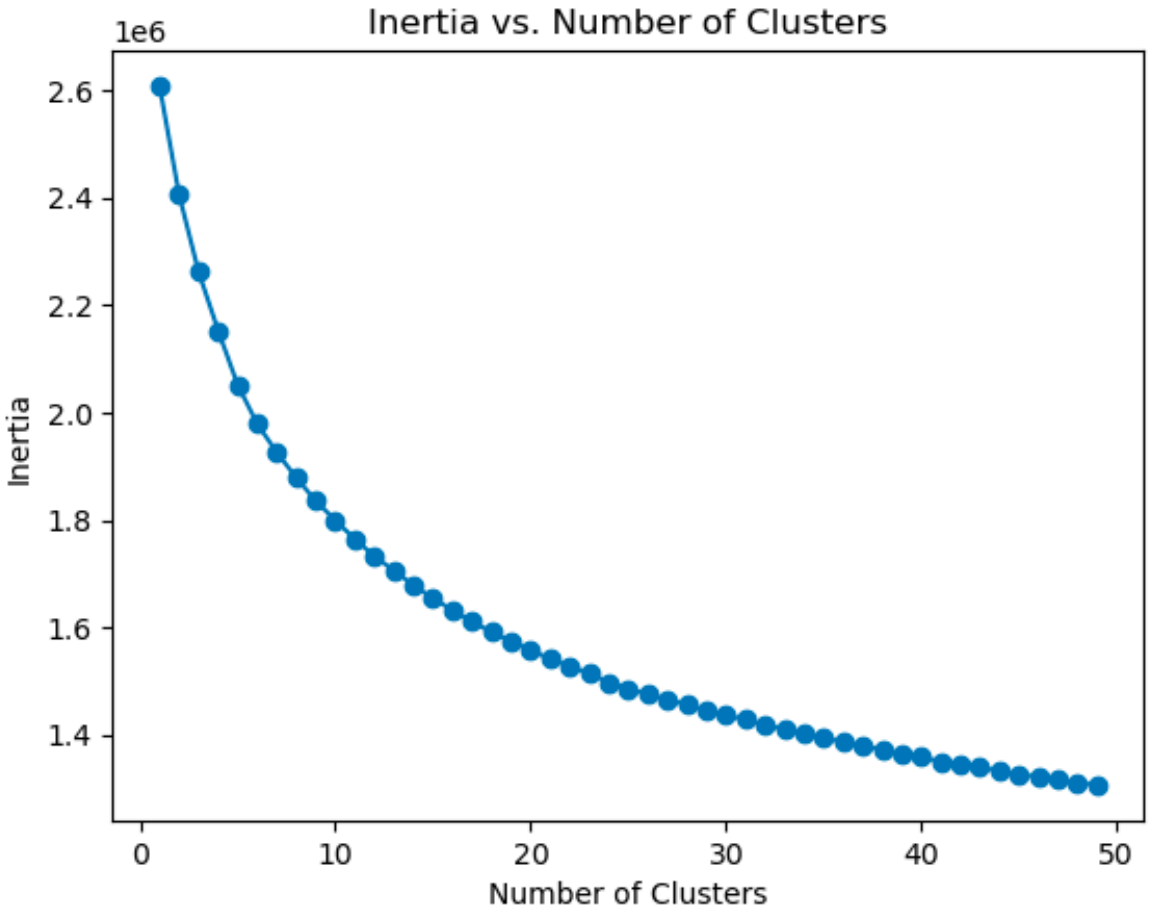
我们把二维像素矩阵展平成一个一维的向量，把像素值的范围缩放到 [-1，1] 之间归一化，防止对距离计算的影响。然后整理成二维数据的形式。

感觉整体维度并不是非常高，所以使用 PCA 等技术来进行降维操作并不是太有必要，因此就没有使用。当然从原理上，使用 PCA 等降维技术是有道理的，因为每个数字一般会经过一些特定的坐标点，所以有一些坐标点可能永远不会被经过（例如位于最角落上的点，可能就不会有多大的影响），使用主成分分析等方式把这些点在降维过程中去除掉，或许可以提升识别的精度。

2、如何确定聚类数量?

综合考虑实现难度和具体效果，我们使用肘部法则来确定聚类的数量。

首先，我们使用主成分分析的方式对数据进行降维（从784维降维成 50维的情况），以便对数据进行简易的快速聚类分析。然后遍历从1到50中所有可能的聚类数量，采用聚类回归的方式进行测试，衡量每个聚类数量对应的聚类误差整体是多少（使用所有样本到自己所属聚类的欧氏距离来定义）。以聚类误差作为纵坐标，聚类数量作为横坐标，绘制图表如下。



观察图表大致可知，从聚类的数量大约大于15开始，随着聚类数量的增加，聚类整体误差不再明显下降，认为 n = 15 是比较明显的“拐点”。选择这一数值作为最优的聚类数量。

在网络上搜索得知，还有轮廓系数和 gap 统计量等方式可以用来确定最优的聚类数量，但是这些方式的实现和计算相对比较复杂，最终还是选择肘部法则确定。

并且结合实际的情况，即 MNIST 数据集是针对数字的分类，认为类别应该至少不少于 10 个，图中的表现也印证了这一点。

3、如何初始化聚类中心?

我们使用 K-Means++ 的方式确定聚类中心，利用在上一步中进行之后的数据再次求解。首先，从给定的特征向量中随机选择一个数据点作为第一个聚类中心。然后，循环执行以下步骤直到选择了k个聚类中心：

- 对于每个数据点，计算它与已选择的所有聚类中心的距离，并选择其中最近的距离。
- 计算每个数据点被选择为下一个聚类中心的概率。这个概率与它到最近的聚类中心的距离成反比，距离越近的点被选择的概率越高。
- 根据计算出的概率，使用`np.random.choice`函数从数据点中选择下一个聚类中心，并将其添加到聚类中心列表中。

这样，经过k次迭代后，我们就会得到k个初始的聚类中心，这些中心点相对于数据的分布更加均匀，有助于K均值算法更快地收敛到最优解。

4、如何衡量样本距离?

可能就直接是欧氏距离，目前还没有想到什么必要去切换使用别的距离。因为不同维度之间应该是对称的，也就是说可能不会有什么大的差别。

当然，我们可以在最终实验分析的部分中尝试一下使用马氏距离，但在这里预估应该不会有太大的差别。

5、如何设置终止条件?

通过在互联网上的搜索，我了解到了三种比较常用的判断方式：

1. 观察整体距离之和是否连续几轮不再显著变化，如果是的话就取开始不再显著变化之前的值
2. 检查聚类中心和上一次迭代相比是否发生了变化，也可以使用上一条的这种几轮之前的衡量方式
3. 或者是样本点的分配是否发生变化，也可以使用上一条的方式

最终，我使用的模型共同使用两种方式进行检查：

1. **整体距离变化是否显著**：如果整体距离（数据点到聚类中心的欧氏距离之和）的变化不超过预先设定的收敛容忍度 `tol`，或者连续5次迭代中距离没有显著变化，则停止迭代。
2. **质心位置是否稳定**：如果新的聚类中心与前一轮迭代的聚类中心非常接近（通过 `np.allclose()` 函数判断），则认为聚类中心的位置已经收敛，就停止迭代。

二、评价模型表现

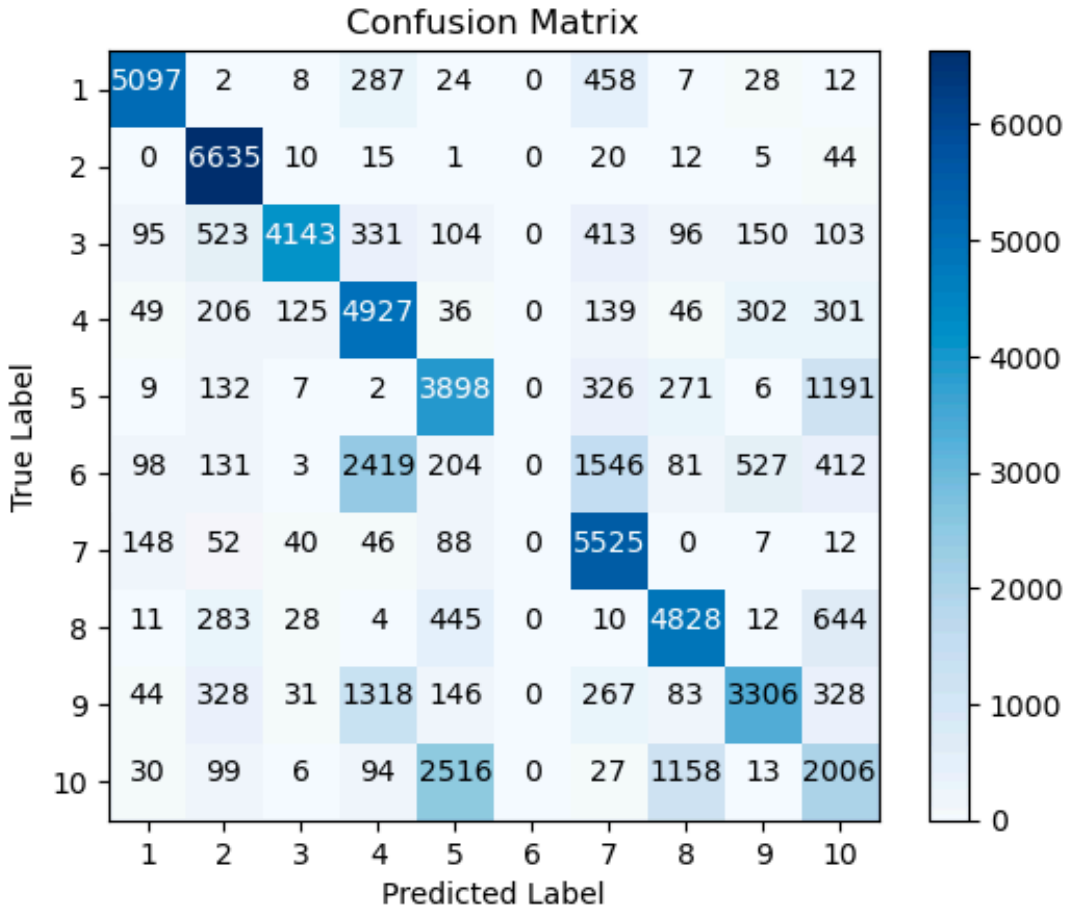
（一）定量评价

总共使用了 15 个聚类进行分类，采用多数投票的方式确定聚类的标签，最终发现 0，1，3，4，7 都对应了两个聚类。

在训练的数据上，最终的预测 accuracy 是67.275%，属于较为不错的成绩。

混淆矩阵具体值和可视化后的热力图如下：

```
Confusion Matrix:
[[5097    2    8  287   24    0  458    7   28   12]
 [   0 6635   10   15    1    0   20   12    5   44]
 [  95  523 4143  331  104    0  413   96  150  103]
 [  49  206  125 4927   36    0  139   46  302  301]
 [   9  132    7    2 3898    0  326  271    6 1191]
 [  98  131    3 2419  204    0 1546   81  527  412]
 [ 148   52   40   46   88    0 5525    0    7   12]
 [  11  283   28    4  445    0   10 4828   12  644]
 [  44  328   31 1318  146    0  267   83 3306  328]
 [  30   99    6   94 2516    0   27 1158   13 2006]]
```



混淆矩阵的热力图标注有误，横轴和纵轴的坐标都应该减一后再具体考虑。从图中可以看出，在大多数数字的识别上整体效果较好，在比较相近的9和4，5和3，7和9上出现了比较多的误判，但整体而言表现尚可。

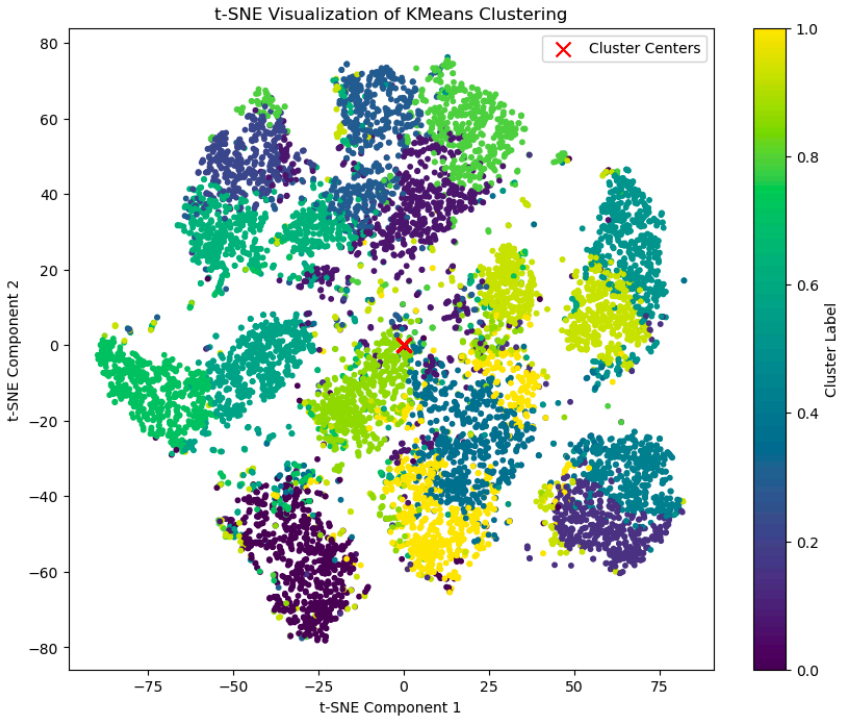
在数据集上的 precision 和 recall 分别是 62.1178% 和 66.1735%，整体而言比较优秀，但仍然有比较大的提升空间。

（二）定性评价

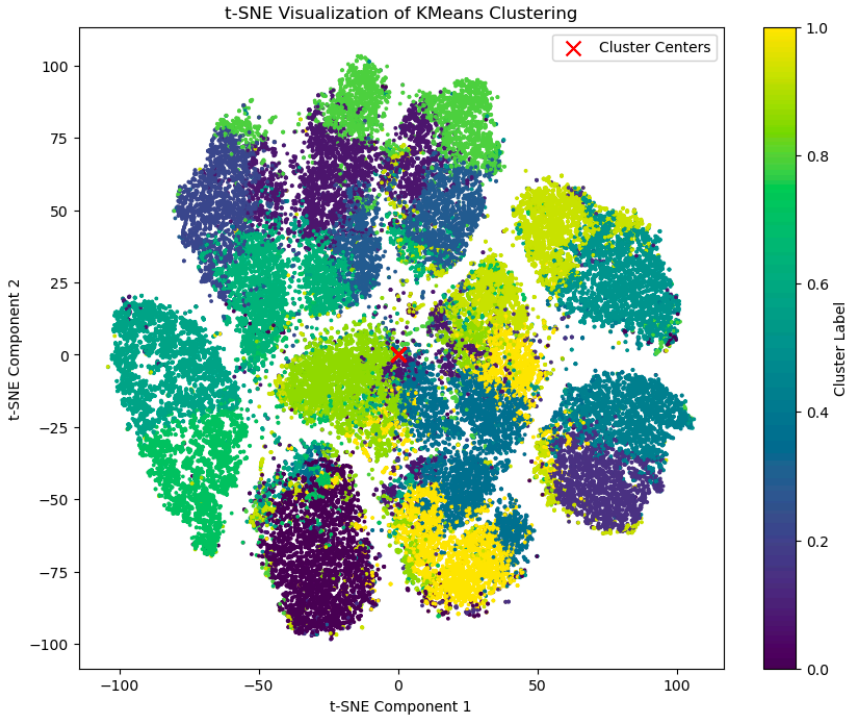
在聚类表现的可视化上，我采用 t-SNE 的方式进行降维，并绘制热力图可视化聚类结果。

为了对比采用部分数据和采用所有数据的区别，我分别绘制了包含最靠前的10000个点的热力图和包含所有点的热力图来比较不同。

包含最靠前的10000个点的热力图的情况如下：

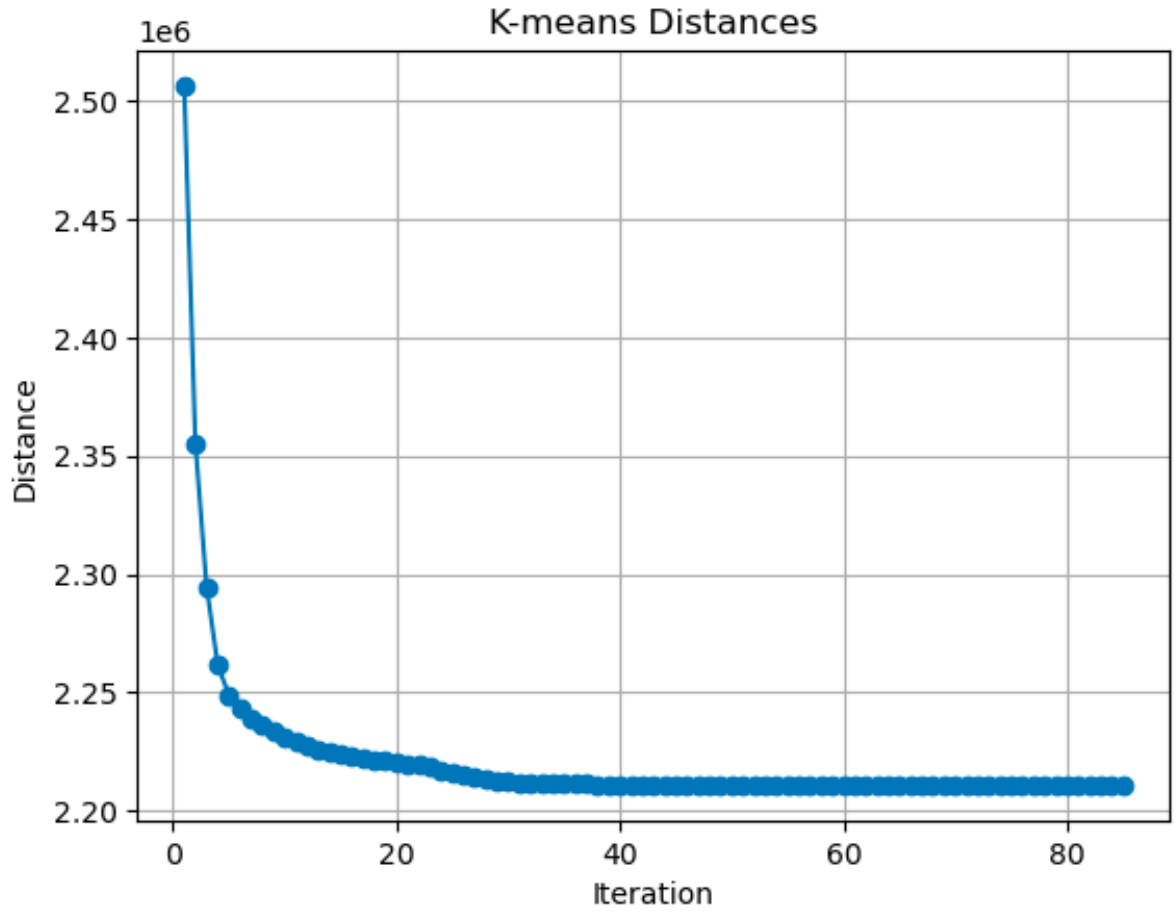


包含所有点的热力图的情况如下：



两者对比可以看出，选取部分点可以比较好地展现整体聚类的分布，但选取所有点的效果也是显著优于部分点的，可以更加清晰地看出不同聚类的重叠和位置关系。

并且，我们可以可视化在 K-Means 寻找聚类的过程中整体的欧氏距离的变化过程，可以发现其实在一开始（前八次迭代）就已经达到了相对比较高的收敛速度，之后收敛的情况非常不明显。在之后的对比测试中，可以放宽一些判断是否收敛的标准，收敛速度达到一定程度就可以停止迭代。



在可视化衡量之外，我们引入轮廓系数来衡量聚类结果紧密度和分离度。轮廓系数的取值范围为[-1, 1]，越接近1表示聚类结果越好，越接近-1表示聚类结果越差。本次聚类的平均轮廓系数为0.06065，属于比较接近中游的轮廓系数，说明聚类的结果比较一般，也符合我们定性和通过热力分布图观察出来的结果。

三、实验分析

（一）实验总结

本次实验尝试使用 K-Means 聚类分类方式实现 MNIST 手写数字分类，我将分为实现的过程中和最终的结果两部分分别总结。

在实现的过程中，我首先关注于算法整体框架的设计，然后去考虑具体细节参数的选择。在实际开始编写、运行代码之前，我主要关注的点是“如何获取更强的性能”，忽略了具体实现的复杂程度以及我是否完全理解这一算法的原理。在具体的实现过程中，我在实际框架构建和代码编写中体会到了有些性能强大的算法的难以理解和应用，转而选择了我能够充分理解的算法，来保证我对框架的掌控，从而有效减少了bug的数量。

在最终的结果上，我们在训练集上取得了67.275%的 accuracy，有着较为不错的识别能力。但在相似数字的判断上有着比较大的提升空间，可能应该在最初初始聚类中心的选取上更加仔细地设计，将针对相似数字的优化调整体现在初始值的选取中。

（二）实验细节分析

在这一部分，我们尝试对比研究某些参数对聚类性能的影响。为了加快对比研究的效率，我们将整体的数据从60000减少为2000，这样在保留了适当数量的样本的同时可以大大加速计算的效率。

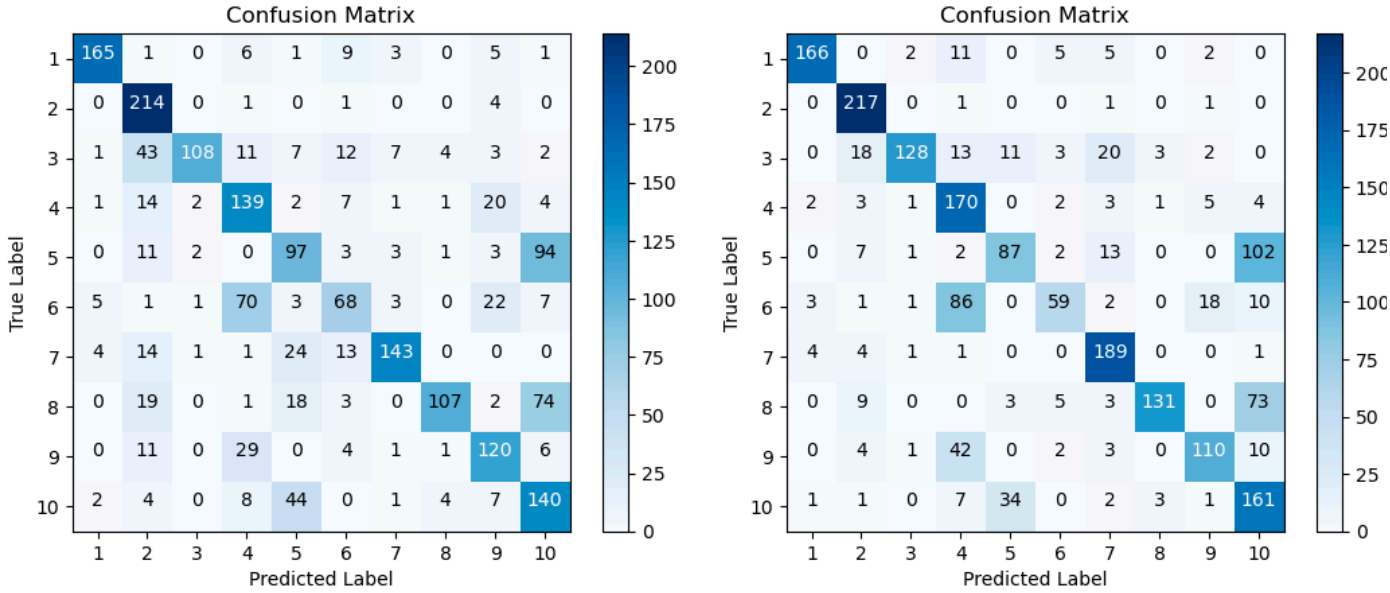
在这一部分中，我们将除了样本减少为2000，其他项相比于第二部分的算法实现中完全不改变的结果作为对照组，和修改参数后的结果进行比较。

1、不同K的取值的影响

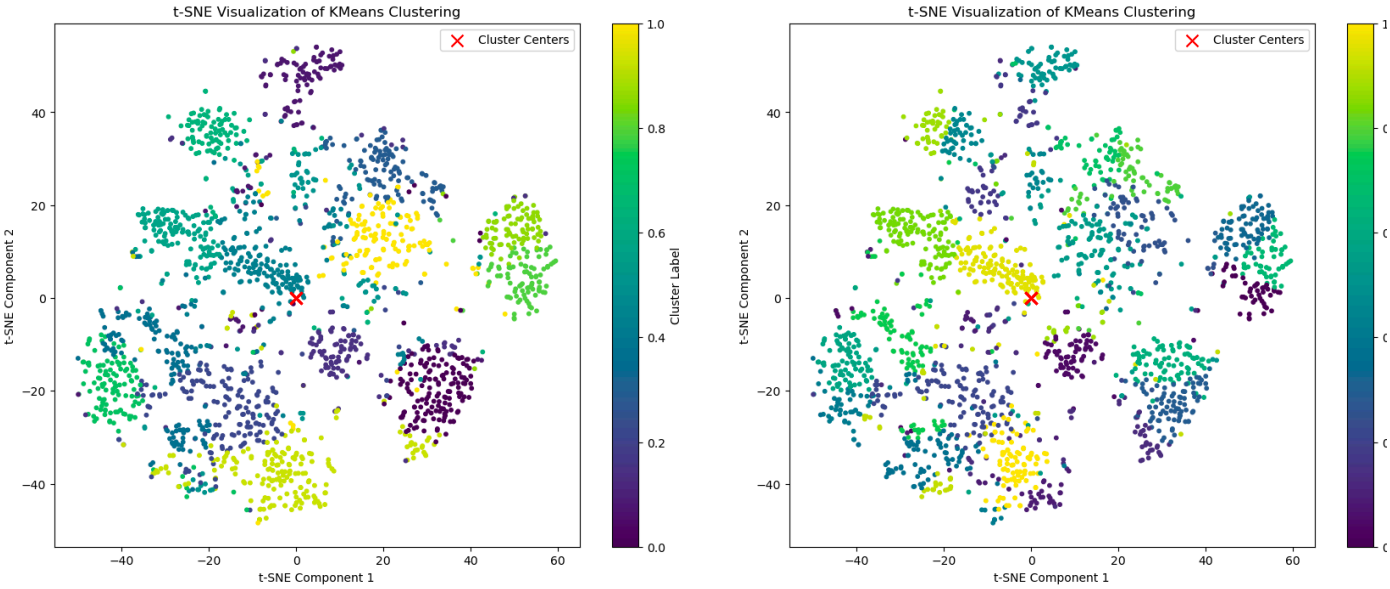
将k从15修改为25后，观察 accuracy，混淆矩阵和热力图的变化。

k为15时，accuracy 为0.6505；k为25时，accuracy 为0.709。可以认为，在目前的取值范围内，当k提升时可以较为显著有效地提升聚类的分类性能。

左图是k=15时的混淆矩阵，右图是k=25时的混淆矩阵。从中我们可以观察得到，两者基本上都有着比较好的性能，随着k的提升，对于数字的识别正确率会变高，但是相似数字被识别错误的概率也会提升，不过把并不相似的数字识别错了的问题会显著下降。



左图是k=15的热力图，右图是k=25的热力图。从中可以看出，随着聚类的总数增加，分类的效果有变得更加显著，多个聚类大面积地重叠在一起的现象右图比左图少了很多。

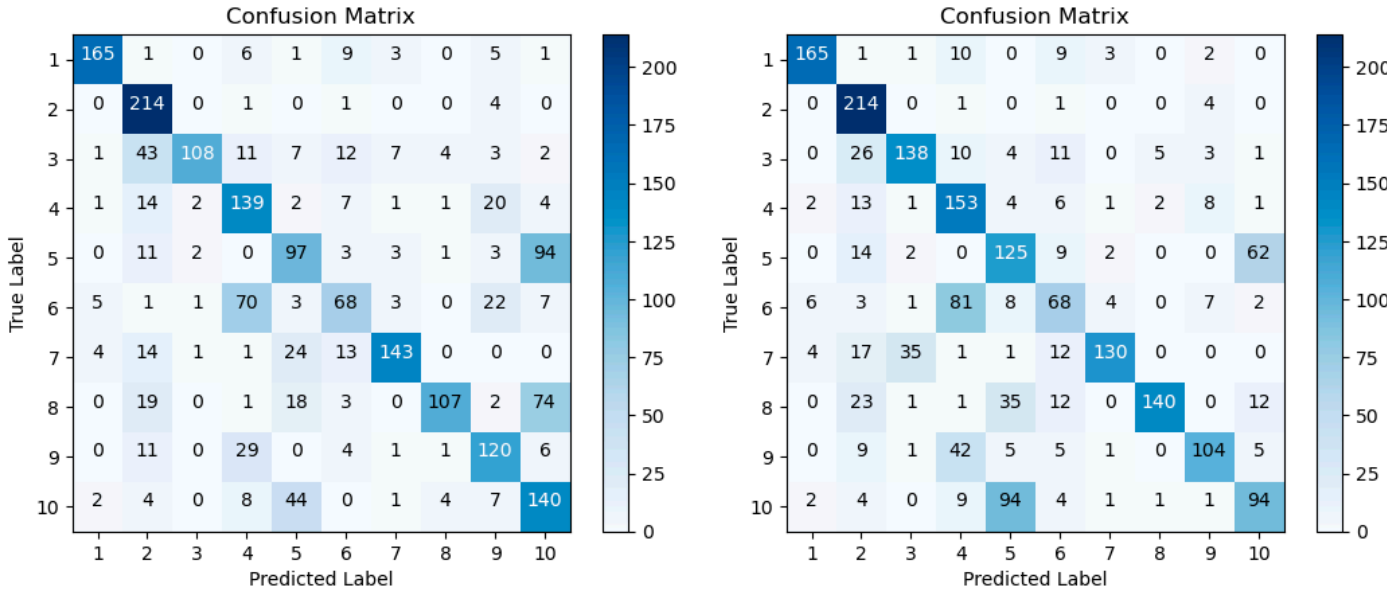


2、初始化聚类中心位置的影响

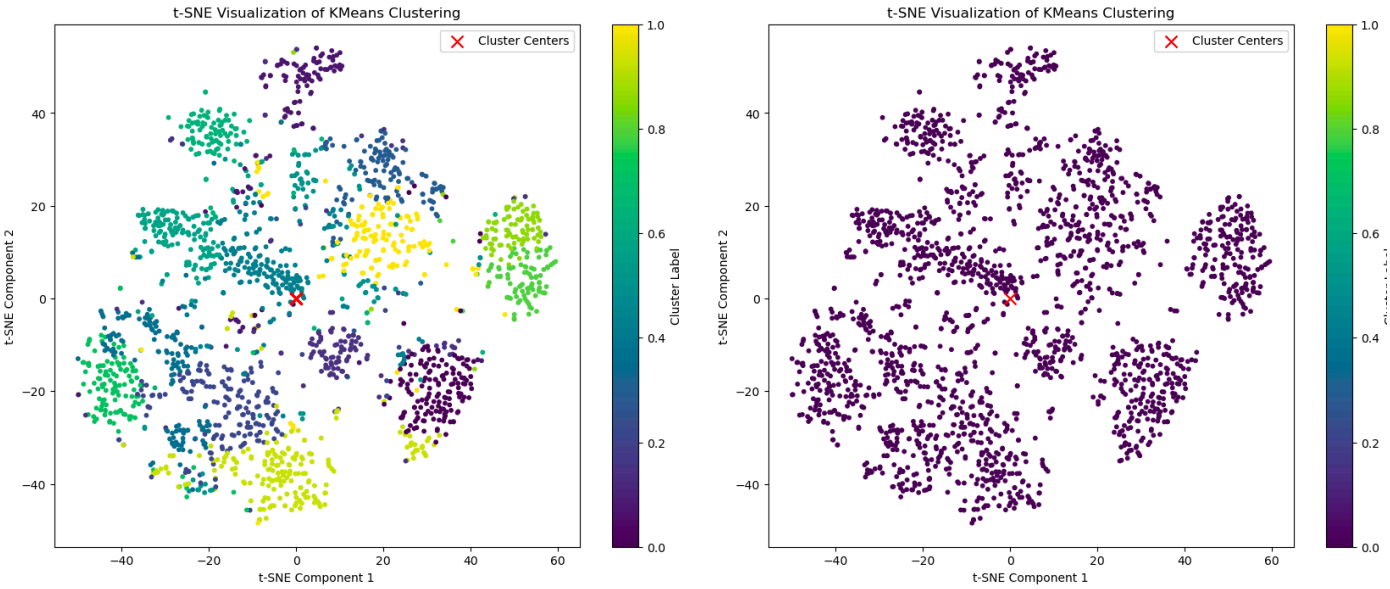
原先的方案中，我们使用 K-Means++ 的方式来选取。在本次探究中，我们把初始的聚类中心修改为随机值，来观察聚类性能的变化。

使用 K-Means++ 的方式来选取时，accuracy 为0.6505；而将初始值修改为随机值之后，accuracy 为0.6655。可以认为，在目前的取值范围内，K-Means++ 的性能并不是太好，接近随机选取的结果。

左图是k=15时的混淆矩阵，右图是k=25时的混淆矩阵。从中发现，把聚类中心修改为随机值之后，整体的效果似乎变化不大，没有什么比较显著的改变。



左图是k=15的热力图，右图是k=25的热力图。由于随机值的计算问题，第二张图只进行了一轮迭代就停止了，因而热力图的比较价值不大。



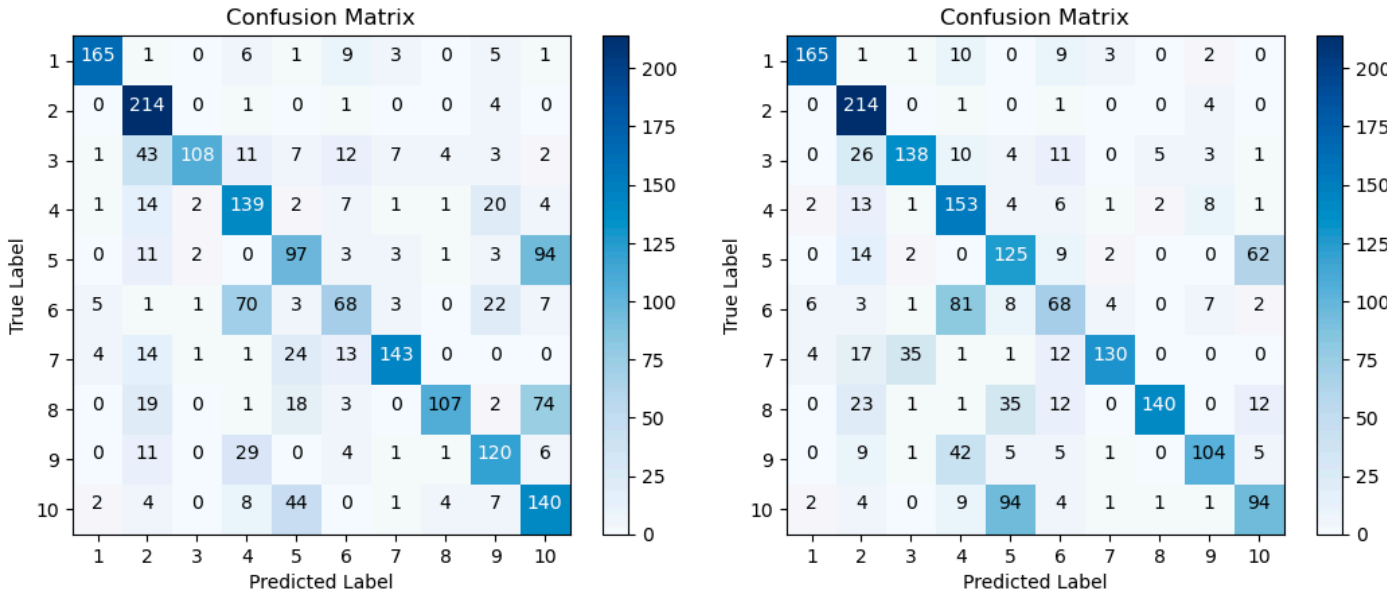
整体而言，把选取样本中心的方式改变为纯粹的随机选取之后，性能反而没有什么很大的改变，说明原先 K-Means++ 的选取方式的实现存在一些问题。

3、样本距离衡量方式影响

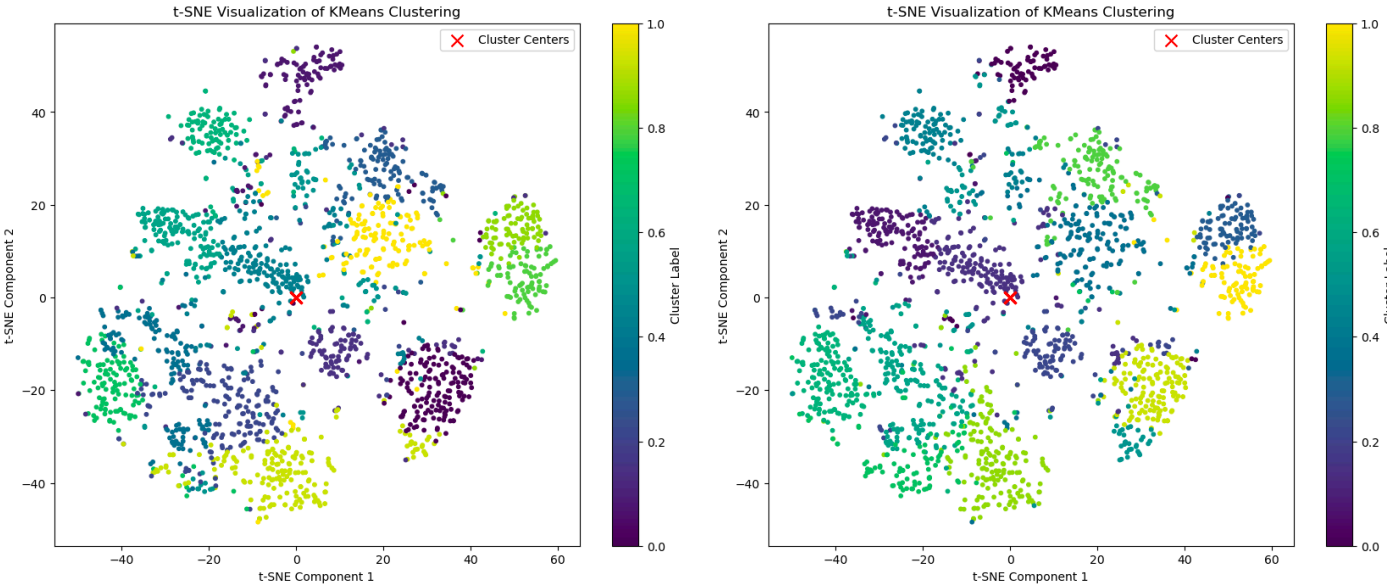
原先我们一直采用欧氏距离来衡量样本和聚类中心的距离，我们可以将欧氏距离修改为马氏距离。

使用欧氏距离时，accuracy 为0.6505；使用马氏距离时，accuracy 为0.6655。可以认为，在目前的取值范围内，改变距离的计算方式对于聚类分类器的性能没有比较显著的提升。

左图是采用欧氏距离时的混淆矩阵，右图是采用马氏距离时的混淆矩阵。对比感觉，采用马氏距离似乎提升了预测的不稳定性，并且主要是增加了把相似数字弄混的可能性。



左图是采用欧氏距离的热力图，右图是采用马氏距离的热力图。感觉采用不同的距离计算方式并没有显著改变最终聚类的分布，整体的分布情况依然极度相似。



整体而言，感觉把欧氏距离的计量方式修改为马氏距离的计量方式，无法显著改变聚类的分类性能和整体最终的结果。

4、可视化结果分析

可视化主要是为了提供相比数值而言更加全面和直观的展示，本次作业中主要通过热力图、混淆矩阵和训练中总距离的变化折线图来实现可视化。

通过对迭代中总距离的变化绘制折线图，我们可以更加清晰地发现存在明显的“山底碎石图”效果，意识到在迭代中应该把停止训练的点设置的更早，或者把收敛的标准制定的更加宽松。

通过混淆矩阵，我们可以更加明确的发现是在哪些数字组合上经常弄混，可以发现犯错是集中在一些组合，还是比较均匀的分散。如果将来会进一步调整优化模型的话，可以通过混淆矩阵可视化热力图观察的结果作为入手点。

整体聚类分布的热力图可以让我们更加直观地观察到聚类整体的分布，可以通过点簇的重合程度来判断聚类是否被有效区分了。

5、其他聚类方法的使用

除了课程中详细介绍的 K-Means 聚类之外，还有一些其他的聚类方法也可以在本次大作业中应用，例如高斯混合模型。

高斯混合模型假设数据由多个高斯分布组成，每个高斯分布代表一个聚类簇。在MNIST分类问题中，我们可以假设每个数字类别对应一个高斯分布。在训练中，模型会使用 MLE 来计算出高斯分布的参数，并在之后使用这些高斯分布的组合计算出样本属于每个高斯分布的概率，将样本分配到概率最大的那个高斯分布所代表的类别中。