K-Means Clustering Algorithm

Tian Tian 2021011048

I. Implementation of the K-Means Clustering Algorithm

(1) Task Analysis

In this assignment, we use the K-Means clustering algorithm to classify the training set portion of the MNIST dataset. This assignment focuses solely on clustering and analyzing the training set, without testing the results on the test set.

(2) Overall Algorithm Logic Design

Given 15 clusters, we use the cluster center coordinates obtained from the K-Means++ method as the initial state. Based on the Euclidean distance between samples and cluster centers, we assign samples to the nearest cluster center, completing one iteration of classification. Then, for each cluster, we recalculate the position of the cluster center using all samples belonging to that cluster (by computing the mean of all data points). This loop operation repeats until the termination condition is satisfied.

Details on how to determine the number of clusters, initialize cluster centers, represent images using feature vectors, measure sample distances, and set termination conditions will be discussed in the next section.

(3) Detailed Algorithm Processing Design

This section will introduce the details of how the algorithm handles specific processes.

1. How to Represent Images Using Feature Vectors?

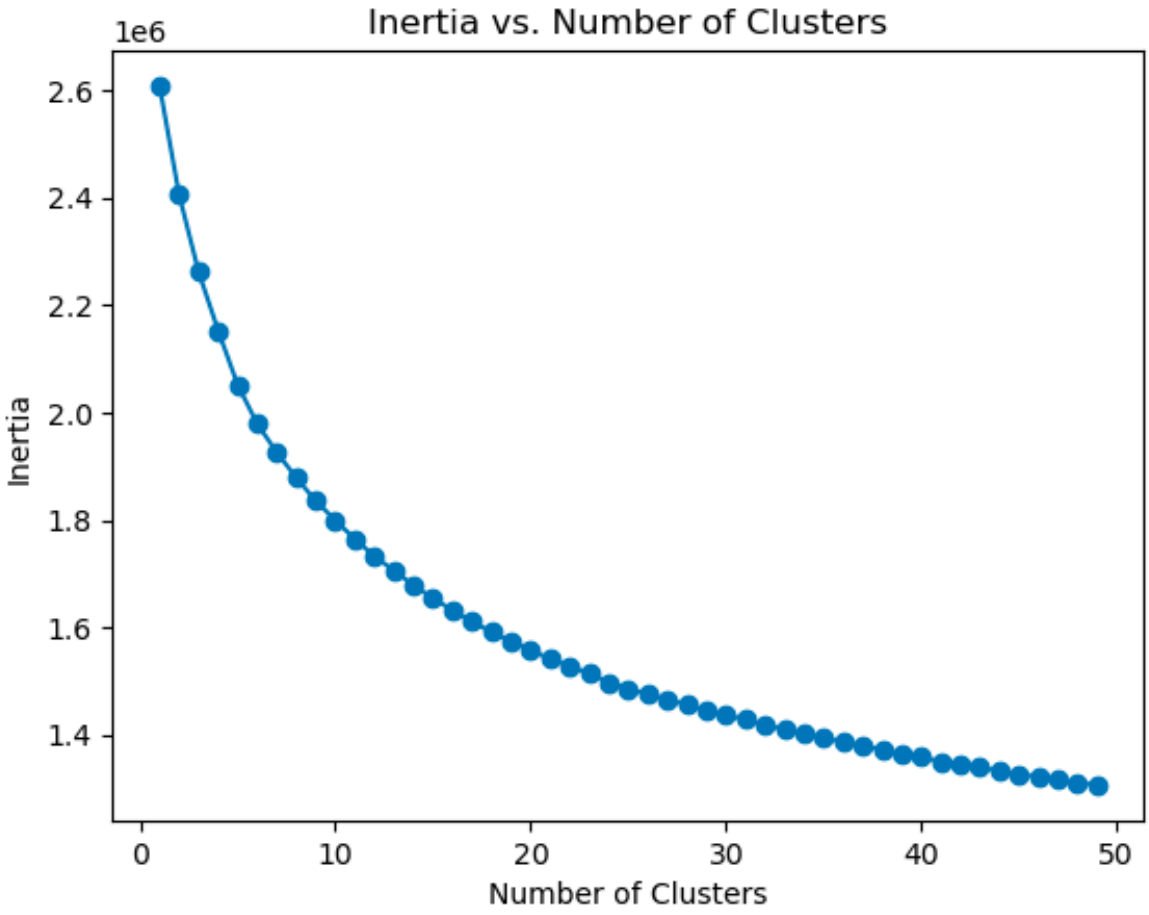
We flatten the 2D pixel matrix into a 1D vector and normalize the pixel values to the range [-1, 1] to prevent them from affecting distance calculations. We then organize this into a 2D data format.

The overall dimensionality does not seem very high, so using techniques like PCA for dimensionality reduction does not appear necessary, and thus, we did not use it. However, in principle, using PCA or other dimensionality reduction techniques is reasonable because each digit typically passes through certain specific coordinate points, meaning some points may never be passed (for example, points located in the corners might have minimal impact). Using principal component analysis to remove these points during dimensionality reduction might improve recognition accuracy.

2. How to Determine the Number of Clusters?

Considering both implementation difficulty and effectiveness, we use the elbow method to determine the number of clusters.

First, we use principal component analysis (PCA) to reduce the dimensionality of the data (from 784 dimensions to 50 dimensions) for quick and simple clustering analysis. Then, we traverse all possible cluster numbers from 1 to 50, and use clustering regression to test and measure the overall clustering error for each cluster number (defined by the Euclidean distance of all samples to their respective clusters). Plotting clustering error as the y-axis and the number of clusters as the x-axis, we get the following chart.



Observing the chart, we can see that the overall clustering error no longer significantly decreases when the number of clusters is greater than about 15. Thus, n = 15 is a clear "elbow point" and is chosen as the optimal number of clusters.

According to online searches, methods such as silhouette coefficient and gap statistic can also be used to determine the optimal number of clusters. However, these methods are relatively complex in implementation and calculation, so we ultimately choose the elbow method.

Additionally, considering the practical situation where the MNIST dataset categorizes digits, the number of categories should be at least 10, and this is also confirmed by the chart.

3. How to Initialize Cluster Centers?

We use the K-Means++ method to determine cluster centers, using the data obtained in the previous step for further solving. First, we randomly select a data point from the given feature vectors as the first cluster center. Then, we repeat the following steps until k cluster centers are chosen:

- For each data point, calculate its distance to all chosen cluster centers and select the minimum distance.
- Calculate the probability of each data point being chosen as the next cluster center. This probability is inversely proportional to its distance to the nearest cluster center—the closer the point, the higher the probability of being chosen.
- Based on the calculated probabilities, use the `np.random.choice` function to select the next cluster center from the data points and add it to the cluster center list.

After k iterations, we obtain k initial cluster centers. These centers are more evenly distributed relative to the data, helping the K-Means algorithm converge to an optimal solution faster.

4. How to Measure Sample Distances?

Currently, we use Euclidean distance and see no necessity to switch to another distance metric since the dimensions should be symmetric, meaning there wouldn't be significant differences.

Of course, we can try using the Mahalanobis distance in the final experimental analysis, but it is expected that the difference will not be substantial.

5. How to Set Termination Conditions?

From online searches, I learned about three commonly used methods:

1. Observe whether the overall distance sum no longer changes significantly for several consecutive iterations. If so, use the value from before the changes became insignificant.
2. Check whether the cluster centers have changed compared to the previous iteration. This can also use the previous method of measuring for several iterations.
3. Check whether the sample point assignments have changed. This can also use the previous method.

Ultimately, the model I use employs two methods for checking:

1. **Significant Change in Overall Distance**: If the change in the overall distance (sum of Euclidean distances from data points to their cluster centers) does not exceed a pre-set convergence tolerance `tol`, or if the distance does not significantly change for five consecutive iterations, stop the iterations.
2. **Stability of Centroid Positions**: If the new cluster centers are very close to the previous iteration's cluster centers (judged by the `np.allclose()` function), the positions of the cluster centers are considered to have converged, and the iterations are stopped.

II. Evaluating Model Performance

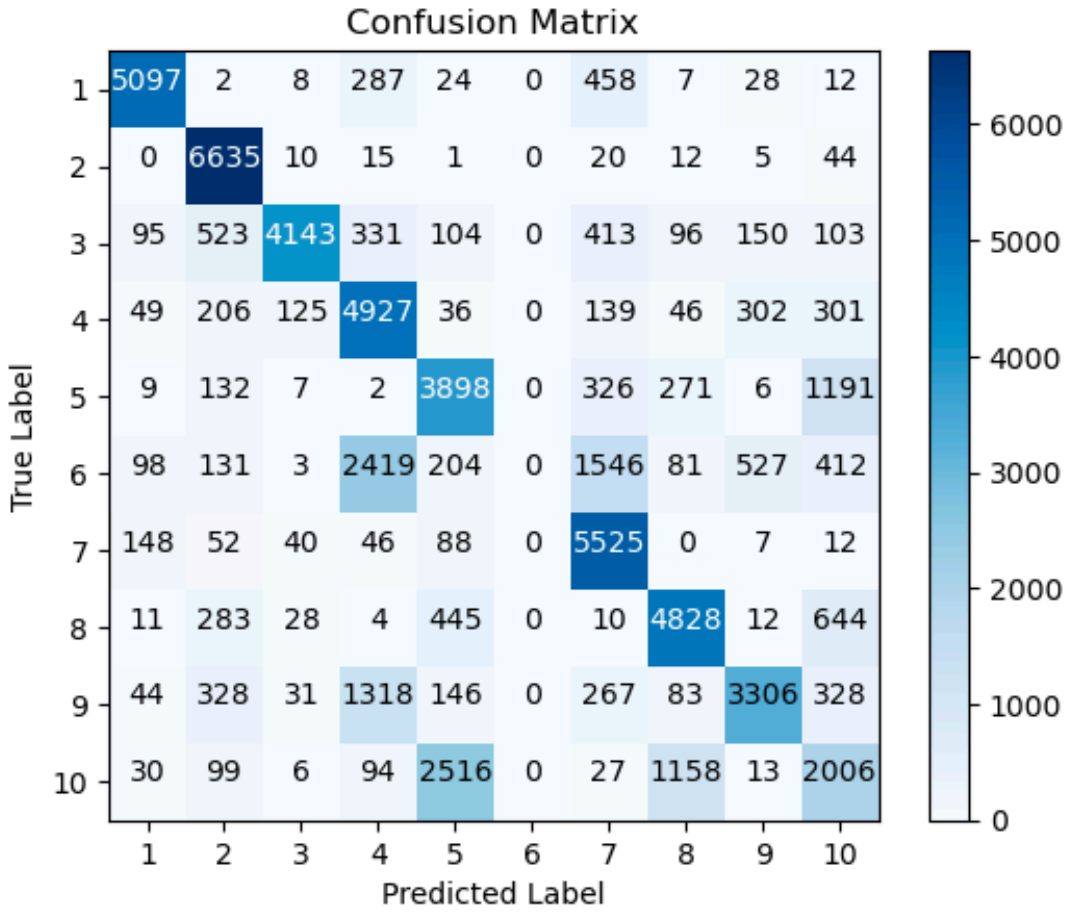
(1) Quantitative Evaluation

Using a total of 15 clusters for classification, we determine cluster labels through majority voting, and find that digits 0, 1, 3, 4, and 7 each correspond to two clusters.

On the training data, the final prediction accuracy is 67.275%, which is relatively good.

The specific values of the confusion matrix and the visualized heatmap are as follows:

```
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]]
```



There is a mistake in the heatmap labeling of the confusion matrix; the coordinates on both axes should be considered one less than shown. From the chart, it can be seen that overall, the recognition performance is good for most digits. However, there are frequent misclassifications between similar digits such as 9 and 4, 5 and 3, and 7 and 9, but overall, the performance is acceptable.

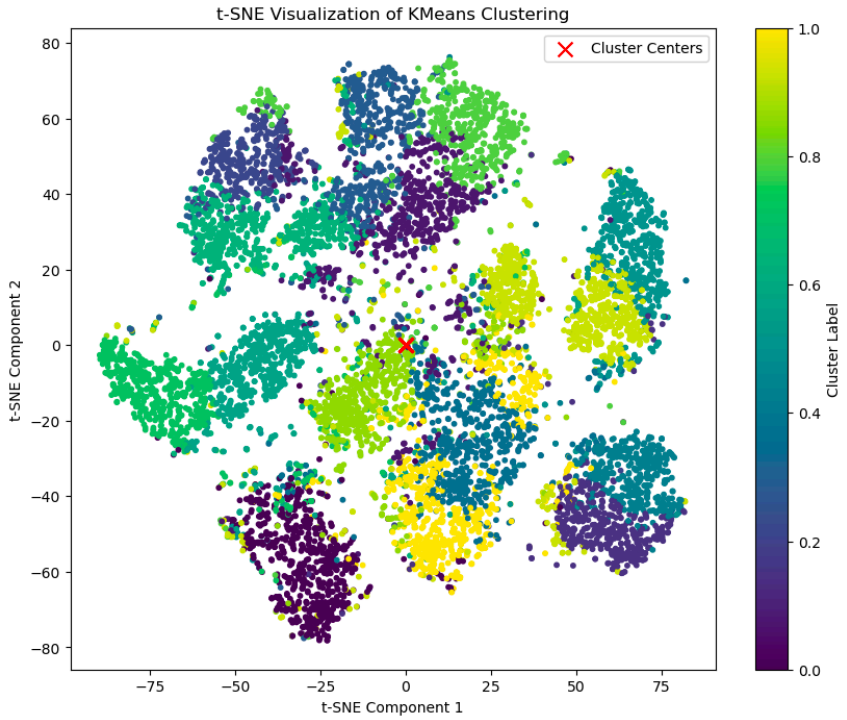
The precision and recall on the dataset are 62.1178% and 66.1735%, respectively. Overall, the results are relatively good, but there is still room for improvement.

(2) Qualitative Evaluation

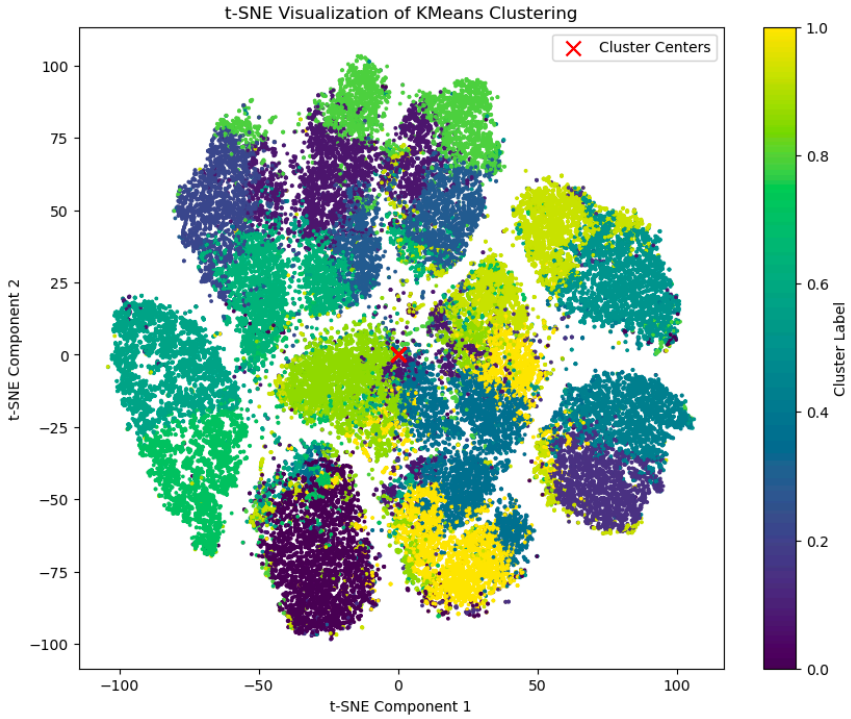
For visualizing the clustering performance, I used t-SNE for dimensionality reduction and plotted heatmaps to visualize the clustering results.

To compare the difference between using a subset of the data and using all the data, I plotted heatmaps for the top 10,000 points and for all points to compare the differences.

The heatmap for the top 10,000 points is as follows:

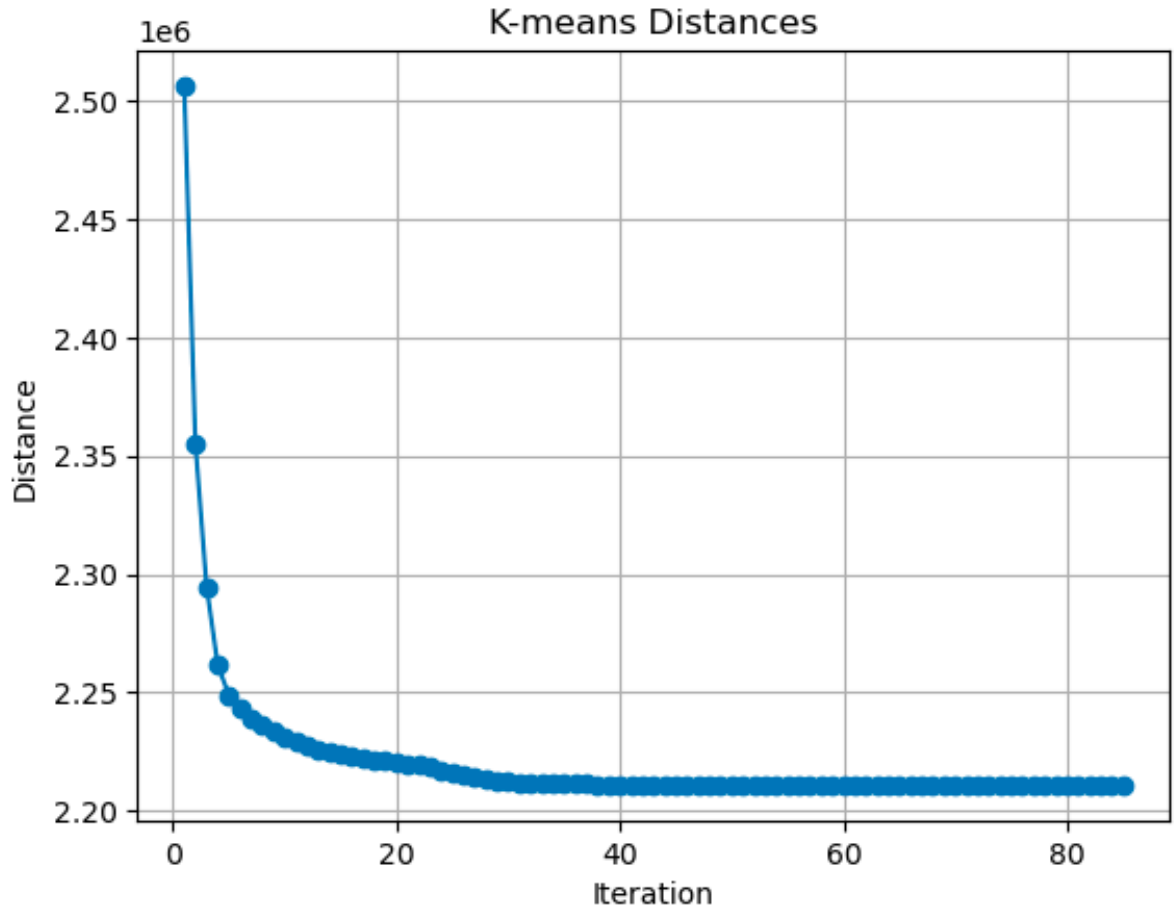


The heatmap for all points is as follows:



Comparing the two, it can be seen that selecting a subset of points can better display the overall clustering distribution, but using all points is significantly better than using a subset, as it more clearly shows the overlap and positional relationships of different clusters.

Additionally, we can visualize the process of the overall Euclidean distance change during the K-Means clustering. It can be observed that convergence reaches a relatively high speed initially (within the first eight iterations), and subsequent convergence is not very significant. In future tests, the criteria for determining convergence can be relaxed, and iterations can be stopped once the convergence speed reaches a certain level.



Beyond visualization, we introduce the silhouette coefficient to measure the tightness and separation of the clustering results. The silhouette coefficient ranges from [-1, 1], with values closer to 1 indicating better clustering results, and values closer to -1 indicating worse clustering results. The average silhouette coefficient for this clustering is 0.06065, which is relatively mediocre, aligning with our qualitative observations and the results observed from the heatmap distribution.

III. Experimental Analysis

(1) Experimental Summary

In this experiment, we attempted to classify MNIST handwritten digits using the K-Means clustering method. The summary is divided into two parts: the implementation process and the final results.

During the implementation process, I first focused on the overall framework design of the algorithm and then considered the selection of specific parameters. Before starting to write and run the code, my primary concern was "how to achieve better performance," neglecting the complexity of implementation and whether I fully understood the principles of the algorithm. During the actual implementation, I realized the difficulty of understanding and applying some high-performance algorithms and chose algorithms that I could fully comprehend. This ensured my control over the framework, effectively reducing the number of bugs.

In the final results, we achieved an accuracy of 67.275% on the training set, which shows relatively good recognition capability. However, there is considerable room for improvement in distinguishing similar digits. A more careful design in the initial selection of cluster centers could potentially optimize this aspect.

(2) Detailed Experimental Analysis

In this section, we attempt to study the impact of certain parameters on clustering performance. To speed up the comparative study, we reduce the overall data from 60,000 to 2,000, retaining an adequate number of samples while significantly accelerating computation.

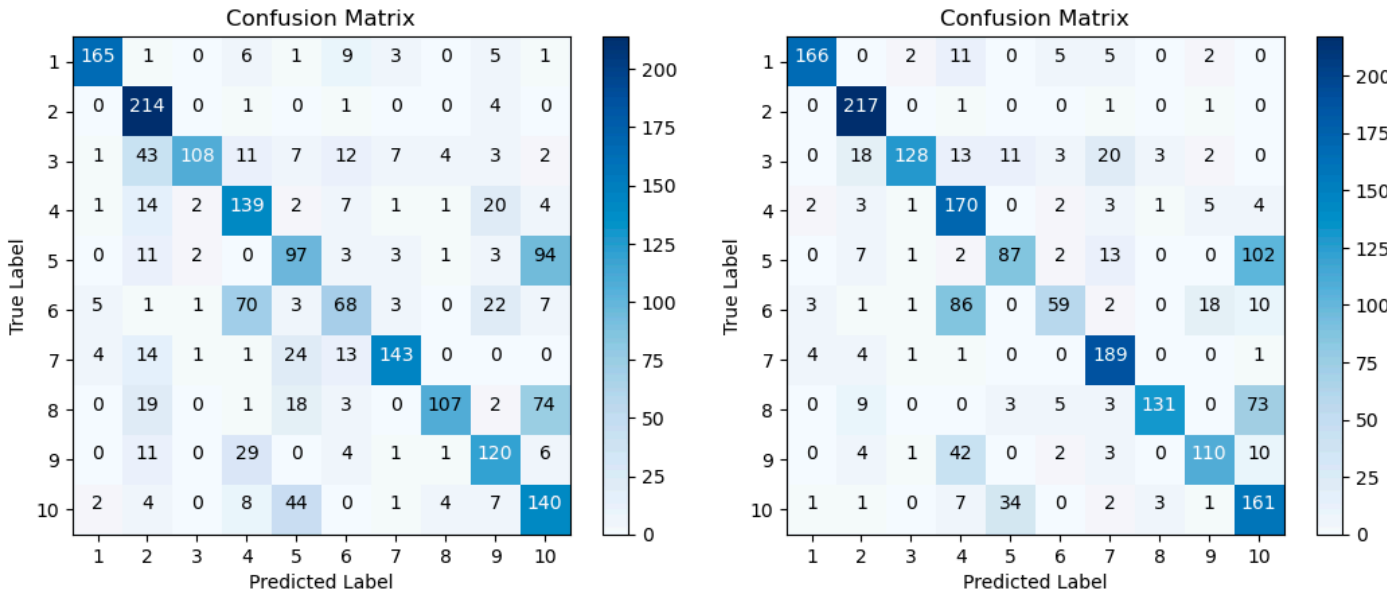
Here, we compare the results by keeping all parameters the same as in the algorithm implementation in the second section, except for reducing the sample size to 2,000. This serves as the control group to compare with the modified parameters.

1. Impact of Different Values of K

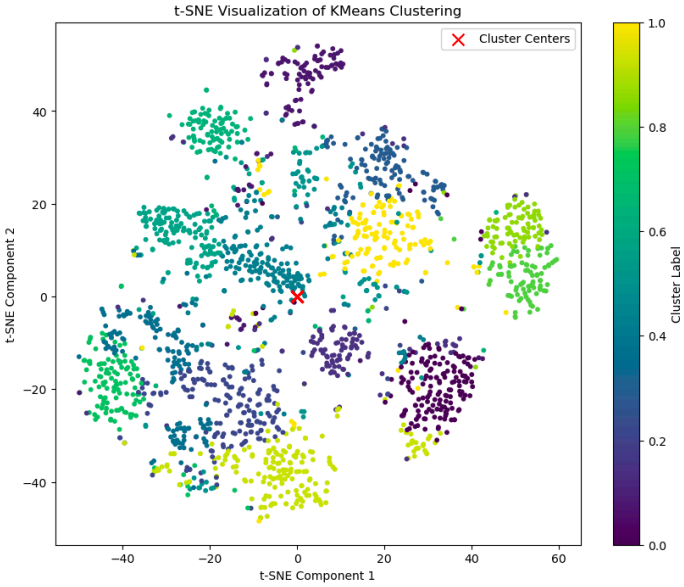
We change K from 15 to 25 and observe the changes in accuracy, confusion matrix, and heatmap.

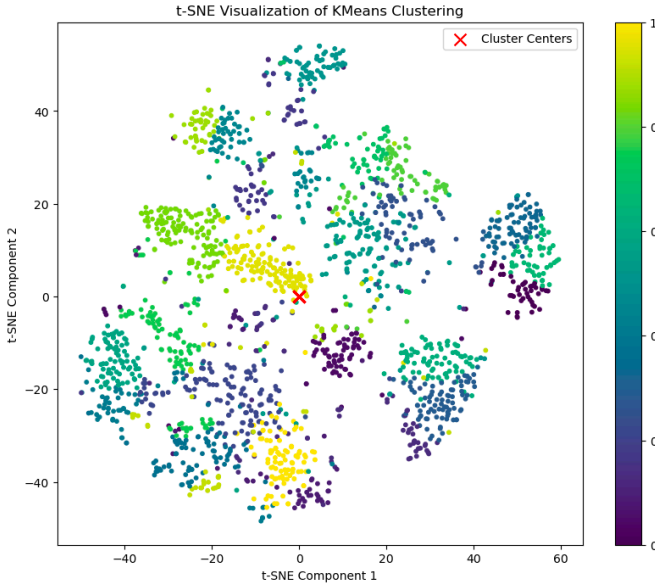
When K is 15, the accuracy is 0.6505; when K is 25, the accuracy is 0.709. It can be concluded that within the current range, increasing K can significantly and effectively improve the clustering classification performance.

The left image shows the confusion matrix for K=15, and the right image shows the confusion matrix for K=25. It can be observed that both have relatively good performance. With the increase in K, the recognition accuracy for digits improves, but the probability of misclassifying similar digits also increases. However, the probability of misclassifying dissimilar digits decreases significantly.



The left image shows the heatmap for K=15, and the right image shows the heatmap for K=25. It can be seen that with the increase in the total number of clusters, the classification effect becomes more significant, and the phenomenon of multiple clusters overlapping greatly reduces in the right image compared to the left.



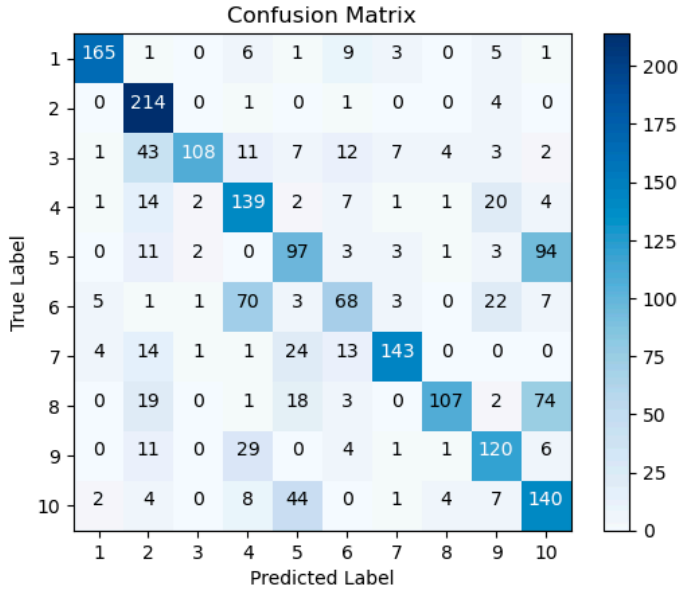


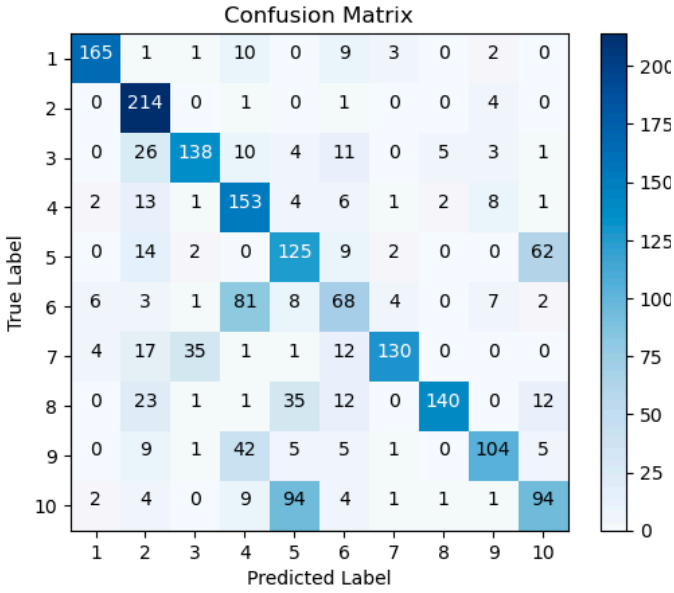
2. Impact of Initial Cluster Center Positions

In the original plan, we used the K-Means++ method to select initial cluster centers. In this exploration, we change the initial cluster centers to random values to observe the change in clustering performance.

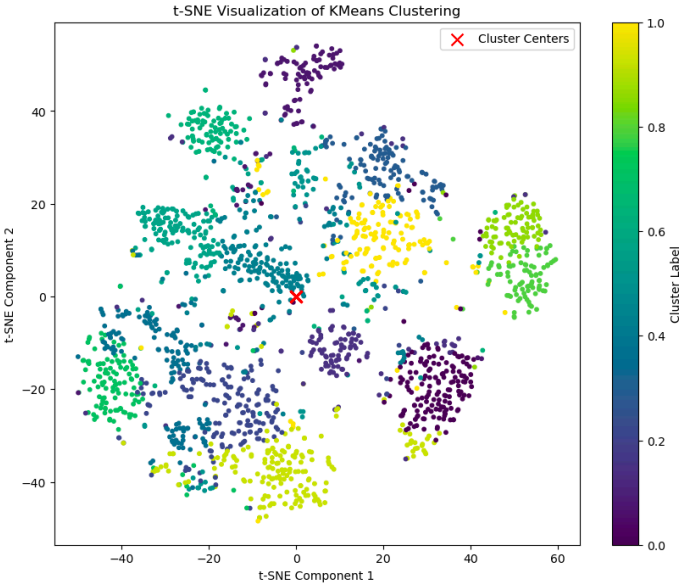
When using the K-Means++ method, the accuracy is 0.6505; when the initial values are changed to random values, the accuracy is 0.6655. This suggests that within the current range, the performance of K-Means++ is not significantly better, close to the result of random selection.

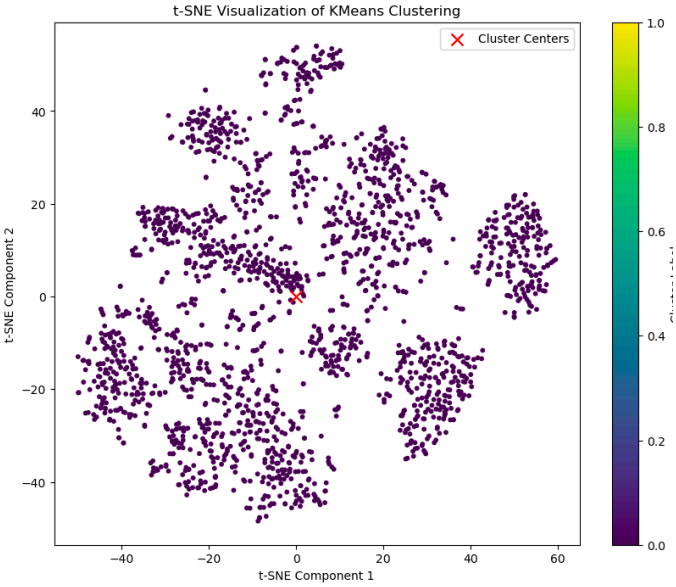
The left image shows the confusion matrix for K=15, and the right image shows the confusion matrix for K=25. After changing the cluster centers to random values, the overall effect does not seem to change significantly.





The left image shows the heatmap for K=15, and the right image shows the heatmap for K=25. Due to the computation issue with random values, the second image stopped after one iteration, making the heatmap comparison less valuable.





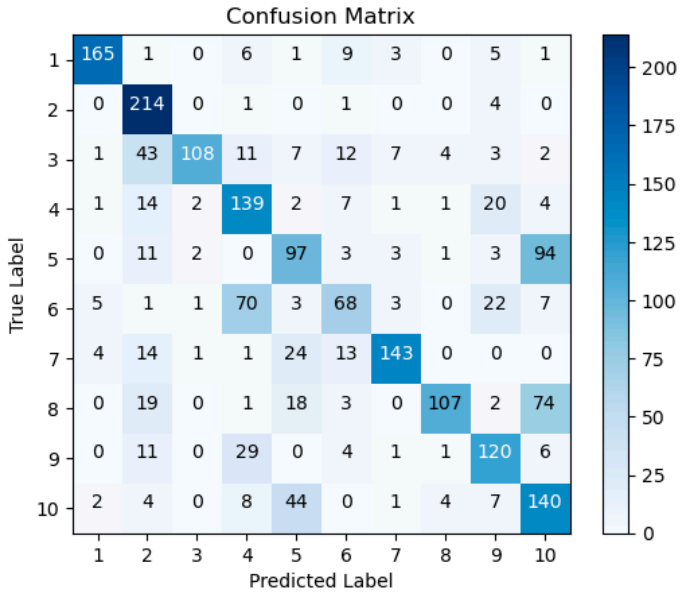
Overall, changing the method for selecting sample centers to pure random selection did not significantly change the performance, indicating potential issues with the implementation of the original K-Means++ selection method.

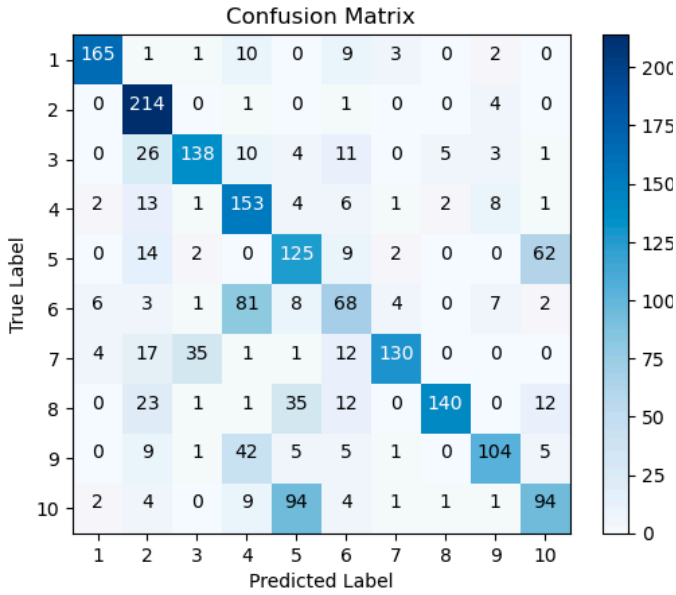
3. Impact of Distance Metric

Initially, we used Euclidean distance to measure the distance between samples and cluster centers. We can change this to Mahalanobis distance.

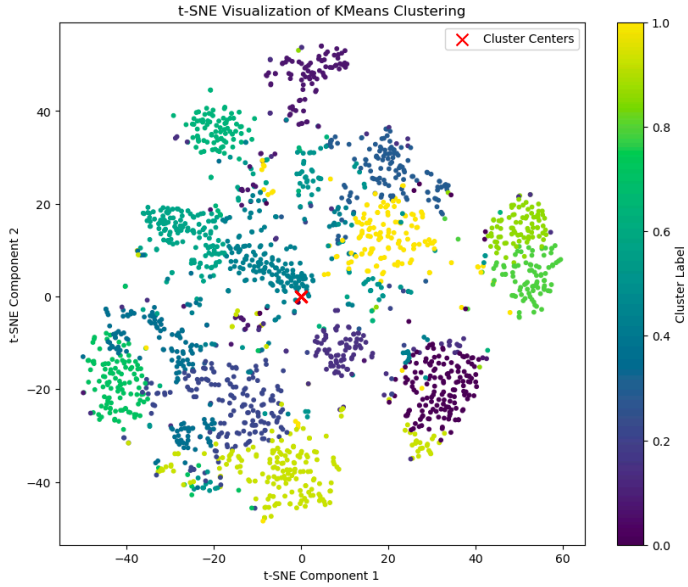
When using Euclidean distance, the accuracy is 0.6505; when using Mahalanobis distance, the accuracy is 0.6655. This suggests that within the current range, changing the distance calculation method does not significantly improve the clustering classifier's performance.

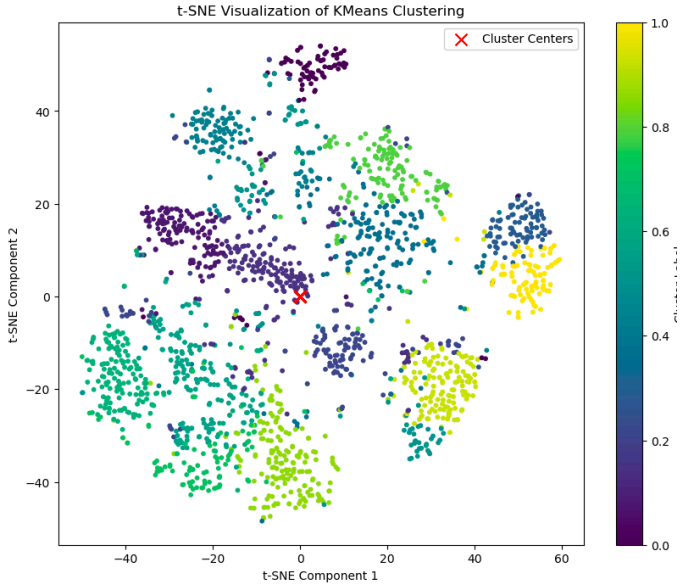
The left image shows the confusion matrix using Euclidean distance, and the right image shows the confusion matrix using Mahalanobis distance. It appears that using Mahalanobis distance increases the instability of predictions, mainly increasing the probability of confusing similar digits.





The left image shows the heatmap using Euclidean distance, and the right image shows the heatmap using Mahalanobis distance. Changing the distance calculation method does not significantly alter the final clustering distribution, as the overall distribution remains very similar.





Overall, changing the distance metric from Euclidean to Mahalanobis does not significantly alter the classification performance and overall final results.

4. Visualization Results Analysis

Visualization is mainly used to provide a more comprehensive and intuitive display compared to numerical values. In this assignment, we used heatmaps, confusion matrices, and line charts of the total distance change during training for visualization.

By plotting the total distance change during iterations, we can more clearly observe the significant "bottoming-out effect" and realize that the stopping point for training should be set earlier, or the convergence criteria should be more relaxed.

Using confusion matrices, we can more clearly identify which digit combinations are frequently confused, and determine whether errors are concentrated in certain combinations or evenly distributed. If further model optimization is planned, the results observed from the confusion matrix heatmap can serve as a starting point.

The overall clustering distribution heatmap allows us to intuitively observe the overall clustering distribution and judge whether clusters are effectively distinguished based on the degree of overlap among point clusters.

5. Using Other Clustering Methods

Besides the K-Means clustering method detailed in the course, other clustering methods, such as the Gaussian Mixture Model (GMM), can also be applied in this assignment.

The Gaussian Mixture Model assumes that data is composed of multiple Gaussian distributions, with each Gaussian distribution representing a cluster. In the MNIST classification problem, we can assume that each digit category corresponds to a Gaussian distribution. During training, the model uses Maximum Likelihood Estimation (MLE) to compute the parameters of the Gaussian distributions and later uses these Gaussian combinations to calculate the probability of samples belonging to each Gaussian distribution, assigning samples to the category represented by the Gaussian distribution with the highest probability.