K-means Clustering Algorithm for MNIST Digits

Vianne Gao

1. **Run the K-means algorithm once with K = 10 (the "true" number of clusters) and initialization of random. Plot the K-means objective (SSE value) as a function of iteration, and verify that it monotonically decreases. Do the same thing with the initialization of other.**

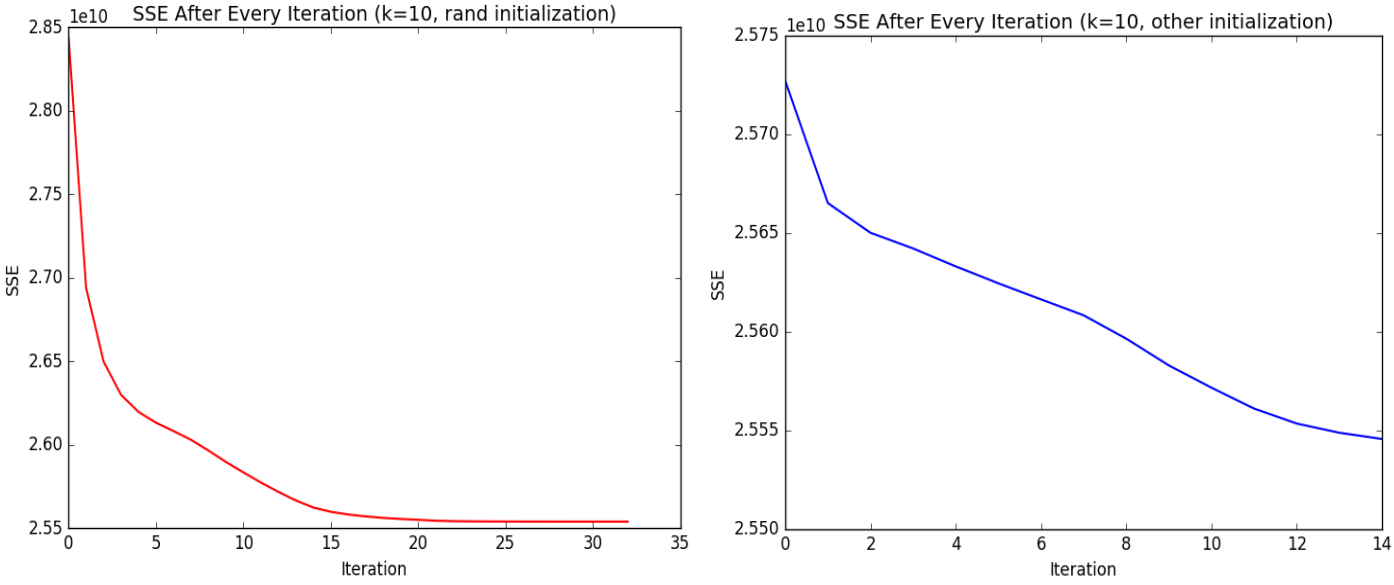

Figure 1. *Plots showing the SSE values after each iteration of clustering. (red) Plot for SSE using K-Means with parameters K=10 and random initialization. (blue) Plot for SSE using K-Means with parameters K=10 and refined initialization.*

From Figure 1, it is clear that SSE value decreases monotonically after every iteration when running the K-means algorithm with K=10 and both random and refined centroid initialization (Fayyad 1998).
In the figure for random initialization, SSE decreased monotonically from $2.84 * 10^{10}$ to around $2.56 * 10^{10}$. In the figure for refined initialization, SSE decreased monotonically from $2.572 * 10^{10}$ to around $2.555 * 10^{10}$.

(However, the rate of decrease does not always decrease monotonically, because the data points may be assigned to different clusters after each recalculation of centroids.
Observe that the initial SSE using the initialized centroids is lower for the refined initialization than the random initialization, and the number of iterations required until convergence is also fewer. This suggests that the run-time of the actual k-means algorithm is faster using the refined initialization. In addition, the final SSE achieved is lower on average as well. Hence, the refined initialization method is able to improve the outcome of classification.)

2. Run the K-means algorithm for the following values of K : 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 and initialization of random. Plot the final K-means objective (SSE) as a function of K. Do the same thing with the initialization of other. What value of K has the lowest SSE? What value of K do you think best "fits" this dataset? Why?

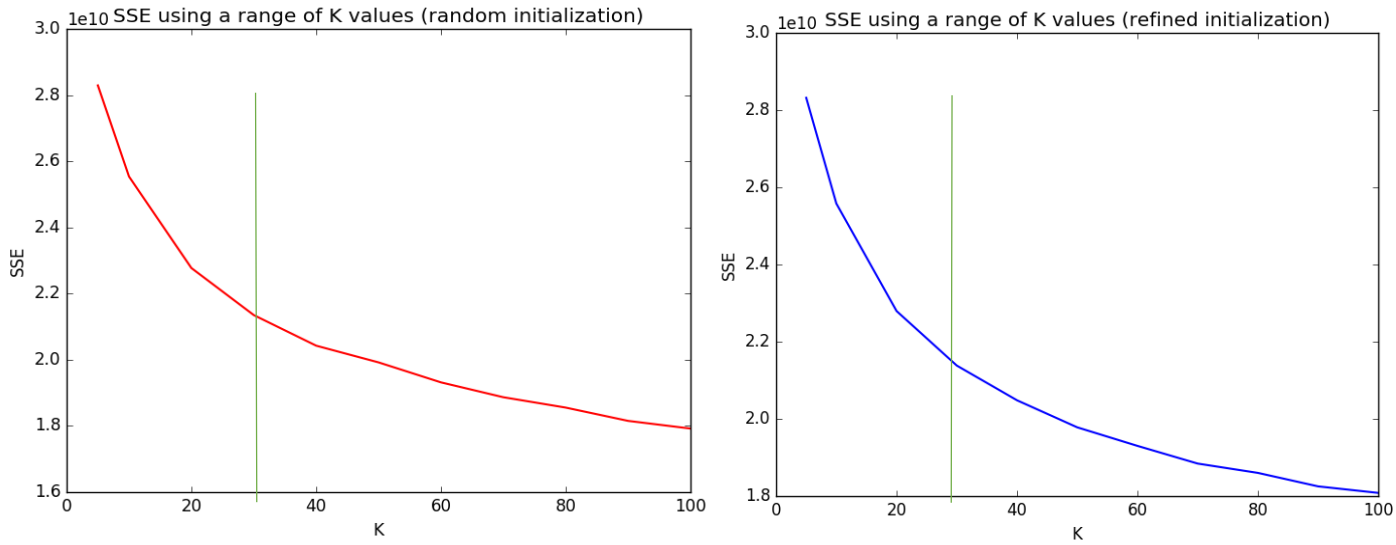

Figure 2. *Plots showing final SSE value after running K-Means with parameters K=5,10,20,30,40,50,60,70,80,90,100 and either (red) random initialization or (blue) refined initialization.*

Observe that using either method of initialization, final SSE decreases as K increases. Hence, among the K values that were tested, using K=100 had the lowest SSE at around $1.8 * 10^{10}$. Although we want to minimize SSE, we cannot increase K indefinitely because SSE will tend towards 0 as K tends closer to the actual number of data points. This results in overfitting. Thus, we want a small K with low SSE.

First, we will apply the Elbow/Knee method. But we see that the plots in figure 2 look like two relatively smooth curves, so there is not a clear 'elbow' point. The green lines at K=30 in Figure2 indicate the k value greater than which the rate of decrease in SSE becomes relatively low. Justification for this K value is that people write the same digits in very different ways, thus the same digit can be classified into different clusters based on how they are written. Hence we can claim that K=30 is a possible K value that fits our data, if we did not have previous knowledge about it.

Secondly, we can use our knowledge that there are 10 digits in our dataset. We want to cluster the same digits together and create a cluster for each digit. Hence K=10 is a reasonable k value to use by previous knowledge.

3. Run the K-means algorithm ten times with K = 10 (the "true" number of clusters)

and initialization of random. Report your final SSE values across all 10 runs.
Describe how might you evaluate the degree to which the resulting clustering has changed between these different runs of the algorithm?

Trial	1	2	3	4	5
Final SSE	25507836687.1	25660539567.6	25772909350.7	25566140075.2	25589999410.0
Trial	6	7	8	9	10
Final SSE	25575186591.5	25621165683.8	25540830618.8	25507948667.6	25589841601.0

Table 1. Final SSE value after running K-means using K=10 and random initialization of centroids.

The final SSE values across 10 runs are shown in Table 1. The lowest SSE obtained is 25507948667.6 and the largest SSE obtained is 25772909350.7.
The variations in the numbers obtained are likely due to the position of the initial centroids which can lead to the final clustering error being in different local minima.

However, for the dataset we are looking at, there are many other ways to analyze clustering.

First, assume that we do not have any previous knowledge about the data.
One possible measure is **the SSE of each resulting cluster.** Since we want the clusters to be compact, we can look at individual SSEs to check if the compactness of each cluster changed.
A third measure is the separateness of clusters. We can compute the **lowest average distance to any other cluster** to see how separated each cluster is from the rest. These measures will provide information on the consistency and quality of clustering, and are used to compute the **silhouette value**.
Another measure is how the **number of data points in each cluster** changes.

On the other hand, assuming we have the true label, we can analyze the clustering in other ways. For instance, we can **count the number of data points with each label present in each cluster** to evaluate the quality of clustering. As shown in Table 2 below, we can claim that trial 2 is probably a 'better clustering of distinct digits' than trial 1, even though the final SSE is slightly higher. This is because 9 out of the 10 digits were clustered into distinct clusters in the second trial, while only 8 out of 10 clustered distinctly in trial 1.

Trial	**1**	**2**
Final SSE	25507836687.1	25660539567.6
SSE of each cluster (compactness of clusters)	1. 2312710526.0803843, 2. 2800942446.0083294, 3. 2196774181.2608194, 4. 2159398624.8542914, 5. 2805712538.1216044, 6. 2544495543.9433765, 7. 3513281203.3263102, 8. 1010405502.7285508, 9. 2842422159.2211776, 10. 3321693961.6023831	1. 1899982633.3538477, 2. 2424035716.6249981, 3. 2259413161.7502975, 4. 2223141320.9072742, 5. 3996458183.0656505, 6. 2072875545.8057735, 7. 2535130414.1310668, 8. 2921266285.6210747, 9. 3029257387.5874949, 10. 10 2298978918.7069664
Digit Count	cluster 0 {**10.0**: 777, 5.0: 16, 6.0: 14, 2.0: 7, 3.0: 6, 9.0: 5, 8.0: 4, 7.0: 1} cluster 1 {**1.0**: 530, 2.0: 69, 8.0: 40, 7.0: 36, 4.0: 24, 3.0: 5, 9.0: 4, 5.0: 2, 6.0: 1} cluster 2 {**2.0**: 666, 3.0: 32, 6.0: 9, 8.0: 8, 10.0: 8, 4.0: 5, 7.0: 3, 5.0: 2, 1.0: 1} cluster 3 {**3.0**: 569, 5.0: 275, 8.0: 213, 2.0: 56, 10.0: 50, 9.0: 16, 6.0: 5, 1.0: 1, 7.0: 1} cluster 4 {**4.0**: 361, 9.0: 254, 7.0: 98, 6.0: 89, 8.0: 37, 5.0: 31, 2.0: 19, 10.0: 18, 3.0: 13, 1.0: 1} cluster 5 {**1.0**: 463, 5.0: 307, 6.0: 141, 2.0: 129, 8.0: 103, 3.0: 101, 4.0: 54, 7.0: 50, 9.0: 35, 10.0: 5} cluster 6 {**6.0**: 729, 10.0: 39, 2.0: 23, 5.0: 21, 8.0: 16, 4.0: 15, 3.0: 9, 9.0: 3} cluster 7 {**7.0**: 404, 9.0: 277, 4.0: 257, 8.0: 25, 5.0: 20, 3.0: 11, 2.0: 6, 1.0: 1, 10.0: 1} cluster 8 {**8.0**: 525, 5.0: 274, 3.0: 234, 10.0: 100, 2.0: 23, 6.0: 12, 1.0: 2, 7.0: 1, 9.0: 1} cluster 9 {**7.0**: 406, 9.0: 405, 4.0: 284, 5.0: 52, 8.0: 29, 3.0: 20, 2.0: 2, 10.0: 2, 1.0: 1}	cluster 0 {**10.0**: 723, 6.0: 13, 5.0: 8, 2.0: 6, 9.0: 6, 8.0: 3, 3.0: 1, 7.0: 1} cluster 1 {**1.0**: 981, 2.0: 162, 8.0: 90, 7.0: 68, 3.0: 62, 6.0: 26, 4.0: 23, 9.0: 20, 5.0: 10} cluster 2 {**2.0**: 654, 3.0: 15, 4.0: 9, 8.0: 7, 6.0: 6, 9.0: 4, 1.0: 2, 5.0: 2, 7.0: 2} cluster 3 {**3.0**: 420, 5.0: 240, 8.0: 198, 10.0: 60, 2.0: 45, 9.0: 14, 6.0: 5, 1.0: 2, 7.0: 2, 4.0: 1} cluster 4 {**4.0**: 537, 9.0: 464, 7.0: 305, 5.0: 51, 3.0: 27, 2.0: 20, 8.0: 18, 6.0: 14, 10.0: 8, 1.0: 3} cluster 5 **{5.0**: 405, 6.0: 182, 4.0: 134, 8.0: 63, 2.0: 50, 9.0: 43, 10.0: 39, 3.0: 34, 7.0: 27, 1.0: 7} cluster 6 {**6.0**: 746, 2.0: 24, 10.0: 21, 4.0: 16, 5.0: 13, 8.0: 12, 3.0: 5, 9.0: 2, 1.0: 1, 7.0: 1} cluster 7 {**7.0**: 590, 9.0: 442, 4.0: 276, 5.0: 16, 8.0: 15, 3.0: 12, 2.0: 6, 1.0: 3} cluster 8 {**8.0**: 560, 5.0: 72, 3.0: 53, 2.0: 17, 4.0: 4, 7.0: 4, 9.0: 2, 10.0: 2, 1.0: 1} cluster 9 {**3.0**: 371, 5.0: 183, 10.0: 147, 8.0: 34, 2.0: 16, 6.0: 8, 9.0: 3}

Table 2. Examples of ways to analyze clustering difference in addition to final SSE value after running K-means using K=10 and random initialization of centroids.

4. Plot the final centroids found when you the K-means algorithm ten times with K = 10 and initialization of random.

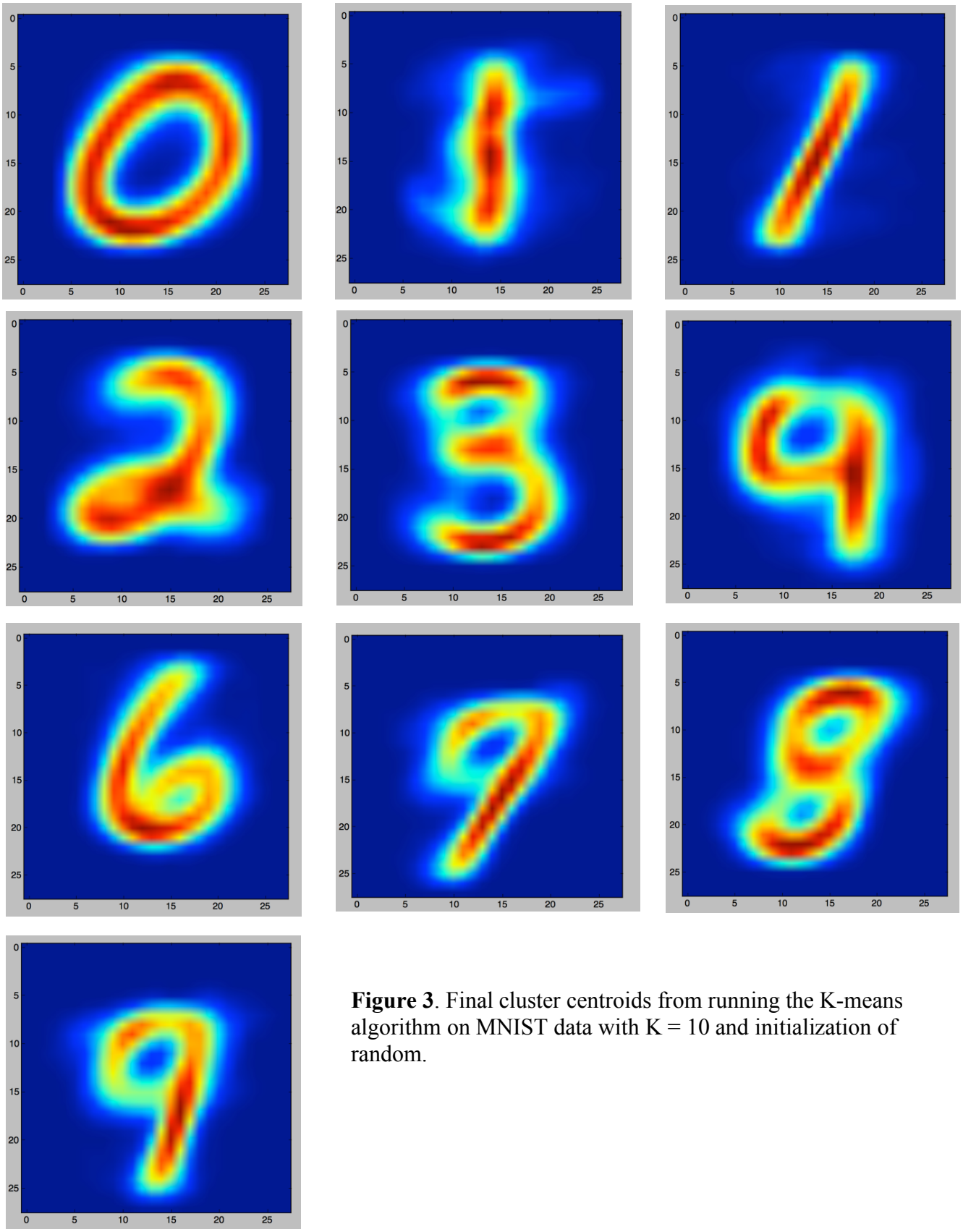

Figure 3. Final cluster centroids from running the K-means algorithm on MNIST data with K = 10 and initialization of random.

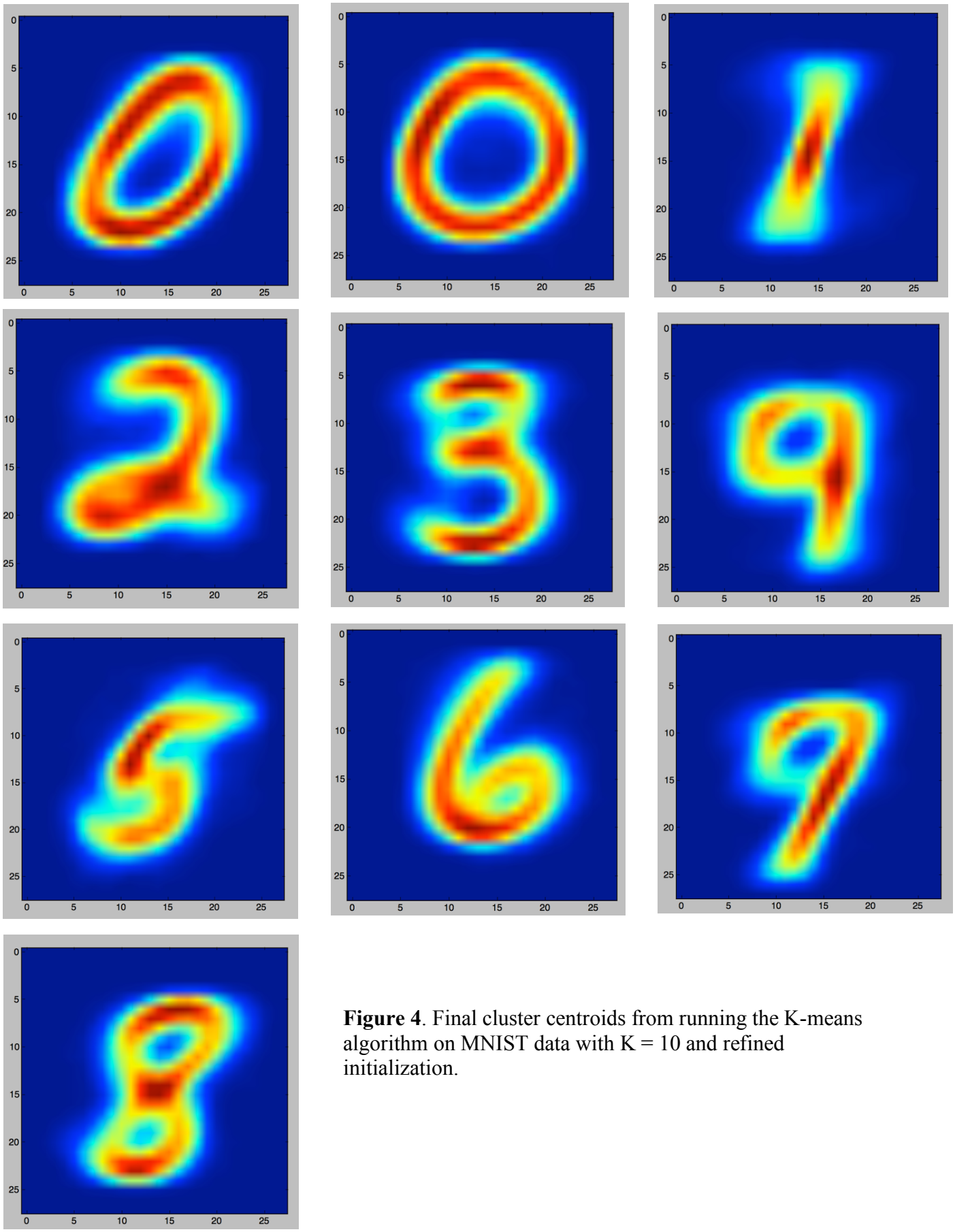

Figure 4. Final cluster centroids from running the K-means algorithm on MNIST data with K = 10 and refined initialization.

a) Are these what you would expect?

These figures show the results of the clustering that resulted in the least duplicates in terms of the digits represented by the 10 centroids, selected from 10 repeats. Each plot shows the average of all the data points assigned to each cluster. Specifically, each pixel is the average value of the corresponding pixels in each data point assigned to the cluster. Thus, assuming K-Means algorithm worked and the majority of the same digits are clustered into the same clusters, then these plots are expected to look like the 10 digits present in the data we are clustering.

In 7 out of the 10 runs, less than 9 clusters showed distinct digits. In the 'best' outcome, 9 of the 10 digits are represented by a centroid as shown in Figure 3. The presence of duplicates is expected, because some digits can be written in such different ways such that the same digit may be clustered into different clusters. On the other hand, some digits look very similar such that distinct digits may be clustered together. Also, this can be because the initial centroids are randomly selected, and the location of the initial centroids can affect the ultimate clustering outcome. This is because the clustering can end up in some local minima.

As shown in Figure 3, there are two digit 1's but no digit 5's. This is probably because there are multiple ways people write the digit 1 (straight or slanted). This might also be due to the fact that the digit 5's are absorbed into other clusters such as cluster of digit 1 (middle column top row), cluster of digit 3 and cluster of digit 8, due to the similar shape of these digits.

Also observe that the centroid plot that looks like digit 4, 7 and 9 look very similar. This might be due to the inherent similarity in the structure of these digits.

b) How do the centroids compare to what you found with the random initialization?

Using the refined initialization allows K-Means to achieve 'good quality' clustering more consistently, with less duplicates and noise in the plots (where at least the plots of 9 out of 10 clusters show distinct digits). This is because we are using the clustering center of clusters to be the initial centroids, so the initial centroids are more likely to have a value close to a dense number of data.

In figure 4, observe that there are two centroids for the figure zero. The centroids for 0 reflect that some people draw their zeros like a, and some people draw their zeros like an egg.

However, like using random initialization, the results seldom produced 10 distinct digits for each cluster, because of the variation in the way people write digits, as well as

5. Prove that this procedure for handling empty clusters will reduce clustering error.

Suppose we currently have an empty cluster represented as e. Then no data points are assigned to the centroid of e (the centroid is denoted as Ce) because for all data points, there exits some centroid that is closer than this centroid in Euclidean distance. Observe that removing this cluster does not affect the total SSE, since this cluster does not contain any data points, thus is not involved in the calculation of SSE.

Now, if we remove this cluster, and assign the data point that is furthest in distance to its current centroid (denoted as Cf) to be the new cluster centroid, then the data point Cf itself and any other data point that is closer to Cf than to their current assigned centroid will be assigned to Cf. Observe that every data point is either reassigned to Cf, or remains in its original cluster.
Since the unchanged data points are still in the same cluster, their distance to their corresponding centroid do not change, hence they contribute the same value to SSE. Thus, it is sufficient to examine all the points that were reassigned to Cf and verify that SSE decreases after the current iteration.

First of all, the squared distance between Cf and its original cluster is deducted from the original SSE, because Cf is assigned to Cf itself, so the distance is 0 and this data point now contributes 0 to the total SSE.
Then, we consider any other data points that are reassigned to Cf. If a data point x is assigned to Cf, this implies that the distance between x and Cf is smaller than the distance between x and its original centroid. Hence, the clustering error of x is reduced, and contributes less towards the total SSE. Similar argument applies to all reassigned data.

In all, the clustering error of each data point is either unchanged or reduced, which implies that the final clustering error after the current iteration will always decrease.

6. Modify your code to accept a third possible type of initialization cheating. How do your results compare to your previous results with the other types of initialization?
The final SSE obtained is 25652926837.5.

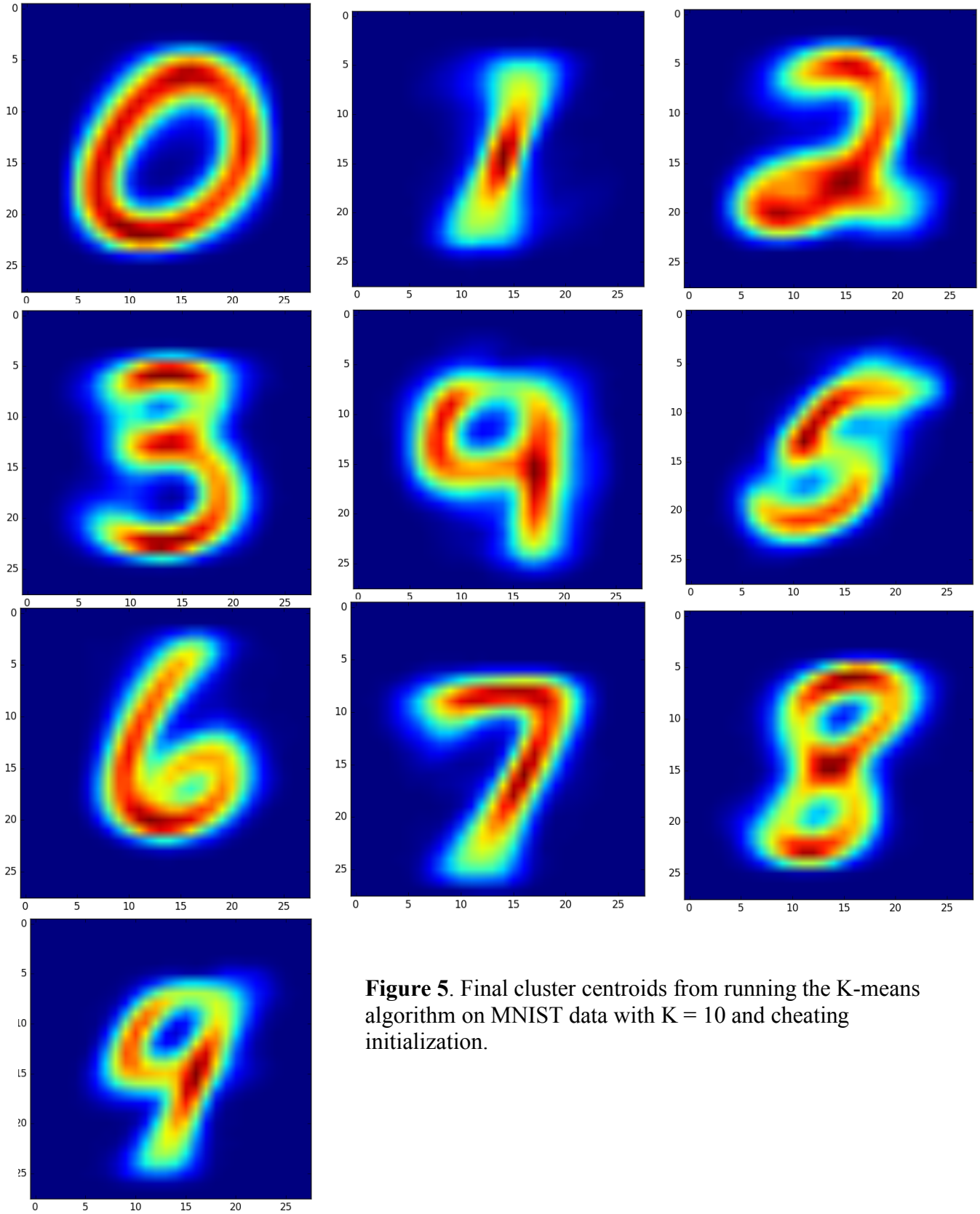

Figure 5. Final cluster centroids from running the K-means algorithm on MNIST data with K = 10 and cheating initialization.

Using the cheating method to initialize the centroids, the final SSE is around 25652926837.5, which is higher than some SSE obtained using other methods of initialization. However, the clustering outcome, as shown in figure 5, seems to be ‘better’ than those obtained using other methods.

First of all, each cluster represents a distinct digit. This implies that the majority of data points assigned to a cluster are often the same digit. This is probably because using the average of the data points of each digit to calculate the initial centroids simulates the process of assigning the data points with the same label into one cluster, and then re-computing the centroid, and then continue with the objective optimization. Hence, we are more likely to obtain a clustering based on the 10 distinct digits.

Reference:

Bradley, Paul S., and Usama M. Fayyad. "Refining Initial Points for K-Means Clustering." *ICML*. Vol. 98. 1998.