Problem 2 (K-Means and HAC, 20pts)

For this problem you will implement K-Means and HAC from scratch to cluster image data. You may use `numpy` but no third-party ML implementations (eg. `scikit-learn`).

We've provided you with a subset of the MNIST dataset, a collection of handwritten digits used as a benchmark for image recognition (learn more at http://yann.lecun.com/exdb/mnist/). MNIST is widely used in supervised learning, and modern algorithms do very well.

You have been given representations of MNIST images, each of which is a 784×1 greyscale handwritten digit from 0-9. Your job is to implement K-means and HAC on MNIST, and to test whether these relatively simple algorithms can cluster similar-looking images together.

The code in `T4_P2.py` loads the images into your environment into two arrays – `large_dataset`, a 5000x784 array, will be used for K-means, while `small_dataset`, a 300x784 array, will be used for HAC. In your code, you should use the ℓ_2 norm (i.e. Euclidean distance) as your distance metric.

Important: Remember to include all of your plots in your PDF submission!

Checking your algorithms: Instead of an Autograder file, we have provided a similar dataset, `P2_Autograder_Data`, and some visualizations, `HAC_visual` and `KMeans_visual`, for how K-means and HAC perform on this data. Run your K-means (with $K = 10$ and `np.random.seed(2)`) and HAC on this second dataset to confirm your answers against the provided visualizations. Do **not** submit the outputs generated from `P2_Autograder_Data`. Load this data with `data = np.load('P2_Autograder_Data.npy')`.

1. Starting at a random initialization and $K = 10$, plot the K-means objective function (the residual sum of squares) as a function of iterations and verify that it never increases.

2. Run K-means for 5 different restarts for different values of $K = 5, 10, 20$. Make a plot of the final K-means objective value after your algorithm converges (y-axis) v. the values of K (x-axis), with each data point having an error bar. To compute these error bars, you will use the 5 final objective values from the restarts for each K to calculate a standard deviation for each K.

   How does the final value of the objective function and its standard deviation change with K? (Note: Our code takes 10 minutes to run for this part.)

3. For $K = 10$ and for 3 random restarts, show the mean image (aka the centroid) for each cluster. To render an image, use the pyplot `imshow` function. There should be 30 total images. Include all of these images as part of a single plot; your plot must fit on one page.

4. Repeat Part 3, but before running K-means, standardize or center the data such that each pixel has mean 0 and variance 1 (for any pixels with zero variance, simply divide by 1). For $K = 10$ and 3 random restarts, show the mean image (centroid) for each cluster. Again, present the 30 total images in a single plot. Compare to Part 3: How do the centroids visually differ? Why?

5. Implement HAC for min, max, and centroid-based linkages. Fit these models to the `small_dataset`. For each of these 3 linkage criteria, find the mean image for each cluster when using 10 clusters. Display these images (30 total) on a single plot. How do these centroids compare to those found with K-means? **Important Note:** For this part ONLY, you may use `scipy`'s `cdist` function to calculate Euclidean distances between every pair of points in two arrays.

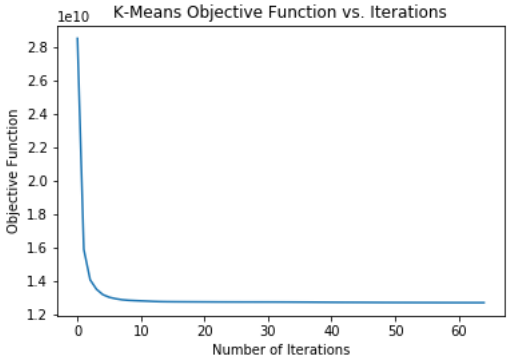
6. For each of the 3 HAC linkages (max/min/centroid), plot "Distance between most recently merged clusters" (y-axis) v. "Total number of merges completed" (x-axis). Does this plot suggest that there are any natural cut points?

7. For each of the max and min HAC linkages, make a plot of "Number of images in cluster" (y-axis) v. "Cluster index" (x-axis) reflecting the assignments during the phase of the algorithm when there were $K = 10$ clusters. Intuitively, what do these plots tell you about the difference between the clusters produced by the max and min linkage criteria?

8. For your K-means with $K = 10$ model and HAC min/max/centroid models using 10 clusters on the `small_dataset` images, use the `seaborn` module's `heatmap` function to plot a confusion matrix of clusters v. actual digits. This is 4 matrices, one per method, each method vs. true labeling. The cell at the ith row, jth column of your confusion matrix is the number of times that an image with the true label of j appears in cluster i. How well do the different approaches match the digits? Is this matching a reasonable evaluation metric for the clustering? Explain why or why not.

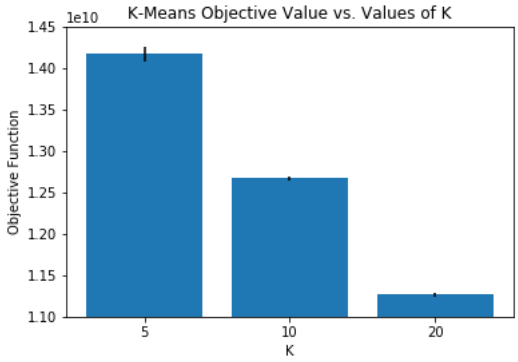
Solution

1. The K-Means objective function, as a function of iterations, is shown in the plot below:



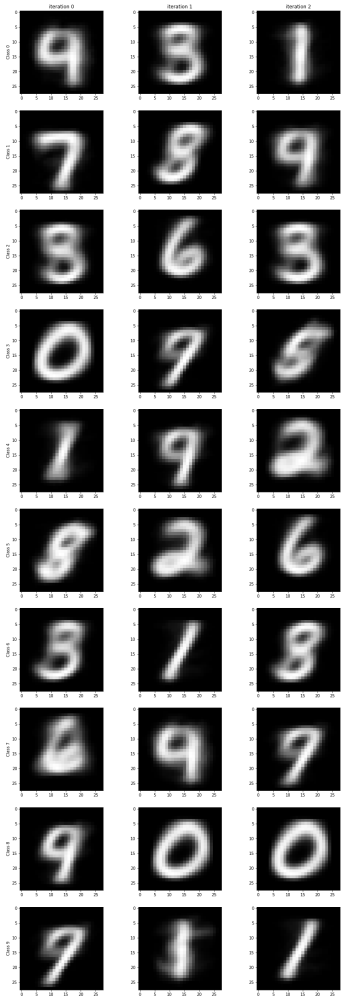
   From the plot we can see that the objective function is constantly decreasing with increasing number of iterations.

2. The mean and standard deviation of the objective values for K-means on 5 different restarts for different values of $K = 5, 10, 20$ is shown below:

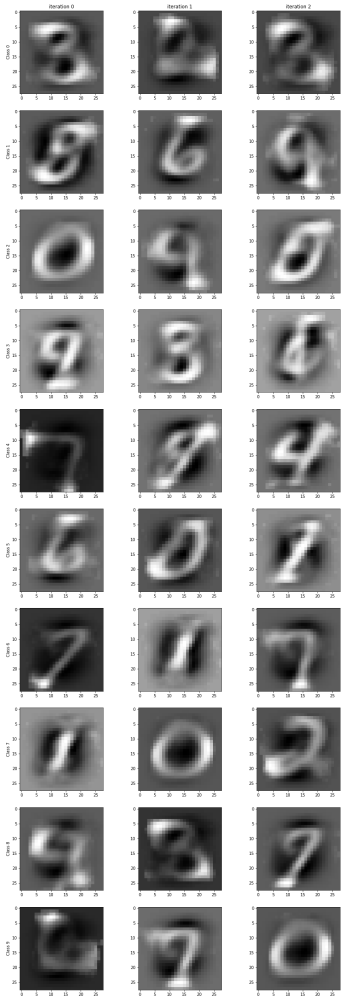


   The means for these 15 iterations were, for $K = 5, 10, 20$ respectively, [1.416e10, 1.267e10, 1.127e10], and the standard deviations were [8.989e7, 2468e7, 2.057e7]. It should be noted that the results on different iterations varied. Across all iterations, the mean objective value decreased as K increased, as shown by the decreasing means for $K = 5, 10, 20$. However, the standard deviations varied widely across iterations, so it is difficult to make a conclusion on how the standard deviation varied with K. On this particular iteration, we found that the standard deviation decreased as K increased, as shown by the decreasing standard deviations for $K = 5, 10, 20$.

3. The plot of 30 mean images for 3 random restarts is shown below:



4. The plot of 30 mean images for 3 random restarts with standardized data is shown below:



As the question only asks how the centroids differ, we shall only compare the mean image results for the centroid K-Means model (the far right column).

The non-standardized mean images appear much clearer than the standardized mean images. While the non-standardized images are all fairly clear to read, there are a couple of non-standardized images that are either difficult to read, or entirely indecipherable. Moreover, the background in the non-standardized data is a solid black, while the backgrounds for the standardized data are various shades of grey, which further makes the non-standardized images to appear clearer.

The difference in quality of the images is due to the standardizing effect of the data. By standardizing, any pixel that is a unique datapoint in our data will be pushed to mean 0 and not appear in our mean

images. This is further contributed to by the averaging effect of the centroid model. For example, if a cluster contain many 1s and a few 4s, then the pixels not on the vertical line of the 1 and 4 are pushed to mean 0, causing the indecipherable standardized images.

5. The 30 mean images for Min HAC, Max HAC, and Centroid HAC is shown below:

As the question only asks how the centroids differ, we shall only compare the mean image results for the centroid K-Means model and the centroid HAC model (the far right column).

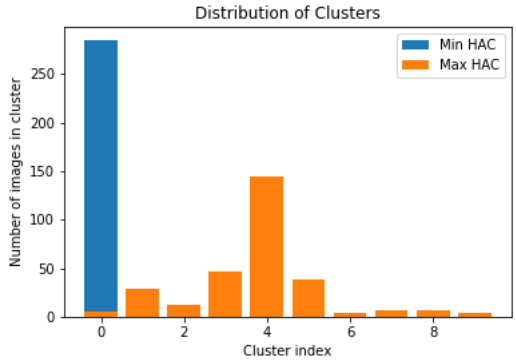
The centroid images appear much clearer than the centroid images for both the K-Means non-standardized and standarized models. The on exception is the first class (top right image), where it is indecipherable.

Upon closer examination of the sizes of each class in the centroid HAC, we find that all clusters except one had very few datapoints contained in them, and there was one cluster that contained most of the data points. This explains the results of the centroid HAC images, as the clusters with very few data points are expected to appear very clear, while the cluster with most of the datapoints appears very obscured, as evidenced in the images.

6. The plot of distances between most recently merged clusters vs. total number of merges completed is shown below:

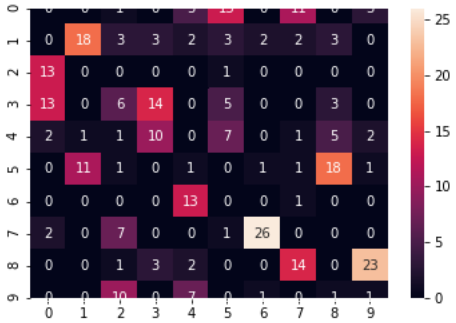
The plot suggests a natural cut point of about 260-270 merges, leaving us with 30-40 clusters. This would be a relatively reasonable number of classes to have to be interpretable. The plot of distances shows that around that cutpoint, all three of our HAC models have distances between merged clusters increasing at an increasing rate with the number of merges. In particular, the Max HAC model shows a significant bend upwards, while the Min HAC and Centroid HAC model show slight bends upwards at this cut point. This would be an ideal cut point, since for these iterations the HAC models are merging some of the largest distances between clusters, which reduces the accuracy of our clusters.

7. The plot of number of images in cluster vs. cluster index for max and min HAC is shown below:

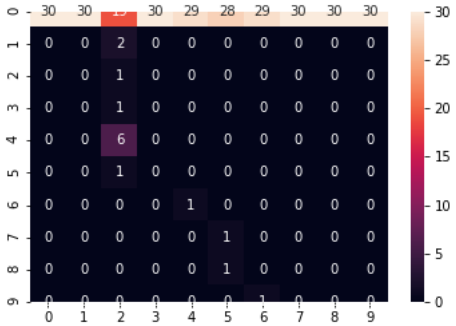


The difference in the sizes of the clusters, or the distribution of the data points in the clusters, illustrates expected behavior when comparing the Min HAC and Max HAC. As discussed in lecture, Min HAC leads to longer and "stringier" clusters, which causes nearly all of the datapoints to appear in a single class for Min HAC. Whereas the distribution of datapoints for Max HAC appears more uniform, consistent with the discussion in lecture on how Max HAC leads to more compact clusters. Considering the datapoints, we expect that most of the writing is done on the middle of the image, so Min HAC will put most of the data points in the same cluster besides a few outlier data points.

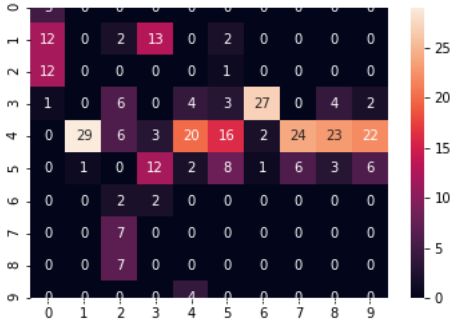
8. The confusion matrix for K-Means with $K = 10$ is shown below:



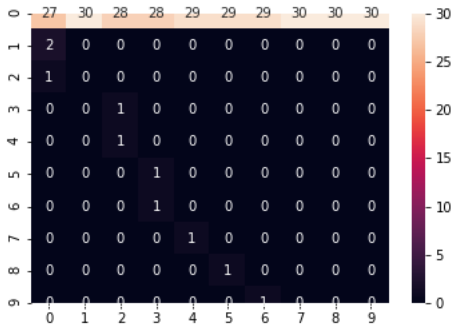
The confusion matrix for min HAC is shown below:



The confusion matrix for max HAC is shown below:



The confusion matrix for centroid HAC is shown below:



Good performance on the confusion matrix would be evidenced by only one large cell (value of approximately 30) in each row and each column.

From the confusion matrix, it seems that the K-Means model classifies the digits best. In particular, the digits with true class 6 are well represented by cluster 7 in our K-Means model. However, for the rest of the true classes, the K-Means model does not perform well, for example true label 7 and true label 9 are both represented by cluster 8 in our K-Means model.

Max HAC does the next best on the confusion matrix, as the distribution of its clusters is not as well spread as K-Means. However, it does a good job of classifying true label 6. A main issue is that cluster 4 of our HAC contains the majority of the data points, and represents most of the data points for multiple true labels, which hurts this model's ability to classify correctly.

Both min HAC and centroid HAC exhibit similar performance on the confusion matrix. Both classify nearly all of the data points into cluster 0, which makes these not suited at all to classify any digits since they will almost all be matched to cluster 0.

The matching represented by the confusion matrix is not a reasonable evaluation metric for the clustering, as evidenced by the extremely poor results for the min HAC and centroid HAC. Moreover, even for the K-Means and Max HAC model, which show decent distributions in their assignment to the 10 clusters, the confusion matrix is not an effective means at displaying these distributions.