Project 1 Report:

K++ Initialization, the Elbow Method, and Application to Handwritten Digits

Anthony Wang, Nikita Anistratov, Erica Council

North Carolina State University

MA 493: Mathematical Foundations of Data Science

Dr. Mansoor Haider

March 9th, 2022

Note: for references to figures, see Appendix at the end of this report.

Part I: K++ Initialization

In part one, we coded for k-means clustering to be run on a given data set, Q1Data, while graphing the data and clusters on a scatter plot. We created two different sections of code: one for k-means clustering using random initialization, and one for k-means clustering using k++ initialization. We ran the code ten times using each form of initialization, thus creating 10 figures for each. Based on the figures, we can conclude that k++ initialization yields significantly better performance of k-means clustering.

K++ initialization overall yields better results of clustering the data because the distance between the initial representative vectors is maximized. In this initialization, the first weight vector is the data vector of a randomly selected data point from Q1 Data. The next representative vector is chosen by finding the data point that is furthest from the first data point chosen. Following this, the next is chosen by measuring the distance from each data point and its closest centroid, whichever data point has the maximum value of such distance is chosen as the next representative vector. In other words, the data point furthest from the already-determined centroids is then chosen to be the next centroid. We will end up with k centroids initialized after k++ initialization, in our case, k = 5.

Using random initialization, it is quite possible for initial weight vectors to be randomly selected as points that are very close to each other. When this occurs, not only does it take more steps to match each data point with a cluster; oftentimes, there will be weight vectors that end up without any data points associated with it, as can be seen in figures 1.1.3-7 and 1.1.9. However, with k++ initialization, the initial positions of the clusters are not chosen at random, but are at maximal distance from each other and are at the location of points within the data set. Thus, as can be seen in figures 1.2.1-10, k++ initialization eliminates the possibility of clusters being left without any data points associated with them. Consequently, the measure of coherence using k++ is usually smaller than when using random initialization as data points are generally closer to their associated weight vectors from the start. This idea will be further investigated in Part II. K++ initialization also leads to fewer steps in the alternating minimization scheme as the weight vectors are very close to ideal since initialization, so they do not have to update as much as in random initialization. Therefore, k++ initialization is overall more efficient and effective than random initialization.

Part II: the Elbow Method

In part two, we used the data and k-means codes from part one to determine the optimal number of clusters for both random and k++ initialization via the elbow method. We can utilize the elbow method to determine the optimal number of clusters by computing and plotting the overall coherence of the clustering with different k values. The overall coherence of clustering is a measure of the clustering performance: the smaller the overall coherence, the better the clustering. It is the sum of squared distance between each data point and their respective cluster center. The following equation is the mathematical basis for determining the coherence for 1 cluster, cluster l:

$$q_l = coherence = \sum_{j \in Il} \left| X_j - C_l \right|^2$$

where Xj is the vector representing the set of data points associated with Cl, and Cl is the representative vector representing a given cluster l.

Then, the overall coherence sums up the coherence for each cluster:

$$Q \ = \ overall\, coherence \ = \ \sum_{l=1}^{k} q_l$$

The elbow method uses line plots to help determine which k value (number of clusters) results in the smallest measure of overall coherence. By modifying our code from part one, we were able to run both k++ and random initialization, then alternating minimization scheme, with the number of clusters ranging from one to eight, for five realizations of clustering each. We also produced line plots of the overall coherence over different values of k for each of these clusterings. To interpret the graph, we looked for which value of k, or number of clusters, resulted in a sudden change in the slopes of the coherence measure lines. The change of the lines from having steep negative slopes to having slopes of nearly zero (reminiscent of an elbow, hence the name "elbow method",) indicates that after a threshold value for k, having more clusters produces the same level of overall coherence. We can understand this to be true also because the line change slope at a point with very small overall coherence, showing that at, and beyond, the threshold value of k, the clustering will perform equally as well. Therefore we can select that threshold value of k for the optimal number of clusters.

We determined that the best choice for k-means clustering with k++ initialization on this data set is k = 4. As seen in Figure 2.2, the slope of each line levels off at k = 4, indicating that for all 5 realizations of the clustering, overall coherence becomes very minimal for k = 4 through k = 8. Thus, only four clusters are needed. However, with random initialization, we are not as certain about the proper choice for the number of clusters. As seen in Figure 2.1, lines begin to level off at either k = 4 or k = 5, which means some realizations of clustering suggest using 4 clusters is better and some suggest 5. It does not provide a clear answer of the optimal value of k. This makes sense based on a visual inspection of the dataset. When viewing the data as plotted on a scatter plot, there is a separation between the leftmost ([0.2,0.5], [0.7,1]) and rightmost ([0.5,0.9],[0.5,0.9]) data found in the upper right corner ([0.2,0.9],[0.4,1]). Thus, it can be understood that this cluster of data in the upper right corner could be grouped into these leftmost and rightmost clusters, resulting in five clusters in total. However, it can be argued that all of the data in the upper right corner be grouped together as its distinction from the data appearing in the upper left, bottom left, and bottom right corners of the graph are significantly more obvious, resulting in four clusters.

Part III: Application to Handwritten Digits

For part three of our project, we will be applying k-means clustering to the first 100 images from the classic MNIST image set. Each image has 20x20 pixels, so after converting these images into numerical data, we have 100 data observations, each with 400 features, each feature representing the greyscale value of one pixel. We were able to modify our previous code in order to perform k-means clustering and use the elbow method on the converted image data. We used k++ initialization and assigned the forty-second data point in the data set to be the first weight vector to be initialized. We ran one realization of clustering for each possible k value (3 to 10) to produce a one-line elbow plot.

Upon k-means clustering the data with the 8 different numbers of clusters, we were also able to generate a graph of the overall coherence for each k value using the elbow method, as depicted in figure 3.1. Because this image data set has 400 features, as compared to the 2-featured Q1Data used earlier in the project, determining the best k-value is not as straightforward. This is visualized in the elbow method graph as there is no distinct change from a steep slope to a nearly horizontal slope, and thus many of our selected k-values result in a large measure of coherence. The threshold k value at which the change in slope most significantly is somewhere between 6 and 7. We ran k-means clustering with k++ for both k = 6 and k = 7, but upon finding that one cluster using k = 7 only contains one data point, we decided that k = 6 is the better k value for this dataset. Then we find the success score of clustering with k = 6, which is the number of data vectors from our 100 data vectors that are correctly labeled with their true label after the clustering. The success score for our chosen value k = 6 is 45.

We then obtained each cluster's size (clstSize) and individual success score (clstSvector) and computed the proportions of correct labeling (clstRatio) for each cluster. After analyzing the clstRatio variable, we determined that cluster 2 and cluster 4 are 2 clusters that performed the best in detecting similar patterns, as they both have a perfect labeling accuracy of 100% (clstRatio = 1). Cluster 2 is labeled as "0" because 0 is the most frequent digit in cluster 2; Cluster 4 is labeled as "2" because 2 is the most frequent digit in cluster 4. As we convert the data vectors from these 2 clusters back to their image form and render them, we see that cluster 2 shows 4 images of handwritten digit 0 (see Figure 3.2.1-4) and cluster 4 shows 3 images of handwritten digit 2 (see Figure 3.3.1-3).

We also determined that the worst cluster in detecting similar patterns is cluster 1 (clstSize = 52, clstRatio = 0.2692). However, cluster 1 has 52 data vectors, which will be too many images to display in this report, so we choose cluster 6 (clstSize = 7, clstRatio = 0.5714) as our example of one cluster that performed less well. Cluster 6 is labeled as "0" because 0 is the most frequent digit in cluster 6. As we convert the data vectors from cluster 6 back to their image form and rendered them, we see from Figures 3.4.1-7 that there is an assortment of handwritten zeros, fours, and sixes with zero being the most frequent (4 out of 7 images).

These figures prove to us that the elbow method for k-means clustering with k++ initialization generally performs well, as it resulted in two perfect clusters. Another strength of this method is it allows analysts to actively choose a value for k by intuitively evaluating the elbow plot, rather than leaving every decision up to the machine. A significant limitation of this method would be the complexity of the data. The MNIST image data we are applying our method to are 20x20 pixels, which means each data point has 400 features. Comparing our analysis of MNIST data with our analysis of the 2-dimensional Q1Data, we see that the high-dimensional MNIST data produces an elbow graph that is less obvious for the analyst to decide on the k value. Another flaw with the elbow method's application on handwritten digits data is that we would intuitively want ten distinct clusters since there are ten distinct digits, so we would not need to use the elbow method to determine the best k value at all. However, the elbow method suggests six clusters is the optimal choice, which does not offer practical significance. This will be further investigated in the bonus section. In conclusion, although we had some high success scores with a few clusters, the elbow method may not be the most effective method for this data context.

Bonus: Elbow Method Effectiveness

To determine if k = 6 provided by the elbow method is the ideal selection of k for our data, we run the code that computes success scores once for each possible k value (k = 3, 4, 5, 6, 7, 8, 9, 10), and recorded the success scores:

k	S (success score)
3	26
4	27
5	39
6	45
7	49
8	50
9	50
10	51

Above are all of the success scores for each corresponding k. As we have explained in Part III, by utilizing the elbow method we came to the conclusion that k = 6 (with S = 45) is our optimal choice. However, the success scores reflect that as k increases, the success score also increases, leading us to believe that 10 is a better choice for k, as 51 out of 100 image data are correctly labeled, which is higher than 45 when we use k = 6. Hence, the elbow method does not yield the highest score of S. The highest S occurs for k = 10. As mentioned in Part III, intuitively, separating images of single handwritten digits into 10 groups makes more practical sense than separating them into 6 groups, because there are 10 unique digits, not 6.

Appendix. Figures

Part I Figures

1.1 Random Initialization

Figure 1.1.1

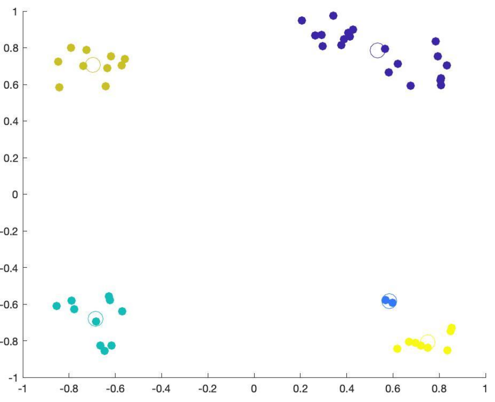


Figure 1.1.2

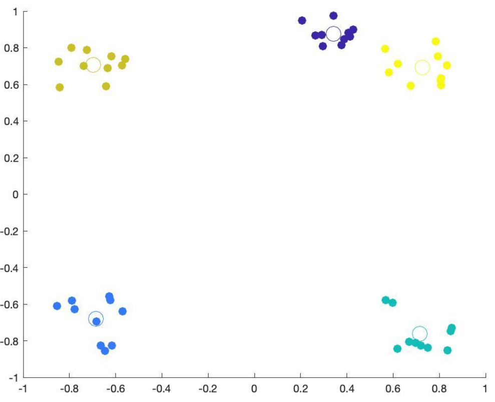


Figure 1.1.3

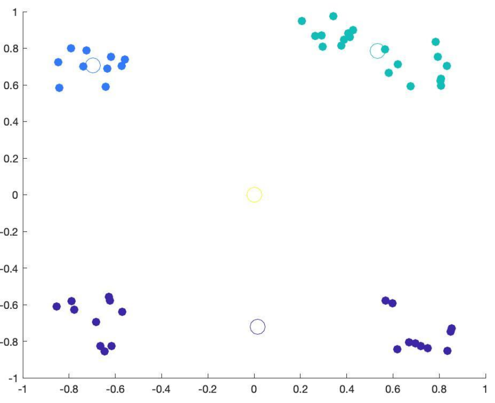
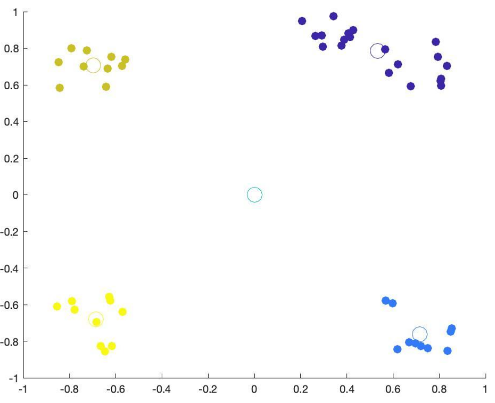


Figure 1.1.4



Appendix. Figures

Figure 1.1.5

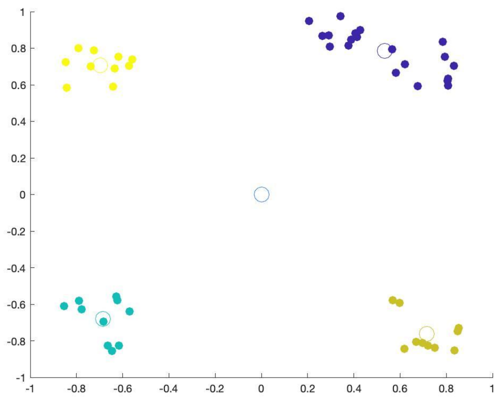


Figure 1.1.6

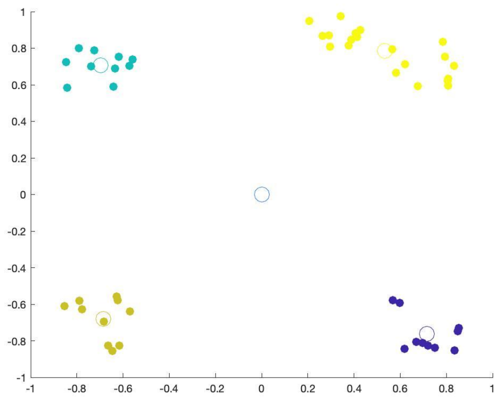


Figure 1.1.7

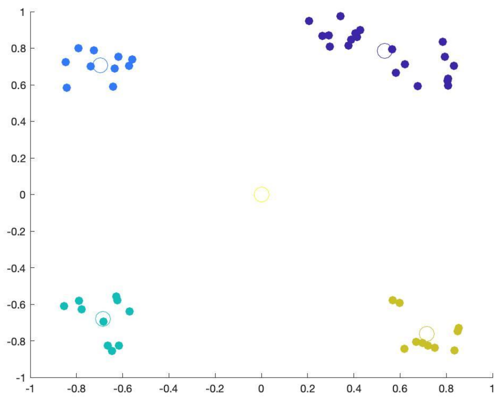


Figure 1.1.8

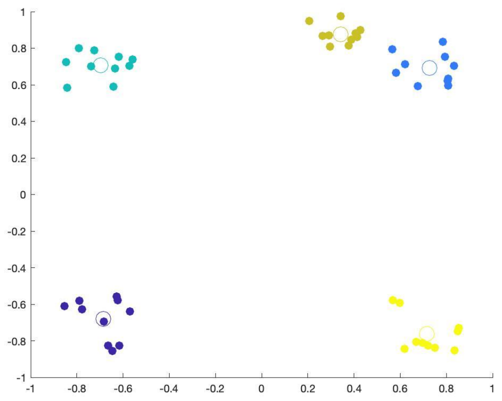


Figure 1.1.9

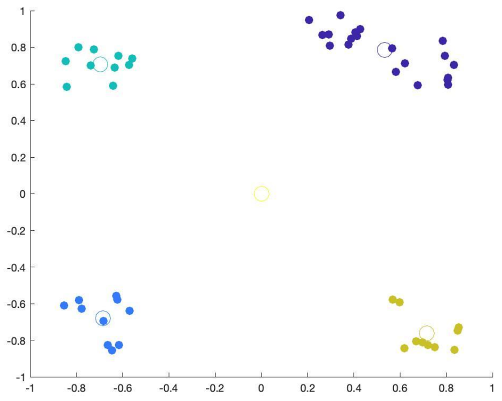
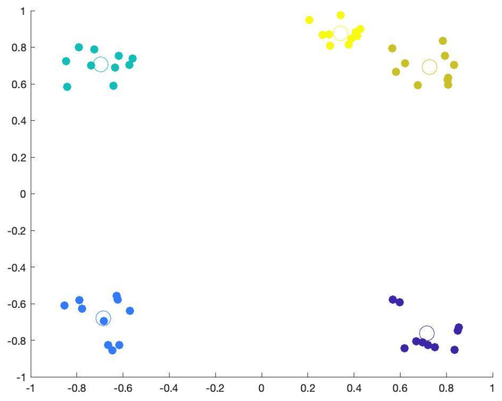


Figure 1.1.10



Appendix. Figures

1.2 K++ Initialization

Figure 1.2.1

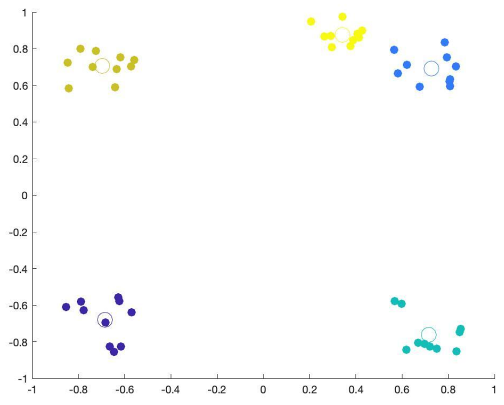


Figure 1.2.2

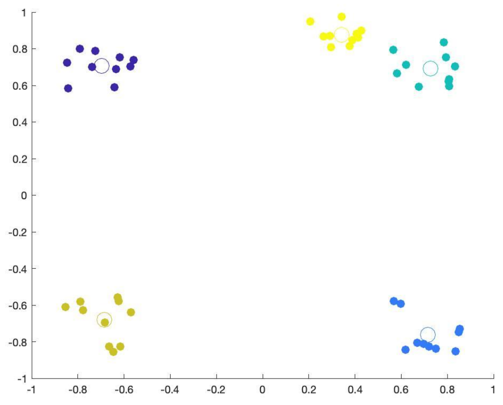


Figure 1.2.3

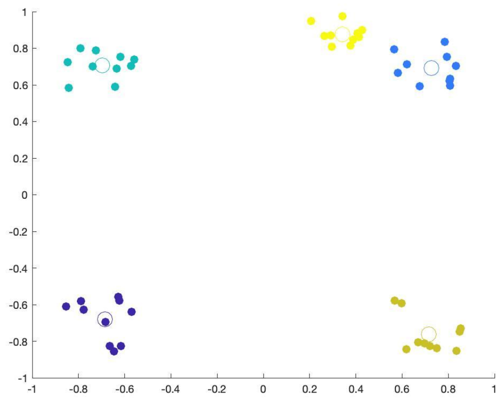
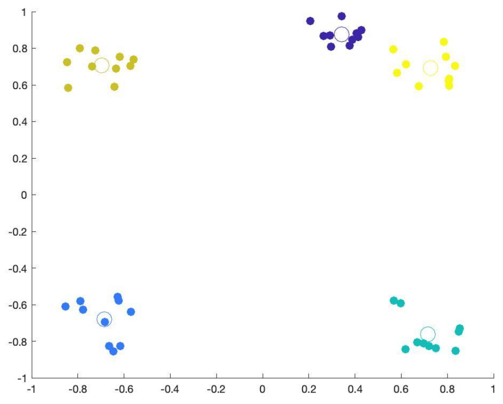


Figure 1.2.4



Appendix. Figures

Figure 1.2.5

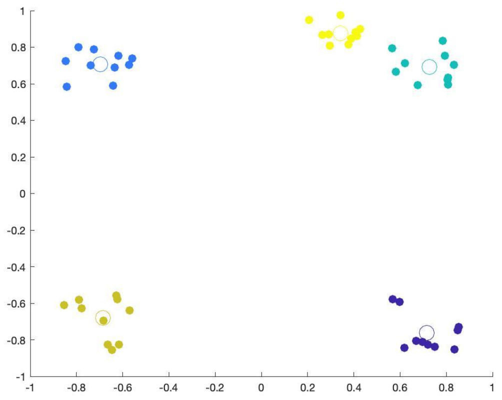


Figure 1.2.6

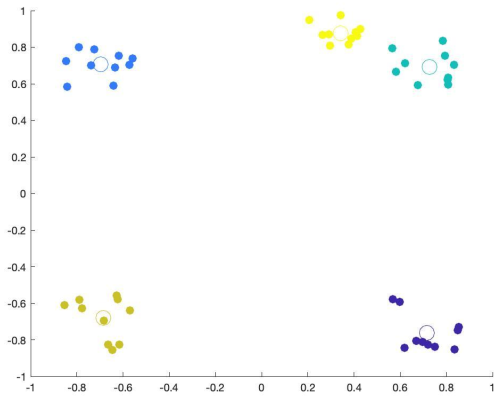


Figure 1.2.7

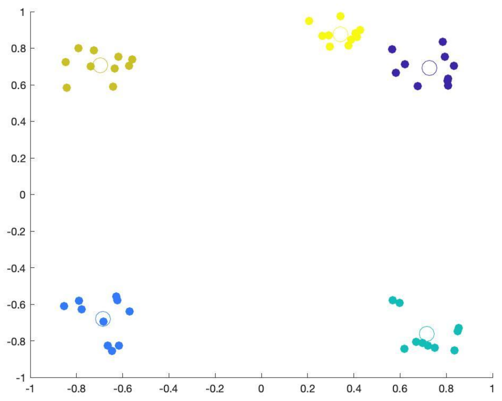


Figure 1.2.8

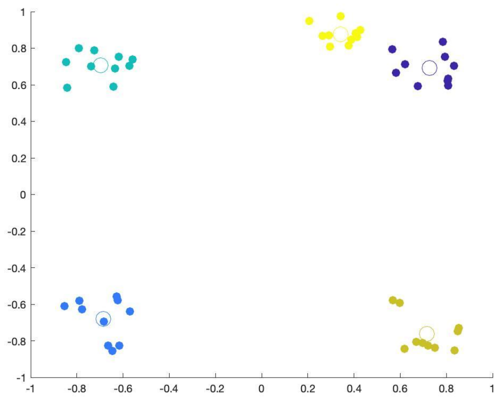


Figure 1.2.9

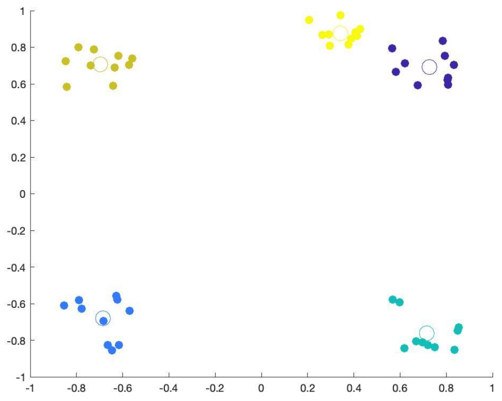
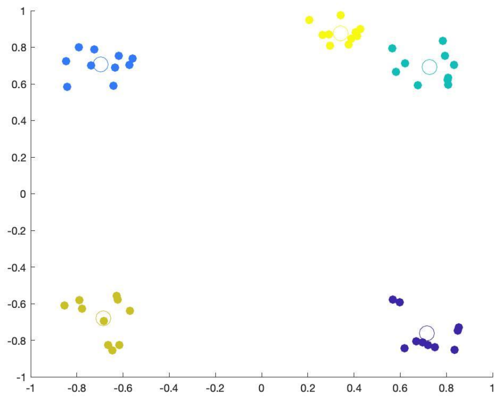
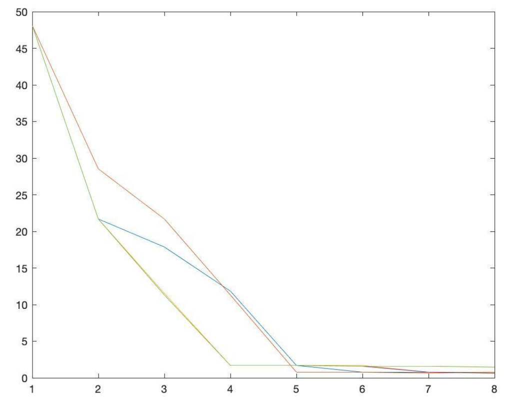
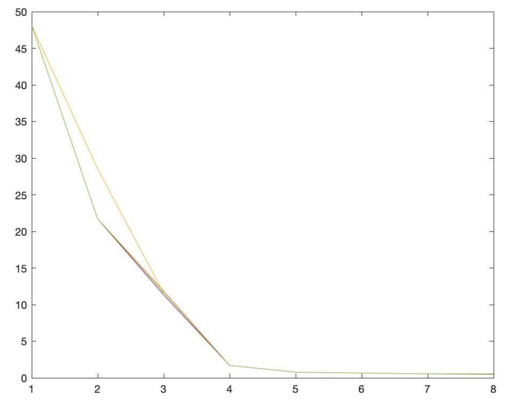


Figure 1.2.10

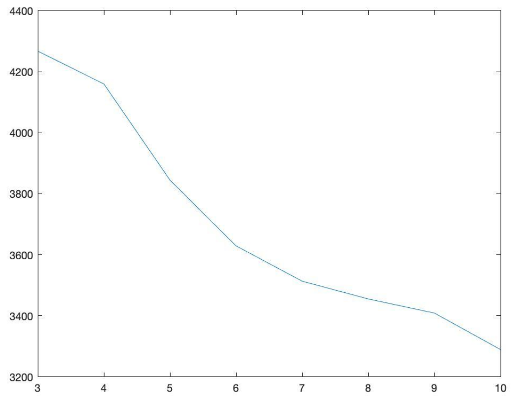


Appendix. Figures

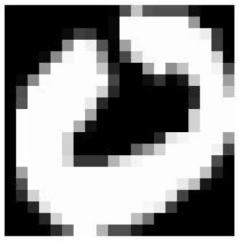
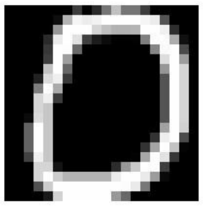
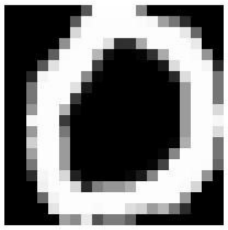
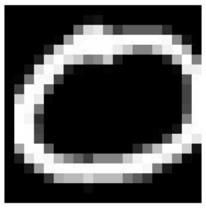
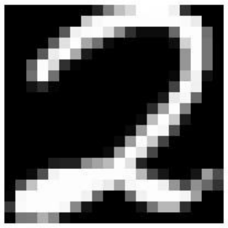
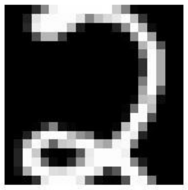
Part II Figures

Figure 2.1: Random Initialization	Figure 2.2: K++ Initialization
	

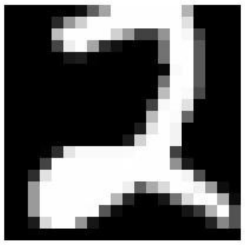
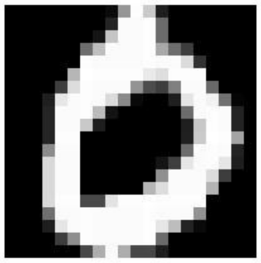
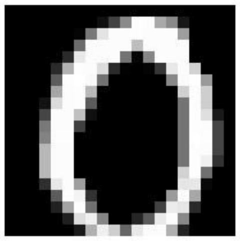
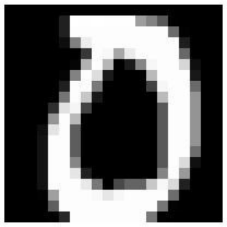
Part III Figures

Figure 3.1	
	

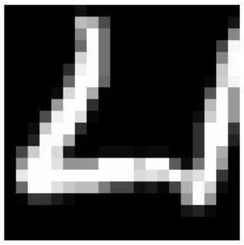
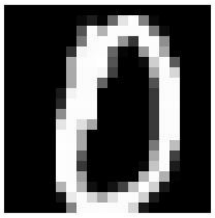
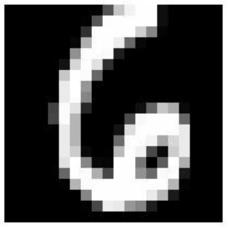
Appendix. Figures

3.2 Cluster 2 images	
Figure 3.2.1 	**Figure 3.2.2** 
Figure 3.2.3 	**Figure 3.2.4** 
3.3 Cluster 4 images	
Figure 3.3.1 	**Figure 3.3.2** 

Appendix. Figures

Figure 3.3.3 	
3.4 Cluster 6 images	
Figure 3.4.1 	
Figure 3.4.2 	**Figure 3.4.3** 

Appendix. Figures

Figure 3.4.4 	**Figure 3.4.5** 
Figure 3.4.6 	**Figure 3.4.7**