GLENN CHIA 1003118 ML Homework 2

1. K-Means [30 points]

1(a) How many clusters there are in the end. (A cluster can “disappear” in one iteration of the algorithm if no vectors are closest to its centroid.)

The number of clusters at the end is 6

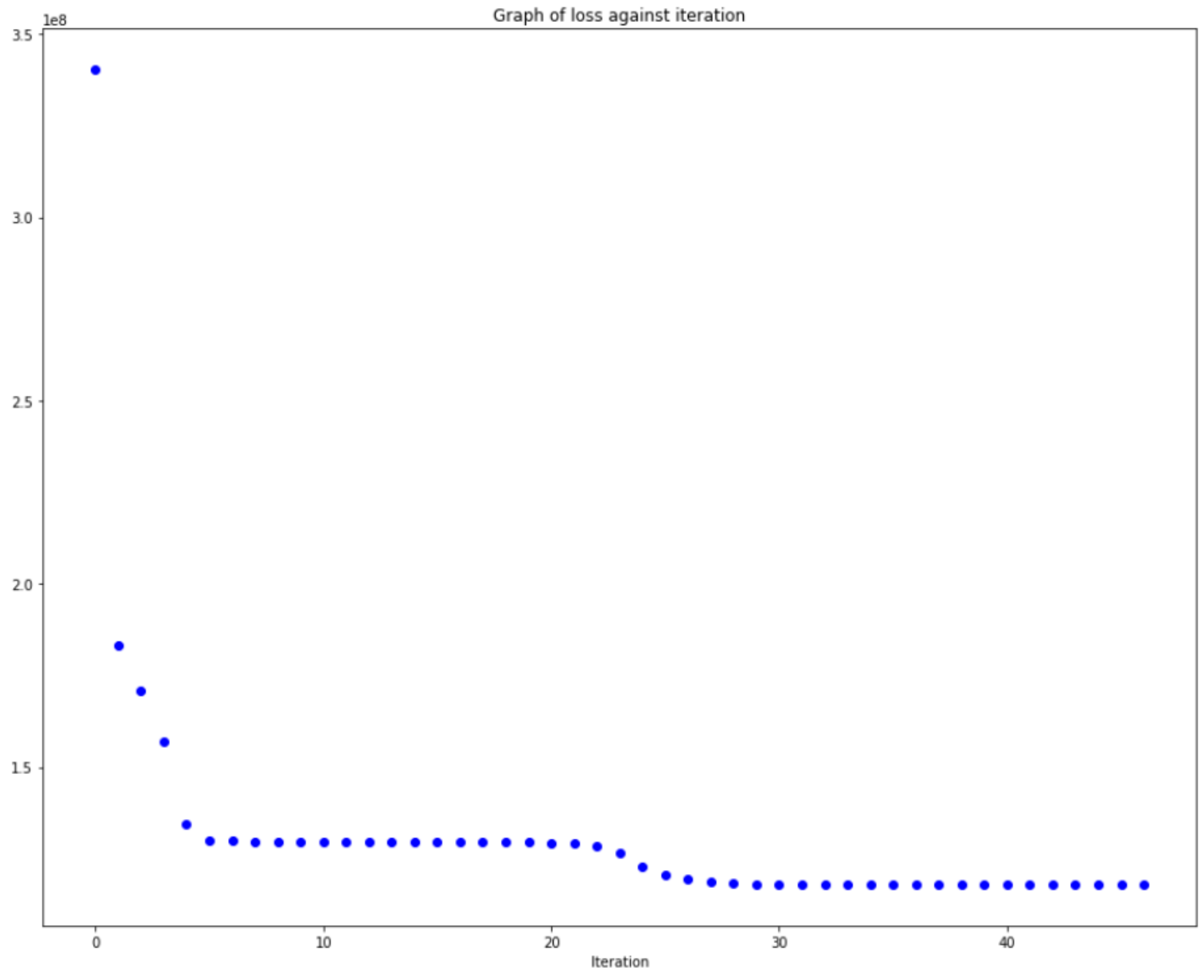
1(b) The final centroids of each cluster.

```
Cluster 0 has centroid [241.2296146  238.62515213 233.86288032]
Cluster 1 has centroid [194.41158657 136.33311389  90.94364714]
Cluster 2 has centroid [136.2655563   61.08973066  10.10385457]
Cluster 3 has no centroid
Cluster 4 has centroid [157.29173273  97.59397508  51.43329558]
Cluster 5 has no centroid
Cluster 6 has centroid [78.92743714 37.10828688 13.07070482]
Cluster 7 has centroid [25.97800232 23.23575423 23.60599063]
```

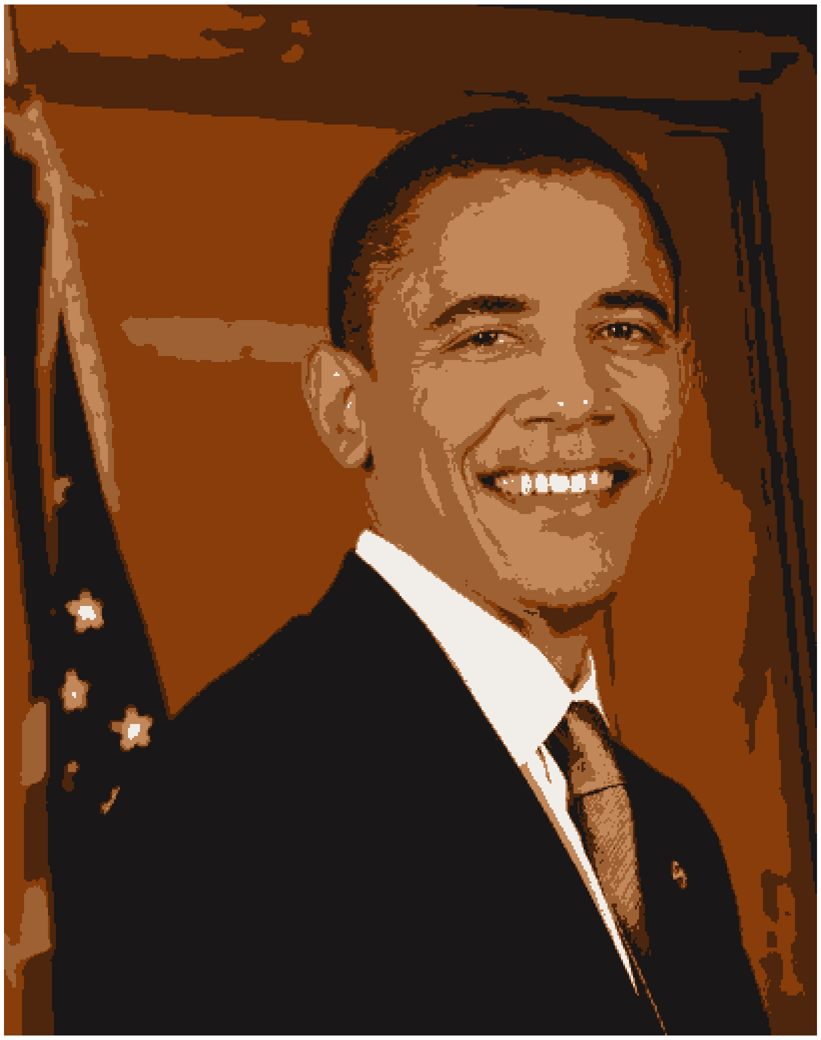
1(c) The number of pixels associated to each cluster.

```
Cluster 0, [241.2296146  238.62515213 233.86288032] has 4930 pixels
Cluster 1, [194.41158657 136.33311389  90.94364714] has 15190 pixels
Cluster 2, [136.2655563   61.08973066  10.10385457] has 52535 pixels
Cluster 3, 0 has 0 pixels
Cluster 4, [157.29173273  97.59397508  51.43329558] has 22075 pixels
Cluster 5, 0 has 0 pixels
Cluster 6, [78.92743714 37.10828688 13.07070482] has 40365 pixels
Cluster 7, [25.97800232 23.23575423 23.60599063] has 74917 pixels
```

1(d) Plot the squared Euclidean distance from each pixel to the nearest centroid after every iteration of the algorithm.



Visualize your result by replacing each pixel with the centroid to which it is closest, and displaying the resulting image



2. K-Mediods [10 points]

- **Figure A** uses *l*∞ distance
- **Figure B** uses Euclidean Distance
- **Figure C** uses Manhattan Distance

I implemented this question in code but below is the hand calculated version

- **Figure A** uses *l*∞ distance
  - Distance from x_0 to x_1 is 4
  - Distance from x_0 to x_2 is 6
  - Distance from x_0 to x_3 is 5
  - Hence the smallest distance is from x_0 to x_1
- **Figure B** uses Euclidean Distance
  - Distance from x_0 to x_1 is $\sqrt{32}$
  - Distance from x_0 to x_2 is $\sqrt{36}$
  - Distance from x_0 to x_3 is $\sqrt{29}$
  - Hence the smallest distance is from x_0 to x_3
- **Figure C** uses Manhattan Distance
  - Distance from x_0 to x_1 is 8
  - Distance from x_0 to x_2 is 6
  - Distance from x_0 to x_3 is 7
  - Hence the smallest distance is from x_0 to x_2

3. K-Means vs K-Medoids [10 points]

What are the advantages and disadvantages of K-medoids, compared to K-means?

Advantage 1 of K-medoids - available for more contexts

Context is an important factor to determine the relevance of K-Means vs K-Medoids. There are certain scenarios which do not allow for K Means clustering and only allows K-medoids. For example, if we want to classify 2 distinct movie genres, `Horror` and `Comedy`, it does not make sense to blend them together to evaluate the "mean" (K means clustering) as there should not be a `Horror-comedy` genre since the genres themselves are vastly different. Thus, in situations where there are 2 distinct groups that should not overlap, the K-Medoids algorithm is the only one that will work effectively.

Thus, K-medoids is more flexible and can be used with any similarity measure

Advantage 2 of K-medoids - Cluster representation

K-medoids can provide a better centroid to represent the cluster. This is best illustrated with an example.

For example: We have a 1-Dimensional clusters [4,5,6,7,8,3000]

If we use K-means, the mean will be 505 which does not represent the bulk of the data points which are around the smaller values.

Alternatively, if we use K-medoids, we get the centroid as 8 which is a better representation for the set as it is closer to the bulk of the points.

Disadvantage 1 of K-medoids - More expensive

The K medoids clustering algorithm requires us to iterate through all the points and calculate the loss for each one. The loss is found by comparing this point to all the other points in the cluster. Hence, the complexity for locating a centroid is $O(n^2)$. Including locating the clusters and the number of iterations, this becomes $O(i * (n^2 + n*k))$ where i is the number of iterations and k is the number of clusters.

On the other hand, during the search for centroids, the K means algorithm does not require us to iterate through all the data points twice. Instead, we just have to iterate through the data points for each cluster to compute the mean. Hence, the complexity of this operation is O(n). Including the number of clusters and iterations will produce a complexity of O(i * (n + n * k)) which is simplified to `O(i*n*k)`

	K-Medoids	K-Means
Complexity of computing the centroid	$O(n^2)$	O(n)

To summarize, from the table we can see that the cost for K-medoids is higher and it will take a longer time for clustering