CM146, Winter 2019
Problem Set 4: Clustering and PCA
Due Mar 16, 2019

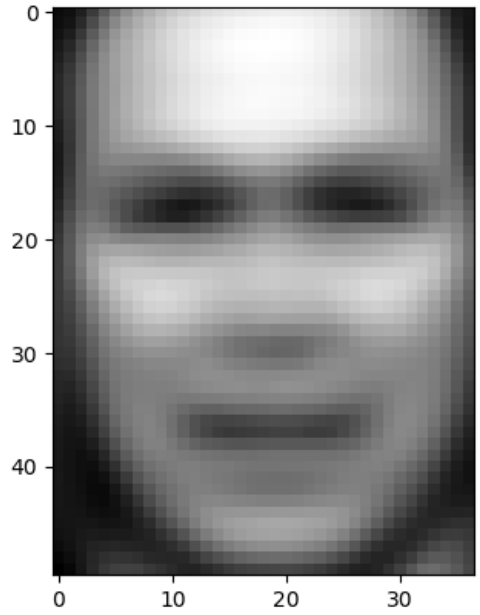
Jingjing Nie

1 Problem 1

(a) Problem 1a

Solution:

After using "show_image()" to plot several face images, I have then plotted the averaged face image, as shown below. Since this image has taken average values of all data points, which all have a lot varying features, the generated image is not well defined and does not have a clear shape. But the basic outlines of the faces can be recognized.



(b) Problem 1b

Solution:

The top twelve eigenfaces are:

These twelve plots generated are all different from each other in terms of darkness and lights, and facial features and shapes. Since eigenfaces are obtained from the covariance matrix, these top twelve images can then cover the major variances in human faces data stored. Therefore, the reason why they are picked as the eigenfaces is that their facial features are more representative of human faces in the training set, so using these plots can better generate typical and average human faces in the training set data.

(c) **Solution:**

The images generated are:

$l = 1$

$l = 10$

$l = 50$

$l = 100$

$l = 500$

$l = 1288$

The above plots have shown that the images are less clear with smaller l values since we have projected the original data to lower-dimensional spaces. This will benefit us from computing efficiency as well as storage efficiency, but it will also make the images too unclear and lose a lot of important features. For instance, the first figure with l as 1 only has the most basic features of human faces due to the lack of training features, and it is nearly impossible to distinguish between these faces. Therefore, differing the values of l is pretty effective in impacting the recognition of the images plotted, and in order to be able to distinguish between faces, we cannot use a l values that is too small, though higher l values might lead to the trade off of computing and space efficiency.

2 Problem 2

(a) Problem 2a

Solution:

In k-means, we attempt to find k cluster centers and n cluster assignments such that the total distance between each data point and the nearest cluster center is minimized. Therefore, if k is no longer set as a fixed value, we can set $k = n$, which suggests that there will be n cluster centers and each data point will form its own cluster. In this case, the minimum distance between each data point and the corresponding cluster center will be 0. So the minimum value of $J(c, u, k) = \sum_{i=1}^{n} ||x_i - \mu_{c^{(i)}}||^2 = 0$, when the value of k is n, the value of $c^{(i)}$ is i, and the value of μ_j is x_j, since each data point is in its own cluster, with zero distance away from the cluster center.
This is a bad idea, because assigning each data point as its own cluster cannot help us group data points that are similar together and thus cannot give us a lot useful information.

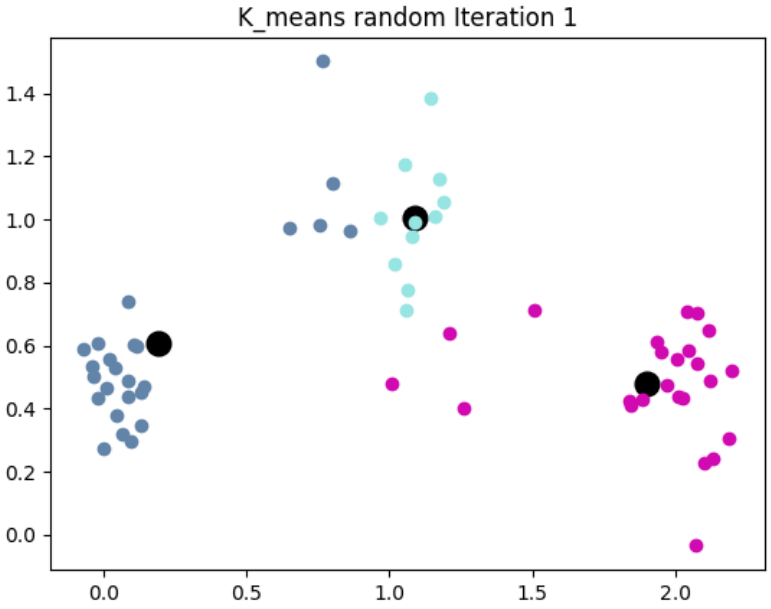
(b)

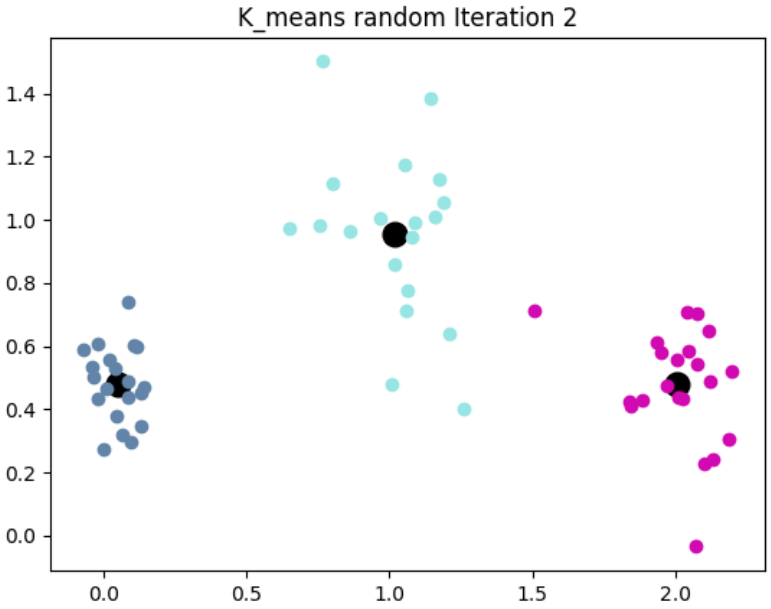
(c)

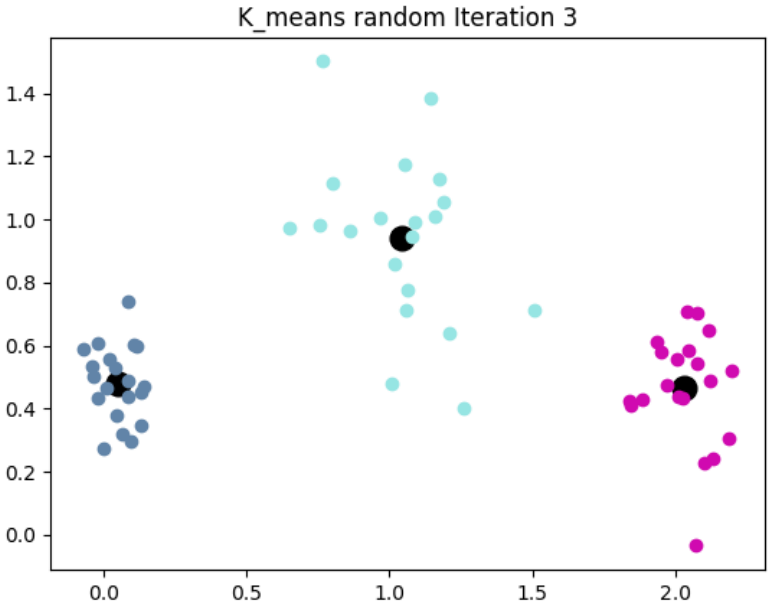
(d) Problem 2d

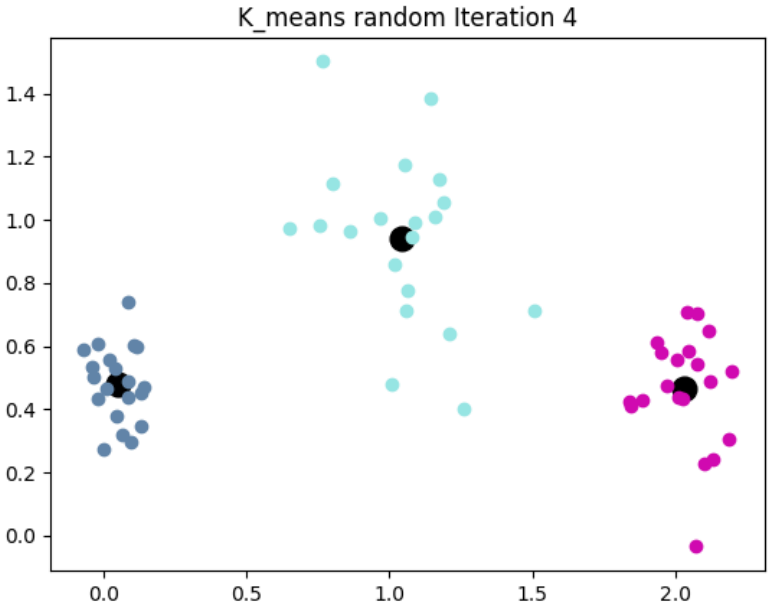
Solution:

The plots for the k-means cluster assignments and corresponding cluster centers for each iteration when using random initialization are shown below.


K_means random Iteration 1
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0


K_means random Iteration 2
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

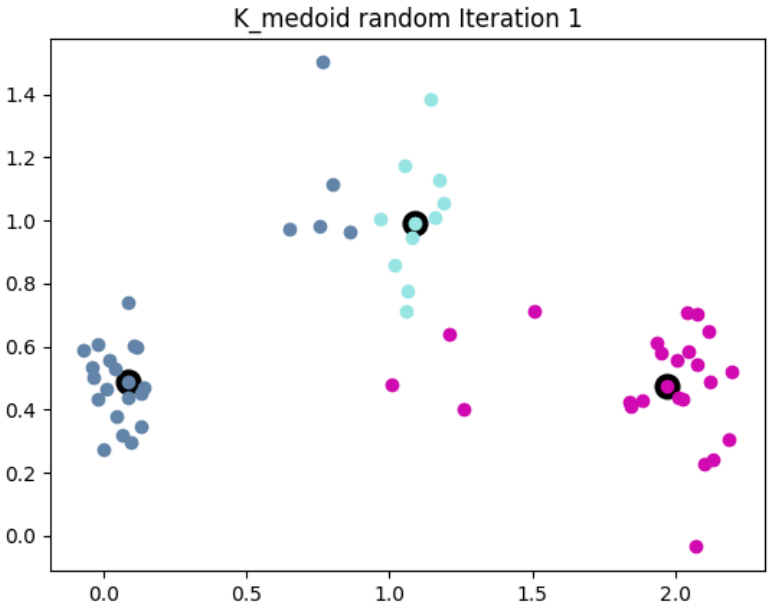

K_means random Iteration 3
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

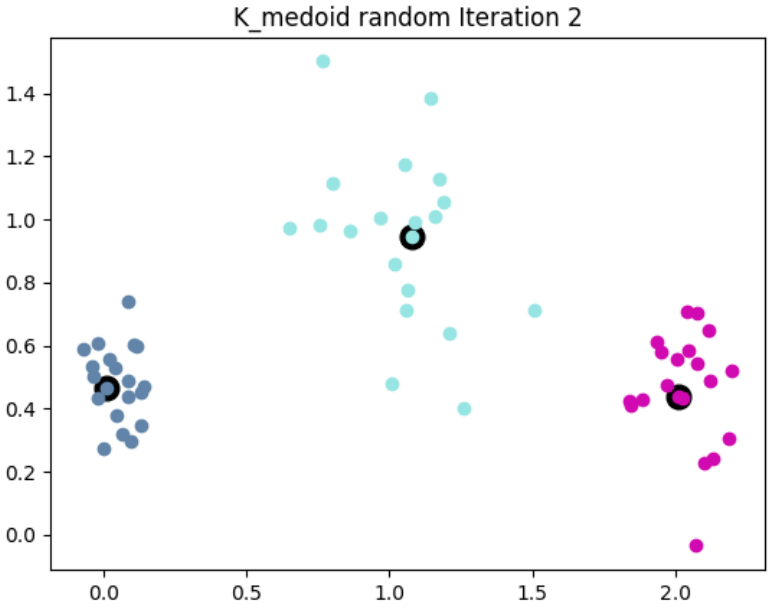

K_means random Iteration 4
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

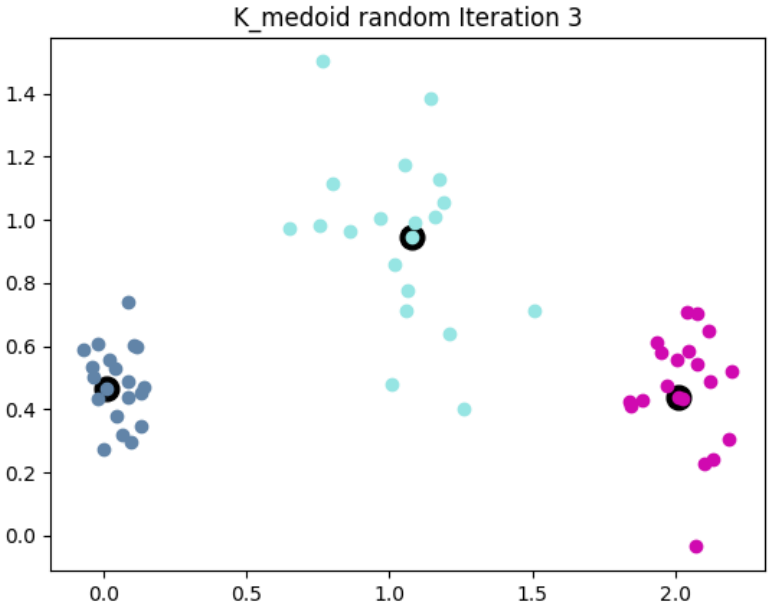
(e) Problem 2e

Solution:

The plots for the k-medoids cluster assignments and corresponding cluster centers for each iteration when using random initialization are shown below.



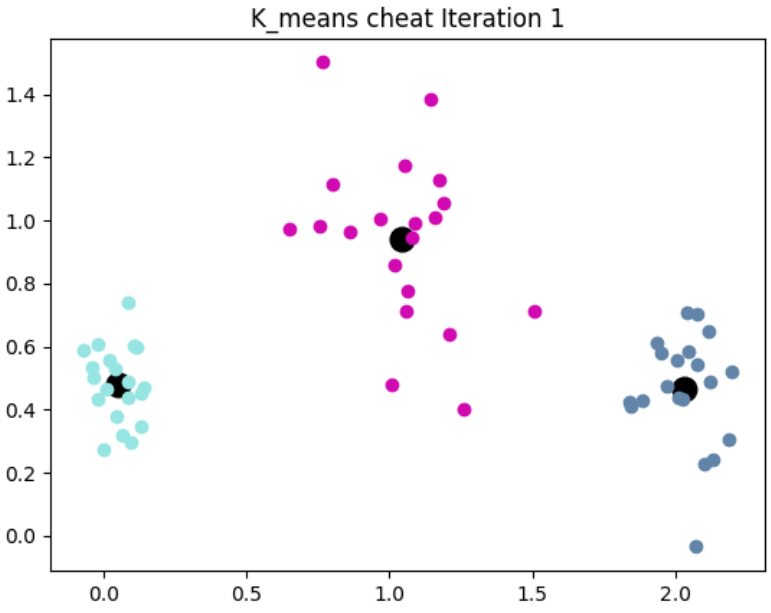

K_medoid random Iteration 2
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

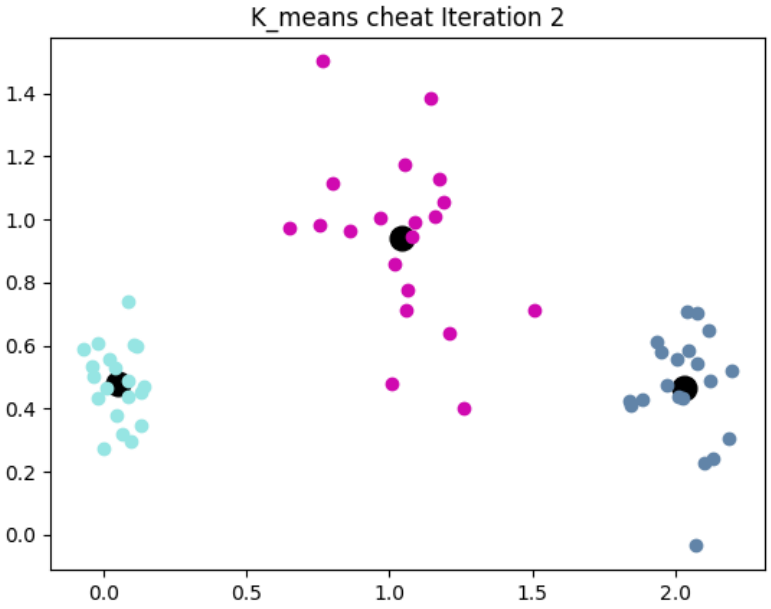

K_medoid random Iteration 3
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

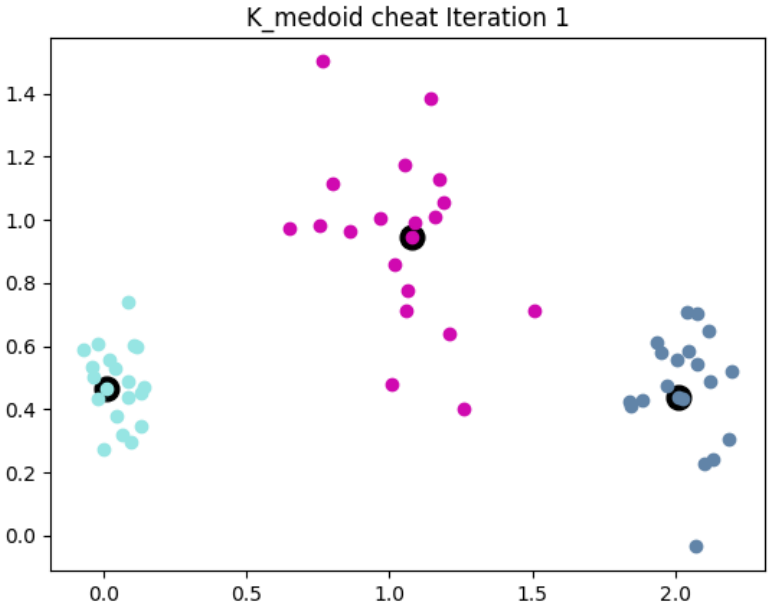
(f) Problem 2f

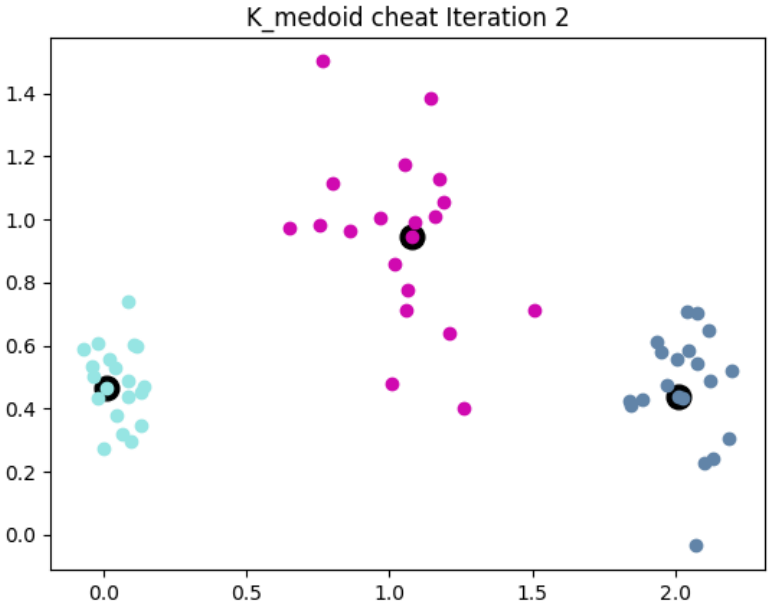
Solution:

The plots for the k-mean and k-medoids cluster assignments and corresponding cluster centers for each iteration when using cheat initialization are shown below.




K_means cheat Iteration 2
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0


K_medoid cheat Iteration 1
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0



3 Problem 3

(a) Problem 3a

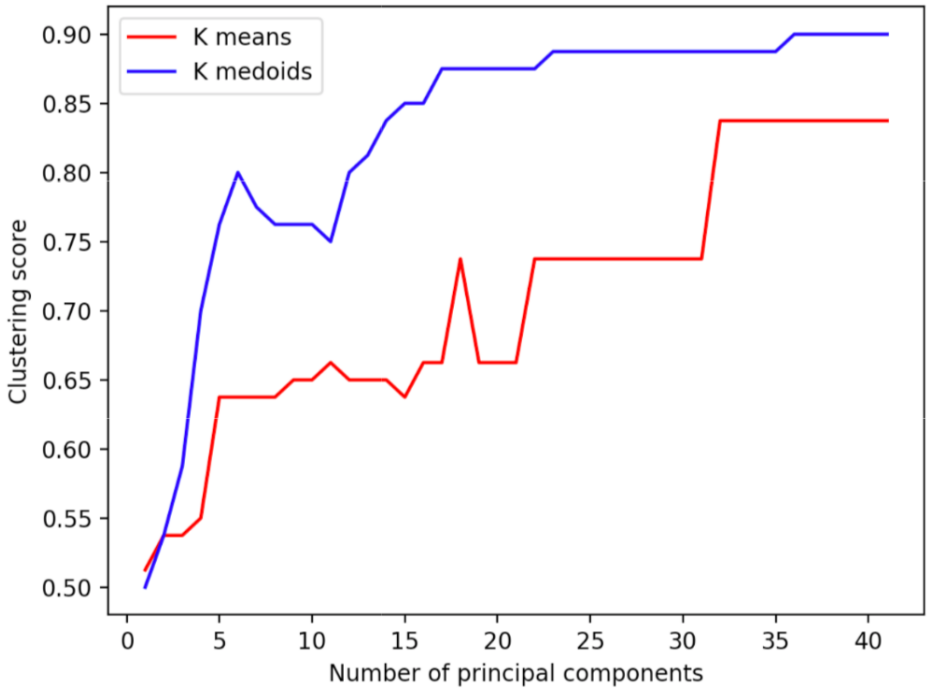
Solution:

	average	min	max
k_means	0.625625	0.5875	0.66875
k_medoids	0.638125	0.59375	0.725

According to the results, k_medoids can generate better performance for all average, minimum, and maximum aspects. But the runtime for k_medoids is longer than that of k_means.

(b) Problem 3b

Solution:



The results suggest that both of the two methods will have higher clustering scores as the number of components increases. Also, the results show that the scores for both the two methods will reach constant levels as the number of components reach values above around 30. Moreover, in general, the clustering scores achieved by *K_medoids* method is higher than that achieved by *K_means*. Therefore, using *K_medoids* can help generate better performance.

(c) Problem 3c

Solution:

In order to find out the pair that clustering can discriminate very well and another pair that it finds very difficult, I used a nested for loop to iterate through all the data points and to check for the pair that has the highest score, and the pair that has the lowest score.
The two pairs with the highest and lowest clustering scores are presented below respectively:

Best

Worst

From the two above pairs, it can be seen that the two faces have big differences in terms of features in the pair that has the higher score. Therefore, this pair can be easily distinguished from each other and has high score. On the other hand, the two faces in the other pair share similarities that can make them hard to be distinguished.