Problem Set 4

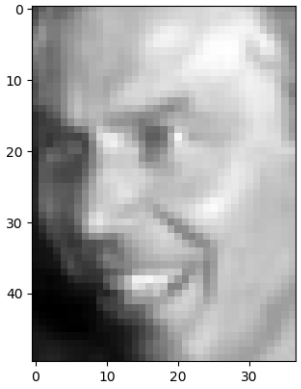
Shalin Shah

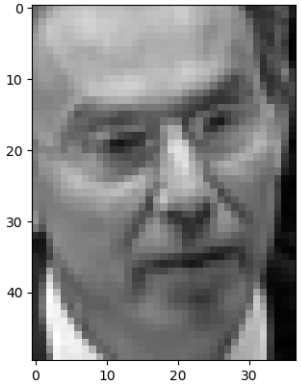
March 15, 2020

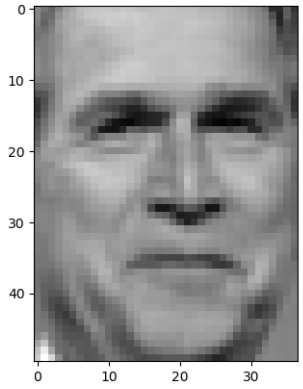
1 PCA and Image Reconstruction

1.a Seeing the data

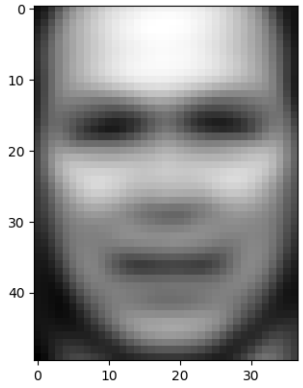
For this part, we were to plot all of the images using show_images and then compute the mean of all the images and plot that as well. Here are the first three images:







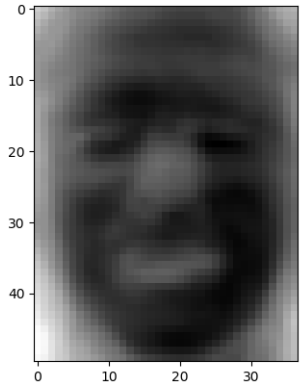
And then the average is found here:



What we can see here is that the picture that comes out when averaged has no real eyes, nose, or mouth that corresponds to a single person. In general, most people have a very similar structure of their face, but very different ways that the specifics of their face works out. So, we see the specifics are either blacked out (like the eyes) or smoothed out (like the forehead) but have a general look of a human face.

1.b Seeing the data pt. 2

Here is the output of the first column of the principal component matrix:



Now, here is the output of the first twelve columns of the principal component matrix (using the function given):

These are likely the top eigenfaces because they contain the most amount of variance compared to either eigenfaces. This means that they give more information than the other faces and are most likely to be reconstructed into the image that we want it to correspond to. Basically, they're the most unique

in a sense and thus are the ones that we think are better for our purposes.

1.c Seeing the data pt. 3

Here, we are performing PCA with different l components, where l is the total number of principal component that we should retain. We chose values from the list given, and when we perform plotGallery() on this, we get the following six pictures:

From these pictures, we see quite obviously that as we increase the l value (which means that we keep more dimensions of the original dimensionality), the more quailty the reconstructed picture is (meaning the closer it is to the original picture). This makes sense, since losing dimensionality means that we're losing imformation about the original image. Commenting on the effectiveness of differeing values of l with respect to facial recognition, I can say that increasing the l value leads to having better facial recognition since we retain more information about the original image.

2 K-means and K-Mediods

2.a K-Means Objective Function

The minimum possible value of $J(\mathbf{c}, \mu, k)$ comes when $\mu(i) = \mathbf{c}(i)$. This is because when this happens, the norm of the difference of the two values squared equals 0 which is the minimum for a squared value. Since here the total cost would be zero for every cluster (which consists of one point), minimizing the function is a bad idea.

2.b Implement methods in TODO in Cluster and ClusterSet classes

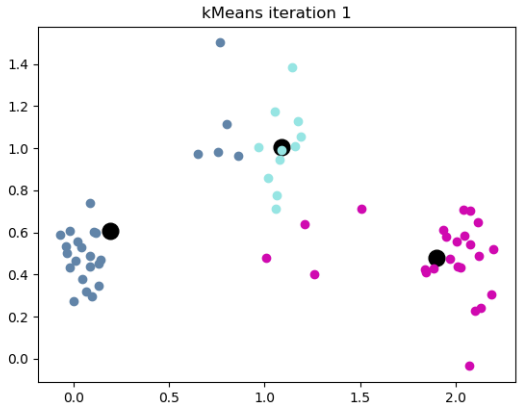
Done.

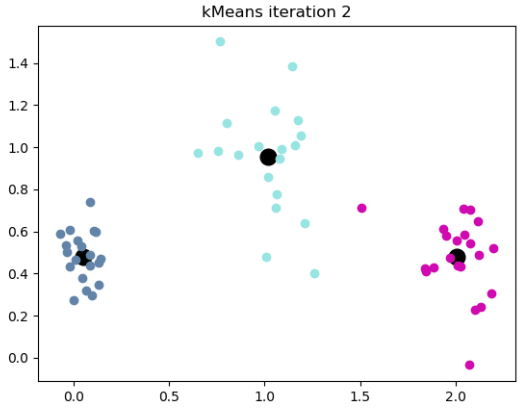
2.c Implement random_init() and kMeans()

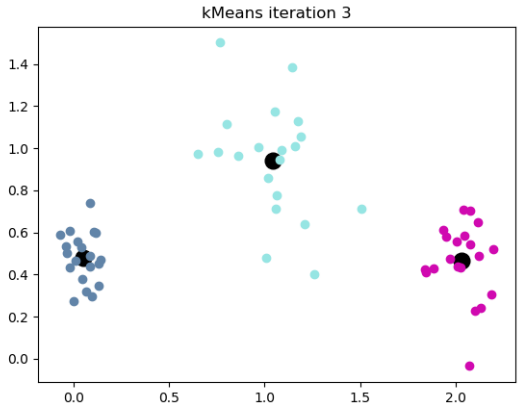
Done.

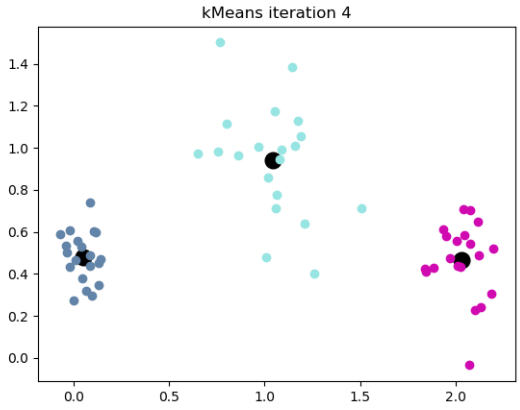
2.d Use this for k-means on toy dataset

Here, we're trying to create three different clusters, and we see how this happens. Here, the black dot serves as a center for each iteration (which is included at the top as the title):



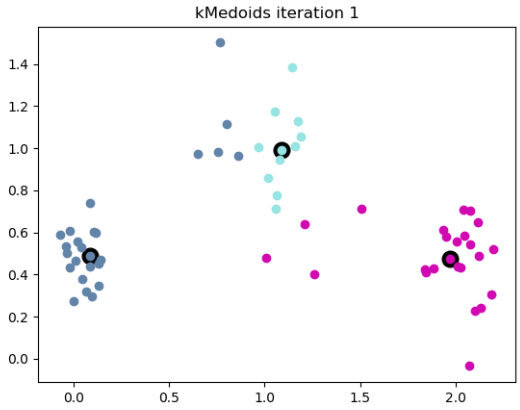


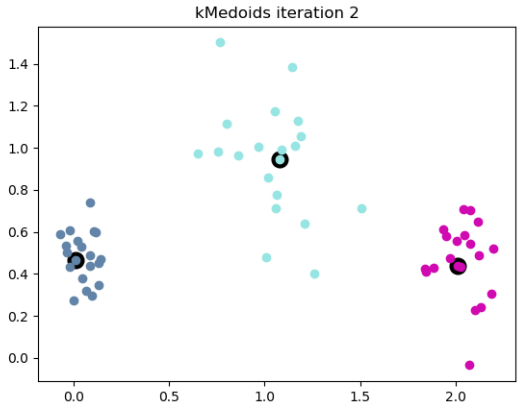


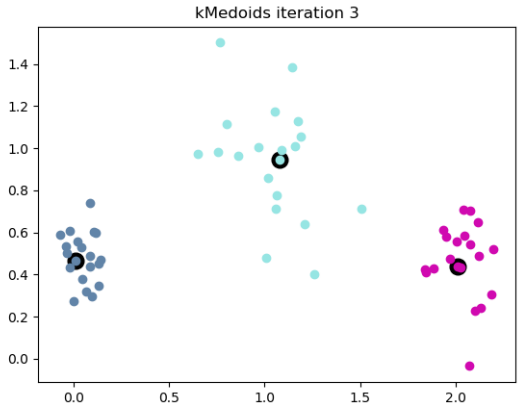


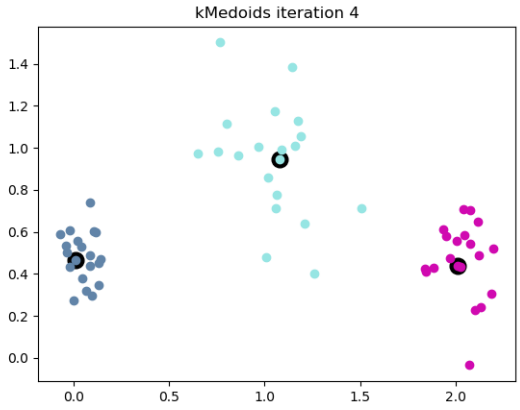
2.e Implement kMediods and use kAverages as a helper function. Then plot k-mediods clustering for each iteration.

Here, we're trying to create three different clusters, and we see how this happens. Here, the black dot serves as a center for each iteration (which is included at the top as the title):


kMedoids 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

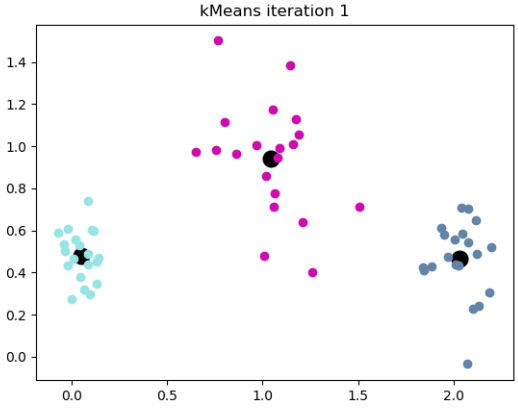

kMedoids iteration 2
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

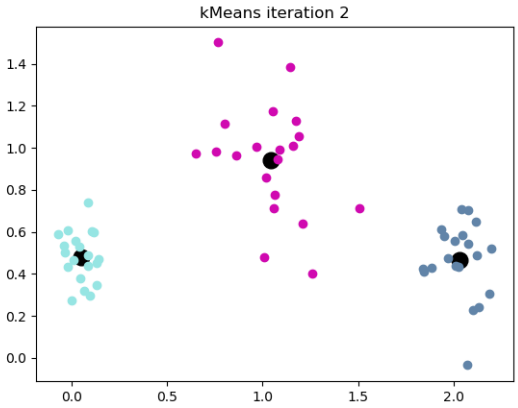




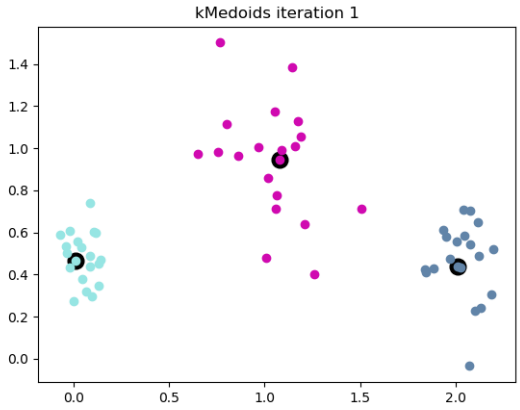
2.f Now we explore the effect of initialization by implementing and using cheat_init() and then plot the k-means and k-medoids for each iteration.

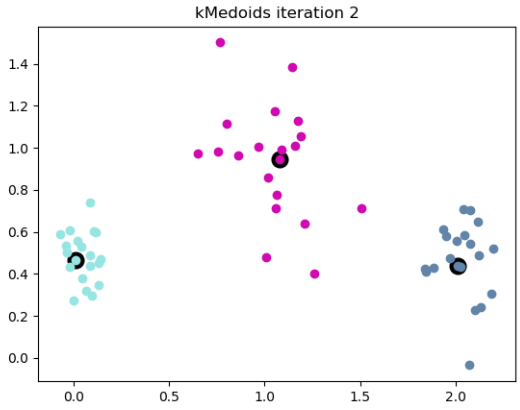
For kMeans:





For kMedoids:





We see here that if we have an initialization that the amount of time necessary for each set of clusters to be properly classified is drastically reduced (from four to two for each case).

3 Clustering Faces

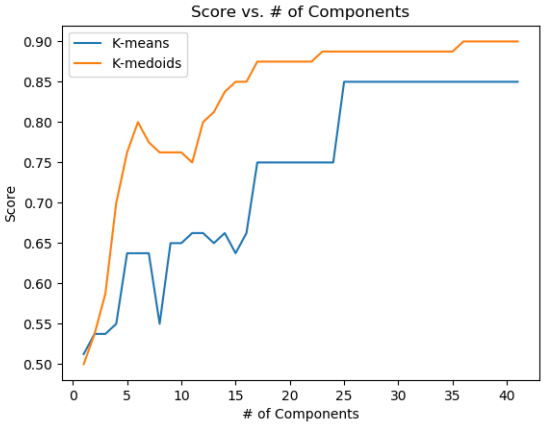
3.a Use k-means and k-medoids with k=4 and modified dataset to fill out the chart.

After running the algorithms and recording their average and minimum/maximum performances, I got the following values:

	average	min	max
k-means	0.66375	0.5875	0.775
k-medoids	0.659375	0.5125	0.75

From a performance and accuracy stand point, it seems that k-medoids slightly outperforms k-means. k-medoid's runtime was 0.2356 on average while k-mean's runtime was 0.3340 on average. The better performance could be due to the fact that centroids are chosen in a way that's more resistant to outliers and thus the algorithm has less work to do if (and most likely when) there are outliers.

3.b Explore effect of lower-dimensional representations on clustering performance.



Here, we see the graph of the clustering score versus the # of principle components (score on the y and # of components on the x). We see that K-medoids, in general (except for the smallest case), has a better score than K-means when it comes to our clustering here. We saw this earlier as well; k-medoids performs slightly better than k-means. We also see another trend that we saw earlier; as the value of l increases, so does the score. This makes sense because the more dimensions we use, the more information we have that we can use and don't lose out (compared to earlier dimensions). Finally, we see that the score-dimensionality relationship slightly tapers off after components hits around 25, and this is likely because we don't need that more information anymore to accurately get what we need.

3.c Find two pictures that clustering discriminates well and two that clustering doesn't discriminate well.

Here is the pair that the clusering does discriminate well:

Here is the pair that the clusering does NOT discriminate well:

What I did for this part was let my algorithm pick which pair was the best and which was the worst. I looped through and ran kMedoids (not kMeans since kMedoids runs better as we've figured out previously in this project) on every set of pair of pictures and checked the score to see if it was the minimum or maximum score we've achieved thus far and if so, set the min/max picture pairing to that pair. From here, we get the two pairs as shown above. It seems that individuals with similar face structures, complextions, and rotations in space are well classified against each other but ones that don't have those similarites are not.