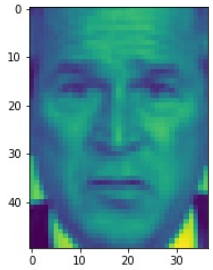
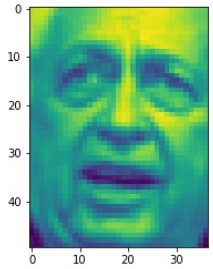
CM146, Fall 2017
Problem Set 4: Clustering and PCA
Due Dec 10, 2017

Author: Shivraj Gill

1 Problem 1

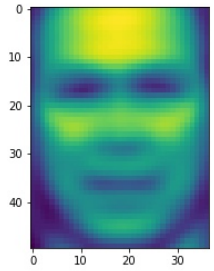
(a) Problem 1a **Solution:** Solution to problem 1a



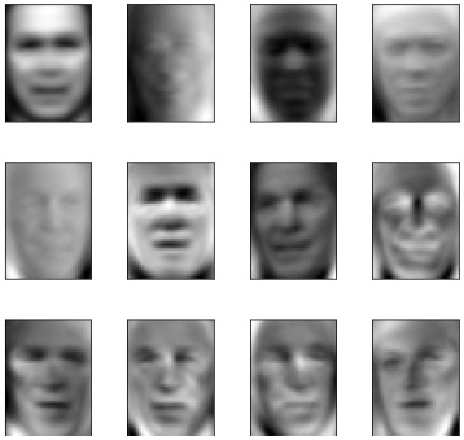


We see that the image of the average face looks a lot blurrier than the actual images. This makes sense since the average face is the mean pixel in each image, so facial components like the nose and mouth are blurry due to variation in each image. Nonetheless, we can see a general outline of a face and the facial components, since each image shares those components.

Figure 1: Average Face



(b) Problem 1b **Solution:** Solution to problem 1b



We see that the beginning eigenfaces (first and second row) are blurrier and differ greatly from one another whereas the later eigenfaces (last row) look more defined and similar since they represent less variation among the features. The first eigenface seems to capture the variation in the nose, mouth and eye as well as the overall outline of the face. Subsequent eigenfaces take into account features such as lighting, facial components, and a combination of the two. Overall, the eigenfaces seem to focus in on a fewer set of features (lesser variation) for smaller eigenvalues (last row). These are selected as the top eigenfaces since they account for the most variation in our data set.

(c) Problem 1c **Solution:** Solution to problem 1c

Figure 2: $l = 1$

Figure 3: $l = 10$

Figure 4: l = 50

Figure 5: l = 100

Figure 6: l = 500

Figure 7: l = 1288

As we increase the value of l (i.e the number of principal components), we get a clearer representation of the first 12 images in our dataset. For fewer principal components (l = 1, 10, 50), the faces are difficult to distinguish. For l > 100, the faces become increasingly more distinguishable. This makes sense since increasing l allows us to represent more variation in our images.

2 Problem 2

(a) Problem 2a **Solution:** Solution to problem 2a

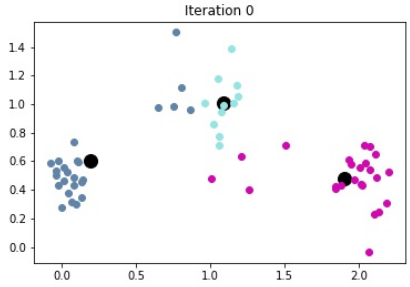
The minimum possible value of J(c, μ, k) would be 0 when k = n, c_i = i, and $\mu_i = x^{(i)}$. This basically means each of our data points is a centroid and thus J = 0, since the distance between a data point and it's respective centroid would be 0.

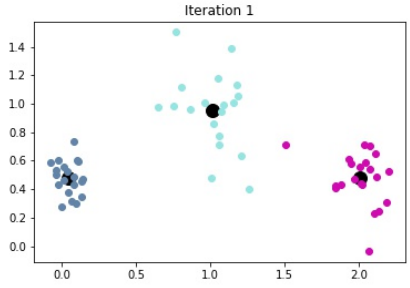
(b) Problem 2b **Solution:** Solution to problem 2b

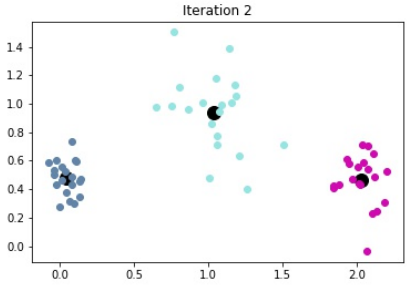
(c) Problem 2c **Solution:** Solution to problem 2c

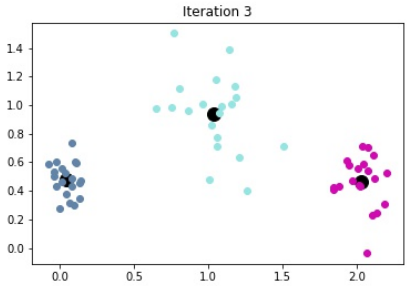
(d) Problem 2d **Solution:** Solution to problem 2d

The K-means algorithm had 4 complete iterations. Most notably, the last two iterations look the same since our data points remain in the same cluster.



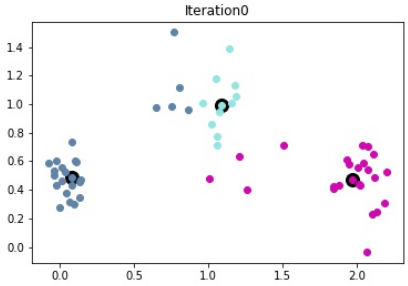


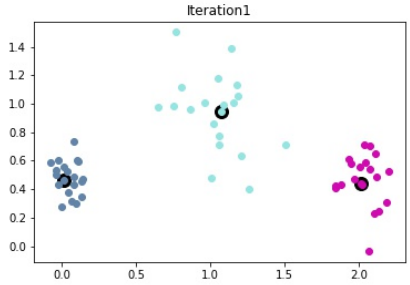


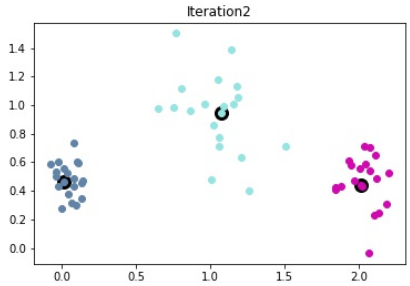


(e) Problem 2e **Solution:** Solution to problem 2e

The K-Medoids algorithm had 3 complete iterations. This time the black circled points represent medoids which are actual points in their respective cluster. Again, for the last two iterations, our plot looks the same since the data points remain in the same cluster.

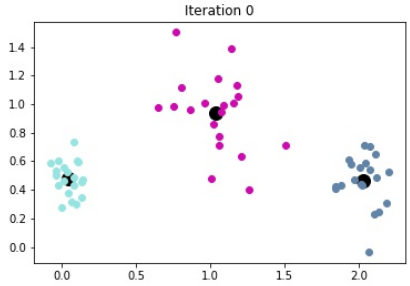


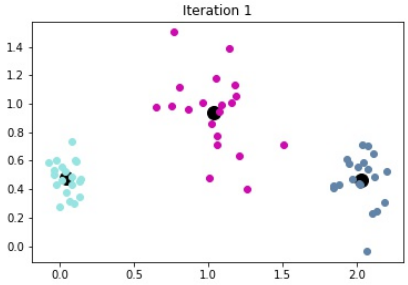




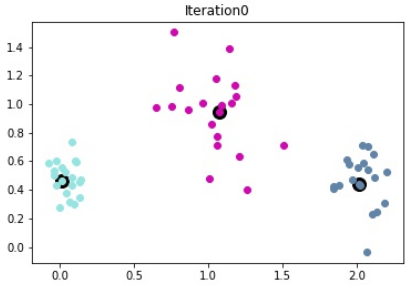
(f) Problem 2f **Solution:** Solution to problem 2f

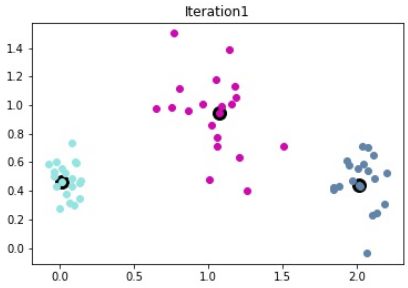
K-means with cheat initialization:





K-medoids with cheat initialization:





Both K-means and K-medoids converge faster with cheat initialization since both algorithms get a head start in finding the appropriate prototype for the cluster. This is because cheat init finds the initial medoids of the clusters based on the given labels. Both algorithms converge after 1 iteration since the second plot for each algorithm looks identical to the first plot.

3 Problem 3

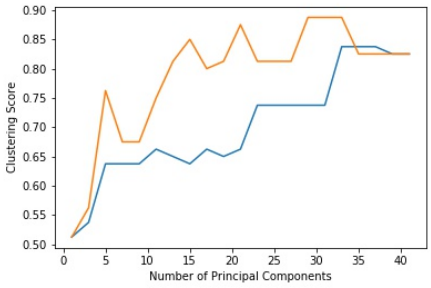
(a) Problem 3a **Solution:** Solution to problem 3a

After running K-means and K-medoids 10 times with randominit on the specified classes of images, we get the following results:

Algorithm	Average	Worst	Best
K-Means	0.6613	0.5875	0.775
K-Medoids	0.6531	0.5188	0.7625

The K-means algorithm took about an average of 0.157 seconds to run and K-medoids took about an average of 0.217 seconds to run. K-means performs slightly better on average, and has higher minimum/maximum performances.

(b) Problem 3b **Solution:** Solution to problem 3b



The orange line represents the cluster score for K-Medoids and the blue line represents the cluster score for K-Means.

We see that, overall, the cluster score for K-Means and K-Medoids increases as the number of principal components increases from 1 to 33. This makes sense since more principal components allows us to represent more features in our two image classes, thus we have more flexibility on separating the two classes. Note that both algorithms plateau after 33 principal components.
Another important observation is that K-medoids performs better

than K-Means for the most part. This might be due to the fact that we have outliers in our dataset that are affecting our K-means algorithm. Since K-medoids is robust to outliers, it consistently performs better than K-means.

(c) Problem 3c **Solution:** Solution to problem 3c

To find the most distinguishable and indistinguishable pair, I used K-Means, and ran it 10 times for each pair with randominit. For the most distinguishable pair, I chose the pair that gave me the best average performance. I found that the image classes [6, 16] were the most distinguishable and the highest average performance score, 0.981. We see that these two faces differ greatly by complexion and by key facial components, most notably a headband.

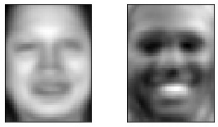


Figure 8: Most distinguishable

Figure 9: least distinguishable

For the most indistinguishable pair, I chose the pair that gave me the worst average performance. I found that the image classes [4, 5] were the most indistinguishable and the lowest average performance score, 0.508. We see that these two faces are very similar in both complexion and facial components, most notably the wrinkles.