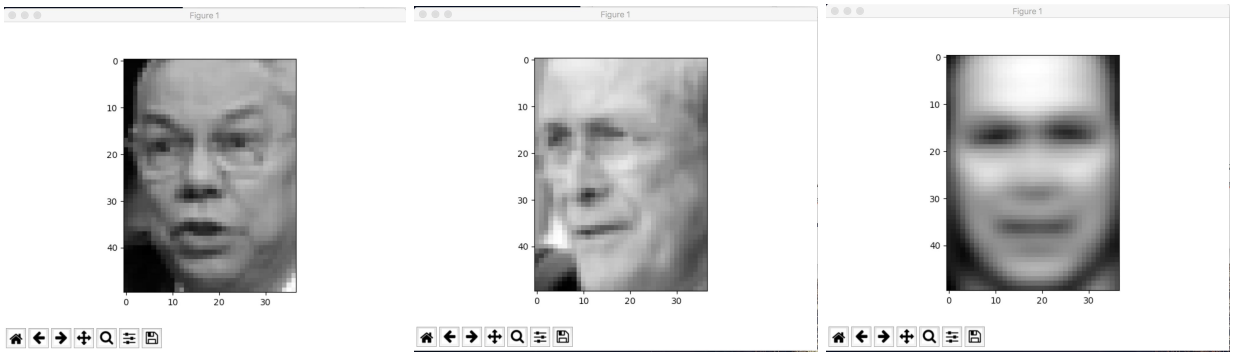
Henry (Ziheng) Yang
204584728

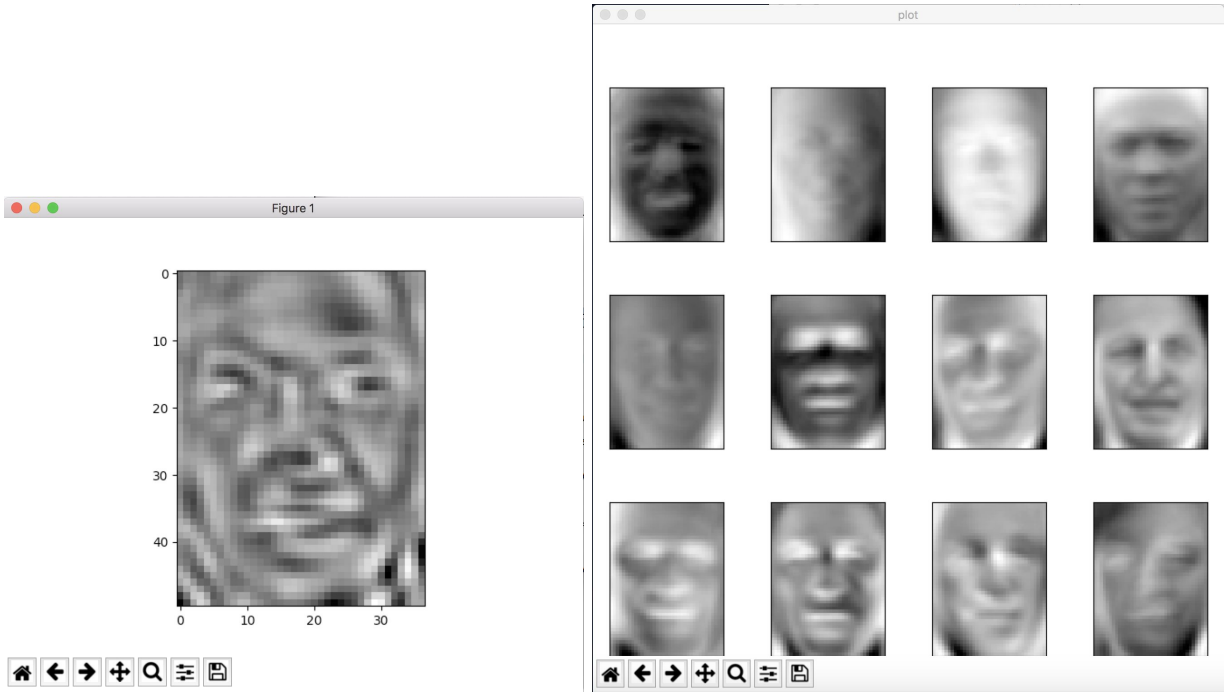
CS 188 - Pset 4

Problem 1A



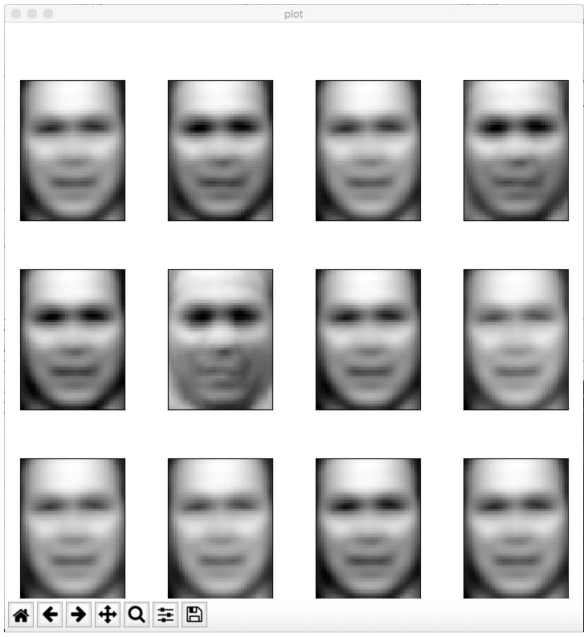
This average face has some distinctive features - namely, it shows us the average location of the eyes, nose, mouth, and general outline of the face. The chin and cheekbones are also easily identifiable.

Problem 1B

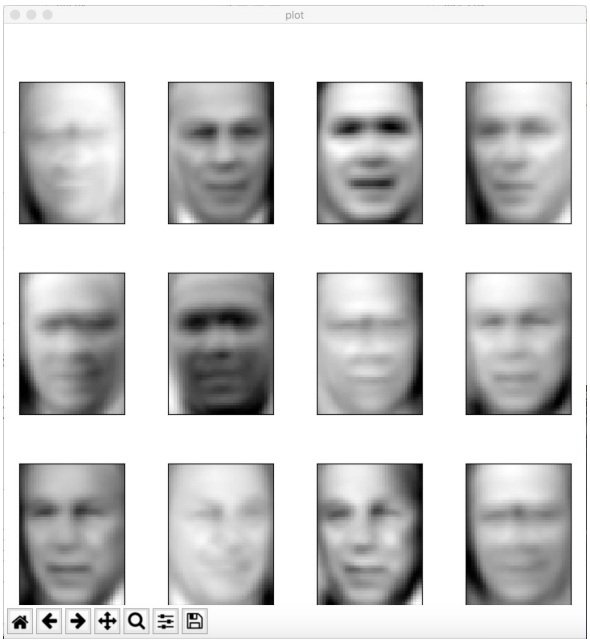


The 12 top eigenfaces should represent the features that carry the most variation. From my observations, it seems some of the top eigenfaces capture the lighting differences in the images, and the subsequent ones are all very different from one another, yet none of them looks particularly like a specific individual. This makes sense as these eigenfaces should carry the most information about the faces in the dataset, so it shouldn't be particularly biased towards one face.

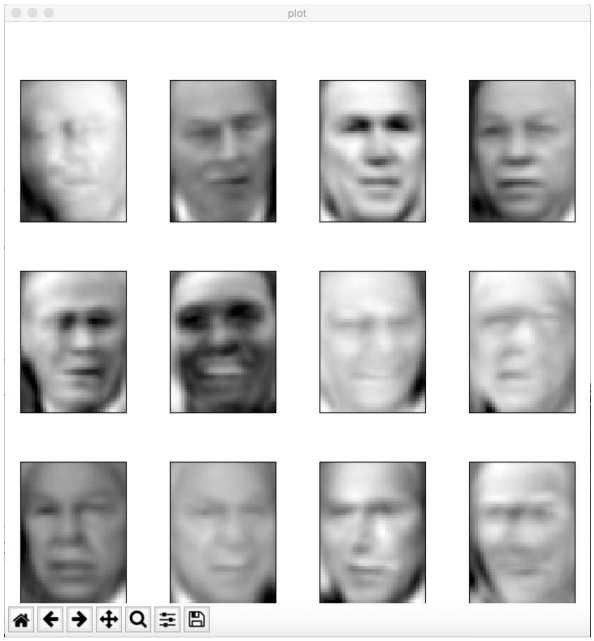
Problem 1C



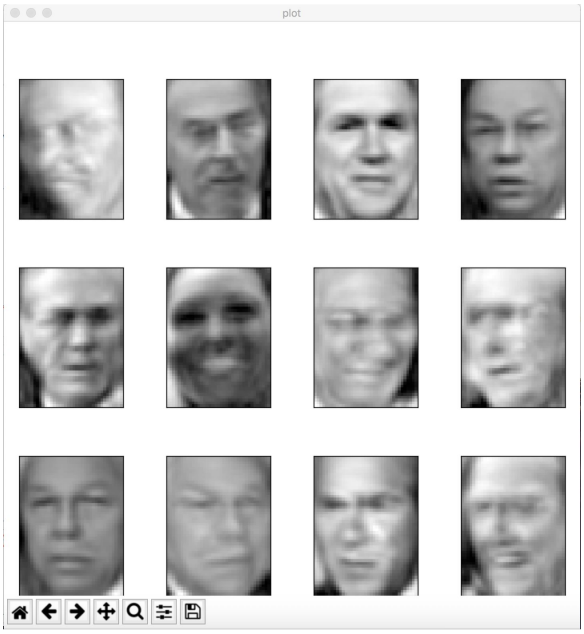
L = 1



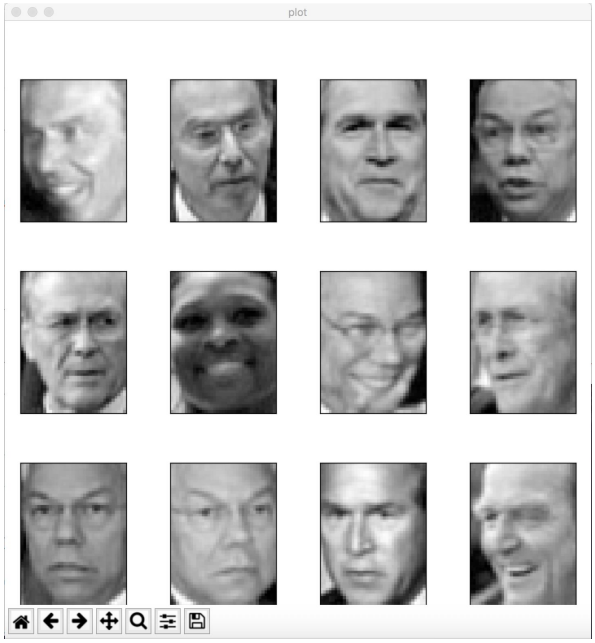
L = 10



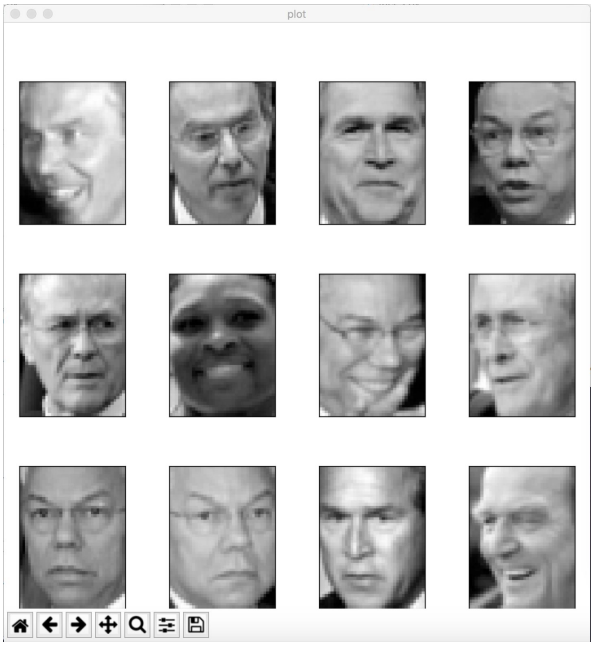
L = 50



L = 100



L = 500



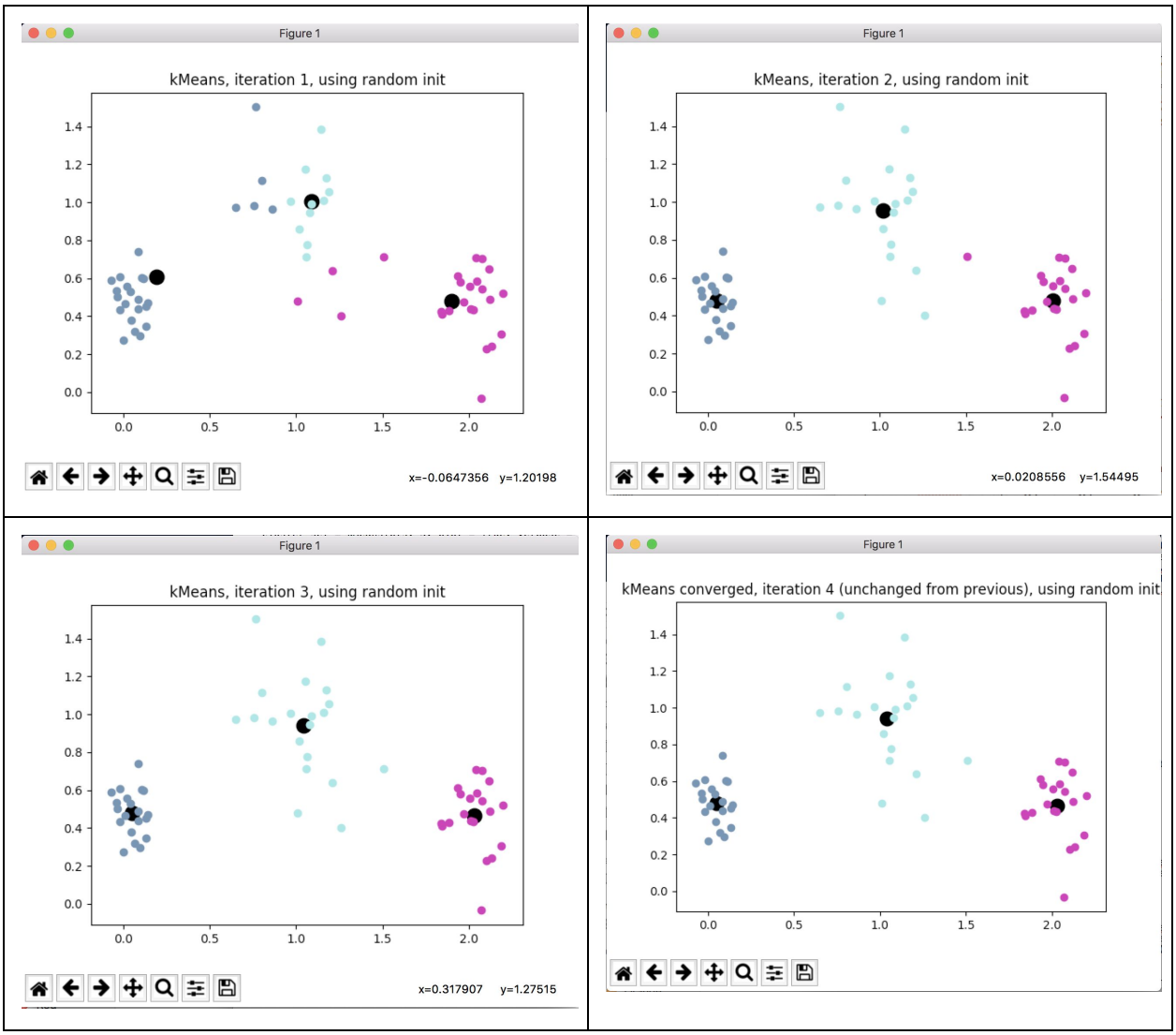
L = 1288

As the value of L increases, the noise in the image decreases and the image looks more clearly like a specific face. Larger values of L seem to make an image more recognizable, but it comes at the cost of processing more data. However, there is a quick drop-off in return for using more eigenfaces. The improvement from L = 100 to L = 1288 is marginal, which means that over 1000 eigenfaces can be discarded when conducting facial recognition.

Problem 2A

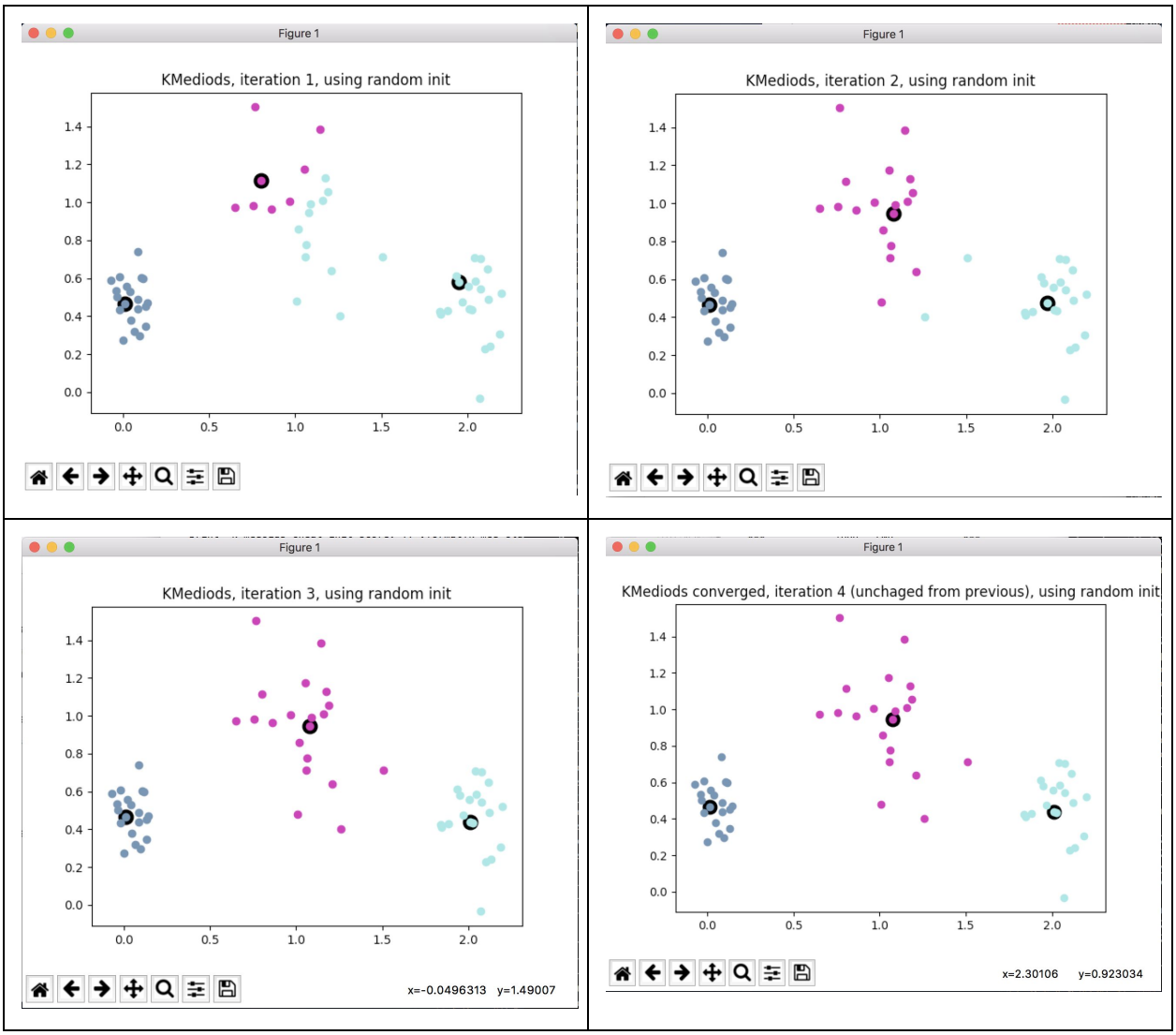
The minimum value of the cost function J when we optimize with k is 0. This is a bad idea, as this can only be achieved when k = n, which means that each data point in the training data is considered as a cluster. Specifically, this means that (mu) will be exactly each data point, (c) will map each data point to itself in (mu), and k will equal n.

Problem 2D



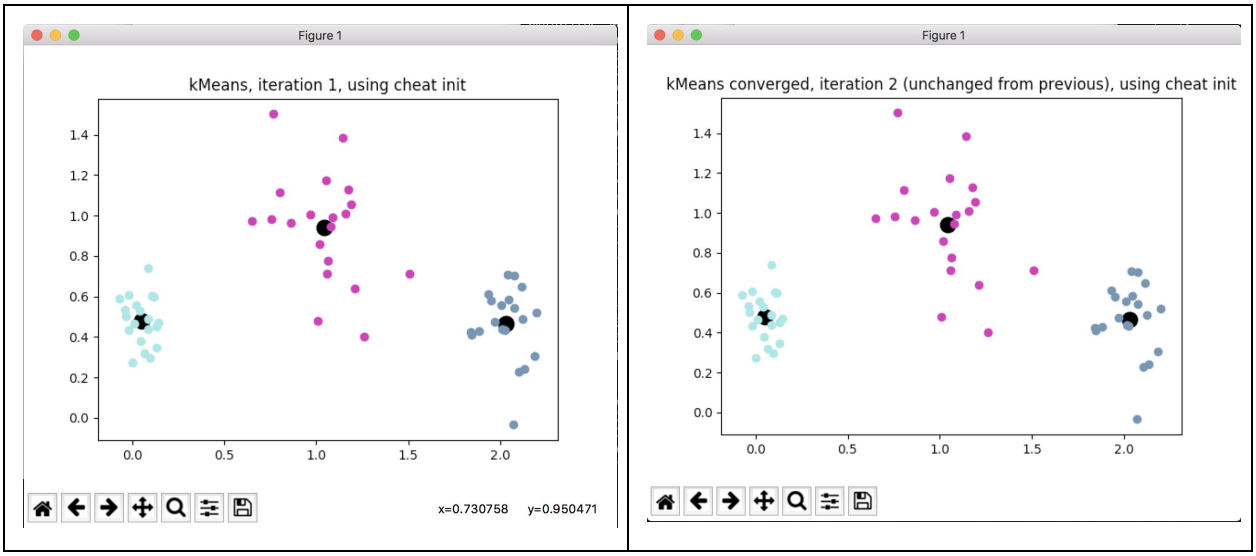
The table displays the iterations taken by the K-Means algorithm to converge. It technically only took 3 iterations to converge -- the 4th iteration was unchanged from the third thus satisfying the condition to terminate. Visually, we can see that the centroids moved closer and closer to the ‘center’ of the clusters present in the data from their randomly initialized starting positions.

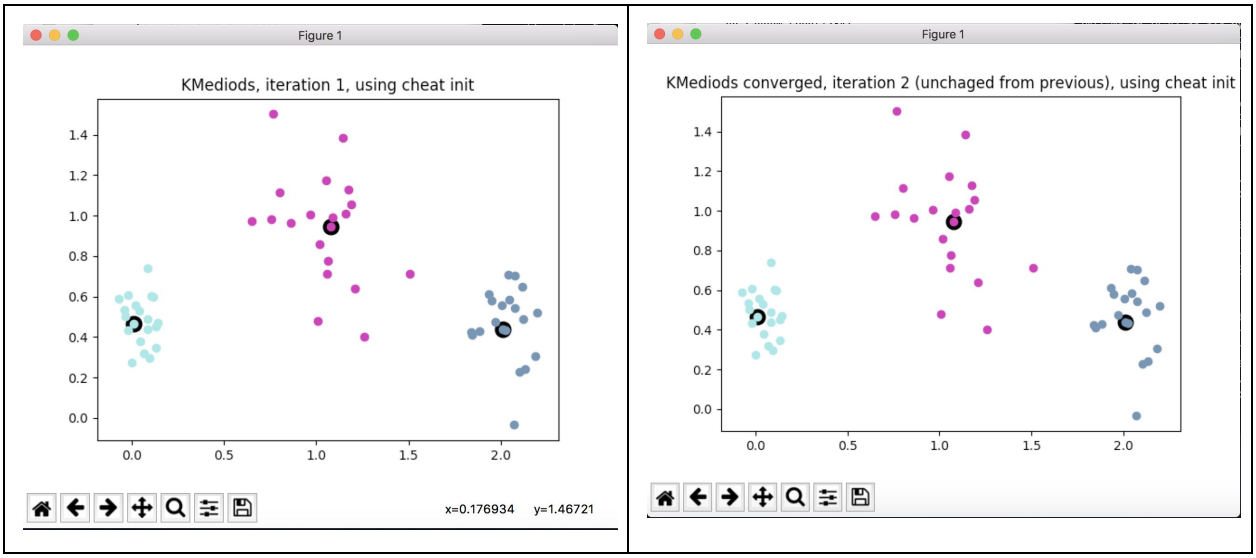
Problem 2E



This table displays the iterations taken by the K-Medoids algorithm to converge. Similar to the K-Means in 2D, it took 4 iterations where the 4th iteration satisfied the stopping condition. In these plots, we see that the algorithm repeated chooses a different data point as the center rather than a strict mean, starting from the random initialized points.

Problem 2F





Cheat init initializes the starting prototypes based on labels, which is why this converges much more quickly than the random initialization. In fact, it only took 2 iterations, and the second one was just to satisfy the stopping condition.

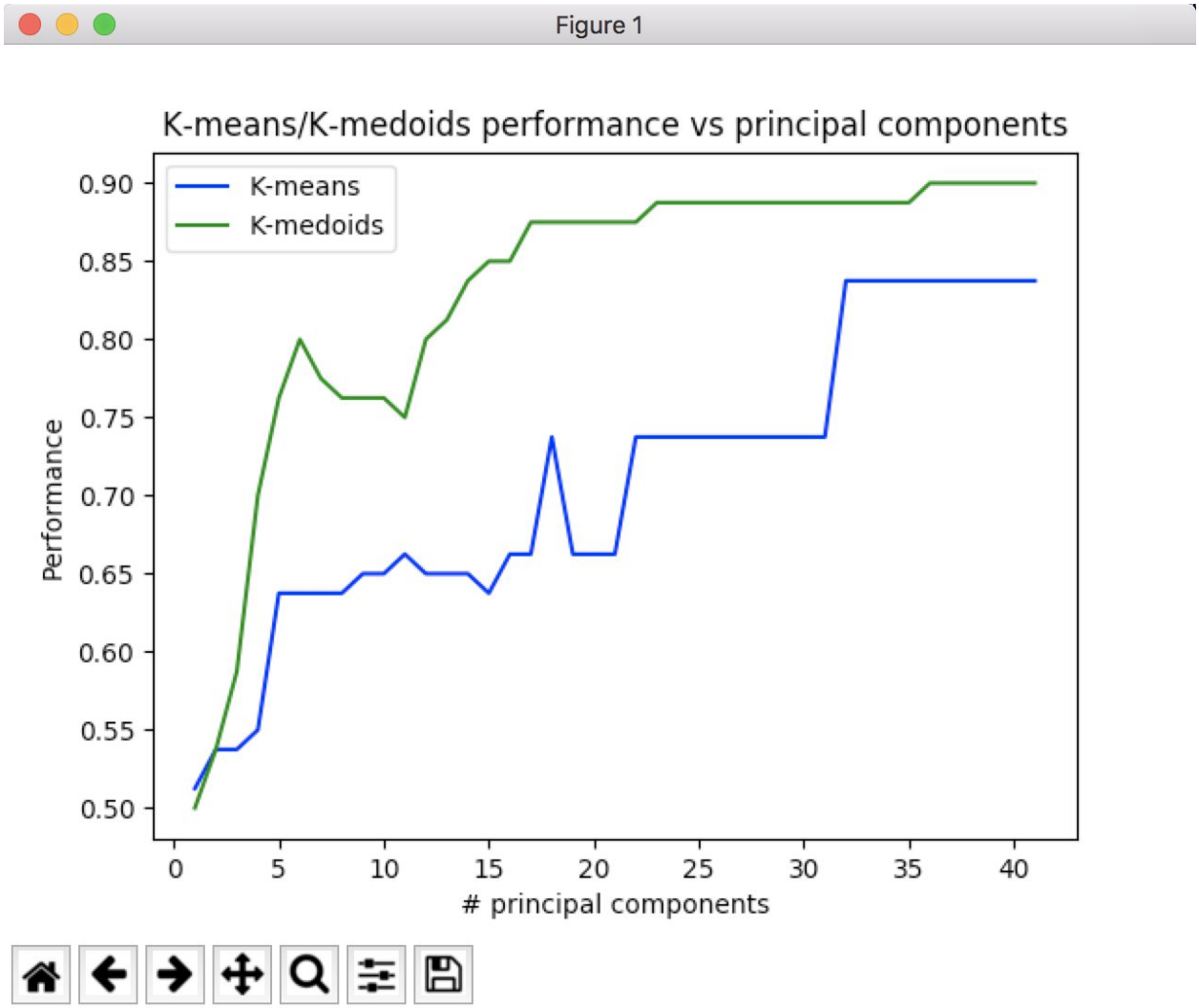
Problem 3A

	Average	Min	Max	Average Time
K-Means	0.616875	0.55	0.775	0.159228372574

K-Medoids	0.63125	0.575	0.725	0.221807599068

From these statistics, we see that although K-means terminates more quickly, K-medoids actually performs better both on average and on the minimum performance metric. This suggests that K-means may be a quicker algorithm, but at the cost of performance. K-means does perform better on the maximum performance metric, but the other two performance metrics are more important from a practical sense.

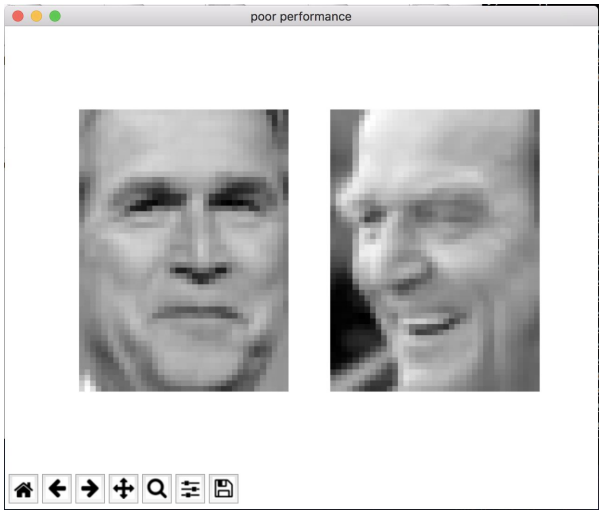
Problem 3B



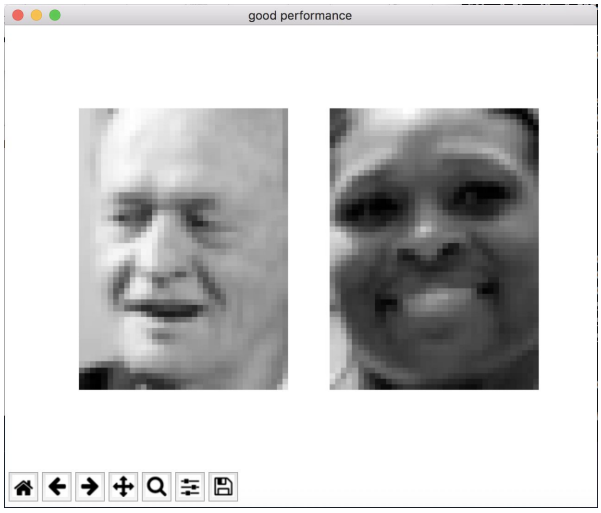
We see here again that K-medoids outperforms K-means pretty much all the time, leading us to think that having actual data points as centers allows us to cluster the data more effectively than means, which are somewhat more arbitrary. In addition, the performance for both clustering algorithms increase with

number of principal components, with the addition of the first couple principal components driving up performance significantly and dropping off later on at around 35 principal components. This means that for exceptionally low numbers of principal components, there simply isn't enough information (variance) to tell the faces apart, but also after a certain point, additional principal components do not contribute significant amounts of information to affect the clustering performance.

Problem 3C



4 and 5, score = 0.5125



9 and 16, score = 0.9875

The methodology I used was simply comparing every pair of faces available in the dataset using K-medoids, which we have seen to perform better than K-means, at least for this dataset.

It seems reasonable for K-medoids to do poorly differentiating 4 and 5. The two faces are similar in skin tone as well as facial features, such as wrinkles, eyes, nose shape, and mouth. At a glance, they also seem to me to be the same face. It also seems reasonable for K-medoids to do well differentiating 9 and 16. Not only are their skin tones different, there are further differences in facial features, especially in the nose, eyes, and eyebrows.