Milad Nourian
ECE M146
HW4

i	label	P_0	f_1 sign $(x_1 - 2)$	f_2 sign $(x_2 - \cdot)$	h_1
1	−	1/10	−		−
2	−	1/10	−		−
3	+	1/10	+		+
4	−	1/10	−		−
5	−	1/10	−		−
6	−	1/10	+		+
7	+	1/10	+		+
8	−	1/10	−		−
9	+	1/10	−		−
10	+	1/10	+		+

i	label	P_ϕ	f_1 sign$(x_1 - \cdot)$	f_2 sign $(y - 12)$	h_2
1	−	0.044		−	−
2	−	0.044		−	−
3	+	0.044		−	−
4	−	0.044		−	−
5	−	0.044		+	+
6	−	0.324		−	−
7	+	0.044		−	−
8	−	0.044		−	−
9	+	0.324		+	+
10	−	0.044		−	−

On iteration $t=1$:

$$\epsilon_{t=1} = \frac{1}{10} + \frac{1}{10} = \frac{2}{10}$$

$$\beta_{t=1} = \frac{1}{2}\log_2 \frac{1-\epsilon_{t=1}}{\epsilon_{t=1}} = 1$$

Find weights of D_1:

$$w_{t+1}(n) \propto w_t(n) e^{-\beta_t y_n h_t(x_n)}$$

→ $y_n h_t(x_n) = +1$ correctly classified

→ $y_n h_t(x_n) = -1$ incorrectly "

$$w_{t+1}(n) \propto w_t(n) e^{-1} = w_t(n)[0.3678] \text{ correct}$$

$$w_{t+1}(n) \propto w_t(n) e^{+1} = w_t(n)[2.718] \text{ incorrect}$$

Note: I used $w_{t+1}(n) \propto w_t(n) e^{-\beta_t y_n h_t(x)}$ and not $w_{t+1}(n) \propto w_t(n) 2^{-\beta_t y_n h_t(x)}$

Normalize the weights

$$\sum w_{t+1}(n) = \frac{\left(\frac{8}{10}\right)e^{-1}}{7} + \frac{\left(\frac{2}{10}\right)e}{7} \rightarrow z = 0.828$$

$$w_{t=1}(n) = \frac{1}{10} e^{-1}\left(\frac{1}{0.828}\right) = 0.044 \text{ (correctly)}$$

$$w_{t=1}(n) = \frac{1}{10}(e)\left(\frac{1}{0.828}\right) = 0.324 \text{ (incorrect)}$$

$\varepsilon_{t=2} = 4 \times 0.044 = 0.176$

$\beta_{t=2} = \frac{1}{2} \log\left[\frac{1-\varepsilon_t}{\varepsilon_t}\right] = 1.1135$

$h(x) = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(x)\right]$

$H_{\text{final}}(n) = h_{t=1}(1)^{\beta_{t=1}} + h_{t=2}(1.1135)^{\beta_{t=2}}$

P2)a) When $k=n$, the minimum value of objective is zero as every point is its own cluster so the error (objective) is zero.

P2)b)

$$\mu^* = \text{argmin}\ \lambda(\mu^T\mu) + \sum (x_j - \mu)^T(x_j - \mu)$$

$$= \lambda\mu^T\mu + \sum x_j^T x - x_j^T\mu - \mu^T x_j + \mu^T\mu$$

Take gradient: $\lambda[\mu] + 2\mu - 2x_j = 0$

$$\sum 2x_j = \mu[\lambda + 2|C|]$$

$$\mu^*_{\text{optimal}} = \mu = \frac{\sum 2x_j}{\lambda + 2|C_j|}$$

P2)3)

To solve the optimal problem, we need to solve the regularized k-mean problem.

define the problem as such:

let x_j be the position of j^{th} student.

let μ_i be the location of the TA.

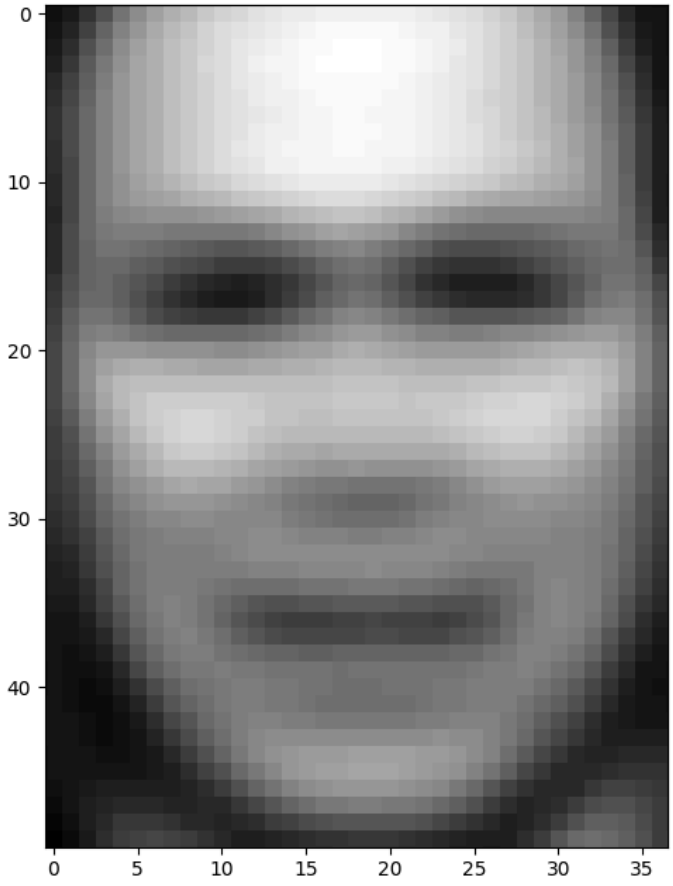
We want the TA to be close to student so students walk less, and also we want TA's to be close to desk (0,0), so TA's walk the shortest path.

So we want to find the optimal solution to:

$$\sum_{i=1}^{K} \left(\lambda \|\mu_i\|_2^2 + \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 \right)$$

and λ is the hyperparameter that we need to choose.

3.1)a)
For this problem, I first retrieved the images data set, and then used show_image to display some of the images. Then to obtain the average image, I averaged the columns and divided by the number of total images.
Below is the average photo by obtaining the mean of all of the images in the dataset.
Comment briefly on this "average" face.
As seen in the photo, the photo contains all the components of a face (eyes, mouth, nose, etc); however, since it has been averaged over all the data points, it does not have any particular shape and no specific person can is identified from the image.



3.1)b)
As provided in the homework specs, I used
plot_gallery([vec_to_image(U[:,i]) for i in xrange(12)])
to get the first 12 images from the covariance matrix U (we display the first 12 columns, or eigenfaces).

Image is shown below.
Comment briefly on your observations. Why do you think these are selected as the top eigenfaces?

Eigenfaces are obtained from covariance matrix. Each eigenface recognizes a pattern in the image (some recognize symmetry, facial hair, …). The dark and light areas of the image highlight the fact that each feature of a face is singled out to be evaluated and scored. Basically, using eigenfaces help to reduce the dimension and storage needed to store the images as we can for example, store an image as average face, plus 10% of eigenface 1, and so on.
The reason that these images are the top eigenfaces can be that they highlight the important features of a face that are important for face recognition. They can also be selected as the top eigenfaces because they have a considerable amount of data

3.1)c)
The reconstructed images for all values of l = 1, 10, 50, 100, 500, 1288 are shown below.
Comment briefly on the effectiveness of differing values of l with respect to facial recognition.
Reducing the dimension of images (using smaller l values), causes the images to be more unclear and blurry. This is because the feature set (which in this case are the pixels of the images) are not as rich and therefore, the images with high dimension reduction (very small l) are seen as very unclear (the images just represent the basic features of the face and recognizing the individuals in the photos are impossible). In other words, for small l values, the number of eigenvectors used to represent the image are few, therefore the image produced is far from the original image. As the l increase, the feature space become richer and now more pixels are used to reconstruct the image, which results in a clearer result. Reducing l is sort of a tradeoff, it takes less space to store the images, but the images are blurry and do not capture some of important features.

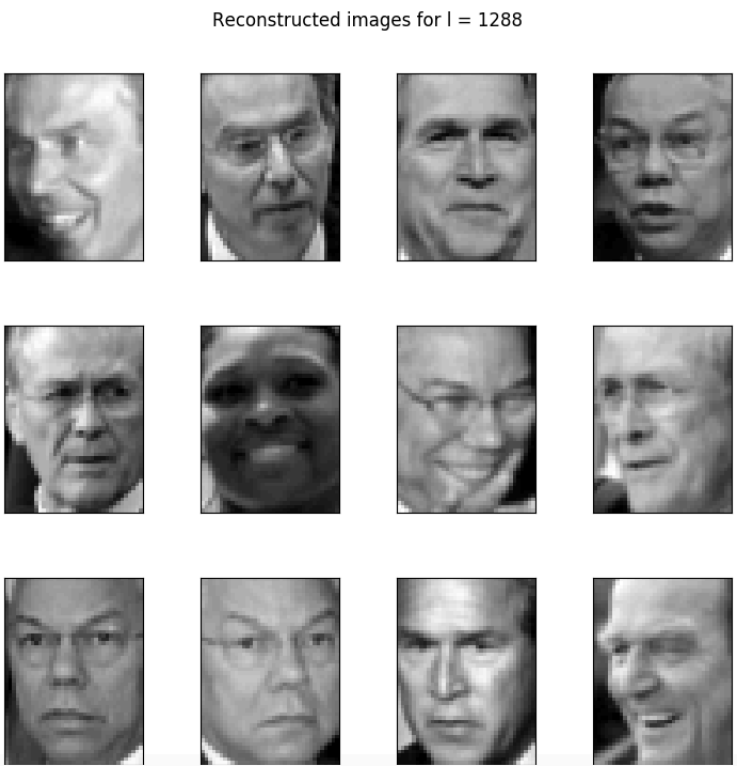
Reconstructed images for l = 1

Reconstructed images for l = 10

Reconstructed images for l = 50

Reconstructed images for l = 100

Reconstructed images for l = 500



---

3)2)a

We already know that the objective function for the kMeans algorithm is $J(c, \mu)$ = $\sum_{i=1}^{n} ||x^{(i)} - \mu_{C^{(i)}}||^2$

Now, we consider the case where were value of k can be changed, so our new objective function is J(c, μ, k). The problem is that of is J(c, μ, k) is always min J(c, μ, k) = 0. This happens when $\mu_{c^{(i)}} = x^{(i)}$. In other words, $\mathrm{J}(\mathrm{c}, \mu, \mathrm{k}) = 0$ when every datapoint $x^{(i)}$ is its own cluster (its mean is the same as the point itself), and this is bad, because cost function J(c, μ, k) does not provide much helpful information.

---

3)2)b)

TODO parts of the source code are implemented.

---

3)2)c)

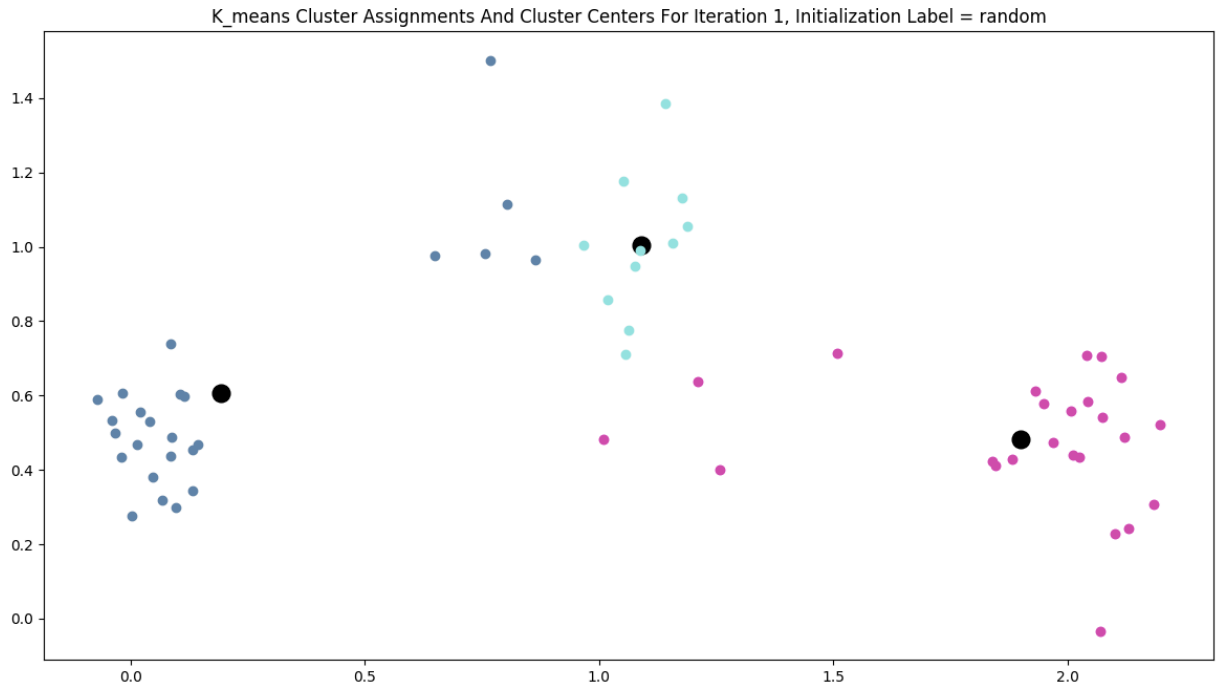
Implemented

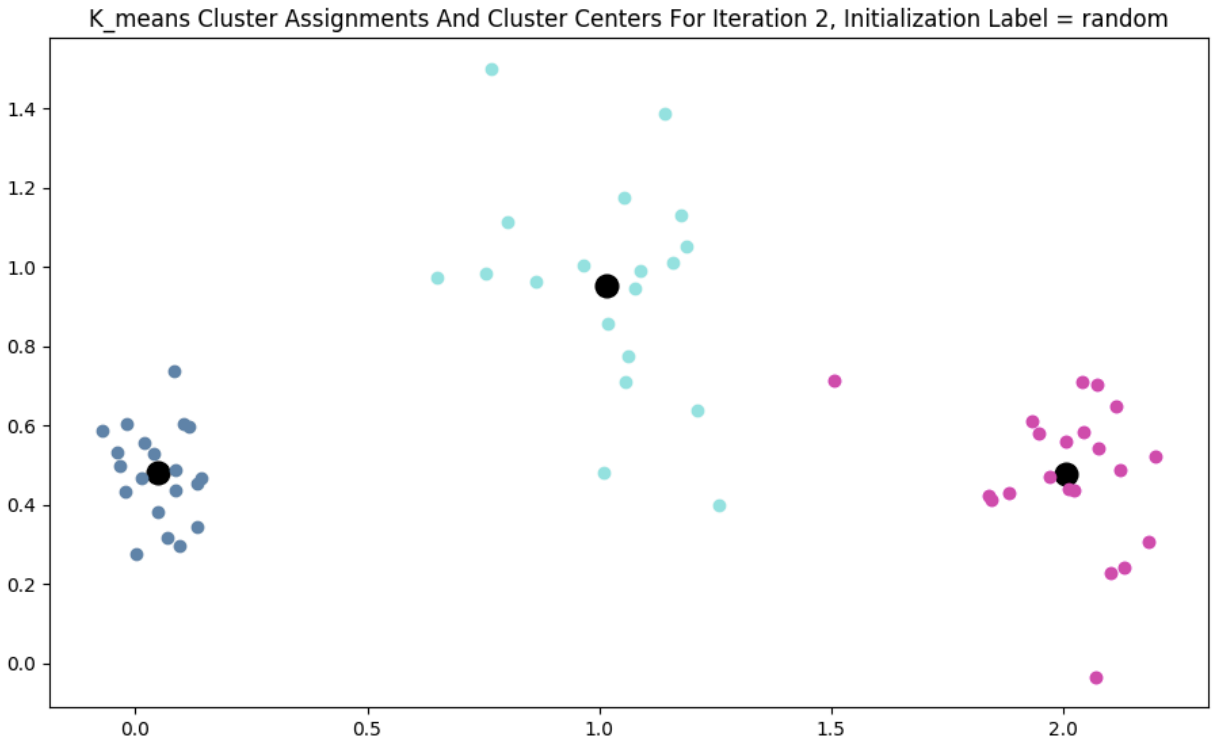
---

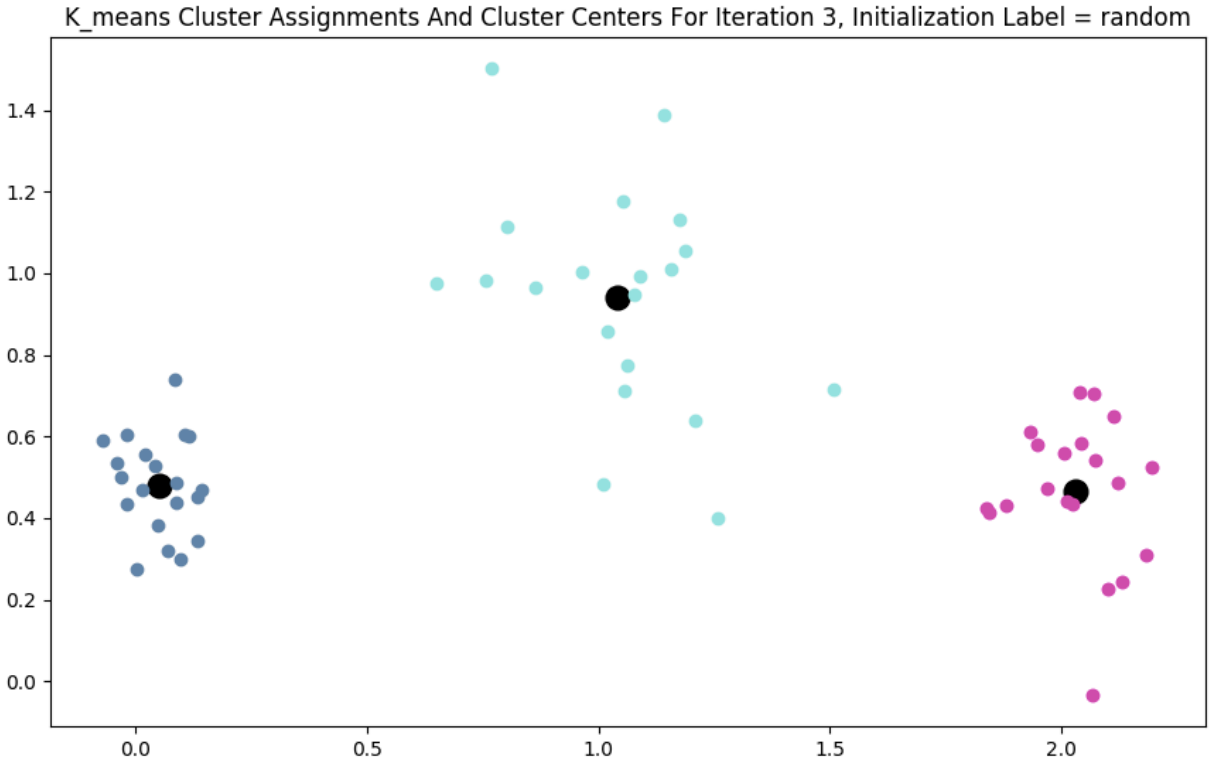
3)2)d)

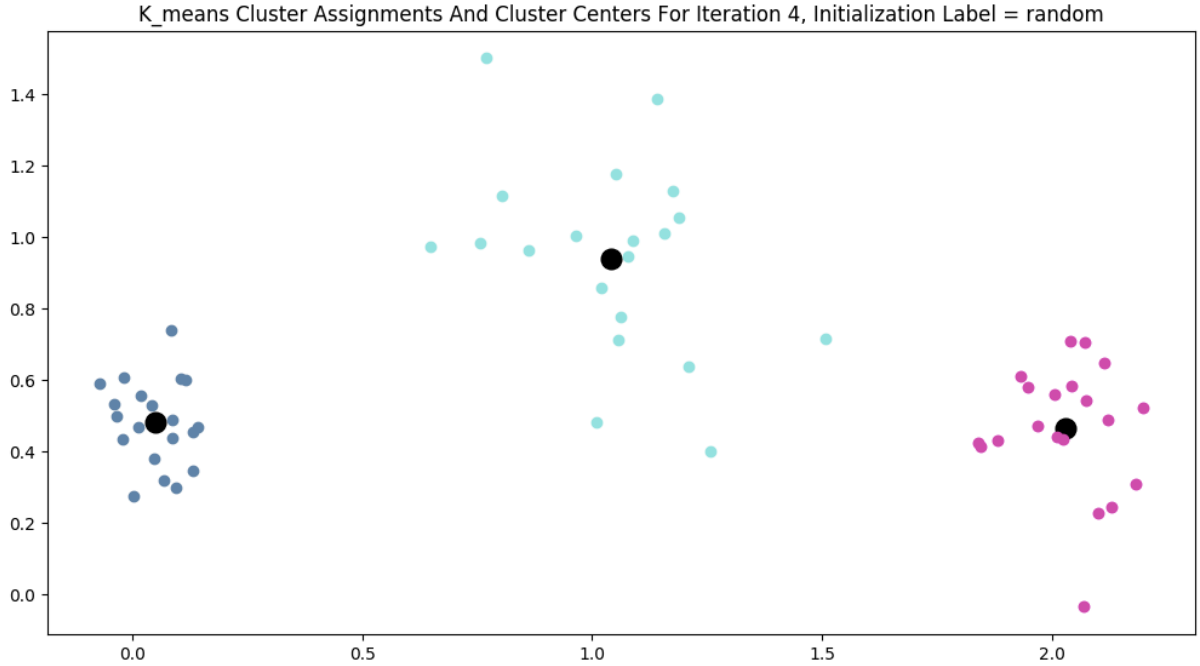
Generate_points_2d () was used to generate 3 clusters.

The plots for k-means and cluster centers are shown below using random initialization. From the plots, we see that the centroids are selected in random initially. Then, the center of kmean is recalculated, and after 4 iterations, the kMean algorithm converges and provides 3 clusters as shown below.


K_means Cluster Assignments And Cluster Centers For Iteration 1, Initialization Label = random
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

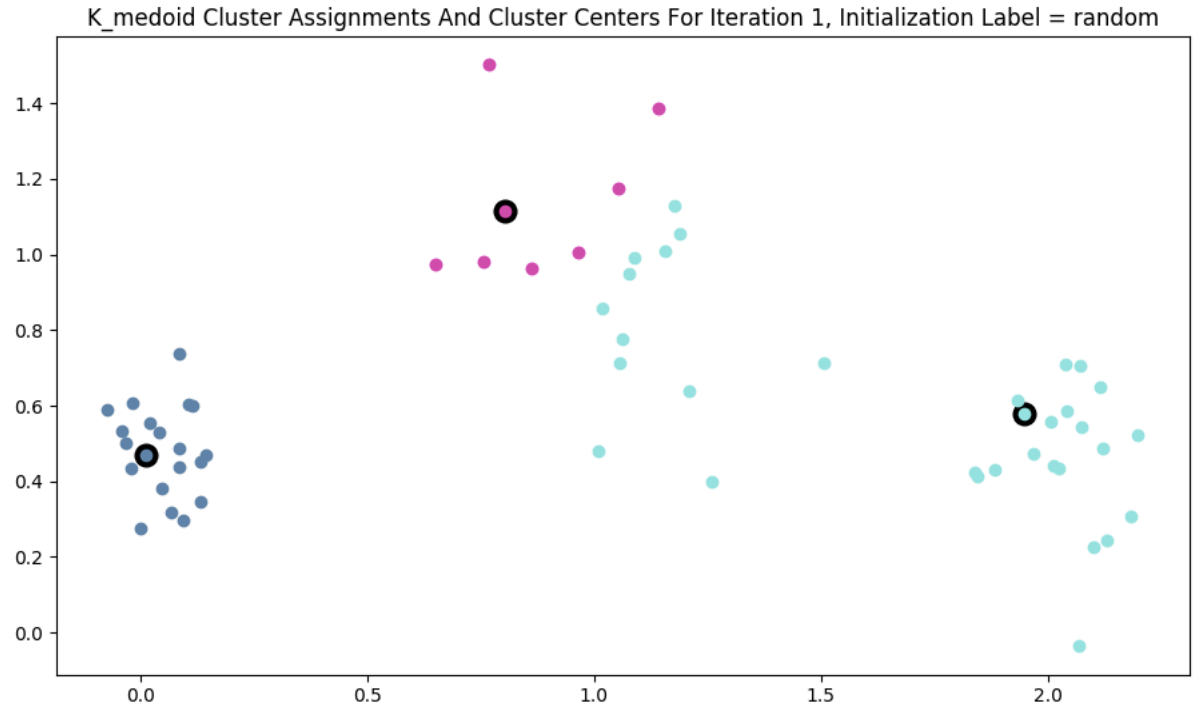

K_means Cluster Assignments And Cluster Centers For Iteration 2, Initialization Label = random
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

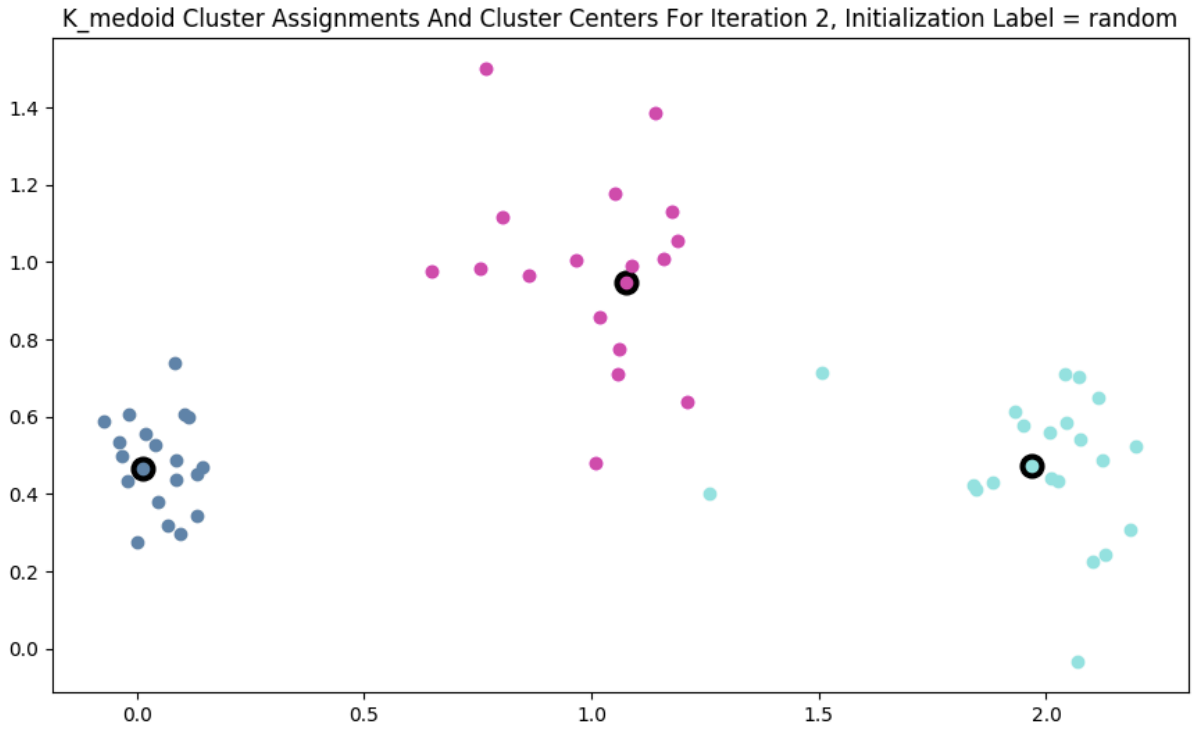

K_means Cluster Assignments And Cluster Centers For Iteration 3, Initialization Label = random
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

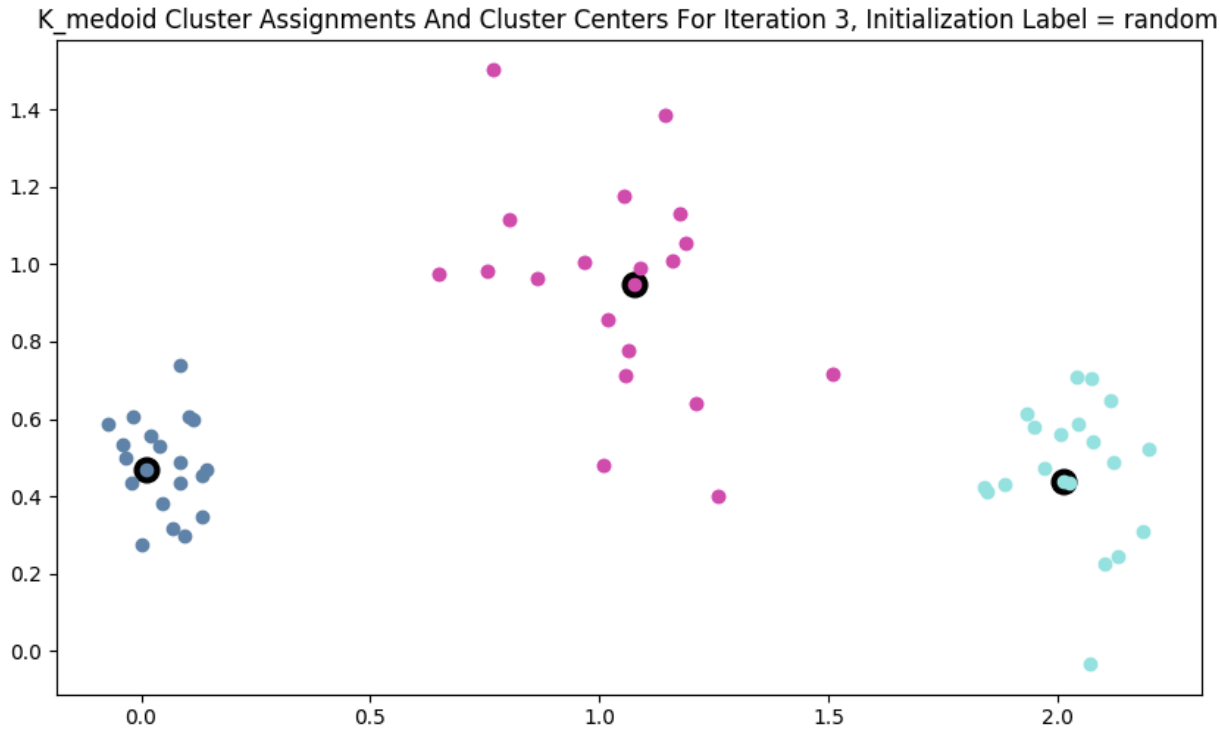

K_means Cluster Assignments And Cluster Centers For Iteration 4, Initialization Label = random
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0

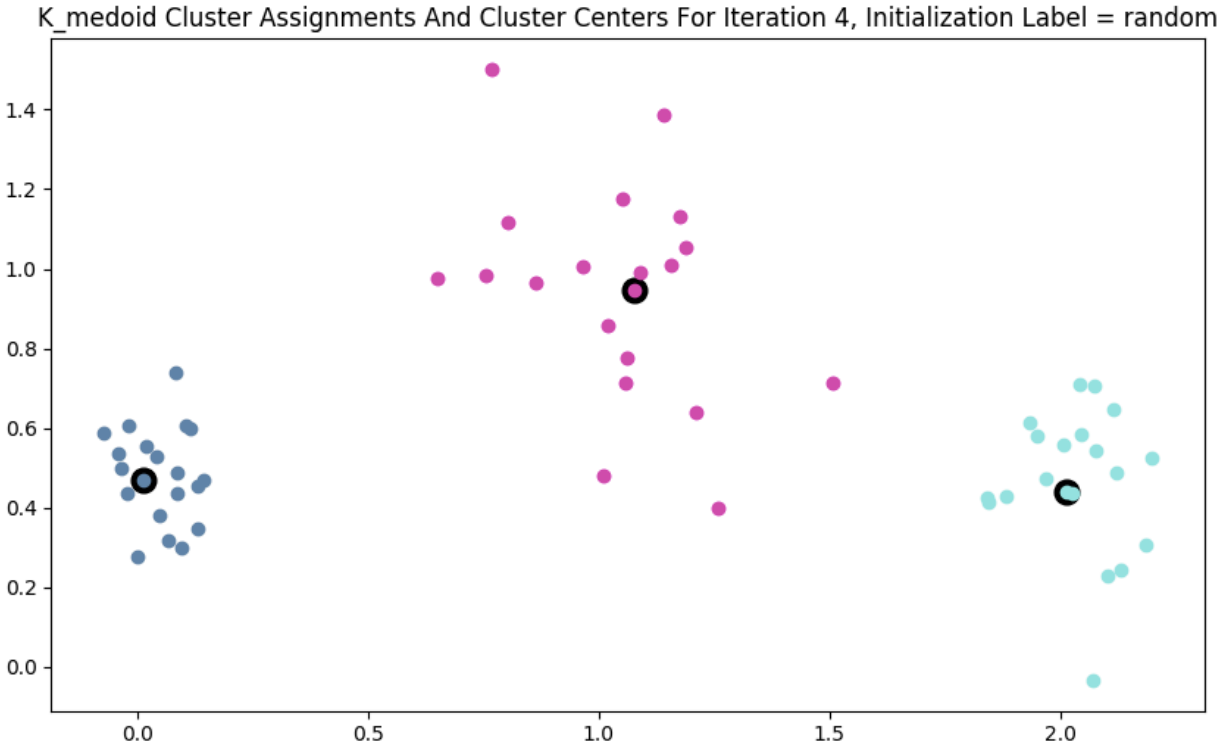
3)2)e)

The plots for the kMedoids is shown below, where the algorithm eventually converges with 3 clusters after 4 iterations. The algorithm used for kMedoids is very similar to kMeans, with the difference that at each iteration instead of finding the mean point of the clusters (kMeans), we find one of the points in the clusters that is closest to all other points in the cluster.



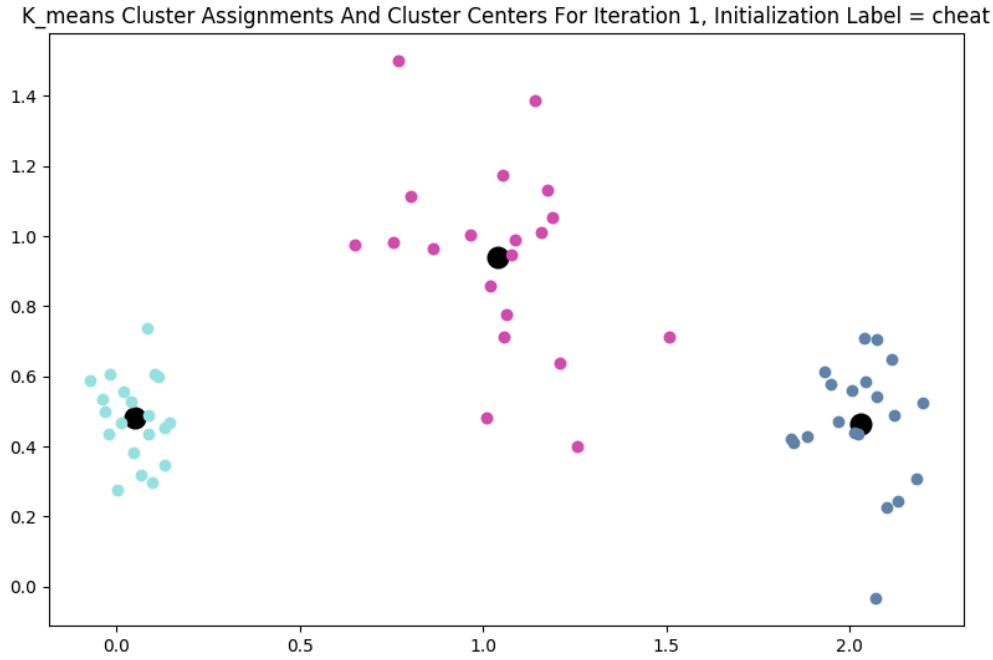


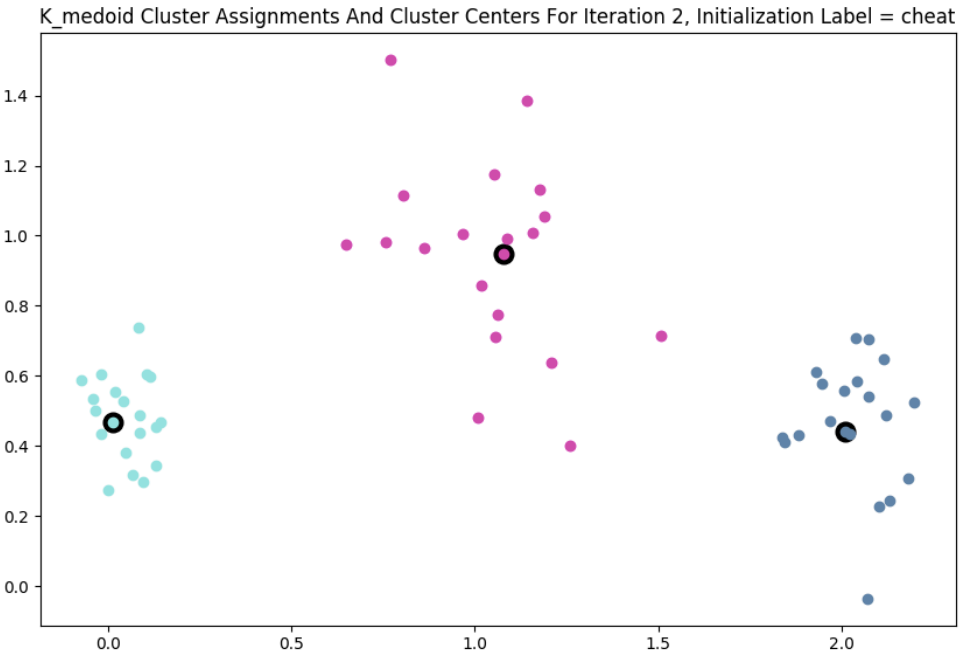

K_medoid Cluster Assignments And Cluster Centers For Iteration 3, Initialization Label = random
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0


K_medoid Cluster Assignments And Cluster Centers For Iteration 4, Initialization Label = random
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)2)f)

Now, we look at the effect of initialization on both of kMean and kMedoid algorithm. The plots are shown below. Both kMean and kMedoid converge in 1 iteration. We see that when we start with a centroid or medoid at a “good” location by “cheating” (close to the point where the algorithm will converge at), the algorithm converges faster and it will produce more accurate results. This emphasizes the importance of initialization in the kMean and kMedoid algorithms.





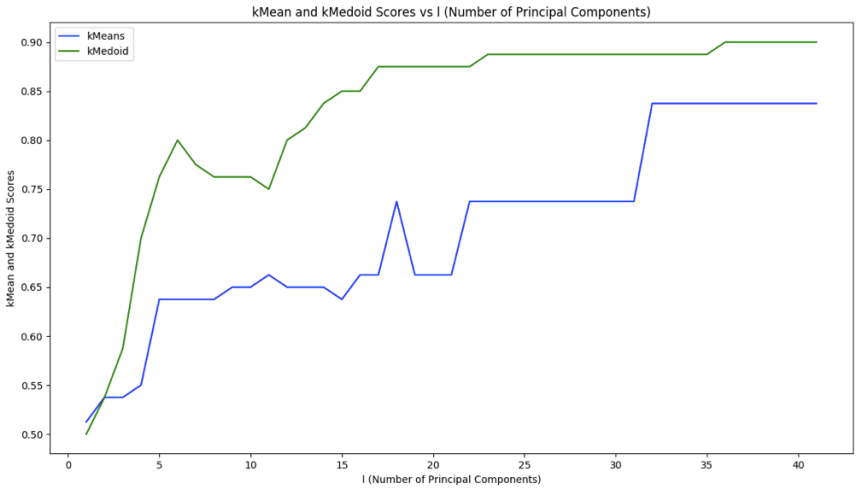
3)3)a)

	AVERAGE	MIN	MAX
K-MEANS	0.61749	0.55	0.775
K-MEDOIDS	0.6325	0.575	0.725

How do the clustering methods compare in terms of clustering performance and runtime?
We can see from the above table that on average, k-Medoid performs slightly better than k-Means. But the maximum score of the k-Mean algorithm is found to be larger than score of k-Medoid algorithm. The run time of k-Medoid is slower, as to find the medoid (mean point), we need to go over all the points of the cluster and also find the distance to all other points of the cluster $O(N^2)$, therefore, kMeans is more computationally effieicent.

---

3)3)b)
Now, we will explore how the clustering algorithm, when PCA method is applied to lower the dimension of the data set by using the principal components (top eigenvectors).
I started by finding the covariance matrix.
Then performed the projection onto the l first eigenvectors.
Then, reconstructed the signal and looped over different values of l to compare the performance. The plots are shown below.
Discuss the results in a few sentences.
We see from the below graph that the performance of the clustering algorithms (kMean and kMedoid) improves as the value of **l** increases. We can see this since the plot has an upward direction. The reason this happens is that when l is small, we used only a few of eigenvectors (eigenfaces) to represent the image, and this causes a distortion in the image as the new image does not captures the important features of the original image (since l is now the dimension of the feature set). As the value of **l** increases, we use more number of eigenvectors to reconstruct the image and this leads to a better reconstruction of the image signal and therefore, we achieve higher purity score of clusters for larger l values.



3)3)c)
For performing the task of finding a pair of individuals that discriminate well with clustering and a pair that don't do well on clustering, I used a nested for loop to go over every data point and find which pair provides the maximum score and which 2 pair provides the minimum score. Result is shown below. The result can be interpreted in the following way:
We would expect the images that are very different (have very different and far coordinates) to be clustered better than the images that are very similar and have close coordinates. For instance if one of the 2 clusters centroid is very far from the other centroid, clustering would be more accurate. In the following images, the best clustering is found in the pair that individuals have strong differences and their coordinates are far from each other (e.g. they don't overlap). But when the images are very similar and the persons in the images have similar position and face features, then clustering would be done poorly and the score would be low.