ISYE 6740 Fall 2020
Homework 1

Farshad Rafiei

3 Political blogs dataset [35 points]

We will study a political blogs dataset first compiled for the paper Lada A. Adamic and Natalie Glance, "The political blogosphere and the 2004 US Election", in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem (2005). The dataset `nodes.txt` contains a graph with $n = 1490$ vertices ("nodes") corresponding to political blogs. Each vertex has a 0-1 label (in the 3rd column) corresponding to the political orientation of that blog. We will consider this as the true label and try to reconstruct the true label from the graph using the spectral clustering on the graph. The dataset `edges.txt` contains edges between the vertices. You may remove isolated nodes (nodes that are not connected any other nodes).

1. (5 points) Assume the number of clusters in the graph is k. Explain the meaning of k here intuitively.
   Answer: In spectral clustering, the goal is to find k clusters such that some metrics relative to the centroid of the clusters is minimized for connected data points. Intuitively, for this dataset, that means to find k clusters of data points (blogs) in which the blogs have very similar political thoughts and orientation. e.g. if $k = 2$, then the goal is to find two clusters containing blogs which has either conservative (cluster 1) or liberal (cluster 2) view to the political problems.

2. (10 points) Use spectral clustering to find the $k = 2$, $k = 3$, and $k = 4$ clusters in the network of political blogs (each node is a blog, and their edges are defined in the file `edges.txt`). **We will treat the network as an undirectly graph; thus, when constructing the adjacency matrix, make sure to it is symmetrical.** Then report the majority labels in each cluster, when $k = 2, 3, 4$, respectively. For example, if there are $k = 2$ clusters, and their labels are $\{0, 1, 1, 1\}$ and $\{0, 0, 1\}$ then the majority label for the first cluster is 1 and for the second cluster is 0. **It is required you implementing the algorithms yourself rather than calling from a package**.

   Answer: To implement the algorithm, we follow the instructions provided for spectral clustering implementation in class lectures. The algorithms contains the following steps:

   - **Step 1:** Create adjacency matrix A and its corresponding degree matrix D
   - **Step 2:** Compute graph Laplacian using $L = D - A$
   - **Step 3:** Compute k smallest eignevectors of L Corresponding to the k smallest eigenvalues
   - **Step 4:** Run k-means algorithm on the data points projected on new eigenvectors

   Results indicate that there are two eignevalues which are very close to zero. This means that there are two connected clusters in the dataset (which makes sense because they ideally correspond to conservative and liberal parties).

   In addition, we run the algorithm for $k = 2$, $k = 3$ and $k = 4$ clusters to find the majority labels. For $k = 2$ clusters, majority label for first cluster is 1, and for second cluster is 0. In case of $k = 3$ clusters, two of the clusters has majority label of 1. The other cluster has majority label of 0. Finally, in case of using $k = 4$ clusters, majority label of 1 assigned to three clusters in front of only cluster which got majority label of 0.

3. (5 points) Now compare the majority label with the individual labels in each cluster, and report the *mismatch rate* for each cluster, when $k = 2, 3, 4$. For instance, in the example above, the mismatch rate for the first cluster is 1/4 (only the first node differs from the majority) and the the second cluster is 1/3.
   Answer: To find the overall mismatch rate, we compute the mismatch rate for each cluster separately and report maximum mismatch rate across all clusters. Mismatch rate in case of using $k = 2$, $k = 3$ and $k = 4$ clusters is equal to 0.4795, 0.4811, 0.4819, respectively.

4. (10 points) Now tune your k and find the number of clusters to achieve a reasonably small *mismatch rate*. Please explain how you tune k and what is the achieved mismatch rate.
   Answer: To find a reasonable number of clusters with minimum mismatch rate, we need to run the algorithm for different values of k. Similar to what we explained in part (3), to find the mismatch rate, we simply compute the maximum mismatch rate across all clusters. Figure (1) shows that mismatch rate is minimum for $k = 149$.

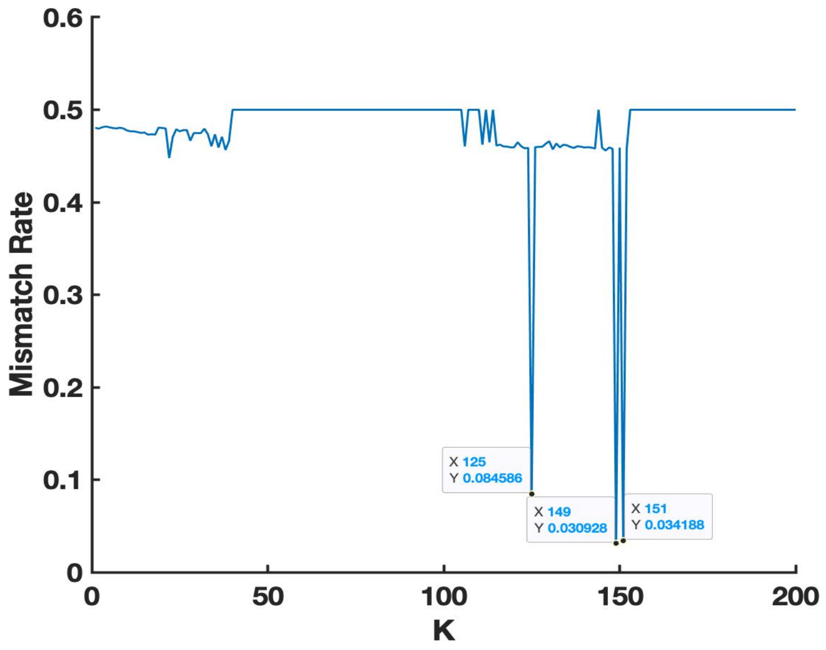


Figure 1: Mismatch rate as a function of number of clusters

5. (5 points) Please explain the finding and what can you learn from this data analysis (e.g., node within same community tend to share the same political view, or now? Did you find blogs that share the same political view more tightly connected than otherwise?)
   Answer: One important point is that outliers can affect the final results drastically. In this problem we know that the there are two clusters in the dataset. This was also drawn by finding two eigenvalues close to zero. However, k-means algorithm was not able to reliably assign connected data points to a single cluster. This happened because the algorithm assigned outliers to a single cluster and all other data points to another cluster. Hence, it includes huge amount of blogs corresponding to both conservative and liberal party. This is not ideal when $k = 2$ and tends to show a weak result.
   The mismatch rate was decreased when large number of clusters were used, but given that the number of clusters which makes the mismatch rate is significantly greater than the real number of clusters,

This sound not to be a reasonable clustering scheme. One approach to avoid this problem is to remove outliers on projected space and run k-means algorithm (or an algorithm which performs better on separating two clusters, such as SVM) without them. Performance of algorithm after removinf outliers is shown in Figure (2).

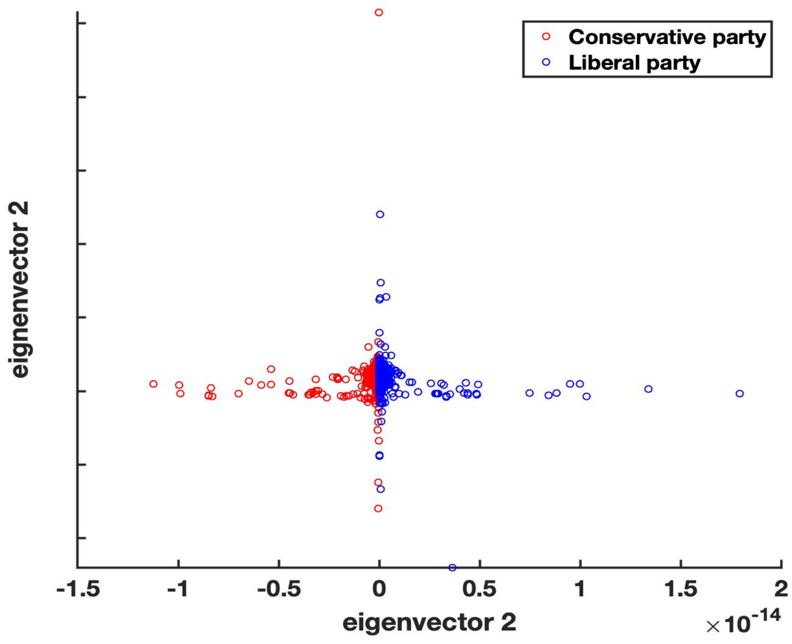


Figure 2: Performance of spectral clustering after removing outliers (typo: y-axis shows eignevector 1)

ISYE 6740, Fall 2020, Homework 2

100 points + 15 bonus points

Farshad Rafiei

1. PCA: Food consumption in European countries [50 points]

The data food-consumption.csv contains 16 countries in Europe and their consumption for 20 food items, such as tea, jam, coffee, yogurt, and others. We will perform principal component analysis to explore the data. In this question, please implement PCA by writing your own code (you can use any basic packages, such as numerical linear algebra, reading data, in your file).

First, we will perform PCA analysis on the data by treating each country's food consumption as their "feature" vectors. In other words, we will find weight vectors to combine 20 food-item consumptions for each country.

1. (10 points) For this problem of performing PCA on countries by treating each country's food consumption as their "feature" vectors, explain how the data matrix is set-up in this case (e.g., the columns and the rows of the matrix correspond to what).
Answer: Columns of food consumption matrix correspond to the foods and rows correspond to the countries. For this part of the question, each country (row) serve as a data point and each food (column) serve as a feature.

2. (10 points) Suppose we aim to find top k principal components. Write down the mathematical optimization problem for solving this problem (i.e., PCA optimization problem). Show why the first principal component is obtained by using a weight vector corresponding to the eigenvectors associated with the largest eigenvalue. Explain how to find the rest of the principal components.
Answer: Let's assume that we have m data points $\{x^1, x^2, ..., x^m\} \in \mathrm{R}^n$ and we want to perform PCA analysis. By definition, PCA projects the data to a directions $||w^i|| < 1$ in which the variability (variance) of the data is maximum. Let's suppose that the direction which shows maximum variability of data is w. Then our problem is to minimize the following variance function with respect to w:

$$\max_{w:||w||<1} \frac{1}{m} \sum_{i=0}^{m} \left(\, w^T x^i - w^T \mu \, \right)^2 \tag{1}$$

in which μ is the mean of the data points. Based on the lecture materials, we can rewrite equation (1) in the following format:

$$\max_{w:||w||<1} w^T(\frac{1}{m}\sum_{i=0}^{m}\big(\ x^i-\mu)(x^i-\mu)^T\big)\ w \tag{2}$$

Now, we can form Lagrangian function of the optimization problem as following:

$$L(w,\lambda) = w^T C w + \lambda(1-||w||^2) \tag{3}$$

$$where,\ C = \frac{1}{m}\sum_{i=0}^{m}(\ x^i-\mu)(x^i-\mu)^T \tag{4}$$

In equation (3), if w is the maximum of the original optimization problem, then there exists a λ, where (w,λ) is a stationary point of the Lagrangian function. By definition, in stationary point, the derivative of the function should be equal to zero. This implies that:

$$\frac{\partial L}{\partial w} = 0 \iff Cw = \lambda w \tag{5}$$

This indicates that the optimal solution w should be eigenvector of C and the objective function becomes:

$$\max_{w:||w||<1} w^T\lambda w = \max_{w:||w||<1} \lambda||w||^2 \tag{6}$$

In fact, to maximize the objective function in equation (6), we need to maximize λ. The problem becomes finding the largest eigenvalue associated with C.

To find rest of the principal components, we have to to find the eigenvectors associated with the largest eignevalues. That is, to find k pricipal components, we have to find top k eigenvectors associated with largest eignevalues. In this case, the maximum variance will be explained by these principal components, since they are associated with largest values of objective function and at the same time they are orthogonal which cause the projection not intervene with each other.

3. (10 points) Now assume $k = 2$, i.e., we will find the first two principal components for each data point. Find the weight vectors w_1 and w_2 to extract these two principal components. Plot these two weight vectors, respectively (e.g., in MATLAB, you can use `stem(w)` to plot the entries of a vector w; similar things can be done in Python). Explain if you find any interesting patterns in the weight vectors.
Answer: Weight vectors (eigenvectors) corresponding to two largest eigenvalues are shown in Figure (1). One pattern is that the weights in largest eigenvector are mostly positive, whereas in the other eignevector the weights are mostly negative. Also, these two vectors are orthogonal.

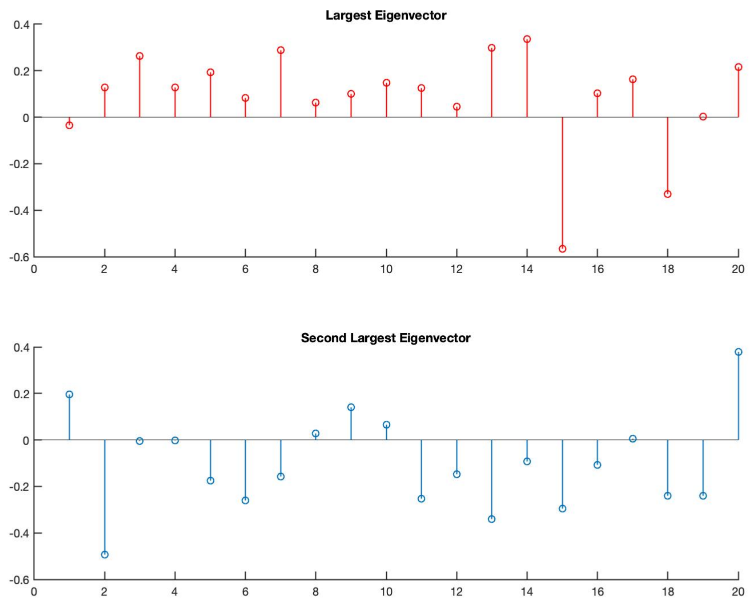


Figure 1: Eigenvectors associated with two largest eigenvalues when food consumption is used as feature vectors and countries as data points.

4. (10 points) Now extract the first two principal components for each data point (thus, this means we will represent each data point using a two-dimensional vector). Draw a scatter plot of two-dimensional representations of the countries using their two principal components. Mark the countries on the lot (you can do this by hand if you want). Please explain any pattern you observe in the scatter plot.

   Answer: The location of countries (data points) in the projected space is shown in figure (2). The figure shows the countries with similar food consumption habits close to each other. For example, Portugal, Italy and Spain consume a lot of garlic and olive. Another Interesting pattern is that the European countries which are close to each other, have similar food consumption as well. For example, Finland, Sweden, Norway and Denmark are roughly clustered together and use of crisp bread is the largest in these countries. One another interesting finding is the highest tea consumption in England. English tea is one the most famous tea in the world and figure (2) clearly validates high consumption of it in this country.

Now, we will perform PCA analysis on the data by treating country consumptions as "feature" vectors for each food item. In other words, we will now find weight vectors to combine country consumptions for each food item to perform PCA another way.

5. (10 points) Project data to obtain their two principle components (thus, again each data point – for each food item – can be represented using a two-dimensional vector). Draw a scatter plot of food items. Mark the food items on the plot (you can do this by

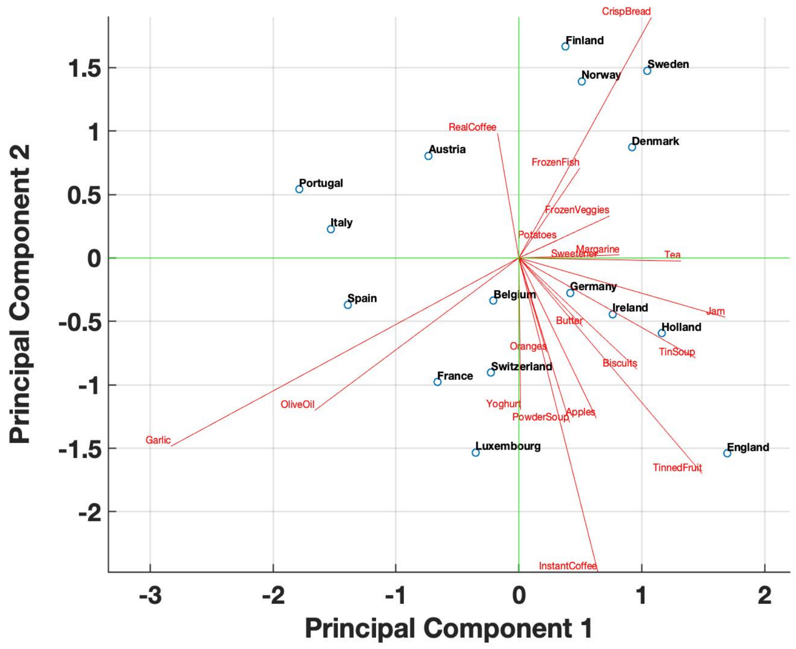


Figure 2: Location of data points (countries) in the new projected space. x-axis shows the principal component associated with largest eigenvalue and y-axis indicates the principal component corresponding to second largest eigenvalue. Red lines show the location of first two eigenvectors in the projected space. These red lines can give us an intuition of principal components and how the things change as move on these axes.

hand if you do not want). Please explain any pattern you observe in the scatter plot.
Answer: Figure (3) shows the scatter plot for a situation where we consider foods as data points (rows) and countries as features (columns). We observe the similar results. Countries like Portugal, Italy and Spain which showed to have similar food consumption in figure (2), have relatively well-aligned directions in reduced principal component coordination. Garlic and olive oil are on top of the y-axis and show high consumption rate for mentioned countries. One interesting pattern here, is that it is easier to find the consumption rate of a specific food for each country. For example, in Finland, Denmark and Norway consumption rate of tea is very high. However, the consumption rate of yoghurt is pretty much low in these countries.

2. Order of faces using ISOMAP [50 points]

This question aims to reproduce the ISOMAP algorithm results in the original paper for ISOMAP, J.B. Tenenbaum, V. de Silva, and J.C. Langford, Science 290 (2000) 2319-2323 that we have also seen in the lecture as an exercise (isn't this exciting to go through the process of generating results for a high-impact research paper!)

The file isomap.mat (or isomap.dat) contains 698 images, corresponding to different poses

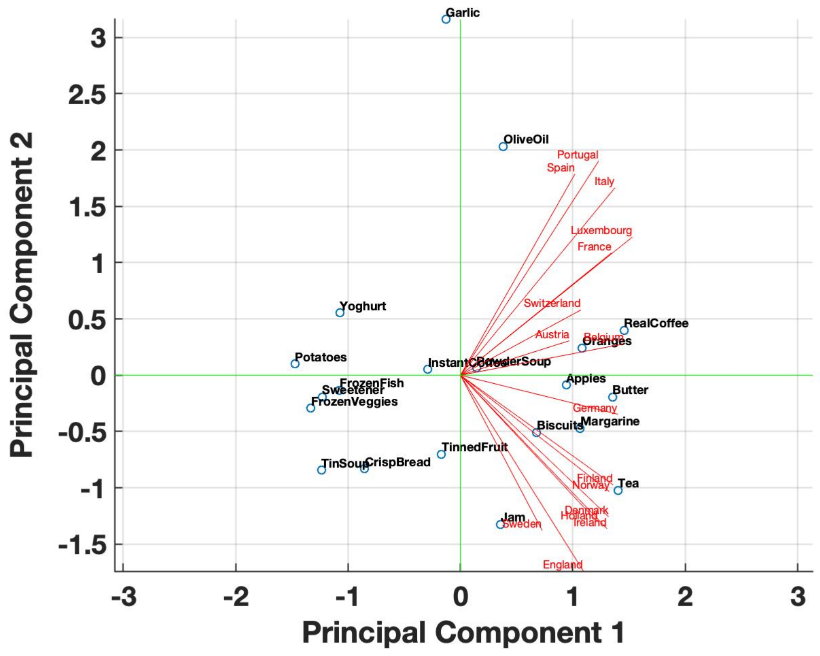
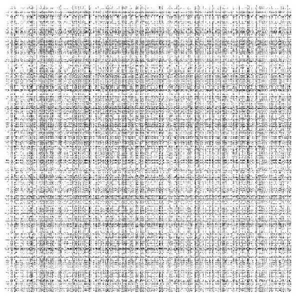


Figure 3: Location of data points (foods) in the first two principal component space. x-axis shows the principal component associated with largest eigenvalue and y-axis indicates the principal component corresponding to second largest eigenvalue. Red lines show the location of first two eigenvectors in the projected space. These red lines can give us an intuition of principal components and how the things change as move on these axes.

of the same face. Each image is given as a 64 $\times$ 64 luminosity map, hence represented as a vector in $\mathbb{R}^{4096}$. This vector is stored as a row in the file. [This is one of the datasets used in the original paper] In this question, you are expected to implement the ISOMAP algorithm by coding it up yourself. You may use the provided functions in ShortestPath.zip to find the shortest path as required by one step of the algorithm.

Choose the Euclidean distance (i.e., in this case, a distance in $\mathbb{R}^{4096}$) to construct the nearest neighbor graph—vertices corresponding to the images. Construct a similarity graph with vertices corresponding to the images, and tune the threshold ϵ so that each node has *at least* 100 neighbors (this approach corresponds to the so-called ϵ-Isomap).

(a) (10 points) Visualize the similarity graph (you can either show the adjacency matrix, or similar to the lecture slides, visualize the graph using graph visualization packages such as Gephi (`https://gephi.org`) and illustrate a few images corresponds to nodes at different parts of the graph, e.g., mark them by hand or use software packages).
Answer: For this question, I'm using two different schemes. In the first method, as the problem states, to acquire the similarity matrix, I use ϵ-Isomap and ensure that each node has at least 100 neighbors. In the second scheme, I us *knn* method and find exactly 100 nearest neighbors for each node. Similarity matrices for these two schemes



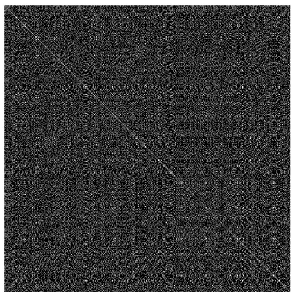


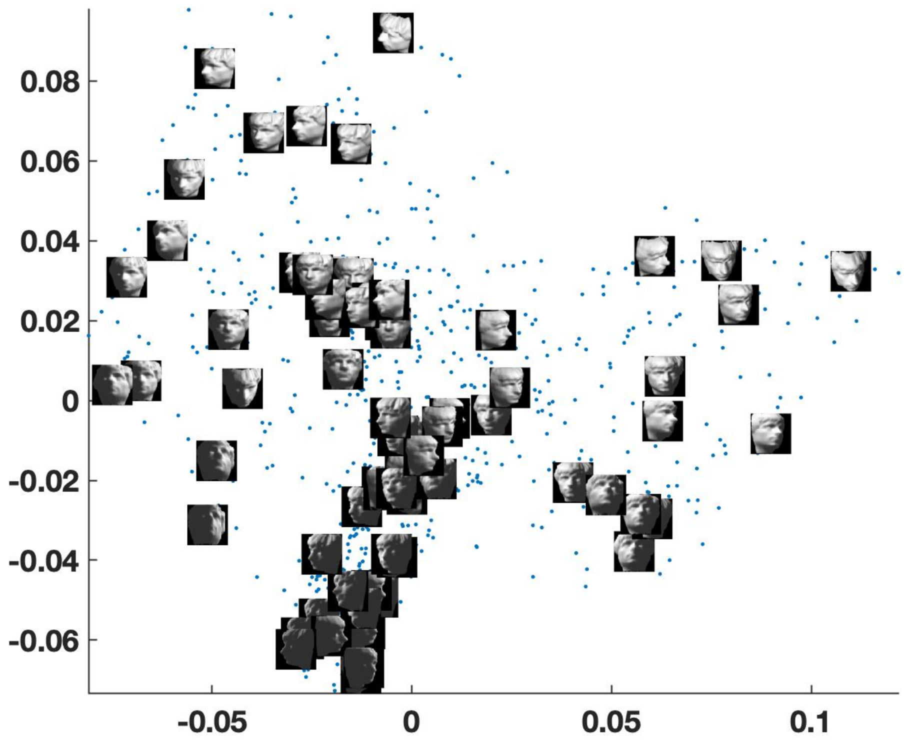
Figure 4: Similarity matrix. Left figure shows the similarity matrix obtained from ϵ-Isomap ($\epsilon = 25$) and right figure shows the similarity matrix obtained from using KNN ($k = 100$)

are shown in figure (4).

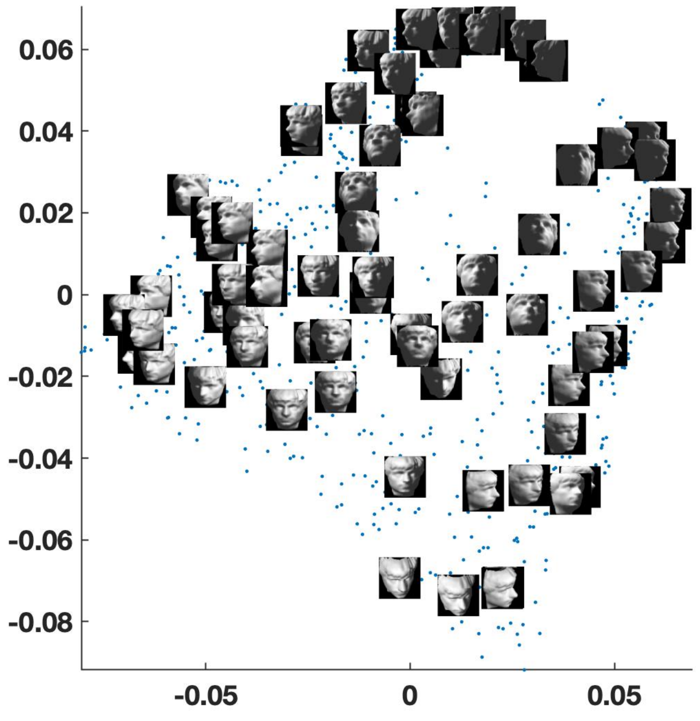
(b) (20 points) Implement the ISOMAP algorithm yourself to obtain a $k = 2$-dimensional embedding. This means, each picture is represented by a two-dimensional vector (Z in the lecture), which we called "embedding" of pictures. Plot the embeddings using a scatter plot, similar to the plots in lecture slides. Find a few images in the embedding space and show what these images look like. Comment on do you see any visual similarity among them and their arrangement, similar to what you seen in the paper?
Answer: Results are shown in figure 5. Two schemes were used to obtain the results. Using ϵ-Isomap method did not perform very well on this data. However, there still are some similarities in the clusters generated by the algorithm. On the other hand. the performance of KNN was very well and generated results which are similar to the original paper. Faces which pose to the left are clearly clustered together. The faces that pose toward right are all clustered together. In the middle, the faces are all posing forward with them looking up at top part of figure and looking down at bottom.

(c) (10 points) Now choose ℓ_1 distance (or Manhattan distance) between images (recall the definition from "Clustering" lecture)). Repeat the steps above. Use ϵ-ISOMAP to obtain a $k = 2$ dimensional embedding. Present a plot of this embedding. Do you see any difference by choosing a different similarity measure by comparing results in Part (b) and Part (c)?
Answer: The results of using ℓ_1 distance instead of Euclidean distance are shown in figure (6). Using this measure improved the performance of the algorithm. ϵ-ISOMAP method is now capable of producing better results and is able to cluster the similar poses of faces with higher accuracy. This highlights the importance of choosing an appropriate distance measure for a specific dataset. Note that using Manhattan distance did not change the performance of KNN-based Isomap algorithm significantly for this dataset.

(d) (10 points) Perform PCA (you can now use your implementation written in Question 1) on the images and project them into the top 2 principal components. Again show

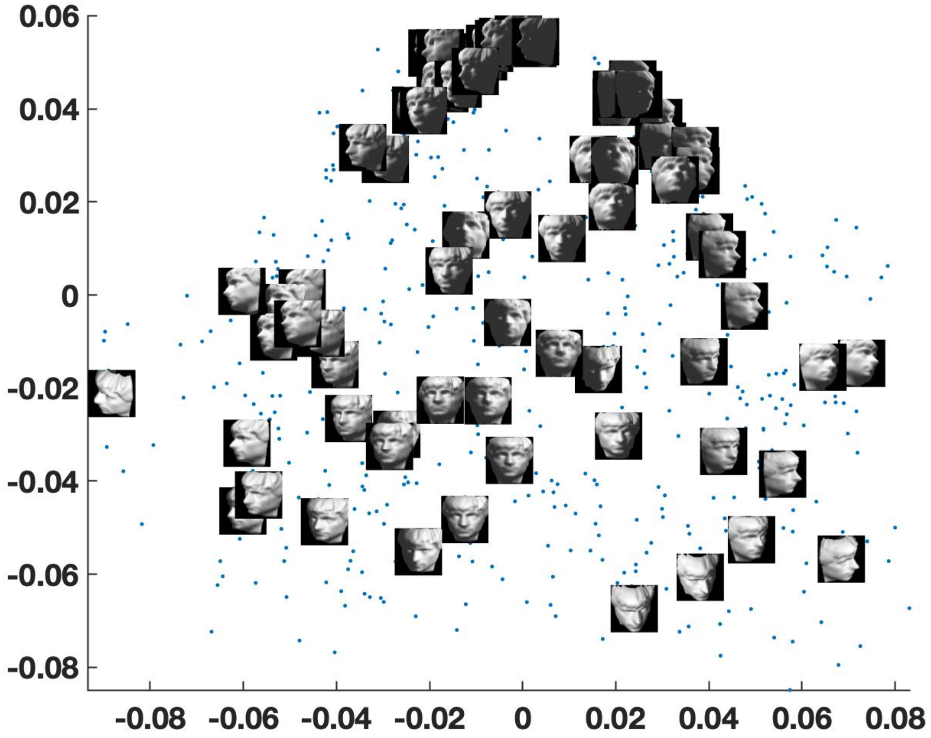


ϵ-Isomap

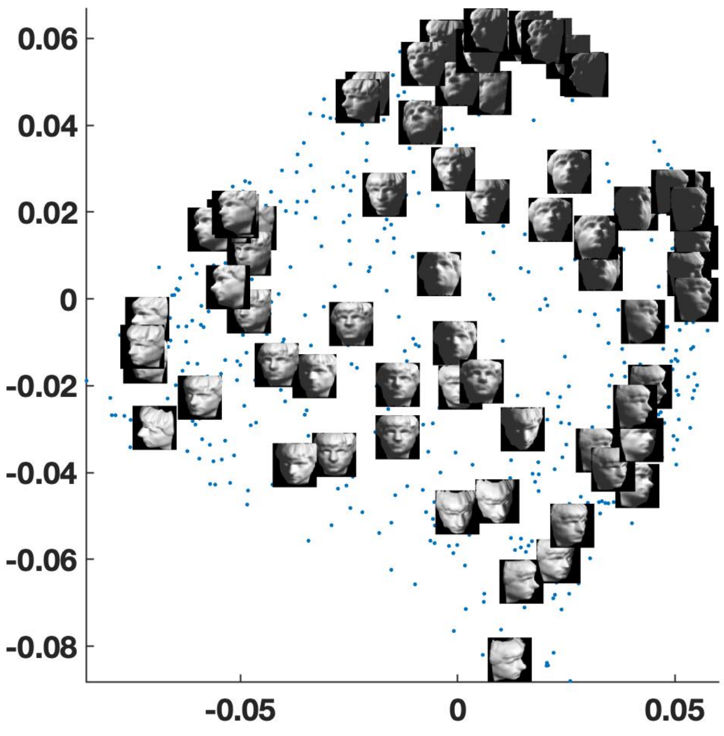


KNN

Figure 5: Embedding scatter plot using Euclidean distance. Left figure shows the embeddings obtained from ϵ-Isomap ($\epsilon = 25$) and right figure shows the embeddings obtained from using KNN ($k = 100$).



ϵ-Isomap



KNN

Figure 6: Embedding scatter plot using Manhattan distance. Left figure shows the embeddings obtained from ϵ-Isomap ($\epsilon = 25$) and right figure shows the embeddings obtained from using KNN ($k = 100$).

them on a scatter plot. Explain whether you see a more meaningful projection using ISOMAP than PCA.

Answer: PCA algorithm which was implemented in previous question was used to perform PCA analysis on this dataset. The results are shown in figure (7). PCA is able to cluster images with similar face poses but Isomap algorithm outperforms PCA. Specifically, left side of figure (7) shows lots of discrepancy in PCA analysis. However, Isomap algorithm was able to capture the variability very well.

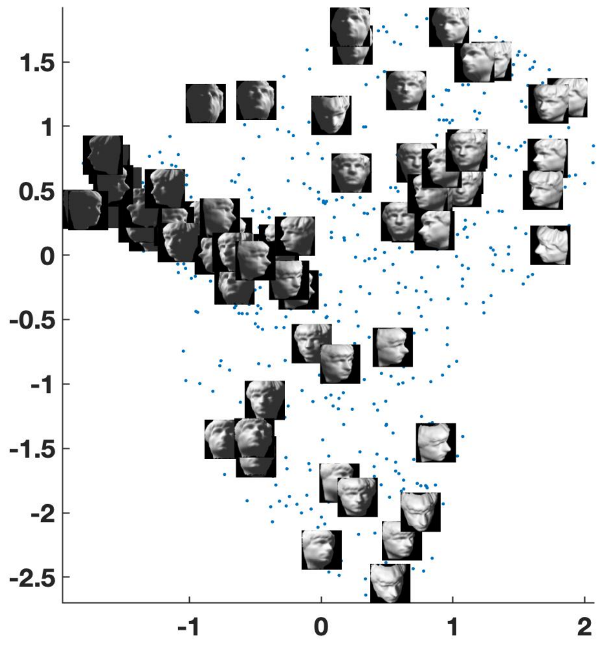


Figure 7: Embedding scatter plot using principal component analysis (PCA).

3. (Bonus) Eigenfaces and simple face recognition [15 points]

This question is a simplified illustration of using PCA for face recognition. We will use a subset of data from the famous Yale Face dataset. **Remark:** You will have to perform downsampling of the image by a factor of 4 to turn them into a lower resolution image as a preprocessing (e.g., reduce a picture of size 16-by-16 to 4-by-4). In this question, you can implement your own code or call packages.

First, given a set of images for each person, we generate the eigenface using these images. You will treat one picture from the same person as one data point for that person. Note that you will first vectorize each image, which was originally a matrix. Thus, the data matrix (for each person) is a matrix; each row is a vectorized picture. You will find weight vectors to combine the pictures to extract different "eigenfaces" that correspond to that person's pictures' first few principal components.

1. (10 points) Perform analysis on the Yale face dataset for Subject 1 and Subject 2, respectively, using all the images EXCEPT for the two pictures named subject01-test.gif and subject02-test.gif. **Plot the first 6 eigenfaces for each subject.** When visualizing, please reshape the eigenvectors into proper images. Please explain can you see any patterns in the top 6 eigenfaces?
   Answer: There are 21 images in this dataset in which subject 1 own 11 of them. After excluding the test images, we had total of 19 images. The size of the actual images are 243×320, which shrinks to 61×80 after resizing them by scale factor of 0.25. I used the PCA code which was written in problem (1). The visualization of weights (eigenvectors) for top 6 principal components are shown in figure (8).
   Weight vectors (eigenvectors) act as a edge detector. figure (8) show that each weight matrix tend to show a specific edge detector for each subject. One important point is that the extracted faces seems to be robust to the facial expressions (or even glasses), which can make the face recognition algorithm more efficient. However, the patterns implies that the principal components are fairly sensitive to the illumination. The first principal component sound to capture shadows as important features in the face images. this in turn can degrade the performance of face recognition algorithm when luminance of images are different or they captured from different angles. Another important point is that this method is fast and can be used in real time once the principal components are captured.

2. (5 points) Now we will perform a simple face recognition task.

   Face recognition through PCA is proceeded as follows. Given the test image subject01-test.gif and subject02-test.gif, first downsize by a factor of 4 (as before), and vectorize each image. Take the top eigenfaces of Subject 1 and Subject 2, respectively. Then we calculate the *normalized inner product score* of the 2 vectorized test images with the vectorized eigenfaces:

$$s_{ij} = \frac{(\mathsf{eigenface})_i^T (\mathsf{test\ image})_j}{\|(\mathsf{eigenface}_i)\| \cdot \|(\mathsf{test\ image})_j\|}$$

   Report all four scores: s_{ij}, $i = 1, 2$, $j = 1, 2$. Explain how to recognize the faces of the test images using these scores. Explain if face recognition can work well and discuss how we can improve it, possibly.
   Answer: We only consider the top eignefaces and compute normalized inner product score for each test image based on that. This score is in fact a measure of correlation. Therefore, higher correlation rates will make us assign a test image to a specific identity. Calculated scores are: $s_{11} = -0.8765$, $s_{12} = -0.6989$, $s_{21} = 0.0939$ and $s_{22} = 0.4182$. This clearly indicates that the top eigenface and test image vector for a single subject are correlated more than those for different subjects. Therefore, when we are performing a *face recognition* task, we have to assign the new image to a subject with which it has a higher absolute value correlation.
   The algorithm works perfectly for this specific dataset. However, it could be improved

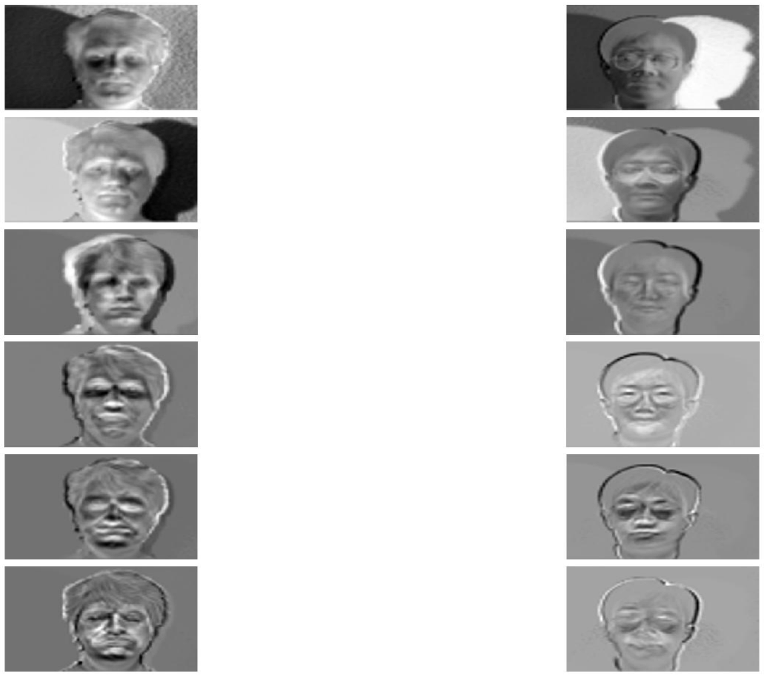


Figure 8: Visualization of weight vectors for two subjects in Yale face dataset. Left column shows the weight images for subject 1 and right column show the weight images for subject number 2. These weight images correspond ti the top 6 principal components. The first row shows the eigenvectors corresponding to the largest eigenvalue, the second row shows the eigenvectors corresponding to the second largest eigenvalue and so on and so forth.

using more than one eigenface for discriminating different faces. Also, one other way to improve the algorithm is to build faces with different angles and different luminances and find the principal components including those images in dataset. In this case, results tend to be more robust. These days' famous deep learning structures are known to account for all these variability.