ISYE 6740 Spring 2024
Homework 2
(100 points + 5 bonus points)

Yuxi Chen

1. Conceptual questions [30 points].

1. (5 points) Please prove the first principle component direction v corresponds to the largest eigenvector of the sample covariance matrix:

$$v = \arg \max_{w:\|w\|\leq 1} \frac{1}{m} \sum_{i=1}^{m} (w^T x^i - w^T \mu)^2.$$

You may use the proof steps in the lecture, but please write them logically and cohesively.

Start by forming the Lagrangian function

$$L(w, \lambda) = w^T C w - \lambda(1 - ||w||^2)$$

If w is the max, based on the original problem then there exists λ, where (w, λ) is a stationary point of $L(w, \lambda)$, therefore:

$$\frac{\partial L}{\partial w} = 0 = 2Cw - 2\lambda w$$

Where optimal solution w is the eigenvector of C, and λ is the corresponding eigenvalue. So we get,

$$\Rightarrow w^T C w = \lambda w^T w = \lambda ||w||^2$$

The problem now becomes, finding the largest eigenvalue of C.

2. (5 points) Based on your answer to the question above, explain how to further find the third largest principle component directions.

To find the third largest princple component directions, we need to remove the influence of the first and largest principle component. We can do so by subtracting datapoints and eigenvectors associated with w_1. This creates a new dataset in which we can perform PCA on, to find the second largest principle component, w_2. We can repeat the process and find w_3 after removing w_2

3. (5 points) Based on the outline given in the lecture, show that the maximum likelihood estimate (MLE) for Gaussian random variable using observations $x^1, \ldots, x^m$, that are *i.i.d.* (independent and identically distributed) following the distribution $\mathcal{N}(\mu, \sigma^2)$, and the mean and variance parameters are given by

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} x^i, \quad \hat{\sigma}^2 = \frac{1}{m} \sum_{i=1}^{m} (x^i - \hat{\mu})^2,$$

respectively. Please show the work for your derivations in full detail.

4. (5 points) Explain the three key ideas in ISOMAP (for manifold learning and non-linear dimensionality reduction).

   The key idea of ISOMAP is to produce low dimensional representation which preserves "walking distance" over the manifold. This can be accomplished three ways:
   1. Find $N(i)$ of each data point, x^i, within distance ε and let A be the adjacency matrix recording neighbor Euclidean distance.
   2. Find shortest path distance matrix D between pairs of points, x^i and x^j, based on A.
   3. Find low dimensional representation which preserves the distance information in D.

5. (5 points) Explain how to decide k, the number of principle components, from data.

   There are various ways to decide the number of principle components. You can use a scree plot, which returns the eigenvalue to its corresponing principle component. Visually, you can look for an elbow curve when the eigenvalue creates a sharp decrease when increasing the principle component you can decide to stop there. Another method, is cross validation, where you decide on a threshold for variance and choose a starting k value until it reaches the threshold.

6. (5 points) How do outliers affect the performance of PCA? You can create numerical examples to study and show this.

   PCA is known to be sensitive to outliers. Outliers introduces noise and can distort the covariance matrix causing the direction of the eigenvector to be shifted. In the numerical example provided below, array x2 contains an outlier, which changes the value of the eigenvector and eigenvalue produced from array x1.

In [14]:
```
import numpy as np
```

In [15]:
```
x1 = np.array([[1, 2], [3, 4], [5, 6], [7,8]])
x2 = np.array([[1, 2], [2, 3], [5,6], [70, 80]])

evalue_x1, evector_x1 = np.linalg.eig(np.cov(x1))
evalue_x2, evector_x2 = np.linalg.eig(np.cov(x2))
```

In [16]:
```
print("eigenvalue for x1:", evalue_x1)
print("eigenvalue for x2:", evalue_x2)
```

```
eigenvalue for x1: [-2.22044605e-16  2.00000000e+00  0.0000000
0e+00  0.00000000e+00]
eigenvalue for x2: [-7.10542736e-15  5.15000000e+01  0.0000000
0e+00  9.58658933e-16]
```

In [17]:
```
print("eigenvector for x1:", evector_x1)
print("eigenvector for x2:", evector_x2)
```

```
eigenvector for x1: [[-8.66025404e-01  5.00000000e-01 -3.71854
204e-18 -3.71854204e-18]
 [ 2.88675135e-01  5.00000000e-01 -5.77350269e-01 -5.77350269e
-01]
 [ 2.88675135e-01  5.00000000e-01  7.88675135e-01 -2.11324865e
-01]
 [ 2.88675135e-01  5.00000000e-01 -2.11324865e-01  7.88675135e
-01]]
eigenvector for x2: [[-0.99513379  0.09853293  0.         -0.0
8214331]
 [ 0.00975621  0.09853293  0.09901475 -0.69694538]
 [ 0.00975621  0.09853293 -0.99107935 -0.69694538]
 [ 0.09756214  0.98532928  0.08920646  0.14760341]]
```

2. PCA: Food consumption in European countries [20 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.

(a) (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). 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 plot (you can do this by hand if you want). Please explain any pattern you observe in the scatter plot.

The dataset provided contains 16 countries(rows), and 20 different foods(columns). We set the attributes of the 16 x 20 matrix to that of the values corresponding to each country and food. Since we will be considering each country's food consumption as the "feature" vector we will need to transpose the attributes before performing PCA.

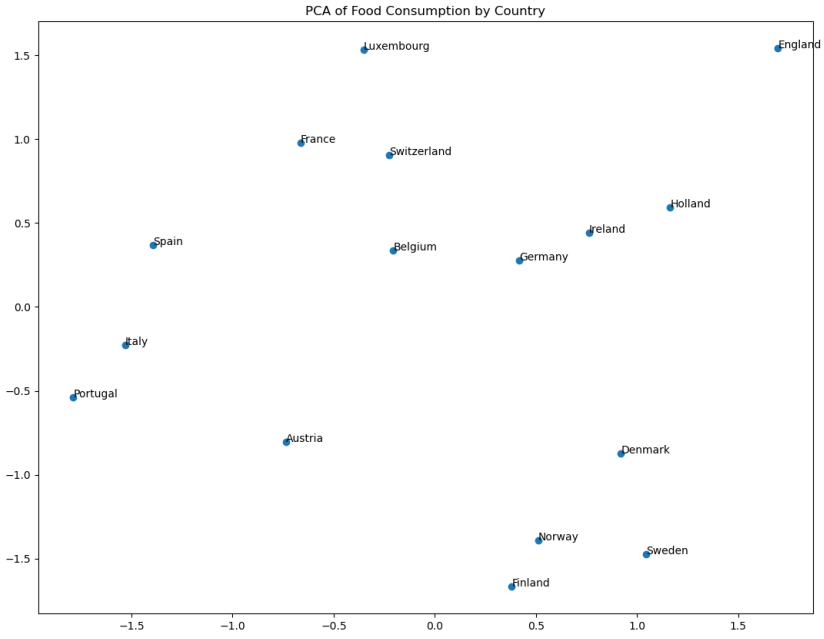

Figure 1: PCA of Food Consumption by Country

In Figure 1, there are certain grouping of countries, such as Sweden, Norway and Finland who have similar food preferences. A quick assumption one may assume can be due to their geographical proximity. It can also be seen that Luxembourg, has different preferences in comparison to this group of

countries. Lastly, countries like England and Austria seem to have the further distances from other countries indicating more unique food preferences.

(b) (10 points) 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. 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 hand if you want). Please explain any pattern you observe in the scatter plot.

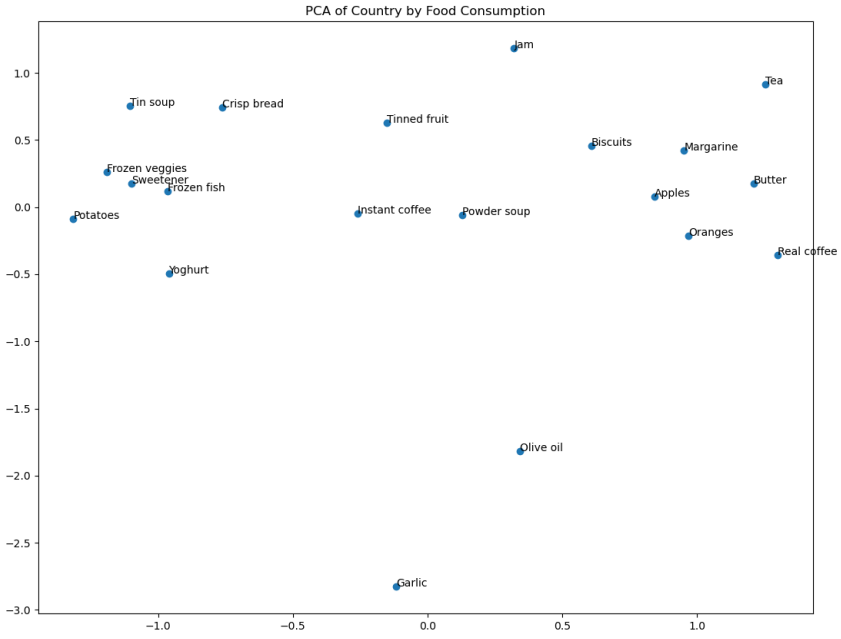

Figure 2: PCA of Country by Food Consumption

In Figure 2, foods such as margarine, butter and biscuits are grouped together. Indicating that countries that eat those butter for instance, also enjoy eating biscuits and margarine. Whereas, garlic and olive oil are the closest to each other but furthest from the rest of the grouping of foods. Such that having garlic as a food preference does not have a significant impact on other food type preferences.

3. Order of faces using ISOMAP [25 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 of the same face. Each image is given as a 64×64 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 find the shortest path (required by one step of the algorithm), using `https://scikit-learn.org/stable/modules/generated/sklearn.utils.graph_shortest_path.graph_shortest_path.html`.

Using Euclidean distance (i.e., in this case, a distance in $\mathbb{R}^{4096}$) to construct the ϵ-ISOMAP (follow the instructions in the slides.) You will tune the ϵ parameter to achieve the most reasonable performance. Please note that this is different from K-ISOMAP, where each node has exactly K nearest neighbors.

(a) (5 points) Visualize the nearest neighbor graph (you can either show the adjacency matrix (e.g., as an image), or visualize the graph similar to the lecture slides 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).

(b) (10 points) Implement the ISOMAP algorithm yourself to obtain a two-dimensional low-dimensional embedding. 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 and specify the face locations on the scatter plot. Comment on do you see any visual similarity among them and their arrangement, similar to what you seen in the paper?

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

4. Eigenfaces and simple face recognition [25 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.

(a) (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?

In both subjects, it can be seen that Eigenface 1 contains the most detailed face and Eigenface 6 contains the least detailed face. We can attribute this to amount of variance in each Eigenface, which decreases as we progress from Eigenface 1 to Eigenface 6.

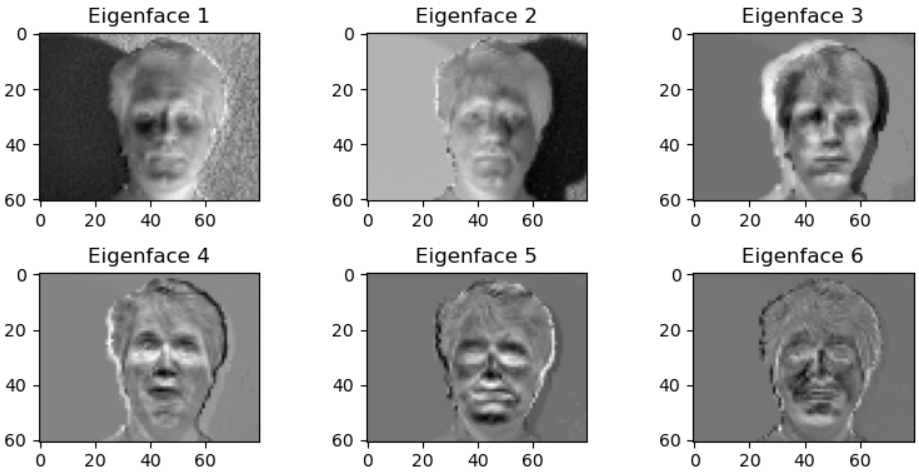

Figure 3: Eigenfaces of Subject01

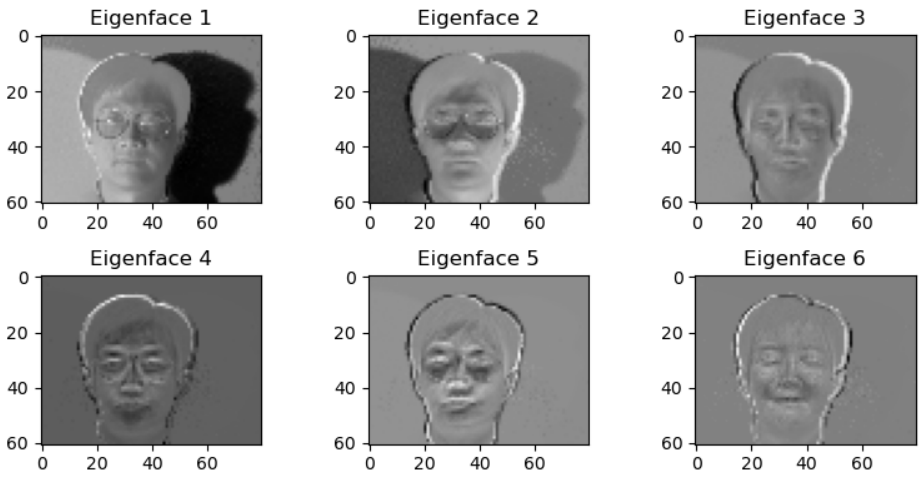

Figure 4: Eigenfaces of Subject02

(b) (10 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 *projection residual* of the 2 vectorized test images with the vectorized eigenfaces:

$$s_{ij} = \|(\text{test image})_j - (\text{eigenface}_i)(\text{eigenface})_i^T(\text{test image})_j\|_2^2$$

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.

```
s_11: 6091271.162235053
s_21: 6267943.445194438
s_12: 4503195.206097447
s_22: 2560011.928705806
```

Figure 5: Scores

If we look at the score values produced for subject01-test represented by s_{11} and s_{21}, with values of 6091271 and 6267943 respectively. It can be noted that s_{11} has a lower residual score than s_{21}, indicating that it is a closer match to that of the subject01 test image. This can also be applied to s_{12} and s_{22} where, s_{22} has a lower residual value indicating it to likely be subject02 test image.

(c) (5 points) Comment if your face recognition algorithm works well and discuss how you would like to improve it if possible.

The face recognition algorithm works well as we are able to differentiate the faces based on the residual score. We can increase the number of images provided to train and test the algorithm. Another method we can perform is if we can increase the number principle components or number of eigenfaces in the original algorithm as this will give us higher eigenvalues. When fitted into the algorithm it will provide additional information to reduce the residual score, however, we do have to account for overfitting.

5. To subtract or not to subtract, that is the question [Bonus: 5 points].

In PCA, we have to subtract the mean to form the covariance matrix

$$C = \frac{1}{m}\sum_{i=1}^{m}(x^i - \mu)(x^i - \mu)^T$$

before finding the weight vectors, where $\mu = \frac{1}{m}\sum_{i=1}^{m} x^i$. For instance, we let

$$Cw^1 = \lambda_1 w^1$$

where λ_1 is the largest eigenvalue of C, and w^1 is the corresponding largest eigenvector.

Now suppose Prof. X insisting not subtracting the mean, and uses the eigenvectors of

$$\tilde{C} = \frac{1}{m}\sum_{i=1}^{m} x^i x^{iT}$$

to form the weight vectors. For instance, she lets $\tilde{w}^1$ to be such that

$$\tilde{C}\tilde{w}^1 = \tilde{\lambda}_1 \tilde{w}^1$$

where $\tilde{\lambda}_1$ is the largest eigenvalue of $\tilde{C}$.

Now the question is, are they the same (with and without subtract the mean)? Is w^1 equal or not equal to $\tilde{w}^1$? Use mathematical argument to justify your answer.