Homework 2

1. Conceptual questions

1. We can start with the function to be proved:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m}\left(w^T x^i - w^T \mu\right)^2$$

Let's transform and expand it, as below:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m}\left(w^T (x^i - \mu)\right)^2 = w^T \left(\frac{1}{m}\sum_{i=1}^{m}(x^i - \mu)(x^i - \mu)^T\right) w$$

$$\Rightarrow J(w) = w^T C w$$

Where C is the sample covariance matrix. Next we can impose the constraint $\|w\|^2 = 1$:

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

Then we get:

$$\frac{\partial L}{\partial w} = 0$$

$$\Rightarrow\ 0 = 2Cw - 2\lambda w$$

$$\Rightarrow\quad Cw = \lambda w$$

Hence we can prove the first principle component direction v corresponds to the largest eigenvector of the sample covariance matrix.

2. To further find the second largest principal component direction, we can first perform eigenvalue decomposition of the sample covariance matrix. The first principal component v_1 is the eigenvector corresponding to the largest eigenvalue λ_1, capturing the maximum variance in the data.

   The second principal component v_2 is the eigenvector corresponding to the second largest eigenvalue λ_2. Since the eigenvectors are orthogonal, v_2 is orthogonal to v_1, and it captures the second most variance in the data in a direction different from v_1.

3. Using observations $x^1, \ldots, x^m$, that are *i.i.d.* following the distribution $\mathcal{N}(\mu, \sigma^2)$, we can create the likelihood as below:

$$L(\mu, \sigma^2, x^1, x^2, \ldots, x^m) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x^i - \mu)^2}{2\sigma^2}\right)$$

We can simplify it:

$$L(\mu, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{m}(x_i - \mu)^2\right)$$

Then convert the function to log:

$$\ell(\mu, \sigma^2) = -\frac{m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(x^i - \mu)^2$$

Next we are able to find the maximum likelihood estimate for μ by taking the derivative of $\ell(\mu, \sigma^2)$ with respect to μ:

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

Then set the value to 0:

$$\sum_{i=1}^{m}(x^i - \mu) = 0$$

$$\Rightarrow \mu = \frac{1}{m}\sum_{i=1}^{m} x^i$$

The next step is to find the maximum likelihood estimate for σ^2, we can take the derivative of $\ell(\mu, \sigma^2)$ with respect to σ^2:

$$\frac{\partial \ell(\mu, \sigma^2)}{\partial \sigma^2} = -\frac{m}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{m}(x^i - \mu)^2$$

Set the above value to 0, then we can get the following:

$$\Rightarrow \sum_{i=1}^{m}(x_i - \mu)^2 = m\sigma^2$$

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

Hence we can get the following maximum likelihood of estimate:

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^i$$

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

4. ISOMAP is a method for dimension reduction for non-linear high dimensional data. It begins by finding neighbors N_i of each data point within a distance ϵ, and constructs an adjacency matrix A that records the Euclidean distances between the data points. The matrix captures the local structure by connecting each data point to its nearby neighbors within the specified distance, as mentioned in the lecture.

   After the nearest-neighbor graph is built, ISOMAP computes the shortest path distance matrix D between all pairs of points x^i and x^j based on the adjacency matrix A. The shortest paths approximate the geodesic distances on the manifold, representing the intrinsic distances between data points.

   After the steps above, ISOMAP uses Multidimensional Scaling (MDS) to find low dimensional representation that preserves the distance information in the matrix to map the data points into a lower dimensional space, to minimize the distances between the points. The geodesic distances are embedded into a lower dimensional space by preserving pairwise distances as closely as possible. MDS constructs a lower dimensional representation that minimizes the difference between the data distances in the lower dimensional space and the distances computed in the last step.

5. To determine the appropriate number of principal components k from the data, a common method is to analyze the eigenvalues of the covariance matrix, which represent the variance explained by each principal component. The goal is to select k such that the selected components capture the majority of the variance in the data, reducing its dimensionality while retaining important information. Principal components associated with larger eigenvalues are prioritized, as they explain more variance in the dataset.

   We can also leverage plot to find the number of principal components k in PCA. We can plot eigenvalues of the principal components (in descending order). Based on the plot, we can locate the point where the decrease in eigenvalues begins to slow down and stabilize. This "elbow" point is the number of principal components k

   Another way to decide k is to cross validate different numbers of principal components. First we can perform PCA with different valuses of k. Then we train and validate each of the algorithms using the corresponding k value. Based on the metrics of performances, we can then choose the best result as the number of principal component k

   [backend=biber]biblatex hyperref

References

[1] https://stackoverflow.com/questions/12067446/how-many-principal-components-to-take

[2] https://towardsdatascience.com/pca-102-should-you-use-pca-how-many-components-to-use-how-to-interpret-them-da0c8e3b11f0

6. graphicx float

We can create two datasets using Python to check the outlier impact to PCA performances. The first dataset is without outlier, the second dataset is with outlier. As shown in Figure 1 in the next page, we evaluated the performance of PCA using two key metrics: the explained variance ratio and the reconstruction error. The explained variance ratio indicates the proportion of the total variance in the dataset that is captured by each principal component. It is calculated by dividing the eigenvalues of the principal components by the sum of all eigenvalues. This metric provides insight into how well the PCA reduces dimensionality while retaining the most significant variance of the original data. The reconstruction error measures the difference between the original data and its reconstruction from the lower dimensional representation. It is computed using the mean squared error between the original and the reconstructed datasets, reflecting how well PCA can reconstruct the original data space after dimension reduction.

The output shows that with outliers present the first principal component explains a slightly lower proportion of the variance, and the second component (with outlier) accounts for a more significant fraction when outliers are included. This shift indicates that outliers affect the distribution of variance among principal components. Additionally, the reconstruction error increases with outliers, though it remains very small. This increase suggests that the presence of outliers slightly degrades the reconstruction quality, meaning that PCA with outliers introduces some loss of information in reconstructing the original data. Overall, while PCA remains effective, the presence of outliers has a minor but noticeable impact on its performance.

References

[1] VanderPlas, J. *Python Data Science Handbook: Principal Component Analysis.* Available at: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

[2] Hadad, Y. *Detect Outliers with PCA*. Available at: https://www.kaggle.com/code/yairhadad1/detect-outliers-with-pca

[3] GeeksforGeeks. *Reduce Data Dimensionality Using PCA in Python.* Available at: https://www.geeksforgeeks.org/reduce-data-dimentionality-using-pca-python/

[59]:
```
# Define datasets for two scenarios for comparison
X_wo_outliers = np.array([[1, 1], [2, 2], [3, 3], [4, 4], [5, 5]])
X_outliers = np.array([[1, 1], [2, 2], [3, 3], [4, 4], [5, 5], [12, 16]])
# Scale the datasets
scaler_wo_outliers = StandardScaler()
scaler_outliers = StandardScaler()
X_wo_outliers_scaled = scaler_wo_outliers.fit_transform(X_wo_outliers)
X_outliers_scaled = scaler_outliers.fit_transform(X_outliers)

# Use PCA to fit the scaled datasets and compute explained variance ratio
pca_wo_outliers = PCA()
pca_outliers = PCA()
pca_wo_outliers.fit(X_wo_outliers_scaled)
pca_outliers.fit(X_outliers_scaled)

# Explained Variance Ratio
explained_variance_ratio_wo_outliers = pca_wo_outliers.explained_variance_ratio_
explained_variance_ratio_outliers = pca_outliers.explained_variance_ratio_

# Print Explained Variance Ratios
print("Explained Variance Ratio Without Outliers:\n", explained_variance_ratio_wo_outliers)
print("Explained Variance Ratio With Outliers:\n", explained_variance_ratio_outliers)

# Compute Reconstruction Error
# Transform data to lower dimensions and back to original space
X_wo_outliers_reconstructed = pca_wo_outliers.inverse_transform(pca_wo_outliers.transform(X_wo_outliers_scaled))
X_outliers_reconstructed = pca_outliers.inverse_transform(pca_outliers.transform(X_outliers_scaled))

# Compute Mean Squared Error between original and reconstructed data
reconstruction_error_wo_outliers = mean_squared_error(X_wo_outliers_scaled, X_wo_outliers_reconstructed)
reconstruction_error_outliers = mean_squared_error(X_outliers_scaled, X_outliers_reconstructed)

# Print Reconstruction Errors
print("Reconstruction Error Without Outliers:", reconstruction_error_wo_outliers)
print("Reconstruction Error With Outliers:", reconstruction_error_outliers)
```

```
Explained Variance Ratio Without Outliers:
 [1.00000000e+00 6.23711839e-34]
Explained Variance Ratio With Outliers:
 [0.99714028 0.00285972]
Reconstruction Error Without Outliers: 2.465190328815662e-32
Reconstruction Error With Outliers: 9.334741699201884e-32
```

Table 1: Outlier Impact

2. PCA: Food consumption in European countries

1. As shown in Figure 1 in the next page, the scatterplot shows the food consumption patterns of different countries after reducing the dimensions using PCA. Each data point (green dot in the plot) represents a country, with its position reflecting the corresponding diet preferences.

   Based on the output, we observed that warmer climate countries such as Spain, Italy and Portugal tend to cluster in the right side of the plot, while colder countries are located at the left hand side. This suggests a potential link between climate and food consumption habits.

   We also noticed some outliers that are far away from the other clusters. Some countries such as England is on the top left hand side, suggesting unique diet habit. On the other hand, Belgium, Germany, and Ireland form a central cluster, implying similar food consumption patterns.

   Countries near each other share similar food consumption habits. e.g., Norway, Denmark, Switzerland, and Holland are closely grouped. On the other hand, countries further apart have more distinct preferences, a good example of this is Italy and Denmark. This suggests that climate, geography, and potentially cultural may all contribute to food consumption habits.

   hyperref

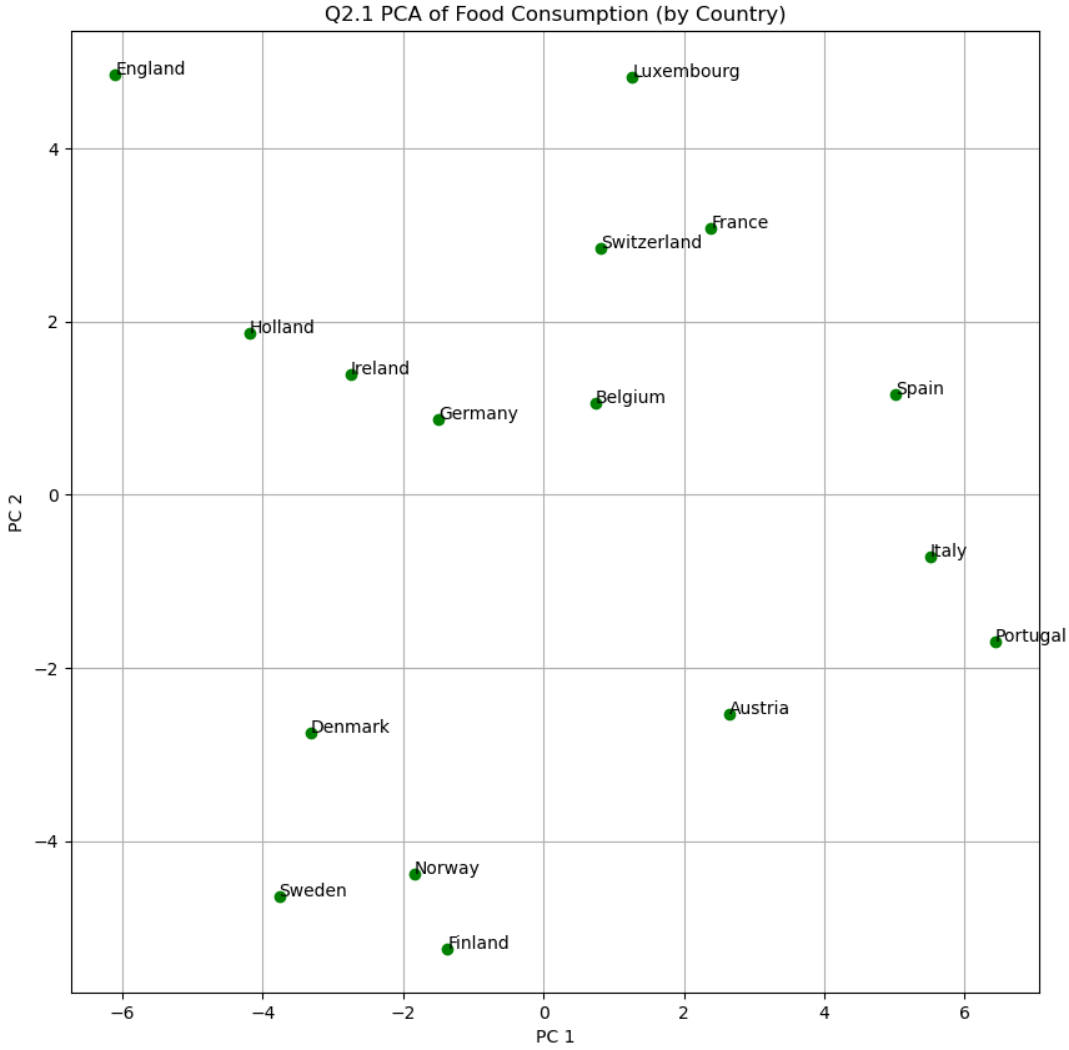

Figure 1: Q2.1 PCA of Food Consumption

2. As shown in Figure 2 below, the scatterplot represents the distribution of food items based on their principal component - country consumption. Each data point represents a food item. From the plot we observed a few interesting patterns, as below:

   First of all, similar food items tend to be in the same cluster. For example, frozen fish and frozen veggies foods are grouped together, oranges and apples are close to each other. On the other hand, tea and potatoes are far apart, which suggests they belong to different groups.

   The second thing we noticed is outliers. Garlic appears to be one of the outliers, as it is sitting on the top side of the plot, not close to any cluster, suggesting garlic is more extreme food preference. Similar to olive oil, it is isolated to any other food items, suggesting it could be a very unique food preference. either of them can be the most favorable food item (or least favorable item).

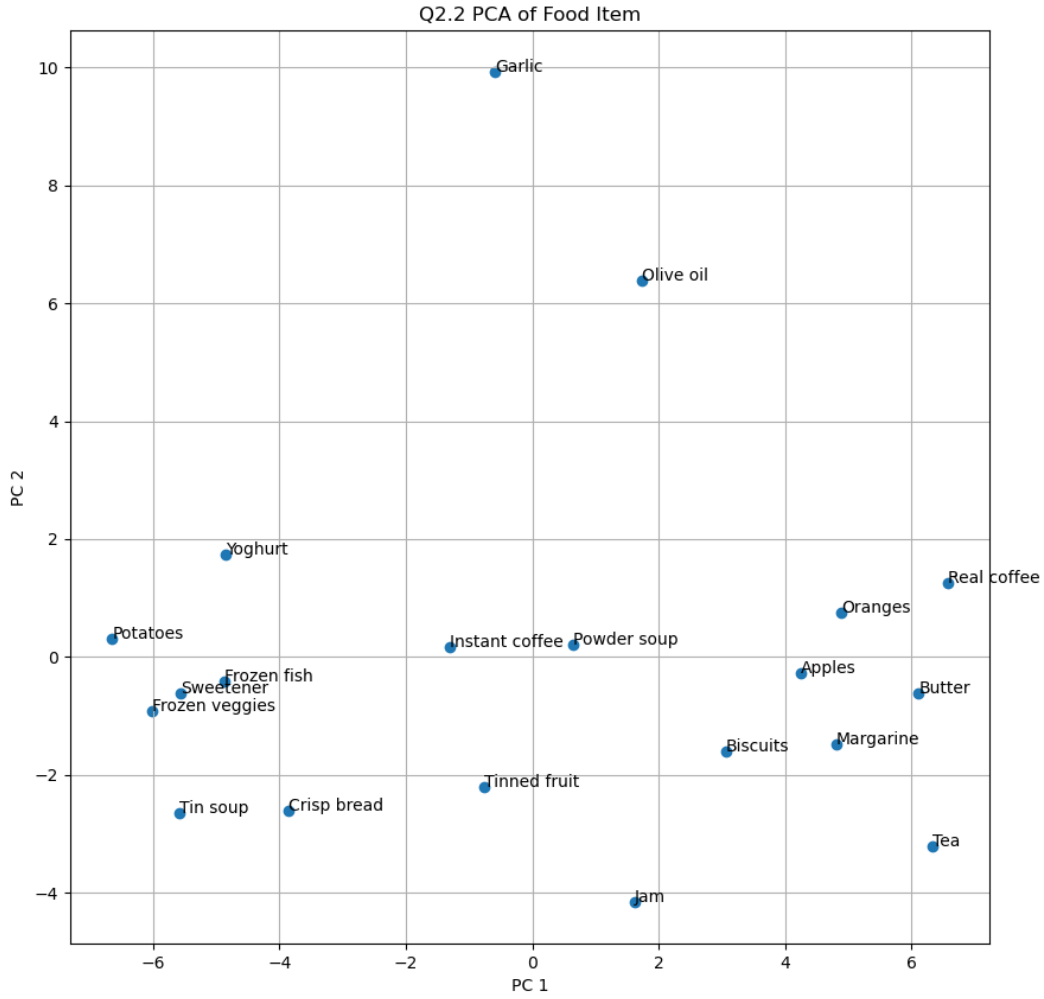

Figure 2: Q2.2 PCA of Food Item

3. Order of faces using ISOMAP

1. As shown in Figure 3 of the adjacency matrix, it visualizes how the images correspond to nodes at different parts. The ϵ of the matrix is 600.

   float

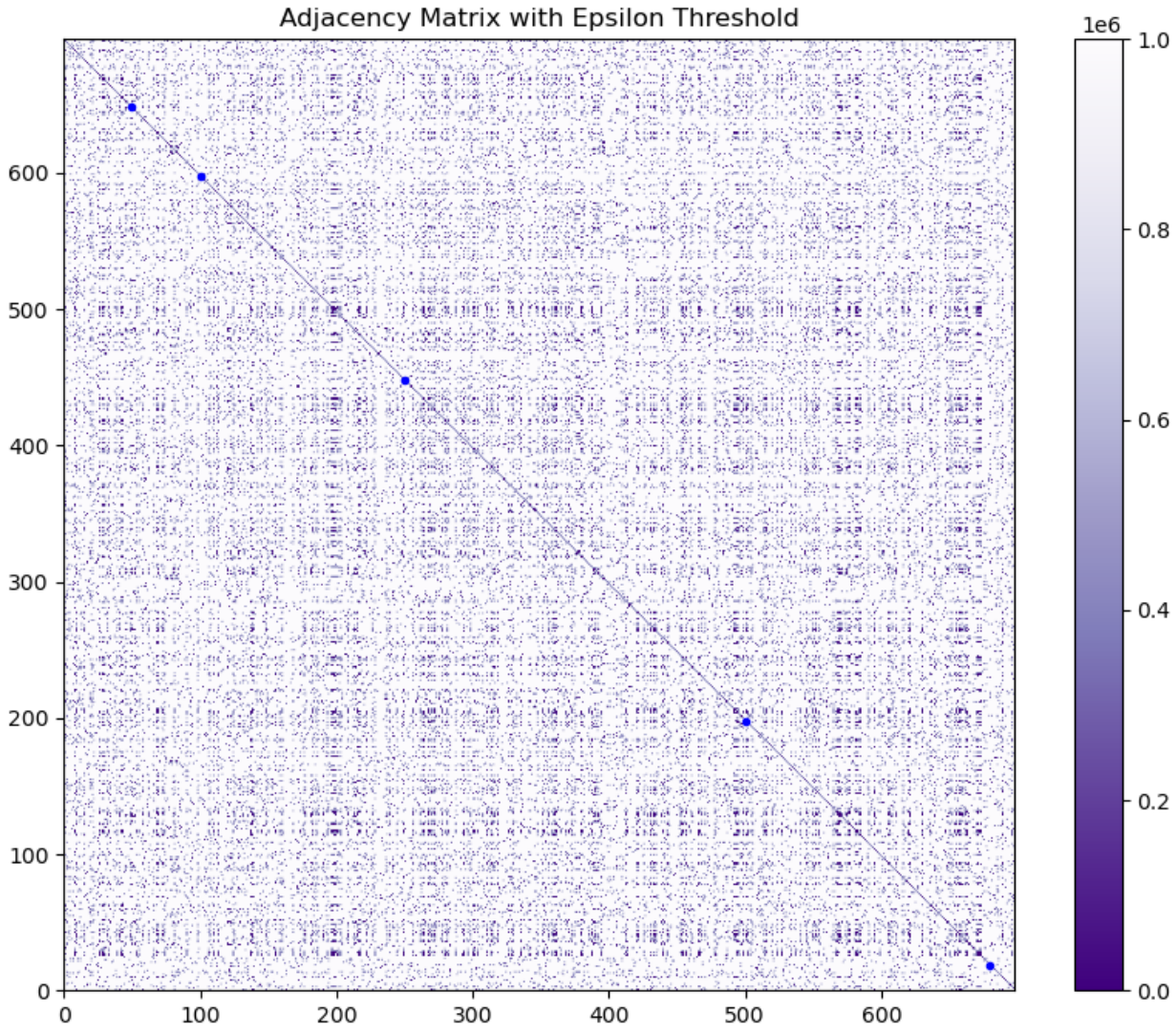

Figure 3: Q3.1 Adjacency Matrix

2. We implemented algorithm to calculate the shortest path between each pair of images using adjacency matrix that derived from Euclidean distances. Then we created kernel matrix using geodesic distances and obtained the eigenvalues and eigenvectors. Then we selected the eigenvectors corresponding to the largest two eigenvalues and created the following two dimensional plot. Figure 4 ISOMAP shows the two dimensional scatter plot, with specific face locations embedded on the plot.

   As shown in the figure, each point in the scatter plot represent an image, as well as the distance between points corresponding to the geodesic distance of each pair of the images. We observed that image that are closer to each other tend to have similar visuals. For example, faces that facing different directions tend to cluster together. Faces facing left tend to cluster on the left side of the plot, while faces facing right cluster on the right side. Faces facing front tend to be located in the middle. Another observation is faces with more contrast with noticeable shadows appear on the upper side of the plot, while faces with more exposure with fewer shadows tend to be located on the lower side.

   By observing the locations of images on the plot, we can see that ISOMAP was able to captures the underlying face manifold pretty well. The observed clustering patterns on face orientation and lighting and exposure both effectively represent the non-linear relationships between the images.

   float

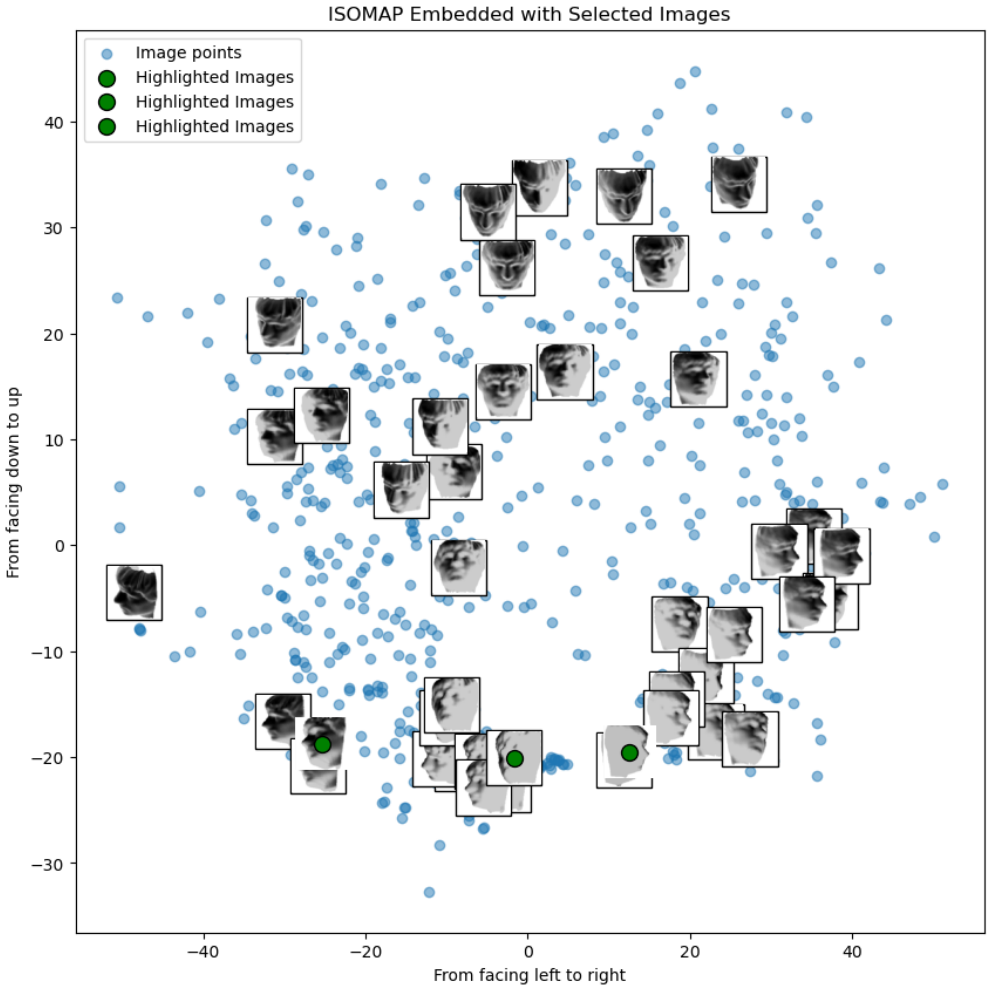

Figure 4: Q3.2 ISOMAP with highlighted image indices: [202, 137, 515]

3. When perform PCA on the images and project them into the top 2 principal components, we were able to get the following scatter plot. As shown in Figure 5 below, the difference between the patterns of the clusters or groupings are not as significant as the scatter plot of ISOMAP. We can see that there are some images with more contrast grouped together. However, on the other side of the plot, there are a similar group, without noticeable difference than the first group. Same applies to the images with more exposures (less shadow), unlike the obvious pattern in ISOMAP scatter plot, we can see they are located in separated groups and each group is far from the other.

   Based on the above observation, it is reasonable to conclude that this is due to the different nature of PCA and ISOMAP. PCA is more applicable of handling linear relationships between data points. However, faces with variations of poses, orientations, lighting conditions etc. are not very typical linear structure. Therefore, in this case, ISOMAP seems to be better handles the situation and provided better results of face image pattern recognition. It is likely because ISOMAP uses intrinsic geometry of the manifold, which is more effective to represent the relationships between the images for the face dataset.

   float

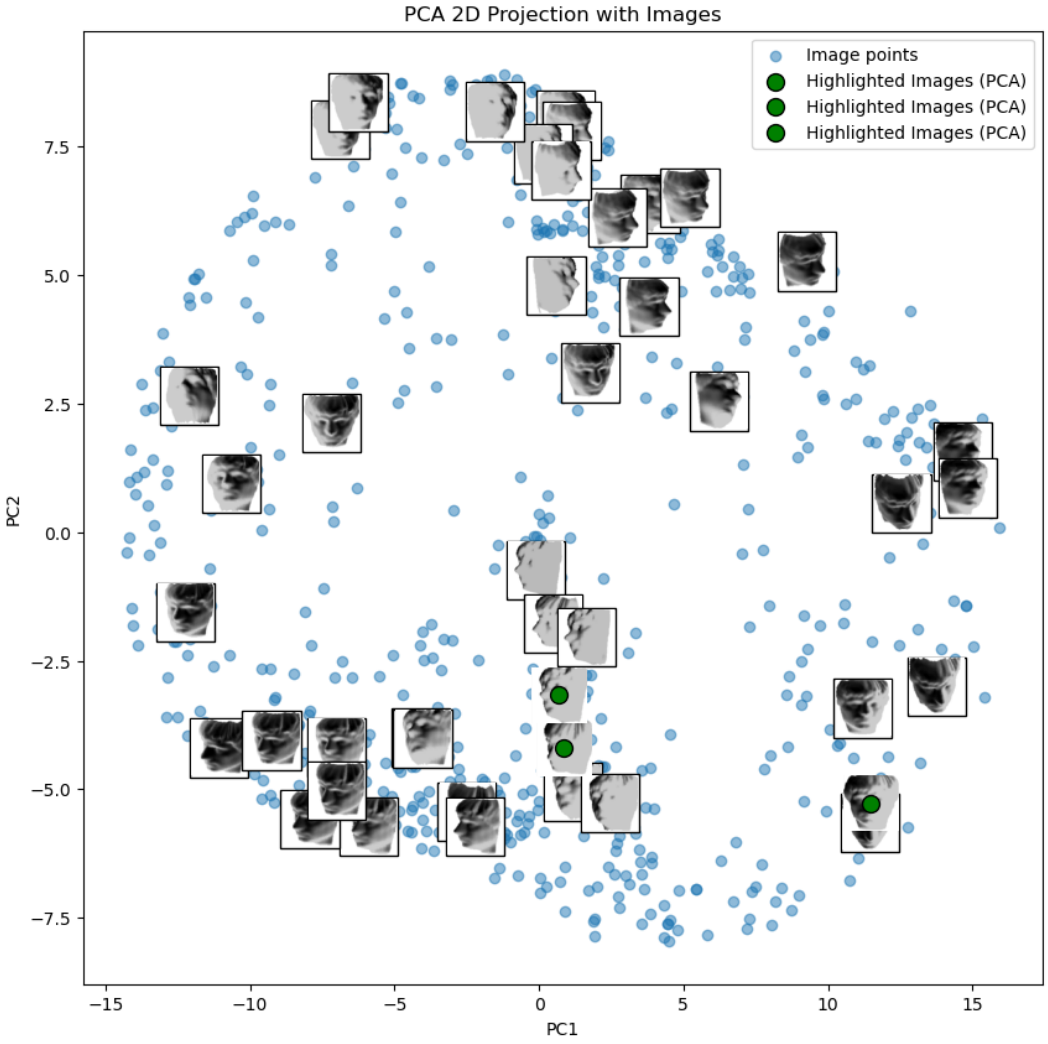

Figure 5: Q3.3 PCA with highlighted image indices (PCA): [420, 590, 169]

4. Eigenfaces and simple face recognition

1. As shown below in Figure 6, the output of the first 6 eigenfaces for Subject 1 and Subject 2. We noticed the following patterns while analyzing the top 6 eigenfaces for both Subject 1 and Subject 2 from the Yale face dataset. Fist of all, we can easily tell the facial expressions in both Subjects. For subject 1, we can see Eigenface 0, 1, 2 are neutral normal faces, while Eigenface 3 is surprised, Eigenface 4 is smiling while Eigenface 5 is sleepy. On the other hand, in Subject 2, Eigenface 0 through Eigenface 3 seem to be neutral, while Eigenface 4 is smiling, Eigenface 5 is sleepy.

   Secondly, the first two eigenfaces (Eigenfaces 0 and 1) from each Subject capture the most significant contrast, which are likely influenced by lighting conditions. Whereas, Eigenfaces 2 through 5 from Subject 1, show more details compared to the corresponding eigenfaces from Subject 2, likely due to differences in facial structure.

   Another thing is that glasses are more noticeable in Subject 2, where we can clearly see 3 images (Eigenface 0, 1, 3) with glasses. Meanwhile, none detected in Subject 1. This suggests that for Subject 1, the presence of glasses didn't significantly contribute to the eigenfaces, hence this variation is not well captured in the eigenfaces.

   float

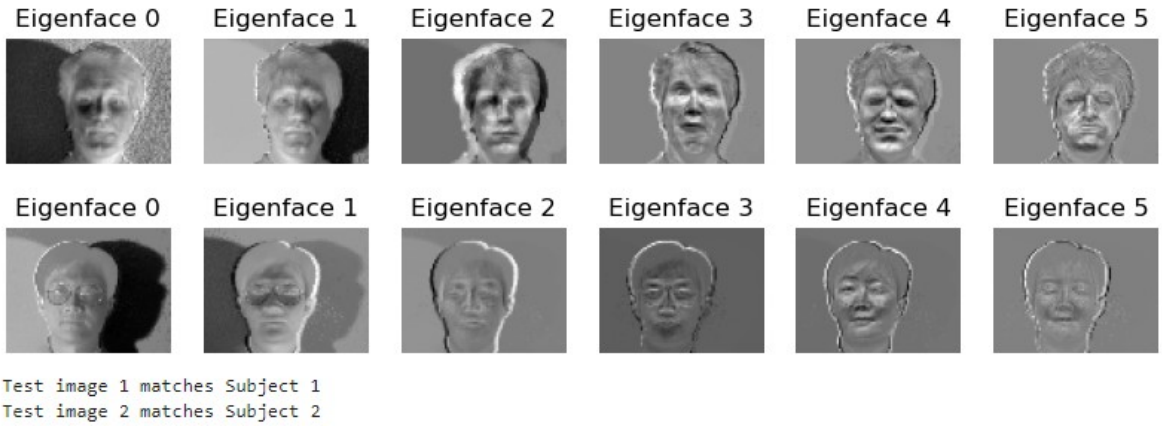

Figure 6: Q4.1 Eigenfaces

2. By performing face pattern recognition by downsizing and vectorizing the test images subject01-test.gif and subject02-test.gif, then calculating the projection residuals using the top eigenfaces of Subject 1 and Subject 2. The residuals indicate how well each test image matches the eigenfaces of each subject. The smaller the residual, the better match of the images. The result outputs are as follows:

	j=1	**j=2**
Subject 1	6742872.4473	32746524.6021
Subject 2	39491807.8305	3735545.1083

   Based on the residual scores, the face recognition algorithm correctly identifies subject01-test.gif as Subject 1 and subject02-test.gif as Subject 2. Test image 1 has the smallest residual for Subject 1, and test image 2 has the smallest residual for Subject 2, proves that using eigenfaces is effective in this case.

3. The face recognition algorithm works well for the given test images, as it correctly identifies subject01-test.gif as Subject 1 and subject02-test.gif as Subject 2 based on the smallest projection residuals. This demonstrates that the top eigenfaces capture key variations that distinguish between the subjects effectively in this scenario.

   However, there is room for improvement. The current method uses only the top eigenface for each subject, which may not capture enough variation for more complex cases, such as when there are significant changes in lighting, facial expression, or occlusions like glasses. To improve the accuracy, we could increase the number of eigenfaces used in the recognition process, as this would better capture subtle differences between the test images and the training set.