

ESE 5420 Homework 6

Mohammed Raza Syed - Penn ID: 37486255

Problem 1

Consider the following four points on the 2-D plane, where t is a positive number larger than 3:

$$\{(-1, -t), (+1, -t), (-1, t), (+1, t)\}$$

We want to use K-Means to cluster these points into two clusters.

(a) Optimal Centers for the 2-Means Problem

The objective is to find two centers that minimize the 2-Means objective function over the given points.

Step 1: Visualization of the Points

The points form a rectangle on the plane:

$$\{(-1, -t), (+1, -t), (-1, t), (+1, t)\}$$

with a width of 2 and a height of $2t$.

Step 2: Horizontal Splitting

To minimize the sum of squared distances (SSD), the points are split horizontally into two clusters:

- Cluster 1: $(-1, -t), (+1, -t)$
- Cluster 2: $(-1, t), (+1, t)$

Calculating the Centers:

- Center of Cluster 1:

$$\text{Mean of } \{(-1, -t), (+1, -t)\} = \left(\frac{-1 + 1}{2}, -t \right) = (0, -t)$$

- Center of Cluster 2:

$$\text{Mean of } \{(-1, t), (+1, t)\} = \left(\frac{-1 + 1}{2}, t \right) = (0, t)$$

Calculating the Distances: For Cluster 1:

$$\text{Distance from } (-1, -t) \text{ to } (0, -t) = \sqrt{(0 - (-1))^2 + (-t - (-t))^2} = \sqrt{1^2} = 1$$

$$\text{Distance from } (+1, -t) \text{ to } (0, -t) = \sqrt{(0 - (+1))^2 + (-t - (-t))^2} = \sqrt{1^2} = 1$$

For Cluster 2:

$$\text{Distance from } (-1, t) \text{ to } (0, t) = \sqrt{(0 - (-1))^2 + (t - t)^2} = \sqrt{1^2} = 1$$

$$\text{Distance from } (+1, t) \text{ to } (0, t) = \sqrt{(0 - (+1))^2 + (t - t)^2} = \sqrt{1^2} = 1$$

Sum of Squared Distances (SSD):

$$\text{SSD}_{\text{horizontal}} = (1^2 + 1^2) + (1^2 + 1^2) = 4$$

Step 3: Vertical Splitting

If the points are split vertically, the clusters are:

- Cluster 1: $(-1, -t), (-1, t)$
- Cluster 2: $(+1, -t), (+1, t)$

Calculating the Centers:

- Center of Cluster 1:

$$\text{Mean of } \{(-1, -t), (-1, t)\} = (-1, 0)$$

- Center of Cluster 2:

$$\text{Mean of } \{(+1, -t), (+1, t)\} = (+1, 0)$$

Calculating the Distances: For Cluster 1:

$$\text{Distance from } (-1, -t) \text{ to } (-1, 0) = \sqrt{(-1 - (-1))^2 + (-t - 0)^2} = \sqrt{t^2} = t$$

$$\text{Distance from } (-1, t) \text{ to } (-1, 0) = \sqrt{(-1 - (-1))^2 + (t - 0)^2} = \sqrt{t^2} = t$$

For Cluster 2:

$$\text{Distance from } (+1, -t) \text{ to } (+1, 0) = \sqrt{(1 - 1)^2 + (-t - 0)^2} = \sqrt{t^2} = t$$

$$\text{Distance from } (+1, t) \text{ to } (+1, 0) = \sqrt{(1 - 1)^2 + (t - 0)^2} = \sqrt{t^2} = t$$

Sum of Squared Distances (SSD):

$$\text{SSD}_{\text{vertical}} = (t^2 + t^2) + (t^2 + t^2) = 4t^2$$

Conclusion for Part (a)

The horizontal splitting minimizes the SSD, with:

$$\text{SSD}_{\text{horizontal}} = 4$$

The vertical splitting results in a higher SSD:

$$\text{SSD}_{\text{vertical}} = 4t^2$$

Thus, the optimal centers are:

$$(0, -t) \text{ and } (0, t)$$

The goal of the K-Means algorithm is to minimize the sum of squared distances (SSD) between the points and their nearest cluster centers.

From the calculations in part (a), the SSD for horizontal splitting is:

$$\text{SSD}_{\text{horizontal}} = 4$$

whereas the SSD for vertical splitting is:

$$\text{SSD}_{\text{vertical}} = 4t^2$$

Since $t > 3$, it follows that $4t^2 > 4$. Thus, horizontal splitting has a lower SSD compared to vertical splitting.

Therefore, we select horizontal splitting as the optimal clustering solution, with the optimal centers being:

$$(0, -t) \text{ and } (0, t).$$

(b) Suboptimal Result of K-Means

Consider a poor initialization where the centers are:

- Center 1: $(-1, 0)$
- Center 2: $(+1, 0)$

Step 1: Initial Assignment of Points to Clusters

Given these initial centers, the points will be assigned to the nearest center based on the Euclidean distance:

- $(-1, -t)$ and $(-1, t)$ are closer to $(-1, 0)$, so they form Cluster 1.
- $(+1, -t)$ and $(+1, t)$ are closer to $(+1, 0)$, so they form Cluster 2.

This leads to the following clusters:

$$\text{Cluster 1: } \{(-1, -t), (-1, t)\}$$

$$\text{Cluster 2: } \{(+1, -t), (+1, t)\}$$

Step 2: Recalculating the Centers

The centers are updated as the mean of the points in each cluster:

- Center of Cluster 1:

$$\text{Mean of } \{(-1, -t), (-1, t)\} = (-1, 0)$$

- Center of Cluster 2:

$$\text{Mean of } \{(+1, -t), (+1, t)\} = (+1, 0)$$

Since the updated centers are the same as the initial centers, the algorithm has converged and will not make further updates.

Step 3: Sum of Squared Distances (SSD)

As calculated in part (a) for vertical splitting:

$$\text{SSD}_{\text{vertical}} = 4t^2$$

This is significantly higher than the optimal SSD for horizontal splitting:

$$\text{SSD}_{\text{horizontal}} = 4$$

Step 4: Why This Solution is Suboptimal

The K-Means algorithm converges to a **local minimum** due to its greedy nature:

- The algorithm assigns points to their nearest initial centers, which results in vertical clusters.
- The recalculated centers do not shift significantly, as they remain at $(-1, 0)$ and $(+1, 0)$.
- The algorithm stops once the assignments no longer change, even though this configuration does not minimize the overall SSD.

Conclusion for Part (b)

If initialized with $(-1, 0)$ and $(+1, 0)$, the K-Means algorithm converges to a suboptimal solution with:

$$\text{Centers: } (-1, 0) \text{ and } (+1, 0)$$

$$\text{SSD}_{\text{vertical}} = 4t^2$$

This result is suboptimal compared to the global minimum SSD of 4 achieved with horizontal splitting.

Problem 2

(a) Minimizing the Sum of Squared Euclidean Distances

Objective Function:

The given objective function is:

$$\sum_{k=1}^K \sum_{x \in C_k} \sum_{i=1}^p (c_{k,i} - x_i)^2$$

Where:

- K : Total number of clusters.
- C_k : The k -th cluster.
- $c_k = (c_{k,1}, c_{k,2}, \dots, c_{k,p})$: Centroid of the k -th cluster.
- $x = (x_1, x_2, \dots, x_p)$: Data points in p -dimensional space.

Simplify the Objective Function for a Single Cluster:

For a single cluster C_k , the objective can be written as:

$$\sum_{x \in C_k} \sum_{i=1}^p (c_{k,i} - x_i)^2$$

Optimization:

To minimize the function, we take the derivative with respect to each component $c_{k,i}$ of the centroid and set it to zero.

For the i -th component, the relevant part of the objective function is:

$$\sum_{x \in C_k} (c_{k,i} - x_i)^2$$

Take the derivative with respect to $c_{k,i}$:

$$\frac{\partial}{\partial c_{k,i}} \sum_{x \in C_k} (c_{k,i} - x_i)^2 = \sum_{x \in C_k} 2(c_{k,i} - x_i)$$

Set the derivative to zero:

$$\sum_{x \in C_k} 2(c_{k,i} - x_i) = 0$$

Simplify:

$$\begin{aligned} \sum_{x \in C_k} c_{k,i} - \sum_{x \in C_k} x_i &= 0 \\ |C_k|c_{k,i} &= \sum_{x \in C_k} x_i \end{aligned}$$

Solve for $c_{k,i}$:

$$c_{k,i} = \frac{1}{|C_k|} \sum_{x \in C_k} x_i$$

Here, $|C_k|$ denotes the number of points in the cluster C_k .

Conclusion:

Since this holds for all $i = 1, 2, \dots, p$, the centroid c_k is:

$$c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$$

Thus, the centroid is the mean of all points in the cluster.

(b) Minimizing the Sum of Manhattan Distances

Objective Function:

The given objective function is:

$$\sum_{k=1}^K \sum_{x \in C_k} \sum_{i=1}^p |c_{k,i} - x_i|$$

Simplify the Objective Function for a Single Cluster:

For a single cluster C_k , the objective can be written as:

$$\sum_{x \in C_k} \sum_{i=1}^p |c_{k,i} - x_i|$$

Focusing on a single component $c_{k,i}$, the relevant part of the objective function is:

$$\sum_{x \in C_k} |c_{k,i} - x_i|$$

Key Insight:

The sum of absolute differences (L_1 -norm or Manhattan distance) is minimized when $c_{k,i}$ is the **median** of the data points $\{x_i : x \in C_k\}$. To prove this consider the following steps:

1. One-Dimensional Case:

Let $\{x_1, x_2, \dots, x_n\}$ be the set of points in one dimension for the i -th feature of the cluster C_k . The objective function becomes:

$$f(c_{k,i}) = \sum_{j=1}^n |c_{k,i} - x_j|$$

2. Analyze the Behavior of the Objective Function:

The function $f(c_{k,i})$ measures the total distance between $c_{k,i}$ and all points in the set. To understand the behavior of $f(c_{k,i})$, consider its derivative.

3. Left and Right Derivatives:

The absolute value function $|c_{k,i} - x_j|$ is piecewise linear. This makes $f(c_{k,i})$ piecewise linear and continuous, with a slope that changes only at the data points x_j .

1. Derivative to the Left of a Point x_j :

$$\frac{\partial}{\partial c_{k,i}} |c_{k,i} - x_j| = -1 \quad \text{if } c_{k,i} < x_j$$

2. Derivative to the Right of a Point x_j :

$$\frac{\partial}{\partial c_{k,i}} |c_{k,i} - x_j| = +1 \quad \text{if } c_{k,i} > x_j$$

3. At $c_{k,i} = x_j$: The derivative is undefined, but the function is continuous.

The total derivative is therefore the sum of -1 for points to the left of $c_{k,i}$ and $+1$ for points to the right of $c_{k,i}$:

$$\frac{\partial f(c_{k,i})}{\partial c_{k,i}} = (\text{number of points to the right of } c_{k,i}) - (\text{number of points to the left of } c_{k,i}).$$

4. Minimizing the Objective Function:

1. To minimize $f(c_{k,i})$, the derivative must equal zero, or equivalently:

Number of points to the left of $c_{k,i}$ = Number of points to the right of $c_{k,i}$.

2. This condition is satisfied when $c_{k,i}$ is the **median** of $\{x_1, x_2, \dots, x_n\}$: - If n is odd, the median is the middle value of the sorted points. - If n is even, any value between the two middle points minimizes the objective.

5. Generalize to All Dimensions:

The same reasoning applies independently to all p dimensions. For each dimension $i = 1, 2, \dots, p$, the optimal $c_{k,i}$ is the median of the i -th components of the points in C_k .

Thus, the optimal centroid c_k is:

$$c_k = (\text{median of } x_1\text{-values}, \text{median of } x_2\text{-values}, \dots, \text{median of } x_p\text{-values}).$$

Conclusion:

By setting the objective to minimize the sum of Manhattan distances, the optimal centroid for each cluster is the **median** of the points in the cluster along each dimension.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

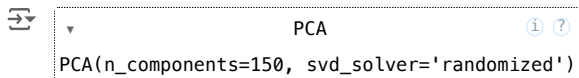
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)

images = faces.images
data = faces.data

from sklearn.decomposition import PCA
```

- Q3)a Perform PCA on the dataset to find the first 150 components. Since this is a large dataset,
- ✓ you should use randomized PCA instead, which can also be found on sklearn. Show the eigenfaces associated with the first 1 through 25 principal components

```
pca = PCA(n_components=150, svd_solver='randomized')
pca.fit(data)
```



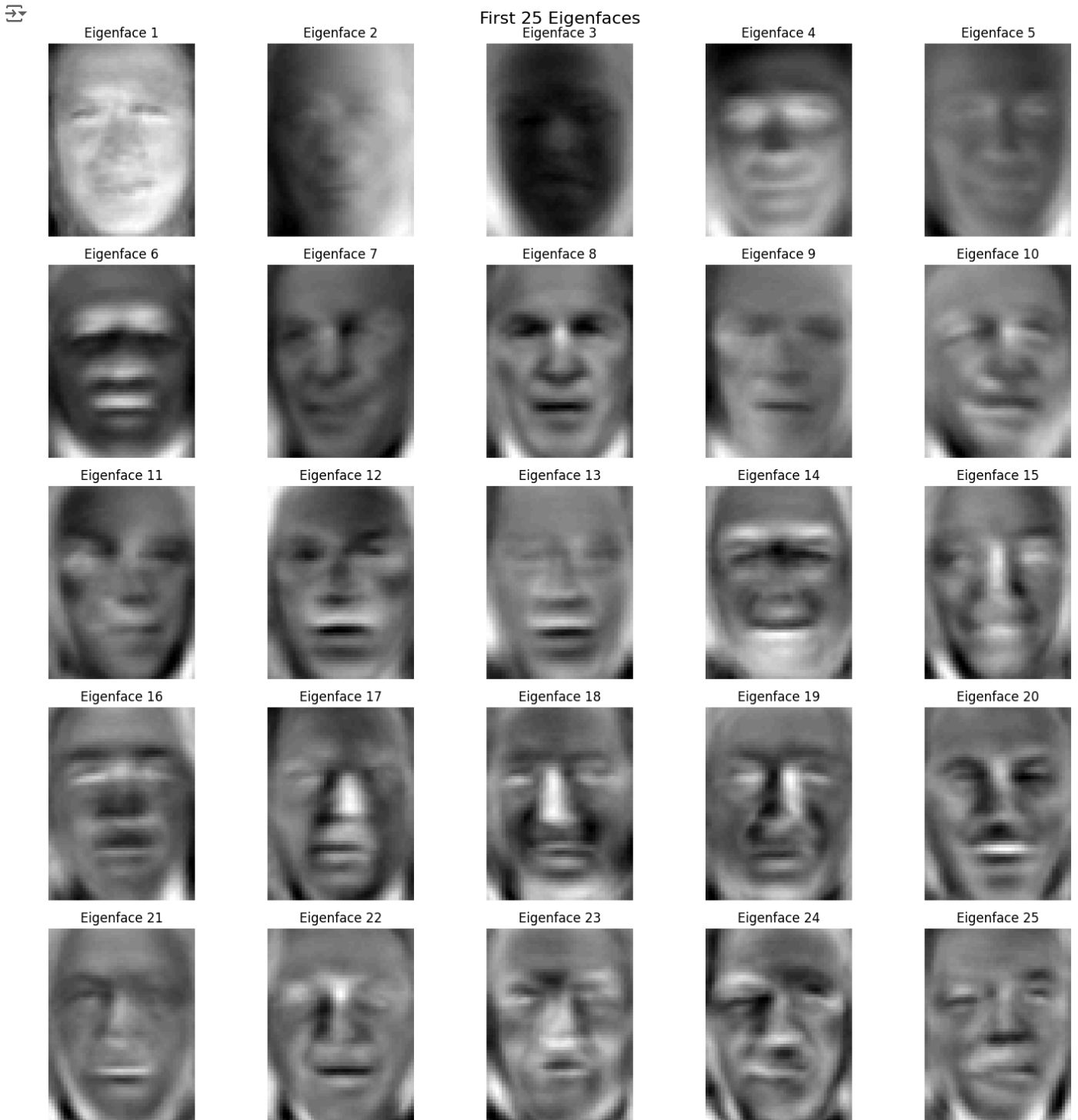
```
PCA(n_components=150, svd_solver='randomized')
```

```
eigen_faces = pca.components_[:25]

fig, axes = plt.subplots(5, 5, figsize=(15, 15))
fig.suptitle('First 25 Eigenfaces', fontsize=16)

for i, ax in enumerate(axes.ravel()):
    eigen_face = eigen_faces[i].reshape(faces.images[0].shape)
    ax.imshow(eigen_face, cmap='gray')
    ax.axis('off')
    ax.set_title(f'Eigenface {i+1}')

plt.tight_layout()
plt.show()
```

Q3)b Using the first 150 components you found, reconstruct a few faces of your choice and compare them with the original input images.

```
low_dim = pca.transform(data)
recon = pca.inverse_transform(low_dim)

n = 3
ind = np.random.choice(range(len(images)), n, replace=False)

fig, axes = plt.subplots(n, 2, figsize=(6, 8))
fig.suptitle('Original vs Reconstructed Faces', fontsize=12)
```

```

for i, idx in enumerate(ind):
    axes[i, 0].imshow(images[idx], cmap='gray')
    axes[i, 0].set_title(f'Original {idx}')
    axes[i, 0].axis('off')

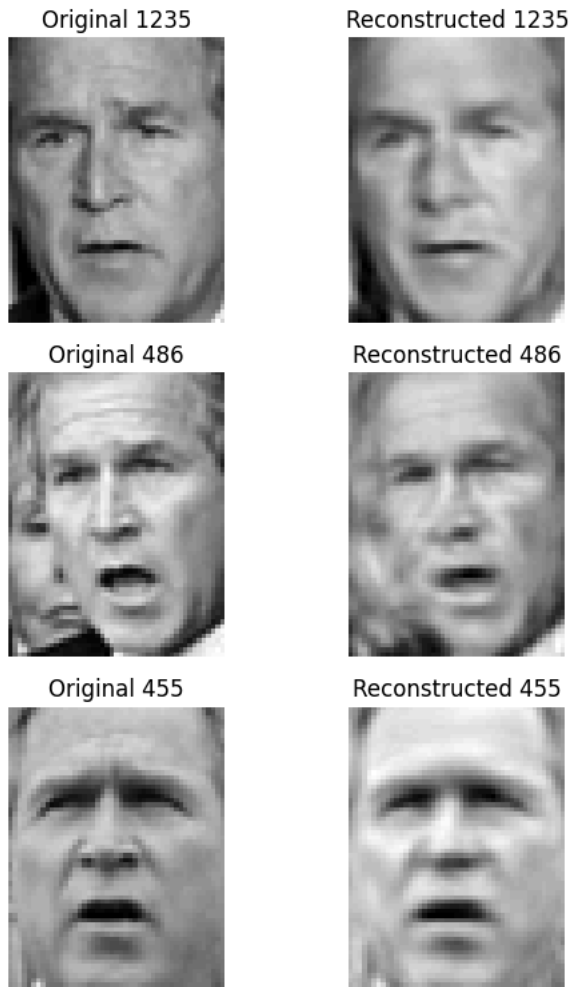
    reconstructed_image = recon[idx].reshape(images[idx].shape)
    axes[i, 1].imshow(reconstructed_image, cmap='gray')
    axes[i, 1].set_title(f'Reconstructed {idx}')
    axes[i, 1].axis('off')

plt.tight_layout()
plt.show()

```



Original vs Reconstructed Faces



- ✓ Hence we have performed PCA by reconstructing on 150 components and compared their transform with respect to original image