Department of Electrical Engineering

Indian Institute of Technology Kharagpur

Machine Learning for Signal Processing Laboratory

(EE69210)

Spring, 2022-23

Experiment 1: Dimensionality Reduction

Grading Rubric

	Tick the best applicable per row			Points
	Below Expectation	Lacking in Some	Meets all Expectation	
Completeness of the report				
Organization of the report (5 pts)				
Quality of figures (5 pts)				
PCA Dataset 1 experiment concepts (20 pts)				
PCA Dataset 2 experiment concepts (25 pts)				
SVD Dataset 1 experiment concepts (20 pts)				
SVD Dataset 2 experiment concepts (25 pts)				
			TOTAL (100 pts)	

1 Code of Conduct

Students are expected to behave ethically both in and out of the lab. Unethical behaviour includes, but is not limited to, the following:

- Possession of another person's laboratory solutions from the current or previous years.
- Reference to, or use of another person's laboratory solutions from the current or previous years.
- Submission of work that is not done by your laboratory group.
- Allowing another person to copy your laboratory solutions or work.
- Cheating on quizzes.

The rules of laboratory ethics are designed to facilitate these goals. We emphasize that laboratory TAs are available to help the student both understand the basic concepts and answer the questions being asked in the laboratory exercises. By performing the laboratories independently, students will likely learn more and improve their performance in the course as a whole. Please note that it is the responsibility of the student to make sure that the content of their graded laboratories is not distributed to other students. If there is any question as to whether a given action might be considered unethical, please see the professor or the TA before you engage in such actions.

2 Principal Component Analysis

Principal component analysis (PCA) also known as the Kosambi-Karhunen-Loeve transform, is one of the most popular methods for dimensionality reduction in pattern recognition.

Let $\mathbf{x}_i$ be a column vector of dimension $D \times 1$ denoting a sample of signal, consisting of real valued numbers such that it is represented as $\mathbf{x}_i \in \mathbb{R}^{D\times 1}$.

$$\mathbf{x}_i = \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(D-1)} \end{bmatrix} \tag{1}$$

In case of mono-phonic audio signals we have $\mathbf{x}_i \in \mathbb{R}^{1\times 1}$, while for stereophonic signals we have $\mathbf{x}_i \in \mathbb{R}^{2\times 1}$. Similarly in case of a RGB color image represented as $\mathcal{I} \in \mathbb{R}^{\alpha\times\beta\times 3}$ we would have $\mathbf{x}_i \in \mathbb{R}^{3\alpha\beta\times 1}$.

We further represent the set of all such N samples as a matrix $\mathbf{X}$ formed by stacking all the $\mathbf{x}_i$ in a row-wise manner such that $\mathbf{X} \in \mathbb{R}^{D\times N}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0, & \mathbf{x}_1, & \cdots, & \mathbf{x}_i, & \cdots, & \mathbf{x}_{N-1} \end{bmatrix} \tag{2}$$

$$\mathbf{X} = \begin{bmatrix} \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(D-1)} \end{bmatrix}_0, & \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(D-1)} \end{bmatrix}_1, & \cdots, & \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(D-1)} \end{bmatrix}_i, & \cdots, & \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ \vdots \\ x^{(D-1)} \end{bmatrix}_{N-1} \end{bmatrix} \tag{3}$$

The task of dimensionality reduction involves representing the sample $\mathbf{x}_i$ to an alternate form with lower dimension viz. $\hat{\mathbf{y}}_i \in \mathbb{R}^{m\times 1}$ where $m \leq D$. The dataset $\mathbf{X}$ can thus be represented in an alternate form with reduced dimension as $\hat{\mathbf{Y}} \in \mathbb{R}^{m\times N}$.

2.1 Orthogonalization

We begin the process with representing $\mathbf{x}_i$ in an alternate form $\mathbf{y}_i$, such that its components are projections of $\mathbf{x}_i$ on to a set of orthonormal bases,

$$\mathbf{y}_i = \mathbf{A}^T \mathbf{x}_i \tag{4}$$

where $\mathbf{y}_i = [y_i^{(0)}, y_i^{(1)}, \cdots, y_i^{(j)}, \cdots, y_i^{(D-1)}]^T \in \mathbb{R}^{D\times 1}$ and $\mathbf{A} \in \mathbb{R}^{D\times D}$ represents the matrix constituting a set of orthonormal vectors along which $\mathbf{x}_i$ is projected. We can further represent the dataset $\mathbf{X}$ through this transformation to an alternate representation $\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{A}^T \mathbf{X} \tag{5}$$

where $\mathbf{Y} \in \mathbb{R}^{D\times N}$.

2.1.1 Necessary condition

In order to achieve this desired solution, $\mathbf{X}$ requires to have a form such that $\mathbb{E}[\mathbf{x}_i] = \mathbf{0}$, which also implies that $\mathbb{E}[\mathbf{y}_i] = \mathbf{0}$.

2.1.2 Solution

The matrix $\mathbf{A}$ is chosen such that its columns are orthonormal eigenvectors $\mathbf{a}_j \in \mathbb{R}^{D\times 1}$, $j = 0, 1, \cdots, D-1$, and $\mathbf{a}_j = [a^{(0)}, a^{(1)}, \cdots, a^{(D-1)}]^T$, so as to form

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_0, & \mathbf{a}_1, & \cdots & \mathbf{a}_j, & \cdots & \mathbf{a}_{D-1} \end{bmatrix} \tag{6}$$

$$\mathbf{A} = \left[\begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_0, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_1, \cdots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_j \cdots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_{D-1} \right] \tag{7}$$

The constituents $\mathbf{a}_j$ and their corresponding eigenvalues λ_j are obtained by eigenvalue decomposition (EVD) of the sample covariance matrix of $\mathbf{X}$ represented as $\mathbf{R}_X \in \mathbb{R}^{D\times D}$ which is estimated as

$$\mathbf{R}_X \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \tag{8}$$

Further from the definition of correlation matrix we have

$$\mathbf{R}_Y = \mathbb{E}\left[\mathbf{Y}\mathbf{Y}^T\right] = \mathbb{E}\left[\mathbf{A}^T \mathbf{X}\mathbf{X}^T \mathbf{A}\right] = \mathbf{A}^T \mathbf{R}_X \mathbf{A} \tag{9}$$

where $\mathbf{R}_Y \in \mathbb{R}^{D\times D}$.

2.2 Energy compaction

The constituents of $\mathbf{A}$ are arranged according to a descending order of their corresponding eigenvalues viz. ${}_0 \geq{}_1 \geq \cdots{}_j \geq \cdots{}_{D-1}$. In this arrangement, the components of $\mathbf{y}_i = [y_i^{(0)}, y_i^{(1)}, \cdots, y_i^{(j)}, \cdots, y_i^{(D-1)}]^T$ correspond to projection of $\mathbf{x}_i$ on each of the eigenvectors $\mathbf{a}_j$ according to their dominance.

$$y_i^{(j)} = \mathbf{a}_j^T \mathbf{x}_i \tag{10}$$

We can consider to represent $\mathbf{y}_i$ as a reduced dimensional vector $\hat{\mathbf{y}}_i \in \mathbb{R}^{m\times 1}$ with $m \leq D$. Thus the reduced dimensional equivalent of the dataset $\mathbf{X}$ can be represented as

$$\hat{\mathbf{Y}} = \hat{\mathbf{A}}^T\mathbf{X} \tag{11}$$

where we have $\hat{\mathbf{A}} \in \mathbb{R}^{D\times m}$ such that

$$\hat{\mathbf{A}} = \left[\begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_0, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_1, \cdots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_j \cdots, \begin{bmatrix} a^{(0)} \\ a^{(1)} \\ \vdots \\ a^{(D-1)} \end{bmatrix}_{m-1}\right] \tag{12}$$

Accordingly, we would also have

$$\hat{\mathbf{Y}} = \left[\begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_0, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_1, \cdots, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_i, \cdots, \begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m-1)} \end{bmatrix}_{N-1}\right] \tag{13}$$

We can now recover the sample $\mathbf{x}_i$ through its corresponding reduced dimension form $\hat{\mathbf{y}}_i = [y_i^{(0)}, y_i^{(1)}, \cdots, y_i^{(j)}, \cdots, y_i^{(m-1)}]^T$ accordingly

$$\hat{\mathbf{x}}_i = \sum_{j=0}^{m-1} y_i^{(j)}\mathbf{a}_j \tag{14}$$

where $\hat{\mathbf{x}}_i = [\hat{x}_i^{(0)}, \hat{x}_i^{(1)}, \cdots, \hat{x}_i^{(j)}, \cdots, \hat{x}_i^{(D-1)}]^T \in \mathbb{R}^{D\times 1}$. This representation has the energy of sample $\mathbf{x}_i$ compacted along the eigenvectors corresponding to the m dominant eigenvalues, and this recovered dataset can be represented as

$$\hat{\mathbf{X}} = \begin{bmatrix}\hat{\mathbf{x}}_0, & \hat{\mathbf{x}}_1, & \cdots, & \hat{\mathbf{x}}_i, & \cdots, & \hat{\mathbf{x}}_{N-1}\end{bmatrix} \tag{15}$$

$$\hat{\mathbf{X}} = \left[\begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_0, \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_1, \cdots, \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_i, \cdots, \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_{N-1}\right] \tag{16}$$

2.3 Estimating energy compaction

Let ϵ_m be the error between $\mathbf{x}_i$ and its recovered version $\hat{\mathbf{x}}_i$ obtained from $\hat{\mathbf{y}}_i$ consisting of representation along m dominant eigenvectors.

$$\epsilon_m = \mathbb{E}\left[\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2\right] = \mathbb{E}\left[\left\|\sum_{j=m}^{D-1} y_i^{(j)}\mathbf{a}_j\right\|^2\right] \approx \frac{1}{N}\sum_{i=0}^{N-1}\left(\left\|\sum_{j=m}^{D-1} y_i^{(j)}\mathbf{a}_j\right\|^2\right) \tag{17}$$

$$\epsilon_m = \sum_{j=m}^{D-1} \lambda_j \tag{18}$$

Thus, with $m = 1$ we obtain the highest error due to projection along only the *first* dominant eigenvector, while with $m = D$ we would obtain $\epsilon_m = 0$.

3 Experiments on PCA

3.1 Dataset 1

3.1.1 Generation of the dataset

Let us first generate a sample dataset $\mathbf{X}^* \in \mathbb{R}^{3\times 100}$ as below

$$\mathbf{X}^* = \left[\begin{bmatrix} x^{*(0)} \\ x^{*(1)} \\ x^{*(2)} \end{bmatrix}_0 , \begin{bmatrix} x^{*(0)} \\ x^{*(1)} \\ x^{*(2)} \end{bmatrix}_1 , \cdots , \begin{bmatrix} x^{*(0)} \\ x^{*(1)} \\ x^{*(2)} \end{bmatrix}_i , \cdots , \begin{bmatrix} x^{*(0)} \\ x^{*(1)} \\ x^{*(2)} \end{bmatrix}_{99} \right] = \begin{bmatrix} \mathbf{x}^{*(0)} \\ \mathbf{x}^{*(1)} \\ \mathbf{x}^{*(2)} \end{bmatrix} \tag{19}$$

Here we employ three row vectors $\mathbf{x}^{*(0)}$, $\mathbf{x}^{*(1)}$ and $\mathbf{x}^{*(2)}$ to achieve it.

$$\begin{aligned} \mathbf{x}^{*(0)} &= \left[x_0^{*(0)}, \quad x_1^{*(0)}, \quad \cdots, \quad x_i^{*(0)}, \quad \cdots, \quad x_{99}^{*(0)} \right] \\ \mathbf{x}^{*(1)} &= \left[x_0^{*(1)}, \quad x_1^{*(1)}, \quad \cdots, \quad x_i^{*(1)}, \quad \cdots, \quad x_{99}^{*(1)} \right] \\ \mathbf{x}^{*(2)} &= \left[x_0^{*(2)}, \quad x_1^{*(2)}, \quad \cdots, \quad x_i^{*(2)}, \quad \cdots, \quad x_{99}^{*(2)} \right] \end{aligned} \tag{20}$$

$x_i^{*(0)}$ is randomly drawn from a uniform distribution of numbers in the range $[0, 100]$, such that

$$x_i^{*(0)} \sim U[0, 100] \tag{21}$$

$$x_i^{*(1)} = x_i^{*(0)} + \epsilon_i \ ; \ \epsilon_i \sim U[-10, 10] \tag{22}$$

$$x_i^{*(2)} = x_i^{*(0)} + \phi_i \ ; \ \phi_i \sim U[-1, 1] \tag{23}$$

We can generate the dataset following (19) using Listing 1. The generated dataset is illustrated in Fig. 1.

```python
import numpy as np
import matplotlib.pyplot as plt

# Generating Data point vectors x_0, x_1 and x_2 for dataset X.
x_ast0 = np.random.uniform(0,100,100)
x_ast1 = x_ast0 + np.random.uniform(-10,10,100)
x_ast2 = x_ast0 + np.random.uniform(-1,1,100)

#Plotting dataset points in 3D scatter plot
ax = plt.axes(projection='3d')
ax.scatter(x_ast0, x_ast1, x_ast2, linewidth=0.5);
ax.set_xlabel('$x^{*(0)}$')
ax.set_ylabel('$x^{*(1)}$')
ax.set_zlabel('$x^{*(2)}$')

#Stacking the x_ast0, x_ast1 and x_ast2 in X
X_ast = np.stack((x_ast0,x_ast1,x_ast2))
```

Listing 1: Code to generate the dataset

3.1.2 Preparation of the dataset for compliance

It being required that the dataset should follow zero-mean according to §2.1.1, such that $\mathbb{E}[\mathbf{x}_i] = \mathbf{0}$, we need to prepare them as

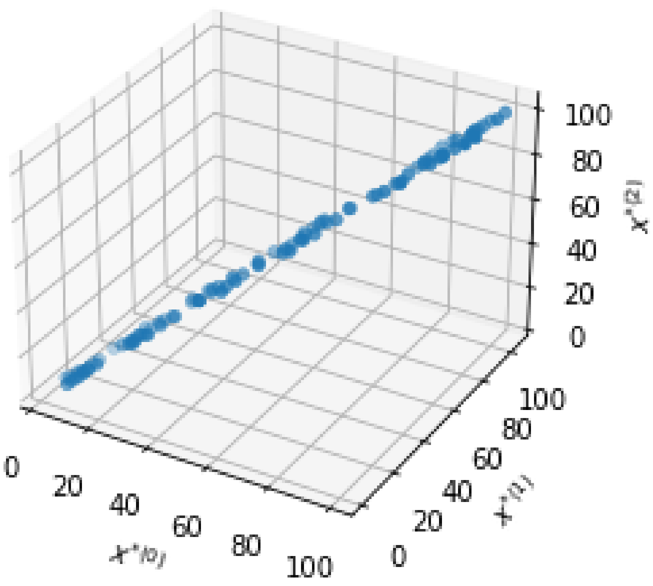

Figure 1: Point cloud illustration of dataset $\mathbf{X}^* \in \mathbb{R}^{3\times100}$ where a sample $\mathbf{x}^*{}_i \in \mathbb{R}^{3\times1}$ generated following (19) and (20).

$$\mathbf{x}_i = \mathbf{x}_i^* - \mu \tag{24}$$

$$\mu = \mathbb{E}[\mathbf{x}_i^*] \approx \frac{1}{N}\sum_{i=0}^{N-1} \mathbf{x}_i^* \tag{25}$$

which can be obtained using Listing 2 as is illustrated in Fig. 2.

```
#Zero centering the data
mu = np.mean(X_ast,axis=1)
X= np.zeros(X_ast.shape)
i = 0
for x_1 in X_ast.transpose():
    X[:,i] = x_1 - mu
    i=i+1
ax = plt.axes(projection='3d')
ax.scatter(X[0,:], X[1,:], X[2,:], linewidth=0.5);

ax.set_xlabel('$x^{(0)}$')
ax.set_ylabel('$x^{(1)}$')
ax.set_zlabel('$x^{(2)}$')
```

Listing 2: Code for zero centering of the dataset

3.1.3 Computing eigenvectors

The sample covariance matrix $\mathbf{R}_X$ in (8) can be computed using Listing 3.

```
#Finding Correlation matrix
R_X = np.dot(X,X.transpose())/X.shape[1]
```

Listing 3: Code for estimating the sample covariance matrix

The eigenvectors constituting $\mathbf{A}$ in (7) can be obtained using Listing 4.

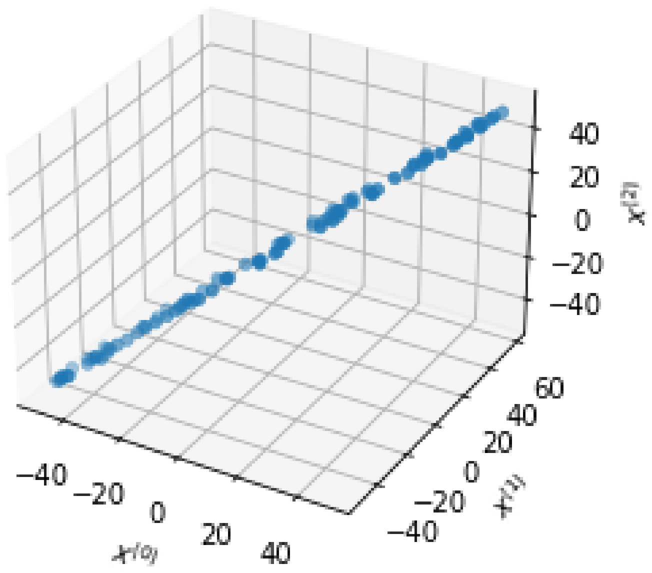

Figure 2: Point cloud illustration of zero centered dataset $\mathbf{X} \in \mathbb{R}^{3\times100}$ where a sample $\mathbf{x}_i \in \mathbb{R}^{3\times1}$ generated following (24)

```python
#Finding eigenvectors and correspondinng eigenvalues

from numpy.linalg import eig
lmd,A= eig(R_X)
```

Listing 4: Code for estimating the eigenvectors and eigenvalues of the sample covariance matrix

On computing the eigenvectors, we can now visualize them along with the dataset using Listing 5 as is illustrated in Fig. 3.

```python
#Plotting eigenvectors along with the samples
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        FancyArrowPatch.__init__(self, (0,0), (0,0), *args, **kwargs)
        self._verts3d = xs, ys, zs

    def draw(self, renderer):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
        self.set_positions((xs[0],ys[0]),(xs[1],ys[1]))
        FancyArrowPatch.draw(self, renderer)

ax = plt.axes(projection='3d')
ax.scatter(X[0,:], X[1,:], X[2,:], linewidth=0.5);
ax.set_xlabel('$x^{(0)}$')
ax.set_ylabel('$x^{(1)}$')
ax.set_zlabel('$x^{(2)}$')

it=0
str = ['red','green','blue']
for a1 in A.transpose():
    a = Arrow3D([0,((a1[0]*lmd[it])/20)],[0,((a1[1]*lmd[it])/20)],[0,((a1[2]*
   lmd[it])/20)], arrowstyle="-|>",mutation_scale=20,color=str[it])
```

```
    ax.add_artist(a)
    it = it + 1
```

Listing 5: Code for plotting the eigenvectors

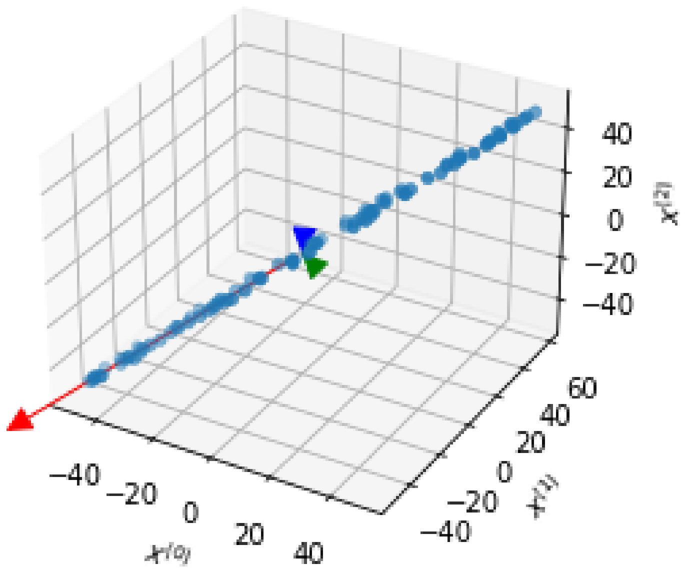

Figure 3: Eigenvalue weighted eigenvectors along with point cloud of dataset $\mathbf{X} \in \mathbb{R}^{3\times 100}$ where a sample $\mathbf{x}_i \in \mathbb{R}^{3\times 1}$.

3.1.4 Assignments to solve and report

Write down the codes for the following tasks in a .ipynb file, including all visualizations and submit the executed file. Submit also a separate pdf of your report of this experiment written in Latex using the documentclass *article*. This report should describe your observations and reasoning while executing these experiments.

1. Rerun all the steps discussed till now, rerunning all the codes. Check your consistency in being able to generate plots similar to those presented earlier.

2. Project each sample $\mathbf{x}_i$ on to the eigenvectors to obtain each component of $\mathbf{y}_i$, and plot projection of each sample separately. The plot would be similar to Figs. 4, 5 and 6.

3. Numerically estimate $\mathbf{R}_Y$ and verify it per (9).

4. Recover the samples $\hat{\mathbf{x}}_i$ from the reduced dimensional representation $\hat{\mathbf{y}}_i$ with $m = 1$ and $m = 2$ according to (14). Plot the recovered samples. These would appear similar to Figs. 7 and 8.

5. Plot the recovered samples $\hat{\mathbf{x}}_i$ in the domain of $\mathbf{X}^*$ for $m = 1$ and $m = 2$ by undoing the effect of (24). These would appear similar to Figs. 9 and 10.

6. Plot the eigenvalue weighted eigenvectors used for recovering the samples $\hat{\mathbf{x}}_i$ for $m = 1$ and $m = 2$ in the domain of $\mathbf{X}^*$. These would appear similar to Figs. 11 and 12.

7. Calculate the mean squared error (MSE) in the recovered signal using (17) for $m = 1$ and $m = 2$.

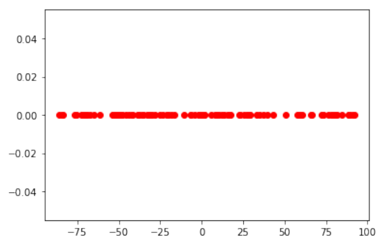

Figure 4: Projection of $\mathbf{X}$ on $\mathbf{a}_0$.

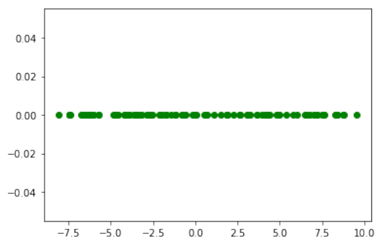

Figure 5: Projection of $\mathbf{X}$ on $\mathbf{a}_1$.

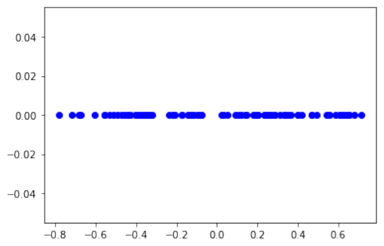

Figure 6: Projection of $\mathbf{X}$ on $\mathbf{a}_2$.

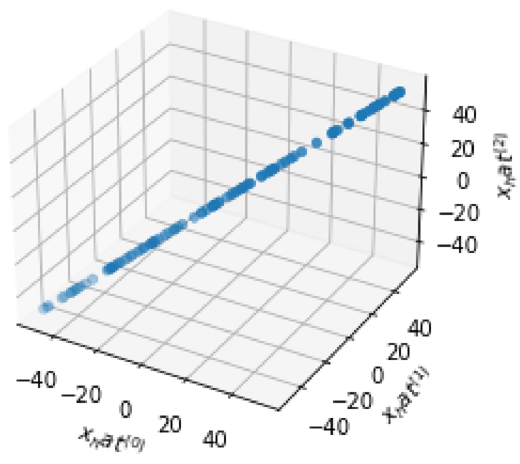

Figure 7: Recovered $\hat{\mathbf{X}}$ for $m = 1$

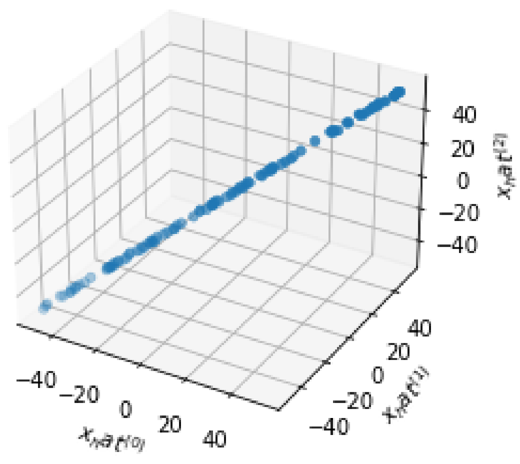

Figure 8: Recovered $\hat{\mathbf{X}}$ for $m = 2$

3.2 Dataset 2

3.2.1 Rearranging the matrix to form the dataset

Here we would strive to solve PCA on grayscale images of human faces of size 64×64, from the Olivetti faces dataset[1] with $N = 400$ samples. Since each sample image $\mathcal{I}_i \in \mathbb{Z}^{64\times64\times1}$, so we can rearrange it to form $\mathbf{x}_i^* \in \mathbb{Z}^{4,096\times1}$. The dataset can be accessed and rearranged according using Listing 6.

```python
from sklearn.datasets import fetch_olivetti_faces
from sklearn import cluster
from sklearn import decomposition

rng = RandomState(0)

faces, _ = fetch_olivetti_faces(return_X_y=True, shuffle=True, random_state=rng
    )
n_samples, n_features = faces.shape

# Zero centering of sample images
faces_centered = faces - faces.mean(axis=0)

#Dataset
X = faces_centered.transpose()
```

Listing 6: Code for dataset creation for Olivetti face dataset

[1]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_olivetti_faces.html

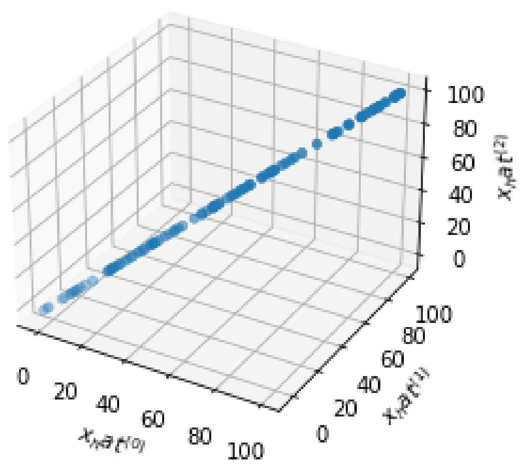
Figure 9: Recovered $\hat{\mathbf{X}}^*$ for $m = 1$

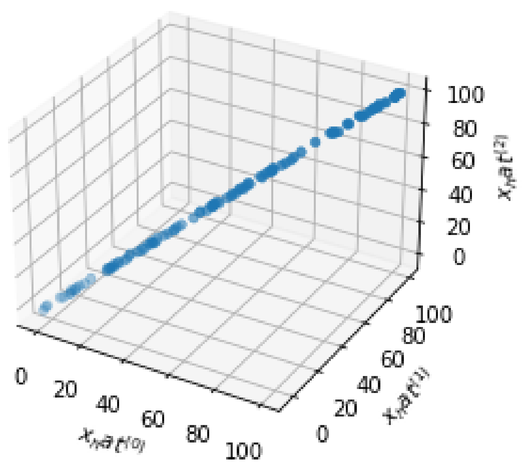
Figure 10: Recovered $\hat{\mathbf{X}}^*$ for $m = 2$

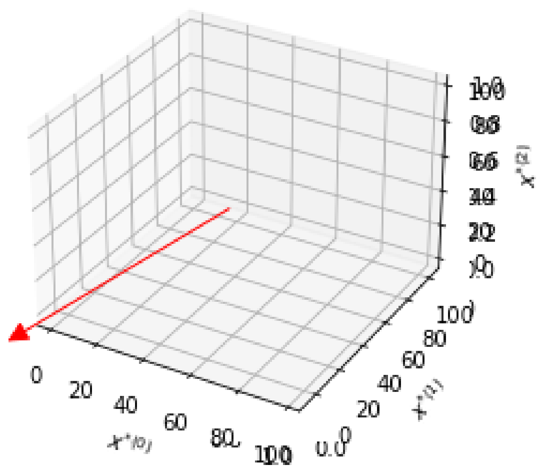
Figure 11: Eigenvalue weighted eigenvector for $m = 1$

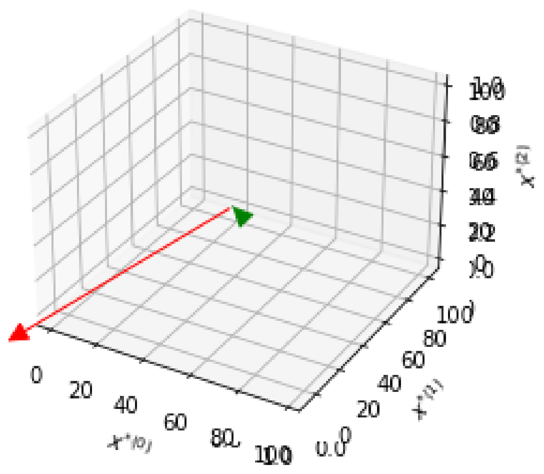
Figure 12: Eigenvalue weighted eigenvector for $m = 2$

3.2.2 Preparation of the dataset for compliance

Since $\mathbf{x}_i^* \in \mathbb{Z}^{4,096 \times 1}$, we need to prepare it following §2.1.1, such that $\mathbb{E}[\mathbf{x}_i] = \mathbf{0}$,

$$\mathbf{x}_i = \mathbf{x}_i^* - \mu \tag{26}$$
$$\mu = \mathbb{E}[\mathbf{x}_i^*] \tag{27}$$

3.2.3 Assignments to solve and report

Write down the codes for the following tasks in .ipynb file, including visualizations, submit the executed file. Submit a separate pdf describing your observations and reasoning while executing these experiments.

1. Estimate $\mathbf{R}_X$, and compute the matrix $\mathbf{A}$.
2. Visualize the top-5 eigenvectors of $\mathbf{A}$ by scaling and reshaping it in the domain of $\mathcal{I}_i \in \mathbb{Z}^{64 \times 64}$. These would appear similar to Fig. 13.

3. Find the number of components m required to recover $\hat{\mathbf{X}}$ such that $\epsilon_m \leq 5\%$. Describe your logic to obtain the solution.
4. Pick an image randomly from the dataset and represent its recovered form obtained by varying m such that the error in recovery is within 1%, 5%, 10%, 20%. Calculate the MSE between the original image and the recovered image. Describe all the steps involved along with the relevant mathematical steps involved. These would appear similar to Fig. 14.

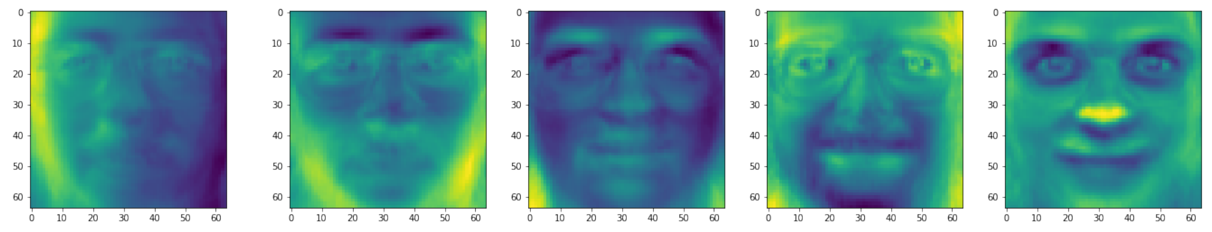

Figure 13: Eigenfaces corresponding to the top-5 eigenvectors

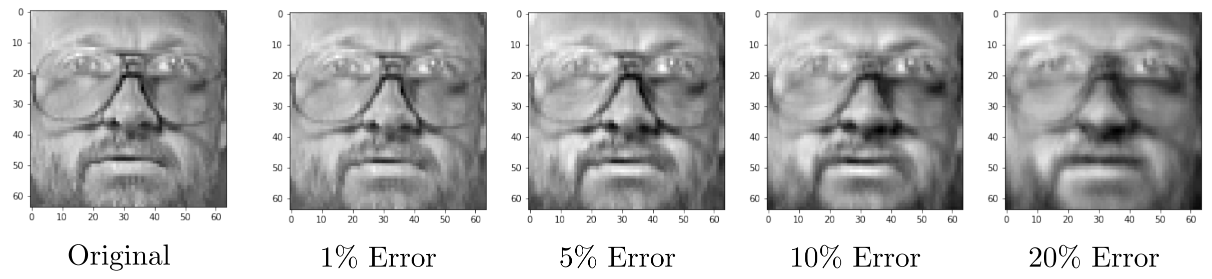

Figure 14: Original sample image and recovered images for 1%,5%,10% and 20% error.

4 Singular Value Decomposition

Singular value decomposition (SVD) is extensively used for rank and dimension reduction for a matrix, and is of relevance to pattern recognition an information retrieval problems. Let us consider having real valued signal where each sample is represented as $\mathbf{x}_i \in \mathbb{R}^{D\times 1}$ organized similar to (1), such that the dataset with N sampls is represented as $\mathbf{X} \in \mathbb{R}^{D\times N}$ similar to (3). We can now further consider representing it as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{28}$$

where $\mathbf{U} \in \mathbb{R}^{D\times D}$, $\mathbf{\Sigma} \in \mathbb{R}^{D\times N}$, and $\mathbf{V} \in \mathbb{R}^{N\times N}$. If $\mathbf{X}$ is of a lower rank $r \leq \min(D, N)$, then we can further write

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Lambda}^{\frac{1}{2}} & \mathbf{0}_a \\ \mathbf{0}_b & \mathbf{0}_c \end{bmatrix} \tag{29}$$

where $\mathbf{0}_a \in 0^{r\times(N-r)}$, $\mathbf{0}_b \in 0^{(D-r)\times r}$, and $\mathbf{0}_c \in 0^{(D-r)\times(N-r)}$ are respectively zero matrices of the specified dimension. Here $\mathbf{\Lambda}^{\frac{1}{2}} \in \mathbb{R}^{r\times r}$ is a diagonal matrix made up of $\sqrt{\lambda_j}$ where $\{\lambda_j\}$ is the set of r non-zero eigenvalues of $\mathbf{X}^H\mathbf{X}$. (28) can be simplified further and re-written as

$$\mathbf{X} = \mathbf{U}_r\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}_r^H \tag{30}$$

$$\mathbf{\Lambda}^{\frac{1}{2}} = \begin{bmatrix} \sqrt{\lambda_0} & 0 & \cdots & 0 \\ 0 & \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\lambda_{r-1}} \end{bmatrix} \tag{31}$$

where $\mathbf{U}_r \in \mathbb{R}^{D \times r}$, $\mathbf{V}_r \in \mathbb{R}^{N \times r}$, such that

$$\mathbf{U}_r = \begin{bmatrix} \mathbf{u}_0, & \mathbf{u}_1, & \cdots, & \mathbf{u}_j, & \cdots, & \mathbf{u}_{r-1} \end{bmatrix} \tag{32}$$

$$\mathbf{U}_r = \begin{bmatrix} \begin{bmatrix} u^{(0)} \\ u^{(1)} \\ \vdots \\ u^{(D-1)} \end{bmatrix}_0, & \begin{bmatrix} u^{(0)} \\ u^{(1)} \\ \vdots \\ u^{(D-1)} \end{bmatrix}_1, & \cdots, & \begin{bmatrix} u^{(0)} \\ u^{(1)} \\ \vdots \\ u^{(D-1)} \end{bmatrix}_j, & \cdots, & \begin{bmatrix} u^{(0)} \\ u^{(1)} \\ \vdots \\ u^{(D-1)} \end{bmatrix}_{r-1} \end{bmatrix} \tag{33}$$

where $\mathbf{u}_j \in \mathbb{R}^{D \times 1}$ denotes the eigenvector corresponding to the non-zero eigenvalue of $\mathbf{X}\mathbf{X}^H$. Similarly we have

$$\mathbf{V}_r = \begin{bmatrix} \mathbf{v}_0, & \mathbf{v}_1, & \cdots, & \mathbf{v}_j, & \cdots, & \mathbf{v}_{r-1} \end{bmatrix} \tag{34}$$

$$\mathbf{V}_r = \begin{bmatrix} \begin{bmatrix} v^{(0)} \\ v^{(1)} \\ \vdots \\ v^{(N-1)} \end{bmatrix}_0, & \begin{bmatrix} v^{(0)} \\ v^{(1)} \\ \vdots \\ v^{(N-1)} \end{bmatrix}_1, & \cdots, & \begin{bmatrix} v^{(0)} \\ v^{(1)} \\ \vdots \\ v^{(N-1)} \end{bmatrix}_j, & \cdots, & \begin{bmatrix} v^{(0)} \\ v^{(1)} \\ \vdots \\ v^{(N-1)} \end{bmatrix}_{r-1} \end{bmatrix} \tag{35}$$

where $\mathbf{v}_j \in \mathbb{R}^{N \times 1}$ denotes the eigenvector corresponding to the non-zero eigenvalue of $\mathbf{X}^H\mathbf{X}$.

4.1 Low rank approximation

Following up from (30), we can write that

$$\mathbf{X} = \sum_{j=0}^{r-1} \sqrt{\lambda_j}\mathbf{u}_j\mathbf{v}_j^H \tag{36}$$

If we consider using $k \leq r$ number of terms in the summation, then $\mathbf{X}$ can be approximated and recovered as $\hat{\mathbf{X}}$

$$\hat{\mathbf{X}} = \sum_{j=0}^{k-1} \sqrt{\lambda_j}\mathbf{u}_j\mathbf{v}_j^H \tag{37}$$

$$\hat{\mathbf{X}} = \begin{bmatrix} \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_0, & \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_1, & \cdots, & \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_i, & \cdots, & \begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_{N-1} \end{bmatrix} \tag{38}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{D \times N}$, is the sum of $k \leq r$ rank-one independent matrices, is of rank k.

4.2 Dimensionality reduction

Following up from (37), employing (31), (32) and (34), we can write

$$\hat{\mathbf{X}} = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{k-1}] \begin{bmatrix} \sqrt{\lambda_0}\mathbf{v}_0^H \\ \sqrt{\lambda_1}\mathbf{v}_1^H \\ \vdots \\ \sqrt{\lambda_{k-1}}\mathbf{v}_{k-1}^H \end{bmatrix} = \mathbf{U}_k \mathbf{A}_k \tag{39}$$

where $\mathbf{U}_k = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{k-1}] \in \mathbb{R}^{D\times k}$ consists of the first k columns of $\mathbf{U}$, and $\mathbf{A}_k = [\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_{N-1}] \in \mathbb{R}^{k\times N}$ consists of the column vectors of the product matrix $\mathbf{\Lambda}_k^{\frac{1}{2}}\mathbf{V}_k^H$, where $\mathbf{V}_k$ consists of the first k columns of $\mathbf{V}$, and $\mathbf{\Lambda}_k^{\frac{1}{2}}$ is the diagonal matrix having the square root of the respective k eigenvalues.

Each sample $\mathbf{x}_i$ can be approximated by a column vector $\hat{\mathbf{x}}_i$ as

$$\hat{\mathbf{x}}_i = \mathbf{U}_k \mathbf{a}_i \tag{40}$$

$$\begin{bmatrix} \hat{x}^{(0)} \\ \hat{x}^{(1)} \\ \vdots \\ \hat{x}^{(D-1)} \end{bmatrix}_i = \sum_{j=0}^{k-1} \mathbf{u}_j a_i^{(j)} \tag{41}$$

where $\mathbf{a}_i = \left[a_i^{(0)}, a_i^{(1)}, \cdots, a_i^{(k-1)}\right]^T \in \mathbb{R}^{k\times 1}$. Simplifying it, the higher D-dimensional sample vector $\mathbf{x}_i$ can be approximated by the k-dimensional vector $\mathbf{a}_i$, lying in the subspace spanned by $\mathbf{u}_j$. This also denotes that $\mathbf{a}_i$ is the projection of $\mathbf{x}_i$ on the subspace spanned by $\mathbf{u}_j$.

Further, due to orthonormality of the columns $\mathbf{u}_j$ of $\mathbf{U}_k$ we see that the distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_m$ where $i, m = 0, 1, \cdots, n-1$ can be represented as

$$\|\mathbf{x}_i - \mathbf{x}_m\| \simeq \|\mathbf{U}_k(\mathbf{a}_i - \mathbf{a}_m)\| = \left\| \sum_{j=0}^{k-1} \mathbf{u}_j \left(a_i^{(j)} - a_m^{(j)}\right) \right\| = \|\mathbf{a}_i - \mathbf{a}_m\| \tag{42}$$

This has an important implication in pattern recognition. In order to search for a pair of similar patters $(\mathbf{x}_i, \mathbf{x}_m)$, instead of computing the distance between a pair of D-dimensional vectors, we can now compute the same value using their corresponding k-dimensional representation $(\mathbf{a}_i, \mathbf{a}_m)$. When $k \ll D$ a substantial computational savings are obtained.

4.3 Error in low rank approximation

If the k largest eigenvalues are involved than it can be seen that the squared error between the signal recovered using k components and the corresponding original signal in the complete dataset can be written as

$$\epsilon_k^2 = \sum_{i=0}^{N-1} \sum_{j=0}^{D-1} \|x_i^{(j)} - \hat{x}_i^{(j)}\|^2 = \sum_{j=k}^{r-1} \lambda_j \tag{43}$$

Thus if we order the eigenvalues in the descending order with $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{r-1}$, then for a given number of k terms in the expansion, the SVD leads to the minimum square error. There $\hat{\mathbf{X}}$ is the best rank-k approximation of $\mathbf{X}$ in the Frobenius norm sense.

5 Experiments on SVD

5.1 Dataset 1

5.1.1 Generation of the dataset

Let us generate a sample dataset $\mathbf{X} \in \mathbb{R}^{3\times 1{,}000}$ as below

$$\mathbf{X} = \left[\begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_0, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_1, \cdots, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_i, \cdots, \begin{bmatrix} x^{(0)} \\ x^{(1)} \\ x^{(2)} \end{bmatrix}_{999} \right] = \begin{bmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} \tag{44}$$

where we generate these components as

$$x_i^{(0)} \sim U[-10, 10] \tag{45}$$

$$x_i^{(1)} \sim U[-10, 10] \tag{46}$$

$$x_i^{(2)} \sim U[-1, 1] \tag{47}$$

5.1.2 Assignments to solve and report

Write down the codes for the following tasks in a .ipynb file, including all visualizations and submit the executed file. Submit also the separate pdf of your report of this experiment. This report should describe your observations and reasoning while executing these experiments.

1. Write the code to generate the dataset.
2. Compute the matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$.
3. Compute the matrices $\mathbf{U}_r$, $\mathbf{\Lambda}$ and $\mathbf{V}_r$.
4. Derive the equation of the subspace spanned by the first two dominant eigenvectors?
5. Plot the projection of the samples in $\mathbf{X}$ on the subspace spanned by the first two dominant eigenvectors. This should appear similar to Fig. 15.
6. Select any 10 random samples from $\mathbf{X}$, and compute the pairwise Euclidean distance matrix $\mathbf{D} \in \mathbb{R}^{10\times 10}$ between these samples.
7. Compute the pairwise Euclidean distance matrix $\hat{\mathbf{D}} \in \mathbb{R}^{10\times 10}$ using the projection $\mathbf{a}_i$ obtained with first two eigenvectors.
8. Find the Frobenius norm of difference between $\mathbf{D}$ and $\hat{\mathbf{D}}$. Relate this with the property obtained in (43) for low rank approximation with SVD of $\mathbf{X}$.

5.2 Dataset 2

5.2.1 Preparation of the dataset

Prepare the dataset with images of faces accordingly to the steps described earlier in §3.2.1. Here we have the dataset $\mathbf{X} \in \mathbb{R}^{4{,}096\times 400}$.

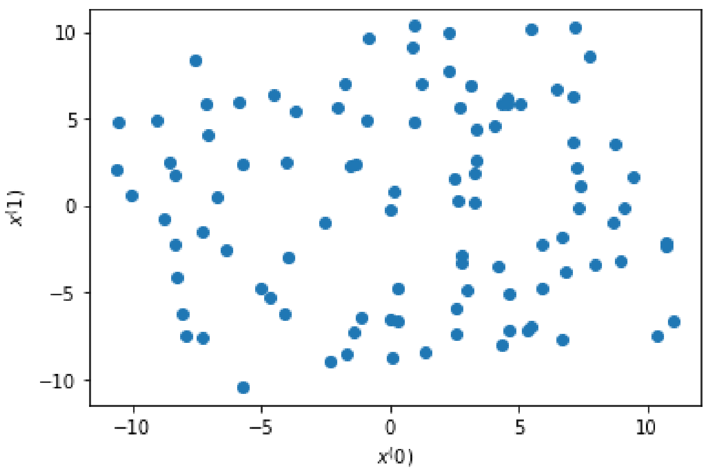

Figure 15: Projection of X on first two dominant Eigen vectors.

5.2.2 Assignments to solve and report

Write down the codes for the following tasks in a .ipynb file, including all visualizations and submit the executed file. Submit also the separate pdf of your report of this experiment. This report should describe your observations and reasoning while executing these experiments.

1. Perform SVD of $\mathbf{X}$ to obtain $\mathbf{U}_r$, $\mathbf{\Lambda}^{\frac{1}{2}}$ and $\mathbf{V}_r$.
2. Plot and visualize the top-5 entries of $\mathbf{U}_r$ sorted according to descending order of their eigenvalues. Compare and contrast them with the top-5 Eigenfaces obtained with PCA.
3. Using (43) find the value of k so that the error in recovery is within 1%, 5%, 10% and 20% when using $\mathbf{U}_k$ and $\mathbf{A}_k$ as in (39).
4. Take 2 sample images from $\mathbf{X}$ and present their low rank approximated form as an image corresponding to $\hat{\mathbf{x}}_i$ for the case of varying k so that error of recovery is within 1%, 5%, 10% and 20%.
5. Measure the Frobenius norm for each of the recovered lowrank approximated images comparing with their original version and comment on the measurements.
6. Take any 5 images randomly selected from $\mathbf{X}$ and search for their closest similar neighbors from the remaining 395 images. Compute the Euclidean distance using $\mathbf{x}_i$ only and measure the computation time requirement. Alternatively employ $\mathbf{a}_i$ with varying k so that error of recovery is within 1%, 5%, 10% and 20%, in order to compute the Euclidean distance and measure the computation time requirement. Plot the time complexity of each of these methods. Comment on the time complexity.
7. Visualize the top-10 closest neighbor images searched for each of the 5 images above, using each of the different approaches of search applied.