SMAI-Miniproject(Part 1 Report)

September 30, 2018

Submitted By: Aditya Agarwal(20161104)

1.1 I. PCA Using Eigenvectors and Covariance Matrix

In this section, a brief procedural description of PCA is provided. Assume that we are given by a m-by-n data matrix X consists of n number of m-dim vectors.

Step 1: Converting Image to Greyscale and Resizing It

My first step was to convert image to greyscale and resize it to 64 * 64 from 256 * 256 as it does not fit into RAM.

Step 2: Normalize the data

Second step is to normalize the data that we have so that PCA works properly. This is done by subtracting the respective means from the numbers in the respective column. So if we have two dimensions X and Y, all X become x- and all Y become y-. This produces a dataset whose mean is zero.

Step 3: Calculate the covariance matrix

Since the dataset we took is 2-dimensional, this will result in a 2x2 Covariance matrix.

Please note that Var[X1] = Cov[X1,X1] and Var[X2] = Cov[X2,X2].

Step 4: Calculate the eigenvalues and eigenvectors

Next step is to calculate the eigenvalues and eigenvectors for the covariance matrix. The same is possible because it is a square matrix. λ is an eigenvalue for a matrix $\boldsymbol{A}$ if it is a solution of the characteristic equation:

$$det(\lambda I - A) = 0$$

Where, ***I*** is the identity matrix of the same dimension as ***A*** which is a required condition for the matrix subtraction as well in this case and '***det'*** is the determinant of the matrix. For each eigenvalue λ, a corresponding eigen-vector ***v***, can be found by solving:

$$(\lambda I - A)v = 0$$

Step 5: Choosing components and forming a feature vector:

We order the eigenvalues from largest to smallest so that it gives us the components in order or significance. Here comes the dimensionality reduction part. If we have a dataset with *n* variables, then we have the corresponding *n*eigenvalues and eigenvectors. It turns out that the eigenvector corresponding to the highest eigenvalue is the principal component of the dataset and it is our call as to how many eigenvalues we choose to proceed our analysis with. To reduce the dimensions, we choose the first *p* eigenvalues and ignore the rest. We do lose out some information in the process, but if the eigenvalues are small, we do not lose much.

1.1 Reading the images

```
from PIL import Image

import os, glob, re

import numpy as np

import matplotlib.pyplot as plt

path1 = '/home/aditya3498/Downloads/mini-project-1/dataset'

with open(path2) as f:

        y = f.readlines()

        for i in y:

                i = i.strip('\n')
```

```
                i = re.sub(r'[ ]+', ' ', i)
                i = re.split(' ', i)
                arr1.append(i)
        i = 0
        while i < len(arr1):
                list_full += arr1[i]
                i += 1
i = 0
while i < len(list_full):
        if i % 2 == 0:
                list_path.append(list_full[i])
        else:
                list_labels.append(list_full[i])
        i += 1
for i in list_labels:
        if i not in list_dis_lab:
                list_dis_lab.append(i)
allfiles = list_path
for file in allfiles:
        list_img.append(file)
```

1.2 Converting to Grayscale and forming the X matrix

```
matrix = np.asarray([np.array(Image.open(i).resize((64, 64)).convert('L')).flatten() for i in list_img], 'f')
```

1.3 Calculating Mean Image Matrix and centering the images

```
final_matrix = np.transpose(matrix)

mean = final_matrix.mean(axis = 0)

final_matrix -= mean
```

2. Principal Component Analysis

2.0.1 Procedure behind PCA :

1. Covariance matrix of the centered images calculated.

2. Covariance Matrix used to calculate eigen vector matrix.

3. First k components selected.

4. Images reconstructed back

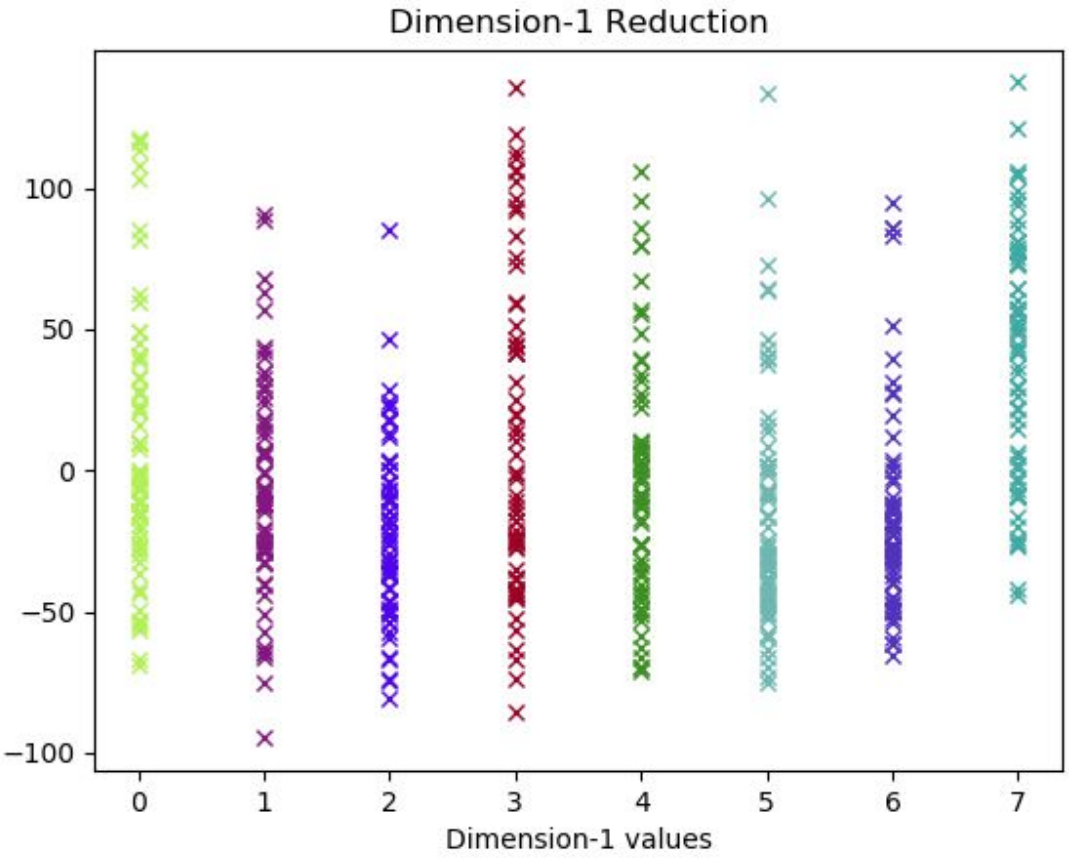
5. Returned the reconstructed images and the eigen vector matrix

```
M = np.dot(final_matrix, final_matrix.T)

e, EV = np.linalg.eigh(M)

idx = e.argsort()[::-1]

e = e[idx]

EV = EV[:,idx]

vectors_32 = EV.T[:32].T

Final_reduced = np.dot(matrix, vectors_32)
```

2.1 Scatter Plots

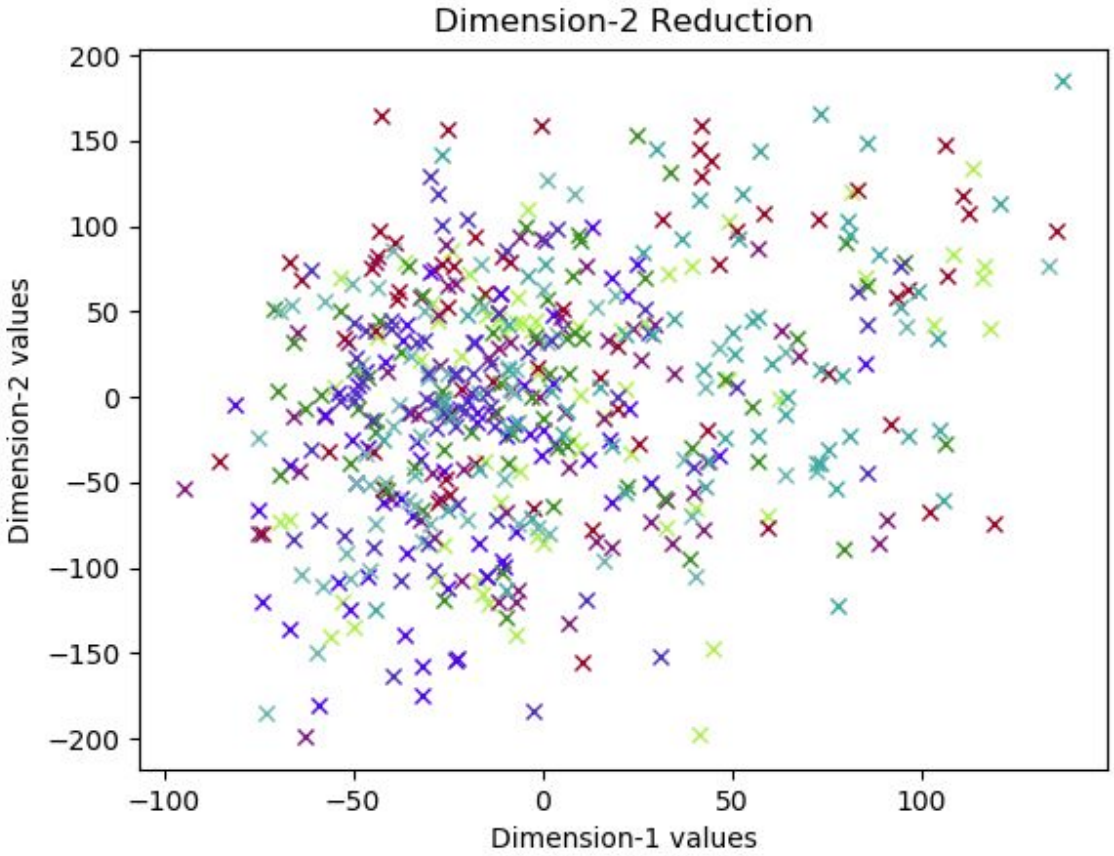
2.1 a) Scatter Plot for number of components k = 1

```
eigC = eigF[:1]
evecC=evecF[:][:1]
pca_images=np.matmul(Images-Images.mean(0),evecC.T)
plt.figure(1)
for i in range(520):
        plt.plot(i//65,pca_images[i], marker = 'x', color = colors_class[i])
plt.xlabel("Dimension-1 values")
plt.title("Dimension-1 Reduction")
```



2.1 b) Scatter Plots for number of components k = 2

```
eigC = eigF[:2]
evecC = evecF[:][:2]
pca_images=np.matmul(Images-Images.mean(0),evecC.T)
plt.figure(2)
for i in range(520):
        plt.plot(pca_images[i][0],pca_images[i][1],marker='x',color=col
ors_class[i])
plt.xlabel("Dimension-1 values")
plt.ylabel("Dimension-2 values")
plt.title("Dimension-2 Reduction")
```



2.1 c) Scatter Plots for number of components k = 3

```
ax = Axes3D(plt.gcf())
ax.scatter(pca_images[:,0],pca_images[:,1],pca_images[:,2],marker='x
',color=colors_class)
ax.set_xlabel("Dimension-1 values")
ax.set_ylabel("Dimension-2 values")
ax.set_ylabel("Dimension-3 values")
ax.set_title("Dimension-3 Reduction")
```