CS 663 - Assignment 4

Shaan Ul Haque
180070053

Samarth Singh
180050090

Niraj Mahajan
180050069

October 30, 2020

1 Code

Below is the MATLAB routine function for SVD of a matrix.

```
function [U S V]=MySVD(A)
    [U eig_values1]=eig(A*A');
    [V eig_values2]=eig(A'*A);
    [m m]=size(U);
    [n n]=size(V);
    if m>n
        S=eig_values1(:,m-n+1:end);
    else
        S=eig_values2(n-m+1:end,:);
    end
    S=S.^(0.5);
    LHS=A*V;
    RHS=U*S;
    % To check for the sign of eigenvectors because with the constraint
    % a'a=1 two vectors a and -a are possible
    for i=1:n
        if norm(LHS(:,i)-RHS(:,i))>=10e-2
            U(:,i)=-1*U(:,i);
        end
end
```

2 Example

Let us generate a random 5×3 matrix using randi function and find its SVD and verify the theorem.

2.1 Parameters

- Seed = 0
- Range of random numbers = [0,10]
- Size = 5×3

$$A=\begin{bmatrix} 8 & 1 & 1 \\ 9 & 3 & 10 \\ 1 & 6 & 10 \\ 10 & 10 & 5 \\ 6 & 10 & 8 \end{bmatrix}$$

2.2 Output

$$U=\begin{bmatrix} 0.1894 & -0.7053 & -0.2011 & -0.6151 & 0.2187 \\ -0.0020 & 0.4422 & -0.7470 & -0.0899 & 0.4882 \\ -0.3272 & -0.5211 & -0.1428 & 0.6775 & 0.3770 \\ -0.5596 & 0.1702 & 0.4984 & -0.3370 & 0.5440 \\ 0.7375 & 0.0803 & 0.3644 & 0.2026 & 0.5251 \end{bmatrix}$$

$$S=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 6.6128 & 0 & 0 \\ 0 & 9.0725 & 0 \\ 0 & 0 & 26.3051 \end{bmatrix}$$

$$V=\begin{bmatrix} -0.1972 & -0.7944 & 0.5745 \\ 0.8059 & 0.2023 & 0.5564 \\ -0.5583 & 0.5727 & 0.6003 \end{bmatrix}$$

Computing $U * S * V^T$ gives us back A. Hence, SVD stands verified. For better scrutiny, we have uploaded the MATLAB code as well.

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

1 Answer

We need to prove that after the eigen-vector corresponding to largest eigen-value is chosen the next best choice is to choose the eigen-vector with second largest eigen value. Let **e**, be the eigen-vector with largest eigen-value. Let **f** be a unit vector perpendicular to **e**. Then according to question we want to minimize the cost function:

$$J(\mathbf{f}) = \sum_{i=1}^{N} ||b_i\mathbf{f} - (\mathbf{x_i} - \mathbf{x} - a_i\mathbf{e})||^2$$

where, $a_i = \mathbf{e^T}(\mathbf{x_i}\text{-}\mathbf{x})$; $b_i = \mathbf{f^T}(\mathbf{x_i}\text{-}\mathbf{x})$; **x**=average value of $\mathbf{x_i}$

$$J(\mathbf{f}) = \sum_{i=1}^{N} ||b_i\mathbf{f}||^2 + \sum_{i=1}^{N} ||\mathbf{x_i} - \mathbf{x} - a_i\mathbf{e}||^2 - 2\sum_{i=1}^{N} b_i\mathbf{f}^T(\mathbf{x_i} - \mathbf{x} - a_i\mathbf{e})$$

$$J(\mathbf{f}) = \sum_{i=1}^{N} b_i^2 + \sum_{i=1}^{N} ||\mathbf{x_i} - \mathbf{x} - a_i\mathbf{e}||^2 - 2\sum_{i=1}^{N} ||b_i\mathbf{f}||^2$$

$$\implies J(\mathbf{f}) = -\sum_{i=1}^{N} b_i^2 + \sum_{i=1}^{N} ||\mathbf{x_i} - \mathbf{x} - a_i\mathbf{e}||^2$$

(Since, **f** is a unit vector, $\mathbf{f}^T\mathbf{e} = 0$, $\mathbf{f}^T(\mathbf{x_i} - \mathbf{x}) = b_i$)
We want to minimize J(**f**) and to minimize it we should maximize the first term in the latter equation.

$$\sum_{i=1}^{N} b_i^2 = \sum_{i=1}^{N} \mathbf{f}^T(\mathbf{x_i} - \mathbf{x})(\mathbf{x_i} - \mathbf{x})^T\mathbf{f}$$

We define $\sum_{i=1}^{N}(\mathbf{x_i} - \mathbf{x})(\mathbf{x_i} - \mathbf{x})^T = S$, where S = (N-1)C and C is the covariance matrix
Thus the above equation becomes,

$$\sum_{i=1}^{N} b_i^2 = \mathbf{f}^T\mathbf{S}\mathbf{f}$$

Thus, we need to maximize the above equation under the constraint, $\mathbf{f}^T\mathbf{f} = 1$. Using Langrange's multiplier method we get,

$$G(\mathbf{f}) = \mathbf{f}^T\mathbf{S}\mathbf{f} - \lambda(\mathbf{f}^T\mathbf{f} - 1)$$
$$G'(\mathbf{f}) = \mathbf{S}\mathbf{f} - \lambda\mathbf{f} = 0$$
$$\implies \mathbf{S}\mathbf{f} = \lambda\mathbf{f}$$

Thus, **f** needs to be an eigenvector of **S**. Rearranging the equation gives,

$$\mathbf{f}^T\mathbf{S}\mathbf{f} = \lambda$$

Thus, to maximize $\mathbf{f}^T\mathbf{S}\mathbf{f}$ we need maximum eigen-value. But we have already chosen the largest eigenvalue and its corresponding vector, thus the next best option is the second largest eigenvalue. Hence, proved that **f**, which is perpendicular to **e**, is the eigenvector corresponding to the second largest eigenvalue.

1.1 Another approach

We wanted the maximize $\mathbf{f}^T\mathbf{S}\mathbf{f}$. A more intuitive solution which also gives a physical interpretation is as follows:

$$\mathbf{f}^T\mathbf{S}\mathbf{f} = \mathbf{f}^T(\mathbf{S}\mathbf{f})$$

Let $\mathbf{f}^T\mathbf{S} = \mathbf{c}$, and the above equation becomes $\mathbf{f}^T\mathbf{c}$. The dot product of two vectors is maximized (under the constraint of constant magnitude of vectors) when angle between the two is $0°$ and thus $\mathbf{c}$ should be $\alpha\mathbf{f}$, where α is some constant. Written mathematically,

$$\mathbf{S}\mathbf{f} = \alpha\mathbf{f}$$

Thus, $\mathbf{f}$ is an eigenvector of $\mathbf{S}$ matrix.

Question 3

Part a:

Given a m x n matrix **A**,

- $\mathbf{P} = \mathbf{A}^T\mathbf{A}$ is a n x n matrix
  Consider a vector **y** with n elements

  $$\mathbf{y}^t\mathbf{P}\mathbf{y} = \mathbf{y}^t\mathbf{A}^T\mathbf{A}\mathbf{y} = (\mathbf{A}\mathbf{y})^t\mathbf{A}\mathbf{y} = ||\mathbf{A}\mathbf{y}||_2^2 \geq 0$$

  Note: $||\mathbf{x}||_2^2$ is the square of 2-norm of **x**

- $\mathbf{Q} = \mathbf{A}\mathbf{A}^T$ is a m x m matrix
  Consider a vector **z** with m elements

  $$\mathbf{z}^t\mathbf{Q}\mathbf{z} = \mathbf{z}^t\mathbf{A}\mathbf{A}^T\mathbf{z} = (\mathbf{A}^T\mathbf{z})^t\mathbf{A}^T\mathbf{z} = ||\mathbf{A}^T\mathbf{z}||_2^2 \geq 0$$

- Consider an eigenvector **y**, with n elements and eigenvalue λ_P, of **P**

  $$\mathbf{P}\mathbf{y} = \lambda_P\mathbf{y}$$

  Pre-multiplication of the above equation with $\mathbf{y}^t$

  $$\mathbf{y}^t\mathbf{P}\mathbf{y} = \lambda_P\mathbf{y}^t\mathbf{y}$$

  Replacing $\mathbf{y}^t\mathbf{P}\mathbf{y} = ||\mathbf{A}\mathbf{y}||_2^2$ from 1st point

  $$\lambda_P = \frac{||\mathbf{A}\mathbf{y}||_2^2}{||\mathbf{y}||_2^2} \geq 0$$

  $\therefore$ The eigenvalues of P are non-negative

- Consider an eigenvector **z**, with m elements and eigenvalue λ_Q, of **Q**

  $$\mathbf{Q}\mathbf{z} = \lambda_Q\mathbf{z}$$

  Pre-multiplication of the above equation with $\mathbf{z}^t$

  $$\mathbf{z}^t\mathbf{Q}\mathbf{z} = \lambda_Q\mathbf{z}^t\mathbf{z}$$

  Replacing $\mathbf{z}^t\mathbf{Q}\mathbf{z} = ||\mathbf{A}^T\mathbf{z}||_2^2$ from 2nd point

  $$\lambda_Q = \frac{||\mathbf{A}^T\mathbf{z}||_2^2}{||\mathbf{z}||_2^2} \geq 0$$

  $\therefore$ The eigenvalues of Q are non-negative

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Part b:

- Consider an eigenvector $\mathbf{u}$, with n elements and eigenvalue λ, of $\mathbf{P}$

  $$\mathbf{P}\mathbf{u} = \lambda\mathbf{u}$$

  $$\mathbf{A}^T\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

  Pre-multiplication of the above equation with $\mathbf{A}$

  $$\mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{u}) = \lambda(\mathbf{A}\mathbf{u})$$

  $\therefore$ $\mathbf{A}\mathbf{u}$ is an eigenvector, with m elements and eigenvalue λ, of $\mathbf{A}\mathbf{A}^T$

- Consider an eigenvector $\mathbf{v}$, with m elements and eigenvalue μ, of $\mathbf{Q}$

  $$\mathbf{Q}\mathbf{v} = \mu\mathbf{v}$$

  $$\mathbf{A}\mathbf{A}^T\mathbf{v} = \mu\mathbf{v}$$

  Pre-multiplication of the above equation with $\mathbf{A}^T$

  $$\mathbf{A}^T\mathbf{A}(\mathbf{A}^T\mathbf{v}) = \mu(\mathbf{A}^T\mathbf{v})$$

  $\therefore$ $\mathbf{A}^T\mathbf{v}$ is an eigenvector, with n elements and eigenvalue μ, of $\mathbf{A}^T\mathbf{A}$

Part c:

Consider an eigenvector $\mathbf{v}_i$, with m elements and eigenvalue λ_i, of $\mathbf{Q}$

$$\mathbf{Q}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

$$\mathbf{A}\mathbf{A}^T\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

Consider a vector $\mathbf{u}_i = \dfrac{\mathbf{A}^T\mathbf{v}_i}{||\mathbf{A}^T\mathbf{v}_i||_2}$

$$\mathbf{A}\mathbf{u}_i = \frac{\lambda_i}{||\mathbf{A}^T\mathbf{v}_i||_2}\mathbf{v}_i$$

Replacing $\gamma_i = \dfrac{\lambda_i}{||\mathbf{A}^T\mathbf{v}_i||_2}$ in the above equation

$$\mathbf{A}\mathbf{u}_i = \gamma_i\mathbf{v}_i$$

γ_i is non-negative because $\lambda_i \geq 0$ (proved in Part a 3^{rd} point)
γ_i is real because $\mathbf{A}$ and $\mathbf{v}_i$ are real-valued (given in question)
$\therefore$ There exist a real, non-negative γ_i such that $\mathbf{A}\mathbf{u}_i = \gamma_i\mathbf{v}_i$

Part d:

Given in the question, $\mathbf{u}_i^t\mathbf{u}_j = 0$ for $i \neq j$
As defined in the previous part, $\mathbf{u}_i^t\mathbf{u}_i = \frac{(\mathbf{A}^T\mathbf{v}_i)^t\mathbf{A}^T\mathbf{v}_i}{||\mathbf{A}^T\mathbf{v}||_2^2} = 1$
$\mathbf{V} = [\mathbf{u}_1|\mathbf{u}_2|...|\mathbf{u}_n]$ is a n x m matrix
Columns of $\mathbf{V}$ are orthonormal $\Rightarrow \mathbf{V}^T\mathbf{V} = \mathbf{I}_m$

Similarily we can say, $\mathbf{U} = [\mathbf{v}_1|\mathbf{v}_2|...|\mathbf{v}_m]$ also has orthonormal columns
Here we are assuming $\mathbf{v}_i$ to be a unit vector for consistency
$\therefore \mathbf{U}^T\mathbf{U} = \mathbf{I}_m$

$$\mathbf{A} = \mathbf{U}\mathbf{\Gamma}\mathbf{V}^T$$

Pre-multiply with $\mathbf{U}^T$ and post-multiply with $\mathbf{V}$

$$\mathbf{U}^T\mathbf{A}\mathbf{V} = \mathbf{U}^T\mathbf{U}\mathbf{\Gamma}\mathbf{V}^T\mathbf{V} = \mathbf{\Gamma}$$

Thus, $\mathbf{A} = \mathbf{U}\mathbf{\Gamma}\mathbf{V}^T \Leftrightarrow \mathbf{U}^T\mathbf{A}\mathbf{V} = \mathbf{\Gamma}$

$$\begin{aligned}\mathbf{U}^T\mathbf{A}\mathbf{V} &= \begin{bmatrix}\mathbf{v}_1^t\\ \mathbf{v}_2^t\\ ...\\ \mathbf{v}_m^t\end{bmatrix}\mathbf{A}[\mathbf{u}_1|\mathbf{u}_2|...|\mathbf{u}_m]\\ &= \begin{bmatrix}\mathbf{v}_1^t\\ \mathbf{v}_2^t\\ ...\\ \mathbf{v}_m^t\end{bmatrix}[\mathbf{A}\mathbf{u}_1|\mathbf{A}\mathbf{u}_2|...|\mathbf{A}\mathbf{u}_m]\\ &= \begin{bmatrix}\mathbf{v}_1^t\\ \mathbf{v}_2^t\\ ...\\ \mathbf{v}_m^t\end{bmatrix}[\gamma_1\mathbf{v}_1|\gamma_2\mathbf{v}_2|...|\gamma_m\mathbf{v}_m]\end{aligned}$$

Using result of part c: $\mathbf{A}\mathbf{u}_i = \gamma_i\mathbf{v}_i$ above
$(\mathbf{U}^T\mathbf{A}\mathbf{V})_{ij} = \gamma_j\mathbf{v}_i^t\mathbf{v}_j = \begin{cases}\gamma_j & i = j\\ 0 & i \neq j\end{cases}$
As defined, $\mathbf{\Gamma}$ is a m x n diagonal matrix with i^{th} diagonal entry equal to γ_i

$$\therefore \mathbf{U}^T\mathbf{A}\mathbf{V} = \mathbf{\Gamma} \Rightarrow \mathbf{A} = \mathbf{U}\mathbf{\Gamma}\mathbf{V}^T$$

Hence proved.

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Question 4

4.1: Results on the ORL datset

PCA using SVD

In this section, we performed PCA by performing SVD of the X_{dxn} data matrix using the matlab library function.
The maximum test accuracy of 95.31% was obtained at k = 50. There is a subsequent dip in the accuracy as the dataset is really small, and eventually the model starts to overfit.

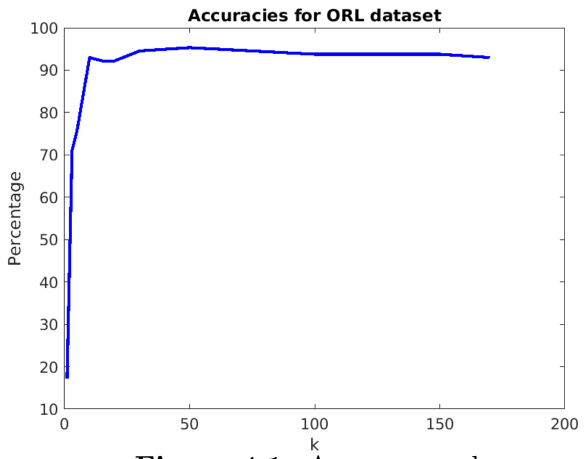

Figure 4.1: Accuracy vs k

PCA using Eigenvector decomposition

In this section, we performed PCA by performing Eigenvector decomposition of the $L = \frac{1}{n-1}X^TX$ matrix using the matlab library function *eig*, where X_{dxn} is the data matrix.
The results were exactly identical to the ones obtained in the previous part, since SVD of a matrix X simply generates the eigenvectors of the matrices X^TX and XX^T (ie the left and right eigenvectors). the The maximum test accuracy of 95.31% was obtained at k = 50.

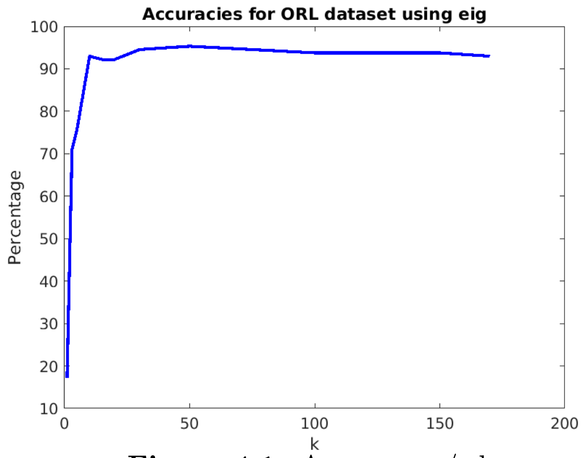

Figure 4.1: Accuracy v/s k

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

4.2: Results on the Yale dataset

Conventional PCA (Using all the top k eigenvectors)

Using the same method used in Part A of Section 4.1, we perform face recognition on the Yale Dataset. Since there is a huge contribution to the variance in the images arising from the lighting conditions, the eigenvectors corresponding to the highest variance are dominated by a variance in the lighting. Hence the accuracy obtained in this section is suboptimal.
The highest accuracy of 35.65% was recorded for k = 1000.

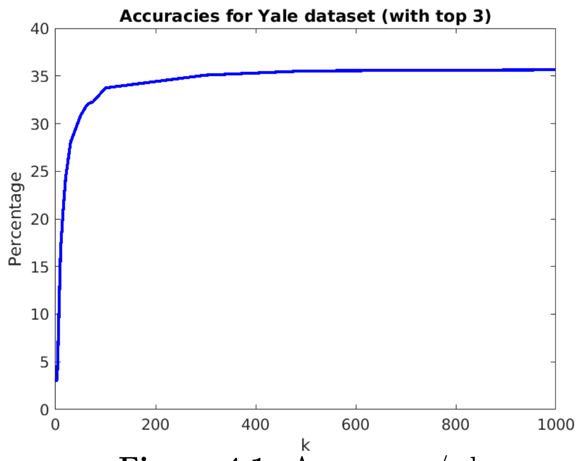

Figure 4.1: Accuracy v/s k

PCA while ignoring the top 3 eigenvectors

As stated in the previous sub section, the first few eigenvectors are dominated by the variance arising due to lighting changes in the images. Hence to counter these effects, we ignore the top 3 eigenvectors, and recompute our accuracies.
The highest accuracy of 60.67% was recorded for k = 1000.

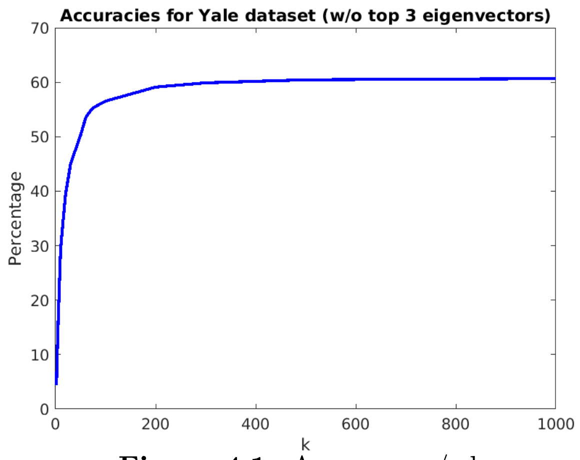

Figure 4.1: Accuracy v/s k

4.3: Usage of code

- Execute the **myMainScript.m** function to display the results and the plots. This takes around 68.22 seconds.
- Since the dataset is being used for Q4, Q5, Q6 in the assignment, we have kept the datasets in a common directory just inside our submission directory. The ORL dataset is expected to be in a relative directory **'../../datasets/'**, and the Yale dataset is expected to be in a relative directory **'../../datasets/CroppedYale/'** That is the per person directories should be as follows:
  '../../datasets/ORL/s*/' and
  '../../datasets/CroppedYale/yaleB/'**
- The **loadOrl.m** and **loadYale.m** functions load the data.
- The **fitPCA.m** and **fitPCAeig.m** functions generate the eigenvectors of the data matrix using the *svd* and *eig* functions respectively.
- The **getPredictor.m** functions returns a struct which stores the relevant eigenvectors, and the subspace transformation operator using the said eigenvectors.
- The **predict.m** function computes the test accuracy for a given data matrix, using the predictor object generated in the previous point.

Question 5

5.1: Reconstruction of Images Compressed by PCA

Using the PCA performed in the previous question, we compressed several images from the ORL dataset (d = 92x112 ≡ 10304) to a mere k dimensional space for $k << d$. In this section, we reconstruct images back from their compressed state, and compare the information retention for different values of k

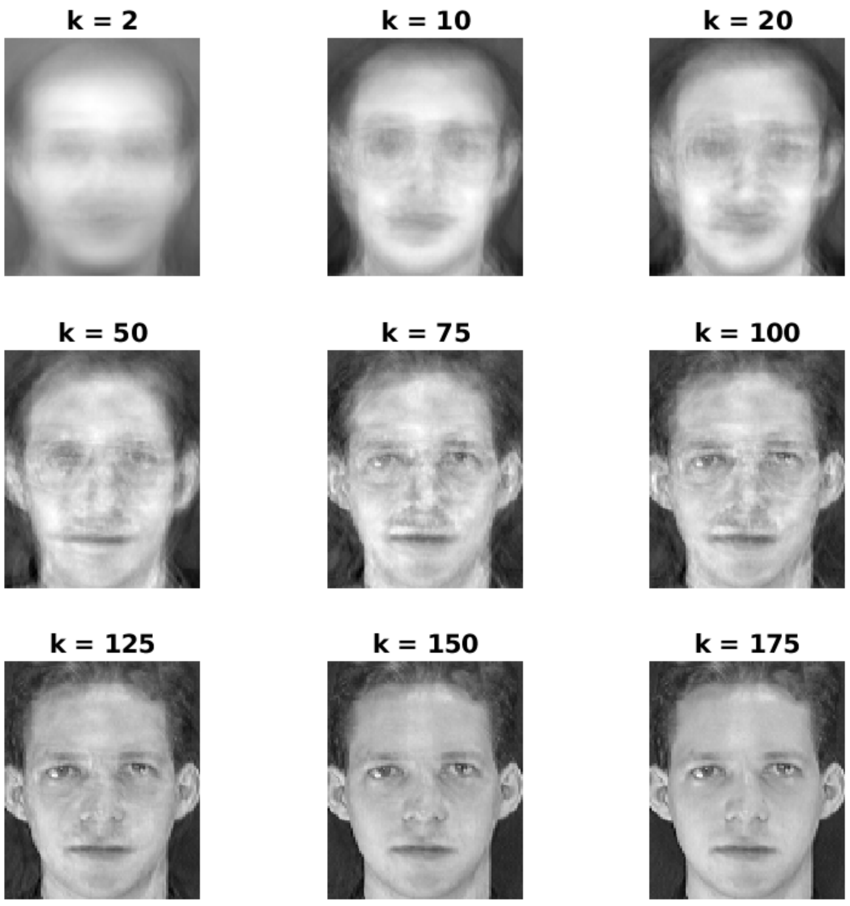

Figure 5.1: Reconstruction of images of person 1 compressed using various values of k

5.2: Visualising Eigenfaces

In this section, we visualise the eigenvectors corresponding to the highest 25 eigenvalues, by reshaping them into a 112x92 dimensional image.

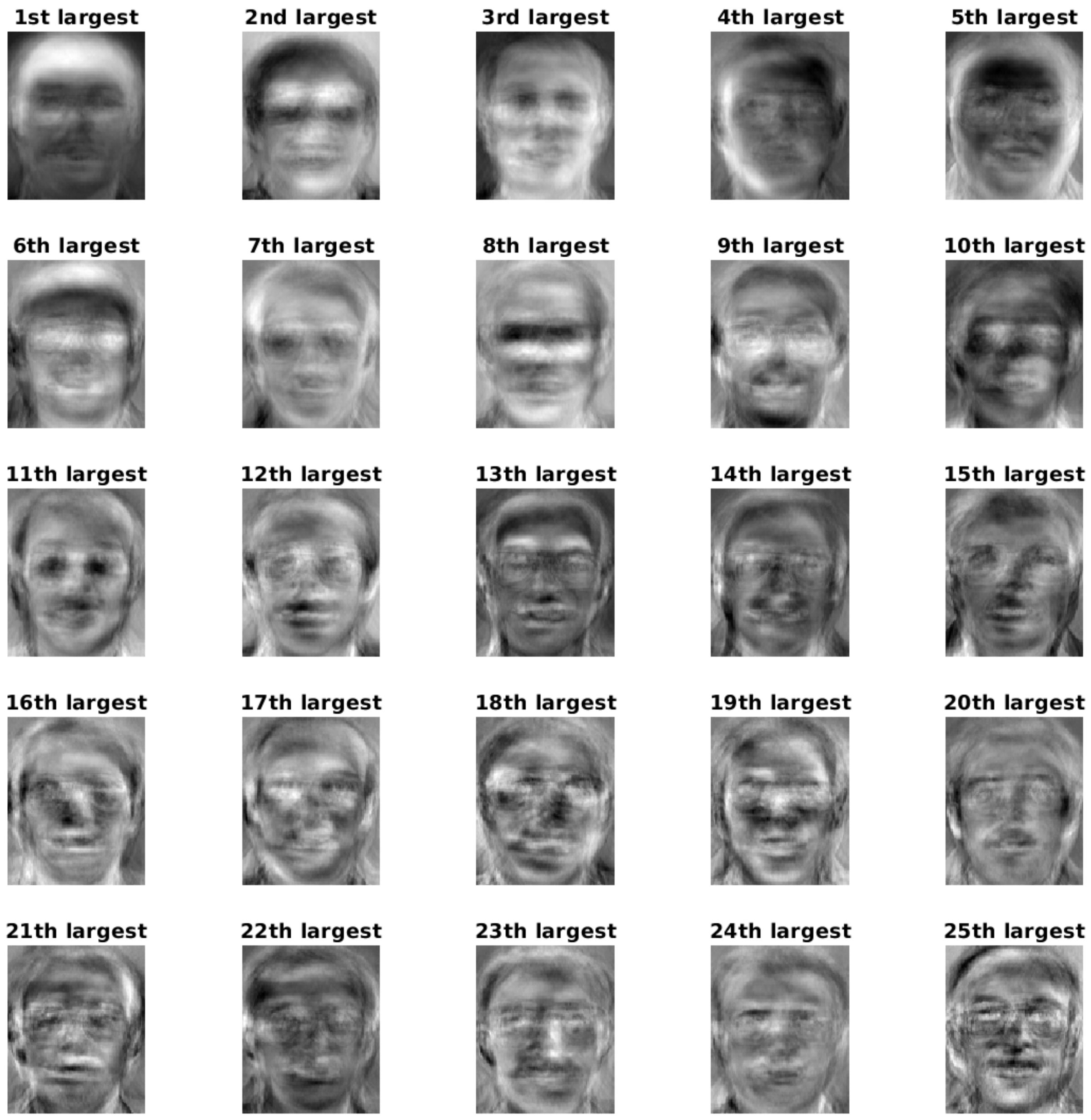

Figure 5.2: Top 25 eigenfaces

5.3: Usage of code

- Execute the **myMainScript.m** function to display the results and the plots. This takes around 13.5 seconds.
- Since the dataset is being used for Q4, Q5, Q6 in the assignment, we have kept the datasets in a common directory just inside our submission directory. The ORL dataset is expected to be in a relative directory **'../../datasets/'**, and the Yale dataset is expected to be in a relative directory **'../../datasets/CroppedYale/'** That is the per person directories should be as follows:
'../../datasets/ORL/s*/' and
'../../datasets/CroppedYale/yaleB/'**
- The **loadOrl.m** function loads the data.
- The **fitPCA.m** function generates the eigenvectors of the data matrix using the *svd* library function.
- The **getPredictor.m** functions returns a struct which stores the relevant eigenvectors, and the subspace transformation operator using the said eigenvectors.

Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Question 6

5.1: Detecting Faces Outside our Trainset

The score that we used in the previous questions to classify an image to a person was the Euclidean Distance between the coefficients (for the PCA transform) of the test image and images in the train set. We thus select the image from the train set (say I) whose coefficients are closest to the coefficients of our test image (say I') and assign this image to the class to which the said image I belongs. But if draw our test images from the samples of unknown people (ie from out of the train set) then this image will have a high Euclidean distance from all the images in the train set (or gallery). Hence we can exploit this phenomenon to detect images outside the dataset. Our algorithm uses thresholding to classify images outside the trainset

The dataset was already divided into two parts - train and test in Q4 of this assignment. In our algorithm, we will need to fine tune the threshold hyper parameter, and hence we will further divide the data to create a validation dataset. Note that we cannot fine tune any hyper parameter on the test set, (which would be not just improper but ghastly erroneous), hence the creation of a validation set is a must. So our data is divided as follows:

- **Train set** - First 6 images for 32/40 people
- **Validation set** - 7th image for 32/40 people, and the first 4 images for the remaining 8/40 people
- **Test set** - 8th-10th images for 32/40 people, and the first 5th-10th images for the remaining 8/40 people

So, effectively the trainset has 192 images, the validation set has 64 (32+32) images, and the test set has 144 (96+48) images

The algorithm that we used is simple - For any test image I' compute the coefficients that are obtained using PCA, and get it's distance from the coefficients of all the images in the gallery (train set). Get the minimum such distance for our test image I'. If this distance d lies above some threshold λ, then the image I' is outside the trainset, and if it is lower than the same λ, then this image belongs to a person within the datset.

Results

As our threshold parameter increases, the number of False Positives increases, while te number of False Negatives Decreases. In a way, False Positives carry much more weight as compared to False negatives, as it is sometimes okay if a person in the system is not able to pass the security, but it can be detrimental if a person outside the database can pass the security.

By finetuning on our validation dataset, we establish that the optimal value for our thresholding parameter λ is 2090

At $lambda = 2090$,

- Number of False Positives = 2 out of 48 Negatives
- Number of False Negatives = 32 out of 96 Positives

This means that from our test set, out of the 48 images that lie outside the dataset, only two were classified as "inside", while out of the 96 images that lie inside the dataset, 32 were classified as "outside". This is somewhat acceptable, as we are ensure a tighter security by not allowing images that are outside the dataset to be classified as otherwise.

5.3: Usage of code

- Execute the **myMainScript.m** function to display the results and the plots. This takes less than 1 second.
- Since the dataset is being used for Q4, Q5, Q6 in the assignment, we have kept the datasets in a common directory just inside our submission directory. The ORL dataset is expected to be in a relative directory **'../../datasets/'**, and the Yale dataset is expected to be in a relative directory **'../../datasets/CroppedYale/'** That is the per person directories should be as follows:
  '../../datasets/ORL/s*/' and
  '../../datasets/CroppedYale/yaleB/'**
- The **loadOrl.m** function loads the data and classies it further into train, test and validation.
- The **fitPCA.m** function generates the eigenvectors of the data matrix using the *svd* library function.
- The **getPredictor.m** functions returns a struct which stores the relevant eigenvectors, and the subspace transformation operator using the said eigenvectors.
- The **predict.m** functions returns the number of false positives and false negatives on the specified dataset, with the help of the struct generated in the previous part. This function is used while fine tuning the thresholding hyper parameter, as well as while testing.
- Lines 15 through 20 in the **myMainScript.m** file have the code that is used to fine tune the thresholding parameter. It is currently commented out to prevent it from flooding the stdout.