Homework 3 - Maths Foundation for Machine Learning

kipngeno koech - bkoech

October 28, 2024

Question 1: Eigen Values and Eigen Vectors

Consider the matrix $A = \begin{bmatrix} 9 & 1 & -1 \\ -1 & 11 & 1 \\ -2 & 2 & 10 \end{bmatrix}$.

1. Obtain the characteristic polynomial of A. (Hint: find the determinant of $A - \lambda I$)

$$A - \lambda I = \begin{bmatrix} 9-\lambda & 1 & -1 \\ -1 & 11-\lambda & 1 \\ -2 & 2 & 10-\lambda \end{bmatrix}$$

2. From the characteristic polynomial, obtain the 3 eigenvalues of $\lambda_1, \lambda_2, \lambda_3$, such that $\lambda_1 < \lambda_2 < \lambda_3$.

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

$$\det \begin{bmatrix} 9-\lambda & 1 & -1 \\ -1 & 11-\lambda & 1 \\ -2 & 2 & 10-\lambda \end{bmatrix} = 0$$

$$(9-\lambda)((11-\lambda)(10-\lambda)-(2\times1))-1((-1)(10-\lambda)-(-2\times1))-1((-1)(2)-(-2)(11-\lambda)) = 0$$

$$(9-\lambda)((11-\lambda)(10-\lambda)-2)-1((-10+\lambda)+2)-1(-2+22-2\lambda) = 0$$

$$(9-\lambda)(110-21\lambda+\lambda^2-2)-1(-8+\lambda)-1(20-2\lambda) = 0$$

$$(9-\lambda)(108-21\lambda+\lambda^2)-1(-8+\lambda)-1(20-2\lambda) = 0$$

$$9(108-21\lambda+\lambda^2)-\lambda(108-21\lambda+\lambda^2)+8-\lambda-20+2\lambda = 0$$

$$972-189\lambda+9\lambda^2-108\lambda+21\lambda^2-\lambda^3+8-\lambda-20+2\lambda = 0$$

$$-\lambda^3+30\lambda^2-296\lambda+960 = 0$$

$$\lambda^3-30\lambda^2+296\lambda-960 = 0$$

$$(\lambda-10)(\lambda^2-20\lambda+96) = 0$$

$$(\lambda-10)(\lambda-8)(\lambda-12) = 0$$

$$\lambda_1 = 8, \lambda_2 = 10, \lambda_3 = 12$$

3. Using Gaussian elimination, find the eigenvectors of A corresponding to the eigenvalues $\lambda_1, \lambda_2, \lambda_3$.
for eigen value $\lambda_1 = 8$

$$A - 8I = \begin{bmatrix} 1 & 1 & -1 \\ -1 & 3 & 1 \\ -2 & 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & -1 \\ -1 & 3 & 1 \\ -2 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Augumented matrix:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ -1 & 3 & 1 & 0 \\ -2 & 2 & 2 & 0 \end{array}\right] \xrightarrow{R_2=R_2+R_1} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 4 & 0 & 0 \\ -2 & 2 & 2 & 0 \end{array}\right]$$

$$\xrightarrow{R_3=R_3+2R_1} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{array}\right] \xrightarrow{R_3=R_3-R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$\xrightarrow{R_2=\frac{1}{4}R_2} \left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \xrightarrow{R_1=R_1-R_2} \left[\begin{array}{ccc|c} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$x_1 - x_3 = 0 \Rightarrow x_1 = x_3$$

$$x_2 = 0$$

Let $x_3 = 1$, then $x_1 = 1$ and $x_2 = 0$. Therefore, the eigenvector corresponding to $\lambda_1 = 8$ is $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$.
for eigen value $\lambda_2 = 10$

$$A - 10I = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & 1 \\ -2 & 2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & 1 \\ -2 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Augumented matrix:

$$\left[\begin{array}{ccc|c} -1 & 1 & -1 & 0 \\ -1 & 1 & 1 & 0 \\ -2 & 2 & 0 & 0 \end{array}\right] \xrightarrow{R_1=-R_1} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ -1 & 1 & 1 & 0 \\ -2 & 2 & 0 & 0 \end{array}\right]$$

$$\xrightarrow{R_2=R_2+R_1} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ -2 & 2 & 0 & 0 \end{array}\right] \xrightarrow{R_3=R_3+2R_1} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 \end{array}\right]$$

$$\xrightarrow{R_3=R_3-R_2} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \xrightarrow{R_2=\frac{1}{2}R_2} \left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$\xrightarrow{R_1=R_1+R_2} \left[\begin{array}{ccc|c} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$x_1 - x_2 = 0, x_3 = 0 \Rightarrow x_1 = x_2$$

Let $x_2 = 1$, then $x_1 = 1$ and $x_3 = 0$. Therefore, the eigenvector corresponding to $\lambda_2 = 10$ is $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$.

for eigen value $\lambda_3 = 12$

$$A - 12I = \begin{bmatrix} -3 & 1 & -1 \\ -1 & -1 & 1 \\ -2 & 2 & -2 \end{bmatrix}$$

$$\begin{bmatrix} -3 & 1 & -1 \\ -1 & -1 & 1 \\ -2 & 2 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\left[\begin{array}{ccc|c} -3 & 1 & -1 & 0 \\ -1 & -1 & 1 & 0 \\ -2 & 2 & -2 & 0 \end{array}\right] \xrightarrow{R_1=-\frac{1}{3}R_1} \left[\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & 0 \\ -1 & -1 & 1 & 0 \\ -2 & 2 & -2 & 0 \end{array}\right]$$

$$\xrightarrow{R_2=R_2+R_1} \left[\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & 0 \\ 0 & -\frac{4}{3} & \frac{4}{3} & 0 \\ -2 & 2 & -2 & 0 \end{array}\right] \xrightarrow{R_3=R_3+2R_1} \left[\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & 0 \\ 0 & -\frac{4}{3} & \frac{4}{3} & 0 \\ 0 & \frac{4}{3} & -\frac{4}{3} & 0 \end{array}\right]$$

$$\xrightarrow{R_3=R_3+R_2} \left[\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & 0 \\ 0 & -\frac{4}{3} & \frac{4}{3} & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \xrightarrow{R_2=-\frac{3}{4}R_2} \left[\begin{array}{ccc|c} 1 & -\frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$\xrightarrow{R_1=R_1+\frac{1}{3}R_2} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$x_1 = 0, x_2 = x_3$$

Let $x_2 = 1$, then $x_1 = 0$ and $x_3 = 1$. Therefore, the eigenvector corresponding to $\lambda_3 = 12$ is $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$.

4. Using np.linalg.eig, compute the eigenvalues and eigenvectors of A eigen values:

$$\begin{pmatrix} 12. & 8. & 10. \end{pmatrix}$$

eigen vectors:

$$\begin{bmatrix} 9.00258517 \times 10^{-16} & -7.07106781 \times 10^{-1} & -7.07106781 \times 10^{-1} \\ -7.07106781 \times 10^{-1} & -3.36518470 \times 10^{-16} & -7.07106781 \times 10^{-1} \\ -7.07106781 \times 10^{-1} & -7.07106781 \times 10^{-1} & -5.62025848 \times 10^{-15} \end{bmatrix}$$

5. Compare the eigen vectors computed using np.linalg.eig and that obtained using Gaussian elimination in (3) above. Are they the same? If they are the same, explain why this is the case. Otherwise, explain why they are different
eigen vectors using np.linalg.eig:

$$\begin{pmatrix} 9.00258517e-16 & -7.07106781e-01 & 7.07106781e-01 \\ -7.07106781e-01 & -3.36518470e-16 & -7.07106781e-01 \\ -7.07106781e-01 & -7.07106781e-01 & -5.62025848e-15 \end{pmatrix}$$

eigen vectors using Gaussian elimination:

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

The eigen vectors are different. This is because the eigen vectors obtained using np.linalg.eig normalizes the eigen vectors to have a norm of 1. The eigen vectors obtained using Gaussian elimination are not normalized.

6. Now, given a diagonal matrix D, whose diagonal entries are the eigenvalues of A, and a matrix P whose columns are the eigenvectors of A, use the eigen values you obtained in (2) and the eigenvectors you obtained in (3) to show that A as PDP^{-1}.

$$D = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 12 \end{bmatrix}, P = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, A = \begin{bmatrix} 9 & 1 & -1 \\ -1 & 11 & 1 \\ -2 & 2 & 10 \end{bmatrix}$$

$$PDP^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 12 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}^{-1}$$

$$P^{-1} = \begin{pmatrix} 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 \end{pmatrix}$$

$$PDP^{-1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 12 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix}$$

$$DP^{-1} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 12 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 4 & -4 & 4 \\ 5 & 5 & -5 \\ -6 & 6 & 6 \end{bmatrix}$$

$$P(DP^{-1}) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & -4 & 4 \\ 5 & 5 & -5 \\ -6 & 6 & 6 \end{bmatrix} = \begin{bmatrix} 9 & 1 & -1 \\ -1 & 11 & 1 \\ -2 & 2 & 10 \end{bmatrix}$$

Therefore, $A = PDP^{-1}$.

7. Calculate the determinant of A, $det(A)$:

$\det(A) = 9((11 \times 10) - (2 \times 1)) - 1((-1 \times 10) - (-2 \times 1)) - 1((-1 \times 2) - (11 \times -2))$

$= 9(110-2) - 1(-10+2) - 1(-2+22) = 9(108) - 1(-8) - 1(20) = 972+8-20 = \mathbf{960}$

8. Calculate the trace of A, $tr(A)$:

$$\text{tr}(A) = 9 + 11 + 10 = \mathbf{30}$$

9. confirm the following holds:
   $det(A) = \lambda_1\lambda_2\lambda_3$

$$960 = 8 \times 10 \times 12 = 960$$

   $tr(A) = \lambda_1 + \lambda_2 + \lambda_3$

$$30 = 8 + 10 + 12 = 30$$

Question 2: Image Compression using SVD

In class we explored the concept of low rank approximation where we approximated a matrix as the product of two low rank matrices. With Singular Value Decomposition (SVD), you can approximate a matrix as the sum of several rank-1 matrices, and this was also demonstrated.

1. What are the dimensions of the provided image?

   $$\text{Dimensions of the image} = \textbf{902 x 602}$$

2. Obtain the SVD decomposition of the image and visualize the rank-1 matrix corresponding to the largest singular value.

   $$\text{Largest singular value} = \textbf{in the notebook}$$

3. Add up the rank-1 matrices for the top 3 singular values and visualize your result. What do you observe?
   The more I add up the rank-1 matrices, our rank-3 approximated image approaches the original image, more features are covered

4. Add up the rank-1 matrices for the top 10 singular values and visualize your result. What do you observe?
   The more I add up the rank-1 matrices, our rank-10 approximated image approaches the original image, more features are covered

5. Assuming 1 byte of memory is required for one element of the image matrix, determine the top k rank-1 matrices required to represent the image if a compression ratio of 13:1 is desired. What is the value of k? Add up the rank-1 matrices for the top K singular values and visaulize your results. Remember to show your workings

   $$\text{Compression ratio} = \frac{\text{Original size}}{\text{Compressed size}} = 13$$

   Original size:

   $$\text{Original size} = \text{Number of elements in the image} = 902 \times 602 = \textbf{543004} \text{ bytes}$$

   Compressed size = Number of elements in the top k rank-1 matrices = $k \times 902 + k \times 602 = 1504k$ byte

   $$\frac{543004}{1504k} = 13 = \frac{543004}{13} = 1504k$$

   $$k = \frac{543004}{13 \times 1504} = \textbf{28}$$

Question 3: Linear Regression

In linear regression, SVD can help by decomposing the design matrix, identifying dependencies between predictor variables, and enabling regularization techniques like ridge regression. By reducing the rank or eliminating singular values, SVD assists in stabilizing regression models and improving their predictive accuracy. In this question you will solve a linear regression problem using SVD. Complete the cells in the provided starter code and answer the following:

1. What is the size of the dataset?

   $$\text{Size of the dataset} = \mathbf{489}$$

2. How many singular values were obtained during the calculation of the co-efficients? Why is this number exact - neither more nor less?

   The number of singular values obtained during the calculation of the coefficients is **3**. This number is exact because it is equal to the number of features in the dataset.

3. Given the SVD decomposition of a matrix, $A = U\Sigma V^T$, show that $A^{-1} = U\Sigma^{-1}V^T$

   $$A^{-1} = (U\Sigma V^T)^{-1}$$

   $$A^{-1} = (V^T)^{-1}\Sigma^{-1}U^{-1}$$

   since U is orthogonal, we have $U^{-1} = U^T$

   $$A^{-1} = V\Sigma^{-1}U^T$$

4. After training the model, what coefficients did you obtain? Interpret the significance of this coefficients.

   $$\begin{bmatrix} 9.50068071 \\ -1.29862203 \\ -0.56460349 \end{bmatrix}$$

   The coefficients obtained are the weights assigned to each feature in the linear regression model. The coefficient of the first feature is 9.50068071, the coefficient of the second feature is -1.29862203, and the coefficient of the third feature is -0.56460349. These coefficients indicate the impact of each feature on the target variable. A positive coefficient indicates that the feature has a positive impact on the target variable, while a negative coefficient indicates that the feature has a negative impact on the target variable. The magnitude of the coefficient indicates the strength of the impact of the feature on the target variable. Therefore, the coefficients obtained provide information on the relationship between the features and the target variable in the linear regression model.

5. What is the mean squared error of the train split and test split?

   Mean squared error for training data: **1.1849172161729788**
   Mean squared error for testing data: **1.027395075540607**

6. Visualization: complete the visualization tasks in the starter notebook and report your observation.

   There is a linear relationship between feature 1 and the true value. As feature 1 increases, the true value also increases. There are some outliers in the data, but the overall trend is linear.

   There is a negative linear relationship between feature 2 and the true value. As feature 2 increases, the true value decreases. The data points are scattered and do not follow a clear linear trend. There are some outliers in the data.

   There is a weak linear relationship between feature 3 and the true value. The data points are scattered and do not follow a clear linear trend. There are some outliers in the data.

Question 4: PCA & Eigenfaces

PCA is used to project high dimensional vectors onto a lower dimensional subspace whose basis vectors capture the maximum variance in the data of high dimensional vectors.

An image can be seen as a high dimensional vector. For example, a 100x100 image is a 10000 dimensional vector. PCA can be used to reduce this dimensionality by projecting images onto a lower dimensional subspace, given a dataset of images.

Eigenfaces is an application of PCA to obtain a lower dimensional subspace onto which face images can be projected, such that face images can be reconstructed by linearly combining the basis vectors of the lower dimensional subspace. The basis vectors of this lower dimensional subspace are called the eigenfaces. Eigenfaces can be applied in face recognition. [You can read more about PCA in chapter 10 of the textbook and study this Wikipedia article on Eigenface] In this exercise, you will go through the process of implementing a face recognition system using eigenfaces. You will use faces from the olivetti faces dataset. A starter notebook has been provided for you in the handout.

1. In the starter code, the olivetti faces dataset has been loaded, using a scikit-learn function. A database of faces, a list of test faces, and a list of faces of people not in the database have been created.

   (a) How many face images are in the database?

   Number of face images in the database = **400**

   (b) How many images are in the list of test faces?

   Number of face images in the test faces = **36**

   (c) How many images are in the list of faces of people not in the database?

   Number of face images in the faces of people not in the database = **40**

   (d) What are the dimensions of each face image?

   Dimensions of each face image = **64 x 64**

2. Create a matrix T containing all faces in the database, such that each row corresponds to the pixels of a person's face. Note that the height of this matrix should be equal to the number of faces in the database. What are the dimensions of T ?

   Dimensions of matrix T = **324 x 4096**

3. Obtain and visualize the mean face

   Mean face = **in the notebook**

4. Obtain the covariance matrix, S, of the matrix T

   Covariance matrix, S = **in the notebook**

5. Obtain the eigenfaces of the covariance matrix S. Report the time (in seconds) it took to compute the eigenfaces.

   Time taken to compute the eigenfaces = **1 minute 28 seconds**

6. Why is the process of computing the eigenfaces as done above computationally expensive? Please explain.

   The process of computing the eigenfaces as done above is computationally expensive because the covariance matrix is of size 4096×4096. Computing the eigenfaces involves finding the eigenvectors of the covariance matrix. The eigenvectors are computed by solving the eigenvalue problem, which involves finding the roots of the characteristic equation of the covariance matrix. This process is computationally expensive because it involves finding the roots of a polynomial of degree 4096.

7. Given the mean face $\bar{x}$, and the matrix V whose columns are the eigenvectors of S, if M is a matrix whose rows are equal to the result of subtracting the mean face from the rows of T, show that $V = M^T U$, where U is the matrix whose columns are the eigenvectors of MM^T.

   There are two ways we can get the covariance matrix: row wise and feature wise:
   row wise: $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T = MM^T$
   feature wise: $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^T(x_i - \bar{x}) = M^T M$

   $$V = \text{eigenvectors of } S = \text{eigenvectors of } M^T M$$

   MM^T is a covariance matrix, with eigenvectors U

   $$MM^T U = \lambda U$$

   multiply both sides by M^T:

   $$M^T MM^T U = M^T \lambda U$$

   but, $M^T M = S$, therefore:

   $$SM^T U = \lambda M^T U$$

   and we know that $SV = \lambda V$:

   $$V = M^T U$$

8. Compute the eigenfaces using $M^T U$

   Eigenfaces using $M^T U$ = **in the notebook**

9. Why is obtaining the eigenfaces using $M^T U$ more computationally efficient than the approach used in (5)? Please explain.

Obtaining the eigenfaces using M^TU is more computationally efficient than the approach used in (5) because the matrix M is of size 324×4096, which is smaller than the covariance matrix S of size 4096×4096. Computing the eigenvectors of the covariance matrix S involves finding the roots of a polynomial of degree 4096, which is computationally expensive. On the other hand, computing the eigenvectors of the matrix MM^T involves finding the roots of a polynomial of degree 400, which is less computationally expensive. Therefore, obtaining the eigenfaces using M^TU is more computationally efficient than the approach used in (5).

10. Observe the numerical values of the eigenfaces obtain using both approaches. Are they the same? If they are the same, explain why? otherwise explain why they are different.

    They are not the same. This is because the eigenfaces obtained using the covariance matrix S are the eigenvectors of the covariance matrix, while the eigenfaces obtained using M^TU are the eigenvectors of the matrix MM^T. The eigenvectors of the covariance matrix S and the matrix MM^T are different because the matrices are different. The eigenvectors of the covariance matrix S capture the directions of maximum variance in the data, while the eigenvectors of the matrix MM^T capture the directions of maximum variance in the mean-centered data. Therefore, the eigenfaces obtained using the covariance matrix S and M^TU are different.

11. Now visualize the first eigenface (the eigenface that captures the most variance in faces in the database) obtained using the approach in (5) and (8). Compare the images, they should be similar, why are they similar?

    First eigenface = **in the notebook**

    The first eigenface obtained using the approach in (5) and (8) are similar. This is because the first eigenface captures the direction of maximum variance in the data. The eigenvectors of the covariance matrix S and the matrix MM^T capture the directions of maximum variance in the data. Therefore, the first eigenface obtained using the approach in (5) and (8) are similar.

12. Using the eigenfaces obtain in (5), select the top eigenfaces that capture 90% of the variance in the database. How many eigenfaces were selected?

    Number of eigenfaces selected = **62 faces**

13. Visualize the top 10 eigenfaces

    Top 10 eigenfaces = [**in the notebook**]

14. Given a matrix, A, whose columns are the top eigenfaces selected in (12), show that the projection matrix unto the column space of A is the identity matrix.

    A = top eigenfaces = [**in the notebook**]

15. Given a new face, x, and the mean image $\bar{x}$, show that the coordinate of x in the eigenfaces subspace is given by $A^T(x - \bar{x})$

$$x = \text{new face}, \bar{x} = \text{mean image}, A = \text{top eigenfaces}$$

$$x = \bar{x} + \sum_{i=1}^{62} (A_i^T(x - \bar{x}))A_i$$

$$x = \bar{x} + A^T(x - \bar{x})A$$

$$x = \bar{x} + A^T(x - \bar{x})$$

$$x - \bar{x} = A^T(x - \bar{x})$$

$$A^T(x - \bar{x}) = A^T(x - \bar{x})$$

16. Given the coordinate of a new face, we can calculate the distance between this coordinate and the coordinates of all faces in the database. The new face is recognized if the minimum distance is less than a certain threshold. With the information you now have about eigenfaces, design a face recognition system that recognizes faces similar to those in the face database. Identify a threshold such that the accuracy of your system is greater than 80%.

$$\text{Threshold} = \mathbf{14.214214214214214}$$

17. For the obtained threshold, Evaluate the face recognition system's performance using the following metrics

    (a) accuracy

    0.8333333333333334

    (b) Precision

    1.0

    (c) Recall

    0.8333333333333334

    (d) F1-score

    0.9090909090909091

18. For your face recognition system, since you already know the coordinates of all faces in the database with respect to the eigenfaces, you can decide to discard the database of faces and only work with this coordinates. This is also done when you decide to include faces of new people to the system, so that they can be recognized. Using this information, calculate the compression ratio of your face recognition system if you decide to store only the coordinates of faces. Show your workings.

$$\text{Compression ratio} = \frac{\text{Original size}}{\text{Compressed size}}$$

$$\text{Original size} = \text{Number of elements in the database} \times \text{Number of eigenfaces}$$

$$\text{Compressed size} = \text{Number of elements in the database} \times \text{Number of faces}$$

$$\text{Compression ratio} = \frac{400 \times 62}{400 \times 4096} = \mathbf{0.01513671875}$$

19. Conduct experiments to assess the impact of different parameters and settings on your face recognition system. Consider factors like the number of eigenfaces.

$$\text{Impact of different parameters and settings} = [\textbf{in the notebook}]$$

20. Analyze and interpret the results, discussing the strengths and limitations of the Eigenfaces method in your context.
I ran a simulation to evaluate the performance of the face recognition system with different numbers of eigenfaces. The results show that the accuracy of the system increases as the number of eigenfaces increases. However, the computational cost of the system also increases with the number of eigenfaces. Therefore, there is a trade-off between accuracy and computational cost. The Eigenfaces method is effective in capturing the variations in the data and recognizing faces based on these variations. However, the method has limitations in handling variations in lighting, pose, and facial expressions. The Eigenfaces method is sensitive to variations in the data and may not perform well in the presence of noise or outliers. Overall, the Eigenfaces method is a powerful technique for face recognition, but it has limitations in handling complex variations in the data.