
Complex Systems Data Analysis EE520

Project 1: Singular Value Decomposition and Applications

Name: Anuj Patil

Department: Applied Data Science

Date: 09/20/2023

Introduction

This report presents solutions to a set of problems related to Singular Value Decomposition (SVD) and its applications, including dimension reduction and image compression.

Problem 1:

Problem Statement: Transformation of a Unit Circle and Singular Value Analysis

Given a unit circle in R2, we aim to explore the effects of a matrix transformation on the circle and understand the geometrical significance of the singular value decomposition (SVD).

Tasks:

1. Create a plot of a unit circle with radius *r* cantered at the origin.
2. Define matrix *A*= [3 −2;−1 5].
3. Apply the matrix transformation on each point of the unit circle and plot the transformed figure.
4. Compute the SVD of matrix *A* to extract the left singular vectors $u1$ and $u2$ and singular values $\sigma 1$ and $\sigma 2$.
5. Overlay the scaled left singular vectors $\sigma 1 u1$ and $\sigma 2 u2$ on the plot. Observe their alignment with the axes of the transformed figure (ellipse).

Solution:

When we use the matrix *A* on a unit circle in R2, the circle changes into an ellipse. This shows how matrices can change shapes by stretching or rotating them.

The left singular vectors $\sigma 1 u1$ and $\sigma 2 u2$ match up with the long and short axes of the ellipse. In SVD, these vectors show the main directions of the data spread. The longer axis of the ellipse follows the bigger singular value, $\sigma 1$, showing where the data varies most. The shorter axis follows $\sigma 2$, where the data varies less. Fig1

Upon observing the graphs Fig2, you'll note that while the circle cantered at the origin transforms into an ellipse still cantered at the origin, the circle with an offset centre transform differently. Its centre shifts and might even take on a slightly different shape or orientation based on the transformation matrix and the offset amount. This observation underlines the significance of the initial placement of the circle's centre when undergoing a linear transformation.

MATLAB Code:

```
% Define the transformation matrix
matrix_A = [3 -2; -1 5];

% Generate coordinates for the unit circle
theta = linspace(0, 2*pi, 100);
circle_x_coord = cos(theta);
circle_y_coord = sin(theta);
circle_points = [circle_x_coord; circle_y_coord];

% Transform the unit circle using matrix A
ellipse_coords = matrix_A * circle_points;

% Compute SVD for matrix A
[U_matrix, Sigma_matrix, ~] = svd(matrix_A);

% Determine the scaled singular vectors
scaled_U_matrix = U_matrix * Sigma_matrix;

% Visualization for the circle at the origin and its transformation
figure;
plot(circle_x_coord, circle_y_coord, 'k--', 'LineWidth', 1);  % Displaying unit circle in dashed black
hold on;
plot(ellipse_coords(1,:), ellipse_coords(2,:), 'b-', 'LineWidth', 1.5);
quiver(0, 0, scaled_U_matrix(1, 1), scaled_U_matrix(2, 1), 'r', 'LineWidth', 1.5, 'MaxHeadSize', 0.5);
quiver(0, 0, scaled_U_matrix(1, 2), scaled_U_matrix(2, 2), 'g', 'LineWidth', 1.5, 'MaxHeadSize', 0.5);
xlabel('Horizontal Coordinate (X)');
ylabel('Vertical Coordinate (Y)');
title('Unit Circle, Its Transformation, and Principal Directions');
legend('Unit Circle', 'Transformed Ellipse', 'Principal Direction 1', 'Principal Direction 2');
axis equal;
grid on;

% Offset for the second circle's center
offset = [2; 3];

% Translate the circle's points to the new center
translated_circle = [circle_x_coord; circle_y_coord] + offset;

% Transform both circles using matrix A
transformed_translated_circle = matrix_A * translated_circle;

% Visualization for the offset circle and its transformation
figure;

% First subplot for circle centered at origin
subplot(1, 2, 1);
plot(circle_x_coord, circle_y_coord, 'k--', 'LineWidth', 1);  % Original circle at origin
hold on;
plot(ellipse_coords(1,:), ellipse_coords(2,:), 'b-', 'LineWidth', 1.5);
xlabel('X');
ylabel('Y');
title('Circle at Origin & Its Transformation');
legend('Original Circle', 'Transformed Circle');
axis equal;
axis([-4 4 -4 4]);  % Manually set the axis limits
grid on;

% Second subplot for circle with offset center
subplot(1, 2, 2);
plot(translated_circle(1,:), translated_circle(2,:), 'k--', 'LineWidth', 1);  % Original circle with offset
hold on;
plot(transformed_translated_circle(1,:), transformed_translated_circle(2,:), 'm-', 'LineWidth', 1.5);
xlabel('X');
ylabel('Y');
title('Offset Circle & Its Transformation');
legend('Original Circle (Offset)', 'Transformed Circle');
axis equal;
axis([-4 8 -4 8]);  % Manually set the axis limits
grid on;
```

Graphs:

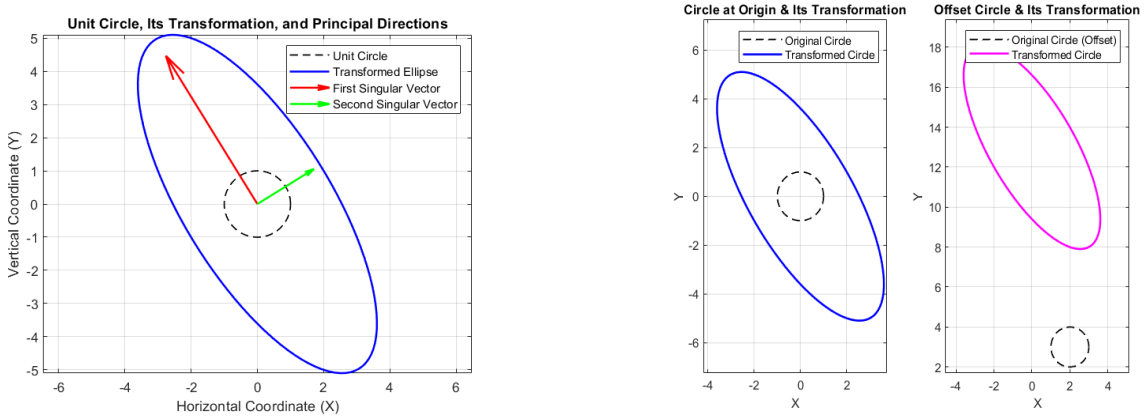


Figure 1 and 2

Problem 2:

Problem Statement: Least Squares Linear Regression

Given a set of points in R2, the objective is to determine the equation of the line that best fits the data in the least squares sense.

Tasks:

1. Using the points (0,1) (1, −1), and (2, −2), plot these data points on a scatter plot.
2. Implement a linear regression algorithm to determine the equation of the line that best fits these points.
3. Plot the resulting regression line on the same scatter plot.
4. Extract and state the slope and y-intercept of the regression line.

Simple Linear Regression Explanation:
Simple linear regression aims to fit a straight line between a dependent variable *y* and an independent variable *x*. The line is represented by the equation $y=mx+c$, where *m* is the slope and *c* is the y-intercept. In the context of least squares, the best fit line minimizes the sum of the squared differences (or "errors") between the given points and the values predicted by the model.

Given the points (0,1), (1, -1), and (2, -2), we will use linear regression to find the best fit line.

Graph:

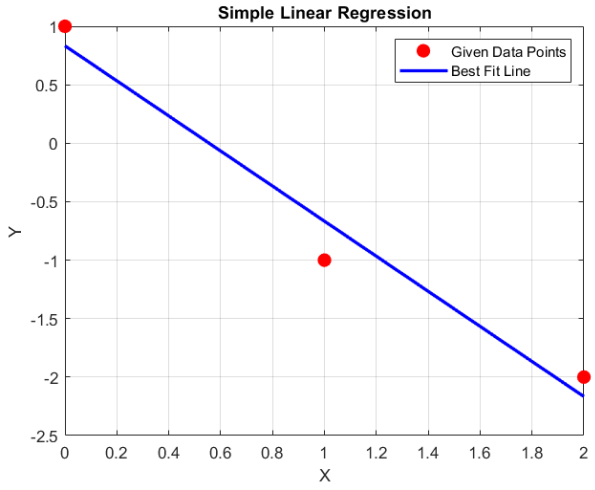


Figure 3

MATLAB Code for Linear Regression:

```
% Given data points
x = [0; 1; 2];
y = [1; -1; -2];

% Using MATLAB's polyfit function for linear regression
coefficients = polyfit(x, y, 1);
slope = coefficients(1);
y_intercept = coefficients(2);

% Predicted y values based on the regression
y_pred = slope * x + y_intercept;

% Visualization
figure;
plot(x, y, 'o', 'MarkerSize', 8, 'MarkerEdgeColor', 'red', 'MarkerFaceColor', 'red'); % Original data points
hold on;
plot(x, y_pred, '-b', 'LineWidth', 2); % Regression line
xlabel('X');
ylabel('Y');
title('Simple Linear Regression');
legend('Given Data Points', 'Best Fit Line');
grid on;
```

Problem 3:

Reduced order modelling with respect to the Euclidean norm, is a big deal, and ubiquitous as an application of SVD.

Yes!

Problem 4:

Problem Statement: Dimension Reduction and Line Approximation in R3

Given a dataset in R3, our goal is to approximate this data with an affine one-dimensional space (a line that does not pass through the origin) and subsequently reduce the dimensionality of the data.

Tasks:

1. **Data Loading and Visualization**:
    - Load the dataset **sdata.csv** which contains a 1000 × 3 matrix. Each row of this matrix represents a point (xi,yi,zi) in R3.
    - Create a 3D scatter plot of the data points using the **plot3** function in MATLAB.
2. **Line Approximation**:
    - Shift the dataset to ensure it has zero mean.
    - Determine the line in R3 that best approximates the data by minimizing the sum of squares of the projections of all points onto this line.
    - Overlay the approximated line on the 3D scatter plot to visually assess the fit.
3. **Dimension Reduction**:
    - Propose a mapping that represents each data point (xi,yi,zi) by a single number wi, which indicates its position along the approximated line.
    - Derive the formula that converts (x,y,z) coordinates to w and the reverse formula that converts w back to a point (x,y,z).

4. **Data Transformation and Visualization**:
    - Transform the dataset into *wi* coordinates using the derived formula.
    - Create a histogram of the *wi* values to analyse the distribution of the projected data points. Use 20 equally spaced bins for the histogram.

Solution:

a) Find the best approximating line:
To determine the best approximating line, we will first find the principal component of the data. This is achieved by performing SVD on the cantered data.

b) Convert (*x*,*y*,*z*) to *w*:
Once we have the principal direction (let's call it *v*), the position *w* of a point *p* along this direction is given by the dot product $w=p\cdot v$. The reverse transformation, from *w* to *p*, is $p=w\times v$.

c) Convert the data set to *wi* coordinates and plot a histogram:
After obtaining *wi* for each data point, we can easily visualize the distribution using a histogram.

Graphs:

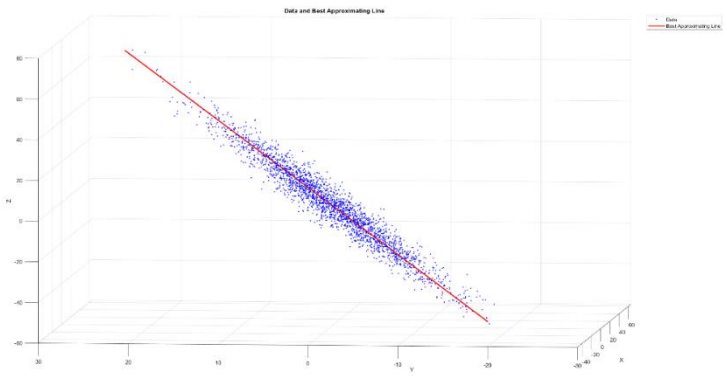


Figure 4

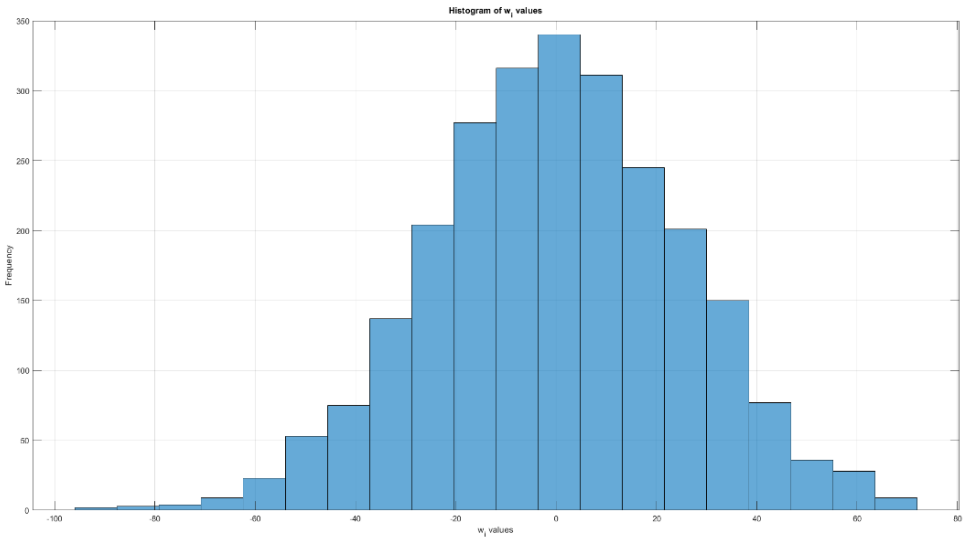


Figure 5

MATLAB Code for the Above Steps:

```
% Load the data
data = csvread('sdata.csv');

% Center the data (make it zero mean)
mean_data = mean(data, 1);
centered_data = data - mean_data;

% Perform SVD on the centered data
[U, S, V] = svd(centered_data, 'econ');

% The principal direction is the first right singular vector
principal_direction = V(:,1);

% a) Plot the data and the approximating line
figure;
scatter3(centered_data(:,1), centered_data(:,2), centered_data(:,3), 'b.');
hold on;
% Create points for the line (extending the line beyond the data range for visualization)
t = linspace(min(centered_data * principal_direction), max(centered_data * principal_direction), 100);
line_points = t' * principal_direction';
plot3(line_points(:,1), line_points(:,2), line_points(:,3), 'r', 'LineWidth', 2);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Data and Best Approximating Line');
legend('Data', 'Best Approximating Line');
grid on;

% b) Convert (x, y, z) to w and vice versa
w = centered_data * principal_direction;

% c) Histogram of the w_i values
figure;
histogram(w, 20);
xlabel('w_i values');
ylabel('Frequency');
title('Histogram of w_i values');
grid on;
```

Problem 5:

Compute the SVD in closed form, by hand, of the matrix, A=[1 -1; 0 1; 1 0];

Solution:

Question 5:-

Given Matrix

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

Step 1:- Find A^TA and AA^T

$$A^TA = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$A^TA = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix}$$

$$AA^T = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Step 2:- Find the Eigenvalues and Eigenvectors of A^TA and AA^T.

For A^TA :-

The characterstic polynomial is given by:-

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

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

Expanding the determinant:-

$$(2-\lambda)(2-\lambda) - (-1)(-1) = 0$$

$$\lambda^2 - 4\lambda + 3 = 0$$

$$(\lambda-3)(\lambda-1) = 0$$

The eigenvalues are:-

$\lambda_1 = 3$

$\lambda_2 = 1$

Eigenvector as follows:-

For $\lambda_1 = 3$

$(A^TA - \lambda I)v_1 = 0$

$\therefore v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

For $\lambda_2 = 1$

$(A^TA - \lambda I)v_2 = 0$

$\therefore v_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

For AA^T :-

$$AA^T = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\det(AA^T - \lambda I) = 0$$

$$AA^T - \lambda I = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & -0 & -0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2-\lambda & -1-0 & 1-0 \\ -1-0 & 1-\lambda & 0-0 \\ 1-0 & 0-0 & 1-\lambda \end{bmatrix}$$

$$\Rightarrow -\lambda^3 + 4\lambda^2 - 3\lambda = 0$$

$$\lambda(-\lambda^2 + 4\lambda - 3) = 0$$

$$\lambda^2 + 4\lambda - 3 = 0$$

$$\therefore \lambda = 3, \lambda_2 = 1, \lambda_3 = 0$$

Eigenvectors :-

$$u_1 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \quad u_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad u_3 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

Thus, the matrices V, Σ an V^T

$$V = \begin{bmatrix} 2 & 0 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}, \Sigma = \begin{bmatrix} \sqrt{9} & 0 \\ 0 & \sqrt{1} \\ 0 & 0 \end{bmatrix}, V^T = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$$

Question 6 :-

Given the SVD matrix G :-

$$G = U \Sigma V^T$$

where,

① U is an m x m matrix whose columns are left singular vectors

② Σ is an m x n diagonal matrix with the singular values

③ V^T is an n x n matrix whose rows are the right singular vectors.

Given :-

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_m \\ | & | & & | \end{bmatrix} \qquad V^T = \begin{bmatrix} - & v_1^T & - \\ - & v_2^T & - \\ & \vdots & \\ - & v_n^T & - \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_p \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

The product $U\Sigma$ gives a matrix where each column u_i is scaled by σ_i:

$$U\Sigma = \begin{bmatrix} | & | & & | & | & & | \\ \sigma_1 u_1 & \sigma_2 u_2 & \cdots & \sigma_p u_p & 0 & \cdots & 0 \\ | & | & & | & | & & | \end{bmatrix}$$

Now,

when we multiply $U\Sigma$ by V^T, the resulting matrix G is formed by multiplying each scaled column $\sigma_i u_i$ by each row v_j^T in V^T

Thus, each element g_{ij} of G is formed by:-

$$g_{ij} = (\sigma_1 u_1)_i \times (v_1^T)_j + (\sigma_2 u_2)_i \times (v_2^T)_j + \cdots$$

However,

Since Σ is a diagonal matrix, only the terms where $i = j$ contribute to the sum All other terms will be multiplied by zero (Because the off-diagonal elements of Σ are zero)

Thus,

we can represent the entire matrix G:

$$G = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_p u_p v_p^T$$

Hence proved!!

Question 7 :-

Ans :-

Given :-

$$\text{Matrix } A = \begin{bmatrix} 11 & 8 & 5 & 8 \\ -10 & -12 & -14 & -12 \end{bmatrix}$$

$$\text{Matrix } \Sigma = \begin{bmatrix} 8\sqrt{13} & 0 & 0 & 0 \\ 0 & 2\sqrt{13} & 0 & 0 \end{bmatrix}$$

$$\text{Matrix } V^T = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 0 & -1/2 & 0 \\ 0 & 1/2 & 0 & -1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \end{bmatrix}$$

a) Given the Singular value decomposition of matrix $A = U \Sigma V^T$

(1) Singular values :-

$\Rightarrow$ From the provided Σ matrix

$$\sigma_1 = 8\sqrt{13}$$

$$\sigma_2 = 2\sqrt{13}$$

(2) Left Singular vectors:-

=> From the provided U matrix:

$$u_1 = \begin{bmatrix} \frac{2}{\sqrt{13}} \\ -\frac{3}{\sqrt{13}} \end{bmatrix} \qquad u_2 = \begin{bmatrix} \frac{3}{\sqrt{13}} \\ \frac{2}{\sqrt{13}} \end{bmatrix}$$

(3) Right singular vectors:-

$$v_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \qquad v_2 = \begin{bmatrix} 1/2 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}$$

b)

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$$

Given the singular values and vectors

$$\sigma_1 = 8\sqrt{13}, \quad u_1 = \begin{bmatrix} \frac{2}{\sqrt{13}} \\ -\frac{3}{\sqrt{13}} \end{bmatrix}, \quad v_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

$$\sigma_2 = 2\sqrt{13}, \quad u_2 = \begin{bmatrix} \frac{3}{\sqrt{13}} \\ \frac{2}{\sqrt{13}} \end{bmatrix}, \quad v_2 = \begin{bmatrix} 1/2 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}$$

we can now compute the two Rank 1 matrices:- $\sigma_1 u_1 v_1^T$ and $\sigma_2 u_2 v_2^T$

$$\sigma_1 u_1 v_1^T = 8\sqrt{13} \begin{bmatrix} \frac{2}{\sqrt{13}} \\ \frac{-3}{\sqrt{13}} \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

$$\sigma_2 u_2 v_2^T = 2\sqrt{13} \begin{bmatrix} \frac{3}{\sqrt{13}} \\ \frac{2}{\sqrt{13}} \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ -1/2 \\ 0 \end{bmatrix}$$

Now, we will sum these two matrices to obtain A rank 1.

$$\therefore A = \begin{bmatrix} 12.2426 & 8 & 3.7574 & 8 \\ -9.1716 & -12 & -14.8284 & -12 \end{bmatrix}$$

This matrix is obtained as a sum of two rank 1 matrices formed from the outer products of the left and right singular vectors, scaled by the singular values.

C) Find an orthonormal basis for the null space of A.

⇒ The null space of a matrix A is a set of all vectors x such that $Ax = 0$

Given:-

$$V^T = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 0 & -1/2 & 0 \\ 0 & 1/2 & 0 & -1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \end{bmatrix}$$

The last two rows of V^T provides the orthonormal basis for the null space of A. These are

$$v_3 = \begin{bmatrix} 0 \\ 1/2 \\ 0 \\ -1/2 \end{bmatrix} \qquad v_4 = \begin{bmatrix} 1/2 \\ -1/2 \\ 1/2 \\ -1/2 \end{bmatrix}$$

Thus,
The orthonormal basis for the null space of A is formed by the vectors v_3 and v_4

d) Find a matrix B in $\mathbb{R}^{2\times 4}$ of one rank, such that $\|B-A\|_2$ is as small as possible.

$\Rightarrow$ we will use the First Singular value and it's associated left and right Singular vectors, since the Singular values are ordered in decreasing magnitude.

The rank one approximation for A is

$$B = \sigma_1 u_1 v_1^T$$

Using the provided values:-

$$\sigma_1 = 8\sqrt{13}$$

$$u_1 = \begin{bmatrix} \frac{2}{\sqrt{13}} \\ -\frac{3}{\sqrt{13}} \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

we will compute matrix B using the above formula

∴ The matrix B in $R^{2\times4}$ of rank one, such that $\|B - A\|_2$ is minimized is

$$B = \begin{bmatrix} 8 & 8 & 8 & 8 \\ -12 & -12 & -12 & -12 \end{bmatrix}$$

Problem 8:

Problem Statement: Image Compression Using SVD in MATLAB

Given an image of a professor from the following URL:
http://www.soarnorthcountry.com/images/g_image/photo-007413.jpg:

1. Load and display the image in MATLAB.
2. Convert the image to grayscale.
3. Use singular value decomposition (SVD) to compress the image.
4. Visualize and compare the original and compressed images.
5. Analyse the trade-off between compression ratio and image quality.

Ans: Reading the Image:

```
H = imread('Professor_Photo.jpg');
imshow(H); % This will display the original image
```

Convert to Grayscale:

```
H_gray = rgb2gray(H);
```

SVD Decomposition:

```
[U, S, V] = svd(double(H_gray));
```

Compress the Image:

```
k = 50;
```

```
H_compressed = U(:, 1:k) * S(1:k, 1:k) * V(:, 1:k)';
```

Display Compressed Image:

```
imshow(uint8(H_compressed));
```

Explanation:

- The function imread reads the image from the file.
- rgb2gray is used to convert a coloured image into grayscale.
- Singular Value Decomposition (SVD) decomposes the image matrix into three matrices U, S, and V. The S matrix contains singular values in descending order.
- By selecting only, the first k singular values (and the corresponding vectors in U and V), we can approximate the original image. The approximation is of lower rank (and thus compressed) compared to the original image.
- The value of k determines the level of compression. A smaller k means more compression but also more loss of detail in the image.

Problem 9:

Problem Statement: Eigenface Analysis with a Provided Dataset

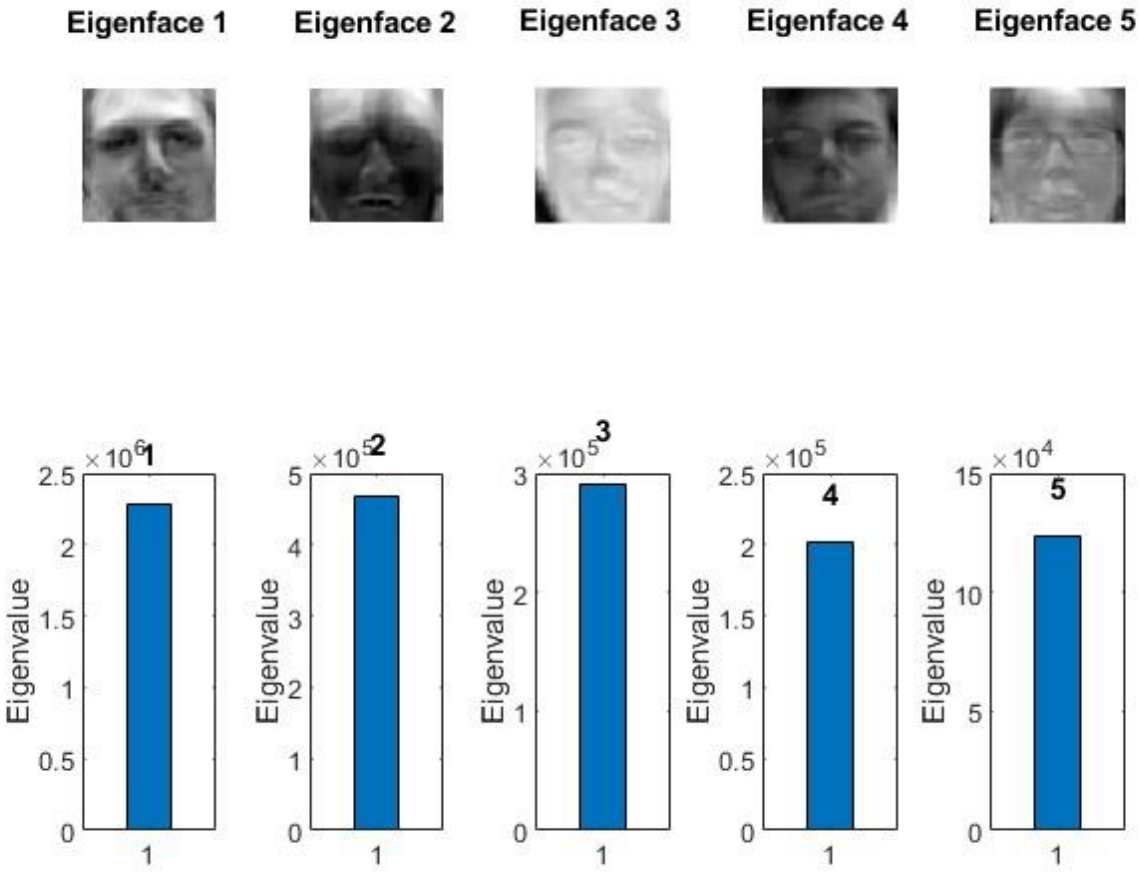
Using a provided dataset of facial images:

1. Load and preprocess the facial images.
2. Compute the eigenfaces by analysing the dataset's covariance matrix.
3. Visualize the top eigenfaces and interpret their significance.
4. Discuss potential applications and implications of eigenface analysis in fields such as facial recognition or emotion detection.

MATLAB Code:

```
% Eigenface analysis

% Load all images from the provided directories
nonsmiling_dir = 'nonsmiling_cropped';
smiling_dir = 'smiling_cropped';

nonsmiling_files = dir(fullfile(nonsmiling_dir, '*.jpg')); % Assuming JPEG images, change extension if needed
smiling_files = dir(fullfile(smiling_dir, '*.jpg'));

% Decide on a standard size based on the first image in nonsmiling_files
reference_image = imread(fullfile(nonsmiling_dir, nonsmiling_files(1).name));
standard_size = size(reference_image);

% Convert each image to grayscale, resize to standard size, and flatten
num_images = length(nonsmiling_files) + length(smiling_files);
flat_size = prod(standard_size(1:2));

all_data = zeros(num_images, flat_size);

for i = 1:length(nonsmiling_files)
    img = imread(fullfile(nonsmiling_dir, nonsmiling_files(i).name));
    gray_img = rgb2gray(img);
    resized_img = imresize(gray_img, standard_size(1:2));
    all_data(i, :) = resized_img(:)';
end

for i = 1:length(smiling_files)
    img = imread(fullfile(smiling_dir, smiling_files(i).name));
    gray_img = rgb2gray(img);
    resized_img = imresize(gray_img, standard_size(1:2));
    all_data(length(nonsmiling_files) + i, :) = resized_img(:)';
end

% Normalize the data by subtracting the mean face
mean_face = mean(all_data, 1);
all_data = all_data - repmat(mean_face, size(all_data, 1), 1);

% Compute the covariance matrix of the normalized data
cov_matrix = cov(all_data);

% Compute the eigenvectors and eigenvalues of the covariance matrix
[V, D] = eig(cov_matrix);
eigenvalues = diag(D);

% Sort eigenvectors based on eigenvalues
[eigenvalues, order] = sort(eigenvalues, 'descend');
V = V(:, order);

% Display the top few eigenfaces and their associated eigenvalues
num_eigenfaces_to_display = 5; % Change this number as needed

figure;
for i = 1:num_eigenfaces_to_display
    subplot(2, num_eigenfaces_to_display, i);
    eigenface = reshape(V(:, i), standard_size(1:2));
    imshow(mat2gray(eigenface));
    title(['Eigenface ' num2str(i)]);

    subplot(2, num_eigenfaces_to_display, i + num_eigenfaces_to_display);
    bar(eigenvalues(i));
    ylabel('Eigenvalue');
    title(['Value for Eigenface ' num2str(i)]);
end
```

Graphs:


Eigenface 1
Eigenface 2
Eigenface 3
Eigenface 4
Eigenface 5
Eigenvalue
1
2
3
4
5

Problem 8:

Problem Statement: Collaborative Learning Evidence

To foster collaborative learning and teamwork:

1. Engage in discussions and brainstorming sessions with at least two other students on at least two separate occasions.
2. Document these collaborative interactions by capturing either a photograph or a screenshot of your meetings.
3. While collaboration is encouraged, the final submission should be your independent effort and represent your own understanding.

Ans: