University of Toronto
Department of Electrical and Computer Engineering

ECE367 MATRIX ALGEBRA AND OPTIMIZATION

Problem Set #3

Autumn 2020

Prof. S. C. Draper **Due:** 5pm (Toronto time) Friday, 23 October 2020

Homework policy: Problem sets must be turned by the due date and time. Late problem sets will receive deductions for lateness. See the information sheet for futher details. The course text "Optimization Models" is abbreviated as "OptM". Also, see PS01 for details of the "Non-graded", "graded" and "optional" problem categories.

Note: In the categorization below Graded problems are highlighted in **red boldface**

Problem Set #3 problem categories: A quick categorization by topic of the problems in this problem set is as follows:

- Symmetric matrices: Problems 3.1, 3.2
- Quadratic constraints and ellipses: Problems 3.3, 3.4, **3.9**
- SVDs: Problems 3.4, 3.5, 3.6
- Applications of symmetric matrices and SVDs: Problems 3.7, 3.8, **3.10**, **3.11**

NON-GRADED PROBLEMS

Problem 3.1 (Eigenvectors of a symmetric 2×2 matrix)
OptM Problem 4.1.

Problem 3.2 (A lower bound on the rank)
OptM Problem 4.9 parts 1 and 2 only (not part 3).

Problem 3.3 (Quadratic constraints)
OptM Problem 4.2.

Problem 3.4 (SVD of an orthogonal matrix)
OptM Problem 5.1.

Problem 3.5 (Practice computing SVDs)
Compute by hand the singular value decomposition of

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}.$$

In other words, express A as $A = U\tilde{\Sigma}V^T$ where $U \in \mathbb{R}^{2,2}$ and $V \in \mathbb{R}^{3,3}$ are orthogonal matrices and $\tilde{\Sigma} \in \mathbb{R}^{2,3}$, specify the values of

(a) Specify the singular values by specifying $\tilde{\Sigma}$.

(b) Specify the right singular vectors by specifying V.

(c) Specify the left singular vectors by specifying U.

(d) Reassemble your calculations of parts (a)-(c) to show that $U\tilde{\Sigma}V^T$ in fact does equal the A you started with. Also express A as a sum of rank-one matrices where each matrix is the outer product of a left singular vector and a right singular value scaled by a singular value.

Problem 3.6 (SVD of a score matrix)
OptM Problem 5.4.

Problem 3.7 (Connected graphs and the Laplacian)
OptM Problem 4.5.

Problem 3.8 (Fitting a hyperplane to data)
OptM problem 5.6.

(Hints: To get the quadratic function in the second part of the problem sum up the m distances—one per data point—and manipulate the resulting sum into a quadratic form, cf. Sec. 4.1.2. In the last part of the question recall our development of Rayleigh quotients in Ch. 4.)

GRADED PROBLEMS

Problem 3.9 (Ellipses, eigenvalues, eigenvectors, and volume)
Make neat and clearly-labelled *sketches* (i.e., draw by hand) of the ellipsoid $\mathcal{E} = \{x|(x-x_c)^T P^{-1}(x-x_c) = 1\}$ for the following sets of parameters:

(a) Center $x_c = [0\ 0]^T$ and $P = [1.5\ \ -0.5; -0.5\ 1.5]$.

(b) Center $x_c = [1\ \ -2]^T$ and $P = [3\ 0; 0\ 1]$.

(c) Center $x_c = [-2\ 1]^T$ and $P = [9\ \ -2; -2\ 6]$.

For each part (a)–(c) also compute each pair of eigenvalues and corresponding eigenvectors.

(d) Recall the geometrically meaningful property of the determinant of a square real matrix A: its magnitude $|\det A|$ is equal to the volume of the parallelepiped $\mathcal{P}$ formed by applying A to the unit cube $\mathcal{C} = \{x|\, 0 \le x_i \le 1, i \in [n]\}$. In other words, if $\mathcal{P} = \{Ax|x \in \mathcal{C}\}$ then $|\det(A)|$ is equal to the volume of $\mathcal{P}$. Furthermore, recall that the determinant of a matrix is zero if any of its eigenvalues are zero. Explain how to interpret this latter fact in terms of the interpretation of $|\det(A)|$ as the volume of $\mathcal{P}$. (This interpretation was mentioned in class so this is just a "I want to make sure you understand that comment" type of question.)

Problem 3.10 (Latent semantic indexing)
In this problem you build off the problem "Angles between word vectors" from PS01, making a connection to the singular value decomposition. First complete the following two parts of OptM problem 5.5:

(a) OptM problem 5.5 part 3.

(b) OptM problem 5.5 part 4.

In the following parts we connect OptM problem 5.5 to the "Angles between word vectors" (ABWV) problem in PS01, and apply the latent semantic indexing method to the Wikipedia article collection. We start by briefly re-describing the ABWV problem set-up below, but please feel free to go back and read the original problem statement.

In ABWV, we considered a set of documents $\mathcal{D}$ the where the number of documents is $|\mathcal{D}|$. The set $\mathcal{W}$ denotes the union of words in all articles, i.e., the lexicon of the set of documents where the cardinality of $\mathcal{W}$ is $|\mathcal{W}|$. We assume the lexicon is ordered "lexiographically" (e.g., alphabetically) so that there is a one-to-one mapping from each word $w \in \mathcal{W}$ to an element of the index set $t \in [|\mathcal{W}|]$. Let $f_{\text{term}}(t, d)$ denote the number of times the word $w \in \mathcal{W}$ that is indexed as $t \in [|\mathcal{W}|]$

appears in the dth article where $d \in [|\mathcal{D}|]$. For ABWV, you were provided with a pre-processed MATLAB data file `wordVecV.mat`. Please re-use the same data file for this problem. You can load the content in the second file into MATLAB by using command `load 'wordVecV.mat'`. After loading, you will see a variable `V` of dimensions 1651×10. We refer to this matrix as V. The value in the tth row and dth column of this matrix is $f_{\text{term}}(t, d)$. Note that in the given dataset $|\mathcal{D}| = 10$ and $|\mathcal{W}| = 1651$.

Now we connect OptM problem 5.5 to the ABWV problem set-up. You will immediately notice that $m = |\mathcal{D}|$ and $n = |\mathcal{W}|$. You can compute the $n \times m$ "(raw) term-by-document matrix" M by noting that $[M]_{i,j} = \mathbb{1}([V]_{i,j})$, where $\mathbb{1}(x)$ is 1 if $x > 0$ and 0 otherwise. The OptM problem 5.5 also describes how to obtain $\tilde{M}$, a normalized version of M.

(c) Use MATLAB `svd` command to compute the singular value decomposition of $\tilde{M}$. List the 10 largest singular values in sorted order.

(d) In part (b) you assumed a low-rank approximation of $\tilde{M}$ and found an expression for the document similarity. Let the distance between ith and jth documents be $d(i, j)$ as per your expression from part (b). Let the rank of your approximation be k where $0 < k \leq \min(m, n)$. Compute $d(i, j)$ for $i, j \in [m]$ by assuming $k = 9$. Write down the titles of two most similar documents.

(e) Repeat what you did in part (d) with $k = 8, 7, \ldots, 1$. What is the lowest k that does not change your answer for part (d)? If your answer for lowest k is greater than 1 what is the pair of most similar documents for $k - 1$?

Problem 3.11 (Eigenfaces and ℓ_2 projection)
In this problem you will familiarize with the concept of Eigenfaces and its uses. Download the dataset `yalefaces.mat` from the course website. This dataset consists of 32×32 gray scale images of faces taken from the *Extended Yale Face Database B* (`http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`). Load the dataset into your MATLAB environment by executing the command `load('yalefaces.mat')`. You will see a new variable `M` of dimension $32 \times 32 \times 2414$ which consists of 2414 grayscale images, 32×32 pixels each. The pixel values of the images range from 0 to 255. You can view the loaded images by making use of MATLAB built-in functions `imshow` or `imagesc`. As an example, the first image of the dataset can be displayed by executing `imshow(M(:,:,1)/255)`.

Let N be the number of images in the dataset and let $d = 1024$, the total number of pixels in each image. An image can be thought of as a matrix with 32 columns and 32 rows consisting of entries in the range $[0, 255]$. In this exercise we consider the images in vector form. Let $\mathcal{J}_i$ be the ith image in the dataset where $i \in [N]$. We formulate a column vector $x^{(i)} \in \mathbb{R}^d$ by flattening the matrix that makes up $\mathcal{J}_i$. In our case we stack all 32 columns vertically to form a d-dimensional vector $x^{(i)}$. In MATLAB you can do this efficiently using the command `reshape`. The 'average face' vector of

the dataset $\bar{x}$ can be computed as $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$. We can (roughly) visualize the $x^{(i)}$s as forming a cloud of data points where the cloud is centered at $\bar{x}$ in d-dimensional space. Translate the cloud center to the origin by subtracting $\bar{x}$ from each $x^{(i)}$. Let the resulting vectors be denoted as $\bar{x}^{(i)} = x^{(i)} - \bar{x}$, which we will refer to as centered image vectors. Construct a matrix $X \in \mathbb{R}^{d\times N}$ by concatenating all the centered vectors $\bar{x}^{(i)}$, i.e., $X = \left[\bar{x}^{(1)}, \bar{x}^{(2)}, \ldots, \bar{x}^{(N)}\right]$. The matrix $C = XX^T$ is N times the covariance matrix of the data samples $x^{(i)}, i \in [N]$.

(a) In class you learned about singular value decomposition (SVD) and eigendecomposition. What is the connection between the singular values of X and the eigenvalues of C? What is the connection between the left-singular vectors of X and the eigenvectors of C? Make sure to describe the reasoning behind your answers, e.g., by describing the singular or eigen decompositions of each matrix.

(b) Compute the eigenvalue/eigenvector pairs of C and arrange them in decreasing order of the magnitude of the eigenvalues. Let the jth pair in the ordered set be denoted as $\lambda_j \in \mathbb{R}, v^{(j)} \in \mathbb{R}^d$ for $j \in [d]$. You may find the MATLAB command `svd` or `eig` helpful. Comment whether the eigenvalues are real and briefly explain why. Plot $\log \lambda_j$ against $j \in [d]$. Include your plot in your solution.

(c) The set of eigenvectors you computed in the previous part form a basis for the centered image vectors. Reshape each eigenvector $v^{(j)}$ to obtain a 32×32 matrix. These matrices can be considered as images and they are known as *eigenfaces*. Include plots of two sets of eigenfaces, those corresponding to the largest 10, and those corresponding to the smallest 10, eigenvalues. Do you observe any difference between the two sets of eigenfaces you plotted? If so briefly explain the reason for this difference.

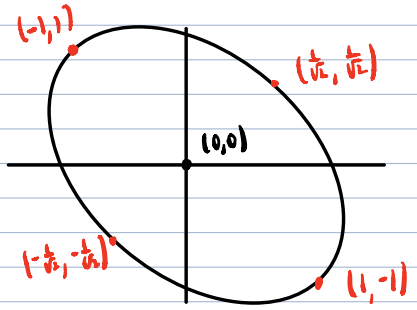
(d) In this part, consider the images $\mathcal{J}_i$ for $i \in \{1, 1076, 2043\}$. Let $\mathcal{B}_j = \{v^{(1)}, v^{(2)}, \ldots, v^{(j)}\}$ where $j = \{2^1, 2^2, 2^3, \ldots, 2^{10}\}$, i.e., j indexes sets each consisting of a different number of the eigenvectors. Let us denote the ℓ_2 projection of $\bar{x}^{(i)}$ onto the basis vector set $\mathcal{B}_j$ as $\bar{y}^{(i,j)}$. Compute $\bar{y}^{(i,j)}$ for the given i, j pairs using MATLAB. (I.e., do this using your numerical tool and not by hand.) Note that $\bar{y}^{(i,j)}$ vectors are computed using the centered image vectors. Translate these vectors back (un-center them) by the cloud center shift $\bar{x}$ to get the image vectors $y^{(i,j)} = \bar{y}^{(i,j)} + \bar{x}$. Reshape the $y^{(i,j)}$ vectors into 32×32 matrices and plot them as images. Note that you will need to plot 30 images and these can be compactly plotted using `subplot` command in MATLAB.

(e) In this part you will learn how the eigenfaces can be used for the task of *face recognition*. Consider the two sets of indices $\mathcal{I}_1 = \{1, 2, 7\}$ and $\mathcal{I}_2 = \{2043, 2044, 2045\}$. The faces in the set $\mathcal{J}_i$ for $i \in \mathcal{I}_1$ belong to one person and those for $i \in \mathcal{I}_2$ belong to a second person. We have carefully picked the image indices so that the corresponding images are taken under similar lighting conditions. Consider $\mathcal{B}_{25}$ where $\mathcal{B}_j$ is defined as in part (d). Compute the projection coefficients obtained by projecting $\bar{x}^{(i)}, i \in \mathcal{I}_1 \cup \mathcal{I}_2$ onto the eigenvectors in $\mathcal{B}_{25}$. Let $c^{(i)} \in \mathbb{R}^{25}$ be the vector that consists of coefficients obtained for ith image. The vector

$c^{(i)}$ can be thought of as a representation of the corresponding original image $\mathcal{J}_i$. Intuitively, images that belong to same person should have similar coefficient vectors. To verify this, compute the pairwise Euclidean distances between the $c^{(i)}$ vectors. Tabulate the values and comment whether the distance between any two $c^{(i)}$ vectors that belong to the same person is smaller than those belonging to the other person. Briefly explain how you can use this to build a simple face recognition scheme.

Problem 3.9

a) $\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}^{-1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$



$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$

$\begin{bmatrix} 0.75x_1 + 0.25x_2 & 0.25x_1 + 0.75x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$

$0.75x_1^2 + 0.5x_1x_2 + 0.75x_2^2 = 1$ (to verify solution)

$P - \lambda I = \begin{bmatrix} 1.5-\lambda & -0.5 \\ -0.5 & 1.5-\lambda \end{bmatrix} \Rightarrow 0 = (1.5-\lambda)^2 - 0.25$

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

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

$\lambda = 2$: $\begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$1.5x_1 - 0.5x_2 = 2x_1$

$-0.5x_1 + 1.5x_2 = 2x_2$

$\therefore x_1 = -x_2$

$v_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ Direction of axis with length $\sqrt{2}$

$\lambda = 1$: $\begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$1.5x_1 - 0.5x_2 = x_1$

$-0.5x_1 + 1.5x_2 = x_2$

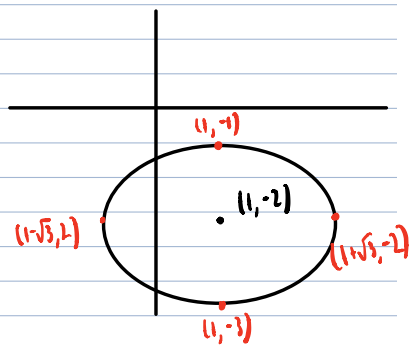
$\therefore x_1 = x_2$

$v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ Direction of axis with length 1

b) $\begin{bmatrix} x_1 - 1 & x_2 + 2 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_1 - 1 \\ x_2 + 2 \end{bmatrix} = 1$

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

$\begin{bmatrix} \frac{1}{3}x_1 - \frac{1}{3} & x_2 + 2 \end{bmatrix} \begin{bmatrix} x_1 - 1 \\ x_2 + 2 \end{bmatrix} = 1$



$\frac{1}{3}x_1^2 - \frac{1}{3}x_1 - \frac{1}{3}x_1 + \frac{1}{3} + x_2^2 + 4x_2 + 4 = 1$

$\frac{1}{3}x_1^2 - \frac{2}{3}x_1 + 4x_2 + x_2^2 = -\frac{10}{3}$ (to verify solution)

$P - \lambda I = \begin{bmatrix} 3-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} \Rightarrow 0 = (3-\lambda)(1-\lambda)$

$\lambda = 3: \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 3 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$3x_1 = 3x_1$

$x_2 = 3x_2$

$\therefore x_2 = 0, \quad x_1 = x_1$

$v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ Direction of axis with length $\sqrt{3}$

$\lambda = 1: \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

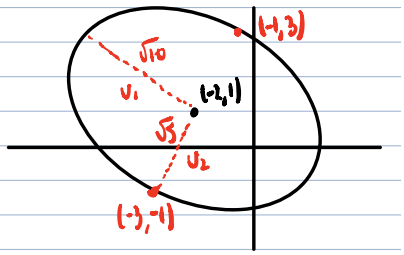
$3x_1 = x_1$

$x_2 = x_2$

$\therefore x_1 = 0, \quad x_2 = x_2$

$v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ Direction of axis with length 1

c) $\begin{bmatrix} x_1+2 & x_2-1 \end{bmatrix}\begin{bmatrix} 9 & -2 \\ -2 & 6 \end{bmatrix}^{-1}\begin{bmatrix} x_1+2 \\ x_2-1 \end{bmatrix} = 1$



$\begin{bmatrix} x_1+2 & x_2-1 \end{bmatrix}\begin{bmatrix} \frac{3}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{9}{50} \end{bmatrix}\begin{bmatrix} x_1+2 \\ x_2-1 \end{bmatrix} = 1$

$\begin{bmatrix} \frac{3}{25}x_1 + \frac{2}{25} + \frac{1}{25}x_2 - \frac{1}{25} & \frac{1}{25}x_1 + \frac{2}{25} + \frac{9}{50}x_2 - \frac{9}{50} \end{bmatrix}\begin{bmatrix} x_1+2 \\ x_2-1 \end{bmatrix} = 1$

$\frac{3}{25}x_1^2 + \frac{2}{25}x_1x_2 + \frac{2}{5}x_1 + \frac{9}{50}x_2^2 - \frac{1}{5}x_2 = \frac{1}{2}$ (to verify solution)

$P-\lambda I = \begin{bmatrix} 9-\lambda & -2 \\ -2 & 6-\lambda \end{bmatrix} \Rightarrow 0 = (9-\lambda)(6-\lambda) - 4$

$= \lambda^2 - 15\lambda + 50$

$= (\lambda-10)(\lambda-5)$

$\lambda = 10$: $\begin{bmatrix} 9 & -2 \\ -2 & 6 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 10\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$9x_1 - 2x_2 = 10x_1$

$-2x_1 + 6x_2 = 10x_2$

$\therefore -x_1 - 2x_2 = 0$

$-2x_1 - 4x_2 = 0$

$\therefore x_1 = -2x_2$

$v_1 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$ Direction of axis with length $\sqrt{10}$

$\lambda = 5$: $\begin{bmatrix} 9 & -2 \\ -2 & 6 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 5\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$9x_1 - 2x_2 = 5x_1$

$-2x_1 + 6x_2 = 5x_2$

$\therefore 4x_1 - 2x_2 = 0$

$-2x_1 + x_2 = 0$

$\therefore x_2 = 2x_1$

$v_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ Direction of axis with length $\sqrt{5}$

d) If A has an eigenvalue of 0, then there exists an $x_0 \in C$ s.t. $Ax_0 = 0$. The definition of $P = \{Ax \mid x \in C\}$ which means that $P = Ax_0 = 0$. This implies that the volume of the parallelpiped is $0 \Rightarrow \det(A) = 0$.

Problem 3.10

a) $\tilde{M} = U\Sigma V^T$ where $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{n \times m}$

By expanding the SVD, we have the following:

$\tilde{M} = \sigma_1 u^{(1)} v^{(1)T} + \sigma_2 u^{(2)} v^{(2)T} + \ldots \sigma_r u^{(r)} v^{(r)T}$ where r is the rank of $\tilde{M}$

The matrix $M^T M \in \mathbb{R}^{m \times m}$ can be used to show the words that document i and j have in common by navigating to the element in the i^{th} row and j^{th} column. Based on SVD properties, we also know that the eigenvectors of $M^T M$ are the rows of V^T. The first r rows of V^T provide a basis of M's row space based on the fact that only the first r rows of V^T are kept in the SVD of $\tilde{M}$.

∴ the first r rows of V^T can describe any document by means of linear combinations

The matrix $M^T M \in \mathbb{R}^{n \times n}$ has elements that represent the number of documents that letters i and j can both be found in. Similarly to above, the eigenvectors of $M^T M$ are the columns of U. Again, similar to above, the first

r columns of U provide a basis for M.
$\therefore$ the first r columns of U can describe any document by means of linear combination

b) $\vec{d_i}$: vector representation of a document
$\vec{q}$: vector representation of some query

$$\text{proj}_{\mathcal{U}_k}\vec{q} = \sum_{i=1}^{k} \frac{\langle \vec{q}, \vec{u_i} \rangle}{\|\vec{u_i}\|} \vec{u_i} \qquad \text{proj}_{\mathcal{U}_k}\vec{d} = \sum_{i=1}^{k} \frac{\langle \vec{d}, \vec{u_i} \rangle}{\|\vec{u_i}\|} \vec{u_i}$$

Given any d: $\cos\theta = \frac{\langle \vec{d}, \vec{q} \rangle}{\|\vec{d}\| \|\vec{q}\|}$, where θ represents the angle between $\vec{q}$ and $\vec{d}$

$$\therefore \cos\theta = \frac{\left\langle \sum_{i=1}^{k} \frac{\langle \vec{d}, \vec{u_i} \rangle}{\|\vec{u_i}\|^2} \vec{u_i}, \sum_{i=1}^{k} \frac{\langle \vec{q}, \vec{u_i} \rangle}{\|\vec{u_i}\|^2} \vec{u_i} \right\rangle}{\|\vec{d}\| \, \|\vec{q}\|}$$

Problem 3.10

Python Code

```
import numpy as np
import scipy.io
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import euclidean_distances, cosine_similarity

v = scipy.io.loadmat('/Users/armaanlalani/Documents/Engineering Science Year 3/ECE367
- Matrix Algebra and Optimization/PS3/wordVecV.mat')['V']

M = np.where(v>0, v, 0)
M_norm = M / np.linalg.norm(M, axis=0)

u, s, v = np.linalg.svd(M_norm)
print(s)

def similar(k):
    u_1 = []
    for i in range(k):
        u_1.append(u[:,i])
    u_1 = np.reshape(u_1, (1651,k))
    d = []
    for i in range(M_norm.shape[1]):
        dj = M_norm[:,i]
        on = np.zeros(1651)
        for j in range(u_1.shape[1]):
            u_2 = u_1[:,j]
            on += u_2 * np.dot(dj,u_2) / np.dot(u_2,u_2)
        d.append(on)

    dist = cosine_similarity(d,d)
    dist = np.where(dist<1-0.01, dist, 0)
    i, j = np.unravel_index(dist.argmax(), dist.shape)

    print('article %d and %d' %(i,j))

for i in range(10):
    print('Using k as %d' %(10-i))
    similar(10-i)
```

Question c

[2.67345552 0.72519713 0.71010595 0.61116441 0.56041118 0.52240735
 0.50401773 0.49789875 0.48154797 0.35798978]

Question d

The two most similar documents when using k as 9 are documents 8 and 9 which are titled Barack Obama and George W. Bush.

Question e

Using k as 9
article 8 and 9
Using k as 8
article 0 and 1
Using k as 7
article 8 and 9
Using k as 6
article 5 and 8
Using k as 5
article 0 and 5
Using k as 4
article 1 and 3
Using k as 3
article 1 and 4
Using k as 2
article 7 and 9
Using k as 1
article 0 and 0

The lowest k that does not change my answer for part d is when k is 7. The pair of most similar documents for k-1 (6) is documents 5 and 8 which are titled John Holland and Barack Obama.

Problem 3.11

Python Code

```
import numpy as np
from numpy import linalg as la
import matplotlib.pyplot as plt
from PIL import Image

M = np.loadtxt('data.csv', delimiter=',') # reshaped data in csv file from matlab

for i in range(M.shape[1]):
    im = np.reshape(M[:,i], (32,32))
    im = np.transpose(im)
    im = np.reshape(im, (-1, 1))
    M[:,i] = im[:,0]
```

```
image = Image.fromarray(np.reshape(M[:,200], (32,32))) # printing a random face to
verify correct matrix adjustment
# image.show()

avg_face = np.zeros((M.shape[0],1)) # vector to store the average of the image vectors
for i in range(M.shape[1]):
    avg_face[:,0] = np.add(avg_face[:,0], M[:,i])
avg_face = avg_face / M.shape[1] # divide average by the number of images

X = np.zeros((M.shape[0], M.shape[1])) # matrix to hold the centered vectors
for i in range(M.shape[1]):
    m = np.expand_dims(M[:,i], axis=1)
    X[:,i] = np.squeeze(m - avg_face) # subtracts the average of the image vectors
from each image vector

im1 = X[:,200] + np.squeeze(avg_face)
im1 = np.expand_dims(im1, axis = 1)
image1 = Image.fromarray(np.reshape(im1, (32,32)))
# image1.show() # verifies that the average was taken correctly

C = np.matmul(X, np.transpose(X)) # C is of shape 1024 by 1024

e_values_ns, e_vectors_ns = la.eig(C)
e_values = np.sort(e_values_ns)[::-1]

e_vectors = np.zeros((e_vectors_ns.shape[0], e_vectors_ns.shape[1]))

order = []
for i in range(e_values.shape[0]):
    val = e_values[i]
    for j in range(e_values.shape[0]):
        if val == e_values_ns[j]:
            order.append(j)
            break

for i in range(len(order)):
    e_vectors[i] = e_vectors_ns[order[i]]

e_values_log = np.log(e_values)

plt.plot(e_values_log)
plt.show()

def convert(data):
    old_min, old_max = np.amin(data), np.amax(data)
    reshape = (((data - old_min) * 255) / (old_max - old_min))
    return reshape
```

```
#for i in range(10):
#    reshape = np.reshape(e_vectors_ns[:,i], (32,32))
#    reshape = convert(reshape)
#    image = Image.fromarray(reshape)
#    image.show()

#for i in range(-10,-1,1):
#    reshape = np.reshape(e_vectors[:,i], (32,32))
#    reshape = convert(reshape)
#    image = Image.fromarray(reshape)
#    image.show()

images = [1,1076,2043]
b_j = [2, 2**2, 2**3, 2**4, 2**5, 2**6, 2**7, 2**8, 2**9, 2**10]

plt.figure(figsize=(20,8))

for i in range (len(images)):
    for j in range(10):
        subspace = e_vectors[:,:b_j[j]]
        proj = np.zeros((1,1024))
        for k in range(subspace.shape[1]):
            proj = proj + np.dot(X[:,images[i]], subspace[:,k]) * subspace[:,k]
        proj += avg_face[:,0]
        proj = np.reshape(proj, (32,32))
        reshape = convert(proj)
        image = Image.fromarray(reshape)
        plt.subplot(6,5,i*10+j+1)
        plt.imshow(image)

# plt.show()

images = [1,2,7,2043,2044,2045]
c = np.zeros((25,6))
for i in range(len(images)):
    subspace = e_vectors[:,:25]
    for k in range(subspace.shape[1]):
        c[k,i] = np.dot(X[:,images[i]], subspace[:,k])

print(c.shape)

for i in range(6):
    for j in range(i+1,6):
        norm = np.linalg.norm(c[:,i]-c[:,j])
        print('Distance between c%d and c%d is %.2f' %(images[i],images[j],norm))
```

Question a

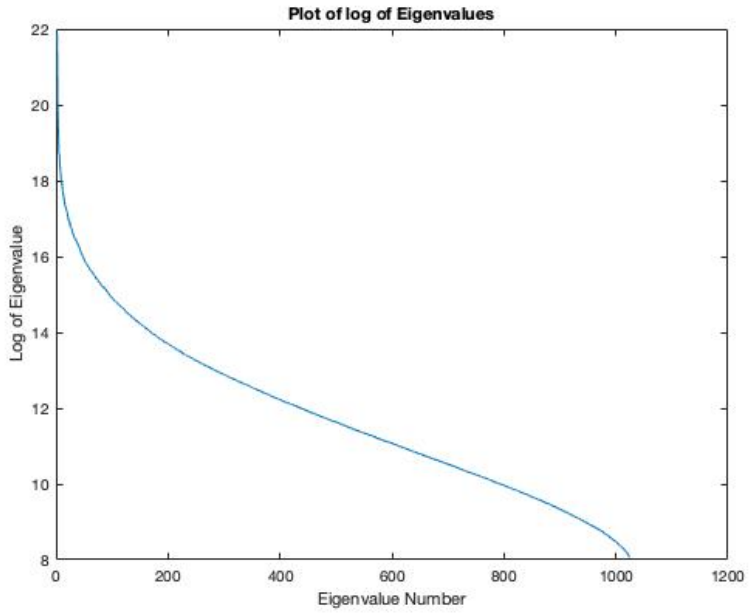
The singular values of X are equivalent to the square root of the eigenvalues of C multiplied by C^T, by definition of singular values. The right singular vectors of X are eigenvectors of C^T while the left singular vectors of X are the eigenvectors of C.

C in general is n times the covariance matrix of $x^{(i)}$, which means that for singular values of X and eigenvalues of C:

$$\lambda_k = \frac{{\sigma_k}^2}{N}$$

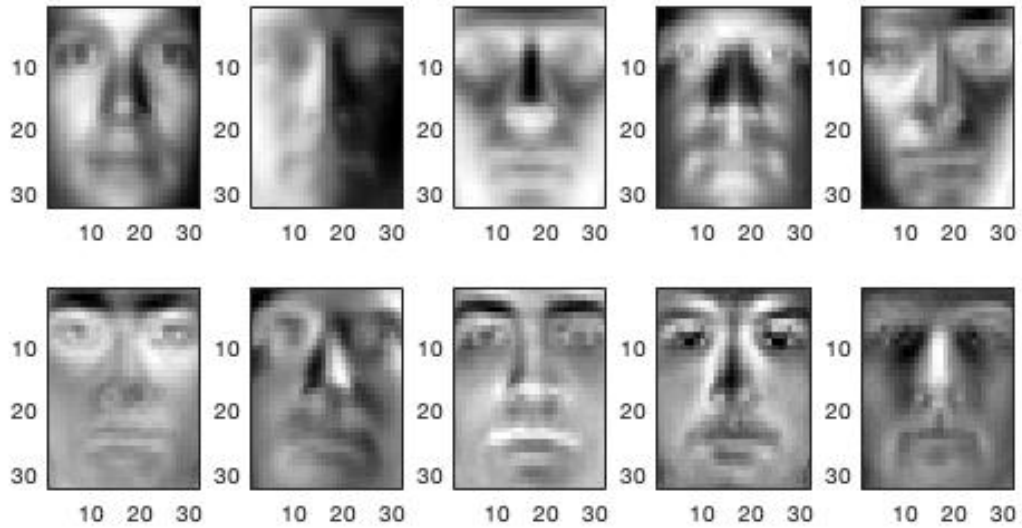
We also know that the eigenvectors of XX^T make the columns of U which are representative of the left singular values.

Question b

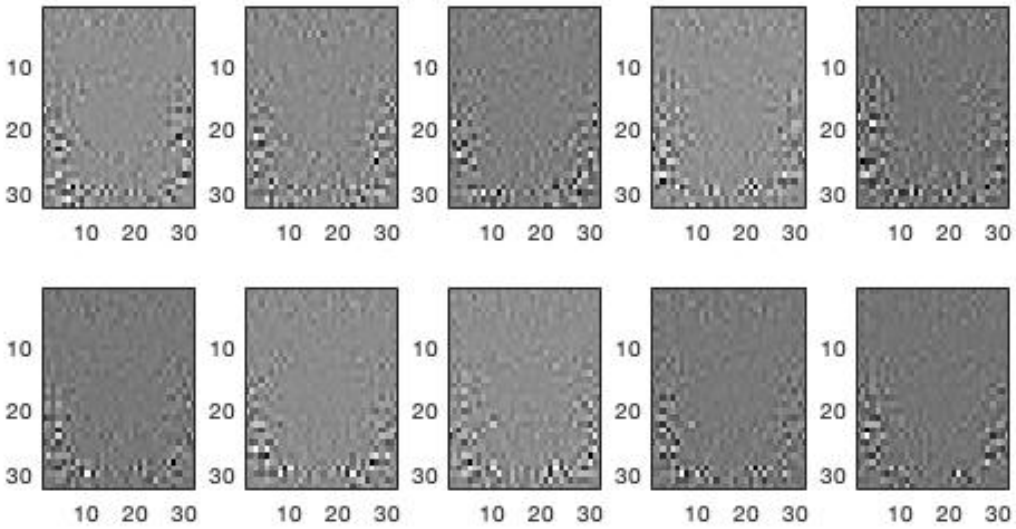


All of the eigenvalues of C are real. The matrix C in this case is symmetric which is why all of the eigenvalues are real valued.

Question c



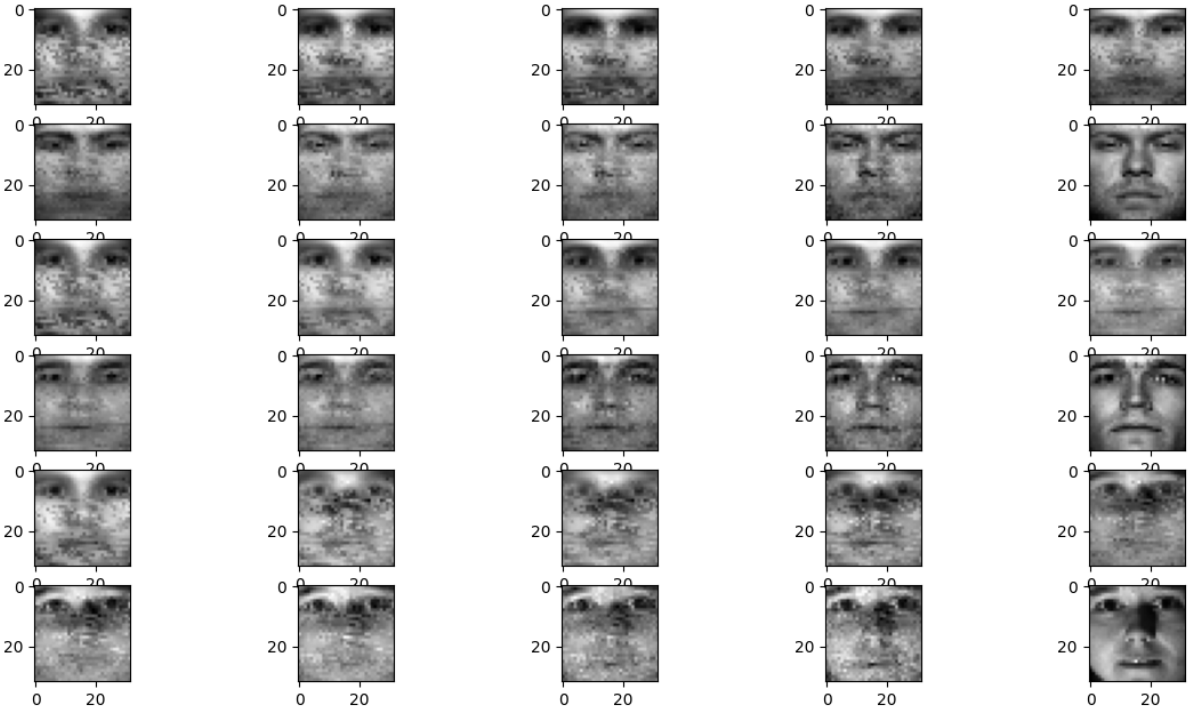
Eigenfaces of largest 10 eigenvalues



Eigenfaces of smallest 10 eigenvalues

As shown from the above two images, the eigenfaces of the largest 10 eigenvalues compared to the eigenfaces of the smallest 10 eigenvalues certainly differ. This is because we are trying to find the components of the matrix C that represent the maximum variance. Higher eigenvalues correspond to components that represent more variance in the data, which therefore does a better job of representing the dataset.

Question d



Question e

Distance between c1 and c2 is 657.44
Distance between c1 and c7 is 696.53
Distance between c1 and c2043 is 1577.40
Distance between c1 and c2044 is 1353.47
Distance between c1 and c2045 is 1208.40
Distance between c2 and c7 is 1275.17
Distance between c2 and c2043 is 1846.90
Distance between c2 and c2044 is 1455.72
Distance between c2 and c2045 is 1296.74
Distance between c7 and c2043 is 1159.51
Distance between c7 and c2044 is 1169.03
Distance between c7 and c2045 is 1115.94
Distance between c2043 and c2044 is 684.63
Distance between c2043 and c2045 is 1097.25
Distance between c2044 and c2045 is 618.22

As shown in the above comparisons, it is evident that the distances between images 1, 2, and 7 are closer to each other than images 2043, 2044, and 2045 and vice versa. This could be expanded to a simply face recognition program as follows:

- Given a particular dataset of images such as the one in this question, apply all the same steps in order to find the coefficients by projecting $x^{(i)}$ onto the eigenvectors in B_j, where j should be large enough to represent a significant amount of variance in the data
- Take in some input image that needs to be recognized
- Determine the coefficients of projecting the new image onto the eigenvectors in B_j
- Compare the Euclidian distance of the coefficients of the new image to the whole set of training data
- Whichever eigenface the input distance is closest to represents the most probable facial match