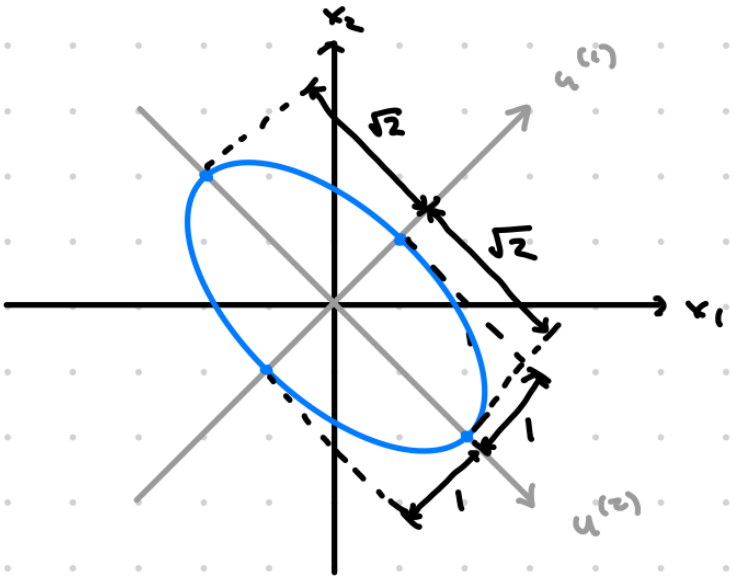
ECE454 Problem Set 3
Benjamin Cheng – 1004838045

Collaboration with Katherine Liang - 1004814542

Q3.9

a)



$$P = \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} \end{bmatrix}$$

$$\det \begin{bmatrix} \frac{3}{2}-\lambda & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2}-\lambda \end{bmatrix} = \left(\frac{3}{2}-\lambda\right)^2 - \frac{1}{4} = 0$$

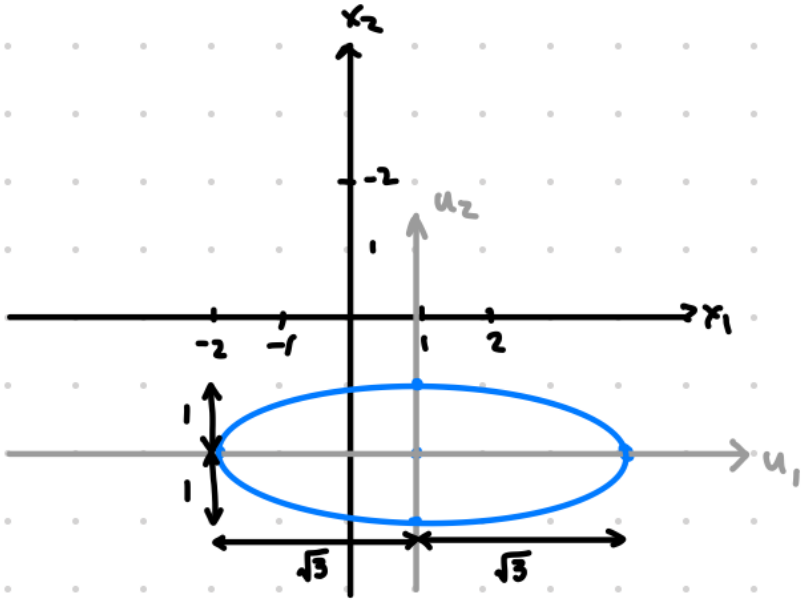
$$\Leftrightarrow \quad \frac{3}{2} - \lambda = \pm\frac{1}{2}$$

$$\lambda_1 = 1, \quad \lambda_2 = 2$$

$\lambda_1 = 1$: $N\left(\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}\right\}$ u_1

$\lambda_2 = 2$: $N\left(\begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}\right\}$ u_2

b)



$$P = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

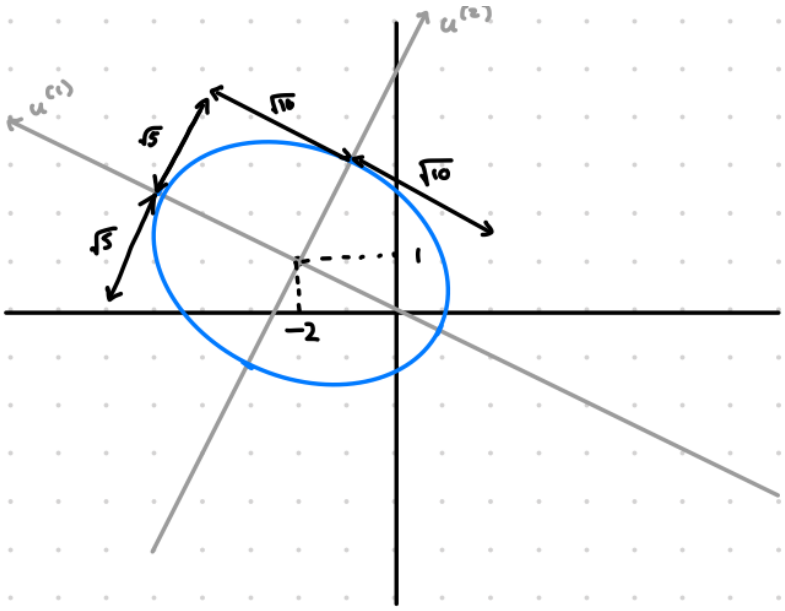
$\lambda_1 = 3$:

$$N\left(\begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}\right) = \operatorname{span}\left\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} \quad u_1$$

$\lambda_2 = 1$:

$$N\left(\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}\right) = \operatorname{span}\left\{\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\} \quad u_2$$

c)



$$P = \begin{bmatrix} 9 & -2 \\ -2 & 6 \end{bmatrix}$$

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

$$54 - 15\lambda + \lambda^2 - 4 = 0$$

$$50 - 15\lambda + \lambda^2 = 0$$

$$(\lambda - 10)(\lambda - 5) = 0$$

$$\lambda_1 = 10, \quad \lambda_2 = 5$$

$$\lambda_1 = 10: \quad N\left(\begin{bmatrix} -1 & -2 \\ -2 & -4 \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} -2 \\ 1 \end{bmatrix}\right\} = \text{span}\left\{\begin{bmatrix} -\frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}\right\} \quad u^{(1)}$$

$$\lambda_2 = 5: \quad N\left(\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right\} = \text{span}\left\{\begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}\right\} \quad u^{(2)}$$

d) The eigenvectors are “invariant directions” for the transformation represented by the matrix. If one of the eigenvalues is zero this means that its corresponding eigenvector reduces down to 0. The unit cube U in n-space is n-dimensional because its contains vectors in all directions in n-space. Now if k eigenvalues are zero, k of these directions will be reduced to zero, making the resulting paralellpiped P have n-k dimensions. The volume of an object with less than n-dimensions in n-space is zero since it is “flat”. With our interpretation of the determinant of A as the volume of the transformed unit cube, this means that the determinant of A is zero if it has a zero eigenvalue.

Q3.10

a) Viewing the SVD as a way to perform PCA, we can say that u_i can be interpreted as the top r "meanings" in the set of documents. When projecting a specific document onto this set of vectors this components of the projection show which meanings are present in this document or query. v_i then indicates the amount of these meanings on each document in the set.

b) The vectors $u_1 \dots u_k$ are orthonormal by definition of U in the SVD. If we let α be the coordinates of the projection of a vector $p \in \mathbb{R}^n$ onto $\text{span}\{u_1 \dots u_k\}$, we get the following:

$$I_n \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \langle v, u_1 \rangle \\ \vdots \\ \langle v, u_n \rangle \end{bmatrix} = \begin{bmatrix} u_1^\intercal \\ \vdots \\ u_k^\intercal \end{bmatrix} v$$

Writing out the full projection v^* in matrix form:

$$v^* = \alpha_1 u_1 + \cdots \alpha_n u_n = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = U_k U_k^\intercal v = Cv$$

with $U_k = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}, C = U_k U_k^\intercal$.

Now we can project a query q and a document d_i in this manner, giving the cosine similarity:

$$\frac{\langle d_i^*, q^* \rangle}{\|d_i^*\|_2 \|q^*\|_2} = \frac{\langle Cd_i, Cq \rangle}{\|Cd_i\|_2 \|Cq\|_2}$$

c) The 10 largest singular values were:

1. 1.5366
2. 1.0192
3. 0.9587
4. 0.9539
5. 0.9413
6. 0.9289
7. 0.8977
8. 0.8919
9. 0.8687
10. 0.8161

Source (MATLAB):

```
%% Part c)
M = double(V > 0);
M_norm = M ./ vecnorm(M);

[U,S,V] = svd(M_norm);
s = diag(S);
s(1:10)
```

d) The closest two documents were documents 9 and 10. The titles of these are Barack Obama and George W. Bush.

Source (MATLAB):

```
%% Part d)

m = size(M, 2);
d = zeros(m, m);
C = U(:,1:9) * U(:,1:9)';
for i = 1:m
    for j = 1:m
        if i >= j
            d(i,j) = 0;
        else
            di = C * M_norm(:,i);
            dj = C * M_norm(:,j);
            d(i,j) = di' * dj ./ (norm(C * di) * norm(C * dj));
        end
    end
end

[dists, inds] = maxk(d(:), 1);
[r, c] = ind2sub(size(d), inds)
```

e) The lowest value was k=3. K=2 gives documents 1 and 6 as the closest pair.

Source (MATLAB):

```
for k = 1:8
    m = size(M, 2);
    d = zeros(m, m);
    C = U(:,1:k) * U(:,1:k)';
    for i = 1:m
        for j = 1:m
            if i >= j
                d(i,j) = 0;
            else
                di = C * M_norm(:,i);
                dj = C * M_norm(:,j);
                d(i,j) = di' * dj ./ (norm(C * di) * norm(C * dj));
            end
        end
    end

    [dists, inds] = maxk(d(:), 1);
    [r, c] = ind2sub(size(d), inds);
    fprintf("k=%d, d(%d,%d)\n", k, r, c);
end
```

Q3.11

a) A singular value decomposition exists for any matrix. In particular we have that $X = U\Sigma V^\intercal$, i.e.

$$\begin{aligned} C &= XX^\intercal \\ &= U\Sigma V^\intercal (U\Sigma V^\intercal)^\intercal \\ &= U\Sigma V^\intercal V \Sigma^\intercal U^\intercal \\ &= U\Sigma I \Sigma^\intercal U^\intercal && \text{(Since } V \text{ is unitary)} \\ &= U\Sigma\Sigma^\intercal U^\intercal \end{aligned}$$

with $U \in \mathbb{R}^{d\times d}$, $\Sigma \in \mathbb{R}^{d\times N}$, $V \in \mathbb{R}^{N\times N}$. Note that these are real matricies since X is real.

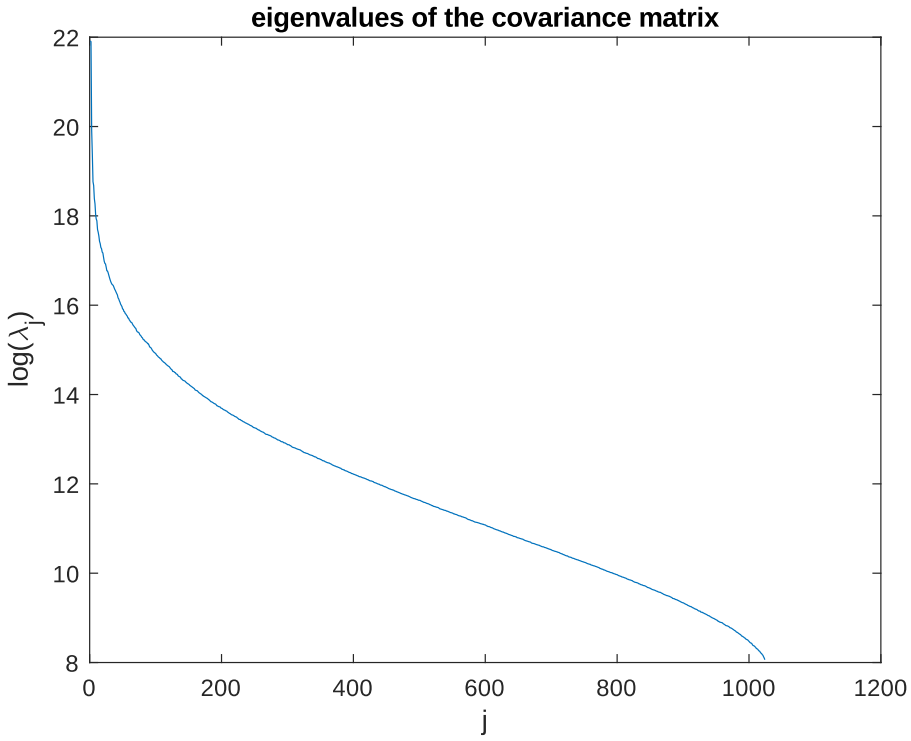
We can also do an eigendecomposition of C, noting that $Q^{-1} = Q^\intercal$ since C satifies the conditions of the spectral theorem. Equating it to the SVD from above yields

$$C = Q\Lambda Q^\intercal = U\Sigma\Sigma^\intercal U^\intercal$$

with $Q, \Lambda \in \mathbb{R}^{d\times d}$.

This representation suggests that $Q = U$ and $\Lambda = \Sigma\Sigma^\intercal$. Dimensions and properties agree in both since Q and U are both unitary, and $\Sigma\Sigma^\intercal$ is diagonal. The former expression indicates that the left-singular vectors of X are identical to the eigenvectors of C. The latter expression indicates that the eigenvalues of C are the square of the singular values of X.

b) The eigenvalues of C are real since $C \in S^d$, i.e. C is symmetric, by the spectral theorem. The logarithmic plot of the eigenvalues are shown in the plot.



Source (Matlab):

```
%% Data loading and SVD

load('data/yalefaces.mat')

X_orig = reshape(M, 1024, 2414);
X_mean = mean(X_orig, 2);
X = X_orig - X_mean;

C = X * X';
[U,S,V] = svd(C);
```

```
%% Part b
% Plot eigenvalues
figure
plot(log(diag(S)))
ylabel('log(\lambda_j)')
xlabel('j')
title('eigenvalues of the covariance matrix')

eigenfaces = reshape(U, 32, 32, 1024);
```

c) The eigenfaces are shown below:

Top 10 Eigenfaces

Bottom 10 Eigenfaces

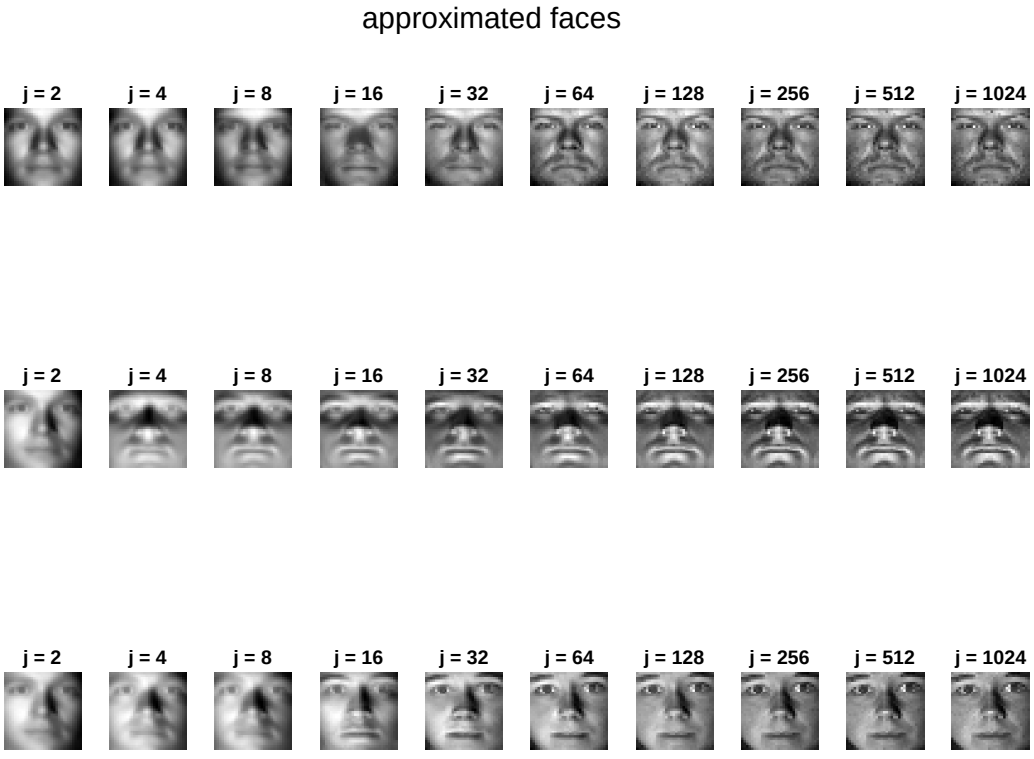
In the top 10 eigenfaces, the outline of a face can be seen, with certain features distinguishable. However in the bottom 10, the eigenfaces look like noise and not like a face at all. The large eigenvalues are associated with faces that form the most prominant features, while the small eigenvalues are associated with the remenants, which happen to look like noise.

Source (MATLAB):

```
%% Part c
% Top 10
figure
for i = 1:10
    subplot(2,5,i)
    imshow(eigenfaces(:,:,i), [])
end
sgtitle('Top 10 Eigenfaces')

% Bottom 10
figure
for i = 1:10
    subplot(2,5,i)
    imshow(eigenfaces(:,:,1025-i), [])
end
sgtitle('Bottom 10 Eigenfaces')
```

d) The approximated faces using the first j-th eigenfaces is shown in the image below. The first row corresponds to image 1, the second row corresponds to image 1076, and the third row corresponds to image 2043.



Source (MATLAB):

```
%% Part d
% Projection
is = [1,1076,2043];
js = 2.^(1:10);
Y = zeros(size(X,1), length(is), length(js));
for ii = 1:length(is)
    for jj = 1:length(js)
        i = is(ii);
        j = js(jj);
        B = U(:,1:j);
        Y(:,ii,jj) = X(:,i)' * B * B';
    end
end
  Y = reshape(Y, 1024, 3, 10);
  Y_orig = Y + X_mean;
  Y_disp = reshape(Y, 32, 32, 3, 10);

  figure
  for ii = 1:length(is)
      for jj = 1:length(js)
          i = (ii - 1) * 10 + jj;
          subplot(3, 10, i)
          imshow(Y_disp(:,:,ii,jj), [])
          title(sprintf('j = %d', js(jj)))
      end
  end
  sgtitle('approximated faces')
```

e)
The Euclidan distances are tabulated below. The values are written with a scaling factor of 10^{-3}.

	1	2	7	2043	2044	2045
1	0	0.593	0.472	1.08	1.53	1.37
2	0.593	0	0.392	1.40	1.90	1.61
7	0.472	0.392	0	1.26	1.79	1.58
2043	1.08	1.40	1.26	0	0.759	0.681
2044	1.53	1.90	1.79	0.759	0	0.828
2045	1.37	1.61	1.58	0.681	0.828	0

Very clearly, images 1-3 are very similar to each other while further from images 4-6. Images 4-6 are also much closer to each other than to images 1-3. To build a facial recognition scheme we could simply generate eigenfaces for a specific person, basically taking the place of X in this problem. Then a query would be projected onto the left singular vectors, or a subset (to compress the model). The error of this projection would be thresholded to determine whether it is a match or not. A high error corresponds with a higher likelihold of being a different face. Vice versa, a low error corresponds with a more similar face.

Source (MATLAB):

```
%% Part e
I = [1, 2, 7, 2043, 2044, 2045];
B25 = U(:,1:25);

C = zeros(size(B25,2), length(I));

for ii = 1:length(I)
    i = I(ii);
    C(:,ii) = X(:,i)' * B25;
end

diff = squareform(pdist(C'));
```