Name, Surname: Cansu YANIK

Student Number: 150170704

ITU Computer and Informatics Faculty

BLG 202E Numerical Methods in CE 2018-2019 Spring

Homework-3

1. Let A is a *m x n* (rectangular) matrix. Then there are orthogonal matrices *U*, *V* such that

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

Where,

$$\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}, \quad S = \mathrm{diag}(\sigma_1, \ldots\ldots, \sigma_r);$$

with the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, $\sigma_{r+1} = \cdots = \sigma_n = 0$.

Where the columns of U are the left singular vectors, S (the same dimensions as A) has singular values and is diagonal; and V^T has rows that are the right singular vectors.

$$A = (U_1 \quad U_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \tag{2}$$

We can decompose a square (*n x n*) matrix into its eigenvalues and eigenvectors. Let B is a square matrix.

$$B\,x = \lambda\,x \tag{3}$$

For some scalar λ. Then the scalar λ is called an eigenvalue of B, and x is said to be an eigenvector of B corresponding to λ.

$$B\,x - \lambda\, I\, x = 0 \qquad \text{(I is the identity matrix)}$$
$$\det(B - \lambda\, I) = 0 \tag{4}$$

For a diagonalizable n x n (square) matrix, B there are n eigenpairs (eigenvectors and eigenvalues). Then,

$$BX = X\Lambda \iff B = X\Lambda X^{-1}$$

with $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Because eigenvectors corresponding to different eigenvalues are orthogonal, it is possible to store all the eigenvectors in an orthogonal matrix. This implies the following equality:

$$X^{-1} = X^T$$

Therefore,

$$B = X\,\Lambda\,X^T \tag{5}$$

By using equation (5), we can find eigenvalues of B matrix. Λ is λ for B.

To be able to find eigenvalues of A matrix, we need a square matrix. We know that A^TA and AA^T are square matrices. Therefore, we can find the eigenvalues of A^TA or AA^T by using eigenvalue decomposition.

$A^TA = (V\,\Sigma^T\,U^T)\,U\,\Sigma\,V^T = V\,(\Sigma^T\,\Sigma)\,V^T$ $\quad\quad$ $(\Sigma^T\,\Sigma)$ is λ for A^TA

(Since U is an orthogonal matrix, there is an identity in U^TU)

$AA^T = U\,\Sigma\,V^T\,V\,\Sigma^T\,U = U\,(\Sigma^T\,\Sigma)\,U^T$ $\quad\quad$ $(\Sigma^T\,\Sigma)$ is λ for AA^T

(Since V is an orthogonal matrix, there is an identity in V^TV)

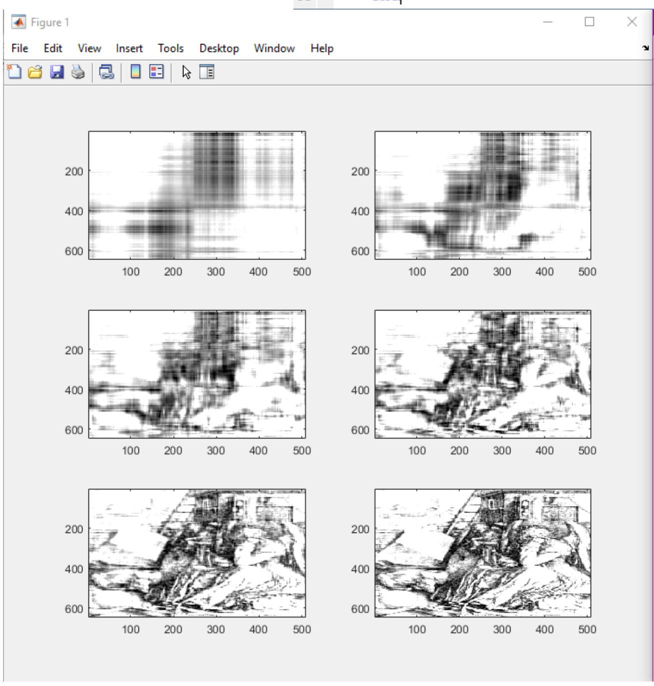
Both A^TA and AA^T has same eigenvalues because these matrices are symmetric and square. Also these matrices are positive matrices. Therefore,

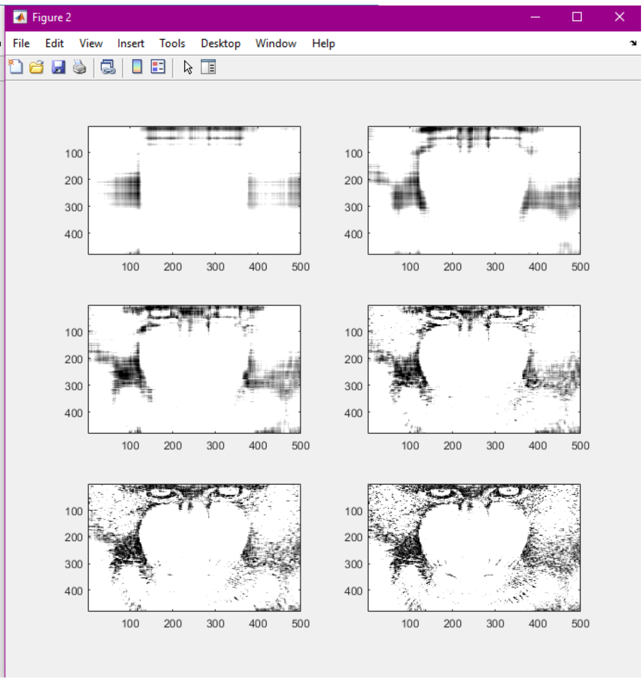
$(\Sigma^T\,\Sigma)$ is σ^2 for A. $\quad$ $\sigma i = \sqrt{\lambda i}$, $\quad$ (λi are eigenvalues of A^TA or AA^T.)

σi are eigenvalues of A.

2. a.

```
filename = 'durer';
myVars = {'X','caption'};
S(1) = load(filename,myVars{:});

filename = 'mandrill';
myVars = {'X','caption'};
S(2) = load(filename,myVars{:});

colormap(gray);
image(S(1).X);
image(S(2).X);

[U1,S1,V1]=svd(S(1).X);
[U2,S2,V2]=svd(S(2).X);

for i=1:6
    r = 2^i;
    % Stores the all singular values
    temp1 = S1;
    temp2 = S2;

    % Discards the not required diagonal values
    temp1(r+1:end,:)=0;
    temp1(:,r+1:end)=0;

    temp2(r+1:end,:)=0;
    temp2(:,r+1:end)=0;

    % Using the selected singular values, compute truncated SVD
    tsvd1 = U1*temp1*V1';
    tsvd2 = U2*temp2*V2';

    figure(1);
    subplot(3,2,i);
    colormap(gray);
    image(tsvd1);

    figure(2);
    subplot(3,2,i);
    colormap(gray);
    image(tsvd2);

end
```





b. As the number of rank increases, the resulting image becomes more similar to the original image, and truncated SVD starts to give better results. In other words, when we choose a higher rank number, we will get a closer approximation to the original image.

r	64
S	1x2 struct
S1	648x509 double
S2	480x500 double
temp1	648x509 double
temp2	480x500 double

For the original image "Durer", we need a double matrix with the 648 x 509 dimensions, and for the original image "Mandrill", we need another double matrix with the 480 x 500 dimensions.

Durer :

$X_{648 \text{ x } 509} = U_{648 \text{ x } 648}\ S_{648 \text{ x } 509}\ V^T_{509 \text{ x } 509}$

Mandrill:

$X_{480 \text{ x } 500} = U_{480 \text{ x } 480}\ S_{480 \text{ x } 500}\ V^T_{500 \text{ x } 500}$

As a function of r, a double matrix with r x r dimensions is required.

$X_{m \text{ x } n} = U_{m \text{ x } r}\ S_{r \text{ x } r}\ V^T_{r \text{ x } n}$

If the image is m x n and one uses r singular values, then one needs to store mr+nr+r=(m+n+1)r values to compress the image.

Singular value decomposition gives the closest rank k approximation of a matrix. But sometimes, choosing a smaller rank(r) will save more work and time. Also we can save space or reduce dimensionality by using truncated SVD.

3.

```
function [evector, eval] = PowerMethod(A, v0)

 x1 = v0';  %Takes the transpose of the initial guess vector

for i = 1:5

    y1 = A*x1;
    [max1, index] = max(abs(y1)); %Finds max entry
    max1 = max1 * sign(y1(index));
    x1 = y1 / max1; %For simplification

end
eval = max1;     %eigenvalue
evector = x1;    %eigenvector
display(eval);

%[V,D] = eig(A);
%V1 = -1*V(:,1) / max(V(:,1));
%V2 = V(:,3) / max(V(:,3));

%display(V);
%display(V1);
%display(V2);
%display(D);

end
```

- First 5 iterates for the first initial vector , $v_0 = (1,2,-1)^T$

```
>> A= [-2 1 4; 1 1 1; 4 1 -2];
>> v0 = [1 2 -1];
>> PowerMethod(A,v0)

max1 =

     8


x1 =

   -0.5000
    0.2500
    1.0000


max1 =

    5.2500


x1 =

    1.0000
    0.1429
   -0.7143
```

```
max1 =

    5.5714


x1 =

   -0.8462
    0.0769
    1.0000


max1 =

    5.7692


x1 =

    1.0000
    0.0400
   -0.9200
```

```
max1 =

    5.8800


x1 =

   -0.9592
    0.0204
    1.0000
```

After 5 iterates, eigenvalue is 5.8800 and eigenvector is $(-0.9592, 0.0204, 1.0000)^T$

- First 5 iterates for the second initial vector , $v_0 = (1,2,1)^T$

```
>> A= [-2 1 4; 1 1 1; 4 1 -2];
>> v0 = [1 2 1];
>> PowerMethod(A,v0)

max1 =

     4


x1 =

     1
     1
     1


max1 =

     3


x1 =

     1
     1
     1
```

```
max1 =

     3


x1 =

     1
     1
     1


max1 =

     3


x1 =

     1
     1
     1
```

```
max1 =

     3


x1 =

     1
     1
     1
```

After 5 iterates, eigenvalue is 3 and eigenvector is $(1, 1, 1)^T$.

Then I use MATLAB's eig(A) to examine the eigenvalues and eigenvectors of A.

```
>> A= [-2 1 4; 1 1 1; 4 1 -2];
>> v0 = [1 2 -1];
>> [V,D] = eig(A)

V =

    0.7071    0.4082   -0.5774
    0.0000   -0.8165   -0.5774
   -0.7071    0.4082   -0.5774


D =

   -6.0000         0         0
         0    0.0000         0
         0         0    3.0000
```

I have simplified the results to do better comparison. I divided the dominant vector by the largest value so that I get normalized vector. Eigenvectors are not unique and are accurate up to scale.

```
>> V(:,1) / max(V(:,1))

ans =

    1.0000
    0.0000
   -1.0000

>> -1*V(:,1) / max(V(:,1))

ans =

   -1.0000
   -0.0000
    1.0000
```

```
>> V(:,3) / max(V(:,3))

ans =

    1.0000
    1.0000
    1.0000
```

The convergence rate of the power method depends on $|\lambda_2 / \lambda_1|$. λ_1 is the largest eigenvalue, and λ_2 is the second largest eigenvalue of A. If this ratio is smaller than 1, the power method allows adequate convergence, but if this ratio is very close to 1, the power method will converge very slowly.

For the first case, the sequences converge at the fifth iteration, but for the second one, it will converge at the second iteration. Limits are not seeming to be the same because the initial guess vectors were selected differently.