Math 521 Homework 1

Experimenting with SVD and Random Matrices

Angel Silvar

Math 521
California State University - Long Beach
February 17, 2021

1 Problem 1

1.1 Change of Basis

Let the basis for $\mathscr{A}_1$ be $e^{(1)} = \begin{pmatrix}1\\0\end{pmatrix}$ and $e^{(2)} = \begin{pmatrix}0\\1\end{pmatrix}$. In other words,$\mathscr{A}_1$ is the standard basis on $\mathbb{R}^2$

Let the basis $\mathscr{A}_2$ be defined by $v^{(1)} = \begin{pmatrix}1\\1\end{pmatrix}$ and $v^{(2)} = \begin{pmatrix}-1\\1\end{pmatrix}$

We have $u_{\mathscr{A}_1} = \begin{pmatrix}1\\1\end{pmatrix}$. Find $u_{\mathscr{B}_2}$

Leverage the formula:

$$\mathscr{B}_1 u_{\mathscr{B}_1} = \mathscr{B}_2 u_{\mathscr{B}_2} \longrightarrow u_{\mathscr{B}_2} = \mathscr{B}_2^{-1}\mathscr{B}_1 u_{\mathscr{B}_1}$$

where ${\mathscr{B}_2}^{-1} = \frac{1}{det(u_{\mathscr{B}_2})} = \frac{1}{2}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix}$

Then, it follows:

$$u_{\mathscr{B}_2} = \frac{1}{2}\begin{pmatrix}1 & 1\\-1 & 1\end{pmatrix}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}$$

2 Problem 2

2.1 Random 2x2 Matrix and the Unit Circle

The following algorithm is to generate a $\beta \in \mathbb{R}^{1000x1}$. Then apply random matrices on this vector in order to to see how the output changes.

```
1
2  %This creates random values within the unit interval. It also creates
3  % a column vector of 1000 elements
4
5  beta = 2 * pi * rand(1000,1);
6
7  % Circle argument is meant to be matrix of 1000 by 2, where the 1st
8  % row is a transformation of Cosine while the 2nd is a transformation
9  % of Sine. This creates a matrix where each element represents a
10 % part of a unit circle.
11
12 circle = [cos(beta), sin(beta)];
13
14 % plot the values outputs Figure 1
15
16 scatter(circle(:,1) , circle(:,2))
17
18 % Create a A1 random matrix.
19
20 A1 = rand(2,2);
21
22 % Hit the circle with the random matrix to create a new matrix.
23 % This creates Figure 2
24
25 circle2 = circle * A1
26 pause(2)
27
28 % Repeat the process using a different A2 matrix
```

```
scatter(circle2(:,1) , circle(:,2))
A2 = rand(2,2);
circle3 = circle * A2
pause(2)
scatter(circle3(:,1),circle3(:,2))

% Orthogonalize A1 and A2


Q1 = orth ( A1 );
Q2 = orth ( A2 );
circle4 = circle * Q1 ;
circle5 = circle * Q2 ;
scatter ( circle4 (: ,1) , circle4 (: ,2))
scatter ( circle5 (: ,1) , circle4 (: ,2))
% The result is identical to figure 1
```

Here we know that the columns of the 1st circle matrix is an orthogonal matrix. Note that the values from the randomly generated values from a uniform distribution of given to the Cosine and Sine function. This converts these values to polar coordinates. Then, We create 2 more random 2 by 2 matrices and transform the original circle matrix by mapping its values to a new basis. Since the the matrices are not orthogonal, we have a shearing effect our of original circle. This means that the transformation stretches the original circle and changes its radius. This is shown in figure 2 and 3.

However, when we orthongonlize A1 and A2, and we transform our circle matrix, the circle does not change. The resulting matrix is identical to figure 1. Mathematically, this does not change since an orthogonal matrix will not change the value of a vector. i.e.

$$||Qx||_2^2 = (Qx)^T(Qx) = x^T Q^T Q x = x^T I x = ||x||_2^2$$

In other words, the Euclidean length of the radius does not change.

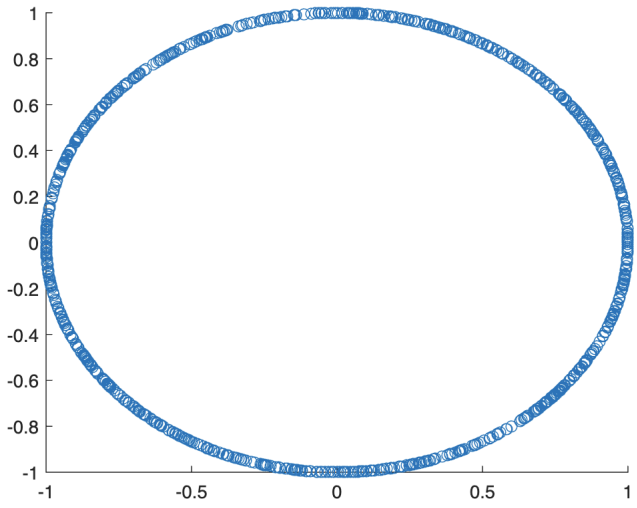

Figure 1 – A random array of points in a circle

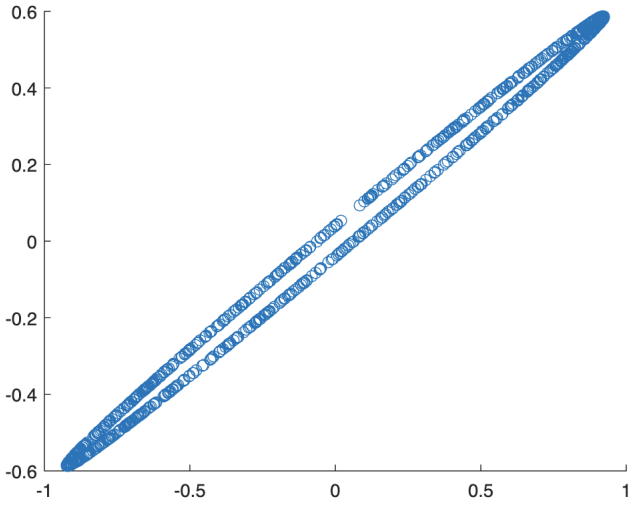

Figure 2 – A random array of points in a transformed circle.

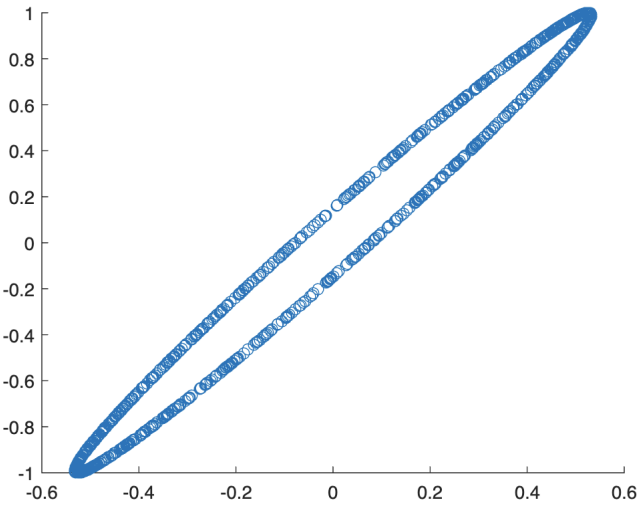

Figure 3 – A random array of points in a transformed circle.

2.2 SVD

a.

Point of the this problem is to compute principal angles between 2 sub spaces using any Real X and Y of size n x p and n x q respectively. In other words, X $\in \mathbb{R}^{nxp}$ and $Y \in \mathbb{R}^{nxq}$. Note the definition of Principal Angles: $\theta \in [0, \pi/2]$ and θ is listed in ascending order.

The tasks to complete in order to achieve the results that are the Principal Angles are:
1. Find orthonormal bases Q_x and Q_y for X and Y.
2. Compute SVD for cosine: $Q_x^{'} Q_y$
3. Compute matrix

$$Y = \begin{cases} \mathrm{Q}_y - Q_x[Q_x^{'} Q_y] & \text{if rank}(\mathrm{Q}_x) \geq rank(Q_y) \\ \mathrm{Q}_x - Q_y[Q_y^{'} Q_x] & \text{else} \end{cases}$$

4. Compute SVD Sine (this is to find Sine values to account for machine error) 5. Compute Principal Angles for k = 1 , ... , q

$$\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if } \sigma_k < 1/2 \\ \text{arcsine}(\mu_k) & \text{if } \mu_k \leq 1/2 \end{cases}$$

The following is a Matlab function created to take in any Real X and Y matrices.

```
% Function parameters are X1 and X2

function [theta] = prinAngles(X1,X2)

%Orthogonalize X1 and X2
Qx = orth(X1);
Qy = orth(X2);

% Calculate the Ranks
Rx = rank(Qx);
Ry = rank(Qy);

% Let M be the matrix for SVD
M = Qx' * Qy ;

% Get sigma values from M via SVD
Sx = svd(M);

% Calculate Y matrix for SVD
if rank(Qx) >= rank(Qy)
    y  =  Qy - Qx *(Qx'*Qy);
else
    y =   Qx - Qy *(Qy'* Qx);
end
% SVD on Y
sy = svd(y);

% Flip values since SVD produces our desired results in a different
%order. We flip since we want the largest principal angle
% for Sine first.
Sy = sort( sy );

% Use q to determine length of for loop and create a
%column vector of size q
q = min( size( Qx, 2), size(Qy,2));
theta = zeros(q,1);

% Follow if statement to assigned correct sigma or mu to cosine and
%sine
for  k = 1:q
    if  (Sx(k))^2 < .5
        theta(k) = acos(Sx(k));
    else
        theta(k) =asin(Sy(k));
    end
end
end
```

b.

This part is to check if the values are correct.

```
% Call in Datasets
load face1;
load face2;

%Apply function
theta = prinAngles(face1,face2)
theta1 = prinAngles(face1(:,1:10),face1(:,11:21))
imagesc(reshape(face1(:,1),160,138)), colormap(gray), axis off
```

By definition of Principal Angles, we have the following:

$$\cos\theta_k = \max_{u\in X}\max_{v\in Y}|u^T v| = |u_k^T v_k|$$

subject to $||u||_2 = ||v||_2 = 1$ and $u^T u_i = 0, v^T v_i = 0$ for i = 1 , 2 ,..., k-1

This tells us that the unit principal vectors, u and v, span into X and Y , respectively, in order to find the smallest angle compared to any other combination that yields larger angle values. The smallest angle for the i^{th} column of X is called the principal angle. Furthermore, the theory is that the (θ_k) yields a principal angle for the k^{th} column vector. Geometrically, this compares the i^{th} principal angle value how "close" or similar it is to the rest of the data set. The larger the value the closer it is.

In this case, since the smallest angle gives that largest cosine value. cos(0.3) has to be the largest singular value for our SVD. Moreover, since the elements in theta are increasing, our $\cos(\theta_k)$ $decreasesask\rightarrow$ 21 . This satisfies the theory of the singular values, where the first value is the largest element and anything there after is decreasing. This is a good sign that the Matlab function is correct.

The following table is the principal values that we get from comparing face 1 with another data set called "face2". Here we find that the angle between the first element is close to 17 degrees. As the principal angles increase, the images start to differ. This means that the images are distinct to begin with. See below for person 2.

	1
1	0.3026
2	0.4689
3	0.7318
4	1.0081
5	1.1159
6	1.2027
7	1.2276
8	1.2915
9	1.3709
10	1.3860
11	1.4227
12	1.4294
13	1.4682
14	1.4865
15	1.5117
16	1.5185
17	1.5286
18	1.5433
19	1.5480
20	1.5573
21	1.5690

Figure 4 – The angle values are starting off a bit away from 0. This means that these columns are not as similar to each other. i.e. the picture does not look similar. Since values become larger, then it implies that the pictures are becoming more different.

We test the Matlab function on a data set itself. The following is called "face1" where it is comprised of face images of 1 person. Then, we use the function to see how similar the pictures are towards one another. We tested the first 10 images to the last 10 images.

Images of face1:

Figure 5 – Person 1.

The rule of thumb is: the smaller the angle the larger the cosine value. Where the goal is to minimize the principal angles given the singular values found via SVD. Here, we see that our principal angle vector's 1st entry gives us the value of 0.0581 rad is 3 degrees. Meaning that the image is very similar to the sub data set that contains 11th - 21st images. This makes sense since it is the same person. However, we see as k→10 , that the angles are becoming much larger, this also makes sense since we see a change in lighting throughout the images of person 1. Hence, image 1 will be different from an image with different lighting.

	1
1	0.0581
2	0.1986
3	0.6144
4	0.7633
5	1.1839
6	1.3237
7	1.4040
8	1.4649
9	1.5064
10	1.5500

Figure 6 – The values are starting off close to zero. This means that these columns are very similar to each other.i.e. the picture looks very similar. Since values become larger, then it implies that the pictures are becoming more different.

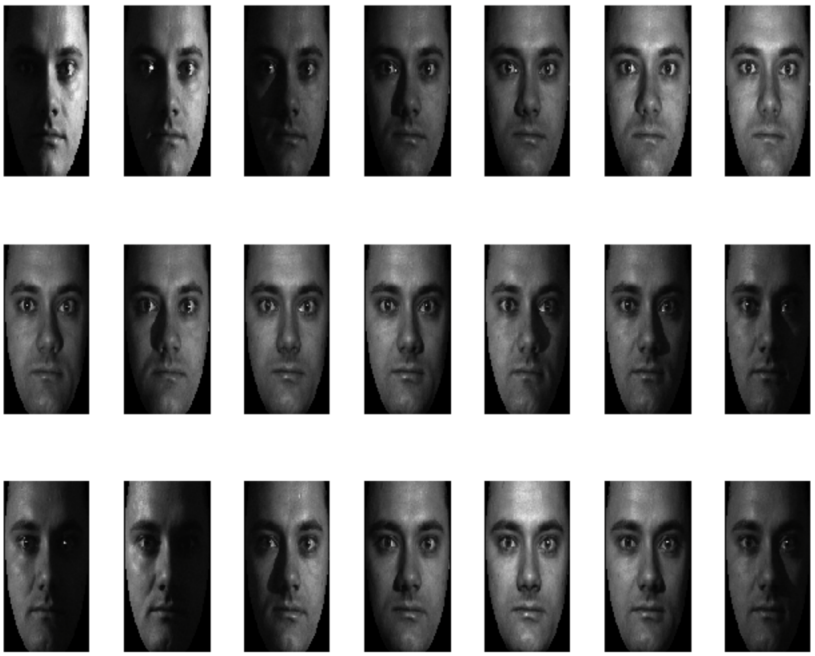

Figure 7 – Person 1.