Project Two Template

MAT-350: Applied Linear Algebra

Gabrielle Maitland

10/13/22

Problem 1

Use the svd() function in MATLAB to compute A_1, the **rank-1 approximation of** A**.** Clearly state what A_1 is, rounded to 4 decimal places. Also, **compute** the root-mean square error (RMSE) between A and A_1.

Solution:

```
%code
%Declare matrix
A = [1 2 2; 3 4 5; 6 7 8]
```

```
A = 3×3
     1     2     2
     3     4     5
     6     7     8
```

```
%compute rank-1 approximation
[U,S,V] = svd(A)
```

```
U = 3×3
   -0.2055   -0.6658   -0.7172
   -0.4900   -0.5644    0.6643
   -0.8471    0.4880   -0.2103
S = 3×3
   14.4042         0         0
         0    0.6450         0
         0         0    0.3229
V = 3×3
   -0.4692    0.8820    0.0433
   -0.5763   -0.2687   -0.7718
   -0.6691   -0.3871    0.6344
```

```
%making k equal to 1 to make formula easier for me to understand in alignment with SVD
%is the rank we are looking for
k = 1
```

```
k = 1
```

```
A1 = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';

%compute root-mean-square-error
RMSE = rms(sqrt(mean((A-A1).^2)))
```

```
RMSE = 0.2404
```

Enter your equation. The RMSE values are 0.3286, 0.1752, and 0.1865

Problem 2

Use the svd() function in MATLAB to compute A_2, the **rank-2 approximation of** A. Clearly state what A_2 is, rounded to 4 decimal places. Also, **compute** the root-mean square error (RMSE) between A and A_2. Which approximation is better, A_1 or A_2? Explain.

Solution:

```
%code
%Declare matrix
A = [1 2 2; 3 4 5; 6 7 8]
```

```
A = 3×3
     1     2     2
     3     4     5
     6     7     8
```

```
%compute rank-2 approximation
[U,S,V] = svd(A)
```

```
U = 3×3
   -0.2055   -0.6658   -0.7172
   -0.4900   -0.5644    0.6643
   -0.8471    0.4880   -0.2103
S = 3×3
   14.4042         0         0
         0    0.6450         0
         0         0    0.3229
V = 3×3
   -0.4692    0.8820    0.0433
   -0.5763   -0.2687   -0.7718
   -0.6691   -0.3871    0.6344
```

```
k = 2
```

```
k = 2
```

```
A2 = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';

%compute root-mean-square-error
RMSE = rms(sqrt(mean((A-A2).^2)))
```

```
RMSE = 0.1076
```

Explain: The A1 and A2 approximations are different with A2 being lower. Because A2 has a lower error value, I would say that A2 is better.

Problem 3

For the 3×3 matrix A, the singular value decomposition is $A = USV'$ where $U = [\mathbf{u}_1 \mathbf{u}_2 \mathbf{u}_3]$. Use MATLAB to **compute** the dot product $d_1 = dot(\mathbf{u}_1, \mathbf{u}_2)$.

Also, use MATLAB to **compute** the cross product $\mathbf{c} = cross(\mathbf{u}_1, \mathbf{u}_2)$ and dot product $d_2 = dot(\mathbf{c}, \mathbf{u}_3)$. Clearly state the values for each of these computations. Do these values make sense? **Explain**.

Solution:

```
%code
%declare matrix
A = [1 2 2; 3 4 5; 6 7 8]
```

```
A = 3×3
     1     2     2
     3     4     5
     6     7     8
```

```
%Complete singular value decomposition
[U,S,V] = svd(A)
```

```
U = 3×3
   -0.2055   -0.6658   -0.7172
   -0.4900   -0.5644    0.6643
   -0.8471    0.4880   -0.2103
S = 3×3
   14.4042         0         0
         0    0.6450         0
         0         0    0.3229
V = 3×3
   -0.4692    0.8820    0.0433
   -0.5763   -0.2687   -0.7718
   -0.6691   -0.3871    0.6344
```

```
u1 = U(:,1)
```

```
u1 = 3×1
   -0.2055
   -0.4900
   -0.8471
```

```
u2 = U(:,2)
```

```
u2 = 3×1
   -0.6658
   -0.5644
    0.4880
```

```
u3 = U(:,3)
```

```
u3 = 3×1
   -0.7172
    0.6643
   -0.2103
```

```
%Complete dot product d1
```

```
d1 = dot(u1,u2)
```

```
d1 = -2.4980e-16
```

```
%Complete cross product
c = cross(u1,u2)
```

```
c = 3×1
   -0.7172
    0.6643
   -0.2103
```

```
%Complete dot product d2
d2 = dot(c,u3)
```

```
d2 = 1
```

Explain: u1 and u2 multipled together equate to u3, and d2 is 1 because c and u3 equate to 1.

Problem 4

Using the matrix $U = [\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3]$, d**etermine whether or not the columns of U span $\mathbb{R}^3$. Explain your approach.**

Solution:

```
%code
%Declare matrix
U = [u1 u2 u3]
```

```
U = 3×3
   -0.2055   -0.6658   -0.7172
   -0.4900   -0.5644    0.6643
   -0.8471    0.4880   -0.2103
```

```
%Use rref to determine spanning
reducedU = rref(U)
```

```
reducedU = 3×3
     1     0     0
     0     1     0
     0     0     1
```

Explain: You can check to see if this matrix spans R3 by using rref because if it does, it will have 3 pivot columns

Problem 5

Use the MATLAB imshow() function to load and display the image A stored in the image.mat file, available in the Project Two Supported Materials area in Brightspace. For the loaded image, **derive the value of k** that will result in a compression ratio of $CR \approx 2$. For this value of k, **construct the rank-*k* approximation of the image**.

Solution:

```
%code
%Add the img file to environment
load('MAT 350 Project Two MATLAB Image.mat')
%Initialize image w figure command
figure;
%Show image
imshow(A)
```

```
%Found this command on matlab's help site to see how much space my image
%needs. It gives information about the specific file.
whos -file 'MAT 350 Project Two MATLAB Image.mat'
```

```
  Name         Size                Bytes  Class    Attributes

  A         2583x4220            10900260  uint8
```

```
%Find svd
[U,S,V] = svd(double(A))
```

```
U = 2583×2583
   -0.0106   -0.0360   -0.0006    0.0032   -0.0032    0.0041   -0.0066    0.0022 ···
   -0.0105   -0.0361   -0.0006    0.0030   -0.0035    0.0049   -0.0062    0.0020
   -0.0105   -0.0362   -0.0006    0.0034   -0.0037    0.0042   -0.0064    0.0025
   -0.0105   -0.0362   -0.0009    0.0029   -0.0035    0.0052   -0.0056    0.0028
   -0.0106   -0.0361   -0.0011    0.0034   -0.0035    0.0046   -0.0061    0.0022
   -0.0106   -0.0363   -0.0011    0.0031   -0.0030    0.0049   -0.0061    0.0031
   -0.0106   -0.0364   -0.0008    0.0032   -0.0032    0.0043   -0.0057    0.0033
   -0.0106   -0.0365   -0.0006    0.0029   -0.0033    0.0050   -0.0052    0.0031
   -0.0106   -0.0366   -0.0007    0.0033   -0.0031    0.0040   -0.0053    0.0033
```

```
   -0.0106   -0.0368   -0.0009    0.0030   -0.0034    0.0044   -0.0052    0.0032
     ⋮
S = 2583×4220
10^5 ×
    4.0600         0         0         0         0         0         0         0 ···
         0    0.8702         0         0         0         0         0         0
         0         0    0.4169         0         0         0         0         0
         0         0         0    0.4104         0         0         0         0
         0         0         0         0    0.3405         0         0         0
         0         0         0         0         0    0.2992         0         0
         0         0         0         0         0         0    0.2550         0
         0         0         0         0         0         0         0    0.2268
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
     ⋮
V = 4220×4220
   -0.0130    0.0044   -0.0358    0.0028   -0.0085    0.0177    0.0128   -0.0163 ···
   -0.0130    0.0045   -0.0357    0.0024   -0.0079    0.0184    0.0134   -0.0162
   -0.0130    0.0045   -0.0359    0.0025   -0.0078    0.0181    0.0124   -0.0168
   -0.0130    0.0046   -0.0361    0.0030   -0.0087    0.0185    0.0125   -0.0158
   -0.0130    0.0045   -0.0366    0.0032   -0.0095    0.0182    0.0116   -0.0149
   -0.0129    0.0046   -0.0369    0.0034   -0.0095    0.0185    0.0112   -0.0151
   -0.0130    0.0047   -0.0372    0.0046   -0.0104    0.0178    0.0103   -0.0141
   -0.0130    0.0048   -0.0377    0.0045   -0.0097    0.0179    0.0108   -0.0145
   -0.0130    0.0046   -0.0375    0.0049   -0.0094    0.0170    0.0102   -0.0147
   -0.0130    0.0045   -0.0379    0.0043   -0.0090    0.0166    0.0095   -0.0142
     ⋮
```

```
%Find value that makes Cr equal to 2.
%2 = (2583*4220)/(k*(2583+4220+1))
%k = 801
CR = (2583*4220)/(801*(2583+4220+1))
```

```
CR = 2.0000
```

```
A801 = U(:,1:801)*S(1:801, 1:801)*V(:,1:801)';
%The below formula ensures everything is in the correct data format.
A801 = uint8(round(A801))
```

```
A801 = 2583×4220 uint8 matrix
   26   27   31   29   23   26   35   29   33   36   30   26   32   27   28   34 ···
   33   34   28   31   28   28   36   31   30   27   22   28   40   34   34   34
   36   31   19   20   20   17   25   26   30   23   22   27   35   35   37   34
   34   30   26   30   27   16   25   26   28   25   22   24   33   35   32   30
   28   26   31   37   35   30   35   25   25   25   23   27   33   33   27   27
   27   26   27   29   26   29   34   23   22   27   29   34   33   33   28   26
   32   32   28   23   20   25   33   27   27   29   35   33   27   28   31   29
   26   32   27   18   21   28   32   26   29   32   36   31   28   31   34   34
   23   26   22   25   29   30   35   31   30   35   34   27   20   24   34   34
   21   21   19   26   27   25   30   30   28   34   29   20   16   22   29   32
     ⋮
```

Explain: I loaded my image into the environment and then used the formula given in the Word doc to help find K by substituting the values of my image. I found the values of my image by using the whos command to see the size dimensions. I then ensured that the data format was the acceptable approximation, which is uint8.

Problem 6

Display the image and compute the root mean square error (RMSE) between the approximation and the original image. Make sure to include a copy of the approximate image in your report.

Solution:

```
%code
%show image
imshow(A801)
```

```
%Find RMSE between A and A801
RMSE = rms(sqrt(mean((A-A801).^2)))
```

```
RMSE = 2.2389
```

Problem 7

Repeat Problems 5 and 6 for $CR \approx 10$, $CR \approx 25$, and $CR \approx 75$. **Explain** what trends you observe in the image approximation as CR increases and provide your recommendation for the best CR based on your observations. Make sure to include a copy of the approximate images in your report.

Solution:

```
%code
```

```
%Find value that makes Cr equal to 10.
%10 = (2583*4220)/(k*(2583+4220+1))
%k = 160
CR = (2583*4220)/(160*(2583+4220+1))
```

```
CR = 10.0127
```

```
A160 = U(:,1:160)*S(1:160, 1:160)*V(:,1:160)';
%The below formula ensures everything is in the correct data format.
A160 = uint8(round(A160))
```

```
A160 = 2583×4220 uint8 matrix
   29   29   30   29   28   26   27   25   28   27   26   25   28   27   27   28 ···
   29   29   30   28   28   26   28   26   28   27   26   25   28   27   28   29
   26   26   27   25   25   23   24   22   24   24   24   23   26   25   27   28
   28   29   30   27   28   26   27   25   26   26   26   24   28   27   27   28
   29   28   29   28   29   27   28   26   27   26   27   25   28   27   27   28
   27   27   27   26   27   25   27   25   27   26   26   24   26   26   27   27
   28   27   27   26   26   25   27   25   26   25   26   24   26   26   27   27
   27   26   27   26   27   26   28   25   27   25   25   23   25   24   25   25
   25   25   25   25   25   25   26   24   25   24   24   21   23   23   23   23
   24   23   23   23   23   23   24   22   23   21   21   19   21   21   22   23
    ⋮
```

```
%show image
imshow(A160)
```

```
%Find RMSE between A and A160
```

```
RMSE = rms(sqrt(mean((A-A160).^2)))
```

```
RMSE = 5.1769
```

```
%%%%%%%%%%
%Find value that makes Cr equal to 25.
%25 = (2583*4220)/(k*(2583+4220+1))
%k = 64
CR = (2583*4220)/(64*(2583+4220+1))
```

```
CR = 25.0318
```

```
A64 = U(:,1:64)*S(1:64, 1:64)*V(:,1:64)';
%The below formula ensures everything is in the correct data format.
A64 = uint8(round(A64))
```

```
A64 = 2583×4220 uint8 matrix
   25   24   22   22   22   22   23   24   25   26   27   27   29   29   28   30 ···
   25   24   22   22   22   23   24   24   25   26   27   27   30   29   29   31
   24   23   21   21   21   21   22   23   24   25   26   26   29   28   28   29
   24   24   22   22   22   22   23   24   25   26   27   27   29   28   28   29
   25   24   22   22   22   22   23   24   25   26   27   27   29   29   28   29
   24   23   21   22   22   22   23   24   24   25   26   26   28   28   27   28
   24   23   21   21   21   22   23   23   24   25   26   26   28   27   26   27
   25   24   22   23   23   23   24   24   25   26   27   27   29   28   27   28
   24   23   21   21   22   22   22   23   24   25   26   25   27   27   26   27
   22   22   19   20   20   20   21   22   23   24   25   25   27   26   26   26
    ⋮
```

```
%show image
imshow(A64)
```

```
%Find RMSE between A and A64
RMSE = rms(sqrt(mean((A-A64).^2)))
```

```
RMSE = 6.3933
```

```
%%%%%%%%%%
%Find value that makes Cr equal to 75.
%75 = (2583*4220)/(k*(2583+4220+1))
%k = 29
CR = (2583*4220)/(29*(2583+4220+1))
```

```
CR = 55.2427
```

```
A29 = U(:,1:29)*S(1:29, 1:29)*V(:,1:29)';
%The below formula ensures everything is in the correct data format.
A29 = uint8(round(A29))
```

```
A29 = 2583×4220 uint8 matrix
   33   33   32   32   32   32   34   33   35   35   36   35   36   36   35   36 ···
   33   32   32   31   32   32   33   33   34   35   36   35   36   36   36   36
   31   31   30   30   30   30   32   32   33   33   34   34   35   35   34   35
   33   33   32   31   32   32   33   33   34   35   36   35   36   36   35   36
   33   32   32   31   32   32   33   33   34   35   36   35   37   37   36   36
   34   34   33   32   33   33   35   34   35   36   37   36   37   37   36   37
   33   33   32   31   32   32   33   33   34   35   36   35   36   37   36   36
   34   34   33   33   34   34   35   34   36   36   37   37   38   38   37   37
   33   32   32   31   32   32   33   33   34   34   36   35   36   36   35   36
   34   33   33   32   33   33   34   34   35   35   37   36   37   37   36   37
```

⋮

```
%show image
imshow(A29)
```

```
%Find RMSE between A and A29
RMSE = rms(sqrt(mean((A-A29).^2)))
```

```
RMSE = 7.3210
```

Explain:

In repeating steps 5 and 6, the trend shows that the bluriness of the picture increased as the compression rate increased. The compression rate of 2 gave the clearest image, while the compression rate of 75 made it harder to decipher. Because of that, I would recommend a compression rate of 2 the most. Even the compression rate of 10 would be an acceptable alternative as well. The RMSE values also increased as well between the original image and the different compression rates, which tells me there is more error in generating the images with higher compression.