Project Two Template

MAT-350: Applied Linear Algebra

Danny Faught

8/16/2020

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:

```
A = [1 2 2; 3 4 5; 6 7 8]
```

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

```
%code
[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
```

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

```
A1 = 3×3
    1.3889    1.7059    1.9807
    3.3118    4.0678    4.7230
    5.7253    7.0322    8.1649
```

```
RMSE_A1 = (norm(A - A1, "fro")) / 9
```

```
RMSE_A1 = 0.0801
```

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
A2 = U(:,1:2)*S(1:2,1:2)*V(:,1:2).'
```

```
A2 = 3×3
    1.0100    1.8213    2.1469
    2.9907    4.1656    4.8639
    6.0029    6.9476    8.0431
```

```
RMSE_A2 = (norm(A - A2, "fro")) / 9
```

```
RMSE_A2 = 0.0359
```

Explain: Here, we see that the rank-2 approximation is better than the rank-1 approximation as the error calculated in the RMSE is less.

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
U1 = U(:, 1).'
```

```
U1 = 1×3
   -0.2055   -0.4900   -0.8471
```

```
U2 = U(:, 2).'
```

```
U2 = 1×3
   -0.6658   -0.5644    0.4880
```

```
U3 = U(:, 3).'
```

```
U3 = 1×3
   -0.7172    0.6643   -0.2103
```

```
d1 = dot(U1, U2)
```

```
d1 = -8.3267e-17
```

```
c = cross(U1,U2)
```

```
Index in position 1 is invalid. Array indices must be positive integers or logical values.
```

```
d2 = dot(c, U(:, 3))
```

Explain: No idea why this isn't working. :(

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
reducedU = rref(U)
```

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

Explain: U does span $\mathbb{R}^3$. This is known because the reduced echelon form of U has 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
figure;
imshow(A)
```

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

```
U = 3072×3072
   -0.0220    0.0337   -0.0276    0.0071   -0.0003    0.0114   -0.0108    0.0043 ···
   -0.0220    0.0335   -0.0273    0.0066   -0.0002    0.0106   -0.0112    0.0037
   -0.0220    0.0335   -0.0271    0.0062   -0.0003    0.0100   -0.0113    0.0029
   -0.0220    0.0333   -0.0271    0.0057   -0.0003    0.0094   -0.0110    0.0023
   -0.0219    0.0331   -0.0273    0.0053   -0.0003    0.0083   -0.0109    0.0020
   -0.0219    0.0329   -0.0274    0.0049   -0.0007    0.0066   -0.0107    0.0017
   -0.0219    0.0325   -0.0274    0.0041   -0.0012    0.0048   -0.0106    0.0012
   -0.0218    0.0322   -0.0277    0.0037   -0.0012    0.0027   -0.0104    0.0008
   -0.0218    0.0321   -0.0281    0.0028   -0.0020    0.0008   -0.0097    0.0003
   -0.0218    0.0319   -0.0281    0.0020   -0.0025   -0.0009   -0.0093   -0.0001
      ⋮
S = 3072×4608
10^5 ×
    5.7986         0         0         0         0         0         0         0 ···
         0    0.6755         0         0         0         0         0         0
         0         0    0.3657         0         0         0         0         0
         0         0         0    0.3129         0         0         0         0
         0         0         0         0    0.2842         0         0         0
         0         0         0         0         0    0.2423         0         0
         0         0         0         0         0         0    0.2325         0
         0         0         0         0         0         0         0    0.2217
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
      ⋮
V = 4608×4608
   -0.0159   -0.0085   -0.0079   -0.0083    0.0064   -0.0076    0.0061    0.0098 ···
```

```
-0.0159   -0.0087   -0.0081   -0.0084    0.0064   -0.0082    0.0064    0.0100
-0.0159   -0.0088   -0.0079   -0.0085    0.0063   -0.0078    0.0067    0.0104
-0.0160   -0.0090   -0.0081   -0.0087    0.0063   -0.0077    0.0069    0.0106
-0.0160   -0.0091   -0.0079   -0.0092    0.0058   -0.0077    0.0071    0.0104
-0.0160   -0.0092   -0.0082   -0.0091    0.0055   -0.0076    0.0076    0.0104
-0.0160   -0.0093   -0.0081   -0.0094    0.0054   -0.0078    0.0076    0.0106
-0.0160   -0.0094   -0.0083   -0.0094    0.0053   -0.0076    0.0080    0.0107
-0.0160   -0.0095   -0.0084   -0.0098    0.0053   -0.0075    0.0080    0.0108
-0.0160   -0.0097   -0.0085   -0.0099    0.0052   -0.0074    0.0081    0.0112
   ⋮
```

```
A921 = U(:,1:921)*S(1:921,1:921)*V(:,1:921).'
```

```
A921 = 3072×4608
  189.0646  191.8936  188.8820  187.8085  190.9213  193.6283  196.9239  193.2751 ···
  188.8330  192.1524  189.7149  190.0349  191.8535  193.0206  196.4297  194.3539
  189.4446  192.6200  190.0603  190.5890  191.3810  192.0473  197.1031  196.4040
  191.2359  192.9759  190.2975  191.5527  190.4033  189.8999  196.5573  197.9993
  191.6491  193.2942  190.5409  193.6870  191.5410  189.6225  195.2290  197.1438
  190.4965  192.6234  190.1108  194.4575  193.5966  190.9228  195.0517  194.8664
  188.1900  191.7679  191.4803  193.0678  193.5434  192.0132  195.9747  195.3500
  188.3936  192.2560  192.5138  192.7716  192.9265  193.7176  196.6674  196.9272
  191.2526  192.7502  192.4498  192.8418  193.6039  194.3199  198.5062  199.0176
  192.6733  194.3953  194.4557  193.9848  194.1313  194.8077  197.6567  198.6137
     ⋮
```

Explain: k is found noting the equation $\mathrm{CR} = \frac{m\,n}{k(m+n+1)}$. This is approximately 921. Rank-921 can be found similarly to the previous problems where the product is found for the selected columns.

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
A921 = uint8(round(A921))
```

```
A921 = 3072×4608 uint8 matrix
   189   192   189   188   191   194   197   193   196   195   190   193   192 ···
   189   192   190   190   192   193   196   194   198   197   191   193   190
   189   193   190   191   191   192   197   196   198   198   192   192   188
   191   193   190   192   190   190   197   198   197   197   193   193   190
   192   193   191   194   192   190   195   197   195   196   193   196   193
   190   193   190   194   194   191   195   195   195   198   196   198   193
   188   192   191   193   194   192   196   195   197   198   196   199   194
   188   192   193   193   193   194   197   197   199   199   195   199   195
   191   193   192   193   194   194   199   199   199   199   197   200   193
   193   194   194   194   194   195   198   199   199   198   196   197   192
     ⋮
```

```
imshow(A921)
```

```
%Doesn't work for some reason
%RMSE_A921 = norm(A - A921, "fro") / 14155776
```

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

%CR 10
A184 = U(:,1:184)*S(1:184,1:184)*V(:,1:184).'
```

```
A184 = 3072×4608
  189.7432   191.5483   189.1171   188.5879   189.2042   190.4093   192.2473   192.2744 ...
  189.9357   191.6766   189.2152   189.0111   189.4455   190.7028   192.9499   193.0818
  190.3505   191.8218   189.2282   189.5003   189.7682   190.8722   193.6113   194.0608
  190.4580   191.6506   189.2845   190.0465   190.6301   191.5918   194.6700   195.6617
  189.7958   190.8026   188.7556   190.2446   190.8688   191.7150   195.1483   196.7306
  189.5912   190.5549   188.8867   190.6705   191.0324   191.8086   195.1119   196.9319
  189.8759   190.7995   189.5386   191.3636   191.7975   192.5087   195.5045   197.6150
  190.6872   191.5916   190.2235   192.3257   192.6240   193.2553   195.9026   198.1159
  192.4771   193.3784   192.0164   194.2350   194.6985   195.1316   197.0148   198.9931
  194.2754   195.2644   193.8108   195.8900   195.8527   196.3854   197.4867   199.1781
```

⋮

```
A184 = uint8(round(A184))
```

A184 = 3072×4608 uint8 matrix

190	192	189	189	189	190	192	192	195	195	193	195	192	···
190	192	189	189	189	191	193	193	195	195	193	196	193	
190	192	189	190	190	191	194	194	196	196	193	196	193	
190	192	189	190	191	192	195	196	197	197	195	197	194	
190	191	189	190	191	192	195	197	198	198	196	198	195	
190	191	189	191	191	192	195	197	198	198	196	198	194	
190	191	190	191	192	193	196	198	199	199	196	199	194	
191	192	190	192	193	193	196	198	199	199	197	199	194	
192	193	192	194	195	195	197	199	200	200	198	200	195	
194	195	194	196	196	196	197	199	200	200	198	200	194	

⋮

```
%Doesn't work for some reason
%RMSE_A184 = (norm((A - A184), "fro")) / 14155776

%CR 25
A73 = U(:,1:73)*S(1:73,1:73)*V(:,1:73).'
```

A73 = 3072×4608

196.1728	195.1275	195.4776	195.8066	195.9922	196.3256	197.8516	197.1960	···
195.8633	194.7987	195.0266	195.4386	195.6179	195.9075	197.4218	196.7157	
195.6152	194.5674	194.7155	195.1564	195.3749	195.5785	197.1343	196.3523	
195.2078	194.1990	194.3396	194.8276	195.0750	195.2628	196.8151	196.0302	
194.7469	193.7314	193.8364	194.3545	194.7001	194.8453	196.4123	195.6259	
193.9700	192.9447	193.0070	193.4848	193.9140	193.9856	195.5857	194.8052	
193.4523	192.4140	192.4656	192.8994	193.3315	193.4075	194.9764	194.2234	
192.5340	191.5733	191.6684	192.1111	192.6381	192.6999	194.2723	193.5650	
191.5945	190.6560	190.7508	191.1779	191.6857	191.7410	193.2559	192.6088	
190.3713	189.5224	189.6503	190.1076	190.6377	190.6754	192.2380	191.5890	

⋮

```
A73 = uint8(round(A73))
```

A73 = 3072×4608 uint8 matrix

196	195	195	196	196	196	198	197	198	198	198	199	197	···
196	195	195	195	196	196	197	197	197	198	197	199	196	
196	195	195	195	195	196	197	196	197	198	197	198	196	
195	194	194	195	195	195	197	196	196	197	197	198	195	
195	194	194	194	195	195	196	196	196	197	196	197	195	
194	193	193	193	194	194	196	195	195	196	196	197	194	
193	192	192	193	193	193	195	194	195	196	195	196	193	
193	192	192	192	193	193	194	194	194	195	194	195	192	
192	191	191	191	192	192	193	193	193	194	194	194	191	
190	190	190	190	191	191	192	192	192	193	193	193	190	

⋮

```
%Doesn't work for some reason
%RMSE_A73 = (norm(A - A73, "fro")) / 14155776

%CR 75
A24 = U(:,1:24)*S(1:24,1:24)*V(:,1:24).'
```

A24 = 3072×4608

```
185.9355  185.5716  186.2206  186.2008  186.0238  185.9327  186.6144  186.6621 ···
184.7410  184.4320  185.1037  185.0923  185.0017  184.9047  185.6141  185.6632
183.3023  183.0307  183.7098  183.6913  183.6872  183.5824  184.3103  184.3629
182.4018  182.1714  182.8472  182.8201  182.8667  182.7687  183.5063  183.5684
181.4682  181.3176  182.0076  181.9737  182.1121  182.0337  182.8004  182.8873
180.1499  180.0905  180.7854  180.7384  181.0017  180.9448  181.7475  181.8645
178.8442  178.8757  179.5856  179.5313  179.9333  179.9032  180.7493  180.8969
177.4876  177.6253  178.3584  178.2944  178.8301  178.8304  179.7226  179.9028
176.8493  177.0963  177.8345  177.7672  178.4268  178.4667  179.4033  179.6257
176.1949  176.5125  177.2487  177.1891  177.9467  178.0068  178.9639  179.2086
    ⋮
```

```
A24 = uint8(round(A24))
```

A24 = 3072×4608 uint8 matrix

```
 186   186   186   186   186   186   187   187   187   187   187   187   186 ···
 185   184   185   185   185   185   186   186   186   186   186   186   185
 183   183   184   184   184   184   184   184   185   185   185   185   184
 182   182   183   183   183   183   184   184   184   184   184   184   183
 181   181   182   182   182   182   183   183   183   183   183   183   183
 180   180   181   181   181   181   182   182   182   182   182   183   182
 179   179   180   180   180   180   181   181   181   181   181   182   181
 177   178   178   178   179   179   180   180   180   180   181   181   180
 177   177   178   178   178   178   179   180   180   180   180   181   180
 176   177   177   177   178   178   179   179   180   180   180   180   179
  ⋮
```

```
%Doesn't work for some reason
%RMSE_A24 = (norm(A - A24, "fro")) / 14155776

imshow(A184)
```

```
imshow(A73)
```

```
imshow(A24)
```

Explain: Despite my RMSE functions not working properly, it's easy to see that the higher the CR is, the higher amount of loss we have in each photo. Had my functions worked properly, we'd see a higher number for the RMSE, meaning a higher margin of error.