COMP4107 - Assignment 1

Student Name: Yunkai Wang
Student Number: 100968473

Student Name: Jules Kuehn
Student Number: 100661464

Fall 2018

1. Question 1
   Implementation for question 1 can be found in q1.py. The result is

```
A =
 [[3 1 2 3]
 [4 3 4 3]
 [3 2 1 5]
 [1 6 5 2]]

U =
 [[-0.3593326  -0.36767659  0.29605046 -0.80501437]
 [-0.56750746 -0.08799758  0.62845599  0.52462823]
 [-0.4428526  -0.56862492 -0.65902357  0.21502376]
 [-0.59388293  0.73057242 -0.28824492 -0.17459057]]

S =
 [12.22151125  4.92815942  2.06380839  0.29766152]

V =
 [[-0.43124523 -0.53273754 -0.52374556 -0.50587435]
 [-0.49315012  0.53052572  0.40520071 -0.5578152 ]
 [ 0.55075835 -0.41966021  0.48729169 -0.53206894]
 [ 0.51719991  0.50854546 -0.5692537  -0.38708653]]

Alice2D =
 [-0.62562864 -0.295158  ]

Alice4D =
 [-0.62562864 -0.295158   -0.20518073 -7.69146926]

PredictAliceEPL =
 5.354514697523497

Closest to Alice in 2D and 4D =
 0 2
```

   Notice that the matrices U, V are different with the matrices in the slide, however, in order to generate the same result, we have to take the transpose of the original matrix, which is not something that we decided to go with since I don't think we are allowed to do that. The closest user in 2D is Alicia, and the closest user in 4D is Mary.

2. Question 2
   Implementation for question 2 can be found in q2.py. The result is

```
A =
 [[1 2 3]
 [2 3 4]
 [4 5 6]
 [1 1 1]]

U =
 [[-0.33306893 -0.73220483  0.57543613 -0.1476971 ]
 [-0.48640367 -0.34110504 -0.56984703  0.56774394]
 [-0.79307315  0.44109455 -0.0055891  -0.42004684]
 [-0.15333474  0.39109979  0.58661434  0.69239659]]

S =
 [[1.10528306e+01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 9.13748280e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.00674393e-16]]

V =
 [[-0.41903326 -0.56492763 -0.71082199]
 [ 0.81101447  0.11912225 -0.57276996]
 [ 0.40824829 -0.81649658  0.40824829]]
```

3. Question 3
   Implementation for question 3 can be found in q3.py. The best rank(2) matrix takes the first 2 values since s is sorted descending. The result is very good.

```
||A - A2|| = 1.331189632858722

||A - A2|| / ||A|| = 0.116%
```

4. Question 4
   Implementation for question 4 can be found in q4.py. The only learning rate that will work is when $\varepsilon = 0.01$, which will lead to the correct result with ≈ 420 iterations. The other ones won't work as we are descending too quickly, and therefore we will miss the correct answer and failed to come back. It "bounces around" the correct result, while the norm continues to increase. We set the program to stop once the norm of the vector is greater than 100000, which is obviously too big and therefore it's impossible to get the result.

```
 P-inv sol'n:    -0.147059     0.058824     0.264706
|    Step     | RESULT |                 x                 |  Iter  |
============================================================
 step: 0.010:  success    -0.156771     0.057397     0.271565    420
 step: 0.050:  failed  -5133.149689 -6919.777001 -8706.404313    5
```

```
step: 0.100:  failed  23356.638100 31489.229900 39621.821700    4
step: 0.150:  failed  -7673.780375 -10345.031125 -13016.281875    3
step: 0.200:  failed  -18987.992000 -25598.536000 -32209.080000    3
step: 0.250:  failed  -38043.265625 -51288.296875 -64533.328125    3
step: 0.500:  failed  -320060.375000 -431495.125000 -542929.875000    3
```

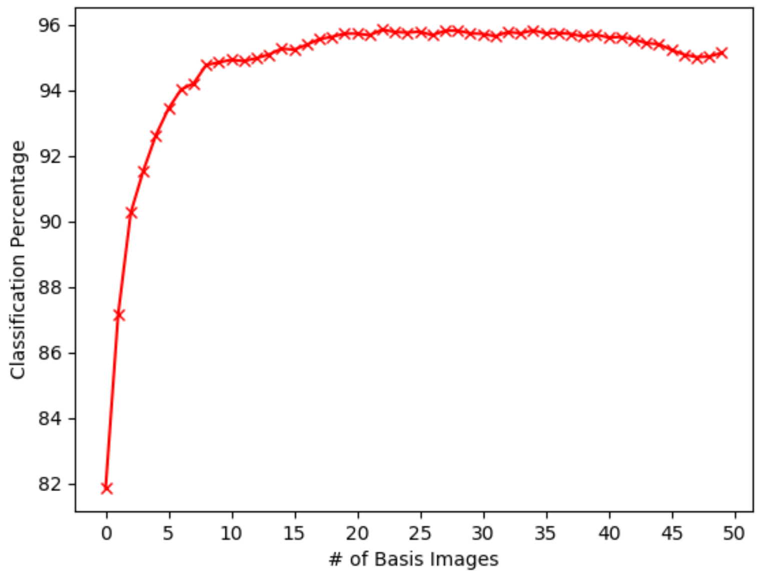
5. Question 5

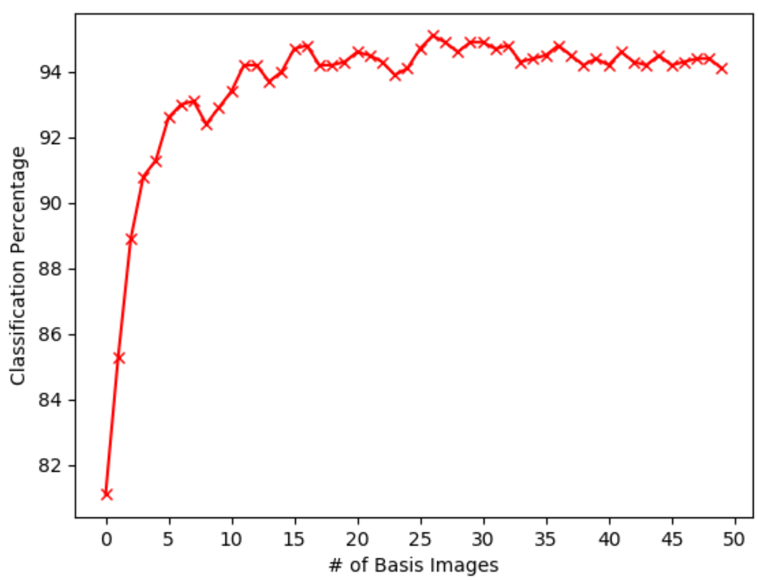
   The implementation for q5 can be found in q5.py. $A = \begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 2 & 3 \\ -2 & -2 & 3 & 3 \end{bmatrix}$. It's clear that the rows of A are not linearly independent. To see this, we reduce A to its RREF form and we will get $A = \begin{bmatrix} 1 & 0 & 2 & 3 \\ 0 & 1 & -7/2 & -5/2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, rank of rows of A equals to 2. The columns of A are not linearly independent either. Since as we reduce A^T to RREF form, we will get $A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, clearly that rank of the columns of A equal to 2, so the columns are not linearly independent. The inverse of A cannot be calculated as A is not a square matrix, and the rows and columns of A are not linearly independent.

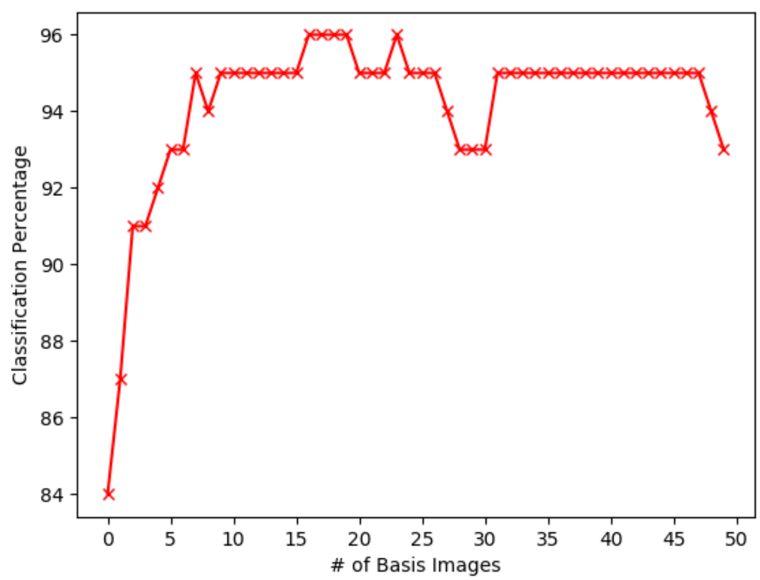
```
A in rref:
 Matrix([[1, 0, 2, 3], [0, 1, -7/2, -5/2], [0, 0, 0, 0]])
Independent rows: 2

A^T in rref:
 Matrix([[1, 0, -1], [0, 1, 1], [0, 0, 0], [0, 0, 0]])
Independent cols: 2
```

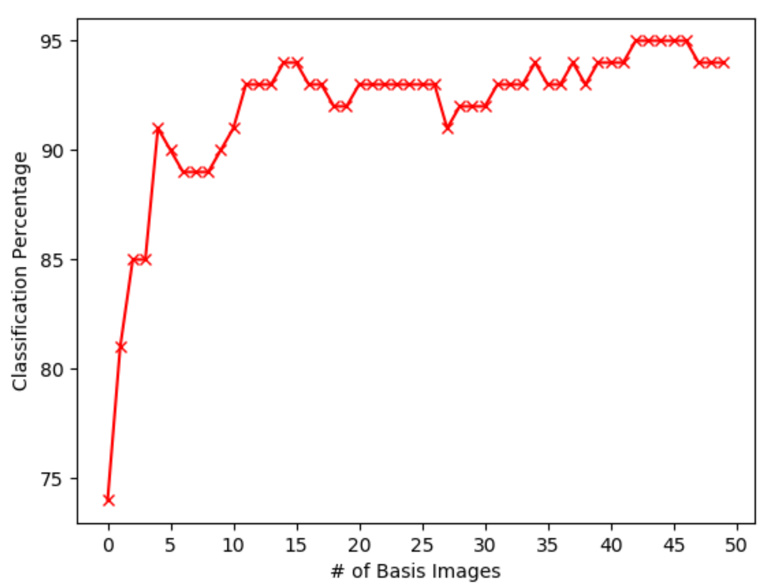
6. Question 6

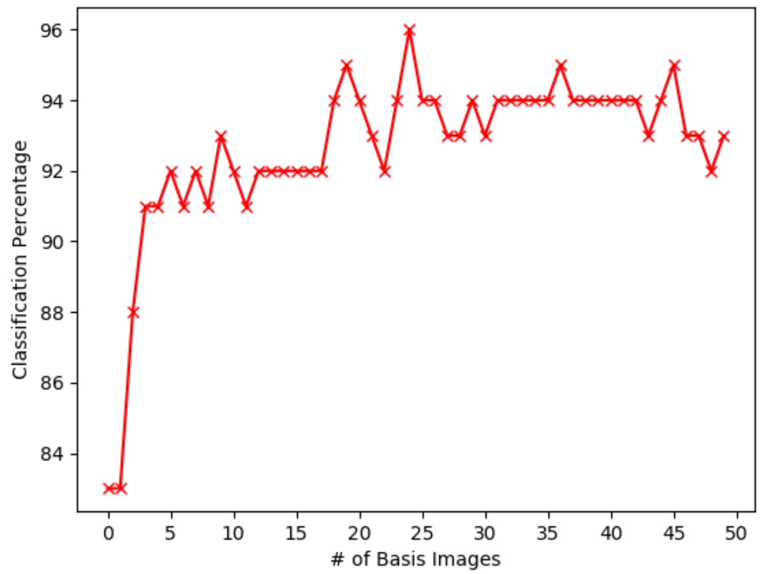
   Implementation for question 6 can be found in q6.py. The implementation correctly replicated the results of the paper, for the entire set of 10000.
   Also included are plots from a random sample of 1000, followed by 3 random samples of 100.


96
94
92
90
88
86
84
82
Classification Percentage
0 5 10 15 20 25 30 35 40 45 50
# of Basis Images


94
92
90
88
86
84
82
Classification Percentage
0 5 10 15 20 25 30 35 40 45 50
# of Basis Images


96
94
92
90
88
86
84
Classification Percentage
0
5
10
15
20
25
30
35
40
45
50
# of Basis Images

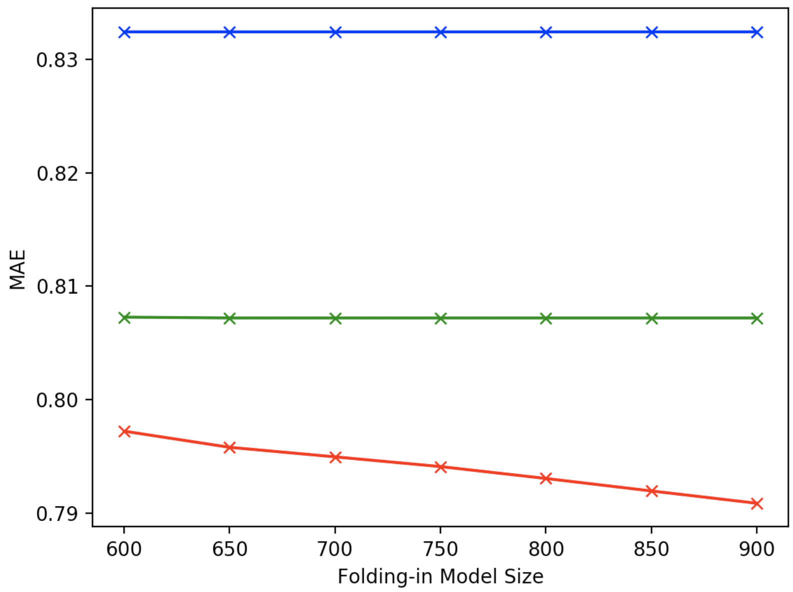

95
90
85
80
75
Classification Percentage
0
5
10
15
20
25
30
35
40
45
50
# of Basis Images

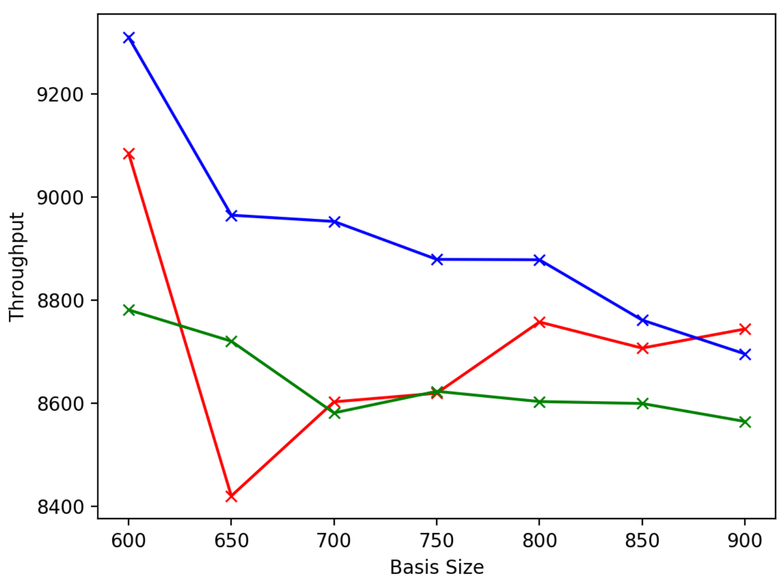


7. Question 7

   Implementation for question 7 can be found in q7.py. To accomplish question 7, we read another paper online which can be found at here. We used the filling-in of missing data, and normalization method from this paper. We used the folding in method mentioned in the paper referenced by the assignment, producing the following results.

```
Using  0.8  as training/test rate
Basis size: 600 , Error: 0.7972111972363619
Basis size: 650 , Error: 0.7957876675655522
Basis size: 700 , Error: 0.7949468783224024
Basis size: 750 , Error: 0.7940886277044001
Basis size: 800 , Error: 0.7930407111576059
Basis size: 850 , Error: 0.791941014251666
Basis size: 900 , Error: 0.7908558142151606
---------------------------------
Using  0.5  as training/test rate
Basis size: 600 , Error: 0.8072631069473807
Basis size: 650 , Error: 0.8071935186411143
Basis size: 700 , Error: 0.8071934221331365
Basis size: 750 , Error: 0.8071933384906437
Basis size: 800 , Error: 0.8071932653017063
Basis size: 850 , Error: 0.8071932007219131
Basis size: 900 , Error: 0.8071931433165886
---------------------------------
Using  0.2  as training/test rate
Basis size: 600 , Error: 0.8324159212635108
Basis size: 650 , Error: 0.8324158286720662
Basis size: 700 , Error: 0.8324157493082648
Basis size: 750 , Error: 0.8324156805264993
Basis size: 800 , Error: 0.8324156203426499
Basis size: 850 , Error: 0.8324155672393692
Basis size: 900 , Error: 0.8324155200365596
```


MAE
Folding-in Model Size
0.83
0.82
0.81
0.80
0.79
600
650
700
750
800
850
900


Throughput
Basis Size
9200
9000
8800
8600
8400
600
650
700
750
800
850
900