Connor Raymond Stewart
ID: 101041125

COMP 4107 Assignment One

Question One

As can be seen from q1.py, we get the following output:

[Note for SVD: $A=U\sum V^T$]

$$A = \begin{bmatrix}1 & 2 & 3\\ 2 & 3 & 4\\ 4 & 5 & 6\\ 1 & 1 & 1\end{bmatrix} =$$

$$\begin{bmatrix}-0.33306893 & -0.73220483 & 0.57543613 & -0.1476971\\ -0.48640367 & -0.34110504 & -0.56984703 & 0.56774394\\ -0.79307315 & 0.44109455 & -0.0055891 & -0.42004684\\ -0.15333474 & 0.39109979 & 0.58661434 & 0.69239659\end{bmatrix}\begin{bmatrix}11.05283059 & 0 & 0\\ 0 & 0.91374828 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}-0.41903326 & -0.56492763 & -0.71082199\\ 0.81101447 & 0.11912225 & -0.57276996\\ 0.40824829 & -0.81649658 & 0.40824829\end{bmatrix}$$

Since, $U = \begin{bmatrix}-0.33306893 & -0.73220483 & 0.57543613 & -0.1476971\\ -0.48640367 & -0.34110504 & -0.56984703 & 0.56774394\\ -0.79307315 & 0.44109455 & -0.0055891 & -0.42004684\\ -0.15333474 & 0.39109979 & 0.58661434 & 0.69239659\end{bmatrix}$, $S=\begin{bmatrix}11.05283059 & 0 & 0\\ 0 & 0.91374828 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$,

And $V = \begin{bmatrix}-0.41903326 & -0.56492763 & -0.71082199\\ 0.81101447 & 0.11912225 & -0.57276996\\ 0.40824829 & -0.81649658 & 0.40824829\end{bmatrix}$

This is from the following python output:

```
[1] [[-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]]
[2] [11.05283059  0.91374828  0.        ]
[3] [[-0.41903326 -0.56492763 -0.71082199]
     [ 0.81101447  0.11912225 -0.57276996]
     [ 0.40824829 -0.81649658  0.40824829]]
```

Where line [1] represents matrix U, line [2] represents the singular values, and line [3] represents matrix V. Also note that in the matrix above, the singular values are converted to a diagonal matrix using the singular values array-output.

Question Two

a) Compute the best rank(2) matrix, A_2, approximation to the matrix A. What is $\| A - A2 \|_2$?

We know that the best rank(2) matrix is given from the two initial values as the diagonal matrix (named S, D, or sometimes Σ) is sorted in a descending order.

We get the following python output: (using Euclidian Norms for the norm type 2's)

The Rank(2) Matrix is:

```
[[0.17447807 0.17754332 0.18056607 ... 0.18056607 0.17754332 0.17447807]
 [0.17754332 0.18059153 0.18359756 ... 0.18359756 0.18059153 0.17754332]
 [0.18056607 0.18359756 0.18658718 ... 0.18658718 0.18359756 0.18056607]
 ...
```

[0.18056607 0.18359756 0.18658718 ... 0.18658718 0.18359756 0.18056607]

[0.17754332 0.18059153 0.18359756 ... 0.18359756 0.18059153 0.17754332]

[0.17447807 0.17754332 0.18056607 ... 0.18056607 0.17754332 0.17447807]]

The norm ||A-A_2|| = 1.3311896328587232

Since ||A|| = 1149.2178184026561, we know that ||A-A_2|| / ||A|| = 0.0011583440593611722

Thus, we get a norm of $||A - A_2||_2 = 1.3311896328587232$, and the above truncated A_2 matrix.

b) Is this a good low-rank approximation to use? Why or why not?

In the above part a, we note that:

Since ||A|| = 1149.2178184026561, we know that ||A-A_2|| / ||A|| = 0.0011583440593611722

This shows us that $||A-A_2|| / ||A|| = 0.0011583440593611722$, which means $||A-A_2|| / ||A|| = 0.11583440593\%$ which is approximately 0.1158% when rounded. This is a very good rank(2) matrix since the percentage difference in magnitude between the matrix A and A_2 is less then a half percentage. Since Euclidian norms measures the distance from the vector coordinate to the origin of the vector space. Thus, since the Euclidian norm of A minus A_2 is small (approximately 1.33) this value is quite tiny. We can see that the Euclidian norm of A minus A_2 over the Euclidian norm of A is a very small percentage (approximately 0.1158%) showing that the overall distance to the origin is a very tiny percentage of A's overall distance to the origin. This indicates that since the Euclidian norm of A minus A_2 ($||A-A_2||$) is very small, A and A_2 are very close together in terms of their coordinates; and since the Euclidian norm of A minus A2 over the Euclidian norm of A is a very small percentage ($||A-A_2|| / ||A||$), we can easily see that there is a very small overall difference in distance between A_2 and A. By extension, its clear that the closer ($||A-A_2|| / ||A||$) is to 0%, the better the fit is between A and A_2, and the further it is from 0%, the worse the fit. Finally, this lets us conclude that in general A_2 is a very-good fit since ($||A-A_2|| / ||A||$) is close to zero and $||A-A_2||$ has a small magnitude in general.

Therefore, we can conclude that *the low-rank approximation is a very-good approximation to use*, as explained by the reasoning above.

Question Three

We get the following results from this calculation (full results in appendix):

X-Values	Status	Epsilon	Number of Iterations
[[-0.08244796] [-0.04971168] [0.31287608]]	Success	0.01	310
[[148694.87649648] [200465.77802089] [252237.00939677]]	Divergence	0.05	8
[[638649.11998771] [861006.89507269] [1083365.00000914]]	Divergence	0.1	6
[[8672014.15134636] [11691340.22733588] [14710666.63317687]]	Divergence	0.15	6

[[96700.57637215] [130368.71267648] [164037.17883229]]	Divergence	0.2	4
[[244249.12483453] [329289.09846073] [414329.4019384]]	Divergence	0.25	4
[[4179358.00540714] [5634480.75434705] [7089603.83313843]]	Divergence	0.5	4

As can be seen above, divergence occurs when the epsilon is greater then 0.01 in every case. Divergence was tested by checking if any of the x-values in the x-vector became greater then 120 000. Since 120 000 is already way too big for matrix A, this indicates that the model bounced up the parabolic-estimation curve and is now far out of bounds.

Question 4:

(4) Find two linearly independent vectors belonging to the null space of the matrix:

$$A=\begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 2 & 3 \\ -2 & -2 & 3 & -1 \end{bmatrix}$$

We can find the null-space by using elimination:

$$\begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 2 & 3 \\ -2 & -2 & 3 & -1 \end{bmatrix} \xrightarrow{R_1'=R_3} \begin{bmatrix} 1 & 0 & 2 & 3 \\ 3 & 2 & -1 & 4 \\ -2 & -2 & 3 & -1 \end{bmatrix} \xrightarrow[R_3'=R_3+2R_1]{R_2'=R_2-3R_1} \begin{bmatrix} 1 & 0 & 2 & 3 \\ 0 & 2 & -7 & -5 \\ 0 & -2 & 7 & 5 \end{bmatrix}$$

$$\xrightarrow{R_3'=R_3+R_2} \begin{bmatrix} 1 & 0 & 2 & 3 \\ 0 & 2 & -7 & -5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{R_2'=R_2/2} \begin{bmatrix} \textcircled{1} & 0 & 2 & 3 \\ 0 & \textcircled{1} & -\frac{7}{2} & -\frac{5}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

this corresponds to the following system:

$$\left.\begin{aligned} x_1+2x_3+3x_4=0 \\ x_2-\tfrac{7}{2}x_3-\tfrac{5}{2}x_4=0 \end{aligned}\right\} \begin{aligned} x_1&=-2x_3-3x_4 \\ x_2&=\tfrac{7}{2}x_3+\tfrac{5}{2}x_4 \end{aligned}$$ So x_3 & x_4 are free

thus, we see:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 7/2 \\ 1 \\ 0 \end{bmatrix} x_3 + \begin{bmatrix} -3 \\ 5/2 \\ 0 \\ 1 \end{bmatrix} x_4$$

therefore, the null space has the basis formed by the following set:

$$\left\{ \begin{bmatrix} -2 \\ 7/2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -3 \\ 5/2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

thus, two linearly independent vectors belonging to the null space are:

$$\begin{bmatrix} -2 \\ 7/2 \\ 1 \\ 0 \end{bmatrix} \& \begin{bmatrix} -3 \\ 5/2 \\ 0 \\ 1 \end{bmatrix}$$ Since the null space is a linearly independent set

a) Are the columns of A linearly independent in $\mathbb{R}^3$? Why or why not?
As seen in the above calculations:
the null space is NOT the zero vector, meaning the columns must be linearly dependent.
This is also evidenced by the fact that matrix A's RES-form is:

$$\begin{bmatrix} \textcircled{1} & 0 & 2 & 3 \\ 0 & \textcircled{1} & -\frac{7}{2} & -\frac{5}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ which shows that x_3 & x_4 are free

Thus, the lack of a trivial nullspace in A means A's columns are NOT linearly independent in $\mathbb{R}^3$ (they are linearly dependent)

b) For example:

$$\begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} = 2\begin{bmatrix} 3 \\ 1 \\ -2 \end{bmatrix} - \frac{7}{2}\begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix}$$ Showing this linear dependence
(A linear combination of vectors exists)

b) Are the rows of A linearly independent? why or why not?
As was seen in the RES:

$$\begin{bmatrix} \textcircled{1} & 0 & 2 & 3 \\ 0 & \textcircled{1} & -\frac{7}{2} & -\frac{5}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ the last row is empty & the rank is 2.
↳ this rowspace is NOT linearly independent

For example:
$$[-2 \ -2 \ 3 \ -1] = -[3 \ 2 \ -1 \ 4] + [1 \ 0 \ 2 \ 3]$$ Proving this is the case.
(A linear combination of Rows exists)

c) See Page Below:

C) Calculate the inverse of matrix A.

We cannot calculate the standard inverse of a non-square matrix; however, we still can calculate the pseudo-inverse via the Moore-Penrose method in NumPy. The solution is:

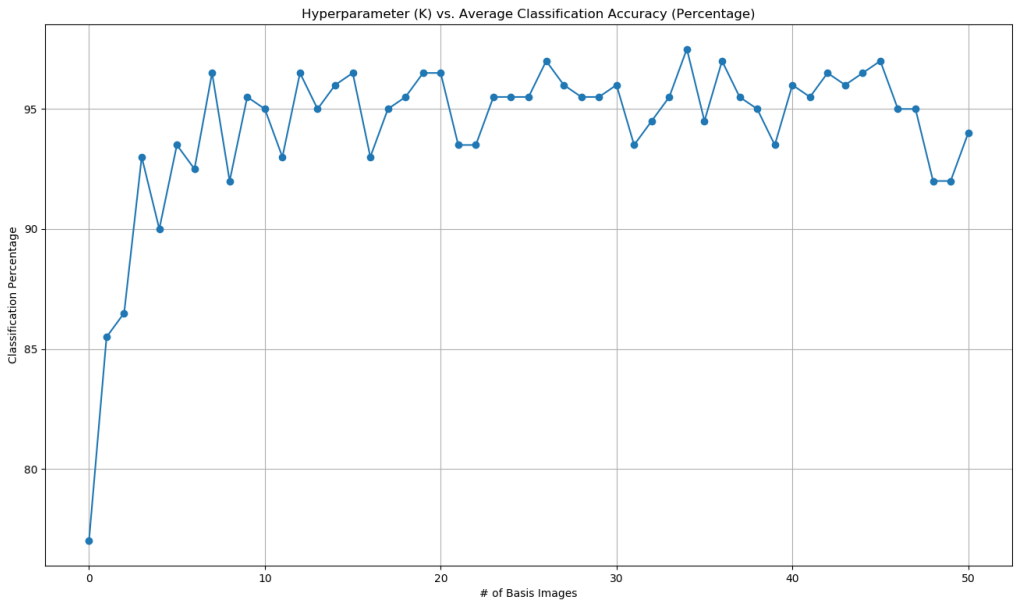
$$A^{-1} = \text{inverse of } \begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 2 & 3 \\ -2 & -2 & 3 & -1 \end{bmatrix} = \begin{bmatrix} 0.06507304 & 0.01460823 & -0.05046481 \\ 0.03984064 & -0.03187251 & -0.07171315 \\ -0.00929615 & 0.14077025 & 0.1500664 \\ 0.09561753 & 0.12350598 & 0.02788845 \end{bmatrix}$$

See the attached python code (q4.py) for more details on this.

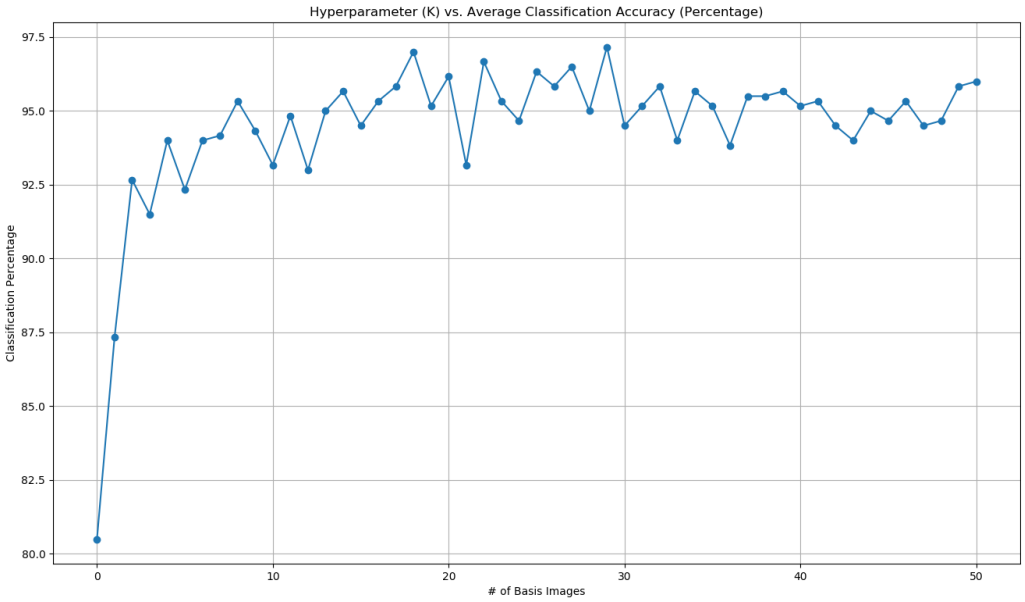
Question Five

The implementation for this problem can be found under q5.py. As can be seen for a set of 1000 random samples from each digit (0 to 9), we see the following graph as a result:

Without smoothing: (200 initial images)



With smoothing: (600 initial images)



We can see a pattern which follows that seen from the original paper by Lassiter (2013). The initial value for classification percentage starts off low and increases exponentially with an increase in the number basis images during the first few steps. After this, the data starts to flatten out over the remaining iterations. The graph appears to have some fluctuation in it, but this is in part caused by the smoothing of the graph being different from the papers. The paper by Lassiter (2013) has 26 data points in its graph whereas the one plotted from matplotlib has all 50. When smoothing these points out in another run of the program (by using a larger batch of testing images), we see the results from the second image above. The images above could be further smoothed out by adding a filter to average the points from the figure to match the number of data points from the paper. Further, using more initial images or a larger subset of the MNIST dataset could also help. Finally, despite the ‘jumpiness’ of the graph above, it is clear that (when smoothed out to be flat like a curve) it matches the papers general trend.

Appendix – Raw Output from q3.py code used for table

Using an epsilon of 0.01 we see that we get the following least-squares vector:

---Success---(results below)

[[-0.08244796]

[-0.04971168]

[0.31287608]]

After 310 iterations in the loop

Using an epsilon of 0.05 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[148694.87649648]

[200465.77802089]

[252237.00939677]]

After 8 iterations in the loop

Using an epsilon of 0.1 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[638649.11998771]

[861006.89507269]

[1083365.00000914]]

After 6 iterations in the loop

Using an epsilon of 0.15 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[8672014.15134636]

[11691340.22733588]

[14710666.63317687]]

After 6 iterations in the loop

Using an epsilon of 0.2 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[96700.57637215]

[130368.71267648]

[164037.17883229]]

After 4 iterations in the loop

Using an epsilon of 0.25 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[244249.12483453]

[329289.09846073]

[414329.4019384]]

After 4 iterations in the loop

Using an epsilon of 0.5 we see that we get the following least-squares vector:

---Divergence---(last estimate below)

[[4179358.00540714]

[5634480.75434705]

[7089603.83313843]]

After 4 iterations in the loop