main

July 21, 2024

1 Question 8

Part A: Define Matrix A and SVD

[ ]:
```
import numpy as np

A = np.array([
    [1, 2, 3, 4],
    [4, 3, 2, 1],
    [1, 1, 1, 1],
    [0, 1, 2, 3],
    [1, 3, 2, 4]
])

U, S, Vt = np.linalg.svd(A)
print(f"U: {U}")
print(f"S: {S}")
print(f"Vt: {Vt}")

Sigma = np.diag(S)
print(f"Sigma: {Sigma}")

Sigma_full = np.zeros((A.shape[0], A.shape[1]))
Sigma_full[:Sigma.shape[0], :Sigma.shape[1]] = Sigma

print(f"Sigma_full: {Sigma_full}")
```

```
U: [[-5.57226401e-01  2.85816990e-01 -4.55858954e-01  5.61572987e-01
  -2.90922293e-01]
 [-4.59476939e-01 -8.42323299e-01 -1.12719724e-01 -2.02015835e-01
  -1.60798847e-01]
 [-2.03340668e-01 -1.11301262e-01 -1.13715736e-01  2.46490352e-01
   9.34117680e-01]
 [-3.53885733e-01  3.97118252e-01 -3.42143218e-01 -7.63588822e-01
   1.30123446e-01]
 [-5.58387984e-01  1.96746617e-01  8.05911675e-01 -8.32667268e-17
  -2.08166817e-17]]
S: [9.58659829e+00 3.86203400e+00 1.08711850e+00 5.85762774e-16]
```

```
Vt: [[-0.32929958 -0.49290414 -0.48176851 -0.64537307]
 [-0.77628287 -0.27945872  0.06453409  0.56135824]
 [-0.19734915  0.65494147 -0.71675271  0.13553791]
 [ 0.5        -0.5        -0.5         0.5       ]]
Sigma:  [[9.58659829e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 3.86203400e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.08711850e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 5.85762774e-16]]
Sigma_full: [[9.58659829e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 3.86203400e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.08711850e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 5.85762774e-16]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```

Part B: Rank of A

```
[ ]: rank = np.linalg.matrix_rank(A)
     print(f"rank of A: {rank}")
```

```
rank of A: 3
```

Part C: Find Sigma_dagger and U^T

```
[ ]: Sigma_pseudo_inv = np.linalg.pinv(Sigma_full)
     U_transpose = U.T
     print(f"Sigma_dagger: {Sigma_pseudo_inv}")
     print(f"U Transpose:: {U_transpose}")
```

```
Sigma_dagger: [[0.10431229 0.         0.         0.         0.        ]
 [0.         0.25893092 0.         0.         0.        ]
 [0.         0.         0.91986293 0.         0.        ]
 [0.         0.         0.         0.         0.        ]]
U Transpose:: [[-5.57226401e-01 -4.59476939e-01 -2.03340668e-01 -3.53885733e-01
  -5.58387984e-01]
 [ 2.85816990e-01 -8.42323299e-01 -1.11301262e-01  3.97118252e-01
   1.96746617e-01]
 [-4.55858954e-01 -1.12719724e-01 -1.13715736e-01 -3.42143218e-01
   8.05911675e-01]
 [ 5.61572987e-01 -2.02015835e-01  2.46490352e-01 -7.63588822e-01
  -8.32667268e-17]
 [-2.90922293e-01 -1.60798847e-01  9.34117680e-01  1.30123446e-01
  -2.08166817e-17]]
```

Part D: Psuedoinverse of A

```
[ ]: A_pseudo_inv = Vt.T @ Sigma_pseudo_inv @ U_transpose
     print(f"A_pseudo_inv: {A_pseudo_inv}")
```

```
A_pseudo_inv: [[ 0.04444444  0.20555556  0.05       -0.00555556 -0.16666667]
 [-0.26666667  0.01666667 -0.05       -0.21666667  0.5       ]
```

```
 [ 0.33333333  0.08333333  0.08333333  0.25        -0.5        ]
 [ 0.02222222 -0.10555556 -0.01666667  0.03888889  0.16666667]]
```

2 Part e: Verify moore-penrose conditions

[ ]:
```python
condition_1 = np.allclose(A @ A_pseudo_inv @ A, A)
condition_2 = np.allclose(A_pseudo_inv @ A @ A_pseudo_inv, A_pseudo_inv)
condition_3 = np.allclose((A @ A_pseudo_inv).T, A @ A_pseudo_inv)
condition_4 = np.allclose((A_pseudo_inv @ A).T, A_pseudo_inv @ A)
print(f"condition_1 (AA†A = A): {condition_1}")
print(f"condition_2 (A†AA† = A†): {condition_2}")
print(f"condition_3 ((AA†)T = AA†): {condition_3}")
print(f"condition_4  ((A†A)T = A†A): {condition_4}")
```

```
condition_1 (AA†A = A): True
condition_2 (A†AA† = A†): True
condition_3 ((AA†)T = AA†): True
condition_4  ((A†A)T = A†A): True
```