

HW1-Q1.py

```

1 # Peyton Miller, Homework 1
2
3 #Question 1:
4
5 #Asked ChatGPT the following:
6 # "Develop an algorithm in python without numpy that numerically determines the SVD
  of a matrix, A.
7 # The algorithm should compute the SVD by
8 # (i) transforming A into a bidiagonal matrix B using Householder reflections,
9 # (ii) determine the QR-factorization of B using Givens rotations to find the SVD of
  B,
10 # (iii) determine the SVD of A using the SVD of B. Provide specific comments on each
   step above.
11 # You do not need to provide comments on the details of how the Givens rotations or
   Householder reflections are applied.
12 # Apply your algorithm to the matrix below, and provide each matrix component of the
   decomposition  $A = U\Sigma V^T$ .
13 # A = [ 1 4 9
14 #       3 2 3
15 #       6 4 5 ]"

```

```

16
17 import math
18 import copy

```

```

19
20
21 def matmul(A, B):
22     m, n, p = len(A), len(B), len(B[0])
23     C = [[0.0]*p for _ in range(m)]
24     for i in range(m):
25         for j in range(p):
26             for k in range(n):
27                 C[i][j] += A[i][k] * B[k][j]
28     return C

```

setting up
matrix $A \in \mathbb{R}^{m \times n}$
and $B \in \mathbb{R}^{n \times n}$
with transpose,
identity, and norm
properties

```

29
30 def transpose(A):
31     return [list(row) for row in zip(*A)]
32
33 def identity(n):
34     return [[1.0 if i == j else 0.0 for j in range(n)] for i in range(n)]
35
36 def norm(v):
37     return math.sqrt(sum(x*x for x in v))

```

```

38
39 def householder_vector(x):
40     v = x[:]
41     alpha = norm(v)
42     if alpha == 0:
43         return v
44     if v[0] >= 0:

```

Householder reflections (step 1)
for $A = U, B, V^T$ U, V are
orthog.
 $H_u(x) = \tilde{x} - 2\tilde{u}\tilde{u}^T$, $\tilde{u} \in \mathbb{R}^n$
bidiagonal

```

45     alpha = -alpha
46     v[0] -= alpha
47     beta = norm(v)
48     return [vi / beta for vi in v]
49
50 def apply_householder_left(A, v, k):
51     for i in range(k, len(A)):
52         s = sum(v[j-k] * A[j][i] for j in range(k, len(A)))
53         for j in range(k, len(A)):
54             A[j][i] -= 2 * v[j-k] * s
55
56 def apply_householder_right(A, v, k):
57     for i in range(len(A)):
58         s = sum(v[j-k] * A[i][j] for j in range(k, len(A[0])))
59         for j in range(k, len(A[0])):
60             A[i][j] -= 2 * v[j-k] * s
61
62

```

applying
to the left
and to the right
unit $A = UBV^T$

```

63 def bidiagonalize(A):
64     A = copy.deepcopy(A)
65     m, n = len(A), len(A[0])
66     U = identity(m)
67     V = identity(n)
68
69     for k in range(min(m, n)):
70
71         x = [A[i][k] for i in range(k, m)]
72         v = householder_vector(x)
73         apply_householder_left(A, v, k)
74         apply_householder_left(U, v, k)
75
76         if k < n - 1:
77
78             x = [A[k][j] for j in range(k+1, n)]
79             v = householder_vector(x)
80             apply_householder_right(A, v, k+1)
81             apply_householder_right(V, v, k+1)
82
83     return U, A, V

```

step 2: computing $B = UBV^T$ using QR

app'ly left + householder

app'ly to right + householder

$A = U \Sigma V^T$

GIVEN'S ROTATION from QR factorization,

```

86 def givens(a, b):
87     r = math.hypot(a, b)
88     return a/r, b/r

```

B becomes
 $B = U_2 \Sigma V_2^T$
↑ bidiagonal
STEP 3:

```

91 def svd_bidiagonal(B, iterations=50):
92     n = len(B)
93     U = identity(n)
94     V = identity(n)

```

determine SVD of A
using B.

```

95 B = copy.deepcopy(B)
96
97 for _ in range(iterations):
98
99     for i in range(n-1):
100         c, s = givens(B[i][i], B[i+1][i])
101         for j in range(n):
102             t1 = c*B[i][j] + s*B[i+1][j]
103             t2 = -s*B[i][j] + c*B[i+1][j]
104             B[i][j], B[i+1][j] = t1, t2
105
106             t1 = c*U[i][j] + s*U[i+1][j]
107             t2 = -s*U[i][j] + c*U[i+1][j]
108             U[i][j], U[i+1][j] = t1, t2
109
110     for i in range(n-1):
111         c, s = givens(B[i][i], B[i][i+1])
112         for j in range(n):
113             t1 = c*B[j][i] + s*B[j][i+1]
114             t2 = -s*B[j][i] + c*B[j][i+1]
115             B[j][i], B[j][i+1] = t1, t2
116
117             t1 = c*V[j][i] + s*V[j][i+1]
118             t2 = -s*V[j][i] + c*V[j][i+1]
119             V[j][i], V[j][i+1] = t1, t2
120
121 Sigma = [[0.0]*n for _ in range(n)]
122 for i in range(n):
123     Sigma[i][i] = abs(B[i][i])
124
125 return U, Sigma, V

```

 U, U_2
 $V_1^T V_2^T$
 Σ

for lines 95-125:

$$A = U_1 B V_1^T \quad B = U_2 \Sigma V_2^T$$

sub B

$$A = \underbrace{(U_1, U_2)}_{99-108} \underbrace{\Sigma}_{121-123} \underbrace{(V_1, V_2)^T}_{110-119}$$

Applied to A

$$A = \begin{bmatrix} 1 & 4 & 9 \\ 3 & 2 & 3 \\ 6 & 4 & 5 \end{bmatrix} \Rightarrow V = \begin{bmatrix} -0.3917 & -0.8293 & 0.3975 \\ -0.3471 & 0.5306 & 0.7731 \\ -0.8519 & 0.1793 & -0.9924 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 13.7486 & 0 & 0 \\ 0 & 5.6949 & 0 \\ 0 & 0 & 0.4583 \end{bmatrix} U = \begin{bmatrix} -0.5654 & -0.26293 & 0.5335 \\ -0.3397 & 0.7756 & 0.5324 \\ -0.7516 & 0.0509 & -0.6577 \end{bmatrix}$$

HW1-Q2.py

```

1 #Question 2
2
3 #Asked ChatGPT the following:
4 #To solve a linear inverse problem (i.e.,  $Ax = b$ ) using the SVD,  $A = U\Sigma V^T$ ,
5 # we typically solve three linear systems for the problem  $x = A^{-1}b = V\Sigma^{-1}U^Tb$ :
6 # Solve  $y = U^Tb$  for  $y$ .
7 # Solve  $z = \Sigma^{-1}y$ , where  $\Sigma^{-1} = \text{diag}(1/\sigma_i)$  for  $z$ .
8 # Solve  $x = Vz$  for  $x$ .
9 # Develop an algorithm in python without numpy using your SVD derivation that solves
  these three systems.
10 # Apply this to the previous problem to find  $x$  with  $A = [1\ 4\ 9\ 3\ 2\ 3\ 6\ 4\ 5]$ ,  $b =$ 
     $[18, 10, 19]^T$ 

```

want to get to $A\vec{x} = \vec{b} = U\Sigma V^T\vec{x} = \vec{b} \Rightarrow V\Sigma^{-1}U^T\vec{b}$

```

① 14 def matvec(A, x):
    15     return [sum(A[i][j] * x[j] for j in range(len(x)))
    16             for i in range(len(A))]
    17
    18 def transpose(A):
    19     return [list(row) for row in zip(*A)]
    20
    21
    22 def svd_solve(U, Sigma, V, b, tol=1e-12):
    23
    24
    25
    26     UT = transpose(U)
    27     y = matvec(UT, b)
    28
    29
    30     z = []
    31     for i in range(len(Sigma)):
    32         sigma = Sigma[i][i]
    33         if abs(sigma) < tol:
    34             z.append(0.0)
    35         else:
    36             z.append(y[i] / sigma)
    37
    38     x = matvec(V, z)
    39
    40
    41     return x
    42

```

Defn. matrix A and vector \vec{x}

also defined A as invertible
 $A^T A = I$

Defn the SVD decomp
 $A = U\Sigma V^T$

and y as $y = U^T b$

← singular values

$z = \Sigma^{-1} y$

$\vec{x} = Vz = V\Sigma^{-1}y = V\Sigma^{-1}U^T\vec{b}$

Apply to,

$$A = \begin{bmatrix} 1 & 4 & 9 \\ 3 & 2 & 3 \\ 6 & 4 & 5 \end{bmatrix}, \vec{b} = \begin{bmatrix} 18 \\ 10 \\ 19 \end{bmatrix} \Rightarrow \vec{x} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Question 3:

Asked ChatGPT the following:

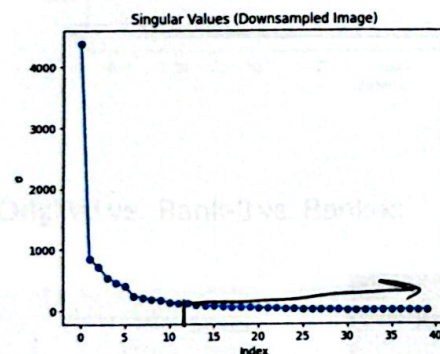
1. Construct the SVD in python without numpy of the image, which you can treat as a matrix A , with the SVD decomposition given by $A = U\Sigma V^T$
 2. Plot the singular values of the image, which should be a decreasing function. For each image, determine a reasonable location (call this k) at which the singular values level off and are no longer decreasing rapidly.
 3. Construct two matrices, A_3 and A_k $A_3 = \sum_{i=1}^3 u_i \sigma_i v_i^T$ and $A_k = \sum_{i=1}^k u_i \sigma_i v_i^T$ where k is the cutoff you found for the image.
 4. Visualize and compare A_3 and A_k to the full image, A .
- Also generate the errors, $E_3 = A_3 - A$ and $E_k = A_k - A$ and comment on where the errors are located.

Repeated the same for image 1 and image 2 in blackboard

Image 1:



Singular Value Graph:



⇒ Treated as $A \in \mathbb{R}^{40 \times 40}$

$$A = U \Sigma V^T \quad \left. \begin{array}{l} U = U_1 U_2 \\ A = U_1 B V_1^T \end{array} \right\} \quad \left. \begin{array}{l} V = V_1 V_2 \end{array} \right\}$$

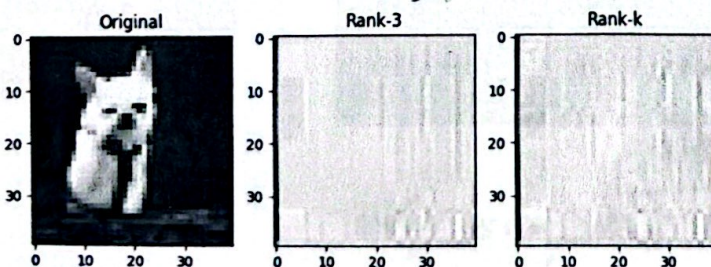
③ Rank-3 errors:

- edges
- facial featur
- background

Rank-12 errors:

- for texture
- facial
- background

Original vs. Rank-3 vs. Rank-k



still not as formed as original

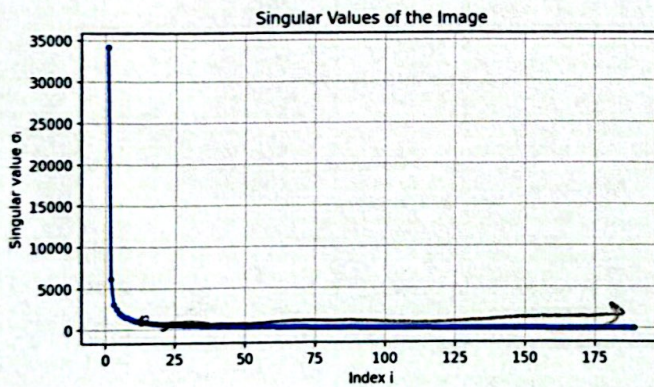
Image 2:

$$\textcircled{1} A \in \mathbb{R}^{40 \times 70}$$



$$A = U \Sigma V^T \quad \begin{cases} U = u_1, u_2 \\ \downarrow \\ A = U B V_1^T \end{cases} \quad \begin{cases} V = v_1, v_2 \end{cases}$$

Singular Value Graph:

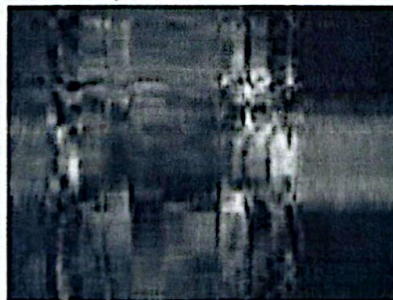


cut off at ~~rank~~ 14
because that is when
the graph levels

Original vs. Rank-3 vs. Rank-k:



Rank-k



Rank-3



ERRORS

• dog shape is
more visible
but not as
that of original

• cannot recognize
subject