Singular Value Decomposition

Name: Sanghyeon Kim

Student ID: 20181605

Date: 10/17/2020

The program performs SVD on a 2 by 3 matrix, $\boldsymbol{A}$, by first getting the left singular vectors, $\boldsymbol{U}$, and the singular value matrix $\boldsymbol{\Sigma}$. The symmetric matrix, $\boldsymbol{A}\boldsymbol{A}^\top$, and its eigen decomposition is used to find the left singular vectors and their corresponding sigma values, which are the square roots of the eigenvalues found by the decomposition. Then, using the left singular vectors and the singular values, the equation $\boldsymbol{v}_i = \frac{1}{\sigma_i}\boldsymbol{A}^\top\boldsymbol{u}_i$, the program finds the first and the second right singular vectors. However, using the equation above can only yield two right singular vectors corresponding to the two singular values. By the construction of the SVD, we know that the right singular vectors must have three linearly independent orthonormal vectors. Therefore, using the first two right singular vectors of the matrix, we can find a third vector $\boldsymbol{v}_3$ that is orthonormal to the plane formed by the first two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Using the first two orthonormal vectors, a linear system of equation equivalent to $\boldsymbol{v}_1\boldsymbol{v}_3 = 0, \boldsymbol{v}_2\boldsymbol{v}_3 = 0$ can be made. Because the first two vectors are linearly independent, the system is guaranteed to have two pivots. Thus, the solution will be a line formed by a vector. The normalized form of this vector is the third right singular vector. Now the left singular vectors $\boldsymbol{U}$, singular value matrix $\boldsymbol{\Sigma}$, and right singular vectors $\boldsymbol{V}$ have been found. Therefore, the SVD of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is completed.

The SVD of $\boldsymbol{A}$ could have been found by first getting right singular vectors. Then, finding left singular vectors could have been easier with the equation $\boldsymbol{u}_i = \frac{1}{\sigma_i}\boldsymbol{A}\boldsymbol{v}_i$. However, from experimentation, the eigen decomposition of a symmetric matrix appeared to be less numerically accurate as the size of the matrix increased. Therefore, the choice was made to use $\boldsymbol{A}\boldsymbol{A}^\top$ in order to use a two by two matrix instead of a three by three matrix.

```
# Name: svd.py
# Author: Sanghyeon Kim
# Date: 10/17/20

import numpy as np
import math

# Row reduce a 2*3 matrix into echelon form
def gaussian_2by3(matrix):
    for col_n in range(3):
        # Find the first nonzero column of the matrix
        if(matrix[0,col_n] == 0 and matrix[1,col_n] == 0):
            continue
        else:
            # Exchange rows of the matrix if the pivot is not the largest valu
e in the column
            if(matrix[0,col_n] < matrix[1,col_n]):
                new_matrix = np.array([matrix[1],matrix[0]])

            # Reduce the matrix into echelon form
            new_matrix[0] = new_matrix[0]/new_matrix[0,col_n]
            new_matrix[1] = new_matrix[1] - (new_matrix[0]*new_matrix[1,col_n]
)

            break

    return new_matrix

# Find the third right singular vector by construction
def find_v3(matrix):

    matrix = gaussian_2by3(matrix)

    # Find the first pivot column of the matrix
    for col_n in range(3):
        if(matrix[0,col_n] != 0):
            pivot_1_col = col_n
            break

    # Find the second pivot column of the matrix
    for col_n in range(3):
        if(matrix[1,col_n] != 0):
            pivot_2_col = col_n
            break

    # Find the nonpivot column of the matrix
    for col_n in range(3):
        if(col_n != pivot_1_col and col_n != pivot_2_col):
```

```
            free_col = col_n

    # Setting a free variable to 1, find the solution of the linear system
    b = -matrix[1,free_col]/matrix[1,pivot_2_col]
    a = -(matrix[0,pivot_2_col]*b + matrix[0,free_col])/matrix[0,pivot_1_col]
    c = 1

    # Find the norm of the third right singular vector
    norm = math.sqrt(a**2 + b**2 + c**2)

    row_3 = np.array([0,0,0], dtype=float)

    row_3[pivot_1_col] = a/norm
    row_3[pivot_2_col] = b/norm
    row_3[free_col] = c/norm

    return row_3

def main():
    row_1 = list(map(float, input("Type the elements of the first row of the m
atrix: ").split()))
    row_2 = list(map(float, input("Type the elements of the first row of the m
atrix: ").split()))

    matrix = np.array([row_1, row_2])

    # Make the transpose of the input matrix A
    matrix_tp = matrix.T

    # Create a symmetric matrix of the form A @ A^T to get the left singular v
ectors
    sym_matrix = np.matmul(matrix, matrix_tp)

    # Get the eigenvalues and the left singular vectors of A @ A^T, which is a
 real symmetric matrix
    eig_val, U = np.linalg.eigh(sym_matrix)

    # Create an empty matrix of the shape shape as A
    sigma = np.zeros_like(matrix)

    # Fill in Sigma with the ordered square roots of the nonzero eigenvalues
    for idx in range(len(eig_val)):
        sigma[idx, idx] = math.sqrt(eig_val[idx])

    # Find the first two right singular vectors
    r_singular_vectors = np.matmul(matrix_tp, U)
    row_1 = r_singular_vectors[:,0]/sigma[0,0]
    row_2 = r_singular_vectors[:,1]/sigma[1,1]
```

```
    submatrix = np.array([row_1, row_2])

    row_3 = find_v3(submatrix)

    V_T = np.array([row_1, row_2, row_3])

    print("\nU =", U)
    print("\nSigma =", sigma)
    print("\nV^T =",V_T)

main()
```