Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 9

по курсу «Численные методы линейной алгебры»

«Реализация сингулярного разложения матрицы (SVD)»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Целью работы является самостоятельное изучение сингулярного разложения матрицы или SVD разложения.

2 Задание

1. Реализовать SVD разложение матрицы;
2. Проверить корректность реализации прямым перемножением матриц;
3. Проверить через numpy.

3 Реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Исходный код

```
import numpy as np
from tabulate import tabulate

def generate_matrix(left: float, right: float, n: int, m: int) -> np.ndarray:
    return np.random.uniform(left, right, (n, m))

def print_matrix(a):
    for i in range(len(a)):
        print(a[i])

def svd(A):
    AtA = np.dot(A.T, A)
    eigenvalues, V = np.linalg.eig(AtA)

    singular_values = np.sqrt(eigenvalues)
    singular_vectors = V

    # U = A * V * S^-1
    left_singular_vectors = A.dot(singular_vectors) / singular_values
    # V
    right_singular_vectors = singular_vectors

    n = len(singular_values)
```

```
    S = np.zeros((n, n))

    for i in range(n):
        S[i, i] = singular_values[i]

    return left_singular_vectors, S, right_singular_vectors.T


def ordinary_matrix_mul(matrix1, matrix2):
    rows_matrix1 = len(matrix1)
    cols_matrix1 = len(matrix1[0])
    cols_matrix2 = len(matrix2[0])
    rows_matrix2 = len(matrix2)

    if rows_matrix1 != cols_matrix2:
        return f'                :
                , . .                  {
    rows_matrix1}x{cols_matrix1}    {rows_matrix2}x{cols_matrix2}'
    else:
        res = []
        for i in range(0, rows_matrix1):
            tmp = []
            for j in range(0, cols_matrix2):
                el = 0
                for k in range(cols_matrix1):
                    el += matrix1[i][k] * matrix2[k][j]
                tmp.append(el)
            res.append(tmp)
        return np.array(res)



A = generate_matrix(left=1, right=10, n=3, m=3)
U, S, V_T = svd(A)
check_classic = ordinary_matrix_mul(ordinary_matrix_mul(U, S), V_T)

U_np, singular_values_np, V_T_np = np.linalg.svd(A)
ns = len(singular_values_np)
s = [[0] * ns for _ in range(ns)]
for i in range(ns):
    s[i][i] = singular_values_np[i]

print('Original matrix A:')
print(A)

print('\nU matrix:')
```

```
print(U)
print('\nU matrix from numpy SVG:')
print(U_np)

print('\nSigma matrix:')
print(S)
print('\nSigma matrix from numpy SVG:')
print(S)

print('\nV_T matrix:')
print(V_T)
print('\nV_T matrix from numpy SVG:')
print(V_T_np)


print('\nCheck classy mul matrix A = USV_T:')
print(check_classic)
print('\nCheck numpy A = USV_T')
print(U @ S @ V_T)

print(f'\ncheck that U ortogonal')
I_n = U.T @ U
print_matrix(I_n)

print('check that V ortogonal')

I_m = V_T @ V_T.T
print_matrix(I_m)
```

4 Результаты

На рисунке 1 представлены:

1. исходная матрица размером 3x3;
2. матрицы U, Σ, V^T, вычисленные с помощью реализованного алгоритма SVD и с помощью метода библиотеки NumPy;
3. результаты умножения этих матриц, полученные с помощью стандартного алгоритма перемножения матриц и с помощью библиотечного метода.

Полученные произведения матриц доказали корректность реализации метода SVD.

```
Original matrix A:
[[9.95051791 6.88333031 1.89379859]
 [1.83952968 9.84753223 2.54786148]
 [8.63528218 1.45508404 5.92448686]]

U matrix:
[[-0.7012773  -0.05761986 -0.71055619]
 [-0.47284725  0.78351555  0.40313629]
 [-0.53350317 -0.61869487  0.57670706]]

U matrix from numpy SVG:
[[-0.7012773   0.05761986 -0.71055619]
 [-0.47284725 -0.78351555  0.40313629]
 [-0.53350317  0.61869487  0.57670706]]

Sigma matrix:
[[17.11148771  0.          0.        ]
 [ 0.          8.02409557  0.        ]
 [ 0.          0.          3.38003931]]

Sigma matrix from numpy SVG:
[[17.11148771  0.          0.        ]
 [ 0.          8.02409557  0.        ]
 [ 0.          0.          3.38003931]]

V_T matrix:
[[-0.7278642  -0.59958514 -0.33273318]
 [-0.55765189  0.79994375 -0.22161762]
 [-0.39904646 -0.02424176  0.9166102 ]]

V_T matrix from numpy SVG:
[[-0.7278642  -0.59958514 -0.33273318]
 [ 0.55765189 -0.79994375  0.22161762]
 [-0.39904646 -0.02424176  0.9166102 ]]

Check classy mul matrix A = USV_T:
[[9.95051791 6.88333031 1.89379859]
 [1.83952968 9.84753223 2.54786148]
 [8.63528218 1.45508404 5.92448686]]

Check numpy A = USV_T
[[9.95051791 6.88333031 1.89379859]
 [1.83952968 9.84753223 2.54786148]
 [8.63528218 1.45508404 5.92448686]]
```

Рис. 1 — Результаты методов SVD

Дополнительно была проведена проверка матриц U, V на ортогональность, котоая показала, что полученные матрицы действительно ортогональны и равенства $UU^T = E, VV^T = E$ выполнены.

```
check that U ortogonal
[ 1.00000000e+00  1.99310768e-17 -7.34784208e-16]
[1.99310768e-17 1.00000000e+00 9.83053066e-16]
[-7.34784208e-16  9.83053066e-16  1.00000000e+00]
check that V ortogonal
[ 1.00000000e+00  3.60822483e-16 -1.11022302e-16]
[3.60822483e-16 1.00000000e+00 3.05311332e-16]
[-1.11022302e-16  3.05311332e-16  1.00000000e+00]
```

Рис. 2 — Проверка матриц на ортогональность

5 Выводы

В результте выполнения данной работы был реализован метод сингулярного разложения матриц или SVD разложение на языке программирования Python. Также была проведена проверка матриц U и V из данного метода на ортогональность. Сравнение реализованного метода с библиотечным позволили сделать вывод о том, что собственно реализованный метод выполнен корректно.