**AKADEMIA GÓRNICZO-HUTNICZA**

**Wydział Informatyki**

Projekt nr 4
Z przedmiotu Rachunek Macierzowy

# *"Wartości i wektory własne. Dekompozycja na wartości osobliwe"*

Autor: *Jacek Tyszkiewicz i Michał Godek*
Kierunek studiów: *Informatyka*

Kraków, 2025

# Spis treści

# 1. Wartości i wektory własne. Dekompozycja na wartości osobliwe

## 1.1. Cel projektu

Celem projektu jest ręczne przeprowadzenie rozkładu SVD ($A = USV^T$) dla dowolnej macierzy prostokątnej $A \in \mathbb{R}^{n \times m}$ przy użyciu wybranego języka programowania, z wykorzystaniem podstawowych operacji algebry liniowej oraz bibliotek numerycznych.

## 1.2. Zadania

1. Wypisać lub narysować (np. `spy(A)` w MATLAB) macierz $A$.
2. Obliczyć i wypisać/narysować macierz $AA^T$ ($n \times n$).
3. Obliczyć (korzystając z biblioteki) wartości własne $\lambda_i$ i wektory własne $U_i$ macierzy $AA^T$.
4. Zbudować macierz wektorów własnych $[U_1\ U_2\ \ldots\ U_n]$ oraz diagonalną macierz $S$ taką, że $S_{ii} = \sqrt{\lambda_i}$.
5. Obliczyć macierz $V$ wykorzystując własność $V = A^T U S^{-1}$.
6. Wypisać macierz $[V_1\ V_2\ \ldots\ V_m]^T$ (czyli wektory $V_i$ jako wiersze).
7. Obliczyć i wypisać/narysować macierz $A^T A$ ($m \times m$).
8. Obliczyć (korzystając z biblioteki) wartości własne $\lambda_i$ i wektory własne $V_i$ macierzy $A^T A$.
9. Wypisać macierz wektorów własnych $[V_1\ V_2\ \ldots\ V_m]^T$ oraz macierz diagonalną $S$ taką, że $S_{ii} = \sqrt{\lambda_i}$.
10. Obliczyć macierz $U$ korzystając z własności $U = AVS^{-1}$, pamiętając że $S_{ii}^{-1} = \frac{1}{S_{ii}}$.

**11.** Wypisać macierz $[U_1\ U_2\ \ldots\ U_m]$.

**12.** Porównać uzyskane dwie dekompozycje (z $AA^T$ oraz $A^TA$).

**13.** Wyznaczyć: $\dim R(A)$ oraz $\dim N(A)$, gdzie $R(A)$ to przestrzeń obrazu operatora $A$, a $N(A)$ to jego jądro (null space).

## 1.3. Wypisać lub narysować macierz $A$.

```
import matplotlib.pyplot as plt
```
✓ 0.0s

```
np.random.seed(0)
n, m = 6, 4
A = np.random.randn(n, m)
A
```
✓ 0.0s

```
array([[ 1.76405235,  0.40015721,  0.97873798,  2.2408932 ],
       [ 1.86755799, -0.97727788,  0.95008842, -0.15135721],
       [-0.10321885,  0.4105985 ,  0.14404357,  1.45427351],
       [ 0.76103773,  0.12167502,  0.44386323,  0.33367433],
       [ 1.49407907, -0.20515826,  0.3130677 , -0.85409574],
       [-2.55298982,  0.6536186 ,  0.8644362 , -0.74216502]])
```

## 1.4. Obliczyć i wypisać/narysować macierz $AA^T$ $(n \times n)$.

```
AAT = A @ A.T
AAT
```
✓ 0.0s

```
array([[ 9.25153684,  3.49411755,  3.38207302,  2.57335386,  0.94601205,
        -5.05911349],
       [ 3.49411755,  5.36842191, -0.67729667,  1.67357708,  3.41749149,
        -4.47300068],
       [ 3.38207302, -0.67729667,  2.31490524,  0.52059552, -1.43544822,
        -0.42290296],
       [ 2.57335386,  1.67357708,  0.52059552,  0.90233635,  0.96605732,
        -1.72734248],
       [ 0.94601205,  3.41749149, -1.43544822,  0.96605732,  3.10185311,
        -3.04395688],
       [-5.05911349, -4.47300068, -0.42290296, -1.72734248, -3.04395688,
         8.24303313]])
```

## 1.5. Obliczyć (korzystając z biblioteki) wartości własne $\lambda_i$ i wektory własne $U_i$ macierzy $AA^T$.

```
wartosci_wlasne, wektory_wlasne = la.eigh(AAT)

indeksy_sortowania = np.argsort(wartosci_wlasne)[::-1]

wartosci_wlasne = wartosci_wlasne[indeksy_sortowania]
wektory_wlasne = wektory_wlasne[:, indeksy_sortowania]


print("Wartości własne λᵢ macierzy A·Aᵀ:")
print(wartosci_wlasne)

print("\nWektory własne Uᵢ (kolumny macierzy U):")
print(wektory_wlasne)
```

✓ 0.0s

```
Wartości własne λᵢ macierzy A·Aᵀ:
[ 1.86722494e+01  7.61863674e+00  2.46396324e+00  4.27237224e-01
  6.94573724e-16 -1.17867187e-17]

Wektory własne Uᵢ (kolumny macierzy U):
[[-0.59390984 -0.57165236  0.25529267 -0.18762865  0.46915437  0.        ]
 [-0.43689271  0.36888928  0.48893504  0.64652746 -0.11107838  0.06043033]
 [-0.10435984 -0.52391629 -0.07491713  0.12325528 -0.68042914  0.48045029]
 [-0.20070058 -0.0455066   0.20141986 -0.28402251 -0.53271118 -0.74338985]
 [-0.24931362  0.46218713  0.22349074 -0.66809308 -0.14125073  0.45604553]
 [ 0.58571911 -0.21669841  0.77436266 -0.06774007  0.02896509  0.07006921]]
```

## 1.6. Zbudować macierz wektorów własnych $[U_1\ U_2\ \ldots\ U_n]$ oraz diagonalną macierz $S$ taką, że $S_{ii} = \sqrt{\lambda_i}$.

```
S_diag = np.sqrt(np.clip(wartosci_wlasne, 0, None))
S = np.zeros((n, m))
np.fill_diagonal(S, S_diag[:min(n, m)])
print('Macierz U:')
print(wektory_wlasne)
print('\nMacierz S:')
print(S)
```

```
Macierz U:
[[-0.59390984 -0.57165236  0.25529267 -0.18762865  0.46915437  0.        ]
 [-0.43689271  0.36888928  0.48893504  0.64652746 -0.11107838  0.06043033]
 [-0.10435984 -0.52391629 -0.07491713  0.12325528 -0.68042914  0.48045029]
 [-0.20070058 -0.0455066   0.20141986 -0.28402251 -0.53271118 -0.74338985]
 [-0.24931362  0.46218713  0.22349074 -0.66809308 -0.14125073  0.45604553]
 [ 0.58571911 -0.21669841  0.77436266 -0.06774007  0.02896509  0.07006921]]

Macierz S:
[[4.32113982 0.         0.         0.        ]
 [0.         2.76018781 0.         0.        ]
 [0.         0.         1.56970164 0.        ]
 [0.         0.         0.         0.65363386]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
```

## 1.7. Obliczyć macierz $V$ wykorzystując własność $V = A^T U S^{-1}$.

```
# Wyciągnij tylko te kolumny U, które odpowiadają niezerowym wartościom singularnym
r = np.sum(S_diag > 1e-12)  # liczba niezerowych wartości
U_r = wektory_wlasne[:, :r]
S_r_inv = np.diag(1.0 / S_diag[:r])


V = A.T @ U_r @ S_r_inv


V, _ = np.linalg.qr(V)

print("Macierz V:")
print(V)
```

```
Macierz V:
[[-0.89638583  0.34190289  0.07551881 -0.27183039]
 [ 0.12867524 -0.37909523 -0.04992386 -0.91500661]
 [-0.15556544 -0.12582912 -0.97621261  0.08351864]
 [-0.39463212 -0.8506211   0.19701131  0.28617438]]
```

## 1.8. Wypisać macierz $[V_1\ V_2\ \ldots\ V_m]^T$ (czyli wektory $V_i$ jako wiersze).

```
print("Macierz V wypisana wierszami:")
for i, wiersz in enumerate(V):
    print(f"V_{i+1}ᵀ =", np.round(wiersz, 4))  # zaokrąglenie do 4 miejsc
```

```
Macierz V wypisana wierszami:
V_1ᵀ = [-0.8964  0.3419  0.0755 -0.2718]
V_2ᵀ = [ 0.1287 -0.3791 -0.0499 -0.915 ]
V_3ᵀ = [-0.1556 -0.1258 -0.9762  0.0835]
V_4ᵀ = [-0.3946 -0.8506  0.197   0.2862]
```

## 1.9. Obliczyć i wypisać/narysować macierz $A^T A$ ($m \times m$).

```
ATA = A.T @ A
ATA
```

```
array([[15.93951535, -3.04421137,  2.08466999,  4.39286802],
       [-3.04421137,  1.76790097,  0.07708295,  1.37248191],
       [ 2.08466999,  0.07708295,  2.92362049,  1.49808498],
       [ 4.39286802,  1.37248191,  1.49808498,  8.55104977]])
```

## 1.10. Obliczyć (korzystając z biblioteki) wartości własne $\lambda_i$ i wektory własne $V_i$ macierzy $A^T A$.

```
ATA = A.T @ A

wartosci_wlasne_V, wektory_wlasne_V = la.eigh(ATA)

indeksy_sortowania_V = np.argsort(wartosci_wlasne_V)[::-1]
wartosci_wlasne_V = wartosci_wlasne_V[indeksy_sortowania_V]
wektory_wlasne_V = wektory_wlasne_V[:, indeksy_sortowania_V]

print("Wartości własne λᵢ macierzy Aᵀ·A:")
print(wartosci_wlasne_V)

print("\nWektory własne Vᵢ (kolumny macierzy V):")
print(wektory_wlasne_V)
```

```
Wartości własne λᵢ macierzy Aᵀ·A:
[18.67224937  7.61863674  2.46396324  0.42723722]

Wektory własne Vᵢ (kolumny macierzy V):
[[ 0.89638583  0.34190289  0.07551881 -0.27183039]
 [-0.12867524 -0.37909523 -0.04992386 -0.91500661]
 [ 0.15556544 -0.12582912 -0.97621261  0.08351864]
 [ 0.39463212 -0.8506211   0.19701131  0.28617438]]
```

## 1.11. Wypisać macierz wektorów własnych $[V_1\ V_2\ \ldots\ V_m]^T$ oraz macierz diagonalną $S$ taką, że $S_{ii} = \sqrt{\lambda_i}$.

```
# 1. Wypisz macierz Vᵀ – czyli wektory własne Vᵢ jako wiersze
print("Macierz wektorów własnych Vᵢᵀ (wierszami):")
for i, kolumna in enumerate(wektory_wlasne_V.T):  # kolumny jako wiersze
    print(f"V_{i+1}ᵀ =", np.round(kolumna, 4))

# 2. Oblicz macierz diagonalną S na podstawie λᵢ (dla Aᵀ·A)
S_diag_V = np.sqrt(np.clip(wartosci_wlasne_V, 0, None))  # dla bezpieczeństwa
S_macierz_V = np.diag(S_diag_V)

# 3. Wypisz przekątną macierzy S
print("\nPrzekątna macierzy S (pierwiastki z λᵢ):")
print(np.round(S_diag_V, 4))
```

```
Macierz wektorów własnych Vᵢᵀ (wierszami):
V_1ᵀ = [ 0.8964 -0.1287  0.1556  0.3946]
V_2ᵀ = [ 0.3419 -0.3791 -0.1258 -0.8506]
V_3ᵀ = [ 0.0755 -0.0499 -0.9762  0.197 ]
V_4ᵀ = [-0.2718 -0.915   0.0835  0.2862]

Przekątna macierzy S (pierwiastki z λᵢ):
[4.3211 2.7602 1.5697 0.6536]
```

## 1.12. Obliczyć macierz $U$ korzystając z własności $U = AVS^{-1}$, pamiętając że $S_{ii}^{-1} = \frac{1}{S_{ii}}$.

```
r = np.sum(wartosci_wlasne_V > 1e-12)


V_r = wektory_wlasne_V[:, :r]
S_r_inv = np.diag(1.0 / np.sqrt(wartosci_wlasne_V[:r]))


U_odtworzone = A @ V_r @ S_r_inv
```

## 1.13. Wypisać macierz $[U_1\ U_2\ \ldots\ U_m]$.

```
print("Macierz U obliczona z AVS⁻¹:")
print(np.round(U_odtworzone, 4))
```

```
Macierz U obliczona z AVS⁻¹:
[[ 0.5939 -0.5717 -0.2553 -0.1876]
 [ 0.4369  0.3689 -0.4889  0.6465]
 [ 0.1044 -0.5239  0.0749  0.1233]
 [ 0.2007 -0.0455 -0.2014 -0.284 ]
 [ 0.2493  0.4622 -0.2235 -0.6681]
 [-0.5857 -0.2167 -0.7744 -0.0677]]
```

## 1.14. Porównać uzyskane dwie dekompozycje (z $AA^T$ oraz $A^TA$).

```
U_spektralna = wektory_wlasne[:, :r]
U_z_AVS = U_odtworzone[:, :r]

# Dopasuj znaki kolumn U_z_AVS do U_spektralna
U_z_AVS_aligned = U_z_AVS.copy()

for i in range(r):
    # Jeśli kierunki wektorów są przeciwne, zmień znak
    if np.dot(U_z_AVS[:, i], U_spektralna[:, i]) < 0:
        U_z_AVS_aligned[:, i] *= -1

# Teraz porównanie ma sens
print("||U (z AVS⁻¹) - U (z AAᵀ)|| =", np.linalg.norm(U_z_AVS_aligned - U_spektralna))
```

```
||U (z AVS⁻¹) - U (z AAᵀ)|| = 5.462937011122305e-15
```

## 1.15. Wyznaczyć: $\dim R(A)$ oraz $\dim N(A)$, gdzie $R(A)$ to przestrzeń obrazu operatora $A$, a $N(A)$ to jego jądro (null space).

- **R(A)** (range) – przestrzeń kolumn, ranga = liczba niezerowych wartości singularnych.
- **N(A)** (null space) – jądro operatora.

```
rank = np.linalg.matrix_rank(A, tol=1e-10)
dim_range = rank
dim_null = m - rank
print(f'dim R(A) = {dim_range}')
print(f'dim N(A) = {dim_null}')
```

```
dim R(A) = 4
dim N(A) = 0
```

## 3. Wyniki

– Macierze $U$, $\Sigma$ i $V$ zostały obliczone i porównane przy użyciu dwóch metod.

– Obie metody dają spójne wyniki (z uwzględnieniem możliwej zmiany znaku wektorów własnych).

– Wymiary przestrzeni obrazów i jądra zostały zidentyfikowane:

  – **Ranga (Rank)** = liczba niezerowych wartości singularnych.

  – **Jądro (Null space)** = przestrzeń odpowiadająca zerowym wartościom singularnym.

## Wnioski:

– Rozkład SVD pozwala precyzyjnie analizować strukturę macierzy, nawet jeśli nie jest kwadratowa.

– SVD daje wgląd w:

  – kierunki największej wariancji (istotne w PCA),

  – rangi i stabilność numeryczną macierzy,

  – możliwość kompresji i aproksymacji macierzy (np. przez obcięcie do największych singularnych).

– Różne metody prowadzą do tych samych wartości singularnych, co świadczy o matematycznej spójności i możliwości wyboru metody zależnie od kontekstu.