Мататематички методи за машинско учење 2023

Домаћи задатак број 4

```
import numpy as np
import numpy.random as rndm
import matplotlib as mplb
import matplotlib.pyplot as plt
```

Задатак 1. а) Неким од критеријума за позитивну дефинитност проверити да ли је матрица A позитивно дефинитна.

(5 поена)

```
import numpy as np
import numpy.random as rndm

# Definisanje matrice A
A = rndm.uniform(-2, 3, (50, 50))
A = np.tril(A) + (np.tril(A)).T

# Izračunavanje sopstvenih vrednosti matrice A
sopstvene_vrednosti = np.linalg.eigvals(A)

# Provera da li su sve sopstvene vrednosti strogo pozitivne
pozitivno_definitna = np.all(sopstvene_vrednosti > 0)

print("Matrica A je pozitivno definitna:", pozitivno_definitna)
```

```
Matrica A je pozitivno definitna: False
```

б) Какве су дефинитности матрице $A\,A^T$ и $A^T\,A$ ако је $A \in M_{m\times n}$ произвољна матрица пуног ранга колона?

(5 поена)

matrica $A^T\,A$ će uvek biti pozitivno semidefinitna za matricu A punog ranga kolona, dok će matrica $A\,A^T$ biti pozitivno semidefinitna ako je rang matrice A jednak m. U slučaju kada je rang matrice A manji od m, matrica $A\,A^T$ može biti indefinitna.

```
import numpy as np

# Postavljanje dimenzija matrice A (m > n za pun rang kolona)
m, n = 6, 4

# Generisanje proizvoljne matrice A dimenzija m x n
A = np.random.rand(m, n)

# Provera da li je rang matrice A jednak n (puni rang kolona)
```

```
rang = np.linalg.matrix_rank(A)
print("Rang matrice A:", rang)

# Izračunavanje matrica AA^T i A^TA
AAt = A @ A.T
AtA = A.T @ A

# Izračunavanje sopstvenih vrednosti matrica AA^T i A^TA
sopstvene_vrednosti_AAt = np.linalg.eigvals(AAt)
sopstvene_vrednosti_AtA = np.linalg.eigvals(AtA)

# Provera definitnosti matrica AA^T i A^TA
definitnost_AAt = "pozitivno semidefinitna" if
np.all(sopstvene_vrednosti_AAt >= 0) else "indefinitna"
definitnost_AtA = "pozitivno semidefinitna" if
np.all(sopstvene_vrednosti_AtA >= 0) else "indefinitna"

print("Matrica AA^T je", definitnost_AAt)
print("Matrica A^TA je", definitnost_AtA)
```

```
Rang matrice A: 4
Matrica AA^T je indefinitna
Matrica A^TA je pozitivno semidefinitna
```

Задатак 2. а) Која од следећих матрица је косо-симетрична?

$$A=\begin{pmatrix} 1 & -3 & 2 \\ 3 & -5 & 4 \\ -2 & -4 & 2 \end{pmatrix}, B=\begin{pmatrix} 0 & -3 & 2 \\ 3 & 0 & 4 \\ -2 & -4 & 0 \end{pmatrix}.$$

(5 поена)

```
import numpy as np

A = np.array([[1, -3, 2],
              [3, -5, 4],
              [-2, -4, 2]])

B = np.array([[0, -3, 2],
              [3, 0, 4],
              [-2, -4, 0]])

# Provera da li je matrica A kososimetrična
kososimetrična_A = np.all(A.T == -A)
print("Matrica A je kososimetrična:" , kososimetrična_A)

# Provera da li je matrica B kososimetrična
kososimetrična_B = np.all(B.T == -B)
print("Matrica B je kososimetrična:" , kososimetrična_B)
```

```
Matrica A je kososimetrična: False
Matrica B je kososimetrična: True
```

б) Ако је A косо-симетрична матрица чему је једнако $A^T A^{-1}$?

(5 поена)

```
import numpy as np

# Kreiranje kososimetrične matrice A
A = np.array([[1, -3, 2],
              [3, -5, 4],
              [-2, -4, 2]])

# Izračunavanje transponovane matrice A
A_transpose = A.T

# Provera da li je A kososimetrična matrica (A^T = -A)
if np.array_equal(A_transpose, -A):
    # Ako je kososimetrična, izračunaj A^T * A^(-1)
    A_inverse = np.linalg.inv(A)
    C = A_transpose @ A_inverse
    print("Matrica A^TA^(-1) je:")
    print(C)
else:
    print("Matrica A nije kososimetrična.")
```

```
Matrica A nije kososimetrična.
```

```
import numpy as np

# Kreiranje kososimetrične matrice A
A = np.array([[0, -3, 2],
              [3, 0, 4],
              [-2, -4, 0]])

# Izračunavanje transponovane matrice A
A_transpose = A.T

# Provera da li je A kososimetrična matrica (A^T = -A)
if np.array_equal(A_transpose, -A):
    # Ako je kososimetrična, izračunaj A^T * A^(-1)
    A_inverse = np.linalg.inv(A)
    C = A_transpose @ A_inverse
    print("Matrica A^TA^(-1) je:")
    print(C)
else:
    print("Matrica A nije kososimetrična.")
```

```
Matrica A^TA^(-1) je:
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Задатак 3. а) Показати да се свака матрица може представити као збор једне симетричне и једне косо-симетричне матрице.

(5 поена)

а) Да бисмо показали да се свака матрица може представити као збор једне симетричне и једне косо-симетричне матрице, можемо користити следећи разломак матрице:

Симетрична матрица: $S=\frac{1}{2}\left(A+A^T\right)$ Косо-симетрична матрица: $K=\frac{1}{2}\left(A-A^T\right)$

Тако да је збир ове две матрице:

$$A=S+K=\frac{1}{2}\left(A+A^T\right)+\frac{1}{2}\left(A-A^T\right)=\frac{1}{2}\left(A+A\right)=A$$

Дакле, свака матрица може да се представи као збор једне симетричне и једне косо-симетричне матрице.

Следећи код демонстрира ово тврђење:

```python
import numpy as np

def decompose_matrix(A):
    A_T = A.T
    S = 0.5 * (A + A_T)
    K = 0.5 * (A - A_T)
    return S, K

# Primer matrice
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Dekompozicija matrice
S, K = decompose_matrix(A)

print("Simetrična matrica S:\n", S)
print("Koso-simetrična matrica K:\n", K)
print("A = S + K:\n", S + K)
```

```
Simetrična matrica S:
 [[1. 3. 5.]
 [3. 5. 7.]
 [5. 7. 9.]]
```

```
Koso-simetrična matrica K:
 [[ 0. -1. -2.]
 [ 1.  0. -1.]
 [ 2.  1.  0.]]
A = S + K:
 [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

б) Проверити позитивну дефинитност несиметричне квадратне матрице A неким од критеријума.

(5 поена)

```
import numpy as np

A = np.random.uniform(-2, 3, size=(50, 50))
eigenvalues = np.linalg.eigvals(A)

if np.all(eigenvalues > 0):
    print("Matrica A je simetrično pozitivno definitna.")
else:
    print("Matrica A nije simetrično pozitivno definitna.")
```

```
Matrica A nije simetrično pozitivno definitna.
```

Задатак 4. За матрицу $A=B^TB$ одредити матрицу пројекције на $R(A)$ у односу на скаларни производ $(\cdot,\cdot)_C$.

(10 поена)

```
B=rndm.normal(-1,2,(25,5))
B=B@np.array([[1,0,1,-1,2],[0,1,1,1,-1],[0,0,0,0,0],[0,0,0,0,0],
[0,0,0,0,0]])
A=B.T@B
A
```

```
array([[  68.27602794,   -8.94409941,   59.33192853,  -77.22012735,
         145.49615529],
       [  -8.94409941,  129.41651021,  120.47241079,  138.36060962,
        -147.30470903],
       [  59.33192853,  120.47241079,  179.80433932,   61.14048227,
          -1.80855374],
       [ -77.22012735,  138.36060962,   61.14048227,  215.58073697,
        -292.80086433],
       [ 145.49615529, -147.30470903,   -1.80855374, -292.80086433,
         438.29701962]])
```

```
np.linalg.matrix_rank(A)
```

```
2
```

```
C=rndm.rand(5,5)
C=np.tril(C)+(np.tril(C)).T+5*np.eye(5,5)

np.linalg.matrix_rank(C)
```

```
5
```

```
import numpy as np
import numpy.random as rndm

# Generisanje matrice B
B = rndm.normal(-1, 2, (25, 5))
B = B @ np.array([[1, 0, 1, -1, 2], [0, 1, 1, 1, -1], [0, 0, 0, 0, 0],
[0, 0, 0, 0, 0], [0, 0, 0, 0, 0]])

# Izračunavanje matrice A
A = B.T @ B
print("Matrica A:\n", A)

# Rang matrice A
rank_A = np.linalg.matrix_rank(A)
print("Rang matrice A:", rank_A)

# Generisanje matrice C
C = rndm.rand(5, 5)
C = np.tril(C) + (np.tril(C)).T + 5 * np.eye(5, 5)
print("Matrica C:\n", C)

# Rang matrice C
rank_C = np.linalg.matrix_rank(C)
print("Rang matrice C:", rank_C)

# Izračunavanje matrice projekcije P
CA_inv = np.linalg.inv(C @ A)  # Inverz matrice (C * A)
C_inv = np.linalg.inv(C)       # Inverz matrice C

P = A @ CA_inv @ C_inv          # Matrica projekcije P
print("Matrica projekcije P:\n", P)
```

```
Matrica A:
 [[ 141.05855303   12.46010085  153.51865388 -128.59845218
269.65700522]
 [  12.46010085   88.7815593   101.24166015   76.32145845  -63.8613576
]
 [ 153.51865388  101.24166015  254.76031403  -52.27699373
205.79564762]
 [-128.59845218   76.32145845  -52.27699373  204.91991063 -
333.51836282]
 [ 269.65700522  -63.8613576   205.79564762 -333.51836282
603.17536803]]
```

```
Rang matrice A: 2
Matrica C:
 [[6.29599512 0.1335287  0.73693724 0.56347077 0.94993816]
 [0.1335287  5.37799669 0.33152311 0.5344945  0.21045848]
 [0.73693724 0.33152311 6.78688224 0.92370471 0.51305958]
 [0.56347077 0.5344945  0.92370471 5.76481187 0.66256846]
 [0.94993816 0.21045848 0.51305958 0.66256846 6.61993313]]
Rang matrice C: 5
Matrica projekcije P:
 [[ 0.02421092 -0.01135185  0.00429209 -0.01265484 -0.00194601]
 [ 0.01608791  0.02349472 -0.01403281  0.0098244  -0.00410615]
 [ 0.00156723  0.01722347  0.01575283 -0.01710487  0.00223145]
 [-0.01463618  0.03299285 -0.01067163  0.01758939  0.00202267]
 [ 0.00347627 -0.06135716  0.01156715 -0.01407954  0.02068742]]
```

Задатак 5. Прилагодити метод коњугованих градијената за решење матричне једначине $AX=B$.

(10 поена)

```
def konjugovani_gradijenti(A, b, x0, tolerancija=1e-10,
maks_iteracija=None):
    m, n = A.shape

    # Inicijalizacija rešenja (vrednosti x)
    x = np.array(x0)

    # Izračunavanje reziduala
    r = b - A @ x

    # Kvadrat norme reziduala
    nr2 = np.inner(r, r)

    # Postavljanje vektora pretrage na rezidual
    v = np.copy(r)

    k = 0     # Brojač iteracija

    # Glavna iteracija metode konjugovanih gradijenata
    while (k < n) and not(np.array_equal(v, 0)) and (nr2 >
tolerancija):
        Av = A @ v
        t = nr2 / np.inner(v, Av)
        x = x + t * v
        r = r - t * Av
        c = nr2
        nr2 = np.inner(r, r)
        if nr2 > tolerancija:
            s = nr2 / c
            v = r + s * v
        k += 1
```

```
    return x

def resi_AX_B(A, B, x0, tolerancija=1e-10, maks_iteracija=None):
    # Inicijalizacija matrice rešenja X
    X = np.zeros_like(B)

    # Primena metode konjugovanih gradijenata na svaku kolonu matrice
B
    for i in range(B.shape[1]):
        X[:, i] = konjugovani_gradijenti(A, B[:, i], x0, tolerancija,
maks_iteracija)

    return X

# Testiranje funkcije resi_AX_B
A = np.array([[10, 1, 2, 3, 4],
              [1, 9, -1, 2, -3],
              [2, -1, 7, 3, -5],
              [3, 2, 3, 12, -1],
              [4, -3, -5, -1, 15]])

B = np.array([[12, -27, 14, -17, 12],
              [15, 30, -9, 20, -13],
              [8, 15, 25, 33, -5],
              [23, 12, -3, 12, -1],
              [14, -3, -5, -1, 15]])

n, _ = A.shape
x0 = np.zeros(n)

X = resi_AX_B(A, B, x0)
print("Rešenje X:")
print(X)
print("Provera: A @ X =")
print(A @ X)
print("B =")
print(B)
```

```
Rešenje X:
[[-1 -8  0 -8  2]
 [ 3  8  0  8 -2]
 [ 4 11  5 15 -2]
 [ 1  0 -1 -1  0]
 [ 3  7  1  8  0]]
Provera: A @ X =
[[ 16 -22  11 -13  14]
 [ 15  32 -10  23 -14]
 [ 11  18  27  38  -8]
 [ 24  18   2  17  -4]
```

```
 [ 11  -6  -9 -10  24]]
B =
[[ 12 -27  14 -17  12]
 [ 15  30  -9  20 -13]
 [  8  15  25  33  -5]
 [ 23  12  -3  12  -1]
 [ 14  -3  -5  -1  15]]
```