metody_matematyczne_prac2

January 15, 2024

1 Wybrane metody matematyczne w analizie danych - pracownia 2

1.0.1 Marcin Koźniewski

25 listopada 2023 Kilka przydatnych zasobów

- routines.linalg
- np.concatenate

1.0.2 Ćwiczenia

Ćwiczenie 1. Policz wyznacznik macierzy $A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$

[64]:
```
import numpy as np
from numpy.linalg import det
from numpy.linalg import inv
from math import sqrt,pi,cos,sin
import matplotlib.pyplot as plt

A=np.array( [[1, 2, 1], [1, 0, -1], [2,1,0]] )
det(A)
```

[64]:
```
-2.0
```

Ćwiczenie 2. Oblicz odwrotność macierzy A z ćwiczenia 1.

Wyznacznik macierzy A jest **różny od zera** więc macierz odwrotna do A istnieje.

[2]:
```
inv(A)
```

[2]:
```
array([[-0.5, -0.5,  1. ],
       [ 1. ,  1. , -1. ],
       [-0.5, -1.5,  1. ]])
```

Ćwiczenie 3. Sprawdź czy wektory $[1, 2, 3], [1, 2, 2], [0, 0, 1]$ są liniowo zależne.

[3]:
```
B=np.array( [[1, 2, 3], [1, 2, 2], [0, 0, 1]] )
det(B)
```

[3]: 0.0

Wyznacznik jest **równy 0**, zatem wskazane wektory są liniowo **zależne**.

Ćwiczenie 4. Sprawdź czy wektory $[1, 2, 3], [1, 2, 2], [1, 0, 0]$ są liniowo zależne.

[42]:
```
B=np.matrix( [[1, 2, 3], [1, 2, 2], [1, 0, 0]]  )

det(B)
B
```

[42]:
```
matrix([[1, 2, 3],
        [1, 2, 2],
        [1, 0, 0]])
```

Wyznacznik jest **różny od 0**, zatem wskazane wektory są liniowo **niezależne**.

Ćwiczenie 5. Policz normy wektorów $\mathbf{a} = [1, 2, 2]$ i $\mathbf{b} = [3, 0, 4]$. Następnie je znormalizuj.

Przypomnijmy, że normę liczymy za pomocą wzoru:

$$||\mathbf{a}|| = \sqrt{aa^T} = \sqrt{a_1a_1 + a_2a_2 + a_3a_3}$$

a normalizacja wektora polega na podzieleniu wektora przez jego normę

[5]:
```
a=np.array( [[1, 2, 2]] )
print(a)
print(a.T)
sqrt(a@a.T)
```

```
[[1 2 2]]
[[1]
 [2]
 [2]]
```

[5]: 3.0

[6]:
```
anorm = a/sqrt(a@a.T)

anorm
```

[6]: array([[0.33333333, 0.66666667, 0.66666667]])

[7]:
```
sqrt(anorm@anorm.T)
```

[7]: 1.0

[8]:
```
b=np.array( [[3, 0, 4]])
sqrt(b@b.T)
```

[8]: 5.0

[9]:
```
bnorm = b/sqrt(b@b.T)
print(bnorm)
print(bnorm@bnorm.T)
```

```
[[0.6 0.  0.8]]
[[1.]]
```

Ćwiczenie 6. Dokonaj rzutowania wektora $\mathbf{a} = [1, 2, 2]$ na wektor $\mathbf{b} = [3, 0, 4]$ i odwrotnie

[10]:
```
print(((a@b.T)/(b@b.T)))
(a@b.T)/(b@b.T) @ b
```

```
[[0.44]]
```

[10]:
```
array([[1.32, 0.  , 1.76]])
```

[11]:
```
((b@a.T)/(a@a.T)) @ a
```

[11]:
```
array([[1.22222222, 2.44444444, 2.44444444]])
```

Ćwiczenie 7. Podziałaj przekształceniem zdefiniowanym za pomocą poniższej macierzy i zobrazuj je wykresem:

$$\mathbf{a} = [1, 2]\mathbf{b} = [3, 0]\mathbf{c} = [3, 3]M = \begin{bmatrix} 2 & 0 \\ 0 & 2.5 \end{bmatrix}$$

[12]:
```
a = np.array( [[1, 2]] )
b = np.array( [[3, 0]] )
c = np.array( [(3, 3)] )
M = np.array( [[2, 0], [0, 2.5]])

Ma = M @ a.T
Mb = M @ b.T
Mc = M @ c.T
Ma, Mb, Mc
```

[12]:
```
(array([[2.],
        [5.]]),
 array([[6.],
        [0.]]),
 array([[6. ],
        [7.5]]))
```

Aby uprościć przetwarzanie możemy złożyć wektory przekształcane w macierz co pozwoli na wykonanie obliczeń na wszystkich wektorach na raz.

np.concatenate

[33]:
```
toplot = np.concatenate((a.T, b.T, c.T), axis=1)
print(toplot)
toplotM = np.concatenate((Ma, Mb, Mc), axis=1)
```

```
print(toplotM)

a.T
```

```
[[1 3 3]
 [2 0 3]]
[[2.  6.  6. ]
 [5.  0.  7.5]]
```

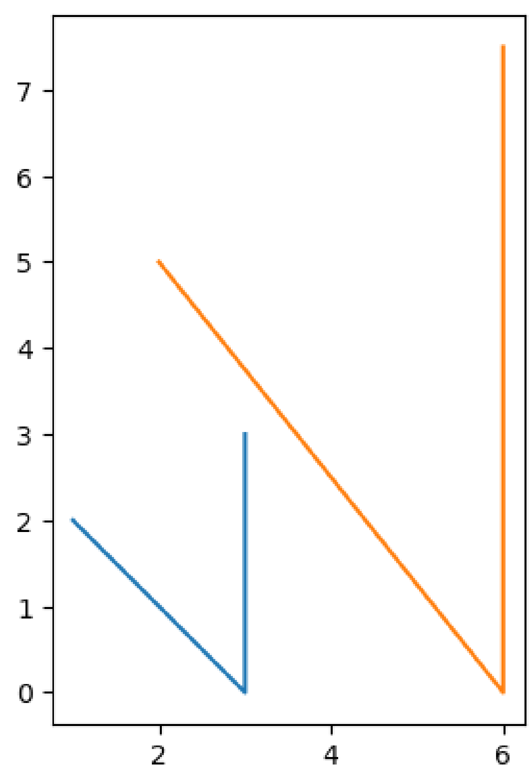
[33]:
```
array([[1],
       [2]])
```

I wyrysowanie

[28]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(toplot[0,], toplot[1,])
ax.plot(toplotM[0,], toplotM[1,])
ax.set_aspect('equal')

toplot
```

[28]:
```
array([[1, 3, 3],
       [2, 0, 3]])
```



Powyższe przekształcenie jest przekształceniem skalowania (na pomarańczowo są narysowane linie łączące przekształcone punkty.

1.0.3 Zadania samodzielne - praca domowa

Proszę sporządzić raport z wykonania poniższych zadań. Może to być prosty plik tekstowy lub notes Jupyter z wykonanymi poleceniami i wynikiem ich wywołania z dodatkiem komentarzy, jeśli są one potrzebne (wystąpił błąd).

Zadanie 1. Dla zadanych wektorów i macierzy

- $\mathbf{w} = [1, 2, 1, 0]$
- $\mathbf{u} = [1, 0, -1, 1]$

- $\mathbf{v} = [2, 1, 0, 0]$
- $A = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & -2 & 0 \end{bmatrix}$
- $B = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 0 & -1 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix}$

policz: 1. $A\mathbf{w}^T$ 2. $A\mathbf{u}^T$ 3. $A\mathbf{v}^T$ 4. AB^T

Porównaj i skomentuj wyniki (Zwróć uwagę na wektory i macierz B. Czym jest macierz B?)

Zadanie 2. Sprawdź za pomocą wyznacznika macierzy czy wektory * $\mathbf{v} = [1, -1, 1]$ * $\mathbf{w} = [1, 2, 3]$ * $\mathbf{u} = [2, -1, 2]$

są liniowo zależne

To samo wykonaj dla wektorów:

- $\mathbf{x} = [1, -1, 1]$
- $\mathbf{y} = [4, 4, 4]$
- $\mathbf{z} = [1, 2, 1]$

Zadanie 3. Policz normy wektorów v, w, u.

Zadanie 4. Dokonaj rzutowania wektorów x, y, z na każdy z wektorów u, v, w.

Zadanie 5. Znormalizuj wektory u, v, w. Wykonaj rzutowanie wektorów na x, y, z na powstałe wektory.

Zadanie 6. Podziałaj kolejno obydwoma przekształceniami zdefiniowanym za pomocą poniższych macierzy i zobrazuj je wykresem:

$$\mathbf{a} = [3, 1]\mathbf{b} = [1, 0]\mathbf{c} = [0, 2]\mathbf{d} = [0, 0]R = \begin{bmatrix} 2 & 0 \\ 0 & 2.5 \end{bmatrix} S = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix}$$

Zadanie 7. Dokonaj rzutowania wektoróów uzyskanych w zadaniu 6 na wektory

$$\mathbf{x} = [1, 1]\mathbf{y} = [1, -1]$$

Które współrzędne są równe (bliskie) 0?

[37]:
```
#Zad1
w = np.array([[1,2,1,0]])
u = np.array([[1,0,-1,1]])
v = np.array([[2,1,0,0]])
A = np.array([[1,0,1,2],[0,1,-2,0]])
B = np.array([[1,2,1,0],[1,0,-1,1],[2,1,0,0]])

mat1 = A@w.T
mat2 = A@u.T
mat3 = A@v.T
mat4 = A@B.T

# macierz B jest połaczeniem wektorow w,u,v
# kolejne przekształcenia są operacjami skalowania macierzy A
# zmienna mat4 przechowuje w każdej z kolumn wyniki skalowania wszystkimi wcześniejszymi wektorami
```

[47]:
```
#Zad2
 =np.array([[1,-1,1]])
 =np.array([[1,2,3]])
 =np.array([[2,-1,2]])
det(np.concatenate((w,u,v)))

#wektory niezależne
```

[47]: -2.0000000000000004

[52]:
```
#Zad3

normw = w / sqrt(w@w.T)
normv = v / sqrt(v@v.T)
normu = u / sqrt(u@u.T)

print(normw)
print(normv)
print(normu)

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

normx = x / sqrt(x@x.T)
normy = y / sqrt(y@y.T)
normz = z / sqrt(z@z.T)

print(normx)
print(normy)
```

```
print(normz)
```

```
[[0.26726124 0.53452248 0.80178373]]
[[ 0.57735027 -0.57735027  0.57735027]]
[[ 0.66666667 -0.33333333  0.66666667]]
[[ 0.57735027 -0.57735027  0.57735027]]
[[0.57735027 0.57735027 0.57735027]]
[[0.40824829 0.81649658 0.40824829]]
```

[54]:
```
#Zad4

rzutx_u = (x@u.T)/(u@u.T) @ u
rzutx_w = (x@w.T)/(w@w.T) @ w
rzutx_v = (x@v.T)/(v@v.T) @ v



rzuty_u = (y@u.T)/(u@u.T) @ u
rzuty_w = (y@w.T)/(w@w.T) @ w
rzuty_v = (y@v.T)/(v@v.T) @ v

rzutz_u = (z@u.T)/(u@u.T) @ u
rzutz_w = (z@w.T)/(w@w.T) @ w
rzutz_v = (z@v.T)/(v@v.T) @ v
```

[60]:
```
#Zad5
normw = w / sqrt(w@w.T)
normv = v / sqrt(v@v.T)
normu = u / sqrt(u@u.T)

rzutx_normu = (x@normu.T)/(normu@normu.T) @ normu
rzutx_normw = (x@normw.T)/(normw@normw.T) @ normw
rzutx_normv = (x@normv.T)/(normv@normv.T) @ normv

rzuty_normu = (y@normu.T)/(normu@normu.T) @ normu
rzuty_normw = (y@normw.T)/(normw@normw.T) @ normw
rzuty_normv = (y@normv.T)/(normv@normv.T) @ normv

rzutz_normu = (z@normu.T)/(normu@normu.T) @ normu
rzutz_normw = (z@normw.T)/(normw@normw.T) @ normw
rzutz_normv = (z@normv.T)/(normv@normv.T) @ normv
```

[73]:
```
import math
a=np.array([[3,1]])
b=np.array([[1,0]])
c=np.array([[0,2]])
d=np.array([[0,0]])

R=np.array([[2,0],[0,2.5]])
```

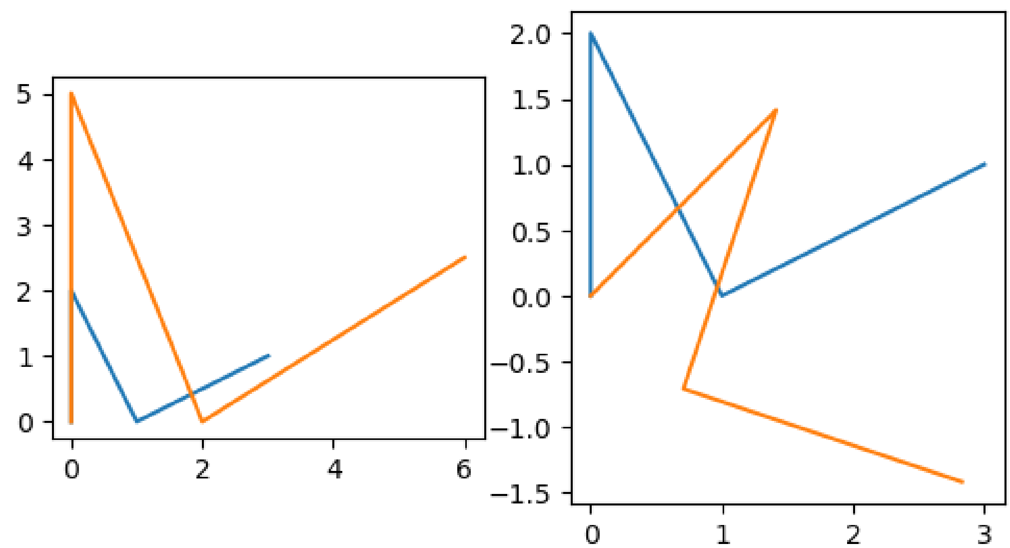
```
S = np.array([[cos(pi/4),sin(pi/4)],[-sin(pi/4),cos(pi/4)]])

abcd = np.concatenate((a,b,c,d))

skal1 = R @ abcd.T
skal2 = S @ abcd.T


fig = plt.figure()
ax = fig.add_subplot(121)
ax.plot(abcd[:,0], abcd[:,1])
ax.plot(skal1[0,], skal1[1,])
ax.set_aspect('equal')

ax2 = fig.add_subplot(122)
ax2.plot(abcd[:,0], abcd[:,1])
ax2.plot(skal2[0,], skal2[1,])
ax2.set_aspect('equal')
```



2 Kilka uwag:

łączenie wektorów w macierz

[22]:
```
np.concatenate((a,b), axis=0)
```

[22]:
```
array([[1, 2],
       [3, 0]])
```

Budowanie macierzy z wyliczonych wartości

[23]:
```python
import math
print(np.pi)
print(-np.cos(np.pi/4))
```

```
3.141592653589793
-0.7071067811865476
```

[25]:
```python
aa=-np.cos(np.pi/4)
```

[26]:
```python
np.array([[aa,aa],[aa,aa]])
```

[26]:
```
array([[-0.70710678, -0.70710678],
       [-0.70710678, -0.70710678]])
```