vjezbe-01-dio-1-rjesenja

March 9, 2022

1 Vježbe 1

1.1 Prvi dio

Teme:

1. ponavljanje linearne algebre i matematičke analize
2. upoznavanje s JupyterLab radnom okolinom
   - kako koristiti JupyterLab
   - Markdown
   - predaja zadaće
3. upoznavanje biblioteke Numpy
4. upoznavanje biblioteke Matplotlib
5. upoznavanje biblioteke scikit-learn

1. ponavljanje linearne algebre & matematičke analize U datotekama na Teams kanalu se nalazi datoteka Ponavljanje.pdf sa osnovnim pojmovima iz analize i linearne algebre.

2. Upoznavanje s JupyterLab radnom okolinom Za rješavanje zadataka na vježbama i za rješavanja/predaju zadaće koristit ćemo JupyterLab radnu okolinu. Jedan od načina kako možemo koristiti Jupyter je da ga lokalno instaliramo koristeći Anacondu ili Pip.

Alternativno, možete koristiti Google Colab (https://colab.research.google.com/notebooks/intro.ipynb#recent=tru - koristi se online - preporuka je da se ulogirate sa svojim Google računom - podatke koje ćete učitati možete staviti u direktorij koji se prikaže na lijevoj strani, **ali** ti podaci se brišu nakon kratkog vremena - bolji način -> podatke učitavate na vaš Google Drive, detaljne upute imate na internetu, npr: https://colab.research.google.com/notebooks/io.ipynb

Zašto koristimo JupyterLab (ili Colab) ?

- grafičko korisničko sučelje, jednostavno za koristiti
- sve mogućnosti Pythona
- sve dodatne biblioteke za Data Science, Machine Learning i sl.
- pisanje teksta (Markdown) i stručnog teksta (Latex)
- umetanje datoteka/slika/grafova

Kako ćete predavati zadaću?

- Zadaću ćete pisati u JupyterLab bilježnici (Colab bilježnici)
- Sve zadatke možete riješiti u jednoj bilježnici

- Programerske zadatke rješavate na uobičajen način (pišete kod u ćelijama bilježnice ili napišete skripte (.py datoteke) koje onda pozivate u ćelijama bilježnice)
- Teorijske zadatke pišete u Markdownu (Latexu) ili (ukoliko su neki složeni raspisi) skenirane/uslikane datoteke umetnene u Jupyter bilježnicu
- JupyterLab: Bilježnicu/zip file ili link na Google Colab bilježnicu postavljate na Teams->Assignments

Kako koristiti Markdown? Cijeli ovaj prethodni tekst je napisan u Markdownu. Markdown je markup jezik, super jednostavan za korištenje (s kojim ste se dosad markup jezicima susreli?) Jupyter bilježnica se sastoji od ćelija. Kada dodamo novu ćeliju možemo odabrati njen format, ponuđeni su: Markdown, Code, Raw. Mi ćemo koristiti Markdown za pisanje teksta i code za pisanje Python koda.

Nakon što ste dodali novu ćeliju i označili je kao Markdown, možete početi pisati tekst.

Naslove dodajemo stavljajući znak # na početak reda željeni broj puta, tako imamo naslove u različitim veličinama # H1 ## H2 ### H3 #### H4 ##### H5 ###### H6

Ako želimo razdovojiti tekst horizontalnom linijom preko cijele stranice napišemo tri uzastopne crtice — ili ____ na početak reda (dodani jedan prazan red)

U tekstu možemo dijelove naglašavati **podebljavanjem** na **dva načina** ili ukošenim *slovima* također na *dva načina*.

Ako želimo tekst posložiti u numeriranu ili nenumeriranu listu to radimo na sljedeći način: 1. stavimo redni broj 2. iza njega jedan razmak 3. i samo nižemo stavke sa liste 4. jednu ispod druge

Dok za neneumeriranu listu je dovoljno - staviti jednu crticu - i iza nje jedan razmak - također možemo korstiti * znak asterisk + ili znak plus

Liste možemo ugnijezditi koristeći dovoljan razmak (tab) 1. Zadatak 1. podzadatak 2. podzadatak 2. Zadatak 1. podzadatak - Treći zadatak - prvi podzadatak - drugi podzadatak - prvi podpodzadatak

Latex Unutar Markdowna možemo pisati Latex kod (tekst). Latex pišemo unutar simbola $

Primjeri: - $f(x) = \frac{1}{2}x^3$ - a_0^n - $A \in \mathbb{R}^{n \times n}$ - $||x||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

Jupyter bilježnicu možete izvesti u pdf ili html oblik, zatim kao Python skriptu i sl. - Upute: gore lijevo u izborniku File/Export notebook as

Upoznavanje biblioteka

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
import sklearn.datasets
```

1.1.1 Arrays

Zadatak 1: Kreiranje Numpy arraya

Shape Numpy arrays imaju atribut **shape** čiji je tip uređeni par. - array duljine n ima shape (n,) - matrica sa m redaka i n stupaca ima shape (m,n) - arrayji mogu imati više od dvije dimenzije (axis)

[ ]:
```
# mozemo konstruirati np.array koristeci listu ili tuple
a = np.array([1,2,3])
b = np.array([[1,2,3],[4,5,6]])
print(a)
print(b)

# ili komprehenziju listi
array = np.array([x/2 for x in range(1,5)])
print(array)

# svi numpy arrayi imaju svoj oblik (shape) u obliku tuplea
print(a, a.shape)

# mozemo dodavati i uklanjati dimenzije
a = a.reshape(-1,1)
print(a, a.shape)

a = a.squeeze(-1)
print(a, a.shape)

# mozemo mijenjati oblik arraya, ali samo u odgovarajuce dimenzije
b = b.reshape(3, 2)
print(b, b.shape)
```

Zadatak 2: Kreiranje višedimenzionalnih numpy arrayja

[ ]:
```
# array iz liste listi
array = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(array)

# array iz ugnijezdene komprehenzije listi
array = np.array([[x for x in range(i, i + 5)]
                  for i in [1, 7, 9]])
print(array)

# mozemo konstruirati arraye proizvoljnih dimenzija
# 3D array je koristan za prikaz slika
array = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
```

```
print(array, array.shape)

# indeksiranje zapocinje od 0
# slicing ide redom po dimenzijama

array = np.array([[0,1,2],[3,4,5]])
print('slice:   [0,1]:', array[0,1])
print('access: [0][1]:', array[0][1])
```

Zadatak 3: Ostali načini kreiranja numpy arraya

[ ]:
```
array = np.empty(4)
print("empty\n",array)

array = np.zeros((2,3))
print("zeroes\n",array)

array = np.identity(4)
print("identity\n",array)

array = np.eye(5,4)
print("eye(5,4)\n",array)

array = np.eye(4,5)
print("eye(4,5)\n",array)

array = np.random.random(5)
print("random(5)\n",array)

array = np.random.random([5,5])
print("random([5,5])\n",array)
```

Vježba Pogledati dokumentaciju: Numpy docs

Kako biste kreirali array koji sadrži samo nule, a istih je dimenzija kao neki zadani array?

Zadatak 4: Osnovne aritmetičke operacije

[ ]:
```
# Neke operacije su element-wise, sto znaci da se operacija izvrsava po␣
  ↪koordinatama

x = np.array([1,2,3,4])
y = np.array([2,4,6,8])
# Zbrajanje
print("x+y= ", x+y)
print("x+y= ",np.add(x,y))

# Oduzimanje
print("x-y= ",x-y)
```

```
print("x-y= ",np.subtract(x,y))

# Mnozenje
print("x*y= ",x*y)
print("x*y= ",np.multiply(x,y))

# Mnozenje skalarom
print("2*x= ",2*x)

# sqrt funkcija
print("sqrt(x)= ",np.sqrt(x))
```

[ ]:
```
# Broadcasting
# Sto bi sljedeca operacija trebala izracunati?
array = np.array([1,2,3]) + 1
print(array)

# Sto s ovim?

array = np.array([1,2,3]) * np.array([[1],[2],[3]])
print(array)
print(np.array([[1],[2],[3]]).shape)
# A s ovim?

try:
    array = np.array([1,2,3]) + np.array([1,2])
except ValueError as e:
    print('caught ValueError:', e)
```

Računanje se izvršava dimenziju po dimenziju. Ako su dimenzije jednake, računaj operaciju u Hadamard stilu. U suprotnom zahtjevamo da je jedna dimenzija jednaka 1, a druga k te array "podebljavamo" do dimenzije k.

Vidi Broadcasting za detaljnije informacije.

Zadatak 5: Osnovne aritmetičke operacije - množenje

[ ]:
```
#                vectors     matrices      tensors
# ------------------------------------------------
# @             | <a,b>     | AA          | ???
# numpy.inner   | <a,b>     | A^T A       | ???
# numpy.dot     | <a,b>     | AA          | ???
# numpy.vdot    | <a*,b>    | ???         | ???
# numpy.outer   | <a,b^T>   | ???         | ???


#### skalarni produkt (inner product)
x = np.array([1,2,3,4])
y = np.array([2,4,6,8])
```

```
print(np.inner(x, y))

skalarni_produkt = 0
for i in range(x.shape[0]):
    skalarni_produkt += x[i]*y[i]
print(skalarni_produkt)

# Numpy operacije:
print(np.dot(x,y))
print(x.dot(y))
print(np.inner(x,y))

#### vanjski produkt (outer product)
n = x.shape[0]
A = np.empty((n, n))
for i in range(n):
    for j in range(n):
        A[i,j] = x[i]*y[j]
print(A)

# Numpy operacija:
print(np.outer(x,y))

#### mnozenje matrice i vektora: Ax

A = 2 * np.eye(4,4)
b = np.zeros(4)
for i in range(4):
    for j in range(4):
        b[i] += A[i,j]*x[j]
print(b)

# Numpy operacija:
print(A.dot(x))

#### Mnozenje matrica
A = np.array([[1,2,3],[1,0,1],[1,0, 0]])
B = np.array([[2,2,2],[1,0,-1],[-1, 1, 0]])

C = np.zeros((A.shape[0], B.shape[1]))
for i in range(A.shape[0]):
    for j in range(B.shape[1]):
        for k in range(A.shape[1]):
            C[i,j] += A[i,k]*B[k,j]
print(C)

# Numpy operacije:
```

```
print(np.dot(A,B))
print(np.matmul(A,B))
```

Zadatak 6: Intervali

[ ]:
```
# osnovni intervali:
# arange([start],stop,[step])
# linspace(start,stop,n=50,endpoint=True)
#
# slozeniji ranges:
# logspace
# geomspace

array = np.arange(3,8)
print(array)

array = np.arange(3,11,2)
print(array)

array = np.arange(10,2,-2)
print(array)
```

Zadatak 7: Rješavanje Ax = b, gdje je A gornjetrokutasta Rjesenje je oblika

$$x_n = \frac{b_n}{A_{nn}} x_i = \frac{y_i - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}, i = 1, \ldots, n-1$$

[ ]:
```
def backsub(A, b):
    m, n = A.shape
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = b[i] - np.dot(x[i+1:], A[i, i+1:])
        if A[i,i] != 0: x[i] /= A[i, i]
        elif x[i] == 0: continue
        else: raise ValueError('backsub not solvable')
    return x

A = np.array([[1,2,3],
              [0,2,1],
              [0,0,4]])
b = np.array([12,10,8])
print(backsub(A,b), np.dot(A, backsub(A, b)))
```

Zadatak 8: Stack

[ ]:
```
# ponekad zelimo konstruirati novi array koristeci vec postojece arraye
# koristit cemo operacije stack i concatenate

# stack prihvaca niz arraya jednake dimenzije i "slaze" ih jedan po jedan
```

```
# po defaultu ce dodati dimenziju na vanjsku os

a = np.arange(4)
print(np.stack((a,a,a)))
print(np.stack((a,a,a), axis=1))

a = np.arange(3*4).reshape((3,4))
b = np.arange(3*4).reshape((3,4))

print(a)
print(b)

print('axis=0',np.stack((a,b)),sep='\n')
print('axis=1',np.stack((a,b),axis=1),sep='\n')
print('axis=2',np.stack((a,b),axis=2),sep='\n')
```

Zadatak 9: Concatenate

[ ]:
```
# konkatenacija je slicna stacku, ali ne dodaje novu dimenziju

a = np.arange(2*3).reshape((2,3))
print(a)
print(np.concatenate((a,a,a)))
print(np.concatenate((a,a,a),axis=1))
```

Zadatak 10: Razno

[ ]:
```
# postoje razne operacije za manipuliranje dimenzijama/osima arraya

a = np.arange(2*3*4*5*6)
b = a.reshape(2,3,4,5,6)
c = np.moveaxis(b,0,2)
d = np.transpose(b)

print('a', a.shape)
print('b', b.shape)
print('c', c.shape)
print('d', d.shape)

# Transpose
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(a.T)
```

Zadatak 11: Rad s podacima

1. učitavanje podataka
   - neke od standardnih funkcija koje ćete koristiti iz paketa numpy i pandas
   - podaci su najčešće u formatima: csv, txt, slike
2. spremanje podataka

- neke od standardnih funkcija za spremanje podataka

[ ]:
```
# Primjer: Numpy
# Učitamo neke podatke koji su lokalno spremljeni u .csv
podaci = np.loadtxt('./Podaci/data.csv', delimiter =',')

# Pogledamo koliko podataka ima
podaci.shape
```

[ ]:
```
# Primjer: Pandas - alat za jednostavno manipuliranje podacima
# Učitamo neke podatke koji su lokalno spremljeni u .csv
podaci = pd.read_csv('./Podaci/data.csv', names=['prvi', 'drugi'])

# Pregled podataka
print(podaci.head())

# Pogledamo koliko podataka ima
print(podaci.shape)
```

[ ]:
```
# Primjer:
X = np.random.random([10,2])
np.savetxt('Podaci-1', X)

X_df = pd.DataFrame(X)
X_df.to_csv('Podaci-2', header=['prvi', 'drugi'])
```

1.1.2 Upoznavanje s matplotlib, scikit-learn

Zadatak 12: Vizualizacija podataka

- najčešće korišteno: matplotlib.pyplot
- podaci predstavljaju točke u ravnini: scatter plot
- crtanje grafa ovisnosti jedne varijable o drugoj: plot
- crtanje histograma, matrice korelacije
- crtanje vektora

[ ]:
```
# Učitamo neke podatke koji su lokalno spremljeni u .csv
podaci = np.loadtxt('./Podaci/data.csv', delimiter =',')

# Podaci su neke točke oblika (x,y)
plt.scatter(podaci[:,0], podaci[:,1])
plt.show()

# Crtanje grafa funkcije
x = np.arange(0,10,0.5)
y = x**2 + 2*x + 5
plt.plot(x,y, color='mediumaquamarine')
plt.scatter([4],[29], color='midnightblue')
```

```
plt.show()

# Drugi način
plt.scatter(x,y)
plt.show()

# Crtanje pravca
x = np.array([2,5])
y = np.array([2,5])
plt.plot(x,y)
plt.show()
```

[ ]:
```
# Unutar paketa scikit-learn postoje neki standardni datasetovi koje možete␣
↪koristiti
# vidi https://scikit-learn.org/stable/datasets.html
# Svi se učitavaju na sljedeći način:
dataset = sklearn.datasets.load_breast_cancer()
df = pd.DataFrame(data=dataset['data'], columns=dataset['feature_names'])
df
```

[ ]:
```
df[['mean radius', 'mean area', 'worst symmetry']].to_numpy()
```