Wizualizacja danych
semestr letni 2024

Dr Anna Muranova
UWM w Olsztynie

Wykład 12

Instalowanie pakietów

W przypadku metod obliczeniowych jedną z najpopularniejszych bibliotek jest właśnie **NumPy** (Numeric Python). Jak nazwa wskazuje, skupia się ona głównie na wsparciu pracy z danymi numerycznymi i umożliwia efektywne i wydajne dokonywanie operacji na macierzach, tablicach czy wektorach, które zawierają elementy tego samego typu.

Python packages –> Wyszukać pakiet **numpy** i zainstalować (lepiej w wersie `1.26.3`)

```
import numpy as np
```

Tablica

Tablica (ang. array) w `NumPy` to struktura o jednym wymiarze lub większej liczbie wymiarów pozwalająca na działania ze zbiorami danych numerycznych, zaczynając od kilkuelementowych po ogromne zbiory przechowywane np. w chmurze. Pamiętajmy, że nie tylko wymiary charakteryzują tablicę – istnieje też kilka innych, równie ważnych cech.

- Tablice są stałej wielkości – nie możemy zmienić rozmiaru po jej utworzeniu.
- Przechowują elementy tego samego typu, np. liczby całkowite lub zmiennoprzecinkowe.
- Wykorzystywane „pod spodem" algorytmy pozwalają na bardzo szybkie operacje i efektywne wykorzystanie pamięci.

```
arr_1d = np.array([1, 2, 3])
print(arr_1d)

arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr_2d)
```

Podstawowe operacje

```
import numpy as np

# tworzenie jednowymiarych tablic a i b
a = np.random.rand(3)
b = np.ones(3)
print(a)
print(b)

# dodawanie tablicy b do tablicy a
c = a + b
print(c)
# dodawanie 5 do tablicy a
print(5+a)
# mnożenie tablicy a przez stałą 2
d = 2 * a
print(d)
# tablica z 10 równo rozłożonymi wartościami w zakresie od 0 do 2
x = np.linspace(0, 2, 10)
print(x)
```

Macierze

```
import numpy as np

# dwuwymiarowa tablica z losowymi
#wartościami z rozkładu normalnego o wymiarach 3x3
M1 = np.random.randn(3, 3)
print(M1)

a = np.array([3, 7, 3, 3, 2, 9,  1, 5, 4])
print(np.shape(a))
M2 = a.reshape(3,3)
print(M2)
```

Operacje algebraiczne

```
print(M2.transpose())
print(np.linalg.det(M2))
print(M1+M2)
print(M1*M2)
print(M1@M2)
print(M2**2)
print(np.linalg.matrix_power(M2,2))
print(np.linalg.matrix_power(M2,-1))
print(np.linalg.inv(M2))
```

numpy: dane

Według następnej tabeli oblicz następny czynności (programistyczne, tabela wprowadzić ręcznie, każdą kolumna jako wektor, nazwa kolumny – jako nazwa wektora):

Nazwa rzeki	Kontynent	Długość w km	Powierzchnia dorzecza w tys. km^2	Liczba państw przez które przepływa
Amazonka	Ameryka Południowa	7040	7200	3
Nil	Afryka	6695	2870	7
Jangcy	Eurazja	6300	1807	1
Missisipi-Missouri	Ameryka Północna	6020	3229	1
Huang He	Eurazja	5464	752	1
Ob Irtysz	Eurazja	5410	2972	3
Kongo	Afryka	4700	3690	3
Mekong	Eurazja	4500	810	6
Amur	Eurazja	4440	1855	3
Lena	Eurazja	4400	2490	1
Parana	Ameryka Południowa	4380	3100	3
Mackanzie	Ameryka Północna	4240	1760	1
Niger	Afryka	4160	2117	4
Jenisej	Eurazja	4102	2580	2
Wołga	Eurazja	3530	1360	2

numpy: dane 1

```
import numpy as np
#import matplotlib.pyplot as plt

river = np.array(['Amazonka','Nil','Jangcy','Missisipi-Missouri',
                  'Huang He','Ob Irtysz','Kongo','Mekong',
                  'Amur', 'Lena', 'Parana','Mackanzie','Niger',
                  'Jenisej','Wolga'])
kontynent = np.array(['APolud','A','E','APolud','E','E','A','E',
                      'E','E','APolud','APolun','A','E','E']
lenght = np.array([7040, 6695,6300, 6020,5464,5410,
                   4700, 4500,4440,4400,4380,4240,
                   4160,4102,3530])
area=np.array([7200,2870,1807,3229,752,2972,3690,
               810,1855,2490,3100,1760,2117,2580,1360])
countries = np.array([3,7,1,1,1,3,3,6,3,1,3,1,4,2,2])
```

numpy: dane 2

```
#max dlugosc rzeki:
print('max dlugosc: ', np.max(lenght))
#srednia dlugosc rzeki
print('srednia dlugosc: ', np.mean(lenght))
#rzeka o max dlugosci:
print('rzeka o max dlugosc: ', river[lenght==np.max(lenght)])
#Ile rzek są w tabelce?
print('Ilosc rzek:',len(river))
```

numpy: dane 3

```
#Rzeki przez 3 Panstwa:
print("3 Panstwa: ", river[countries==3])

#Ile rzek maja długość mniej niż $5000$ km?

print('długość mniej niż 5000:',len(lenght[lenght<5000]))

#Rzeki z M
print("Rzeki na M:", river[(river>='M')*(river<'N')])

#Posortuj nazwy rzek według powierzchni rosnanco.
print("Posortowane wedlug arei:", river[np.argsort(area)][::-1])

#Wypisz nazwy rzek z powierzchnia większą niż $2000$ tys. km$^2$,
#który są w Ameryce
print("Pow > 2000 w Am:", river[(area>2000) & ((kontynent=='APolud') |
                                                (kontynent=='APolun'))])
```

Biblioteka `matplotlib`

`https://matplotlib.org/`

Zazwyczaj importuje się główny moduł tej biblioteki, czyli 'pyplot'. Zawiera on niezbędne funkcje do generowania wykresów.

```
import matplotlib.pyplot as plt
```

Szczególnie wydajne jest połączenie możliwości tej biblioteki z innymi bibliotekami naukowymi, takimi jak NumPy, Pandas czy SciPy.

`https://www.w3schools.com/python/matplotlib_intro.asp`

Wykres liniowy

```
import numpy as np
import matplotlib.pyplot as plt

#print(plt.style.available)
plt.style.use('_mpl-gallery')

# Przygotowanie danych
x = np.linspace(0, 10, 100)
y = x*(np.cos(x))**2

plt.plot(x, y)  # Rysowanie wykresu liniowego
plt.show()  # Wyświetlenie wykresu
```

Dwa wykresy

```
import numpy as np
import matplotlib.pyplot as plt

plt.rc('text', usetex=True)

# Przygotowanie danych
x = np.linspace(0, 10, 100)
y1 = x*(np.cos(x))**2
y2 = x*(np.sin(x))**2

plt.plot(x, y1, color = 'b', linestyle=':', label='$x \cos^2 x$')
plt.plot(x, y2, color='r', linewidth = 1.5, label='$x \sin^2 x$')
plt.legend()
plt.show()
```

Kilka wykresów w jednym okienku

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-1,2,100)
y = np.exp(x)
y1 = 0.2*np.exp(2*x)
fig, ax  = plt.subplots(2)
ax[0].plot (x,y,'blue', linestyle = "-", label = "e^x " )
ax[0].set_ylim(0,10)
ax[1].plot (x,y1, 'darkgreen', linestyle = "--" , label = "1/5 e^(2x)"
fig.legend (title = 'Wykres', loc=9)
ax[1].set_ylim(0,10)
plt.show()
```

Wykres trzech wymiarowe 1

```
import matplotlib.pyplot as plt
import numpy as np

# set up a figure twice as wide as it is tall
fig = plt.figure(figsize=plt.figaspect(0.5))

# =============
# First subplot
# =============
# set up the axes for the first plot
ax = fig.add_subplot(1, 2, 1, projection='3d')

# plot a 3D surface like in the example mplot3d/surface3d_demo
X = np.linspace(0, 2, 100)
Y = np.linspace(0, 2*np.pi, 100)
X, Y = np.meshgrid(X, Y)
Z1 = X*np.sin(Y)
Z2 = X+np.sin(Y)
```

Wykres trzech wymiarowe 2

```
surf = ax.plot_surface(X, Y, Z1, rstride=1, cstride=1, cmap='coolwarm',
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=10)

# ==============
# Second subplot
# ==============
# set up the axes for the second plot
ax = fig.add_subplot(1, 2, 2, projection='3d')

# plot a 3D wireframe like in the example mplot3d/wire3d_demo
surf = ax.plot_surface(X, Y, Z2, rstride=1, cstride=1, cmap="coolwarm",
                       linewidth=0, antialiased=False)
fig.colorbar(surf, shrink=0.5, aspect=10)
```

Wykresy słupkowe i kołowe

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
plt.pie(y, labels=x)
plt.show()
```

Wykresy słupkowe i kołowe

```
import numpy as np
#import matplotlib.pyplot as plt

river = np.array(['Amazonka','Nil','Jangcy','Missisipi-Missouri',
                  'Huang He','Ob Irtysz','Kongo','Mekong',
                  'Amur', 'Lena', 'Parana','Mackanzie','Niger',
                  'Jenisej','Wolga'])
area=np.array([7200,2870,1807,3229,752,2972,3690,
               810,1855,2490,3100,1760,2117,2580,1360])
countries = np.array([3,7,1,1,1,3,3,6,3,1,3,1,4,2,2])

plt.bar(river, countries, color='green')
plt.xticks(rotation=60)
plt.show()
plt.pie(area, labels=river)
plt.show()
```

Biblioteka SciPy

```
from scipy import constants

print(constants.pi)
```

Lista wszystkich stałych:

```
from scipy import constants

print(dir(constants))
```

https://docs.scipy.org/doc/scipy/reference/constants.html

Jeszcze:

`https://www.w3schools.com/python/scipy/scipy_constants.php`

- **Przedrostki metryczne (SI)**
- **Przedrostki binarne (SI)**
- **Waga (SI)**
- **Kąty**: zwraca określoną jednostkę w radianach
- **Czas**: zwraca określoną jednostkę w sekundach
- **Długość**: zwraca określoną jednostkę w metrach
- **Ciśnienie**: zwraca określoną jednostkę w paskalach
- **Obszar**: zwraca określoną jednostkę w metrach kwadratowych
- **Objętość**: zwraca określoną jednostkę w metrach kubicznych
- **Prędkość**: zwraca określoną jednostkę w metrach na sekundą
- **Temperatura**: zwraca określoną jednostkę w Kelvinach
- **Energia**: zwraca określoną jednostkę w Dżulach
- **Moc**: zwraca określoną jednostkę w wattach
- **Siła**: zwraca określoną jednostkę w Newtonach

Optymizacja: pierwiastki równania

Znaleźć pierwiastek równania $x^2 + \sin(x) = 1$

```
from scipy.optimize import root
from math import sin


def eqn(x):
  return x**2 + sin(x)-1

myroot = root(eqn, 0)

print(myroot)
print(myroot.x)
```

Optymizacja: minimum, maximum

Aby zminimalizować funkcję, możemy użyć funkcji `scipy.optimize.minimize()`.

```
from scipy.optimize import minimize
from math import sin

def eqn(x):
  return  x**2 + sin(x)-1

mymin = minimize(eqn, 0, method='BFGS')

print(mymin.x)
print(mymin)
```