Laboratorium 3

Analiza Danych Pomiarowych
Ćwiczenia Laboratoryjne
Inżynieria Biomedyczna 2019/2020

Zagadnienia

Zagadnienia do samodzielnego opracowania:

- Importowanie bibliotek (import/from/as).
- Podstawy biblioteki NumPy. Tablica. Podstawowe operacje na tablicach. Manipulowanie tablicami.
- Podstawowe funkcje numeryczne, tablica jako macierz i wektor.
- Wektoryzacja obliczeń.
- Budowanie relatywnych i absolutnych ścieżek do plików (biblioteka os).
- Wczytywanie i zapisywanie plików tekstowych oraz w formacie .csv (biblioteka csv).

Zadania

Zadanie 1.
Zdefiniuj tablicę jednowymiarową zawierającą 101 elementów od 0 do 100 (włącznie). Następnie zdefiniuj drugą tablicę jednowymiarową zawierającą elementy od 100 do 0 (włącznie). Wyobraź sobie, że tablice reprezentują pomiar temperatury z dwóch czujników. Wykonaj następujące operacje: dodaj do siebie zadane temperatury, przemnóż je przez siebie, wybierz z tablicy pierwszej 10 pierwszych elementów i odejmij od nich 10 ostatnich elementów drugiej tablicy. Wybierz z tablicy pierwszej tylko elementy, które są podzielne przez 3. W przypadku wszystkich powyższych operacji nie używaj żadnych pętli oraz instrukcji warunkowych. Korzystaj jedynie z funkcji dostępnych w bibliotece NumPy.
Przykład i przewidywany rezultat:

```
>>> [   0    1    2    3    4    5    6    7    8    9   10 ...
>>> 90   91   92   93   94   95   96   97   98   99  100]
>>> [100   99   98   97   96   95   94   93   92   91   90   ...
>>> 10    9    8    7    6    5    4    3    2    1    0]
>>> [100  100  100  100  100  100  100  100 ...  100  100  100  100  100  100  100]
>>> [    0    99   196   291   384   475   564   651   736   819 ...
>>> 819   736   651   564   475   384   291  196    99     0]
>>> [-9 -7 -5 -3 -1  1  3  5  7  9]
>>> [ 0  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45
48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99]
```

Zadanie 2.
Zadeklaruj listę pomiarów zawierającą elementy od 0 do 1000000. Następnie korzystając z pętli for stwórz nową listę składającą się z każdego pomiaru podniesionego do kwadratu. Korzystając z biblioteki **time** zmierz czas tego zadania.
Następnie zadeklaruj tablicę pomiarów zawierającą elementy od 0 do 1000000. Podnieś wszystkie pomiary do kwadratu (bez korzystania z jakiejkolwiek pętli) i przypisz do nowej tablicy. Analogicznie jak w poprzednim przypadku zmierz czas tego zadania.
Skąd bierze się różnica?
Przewidywany rezultat:

```
>>> Elapsed time:  your_time_here  seconds. # list
>>> Elapsed time:  your_time_here  seconds. # array
```

Zadanie 3.
Wygeneruj losową macierz (tablicę) pomiarów o wymiarze 7x7 i rozkładzie normalnym o wartości średniej równej 50 i odchyleniu standardowym równym 20. Zrzutuj ją na typ całkowity. Oblicz jej wartości własne oraz odpowiadające im wektory własne (załóż, że nie wystąpią przypadki zdegenerowane). Wyznacz macierz odwrotną. Przemnóż wygenerowaną macierz przez wektor [1, 2, 3, 4, 5, 6, 7]. Dokonaj rozkładu pierwotnej macierzy na wartości osobliwe (SVD).
Przewidywany, przykładowy rezultat:

```
Wygenerowana macierz:
[[ 9 53 61 68 62 40 36]
[47 52 49 44 87 75 18]
[37 25 15 61 25 46 38]
[49 32 73 31 18 59 44]
[43 48 31 63 30 86 57]
[77 35 42 57 40 51 62]
[64 24 61 55 55 58 97]]
Wyznacznik:  xxx
Slad:  xxx
Wartosci i wektory wlasne:  xxx
Macierz odwrotna:  xxx
Przemnozenie przez wektor:  xxx
Glowna przekatna do kwadratu:  xxx
SVD:  xxx
```

Zadanie 4.
Wczytaj plik **iris.csv**. Dane zapisz do słownika, który do każdej cechy danych wejściowych przyporządkuje listę wszystkich wartości. Policz i wypisz średnią oraz odchylenie standardowe dla każdego atrybutu. Wyjście sformatuj do czterech miejsc po przecinku. Wyznacz i wypisz liczebność każdej klasy.
Przewidywany, przykładowy rezultat:

```
>>> Sepal Length Mean:    5.8433
>>> Sepal Length Std:    0.8253
>>> Sepal Width Mean:    3.0573
>>> Sepal Width Std:    0.4344
>>> Petal Length Mean:    3.7580
>>> Petal Length Std:    1.7594
>>> Petal Width Mean:    1.1993
>>> Petal Width Std:    0.7597
>>> Number of Setosa:  50
>>> Number of Versicolor:  50
>>> Number of Virginica:  50
```

Zadanie 5.
Napisz funkcję, która jako argument przyjmuje ścieżkę do folderu. Funkcja powinna zwrócić listę zawierającą **absolutne** ścieżki do każdego pliku zawierającego rozszerzenie .png. Pozostałe pliki powinny zostać zignorowane.
Przewidywany, przykładowy rezultat:

```
cifar_path = xxx
files = get_absolute_paths(cifar_path)
print(files)
```

```
# Here filter the results to contain only airplanes
filtered_files = # TO DO
print(filtered_files)
```

Zadanie 6.
Korzystając z rozwiązania Zadania 4. wczytaj dane z pliku **iris.csv**. Następnie do każdego przypadku atrybutu "Sepal Length" dodaj 1, od każdego atrybutu "Sepal Width" odejmij 1, "Petal Length" podnieś do kwadratu, a "Petal Width" spierwiastkuj. Wyniki zapisz do pliku **iris_zad_6.csv** o takim samym formacie jak oryginalny plik **iris.csv**. Zapisz plik do tego samego folderu co plik **iris.csv**.
Przewidywany, przykładowy rezultat:

```
# Nothing to print
```

Zadanie 7.
Wygeneruj losową tablicę zawierającą 1000 elementów o typie float32 (korzystając z dowolnego rozkładu). Następnie pomnóż tablicę przez 1000 i wynik zapisz do pliku **zad_7.txt**. Następnie wczytaj dane z pliku do nowej tablicy i podziel ją przez 1000. Oblicz błąd średniokwadratowy pomiędzy oryginalnie wygenerowaną tablicą, a tablicą wynikową.
Przewidywany, przykładowy rezultat:

```
>>> MSE: 9.289791e-17
```

Zadanie 8.
Wygeneruj tablicę trójwymiarową zawierającą 50 losowych macierzy reprezentujących pomiary o wymiarze 100x100. Następnie wygeneruj losowe 50 wektorów o wymiarze 100x1. Przemnóż każdą z losowo wygenerowanych macierzy przez odpowiadający jej (indeksem) losowo wygenerowany wektor. Nie używaj jakichkolwiek pętli, instrukcji warunkowych, bądź instrukcji sterujących. Następnie zrób to samo ale użyj pętli for i licz wyznacznik oraz przemnażaj macierze przez wektory jeden po drugim. Korzystając z biblioteki **time** porównaj czas potrzebny na przeprowadzenie obliczeń.
Przewidywany, przykładowy rezultat:

```
# Nothing to print
```

Zadanie 9.
Zdefiniuj dowolny wektor o długości 10 elementów. Każdy element o parzystym indeksie podnieś do kwadratu, a każdy element o nieparzystym indeksie wyzeruj.
Przykładowy rezultat:

```
>>> [ 0.8208 -0.2486 -1.0394  0.0131 -0.2143  0.7
0.9556  1.4173  0.0986 0.7532]
>>> [ 0.6737  0. 1.0803  0.  0.0459  0.  0.9131  0. 0.0097 0. ]
```

Zadanie 10.
Wygeneruj losowy wektor 1000 liczb całkowitych. Napisz funkcję, która zwróci n (argument) największych wartości.
Przykładowy rezultat:

```
print(n_largest(array, 5))
>>> [ 3.3238  3.3101  2.9439  2.8311  2.7777]
```

Zadanie 11.
Zdefiniuj dowolną tablicę. Spraw by próba zmiany wartości zadanej tablicy skutkowała zgłoszeniem błędu.
Przykładowy rezultat:

```
try:
array[5] = 7
```

```
except Exception as e:
print(e)
>>> assignment destination is read-only
```

Zadanie 12.
Zdefiniuj ustrukturyzowaną tablicę zawierającą w każdym polu pozycję (x, y, z), a następnie wypełnij ją losowymi wartościami.
Przykładowy rezultat:

```
>>> [( 0.0454, -0.2329,  0.8268) (-0.4406, -1.4612, -0.389 )
(-1.1845,  0.1103,  0.412 ) (-1.7319, -0.5615, -0.4756)
(-1.091 , -0.9772,  0.3296) ( 0.2208,  0.3394, -0.1366)
( 0.8412,  0.9552,  0.4302) (-0.2491, -1.246 , -1.2971)
( 0.2971, -0.0031, -0.0996) (-0.5548,  1.4346, -0.3918)]
```