Lab report 2 (Variant 2)

Student :
Herashchenko Serhii

Teacher :
Zhuravlova Zinaida Yuriyivna

16 жовтня 2024 р.

Зміст

1 Task 1

1.1 Condition

Дано 10 випадкових точок одиничного кола в $\mathbf{R}^2$.

а) Описати функцію point2angle(x), що по заданій точці x повертає відповідний їй кут від 0 до 2π.Вивести масив кутів за допомогою цієї функції.

б) Знайти точку, якій відповідає найменший кут.

в) Відсортувати точки в порядку зростання кутів, які їм відповідають.

г) Профільтрувати отриманий масив, залишивши тільки кути від 0 до $\frac{\pi}{2}$.

1.2 Solution code

```
import numpy as np

def point2angle(points):
    points = np.array(points)

    x = points[:, 0]
    y = points[:, 1]

    angles = np.arctan2(y, x)
    angles = np.where(angles < 0, angles + 2 * np.pi, angles)

    return angles

random_angles = np.random.uniform(0, 2 * np.pi, 10)

points = np.column_stack((np.cos(random_angles), np.sin(random_angles)))

angles = point2angle(points)

sorted_indices = np.argsort(angles)
sorted_points = points[sorted_indices]
sorted_angles = angles[sorted_indices]

#-----------------------------------------
print(np.round(sorted_points[0], 3))
#-----------------------------------------

filtered_indices = np.where((sorted_angles >= 0) & (sorted_angles <= np.pi / 2))[0]

filtered_points = sorted_points[filtered_indices]
filtered_angles = sorted_angles[filtered_indices]
```

```
#----------------------------------------
print("Filtered points:")
print(np.round(filtered_points, 3))
print("Filtered angles:")
print(np.round(filtered_angles, 3))
#----------------------------------------
```

1.3 Output

[0.924 0.382]
Filtered points:
[[0.924 0.382]
[0.915 0.404]
[0.656 0.754]
[0.121 0.993]]
Filtered angles:
[0.392 0.415 0.855 1.45]

2 Task 2

2.1 Condition

Сконструювати блочну матрицю (використовуючи функції для заповнення стандартних матриць) і застосувати функції обробки даних і поелементні операції для знаходження заданих величин.

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & 3 & -2 & 1 & -1 \\ 1 & -1 & 3 & -2 & 1 & -1 \\ 1 & -1 & 3 & -2 & 1 & -1 \\ 1 & -1 & 3 & -2 & 1 & -1 \\ 1 & -1 & 3 & -2 & 1 & -1 \\ 1 & -1 & 3 & -2 & 1 & -1 \end{pmatrix} \qquad s = \sum_{i=0}^{5} \sum_{j=0}^{5} a_{ij}$$

2.2 Solution code

```
import numpy as np

matrix = np.matrix([[1, -1, 3, -2, 1, -1]] * 6)

print(matrix)

s = np.sum(np.abs(matrix))

print(s)
```

2.3 Output

```
[[ 1 -1 3 -2 1 -1]
 [ 1 -1 3 -2 1 -1]
 [ 1 -1 3 -2 1 -1]
 [ 1 -1 3 -2 1 -1]
 [ 1 -1 3 -2 1 -1]
 [ 1 -1 3 -2 1 -1]]
54
```

3 Task 3

3.1 Condition

Написати функцію, що перевіряє по критерію Сильвестра, чи є матриця додатньо визначеною. Перевірити роботу функції на наступних матрицях:

$$\mathbf{A} = \begin{pmatrix} 22 & -9 & 7 \\ -9 & 22 & -19 \\ 7 & -19 & 17 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 9 & 9 & 2 & 5 & 5 \\ -4 & 1 & 1 & 2 & 8 \\ 8 & 7 & 0 & 4 & 2 \\ 4 & -1 & 9 & 7 & -5 \\ 2 & 5 & 2 & -4 & 8 \end{pmatrix}$$

3.2 Solution code

```python
import numpy as np

def is_positive_definite(matrix):
    n = matrix.shape[0]

    for i in range(1, n + 1):
        minor = matrix[:i, :i]
        if np.linalg.det(minor) <= 0:
            return False

    return True

A = np.matrix([[22, -9, 7],
               [-9, 22, -19],
               [7, -19, 17]])

B = np.matrix([[9, 9, 2, 5, 5],
               [-4, 1, 1, 2, 8],
               [8, 7, 0, 4, 2],
               [4, -1, 9, 7, -5],
               [2, 5, 2, -4, 8]])
```

```
print(is_positive_definite(A))
print(is_positive_definite(B))
```

3.3 Output

True
False

4 Task 4

4.1 Condition

При розв'язання багатьох задач транспортної логістики виникає необхідність в побудові матриці відстаней між об'єктами. Спробуємо розв'язати цю підзадачу у NumPy. Згенерувати 10 випадкових точок на площині. Нехай, для визначеності, ці точки потрапляють у квадрат, що обмежений прямими . Необхідно побудувати матрицю попарних відстаней між ними, якщо в якості відстані обрана менхентенська метрика.

$$\rho(A, B) = |x_A - x_B| + |y_A - y_B|$$

На основі цієї матриці знайти 2 найбільш віддалені (в указаній метриці) точки.

4.2 Solution code

```
import numpy as np

points = np.random.uniform(0, 20, (10, 2))

def manhattan_distance_matrix(points):
    num_points = points.shape[0]
    distance_matrix = np.zeros((num_points, num_points))

    for i in range(num_points):
        for j in range(num_points):
            distance_matrix[i, j] = np.abs(points[i, 0] - points[j, 0]) + np.abs(point

    return distance_matrix

distance_matrix = manhattan_distance_matrix(points)

max_distance_index = np.unravel_index(np.argmax(distance_matrix, axis=None), distance_
point1 = points[max_distance_index[0]]
point2 = points[max_distance_index[1]]

print("Generated points:")
for point in points:
    print(f"({point[0]:.2f}, {point[1]:.2f})")
```

```
print("\nDistances:")
print(np.round(distance_matrix, 2))

print(f"\nMost remote points: ({point1[0]:.2f}, {point1[1]:.2f}) and ({point2[0]:.2f}
```

4.3 Output

Generated points:
(12.66, 12.43)
(6.30, 18.96)
(18.68, 13.60)
(5.26, 12.03)
(16.79, 6.95)
(9.20, 7.81)
(13.17, 11.25)
(10.38, 10.80)
(9.22, 2.96)
(11.49, 5.78)

Distances:
[[0. 12.89 7.19 7.8 9.62 8.07 1.69 3.9 12.91 7.83]
[12.89 0. 17.74 7.97 22.51 14.05 14.58 12.25 18.92 18.38]
[7.19 17.74 0. 14.99 8.54 15.26 7.87 11.09 20.1 15.02]
[7.8 7.97 14.99 0. 16.62 8.16 8.69 6.35 13.03 12.48]
[9.62 22.51 8.54 16.62 0. 8.46 7.93 10.26 11.56 6.48]
[8.07 14.05 15.26 8.16 8.46 0. 7.4 4.17 4.87 4.32]
[1.69 14.58 7.87 8.69 7.93 7.4 0. 3.23 12.23 7.15]
[3.9 12.25 11.09 6.35 10.26 4.17 3.23 0. 9.01 6.13]
[12.91 18.92 20.1 13.03 11.56 4.87 12.23 9.01 0. 5.09]
[7.83 18.38 15.02 12.48 6.48 4.32 7.15 6.13 5.09 0.]]

Most remote points: (6.30, 18.96) and (16.79, 6.95)

5 Task 5

5.1 Condition

Написати функцію, що замінює максимальний елемент вектору середнім значенням усіх його елементів. Функція зчитує вектор з файлу input.csv і записує результуючий вектор в файл output.csv.

5.2 Solution code

```
import numpy as np

file = open("C:\\ALL\\OTHER\\GitHub\\HOMEWORK\\Python\\L2_V2_Herashchenko\\5_task\\inp
```

```
data = file.read().split(" ")
vector = np.array(data, float)

average = np.sum(vector)/vector.size

max_index = np.argmax(vector)

vector[max_index] = average

file.close()

file = open("C:\\ALL\\OTHER\\GitHub\\HOMEWORK\\Python\\L2_V2_Herashchenko\\5_task\\out
file.write(str(np.round(vector,3)))
file.close
```

5.3 Input

1 2 7 4 8 6 3

5.4 Output

[1. 2. 7. 4. 4.429 6. 3.]