МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчѐт по практическому занятию №3.3

«Исследование методов работы с матрицами и векторами с помощью библиотеки NumPy»

по дисциплине «Теории распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова Виктория Николаевна
« »_____20__г.
Подпись студента _____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore). Выполните клонирование созданного репозитория на рабочий компьютер.Организуйте свой репозиторий в соответствие с моделью ветвления git-flow. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

2. Проработать примеры лабораторной работы.

```
In [1]: import numpy as np

In [2]: v_hor_np=np.array([1,2])
        print(v_hor_np)

        [1 2]

In [3]: v_hor_zeros_v1=np.zeros((1,5))
        print(v_hor_zeros_v1)

        [[0. 0. 0. 0. 0.]]

In [4]: v_hor_one_v1=np.ones((5,))
        print(v_hor_one_v1)
        v_hor_one_v2=np.ones((1,5))
        print(v_hor_one_v2)

        [1. 1. 1. 1. 1.]
        [[1. 1. 1. 1. 1.]]

In [5]: v_vert_np=np.array([[1],[2]])
        print(v_vert_np)

        [[1]
         [2]]

In [6]: v_vert_zeros=np.zeros((5,1))
        print(v_vert_zeros)

        [[0.]
         [0.]
         [0.]
         [0.]
         [0.]]

In [7]: v_vert_ones = np.ones((5, 1))
        print(v_vert_ones)

        [[1.]
         [1.]
         [1.]
         [1.]
         [1.]]
```

In [8]:
```
m_sqr_arr=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [9]:
```
m_sqr=[[1,2,3],[4,5,6],[7,8,9]]
m_sqr_arr = np.array(m_sqr)
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [10]:
```
m_sqr_mx=np.matrix([[1,2,3],[4,5,6],[7,8,9]])
print(m_sqr_mx)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

3. Создать ноутбук, в котором будут приведены собственные примеры на языке Python для каждого из представленных свойств матричных вычислений.

In [1]:
```
import numpy as np
```

Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел:

In [2]:
```
A=np.matrix('45 15 6; 7 4 6; 3 34 16')
p=3
q=7
L=(p+q)*A
R=p*A+q*A
print(f"L= {L}")
print(f"R= {R}")
```

```
L= [[450 150  60]
 [ 70  40  60]
 [ 30 340 160]]
R= [[450 150  60]
 [ 70  40  60]
 [ 30 340 160]]
```

Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число:

In [3]:
```
A = np.matrix('9 18 2; 4 13 4')
B = np.matrix('5 6 26; 4 47 8')
k = 3
L = k * (A + B)
R = k * A + k * B
print(f"L= {L}")
print(f"R= {R}")
```

```
L= [[ 42  72  84]
 [ 24 180  36]]
R= [[ 42  72  84]
 [ 24 180  36]]
```

Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться:

In [4]:
```
A = np.matrix('11 23 4 ; 21 3 4')
B = np.matrix('25 32 6; 7 11 28')
C = np.matrix('1 22 7; 19 23 3')
L = A + (B + C)
R = (A + B) + C
```

При перестановке строк матрицы знак ее определителя меняется на противоположный:

In [5]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(f"A= {A}")
B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
print(f"B= {B}")
print(round(np.linalg.det(A),3))
print(round(np.linalg.det(B),3))
```

```
A= [[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
B= [[10  4 -1]
 [-4 -1  2]
 [ 8  3  1]]
-14.0
14.0
```

Создать ноутбук, в котором будут приведены собственные примеры решения систем линейных уравнений матричным методом и методом Крамера.

Матричный метод

In [2]:
```
import numpy as np
A = np.matrix('9 1 2; 5 2 1; 7 7 4')
B = np.matrix('3; 9; 2')
if round(np.linalg.det(A), 3) != 0:
    A_inv = np.linalg.inv(A)
    R = A_inv.dot(B)
    i = 1
    for elem in R:
        print(f"x{i} = {elem}")
        i += 1
else:
    print("Определитель равен 0, систему решить нельзя")
```

```
x1 = [[2.28947368]]
x2 = [[4.23684211]]
x3 = [[-10.92105263]]
```

Метод крамера

In [5]:
```
import numpy as np
A = np.matrix('7 1 0; 5 4 2; 7 5 3')
B = np.matrix('5; 1; 4')
if round(np.linalg.det(A), 3) != 0:
    C = A.copy()
    C[:, 0] = B[:, 0]
    x1 = round(np.linalg.det(C), 3) / round(np.linalg.det(A), 3)
    F = A.copy()
    F[:, 1] = B[:, 0]
    x2 = round(np.linalg.det(F), 3) / round(np.linalg.det(A), 3)
    J = A.copy()
    J[:, 2] = B[:, 0]
    x3 = round(np.linalg.det(J), 3) / round(np.linalg.det(A), 3)

    print("x1:", x1, "x2:", x2, "x3:", x3)
```

```
x1: 1.1538461538461537 x2: -3.076923076923077 x3: 3.769230769230769
```

Вопросы для защиты работы

Вопросы для защиты работы

1. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python.

вектор строки

In [ ]:
```
v_hor_zeros_v1 = np.zeros((5,))
print(v_hor_zeros_v1 )
```

```
[0. 0. 0. 0. 0.]
```

In [ ]:
```
 v_hor_zeros_v2 = np.zeros((1, 5))
print(v_hor_zeros_v2)
```

```
[[0. 0. 0. 0. 0.]]
```

In [ ]:
```
v_hor_one_v1 = np.ones((5,))
print(v_hor_one_v1)

v_hor_one_v2 = np.ones((1, 5))
print(v_hor_one_v2)
```

```
[1. 1. 1. 1. 1.]
[[1. 1. 1. 1. 1.]]
```

веткор столбец

In [ ]:
```
v_vert_np = np.array([[1], [2]])
print(v_vert_np)
```

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

In [ ]:
```
v_vert_zeros = np.zeros((5,1))
print(v_vert_zeros)
```

```
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

In [ ]:
```
v_vert_ones = np.ones((5, 1))
print(v_vert_ones)
```

```
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

квадратная матрица

```
 m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
m_sqr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
m_sqr_arr = np.array(m_sqr)
print(m_sqr_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
print(m_sqr_mx)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

диагональные матрицы

```
m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
m_diag_np = np.matrix(m_diag)
print(m_diag_np)
```

```
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

```
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
diag = np.diag(m_sqr_mx)
print(diag)
print('\n\n')
m_diag_np = np.diag(np.diag(m_sqr_mx))
print(m_diag_np)
```

```
[1 5 9]



[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

единичная матрица

In [ ]:
```
m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
m_e_np = np.matrix(m_e)
print(m_e_np)
```

```
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

In [ ]:
```
m_eye = np.eye(3)
print(m_eye)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:
```
m_idnt = np.identity(3)
print(m_idnt)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

нулевая матрица

In [ ]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

2. Как выполняется транспонирование матриц?

С помощью transpose()

A = np.matrix('1 2 3; 4 5 6')

A_t = A.transpose()

3. Приведите свойства операции транспонирования матриц.

Свойство 1. Дважды транспонированная матрица равна исходной матрице.

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц.

Свойство 3. Транспонирование произведения матриц равно произведению транспонированных матриц, расставленных в обратном порядке.

Свойство 4. Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу.

Свойство 5. Определители исходной и транспонированной матрицы совпадают.

4. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц?

В библиотеки NumPy для транспонирования двумерных матриц используется метод transpose().

5. Какие существуют основные действия над матрицами?

- Сложение матриц
- Умножение матрицы на число
- Умножение матриц

6. Как осуществляется умножение матрицы на число?

Умножение матрицы на число

```
In [ ]: A = np.matrix('1 2; 3 4')
        print(A * 3)

        [[ 3  6]
         [ 9 12]]
```

7. Какие свойства операции умножения матрицы на число?

Свойство 1. Произведение единицы и любой заданной матрицы равно заданной матрице.

Свойство 2. Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы.

Свойство 3. Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел.

Свойство 4. Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число.

Свойство 5. Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число.

8. Как осуществляется операции сложения и вычитания матриц?

```
In [ ]: A = np.matrix('1 2; 3 4')
        B = np.matrix('5 6; 7 8')
        L = A + B
        R = B + A
        print(L)
        print(R)
```

```
[[ 6  8]
 [10 12]]
[[ 6  8]
 [10 12]]
```

```
In [ ]: A = np.matrix('1 2; 3 4')
        B = np.matrix('5 6; 7 8')
        C = np.matrix('1 7; 9 3')
        L = A + (B + C)
        R = (A + B) + C
        print(L)
        print(R)
```

```
[[ 7 15]
 [19 15]]
[[ 7 15]
 [19 15]]
```

```
In [ ]: A = np.matrix('1 2; 3 4')
        Z = np.matrix('0 0; 0 0')
        L = A + (-1)*A
        print(L)
        print(Z)
```

```
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
```

Аналогично вычитание

9. Каковы свойства операций сложения и вычитания матриц?

Свойство 1. Коммутативность сложения. От перестановки матриц их сумма не изменяется.

Свойство 2. Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться.

Свойство 3. Для любой матрицы существует противоположная ей, такая, что их сумма является нулевой матрицей

10. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц?

Для сложения и вычитания используется + и -.

11. Как осуществляется операция умножения матриц?

Умножение матриц

```
]: A = np.matrix('1 2 3; 4 5 6')
   B = np.matrix('7 8; 9 1; 2 3')
   C = A.dot(B)
   print(C)

   [[31 19]
    [85 55]]
```

12. Каковы свойства операции умножения матриц?

Свойство 1. Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет выполняться эта операция.

Свойство 2. Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц.

Свойство 3. Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется правило независимости произведения от перестановки множителей.

Свойство 4. Произведение заданной матрицы на единичную равно исходной матрице.

Свойство 5. Произведение заданной матрицы на нулевую матрицу равно нулевой матрице.

13. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц?

В библиотеки NumPy для выполнения операции умножения матриц используется функция dot().

14. Что такое определитель матрицы? Каковы свойства определителя матрицы?

Определитель матрицы размера (n-го порядка) является одной из ее численных характеристик. Определитель матрицы A обозначается как |A| или det(A), его также называют детерминантом.

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании.

Свойство 2. Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю.

Свойство 3. При перестановке строк матрицы знак ее определителя меняется на противоположный.

Свойство 4. Если у матрицы есть две одинаковые строки, то ее определитель равен нулю.

Свойство 5. Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число.

Свойство 6. Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц.

Свойство 7. Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится.

Свойство 8. Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю

15. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?

В библиотеки NumPy для нахождения значения определителя матрицы используется функция linalg.det().

16. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству: $A * A^{-1} = E$

Обратную матрицу A − 1 A^{-1}A −1 к матрице A AA можно найти по формуле:

$A^{-1} = 1/\det A \cdot A^{*}$,

где detA — определитель матрицы A,

A^{*} – транспонированная матрица алгебраических дополнений к матрице A.

17. Каковы свойства обратной матрицы?

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица.

Свойство 2. Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы.

Свойство 3. Обратная матрица произведения матриц равна произведению обратных матриц.

18. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы?

В библиотеки NumPy для нахождения обратной матрицы используется функция linalg.inv().

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Метод Крамера

Для решения системы линейных уравнений методом Крамера из коэффициентов при неизвестных составляется главный определитель системы

Создадим и заполним матрицу

```
A = np.random.randint(-5, 5,(3,3))
B = np.random.randint(-5, 5,(3,1))
print(A)
print(B)
```

```
[[-3 -2  1]
 [ 2 -5 -5]
 [-4  0 -2]]
[[ 1]
 [ 1]
 [-2]]
```

```
#Найдем определитель
A_det  = np.linalg.det(A)

if round(A_det) != 0:
    #Найдем дополнительные определители
    d_1 = A.copy()
    d_1 [:, 0] = B[:, 0]
    x1 = round(np.linalg.det(d_1), 3) / round(A_det, 3)

    d_2 = A.copy()
    d_2[:, 1] = B[:, 0]
    x2 = round(np.linalg.det(d_2), 3) / round(A_det, 3)

    d_3 = A.copy()
    d_3[:, 2] = B[:, 0]
    x3 = round(np.linalg.det(d_3), 3) / round(A_det, 3)

    print("x1:", x1, "\nx2:", x2, "\nx3:", x3)
else:
    print("Матрица вырожденная, нельзя продолжить")
```

```
x1: 0.24489795918367346
x2: -0.6122448979591837
x3: 0.5102040816326531
```

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy

Матричный метод

]:
```
import numpy as np
```

решение системы линейных алгебраических уравнений по матричному методу определяется равенством

$$X = A^{-1} * B$$

В случае, если det A не равен нулю, у системы имеется только один вариант решения: при помощи метода обратной матрицы. Если det A=0, то систему нельзя решить данным методом.

Создадим и заполним матрицу

]:
```
A = np.random.randint(-5, 5,(3,3))
B = np.random.randint(-5, 5,(3,1))
print(A)
print(B)
```

```
[[ 4 -5  1]
 [-3 -1  0]
 [ 0  4  3]]
[[-1]
 [-4]
 [-3]]
```

]:
```
#Найдем определитель
A_det = np.linalg.det(A)

if round(A_det) != 0:
    #Найдем обратную и присоединенную матрицу
    A_inv = np.linalg.inv(A)
    R = A_inv.dot(B)
    i = 1
    for elem in R:
        print(f"x{i} = {elem}")
        i += 1
else:
    print("определитель равен 0, систему решить нельзя")
```

```
x1 = [1.10144928]
x2 = [0.69565217]
x3 = [-1.92753623]
```