МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО - КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ
КАФЕДРА ИНФОКОММУНИКАЦИЙ

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

Отчет
по лабораторной работе №3
дисциплины
«Теория распознаваныя образов»

Выполнил:
Гълбачева Доротея Андреева
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения
_____
(подпись) Проверил:
_____
(подпись)

Отчет защищен с оценкой_____ Дата защиты_____

Ставрополь, 2023 г.

Цель работы: Исследовать методы работы с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.

Выполнения лабораторной работы:

1. **Проработка примеров из лабораторной работы:**

Вектор

- Вектор-строка

```
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((5,))
        print(v_hor_zeros_v1)
        [0. 0. 0. 0. 0.]

In [5]: v_hor_zeros_v2 = np.zeros((1, 5))
        print(v_hor_zeros_v2)
        [[0. 0. 0. 0. 0.]]

In [7]: v_hor_one_v1 = np.ones((5,))
        print(v_hor_one_v1)
        [1. 1. 1. 1. 1.]

In [8]: v_hor_one_v2 = np.ones((1, 5))
        print(v_hor_one_v2)
        [[1. 1. 1. 1. 1.]]
```

Рисунок 3.1 – Использовать функции для создания нулевых и единичных векторов-строков

- Вектор-столбец

```
In [9]: v_vert_np = np.array([[1], [2]])
        print(v_vert_np)

[[1]
 [2]]

In [10]: v_vert_zeros = np.zeros((5, 1))
         print(v_vert_zeros)

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

In [11]: v_vert_ones = np.ones((5, 1))
         print(v_vert_ones)

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

Рисунок 3.2 – Использовать функции для создания нулевых и единичных векторов-столбцов

Матрица

- Квадратная матрица

```
In [12]: 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 [13]: 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]]
```

Рисунок 3.3 – Использовать функции array() и np.array() для создания квадратных матриц

```
In [16]: 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]]
```

Рисунок 3.4 – Создания квадратных матриц с использованием метода matrix

```
In [15]: 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]]
```

Рисунок 3.5 – Создания квадратных матриц с использованием стиля Matlab

- Диагональная матрица

```
In [17]: 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]]
```

Рисунок 3.6 – Построения диагональную матрицу вручную

```
In [18]: m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
         diag = np.diag(m_sqr_mx)
         print(diag)

         [1 5 9]
```

Рисунок 3.7 – Извлечения главную диагональ из матрицы

```
In [19]: m_diag_np = np.diag(np.diag(m_sqr_mx))
         print(m_diag_np)

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

Рисунок 3.8 – Построения диагональную матрицу на базе полученной диагонали

- Единичная матрица

```
In [20]: 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]]
```

Рисунок 3.9 – Создания единичную матрицу на базе списка

```
In [21]: m_eye = np.eye(3)
         print(m_eye)

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

Рисунок 3.10 – Построения единичную матрицу с помощью функции eye()

```
In [22]: m_idnt = np.identity(3)
         print(m_idnt)

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

Рисунок 3.11 – Построения единичную матрицу с помощью функции identity()

- Нулевая матрица

```
In [5]: m_zeros = np.zeros((3, 3))
        print(m_zeros)

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

Рисунок 3.12 – Построения нулевую матрицу с помощью функции zeros()

- Задание матрицы в общем виде

```
In [6]: m_mx = np.matrix('1 2 3; 4 5 6')
        print(m_mx)

[[1 2 3]
 [4 5 6]]
```

Рисунок 3.13 – Создания матрицу с помощью функции matrix()

- Транспонирование матрицы

```
In [9]: A = np.matrix('1 2 3; 4 5 6')
        print(A)

[[1 2 3]
 [4 5 6]]
```

Рисунок 3.14 – Создания матрицу A

```
In [10]: A_t = A.transpose()
         print(A_t)

[[1 4]
 [2 5]
 [3 6]]
```

Рисунок 3.15 – Транспонировка матрицу с помощью метода transpose()

```
In [11]: print(A.T)

[[1 4]
 [2 5]
 [3 6]]
```

Рисунок 3.16 – Сокращенный вариант получения транспонированной матрицы

```
In [12]: A = np.matrix('1 2 3; 4 5 6')
         print(A)

[[1 2 3]
 [4 5 6]]

In [13]: R = (A.T).T
         print(R)

[[1 2 3]
 [4 5 6]]
```

Рисунок 3.17 – Матрица транспонированная дважды

```
In [14]: A = np.matrix('1 2 3; 4 5 6')
         B = np.matrix('7 8 9; 0 7 5')
         L = (A + B).T
         print(L)

[[ 8  4]
 [10 12]
 [12 11]]

In [15]: R = A.T + B.T
         print(R)

[[ 8  4]
 [10 12]
 [12 11]]
```

Рисунок 3.18 – Суммы матриц равно сумме транспонированных матриц

```
In [16]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         L = (A.dot(B)).T
         print(L)

[[19 43]
 [22 50]]

In [17]: R = (B.T).dot(A.T)
         print(R)

[[19 43]
 [22 50]]
```

Рисунок 3.19 – Произведению транспонированных матриц

```
In [18]: A = np.matrix('1 2 3; 4 5 6')
         k = 3
         L = (k * A).T
         print(L)

[[ 3 12]
 [ 6 15]
 [ 9 18]]

In [19]: R = k * (A.T)
         print(R)

[[ 3 12]
 [ 6 15]
 [ 9 18]]
```

Рисунок 3.20 – Транспонирование произведения матрицы на число

```
In [20]: A = np.matrix('1 2; 3 4')
         A_det = np.linalg.det(A)
         A_T_det = np.linalg.det(A.T)
         print(format(A_det, '.9g'))

-2

In [21]: print(format(A_T_det, '.9g'))

-2
```

Рисунок 3.21 – Определители исходной и транспонированной матрицы

Действия над матрицами

- Умножение матрицы на число

```
In [22]: A = np.matrix('1 2 3; 4 5 6')
         C = 3 * A
         print(C)

[[ 3  6  9]
 [12 15 18]]
```

Рисунок 3.22 – Умножении матрицы на 3

```
In [23]: A = np.matrix('1 2; 3 4')
         L = 1 * A
         print(L)

[[1 2]
 [3 4]]

In [24]: R = A
         print(R)

[[1 2]
 [3 4]]
```

Рисунок 3.23 – Умножении матрицы на 1

```
In [25]: A = np.matrix('1 2; 3 4')
         Z = np.matrix('0 0; 0 0')
         L = 0 * A
         print(L)

[[0 0]
 [0 0]]

In [26]: R = Z
         print(R)

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

Рисунок 3.24 – Умножении матрицы на 0

```
In [27]: A = np.matrix('1 2; 3 4')
         p = 2
         q = 3
         L = (p + q) * A
         print(L)

[[ 5 10]
 [15 20]]

In [28]: R = p * A + q * A
         print(R)

[[ 5 10]
 [15 20]]
```

Рисунок 3.25 – Произведение матрицы на сумму чисел

```
In [29]: A = np.matrix('1 2; 3 4')
         p = 2
         q = 3
         L = (p * q) * A
         print(L)

[[ 6 12]
 [18 24]]

In [30]: R = p * (q * A)
         print(R)

[[ 6 12]
 [18 24]]
```

Рисунок 3.26 – Произведение матрицы на произведение двух чисел

```
In [31]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         k = 3
         L = k * (A + B)
         print(L)

[[18 24]
 [30 36]]

In [32]: R = k * A + k * B
         print(R)

[[18 24]
 [30 36]]
```

Рисунок 3.27 – Произведение суммы матриц

- Сложение матриц

```
In [33]: A = np.matrix('1 6 3; 8 2 7')
         B = np.matrix('8 1 5; 6 9 12')
         C = A + B
         print(C)

         [[ 9  7  8]
          [14 11 19]]
```

Рисунок 3.28 – Складывание матрицы

```
In [34]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         L = A + B
         print(L)

         [[ 6  8]
          [10 12]]

In [35]: R = B + A
         print(R)

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

Рисунок 3.29 – Результат коммутативность сложения

```
In [36]: 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)
         print(L)

         [[ 7 15]
          [19 15]]

In [37]: R = (A + B) + C
         print(R)

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

Рисунок 3.30 – Результат ассоциативность сложения

```
In [38]: A = np.matrix('1 2; 3 4')
         Z = np.matrix('0 0; 0 0')
         L = A + (-1)*A
         print(L)

         [[0 0]
          [0 0]]

In [39]: print(Z)

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

Рисунок 3.31 – Результат противоположная нулевая матрица

- Умножение матриц

```
In [40]: 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]]
```

Рисунок 3.32 – Результат умножения матриц

```
In [41]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         C = np.matrix('2 4; 7 8')
         L = A.dot(B.dot(C))
         R = (A.dot(B)).dot(C)
         print(L)

         [[192 252]
          [436 572]]

In [42]: print(R)

         [[192 252]
          [436 572]]
```

Рисунок 3.33 – Результат ассоциативность умножения

```
In [43]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         C = np.matrix('2 4; 7 8')
         L = A.dot(B + C)
         R = A.dot(B) + A.dot(C)
         print(L)

         [[35 42]
          [77 94]]

In [44]: print(R)

         [[35 42]
          [77 94]]
```

Рисунок 3.34 – Результат дистрибутивность умножения

```
In [45]: A = np.matrix('1 2; 3 4')
         B = np.matrix('5 6; 7 8')
         L = A.dot(B)
         R = B.dot(A)
         print(L)

         [[19 22]
          [43 50]]

In [46]: print(R)

         [[23 34]
          [31 46]]
```

Рисунок 3.35 – Результат умножение матриц в общем виде не коммутативно

```
In [47]: A = np.matrix('1 2; 3 4')
         E = np.matrix('1 0; 0 1')
         L = E.dot(A)
         R = A.dot(E)
         print(L)

         [[1 2]
          [3 4]]

In [48]: print(R)

         [[1 2]
          [3 4]]

In [49]: print(A)

         [[1 2]
          [3 4]]
```

Рисунок 3.36 – Произведение заданной матрицы на единичную

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

        [[0 0]
         [0 0]]

In [4]: print(R)

        [[0 0]
         [0 0]]

In [5]: print(Z)

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

Рисунок 3.37 – Произведение заданной матрицы на нулевую матрицу

Определитель матрицы

- Свойства определителя матрицы

```
In [8]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
        print(A)

        [[-4 -1  2]
         [10  4 -1]
         [ 8  3  1]]

In [9]: print(A.T)

        [[-4 10  8]
         [-1  4  3]
         [ 2 -1  1]]

In [10]: det_A = round(np.linalg.det(A), 3)
         det_A_t = round(np.linalg.det(A.T), 3)
         print(det_A)

         -14.0

In [11]: print(det_A_t)

         -14.0
```

Рисунок 3.38 – Определитель матрицы остается неизменным при ее транспонировании

```
In [12]: A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
         print(A)

         [[-4 -1  2]
          [ 0  0  0]
          [ 8  3  1]]

In [13]: np.linalg.det(A)

Out[13]: 0.0
```

Рисунок 3.39 – Определитель матрицы равен нулю, когда у матрицы есть строка или столбец, состоящие из нулей

```
In [14]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [15]: B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
         print(B)

         [[10  4 -1]
          [-4 -1  2]
          [ 8  3  1]]

In [16]: round(np.linalg.det(A), 3)

Out[16]: -14.0

In [17]: round(np.linalg.det(B), 3)

Out[17]: 14.0
```

Рисунок 3.40 – Результат знака определителя противоположный, после перестановке строк матрицы

```
In [18]: A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
         print(A)

         [[-4 -1  2]
          [-4 -1  2]
          [ 8  3  1]]

In [19]: np.linalg.det(A)

Out[19]: 0.0
```

Рисунок 3.41 – Результат определителя равен нулю, когда у матрицы есть две одинаковые строки

```
In [20]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [21]: k = 2
         B = A.copy()
         B[2, :] = k * B[2, :]
         print(B)

         [[-4 -1  2]
          [10  4 -1]
          [16  6  2]]

In [22]: det_A = round(np.linalg.det(A), 3)
         det_B = round(np.linalg.det(B), 3)
         det_A * k

Out[22]: -28.0

In [23]: det_B

Out[23]: -28.0
```

Рисунок 3.42 – Результат определителя будет умножен на то число, на которой все элементы строки или столбца матрицы умноженные

```
In [24]: A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
         B = np.matrix('-4 -1 2; 8 3 2; 8 3 1')
         C = A.copy()
         C[1, :] += B[1, :]
         print(C)

         [[-4 -1  2]
          [ 4  2  4]
          [ 8  3  1]]

In [25]: print(A)

         [[-4 -1  2]
          [-4 -1  2]
          [ 8  3  1]]

In [26]: print(B)

         [[-4 -1  2]
          [ 8  3  2]
          [ 8  3  1]]

In [27]: round(np.linalg.det(C), 3)

Out[27]: 4.0

In [28]: round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)

Out[28]: 4.0
```

Рисунок 3.43 – Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц

```
In [29]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         k = 2
         B = A.copy()
         B[1, :] = B[1, :] + k * B[0, :]
         print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [30]: print(B)

         [[-4 -1  2]
          [ 2  2  3]
          [ 8  3  1]]

In [31]: round(np.linalg.det(A), 3)

Out[31]: -14.0

In [32]: round(np.linalg.det(B), 3)

Out[32]: -14.0
```

Рисунок 3.44 – Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится

```
In [33]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [34]: k = 2
         A[1, :] = A[0, :] + k * A[2, :]
         round(np.linalg.det(A), 3)

Out[34]: 0.0
```

Рисунок 3.45 – Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю

```
In [35]: A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [36]: k = 2
         A[1, :] = k * A[0, :]
         print(A)

         [[-4 -1  2]
          [-8 -2  4]
          [ 8  3  1]]

In [37]: round(np.linalg.det(A), 3)

Out[37]: 0.0
```

Рисунок 3.46 – Если матрица содержит пропорциональные строки, то ее определитель равен нулю

Обратная матрица

```
In [38]: A = np.matrix('1 -3; 2 5')
         A_inv = np.linalg.inv(A)
         print(A_inv)

         [[ 0.45454545  0.27272727]
          [-0.18181818  0.09090909]]
```

Рисунок 3.47 – Результат получения обратной матрицы с использованием функцию *inv()*

```
In [39]: A = np.matrix('1. -3.; 2. 5.')
         A_inv = np.linalg.inv(A)
         A_inv_inv = np.linalg.inv(A_inv)
         print(A)

         [[ 1. -3.]
          [ 2.  5.]]

In [40]: print(A_inv_inv)

         [[ 1. -3.]
          [ 2.  5.]]
```

Рисунок 3.48 – Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы

```
In [41]: A = np.matrix('1. -3.; 2. 5.')
         B = np.matrix('7. 6.; 1. 8.')
         L = np.linalg.inv(A.dot(B))
         R = np.linalg.inv(B).dot(np.linalg.inv(A))
         print(L)

[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]

In [42]: print(R)

[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
```

Рисунок 3.49 – Обратная матрица произведения матриц равна произведению обратных матриц

Ранг матрицы

```
In [43]: m_eye = np.eye(4)
         print(m_eye)

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

In [44]: rank = np.linalg.matrix_rank(m_eye)
         print(rank)

4
```

Рисунок 3.50 – Создание единичную матрицу с использованием функцией matrix_rank()

```
In [45]: m_eye[3][3] = 0
         print(m_eye)

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

In [46]: rank = np.linalg.matrix_rank(m_eye)
         print(rank)

3
```

Рисунок 3.51 – Ранг станет равен трем, когда приравняем элемент в нижнем правом углу к нулю

3. Вопросы для защиты работы

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

2. Как выполняется транспонирование матриц?

3. Приведите свойства операции транспонирования матриц.

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

5. Какие существуют основные действия над матрицами?

6. Как осуществляется умножение матрицы на число?

7. Какие свойства операции умножения матрицы на число?

8. Как осуществляется операции сложения и вычитания матриц?

9. Каковы свойства операций сложения и вычитания матриц?

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

11. Как осуществляется операция умножения матриц?

12. Каковы свойства операции умножения матриц?

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

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

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

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

17. Каковы свойства обратной матрицы?

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

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений.

Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

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