МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт о лабораторной работе №3 по дисциплине технологии распознавания образов

Выполнила:
Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np)
[1 2]
```

Рисунок 1 – Пример создания вектора

```
>>> import numpy as np
>>> v_hor_zeros_v1 = np.zeros((5,))
>>> print(v_hor_zeros_v1)
[0. 0. 0. 0. 0.]
```

Рисунок 2 – Пример создания нулевого вектора-строки

```
>>> v_hor_one_v1 = np.ones((5,))
>>> print(v_hor_one_v1)
[1. 1. 1. 1. 1.]
```

Рисунок 3 – Пример создания нулевого вектора-строки размера 5

```
>>> v_hor_one_v2 = np.ones((1, 5))
>>> print(v_hor_zeros_v2)
[[0. 0. 0. 0. 0.]]
```

Рисунок 4 – Пример создания вектор-строки, который является элементов массива

```
>>> v_hor_one_v1 = np.ones((5,))
>>> print(v_hor_one_v1)
[1. 1. 1. 1. 1.]
>>> v_hor_one_v2 = np.ones((1, 5))
>>> print(v_hor_one_v2)
[[1. 1. 1. 1. 1.]]
```

Рисунок 5 – Пример создания единичной вектор-строки в обоих из представленных для нулевого вектора-строки форм

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

Рисунок 6 – Пример вектор-столбца в общем виде

```
>>> v_vert_zeros = np.zeros((5, 1))
>>> print(v_vert_zeros)
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

Рисунок 7 – Пример создания нулевого вектор-столбца

```
>>> v_vert_zeros = np.ones((5, 1))
>>> print(v_vert_zeros)
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

Рисунок 8 – Пример создания единичного вектор-столбца

```
>>> 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 9 – Пример создания и вывода квадратной матрицы

```
>>> 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 10 – Пример создания объекта типа matrix()

```
>>> 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 11 – Пример создания объекта типа matrix()

```
>>> 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 12 – Пример создания диагональной матрицы

```
>>> 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 13 – Пример вывода элементов диагональной матрицы

```
>>> m_diag_np = np.diag(np.diag(m_sqr_mx))
>>> print(m_diag_np)
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

Рисунок 14 – Пример построения диагональной матрицы на базе полученной диагонали

```
>>> 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 15 – Пример создания единичной матрицы

```
>>> m_eye = np.eye(3)
>>> print(m_eye)
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Рисунок 16 – Пример использования функции eye()

```
>>> m_idnt = np.identity(3)
>>> print(m_idnt)
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Рисунок 17 – Пример использования функции identity()

```
>>> m_zeros = np.zeros((3, 3))
>>> print(m_zeros)
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Рисунок 18 – Пример создания нулевой матрицы

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

Рисунок 18 – Пример задания матрицы в общем виде

```
>>> m_var = np.zeros((2, 5))
>>> print(m_var)
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Рисунок 19 – Пример задания матрицы с произвольным содержим и использование функции zeros()

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

Рисунок 20 – Пример создания матрицы

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
 [2 5]
 [3 6]]
```

Рисунок 21 – Пример транспонирования матрицы с помощью метода transpose()

```
>>> print(A.T)
[[1 4]
 [2 5]
 [3 6]]
```

Рисунок 22 – Пример транспонирования матрицы с помощью метода .T

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
[[1 2 3]
 [4 5 6]]
>>> R = (A.T).T
>>> print(R)
[[1 2 3]
 [4 5 6]]
```

Рисунок 23 – Пример свойства транспонирования матриц

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8 9; 0 7 5')
>>> L = (A + B).T
>>> R = A.T + B.T
>>> print(L)
[[ 8  4]
 [10 12]
 [12 11]]
>>> print(R)
[[ 8  4]
 [10 12]
 [12 11]]
```

Рисунок 24 – Пример свойства транспонирования матриц

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> k = 3
>>> L = (k * A).T
>>> R = k * (A.T)
>>> print(L)
[[ 3 12]
 [ 6 15]
 [ 9 18]]
>>> print(R)
[[ 3 12]
 [ 6 15]
 [ 9 18]]
```

Рисунок 25 – Пример решения задания №4

```
>>> 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
>>> print(format(A_T_det, '.9g'))
-2
```

Рисунок 26 – Пример свойства транспонирования матриц

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
[[ 3  6  9]
 [12 15 18]]
```

Рисунок 27 – Пример умножения матрицы на число

```
>>> A = np.matrix('1 2; 3 4')
>>> L = 1 * A
>>> R = A
>>> print(L)
[[1 2]
 [3 4]]
>>> print(R)
[[1 2]
 [3 4]]
```

Рисунок 28 – Пример свойства умножения матрицы на число

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

Рисунок 29 – Пример свойства умножения матрицы на число

```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p + q) * A
>>> R = p * A + q * A
>>> print(L)
[[ 5 10]
 [15 20]]
>>> print(R)
[[ 5 10]
 [15 20]]
```

Рисунок 30 – Пример свойства умножения матрицы на число

```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p * q) * A
>>> R = p * (q * A)
>>> print(L)
[[ 6 12]
 [18 24]]
>>> print(R)
[[ 6 12]
 [18 24]]
```

Рисунок 31 – Пример свойства умножения матрицы на число

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> k = 3
>>> L = k * (A + B)
>>> R = k * A + k * B
>>> print(L)
[[18 24]
 [30 36]]
>>> print(R)
[[18 24]
 [30 36]]
```

Рисунок 32 – Пример свойства умножения матрицы на число

```
>>> 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 33 – Пример сложения матриц

```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A + B
>>> R = B + A
>>> print(L)
[[ 6  8]
 [10 12]]
>>> print(R)
[[ 6  8]
 [10 12]]
```

Рисунок 34 – Пример свойства сложения матриц

```
>>> 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)
[[ 7 15]
 [19 15]]
>>> print(R)
[[ 7 15]
 [19 15]]
```

Рисунок 35 – Пример свойства сложения матриц

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

Рисунок 36 – Пример свойства сложения 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]]
```

Рисунок 37 – Пример умножения матриц

```
>>> 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]]
>>> print(R)
[[192 252]
 [436 572]]
```

Рисунок 38 – Пример свойства умножения матриц

```
>>> 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]]
>>> print(R)
[[35 42]
 [77 94]]
```

Рисунок 39 – Пример свойства умножения матриц

```
>>> 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]]
>>> print(R)
[[23 34]
 [31 46]]
```

Рисунок 40 – Пример свойства умножения матриц

```
>>> 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]]
>>> print(R)
[[1 2]
 [3 4]]
>>> print(A)
[[1 2]
 [3 4]]
```

Рисунок 41 – Пример свойства умножения матриц

```
>>> 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]]
>>> print(R)
[[0 0]
 [0 0]]
>>> print(Z)
[[0 0]
 [0 0]]
```

Рисунок 42 – Пример свойства умножения матриц

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

Рисунок 43 – Пример задания матрицы

```
>>> np.linalg.det(A)
-14.000000000000009
```

Рисунок 44 – Пример вычислителя определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
>>> print(A.T)
[[-4 10  8]
 [-1  4  3]
 [ 2 -1  1]]
>>> det_A = round(np.linalg.det(A), 3)
>>> det_A_t = round(np.linalg.det(A.T), 3)
>>> print(det_A)
-14.0
>>> print(det_A_t)
-14.0
```

Рисунок 45 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [ 0  0  0]
 [ 8  3  1]]
>>> np.linalg.det(A)
0.0
```

Рисунок 46 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
>>> B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
>>> print(B)
[[10  4 -1]
 [-4 -1  2]
 [ 8  3  1]]
>>> round(np.linalg.det(A), 3)
-14.0
>>> round(np.linalg.det(B), 3)
14.0
```

Рисунок 47 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [-4 -1  2]
 [ 8  3  1]]
>>> np.linalg.det(A)
0.0
```

Рисунок 48 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
>>> k = 2
>>> B[2, :] = k * B[2, :]
>>> print(B)
[[10  4 -1]
 [-4 -1  2]
 [16  6  2]]
>>> det_A = round(np.linalg.det(A), 3)
>>> det_B = round(np.linalg.det(B), 3)
>>> det_A * k
-28.0
>>> det_B
28.0
```

Рисунок 49 – Пример свойства определителя матрицы

```
>>> 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]]
>>> print(A)
[[-4 -1  2]
 [-4 -1  2]
 [ 8  3  1]]
>>> print(B)
[[-4 -1  2]
 [ 8  3  2]
 [ 8  3  1]]
>>> round(np.linalg.det(C), 3)
4.0
>>> round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
4.0
```

Рисунок 50 – Пример свойства определителя матрицы

```
>>> 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]]
>>> print(B)
[[-4 -1  2]
 [ 2  2  3]
 [ 8  3  1]]
>>> round(np.linalg.det(A), 3)
-14.0
>>> round(np.linalg.det(B), 3)
-14.0
```

Рисунок 51 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
>>> A[1, :] = A[0, :] + k * A[2, :]
>>> round(np.linalg.det(A), 3)
0.0
```

Рисунок 52 – Пример свойства определителя матрицы

```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
>>> k = 2
>>> A[1, :] = k * A[0, :]
>>> print(A)
[[-4 -1  2]
 [-8 -2  4]
 [ 8  3  1]]
>>> round(np.linalg.det(A), 3)
0.0
```

Рисунок 53 – Пример свойства определителя матрицы

```
>>> 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 54 – Пример вычисления обратной матрицы

```
>>> 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.]]
>>> print(A_inv_inv)
[[ 1. -3.]
 [ 2.  5.]]
```

Рисунок 55 – Пример свойства вычисления обратной матрицы

```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> L = np.linalg.inv(A.T)
>>> R = (np.linalg.inv(A)).T
>>> print(L)
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
>>> print(R)
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
```

Рисунок 56 – Пример свойства вычисления обратной матрицы

```
>>> 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]]
>>> print(R)
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
```

Рисунок 57 – Пример свойства вычисления обратной матрицы

```
>>> m_eye = np.eye(4)
>>> print(m_eye)
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Рисунок 58 – Пример вычисления ранга матрицы

```
>>> rank = np.linalg.matrix_rank(m_eye)
>>> print(rank)
4
```

Рисунок 59 – Пример использования функции matrix_rank()

```
>>> m_eye[3][3] = 0
>>> print(m_eye)
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
>>> rank = np.linalg.matrix_rank(m_eye)
>>> print(rank)
3
```

Рисунок 60 – Пример использования функции matrix_rank()

2. Вопросы для защиты

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

Способ задания вектора-строки:

```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np )
[1 2]
```

Способ создания нулевого вектора-строки:

```
>>> v_hor_zeros_v1 = np.zeros((5,))
>>> print(v_hor_zeros_v1 )
[0. 0. 0. 0. 0.]
```

Способ создания единичного вектора-строки:

```
>>> v_hor_one_v1 = np.ones((5,))
>>> print(v_hor_one_v1)
[1. 1. 1. 1. 1.]
```

Способ создания вектора-столбца:

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

Способ создания нулевого вектора-столбца:

```
>>> v_vert_zeros = np.zeros((5, 1))
>>> print(v_vert_zeros)
[[0.]
[0.]
[0.]
[0.]
[0.]]
```

Способ создания единичного вектора-столбца:

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

Способ создания квадратной матрицы:

```
>>> 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]]
```

Способ создания диагональной 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]]
```

Способ создания единичной матрицы:

```
>>> 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]]
```

Способ создания нулевой матрицы:

```
>>> m_zeros = np.zeros((3, 3))
>>> print(m_zeros)
[[ 0. 0. 0.]
[ 0. 0. 0.]
[ 0. 0. 0.]]
```

2. **Как выполняется транспонирование матриц?**

Пример №1

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
[2 5]
[3 6]]
```

Пример №2

```
>>> print(A.T)
[[1 4]
[2 5]
[3 6]]
```

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?**

Умножение матрицы на число

Сложение матриц

Умножение матриц

6. **Как осуществляется умножение матрицы на число?**

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
[[ 3  6 9]
[12 15 18]]
```

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?

Сложение матриц:

```
>>> 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]]
```

Вычитание:

```
>>> import numpy as np
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('1 2 3; 4 5 6')
>>> C = A - B
>>> print(C)
[[0 0 0]
 [0 0 0]]
```

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?

```
>>> 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?

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?

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

Для того, чтобы у квадратной матрицы A была обратная матрица необходимо и достаточно чтобы определитель $|A|$ был не равен нулю. Введем понятие **союзной матрицы**. Союзная матрица A строится на базе исходной A путем замены всех элементов матрицы A на их алгебраические дополнения.

Исходная матрица:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Союзная ей матрица A:

$$A^* = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

Транспонируя матрицу A, мы получим так называемую присоединенную матрицу A^T:

$$A^{*T} = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}.$$

Теперь, зная как вычислять определитель и присоединенную матрицу, мы можем определить матрицу A^{-1}, обратную матрице A:

$$A^{-1} = \frac{1}{det(A)} \times A^{*T}.$$

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?

```
>>> A = np.matrix('1 -3; 2 5')
>>> A_inv = np.linalg.inv(A)
>>> print(A_inv)
[[ 0.45454545 0.27272727]
 [-0.18181818 0.09090909]]
```

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

1. Необходимо вычислить определитель матрицы системы и убедиться, что он не равен нулю.
2. Найти определители матриц.
3. Вычислить неизвестные переменные при помощи деления определителя новых матриц, а определитель исходной матрицы.
4. Выполнить проверку результатов: если все определители являются тождествами, то решение найдено верно.

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

1. Необходимо вычислить определитель матрицы системы и убедиться, что он не равен нулю.
2. Найти обратную матрицу.
3. Умножить обратную матрицу на определитель и на столбец свободных членов.
4. Выполнить проверку результатов: если все определители являются тождествами, то решение найдено верно.

Ссылка на репозиторий: https://github.com/tamaranesterenko/TRO_LR_3