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

Отчет по лабораторной работе №3 по дисциплине:

Технологии распознавания образов

Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Выполнение примеров из методички:

In [2]:
```
import numpy as np
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 [4]:
```
v_hor_zeros_v2 = np.zeros((1, 5))
print(v_hor_zeros_v2 )
```
```
[[0. 0. 0. 0. 0.]]
```

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

In [6]:
```
v_hor_one_v2 = np.ones((1, 5))
print(v_hor_one_v2)
```
```
[[1. 1. 1. 1. 1.]]
```

In [7]:
```
v_vert_np = np.array([[1], [2]])
print(v_vert_np)
```
```
[[1]
 [2]]
```

In [8]:
```
v_vert_zeros = np.zeros((5, 1))
print(v_vert_zeros)
```
```
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

In [9]:
```
v_vert_ones = np.ones((5, 1))
print(v_vert_ones)
```
```
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

In [10]:
```
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 [11]:
```
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 [12]:
```
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]]
```

In [13]:
```
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
```

In [14]:
```
diag = np.diag(m_sqr_mx)
print(diag)
```
```
[1 5 9]
```

In [15]:
```
m_diag_np = np.diag(np.diag(m_sqr_mx))
print(m_diag_np)
```
```
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

In [16]:
```
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 [17]:
```
m_eye = np.eye(3)
print(m_eye)
```

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

In [18]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

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

In [19]:
```
m_mx = np.matrix('1 2 3; 4 5 6')
print(m_mx)
```

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

In [20]:
```
m_var = np.zeros((2, 5))
print(m_var)
```

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

In [21]:
```
A = np.matrix('1 2 3; 4 5 6')
print(A)
```

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

In [22]:
```
A_t = A.transpose()
print(A_t)
```

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

In [23]:
```
A = np.matrix('1 2 3; 4 5 6')
print(A)
R = (A.T).T
print(R)
```

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

In [24]:
```
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)
print(R)
```

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

In [25]:
```
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
L = (A.dot(B)).T
R = (B.T).dot(A.T)
print(L)
print(R)
```

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

In [26]:
```
A = np.matrix('1 2 3; 4 5 6')
k = 3
L = (k * A).T
R = k * (A.T)
print(L)
print(R)
```

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

In [27]:
```
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'))
print(format(A_T_det, '.9g'))
```

```
-2
-2
```

In [28]:
```
A = np.matrix('1 2 3; 4 5 6')
C = 3 * A
print(C)
```

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

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

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

In [30]:
```
A = np.matrix('1 2; 3 4')
Z = np.matrix('0 0; 0 0')
L = 0 * A
R = Z
print(L)
print(R)
```

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

In [31]:
```
A = np.matrix('1 2; 3 4')
p = 2
q = 3
L = (p + q) * A
R = p * A + q * A
print(L)
print(R)
```

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

In [32]:
```
A = np.matrix('1 2; 3 4')
p = 2
q = 3
L = (p * q) * A
R = p * (q * A)
print(L)
print(R)
```

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

In [33]:
```
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)
print(R)
```

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

In [34]:
```
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]]
```

In [35]:
```
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 [38]:
```
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]]
```

In [39]:
```
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]]
```

In [40]:
```
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)
print(R)
```

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

In [41]:
```
A = np.matrix('1 2; 3 4')
E = np.matrix('1 0; 0 1')
L = E.dot(A)
R = A.dot(E)
print(L)
print(R)
print(A)
```

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

In [42]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
```

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

In [43]:
```
np.linalg.det(A)
```

Out[43]:
```
-14.000000000000009
```

In [44]:
```
A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
print(A)
np.linalg.det(A)
```

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

Out[44]:
```
0.0
```

In [45]:
```
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)
print(A)
print(B)
round(np.linalg.det(C), 3)
round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
```

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

Out[45]:
```
4.0
```

In [46]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
k = 2
A[1, :] = k * A[0, :]
print(A)
round(np.linalg.det(A), 3)
```

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

Out[46]:
```
0.0
```

In [47]:
```
A = np.matrix('1 -3; 2 5')
A_inv = np.linalg.inv(A)
print(A_inv)
```

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

In [48]:
```
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 [49]:
```
rank = np.linalg.matrix_rank(m_eye)
print(rank)
```

```
4
```

Выполнение заданий:

In [3]:
```
import numpy as np
A = np.matrix('1 3 5; 6 2 9; 2 5 4')
B = np.matrix('2; 0; 4')
if round(np.linalg.det(A), 3) != 0: #Проверка значения определителя. Если определитель равен 0, то систему решить нельзя.
    A_inv = np.linalg.inv(A) #Вычисляем обратную матрицу
    R = A_inv.dot(B) #Умножаем обратную матрицу на столбец свободных членов B и получаем решение системы
    idx = 1
    for elem in R: #Выводим решение
        print(f"x{idx} = {elem}")
        idx += 1
```

```
x1 = [[-0.08]]
x2 = [[0.96]]
x3 = [[-0.16]]
```

In [4]:
```
A = np.matrix('1 3 5; 6 2 9; 2 5 4')
B = np.matrix('2; 0; 4')
if round(np.linalg.det(A), 3) != 0: #Проверка значения определителя. Если определитель равен 0, то систему решить нельзя.
    C = A.copy() #Копируем матрицу для работы с первым столбцом матрицы
    C[:, 0] = B[:, 0] #Заменяем первый столбец матрицы C на столбец свободных членов B
    x1 = round(np.linalg.det(C), 3) / round(np.linalg.det(A), 3) #Делим определитель новой матрицы на исходную матрицу A
    D = A.copy() #Копируем матрицу для работы со вторым столбцом матрицы
    D[:, 1] = B[:, 0] #Заменяем второй столбец матрицы C на столбец свободных членов B
    x2 = round(np.linalg.det(D), 3) / round(np.linalg.det(A), 3) #Делим определитель новой матрицы на исходную матрицу A
    F = A.copy() #Копируем матрицу для работы с третьим столбцом матрицы
    F[:, 2] = B[:, 0] #Заменяем третий столбец матрицы C на столбец свободных членов B
    x3 = round(np.linalg.det(F), 3) / round(np.linalg.det(A), 3) #Делим определитель новой матрицы на исходную матрицу A

    print("x1:", x1, "\nx2:", x2, "\nx3:", x3) #Выводим решение
```

```
x1: -0.08
x2: 0.96
x3: -0.16
```

Вопросы для защиты:

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

Вектор-строка:

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

Единичный вектор:

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

Вектор-столбец:

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

Нулевой вектор-столбец:

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

Единичный вектор-столбец:

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

Диагональная матрица:

```
>>> 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]]
```

Единичная матрица:

```
>>> 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]]
```

Нулевая матрица:

```
>>> 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 метода transpose():

```
>>> A_t = A.transpose()
>>> 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?

С помощью метода 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?

```
>>> 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]]
```

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 – соответственно.

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.
Свойство 9. Если матрица содержит пропорциональные строки, то ее определитель равен нулю.

15. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?
Для вычисления определителя этой матрицы воспользуемся функцией det() из пакета linalg (np.linalg.det(A))

16. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?
Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству:

$$A \times A^{-1} = A^{-1} \times A = E,$$

где – E это единичная матрица.

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.

– Необходимо вычислить определитель матрицы системы и убедиться, что он не равен нулю;

– Найти определители матрицы;

– Вычислить неизвестные переменные при помощи деления определителя новых матриц, а определитель исходной матрицы;

– 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.

– Необходимо вычислить определитель матрицы системы и убедиться, что он не равен нулю;

– Найти обратную матрицу;

– Умножить обратную матрицу на определитель и на столбец свободных членов;

– Выполнить проверку результатов: если все определители являются тождествами, то решение найдено верно.