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

Отчет по лабораторной работе №9

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

Дисциплина «Введение в системы искусственного интеллекта»

Выполнил: студент группы ИВТ-б-о-18-1 (1)
Скориков А.Ю.
_____(подпись)
Проверил: доцент кафедры инфокоммуникаций
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

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

Ход работы

Таблица 1 – Исходные данные

Номер варианта	10

Решение:

Ввод [1]:
```
#Импорт библиотеки
import numpy as np

#вектор-строки
v_hor_np = np.array([1, 2])
print("Создание вектора-строки")
print(v_hor_np)

#нулеовой вектор
print("Создание нулеового вектора")
print(np.zeros((5,)))

#единичный вектор
print("Создание единичного вектора")
print(np.ones((5,)))

#вектор-столбца
print("Создание вектора-столбца")
v_vert_np = np.array([[1], [2]])
print(v_vert_np)

#нулевой вектор-столбца
print("Создание нулевого вектора-столбца")
v_vert_zeros = np.zeros((5, 1))
print(v_vert_zeros)

#нулевой вектор-столбца
print("Создание единичного вектора-столбца")
v_vert_ones = np.ones((5, 1))
print(v_vert_ones)
```

```
Создание вектора-строки
[1 2]
Создание нулеового вектора
[0. 0. 0. 0. 0.]
Создание единичного вектора
[1. 1. 1. 1. 1.]
Создание вектора-столбца
[[1]
 [2]]
Создание нулевого вектора-столбца
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]]
Создание единичного вектора-столбца
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

Рисунок 1 – Решение задачи 1

```python
#Квадратная матрица
#Array
print("Методом массива")
m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_arr)

#Matrix
print("Методом Matrix")
m_sqr_mx = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(m_sqr_mx)

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

```
Методом массива
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Методом Matrix
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Методом Matlab
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Рисунок 2 – Решение задачи 2

```python
#Диагональная матрица
#Вручную
print("Вручную")
m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
m_diag_np = np.matrix(m_diag)
print(m_diag_np)

#средством NumPy
print("NumPy")
m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
diag = np.diag(m_sqr_mx)
m_diag_np = np.diag(np.diag(m_sqr_mx))
print(m_diag_np)
```

```
Вручную
[[1 0 0]
 [0 5 0]
 [0 0 9]]
NumPy
[[1 0 0]
 [0 5 0]
 [0 0 9]]
```

Рисунок 3 – Решение задачи 3

```
#Единичная матрица
# Вручную
print("Вручную")
m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
m_e_np = np.matrix(m_e)
print(m_e_np)

#Функция - eye()
print("eye()")
m_eye = np.eye(3)
print(m_eye)

#Функция - identity()
print("identity()")
m_idnt = np.identity(3)
print(m_idnt)
```

```
Вручную
[[1 0 0]
 [0 1 0]
 [0 0 1]]
eye()
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
identity()
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Рисунок 4 – Решение задачи 4

```
#Нулевая матрица
#Функция zeros()
print("zeros()")
m_zeros = np.zeros((3, 3))
print(m_zeros)

print("2x5")
m_var = np.zeros((2, 5))
print(m_var)
```

```
zeros()
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
2x5
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Рисунок 5 – Решение задачи 5

```
import numpy as np

#Транспонирование матрицы
print("Транспонирование матрицы")
A = np.matrix('1 2 3; 4 5 6')
print(A)
print("transpose():")
A_t = A.transpose()
print(A_t)
print("\n")

#Сокращенный вариант
print("Сокращенный вариант:")
print(A.T)
print("\n")

#Двойное транспонирование
print("Двойное транспонирование")
print(A)
print(A.T)
print((A.T).T)
print("\n")

#Сумма транспонированных матриц
print("Сумма транспонированных 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)
print(R)
print("\n")

#Произведение транспонированных матриц
print("Произведение транспонированных матриц")
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)
print("\n")

#Транспонирование произведения матрицы на число
print("Транспонирование произведения матрицы на число")
A = np.matrix('1 2 3; 4 5 6')
k = 3
L = (k * A).T
R = k * (A.T)
print(L)
print(R)
print("\n")
```

Рисунок 6 – Решение задачи 6 Ч.1

```python
#Определители исходной и транспонированной матрицы совпадают
print("Определители исходной и транспонированной матрицы совпадают")
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'))
```

```
Транспонирование матрицы
[[1 2 3]
 [4 5 6]]
transpose():
[[1 4]
 [2 5]
 [3 6]]


Сокращенный вариант:
[[1 4]
 [2 5]
 [3 6]]


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


Сумма транспонированных матриц
[[ 8  4]
 [10 12]
 [12 11]]
[[ 8  4]
 [10 12]
 [12 11]]


Произведение транспонированных матриц
[[19 43]
 [22 50]]
[[19 43]
 [22 50]]


Транспонирование произведения матрицы на число
[[ 3 12]
 [ 6 15]
 [ 9 18]]
[[ 3 12]
 [ 6 15]
 [ 9 18]]


Определители исходной и транспонированной матрицы совпадают
-2
-2
```

Рисунок 7 – Решение задачи 6 Ч.2

```
#Умножение матрицы на число
print("Умножение матрицы на число")
A = np.matrix('1 2 3; 4 5 6')
C = 3 * A
print(C)
print("\n")

#Произведение единицы и матрицы
print("Произведение единицы и матрицы")
A = np.matrix('1 2; 3 4')
L = 1 * A
R = A
print(L)
print(R)
print("\n")

#Произведение нуля и матрицы
print("Произведение нуля и матрицы")
A = np.matrix('1 2; 3 4')
Z = np.matrix('0 0; 0 0')
L = 0 * A
R = Z
print(L)
print(R)
print("\n")

#Произведение матрицы на сумму чисел
print("Произведение матрицы на сумму чисел")
A = np.matrix('1 2; 3 4')
p = 2
q = 3
L = (p + q) * A
R = p * A + q * A
print(L)
print(R)
print("\n")

#Произведение матрицы на произведение двух чисел
print("Произведение матрицы на произведение двух чисел")
A = np.matrix('1 2; 3 4')
p = 2
q = 3
L = (p * q) * A
R = p * (q * A)
print(L)
print(R)
print("\n")
```

Рисунок 8 – Решение задачи 7 Ч.1

```python
#Произведение суммы матриц на число
print("Произведение суммы матриц 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)
print(R)
print("\n")
```

```
Умножение матрицы на число
[[ 3  6  9]
 [12 15 18]]


Произведение единицы и матрицы
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]


Произведение нуля и матрицы
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]


Произведение матрицы на сумму чисел
[[ 5 10]
 [15 20]]
[[ 5 10]
 [15 20]]


Произведение матрицы на произведение двух чисел
[[ 6 12]
 [18 24]]
[[ 6 12]
 [18 24]]


Произведение суммы матриц на число
[[18 24]
 [30 36]]
[[18 24]
 [30 36]]
```

Рисунок 9 – Решение задачи 7 Ч.2

```python
#Сложение матриц
print("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)
print("\n")

#Коммутативность сложения
print("Коммутативность сложения")
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
L = A + B
R = B + A
print(L)
print(R)
print("\n")

#Ассоциативность сложения
print("Ассоциативность сложения")
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)
print("\n")

#Для любой матрицы существует противоположная ей
print("Для любой матрицы существует противоположная ей")
A = np.matrix('1 2; 3 4')
Z = np.matrix('0 0; 0 0')
L = A + (-1) * A
print(L)
print(Z)
print("\n")
```

Рисунок 10 – Решение задачи 8 Ч.1

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


Коммутативность сложения
[[ 6  8]
 [10 12]]
[[ 6  8]
 [10 12]]


Ассоциативность сложения
[[ 7 15]
 [19 15]]
[[ 7 15]
 [19 15]]


Для любой матрицы существует противоположная ей
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
```

Рисунок 11 – Решение задачи 8 Ч.2

```
#Умножение матриц
#Ассоциативность умножения
print("Ассоциативность умножения")
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)
print("\n")

#Дистрибутивность умножения
print("Дистрибутивность умножения")
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)
print(R)
print("\n")

#Умножение матриц в общем виде не коммутативно
print("Умножение матриц в общем виде не коммутативно")
A = np.matrix('1 2; 3 4')
B = np.matrix('5 6; 7 8')
L = A.dot(B)
R = B.dot(A)
print(L)
print(R)
print("\n")

#Произведение заданной матрицы на единичную
print("Произведение заданной матрицы на единичную")
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)
print("\n")

#Произведение заданной матрицы на нулевую матрицу
print("Произведение заданной матрицы на нулевую матрицу")
A = np.matrix('1 2; 3 4')
Z = np.matrix('0 0; 0 0')
L = Z.dot(A)
R = A.dot(Z)
print(L)
print(R)
print(Z)
print("\n")
```

Рисунок 12 – Решение задачи 9 Ч.1

```
Ассоциативность умножения
[[192 252]
 [436 572]]
[[192 252]
 [436 572]]


Дистрибутивность умножения
[[35 42]
 [77 94]]
[[35 42]
 [77 94]]


Умножение матриц в общем виде не коммутативно
[[19 22]
 [43 50]]
[[23 34]
 [31 46]]


Произведение заданной матрицы на единичную
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]


Произведение заданной матрицы на нулевую матрицу
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
[[0 0]
 [0 0]]
```

Рисунок 13 – Решение задачи 9 Ч.2

```
import numpy as np
#Определитель матрицы
print("Определитель матрицы")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
print(np.linalg.det(A))
print("\n")

#Определитель матрицы остается неизменным при ее транспонировании
print("Определитель матрицы остается неизменным при ее транспонировании")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
print(A.T)
det_A = round(np.linalg.det(A), 3)
det_A_t = round(np.linalg.det(A.T), 3)
print(det_A)
print(det_A_t)
print("\n")

#Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю
print("Если у матрицы есть строка или столбец, состоящие из нулей,")
print("то определитель такой матрицы равен нулю")
A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
print(A)
print(np.linalg.det(A))
print("\n")

#При перестановке строк матрицы знак ее определителя меняется на противоположный
print("При перестановке строк матрицы знак ее определителя")
print("меняется на противоположный")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
print(B)
print(np.linalg.det(A))
print(np.linalg.det(B))
print("\n")

#Если у матрицы есть две одинаковые строки, то ее определитель равен нулю
print("Если у матрицы есть две одинаковые строки, то ее определитель равен нулю")
A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
print(A)
print(np.linalg.det(A))
print("\n")

#Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число
print("Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
k = 2
B = A.copy()
B[2, :] = k * B[2, :]
print(B)
det_A = round(np.linalg.det(A), 3)
det_B = round(np.linalg.det(B), 3)
print(det_A * k)
print(det_B)
print("\n")
```

Рисунок 14 – Решение задачи 10 Ч.1

```
#Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен
#сумме определителей двух соответствующих матриц:
print("Если все элементы строки или столбца можно представить как сумму двух слагаемых,то определитель такой матрицы равен")
print("сумме определителей двух соответствующих матриц:")
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)
print(round(np.linalg.det(C), 3))
print(round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3))
print("\n")

#Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель
#матрицы не изменится:
print("Если к элементам одной строки прибавить элементы другой строки,умноженные на одно и тоже число, то определитель")
print("матрицы не изменится:")
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)
print(B)
print(round(np.linalg.det(A), 3))
print(round(np.linalg.det(B), 3))
print("\n")

#Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы
#равен нулю:
print("Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы")
print("равен нулю:")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
K=2
A[1, :] = A[0, :] + k * A[2, :]
print(round(np.linalg.det(A), 3))
print("\n")

#Если матрица содержит пропорциональные строки, то ее определитель равен нулю:
print("Если матрица содержит пропорциональные строки, то ее определитель равен нулю:S")
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
k=2
A[1, :] = k * A[0, :]
print(A)
print(round(np.linalg.det(A), 3))
print("\n")
```

Рисунок 15 – Решение задачи 10 Ч.2

```
Определитель матрицы
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
-14.000000000000009


Определитель матрицы остается неизменным при ее транспонировании
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 10  8]
 [-1  4  3]
 [ 2 -1  1]]
-14.0
-14.0


Если у матрицы есть строка или столбец, состоящие из нулей,
то определитель такой матрицы равен нулю
[[-4 -1  2]
 [ 0  0  0]
 [ 8  3  1]]
0.0


При перестановке строк матрицы знак ее определителя
меняется на противоположный
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[10  4 -1]
 [-4 -1  2]
 [ 8  3  1]]
-14.000000000000009
14.000000000000009


Если у матрицы есть две одинаковые строки, то ее определитель равен нулю
[[-4 -1  2]
 [-4 -1  2]
 [ 8  3  1]]
0.0


Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [10  4 -1]
 [16  6  2]]
-28.0
-28.0
```

Рисунок 16 – Решение задачи 10 Ч.3

```
Если все элементы строки или столбца можно представить как сумму двух слагаемых,то определитель такой матрицы равен
сумме определителей двух соответствующих матриц:
[[-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]]
4.0
4.0


Если к элементам одной строки прибавить элементы другой строки,умноженные на одно и тоже число, то определитель
матрицы не изменится:
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [ 2  2  3]
 [ 8  3  1]]
-14.0
-14.0


Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы
равен нулю:
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
0.0


Если матрица содержит пропорциональные строки, то ее определитель равен нулю:S
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [-8 -2  4]
 [ 8  3  1]]
0.0
```

Рисунок 17 – Решение задачи 10 Ч.4

```python
#Обратная матрица
import numpy as np
#inv()
print("inv()")
A = np.matrix('1 -3; 2 5')
A_inv = np.linalg.inv(A)
print(A_inv)
print("\n")

#Обратная матрица обратной матрицы есть исходная матрица:
print("Обратная матрица обратной матрицы есть исходная матрица:")
A = np.matrix('1. -3.; 2. 5.')
A_inv = np.linalg.inv(A)
A_inv_inv = np.linalg.inv(A_inv)
print(A)
print(A_inv_inv)
print("\n")

#Обратная матрица транспонированной матрицы равна транспонированной
#матрице от обратной матрицы:
print("Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы:")
A = np.matrix('1. -3.; 2. 5.')
L = np.linalg.inv(A.T)
R = (np.linalg.inv(A)).T
print(L)
print(R)
print("\n")

#Обратная матрица произведения матриц равна произведению обратных матриц:
print("Обратная матрица произведения матриц равна произведению обратных матриц:")
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)
print(R)
print("\n")

#Ранг матрицы
#matrix_rank():
print("matrix_rank()")
m_eye = np.eye(4)
print(m_eye)
rank = np.linalg.matrix_rank(m_eye)
print(rank)
print("\n")
m_eye[3][3] = 0
print(m_eye)
rank = np.linalg.matrix_rank(m_eye)
print(rank)
print("\n")
```

Рисунок 18 – Решение задачи 11 Ч.1

```
inv()
[[ 0.45454545  0.27272727]
 [-0.18181818  0.09090909]]


Обратная матрица обратной матрицы есть исходная матрица:
[[ 1. -3.]
 [ 2.  5.]]
[[ 1. -3.]
 [ 2.  5.]]


Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы:
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]


Обратная матрица произведения матриц равна произведению обратных матриц:
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]


matrix_rank()
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
4


[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
3
```

Рисунок 19 – Решение задачи 11 Ч.2

Вывод: были получены навыки по работе с матрицами и векторами с помощью библиотеки NumPy языка программирования Python.