

**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 №2 по дисциплине:
технологии распознавания образов**

Выполнил:

студент группы ПИЖ-б-о-21-1

Прокопов Дмитрий Владиславович

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2023 г.

ВЫПОЛНЕНИЕ

Примеры

```
Ввод [1]: import numpy as np  
m=np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')  
print(m)
```

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

```
Ввод [2]: m[1, 0]
```

```
Out[2]: 5
```

```
Ввод [4]: m[1,:]
```

```
Out[4]: matrix([[5, 6, 7, 8]])
```

```
Ввод [5]: m[:,2]
```

```
Out[5]: matrix([[3],  
               [7],  
               [5]])
```

```
Ввод [6]: m[1, 2 :]
```

```
Out[6]: matrix([[7, 8]])
```

```
Ввод [7]: m[0:2,1]
```

```
Out[7]: matrix([[2],  
               [6]])
```

```
Ввод [8]: m[0:2, 1:3]
```

```
Out[8]: matrix([[2, 3],  
               [6, 7]])
```

```
Ввод [9]: cols=[0,1,3]  
m[:, cols]
```

```
Out[9]: matrix([[1, 2, 4],  
               [5, 6, 8],  
               [9, 1, 7]])
```

```
Ввод [10]: m=np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')  
print(m)
```

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

```
Ввод [11]: type(m)
```

```
Out[11]: numpy.matrix
```

```
Ввод [12]: m=np.array(m)  
type(m)
```

```
Out[12]: numpy.ndarray
```

```
Ввод [13]: m.shape
```

```
Out[13]: (3, 4)
```

```
Ввод [14]: m.max()
```

```
Out[14]: 9
```

```
Ввод [17]: m.max(axis=1)
```

```
Out[17]: matrix([[4],  
                [8],  
                [9]])
```

```
Ввод [18]: m.max(axis=0)
```

```
Out[18]: matrix([[9, 6, 7, 8]])
```

```
Ввод [22]: nums=np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])  
letters=np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])
```

```
Ввод [23]: a=True
```

```
Ввод [24]: b=5>7
```

```
Ввод [25]: print(b)
```

```
False
```

```
Ввод [26]: less_than_5=nums<5
```

```
Ввод [27]: less_than_5
```

```
Out[27]: array([ True,  True,  True,  True, False, False, False, False, False,  
                False])
```

```
Ввод [28]: pos_a=letters=='a'
```

```
Ввод [29]: pos_a
```

```
Out[29]: array([ True, False, False, False,  True, False, False])
```

```
Ввод [30]: nums[less_than_5]
```

```
Out[30]: array([1, 2, 3, 4])
```

```
Ввод [33]: mod_m=np.logical_and(m>=3,m<=7)
```

```
Ввод [34]: mod_m
```

```
Out[34]: matrix([[False, False,  True,  True],  
                [ True,  True,  True, False],  
                [False, False,  True,  True]])
```

```
Ввод [35]: m[mod_m]
```

```
Out[35]: matrix([[3, 4, 5, 6, 7, 5, 7]])
```

```
Ввод [36]: nums=np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
Ввод [37]: nums[nums<5]
```

```
Out[37]: array([1, 2, 3, 4])
```

```
Ввод [38]: nums[nums<5]=10
```

```
Ввод [39]: print(nums)
```

```
[10 10 10 10 5 6 7 8 9 10]
```

```
Ввод [40]: m[m>7]=25
```

```
Ввод [41]: print(m)
```

```
[[ 1  2  3  4]  
 [ 5  6  7 25]  
 [25  1  5  7]]
```

```
Ввод [42]: np.arange(10)
Out[42]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Ввод [43]: np.arange(5,12)
Out[43]: array([ 5, 6, 7, 8, 9, 10, 11])

Ввод [44]: np.arange(1,5,0.5)
Out[44]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

```
Ввод [45]: a=[[1,2],[3,4]]
```

```
Ввод [46]: np.matrix(a)
Out[46]: matrix([[1, 2],
                 [3, 4]])
```

```
Ввод [47]: b=np.array([[5,6],[7,8]])
```

```
Ввод [48]: np.matrix(b)
Out[48]: matrix([[5, 6],
                 [7, 8]])
```

```
Ввод [49]: np.matrix('[1,2;3,4]')
Out[49]: matrix([[1, 2],
                 [3, 4]])
```

```
Ввод [50]: np.zeros((3,4))
Out[50]: array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.]])
```

```
Ввод [51]: np.eye(3)
Out[51]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

```
Ввод [52]: A=np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```
Ввод [53]: A
Out[53]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

```
Ввод [54]: np.ravel(A)
Out[54]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
Ввод [55]: np.ravel(A, order='C')
Out[55]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
Ввод [56]: np.ravel(A, order='F')
Out[56]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

```

Ввод [57]: a=np.array([0,1,2,3,4,5,6,7,8,9])

Ввод [58]: np.where(a%2==0,a*10,a/10)
Out[58]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

Ввод [59]: a=np.random.rand(10)

Ввод [60]: a
Out[60]: array([0.84084703, 0.60204262, 0.85939559, 0.16051418, 0.22293724,
 0.47226327, 0.09990886, 0.67620995, 0.21949436, 0.0256525 ])

Ввод [61]: np.where(a>0.5, True, False)
Out[61]: array([ True,  True,  True, False, False, False, False,  True, False,
  False])

Ввод [62]: np.where(a>0.5,1,-1)
Out[62]: array([ 1,  1,  1, -1, -1, -1, -1,  1, -1, -1])

Ввод [8]: a=np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Ввод [9]: np.where(a%2==0, a*10, a/10)
Out[9]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

Ввод [10]: a=np.random.rand(10)

Ввод [11]: a
Out[11]: array([0.74045847, 0.38317699, 0.54463184, 0.00734538, 0.57716995,
 0.0384941 , 0.78457164, 0.67723515, 0.23376135, 0.36237431])

Ввод [12]: np.where(a>0.5, True, False)
Out[12]: array([ True, False,  True, False,  True, False,  True,  True, False,
  False])

Ввод [13]: np.where(a>0.5, 1, -1)
Out[13]: array([ 1, -1,  1, -1,  1, -1,  1,  1, -1, -1])

```

8 задание

Тестирование работы нейросети считывания цифр на Python на основе набора MNIST

Написанная нейросеть считывает набор тренировочных данных из 100 элементов MNIST. На основе тренировочного набора производится считывание 10 символов и сравнивается с результатом работы нейросети

Обновление весов высчитывается по формуле $\Delta W = a * E * \text{Сигмоида}(O)(1 - \text{Сигмоида}(O)) * O^T$

Описание

Нейросеть состоит из 784 входных узлов (по одному на каждый пиксель), 100 скрытых узлов и 10 выходных узлов (по одному на цифру).

Код:

```

import numpy
import scipy
import matplotlib.pyplot

class NeuralNetwork:
    def __init__(self, inputNodes, hiddenNodes, outputNodes, learningGrate):
        # Задать количество узлов в слоях
        self.inodes = inputNodes
        self.hnodes = hiddenNodes
        self.onodes = outputNodes

        # Коэффициент обучения
        self.lr = learningGrate

        # Матрицы весов
        self.wih = (numpy.random.rand(self.hnodes, self.inodes) - 0.5)
        self.who = (numpy.random.rand(self.onodes, self.hnodes) - 0.5)

        # Сигмоида
        self.activation_function = lambda x: scipy.special.expit(x)

    pass

    def train(self, inputs_list, targets_list):
        # преобразовать список входных значений в двумерный массив
        inputs = numpy.array(inputs_list, ndmin=2).T
        targets = numpy.array(targets_list, ndmin=2).T

        # рассчитать входящие сигналы для скрытого слоя
        hidden_inputs = numpy.dot(self.wih, inputs)
        # рассчитать исходящие сигналы для скрытого слоя
        hidden_outputs = self.activation_function(hidden_inputs)

```

```

# рассчитать входящие сигналы для скрытого слоя
hidden_inputs = numpy.dot(self.wih, inputs)
# рассчитать исходящие сигналы для скрытого слоя
hidden_outputs = self.activation_function(hidden_inputs)

# рассчитать входящие сигналы для выходного слоя
final_inputs = numpy.dot(self.who, hidden_outputs)
# рассчитать исходящие сигналы для выходного слоя
final_outputs = self.activation_function(final_inputs)

# Поиск ошибки
output_errors = targets - final_outputs
hidden_errors = numpy.dot(self.who.T, output_errors)
self.who += self.lr * numpy.dot((output_errors * final_outputs *
                                (1.0 - final_outputs)), numpy.transpose(hidden_outputs))

self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs *
                                (1.0 - hidden_outputs)), numpy.transpose(inputs))

pass

def query(self, inputs_list):
    # Преобразовать входные значения в массив
    inputs = numpy.array(inputs_list, ndmin=2).T

    # Рассчитать входящие сигналы для скрытого слоя
    hidden_inputs = numpy.dot(self.wih, inputs)
    # Рассчитать исходящие сигналы для скрытого слоя
    hidden_outputs = self.activation_function(hidden_inputs)

    # Рассчитать входящие сигналы для выходного слоя
    final_inputs = numpy.dot(self.who, hidden_outputs)
    # Рассчитать исходящие сигналы для выходного слоя
    final_outputs = self.activation_function(final_inputs)

    return final_outputs

```

Укажем количество узлов

```

input_nodes = 784
hidden_nodes = 100
output_nodes = 10
learning_rate = 0.3

n = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_rate)

```

Существует коллекция изображений рукописных цифр, используемых исследователями искусственного интеллекта в качестве популярного набора для тестирования идей и алгоритмов.

Этим тестовым набором является база данных рукописных цифр под названием "MNIST"

1. Первое значение — это маркер, т.е. фактическая цифра, например "7" или "9" которую должен представлять данный рукописный экземпляр. Это ответ, правильному получению которого должна обучиться нейронная сеть.
2. Последующие значения, разделенные запятыми, — это значения пикселей рукописной цифры. Пиксельный массив имеет размерность 28x28, поэтому за каждым маркером следуют 784 пикселя.

Считываем 10 тестовых значений MNIST

```

test_data_file = open("mnist_test_10.csv", 'r')
data_list = test_data_file.readlines()
test_data_file.close()

```

Первый символ в каждой строке указывает какой символ зашифрован

```

all_values = data_list[1].split(',')
print(all_values[0])

```

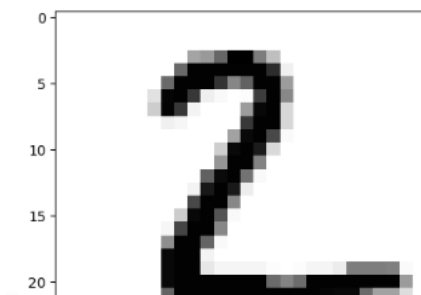
2

Выведем весь набор значений, чтобы посмотреть как выглядит цифра в программе

```

image_array = numpy.asfarray(all_values[1:]).reshape((28,28))
matplotlib.pyplot.imshow(image_array, cmap='Greys', interpolation='None')
<matplotlib.image.AxesImage at 0x27071937c70>

```



Обучим нейросеть на основе 100 элементов набора MNIST

```

training_data_file = open("mnist_train_100.csv", 'r')
training_data_list = training_data_file.readlines()
training_data_file.close()

for record in training_data_list:
    all_values = record.split(',')
    inputs = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
    targets = numpy.zeros(output_nodes) + 0.01
    targets[int(all_values[0])] = 0.99
    n.train(inputs, targets)
    pass

```

мы создаем пустой список scorescard, который будет служить нам журналом оценок работы сети, обновляемым после обработки каждой записи.

Данная переменная будет служить счетчиком правильных значений

```

scorecard = []

for record in data_list:
    # получить список значений из записи, используя символы
    # запятой (',') в качестве разделителей
    all_values = record.split(',')
    # правильный ответ - первое значение
    correct_label = int(all_values[0])
    print(correct_label, "истинный маркер")
    # масштабировать и сдвинуть входные значения
    inputs = (numpy.asarray(all_values[1:]) / 255.0 * 0.99) + 0.01
    # опрос сети
    outputs = n.query(inputs)
    # индекс наибольшего значения является маркерным значением
    label = numpy.argmax(outputs)
    print(label, "ответ сети")
    # присоединить оценку ответа сети к концу списка
    if (label == correct_label):
        # в случае правильного ответа сети присоединить
        # к списку значение 1
        scorecard.append(1)
    else:
        # в случае неправильного ответа сети присоединить
        # к списку значение 0
        scorecard.append(0)
    pass
pass

```

```

7 истинный маркер
7 ответ сети
2 истинный маркер
3 ответ сети
1 истинный маркер
1 ответ сети
0 истинный маркер
0 ответ сети
4 истинный маркер
4 ответ сети
1 истинный маркер
1 ответ сети
4 истинный маркер
4 ответ сети
9 истинный маркер
3 ответ сети
5 истинный маркер
4 ответ сети
9 истинный маркер
7 ответ сети

```

scorecard

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

Таким образом видно что нейросеть на основе 100 тренировочных данных смогла определить правильно 6 из 10 символов, что составляет 60%

Индивидуальная работа

3. Дана целочисленная прямоугольная матрица. Определить:
 количество столбцов, содержащих хотя бы один нулевой элемент;
 номер строки, в которой находится самая длинная серия одинаковых элементов.

```
import numpy as np
```

```
m=np.array([[3, 2, 1, 4, 5, 0, 5, 1], [5, 4, 3, 0, 0, 4, 3, 2],
            [5, 5, 5, 5, 0, 3, 1, 1], [3, 4, 5, 5, 0, 1, 4, 5], [0, 0, 2, 3, 5, 8, 7, 2]])
m
```

```
array([[3, 2, 1, 4, 5, 0, 5, 1],
       [5, 4, 3, 0, 0, 4, 3, 2],
       [5, 5, 5, 5, 0, 3, 1, 1],
       [3, 4, 5, 5, 0, 1, 4, 5],
       [0, 0, 2, 3, 5, 8, 7, 2]])
```

```
m[m!=0]=1
m
```

```
array([[1, 1, 1, 1, 1, 0, 1, 1],
       [1, 1, 1, 0, 0, 1, 1, 1],
       [1, 1, 1, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 0, 1, 1, 1],
       [0, 0, 1, 1, 1, 1, 1, 1]])
```

```
ans =8 - m.prod(axis = 0).sum()
ans
```

```
5
```

```
matrix = np.array([
    [1, 2, 1, 4, 4, 4, 4, 4],
    [1, 1, 1, 1, 1, 2, 2, 2],
    [1, 4, 2, 2, 2, 2, 2, 2]
])
```

```
np.argmax([np.max(np.diff(np.where(np.concatenate((row[0:], row[:-1] != row[1:],
[True])))[0]))) + 1 for row in matrix])
```

```
2
```

Контрольные вопросы:

- 1) Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB
- 2) Narray - это объект n-мерного массива, определенный в numpy, который хранит коллекцию элементов одинакового типа. Другими словами, мы можем определить ndarray как коллекцию объектов типа данных (dtype). Доступ к объекту ndarray можно получить с помощью индексации, основанной на 0.
- 3) N-мерный (многомерный) массив имеет фиксированный размер и содержит элементы одного типа. к содержимому многомерного массива можно получить доступ и изменить, используя индексацию и нарезку массива по желанию.
- 4) Простым примером расчет статистик является функция max(). Если необходимо найти максимальный элемент в каждой строке, то для этого нужно передать в качестве аргумента параметр axis=1, а для статистики по столбцам, передайте в качестве параметра аргумент axis=0
- 5) При помощи np.where