Практическая работа 4.

Введение в проектирование нейронных сетей с помощью Python

**ФИО: Пестов Андрей Витальевич**

**Группа: R4136c (КСИТ\_1.1)**

**ЗАДАЧА**

Одна из задач киберфизических систем - распознавание изображений с помощью машинного зрения. В этой лабораторной работе мы рассмотрим решение проблемы такого типа.

Мы будем использовать два файла данных с массивом рукописных чисел: mnist\_train.csv и mnist\_test.csv. Первый файл необходимо использовать для обучения нейронной сети, второй файл необходимо использовать для проверки работы нейронной сети. Каждый файл содержит строки (60000 в первом файле и 10000 во втором файле), в каждой из которых хранится массив размером 28x28 пикселей с цифровым изображением и номером, который соответствует изображению. Пример данных показан на следующем рисунке:

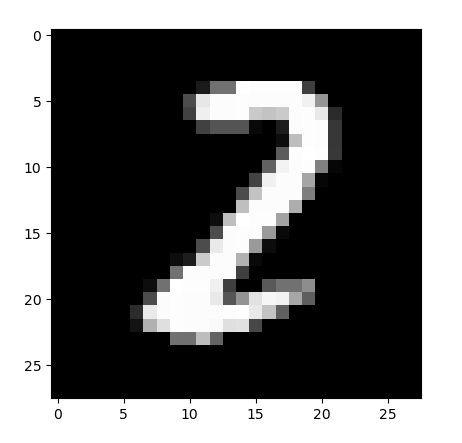


Рисунок 1.

Основная задача - спроектировать нейронную сеть на Python, способную распознавать изображения чисел. Необходимо:

1. Импортировать библиотеки в Python.

2. Написать функцию для установки основных сетевых параметров.

3. Создать функцию, которая устанавливает начальные значения весов нейронной сети.

4. Создать функцию, которая вычисляет выходной сигнал нейронной сети.

5. Создать функцию для обучения нейронной сети.

6. Написать функцию для обучения сети на реальных данных.

7. Написать функцию проверки сети.

8. Написать функцию, отображающую изображения чисел из набора данных.

9. Обучите сеть и рассчитайте ее эффективность.

**1. Импортируйте библиотеки.**

import numpy as npy

import matplotlib.pyplot as mpl

from scipy.special import expit as f\_a

**2. Напишите функцию для установки основных сетевых параметров.**

def init\_net():

input\_nodes = 784

print('Input the number of hidden neurons:')

hidden\_nodes = int(input())

out\_nodes = 10

print('Input the training speed (0.5):')

learn\_speed = float(input())

return input\_nodes, hidden\_nodes, out\_nodes, learn\_speed

**3. Создайте функцию, которая устанавливает начальные значения весов нейронной сети.**

def create\_net(input\_nodes, hidden\_nodes, out\_nodes):

w\_in2hidden = npy.random.uniform(-0.5, 0.5, (hidden\_nodes, input\_nodes))

w\_hidden2out = npy.random.uniform(-0.5, 0.5, (out\_nodes, hidden\_nodes))

return w\_in2hidden, w\_hidden2out

**4. Создайте функцию, которая вычисляет выходной сигнал нейронной сети.**

def net\_output(w\_in2hidden, w\_hidden2out, input\_signal, return\_hidden):

input = npy.array(input\_signal, ndmin=2).T

hidden\_in = npy.dot(w\_in2hidden, input)

hidden\_out = f\_a(hidden\_in)

final\_in = npy.dot(w\_hidden2out, hidden\_out)

final\_out = f\_a(final\_in)

if return\_hidden == 0:

return final\_out

else:

return final\_out, hidden\_out

**5. Создайте функцию для обучения нейронной сети.**

def net\_train(target\_list, input\_signal, w\_in2hidden,

w\_hidden2out, learn\_speed):

targets = npy.array(target\_list, ndmin=2).T

inputs = npy.array(input\_signal, ndmin=2).T

final\_out, hidden\_out = net\_output(

w\_in2hidden, w\_hidden2out, input\_signal, 1)

out\_errors = targets-final\_out

hidden\_errors = npy.dot(w\_hidden2out.T, out\_errors)

w\_hidden2out += learn\_speed \* \

npy.dot((out\_errors\*final\_out\*(1 - final\_out)), hidden\_out.T)

w\_in2hidden += learn\_speed \* \

npy.dot((hidden\_errors\*hidden\_out\*(1 - hidden\_out)), inputs.T)

**6. Напишите функцию для обучения сети на реальных данных.**

def train\_set(w\_in2hidden, w\_hidden2out, learn\_speed):

data\_file = open("mnist\_train.csv", 'r')

training\_list = data\_file.readlines()

data\_file.close()

for record in training\_list:

all\_values = record.split(',')

# range of input data is scaled from [0.0,255] to [0.001,1.0]

inputs = (npy.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001

targets = npy.zeros(10)+0.001

# digits 0-9

targets[int(all\_values[0])] = 1.0

net\_train(targets, inputs, w\_in2hidden, w\_hidden2out, learn\_speed)

return w\_in2hidden, w\_hidden2out

**7. Напишите функцию проверки сети.**

def test\_set(w\_in2hidden, w\_hidden2out):

data\_file = open("mnist\_test.csv", 'r')

test\_list = data\_file.readlines()

data\_file.close()

test = []

for record in test\_list:

all\_values = record.split(',')

# range of input data is scaled from [0.0,255] to [0.001,1.0]

inputs = (npy.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001

out\_session = net\_output(w\_in2hidden, w\_hidden2out, inputs, 0)

if int(all\_values[0]) == npy.argmax(out\_session):

test.append(1)

else:

test.append(0)

test = npy.asarray(test)

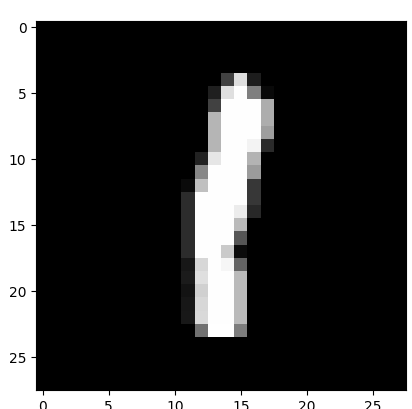
print('Net efficiency % =', (test.sum()/test.size)\*100)

**8. Напишите функцию, отображающую изображения чисел из набора данных.**

def plot\_image(pixels: npy.array):

mpl.imshow(pixels.reshape((28, 28)), cmap='gray')

mpl.show()



*Рисунок 1. Пример изображения, хранящегося в наборе данных.*

**9. Обучение сети и расчет ее эффективности.**

input\_nodes, hidden\_nodes, out\_nodes, learn\_speed = init\_net()

w\_in2hidden, w\_hidden2out = create\_net(input\_nodes, hidden\_nodes, out\_nodes)

My\_Variant = 40

for i in range(5):

print('Test#', i+1)

train\_set(w\_in2hidden, w\_hidden2out, learn\_speed)

test\_set(w\_in2hidden, w\_hidden2out)

data\_file = open("lab2/mnist\_test.csv", 'r')

test\_list = data\_file.readlines()

data\_file.close()

all\_values = test\_list[int(My\_Variant-1)].split(',')

inputs = (npy.asfarray(all\_values[1:]) / 255.0 \* 0.999) + 0.001

out\_session = net\_output(w\_in2hidden, w\_hidden2out, inputs, 0)

print(npy.argmax(out\_session))

plot\_image(npy.asfarray(all\_values[1:]))

Найдите номер, соответствующий n-му элементу набора данных, где n - ваш вариант.

Ответ: 8