МИНЕСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Брестский государственный технический университет»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №5

По дисциплине «Интеллектуальные методы обработки данных»

За 7 семестр

Тема: «Нелинейные искусственные нейронные сети в задачах распознавания образов»

Выполнила:  
студентка 4 курса  
группы АС-56  
Карпенко М.В.

Проверил:

Савицкий Ю.В.

Брест 2022

*Цель работы:* Изучить обучение и функционирование нелинейной искусственной нейронной сети (ИНС) при решении задач распознавания образов.

*Задание:* Написать на любом языке высокого уровня программу моделирования нелинейной ИНС для распознавания образов. Рекомендуется использовать сигмоидную функцию (но это не является обязательным). Количество нейронных сетей в скрытом слое взять согласно варианту работы №4. Его можно варьировать, если сеть не обучается или некорректно функционирует. Для организации обучающей выборки использовать данные таблиц 5.1 и 5.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | *a* | *b* | *с* | *d* | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |
| 9 | 0.1 | 0.3 | 0.08 | 0.3 | 10 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Вектор 1 | Вектор 2 | Вектор 3 |
| 9 | 2 | 4 | 8 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Данные вектора | | | | | | | | | | | | | | | | | | | |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

*Текст программы:*

import random

import math

import matplotlib.pyplot as plt

STEP = 0.5

MIN\_ERROR = 1e-4

input\_neuron\_number = 10

hidden\_neuron\_number = 4

output\_neuron\_number = 3

Wij = [[random.uniform(-0.1, 0.1) for \_ in range(hidden\_neuron\_number)] for \_ in range(input\_neuron\_number - 1)]

Wjk = [[random.uniform(-0.1, 0.1) for \_ in range(hidden\_neuron\_number)] for \_ in range(output\_neuron\_number)]

Tj = [random.uniform(-0.5, 0.5) for \_ in range(hidden\_neuron\_number)]

Tk = [random.uniform(-0.5, 0.5) for \_ in range(output\_neuron\_number)]

vector\_2 = [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

vector\_4 = [1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0]

vector\_8 = [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1]

all\_vectors = [vector\_2, vector\_4, vector\_8]

def sigmoidFunction(S):

return 1 / (1 + math.exp(-S))

def Sj\_onHiddenLayer(y):

Sj = []

for j in range(hidden\_neuron\_number):

value = 0

for i in range(input\_neuron\_number - 1):

value += y[i] \* Wij[i][j]

value -= Tj[j]

Sj.append(sigmoidFunction(value))

return Sj

def Sk\_onOutputLayer(Yj):

Sk = []

for j in range(output\_neuron\_number):

value = 0

for i in range(hidden\_neuron\_number):

value += Yj[i] \* Wjk[j][i]

value -= Tk[j]

Sk.append(sigmoidFunction(value))

return Sk

def Wjk\_change(Yj, Yk, error):

global Tk

for j in range(output\_neuron\_number):

for i in range(hidden\_neuron\_number):

Wjk[j][i] -= STEP \* error[j] \* Yk[j] \* (1 - Yk[j]) \* Yj[i]

Tk[j] += error[j] \* STEP \* Yk[j] \* (1 - Yk[j])

def Wij\_change(Yj, hiddenLayer\_error, y):

for j in range(hidden\_neuron\_number):

for i in range(input\_neuron\_number - 1):

Wij[i][j] -= STEP \* hiddenLayer\_error[j] \* y[i] \* Yj[j] \* (1 - Yj[j])

Tj[j] += STEP \* hiddenLayer\_error[j] \* Yj[j] \* (1 - Yj[j])

def main():

drawing\_data = ([], [])

errors = [0] \* output\_neuron\_number

reference = [0] \* output\_neuron\_number

hiddenLayer\_error = [0] \* hidden\_neuron\_number

iteration = 1

epoch = 0

error = 1

while error > MIN\_ERROR:

error = 0

for N in range(output\_neuron\_number):

reference[N] = 1

for i in range(iteration):

y = all\_vectors[N]

Yj = Sj\_onHiddenLayer(y)

Yk = Sk\_onOutputLayer(Yj)

for index in range(output\_neuron\_number):

errors[index] = Yk[index] - reference[index]

for j in range(hidden\_neuron\_number):

for k in range(output\_neuron\_number):

hiddenLayer\_error[j] += errors[k] \* Yk[k] \* (1 - Yk[k]) \* Wjk[k][j]

Wjk\_change(Yj, Yk, errors)

Wij\_change(Yj, hiddenLayer\_error, y)

error += errors[N] \*\* 2

error /= 2

drawing\_data[0].append(epoch)

drawing\_data[1].append(error)

epoch += 1

plt.plot(\*drawing\_data)

plt.xlabel("Epoch")

plt.ylabel("Error")

plt.show()

for i in range(len(all\_vectors)):

input = all\_vectors[i]

print("Result vector :", i + 1, end=" : ")

for j in range(len(vector\_2)):

print(input[j], end='')

print("\nResult : ", end='')

hiddenLayer\_prev = Sj\_onHiddenLayer(input)

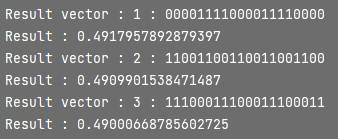
Values = Sk\_onOutputLayer(hiddenLayer\_prev)

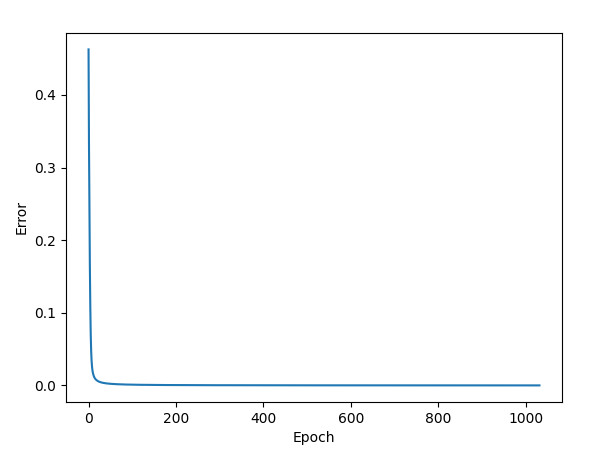
print(Values[0])

if \_\_name\_\_ == '\_\_main\_\_':

main()

*Результаты работы программы:*





*Выводы по лабораторной работы:* Изучила обучение и функционирование нелинейной искусственной нейронной сети (ИНС) при решении задач распознавания образов.