Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине: “Методы и алгоритмы принятия решений”

## Тема: «Нелинейные ИНС в задачах распознавания образов»

## Вариант №10

**Выполнил**:

студент 2 курса

группы ПО-7

Курмыса Е.Е.

**Проверил:**

Крощенко А.А.

Брест 2021

**Цель работы:** Изучить обучение и функционирование нелинейной ИНС при решении задач распознавания образов.

**Постановка задачи:** Написать на любом ЯВУ программу моделирования нелинейной ИНС для распознавания образов. Рекомендуется использовать сигмоидную функцию, но это не является обязательным. Количество НЭ в скрытом слое взять согласно варианту работы №3. Его можно варьировать, если сеть не обучается или некорректно функционирует. Провести исследование полученной модели. При этом на вход сети необходимо подавать искаженные образы, в которых инвертированы некоторые биты. Критерий эффективности процесса распознавания - максимальное кодовое расстояние (количество искаженных битов) между исходным и поданным образом.

**Код**

from random import uniform, randint

from math import exp

vector\_1 = [1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]

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

vector\_3 = [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1]

vectors = [vector\_1, vector\_2, vector\_3]

omega = [[[uniform(0, 1) for \_ in range(6)] for \_ in range(2)], [uniform(0,1) for \_ in range(2)]] # коэффициенты

T = [0 for \_ in range(3)] # пороги обучения для двух скрытых нейронов и одного выходного

count\_of\_epochs = 1 # порядковый номер эпох из 3 итераций

last\_err = 0 # предыдущая погрешность относительно текущей

Err\_min = float(input('Введите максимально возможную квадратичную погрешность при обучении: '))

Err\_sum = 1 # текущая ошибка

print('Даны следующие вектора:')

print('Вектор 1:', vector\_1, '\nВектор 2:', vector\_2, '\nВектор 3:', vector\_3)

while Err\_sum >= Err\_min:

Err\_sum = 0 # для высчитывания тестовой среднеквадратической ошибки

for vector\_number in range(3):

sum\_sq\_y = 0

x = vectors[vector\_number] # для обучения, проверять на практике будем другие значения

y\_prom\_test, S\_prom\_test, gamma\_prom\_test = [0, 0], [0, 0], [0, 0]

# выходные значения, S и погрешности для промежуточного слоя (тестовые)

t = 1 # идеальное выходное значение

for j in range(2):

for k in range(6):

S\_prom\_test[j] += omega[0][j][k] \* x[k]

S\_prom\_test[j] -= T[j]

y\_prom\_test[j] = 1 / (1 + exp(-S\_prom\_test[j]))

sum\_sq\_y += y\_prom\_test[j]\*\*2

S = 0 # функция активации для выходного нейрона (линейная :3)

for j in range(2):

S += omega[1][j] \* y\_prom\_test[j]

y = S - T[2] # предполагаемое выходное значение

delta = y - t # погрешность между идеальным и полученным значениями

err = delta\*\*2 / 2

Err\_sum += err

for j in range(2):

gamma\_prom\_test[j] = delta \* y \* (1 - y) \* omega[1][j]

S1, S2, S3 = 0, 1, 0

for j in range(2):

S1 += 4 \* gamma\_prom\_test[j]\*\*2 \* y\_prom\_test[j] \* (1 - y\_prom\_test[j])

S2 += y\*\*2

S3 += gamma\_prom\_test[j]\*\*2 \* y\_prom\_test[j]\*\*2 \* (1 - y\_prom\_test[j])\*\*2

if S3 == 0:

break

alpha\_1 = S1 / S2 / S3

alpha\_2 = 1 / (1 + sum\_sq\_y)

for j in range(2):

for k in range(6):

omega[0][j][k] -= alpha\_1 \* gamma\_prom\_test[j] \* y\_prom\_test[j] \* (1 - y\_prom\_test[j]) \* x[k]

T[j] += alpha\_1 \* gamma\_prom\_test[j] \* y\_prom\_test[j] \* (1 - y\_prom\_test[j])

omega[1][j] -= alpha\_2 \* delta \* y\_prom\_test[j]

T[2] += alpha\_2 \* delta

avg\_Err = Err\_sum / 3

if avg\_Err != last\_err:

if avg\_Err < Err\_min:

break

last\_err = avg\_Err

count\_of\_epochs += 1

print('\nПосле обучения на', count\_of\_epochs, 'эпохах имеем следующие конфигурации:\nКоэффициенты -', omega, '\nПороги функции T -', T)

print('\nПроводим тестирование/предсказание для вектора для кодового расстояния от 0 до 20 (каждое - по 1000 раз)\n')

error\_sum = 0 # для высчитывания итоговой среднеквадратической ошибки

vector\_difference = 0 # кодовое расстояние, для кторого будет проверяться НС

while vector\_difference <= 20:

test\_number = 1 # номер теста

error\_mid\_sum = 0 # для вычисления среднеквадратической ошибки при некотором расстоянии

while test\_number <= 1000:

x = vectors[randint(0, 2)] # значения x для генерации ответа

indexes\_to\_change = [] # значения индексов, которые нужно инвертировать

while len(indexes\_to\_change) < vector\_difference:

cand\_index = randint(0, 19)

if cand\_index not in indexes\_to\_change:

indexes\_to\_change.append(cand\_index)

for index in indexes\_to\_change:

x[index] = (x[index] + 1) % 2 # инвертация

y\_prom, S\_prom, gamma\_prom = [0, 0], [0, 0], [0, 0]

# выходные значения, S и погрешности для промежуточного слоя (при прогнозировании)

ideal = 1 # идеальное выходное значение

for j in range(2):

for k in range(6):

S\_prom[j] += omega[0][j][k] \* x[k]

S\_prom[j] -= T[j]

y\_prom[j] = 1 / (1 + exp(-S\_prom[j]))

S = 0 # сумматор произведений

for j in range(2):

S += omega[1][j] \* y\_prom[j]

y = S - T[2] # предполагаемое выходное значение

delta = y - ideal

err = delta\*\*2 / 2

error\_mid\_sum += err

test\_number += 1

avg\_mid\_error = error\_mid\_sum / 15

print('Среднеквадратическая погрешность для кодового расстояния', vector\_difference, 'равна', avg\_mid\_error)

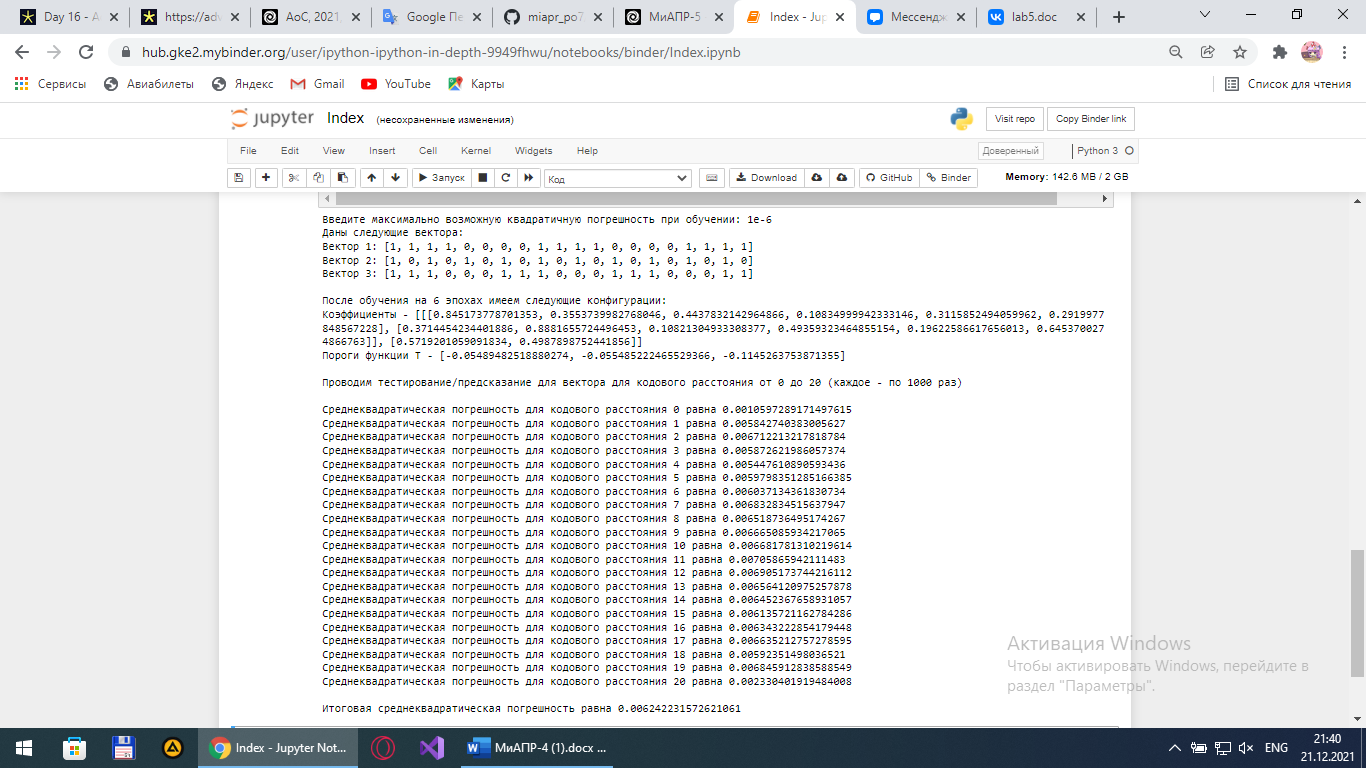
error\_sum += avg\_mid\_error

vector\_difference += 1

avg\_error = error\_sum / 20 # среднеквадратическая погрешность

print('\nИтоговая среднеквадратическая погрешность равна', avg\_error)

**Вывод**



В целом можно сказать, что после обучения нейронная сеть способна достаточно хорошо распознавать произвольные вектора с некоторыми кодовыми расстояниями. Более того, показания погрешностей становятся более постоянными, если мы будем проверять различные вектора не 15 раз, а большее число раз (здесь – 1000) — в таком случае можно выделить несколько особенностей:

1. Если расстояние равно 20, то погрешность для него приблизительно равна погрешности для расстояния 0 (поскольку, по идее, на вход подаются вектора с противоположными значениями);
2. Погрешности с остальными расстояниями примерно являются одинаковыми и составляют общее представление о том, какова погрешность у НС.

**Вывод:** я изучил обучение и функционирование нелинейной ИНС при решении задач прогнозирования.