Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине: “Методы и алгоритмы принятия решений”

## Тема: «Нелинейные ИНС в задачах прогнозирования»

## 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 прогнозирования.

**Постановка задачи:** спрогнозировать нелинейный временной ряд,  применяя параметры лабораторной работы №3. При этом необходимо использовать алгоритм обучения многослойной ИНС с адаптивным шагом.

**Код**

from random import uniform

from math import sin, cos, exp

def f(x):

return 0.2 \* cos(0.4 \* x) + 0.09 \* sin(0.4 \* x)

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\_iterations = 1 # порядковый номер итерации

last\_err = 0 # предыдущая погрешность относительно текущей

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

print('Изначальные данные:\nИзначальные коэфф. -', omega, '\nПороги функций T -', T)

# обучение на тридцати итерациях, после которых мы получаем коэффициенты и T для непосредственного прогноза

print(f'Обучение:\nЭпоха # N: {"Идеальное значение t":21} {"Полученное значение":21} {"Отклонение между знач.":26} {"Ср. квадр. погр.":26}')

while True:

Err\_sum = 0 # для высчитывания тестовой среднеквадратической ошибки

for i in range(30):

sum\_sq\_y = 0

x = [f(j / 10) for j in range(i, i + 6)] # для обучения, проверять на практике будем другие значения

X = (i + 6) / 10 # аргумент функции для проверки погрешности

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 = f(X) # идеальное значение f(X) напрямую

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

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 / 30

if avg\_Err != last\_err:

if count\_of\_iterations % 500 == 0:

print(f'Эпоха # {count\_of\_iterations:4}: {t:21} {y:21} {err:24} {avg\_Err:24}')

if avg\_Err < Err\_min:

break

last\_err = avg\_Err

count\_of\_iterations += 1

print('\nПосле обучения на', count\_of\_iterations, 'итерациях(и, й) имеем следующие конфигурации:\nКоэффициенты -', omega, '\nПороги функции T -', T)

print('\nПроводим тестирование/предсказание значения функции на 15-ти случайных рациональных значениях от -100 до 100:')

test\_number = 1 # номер теста

error\_sum = 0 # для высчитывания итоговой среднеквадратической ошибки

print(f'Тестирование:\nЭпоха # N: {"Идеальное значение t":21} {"Полученное значение":21} {"Отклонение между знач.":26} {"Квадр. погр.":26}')

while test\_number <= 15:

X = uniform(-100, 100) # аргумент для тестирования

x = [f(X - (i / 10)) for i in range(6, 0, -1)] # значения x для генерации ответа

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

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

ideal = f(X) # идеальное значение функции

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\_sum += err

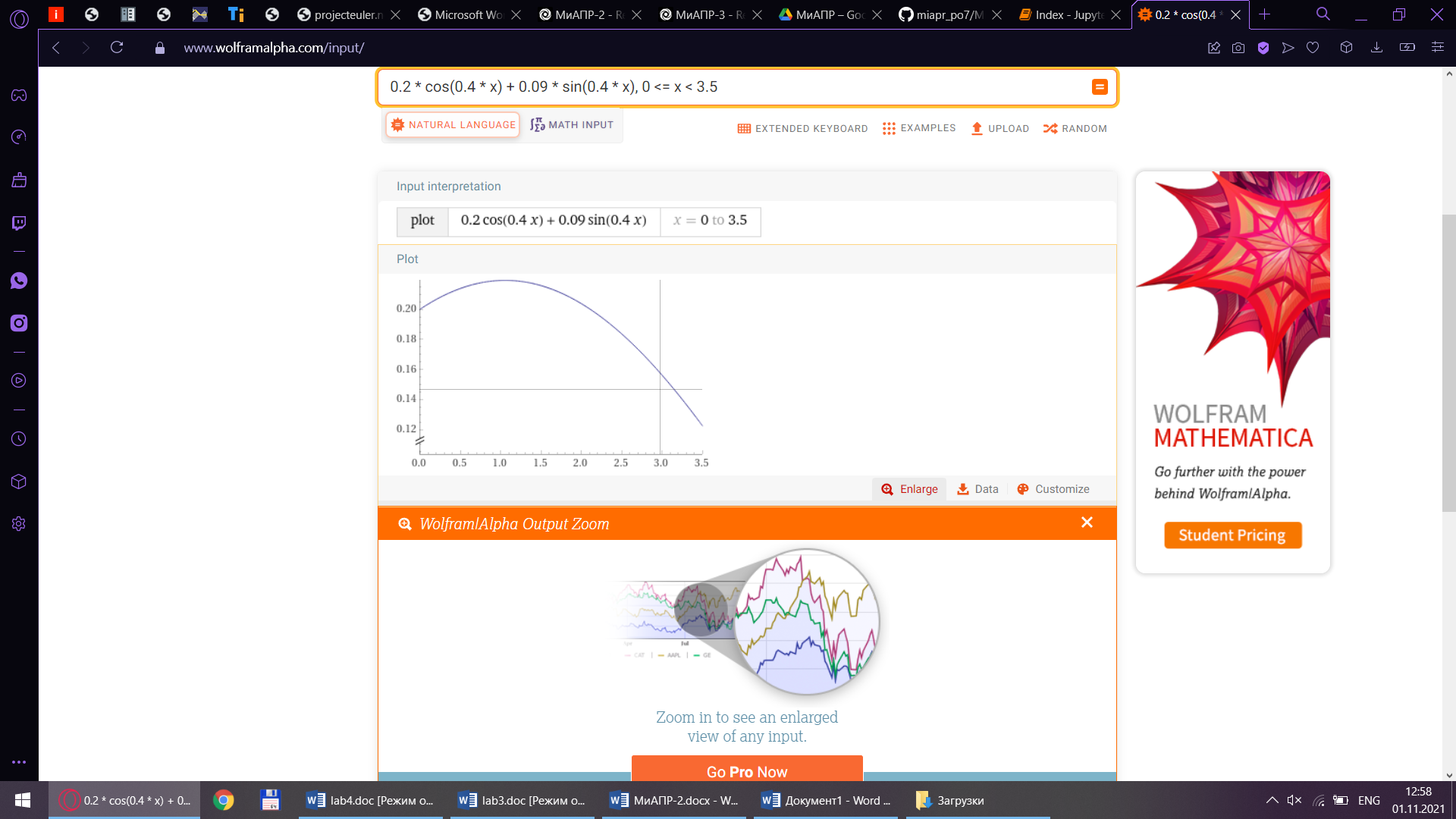
print(f'Эпоха #{test\_number:2}: {ideal:21} {y:21} {delta:24} {err:24}')

test\_number += 1

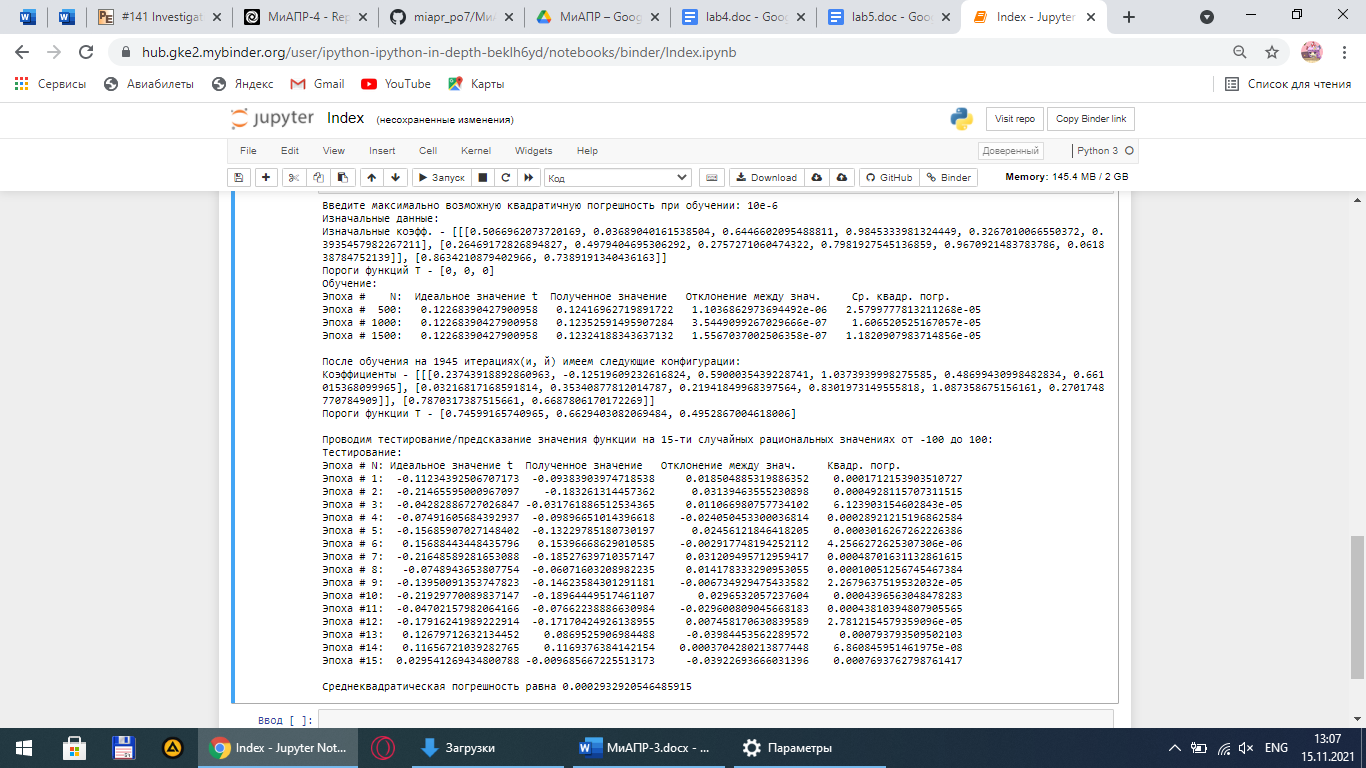
avg\_error = error\_sum / 15 # среднеквадратическая погрешность

print('\nСреднеквадратическая погрешность равна', avg\_error)

**График функции на промежутке [0; 3.5)**



**Вывод**



Если сравнивать полученные результаты с результатами предыдущей лабораторной работы, то мы видим, что программа работает постоянно за несколько тысяч итераций, в то время как ранее она могла работать более 100 тысяч раз. Это значительно ускоряет работу программы и позволяет получать примерно такую же погрешность за куда меньший промежуток времени.

Недостатком данного метода является то, что при локальных минимумах НС «застревает» в них и не изменяет погрешность, из-за чего приходится аварийно завершать работу программы. В таком случае можно лишь пытаться учитывать такие случаи посредством более приспособленных алгоритмов, а также надеяться, что текущие рандомные иниц. значения переменных будут способствовать получению требуемой погрешности.

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