Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах прогнозирования. Адаптивный шаг обучения»

Вариант №4

Выполнил:

Студент 2 курса

Группы ПО-7

Дмитрук М.А.

Проверил:

Крощенко А.А.

2021

**Цель работы:** изучить обучение и функционирование линейной ИНС при решении

задач прогнозирования с использованием адаптивного шага.

**Задание:** Написать на любом ЯВУ программу моделирования прогнозирующей нелинейной ИНС. Для тестирования использовать функцию

y = a\*cos(b\*x) + c\*sin(d\*x)

a=0.4, b=0.4, c=0.08, d=0.4, кол-во входов ИНС = 6, кол-во НЭ в скрытом слое = 2.

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

**Код программы:**

import random

from math import sin, cos, exp

def Reference\_f(x):

a, b, c, d = 0.4, 0.4, 0.08, 0.4

return a \* cos(b \* x) + c \* sin(d \* x)

def Calculate\_S\_Hidden(arr\_Hidden\_Y, arr\_Hidden\_W, arr\_RefValues, arr\_Hidden\_T, amount\_Input, amount\_Hidden, iter):

for i in range(len(arr\_Hidden\_Y)):

arr\_Hidden\_Y[i] = 0

for i in range(amount\_Hidden):

for j in range(amount\_Input):

arr\_Hidden\_Y[i] += arr\_Hidden\_W[i][j] \* arr\_RefValues[iter + j]

arr\_Hidden\_Y[i] -= arr\_Hidden\_T[i]

return arr\_Hidden\_Y

def Calculate\_Y\_Hidden(S\_hid):

return 1 / 1 + exp(-S\_hid)

def Calculate\_y(arr\_Hidden\_Y, arr\_W\_Out, t\_Out, amount\_Hidden):

y = 0

for i in range(amount\_Hidden):

y += arr\_Hidden\_Y[i] \* arr\_W\_Out[i]

return y - t\_Out

def change\_w\_out(arr\_W\_Out, gamma\_Out, arr\_Hidden\_Y, amount\_Hidden, alpha):

for i in range(amount\_Hidden):

arr\_W\_Out[i] -= alpha \* gamma\_Out \* arr\_Hidden\_Y[i]

return arr\_W\_Out

def change\_t\_Out(t\_Out, gamma\_Out, alpha\_Out):

t\_Out += alpha\_Out \* gamma\_Out

return t\_Out

def change\_w\_Hidden(arr\_Hidden\_W, gamma\_Hidden, arr\_Hidden\_Y, arr\_RefValues, amount\_Input, amount\_Hidden, iter, alpha):

for i in range(amount\_Hidden):

for j in range(amount\_Input):

arr\_Hidden\_W[i][j] -= (

alpha

\* gamma\_Hidden[i]

\* arr\_Hidden\_Y[i]

\* (1 - arr\_Hidden\_Y[i])

\* arr\_RefValues[j + iter]

)

return arr\_Hidden\_W

def change\_T\_Hidden(arr\_Hidden\_T, gamma\_Hidden, arr\_Hidden\_Y, amount\_Hidden, alpha):

for i in range(amount\_Hidden):

arr\_Hidden\_T[i] += alpha \* gamma\_Hidden[i] \* arr\_Hidden\_Y[i] \* (1 - arr\_Hidden\_Y[i])

return arr\_Hidden\_T

def new\_alpha\_hid(y\_hid\_arr, y, gamma\_hidden):

numerator, denum, new\_alpha = 0, 0, 0

for i in range(len(y\_hid\_arr)):

numerator += (gamma\_hidden[i] \*\* 2) \* y\_hid\_arr[i] \* (1 - y\_hid\_arr[i])

denum += (

(gamma\_hidden[i] \*\* 2) \* (y\_hid\_arr[i] \*\* 2) \* ((1 - y\_hid\_arr[i]) \*\* 2)

)

new\_alpha = (4 \* numerator) / ((1 + y \*\* 2) \* denum)

return new\_alpha

def main():

step = 0.1

amount\_Input = 6

amount\_Hidden = 2

amount\_Train = 30

amount\_Test = 15

arr\_RefValues = [Reference\_f(x \* step) for x in range(amount\_Train)]

arr\_Hidden\_W = []

for i in range(amount\_Hidden):

row = []

for j in range(amount\_Input):

row.append(random.uniform(0, 1))

arr\_Hidden\_W.append(row)

print('Hidden weights: ',arr\_Hidden\_W)

arr\_Hidden\_T = [0 for i in range(amount\_Hidden)]

arr\_Hidden\_Y = [0 for i in range(amount\_Hidden)]

arr\_W\_Out = [random.uniform(0, 1) for i in range(amount\_Hidden)]

t\_Out = random.uniform(0, 1)

gamma\_Out = 0

gamma\_Hidden = [0 for i in range(amount\_Hidden)]

gen\_Count = 0

y = 0

error = 1

min\_error = 1e-5

while error > min\_error:

error = 0

for iter in range(amount\_Train - amount\_Input):

arr\_Hidden\_Y = Calculate\_S\_Hidden(

arr\_Hidden\_Y, arr\_Hidden\_W, arr\_RefValues, arr\_Hidden\_T, amount\_Input, amount\_Hidden, iter

)

for i in range(amount\_Hidden):

arr\_Hidden\_Y[i] = Calculate\_Y\_Hidden(arr\_Hidden\_Y[i])

y = Calculate\_y(arr\_Hidden\_Y, arr\_W\_Out, t\_Out, amount\_Hidden)

gamma\_Out = y - arr\_RefValues[iter + amount\_Input]

for i in range(amount\_Hidden):

gamma\_Hidden[i] = gamma\_Out \* y \* (1 - y) \* arr\_W\_Out[i]

y\_hid\_arr\_sum\_square = 0

for i in range(len(arr\_Hidden\_Y)):

y\_hid\_arr\_sum\_square += arr\_Hidden\_Y[i] \*\* 2

alpha\_Out = 1 / y\_hid\_arr\_sum\_square

alpha\_Hidden = new\_alpha\_hid(arr\_Hidden\_Y, y, gamma\_Hidden)

arr\_W\_Out = change\_w\_out(arr\_W\_Out, gamma\_Out, arr\_Hidden\_Y, amount\_Hidden, alpha\_Out)

t\_Out = change\_t\_Out(t\_Out, gamma\_Out, alpha\_Out)

arr\_Hidden\_W = change\_w\_Hidden(

arr\_Hidden\_W, gamma\_Hidden, arr\_Hidden\_Y, arr\_RefValues, amount\_Input, amount\_Hidden, iter, alpha\_Hidden

)

arr\_Hidden\_T = change\_T\_Hidden(arr\_Hidden\_T, gamma\_Hidden, arr\_Hidden\_Y, amount\_Hidden, alpha\_Hidden)

error += ((y - arr\_RefValues[iter + amount\_Input]) \*\* 2) / 2

gen\_Count += 1

error /= amount\_Train - amount\_Input

if gen\_Count % 5000 == 0:

print('Error value: ', error)

print("Training end\nTraining result")

print("{:^25}{:^25}{:^25}".format("Reference value", "Prediction value", "Deviation"))

for iter in range(amount\_Train - amount\_Input):

arr\_Hidden\_Y = Calculate\_S\_Hidden(arr\_Hidden\_Y, arr\_Hidden\_W, arr\_RefValues, arr\_Hidden\_T, amount\_Input, amount\_Hidden, iter)

for i in range(amount\_Hidden):

arr\_Hidden\_Y[i] = Calculate\_Y\_Hidden(arr\_Hidden\_Y[i])

y = Calculate\_y(arr\_Hidden\_Y, arr\_W\_Out, t\_Out, amount\_Hidden)

print(

"{:< 25}{:< 25}{:< 25}".format(

arr\_RefValues[iter + amount\_Input],

y,

arr\_RefValues[iter + amount\_Input] - y,

)

)

ref\_test\_arr = [

Reference\_f(x \* step) for x in range(amount\_Train - amount\_Input, amount\_Test + amount\_Train)

]

print("Testing result")

print("{:^25}{:^25}{:^25}".format("Reference value", "Prediction value", "Deviation"))

for iter in range(amount\_Test):

arr\_Hidden\_Y = Calculate\_S\_Hidden(

arr\_Hidden\_Y, arr\_Hidden\_W, ref\_test\_arr, arr\_Hidden\_T, amount\_Input, amount\_Hidden, iter

)

for i in range(amount\_Hidden):

arr\_Hidden\_Y[i] = Calculate\_Y\_Hidden(arr\_Hidden\_Y[i])

y = Calculate\_y(arr\_Hidden\_Y, arr\_W\_Out, t\_Out, amount\_Hidden)

print(

"{:< 25}{:< 25}{:< 25}".format(

ref\_test\_arr[iter + amount\_Input],

y,

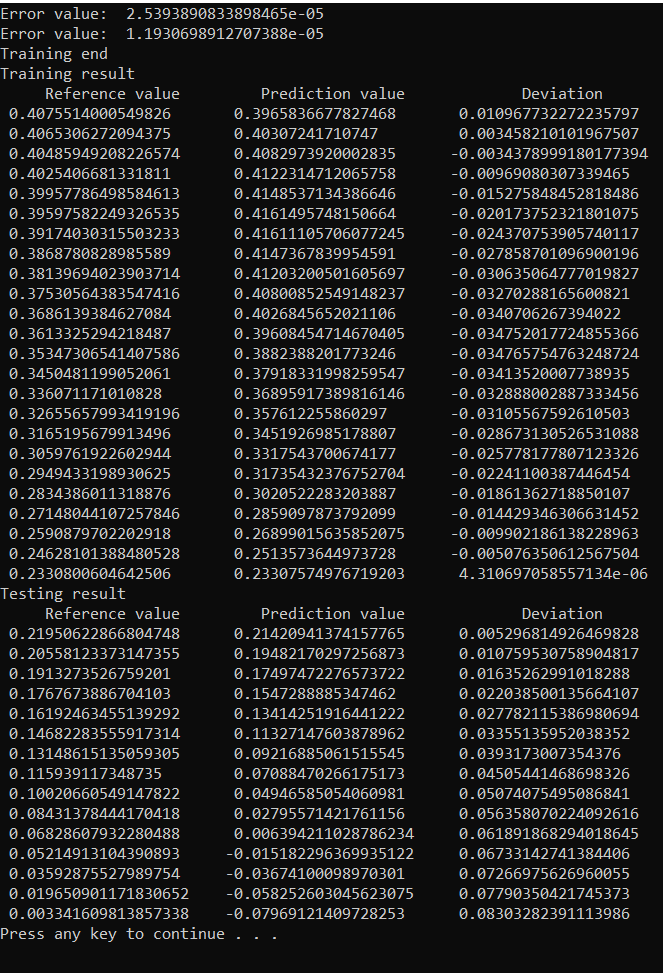
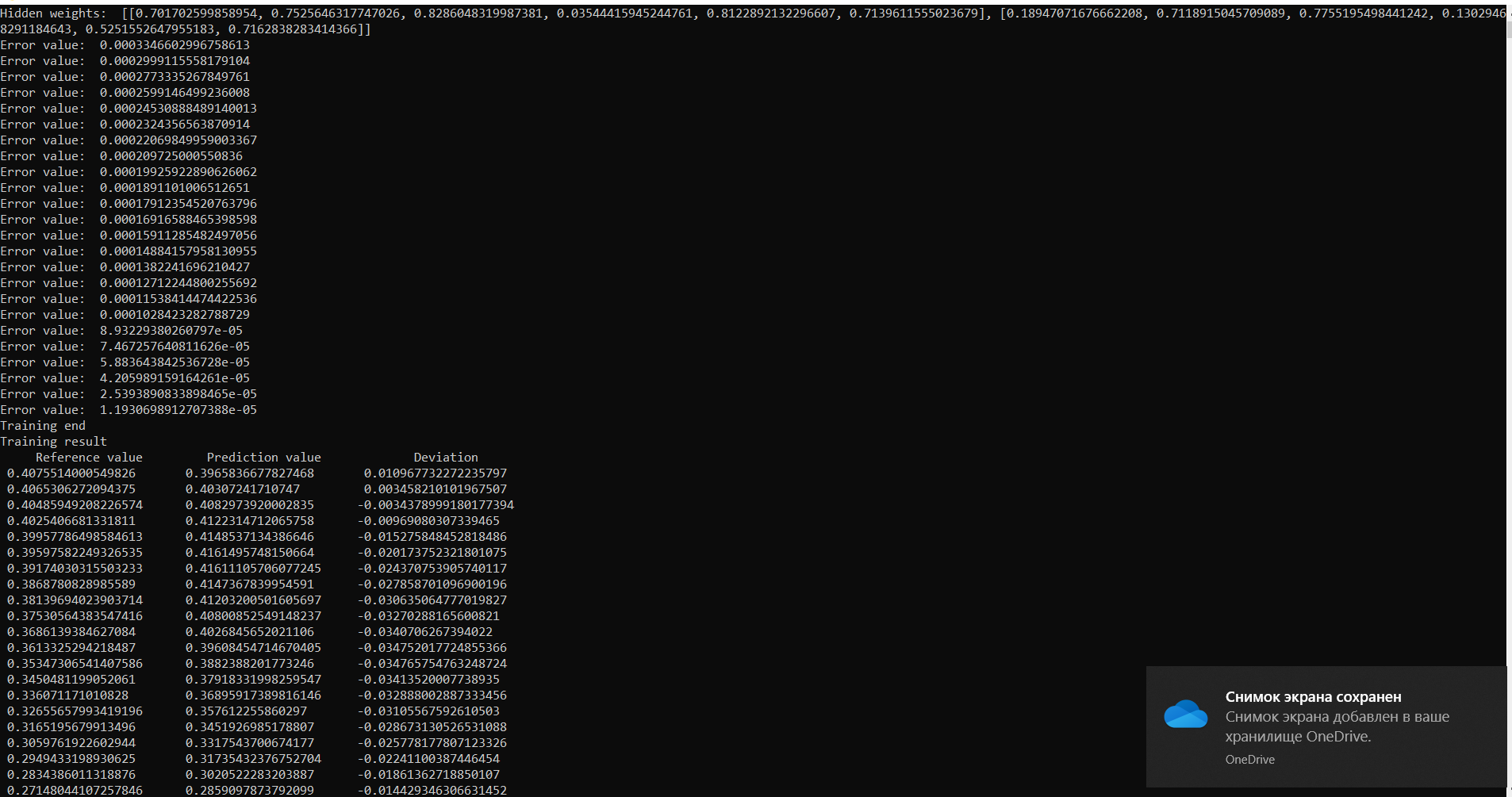
ref\_test\_arr[iter + amount\_Input] - y,

)

)

main()

**Результат работы программы:**



**Вывод:** В среднем, адаптивный шаг обучения даёт лучший результат(сеть с адаптивным шагом должна обучается намного быстрее). Я изучил обучение и функционирование линейной ИНС при решении задач прогнозирования с использованием адаптивного шага.