Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть.

Адаптивный шаг обучения»

Вариант №4

Выполнил:

Студент 2 курса

Группы ПО-7

Дмитрук М.А.

Проверил:

Крощенко А.А.

Брест 2021

**Цель работы:** Изучить обучение и функционирование линейной ИНС с применением адаптивного шага.

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

y = a\*sin(bx) + d

a = 4, b = 8, d = 0.4, кол-во входов ИНС = 3.

**Код программы:**

import math

import random

import matplotlib.pyplot as Graph

def ReferenceFunction(x):

a, b, d = 4, 8, 0.4

return a \* math.sin(b \* x) + d

def Calculate\_W(w\_arr, alpha, reference\_Array, value, i, amount\_Inputs) -> list:

for j in range(len(w\_arr)):

w\_arr[j] -= alpha \* reference\_Array[i + j] \* (value - reference\_Array[i + amount\_Inputs])

return w\_arr

def Calculate\_T(T, alpha, reference\_Array, value, i, amount\_Inputs) -> float:

T += alpha \* (value - reference\_Array[i + amount\_Inputs])

return T

def GetValue(w\_arr, reference\_Array, T, i, amount\_Inputs) -> float:

value = 0

for j in range(amount\_Inputs):

value += w\_arr[j] \* reference\_Array[j + i]

return value - T

def Calculate\_Alpha(etalon\_arr, i, amount\_inputs) -> float:

alpha = 0

for j in range(amount\_inputs):

alpha += etalon\_arr[j + i] \*\* 2

return (1 / (1 + alpha))

def main():

amount\_Inputs = 3 # ---Входные нейроны

min\_Error = 1.0e-29 # -Мин. ошибка

step = 0.1 # ==========Шаг табуляции

amount\_Train = 30 # ---Итерации обучения

amount\_Test = 15 # ====Итерации прогнозирования

w\_arr = [ random.uniform(0, 1) for i in range(amount\_Inputs) ] # генерация весов

T = random.uniform(0, 1) # промежуток

graph\_Arr = [] # массив для построения графика (будут записаны результаты)

error = 10

# Значения эталонной функции для обучения

reference\_Array = [

ReferenceFunction(i \* step) for i in range(amount\_Train)]

# Значения эталонной ф-ции для прогнозирования

test\_RefValues = [

ReferenceFunction(i \* step) for i in range(amount\_Train, amount\_Test + amount\_Train)]

Gen\_iteration = 0

while error > min\_Error:

error = 0

for i in range(amount\_Train - amount\_Inputs):

value = GetValue(w\_arr, reference\_Array, T, i, amount\_Inputs)

alpha = Calculate\_Alpha(reference\_Array, i, amount\_Inputs)

w\_arr = Calculate\_W(w\_arr, alpha, reference\_Array, value, i, amount\_Inputs)

T = Calculate\_T(T, alpha, reference\_Array, value, i, amount\_Inputs)

error += (value - reference\_Array[i + amount\_Inputs]) \*\* 2

error /= amount\_Train - amount\_Inputs

Gen\_iteration += 1

graph\_Arr.append(error)

print("Training end\nTraining result")

print("{:^25}{:^25}{:^25}".format("reference value", "predicted value", "Deviation"))

for i in range(amount\_Train - amount\_Inputs):

value = GetValue(w\_arr, reference\_Array, T, i, amount\_Inputs)

print(

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

reference\_Array[i + amount\_Inputs],

value,

reference\_Array[i + amount\_Inputs] - value,

)

)

print("Testing result:", Gen\_iteration, "epoch")

print("{:^25}{:^25}{:^25}".format("reference value", "predicted value", "Deviation"))

for i in range(amount\_Test - amount\_Inputs):

value = GetValue(w\_arr, test\_RefValues, T, i, amount\_Inputs)

print(

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

test\_RefValues[i + amount\_Inputs],

value,

test\_RefValues[i + amount\_Inputs] - value,

)

)

Graph.plot(graph\_Arr)

Graph.xlabel("Generation")

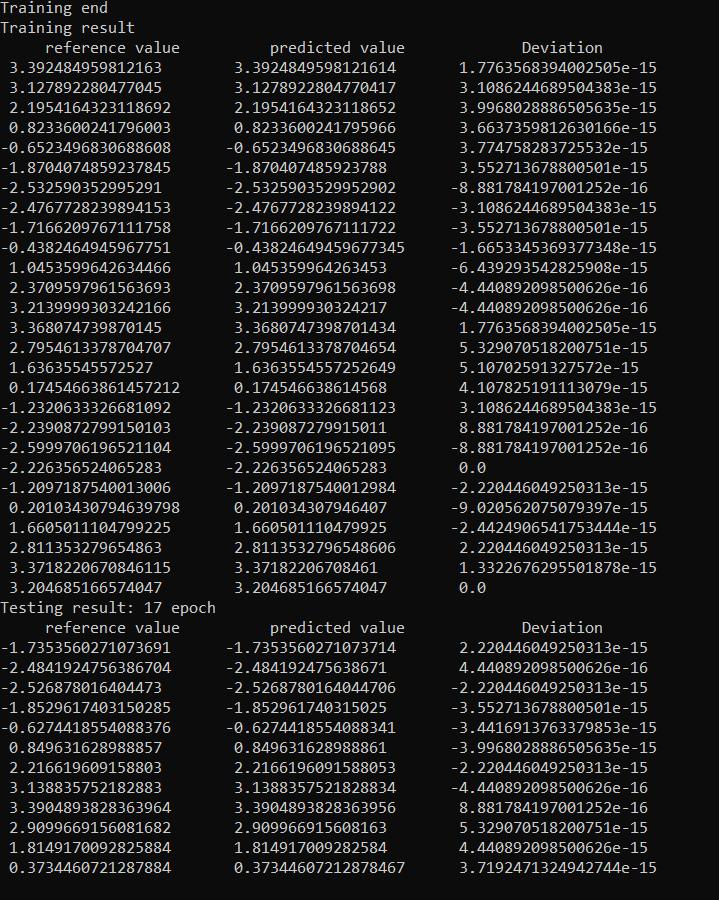
Graph.ylabel("Error")

Graph.grid()

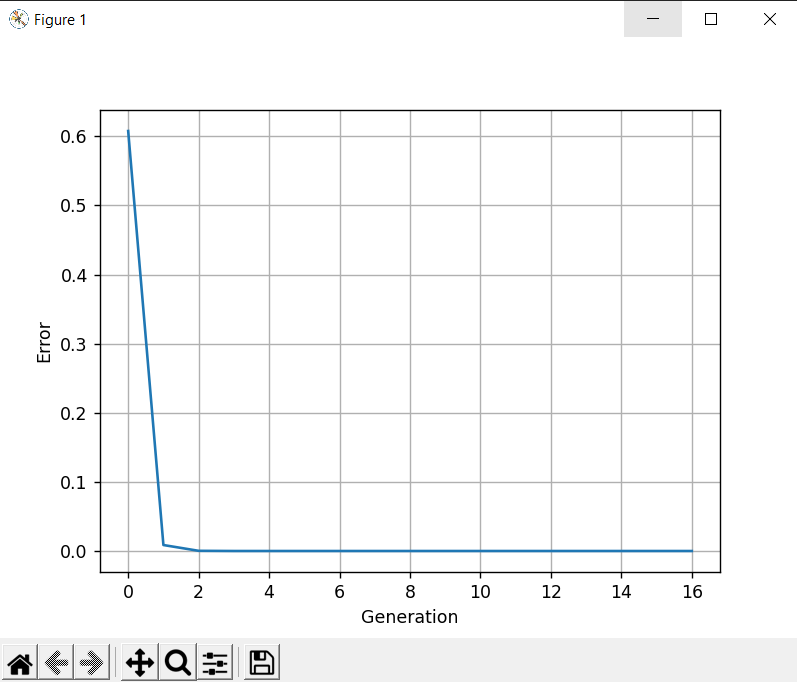
Graph.show()

main()

**Результат работы программы:**



**График изменения ошибки в зависимости от количества эпох:**



**Вывод:** Я изучил обучение и функционирование линейной ИНС с применением адаптивного шага.