Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть. Правило Видроу-Хоффа»

Выполнил:

Студент 2 курса

Группы ПО-7

Рекун И.В.

Проверил:

Крощенко А.А.

Брест 2021

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

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

y = a\*sin(bx) + d

а, b, d = 2, 9, 0.4, кол-во входов ИНС = 3.

**Код программы:**

**import math**

**import random**

**import matplotlib.pyplot as plt**

**def func(x):**

**a, b, d = 2, 9, 0.4**

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

**def output(weight, reference, T, shift, number\_inputs) -> float: # Формула 1.2**

**# Формула 1.2 нахождение выходного значения**

**value = 0**

**for j in range(number\_inputs):**

**value += weight[j] \* reference[j + shift]**

**return value - T**

**def calc(weight, learning\_rate, reference, value, shift, number\_inputs, T):**

**# Формула 1.7 обновление весов**

**for j in range(len(weight)):**

**weight[j] -= (learning\_rate \* (value - reference[number\_inputs + shift]) \* reference[shift + j])**

**def main():**

**training = 30 # количество значений для обучения**

**prediction = 15 # количество значений для прогнозирования**

**number\_inputs = 3 # количество входных нейронов**

**learning\_rate = 0.01 # скорость обучения**

**step = 0.1 # шаг табулирования**

**min\_error = 1.0e-27**

**weight = [random.uniform(0, 1) for i in range(number\_inputs)] # массив рандомных весов**

**T = random.uniform(0, 1) # порог**

**reference = [func(i \* step) for i in range(training)] # эталонные значения для обучения**

**error = 1**

**plot\_arr = []**

**while error > min\_error:**

**error = 0**

**for i in range(training - number\_inputs):**

**value = output(weight, reference, T, i, number\_inputs)**

**calc(weight, learning\_rate, reference, value, i, number\_inputs, T)**

**T += learning\_rate \* (value - reference[number\_inputs + i]) # формула 1.8 обновление порога**

**error += (value - reference[i + number\_inputs]) \*\* 2**

**error /= training - number\_inputs**

**plot\_arr.append(error)**

**plt.plot(plot\_arr)**

**plt.xlabel("Iteration")**

**plt.ylabel("Error")**

**plt.show()**

**print("Training results:")**

**print(f"Weight arr:{weight},T:{T}")**

**print("{:30}{:30}{:30}".format("Reference value", "Output value", "Difference"))**

**for i in range(training - number\_inputs):**

**value = output(weight, reference, T, i, number\_inputs)**

**print(**

**"{:<30}{:<30}{:<30}".format(**

**reference[number\_inputs + i],**

**value,**

**reference[number\_inputs + i] - value,**

**)**

**)**

**reference\_t = [func(i \* step) for i in range(training, training + prediction)] # массив эталонных значений для прогнозирования**

**print("Testing results:")**

**print("{:30}{:30}{:30}".format("Reference value", "Output value", "Difference"))**

**for i in range(prediction - number\_inputs):**

**value = output(weight, reference\_t, T, i, number\_inputs)**

**print(**

**"{:<30}{:<30}{:<30}".format(**

**reference\_t[number\_inputs + i],**

**value,**

**reference\_t[number\_inputs + i] - value,**

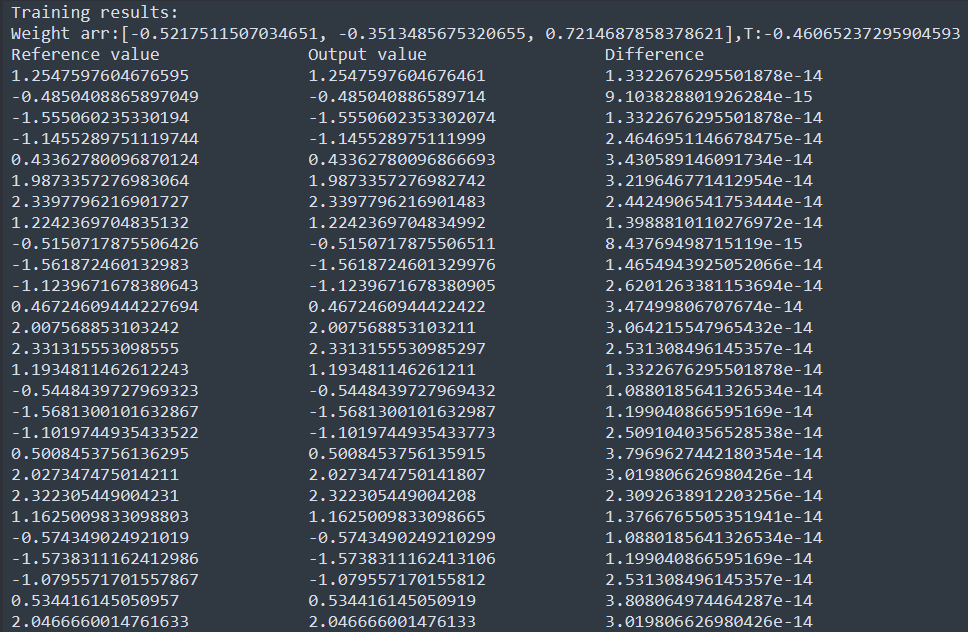
**)**

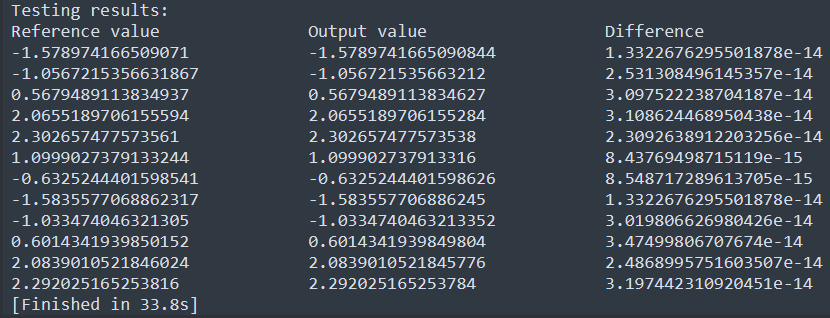
**)**

**if \_\_name\_\_ == "\_\_main\_\_":**

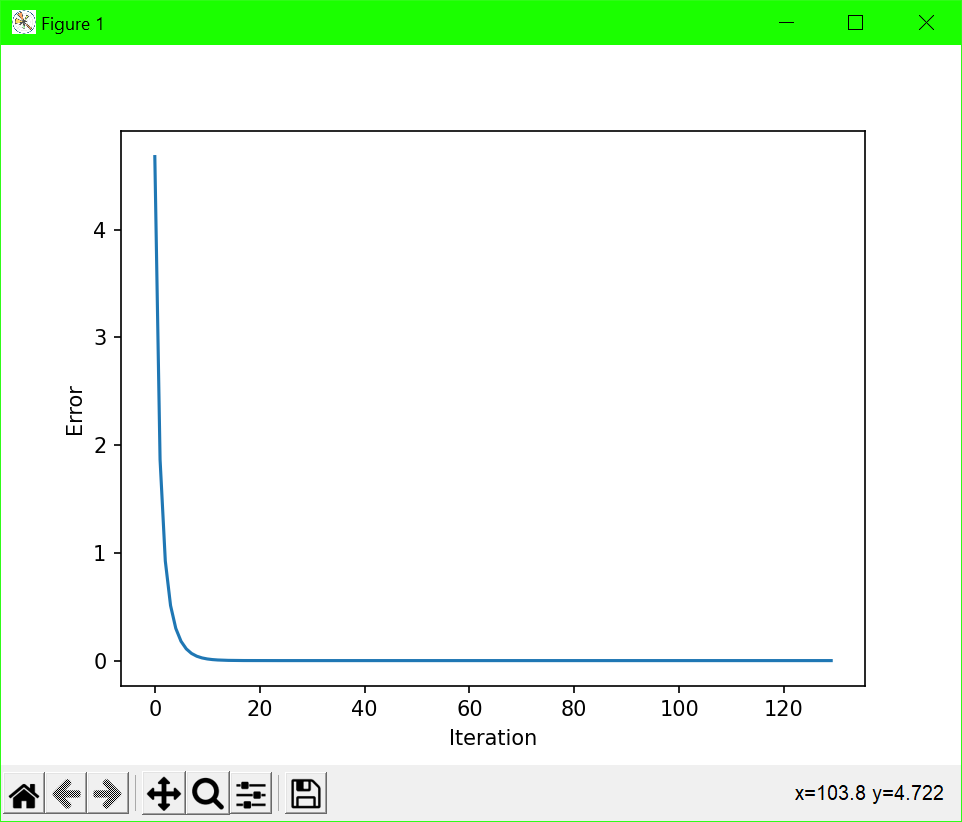
**main()**

**Результат:**





**График изменения ошибки:**



**Вывод:** Изучил работу нейронных сетей и благодаря этому увеличил количество нейронных связей в своем мозгу.