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(2)

Панкратов Р.С.

Проверил:

Крощенко А.А.

2021

Вариант 3

**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

a = 3, b = 7, d = 0.3, кол-во входов ИНС = 5.

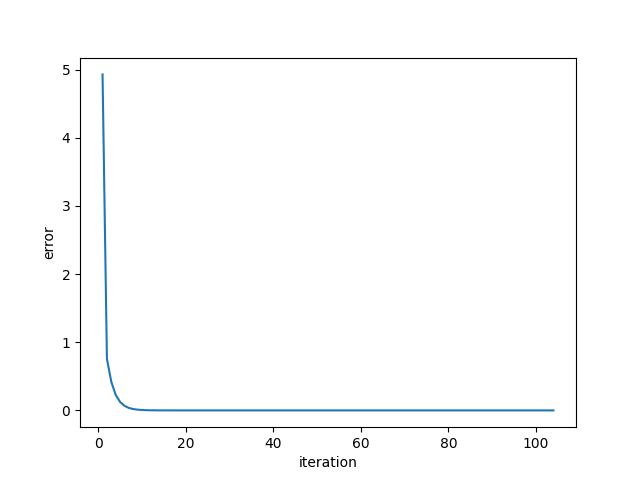
**Код программы:**

**import** math  
**import** random  
**from** typing **import** Tuple, List, Sequence  
**from** dataclasses **import** dataclass  
  
**import** matplotlib.pyplot **as** plt  
  
MIN\_ERROR = 1.0e-27 *# Минимальная ошибка для остановки обучени*TRAINING\_SPEED = 0.01 *# Скорость обучения нейронной сети*@dataclass(frozen=**True**)  
**class** TrainingResult:  
 w: List[float]  
 T: float  
 data\_for\_drawing: Tuple[List[int], List[float]]  
  
**def** func(x: float, a: int, b: int, d: float) -> float:  
 **return** a \* math.sin(b \* x) + d  
  
**def** calculate\_output(  
 nn\_inputs: int, w: Sequence[float], t: float, training\_outputs: Sequence[float], offset: int  
) -> float:  
 output = 0  
 **for** j **in** range(nn\_inputs):  
 output += w[j] \* training\_outputs[j + offset]  
 **return** output - t  
  
**def** training\_nn(nn\_inputs: int, training\_epoch: int, training\_outputs: Sequence[float]) -> TrainingResult:  
 w = [random.uniform(0, 1) **for** \_ **in** range(nn\_inputs)] *# Веса* T = random.uniform(0.5, 1) *# Порог* data\_for\_drawing = ([], [])  
  
 error = 1  
 iteration = 0  
 **while** error > MIN\_ERROR:  
 error = 0  
 iteration += 1  
  
 **for** i **in** range(training\_epoch - nn\_inputs):  
 *# Получение выходного значения нейронной сети, формула(1.2)* output = calculate\_output(nn\_inputs, w, T, training\_outputs, i)  
  
 *# Обновление весов нейронной сети, формула(1.7)* **for** j **in** range(nn\_inputs):  
 ideal\_output = training\_outputs[i + nn\_inputs]  
 w[j] -= TRAINING\_SPEED \* (output - ideal\_output) \* training\_outputs[i + j]  
  
 *# Обновление порога нейронной сети, формула(1.8)* T += TRAINING\_SPEED \* (output - training\_outputs[i + nn\_inputs])  
  
 *# Обновление среднеквадратичной ошибки нейронной сети, формула(1.3)* error += (output - training\_outputs[i + nn\_inputs]) \*\* 2  
  
 error /= training\_epoch - nn\_inputs  
  
 data\_for\_drawing[0].append(iteration)  
 data\_for\_drawing[1].append(error)  
  
 **return** TrainingResult(w, T, data\_for\_drawing)  
  
**def** main():  
 a, b, d = 3, 7, 0.3  
 nn\_inputs = 5 *# Количество входных значений(входов)* training\_epoch, testing\_epoch = 30, 15 *# Количество значений функции для обучения и прогнозирования* step = 0.1 *# Шаг табуляции функции* training\_outputs = [func(i \* step, a, b, d) **for** i **in** range(training\_epoch)]  
 testing\_outputs = [func(i \* step, a, b, d) **for** i **in** range(training\_epoch, training\_epoch + testing\_epoch)]  
 **try**:  
 training\_result = training\_nn(nn\_inputs, training\_epoch, training\_outputs)  
 **except** OverflowError:  
 **return** print(**"Нейронная сеть расходящаеся, скорость обучения слишком большая, уменьшите ее"**)  
  
 plt.plot(\*training\_result.data\_for\_drawing)  
 plt.ylabel(**'error'**)  
 plt.xlabel(**'iteration'**)  
 plt.show()  
  
 print(**"Training result:"**)  
 print(**f"Weight: {**training\_result.w**}, T: {**training\_result.T**}"**)  
 print(**"{:<22}{:<23}{}"**.format(**"Reference value"**, **"Receive value"**, **"Deviation"**))  
  
 **for** i **in** range(training\_epoch - nn\_inputs):  
 output = calculate\_output(nn\_inputs, training\_result.w, training\_result.T, training\_outputs, i)  
 print(**f"{**training\_outputs[i+nn\_inputs]**:<20} {**output**: <21} {**(training\_outputs[i+nn\_inputs] - output)**}"**)  
  
 print(**"\nTesting result:"**)

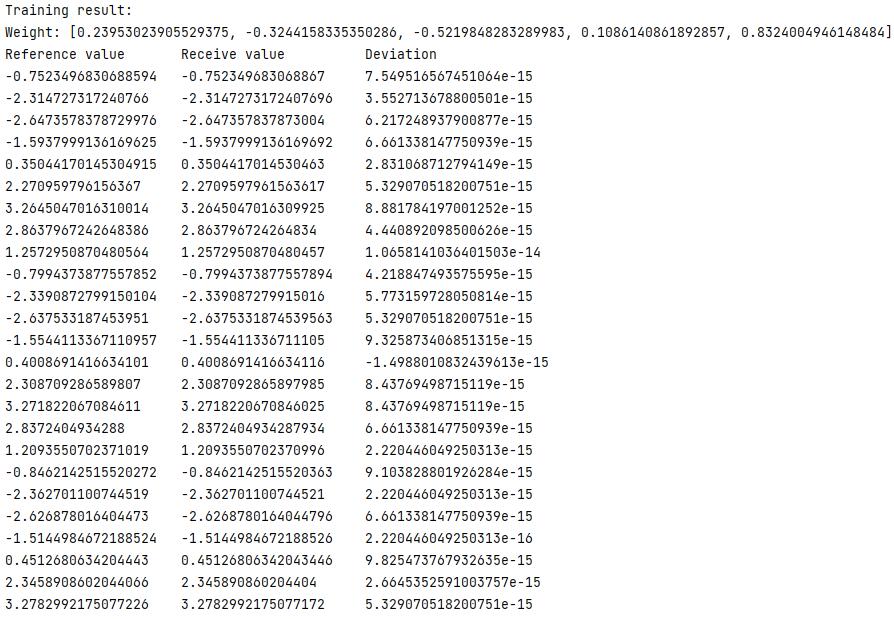
**for** i **in** range(testing\_epoch - nn\_inputs):  
 output = calculate\_output(nn\_inputs, training\_result.w, training\_result.T, testing\_outputs, i)  
 print(**f"{**testing\_outputs[i+nn\_inputs]**:<20} {**output**: <21} {**(testing\_outputs[i+nn\_inputs] - output)**}"**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

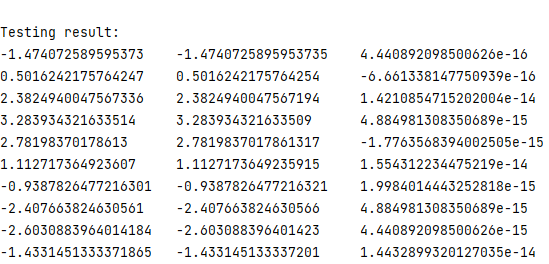
**Результат:**

График:



Вывод программы:





**Вывод:**

Изучил обучение и функционирование линейной ИНС при решении задач прогнозирования на языке программирования Python.