Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторные работы №2**

По дисциплине: “Методы и алгоритмы принятия решений”

Тема: “Нейронные сети”

**Вариант №8**

**Выполнил**:

Крук Д.Н.

студент 2 курса группы ПО-7

**Проверил:**

Крощенко А.А.

Брест 2021

**Цель работы**:

Изучить обучение и функционирование линейной ИНС при решении задач прогнозирования.

**Задание**:

Написать программу моделирования прогнозирующей линейной ИНС. Для тестирования использовать функцию y = a \* sin(b \* x) + d. (a = 4, b = 7, d =

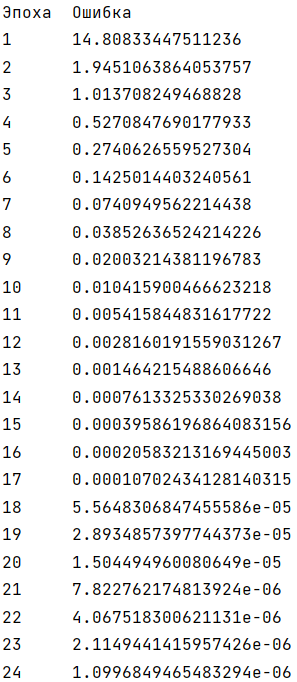
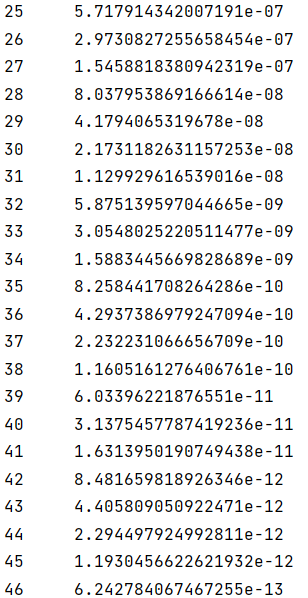
0.2, количество входов ИНС: 4)

**Выполнение:**

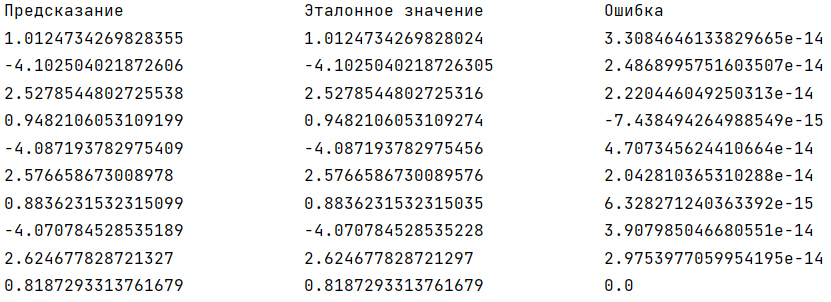
**Код программы:**

**from** math **import** sin  
**import** random  
  
inp, E, alpha = 4, 0.000000000001, 0.05  
W, T = [random.uniform(-1, 1) **for** \_ **in** range(inp)], random.uniform(-1, 1)  
step = 0.3  
  
  
**def** func\_value(x) -> float:  
 a, b, d = 4, 7, 0.2  
 **return** a \* sin(b \* x) - d  
  
  
**def** value\_data() -> list:  
 start = 1  
 data\_size = 10  
 line = []  
 **for** \_ **in** range(data\_size):  
 x = start  
 vector = []  
 **for** \_ **in** range(inp):  
 vector.append(func\_value(x))  
 x += step  
 line.append([vector, func\_value(x)])  
 start += step  
 **return** line  
  
  
**def** adaptive\_step(vector: list):  
 **global** alpha  
 alpha = 1 / (1 + sum([x \*\* 2 **for** x **in** vector]))  
  
  
**def** change(error: float, vector: list) -> **None**:  
 **global** T, W  
 **for** index **in** range(inp):  
 W[index] -= alpha \* error \* vector[index]  
 T += alpha \* error  
  
  
**def** predict(vector: list) -> float:  
 output = 0  
 **for** ind, val **in** enumerate(vector):  
 output += W[ind] \* val  
 output -= T  
 **return** output  
  
  
**def** education() -> **None**:  
 f\_error = E + 1  
 value = value\_data()  
 ep = 0  
 print(**'{0:7}{1:<30}'**.format(**"Эпоха"**, **"Ошибка"**))  
 **while** abs(f\_error) > E:  
 f\_error = 0  
 **for** val **in** value:  
 error = predict(val[0]) - val[1]  
 adaptive\_step(val[0])  
 change(error, val[0])  
 f\_error += abs(error)  
 ep += 1  
 print(**'{0:<7}{1:<30}'**.format(ep, f\_error))  
 print()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 education()  
 data = value\_data()  
 print(**'{0:<30}{1:<30}{2:<30}'**.format(**"Предсказание"**, **"Эталонное значение"**, **"Ошибка"**))  
 **for** d **in** data:  
 p = predict(d[0])  
 print(**'{0:<30}{1:<30}{2:<30}'**.format(p, d[1], p - d[1]))

**Результат программы**

****

Средняя квадратичная ошибка на каждой эпохе

****

Предсказание