Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине: “Методы и алгоритмы принятия решений”

Тема: “Линейная искусственная нейронная сеть.

Правило обучения Видроу-Хоффа”

**Вариант №9**

**Выполнил**: студент 2 курса группы ПО-7 Крупенков Михаил Дмитриевич

**Проверил:** Крощенко А.А.

Брест 2021

**Цель работы:**

изучить обучение и функционирование линейной ИНС при решении задач прогнозирования.

**Постановка задачи:**

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

**

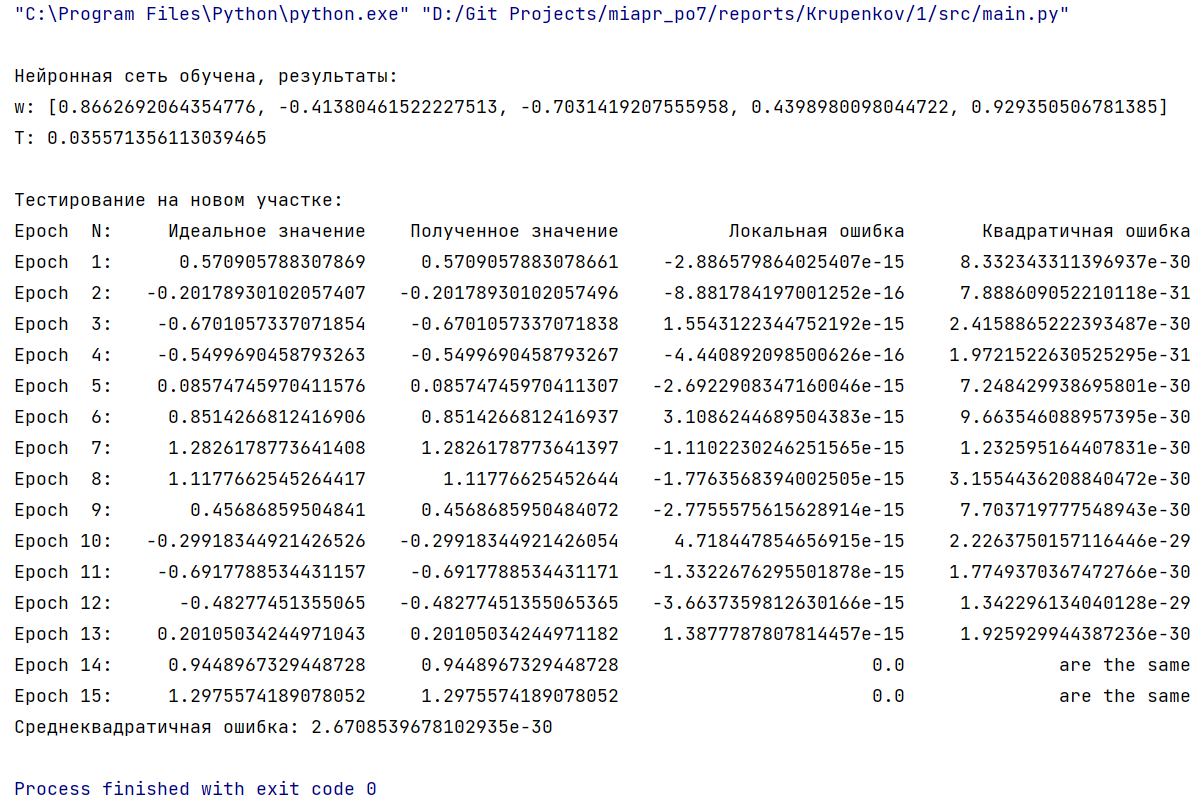
Согласно варианту 9 в таблице

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | a | b | d | Кол-во входов ИНС |
| 9 | 1 | 8 | 0.3 | 5 |

**Код программы:**

**from** math **import** sin  
**from** random **import** uniform  
**from** typing **import** List  
  
*# Гиперпараметры обучения*MIN\_SQUARE\_ERROR = 1e-32 *# Минимальная ошибка для остановки обучения*TRAINING\_SPEED = 1.0e-1 *# Скорость обучения нейронной сети*TRAINING\_EPOCH\_AMOUNT = 30 *# Количество значений функции (эпох) для обучения*TESTING\_EPOCH\_AMOUNT = 15 *# Количество значений функции (эпох) для прогнозирования*MAX\_ITERATIONS\_AMOUNT = 20  
  
  
*# Функция по условию (Вариант 9)***def** function(x: float) -> float:  
 **return** sin(8 \* x) + 0.3  
  
  
**def** main() -> **None**:  
 inputs\_amount = 5 *# Количество входов* step = 0.1 *# Шаг табуляции функции  
  
 # Значения функции для обучения* training\_outputs: List[float] = [function(i \* step) **for** i **in** range(TRAINING\_EPOCH\_AMOUNT + inputs\_amount)]  
  
 *# Значения функции для прогнозирования* testing\_outputs: List[float] = [  
 function(i \* step) **for** i **in** range(TRAINING\_EPOCH\_AMOUNT, TRAINING\_EPOCH\_AMOUNT + TESTING\_EPOCH\_AMOUNT + inputs\_amount)  
 ]  
  
 w: List[float] = [uniform(0, 1) **for** \_ **in** range(inputs\_amount)] *# Список всех весов* t: float = uniform(0, 1) *# Порог* **try**: *# Проверка на расходимость* iteration = 0 *# Счетчик итераций* square\_error: float = MIN\_SQUARE\_ERROR *# Среднеквадратичная ошибка  
  
 # Обучение* **while** square\_error >= MIN\_SQUARE\_ERROR **and** iteration < MAX\_ITERATIONS\_AMOUNT:  
 square\_error\_sum = 0  
 iteration += 1  
  
 **for** epoch **in** range(TRAINING\_EPOCH\_AMOUNT):  
 *# Вычисление выходного значения (Формула 1.2)* output: float = 0  
 **for** j **in** range(inputs\_amount):  
 output += w[j] \* training\_outputs[epoch + j]  
 output -= t  
  
 ideal\_output: float = training\_outputs[epoch + inputs\_amount] *# Истинное значение функции* error: float = output - ideal\_output *# Отклонение от функции  
  
 # Обновление весов нейронной сети (Формула 1.7)* **for** i **in** range(inputs\_amount):  
 w[i] -= TRAINING\_SPEED \* error \* training\_outputs[epoch + i]  
  
 *# Обновление порога нейронной сети (Формула 1.8)* t += TRAINING\_SPEED \* error  
  
 *# Обновление среднеквадратичной ошибки нейронной сети (Формула 1.3)* square\_error\_sum += error \*\* 2  
  
 *# Вывод результатов  
 # print(f'Iteration {iteration:3} Epoch {epoch + 1:2}: {ideal\_output:21} {output:21} '  
 # f'{error:24} {error \*\* 2 if error else " are the same":24}')* square\_error = square\_error\_sum / TRAINING\_EPOCH\_AMOUNT  
 *# print(f'Iteration {iteration:3} Square error: {square\_error}')* print(**'\nНейронная сеть обучена, результаты:'**)  
 print(**f'w: {**w**}\n'  
 f'T: {**t**}\n'**)  
 print(**'Тестирование на новом участке:'**)  
 print(**'Epoch N: Идеальное значение Полученное значение '  
 'Локальная ошибка Квадратичная ошибка '**)  
  
 square\_error\_sum = 0  
 *# Тестирование* **for** epoch **in** range(TESTING\_EPOCH\_AMOUNT):  
 *# Вычисление выходного значения (Формула 1.2)* output: float = 0  
 **for** j **in** range(inputs\_amount):  
 output += w[j] \* testing\_outputs[epoch + j]  
 output -= t  
  
 ideal\_output: float = testing\_outputs[epoch + inputs\_amount] *# Истинное значение функции* error: float = output - ideal\_output *# Отклонение от функции  
  
 # Обновление среднеквадратичной ошибки нейронной сети (Формула 1.3)* square\_error\_sum += error \*\* 2  
  
 *# Вывод результатов* print(**f'Epoch {**epoch + 1**:2}: {**ideal\_output**:21} {**output**:21} '  
 f'{**error**:24} {**error \*\* 2 **if** error **else " are the same":24}'**)  
  
 square\_error = square\_error\_sum / TRAINING\_EPOCH\_AMOUNT  
 print(**f'Среднеквадратичная ошибка: {**square\_error**}'**)  
  
 **except** OverflowError:  
 print(**'Слишком большая скорость обучения, выход из программы'**)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

**Результаты программы:**



**Вывод:**

Изучил обучение и функционирование линейной ИНС при решении задач прогнозирования