Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 5
по дисциплине «Теория систем и системный анализ»

Тема: «Двумерный поиск для подбора коэффициентов простейшей нейронной сети на примере решения задачи линейной регрессии экспериментальных данных»

Вариант 1

Выполнил:
Александров А. А.,
студент группы ИУ8-31

Проверил: Коннова Н.С.,
доцент каф. ИУ8

Цель работы

Знакомство с простейшей нейронной сетью и реализация алгоритма поиска ее весовых коэффициентов на примере решения задачи регрессии экспериментальных данных.

Условие задачи

В зависимости от варианта работы найти линейную регрессию функции $y(x)$

(коэффициенты наиболее подходящей прямой c, d) по набору ее N дискретных значений, заданных равномерно на интервале $[a, b]$ со случайными ошибками $e_i \square A\ \mathrm{rnd}(\square 0.5; 0.5)$.

Выполнить расчет параметров c, d градиентным методом. Провести двумерный пассивный поиск оптимальных весовых коэффициентов нейронной сети (НС) регрессии.

c = 3	a = -2	N = 16
d = 1	b = 2	A = 2

График заданной функции

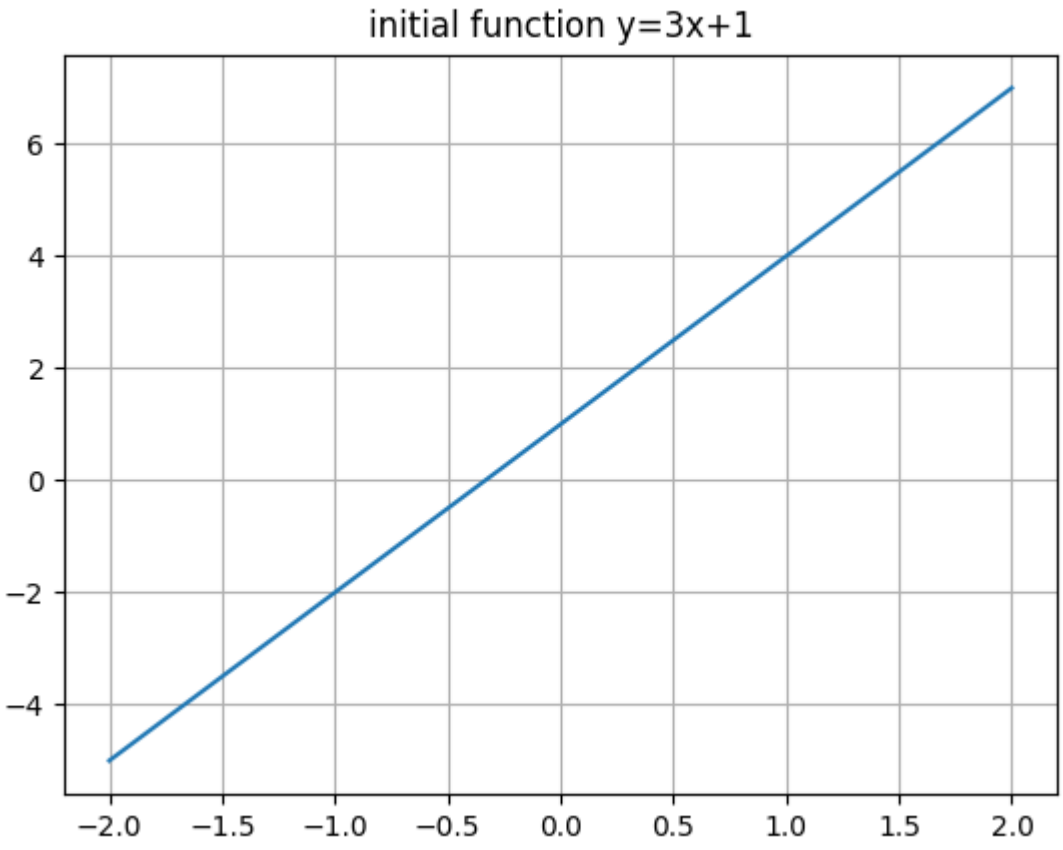

Рисунок 1 - График функции y = 3x + 1

Результат работы программы

c = 3.0

d = 1.187284279798289

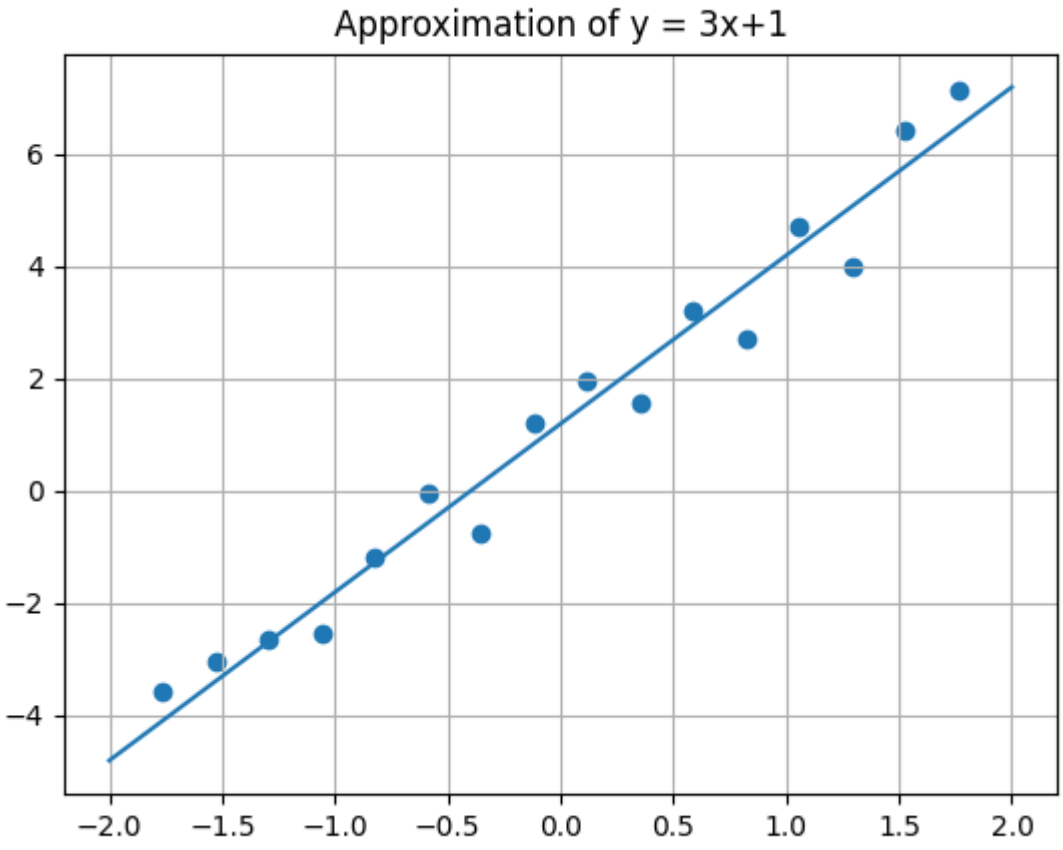

Рисунок 2 – График аппроксимированной функции с экспериментами

Программа предоставлена в репозитории https://github.com/Vumba798/tsisa_lab05

Выводы

Была построена простейшая нейронная сеть, состоящая из одного нейрона. Был использован метод наименьших квадратов в условиях нахождения весовых коэффициентов нейронной сети. Полученная программа позволяет получать приближённый график линейной функции по заданным экспериментам.

Ответ на контрольный вопрос

1. Поясните суть метода наименьших квадратов.

Метод наименьших квадратов позволяет определить оптимальные весовые коэффициенты путём минимизации суммы квадратов ошибок с помощью предложенных в задании методов поиска минимума.

$$E^2(w_1, w_0) = \sum_{i=1}^{N} \left[y(x_i) - t_i \right]^2 \to \min_{c,d} .$$

Иными словами, решение задачи сводится к нахождению минимума функции двух переменных.

Приложение 1. Исходный код файла approximate.py

```
import matplotlib.pyplot as plt
import numpy as np
from random import uniform


tau = 1.618034
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def print(self):
        print("X =", self.x)
        print("Y =", self.y)
        print()
class NeuralNetwork:
    def __init__(self, c, d, a, b, N):
        self.a = a
        self.b = b
        self.c = c
        self.d = d
        self.N = N
        self.experiments = list()
        for i in range(self.N):
            tmpX = ((self.b-self.a) / (self.N+1)*(i+1) + self.a)
            tmpY = tmpX*self.c+self.d+2*uniform(-0.5,0.5)
            self.experiments.append(Point(tmpX, tmpY))
    def f(self, c, d, x):
        return c*x + d
    def sum_of_squares(self, c, d):
        sum = 0
        for point in self.experiments:
            sum += pow(self.f(c, d, point.x) - point.y, 2)
        return sum
    def passive_search_(self, epsilon, c_min, c_max, d_min, d_max):
        N = 0
        while (c_max - c_min) / (N + 1) > epsilon:
            N += 1
        c_arr = []
        for i in range(N):
            c_arr.append(i/(N+1)*(c_max-c_min) + c_min)
        squares = []
        for c_tmp in c_arr:
            d = self.golden_section_(epsilon, d_min, d_max, c_tmp)
            squares.append(self.sum_of_squares(c_tmp, d))
        self.c_ = c_arr[squares.index(min(squares))]
        self.d_ = self.golden_section_(epsilon, d_min, d_max, self.c_)
    def golden_section_(self, epsilon,
                        d_min, d_max, c):
        left = d_min
        right = d_max
        while right - left > epsilon:
            d_right = (right - left) / tau + left;
            d_left = right - (right - left) / tau
            if abs(self.sum_of_squares(c, d_left)) <
abs(self.sum_of_squares(c, d_right)):
                right = d_right
```

```
            else:
                left = d_left
        return (left + right) / 2
    def search(self):
        self.passive_search_(0.1, 2.5, 3.5, -4, 4)
        self.golden_section_(0.1, 0, 2, self.c_)
    def print(self):
        print("c = ", self.c_)
        print("d = ", self.d_)
    def draw(self):
        x_arr = []
        y_arr = []
        x = np.linspace(self.a, self.b)
        y = self.c_ * x + self.d_
        for point in self.experiments:
            x_arr.append(point.x)
            y_arr.append(point.y)
        plt.grid(True)
        plt.title("Approximation of y = 3x+1")
        plt.scatter(x_arr, y_arr)
        plt.plot(x, y)
        plt.savefig("result")
        plt.show()

if __name__ == "__main__":
    nn = NeuralNetwork(3, 1, -2, 2, 16)
    nn.search()
    nn.print()
    nn.draw()
```