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 15

Выполнил: Ушаков З. М.,
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 $e_i = A\,\mathrm{rnd}(-0.5;0.5)$. Выполнить расчет параметров c,d градиентным методом. Провести двумерный пассивный поиск оптимальных весовых коэффициентов нейронной сети (НС) регрессии.

c = 0.1	a = -5	N = 32
d = 2	b = 0	A = 0.2

График функции

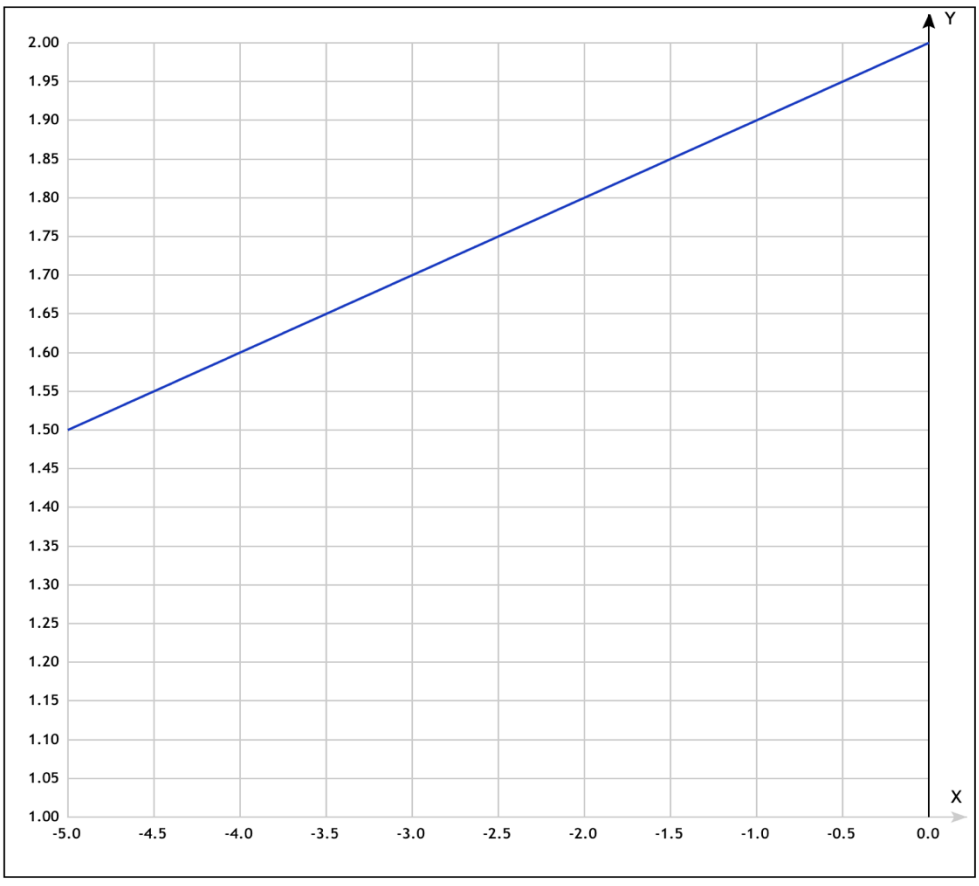


График функции $y = 0.1x + 2$

Результат работы программы

Вариант 15

x: -4.84848 | y: 1.53676

x: -4.69697 | y: 1.44998

x: -4.54545 | y: 1.45403

x: -4.39394 | y: 1.54008

x: -4.24242 | y: 1.64653

x: -4.09091 | y: 1.53021

x: -3.93939 | y: 1.68222

x: -3.78788 | y: 1.57126

x: -3.63636 | y: 1.64093

x: -3.48485 | y: 1.67936

x: -3.33333 | y: 1.7472

x: -3.18182 | y: 1.66826

x: -3.0303 | y: 1.61898

x: -2.87879 | y: 1.66513

x: -2.72727 | y: 1.74196

x: -2.57576 | y: 1.75504

x: -2.42424 | y: 1.84415

x: -2.27273 | y: 1.81295

x: -2.12121 | y: 1.70658

x: -1.9697 | y: 1.85931

x: -1.81818 | y: 1.84553

x: -1.66667 | y: 1.75826

x: -1.51515 | y: 1.878

x: -1.36364 | y: 1.94719

x: -1.21212 | y: 1.87164

x: -1.06061 | y: 1.83352

x: -0.909091 | y: 1.87351

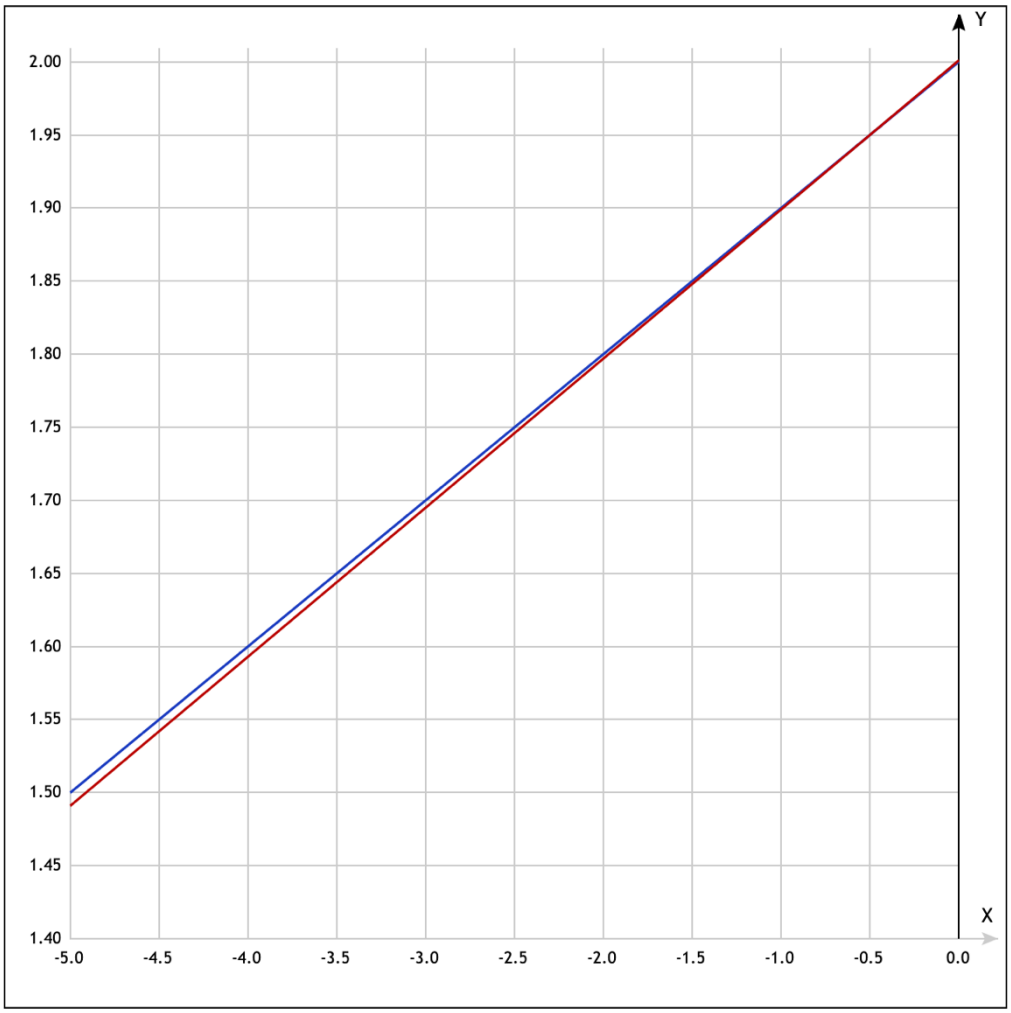
x: -0.757576 | y: 1.88675

x: -0.606061 | y: 1.86207

x: -0.454545 | y: 1.90431

x: -0.30303 | y: 2.05103

```
x: -0.151515 | y: 2.01707
=======================Result=======================
w1 = 0.102009 | w0 = 2.00127
```



Результат работы программы

Выводы

Реализовал простейшую нейронную сеть и научился использовать метод наименьших квадратов в условиях нахождения весовых коэффициентов нейронной сети. С помощью данной нейронной сети можно показать линейную зависимость между некими переменными.

Приложение 1. Код программы

Файл neural_network.hpp:

```
//
// Created by Захар on 22.11.2020.
//

#ifndef TSISA_LAB05_NEURAL_NETWORK_HPP
#define TSISA_LAB05_NEURAL_NETWORK_HPP

#include <random>
#include <vector>
#include <cmath>
#include <iostream>
#include <utility>
#include <algorithm>

using std::cout;
using std::endl;

class neuron {
private:
    double x;
    double y;

public:
    neuron() = default;

    void set_x(double buf_x) {
        x = buf_x;
    }

    void set_y(double buf_y) {
        y = buf_y;
    }

    [[nodiscard]] auto get_x() const noexcept -> double {
        return x;
    }

    [[nodiscard]] auto get_y() const noexcept -> double {
        return y;
    }

    auto operator<(neuron& neuron1) const -> bool {
        return y < neuron1.y;
    }

    auto operator==(neuron& neuron1) const -> bool {
        return y == neuron1.y && x == neuron1.x;
    }
```

```
    neuron(neuron& neuron1) {
        x = neuron1.x;
        y = neuron1.y;
    }

    neuron& operator=(const neuron& neuron1) = default;

};

auto random(const double a, const double b) -> double {
    if (a > b) throw std::invalid_argument("Invalid segment");
    std::random_device rd;
    std::mt19937_64 rng(rd());
    std::uniform_real_distribution<double> rand(a, b);
    return rand(rng);
}

auto linear_function(double c, double d, double x) -> double {
    return  c * x + d;
}

auto random_error(double a, double b, double A) -> double {
    return A * random(a, b);
}

auto fill_network(double c, double d, double A ,double a, double b,
size_t numbers_neurons) -> std::vector<neuron> {
    auto neuron_distance = (b - a) /
static_cast<double>(numbers_neurons + 1);
    std::vector<neuron> neurons(numbers_neurons);
    auto current_x = a;
    for (auto& neuron : neurons) {
        current_x += neuron_distance;
        neuron.set_x(current_x);
        neuron.set_y(linear_function(c, d, neuron.get_x()) +
random_error(-0.5, 0.5, A));
    }
    return neurons;
}


auto sum_x(const std::vector<neuron>& neurons) -> double {
    auto sum = 0.0;
    for (const auto& _neuron : neurons) {
        sum += _neuron.get_x();
    }
    return sum;
}

auto sum_y(const std::vector<neuron>& neurons) -> double {
```

```
    auto sum = 0.0;
    for (const auto& _neuron : neurons) {
        sum += _neuron.get_y();
    }
    return sum;
}

auto sum_square_x(const std::vector<neuron>& neurons) -> double {
    auto sum = 0.0;
    for (const auto& _neuron : neurons) {
        sum += std::pow(_neuron.get_x(), 2);
    }
    return sum;
}

auto sum_xy(const std::vector<neuron>& neurons) -> double {
    auto sum = 0.0;
    for (const auto& _neuron : neurons) {
        sum += (_neuron.get_x() * _neuron.get_y());
    }
    return sum;
}

auto result(const std::vector<neuron>& neurons) -> std::pair<double,
double> { // MNK
    double w0, w1;
    w1 = (sum_x(neurons) * sum_y(neurons) - neurons.size() *
sum_xy(neurons))
         / (std::pow(sum_x(neurons), 2) - neurons.size() *
sum_square_x(neurons));
    w0 = (sum_y(neurons) - w1 * sum_x(neurons)) / neurons.size();
    return std::make_pair(w0, w1);
}

[[maybe_unused]] void print(const std::vector<neuron>& neurons){
    for(const auto& neuron : neurons) {
        std::cout << "x: " << neuron.get_x() << " | " << "y: " <<
neuron.get_y() << std::endl;
    }
}



#endif //TSISA_LAB05_NEURAL_NETWORK_HPP
```

Файл main.cpp:

```
#include "../include/neural_network.hpp"

int main () {
    // Variant 15
```

```
    const double a = -5;
    const double b = 0;
    const double c = 0.1;
    const double d = 2;
    const size_t N = 32;
    const double A = 0.2;

    auto network = fill_network(c, d, A, a, b, N);

    auto w_result = result(network);

    print(network);

    std::cout <<
"======================Result======================" << std::endl;
    std::cout << "w1 = " << w_result.second << " | " << "w0 = " <<
w_result.first << std::endl;

    return 0;
}
```

Контрольный вопрос

Поясните суть метода наименьших квадратов.

Метод наименьших квадратов используется решения задач, основанных на минимизации суммы квадратов отклонений некоторых функций от искомых переменных. С его помощью можно находить линейные (и не только зависимости) нескольких переменных и предсказывать дальнейшие значения этих переменных.