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 10

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

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

г. Москва,
2020 г.

1. Цель работы

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

2. Условие задачи

Вариант № 10.

В зависимости от варианта работы (табл. 1) найти линейную регрессию функции $y(x)$ (коэффициенты наиболее подходящей прямой c, d) по набору 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.

w1 = 1, w0 = 0, a = -2, b = 2, N = 24, A = 2. Алгоритм поиска c - дихотомия, алгоритм поиска d — золотое сечение.

3. Графики

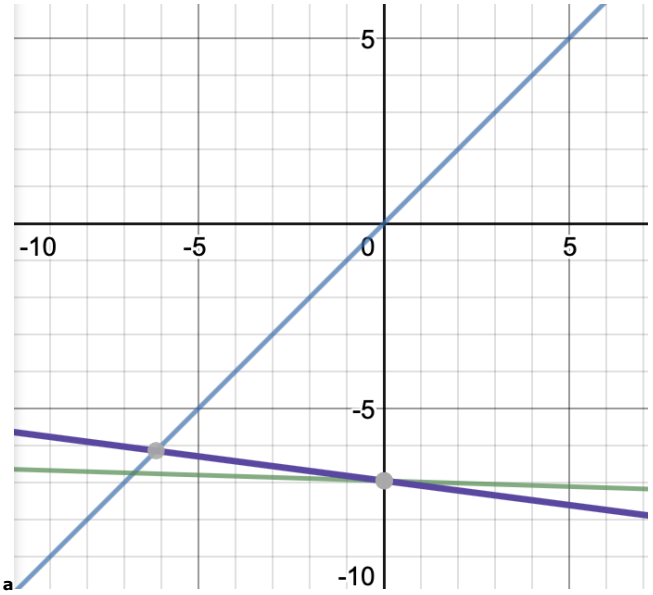


Рисунок 1. Графики, постороенные по результатам работы программы

Зеленый график – y=x, синий график построен при шуме A=0, фиолетовый график построен при шуме A=2.

4. Результат работы программы

```
Function y = x + 0
Without noise
Cmin = 6.95327e-310
Cmax = 6.95327e-310
Dmin = 6.95327e-310
Dmax = 6.95327e-310
w1 = -0.0310937
w0 = -6.95327e-310
With noise A = 2
Cmin = 6.95327e-310
Cmax = 6.95327e-310
Dmin = 6.95327e-310
Dmax = 6.95327e-310
w1 = -0.0310937
w0 = -6.95327e-310
```

5. Выводы

В процессе выполнения лабораторной работы была реализована простейшая нейронная сеть, используя метод наименьших квадратов в условиях нахождения весовых коэффициентов нейронной сети. Результаты работы совпали с ожидаемыми, при отсутствии шума алгоритм дает точные значения параметров регрессии.

6. Контрольные вопросы

1. Поясните суть метода наименьших квадратов.

Решение данным методом сводится к нахождению экстремума функции двух переменных.

Задача заключается в нахождении коэффициентов линейной зависимости, при которых функция двух переменных a и b:

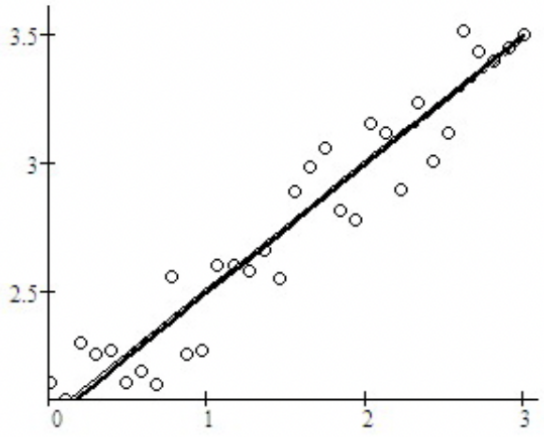
$$F(a,b) = \sum_{i=1}^{n} (y_i - (a\,x_i + b)^2)$$

принимает наименьшее значение. То есть, при данных a и b сумма квадратов отклонений экспериментальных данных от найденной прямой будет наименьшей. В этом вся суть метода наименьших квадратов.

2. Сформулируйте нейросетевой подход к задачам регрессии.

В отсутствие шума ($A = 0$) МНК дает точные значения параметров регрессии (3): $c^* = c, \quad d^* = d$.

Графики на рис. 2 иллюстрируют погрешности приближения в условиях шума (полужирная линия – точная зависимость, круглые маркеры – зашумленные отсчеты, тонкая сплошная линия – нейросетевая регрессия).



Нейросетевая линейная регрессия экспериментальных данных

Приложение. Исходный код программы

```
#include
<iostream>
            #include <random>
            #include <vector>
            #include <algorithm>


            const int N = 24;
            const double a = -2.0;
            const double b = 2.0;
            const double stage = (double)((b - a) / (N - 1));
            const double c = 1.0;
            const double d = 0.0;
            const double A = 2.0;


            double function(const double &x) {
                return c * x + d;
            }
            struct Pair {
                double x;
                double y;
            };




            std::vector<std::pair<double, double>> f(24);


            double recurssion_func(double C, double D) {
                double sum_E = 0;
                for (int i = 0; i < N; i++) {
                    double x = f[i].first;
                    double t = f[i].second;
                    double y = C * x + D;
                    sum_E = sum_E + pow((y - t), 2);
                }
                return sum_E;
            }


            std::vector<Pair> rand(const int N, const double noise) {
                std::vector<Pair> pair (N);
                std::random_device rand;
```

```
    std::mt19937 gen(rand());
    std::uniform_real_distribution<double> er(-0.5, 0.5);
    for (size_t i = 0; i < N; ++i) {
        pair[i].x = a + i * stage;
        pair[i].y = function( a + i * stage) + noise * er(gen);
    }
    return pair;
}


double Metod_Dichotomy(double a, double b, double c){
    const double Epsilon = 0.1;
    const double Delta = 0.01;
    double x_left, x_right, y_left, y_right;
    do
    {
        x_left = 0.5 * (b + a) - Delta;
        x_right = 0.5 * (b + a) + Delta;
        y_left = recurssion_func(c, x_left);
        y_right = recurssion_func(c, x_right);
        if (y_left > y_right)
        {
            a = x_left;
        }
        else{
            b = x_right;
        }
    } while ((b - a) > Epsilon);
    return (a + b) / 2;
}
double Method_golden(std::vector<Pair>& p, double Dmin, double Dmax) {
    double length = std::abs(Dmax - Dmin);
    std::swap(Dmin, Dmax);
    Dmin = std::fabs(Dmin);
    Dmax = std::fabs(Dmax);
    const double e = 0.1;
    const double t = (std::sqrt(5) + 1) / 2;
    double d_k1 = Dmin + (1 - 1/t)*Dmax;
    double d_k2 = Dmin + Dmax / t;
    double f_k1 = function(-d_k1);
    double f_k2 = function(-d_k2);
    while (length > e){
        if (f_k1 < f_k2){
            Dmax = d_k2;
            d_k2 = Dmin + Dmax - d_k1;
            f_k2 = function(-d_k2);
        } else {
```

```
            Dmin = d_k1;
            d_k1 = Dmin + Dmax - d_k2;
            f_k1 = function(-d_k1);
        }
        if (d_k1 > d_k2){
            std::swap(d_k1, d_k2);
            std::swap(f_k1, f_k2);
        }
        length = std::abs(Dmax - Dmin);
    }
    return -((Dmax + Dmin) / 2);
}


void print(const double noise)
{
    std::vector<Pair> p = rand(N, noise);
    double Cmin, Cmax, Dmin, Dmax;
    std::cout << "Cmin = " << Cmin << "\nCmax = " << Cmax << "\nDmin = " << Dmin <<
"\nDmax = " << Dmax << std::endl;
    double w1 = Metod_Dichotomy(a,b,c);
    double w0 = Method_golden(p, Cmin, Cmax);
    std::cout << "w1 = " << w1 <<std::endl;
    std::cout<< "w0 = " << w0 << std::endl;
}


int main() {


    std::cout << "Function y = x + 0"<<std::endl; // c=1, d=0
    std::cout << "Without noise"<<std::endl;
    print(0.0);
    std::cout << "With noise A = " << A <<std::endl;
    print(A);
    return 0;
}
```