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 12

Выполнил:
Мишков А.О.,
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 № 12.

В зависимости от варианта работы (табл. 1) найти линейную регрессию функции $y(x)$ (коэффициенты наиболее подходящей прямой c, d) по набору N ее 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 = 3, a = 0, b = 3, N = 10, A = 3.

Алгоритм поиска - золотое сечение, алгоритм поиска d — метод Фибоначчи.

3. Результат работы программы

```
bezshu
Cm = 1.82215e-307
C = 1.27344e+231
Dm = 1.7891e-307
D = 1.09974e-303
w1 = -1.27344e+231
w0 = 1.7891e-307
s shu
Cm = 1.82218e-307
C = 1.27344e+231
Dm = 1.09974e-303
D = 1.09974e-303
w1 = -1.27344e+231
w0 = 1.09974e-303
```

4. Выводы

В результате работы был реализован алгоритм поиска весовых коэффициентов функции на примере решения задачи регрессии экспериментальных данных.

5. Ответ на контрольный вопрос

1. Поясните суть метода наименьших квадратов.

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

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

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. Таким образом, решение сводится к нахождению экстремума функции двух переменных.

Приложение 1. Исходный код программы «Задача 1»

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

const double a = 0.0; const double b = 3.0; const double c = -1.0; const double
d = 3.0; const size_t N = 10; const double A = 3.0; const double step =
(double)((b - a) / (N - 1));

struct Point {
   double x;
   double y;
};

double f(const double& x) {
   return c * x + d;
}

std::vector<Point> random(const size_t N, const double shu) {
   std::vector<Point> points(N);
   std::random_device rd;
   std::mt19937 gen(rd());
   std::uniform_real_distribution<double> error(-0.5, 0.5);
   for (size_t i = 0; i < N; ++i) {
      points[i].x = a + i * step;
      points[i].y = f(a + i * step) + shu * error(gen);
   }
   return points;
}

std::vector<Point> edge(const double niz, const double vver,
   const size_t num, const double shu) {
   std::vector<Point> points(num);
   const double step = (vver - niz) / static_cast<double>(num - 1);
   for (size_t i = 0; i < num; ++i) {
      points[i].x = niz + i * step;
```

```
        points[i].y = f(points[i].x) - shu / 2 + rand() * 1. / RAND_MAX * (shu);
    }
    return points;
}

double error(const std::vector<Point>& pra, const double c, const double d) {
    double sum = 0.;
    for (auto point : pra) {
        sum += pow(point.y - (c * f(point.x)), 2);
    }
    return sum;
}

double golden_ratio(std::vector<Point>& p, double Cm, double C) {
    double l = std::abs(C - Cm);
    std::swap(Cm, C);
    Cm = std::fabs(Cm);
    C = std::fabs(C);
    const double e = 0.1;
    const double t = (std::sqrt(5) + 1) / 2;
    double c1 = Cm + (1 - 1 / t) * C;
    double c2 = Cm + C / t;
    double f1 = f(-c1);
    double f2 = f(-c2);
    while (l > e) {
        if (f1 < f2) {
            C = c2;
            c2 = Cm + C - c1;
            f2 = f(-c2);
        }
        else {
            Cm = c1;
            c1 = Cm + C - c2;
            f1 = f(-c1);
        }
        if (c1 > c2) {
            std::swap(c1, c2);
            std::swap(f1, f2);
        }
        l = std::abs(C - Cm);
    }
    return -((C + Cm) / 2);
}

int F(int f)
{
    if (f == 1) return 1;
    else if (f == 2) return 1;
    else if (f > 2)
        return (F(f - 1) + F(f - 2));
}

double Fibon(std::vector<Point>& p, double Dm, double D) {
    double a = Dm, b = D, x1, x2, y1, y2;
```

```
    int G = 10;
    x1 = a + (double)F(G - 2) / F(G) * (b - a);
    x2 = a + (double)F(G - 1) / F(G) * (b - a);
    y1 = error(p, x1, 0);
    y2 = error(p, x2, 0);
    for (int i = G; i >= 1; --i) {
        if (y1 > y2) {
            a = x1;
            x1 = x2;
            x2 = b - (x2 - a);
            y1 = y2;
            y2 = error(p, x2, 0);
        }
        else {
            b = x2;
            x2 = x1;
            x1 = a + (b - x2);
            y2 = y1;
            y1 = error(p, x1, 0);
        }
    }
    return (x1 + x2) / 2;
}

void print(const double noise)
{
    std::vector<Point> p = random(N, noise);
    double Cm, C, Dm, D;
    edge(Cm, C, Dm, D);
    std::cout << "Cm = " << Cm << "\nC = " << C << "\nDm = " << Dm <<
"\nD = " << D << std::endl;
    double w1 = golden_ratio(p, Cm, C);
    double w0 = Fibon(p, Dm, D);
    std::cout << "w1 = " << w1 << std::endl;
    std::cout << "w0 = " << w0 << std::endl;
}


int main() {
    std::cout << "bezshu" << std::endl;
    print(0.0);
    std::cout << "s shu " << A << std::endl;
    print(A);
    return 0;
}
```