|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 16

Выполнила: Частиков А.Д., студент группы ИУ8-31

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

г. Москва, 2020 г.

# 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 (табл. 1) 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 *ei =* Arnd(-0.5, 0.5). Выполнить расчет параметров *c, d* градиентным методом. Провести двумерный пассивный поиск оптимальных весовых коэффициентов нейронной сети (НС) регрессии.

Условие варианта: *c = -5; d = 5; a = 0; b = 4; N = 16; A = 2*; алгоритм поиска *c* – дихотомия, алгоритм поиска *d* – случайный.

# Графики

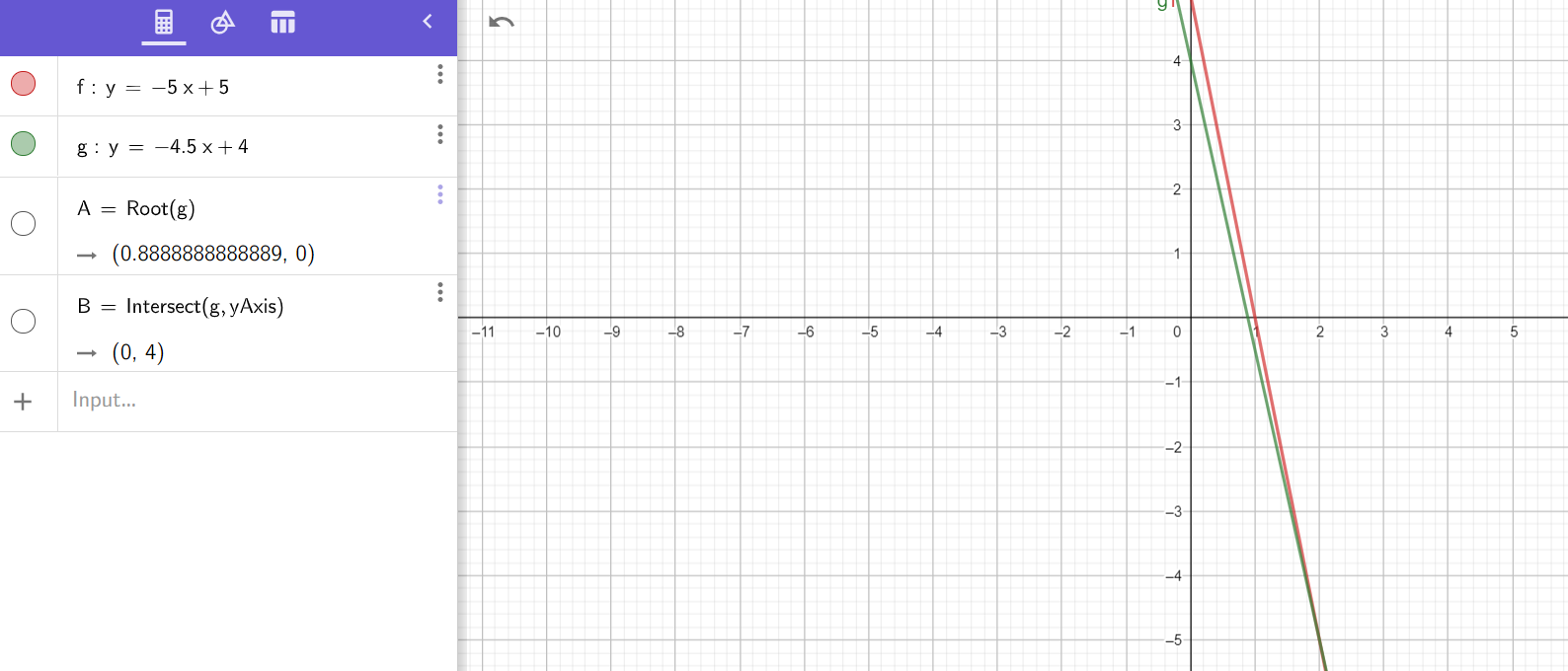


Рисунок 1 – Графики, построенные по результатам работы программы

Красный график – исходный график, заданный в варианте. Зеленый график построен при зашумлении A = 2.

Зашумленная функция:

| 0.235294 | 2.82603 |

| 0.470588 | 2.77423 |

| 0.705882 | 0.857197 |

| 0.941176 | 0.911599 |

| 1.17647 | -0.712334 |

| 1.41176 | -2.09908 |

| 1.64706 | -3.53471 |

| 1.88235 | -3.61984 |

| 2.11765 | -4.94256 |

| 2.35294 | -6.2715 |

| 2.58824 | -8.59296 |

| 2.82353 | -8.39976 |

| 3.05882 | -9.87311 |

| 3.29412 | -11.4435 |

| 3.52941 | -13.0391 |

| 3.76471 | -14.7936 |

Result :

w1 = -4.5044 ; w0 = 4

Результат работы программы:

1. С зашумлением: y = -0.45044x - 4

# Выводы

По результатам работы программы видно, что алгоритм успешно аппроксимирует зашумленные функции. Отклонения от исходного графика на заданном интервале очень малы.

# Приложение. Исходный код программы

//Variant 16

#include <random>

#include <vector>

#include <iostream>

#include <iomanip>

using namespace std;

struct neuron {

double x;

double y;

};

double lin\_func(const double c, const double d, const double x) {

return c \* x + d;

}

double random(const double a, const double b) {

return a + rand() \* 1. / RAND\_MAX \* (b - a);

}

double rand\_er(const double a, const double b, const double A) {

return A \* random(a, b);

}

std::vector<neuron> fill(const double c, const double d, const double A, const double a, const double b, const size\_t N) {

double delta = (b - a) / double(N + 1);

std::vector<neuron> n\_(N);

double current\_x = a;

for (auto& n : n\_) {

current\_x += delta;

n.x = current\_x;

n.y = lin\_func(c, d, n.x) + rand\_er(-0.5, 0.5, A);

}

return n\_;

}

double sum\_of\_squares(const double c, const double d, std::vector<neuron> n\_) {

double s = 0.0;

for (auto& n : n\_) {

double x = n.x;

double t = n.y;

double y = c \* x + d;

s += (y - t) \* (y - t);

return s;

}

}

double comp(const std::vector<neuron>& n\_, const double c, const double d) {

double sum = 0.;

for (auto n : n\_) {

sum += pow(n.y - (c \* n.x + d), 2);

}

return sum;

}

double random\_search(const std::vector<neuron>& n, double c) {

const double D\_MIN = 4;

const double D\_MAX = 6;

const size\_t iterations = 50;

double d = D\_MIN;

for (size\_t i = 0; i < iterations; ++i) {

double new\_d = random(D\_MIN, D\_MAX);

if (comp(n, c, new\_d) < comp(n, c, d)) {

d = new\_d;

}

}

return d;

}

double dichotomy(double c\_min, double c\_max, std::vector<neuron>& n) {

double eps = 0.01;

double delta = 0.001;

while ((c\_max - c\_min) > eps) {

double c1 = 0.5 \* (c\_min + c\_max) - delta;

double c2 = 0.5 \* (c\_min + c\_max) + delta;

double d1 = random\_search(n, c1);

double d2 = random\_search(n, c2);

if (sum\_of\_squares(c2, d2, n) > sum\_of\_squares(c1, d1, n)) {

c\_max = c2;

}

else {

c\_min = c1;

}

}

return (c\_min + c\_max) / 2;

}

void print(const std::vector<neuron>& n\_) {

for (const auto& n : n\_) {

std::cout << "| " << std::setw(10) << n.x

<< " | " << std::setw(10) << n.y << " |" << std::endl;

}

}

int main() {

const double a = 0;

const double b = 4;

const double c = -5;

const double d = 5;

const size\_t N = 16;

const double A = 2;

std::vector<neuron> n = fill(c, d, A, a, b, N);

double res\_1 = dichotomy(-6, -4, n);

double res\_2 = random\_search(n, res\_1);

print(n);

std::cout << "Result :" << std::endl;

std::cout << "w1 = " << res\_1 << " ; " << "w0 = " << res\_2 << std::endl;

return 0;

}

# Ответ на контрольный вопрос

1. Поясните суть метода наименьших квадратов.

Задача заключается в нахождении коэффициентов линейной зависимости, при которой функция двух переменных 𝑤1 и 𝑤0:

𝑁

𝐸2(𝑤1, 𝑤0) = ∑[𝑦(𝑥𝑖) − 𝑡𝑖]2

𝑖=1

→ 𝑚𝑖𝑛

𝑐,𝑑

Принимает наименьшее значение. То есть, при данных 𝑤1 и 𝑤0 сумма квадратов отклонений экспериментальных данных от исходной прямой будет наименьшей. В этом суть метода наименьших квадратов.