|  |  |
| --- | --- |
| 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 4

Выполнил: Велинецкий А. В.,

студент группы ИУ8-32

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. Постановка задачи**

В зависимости от варианта работы (табл. 4.1) 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 = A\*rnd(-0.5, 0.5). Выполнить расчет параметров c,d градиентным методом. Провести двумерный пассивный поиск оптимальных весовых коэффициентов нейронной сети (НС) регрессии.

**3. Ход работы**

1. Рассчитать равномерно распределенные на вашем интервале точки (N дискретных значений, N вводит пользователь программы),

2. Построить график заданной по варианту функций f (x) ;

3. При помощи генератора псевдослучайных чисел с указанными параметрами осуществить зашумление рассчитанных точек, далее использовать их при регрессии;

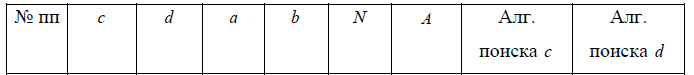
4. Отобразить эти зашумленные точки на графике;

5. Реализовать программно нейросетевую модель,

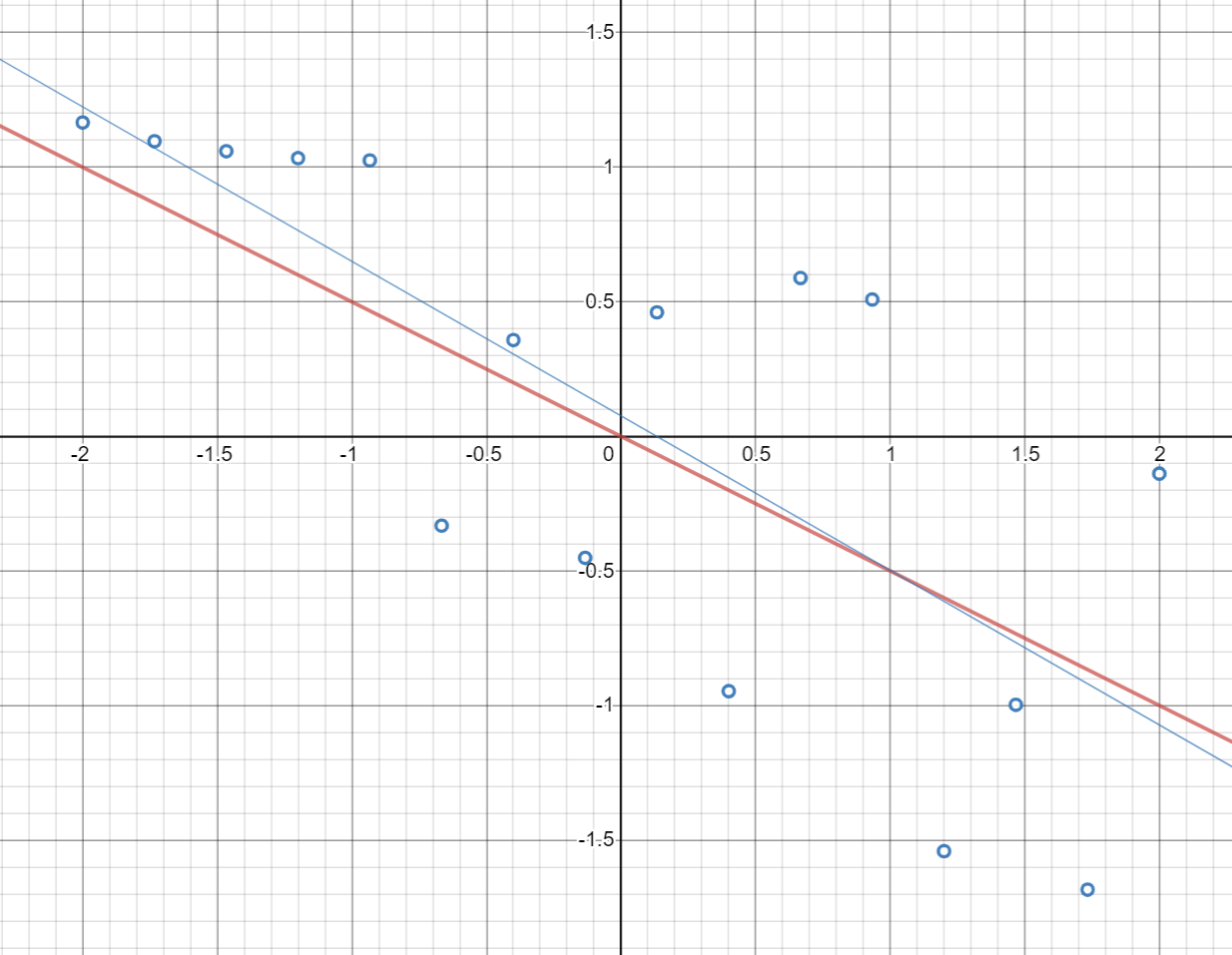
6. Используя указанные по варианту методы, осуществить поиск параметров c\*, d\*;

7. Построить график полученной приближенной регрессии;

8. Рассчитайте и выведите получившуюся погрешность приближения, используя рассчитанные при помощи найденных параметров c\*, d \* точки и исходные отсчеты.







**Рисунок 1** – график функций + зашумленные точки

 - график функции y(x) = -0,5x

 - зашумленные точки

 - график полученной приближенной регрессии

Ход работы программы по поиску экстремума приведен в приложении 2.

**4. Выводы**

В ходе проделанной работы я познакомился с простейшей нейронной сетью и реализовал алгоритма поиска ее весовых коэффициентов на примере решения задачи регрессии экспериментальных данных.

**5. Контрольные вопросы**

1. Поясните суть метода наименьших квадратов.

Суть метода состоит в том, чтобы минимизировать сумму квадратов ошибок для всех точек, выбрав соответствующие параметры. В данном случае, в модели линейной регрессии вида y = cx+d этими параметрами служат веса w1 = c, w2 = d.

**Приложение 1. Исходный код программы. Файл main.cpp**

#include <iostream>  
#include <random>  
#include <map>  
#include <algorithm>  
#include <ctime>  
#include <iomanip>  
  
using namespace std;  
  
const double a = -2.;  
const double b = 2.;  
const double c = -0.5;  
const double d = 0.;  
const int N = 16;  
const double A = 2.;  
const double step = (b - a) / (1. \* (N - 1));  
  
double myfun(const double &c\_coef, const double &d\_coef, const double &x) {  
 return c\_coef \* x + d\_coef;  
}  
  
double Random(double low\_limit, double high\_limit)  
{  
 return low\_limit + (1.\*rand()/RAND\_MAX)\*(high\_limit - low\_limit); // (1.\*rand()/RAND\_MAX)-генерация числа от 0 до 1  
}  
  
auto XandYMapWithNoise(const double &low\_limit, const double &noise) {  
 map<double, double> XandYmap;  
 for (int i = 0; i < N; ++i) {  
 double x = low\_limit + i \* step;  
 double y = myfun(c, d, low\_limit + i \* step) + noise \* Random(-0.5, 0.5);  
 XandYmap[x] = y;  
 }  
 return XandYmap;  
}  
  
double SumErrors(const map<double, double> &XandYmap, const double &w1, const double &w0) {  
 double sum = 0;  
 for (auto p : XandYmap) {  
 sum += (myfun(w1, w0, p.first) - p.second) \* (myfun(w1, w0, p.first) - p.second);  
 }  
 return sum;  
}  
  
int N\_count(double q, double P) {  
 return (ceil(log(1. - P) / log(1. - q)));  
}  
  
double PassiveSearchForW1(const map<double, double> &XandYmap, const double &low\_limit, const double &high\_limit) { //search for w1  
 int i = 1;  
 const float eps = 0.01;  
 double current\_error = 1;  
 double x;  
 double minx = 0;  
 double w0 = 0;  
 while (current\_error > eps){  
 double min = 1.7976931348623158\*pow(10, 308);  
 for (int k = 1; k < i + 1; k++) {  
 x = ((high\_limit - low\_limit) \* k / (i + 1)) + low\_limit;  
 if (SumErrors(XandYmap,x, w0) < min){  
 min = SumErrors(XandYmap,x, w0);  
 minx = x;  
 }  
 }  
 current\_error = (high\_limit - low\_limit) / (i + 1);  
 i++;  
 }  
 return minx;  
}  
  
double RandomSearchFowW0(const map<double, double> &XandYmap, const double &low\_limit, const double &high\_limit, const double &w1){  
 int z;  
 double MAX\_DOUBLE = 1.7976931348623158\*pow(10, 308);  
 double x;  
 double q = 0.005;  
 double p = 0.9;  
 double min;  
 double minx;  
 for (int i = 0; i < 20; i++) {  
 for (int j = 0; j < 10; j++) {  
 z = N\_count(q, p);  
 min = MAX\_DOUBLE;  
 for (auto k = 0; k < z; k++) {  
 x = Random(low\_limit, high\_limit);  
 if (SumErrors(XandYmap, w1, x) < min){  
 min = SumErrors(XandYmap, w1, x);  
 minx = x;  
 }  
 }  
 }  
 }  
 return minx;  
}  
  
void Print(const double &noise) {  
 map<double, double> XandYmap = XandYMapWithNoise(a, noise);  
 size\_t i = 1;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"  
 "| Номер | Значение | Значение |\n"  
 "| точки | аргумента | функции |\n"  
 "| (i) | (x) | (y) |\n"  
 "-------------------------------------" << endl;  
 for (auto p : XandYmap) {  
 cout << "|" << setw(11) << i << "|" << setw(11) << p.first << "|"  
 << setw(11) << p.second << "|" << endl;  
 ++i;  
 }  
 cout << "-------------------------------------" << endl;  
  
 double Cmin = -1, Cmax = 0;  
 double Dmin = -1, Dmax = 1;  
 cout << "Cmin = " << Cmin << "\nCmax = " << Cmax << "\nDmin = " << Dmin << "\nDmax = " << Dmax << "\n";  
 double w1 = PassiveSearchForW1(XandYmap, Cmin, Cmax);  
 cout << "c = " << w1 << "\n";  
 double w0 = RandomSearchFowW0(XandYmap, Dmin, Dmax, w1);  
 cout << "d = " << w0 << "\n";  
 cout << "Error = " << SumErrors(XandYmap, w1, w0) << "\n";  
}  
  
int main() {  
 srand(time(NULL));  
 cout << "Результаты для функции y = " << c << " \* x + " << d <<  
 " на интервале [ " << a << " , " << b << " ]:\n";  
 Print(0.);  
 cout <<"Результаты для функции y = " << c << " \* x + " << d << "с шумом A = " << A <<  
 " на интервале [ " << a << " , " << b << " ]:\n";  
 Print(A);  
}

}

**Приложение 2. Ход работы программы.**

