МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Кафедра «Информационная безопасность» (ИУ8)

Лабораторная работа № 5

Двумерный поиск для подбора коэффициентов простейшей нейронной сети на примере решения задачи линейной регрессии экспериментальных данных

Выполнила: Броцкий К.А.

студент группы ИУ8-32

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.

Постановка задачи

Вариант 1:

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

Табл. 1. Исходные данные

№ пп	c	d	a	b	N	A	Алг. поиска c	Алг. поиска d
1	3	1	–2	2	16	2	пассивный	золотое сечение

График функции

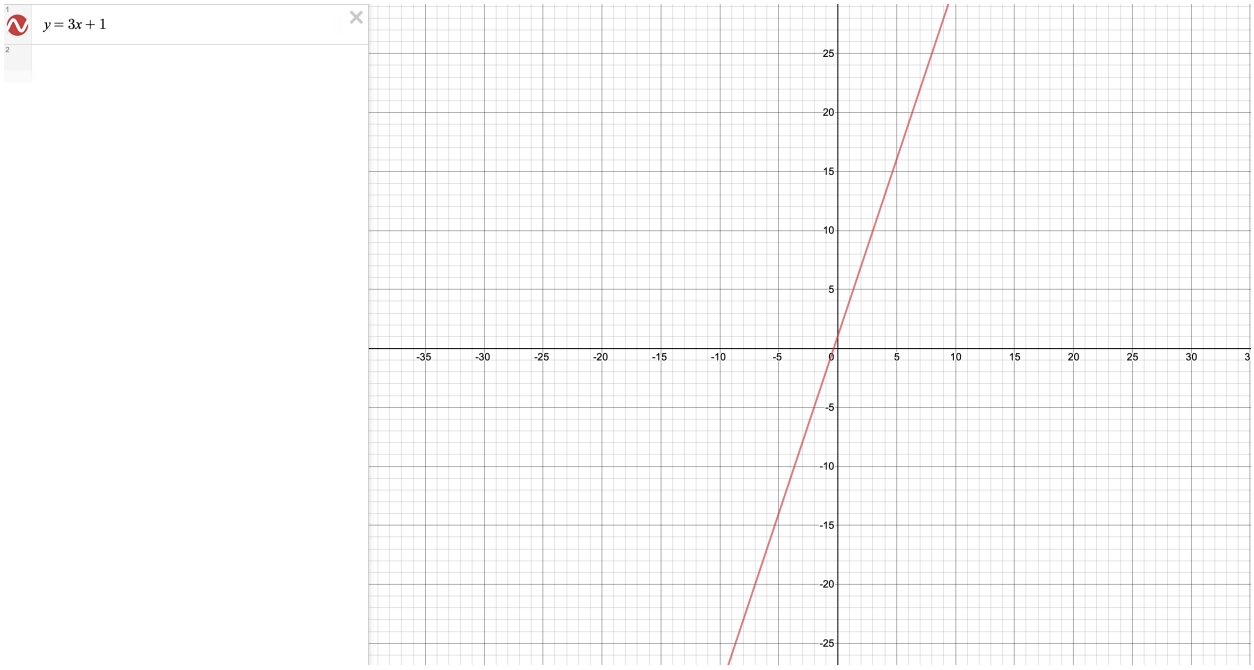
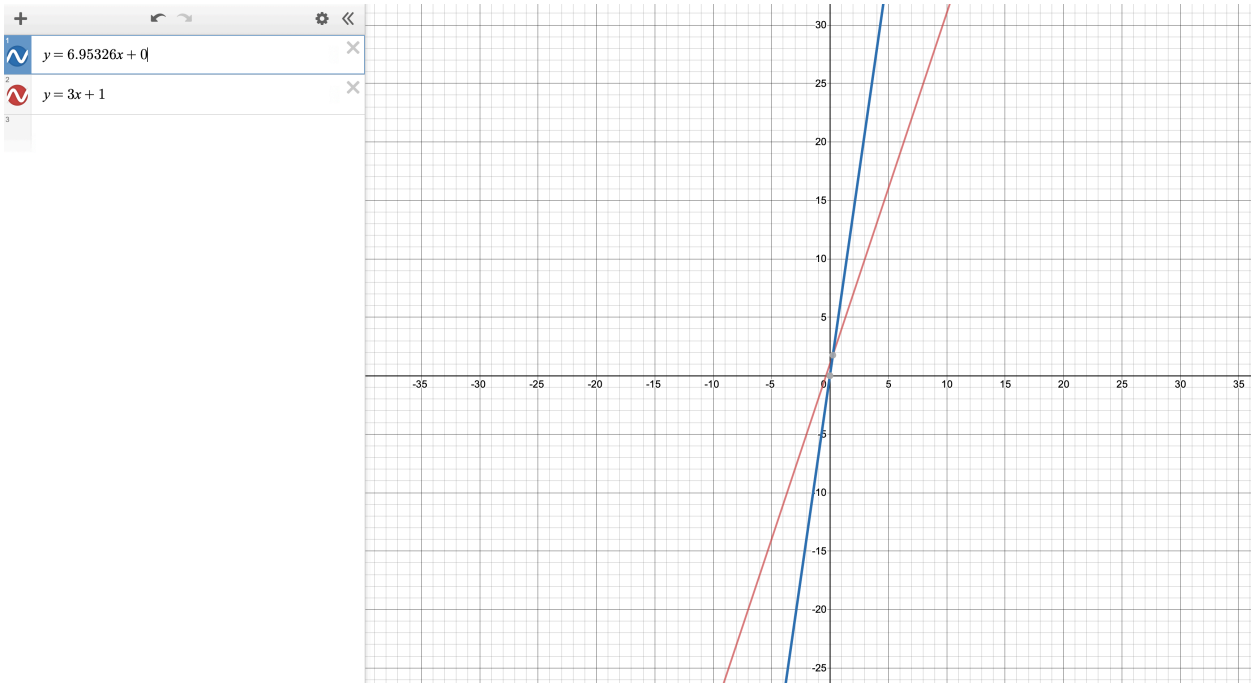


*Рис.1. График функции y=3*x+1*

Красный график - исходный, синий - по результатам выполнения программы

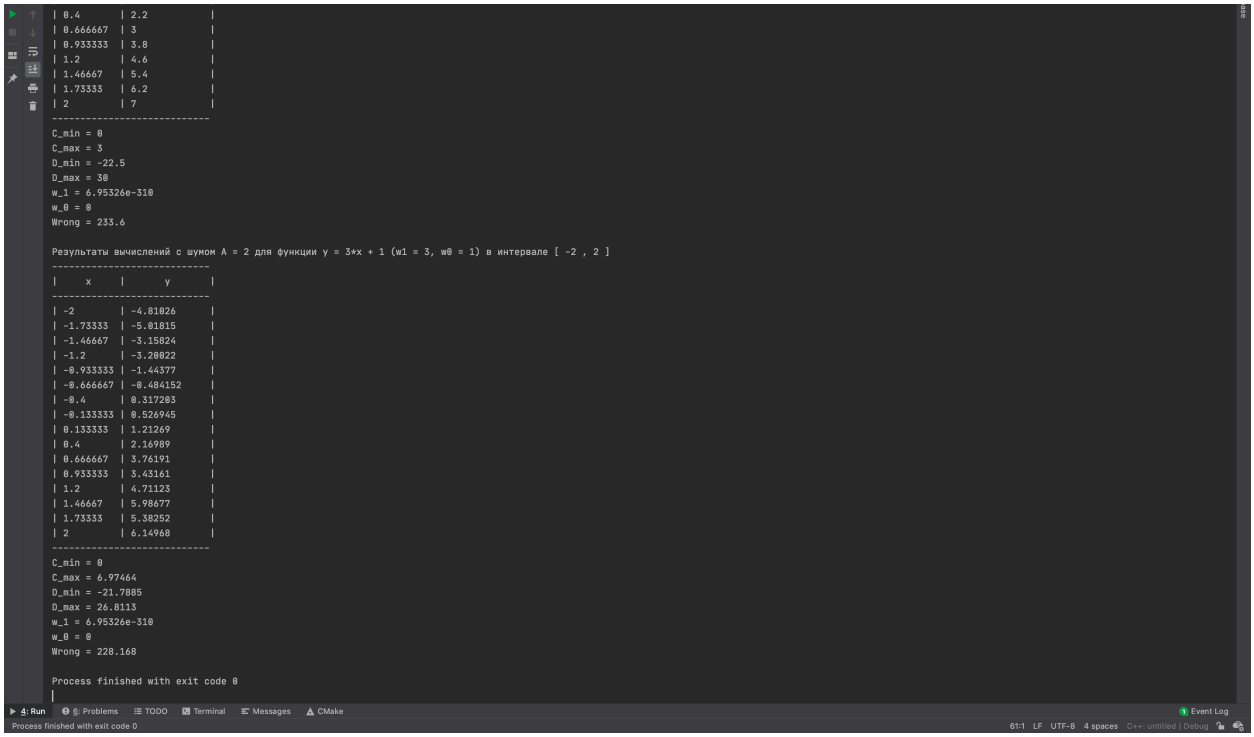


Скриншот консоли

```
/Users/kirillbrockij/TSISA/untitled/cmake-build-debug/untitled
Результаты вычислений с шумом A=0 для функции y = 3*x + 1 (w1 = 3 w0 = 1) в интервале [ -2 , 2 ]
---------------------------
|     x     |       y       |
---------------------------
| -2        | -5            |
| -1.73333  | -4.2          |
| -1.46667  | -3.4          |
| -1.2      | -2.6          |
| -0.933333 | -1.8          |
| -0.666667 | -1            |
| -0.4      | -0.2          |
| -0.133333 | 0.6           |
| 0.133333  | 1.4           |
| 0.4       | 2.2           |
| 0.666667  | 3             |
| 0.933333  | 3.8           |
| 1.2       | 4.6           |
| 1.46667   | 5.4           |
| 1.73333   | 6.2           |
| 2         | 7             |
---------------------------
C_min = 0
C_max = 3
D_min = -22.5
D_max = 30
w_1 = 6.95326e-310
w_0 = 0
Wrong = 233.6

Результаты вычислений с шумом A = 2 для функции y = 3*x + 1 (w1 = 3, w0 = 1) в интервале [ -2 , 2 ]
---------------------------
|     x     |       y       |
---------------------------
| -2        | -4.81026      |
| -1.73333  | -5.01815      |
| -1.46667  | -3.15824      |
| -1.2      | -3.20022      |
| -0.933333 | -1.44377      |
| -0.666667 | -0.484152     |
| -0.4      | 0.317203      |
| -0.133333 | 0.526945      |
| 0.133333  | 1.21269       |
| 0.4       | 2.16989       |
| 0.666667  | 3.76191       |
| 0.933333  | 3.43161       |
| 1.2       | 4.71123       |

4: Run  6: Problems  TODO  Terminal  Messages  CMake  Event Log
Process finished with exit code 0
61:1  LF  UTF-8  4 spaces  C++: untitled | Debug
```



```
| 0.4       | 2.2           |
| 0.666667  | 3             |
| 0.933333  | 3.8           |
| 1.2       | 4.6           |
| 1.46667   | 5.4           |
| 1.73333   | 6.2           |
| 2         | 7             |
---------------------------
C_min = 0
C_max = 3
D_min = -22.5
D_max = 30
w_1 = 6.95326e-310
w_0 = 0
Wrong = 233.6

Результаты вычислений с шумом A = 2 для функции y = 3*x + 1 (w1 = 3, w0 = 1) в интервале [ -2 , 2 ]
---------------------------
|     x     |       y       |
---------------------------
| -2        | -4.81026      |
| -1.73333  | -5.01815      |
| -1.46667  | -3.15824      |
| -1.2      | -3.20022      |
| -0.933333 | -1.44377      |
| -0.666667 | -0.484152     |
| -0.4      | 0.317203      |
| -0.133333 | 0.526945      |
| 0.133333  | 1.21269       |
| 0.4       | 2.16989       |
| 0.666667  | 3.76191       |
| 0.933333  | 3.43161       |
| 1.2       | 4.71123       |
| 1.46667   | 5.98677       |
| 1.73333   | 5.38252       |
| 2         | 6.14968       |
---------------------------
C_min = 0
C_max = 6.97464
D_min = -21.7885
D_max = 26.8113
w_1 = 6.95326e-310
w_0 = 0
Wrong = 228.168

Process finished with exit code 0

4: Run  6: Problems  TODO  Terminal  Messages  CMake  Event Log
Process finished with exit code 0
61:1  LF  UTF-8  4 spaces  C++: untitled | Debug
```

Рис.2. Скриншот консоли

Листинг программы с реализацией алгоритмов на C++

```
#include <iostream>
#include <random>
#include <vector>
#include <map>
#include <algorithm>
#include <cmath>
#include <iomanip>

const int N=16;
const double a = -2.0, b = 2.0, c = 3.0, d = 1.0, A = 2.0, count = (b - a) / (N - 1);
size_t i=0;
double x=0, y=0, sum=0;
size_t number=0;

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

std::map<double, double> Create_of_Numbers(const double &a, const double &b,
const size_t N, const double &shum) {
   std::map<double, double> points;
   std::random_device ranD;
   std::mt19937 gen(ranD());
   std::uniform_real_distribution<double> error(-0.5, 0.5);
   for (i = 0; i < N; ++i) {
      x = a + i * count;
      y = Function_Line(c, d, a + i * count) + shum * error(gen);
      points.insert(std::make_pair(x, y));
   }
   return points;
}

double Sum_of_wrong(const std::map<double, double> &points, const double
&w1, const double &w0) {
   sum = 0;
   for (const auto &p : points) {
      sum = sum + (Function_Line(w1, w0, p.first) - p.second) *
(Function_Line(w1, w0, p.first) - p.second);
   }
   return sum;
}
```

```
void Interval_for_min(const std::map<double, double> &points, double &c_min,
double &c_max, double &d_min, double &d_max) {
    static_cast<void>(c_min = 0), c_max = 0;
    auto future_y = ++points.begin();
    for (const auto &before_y : points) {
        if (future_y != points.end()) {
            if (future_y->second - before_y.second > c_max) {
                c_max = (future_y->second - before_y.second) / count;
            }
            if (future_y->second - before_y.second < c_min) {
                c_min = (future_y->second - before_y.second) / count;
            }
        } else {
            break;
        }
        ++future_y;
    }
    if (points.begin()->second < points.rbegin()->second) {
        c_min = 0;
    }
    if (points.begin()->second > points.rbegin()->second) {
        c_max = 0;
    }
    d_max = (points.begin()->second < points.rbegin()->second ? points.rbegin()-
>second : points.begin()->second) +
            A * 0.5;
    d_min = (points.begin()->second > points.rbegin()->second ? points.rbegin()-
>second : points.begin()->second) -
            A * 0.5;
    d_min = d_min / count;
    d_max = d_max / count;
}

double Algorithm_of_golden(const std::map<double, double> &points, double
lower, double upper) { //search for w1
    double goldenNumber = (1 + sqrt(5)) / 2;
    double x_left = lower + (1 - 1 / goldenNumber) * upper;
    double x_right = lower + upper / goldenNumber;
    double epsilon = 0.01;
    double w0 = 0;
    while (upper - lower > epsilon) {
        if (Sum_of_wrong(points, x_left, w0) < Sum_of_wrong(points, x_right, w0))
{
            upper = x_right;
            x_right = lower + upper - x_left;
```

```
        } else {
            lower = x_left;
            x_left = lower + upper - x_right;
        }
        if (x_left > x_right)
            std::swap(x_left, x_right);
    }
    return (x_left + x_right) / 2;
}

double Algorithm_of_passive(const std::map<double, double> &points, const
double &lower, const double &upper,
                            const double &w1) {
    number = 1;
    double epsilond = 0.08;
    double delta = (upper - lower) / (number + 1);
    double x_for_min_of_y = 0.0;
    while (delta > epsilond) {
        std::vector<double> VectorOfYk;
        delta = (upper - lower) / (number + 1);
        for (uint k = 1; k <= number; ++k) {
            VectorOfYk.push_back(Sum_of_wrong(points, w1, delta * k + lower));
        }
        unsigned long int kForMinY = std::min_element(VectorOfYk.begin(),
VectorOfYk.end()) - VectorOfYk.begin() + 1;
        x_for_min_of_y = (upper - lower) / (number + 1) * kForMinY + lower;
        ++number;
    }
    return x_for_min_of_y;
}


void Print(const double &shum) {
    std::map<double, double> map_of_points = Create_of_Numbers(a, b, N, shum);
    i=0;
    std::cout << "----------------------------" << std::endl;
    std::cout << "| " << std::setw(9) << std::left << "    x" << " | " << std::setw(13)
<< std::left << "      y" << " | " << std::endl;
    std::cout << "----------------------------" << std::endl;
    for (const auto &p : map_of_points) {
        std:: cout << "| " << std::setw(9) << std::left <<p.first << " | " << std::setw(13)
<< std::left << p.second << std::setw(9) <<" | "<< "\n";
        ++i;
    }
    std::cout << "----------------------------" << std::endl;
```

```
    double c_min, c_max, d_min, d_max;
    double w1 = Algorithm_of_golden(map_of_points, d_min, d_max);
    double w0 = Algorithm_of_passive(map_of_points, c_min, c_max, w1);
    Interval_for_min(map_of_points, c_min, c_max, d_min, d_max);
    std::cout << "C_min = " << c_min << "\nC_max = " << c_max << "\nD_min = "
<< d_min << "\nD_max = " << d_max << std::endl;
    std::cout << "w_1 = " << w1 << std::endl;
    std::cout << "w_0 = " << w0 << std::endl;
    std::cout << "Wrong = " << Sum_of_wrong(map_of_points, w1, w0) <<
std::endl;
}

int main() {
    std::cout << "Результаты вычислений с шумом A=0 для функции y = 3*x + 1
(w1 = 3 w0 = 1) в интервале [ " << a << " , "
        << b << " ]"<< std::endl;
    Print(0);
    std::cout<<std::endl;
    std::cout << "Результаты вычислений с шумом A = " << A << " для функции
y = 3*x + 1 (w1 = 3, w0 = 1) в интервале [ " << a << " , "
        << b << " ]"<<std::endl;
    Print(A);
}
```

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:*

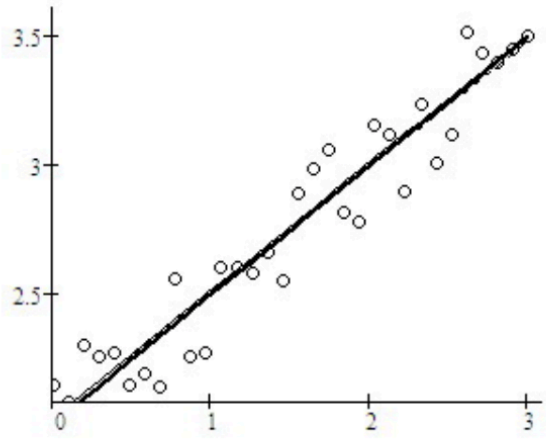
$$F(a,b) = \sum_{i=1}^{n} (y_i - (a\,x_i + b)^2)$$

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

Вывод

В процессе выполнения данной 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.