Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №3**

**По дисциплине МиАПР за 3 семестр**  
**Тема: «Нелинейные ИНС в задачах прогнозирования»**

**Выполнил:**

Студент группы ПО-6(1)  
 2-го курса

Мартынович Даниил

**Проверил:**

Михняев А. Л.

Брест 2021

Цель работы: Изучить обучение и функционирование нелинейной ИНС при решении задач прогнозирования.

***Задание.***

1. Написать на любом ЯВУ программу моделирования прогнозирующей нелинейной ИНС. Для тестирования использовать функцию

** .

**Вариант 10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | a | b | с | d | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 0.2 | 0.4 | 0.09 | 0.4 | 6 | 2 |

Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

Файл Header.h

#pragma once

#ifndef LAB

#define LAB

#include <iostream>

using namespace std;

class INNeuron {

public:

double Ves;

void random\_Ves() {

Ves = ((rand() % 10) \* 0.1) - 1;

}

void izm\_ves(double a, double y, double t, double x) {

Ves -= a \* (y - t) \* x;

}

void set\_Ves(int Ves) {

this->Ves = Ves;

}

double get\_Ves() {

return Ves;

}

};

class StealthNeuron {

public:

double Ves;

double Sum, znach;

void random\_Ves() {

Ves = ((rand() % 10) \* 0.1) - 1;

}

void izm\_ves(double a, double y1, double gamma, double y2) {//изменение весов

Ves -= a \* gamma \* y1 \* (1 - y1) \* y2;

}

};

#endif // !1

Файл Source.cpp

#include "Header.h"

int main() {

setlocale(LC\_ALL, "ru");

system("color f0");

const int n = 30, n\_in\_neuron = 6, n\_hi\_neuron = 2;//Кол-во входов ИНС, Кол-во НЭ в скрытом слое

double a = 0.2, b = 0.4, c = 0.09, d = 0.4;//Значения по варианту

double Em = 0.00001, A = 0.1, Emax;//Среднеквадратичная ошибка, Скорость обучения, Максимальня ошибка

double y\_etalon[n + 15], y, Tx[n\_hi\_neuron], Ty = 2 \* ((rand() % 10) \* 0.1) - 1, Error, Error\_i[n - n\_in\_neuron];//Эталонные значения, прогнозируемые значения, Пороги,

for (int i = 0; i < n; i++) {//заполнение эталонными значениями

double x = 0.1 \* i;

y\_etalon[i] = a \* cos(b \* x) + c \* sin(d \* x);

}

INNeuron W[n\_hi\_neuron \* n\_in\_neuron];//веса

StealthNeuron Sig[n\_hi\_neuron];//скрытый слой

for (int i = 0; i < n\_in\_neuron \* n\_hi\_neuron; i++) {//заполняем веса

W[i].random\_Ves();

}

for (int i = 0; i < n\_hi\_neuron; i++) {//скрытый слой

Sig[i].random\_Ves();

Sig[i].Sum = 0;

Tx[i] = ((rand() % 10) \* 0.1) - 1;

}

do {

y = 0;

Emax = 0;

for (int count = 0; count < n - n\_in\_neuron; count++) {

y = 0;

for (int i = 0; i < n\_hi\_neuron; i++) {

for (int j = 0; j < n\_in\_neuron; j++) {

Sig[i].Sum += W[j].Ves \* y\_etalon[count + j];

}

Sig[i].Sum -= Tx[i];

Sig[i].znach = (1 / (1 + exp(-Sig[i].Sum)));

Sig[i].Sum = 0;

}

for (int i = 0; i < n\_hi\_neuron; i++) {

y += Sig[i].znach \* Sig[i].Ves;

}

y -= Ty;

Error = y - y\_etalon[count + n\_in\_neuron];//отклонение

Error\_i[count] = Error;

for (int i = 0; i < n\_hi\_neuron; i++) {

Sig[i].Ves -= A \* Sig[i].znach \* Error;

}

Ty += A \* Error;

for (int i = 0; i < n\_hi\_neuron; i++) {

for (int j = 0; j < n\_in\_neuron; j++)

W[(i \* 10) + j].Ves -= A \* y\_etalon[count + j] \* (Sig[i].Ves \* Error \* Sig[i].znach \* (1 - Sig[i].znach));

Tx[i] += A \* (Sig[i].Ves \* Error \* Sig[i].znach \* (1 - Sig[i].znach));

}

}

for (int i = 0; i < n - n\_in\_neuron; i++) {

Emax += (pow(Error\_i[i], 2.0) \* 0.5);

}

} while (Emax > Em);

cout << "Эталон" << "\t\t\t\t" << "Прогноз" << "\t\t\t\tОтклонение" << endl;

for (int Feature = 0; Feature < 15; Feature++) {

y = 0;

for (int i = 0; i < n\_hi\_neuron; i++) {

for (int j = 0; j < n\_in\_neuron; j++) {

Sig[i].Sum += W[j].Ves \* y\_etalon[n - n\_in\_neuron + Feature + j];

}

Sig[i].Sum -= Tx[i];

Sig[i].znach = (1 / (1 + exp(-Sig[i].Sum)));

Sig[i].Sum = 0;

}

for (int i = 0; i < n\_hi\_neuron; i++) {

y += Sig[i].znach \* Sig[i].Ves;

}

y -= Ty;

y\_etalon[n + Feature] = y;

double x = 0.1 \* ((double)Feature + (double)n);

cout << a \* cos(b \* x) + c \* sin(d \* x) << "\t\t\t";

cout << y\_etalon[n + Feature] << "\t\t\t";

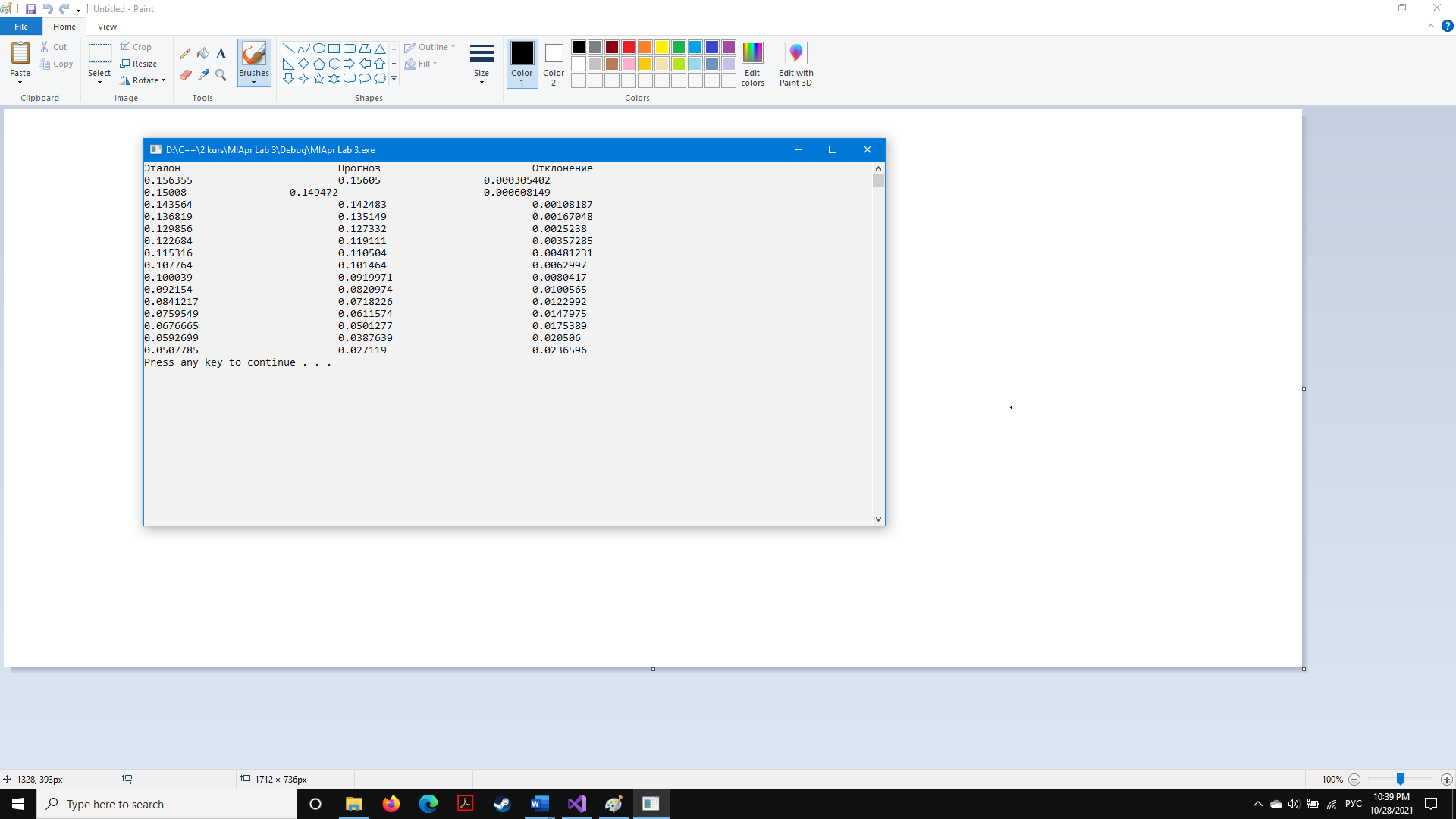
cout << a \* cos(b \* x) + c \* sin(d \* x) - y\_etalon[n + Feature] << endl;

}

system("pause");

return 0;

}



Вывод: Изучил обучение и функционирование нелинейной ИНС при решении задач прогнозирования.