Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть. Правило Видроу-Хоффа»

Выполнил:

Студент 2 курса

Группы ПО-4(2)

Синяк Д.А.

Проверил:

Крощенко А.А.

2020

**Лабораторная работа №1**

**Линейная искусственная нейронная сеть. Правило Видроу-Хоффа**

**Цель работы:** *изучить обучение и функционирование линейной ИНС при решении задач прогнозирования.*

**Вариант 8**

Задание:

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

y = a\*sin(bx) + d

a = 4, b = 7, d = 0.2, кол-во входов ИНС = 4.

Обучение и прогнозирование производить на 30 и 15 значениях соответственно табулируя функцию с шагом 0.1. Скорость обучения выбирается студентом самостоятельно, для чего моделирование проводится несколько раз для разных α. Результаты оцениваются по двум критериям - скорости обучения и минимальной достигнутой ошибке. Необходимо заметить, что эти критерии в общем случае являются взаимоисключающими, и оптимальные значения для каждого критерия достигаются при разных α.

**Код программы:**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

system("color f0");

int a = 4,

b = 7,

num\_enteries = 4, //количество входов ИНС

n = 30, //количесвто значений, на которых производится обучение

num\_predicated\_values = 15; //количесвто значений, на которых производится прогнозирование

double d = 0.2,

Em = 0.01, //минимальная среднеквадратичная ошибка сети

E, //суммарная среднеквадратичная ошибка

T = 1; //порог нейронной сети

double\* W = new double[num\_enteries]; //весовые коэффициенты

for (int i = 0; i < num\_enteries; i++)

{ //задаем случайным образом весовые коэффициенты

W[i] = static\_cast <double> (rand()) / (static\_cast <double>(RAND\_MAX / 10));

}

double\* reference\_value\_y = new double[n + num\_predicated\_values]; //эталонные значения y

for (int i = 0; i < n + num\_predicated\_values; i++) //вычисляем эталонные значения

{

double step = 0.1, //шаг

x = step \* i;

reference\_value\_y[i] = a \* sin(b \* x) + d;

}

do

{

double y1, //выходное значение нейронной сети

A = 0.0005; //скорость обучения

E = 0;

for (int i = 0; i < n - num\_enteries; i++)

{

y1 = 0;

for (int j = 0; j < num\_enteries; j++)//векторы выходной активности сети

{

y1 += W[j] \* reference\_value\_y[i + j];

}

y1 -= T;

for (int j = 0; j < num\_enteries; j++) //изменение весовых коэффициентов

{

W[j] -= A \* (y1 - reference\_value\_y[i + num\_enteries]) \* reference\_value\_y[i + j];

}

T += A \* (y1 - reference\_value\_y[i + num\_enteries]); //изменение порога нейронной сети

E += 0.5 \* pow(y1 - reference\_value\_y[i + num\_enteries], 2); //расчет суммарной среднеквадратичной ошибки

}

} while (E > Em);

cout << "Результаты обучения" << endl;

cout << setw(27) << left << "Эталонные значения" << setw(23) << left << "Полученные значения" << "Отклонение" << endl;

double\* predicated\_values = new double[n + num\_predicated\_values];

for (int i = 0; i < n; i++)

{

predicated\_values[i] = 0;

for (int j = 0; j < num\_enteries; j++)

{

predicated\_values[i] += W[j] \* reference\_value\_y[j + i - num\_enteries]; //получаемые значения в результате обучения

}

predicated\_values[i] -= T;

cout << "y[" << i << "] = " << setw(20) << left << reference\_value\_y[i] << setw(23) << left;

cout << predicated\_values[i] << reference\_value\_y[i] - predicated\_values[i] << endl;

}

cout << endl << "Результаты прогнозирования" << endl;

cout << setw(28) << left << "Эталонные значения" << setw(23) << left << "Полученные значения" << "Отклонение" << endl;

for (int i = 0; i < num\_predicated\_values; i++) {

predicated\_values[i + n] = 0;

for (int j = 0; j < num\_enteries; j++) {

//прогнозируемые значения

predicated\_values[i + n] += W[j] \* reference\_value\_y[n - num\_enteries + j + i];

}

predicated\_values[i + n] += T;

cout << "y[" << n + i << "] = " << setw(20) << left << reference\_value\_y[i + n] << setw(23) << left;

cout << predicated\_values[i + n] << reference\_value\_y[i + n] - predicated\_values[i + n] << endl;

}

delete[]reference\_value\_y;

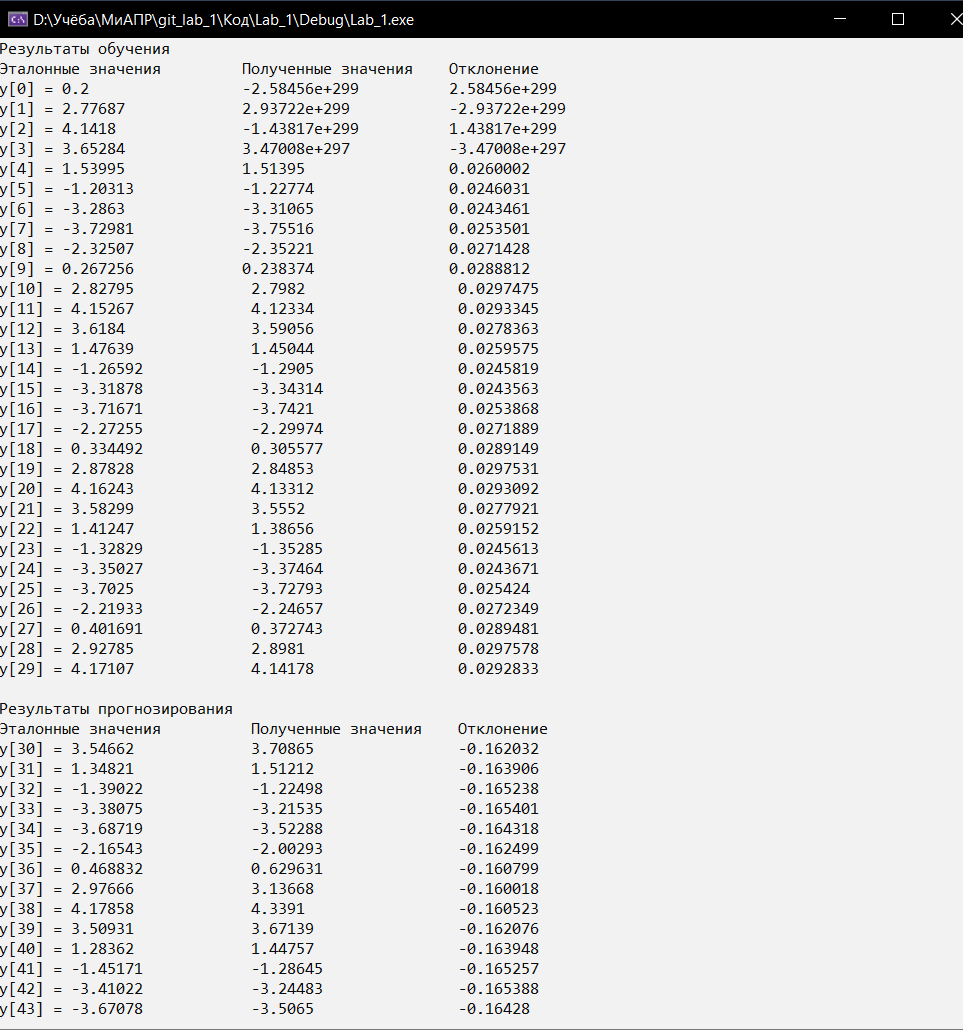
delete[]predicated\_values;

delete[]W;

system("pause");

return 0;

}

****



**Вывод:** *изучил обучение и функционирование линейной ИНС при решении задач прогнозирования.*