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»

Выполнила:

Студент 2 курса

Группы ПО-4(1)

Грицак Александра

Вячеславовна

Проверил:

Крощенко А.А.

Брест, 2020

Лабораторная работа №1

Линейная искусственная нейронная сеть. Правило Видроу-Хоффа

Цель работы: изучить обучение и функционирование линейной ИНС при решении

задач прогнозирования.

Вариант 7

Задание:

Написать на любом ЯВУ программу моделирования прогнозирующей линейной

ИНС. Для тестирования использовать функцию

y = a\*sin(bx) + d

a = 3, b = 6, d = 0.1, кол-во входов ИНС = 3.

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 альфа.

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 альфа.

Нейронная сеть представляет собой последовательность связанных нейронов. К нейрону поступают входящие сигналы, каждому из которых присвоен определенный вес. Сигнал умножается на свой вес, значения суммируются, и получается единое число, которое получает активационная функция. На выходе она принимает решение, транслировать ли сигнал дальше. Самая сложная задача в работе с нейросетью – грамотно подобрать коэффициенты к нейронам. Для этого используется обучение – процесс нахождения корректных весов для нейросети. От того, как именно обучат нейросеть, будут зависеть ее решения.

Правило обучения Видроу-Хоффа

Используется для обучения нейронной сети, состоящей из распределительных нейронов и одного выходного нейрона, который имеет линейную функцию активации. Правило обучения Видроу-Хоффа базируется на методе градиентного спуска в пространстве весовых коэффициентов и порогов нейронной сети.

Алгоритм обучения, в основе которого лежит дельта правило состоит из следующих шагов:

1. Задается скорость обучения и минимальная среднеквадратичная ошибка

сети, которой необходимо достичь в процессе обучения.

2. Случайным образом инициализируются весовые коэффициенты и порог нейронной сети.

3. Подаются входные образы на нейронную сеть и вычисляются векторы выходной

активности сети.

4. Производится изменение весовых коэффициентов и порога нейронной сети согласно

выражениям (1.7) и (1.8).

5. Алгоритм продолжается до тех пор, пока суммарная среднеквадратичная ошибка сети не

станет меньше заданной, т. е. .

Текст программы:

#include <iostream>

#include <iomanip>

#include <ctime>

using namespace std;

int main() {

setlocale(0, "");

int a = 3,

b = 6,

enteries = 3, //входы ИНС

n = 30, //количество значений для обучения

values = 15; //количество значений для прогнозирования

double d = 0.1,

Em = 0.05, //минимальная среднеквадратичная ошибка сети

E, //суммарная среднеквадратичная ошибка сети

T = 1; //порог НС

double\* W = new double[enteries]; //весовые коэффициенты (3)

//srand(time(NULL)); //для разного рандома

for (int i = 0; i < enteries; i++) { //генерирует весовые коэффициенты

W[i] = (double)(rand()) / RAND\_MAX; //от 0 до 1

cout << "W[" << i << "] = " << W[i] << endl; //вывод весовых коэффициентов

}

cout << endl;

double\* etalon\_values = new double[n + values]; //эталонные значения y

for (int i = 0; i < n + values; i++) { //вычисляем эталонные значения

double step = 0.1; //шаг

double x = step \* i;

etalon\_values[i] = a \* sin(b \* x) + d; //формула для проверки

}

int era = 0; //для индексов

while (1) {

double y1; //выходное значение нейронной сети

double speed = 0.01; //скорость обучения

E = 0; //ошибка

for (int i = 0; i < n - enteries; i++) {

y1 = 0;

for (int j = 0; j < enteries; j++) { //векторы выходной активности сети

y1 += W[j] \* etalon\_values[j + i];

}

y1 -= T;

for (int j = 0; j < enteries; j++) { //изменение весовых коэффициентов

W[j] -= speed \* (y1 - etalon\_values[i + enteries]) \* etalon\_values[i + j];

}

T += speed \* (y1 - etalon\_values[i + enteries]); //изменение порога нейронной сети

E += 0.5 \* pow(y1 - etalon\_values[i + enteries], 2); //расчет суммарной среднеквадратичной ошибки

era++;

}

cout << era << " | " << E << endl;

if (E < Em) break;

} //далее сеть обучена

cout << endl;

cout << "РЕЗУЛЬТАТЫ ОБУЧЕНИЯ" << endl;

cout << setw(27) << right << "Эталонные значения" << setw(23) << right << "Полученные значения";

cout << setw(23) << right << "Отклонение" << endl;

double\* prognoz\_values = new double[n + values];

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

prognoz\_values[i] = 0;

for (int j = 0; j < enteries; j++) {

prognoz\_values[i] += W[j] \* etalon\_values[j + i]; //получаемые значения в результате обучения

}

prognoz\_values[i] -= T;

cout << "y[" << i + 1 << "] = " << setw(20) << right << etalon\_values[i + enteries] << setw(23) << right;

cout << prognoz\_values[i] << setw(23) << right << etalon\_values[i + enteries] - prognoz\_values[i] << endl;

}

cout << endl << "РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ" << endl;

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

for (int i = 0; i < values; i++) {

prognoz\_values[i + n] = 0;

for (int j = 0; j < enteries; j++) {

//прогнозируемые значения

prognoz\_values[i + n] += W[j] \* etalon\_values[n - enteries + j + i];

}

prognoz\_values[i + n] -= T;

cout << "y[" << n + i + 1 << "] = " << setw(20) << right << etalon\_values[i + n] << setw(23) << right;

cout << prognoz\_values[i + n] << setw(23) << right << etalon\_values[i + n] - prognoz\_values[i + n] << endl;

}

delete[]etalon\_values;

delete[]prognoz\_values;

delete[]W;

system("pause");

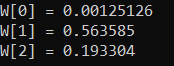
return 0;

}

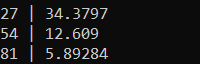
Результаты тестирования программы:

1. **шаг обучения 0.1**

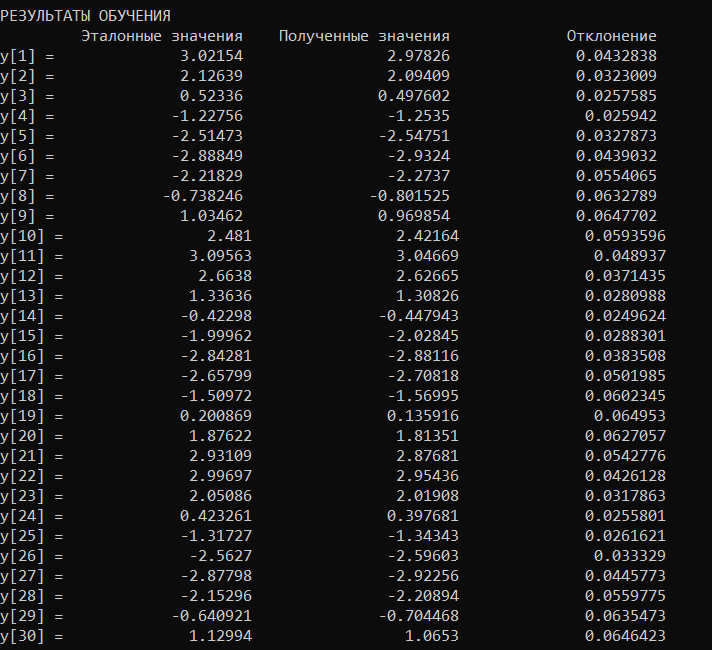
* Вывод весовых коэффициентов



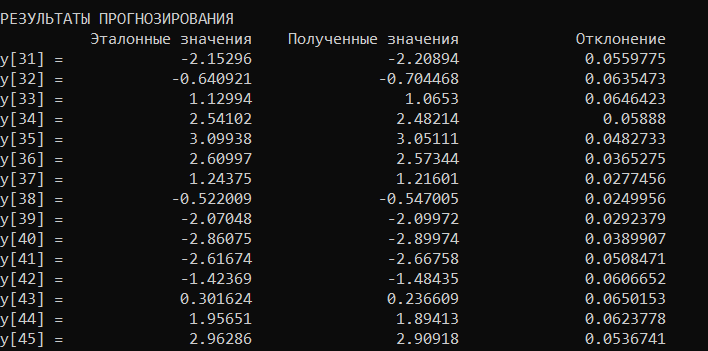
* Вывод итерации и суммарной квадратичной ошибки



* Вывод результатов обучения

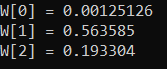


* Вывод результатов прогнозирования

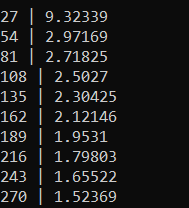


1. **шаг обучения 0.01**

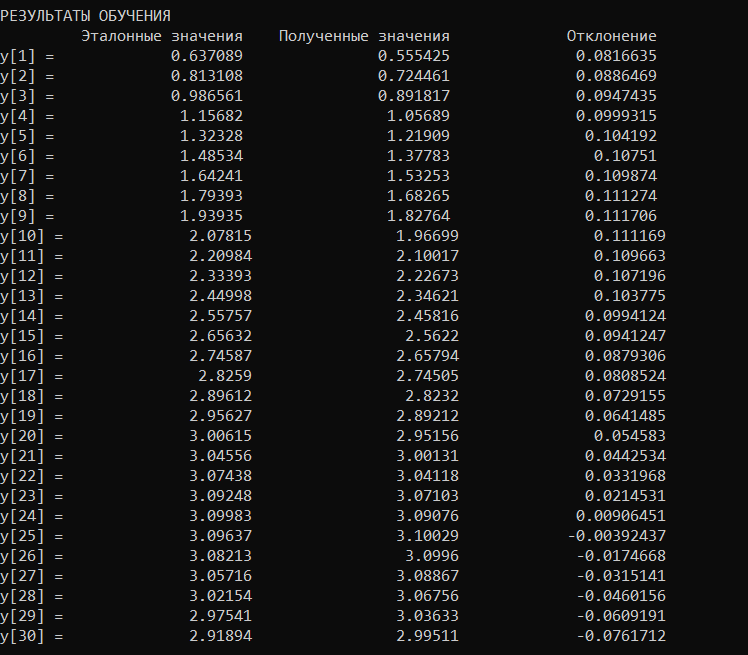
* Вывод весовых коэффициентов



* Вывод итерации и суммарной квадратичной ошибки



* Вывод результатов обучения



* Вывод результатов прогнозирования

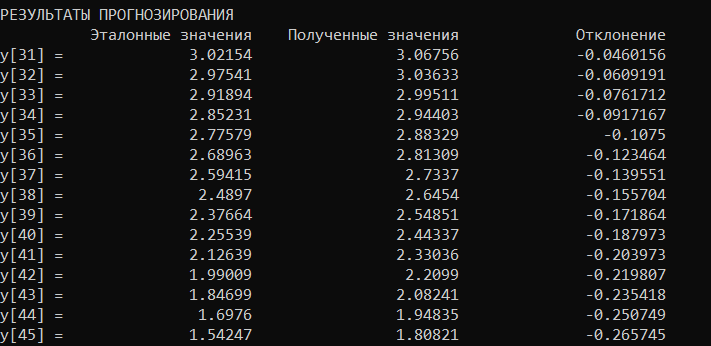
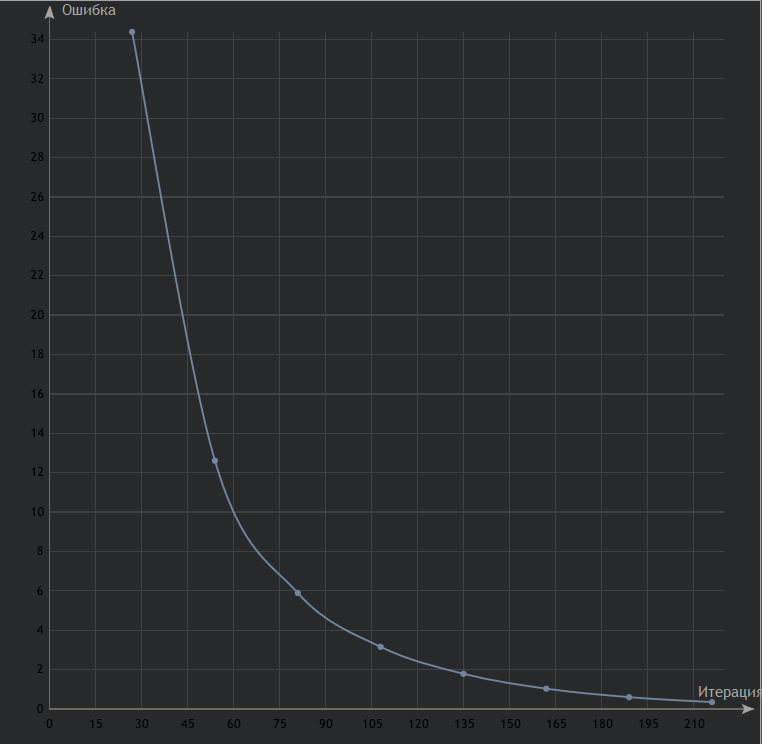


График изменения ошибки в зависимости от итерации:



Вывод:

Изучила обучение и функционирование линейной ИНС при решении задач

прогнозирования.

В алгоритме Видроу-Хоффа существует проблема выбора значения шага обучения.

Если коэффициент слишком мал, то процесс обучения является очень длительным. В случае, когда шаг обучения большой, процесс обучения может оказаться расходящимся, то есть не привести к решению задачи. Таким образом сходимость алгоритма обучения не избавляет от разумного выбора значения шага обучения.