Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть. Адаптивный шаг обучения»

Выполнила:

Студент 2 курса

Группы ПО-4(1)

Грицак Александра

Вячеславовна

Проверил:

Крощенко А.А.

Брест, 2020

Лабораторная работа №2

Линейная искусственная нейронная сеть. Адаптивный шаг обучения

Цель работы: изучить обучение и функционирование линейной ИНС с применением адаптивного шага.

Вариант 7

Задание:

Модифицировать программу из лабораторной работы №1, используя правило адаптивного шага обучения. Произвести исследование получившейся модели ИНС на задачах прогнозирования, согласно варианту лабораторной работы №1.

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.

Для ускорения процедуры обучения градиентного спуска, вместо постоянного шага обучения можно использовать адаптивный шаг обучения. Назовем адаптивным шагом обучения такой шаг, который целенаправленно выбирается на каждом этапе алгоритма таким образом, чтобы минимизировать среднеквадратичную ошибку сети.

Для нахождения адаптивного шага обучения будем использовать метод наискорейшего спуска. Он заключается в том, чтобы на каждом шаге обучения нейронной сети необходимо выбирать такую скорость обучения, чтобы минимизировать среднеквадратичную ошибку.

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;

double temp = 0.0;

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

temp += pow(etalon\_values[i + j], 2);

}

speed = 1 / (1 + temp); //адаптивный шаг

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;

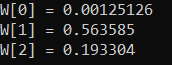
system("pause");

return 0;

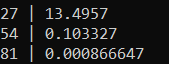
}

Результаты тестирования программы:

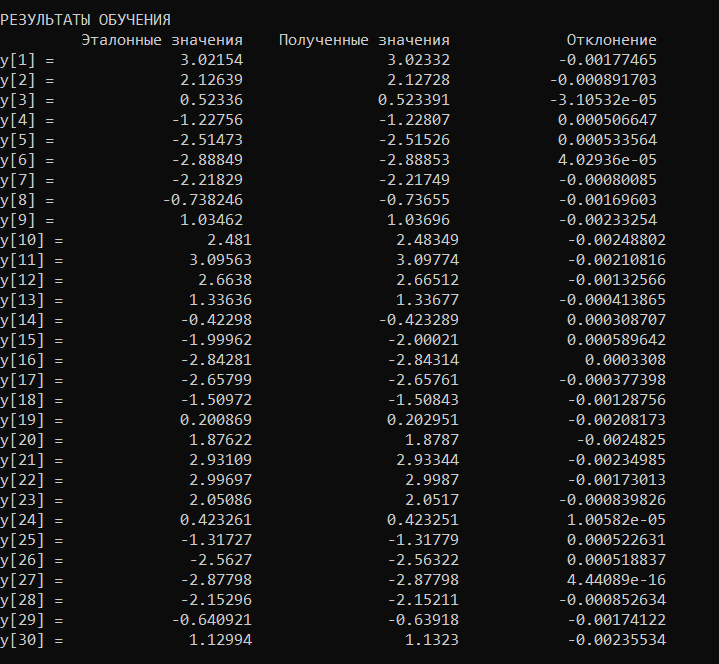
* Вывод весовых коэффициентов



* Вывод итерации и суммарной квадратичной ошибки



* Вывод результатов обучения



* Вывод результатов прогнозирования

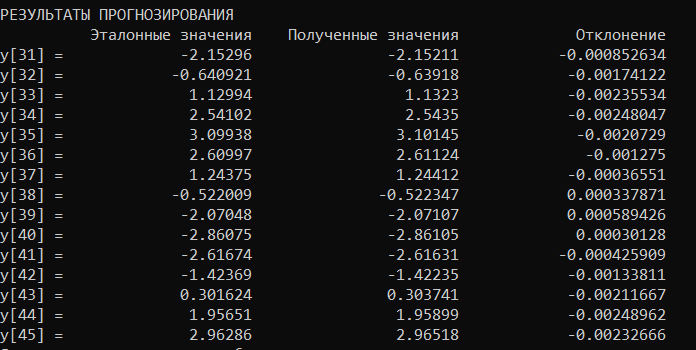
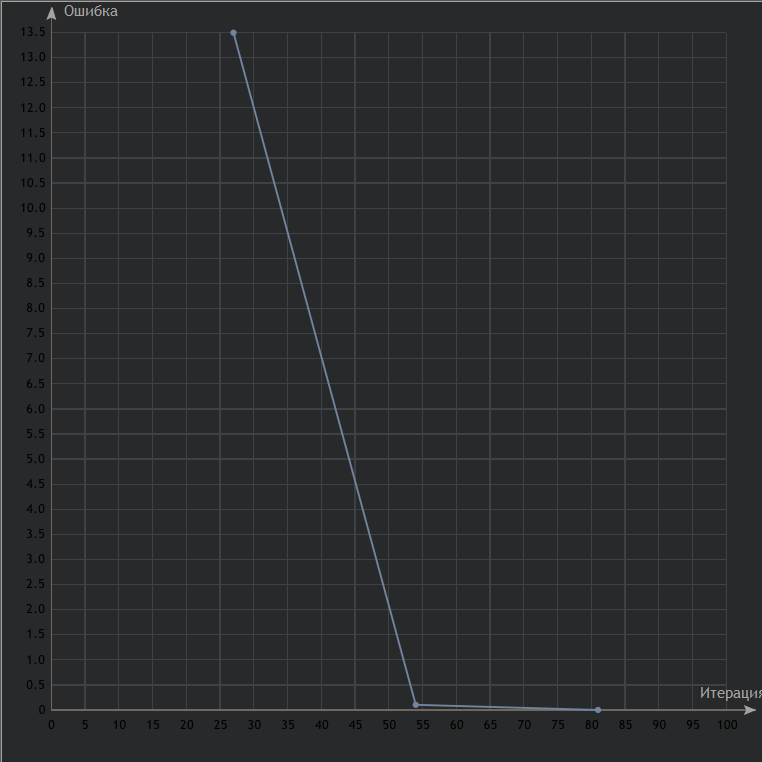


График изменения ошибки в зависимости от итерации:



Вывод:

Изучила обучение и функционирование линейной ИНС с использованием адаптивного шага.

При использовании адаптивного шага процедура обучения градиентного спуска проходит быстрее.