Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах прогнозирования. Адаптивный шаг»

Выполнила:

Студент 2 курса

Группы ПО-4(1)

Грицак Александра

Вячеславовна

Проверил:

Крощенко А.А.

Брест, 2020

Лабораторная работа №4

Нелинейные ИНС в задачах прогнозирования. Адаптивный шаг

Цель работы: изучить обучение и функционирование нелинейной ИНС при решении задач прогнозирования с использованием адаптивного шага обучения.

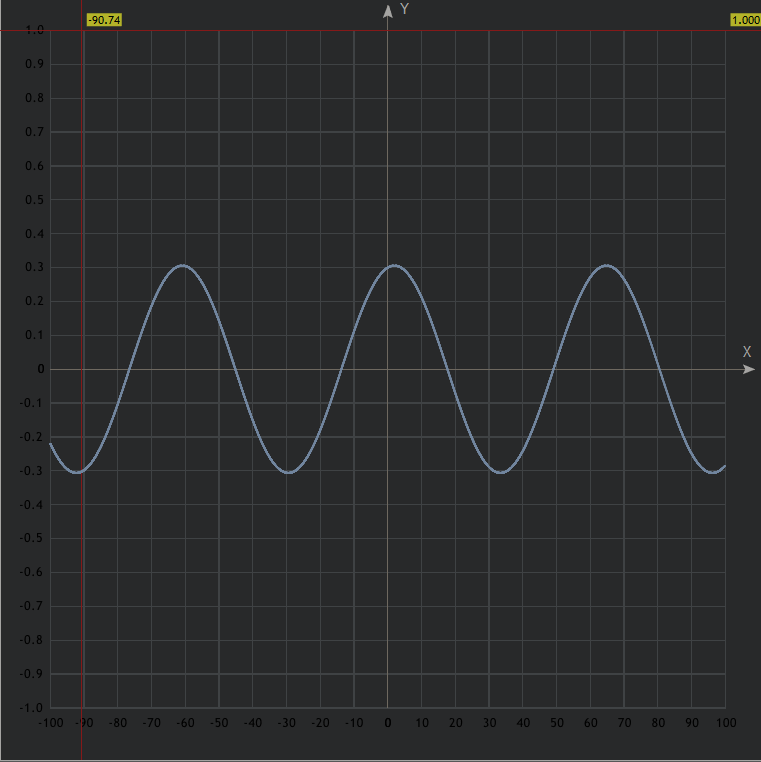
Вариант 7

Задание:

Спрогнозировать нелинейный временной ряд, применяя параметры лабораторной работы №3. При этом необходимо использовать алгоритм обучения многослойной ИНС с адаптивным шагом. Для тестирования использовать функцию **

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | a | b | с | d | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |
| 7 | 0.3 | 0.1 | 0.06 | 0.1 | 6 | 2 |

График функции 0.3 \* cos(0.1\* x) + 0.06 \* sin(0.1 \* x):



Текст программы:

#include <iostream>

#include <iomanip>

using namespace std;

double Func(double x);

double Sigmoid(double x);

double\* Hidden(double x, double Wes1[2][6], double T[2]);

double output(double x, double Wes1[2][6], double Wes2[2], double T[2 + 1]);

double Adapt(double Wes2[], double error, double output, double hiddens[]);

int main() {

setlocale(0, "");

double Wes1[2][6], Wes2[2], T[2 + 1],

ethelon\_value, current,

Alpha = 0.7, Alpha2, x = 4,

Emin = 0.00001, Emax = 0;

int eras = 0;

for (int i = 0; i < 2; i++) {

for (int k = 0; k < 6; k++) {

Wes1[i][k] = ((double)rand() / RAND\_MAX);

}

Wes2[i] = ((double)rand() / RAND\_MAX);

T[i] = ((double)rand() / RAND\_MAX);

}

T[4] = ((double)rand() / RAND\_MAX);

do {

Emax = 0;

for (int q = 0; q < 700; q++) {

current = output(x, Wes1, Wes2, T);

ethelon\_value = Func(x + 6 \* 0.1);

double error = current - ethelon\_value;

double\* hiddens = Hidden(x, Wes1, T);

Alpha2 = Adapt(Wes2, error, current, hiddens);

for (int j = 0; j < 2; j++) {

Wes2[j] -= Alpha \* error \* hiddens[j];

}

T[4] += Alpha \* error;

for (int k = 0; k < 2; k++) {

for (int i = 0; i < 6; i++) {

Wes1[k][i] -= Alpha2 \* Func(x + i \* 0.1) \* hiddens[k] \* (1 - hiddens[k]) \* Wes2[k] \* error;

}

T[k] += Alpha2 \* hiddens[k] \* (1 - hiddens[k]) \* Wes2[k] \* error;

}

x += 0.1;

Emax += pow(error, 2);

}

Emax /= 2;

eras++;

cout << "\rError: " << Emax;

} while (Emax > Emin);

cout << endl;

cout << "Эпохи: " << eras << endl;

cout << setw(27) << left << "Эталонs" << setw(29) << left << "Получ. знач." << setw(20) << left << "Отклонение" << endl;

for (int i = 0; i < 20; i++) {

double result = output(x, Wes1, Wes2, T),

ethelon\_value = Func(x + 6 \* 0.1);

cout << setw(27) << left << ethelon\_value << setw(27) << left << result << setw(30) << (result - ethelon\_value) \* (result - ethelon\_value) << endl;

x += 0.1;

}

system("pause");

return 0;

}

double Func(double x) {

double a = 0.3, b = 0.1, c = 0.06, d = 0.1;

return a \* cos(b \* x) + c \* sin(d \* x);

}

double Sigmoid(double x) {

return 1 / (1 + pow(2, -x));

}

double\* Hidden(double x, double Wes1[2][6], double T[2]) {

double\* result\_value = new double[2];

for (int i = 0; i < 2; i++) {

result\_value[i] = 0;

}

double entrances[6];

for (int k = 0; k < 6; k++, x += 0.1) {

entrances[k] = Func(x);

}

for (int i = 0; i < 2; i++) {

for (int k = 0; k < 6; k++) {

result\_value[i] += entrances[k] \* Wes1[i][k];

}

result\_value[i] -= T[i];

result\_value[i] = Sigmoid(result\_value[i]);

}

return result\_value;

}

double output(double x, double Wes1[2][6], double Wes2[2], double T[2 + 1]) {

double result = 0;

double\* hidden\_neuron = Hidden(x, Wes1, T);

for (int j = 0; j < 2; j++) {

result += hidden\_neuron[j] \* Wes2[j];

}

result -= T[4];

return result;

}

double Adapt(double Wes2[], double error, double output, double hiddens[]) {

double Alpha2 = 0, A = 0, B = 0;

for (int i = 0; i < 2; i++) {

A += pow(error \* Wes2[i] \* (1 - hiddens[i]) \* hiddens[i], 2) \* hiddens[i] \* (1 - hiddens[i]);

B += pow(error \* Wes2[i] \* (1 - hiddens[i]) \* hiddens[i], 2) \* hiddens[i] \* hiddens[i] \* (1 - hiddens[i]) \* (1 - hiddens[i]);

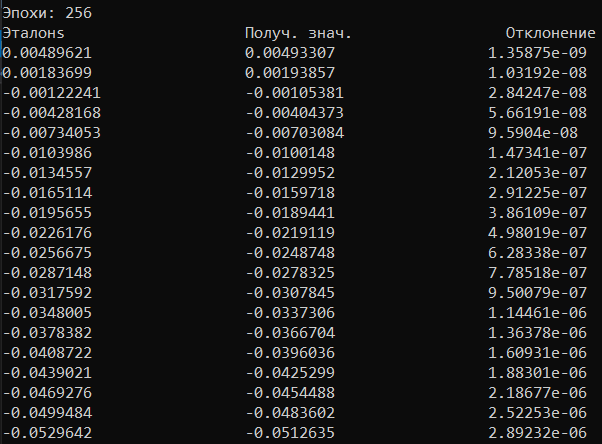
}

Alpha2 = 4 \* A / (B \* (1 + output \* output));

return Alpha2;

}

Результаты тестирования программы:



Вывод:

Изучила обучение и функционирование нелинейной ИНС при решении задач прогнозирования с использованием адаптивного шага обучения.

**Примечание:**

При использовании адаптивного шага процедура обучения градиентного спуска проходит быстрее. Так же при увеличении количества элементов из обучающей выборки количество эпох уменьшается.