Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По ИМОД

Тема: «Нелинейные искусственные

нейронные сети в задачах прогнозирования»

Выполнил:

студент 4-го курса

группы АС-56

*Соротокин С.В.*

Проверил:

*Савицкий Ю. В*.

Брест 2022

**Цель работы:** изучить обучение и функционирование нелинейной искусственной нейронной сети (ИНС) при решении задач прогнозирования.

Вариант 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № 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 |

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

**.

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

Текст программы

Random random = new Random();

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

int input = 6, hidden = 2;

double step = 0.1;

int trainCount = 30, testCount = 15;

List<double> arrForTable = new();

List<double> ethalonArr = new();

List<List<double>> hiddenArrW = new();

for (int x = 0; x < trainCount; x++) { ethalonArr.Add(a \* Math.Cos(b \* x \* step) + c \* Math.Sin(d \* x \* step)); }

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

{

List<double> row = new();

for (int j = 0; j <= input; j++)

{

row.Add(random.NextDouble());

}

hiddenArrW.Add(row);

}

double gammaOut = 0;

List<double> hiddenArrT = new();

List<double> hiddenArrY = new();

List<double> gammaHiden = new();

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

{

hiddenArrT.Add(0);

hiddenArrY.Add(0);

gammaHiden.Add(0);

}

List<double> arrOutW = new();

for (int j = 0; j < hidden; j++)

arrOutW.Add(random.NextDouble());

double outT = random.NextDouble();

double epoch = 0, y = 0, err = 1;

double minErr = 1e-6;

while (err > minErr)

{

err = 0;

for (int i = 0; i < trainCount - input; i++)

{

hiddenArrY = HiddenS(hiddenArrY, hiddenArrW, ethalonArr, hiddenArrT, input, hidden, i);

for (int j = 0; j < hidden; j++)

hiddenArrY[j] = HiddenY(hiddenArrY[j]);

y = Y(hiddenArrY, arrOutW, outT, hidden);

gammaOut = y - ethalonArr[i + input];

for (int j = 0; j < hidden; j++)

{

gammaHiden[j] = gammaOut \* y \* (1 - y) \* arrOutW[j];

arrOutW[j] -= step \* gammaOut \* hiddenArrY[j];

hiddenArrT[j] += step \* gammaHiden[j] \* hiddenArrY[j] \* (1 - hiddenArrY[j]);

}

outT += step \* gammaOut;

hiddenArrW = WHiddenChanging(hiddenArrW, gammaHiden, hiddenArrY, ethalonArr, input, hidden, i, step);

err += (Math.Pow((y - ethalonArr[i + input]), 2)) / 2;

}

epoch++;

err /= trainCount - input;

if (epoch % 5000 == 0)

{

Console.WriteLine("Эпоха: "+ epoch+ " | Ошибка: "+ err);

arrForTable.Add(err);

}

}

Console.WriteLine("Результаты тренировки");

Console.WriteLine("Эталонные значения\t\t" + "Полученные значения\t\t" + "Разница\t\t");

for (int i = 0; i < trainCount - input; i++)

{

hiddenArrY = HiddenS(hiddenArrY, hiddenArrW, ethalonArr, hiddenArrT, input, hidden, i);

for (int j = 0; j < hidden; j++)

hiddenArrY[j] = HiddenY(hiddenArrY[j]);

y = Y(hiddenArrY, arrOutW, outT, hidden);

Console.WriteLine(ethalonArr[i + input].ToString() + "\t\t" + y + "\t\t" + (ethalonArr[i + input] - y) + "\t\t");

}

List<double> ethalonTestArr = new();

for (int i = testCount - input; i < testCount + trainCount; i++)

{

ethalonTestArr.Add(a \* Math.Cos(b \* i \* step) + c \* Math.Sin(d \* i \* step));

}

Console.WriteLine("Результаты тренировки");

Console.WriteLine("Эталонные значения\t\t" + "Полученные значения\t\t" + "Разница\t\t");

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

{

hiddenArrY = HiddenS(hiddenArrY, hiddenArrW, ethalonArr, hiddenArrT, input, hidden, i);

for (int j = 0; j < hidden; j++)

hiddenArrY[j] = HiddenY(hiddenArrY[j]);

y = Y(hiddenArrY, arrOutW, outT, hidden);

Console.WriteLine(ethalonArr[i + input].ToString() + "\t\t" + y + "\t\t" + (ethalonArr[i + input] - y) + "\t\t");

}

List<double> HiddenS(List<double> hiddenArrY, List<List<double>> hiddenArrW, List<double> ethalonArr, List<double> hiddenArrT, int input, int hiden, int iter)

{

for (int i = 0; i < hiddenArrY.Count(); i++)

hiddenArrY[i] = 0;

for (int j = 0; j < hiden; j++)

{

for (int k = 0; k < input; k++)

{

hiddenArrY[j] += hiddenArrW[j][k] \* ethalonArr[iter + j];

}

hiddenArrY[j] -= hiddenArrT[j];

}

return hiddenArrY;

}

double HiddenY(double hiddenS)

{

return 1 / 1 + Math.Exp(-Math.Abs(hiddenS));

}

double Y(List<double> hiddenArrY, List<double> arrOutW, double outT, int hidden)

{

double y = 0;

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

{

y += hiddenArrY[i] \* arrOutW[i];

}

return y - outT;

}

List<List<double>> WHiddenChanging(List<List<double>> hiddenArrW, List<double> gammaHiden, List<double> hiddenArrY, List<double> ethalonArr, int input, int hidden, int iter, double step)

{

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

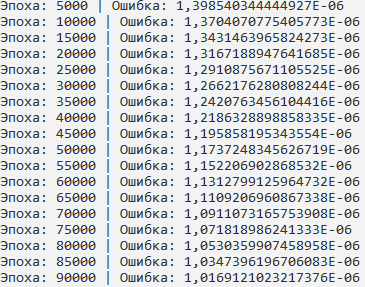
for (int j = 0; j < input; j++)

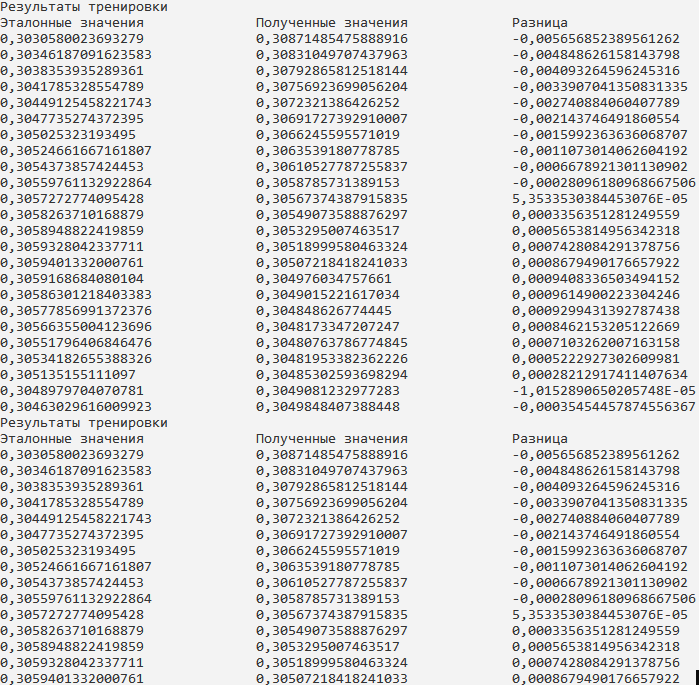
hiddenArrW[i][j] -= step \* gammaHiden[i] \* hiddenArrY[i] \* (1 - hiddenArrY[i]) \* ethalonArr[j + iter];

return hiddenArrW;

}

Вывод программы:





**Вывод:** изучил обучение и функционирование нелинейной искусственной нейронной сети (ИНС) при решении задач прогнозирования.