Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №4

За 3 семестр

По дисциплине «Методы и алгоритмы принятия решений»

Тема: «Нелинейные ИНС в задачах прогнозирования»

Выполнил: студент 2 курса

Группы ПО-4(2)

Юрьев В. А.

Проверил: Крощенко А.А.

Брест 2020

**Лабораторная работа №4**

**Нелинейные ИНС в задачах прогнозирования**

**Цель:** Изучить обучение и функционирование нелинейной ИНС при решении задач прогнозирования.

**Вариант 11**

**Задание**

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | a | b | с | d | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |
| 11 | 0.3 | 0.5 | 0.05 | 0.5 | 8 | 3 |

**Код программы:**

#include <iostream>

#include <math.h>

#include <iomanip>

#define input\_layer 8

#define hidden\_layer 3

using namespace std;

double sigmoid(double x)

{

return 1 / (1 + pow(2.7, -x));

}

double function(double x)

{

double a = 0.1, b = 0.1, c = 0.05, d = 0.1;

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

}

double\* hidden(double x, double w1[hidden\_layer][input\_layer], double\* T)

{

double\* result = new double[hidden\_layer];

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

result[i] = 0;

double Inputs[input\_layer];

for (int k = 0; k < input\_layer; k++, x += 0.1)

Inputs[k] = function(x);

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

{

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

result[i] += Inputs[k] \* w1[i][k];

result[i] -= T[i];

result[i] = sigmoid(result[i]);

}

return result;

}

double get\_alpha(double\* w2, double Error, double Output, double\* Hiddens)

{

double alpha = 0, A = 0, B = 0;

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

{

A += pow(Error \* w2[i] \* (1 - Hiddens[i]) \* Hiddens[i], 2) \* Hiddens[i] \* (1 - Hiddens[i]);

B += pow(Error \* w2[i] \* (1 - Hiddens[i]) \* Hiddens[i], 2) \* Hiddens[i] \* Hiddens[i] \* (1 - Hiddens[i]) \* (1 - Hiddens[i]);

}

alpha = 4 \* A / (B \* (1 + Output \* Output));

return alpha;

}

double output(double x, double w1[hidden\_layer][input\_layer], double\* w2, double\* T)

{

double Result = 0;

double\* hidden\_result = hidden(x, w1, T);

for (int j = 0; j < hidden\_layer; j++) {

Result += hidden\_result[j] \* w2[j];

}

Result -= T[4];

return Result;

}

int main()

{

setlocale(LC\_ALL, "rus");

int epox = 0;

double w1[hidden\_layer][input\_layer], w2[hidden\_layer], T[hidden\_layer + 1], Reference, E\_min = 0.00002, alpha = 0.4, alpha1 = 0.4, x = 4, current, E = 0;

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

{

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

{

w1[i][k] = ((double)rand() / RAND\_MAX) \* 0.05;

}

w2[i] = ((double)rand() / RAND\_MAX) \* 0.05;

T[i] = ((double)rand() / RAND\_MAX) \* 0.05;

}

T[4] = ((double)rand() / RAND\_MAX) \* 0.05;

do

{

E = 0;

for (int q = 0; q < 1000; q++)

{

current = output(x, w1, w2, T);

Reference = function(x + 8 \* 0.1);

double error = current - Reference;

double\* Hiddens = hidden(x, w1, T);

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

w2[j] -= alpha \* error \* Hiddens[j];

T[4] += alpha \* error;

for (int k = 0; k < hidden\_layer; k++)

{

for (int i = 0; i < input\_layer; i++)

w1[k][i] -= alpha1 \* function(x + i \* 0.1) \* Hiddens[k] \* (1 - Hiddens[k]) \* w2[k] \* error;

T[k] += alpha1 \* Hiddens[k] \* (1 - Hiddens[k]) \* w2[k] \* error;

}

alpha1 = get\_alpha(w2, error, current, Hiddens);

x += 0.1;

E += pow(error, 2);

}

E /= 2;

// cout << "Error " << E << endl;

epox++;

} while (E > E\_min);

cout << epox << endl;

cout << "Эталон" << setw(23) << "Прогноз" << setw(20) << "Отклонение" << endl;

for (int i = 0; i < 100; i++)

{

double Result = output(x, w1, w2, T), Ethalonn = function(x + 8 \* 0.1);

cout << fixed << setprecision(5) << Ethalonn << setw(21) << Result << setw(29) << Result - Ethalonn << endl;

x += 0.1;

}

system("pause");

}

**Результат работы программы:**

814

Эталон Прогноз Отклонение

-0.11055 -0.11053 0.00002

-0.11038 -0.11033 0.00005

-0.11020 -0.11012 0.00007

-0.11000 -0.10991 0.00009

-0.10980 -0.10969 0.00011

-0.10958 -0.10946 0.00012

-0.10935 -0.10923 0.00013

-0.10911 -0.10899 0.00013

-0.10886 -0.10874 0.00012

-0.10860 -0.10850 0.00011

-0.10833 -0.10824 0.00009

-0.10805 -0.10798 0.00007

-0.10776 -0.10771 0.00005

-0.10746 -0.10744 0.00002

-0.10714 -0.10716 -0.00002

-0.10682 -0.10688 -0.00006

-0.10648 -0.10659 -0.00011

-0.10614 -0.10630 -0.00016

-0.10578 -0.10600 -0.00022

-0.10541 -0.10569 -0.00028

-0.10503 -0.10538 -0.00035

-0.10464 -0.10506 -0.00042

-0.10425 -0.10474 -0.00049

-0.10384 -0.10441 -0.00058

-0.10342 -0.10408 -0.00066

-0.10299 -0.10374 -0.00076

-0.10255 -0.10340 -0.00085

-0.10210 -0.10305 -0.00096

-0.10164 -0.10270 -0.00106

-0.10116 -0.10234 -0.00117

-0.10068 -0.10198 -0.00129

-0.10019 -0.10161 -0.00141

-0.09969 -0.10123 -0.00154

-0.09918 -0.10085 -0.00167

-0.09866 -0.10047 -0.00181

-0.09813 -0.10008 -0.00195

-0.09759 -0.09969 -0.00210

-0.09704 -0.09929 -0.00225

-0.09648 -0.09889 -0.00241

-0.09591 -0.09848 -0.00257

-0.09533 -0.09806 -0.00274

-0.09474 -0.09765 -0.00291

-0.09414 -0.09722 -0.00308

-0.09353 -0.09680 -0.00326

-0.09292 -0.09636 -0.00345

-0.09229 -0.09593 -0.00364

-0.09165 -0.09549 -0.00383

-0.09101 -0.09504 -0.00403

-0.09035 -0.09459 -0.00424

-0.08969 -0.09414 -0.00445

-0.08902 -0.09368 -0.00466

-0.08834 -0.09322 -0.00488

-0.08765 -0.09275 -0.00510

-0.08695 -0.09228 -0.00533

-0.08624 -0.09180 -0.00556

-0.08553 -0.09132 -0.00580

-0.08480 -0.09084 -0.00604

-0.08407 -0.09035 -0.00628

-0.08333 -0.08986 -0.00653

-0.08258 -0.08937 -0.00679

-0.08182 -0.08887 -0.00704

-0.08106 -0.08836 -0.00731

-0.08028 -0.08786 -0.00757

-0.07950 -0.08735 -0.00785

-0.07871 -0.08683 -0.00812

-0.07791 -0.08631 -0.00840

-0.07711 -0.08579 -0.00869

-0.07629 -0.08527 -0.00897

-0.07547 -0.08474 -0.00927

-0.07464 -0.08421 -0.00956

-0.07381 -0.08367 -0.00986

-0.07296 -0.08313 -0.01017

-0.07211 -0.08259 -0.01048

-0.07125 -0.08204 -0.01079

-0.07039 -0.08150 -0.01111

-0.06952 -0.08095 -0.01143

-0.06864 -0.08039 -0.01175

-0.06775 -0.07983 -0.01208

-0.06686 -0.07927 -0.01241

-0.06596 -0.07871 -0.01275

-0.06505 -0.07814 -0.01309

-0.06414 -0.07757 -0.01343

-0.06322 -0.07700 -0.01378

-0.06230 -0.07643 -0.01413

-0.06137 -0.07585 -0.01449

-0.06043 -0.07527 -0.01485

-0.05948 -0.07469 -0.01521

-0.05853 -0.07411 -0.01557

-0.05758 -0.07352 -0.01594

-0.05662 -0.07293 -0.01631

-0.05565 -0.07234 -0.01669

-0.05468 -0.07175 -0.01707

-0.05370 -0.07115 -0.01745

-0.05272 -0.07055 -0.01784

-0.05173 -0.06996 -0.01823

-0.05073 -0.06935 -0.01862

-0.04974 -0.06875 -0.01901

-0.04873 -0.06815 -0.01941

-0.04772 -0.06754 -0.01982

-0.04671 -0.06693 -0.02022

**Вывод:** в ходе работы разработал нелинейную ИНС для задач прогнозирования