Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №5

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 №5**

**Нелинейные ИНС в задачах прогнозирования**

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

**Вариант 11**

**Задача**

1. Изучить теоретические сведения, приведенные в данных методических указания и методических указаниях для работ №3 и №4.
2. Написать на любом ЯВУ программу моделирования нелинейной ИНС для распознавания образов. Рекомендуется использовать сигмоидную функцию, но это не является обязательным. Количество НЭ в скрытом слое взять согласно варианту работы №3. Его можно варьировать, если сеть не обучается или некорректно функционирует.
3. Провести исследование полученной модели. При этом на вход сети необходимо подавать искаженные образы, в которых инвертированы некоторые биты. Критерий эффективности процесса распознавания - максимальное кодовое расстояние (количество искаженных битов) между исходным и поданным образом.

**Код программы:**

#include <iostream>

#include <math.h>

#define INPUT 8

#define HIDDEN 3

#define OUTPUT 1

using namespace std;

double sigmoid(double x)

{

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

}

double\* get\_hiddens(bool\* Inputs, double w12[INPUT][HIDDEN], double T\_Hid[])

{

double\* Hiddens = new double[HIDDEN];

for (int i = 0; i < HIDDEN; i++) Hiddens[i] = 0;

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

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

Hiddens[i] += w12[j][i] \* Inputs[j];

}

Hiddens[i] -= T\_Hid[i];

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

}

return Hiddens;

}

double\* get\_result(bool\* Inputs, double w12[INPUT][HIDDEN], double T\_Hid[], double w23[HIDDEN][OUTPUT], double T\_Out[], double Hiddens[HIDDEN])

{

double\* Results = new double[OUTPUT];

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

Results[i] = 0;

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

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

Results[j] += Hiddens[i] \* w23[i][j];

}

Results[j] -= T\_Out[j];

Results[j] = sigmoid(Results[j]);

}

return Results;

}

int main()

{

setlocale(LC\_ALL, "rus");

bool Vect1[] = { 0,1,0,0,1,1,0,1,0,0,0,0,1,0,1,0,1,0,0,0 };

bool Vect2[] = { 1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0 };

bool Vect3[] = { 1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1 };

bool\* Inputs = new bool[INPUT];

for (int i = 0; i < INPUT; i++) Inputs[i] = 0;

bool\*\* Vectors = new bool\* [8];

Vectors[0] = Vect1;

Vectors[1] = Vect2;

Vectors[2] = Vect3;

double w12[INPUT][HIDDEN], w23[HIDDEN][OUTPUT], T\_Hid[HIDDEN], T\_Out[OUTPUT], E\_min = 0.0001, alpha = 0.04, Ethalon, E = 0, Outputs[OUTPUT] = { 0 };

double\* Currents = new double[OUTPUT];

double\* Hiddens = new double[HIDDEN];

double Mistakes[OUTPUT] = { 0 };

double Ethalons[OUTPUT] = { 0 };

double MistakesHid[HIDDEN] = { 0 };

int Iter = 1;

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

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

w12[i][j] = ((double)rand() / (RAND\_MAX)) - 0.5;

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

w23[j][k] = ((double)rand() / (RAND\_MAX)) - 0.5;

T\_Out[k] = ((double)rand() / (RAND\_MAX)) - 0.5;

}

T\_Hid[j] = ((double)rand() / (RAND\_MAX)) - 0.5;

}

}

int H = 0;

do {

E = 0;

for (int N = 0; N < OUTPUT; N++) {

Ethalons[0] = 0;

Ethalons[N] = 1;

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

Inputs = Vectors[N];

Hiddens = get\_hiddens(Inputs, w12, T\_Hid);

Currents = get\_result(Inputs, w12, T\_Hid, w23, T\_Out, Hiddens);

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

Mistakes[i] = Currents[i] - Ethalons[i];

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

for (int m = 0; m < OUTPUT; m++) {

MistakesHid[j] += Mistakes[m] \* Currents[m] \* (1 - Currents[m]) \* w23[j][m];

}

}

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

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

w23[i][j] -= alpha \* Mistakes[j] \* Currents[j] \* (1 - Currents[j]) \* Hiddens[i];

}

T\_Out[j] += alpha \* Mistakes[j] \* Currents[j] \* (1 - Currents[j]);

}

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

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

w12[i][j] -= alpha \* MistakesHid[j] \* Hiddens[j] \* (1 - Hiddens[j]) \* Inputs[i];

}

T\_Hid[j] += alpha \* MistakesHid[j] \* Hiddens[j] \* (1 - Hiddens[j]);

}

E += pow(Mistakes[N], 2);

}

}

E /= 2;

cout << E << endl;

H++;

} while (E > E\_min);

cout << H << endl;

//Прогнозирование

double\* HiddenPred;

double\* Values;

bool Vectors3[] = { 1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0 };

bool Vectors4[] = { 0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0 };

bool Vectors5[] = { 1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1 };

bool Vectors6[] = { 1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0 };

bool Vectors7[] = { 0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1 };

Vectors[3] = Vectors3;

Vectors[4] = Vectors4;

Vectors[5] = Vectors5;

Vectors[6] = Vectors6;

Vectors[7] = Vectors7;

for (int i = 0; i < 8; i++) {

Inputs = Vectors[i];

cout << "Вектор " << i + 1 << " - ";

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

cout << Inputs[j] << ' ';

}

cout << endl << "Результат ";

HiddenPred = get\_hiddens(Inputs, w12, T\_Hid);

Values = get\_result(Inputs, w12, T\_Hid, w23, T\_Out, HiddenPred);

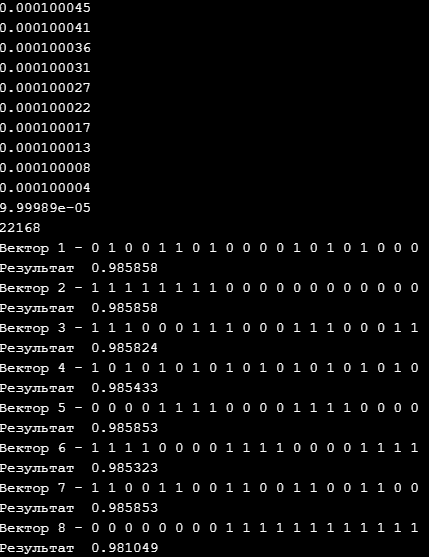
cout << Values[0] << endl;

}

system("pause");

}

**Результат работы программы:**



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