Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах распознавания образов»

Выполнила:

Савицкая А.Ю., ПО-7

Проверил:

Крощенко А.А.

Брест, 2021

**Цель работы:** изучить обучение и функционирование нелинейной ИНС при решении задач распознавания образов.

Для решения задач распознавания образов используется следующая нейросетевая архитектура. За основу берется многослойная ИНС прямого распространения. Количество нейронов во входном слое равно длине вектора, представляющего образ. Количество выходных нейронов равно количеству образов и каждому нейрону ставится в соответствие образ. Число нейронных элементов в скрытом слое варьируется для достижения оптимального качества распознавания. Распознавание происходит так. Сначала сеть обучается на наборе пар вектор (он же входной образ) - код этого вектора. Код представляет собой битовую последовательность, в которой один из битов выставлен в единицу, а все остальные нули. Номер этого бита - номер образа, подаваемого на вход. Затем происходит собственно распознавание. На вход сети подается неизвестный вектор. Он обрабатывается ИНС, и на выходе мы получаем код этого вектора. Вектор может быть искажен помехами.

**Задание:**

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.

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Вектор 1 | Вектор 2 | Вектор 3 |
| 7 | 7 | 6 | 8 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Данные вектора | | | | | | | | | | | | | | | | | | | |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

**Текст программы:**

#include <iostream>

#include <math.h>

#define e 2.71828

using namespace std;

double sigmoid(double x) { //Сигмоид

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

}

//работа со скрытым слоем

double\* get\_hiddens(bool\* Inputs, double w12[6][2], double T\_Hid[]) {

double\* Hiddens = new double[2];

for (int i = 0; i < 2; i++) Hiddens[i] = 0;

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

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

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

}

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

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

}

return Hiddens;

}

//получение результатов

double\* get\_result(bool\* Inputs, double w12[6][2], double T\_Hid[], double w23[2][1], double T\_Out[], double Hiddens[2]) {

double\* Results = new double[1];

for (int i = 0; i < 1; i++)

Results[i] = 0;

for (int j = 0; j < 1; j++) {

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

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

}

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

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

}

return Results;

}

int main() {

system("color f0");

setlocale(0, "");

///Векторы согласно варианту

bool Vector1[] = { 0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1 };

bool Vector2[] = { 1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0 };

bool Vector3[] = { 1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1 };

///ввод входных сигналов и векторов

bool\* Inputs = new bool[6];

for (int i = 0; i < 6; i++) Inputs[i] = 0;

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

Vectors[0] = Vector1;

Vectors[1] = Vector2;

Vectors[2] = Vector3;

///

double w12[6][2], w23[2][1], T\_Hid[2], T\_Out[1], E\_min = 0.0001, alpha = 0.04, Ethalon, E = 0, Outputs[1] = { 0 };

double\* Currents = new double[1];

double\* Hiddens = new double[2];

double Mistakes[1] = { 0 };

double Ethalons[1] = { 0 };

double MistakesHid[2] = { 0 };

int Iter = 1; //начальное количество итераций

///рандомизация весов и порогов

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

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

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

for (int k = 0; k < 1; 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 < 1; N++) {

Ethalons[0] = 0;

Ethalons[N] = 1;

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

///нахождение результатов и ошибок

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 < 1; i++)

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

///ошибки на скрытом слое

for (int j = 0; j < 1; j++) {

for (int m = 0; m < 1; m++) {

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

}

}

for (int j = 0; j < 1; j++) {

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

for (int i = 0; i < 6; 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;

if (H % 100 == 0 || H < 300) {

cout << H << ";" << E << endl;

}

H++;

} while (E > E\_min);

//прогнозирование

double\* HiddenPred;

double\* Values;

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

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 << "Результат вектора " << i + 1 << " значение которого ";

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);

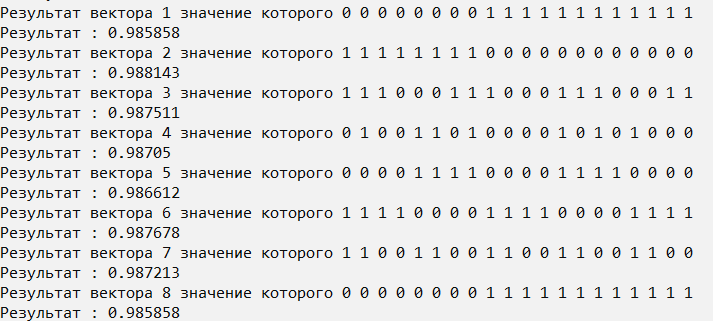
cout << Values[0] << endl;

}

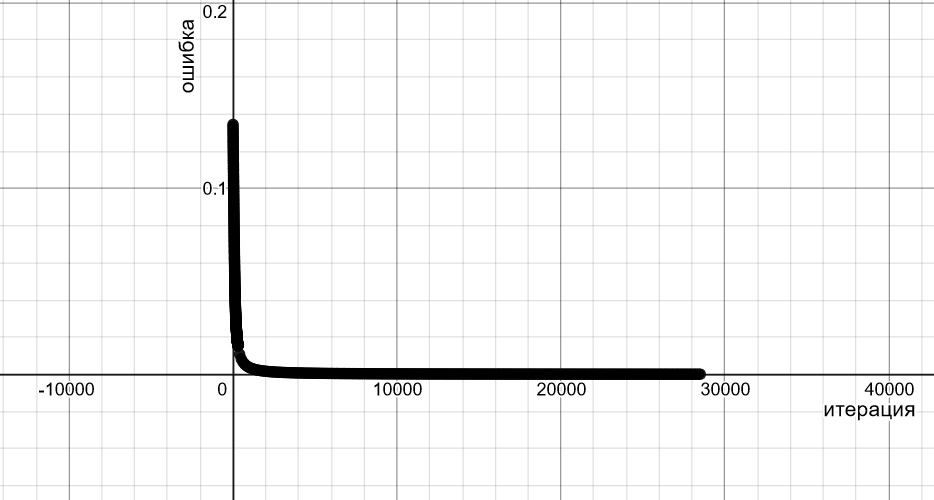
system("pause");

}

**Результаты тестирования программы:**



**График изменения ошибки в зависимости от итерации:**



**Вывод:**

Изучила обучение и функционирование нелинейной ИНС при решении задач распознавания образов.