Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По ИМОД

Тема: «Нелинейные искусственные нейронные сети в задачах распознавания образов»

Выполнил:

студент 4-го курса

группы АС-56

*Марук К.В.*

Проверил:

*Савицкий Ю. В*.

Брест 2022

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

Вариант 11

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Данные вектора | | | | | | | | | | | | | | | | | | | |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

**Задание.** Написать на любом языке высокого уровня программу моделирования нелинейной ИНС для распознавания образов. Рекомендуется использовать сигмоидную функцию (но это не является обязательным). Количество нейронных сетей в скрытом слое взять согласно варианту работы №4. Его можно варьировать, если сеть не обучается или некорректно функционирует. Для организации обучающей выборки использовать данные таблиц 5.1 и 5.2.

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.

**Текст программы:**

Random random = new();

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

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

List<double> Tj = new();

List<double> Tk = new();

List<int> vector\_7 = new(new int[] { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 });

List<int> vector\_6 = new(new int[] { 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 });

List<int> vector\_8 = new(new int[] { 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1 });

List<List<int>> all\_vectors = new();

all\_vectors.Add(vector\_7);

all\_vectors.Add(vector\_6);

all\_vectors.Add(vector\_8);

double step = 0.5;

double MIN\_ERROR = 1e-6;

int input\_neuron\_number = 6;

int hidden\_neuron\_number = 2;

int output\_neuron\_number = 1;

List<List<double>> drawing\_data = new() { new List<double>(), new List<double>() };

List<double> errors = new(output\_neuron\_number) { 0 };

List<double> reference = new(output\_neuron\_number) { 0 };

List<double> hiddenLayer\_error = new(hidden\_neuron\_number) { 0, 0 };

int iteration = 1;

int epoch = 0;

double error = 0;

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

Tj.Add(random.NextDouble(-0.5, 0.5));

for (int i = 0; i < output\_neuron\_number; i++)

Tk.Add(random.NextDouble(-0.5, 0.5));

for (int i = 0; i < output\_neuron\_number; i++)

{

List<double> tempList = new();

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

{

var a = random.NextDouble(-0.1, 0.1);

tempList.Add(a);

if (j == 1) Wjk.Add(tempList);

}

}

for (int i = 0; i < input\_neuron\_number - 1; i++)

{

List<double> tempList = new();

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

{

var a = random.NextDouble(-0.1, 0.1);

tempList.Add(a);

if (j == 1) Wij.Add(tempList);

}

}

do

{

error = 0;

for (int N = 0; N < output\_neuron\_number; N++)

{

reference[N] = 1;

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

{

var y = all\_vectors[N];

var Yj = Sj\_onHiddenLayer(y);

var Yk = Sk\_onOutputLayer(Yj);

for (int index = 0; index < output\_neuron\_number; index++)

{

errors[index] = Yk[index] - reference[index];

}

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

{

for (int k = 0; k < output\_neuron\_number; k++)

{

hiddenLayer\_error[j] += errors[k] \* Yk[k] \* (1 - Yk[k]) \* Wjk[k][j];

}

}

Wjk\_change(Yj, Yk, errors);

Wij\_change(Yj, hiddenLayer\_error, y);

error += Math.Pow(errors[N], 2);

}

}

error /= 2;

drawing\_data[0].Add(epoch);

drawing\_data[1].Add(error);

epoch += 1;

} while (error > MIN\_ERROR);

for (int i = 0; i < all\_vectors.Count; i++)

{

var input = all\_vectors[i];

Console.Write($"Result vector: {i + 1} : ");

for (int j = 0; j < vector\_7.Count; j++)

{

Console.Write(input[j]);

}

Console.WriteLine();

Console.Write($"Result: ");

var hiddenLayer\_prev = Sj\_onHiddenLayer(input);

var Values = Sk\_onOutputLayer(hiddenLayer\_prev);

Console.WriteLine(Values[0]);

}

int co = 0;

StreamWriter f;

foreach (var item in drawing\_data)

{

if (co == 0)

{

f = new StreamWriter("epohi.txt", true);

co++;

}

else f = new StreamWriter("errors.txt", true);

foreach (var item1 in item)

{

f.WriteLine(item1);

}

f.Close();

}

double sigmoidFunction(double S) =>

1 / (1 + Math.Exp(-S));

List<double> Sj\_onHiddenLayer(List<int> y)

{

List<double> Sj = new();

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

{

double value = 0.0;

for (int i = 0; i < input\_neuron\_number - 1; i++)

{

value += y[i] \* Wij[i][j];

}

value -= Tj[j];

Sj.Add(sigmoidFunction(value));

}

return Sj;

}

List<double> Sk\_onOutputLayer(List<double> Yj)

{

List<double> Sk = new();

for (int j = 0; j < output\_neuron\_number; j++)

{

var value = 0.0;

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

{

value += Yj[i] \* Wjk[j][i];

}

value -= Tk[j];

Sk.Add(sigmoidFunction(value));

}

return Sk;

}

void Wjk\_change(List<double> Yj, List<double> Yk, List<double> error)

{

for (int j = 0; j < output\_neuron\_number; j++)

{

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

{

Wjk[j][i] -= step \* error[j] \* Yk[j] \* (1 - Yk[j]) \* Yj[i];

}

Tk[j] += error[j] \* step \* Yk[j] \* (1 - Yk[j]);

}

}

void Wij\_change(List<double> Yj, List<double> hiddenLayer\_error, List<int> y)

{

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

{

for (int i = 0; i < input\_neuron\_number - 1; i++)

{

Wij[i][j] -= step \* hiddenLayer\_error[j] \* y[i] \* Yj[j] \* (1 - Yj[j]);

}

Tj[j] += step \* hiddenLayer\_error[j] \* Yj[j] \* (1 - Yj[j]);

}

}

public static class RandomExtensions

{

public static double NextDouble(this Random random, double minValue, double maxValue)

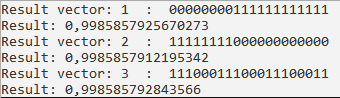
{

return random.NextDouble() \* (maxValue - minValue) + minValue;

}

}

**Результат выполнения программы:**



**Вывод:** в ходе лабораторной работы реализовал нелинейную ИНС для распознавания образов.