МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Лабораторная работа 4

По предмету «Системы искусственного интеллекта»

Р-модель распознавания

Выполнил студент: Пшеничный Д.О.

Факультет, группа: ФКФН, ПО(б)-81

Руководитель работы: Тормозов В.С.

Хабаровск – 2022г.

**Задание:** пусть образы объектов описываются группами из двух целочисленных параметров (x, y). Имеется два непересекающихся класса объектов. Требуется провести границу между классами. Способ построения разграничивающей прямой предлагается разработать самостоятельно.

Исходные данные. Два натуральных числа N1 – количество образцов из первого класса и N2 – количество образцов из второго класса. N1 + N2 пар чисел (xk , yk ) для образцов из первого и второго классов.

Требуется выполнить графическую иллюстрацию Р-модели с использованием нейросетей.

**Исходный код программы:**

public class NeuralNetwork

{

public delegate double activation(double x);

public delegate double derivatuve(double x);

private double learningRate;

private Layer[] layers;

private activation Activation;

private derivatuve Derivatuve;

public NeuralNetwork(double learningRate, activation \_activation, derivatuve \_derivative, int[] sizes)

{

this.learningRate = learningRate;

Activation = \_activation;

Derivatuve = \_derivative;

layers = new Layer[sizes.Length];

Random r = new Random();

for (int i = 0; i < sizes.Length; i++)

{

int nextSize = 0;

if (i < sizes.Length - 1)

nextSize = sizes[i + 1];

layers[i] = new Layer(sizes[i], nextSize);

for (int j = 0; j < sizes[i]; j++)

{

layers[i].biases[j] = r.NextDouble() \* 2.0 - 1.0;

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

{

layers[i].weights[j, k] = r.NextDouble() \* 2.0 - 1.0;

}

}

}

}

public double[] feedForward(double[] inputs)

{

//System.arraycopy(inputs, 0, layers[0].neurons, 0, inputs.length);

for (int i = 0; i < inputs.Length; i++)

layers[0].neurons[i] = inputs[i];

for (int i = 1; i < layers.Length; i++)

{

Layer l = layers[i - 1];

Layer l1 = layers[i];

for (int j = 0; j < l1.size; j++)

{

l1.neurons[j] = 0;

for (int k = 0; k < l.size; k++)

{

l1.neurons[j] += l.neurons[k] \* l.weights[k,j];

}

l1.neurons[j] += l1.biases[j];

l1.neurons[j] = Activation(l1.neurons[j]);

}

}

return layers[layers.Length - 1].neurons;

}

public void backpropagation(double[] targets)

{

double[] errors = new double[layers[layers.Length - 1].size];

for (int i = 0; i < layers[layers.Length - 1].size; i++)

{

errors[i] = targets[i] - layers[layers.Length - 1].neurons[i];

}

for (int k = layers.Length - 2; k >= 0; k--)

{

Layer l = layers[k];

Layer l1 = layers[k + 1];

double[] errorsNext = new double[l.size];

double[] gradients = new double[l1.size];

for (int i = 0; i < l1.size; i++)

{

gradients[i] = errors[i] \* Derivatuve(layers[k + 1].neurons[i]);

gradients[i] \*= learningRate;

}

double[,] deltas = new double[l1.size, l.size];

for (int i = 0; i < l1.size; i++)

{

for (int j = 0; j < l.size; j++)

{

deltas[i,j] = gradients[i] \* l.neurons[j];

}

}

for (int i = 0; i < l.size; i++)

{

errorsNext[i] = 0;

for (int j = 0; j < l1.size; j++)

{

errorsNext[i] += l.weights[i,j] \* errors[j];

}

}

errors = new double[l.size];

//System.arraycopy(errorsNext, 0, errors, 0, l.size);

for (int i = 0; i < errorsNext.Length; i++)

errors[i] = errorsNext[i];

Console.WriteLine();

double[,] weightsNew = new double[l.weights.GetLength(0), l.weights.GetLength(1)];

for (int i = 0; i < l1.size; i++)

{

for (int j = 0; j < l.size; j++)

{

weightsNew[j,i] = l.weights[j,i] + deltas[i,j];

}

}

l.weights = weightsNew;

for (int i = 0; i < l1.size; i++)

{

l1.biases[i] += gradients[i];

}

}

}

public class Layer

{

public int size;

public double[] neurons;

public double[] biases;

public double[,] weights;

public Layer(int size, int nextSize)

{

this.size = size;

neurons = new double[size];

biases = new double[size];

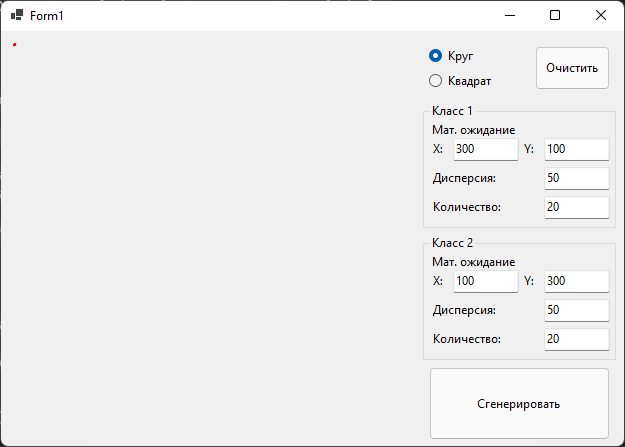
weights = new double[size, nextSize];

}

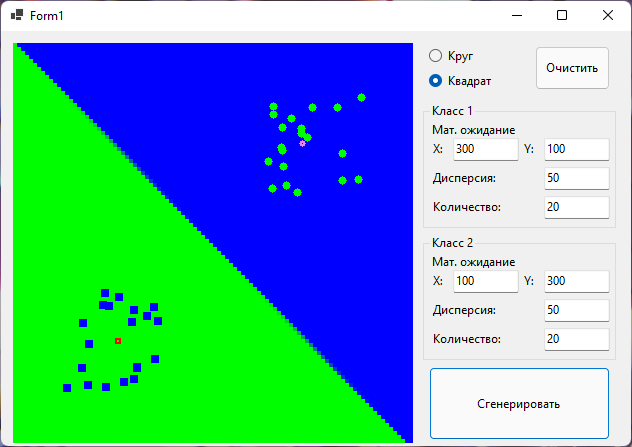
}

}

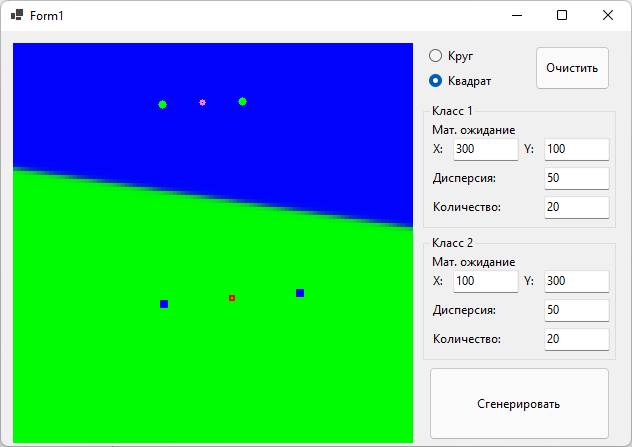
**Скриншоты работы программы:**



*Основное окно программы*



*Сгенерированные автоматически точки*



*Расставленные вручную точки*