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 2

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 Замечание.

**Исходный код программы:**

namespace R\_Model

{

public partial class Form1 : Form

{

List<RObject> points = new List<RObject>();

List<RObject> squares = new List<RObject>();

Point averagePoint = new Point(0, 0);

Point averageSquare = new Point(0, 0);

RObject interactionObject = null;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

void RefreshForm()

{

averagePoint = MathDealer.GetAriphmeticAveragePoint(points);

averageSquare = MathDealer.GetAriphmeticAveragePoint(squares);

panel1.Invalidate();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

Graphics graphics = panel1.CreateGraphics();

foreach(var i in points)

{

graphics.DrawEllipse(new Pen(Color.Green, 4), new Rectangle(i.Position.X - 2, i.Position.Y - 2, 4, 4));

}

foreach (var i in squares)

{

graphics.DrawRectangle(new Pen(Color.Blue, 4), new Rectangle(i.Position.X - 2, i.Position.Y - 2, 4, 4));

}

graphics.DrawEllipse(new Pen(Color.Violet, 2), new Rectangle(averagePoint.X - 2, averagePoint.Y - 2, 4, 4));

graphics.DrawRectangle(new Pen(Color.Red, 2), new Rectangle(averageSquare.X - 2, averageSquare.Y - 2, 4, 4));

if (averagePoint != new Point(0, 0) && averageSquare != new Point(0, 0))

{

try

{

double pDistance = MathDealer.GetAverageDistance(points);

double sqDictance = MathDealer.GetAverageDistance(squares);

double lamba = pDistance / (pDistance + sqDictance);

Debug.WriteLine(lamba);

Point perpendCenter = new Point();

if (pDistance < sqDictance)

perpendCenter = MathDealer.GetPerpendPoint(averagePoint, averageSquare, lamba);

else

perpendCenter = MathDealer.GetPerpendPoint(averageSquare, averagePoint, lamba);

float tangenc = (float)(averagePoint.Y - averageSquare.Y) / (float)(averagePoint.X - averageSquare.X);

Point startPoint = MathDealer.GetPerpendicularLinePoint(perpendCenter, tangenc, 0);

Point endPoint = MathDealer.GetPerpendicularLinePoint(perpendCenter, tangenc, 400);

graphics.DrawLine(new Pen(Color.Black), startPoint, endPoint);

}

catch

{

}

}

}

private void panel1\_MouseDoubleClick(object sender, MouseEventArgs e)

{

if(radioButton1.Checked)

{

points.Add(new RObject(new Point(e.X, e.Y)));

}

else if(radioButton2.Checked)

{

squares.Add(new RObject(new Point(e.X, e.Y)));

}

RefreshForm();

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

foreach (var x in points)

{

if (e.X <= x.Position.X + 7 && e.X >= x.Position.X - 7 && e.Y <= x.Position.Y + 7 && e.Y >= x.Position.Y - 7)

{

interactionObject = x;

RefreshForm();

return;

}

}

foreach (var x in squares)

{

if (e.X <= x.Position.X + 7 && e.X >= x.Position.X - 7 && e.Y <= x.Position.Y + 7 && e.Y >= x.Position.Y - 7)

{

interactionObject = x;

RefreshForm();

return;

}

}

}

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (interactionObject != null)

{

interactionObject.Position = new Point(e.X, e.Y);

RefreshForm();

}

}

private void panel1\_MouseUp(object sender, MouseEventArgs e)

{

if(interactionObject != null)

{

interactionObject = null;

}

}

private void button1\_Click(object sender, EventArgs e)

{

points.Clear();

squares.Clear();

RefreshForm();

}

private void button2\_Click(object sender, EventArgs e)

{

squares.Clear();

points.Clear();

Random random = new Random();

int x1 = int.Parse(textBox1.Text);

int y1 = int.Parse(textBox2.Text);

int sig1 = int.Parse(textBox3.Text);

int x2 = int.Parse(textBox8.Text);

int y2 = int.Parse(textBox7.Text);

int sig2 = int.Parse(textBox6.Text);

for (int i = 0; i < int.Parse(textBox4.Text); i++)

{

points.Add(new RObject(new Point(x1 + random.Next(-sig1, sig1), y1 + random.Next(-sig1, sig1))));

}

for (int i = 0; i < int.Parse(textBox5.Text); i++)

{

squares.Add(new RObject(new Point(x2 + random.Next(-sig2, sig2), y2 + random.Next(-sig2, sig2))));

}

RefreshForm();

}

}

class RObject

{

public Point Position { get; set; }

public RObject(Point \_position)

{

Position = \_position;

}

}

static class MathDealer

{

public static Point GetAriphmeticAveragePoint(List<RObject> rObjects)

{

Point res = new Point(0, 0);

if (rObjects.Count == 0)

return res;

foreach (var i in rObjects)

{

res.X += i.Position.X;

res.Y += i.Position.Y;

}

res.X = res.X / rObjects.Count;

res.Y = res.Y / rObjects.Count;

return res;

}

public static Point GetPerpendicularLinePoint(Point point, float k, int x)

{

Point res = new Point();

res.X = x;

res.Y = (int)((-1 / (float)k) \* (x - point.X) + point.Y);

return res;

}

public static Point GetPerpendPoint(Point point1, Point point2)

{

Point res = new Point();

res = new Point((point1.X + point2.X) / 2, (point1.Y + point2.Y) / 2);

return res;

}

public static Point GetPerpendPoint(Point point1, Point point2, double lambda)

{

Point res = new Point();

double x = (point1.X + lambda \* point2.X) / (1 + lambda);

double y = (point1.Y + lambda \* point2.Y) / (1 + lambda);

res = new Point((int)x, (int)y);

return res;

}

public static double GetAverageDistance(List<RObject> objects)

{

Point averagePoint = GetAriphmeticAveragePoint(objects);

double res = 0;

foreach(var i in objects)

{

res += Math.Sqrt(Math.Pow(averagePoint.X - i.Position.X, 2) + Math.Pow(averagePoint.Y - i.Position.Y, 2));

}

res /= objects.Count;

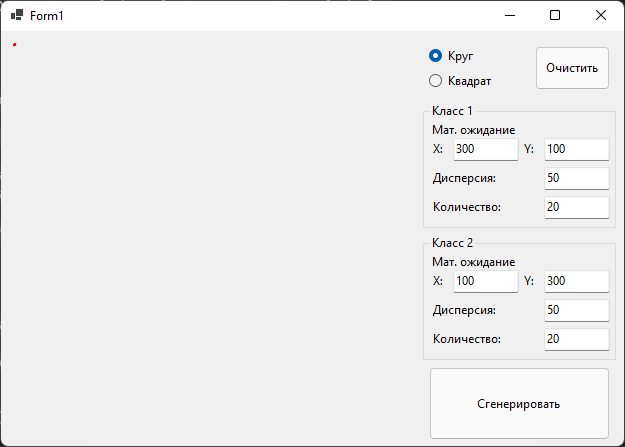
return res;

}

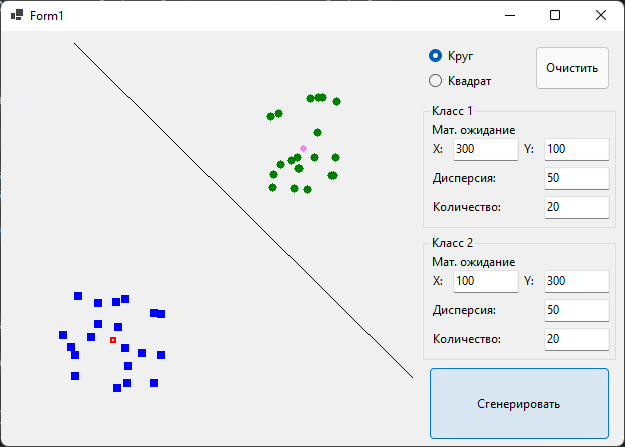
}

}

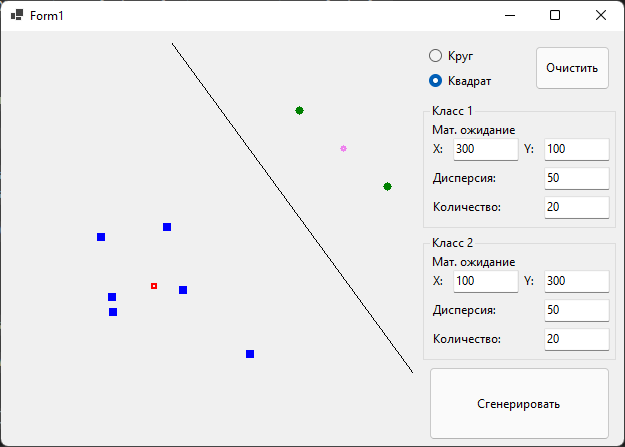
**Скриншоты работы программы:**



*Основное окно программы*



*Сгенерированные автоматически точки*



*Расставленные вручную точки*