Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Алтайский государственный технический   
университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

наименование кафедры

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н.Тушев (подпись) (и.о., фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.

дата

Отчет по лабораторным работам 1-8

\_\_\_\_\_\_\_\_\_\_\_ИЗ 09.04.04.27.000 О\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

обозначение документа

Дисциплина:

«Инженерия знаний и системы искусственного интеллекта»

наименование дисциплины

Студент группы 8ПИ-61 А. О. Корней (и.о., фамилия)

Барнаул 2016

Оглавление

[1. Применение генетического алгоритма для приближенного решения задач большой размерности 3](#_Toc468920564)

[2. Нейронные сети. Обучение методом обратного распространения ошибки 24](#_Toc468920565)

[7. Реализовать модификации алгоритмов обучения адаптивный метод и метод моментов. 24](#_Toc468920566)

[3. Экспертные системы. Правило Байеса. 52](#_Toc468920567)

[4. Карты Кохонена 65](#_Toc468920568)

[5. Визуализация многомерных образов. Алгоритм 1 и 2 порядка.. 81](#_Toc468920569)

[6. Муравьинный алгоритм 83](#_Toc468920570)

[8. Итеративный метод Браун-Робинсона 100](#_Toc468920571)

[Альфа-бета отсечение 110](#_Toc468920572)

# 1. Применение генетического алгоритма для приближенного решения задач большой размерности

**Задание**

Верно ли, что заданное семейство трехэлементных подмножеств заданного конечного множества такого, что для некоторого натурального , содержит *точное покрытие* множества , т.е. такое подсемейство , что каждый элемент из содержится ровно в одном элементе из ?

**Кодирование организма**

Организм кодируется следующим образом:

1 – если подмножество включается в решение, 0 – если не включается.

**Функция приспособленности**

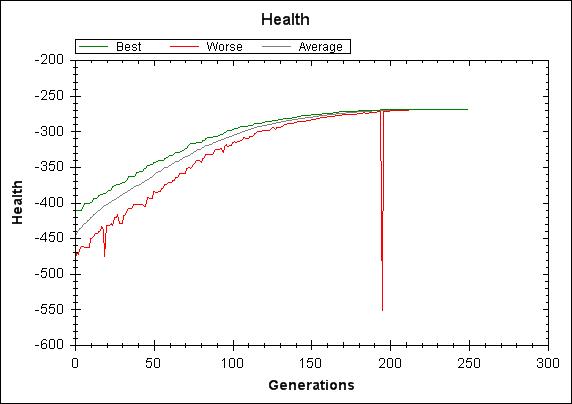
Штрафы вводились следующим образом:

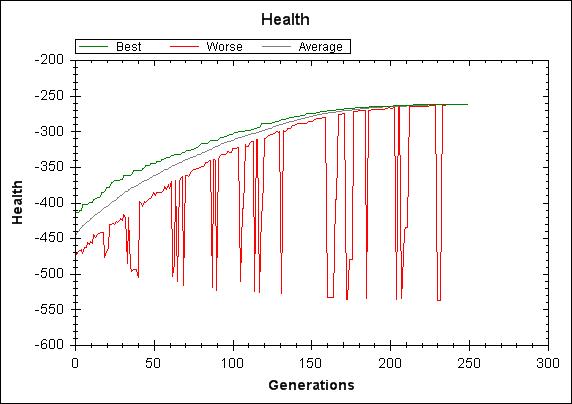
* Для каждого элемента из X, не попавшего в решение, добавить к штрафу 1.
* Для каждого элемента из Х, встретившегося более одного раза, добавить к штрафу lg от числа повторов.

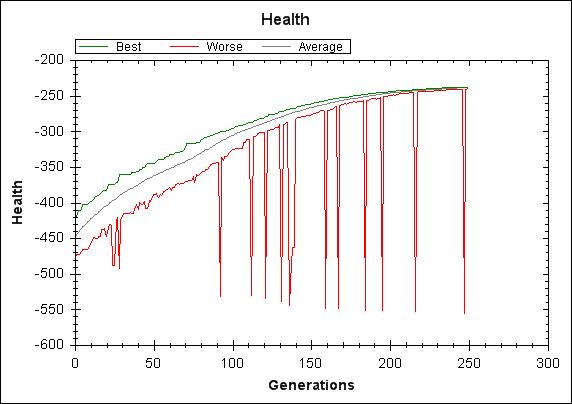
Функция приспособленности = - размер штрафа.

Таким образом, идеальная оценка приспособленности соответствует нулевому штрафу.

**Результаты работы программы**







**Код программы**

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

using ZedGraph;

namespace GenAlg

{

public class MyOrganizm

{

public BitArray Organizm { get; set; }

public double Fitness { get; set; }

}

class Solver

{

//---------------------- F I E L D S -------------------------------

private PointPairList

best,

worse,

average;

private int XSize, CCount;

private int[][] C;

private int Z, Q, G;

private double Pv, Pvm, Pm;

private Random Generator;

//private List<BitArray> Generation;

private List<MyOrganizm> Generation;

//------------------ P R O P E R T I E S -------------------------

public PointPairList Best { get { return best; } }

public PointPairList Worse { get { return worse; } }

public PointPairList Average { get { return average; } }

//------------------- C O N S T R U C T O R S ----------------------

public Solver(int xSize,int cCount,int z, int q, int g, double pv, double pvm, double pm)

{

XSize = xSize;

CCount = cCount;

Generation = new List<MyOrganizm>();

best=new PointPairList();

worse = new PointPairList();

average=new PointPairList();

Z = z; Q = q; G = g;

Pv = pv; Pm = pm; Pvm = pvm;

C = new int[1][];

Generator = new Random();

GenerateC(CCount);

}

//---------------- P R E P E A R I N G -----------------------------

private void GenerateX(int xSize)

{

if (xSize % 3 != 0)

throw new Exception("X must contain 3n of elements");

XSize = xSize;

}

private void GenerateC(int cCount)

{

CCount = cCount;

if (cCount < XSize / 3)

throw new Exception("No solve");

C = new int[cCount][];

//int j = 0;

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

{

C[i] = new int[3];

int x;

x = Generator.Next() % XSize;

C[i][0] = x;

//C[i][1] = (Math.Abs(C[i][0] + Generator.Next() % (4 \* XSize)\*Math.Sign(Generator.Next()%3-1)))%XSize;

//C[i][1] = (Math.Abs(C[i][0] + Generator.Next() % (4 \* XSize) \* Math.Sign(Generator.Next() % 3 - 1))) % XSize;

do

{

x = Convert.ToInt32(Generator.NextDouble() \* XSize) % XSize;

//x = Generator.Next() % XSize;

//x = (Math.Abs(C[i][0] + Generator.Next() % (4 \* XSize) \* Math.Sign(Generator.Next() % 3 - 1))) % XSize;

} while (x == C[i][0]);

C[i][1] = x;

do

{

x = Convert.ToInt32(Generator.NextDouble() \* XSize) % XSize;

//x = (Math.Abs(C[i][1] + Generator.Next() % (2 \* XSize) \* Math.Sign(Generator.Next() % 3 - 1))) % XSize;

//x = Generator.Next() % XSize;

} while (x == C[i][0] || x == C[i][1]);

C[i][2] = x;

}

}

//--------------------- S O L V E R O U T I N E ------------------

/// <summary>

/// Генерация первого поколения организмов

/// </summary>

private void InitFirstGeneration()

{

//alternG = new List<MyOrganizm>();

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

{

BitArray Organizm = new BitArray(CCount);

for (int j = 0; j < CCount; j++)

Organizm.Set(j, Generator.NextDouble() < 0.5);

MyOrganizm p = new MyOrganizm();

p.Organizm = Organizm;

p.Fitness = 0.0;

Generation.Add(p);

}

}

/// <summary>

/// Расчет приспособленности организма по формуле

/// </summary>

/// <param name="Organizm">Организм, представленный набором битов</param>

/// <returns></returns>

private double CalculateFitness(BitArray Organizm)

{

//Считаем, что особь идеальна

double Fitness = 0.0;

//Массив, равный по мощности Х.

//В нем накапливаем повторы каждого из

//элеменов Х

int[] Repeats = new int[XSize];

//Набор уникалных значений - с его

//помощь посчитаем, сколько разных элементов

//встречается и сколько не встречается вообще

///\*

//HashSet<int> checker;

//checker = new HashSet<int>();

//Пробежимся по всему организму

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

{

//Если Ci включается в особь, соберем статистику

if (Organizm.Get(i))

{

//Если checker.Add вернет true, значит Cij впервые попало к нам

//Иначе нужно накинуть повтор

Repeats[C[i][0]]++;//= checker.Add(C[i][0]) ? 0 : 1;

Repeats[C[i][1]]++;//= checker.Add(C[i][1]) ? 0 : 1;

Repeats[C[i][2]]++;//= checker.Add(C[i][2]) ? 0 : 1;

}

}

//Штраф за непопадание

//\*/

//Fitness -= 1.0 \* (XSize - checker.Count);

//Штраф за повторы

foreach (int locRep in Repeats)

{

if (locRep == 0) Fitness -= 1.0;

else

Fitness -= Math.Log10(locRep);

}

return Fitness;

}

/// <summary>

/// Классическое скрещивание двух организмов со случайной позиции

/// </summary>

/// <param name="parentA">Копия родителя А,

/// станет первым потомком</param>

/// <param name="parentB">Копия родителя Б,

/// станет вторым потомком</param>

private void Mix(MyOrganizm parentA, MyOrganizm parentB)

{

//Сгенерируем случайное число, оно станет позицией

//скрещивания. Чтоб хоть что-то делалось, ограничим

//максимально допустимое значение.

int mixPosition = Generator.Next() % (CCount - 1);

//временная переменная для свопа

bool buffer;

for (int i = mixPosition; i < CCount; i++)

{

//обмен между двумя битовыми массивами

buffer = parentA.Organizm.Get(i);

parentA.Organizm.Set(i, parentB.Organizm.Get(i));

parentB.Organizm.Set(i, buffer);

}

//у новых организмов посчитаем здоровье

parentA.Fitness = CalculateFitness(parentA.Organizm);

parentB.Fitness = CalculateFitness(parentB.Organizm);

}

/// <summary>

/// Инвертирует все гены организма

/// </summary>

/// <param name="org">организм, подверженный мутации</param>

private void Mutate(MyOrganizm org)

{

//инвертируем биты с исползованием встроенного метода

org.Organizm.Not();

//пересчитаем здоровье

org.Fitness = CalculateFitness(org.Organizm);

}

//----------------------------- R U N N I N G ----------------------

/// <summary>

/// Запускает основной цикл гибели и размножения:

/// 1. Генерирует начальную популяцию.

/// 2. Последовательно, столько раз, сколько поколений задано

/// проходит через стадии

/// а) воспроизводства (добавить 2Q потомков)

/// б) мутации (численность не меняется)

/// в) дуэлей (гибель 2Q потомков)

/// 3. На каждом шаге собирается статистика приспособленности.

/// </summary>

public void Run()

{

InitFirstGeneration();

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

{

Generation[i].Fitness = CalculateFitness(Generation[i].Organizm);

}

for (int i = 0; i < G; i++)//цикл по поколениям

{

#region Статистика приспособленности

Double BestFit = Double.MinValue;

Double WorseFit = Double.MaxValue;

Double AverageFit = 0;

foreach (MyOrganizm org in Generation)

{

if (org.Fitness > BestFit)

BestFit = org.Fitness;

if (org.Fitness < WorseFit)

WorseFit = org.Fitness;

AverageFit += org.Fitness;

}

best.Add(i, BestFit);

worse.Add(i, WorseFit);

average.Add(i, AverageFit / Z);

#endregion

#region Воспроизводство

for (int j = 0; j < Q && j< Z/2; j++)

{

int[] mask = new int[Z];

int indexA;

do

{

indexA = Generator.Next() % Z;

} while (mask[indexA] == 1);

mask[indexA] = 1;

int indexB;

do

{

indexB = Generator.Next() % Z;

} while (mask[indexB] == 1);

mask[indexB] = 1;

MyOrganizm childA = new MyOrganizm(),childB = new MyOrganizm();

childA.Organizm = (BitArray)Generation[indexA].Organizm.Clone();

childB.Organizm = (BitArray)Generation[indexB].Organizm.Clone();

this.Mix(childA, childB);

Generation.Add(childA);

Generation.Add(childB);

}

int a = 0;

#endregion

#region Мутация

for (int j = 0; j < Z; j++)

{

double p = Generator.NextDouble();

if (p < Pm)

{

double f0 = Generation[j].Fitness,f1;

MyOrganizm tmp = new MyOrganizm(),weak,powfl;

tmp.Organizm = (BitArray)Generation[j].Organizm.Clone();

this.Mutate(tmp);

f1 = tmp.Fitness;

if (f0 < f1)

{

weak = Generation[j];

powfl = tmp;

}

else

{

powfl = Generation[j];

weak = tmp;

}

p = Generator.NextDouble();

if (p < Pvm)

Generation[j] = powfl;

else

Generation[j] = weak;

}

}

#endregion

#region Дуэли

int Q2 = 2\*Q;

for (int j = 0; j < Q2; j++)

{

int indexA = Generator.Next() % Generation.Count;

MyOrganizm forceA = Generation[indexA];

Generation.RemoveAt(indexA);

int indexB = Generator.Next() % Generation.Count;

MyOrganizm forceB = Generation[indexB];

Generation.RemoveAt(indexB);

double f0 = forceA.Fitness, f1 = forceB.Fitness;

MyOrganizm weak, powfl;

if (f0 < f1)

{

weak = forceA;

powfl = forceB;

}

else

{

powfl = forceA;

weak = forceB;

}

double p = Generator.NextDouble();

if (p < Pv)

Generation.Add(powfl);

else

Generation.Add(weak);

}

#endregion

}

}

}

}

using System;

using System.Collections.Generic;

using System.Collections;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using ZedGraph;

using System.Threading;

namespace GenAlg

{

public partial class MainForm : Form

{

Solver solver;// = new Solver(0,0,200, 100, 400, 0.9, 0.95, 0.07);

public MainForm()

{

InitializeComponent();

//----------------------------------------------------------------

//Влючим всплывающие подсказки

zGraph.IsShowPointValues = true;

//Настроим внешний вид - заголовок и названия осей

zGraph.GraphPane.Title.Text = "Health";

zGraph.GraphPane.XAxis.Title.Text = "Generations";

zGraph.GraphPane.YAxis.Title.Text = "Health";

//solver.GenerateX(600);

//solver.GenerateC(200);

//solver.Run();

}

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

{

Int32 xSize, cSize, Z, Q, G;

Double Pm, Pvm, Pv;

#region parametres

//------------------ checking X

if (Int32.TryParse(xSinput.Text, out xSize))

{

//xSize = Int32.Parse(xSinput.Text);

if (!(xSize % 3 == 0))

{

MessageBox.Show("Мощность множества X должны быть кратна 3");

return;

}

}

else return;

//---------------- checking C

if (Int32.TryParse(cSinput.Text, out cSize))

{

//cSize = Int32.Parse(cSinput.Text);

//flag = (cSize > xSize / 3);

//if(!flag)

// MessageBox.Show("Суммарный размер С меньше Х, задача заведомо неразрешима");

}

else return;

//------------ checking G

if (Int32.TryParse(gInput.Text, out G))

{

//G = Int32.Parse(gInput.Text);

if (G < 1)

{

MessageBox.Show("Число популяций не должно быть меньше 1");

return;

}

}

else return;

//------------ checking Z

if (Int32.TryParse(zInput.Text, out Z))

{

//Z = Int32.Parse(zInput.Text);

if (Z < 1)

{

MessageBox.Show("Размер популяции - положительное число!");

return;

}

}

else return;

//---------- checking Q

if (Int32.TryParse(qInput.Text, out Q))

{

//Q = Int32.Parse(qInput.Text);

if (Q < 1)

{

MessageBox.Show("Q неправильное )");

return;

}

}

else return;

//------------------ Pvm

if (Double.TryParse(pvmInput.Text.Replace(".",","), out Pvm))

{

//Pvm = Double.Parse(pvmInput.Text.Replace(".", ","));

if (Pvm > 1.0 || Pvm < 0)

{

MessageBox.Show("Вероятность - число от 0 до 1");

return;

}

}

else return;

//------------------ Pm

if (Double.TryParse(pmInput.Text.Replace(".", ","), out Pm))

{

//Pm = Double.Parse(pmInput.Text.Replace(".", ","));

if (Pm > 1.0 || Pm < 0)

{

MessageBox.Show("Вероятность - число от 0 до 1");

return;

}

}

else return;

//------------------ Pvm

if (Double.TryParse(pvInput.Text.Replace(".", ","), out Pv))

{

//Pv = Double.Parse(pvInput.Text.Replace(".", ","));

if (Pv > 1.0 || Pv < 0)

{

MessageBox.Show("Вероятность - число от 0 до 1");

return;

}

}

else return;

#endregion

solver = new Solver(xSize, cSize, Z, Q, G, Pv, Pvm, Pm);

solver.Run();

DrawGraph(solver.Best, solver.Worse, solver.Average);

}

//-------------------------------------------------------------------

//---------------------- Graphics -----------------------

//---------------------- Drawing -----------------------

//---------------------- Zone -----------------------

//-----------------------------|-------------------------------------

//-----------------------------|-------------------------------------

//-----------------------------|-------------------------------------

//----------------------------\|/------------------------------------

//-----------------------------|-------------------------------------

/// <summary>

/// Отображает на графике кривые здоровья согласно переданным данным.

/// Каждый узел - пара координат "номер поколения - показатель здоровья"

/// </summary>

/// <param name="BestHealth">Лучшие показатели каждого поколения</param>

/// <param name="WorseHealth">Худшие показатели</param>

/// <param name="AverageHealth">Средние показатели</param>

private void DrawGraph(PointPairList BestHealth,

PointPairList WorseHealth,

PointPairList AverageHealth)

{

//Получим указатель на панель для рисования

GraphPane Pane = zGraph.GraphPane;

//Очистим список отображаемых кривых

Pane.CurveList.Clear();

//Добавим к списку отображения массивы точек, описывающие данные задачи

LineItem BestCurve = Pane.AddCurve("Best", BestHealth, Color.Green, SymbolType.None);

LineItem WorseCurve = Pane.AddCurve("Worse", WorseHealth, Color.Red, SymbolType.None);

LineItem AverageCurve = Pane.AddCurve("Average", AverageHealth, Color.Gray, SymbolType.None);

//Обновляем данные по осям, чтоб отобразились полные данные

zGraph.AxisChange();

//Обновляем график

zGraph.Invalidate();

}

/// <summary>

/// Обработчик события для отображения всплывающей подсказки

/// </summary>

/// <param name="sender">Элемент, вызвавший событие</param>

/// <param name="pane">Указатель на графическую панель, над которой находится курсор</param>

/// <param name="curve">Указатель на кривую, к которой курсор подведен</param>

/// <param name="iPt">Точка на кривой, ближе всего к которой находится курсор</param>

/// <returns></returns>

private string zGraph\_PointValueEvent(ZedGraphControl sender,

GraphPane pane, CurveItem curve, int iPt)

{

// Получим точку, около которой находимся

PointPair point = curve[iPt];

// Сформируем строку, в которой укажем имя кривой из легенды, а так же координаты точки

string result = string.Format("Curve: " + curve.Label.Text+", X: {0:F3}\nY: {1:F3}", point.X, point.Y);

return result;

}

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

{

}

//-------------------------------------------------------------------

}

}

namespace GenAlg

{

partial class MainForm

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.zGraph = new ZedGraph.ZedGraphControl();

this.RunButton = new System.Windows.Forms.Button();

this.parametres = new System.Windows.Forms.GroupBox();

this.label6 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label1 = new System.Windows.Forms.Label();

this.pvInput = new System.Windows.Forms.TextBox();

this.pvmInput = new System.Windows.Forms.TextBox();

this.pmInput = new System.Windows.Forms.TextBox();

this.qInput = new System.Windows.Forms.TextBox();

this.gInput = new System.Windows.Forms.TextBox();

this.zInput = new System.Windows.Forms.TextBox();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.label8 = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.cSinput = new System.Windows.Forms.TextBox();

this.xSinput = new System.Windows.Forms.TextBox();

this.parametres.SuspendLayout();

this.groupBox1.SuspendLayout();

this.SuspendLayout();

//

// zGraph

//

this.zGraph.AutoSize = true;

this.zGraph.BackColor = System.Drawing.SystemColors.ControlLightLight;

this.zGraph.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;

this.zGraph.Location = new System.Drawing.Point(12, 13);

this.zGraph.Name = "zGraph";

this.zGraph.ScrollGrace = 0;

this.zGraph.ScrollMaxX = 0;

this.zGraph.ScrollMaxY = 0;

this.zGraph.ScrollMaxY2 = 0;

this.zGraph.ScrollMinX = 0;

this.zGraph.ScrollMinY = 0;

this.zGraph.ScrollMinY2 = 0;

this.zGraph.Size = new System.Drawing.Size(572, 404);

this.zGraph.TabIndex = 0;

this.zGraph.TabStop = false;

this.zGraph.PointValueEvent += new ZedGraph.ZedGraphControl.PointValueHandler(this.zGraph\_PointValueEvent);

//

// RunButton

//

this.RunButton.Location = new System.Drawing.Point(590, 378);

this.RunButton.Name = "RunButton";

this.RunButton.Size = new System.Drawing.Size(281, 39);

this.RunButton.TabIndex = 1;

this.RunButton.Text = "Расчет";

this.RunButton.UseVisualStyleBackColor = true;

this.RunButton.Click += new System.EventHandler(this.RunButton\_Click);

//

// parametres

//

this.parametres.Controls.Add(this.label6);

this.parametres.Controls.Add(this.label5);

this.parametres.Controls.Add(this.label4);

this.parametres.Controls.Add(this.label3);

this.parametres.Controls.Add(this.label2);

this.parametres.Controls.Add(this.label1);

this.parametres.Controls.Add(this.pvInput);

this.parametres.Controls.Add(this.pvmInput);

this.parametres.Controls.Add(this.pmInput);

this.parametres.Controls.Add(this.qInput);

this.parametres.Controls.Add(this.gInput);

this.parametres.Controls.Add(this.zInput);

this.parametres.Location = new System.Drawing.Point(590, 88);

this.parametres.Name = "parametres";

this.parametres.Size = new System.Drawing.Size(281, 284);

this.parametres.TabIndex = 2;

this.parametres.TabStop = false;

this.parametres.Text = "Параметры алгоритма";

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(6, 211);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(204, 13);

this.label6.TabIndex = 1;

this.label6.Text = "Вероятность победы сильного в дуэли";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(6, 172);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(249, 13);

this.label5.TabIndex = 1;

this.label5.Text = "Вероятность выживания сильного при мутации";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(6, 133);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(117, 13);

this.label4.TabIndex = 1;

this.label4.Text = "Вероятность мутации";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(6, 94);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(160, 13);

this.label3.TabIndex = 1;

this.label3.Text = "Скрещиваний на одном цикле";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(6, 55);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(126, 13);

this.label2.TabIndex = 1;

this.label2.Text = "Количество поколений ";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(6, 16);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(105, 13);

this.label1.TabIndex = 1;

this.label1.Text = "Размер популяции ";

//

// pvInput

//

this.pvInput.Location = new System.Drawing.Point(6, 227);

this.pvInput.Name = "pvInput";

this.pvInput.Size = new System.Drawing.Size(261, 20);

this.pvInput.TabIndex = 0;

this.pvInput.Text = "0.94";

//

// pvmInput

//

this.pvmInput.Location = new System.Drawing.Point(6, 188);

this.pvmInput.Name = "pvmInput";

this.pvmInput.Size = new System.Drawing.Size(261, 20);

this.pvmInput.TabIndex = 0;

this.pvmInput.Text = "0.94";

//

// pmInput

//

this.pmInput.Location = new System.Drawing.Point(6, 149);

this.pmInput.Name = "pmInput";

this.pmInput.Size = new System.Drawing.Size(261, 20);

this.pmInput.TabIndex = 0;

this.pmInput.Text = "0.008";

//

// qInput

//

this.qInput.Location = new System.Drawing.Point(6, 110);

this.qInput.Name = "qInput";

this.qInput.Size = new System.Drawing.Size(261, 20);

this.qInput.TabIndex = 0;

this.qInput.Text = "100";

//

// gInput

//

this.gInput.Location = new System.Drawing.Point(6, 71);

this.gInput.Name = "gInput";

this.gInput.Size = new System.Drawing.Size(261, 20);

this.gInput.TabIndex = 0;

this.gInput.Text = "100";

//

// zInput

//

this.zInput.Location = new System.Drawing.Point(6, 32);

this.zInput.Name = "zInput";

this.zInput.Size = new System.Drawing.Size(261, 20);

this.zInput.TabIndex = 0;

this.zInput.Text = "200";

//

// groupBox1

//

this.groupBox1.Controls.Add(this.label8);

this.groupBox1.Controls.Add(this.label7);

this.groupBox1.Controls.Add(this.cSinput);

this.groupBox1.Controls.Add(this.xSinput);

this.groupBox1.Location = new System.Drawing.Point(590, 13);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(280, 69);

this.groupBox1.TabIndex = 3;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Параметры задачи";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(6, 42);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(177, 13);

this.label8.TabIndex = 1;

this.label8.Text = "Количество троек в множестве С";

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(6, 16);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(116, 13);

this.label7.TabIndex = 1;

this.label7.Text = "Размер множества Х";

//

// cSinput

//

this.cSinput.Location = new System.Drawing.Point(189, 35);

this.cSinput.Name = "cSinput";

this.cSinput.Size = new System.Drawing.Size(78, 20);

this.cSinput.TabIndex = 0;

this.cSinput.Text = "333";

//

// xSinput

//

this.xSinput.Location = new System.Drawing.Point(189, 9);

this.xSinput.Name = "xSinput";

this.xSinput.Size = new System.Drawing.Size(78, 20);

this.xSinput.TabIndex = 0;

this.xSinput.Text = "900";

//

// MainForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(878, 431);

this.Controls.Add(this.groupBox1);

this.Controls.Add(this.parametres);

this.Controls.Add(this.RunButton);

this.Controls.Add(this.zGraph);

this.Name = "MainForm";

this.Text = "Сис. ИИ - Лаб. раб. №1";

this.Load += new System.EventHandler(this.MainForm\_Load);

this.parametres.ResumeLayout(false);

this.parametres.PerformLayout();

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private ZedGraph.ZedGraphControl zGraph;

private System.Windows.Forms.Button RunButton;

private System.Windows.Forms.GroupBox parametres;

private System.Windows.Forms.TextBox pvInput;

private System.Windows.Forms.TextBox pvmInput;

private System.Windows.Forms.TextBox pmInput;

private System.Windows.Forms.TextBox qInput;

private System.Windows.Forms.TextBox gInput;

private System.Windows.Forms.TextBox zInput;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.TextBox cSinput;

private System.Windows.Forms.TextBox xSinput;

}

}

# 2. Нейронные сети. Обучение методом обратного распространения ошибки

# 7. Реализовать модификации алгоритмов обучения адаптивный метод и метод моментов.

**Результаты исследований:**

В ходе многочисленных тестов было обнаружено, что на качество обучения нейронной сети влияет первоначальная настройка весов. При генерации случайных весов в диапазоне [-0.5,0.5] качество обучения намного выше, сходимость намного быстрее, чем при начальных весах в диапазоне [0,1].

Однако наиболее серьезный прогресс в обучении нейросети дает добавление к каждому слою bias-node:



*Рисунок 1 – нейронная сеть, слои которой содержат bias node (обозначен +1)*

График обучения битовых образов 6x6:

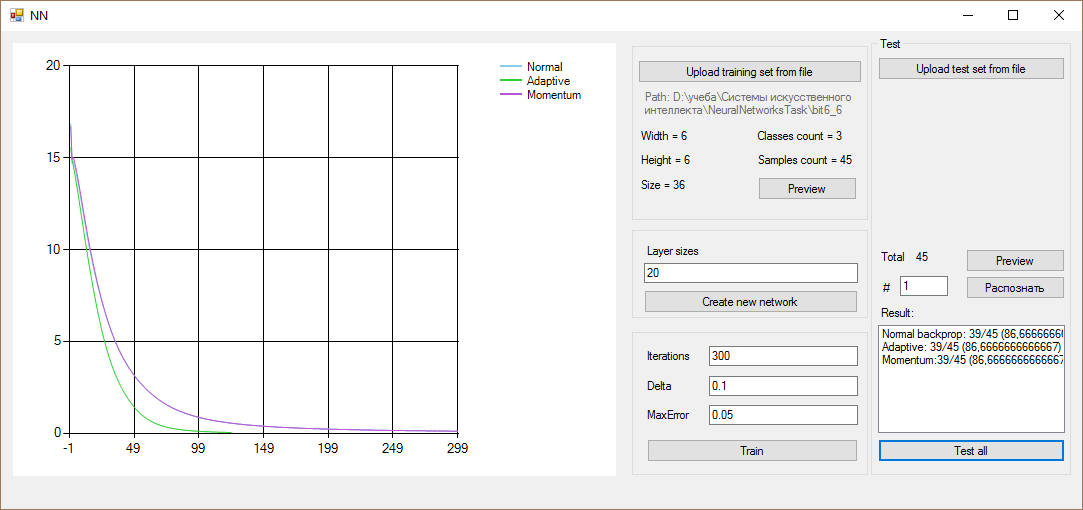


График обучения битовых цифр 16\*20:

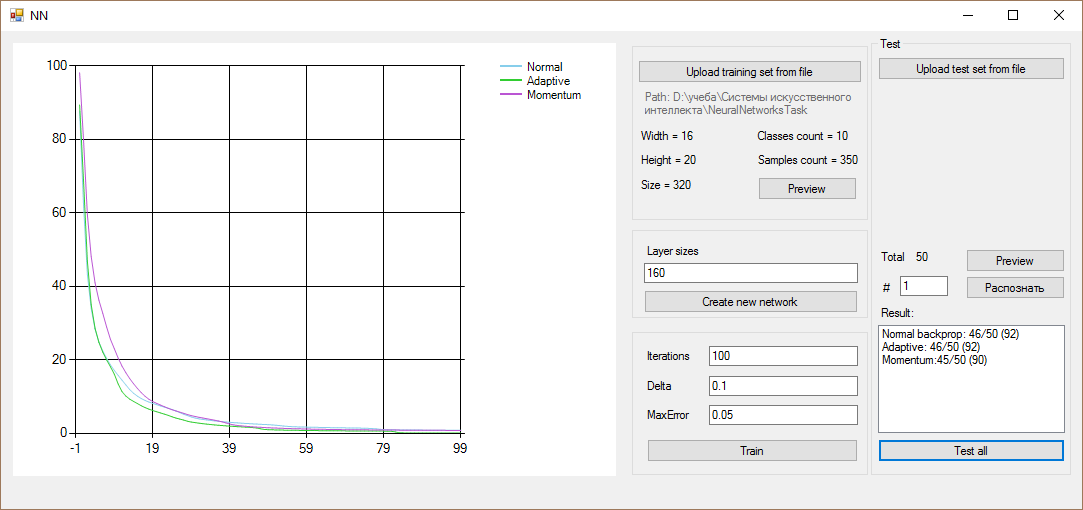
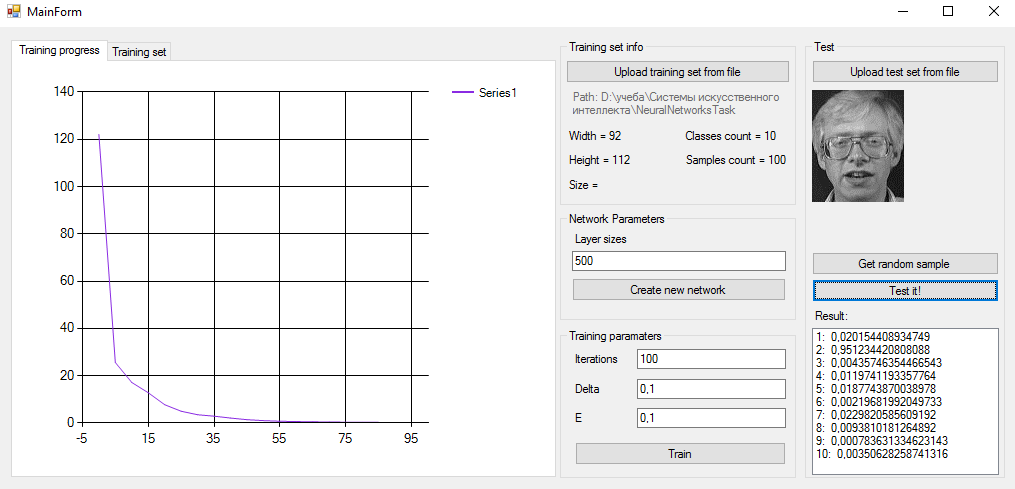
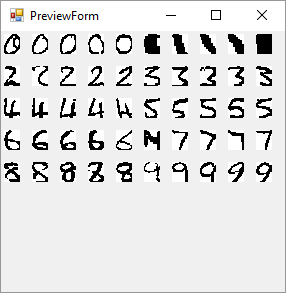


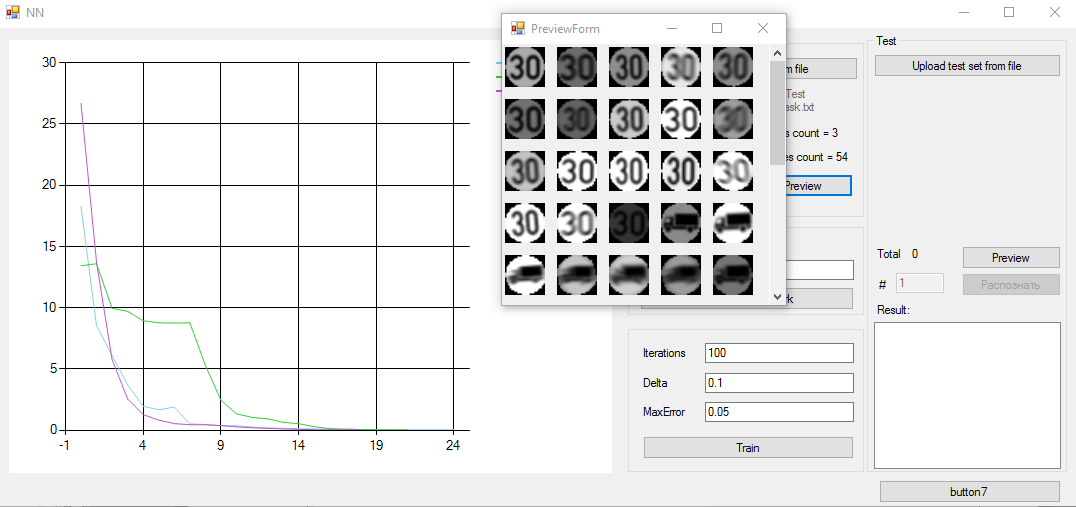
График обучения лиц, 10 классов



Предпросмотр тестовых данных:



Пример обучения нейросети для распознавания дорожных знаков



Исходный код программы:

using nn.classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace nn

{

public partial class MainForm : Form

{

private TrainingSet \_trainingSet;

private TrainingSet \_testSet;

private NeuralNetwork \_networkNormal, \_networkAdaptive, \_networkMomentum;

private int \_indexToTest;

private static Random \_randomGenerator = new Random();

private int \_step = 1;

public MainForm()

{

InitializeComponent();

}

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

{

//var nw = new NeuralNetwork("normal.xml");

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog() { Filter = @"Text files (\*.txt)|\*.txt" })

{

if(openFileDialog.ShowDialog() == DialogResult.OK)

{

\_trainingSet = new TrainingSet(openFileDialog.FileName, \_step);

widthLabel.Text = string.Format("Width = {0}", \_trainingSet.SampleWidth);

heightLabel.Text = string.Format("Height = {0}", \_trainingSet.SampleHeight);

sizeLabel.Text = string.Format("Size = {0}", \_trainingSet.SampleSize);

classesLabel.Text = string.Format("Classes count = {0}", \_trainingSet.ClassesCount);

countLabel.Text = string.Format("Samples count = {0}", \_trainingSet.SamplesCount);

richTextBox1.Text = string.Format("Path: {0}", openFileDialog.FileName);

//DrawBitmaps();

textBox1.Text = String.Empty;

groupBox2.Enabled = true;

groupBox3.Enabled = false;

groupBox4.Enabled = false;

button5.Enabled = true;

button4.Enabled = vectorNumber.Enabled = false;

}

}

}

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

{

if(\_trainingSet != null)

{

string layers = textBox1.Text;

string[] buffer = layers.Split(new Char[] { ' ', ',', ';' }, StringSplitOptions.RemoveEmptyEntries);

int[] networkParams = new int[buffer.Length + 1];

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

{

int.TryParse(buffer[i], out networkParams[i]);

}

networkParams[buffer.Length] = \_trainingSet.ClassesCount;

//\_networkNormal = new NeuralNetwork("normal.xml");

//\_networkAdaptive = new NeuralNetwork("ad.xml");

//\_networkMomentum = new NeuralNetwork("mom.xml");

\_networkNormal = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Normal };

\_networkAdaptive = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Adaptive };

\_networkMomentum = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Momentum };

//\_networkNormal.Save("normal.xml");

groupBox3.Enabled = true;

groupBox4.Enabled = false;

}

}

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

{

int iterationsCount;

double e0, delta;

int.TryParse(textBox2.Text, out iterationsCount);

double.TryParse(textBox3.Text, NumberStyles.Any, CultureInfo.InvariantCulture, out delta);

double.TryParse(textBox4.Text, NumberStyles.Any, CultureInfo.InvariantCulture, out e0);

chart1.Series[0].Points.Clear();

\_networkNormal.Delta

= \_networkAdaptive.Delta

= \_networkMomentum.Delta

= delta;

TaskFactory factory = new TaskFactory();

factory.StartNew(() =>

{

\_networkNormal.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[0],

1);

});

factory.StartNew(() =>

{

\_networkAdaptive.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[1],

1);

});

factory.StartNew(() =>

{

\_networkMomentum.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[2],

1);

});

groupBox4.Enabled = true;

//forSisII1.Enabled = button4.Enabled = true;

}

private void button3\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog() { Filter = @"Text files (\*.txt)|\*.txt" })

{

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

\_testSet = new TrainingSet(openFileDialog.FileName, \_step);

if (\_testSet.SampleHeight != \_trainingSet.SampleHeight

|| \_testSet.SampleWidth != \_trainingSet.SampleWidth)

{

MessageBox.Show("Test samples have wrong size", "Invalid test set", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (\_testSet.ClassesCount != \_trainingSet.ClassesCount)

{

MessageBox.Show("Classes count doesn't match", "Invalid test set", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

total.Text = \_testSet.SamplesCount.ToString();

vectorNumber.Enabled = true;

button4.Enabled = true;

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

\_indexToTest = int.Parse(vectorNumber.Text);// \_randomGenerator.Next(\_testSet.SamplesCount);

System.Drawing.Bitmap bmp = new System.Drawing.Bitmap(\_testSet.SampleWidth, \_testSet.SampleHeight);

for (int x = 0; x < bmp.Height; ++x)

{

for (int y = 0; y < bmp.Width; ++y)

{

//bmp.SetPixel(x, y, Color.White);

if (\_trainingSet.IsBinary)

{

int val = (int)\_testSet.Samples[\_indexToTest][x \* \_trainingSet.SampleWidth + y];

bmp.SetPixel(y, x, val > 0 ? Color.Black : Color.White);

}

else

{

int val = (int)\_testSet.Samples[\_indexToTest][x \* \_trainingSet.SampleWidth + y];

Color color = Color.FromArgb(val, val, val);

bmp.SetPixel(y, x, Color.FromArgb(val, val, val));

}

}

}

pictureBox1.Image = bmp;

button4.Enabled = true;

}

private void button4\_Click(object sender, EventArgs e)

{

button5\_Click(sender, e);

double[] netIn = \_testSet.Samples[\_indexToTest];

double[] result = new double[\_testSet.ClassesCount];

\_networkNormal.NetworkOut(netIn, out result);

listBox1.Items.Clear();

listBox1.Items.Add("n");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

listBox1.Items.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

\_networkAdaptive.NetworkOut(netIn, out result);

listBox1.Items.Add("a");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

listBox1.Items.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

\_networkMomentum.NetworkOut(netIn, out result);

listBox1.Items.Add("m");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

}

private void button7\_Click(object sender, EventArgs e)

{

int nCount, aCount, mCount;

nCount = aCount = mCount = 0;

for (int i = 0; i < \_testSet.SamplesCount; i++)

{

double[] nAns;

double[] aAns;

double[] mAns;

int nClass = \_networkNormal.NetworkOut(\_testSet.Samples[i], out nAns);

if(\_testSet.Answers[i][nClass] > 0.99)

{

nCount++;

}

int aClass = \_networkAdaptive.NetworkOut(\_testSet.Samples[i], out aAns);

if (\_testSet.Answers[i][aClass] > 0.99)

{

aCount++;

}

int mClass = \_networkMomentum.NetworkOut(\_testSet.Samples[i], out mAns);

if (\_testSet.Answers[i][mClass] > 0.99)

{

mCount++;

}

}

listBox1.Items.Clear();

listBox1.Items.Add(string.Format("Normal backprop: {0}/{1} ({2})", nCount, \_testSet.SamplesCount,

(100.0\*nCount)/\_testSet.SamplesCount));

listBox1.Items.Add(string.Format("Adaptive: {0}/{1} ({2})", aCount, \_testSet.SamplesCount,

(100.0 \* aCount) / \_testSet.SamplesCount));

listBox1.Items.Add(string.Format("Momentum:{0}/{1} ({2})", mCount, \_testSet.SamplesCount,

(100.0 \* mCount) / \_testSet.SamplesCount));

}

private void button5\_Click\_1(object sender, EventArgs e)

{

using (var PreviewForm = new PreviewForm(\_trainingSet))

{

PreviewForm.ShowDialog();

}

}

private void button6\_Click(object sender, EventArgs e)

{

using (var PreviewForm = new PreviewForm(\_testSet))

{

PreviewForm.ShowDialog();

}

}

}

}using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Xml;

using Visn = System.Windows.Forms.DataVisualization;

namespace nn.classes

{

class Sigmoid

{

#region static members

public static readonly double betta = 1.0;

/// <summary>

/// сигмоидальная функция нейрона

/// </summary>

/// <param name="input">Вход</param>

/// <returns>Значение Sg(x)</returns>

public static double Sg(double input)

{

return (1 / (1 + Math.Exp(-betta \* input)));

}

#endregion

}

/// <summary>

/// Класс, описывающий нейронную сеть,

/// которая может состоять из множества слоев.

/// Число входов, выходов и размеры слоев задаются.

/// </summary>

class NeuralNetwork

{

#region fields

/// <summary>

/// массив слоев сети

/// </summary>

private NeuralLayer[] layers;

/// <summary>

/// число входов, выходов, слоев

/// </summary>

private int IN=0, OUT=0, layersCount=0;

private double delta;

private Method \_method;

#endregion

#region methods

/// <summary>

/// Генерирует полносвязную многослойную

/// нейросеть

/// </summary>

/// <param name="inputs">Число входных сигналов</param>

/// <param name="layersSizes">Набор параметров переменной длины,

/// содержит размеры слоев сети</param>

public NeuralNetwork(int inputs, int[] layersSizes)

{

//число слоев определяется длиной массива с их размерами

layersCount = layersSizes.Length;

//выделим память под слои

layers = new NeuralLayer[layersCount];

//текущее число входных сигналов.

//сначала равно числу входов всей сети

int currentInputs = inputs;

//создадим слои сети

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

{

//создадим слой с числом входов currentInputs

//и числом нейронов (и выходов) равным его размеру

layers[i] = new NeuralLayer(currentInputs, layersSizes[i]);

//сгенерируем синапсы (весовые коэффициенты)

//случайным образом

layers[i].GenerateWeights();

//для следующего слоя размер текущего

//станет числом входов

currentInputs = layersSizes[i];

}

//число входов известно

IN = inputs;

//число выходов = размер последнего слоя

OUT = currentInputs;

}

/// <summary>

/// Расчитывает выходной вектор для нейросети

/// </summary>

/// <param name="inX">Вектор входный сигналов</param>

/// <param name="outY">Вектор выхода</param>

public int NetworkOut(double[] inX, out double[] outY)

{

double[] input = inX;

//по всем слоям сети

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

{

double[] output;

double[] currentInput;

output = new double[layers[i].OutputsCount];

currentInput = new double[layers[i].OutputsCount];

//по всем нейронам в слое

for (int j = 0; j < layers[i].OutputsCount; j++)

{

//расчитаем входное значение

double arg = 0.0;

//по всем входам нейрона

for (int k = 0; k < layers[i].InputsCount+1; k++)

{

//на каждом шаге надо добавить очередной "вклад"

//для j нейрона в i слое по k входу:

if (k == layers[i].InputsCount)

{//bias

arg += 1 \* layers[i][k, j];

}

else

{

arg += input[k] \* layers[i][k, j];

}

}

currentInput[j] = arg;

//выход нейрона = сигмоидальная функция от входного значения

output[j] = Sigmoid.Sg(currentInput[j]);

}

//сохраним состояние слоя

layers[i].LastOut = output;

layers[i].LastIn = currentInput;

//выход текущего слоя становится входом для следующего

input = output;

}

//когда прошли через выходной слой, в input попал как раз

//его выходной вектор возбуждения

outY = input;

int maxIndex = -1;

double max = double.MinValue;

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

{

if(outY[i] > max)

{

max = outY[i];

maxIndex = i;

}

}

return maxIndex;

}

/// <summary>

/// Расчет ошибки сети

/// </summary>

/// <param name="realVector">Вектор возбуждения выходных нейронов</param>

/// <param name="ideal">"Идеальный" вектор - желаемый ответ сети</param>

/// <returns></returns>

private double E(double[] realVector, double[] ideal)

{

double Err = 0.0;

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

Err += (realVector[i] - ideal[i])\* (realVector[i] - ideal[i]);

return Err / 2;

}

/// <summary>

/// Настройка весов под вектора входа и желаемого выхода

/// </summary>

/// <param name="ideal"></param>

/// <param name="inX"></param>

private void CorrectError(double[] ideal, double[] inX,Method method)

{

//надо пройти по всем слоям и посчитать производные

for (int i = layersCount - 1; i >= 0; i--)

{

//Если слой выходной, считаем ошибку по выходу

//из желаемого вектора

if (i == layersCount - 1)

layers[i].CountOutDetivativesAsLast(ideal);

//для промежуточных слоев - используем исходящие веса и

//производные по входам для исходящих связей

else

layers[i].CountOutDerivatives(layers[i + 1].DEDX, layers[i + 1].W);

//перепрыгиваем с выхода на вход

layers[i].CountInDerivatives();

//посчитаем производные по слою

//если это входной слой, то для него входным вектором служит inX

if (i == 0)

layers[i].CountWeightDerivatives(inX);

//Если слой промежуточный, то на его входе "висит"

//выход предыдущего

else

layers[i].CountWeightDerivatives(layers[i - 1].LastOut);

}

//Все dE/dW посчитаны, обновим веса

for (int i = layersCount - 1; i >= 0; i--)

{

if (\_method == Method.Momentum)

{

layers[i].UpdateWithMomentum(delta);

}

else

{

layers[i].Update(delta);

}

}

}

/// <summary>

/// Функция обучения нейросети

/// </summary>

/// <param name="sample">Обучающая выборка</param>

/// <param name="ideals">Идеальные вектора</param>

/// <param name="Epochs">Число итераций</param>

/// <param name="e0">Порог допустимой ошибки</param>

/// <param name="sampleSize">Размер выборки</param>

/// <returns>"Successfull" - если за Epochs итераций ошибка стала меньше порогового значения

/// "Unsuccessfull" - если по прошествии Epochs итераций ошибка все еще больше пороговой</returns>

public String Train(double[][] sample, double[][] ideals, int Epochs, double e0,

int sampleSize, Visn.Charting.Chart chart, Visn.Charting.Series series, int displayStep)

{

chart.Invoke((MethodInvoker)delegate {

series.Points.Clear(); // runs on UI thread

});

// = new Visn.Charting.Series();

//накопленная ошибка итерации

double EpochError=0;

double oldError;

//номер текущей итерации

int epoch;

//цикл по всем итерациям

for (epoch = 0; epoch < Epochs; epoch++)

{

oldError = EpochError;

//обнулить накопленную ошибку

EpochError = .0;

if (Method == Method.Adaptive && epoch > 0)

{

foreach (var layer in layers)

{

layer.SaveState();

}

}

//по всем векторам выборки

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

{

//возьмем текущий вектор выборки

//и идеальный для него выходной вектор

double[] Sample = sample[i];

double[] Ideal = ideals[i];

double[] nwOut;

//пропустим вектор из выборки через сеть

this.NetworkOut(Sample, out nwOut);

//расчитаем ошибку сети для данного элемента выборки

EpochError += E(nwOut, Ideal);

//скорректируем веса методом backpropagation

CorrectError(Ideal,Sample,\_method);

}

if (Method == Method.Adaptive && epoch>0)

{

if (EpochError < oldError)

{

delta = delta \* 1.02;

}

if (EpochError > oldError \* 1.04)

{

foreach (var layer in layers)

{

layer.RestoreState();

}

delta = delta \* 0.7;

}

}

//if (epoch % displayStep == 0)

{

chart.Invoke((MethodInvoker)delegate {

series.Points.AddXY(epoch, EpochError);

chart.Invalidate(chart.ClientRectangle);

chart.Update();

});

}

//если после просмотра всей выборки ошибка меньше минимальной

//можно уже сейчас прекратить обучение

if (EpochError < e0) break;

}

//проверим, по какому условию завершился цикл обучения

if (epoch < Epochs) return "Successfull";

if (EpochError > e0) return "Unsuccessfull";

return "Finished";

}

#endregion

public double Delta { get { return delta; } set { delta = value; } }

public Method Method

{

get

{

return \_method;

}

set

{

\_method = value;

}

}

public void Save(string fileName)

{

using (XmlWriter writer = XmlWriter.Create(fileName))

{

writer.WriteStartElement("NeuralNetwork");

writer.WriteElementString("LayerCount", layersCount.ToString());

writer.WriteElementString("ClassesCount", layers[layersCount - 1].OutputsCount.ToString());

writer.WriteStartElement("Layers");

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

{

writer.WriteElementString("LayerInputs", layers[i].InputsCount.ToString());

}

writer.WriteEndElement();

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

{

writer.WriteElementString("Layer", layers[i].GetWeightsStringRepr().ToString());

}

writer.WriteEndElement();

}

}

public NeuralNetwork(string fileName)

{

using (XmlReader reader = XmlReader.Create(fileName))

{

List<int> szList = new List<int>();

reader.Read();

reader.Read();

if (reader.Name != "NeuralNetwork") return;

reader.Read();

this.layersCount = reader.ReadElementContentAsInt();

this.OUT = reader.ReadElementContentAsInt();

this.layers = new NeuralLayer[layersCount + 1];

reader.Read();

if (reader.Name != "LayerInputs") return;

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

{

szList.Add(reader.ReadElementContentAsInt());

}

szList.Add(OUT);

reader.Read();

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

{

layers[i] = new NeuralLayer(szList[i], szList[i + 1]);

}

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

{

layers[i].Parse(reader.ReadElementContentAsString());

}

}

}

}

public enum Method

{

Normal,

Adaptive,

Momentum,

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

namespace nn.classes

{

public class TrainingSet

{

private static Char [] separators = new Char[] { ' ' };

private int \_fullSamplesCount;

private int \_step;

public int SampleWidth { get; private set; }

public int SampleHeight { get; private set; }

public int SampleSize

{

get; private set;

}

public int SamplesCount { get; private set; }

public bool IsBinary { get; private set; }

public double[][] Samples { get; private set; }

public double[][] Answers { get; private set; }

public int ClassesCount { get; private set; }

public TrainingSet(string filePath, int step)

{

\_step = step;

using (FileStream fileStream = new FileStream(filePath, FileMode.Open))

{

using (StreamReader reader = new StreamReader(fileStream))

{

string trainingInfoString = reader.ReadLine();

ParseInfo(trainingInfoString);

Prepare();

for (int sampleIndex = 0; sampleIndex < \_fullSamplesCount; sampleIndex++)

{

//parse one sample

StringBuilder sampleStringBuilder = new StringBuilder();

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

{

sampleStringBuilder.Append(" ");

sampleStringBuilder.Append(reader.ReadLine());

}

if (sampleIndex % step == 0)

{

string sampleContent = sampleStringBuilder.ToString().Trim();

string[] sampleValuesBuffer = sampleContent.Split(separators, StringSplitOptions.RemoveEmptyEntries);

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

{

double currentValue;

double.TryParse(sampleValuesBuffer[i], out currentValue);

Samples[sampleIndex / \_step][i] = currentValue;

}

}

int expectedClass;

int.TryParse(reader.ReadLine(), out expectedClass);

if (sampleIndex % step == 0)

{

Answers[sampleIndex / \_step][expectedClass - 1] = 1.0;

}

}

}

}

}

private void Prepare()

{

Samples = new double[SamplesCount][];

Answers = new double[SamplesCount][];

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

{

Samples[i] = new double[SampleSize];

Answers[i] = new double[ClassesCount];

}

}

private void ParseInfo(string trainingInfoString)

{

int binaryFlag, width, height, classesCount, samplesCount;

string[] infoBuffer = trainingInfoString.Split(separators,

StringSplitOptions.RemoveEmptyEntries);

int.TryParse(infoBuffer[0], out binaryFlag);

IsBinary = binaryFlag == 0;

int.TryParse(infoBuffer[1], out width);

int.TryParse(infoBuffer[2], out height);

int.TryParse(infoBuffer[3], out classesCount);

int.TryParse(infoBuffer[4], out samplesCount);

SampleWidth = width;

SampleHeight = height;

ClassesCount = classesCount;

\_fullSamplesCount = samplesCount;

SamplesCount = samplesCount / \_step;

if (\_step > 1 && samplesCount % \_step > 0)

{

SamplesCount += 1;

}

SampleSize = SampleWidth \* SampleHeight;

}

}

}

namespace nn

{

partial class MainForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new System.Windows.Forms.DataVisualization.Charting.ChartArea();

System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new System.Windows.Forms.DataVisualization.Charting.Legend();

System.Windows.Forms.DataVisualization.Charting.Series series1 = new System.Windows.Forms.DataVisualization.Charting.Series();

System.Windows.Forms.DataVisualization.Charting.Series series2 = new System.Windows.Forms.DataVisualization.Charting.Series();

System.Windows.Forms.DataVisualization.Charting.Series series3 = new System.Windows.Forms.DataVisualization.Charting.Series();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.button5 = new System.Windows.Forms.Button();

this.richTextBox1 = new System.Windows.Forms.RichTextBox();

this.countLabel = new System.Windows.Forms.Label();

this.classesLabel = new System.Windows.Forms.Label();

this.pathLabel = new System.Windows.Forms.Label();

this.sizeLabel = new System.Windows.Forms.Label();

this.heightLabel = new System.Windows.Forms.Label();

this.widthLabel = new System.Windows.Forms.Label();

this.uploadButton = new System.Windows.Forms.Button();

this.groupBox2 = new System.Windows.Forms.GroupBox();

this.button1 = new System.Windows.Forms.Button();

this.label1 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.groupBox3 = new System.Windows.Forms.GroupBox();

this.button2 = new System.Windows.Forms.Button();

this.textBox4 = new System.Windows.Forms.TextBox();

this.label4 = new System.Windows.Forms.Label();

this.textBox3 = new System.Windows.Forms.TextBox();

this.label3 = new System.Windows.Forms.Label();

this.textBox2 = new System.Windows.Forms.TextBox();

this.label2 = new System.Windows.Forms.Label();

this.groupBox4 = new System.Windows.Forms.GroupBox();

this.button6 = new System.Windows.Forms.Button();

this.total = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.label6 = new System.Windows.Forms.Label();

this.vectorNumber = new System.Windows.Forms.TextBox();

this.label5 = new System.Windows.Forms.Label();

this.listBox1 = new System.Windows.Forms.ListBox();

this.button4 = new System.Windows.Forms.Button();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.button3 = new System.Windows.Forms.Button();

this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();

this.button7 = new System.Windows.Forms.Button();

this.groupBox1.SuspendLayout();

this.groupBox2.SuspendLayout();

this.groupBox3.SuspendLayout();

this.groupBox4.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();

this.SuspendLayout();

//

// groupBox1

//

this.groupBox1.Controls.Add(this.button5);

this.groupBox1.Controls.Add(this.richTextBox1);

this.groupBox1.Controls.Add(this.countLabel);

this.groupBox1.Controls.Add(this.classesLabel);

this.groupBox1.Controls.Add(this.pathLabel);

this.groupBox1.Controls.Add(this.sizeLabel);

this.groupBox1.Controls.Add(this.heightLabel);

this.groupBox1.Controls.Add(this.widthLabel);

this.groupBox1.Controls.Add(this.uploadButton);

this.groupBox1.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.groupBox1.Location = new System.Drawing.Point(631, 9);

this.groupBox1.Margin = new System.Windows.Forms.Padding(0);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(236, 181);

this.groupBox1.TabIndex = 0;

this.groupBox1.TabStop = false;

//

// button5

//

this.button5.Enabled = false;

this.button5.Location = new System.Drawing.Point(126, 137);

this.button5.Name = "button5";

this.button5.Size = new System.Drawing.Size(99, 23);

this.button5.TabIndex = 9;

this.button5.Text = "Preview";

this.button5.UseVisualStyleBackColor = true;

this.button5.Click += new System.EventHandler(this.button5\_Click\_1);

//

// richTextBox1

//

this.richTextBox1.BackColor = System.Drawing.SystemColors.Control;

this.richTextBox1.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.richTextBox1.Enabled = false;

this.richTextBox1.Location = new System.Drawing.Point(12, 50);

this.richTextBox1.Name = "richTextBox1";

this.richTextBox1.ScrollBars = System.Windows.Forms.RichTextBoxScrollBars.None;

this.richTextBox1.Size = new System.Drawing.Size(214, 27);

this.richTextBox1.TabIndex = 7;

this.richTextBox1.Text = "";

//

// countLabel

//

this.countLabel.AutoSize = true;

this.countLabel.Location = new System.Drawing.Point(123, 113);

this.countLabel.Name = "countLabel";

this.countLabel.Size = new System.Drawing.Size(89, 13);

this.countLabel.TabIndex = 6;

this.countLabel.Text = "Samples count = ";

//

// classesLabel

//

this.classesLabel.AutoSize = true;

this.classesLabel.Location = new System.Drawing.Point(123, 89);

this.classesLabel.Name = "classesLabel";

this.classesLabel.Size = new System.Drawing.Size(85, 13);

this.classesLabel.TabIndex = 5;

this.classesLabel.Text = "Classes count = ";

//

// pathLabel

//

this.pathLabel.AutoSize = true;

this.pathLabel.Location = new System.Drawing.Point(9, 50);

this.pathLabel.Name = "pathLabel";

this.pathLabel.Size = new System.Drawing.Size(35, 13);

this.pathLabel.TabIndex = 4;

this.pathLabel.Text = "Path: ";

//

// sizeLabel

//

this.sizeLabel.Location = new System.Drawing.Point(6, 138);

this.sizeLabel.Name = "sizeLabel";

this.sizeLabel.Size = new System.Drawing.Size(67, 14);

this.sizeLabel.TabIndex = 3;

this.sizeLabel.Text = "Size =";

//

// heightLabel

//

this.heightLabel.AutoSize = true;

this.heightLabel.Location = new System.Drawing.Point(6, 113);

this.heightLabel.Name = "heightLabel";

this.heightLabel.Size = new System.Drawing.Size(50, 13);

this.heightLabel.TabIndex = 2;

this.heightLabel.Text = "Height = ";

//

// widthLabel

//

this.widthLabel.AutoSize = true;

this.widthLabel.Location = new System.Drawing.Point(6, 89);

this.widthLabel.Name = "widthLabel";

this.widthLabel.Size = new System.Drawing.Size(44, 13);

this.widthLabel.TabIndex = 1;

this.widthLabel.Text = "Width =";

//

// uploadButton

//

this.uploadButton.Location = new System.Drawing.Point(6, 20);

this.uploadButton.Name = "uploadButton";

this.uploadButton.Size = new System.Drawing.Size(224, 23);

this.uploadButton.TabIndex = 0;

this.uploadButton.Text = "Upload training set from file";

this.uploadButton.UseVisualStyleBackColor = true;

this.uploadButton.Click += new System.EventHandler(this.uploadButton\_Click);

//

// groupBox2

//

this.groupBox2.Controls.Add(this.button1);

this.groupBox2.Controls.Add(this.label1);

this.groupBox2.Controls.Add(this.textBox1);

this.groupBox2.Enabled = false;

this.groupBox2.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.groupBox2.Location = new System.Drawing.Point(631, 193);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(236, 95);

this.groupBox2.TabIndex = 1;

this.groupBox2.TabStop = false;

//

// button1

//

this.button1.Location = new System.Drawing.Point(12, 66);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(214, 23);

this.button1.TabIndex = 2;

this.button1.Text = "Create new network";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(12, 20);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(59, 13);

this.label1.TabIndex = 1;

this.label1.Text = "Layer sizes";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(12, 39);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(214, 20);

this.textBox1.TabIndex = 0;

//

// groupBox3

//

this.groupBox3.Controls.Add(this.button2);

this.groupBox3.Controls.Add(this.textBox4);

this.groupBox3.Controls.Add(this.label4);

this.groupBox3.Controls.Add(this.textBox3);

this.groupBox3.Controls.Add(this.label3);

this.groupBox3.Controls.Add(this.textBox2);

this.groupBox3.Controls.Add(this.label2);

this.groupBox3.Enabled = false;

this.groupBox3.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.groupBox3.Location = new System.Drawing.Point(631, 295);

this.groupBox3.Name = "groupBox3";

this.groupBox3.Size = new System.Drawing.Size(236, 150);

this.groupBox3.TabIndex = 2;

this.groupBox3.TabStop = false;

//

// button2

//

this.button2.Location = new System.Drawing.Point(15, 113);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(211, 23);

this.button2.TabIndex = 6;

this.button2.Text = "Train";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(77, 79);

this.textBox4.Name = "textBox4";

this.textBox4.Size = new System.Drawing.Size(149, 20);

this.textBox4.TabIndex = 5;

this.textBox4.Text = "0.05";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(12, 82);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(49, 13);

this.label4.TabIndex = 4;

this.label4.Text = "MaxError";

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(77, 50);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(149, 20);

this.textBox3.TabIndex = 3;

this.textBox3.Text = "0.1";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(12, 53);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(32, 13);

this.label3.TabIndex = 2;

this.label3.Text = "Delta";

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(77, 20);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(149, 20);

this.textBox2.TabIndex = 1;

this.textBox2.Text = "100";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(12, 23);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(50, 13);

this.label2.TabIndex = 0;

this.label2.Text = "Iterations";

//

// groupBox4

//

this.groupBox4.Controls.Add(this.button7);

this.groupBox4.Controls.Add(this.button6);

this.groupBox4.Controls.Add(this.total);

this.groupBox4.Controls.Add(this.label7);

this.groupBox4.Controls.Add(this.label6);

this.groupBox4.Controls.Add(this.vectorNumber);

this.groupBox4.Controls.Add(this.label5);

this.groupBox4.Controls.Add(this.listBox1);

this.groupBox4.Controls.Add(this.button4);

this.groupBox4.Controls.Add(this.pictureBox1);

this.groupBox4.Controls.Add(this.button3);

this.groupBox4.Enabled = false;

this.groupBox4.Location = new System.Drawing.Point(870, 6);

this.groupBox4.Name = "groupBox4";

this.groupBox4.Size = new System.Drawing.Size(200, 439);

this.groupBox4.TabIndex = 4;

this.groupBox4.TabStop = false;

this.groupBox4.Text = "Test";

//

// button6

//

this.button6.Location = new System.Drawing.Point(95, 212);

this.button6.Name = "button6";

this.button6.Size = new System.Drawing.Size(99, 23);

this.button6.TabIndex = 10;

this.button6.Text = "Preview";

this.button6.UseVisualStyleBackColor = true;

this.button6.Click += new System.EventHandler(this.button6\_Click);

//

// total

//

this.total.AutoSize = true;

this.total.Location = new System.Drawing.Point(42, 213);

this.total.Name = "total";

this.total.Size = new System.Drawing.Size(13, 13);

this.total.TabIndex = 8;

this.total.Text = "0";

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(7, 213);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(31, 13);

this.label7.TabIndex = 7;

this.label7.Text = "Total";

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(9, 244);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(14, 13);

this.label6.TabIndex = 6;

this.label6.Text = "#";

//

// vectorNumber

//

this.vectorNumber.Location = new System.Drawing.Point(29, 239);

this.vectorNumber.Name = "vectorNumber";

this.vectorNumber.Size = new System.Drawing.Size(48, 20);

this.vectorNumber.TabIndex = 5;

this.vectorNumber.Text = "1";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(7, 269);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(40, 13);

this.label5.TabIndex = 4;

this.label5.Text = "Result:";

//

// listBox1

//

this.listBox1.FormattingEnabled = true;

this.listBox1.Location = new System.Drawing.Point(7, 288);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(187, 108);

this.listBox1.TabIndex = 3;

//

// button4

//

this.button4.Enabled = false;

this.button4.Location = new System.Drawing.Point(95, 239);

this.button4.Name = "button4";

this.button4.Size = new System.Drawing.Size(99, 23);

this.button4.TabIndex = 2;

this.button4.Text = "Распознать";

this.button4.UseVisualStyleBackColor = true;

this.button4.Click += new System.EventHandler(this.button4\_Click);

//

// pictureBox1

//

this.pictureBox1.Location = new System.Drawing.Point(7, 50);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(187, 156);

this.pictureBox1.TabIndex = 1;

this.pictureBox1.TabStop = false;

//

// button3

//

this.button3.Location = new System.Drawing.Point(7, 20);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(187, 23);

this.button3.TabIndex = 0;

this.button3.Text = "Upload test set from file";

this.button3.UseVisualStyleBackColor = true;

this.button3.Click += new System.EventHandler(this.button3\_Click);

//

// chart1

//

this.chart1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

chartArea1.Name = "ChartArea1";

this.chart1.ChartAreas.Add(chartArea1);

legend1.Name = "Legend1";

this.chart1.Legends.Add(legend1);

this.chart1.Location = new System.Drawing.Point(12, 12);

this.chart1.Name = "chart1";

this.chart1.Palette = System.Windows.Forms.DataVisualization.Charting.ChartColorPalette.Pastel;

series1.ChartArea = "ChartArea1";

series1.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

series1.Legend = "Legend1";

series1.Name = "Normal";

series2.ChartArea = "ChartArea1";

series2.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

series2.Legend = "Legend1";

series2.Name = "Adaptive";

series3.ChartArea = "ChartArea1";

series3.ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

series3.Legend = "Legend1";

series3.Name = "Momentum";

this.chart1.Series.Add(series1);

this.chart1.Series.Add(series2);

this.chart1.Series.Add(series3);

this.chart1.Size = new System.Drawing.Size(603, 433);

this.chart1.TabIndex = 6;

this.chart1.Text = "chart1";

//

// button7

//

this.button7.Location = new System.Drawing.Point(7, 402);

this.button7.Name = "button7";

this.button7.Size = new System.Drawing.Size(187, 23);

this.button7.TabIndex = 7;

this.button7.Text = "Test all";

this.button7.UseVisualStyleBackColor = true;

this.button7.Click += new System.EventHandler(this.button7\_Click);

//

// MainForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1081, 478);

this.Controls.Add(this.chart1);

this.Controls.Add(this.groupBox4);

this.Controls.Add(this.groupBox3);

this.Controls.Add(this.groupBox2);

this.Controls.Add(this.groupBox1);

this.Name = "MainForm";

this.Text = "NN";

this.Load += new System.EventHandler(this.MainForm\_Load);

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.groupBox3.ResumeLayout(false);

this.groupBox3.PerformLayout();

this.groupBox4.ResumeLayout(false);

this.groupBox4.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.Label heightLabel;

private System.Windows.Forms.Label widthLabel;

private System.Windows.Forms.Button uploadButton;

private System.Windows.Forms.Label countLabel;

private System.Windows.Forms.Label classesLabel;

private System.Windows.Forms.Label pathLabel;

private System.Windows.Forms.Label sizeLabel;

private System.Windows.Forms.RichTextBox richTextBox1;

private System.Windows.Forms.GroupBox groupBox2;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.GroupBox groupBox3;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.GroupBox groupBox4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.ListBox listBox1;

private System.Windows.Forms.Button button4;

private System.Windows.Forms.PictureBox pictureBox1;

private System.Windows.Forms.Button button3;

private System.Windows.Forms.DataVisualization.Charting.Chart chart1;

private System.Windows.Forms.TextBox vectorNumber;

private System.Windows.Forms.Label total;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Button button5;

private System.Windows.Forms.Button button6;

private System.Windows.Forms.Button button7;

}

}

using nn.classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace nn

{

public partial class MainForm : Form

{

private TrainingSet \_trainingSet;

private TrainingSet \_testSet;

private NeuralNetwork \_networkNormal, \_networkAdaptive, \_networkMomentum;

private int \_indexToTest;

private static Random \_randomGenerator = new Random();

private int \_step = 1;

public MainForm()

{

InitializeComponent();

}

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

{

//var nw = new NeuralNetwork("normal.xml");

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog() { Filter = @"Text files (\*.txt)|\*.txt" })

{

if(openFileDialog.ShowDialog() == DialogResult.OK)

{

\_trainingSet = new TrainingSet(openFileDialog.FileName, \_step);

widthLabel.Text = string.Format("Width = {0}", \_trainingSet.SampleWidth);

heightLabel.Text = string.Format("Height = {0}", \_trainingSet.SampleHeight);

sizeLabel.Text = string.Format("Size = {0}", \_trainingSet.SampleSize);

classesLabel.Text = string.Format("Classes count = {0}", \_trainingSet.ClassesCount);

countLabel.Text = string.Format("Samples count = {0}", \_trainingSet.SamplesCount);

richTextBox1.Text = string.Format("Path: {0}", openFileDialog.FileName);

//DrawBitmaps();

textBox1.Text = String.Empty;

groupBox2.Enabled = true;

groupBox3.Enabled = false;

groupBox4.Enabled = false;

button5.Enabled = true;

button4.Enabled = vectorNumber.Enabled = false;

}

}

}

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

{

if(\_trainingSet != null)

{

string layers = textBox1.Text;

string[] buffer = layers.Split(new Char[] { ' ', ',', ';' }, StringSplitOptions.RemoveEmptyEntries);

int[] networkParams = new int[buffer.Length + 1];

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

{

int.TryParse(buffer[i], out networkParams[i]);

}

networkParams[buffer.Length] = \_trainingSet.ClassesCount;

//\_networkNormal = new NeuralNetwork("normal.xml");

//\_networkAdaptive = new NeuralNetwork("ad.xml");

//\_networkMomentum = new NeuralNetwork("mom.xml");

\_networkNormal = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Normal };

\_networkAdaptive = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Adaptive };

\_networkMomentum = new NeuralNetwork(\_trainingSet.SampleSize, networkParams) { Method = Method.Momentum };

//\_networkNormal.Save("normal.xml");

groupBox3.Enabled = true;

groupBox4.Enabled = false;

}

}

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

{

int iterationsCount;

double e0, delta;

int.TryParse(textBox2.Text, out iterationsCount);

double.TryParse(textBox3.Text, NumberStyles.Any, CultureInfo.InvariantCulture, out delta);

double.TryParse(textBox4.Text, NumberStyles.Any, CultureInfo.InvariantCulture, out e0);

chart1.Series[0].Points.Clear();

\_networkNormal.Delta

= \_networkAdaptive.Delta

= \_networkMomentum.Delta

= delta;

TaskFactory factory = new TaskFactory();

factory.StartNew(() =>

{

\_networkNormal.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[0],

1);

});

factory.StartNew(() =>

{

\_networkAdaptive.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[1],

1);

});

factory.StartNew(() =>

{

\_networkMomentum.Train(\_trainingSet.Samples,

\_trainingSet.Answers,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[2],

1);

});

groupBox4.Enabled = true;

//forSisII1.Enabled = button4.Enabled = true;

}

private void button3\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog() { Filter = @"Text files (\*.txt)|\*.txt" })

{

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

\_testSet = new TrainingSet(openFileDialog.FileName, \_step);

if (\_testSet.SampleHeight != \_trainingSet.SampleHeight

|| \_testSet.SampleWidth != \_trainingSet.SampleWidth)

{

MessageBox.Show("Test samples have wrong size", "Invalid test set", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (\_testSet.ClassesCount != \_trainingSet.ClassesCount)

{

MessageBox.Show("Classes count doesn't match", "Invalid test set", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

total.Text = \_testSet.SamplesCount.ToString();

vectorNumber.Enabled = true;

button4.Enabled = true;

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

\_indexToTest = int.Parse(vectorNumber.Text);// \_randomGenerator.Next(\_testSet.SamplesCount);

System.Drawing.Bitmap bmp = new System.Drawing.Bitmap(\_testSet.SampleWidth, \_testSet.SampleHeight);

for (int x = 0; x < bmp.Height; ++x)

{

for (int y = 0; y < bmp.Width; ++y)

{

//bmp.SetPixel(x, y, Color.White);

if (\_trainingSet.IsBinary)

{

int val = (int)\_testSet.Samples[\_indexToTest][x \* \_trainingSet.SampleWidth + y];

bmp.SetPixel(y, x, val > 0 ? Color.Black : Color.White);

}

else

{

int val = (int)\_testSet.Samples[\_indexToTest][x \* \_trainingSet.SampleWidth + y];

Color color = Color.FromArgb(val, val, val);

bmp.SetPixel(y, x, Color.FromArgb(val, val, val));

}

}

}

pictureBox1.Image = bmp;

button4.Enabled = true;

}

private void button4\_Click(object sender, EventArgs e)

{

button5\_Click(sender, e);

double[] netIn = \_testSet.Samples[\_indexToTest];

double[] result = new double[\_testSet.ClassesCount];

\_networkNormal.NetworkOut(netIn, out result);

listBox1.Items.Clear();

listBox1.Items.Add("n");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

listBox1.Items.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

\_networkAdaptive.NetworkOut(netIn, out result);

listBox1.Items.Add("a");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

listBox1.Items.Add("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

\_networkMomentum.NetworkOut(netIn, out result);

listBox1.Items.Add("m");

for (int i = 0; i < \_testSet.ClassesCount; i++)

{

listBox1.Items.Add(String.Format("{0}: {1}", i + 1, result[i]));

}

}

private void button7\_Click(object sender, EventArgs e)

{

int nCount, aCount, mCount;

nCount = aCount = mCount = 0;

for (int i = 0; i < \_testSet.SamplesCount; i++)

{

double[] nAns;

double[] aAns;

double[] mAns;

int nClass = \_networkNormal.NetworkOut(\_testSet.Samples[i], out nAns);

if(\_testSet.Answers[i][nClass] > 0.99)

{

nCount++;

}

int aClass = \_networkAdaptive.NetworkOut(\_testSet.Samples[i], out aAns);

if (\_testSet.Answers[i][aClass] > 0.99)

{

aCount++;

}

int mClass = \_networkMomentum.NetworkOut(\_testSet.Samples[i], out mAns);

if (\_testSet.Answers[i][mClass] > 0.99)

{

mCount++;

}

}

listBox1.Items.Clear();

listBox1.Items.Add(string.Format("Normal backprop: {0}/{1} ({2})", nCount, \_testSet.SamplesCount,

(100.0\*nCount)/\_testSet.SamplesCount));

listBox1.Items.Add(string.Format("Adaptive: {0}/{1} ({2})", aCount, \_testSet.SamplesCount,

(100.0 \* aCount) / \_testSet.SamplesCount));

listBox1.Items.Add(string.Format("Momentum:{0}/{1} ({2})", mCount, \_testSet.SamplesCount,

(100.0 \* mCount) / \_testSet.SamplesCount));

}

private void button5\_Click\_1(object sender, EventArgs e)

{

using (var PreviewForm = new PreviewForm(\_trainingSet))

{

PreviewForm.ShowDialog();

}

}

private void button6\_Click(object sender, EventArgs e)

{

using (var PreviewForm = new PreviewForm(\_testSet))

{

PreviewForm.ShowDialog();

}

}

}

}

namespace nn

{

partial class PreviewForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.flowLayoutPanel1 = new System.Windows.Forms.FlowLayoutPanel();

this.SuspendLayout();

//

// flowLayoutPanel1

//

this.flowLayoutPanel1.AutoScroll = true;

this.flowLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;

this.flowLayoutPanel1.Location = new System.Drawing.Point(0, 0);

this.flowLayoutPanel1.Name = "flowLayoutPanel1";

this.flowLayoutPanel1.Size = new System.Drawing.Size(284, 261);

this.flowLayoutPanel1.TabIndex = 1;

//

// PreviewForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(284, 261);

this.Controls.Add(this.flowLayoutPanel1);

this.Name = "PreviewForm";

this.Text = "PreviewForm";

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.FlowLayoutPanel flowLayoutPanel1;

}

}

using nn.classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace nn

{

public partial class PreviewForm : Form

{

public PreviewForm(TrainingSet trainingSet)

{

InitializeComponent();

this.Load += PreviewForm\_Load;

\_trainingSet = trainingSet;

}

private TrainingSet \_trainingSet;

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

{

flowLayoutPanel1.BeginInvoke((Action)(() => DrawBitmaps()));

}

private void DrawBitmaps()

{

flowLayoutPanel1.Controls.Clear();

for (int i = 0; i < \_trainingSet.SamplesCount; i++)

{

PictureBox box = new PictureBox()

{

Size = new Size()

{ Height = \_trainingSet.SampleHeight + 6, Width = \_trainingSet.SampleWidth + 6 }

};

System.Drawing.Bitmap bmp = new System.Drawing.Bitmap(\_trainingSet.SampleWidth, \_trainingSet.SampleHeight);

for (int x = 0; x < bmp.Height; ++x)

{

for (int y = 0; y < bmp.Width; ++y)

{

//bmp.SetPixel(x, y, Color.White);

if (\_trainingSet.IsBinary)

{

int val = (int)\_trainingSet.Samples[i][x \* \_trainingSet.SampleWidth + y];

bmp.SetPixel(y, x, val > 0 ? Color.Black : Color.White);

}

else

{

int val = (int)\_trainingSet.Samples[i][x \* \_trainingSet.SampleWidth + y];

Color color = Color.FromArgb(val, val, val);

bmp.SetPixel(y, x, Color.FromArgb(val, val, val));

}

}

}

flowLayoutPanel1.Controls.Add(box);

box.Image = bmp;

}

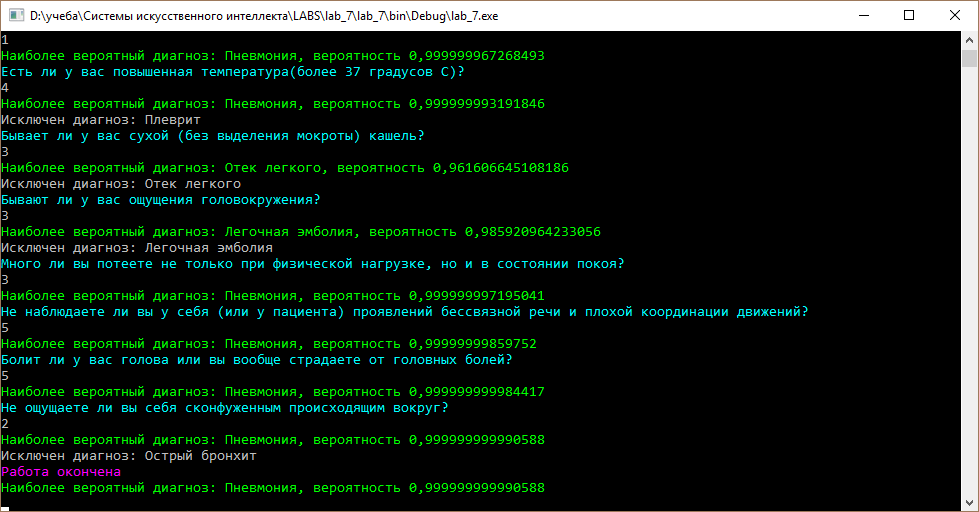
}

}

}

# 3. Экспертные системы. Правило Байеса.

**Результаты работы программы**



**Исходный код**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace lab\_7

{

class Diagnos : IComparable

{

Double P;

int n;

#region Члены IComparable

public int CompareTo(object obj)

{

Diagnos d = (Diagnos)obj;

return d.P.CompareTo(P);

}

#endregion

}

class Question :IComparable

{

//public String text { get; set; }

public Int32 number { get; set; }

public Int32 links { get; set; }

//public Boolean was { get; set; }

public Question(int n, int lin)

{

links = lin;

number = n;

}

public int CompareTo(object obj)

{

Question q = (Question)obj;

return q.links.CompareTo(links);

}

}

class Expert

{

private static Random randomGenerator = new Random();

private Int32 DiagnosisCount = 8;

private Int32 QuestionsCount = 68;

private Double[] P0;

private Double[] PC;

private Double[,] PYes;

private Double[,] PNo;

private Boolean[] flagDiagnosis;

private Boolean[] flagQuestionWas;

private Boolean[] flagQuestionValid;

private String[] Diagnosis;

private String[] Questions;

private Int32[] QuestionLinks;

private void Memory()

{

Diagnosis = new String[DiagnosisCount];

Questions = new String[QuestionsCount];

QuestionLinks = new Int32[QuestionsCount];

P0 = new Double[DiagnosisCount];

PC = new Double[DiagnosisCount];

PYes = new Double[DiagnosisCount, QuestionsCount];

PNo = new Double[DiagnosisCount, QuestionsCount];

flagDiagnosis = new Boolean[DiagnosisCount];

flagQuestionValid = new Boolean[QuestionsCount];

flagQuestionWas = new Boolean[QuestionsCount];

}

public void GetData(String diagnosis, String questions)

{

Memory();

using (FileStream fileStream = File.Open(questions, FileMode.Open))

{

using (StreamReader reader = new StreamReader(fileStream))

{

int i = 0;

String ques;

for (; ; )

{

ques = reader.ReadLine();

if (ques == null) break;

if (ques == "") continue;

Questions[i++] = ques;

}

}

}

using (FileStream fileStream = File.Open(diagnosis, FileMode.Open))

{

using (StreamReader reader = new StreamReader(fileStream))

{

int N;

String s;

String []buf1;

N = Int32.Parse(reader.ReadLine());

int i = 0;

for (i = 0; i < N; i++)

{

s = reader.ReadLine();

buf1 = s.Split(new Char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int j = 0;

Diagnosis[i] = "";

for (j = 0; j < buf1.Length - 1; j++)

{

Diagnosis[i] += buf1[j];

Diagnosis[i] += " ";

}

Diagnosis[i] = Diagnosis[i].TrimEnd();

P0[i] = PC[i] = Double.Parse(buf1[j]);//.Replace(',', '.'));

flagDiagnosis[i] = true;

}

while (true)

{

s = reader.ReadLine();

if (s == null) break;

if (s == "") continue;

buf1 = s.Replace(',', '.').Split(new Char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int bn = Int32.Parse(buf1[0]) - 1;

int qn = Int32.Parse(buf1[1]) - 1;

PYes[bn, qn] = Double.Parse(buf1[2].Replace('.', ','));

QuestionLinks[qn]++;

if (!flagQuestionValid[qn])

flagQuestionValid[qn] = true;

PNo[bn, qn] = Double.Parse(buf1[3].Replace('.', ','));

}

}

}

}

public void Run()

{

bool FlagEnd = true;

Int32 mvq,ans;

while (FlagEnd)

{

#region MOST VALUE QUESTION

//выбор вопроса

List<Question> q = new List<Question>();

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

if (!flagQuestionWas[i])

q.Add(new Question(i, QuestionLinks[i]));

q.RemoveAll(qu => qu.links == 0);

if (q.Count == 0) break;

q.Sort();

int n = q[0].links;

//оставить только значимые

q.RemoveAll(qw => qw.links < n);

n = randomGenerator.Next() % q.Count;

mvq = q[n].number;

//процедура выбора закончилась

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine(Questions[mvq]);

Console.ResetColor();

flagQuestionWas[mvq] = true;

#endregion

#region PC REFRESH

ans = Int32.Parse(Console.ReadLine());

int mi = -1;

Double maxp = Double.MinValue;

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

{

if (!flagDiagnosis[i]) continue;

Double py, pn, pPlus, pMinus;

py = PYes[i, mvq];

pn = PNo[i, mvq];

if (py == 0 && pn == 0) continue;

pPlus = py \* PC[i] / (py \* PC[i] + pn \* (1 - PC[i]));

pMinus = (1 - py) \* PC[i] / ((1 - py) \* PC[i] + (1 - pn) \* (1 - PC[i]));

if (ans == 5)

PC[i] = pPlus;

if (ans > 0 && ans < 5)

PC[i] = PC[i] + (pPlus - PC[i]) \* (double)ans / 5.0;

if (ans == -5)

PC[i] = pMinus;

if (ans < 0 && ans > -5)

PC[i] = PC[i] + (PC[i] - pMinus) \* (double)ans / 5.0;

if (PC[i] > maxp)

{

maxp = PC[i];

mi = i;

}

}

Console.ForegroundColor = ConsoleColor.Green;

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

{

if(flagDiagnosis[i] && PC[i] == maxp)

Console.WriteLine("Наиболее вероятный диагноз: {0}, вероятность {1}", Diagnosis[i], PC[i]);

}

Console.ResetColor();

#endregion

#region CHECK DIAGNOSIS

Double[] pMax = new Double[DiagnosisCount];

Double[] pMin = new Double[DiagnosisCount];

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

{

if (!flagDiagnosis[i]) continue;

pMax[i] = PC[i];

pMin[i] = PC[i];

for (int j = 0; j < QuestionsCount; j++)

{

if (flagQuestionWas[j]) continue;

if (PYes[i, j] == 0 && PNo[i, j] == 0) continue;

Double py = PYes[i, j], pn = PNo[i, j];

Double p1, p2, p3, p4;

p1 = py \* pMax[i] / (py \* pMax[i] + pn \* (1 - pMax[i]));

p2 = (1 - py) \* pMax[i] / ((1 - py) \* pMax[i] + (1 - pn) \* (1 - pMax[i]));

p3 = py \* pMin[i] / (py \* pMin[i] + pn \* (1 - pMin[i]));

p4 = (1 - py) \* pMin[i] / ((1 - py) \* pMin[i] + (1 - pn) \* (1 - pMin[i]));

pMax[i] = Math.Max(pMax[i],

Math.Max(p1,

Math.Max(p2,

Math.Max(p3, p4))));

pMin[i] = Math.Min(pMax[i],

Math.Min(p1,

Math.Min(p2,

Math.Min(p3, p4))));

}

}

Console.ForegroundColor = ConsoleColor.Gray;

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

{

if (!flagDiagnosis[i]) continue;

for (int j = 0; j < DiagnosisCount; j++)

{

if (!flagDiagnosis[j]) continue;

if (pMax[i] < pMin[j])

{

if (!flagDiagnosis[i]) continue;

Console.WriteLine("Исключен диагноз: {0}", Diagnosis[i]);

flagDiagnosis[i] = false;

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

{

if (PYes[i, k] == 0 && PNo[i, k] == 0) continue;

QuestionLinks[k]--;

}

}

}

}

Console.ResetColor();

var ndd = flagDiagnosis.Count(fd => fd);

if (ndd == 1) break;

#endregion

bool flag=false;

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

{

if (!flagQuestionWas[i]) continue;

if (QuestionLinks[i] > 0)

{

flag = true;

break;

}

}

if (!flag) break;

}

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Работа окончена");

int m = -1;

Double max = Double.MinValue;

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

{

if (!flagDiagnosis[i]) continue;

if (PC[i] > max)

{

max = PC[i];

m = i;

}

}

Console.ForegroundColor = ConsoleColor.Green;

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

{

if (flagDiagnosis[i] && PC[i] == max)

Console.WriteLine("Наиболее вероятный диагноз: {0}, вероятность {1}", Diagnosis[i], PC[i]);

}

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

{

if (flagDiagnosis[i] && PC[i] < max)

Console.WriteLine("Дополнительный вероятный диагноз: {0}, вероятность {1}", Diagnosis[i], PC[i]);

}

Console.ResetColor();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab\_7

{

class Program

{

static void Main(string[] args)

{

Expert e = new Expert();

e.GetData("var3uc.txt", "simptomsuc.txt");

e.Run();

Console.ReadKey();

}

}

}

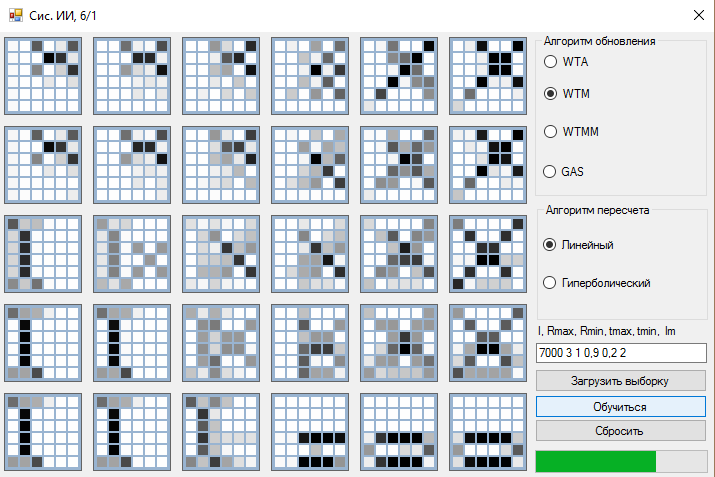
# 4. Карты Кохонена

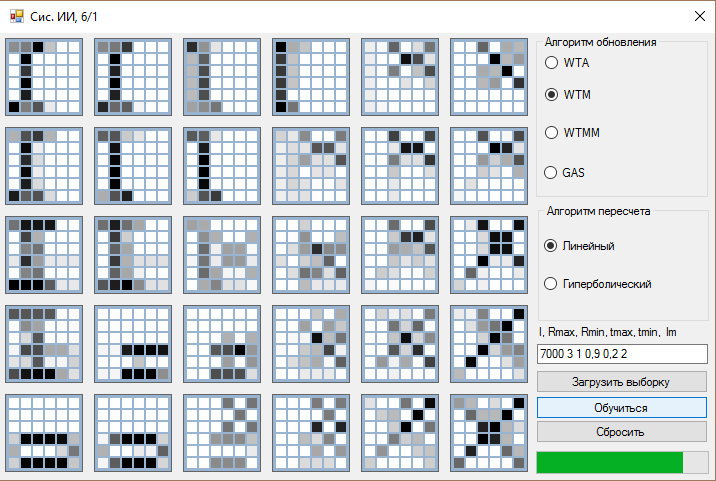
**Задание:**

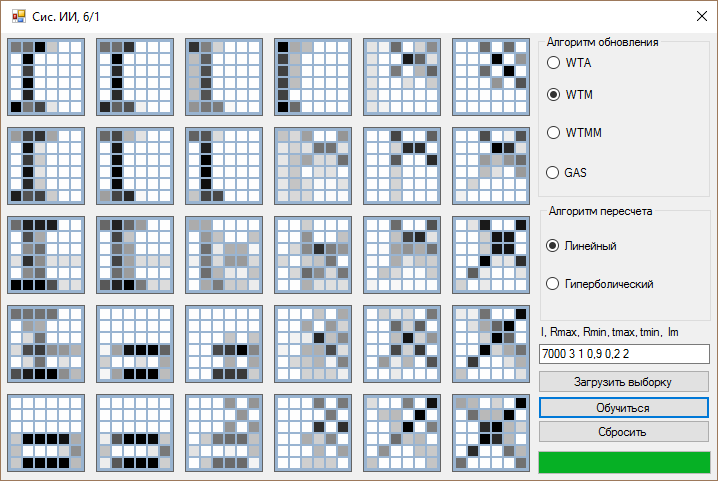
Построить карту Кохонена для символов F, L ,].

**Результаты работы программы (разные этапы обучения):**

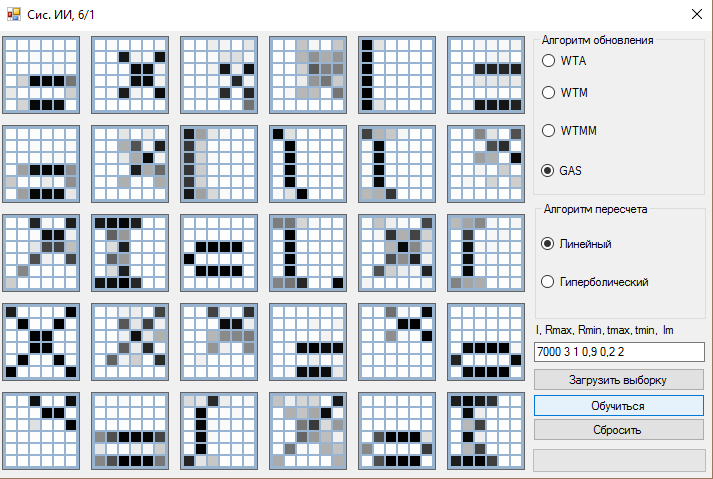
Метод WTM

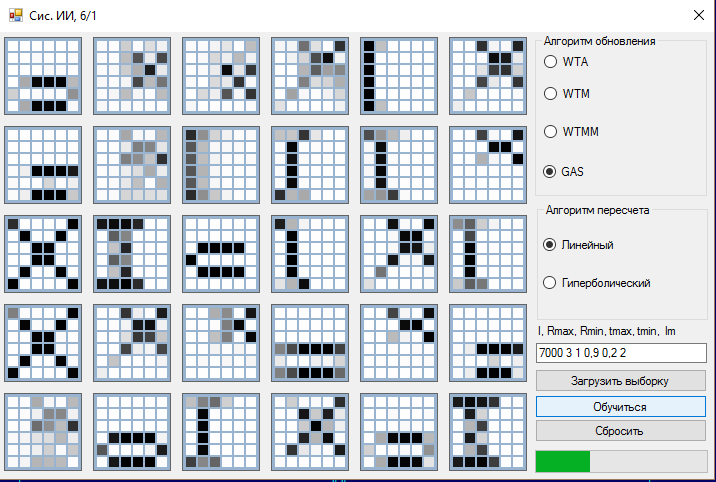


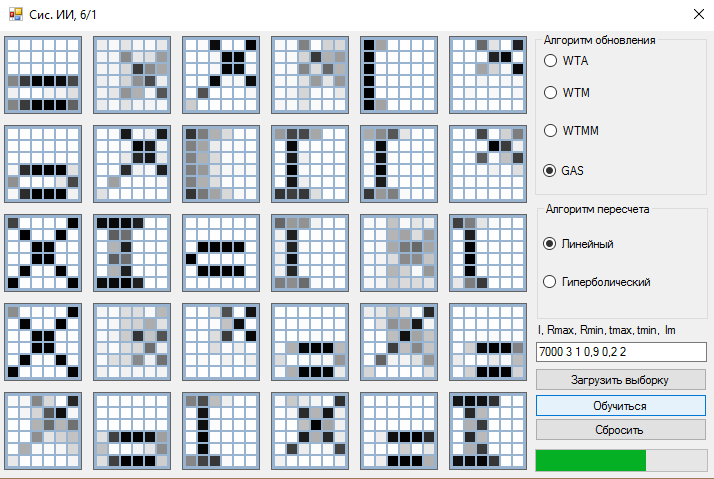




Нейронный газ







**Код программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace kohonen

{

class DPair : IComparable

{

public Int32 N { get; set; }

public Double D { get; set; }

public int CompareTo(Object obj)

{

DPair o = (DPair)obj;

return this.D.CompareTo(o.D);

}

}

class SOFM

{

private Double[][] sample;

private List<Double[]> neurons = new List<Double[]>();

private Double[,] TDistances;

private int mapSize,columns=6,lines=5,sampleCount;

private int updateMethod = 4, coeffMethod = 1;

private double Rmin, Rmax, R, tmin, tmax, t, lamb;

/// <summary>

/// Евклидова мера (расстояние между образами). Размерность должна совпадать

/// </summary>

/// <param name="x1">образ 1</param>

/// <param name="x2">образ 2</param>

/// <returns>Евклидово расстояние</returns>

Double EuqlidianMes(Double[] x1, Double[] x2)

{

Double result = 0;

for (int index = 0; index < x1.Length; index++)

{

result += Math.Pow(x1[index] - x2[index], 2);

}

return Math.Sqrt(result);

}

private void Update(Double[] x, Double[] dist, Double r, Double theta, Double lambda, int nearest)

{

if (updateMethod == 1)//wta

{

Double G = 1;

Double[] neuron = this.neurons[nearest];

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

neuron[i] += theta \* G \* (x[i] - neuron[i]);

return;

}

if (updateMethod == 2)//wtm

{

Double G = 1;

List<int> toChange = new List<int>();

for (int i1 = 0; i1 < mapSize; i1++)

if (TDistances[nearest, i1] < r)

toChange.Add(i1);

Double[] neuron;

foreach (int I in toChange)

{

neuron = this.neurons[I];

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

neuron[i] += theta \* G \* (x[i] - neuron[i]);

}

return;

}

if (updateMethod == 3)//wtmm

{

Double G = 1;

List<int> toChange = new List<int>();

for (int i1 = 0; i1 < mapSize; i1++)

if (TDistances[nearest, i1] < r)

toChange.Add(i1);

Double[] neuron;

Double d0 = dist[nearest],d,a1,a2=lambda\*lambda;

foreach (int I in toChange)

{

neuron = this.neurons[I];

d = dist[I];

a1 = -Math.Pow(d0 - d, 2);

G = Math.Exp(a1 / a2);

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

{

neuron[i] += theta \* G \* (x[i] - neuron[i]);

}

}

return;

}

if (updateMethod == 4)//gas

{

Double G = 1;

List<DPair> sdist = new List<DPair>();

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

{

DPair dp = new DPair();

dp.N = i;

dp.D = dist[i];

sdist.Add(dp);

}

sdist.Sort();

Double[] neuron;

for (int I = 0; I < sdist.Count; I++)

{

neuron = this.neurons[sdist[I].N];

G = Math.Exp(- I/ lambda);

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

{

neuron[i] += theta \* G \* (x[i] - neuron[i]);

}

}

}

}

public void SetParsams(Double Rmax, Double Rmin, Double tmax, Double tmin, Double lamb)

{

this.Rmin = Rmin;

this.Rmax = Rmax;

//this.R = R;

this.tmin = tmin;

this.tmax = tmax;

// this.t = t;

this.lamb = lamb;

}

/// <summary>

///

/// </summary>

public void CalcTDistances()

{

mapSize = neurons.Count;

TDistances = new Double[mapSize, mapSize];

int x1, y1, x2, y2;

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

{

x1 = i / columns;

y1 = i % columns;

for (int j = 0; j < mapSize; j++)

{

if (i == j)

{

TDistances[i, j] = .0;

continue;

}

x2 = j / columns;

y2 = j % columns;

double dx = (x1 - x2);

double dy = (y1 - y2);

TDistances[i, j] = Math.Sqrt(dx \* dx + dy \* dy);

}

}

}

/// <summary>

///

/// </summary>

/// <param name="Epochs"></param>

public void Run(Int32 Epochs, Double R0, List<ForSisII> updateList, int uStep, Form1 owner)

{

Double dR = (Rmax - Rmin)/Epochs;

Double dT = (tmax - tmin)/Epochs;

Double oR = Rmin / Rmax;

Double oT = tmin / tmax;

Double[] currentDistances = new Double[mapSize];

//по итерациям

for (int epoch = 0; epoch < Epochs; epoch++)

{

//по выборке

if (coeffMethod == 2)

{

t = tmax \* Math.Pow(oT, (double)epoch / Epochs);

R = Rmax \* Math.Pow(oR, (double)epoch / Epochs);

}

else

{

t = tmax - dT \* epoch;

R = Rmax - dR \* epoch;

}

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

{

Double minDist = Double.MaxValue;

int minJ = -1;

//ищем ближайший нейрон для элемента выборки

for (int j = 0; j < mapSize; j++)

{

currentDistances[j] = EuqlidianMes(sample[i], neurons[j]);

if (currentDistances[j] < minDist)

{

minDist = currentDistances[j];

minJ = j;

}

}

this.Update(sample[i], currentDistances, R, t, 2, minJ);

}

if (epoch % uStep == 0)

{

int n = 0;

foreach (ForSisII fs in updateList)

{

fs.Values = neurons[n++];

fs.Invalidate(fs.ClientRectangle);

fs.Update();

}

owner.Tag = epoch;

owner.Refresh();

}

}

}

public virtual void LoadSample(String Path)

{

using (FileStream fileStream = new FileStream(Path, FileMode.Open))

{

using (StreamReader streamReader = new StreamReader(fileStream))

{

///\*

String content = streamReader.ReadToEnd();

content = content.Replace('\n', ' ');

content = content.Replace('\r', ' ');

String[] buffer = content.Split(new Char[] { '#' }, StringSplitOptions.RemoveEmptyEntries);

Int32.TryParse(buffer[0], out sampleCount);

sample = new Double[sampleCount][];

//classes = new Int32[sampleCount];

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

{

sample[i] = new Double[36];

String current = buffer[i + 1];

String[] bitBuffer = current.Split(new Char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int j;

for (j = 0; j < 36; j++)

{

Double.TryParse(bitBuffer[j], out sample[i][j]);

}

//int ind;

//Int32.TryParse(bitBuffer[j], out ind);

//classes[i] = ind;

}

//\*/

}

}

}

public Double[][] Sample { set { sample = value; } }

public List<Double[]> Newrons { get { return neurons; } set { neurons = value; } }

public Int32 UpdateMethod { get { return updateMethod; } set { updateMethod = value; } }

public Int32 CoeffMethod { get { return coeffMethod; } set { coeffMethod = value; } }

}

}

namespace kohonen

{

partial class ForSisII

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором компонентов

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.panel1 = new System.Windows.Forms.Panel();

this.panel37 = new System.Windows.Forms.Panel();

this.panel31 = new System.Windows.Forms.Panel();

this.panel25 = new System.Windows.Forms.Panel();

this.panel19 = new System.Windows.Forms.Panel();

this.panel13 = new System.Windows.Forms.Panel();

this.panel7 = new System.Windows.Forms.Panel();

this.panel36 = new System.Windows.Forms.Panel();

this.panel30 = new System.Windows.Forms.Panel();

this.panel24 = new System.Windows.Forms.Panel();

this.panel18 = new System.Windows.Forms.Panel();

this.panel12 = new System.Windows.Forms.Panel();

this.panel6 = new System.Windows.Forms.Panel();

this.panel35 = new System.Windows.Forms.Panel();

this.panel29 = new System.Windows.Forms.Panel();

this.panel23 = new System.Windows.Forms.Panel();

this.panel17 = new System.Windows.Forms.Panel();

this.panel11 = new System.Windows.Forms.Panel();

this.panel5 = new System.Windows.Forms.Panel();

this.panel34 = new System.Windows.Forms.Panel();

this.panel28 = new System.Windows.Forms.Panel();

this.panel22 = new System.Windows.Forms.Panel();

this.panel16 = new System.Windows.Forms.Panel();

this.panel10 = new System.Windows.Forms.Panel();

this.panel4 = new System.Windows.Forms.Panel();

this.panel33 = new System.Windows.Forms.Panel();

this.panel27 = new System.Windows.Forms.Panel();

this.panel21 = new System.Windows.Forms.Panel();

this.panel15 = new System.Windows.Forms.Panel();

this.panel9 = new System.Windows.Forms.Panel();

this.panel3 = new System.Windows.Forms.Panel();

this.panel32 = new System.Windows.Forms.Panel();

this.panel26 = new System.Windows.Forms.Panel();

this.panel20 = new System.Windows.Forms.Panel();

this.panel14 = new System.Windows.Forms.Panel();

this.panel8 = new System.Windows.Forms.Panel();

this.panel2 = new System.Windows.Forms.Panel();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// panel1

//

this.panel1.BackColor = System.Drawing.SystemColors.ActiveCaption;

this.panel1.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Stretch;

this.panel1.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;

this.panel1.Controls.Add(this.panel37);

this.panel1.Controls.Add(this.panel31);

this.panel1.Controls.Add(this.panel25);

this.panel1.Controls.Add(this.panel19);

this.panel1.Controls.Add(this.panel13);

this.panel1.Controls.Add(this.panel7);

this.panel1.Controls.Add(this.panel36);

this.panel1.Controls.Add(this.panel30);

this.panel1.Controls.Add(this.panel24);

this.panel1.Controls.Add(this.panel18);

this.panel1.Controls.Add(this.panel12);

this.panel1.Controls.Add(this.panel6);

this.panel1.Controls.Add(this.panel35);

this.panel1.Controls.Add(this.panel29);

this.panel1.Controls.Add(this.panel23);

this.panel1.Controls.Add(this.panel17);

this.panel1.Controls.Add(this.panel11);

this.panel1.Controls.Add(this.panel5);

this.panel1.Controls.Add(this.panel34);

this.panel1.Controls.Add(this.panel28);

this.panel1.Controls.Add(this.panel22);

this.panel1.Controls.Add(this.panel16);

this.panel1.Controls.Add(this.panel10);

this.panel1.Controls.Add(this.panel4);

this.panel1.Controls.Add(this.panel33);

this.panel1.Controls.Add(this.panel27);

this.panel1.Controls.Add(this.panel21);

this.panel1.Controls.Add(this.panel15);

this.panel1.Controls.Add(this.panel9);

this.panel1.Controls.Add(this.panel3);

this.panel1.Controls.Add(this.panel32);

this.panel1.Controls.Add(this.panel26);

this.panel1.Controls.Add(this.panel20);

this.panel1.Controls.Add(this.panel14);

this.panel1.Controls.Add(this.panel8);

this.panel1.Controls.Add(this.panel2);

this.panel1.Location = new System.Drawing.Point(3, 3);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(78, 78);

this.panel1.TabIndex = 0;

//

// panel37

//

this.panel37.BackColor = System.Drawing.Color.White;

this.panel37.Location = new System.Drawing.Point(63, 63);

this.panel37.Margin = new System.Windows.Forms.Padding(1);

this.panel37.Name = "panel37";

this.panel37.Size = new System.Drawing.Size(10, 10);

this.panel37.TabIndex = 0;

this.panel37.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel31

//

this.panel31.BackColor = System.Drawing.Color.White;

this.panel31.Location = new System.Drawing.Point(63, 51);

this.panel31.Margin = new System.Windows.Forms.Padding(1);

this.panel31.Name = "panel31";

this.panel31.Size = new System.Drawing.Size(10, 10);

this.panel31.TabIndex = 0;

this.panel31.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel25

//

this.panel25.BackColor = System.Drawing.Color.White;

this.panel25.Location = new System.Drawing.Point(63, 39);

this.panel25.Margin = new System.Windows.Forms.Padding(1);

this.panel25.Name = "panel25";

this.panel25.Size = new System.Drawing.Size(10, 10);

this.panel25.TabIndex = 0;

this.panel25.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel19

//

this.panel19.BackColor = System.Drawing.Color.White;

this.panel19.Location = new System.Drawing.Point(63, 27);

this.panel19.Margin = new System.Windows.Forms.Padding(1);

this.panel19.Name = "panel19";

this.panel19.Size = new System.Drawing.Size(10, 10);

this.panel19.TabIndex = 0;

this.panel19.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel13

//

this.panel13.BackColor = System.Drawing.Color.White;

this.panel13.Location = new System.Drawing.Point(63, 15);

this.panel13.Margin = new System.Windows.Forms.Padding(1);

this.panel13.Name = "panel13";

this.panel13.Size = new System.Drawing.Size(10, 10);

this.panel13.TabIndex = 0;

this.panel13.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel7

//

this.panel7.BackColor = System.Drawing.Color.White;

this.panel7.Location = new System.Drawing.Point(63, 3);

this.panel7.Margin = new System.Windows.Forms.Padding(1);

this.panel7.Name = "panel7";

this.panel7.Size = new System.Drawing.Size(10, 10);

this.panel7.TabIndex = 0;

this.panel7.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel36

//

this.panel36.BackColor = System.Drawing.Color.White;

this.panel36.Location = new System.Drawing.Point(51, 63);

this.panel36.Margin = new System.Windows.Forms.Padding(1);

this.panel36.Name = "panel36";

this.panel36.Size = new System.Drawing.Size(10, 10);

this.panel36.TabIndex = 0;

this.panel36.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel30

//

this.panel30.BackColor = System.Drawing.Color.White;

this.panel30.Location = new System.Drawing.Point(51, 51);

this.panel30.Margin = new System.Windows.Forms.Padding(1);

this.panel30.Name = "panel30";

this.panel30.Size = new System.Drawing.Size(10, 10);

this.panel30.TabIndex = 0;

this.panel30.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel24

//

this.panel24.BackColor = System.Drawing.Color.White;

this.panel24.Location = new System.Drawing.Point(51, 39);

this.panel24.Margin = new System.Windows.Forms.Padding(1);

this.panel24.Name = "panel24";

this.panel24.Size = new System.Drawing.Size(10, 10);

this.panel24.TabIndex = 0;

this.panel24.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel18

//

this.panel18.BackColor = System.Drawing.Color.White;

this.panel18.Location = new System.Drawing.Point(51, 27);

this.panel18.Margin = new System.Windows.Forms.Padding(1);

this.panel18.Name = "panel18";

this.panel18.Size = new System.Drawing.Size(10, 10);

this.panel18.TabIndex = 0;

this.panel18.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel12

//

this.panel12.BackColor = System.Drawing.Color.White;

this.panel12.Location = new System.Drawing.Point(51, 15);

this.panel12.Margin = new System.Windows.Forms.Padding(1);

this.panel12.Name = "panel12";

this.panel12.Size = new System.Drawing.Size(10, 10);

this.panel12.TabIndex = 0;

this.panel12.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel6

//

this.panel6.BackColor = System.Drawing.Color.White;

this.panel6.Location = new System.Drawing.Point(51, 3);

this.panel6.Margin = new System.Windows.Forms.Padding(1);

this.panel6.Name = "panel6";

this.panel6.Size = new System.Drawing.Size(10, 10);

this.panel6.TabIndex = 0;

this.panel6.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel35

//

this.panel35.BackColor = System.Drawing.Color.White;

this.panel35.Location = new System.Drawing.Point(39, 63);

this.panel35.Margin = new System.Windows.Forms.Padding(1);

this.panel35.Name = "panel35";

this.panel35.Size = new System.Drawing.Size(10, 10);

this.panel35.TabIndex = 0;

this.panel35.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel29

//

this.panel29.BackColor = System.Drawing.Color.White;

this.panel29.Location = new System.Drawing.Point(39, 51);

this.panel29.Margin = new System.Windows.Forms.Padding(1);

this.panel29.Name = "panel29";

this.panel29.Size = new System.Drawing.Size(10, 10);

this.panel29.TabIndex = 0;

this.panel29.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel23

//

this.panel23.BackColor = System.Drawing.Color.White;

this.panel23.Location = new System.Drawing.Point(39, 39);

this.panel23.Margin = new System.Windows.Forms.Padding(1);

this.panel23.Name = "panel23";

this.panel23.Size = new System.Drawing.Size(10, 10);

this.panel23.TabIndex = 0;

this.panel23.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel17

//

this.panel17.BackColor = System.Drawing.Color.White;

this.panel17.Location = new System.Drawing.Point(39, 27);

this.panel17.Margin = new System.Windows.Forms.Padding(1);

this.panel17.Name = "panel17";

this.panel17.Size = new System.Drawing.Size(10, 10);

this.panel17.TabIndex = 0;

this.panel17.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel11

//

this.panel11.BackColor = System.Drawing.Color.White;

this.panel11.Location = new System.Drawing.Point(39, 15);

this.panel11.Margin = new System.Windows.Forms.Padding(1);

this.panel11.Name = "panel11";

this.panel11.Size = new System.Drawing.Size(10, 10);

this.panel11.TabIndex = 0;

this.panel11.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel5

//

this.panel5.BackColor = System.Drawing.Color.White;

this.panel5.Location = new System.Drawing.Point(39, 3);

this.panel5.Margin = new System.Windows.Forms.Padding(1);

this.panel5.Name = "panel5";

this.panel5.Size = new System.Drawing.Size(10, 10);

this.panel5.TabIndex = 0;

this.panel5.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel34

//

this.panel34.BackColor = System.Drawing.Color.White;

this.panel34.Location = new System.Drawing.Point(27, 63);

this.panel34.Margin = new System.Windows.Forms.Padding(1);

this.panel34.Name = "panel34";

this.panel34.Size = new System.Drawing.Size(10, 10);

this.panel34.TabIndex = 0;

this.panel34.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel28

//

this.panel28.BackColor = System.Drawing.Color.White;

this.panel28.Location = new System.Drawing.Point(27, 51);

this.panel28.Margin = new System.Windows.Forms.Padding(1);

this.panel28.Name = "panel28";

this.panel28.Size = new System.Drawing.Size(10, 10);

this.panel28.TabIndex = 0;

this.panel28.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel22

//

this.panel22.BackColor = System.Drawing.Color.White;

this.panel22.Location = new System.Drawing.Point(27, 39);

this.panel22.Margin = new System.Windows.Forms.Padding(1);

this.panel22.Name = "panel22";

this.panel22.Size = new System.Drawing.Size(10, 10);

this.panel22.TabIndex = 0;

this.panel22.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel16

//

this.panel16.BackColor = System.Drawing.Color.White;

this.panel16.Location = new System.Drawing.Point(27, 27);

this.panel16.Margin = new System.Windows.Forms.Padding(1);

this.panel16.Name = "panel16";

this.panel16.Size = new System.Drawing.Size(10, 10);

this.panel16.TabIndex = 0;

this.panel16.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel10

//

this.panel10.BackColor = System.Drawing.Color.White;

this.panel10.Location = new System.Drawing.Point(27, 15);

this.panel10.Margin = new System.Windows.Forms.Padding(1);

this.panel10.Name = "panel10";

this.panel10.Size = new System.Drawing.Size(10, 10);

this.panel10.TabIndex = 0;

this.panel10.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel4

//

this.panel4.BackColor = System.Drawing.Color.White;

this.panel4.Location = new System.Drawing.Point(27, 3);

this.panel4.Margin = new System.Windows.Forms.Padding(1);

this.panel4.Name = "panel4";

this.panel4.Size = new System.Drawing.Size(10, 10);

this.panel4.TabIndex = 0;

this.panel4.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel33

//

this.panel33.BackColor = System.Drawing.Color.White;

this.panel33.Location = new System.Drawing.Point(15, 63);

this.panel33.Margin = new System.Windows.Forms.Padding(1);

this.panel33.Name = "panel33";

this.panel33.Size = new System.Drawing.Size(10, 10);

this.panel33.TabIndex = 0;

this.panel33.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel27

//

this.panel27.BackColor = System.Drawing.Color.White;

this.panel27.Location = new System.Drawing.Point(15, 51);

this.panel27.Margin = new System.Windows.Forms.Padding(1);

this.panel27.Name = "panel27";

this.panel27.Size = new System.Drawing.Size(10, 10);

this.panel27.TabIndex = 0;

this.panel27.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel21

//

this.panel21.BackColor = System.Drawing.Color.White;

this.panel21.Location = new System.Drawing.Point(15, 39);

this.panel21.Margin = new System.Windows.Forms.Padding(1);

this.panel21.Name = "panel21";

this.panel21.Size = new System.Drawing.Size(10, 10);

this.panel21.TabIndex = 0;

this.panel21.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel15

//

this.panel15.BackColor = System.Drawing.Color.White;

this.panel15.Location = new System.Drawing.Point(15, 27);

this.panel15.Margin = new System.Windows.Forms.Padding(1);

this.panel15.Name = "panel15";

this.panel15.Size = new System.Drawing.Size(10, 10);

this.panel15.TabIndex = 0;

this.panel15.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel9

//

this.panel9.BackColor = System.Drawing.Color.White;

this.panel9.Location = new System.Drawing.Point(15, 15);

this.panel9.Margin = new System.Windows.Forms.Padding(1);

this.panel9.Name = "panel9";

this.panel9.Size = new System.Drawing.Size(10, 10);

this.panel9.TabIndex = 0;

this.panel9.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel3

//

this.panel3.BackColor = System.Drawing.Color.White;

this.panel3.Location = new System.Drawing.Point(15, 3);

this.panel3.Margin = new System.Windows.Forms.Padding(1);

this.panel3.Name = "panel3";

this.panel3.Size = new System.Drawing.Size(10, 10);

this.panel3.TabIndex = 0;

this.panel3.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel32

//

this.panel32.BackColor = System.Drawing.Color.White;

this.panel32.Location = new System.Drawing.Point(3, 63);

this.panel32.Margin = new System.Windows.Forms.Padding(1);

this.panel32.Name = "panel32";

this.panel32.Size = new System.Drawing.Size(10, 10);

this.panel32.TabIndex = 0;

this.panel32.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel26

//

this.panel26.BackColor = System.Drawing.Color.White;

this.panel26.Location = new System.Drawing.Point(3, 51);

this.panel26.Margin = new System.Windows.Forms.Padding(1);

this.panel26.Name = "panel26";

this.panel26.Size = new System.Drawing.Size(10, 10);

this.panel26.TabIndex = 0;

this.panel26.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel20

//

this.panel20.BackColor = System.Drawing.Color.White;

this.panel20.Location = new System.Drawing.Point(3, 39);

this.panel20.Margin = new System.Windows.Forms.Padding(1);

this.panel20.Name = "panel20";

this.panel20.Size = new System.Drawing.Size(10, 10);

this.panel20.TabIndex = 0;

this.panel20.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel14

//

this.panel14.BackColor = System.Drawing.Color.White;

this.panel14.Location = new System.Drawing.Point(3, 27);

this.panel14.Margin = new System.Windows.Forms.Padding(1);

this.panel14.Name = "panel14";

this.panel14.Size = new System.Drawing.Size(10, 10);

this.panel14.TabIndex = 0;

this.panel14.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel8

//

this.panel8.BackColor = System.Drawing.Color.White;

this.panel8.Location = new System.Drawing.Point(3, 15);

this.panel8.Margin = new System.Windows.Forms.Padding(1);

this.panel8.Name = "panel8";

this.panel8.Size = new System.Drawing.Size(10, 10);

this.panel8.TabIndex = 0;

this.panel8.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// panel2

//

this.panel2.BackColor = System.Drawing.Color.White;

this.panel2.Location = new System.Drawing.Point(3, 3);

this.panel2.Margin = new System.Windows.Forms.Padding(1);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(10, 10);

this.panel2.TabIndex = 0;

this.panel2.Paint += new System.Windows.Forms.PaintEventHandler(this.panel37\_Paint);

//

// ForSisII

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.Controls.Add(this.panel1);

this.Name = "ForSisII";

this.Size = new System.Drawing.Size(83, 83);

this.panel1.ResumeLayout(false);

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel37;

private System.Windows.Forms.Panel panel31;

private System.Windows.Forms.Panel panel25;

private System.Windows.Forms.Panel panel19;

private System.Windows.Forms.Panel panel13;

private System.Windows.Forms.Panel panel7;

private System.Windows.Forms.Panel panel36;

private System.Windows.Forms.Panel panel30;

private System.Windows.Forms.Panel panel24;

private System.Windows.Forms.Panel panel18;

private System.Windows.Forms.Panel panel12;

private System.Windows.Forms.Panel panel6;

private System.Windows.Forms.Panel panel35;

private System.Windows.Forms.Panel panel29;

private System.Windows.Forms.Panel panel23;

private System.Windows.Forms.Panel panel17;

private System.Windows.Forms.Panel panel11;

private System.Windows.Forms.Panel panel5;

private System.Windows.Forms.Panel panel34;

private System.Windows.Forms.Panel panel28;

private System.Windows.Forms.Panel panel22;

private System.Windows.Forms.Panel panel16;

private System.Windows.Forms.Panel panel10;

private System.Windows.Forms.Panel panel4;

private System.Windows.Forms.Panel panel33;

private System.Windows.Forms.Panel panel27;

private System.Windows.Forms.Panel panel21;

private System.Windows.Forms.Panel panel15;

private System.Windows.Forms.Panel panel9;

private System.Windows.Forms.Panel panel3;

private System.Windows.Forms.Panel panel32;

private System.Windows.Forms.Panel panel26;

private System.Windows.Forms.Panel panel20;

private System.Windows.Forms.Panel panel14;

private System.Windows.Forms.Panel panel8;

private System.Windows.Forms.Panel panel2;

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace kohonen

{

public partial class ForSisII : UserControl

{

private static Random randomGenerator=new Random();

public ForSisII()

{

InitializeComponent();

values = new Double[36];

}

public void Generate()

{

for (int i = 0; i < 36; i++)

values[i] = randomGenerator.NextDouble();

this.Ref();

}

private Double[] values;

/// <summary>

/// инвертирует цвет квадратика по клику

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void InvertColor(object sender, EventArgs e)

{

/\*

Panel panel = ((Panel)sender);

if (panel.BackColor == Color.White)

panel.BackColor = Color.Black;

else

panel.BackColor = Color.White;

String numString = (panel.Name.Substring(5));

int index = Int32.Parse(numString) - 2;

values[index] = panel.BackColor == Color.Black ? 0 : 1;

\*/

}

/// <summary>

/// получает массив со значениями, белый закодирован единицей

/// </summary>

/// <returns></returns>

public Double[] Values

{

get { return values; }

set { values = value; }

}

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

{

foreach (object p in panel1.Controls)

{

//if(p!=button1)

((Panel)p).BackColor = Color.White;

}

for (int i = 0; i < 36; i++)

{

values[i] = 1;

}

}

public void Ref()

{

foreach (object p in panel1.Controls)

{

//if (p != button1)

{

Panel panel = (Panel)p;

String numString = panel.Name.Substring(5);

Int32 index = Int32.Parse(numString) - 2;

Int32 Col = (Int32)(values[index] \* 255);

panel.BackColor = Color.FromArgb(Col, Col, Col);

}

}

}

private void panel37\_Paint(object sender, PaintEventArgs e)

{

foreach (object p in panel1.Controls)

{

//if (p != button1)

{

Panel panel = (Panel)p;

String numString = panel.Name.Substring(5);

Int32 index = Int32.Parse(numString) - 2;

Int32 Col = (Int32)(values[index] \* 255);

panel.BackColor = Color.FromArgb(Col, Col, Col);

}

}

}

}

}

namespace kohonen

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.button1 = new System.Windows.Forms.Button();

this.button2 = new System.Windows.Forms.Button();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.radioButton4 = new System.Windows.Forms.RadioButton();

this.radioButton3 = new System.Windows.Forms.RadioButton();

this.radioButton2 = new System.Windows.Forms.RadioButton();

this.radioButton1 = new System.Windows.Forms.RadioButton();

this.groupBox2 = new System.Windows.Forms.GroupBox();

this.radioButton6 = new System.Windows.Forms.RadioButton();

this.radioButton5 = new System.Windows.Forms.RadioButton();

this.textBox1 = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.progressBar1 = new System.Windows.Forms.ProgressBar();

this.forSisII30 = new kohonen.ForSisII();

this.forSisII29 = new kohonen.ForSisII();

this.forSisII28 = new kohonen.ForSisII();

this.forSisII27 = new kohonen.ForSisII();

this.forSisII26 = new kohonen.ForSisII();

this.forSisII25 = new kohonen.ForSisII();

this.forSisII21 = new kohonen.ForSisII();

this.forSisII14 = new kohonen.ForSisII();

this.forSisII7 = new kohonen.ForSisII();

this.forSisII20 = new kohonen.ForSisII();

this.forSisII13 = new kohonen.ForSisII();

this.forSisII6 = new kohonen.ForSisII();

this.forSisII19 = new kohonen.ForSisII();

this.forSisII12 = new kohonen.ForSisII();

this.forSisII5 = new kohonen.ForSisII();

this.forSisII18 = new kohonen.ForSisII();

this.forSisII11 = new kohonen.ForSisII();

this.forSisII4 = new kohonen.ForSisII();

this.forSisII24 = new kohonen.ForSisII();

this.forSisII17 = new kohonen.ForSisII();

this.forSisII10 = new kohonen.ForSisII();

this.forSisII3 = new kohonen.ForSisII();

this.forSisII23 = new kohonen.ForSisII();

this.forSisII16 = new kohonen.ForSisII();

this.forSisII9 = new kohonen.ForSisII();

this.forSisII2 = new kohonen.ForSisII();

this.forSisII22 = new kohonen.ForSisII();

this.forSisII15 = new kohonen.ForSisII();

this.forSisII8 = new kohonen.ForSisII();

this.forSisII1 = new kohonen.ForSisII();

this.button3 = new System.Windows.Forms.Button();

this.groupBox1.SuspendLayout();

this.groupBox2.SuspendLayout();

this.SuspendLayout();

//

// button1

//

this.button1.Location = new System.Drawing.Point(537, 363);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(172, 23);

this.button1.TabIndex = 2;

this.button1.Text = "Обучиться";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// button2

//

this.button2.Location = new System.Drawing.Point(537, 337);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(172, 23);

this.button2.TabIndex = 3;

this.button2.Text = "Загрузить выборку";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// groupBox1

//

this.groupBox1.Controls.Add(this.radioButton4);

this.groupBox1.Controls.Add(this.radioButton3);

this.groupBox1.Controls.Add(this.radioButton2);

this.groupBox1.Controls.Add(this.radioButton1);

this.groupBox1.Location = new System.Drawing.Point(537, 2);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(172, 163);

this.groupBox1.TabIndex = 4;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Алгоритм обновления";

this.groupBox1.Enter += new System.EventHandler(this.groupBox1\_Enter);

//

// radioButton4

//

this.radioButton4.AutoSize = true;

this.radioButton4.Location = new System.Drawing.Point(8, 129);

this.radioButton4.Name = "radioButton4";

this.radioButton4.Size = new System.Drawing.Size(47, 17);

this.radioButton4.TabIndex = 0;

this.radioButton4.Text = "GAS";

this.radioButton4.UseVisualStyleBackColor = true;

//

// radioButton3

//

this.radioButton3.AutoSize = true;

this.radioButton3.Location = new System.Drawing.Point(9, 89);

this.radioButton3.Name = "radioButton3";

this.radioButton3.Size = new System.Drawing.Size(61, 17);

this.radioButton3.TabIndex = 0;

this.radioButton3.Text = "WTMM";

this.radioButton3.UseVisualStyleBackColor = true;

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Checked = true;

this.radioButton2.Location = new System.Drawing.Point(9, 51);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(52, 17);

this.radioButton2.TabIndex = 0;

this.radioButton2.TabStop = true;

this.radioButton2.Text = "WTM";

this.radioButton2.UseVisualStyleBackColor = true;

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Location = new System.Drawing.Point(9, 19);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(50, 17);

this.radioButton1.TabIndex = 0;

this.radioButton1.Text = "WTA";

this.radioButton1.UseVisualStyleBackColor = true;

//

// groupBox2

//

this.groupBox2.Controls.Add(this.radioButton6);

this.groupBox2.Controls.Add(this.radioButton5);

this.groupBox2.Location = new System.Drawing.Point(539, 171);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(171, 118);

this.groupBox2.TabIndex = 5;

this.groupBox2.TabStop = false;

this.groupBox2.Text = "Алгоритм пересчета";

//

// radioButton6

//

this.radioButton6.AutoSize = true;

this.radioButton6.Location = new System.Drawing.Point(6, 71);

this.radioButton6.Name = "radioButton6";

this.radioButton6.Size = new System.Drawing.Size(114, 17);

this.radioButton6.TabIndex = 0;

this.radioButton6.Text = "Гиперболический";

this.radioButton6.UseVisualStyleBackColor = true;

//

// radioButton5

//

this.radioButton5.AutoSize = true;

this.radioButton5.Checked = true;

this.radioButton5.Location = new System.Drawing.Point(6, 33);

this.radioButton5.Name = "radioButton5";

this.radioButton5.Size = new System.Drawing.Size(77, 17);

this.radioButton5.TabIndex = 0;

this.radioButton5.TabStop = true;

this.radioButton5.Text = "Линейный";

this.radioButton5.UseVisualStyleBackColor = true;

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(538, 311);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(171, 20);

this.textBox1.TabIndex = 6;

this.textBox1.Text = "7000 3 1 0.9 0.2 2";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(537, 292);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(145, 13);

this.label1.TabIndex = 7;

this.label1.Text = "I, Rmax, Rmin, tmax, tmin, lm";

//

// progressBar1

//

this.progressBar1.Location = new System.Drawing.Point(537, 418);

this.progressBar1.Name = "progressBar1";

this.progressBar1.Size = new System.Drawing.Size(173, 23);

this.progressBar1.TabIndex = 8;

//

// forSisII30

//

this.forSisII30.Location = new System.Drawing.Point(448, 358);

this.forSisII30.Name = "forSisII30";

this.forSisII30.Size = new System.Drawing.Size(83, 83);

this.forSisII30.TabIndex = 1;

this.forSisII30.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII29

//

this.forSisII29.Location = new System.Drawing.Point(359, 358);

this.forSisII29.Name = "forSisII29";

this.forSisII29.Size = new System.Drawing.Size(83, 83);

this.forSisII29.TabIndex = 1;

this.forSisII29.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII28

//

this.forSisII28.Location = new System.Drawing.Point(270, 358);

this.forSisII28.Name = "forSisII28";

this.forSisII28.Size = new System.Drawing.Size(83, 83);

this.forSisII28.TabIndex = 1;

this.forSisII28.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII27

//

this.forSisII27.Location = new System.Drawing.Point(181, 358);

this.forSisII27.Name = "forSisII27";

this.forSisII27.Size = new System.Drawing.Size(83, 83);

this.forSisII27.TabIndex = 1;

this.forSisII27.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII26

//

this.forSisII26.Location = new System.Drawing.Point(92, 358);

this.forSisII26.Name = "forSisII26";

this.forSisII26.Size = new System.Drawing.Size(83, 83);

this.forSisII26.TabIndex = 1;

this.forSisII26.Values = new double[] {

0.030630804146933744D,

0.59983983989797529D,

0.85516131010612539D,

0.28582640284943694D,

0.63645630825145927D,

0.71228530430807047D,

0.71532702572426154D,

0.41830261024567422D,

0.35560618543792805D,

0.54895327545188055D,

0.69155330941619042D,

0.88530132401981454D,

0.48095011593818204D,

0.50501657719957482D,

0.99142113141316046D,

0.61081074812021607D,

0.88437328295985851D,

0.10696957684446572D,

0.53739905894612849D,

0.095145604617495841D,

0.94500066849635944D,

0.10701355203381439D,

0.64212648274476014D,

0.47465358044703193D,

0.31639747802000373D,

0.29411302706883896D,

0.0913197310135326D,

0.24855489248808235D,

0.15671738104741897D,

0.66912166526034556D,

0.86597344785275565D,

0.35234643488765993D,

0.20207364633775951D,

0.60034810686500184D,

0.748585246386279D,

0.69290315904324085D};

//

// forSisII25

//

this.forSisII25.Location = new System.Drawing.Point(3, 358);

this.forSisII25.Name = "forSisII25";

this.forSisII25.Size = new System.Drawing.Size(83, 83);

this.forSisII25.TabIndex = 1;

this.forSisII25.Values = new double[] {

0.10174499736248749D,

0.14172802080480756D,

0.53863698502007729D,

0.5889561491035652D,

0.14718602371736711D,

0.47326707256644363D,

0.96881003396995835D,

0.886750604438945D,

0.76593287324809134D,

0.88426247839083083D,

0.41464522965934369D,

0.58701040716236941D,

0.13422517391584124D,

0.8716158321460783D,

0.18816828037992506D,

0.93146333654013613D,

0.56332817047989381D,

0.57373712471394667D,

0.08823280552785509D,

0.76864029596030725D,

0.95903972301587448D,

0.949060135497274D,

0.84895727543577426D,

0.34943150232985221D,

0.77836776095366467D,

0.33119670596495115D,

0.35915390418803034D,

0.33355547642966521D,

0.13354153751094897D,

0.45164159985801278D,

0.4999481921549645D,

0.37614549760527233D,

0.092598688831831644D,

0.995353570671451D,

0.70986684817348922D,

0.55070570788844753D};

//

// forSisII21

//

this.forSisII21.Location = new System.Drawing.Point(181, 269);

this.forSisII21.Name = "forSisII21";

this.forSisII21.Size = new System.Drawing.Size(83, 83);

this.forSisII21.TabIndex = 0;

this.forSisII21.Values = new double[] {

0.49527310556512005D,

0.38349496078840223D,

0.77734852106186958D,

0.47678007347359325D,

0.46067751546421903D,

0.035599667129851721D,

0.25269243784839868D,

0.020351380119263837D,

0.15350857151369962D,

0.28887762422155477D,

0.548519163182247D,

0.188003528019415D,

0.46981522183391045D,

0.63654346514332272D,

0.39410643949830271D,

0.60456699766431332D,

0.37952760997206325D,

0.54800680072419661D,

0.081449253056873214D,

0.027532596153920792D,

0.32216856643658531D,

0.34552892173897892D,

0.52835623618604444D,

0.11590381949949256D,

0.16733835459097213D,

0.14612416417623134D,

0.26602139941697073D,

0.93426865708747353D,

0.99011865630285756D,

0.26739906811499925D,

0.44025367518898739D,

0.49699713825108349D,

0.28531352583566377D,

0.90936762462806309D,

0.36259514762209505D,

0.36127020295768519D};

//

// forSisII14

//

this.forSisII14.Location = new System.Drawing.Point(92, 180);

this.forSisII14.Name = "forSisII14";

this.forSisII14.Size = new System.Drawing.Size(83, 83);

this.forSisII14.TabIndex = 0;

this.forSisII14.Values = new double[] {

0.77983346524640618D,

0.781154889045821D,

0.63644801109863813D,

0.30004559750670828D,

0.1124151675554063D,

0.80562415523716446D,

0.240419836826818D,

0.049859585729362249D,

0.71091335812160439D,

0.46064878928505293D,

0.11258858121586432D,

0.8724238564597554D,

0.23492178331824104D,

0.49907613429197861D,

0.834819307939531D,

0.47607096679325728D,

0.18732571889987482D,

0.88346463622686666D,

0.33714831915551252D,

0.14974418382614116D,

0.85513872460235785D,

0.661444701562377D,

0.30944171888262112D,

0.31455335128798773D,

0.769578267712881D,

0.31842378076092515D,

0.030232723816406321D,

0.88610950339870043D,

0.84862394903256744D,

0.051522024931163542D,

0.90269000218375117D,

0.56044759720584736D,

0.27394831752122767D,

0.032836236540617535D,

0.82473353241790714D,

0.59837272092624227D};

//

// forSisII7

//

this.forSisII7.Location = new System.Drawing.Point(3, 91);

this.forSisII7.Name = "forSisII7";

this.forSisII7.Size = new System.Drawing.Size(83, 83);

this.forSisII7.TabIndex = 0;

this.forSisII7.Values = new double[] {

0.91155878962555847D,

0.78140365555016489D,

0.91511742859851453D,

0.51654441119942085D,

0.10510908491216092D,

0.47849665045668216D,

0.40499046137788819D,

0.7066895653059192D,

0.033535582960367009D,

0.39359754295721533D,

0.69934687376923244D,

0.49104252201087889D,

0.43257976017546829D,

0.96418270839573006D,

0.30967141935120868D,

0.40184888960879711D,

0.57221930547255062D,

0.21285096379595389D,

0.50613147835532735D,

0.11838876368402912D,

0.47171317016319986D,

0.32189465981065046D,

0.53046732979382727D,

0.79399138679448111D,

0.77539143142075806D,

0.35431033342811757D,

0.20123563669679484D,

0.65939133319044085D,

0.55795878710130176D,

0.552140984010017D,

0.59847553893852767D,

0.2020855467776235D,

0.67434260187407147D,

0.23644658701328866D,

0.56451217716769886D,

0.40592206334970987D};

//

// forSisII20

//

this.forSisII20.Location = new System.Drawing.Point(92, 269);

this.forSisII20.Name = "forSisII20";

this.forSisII20.Size = new System.Drawing.Size(83, 83);

this.forSisII20.TabIndex = 0;

this.forSisII20.Values = new double[] {

0.84441024523433772D,

0.25769853417654454D,

0.54878723833187826D,

0.09644972071817598D,

0.9462115620012449D,

0.50495175016343208D,

0.85001816127915786D,

0.9604666176999298D,

0.82805840057696145D,

0.09161310367826983D,

0.91748190667362972D,

0.25774381880543373D,

0.82188053839927566D,

0.84870208699661409D,

0.08634613691193338D,

0.42482248387524041D,

0.74876361840812655D,

0.58628051615612609D,

0.67216446933903007D,

0.87437951745203679D,

0.030108343823863355D,

0.35107941150249888D,

0.4149396412144134D,

0.24125372210575907D,

0.57362788430118372D,

0.053098899802704762D,

0.51371094794651073D,

0.643384020143833D,

0.83202368665115145D,

0.079439478032029928D,

0.98977824393183844D,

0.32729979805988252D,

0.64449669636995377D,

0.27816543461669491D,

0.12358758790585565D,

0.85208850207370168D};

//

// forSisII13

//

this.forSisII13.Location = new System.Drawing.Point(3, 180);

this.forSisII13.Name = "forSisII13";

this.forSisII13.Size = new System.Drawing.Size(83, 83);

this.forSisII13.TabIndex = 0;

this.forSisII13.Values = new double[] {

0.42606080902091265D,

0.058536016875196259D,

0.90100960382307393D,

0.57449917801399675D,

0.44381818615077911D,

0.13891444268585856D,

0.68279671142008D,

0.36240827169381468D,

0.882141262703641D,

0.32311490193154424D,

0.29993504160080803D,

0.87378819467210589D,

0.17105043314911911D,

0.059948225533565609D,

0.08933763117032946D,

0.48105891443838317D,

0.56444842487780766D,

0.018501971391263406D,

0.3671070101517751D,

0.49333083373183889D,

0.84275889296213113D,

0.30753351622611913D,

0.52282183641699231D,

0.89311266219854013D,

0.99124080221692135D,

0.2066341411353248D,

0.12844293104877833D,

0.74861892254493145D,

0.1018348597464314D,

0.5901821086137472D,

0.61324712243548D,

0.5437151037825807D,

0.72511449909075842D,

0.23425763483823167D,

0.99876167485432776D,

0.69022760153293028D};

//

// forSisII6

//

this.forSisII6.Location = new System.Drawing.Point(448, 2);

this.forSisII6.Name = "forSisII6";

this.forSisII6.Size = new System.Drawing.Size(83, 83);

this.forSisII6.TabIndex = 0;

this.forSisII6.Values = new double[] {

0.68527091233305215D,

0.097665291325033307D,

0.43056133130125762D,

0.89119390113800478D,

0.66828270008241886D,

0.052531369520598732D,

0.35911245940211811D,

0.25051298236963943D,

0.75316385820189669D,

0.63992275327440484D,

0.47233358699471389D,

0.77207546111758585D,

0.99010184686170044D,

0.50871932949345533D,

0.76285137318207485D,

0.62599472497869035D,

0.9110584244649198D,

0.63025675417401672D,

0.0093296091115705714D,

0.11852729279479352D,

0.535714180458204D,

0.0078969416245338233D,

0.5832583757970754D,

0.23718404501545432D,

0.010471511637080233D,

0.93417778887514857D,

0.26057341194738326D,

0.29195915408989376D,

0.70986777949606428D,

0.75621993781822727D,

0.14867369558134755D,

0.93679279831088746D,

0.061186550679237839D,

0.39911002963739917D,

0.79578800210533107D,

0.4667831335527744D};

//

// forSisII19

//

this.forSisII19.Location = new System.Drawing.Point(3, 269);

this.forSisII19.Name = "forSisII19";

this.forSisII19.Size = new System.Drawing.Size(83, 83);

this.forSisII19.TabIndex = 0;

this.forSisII19.Values = new double[] {

0.71012814608874175D,

0.89110809466387519D,

0.80942583308062788D,

0.28246658867339908D,

0.6904937954109599D,

0.22661834779503678D,

0.55844419801535283D,

0.12450007587880831D,

0.60252488665400306D,

0.81802982083430043D,

0.873380661883103D,

0.43743310702845134D,

0.82367601609960017D,

0.52993674181864447D,

0.34639520866162854D,

0.59451729599131142D,

0.343692021604484D,

0.64349073108448218D,

0.83995459593830379D,

0.71426539808244693D,

0.22083251980172122D,

0.28883848445901578D,

0.53174167523707339D,

0.72494325401491633D,

0.3718713356050995D,

0.30412372541805904D,

0.0086254822130433662D,

0.0081354547329877749D,

0.78727367882955523D,

0.93336592285584929D,

0.77341706947116973D,

0.92605189463405491D,

0.764166826272461D,

0.70603883252760336D,

0.47082402253096178D,

0.3534885138987976D};

//

// forSisII12

//

this.forSisII12.Location = new System.Drawing.Point(448, 91);

this.forSisII12.Name = "forSisII12";

this.forSisII12.Size = new System.Drawing.Size(83, 83);

this.forSisII12.TabIndex = 0;

this.forSisII12.Values = new double[] {

0.3089881983161849D,

0.6699096293514174D,

0.70406955839324259D,

0.36857237171780893D,

0.532499024426797D,

0.87376948998950865D,

0.27874286485777372D,

0.91363383033947732D,

0.269624221729871D,

0.422635884221008D,

0.3750837493571843D,

0.11554181767419997D,

0.8211300497973012D,

0.28566690687354046D,

0.18757047419788803D,

0.30570758148362281D,

0.35755676792820767D,

0.57538372630969792D,

0.78499342025490171D,

0.42128966162972603D,

0.35936641942680181D,

0.084482579065711508D,

0.91059525306829958D,

0.38637006999290086D,

0.21799286558199341D,

0.550308743282365D,

0.33722639704925306D,

0.66915896379815365D,

0.044612751363130636D,

0.94732876724904813D,

0.67326628075599027D,

0.11763718357199672D,

0.059112719287682657D,

0.992906694762831D,

0.28552909767512658D,

0.67378239225306658D};

//

// forSisII5

//

this.forSisII5.Location = new System.Drawing.Point(359, 2);

this.forSisII5.Name = "forSisII5";

this.forSisII5.Size = new System.Drawing.Size(83, 83);

this.forSisII5.TabIndex = 0;

this.forSisII5.Values = new double[] {

0.93942117269123959D,

0.47138222422049486D,

0.18176637365565002D,

0.3470630298122126D,

0.010095619601242067D,

0.61810784489759607D,

0.45531903228504539D,

0.94923545138409149D,

0.92903997606087474D,

0.89308366453884336D,

0.18700540493568657D,

0.50160677195601477D,

0.74579544865796132D,

0.46770948798754697D,

0.56849512670584723D,

0.18878309996276307D,

0.92104541227270165D,

0.049534360901235773D,

0.99412209447199573D,

0.22450561925047338D,

0.75931437628311771D,

0.31769948840034168D,

0.15057950613581553D,

0.982190281144432D,

0.53654309294025559D,

0.60958390105962D,

0.86902107897634673D,

0.32229545448082286D,

0.74936960346501769D,

0.25744656578518754D,

0.056429098386517308D,

0.82822335503447031D,

0.00013780919841388669D,

0.5137880819448214D,

0.36628640879238322D,

0.88617454370771276D};

//

// forSisII18

//

this.forSisII18.Location = new System.Drawing.Point(448, 180);

this.forSisII18.Name = "forSisII18";

this.forSisII18.Size = new System.Drawing.Size(83, 83);

this.forSisII18.TabIndex = 0;

this.forSisII18.Values = new double[] {

0.39361735265404796D,

0.43793039044268911D,

0.411194042028484D,

0.74125857452920574D,

0.62916354678066611D,

0.96135980168420809D,

0.45733009393202612D,

0.32490920104315D,

0.36330333834667844D,

0.83561962509323828D,

0.92336351514019233D,

0.57690326337558373D,

0.37883364054320084D,

0.48448318079322722D,

0.19659177129929503D,

0.0095783583864468887D,

0.99878460029083516D,

0.061023478424653166D,

0.91446801596994887D,

0.62172168429089791D,

0.32080271808467931D,

0.19957609251121808D,

0.810519936871957D,

0.40051171854162204D,

0.74908676592124945D,

0.1330235778042225D,

0.1998658479190738D,

0.6715934102756872D,

0.83665456615232614D,

0.35878204990121632D,

0.50146896275760089D,

0.23200736671313987D,

0.10142307919516372D,

0.68232058299813447D,

0.79516574730871514D,

0.48311502183001259D};

//

// forSisII11

//

this.forSisII11.Location = new System.Drawing.Point(359, 91);

this.forSisII11.Name = "forSisII11";

this.forSisII11.Size = new System.Drawing.Size(83, 83);

this.forSisII11.TabIndex = 0;

this.forSisII11.Values = new double[] {

0.46568140595484592D,

0.11651344137103922D,

0.7613522898225823D,

0.49513241299201383D,

0.58068914971346464D,

0.10959646902493968D,

0.71429133262219435D,

0.4569689503204864D,

0.18956176386660048D,

0.93497942524728339D,

0.6107386157897946D,

0.78980382615225564D,

0.52934329515758127D,

0.56235463198384017D,

0.63268730073826729D,

0.35685835469367838D,

0.11694875737510098D,

0.60112448716588529D,

0.85570456825928043D,

0.62765799538588984D,

0.26069171133483376D,

0.099202103493363641D,

0.036460620833775319D,

0.074632710811976671D,

0.3374156702018416D,

0.61436565435229129D,

0.36677280551138D,

0.99486423749237518D,

0.816637631420343D,

0.24694875639255567D,

0.57556898918727828D,

0.34737804501661007D,

0.67561270048637534D,

0.083517674395589936D,

0.4734329248189148D,

0.48280662460383339D};

//

// forSisII4

//

this.forSisII4.Location = new System.Drawing.Point(270, 2);

this.forSisII4.Name = "forSisII4";

this.forSisII4.Size = new System.Drawing.Size(83, 83);

this.forSisII4.TabIndex = 0;

this.forSisII4.Values = new double[] {

0.98777237673652007D,

0.337890583713488D,

0.33126700452122232D,

0.72548885304736388D,

0.79262389838352054D,

0.2099460923159244D,

0.89963464294543238D,

0.63896486146327336D,

0.13918766106441041D,

0.5949767751595828D,

0.93619538980359D,

0.69801580193360135D,

0.66738445622259024D,

0.48737604286865149D,

0.48960586380660809D,

0.93203265635857013D,

0.59828830538237854D,

0.10695362235743255D,

0.40275380406656947D,

0.29649285566829742D,

0.9125226698408474D,

0.647028615068192D,

0.77499032103223275D,

0.57696389573484841D,

0.89487153938732644D,

0.36396519763579832D,

0.46710409245784584D,

0.91163749709335973D,

0.58425968307268794D,

0.0293523725258896D,

0.6828743231868718D,

0.90878817155435132D,

0.74497590947196624D,

0.53285495542588412D,

0.10042056958210681D,

0.519881597496514D};

//

// forSisII24

//

this.forSisII24.Location = new System.Drawing.Point(448, 269);

this.forSisII24.Name = "forSisII24";

this.forSisII24.Size = new System.Drawing.Size(83, 83);

this.forSisII24.TabIndex = 0;

this.forSisII24.Values = new double[] {

0.41231303960658283D,

0.75974051596584757D,

0.57385540407796176D,

0.8701067282213395D,

0.78708493606517316D,

0.51875188132689887D,

0.82713583429676285D,

0.14784688835397683D,

0.78323170532622921D,

0.942315516966542D,

0.27522855125145451D,

0.87203439319135356D,

0.77375470696657644D,

0.18215846930731017D,

0.84753731537961274D,

0.98196808480749287D,

0.74712687206786443D,

0.077153064346477881D,

0.13418667490323386D,

0.34074376166832809D,

0.56290026268125526D,

0.754303108786374D,

0.83061731366003744D,

0.42865870074772217D,

0.74284199985807853D,

0.98799714585207266D,

0.054705178390585432D,

0.10983528853852083D,

0.91210245197271111D,

0.027407218249238665D,

0.95303989246163512D,

0.13452950079670617D,

0.38695547328654467D,

0.969724266310094D,

0.5197196167519873D,

0.838547789416531D};

//

// forSisII17

//

this.forSisII17.Location = new System.Drawing.Point(359, 180);

this.forSisII17.Name = "forSisII17";

this.forSisII17.Size = new System.Drawing.Size(83, 83);

this.forSisII17.TabIndex = 0;

this.forSisII17.Values = new double[] {

0.53309821827947079D,

0.53264707584523D,

0.50940791960312426D,

0.39377078851394859D,

0.81989278077142913D,

0.6271434326782559D,

0.7937321904086192D,

0.95255602242078452D,

0.088736646384343809D,

0.59506969926649222D,

0.1378827901267832D,

0.40210121376537772D,

0.18181505714627683D,

0.70090623837937893D,

0.16166129948648686D,

0.6678228451254884D,

0.39866828052265024D,

0.75967680791377867D,

0.95698133481525882D,

0.658009930820209D,

0.92912320230581014D,

0.1451967033302396D,

0.12726472836326097D,

0.7990877902131005D,

0.46404670293631345D,

0.71730054575824209D,

0.23574443638126572D,

0.75582448707699057D,

0.98927562450490691D,

0.14069905045474834D,

0.48507891990480895D,

0.80403044065648244D,

0.66243885255532287D,

0.0089895259630817584D,

0.448869866528022D,

0.21447979622263452D};

//

// forSisII10

//

this.forSisII10.Location = new System.Drawing.Point(270, 91);

this.forSisII10.Name = "forSisII10";

this.forSisII10.Size = new System.Drawing.Size(83, 83);

this.forSisII10.TabIndex = 0;

this.forSisII10.Values = new double[] {

0.23882582701687971D,

0.021224504812259463D,

0.69743901244245421D,

0.05106191106655724D,

0.8052746489668613D,

0.97949242777167467D,

0.948973434487811D,

0.8114427383111058D,

0.7892803143659981D,

0.024548222787933529D,

0.8900785683142387D,

0.70405857716876019D,

0.40483783018069242D,

0.857800254532043D,

0.79128750916164714D,

0.80072916662354454D,

0.09956493885236091D,

0.060968351578790859D,

0.80909744874066558D,

0.85742501535332993D,

0.81069180081165015D,

0.418258745883712D,

0.5443578337060091D,

0.922663215046126D,

0.91478089332337531D,

0.39188445843378289D,

0.037781521229902991D,

0.51626200625498875D,

0.91294160341515285D,

0.75519470020904889D,

0.95915984174197533D,

0.91856441549889944D,

0.185139346954012D,

0.077608826140690979D,

0.38845283835588623D,

0.99151842668257584D};

//

// forSisII3

//

this.forSisII3.Location = new System.Drawing.Point(181, 2);

this.forSisII3.Name = "forSisII3";

this.forSisII3.Size = new System.Drawing.Size(83, 83);

this.forSisII3.TabIndex = 0;

this.forSisII3.Values = new double[] {

0.47834148559642092D,

0.38320752903037125D,

0.096078320451117263D,

0.69241045354512076D,

0.10494682523652298D,

0.17750706019695245D,

0.77483408002873611D,

0.6555590725762579D,

0.84868469268487989D,

0.78559099640026264D,

0.97369716594633515D,

0.97400840882864237D,

0.68193410787821473D,

0.76809182146940935D,

0.78048036097571272D,

0.94248666192520725D,

0.23004823281897616D,

0.5678857572227185D,

0.51465175557632548D,

0.38949487609299593D,

0.7220816583009817D,

0.16923667544929155D,

0.48417901316852263D,

0.32863526666939036D,

0.21978021237057643D,

0.27922630649023983D,

0.30146239246309847D,

0.17631498220205075D,

0.74030259053236924D,

0.21597076170890162D,

0.7828431086534835D,

0.74279575782958218D,

0.30247470145229005D,

0.83205832952263692D,

0.71272653467614511D,

0.75487652549281559D};

//

// forSisII23

//

this.forSisII23.Location = new System.Drawing.Point(359, 269);

this.forSisII23.Name = "forSisII23";

this.forSisII23.Size = new System.Drawing.Size(83, 83);

this.forSisII23.TabIndex = 0;

this.forSisII23.Values = new double[] {

0.14657436457768752D,

0.15515013977659406D,

0.7156202558966448D,

0.29935961090929786D,

0.99994171736759219D,

0.480721945166924D,

0.56470549784819857D,

0.15611991014150897D,

0.45901426228648717D,

0.92449272606731059D,

0.35215150627873443D,

0.29729655957654888D,

0.19103379183962652D,

0.57643256828954093D,

0.097180771221025281D,

0.90146165895343833D,

0.28511732690274588D,

0.24696361704122444D,

0.51912060450721564D,

0.30910481014712937D,

0.89902851586184862D,

0.76744305378172684D,

0.47263024117454433D,

0.82572051874628316D,

0.876044667733854D,

0.59851909782668533D,

0.91525648204388865D,

0.63271393097597828D,

0.0027478877467791958D,

0.23090140811675294D,

0.67153370737635243D,

0.84987577835557782D,

0.055365286793264232D,

0.025603835482897162D,

0.79591229734751967D,

0.074898093042382083D};

//

// forSisII16

//

this.forSisII16.Location = new System.Drawing.Point(270, 180);

this.forSisII16.Name = "forSisII16";

this.forSisII16.Size = new System.Drawing.Size(83, 83);

this.forSisII16.TabIndex = 0;

this.forSisII16.Values = new double[] {

0.8522655013260737D,

0.21529214466702759D,

0.3895531587254038D,

0.24135971313405769D,

0.60453117760109309D,

0.32805910302701363D,

0.86962100438290324D,

0.2952874863032659D,

0.92707480021150546D,

0.0041658328865495659D,

0.98528119036242423D,

0.16387002224282829D,

0.11878999048787635D,

0.88138144970004517D,

0.4576784309268363D,

0.0555110844110656D,

0.31293772501542128D,

0.40362172452901568D,

0.855848009630967D,

0.37913131079596063D,

0.68251989860204976D,

0.88989973715036164D,

0.42331494317544388D,

0.4014226195409068D,

0.56546546312303536D,

0.93199156687222029D,

0.1533720223947298D,

0.22811285416973423D,

0.25295901869095816D,

0.50227572792315656D,

0.24193127278328466D,

0.16542995635672936D,

0.78052027094202125D,

0.022282678178643191D,

0.049196157627364692D,

0.069825182235718319D};

//

// forSisII9

//

this.forSisII9.Location = new System.Drawing.Point(181, 91);

this.forSisII9.Name = "forSisII9";

this.forSisII9.Size = new System.Drawing.Size(83, 83);

this.forSisII9.TabIndex = 0;

this.forSisII9.Values = new double[] {

0.69904809198297935D,

0.284878596796132D,

0.481309261397137D,

0.50517714373077138D,

0.99125401861558393D,

0.5514361083281395D,

0.78818471207664564D,

0.814079003321975D,

0.41806762638411837D,

0.032005584347995733D,

0.45488987697981759D,

0.6369363957256714D,

0.965337674583D,

0.59018771051903618D,

0.99567413050479914D,

0.36373928439046221D,

0.62990448373831087D,

0.086830393451652679D,

0.27784798074413464D,

0.86927991913132363D,

0.60186664136213563D,

0.032034021817163576D,

0.3962823322025511D,

0.25631389918565467D,

0.13501428725897069D,

0.47384664624642892D,

0.10964427474403952D,

0.79338667345856628D,

0.36803670850025338D,

0.41741662817886871D,

0.28918114597405359D,

0.38536972849879869D,

0.41792199687004183D,

0.50176121969789322D,

0.95269547028126911D,

0.88487722905579824D};

//

// forSisII2

//

this.forSisII2.Location = new System.Drawing.Point(92, 2);

this.forSisII2.Name = "forSisII2";

this.forSisII2.Size = new System.Drawing.Size(83, 83);

this.forSisII2.TabIndex = 0;

this.forSisII2.Values = new double[] {

0.11571864789152456D,

0.037306887580690389D,

0.66933409202347238D,

0.74005139467308834D,

0.77756203747240926D,

0.253600835918263D,

0.069695625952303233D,

0.72962221537233429D,

0.014385248541080043D,

0.93759742515981082D,

0.9692095699576706D,

0.34842624158990859D,

0.95081379960794643D,

0.54382624735302587D,

0.27536446055181535D,

0.089101289906120526D,

0.068451027417765473D,

0.28568140011545334D,

0.99628096446221737D,

0.71047636294293981D,

0.51703819423776032D,

0.45370437319097312D,

0.79874925957934428D,

0.20514778848977192D,

0.41577295186732566D,

0.25382098613950471D,

0.399857057910346D,

0.11709812475233251D,

0.18637767303100677D,

0.99588494282024209D,

0.53455460934646182D,

0.71338020205189479D,

0.59134290767430464D,

0.72389939554217242D,

0.62963625072950324D,

0.072698380366292961D};

//

// forSisII22

//

this.forSisII22.Location = new System.Drawing.Point(270, 269);

this.forSisII22.Name = "forSisII22";

this.forSisII22.Size = new System.Drawing.Size(83, 83);

this.forSisII22.TabIndex = 0;

this.forSisII22.Values = new double[] {

0.65948661494044891D,

0.69325879760703946D,

0.88945203269340656D,

0.63499542867531789D,

0.15529973625917906D,

0.64054064109946629D,

0.503022123828075D,

0.5041944084242892D,

0.025588468660408849D,

0.097616053231813035D,

0.04983719859730322D,

0.19899819102091632D,

0.79415568792920355D,

0.12991310894950903D,

0.39512542746734125D,

0.35181084012231362D,

0.11451593698678349D,

0.71779119629310029D,

0.22566523879098949D,

0.66201427470055141D,

0.23855762800134608D,

0.46418630353370044D,

0.32427844280576262D,

0.52374105133290449D,

0.853743778007917D,

0.95259750119997078D,

0.54324454234132757D,

0.018500305720837929D,

0.403042815813349D,

0.25582936790577571D,

0.757083333915604D,

0.226914404065774D,

0.91418999662352263D,

0.20266608018552237D,

0.42961467496567157D,

0.37519222981072603D};

//

// forSisII15

//

this.forSisII15.Location = new System.Drawing.Point(181, 180);

this.forSisII15.Name = "forSisII15";

this.forSisII15.Size = new System.Drawing.Size(83, 83);

this.forSisII15.TabIndex = 0;

this.forSisII15.Values = new double[] {

0.39622936742204679D,

0.36128553578689954D,

0.55517662668376078D,

0.876497553138294D,

0.95068224936289814D,

0.29455485115505514D,

0.17955931982936305D,

0.31815689863551266D,

0.20398378754220148D,

0.20085886688942969D,

0.32294243682312895D,

0.056464564081497755D,

0.60075951535290084D,

0.18274376922414815D,

0.59886426506511137D,

0.87355171138120424D,

0.49823415675118293D,

0.96762197602895184D,

0.83414075236494689D,

0.19530031140674853D,

0.36898035480127689D,

0.36571098136050206D,

0.78125165066740088D,

0.20270223505920834D,

0.097296098758138758D,

0.48452181810723705D,

0.10115159261094014D,

0.76975910075463316D,

0.34053271931620904D,

0.82292279453152917D,

0.28480819439739369D,

0.5914896077436812D,

0.70029843398383751D,

0.019933197656615262D,

0.95558147270026683D,

0.75323040119988394D};

//

// forSisII8

//

this.forSisII8.Location = new System.Drawing.Point(92, 91);

this.forSisII8.Name = "forSisII8";

this.forSisII8.Size = new System.Drawing.Size(83, 83);

this.forSisII8.TabIndex = 0;

this.forSisII8.Values = new double[] {

0.16261456960933962D,

0.34916768565269546D,

0.71133202533765327D,

0.944002776846291D,

0.28462698370433737D,

0.0061215441702499726D,

0.319757263790703D,

0.65288491111848734D,

0.62965506437684182D,

0.48677997825982977D,

0.4177407903679371D,

0.22029904658920088D,

0.65696510284066434D,

0.88353162253439965D,

0.728680247314591D,

0.94656802059457079D,

0.36852532782057551D,

0.23431436355892307D,

0.0062118750094491406D,

0.030518386061544708D,

0.5800338851194986D,

0.35247439162455241D,

0.77920145438015531D,

0.46616043125566115D,

0.193403258544115D,

0.40980021907472991D,

0.97762417931930357D,

0.38106099301067226D,

0.916050672492036D,

0.73145282907944775D,

0.35616613009766029D,

0.58082631769628557D,

0.22716229652388129D,

0.84563386386522732D,

0.71093714177186462D,

0.14906647109848747D};

//

// forSisII1

//

this.forSisII1.Location = new System.Drawing.Point(3, 2);

this.forSisII1.Name = "forSisII1";

this.forSisII1.Size = new System.Drawing.Size(83, 83);

this.forSisII1.TabIndex = 0;

this.forSisII1.Values = new double[] {

0.25628995069129856D,

0.89013797551865592D,

0.91067332770241116D,

0.36285881063102687D,

0.045953717569799032D,

0.12836673954891356D,

0.57304717068236655D,

0.610516120498309D,

0.066781027739299934D,

0.88085254602173924D,

0.11279399791396876D,

0.45700109678180939D,

0.09424254721693813D,

0.3382401738028229D,

0.22296427992310575D,

0.46598407042491441D,

0.013721322647166124D,

0.92506308663872217D,

0.1731965160803853D,

0.81014017798478721D,

0.5699662312725402D,

0.2451715940819921D,

0.750599518302176D,

0.8748267646296074D,

0.028497364850946408D,

0.93869627078003082D,

0.73683423862645137D,

0.89820223529739407D,

0.13061384816216951D,

0.83691447267165153D,

0.99313675006531954D,

0.811331238975437D,

0.172594480762535D,

0.57961377621610355D,

0.69027806990327223D,

0.47838735230191953D};

//

// button3

//

this.button3.Location = new System.Drawing.Point(537, 387);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(172, 23);

this.button3.TabIndex = 9;

this.button3.Text = "Сбросить";

this.button3.UseVisualStyleBackColor = true;

this.button3.Click += new System.EventHandler(this.button3\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(716, 447);

this.Controls.Add(this.button3);

this.Controls.Add(this.progressBar1);

this.Controls.Add(this.label1);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.groupBox2);

this.Controls.Add(this.groupBox1);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.forSisII30);

this.Controls.Add(this.forSisII29);

this.Controls.Add(this.forSisII28);

this.Controls.Add(this.forSisII27);

this.Controls.Add(this.forSisII26);

this.Controls.Add(this.forSisII25);

this.Controls.Add(this.forSisII21);

this.Controls.Add(this.forSisII14);

this.Controls.Add(this.forSisII7);

this.Controls.Add(this.forSisII20);

this.Controls.Add(this.forSisII13);

this.Controls.Add(this.forSisII6);

this.Controls.Add(this.forSisII19);

this.Controls.Add(this.forSisII12);

this.Controls.Add(this.forSisII5);

this.Controls.Add(this.forSisII18);

this.Controls.Add(this.forSisII11);

this.Controls.Add(this.forSisII4);

this.Controls.Add(this.forSisII24);

this.Controls.Add(this.forSisII17);

this.Controls.Add(this.forSisII10);

this.Controls.Add(this.forSisII3);

this.Controls.Add(this.forSisII23);

this.Controls.Add(this.forSisII16);

this.Controls.Add(this.forSisII9);

this.Controls.Add(this.forSisII2);

this.Controls.Add(this.forSisII22);

this.Controls.Add(this.forSisII15);

this.Controls.Add(this.forSisII8);

this.Controls.Add(this.forSisII1);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;

this.MaximizeBox = false;

this.MinimizeBox = false;

this.Name = "Form1";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Сис. ИИ, 6/1";

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private ForSisII forSisII1;

private ForSisII forSisII2;

private ForSisII forSisII3;

private ForSisII forSisII4;

private ForSisII forSisII5;

private ForSisII forSisII6;

private ForSisII forSisII7;

private ForSisII forSisII8;

private ForSisII forSisII9;

private ForSisII forSisII10;

private ForSisII forSisII11;

private ForSisII forSisII12;

private ForSisII forSisII13;

private ForSisII forSisII14;

private ForSisII forSisII15;

private ForSisII forSisII16;

private ForSisII forSisII17;

private ForSisII forSisII18;

private ForSisII forSisII19;

private ForSisII forSisII20;

private ForSisII forSisII21;

private ForSisII forSisII22;

private ForSisII forSisII23;

private ForSisII forSisII24;

private ForSisII forSisII25;

private ForSisII forSisII26;

private ForSisII forSisII27;

private System.Windows.Forms.Button button1;

private ForSisII forSisII28;

private ForSisII forSisII29;

private ForSisII forSisII30;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.RadioButton radioButton4;

private System.Windows.Forms.RadioButton radioButton3;

private System.Windows.Forms.RadioButton radioButton2;

private System.Windows.Forms.RadioButton radioButton1;

private System.Windows.Forms.GroupBox groupBox2;

private System.Windows.Forms.RadioButton radioButton6;

private System.Windows.Forms.RadioButton radioButton5;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.ProgressBar progressBar1;

private System.Windows.Forms.Button button3;

}

}

namespace kohonen

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.button1 = new System.Windows.Forms.Button();

this.button2 = new System.Windows.Forms.Button();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.radioButton4 = new System.Windows.Forms.RadioButton();

this.radioButton3 = new System.Windows.Forms.RadioButton();

this.radioButton2 = new System.Windows.Forms.RadioButton();

this.radioButton1 = new System.Windows.Forms.RadioButton();

this.groupBox2 = new System.Windows.Forms.GroupBox();

this.radioButton6 = new System.Windows.Forms.RadioButton();

this.radioButton5 = new System.Windows.Forms.RadioButton();

this.textBox1 = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.progressBar1 = new System.Windows.Forms.ProgressBar();

this.forSisII30 = new kohonen.ForSisII();

this.forSisII29 = new kohonen.ForSisII();

this.forSisII28 = new kohonen.ForSisII();

this.forSisII27 = new kohonen.ForSisII();

this.forSisII26 = new kohonen.ForSisII();

this.forSisII25 = new kohonen.ForSisII();

this.forSisII21 = new kohonen.ForSisII();

this.forSisII14 = new kohonen.ForSisII();

this.forSisII7 = new kohonen.ForSisII();

this.forSisII20 = new kohonen.ForSisII();

this.forSisII13 = new kohonen.ForSisII();

this.forSisII6 = new kohonen.ForSisII();

this.forSisII19 = new kohonen.ForSisII();

this.forSisII12 = new kohonen.ForSisII();

this.forSisII5 = new kohonen.ForSisII();

this.forSisII18 = new kohonen.ForSisII();

this.forSisII11 = new kohonen.ForSisII();

this.forSisII4 = new kohonen.ForSisII();

this.forSisII24 = new kohonen.ForSisII();

this.forSisII17 = new kohonen.ForSisII();

this.forSisII10 = new kohonen.ForSisII();

this.forSisII3 = new kohonen.ForSisII();

this.forSisII23 = new kohonen.ForSisII();

this.forSisII16 = new kohonen.ForSisII();

this.forSisII9 = new kohonen.ForSisII();

this.forSisII2 = new kohonen.ForSisII();

this.forSisII22 = new kohonen.ForSisII();

this.forSisII15 = new kohonen.ForSisII();

this.forSisII8 = new kohonen.ForSisII();

this.forSisII1 = new kohonen.ForSisII();

this.button3 = new System.Windows.Forms.Button();

this.groupBox1.SuspendLayout();

this.groupBox2.SuspendLayout();

this.SuspendLayout();

//

// button1

//

this.button1.Location = new System.Drawing.Point(537, 363);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(172, 23);

this.button1.TabIndex = 2;

this.button1.Text = "Обучиться";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// button2

//

this.button2.Location = new System.Drawing.Point(537, 337);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(172, 23);

this.button2.TabIndex = 3;

this.button2.Text = "Загрузить выборку";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// groupBox1

//

this.groupBox1.Controls.Add(this.radioButton4);

this.groupBox1.Controls.Add(this.radioButton3);

this.groupBox1.Controls.Add(this.radioButton2);

this.groupBox1.Controls.Add(this.radioButton1);

this.groupBox1.Location = new System.Drawing.Point(537, 2);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(172, 163);

this.groupBox1.TabIndex = 4;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Алгоритм обновления";

this.groupBox1.Enter += new System.EventHandler(this.groupBox1\_Enter);

//

// radioButton4

//

this.radioButton4.AutoSize = true;

this.radioButton4.Location = new System.Drawing.Point(8, 129);

this.radioButton4.Name = "radioButton4";

this.radioButton4.Size = new System.Drawing.Size(47, 17);

this.radioButton4.TabIndex = 0;

this.radioButton4.Text = "GAS";

this.radioButton4.UseVisualStyleBackColor = true;

//

// radioButton3

//

this.radioButton3.AutoSize = true;

this.radioButton3.Location = new System.Drawing.Point(9, 89);

this.radioButton3.Name = "radioButton3";

this.radioButton3.Size = new System.Drawing.Size(61, 17);

this.radioButton3.TabIndex = 0;

this.radioButton3.Text = "WTMM";

this.radioButton3.UseVisualStyleBackColor = true;

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Checked = true;

this.radioButton2.Location = new System.Drawing.Point(9, 51);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(52, 17);

this.radioButton2.TabIndex = 0;

this.radioButton2.TabStop = true;

this.radioButton2.Text = "WTM";

this.radioButton2.UseVisualStyleBackColor = true;

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Location = new System.Drawing.Point(9, 19);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(50, 17);

this.radioButton1.TabIndex = 0;

this.radioButton1.Text = "WTA";

this.radioButton1.UseVisualStyleBackColor = true;

//

// groupBox2

//

this.groupBox2.Controls.Add(this.radioButton6);

this.groupBox2.Controls.Add(this.radioButton5);

this.groupBox2.Location = new System.Drawing.Point(539, 171);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(171, 118);

this.groupBox2.TabIndex = 5;

this.groupBox2.TabStop = false;

this.groupBox2.Text = "Алгоритм пересчета";

//

// radioButton6

//

this.radioButton6.AutoSize = true;

this.radioButton6.Location = new System.Drawing.Point(6, 71);

this.radioButton6.Name = "radioButton6";

this.radioButton6.Size = new System.Drawing.Size(114, 17);

this.radioButton6.TabIndex = 0;

this.radioButton6.Text = "Гиперболический";

this.radioButton6.UseVisualStyleBackColor = true;

//

// radioButton5

//

this.radioButton5.AutoSize = true;

this.radioButton5.Checked = true;

this.radioButton5.Location = new System.Drawing.Point(6, 33);

this.radioButton5.Name = "radioButton5";

this.radioButton5.Size = new System.Drawing.Size(77, 17);

this.radioButton5.TabIndex = 0;

this.radioButton5.TabStop = true;

this.radioButton5.Text = "Линейный";

this.radioButton5.UseVisualStyleBackColor = true;

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(538, 311);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(171, 20);

this.textBox1.TabIndex = 6;

this.textBox1.Text = "7000 3 1 0.9 0.2 2";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(537, 292);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(145, 13);

this.label1.TabIndex = 7;

this.label1.Text = "I, Rmax, Rmin, tmax, tmin, lm";

//

// progressBar1

//

this.progressBar1.Location = new System.Drawing.Point(537, 418);

this.progressBar1.Name = "progressBar1";

this.progressBar1.Size = new System.Drawing.Size(173, 23);

this.progressBar1.TabIndex = 8;

//

// forSisII30

//

this.forSisII30.Location = new System.Drawing.Point(448, 358);

this.forSisII30.Name = "forSisII30";

this.forSisII30.Size = new System.Drawing.Size(83, 83);

this.forSisII30.TabIndex = 1;

this.forSisII30.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII29

//

this.forSisII29.Location = new System.Drawing.Point(359, 358);

this.forSisII29.Name = "forSisII29";

this.forSisII29.Size = new System.Drawing.Size(83, 83);

this.forSisII29.TabIndex = 1;

this.forSisII29.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII28

//

this.forSisII28.Location = new System.Drawing.Point(270, 358);

this.forSisII28.Name = "forSisII28";

this.forSisII28.Size = new System.Drawing.Size(83, 83);

this.forSisII28.TabIndex = 1;

this.forSisII28.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII27

//

this.forSisII27.Location = new System.Drawing.Point(181, 358);

this.forSisII27.Name = "forSisII27";

this.forSisII27.Size = new System.Drawing.Size(83, 83);

this.forSisII27.TabIndex = 1;

this.forSisII27.Values = new double[] {

0.33178756028962675D,

0.6082145839967833D,

0.36348531924350436D,

0.765521412140467D,

0.1140112327942677D,

0.077116242180166883D,

0.091800876004528659D,

0.99558315006810383D,

0.29988437672140278D,

0.36317226447312734D,

0.78250905954395844D,

0.27873250482544887D,

0.64119600208531879D,

0.0014147958724828418D,

0.52935632156643841D,

0.39265398652882033D,

0.82702095146617893D,

0.64167594054791888D,

0.10445988089985209D,

0.568676001191454D,

0.5786719106038436D,

0.069087485814973468D,

0.30045375335051389D,

0.031386088128847113D,

0.0997950374613493D,

0.61099255066876879D,

0.11189087252686307D,

0.78541736760428982D,

0.70368282250300185D,

0.57122083919645328D,

0.79018988869627471D,

0.8482957900726682D,

0.9944998770926613D,

0.3570318717309422D,

0.82665206111346D,

0.97637193462642469D};

//

// forSisII26

//

this.forSisII26.Location = new System.Drawing.Point(92, 358);

this.forSisII26.Name = "forSisII26";

this.forSisII26.Size = new System.Drawing.Size(83, 83);

this.forSisII26.TabIndex = 1;

this.forSisII26.Values = new double[] {

0.030630804146933744D,

0.59983983989797529D,

0.85516131010612539D,

0.28582640284943694D,

0.63645630825145927D,

0.71228530430807047D,

0.71532702572426154D,

0.41830261024567422D,

0.35560618543792805D,

0.54895327545188055D,

0.69155330941619042D,

0.88530132401981454D,

0.48095011593818204D,

0.50501657719957482D,

0.99142113141316046D,

0.61081074812021607D,

0.88437328295985851D,

0.10696957684446572D,

0.53739905894612849D,

0.095145604617495841D,

0.94500066849635944D,

0.10701355203381439D,

0.64212648274476014D,

0.47465358044703193D,

0.31639747802000373D,

0.29411302706883896D,

0.0913197310135326D,

0.24855489248808235D,

0.15671738104741897D,

0.66912166526034556D,

0.86597344785275565D,

0.35234643488765993D,

0.20207364633775951D,

0.60034810686500184D,

0.748585246386279D,

0.69290315904324085D};

//

// forSisII25

//

this.forSisII25.Location = new System.Drawing.Point(3, 358);

this.forSisII25.Name = "forSisII25";

this.forSisII25.Size = new System.Drawing.Size(83, 83);

this.forSisII25.TabIndex = 1;

this.forSisII25.Values = new double[] {

0.10174499736248749D,

0.14172802080480756D,

0.53863698502007729D,

0.5889561491035652D,

0.14718602371736711D,

0.47326707256644363D,

0.96881003396995835D,

0.886750604438945D,

0.76593287324809134D,

0.88426247839083083D,

0.41464522965934369D,

0.58701040716236941D,

0.13422517391584124D,

0.8716158321460783D,

0.18816828037992506D,

0.93146333654013613D,

0.56332817047989381D,

0.57373712471394667D,

0.08823280552785509D,

0.76864029596030725D,

0.95903972301587448D,

0.949060135497274D,

0.84895727543577426D,

0.34943150232985221D,

0.77836776095366467D,

0.33119670596495115D,

0.35915390418803034D,

0.33355547642966521D,

0.13354153751094897D,

0.45164159985801278D,

0.4999481921549645D,

0.37614549760527233D,

0.092598688831831644D,

0.995353570671451D,

0.70986684817348922D,

0.55070570788844753D};

//

// forSisII21

//

this.forSisII21.Location = new System.Drawing.Point(181, 269);

this.forSisII21.Name = "forSisII21";

this.forSisII21.Size = new System.Drawing.Size(83, 83);

this.forSisII21.TabIndex = 0;

this.forSisII21.Values = new double[] {

0.49527310556512005D,

0.38349496078840223D,

0.77734852106186958D,

0.47678007347359325D,

0.46067751546421903D,

0.035599667129851721D,

0.25269243784839868D,

0.020351380119263837D,

0.15350857151369962D,

0.28887762422155477D,

0.548519163182247D,

0.188003528019415D,

0.46981522183391045D,

0.63654346514332272D,

0.39410643949830271D,

0.60456699766431332D,

0.37952760997206325D,

0.54800680072419661D,

0.081449253056873214D,

0.027532596153920792D,

0.32216856643658531D,

0.34552892173897892D,

0.52835623618604444D,

0.11590381949949256D,

0.16733835459097213D,

0.14612416417623134D,

0.26602139941697073D,

0.93426865708747353D,

0.99011865630285756D,

0.26739906811499925D,

0.44025367518898739D,

0.49699713825108349D,

0.28531352583566377D,

0.90936762462806309D,

0.36259514762209505D,

0.36127020295768519D};

//

// forSisII14

//

this.forSisII14.Location = new System.Drawing.Point(92, 180);

this.forSisII14.Name = "forSisII14";

this.forSisII14.Size = new System.Drawing.Size(83, 83);

this.forSisII14.TabIndex = 0;

this.forSisII14.Values = new double[] {

0.77983346524640618D,

0.781154889045821D,

0.63644801109863813D,

0.30004559750670828D,

0.1124151675554063D,

0.80562415523716446D,

0.240419836826818D,

0.049859585729362249D,

0.71091335812160439D,

0.46064878928505293D,

0.11258858121586432D,

0.8724238564597554D,

0.23492178331824104D,

0.49907613429197861D,

0.834819307939531D,

0.47607096679325728D,

0.18732571889987482D,

0.88346463622686666D,

0.33714831915551252D,

0.14974418382614116D,

0.85513872460235785D,

0.661444701562377D,

0.30944171888262112D,

0.31455335128798773D,

0.769578267712881D,

0.31842378076092515D,

0.030232723816406321D,

0.88610950339870043D,

0.84862394903256744D,

0.051522024931163542D,

0.90269000218375117D,

0.56044759720584736D,

0.27394831752122767D,

0.032836236540617535D,

0.82473353241790714D,

0.59837272092624227D};

//

// forSisII7

//

this.forSisII7.Location = new System.Drawing.Point(3, 91);

this.forSisII7.Name = "forSisII7";

this.forSisII7.Size = new System.Drawing.Size(83, 83);

this.forSisII7.TabIndex = 0;

this.forSisII7.Values = new double[] {

0.91155878962555847D,

0.78140365555016489D,

0.91511742859851453D,

0.51654441119942085D,

0.10510908491216092D,

0.47849665045668216D,

0.40499046137788819D,

0.7066895653059192D,

0.033535582960367009D,

0.39359754295721533D,

0.69934687376923244D,

0.49104252201087889D,

0.43257976017546829D,

0.96418270839573006D,

0.30967141935120868D,

0.40184888960879711D,

0.57221930547255062D,

0.21285096379595389D,

0.50613147835532735D,

0.11838876368402912D,

0.47171317016319986D,

0.32189465981065046D,

0.53046732979382727D,

0.79399138679448111D,

0.77539143142075806D,

0.35431033342811757D,

0.20123563669679484D,

0.65939133319044085D,

0.55795878710130176D,

0.552140984010017D,

0.59847553893852767D,

0.2020855467776235D,

0.67434260187407147D,

0.23644658701328866D,

0.56451217716769886D,

0.40592206334970987D};

//

// forSisII20

//

this.forSisII20.Location = new System.Drawing.Point(92, 269);

this.forSisII20.Name = "forSisII20";

this.forSisII20.Size = new System.Drawing.Size(83, 83);

this.forSisII20.TabIndex = 0;

this.forSisII20.Values = new double[] {

0.84441024523433772D,

0.25769853417654454D,

0.54878723833187826D,

0.09644972071817598D,

0.9462115620012449D,

0.50495175016343208D,

0.85001816127915786D,

0.9604666176999298D,

0.82805840057696145D,

0.09161310367826983D,

0.91748190667362972D,

0.25774381880543373D,

0.82188053839927566D,

0.84870208699661409D,

0.08634613691193338D,

0.42482248387524041D,

0.74876361840812655D,

0.58628051615612609D,

0.67216446933903007D,

0.87437951745203679D,

0.030108343823863355D,

0.35107941150249888D,

0.4149396412144134D,

0.24125372210575907D,

0.57362788430118372D,

0.053098899802704762D,

0.51371094794651073D,

0.643384020143833D,

0.83202368665115145D,

0.079439478032029928D,

0.98977824393183844D,

0.32729979805988252D,

0.64449669636995377D,

0.27816543461669491D,

0.12358758790585565D,

0.85208850207370168D};

//

// forSisII13

//

this.forSisII13.Location = new System.Drawing.Point(3, 180);

this.forSisII13.Name = "forSisII13";

this.forSisII13.Size = new System.Drawing.Size(83, 83);

this.forSisII13.TabIndex = 0;

this.forSisII13.Values = new double[] {

0.42606080902091265D,

0.058536016875196259D,

0.90100960382307393D,

0.57449917801399675D,

0.44381818615077911D,

0.13891444268585856D,

0.68279671142008D,

0.36240827169381468D,

0.882141262703641D,

0.32311490193154424D,

0.29993504160080803D,

0.87378819467210589D,

0.17105043314911911D,

0.059948225533565609D,

0.08933763117032946D,

0.48105891443838317D,

0.56444842487780766D,

0.018501971391263406D,

0.3671070101517751D,

0.49333083373183889D,

0.84275889296213113D,

0.30753351622611913D,

0.52282183641699231D,

0.89311266219854013D,

0.99124080221692135D,

0.2066341411353248D,

0.12844293104877833D,

0.74861892254493145D,

0.1018348597464314D,

0.5901821086137472D,

0.61324712243548D,

0.5437151037825807D,

0.72511449909075842D,

0.23425763483823167D,

0.99876167485432776D,

0.69022760153293028D};

//

// forSisII6

//

this.forSisII6.Location = new System.Drawing.Point(448, 2);

this.forSisII6.Name = "forSisII6";

this.forSisII6.Size = new System.Drawing.Size(83, 83);

this.forSisII6.TabIndex = 0;

this.forSisII6.Values = new double[] {

0.68527091233305215D,

0.097665291325033307D,

0.43056133130125762D,

0.89119390113800478D,

0.66828270008241886D,

0.052531369520598732D,

0.35911245940211811D,

0.25051298236963943D,

0.75316385820189669D,

0.63992275327440484D,

0.47233358699471389D,

0.77207546111758585D,

0.99010184686170044D,

0.50871932949345533D,

0.76285137318207485D,

0.62599472497869035D,

0.9110584244649198D,

0.63025675417401672D,

0.0093296091115705714D,

0.11852729279479352D,

0.535714180458204D,

0.0078969416245338233D,

0.5832583757970754D,

0.23718404501545432D,

0.010471511637080233D,

0.93417778887514857D,

0.26057341194738326D,

0.29195915408989376D,

0.70986777949606428D,

0.75621993781822727D,

0.14867369558134755D,

0.93679279831088746D,

0.061186550679237839D,

0.39911002963739917D,

0.79578800210533107D,

0.4667831335527744D};

//

// forSisII19

//

this.forSisII19.Location = new System.Drawing.Point(3, 269);

this.forSisII19.Name = "forSisII19";

this.forSisII19.Size = new System.Drawing.Size(83, 83);

this.forSisII19.TabIndex = 0;

this.forSisII19.Values = new double[] {

0.71012814608874175D,

0.89110809466387519D,

0.80942583308062788D,

0.28246658867339908D,

0.6904937954109599D,

0.22661834779503678D,

0.55844419801535283D,

0.12450007587880831D,

0.60252488665400306D,

0.81802982083430043D,

0.873380661883103D,

0.43743310702845134D,

0.82367601609960017D,

0.52993674181864447D,

0.34639520866162854D,

0.59451729599131142D,

0.343692021604484D,

0.64349073108448218D,

0.83995459593830379D,

0.71426539808244693D,

0.22083251980172122D,

0.28883848445901578D,

0.53174167523707339D,

0.72494325401491633D,

0.3718713356050995D,

0.30412372541805904D,

0.0086254822130433662D,

0.0081354547329877749D,

0.78727367882955523D,

0.93336592285584929D,

0.77341706947116973D,

0.92605189463405491D,

0.764166826272461D,

0.70603883252760336D,

0.47082402253096178D,

0.3534885138987976D};

//

// forSisII12

//

this.forSisII12.Location = new System.Drawing.Point(448, 91);

this.forSisII12.Name = "forSisII12";

this.forSisII12.Size = new System.Drawing.Size(83, 83);

this.forSisII12.TabIndex = 0;

this.forSisII12.Values = new double[] {

0.3089881983161849D,

0.6699096293514174D,

0.70406955839324259D,

0.36857237171780893D,

0.532499024426797D,

0.87376948998950865D,

0.27874286485777372D,

0.91363383033947732D,

0.269624221729871D,

0.422635884221008D,

0.3750837493571843D,

0.11554181767419997D,

0.8211300497973012D,

0.28566690687354046D,

0.18757047419788803D,

0.30570758148362281D,

0.35755676792820767D,

0.57538372630969792D,

0.78499342025490171D,

0.42128966162972603D,

0.35936641942680181D,

0.084482579065711508D,

0.91059525306829958D,

0.38637006999290086D,

0.21799286558199341D,

0.550308743282365D,

0.33722639704925306D,

0.66915896379815365D,

0.044612751363130636D,

0.94732876724904813D,

0.67326628075599027D,

0.11763718357199672D,

0.059112719287682657D,

0.992906694762831D,

0.28552909767512658D,

0.67378239225306658D};

//

// forSisII5

//

this.forSisII5.Location = new System.Drawing.Point(359, 2);

this.forSisII5.Name = "forSisII5";

this.forSisII5.Size = new System.Drawing.Size(83, 83);

this.forSisII5.TabIndex = 0;

this.forSisII5.Values = new double[] {

0.93942117269123959D,

0.47138222422049486D,

0.18176637365565002D,

0.3470630298122126D,

0.010095619601242067D,

0.61810784489759607D,

0.45531903228504539D,

0.94923545138409149D,

0.92903997606087474D,

0.89308366453884336D,

0.18700540493568657D,

0.50160677195601477D,

0.74579544865796132D,

0.46770948798754697D,

0.56849512670584723D,

0.18878309996276307D,

0.92104541227270165D,

0.049534360901235773D,

0.99412209447199573D,

0.22450561925047338D,

0.75931437628311771D,

0.31769948840034168D,

0.15057950613581553D,

0.982190281144432D,

0.53654309294025559D,

0.60958390105962D,

0.86902107897634673D,

0.32229545448082286D,

0.74936960346501769D,

0.25744656578518754D,

0.056429098386517308D,

0.82822335503447031D,

0.00013780919841388669D,

0.5137880819448214D,

0.36628640879238322D,

0.88617454370771276D};

//

// forSisII18

//

this.forSisII18.Location = new System.Drawing.Point(448, 180);

this.forSisII18.Name = "forSisII18";

this.forSisII18.Size = new System.Drawing.Size(83, 83);

this.forSisII18.TabIndex = 0;

this.forSisII18.Values = new double[] {

0.39361735265404796D,

0.43793039044268911D,

0.411194042028484D,

0.74125857452920574D,

0.62916354678066611D,

0.96135980168420809D,

0.45733009393202612D,

0.32490920104315D,

0.36330333834667844D,

0.83561962509323828D,

0.92336351514019233D,

0.57690326337558373D,

0.37883364054320084D,

0.48448318079322722D,

0.19659177129929503D,

0.0095783583864468887D,

0.99878460029083516D,

0.061023478424653166D,

0.91446801596994887D,

0.62172168429089791D,

0.32080271808467931D,

0.19957609251121808D,

0.810519936871957D,

0.40051171854162204D,

0.74908676592124945D,

0.1330235778042225D,

0.1998658479190738D,

0.6715934102756872D,

0.83665456615232614D,

0.35878204990121632D,

0.50146896275760089D,

0.23200736671313987D,

0.10142307919516372D,

0.68232058299813447D,

0.79516574730871514D,

0.48311502183001259D};

//

// forSisII11

//

this.forSisII11.Location = new System.Drawing.Point(359, 91);

this.forSisII11.Name = "forSisII11";

this.forSisII11.Size = new System.Drawing.Size(83, 83);

this.forSisII11.TabIndex = 0;

this.forSisII11.Values = new double[] {

0.46568140595484592D,

0.11651344137103922D,

0.7613522898225823D,

0.49513241299201383D,

0.58068914971346464D,

0.10959646902493968D,

0.71429133262219435D,

0.4569689503204864D,

0.18956176386660048D,

0.93497942524728339D,

0.6107386157897946D,

0.78980382615225564D,

0.52934329515758127D,

0.56235463198384017D,

0.63268730073826729D,

0.35685835469367838D,

0.11694875737510098D,

0.60112448716588529D,

0.85570456825928043D,

0.62765799538588984D,

0.26069171133483376D,

0.099202103493363641D,

0.036460620833775319D,

0.074632710811976671D,

0.3374156702018416D,

0.61436565435229129D,

0.36677280551138D,

0.99486423749237518D,

0.816637631420343D,

0.24694875639255567D,

0.57556898918727828D,

0.34737804501661007D,

0.67561270048637534D,

0.083517674395589936D,

0.4734329248189148D,

0.48280662460383339D};

//

// forSisII4

//

this.forSisII4.Location = new System.Drawing.Point(270, 2);

this.forSisII4.Name = "forSisII4";

this.forSisII4.Size = new System.Drawing.Size(83, 83);

this.forSisII4.TabIndex = 0;

this.forSisII4.Values = new double[] {

0.98777237673652007D,

0.337890583713488D,

0.33126700452122232D,

0.72548885304736388D,

0.79262389838352054D,

0.2099460923159244D,

0.89963464294543238D,

0.63896486146327336D,

0.13918766106441041D,

0.5949767751595828D,

0.93619538980359D,

0.69801580193360135D,

0.66738445622259024D,

0.48737604286865149D,

0.48960586380660809D,

0.93203265635857013D,

0.59828830538237854D,

0.10695362235743255D,

0.40275380406656947D,

0.29649285566829742D,

0.9125226698408474D,

0.647028615068192D,

0.77499032103223275D,

0.57696389573484841D,

0.89487153938732644D,

0.36396519763579832D,

0.46710409245784584D,

0.91163749709335973D,

0.58425968307268794D,

0.0293523725258896D,

0.6828743231868718D,

0.90878817155435132D,

0.74497590947196624D,

0.53285495542588412D,

0.10042056958210681D,

0.519881597496514D};

//

// forSisII24

//

this.forSisII24.Location = new System.Drawing.Point(448, 269);

this.forSisII24.Name = "forSisII24";

this.forSisII24.Size = new System.Drawing.Size(83, 83);

this.forSisII24.TabIndex = 0;

this.forSisII24.Values = new double[] {

0.41231303960658283D,

0.75974051596584757D,

0.57385540407796176D,

0.8701067282213395D,

0.78708493606517316D,

0.51875188132689887D,

0.82713583429676285D,

0.14784688835397683D,

0.78323170532622921D,

0.942315516966542D,

0.27522855125145451D,

0.87203439319135356D,

0.77375470696657644D,

0.18215846930731017D,

0.84753731537961274D,

0.98196808480749287D,

0.74712687206786443D,

0.077153064346477881D,

0.13418667490323386D,

0.34074376166832809D,

0.56290026268125526D,

0.754303108786374D,

0.83061731366003744D,

0.42865870074772217D,

0.74284199985807853D,

0.98799714585207266D,

0.054705178390585432D,

0.10983528853852083D,

0.91210245197271111D,

0.027407218249238665D,

0.95303989246163512D,

0.13452950079670617D,

0.38695547328654467D,

0.969724266310094D,

0.5197196167519873D,

0.838547789416531D};

//

// forSisII17

//

this.forSisII17.Location = new System.Drawing.Point(359, 180);

this.forSisII17.Name = "forSisII17";

this.forSisII17.Size = new System.Drawing.Size(83, 83);

this.forSisII17.TabIndex = 0;

this.forSisII17.Values = new double[] {

0.53309821827947079D,

0.53264707584523D,

0.50940791960312426D,

0.39377078851394859D,

0.81989278077142913D,

0.6271434326782559D,

0.7937321904086192D,

0.95255602242078452D,

0.088736646384343809D,

0.59506969926649222D,

0.1378827901267832D,

0.40210121376537772D,

0.18181505714627683D,

0.70090623837937893D,

0.16166129948648686D,

0.6678228451254884D,

0.39866828052265024D,

0.75967680791377867D,

0.95698133481525882D,

0.658009930820209D,

0.92912320230581014D,

0.1451967033302396D,

0.12726472836326097D,

0.7990877902131005D,

0.46404670293631345D,

0.71730054575824209D,

0.23574443638126572D,

0.75582448707699057D,

0.98927562450490691D,

0.14069905045474834D,

0.48507891990480895D,

0.80403044065648244D,

0.66243885255532287D,

0.0089895259630817584D,

0.448869866528022D,

0.21447979622263452D};

//

// forSisII10

//

this.forSisII10.Location = new System.Drawing.Point(270, 91);

this.forSisII10.Name = "forSisII10";

this.forSisII10.Size = new System.Drawing.Size(83, 83);

this.forSisII10.TabIndex = 0;

this.forSisII10.Values = new double[] {

0.23882582701687971D,

0.021224504812259463D,

0.69743901244245421D,

0.05106191106655724D,

0.8052746489668613D,

0.97949242777167467D,

0.948973434487811D,

0.8114427383111058D,

0.7892803143659981D,

0.024548222787933529D,

0.8900785683142387D,

0.70405857716876019D,

0.40483783018069242D,

0.857800254532043D,

0.79128750916164714D,

0.80072916662354454D,

0.09956493885236091D,

0.060968351578790859D,

0.80909744874066558D,

0.85742501535332993D,

0.81069180081165015D,

0.418258745883712D,

0.5443578337060091D,

0.922663215046126D,

0.91478089332337531D,

0.39188445843378289D,

0.037781521229902991D,

0.51626200625498875D,

0.91294160341515285D,

0.75519470020904889D,

0.95915984174197533D,

0.91856441549889944D,

0.185139346954012D,

0.077608826140690979D,

0.38845283835588623D,

0.99151842668257584D};

//

// forSisII3

//

this.forSisII3.Location = new System.Drawing.Point(181, 2);

this.forSisII3.Name = "forSisII3";

this.forSisII3.Size = new System.Drawing.Size(83, 83);

this.forSisII3.TabIndex = 0;

this.forSisII3.Values = new double[] {

0.47834148559642092D,

0.38320752903037125D,

0.096078320451117263D,

0.69241045354512076D,

0.10494682523652298D,

0.17750706019695245D,

0.77483408002873611D,

0.6555590725762579D,

0.84868469268487989D,

0.78559099640026264D,

0.97369716594633515D,

0.97400840882864237D,

0.68193410787821473D,

0.76809182146940935D,

0.78048036097571272D,

0.94248666192520725D,

0.23004823281897616D,

0.5678857572227185D,

0.51465175557632548D,

0.38949487609299593D,

0.7220816583009817D,

0.16923667544929155D,

0.48417901316852263D,

0.32863526666939036D,

0.21978021237057643D,

0.27922630649023983D,

0.30146239246309847D,

0.17631498220205075D,

0.74030259053236924D,

0.21597076170890162D,

0.7828431086534835D,

0.74279575782958218D,

0.30247470145229005D,

0.83205832952263692D,

0.71272653467614511D,

0.75487652549281559D};

//

// forSisII23

//

this.forSisII23.Location = new System.Drawing.Point(359, 269);

this.forSisII23.Name = "forSisII23";

this.forSisII23.Size = new System.Drawing.Size(83, 83);

this.forSisII23.TabIndex = 0;

this.forSisII23.Values = new double[] {

0.14657436457768752D,

0.15515013977659406D,

0.7156202558966448D,

0.29935961090929786D,

0.99994171736759219D,

0.480721945166924D,

0.56470549784819857D,

0.15611991014150897D,

0.45901426228648717D,

0.92449272606731059D,

0.35215150627873443D,

0.29729655957654888D,

0.19103379183962652D,

0.57643256828954093D,

0.097180771221025281D,

0.90146165895343833D,

0.28511732690274588D,

0.24696361704122444D,

0.51912060450721564D,

0.30910481014712937D,

0.89902851586184862D,

0.76744305378172684D,

0.47263024117454433D,

0.82572051874628316D,

0.876044667733854D,

0.59851909782668533D,

0.91525648204388865D,

0.63271393097597828D,

0.0027478877467791958D,

0.23090140811675294D,

0.67153370737635243D,

0.84987577835557782D,

0.055365286793264232D,

0.025603835482897162D,

0.79591229734751967D,

0.074898093042382083D};

//

// forSisII16

//

this.forSisII16.Location = new System.Drawing.Point(270, 180);

this.forSisII16.Name = "forSisII16";

this.forSisII16.Size = new System.Drawing.Size(83, 83);

this.forSisII16.TabIndex = 0;

this.forSisII16.Values = new double[] {

0.8522655013260737D,

0.21529214466702759D,

0.3895531587254038D,

0.24135971313405769D,

0.60453117760109309D,

0.32805910302701363D,

0.86962100438290324D,

0.2952874863032659D,

0.92707480021150546D,

0.0041658328865495659D,

0.98528119036242423D,

0.16387002224282829D,

0.11878999048787635D,

0.88138144970004517D,

0.4576784309268363D,

0.0555110844110656D,

0.31293772501542128D,

0.40362172452901568D,

0.855848009630967D,

0.37913131079596063D,

0.68251989860204976D,

0.88989973715036164D,

0.42331494317544388D,

0.4014226195409068D,

0.56546546312303536D,

0.93199156687222029D,

0.1533720223947298D,

0.22811285416973423D,

0.25295901869095816D,

0.50227572792315656D,

0.24193127278328466D,

0.16542995635672936D,

0.78052027094202125D,

0.022282678178643191D,

0.049196157627364692D,

0.069825182235718319D};

//

// forSisII9

//

this.forSisII9.Location = new System.Drawing.Point(181, 91);

this.forSisII9.Name = "forSisII9";

this.forSisII9.Size = new System.Drawing.Size(83, 83);

this.forSisII9.TabIndex = 0;

this.forSisII9.Values = new double[] {

0.69904809198297935D,

0.284878596796132D,

0.481309261397137D,

0.50517714373077138D,

0.99125401861558393D,

0.5514361083281395D,

0.78818471207664564D,

0.814079003321975D,

0.41806762638411837D,

0.032005584347995733D,

0.45488987697981759D,

0.6369363957256714D,

0.965337674583D,

0.59018771051903618D,

0.99567413050479914D,

0.36373928439046221D,

0.62990448373831087D,

0.086830393451652679D,

0.27784798074413464D,

0.86927991913132363D,

0.60186664136213563D,

0.032034021817163576D,

0.3962823322025511D,

0.25631389918565467D,

0.13501428725897069D,

0.47384664624642892D,

0.10964427474403952D,

0.79338667345856628D,

0.36803670850025338D,

0.41741662817886871D,

0.28918114597405359D,

0.38536972849879869D,

0.41792199687004183D,

0.50176121969789322D,

0.95269547028126911D,

0.88487722905579824D};

//

// forSisII2

//

this.forSisII2.Location = new System.Drawing.Point(92, 2);

this.forSisII2.Name = "forSisII2";

this.forSisII2.Size = new System.Drawing.Size(83, 83);

this.forSisII2.TabIndex = 0;

this.forSisII2.Values = new double[] {

0.11571864789152456D,

0.037306887580690389D,

0.66933409202347238D,

0.74005139467308834D,

0.77756203747240926D,

0.253600835918263D,

0.069695625952303233D,

0.72962221537233429D,

0.014385248541080043D,

0.93759742515981082D,

0.9692095699576706D,

0.34842624158990859D,

0.95081379960794643D,

0.54382624735302587D,

0.27536446055181535D,

0.089101289906120526D,

0.068451027417765473D,

0.28568140011545334D,

0.99628096446221737D,

0.71047636294293981D,

0.51703819423776032D,

0.45370437319097312D,

0.79874925957934428D,

0.20514778848977192D,

0.41577295186732566D,

0.25382098613950471D,

0.399857057910346D,

0.11709812475233251D,

0.18637767303100677D,

0.99588494282024209D,

0.53455460934646182D,

0.71338020205189479D,

0.59134290767430464D,

0.72389939554217242D,

0.62963625072950324D,

0.072698380366292961D};

//

// forSisII22

//

this.forSisII22.Location = new System.Drawing.Point(270, 269);

this.forSisII22.Name = "forSisII22";

this.forSisII22.Size = new System.Drawing.Size(83, 83);

this.forSisII22.TabIndex = 0;

this.forSisII22.Values = new double[] {

0.65948661494044891D,

0.69325879760703946D,

0.88945203269340656D,

0.63499542867531789D,

0.15529973625917906D,

0.64054064109946629D,

0.503022123828075D,

0.5041944084242892D,

0.025588468660408849D,

0.097616053231813035D,

0.04983719859730322D,

0.19899819102091632D,

0.79415568792920355D,

0.12991310894950903D,

0.39512542746734125D,

0.35181084012231362D,

0.11451593698678349D,

0.71779119629310029D,

0.22566523879098949D,

0.66201427470055141D,

0.23855762800134608D,

0.46418630353370044D,

0.32427844280576262D,

0.52374105133290449D,

0.853743778007917D,

0.95259750119997078D,

0.54324454234132757D,

0.018500305720837929D,

0.403042815813349D,

0.25582936790577571D,

0.757083333915604D,

0.226914404065774D,

0.91418999662352263D,

0.20266608018552237D,

0.42961467496567157D,

0.37519222981072603D};

//

// forSisII15

//

this.forSisII15.Location = new System.Drawing.Point(181, 180);

this.forSisII15.Name = "forSisII15";

this.forSisII15.Size = new System.Drawing.Size(83, 83);

this.forSisII15.TabIndex = 0;

this.forSisII15.Values = new double[] {

0.39622936742204679D,

0.36128553578689954D,

0.55517662668376078D,

0.876497553138294D,

0.95068224936289814D,

0.29455485115505514D,

0.17955931982936305D,

0.31815689863551266D,

0.20398378754220148D,

0.20085886688942969D,

0.32294243682312895D,

0.056464564081497755D,

0.60075951535290084D,

0.18274376922414815D,

0.59886426506511137D,

0.87355171138120424D,

0.49823415675118293D,

0.96762197602895184D,

0.83414075236494689D,

0.19530031140674853D,

0.36898035480127689D,

0.36571098136050206D,

0.78125165066740088D,

0.20270223505920834D,

0.097296098758138758D,

0.48452181810723705D,

0.10115159261094014D,

0.76975910075463316D,

0.34053271931620904D,

0.82292279453152917D,

0.28480819439739369D,

0.5914896077436812D,

0.70029843398383751D,

0.019933197656615262D,

0.95558147270026683D,

0.75323040119988394D};

//

// forSisII8

//

this.forSisII8.Location = new System.Drawing.Point(92, 91);

this.forSisII8.Name = "forSisII8";

this.forSisII8.Size = new System.Drawing.Size(83, 83);

this.forSisII8.TabIndex = 0;

this.forSisII8.Values = new double[] {

0.16261456960933962D,

0.34916768565269546D,

0.71133202533765327D,

0.944002776846291D,

0.28462698370433737D,

0.0061215441702499726D,

0.319757263790703D,

0.65288491111848734D,

0.62965506437684182D,

0.48677997825982977D,

0.4177407903679371D,

0.22029904658920088D,

0.65696510284066434D,

0.88353162253439965D,

0.728680247314591D,

0.94656802059457079D,

0.36852532782057551D,

0.23431436355892307D,

0.0062118750094491406D,

0.030518386061544708D,

0.5800338851194986D,

0.35247439162455241D,

0.77920145438015531D,

0.46616043125566115D,

0.193403258544115D,

0.40980021907472991D,

0.97762417931930357D,

0.38106099301067226D,

0.916050672492036D,

0.73145282907944775D,

0.35616613009766029D,

0.58082631769628557D,

0.22716229652388129D,

0.84563386386522732D,

0.71093714177186462D,

0.14906647109848747D};

//

// forSisII1

//

this.forSisII1.Location = new System.Drawing.Point(3, 2);

this.forSisII1.Name = "forSisII1";

this.forSisII1.Size = new System.Drawing.Size(83, 83);

this.forSisII1.TabIndex = 0;

this.forSisII1.Values = new double[] {

0.25628995069129856D,

0.89013797551865592D,

0.91067332770241116D,

0.36285881063102687D,

0.045953717569799032D,

0.12836673954891356D,

0.57304717068236655D,

0.610516120498309D,

0.066781027739299934D,

0.88085254602173924D,

0.11279399791396876D,

0.45700109678180939D,

0.09424254721693813D,

0.3382401738028229D,

0.22296427992310575D,

0.46598407042491441D,

0.013721322647166124D,

0.92506308663872217D,

0.1731965160803853D,

0.81014017798478721D,

0.5699662312725402D,

0.2451715940819921D,

0.750599518302176D,

0.8748267646296074D,

0.028497364850946408D,

0.93869627078003082D,

0.73683423862645137D,

0.89820223529739407D,

0.13061384816216951D,

0.83691447267165153D,

0.99313675006531954D,

0.811331238975437D,

0.172594480762535D,

0.57961377621610355D,

0.69027806990327223D,

0.47838735230191953D};

//

// button3

//

this.button3.Location = new System.Drawing.Point(537, 387);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(172, 23);

this.button3.TabIndex = 9;

this.button3.Text = "Сбросить";

this.button3.UseVisualStyleBackColor = true;

this.button3.Click += new System.EventHandler(this.button3\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(716, 447);

this.Controls.Add(this.button3);

this.Controls.Add(this.progressBar1);

this.Controls.Add(this.label1);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.groupBox2);

this.Controls.Add(this.groupBox1);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.forSisII30);

this.Controls.Add(this.forSisII29);

this.Controls.Add(this.forSisII28);

this.Controls.Add(this.forSisII27);

this.Controls.Add(this.forSisII26);

this.Controls.Add(this.forSisII25);

this.Controls.Add(this.forSisII21);

this.Controls.Add(this.forSisII14);

this.Controls.Add(this.forSisII7);

this.Controls.Add(this.forSisII20);

this.Controls.Add(this.forSisII13);

this.Controls.Add(this.forSisII6);

this.Controls.Add(this.forSisII19);

this.Controls.Add(this.forSisII12);

this.Controls.Add(this.forSisII5);

this.Controls.Add(this.forSisII18);

this.Controls.Add(this.forSisII11);

this.Controls.Add(this.forSisII4);

this.Controls.Add(this.forSisII24);

this.Controls.Add(this.forSisII17);

this.Controls.Add(this.forSisII10);

this.Controls.Add(this.forSisII3);

this.Controls.Add(this.forSisII23);

this.Controls.Add(this.forSisII16);

this.Controls.Add(this.forSisII9);

this.Controls.Add(this.forSisII2);

this.Controls.Add(this.forSisII22);

this.Controls.Add(this.forSisII15);

this.Controls.Add(this.forSisII8);

this.Controls.Add(this.forSisII1);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;

this.MaximizeBox = false;

this.MinimizeBox = false;

this.Name = "Form1";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Сис. ИИ, 6/1";

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private ForSisII forSisII1;

private ForSisII forSisII2;

private ForSisII forSisII3;

private ForSisII forSisII4;

private ForSisII forSisII5;

private ForSisII forSisII6;

private ForSisII forSisII7;

private ForSisII forSisII8;

private ForSisII forSisII9;

private ForSisII forSisII10;

private ForSisII forSisII11;

private ForSisII forSisII12;

private ForSisII forSisII13;

private ForSisII forSisII14;

private ForSisII forSisII15;

private ForSisII forSisII16;

private ForSisII forSisII17;

private ForSisII forSisII18;

private ForSisII forSisII19;

private ForSisII forSisII20;

private ForSisII forSisII21;

private ForSisII forSisII22;

private ForSisII forSisII23;

private ForSisII forSisII24;

private ForSisII forSisII25;

private ForSisII forSisII26;

private ForSisII forSisII27;

private System.Windows.Forms.Button button1;

private ForSisII forSisII28;

private ForSisII forSisII29;

private ForSisII forSisII30;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.RadioButton radioButton4;

private System.Windows.Forms.RadioButton radioButton3;

private System.Windows.Forms.RadioButton radioButton2;

private System.Windows.Forms.RadioButton radioButton1;

private System.Windows.Forms.GroupBox groupBox2;

private System.Windows.Forms.RadioButton radioButton6;

private System.Windows.Forms.RadioButton radioButton5;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.ProgressBar progressBar1;

private System.Windows.Forms.Button button3;

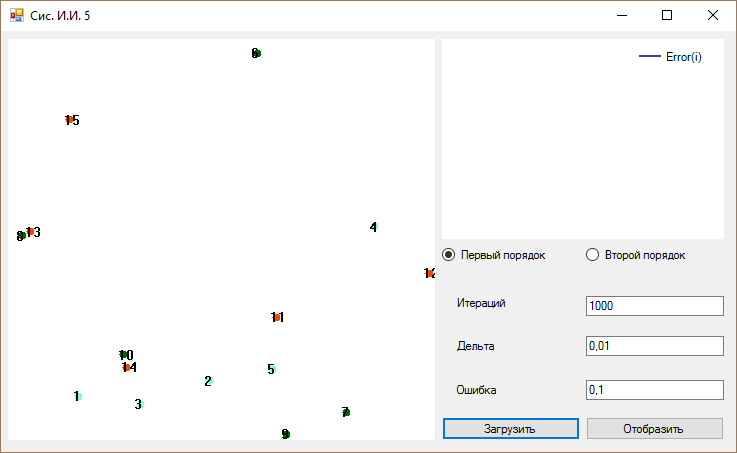
}

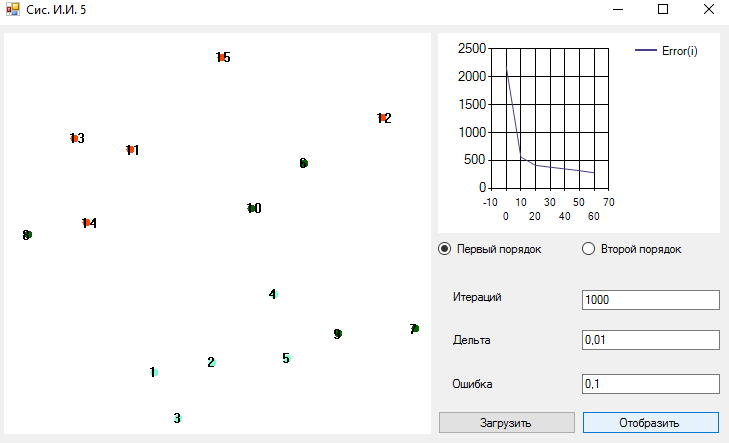
}

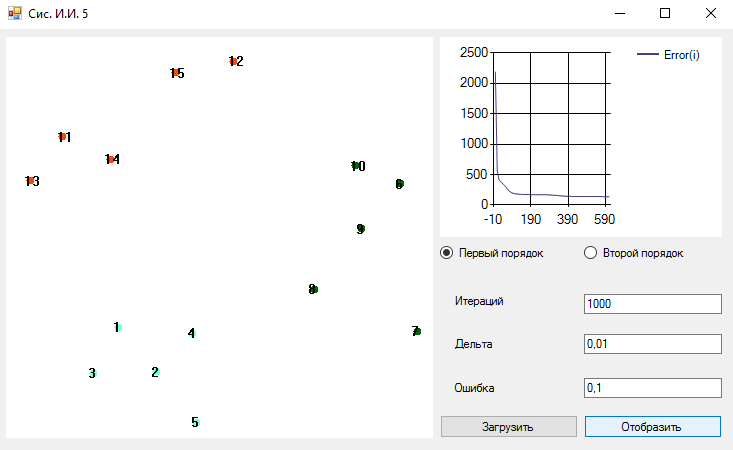
# 5. Визуализация многомерных образов. Алгоритм 1 и 2 порядка. Динамическая картина.

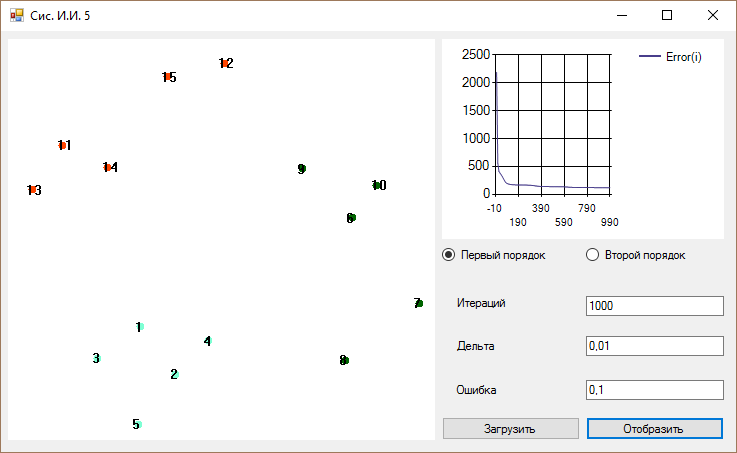
**Результаты работы программы:**

Первый порядок:

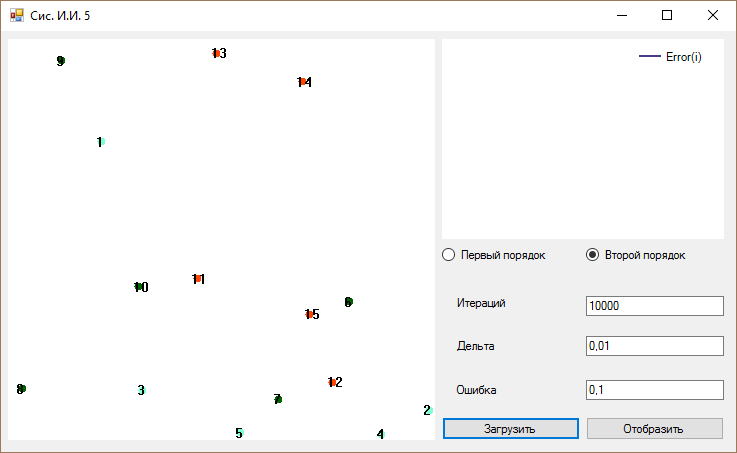


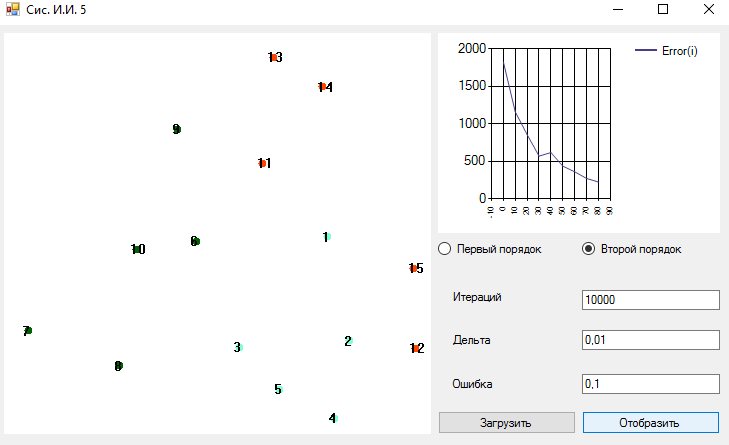


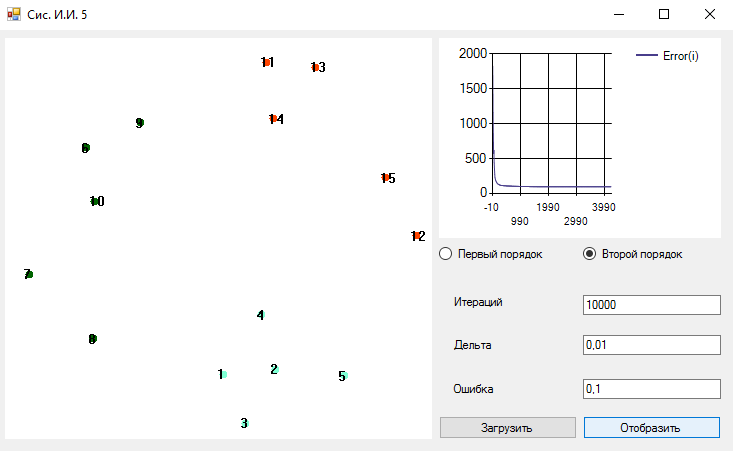




Второй порядок:







**Код программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

using System.Windows.Forms;

using Visn = System.Windows.Forms.DataVisualization;

namespace lab\_5.Classes

{

class \_MyVisualisation

{

#region fields

/// <summary>

/// Массив для внутреннего представления точек, тут все координаты маленькие, изначально

/// [0...1].

/// </summary>

Double[][] pointsArray;

/// <summary>

/// Массив точек для отображения. Внутренний массив проецируется на координаты

/// поля для отрисовки (нормируется и сдвигается)

/// </summary>

Double[][] pointsView;

/// <summary>

/// Порядок метода коррекции

/// </summary>

Int32 p = 1;

/// <summary>

/// Генератор случайных чисел

/// </summary>

Random randomGenerator = new Random();

#endregion

/// <summary>

/// Евклидова мера (расстояние между образами). Размерность должна совпадать

/// </summary>

/// <param name="x1">образ 1</param>

/// <param name="x2">образ 2</param>

/// <returns>Евклидово расстояние</returns>

Double EuqlidianMes(Double[] x1, Double[] x2)

{

Double result = 0;

for (int index = 0; index < x1.Length; index++)

{

result += Math.Pow(x1[index] - x2[index], 2);

}

return Math.Sqrt(result);

}

/// <summary>

/// Заполняет массивы координат случайными значениями из предложенного диапазона,

/// все точки попадают в очерченный прямоугольник

/// </summary>

/// <param name="minX"></param>

/// <param name="minY"></param>

/// <param name="maxX"></param>

/// <param name="maxY"></param>

/// <param name="pointsCount"></param>

public void GeneratePoints(Double minX, Double minY, Double maxX, Double maxY,int pointsCount)

{

Double xSize = maxX - minX;

Double ySize = maxY - minY;

pointsArray = new Double[pointsCount][];

pointsView = new Double[pointsCount][];

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

{

pointsArray[i] = new Double[2];

pointsView[i] = new Double[2];

//сгенерируем координаты точки случайно

pointsArray[i][0] = 5 \* randomGenerator.NextDouble() \* (randomGenerator.Next() % 100 < 60 ? 1 : -1);

pointsArray[i][1] = 5 \* randomGenerator.NextDouble() \* (randomGenerator.Next() % 100 < 60 ? 1 : -1);

//спроецируем их на плоскость отображения

//pointsView[i][0] = minX + pointsArray[i][0] \* xSize;

//pointsView[i][1] = minY + pointsArray[i][1] \* ySize;

}

this.Norm(minX, minY, maxX, maxY, pointsCount);

}

/// <summary>

/// Корректирует координаты методом первого порядка

/// </summary>

/// <param name="D">двумерный массив расстояний между образами 6х6</param>

/// <param name="d">двумерный массив расстояний между точками на плоскости</param>

/// <param name="sampleCount">размер выборки</param>

/// <param name="delta">шаг изменения координат</param>

/// <param name="i">номер точки, координаты которой правятся</param>

void CorrectI(Double[,] D,Double[,] d,int sampleCount,Double delta,int i)

{

Double dEdyi = .0;

Double dEdxi = .0;

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

{

//согласно формулам посчитаем производные

if (k != i)

{

dEdyi += 2 \* (d[i, k] - D[i, k]) \* (pointsArray[i][1] - pointsArray[k][1]) / (d[i, k] \* D[i, k]);

dEdxi += 2 \* (d[i, k] - D[i, k]) \* (pointsArray[i][0] - pointsArray[k][0]) / (d[i, k] \* D[i, k]);

}

}

//скорректируем координаты

pointsArray[i][0] -= delta \* dEdxi;

pointsArray[i][1] -= delta \* dEdyi;

}

void CorrectII(Double[,] D, Double[,] d, int sampleCount, Double delta, int i)

{

Double dEdyi = .0;

Double dEdxi = .0;

Double d2Edxi2 = .0, d2Edyi2 = .0;

Double s1, s2, s3;

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

{

//согласно формулам посчитаем производные

if (k != i)

{

double dx = (pointsArray[i][0] - pointsArray[k][0]);

double dy = (pointsArray[i][1] - pointsArray[k][1]);

s1 = 2 \* (d[i, k] - D[i, k]) / (d[i, k] \* D[i, k]);

s2 = 2 / (d[i, k] \* d[i, k] \* D[i, k]);

s3 = -s2 \* (d[i, k] - D[i, k]);

dEdxi += s1 \* dx;

dEdyi += s1 \* dy;

d2Edxi2 += s1;

d2Edxi2 += s2 \* dx \* dx;

d2Edxi2 += s3 \* dx \* dx;

d2Edyi2 += s1;

d2Edyi2 += s2 \* dy \* dy;

d2Edyi2 += s3 \* dy \* dy;

}

}

//скорректируем координаты

pointsArray[i][0] -= delta \* dEdxi / Math.Abs(d2Edxi2);

pointsArray[i][1] -= delta \* dEdyi / Math.Abs(d2Edyi2);

}

/// <summary>

/// Основной метод обучения

/// </summary>

/// <param name="sample">Выборка</param>

/// <param name="sampleCount">Размер выборки</param>

/// <param name="Epochs">Число итераций</param>

/// <param name="delta">Шаг изменения координат</param>

/// <param name="e0">Пороговое значение ошибки</param>

public void Run(Double[][] sample,Int32 sampleCount,Int32 Epochs,Double delta, Double e0

, Visn.Charting.Chart chart, Int32 displayStep)

{

Double epochError;

chart.Series[0].Points.Clear();

//p = 2;

Double[,] D;

Double[,] d;

D = new Double[sampleCount, sampleCount];

d = new Double[sampleCount, sampleCount];

//цикл по всем итерациям

for (int epoch = 0; epoch < Epochs; epoch++)

{

//накопленная ошибка итерации зануляется

epochError = .0;

//далее цикл по всем точкам

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

{

//вклад данной точки в общую ошибку пока что равен нулю

Double Ei = .0;

//по всем точкам

for (int j = 0; j < sampleCount; j++)

{

//расстояние до самой себя - 0, на ошибку не влияет

if (i != j)

{

//считаем расстояние между образами

D[i, j] = EuqlidianMes(sample[i], sample[j]);

//соответствующее расстояние между точками

d[i, j] = EuqlidianMes(pointsArray[i], pointsArray[j]);

//вклад в ошибку - нормированный

Ei += Math.Pow(D[i, j] - d[i, j], 2);// / D[i, j];

}

}

epochError += Ei;

if(p==1)

CorrectI(D, d, sampleCount, delta, i);

else

CorrectII(D, d, sampleCount, delta, i);

}

if (epoch % displayStep == 0)

{

chart.Series[0].Points.AddXY(epoch, epochError);

chart.Invalidate(chart.ClientRectangle);

chart.Update();

}

if (epochError < e0) break;

else continue;

}

}

public Double[][] PointsView { get { return pointsView; } }

public Int32 P { get { return p; } set { p = value; } }

public void Norm(Double minX, Double minY, Double maxX, Double maxY, int pointsCount)

{

///нормирование координат, чтобы все лежало на нашем рисовальном прямоугольнике

Double minx,miny,maxx,maxy,deltax,deltay;

Double xSize = maxX - minX;

Double ySize = maxY - minY;

maxx=maxy=Double.MinValue;

minx=miny=Double.MaxValue;

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

{

if (pointsArray[i][0] > maxx) maxx = pointsArray[i][0];

if (pointsArray[i][0] < minx) minx = pointsArray[i][0];

if (pointsArray[i][1] > maxy) maxy = pointsArray[i][1];

if (pointsArray[i][1] < miny) miny = pointsArray[i][1];

pointsView[i][0] = pointsArray[i][0];

pointsView[i][1] = pointsArray[i][1];

}

deltax = maxx - minx;

deltay = maxy - miny;

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

{

pointsView[i][0] -= minx;

pointsView[i][0] /= deltax;

pointsView[i][1] -= miny;

pointsView[i][1] /= deltay;

pointsView[i][0] = minX + pointsView[i][0] \* xSize;

pointsView[i][1] = minY + pointsView[i][1] \* ySize;

}

}

}

}

namespace lab\_5

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea2 = new System.Windows.Forms.DataVisualization.Charting.ChartArea();

System.Windows.Forms.DataVisualization.Charting.Legend legend2 = new System.Windows.Forms.DataVisualization.Charting.Legend();

System.Windows.Forms.DataVisualization.Charting.Series series2 = new System.Windows.Forms.DataVisualization.Charting.Series();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();

this.button1 = new System.Windows.Forms.Button();

this.epochs = new System.Windows.Forms.TextBox();

this.delta = new System.Windows.Forms.TextBox();

this.er0 = new System.Windows.Forms.TextBox();

this.radioButton1 = new System.Windows.Forms.RadioButton();

this.radioButton2 = new System.Windows.Forms.RadioButton();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.button2 = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();

this.SuspendLayout();

//

// pictureBox1

//

this.pictureBox1.BackColor = System.Drawing.SystemColors.ActiveCaptionText;

this.pictureBox1.Location = new System.Drawing.Point(7, 8);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(427, 401);

this.pictureBox1.TabIndex = 0;

this.pictureBox1.TabStop = false;

this.pictureBox1.Paint += new System.Windows.Forms.PaintEventHandler(this.pictureBox1\_Paint);

//

// chart1

//

chartArea2.Name = "ChartArea1";

this.chart1.ChartAreas.Add(chartArea2);

legend2.Name = "Legend1";

this.chart1.Legends.Add(legend2);

this.chart1.Location = new System.Drawing.Point(441, 8);

this.chart1.Name = "chart1";

series2.ChartArea = "ChartArea1";

series2.Legend = "Legend1";

series2.Name = "Series1";

this.chart1.Series.Add(series2);

this.chart1.Size = new System.Drawing.Size(282, 200);

this.chart1.TabIndex = 1;

this.chart1.Text = "chart1";

//

// button1

//

this.button1.Enabled = false;

this.button1.Location = new System.Drawing.Point(585, 386);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(138, 23);

this.button1.TabIndex = 2;

this.button1.Text = "Отобразить";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// epochs

//

this.epochs.Location = new System.Drawing.Point(585, 265);

this.epochs.Name = "epochs";

this.epochs.Size = new System.Drawing.Size(138, 20);

this.epochs.TabIndex = 3;

this.epochs.Text = "10000";

//

// delta

//

this.delta.Location = new System.Drawing.Point(585, 305);

this.delta.Name = "delta";

this.delta.Size = new System.Drawing.Size(138, 20);

this.delta.TabIndex = 3;

this.delta.Text = "0.01";

//

// er0

//

this.er0.Location = new System.Drawing.Point(585, 349);

this.er0.Name = "er0";

this.er0.Size = new System.Drawing.Size(138, 20);

this.er0.TabIndex = 3;

this.er0.Text = "0.1";

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Checked = true;

this.radioButton1.Location = new System.Drawing.Point(441, 215);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(110, 17);

this.radioButton1.TabIndex = 4;

this.radioButton1.TabStop = true;

this.radioButton1.Text = "Первый порядок";

this.radioButton1.UseVisualStyleBackColor = true;

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Location = new System.Drawing.Point(585, 215);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(106, 17);

this.radioButton2.TabIndex = 4;

this.radioButton2.Text = "Второй порядок";

this.radioButton2.UseVisualStyleBackColor = true;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(453, 265);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(56, 13);

this.label1.TabIndex = 5;

this.label1.Text = "Итераций";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(453, 308);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(45, 13);

this.label2.TabIndex = 5;

this.label2.Text = "Дельта";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(453, 352);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(47, 13);

this.label3.TabIndex = 5;

this.label3.Text = "Ошибка";

//

// button2

//

this.button2.Location = new System.Drawing.Point(441, 386);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(138, 23);

this.button2.TabIndex = 2;

this.button2.Text = "Загрузить";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(735, 421);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.radioButton2);

this.Controls.Add(this.radioButton1);

this.Controls.Add(this.er0);

this.Controls.Add(this.delta);

this.Controls.Add(this.epochs);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.chart1);

this.Controls.Add(this.pictureBox1);

this.Name = "Form1";

this.Text = "Сис. И.И. 5";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.PictureBox pictureBox1;

private System.Windows.Forms.DataVisualization.Charting.Chart chart1;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.TextBox epochs;

private System.Windows.Forms.TextBox delta;

private System.Windows.Forms.TextBox er0;

private System.Windows.Forms.RadioButton radioButton1;

private System.Windows.Forms.RadioButton radioButton2;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Button button2;

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using lab\_5.Classes;

namespace lab\_5

{

public partial class Form1 : Form

{

private Double[][] arr = null;

private Double[][] sample;

private Int32[] classes;

private Int32 sampleCount;

\_MyVisualisation visualizator = new \_MyVisualisation();

public Form1()

{

InitializeComponent();

chart1.Series.Clear();

chart1.Series.Add("Error(i)");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

chart1.Series[0].Color = Color.DarkSlateBlue;

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

Graphics graphics = e.Graphics;

graphics.Clear(Color.White);

if (arr != null)

{

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

{

Color color = Color.Black;

SolidBrush brush;

if (classes[i] == 1)

color = Color.Aquamarine;

if (classes[i] == 2)

color = Color.DarkGreen;

if (classes[i] == 3)

color = Color.OrangeRed;

brush = new SolidBrush(color);

graphics.FillEllipse(brush, (int)arr[i][0], (int)arr[i][1], 8, 8);

graphics.DrawString((i+1).ToString(),new Font(FontFamily.GenericSansSerif,10.0F,FontStyle.Bold),new SolidBrush(Color.Black),(float)(arr[i][0]-4.0), (float)(arr[i][1]-4.0));

}

}

}

public virtual void LoadSample(String Path)

{

using (FileStream fileStream = new FileStream(Path, FileMode.Open))

{

using (StreamReader streamReader = new StreamReader(fileStream))

{

///\*

String content = streamReader.ReadToEnd();

content = content.Replace('\n', ' ');

content = content.Replace('\r', ' ');

String[] buffer = content.Split(new Char[] { '#' }, StringSplitOptions.RemoveEmptyEntries);

Int32.TryParse(buffer[0], out sampleCount);

sample = new Double[sampleCount][];

classes = new Int32[sampleCount];

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

{

sample[i] = new Double[36];

String current = buffer[i + 1];

String[] bitBuffer = current.Split(new Char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int j;

for (j = 0; j < 36; j++)

{

Double.TryParse(bitBuffer[j], out sample[i][j]);

}

int ind;

Int32.TryParse(bitBuffer[j], out ind);

classes[i] = ind;

}

//\*/

}

}

}

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

{

Double deltai, e0i;

Int32 Epochsi;

Double.TryParse(delta.Text, out deltai);

Double.TryParse(er0.Text, out e0i);

Int32.TryParse(epochs.Text, out Epochsi);

visualizator.P = radioButton1.Checked ? 1 : 2;

visualizator.Run(sample, sampleCount, Epochsi,deltai, e0i,chart1,10);

visualizator.Norm(20, 20, pictureBox1.Width - 20, pictureBox1.Height - 20, sampleCount);

arr = visualizator.PointsView;

pictureBox1.Invalidate(pictureBox1.ClientRectangle);

}

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

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == DialogResult.OK)

{

LoadSample(ofd.FileName);

visualizator = new \_MyVisualisation();

visualizator.P = radioButton1.Checked ? 1 : 2;

visualizator.GeneratePoints(10, 10, pictureBox1.Width - 10, pictureBox1.Height - 10, sampleCount);

arr = visualizator.PointsView;

pictureBox1.Invalidate(pictureBox1.ClientRectangle);

button1.Enabled = true;

}

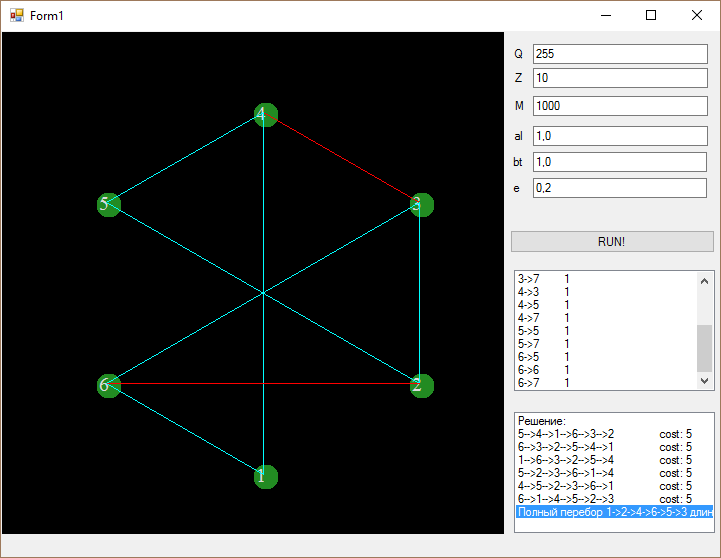
}

}

}

# 6. Муравьинный алгоритм

**Результаты работы программы:**



Муравьинный алгоритм дает хорошее решение, однако оно не оптимально. Полный перебор дает лучшие результаты.

**Исходный код программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ant

{

public class BruteForceAlgorithm

{

public BruteForceAlgorithm(double[,] graph)

{

\_graph = graph;

}

private List<int> \_bestRoute;

private double \_bestLength = double.MaxValue;

public void BruteForceFindBestRoute

(List<int> currentRoute,

List<int> citiesNotInRoute)

{

if (citiesNotInRoute.Count > 0)

{

for (int i = 0; i < citiesNotInRoute.Count; i++)

{

int justRemoved = citiesNotInRoute[0];

citiesNotInRoute.RemoveAt(0);

List<int> newRoute = new List<int>(currentRoute);

newRoute.Add(justRemoved);

BruteForceFindBestRoute(newRoute, citiesNotInRoute);

citiesNotInRoute.Add(justRemoved);

}

}

else

{

if (IsBestRoute(currentRoute))

{

\_bestRoute = currentRoute;

\_bestLength = GetRouteLength(\_bestRoute);

}

}

}

private double[,] \_graph;

public Tuple<string, double> FindBestRoute()

{

var notInListCities = new List<int>();

for (int i = 0; i < \_graph.GetLength(0); i++)

{

notInListCities.Add(i);

}

BruteForceFindBestRoute(new List<int>(), notInListCities);

double bestLength = GetRouteLength(\_bestRoute);

return new Tuple<string, double>(string.Join("->", \_bestRoute.Select((item) => item + 1)), bestLength);

}

private double GetRouteLength(List<int> cities)

{

double bestLength = 0;

for (int i = 0; i < cities.Count - 1; i++)

{

bestLength += \_graph[cities[i], cities[i + 1]];

}

bestLength += \_graph[cities.Last(), cities.First()];

return bestLength;

}

private bool IsBestRoute(List<int> route)

{

return (GetRouteLength(route) < \_bestLength);

}

}

namespace Ant

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.button1 = new System.Windows.Forms.Button();

this.panel1 = new System.Windows.Forms.Panel();

this.label1 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.label2 = new System.Windows.Forms.Label();

this.textBox2 = new System.Windows.Forms.TextBox();

this.label3 = new System.Windows.Forms.Label();

this.textBox3 = new System.Windows.Forms.TextBox();

this.label4 = new System.Windows.Forms.Label();

this.textBox4 = new System.Windows.Forms.TextBox();

this.label5 = new System.Windows.Forms.Label();

this.textBox5 = new System.Windows.Forms.TextBox();

this.label6 = new System.Windows.Forms.Label();

this.textBox6 = new System.Windows.Forms.TextBox();

this.listBox1 = new System.Windows.Forms.ListBox();

this.listBox2 = new System.Windows.Forms.ListBox();

this.SuspendLayout();

//

// button1

//

this.button1.Location = new System.Drawing.Point(509, 199);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(205, 23);

this.button1.TabIndex = 0;

this.button1.Text = "RUN!";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// panel1

//

this.panel1.BackColor = System.Drawing.SystemColors.ControlText;

this.panel1.Location = new System.Drawing.Point(1, 1);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(502, 502);

this.panel1.TabIndex = 1;

this.panel1.Paint += new System.Windows.Forms.PaintEventHandler(this.panel1\_Paint);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(511, 16);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(15, 13);

this.label1.TabIndex = 2;

this.label1.Text = "Q";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(532, 13);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(175, 20);

this.textBox1.TabIndex = 3;

this.textBox1.Text = "255";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(511, 40);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(14, 13);

this.label2.TabIndex = 2;

this.label2.Text = "Z";

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(532, 37);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(175, 20);

this.textBox2.TabIndex = 3;

this.textBox2.Text = "10";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(511, 68);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(16, 13);

this.label3.TabIndex = 2;

this.label3.Text = "M";

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(532, 65);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(175, 20);

this.textBox3.TabIndex = 3;

this.textBox3.Text = "1000";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(511, 98);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(15, 13);

this.label4.TabIndex = 2;

this.label4.Text = "al";

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(532, 95);

this.textBox4.Name = "textBox4";

this.textBox4.Size = new System.Drawing.Size(175, 20);

this.textBox4.TabIndex = 3;

this.textBox4.Text = "1,0";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(510, 124);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(16, 13);

this.label5.TabIndex = 2;

this.label5.Text = "bt";

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(532, 121);

this.textBox5.Name = "textBox5";

this.textBox5.Size = new System.Drawing.Size(174, 20);

this.textBox5.TabIndex = 3;

this.textBox5.Text = "1,0";

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(510, 150);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(13, 13);

this.label6.TabIndex = 2;

this.label6.Text = "e";

//

// textBox6

//

this.textBox6.Location = new System.Drawing.Point(532, 147);

this.textBox6.Name = "textBox6";

this.textBox6.Size = new System.Drawing.Size(174, 20);

this.textBox6.TabIndex = 3;

this.textBox6.Text = "0,2";

//

// listBox1

//

this.listBox1.FormattingEnabled = true;

this.listBox1.Location = new System.Drawing.Point(513, 381);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(201, 121);

this.listBox1.TabIndex = 4;

//

// listBox2

//

this.listBox2.FormattingEnabled = true;

this.listBox2.Location = new System.Drawing.Point(513, 239);

this.listBox2.Name = "listBox2";

this.listBox2.Size = new System.Drawing.Size(201, 121);

this.listBox2.TabIndex = 4;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(719, 526);

this.Controls.Add(this.listBox2);

this.Controls.Add(this.listBox1);

this.Controls.Add(this.textBox6);

this.Controls.Add(this.textBox5);

this.Controls.Add(this.textBox4);

this.Controls.Add(this.textBox3);

this.Controls.Add(this.textBox2);

this.Controls.Add(this.label6);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label5);

this.Controls.Add(this.label4);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.panel1);

this.Controls.Add(this.button1);

this.DoubleBuffered = true;

this.Name = "Form1";

this.Text = "Form1";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.TextBox textBox5;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.TextBox textBox6;

private System.Windows.Forms.ListBox listBox1;

private System.Windows.Forms.ListBox listBox2;

}

}

} using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Reflection;

namespace Ant

{

public partial class Form1 : Form

{

double[,] gr;

double[,] tau, grv;

int [,] lw;

Solver s;

Color[,] colors;

int c;

static Random r = new Random();

public Form1()

{

InitializeComponent();

typeof(Panel).InvokeMember("DoubleBuffered",

BindingFlags.SetProperty | BindingFlags.Instance | BindingFlags.NonPublic, null, panel1, new object[] { true });

String buffer;

String[] bufferArray;

Char[] Separators = new Char[] { ' ' };

using (FileStream inputFileStream = File.Open("inp.txt", FileMode.Open))

{

using (StreamReader streamReader = new StreamReader(inputFileStream))

{

buffer = streamReader.ReadLine();

Int32.TryParse(buffer, out c);

gr = new double[c, c];

grv = new Double[c, 2];

tau = new double[c, c];

colors = new Color[c, c];

lw = new int[c, c];

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

{

buffer = streamReader.ReadLine();

bufferArray = buffer.Split(Separators, StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < c; j++)

{

Double.TryParse(bufferArray[j], out gr[i, j]);

if (gr[i, j] > 0)

listBox2.Items.Add((i + 1).ToString() + "->" + (j + i).ToString() + "\t" + Math.Round(gr[i, j], 1).ToString());

tau[i, j] = 0.5;

}

}

}

}

}

public void UpdateG(object sender, EventArgs e)

{

this.tau = s.Pheromons;

Refresh();

}

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

{

#region parametres

int z, q, m;

double al, bt, ug;

Int32.TryParse(textBox1.Text, out q);

Int32.TryParse(textBox2.Text, out z);

Int32.TryParse(textBox3.Text, out m);

Double.TryParse(textBox4.Text, out al);

Double.TryParse(textBox5.Text, out bt);

Double.TryParse(textBox6.Text, out ug);

#endregion

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

{

for (int j = 0; j < c; j++)

{

tau[i, j] = 0.5;

}

}

this.Refresh();

listBox1.Items.Clear();

s = new Solver();

s.Graph = gr;

s.Pheromons = tau;

s.CityCount = c;

s.IterationFinished += new EventHandler(UpdateG);

var routesAndLength = s.Solve(al, bt, ug, z, m, q);

listBox1.Items.Clear();

listBox1.Items.Clear();

listBox1.Items.Add("Решение:");

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

if (!String.IsNullOrEmpty(routesAndLength[i]))

listBox1.Items.Add(routesAndLength[i]);

BruteForceAlgorithm bruteForce = new BruteForceAlgorithm(gr);

var bestRoute = bruteForce.FindBestRoute();

listBox1.Items.Add($"Полный перебор {bestRoute.Item1} длинной {bestRoute.Item2}");

//s.IterationFinished += new EventHandler(UpdateG);

//string[] str = s.Solve(al, bt, ug, z, m, q);

//listBox1.Items.Clear();

//listBox1.Items.Add("Решение:");

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

// if (!String.IsNullOrEmpty(str[i]))

// listBox1.Items.Add(str[i]);

}

private void fillColors()

{

double maxf = -1;

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

for (int j = 0; j < c; j++)

if (tau[i,j] > maxf)

maxf = tau[i,j];

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

for (int j = 0; j < c; j++)

{

double val = 255 \* tau[i, j] / maxf;

if (val < 50)

{

colors[i, j] = Color.Red;

lw[i, j] = (int)(3.0 \* val / 50 + 1);

}

if (val >= 50 && val < 140)

{

colors[i, j] = Color.Yellow;

lw[i, j] = (int)(3.0 \* val / 140 + 2);

}

if (val >= 140 )

{

colors[i, j] = Color.Aqua;

lw[i, j] = (int)(3.0 \* val / 255 + 4);

}

}

}

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

{

Graphics g = e.Graphics;

fillColors();

int coordx;

int coordy;

double rotate = ((360 / c) \* Math.PI \* 2) / 360;

int startx = panel1.Width / 2;

int starty = panel1.Height / 2;

int rad = panel1.Width / 2 - 70;

Font f = new Font(FontFamily.GenericSerif, 15);

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

{

coordx = (int)(rad \* Math.Sin(rotate \* i) + startx);

coordy = (int)(rad \* Math.Cos(rotate \* i) + starty);

grv[i,0] = coordx;

grv[i,1] = coordy;

e.Graphics.FillEllipse(new SolidBrush(Color.ForestGreen), coordx, coordy, 25, 25);

e.Graphics.DrawString((i+1).ToString(), f, new SolidBrush(Color.Gainsboro), coordx, coordy);

}

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

{

for (int j = 0; j < c; j++)

{

if (gr[i,j] > 0)

{

Pen p = new Pen(colors[i, j], (float)lw[i, j]);

e.Graphics.DrawLine(new Pen(colors[i,j]), (float)grv[i,0] + 10, (float)grv[i,1] + 10, (float)grv[j,0] + 10, (float)grv[j,1] + 10);

float mx, my;

//mx = (float)(grv[i, 0] + grv[j, 0]) / 3;

//my = (float)(grv[i, 1] + grv[j, 1]) / 3;

//e.Graphics.DrawString(gr[i,j].ToString(), f, new SolidBrush(Color.Gainsboro), mx,my);

}

}

}

System.Threading.Thread.Sleep(200);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Ant

{

class MyAnt

{

public Int32 InitialPosition { get; set; }

public List<int> Way { get; set; }

public List<Double> W { get; set; }

public Int32 CurrentPosition {get;set;}

public Double L { get; set; }

//public HashSet<int> VisitedCity { get; set; }

public Int32[] mark { get; set; }

public Int32 Was { get; set; }

public Boolean life { get; set; }

public Boolean done { get; set; }

public void init(int n)

{

//VisitedCity = new HashSet<int>();

Way = new List<int>();

W = new List<double>();

L = 0;

Was = 0;

mark = new Int32[n];

}

}

class Probability : IComparable

{

public Double P { get; set; }

public Int32 N { get; set; }

#region Члены IComparable

public int CompareTo(object obj)

{

return P.CompareTo(((Probability)obj).P);

}

#endregion

}

class Solver

{

#region private

private Double[,] graph;

private static Random r = new Random();

private Double[,] pheromons;

private Int32 iterations;

private Int32 Q = 255, Z;

private Double alpha, betta, e;

private MyAnt[] ants;

private List<Probability> GoNext(MyAnt ant)

{

List<Probability> candidates = new List<Probability>();

int t = ant.CurrentPosition,s;

double propab, sum = 0;

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

{

s = i;

if (graph[t, s] > 0 && ant.mark[s] == 0 )//не были в вершине

{

sum += Math.Pow(pheromons[t, s], alpha) / Math.Pow(graph[t, s],betta);

}

}

if (sum == 0)

return candidates;

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

{

s = i;

if (graph[t, s] > 0 && ant.mark[s] == 0)

{

Probability p = new Probability();

propab = (Math.Pow(pheromons[t, s], alpha) / Math.Pow(graph[t, s], betta)) / sum;

p.N = s;

p.P = propab;

candidates.Add(p);

}

}

candidates.Sort();

return candidates;

}

private Int32 Select(MyAnt ant)

{

List<Probability> candidates = GoNext(ant);

int index=-1;

if (candidates.Count == 0) index = -1;

else

{

if (r.Next() % 100 < 30)

{

for (int i = 1; i < candidates.Count; i++)

candidates[i].P += candidates[i - 1].P;

double p = r.NextDouble();

if (p < candidates[0].P) return candidates[0].N;

for (int i = 0; i < candidates.Count - 1; i++)

if (p > candidates[i].P && p < candidates[i + 1].P)

{

index = candidates[i + 1].N;

break;

}

}

else index = candidates[candidates.Count - 1].N;

}

return index;

}

private void CheckRoute(MyAnt ant)

{

ant.life = false;

if (ant.Was == CityCount && ant.L < MaxLen)

{

//if (graph[ant.CurrentPosition, ant.InitialPosition] > 0)

{

//ant.Way.Add(ant.InitialPosition);

//ant.L += graph[ant.CurrentPosition, ant.InitialPosition];

int start, finish;

start = ant.Way.ElementAt(0);

for (int i = 1; i < ant.Way.Count; i++)

{

finish = ant.Way.ElementAt(i);

ant.L = ant.W.Sum();

pheromons[start, finish] = Q / ant.L;

start = finish;

}

ant.life = true;

return;

}

}

}

#endregion

public String[] Solve(Double alpha, Double betta, Double e, Int32 Z,int M, int Q)

{

this.iterations = M;

this.Z = Z;

this.alpha = alpha;

this.betta = betta;

this.e = e;

this.Q = Q;

MaxLen = Double.MaxValue;

ants = new MyAnt[Z];

for (int iter = 0; iter < iterations; iter++)

{

for (int z = 0; z < Z; z++)

{

ants[z] = new MyAnt();

ants[z].L = 0;

ants[z].done = false;

ants[z].init(CityCount);

ants[z].InitialPosition = ants[z].CurrentPosition = r.Next() % CityCount;

ants[z].Way.Add(ants[z].InitialPosition);

ants[z].mark[ants[z].InitialPosition]++;

ants[z].Was++;

}

while (true)

{

int k = 0;

for (int z = 0; z < Z; z++)

{

int Next = Select(ants[z]);

//while (true)

//{

Next = Select(ants[z]);

if (Next == -1)

{

ants[z].done = true;

//CheckRoute(ants[z]);

}

else

{

k++;

ants[z].Was++;

ants[z].mark[Next]++;

ants[z].Way.Add(Next);

ants[z].L += graph[ants[z].CurrentPosition, Next];

ants[z].W.Add(graph[ants[z].CurrentPosition, Next]);

ants[z].CurrentPosition = Next;

}

//}

//CheckRoute(ants[z]);

}

if (k == 0) break;

}

for (int z = 0; z < Z; z++)

{

CheckRoute(ants[z]);

}

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

for (int j = 0; j < CityCount; j++)

pheromons[i, j] \*= (1 - e);

if (iter % 20 == 0)

{

EventHandler tmp = IterationFinished;

tmp(this, null);

}

}

string[] str = new string[Z];

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

{

str[i] = "";

StringBuilder sb = new StringBuilder();

if (ants[i].life && ants[i].Was == CityCount)

{

for (int j = 0; j < ants[i].Way.Count; j++)

sb.Append((ants[i].Way[j] + 1).ToString() + (j == ants[i].Way.Count - 1 ? "" : "-->"));

sb.Append("\tcost: " +ants[i].L.ToString());

}

if (!str.Contains(sb.ToString()))

str[i] = sb.ToString();

}

return str;

}

public Double[,] Graph { get { return graph; } set { graph = value; } }

public Double[,] Pheromons { get { return pheromons; } set { pheromons = value; } }

public Int32 CityCount { get; set; }

public Double MaxLen { get; set; }

public event EventHandler IterationFinished;

}

}

# 8. Итеративный метод Браун-Робинсона

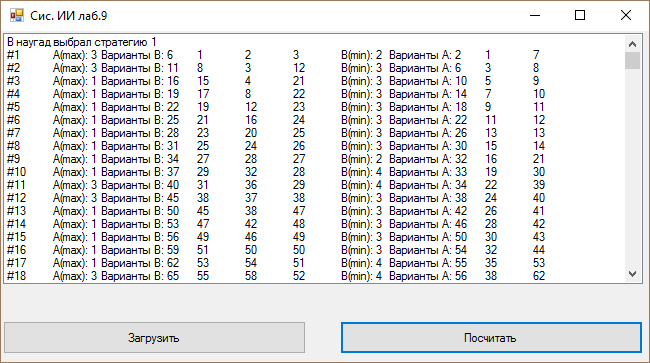
**Результат работы программы для решения игры 3x4**

3 4

3 2 4 1

6 1 2 3

5 7 1 9



**Исходный код**

namespace Iterations

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.listBox1 = new System.Windows.Forms.ListBox();

this.button1 = new System.Windows.Forms.Button();

this.button2 = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// listBox1

//

this.listBox1.FormattingEnabled = true;

this.listBox1.Location = new System.Drawing.Point(2, 2);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(640, 251);

this.listBox1.TabIndex = 0;

//

// button1

//

this.button1.Location = new System.Drawing.Point(2, 290);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(303, 33);

this.button1.TabIndex = 1;

this.button1.Text = "Загрузить";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// button2

//

this.button2.Location = new System.Drawing.Point(339, 290);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(303, 33);

this.button2.TabIndex = 1;

this.button2.Text = "Посчитать";

this.button2.UseVisualStyleBackColor = true;

this.button2.Click += new System.EventHandler(this.button2\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(646, 326);

this.Controls.Add(this.button2);

this.Controls.Add(this.button1);

this.Controls.Add(this.listBox1);

this.Name = "Form1";

this.Text = "Сис. ИИ лаб.9";

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.ListBox listBox1;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Button button2;

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Iterations

{

public partial class Form1 : Form

{

private Random randomGenerator = new Random();

private Double[,] gameTable;

private Int32 aStrategiesCount, bStrategiesCount, IterationsCount;

private Double[] bCurrentChoice, aCurrentChoice;

private Int32[] bChoiceRate, aChoiceRate;

private Int32 bLastChosen, aLastChosen;

Int32 pureA, pureB;

public Form1()

{

InitializeComponent();

}

private void Memory(int a, int b)

{

gameTable = new double[a, b];

bCurrentChoice = new double[b];

aCurrentChoice = new double[a];

bChoiceRate = new int[b];

aChoiceRate = new int[a];

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

String buffer;

String[] bufferArray;

Char[] Separators = new Char[]{' '};

using (FileStream inputFileStream = File.Open(openFileDialog.FileName, FileMode.Open))

{

using (StreamReader streamReader = new StreamReader(inputFileStream))

{

buffer = streamReader.ReadLine();

bufferArray = buffer.Split(Separators, StringSplitOptions.RemoveEmptyEntries);

Int32.TryParse(bufferArray[0], out aStrategiesCount);

Int32.TryParse(bufferArray[1], out bStrategiesCount);

this.Memory(aStrategiesCount, bStrategiesCount);

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

{

buffer = streamReader.ReadLine();

bufferArray = buffer.Split(Separators, StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < bStrategiesCount; j++)

Double.TryParse(bufferArray[j], out gameTable[i, j]);

}

}

}

}

}

private bool validateTable()

{

Int32[] Cminjs = new Int32[aStrategiesCount];

double[] cmins = new double[aStrategiesCount], cmax = new double[bStrategiesCount];

Int32 Cmini;

Int32[] Cmaxis = new Int32[bStrategiesCount];

Int32 Cmaxj, Aw, Bw, Ab, Bb ;

Double tempval;

//for a

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

{

Cminjs[i] = 0;

tempval = gameTable[i, 0];

for(int j=1;j<bStrategiesCount;j++)

if (gameTable[i, j] < tempval)

{

tempval = gameTable[i, j];

Cminjs[i] = j;

}

cmins[i] = tempval;

}

Cmini = 0;

tempval = cmins[0];

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

{

if (cmins[i] > tempval)

{

tempval = cmins[i];

Cmini = i;

}

}

Aw = Cmini;

Bw = Cminjs[Cmini];

for (int j = 0; j < bStrategiesCount; j++)

{

Cmaxis[j] = 0;

tempval = gameTable[0, j];

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

if (gameTable[i, j] > tempval)

{

tempval = gameTable[i, j];

Cmaxis[j] = i;

}

cmax[j] = tempval;

}

Cmaxj = 0;

tempval = cmax[0];

for (int j = 1; j < bStrategiesCount; j++)

{

if (cmax[j] < tempval)

{

tempval = cmax[j];

Cmaxj = j;

}

}

Ab = Cmaxis[Cmaxj];

Bb = Cmaxj;

if (Ab == Aw && Bw == Bb || gameTable[Cmini, Cminjs[Cmini]] >= gameTable[Cmaxis[Cmaxj], Cmaxj])

{

pureA = Ab;

pureB = Bb;

return true;

}

return false;

}

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

{

if (validateTable())

{

listBox1.Items.Add("Есть седловая точка A" + (pureA + 1).ToString() + "B" + (pureB + 1).ToString());

return;

}

IterationsCount = 1000;

listBox1.Items.Clear();

bLastChosen = 0;// randomGenerator.Next() % aStrategiesCount;

double tv;

int frst = bLastChosen;

listBox1.Items.Add("B наугад выбрал стратегию " + (bLastChosen + 1).ToString());

//for (int i = 0; i < aStrategiesCount; i++)

// aCurrentChoice[i] += gameTable[i, bLastChosen];

tv = gameTable[0,bLastChosen];

aLastChosen = 0;

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

if (gameTable[i,bLastChosen] > tv)

{

tv = gameTable[i, bLastChosen];

aLastChosen = i;

}

aChoiceRate[aLastChosen]++;

for (int iter = 0; iter < IterationsCount; iter++)

{

for (int j = 0; j < bStrategiesCount; j++)

bCurrentChoice[j] += gameTable[aLastChosen, j];

tv = bCurrentChoice[0];

bLastChosen = 0;

for(int j=1;j<bStrategiesCount;j++)

if (bCurrentChoice[j] < tv)

{

tv = bCurrentChoice[j];

bLastChosen = j;

}

bChoiceRate[bLastChosen]++;

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

aCurrentChoice[i] += gameTable[i, bLastChosen];

tv = aCurrentChoice[0];

aLastChosen = 0;

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

if (aCurrentChoice[i] > tv)

{

tv = aCurrentChoice[i];

aLastChosen = i;

}

aChoiceRate[aLastChosen]++;

StringBuilder sb = new StringBuilder();

sb.Append("#" + (iter + 1).ToString());

sb.Append("\tA(max): " + (aLastChosen + 1).ToString()+"\tВарианты B: ");

for (int j = 0; j < bStrategiesCount; j++)

sb.Append(Math.Round(bCurrentChoice[j], 2).ToString() + "\t");

sb.Append("B(min): " + (bLastChosen + 1).ToString() + "\tВарианты A: ");

for (int j = 0; j < aStrategiesCount; j++)

sb.Append(Math.Round(aCurrentChoice[j], 2).ToString() + "\t");

listBox1.Items.Add(sb.ToString());

}

StringBuilder sb1 = new StringBuilder();

sb1.Append("Результат\t");

sb1.Append("Частоты B: ");

for (int j = 0; j < bStrategiesCount; j++)

sb1.Append(Math.Round((double)bChoiceRate[j]/IterationsCount, 3).ToString() + "\t");

sb1.Append("\tЧастоты A: ");

for (int j = 0; j < aStrategiesCount; j++)

sb1.Append(Math.Round((double)aChoiceRate[j] / IterationsCount, 3).ToString() + "\t");

listBox1.Items.Add(sb1.ToString());

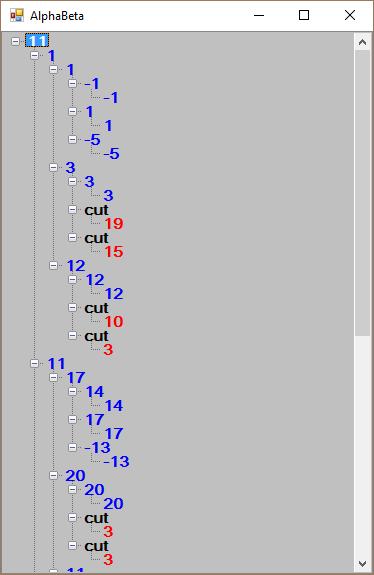
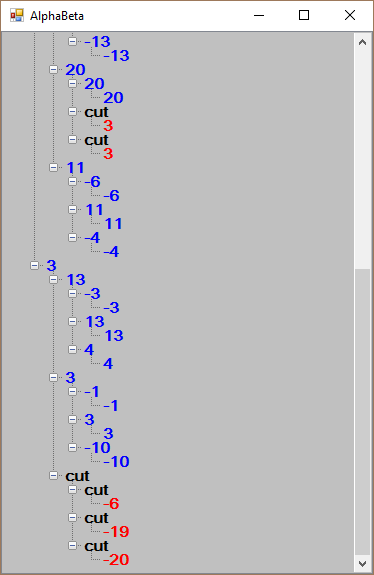
}

}

}

# Альфа-бета отсечение

**Результат работы программы:**

**Исходный код**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace alphabeta

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ReadTree();

AlphaBeta();

}

private void ReadTree()

{

TreeNode currentNode;

using (StreamReader estimationReader = new StreamReader(new FileStream(@"..\..\gameestimation.txt", FileMode.Open)))

{

int vertexCount = int.Parse(estimationReader.ReadLine());

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

{

string input = estimationReader.ReadLine();

var line = input.Split(' ');

int vertexNumber = int.Parse(line[0]);

if (i == 0)

{

treeView1.Nodes.Add(vertexNumber.ToString(), "x");

}

currentNode = treeView1.Nodes.Find(vertexNumber.ToString(), true).First();

int childCount = int.Parse(line[1]);

if (childCount == 0)

{

double weight = double.Parse(line[2]);

TreeNode leafNode = new AlphaBetaNode(vertexNumber.ToString())

{

Weight = weight

};

currentNode.Nodes.Add(leafNode);

}

else

{

foreach (var child in line.Skip(2))

{

currentNode.Nodes.Add(child,"cut");

}

}

}

}

treeView1.ExpandAll();

}

private double AlphaBeta(double alpha, double beta, bool isMaximizing,TreeNode node)

{

double estimation;

node.ForeColor = Color.Blue;

if (node.Nodes.Count == 0)

{

return ((AlphaBetaNode)node).Weight;

}

if (isMaximizing)

{

estimation = double.MinValue;

for (int i = 0; i < node.Nodes.Count; i++)

{

estimation = Math.Max(estimation, AlphaBeta(alpha, beta, false, node.Nodes[i]));

if (estimation > beta)

{

node.Text = estimation.ToString();

return estimation;

}

if (estimation > alpha) alpha = estimation;

}

}

else

{

estimation = double.MaxValue;

for (int i = 0; i < node.Nodes.Count; i++)

{

estimation = Math.Min(estimation, AlphaBeta(alpha, beta, true, node.Nodes[i]));

if (estimation < alpha)

{

node.Text = estimation.ToString();

return estimation;

}

if (estimation < beta) beta = estimation;

}

}

node.Text = estimation.ToString();

return estimation;

}

private void AlphaBeta()

{

AlphaBeta(double.MinValue,double.MaxValue,true, treeView1.Nodes[0]);

}

}

}

namespace alphabeta

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.treeView1 = new System.Windows.Forms.TreeView();

this.SuspendLayout();

//

// treeView1

//

this.treeView1.BackColor = System.Drawing.Color.Silver;

this.treeView1.Dock = System.Windows.Forms.DockStyle.Fill;

this.treeView1.Font = new System.Drawing.Font("Arial Unicode MS", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.treeView1.ItemHeight = 14;

this.treeView1.Location = new System.Drawing.Point(0, 0);

this.treeView1.Name = "treeView1";

this.treeView1.Size = new System.Drawing.Size(284, 270);

this.treeView1.TabIndex = 0;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(284, 270);

this.Controls.Add(this.treeView1);

this.Name = "Form1";

this.Text = "AlphaBeta";

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.TreeView treeView1;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace alphabeta

{

public class AlphaBetaNode:TreeNode

{

public AlphaBetaNode(string text):base(text)

{

ForeColor = System.Drawing.Color.Red;

}

private double \_weight;

public double Weight

{

get { return \_weight; }

set { \_weight = value; Text = \_weight.ToString(); }

} //public double Alpha { get; set; }

public string VertexNumber { get; set; } //public int Beta { get; set; }

}

}