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)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г.

дата

Комплексный отчет

по выполнению индивидуальных заданий

\_\_\_\_\_\_\_\_\_\_\_\_\_ИЗ 09.04.04.28.000 О\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

обозначение документа

Дисциплина: «Инженерия знаний и системы искусственного интеллекта»

наименование дисциплины

Студент группы 8ПИ-61 В. И. Корней

(и.о., фамилия)

Научный руководитель: доцент, к.т.н. А.Н.Тушев

должность, ученое звание (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. Динамическая картина. 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

Кубический подграф

Условие. Задан граф G=(V,E).

Вопрос. Существует ли непустое подмножество E’⊆E, такое, что в графе G’=(V,E’) любая вершина имеет степень, равную 3 или 0?

Кодирование организма

Организм кодируется следующим образом:

1 если ребро включено в подграф, 0 если ребро не включено в подграф. Битов в организме столько, сколько ребер в графе.

Функция приспособленности

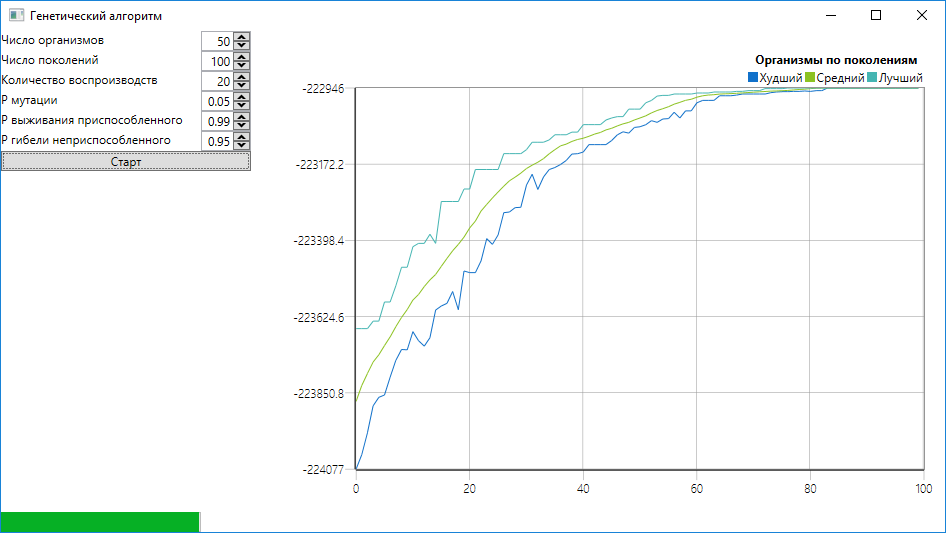
В случае, если ребер в графе нет вводим пенальти.

В случае, если степень вершины 0 или 3 добавляем бонусы.

Для сходимости к степени 3 требуется различать плохие и более плохие состояния. Поэтому в случае если степень болеше 3, то пенальти уможается на степень вершины.

Таким образом, функция приспособленности представляет из себя сумму бонусов – сумма пенальти.

Результаты работы программы



Исходный код

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace ASTU.GeneticAlgorithm

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace ASTU.GeneticAlgorithm

{

internal class Command<T> : ICommand

{

private readonly Action<T> \_action;

private readonly Func<T, bool> \_predicate;

public event EventHandler CanExecuteChanged;

public Command(Action<T> action, Func<T, bool> predicate)

{

this.\_action = action;

this.\_predicate = predicate;

}

public Command(Action<T> action)

{

this.\_action = action;

\_predicate = obj => true;

}

public bool CanExecute(object parameter)

{

var documents = (T)parameter;

return \_predicate(documents);

}

public void Execute(object parameter)

{

\_action((T)parameter);

}

public void RaiseCanExecuteChanged()

{

CanExecuteChanged?.Invoke(this, new EventArgs());

}

}

}

﻿using ASTU.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

internal class CubeGraphGeneticAlgorithm : GeneticAlgorithm

{

public CubeGraphGeneticAlgorithm(Graph graph)

{

\_graph = graph;

}

private Graph \_graph;

public override Organizm CreateOrganizm()

{

return new Organizm(\_graph);

}

internal override Tuple<Organizm, Organizm> ProduceChildren(Organizm parent1, Organizm parent2)

{

return Organizm.ProduceChildren(\_graph, parent1, parent2);

}

public override double MeasureFitness(Organizm organizm)

{

return organizm.MeasureFitness(\_graph);

}

internal override Organizm ProduceMutant(Organizm organizm)

{

return organizm.Mutate(\_graph);

}

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

internal abstract class GeneticAlgorithm

{

public GeneticAlgorithm()

{

\_population = new List<Organizm>();

}

private GeneticAlgoritmParameters \_geneticAlgorithmParameters;

private List<Organizm> \_population;

private List<PopulationHistoryItem> \_history = new List<PopulationHistoryItem>();

public IList<PopulationHistoryItem> History

{

get

{

return \_history;

}

}

public abstract Organizm CreateOrganizm();

public void InitPopulation()

{

for (int i = 0; i < \_geneticAlgorithmParameters.InitialPopulationSize; i++)

{

\_population.Add(CreateOrganizm());

}

}

public void Execute(GeneticAlgoritmParameters geneticParameters,Action<int> progressCallback)

{

\_geneticAlgorithmParameters = geneticParameters;

\_history.Clear();

InitPopulation();

for (int i = 0; i < \_geneticAlgorithmParameters.GenerationCount; i++)

{

\_history.Add(new PopulationHistoryItem()

{

BestOrganismFitness = \_population.Max((organism) => MeasureFitness(organism)),

WorstOrganismFitness = \_population.Min((organism) => MeasureFitness(organism)),

AverageOrganismFitness = \_population.Average((organism) => MeasureFitness(organism)),

Generation = i,

});

ExecuteStep();

progressCallback(i\*100/\_geneticAlgorithmParameters.GenerationCount);

}

}

public void ExecuteStep()

{

ProduceChildren();

Mutate();

NaturalSelect();

}

internal void NaturalSelect()

{

HashSet<int> fightOrganisms = new HashSet<int>();

HashSet<int> diedOrganisms = new HashSet<int>();

for (int i = 0; i < \_geneticAlgorithmParameters.ReproductionNumber \* 2; i++)

{

int randomFighterIndex1 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize + \_geneticAlgorithmParameters.ReproductionNumber \* 2);

while (fightOrganisms.Contains(randomFighterIndex1))

{

randomFighterIndex1 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize + \_geneticAlgorithmParameters.ReproductionNumber \* 2);

}

int randomFighterIndex2 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize + \_geneticAlgorithmParameters.ReproductionNumber \* 2);

while (randomFighterIndex2 != randomFighterIndex1 && fightOrganisms.Contains(randomFighterIndex2))

{

randomFighterIndex2 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize + \_geneticAlgorithmParameters.ReproductionNumber \* 2);

}

fightOrganisms.Add(randomFighterIndex1);

fightOrganisms.Add(randomFighterIndex2);

int deathIndex;

if (RandomHelper.NextDouble() < \_geneticAlgorithmParameters.BadOrganizmDeathProbability)

{

deathIndex = (MeasureFitness(\_population[randomFighterIndex1]) < MeasureFitness(\_population[randomFighterIndex2])) ? randomFighterIndex1 : randomFighterIndex2;

}

else

{

deathIndex = (MeasureFitness(\_population[randomFighterIndex1]) < MeasureFitness(\_population[randomFighterIndex2])) ? randomFighterIndex2 : randomFighterIndex1;

}

diedOrganisms.Add(deathIndex);

}

var newPopulation = new List<Organizm>();

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

{

if (!diedOrganisms.Contains(i))

{

newPopulation.Add(\_population[i]);

}

}

\_population = newPopulation;

}

public abstract double MeasureFitness(Organizm organizm);

internal void Mutate()

{

Parallel.For(0, \_population.Count, (i) =>

{

{

if (RandomHelper.NextDouble() < \_geneticAlgorithmParameters.MutationProbability)

{

var originalOrganizm = \_population[i];

var mutant = ProduceMutant(\_population[i]);

Organizm leftOrganizm;

if (RandomHelper.NextDouble() < \_geneticAlgorithmParameters.GoodOrganizmSurvivalProbability)

{

leftOrganizm = MeasureFitness(mutant) > MeasureFitness(originalOrganizm) ? mutant : originalOrganizm;

}

else

{

leftOrganizm = MeasureFitness(mutant) < MeasureFitness(originalOrganizm) ? mutant : originalOrganizm;

}

\_population[i] = leftOrganizm;

}

}

});

}

internal abstract Tuple<Organizm, Organizm> ProduceChildren(Organizm parent1, Organizm parent2);

internal abstract Organizm ProduceMutant(Organizm organizm);

HashSet<int> \_organizmUsedForReproduction;

internal void ProduceChildren()

{

\_organizmUsedForReproduction = new HashSet<int>();

for (int i = 0; i < \_geneticAlgorithmParameters.ReproductionNumber; i++)

{

int randomParentIndex1 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize);

while (\_organizmUsedForReproduction.Contains(randomParentIndex1))

{

randomParentIndex1 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize);

}

int randomParentIndex2 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize);

while (randomParentIndex2 != randomParentIndex1 && \_organizmUsedForReproduction.Contains(randomParentIndex2))

{

randomParentIndex2 = RandomHelper.NextInt(\_geneticAlgorithmParameters.InitialPopulationSize);

}

\_organizmUsedForReproduction.Add(randomParentIndex1);

\_organizmUsedForReproduction.Add(randomParentIndex2);

var parent1 = \_population[randomParentIndex1];

var parent2 = \_population[randomParentIndex2];

var children = ProduceChildren(parent1, parent2);

\_population.Add(children.Item1);

\_population.Add(children.Item2);

}

}

}

}

﻿using ASTU.Model;

using Sparrow.Chart;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace ASTU.GeneticAlgorithm

{

public class GeneticAlgorithmParametersViewModel : Observable

{

public GeneticAlgorithmParametersViewModel()

{

\_geneticAlgorithm = new CubeGraphGeneticAlgorithm(Graph.FromFile(@"..\..\GraphData.txt"));

\_startGeneticAlgorithmCommand = new Command<object>((mockParams) =>

{

new Task(() =>

{

\_geneticAlgorithm.Execute(\_geneticParameters, (progress) => ProgressValue = progress);

var averageFitness = new PointsCollection();

var maxFitness = new PointsCollection();

var minFitness = new PointsCollection();

var maxPoints = \_geneticAlgorithm.History.Select((historyItem) => new DoublePoint() { Data = historyItem.Generation, Value = historyItem.BestOrganismFitness });

foreach (var point in maxPoints)

{

maxFitness.Add(point);

}

var minPoints = \_geneticAlgorithm.History.Select((historyItem) => new DoublePoint() { Data = historyItem.Generation, Value = historyItem.WorstOrganismFitness });

foreach (var point in minPoints)

{

minFitness.Add(point);

}

var averagePoints = \_geneticAlgorithm.History.Select((historyItem) => new DoublePoint() { Data = historyItem.Generation, Value = historyItem.AverageOrganismFitness });

foreach (var point in averagePoints)

{

averageFitness.Add(point);

}

AverageFitness = averageFitness;

MinFitness = minFitness;

MaxFitness = maxFitness;

NotifyPropertyChanged(() => MaxGridValue);

}).Start();

});

}

private int \_progressValue = 0;

public int ProgressValue

{

get

{

return \_progressValue;

}

set

{

if (value != \_progressValue)

{

\_progressValue = value;

NotifyPropertyChanged(() => ProgressValue);

}

}

}

private GeneticAlgorithm \_geneticAlgorithm;

private GeneticAlgoritmParameters \_geneticParameters = new GeneticAlgoritmParameters();

public int InitialPopulationSize

{

get

{

return \_geneticParameters.InitialPopulationSize;

}

set

{

\_geneticParameters.InitialPopulationSize = value;

NotifyPropertyChanged(()=>InitialPopulationSize);

}

}

public int MaxGridValue

{

get

{

return GenerationCount;

}

}

public int GenerationCount

{

get

{

return \_geneticParameters.GenerationCount;

}

set

{

\_geneticParameters.GenerationCount = value;

NotifyPropertyChanged(() => GenerationCount);

}

}

private ICommand \_startGeneticAlgorithmCommand;

public ICommand StartGeneticAlgorithmCommand

{

get

{

return \_startGeneticAlgorithmCommand;

}

set

{

\_startGeneticAlgorithmCommand = value;

NotifyPropertyChanged(() => StartGeneticAlgorithmCommand);

}

}

public int ReproductionNumber

{

get

{

return \_geneticParameters.ReproductionNumber;

}

set

{

\_geneticParameters.ReproductionNumber = value;

NotifyPropertyChanged(() => ReproductionNumber);

}

}

public double MutationProbability

{

get

{

return \_geneticParameters.MutationProbability;

}

set

{

\_geneticParameters.MutationProbability = value;

NotifyPropertyChanged(() => MutationProbability);

}

}

public double GoodOrganizmSurvivalProbability

{

get

{

return \_geneticParameters.GoodOrganizmSurvivalProbability;

}

set

{

\_geneticParameters.GoodOrganizmSurvivalProbability = value;

NotifyPropertyChanged(() => GoodOrganizmSurvivalProbability);

}

}

public double BadOrganizmDeathProbability

{

get

{

return \_geneticParameters.BadOrganizmDeathProbability;

}

set

{

\_geneticParameters.BadOrganizmDeathProbability = value;

NotifyPropertyChanged(() => BadOrganizmDeathProbability);

}

}

private int \_gridStep;

public int GridStep

{

get

{

return \_gridStep;

}

set

{

\_gridStep = value;

NotifyPropertyChanged(() => GridStep);

}

}

private PointsCollection \_averageFitness = new PointsCollection();

public PointsCollection AverageFitness

{

get

{

return \_averageFitness;

}

set

{

\_averageFitness = value;

NotifyPropertyChanged(() => AverageFitness);

}

}

private PointsCollection \_minFitness = new PointsCollection();

public PointsCollection MinFitness

{

get

{

return \_minFitness;

}

set

{

\_minFitness = value;

NotifyPropertyChanged(() => MinFitness);

}

}

private PointsCollection \_maxFitness = new PointsCollection();

public PointsCollection MaxFitness

{

get

{

return \_maxFitness;

}

set

{

\_maxFitness = value;

NotifyPropertyChanged(() => MaxFitness);

}

}

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

public class GeneticAlgoritmParameters

{

public int InitialPopulationSize { get; set; } = 50;

public int GenerationCount { get; set; } = 100;

public int ReproductionNumber { get; set; } = 20;

public double MutationProbability { get; set; } = 0.05;

public double GoodOrganizmSurvivalProbability { get; set; } = 0.99;

public double BadOrganizmDeathProbability { get; set; } = 0.95;

public GeneticAlgoritmParameters Clone()

{

return new GeneticAlgoritmParameters()

{

InitialPopulationSize = this.InitialPopulationSize,

GenerationCount = this.GenerationCount,

ReproductionNumber = this.ReproductionNumber,

MutationProbability = this.MutationProbability,

GoodOrganizmSurvivalProbability = this.GoodOrganizmSurvivalProbability,

BadOrganizmDeathProbability = this.BadOrganizmDeathProbability,

};

}

}

}

﻿using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.Model

{

public class Graph

{

public Graph(int vertexCount, GraphOrientationType orientation = GraphOrientationType.NotOriented)

{

\_adjacencyMatrix = new double[vertexCount][];

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

{

\_adjacencyMatrix[i] = new double[vertexCount];

}

\_graphOrientationType = orientation;

}

public Graph(double[][] adjacencyMatrix,GraphOrientationType orientation = GraphOrientationType.NotOriented):this(adjacencyMatrix.Length, orientation)

{

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

{

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

{

\_adjacencyMatrix[i][j] = adjacencyMatrix[i][j];

}

}

}

private GraphOrientationType \_graphOrientationType;

private double[][] \_adjacencyMatrix;

public void AddEdge(int vertexFrom, int vertexTo, int weight)

{

\_adjacencyMatrix[vertexFrom][vertexTo] = weight;

if (\_graphOrientationType == GraphOrientationType.NotOriented)

{

\_adjacencyMatrix[vertexTo][vertexFrom] = weight;

}

}

public int VertexCount

{

get

{

return \_adjacencyMatrix.Length;

}

}

public int CalculateVertexDegree(int vertexNumber)

{

int result = \_adjacencyMatrix[vertexNumber].Count((weight) => weight > 0);

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

{

result += \_adjacencyMatrix[i][i] > 0 ? 1 : 0;

}

return result;

}

public Graph GetSubGraph(bool[] edgesToInclude)

{

int edgeCounter = 0;

double[][] subGraphAdjancencyMatrix = new double[VertexCount][];

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

{

subGraphAdjancencyMatrix[i] = new double[VertexCount];

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

{

if (\_adjacencyMatrix[i][j] > 0)

{

if (edgesToInclude[edgeCounter])

{

subGraphAdjancencyMatrix[i][j] = \_adjacencyMatrix[i][j];

}

edgeCounter++;

}

}

}

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

{

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

{

if (\_adjacencyMatrix[i][j] > 0)

{

subGraphAdjancencyMatrix[j][i] = \_adjacencyMatrix[i][j];

}

}

}

return new Graph(subGraphAdjancencyMatrix);

}

public int GetNumberOfEdges()

{

int result = 0;

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

{

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

{

if (\_adjacencyMatrix[i][j] > 0)

{

result++;

}

}

}

return result;

}

public static Graph FromFile(string filePath)

{

using (StreamReader graphReader = new StreamReader(new FileStream(filePath, FileMode.Open)))

{

var graphLine = graphReader.ReadLine().TrimEnd();

var graphLineValues = graphLine.Split(' ');

var vertexCount = graphLineValues.Length;

var graphMatrix = new double[vertexCount][];

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

{

if (i != 0)

{

graphLine = graphReader.ReadLine().TrimEnd();

graphLineValues = graphLine.Split(' ');

}

graphMatrix[i] = new double[vertexCount];

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

{

graphMatrix[i][j] = Convert.ToDouble(graphLineValues[j]);

}

}

return new Graph(graphMatrix);

}

}

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.Model

{

public enum GraphOrientationType

{

Oriented,

NotOriented

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ASTU.GeneticAlgorithm

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

this.DataContext = new GeneticAlgorithmParametersViewModel();

}

}

}

﻿using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Linq.Expressions;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

public abstract class Observable : INotifyPropertyChanged

{

public Observable()

{

}

#region INotifyPropertyChanged Members

public event PropertyChangedEventHandler PropertyChanged;

#endregion

public static string GetPropertyName<TProperty>(Expression<Func<TProperty>> propertyLambda)

{

MemberExpression member = propertyLambda.Body as MemberExpression;

if (member == null)

throw new ArgumentException(string.Format(

"Expression '{0}' refers to a method, not a property.",

propertyLambda.ToString()));

PropertyInfo propInfo = member.Member as PropertyInfo;

if (propInfo == null)

throw new ArgumentException(string.Format(

"Expression '{0}' refers to a field, not a property.",

propertyLambda.ToString()));

return propInfo.Name;

}

public void NotifyPropertyChanged<TProperty>(Expression<Func<TProperty>> propertyLambda)

{

NotifyPropertyChanged(GetPropertyName(propertyLambda));

}

protected void NotifyPropertyChanged(string name)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

}

}

﻿using ASTU.Model;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

public class Organizm

{

public Organizm(Graph graph)

{

\_organizmBits = new bool[graph.GetNumberOfEdges()];

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

{

\_organizmBits[i] = RandomHelper.NextBool();

}

\_graph = graph;

}

public Organizm(Graph graph,bool[] organizmBits)

{

\_organizmBits = organizmBits;

\_graph = graph;

}

public Organizm(Graph graph, bool[] firstParentGenes, bool[] secondParentGenes):this(graph)

{

\_organizmBits = ConcatGenes(firstParentGenes, secondParentGenes);

}

public double MeasureFitness(Graph graph)

{

double penaltyFactor = \_graph.VertexCount;

double vertexScoreFactor = 1;

var penalty = \_organizmBits.Any(bit => bit)?0:-1;

var organizmGraph = \_graph.GetSubGraph(\_organizmBits);

var organizmScore = penalty \* penaltyFactor;

for (int i = 0; i < organizmGraph.VertexCount; i++)

{

var vertexDegree = organizmGraph.CalculateVertexDegree(i);

if (vertexDegree == 0 || vertexDegree == 3)

{

organizmScore += 1 \* vertexScoreFactor;

}

else

{

if (vertexDegree > 3)

{

organizmScore -= vertexDegree \* vertexScoreFactor;

}

}

}

return organizmScore;

}

public Organizm Mutate(Graph graph)

{

var mutantBits = new bool[\_organizmBits.Length];

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

{

mutantBits[i] = !\_organizmBits[i];

}

return new Organizm(graph,mutantBits);

}

private bool[] ConcatGenes(bool[] first, bool[] after)

{

var bools = new bool[first.Length + after.Length];

first.CopyTo(bools, 0);

after.CopyTo(bools, first.Length);

return bools;

}

private Tuple<bool[], bool[]> SliceOrganizm(int cutPoint)

{

var parent = \_organizmBits;

bool[] firstOrganizm = parent.Take(cutPoint).ToArray();

bool[] secondOrganizm = parent.Skip(cutPoint).Take(parent.Length - cutPoint).ToArray();

return new Tuple<bool[], bool[]>(firstOrganizm, secondOrganizm);

}

private bool[] \_organizmBits;

private Graph \_graph;

private int GenesCount

{

get

{

return \_organizmBits.Length;

}

}

public static Tuple<Organizm, Organizm> ProduceChildren(Graph graph,Organizm firstParent, Organizm secondParent)

{

int cutLine = RandomHelper.NextInt(firstParent.GenesCount);

var cuttedGenesParent1 = firstParent.SliceOrganizm(cutLine);

var cuttedGenesParent2 = secondParent.SliceOrganizm(cutLine);

Organizm child1 = new Organizm(graph, cuttedGenesParent1.Item1, cuttedGenesParent2.Item2);

Organizm child2 = new Organizm(graph, cuttedGenesParent2.Item1, cuttedGenesParent1.Item2);

return new Tuple<Organizm, Organizm>(child1,child2);

}

}

}

﻿namespace ASTU.GeneticAlgorithm

{

internal class PopulationHistoryItem

{

public double BestOrganismFitness { get; set; }

public double AverageOrganismFitness { get; set; }

public double WorstOrganismFitness { get; set; }

public int Generation { get; set;}

}

}﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ASTU.GeneticAlgorithm

{

public static class RandomHelper

{

private static Random randomGenerator = new Random();

public static int NextInt(int max)

{

return randomGenerator.Next(max);

}

public static bool NextBool()

{

return Convert.ToBoolean(randomGenerator.Next(2));

}

public static double NextDouble()

{

return randomGenerator.NextDouble();

}

}

}

<Window x:Class="ASTU.GeneticAlgorithm.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:ASTU.GeneticAlgorithm"

xmlns:chart="clr-namespace:Xceed.Wpf.Toolkit.Chart;assembly=Xceed.Wpf.Toolkit"

xmlns:toolkit="clr-namespace:Xceed.Wpf.Toolkit;assembly=Xceed.Wpf.Toolkit"

xmlns:sparrow="http://sparrowtoolkit.codeplex.com/wpf"

mc:Ignorable="d"

Title="Генетический алгоритм" Height="350" Width="525">

<Grid>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="250"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="20"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="20"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="50"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Число организмов" Grid.Row="0" Grid.Column="0"></TextBlock>

<toolkit:IntegerUpDown Value="{Binding InitialPopulationSize}" Grid.Row="0" Grid.Column="1"></toolkit:IntegerUpDown>

<TextBlock Text="Число поколений" Grid.Row="1" Grid.Column="0"></TextBlock>

<toolkit:IntegerUpDown Value="{Binding GenerationCount }" Grid.Row="1" Grid.Column="1"></toolkit:IntegerUpDown>

<TextBlock Text="Количество воспроизводств" Grid.Row="2" Grid.Column="0"></TextBlock>

<toolkit:IntegerUpDown Value="{Binding ReproductionNumber }" Grid.Row="2" Grid.Column="1"></toolkit:IntegerUpDown>

<TextBlock Text="P мутации" Grid.Row="3" Grid.Column="0"></TextBlock>

<toolkit:DoubleUpDown Increment="0.01" Value="{Binding MutationProbability }" Grid.Row="3" Grid.Column="1"></toolkit:DoubleUpDown>

<TextBlock Text="P выживания приспособленного" Grid.Row="4" Grid.Column="0"></TextBlock>

<toolkit:DoubleUpDown Increment="0.01" Value="{Binding GoodOrganizmSurvivalProbability }" Grid.Row="4" Grid.Column="1"></toolkit:DoubleUpDown>

<TextBlock Text="P гибели неприспособленного" Grid.Row="5" Grid.Column="0"></TextBlock>

<toolkit:DoubleUpDown Increment="0.01" Value="{Binding BadOrganizmDeathProbability }" Grid.Row="5" Grid.Column="1"></toolkit:DoubleUpDown>

<Button Grid.Row="6" Grid.ColumnSpan="2" Command="{Binding StartGeneticAlgorithmCommand}" Content="Старт"/>

<!--<TextBlock Text="Шаг сетки графика" Grid.Row="8" Grid.Column="0"></TextBlock>

<toolkit:IntegerUpDown Value="{Binding GridStep}" Grid.Row="8" Grid.Column="1"></toolkit:IntegerUpDown>-->

<ProgressBar Grid.Row="8" Value="{Binding ProgressValue}"/>

</Grid>

<sparrow:SparrowChart Margin="20" Grid.Column="1" Theme="Rainbow" OverlayMode="SeriesFirst">

<sparrow:SparrowChart.Legend>

<sparrow:Legend Header="Организмы по поколениям" LegendPosition="Outside" HorizontalAlignment="Right" VerticalAlignment="Top" ShowIcon="True"></sparrow:Legend>

</sparrow:SparrowChart.Legend>

<sparrow:SparrowChart.XAxis>

<sparrow:LinearXAxis MaxValue="{Binding MaxGridValue}"/>

</sparrow:SparrowChart.XAxis>

<sparrow:SparrowChart.YAxis>

<sparrow:LinearYAxis />

</sparrow:SparrowChart.YAxis>

<sparrow:LineSeries Label="Худший" Points="{Binding MinFitness}"/>

<sparrow:LineSeries Label="Средний" Points="{Binding AverageFitness}"/>

<sparrow:LineSeries Label="Лучший" Points="{Binding MaxFitness}"/>

</sparrow:SparrowChart>

<!--<chart:Chart Grid.Column="1" x:Name="visualizationChart"></chart:Chart>-->

</Grid>

</Grid>

</Window>

# 2. Нейронные сети. Обучение методом обратного распространения ошибки

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

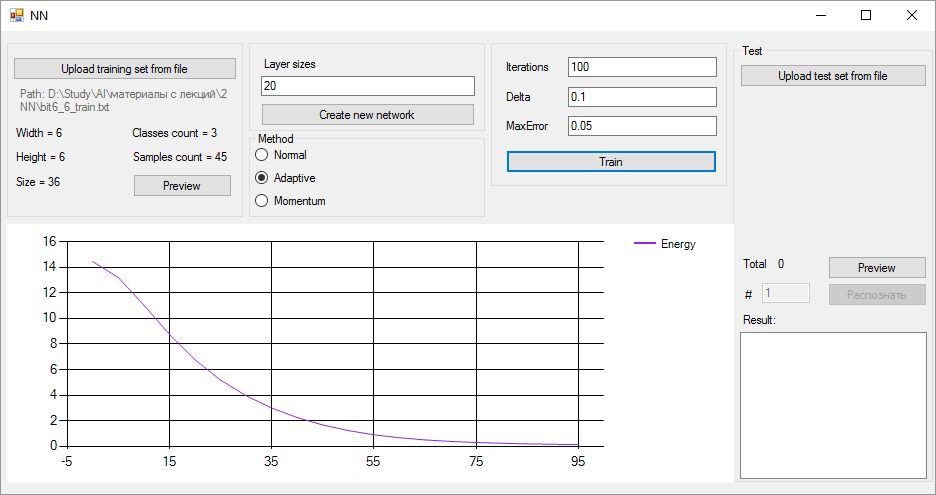
Результаты исследований:

В ходе экспериментов было обнаружено, что добавление bias к слоям существенно улучшают результаты обучения, на картинке обозначены +1



Кроме того, было обнаружено, что начальное значение весов имеет значение, так при смене диапазона случайных чисел от 0 до 1 на диапазон -0.5 до 0.5 сеть стала учиться значительно лучше.

График обучения битовых образов 6x6:



Тестовая выборка 6x6:

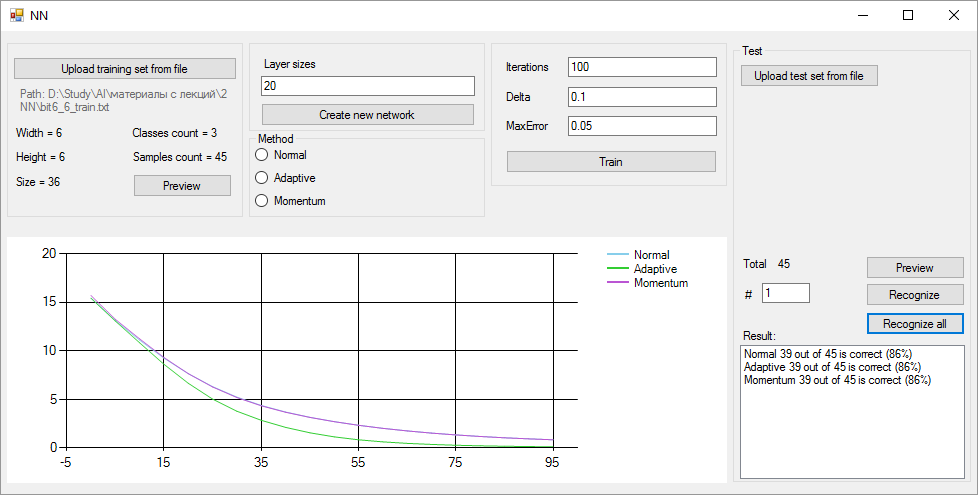


График обучения битовых цифр 16\*20:

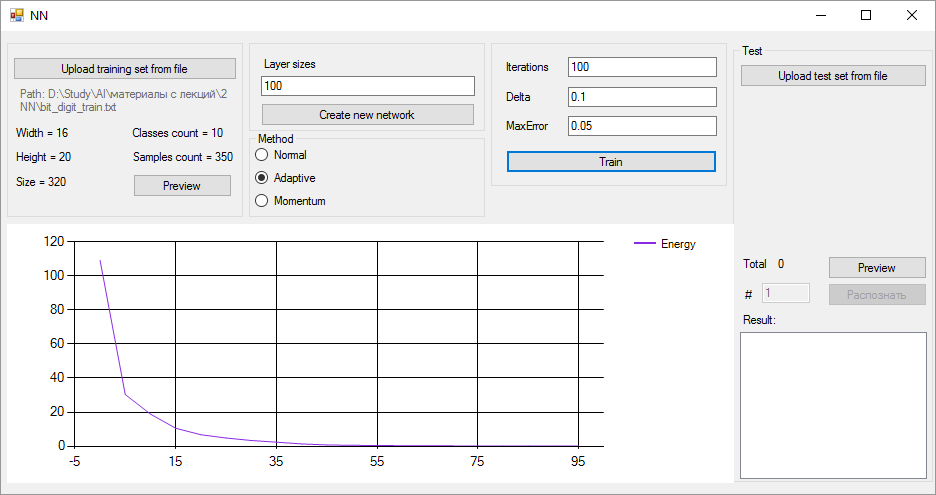
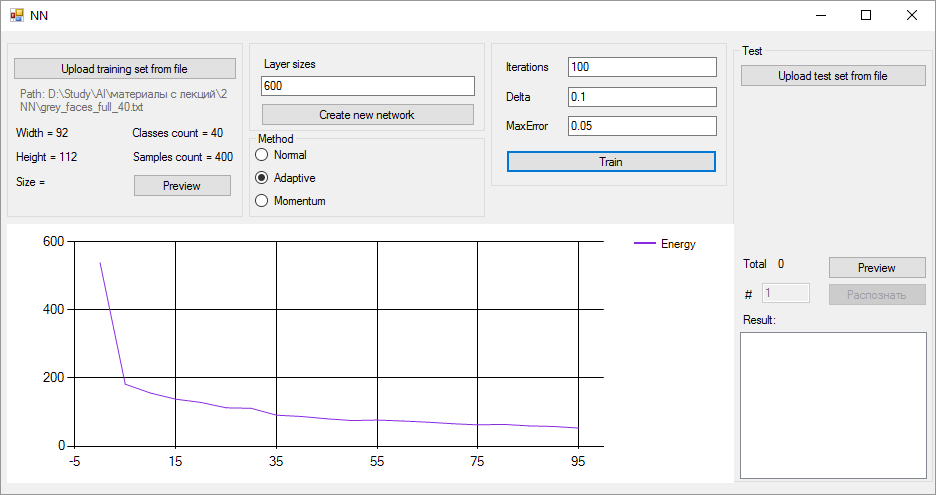
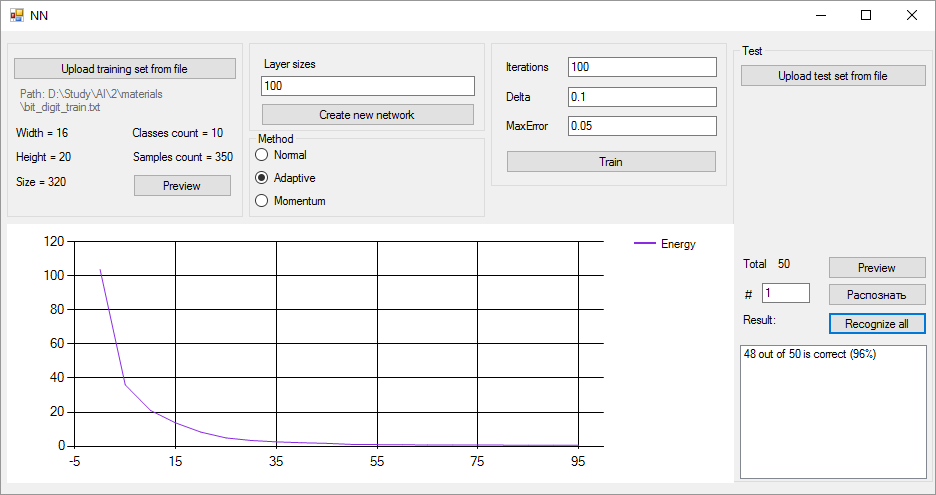


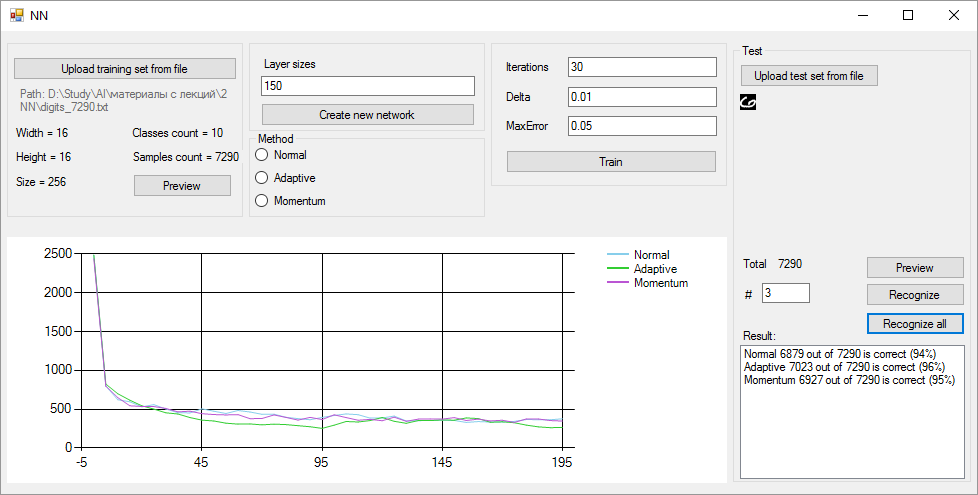
График обучения 40 классов серых лиц:



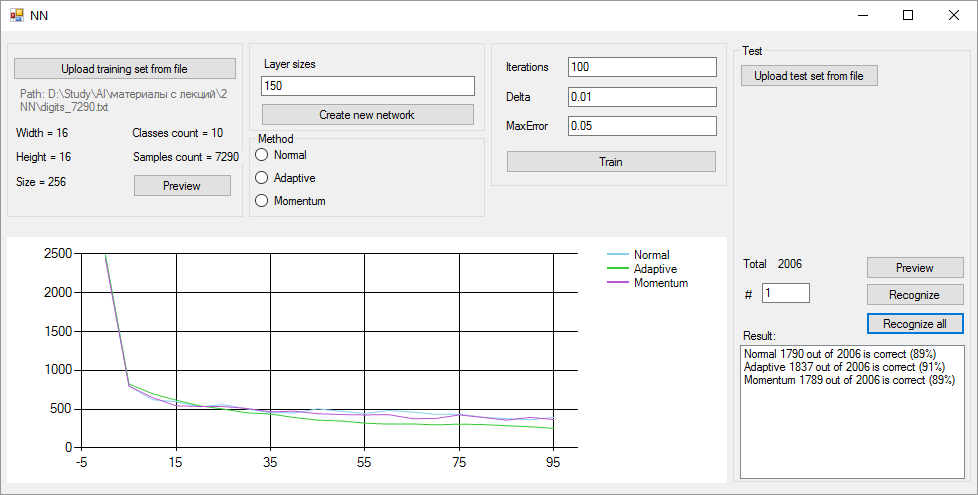
Результаты тестирования тестовой выборки битовых образов:



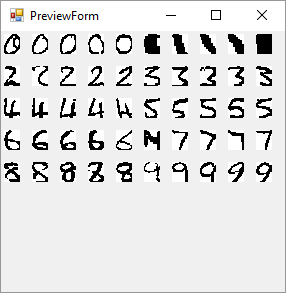
Серые цифры результат обучения 7290 образов:



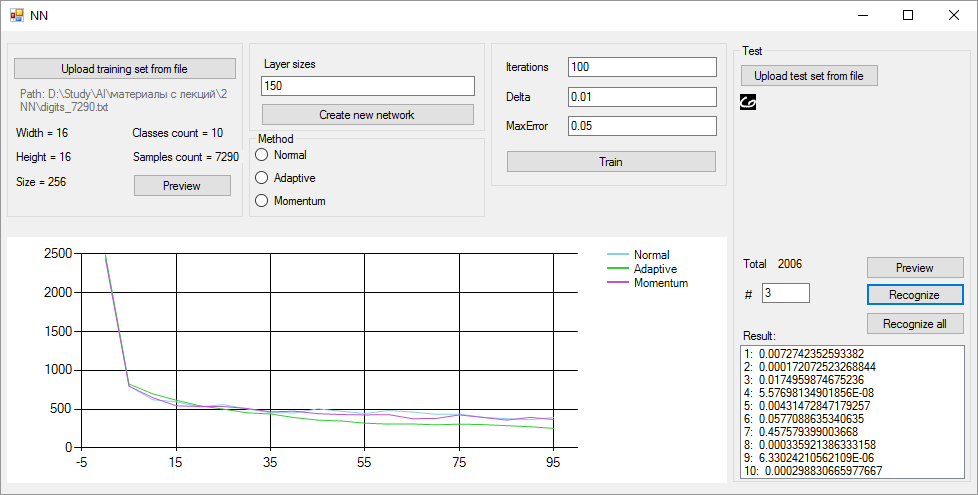
Серые цифры, результат тестирования на 2006 тестовых образах:



Предпросмотр тестовых данных:



Распознавание цифры 6:



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.Tasks;

using System.Windows.Forms;

namespace nn

{

public partial class MainForm : Form

{

private TrainingSet \_trainingSet;

private TrainingSet \_testSet;

private NeuralNetwork \_network;

private NeuralNetwork \_networkAdaptive;

private NeuralNetwork \_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)

{

}

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,true);

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)

{

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

chart1.Series[1].Points.Clear();

chart1.Series[2].Points.Clear();

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;

if (normalRadio.Checked)

{

\_network = new NeuralNetwork(\_trainingSet.SampleSize, networkParams);

\_network.Method = Method.Normal;

}

else if (adaptiveRadio.Checked)

{

\_networkAdaptive = new NeuralNetwork(\_trainingSet.SampleSize, networkParams);//\_network.Clone();

\_networkAdaptive.Method = Method.Adaptive;

}

else if (momentumRadio.Checked)

{

\_networkMomentum = new NeuralNetwork(\_trainingSet.SampleSize, networkParams);// \_network.Clone();

\_networkMomentum.Method = Method.Momentum;

}

else

{

\_network = new NeuralNetwork(\_trainingSet.SampleSize, networkParams);

\_network.Method = Method.Normal;

\_networkAdaptive = \_network.Clone();//new NeuralNetwork(\_trainingSet.SampleSize, networkParams);//\_network.Clone();

\_networkAdaptive.Method = Method.Adaptive;

\_networkMomentum = \_network.Clone();//new NeuralNetwork(\_trainingSet.SampleSize, networkParams);// \_network.Clone();

\_networkMomentum.Method = Method.Momentum;

}

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);

if (normalRadio.Checked || !normalRadio.Checked && !adaptiveRadio.Checked && !momentumRadio.Checked)

{

\_network.Delta = delta;

new TaskFactory().StartNew(() =>

\_network.Train(\_trainingSet,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[0],

5));

}

if (adaptiveRadio.Checked || !normalRadio.Checked && !adaptiveRadio.Checked && !momentumRadio.Checked)

{

\_networkAdaptive.Delta = delta;

new TaskFactory().StartNew(() =>

\_networkAdaptive.Train(\_trainingSet,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[1],

5));

}

if (momentumRadio.Checked || !normalRadio.Checked && !adaptiveRadio.Checked && !momentumRadio.Checked)

{

\_networkMomentum.Delta = delta;

new TaskFactory().StartNew(() =>

\_networkMomentum.Train(\_trainingSet,

iterationsCount,

e0,

\_trainingSet.SamplesCount,

chart1,

chart1.Series[2],

5));

}

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[\_indexToTest].Input[x \* \_trainingSet.SampleWidth + y];

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

}

else

{

int val = (int)\_testSet[\_indexToTest].Input[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[\_indexToTest].Input;

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

\_network.NetworkOut(netIn, out result);

listBox1.Items.Clear();

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

{

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

}

}

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();

}

}

private int GetMaxResultIndex(double[] result)

{

double max = double.MinValue;

int maxIndex = 0;

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

{

if (max < result[j])

{

max = result[j];

maxIndex = j;

}

}

return maxIndex;

}

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

{

int correctResult = 0;

int correctResultAdaptive = 0;

int correctResultMomentum = 0;

listBox1.Items.Clear();

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

{

var answers = \_testSet[i].Answer;

double answerMax = double.MinValue;

int maxAnswerIndex = 0;

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

{

if (answerMax < answers[j])

{

answerMax = answers[j];

maxAnswerIndex = j;

}

}

double[] netIn = \_testSet[i].Input;

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

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

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

\_network.NetworkOut(netIn, out result);

\_networkAdaptive.NetworkOut(netIn,out adaptiveResult);

\_networkMomentum.NetworkOut(netIn, out momentumResult);

;

if (GetMaxResultIndex(result) == maxAnswerIndex)

{

correctResult++;

}

if (GetMaxResultIndex(adaptiveResult) == maxAnswerIndex)

{

correctResultAdaptive++;

}

if (GetMaxResultIndex(momentumResult) == maxAnswerIndex)

{

correctResultMomentum++;

}

}

listBox1.Items.Add(String.Format($"Normal {correctResult} out of {\_testSet.SamplesCount} is correct ({(int)(1.0\*correctResult/\_testSet.SamplesCount\*100)}%)"));

listBox1.Items.Add(String.Format($"Adaptive {correctResultAdaptive} out of {\_testSet.SamplesCount} is correct ({(int)(1.0 \* correctResultAdaptive / \_testSet.SamplesCount \* 100)}%)"));

listBox1.Items.Add(String.Format($"Momentum {correctResultMomentum} out of {\_testSet.SamplesCount} is correct ({(int)(1.0 \* correctResultMomentum / \_testSet.SamplesCount \* 100)}%)"));

}

}

}

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

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.groupBox5 = new System.Windows.Forms.GroupBox();

this.momentumRadio = new System.Windows.Forms.RadioButton();

this.adaptiveRadio = new System.Windows.Forms.RadioButton();

this.normalRadio = new System.Windows.Forms.RadioButton();

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.groupBox5.SuspendLayout();

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(6, 6);

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(248, 6);

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(490, 6);

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 = "0.01";

//

// 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(732, 13);

this.groupBox4.Name = "groupBox4";

this.groupBox4.Size = new System.Drawing.Size(238, 439);

this.groupBox4.TabIndex = 4;

this.groupBox4.TabStop = false;

this.groupBox4.Text = "Test";

//

// button7

//

this.button7.Location = new System.Drawing.Point(133, 268);

this.button7.Name = "button7";

this.button7.Size = new System.Drawing.Size(99, 23);

this.button7.TabIndex = 11;

this.button7.Text = "Recognize all";

this.button7.UseVisualStyleBackColor = true;

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

//

// button6

//

this.button6.Location = new System.Drawing.Point(133, 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, 285);

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, 301);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(225, 134);

this.listBox1.TabIndex = 3;

//

// button4

//

this.button4.Enabled = false;

this.button4.Location = new System.Drawing.Point(133, 239);

this.button4.Name = "button4";

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

this.button4.TabIndex = 2;

this.button4.Text = "Recognize";

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(225, 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(139, 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(6, 206);

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(720, 246);

this.chart1.TabIndex = 6;

this.chart1.Text = "chart1";

//

// groupBox5

//

this.groupBox5.Controls.Add(this.momentumRadio);

this.groupBox5.Controls.Add(this.adaptiveRadio);

this.groupBox5.Controls.Add(this.normalRadio);

this.groupBox5.Location = new System.Drawing.Point(248, 101);

this.groupBox5.Name = "groupBox5";

this.groupBox5.Size = new System.Drawing.Size(236, 86);

this.groupBox5.TabIndex = 7;

this.groupBox5.TabStop = false;

this.groupBox5.Text = "Method";

//

// momentumRadio

//

this.momentumRadio.AutoSize = true;

this.momentumRadio.Location = new System.Drawing.Point(6, 60);

this.momentumRadio.Name = "momentumRadio";

this.momentumRadio.Size = new System.Drawing.Size(77, 17);

this.momentumRadio.TabIndex = 0;

this.momentumRadio.TabStop = true;

this.momentumRadio.Text = "Momentum";

this.momentumRadio.UseVisualStyleBackColor = true;

//

// adaptiveRadio

//

this.adaptiveRadio.AutoSize = true;

this.adaptiveRadio.Location = new System.Drawing.Point(6, 37);

this.adaptiveRadio.Name = "adaptiveRadio";

this.adaptiveRadio.Size = new System.Drawing.Size(67, 17);

this.adaptiveRadio.TabIndex = 0;

this.adaptiveRadio.TabStop = true;

this.adaptiveRadio.Text = "Adaptive";

this.adaptiveRadio.UseVisualStyleBackColor = true;

//

// normalRadio

//

this.normalRadio.AutoSize = true;

this.normalRadio.Location = new System.Drawing.Point(6, 14);

this.normalRadio.Name = "normalRadio";

this.normalRadio.Size = new System.Drawing.Size(58, 17);

this.normalRadio.TabIndex = 0;

this.normalRadio.TabStop = true;

this.normalRadio.Text = "Normal";

this.normalRadio.UseVisualStyleBackColor = true;

//

// MainForm

//

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

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

this.ClientSize = new System.Drawing.Size(976, 463);

this.Controls.Add(this.groupBox5);

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.groupBox5.ResumeLayout(false);

this.groupBox5.PerformLayout();

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.GroupBox groupBox5;

private System.Windows.Forms.RadioButton momentumRadio;

private System.Windows.Forms.RadioButton adaptiveRadio;

private System.Windows.Forms.RadioButton normalRadio;

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.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[i].Input[x \* \_trainingSet.SampleWidth + y];

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

}

else

{

int val = (int)\_trainingSet[i].Input[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;

}

}

}

}

﻿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 System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace nn

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

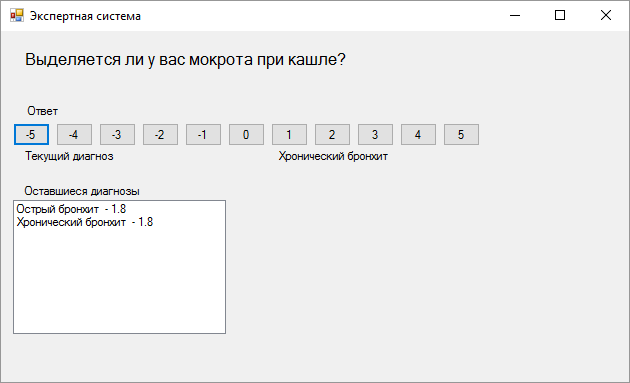
}

}

}

# 3. Экспертные системы. Правило Байеса.

Результаты работы программы



Исходный код

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;

using System.IO;

namespace ExpertSystems

{

public partial class Form1 : Form

{

const int MAXM = 100;

string[] questions; // вопросы

string[] disease; // болезни

double[] probDis; // вероятности болезней

double[,] table; // таблица частотных зависимостей

bool[] flagBoln;

bool[] flagSimpt;

double[] pmax;

double[] pmin;

bool flagEnd;

int simptomsCount; // реальное число вопросов

int deseaseCount; // число болезней

int deseaseSimptoms; // число записей таблицы

int thisQuest;

public Form1()

{

InitializeComponent();

}

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

{

questions = new string[MAXM];

disease = new string[MAXM];

probDis = new double[MAXM];

table = new double[MAXM, 4];

flagBoln = new bool[MAXM];

flagSimpt = new bool[MAXM];

flagEnd = true;

pmax = new double[MAXM];

pmin = new double[MAXM];

readData();

getQuestion();

}

private void readData()

{

StreamReader sr = new StreamReader("simptoms.txt");

string line;

simptomsCount = 0;

while ((line = sr.ReadLine()) != null)

{

questions[simptomsCount++] = line;

}

sr.Close();

sr = new StreamReader("var.txt");

deseaseCount = 0;

while ((line = sr.ReadLine()) != "")

{

string[] tableItems = line.Split(' ');

probDis[deseaseCount] = Convert.ToDouble(tableItems[tableItems.Length - 1]);

for (int i = 0; i < tableItems.Length - 2; i++)

disease[deseaseCount] = disease[deseaseCount] + tableItems[i] + " ";

listBox1.Items.Add(disease[deseaseCount] + " - " + probDis[deseaseCount].ToString());

deseaseCount++;

}

deseaseSimptoms = 0;

while ((line = sr.ReadLine()) != null)

{

if (line == "") continue;

string[] tableItems = line.Split(' ');

for (int i = 0; i < 4; i++)

{

if (i == 0 || i == 1)

{

table[deseaseSimptoms, i] = Convert.ToDouble(tableItems[i])-1;

}

else

{

table[deseaseSimptoms, i] = Convert.ToDouble(tableItems[i]);

}

}

deseaseSimptoms++;

}

sr.Close();

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

{

flagSimpt[i] = true;

}

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

{

flagBoln[i] = true;

}

}

//1

private void getQuestion()

{

int maxCnt = 0;

int numMax = 0;

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

{

if (!flagSimpt[i])

{

continue;

}

int iCnt = 0;

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

if (Convert.ToInt32(table[j, 1]) == i && flagBoln[Convert.ToInt32(table[j, 0])])

iCnt++;

if (iCnt == 0)

flagSimpt[i] = false;

if (iCnt > maxCnt)

{

maxCnt = iCnt;

numMax = i;

}

}

label1.Text = questions[numMax];

thisQuest = numMax;

flagSimpt[thisQuest] = false;

}

private void answerQuestion(int res)

{

//3

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

{

if (!flagBoln[i])

continue;

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

{

if (Convert.ToInt32(table[j, 0]) != i)

continue;

if (Convert.ToInt32(table[j, 1]) != thisQuest)

continue;

// связан с данным вопросом

double py = table[j, 2];

double pn = table[j, 3];

double pc = probDis[i];

double ppl = py \* pc / (py \* pc + pn \* (1 - pc));

double pm = (1 - py) \* pc / ((1 - py) \* pc + (1 - pn) \* (1 - pc));

if (res == 5)

{

probDis[i] = ppl;

}

else if (res == -5)

{

probDis[i] = pm;

}

else if (res > 0)

probDis[i] = res \* (ppl - pc) / 5 + pc;

else if (res < 0)

probDis[i] = res \* (pc - pm) / 5 + pc;

}

}

//4

double maxVal = -1;

int maxDiag = -1;

double secondVal = -1;

int secondDiag = -1;

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

{

if (flagBoln[i])

{

if (probDis[i]>=maxVal)

{

secondVal = maxVal;

secondDiag = maxDiag;

maxVal = probDis[i];

maxDiag = i;

}

}

}

label4.Text = disease[maxDiag].ToString();

//5

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

{

if (!flagBoln[i])

{

continue;

}

pmax[i] = probDis[i];

pmin[i] = probDis[i];

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

{

if (Convert.ToInt32(table[j, 0]) != i)

{

continue;

}

if (Convert.ToInt32(table[j, 1]) != thisQuest)

{

continue;

}

double py = table[j, 2];

double pn = table[j, 3];

double pc = pmax[i];

double pmaxpl = py \* pc / (py \* pc + pn \* (1 - pc));

double pmaxm = (1 - py) \* pc / ((1 - py) \* pc + (1 - pn) \* (1 - pc));

pc = pmin[i];

double pminpl = py \* pc / (py \* pc + pn \* (1 - pc));

double pminm = (1 - py) \* pc / ((1 - py) \* pc + (1 - pn) \* (1 - pc));

pmax[i] = Math.Max(pmax[i], Math.Max(pmaxpl, Math.Max(pminpl, Math.Max(pmaxm, pminm))));

pmin[i] = Math.Min(pmin[i], Math.Min(pmaxpl, Math.Min(pminpl, Math.Min(pmaxm, pminm))));

}

}

//Исключаем болезни

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

{

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

{

if (flagBoln[i] && flagBoln[j] && pmax[i] < pmin[j])

{

flagBoln[i] = false;

}

}

}

int num = 0;

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

{

if (flagBoln[i])

{

num++;

}

}

if (num == 1)

{

flagEnd = false;

}

num = 0;

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

{

if (!flagSimpt[i])

{

continue;

}

int iCnt = 0;

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

{

if (Convert.ToInt32(table[j, 1]) == i && flagBoln[Convert.ToInt32(table[j, 0])])

{

iCnt++;

}

}

if (iCnt == 0)

{

flagSimpt[i] = false;

}

else

{

num++;

}

}

if (num == 0)

{

flagEnd = false;

}

//Оставим только действующие болезни

listBox1.Items.Clear();

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

{

if (flagBoln[i])

{

if (!double.IsNaN(Math.Round(probDis[i], 6)))

{

listBox1.Items.Add(disease[i] + " - " + Math.Round(probDis[i], 6).ToString());

}

}

}

if (!flagEnd)

{

MessageBox.Show("Диагноз" +disease[maxDiag].ToString()+" "+maxVal.ToString());

if (secondVal != -1)

{

MessageBox.Show("Альтернативный Диагноз: " + disease[secondDiag].ToString() + " " + secondVal.ToString());

}

this.Close();

}

else

{

getQuestion();

}

}

private void answerClick(object sender, EventArgs e)

{

var answer = int.Parse((sender as Button).Text);

answerQuestion(answer);

}

}

}

﻿namespace ExpertSystems

{

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

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

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

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

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

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

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

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

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

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

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

this.button8 = new System.Windows.Forms.Button();

this.button9 = new System.Windows.Forms.Button();

this.button10 = new System.Windows.Forms.Button();

this.button11 = new System.Windows.Forms.Button();

this.button12 = new System.Windows.Forms.Button();

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

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label1.Location = new System.Drawing.Point(20, 18);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(64, 20);

this.label1.TabIndex = 1;

this.label1.Text = "Вопрос";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(21, 118);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(96, 13);

this.label2.TabIndex = 3;

this.label2.Text = "Текущий диагноз";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(21, 153);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(122, 13);

this.label3.TabIndex = 4;

this.label3.Text = "Оставшиеся диагнозы";

//

// listBox1

//

this.listBox1.FormattingEnabled = true;

this.listBox1.Location = new System.Drawing.Point(12, 169);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(213, 134);

this.listBox1.TabIndex = 5;

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(275, 118);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(0, 13);

this.label4.TabIndex = 6;

//

// button2

//

this.button2.Location = new System.Drawing.Point(12, 92);

this.button2.Name = "button2";

this.button2.Size = new System.Drawing.Size(37, 23);

this.button2.TabIndex = 10;

this.button2.Text = "-5";

this.button2.UseVisualStyleBackColor = true;

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

//

// button3

//

this.button3.Location = new System.Drawing.Point(55, 92);

this.button3.Name = "button3";

this.button3.Size = new System.Drawing.Size(37, 23);

this.button3.TabIndex = 10;

this.button3.Text = "-4";

this.button3.UseVisualStyleBackColor = true;

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

//

// button4

//

this.button4.Location = new System.Drawing.Point(98, 92);

this.button4.Name = "button4";

this.button4.Size = new System.Drawing.Size(37, 23);

this.button4.TabIndex = 10;

this.button4.Text = "-3";

this.button4.UseVisualStyleBackColor = true;

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

//

// button5

//

this.button5.Location = new System.Drawing.Point(141, 92);

this.button5.Name = "button5";

this.button5.Size = new System.Drawing.Size(37, 23);

this.button5.TabIndex = 10;

this.button5.Text = "-2";

this.button5.UseVisualStyleBackColor = true;

this.button5.Click += new System.EventHandler(this.answerClick);

//

// button6

//

this.button6.Location = new System.Drawing.Point(184, 92);

this.button6.Name = "button6";

this.button6.Size = new System.Drawing.Size(37, 23);

this.button6.TabIndex = 10;

this.button6.Text = "-1";

this.button6.UseVisualStyleBackColor = true;

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

//

// button7

//

this.button7.Location = new System.Drawing.Point(227, 92);

this.button7.Name = "button7";

this.button7.Size = new System.Drawing.Size(37, 23);

this.button7.TabIndex = 10;

this.button7.Text = "0";

this.button7.UseVisualStyleBackColor = true;

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

//

// button8

//

this.button8.Location = new System.Drawing.Point(270, 92);

this.button8.Name = "button8";

this.button8.Size = new System.Drawing.Size(37, 23);

this.button8.TabIndex = 10;

this.button8.Text = "1";

this.button8.UseVisualStyleBackColor = true;

this.button8.Click += new System.EventHandler(this.answerClick);

//

// button9

//

this.button9.Location = new System.Drawing.Point(313, 92);

this.button9.Name = "button9";

this.button9.Size = new System.Drawing.Size(37, 23);

this.button9.TabIndex = 10;

this.button9.Text = "2";

this.button9.UseVisualStyleBackColor = true;

this.button9.Click += new System.EventHandler(this.answerClick);

//

// button10

//

this.button10.Location = new System.Drawing.Point(356, 92);

this.button10.Name = "button10";

this.button10.Size = new System.Drawing.Size(37, 23);

this.button10.TabIndex = 10;

this.button10.Text = "3";

this.button10.UseVisualStyleBackColor = true;

this.button10.Click += new System.EventHandler(this.answerClick);

//

// button11

//

this.button11.Location = new System.Drawing.Point(399, 92);

this.button11.Name = "button11";

this.button11.Size = new System.Drawing.Size(37, 23);

this.button11.TabIndex = 10;

this.button11.Text = "4";

this.button11.UseVisualStyleBackColor = true;

this.button11.Click += new System.EventHandler(this.answerClick);

//

// button12

//

this.button12.Location = new System.Drawing.Point(442, 92);

this.button12.Name = "button12";

this.button12.Size = new System.Drawing.Size(37, 23);

this.button12.TabIndex = 10;

this.button12.Text = "5";

this.button12.UseVisualStyleBackColor = true;

this.button12.Click += new System.EventHandler(this.answerClick);

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(24, 73);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(37, 13);

this.label5.TabIndex = 11;

this.label5.Text = "Ответ";

//

// Form1

//

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

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

this.ClientSize = new System.Drawing.Size(628, 351);

this.Controls.Add(this.label5);

this.Controls.Add(this.button12);

this.Controls.Add(this.button11);

this.Controls.Add(this.button10);

this.Controls.Add(this.button9);

this.Controls.Add(this.button8);

this.Controls.Add(this.button7);

this.Controls.Add(this.button6);

this.Controls.Add(this.button5);

this.Controls.Add(this.button4);

this.Controls.Add(this.button3);

this.Controls.Add(this.button2);

this.Controls.Add(this.label4);

this.Controls.Add(this.listBox1);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Name = "Form1";

this.Text = "Экспертная система";

this.Load += new System.EventHandler(this.Form1\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.ListBox listBox1;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Button button2;

private System.Windows.Forms.Button button3;

private System.Windows.Forms.Button button4;

private System.Windows.Forms.Button button5;

private System.Windows.Forms.Button button6;

private System.Windows.Forms.Button button7;

private System.Windows.Forms.Button button8;

private System.Windows.Forms.Button button9;

private System.Windows.Forms.Button button10;

private System.Windows.Forms.Button button11;

private System.Windows.Forms.Button button12;

private System.Windows.Forms.Label label5;

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ExpertSystems

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

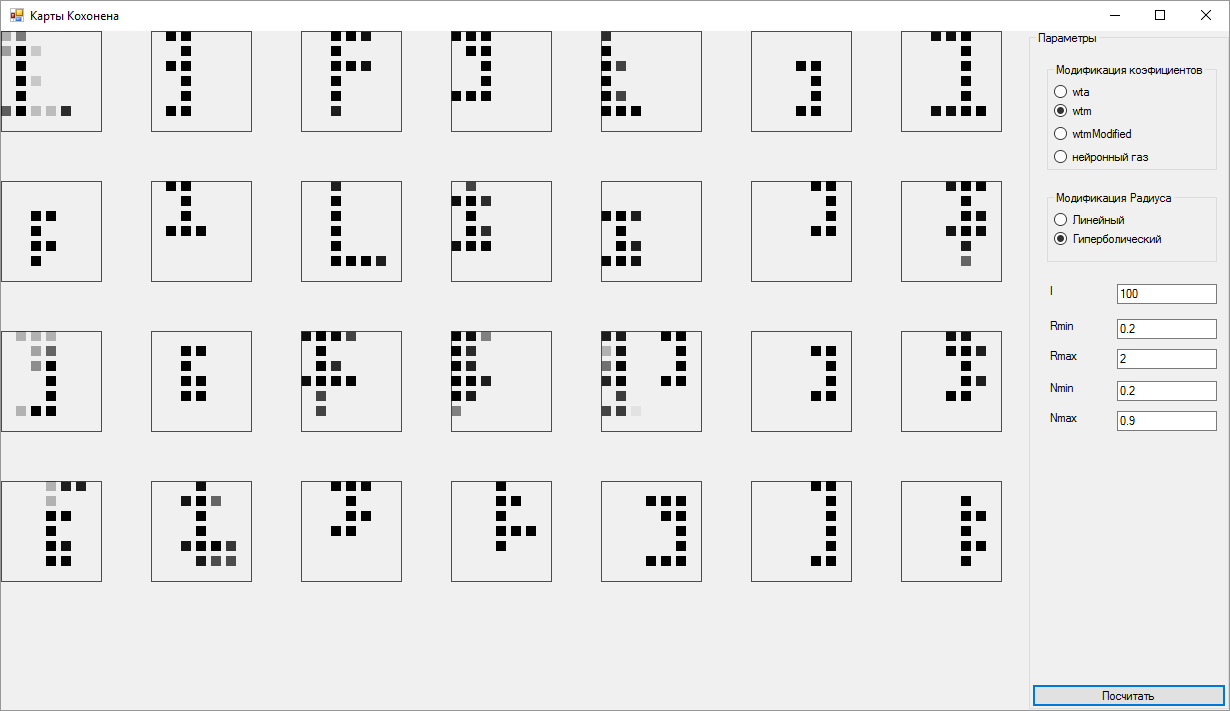
}

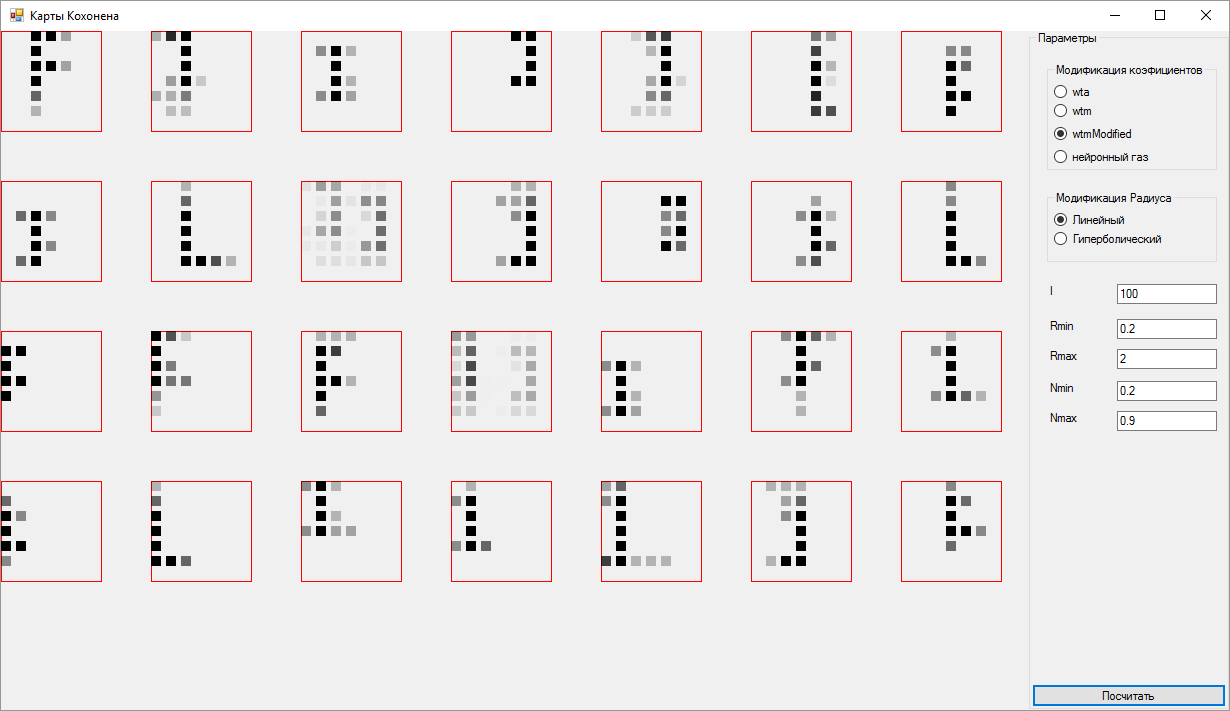
# 4. Карты Кохонена

Задание:

Построить карту Кохонена для символов F, L ,].

Результаты работы программы:





Исходный код:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace II6

{

public partial class Form1 : Form

{

// int M = 36; // 6х6

// int K = 3; // 3 класса

private int M;

private int N;

private int S;

private bool flag = false;

int I = 0;

double e0 = 0;

double delta = 0;

private double lambda = 1;

private double Nu = 0;

private double NuMin = 0;

private double NuMax = 0;

private double R = 1;

private double RMin = 0;

private double RMax = 0;

private double G = 1;

Brush blackBrush, greenBrush, redBrush;

private Pen blackPen, greenPen, redPen;

Graphics g;

private List<List<List<double>>> samples;

private List<int> type;//Тип экземпляра

private List<double> x;//Тип экземпляра

private List<double> y;//Тип экземпляра

private List<List<List<double>>> w;

public Form1()

{

InitializeComponent();

}

void generate()

{

x = new List<double>();

y = new List<double>();

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

Random rand = new Random();

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

{

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

{

x.Add(j);

y.Add(i);

w.Add(new List<List<double>>());

for (int k = 0; k < 6; k++)

{

w[i \* M + j].Add(new List<double>());

for (int t = 0; t < 6; t++)

{

w[i \* M + j][k].Add(rand.NextDouble());

}

}

// MessageBox.Show(j + " " + i);

}

}

}

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

{

this.Location = new Point(0, 0);

this.Width = SystemInformation.PrimaryMonitorSize.Width;

this.Height = SystemInformation.PrimaryMonitorSize.Height-50;

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

type = new List<int>();

string line;

int counter = 0;

int cntofelem = 0;

System.IO.StreamReader file =

new System.IO.StreamReader(@"test.txt");

while ((line = file.ReadLine()) != null)

{

if (counter == 0)

{

samples.Add(new List<List<double>>());

}

if (counter != 6)

{

samples[cntofelem].Add(new List<double>());

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

{

samples[cntofelem][counter].Add(Convert.ToInt32(line[i]) - 48);

}

counter++;

}

else

{

counter = 0;

type.Add(Convert.ToInt32(line[0]) - 48);

// MessageBox.Show(type[0].ToString());

cntofelem++;

}

}

g = this.CreateGraphics();

blackBrush = new SolidBrush(Color.Turquoise);

greenBrush = new SolidBrush(Color.SeaGreen);

redBrush = new SolidBrush(Color.Red);

blackPen = new Pen(blackBrush);

greenPen = new Pen(greenBrush);

redPen = new Pen(redBrush);

M = (this.Width-200)/150;

N = (this.Height)/150;

// MessageBox.Show(Width + " "+ Height);

// Invalidate();

// Update();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

if (flag)

{

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

{

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

{

g.DrawRectangle(redPen, j\*150, i\*150, 100, 100);

for (int k = 0; k < 6; k++)

{

for (int t = 0; t < 6; t++)

{

int brightness = (int) (255\*w[i\*M + j][k][t]);

Color semen = Color.FromArgb((int) brightness, Color.Black);

Brush temp = new SolidBrush(semen);

g.FillRectangle(temp, j\*150 + t\*15, i\*150 + k\*15, 10, 10);

}

}

}

}

}

}

double dist(int a, int b)

{

double ans = 0;

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

{

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

{

ans += (samples[a][i][j] - w[b][i][j])\*(samples[a][i][j] - w[b][i][j]);

}

}

ans = Math.Sqrt(ans);

return ans;

}

void modify(int a,int b)

{

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

{

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

{

//if (Nu == 4.88) MessageBox.Show(i.ToString() + " " + j.ToString());

// try

// {

w[a][i][j] = w[a][i][j] + (Nu\*G\*(samples[b][i][j] - w[a][i][j]));

// }

// catch (Exception e)

// {

// MessageBox.Show(Nu.ToString() + " " + a.ToString() + " " + b.ToString()+" "+i.ToString()+ " "+j.ToString());

//}

}

}

}

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

{

I = Convert.ToInt32(textBox1.Text);

RMin = Convert.ToDouble(textBox2.Text);

RMax = Convert.ToDouble(textBox3.Text);

R = RMax;

NuMin = Convert.ToDouble(textBox5.Text);

NuMax = Convert.ToDouble(textBox4.Text);

Nu = NuMax;

G = 1;

if (radioButton1.Checked == true)

{

R = 1;

}

generate();

for (int cnt = 0; cnt < I; cnt++)

{

for (int i = 0; i < samples.Count; i++)

{

G = 1;

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

int mini = -1;

double miniValue = double.MaxValue;

for (int j = 0; j < w.Count; j++)

{

double temp = dist(i, j);

distances.Add(temp);

if (temp < miniValue)

{

miniValue = temp;

mini = j;

}

}

if (radioButton4.Checked == false)

{

modify(mini,i);

//Также замодифицируем всех тех, кто в радиусе

for (int j = 0; j < w.Count; j++)

{

if (j != mini)

{

int x0 = mini%M;

int y0 = mini/M;

int x1 = j%M;

int y1 = j/M;

double neuronDist = (x0 - x1)\*(x0 - x1) + (y0 - y1)\*(y0 - y1);

neuronDist = Math.Sqrt(neuronDist);

if (neuronDist < R)

{

if (radioButton3.Checked == true)

{

double grade = -(miniValue - distances[j]) \* (miniValue - distances[j]);

grade /= lambda;

grade /= lambda;

G = Math.Exp(grade);

}

modify(j, i);

}

}

}

}

else

{

var aDictionary = new Dictionary<int, double>();

for (int j = 0; j < w.Count; j++)

{

aDictionary.Add(j,distances[j]);

}

int tempCnt = 1;

foreach (KeyValuePair<int,double> item in aDictionary.OrderBy(key=> key.Value))

{

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

modify(item.Key,i);

tempCnt++;

}

}

}

//Уменьшаем коэфициенты и радиус

if (radioButton8.Checked == true)

{

Nu = NuMax - (NuMax - NuMin)\*(cnt + 1)/I;

if (radioButton1.Checked == false)

{

R = RMax - (RMax - RMin) \* (double)(cnt + 1) /(double) I;

}

}

if (radioButton7.Checked == true)

{

Nu = NuMax \* Math.Pow( NuMin/NuMax,(double)(cnt + 1) / (double)I);

if (radioButton1.Checked == false)

{

R = RMax \* Math.Pow(RMin / RMax, (double)(cnt + 1) / (double)I);

}

}

}

flag = true;

Invalidate();

Update();

Thread. Sleep(500);

// flag = false;

}

}

}

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

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

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

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

this.textBox5 = new System.Windows.Forms.TextBox();

this.label5 = 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.textBox1 = new System.Windows.Forms.TextBox();

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

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

this.radioButton7 = new System.Windows.Forms.RadioButton();

this.radioButton8 = new System.Windows.Forms.RadioButton();

this.groupBox2 = 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.groupBox1.SuspendLayout();

this.groupBox3.SuspendLayout();

this.groupBox2.SuspendLayout();

this.SuspendLayout();

//

// groupBox1

//

this.groupBox1.Controls.Add(this.button1);

this.groupBox1.Controls.Add(this.textBox4);

this.groupBox1.Controls.Add(this.label4);

this.groupBox1.Controls.Add(this.textBox5);

this.groupBox1.Controls.Add(this.label5);

this.groupBox1.Controls.Add(this.textBox3);

this.groupBox1.Controls.Add(this.label3);

this.groupBox1.Controls.Add(this.textBox2);

this.groupBox1.Controls.Add(this.label2);

this.groupBox1.Controls.Add(this.textBox1);

this.groupBox1.Controls.Add(this.label1);

this.groupBox1.Controls.Add(this.groupBox3);

this.groupBox1.Controls.Add(this.groupBox2);

this.groupBox1.Dock = System.Windows.Forms.DockStyle.Right;

this.groupBox1.Location = new System.Drawing.Point(444, 0);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(200, 527);

this.groupBox1.TabIndex = 0;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Параметры";

//

// button1

//

this.button1.Dock = System.Windows.Forms.DockStyle.Bottom;

this.button1.Location = new System.Drawing.Point(3, 501);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(194, 23);

this.button1.TabIndex = 15;

this.button1.Text = "Посчитать";

this.button1.UseVisualStyleBackColor = true;

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

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(88, 380);

this.textBox4.Name = "textBox4";

this.textBox4.Size = new System.Drawing.Size(100, 20);

this.textBox4.TabIndex = 14;

this.textBox4.Text = "0.9";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(18, 380);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(34, 13);

this.label4.TabIndex = 13;

this.label4.Text = "Nmax";

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(88, 350);

this.textBox5.Name = "textBox5";

this.textBox5.Size = new System.Drawing.Size(100, 20);

this.textBox5.TabIndex = 12;

this.textBox5.Text = "0.2";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(18, 350);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(31, 13);

this.label5.TabIndex = 11;

this.label5.Text = "Nmin";

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(88, 318);

this.textBox3.Name = "textBox3";

this.textBox3.Size = new System.Drawing.Size(100, 20);

this.textBox3.TabIndex = 10;

this.textBox3.Text = "2";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(18, 318);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(34, 13);

this.label3.TabIndex = 9;

this.label3.Text = "Rmax";

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(88, 288);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(100, 20);

this.textBox2.TabIndex = 8;

this.textBox2.Text = "0.2";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(18, 288);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(31, 13);

this.label2.TabIndex = 7;

this.label2.Text = "Rmin";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(88, 253);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(100, 20);

this.textBox1.TabIndex = 6;

this.textBox1.Text = "100";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(18, 253);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(10, 13);

this.label1.TabIndex = 5;

this.label1.Text = "I";

//

// groupBox3

//

this.groupBox3.Controls.Add(this.radioButton7);

this.groupBox3.Controls.Add(this.radioButton8);

this.groupBox3.Location = new System.Drawing.Point(18, 160);

this.groupBox3.Name = "groupBox3";

this.groupBox3.Size = new System.Drawing.Size(170, 72);

this.groupBox3.TabIndex = 4;

this.groupBox3.TabStop = false;

this.groupBox3.Text = "Модификация Радиуса";

//

// radioButton7

//

this.radioButton7.AutoSize = true;

this.radioButton7.Location = new System.Drawing.Point(7, 39);

this.radioButton7.Name = "radioButton7";

this.radioButton7.Size = new System.Drawing.Size(114, 17);

this.radioButton7.TabIndex = 1;

this.radioButton7.TabStop = true;

this.radioButton7.Text = "Гиперболический";

this.radioButton7.UseVisualStyleBackColor = true;

//

// radioButton8

//

this.radioButton8.AutoSize = true;

this.radioButton8.Checked = true;

this.radioButton8.Location = new System.Drawing.Point(7, 20);

this.radioButton8.Name = "radioButton8";

this.radioButton8.Size = new System.Drawing.Size(77, 17);

this.radioButton8.TabIndex = 0;

this.radioButton8.TabStop = true;

this.radioButton8.Text = "Линейный";

this.radioButton8.UseVisualStyleBackColor = true;

//

// groupBox2

//

this.groupBox2.Controls.Add(this.radioButton4);

this.groupBox2.Controls.Add(this.radioButton3);

this.groupBox2.Controls.Add(this.radioButton2);

this.groupBox2.Controls.Add(this.radioButton1);

this.groupBox2.Location = new System.Drawing.Point(18, 32);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(170, 108);

this.groupBox2.TabIndex = 0;

this.groupBox2.TabStop = false;

this.groupBox2.Text = "Модификация коэфициентов";

//

// radioButton4

//

this.radioButton4.AutoSize = true;

this.radioButton4.Location = new System.Drawing.Point(7, 85);

this.radioButton4.Name = "radioButton4";

this.radioButton4.Size = new System.Drawing.Size(101, 17);

this.radioButton4.TabIndex = 3;

this.radioButton4.TabStop = true;

this.radioButton4.Text = "нейронный газ";

this.radioButton4.UseVisualStyleBackColor = true;

//

// radioButton3

//

this.radioButton3.AutoSize = true;

this.radioButton3.Location = new System.Drawing.Point(7, 62);

this.radioButton3.Name = "radioButton3";

this.radioButton3.Size = new System.Drawing.Size(84, 17);

this.radioButton3.TabIndex = 2;

this.radioButton3.TabStop = true;

this.radioButton3.Text = "wtmModified";

this.radioButton3.UseVisualStyleBackColor = true;

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Location = new System.Drawing.Point(7, 39);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(44, 17);

this.radioButton2.TabIndex = 1;

this.radioButton2.TabStop = true;

this.radioButton2.Text = "wtm";

this.radioButton2.UseVisualStyleBackColor = true;

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Checked = true;

this.radioButton1.Location = new System.Drawing.Point(7, 20);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(42, 17);

this.radioButton1.TabIndex = 0;

this.radioButton1.TabStop = true;

this.radioButton1.Text = "wta";

this.radioButton1.UseVisualStyleBackColor = true;

//

// Form1

//

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

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

this.ClientSize = new System.Drawing.Size(644, 527);

this.Controls.Add(this.groupBox1);

this.Name = "Form1";

this.Text = "Карты Кохонена";

this.Load += new System.EventHandler(this.Form1\_Load);

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.groupBox3.ResumeLayout(false);

this.groupBox3.PerformLayout();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.GroupBox groupBox1;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.GroupBox groupBox3;

private System.Windows.Forms.RadioButton radioButton7;

private System.Windows.Forms.RadioButton radioButton8;

private System.Windows.Forms.GroupBox groupBox2;

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.Button button1;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.TextBox textBox5;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.Label label3;

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace kohonen

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

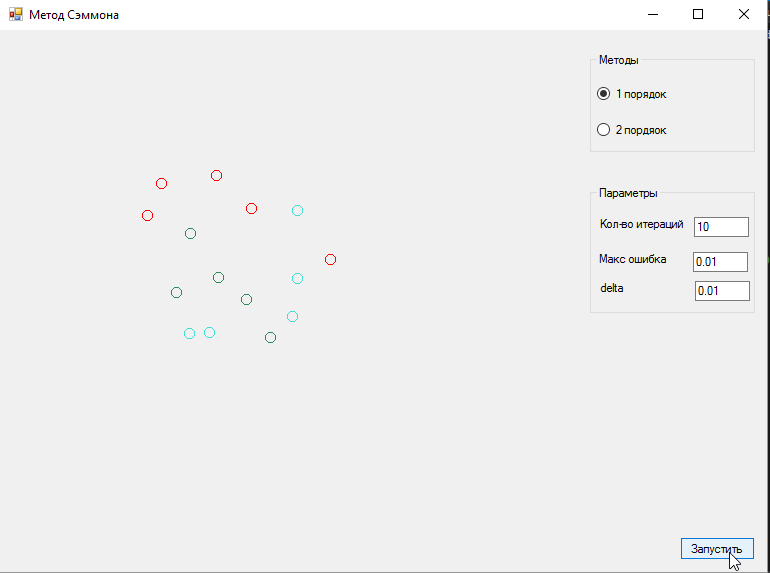
}

}

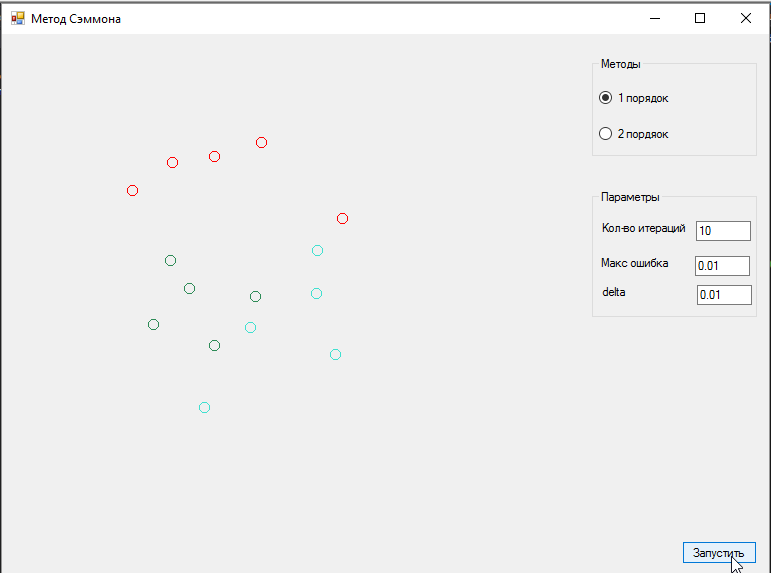
}

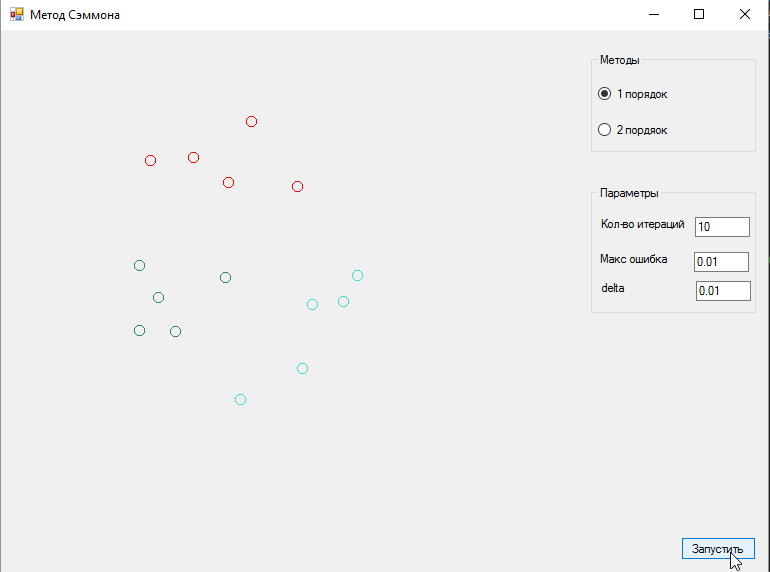
# 5. Визуализация многомерных образов. Алгоритм 1 и 2 порядка. Динамическая картина.

Результаты работы программы:



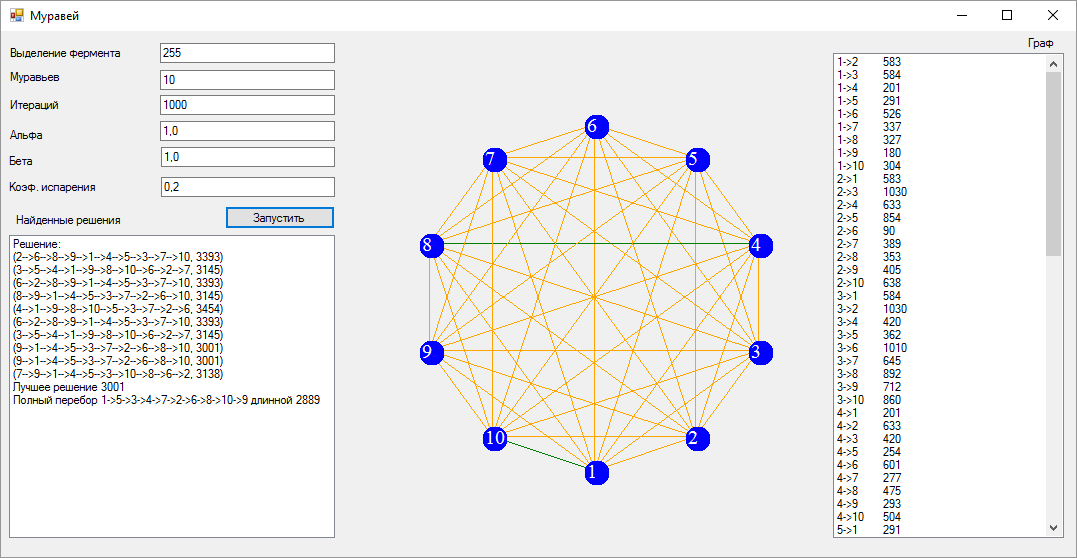






# 6. Муравьинный алгоритм

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;

namespace Ant

{

class AntAgent

{

public int InitialCity { get; set; }

public List<int> Route { get; set; }

public double GetRouteLength(double[,] graph)

{

double bestLength = 0;

for (int i = 0; i < Route.Count - 1; i++)

{

bestLength += graph[Route[i], Route[i + 1]];

}

bestLength += graph[Route.Last(), Route.First()];

return bestLength;

}

public List<double> RouteDistance { get; set; }

public int CurrentCity { get; set; }

public double RouteDistanceSum { get; set; }

//public HashSet<int> VisitedCity { get; set; }

public bool[] VisitedCities { get; set; }

public int PassedCitiesCounter { get; set; }

public bool IsAlive { get; set; }

public bool RouteCompleted { get; set; }

public void init(int n)

{

Route = new List<int>();

RouteDistance = new List<double>();

RouteDistanceSum = 0;

PassedCitiesCounter = 0;

VisitedCities = new bool[n];

}

}

class Probability : IComparable

{

public double P { get; set; }

public int N { get; set; }

#region Члены IComparable

public int CompareTo(object obj)

{

return P.CompareTo(((Probability)obj).P);

}

#endregion

}

class AntsAlgorithm

{

#region private

private double[,] graph;

private static Random r = new Random();

private double[,] pheromons;

private int iterations;

private int Q = 255, Z;

private double alpha, betta, e;

private AntAgent[] ants;

private List<Probability> GoNext(AntAgent ant)

{

List<Probability> candidates = new List<Probability>();

int t = ant.CurrentCity, s;

double propab, sum = 0;

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

{

s = i;

if (graph[t, s] > 0 && !ant.VisitedCities[s]) {

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.VisitedCities[s])

{

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 int ChooseNextCity(AntAgent 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(AntAgent ant)

{

ant.IsAlive = false;

if (ant.PassedCitiesCounter == CityCount && ant.RouteDistanceSum < DistanceMaxThreshold)

{

int start, finish;

start = ant.Route[0];

for (int i = 1; i < ant.Route.Count; i++)

{

finish = ant.Route[i];

ant.RouteDistanceSum = ant.RouteDistance.Sum();

pheromons[start, finish] = Q / ant.RouteDistanceSum;

start = finish;

}

ant.IsAlive = true;

return;

}

}

#endregion

public Tuple<string, double>[] FindRoutes(double alpha, double betta, double e, int Z, int M, int Q)

{

this.iterations = M;

this.Z = Z;

this.alpha = alpha;

this.betta = betta;

this.e = e;

this.Q = Q;

DistanceMaxThreshold = double.MaxValue;

ants = new AntAgent[Z];

for (int iter = 0; iter < iterations; iter++)

{

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

{

ants[z] = new AntAgent();

ants[z].RouteDistanceSum = 0;

ants[z].RouteCompleted = false;

ants[z].init(CityCount);

ants[z].InitialCity = ants[z].CurrentCity = r.Next() % CityCount;

ants[z].Route.Add(ants[z].InitialCity);

ants[z].VisitedCities[ants[z].InitialCity] = true;

ants[z].PassedCitiesCounter++;

}

while (true)

{

int k = 0;

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

{

int Next = ChooseNextCity(ants[z]);

Next = ChooseNextCity(ants[z]);

if (Next == -1)

{

ants[z].RouteCompleted = true;

}

else

{

k++;

ants[z].PassedCitiesCounter++;

ants[z].VisitedCities[Next] = true;

ants[z].Route.Add(Next);

ants[z].RouteDistanceSum += graph[ants[z].CurrentCity, Next];

ants[z].RouteDistance.Add(graph[ants[z].CurrentCity, Next]);

ants[z].CurrentCity = Next;

}

}

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);

}

}

Tuple<string, double>[] pathAndLength = new Tuple<string, double>[Z];

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

{

StringBuilder sb = new StringBuilder();

if (ants[i].IsAlive && ants[i].PassedCitiesCounter == CityCount)

{

for (int j = 0; j < ants[i].Route.Count; j++)

sb.Append((ants[i].Route[j] + 1).ToString() + (j == ants[i].Route.Count - 1 ? "" : "-->"));

pathAndLength[i] = new Tuple<string, double>(sb.ToString(), ants[i].GetRouteLength(graph));

//sb.Append("\tcost: " +ants[i].L.ToString());

}

//if (!str.Contains(sb.ToString()))

// str[i] = sb.ToString();

}

return pathAndLength;

}

public double[,] Graph { get { return graph; } set { graph = value; } }

public double[,] Pheromons { get { return pheromons; } set { pheromons = value; } }

public int CityCount { get; set; }

public double DistanceMaxThreshold { get; set; }

public event EventHandler IterationFinished;

}

}

﻿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);

}

}

}

﻿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;

using MoreLinq;

namespace Ant

{

public partial class Form1 : Form

{

double[,] gr;

double[,] tau, grv;

int [,] lw;

AntsAlgorithm 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();

int.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 + 1).ToString() + "\t" + Math.Round(gr[i, j], 1).ToString());

tau[i, j] = 0.5;

}

}

}

}

}

public void UpdateGraph(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;

int.TryParse(textBox1.Text, out q);

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

int.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 AntsAlgorithm();

s.Graph = gr;

s.Pheromons = tau;

s.CityCount = c;

s.IterationFinished += new EventHandler(UpdateGraph);

var routesAndLength = s.FindRoutes(al, bt, ug, z, m, q);

listBox1.Items.Clear();

listBox1.Items.Add("Решение:");

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

if (!String.IsNullOrEmpty(routesAndLength[i].Item1))

listBox1.Items.Add(routesAndLength[i]);

var best = routesAndLength.MinBy((route) => route.Item2);

listBox1.Items.Add($"Лучшее решение {best.Item2}");

BruteForceAlgorithm bruteForce = new BruteForceAlgorithm(gr);

var bestRoute = bruteForce.FindBestRoute();

listBox1.Items.Add($"Полный перебор {bestRoute.Item1} длинной {bestRoute.Item2}");

}

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.Orange;

lw[i, j] = (int)(3.0 \* val / 50 + 1);

}

if (val >= 50 && val < 140)

{

colors[i, j] = Color.Green;

lw[i, j] = (int)(3.0 \* val / 140 + 2);

}

if (val >= 140 )

{

colors[i, j] = Color.Purple;

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++)

{

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);

}

}

}

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.Blue), coordx, coordy, 25, 25);

e.Graphics.DrawString((i + 1).ToString(), f, new SolidBrush(Color.White), coordx, coordy);

}

}

}

}

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

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

this.SuspendLayout();

//

// button1

//

this.button1.Location = new System.Drawing.Point(224, 175);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(110, 23);

this.button1.TabIndex = 0;

this.button1.Text = "Запустить";

this.button1.UseVisualStyleBackColor = true;

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

//

// panel1

//

this.panel1.BackColor = System.Drawing.SystemColors.Control;

this.panel1.Location = new System.Drawing.Point(340, 5);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(486, 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(6, 15);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(118, 13);

this.label1.TabIndex = 2;

this.label1.Text = "Выделение фермента";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(159, 12);

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(6, 39);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(57, 13);

this.label2.TabIndex = 2;

this.label2.Text = "Муравьев";

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(159, 39);

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(6, 67);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(56, 13);

this.label3.TabIndex = 2;

this.label3.Text = "Итераций";

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(159, 64);

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(6, 97);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(40, 13);

this.label4.TabIndex = 2;

this.label4.Text = "Альфа";

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(159, 90);

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(5, 123);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(31, 13);

this.label5.TabIndex = 2;

this.label5.Text = "Бета";

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(160, 116);

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(5, 149);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(94, 13);

this.label6.TabIndex = 2;

this.label6.Text = "Коэф. испарения";

//

// textBox6

//

this.textBox6.Location = new System.Drawing.Point(160, 146);

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(8, 204);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(326, 303);

this.listBox1.TabIndex = 4;

//

// listBox2

//

this.listBox2.FormattingEnabled = true;

this.listBox2.Location = new System.Drawing.Point(832, 22);

this.listBox2.Name = "listBox2";

this.listBox2.Size = new System.Drawing.Size(231, 485);

this.listBox2.TabIndex = 4;

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(12, 182);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(112, 13);

this.label7.TabIndex = 5;

this.label7.Text = "Найденные решения";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(1024, 5);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(33, 13);

this.label8.TabIndex = 6;

this.label8.Text = "Граф";

//

// Form1

//

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

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

this.ClientSize = new System.Drawing.Size(1075, 526);

this.Controls.Add(this.label8);

this.Controls.Add(this.label7);

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 = "Муравей";

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;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace Ant

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

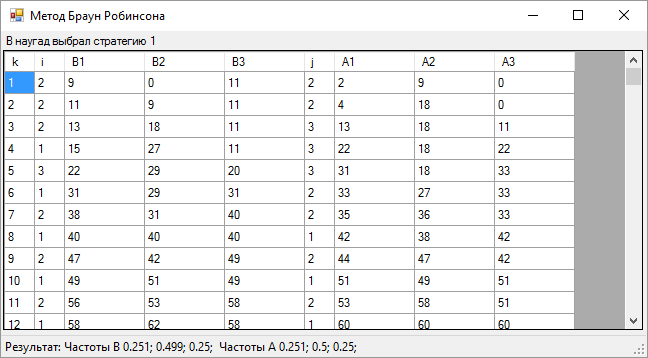
# 8. Итеративный метод Браун-Робинсона

Результат работы программы для решения игры 3x3

7 2 9

2 9 0

9 0 11



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.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();

LoadData();

Load += Form1\_Load;

}

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

{

Calculate();

}

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 LoadData()

{

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();

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 Calculate()

{

if (validateTable())

{

toolStripStatusLabel1.Text = $"Есть седловая точка A{pureA + 1} B{pureB + 1}";

return;

}

IterationsCount = 1000;

dataGridView1.Rows.Clear();

bLastChosen = 0;// randomGenerator.Next() % aStrategiesCount;

double tv;

int frst = bLastChosen;

label1.Text = $"B наугад выбрал стратегию {bLastChosen + 1}";

//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();

dataGridView1.Rows.Add(new string[]

{

$"{iter + 1}"

,$"{aLastChosen + 1}"

,$"{Math.Round(bCurrentChoice[0], 2)}"

,$"{Math.Round(bCurrentChoice[1], 2)}"

,$"{Math.Round(bCurrentChoice[2], 2)}"

,$"{bLastChosen + 1}"

,$"{Math.Round(aCurrentChoice[0], 2)}"

,$"{Math.Round(aCurrentChoice[1], 2)}"

,$"{Math.Round(aCurrentChoice[2], 2)}"

});

}

StringBuilder resultStringBuilder = new StringBuilder();

resultStringBuilder.Append("Результат: ");

resultStringBuilder.Append("Частоты B ");

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

resultStringBuilder.Append(Math.Round((double)bChoiceRate[j]/IterationsCount, 3).ToString() + "; ");

resultStringBuilder.Append(" Частоты A ");

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

resultStringBuilder.Append(Math.Round((double)aChoiceRate[j] / IterationsCount, 3).ToString() + "; ");

toolStripStatusLabel1.Text = resultStringBuilder.ToString();

}

}

}

﻿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.dataGridView1 = new System.Windows.Forms.DataGridView();

this.Columnk = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Columni = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnB1 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnB2 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnB3 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.Columnj = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnA1 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnA2 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.ColumnA3 = new System.Windows.Forms.DataGridViewTextBoxColumn();

this.statusStrip1 = new System.Windows.Forms.StatusStrip();

this.toolStripStatusLabel1 = new System.Windows.Forms.ToolStripStatusLabel();

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

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();

this.statusStrip1.SuspendLayout();

this.SuspendLayout();

//

// dataGridView1

//

this.dataGridView1.AllowUserToAddRows = false;

this.dataGridView1.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridView1.Columns.AddRange(new System.Windows.Forms.DataGridViewColumn[] {

this.Columnk,

this.Columni,

this.ColumnB1,

this.ColumnB2,

this.ColumnB3,

this.Columnj,

this.ColumnA1,

this.ColumnA2,

this.ColumnA3});

this.dataGridView1.Location = new System.Drawing.Point(2, 19);

this.dataGridView1.MultiSelect = false;

this.dataGridView1.Name = "dataGridView1";

this.dataGridView1.RowHeadersVisible = false;

this.dataGridView1.Size = new System.Drawing.Size(640, 280);

this.dataGridView1.TabIndex = 2;

//

// Columnk

//

this.Columnk.FillWeight = 30F;

this.Columnk.HeaderText = "k";

this.Columnk.Name = "Columnk";

this.Columnk.Width = 30;

//

// Columni

//

this.Columni.FillWeight = 30F;

this.Columni.HeaderText = "i";

this.Columni.Name = "Columni";

this.Columni.Width = 30;

//

// ColumnB1

//

this.ColumnB1.FillWeight = 80F;

this.ColumnB1.HeaderText = "B1";

this.ColumnB1.Name = "ColumnB1";

this.ColumnB1.Width = 80;

//

// ColumnB2

//

this.ColumnB2.FillWeight = 80F;

this.ColumnB2.HeaderText = "B2";

this.ColumnB2.Name = "ColumnB2";

this.ColumnB2.Width = 80;

//

// ColumnB3

//

this.ColumnB3.FillWeight = 80F;

this.ColumnB3.HeaderText = "B3";

this.ColumnB3.Name = "ColumnB3";

this.ColumnB3.Width = 80;

//

// Columnj

//

this.Columnj.FillWeight = 30F;

this.Columnj.HeaderText = "j";

this.Columnj.Name = "Columnj";

this.Columnj.Width = 30;

//

// ColumnA1

//

this.ColumnA1.FillWeight = 80F;

this.ColumnA1.HeaderText = "A1";

this.ColumnA1.Name = "ColumnA1";

this.ColumnA1.Width = 80;

//

// ColumnA2

//

this.ColumnA2.FillWeight = 80F;

this.ColumnA2.HeaderText = "A2";

this.ColumnA2.Name = "ColumnA2";

this.ColumnA2.Width = 80;

//

// ColumnA3

//

this.ColumnA3.FillWeight = 80F;

this.ColumnA3.HeaderText = "A3";

this.ColumnA3.Name = "ColumnA3";

this.ColumnA3.Width = 80;

//

// statusStrip1

//

this.statusStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripStatusLabel1});

this.statusStrip1.Location = new System.Drawing.Point(0, 304);

this.statusStrip1.Name = "statusStrip1";

this.statusStrip1.Size = new System.Drawing.Size(646, 22);

this.statusStrip1.TabIndex = 3;

this.statusStrip1.Text = "statusStrip1";

//

// toolStripStatusLabel1

//

this.toolStripStatusLabel1.Name = "toolStripStatusLabel1";

this.toolStripStatusLabel1.Size = new System.Drawing.Size(0, 17);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(2, 3);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(0, 13);

this.label1.TabIndex = 4;

//

// 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.label1);

this.Controls.Add(this.statusStrip1);

this.Controls.Add(this.dataGridView1);

this.Name = "Form1";

this.Text = "Метод Браун Робинсона";

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();

this.statusStrip1.ResumeLayout(false);

this.statusStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.DataGridView dataGridView1;

private System.Windows.Forms.DataGridViewTextBoxColumn Columnk;

private System.Windows.Forms.DataGridViewTextBoxColumn Columni;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnB1;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnB2;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnB3;

private System.Windows.Forms.DataGridViewTextBoxColumn Columnj;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnA1;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnA2;

private System.Windows.Forms.DataGridViewTextBoxColumn ColumnA3;

private System.Windows.Forms.StatusStrip statusStrip1;

private System.Windows.Forms.ToolStripStatusLabel toolStripStatusLabel1;

private System.Windows.Forms.Label label1;

}

}

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace Iterations

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

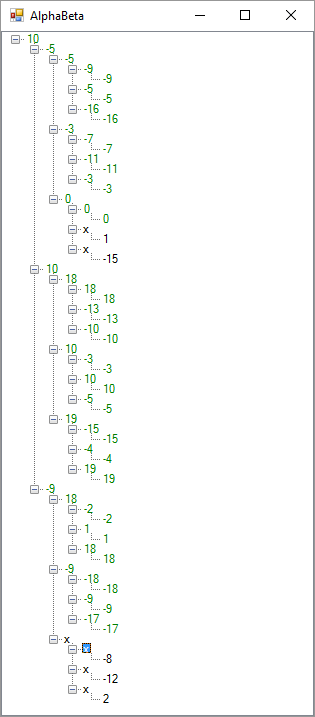
}

}

}

# Альфа-бета отсечение

Результат работы программы:



Исходный код

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)

{

}

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; }

}

}

﻿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,"x");

}

}

}

}

treeView1.ExpandAll();

}

private double AlphaBeta(double alpha, double beta, bool isMaximizing,TreeNode node)

{

double estimation;

node.ForeColor = Color.Green;

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

this.treeView1.ItemHeight = 10;

this.treeView1.Location = new System.Drawing.Point(0, 0);

this.treeView1.Name = "treeView1";

this.treeView1.Size = new System.Drawing.Size(284, 261);

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, 261);

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.Threading.Tasks;

using System.Windows.Forms;

namespace alphabeta

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}