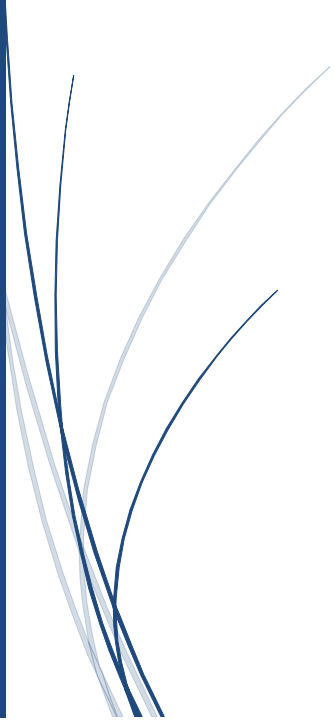
Московский государственный университет им. М. В. Ломоносова
Филиал МГУ в г. Севастополе
Факультет компьютерной математики
Кафедра вычислительной математики

Скаковская А.Н.

Учебно-методическое пособие по дисциплине «Основы проектирования интеллектуальных систем»

для студентов специальности
01.03.02 Прикладная математика и информатика



Севастополь, 2024

УДК 658.512.011.56

ББК 22.18я73

Рассмотрено на заседании учебно-методического совета филиала МГУ в г. Севастополе и рекомендовано к печати Протокол № 2 от 13 июня 2024 г.

С-43 Учебно-методическое пособие по дисциплине «Основы проектирования интеллектуальных систем» / Филиал МГУ в г. Севастополе. — Севастополь, 2024. — 63 с.

Целью пособия является информационное и методическое обеспечение цикла практических работ для разработки интеллектуальной системы с использованием оконных и консольных приложений на языке Visual C# в Microsoft Visual Studio

Пособие адресовано студентам и специалистам, интересующимся проблемами развития теории искусственного интеллекта и ее использования в современных информационных технологиях.

СОДЕРЖАНИЕ

Введение

В стратегии развития информационного общества в Российской Федерации на 2017–2030 годы определены главные направления его развития, среди которых отдельно выделены информационные системы (далее – ИС), которые «стали частью повседневной жизни россиян» и «оказывают существенное влияние на развитие традиционных отраслей экономики» [1].

Цифровая трансформация отечественной экономики, структурные изменения на рынке труда, появление новых специальностей требуют от современных систем высшего образования составления новых образовательных программ по созданию ИС, адаптированных к новым условиям. Сегодня актуальны такие курсы, которые объединяют основные теоретические разделы проектирования информационных систем и специфику их практического использования в трудовой деятельности специалистов.

Предлагается новое учебно-методическое пособие для выполнения практических работ по курсу «Основы проектирования интеллектуальных систем», которое содержит: темы и цели практических работ, теоретические сведения, постановку задач, примеры их реализации на языке C#, результаты программирования и контрольные вопросы по каждой теме.

В процессе изучения курса студенты создают систему, обладающую интеллектуальными способностями, такими как распознавание образов, распознавание речи, принятие решений.

Задачи дисциплины: освоение базовых математических теорий интеллектуальных систем и освоение навыков, необходимых для получения требуемых компетенций в области искусственного интеллекта, математических моделей интеллектуальных систем, дискретной математики,

информатики, робототехники, программирования и моделирования ИС.

Данная разработка может заинтересовать не только высшие учебные заведения, но и центры переподготовки и повышения квалификации.

Автор выражают благодарность студенту Кулику Андрею за участие в подготовке методического пособия.

Практическое занятие 1. Формирование входной обучающей матрицы

Цель: разработать и программно реализовать алгоритм формирования входной обучающей матрицы.

Назначением базового алгоритма обучения является оптимизация геометрических параметров контейнеров. Входной информацией для обучения по базовым алгоритмом является действительный, в общем случае, массив реализаций образа $\left\{y_m^{(j)} \vee m = \overline{1,M};\ j = \overline{1,n}\right\}$; система полей контрольных допусков $\{\delta_{K,i}\}$ и уровень селекции $\{\rho_m\}$, которые по умолчанию равны 0,5 для всех классов распознавания.

Рассмотрим этапы реализации алгоритма:

1.Формирование бинарной обучающей матрицы $\| x_{m,i}^{(j)} \|$, элементы которой равны

$$x_{m,i}^{(j)} = \begin{cases} 1, & if \quad y_{m,i}^{(j)} \in \delta_{K,i}, \\ 0, & if \quad y_{m,i}^{(j)} \notin \delta_{K,i}. \end{cases}$$

2. Формирование массива эталонных двоичных векторов $\{x_{m,i} \vee m = \overline{1,M},\ i = \overline{1,N}\}$, элементы которого определяются по правилу :

$$x_{m,i} = \begin{cases} 1, & if \quad \frac{1}{n} \sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m, \\ 0, & if \quad else, \end{cases}$$

где ρ_m – уровень селекции координат вектора $x_m \in X_m^o$.

3. Разбиение множества эталонных векторов на пары ближайших “соседей”: $\Re_m^{|2|} = \langle x_m, x_l \rangle$, где x_l – эталонный вектор соседнего класса X_l^o, по такому алгоритму:

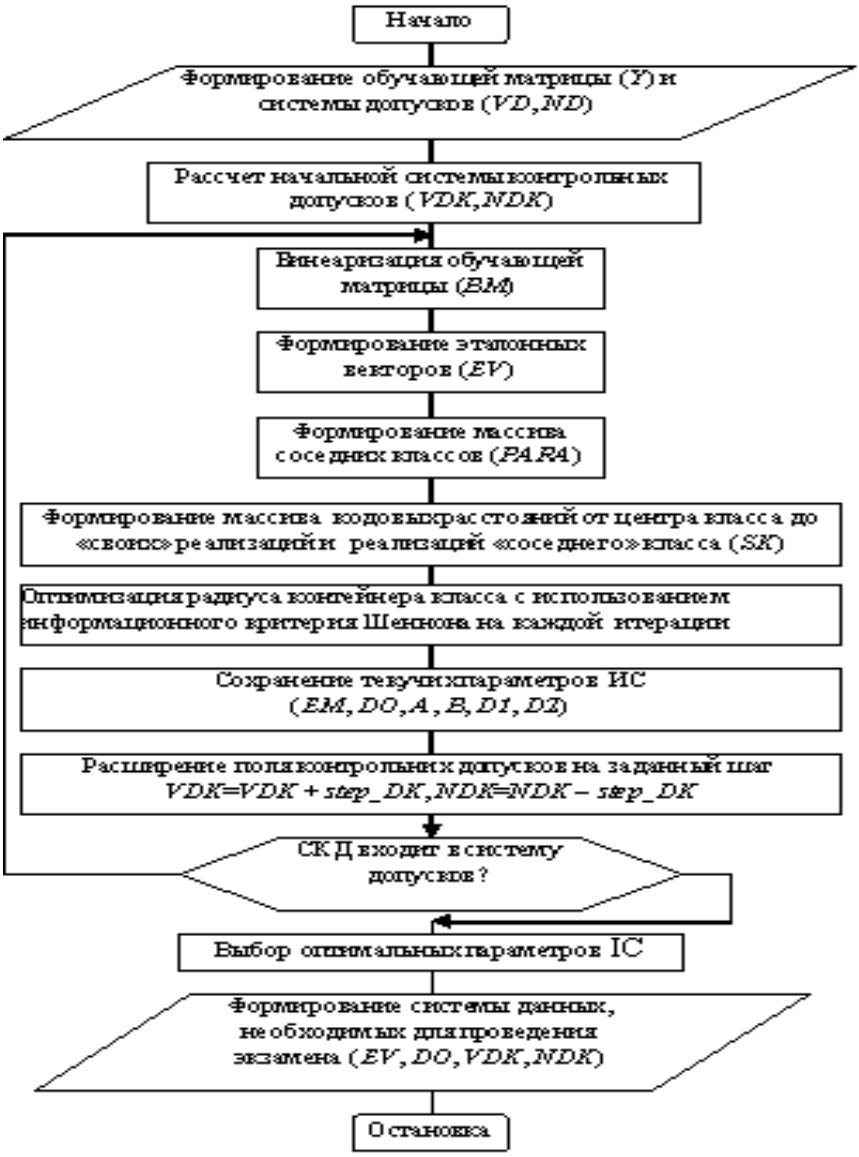


Рисунок 1. Структурная схема базового алгоритма обучения

а) структурируется множество эталонных векторов, начиная с вектора x_1 базового класса X_1^o, характеризующий наибольшую функциональную эффективность системы поддержки принятия решений (СППР);

б) строится матрица кодовых расстояний между эталонными векторами размерности $M \times M$;

в) для каждой строки матрицы кодовых расстояний находится минимальный элемент, который принадлежит столбцу вектора - ближайшего к вектору, который определяет строку. При наличии нескольких одинаковых минимальных элементов выбирается из них любой, поскольку они являются равноправными;

г) формируется структурированное множество элементов парного разбиения $\{\Re_{\mathrm{m}}^{|2|} \mid \mathrm{m} = \overline{1,\mathrm{M}}\}$, задающее план обучения.

4. Оптимизация кодового расстояния d_m происходит рекуррентной процедурой. При этом принимается $E_m(0) = 0$.

5. Процедура заканчивается при нахождении максимума КФЭ в рабочей области его определения $E_m^* = \max\limits_{\{d\}} E_m$, где $\{d\}=\{0,1,...,d<d(x_m \oplus x_l)\}$ - множество радиусов гиперсфер, центр которых определяется вершиной $x_m \square X_m^o$.

Таким образом, базовый алгоритм обучения является итерационной процедурой поиска глобального максимума информационного КФЭ в рабочей области определения его функции

$$d_m^* = \arg \max\limits_{\{d\}} E_m^* .$$

Задание 1 Провести описание основных констант и переменных согласно табл.1.

Таблица 1. Основные константы и переменные

m – количество классов (*k* = 2)
N – количество признаков *распознавания* (*N* = 100)
n – количество реализаций (*n* = 100)
k – текущий класс
I, J – текущие признак и реализация в соответствии
Y[1..*m*, 1..*N*, 1..*n*] – входная обучающая матрица
X[1..*m*, 1..*N*, 1..*n*] – бинарная учебная матрица
VD[1..*N*], ***ND***[1..*N*] – система допусков
VDK[1..*N*], ***NDK***[1..*N*] – система контрольных допусков
EV[1..*m,*1..*N*] – центры классов
PARA[1..*m*] – массив соседних классов
SK[1..2, 1..*n*] – массив кодовых расстояний до реализаций
E[1..*m*], ***A***[1..*m*], ***B***[1..*m*], ***D1***[1..*m*], ***D2***[1..*m*] – оптимальное значение информационного критерия и точностные характеристики соответственно
DO[1..*m*] – оптимальное значение радиуса контейнера класса
step_DK – шаг изменения системы контрольных допусков

Задание 2. Сформировать согласно приведенному примеру входную обучающуюся матрицу *Y*[1..*m*, 1..*N*, 1..*n*] для распознавания стационарных по яркости изображений (текстур).
Задание 3. Разработать предварительный интерфейс системы. Основное внимание при этом уделить созданию диалогового режима с пользователями.

Порядок выполнения работы

1. Записать тему и цель работы.
2. Ознакомиться с базовым алгоритмом обучения интеллектуальной системы, структурная схема которого показана на рис.1.
3. Описать основные константы и переменные согласно табл. 1.
4. В соответствии с вариантом задания скопировать два изображения из библиотеки изображений типа текстуры [1].
5. Сформировать входную обучающуюся матрицу *X[1, ..., m, 1, ... N, 1, ... n]* для распознавания изображений (текстур).
6. Разработать интерактивный интерфейс программной системы.

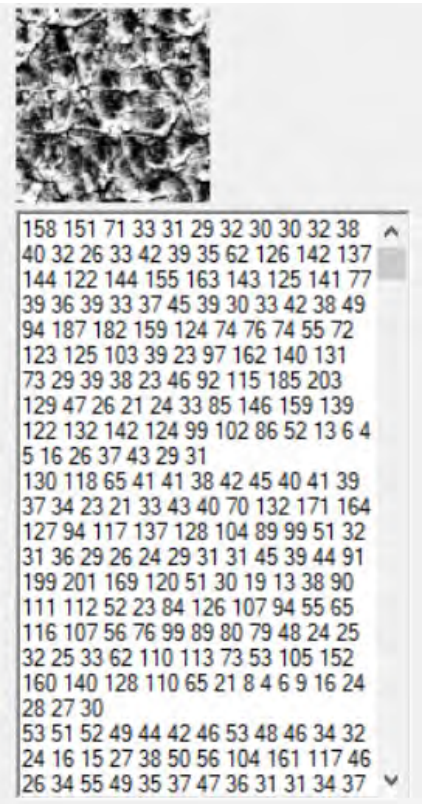
Пример формирования обучающей матрицы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace _3D
{
    static class Program
    {

        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
```

```
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());

        }

    }
}
using System;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using IronXL;

namespace _3D
{
    public partial class Form1 : Form
    {

     PictureBox[] pb_mas = new PictureBox[2];

                Загрузка файлов картинок по кнопке
     private void button1_Click_1(object sender, EventArgs e)
             {
                 OpenFileDialog openFileDialog1 = new
     OpenFileDialog();
                 DialogResult result =
     openFileDialog1.ShowDialog(); // Show the dialog.
                 if (result == DialogResult.OK) // Test result.
                 {
                     string file = openFileDialog1.FileName;
                     try
                     {
                         PictureBox pb1 = new PictureBox();
                         pb1.Image = Image.FromFile(file);
                         pb_mas[key] = pb1;
                         key++;
                         notCalculate = true;
                     }
                     catch (IOException)
                     {
```

```
                }

        }
            Перевод картинки в обучающую матрицу картинок по кнопке
        private void draw() {
        pictureBox1.Image = pb_mas[0].Image;
                            Bitmap image1 = new
        Bitmap(pb_mas[0].Image);
                            for (int x = 0; x <
        pb_mas[0].Image.Width; x++)
                            {
                                for (int y = 0; y <
        pb_mas[0].Image.Height; y++)
                                {
                                    Color pixelColor =
        image1.GetPixel(x, y);
                                    txt += (pixelColor.R +
        pixelColor.B + pixelColor.G) / 3 + " ";
                                }
                                txt += Environment.NewLine;
                            }

                            double[] mean = new double[100];

                            for (int x = 0; x <
        pb_mas[0].Image.Width; x++)
                            {
                                int s = 0;
                                for (int y = 0; y <
        pb_mas[0].Image.Height; y++)
                                {
                                    Color pixelColor =
        image1.GetPixel(y, x);
                                    s += (pixelColor.R +
        pixelColor.B + pixelColor.G) / 3;
                                }
                                mean[x] = s / 100;
                            }
        }
}
```

Графическое отображение обучающейся матрицы для двух классов распознавания



Контрольные вопросы

1. Что такое стационарное изображение? В чем заключается его отличие от нестационарного?
2. Какие различия имеют алгоритмы создания обучающейся матрицы со стационарными и нестационарными изображениями?
4. Базовый алгоритм обучения интеллектуальной системы. Структурная схема базового алгоритма обучения.
5. Как сформировать входную обучающуюся для распознавания стационарных по яркости изображений?

6. Построение бинарной обучающей матрицы.
7. Формирование массива эталонных двоичных векторов.

Практическое занятие 2. Формирование бинарной обучающей матрицы

Цель: разработать и программно реализовать алгоритм формирования бинарной обучающей матрицы.

Теоретические сведения

Рассмотрим обучающую матрицу типа «объект-свойство», которая характеризует *m*-ое функциональное состояние СПР класс распознавания X_m^o:

$$\| y_{m,i}^{(j)} \| = \begin{vmatrix} y_{m,1}^{(1)} & y_{m,2}^{(1)} & \dots & y_{m,1}^{(1)} & \dots & y_{m,N}^{(1)} \\ y_{m,1}^{(2)} & y_{m,2}^{(2)} & \dots & y_{m,1}^{(2)} & \dots & y_{m,N}^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ y_{m,1}^{(j)} & y_{m,2}^{(j)} & \dots & y_{m,1}^{(j)} & \dots & y_{m,N}^{(j)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ y_{m,1}^{(n)} & y_{m,2}^{(n)} & \dots & y_{m,1}^{(n)} & \dots & y_{m,N}^{(n)} \end{vmatrix} \quad (1)$$

Элементами этой матрицы являются экспериментальные данные, отражающие основные свойства объекта и, в общем случае являются значениями случайной величины - признаки распознавания. В матрице (1) строка является реализацией образа $\{y_{m,i}^{(j)} \vee i = \overline{1, N}\}$, где *N* - количество признаков распознавания, а столбец матрицы - случайная обучающая

выборка $\left\{ y_{m,i}^{(j)} \vee j = \overline{1,n} \right\}$, где n - объем выборки, которая формируется в процессе испытаний.

Для изображения обучающаяся матрица формируется путем сканирования рецепторного поля и получения в каждом пикселе значения яркости, которое для черно-белого графического редактора изменяется от 0 до 255 градаций яркости. Таким образом, матрица яркости является целочисленной и рассматривается как входная матрица для системы распознавания.

Задание1. Сформировать бинарную обучающуюся матрицу *X*[1..*m*, 1..*N*, 1..*n*] путем преобразования обучающейся матрицы *Y*[1..*m*, 1..*N*, 1..*n*] в бинарное пространство Хемминга.

Задание2. Разработать модуль для графического отображения бинарной обучающейся матрицы.

Порядок выполнения работы

1. Записать тему и цель работы.
2. Задать значение поля контрольных допусков($\mathrm{delta} = \pm 50$).
3. Перевести обучающуюся матрицу ***Y***[1..*m*, 1..*N*, 1..*n*] в бинарное пространство Хемминга по правилу:

$$X[k,i,j] = \begin{cases} 1, & \textit{если} \quad NDK[i] \le Y[k,i,j] \le VDK[i], \\ 0, & \textit{якщо} \quad NDK[i] \ge Y[k,i,j] \quad \textit{или} \quad Y[k,i,j] \ge VDK[i], \end{cases}$$

где $\mathrm{NDK}[i], \mathrm{VDK}[i]$ – нижний и верхний контрольные допуски $i-го$ признака соответственно.

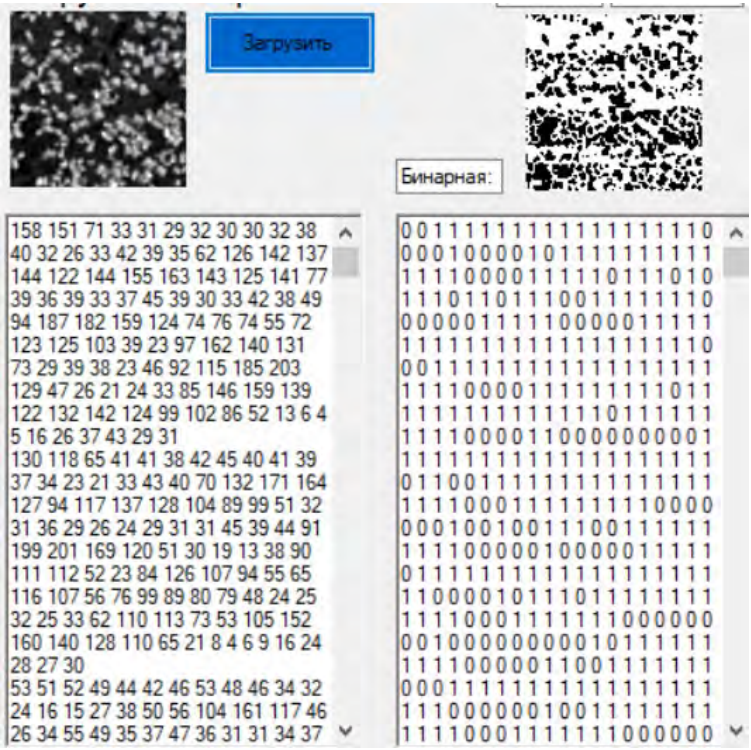
4. Разработать интерактивный интерфейс программной системы.

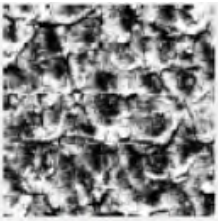
Пример формирования бинарной матрицы

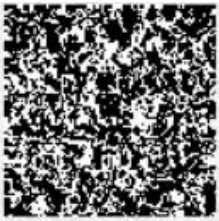
```
/* Указываются изменения в классе Form1 по сравнению с
предыдущей работой */
public partial class Form1 : Form
    {
     /*/
     int[,] pict1 = new int[100, 100];
     int [,] pict2 = new int[100, 100];
     int delta = 50;
     /*/
     private void draw() {
                /*/
          Формирование бинарной матрицы
           for (int x = 0; x < pb_mas[0].Image.Width; x++)
                   {
                       int s = 0;
                       for (int y = 0; y <
pb_mas[0].Image.Height; y++)
                       {
                           Color pixelColor =
image1.GetPixel(y, x);
                           s += (pixelColor.R + pixelColor.B +
pixelColor.G) / 3;
                       }
                       mean[x] = s / 100;
                   }
                   string bin_txt = "";
                   for (int x = 0; x < pb_mas[0].Image.Width;
x++)
                   {
                       for (int y = 0; y <
pb_mas[0].Image.Height; y++)
                       {
                           Color pixelColor =
image1.GetPixel(x, y);
                           int p = (pixelColor.R + pixelColor.B
+ pixelColor.G) / 3;
                           if (p < mean[y] + delta && p >
mean[y] - delta)
                           {
                               pict1[x, y] = 1;
                           }
                           else
```

```
                        {
                            pict1[x, y] = 0;
                        }
                        bin_txt += pict1[x, y] + " ";
                    }
                    bin_txt += Environment.NewLine;
                }
                bin_to_img(pict1, bin1_im);
/*/
}
/*/
Вывод бинарной матрицы в картинку на экран
private void bin_to_img(int[,] mas, PictureBox im)
        {
            Bitmap image1 = new Bitmap(pb_mas[0].Image);
            for (int x = 0; x < pb_mas[0].Image.Width; x++)
            {
                for (int y = 0; y < pb_mas[0].Image.Height; y++)
                {
                    image1.SetPixel(x, y, Color.FromArgb(255 *
mas[x, y], 255 * mas[x, y], 255 * mas[x, y]));
                }

            }
            im.Image = image1;
        }
/*/
```

Графическое отображение бинарной матрицы для двух классов распознавания







Бинарная:

158 151 71 33 31 29 32 30 30 32 38
40 32 26 33 42 39 35 62 126 142 137
144 122 144 155 163 143 125 141 77
39 36 39 33 37 45 39 30 33 42 38 49
94 187 182 159 124 74 76 74 55 72
123 125 103 39 23 97 162 140 131
73 29 39 38 23 46 92 115 185 203
129 47 26 21 24 33 85 146 159 139
122 132 142 124 99 102 86 52 13 6 4
5 16 26 37 43 29 31
130 118 65 41 41 38 42 45 40 41 39
37 34 23 21 33 43 40 70 132 171 164
127 94 117 137 128 104 89 99 51 32
31 36 29 26 24 29 31 31 45 39 44 91
199 201 169 120 51 30 19 13 38 90
111 112 52 23 84 126 107 94 55 65
116 107 56 76 99 89 80 79 48 24 25
32 25 33 62 110 113 73 53 105 152
160 140 128 110 65 21 8 4 6 9 16 24
28 27 30
53 51 52 49 44 42 46 53 48 46 34 32
24 16 15 27 38 50 56 104 161 117 46
26 34 55 49 35 37 47 36 31 31 34 37

0 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0
0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0
0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
1 0 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0
0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1
0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1
0 1 0 1 1 1 0 0 0 1 0 1 1 0 1 1 1 0 0 1
1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0
0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0
1 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0

Контрольные вопросы

1. Как описать систему контрольных допусков на признаки распознавания?
2. С какой целью проводится перевод обучающейся матрицы в бинарное пространство Хемминга?
3. Какой компонент C# использован для графического отображения бинарной матрицы?
4. Что называется контрольным полем допусков на признаки распознавания?
5. Что называется нормированным полем допусков на признаки распознавания?
6. Как формируется обучающаяся матрица?

Практическое занятие 3. Определение эталонных геометрических векторов классов распознавания

Цель: научиться формировать геометрические центры классов распознавания.

Задание 1. Сформировать массив *EV [1..m, 1..N]* геометрических центров классов распознавания, где *m* - количество классов, *N* – количество признаков распознавания.

Теоретические сведения

Эталонный вектор x_m - это математическое ожидание реализаций класса X_m^0. Он представлен в виде детерминированного структурированного бинарного вектора $x_m =< x_m,1,...,x_m,i,..,x_m,N >,..m = \overline{1,M}$, где M – количество классов распознавания, N – количество признаков, x_m,i – ая

координата вектора, которая принимает единичное значение, если значение *i*-го признака распознавания находится в нормированном поле допусков $\delta_{H,i}$, и нулевое значение, если не находится. Элементы $\mathrm{EV}[I,K]$ — двоичного эталонного вектора K – го класса X_1^K вычисляются по правилу:

$$\mathrm{EV[I,1]} = \begin{cases} 1 \ \text{if} \ \dfrac{1}{\mathrm{IMAX}} \sum\limits_{\mathrm{I=1}}^{\mathrm{IMAX}} \mathrm{X[J,I,1]} > 0{,}5; \\ 0 \ \ \text{if} \ \ \text{else}. \end{cases}$$

При обосновании гипотезы компактности (четкой, или нечеткой) реализаций образа геометрическим центром класса X_m^0 является бинарный эталонный вектор x_m.

Порядок выполнения работы

1. Записать тему и цель работы.
2. Сформировать массив *EV [1..m, 1..N]* геометрических центров классов распознавания. При этом значение каждого элемента массива *EV [k, i]* рассчитывается по правилу:

$$EV[k,i] = \begin{cases} 1, \quad \textit{если} \quad \dfrac{1}{n}\sum\limits_{j=1}^{n} X[k,i,j] > 0{,}5, \\ 0, \quad \textit{если} \quad \dfrac{1}{n}\sum\limits_{j=1}^{n} X[k,i,j] \le 0{,}5. \end{cases}$$

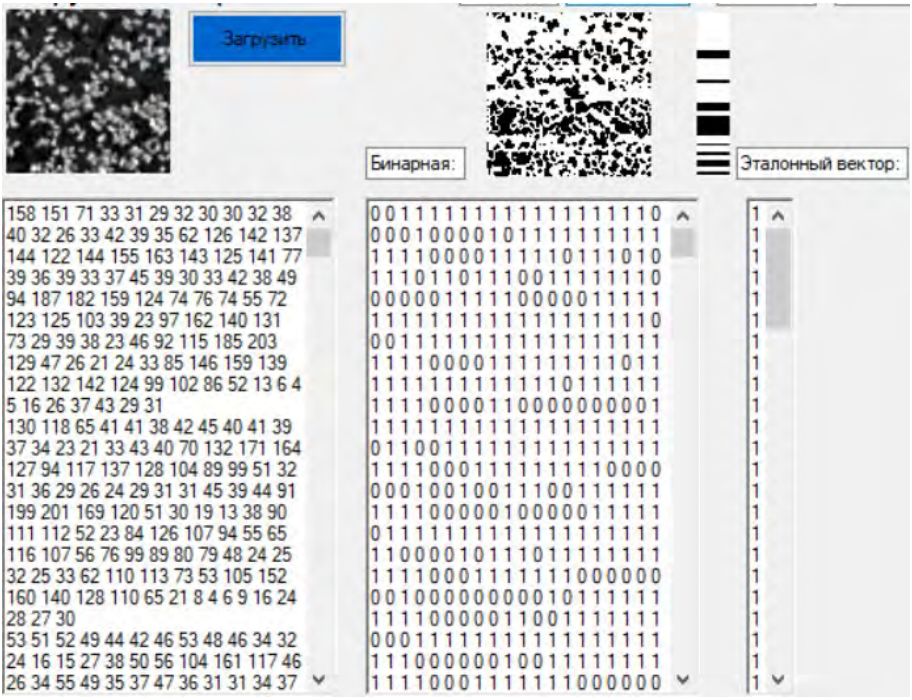
3. Разработать интерактивный интерфейс программной системы.

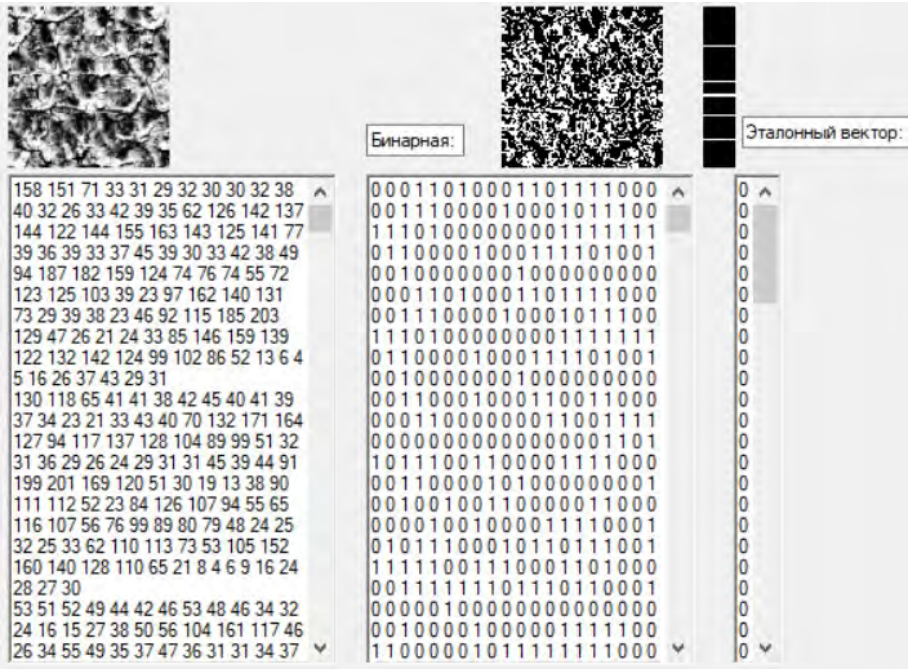
Пример формирования эталонных векторов классов распознавания

```
/* Указываются изменения в классе Form1 по сравнению с
предыдущей работе */
public partial class Form1 : Form
    {
     /*/
     int[] vector2 = new int[100];
     int[] vector1 = new int[100];
     /*/
              Рассчет эталонного вектора для матрицы
private void draw() {
     /*/
     string el_txt = "";
     for (int x = 0; x < pb_mas[0].Image.Width; x++)
          {
           float sum2 = 0;
           for (int y = 0; y < pb_mas[0].Image.Height; y++)
                {
                   sum2 += pict1[y, x];
                }
                if (sum2 / pb_mas[0].Image.Height > ro)
                      {
                          vector1[x] = 1;
                      }
                else
                      {
                          vector1[x] = 0;
                      }

          el_txt += vector1[x].ToString() +
     Environment.NewLine;
          }
vec_to_img(vector1, vec1_img);
     /*/
}
Вывод эталонного вектора на экран
private void vec_to_img(int[] vec, PictureBox im)
        {
            Bitmap image1 = new Bitmap(20, 100);
            for (int x = 0; x < 20; x++)
            {
                for (int y = 0; y < pb_mas[0].Image.Height; y++)
```

```
                {
                    image1.SetPixel(x, y, Color.FromArgb(255 *
vec[y], 255 * vec[y], 255 * vec[y]));
                }

            }
            im.Image = image1;
        }
    /*/
```

Графическое отображение эталонных векторов классов распознавания





Контрольные вопросы

1. Что называется эталонным вектором класса распознавания?
2. Как формируются элементы двоичного эталонного вектора?
3. Что является геометрическим центром класса распознавания?
4. Какой компонент C# был использован для корректного отображения геометрических центров классов распознавания?

Практическое занятие 4. Формирование массива кодовых расстояний между центрами классов распознавания

Цель: разработать и программно реализовать алгоритм формирования массива кодовых расстояний между центрами классов распознавания.

Теоретические сведения

Матрица кодовых расстояний используется для разбиения множества эталонных векторов на пары ближайших соседей: $\Re_{\mathrm{m}}^{|2|} =< \mathrm{xm}, \mathrm{xl} >$, где xl – эталонный вектор соседнего класса X_l^o. Данное разбиение осуществляется по следующему алгоритму:

а) структурируется множество эталонных векторов, начиная с вектора *x1* базового класса X_1^o, характеризующий наибольшую функциональную эффективность СПР;

б) строится матрица кодовых расстояний между эталонными векторами размерности MxN;

в) для каждой строки матрицы кодовых расстояний находится минимальный элемент, который принадлежит столбцу вектора, ближайшего к вектору, который определяет строку. При наличии нескольких одинаковых минимальных элементов выбирается из них любой, поскольку они являются равноправными;

г) формируется структурированное множество элементов парного разбиения $\{\Re_{\mathrm{m}}^{|2|} \mid \mathrm{m} = \overline{1,\mathrm{M}}\}$, задающее план обучения.

Задание1. Сформировать массив *SK[1..2,1..N]* (*N* – количество признаков распознавания) кодовых расстояний от геометрических центров контейнеров классов до их реализаций.

Задание2. Сформировать массив *SK_PARA[1..2,1..N],* аналогичный предыдущему массиву *SK,* но за текущий класс взять ближайший «соседний» класс.
Задание3. Разработать модуль для отображения распределения реализаций между текущим классом и его ближайшим «соседом».

Порядок выполнения работы

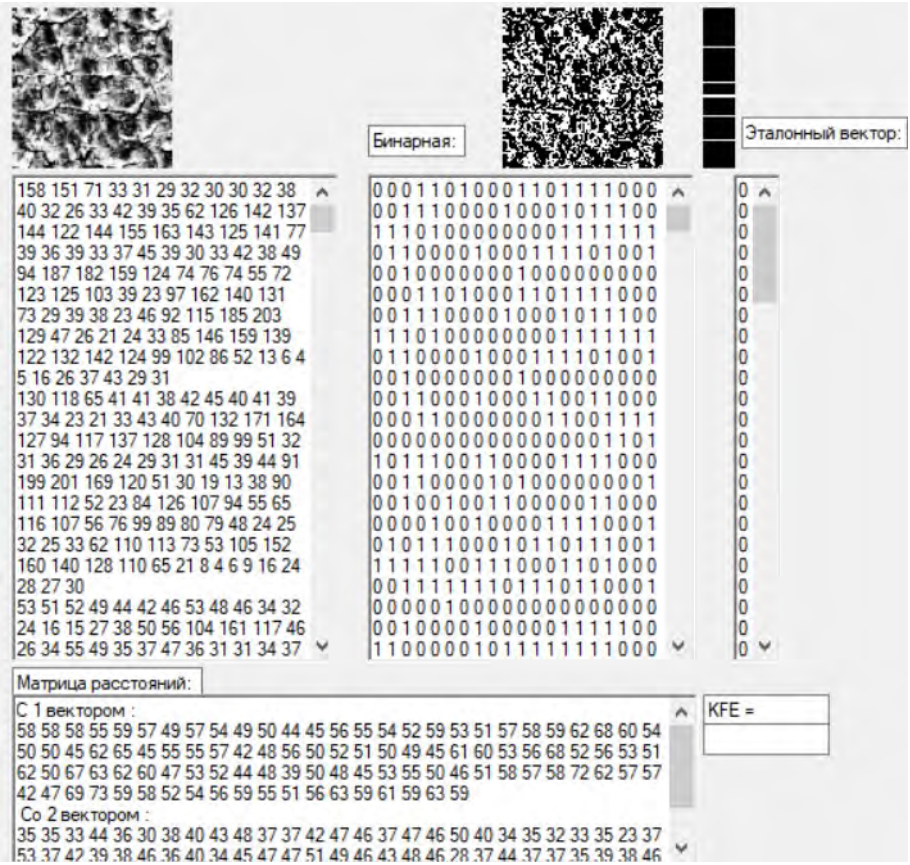
1. Сформировать массив *SK [1..2,1..N]* (*N* – количество признаков распознавания) кодовых расстояний от геометрических центров контейнеров классов до их реализаций. Причем массив *SK[1,1..N]* должен содержать кодовые расстояния между геометрическим центром текущего класса и реализациями этого класса; массив *SK[2,1..N]* должен содержать кодовые расстояния между геометрическим центром текущего класса и реализациями ближайшего соседнего класса.
2. Сформировать массив *SK_PARA[1..2,1..N],* аналогичный предыдущему массиву SK, но за текущий класс взять ближайший соседний класс.
3. Разработать модуль для отображения распределения реализаций между текущим классом и его ближайшим соседом. Для определения положения каждой реализации использовать значение массивов *SK* и *SK_PARA*. Центры классов разместить на горизонтальной прямой.

Пример функции для отображения распределения реализаций между текущим классом и его ближайшим соседом

```
/* Указываются изменения в классе Form1 по сравнению с
предыдущей работой */
public partial class Form1 : Form
    {
     /*/
```

```
      int[] distanse11 = new int[100];
      int[] distanse12 = new int[100];
      int[] distanse21 = new int[100];
      int[] distanse22 = new int[100];
      double ro = 0.5;
      /*/

Расчет расстояний до эталонного вектора
private void draw() {
      string rast_txt = "С 1 вектором : \n";
      for (int x = 0; x < pb_mas[0].Image.Width; x++)
            {
            int different = 0;
            for (int y = 0; y < pb_mas[0].Image.Height; y++)
                  {
                  if (pict1[x, y] != vector1[y])
                  {
                        different += 1;
                  }
            }
            distanse11[x] = different;
            rast_txt += different + " ";
            }
      rast_txt += "\n Со 2 вектором : \n";
      for (int x = 0; x < pb_mas[0].Image.Width; x++)
      {
            int different = 0;
            for (int y = 0; y < pb_mas[0].Image.Height; y++)
            {
                  if (pict1[x, y] != vector2[y])
                  {
                  different += 1;
                  }
            }
            distanse12[x] = different;
            rast_txt += different + " ";
      }
}
```

Графическое отображение распределения реализаций между текущим классом и его ближайшим соседом



Контрольные вопросы

1. Алгоритм формирования массива кодовых расстояний между центрами классов распознавания.
2. Что такое кодовое расстояние? Как строится матрица кодовых расстояний?
3. По какому алгоритму определяется ближайший «соседний» класс распознавания?

4. Какой компонент C# был использован для отображения распределения реализаций?

Практическое занятие 5. Вычисление информационного критерия функциональной эффективности обучения интеллектуальной системы

Цель: разработать и программно реализовать алгоритм вычисления информационного критерия функциональной эффективности обучения системы распознавания.

Теоретические сведения

Информационный подход базируется на использовании для оценки функциональной эффективности информационного критерия, который, например, по Шеннону имеет такой нормированный вид:

$$E = \frac{H_0 - H(\gamma)}{H_0}, \qquad (2)$$

где $H_0 -$ априорное (безусловное) энтропия:

$$H_0 = -\sum_{l=1}^{M} p(\gamma_l)\log_2 p(\gamma_l), \qquad (3)$$

$H(\gamma)$ – апостериорная условная энтропия, характеризующая остаточную неопределенность после принятия решений:

$$H(\gamma) = -\sum_{l=1}^{M} p(\gamma_l)\sum_{l=1}^{M} p(\mu_m/\gamma_l)\log_2 p(\mu_m/\gamma_l), \qquad (4)$$

где $p(\gamma_l)$ – априорная вероятность принятия гипотезы γ_l; $p(\mu_m/\gamma_l)$ – апостериорная вероятность появления события μ_m при условии принятия гипотезы γ_l; M – число альтернативных гипотез.

На практике при оценке функциональной эффективности обучающейся системы управления, могут иметь место такие предположения:

- решение двухальтернативное ($M = 2$);
- поскольку, обучающаяся система управления слабо формализованным процессом функционирует в условиях неопределенности, то по принципу Бернулли-Лапласа оправдано принятие равновероятных гипотез: $p(\gamma_1) = p(\gamma_2) = 0{,}5$

Тогда критерий (2) с учетом выражений (3) и (4) принимает частный вид:

$$E = 1 + \frac{1}{2}\sum_{l=1}^{2}\sum_{m=1}^{2} p(\mu_m / \gamma_l)\log_2 p(\mu_m / \gamma_l). \qquad (5)$$

При двухальтернативном решении (M = 2) основной считаем гипотезу γ_1 о нахождении значения признака распознавания в поле допусков δ и как альтернативную ей - гипотезу γ_2. При этом имеют место четыре возможных результата оценки измерения признаков:

ошибка первого рода - $\alpha = \rho(x \notin \delta / z \in \delta)$;

ошибка второго рода – $\beta = \rho(x \in \delta / z \notin \delta)$;

первая достоверность – $D_1 = \rho(x \in \delta / z \in \delta)$;

вторая достоверность – $D_2 = \rho(x \notin \delta / z \notin \delta)$,

где x, z – измеренное и действительное значение признака распознавания соответственно.

Разобьем множество значений признаков на области μ_1 и μ_2. Область μ_1 включает значения, находящиеся в допуске δ, а μ_2 – не в допуске.

Тогда можно записать $\alpha = p(\gamma_2 / \mu_1)$, $\beta = p(\gamma_1 / \mu_2)$, $D_1 = p(\gamma_1/\mu_1)$, $D_2 = p(\gamma_2/\mu_2)$.

Выразим апостериорные вероятности $p(\mu_m / \gamma_l)$ через априорные по формуле Байеса :

$$p(\mu_m/\gamma_l) = \frac{p(\mu_m)\, p(\gamma_l/\mu_m)}{p(\mu_1)\, p(\gamma_l/\mu_1) + p(\mu_2)\, p(\gamma_l/\mu_2)} \qquad (6)$$

и, взяв $p(\mu_1)=p(\mu_2)=0{,}5$, получаем:

$$p(\mu_1/\gamma_1) = \frac{D_1}{D_1+\beta}, \quad p(\mu_2/\gamma_1) = \frac{\beta}{D_1+\beta},$$
$$p(\mu_1/\gamma_2) = \frac{\alpha}{\alpha+D_2}, \quad p(\mu_2/\gamma_2) = \frac{p_2 D_2}{p_1\alpha + p_2 D_2}. \qquad (7)$$

После подстановки (7) в (6) получим формулу для вычисления КФЭ по Шеннону :

$$E = 1 + \frac{1}{2}\left(\frac{\alpha}{\alpha+D_2}\log_2\frac{\alpha}{\alpha+D_2} + \frac{D_1}{D_1+\beta}\log_2\frac{D_1}{D_1+\beta} + \right.$$
$$\left. + \frac{\beta}{D_1+\beta}\log_2\frac{\beta}{D_1+\beta} + \frac{D_2}{\alpha+D_2}\log_2\frac{D_2}{\alpha+D_2} \right). \qquad (8)$$

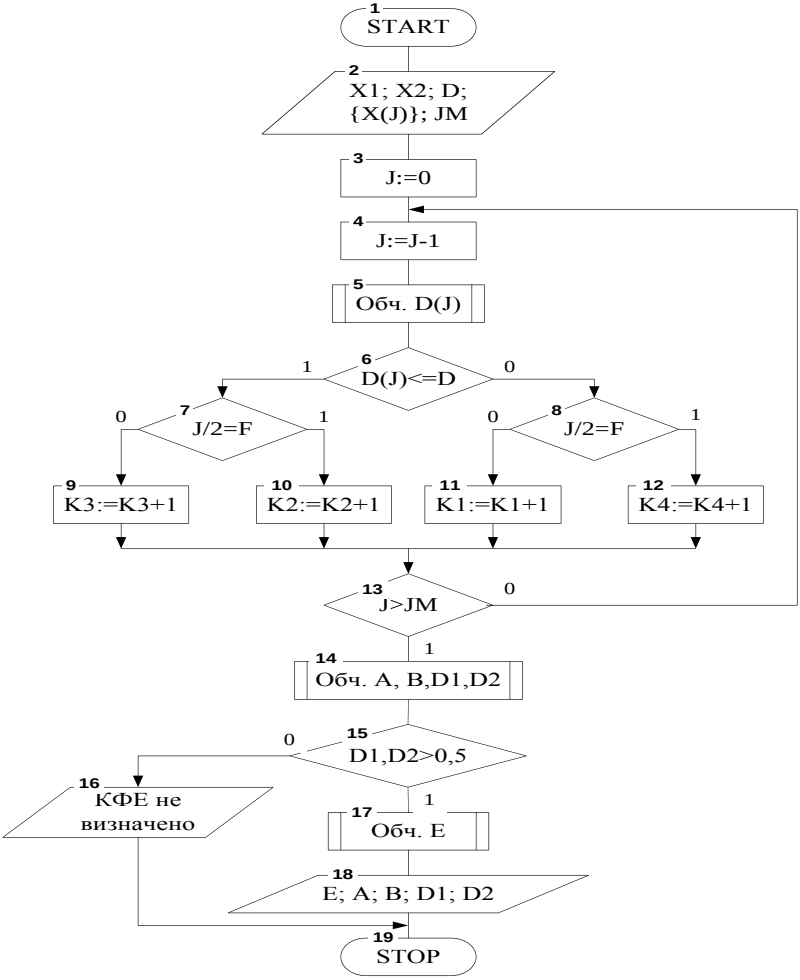


Рисунок 2. Структурная схема вычисления информационного критерия функциональной эффективности обучения системы распознавания

На рис. 2 приведены следующие входные данные: X_1, X_2 – эталонные двоичные векторы классов X_1^o и X_2^o соответственно;

$\{X(N)\}$ – обучающаяся матрица, состоящая из реализаций этих классов; $N = 1, \overline{\mathrm{NM}}$, где *NM* – объем репрезентативной обучающейся выборки, *D* - радиус контейнера класса X_1^o. Выходные данные: *E* - значение КФЭ; α, β, D_1, D_2 – значения точностных характеристик процесса обучения: ошибки первого и второго рода, первая и вторая достоверности соответственно.

Блок 5 вычисляет при каждом испытании кодовое расстояние D(J) путем сложения по модулю два вектора X_1 с текущим вектором-реализацией X(J) и подсчета количества единиц в полученной сумме. При каждом нечетном испытании определяется расстояние D(J) между вектором X_1 и реализацией своего класса, а на каждом четном - между вектором X_1 и реализацией другого класса. Вычисление коэффициентов K_1, K_2, K_3 и K_4 осуществляется по следующему алгоритму (блоки 6 – 12):

а) сравнения (блок 6): если $D(J) \le D$ (реализация принадлежит области класса X_1^o), то при нечетном испытании вычисляется $K_1 := K_1 + 1$ (″своя″ реализация), а при парном – $K_3 := K_3 + 1$ (″чужая″ реализация). Определение четности или нечетности реализаций осуществляют блоки 7 и 8, которые проверяют выполнение условия *J/2 = F*, где *F* - целое число. Если условие выполняется, то испытание четное, иначе - нечетное. Если $D(J) > D$ (реализация не принадлежит области класса X_1^o), то при нечетном испытании вычисляется коэффициент $K_2 := K_2 + 1$ (″своя″ реалтзация), а при четном – $K_4 := K_4 + 1$ (″чужая″ реализация);

б) сравнение (блок 13): $J > JM$ (обработано все реализации обучающей матрицы), тогда вычисляются оценки точностных характеристик по (14), иначе обрабатывается следующая реализация (блок 4);

в) при выполнении условия блока 15 : ($D_1 > 0{,}5$ *и* $D_2 > 0{,}5$) вычисляется информационный критерий, иначе выдается сообщение «КФЕ не определен».

Задание 1. Вычислить значение точностных характеристик.
Задание 2. Программно реализовать функцию вычисления значения критерия функциональной эффективности по Кульбаку при указанных геометрических параметрах классов обучения.
Задание 3. Программно реализовать функцию вычисления значения критерия функциональной эффективности по Шеннону при указанных геометрических параметрах классов обучения.
Задание 4. Выделить рабочую область определения критерия функциональной эффективности на графике и в таблице.

Порядок выполнения работы

1. Вычислить значения точностных характеристик при указанных геометрических параметрах классов обучения:

$$D_1^{(k)}=\frac{K_1^{(k)}}{n},\quad \alpha^{(k)}=\frac{K_2^{(k)}}{n},\quad \beta^{(k)}=\frac{K_3^{(k)}}{n},\quad D_2^{(k)}=\frac{K_4^{(k)}}{n}.$$

2. Программно реализовать функцию вычисления значения критерия функциональной эффективности при указанных геометрических параметрах классов обучения. В качестве критерия использовать критерий Кульбака:

$$J_m^{(k)}=\frac{1}{n}\log_2\left\{\frac{2n+10^{-r}-\left[K_2^{(k)}+K_3^{(k)}\right]}{\left[K_2^{(k)}+K_3^{(k)}\right]+10^{-r}}\right\}*\left[n-(K_2^{(k)}+K_3^{(k)})\right],\tag{9}$$

$K_1^{(k)}$, $K_2^{(k)}$ – количество событий, которые означают соответственно принадлежность и непринадлежность реализаций образа контейнеру $K_{1,k}^{o}$, если действительно

$\{x_1^{(j)}\} \in X_1^o$;

$K_3^{(k)}$, $K_4^{(k)}$ – количество событий, которые означают соответственно принадлежность и непринадлежность реализаций контейнеру $K_{1,k}^o$, если они на самом деле принадлежат классу X_2^o;

k – шаг обучения системы распознавания;

r – число цифр в мантиссе значения критерия;

n – количество реализаций.

3. Программно реализовать функцию вычисления значения критерия функциональной эффективности при указанных геометрических параметрах классов обучения. Использовать информационный критерий Шеннона для двохальтернативного решение:

$$E_1^{(k)} = 1 + \frac{1}{2}\left(\frac{K_1^{(k)}}{K_1^{(k)} + K_3^{(k)}} \log_2 \frac{K_1^{(k)}}{K_1^{(k)} + K_3^{(k)}} + \frac{K_2^{(k)}}{K_2^{(k)} + K_4^{(k)}} \log_2 \frac{K_2^{(k)}}{K_2^{(k)} + K_4^{(k)}} + \right.$$

$$\left. + \frac{K_3^{(k)}}{K_1^{(k)} + K_3^{(k)}} \log_2 \frac{K_3^{(k)}}{K_1^{(k)} + K_3^{(k)}} + \frac{K_4^{(k)}}{K_2^{(k)} + K_4^{(k)}} \log_2 \frac{K_4^{(k)}}{K_2 + K_4^{(k)}} \right).$$

4. Построить графики зависимости информационного критерия от параметров оптимизации.

5. Провести проверку и анализ полученных результатов.

Пример функции вычисления точностных характеристик и значения критерия функциональной эффективности

```
/* Указываются изменения в классе Form1 по сравнению с
предыдущей работой */
public partial class Form1 : Form
    {
Запускаем процесс обучения по нажатию на кнопку
private void button2_Click(object sender, EventArgs e)
        {
```

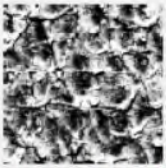
```
            kulbak(distanse11, distanse12, "first_opt.xlsx",
textBox11, ref optimMaximum1);
            kulbak(distanse22, distanse21, "second_opt.xlsx",
textBox9, ref optimMaximum2);
        }
Процесс обучения по для одной из картинок
        void kulbak(int[] distance1, int[] distance2, string
name, TextBox textBox, ref int optim)
        {
            int[] k1 = new int[100];
            int[] k2 = new int[100];
            int[] k3 = new int[100];
            int[] k4 = new int[100];
            float[] D1 = new float[100];
            float[] a = new float[100];
            float[] b = new float[100];
            float[] D2 = new float[100];
            double[] J = new double[100];
            for (int i = 0; i < 100; i++)
            {
                k1[i] = 0;
                k2[i] = 0;
                k3[i] = 0;
                k4[i] = 0;
                D1[i] = 0;
                D2[i] = 0;
                a[i] = 0;
                b[i] = 0;
            }
            for (int i = 0; i < 100; i++)
            {
                for (int j = 0; j < 100; j++)
                {
                    if (distance1[j] <= i)
                        k1[i] += 1;
                    else
                        k3[i] += 1;
                    if (distance2[j] <= i)
                        k2[i] += 1;
                    else
                        k4[i] += 1;
                }
                D1[i] = k1[i] / 100f;
                a[i] = k3[i] / 100f;
```

```
                b[i] = k2[i] / 100f;
                D2[i] = k4[i] / 100f;
            }
            for(int i=0; i < 100; i++)
            {
                if (D1[i] >= 0.5f && D2[i] >= 0.5f )
                {
                    double newa = a[i];
                    double newb = b[i];

                    if (newa + newb == 0)
                    {
                        J[i] = J[i - 1];
                        continue;
                    }

                    double tmp = (Math.Log((2 - (newa + newb)) /
(newa + newb)) / Math.Log(2)) * (1 - (newa + newb));
                    J[i] = tmp;
                }
                else
                    J[i] = 0;
            }
            double max = J.Max();
            optim = Array.FindLastIndex(J, delegate (double i) {
return i == max; });
Вывод результата в Excel
            textBox.Text = max.ToString();
            WorkBook workbook =
WorkBook.Load($@"{Directory.GetCurrentDirectory()}\test.xlsx");
            WorkSheet sheet = workbook.WorkSheets.First();
            int k = 0;
            foreach(var cell in sheet["A2:A" + (dist + 1)])
            {
                cell.Value = k;
                k++;
            }
            k = 0;
            foreach (var cell in sheet["B2:B" + (dist + 1)])
            {
                cell.Value = D1[k];
                k++;
            }
            k = 0;
```

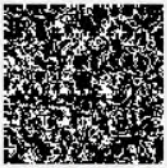
```
        foreach (var cell in sheet["C2:C" + (dist + 1)])
        {
            cell.Value = D2[k];
            k++;
        }
        k = 0;
        foreach (var cell in sheet["D2:D" + (dist + 1)])
        {
            cell.Value = a[k];
            k++;
        }
        k = 0;
        foreach (var cell in sheet["E2:E" + (dist + 1)])
        {
            cell.Value = b[k];
            k++;
        }
        k = 0;
        foreach (var cell in sheet["F2:F" + (dist + 1)])
        {
            cell.Value = J[k];
            k++;
        }
        workbook.SaveAs(name);
    }
```

Графическое отображение значения критерия функциональной эффективности

20	0,01	1	0,99	0	0
21	0,02	1	0,98	0	0
22	0,03	1	0,97	0	0
23	0,06	1	0,94	0	0
24	0,08	1	0,92	0	0
25	0,1	1	0,9	0	0
26	0,15	1	0,85	0	0
27	0,16	0,99	0,84	0,01	0
28	0,21	0,99	0,79	0,01	0
29	0,29	0,99	0,71	0,01	0
30	0,35	0,99	0,65	0,01	0
31	0,42	0,99	0,58	0,01	0
32	0,48	0,97	0,52	0,03	0
33	0,53	0,97	0,47	0,03	0,792481
34	0,59	0,95	0,41	0,05	0,941341
35	0,66	0,94	0,34	0,06	1,2
36	0,71	0,93	0,29	0,07	1,400081
37	0,77	0,93	0,23	0,07	1,75175
38	0,81	0,91	0,19	0,09	1,885615



Бинарная:



Эталонный вектор:

158 151 71 33 31 29 32 30 30 32 38
40 32 26 33 42 39 35 62 126 142 137
144 122 144 155 163 143 125 141 77
39 36 39 33 37 45 39 30 33 42 38 49
94 187 182 159 124 74 76 74 55 72
123 125 103 39 23 97 162 140 131
73 29 39 38 23 46 92 115 185 203
129 47 26 21 24 33 85 146 159 139
122 132 142 124 99 102 86 52 13 6 4
5 16 26 37 43 29 31
130 118 65 41 41 38 42 45 40 41 39
37 34 23 21 33 43 40 70 132 171 164
127 94 117 137 128 104 89 99 51 32
31 36 29 26 24 29 31 31 45 39 44 91
199 201 169 120 51 30 19 13 38 90
111 112 52 23 84 126 107 94 55 65
116 107 56 76 99 89 80 79 48 24 25
32 25 33 62 110 113 73 53 105 152
160 140 128 110 65 21 8 4 6 9 16 24
28 27 30
53 51 52 49 44 42 46 53 48 46 34 32
24 16 15 27 38 50 56 104 161 117 46
26 34 55 49 35 37 47 36 31 31 34 37

0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0
0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1
0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1
0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0
0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0
1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1
0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0
0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1
0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1
1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0
0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0 0
1 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Матрица расстояний:

С 1 вектором :
46 46 39 58 46 43 43 46 49 48 35 40 45 52 48 46 49 47 44 41 54 43 42 49 47 37 45
47 47 33 50 53 46 47 44 39 42 43 45 60 49 50 51 38 35 46 50 46 49 56 50 52 52 45
56 37 45 46 44 47 44 49 42 40 40 43 52 47 40 36 42 41 41 35 47 46 39 51 54 49 49
41 44 56 54 45 47 40 37 48 46 45 49 55 48 45 41 47 51 56

Со 2 вектором :
31 31 30 39 29 26 30 29 36 45 30 27 34 35 39 27 40 36 41 34 29 24 25 28 30 22 32
36 30 32 27 32 33 34 37 28 39 40 40 43 48 37 34 39 32 17 29 33 24 35 29 29 27 34

KFE =
2,4422836918E

20	0,04	1	0,96	0	0
21	0,04	1	0,96	0	0
22	0,07	1	0,93	0	0
23	0,08	1	0,92	0	0
24	0,15	1	0,85	0	0
25	0,2	1	0,8	0	0
26	0,26	1	0,74	0	0
27	0,33	1	0,67	0	0
28	0,38	1	0,62	0	0
29	0,45	1	0,55	0	0
30	0,51	1	0,49	0	0,828084
31	0,56	1	0,44	0	1,022544
32	0,63	1	0,37	0	1,347743
33	0,66	0,99	0,34	0,01	1,454075
34	0,72	0,99	0,28	0,01	1,817509
35	0,77	0,96	0,23	0,04	1,956211
36	0,81	0,95	0,19	0,05	2,184597
37	0,86	0,92	0,14	0,08	2,352715
38	0,87	0,91	0,13	0,09	2,352715

Контрольные вопросы

1. Оценка функциональной эффективности интеллектуальной системы.
2. Алгоритм вычисления информационного критерия функциональной эффективности обучения системы распознавания.
3. Как вычисляется первая (вторая) достоверности распознавания?
4. Как вычисляются ошибки первого и второго рода?
5. Обоснуйте целесообразность использования информационного критерия функциональной эффективности для оценки функциональной эффективности обущающей интеллектуальной системы.
6. Как вычисляется рабочая область определения критерия функциональной эффективности

Практическое занятие 6. Оптимизация системы контрольных допусков на признаки распознавания

Цель: разработать и программно реализовать алгоритм оптимизации геометрических параметров контейнеров классов распознавания.

Теоретические сведения

Рассмотрим алгоритм параллельной оптимизации системы контрольных допусков на признаки распознавания в рамках метода функционально-статистических испытаний [2]. На рис. 3 показано симметричное (двустороннее) поле допусков на значение признаков распознавания $y_{m,i}^{(j)}, i = \overline{1,N}$.

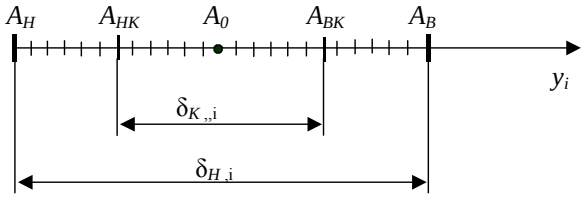


Рисунок 3. Симметричное поле допусков на значения признаков распознавания

Здесь A_0 – номинальное значение признака y_i;

A_H, A_B – нижний и верхний нормированные допуски соответственно;

A_{HK} , A_{BK} – нижний и верхний контрольные допуски соответственно;

$\delta_{H,i}$ – нормированное поле допусков;

$\delta_{K,i}$ – контрольное поле допусков.

Существует несколько возможных стратегий изменения поля допусков $\delta_{K,i}$, среди которых выделим две основные:

- симетричная стратегия $S_1(\overrightarrow{\leftarrow}\operatorname{var} A_{HK}, \overleftarrow{\rightarrow}\operatorname{var} A_{BK})$, которая оправдана, например, при условии подтверждения разведывательным анализом совпадения номинального значения A_0 с теоретическим центром рассеивания значений обучающейся выборки $\left\{y_{m,i}^{(j)} \vee j = \overline{1,n}\right\}$;

- асимметричная стратегия $S_2(\overleftarrow{\rightarrow}\operatorname{var} A_{HK}, \overleftarrow{\rightarrow}\operatorname{var} A_{BK})$, которая имеет место при отклонении значения A_0 от центра рассеивания значений выборки $\left\{y_{m,i}^{(j)} \vee j = \overline{1,n}\right\}$.

Задача оптимизации контрольных допусков на признаки распознавания является частичной задачей информационного синтеза, в которой необходимо определить экстремальные значения параметра поля контрольных допусков $\delta = |A_0 - A_{HK}|$:

$$\delta^* = \arg \max_{G_\delta} \{ \max_{G_\Omega} \{ \max_{G_d} \overline{E} \} \},$$

где G_δ, G_Ω, G_d – допустимые области значений параметра поля контрольных допусков δ, пространства признаков распознавания и радиусов контейнеров классов распознавания соответственно.

Алгоритм оптимизации контрольных допусков, как и других параметров обучения в рамках метода функционально-статистических испытаний, заключается в приближении глобального максимума информационного критерия оптимизации к предельному его значению в рабочей области значений функции критерия.

Задание 1. Разработать алгоритм последовательной оптимизации системы контрольных допусков на признаки распознавания. На каждом шаге оптимизации вычислить оптимальные значения геометрических параметров плана обучения.

Задание 2. Разработать систему визуализации процесса оптимизации системы контрольных допусков и геометрических параметров плана обучения.

Порядок выполнения работы

1. В качестве входных данных использовать бинарную обучающуюся матрицу яркости $\{X[J,I,K]\}$ и эталонные векторы-реализации изображений.
2. Определить область значений параметра оптимизации $\delta \in [0;\ \delta_H / 2]$, где δ_H – нормированное (эксплуатационное) поле допусков. Для черно-белых изображений рекомендуется выбирать $\delta_{\max} \geq 20$ градаций яркости.
3. Программно реализовать итерационную процедуру параллельной оптимизации системы контрольных допусков по информационному критерию Шеннона.
4. Построить графики зависимости информационного критерия от параметров оптимизации. Провести проверку и анализ полученных результатов .

Пример функции оптимизации системы контрольных допусков и радиусов контейнеров классов распознавания

```
Оптимизация параметров обучения по нажатию на кнопку
private void button2_Click(object sender, EventArgs e)
        {
            kulbak(distanse11, distanse12, "first_opt.xlsx",
textBox11, ref optimMaximum1);
```

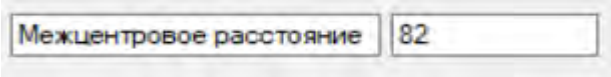
```
            kulbak(distanse22, distanse21, "second_opt.xlsx",
textBox9, ref optimMaximum2);
        }
        void kulbak(int[] distance1, int[] distance2, string
name, TextBox textBox, ref int optim)
        {
            int[] k1 = new int[100];
            int[] k2 = new int[100];
            int[] k3 = new int[100];
            int[] k4 = new int[100];
            float[] D1 = new float[100];
            float[] a = new float[100];
            float[] b = new float[100];
            float[] D2 = new float[100];
            double[] J = new double[100];
            for (int i = 0; i < 100; i++)
            {
                k1[i] = 0;
                k2[i] = 0;
                k3[i] = 0;
                k4[i] = 0;
                D1[i] = 0;
                D2[i] = 0;
                a[i] = 0;
                b[i] = 0;
            }
            for (int i = 0; i < 100; i++)
            {
                for (int j = 0; j < 100; j++)
                {
                    if (distance1[j] <= i)
                        k1[i] += 1;
                    else
                        k3[i] += 1;
                    if (distance2[j] <= i)
                        k2[i] += 1;
                    else
                        k4[i] += 1;
                }
                D1[i] = k1[i] / 100f;
                a[i] = k3[i] / 100f;
                b[i] = k2[i] / 100f;
                D2[i] = k4[i] / 100f;
            }
```

```
            for(int i=0; i < 100; i++)
            {
                if (D1[i] >= 0.5f && D2[i] >= 0.5f )
                {
                    double newa = a[i];
                    double newb = b[i];

                    if (newa + newb == 0)
                    {
                        J[i] = J[i - 1];
                        continue;
                    }

                    double tmp = (Math.Log((2 - (newa + newb)) /
(newa + newb)) / Math.Log(2)) * (1 - (newa + newb));
                    J[i] = tmp;
                }
                else
                    J[i] = 0;
            }
            double max = J.Max();
            optim = Array.FindLastIndex(J, delegate (double i) {
return i == max; });
Вывод результата в Excel
            textBox.Text = max.ToString();
            WorkBook workbook =
WorkBook.Load($@"{Directory.GetCurrentDirectory()}\test.xlsx");
            WorkSheet sheet = workbook.WorkSheets.First();
            int k = 0;
            foreach(var cell in sheet["A2:A" + (dist + 1)])
            {
                cell.Value = k;
                k++;
            }
            k = 0;
            foreach (var cell in sheet["B2:B" + (dist + 1)])
            {
                cell.Value = D1[k];
                k++;
            }
            k = 0;
            foreach (var cell in sheet["C2:C" + (dist + 1)])
            {
                cell.Value = D2[k];
```

```
            k++;
        }
        k = 0;
        foreach (var cell in sheet["D2:D" + (dist + 1)])
        {
            cell.Value = a[k];
            k++;
        }
        k = 0;
        foreach (var cell in sheet["E2:E" + (dist + 1)])
        {
            cell.Value = b[k];
            k++;
        }
        k = 0;
        foreach (var cell in sheet["F2:F" + (dist + 1)])
        {
            cell.Value = J[k];
            k++;
        }
        workbook.SaveAs(name);
    }
```

Оптимизация проходит за счет изменения delta и ro на форме

Графическое отображение процесса оптимизации системы контрольных допусков и геометрических параметров контейнеров классов обучения



31	0,58	1	0,42	0	1,108649
32	0,67	1	0,33	0	1,567338
33	0,71	1	0,29	0	1,817509
34	0,77	1	0,23	0	2,266914
35	0,81	1	0,19	0	2,634054
36	0,86	1	0,14	0	3,209351
37	0,92	1	0,08	0	4,218166
38	0,95	1	0,05	0	5,021132
39	0,95	1	0,05	0	5,021132
40	0,95	1	0,05	0	5,021132
41	0,97	1	0,03	0	5,855977
42	0,98	1	0,02	0	6,49677
43	0,98	1	0,02	0	6,49677
44	1	1	0	0	6,49677
45	1	0,99	0	0,01	7,560258
46	1	0,99	0	0,01	7,560258
47	1	0,99	0	0,01	7,560258
48	1	0,99	0	0,01	7,560258
49	1	0,99	0	0,01	7,560258
50	1	0,97	0	0,03	5,855977
51	1	0,97	0	0,03	5,855977
52	1	0,95	0	0,05	5,021132
53	1	0,95	0	0,05	5,021132
54	1	0,93	0	0,07	4,450145

38	0,4	1	0,6	0	0
39	0,47	1	0,53	0	0
40	0,51	1	0,49	0	0,828084
41	0,55	1	0,45	0	0,981349
42	0,58	1	0,42	0	1,108649
43	0,62	0,99	0,38	0,01	1,247764
44	0,64	0,97	0,36	0,03	1,247764
45	0,69	0,96	0,31	0,04	1,454075
46	0,72	0,92	0,28	0,08	1,400081
47	0,73	0,9	0,27	0,1	1,347743
48	0,77	0,85	0,23	0,15	1,296992
49	0,78	0,82	0,22	0,18	1,2
50	0,83	0,79	0,17	0,21	1,296992
51	0,87	0,76	0,13	0,24	1,347743
52	0,93	0,7	0,07	0,3	1,347743
53	0,93	0,65	0,07	0,35	1,108649
54	0,95	0,6	0,05	0,4	0,981349
55	0,98	0,57	0,02	0,43	0,981349
56	1	0,51	0	0,49	0,828084

Контрольные вопросы

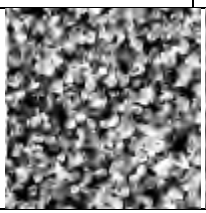
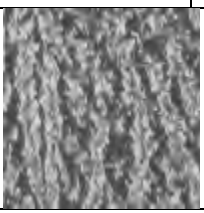
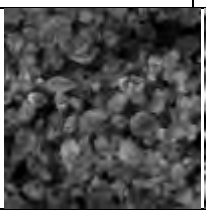
1. Что такое «рабочая область» при оптимизации геометрических параметров классов распознавания? Какова ее роль в этом процессе?
2. Как отличаются значение оптимальных геометрических параметров классов обучения, полученные с использованием различных информационных критериев?

3. Как определить область значений параметров оптимизации?
4. В чем заключается алгоритм оптимизации контрольных допусков на признаки распознавания?
5. Какие существуют стратегии изменения поля допусков $\delta_{K,i}$?
6. Какие компоненты C# были использованы при создании системы визуализации процесса оптимизации геометрических параметров плана обучения?

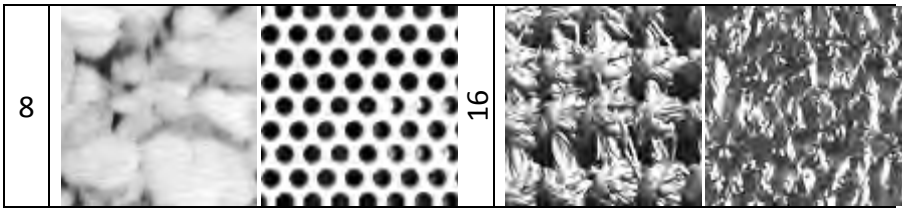
Практическое занятие 7. Реализация алгоритма обучения интеллектуальной системы

Цель: разработать и программно реализовать алгоритм обучения с оптимизацией контрольных допусков системы распознавания.

Задача 1. Провести обучение с оптимизацией системы контрольных допусков и геометрических параметров классов распознавания. Обучающуюся матрицу сформировать, используя приведенные в таблице текстуры.

Номер	1-й класс	2-й класс	Номер	1-й класс	2-й класс
1			9		

2
10
3
11
4
12
5
13
6
14
7
15



Порядок выполнения работы

1. Записать тему и цель работы.
2. Провести обучение с оптимизацией системы контрольных допусков и геометрических параметров классов распознавания.
3. Провести проверку и анализ полученных результатов.

Контрольные вопросы

1. Как вычислить оптимальные значения геометрических параметров классов и системы контрольных допусков на признаки распознавания?
2. Как получить графические изображения бинарных обучающих матриц при начальных и оптимальных параметрах обучения?
3. Как происходит изменение критерия функциональной эффективности в процессе оптимизации системы контрольных допусков и геометрических параметров классов распознавания?

Практическое занятие 8. Реализация алгоритма экзамена

Цель: разработать и программно реализовать алгоритм экзамена системы распознавания.

Теоретические сведения

Алгоритмы экзамена по методу функционально-статистических испытаний [2] могут иметь разную структуру в зависимости от распределения реализаций образа. Обязательным условием их реализации является обеспечение равных условий структурированности и параметров формирования как для обучающей, так и для экзаменационной матриц.

Реализации алгоритма экзамена:

1.Формирование счетчика $m := m + 1$ классов распознавания.

2. Формирование счетчика числа реализаций, которые распознаются $j := j + 1$.

3. Вычисления кодового расстояния $d(x_m^* \oplus x^{(j)})$.

4. Вычисления функции принадлежности:

$$\mu_m = 1 - \frac{d(x_m^* \oplus x^{(j)})}{d_m^*}.$$

5. Сравнения: если *j* ≤ *n*, то выполняется шаг 2, иначе - шаг 6.

6. Сравнения: если m ≤ M, то выполняется шаг 1, иначе- шаг 7.

7.Определение класса X_m^o, к которому принадлежит экзаменационная реализация, например,при условии

$$\overline{\mu}_m^* = \max_{\{m\}} \overline{\mu}_m,$$

где $\overline{\mu}_m = \frac{1}{\mathrm{n}} \sum_{j=1}^{n} \mu_{\mathrm{m,j}}$ – усредненное значение функции принадлежности для реализаций класса X_m^o, или выдача

уведомления : «Класс не определен», если $\overline{\mu}_m \leq c$, где c – пороговое значение.

Задание 1. Сформировать экзаменационную матрицу.
Задание 2. Программно реализовать алгоритм экзамена интеллектуальной системы.

Порядок выполнения работы

1. Сформировать экзаменационную матрицу. Для формирования можно использовать обучающиеся матрицы, которые были определенным образом деформированы.
2. Программно реализовать алгоритм экзамена интеллектуальной системы.

Пример функции этапа экзамена

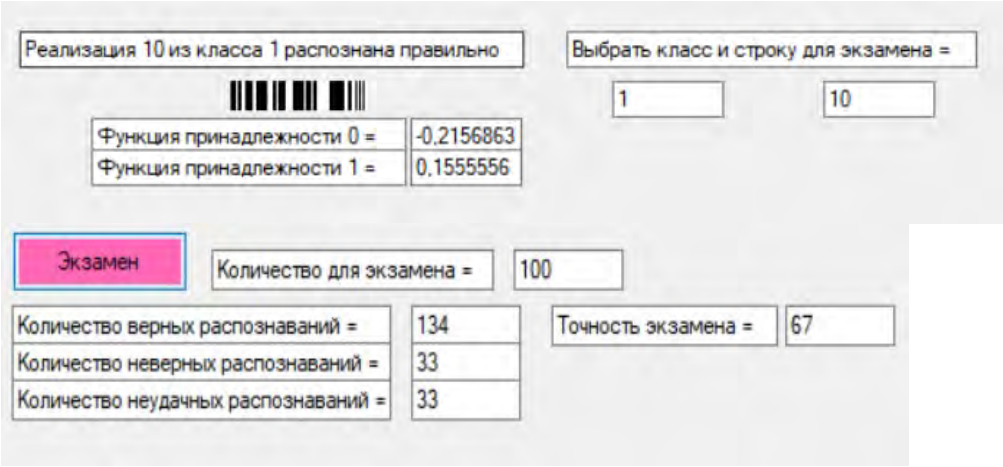
```
/* Указываются изменения в классе Form1 по сравнению с
предыдущей работе */
public partial class Form1 : Form
    {
     int optimMaximum1, optimMaximum2;
                    Нажатие по кнопке Экзамен

private void button3_Click(object sender, EventArgs e)
        {
            int correct = 0, incorrect = 0;
            exam(numOfEx, ref correct, ref incorrect);
            textBox15.Text = correct.ToString();
            textBox17.Text = incorrect.ToString();
            textBox19.Text = ((numOfEx * 2) - correct -
incorrect).ToString();
            textBox21.Text = (100 * correct / (numOfEx *
2)).ToString();
        }
                          Рассчет экзамена
public void exam(int numOfChecking, ref int correct, ref int
incorrect)
```

```
        {
            bool flag = true;
            int[][,] binaries = new int[][,]
            {
                pict1, pict2
            };
            for (int m = 0; m < 2; m++)
            {
                for (int j = 0; j < numOfChecking; j++)
                {
                    int distance1 = 0;
                    for (int i = 0; i < 100; i++)
                    {
                        distance1 += vector1[i] ==
binaries[m][i, j] ? 0 : 1;
                    }
                    int distance2 = 0;
                    for (int i = 0; i < 100; i++)
                    {
                        distance2 += vector2[i] ==
binaries[m][i, j] ? 0 : 1;
                    }
                    float nu1 = 1 - (float)distance1 /
optimMaximum1;
                    float nu2 = 1 - (float)distance2 /
optimMaximum2;
                    float maxNu = Math.Max(nu1, nu2);
                    if (maxNu >= 0)
                    {
                        if ((maxNu == nu1 && m == 0) || (maxNu
== nu2 && m == 1))
                        {
                            correct += 1;
                            if (textBox32.Text == (j).ToString()
&& (m).ToString() == textBox30.Text)
                            {
                                flag = false;
                                textBox25.Text = nu2.ToString();
                                textBox27.Text = nu1.ToString();
                                textBox29.Text = "Реализация " +
(j) + " из класса " + (m) + " распознана правильно";
                                Bitmap image1 = new Bitmap(100,
20);
```

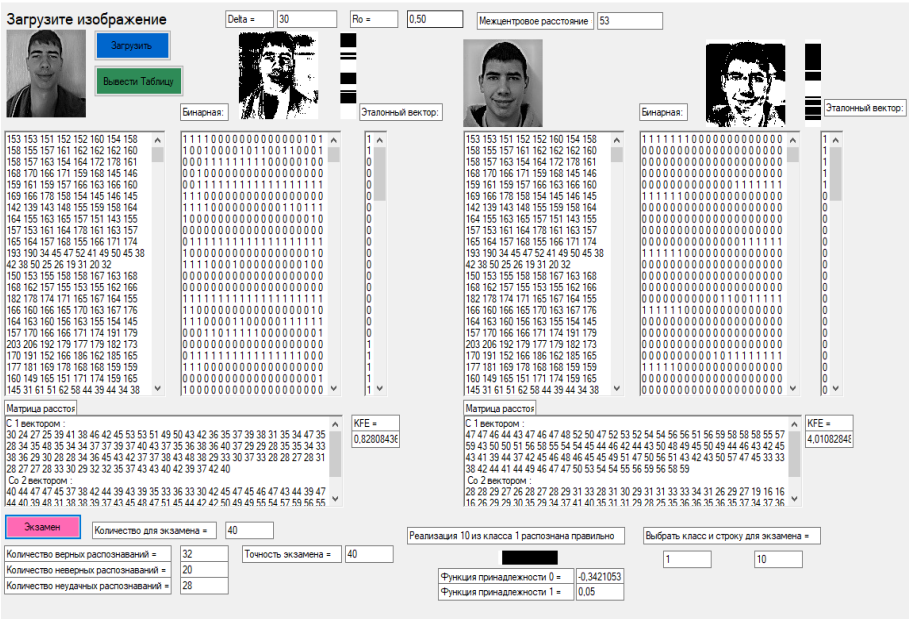
```
                                for (int x = 0; x <
pb_mas[0].Image.Height; x++)
                                {
                                    for (int y = 0; y < 20; y++)
                                    {
                                        image1.SetPixel(x, y,
Color.FromArgb(255 * binaries[m][j, x], 255 * binaries[m][j, x],
255 * binaries[m][j, x]));
                                    }

                                }
                                pictureBox3.Image = image1;
                            }
                        }
                        else
                        {
                            incorrect += 1;
                                if (textBox32.Text ==
(j).ToString() && (m).ToString() == textBox30.Text)
                                {
                                    textBox25.Text =
nu2.ToString();
                                    textBox27.Text =
nu1.ToString();
                                    textBox29.Text = "Реализация
" + (j) + " из класса " + (m) + " распознана не правильно";
                                    Bitmap image1 = new
Bitmap(100, 20);
                                    for (int x = 0; x <
pb_mas[0].Image.Height; x++)
                                    {
                                        for (int y = 0; y < 20;
y++)
                                        {
                                            image1.SetPixel(x,
y, Color.FromArgb(255 * binaries[m][j, x], 255 * binaries[m][j,
x], 255 * binaries[m][j, x]));
                                        }

                                    }
                                    pictureBox3.Image = image1;
                                }
                            }
                        }
```

```
                }
            }
}
```

Графическое отображение этапа экзамена



Пример работы программы на реальных изображениях людей



Точность составила всего 40%, это связано с тем, что алгоритм работает в основном со стационарными изображениями.

Контрольные вопросы

1. Какое изображение называется стационарным (нестационарным) по яркости?
2. Как формируется эталонный вектор-реализация для обучающей матрицы?
3. Как формируется бинарная обучающая матрица для изображения классов?
4. Объясните преимущество обработки изображений в дискретном пространстве признаков распознавания.
5. Как найти межцентровое кодовое расстояние для классов, контейнеры которых построены в радиальном базисе?

6. Какую структуру имеет входное математическое описание системы распознавания изображений?
Как вычисляется функция принадлежности?

Список литературы

1. Указ Президента РФ от 9 мая 2017 г. № 203 «О Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы».
2. Проектирование систем управления на ЭВМ //А. Ю. Соколов, Ю. Н. Соколов, В. М. Ильюшко, М. М. Митрахович, Д. Н. Гайсёнок // под ред. Ю. Н. Соколова. – Харьков : «ХАИ», 2005. – 590 с.
2. Скаковская А.Н. Функционально-статистический метод управления растровым электронным микроскопом // Радіоелектронні і комп'ютерні системи. – Харків: НАКУ ім. М.Є.Жуковського «ХАІ», 2007. №2(21). С.16-20.
3. *Соколов А.Ю*. Интеллектуальные методы управления растровым электронным микроскопом на основе функционально-статистического подхода. / А.Ю.Соколов, А.Н.Скаковская // Матеріали 14-ї Міжнар. конф. з автоматичного управління «Автоматика-2007». – Севастополь: СНУЯЄтаП, 2007. – С.176-178.
4. Общая характеристика задач распознавания образов и их типы. http://www.intuit.ru/department/human/isrob/4/#image.4.1
5. Искусственный интеллект. Базы знаний и экспертные системы: Учебн. пособ. / А.М. Дворянкин, А.В. Кизим, И.Г. Жукова и др. // – Волгоград: Политехник, – 2003. –139 с.
6. Сироджа И.Б. Модели и методы искусственного интеллекта для идентификации и управления динамическими объектами / Сироджа И.Б., Соколов А.Ю., Мокляк Н.Г., Резниченко О.В. // Авиационно- космическая техника и технология. – Харьков: ХАИ. – 1998. – С.406-411.
7. Гонсалес Р. Цифровая обработка изображений. / Гонсалес Р., Вудс Р. // – М.: Техносфера, – 2005. – С.1110-1148
8. .NET Framework Developer's Guide. Reflection. – Microsoft Developers Network. http://msdn.microsoft.com/en-us/library/cxz4wk15. aspx.

9. Windows Forms. — Домашняя страница документации по Windows Forms .NET.
http://ru.wikipedia.org/wiki/Windows_Forms.