Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Методы и алгоритмы принятия решений (МиАПР)

ОТЧЕТ

по лабораторной работе №4

по теме:

«Классификация объектов на N классов методом персептрона»

Выполнил

студент: гр. 851006 Верещагин Н.В.

Проверил: Марина И.М.

Минск 2021

Содержание

[1 Постановка задачи 3](#_Toc64459015)

[1.1 Цель работы 3](#_Toc64459016)

[1.2 Исходные данные 3](#_Toc64459017)

[1.3 Результат работы алгоритма 3](#_Toc64459018)

[2 Алгоритм 4](#_Toc64459019)

[3 Решение задачи 8](#_Toc64459020)

# Постановка задачи

## Цель работы

Изучить особенности классификации объектов методом персептрона, а также научиться применять этот метод на практике.

## Исходные данные

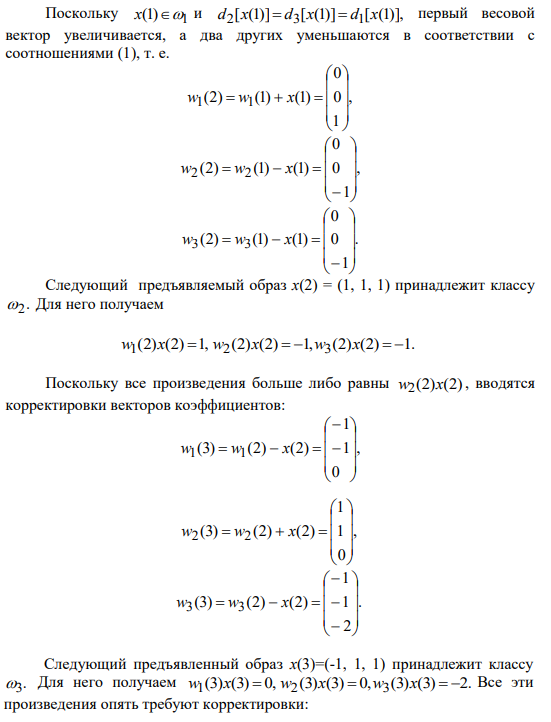
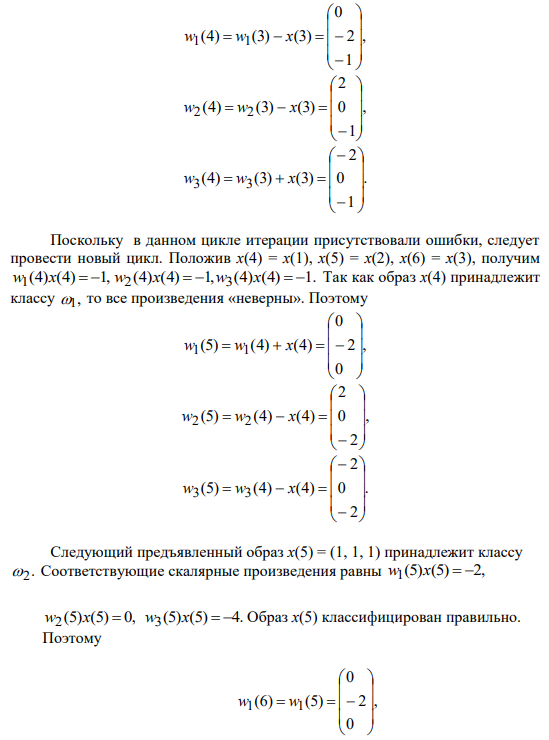
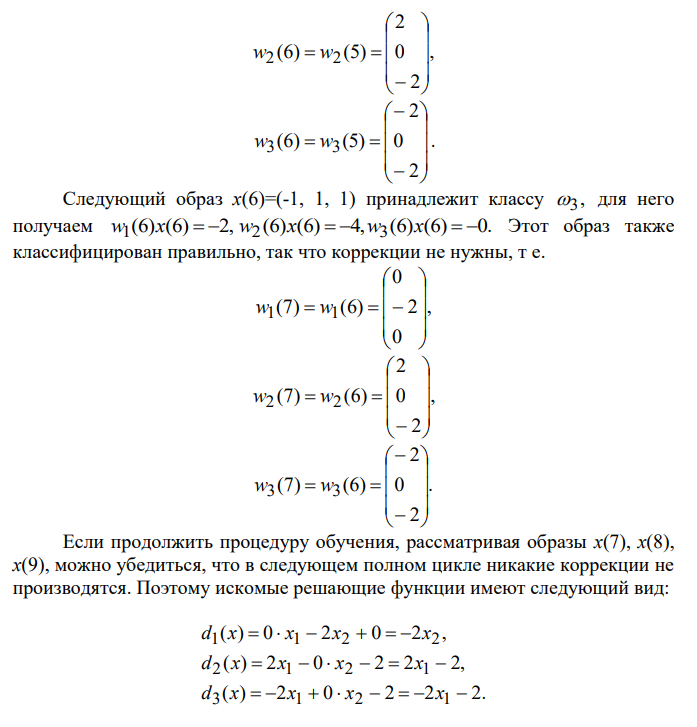
– N – Количество классов на которые требуется разделить объекты

– Обучающая выборка, представленная векторами с наборами признаков.

## Результат работы алгоритма

N –решающих функций.

# Алгоритм

# Решение задачи

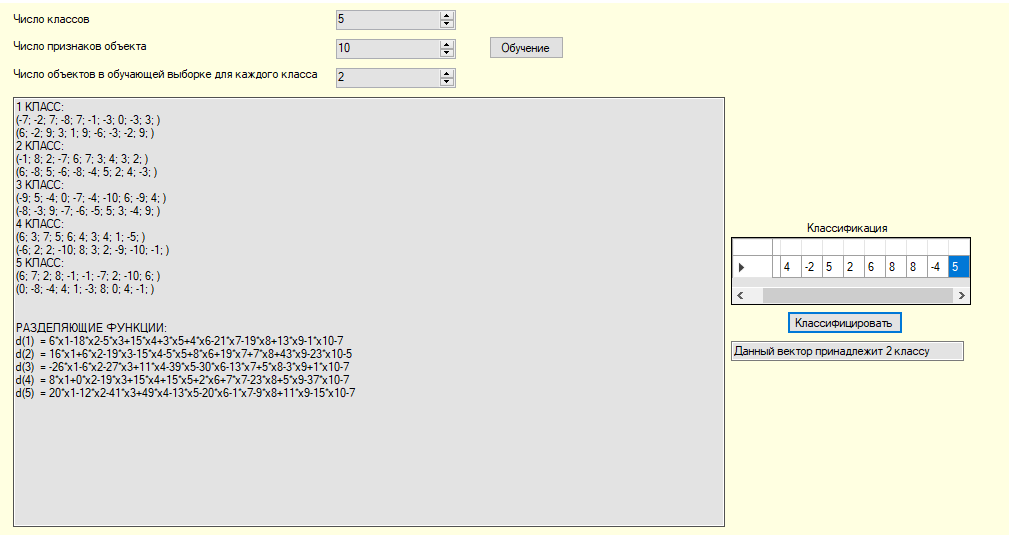


Рисунок 1 – Пример работы программы 1

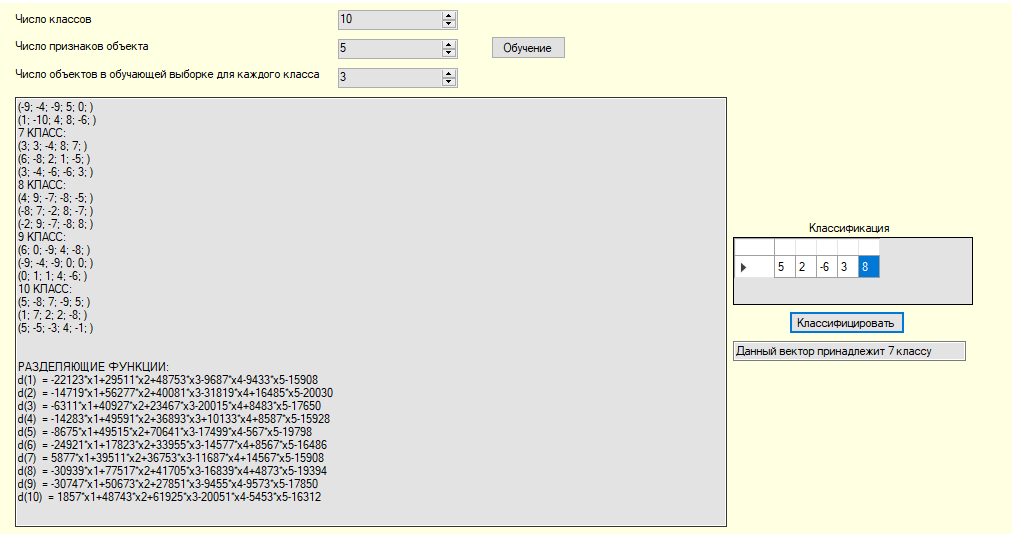


Рисунок 2 – Пример работы программы 2

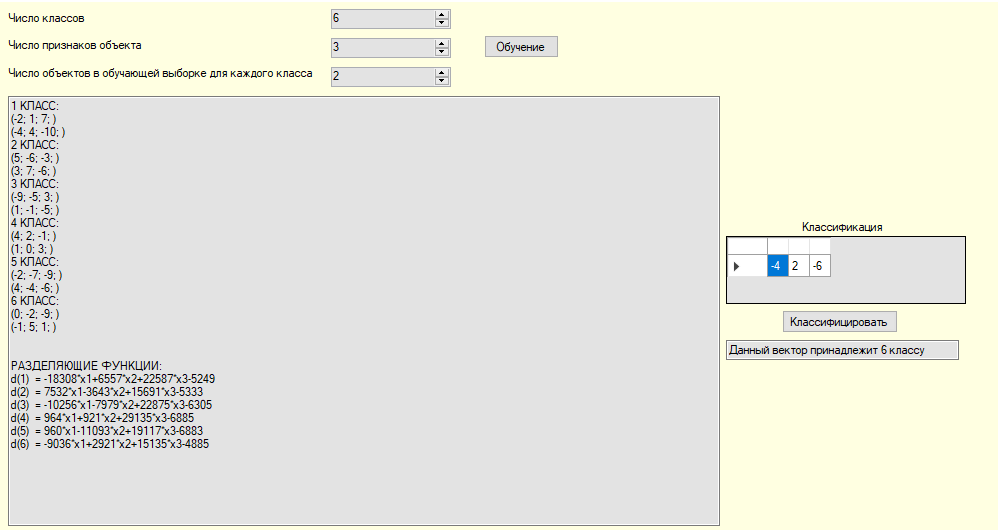


Рисунок 3 – Пример работы программы 3

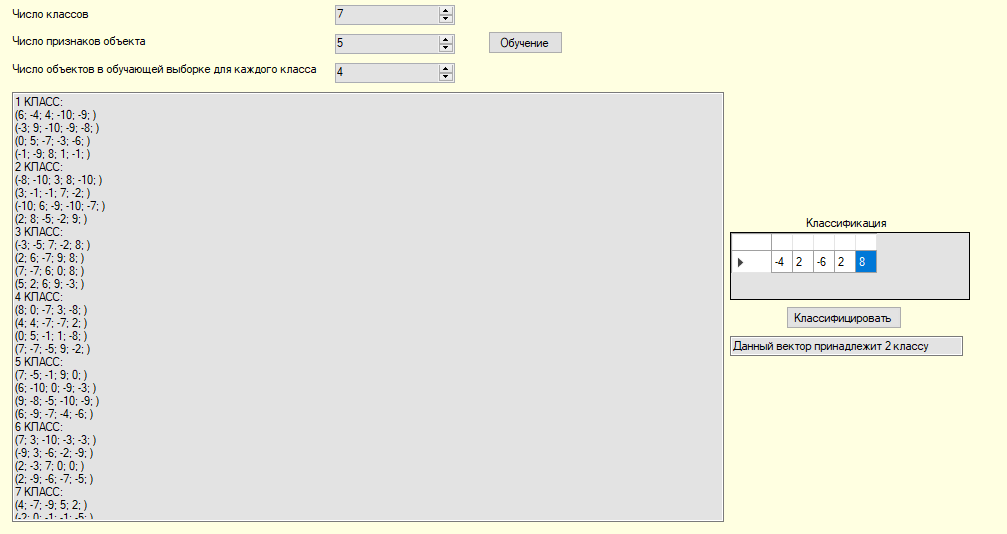


Рисунок 4 – Пример работы программы 4

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab4

{

    public class Persiptron

    {

        private readonly int classCount;

        private readonly int vectorsSize;

        public bool warningFlag;

        public Persiptron(int classCount, int vectorsSize)

        {

            this.classCount = classCount;

            this.vectorsSize = vectorsSize;

        }

        public Function[] GetSepareteFunctions(Vector[][] teachingVectors)

        {

            var result = EmptyFunctions();

            warningFlag = false;

            bool nextIteration = true;

            int iterationNumber = 0;

            while (nextIteration && iterationNumber < 1000)

            {

                iterationNumber++;

                nextIteration = DoOneIteration(teachingVectors, result);

            }

            if (iterationNumber == 1000) warningFlag = true;

            return result;

        }

        private Function[] EmptyFunctions()

        {

            var result = new Function[classCount];

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

            {

                result[i] = new Function(vectorsSize);

            }

            return result;

        }

        private bool DoOneIteration(Vector[][] teachingVectors, Function[] result)

        {

            bool nextIteration = false;

            if (teachingVectors.Length != classCount) throw new Exception();

            for (int classNumber = 0; classNumber < classCount; classNumber++)

            {

                for (int i = 0; i < teachingVectors[classNumber].Length; i++)

                {

                    if (WorkWithVector(teachingVectors[classNumber][i], result, classNumber))

                    {

                        nextIteration =true;

                    }

                }

            }

            return nextIteration;

        }

        private bool WorkWithVector(Vector currentVector, Function[] result, int vectorsClass)

        {

            var maxClass = GetMaxVectorClass(result, currentVector);

            if (maxClass != vectorsClass)

            {

                Panish(currentVector, result, vectorsClass);

                return true;

            }

            return false;

        }

        private void Panish(Vector currentVector, Function[] result, int vectorsClass)

        {

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

            {

                if (i == vectorsClass)

                {

                    result[i] += currentVector;

                }

                else

                {

                    result[i] += -1\*currentVector;

                }

            }

        }

        public int GetMaxVectorClass(Function[] result, Vector currentVector)

        {

            int max = result[0].GetValue(currentVector);

            int maxClass = 0;

            int maxCount = 1;

            for (int j = 1; j < classCount; j++)

            {

                int currentValue = result[j].GetValue(currentVector);

                if (currentValue > max)

                {

                    maxCount = 0;

                    max = currentValue;

                    maxClass = j;

                }

                if (currentValue == max)

                {

                    maxCount++;

                }

            }

            return maxCount == 1 ?  maxClass : -1;

        }

    }

}