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 №2

по теме:

«Распознавание образов на основе самообучения»

Выполнил

студент: гр. 851006 Верещагин Н.В.

Проверил: Марина И.М.

Минск 2021

Содержание

[1 Постановка задачи 3](#_Toc63740341)

[1.1 Цель работы 3](#_Toc63740342)

[1.2 Исходные данные 3](#_Toc63740343)

[1.3 Цели и результат работы алгоритма 3](#_Toc63740344)

[2 Алгоритм максимина 4](#_Toc63740345)

[3 Решение задачи 5](#_Toc63740346)

# Постановка задачи

## Цель работы

Изучить особенности распознавания образов в самообучающихся системах и научиться классифицировать объекты с помощью алгоритма *максимина*.

## Исходные данные

* Количество образов в диапазоне от 1000 до 100 000.
* Признаки объектов задаются случайным образом, это координаты векторов.

## Цели и результат работы алгоритма

Исходя из произвольного выбора максимально компактно разделить объекты на классы, определив ядро каждого класса.

# Алгоритм максимина

1. Из множества векторов *X* = {*X*(1), *X*(2), *X*(3), …, *X*(V)} произвольно выбирается один и назначается ядром первого класса. Пусть *N*1 = *X*(1). Затем будут определяться другие ядра *N*2, *N*3,…, *N*m, число которых заранее неизвестно.

2. Вычисляются расстояния Ядро *N*2 выбирается следующим образом: .

3. Выполняется распределение оставшихся объектов по классам по критерию минимального расстояния.

4. В каждом классе вычисляются расстояния от ядра до каждого объекта данного класса: , среди которых находятся небольшие (пока имеется два максимума).

5. Выбирается максимальное среди всех максимальных расстояний, которое становится претендентом на очередное ядро. Это значение . Если больше половины среднего арифметического расстояния между всеми ядрами, то создается очередное ядро и выполняется переход к шагу 3, иначе алгоритм останавливается.

# Решение задачи

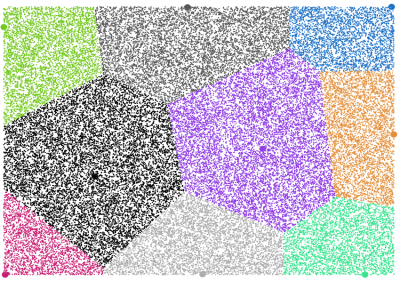


Рисунок 1 – Пример работы программы 1

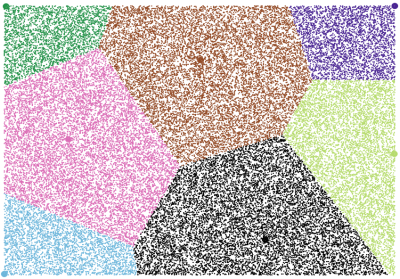


Рисунок 2 – Пример работы программы 2

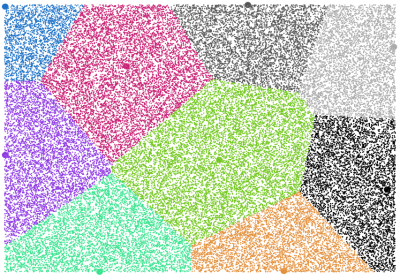


Рисунок 3 – Пример работы программы 3

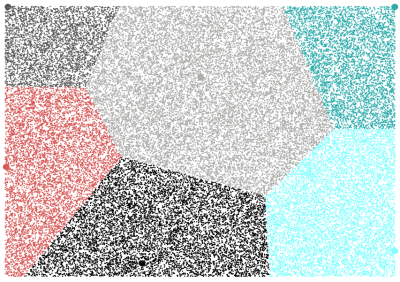


Рисунок 4 – Пример работы программы 4

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

namespace Algorithms

{

    public class MaxMin : AlgorithmBase

    {

        private readonly Random random = new Random();

        public MaxMin(IEnumerable<Point> points)

        {

            Points = new List<Point>(points);

            Point firstCenter = Points[random.Next(Points.Count)];

            Classes = new List<PointsClass> { new PointsClass(firstCenter) };

        }

        public List<PointsClass> GetReadyClasses()

        {

            Point? newCenter;

*do*

            {

                ClearClasses();

                AddPointsToClasses();

                newCenter = GetNewCenter();

                AddCenter(newCenter);

             } *while* (newCenter!= null);

*return* Classes;

        }

        private void AddCenter(Point? newCenter)

        {

*if* (newCenter != null)

            {

                Classes.Add(new PointsClass(newCenter.Value));

            }

        }

        private Point? GetNewCenter()

        {

            double averageCenterDistance = GetAverageCenterDistance();

            ClassMaxPoint newCenterCandidate = GetMaxPoint(GetClassesMaxPoints());

*if* (newCenterCandidate.PointDistance > averageCenterDistance/2)

            {

*return* newCenterCandidate.MaxPoint;

            }

*return* null;

        }

        private double GetAverageCenterDistance()

        {

            double distanceSum = 0.0;

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

            {

*for* (int j = i + 1; j < Classes.Count; j++)

                {

                    distanceSum += GetPointsInstance(Classes[i].Center, Classes[j].Center);

                }

            }

            int count = Enumerable.Range(1, Classes.Count - 1).Sum();

*return* count == 0 ? 0 : distanceSum/count;

        }

        private ClassMaxPoint GetMaxPoint(IEnumerable<ClassMaxPoint> points)

        {

            var maxPoint = new ClassMaxPoint { PointDistance = 0 };

*foreach* (var point *in* points)

            {

*if* (point.PointDistance > maxPoint.PointDistance)

                {

                    maxPoint = point;

                }

            }

*return* maxPoint;

        }

        private IEnumerable<ClassMaxPoint> GetClassesMaxPoints()

        {

*foreach* (PointsClass pointsClass *in* Classes)

            {

*yield* *return* GetClassMaxPoint(pointsClass);

            }

        }

        private ClassMaxPoint GetClassMaxPoint(PointsClass pointClass)

        {

            var maxPoint = new ClassMaxPoint {PointDistance = 0};

*foreach* (var point *in* pointClass.Points)

            {

                double pointDistanse = GetPointsInstance(point, pointClass.Center);

*if* (pointDistanse > maxPoint.PointDistance)

                {

                    maxPoint = new ClassMaxPoint {PointDistance = pointDistanse,

                        MaxPoint = point};

                }

            }

*return*  maxPoint;

        }

        private class ClassMaxPoint

        {

            public double PointDistance { get; set; }

            public Point MaxPoint { get; set; }

        }

    }

}