Путин Павел Александрович, группа 7-1

Лабораторная работа № 7

Вариант № 1

Распознавание образов с использованием машины опорных векторов

Цель работы

Исследовать алгоритмы распознавания образов на основе аппарата машины опорных векторов (Support Vector Machine).

Задание

Получить у преподавателя вариант задания и написать код, реализующий соответствующий алгоритм обработки информации. Для ответа на поставленные в задании вопросы провести численный эксперимент или статистическое имитационное моделирование и представить соответствующие графики. Провести анализ полученных результатов и представить его в виде выводов по проделанной работе.

m1=[1 2], m2=[1 -1], C1=[3 -1; -1 3], C2=[5 2; 2 6].
Воспользовавшись классификатором SVM, вычислите вероятности ошибки при классификации линейно разделимых выборок двух классов (генерацию выборок можно реализовать на основе лаб.2/ лаб.3)

Код программы (внесённые изменения в шаблон кода выделены)

```
% clear all;
close all;
%% 1. Задание исходных данных
n = 2; % размерность признакового пространства
M = 2; % число классов
K = 200; % количество статистических испытаний
% Априорные вероятности, математические ожидания и матрицы ковариации классов
dm = 3.1623; % расстояние между математическими ожиданиями классов по координатным
осям
C = zeros(n, n, M);
C_ = C; % матрица ковариации вектора признаков различных классов
pw = [0.5 0.5]; % для двух классов (M = 2)
pw = pw / sum(pw);
D = 3 * eye(2);
m = [2 2; 1 -1]'; % для двух классов (M=2)
C(:,:,1) = [5 2; 2 5];
C(:,:,2) = [5 2; 2 5];
for k = 1 : M
    C_(:, :, k) = C(:, :, k) ^ -1;
end
np = sum(pw);
pw = pw / np; % исключение некорректного задания априорных вероятностей
%% 2. Обучение svm классификаторов для каждой пары классов
% 2.1. Генерация обучающих выборок классов
% Объемы выборок каждого класса
Ks = fix(K * pw);
Ks(end) = K - sum(Ks(1 : end - 1));
X = []; % общая обучающая выборка (все образы всех классов)
Y = []; % номера классов для каждого образа
% Генерация выборок
for i = 1 : M % цикл по классам
    XN{i} = repmat(m(:, i), [1, Ks(i)]) + randncor(n, Ks(i), C(:, :, i)); %
генерация Ks(i) образов i-го класса
    X = cat(1, X, XN{i}'); % помещаем образы в общую выборку
    Y = cat(1, Y, i * ones(Ks(i), 1)); % номер класса для каждого образа
end
%% 2.2. Сначала обучаем классификаторы, чтобы на эксперименте дёргать уже
обученные.
% Поскольку классов может быть 3, а svm осуществляет только попарное
% сравнение, то обучаем свой классификатор для каждой пары классов
% (для сравнения 1го и 2го, 2го и 3го, 1го и 3го классов)
% Обучаем svm-классификаторы для каждой пары классов
r = 1; % параметр регуляризации (балансирует положение разделяющей границы
относительно положения опорных векторов и "объему" заступов за границу)
svm_strs = cell(M);
for i = 1 : M - 1 % цикл по парам классов (как в 4й лабе с 3мя буквами)
    for j = i + 1 : M
        % Формирование смешанной обучающей выборки для пары классов
        Xij = [XN{i}'; XN{j}']; % помещаем образы 2х классов в смешаную выборку
        D = [true(Ks(i), 1); false(Ks(j), 1)]; % метки классов
        % Вар. а) Классификатор для линейно разделимых данных (разделяющая граница
- прямая)
```

```
        svm_strs{i, j} = fitcsvm(Xij, D, 'Solver', 'L1QP', 'KernelFunction',
'linear', 'BoxConstraint', r);
        % Вар. 6) Классификатор для линейно неразделимых данных (разделяющая
граница - кривая)
        % svm_strs{i, j} = fitcsvm(Xij, D, 'Solver', 'L1QP', 'KernelFunction',
'polynomial', 'PolynomialOrder', 4);
        % Другие варианты Kernel_Function и её параметров (смотри файл fitcsvm.m)
        % svm_strs{i, j} = fitcsvm(Xij, D, 'Solver', 'L1QP', 'KernelFunction',
'rbf', 'KernelScale', 1);
    end
end
%% 2.3. Отображение областей локализации классов
show = true; % визуализация результатов обучения (если n = 3, можно установить
false - 3d пространство не визуализируется)
if show
    % 2.3.1 Формируем дискретную двумерную сетку отсчётов
    d = 0.05; % шаг сетки отсчётов
    [x1Grid, x2Grid] = meshgrid(min(X(:, 1)) : d : max(X(:, 1)), ...
        min(X(:, 2)) : d : max(X(:, 2)));
    xGrid = [x1Grid(:), x2Grid(:)];
    N = size(xGrid, 1);
    idxs = []; % индексы классов
    sv = []; % массив опорных векторов

    % 2.3.2 Классифицируем узлы дискретной сетки
    for i = 1 : M - 1 % цикл по парам классов (как в 4й лабе с 3мя буквами)
        for j = i + 1 : M
            sv = cat(1, sv, svm_strs{i, j}.SupportVectors); % опорные векторы
            cls = predict(svm_strs{i, j}, xGrid); % логические метки классов (true
и false)
            % переводим метки в числовые индексы классов
            idx = i * cls + j * ~cls; % если метки == true - то это i-й класс, если
false - то j-й
            idxs = cat(2, idxs, idx); % фиксируем индексы классов от всех
классификаторов
        end
    end
    iai = mode(idxs, 2); % выбираем превалирующий по строке номер класса для
каждого отсчёта
    % 2.3.3 Отрисовка узлов двумерной сетки разными цветами
    figure; % создаём графическое окно
    if M == 2
        h(1 : M) = gscatter(xGrid(:, 1), xGrid(:, 2), iai, [0.5 0.1 0.5; 0.1 0.5
0.5]);
    elseif M == 3
        h(1 : M) = gscatter(xGrid(:, 1), xGrid(:, 2), iai, [0.5 0.1 0.5; 0.5 0.5
0.1; 0.1 0.5 0.5]);
    end
    hold on
    % 2.3.4 Отрисовка образов обучающих выборок
    h(M + 1 : 2 * M) = gscatter(X(:, 1), X(:, 2), Y);
    axis tight
    % 2.3.5 Отображение опорных векторов и подпись
    plot(sv(:, 1), sv(:, 2), 'ko', 'MarkerSize', 10)
    if M == 2
```

```
        legend('class 1 region', 'class 2 region', 'class 1', 'class 2', 'Support
Vector');
    elseif M == 3
        legend('class 1 region', 'class 2 region', 'class 3 region', ...
               'class 1', 'class 2', 'class 3', 'Support Vector');
    end
    hold off
end % подпись, если включена визуализация
%% 3. Расчет теоретических матриц вероятностей ошибок распознавания
PIJ = zeros(M);
PIJB = zeros(M);
mg = zeros(M);
Dg = zeros(M);
l0_ = zeros(M);
for i = 1 : M
    for j = i + 1 : M
           dmij = m(:, i) - m(:, j);
           l0_(i, j) = log(pw(j) / pw(i));
           dti = det(C(:, :, i));
           dtj = det(C(:, :, j));
           trij = trace(C_(:, :, j) * C(:, :, i) - eye(n));
           trji = trace(eye(n) - C_(:, :, i) * C(:, :, j));
           mg1 = 0.5 * (trij + dmij' * C_(:, :, j) * dmij - log(dti / dtj));
           Dg1 = 0.5 * trij ^ 2 + dmij' * C_(:, :, j) * C(:, :, i) * C_(:, :, j) *
dmij;
           mg2 = 0.5 * (trji - dmij' * C_(:, :, i) * dmij + log(dtj / dti));
           Dg2 = 0.5 * trji ^ 2 + dmij' * C_(:, :, i) * C(:, :, j) * C_(:, :, i) *
dmij;
           sD1 = sqrt(Dg1); sD2 = sqrt(Dg2);
           PIJ(i, j) = normcdf(l0_(i, j), mg1, sD1); PIJ(j, i) = 1 - normcdf(l0_(i,
j), mg2, sD2);
           mu2 = (1 / 8) * dmij' * ((C(:, :, i) / 2 + C(:, :, j) / 2) ^  - 1) *
dmij...
               +0.5 * log((dti + dtj) / (2 * sqrt(dti * dtj))); % расстояние
Бхатачария
           PIJB(i, j) = sqrt(pw(j) / pw(i)) * exp( - mu2);
           PIJB(j, i) = sqrt(pw(i) / pw(j)) * exp( - mu2); % границы Чернова
    end
    PIJB(i, i) = 1 - sum(PIJB(i, :)); % нижняя граница вероятности правильного
распознавания
    PIJ(i, i) = 1 - sum(PIJ(i, :)); % теоретические вероятности правильного
распознавания
end
%% 4. Тестирование алгоритма методом статистических испытаний
Pcv = zeros(M); % + инициализация экспериментальной матрицы ошибок
x = ones(n, 1);
u = zeros(M, 1);
Pc_ = zeros(M); % экспериментальная матрица вероятностей ошибок из 3й лабы
for k = 1 : K % цикл по числу испытаний
    for i = 1 : M % цикл по классам
        [x, px] = randncor(n, 1, C(:, :, i));
        x = x + m(:, i); % генерация образа i - го класса
        for j = 1 : M % вычисление значения разделяющих функций из 3й лабы
            u(j) = -0.5 * (x - m(:, j))' * C_(:, :, j) * (x - m(:, j)) - 0.5 *
log(det(C(:, :, j))) + log(pw(j));
```

```
        end
        [ui, iai] = max(u); % определение максимума
        Pc_(i, iai) = Pc_(i, iai) + 1; % фиксация результата распознавания
        % Прогон по всем классификаторам SVM в том же порядке как мы их обучали
        iais = []; % массив для фиксации результатов от разных классификаторов
        for ii = 1 : M - 1 % цикл по парам классов
            for jj = ii + 1 : M
                % Вызываем классификатор для сравнения ii-го и jj-го
                % классов из массива svm_strs и распознаём им образ x
                cl = predict(svm_strs{ii, jj}, x');
                if cl % cl будет true, если класс ii-й
                    iai = ii;
                else % cl будет false, если класс jj-й (потому что так формировался
массив меток при обучении)
                    iai = jj;
                end
                iais = [iais, iai]; % фиксируем результат распознавания
            end
        end
        iai = mode(iais); % выбираем класс, за который проголосовало большинство
классификаторов
        Pcv(i, iai) = Pcv(i, iai) + 1; % фиксация результата распознавания
    end % цикл по классам
end
Pc_ = Pc_ / K;
Pcv = Pcv / K; % + нормировка экспериментальной матрицы по svm на число испытаний
disp('Теоретическая матрица вероятностей ошибок');
disp(PIJ);
disp('Экспериментальная матрица вероятностей ошибок');
disp(Pc_);
disp('Экспериментальная матрица вероятностей ошибок SVM');
disp(Pcv);
```

Результаты выполнения задания

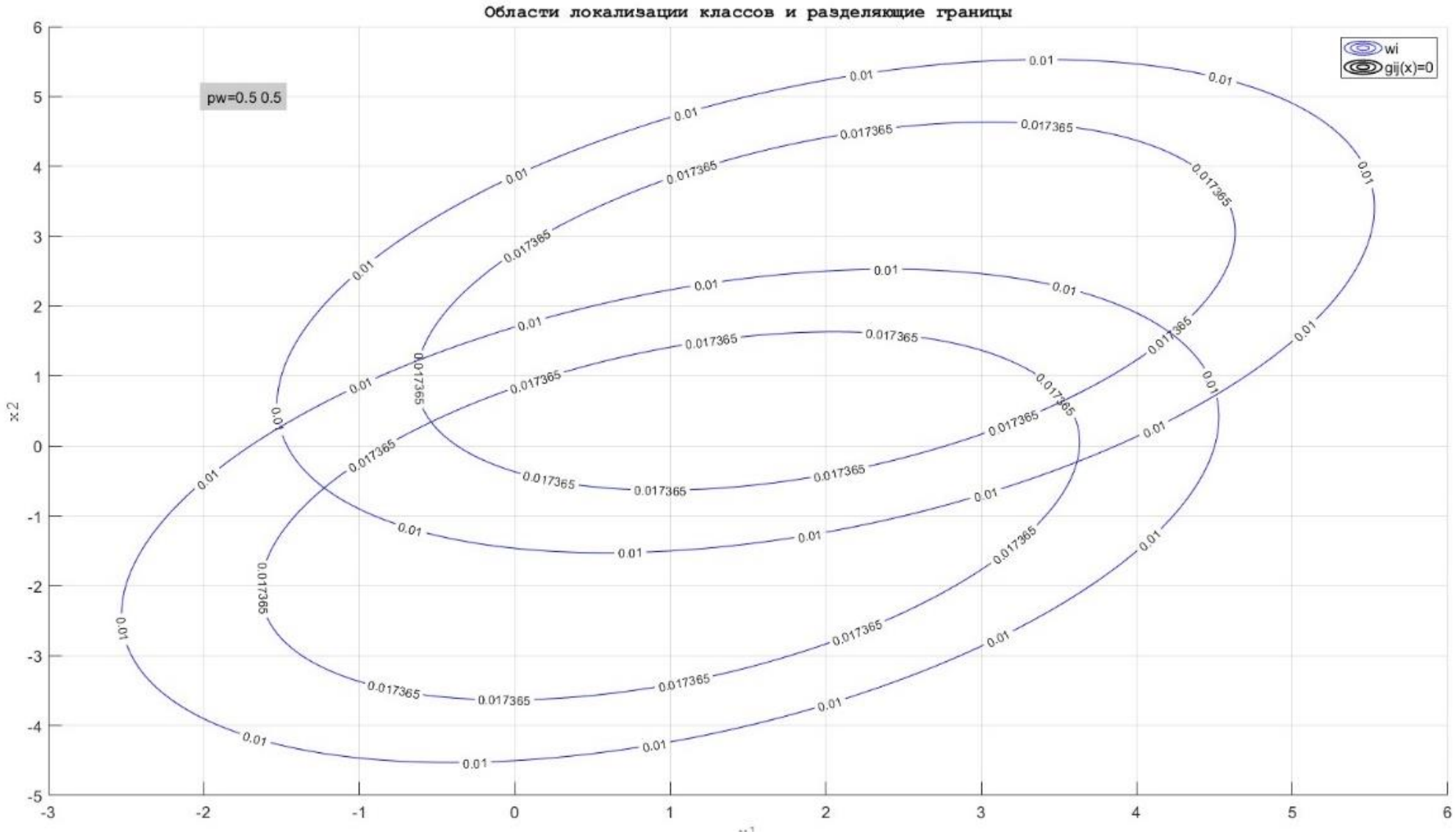

Рисунок 1 - Области локализации классов

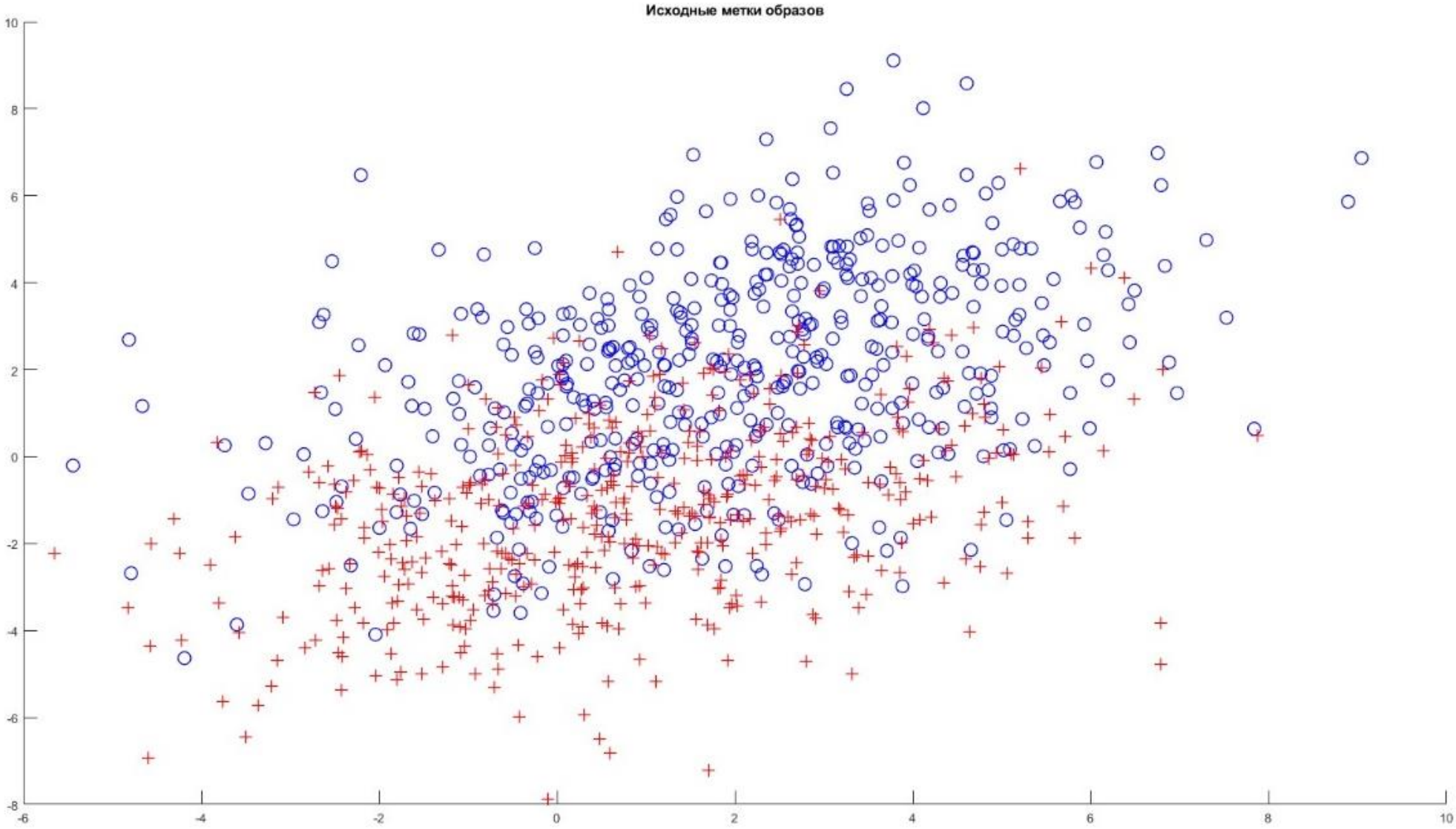

Рисунок 2 - Исходные метки образов

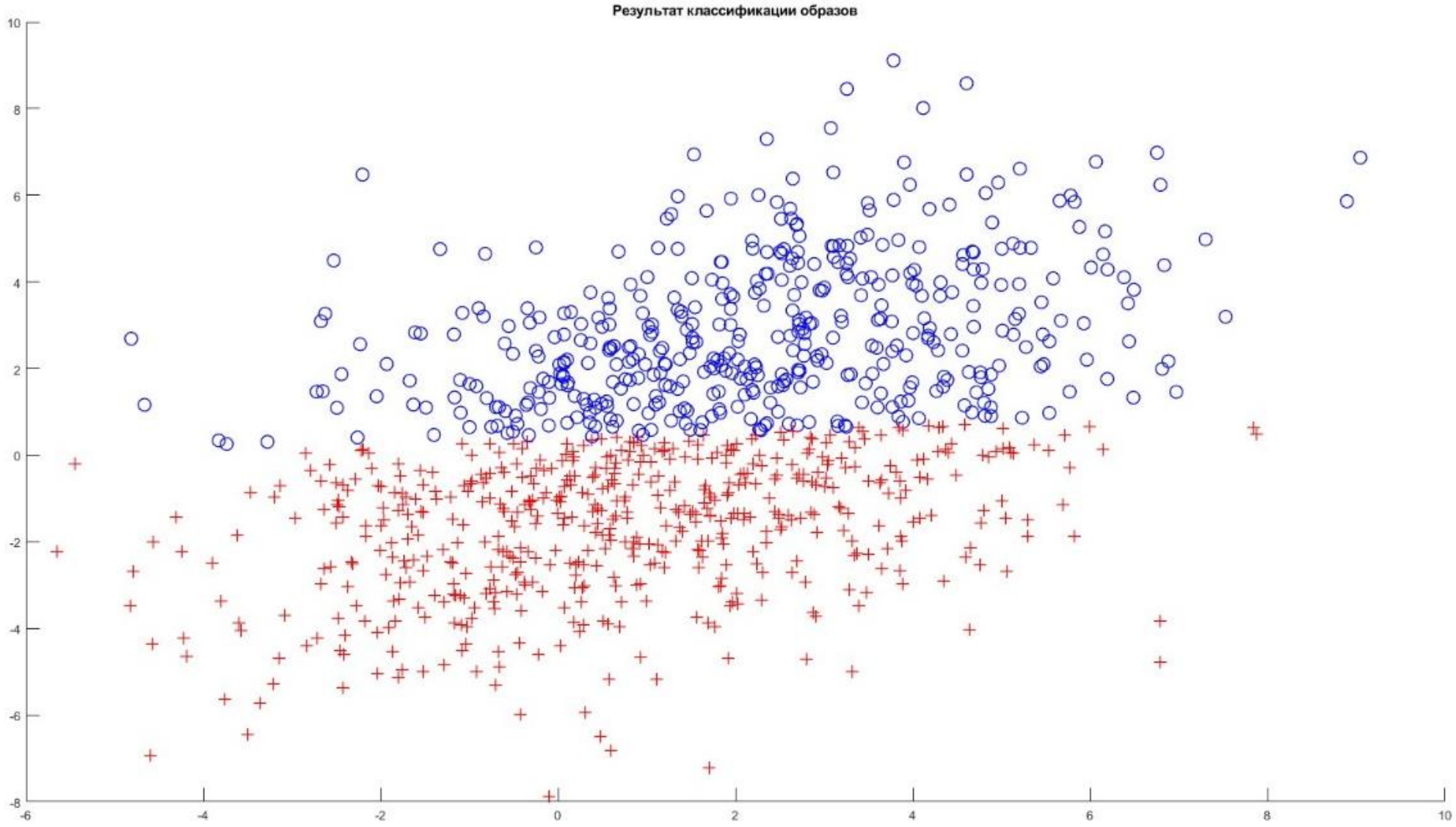

Рисунок 3 – Результат классификации (байесовский метод)

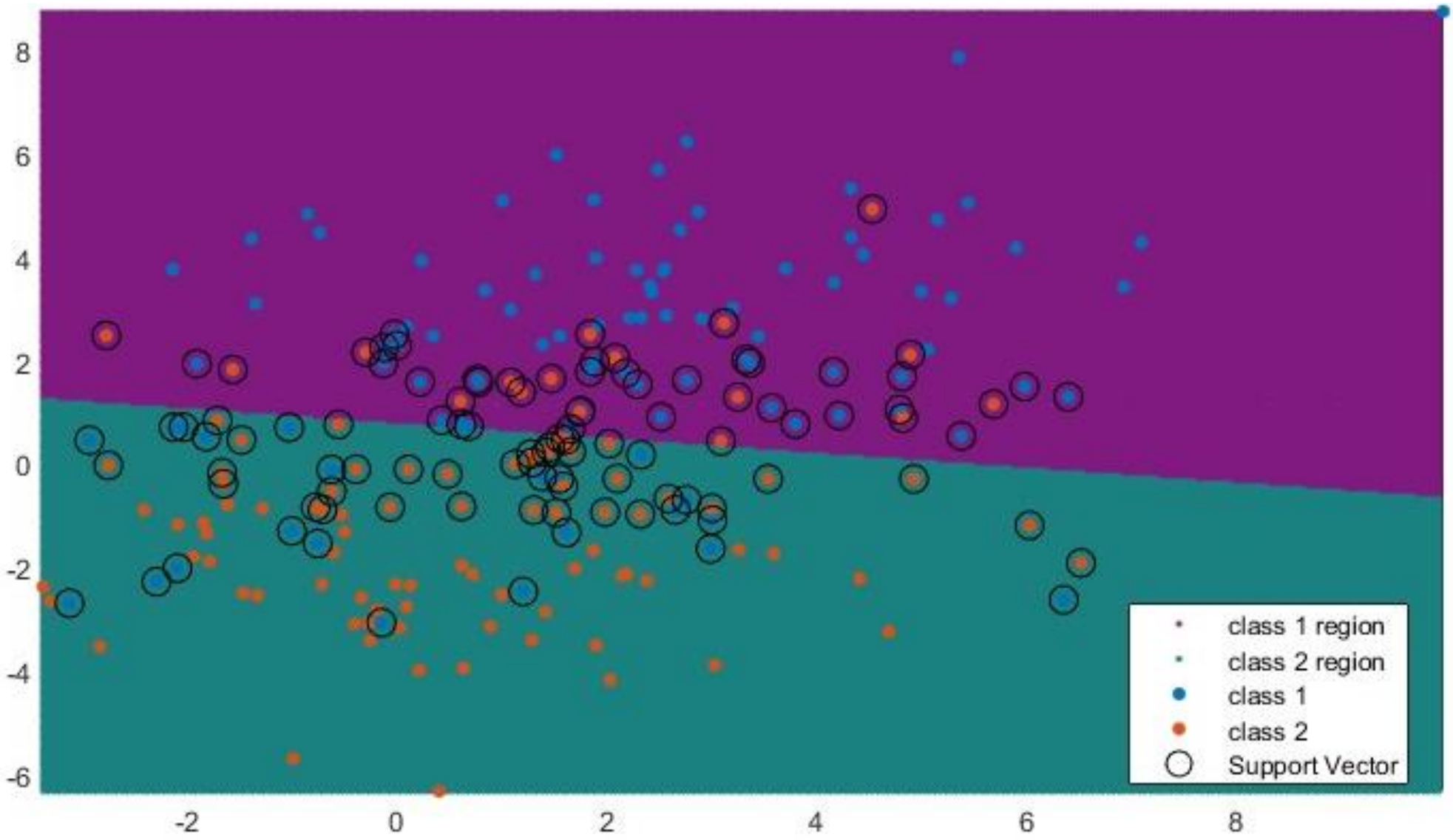

Рисунок 4 - Результат классифкаци (метод опорных векторов)

Таблица 1 - Матрицы ошибок

Способ расчёта	**Матрица ошибок**	
Теоретическая	0,7494	0,2506
	0,2506	0,7494
Экспериментальная	0,7100	0,2900
	0,2500	0,7500
Экспериментальная (SVM)	0,7250	0,2750
	0,2500	0,7500

Таблица 2 - Матрицы ошибок при разных ядрах

Функция ядра	**Матрица ошибок**	
Радиальная базисная функция	0,6550	0,3450
	0,2050	0,7950
Линейная	0,7250	0,2750
	0,2500	0,7500
Полиномиальная	0,5650	0,4350
	0,1700	0,8300

Выводы

1. Параметр регуляризации C позволяет находить компромисс между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки. Его введение позволяет повысить устойчивость решения w.
2. Наилучшее качество оценивания, согласно таблице 2, на рассматриваемых данных обеспечивает линейная фукнция ядра