Путин Павел Александрович, группа 7-1

Лабораторная работа № 2

Вариант № 10-d

Распознавание образов, описываемых гауссовскими случайными векторами с одинаковыми матрицами ковариаций

Цель работы

Синтезировать алгоритмы распознавания образов, описываемых гауссовскими случайными векторами с одинаковыми матрицами ковариаций. Исследовать синтезированные алгоритмы распознавания с точки зрения ожидаемых потерь и ошибок.

Задание

Получить у преподавателя вариант задания и написать код, реализующий алгоритм распознавания образов, описываемых гауссовскими случайными векторами с заданными параметрами. Получить матрицы ошибок на основе аналитических выражений и вычислительного эксперимента. Провести анализ полученных результатов и представить его в виде выводов по проделанной работе.

Изменить исходные данные таким образом, чтобы в теоретической матрице ошибок увеличилась ошибка второго рода, а ошибка первого рода уменьшилась.

m1=[2 2], m2=[1 -1], C=[5 1; 1 5]

Код программы (внесённые изменения в шаблон кода выделены)

```
%% Вариант 10d. Синтез и анализ алгоритмов распознавания ГСВ
% с одинаковой матрицей ковариации  (двумерный вектор признаков)
clear all
close all
%% 1. Задание исходных данных

n = 2;              % размерность признакового пространства
M = 2;              % число классов
K = 1000;           % количество статистических испытаний
m = [2 2; 1 -1]';   % мат. ожидания - координаты центров классов (2,-3) и (1,6)
pw = [0.8, 0.2];    % априорные вероятности классов (доля образов каждого класса в
общей выборке)
C = [5 1; 1 5];     % матрица ковариаций классов
Cinv = C ^ -1;      % обратная ков. матрица
IMS = [];           % общая совокупность образов (общая выборка)

% 1.1. Визуализация исходной совокупности образов
% Определение числа образов в каждом классе, пропорционально pw
Ks = fix(pw .* K);
Ks(end) = K - sum(Ks(1 : end - 1));
label = {'bo', 'r+', 'k*', 'gx'}; % маркеры классов для визуализации

figure;
hold on;
title('Исходные метки образов');
for i = 1 : M % цикл по классам
    % генерация Ks(i) образов i-го класса
    ims = repmat(m(:, i), [1, Ks(i)]) + randncor(n, Ks(i), C);
    if n == 2
        plot(ims(1, :), ims(2, :), label{i}, 'MarkerSize', 10, 'LineWidth', 1);
    elseif n == 3
        plot3(ims(1, :), ims(2, :), ims(3, :), label{i}, 'MarkerSize', 10,
'LineWidth', 1);
    end
    IMS = [IMS, ims];   % добавление в общую совокупность образов
end
%% 2. Расчет разделяющих функций и матрицы вероятностей ошибок распознавания
G = zeros(M, n + 1);    % разделяющие функции ???
PIJ = zeros(M);         % теоретическая матрица ошибок
l0_ = zeros(M);         % порог принятия ошибки
for i = 1 : M % цикл по классам
    G(i, 1 : n) =(Cinv * m(:, i))';
    G(i, n + 1) = -0.5 * m(:, i)' * Cinv * m(:, i);
    for j = i + 1 : M
        l0_(i, j) = log(pw(j) / pw(i));
        h = 0.5 * (m(:, i) - m(:, j))' * Cinv * (m(:, i) - m(:, j));
        sD = sqrt(2 * h);
        PIJ(i, j) = normcdf(l0_(i, j), h, sD);
        PIJ(j, i) = 1 - normcdf(l0_(i, j), -h , sD);
    end
    % нижняя граница вероятности правильного распознавания (на главной диагонали)
    PIJ(i, i) = 1 - sum(PIJ(i, :));
end
```

```
% 2.1. Визуальзация результатов распознавания образов
figure;
hold on;
title('Результат классификации образов');
for i = 1 : K                % цикл по всем образам совокупности
    z = [IMS(:, i); 1];      % значение очердного образа из общей совокупности
    u = G * z + log(pw');    % вычисление значения разделяющих функций
    [ui, iai] = max(u);      % определение максимума (iai - индекс класса)
    if n == 2
        plot(IMS(1, i), IMS(2, i), label{iai}, 'MarkerSize', 10, 'LineWidth', 1);
    elseif n == 3
        plot3(IMS(1, i), IMS(2, i), IMS(3, i), label{iai}, 'MarkerSize', 10,
'LineWidth', 1);
    end
end

%% 3. Тестирование алгоритма методом статистических испытаний
x = ones(n + 1, 1);      % образы классов ???
Pc_ = zeros(M);          % экспериментальная матрица вероятностей ошибок

for k = 1 : K        % цикл по числу испытаний
    for i = 1 : M    % цикл по классам
        [x_, px] = randncor(n, 1, C);
        x(1 : n, 1) =m(:, i) + x_;      % генерация образа i-го класса
        u = G * x + log(pw');           % вычисление значения разделяющих функций
        [ui, iai] = max(u);             % определение максимума
        Pc_(i, iai) = Pc_(i, iai) + 1;  % фиксация результата распознавания
    end
end

% матрица ошибок, полученная экспериментально
% у нее такая же структура, как и в PIJ,
% только вычисляется численно, а не по формулам
Pc_ = Pc_ / K;

disp('Теоретическая матрица вероятностей ошибок');
disp(PIJ);
disp('Экспериментальная матрица вероятностей ошибок');
disp(Pc_);
%% 4. Визуализация областей принятия решений для двумерного случая
if n == 2
    D = 1;
    xmin1 = -4 * sqrt(D) + min(m(1,:)); % левая граница графика по оси абсцисс
    xmax1 = 4 * sqrt(D) + max(m(1,:));  % правая граница графика по оси абсцисс
    xmin2 = -4 * sqrt(D) + min(m(2,:)); % нижняя граница графика по оси ординат
    xmax2 = 4 * sqrt(D) + max(m(2,:));  % верхняя граница графика по оси ординат
    x1 = xmin1 : 0.05 : xmax1;          % отсчёты по оси абсцисс
    x2 = xmin2 : 0.05 : xmax2;          % отсчёты по оси ординат
    [X1, X2] = meshgrid(x1, x2); % матрицы значений координат случайного вектора
    x12 = [X1(:), X2(:)];

    figure;
    hold on;
    grid on;
```

```
    axis([xmin1, xmax1, xmin2, xmax2]); % установка границ поля графика по осям

    for i = 1 : M % цикл по классам
        f2 = mvnpdf(x12, m(:, i)', C); % массив значений плотности распределения
        f3 = reshape(f2, length(x2), length(x1)); % матрица значений плотности
распределения
        [Ch, h] = contour(x1, x2, f3, [0.01, 0.5 * max(f3(:))], 'Color', 'b',
'LineWidth', 0.75);
        clabel(Ch, h);
        for j = i + 1 : M % изображение разделяющих границ
            wij = Cinv * (m(:, i) - m(:, j));
            wij0 = -0.5 * (m(:, i) + m(:, j))' * Cinv * (m(:, i) - m(:, j));
            f4 = wij' * x12' + wij0;
            f5 = reshape(f4, length(x2), length(x1));
            [Ch_, h_] = contour(x1, x2, f5, -l0_(i, j) + 0.0001, 'Color', 'k',
'LineWidth', 1.25);
        end
    end
    set(gca, 'FontSize', 13);
    title('Области локализации классов и разделяющие границы', 'FontName',
'Courier');
    xlabel('x1', 'FontName', 'Courier');
    ylabel('x2', 'FontName', 'Courier');
    strv1 = ' pw=';
    strv2 = num2str(pw, '% G');
    text(xmin1 + 1, xmax2 - 1, [strv1, strv2], 'HorizontalAlignment', 'left',
'BackgroundColor', ...
        [.8 .8 .8], 'FontSize', 12);
    legend('wi', 'gij(x)=0');
    hold off;
end
```

Результаты выполнения задания

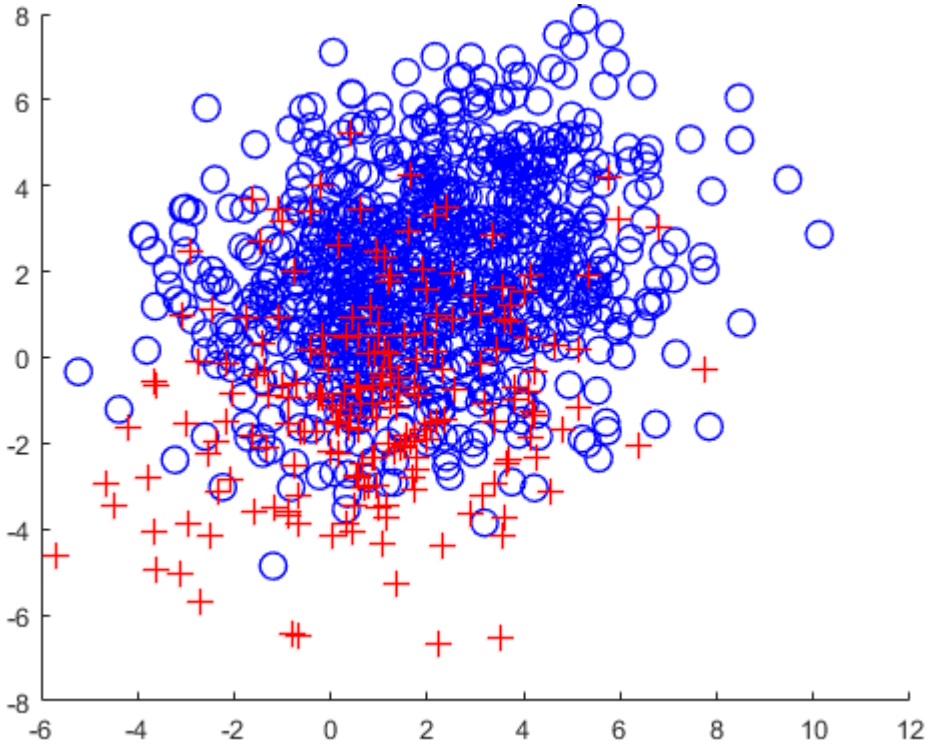

Рисунок 1 - Исходные метки образов

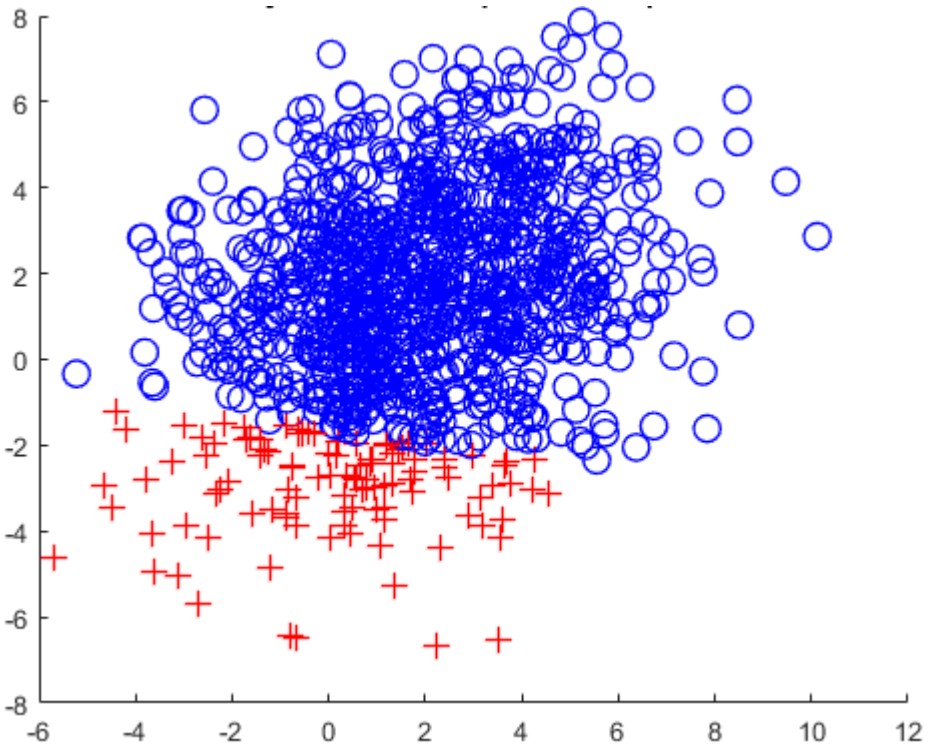

Рисунок 2 - Результат классификации образов

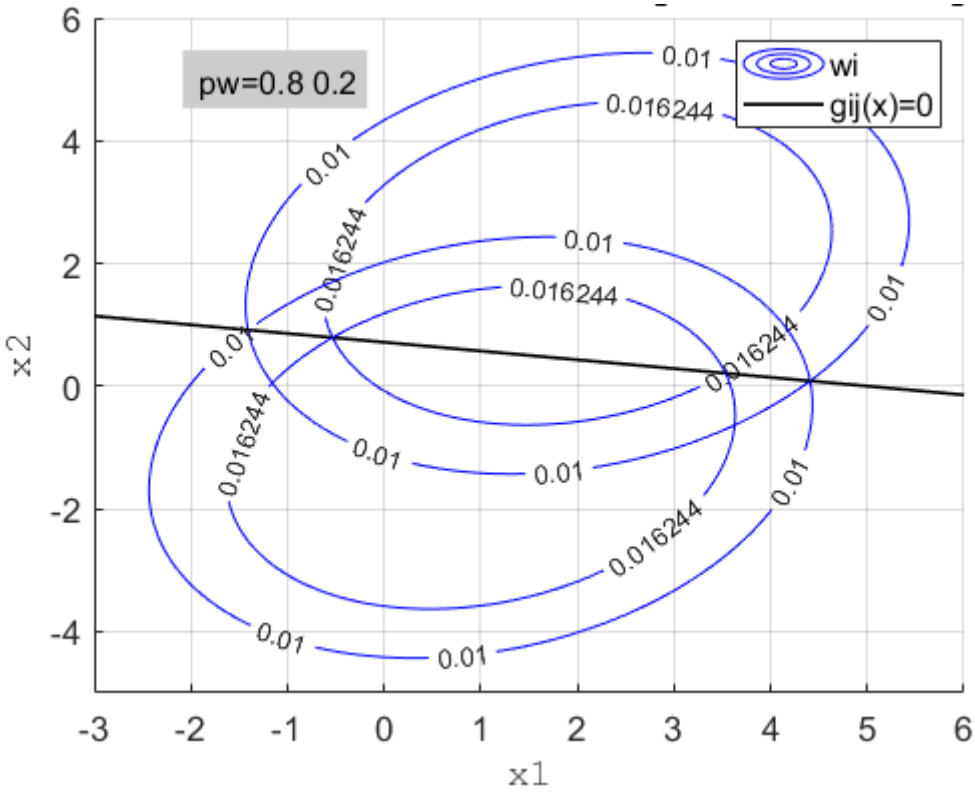

Рисунок 3 - Области локализации классов и разделяющие границы

При равных априорных вероятностях классов ошибки первого и второго рода одинаковы. Если увеличить априорную вероятность первого класса, то ошибка первого рода уменьшится, а ошибка второго рода увеличится. Значения ошибок показаны в таблице 1:

Таблица 1 – Значения ошибок первого и второго рода

Значения pw	Ошибка первого рода	Ошибка второго рода
[0.5, 0.5]	0.2492	0.2492
[0.8, 0.2]	0.0445	0.6356

Выводы

1. Для увеличения ошибки второго рода и уменьшения ошибки первого рода необходимо увеличить априорную вероятность первого класса.
2. Элементы главной диагонали матрицы ошибок показывают вероятность принятия правильного решения при классификации объекта в данный класс.
3. Элементы побочной диагонали характеризуют вероятность ошибки отнесения объекта к неправильному классу: объекта первого класса ко второму классу (ошибка первого рода), а объекта второго класса к первому (ошибка первого рода).
4. Формы кластеров объектов в пространстве используемых признаков определяются матрицей ковариации.