```matlab
% Author: Noah Rondeau
% ECE 485: Data Analysis and Pattern Recognition
% Assignment 2
% Question 2
clc;
clear all;
close all;

fprintf('ECE 485: Data Analysis and Pattern Recognition\n');
fprintf('Author: Noah Rondeau\n');
fprintf('Assignment 2, Question 3\n\n');

%load data and extract into classes
load Data/Data3;
class1 = Data3( Data3(:,3)==1, 1:2);
class2 = Data3( Data3(:,3)==2, 1:2);
class3 = Data3( Data3(:,3)==3, 1:2);

length_c1 = length(class1);
length_c2 = length(class2);
length_c3 = length(class3);
length_d3 = length(Data3);

prior_c1 = length_c1 / length_d3;
prior_c2 = length_c2 / length_d3;
prior_c3 = length_c3 / length_d3;


mu1 = mean(class1)';
Sigma1 = cov(class1);
mu2 = mean(class2)';
Sigma2 = cov(class2);
mu3 = mean(class3)';
Sigma3 = cov(class3);

fprintf('PART A: estimated class means and covariances\n');
fprintf('Class 1:\n');
disp('Mean:');
disp(mu1);
disp('Covariance:');
disp(Sigma1);
fprintf('Class 2:\n');
disp('Mean:');
disp(mu2);
disp('Covariance:');
disp(Sigma2);
fprintf('Class 3:\n');
disp('Mean:');
disp(mu3);
disp('Covariance:');
disp(Sigma3);
```

```matlab
if prior_c1 == prior_c2 && prior_c2 == prior_c3
    fprintf('Note: all three classes have the same apriori
 probabilities\n');
    fprintf('All three classes have the same number of records in the
 data\n');
end

% PART B
fprintf('\nPART B: Test point classification by Mahalanobis distance
\n');

x1 = [10,2];
x2 = [-3,4];
x3 = [2,2];
x4 = [5, -7];
X = [x1;x2;x3;x4];

% determine mahalanobis distance for each x relative to each class
dist1 = mahal(X, class1);
dist2 = mahal(X, class2);
dist3 = mahal(X, class3);

% determine class for each x
for i=1:4
    fprintf('x_%d Mahalanobis Distances:\n', i);
    fprintf('From:\n\tClass1: %f\n\tClass2: %f\n\tClass3: %f\n',
 dist1(i), dist2(i), dist3(i));
    min_dist = min([dist1(i), dist2(i), dist3(i)]);
    if min_dist == dist1(i)
        x_class(i) = 1;
    elseif min_dist == dist2(i)
        x_class(i) = 2;
    elseif min_dist == dist3(i)
        x_class(i) = 3;
    end
    fprintf('Point belongs to: Class%d\n\n', x_class(i));
end

% Tag each x with its class value and label
X = [X x_class'];
for i=1:4
    data_labels(i,1) = cellstr(sprintf('x%d, (C%d)', i, X(i,3)));
end

% PART D
fprintf('PART D: Supervised vs Unsupersised Learning\n');
fprintf('If we did not know which points belonged to which class, the
 approach above does not work.\n');
fprintf('It becomes a clustering problem, not a classification
 problem.\n');
fprintf('We have not yet gone over these issues in class.\n');
fprintf('But, I could imagine a sort of finite-element analysis which
 would\n');
```

```matlab
fprintf('assign a density to a region based on the number of data
 points it contains.\n');
fprintf('Assuming either:\n\ta) that I know the number of classes the
 data represents, or\n');
fprintf('\tb) that I assume they are distributed according to a
 particular distribution:\n');
fprintf('I would then attempt to fit distributions centered at the
 density peaks (assuming they exist).\n');
fprintf('This would not generally work if two classes overlapped
 significantly\n\n');

%PART E: Get decision boundaries
coeff1 = log(prior_c1 / (2*pi*sqrt(det(Sigma1))));
coeff2 = log(prior_c2 / (2*pi*sqrt(det(Sigma2))));
coeff3 = log(prior_c3 / (2*pi*sqrt(det(Sigma3))));
syms p1(xs1,xs2) p2(xs1, xs2) p3(x21, xs2);
syms b12(xs1) b13(xs1) b23(xs1);
assume(xs1, 'real');
assume(xs2, 'real');
p1(xs1,xs2) = coeff1 - 0.5 * ([xs1 xs2]' - mu1)' * inv(Sigma1) *([xs1
 xs2]' - mu1);
p2(xs1,xs2) = coeff2 - 0.5 * ([xs1 xs2]' - mu2)' * inv(Sigma2) *([xs1
 xs2]' - mu2);
p3(xs1,xs2) = coeff3 - 0.5 * ([xs1 xs2]' - mu3)' * inv(Sigma3) *([xs1
 xs2]' - mu3);
b12 = simplify(p1==p3);
b13 = simplify(p1==p3);
b23 = simplify(p2==p3);

fprintf('PART E: 2-class decision boundaries\n\n');
fprintf('Boundary between classes 1 and 2:\n');
disp(vpa(b12/(1e48), 4));

fprintf('Boundary between classes 1 and 3:\n');
disp(vpa(b13/(1e48), 4));

fprintf('Boundary between classes 2 and 3:\n');
disp(vpa(b23/(1e20), 4));


figure(1);
hold on;
scatter(class1(:,1), class1(:,2), '+b');
scatter(class2(:,1), class2(:,2), 'om');
scatter(class3(:,1), class3(:,2), 'xy');
scatter(X(:,1), X(:,2), '^k', 'HandleVisibility', 'off');
text(X(:,1) + 0.5, X(:,2) + 0.2, data_labels);
Q2_PlotEllipse(mu1, Sigma1, 1, 'on');
Q2_PlotEllipse(mu1, Sigma1, 2, 'off');
Q2_PlotEllipse(mu1, Sigma1, 3, 'off');
Q2_PlotEllipse(mu2, Sigma2, 1, 'off');
Q2_PlotEllipse(mu2, Sigma2, 2, 'off');
Q2_PlotEllipse(mu2, Sigma2, 3, 'off');
Q2_PlotEllipse(mu3, Sigma3, 1, 'off');
```

```matlab
    Q2_PlotEllipse(mu3, Sigma3, 2, 'off');
    Q2_PlotEllipse(mu3, Sigma3, 3, 'off');
    Q2_PlotEigen(mu1, Sigma1, 1, 'on', 'off');
    Q2_PlotEigen(mu2, Sigma2, 1, 'off', 'off');
    Q2_PlotEigen(mu3, Sigma3, 1, 'off', 'off');

    axis equal;
    grid on;

    % plot where the boundaries are. Do this by creating a mesh grid and
    % calculating all three mvnpdf() against it. Then subtract the
     resulting
    % matrices to get the subtraction of the two functions of interest.
     Then
    % plot the contour where that new matrix is 0.

    %use the existing plot bounds
    x1_lims = xlim;
    x2_lims = ylim;
    x1_range = x1_lims:0.1:x1_lims(2);
    x2_range = x2_lims(1):0.1:x2_lims(2);
    [X1, X2] = meshgrid(x1_range, x2_range);

    pdf1 = mvnpdf([X1(:) X2(:)], mu1', Sigma1);
    pdf2 = mvnpdf([X1(:) X2(:)], mu2', Sigma2);
    pdf3 = mvnpdf([X1(:) X2(:)], mu3', Sigma3);

    bound12 = pdf1 - pdf2;
    bound12 = reshape(bound12, length(x2_range), length(x1_range));
    bound13 = pdf1 - pdf3;
    bound13 = reshape(bound13, length(x2_range), length(x1_range));
    bound23 = pdf2 - pdf3;
    bound23 = reshape(bound23, length(x2_range), length(x1_range));

    contour(X1, X2, bound12, [0 0], '-k', 'LineWidth', 1);
    contour(X1, X2, bound13, [0 0], ':k', 'LineWidth', 1);
    contour(X1, X2, bound23, [0 0], '--k', 'LineWidth', 1);

    title({'Data with 1,2,3-\sigma contours,','eigenvectors, and decision
     boundaries'});
    xlabel('x_1');
    ylabel('x_2');

    legend('Class1', 'Class2', 'Class3', ...
           '1,2,3-\sigma ellipses',...
           'Eigenvectors', ...
           'Decision bound 1-2', ...
           'Decision bound 1-3', ...
           'Decision bound 2-3', ...
           'Location', 'bestoutside' );

    hold off;

    ECE 485: Data Analysis and Pattern Recognition
```

Author: Noah Rondeau
Assignment 2, Question 3

PART A: estimated class means and covariances
Class 1:
Mean:
     0.0355
     7.4948

Covariance:
     2.2196   -1.9762
    -1.9762   2.2166

Class 2:
Mean:
    -0.9853
    -1.0147

Covariance:
     1.5050   3.0342
     3.0342   7.6511

Class 3:
Mean:
     2.0016
    -2.9851

Covariance:
     1.0089   0.1126
     0.1126   0.7451

Note: all three classes have the same apriori probabilities
All three classes have the same number of records in the data

PART B: Test point classification by Mahalanobis distance
x_1 Mahalanobis Distances:
From:
 Class1: 69.688371
 Class2: 318.835858
 Class3: 86.275973
Point belongs to: Class1

x_2 Mahalanobis Distances:
From:
 Class1: 88.183168
 Class2: 56.407509
 Class3: 102.463145
Point belongs to: Class2

x_3 Mahalanobis Distances:
From:
 Class1: 32.434386
 Class2: 11.804292
 Class3: 33.924289

Point belongs to: Class2

x_4 Mahalanobis Distances:
From:
 Class1: 233.164291
 Class2: 236.259443
 Class3: 34.734344
Point belongs to: Class3

PART D: Supervised vs Unsupersised Learning
If we did not know which points belonged to which class, the approach
 above does not work.
It becomes a clustering problem, not a classification problem.
We have not yet gone over these issues in class.
But, I could imagine a sort of finite-element analysis which would
assign a density to a region based on the number of data points it
 contains.
Assuming either:
 a) that I know the number of classes the data represents, or
 b) that I assume they are distributed according to a particular
 distribution:
I would then attempt to fit distributions centered at the density
 peaks (assuming they exist).
This would not generally work if two classes overlapped significantly

PART E: 2-class decision boundaries

Boundary between classes 1 and 2:
0.4299*xs1^2 + 1.535*xs1*xs2 + 0.3006*xs2^2 + 38.81 == 8.918*xs1 +
 15.23*xs2
symbolic function inputs: xs1, xs2
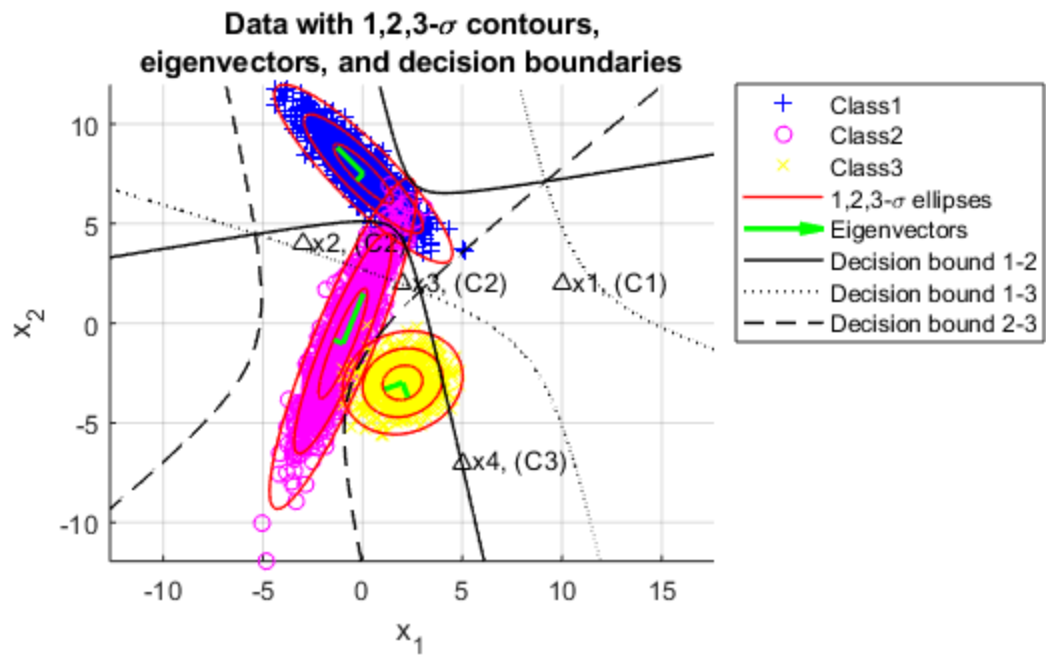
Boundary between classes 1 and 3:
0.4299*xs1^2 + 1.535*xs1*xs2 + 0.3006*xs2^2 + 38.81 == 8.918*xs1 +
 15.23*xs2
symbolic function inputs: xs1, xs2

Boundary between classes 2 and 3:
4.504e-5*xs1*(9.355e31*xs1 + 3.573e32) == 1.832e28*xs2 +
 4.246e27*xs1*xs2 + 1.303e27*xs2^2 + 2.854e28
symbolic function inputs: xs1, xs2

Data with 1,2,3-σ contours, eigenvectors, and decision boundaries

Published with MATLAB® R2018b