Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

EXERCISE 6a: Object recognition with mVision

Concepts: Bayesian Classifier

1. **Running the GUI:** Launch the **recognition** GUI of mVision, and follow the instructions on the bottom part of it to insert objects (points) in a similar way to how they are shown in the following figure (try to insert them in similar positions). The red points represent two features of objects of one class, the red ones features for other class, and the magenta point is the object to classify:

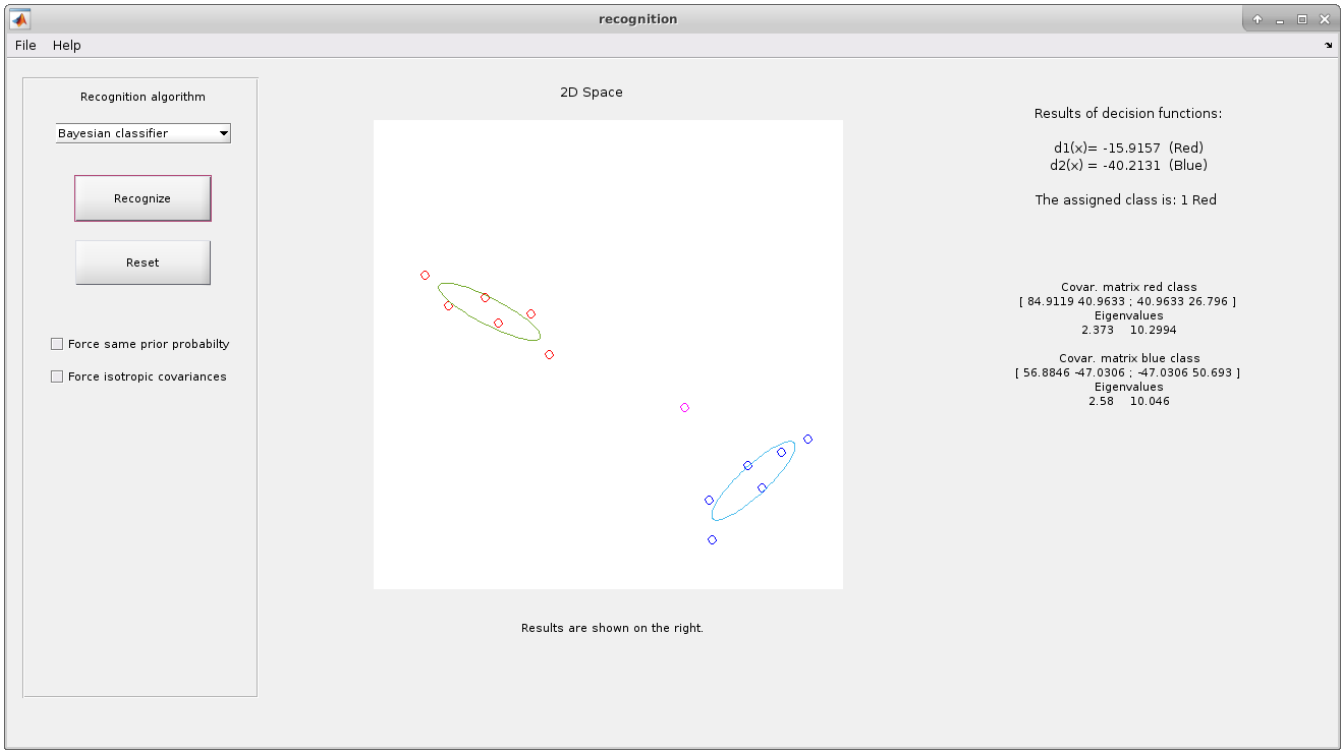


Image 1

2. **Recognition:** Push the **Recognize** button and interpret the results obtained on the right part of the GUI (decision functions, covariance matrices, etc.).

Image 2

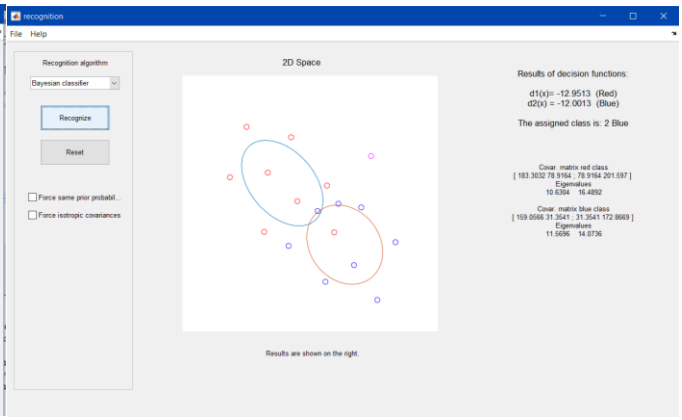


Image 3

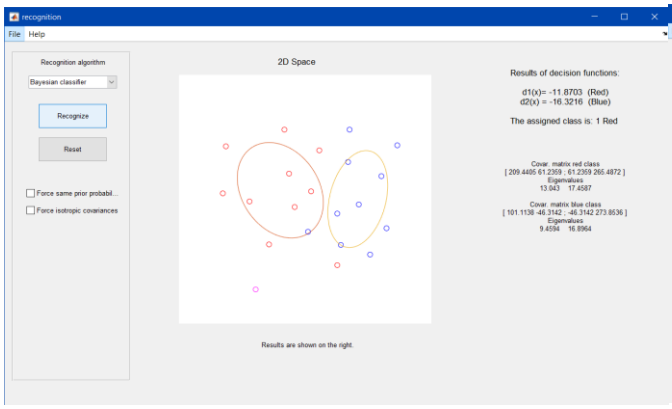


Image 4

Image 5

As we see in all the images, the result part shows the result of each decision function, which will be used to decided which class will be assigned the element to classify. Also, it shows the covariance and the eigenvalues of each gaussian distribution, which between them are different. However, in Image 5 (and XOR patron) there is no result because we cannot use this classifier to classify that patron.

3. **Same probability and covariance:** Now force the classes to have the same probability. With that the GUI also forces the classes to have the same covariance matrices. This emulates situations where both classes have the same dispersion. Discuss the new results.

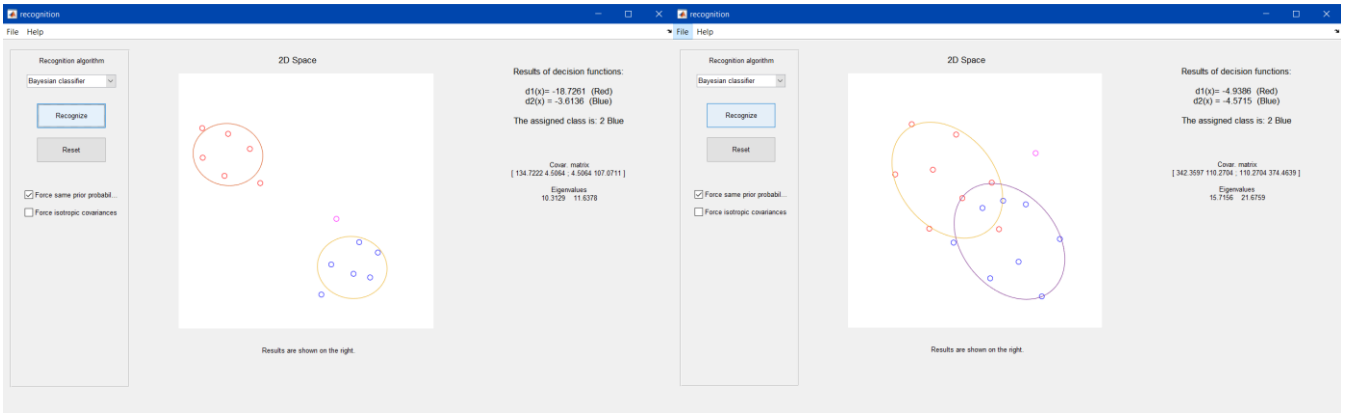


Image 6 *Image 7*

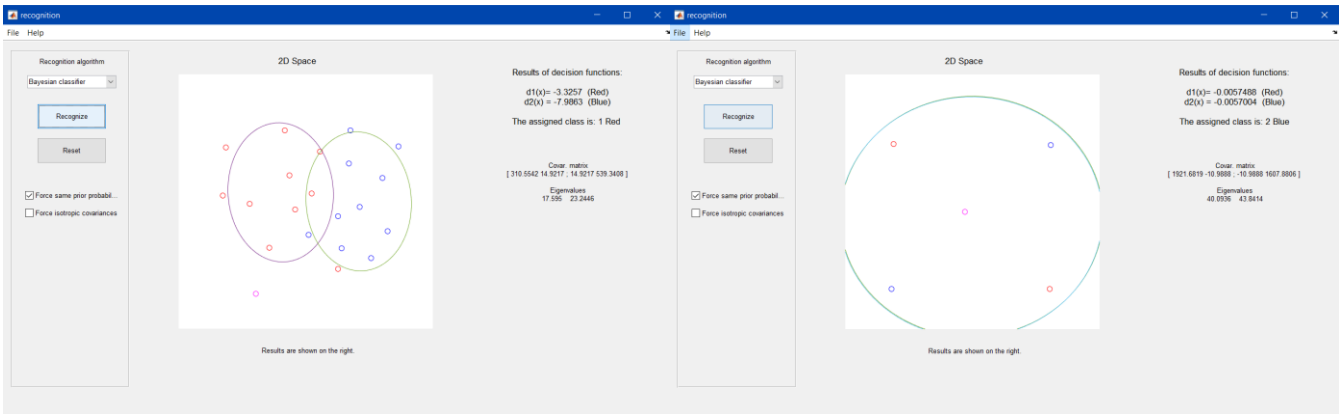


Image 8 *Image 9*

The only difference we can appreciate is that there is only one covariance and eigenvalues for each gaussian distribution. In this case, in Image 9 (XOR patron), we can see there is now result for this patron, however when we execute the same, not always have the same result.

4. **Isotropic covariance:** Force them to have isotropic covariance. Discuss the obtained results.

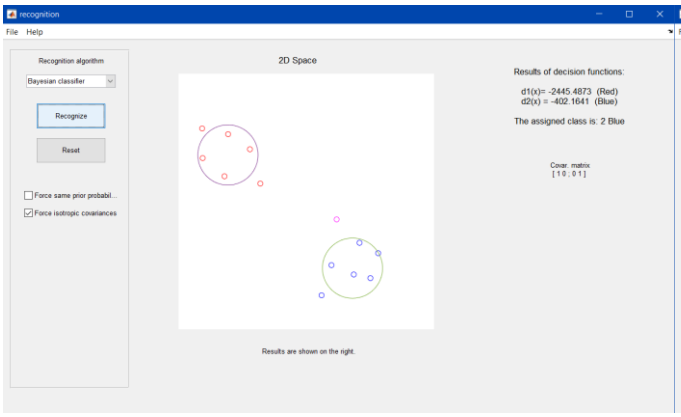


Image 10

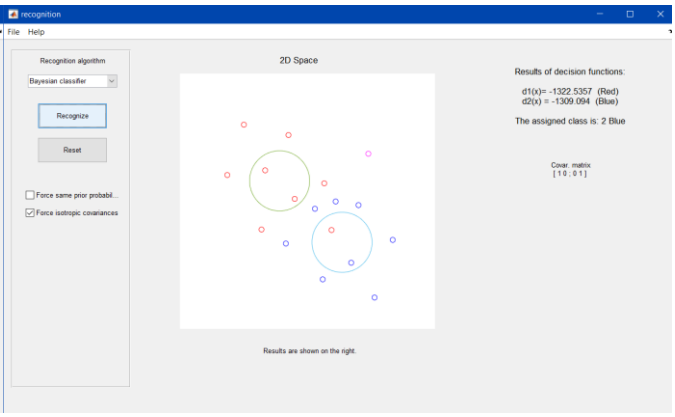


Image 11

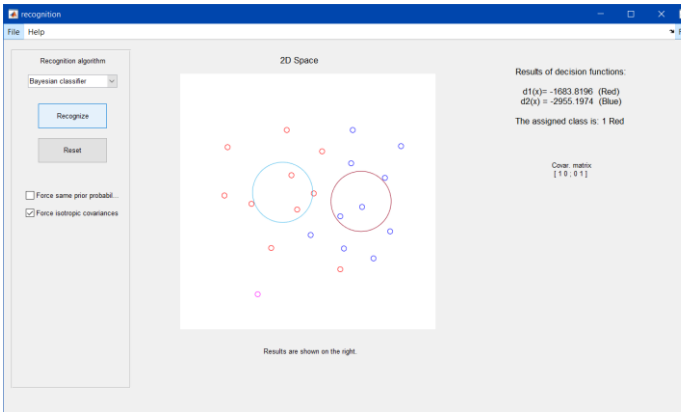


Image 12

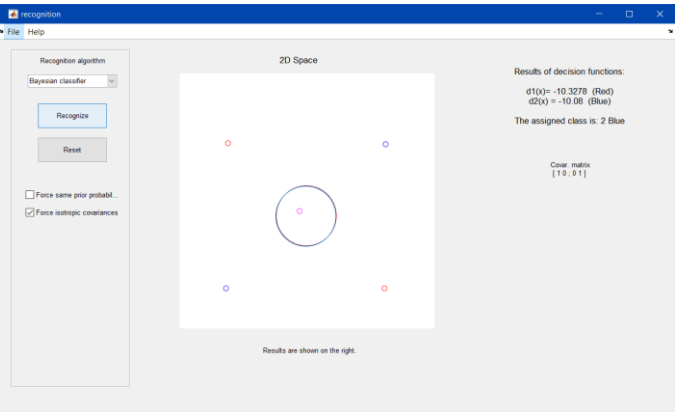


Image 13

In this case we have the same result as the last one. However, the covariance is spherical and the determinant have the value of 1. So it have a gaussian of N(x,[1 0; 0 1]), which the only important result is where the center of the gaussian.

5. **Distances:** Finally, comment which distances have been used in each section (2 to 4) to compute the decision functions: Mahalanobis or Euclidean?

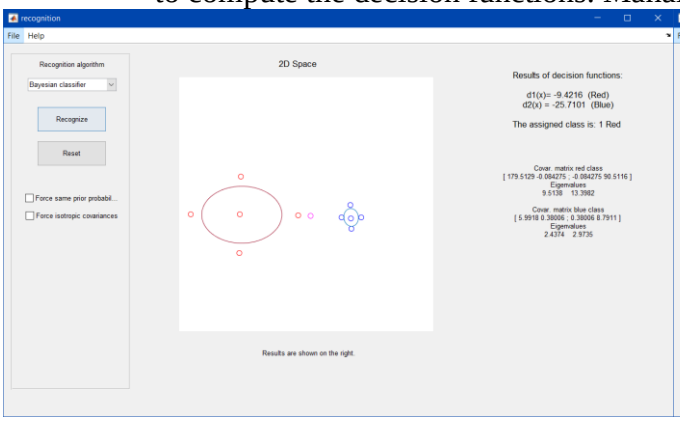


Image 14. Execution with the first configuration

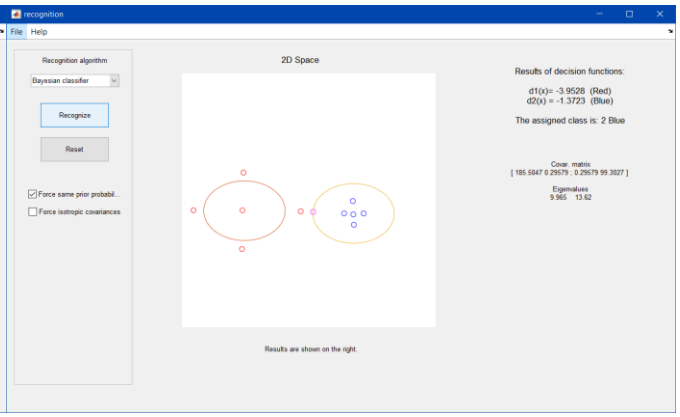


Image 15. Execution with the second configuration

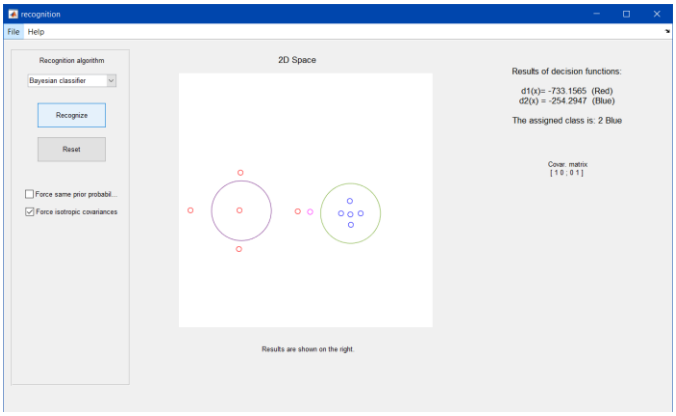


Image 16. Execution with the third configuration

As we see in the images, with the first configuration the algorithm uses the Mahalanobis distance, whereas the second and third ones use the Euclidean distance. It uses the Mahalanobis because the covariances of each gaussian are not equals, and thus there is not equal probability between the two gaussian. In the other hand, the Euclidean is used because the covariances of each gaussian are equal, and thus the probabilities are equals.

EXERCISE 6b: Implementing a classifier

Concepts: Bayesian Classifier

The attached Matlab code partially implements a code for recognizing bottles using the Hu moments as descriptors and a Bayesian Classifier. To complete it:

1. **Centroid and covariance computation:** Develop the code that computes the statistical parameters (centroid and covariance matrix) of the Hu moments of a set of 15 images of three different bottle classes. That is, three centroids and covariance matrices are needed.

```
centroids = [mean(MHu(:,1,1)) mean(MHu(:,1,2)) mean(MHu(:,1,3));
             mean(MHu(:,2,1)) mean(MHu(:,2,2)) mean(MHu(:,2,3));
             mean(MHu(:,3,1)) mean(MHu(:,3,2)) mean(MHu(:,3,3));
             mean(MHu(:,4,1)) mean(MHu(:,4,2)) mean(MHu(:,4,3));
             mean(MHu(:,5,1)) mean(MHu(:,5,2)) mean(MHu(:,5,3));
             mean(MHu(:,6,1)) mean(MHu(:,6,2)) mean(MHu(:,6,3));
             mean(MHu(:,7,1)) mean(MHu(:,7,2)) mean(MHu(:,7,3))];

plot (centroids(1,1),
centroids(2,1),'ks','MarkerSize',8,'MarkerFaceColor','b')
text (centroids(1,1)+0.02, centroids(2,1),'Centroid of bottle type
A','Color','blue')
plot (centroids(1,2),
centroids(2,2),'ks','MarkerSize',8,'MarkerFaceColor','r')
text (centroids(1,2)+0.02, centroids(2,2),'Centroid of bottle type
B','Color','red')
plot (centroids(1,3),
centroids(2,3),'ks','MarkerSize',8,'MarkerFaceColor','g')
text (centroids(1,3)+0.02, centroids(2,3),'Centroid of bottle type
C','Color','green')
```

2. **Visualization:** Show graphically the centroid and covariance of each class (only the two first Hu moments). Use the function **error_ellipse** to represent the covariance matrix. Which information could we get from that ellipses? *Note: we multiply the covariance matrices by a scalar (10) for improving their visibility.*

```
for bt = 1:1:N_bottle_types
    desv=MHu(:,:,bt)'-centroids(:,bt);
    covars(:,:,bt) = desv*desv'/N_bottles;
    error_ellipse(covars(1:2,1:2,bt)*10,centroids(1:2,bt));
end
```

3. **Classifying**: Complete the given loop that reads each of the 5 unused images of the three bottle classes and classify them. For that:
   a. Enable the classification part of the code with **execute_classification.**
   b. Complete the function **evaluateDecisionFunction** that computes the expression $\mathbf{d_k(x)}$ shown below. The input of this function should be: the 7 Hu moments of the initial image, the number of classes in the problem (number of bottle types), he centroid μ^k and covariance Σ^k of the class C^i., and it's a priori probability. *Note: The prior probability is the same for all the classes (p(Ci)= 1/3, i=1,2,3).*

$$d_k(\mathbf{x}) = \ln P(C_k) + \ln p(\mathbf{x}/C_k) = \ln P(C_k) + \ln \frac{1}{(2\pi)^{n/2}|\Sigma^k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}^k)^T(\Sigma^k)^{-1}(\mathbf{x}-\boldsymbol{\mu}^k)}$$

```
function d = evaluateDecisionFunction(x, num_classes, mu, covar,
prior)
    diff=x-mu;
    B=-
(num_classes*log(2*pi)+log(det(covar))+diff'*inv(covar)*diff)/2;
    d =A+B;
end
```

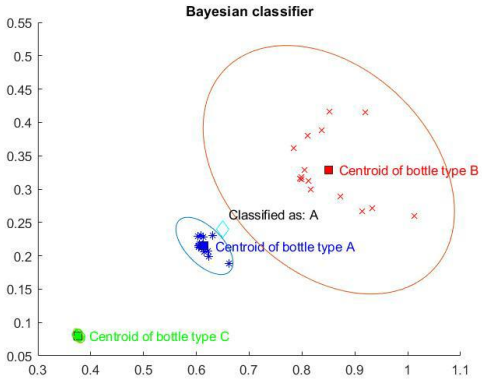
   c. Implement the code that calculates and displays the values of the decision function for the three different classes, and obtains the belonging class of the input image. Use the **max** function to determine the class.

```
N_starting_bottle = N_bottles+1;
N_bottles = 5;

for type = 1:3
      for i_bottle = N_starting_bottle:N_starting_bottle+N_bottles-1
      im_file_name =
      strcat(path,Bottle_types(type),int2str(i_bottle),'.bmp');
      im = imread(im_file_name{1});
      MHu_to_classify(i_bottle,:,type) = momentos_Hu(im);

      features = MHu_to_classify(i_bottle,:,type)';

      % Call the three decisions functions
      d1 =
evaluateDecisionFunction(features,N_bottle_types,centroids(:,1),covars
(:,:,1),1/N_bottle_types);
      d2 =
evaluateDecisionFunction(features,N_bottle_types,centroids(:,2),covars
(:,:,2),1/N_bottle_types);
      d3 =
evaluateDecisionFunction(features,N_bottle_types,centroids(:,3),covars
(:,:,3),1/N_bottle_types);

      % Get the winner!
      [~,classified_as] = max([d1 d2 d3]);
```

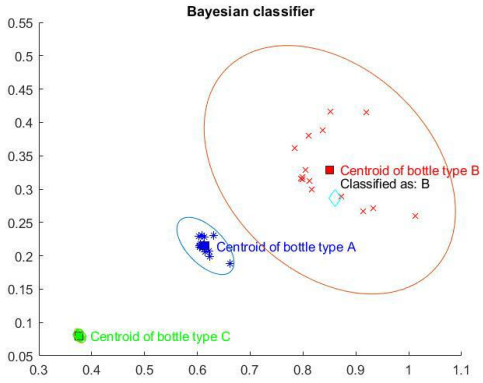
d. Represent in the figure the two first Hu moments of each image.
e. Shows the assigned class side to such first two moments (**text** function).

```
            % Show graphically the results
            res = sprintf('Classified as: %c\n',
types(classified_as));
            handler1 =
plot(features(1),features(2),'dc','MarkerSize',10);
            handler2 = text(features(1)+0.01,features(2)+0.01,res);
            pause

            delete(handler1);
            delete(handler2);
        end
    end
```

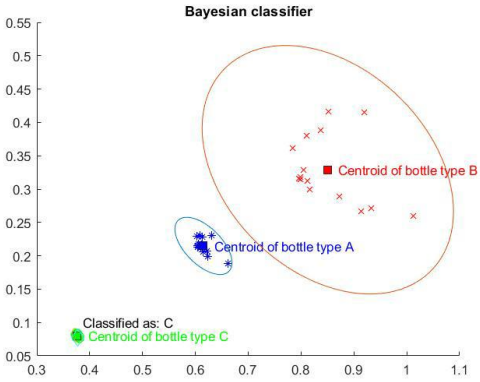
f. Comment the obtained classifications.



Plot 1



Plot 2



Plot 3

As we see in the images, all are classified correctly. We can see it uses the Mahalanobis distance because for the element to classify for Plot 1 is classified as a type A because is more near to the gaussian distribution of A than B. Furthermore, in the calculation of the decision function uses this distance in its quadratic part.

4. **Decision boundaries.** Retrieve the decision boundaries between the pairs of bottle types and show them graphically over the figure using the given function **plotConic.m**. For that we are going to employ the two first Hu moments (it can not be represented using the 7 moments). Recall that the decision boundaries are obtained using the following expression, and that they look as conic sections:

$$d_{ij}(x) = d_i(x) - d_j(x)$$
$$d_i(x) = w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2 + w_4 x_1 + w_5 x_2 + w_6$$

where d_{ij} is the boundary between classes *i* and *j*, and $d_i(x)$ is the decision function of the class *i* assuming that we only employ 2 Hu moments (x_1 and x_2). In this way:

   a. Enable the boundaries computation with **execute_boundaries**.
   b. Obtain the matrix representing the conic section of each decision function using **get_conic_matrix**.

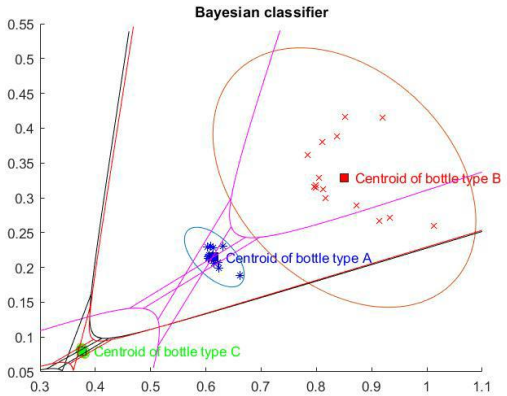
```
conic_matrix_A =
get_conic_matrix(covars(1:2,1:2,1),centroids(1:2,1)');
conic_matrix_B =
get_conic_matrix(covars(1:2,1:2,2),centroids(1:2,2)');
conic_matrix_C =
get_conic_matrix(covars(1:2,1:2,3),centroids(1:2,3)');
```

   c. Get the three decision boundaries by subtracting the corresponding "conic matrix".

```
decision_boundaryAB = conic_matrix_A-conic_matrix_B;
decision_boundaryAC = conic_matrix_A-conic_matrix_C;
decision_boundaryBC = conic_matrix_B-conic_matrix_C;
```

   d. Plot them and discuss the results.

```
plotConic(decision_boundaryAB, 'm');
plotConic(decision_boundaryAC, 'k');
plotConic(decision_boundaryBC, 'r');
```



Plot 4

Depending of the two classes to classify, the result is different. For classes A and B, A elements are going to classify well, however some of the B elements is going to classify as an A element. For the other, all their elements are going to classify well. Thus, if we joint all these boundaries, all the elements are going to classify well, expect for some B elements.

Attached code:

```
%--------------------------------------------------------------------------
%                      EXERCISE: BAYESIAN CLASSIFIER
%--------------------------------------------------------------------------

function exercise_bayesian_clasifier

% Clean the workspace
close all;
clear variables;

% Control the execution of certain parts of the exercise
execute_classification = 0;
execute_boundaries     = 0;

% Load bottle images from file and store their Hu moments
N_bottles = 15;
N_bottle_types = 3;
Bottle_types = {'botella_A_','botella_B_','botella_C_'};
types = ['A','B','C'];
path = 'imagenes botellas/';

for type = 1:3
   for i_bottle = 1:N_bottles
       im_file_name = strcat(path,Bottle_types(type),int2str(i_bottle),'.bmp');
       im = imread(im_file_name{1});
       MHu(i_bottle,:,type) = momentos_Hu(im);
   end
end

% Show the first two moments for the images of each type
figure()
hold on;
title('Bayesian classifier')
plot (MHu(:,1,1),MHu(:,2,1),'b*')
plot (MHu(:,1,2),MHu(:,2,2),'rx')
plot (MHu(:,1,3),MHu(:,2,3),'go')

% Compute their mean and show the centroids of the two first Hu moments
centroids = [mean(MHu(:,1,1)) mean(MHu(:,1,2)) mean(MHu(:,1,3));
             mean(MHu(:,2,1)) mean(MHu(:,2,2)) mean(MHu(:,2,3));
             mean(MHu(:,3,1)) mean(MHu(:,3,2)) mean(MHu(:,3,3));
             mean(MHu(:,4,1)) mean(MHu(:,4,2)) mean(MHu(:,4,3));
             mean(MHu(:,5,1)) mean(MHu(:,5,2)) mean(MHu(:,5,3));
             mean(MHu(:,6,1)) mean(MHu(:,6,2)) mean(MHu(:,6,3));
             mean(MHu(:,7,1)) mean(MHu(:,7,2)) mean(MHu(:,7,3))];


plot (centroids(1,1),
centroids(2,1),'ks','MarkerSize',8,'MarkerFaceColor','b')
text (centroids(1,1)+0.02, centroids(2,1),'Centroid of bottle type
A','Color','blue')
plot (centroids(1,2),
centroids(2,2),'ks','MarkerSize',8,'MarkerFaceColor','r')
text (centroids(1,2)+0.02, centroids(2,2),'Centroid of bottle type
B','Color','red')
plot (centroids(1,3),
centroids(2,3),'ks','MarkerSize',8,'MarkerFaceColor','g')
text (centroids(1,3)+0.02, centroids(2,3),'Centroid of bottle type
C','Color','green')


% Compute their covariance matrices and show them
for bt = 1:1:N_bottle_types
    desv=MHu(:,:,bt)'-centroids(:,bt);
```

```
    covars(:,:,bt) = desv*desv'/N_bottles;
    error_ellipse(covars(1:2,1:2,bt)*10,centroids(1:2,bt));
end

if execute_classification == 1
    %Ok, now load the bottles not used for training, and classify them using
    % a Bayesian classifier

    N_starting_bottle = N_bottles+1;
    N_bottles = 5;

    for type = 1:3
       for i_bottle = N_starting_bottle:N_starting_bottle+N_bottles-1
           im_file_name = strcat(path,Bottle_types(type),int2str(i_bottle),'.bmp');
           im = imread(im_file_name{1});
           MHu_to_classify(i_bottle,:,type) = momentos_Hu(im);

           features = MHu_to_classify(i_bottle,:,type)';

           % Call the three decisions functions
           d1 = evaluateDecisionFunction(features,N_bottle_types,centroids(:,1),
           covars(:,:,1),1/N_bottle_types);
           d2 = evaluateDecisionFunction(features,N_bottle_types,centroids(:,2),
           covars(:,:,2),1/N_bottle_types);
           d3 = evaluateDecisionFunction(features,N_bottle_types,centroids(:,3),
           covars(:,:,3),1/N_bottle_types);

           % Get the winner!
           [~,classified_as] = max([d1 d2 d3]);


           % Show graphically the results
           res = sprintf('Classified as: %c\n', types(classified_as));
           handler1 = plot(features(1),features(2),'dc','MarkerSize',10);
           handler2 = text(features(1)+0.01, features(2)+0.01,res);
           pause

           delete(handler1);
           delete(handler2);
       end
    end
end

if execute_boundaries == 1
    % Plot the decision boundary between all the possible pairs of bottle
    % types: AB, AC and BC.
    % First, get the matrix representing each conic section
    conic_matrix_A = get_conic_matrix(covars(1:2,1:2,1),centroids(1:2,1)');
    conic_matrix_B = get_conic_matrix(covars(1:2,1:2,2),centroids(1:2,2)');
    conic_matrix_C = get_conic_matrix(covars(1:2,1:2,3),centroids(1:2,3)');
    % Get decision boundary between all the types (their conic section
    % representation)
    decision_boundaryAB = conic_matrix_A-conic_matrix_B;
    decision_boundaryAC = conic_matrix_A-conic_matrix_C;
    decision_boundaryBC = conic_matrix_B-conic_matrix_C;

    % Plot them using plotConic
    plotConic(decision_boundaryAB, 'm');
    plotConic(decision_boundaryAC, 'k');
    plotConic(decision_boundaryBC, 'r');
end

end

function d = evaluateDecisionFunction(x, num_classes, mu, covar, prior)
% Evaluate the gaussian deccision function of a vector of features given:
% x: vector of features
```

```
% num_classes: number of classes in the problem
% covar: covariance matrix of the class
% prior: a priori probability of that class
    diff=x-mu;
    B=-(num_classes*log(2*pi)+log(det(covar))+diff'*inv(covar)*diff)/2;
    d =A+B;
end

function conic_matrix = get_conic_matrix(covariance,centroid)
% Returns the matrix representation of a conic section given the:
% covariance: covariance matrix of a two dimensional gaussian
% centoid: centroid [mu1 mu2] of that gaussian
    syms x1 x2
    X = [x1 x2];
    independiente = -1/2*((log(det(covariance))+centroid*covariance.^(-
1)*centroid'));
    lineal = X*covariance.^(-1)*centroid'; % simplify?
    cuadratico = -1/2*X*covariance.^(-1)*X.';
    ecuacion = cuadratico + lineal + independiente;
    [decision,terminos] = coeffs(ecuacion);
    A = double(decision(find(terminos==x1^2)));
    B = double(decision(find(terminos==x1*x2)));
    C = double(decision(find(terminos==x2^2)));
    D = double((decision(find(terminos==x1))));
    E = double(decision(find(terminos==x2)));
    F = double(decision(find(terminos==1)));
    conic_matrix = [A B/2 D/2; B/2 C E/2; D/2 E/2 F];
end
```