ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Προγραμματιστική εργασία σε MATLAB(Αναγνώριση Προτύπων)

PROJECT REPORT

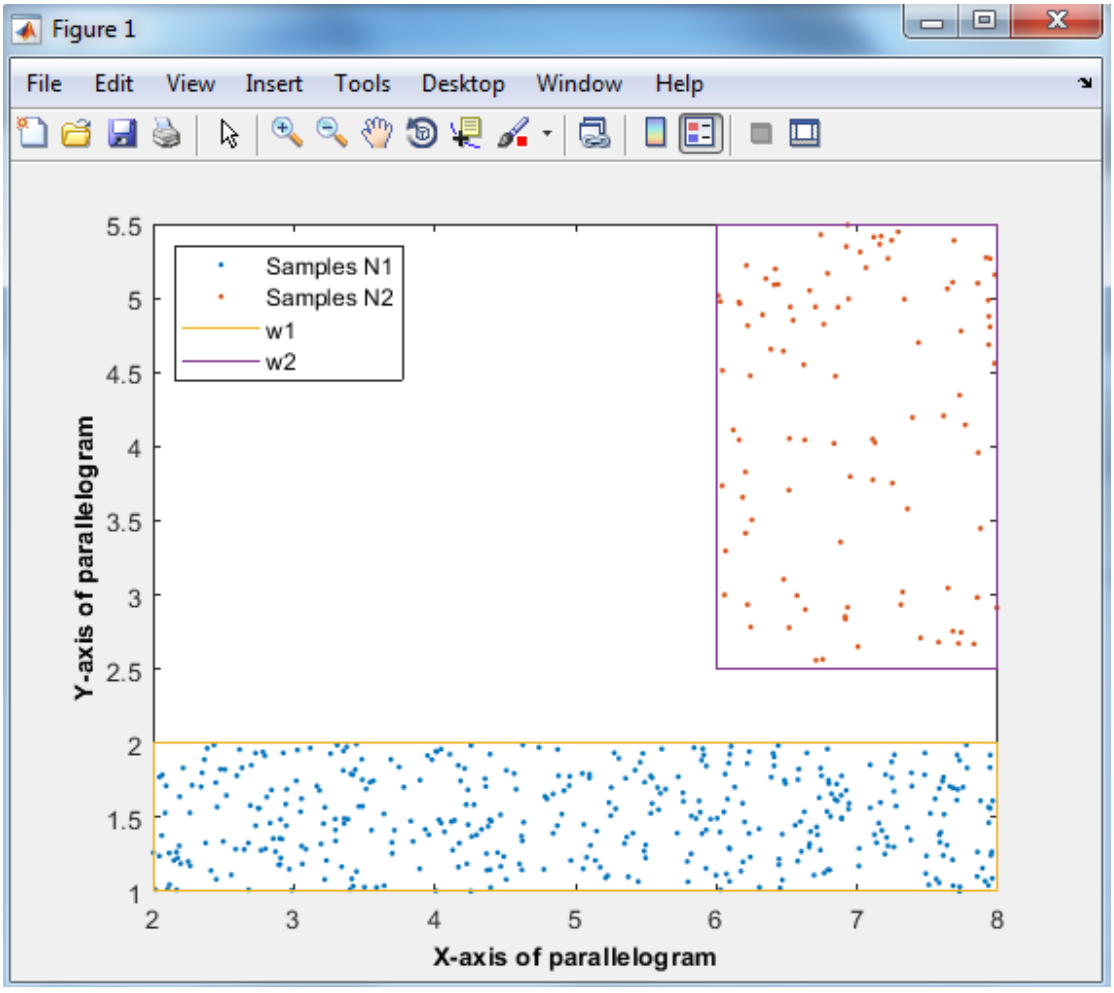
ΟΝΟΜΑΤΕΠΩΝΥΜΟ:ΙΩΑΝΝΗΣ ΜΙΤΡΟ ΑΕΜ:2210

ΟΝΟΜΑΤΕΠΩΝΥΜΟ:ΓΕΩΡΓΙΟΣ ΦΡΑΓΚΙΑΣ ΑΕΜ:2118

Στάδια Υλοποίησης

ΜΕΡΟΣ Α:Γέννηση Δεδομένων

Το πρώτο μας βήμα ήταν η δημιουργία ενός πίνακα μεγέθους 1x400 των δειγμάτων της κλάσης ω1 στο διάστημα [2,8] με την χρήση της συνάρτησης rand. Ο πίνακας αυτός δημιουργήθηκε για τα σημεία του x-άξονα του παραλληλογράμμου της κλάσης ω1.Με την ίδια λογική δημιουργήσαμε πάλι έναν πίνακα μεγέθους 1x400 για τον y-άξονα του παραλληλογράμμου της κλάσης ω1 και αυτή τη φορά. Την ίδια ακριβώς διαδικασία ακολουθήσαμε και την κλάση ω2 με διαστήματα [6,8] και [2.5,5.5] για τον x και y άξονα του παραλληλογράμμου αντίστοιχα. Τα αποτελέσματα των δειγμάτων στις περιοχές των παραλληλογράμμων κάθε κλάσης απεικονίζονται παρακάτω:



ΜΕΡΟΣ Β: Bayesian Ταξινόμηση στον 2-D χώρο

Β1

Το δεύτερο βήμα μας ήταν η εκτίμηση των μέσων τιμών και μητρώων συνδιασποράς των δύο υπό συνθήκη συναρτήσεων πυκνότητας πιθανότητας χρησιμοποιώντας εκτίμηση μέγιστης πιθανοφάνειας (maximum likelihood estimation).Τα δεδομένα μας θεωρούμε ότι ακολουθούν δυσδυάστατες κανονικές κατανομές(multivariate Gaussian distributions).Για την εκτίμηση των παραπάνω έγινε χρήση της συνάρτησης Gaussian_ML_estimate που υλοποιείται στο αρχείο Gaussian_ML_estimate.m για τα δεδομένα της κάθε κλάσης.

Β2:

Για τον ταξινομητή Ευκλείδειας απόστασης, χρησιμοποιούμε τις εκτιμήσεις maximum likelihood των μέσων για την ταξινόμηση των δεδομένων X1(κλάσης ω1) και X2(κλάσης ω2) αντιστοίχως, όπου το z_euclidean είναι ένας N-διάστατος πίνακας που περιέχει τα ορίσματα των κλάσεων όπου τα αντίστοιχα δεδομένα εκχωρούνται από τον ευκλείδειο ταξινομητή. Στη συνέχεια βρήκαμε το σφάλμα του ευκλείδειου ταξινομητή κάθε κλάσης. Σημειώνεται πως για την ταξινόμηση με βάση τον ταξινομητή Ευκλείδειας απόστασης χρησιμοποιήθηκε το αρχείο συνάρτησης euclidean_classifier.m,όπου περιλαμβάνεται η υλοποίηση της Ευκλείδειας ταξινόμησης.

Β3:
Για τον ταξινομητή Mahalanobis απόστασης, ακολουθούμε παρόμοια διαδικασία με αυτή που ακολουθήσαμε στο βήμα Β2. Σημειώνεται πως για την ταξινόμηση με βάση την Mahalanobis απόσταση χρησιμοποιήθηκε το αρχείο συνάρτησης mahalanobis_classifier.m,όπου περιλαμβάνεται η υλοποίηση της .

Β4:

Για τον Bayesian ταξινομητή, χρησιμοποιήστε τη συνάρτηση bayes_classifier του αρχείου bayes_classifier.m και δίνουμε ως είσοδο τους πίνακες S_hat, P, που χρησιμοποιήθηκαν για τη δημιουργία των συνόλων δεδομένων. Με άλλα λόγια, χρησιμοποιούμε τις πραγματικές τιμές του πίνακα μέσου(μέσων τιμών).

ΜΕΡΟΣ Γ: Μείωση Δίαστασης Χαρακτηριστικων

Γ1:

Στο ερώτημα αυτό έπρεπε να βρούμε τη μονοδιάστατη προβολή των δεδομένων του συνόλου εκπαίδευσης κατά μήκος του ιδιο-διανύσματος που αντιστοιχεί στη μεγαλύτερη ιδιοτιμή με βάση τον μετασχηματισμό PCA. Για το συγκεκριμένο μέρος έγινε χρήση του αρχείου pca_fun.m. Τα αποτελέσματα μας φαίνονται παρακάτω:

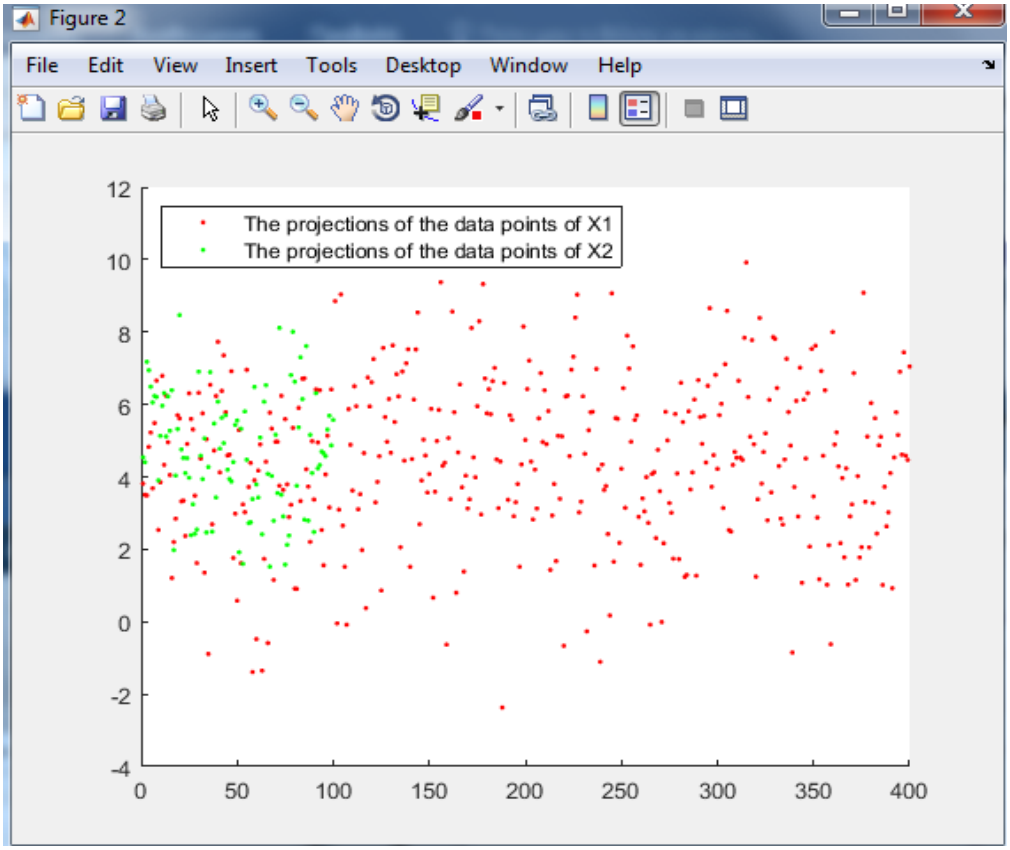


Figure 2
File Edit View Insert Tools Desktop Window Help
The projections of the data points of X1
The projections of the data points of X2
12
10
8
6
4
2
0
-2
-4
0 50 100 150 200 250 300 350 400

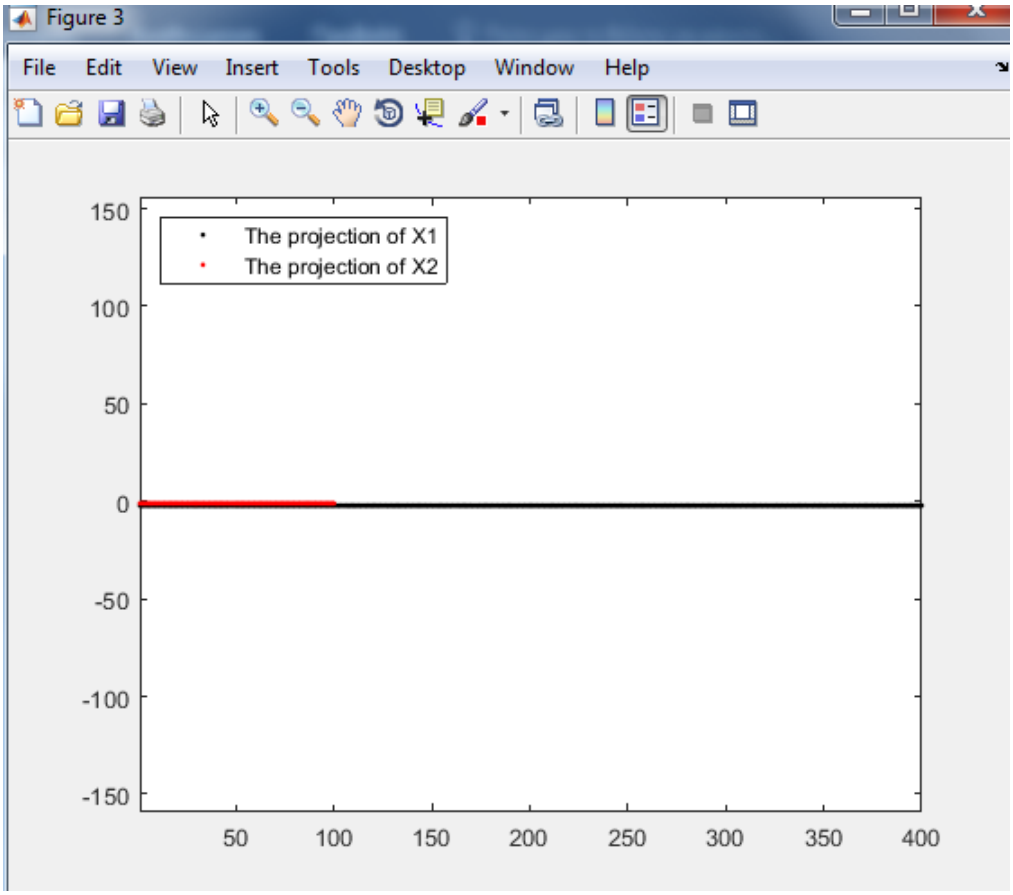
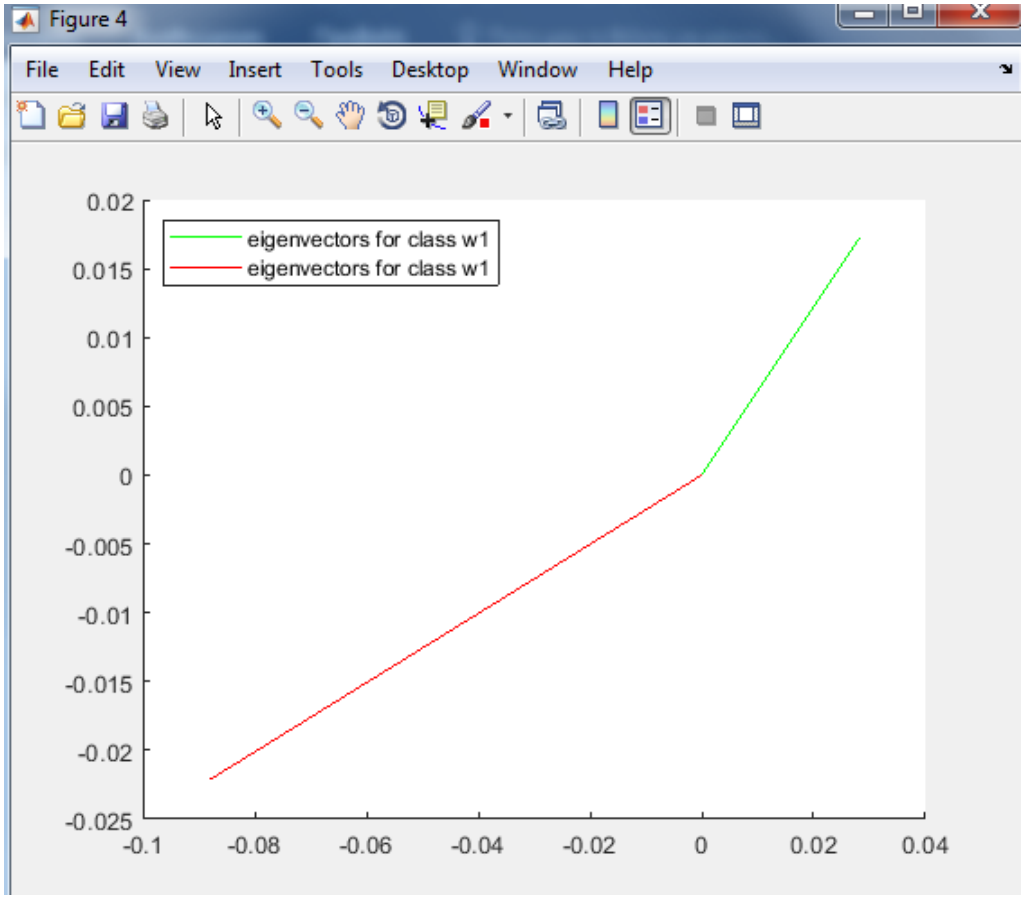


Figure 3
File Edit View Insert Tools Desktop Window Help
The projection of X1
The projection of X2
150
100
50
0
-50
-100
-150
50 100 150 200 250 300 350 400



Γ2:

Σε αυτό το βήμα στόχος ήταν η ταξινόμηση των μονοδιάστατων δεδομένων που προέκυψαν από το βήμα Γ1 με βάση τον ταξινομητή Ευκλείδειας απόστασης και η εύρεση των σφαλμάτων ταξινόμησης .

Γ3:

Στο ερώτημα αυτό έπρεπε να βρούμε τη μονοδιάστατη προβολή των δεδομένων του συνόλου εκπαίδευσης με βάση τον μετασχηματισμό LDA(Linear Discriminant Analysis).Για το συγκεκριμένο μέρος έγινε χρήση του αρχείου scatter_mat.m,η υλοποίηση του οποίου υπολογίζει την κλάση Sw, Sb, Sm . Τα αποτελέσματα μας φαίνονται παρακάτω:

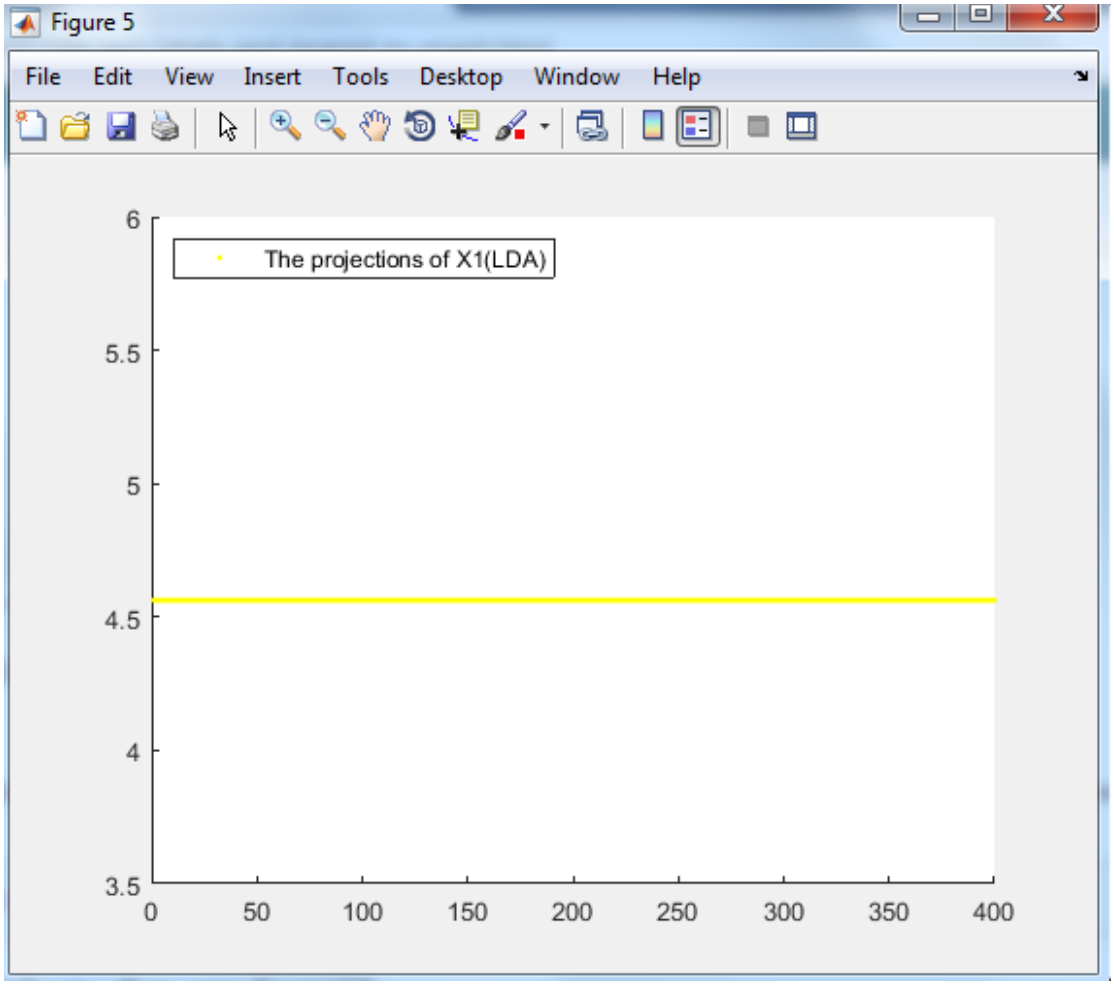
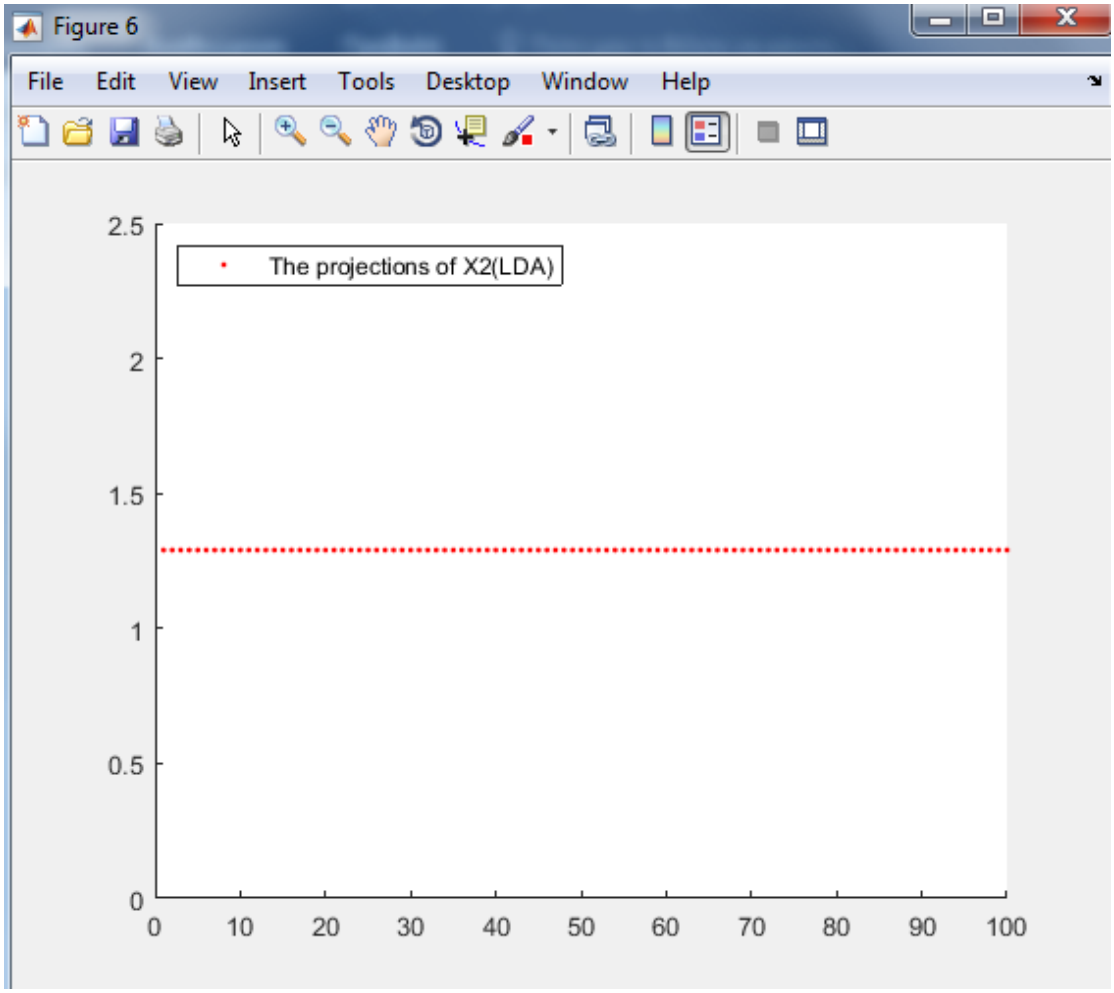


Figure 5
File Edit View Insert Tools Desktop Window Help
The projections of X1(LDA)
6
5.5
5
4.5
4
3.5
0 50 100 150 200 250 300 350 400



Γ4:

Σε αυτό το βήμα στόχος ήταν η ταξινόμηση των μονοδιάστατων δεδομένων που προέκυψαν από το βήμα Γ3 με βάση τον ταξινομητή Ευκλείδειας απόστασης και η εύρεση των σφαλμάτων ταξινόμησης .

ΜΕΡΟΣ Δ: Γραμμική Ταξινόμηση με Διάφορες Συναρτήσεις Κόστους

Δ1:

Σε αυτό το βήμα στόχος ήταν η εύρεση του γραμμικού ταξινομητή(ευθείας)που ελαχιστοποιεί το κριτήριο ελαχίστων τετραγώνων για τα δεδομένα εκπαίδευσης του μέρους Α ,όπου η κλάση ω1 αντιστοιχεί στην ετικέτα y=+1 και η κλάση ω2 στην ετικέτα y=-1.Για τον υπολογισμό του σφάλματος ταξινόμησης χρησιμοποιήθηκε η συνάρτηση SSErr .

Σημείωση

Για την υλοποίηση του τελικού αρχείου μας pattern.m μας έγινε χρήση των παρακάτω αρχείων συναρτήσεων,ο σχετικός κώδικας των οποίων φαίνεται παρακάτω :

- Gaussian_ML_estimate
- euclidean_classifier
- mahalanobis_classifier
- bayes_classifier

- pca_fun
- scatter_mat
- SSErr
- comp_gauss_dens_val.m

pattern.m:

```
%----PART A OF PATTERN RECOGNITION RPOJECT-----
%NAME: IOANNIS MITRO ,AEM: 2210
%NAME: GEORGIOS FRAGKIAS ,AEM:2118



x1min=2
x1max=8
n1=400                    %Number of samples for class w1
x1= x1min +rand(1,n1)*(x1max-x1min) %create a 1x400 size array in the
interval [2,8] for the x-axis of the first parallelogram



y1min=1
y1max=2
y1= y1min +rand(1,n1)*(y1max-y1min) %create a 1x400 size array in the
interval [1,2] for the y-axis of the first parallelogram
p1=plot(x1,y1,'.')          %plot the samples of class w1 represented
with dots inside the first parallelogram
hold on

x2min=6
x2max=8
n2=100                    %Number of samples for class w2
x2= x2min +rand(1,n2)*(x2max-x2min) %create a 1x400 size array with
random values in the interval [6,8] for the x-axis of the second
parallelogram


y2min=2.5
y2max=5.5
y2= y2min +rand(1,n2)*(y2max-y2min) %create a 1x400 size array with
random values in the interval [2.5,5.5] for the y-axis of the second
parallelogram
p2=plot(x2,y2,'.')          %plot the samples of class w2 represented
with dots inside the first parallelogram


%create the first parallelogram as polygon matching  x  with y
coordinates of each edge and piecing them together
xparal1=[2 8 8 2 2]
yparal1=[1 1 2 2 1]
p3=plot(xparal1,yparal1)        %plot the first parallelogram
```

```
%create the second parallelogram as polygon matching  x  with y
coordinates of each edge and piecing them together
xparal2=[6 8 8 6 6]
yparal2=[2.5 2.5 5.5 5.5 2.5]
p4=plot(xparal2,yparal2)        %plot the second parallelogram

xlabel('X-axis of
parallelogram','FontSize',10,'FontWeight','bold','Color','k')
ylabel('Y-axis of
parallelogram','FontSize',10,'FontWeight','bold','Color','k')


legend([p1 p2 p3 p4],{'Samples N1','Samples
N2','w1','w2'},'Location','northwest')


hold off


%-------------------PART B--------------------

%------------------B1---------------------

A=[x1;y1]   %create an array which contains samples of class w1
B=[x2;y2]   %create an array which contains samples of class w2


%alternative way of calculatiing

mean1 = mean(A);    %create the mu matrix
Sigma1 = std(A);    %create matrix S
mean2 = mean(B);
Sigma2 = std(B);

X1 = mvnrnd(mean1,Sigma1,400);  %generate X1
X2 = mvnrnd(mean2,Sigma2,100);  %generate X2


[m,n] = size(X1);
estimated_mean1 = sum(X1)/m;
tmp1=zeros(m,n);
for i=1:n
tmp1(:,i)= ((X1(:,i) - estimated_mean1(i)));
end
covar1 = (tmp1.'*tmp1)/m;

%calculate the estimated mean value and the covariance value using
maximum likelihood estimation for class w2
[h,w] = size(X2);
estimated_mean2 = sum(X2)/h;
tmp2=zeros(h,w);
for i=1:w
tmp2(:,i)= ((X2(:,i) - estimated_mean2(i)));
end
covar2 = (tmp1.'*tmp1)/h;

%i should print the estimated mean_1,estimated_mean2 and
covar1,covar2 in order to see tha matrixes's values
%compute the ML estimates of mean1 and Sigma1 and mean2 and Sigma2
respectively using Gaussian_ML_estimate function
```

```
[m_hat1, S_hat1]=Gaussian_ML_estimate(X1)
[m_hat2, S_hat2]=Gaussian_ML_estimate(X2)
P=[0.8 0.2] %Propability matrix of N1,N2




%i should print the estimated mean_1,estimated_mean2 and
covar1,covar2 in order to see tha matrixes's values
%Note that the returned values depend on the
%initialization of the random generator (involved in function
mvnrnd), so there is a slight deviation
%among experiments


%--------------------B2-----------------------------------
%For the Euclidean distance classifier,we use the ML estimates of the
means to classify the data
%vectors of X1 and X2 respectively,where z_euclidean is an N
-dimensional vector containing the labels of the classes where the
%respective data vectors are assigned by the Euclidean classifier

z_euclidean1=euclidean_classifier(m_hat1,X1)
z_euclidean2=euclidean_classifier(m_hat2,X2)


%error probabilities for the Euclidean classifier
err_euclidean1 = (1-length(find(n==z_euclidean1))/length(n));
err_euclidean2 = (1-length(find(w==z_euclidean2))/length(w));

%-------------------B3-----------------------------------
%Similarly for the Mahalanobis distance classifier


z_mahalanobis1=mahalanobis_classifier(m_hat1,S_hat1,X1)
z_mahalanobis2=mahalanobis_classifier(m_hat2,S_hat2,X2)


%-------------------B4-----------------------------------
%For the Bayesian classifier, use function bayes_classifier and
provide as input the matrices mean,
%Sigma, P, which were used for the data set generation. In other
words, use the true values of mean, Sigma, and P
%and not their estimated values


z_bayesian1=bayes_classifier(m_hat1,S_hat1,P,X1)
z_bayesian2=bayes_classifier(m_hat2,S_hat2,P,X2)


%error probabilities for the  Mahalanobis and Bayesian classifiers
err_mahalanobis1 = (1-length(find(n==z_mahalanobis1))/length(n));
err_mahalanobis2 = (1-length(find(w==z_mahalanobis2))/length(w));
err_bayesian1 = (1-length(find(n==z_bayesian1))/length(n));
err_bayesian2 = (1-length(find(w==z_bayesian2))/length(w));


%i should compare the results in order to conclude about the error
propabilities
```

```
%--------------------PARTC -------------------------
%-------------------PART C1----------------------
%generate data set X1,X2 and the vector y1,y2 respectively


[l1,l2]=size(S_hat1)
mv1= mean1';
N1=400;
X1=[mvnrnd(mv1(:,1),Sigma1,N1)]';
y1=[ones(1,N1)];

%compute the eigenvalues/eigenvectors and variance percentages
M=1;
[eigenval,eigenvec,explained,Y,mean_vec]=pca_fun(X1,M);


[l3,l4]=size(Sigma2)
mv2= mean2';
N2=100;
X2=[mvnrnd(mv2(:,1),Sigma2,N2)]';
y2=[ones(1,N2)];
[eigenval2,eigenvec2,explained2,Y2,mean_vec2]=pca_fun(X2,M);


%Plot of X1
%The projections of the data points of X1 along the direction of the
first principal component
%are contained in the first row of Y, returned by the function pca_
fun
figure(2), hold on
figure(2), p5=plot(X1(1,y1==1),'r.')
%Computation of the projections of X1
w=eigenvec(:,1);
t1=w'*X1(:,y1==1);
X_proj1=[t1;t1]*((w/(w'*w))*ones(1,length(t1)));
%Plot of the projection of X1
figure(3), p6=plot(X_proj1(1,:),'k.')
figure(3), axis equal
%Plot of the eigenvectors for class w1
figure(4), line([0; eigenvec(1,1)], [0;
eigenvec(2,1)],'Color','green')



%Plot of X2
%The projections of the data points of X2 along the direction of the
first principal component
%are contained in the first row of Y, returned by the function pca_
fun
figure(2), hold on
figure(3), hold on
figure(4), hold on

figure(2), p7=plot(X2(1,y2==1),'g.')
%Computation of the projections of X2
```

```
w_new=eigenvec2(:,1);
t2_new=w_new'*X2(:,y2==1);
X_proj2_new=[t2_new;t2_new]*((w_new/
(w_new'*w_new))*ones(1,length(t2_new)));
%Plot of the projection of X2
figure(3), p8=plot(X_proj2_new(1,:),'r.')
figure(3), axis equal
%Plot of the eigenvectors fow class w2
figure(4), line([0; eigenvec2(1,1)], [0;
eigenvec2(2,1)],'Color','red')



legend([p5,p7],{'The projections of the data points of X1','The
projections of the data points of X2'},'Location','northwest')
legend([p6,p8],{'The projection of X1','The projection of
X2'},'Location','northwest')
legend({'eigenvectors for class w1','eigenvectors for class
w1'},'Location','northwest')

%-------------PART C2----------------------
%%error probabilities for the Euclidean classifier for the produced
data after PCA performed

z_euclidean1_2=euclidean_classifier(m_hat1,X1)
z_euclidean2_2=euclidean_classifier(m_hat2,X2)

err_euclidean1_2 = (1-length(find(y1==z_euclidean1_2))/length(y1));
err_euclidean2_2 = (1-length(find(y2==z_euclidean2_2))/length(y2));


%--------------PART C3-----------------------------
%In this part,we are going to apply LDA
%estimate the mean vectors of each class using the available samples,

mv_est1(:,1)=mean(X1(:,y1==1)')';


mv_est2(:,1)=mean(X2(:,y2==1)')';


%ompute the within-scatter matrix S w , use the scatter_mat function,
which computes the within
%class (Sw ), the between class (Sb), and the mixture class (Sm )
for a c-class
%classification problem based on a set of data vector

[Sw1,Sb1,Sm1]=scatter_mat(X1,y1);
[Sw2,Sb2,Sm2]=scatter_mat(X2,y2);

%Since the two classes are not equiprobable, the direction w along
which Fisher's discriminant ratio is
%maximized is computed as follows
w1=inv(Sw1)*(mv_est1(:,1))
w2=inv(Sw2)*(mv_est2(:,1))


%Computation of the new projections
t1=w1'*X1(:,y1==1);
```

```
t2=w2'*X2(:,y2==1);
X_proj1_1=[t1;t1]*((w1/(w1'*w1))*ones(1,length(t1)));
X_proj2_2=[t2;t2]*((w2/(w2'*w2))*ones(1,length(t2)));
figure(5), hold on
figure(5), p9=plot(X_proj1_1(1,y1==1),'y.')
legend(p9,{'The projections of X1(LDA)'},'Location','northwest')

figure(6), hold on
figure(6), p10=plot(X_proj2_2(1,y2==1),'r.')


legend(p10,{'The projections of X2(LDA)'},'Location','northwest')
```

```
%-----------PART C4---------------------------------
z_euclidean1_3=euclidean_classifier(m_hat1,X1)
z_euclidean2_3=euclidean_classifier(m_hat2,X2)

err_euclidean1_3 = (1-length(find(y1==z_euclidean1_3))/length(y1));

err_euclidean2_3 = (1-length(find(y2==z_euclidean2_3))/length(y2));

%Comparing the result depicted in MATLAB figure 1, which was produced
by the execution
%of the previous code, to the corresponding result obtained by the
PCA analysis, it is readily observed
%that the classes remain well separated when the vectors of X2 are
projected along the w direc-
%tion that results from Fisher's discriminant analysis. In contrast,
classes were heavily overlapped
%when they were projected along the principal direction provided by
PCA


%-----------------PART D---------------------------------

%To augment the data vectors of X1 by an additional coordinate that
equals +1, and to change
%the class labels from 1, 2 (used before) to -1, +1, respectively


z1=[ones(1,fix(N1/2)) 2*ones(1,N1-fix(N1/2))];
X1_1=[X1; ones(1,sum(N1))];
y1_1=2*z1-3;

%The set X2 is treated similarly. To compute the classification error
of the LS classifier based on X2

[w]=SSErr(X1_1,y1_1,0);
SSE_out=2*(w'*X1_1>0)-1;
err_SSE=sum(SSE_out.*y1_1<0)/sum(N1)


z2=[ones(1,fix(N2/2)) 2*ones(1,N2-fix(N2/2))];
X2_1=[X2; ones(1,sum(N2))];
y2_1=2*z2-3;
```

```
%The set X2 is treated similarly. To compute the classification error
of the LS classifier based on X2

[w2]=SSErr(X2_1,y2_1,0);
SSE_out_2=2*(w2'*X2_1>0)-1;
err_SSE_2=sum(SSE_out_2.*y2_1<0)/sum(N2)
```

Gaussian_ML_estimate:

```
function [m_hat,S_hat]=Gaussian_ML_estimate(X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FUNCTION
%   [m_hat,S_hat]=Gaussian_ML_estimate(X)
% Maximum Likelihood parameters estimation of a multivariate Gaussian
% distribution, based on a data set X.
%
% INPUT ARGUMENTS:
%   X:      lxN  matrix, whose columns are the data vectors.
%
% OUTPUT ARGUMENTS:
%   m_hat:  l-dimensional estimate of the mean vector of the
distribution.
%   S_hat:  lxl estimate of the covariance matrix of the
distribution.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

[l,N]=size(X);
m_hat=(1/N)*sum(X')';
S_hat=zeros(l);
for k=1:N
    S_hat=S_hat+(X(:,k)-m_hat)*(X(:,k)-m_hat)';
end
S_hat=(1/N)*S_hat;
```

euclidean_classifier:

```
function [z]=euclidean_classifier(m,X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FUNCTION
%   [z]=euclidean_classifier(m,X)
% Euclidean classifier for the case of c classes.
%
% INPUT ARGUMENTS:
%   m:  lxc matrix, whose i-th column corresponds to the mean of the
i-th
%       class.
%   X:  lxN matrix whose columns are the data vectors to be
classified.
%
% OUTPUT ARGUMENTS:
%   z:  N-dimensional vector whose i-th element contains the label
%       of the class where the i-th data vector has been assigned.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        de(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));
    end
    [num,z(i)]=min(de);
end
```

mahalanobis_classifier:

```
function z=mahalanobis_classifier(m,S,X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FUNCTION
%   [z]=mahalanobis_classifier(m,S,X)
% Mahalanobis classifier for c classes.
%
% INPUT ARGUMENTS:
%   m:  lxc matrix, whose i-th column corresponds to the
%       mean of the i-th class
%   S:  lxl matrix which corresponds to the matrix
%       involved in the Mahalanobis distance (when the classes have
%       the same covariance matrix, S equals to this common
covariance
```

```
%       matrix).
%   X:  lxN matrix, whose columns are the data vectors to be
classified.
%
% OUTPUT ARGUMENTS:
%   z:  N-dimensional vector whose i-th component contains the label
%       of the class where the i-th data vector has been assigned.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        dm(j)=sqrt((X(:,i)-m(:,j))'*inv(S)*(X(:,i)-m(:,j)));
    end
    [num,z(i)]=min(dm);
end
```

bayes_classifier:

```
function [z]=bayes_classifier(m,S,P,X)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FUNCTION
%   [z]=bayes_classifier(m,S,P,X)
% Bayesian classification rule for c classes, modeled by Gaussian
% distributions (also used in Chapter 2).
%
% INPUT ARGUMENTS:
%   m:      lxc matrix, whose j-th column is the mean of the j-th
class.
%   S:      lxlxc matrix, where S(:,:,j) corresponds to
%           the covariance matrix of the normal distribution of the
j-th
%           class.
%   P:      c-dimensional vector, whose j-th component is the a
priori
%           probability of the j-th class.
%   X:      lxN matrix, whose columns are the data vectors to be
%           classified.
%
```

```
% OUTPUT ARGUMENTS:
%   z:      N-dimensional vector, whose i-th element is the label
%           of the class where the i-th data vector is classified.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%


[l,c]=size(m);
[l,N]=size(X);

for i=1:N
    for j=1:c
        t(j)=P(j)*comp_gauss_dens_val(m(:,j),S(:,:,j),X(:,i));
    end
    [num,z(i)]=max(t);
end
```

pca_fun:

```
function [eigenval,eigenvec,explain,Y,mean_vec]=pca_fun(X,m)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% FUNCTION
%   [eigenval,eigenvec,explain,Y,mean_vec]=pca_fun(X,m)
% Performs principal component analysis on a data set X and
% returns: (a) the eigenvalues, eigenvectors of the first m principal
% components, (b) the percentage of the total variance explained by
each
% principal component, (c) the projections of the data points to the
% space spanned by the first m principal components and (d) the mean
of X.
%
% INPUT ARGUMENTS:
%   X:      lxN matrix whose columns are the data vectors.
%   m:      the number of the most significant principal components
that
%           are taken into account.
%
% OUTPUT ARGUMENTS:
%   eigenval:   m-dimensional column vector containing the m largest
%               eigenvalues of the covariance matrix of X, in
descending order.
%   eigenvec:   lxm matrix whose i-th column corresponds to
%               the i-th largest eigenvalue of the covariance matrix
of X.
%   explain:    l-dimensional column vector, whose i-th element is
the
%               percentage of the total variance explained by the i-
th
%               principal component.
%   Y:          mxN matrix whose i-th column is the projection
%               of the i-th column vector of X on the space spanned
by the
%               first m principal components.
```

```
%   mean_vec:   the mean vector of X (l-dimensional column vector).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

[l,N]=size(X);

% Subtracting the mean
mean_vec=mean(X')';
X_zero=X-mean_vec*ones(1,N);

% Computing the covariance matrix and its eigenvalues/eigenvectors
R=cov(X_zero');
[V,D]=eig(R);

eigenval=diag(D);
[eigenval,ind]=sort(eigenval,1,'descend');
eigenvec=V(:,ind);

explain=eigenval/sum(eigenval);
% Keeping the first m eigenvaules/eigenvectors
eigenval=eigenval(1:m);
eigenvec=eigenvec(:,1:m);

% Computing the transformation matrix
A=eigenvec(:,1:m)';

% Computing the transformed data set
Y=A*X;
```

scatter_mat:

```
function [Sw,Sb,Sm]=scatter_mat(X,y)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FUNCTION
%   [Sw,Sb,Sm]=scatter_mat(X,y)
% Computes the within scatter matrix, the between scatter
% matrix and the mixture scatter matrix for a c-class classification
% problem, based on a given data set.
%
% INPUT ARGUMENTS:
%   X:      lxN matrix whose columns are the data vectors of the data
set.
%   y:      N-dimensional vector whose i-th component is the label of
%           the class where the i-th data vector belongs (allowable
values
%           are 1,2,...,c).
%
% OUTPUT ARGUMENTS:
%   Sw:     lxl within scatter matrix.
%   Sb:     lxl dimensional between scatter matrix.
%   Sm:     lxl mixture scatter matrix.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
[l,N]=size(X);
c=max(y);

%Computation of class mean vectors, a priori prob. and Sw
m=[];
Sw=zeros(l);
for i=1:c
    y_temp=(y==i);
    X_temp=X(:,y_temp);
    P(i)=sum(y_temp)/N;
    m(:,i)=(mean(X_temp'))';
    Sw=Sw+P(i)*cov(X_temp');
end

%Computation of Sb
m0=(sum(((ones(l,1)*P).*m)'))';
Sb=zeros(l);
for i=1:c
    Sb=Sb+P(i)*((m(:,i)-m0)*(m(:,i)-m0)');
end

%Computation of Sm
Sm=Sw+Sb;
```

SSErr:

```
function [w]=SSErr(X,y,C)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%  [w]=SSErr(X,y,lambda)
% Separates the vectors of two classes contained in a data
% set X with a hyperplane, based on the sum of error squares
criterion.
%
% INPUT ARGUMENTS:
%  X:       lxN matrix whose columns are the data vectors to
%           be classified.
%  y:       N-dimensional vector whose i-th  component contains the
%           label of the class where the i-th data vector belongs (+1
or
%           -1).
%  C:       a small positive user-defined constant that guarantees
the
%           inversion of X*X', when it is near singular.
%
% OUTPUT ARGUMENTS:
%  w:       the l-dimensional parameter vector that corresponds to
the
%           hyperplane that separates the two classes.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
[l,N]=size(X);
w=inv(X*X'+C*eye(l))*(X*y');
```

comp_gauss_dens_val.m:

```
function [z]=comp_gauss_dens_val(m,S,x)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [z]=comp_gauss_dens_val(m,S,x)
% Computes the value of a Gaussian distribution, N(m,S), at a
specific point
% (also used in Chapter 2).
%
% INPUT ARGUMENTS:
%   m:  l-dimensional column vector corresponding to the mean vector
of the
%       gaussian distribution.
%   S:  lxl matrix that corresponds to the covariance matrix of the
%       gaussian distribution.
%   x:  l-dimensional column vector where the value of the gaussian
%       distribution will be evaluated.
%
% OUTPUT ARGUMENTS:
%   z:  the value of the gaussian distribution at x.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[l,c]=size(m);
z=(1/( (2*pi)^(l/2)*det(S)^0.5) )*exp(-0.5*(x-m)'*inv(S)*(x-m));
```

Βιβλιογραφία

[1]Aggelos Pikrakis, Konstantinos Koutroumbas, and Sergios Theodoridis(2010) Introduction to Pattern Recognition: A Matlab Approach