PATTERN RECOGNITION 2022-23 SEMESTER PROJECT

VASILIS STERGIOULIS 03166

IOANNIS KALAMAKIS 03225

ΟΙ ΣΕΛΙΔΕΣ ΠΟΥ ΠΕΡΙΕΧΟΥΝ ΚΩΔΙΚΑ ,ΠΙΝΑΚΕΣ Η ΕΙΚΟΝΕΣ ΠΑΡΑΜΕΝΟΥΝ ΕΠΙΤΗΔΕΣ ΚΕΝΕΣ ΣΤΟ ΥΠΟΛΟΙΠΟ ΤΟΥΣ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Υποσημείωση: Ο συνολικός κώδικας βρίσκεται στο ZIP αρχείο με ονομα **pattern_recognition_code_03166_03225** και ανά ερώτημα θα παρουσιάζεται το μέρος του κώδικα που αντιστοιχεί στο συγκεκριμένο κομμάτι

ΜΕΡΟΣ Α : ΓΕΝΝΗΣΗ ΔΕΔΟΜΕΝΩΝ

Για να μπορέσουμε να δημιουργήσουμε τα ζητούμενα δεδομένα , χρησιμοποιούμε την εντολή **rand**[1] που μας βοηθάει να δημιουργήσουμε πίνακες από ομοιόμορφα κατανεμημένους αριθμούς στο διάστημα [2,8] (επάνω στον άξονα x_1[2]) και [1,2] (επάνω στον άξονα x_2) για την κλάση ω_1 και αντίστοιχα στο διάστημα [6,8] και [2.5 , 5.5] για την ω_2 . Στην συνέχεια , με την βοήθεια την εντολής **scatter**[3] καταφέρνουμε να «οπτικοποιήσουμε » τα δεδομένα τις κάθε κλάσης .Το αποτέλεσμα αυτού εμφανίζεται στο *Σχήμα 1.1* . Επίσης παρατηρούμε ότι λόγου του πλήθους των σημείων των 2 κλάσεων , η $P(\omega_1) = 400/500 = 0.8$ και $P(\omega_2) = 100/500 = 0.2$, άρα οι δύο κλάσεις δεν είναι ισοπίθανες!

Κώδικας Α' μέρους:

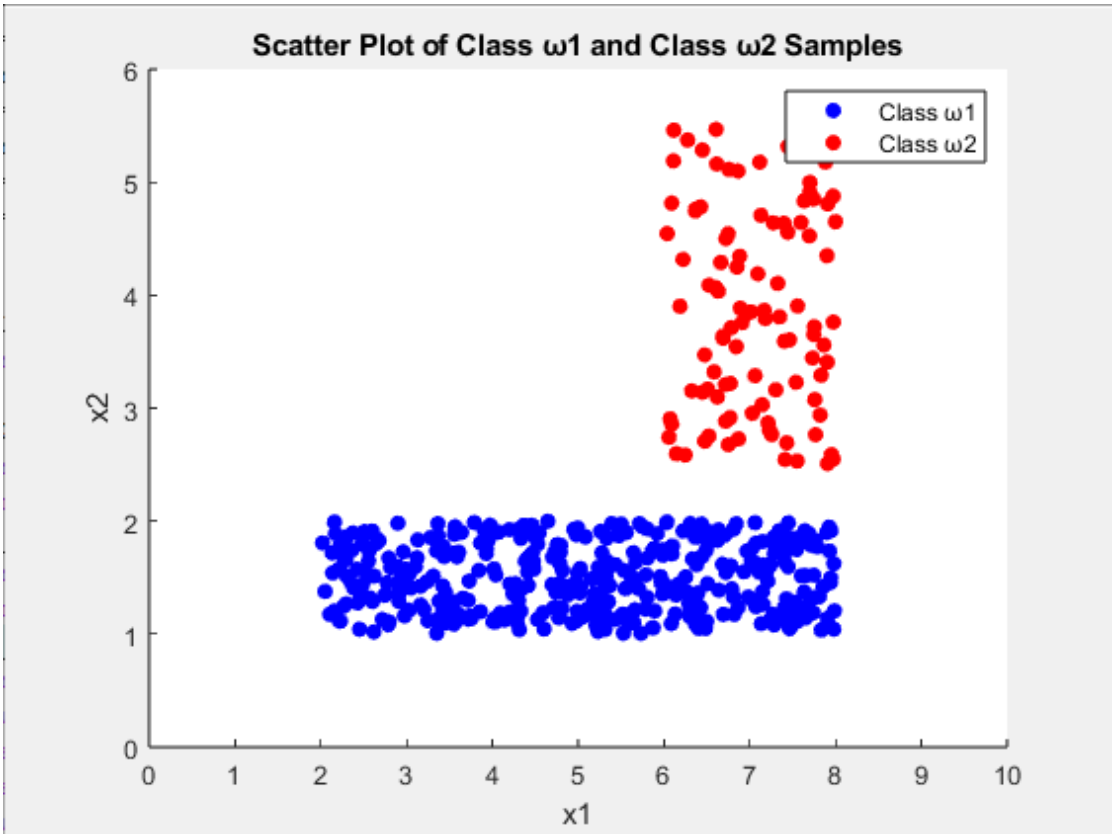
```
% Set the number of samples for each class
    N1 = 400; % Number of samples for class ω1
    N2 = 100; % Number of samples for class ω2

    % Generate the data for ω1
    x1 = rand(N1,1)*6+2; % x1 between [2,8]
    y1 = rand(N1,1)*1+1; % x2 between [1,2]

    % Generate the data for ω1
    x2 = rand(N2,1)*2+6;    % x1 between [6,8]
    y2 = rand(N2,1)*3+2.5;  % x2 between [2.5,5.5]

    % Stack each x and y coordinate for each class
    omega_1 = [x1,y1];
    omega_2 = [x2,y2];

    % Plot the samples
    figure;
    scatter(x1, y1, 'b', 'filled'); % Scatter plot for class ω1
(blue)
    hold on;
    scatter(x2, y2, 'r', 'filled'); % Scatter plot for class ω2 (red)
    hold off;

    % Set plot title and labels
    title('Scatter Plot of Class ω1 and Class ω2 Samples');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);

    % Add legend
    legend('Class ω1', 'Class ω2');
end
```

[1] Η εντολή rand, μας επιτρέπει να δημιουργήσουμε έναν πίνακα από τυχαίους αριθμούς ομοιόμορφα κατανεμημένους στο διάστημα [0,1].

[2] Για λόγους ευκολίας , θεωρούμε στον κώδικα με y1 (για ω1) και y2 για (ω2) τις συντεταγμένες των διανυσμάτων στους άξονες x_1 , x_2

[3] Η εντολή scatter , μας επιτρέπει να δημιουργήσουμε ένα plot στις θέσεις οι οποίες ορίζονται από τις παραμέτρους x,y (Εδώ x: τα σημεία τις κάθε κλάσης στον x_1 , y: τα σημεία τις κάθε κλάσης στον x_2).

Σχήμα 1.1:


Scatter Plot of Class ω1 and Class ω2 Samples
Class ω1
Class ω2
x2
x1
0
1
2
3
4
5
6
7
8
9
10

ΜΕΡΟΣ Β: Bayesian Ταξινομηση στον Δυσδιαστατο Χωρο

Ερώτημα Β1:

Στο συγκεκριμένο ερώτημα , μας ζητείται να υπολογίσουμε τους μέγιστους εκτιμητές πιθανοφάνειας για την μέση τιμή , καθώς και για τον πίνακα συνδιασποράς των δύο κλάσεων . Όμως ο απευθείας προγραμματισμός του ερωτήματος κρίνεται αναγκαία δύσκολος αφού κυρίως θα επιβαρύνει αρκετά το πρόγραμμα και λόγω πολλών υπολογισμών , αλλά είναι αρκετά δύσκολος και αλγόριθμος επίλυσής του. Για τον λόγο αυτό οφείλουμε πρώτα να «βρούμε» τους εκτιμητές στο χαρτί και στην συνέχεια από τον τύπο τους να υπολογίσουμε τις τιμές τους.

Γνωρίζουμε ότι τα δεδομένα μας ακολουθούνε δυσδιάστατες Gaussian κατανομές , άρα και ο τύπος τους:

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\cdot \det(\Sigma_i)^{1/2}} \cdot \exp\left(-\frac{1}{2}\,(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)\ ,\ \ i=1,2$$

Όπου i υποδηλώνει κλάση

Στην συνέχεια , πρέπει να δημιουργήσουμε την συνάρτηση πιθανοφάνειας ως προς τα μ_i και Σ_i.

Με τον τρόπο αυτό και επειδή τα δείγματα είναι στατιστικά ανεξάρτητα έχουμε ότι η συνάρτηση μέγιστης πιθανοφάνειας (για την κάθε κλάση ξεχωριστά) , είναι η :

$$p(x\,;\mu_i,\Sigma_i) = \prod_{n=1}^{N_i} p(x_n\,;\mu_i,\Sigma_i)\ ,\quad i=1,2$$

Με N = 400 για την ω₁ και 100 για την ω₂

Στην συνέχει υπολογίζουμε την λογαριθμική συνάρτηση πιθανοφάνειας (*log-likelihood function*) :

$$L(\mu,\Sigma) = ln\prod_{n=1}^{N} p(x_n\,;\mu\,,\Sigma)^4 \Leftrightarrow$$

$$L(\mu,\Sigma) = \sum_{n=1}^{N} ln[p(x_n\,;\mu\,,\Sigma)] \Leftrightarrow$$

$$L(\mu,S) = 1 - \frac{1}{2}\sum_{n=1}^{N}\ln(2\pi) - \frac{1}{2}\sum_{n=1}^{N}\ln(\det(S)) \quad - \frac{1}{2}\sum_{n=1}^{N}[\,(x_n-\mu)^T S^{-1}(x_n-\mu)]\ ^5$$

Με τον όρο: $\sum_{n=1}^{N}[\,(x_n-\mu)^T S^{-1}(x_n-\mu)]$, να γίνεται , όταν τον αναπτύσσουμε:

---

[4] Εδώ τα indicators δεν αφαιρούνται για να μην υπάρχει σύγχυση , αλλά εννοείται ότι όταν N=400 , τότε μ = μ₁ , Σ= Σ₁ και αντίστοιχα όταν N =100

[5] Για να μην υπάρχει σύγχυση , τον πίνακα συνδιασποράς Σ τον συμβολίζουμε ως S

$$\sum_{n=1}^{N}[\,(x-\,\mu)^T S^{-1}(x-\,\mu)] = \sum_{n=1}^{N} S^{-1}{x_n}^T x_n - \sum_{n=1}^{N} S^{-1}\mu^T x_n - \sum_{n=1}^{N} S^{-1}{x_n}^T \mu \,+ \sum_{n=1}^{N} S^{-1}\mu^T \mu =$$

$$S^{-1}\sum_{n=1}^{N}{x_n}^T x_n - S^{-1}\,\mu^T \sum_{n=1}^{N} x_n - \,S^{-1}\mu \sum_{n=1}^{N}{x_n}^T \,+\, NS^{-1}\mu^T\,\mu$$

Στην συνέχεια υπολογίζουμε την μερική παράγωγο της λογαριθμικής συνάρτησης πιθανοφάνειας δύο φορές , ως προς μ και ως προς S και εξισώνουμε με το 0 , ώστε να υπολογίσουμε τους : $\boldsymbol{\mu}_{\text{MLE}}$, $\mathbf{S}_{\text{MLE}}$.

Για τον $\boldsymbol{\mu}_{\text{MLE}}$:

$$\frac{\partial L(\mu,S)}{\partial \mu} = \,-\,\frac{1}{2}\sum_{n=1}^{N}[-\tfrac{1}{2}\,(x-\,\mu)^T S^{-1}(x-\,\mu)] \Leftrightarrow$$

$$\tfrac{\partial L(\mu,S)}{\partial \mu} = -\,\tfrac{1}{2}[-S^{-1}\,\textstyle\sum_{n=1}^{N} x_n - \,S^{-1}\sum_{n=1}^{N}{x_n}^T \,+\, NS^{-1}2\,\mu]\,,$$

διότι γνωρίζουμε, από την γραμμική άλγεβρα , ότι αν ο X είναι πίνακας, τότε: $\frac{\partial}{\partial X}X^T X = 2X$

και $\frac{\partial}{\partial X}X^T\beta = \beta$

$$\frac{\partial L(\mu,S)}{\partial \mu} = 0 \,\Leftrightarrow$$

$$-S^{-1}\sum_{n=1}^{N} x_n - \,S^{-1}\sum_{n=1}^{N}{x_n}^T \,+\, NS^{-1}2\mu = 0 \Leftrightarrow$$

$$-\sum_{n=1}^{N} x_n - \sum_{n=1}^{N}{x_n}^T \,+\, N2\mu = 0 \Leftrightarrow -\sum_{n=1}^{N} x_n - \sum_{n=1}^{N}{x_n}^T \,+\, N2\mu = 0 \Leftrightarrow$$

$$N2\mu \,= 2\sum_{n=1}^{N} x_n \Leftrightarrow$$ [6]

[6] Διότι $\sum_{n=1}^{N} x_n = \sum_{n=1}^{N}{x_n}^T$, το οποίο είναι λογικό αφού x^T και x έχουν τα ίδια στοιχεία

$$\boldsymbol{\mu_{MLE}} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \boldsymbol{x_n}$$

Δηλαδή, ο maximum likelihood estimator της μέσης τιμής είναι η μέση τιμή (άρα είναι και αμερόληπτος) !

Για τον $\mathbf{S}_{\text{MLE}}$:

$$\frac{\partial L(\mu, S)}{\partial S} = -\frac{\partial}{\partial S} \ln(\det(S))^{7} \frac{1}{2} \sum_{n=1}^{N} \quad --\frac{\partial}{\partial S} \frac{1}{2} \sum_{n=1}^{N} [-\tfrac{1}{2} (x_n - \mu)^T S^{-1} (x_n - \mu)] \Leftrightarrow$$

$$\frac{\partial L(\mu, S)}{\partial S} = -\frac{N}{2} {S^{-1}}^T + \frac{1}{2} S^{-1} \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T] {S^{-1}}^{8}$$

Επειδή ο S^{-1} όμως είναι συμμετρικός τότε ισχύει ότι ${S^{-1}}^T = S^{-1}$

$$\frac{\partial L(\mu, S)}{\partial S} = 0$$

$$\frac{N}{2} S^{-1} = \frac{1}{2} S^{-1} \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T] S^{-1} \underset{\cdot S^{-1}}{\Longleftrightarrow}$$

$$\frac{N}{2} I = \frac{1}{2} \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T] S^{-1} \Leftrightarrow$$

,όπου I ο μοναδιαίος πίνακας

$$NI = \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T] S^{-1} \underset{\cdot S^{-1}}{\Longleftrightarrow}$$

$$SN = \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T]$$

[7] Με βάση το : Derivative of log (det X) | Statistical Odds & Ends (wordpress.com)
[8] Με βάση το : http://www.gatsby.ucl.ac.uk/teaching/courses/sntn/sntn-2017/resources/Matrix_derivatives_cribsheet.pdf

$$\boldsymbol{S_{MLE} = \sum_{n=1}^{N} [(x_n - \mu)(x_n - \mu)^T]}$$

Τέλος , καταφέραμε να βρούμε τους τύπους για το κάθε maximum likelihood estimator , ο υπολογισμός τους καθίσταται υπερβολικά εύκολος.

Για τις κλάσεις του προηγούμενου ερωτήματος το πρόγραμμα δίνει ως αποτέλεσμα τους:

(I) *Πίνακας 2.1.1* για , το $\boldsymbol{\mu}_{\text{MLE}}$ των 2 κλάσεων και

(I) *Πίνακας 2.1.2* για , το $\mathbf{S}_{\text{MLE}}$ των 2 κλάσεων

Κώδικας Ερωτήματος Β1:

```
function [mean_1,sigma_1,mean_2,sigma_2] =
Parameters(omega_1,omega_2)
    %Calculate the mean values
    mean_1 = mean(omega_1);
    mean_2 = mean(omega_2);

    %Calculate the covariances
    sigma_1=[0 0; 0 0];
    sigma_2=[0 0; 0 0];

    for i =1:size(omega_1,1)
        sigma_1= sigma_1 + (omega_1(i,:)-mean_1).*(omega_1(i,:)-
mean_1)';
    end
    sigma_1=sigma_1/size(omega_1,1);

    for i =1:size(omega_2,1)
        sigma_2= sigma_2 + (omega_2(i,:)-mean_2).*(omega_2(i,:)-
mean_2)';
    end
    sigma_2 = sigma_2/size(omega_2,1);
end
```

Πίνακας 2.1.1

```
Mean Value Maximum Likelihood Estimator for Omega 1 :
    5.0946    1.5096


Mean Value Maximum Likelihood Estimator for Omega 2 :
    7.0483    3.8510
```

Πίνακας 2.1.2

```
Covariance Matrix Maximum Likelihood Estimator for Omega 1 :
    2.9184    0.0102
    0.0102    0.0860


Covariance Matrix Maximum Likelihood Estimator for Omega 2 :
    0.3387   -0.0136
   -0.0136    0.7696
```

Ερώτημα Β2:

Ο ταξινομητής της Ευκλείδειας Απόστασης είναι ίσως από τους πιο απλούς ταξινομητές ελάχιστης απόστασης και λειτουργεί με βάση την Ευκλείδεια νόρμα , δηλαδή:

$$D_2 = ||x - \mu_i|| \equiv \sqrt{(x - \mu_i)^T (x - \mu_i)} \leq ||x - \mu_j||, \forall i \neq j$$

όπου i = 1 (συμβολίζοντας την κλάση ω_1) και j = 2 (συμβολίζοντας την κλάση ω_2) , αφού έχουμε μόνο δύο κλάσεις. Επί της ουσίας μετράει την Ευκλείδεια απόσταση για κάθε δεδομένο-διάνυσμα από το κέντρο έκαστης κλάσης και το ταξινομεί στην κλάση ,με την ελάχιστη D_2 . Δηλαδή μπορεί ένα δεδομένο να ανήκει (στην πραγματικότητα) στην ω_1 και ο ταξινομητής να το τοποθετεί στην ω_2 .

Στο *Σχήμα 2.2.1* παρατηρούμε ότι ένα διάνυσμα της ω_2 ταξινομείται εσφαλμένα στην ω_1 , ενώ αρκετά διανύσματα της ω_1 τοποθετούνται στην ω_2 . Το σφάλμα σε αυτήν την περίπτωση αναγράφεται στον *Πίνακα Ρ1* (*σελίδα 42*) και είναι ίσο με 8 % !

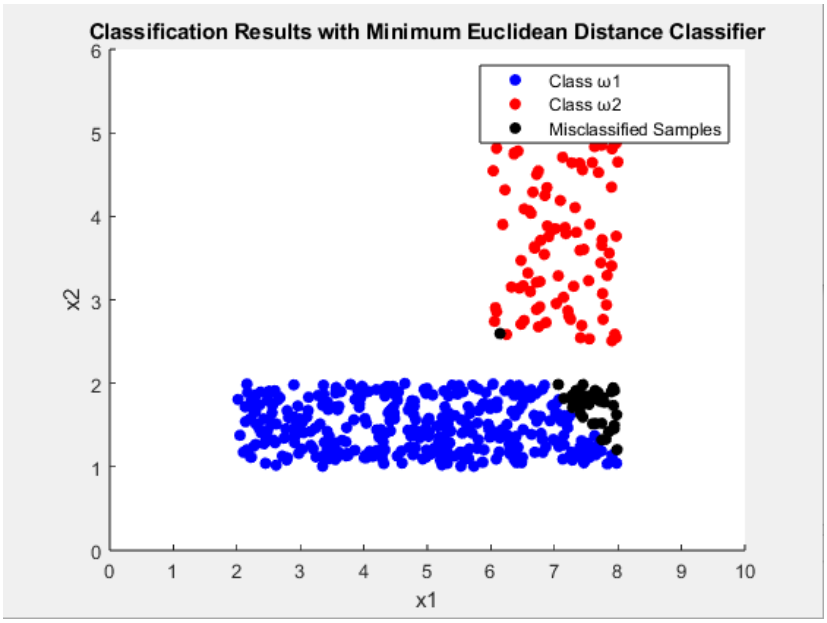
Στην περίπτωση του *Σχήματος 2.2.2* ,όπου τρέξαμε το πρόγραμμα με άλλα δεδομένα για τις 2 κλάσεις , παρατηρούμε ότι μόνο διανύσματα της ω_1 ταξινομούνται στην ω_2 και το σφάλμα είναι ίσο με 3.40 %

Κώδικας Ερωτήματος Β2:

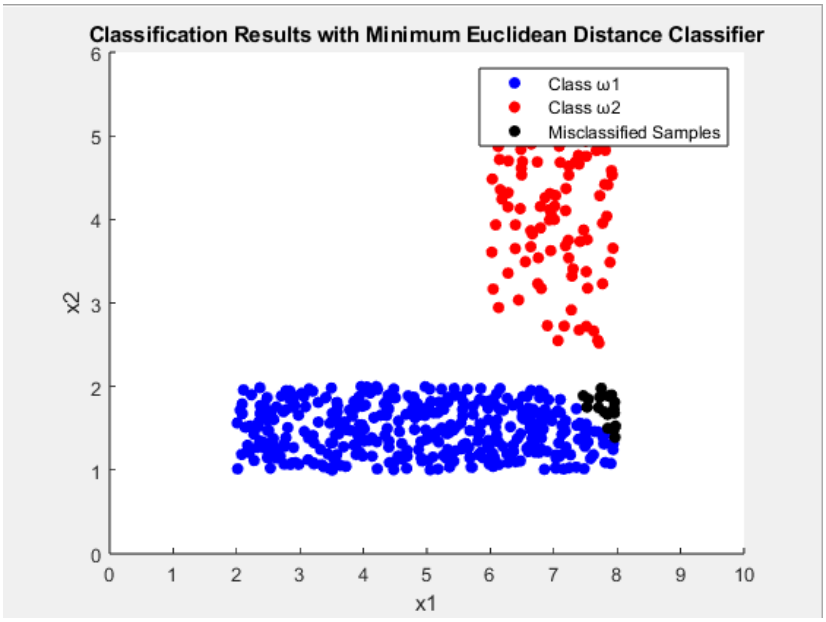
```
function [Classification_error] =
Euclidean_Classifier(omega_1,omega_2,mean_1,mean_2)

    % Classify the training set data using the minimum Euclidean
distance classifier
    N1=400;
    N2=100;
    predictions = zeros(N1+N2,1);
    labels = [ones(N1,1);2*ones(N2,1)];



    % Classify ω1
    for i = 1:N1
        sample_1 = omega_1(i,:);
        distance_1 = norm(sample_1 - mean_1); %Euclidean distance of
a point with the mean (cluster) in ω1
        distance_2 = norm(sample_1 - mean_2); %Euclidean distance of
a point with the mean (cluster) in ω2

        if distance_1<distance_2
            predictions(i) = 1; %Classify to ω1

        else
            predictions(i) = 2; %Classify to ω2
        end
    end
```

```
    % Classify ω2
    for i = 1:N2
        sample_2 = omega_2(i,:);
        distance_1 = norm(sample_2 - mean_1); %Euclidean distance of
a point with the mean (cluster) in ω1
        distance_2 = norm(sample_2 - mean_2); %Euclidean distance of
a point with the mean (cluster) in ω2


        if distance_1<distance_2
            predictions(i+N1) = 1; %Classify to ω1
        else
            predictions(i+N1) = 2; %Classify to ω2
        end
    end

    % Find the error
    Classification_error = sum(predictions ~= labels) / (N1 + N2) *
100;


    misclasses = find(predictions ~= labels);

    misclassed_1 = misclasses(misclasses <= N1);
    misclassed_2 = misclasses(misclasses > N1) - 400;

    % Plot the scatter plot with misclassified samples highlighted
    figure;
    scatter(omega_1(:, 1), omega_1(:, 2), 'b ', 'filled');
    hold on;
    scatter(omega_2(:, 1), omega_2(:, 2), 'r', 'filled');

    %In case there are misclassified data
    if misclassed_1 ~= 0

scatter(omega_1(misclassed_1,1),omega_1(misclassed_1,2),'k','filled')
    end
    if misclassed_2 ~=0

scatter(omega_2(misclassed_2,1),omega_2(misclassed_2,2),'k','filled')
    end
    hold off;

    % Set plot title and labels
    title('Classification Results with Minimum Euclidean Distance
Classifier');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);

    % Add legend
    legend('Class ω1', 'Class ω2', 'Misclassified Samples');

end
```

Σχήμα 2.2.1



Σχήμα 2.2.2



Ερώτημα Β3:

Ο ταξινομητής Mahalanobis , ή αλλιώς η Σ^{-1} νόρμα χαρακτηρίζεται από την :

$$D_{Mahal} = \sqrt{(x - \mu_i)^T \Sigma^{-1} (x - \mu_i)} \leq \sqrt{(x - \mu_j)^T \Sigma^{-1} (x - \mu_j)}, \forall i \neq j$$

Όπου i = 1 , j = 2 και Σ ο κοινός πίνακας συνδιασποράς των δύο κλάσεων.

Στην περίπτωσή μας ο Σ είναι ο ζυγισμένος μέσος όρος των πινάκων συνδιασποράς, δηλαδή είναι ίσος με : $\frac{400}{500} \cdot \Sigma_1 + \frac{100}{500} \cdot \Sigma_2 = 0.8 \cdot \Sigma_1 + 0.2 \cdot \Sigma_2$. Έτσι ακολουθώντας την λογική των ταξινομητών απόστασης , το κάθε σημείο- διάνυσμα συγκαταλέγεται στην κλάση για την οποία η D_{Mahal} είναι η ελάχιστη. Επίσης αξίζει να αναφερθεί ότι με την απόσταση Mahalanobis σταματάμε να μετράμε την απόσταση σημείου από ένα κέντρο , αλλά από μία κατανομή!

Όπως φαίνεται και στο *Σχήμα 2.3.1* τα σημεία με ίση Ευκλείδεια απόσταση « κινούνται » πάνω σε έναν κύκλο , ενώ με ίση Mahalanobis πάνω σε μία έλλειψη. Έτσι , η γραμμή απόφασης σταματάει να είναι κάθετη στο ευθύγραμμο τμήμα που ενώνει τις μέσες τιμές των κλάσεων και η κλίση της εξαρτάται από το σχήμα των ελλείψεων[9]

Τέλος παρατηρούμε στο *Σχήμα 2.3.2* το αποτέλεσμα του ταξινομητή Mahalanobis , ο οποίος ταξινομεί καλύτερα από τον Ευκλείδειο (ταξινομητή) τα δεδομένα των κλάσεων με σφάλμα ίσο με 1 % , *Πίνακας P1* (*σελίδα 42*).

Κώδικας Ερωτήματος Β3:

```
% Classify the training set data using the minimum Mahalanobis
distance classifier
    N1=400;
    N2=100;
    predictions = zeros(500,1);
    labels = [ones(N1,1);2*ones(N2,1)];
    N1=400;
    N2=100;

    %Mutual covariance table
    mutual = (0.8*sigma_1 + 0.2*sigma_2);



    % Classify ω1
    for i = 1:400

        sample_1 = omega_1(i,:);

        distance_1 = mahalanobis_distance(sample_1,mean_1,mutual);
%Mahalanobis distance of a point with the mean (cluster) in ω1
```

[9] Pattern Recognition 4th edition S. Theodoridis , K. Koutrombas , Chapter 2 , page 37-38

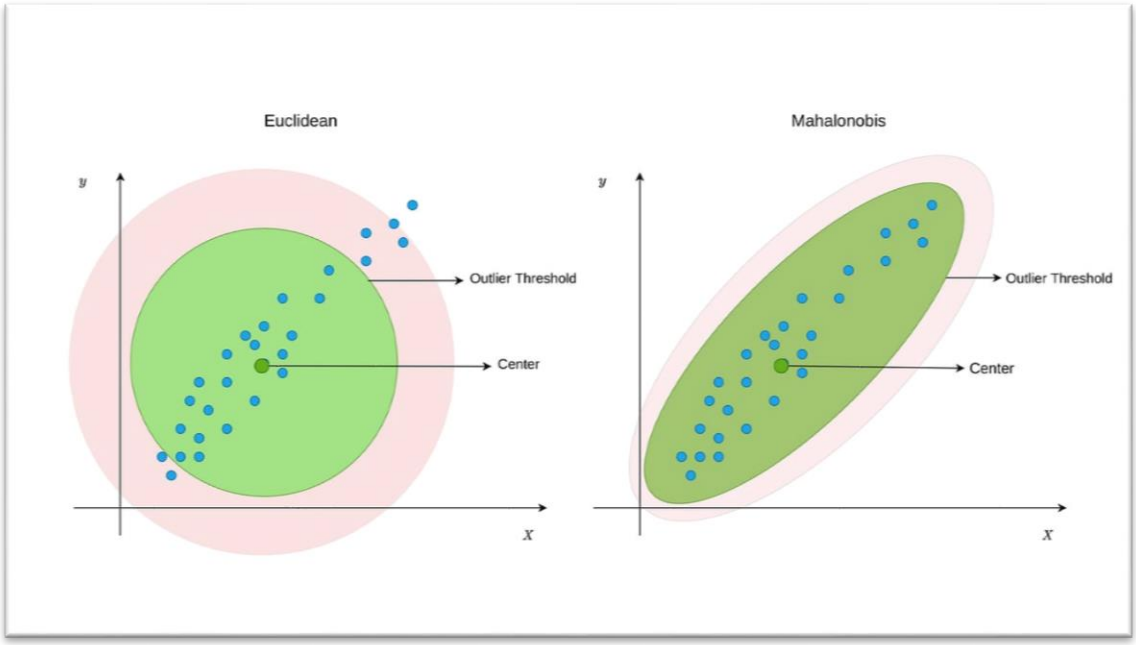
```
        distance_2 = mahalanobis_distance(sample_1,mean_2,mutual);
%Mahalanobis distance of a point with the mean (cluster) in ω2

        if distance_1<distance_2
            predictions(i) = 1; %Classify to ω1

        else
            predictions(i) = 2; %Classify to ω2

        end
    end



    % Classify ω2
    for i = 1:100

        sample_2 = omega_2(i,:);

        distance_1 = mahalanobis_distance(sample_2,mean_1,mutual);
%Mahalanobis distance of a point with the mean (cluster) in ω1
        distance_2 = mahalanobis_distance(sample_2,mean_2,mutual);
%Mahalanobis distance of a point with the mean (cluster) in ω2


        if distance_1<distance_2
            predictions(i+400) = 1; %Classify to ω1

        else
            predictions(i+400) = 2; %Classify to ω2

        end
    end


    % Compute the classification error (%)
    Classification_error = sum(predictions ~= labels) / (N1 + N2) *
100;
    misclasses = find(predictions ~= labels);
    misclassed_1 = misclasses(misclasses <= N1);
    misclassed_2 = misclasses(misclasses > N1) - 400;

    % Plot the scatter plot with misclassified samples highlighted
    figure;
    scatter(omega_1(:, 1), omega_1(:, 2), 'b', 'filled');
    hold on;
    scatter(omega_2(:, 1), omega_2(:, 2), 'r', 'filled');

    %In case there are misclassified data
    if misclassed_1 ~= 0

scatter(omega_1(misclassed_1,1),omega_1(misclassed_1,2),'k','filled')
    end
    if misclassed_2 ~= 0

scatter(omega_2(misclassed_2,1),omega_2(misclassed_2,2),'k','filled')
    end
    hold off;
```

```
    % Set plot title and labels
    title('Classification Results with Minimum Mahalanobis Distance
Classifier');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);

    % Add legend
    legend('Class ω1', 'Class ω2', 'Misclassified Samples');

end
```

, με την εντολή `mahalanobis_distance` :

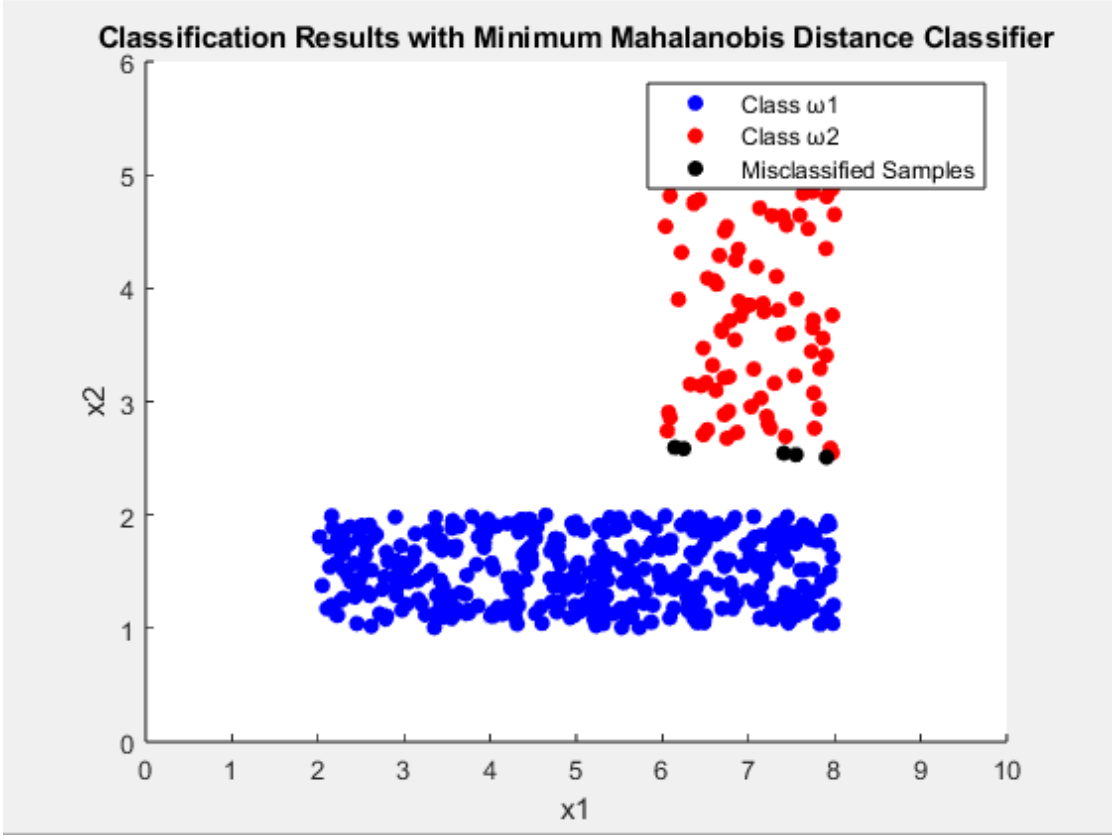
```
distance = ((points - mean)*(inv(covariance))*(points -
mean)')^(1/2);
```

Σχήμα 2.3.1[10]



[10] Η εικόνα πάρθηκε από το : https://towardsdatascience.com/multivariate-outlier-detection-in-python-e946cfc843b3

Σχήμα 2.3.2


Classification Results with Minimum Mahalanobis Distance Classifier
Class ω1
Class ω2
Misclassified Samples
x1
x2
0
1
2
3
4
5
6
7
8
9
10

Ερώτημα Β4:

Ο ταξινομητής κατά Bayes ή αλλιώς Bayesian ταξινομητής , είναι βασισμένος στην θεωρία αποφάσεων κατά Bayes και η κύρια ιδέα του είναι γύρω από τον κανόνα του Bayes , έναν από τους πιο βασικούς κανόνες τις θεωρίας πιθανοτήτων . Αξιόλογο είναι ακόμα και το γεγονός ότι ο συγκεκριμένος ταξινομητής είναι *ο βέλτιστος ως προς την ελαχιστοποίηση της πιθανότητας του σφάλματος ταξινόμησης!*[11]

Κανόνας του Bayes:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \quad , i = 1,2$$

Έτσι η θεωρία αποφάσεων κατά Bayes για 2 κλάσεις:

$$Αν \quad p(x|\omega_1)P(\omega_1) > p(x|\omega_2)P(\omega_2)$$

Τότε χεω_1 , αλλιώς χεω_2 , δηλαδή:

$$p(x|\omega_2)P(\omega_2) > p(x|\omega_1)P(\omega_1)$$

Επίσης αξιοσημείωτο είναι και το εξής (*γενική παρατήρηση*)[12]:

(I) Αν οι κλάσεις είναι ισοπίθανες

(II) Τα δεδομένα σε όλες τις κλάσεις ακολουθούν γκαουσιανές κατανομές

(III) Ο πίνακας συνδιασποράς είναι ο ίδιος για όλες τις κλάσεις

(IV) Και ο πίνακας συνδιασποράς , είναι διαγώνιος και όλα τα στοιχεία της διαγωνίου είναι ίσα (,δηλαδή $\boldsymbol{\Sigma = \sigma^2 I}$)

Τότε ο βέλτιστος Bayesian ταξινομητής εξισώνεται με τον ταξινομητή Ευκλείδειας απόστασης (όπως αυτός παρουσιάζεται και στο *Ερώτημα Β1*) και αν αφαιρέσουμε τον (IV) περιορισμό και είμαστε περισσότερο « χαλαροί » με τους υπόλοιπους , τότε ο βέλτιστοςBayesian ταξινομητής ταυτίζεται με τον ταξινομητή απόστασης Mahalanobis!

Αναγκαστικά λοιπόν , λόγω της φύσης του Bayesian ταξινομητή , αναμένουμε τα αποτελέσματα του να έχουν το ελάχιστο δυνατό σφάλμα από τους δύο προηγούμενους.

Στην περίπτωση των ήδη υπάρχοντων κλάσεων το σφάλμα του , είναι μηδενικό , *Πίνακας Ρ1* (*σελίδα 42*) , και τα δεδομένα έχουν όπως στο *Σχήμα 1:*

Σε μία από τις λίγες περιπτώσεις (*Σχήμα 2.4.1 , Σχήμα 2.4.2*) το σφάλμα του ταξινομητή είναι ίσο με μόλις 0.2 % , τριανταπέντε φορές λιγότερο δηλαδή από αυτό του Euclidean classifier

[11] Pattern Recognition 4th edition , S. Theodoridis , K. Koutrombas , Chapter 2 , page 18

[12] Pattern Recognition A Matlab Approach Kindle edition , S. Theodoridis , K. Koutrombas , A.Pikrakis , D.Kavouras , Chapter 1 , page 6

και υποτριπλάσιο του σφάλματος από την ταξινόμηση κατά Mahalanobis , όπως φαίνεται στον *Πίνακα 2.4.1* .

Κώδικας Ερωτήματος Β4:

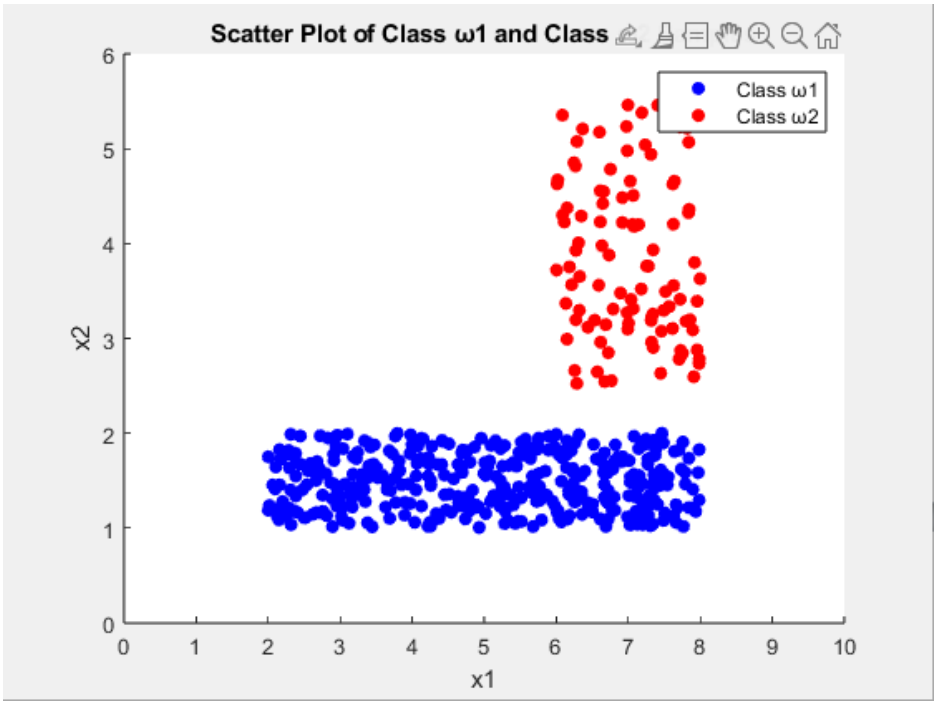
```
    % Classify the training set data using the minimum Euclidean
distance classifier

    N1=400;
    N2=100;
    predictions = zeros(N1+N2,1);
    labels = [ones(N1,1);2*ones(N2,1)];

    for i = 1 :N1

        prob_1 =
(size(omega_1,1)/size(omega_1,1)+size(omega_2,1))*Probability_in_clas
s(omega_1(i,:),mean_1,sigma_1); % Propability of x belongs to ω1
        prob_2 =
(size(omega_2,1)/size(omega_1,1)+size(omega_2,1))*Probability_in_clas
s(omega_1(i,:),mean_2,sigma_2); % Propability of x belongs to ω2

        if prob_1 > prob_2
            predictions(i) = 1; % Classify ω1
        else
            predictions(i) = 2; % Classify ω2
        end
    end

    for i = 1 : N2

        prob_1 =
(size(omega_1,1)/size(omega_1,1)+size(omega_2,1))*Probability_in_clas
s(omega_2(i,:),mean_1,sigma_1); % Propability of x belongs to ω1
        prob_2 =
(size(omega_2,1)/size(omega_1,1)+size(omega_2,1))*Probability_in_clas
s(omega_2(i,:),mean_2,sigma_2); % Propability of x belongs to ω2

        if prob_1 > prob_2
            predictions(i+400) = 1; % Classify ω1
        else
            predictions(i+400) = 2; % Classify ω2
        end

    end

    % Compute the classification error (%)
    Classification_error = sum(predictions ~= labels) / (N1 + N2) *
100;
    misclasses = find(predictions ~= labels);
    misclassed_1 = misclasses(misclasses <= N1);
    misclassed_2 = misclasses(misclasses > N1) - 400;

    % Plot the scatter plot with misclassified samples highlighted
```

```
    figure;
    scatter(omega_1(:, 1), omega_1(:, 2), 'b', 'filled');
    hold on;
    scatter(omega_2(:, 1), omega_2(:, 2), 'r', 'filled');

    %In case there are misclassified data
    if misclassed_1 ~= 0

scatter(omega_1(misclassed_1,1),omega_1(misclassed_1,2),'k','filled')
    end
    if misclassed_2 ~= 0

scatter(omega_2(misclassed_2,1),omega_2(misclassed_2,2),'k','filled')
    end
    hold off;

    % Set plot title and labels
    title('Classification Results with Maximum Bayesian Propability
Classifier');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);

    % Add legend
    legend('Class ω1', 'Class ω2');
    if misclassed_1 ~= 0
        legend('Class ω1', 'Class ω2','Misclassed');
    end
```

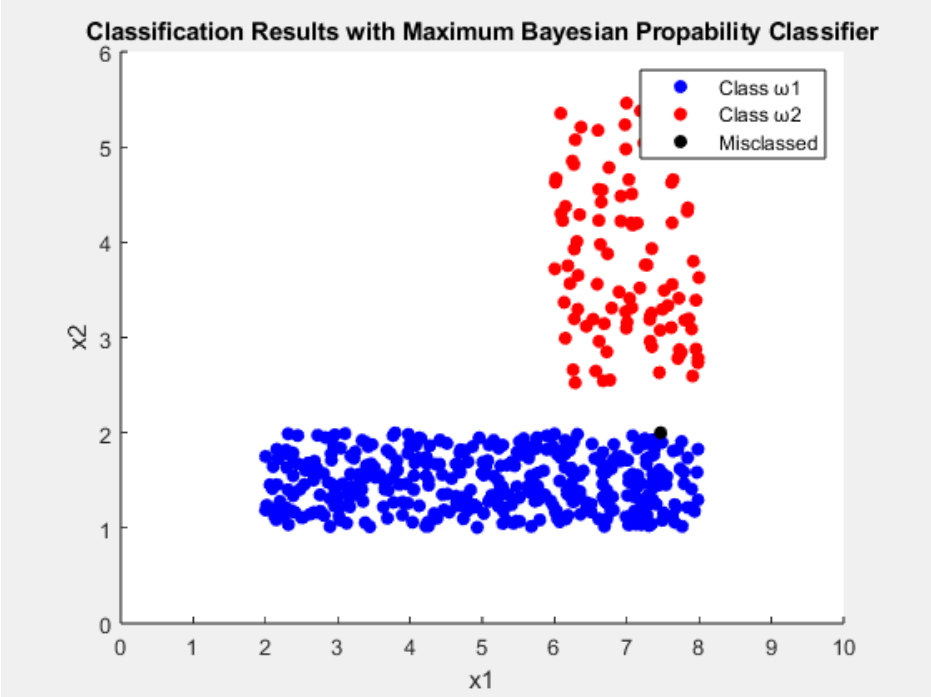
όπου `Probability_in_class:`

```
probability = exp(-(1/2)*(points-means)*(inv(covariance))*(points-
means)')/(2*pi*(det(covariance)^(1/2)));
```

Σχήμα 2.4.1



Σχήμα 2.4.2



Πίνακας 2.4.1 .

```
Euclidean Classifier Error : 7.00%
Mahalanobis Classifier Error : 0.60%
Bayesian Classifier Error : 0.20%
```

ΜΕΡΟΣ Γ : ΜΕΙΩΣΗ ΔΙΑΣΤΑΣΗΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Ερωτήματα Γ1-Γ2:

Κύριο μέλημα του μετασχηματισμού Karhunen-Loève ή αλλιώς PCA (*Principal Component Analysis*) είναι η μείωση τις διάστασης των χαρακτηριστικών και ιδανικά , η μείωση σε μία διάσταση. Ο αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του προβλήματος είναι ο εξής (ελαφρώς παρηλλαγμένος για μεγαλύτερη ευκολία) :

(I) Δημιουργία του πίνακα X , που περιλαμβάνει όλα τα διανύσματα των κλάσεων .

(II) Υπολογισμός της Μέσης τιμής του πίνακα X και η αφαίρεση της από όλα τα στοιχεία του πίνακα X , ώστε να μπορέσουμε να « κεντράρουμε » όλα τα σημεία γύρω από το (0, 0) .

(III) Ο υπολογισμός του πίνακα συνδιασποράς του X (αντί για του πίνακα R_x ή αλλιώς του πίνακα αυτοσυσχέτισης , καθώς η διαφορά μεταξύ των πινάκων είναι ελάχιστη) .

(IV) Υπολογισμός των ιδιοτιμών (*eigenvalues*) και των ιδιοδιανυσμάτων (*eigenvectors*) του πίνακα X και η επιλογή της μέγιστης ιδιοτιμής .

(V) Επιλογή των ιδιοδιανυσμάτων της μέγιστης ιδιοτιμής και η δημιουργία του πίνακα μετασχηματισμού A .

(VI) Δημιουργία του πίνακα που περιέχει τα μετασχηματισμένα δεδομένα επάνω στο επίπεδο της προβολής (W_{pca}).

(VII) Επιλογή του κάθετου διανύσματος στο επίπεδο της προβολής και μείωση της διάστασης των στοιχείων.

Εδώ θεωρούμε ως ευθεία της μειωμένης διάστασης την ευθεία $x_1 = 1$, όπως φαίνεται και στο *Σχήμα 3.1.1* , όπου προβάλλονται τα σημεία μετά το PCA .

Μετά την εφαρμογή του ταξινομητή της Ευκλείδειας απόστασης παρατηρούμε ότι το σφάλμα της ταξινόμησης είναι αρκετά μεγάλο και συγκεκριμένα ίσο με 21 % , *Πίνακας P1* (*σελίδα 42*) , δηλαδή υπερβολικά αυξημένο σε σχέση με την Ευκλείδεια ταξινόμηση στις αρχικές διαστάσεις. Το παραπάνω είναι λογικό , αφού αρκετά σημεία επικαλύπτονται μεταξύ τους

Τέλος , οφείλουμε να αναφέρουμε ότι η μείωση των χαρακτηριστικών με την μέθοδο του PCA είναι μία μη επιβλεπόμενη διαδικασία (*unsupervised*) που στην προβολή των χαρακτηριστικών « κυριαρχεί » η μέγιστη ιδιοτιμή και τα ιδιοδιανύσματα που σχετίζονται με αυτήν.

Κώδικας Ερωτημάτων Γ1- Γ2:

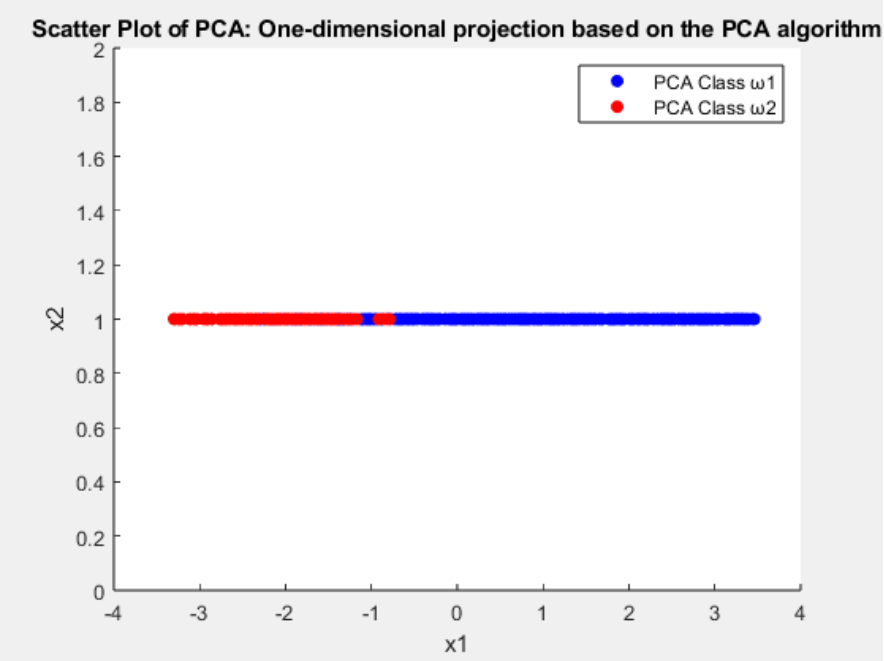
PCA_Classifier

```
    % Compute a new matrix for all classes
    points = cat(1,omega_1,omega_2);
    mean_of_all = mean(points);

    centered_points = points - mean_of_all;
    % Compute the covariance matrix
    matrix = cov(points) ;

    % Compute the eigenvectors and eigenvalues
    [eigenvectors, eigenvalues] = eig(matrix);

    % Sort the eigenvectors in descending order based on eigenvalues
    [~, indices] = max(diag(eigenvalues));
    eigenvectors_sorted = eigenvectors(:, indices);

    % Select the eigenval corresponding to the largest eigenvalue
    projection = centered_points * eigenvectors_sorted;

    % Plot the scatter plot
    figure;
    scatter(projection(1:400),  ones(400, 1), 'b', 'filled');
    hold on;
    scatter(projection(401:500), ones(100,1), 'r', 'filled');
    hold off;

    % Set plot title and labels
    title('Scatter Plot of PCA: One-dimensional projection based on
the PCA algorithm');
    xlabel('x1');
    ylabel('x2');




    % Add legend
    legend('PCA Class ω1', 'PCA Class ω2');
```

Euclidean_classifier_PCA

```
    mean_1 = mean(projection(1:400));
    mean_2 = mean(projection(401:500));
    N1=400;
    N2=100;
    predictions = zeros(N1+N2,1);
    labels = [ones(N1,1);2*ones(N2,1)];
    %points = cat(1,omega_1,omega_2);

    for i = 1:N1
        distance_1 = norm(projection(i)-mean_1);
        distance_2 = norm(projection(i)-mean_2);

        if distance_1 < distance_2
            predictions(i)=1;
            %class_1(p) = projection(i);
            %p = p + 1;

        else
            predictions(i)=2;
            %class_2(j) = projection(i) ;
            %j = j + 1;
            %misclass(k) = projection(i);
            %k = k + 1;
        end
    end



    for i = 400:500

        distance_1 = norm(projection(i)-mean_1);
        distance_2 = norm(projection(i)-mean_2);

        if distance_1 < distance_2
            predictions(i)=1;
            %class_1(p) = projection(i) ;
            %p = p + 1;
            %misclass(k) = projection(i);
            %k = k + 1;
        else
            predictions(i)=2;
            %class_2(j)= projection(i);
            %j=j+1;
        end
    end

    % Compute the classification error (%)
    Classification_error = sum(predictions ~= labels) / (N1 + N2) *
100;
```

Σχήμα 3.1.1


Scatter Plot of PCA: One-dimensional projection based on the PCA algorithm
PCA Class ω1
PCA Class ω2
x2
x1
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-4
-3
-2
-1
0
1
2
3
4

Ερωτήματα Γ3-Γ4:

Το κύριο μέλημα της μεθόδου LDA είναι επίσης η μείωση της διάστασης των χαρακτηριστικών. Η μόνη διαφορά του PCA με LDA , είναι ότι το LDA είναι ένα επιβλεπόμενο (*supervised*) μοντέλο. Στόχος μας αυτή την φορά , είναι η εύρεση της κατεύθυνσης W έτσι ώστε να μεγιστοποιείται ο λόγος διάκρισης Fisher , ή FDR, δηλαδή :

$$\text{FDR} = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

Όπου σ_1^2, σ_2^2 οι διασπορές των δύο κλάσεων (*δεδομένου ότι έχουμε γκαουσιανές κατανομές*).

Με άλλα λόγια θέλουμε τα κέντρα των δύο κλάσεων (άρα συνεπακόλουθα και οι κλάσεις) να είναι όσον το δυνατόν πιο μακριά γίνεται το ένα από το άλλο και οι διασπορές τους , δηλαδή το πόσο «μαζεμένα » είναι τα στοιχεία γύρω από το κέντρο , να είναι όσο το δυνατόν μικρότερες γίνεται . Οι ιδανικές κλάσεις που πληρούν πλήρως τα προηγούμενα κριτήρια εμφανίζεται στο *Σχήμα 3.2.1* .

Επίσης αποδεικνύεται ότι η κατεύθυνση W δίνεται από το μεγαλύτερο ιδιοδιάνυσμα του γινομένου : $\boldsymbol{S_w}^{-1}\boldsymbol{S_b}$ (αν και στις περισσότερες περιπτώσεις απλώς υπολογίζουμε το γινόμενο : ${S_w}^{-1}(\mu_1 - \mu_2)$).

, όπου $\boldsymbol{S_w}$ το μητρώο σκέδασης εντός κλάσης (*within class scatter matrix*) που ορίζεται ως :

$$S_w = \sum_{i=1}^{M} P_i \Sigma_i \quad , M = 2 \text{ (επειδή στο παράδειγμά μας έχουμε 2 κλάσεις)}$$

,με P_i να είναι η πιθανότητα της κάθε κλάσης και Σ_i το μητρώο ή πίνακας συνδιασποράς της κάθε κλάσης και

με $\boldsymbol{S_b}$ το μητρώο σκέδασης μεταξύ κλάσεων (*between class scatter matrix*) που ορίζεται ως :

$$S_b = \sum_{i=1}^{M} P_i(\mu_i - \mu_0)(\mu_i - \mu_0)^T \quad , M = 2$$

, με μ_0 το ολικό μέσο διάνυσμα :

$$\mu_0 = \sum_{i=1}^{M} P_i(\mu_i)$$

Αυτό που παρατηρούμε με την εφαρμογή της LDA μεθόδου στο *Σχήμα 3.2.2* , είναι ότι τα στοιχεία ταξινομούνται ευδιάκριτα (*μεγάλη απόσταση μεταξύ των καινούριων διανυσμάτων των 2 κλάσεων*) , χωρίς κάποιο διάνυσμα της κλάσης ω_1 να επικαλύπτει διάνυσμα της κλάσης ω_2 και το αντίστροφο. Με άλλα λόγια , η διάσταση των υπαρχουσών κλάσεων με την βοήθεια

του LDA , μειώνεται πιο αποτελεσματικά σε σύγκριση με το PCA . Ως ευθεία της μειωμένης διάστασης ,εδώ θεωρούμε επίσης την $x_1 = 1$.

Μετά την χρήση του ταξινομητή της Ευκλείδειας απόστασης παρατηρούμε ότι μπορεί το σφάλμα του LDA να είναι μικρότερο από το σφάλμα του PCA , αλλά και πάλι παραμένει αρκετά μεγάλο , είναι ίσο με 19.60 % *Πίνακας P1* (*σελίδα 42*) , σε σύγκριση με το σφάλμα της ευκλείδειας απόστασης στις αρχικές και μη διαμορφωμένες κλάσεις.

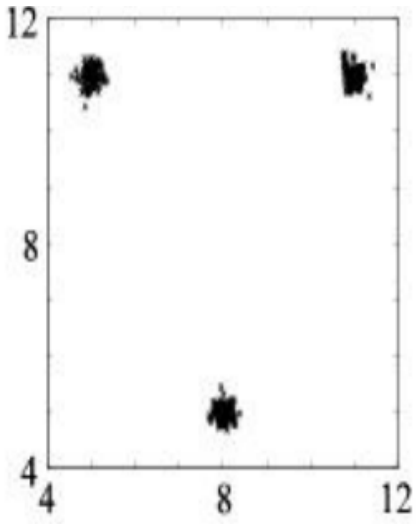
Η παρατήρηση αυτή , όμως , δεν εγγυάται ότι η επαναταξινόμηση των στοιχείων της LDA είναι καλύτερη από αυτή μέσω PCA , όπως φαίνεται στις κλάσεις του *Σχήματος 3.2.3* και συνεπακόλουθα στον *Πίνακα 3.2.1* όπου μπορεί το LDA πάλι να διαχωρίζει καλύτερα τις κλάσεις (*Σχήμα 3.2.4*) σε σχέση με το PCA (*Σχήμα 3.2.5*) αλλά στην επαναταξινόμηση των στοιχείων το σφάλμα του PCA είναι μικρότερο από αυτό του LDA.

Κώδικας Ερωτημάτων Γ3- Γ4:

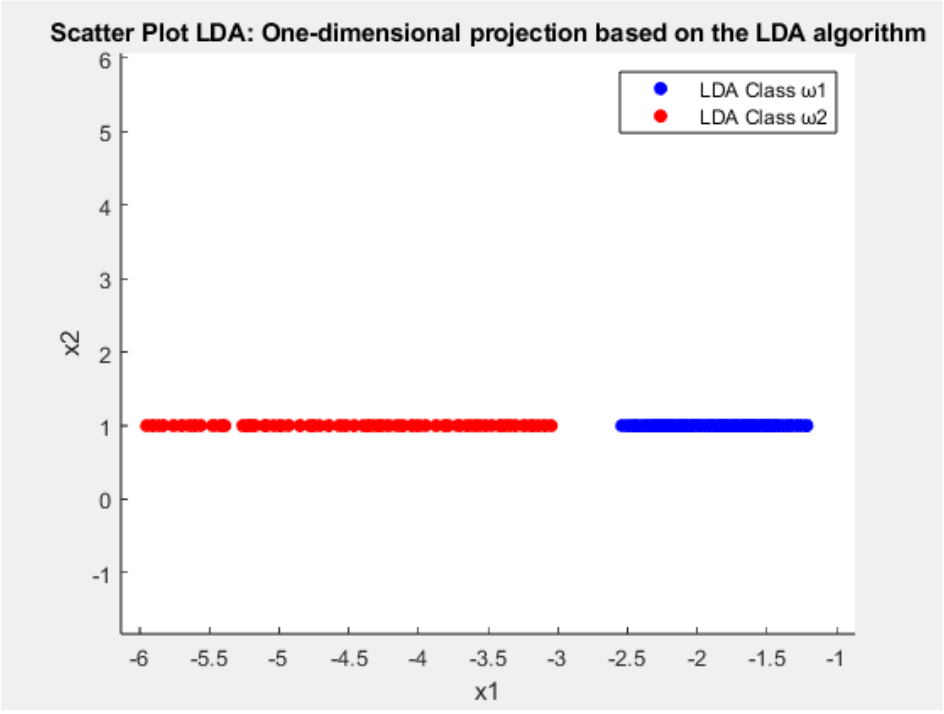
```
    m_omega_1 = mean(omega_1);  % mean of omega 1
    m_omega_2 = mean(omega_2);  % mean of omega 2
    points = cat(1,omega_1,omega_2);

    sigma_1 = cov(omega_1);   %Covariance of ω1
    sigma_2 = cov(omega_2);   %Covariance of ω2

    scatter_matrix = 0.8*sigma_1 + 0.2*sigma_2 ; % P(ω1)* S1 +
P(ω2)*S2
    m0 = 0.8 * m_omega_1 + 0.2 * m_omega_2;
    Sb=0.8*(m_omega_1-m0).*(m_omega_1-m0)' + 0.2*(m_omega_2-
m0).*(m_omega_2-m0)';

    [eigen_vectors_lda, ~] = eig(inv(scatter_matrix) * Sb);
    max_eigen_vector_lda = eigen_vectors_lda(:, end);

    % Project the training data onto the eigenvector
    w_proj = points * max_eigen_vector_lda;
    % Plot the scatter plot
    figure;
    scatter(w_proj(1:400, 1), ones(400,1), 'b', 'filled');
    hold on;
    scatter(w_proj(401:500, 1), ones(100,1), 'r', 'filled');
    hold off;

    % Set plot title and labels
    title('Scatter Plot LDA: One-dimensional projection based on the
LDA algorithm');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([-10, 10]);
    ylim([-10, 20]);
    % Add legend
```

```
    legend('LDA Class ω1', 'LDA Class ω2');


Euclidean_Classifier_LDA


    mean_1 = mean(projection(1:400));
    mean_2 = mean(projection(401:500));
    N1=400;
    N2=100;
    predictions = zeros(N1+N2,1);
    labels = [ones(N1,1);2*ones(N2,1)];

    for i = 1:400
        distance_1 = norm(projection(i)-mean_1);
        distance_2 = norm(projection(i)-mean_2);

        if distance_1<distance_2
            predictions(i)=1;
            %class_1(p) = projection(i);
            %p = p + 1;

        else
            predictions(i)=2;
            %class_2(j) = projection(i) ;
            %j = j + 1;
            %misclass(k) = projection(i);
            %flag_error  = 1+flag_error;
            %k = k + 1;
        end
```

Σχήμα 3.2.1[13]

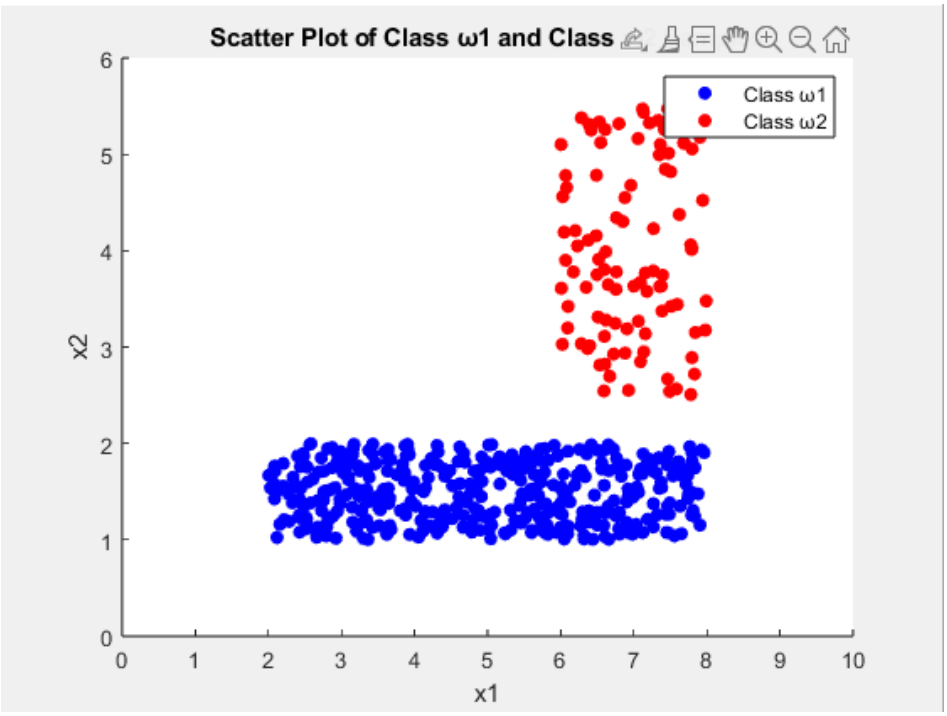


Σχήμα 3.2.2

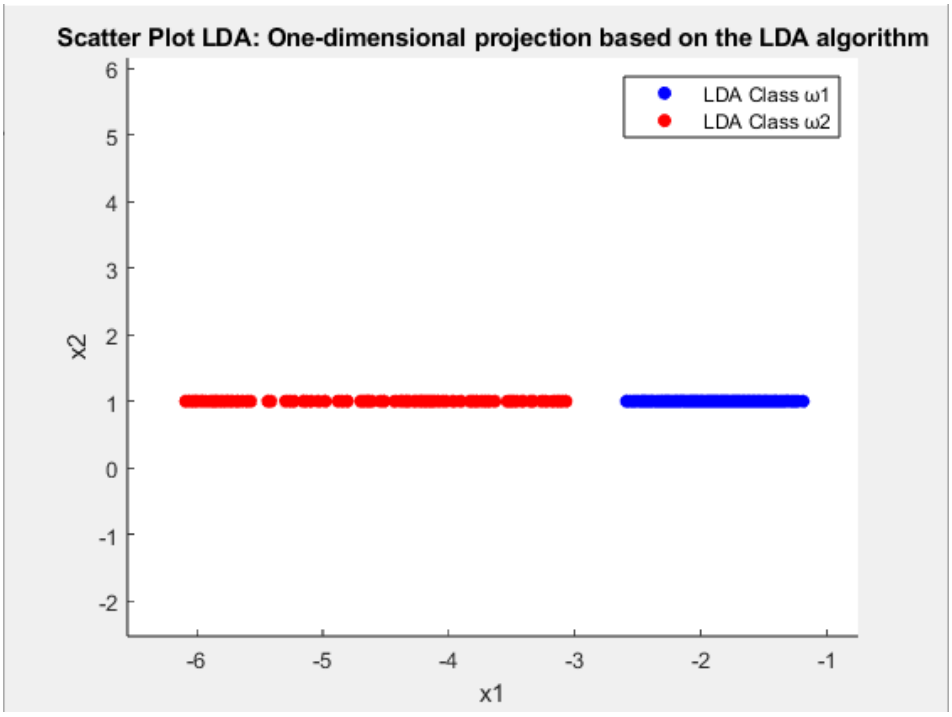


[13] Η φωτογραφία λήφθηκε από την πέμπτη διαφάνεια του μαθήματος

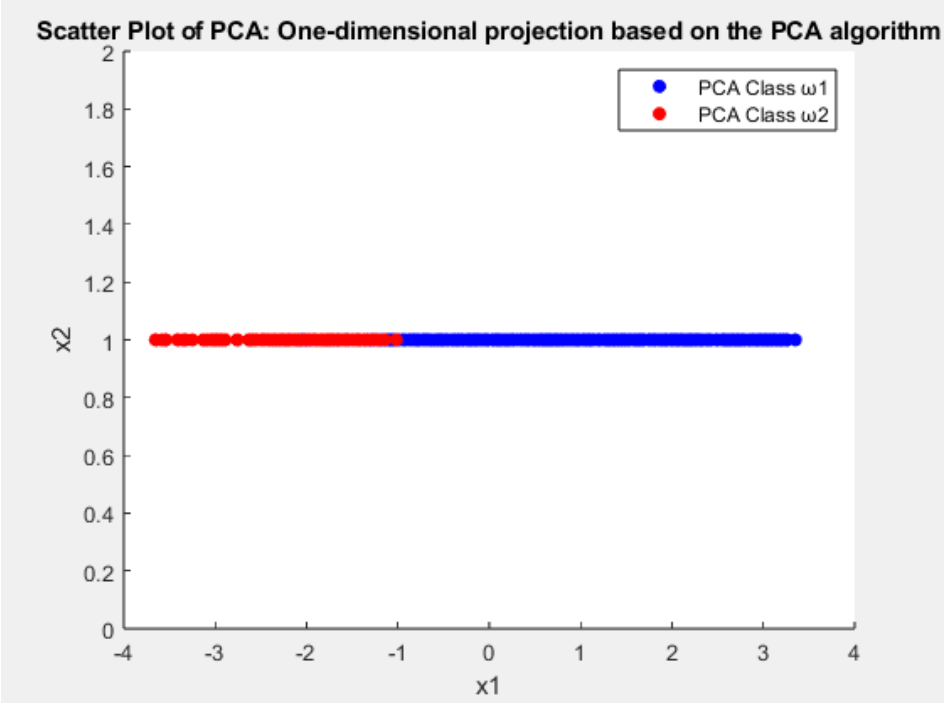
Σχήμα 3.2.3



Σχήμα 3.2.4



Σχήμα 3.2.5



Πίνακας 3.2.1

```
Euclidean Classifier Error : 6.40%
Euclidean Classifier for PCA Error : 18.20%
Euclidean Classifier for LDA Error : 19.00%
```

ΜΕΡΟΣ Δ : ΓΡΑΜΜΙΚΗ ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΔΙΑΦΟΡΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ΚΟΣΤΟΥΣ

Στα τελευταία ερωτήματα , ο στόχος μας , ελαφρώς διαφοροποιείται από τα προηγούμενα. Το κύριο σημείο ενδιαφέροντος , πλέον είναι η απευθείας δημιουργία μιας συνάρτησης-επιφάνειας διακριτοποίησης-διαχωρισμού που διαχωρίζει τις κλάσεις μας (*υπενθυμίζουμε , ότι στο πρόβλημα μας ασχολούμαστε με μόνο δύο κλάσεις*) με τον βέλτιστο δυνατό τρόπο , με την βοήθεια κάποιου κριτηρίου (*δηλαδή μίας συνάρτησης κόστους*) . Επειδή μας επιβάλλεται η ενασχόληση με γραμμικούς ταξινομητές , αναζητούμε ένα βάρος w, τέτοιο ώστε :

$$w^T x + w_0 = 0$$

Ερώτημα Δ1:

Το επιθυμητό αποτέλεσμα του ερωτήματος , είναι ο υπολογισμός ενός διανύσματος παραμέτρων w , στον επαυξημένο χώρο $\boldsymbol{R}^{l+1}$, *όπου* $\lambda = 2$, ώστε :

$$w^T x = 0$$

Όπου x ο επαυξημένος πίνακας κατά μονάδα , όλων στων δεδομένων.

Η συνάρτηση κόστους , που μας ζητείται είναι η : $J[w] = \sum_{i=1}^{N}(y_i - w^T x_i)^2$.

Η μέθοδος αυτή είναι γνωστή και ως Μέθοδος των Ελαχίστων Τετραγώνων και στόχος μας είναι η εύρεση του w , ώστε η παραπάνω συνάρτηση να ελαχιστοποείται . Για τα δεδομένα μας , οι ταμπέλες (*labels*) είναι οι $y_1 = 1$ για την κλάση ω_1 και η $y_2 = -1$ ια την κλάση ω_2 . Λόγω της συνάρτησης κόστους παρατηρούμε ότι αν ελαχιστοποιήσουμε ως προς w , προκύπτει :

$$\sum_{i=1}^{N} x_i(y_i - x_i^T \hat{w}) = 0 \Rightarrow \left(\sum_{i=1}^{N} x_i x_i^T\right)\hat{w} = \sum_{i=1}^{N} y_i x_i$$

Πρώτο βήμα είναι η δημιουργία του πίνακα όλων των στοιχείων X και στην συνέχεια η επαύξησή του με την μονάδα ως τρίτη στήλη , δηλαδή :

$$X = \begin{matrix} x_1^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{matrix}$$

, καθώς και του πίνακα y , όπου:

$$y = \begin{matrix} 1\,, \alpha\nu\ x_n \in \omega_1 \\ -1\,, \alpha\nu\ x_n \in \omega_2 \end{matrix}$$

Στην συνέχεια και ακολουθώντας την μεθοδολογία του βιβλίου[14] , υπολογίζουμε τα γινόμενα $X^T X$ και $X^T y$, διότι $\sum_{i=1}^{N} x_i x_i^T = X^T X$ και $\sum_{i=1}^{N} y_i x_i = X^T y$.Επομένως η εξίσωσή μας γίνεται :

$$(X^T X)\hat{w} = X^T y \Rightarrow \hat{w} = (X^T X)^{-1} X^T y$$

[14] Pattern Recognition 4th edition , S. Theodoridis , K. Koutrombas , Chapter 3 , page 118

Στην πράξη όμως ο υπολογισμός του $X^T X$, δύναται εμφανίζει αρκετές αριθμητικές δυσκολίες σε μεγάλο αριθμό διαστάσεων , συν του ότι ο υπολογισμός του είναι υπολογιστικά περίπλοκος[15] ($O(n^2 ln\ n)$ *και στην χειρότερη περίπτωση* $O(n^k)$ *, όπου* $k \approx 2.373$ *, για τετράγωνικούς πίνακες όπως και ο* $X^T X$) και αρκετές φορές η ορίζουσα (*det*) του πίνακα είναι μηδενική !

Έτσι αρκετές φορές θεωρούμε μια μικρή σταθερά C [16]και την προσθέτουμε στην κύρια διαγώνιο του $X^T X$, με την βοήθεια του μοναδιαίου πίνακα I . Με άλλα λόγια , η εξίσωση γράφεται ως : $\hat{w} = (X^T X + CI)^{-1} X^T y$. Στην περίπτωση μας θεωρούμε κάθε φορά ότι η σταθερά C είναι συνεχώς μηδενική .

Μετά την εφαρμογή του αλγορίθμου η επιφάνεια διαχωρισμού (*Σχήμα 4.1.1*), δεν διαχωρίζει τέλεια όλα τα σημεία και εμφανίζει σφάλμα ίσο με 1.20 % , *Πίνακας P1* (*σελίδα 42*) .Η ευθεία του διαχωρισμού στην συγκεκριμένη περίπτωση είναι η : $-0.507x_1 - 0.6033\ x_2 + 2.9074 = 0$.

Κώδικας Ερωτήματος Δ1 :

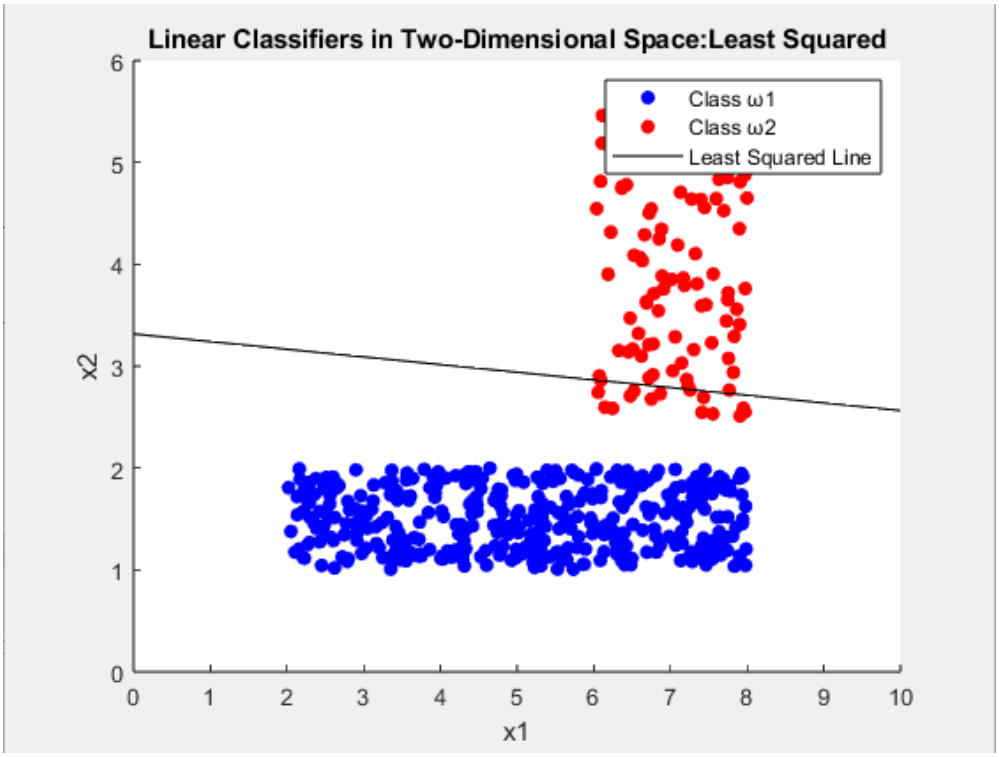
```
    X = cat(1,omega_1,omega_2); % getting all the class points to one
matrix
    y = ones(500,1);    % create a 3rd pseudo - dimension for the
extended matrix
    X = cat(2,X,y);    % Now we have the extended matrix

    for i = 1:100
        y(i+400,1) = -1 ; % Now we have created the y matrix
    end

    cor_mat = X'*X;
    d = X'*y;
    Wcoeff = linsolve(cor_mat,d); % The coefficients of the
seperation line (in order to plot it and find the error

    coeffx1 = Wcoeff(1);
    coeffx2 = Wcoeff(2);
    c = Wcoeff(3);
    sum_error = 0;
    Num_of_mis_points = 0;

    % Computing The J[w] or the Error
    for i = 1:400
        if (coeffx1*omega_1(i,1)+ coeffx2*omega_1(i,2) + c) < 0
            sum_error = sum_error + (1 - (coeffx1*omega(i,1)+
coeffx2*omega_1(i,2) + c))^2 ;
            Num_of_mis_points = Num_of_mis_points +1;
        end
    end
```

[15] Με βάση το : https://math.stackexchange.com/questions/2154503/can-matrices-of-the-form-xtx-be-more-efficiently-inverted

[16] Pattern Recognition A Matlab Approach Kindle edition , S. Theodoridis , K. Koutrombas , A.Pikrakis , D.Kavouras , Chapter 2 , page 35.

```
    for i = 1:100
        if (coeffx1*omega_2(i,1)+ coeffx2*omega_2(i,2) + c) > 0
            sum_error = sum_error + (-1 - (coeffx1*omega_2(i,1)+
coeffx2*omega_2(i,2) + c))^2 ;
            Num_of_mis_points = Num_of_mis_points +1;
        end
    end

    total_error = sum_error/ Num_of_mis_points ;
    % Plot the scatter plot
    x1 = linspace(0, 20);
    %x2 = linspace(0, 20);
    line = -(coeffx1*x1+c)/coeffx2 ;

    figure;
    scatter(omega_1(:, 1), omega_1(:, 2), 'b', 'filled');
    hold on;
    scatter(omega_2(:, 1), omega_2(:, 2), 'r', 'filled');
    plot(x1,line,'k-');
    hold off;

    % Set plot title and labels
    title('Linear Classifiers in Two-Dimensional Space:Least
Squared');
    xlabel('x1');
    ylabel('x2');


    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);

    % Add legend
    legend('Class ω1', 'Class ω2','Least Squared Line');
```

Σχήμα 4.1.1


Linear Classifiers in Two-Dimensional Space:Least Squared
Class ω1
Class ω2
Least Squared Line
x2
x1
0
1
2
3
4
5
6
7
8
9
10

Ερώτημα Δ2:

Γνωρίζουμε ότι για μία μικρή τιμή $\rho_t = \rho = c$, όπου c σταθερά και $0 < \rho < 2$, ο αλγόριθμος perceptron συγκλίνει σε σχετικά μικρό αριθμό βημάτων[17] . Επίσης , παρατηρούμε στο *Σχήμα 4.2.1* ότι η ευθεία $x_2 = 2.3$ είναι μία καλή (αν όχι η βέλτιστη) λύση του προβλήματος , αφού όλα τα διανύσματα χωρίζονται τέλεια (*δεν υπάρχει misclassification error*) . Με την εφαρμογή του perceptron , ουσιαστικά επιθυμούμε να καταλήξουμε σε μία ευθεία « κοντά » στην προηγούμενη . Για το λόγο αυτό , η αρχική ευθεία διαχωρισμού που θεωρούμε είναι η $x_2 = 2.7$, η οποία είναι περίπου ίδια με την εξίσωση διαχωρισμού του προηγούμενου ερωτήματος και ταξινομεί λάθος μερικά διανύσματα της ω_2 . Επειδή η ευθεία που διαλέξαμε , είναι αρκετά κοντά με την καλύτερη λύση του *Σχήματος 4.2.1* , επιλέγουμε ρυθμό μάθησης $\rho = 0.001$ με σκοπό να συγκλίνει γρήγορα ο αλγόριθμος και να έχουμε τα λιγότερα βήματα-επαναλήψεις .

Στην συνέχεια ακολουθούμε τον perceptron αλγόριθμο σε μορφή batch , όπως αυτός αναγράφεται στο βιβλίο[18] και καταλήγουμε μετά από 7 επαναλήψεις στην βέλτιστη ευθεία – λύση με το μεγαλύτερο σφάλμα στην διαδικασία να είναι ίσο με 0.62 % (υποδιπλάσιο από αυτό του προηγούμενου ερωτήματος) ! Τα σφάλματα σε κάθε επανάληψη εμφανίζονται στον *Πίνακα P1* (*σελίδα 42*) και η τελική ευθεία διαχωρισμού (*Σχήμα 4.2.2*) , είναι η : $-0.07x_1 - 1.03\,x_2 + 2.98 = 0$, σχετικά κοντά με την αναμενόμενη ευθεία $x_2 = 2.3$.

Στην περίπτωση των κλάσεων του *Σχήματος 4.2.3*, η τελική ευθεία είναι πάλι η ίδια με μόνη διαφορά , ότι ο αλγόριθμος τελείωσε σε μόλις 3 επαναλήψεις (*Πίνακας 4.2.1 , Σχήμα 4.2.4*) και είχε μικρότερο σφάλμα .

Κώδικας Ερωτήματος Δ2 :

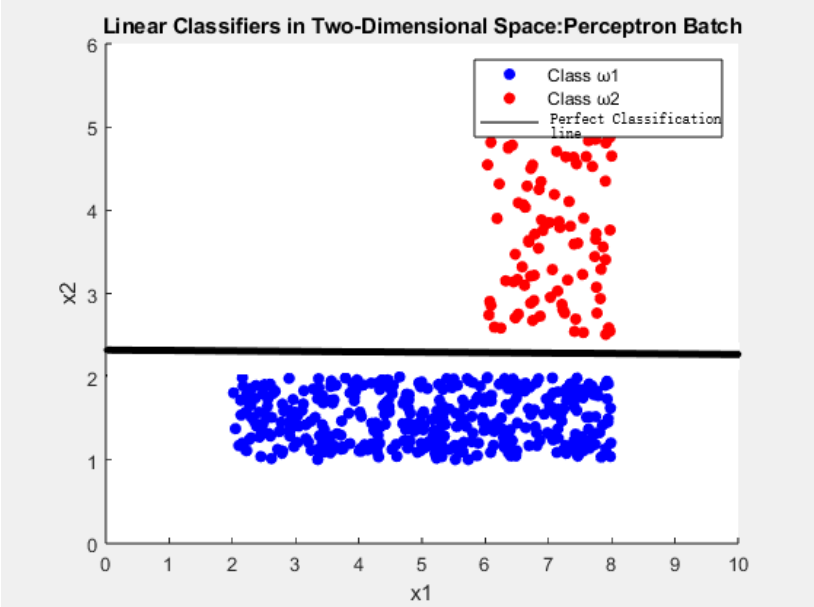
```
    w0 = [ 0 -1 3]'; % wo: y=3
    r = 0.001;
    j = 1;

    coeffx1 = w0(1);
    coeffx2 = w0(2);
    c = w0(3);

    misclassed = zeros(500,1); %Misclassed will return the position
of misclassified points
    iter=0;
    Error = 100;

    y1 = ones(400,1);
    y2 = ones(100,1);
    omega_1 = cat(2,omega_1,y1);
    omega_2 = cat(2,omega_2,y2);

    %Initialization
    while (Error ~=0 )
        j=1;
```

[17] Η απόδειξη του συγκεκριμένου βρίσκεται στο https://www.waxworksmath.com/Authors/N_Z/Theodoridis/WriteUp/Weatherwax_Theodoridis_Solutions.pdf page 37

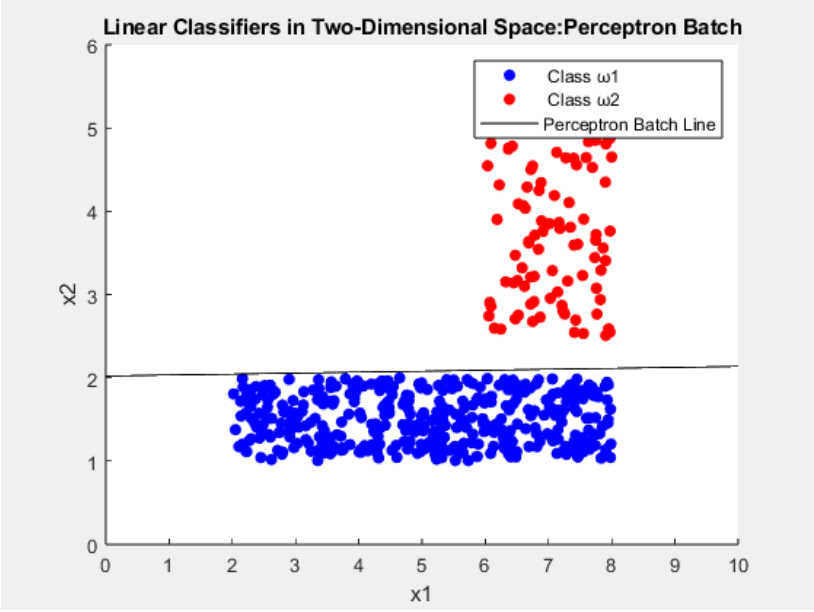
[18] Pattern Recognition 4th edition , S. Theodoridis , K. Koutrombas , Chapter 3 , page 105 , κυρίως στο Παράδειγμα 3.1

```
        for i = 1:400
            if (coeffx1*omega_1(i,1)+ coeffx2*omega_1(i,2) + c) < 0
                misclassed(j) = i;

                j=j+1;
            end
        end

        k=j;

        for i = 1:100
            if (coeffx1*omega_2(i,1)+ coeffx2*omega_2(i,2) + c) > 0
                misclassed(j) = i;

                j=j+1;
            end
        end

        if k==1
            w1 = w0 - r*(sum(omega_2(misclassed(k:j-1),:)',2));
        elseif k==j && k ~=1
            w1 = w0 + r*(sum(omega_1(misclassed(1:j-1),:)',2));
        else
            w1 = w0 + r*(sum(omega_1(misclassed(1:k),:)',2)) -
r*(sum(omega_2(misclassed(k:j-1),:)',2));
        end

        iter = iter+1;
        Error = (j-1)/500;
        fprintf('Perceptron classifier error for %d iteration:
%.4f%%\n',iter,Error)
        w0=w1;
        coeffx1 = w0(1);
        coeffx2 = w0(2);
        c = w0(3);
    end
    x1 = linspace(0, 20);
    line = -(coeffx1*x1+c)/coeffx2 ;

    figure;
    scatter(omega_1(:, 1), omega_1(:, 2), 'b', 'filled');
    hold on;
    scatter(omega_2(:, 1), omega_2(:, 2), 'r', 'filled');
    plot(x1,line,'k-');
    hold off;

    % Set plot title and labels
    title('Linear Classifiers in Two-Dimensional Space:Perceptron
Batch');
    xlabel('x1');
    ylabel('x2');

    % Set plot limits
    xlim([0, 10]);
    ylim([0, 6]);


    % Add legend
    legend(' Class ω1', ' Class ω2', 'Perceptron Batch Line');
```

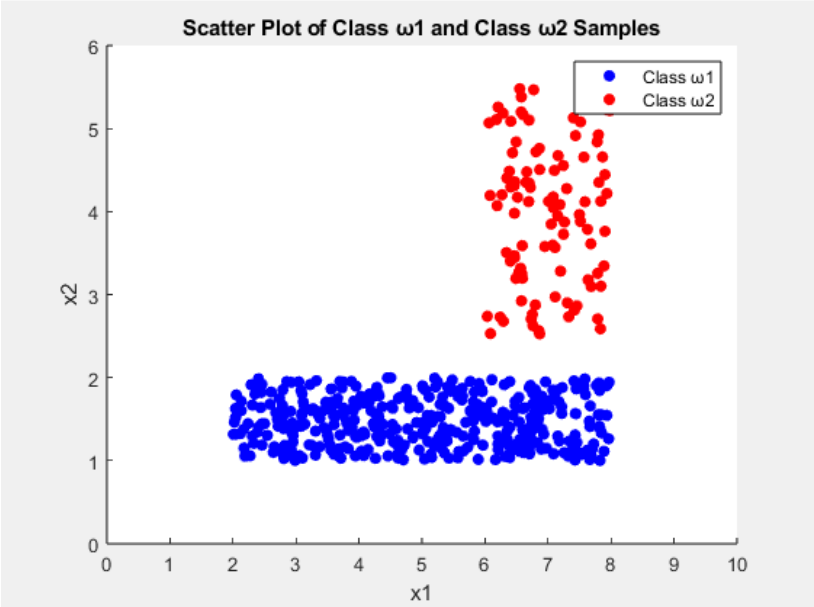
Σχήμα 4.2.1



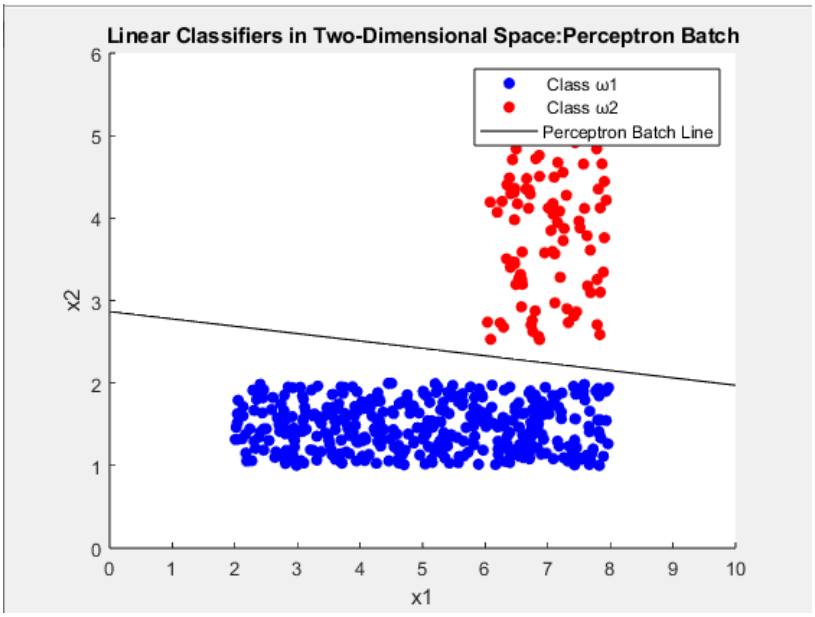
Σχήμα 4.2.2



Σχήμα 4.2.3



Σχήμα 4.2.4



Πίνακας 4.2.1

```
Perceptron classifier error for 1 iteration: 0.0360%
Perceptron classifier error for 2 iteration: 0.0080%
Perceptron classifier error for 3 iteration: 0.0000%
Perceptron Batch Classifier Error : 0.00%
```

ΠΙΝΑΚΑΣ P1

```
Euclidean Classifier Error : 8.00%
Mahalanobis Classifier Error : 1.00%
Bayesian Classifier Error : 0.00%
Euclidean Classifier for PCA Error : 21.00%
Euclidean Classifier for LDA Error : 19.20%
Least Squeares Classifier for LDA Error : 1.20%
Perceptron classifier error for 1 iteration: 0.0460%
Perceptron classifier error for 2 iteration: 0.1040%
Perceptron classifier error for 3 iteration: 0.1520%
Perceptron classifier error for 4 iteration: 0.6200%
Perceptron classifier error for 5 iteration: 0.2000%
Perceptron classifier error for 6 iteration: 0.2000%
Perceptron classifier error for 7 iteration: 0.0000%
Perceptron Batch Classifier Error : 0.00%
```