ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΗΛ 311 – Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων – 2024

Διδάσκων: Βασίλης Διγαλάκης **1η Σειρά Ασκήσεων** Επίλυση μέχρι: 19/05/2024

Αναφορά

Θέμα 1: Principal Component Analysis (PCA)

Σε αυτή την άσκηση, θα χρησιμοποιήσουμε την μέθοδο Principal Component Analysis (PCA) αρχικά πάνω σε ένα σύνολο δεδομένων που περιγράφουν καλοήθεις και κακοήθεις όγκους στο στήθος και στη συνέχεια θα χρησιμοποιήσουμε το PCA σε ένα μεγαλύτερο σύνολο δεδομένων 5000 εικόνων προσώπων.

Μέρος 1:

1. Αρχικά, διαβάζουμε το αρχείο `data/breast_cancer_data.csv` το οποίο περιέχει 569 δείγματα το καθένα εκ των οποίων περιγράφεται με ένα διάνυσμα 30 χαρακτηριστικών. Η τελευταία τιμή στο διάνυσμα είναι η κατηγοριοποίηση του δείγματος σε καλοήθες (0) και κακοήθες (1).

   Σε επίπεδο κώδικα, έχουμε:

```
Data=csvread('data/breast_cancer_data.csv');
NSamples = 100; %Get the first NSamples only for better
   visualization
X=Data(1:NSamples,1:end-1); % Get all features
Y=Data(1:NSamples,end);
```

Listing 1: Κώδικας για την φόρτωση του datset

2. Κατόπιν, επιλέγουμε διάφορα ζεύγη και τα απεικονίζουμε τα δείγματα στον 2D χώρο.Ακόμα, εισάγουμε ένα ακόμη μέτρο σύγκρισης, την απόσταση Mahalanobis, για να επιβεβαιώσουμε ότι τα οπτικά αποτελέσματά επιβεβαιώνονται. Σε έναν p-διάστατο χώρο, η απόσταση Malanobis ορίζεται ως:

$$D(x,y) = \sqrt{(x-y)^T \cdot S^{-1}(x-y)} \tag{1}$$

   όπου x,y είναι p-διάστατα διανύσματα στήλης που αναπαριστούν σημεία του χώρου και S^{-1} ο πίνακας συνδιασποράς των δεδομένων. Προτιμάμε την απόσταση Mahalanobis σε αντίθεση με την Ευκλείδεια απόσταση γιατί δεν απαιτείται η κανονικοποίησή στα δείγματά μας.Φυσικά, αν εφαρμοστεί κανονικοποίηση με μέση τιμή 0 και διασπορά 1, αποδεικνύεται εύκολα, πως η απόσταση Mahalanobis μετατρέπεται στην Ευκλείδεια απόσταση.

Σε επίπεδο κώδικα έχουμε:

```
% Define the feature pairs you want to plot
featurePairs = [3 4; 17 10; 9 12; 1 12; 1 2];  % Each row defines a pair of features to plot
numPairs = size(featurePairs, 1);

% Compute covariance matrix and mean vector
C = cov(X);
mu = mean(X);

for i = 1:numPairs
    % Create a new figure for each pair
    figure;

    % Plot the ith pair of features
    plot(X(Y==0, featurePairs(i, 1)), X(Y==0, featurePairs(i, 2)), 'bo', ...
         X(Y==1, featurePairs(i, 1)), X(Y==1, featurePairs(i, 2)), 'ro');

    % Compute Mahalanobis distance for each observation
    numObs = size(X, 1);  % Number of observations
    D = zeros(numObs, 1);  % Initialize array to store distances
    for j = 1:numObs
        x = X(j, featurePairs(i, :));  % Extract the features for the j-th observation
        D(j) = sqrt((x - mu(featurePairs(i, :))) * inv(C(featurePairs(i, :), featurePairs(i, :))) * (x - mu(featurePairs(i, :)))');
    end

    % Set plot properties
    axis square;
    xlabel(sprintf('Feature %d', featurePairs(i, 1)));
    ylabel(sprintf('Feature %d', featurePairs(i, 2)));
    title(sprintf('Feature %d vs Feature %d (Mahalanobis Distance)', featurePairs(i, 1), featurePairs(i, 2)));

    % Display Mahalanobis distance in the title
    distanceMean = mean(D);
```

```
33      titleText = sprintf('Feature %d vs Feature %d (Mean
            Mahalanobis Distance: %.2f)', featurePairs(i, 1),
            featurePairs(i, 2), distanceMean);
34      title(titleText);
35  end
36
37  pause
```

Listing 2: Κώδικας για απεικόνιση διαφορετικών δειγμάτων στον 2D χώρο

Λαμβάνουμε επίσης, τις εξής εικόνες:

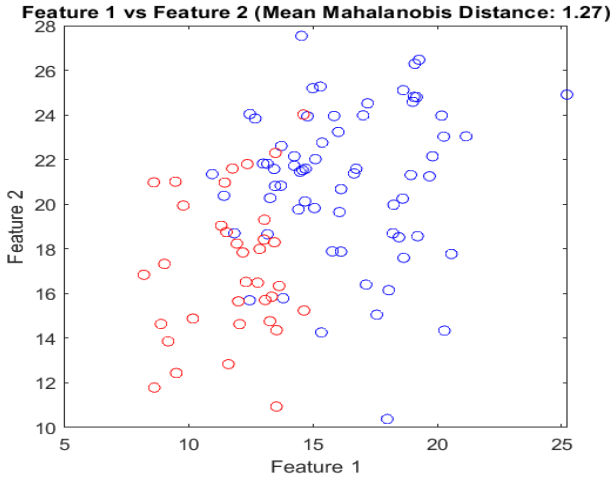

Figure 1: Απεικόνιση Feature 1 vs Feature 2

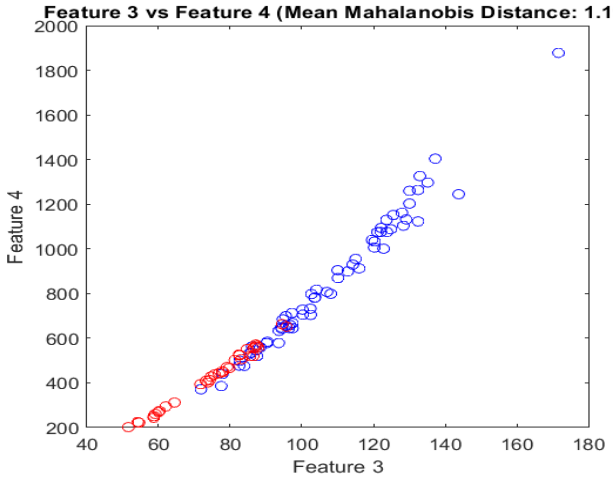

Figure 2: Απεικόνιση Feature 3 vs Feature 4

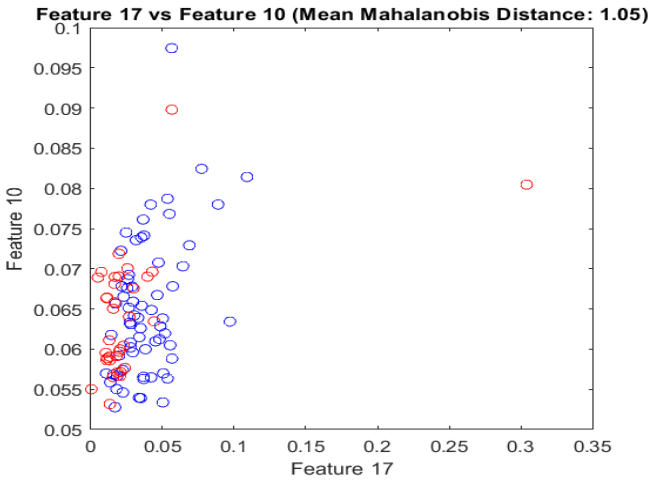

Figure 3: Απεικόνιση Feature 17 vs Feature 10

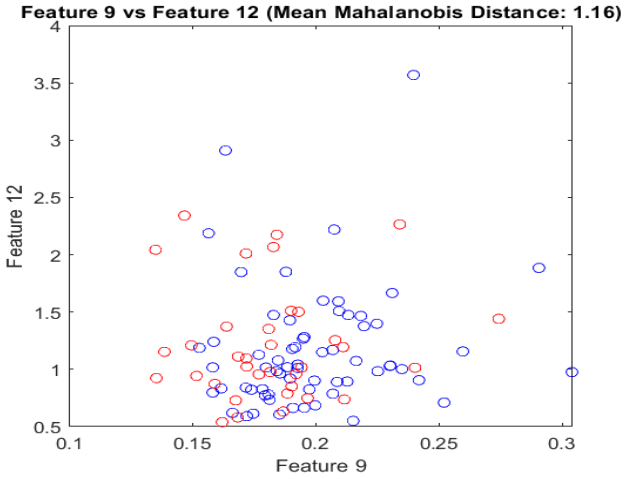

Figure 4: Απεικόνιση Feature 9 vs Feature 12

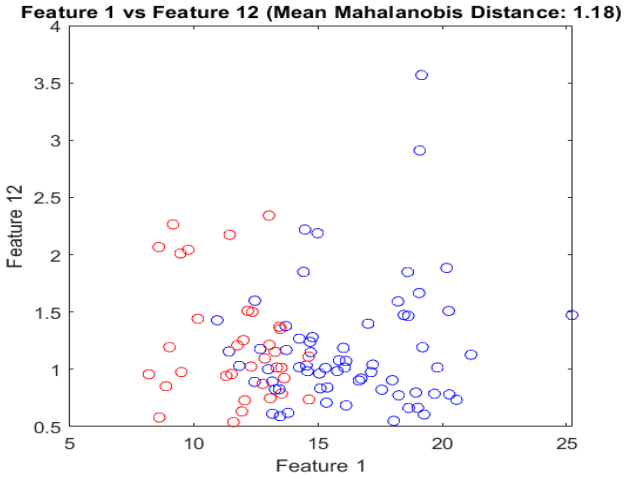

Figure 5: Απεικόνιση Feature 1 vs Feature 12

Συνεχίζουμε εφαρμόζοντας standardization με μέση τιμή μηδέν και διασπορά 1 στα αρχικά μας δείγματα.Αυτό υλοποιείται μέσω της συνάρτησης featureNormalize ο κώδικας της οποίας είναι:

```
function [X_norm, mu, sigma] = featureNormalize(X)
% Get the number of samples and features
[nSamples, nFeat] = size(X);

% Initialize output variables --- Preallocation
X_norm = zeros(nSamples, nFeat);
mu = zeros(1, nFeat);
sigma = zeros(1, nFeat);

% Normalize each feature
for j = 1:nFeat
    mu(j) = mean(X(:, j));
    sigma(j) = std(X(:, j));
    X_norm(:, j) = (X(:, j) - mu(j)) / sigma(j);
end

end
```

Listing 3: Κώδικας για το standardization

Και λαμβάνουμε την εξής εικόνα:

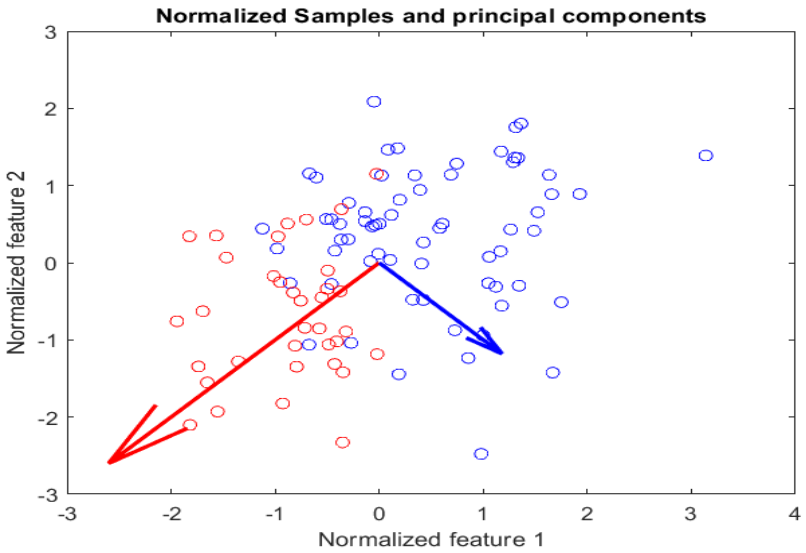

Figure 6: Κανονικοποιημένα δείγματα και Principal Components

Στην ίδια εικόνα, σχεδιάζουμε και τις κύριες συνιστώσες μέσω της συνάρτησης myPCA ως εξής:

```
function [ eigenval, eigenvec, order] = myPCA(X)

% Useful values
[m, n] = size(X);
eigenvec = zeros(m);
eigenval = zeros(n);

% Make sure each feature from the data is zero mean
X_centered = X - mean(X);

% ====================== YOUR CODE HERE ======================

% Compute the covariance matrix
Sigma = (1/(m)) * (X_centered' * X_centered);


% Compute eigenvalues and eigenvectors
[V, D] = eig(Sigma);

% Extract the diagonal of D as a vector
eigenval = diag(D);

% Sort the eigenvalues in descending order and get the order
[eigenval,order]=sort(eigenval,1,'descend'); %Sort them

% Reorder the eigenvectors according to the sorted eigenvalues
eigenvec=V(:,order); %Corresponding eigenvectors
```

Listing 4: Κώδικας για την δημιουργία των principal components

Και λαμβάνουμε την εξής εικόνα:

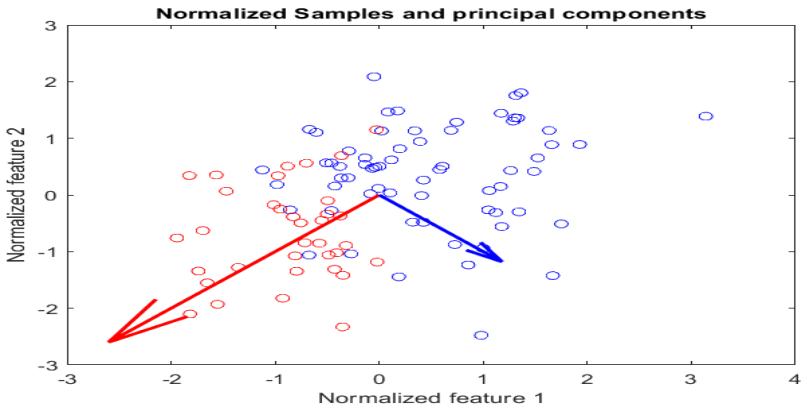

Figure 7: Κανονικοποιημένα δείγματα και Principal Components

Παρατηρούμε ότι μετά την εφαρμογή του standardization τα δείγματά μας είναι κεντραρισμένα με βάση την μέση τιμή.Ακόμα, παρατηρούμε ότι οι κύριες συνιστώσες του PCA είναι κάθετες μεταξύ τους.

Επιπλέον, υπολογίζουμε την συνεισφορά που έχει κάθε συνιστώσα(explained variance) στην συνολική διακύμανση. Ορίζουμε ως explained variance:

$$Var(PC_i) = \frac{\lambda_i}{\sum_{j=1}^{L} \lambda_j}$$

Σε επίπεδο MATLAB έχουμε:

```
ExplainedVar = eigvals/sum(eigvals);
fprintf(' Explained Variance(1st PC = %f) (2nd PC = %f)\n',
   ExplainedVar(1), ExplainedVar(2));
```

Listing 5: Κώδικας για τον υπολογισμό του explained variance

Τα αριθμητικά δεδομένα που λαμβάνουμε είναι τα εξής:

Explained Variances:
1st Principal Component: 0.687891
2nd Principal Component: 0.312109

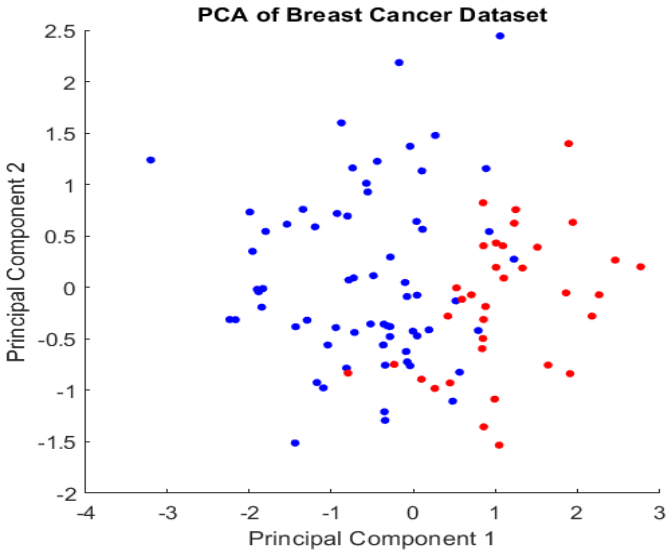

Figure 8: Προβολή των διανυσμάτων πάνω στις κύριες συνιστώσες

Προκειμένου να μειώσουμε την διάσταση των διανυσμάτων μας, συμπληρώνουμε κατάλληλα την συνάρτηση projectData:

```
function Z = projectData(X, U, K)
Z = zeros(size(X, 1), K);

```

```
% Instructions: Compute the projection of the data using only
    the top K
%                eigenvectors
%

Z = X * U(:, 1:K);


end
```

Listing 6: Κώδικας για dimensional reduction

Προβάλλοντας τα δεδομένα σε χώρο μικρότερων διαστάσεων έχουμε ότι:

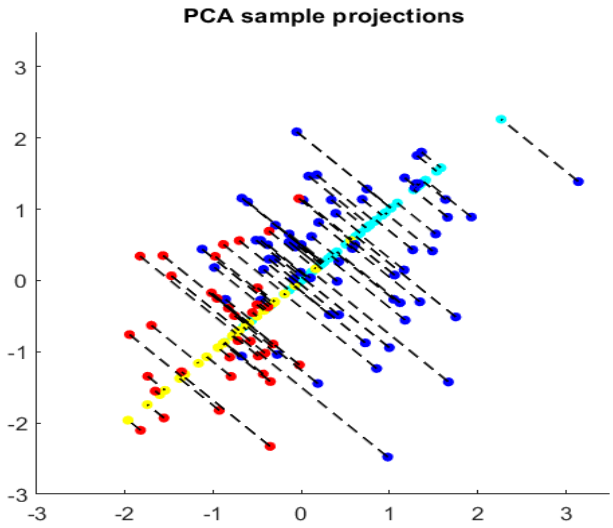

Figure 9: Ανακτημένα δείγματα και προβολή στον χώρο των υψηλών διαστάσεων

Επαναλαμβάνουμε την παραπάνω διαδικασία χρησιμοποιώντας αυτήν την φορά όλο το διάνυσμα των χαρακτηριστικών(30 χαρακτηριστικά) και λαμβάνουμε την εξής εικόνα:

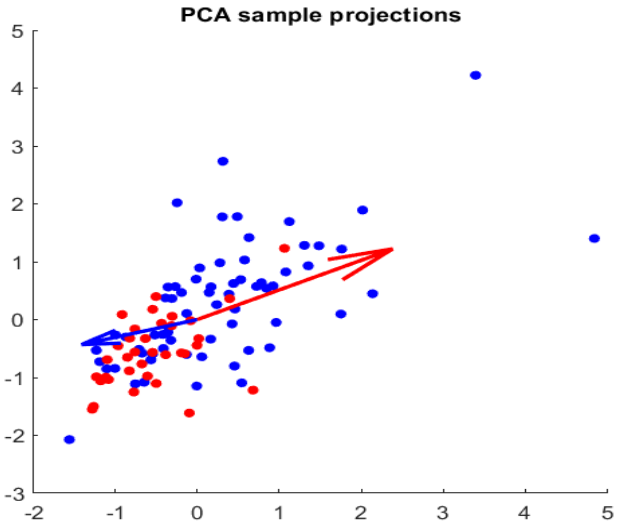

Figure 10: Προβολή δειγμάτων στις κύριες συνιστώσες

Σε αυτήν την εικόνα, παρατηρούμε ότι οι κύριες συνιστώσες δεν φαίνεται να είναι ίσες ωστόσο, η ανάλυσή μας σε αυτήν την περίπτωση έχουμε αξιοποιήσει και τα 30 χαρακτηριστικά ενώ προβάλουμε τα δείγματα σε χώρο 2D.Συνεπώς, είναι αναμενόμενο να παρουσιάζονται και παραμορφώσεις αλλά θεωρητικά είναι κάθετα.

Μέρος 2:

Σε αυτό το μέρος, επαναλαμβάνουμε την παραπάνω διαδικασία χρησιμοποιώντας τα δεδομένα από 5000 εικόνων προσώπου που δίνονται. Ξεκινάμε, σχεδιάζοντας τα πρώτα 100 πρόσωπα από το dataset μας μέσω της συνάρτησης display και λαμβάνουμε την εξής εικόνα:

Figure 11: Σχεδιασμός των πρώτων 100 προσώπων

Στη συνέχεια, εφαρμόζουμε standardization και ακολούθως υπολογίζουμε τις κύριες συνιστώσες

εφαρμόζοντας τη συνάρτηση myPCA.Ολοκληρώνουμε, σχεδιάζοντας μια νέα εικόνα με τις πρώτες 36 κύριες συνιστώσες και το αποτέλεσμα είναι το εξής:

Figure 12: Σχεδιασμός των προσώπων με τις πρώτες 36 κύριες συνιστώσες

Παρατηρούμε ότι τα πρόσωπα εμφανίζουν υψηλή αλλοίωση των χαρακτηριστικών τους ενώ το χρώμα τείνει να κυμαίνεται γύρω από το σκούρο γκρι.

Προχωρώντας, μειώνουμε την διάσταση των δειγμάτων μας χρησιμοποιώντας τις 100 πρώτες κύριες συνιστώσες και κατόπιν σχεδιάζουμε τα δείγματα μειωμένης διάστασης αφού φυσικά τα προβάλουμε στον αρχικό χώρο.Τελικά, λαμβάνουμε την εξής εικόνα:

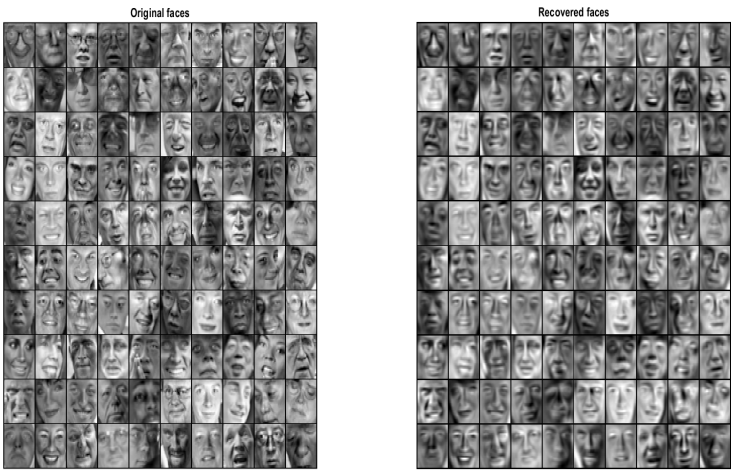

Figure 13: Σύγκριση των αρχικών μας δειγμάτων με τα δείγματα μειωμένης διάστασης

Θέμα 2: Σχεδίαση ταξινομητή LDA(Linear Discriminant Analysis)

Σε αυτό το μέρος θα σχεδιάσουμε έναν ταξινομητή LDA και θα υπολογίσουμε το διάνυσμα προβολής. Ξεκινώντας, εκφράζουμε το κριτηρίο Fisher συναρτήσει των πινάκων S_w και S_b :

$$J(w) = \frac{w^T S_B w}{w^T S_w w}$$

Για να βρούμε το μέγιστο του $J(w)$, παραγωγίζουμε και εξισώνουμε με το μηδέν:

$$\begin{aligned}\frac{d}{dw}J(w) = \frac{d}{dw}\left(\frac{w^T S_B w}{w^T S_w w}\right) = 0 \\ \Rightarrow (w^T S_w w)\frac{d}{dw}(w^T S_B w) - (w^T S_B w)\frac{d}{dw}(w^T S_w w) = 0 \\ \Rightarrow (w^T S_w w)2S_B w - (w^T S_B w)2S_B w = 0\end{aligned}$$

Διαιρούμε με το $2w^T S_w w$:

$$\begin{aligned}\Rightarrow S_B w - J(w)S_w w = 0 \\ \Rightarrow S_B^{-1} S_B w - J(w)w = 0\end{aligned}$$

Τώρα επιλύουμε το γενικευμένο πρόβλημα ιδιοτιμών

$$S_w^{-1} S_B w = \lambda w$$

όπου $\lambda = J(w)$ και έχουμε:

$$w^* = \text{argmax} J(w) = \text{argmax}\left(\frac{w^T S_B w}{w^T S_w w}\right) = S_w^{-1}(\mu_A - \mu_B)$$

Υπολογίζουμε τώρα τον πίνακα S_w :

$$\begin{aligned}S_w &= P(\omega_A)\Sigma_A + P(\omega_B)\Sigma_B \\ &= 0.4\begin{bmatrix}3 & 1\\ 1 & 2\end{bmatrix} + 0.6\begin{bmatrix}2 & 1\\ 1 & 3\end{bmatrix} \\ &= \begin{bmatrix}1.2 & 0.4\\ 0.4 & 0.8\end{bmatrix} + \begin{bmatrix}1.2 & 0.6\\ 0.6 & 1.8\end{bmatrix} \\ &= \begin{bmatrix}2.4 & 1\\ 1 & 2.6\end{bmatrix}\end{aligned}$$

Ακόμα, έχουμε:

$$\mu_A - \mu_B = \begin{bmatrix}0\\ 2\end{bmatrix} - \begin{bmatrix}2\\ 0\end{bmatrix} = \begin{bmatrix}-2\\ 2\end{bmatrix}$$

Ο αντίστροφος του πίνακα ${S_w}^{-1}$ δίνεται από τον τύπο:

$$\begin{bmatrix}a & b\\ c & d\end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix}d & -b\\ -c & a\end{bmatrix}$$

Άρα έχουμε:

$${S_w}^{-1} = \frac{1}{2.4\cdot 2.6 - 1\cdot 1}\begin{bmatrix}2.6 & -1\\ -1 & 2.4\end{bmatrix} = \begin{bmatrix}0.496 & -0.19\\ -0.19 & 0.458\end{bmatrix}$$

Τελικά, έχουμε ότι:

$$w = \begin{bmatrix} 0.496 & -0.19 \\ -0.19 & 0.458 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ 2 \end{bmatrix} = \begin{bmatrix} -1.372 \\ 1.296 \end{bmatrix}$$

Μέρος 2:

Για να υπολογίσουμε την προβολή των διανυσμάτων $x_A = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ και $x_B = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ θα χρησιμοποιήσουμε τον τύπο:

$$y = w^T x$$

Συνεπώς, έχουμε:

$$y_1 = w^T x_A = 2.516 \text{ and } y_2 = w^T x_B = -2.82$$

Θέμα 3: Linear Discriminant Analysis(LDA) vs PCA

Μέρος 1:

Σε αυτήν την άσκηση, θα ασχοληθούμε με την εφαρμογή του Linear Discriminant Analysis προκειμένου να μειώσουμε τη διάσταση ενός feature vector και να συγκρίνουμε τα αποτελέσματά σε σχέση με την μέθοδο PCA.

Ξεκινάμε λοιπόν, φορτώνοντας τα δεδομένα μας, και πραγματοποιούμε standardization με μέση τιμή 0 και διασπορά 1 και λαμβάνουμε την εξής εικόνα:

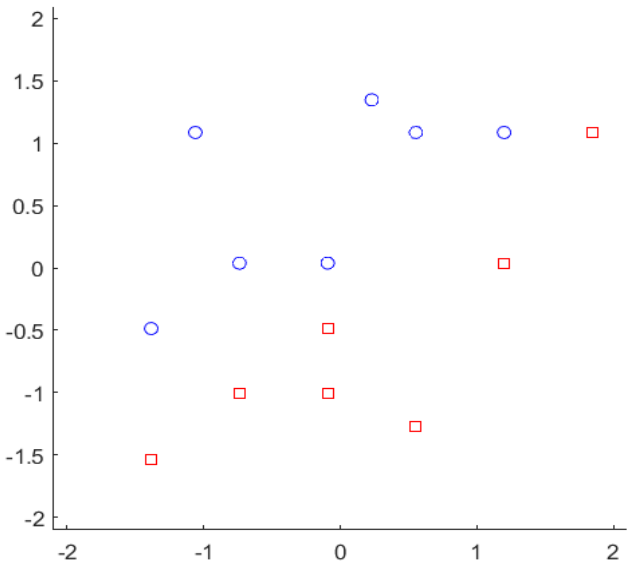

Figure 14: Σύγκριση των αρχικών μας δειγμάτων με τα δείγματα μειωμένης διάστασης

Κατόπιν, υλοποιούμε κατάλληλα την συνάρτηση fisherLinearDisciminant ενώ μειώνουμε την διάσταση στα δείγματά μας:

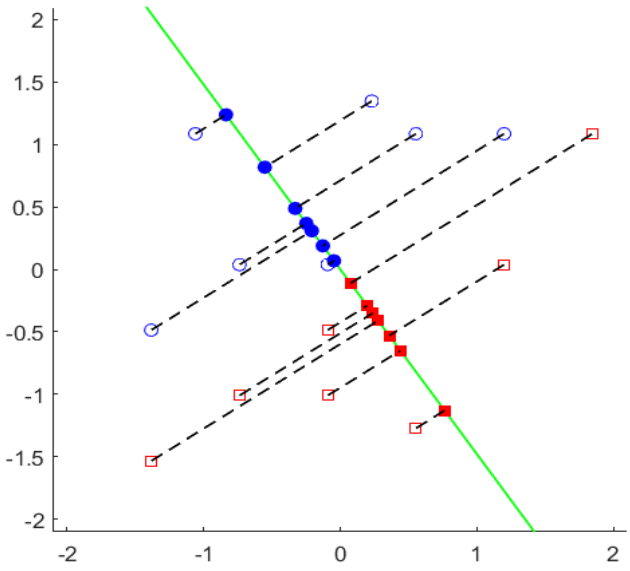

Figure 15: Σύγκριση των αρχικών μας δειγμάτων με τα δείγματα μειωμένης διάστασης

Αντίστοιχη διαδικασία ακολουθούμε και με την μέθοδο PCA και η εικόνα που λαμβάνουμε είναι η εξής:

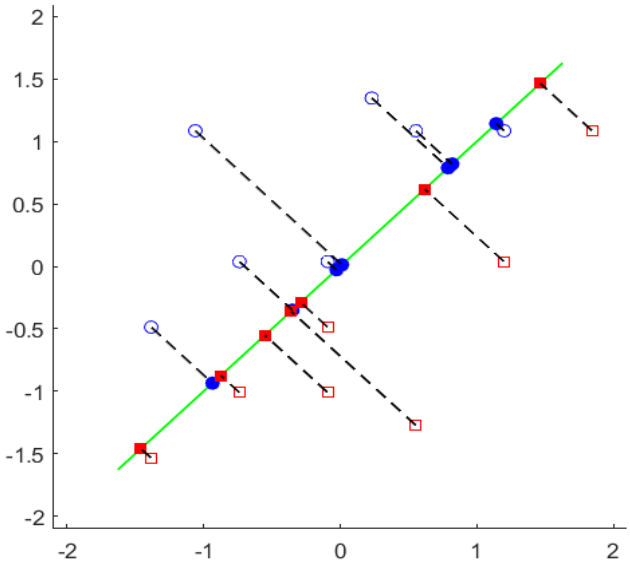

Figure 16: Σύγκριση των αρχικών μας δειγμάτων με τα δείγματα μειωμένης διάστασης

Μέρος 2:

Σε αυτό το μέρος εφαρμόζουμε τον αλγόριθμο LDA σε ένα πολύ δημοφιλές dataset στο χώρο της μηχανικής μάθησης τη βάση δεδομένων Iris, που περιέχει 3 διαφορετικά είδη της οικογένειας του λουλουδιού του είδους Iris.Στο πλαίσιο της αναλυσής μας, φορτώνουμε τα δειγματά μας και

πραγματοποιούμε standardization με μέση τιμή 0 και διασπορά 1 και σχεδιάζουμε τα 2 πρώτα χαρακτηριστικά σε ένα 2D χώρο και λαμβάνουμε την εξής εικόνα:

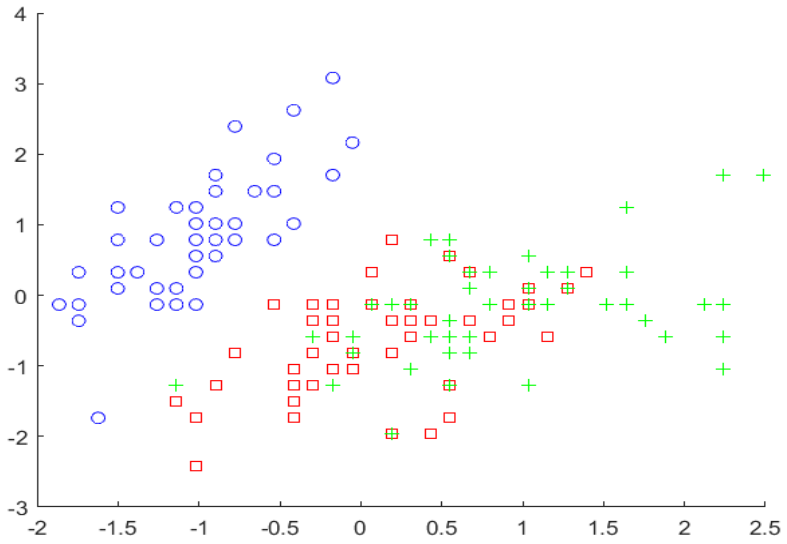

Figure 17: Σχεδίαση των 2 πρώτων χαρακτηριστικών από το σύνολο Iris.

Κατόπιν, υλοποιούμε την συνάρτηση LDA, που υλοποιεί τον ζητούμενο αλγόριθμο και ο κώδικας είναι ο εξής:

```
1  function A = myLDA(Samples, Labels, NewDim)
2  % Input:
3  %   Samples: The Data Samples
4  %   Labels: The labels that correspond to the Samples
5  %   NewDim: The New Dimension of the Feature Vector after applying
       LDA
6
7  %A=zeros(NumFeatures,NewDim);
8
9  [NumSamples, NumFeatures] = size(Samples);
10 NumLabels = length(Labels);
11 if(NumSamples ~= NumLabels) then
12     fprintf('\nNumber of Samples are not the same with the Number
          of Labels.\n\n');
13     exit
14 end
15     Classes = unique(Labels);
16     NumClasses = length(Classes);  %The number of classes
17
18 % Initialize matrices
```

```
Sw = zeros(NumFeatures, NumFeatures); % Within-class scatter matrix
m0 = mean(Samples, 1); % Global mean
Sb = zeros(NumFeatures, NumFeatures); % Between-class scatter
   matrix

%For each class i
%Find the necessary statistics
    for i = 1:NumClasses
        % Class-specific samples and statistics
        classSamples = Samples(Labels == Classes(i), :);
        mu(i, :) = mean(classSamples, 1); % %Calculate the Class
           Mean
        P(i) = size(classSamples, 1) / NumSamples; %Calculate the
           Class Prior Probability

        % Within-class scatter -- Calculate the Within Class
           Scatter Matrix
        classScatter = classSamples - mu(i, :); % Deviation from
           mean
        Sw = Sw + (classScatter' * classScatter); % Sum of squares

        % Between-class scatter -- %Calculate the Between Class
           Scatter Matrix
        meanDiff = (mu(i, :) - m0)';
        Sb = Sb + P(i) * (meanDiff * meanDiff');
    end

%Eigen matrix EigMat=inv(Sw)*Sb
EigMat = inv(Sw)*Sb;

%Perform Eigendecomposition
[eigenvectors, eigenvalues] = eig(EigMat);

% Extract eigenvalues as a vector
eigenvalues = diag(eigenvalues);

% Sort eigenvalues (and vectors) in descending order
[sortedEigenvalues, sortOrder] = sort(eigenvalues, 'descend');
sortedEigenvectors = eigenvectors(:, sortOrder);


%% You need to return the following variable correctly.
```

```
% Select the NewDim eigenvectors corresponding to the top NewDim
    eigenvalues
% Assuming they are NewDim <= NumClasses - 1
A = sortedEigenvectors(:, 1:NewDim);  % Return the LDA projection
    vectors
```

Listing 7: Κώδικας για την υλοποίηση του αλγόριθμου LDA

Τέλος, εφαρμόζουμε τα διανύσματα προβολής που υπολογίσαμε με την LDA πάνω στα αρχικά μας δείγματα με τη συνάρτηση projectDataLDA για να μειώσουμε την διάσταση τους σε 2 και λαμβάνουμε την εξής εικόνα:

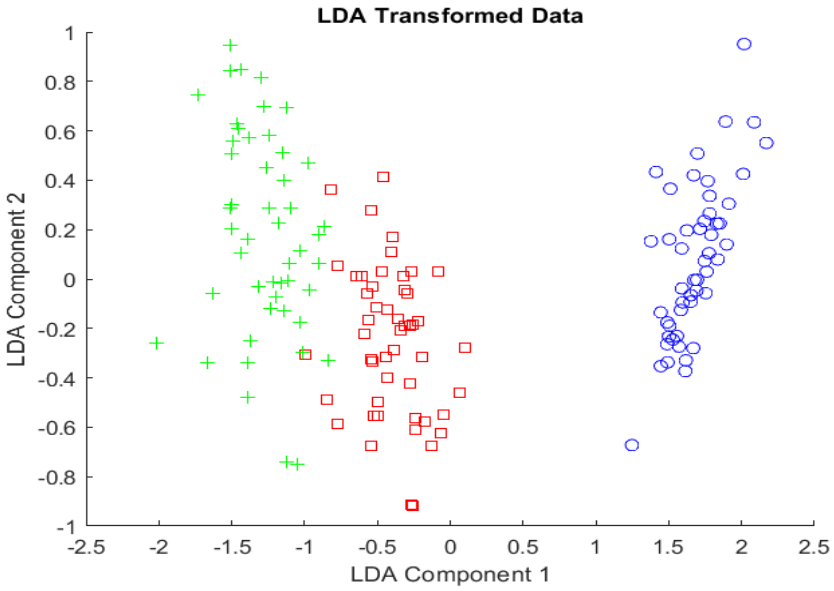

Figure 18: Προβολή των δειγμάτω μέσω της συνάρτησης projectDataLDA.

Θέμα 4: Bayes

Για δύο κλάσεις που ακολουθούν πολυδιάστατη κανονική κατανομή με μέσες τιμές μ_1, μ_2 και πίνακες συνδιασποράς Σ_1, Σ_2 το σύνορο απόφασης δίνεται από την εξίσωση:

$$P(\omega_1|x) = P(\omega_2|x)$$

Λόγω του κανόνα του Bayes όμως η παραπάνω εξίσωση μετασχηματίζεται ως εξής:

$$P(x|\omega_1)P(\omega_1) = P(x|\omega_2)P(\omega_2)$$

Λογαριθμούμε την παραπάνω εξίσωση και έχουμε:

$$\log(P(x|\omega_1)P(\omega_1)) = \log(P(x|\omega_2)P(\omega_2)) \iff$$
$$\log(P(x|\omega_1)) + \log(P(\omega_1)) = \log(P(x|\omega_2)) + \log(P(\omega_2))$$

Φυσικά, για την πιθανότητα $P(x|\omega_1)$ και $P(x|\omega_2)$ ακολουθούν πολυδιάστατη κανονική κατανομή συνεπώς η Σ.Π.Π είναι:

$$f_X(X) = \frac{1}{2\pi^{\frac{n}{2}}|\Sigma_X|^{\frac{1}{2}}} \cdot exp\left(-\frac{1}{2}(X-\mu_X)^T\Sigma_X^{-1}(X-\mu_X)\right)$$

Συνδυάζοντας τις παραπάνω σχέσεις έχουμε:

$$
\begin{aligned}
&-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)+\log(P(\omega_1))-\frac{1}{2}\log(|\Sigma_1|)\\
&=-\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)+\log(P(\omega_2))-\frac{1}{2}\log(|\Sigma_2|) \iff\\
&(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)-(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\\
&=\log\frac{|\Sigma_2|}{|\Sigma_1|}+2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right)
\end{aligned}
$$

Τώρα θα προσπαθήσουμε να φέρουμε την εξίσωση σε μια πιο γνωστή μορφή.
Αναπτύσσουμε τον όρο: $(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)$ και έχουμε:

$$(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)=x^T\Sigma_1^{-1}x-2\mu_1^T\Sigma_1^{-1}x+\mu_1^T\Sigma_1^{-1}\mu_1$$

Όμοια και για τους άλλους όρους έχουμε:

$$(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)=x^T\Sigma_2^{-1}x-2\mu_2^T\Sigma_1^{-1}x+\mu_2^T\Sigma_2^{-1}\mu_2$$

Βάση των παραπάνω εξισώσεων η αρχική μας εξίσωση μπορεί να γραφτεί και ως εξής:

$$
\begin{aligned}
&x^T\Sigma_1^{-1}x-2\mu_1^T\Sigma_1^{-1}x+\mu_1^T\Sigma_1^{-1}\mu_1-(x^T\Sigma_2^{-1}x-2\mu_2^T\Sigma_1^{-1}x+\mu_2^T\Sigma_2^{-1}\mu_2)\\
&=\log\frac{|\Sigma_2|}{|\Sigma_1|}+2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right) \iff\\
&x^T(\Sigma_1^{-1}-\Sigma_2^{-1})x+2(\mu_2^T\Sigma_2^{-1}-\mu_1^T\Sigma_1^{-1})x+(\mu_1^T\Sigma_1^{-1}\mu_1-\mu_2^T\Sigma_2^{-1}\mu_2)\\
&=\log\frac{|\Sigma_2|}{|\Sigma_1|}+2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right)
\end{aligned}
$$

Και θέτουμε:

$$A=\Sigma_1^{-1}-\Sigma_2^{-1} \tag{2}$$

$$B=2(\mu_2^T\Sigma_2^{-1}-\mu_1^T\Sigma_1^{-1}) \tag{3}$$

$$C=\mu_1^T\Sigma_1^{-1}\mu_1-\mu_2^T\Sigma_2^{-1}\mu_2-2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right)-\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) \tag{4}$$

Τελικά, η εξίσωσή συνόρου απόφασης είναι:

$$x^TAx+B^Tx+C=0 \tag{5}$$

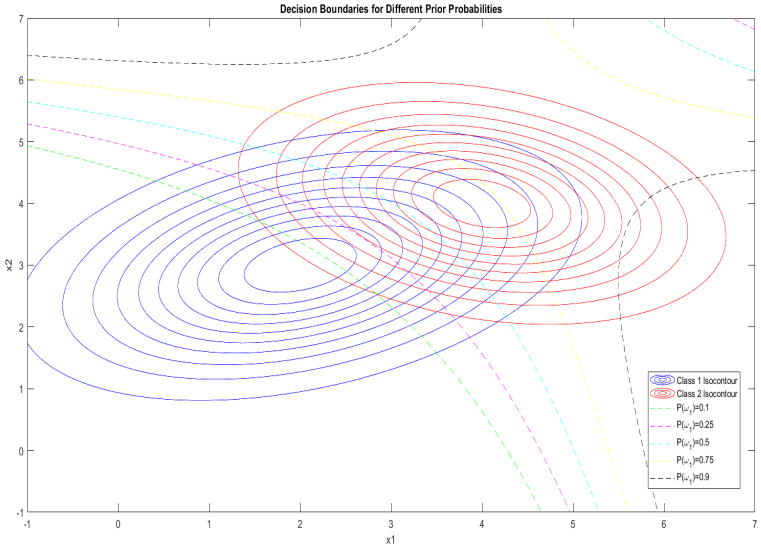

Figure 19: Ισοϋψείς καμπύλες και τα σύνορα απόφασης για διάφορες τιμές

Στην εικόνα παρατηρούμε ότι τα όρια απόφασης έχουν την μορφή καμπυλών γεγονός που οφείλεται στο ότι οι πίνακες συνδιασποράς δεν είναι οι ίδιοι μεταξύ τους.

Στην περίπτωση που οι πίνακες συνδιασποράς είναι τέτοιο ώστε:

$$\Sigma = \Sigma_1 = \Sigma_2 = \begin{bmatrix} 1.2 & 0.4 \\ 0.4 & 1.2 \end{bmatrix}$$

τότε ανακαλώντας την γενικευμένη εξίσωση που υπολογίστηκε προηγουμένως και έχουμε ότι:

$$\begin{aligned} &(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2) \\ &= \log\frac{|\Sigma_2|}{|\Sigma_1|} + 2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right) \iff \\ &(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma^{-1}(x-\mu_2) \\ &= \log\frac{|\Sigma|}{|\Sigma|} + 2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right) \iff \\ &(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma^{-1}(x-\mu_2) \\ &= 2\log\left(\frac{P(\omega_2)}{P(\omega_1)}\right) \end{aligned}$$

Θέτοντας $y_1 = (x-\mu_1)$ και $y_2 = (x-\mu_2)$ έχουμε ότι:

$$y_1^T\Sigma^{-1}(y_1) - (y_2)^T\Sigma^{-1}y_2$$

που η τελευταία εξίσωση είναι η εξίσωση συνόρου απόφασης.

Θέμα 5: Εξαγωγή χαρακτηριστικών και Bayes Classification.

Σε αυτήν την άσκηση καλούμαστε να υλοποιήσουμε έναν Bayes Classifier ο οποίος θα ταξινομεί αρχικά τα ψηφία 1 και 2 χρησιμοποιώντας διάφορα χαρακτηριστικά.Το πρώτο χαρακτηριστικό για την ταξινομηση μας είναι το aspect ratio το οποίο υλοποιείται και ως εξής σε python:

aspect_ratio

```python
def aspect_ratio(image):
  """Calculates the aspect ratio of the bounding box around the foreground
  ↪ pixels."""
  try:
    # Extract image data and reshape it (assuming data is in a column
    ↪ named 'image')
    img = image.values.reshape(28, 28)

    # Find non-zero foreground pixels
    nonzero_pixels = np.nonzero(img)

    # Check if there are any foreground pixels
    if nonzero_pixels[0].size == 0:
      return np.nan  # Return NaN if no foreground pixels found

    # Get minimum and maximum coordinates of foreground pixels
    min_row = np.min(nonzero_pixels[0])
    max_row = np.max(nonzero_pixels[0])
    min_col = np.min(nonzero_pixels[1])
    max_col = np.max(nonzero_pixels[1])

    # Calculate bounding box dimensions
    height = max_row - min_row + 1
    width = max_col - min_col + 1

    # Calculate aspect ratio
    aspect_ratio = width / height if height > 0 else np.nan

    return aspect_ratio
```

Επιβεβαιώνουμε ότι οι υπολογισμοί μας είναι σωστοί σχεδίαζοντας δύο τυχαία δείγματα από τα ψηφία μας μέσα σε ένα παραλληλόγραμμο και κατόπιν, εφαρμόζουμε κανονικοποίηση min-max την οποία υλοποιούμε σε python ως εξής:

min_max_scaling

```python
def min_max_scaling(X, min_val=-1, max_val=1):
  """Scales features to a range between min_val and max_val."""
  X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
  X_scaled = X_std * (max_val - min_val) + min_val
```

```
  return X_scaled
```

Συνεχίζουμε, υλοποιώντας τις εκ των προτέρων πιθανότητες $P(C_1)$ καθώς και $P(C_2)$ μέσω της συνάρτησης train η οποία υλοποιείται ως εξής:

train

```
  def train(self, X, y):
"""
Train the classifier under the assumption of Gaussian distributions:
  calculate priors and Gaussian distribution parameters for each class.

Args:
X (pd.DataFrame): DataFrame with features.
y (pd.Series): Series with target class labels.
"""
self.classes_ = np.unique(y)
for class_label in self.classes_:
  # Filter data by class
  X_class = X[y == class_label]

  # Calculate prior probability for the class
  self.class_priors[class_label] = len(X_class) / len(X)
```

Αν υποθέσουμε ακόμα ότι η κατανομή του aspect ratio χαρακτηριστικού σε κάθε κλάση ακολουθεί κανονική κατανομή με μέση τιμή $\overline{\mu} = \frac{1}{N}\Sigma_{i=1}^{N} x_i$ και τυπική απόκλιση $\sigma = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(x_i - \overline{\mu})^2}$ η συνάρτηση train συμπληρώνεται ως εξής:

train

```
# Calculate mean and covariance for the class
# Adding a small value to the covariance for numerical stability
self.class_stats[class_label] = {
  'mean': X_class.mean(),
  'cov': X_class.cov() + 1e-3 * np.eye(X_class.shape[1])
}
```

Κατόπιν, υπολογίζουμε την πρόβλεψη(predict) του ταξινομητή μας και έχουμε:

predict

```
def predict(self, X):
"""
Predict class labels for each test sample in X.
```

```
    Args:
    X (pd.DataFrame): DataFrame with features to predict.

    Returns:
    np.array: Predicted class labels.
    """
    predictions = []
    for index, sample in X.iterrows():
        log_probs = {}
        for class_label, stats in self.class_stats.items():
            prior = np.log(self.class_priors[class_label])
            mean = stats['mean']
            cov = stats['cov']
            diff = sample.values - mean
            likelihood = multivariate_normal.logpdf(diff, mean=mean,
             ↪  cov=cov)
            log_probs[class_label] = prior + likelihood
        prediction = max(log_probs, key=log_probs.get)
        predictions.append(prediction)
    return np.array(predictions)
```

Εκτιμούμε το αποτέλεσμα και η ακρίβεια του ταξινομητή μας είναι στο 0.9587 συνεπώς ακόμα και με 1 feature λαμβάνουμε αρκετά ικανοποιητικά αποτελέσματά. Ένα ακόμη χαρακτηριστικό που μπορούμε να εισάγουμε είναι ο αριθμός των μη μαύρων pixels που σχηματίζουν τον αριθμό σε κάθε εικόνα (foreground_pixels).Υλοποιώντας την συνάρτηση στην python θα έχουμε:

foreground_pixels

```
def foreground_pixels(image):
    """
    Calculate the pixel density of the image, defined as the
    count of non-zero pixels

    Args:
    image (np.array): A 1D numpy array representing the image.

    Returns:
    int: The pixel density of the image.
    """
    try:
```

```
        # Extract image data and reshape it (assuming data is in a column
        ↪ named 'image')
        img = image.values.reshape(28, 28)

        # Find non-zero foreground pixels
        nonzero_pixels = np.count_nonzero(img)
        if nonzero_pixels == 0:
            print(f"Warning: Couldn't find nonzero pixels on
            ↪ {image.name}")
            return np.nan  # Return NaN if no foreground pixels found
    except (KeyError, ValueError) as e:
        print(f"Error processing image in row  {image.name}: {e}")
        return np.nan  # Return NaN for rows with errors

    return nonzero_pixels
```

Σε αυτήν την περίπτωση, με την προσθήκη ενός ακόμη χαρακτηριστικού αυξάνεται την ακρίβεια του ταξινομητή αλλά όχι σε δραματικό βαθμό(accurancy = 0.9711).Εισάγουμε ένα τελευταίο feature, το κεντροειδές το οποίο ορίζεται ως συντεταγμένες στο επίπεδο της εικόνας.Υλοποιούμε το ζητούμενο σε python και έχουμε :

calculate_centroid

```
  def calculate_centroid(image):
"""
Calculate the normalized centroid (center of mass) of the image.

Returns:
tuple: The (x, y) coordinates of the centroid normalized by image
↪ dimensions.
"""
# Extract image data and reshape it (assuming data is in a column named
↪ 'image')
img = image.values.reshape(28, 28)
rows, cols = img.shape

# Calculate total mass (sum of all pixel values)
total_mass = img.sum()

# Calculate x and y center of mass
x_center = (img.sum(axis=0) * np.arange(cols)).sum() / total_mass
```

```
y_center = (img.sum(axis=1) * np.arange(rows)).sum() / total_mass

# Create a single scalar as a centroid feature using x+(y * cols) where
 ↪  cols is the width of the image
centroid = x_center + (y_center * cols)
return centroid
```

Εκτιμούμε την ακρίβειά του ταξινομητή μας και λαμβάνουμε την υψηλότερη μέχρι στιγμής δηλαδή accurancy = 0.9752 Ωστόσο, όταν εισάγουμε και τον αριθμό 0 στα σύμβολά εκπαίδευσης η ακρίβεια του ταξινομητή μειώνεται δραστικά και συγκεκριμένα λαμβάνουμε accurancy =0.7156.Αυτό οφείλεται στο γεγονός ότι το τρίτο χαρακτηριστικό δεν βοηθάει ιδιαίτερα στην ταξινόμηση αλλά και γιατί τα σύμβολα 2 και 0 παρουσιάζουν παρόμοιες στατιστικές ιδιότητες.

Θέμα 6: Minumum risk:
Από το κριτήριο ελαχιστοποίησης γνωρίζουμε ότι:

$$l_1 = \lambda_{11}P(\omega_1)P(x|\omega_1) + \lambda_{21}P(\omega_2)P(x|\omega_2) = \lambda_{21}P(\omega_2)P(x|\omega_2)$$

και,

$$l_2 = \lambda_{12}P(\omega_1)P(x|\omega_1) + \lambda_{22}P(\omega_2)P(x|\omega_2) = \lambda_{12}P(\omega_1)P(x|\omega_1)$$

Το όριο της απόφασης είναι:

$$l_1 = l_2 \Rightarrow \lambda_{21}P(\omega_2)P(x|\omega_2) = \lambda_{12}P(\omega_1)P(x|\omega_1)$$

Έχουμε ακόμα: $P(\omega_1) = P(\omega_2)$ οπότε:

$$\lambda_{21}\frac{x_0}{\sigma_2^2}e^{-\frac{{x_0}^2}{2\sigma_2^2}} = \lambda_{12}\frac{x_0}{\sigma_1^2}e^{-\frac{{x_0}^2}{2\sigma_1^2}}$$

Ακόμα, γνωρίζουμε ότι: $\lambda_{21} = 1, \lambda_{12} = 0.5, \sigma_1 = 1$ και $\sigma_2 = 2$ συνεπώς:

$$\begin{aligned}
\frac{x_0}{4}e^{-\frac{x_0^2}{8}} &= \frac{x_0}{2}e^{-\frac{x_0^2}{2}} \iff \\
2x_0e^{-\frac{{x_0}^2}{8}} &= 4x_0e^{-\frac{{x_0}^2}{2}} \Rightarrow \\
\ln(e^{-\frac{{x_0}^2}{8}}) &= \ln(2e^{-\frac{{x_0}^2}{2}}) \Rightarrow \\
-\frac{{x_0}^2}{8} &= \ln(2) - \frac{{x_0}^2}{2} \Rightarrow \\
-x_0^2 &= 8\ln(2) - 4x_0^2 \Rightarrow \\
3x_0^2 &= 8\ln(2) \Rightarrow \\
x_0 &= \sqrt{\frac{8\ln 2}{3}}
\end{aligned}$$

Αν το όριο απόφασης είναι 1.36 τότε για $x < 1.36$ κατηγοριοποιείται στην πρώτη κλάση και αντίστοιχα για $x > 1.36$ τότε το x κατηγοριοποιείται στην δεύτερη κλάση.