



Πολυτεχνείο Κρήτης

Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αναφορά 1ης Σειράς Ασκήσεων

Συγγραφέας:
Καπελώνης Χαρίλαος

Τιμεύθυνος Καθηγητής:
Δρ. Β. Διακολουκάς

Η εργασία κατατέθηκε για το μάθημα:
Στατιστική Μοντελοποίηση & Αναγνώριση Προτύπων

Εαρινό Εξάμηνο 2024

Περιεχόμενα

Περιεχόμενα	i
1 Θέμα 1: Principal Component Analysis (PCA)	1
2 Θέμα 2: Σχεδιάστε ένα ταξινομητή LDA (Linear Discriminant Analysis)	26
3 Θέμα 3: LDA vs PCA	29
4 Θέμα 4: Bayes	44
5 Θέμα 5: Εξαγωγή χαρακτηριστικών και Bayes Classification	50
6 Θέμα 6: Minimum risk	62
7 Πηγές	64

Θέμα 1: Principal Component Analysis (PCA)

Μέρος 1

a)

Φόρτωση δεδομένων:

```
% The following command loads the breast cancer dataset. You
    should now have the
% variable X in your environment
Data = csvread('data/breast_cancer_data.csv');
```

b)

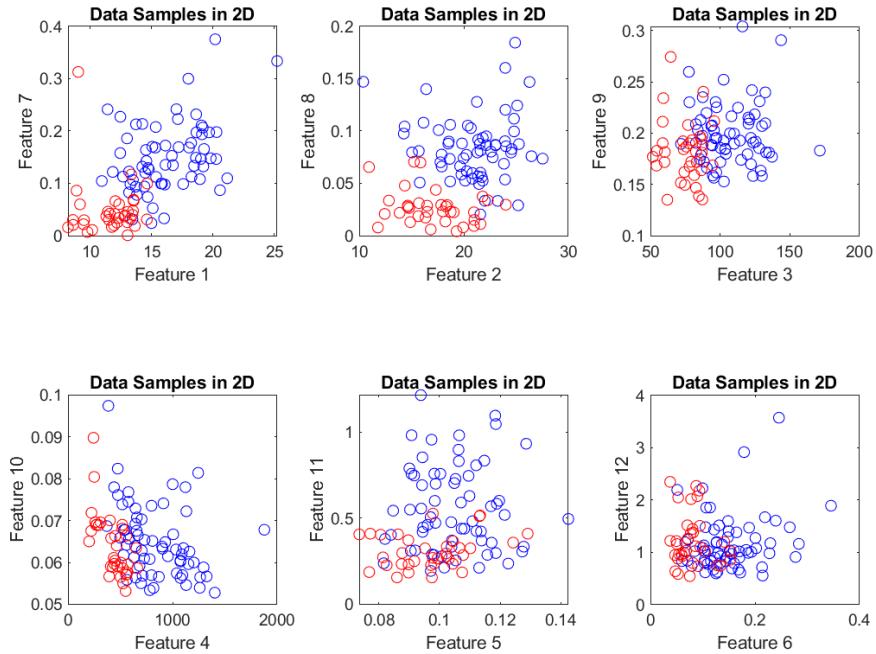
Από τα 569 δείγματα λαμβάνονται μόνο τα πρώτα 100 και από τα 30 χαρακτηριστικά επιλέγονται διάφορα ζέυγη για να απεικονιστούν σε 2D.

```
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);

fprintf('Visualizing example dataset for PCA.\n\n');
% Visualize the example dataset in 2D using sample feature pairs
```

```
% Repeat using different feature pairs on the 2D space
for i = 1:6
    subplot(2, 3, i);
    plot(X(Y==0, i), X(Y==0, i+6), 'bo', X(Y==1, i), X(Y==1, i+6),
        'ro' );
    axis square;
    title('Data Samples in 2D')
    xlabel(sprintf('Feature %d', i));
    ylabel(sprintf('Feature %d', i+6));
end
```

Έτσι, υπάρχει μία εικόνα αρχική εικόνα για τα δεδομένα:



c)

Τα δεδομένα δεν είναι κανονικοποιημένα.

Θα εφαρμοστεί η μέθοδος κανονικοποίησης standardization με την οποία τα νέα δεδομένα έχουν μέση τιμή 0 και διασπορά 1.

Ακολουθεί η συνάρτηση featureNormalize(X) που το υλοποιεί αυτό (βασίστηκε στις διαφάνειες του φροντιστηρίου):

```

function [X_norm, mu, sigma] = featureNormalize(X)
%FEATURENORMALIZE Normalizes the features in X
% FEATURENORMALIZE(X) returns a normalized version of X where
% the mean value of each feature is 0 and the standard
% deviation
% is 1. This is often a good preprocessing step to do when
% working with learning algorithms.

% ADD YOUR CODE
[nSamples, nFeat] = size(X);

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

```

Εδώ καλείται στον κύριο κώδικα ως εξής:

```

fprintf('\nRunning PCA on example dataset taking the first 2
       features.\n\n');
X=Data(1:NSamples, 1:2); % Get 2 first features

% Before running PCA, it is important to first normalize X
% Add the necessary code to perform standardization in
    featureNormalize
[X_norm, ~, ~] = featureNormalize(X);

% Plotting before and after standardization
figure;

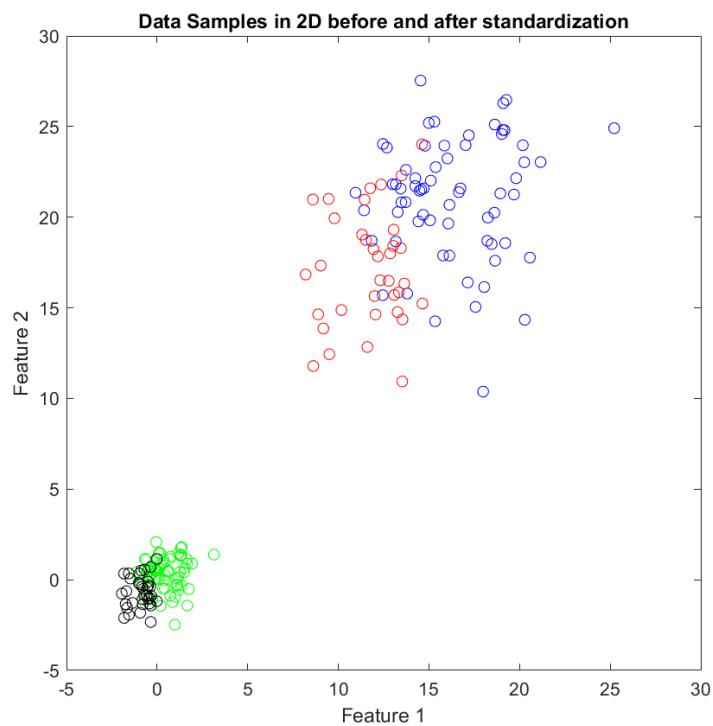
```

```

plot(X(Y==0, 1), X(Y==0, 2), 'bo', X(Y==1, 1), X(Y==1, 2), 'ro' );
axis square;
title('Data Samples in 2D before and after standardization')
xlabel('Feature 1');
ylabel('Feature 2');
hold on
plot(X_norm(Y==0, 1), X_norm(Y==0, 2), 'go', X_norm(Y==1, 1),
X_norm(Y==1, 2), 'ko' );

```

Και ακολουθεί το figure στο οποίο συγχρίνονται τα χανονικοποιημένα με τα αρχικά δεδομένα:



Γενικά, με την χανονικοποίηση τα δεδομένα είναι πιο αξιόπιστα, ενώ διευκολύνονται στη συνέχεια και οι πράξεις.

d)

Για αυτό το ερώτημα χρειάστηκε να υλοποιηθεί η συνάρτηση function [eigenval, eigenvec, order] = myPCA(X) για να εφαρμοστεί η μέθοδος ανάλυσης PCA:

```

function [eigenval, eigenvec, order] = myPCA(X)
%PCA Run principal component analysis on the dataset X
% [ eigenval, eigenvec, order] = mypca(X) computes
% eigenvectors of the autocorrelation matrix of X
% Returns the eigenvectors, the eigenvalues (on diagonal) and
% the order
%
%
% Useful values
[nSamples, nFeat] = size(X);

% Make sure each feature from the data is zero mean
[X_centered, mu, sigma] = featureNormalize(X);

R = 1/nSamples .* transpose(X_centered) * X_centered;

%D: diagonal matrix of eigenvalues
%V: Matrix columns are the eigenvectors
[V, D] = eig(R);

eigenval = diag(D); %Vector of eigenvalues
[eigenval, order] = sort(eigenval, 1, 'descend'); %Sort them
eigenvec = V(:, order); %Corresponding eigenvectors

end

```

Η συνάρτηση βασίστηκε στις διαλέξεις φροντιστηρίου.

Το νόημα της μεθόδου είναι η μεγιστοποίηση της διασποράς των δεδομένων πάνω στο νέο σύστημα βάσεων. Αυτό γίνεται με την ελαχιστοποίηση των τετραγώνων των αποστάσεων από το νέο σύστημα και, άρα, μεγιστοποίηση των προβολών των δεδομένων πάνω στο νέο σύστημα.

Ουσιαστικά επιστρέφει ιδιοδιανύσματα και ιδιοτιμές από ανάλυση eigendecomposition του πίνακα R. Ο πίνακας αυτός είναι ο πίνακας συνδιασποράς των χαρακτηριστικών δεδομένων. Αν ληφθούν σωστά τα

ιδιοδιανύσματα και οι ιδιοτιμές, τότε το αποτέλεσμα είναι ένα διάνυσμα με τις ιδιοτιμές που αντιστοιχούν σε καθένα χαρακτηριστικό και εφημηνεύονται ως το πόσο καλό είναι το αντίστοιχο ιδιοδιάνυσμα στο να επιτύχουμε μεγιστοποίηση των προβολών των δεδομένων. Προσοχή, τα ιδιοδιανύσματα να επιστραφούν σε φθίνουσα σειρά σημαντικότητας (φθίνοντες ιδιοτιμές).

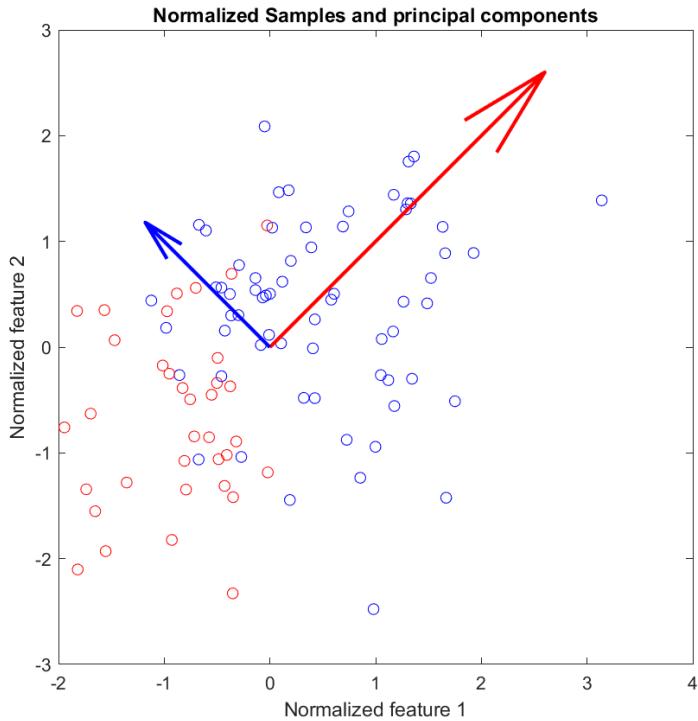
Εδώ γίνεται η κλήση της συνάρτησης, αλλά και μία αξιοσημείωτη απεικόνιση των δεδομένων μαζί με τα ιδιοδιανύσματα (**κύριες συνιστώσες**):

```
% Run PCA
[eigvals, eigvecs, ~] = myPCA(X_norm);

% Plot the samples on the first two principal components
figure;
plot(X_norm(Y==0, 1), X_norm(Y==0, 2), 'bo', X_norm(Y==1, 1),
      X_norm(Y==1, 2), 'ro');
hold on
% Plot the principal component vectors (arrows)
scale = 3; % Scale factor for the arrows
quiver(0, 0, eigvecs(1,1)*scale*eigvals(1),
       eigvecs(2,1)*scale*eigvals(1), 'r', 'LineWidth', 2,
       'MaxHeadSize', 1);
quiver(0, 0, eigvecs(1,2)*scale*eigvals(2),
       eigvecs(2,2)*scale*eigvals(2), 'b', 'LineWidth', 2,
       'MaxHeadSize', 1);
title('Normalized Samples and principal components')
xlabel('Normalized feature 1')
ylabel('Normalized feature 2')
axis square
hold off
```

Ακολουθεί το figure:

Οι κύριες συνιστώσες είναι πάντα **κάθετες** μεταξύ τους.



e)

H explained variance είναι μία μετρική του ποσοστού της συνολικής διασποράς που "εξηγείται" από την κάθε κύρια συνιστώσα και υπολογίζεται παραχάτω:

```
ExplainedVar = eigvals/sum(eigvals);
```

Και το αποτέλεσμα:

```
Explained Variance (1st PC = 0.687891) (2nd PC = 0.312109)
```

Προφανώς αθροίζουν στην μονάδα.

Για τη μέθοδο PCA το μόνο που μένει είναι η μείωση της διάστασης των δεδομένων και εδώ από 2D γίνονται 1D.

Η συνάρτηση function $Z = \text{projectData}(X, U, K)$ επιστρέφει τα νέα δεδομένα, προβλημένα στις K πρώτες κύριες συνιστώσες:

```

function Z = projectData(X, U, K)
    %PROJECTDATA Computes the reduced data representation when
    %projecting only
    %on to the top k eigenvectors
    % Z = projectData(X, U, K) computes the projection of
    % the normalized inputs X into the reduced dimensional space
    % spanned by
    % the first K columns of U. It returns the projected examples
    % in Z.
    %

    % You need to return the following variables correctly.
    Z = zeros(size(X, 1), K);

    % Instructions: Compute the projection of the data using only
    % the top K
    %
    %           eigenvectors
    %
    Z = Z + X * U(:, [1:K])/norm(U(:, [1:K])); % throws error if
    % dimensions do not match

end

```

Οπότε, χρατείται μόνο το πρώτο ιδιοδιάνυσμα και σε αυτό θα γίνει η προβολή των δεδομένων:

```

% Projection of the data onto the principal components
X_PCA = projectData(X_norm, eigvecs, 2);
X_PCA_1 = X_PCA(:, 1)*(eigvecs(:, 1).');
X_PCA_2 = X_PCA(:, 2)*(eigvecs(:, 2).');

% Plot the projection of the data onto the principal components
figure;
hold on
axis square;
plot(X_PCA_1(:, 1), X_PCA_1(:, 2), 'bo', 'MarkerFaceColor', 'b',
      'MarkerEdgeColor', 'none', 'MarkerSize', 4 );
plot(X_PCA_2(:, 1), X_PCA_2(:, 2), 'ro', 'MarkerFaceColor', 'r',
      'MarkerEdgeColor', 'none', 'MarkerSize', 4 );
title('PCA of Breast Cancer Dataset');
xlabel('Principal Component 1');

```

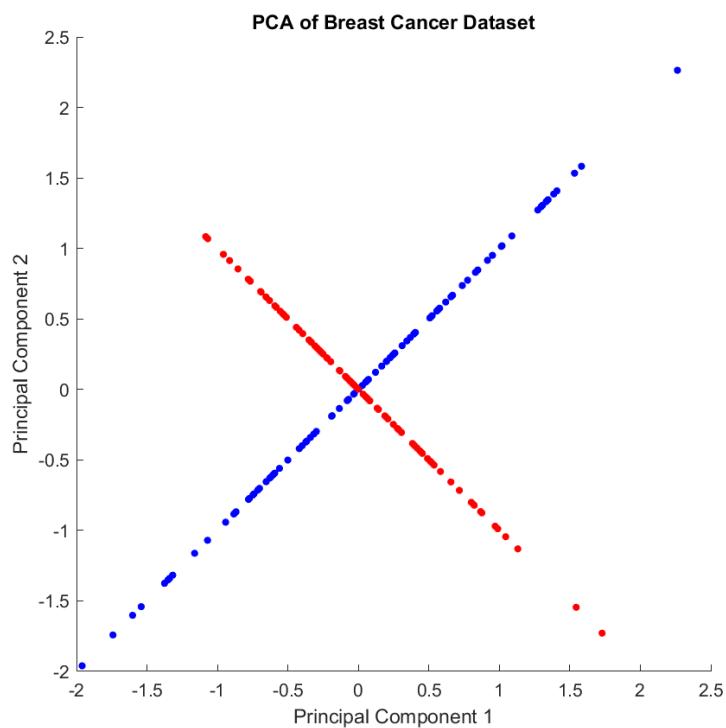
```

ylabel('Principal Component 2');
hold off;

% Print the first principal component
fprintf(' 1st Principal Component = %f %f \n', eigvecs(1,1),
eigvecs(2,1));

```

Τα αποτελέσματα του παραπάνω χώδικα:



Και στην χονσόλα:

```
1st Principal Component = 0.707107 0.707107
```

Δ ηλαδή η ευθεία $y = x$.

f)

Αν αντιστραφεί η διαδικασία προβολής των δεδομένων, μπορεί να επιτευχθεί προσεγγιστικά ανάκτηση δεδομένων, εδώ από 1D σε 2D.

Τα μαθηματικά έχουν ως εξής:

$$\begin{aligned} Z &= X\mathbf{U} \\ Z\mathbf{U}^T &= X\mathbf{U}\mathbf{U}^T \\ Z\mathbf{U}^T &= X\|\mathbf{U}\|^2 \quad (*) \\ Z\mathbf{U}^T &= X \\ X_{\text{rec}} &= X = Z\mathbf{U}^T \end{aligned}$$

(*) μοναδιαίο διάνυσμα

H συνάρτηση function `X_rec = recoverData(Z, U, K)` κάνει ακριβώς αυτό:

```

function X_rec = recoverData(Z, U, K)
%RECOVERDATA Recovers an approximation of the original data when
    % using the
%projected data
% X_rec = RECOVERDATA(Z, U, K) recovers an approximation the
% original data that has been reduced to K dimensions. It
    % returns the
% approximate reconstruction in X_rec.
%

% You need to return the following variables correctly.
X_rec = zeros(size(Z, 1), size(U, 1));

% Instructions: Compute the approximation of the data by
    % projecting back
%         onto the original space using the top K
    % eigenvectors in U.
%
%

X_rec = X_rec + Z * U(:, [1:K]).'; % throws error if dimensions
    % do not match

```

```
end
```

Και ο υπόλοιπος κώδικας την εφαρμόζει και απεικονίζει γραφικά κάποια ενδιαφέροντα figures:

```
fprintf('\nDimensionality reduction on example dataset.\n\n');

% Project the data onto K = 1 dimension
K = 1;
Z = projectData(X_norm, eigvecs, K);
fprintf('Projection of the first example: %f\n', Z(1));

% Plot the data in this reduced dimension space
figure;
plot(Z(Y==0), Z(Y==0), 'bo', Z(Y==1), Z(Y==1), 'ro'); % eigvec
    % is y=x line in 2D space
title('Reduced data');
xlabel('Values of data');

% Use the K principal components to recover the Data in 2D
X_recover = recoverData(Z, eigvecs, K);
fprintf('Approximation of the first example: %f %f\n',
    X_recover(1, 1), X_recover(1, 2));

% Draw the lines connecting the projected points to the
% original points
figure;
hold on;
plot(X_recover(Y==0, 1), X_recover(Y==0, 2),
    'co', 'MarkerFaceColor', 'c', 'MarkerEdgeColor', 'none',
    'MarkerSize', 5);
plot(X_recover(Y==1, 1), X_recover(Y==1, 2), 'yo',
    'MarkerFaceColor', 'y', 'MarkerEdgeColor', 'none',
    'MarkerSize', 5);

plot(X_norm(Y==0, 1), X_norm(Y==0, 2), 'bo', 'MarkerFaceColor',
    'b', 'MarkerEdgeColor', 'none', 'MarkerSize', 5 )
plot(X_norm(Y==1, 1), X_norm(Y==1, 2), 'ro', 'MarkerFaceColor',
    'r', 'MarkerEdgeColor', 'none', 'MarkerSize', 5 );
title('PCA sample projections');

axis square
for i = 1:size(X_norm, 1)
```

```

drawLine(X_norm(i,:), X_recover(i,:), '--k', 'LineWidth', 1);
end
axis([-3 3.5 -3 3.5]);
axis square
hold off

```

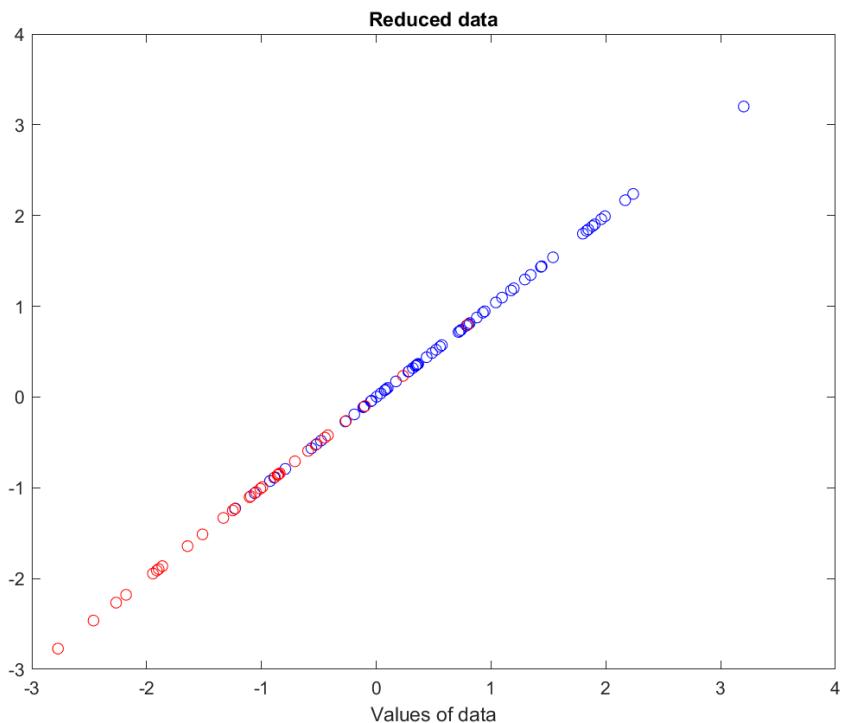


Figure 1.1: Απεικόνιση δεδομένων πάνω στην κύρια συνιστώσα, διαφορετικό χρώμα ανά κλάση

g)

Η ίδια διαδικασία έγινε και για τα 30 χαρακτηριστικά (προηγουμένως ανάλυση για τα 2 πρώτα μόνο):

```

% Apply PCA on the initial dataset and then choose the first 2
% Principal Components
% to reduce the dimensionality of the dataset to the 2D space

```

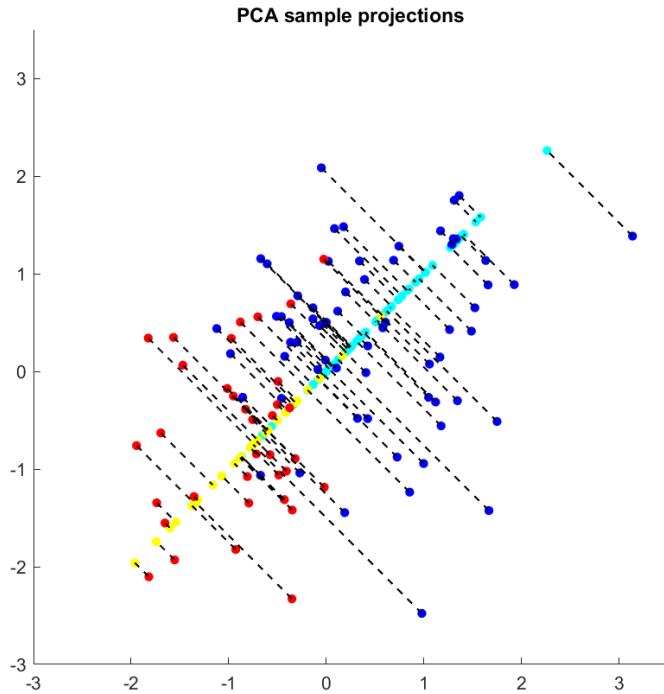


Figure 1.2: Ανακτημένα δεδομένα και προβολές πάνω στην πρώτη κύρια συνιστώσα

```
% The following code will load the dataset into your environment
%
X=Data(1:NSamples,1:end-1); % Get all features from the breast
cancer dataset

% Before running PCA, it is important to first normalize X
% Add the necessary code to perform standardization in
    featureNormalize
[X_norm, ~, ~] = featureNormalize(X);

% Run PCA
[eigvals, eigvecs, order] = myPCA(X_norm);

figure
hold on
```

```
% Plot the samples on the first two principal components
plot(X_norm(Y==0, order(1)), X_norm(Y==0, order(2)),
      'bo', 'MarkerFaceColor', 'b', 'MarkerEdgeColor', 'none',
      'MarkerSize', 5 )
plot(X_norm(Y==1, order(1)), X_norm(Y==1, order(2)),
      'ro', 'MarkerFaceColor', 'r', 'MarkerEdgeColor', 'none',
      'MarkerSize', 5 );
title('PCA sample projections');
% Plot the principal component vectors (arrows)
scale = 1; % Scale factor for the arrows
quiver(0, 0, eigvecs(1,1)*scale*eigvals(1),
       eigvecs(2,1)*scale*eigvals(1), 'r', 'LineWidth', 2,
       'MaxHeadSize', 1);
quiver(0, 0, eigvecs(1,2)*scale*eigvals(2),
       eigvecs(2,2)*scale*eigvals(2), 'b', 'LineWidth', 2,
       'MaxHeadSize', 1);
hold off;
axis square;

% Projection of the data onto the principal components
% ADD YOUR CODE
K = 2;
X_PCA = projectData(X_norm, eigvecs, K);

% Projection of the data onto the principal components
X_PCA_1 = X_PCA(:, 1)*(eigvecs(:, 1).');
X_PCA_2 = X_PCA(:, 2)*(eigvecs(:, 2).');
% Plot the projection of the data onto the principal components
figure;
hold on
axis square;
plot(X_PCA_1(:, 1), X_PCA_1(:, 2), 'bo', 'MarkerFaceColor', 'b',
      'MarkerEdgeColor', 'none', 'MarkerSize', 4 );
plot(X_PCA_2(:, 1), X_PCA_2(:, 2), 'ro', 'MarkerFaceColor', 'r',
      'MarkerEdgeColor', 'none', 'MarkerSize', 4 );
title('Breast Cancer Dataset - PCA reduced in 2D');
xlabel('Principal Component 1');
ylabel('Principal Component 2');
hold off;

% Calculate the Explained Variance of the first 2 Principal
% Components
% ADD YOUR CODE
```

```

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

% Print the first principal component
fprintf(' 1st Principal Component = %f %f \n', eigvecs(1,1),
        eigvecs(2,1));

```

Παρόμοια με πριν, έχει ενδιαφέρον το αποτέλεσμα στην κονσόλα:

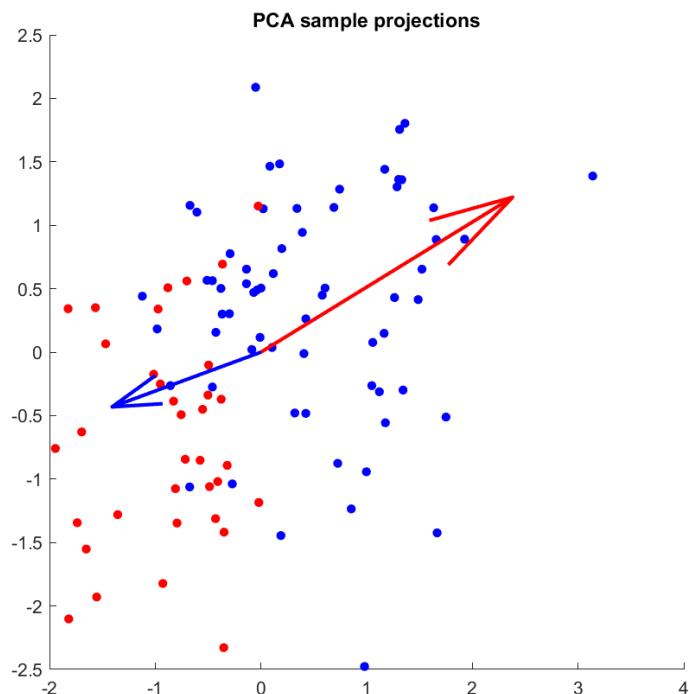
```

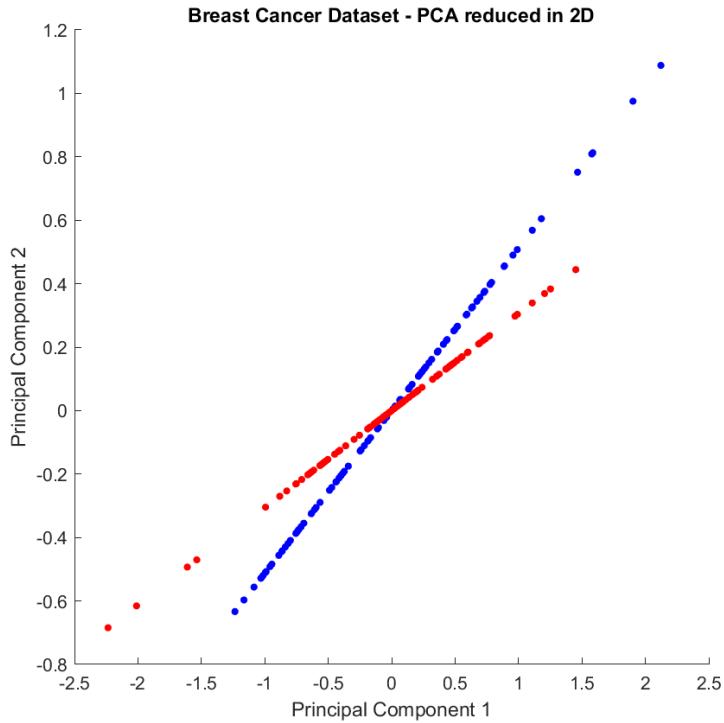
Explained Variance (1st PC = 0.430518) (2nd PC = 0.207239)
1st Principal Component = 0.206988 0.106123

```

Οπότε οι πρώτες δύο συνιστώσες έχουν τα παραπάνω ποσοστά και η πρώτη κύρια συνιστώσα είναι διαφορετική από πριν.

Τα αντίστοιχα figures:





Μέρος 2

a)

Για τη φόρτωση και την απεικόνιση των πρώτων 100 προσώπων από τα 5000:

```
% Load Face dataset
load ('data/faces.mat')

% Display the first 100 faces in the dataset
figure;
displayData(X(1:100, :));
```



Figure 1.3: 100 πρόσωπα

b)

Όπως και στο Μέρος 1, γίνεται η κανονικοποίηση:

```
% Before running PCA, it is important to first normalize X by
% subtracting
% the mean value from each feature
[X_norm, mu, sigma] = featureNormalize(X);
```

Έπειτα ακολουθεί η εφαρμογή της μεθόδου PCA:

```
% Run PCA
[eigvals, eigvecs, order] = myPCA(X_norm);
```

Για την προβολή των 36 πρώτων συνιστωσών:

```
% Visualize the top #numOfComps eigenvectors found (eigenfaces)
displayData(eigvecs(:, 1:36));
```

To `figure` δείχνει κάτι που μοιάζει με θολωμένα πρόσωπα:



Μάλιστα, το πρώτο φαίνεται το πιο ρεαλιστικό και καθαρό, ενώ τα υπόλοιπα όσο πάει και μοιάζουν με απόμακρη σκιά, παρά με πρόσωπο.

c)

Ομοίως με πριν, από 1024 `features` προβάλλονται τα 100 μόνο:

```
K = 100;
Z = projectData(X_norm, eigvecs, K);

fprintf('The projected data Z has a size of: ')
fprintf('%d ', size(Z));
```

Κονσόλα:

```
The projected data Z has a size of: 5000 100
```

d)

Η ανάκτηση των αρχικών διαστάσεων και δεδομένων:

```
fprintf('\nDimension reduction for face dataset.\n\n');

% Display normalized data
figure;
displayData(X_norm(1:100,:));
title('Original faces');
axis square;

for K = [10, 50, 100, 200, 500]
Z = projectData(X_norm, eigvecs, K);

fprintf('The projected data Z has a size of: ')
fprintf('%d ', size(Z));

fprintf('\n\nProgram paused. Press enter to continue.\n');
pause;

%% ===== Part 8: Visualization of Faces after PCA
% Dimensionality Reduction =====
% Project images to the eigen space using the top K eigen
% vectors and
% visualize only using those K dimensions
% Compare to the original input, which is also displayed

fprintf('\nVisualizing the projected (reduced dimension)
faces.\n\n');

X_rec = recoverData(Z, eigvecs, K);

% Display reconstructed data from only k eigenfaces
figure;
displayData(X_rec(1:100,:));
title(sprintf('Recovered faces (%d principal components)',
```

```

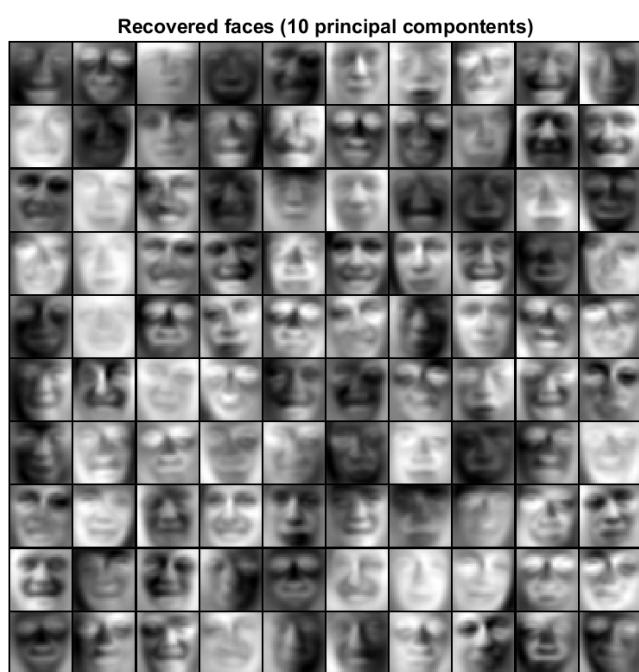
    K));
axis square;
end

```

Ακολουθούν τα `figures` που απεικονίζουν τα αρχικά 100 πρόσωπα και οι ανακατασκευές τους για διαφορετικό αριθμό από κύριες συνιστώσες, δηλαδή πόσες διαστάσεις καταλήγουν να έχουν τα νέα δεδομένα:



Είναι λογικό όταν υποδιπλασιάζονται οι συνιστώσες (βλ. εδώ), δηλαδή από περίπου 1000 σε 500, να γίνεται ανάκτηση δεδομένων με μεγάλη ακρίβεια, αλλά το πόσο επιτυχής είναι η ανάκτηση στην περίπτωση που από 1000 πέφτει σε 10 μόνο κύριες συνιστώσες είναι αξιοσημείωτη (βλ. εδώ).





Recovered faces (100 principal components)



Recovered faces (200 principal components)





Θέμα 2: Σχεδιάστε ένα ταξινομητή LDA (Linear Discriminant Analysis)

Ερώτημα 1

Για την πιθανότητα του $P(\omega_B)$:

$$\begin{aligned} P(\omega_B) &= 1 - P(\omega_A) \\ &= 1 - 0.4 \\ &= 0.6 \end{aligned}$$

Για τους υπολογισμό του πίνακα σκέδασης S_w :

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

Υπολογίζουμε την ορίζουσα του S_w :

$$\begin{aligned} \det(S_w) &= \det\left(\begin{bmatrix} 2.4 & 1 \\ 1 & 2.6 \end{bmatrix}\right) \\ &= 2.4 \cdot 2.6 - 1 \cdot 1 \\ &= 5.24 \end{aligned}$$

Και τον αντίστροφό του (S_w^{-1}):

$$\begin{aligned} S_w^{-1} &= \frac{1}{\det(S_w)} \begin{bmatrix} 2.6 & -1 \\ -1 & 2.4 \end{bmatrix} \\ &= \frac{1}{5.24} \begin{bmatrix} 2.6 & -1 \\ -1 & 2.4 \end{bmatrix} \end{aligned}$$

Το διάνυσμα προβολής \mathbf{w} υπολογίζεται ως εξής:

$$\begin{aligned} \mathbf{w} &= S_w^{-1}(\mu_A - \mu_B) \\ &= \frac{1}{5.24} \begin{bmatrix} 2.6 & -1 \\ -1 & 2.4 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right) \\ &= \frac{1}{5.24} \begin{bmatrix} 2.6 & -1 \\ -1 & 2.4 \end{bmatrix} \begin{bmatrix} -2 \\ 2 \end{bmatrix} \\ &= \frac{1}{5.24} \begin{bmatrix} -7.2 \\ 6.8 \end{bmatrix} \\ &= \begin{bmatrix} \frac{-7.2}{5.24} \\ \frac{6.8}{5.24} \end{bmatrix} \\ &\approx \begin{bmatrix} -1.374 \\ 1.298 \end{bmatrix} \end{aligned}$$

Συνεπώς:

$$\mathbf{w} = \begin{bmatrix} -1.374 \\ 1.298 \end{bmatrix}$$

Ερώτημα 2

Για την προβολή του \mathbf{x}_A πάνω στο \mathbf{w} , έστω y_A , αρχικά υπολογίζεται το μέτρο του \mathbf{w} :

$$\|\mathbf{w}\| = \sqrt{(-1.374)^2 + (1.298)^2} \approx 1.889$$

Έπειτα:

$$\begin{aligned} y_A &= \frac{\mathbf{w}^T \mathbf{x}_A}{\|\mathbf{w}\|} \\ &= \frac{[-1.374 \quad 1.298] \begin{bmatrix} 1 \\ 3 \end{bmatrix}}{1.889} \\ &\approx 1.334 \end{aligned}$$

Ομοίως για \mathbf{x}_B :

$$\begin{aligned} y_B &= \frac{\mathbf{w}^T \mathbf{x}_B}{\|\mathbf{w}\|} \\ &= \frac{[-1.374 \quad 1.298] \begin{bmatrix} 3 \\ -1 \end{bmatrix}}{1.889} \\ &\approx -2.869 \end{aligned}$$

Τελικά:

$$y_A \approx 1.334 \quad \text{και} \quad y_B \approx -2.869$$

Θέμα 3: LDA vs PCA

Μέρος 1

a)

Για τη φόρτωση των δειγμάτων:

```
load('data/data2.mat');
```

Οι μεταβλητές c και X περιέχουν τα χαρακτηριστικά διανύσματα.

b)

Για το standardization επαναχρησιμοποιήθηκε η συνάρτηση featureNormalize(X):

```
% Before running LDA, it is important to first normalize X
[X_norm, ~, ~] = featureNormalize(X);

X1 = X_norm(c==1, :);
X2 = X_norm(c==2, :);

% Visualize the example dataset
figure;
subplot(1, 2, 1)
title('LDA')
```

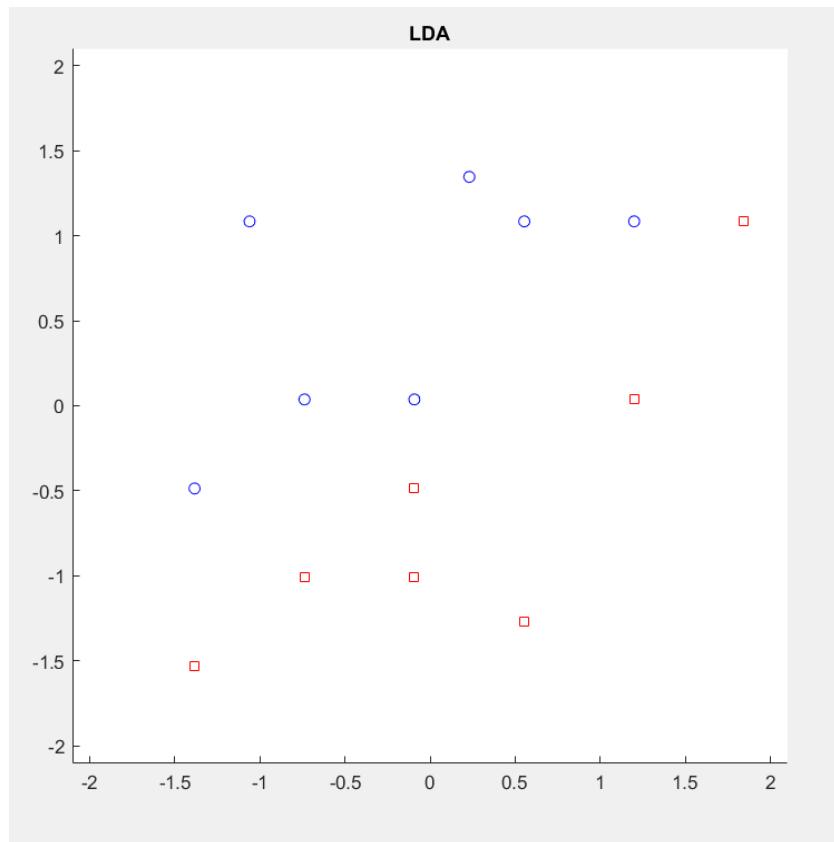
```

hold on
plot(X1(:, 1), X1(:, 2), 'bo');
plot(X2(:, 1), X2(:, 2), 'rs');
axis([-2.1 2.1 -2.1 2.1]); axis square;
hold off

fprintf('Program paused. Press enter to continue.\n');
pause;

```

Ακολουθεί το **figure** των δειγμάτων, με διαφορετικό χρώμα ανάλογα την κλάση:



c)

Η συνάρτηση `fisherLinearDiscriminant.m` υλοποιεί την ανάλυση LDA για την περίπτωση δύο κλάσεων.

Παίρνει ως είσοδο τα δείγματα των δύο κλάσεων και επιστρέφει ένα μονοδιάστατο διάνυσμα - με νόρμα ίση με την μονάδα - έχοντας έτσι μειώσει τη διάσταση και ελαχιστοποιώντας την απώλεια διαχωριστικότητας ανάμεσα στις δύο κλάσεις.

```

function v = fisherLinearDiscriminant(X1, X2)
m1 = size(X1, 1);
m2 = size(X2, 1);
m = m1 + m2;

mu1 = mean(X1);
mu2 = mean(X2);

S1 = 1/m1 * X1.' * X1;
S2 = 1/m2 * X2.' * X2;

Sw = m1/m.*S1 + m2/m.*S2;

v = inv(Sw)*(mu1 - mu2).';

v = v/norm(v);

```

Κοιτάζοντας από κάτω προς τα πάνω, για τον υπολογισμό του διανύσματος **v** χρειάστηκε ο πίνακας σκέδασης **Sw** και οι μέσες τιμές για κάθε κλάση. Για τον πίνακα σκέδασης επίσης υπολογίστηκαν οι πίνακες σκέδασης των κλάσεων **S1**, **S2**.

Οι πράξεις αυτές συνοψίζονται στην προσπάθεια μεγιστοποίησης της μετρικής:

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

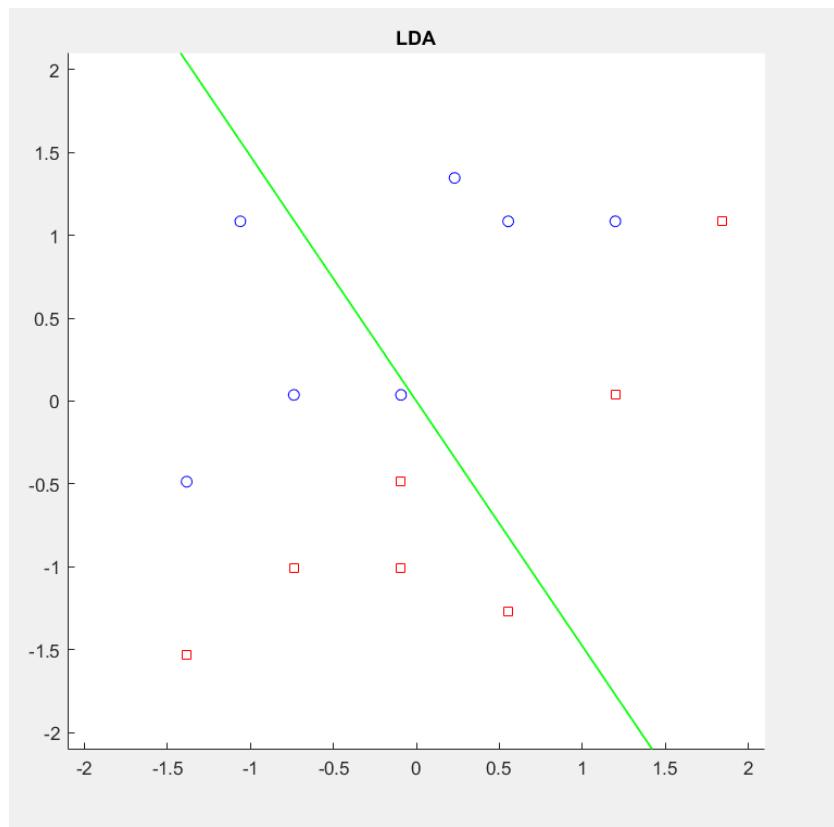
Δηλαδή, μεγάλη απόσταση μεταξύ των μέσων τιμών ανά κλάση (αριθμητής) και μικρή απόσταση του ενός δείγματος με τα υπόλοιπα ανά κλάση (παρονομαστής).

Καλώντας τη συνάρτηση:

```
v = fisherLinearDiscriminant(X1, X2);

hold on
drawLine(-5*v', 5*v', '-g', 'LineWidth', 1);
hold off
```

To figure απεικονίζει το διάνυσμα v (δέκα φορές μεγαλύτερο) ως εξής:



d)

Ο κώδικας για τη συνάρτηση `projectDataLDA(X, v)` είναι ο πολλαπλασιασμός των δειγμάτων με το διάνυσμα v (εφόσον είναι μοναδιαίο δε χρειάζεται κάτι αλλο):

```

function [Z] = projectDataLDA(X, v)

% You need to return the following variables correctly.
Z = zeros(size(X, 1), 1);

Z = Z + X*v;

end

```

e)

Αντίστοιχα, για τη `recoverDataLDA(Z, v)`:

```

function [X_rec] = recoverDataLDA(Z, v)

X_rec = zeros(size(Z, 1), length(v));

X_rec = X_rec + Z*v.';

end

```

Τα μαθηματικά πίσω από τα ερωτήματα d και e έχουν ως εξής:

$$\begin{aligned}
 Z &= Xv \\
 Zv^T &= Xvv^T \\
 Zv^T &= X\|v\|^2 \quad (*) \\
 Zv^T &= X \\
 X_{\text{rec}} &= X = Zv^T
 \end{aligned}$$

(*) μοναδιαίο διάνυσμα

Ακολουθεί ο κώδικας καλώντας τις συναρτήσεις των ερωτημάτων d και e:

```
% Project the data onto the direction of the one dimensional
vector v
[Z1] = projectDataLDA(X1, v);
[Z2] = projectDataLDA(X2, v);

% Reconstruct the data on the line defined by vector v
[X1_rec] = recoverDataLDA(Z1, v);
[X2_rec] = recoverDataLDA(Z2, v);

% Draw lines connecting the projected points to the original
points
fprintf('\nDisplaying LDA on example dataset.\n\n');
hold on;
plot(X1_rec(:, 1), X1_rec(:, 2), 'bo', 'MarkerFaceColor', 'b');
for i = 1:size(X1, 1)
    drawLine(X1(i,:), X1_rec(i,:), '--k', 'LineWidth', 1);
end

plot(X2_rec(:, 1), X2_rec(:, 2), 'rs', 'MarkerFaceColor', 'r');
for i = 1:size(X2, 1)
    drawLine(X2(i,:), X2_rec(i,:), '--k', 'LineWidth', 1);
end
hold off
```

Τα αποτελέσματα σε figure:

Δεν παρατηρείται κάποια επικάλυψη μεταξύ των δειγμάτων των κλάσεων. Ο διαχωρισμός παραμένει.

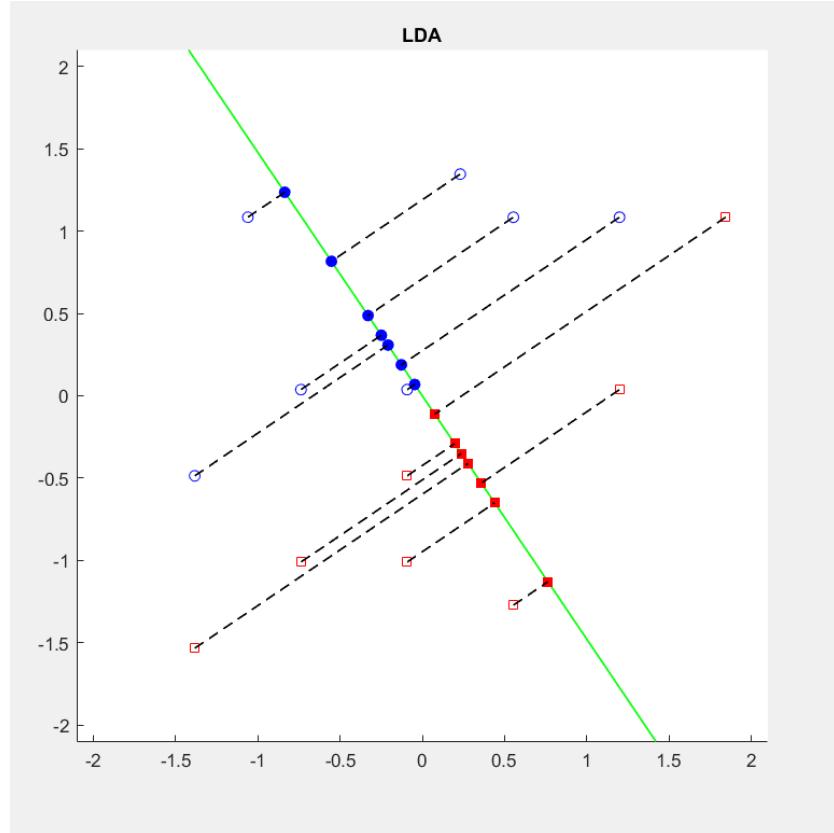
Η ανάλυση LDA φαίνεται σε μεγάλο βαθμό επιτυχής.

f)

Για το ερώτημα αυτό εφαρμόστηκε η μέθοδος PCA όπως και στο Θέμα 1 και τα δεδομένα μειώθηκαν σε μία διάσταση:

```
% Run PCA
[U, S] = myPCA(X_norm); % already standardized

% Project the data onto K = 1 dimension
```



```
K = 1;
Z = projectDataPCA(X_norm, U, K);

X_rec = recoverDataPCA(Z, U, K);
```

Οι συναρτήσεις που καλούνται είναι σχεδόν ακριβώς ίδιες με αυτές του Θέματος 1 για την ανάλυση PCA.

Ο κώδικας για την απεικόνιση (δίνεται από εκφώνηση):

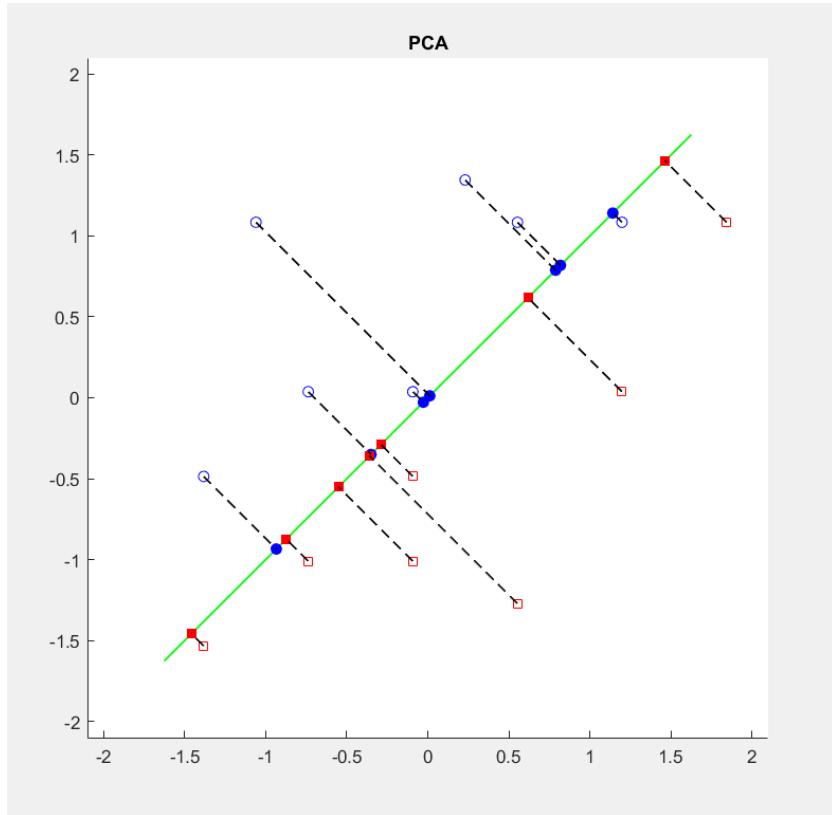
```
% Plot the normalized dataset (returned from
% principalComponentAnalysis)
% Draw lines connecting the projected points to the original
% points
fprintf('\nVisualizing example dataset for PCA.\n\n');
subplot(1, 2, 2)
title('PCA')
hold on;
```

```

axis([-2.1 2.1 -2.1 2.1]); axis square;
drawLine(-2.3*U(:,1), 2.3*U(:,1), '-g', 'LineWidth', 1);
plot(X1(:, 1), X1(:, 2), 'bo');
plot(X2(:, 1), X2(:, 2), 'rs');
plot(X_rec(c==1, 1), X_rec(c==1, 2), 'bo', 'MarkerFaceColor',
'b');
plot(X_rec(c==2, 1), X_rec(c==2, 2), 'rs', 'MarkerFaceColor',
'r');
for i = 1:size(X_norm, 1)
    drawLine(X_norm(i,:), X_rec(i,:), '--k', 'LineWidth', 1);
end
hold off

```

To figure μετά την ανάλυση PCA:



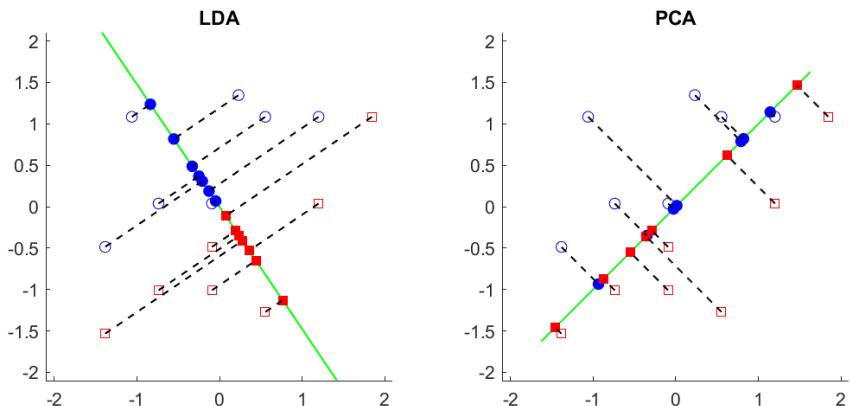
Παρατηρείται επικάλυψη μεταξύ των δειγμάτων των κλάσεων. Ο διαχωρισμός έχει χαθεί.

Η ανάλυση PCA φαίνεται να μην έχει τα επιθυμητά αποτελέσματα.

Σύγκριση LDA - PCA:

Στο παρακάτω figure είναι εμφανής η επιτυχής μείωση διάστασης με τη μέθοδο LDA έναντι της PCA ανάλυσης. Στα αριστερά φαίνεται ότι η LDA έχει καταφέρει να κρατήσει το διαχωρισμό των δειγμάτων, ενώ δεξιά η PCA έχει κάνει τα δείγματα των δύο κλάσεων να αναμειχθούν.

Έχοντας ήδη αναφέρει τη διαφορετική προσέγγιση κάθε μεθόδου (**PCA**, **LDA**) όσον αφορά τη μείωση της διάστασης των δεδομένων είναι προφανές ότι η μέθοδος LDA είναι πιο αποτελεσματική σε αυτό το πρόβλημα, επειδή είναι **supervised** και έχει στόχο τη διαχωριστικότητα των κλάσεων ακόμα και μετά την εφαρμογή της, ενώ η PCA είναι **unsupervised**, άρα δεν προσμετρά τις κλάσεις των δεδομένων.



Στο συγκεκριμένο πρόβλημα, λοιπόν, που δίνονται οι κλάσεις των δεδομένων, η μέθοδος LDA είναι καλύτερη από την PCA.

Mέρος 2

a)

Για παν ενδεχόμενο πρώτα κατέβηκε το dataset από τον σύνδεσμο <http://ucl-cs-grad.github.io/matlabgrad/lectures.html> στον φάκελο data.

```
% Load Fisher Iris Data
load('data/fisheriris.mat');
```

b)

Εδώ έγινε standardization και φορτώθηκαν σε τρεις πίνακες τα δεδομένα ανάλογα την κλάση τους.

```
% Before running LDA, it is important to first normalize X
[meas_norm, mu, sigma] = featureNormalize(meas);

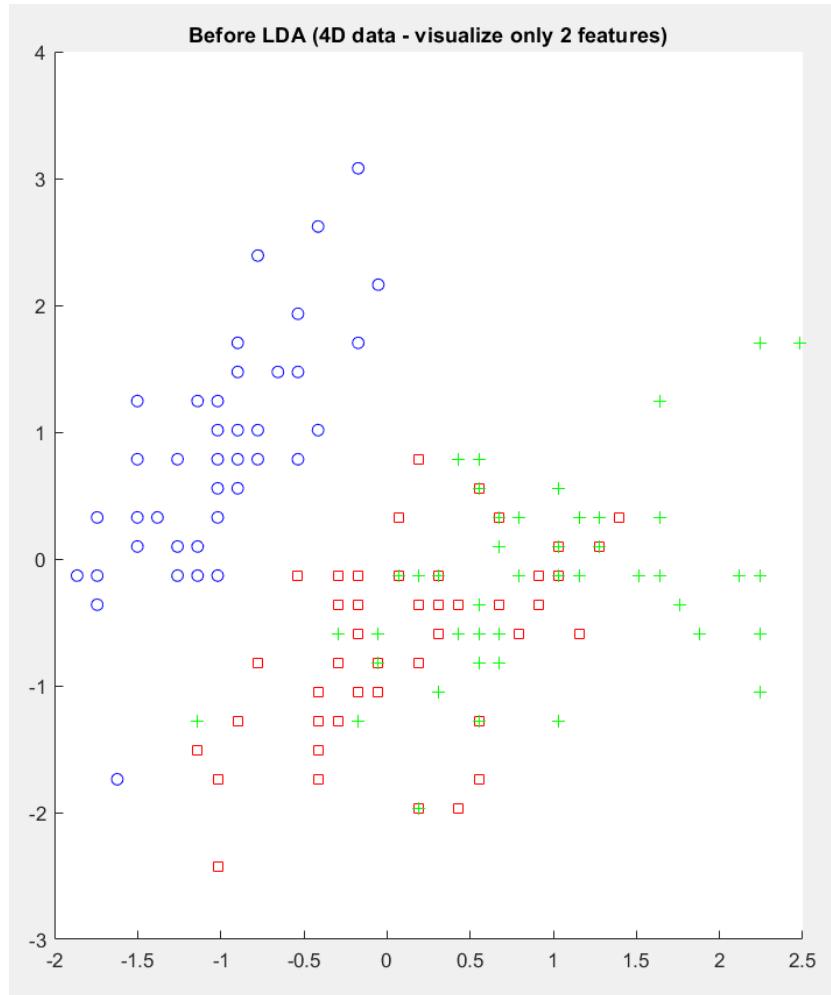
% Get the data for each class
IRIS1 = meas_norm(iris_labels == 0, :); %Samples of Class 0
IRIS2 = meas_norm(iris_labels == 1, :); %Samples of Class 1
IRIS3 = meas_norm(iris_labels == 2, :); %Samples of Class 2
```

Τα features είναι 4, άρα τα δεδομένα είναι 4D, συνεπώς απεικονίζονται σε ένα γράφημα οι δύο πρώτες στήλες μόνο, ξεχωρίζονται με άλλο χρώμα κάθε κλάση:

```
% Visualize the example dataset
figure;
subplot(1, 2, 1)
title('Before LDA (4D data - visualize only 2 features)')
hold on
plot(IRIS1(:, 1), IRIS1(:, 2), 'bo');
plot(IRIS2(:, 1), IRIS2(:, 2), 'rs');
plot(IRIS3(:, 1), IRIS3(:, 2), 'g+');
```

Στο παρακάτω figure διαχρίνει κανείς ότι τα δείγματα της πρώτης κλάσης

(μπλε) είναι χωρισμένα από τα άλλα δύο, τα οποία είναι ανακατεμένα μεταξύ τους (χόκκινα - πράσινα).



c)

Για αυτό το ερώτημα χρειάστηκε να υλοποιηθεί η μέθοδος LDA αλλά για πολλαπλές κλάσεις (όχι binary όπως στο Μέρος 1).

Η υλοποίηση έγινε στη συνάρτηση myLDA(Samples, Labels, NewDim):

```
function A = myLDA(Samples, Labels, NewDim)
% Input:
%   Samples: The Data Samples
```

```

% Labels: The labels that correspond to the Samples
% NewDim: The New Dimension of the Feature Vector after
applying LDA

[NumSamples NumFeatures] = size(Samples);

A = zeros(NumFeatures, NewDim);

NumLabels = length(Labels);
if(NumSamples ~= NumLabels) then
    fprintf('\nNumber of Samples are not the same with the
           Number of Labels.\n\n');
    exit
end

Classes = unique(Labels); %Return the unique elements of
                         Labels
NumClasses = length(Classes); %The number of classes

Sw = zeros(NumFeatures, NumFeatures);
Sb = zeros(NumFeatures, NumFeatures);

%Calculate the Global Mean
m0 = mean(Samples);

%For each class i
%Find the necessary statistics
for i = 1:NumClasses
    %Calculate the Class Prior Probability
    cl = Classes(i);
    P(i) = sum(Labels == cl) / NumLabels;

    %Calculate the Class Mean
    mu(i, :) = mean(Samples(Labels == cl, :));

    %Calculate the Within Class Scatter Matrix
    Sw = Sw + P(i) * cov(Samples(Labels == cl, :));

    %Calculate the Between Class Scatter Matrix
    Sb = Sb + sum(Labels == cl) * (mu(i, :) - m0).'* (mu(i,
                           :) - m0);
end

```

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

%Select the NewDim eigenvectors corresponding to the top
NewDim
%eigenvalues (Assuming they are NewDim<=NumClasses-1)
if(NewDim > NumClasses-1) then
    fprintf('Illegal arguments.\n\n');
    exit
end

%Perform Eigendecomposition

%D: diagonal matrix of eigenvalues
%V: Matrix columns are the eigenvectors
[V, D] = eig(EigMat);

eigenval = diag(D); %Vector of eigenvalues
[eigenval, order] = sort(eigenval, 1, 'descend'); %Sort them
eigenvec = V(:, order); %Corresponding eigenvectors

%% You need to return the following variable correctly.
A = zeros(NumFeatures, NewDim); % Return the LDA projection
vectors
A = A + eigenvec(:, 1:NewDim); % Produce error if dimensions
do not match
```

Η λογική είναι ίδια με το binary class case, ενώ τα μαθηματικά περιπλέχονται ελαφρώς.

Το νόημα είναι να βρεθεί ένας γραμμικός μετασχηματισμός, ένας πίνακας A , ο οποίος θα εφαρμοστεί πάνω στα δεδομένα και θα παράξει ένα νέο διάνυσμα δεδομένων με μειωμένη διάσταση.

Στη συγκεκριμένη περίπτωση επιλέχθηκε η νέα μειωμένη διάσταση να είναι $NewDim = 2$:

```
NewDim = 2; %The new feature dimension after applying LDA
v = myLDA(meas_norm, iris_labels, NewDim);
```

d)

Τέλος, καλείται η συνάρτηση `projectDataLDA(X, v)` για να εφαρμοστεί ο γραμμικός μετασχηματισμός και τα νέα δεδομένα είναι 2D:

```
function [Z] = projectDataLDA(X, v)

% You need to return the following variables correctly.
Z = zeros(size(X, 1), 1);

Z = Z + X*v;

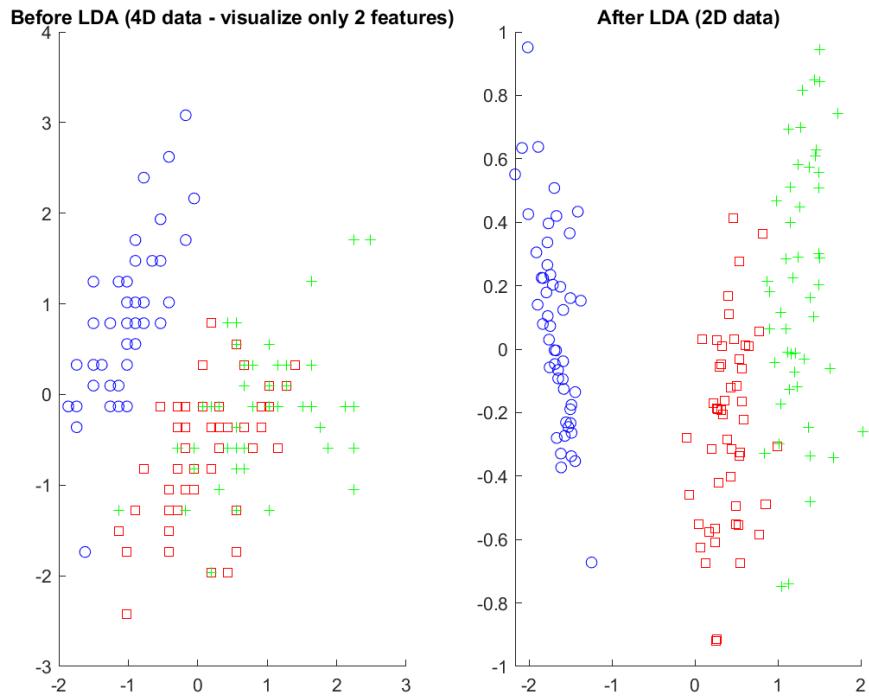
end
```

H κλήση και το figure ακολουθούν:

```
% Project the data on the direction of the two dimensional v
[meas_reduced] = projectDataLDA(meas_norm, v);

IRIS1_reduced = meas_reduced(iris_labels == 0, :); %Samples of
    Class 0
IRIS2_reduced = meas_reduced(iris_labels == 1, :); %Samples of
    Class 1
IRIS3_reduced = meas_reduced(iris_labels == 2, :); %Samples of
    Class 2

% Visualize the sample dataset after LDA is applied
% Use different color/symbol for each class
subplot(1, 2, 2)
title('After LDA (2D data)')
hold on
plot(IRIS1_reduced(:, 1), IRIS1_reduced(:, 2), 'bo');
plot(IRIS2_reduced(:, 1), IRIS2_reduced(:, 2), 'rs');
plot(IRIS3_reduced(:, 1), IRIS3_reduced(:, 2), 'g+');
hold off
```



Σχολιασμός αποτελεσμάτων:

Η μέθοδος LDA λειτουργησε σωστά και τα δεδομένα, παρότι μειώθηκαν σε 2D, φαίνονται διαχωρισμένα ανά την κλάση στην οποία ανήκουν.

Θέμα 4: Bayes

1)

Για την εξίσωση του συνόρου απόφασης θα χρησιμοποιηθεί η ΣΠΙΙ της πολυδιάστατης κανονικής κατανομής, αφού τα δείγματα είναι 2D:

$$f(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$

όπου:

- \mathbf{x} είναι ένας n -διάστατο διάνυσμα που αναπαριστά τις τυχαίες μεταβλητές,
- μ είναι το διάνυσμα των μέσων τιμών,
- Σ είναι ο πίνακας συνδιασποράς,
- $|\Sigma|$ συμβολίζει την ορίζουσα του πίνακα της συνδιασποράς.

Βάσει του συνδέσμου [εδώ](#) και με τη βοήθεια της Matlab:

Ορίστηκαν οι παράμετροι του προβλήματος:

```
%Natural parameters
mu1 = [2; 3];
sigma1 = [2 0.5; 0.5 1];
mu2 = [4; 4];
sigma2 = [1.5 -0.3; -0.3 0.8];
```

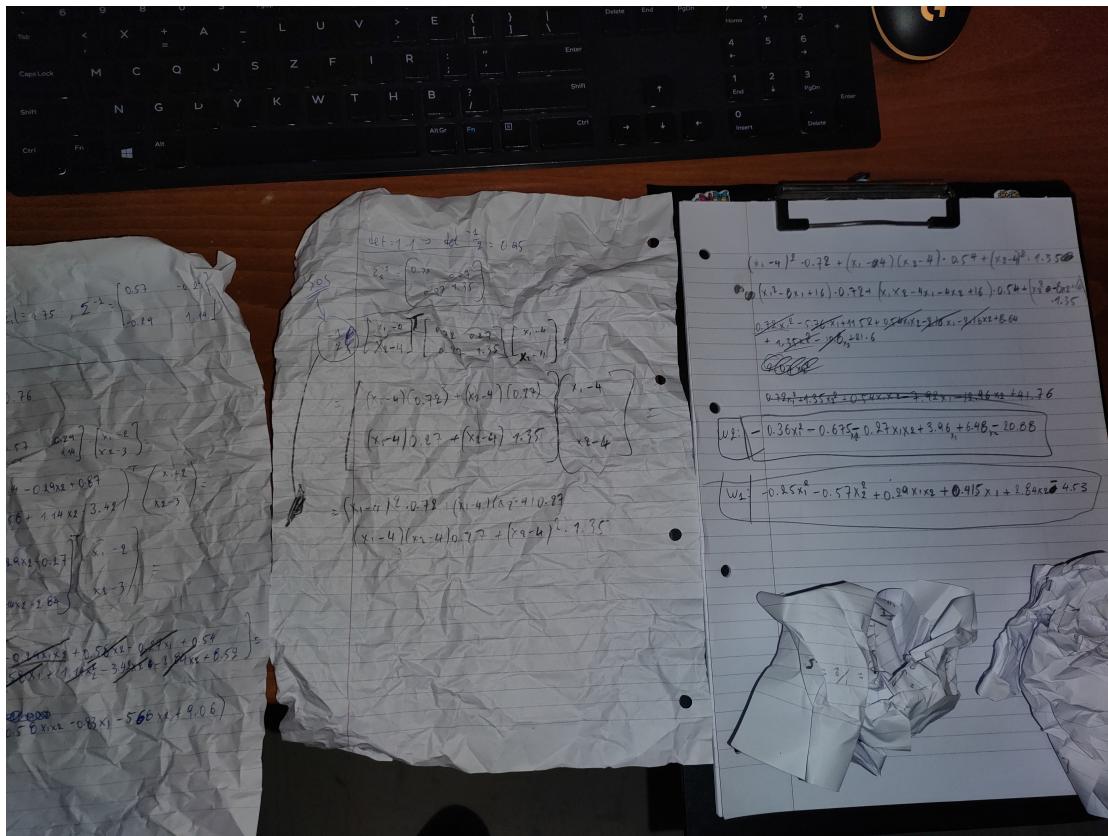


Figure 4.1: Τα μαθηματικά στο χαρτί απέτυχαν στις πράξεις.

Και οι γκαουσιανές (ανώνυμες) συναρτήσεις:

```
%Gaussians as anonymous functions
gauss1 = @(x1, x2)
    exp(-0.5*([x1;x2]-mu1)'*sigma1*([x1;x2]-mu1))/(2*pi*sqrt(det(sigma1)));
gauss2 = @(x1, x2)
    exp(-0.5*([x1;x2]-mu2)'*sigma2*([x1;x2]-mu2))/(2*pi*sqrt(det(sigma2)));
[x1, x2] = meshgrid(-5:0.2:10); %generate grid
g1 = arrayfun(gauss1, x1, x2); %apply functions to grid points
g2 = arrayfun(gauss2, x1, x2);
```

2, 3)

Για το 2 και 3 ερώτημα, σχεδιάστηκαν σε ένα figure οι καμπύλες των δύο κατανομών, αλλά και τα σύνορα απόφασης για κάθε ένα από τα ζεύγη πιθανοτήτων της εκφώνησης:

```

figure;
for p1 = [.1, .25, .5, .75, .9]
    p2 = 1 - p1;

    surf(x1, x2, g1, 'FaceColor', [0.9 0.2 0.2]); %plot
    hold on
    surf(x1, x2, g2, 'FaceColor', [0.2 0.9 0.2]);

    gdiff = p1 * g1 - p2 * g2; %get difference points

    C = contours(x1, x2, gdiff, [0,0]); %get points where contour is
        zero

    %get separate coordinates of contour
    my_range = 2:107;
    x1s = C(1, my_range);
    x2s = C(2, my_range);

    % Fit a quadratic polynomial to the data
    p = polyfit(x1s, x2s, 2);

    % Print the resulting polynomial equation
    fprintf('p1 = %.2f) The fitted function is: %.2fx^2 + %.2fx +
        %.2f\n', p1, p(1), p(2), p(3));

    %interpolate contour on gaussian 1
    boundry = interp2(x1, x2, g1, x1s, x2s);

    %plot it
    line(x1s, x2s, boundry, 'Color', 'k', 'LineWidth', 2);
end

```

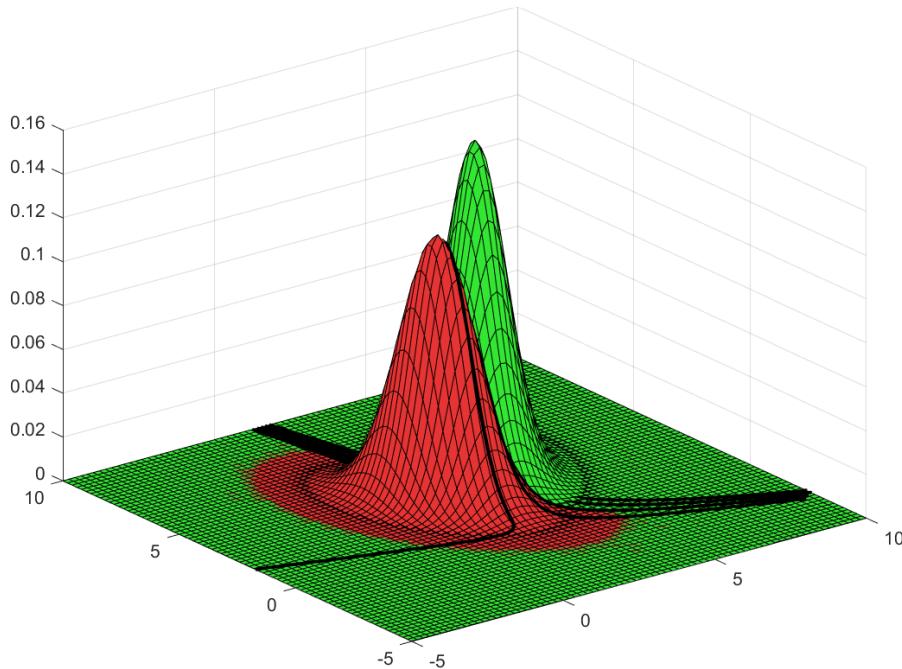


Figure 4.2: Διαφορετικοί πίνακες συνδιασποράς.

Σχολιασμός αποτελεσμάτων:

Τα σύνορα απόφασης είναι μη γραμμικά και μάλιστα quadratic, το οποίο οφείλεται στη μη γραμμικότητα του προβλήματος, αφού η discriminant function είναι:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma_i|) + \ln(P(C_i))$$

Αυτό φαίνεται και στο αποτέλεσμα που τυπώνεται στην κονσόλα:

(p1 = 0.10) The fitted function is: $-0.23x^2 + 0.05x + 4.63$
 (p1 = 0.25) The fitted function is: $0.35x^2 + -4.96x + 14.35$
 (p1 = 0.50) The fitted function is: $0.31x^2 + -4.70x + 14.94$
 (p1 = 0.75) The fitted function is: $0.28x^2 + -4.41x + 15.22$

(p1 = 0.90) The fitted function is: 0.59x^2 + -8.27x + 25.32

Επιπλέον, όσο μεγαλώνει η διαφορά των δύο πιθανοτήτων, τόσο μετακινείται το σύνορο απόφασης πιο κοντά στην κλάση με την μικρότερη πιθανότητα.

4)

Ομοίως για ίδιους πίνακες συνδιασποράς (ο κώδικας δεν αλλάζει παρά μόνο οι παράμετροι):

```
%Natural parameters
sigma = [1.2 .4; .4 1.2];
mu1 = [2; 3];
sigma1 = sigma;
mu2 = [4; 4];
sigma2 = sigma;
```

Σχολιασμός αποτελεσμάτων:

Για τις πιθανότητες ισχύει το ίδιο με πριν.

Όσον αφορά τη γραφικότητα των συνόρων απόφασης, οφείλεται στο ότι έχουμε **ίδιους** πίνακες συνδιασποράς, γεγονός το οποίο επιφέρει μία απλοποίηση στη discriminant function.

Στην κονσόλα τώρα τυπώνεται:

(p1 = 0.10) The fitted function is: -1.40x + 6.60
 (p1 = 0.25) The fitted function is: -1.40x + 7.15
 (p1 = 0.50) The fitted function is: -1.40x + 7.70
 (p1 = 0.75) The fitted function is: -1.40x + 8.25
 (p1 = 0.90) The fitted function is: -1.40x + 8.80

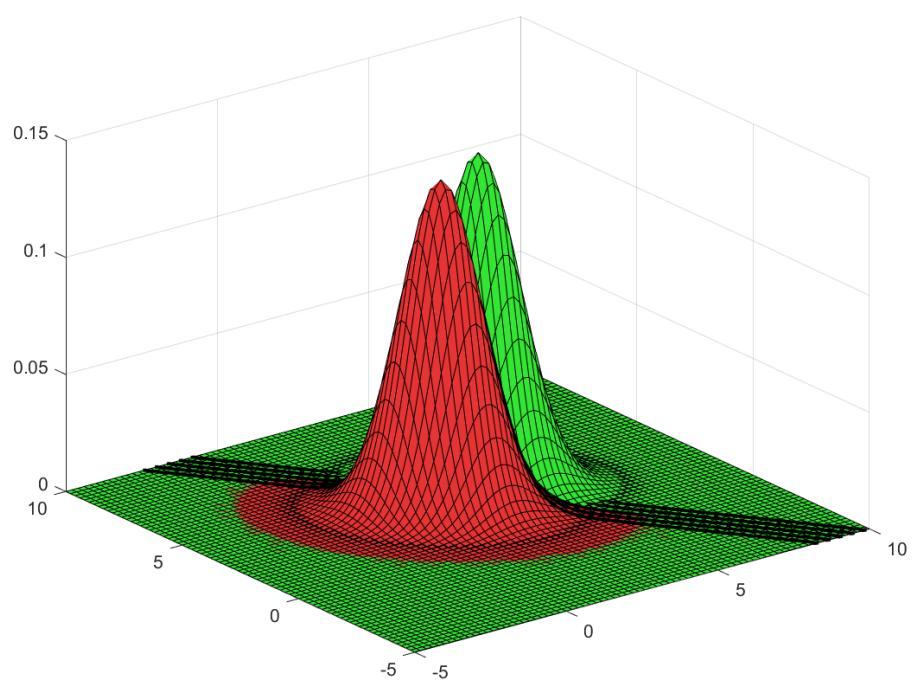


Figure 4.3: Τδιοι πίνακες συνδιασποράς.

Θέμα 5:

Εξαγωγή χαρακτηριστικών και Bayes Classification

Στο αρχείο requirements.txt βρίσκονται τα dependencies για τον κώδικα Python.

Επίσης, σημειώνεται ότι φορτώθηκαν όλα τα δεδομένα:

```
nTrainSamples = None # specify 'None' if you want to read the whole
                      file
nTestSamples = None # specify 'None' if you want to read the whole
                     file
```

a)

Αρχικά, υλοποιήθηκε η συνάρτηση aspect_ratio:

```
# Function to calculate aspect ratio
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]) - 1
max_row = np.max(nonzero_pixels[0]) + 1
min_col = np.min(nonzero_pixels[1]) - 1
max_col = np.max(nonzero_pixels[1]) + 1

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

# Calculate aspect ratio
aspect_ratio = width / height

return aspect_ratio

```

To aspect ratio θα χρησιμοποιηθεί ως πρώτο feature.

Ο κώδικας στη main έχει ως εξής:

```

# a)
# Calculate aspect ratio as the first feature
df_train['aspect_ratio'] = data_train.apply(aspect_ratio, axis=1)

# Find and print the max and min aspect ratio for labels 1 and 2
max_1 = df_train[df_train['label'] == 1]['aspect_ratio'].max()
max_2 = df_train[df_train['label'] == 2]['aspect_ratio'].max()
min_1 = df_train[df_train['label'] == 1]['aspect_ratio'].min()
min_2 = df_train[df_train['label'] == 2]['aspect_ratio'].min()
print(max_1, max_2, min_1, min_2)

```

Η μέγιστη και η ελάχιστη τιμή για κάθε κλάση (1 ή 2) είναι:

1.3125 2.1 0.14285714285714285 0.42857142857142855

b)

Για την επαλήθευση της σωστής λειτουργίας της παραπάνω συνάρτησης, τυπώνονται 10 τυχαία δείγματα μαζί με ένα παραλληλόγραμμο στα όρια του aspect ratio:

```
# b)
# Draw 10 sample images from the training data to make sure
# aspect ratio is correct
for sample in range(10):
    sample_image = data_train.iloc[sample].values.reshape(28, 28)
    visualize_bounding_box(sample_image)
```

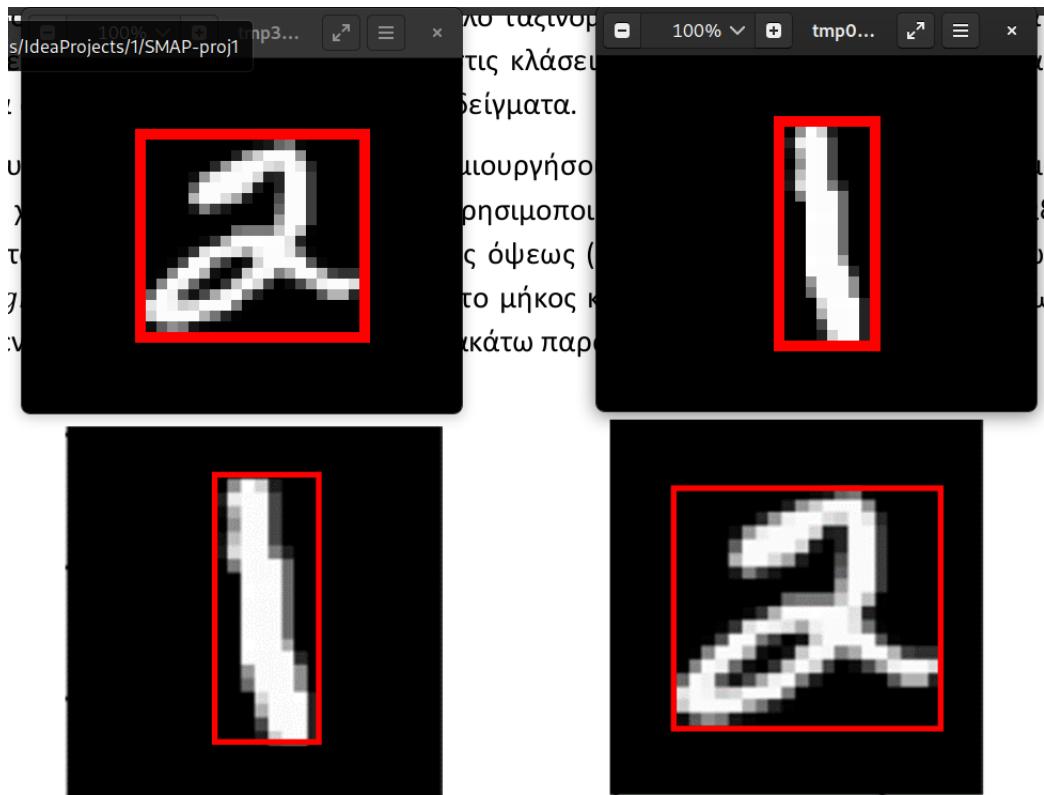


Figure 5.1: Πάνω: αποτελέσματα του χώδικα, κάτω: εκφώνηση

c)

Η κανονικοποίηση που έγινε στα χαρακτηριστικά είναι min-max scaling:

$$X_{\text{scaled}} = \min + (\max - \min) \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

με $\min = -1$ και $\max = +1$ για το συγκεκριμένο πρόβλημα.

Ο κώδικας:

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

Και η εφαρμογή του πάνω στα χαρακτηριστικά:

```
df_train['aspect_ratio'] = min_max_scaling(df_train['aspect_ratio'])
```

Με ελάχιστες εξαιρέσεις, οι Machine Learning αλγόριθμοι δεν έχουν καλή απόδοση όταν τα αριθμητικά χαρακτηριστικά έχουν πολύ διαφορετικές κλίμακες.

Σε αυτό βοηθάει η κανονικοποίηση (που θα εφαρμοστεί ομοίως και στα επόμενα χαρακτηριστικά).

d, e)

Ακολουθεί η μέθοδος train της κλάσης MyBayesClassifier:

```
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.values) / len(X)

        # Calculate mean and covariance for the class
        # Adding a small value to the covariance for numerical
        # stability
        self.class_stats[class_label] = {
            'mean': np.mean(X_class.values, axis=0),
            'cov': np.cov(X_class, rowvar=False)
}

```

Η παραπάνω συνάρτηση δέχεται ως είσοδο ένα DataFrame με features και μία σειρά από τα αντίστοιχα labels τους και υπολογίζει τις a priori πιθανότητες των labels καθώς και τα στατιστικά των features.

```
A priori probabilities: {1: 0.5308661417322834, 2:
0.46913385826771653}
```

f)

Έχοντας υπολογίσει τα στατιστικά για κάθε κλάση, μπορεί να υλοποιηθεί η συνάρτηση predict που θα προβλέπει σε ποια κλάση ανήκει ένα test δείγμα.

Πηγαίνοντας ανάποδα, το πρώτο πράγμα που πρέπει να γίνει είναι ένα single instance να μπορεί να ταξινομηθεί και, για να γίνει αυτό, χρειάζεται η συνάρτηση _calculate_likelihood(self, x, mean, cov) που είναι μία helper για να υπολογιστεί για κάθε δείγμα η πιθανοφάνειά του στο να ανήκει σε μία κλάση τα στατιστικά της οποίας δίνονται στις παραμέτρους.

Αναλυτικά:

```

def _calculate_likelihood(self, x, mean, cov):
    """
    Calculate the Gaussian likelihood of the data x given class
    statistics.

    Args:
        x (pd.Series): Features of the data point.
        mean (pd.Series): Mean feature for the class.
        cov (pd.DataFrame): Covariance matrix for the class.

    Returns:
        float: The likelihood value.
    """
    if mean.shape[0] > 1:
        likelihood = multivariate_normal.pdf(x, mean=mean, cov=cov)
    else:
        likelihood = norm.pdf(x, loc=mean, scale=cov)
    return likelihood

```

Τονίζεται ότι σε αυτό το σημείο ακόμα δεν έχει ληφθεί υπόψη η a priori πιθανότητα για την κλάση.

Ακολουθεί η συνάρτηση `_predict_instance(self, x)` που υπολογίζει σε ποια κλάση είναι πιο πιθανό - με βάση τον ταξινομητή - να αντιστοιχεί το δείγμα που δίδεται στις παραμέτρους:

```

def _predict_instance(self, x):
    """
    Private helper to predict the class for a single instance.

    Args:
        x (pd.Series): A single data point's features.

    Returns:
        The predicted class label.
    """
    posteriors = []

    # Calculate the posterior probability for each class
    for class_label in self.classes_:
        sample_mean, sample_std =
            self.class_stats[class_label].values()

```

```

    p = self._calculate_likelihood(x, sample_mean, sample_std)
    posteriors.append((class_label, p *
        self.class_priors[class_label]))

    # Choose the class with the highest posterior probability
    prediction = max(posteriors, key=lambda z: z[1])
    return prediction[0]

```

Η συνάρτηση predict δεν κάνει τίποτα άλλο πέρα από εφαρμογή της `_predict_instance(self, x)` σε όλα τα δείγματα που της δίνονται (σε αντικείμενο DataFrame).

Επιστρέφει, λοιπόν, μία μήτρα από τις προβλέψεις:

```

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 = X.apply(self._predict_instance, axis=1)
    return np.array(predictions)

```

Φυσικά, πρέπει να γίνουν και οι απαραίτητες ενέργειες στο test dataset (υπολογισμός aspect ratio και κανονικοποίηση).

Η εφαρμογή όλων των παραπάνω στην main:

```

# f)
# Predict on the test samples (for the given feature set)
df_test['aspect_ratio'] = data_test.apply(aspect_ratio, axis=1)
df_test['aspect_ratio'] = min_max_scaling(df_test['aspect_ratio'])
test_data = df_test[features]
predictions = classifier.predict(test_data)

```

g)

Μία μετρική για την εκτίμηση της απόδοσης ακρίβειας της ταξινόμησης είναι το accuracy:

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Πλήθος σωστών προβλέψεων}}{\text{Συνολικό πλήθος προβλέψεων}} \\ &= \frac{TP + TN}{TP + TN + FP + FN} \end{aligned}$$

όπου:

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

Απαραίτητο:

```
from sklearn.metrics import accuracy_score
```

Ο αντίστοιχος κώδικας:

```
# g)
# Calculate accuracy as an example of validation
accuracy = accuracy_score(target_test, predictions)
print(f"Classification accuracy: {round(accuracy * 100, 4)}%")
```

Εκτύπωση αποτελέσματος:

```
Classification accuracy: 89.0632%
```

h)

Για αυτό το ερώτημα προστέθηκε ακόμα ένα χαρακτηριστικό, τα foreground_pixels.

H συνάρτηση που τα υπολογίζει:

```
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.nonzero(img)[0].size

        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
```

Kai εδώ βρίσκεται ο κώδικας της main:

```
# h)
# Calculate the number of non-zero pixels as the second feature
df_train['fg_pixels'] = data_train.apply(foreground_pixels, axis=1)
df_train['fg_pixels'] = min_max_scaling(df_train['fg_pixels'])
classifier = MyBayesClassifier()
features = ['aspect_ratio', 'fg_pixels']
trainData = df_train[features]
classifier.train(trainData, target_train)
assert (sum(classifier.class_priors.values()) == 1)
df_test['fg_pixels'] = data_test.apply(foreground_pixels, axis=1)
```

```
df_test['fg_pixels'] = min_max_scaling(df_test['fg_pixels'])
test_data = df_test[features]
predictions = classifier.predict(test_data)
accuracy = accuracy_score(target_test, predictions)
print(f"Classification accuracy: {round(accuracy * 100, 4)}%")
```

Η απόδοση του μοντέλου αυξήθηκε κατά περίπου 4.5%:

Classification accuracy: 93.401%

i)

Τέλος, προστέθηκε ακόμα ένα χαρακτηριστικό, το 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

    x_center = 0
    y_center = 0
    total_mass = 0
    for y in range(cols):
        for x in range(rows):
            intensity = img[y, x]
            if intensity > 0:
                x_center += x * intensity
                y_center += y * intensity
                total_mass += intensity
```

```

x_center = x_center / total_mass
y_center = y_center / total_mass

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

return centroid

```

Και εδώ βρίσκεται ο κώδικας της main:

```

# i)
# Calculate the centroid feature as the third feature
df_train['centroid'] = data_train.apply(calculate_centroid, axis=1)
df_train['centroid'] = min_max_scaling(df_train['centroid'])
classifier = MyBayesClassifier()
features = ["aspect_ratio", "fg_pixels", "centroid"]
trainData = df_train[features]
classifier.train(trainData, target_train)
assert (sum(classifier.class_priors.values()) == 1)
df_test['centroid'] = data_test.apply(calculate_centroid, axis=1)
df_test['centroid'] = min_max_scaling(df_test['centroid'])
test_data = df_test[features]
predictions = classifier.predict(test_data)
accuracy = accuracy_score(target_test, predictions)
print(f"Classification accuracy: {round(accuracy * 100, 4)}%")

```

Η απόδοση του μοντέλου:

Classification accuracy: 93.4933%

Σχολιασμός σχέσης accuracy - χαρακτηριστικών:

Από το ένα στα δύο χαρακτηριστικά, η απόδοση βελτιώθηκε αισθητά. Όταν προστέθηκε και το τρίτο, όμως, έμεινε σχεδόν σταθερή.

Τα πρώτα δύο χαρακτηριστικά μεν είναι έυστοχα και βοήθησαν την ταξινόμηση, το τρίτο, δε, όχι τόσο.

Αυτό φαίνεται αν εξεταστεί το τρίτο χαρακτηριστικό (centroid) ως μοναδικό:

Classification accuracy: 52.838%

Ενώ, θεωρητικά, είναι ένα χρήσιμο χαρακτηριστικό, η αλήθεια είναι ότι όλα τα νούμερα θα έχουν περίπου ίδιο centroid, το οποίο φαίνεται κι από την αφαίρεση των μέσων τιμών της κλάσης 1 και 2 (-0.08) για το συγκεκριμένο χαρακτηριστικό, σε συνάρτηση με την αφαίρεση των πινάκων συνδιασποράς τους (0.00509).

Αναλυτικά:

$$\begin{pmatrix} -0.47 \\ -0.59 \\ \mathbf{-0.08} \end{pmatrix}$$

$$\begin{pmatrix} 0.00498 & 0.01476 & -0.00169 \\ 0.01476 & -0.03503 & 0.00160 \\ -0.00169 & 0.00160 & \mathbf{0.00509} \end{pmatrix}$$

j)

Ομοίως, για τρεις κλάσεις, παρατηρείται μία βελτίωση της απόδοσης προσθέτοντας το δεύτερο χαρακτηριστικό, ενώ μία μηδαμινή διαφορά προσθέτοντας το τρίτο.

Classification accuracy: 67.0759%

Classification accuracy: 70.3179%

Classification accuracy: 70.4753%

Σχολιασμός δύο έναντι τριών κλάσεων:

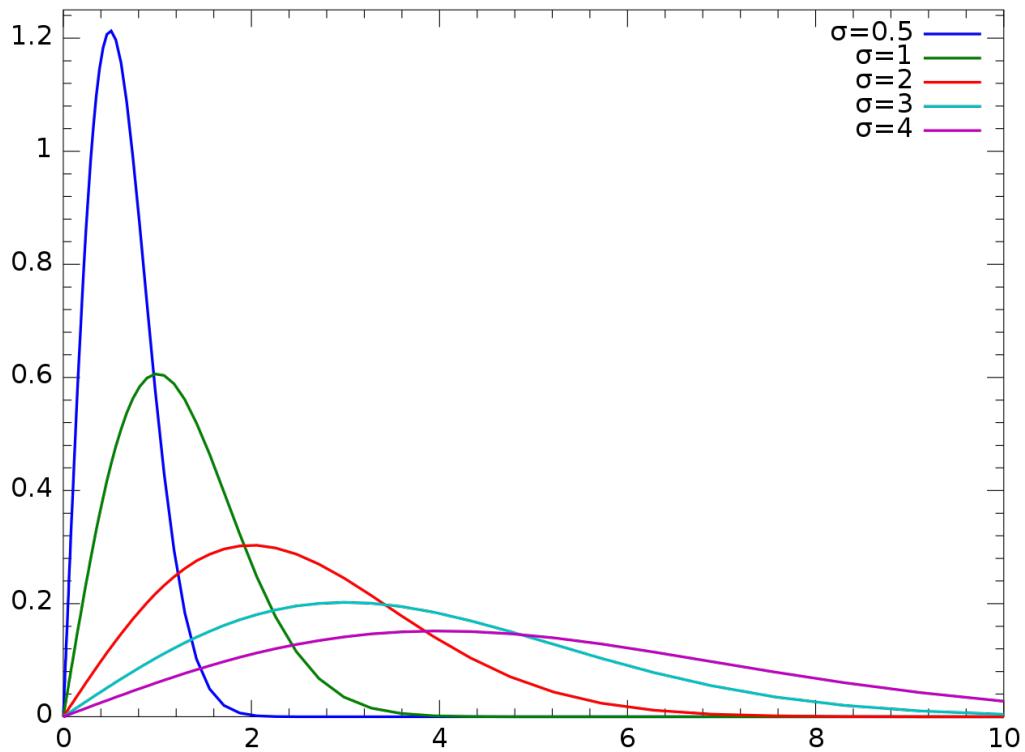
Η πρόσθεση άλλης μίας κλάσης επιφέρει πτώση της απόδοσης κατά περίπου 20%.

Ο ταξινομητής δυσκολεύεται να διαχρίνει ποιο νούμερο είναι πραγματικά αφού έχει τρεις αντί για δύο επιλογές.

Θέμα 6: Minimum risk

Οι κλάσεις έχουν ίδιες *a priori* πιθανότητες, όρα παραλείπονται στις εξισώσεις.

Η κατανομή Rayleigh έχει την εξής μορφή για διάφορες τιμές της τυπικής απόκλισης σ :



Γίνεται εξίσωση των ΣΠΠ για κάθε κλάση λαμβάνοντας υπόψη τον πίνακα ρίσκου:

$$\begin{aligned}
 0.5p(x|\omega_1) &= 1.0p(x|\omega_2) \\
 \frac{x}{\sigma_1^2} e^{-\frac{x^2}{2\sigma_1^2}} &= \frac{2x}{\sigma_2^2} e^{-\frac{x^2}{2\sigma_2^2}} \\
 \frac{1}{1^2} e^{-\frac{x^2}{2 \cdot 1^2}} &= \frac{2}{2^2} e^{-\frac{x^2}{2 \cdot 2^2}} \\
 e^{-\frac{x^2}{2}} &= \frac{1}{2} e^{-\frac{x^2}{8}} \\
 2e^{-\frac{x^2}{2}} &= e^{-\frac{x^2}{8}} \\
 ln(2) - \frac{x^2}{2} &= -\frac{x^2}{8} \\
 \frac{3}{8}x^2 &= ln(2) \\
 x^2 &= \frac{8}{3}ln(2) \\
 x &\approx 1.36
 \end{aligned}$$

Τα παραπάνω ισχύουν για θετικές τιμές x . Για αρνητικές, δεν έχει σημασία γιατί και οι δύο πιθανότητες είναι μηδενικές, οπότε αυθαίρετα μπορεί να αποφασίζει τη μία ή την άλλη κλάση.

Συνεπώς:

$$\boxed{\mathbf{x_0} \approx 1.36}$$

Πηγές

<https://www.geeksforgeeks.org/problem-solving-on-scatter-matrix/>

https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix

<https://www.linkedin.com/advice/0/what-key-differences-between-pca-lda-dimensionality-7memc>

<http://ucl-cs-grad.github.io/matlabgrad/lectures.html>

<https://saturncloud.io/blog/what-is-sklearn-pca-explained-variance-and-explained-variance-ratio-difference/>

<https://www.mathworks.com/matlabcentral/answers/361334-hi-i-want-to-calculate-the-decision-boundary-in-bayes-estimator-can-anyone-help-me-with-it>

<https://docs.python.org/3/library/venv.html>

<https://www.geeksforgeeks.org/python-virtual-environment/>

https://www.w3schools.com/python/pandas/pandas_dataframes.asp

<https://stackoverflow.com/questions/15317822/calculating-covariance-with-python-and-numpy>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html>

<https://www.geeksforgeeks.org/difference-between-variance-and-standard-deviation/>

https://www.w3schools.com/python/python_dictionaries_access.asp

<https://stackoverflow.com/questions/47914154/what-is-the-centroid-of-an-image>

https://en.wikipedia.org/wiki/Rayleigh_distribution

<https://stackoverflow.com/questions/455612/limiting-floats-to-two-decimal-points>