Question 1

$\Rightarrow$ Bayes classifiers minimize conditional risk defined as:

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|w_j)\, P(w_j|x) \quad \text{for } i=1,\dots,c$$

(multiclass case).

Based on the $\overset{\text{(conditional loss)}}{\lambda(\alpha_i|w_j)}$ definition given,

$$R(\alpha_i|x) = \sum_{\substack{j=1 \\ j\neq i}}^{c} \lambda_s . P(w_j|x) \quad \text{for } i=1,\dots,c.$$

$$= \lambda_s \underbrace{\sum_{\substack{j=1 \\ j\neq i}}^{c} P(w_j|x)}$$

$$= \lambda_s . \left[1-P(w_i|x)\right]. \quad \text{for } i=1,\dots,c.$$

For $i=c+1$; $R(\alpha_{c+1}|x)=\lambda_r$.

$\Rightarrow$ Minimum risk is achieved if we decide w_i if $R(\alpha_i|x) \leq R(\alpha_{c+1}|x)$...

$$R(\alpha_i|x) \leq R(\alpha_{c+1}|x)$$

$$\lambda_s . \left[1-P(w_i|x)\right] \leq \lambda_r$$

$$\boxed{P(w_i|x) \geq 1 - \frac{\lambda_r}{\lambda_s}}$$ decide w_i and reject otherwise.

* if $\lambda_r=0$, then we always reject.
* if $\lambda_r > \lambda_s$, we will never reject.

Question 2

Resulting figures are below, codes are attached at the end.

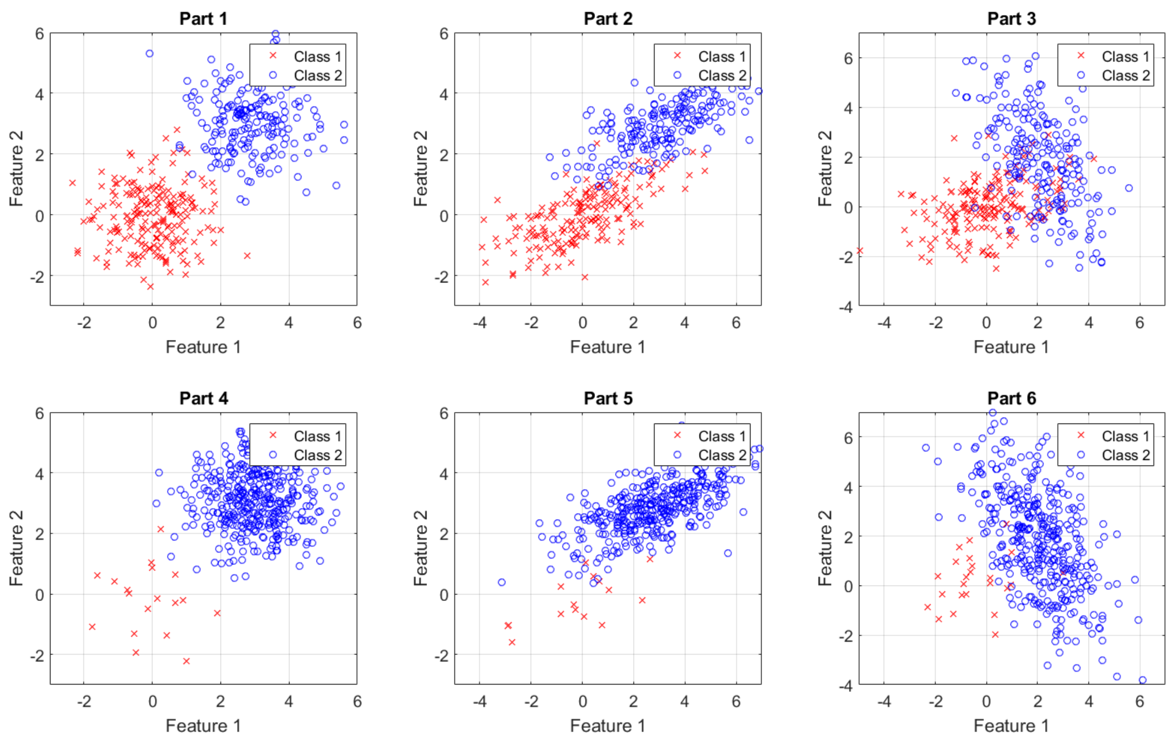

Figure 1: Generated data samples for each six condition.

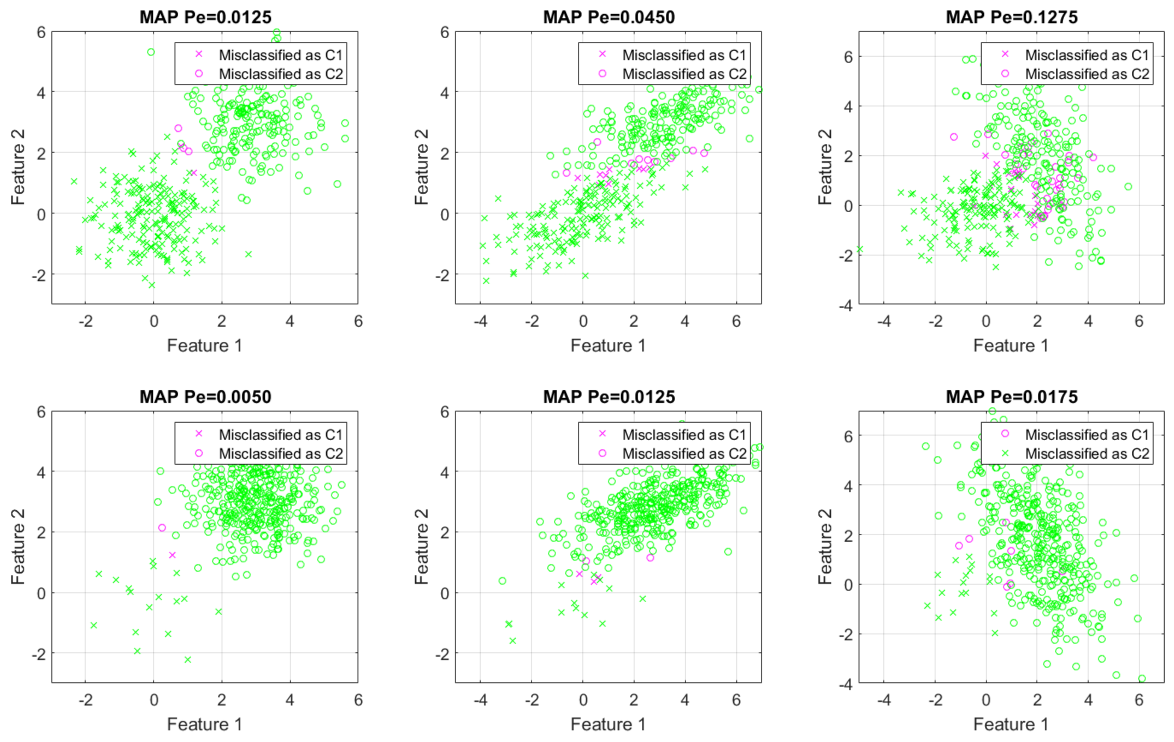

Figure 2: MAP classification output indicating misclassified samples.

Question 3

Resulting figures are below, codes are attached at the end.

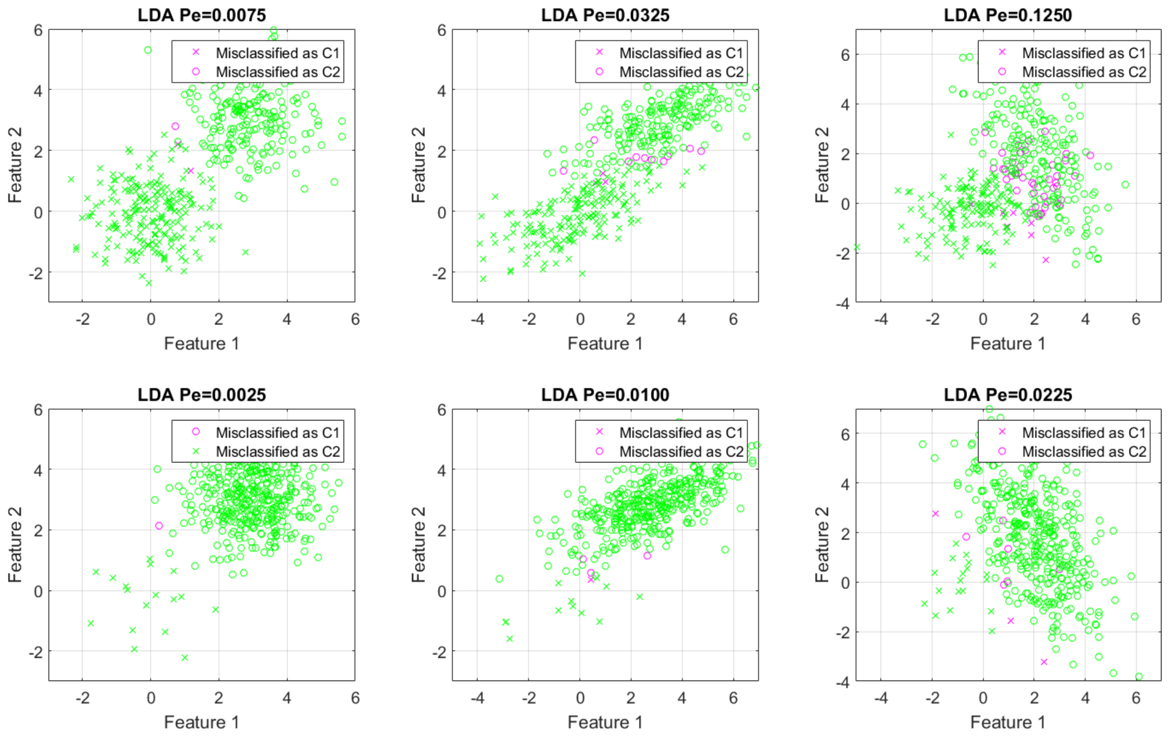

Figure 3: LDA classification output indicating misclassified samples.

```matlab
%% EECE5644 - Homework 2 - Question 2 and 3
clear all; close all; clc;
rng('default');

%% Dataset 1
nSamples = 400;
mu{1} = [0,0]; mu{2} = [3,3];
sigma{1} = eye(2); sigma{2} = eye(2);
prior = [0.5; 0.5];
xMin = -3; xMax = 6;
yMin = -3; yMax = 6;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,1);
titleString = 'Part 1';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,1);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

% LDA Classification and Visualization for Question 3
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
figure(3); subplot(2,3,1);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

%% Dataset 2
nSamples = 400;
mu{1} = [0,0]; mu{2} = [3,3];
sigma{1} = [3, 1; 1, 0.8]; sigma{2} = [3, 1; 1, 0.8];
prior = [0.5; 0.5];
xMin = -5; xMax = 7;
yMin = -3; yMax = 6;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,2);
titleString = 'Part 2';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,2);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

% LDA Classification and Visualization for Question 3
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
```

```
figure(3); subplot(2,3,2);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

%% Dataset 3
nSamples = 400;
mu{1} = [0,0]; mu{2} = [2,2];
sigma{1} = [2 0.5; 0.5 1]; sigma{2} = [2 -1.9; -1.9 5];
prior = [0.5; 0.5];
xMin = -5; xMax = 7;
yMin = -4; yMax = 7;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,3);
titleString = 'Part 3';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,3);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

% LDA Classification and Visualization for Question 3
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
figure(3); subplot(2,3,3);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

%% Dataset 4
nSamples = 400;
mu{1} = [0,0]; mu{2} = [3,3];
sigma{1} = eye(2); sigma{2} = eye(2);
prior = [0.05; 0.95];
xMin = -3; xMax = 6;
yMin = -3; yMax = 6;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,4);
titleString = 'Part 4';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,4);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

% LDA Classification and Visualization for Question 3
```

```
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
figure(3); subplot(2,3,4);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

%% Dataset 5
nSamples = 400;
mu{1} = [0,0]; mu{2} = [3,3];
sigma{1} = [3, 1; 1, 0.8]; sigma{2} = [3, 1; 1, 0.8];
prior = [0.05; 0.95];
xMin = -5; xMax = 7;
yMin = -3; yMax = 6;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,5);
titleString = 'Part 5';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,5);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

% LDA Classification and Visualization for Question 3
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
figure(3); subplot(2,3,5);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);

%% Dataset 6
nSamples = 400;
mu{1} = [0,0]; mu{2} = [2,2];
sigma{1} = [2 0.5; 0.5 1]; sigma{2} = [2 -1.9; -1.9 5];
prior = [0.05; 0.95];
xMin = -5; xMax = 7;
yMin = -4; yMax = 7;

nClass = numel(mu);
[data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior);
figure(1); subplot(2,3,6);
titleString = 'Part 6';
plotSamples(data, classIndex, nClass, titleString);
axis([xMin xMax yMin yMax]);

% MAP Classification and Visualization for Question 2
[ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex, mu, sigma,
nSamples, prior);
figure(2); subplot(2,3,6);
plotDecision(data,ind01MAP,ind10MAP,ind00MAP,ind11MAP);
title(sprintf('MAP Pe=%.4f',pEminERM), 'FontSize', 18);
axis([xMin xMax yMin yMax]);
```

```
% LDA Classification and Visualization for Question 3
[ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex, mu, sigma,
nSamples, prior);
figure(3); subplot(2,3,6);
plotDecision(data,ind01LDA,ind10LDA,ind00LDA,ind11LDA);
title(sprintf('LDA Pe=%.4f',pEminLDA), 'FontSize', 18);
axis([xMin xMax yMin yMax]);
```

```
function [data, classIndex] = generateGaussianSamples(mu, sigma, nSamples, prior)
% Function to simulate data from k Gaussian densities (1 for each class) in d dimensions.
%
% INPUTS:
%   mu          - cell with the class dependent d-dimensional mean vector
%   sigma       - k-by-1 cell with the class dependent d-by-d covariance matrix
%   nSamples    - scalar indicating number of samples to be generated
%   prior       - k-by-1 vector with class dependent mean
%
% OUTPUTS:
%   data        - nSamples-by-d array with the simulated data distributed along the rows
%   classIndex  - vector of length nSamples with the class index for each datapoint

if sum(prior) ~= 1
    error('Priors should add to one!');
end

% First, sample the respective class indexes. We can do this by generating uniformly
% distributed numbers from 0 to 1 and using thresholds based on the prior probabilities

classTempScalar = rand(nSamples, 1);
priorThresholds = cumsum([0; prior]);
nClass = numel(mu);
data = cell(nClass, 1);
classIndex = cell(nClass, 1);

for idxClass = 1:nClass
    nSamplesClass = nnz(classTempScalar>=priorThresholds(idxClass) &
classTempScalar<priorThresholds(idxClass+1));
    % Generate samples according to class dependent parameters
    data{idxClass} = mvnrnd(mu{idxClass}, sigma{idxClass}, nSamplesClass);
    % Set class labels
    classIndex{idxClass} = ones(nSamplesClass,1) * idxClass;
end

data = cell2mat(data);
classIndex = cell2mat(classIndex);
end




function plotSamples(data, classIndex, nClass, titleString)
markerStrings = ['x','o']; colorString = ['r', 'b'];

for idxClass = 1:nClass
    dataClass = data(classIndex==idxClass,:);
    plot(dataClass(:,1), dataClass(:,2) , [colorString(idxClass)
markerStrings(idxClass)]);
    hold on;
end

hold off;
title(titleString, 'FontSize', 18);
xlabel('Feature 1', 'FontSize', 16);
ylabel('Feature 2', 'FontSize', 16);
legend({'Class 1', 'Class 2'});
grid on; box on;
set(gca, 'FontSize', 14);
```

```
function [ind01MAP,ind10MAP,ind00MAP,ind11MAP,pEminERM] = classifyMAP(data, classIndex,
mu, sigma, nSamples, prior)

% Expected Risk Minimization Classifier
discriminantScoreERM = log(evalGaussian(data',mu{2}',sigma{2}))-
log(evalGaussian(data',mu{1}',sigma{1}));

% MAP classifier (is a special case of ERM corresponding to 0-1 loss)
lambdaMAP = [0 1;1 0]; % 0-1 loss values yield MAP decision rule
gammaMAP = (lambdaMAP(2,1)-lambdaMAP(1,1))/(lambdaMAP(1,2)-lambdaMAP(2,2)) *
prior(1)/prior(2); % threshold for MAP
decisionMAP = (discriminantScoreERM >= log(gammaMAP));
ind00MAP = find(decisionMAP==0 & classIndex'==1); p00MAP =
length(ind00MAP)/sum(classIndex==1); % probability of true negative
ind10MAP = find(decisionMAP==1 & classIndex'==1); p10MAP =
length(ind10MAP)/sum(classIndex==1); % probability of false positive
ind01MAP = find(decisionMAP==0 & classIndex'==2); p01MAP =
length(ind01MAP)/sum(classIndex==2); % probability of false negative
ind11MAP = find(decisionMAP==1 & classIndex'==2); p11MAP =
length(ind11MAP)/sum(classIndex==2); % probability of true positive
pEminERM = [p10MAP,p01MAP]*[sum(classIndex==1),sum(classIndex==2)]'/nSamples; %
probability of error for MAP classifier, empirically estimated

function g = evalGaussian(x,mu,Sigma)
% Evaluates the Gaussian pdf N(mu,Sigma) at each column of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2);
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(Sigma)*(x-repmat(mu,1,N))),1);
g = C*exp(E);


function [ind01LDA,ind10LDA,ind00LDA,ind11LDA,pEminLDA] = classifyLDA(data, classIndex,
mu, sigma, nSamples, prior)

% Fisher LDA Classifer (using true model parameters)
Sb = (mu{1}'-mu{2}')*(mu{1}'-mu{2}')';
Sw = sigma{1} + sigma{2};
[V,D] = eig(inv(Sw)*Sb); % LDA solution satisfies alpha Sw w = Sb w; ie w is a
generalized eigenvector of (Sw,Sb)
[~,ind] = sort(diag(D),'descend');
wLDA = V(:,ind(1)); % Fisher LDA projection vector
yLDA = wLDA'*data'; % All data projected on to the line spanned by wLDA
wLDA = sign(mean(yLDA(find(classIndex==2)))-mean(yLDA(find(classIndex==1))))*wLDA; %
ensures class1 falls on the + side of the axis
discriminantScoreLDA = sign(mean(yLDA(find(classIndex==2)))-
mean(yLDA(find(classIndex==1))))*yLDA; % flip yLDA accordingly

% Estimate the ROC curve for this LDA classifier
[ROCLDA,tauLDA] = estimateROC(discriminantScoreLDA,classIndex');
probErrorLDA = [ROCLDA(1,:)',1-
ROCLDA(2,:)']*[sum(classIndex==1),sum(classIndex==2)]'/nSamples; % probability of error
for LDA for different threshold values
pEminLDA = min(probErrorLDA);   % minimum probability of error

ind = find(probErrorLDA == pEminLDA);
decisionLDA = (discriminantScoreLDA >= tauLDA(ind(1))); % use smallest min-error
threshold
ind00LDA = find(decisionLDA==0 & classIndex'==1); % true negatives
ind10LDA = find(decisionLDA==1 & classIndex'==1); % false positives
ind01LDA = find(decisionLDA==0 & classIndex'==2); % false negatives
ind11LDA = find(decisionLDA==1 & classIndex'==2); % true positives
```

```
function [ROC,tau] = estimateROC(discriminantScoreLDA,label)
% Generate ROC curve samples
Nc = [length(find(label==1)),length(find(label==2))];
sortedScore = sort(discriminantScoreLDA,'ascend');
tau = [sortedScore(1)-1,(sortedScore(2:end)+sortedScore(1:end-1))/2,sortedScore(end)+1];
% thresholds at midpoints of consecutive scores in sorted list
for k = 1:length(tau)
    decision = (discriminantScoreLDA >= tau(k));
    ind10 = find(decision==1 & label==1); p10 = length(ind10)/Nc(1); % probability of
false positive
    ind11 = find(decision==1 & label==2); p11 = length(ind11)/Nc(2); % probability of
true positive
    ROC(:,k) = [p10;p11];
end


function plotDecision(data,ind01,ind10,ind00,ind11)

plot(data(ind01,1),data(ind01,2),'xm'); hold on;  % false negatives
plot(data(ind10,1),data(ind10,2),'om'); hold on;  % false positives
plot(data(ind00,1),data(ind00,2),'xg'); hold on;
plot(data(ind11,1),data(ind11,2),'og'); hold on;

xlabel('Feature 1', 'FontSize', 16);
ylabel('Feature 2', 'FontSize', 16);
grid on; box on;
set(gca, 'FontSize', 14);
legend({'Misclassified as C1','Misclassified as C2'});
```