EECE5644: Assignment #1

Due on February 10, 2020

Professor Deniz Erdogmus

Anja Deric

GitHub Repo: https://github.com/AnjaDeric/MachineLearning

Problem 1

The mean and covariance matrix values given in problem 1 were used to first generate 10,000 samples pictured in Figure 1 below.

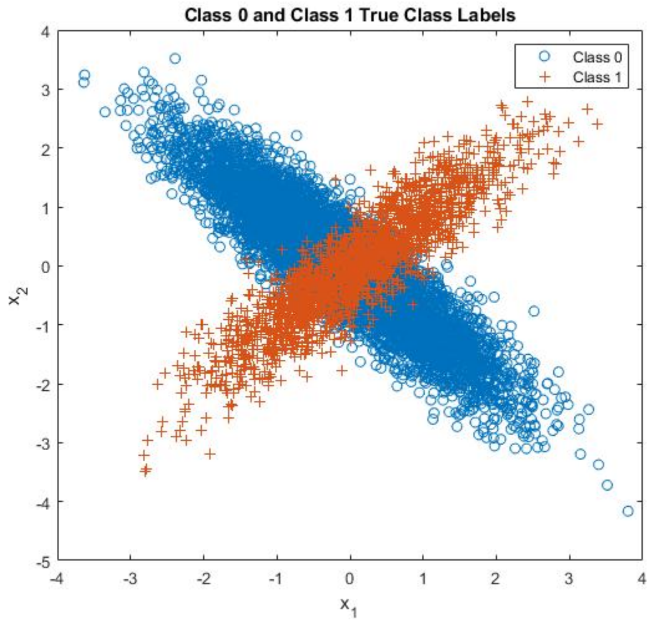

Figure 1: Problem 1 class distributions and true labels of data points

Summarized below are the class priors and conditional PDFs of the two classes, which were given in the problem:

Class 0

$$\mu_1 = \begin{bmatrix} -0.1 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix} \qquad P(L=0) = 0.8$$

Class 1

$$\mu_2 = \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \qquad \Sigma_2 = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \qquad P(L=1) = 0.2$$

These distributions were used for all remaining parts of Problem 1.

Part One

1. The minimum expected risk classification rule:

$$\frac{g(x|m_0, C_0)}{g(x|m_1, C_1)} \gtrless \frac{P(L=0)}{P(L=1)}(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1}) = \frac{0.8}{0.2}(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1})$$

$$(D=1) \qquad \frac{g(x|m_0, C_0)}{g(x|m_1, C_1)} \gtrless 4(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1}) = \gamma \qquad (D=0)$$

2. Figure 2 below displays the ROC curve generated after implementing the classifier from above, applying it to the 10,000 generated samples, and varying the threshold from 0 to ∞.

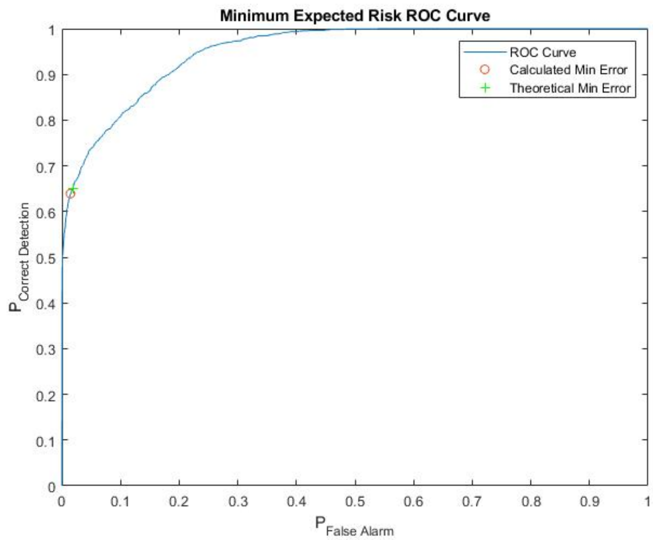

Figure 2: Problem 1 ROC Curve

3. Based on the 10,000 samples generated, the minimum probability of error was calculated to be 8.38% at the threshold value of $\gamma = 4.76$. This error and threshold value were calculated by finding the minimum error as the value of γ was changed from 0 to ∞. The orange circle in Figure 2 above marks this point.
To confirm this calculation, the theoretical minimum probabilty of error was also calculated at the threshold of $\gamma = 4$ ($from\ \frac{0.8}{0.2}$). This error was found to be 8.48%, which closely aligns with the values observed with the 10,000 data points. The green plus sign in Figure 2 above marks this point.
Figure 3 below summarizes these findings for Part 1 of Question 1.

```
Theoretical Results
Minimum probability of error: 8.48%
Threshold Value: 4.00

Calculated Results
Minimum probability of error: 8.38%
Threshold Value: 4.76
```

Figure 3: Problem 1, Part 1 Minimum Error

Part Two

In part 2 of Question 1, a similar process was repeated, but when evaluating the Gaussian distributions, the covariance matrix of both classes was assumed to be an identity matrix.

1. The Naive-Bayesian (NB) expected risk classification rule:

$$\frac{p_{NB}(x|L=1)}{p_{NB}(x|L=0)} \gtrless \frac{P(L=0)}{P(L=1)}(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1}) = \frac{0.8}{0.2}(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1})$$

$$(D=1) \qquad \frac{g(x|m_1, I)}{g(x|m_0, I} \gtrless 4(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1}) = \gamma \qquad (D=0)$$

2. Figure 4 below displays the ROC curve generated after implementing the NB classifier from above, applying it to the 10,000 generated samples, and varying the threshold from 0 to ∞.

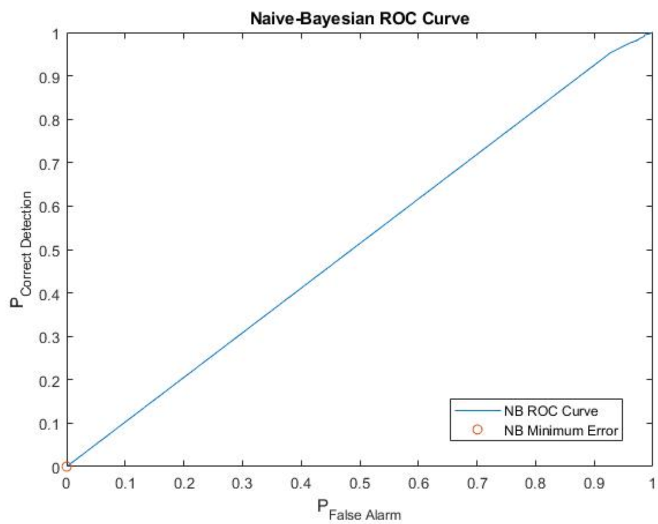

Figure 4: Problem 1 Naive-Bayesian ROC Curve

3. Based on the 10,000 samples generated, the minimum probability of error was calculated to be 20% at the threshold value of $\gamma = 2.07$. This error and threshold value were calculated by finding the minimum error as the value of γ was changed from 0 to ∞. The orange circle in Figure 4 above marks this point.
This result makes sense since assuming that the covariance matrix of both class PDFs is an identity matrix results in the two distributions overalapping. As a result, in the best case scenario, all data points in the smaller of the two classes (in this case, class 1), would be classified wrong, causing 20% error since class 1 has a 0.2 class prior.
The specific threshold value of 2.07 comes from the fact that the greatest threshold value that my code considered was 2.07, but if a greater threshold value was chosen, the same error of 20% would have been achieved. I only chose to go as far as the greatest discriminant score + 1, and 2.07 is the corresponding threshold value for that.

Part Three

In part 3 of Question 1, Fisher Linear Discriminant Analysis (LDA) was used to create a classifier and plot the ROC curve.

The LDA classification rule:

$$(D = 1) \qquad w_{LDA}^T x \gtrless \tau \qquad (D = 0)$$

Figure 5 below displays the ROC curve generated after implementing the LDA classifier from above, applying it to the 10,000 generated samples, and varying the threshold τ from -∞ to ∞.

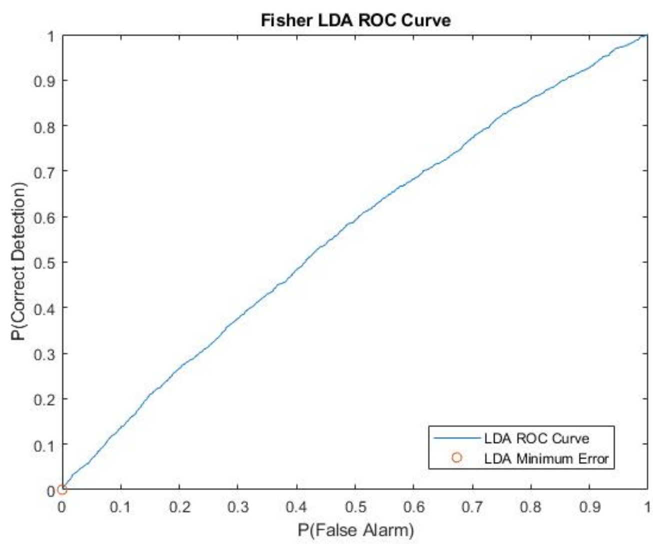

Figure 5: Problem 1 LDA ROC Curve

Based on the 10,000 samples generated, the minimum probability of error was calculated to be 20% at the threshold value of $\tau = 3.66$. This error and threshold value were calculated by finding the minimum error as the value of τ was changed from $-\infty$ to ∞. The orange circle in Figure 5 above marks this point. Similar to the previous case, these error and threshold values make sense since in the best case scenario, the smallest of the classes would get classified incorrectly, which in this case is Class 1 with a class prior of 20%. As expected, the Naive-Bayesian and the LDA classification rules result in significantly worse minimum probability of error as compared to the classifier model from Part 1.

Note: All MATLAB code for Problem 1 can be found in Appendix A.

Problem 2

Generating Data
Chosen class priors:

$$p(x|L=0) = 0.6 \qquad p(x|L=1) = 0.4$$

Class conditional PDFs and mixture coefficients:

Class 0

$$\mu_1 = \begin{bmatrix} 3.5 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \frac{1}{3}\begin{bmatrix} 5 & 1 \\ 1 & 4 \end{bmatrix} \qquad P(Gauss\ 1|L=0) = 0.7$$

$$\mu_2 = \begin{bmatrix} 5.5 \\ 4 \end{bmatrix} \qquad \Sigma_2 = \frac{1}{10}\begin{bmatrix} 3 & -2 \\ -2 & 15 \end{bmatrix} \qquad P(Gauss\ 2|L=0) = 0.3$$

Class 1

$$\mu_3 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \qquad \Sigma_3 = \frac{1}{13}\begin{bmatrix} 3 & -2 \\ -2 & 15 \end{bmatrix} \qquad P(Gauss\ 3|L=0) = 0.25$$

$$\mu_4 = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \qquad \Sigma_4 = \frac{1}{13}\begin{bmatrix} 15 & 1 \\ 1 & 3 \end{bmatrix} \qquad P(Gauss\ 4|L=0) = 0.75$$

Figure 6 below shows a plot of generated data points and their true class labels:

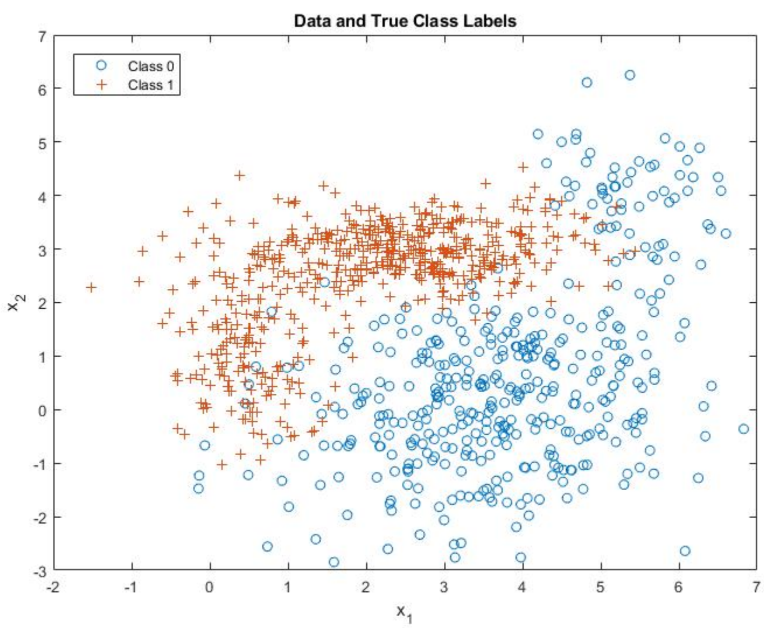

Figure 6: Class distributions and true labels of data points

Classification rule used to make decisions (using 0-1 loss values to achieve minimum probability of error):

$$\frac{p(x|L=1)}{p(x|L=0)} \gtrless \frac{P(L=0)}{P(L=1)}\left(\frac{\lambda_1 0 - \lambda_0 0}{\lambda_0 1 - \lambda_1 1}\right) = \frac{0.6}{0.4}$$

$$\frac{0.7 \cdot g(x|m_1, \Sigma_1) + 0.3 \cdot g(x|m_2, \Sigma_2)}{0.25 \cdot g(x|m_3, \Sigma_3) + 0.75 \cdot g(x|m_4, \Sigma_4)} \gtrless 1.5$$

Figure 7 below shows how the data was classified using this decision boundary. All wrong decisions are marked in red, while correct decisions are marked in green.

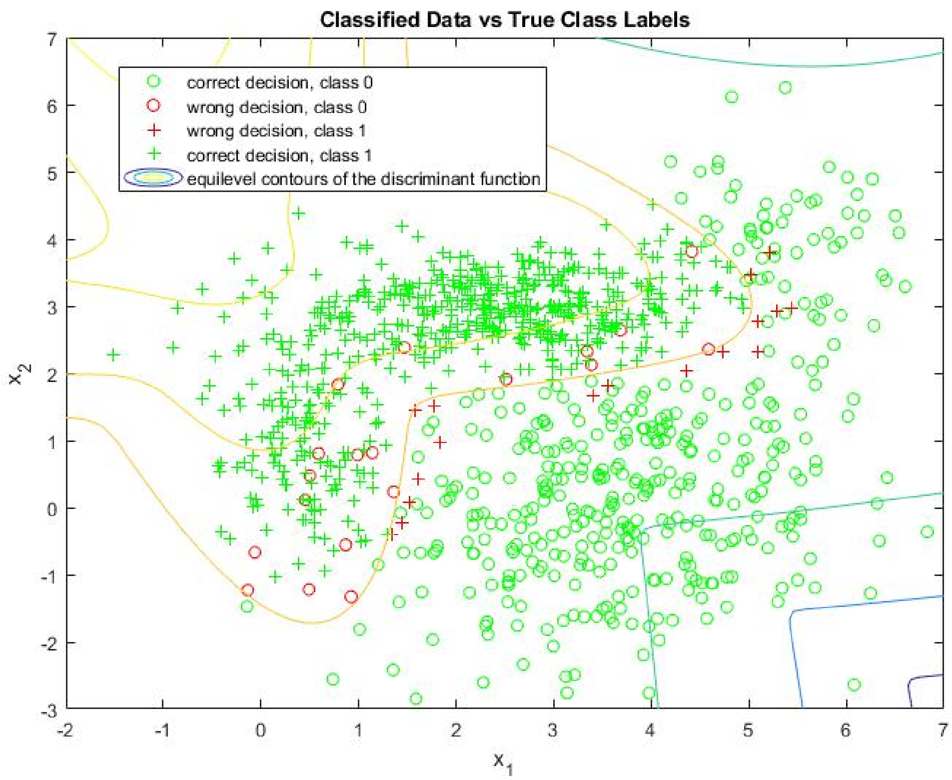

Figure 7: Data point decisions compared to their true labels

Based on this decision boundary, the minimum probability of error was calculated by counting the number of missed detections and false alarms and multiplying their probabilities by the class priors. Ultimately, the minimum probability of error was calculated to be 3.59%.

For Question 2 MATLAB code, please refer to Appendix B. Note: I am one of the individiuals who volunteered to write the example code for the class, which is why my code and the accompanying comments closely resemble the example in the GDrive.

Problem 3

Classification Rule Derivation
Given values:

$$p(x|L=0) \sim N(A_0, \sigma_0), \quad A_0 = -2 \quad \sigma_0 = 1$$
$$p(x|L=1) \sim N(A_1, \sigma_1), \quad A_1 = 2 \quad \sigma_1 = 1$$

Set up the classification rule:

$$(D=1) \qquad p(x|L=1) \gtrless p(x|L=0) \qquad (D=0)$$
$$(D=1) \qquad \frac{p(x|L=1)}{p(x|L=0)} \gtrless 1 \qquad (D=0)$$

Simplify the ratio of class 0 and class 1 probabilities:

$$\frac{\frac{1}{\sqrt{2\pi}\sigma_{x_1}} e^{\frac{-(x-A_1)^2}{2\sigma_{x_1}^2}}}{\frac{1}{\sqrt{2\pi}\sigma_{x_0}} e^{\frac{-(x-A_0)^2}{2\sigma_{x_0}^2}}} = \frac{e^{\frac{-(x-A_1)^2}{2}}}{e^{\frac{-(x-A_0)^2}{2}}} = e^{\frac{-(x-A_1)^2}{2} + \frac{-(x-A_0)^2}{2}}$$

Plug ratio back into the equation and simplify:

$$\frac{-(x-A_1)^2}{2} + \frac{-(x-A_0)^2}{2} \gtrless ln(1)$$
$$-(x-A_1)^2 + -(x-A_0)^2 \gtrless 0$$
$$-x^2 + 2A_1x - A_1^2 + x^2 - 2A_0x + A_0^2 \gtrless 0$$
$$2x(A_1 - A_0) \gtrless A_1^2 - A_0^2$$
$$(D=1) \qquad x \gtrless \frac{A_1^2 - A_0^2}{2(A_1 - A_0)} \qquad (D=0)$$

Plug given values back into equation to calculate the threshold:

$$x \gtrless \frac{2^2 - (-2)^2}{2(2-(-2))} = \frac{4-4}{8} = 0$$

Final classification rule and threshold value:

$$(D=1) \qquad x \gtrless 0 \qquad (D=0)$$

Problem 3 continued on next page...

Probability of Error Calculation

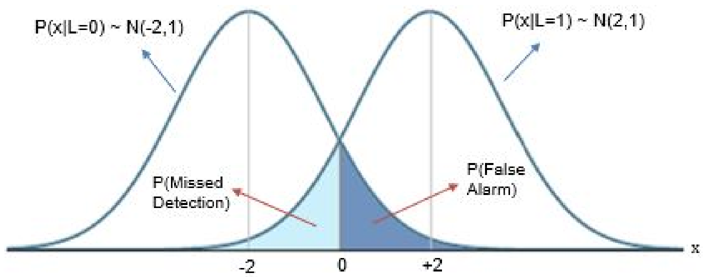

Figure 8: Sketch of Distributions for Question 3

Figure 8 above shows a sketch of the two class distributions and how the probability of false alarm and missed detection can be calculated from those distributions.

Calculating Probability of False Alarm:

$$P_{fa} = \int_{\gamma}^{\infty} p(x|L=0)dx = \int_{0}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-(x+2)^2}{2}} dx$$

Calculating Probability of Missed Detection:

$$P_{md} = \int_{-\infty}^{\gamma} p(x|L=1)dx = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}} e^{\frac{-(x-2)^2}{2}} dx$$

$P_{fa} = P_{md}$ since $p(x|L=0)$ and $p(x|L=1)$ are identically distributed, their class priors are equal, and they are the same distance away from the optimal threshold value.

Calculating (Minimum) Total Probability of Error:

$$P_{error} = P_{fa} \cdot P_0 + P_{md} \cdot P_1$$

$$P_{error} = P_{fa}(P_0 + P_1) = P_{fa}$$

$$P_{error} = P_{fa} = P_{md} = \int_{0}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-(x+2)^2}{2}} dx = 0.0228$$

Appendix A: Question 1 Code

```matlab
%% ============================ Question 1 Setup ============================ %%
% Anja Deric | February 10, 2020 | Take Home Exam #1
clear all; close all; clc;

n = 2;      % # of dimensions
N = 10000;  % # of samples

% Class means and covariances
mu(:,1) = [-0.1;0]; Sigma(:,:,1) = [1 -0.9;-0.9 1];
mu(:,2) = [0.1;0];  Sigma(:,:,2) = [1 0.9; 0.9 1];

% Class priors and true labels
p = [0.8,0.2];
label = rand(1,N) >= p(1);
Nc = [sum(label==0),sum(label==1)];

% Draw samples from each class pdf
x = zeros(n,N);
x(:,label==0) = mvnrnd(mu(:,1),Sigma(:,:,1),Nc(1))';
x(:,label==1) = mvnrnd(mu(:,2),Sigma(:,:,2),Nc(2))';

% Plot true class labels
figure(1);
plot(x(1,label==0),x(2,label==0),'o',x(1,label==1),x(2,label==1),'+');
title('Class 0 and Class 1 True Class Labels')
xlabel('x_1'), ylabel('x_2')
legend('Class 0','Class 1')

%% =========================== Question 1: Part 1 =========================== %%
% Calculate discriminant scores and tau
discriminantScore = log(evalGaussian(x,mu(:,2),Sigma(:,:,2))./evalGaussian(x,mu(:,1),Sigma(:,:,1)));
tau = log(sort(discriminantScore(discriminantScore >= 0)));

% Find midpoints of tau to use as threshold values
mid_tau = [tau(1)-100 tau(1:end-1) + diff(tau)./2 tau(length(tau))+100];

% Make decision for every threshold and calculate error values
for i = 1:length(mid_tau)
    decision = (discriminantScore >= mid_tau(i));
    pFA(i) = sum(decision==1 & label==0)/Nc(1); % False alarm prob.
    pCD(i) = sum(decision==1 & label==1)/Nc(2); % Correct detection prob.
    pE(i) = pFA(i)*p(1)+(1-pCD(i))*p(2);        % Total error prob.
end

% Find minimum error and corresponding threshold
[min_error,min_index] = min(pE);
```

```
min_decision = (discriminantScore >= mid_tau(min_index));
min_FA = pFA(min_index); min_CD = pCD(min_index);

% Find theoretical minimum error(threshold calculated using class priors)
ideal_decision = (discriminantScore >= log(p(1)/p(2)));
ideal_pFA = sum(ideal_decision==1 & label==0)/Nc(1); % False alarm
ideal_pCD = sum(ideal_decision==1 & label==1)/Nc(2); % Correct detection
ideal_error = ideal_pFA*p(1)+(1-ideal_pCD)*p(2);

% Plot ROC curve with minimum error point labeled
figure(2); plot(pFA,pCD,'-',min_FA,min_CD,'o',ideal_pFA,ideal_pCD,'g+');
title('Minimum Expected Risk ROC Curve'); legend('ROC Curve', 'Calculated Min Error', 'Theoretical Min Error');
xlabel('P_{False Alarm}'); ylabel('P_{Correct Detection}');

% Print all results
fprintf('<strong>Theoretical Results</strong>\n');
fprintf('Minimum probability of error: %.2f%%\nThreshold Value: %.2f\n', ideal_error*100,p(1)/p(2));

fprintf('\n<strong>Calculated Results</strong>\n');
fprintf('Minimum probability of error: %.2f%%\nThreshold Value: %.2f\n\n', min_error*100,exp(mid_tau(min_index)));

%% ======================== Question 1: Part 2 ======================== %%
Sigma_NB(:,:,1) = [1 0;0 1]; Sigma_NB(:,:,2) = [1 0;0 1];

% Calculate discriminant scores and tau
discriminantScore_NB = log(evalGaussian(x,mu(:,2),Sigma_NB(:,:,2))./evalGaussian(x,mu(:,1),Sigma_NB(:,:,1)));
tau_NB = log(sort(discriminantScore_NB(discriminantScore_NB >= 0)));

% Find midpoints of tau to use as threshold values
mid_tau_NB = [tau_NB(1)-1 tau_NB(1:end-1) + diff(tau_NB)./2 tau_NB(length(tau_NB))+1];

% Make decision for every threshold and calculate error values
for i = 1:length(mid_tau_NB)
    decision_NB = (discriminantScore_NB >= mid_tau_NB(i));
    pFA_NB(i) = sum(decision_NB==1 & label==0)/Nc(1); % False alarm prob.
    pCD_NB(i) = sum(decision_NB==1 & label==1)/Nc(2); % Correct detection prob.
    pE_NB(i) = pFA_NB(i)*p(1)+(1-pCD_NB(i))*p(2);       % Total error prob.
end

% Find minimum error and corresponding threshold
[min_error_NB,min_index_NB] = min(pE_NB);
min_decision_NB = (discriminantScore >= mid_tau_NB(min_index_NB));
min_FA_NB = pFA_NB(min_index_NB); min_CD_NB = pCD_NB(min_index_NB);
```

```
% Plot ROC curve with minimum error point labeled
figure(4); plot(pFA_NB,pCD_NB,'-',min_FA_NB,min_CD_NB,'o');
title('Naive-Bayesian ROC Curve'); legend('NB ROC Curve', 'NB Minimum Error');
xlabel('P_{False Alarm}'); ylabel('P_{Correct Detection}');

fprintf('\n<strong>Calculated Results, Naive-Bayesian</strong>\n');
fprintf('Minimum probability of error: %.2f%%\nThreshold Value: %.2f\n\n', min_error_NB*100,exp(mid_tau_NB(min_index_NB)));

%% ======================== Question 1: Part 3 ======================== %%
% Code help and example from Prof. Deniz
% LDA setup
Sb = (mu(:,1)-mu(:,2))*(mu(:,1)-mu(:,2))';
Sw = Sigma(:,:,1) + Sigma(:,:,2);

% Calculating Fisher LDA projection vector
[V,D] = eig(inv(Sw)*Sb); % alpha w  = inv(Sw) Sb w
[~,ind] = sort(diag(D),'descend');
wLDA = V(:,ind(1));     % Fisher LDA projection vector
yLDA = wLDA'*x;         % All data projected on to the line spanned by wLDA
wLDA = sign(mean(yLDA(find(label==1)))-mean(yLDA(find(label==0))))*wLDA; % ensures class 1 falls on the + side of the axis
yLDA = sign(mean(yLDA(find(label==1)))-mean(yLDA(find(label==0))))*yLDA; % flip yLDA accordingly

% Plot LDA projection
figure(5);
plot(yLDA(find(label==0)),zeros(1,Nc(1)),'o',yLDA(find(label==1)),zeros(1,Nc(2)),'+');
title('LDA projection of data points and their true labels');
xlabel('x_1'); ylabel('x_2'); legend('Class 0','Class 1');

% Sort LDA projection vector and find midpoints
sorted_yLDA = sort(yLDA);
mid_tau_LDA = [sorted_yLDA(1)-1 sorted_yLDA(1:end-1)+diff(sorted_yLDA)./2 sorted_yLDA(length(sorted_yLDA))+1];

% Make decision for every threshold value and find error probabilities
for i = 1:length(mid_tau_LDA)-1
    decisionLDA = (yLDA >= mid_tau_LDA(i));
    pFA_LDA(i) = sum(decisionLDA==1 & label==0)/Nc(1); % False alarm
    pCD_LDA(i) = sum(decisionLDA==1 & label==1)/Nc(2); % Correct detection
    pE_LDA(i) = pFA_LDA(i)*p(1)+(1-pCD_LDA(i))*p(2);
end

% Find minimum error and corresponding threshold
[min_error_LDA,min_index_LDA] = min(pE_LDA);
min_decision_LDA = (yLDA >= mid_tau_LDA(min_index_LDA));
```

```
min_FA_LDA = pFA_LDA(min_index_LDA); min_CD_LDA = pCD_LDA(min_index_LDA);

% Plot LDA ROC Curve
figure(6); plot(pFA_LDA,pCD_LDA,'-',min_FA_LDA,min_CD_LDA,'o');
title('Fisher LDA ROC Curve'); legend('LDA ROC Curve', 'LDA Minimum Error');
xlabel('P(False Alarm)'); ylabel('P(Correct Detection)');

%Print Results
fprintf('\n<strong>Calculated Results, LDA</strong>\n');
fprintf('Minimum probability of error: %.2f%%\nThreshold Value (tau): %.2f\n\n',min_error_LDA*100,mid_tau_LDA(min_index_LDA));

%% ======================== Question 1: Functions ======================== %%
% Function credit: Prof. Deniz
function g = evalGaussian(x,mu,Sigma)
% Evaluates the Gaussian pdf N(mu,Sigma) at each coumn of X
[n,N] = size(x);

C = ((2*pi)^n * det(Sigma))^(-1/2); % coefficient
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(Sigma)*(x-repmat(mu,1,N))),1); % exponent
g = C*exp(E); % final gaussian evaluation

end
```

Appendix B: Question 2 Code

```
% Question 2 | Take Home Exam #1
% Anja Deric | February 10, 2020
clear all; close all; clc;

n = 2;      % number of feature dimensions
N = 1000;   % number of iid samples

% Class 0 parameters (2 gaussians)
mu(:,1) = [3.5;0]; mu(:,2) = [5.5;4];
Sigma(:,:,1) = [5 1;1 4]/3; Sigma(:,:,2) = [3 -2;-2 15]/10;
p0 = [0.8 0.2]; % Class 0 mixture coefficients

% Class 1 parameters (2 gaussians)
mu(:,3) = [0.5;1]; mu(:,4) = [2.5;3];
Sigma(:,:,3) = [3 -2;-2 15]/13; Sigma(:,:,4) = [15 1;1 3]/13;
p1 = [0.25 0.75]; % Class 1 mixture coefficients

% Class priors for class 0 and 1 respectively
p = [0.4,0.6];

% Generating true class labels
label = rand(1,N) >= p(1);
Nc = [length(find(label==0)),length(find(label==1))];

% Draw samples from each class pdf
x = zeros(n,N); % save up space
for i = 1:N
    % Generating class 0 samples
    if label(i) == 0
        % Split samples based on mixture coefficients for class 0
        if (rand(1,1) > p0(1)) == 0
            x(:,i) = mvnrnd(mu(:,1),Sigma(:,:,1),1)';
        else
            x(:,i) = mvnrnd(mu(:,2),Sigma(:,:,2),1)';
        end
    end

    % Generating class 1 samples
    if label(i) == 1
        % Split samples based on mixture coefficients for class 1
        if (rand(1,1) > p1(1)) == 0
            x(:,i) = mvnrnd(mu(:,3),Sigma(:,:,3),1)';
        else
            x(:,i) = mvnrnd(mu(:,4),Sigma(:,:,4),1)';
        end
    end
end
```

```
% Plot samples with true class labels
figure(1);
plot(x(1,label==0),x(2,label==0),'o',x(1,label==1),x(2,label==1),'+');
legend('Class 0','Class 1'); title('Data and True Class Labels');
xlabel('x_1'); ylabel('x_2');

% Calculate threshold based on loss values
lambda = [0 1;1 0];
gamma = (lambda(2,1)-lambda(1,1))/(lambda(1,2)-lambda(2,2)) * p(1)/p(2);

% Calculate discriminant score based on class pdfs
class0pdf = p0(1)*evalGaussian(x,mu(:,1),Sigma(:,:,1)) + p0(2)*evalGaussian(x,
    mu(:,2),Sigma(:,:,2));
class1pdf = p1(1)*evalGaussian(x,mu(:,3),Sigma(:,:,3)) + p1(2)*evalGaussian(x,
    mu(:,4),Sigma(:,:,4));
discriminantScore = log(class1pdf)-log(class0pdf);

% Compare score to threshold to make decisions
decision = (discriminantScore >= log(gamma));

% Calculate error probabilities
TN = find(decision==0 & label==0); % true negative
FP = find(decision==1 & label==0); % false positive
FN = find(decision==0 & label==1); % false negative
TP = find(decision==1 & label==1); % true positive

% Calculate and print error values
pFA = length(FP)/Nc(1); % prob. false alarm
pMD = length(FN)/Nc(2); % prob. missed detection
pE = pFA*p(1)+pMD*p(2); % toatal prob. of error
fprintf('Minimum Probability of Error: %.2f%%',pE*100);

% Plot correct and incorrect decisions
% class 0 circle, class 1 +, correct green, incorrect red
figure(2);
plot(x(1,TN),x(2,TN),'og'); hold on;
plot(x(1,FP),x(2,FP),'or'); hold on;
plot(x(1,FN),x(2,FN),'+r'); hold on;
plot(x(1,TP),x(2,TP),'+g'); hold on;

% Grid based on class PDFs
horizontalGrid = linspace(floor(min(x(1,:))),ceil(max(x(1,:))),101);
verticalGrid = linspace(floor(min(x(2,:))),ceil(max(x(2,:))),91);
[h,v] = meshgrid(horizontalGrid,verticalGrid);
class0grid = p0(1)*evalGaussian([h(:)';v(:)'],mu(:,1),Sigma(:,:,1)) + p0(2)*
    evalGaussian([h(:)';v(:)'],mu(:,2),Sigma(:,:,2));
class1grid = p1(1)*evalGaussian([h(:)';v(:)'],mu(:,3),Sigma(:,:,3)) + p1(2)*
    evalGaussian([h(:)';v(:)'],mu(:,4),Sigma(:,:,4));
```

```
% Decision boundary grid
discriminantScoreGridValues = log(class1grid)-log(class0grid) - log(gamma);
minDSGV = min(discriminantScoreGridValues);
maxDSGV = max(discriminantScoreGridValues);
discriminantScoreGrid = reshape(discriminantScoreGridValues,91,101);

% Plot discriminant grid contours (level 0 = decision boundary)
contour(horizontalGrid,verticalGrid,discriminantScoreGrid,[minDSGV
    *[0.9,0.6,0.3],0,[0.3,0.6,0.9]*maxDSGV]);
legend('correct decision, class 0','wrong decision, class 0','wrong decision,
    class 1','correct decision, class 1','equilevel contours of the
    discriminant function');
title('Classified Data vs True Class Labels');
xlabel('x_1'); ylabel('x_2');
```