ECE302 MATLAB Detection Exercise Report

Junbum Kim, Toyou Kim
Professor Sam Keene
April 18, 2018

Part 1: Radar Detection

Consider the following scenario with two hypotheses:

$$H_0: \text{Target Not Present} \quad Y = X$$

$$H_1: \text{Target Present} \quad Y = A + X$$

$$P(H_0) = 0.8$$

$$X \sim N(0, \sigma_x^2)$$

$$\text{SNR} = \frac{\text{A}}{\sigma_x^2} = 1.5 \qquad \sigma_x^2 = 3$$

Part a: MAP Decision Rule

MAP decision rule selects the hypothesis that is most likely given the observation and a priori probability of the hypothesis, as follows:

$$\text{P}(\text{Y} = \text{y}|\text{H}_0)\text{P}(\text{H}_0) \gtrless \text{P}(\text{Y} = \text{y}|\text{H}_1)\text{P}(\text{H}_1)$$

$$\text{P}(\text{X} = \text{y}) \times 0.8 \gtrless \text{P}(\text{X} = \text{y} - \text{A}) \times 0.2$$

where $X \sim N(0, \sigma_x^2)$. By solving the equation in terms of y using the pdf of normal random distribution, following decision rule is obtained:

$$\text{y} \gtrless \frac{\text{A}}{2} + \frac{\ln(4)\,\sigma_x^2}{\text{A}} = \Gamma$$

(If y is larger than Γ, choose H_0; otherwise, choose H_1)

1000 iterations of the implementation resulted in 75 errors, which equates to empirical probability of error of 0.075, or 7.5%. This is extremely close to the theoretical probability of error, which is P_{err} = 0.0711, for the given setting. Theoretical probability of error was calculating using the following equation:

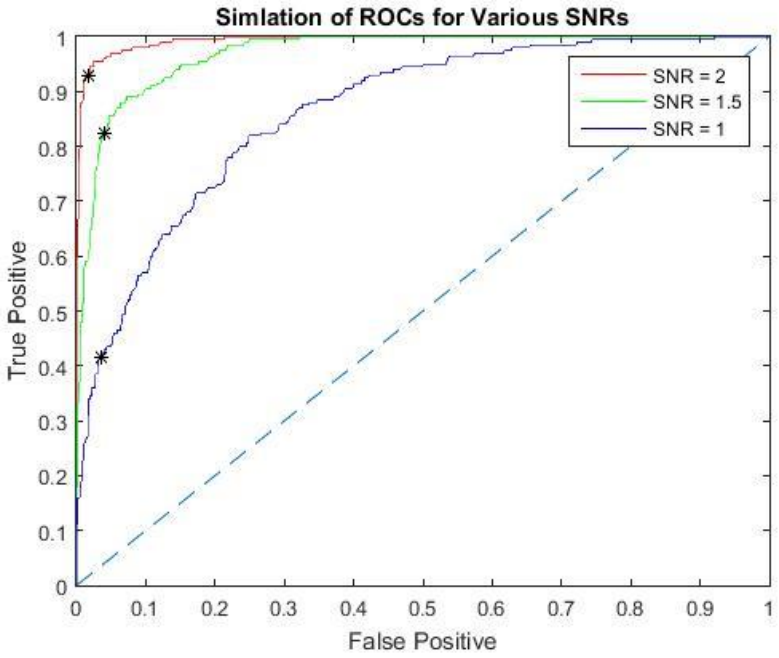
$$\text{P}_{\text{err}} = \text{P}(\text{y} < \Gamma|\text{H}_0)\text{P}(\text{H}_0) + \text{P}(\text{y} > \Gamma|\text{H}_1)\text{P}(\text{H}_1)$$

Note that P(y< Γ|H_0) P(H_0) is the probability of false positive and P(y> Γ|H_1) P(H_1) is the

probability of false negative.

Part b: Receiver Operating Curve

Receiver operating curves for the radar detection system was plotted for several SNRs. For exhaustive threshold values, the minimum and maximum values of the observations of Y were obtained. For each ROC, the threshold value corresponding to the MAP decision rule is marked with black star for reference.



The radar detection system performs better for higher values of SNR, as expected.

Part c: Cost Structure

Assume that missing the target (False negative) is 10 times worse than falsely detecting the target (False positive). To reflect this, costs are assigned to each case, denoted as C_{ij}, which indicates the cost of classifying H_j as H_i. For the given cost structure, consider the following cost matrix:

$$\begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix} = \begin{pmatrix} 0 & 10 \\ 1 & 0 \end{pmatrix}$$

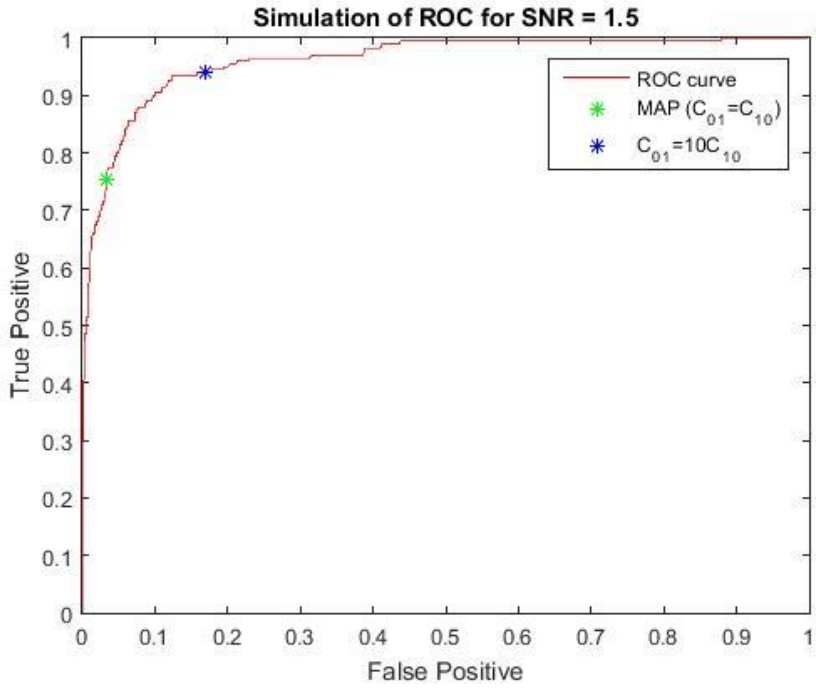
Since the initial MAP decision rule assumes equal cost, in order to minimize the conditional risk based on the cost structure, the decision rule is modified as follows:

$$P(Y = y|H_0)P(H_0)C_{10} \gtrless P(Y = y|H_1)P(H_1)C_{01}$$

By solving the modified expression in terms of y, following decision rule is obtained:

$$y \gtrless \frac{A}{2} + \frac{\ln(0.4)\,\sigma_x^2}{A} = \Gamma_1$$

Threshold corresponding to this decision rule is marked along with the original MAP decision rule in the following figure:



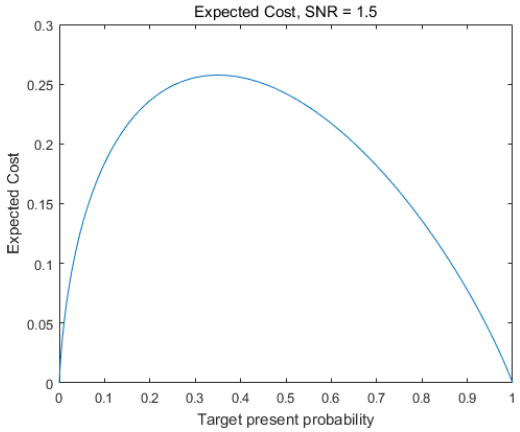
Notice how the new decision rule makes a tradeoff between false negative (since TP = 1-FN) and false positive in comparison with the MAP decision rule.

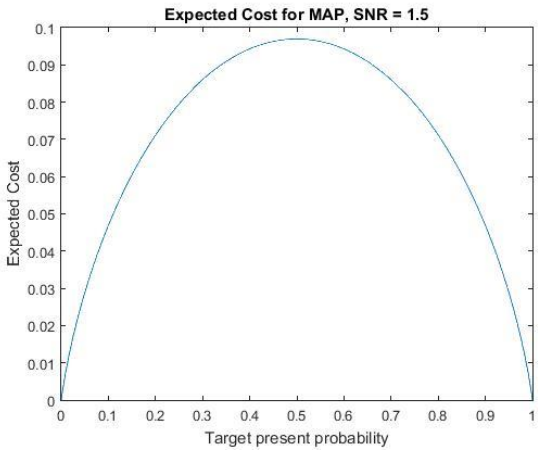
Part d: Expected Cost

Note that the decision rule obtained above only applies to the given a priori probability of the target presence, i.e. target is not present with probability 0.8. Therefore, the numerical value of the decision rule is altered based on the a priori probability. Using the cost structure in c), the optimal decision rule as a function of $P(H_0) = p$ is as follows:

$$y(p) \gtrless \frac{A}{2} + \frac{\ln\left(\frac{p}{10(1-p)}\right)\sigma_x^2}{A}$$

For the special case of p=0, i.e. target always present, and p=1, i.e. target never present, the logarithmic term diverges and result in always choosing one hypothesis over another, regardless of y.





As previously discussed, the expected cost is zero on endpoints. Overall, the plot takes form of a positively skewed parabola. This makes sense as the decision rule is characterized by relatively low false negative rate and relatively high false positive rate, hence higher target present probability is more beneficial to this decision rule compared to a traditional MAP decision rule, which is a symmetric parabola.

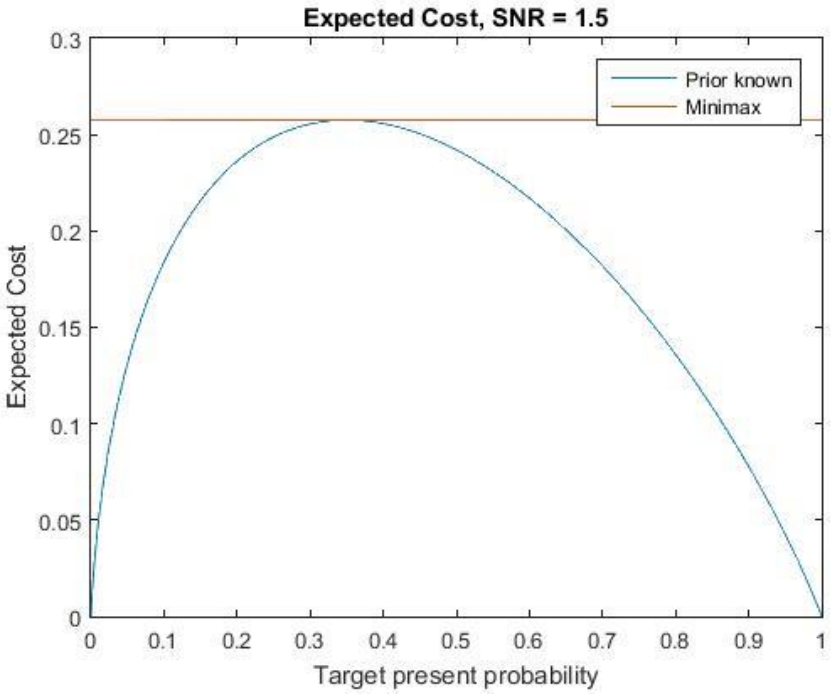
Part e: Minimax Decision Rule

In previous scenarios, the a priori probability of target presence was known. Assume it is unknown. This effectively forces a decision rule that depends on guessed a priori probability. In general, performance of this decision rule will be dependent on the difference between guessed a priori probability and the actual a priori probability. In fact, the expected cost as a function of costs, assumed probability P and true probability P_1 is given by equation (8.83) in Garcia, which simplifies to following in this scenario:

$$E(P, P_1) = (10 - P_F - 10P_D)P_1 + P_F$$

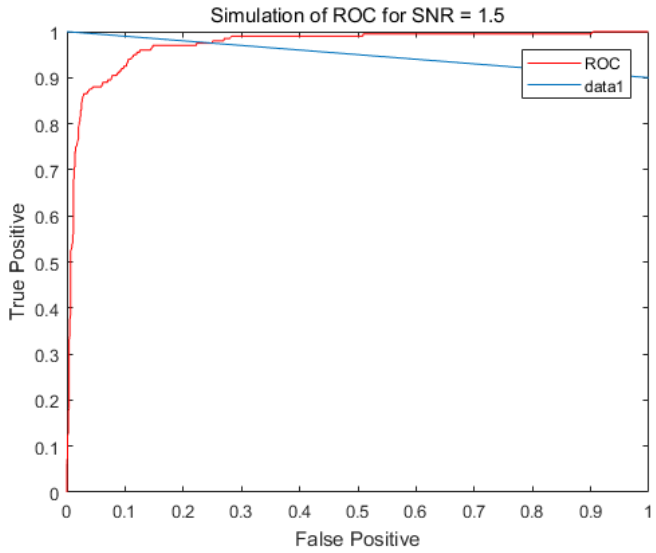
Where P_D and P_F refer to the probability of true positive and false positive at assumed probability P, respectively.

Minimax decision rule is made to minimize the maximum cost; that is, it assumes the a priori probability such that the possible maximum cost at arbitrary true a priori probability is minimized. This is obtained by setting $10\text{-}P_F\text{-}10P_D = 0$, so that expected cost is not affected by true P_1 value. This results in the following expected cost:



The minimax decision rule results in constant expected cost, as expected. Note that this cannot be lower than the expected cost plot of the decision rule with prior known at any target present probability since the latter results in minimum expected cost by construction.

Therefore, based on flatness of expected cost of minimax decision rule and the constraint mentioned above, minimax decision rule can be easily obtained by the extremum of expected cost plot with prior known.



Minimax decision rule is characterized by $10\text{-}P_F\text{-}10P_D = 0$, or $P_D=1\text{-}0.1P_F$, which is simply a

line in (P_F, P_D) space, as shown along the ROC curve above.

Part F: Modified Model

Consider the following modified scenario:

$$H_0: \text{Z Noise} \quad Y = A + Z$$

$$H_1: \text{X Noise} \quad Y = A + X$$

$$P(H_0) = 0.8$$

$$X \sim N(0, \sigma_x^2)$$

$$Z \sim N(0, \sigma_z^2)$$

$$\text{SNR} = \frac{\text{A}}{\sigma_x^2} = 1.5 \qquad \sigma_x^2 = 3 \qquad \text{XZR} = \frac{\sigma_z^2}{\sigma_x^2} = 40$$

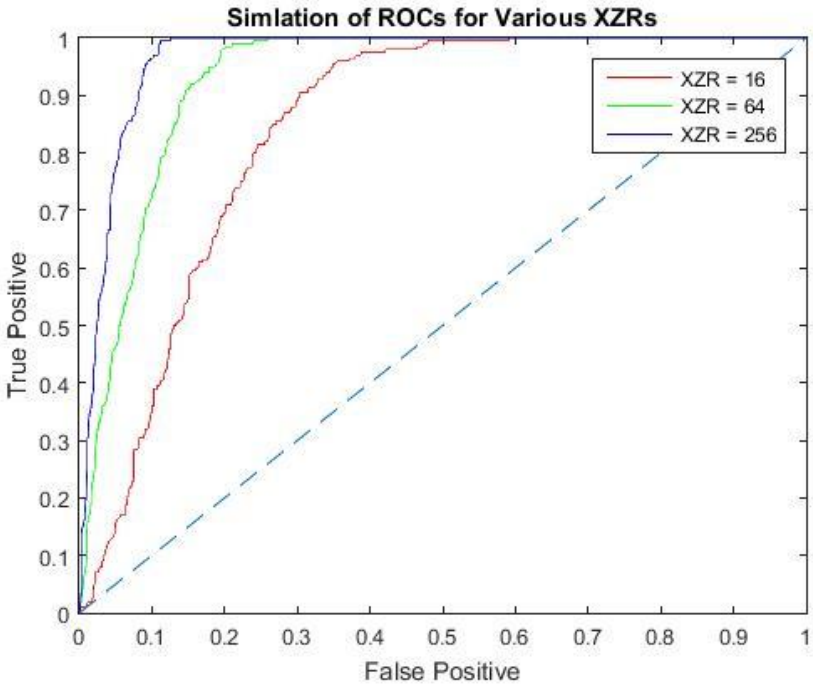
In this scenario, it is assumed the target can be transmitted through different channels with different noise level. Because the target is always present in this scenario, while the decision rule in original scenario classified the hypothesis based on the size of observation, new decision rule accounts for how close the observed signal is to the target A. MAP decision rule is as follows:

$$(\text{y} - \text{A})^2 \gtrless \ln\left(\frac{4\sigma_x}{\sigma_z}\right)\frac{2\sigma_x^2\sigma_z^2}{\sigma_x^2 - \sigma_z^2} \qquad \sqrt{\ln\left(\frac{4\sigma_x}{\sigma_z}\right)\frac{2\sigma_x^2\sigma_z^2}{\sigma_x^2 - \sigma_z^2}} = \Gamma'$$

(If $(y\text{-}A)^2$ is larger than RHS, choose H_0; otherwise, choose H_1)

This decision rule essentially compares the distance between Y and A to Γ'. Notice that the ratio of standard deviation and X and Z must be larger than 4; otherwise the RHS of the expression becomes negative. If the ratio is smaller than 4, since $(y\text{-}A)^2$ is always larger than RHS, the decision rule always chooses H_0, resulting in probability of error of 0.2. For the given XZR=40, the empirical probability of error comes out to be 0.167, which is close to the theoretical probability of error, 0.1639.

The thresholds for ROC of this model are obtained by finding the observation that is farthest away from A, and values between distance from that observation to A and 0 are used as thresholds. The ROC curves for different XZR are shown below:



Part 2: Non-Gaussian Detection

Part 3. Training and Testing Iris Data

The training set and testing set for the Iris data was selected by randomly splitting the sample into halves. Priors were calculated by counting the frequencies of each samples in the test sets. The training process was relatively simple, the mean and the variance for each class was calculated, so that it could be used to construct the multivariate gaussian random variable for the testing. The testing phase just takes the testing vector and constructs three probability distribution functions because each sample could belong to any of the three classes. To find the MAP estimate, prior calculated from testing is multiplied to the probability distribution function and the argmax of the three is selected as the label of the specific sample. This process was repeated 100 times to calculate the total probability of error and the normalized confusion matrix. The MAP estimate always labeled class 1 Iris correctly, whereas it performed not so well for class 2 Iris. The overall probability of error was 0.1896. The MAP estimate was not the best estimate performance wise. The author was able to achieve 0.02 probability of error using back propagating algorithm on the same dataset. However, MAP estimate was certainly much easier implementation wise.

Rows represent actual iris types, columns represent MAP classified iris types for 100 measurements

Normalized_Confusion_Matrix =

1.0000	0	0
0	0.8624	0.0521
0	0.1376	0.9479

Total_Probability_Error =

0.1896

APPENDIX

ECE302Proj2.m – Main file

```
%% Part 1.
clear all, clc, close all;
%% part 1.a.
itr = 1000;
var_x = 3;
sig_x = sqrt(var_x);
SNR = 1.5;
A = SNR*var_x;

not_present = normrnd(0, sig_x, 0.8*itr, 1);
present = normrnd(A, sig_x, 0.2*itr,1);

%calculate for when signal present
%correct
P_present = normpdf(present,A,sig_x);
%incorrect
P_absent = normpdf(present,0,sig_x);
error = P_present*0.2 - P_absent*0.8;

%calculate for when signal not present
%incorrect
P_present = normpdf(not_present,A,sig_x);
%correct
P_absent = normpdf(not_present,0,sig_x);
error2 = P_absent*0.8 - P_present*0.2;

s=sign([error;error2]);
P_error = sum(s(:)==-1)/itr

MAP = A/2 + var_x*log(4)/A;
P_FN = normcdf(MAP,A,sig_x);     % False Negative
P_FP = 1 - normcdf(MAP,0,sig_x);  % False Positive
P_error_theoretical = P_FN*0.2 + P_FP*0.8

%% part 1.b.
itr = 1000;
var_x = 3;
SNR1 = 2;
SNR2 = 1.5;
SNR3 = 1;
[tp1,fp1, th1] = ROC(SNR1, var_x, itr);
[tp2,fp2, th2] = ROC(SNR2, var_x, itr);
[tp3,fp3, th3] = ROC(SNR3, var_x, itr);
plot(smooth(fp1),smooth(tp1), 'r', smooth(fp2),smooth(tp2),
'g',smooth(fp3),smooth(tp3),
'b',fp1(th1),tp1(th1),'k*',fp2(th2),tp2(th2),'k*',fp3(th3),tp3(th3),'k*')
ref=refline(1,0);
ref.LineStyle='--';
xlabel('False Positive')
ylabel('True Positive')
title('Simlation of ROCs for Various SNRs')
legend('SNR = 2', 'SNR = 1.5', 'SNR = 1')
% MAP cutoffs are marked with black stars.
```

```
%% part 1.c
SNRc = 1.5;
Ac = SNRc*var_x;
%TFAE as New Decision Rule: Choose H0 iff
%1) P_FN*P(H1)>P_FP*P(H0)*10
%2) Y>A/2 + var_x*log(0.4)/A

costDR = Ac/2 + var_x*log(0.4)/Ac;

[tpc,fpc,thc,MAPth] = ROC(SNRc, var_x, itr, costDR);
figure;
plot(smooth(fpc),smooth(tpc), 'r', fpc(MAPth),tpc(MAPth), 'g*',
fpc(thc),tpc(thc), 'b*')
xlabel('False Positive')
ylabel('True Positive')
title('Simulation of ROC for SNR = 1.5')
legend('ROC curve', 'MAP (C_0_1=C_1_0)', 'C_0_1=10C_1_0')

% Based on the marked points, it can be analyzed that with the new cost
% structure, true positive rate is increased, i.e. false negative lowered,
% at the expense of increased false positive rate, which makes sense since
% false negative is deemed 10 times as severe compared to false positive.

%% part 1.d

% Y > A/2 + var_x*log((1-p)/10p)/A, p = P(H1)
C01 = 10;   % Cost of false negative
C10 = 1;    % Cost of false positive

p = 0:0.001:1;  % Target present probabilities from 0 to 1.
drule = A/2+var_x*log((1-p)./(10*p))/A;
P_FN = normcdf(drule,A,sig_x);
P_FP = 1 - normcdf(drule,0,sig_x);
expCost = C01*p.*P_FN+C10*(1-p).*P_FP;
drule2 =  A/2+var_x*log((1-p)./p)/A;
P_FN2 = normcdf(drule2,A,sig_x);
P_FP2 = 1 - normcdf(drule2,0,sig_x);
expCost2 = p.*P_FN2+(1-p).*P_FP2;
figure;

plot(p,expCost);
xlabel('Target present probability')
ylabel('Expected Cost')
title('Expected Cost, SNR = 1.5')

figure;
plot(p,expCost2);
xlabel('Target present probability')
ylabel('Expected Cost')
title('Expected Cost for MAP, SNR = 1.5')

%% part 1.e

% Minimax decision rule corresponds to parameter satisfying P_TP = 1-
(C10/C01)P_FP = 1-0.1P_FP
% Since P_TP = 1-P_FN, the equation reduces to the following:
% 0.1normcdf(drule,0,sig_x)+normcdf(drule,A,sig_x)-0.1=0
[m,i] = min(abs(0.1*normcdf(drule,0,sig_x)+normcdf(drule,A,sig_x)-0.1));
```

```
mmd = drule(i)
minimaxCost = C01*p*normcdf(mmd,A,sig_x)+C10*(1-p)*(1-
normcdf(mmd,0,sig_x));     %The value is constant regardless of the a
priori target present probability as expected.

figure;
plot(p,expCost);
xlabel('Target present probability')
ylabel('Expected Cost')
title('Expected Cost, SNR = 1.5')
hold on
plot(p,minimaxCost)
hold off
legend('Prior known','Minimax')
[tp,fp] = ROC(SNR, var_x, itr);
figure;
plot(smooth(fp),smooth(tp),'r')
xlabel('False Positive')
ylabel('True Positive')
title('Simulation of ROC for SNR = 1.5')
legend('ROC','Minimax line')
refline(-0.1,1)    % Minimax decision rule is expressed as linear line in
the (P_TP,P_FP) space. The intersection is used as the minimax operating
point.

%% part 1.fa

itr = 1000;
XZR = 40; % Ratio of variance of X and Z. This needs to be bigger than 16
for nontrivial analysis. Specified in the report.
var_x = 3;
var_z = XZR*var_x;
sig_x = sqrt(var_x);
sig_z = sqrt(var_z);
SNR = 1.5;
A = SNR*var_x;

znoise = normrnd(A, sig_z, 0.8*itr, 1);     % Target not present is
modified to Z noise.
xnoise = normrnd(A, sig_x, 0.2*itr, 1);

%calculate for when signal has X noise.
%correct
P_xnoise = normpdf(xnoise,A,sig_x);
%incorrect
P_znoise = normpdf(xnoise,A,sig_z);
error = P_xnoise*0.2 - P_znoise*0.8;

%calculate for when signal has Z noise
%incorrect
P_xnoise = normpdf(znoise,A,sig_x);
%correct
P_znoise = normpdf(znoise,A,sig_z);
error2 = P_znoise*0.8 - P_xnoise*0.2;

s=sign([error;error2]);
P_error2 = sum(s(:)==-1)/itr
```

```
MAP2 = sqrt(2*var_z*var_x/(var_x-var_z)*log(4*sig_x/sig_z));    %Value of
Y-A, i.e. X or Z, is compared to MAP2 to classify if the noise is X or Z.
P_FZ = 2*normcdf(-MAP2,0,sig_x);      % False Z; P(abs(Y-A)<MAP2 | X)
P_FX = 1-2*normcdf(-MAP2,0,sig_z);       % False X; P(abs(Y-A)>MAP2 | Z)
P_error_theoretical2 = P_FZ*0.2 + P_FX*0.8

%% part 1.fb
itr = 1000;
var_x = 3;
XZR1 = 16;
XZR2 = 64;
XZR3 = 256;
[tp1,fp1] = ROC2(SNR, XZR1, var_x, itr);
[tp2,fp2] = ROC2(SNR, XZR2, var_x, itr);
[tp3,fp3] = ROC2(SNR, XZR3, var_x, itr);
plot(smooth(fp1),smooth(tp1), 'r', smooth(fp2),smooth(tp2),
'g',smooth(fp3),smooth(tp3), 'b')
ref=refline(1,0);
ref.LineStyle='--';
xlabel('False Positive')
ylabel('True Positive')
title('Simlation of ROCs for Various XZRs')
legend('XZR = 16', 'XZR = 64', 'XZR = 256')

%% Part 2
clear all, clc;
% symbol transmission rate, symbols/sec
[r1 cr1, tp1, fp1] = ROC_exp(30,80);
[r2 cr2, tp2, fp2] = ROC_exp(30,400);
[r3 cr3, tp3, fp3] = ROC_exp(30,1000);

plot(smooth(fp1),smooth(tp1), 'r', smooth(fp2),smooth(tp2),
'g',smooth(fp3),smooth(tp3), 'b')
ref=refline(1,0);
ref.LineStyle='--';
xlabel('False Positive')
ylabel('True Positive')
title('Simlation of ROCs for Exponential Random Variables')
legend('R1=30, R2=80', 'R1=30, R2=400', 'R1=30, R2=1000')

sprintf('rows represent actual labels of binary transmission scheme,
columns represent MAP classified scheme')
POE1 = 1-sum(r1 == cr1)/length(cr1)
POE2 = 1-sum(r2 == cr2)/length(cr2)
POE3 = 1-sum(r3 == cr3)/length(cr3)
%% Part 3
clear all, clc;
iris = load('iris.mat');
Confusion_Matrix = zeros(3);

% 100 trials
for i = 1:100
    % randomly mixing samples, splitting samples into half to place to
    % training, and the other half to testing set
    idx = randperm(150);
    iris_features = iris.features(idx,:);
    iris_labels = iris.labels(idx);
```

```
    train_features = iris_features(1:75,:);
    train_labels = iris_labels(1:75);
    test_features = iris_features(76:end,:);
    test_labels = iris_labels(76:end);

    % spllitting training sets into three different classing, finding the
mean,
    % variance and the priors
    ind1 = find(train_labels==1);
    ind2 = find(train_labels==2);
    ind3 = find(train_labels==3);

    priors = [length(ind1), length(ind2), length(ind3)];
    priors = priors/sum(priors);

    f1 = train_features(ind1,:);
    f2 = train_features(ind2,:);
    f3 = train_features(ind3,:);

    % assuming gaussian random variable, construct pdfs according to the
mean
    % and std from training set
    p1 = priors(1)*mvnpdf(test_features, mean(f1), sqrt(var(f1)));
    p2 = priors(2)*mvnpdf(test_features, mean(f2), sqrt(var(f2)));
    p3 = priors(3)*mvnpdf(test_features, mean(f3), sqrt(var(f3)));

    % MAP: find maximum of the arguments calculated, and take the indices
of
    % the argmax
    p = [p1 p2 p3];
    [row, col] = find(p==max(p,[],2));
    result = sortrows([row, col]);
    result = result(:,2);

    for i = 1:length(result)
       Confusion_Matrix(test_labels(i), result(i)) =
Confusion_Matrix(test_labels(i), result(i)) + 1;
    end
    % at this point the confusion matrix is for this specific trial
end

sprintf('rows represent actual iris types, columns represent MAP classified
iris types for 100 measurements')
Normalized_Confusion_Matrix =
[Confusion_Matrix(:,1)/sum(Confusion_Matrix(:,1)),
Confusion_Matrix(:,2)/sum(Confusion_Matrix(:,2)),Confusion_Matrix(:,3)/sum(
Confusion_Matrix(:,3))]
Total_Probability_Error = sum(sum(Normalized_Confusion_Matrix)-
sum(Normalized_Confusion_Matrix.*eye(3)))
```

ROC.m – ROC function for 1.a~e

```
function [ true_positive, false_positive, threshold, MAPthreshold ] =
ROC( SNR, var_x, itr, condition )
A = SNR*var_x;
sig_x = sqrt(var_x);
MAP = A/2 + var_x*log(4)/A;
if nargin<4
    condition = MAP;      % if not specified, return threshold corresponding
to MAP decision rule
end


present = normrnd(A, sig_x, 0.2*itr,1);
not_present = normrnd(0, sig_x, 0.8*itr, 1);

lowbound_threshold = min([present; not_present]);
highbound_threshold = max([present; not_present]);
thresholds = linspace(lowbound_threshold,highbound_threshold,10000);
true_positive_curve = repmat(present,1,10000) - repmat(thresholds,200,1);
false_positive_curve = repmat(not_present,1,10000) -
repmat(thresholds,800,1);
true_positive_curve = sign(true_positive_curve)==1;
false_positive_curve = sign(false_positive_curve)==1;
true_positive = sum(true_positive_curve)/200;
false_positive = sum(false_positive_curve)/800;

[m,i] = min(abs(thresholds-condition));
threshold = i;
[mmap,imap] = min(abs(thresholds-MAP));      % Always return MAP threshold
for reference
MAPthreshold = imap;
end
```

ROC2.m – ROC function for 1.f

```
function [ true_positive, false_positive] = ROC2( SNR,XZR, var_x, itr)

A = SNR*var_x;
sig_x = sqrt(var_x);
var_z = XZR*var_x;
sig_z = sqrt(var_z);

znoise = normrnd(A, sig_z, 0.8*itr, 1);     % Target not present is
modified to Z noise.
xnoise = normrnd(A, sig_x, 0.2*itr, 1);
received = [xnoise; znoise];
lowbound_threshold =  max(abs(received-A));  % find the sample that is
farthest away from A.
highbound_threshold = 0;    % Only accept H1 if it's exactly A.
thresholds = linspace(lowbound_threshold,highbound_threshold,10000);
true_positive_curve = repmat(thresholds,200,1) - repmat(abs(xnoise-
A),1,10000);
false_positive_curve = repmat(thresholds,800,1) - repmat(abs(znoise-
A),1,10000);
true_positive_curve = sign(true_positive_curve)==1;
false_positive_curve = sign(false_positive_curve)==1;
true_positive = sum(true_positive_curve)/200;
false_positive = sum(false_positive_curve)/800;

end
```

ROC_exp.m – ROC function for exponential random variables

```
function [ result, correct_result, true_positive, false_positive] =
ROC_exp( rate1, rate2 )

r = randi(1000,2,1);

p1 = round(exprnd(rate1,r(1),1));
p2 = round(exprnd(rate2,r(2),1));
symbols = [zeros(r(1),1)+1;zeros(r(2),1)+2];
tmp = [p1;p2];
idx = randperm(sum(r));
rcvd = tmp(idx);
correct_result = symbols(idx);

MAP1 = exppdf(rcvd,rate1);
MAP2 = exppdf(rcvd,rate2);
MAP = [MAP1 MAP2];
[row, col] = find(MAP==max(MAP,[],2));
result = sortrows([row, col]);
result = result(:,2);

threshold = linspace(min(tmp),max(tmp), max(tmp)-min(tmp)+1);
curve = repmat(threshold, sum(r),1) - repmat(rcvd, 1, max(tmp)-min(tmp)+1);
prediction = sign(curve)==1;
correct = repmat(correct_result,1,length(threshold))==1;
true_positive_curve = prediction & correct;
false_positive_curve = prediction & not(correct);
true_positive = sum(double(true_positive_curve));
false_positive = sum(double(false_positive_curve));
true_positive = true_positive/max(true_positive);
false_positive = false_positive/max(false_positive);

end
```