Detection Exercise: Radar Detection, Pattern Classification, and Machine Learning - ECE 302

Yuval Ofek and Jason Kurian

April 16, 2020

Part 1. Radar Detection

In Part A we coded a detection system in MATLAB using the MAP rule, which chooses the hypothesis of whether the target is present or not based on whose posterior probability is higher. Essentially, it is a problem of deciding between two Gaussians with a mean difference equal to the target's magnitude. The decision rule thus simplifies to Eqn. 8.27 in the provided pdf on detection theory. This decision rule applied to the two Gaussians was simulated 1000 times in MATLAB, and the detector's probability of error was consistently within 1% of the theoretical value.

In Part B the receiver operating characteristic curves was plotted for this detector for varying SNR values. This required determining the point where H1 became more likely than H0; this point is equivalent to the gamma value provided by the MAP rule. The probability of detections and false alarms were then plotted in Figure 1. It is shown that the higher the SNR value, the more convex the curve is. In other words, the greater the target magnitude is compared to the noise, the more likely the detector will produce detections than false alarms.

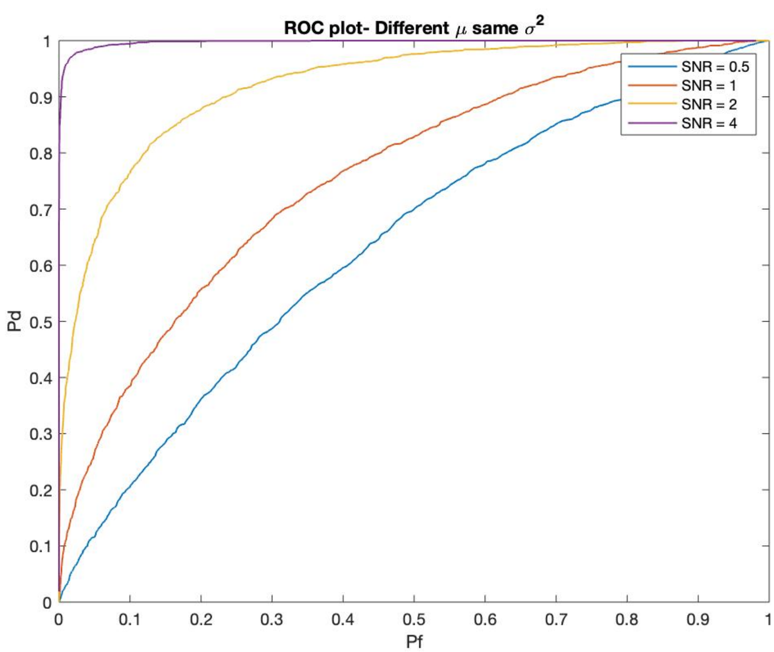

Figure 1: (1.b) ROCs for Radar Detector Using MAP Rule for Different SNR Ratios

In Part C a cost assignment was introduced, where a false negative was 10 times worse than a false detection. This changes the threshold of the likelihood ratio test given by eta, now a tenth of the original. The point that minimizes this conditional risk is shown in Figure 2 for an ROC with an SNR ratio of 1. This point occurs at a threshold value of 0.4.

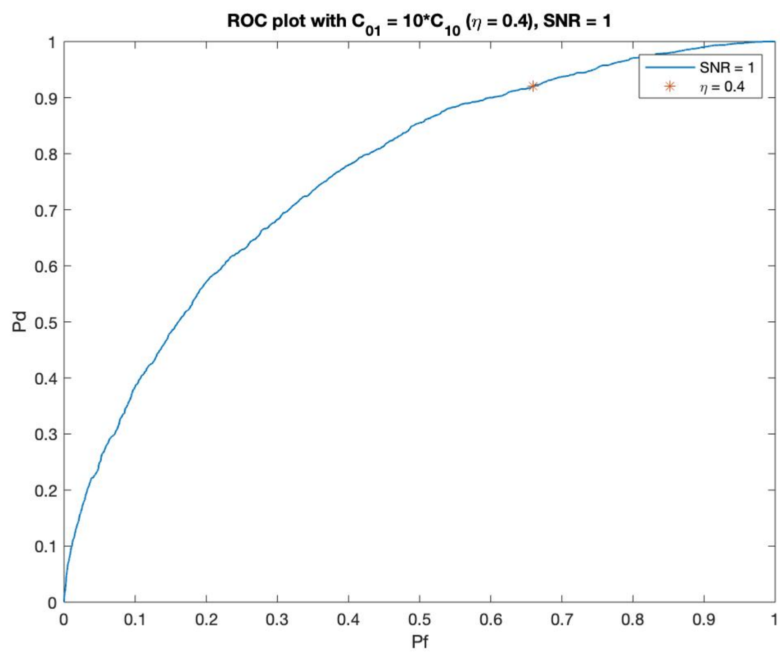

Figure 2: (1.c) ROC with Point Minimizing Risk of New Cost Assignment

In Part D the expected cost using the assignment in Part C was graphed for target present probabilities between 0 and 1, shown in Figure 3. The expected cost is equal to the sum of the expected costs of a false alarm and a miss, or $(C_{10} * P_{10} * P_0) + (P_{01} * P_1)$.

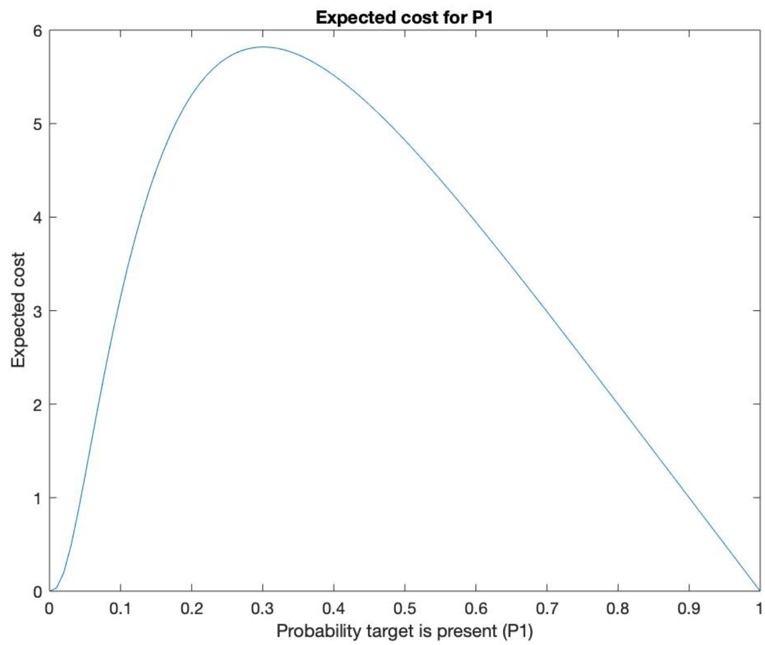

Figure 3: (1.d) Expected Cost for Target Probabilities From 0 to 1

In Part E the signal with the target is modelled the same, while the signal without the target is modelled with the same mean but a greater variance. The receiver operating curves for different ratio of variances between the two signals are shown in Figure 4. A greater ratio of variances means that the signal without the target is more distinguishable from the signal with the target, hence increasing the chances of detection.

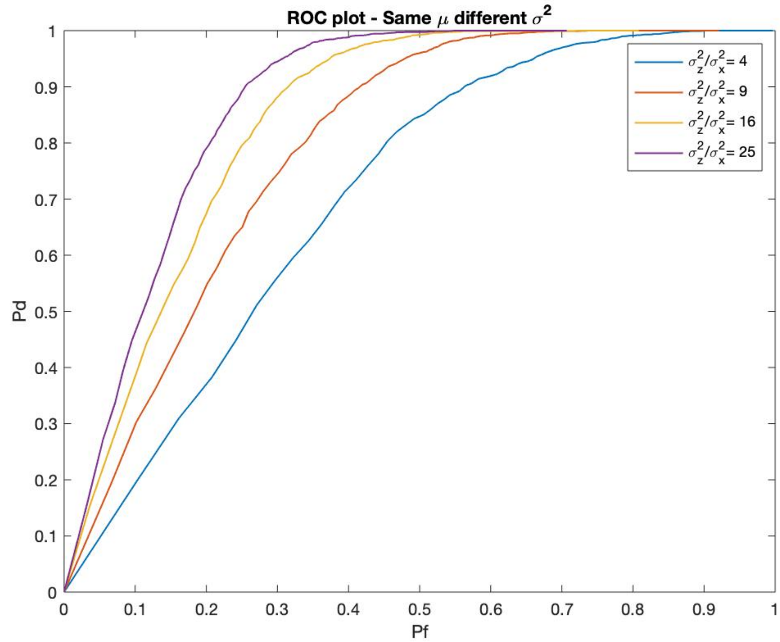

Figure 4: (1.e) ROCs with Added Gaussian to the Signal Without a Target

Part 2. Introduction to Pattern Classification and Machine Learning

The goal of this section is to classify a type of Iris plant based on its features. A data set from the UCI data repository was loaded into MATLAB, where it was split randomly in half into a training set and a testing set. MAP classification using the Gaussian probability model was used to determine the likelihood of each set of features belonging to each class. Creating the multivariate Gaussian distribution required calculating the mean vector and covariance matrices of the training set. The estimated labels of the test features were compared to the actual labels using the confusion function. The function returns a confusion value representing the fraction of samples misclassified, along with a confusion matrix charting how all the features were classified against their target. In one case run, the confusion value was 0.0133, and below is the corresponding confusion matrix:

	Classified as 1	Classified as 2	Classified as 3
Label 1	25	0	0
Label 2	0	23	0
Label 3	0	1	26

Appendix

1 MATLAB Code for Part 1

```
%Stoch Proj4 - Jason Kurian and Yuval Epstain Ofek

%% Part 1, Radar Detection
%% a.
clear all; close all;clc;

%%a.
Niter = 1e3;

%Determining the SNR
Amag = 1;
var = 1;
SNR = Amag/var;

%Probability that target is not there/is there
P0 = 0.8;
P1 = 1-P0;
eta = P0/P1;

%Generate Y vector;
[Y, Trgt] = genYsamestd(Amag, Niter, var, P0);

%The problem is pretty much deciding between 2 gaussians (determined by
%noise level) with a mean difference determined by A. We use the equation
%in the pdf online to find gamma:
Gamma = @(Amag,var, eta) Amag./2+var*log(eta)*ones(size(Amag))./(Amag);
gam = Gamma(Amag, var, eta);


%Check if chose correctly or not:
Perr_exp =1- sum(or(and(Y>gam,Trgt),and(Y<=gam, ~Trgt)))/Niter

%Theory
P10 = 1- normcdf(gam,0, sqrt(var));
P01 = normcdf(gam, Amag, sqrt(var));

Perr_ther = (P10*P0 + P01*P1)
%Approximates theory very accurately
```

```
%% b.
Niter = 1e4;

Amag = [0.5, 1, 2,4]; %SNRs
eta = logspace(-7,7,1e4);
figure
for i = 1:max(size(Amag))
    %Generate Y
    [Y, Trgt] = genYsamestd(Amag(i), Niter, var, P0);

    %Get the Probabilities for the ROC
    [Pd,Pf] = getROC(Amag(i), var, eta, Y, Trgt);
    plot(Pf,Pd, 'DisplayName', ['SNR = ',num2str(Amag(i)/var)], 'linewidth', 1)
    hold on
end
legend
xlabel('Pf')
ylabel('Pd')
title('ROC plot- Different \mu same \sigma^2')

%% c.
% Assume that missing the target is 10 times worse than falsely detecting
% the target. What is the decision rule that minimizes the conditional
% risk? Mark this point on your ROC for at least one SNR value.

% C01 = 10*C10;
Amag = 1;    %SNR = 1;
Niter = 1e4;

%Getting the ROC curve
[Y, Trgt] = genYsamestd(Amag, Niter, var, P0);
[Pd,Pf] = getROC(Amag, var, eta, Y, Trgt);

figure;
plot(Pf,Pd, 'DisplayName', ['SNR = ',num2str(Amag)], 'linewidth', 1)
legend
xlabel('Pf')
ylabel('Pd')
title(['ROC plot with C_{01} = 10*C_{10} (\eta = 0.4), SNR = ', num2str(Amag/var)])
```

```
hold on

%Finding the point on ROC curve:
%eta = (C10-C00)P0/((C01-C11)*P1) => nu = P0/(10*P1)
etac = (.1)*P0/P1;
[Pdc,Pfc] = getROC(Amag, var, etac, Y, Trgt);
plot(Pfc, Pdc, '*', 'DisplayName', '\eta = 0.4')

%% d.

%Generating a bunch of a-priori probabilities
P1 = 0:0.01:1;
P0  = 1-P1;
%Keeping same cost structure
C10 = 10;

%Determining probabilities
eta = P0./(C10.*P1);
gam = (2*var*log(eta)+Amag^2)/(2*Amag);
P10 = 1- normcdf(gam,0, sqrt(var));
P01 = normcdf(gam, Amag, sqrt(var));

%Cost
Ecost = (C10*P10.*P0 + P01.*P1);

figure
plot(P1,Ecost)
title('Expected cost for P1')
xlabel('Probability target is present (P1)')
ylabel('Expected cost')

%% e.
clear all;
%%% Same mean, different variance
%%% a-like:
Niter = 1e6;    %looks a bit nicer with the higher number
    of iterations
%Some parameters we chose/were given
varx = 1;
varz = 25;
sigx = sqrt(varx);
sigz = sqrt(varz);
A = 10;
P0 = 0.8;
P1 = 1-P0;
eta = P0/P1;
```

```
%generate Y
[Y, Trgt] = genYsamemean(sigx,sigz,Niter,A,P0);

%P(y|Hi)
PyH1 = @(Y,varx,A) (1/sqrt(varx*2*pi))*exp(-((Y-A).^2)
    /(2*varx));
PyH0 = @(Y,varz,A) (1/sqrt(varz*2*pi))*exp(-((Y-A).^2)
    /(2*varz));


%P(error) - using the decision based on which likelyhood
    is greater
Perr_exp = sum(or(and(PyH1(Y,varx,A)*P1 >= PyH0(Y,varz,A)
    *P0, Trgt) ...
     , and(PyH1(Y,varx,A)*P1 >= PyH0(Y,varz,A)*P0,~Trgt)))
        /Niter

%Theoretical resiults
Gammameansq = @(varx,varz,eta) sqrt(2*(varx*varz/(varx-
    varz))*log(sqrt(varx/varz)*eta));
gamsq = Gammameansq(varx, varz, eta);   %Thought I would
    use this more, but didn't

P10 = normcdf(gamsq,0,sigz)-normcdf(-gamsq,0,sigz);       %
    middle
P01 = 2*(1-normcdf(gamsq,0,sigx));       %2 ends
Perr_ther = (P10*P0 + P01*P1)
%% b - like

Niter = 1e4;

varz = [4, 9, 16, 25]; %differing variance ratios
sigz = sqrt(varz);

eta = logspace(-5,3,5e2);
figure
for i = 1:max(size(varz))
    %Generate Y
    [Y, Trgt] = genYsamemean(sigx,sigz(i),Niter,A,P0);

    %ROC values
    Pd = sum(and(PyH1(Y,varx,A) >= PyH0(Y,varz(i),A)*eta,
         Trgt))/sum(Trgt);
```

```
    Pf = sum(and(PyH1(Y,varx,A) >= PyH0(Y,varz(i),A)*eta, ~Trgt))/sum(~Trgt);

    plot(Pf,Pd, 'DisplayName', ['\sigma^2_z/\sigma^2_x= ',num2str(varz(i)/varx)], 'linewidth', 1)
    hold on
end
legend
xlabel('Pf')
ylabel('Pd')
title('ROC plot - Same \mu different \sigma^2')
xlim([0,1])

%%

function [Y, Trgt] = genYsamestd(Amag, Niter, var, P0)
%Generate a Y vector for two distributions of variance var and mean difference
%Amag, where the probability of target being there is P0 (Niter is the
%number of elements)
X = sqrt(var)*randn(Niter,1);
Trgt = (rand(Niter,1)>P0);
A = Amag*double(Trgt);
Y = A+X;
end

function [Pd,Pf] = getROC(Amag, var, eta, Y, Trgt)
%Given the Amag, and eta, determines the point where H1 becomes more likely
%than H0 (gam) and then go through the Y and Trgt vectors to determine
%the probability of true positives (Pd) and false positives (Pf).
    gam = Amag./2+var*log(eta)*ones(size(Amag))./(Amag);
    Pd = sum(and(Y>gam,Trgt))./sum(Trgt);
    Pf = sum(and(Y>gam,~Trgt))/sum(~Trgt);
end

function [Y, Trgt] = genYsamemean(sigx,sigz,Niter,A,P0)
%Generate a Y vector for two distributions of std sigx and sigz and same
%mean A, where the probability of target being there is P0 (Niter is the
%number of elements)
    X = sigx*randn(Niter,1);
```

```
    Z = sigz*randn(Niter,1);
    Trgt = (rand(Niter,1)>P0);
    Y = A+X.*Trgt+Z.*(~Trgt);
end
```

2 MATLAB Code for Part 2

```
%%% Stoch Proj 4
% Yuval Epstain Ofek & Jason Kurian
%% Part 2
clear all;close all;clc

%loading the data
load( 'Iris.mat');
%shuffling data randomly
data = [features labels];
rand_pos = randperm(length(data));
data_shuf = zeros(150,5);
for ii = 1:length(data)
    data_shuf(ii,:) = data(rand_pos(ii),:);
end
% split data 50/50 into training and testing sets
trainset = data_shuf(1:2:end,:);
testset = data_shuf(2:2:end,:);

% split data back into features and labels
trainlabels = trainset(:,5);
trainfeatures = trainset(:,1:4);
testlabels = testset(:,5);
testfeatures = testset(:,1:4);

% implement MAP classifier
mu = zeros(length(testset),4);
var = zeros(4,4,length(testset));
likelihoods = zeros(length(testset),3);
% for each label in the training set, find the sample mean vector and
% covariance matrices
for ii = 1:3
    mu(ii,:) = mean(trainset(trainlabels==ii,1:4));
    var(:,:,ii) = cov(trainset(trainlabels==ii,1:4));
    % evaluate likelihood of test features for each label
    likelihoods(:,ii) = mvnpdf(testfeatures,mu(ii,:),var(:,:,ii));
end
% for the confusion function, create the targets matrix
```

```
    where each index of
% 1 indicates which of the test labels is represented
targets = [1:length(testlabels);testlabels';ones(1,length
    (testlabels))]';
targets = full(spconvert(targets))';
[C,CM,~,~] = confusion(targets,likelihoods')
```