Programming Assignment 2
Stochastics & Random Variables - ECE 302

March 27, 2017

Group Members: Thomas Koch
Camilo Gaitan
Mateen Nemati

Introduction

This assignment explores the utility of simple detection and classification methods using simulations in MATLAB. Realistic scenarios in radar and optic communications were presented and detection methods such as the Neyman Pearson likelihood test were used to make predictions based on stochastic, simulated data. Additionally, a multi-class Neyman Pearson test was used to classify the Iris Dataset[1].
The performance of these detectors and classifiers was analyzed primarily using the ROC curve generated by the classifier over a wide range of thresholds, noting that the performance metric is particularly the area under the ROC. Detectors/classifiers which could achieve a high probability of detection (P_d) while admitting a low probability of false alarm (P_{fa}) were to perform well. Additionally, detectors/classifiers which were able to achieve good accuracy with low Signal to Noise Ratio (SNR) (or similar metric to SNR) were considered to perform well.

I - Radar Detection

A radar detection system was implemented as follows. Simulated sampled data points were generated according to two hypothesis: the target was present or not. If the target was present, a data point would be generated as Y = A + X where A is a known constant and X is N(0,σ^2). If the target was not present, the data point would be generated as Y = X. The ratio of A to σ^2 was taken to be the SNR.

1 Experiments

We first attempted to use the MAP (maximum a-priori) detection rule to detect the presence of the signal (H_1). The MAP rule is as follows.

$$max \quad p(s|r) = \frac{p(r|s)p(s)}{p(r)} \tag{1}$$

Here, the maximum is taken with respect to all possibly sent signals, s, and r refers to the received signal (Y). Since the denominator in (1) is the same over all sent signals, the problem reduces to maximizing the numerator over the two possible signals Y = A, and Y = A + X where X $\sim$ N(0,σ^2). This reduces to the following problem.

$$p(r|H_0)p(H_0) \sim \ p(r|H_1)p(H_1) \tag{2}$$

Here, p(r|x) is Gaussian with mean 0 if x = H_0 and mean A if x = H_1. In this experiment, p(H_0) = .8 and p(H_1) = .2. If the left hand side of (2) is greater than the right hand side, the detector selects H_0 as the correct answer. We can then calculate the theoretical accuracy of this detector by calculating the probability of error. Using the total probability theorem we write

$$P_e = P(H_0)P(say\ H_1|sent\ H_0) + P(H_1)P(say\ H_0)|sent\ H_1) \tag{3}$$

In order to calculate P(say H_1 | sent H_0) we note that .8 * f_0(x) < .2 * f_1(x) where f_i is the Gaussian pdf corresponding to the i'th hypothesis and x $\sim$ N(0,σ^2). It then follows that, for a false alarm, the following condition must hold on x.

$$x > \frac{2\sigma^2 ln(\frac{P_0}{P_1}) + A^2}{2A} \tag{4}$$

Since x $\sim$ N(0,σ^2), the probability that x satisfies the condition in (4) is

$$P_{fa} = 1 - G(\frac{2\sigma^2 ln(\frac{P_0}{P_1}) + A^2}{2A\sigma}) \tag{5}$$

where G(.) is the Gaussian cdf.
For the probability of a failed detection, one simply reverses the inequality in (4) where x $\sim$ N(A,σ^2) which leads to the following probability of failed detection

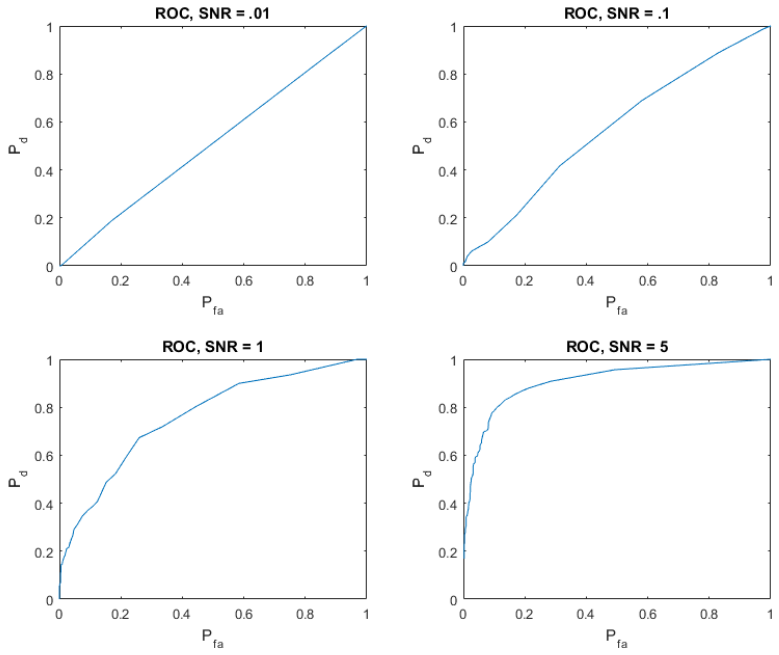
$$P_{fd} = G(\frac{2\sigma^2 ln(4) + A^2}{2A\sigma} - A) \tag{6}$$

Numerically, this results in a probability of error of about 4.27%. In MATLAB, a typical simulated probability of error was 4.28%.

We further evaluated this detector by observing how the ROC varied with respect to the SNR. Figure 1 displays the general trend of the ROC as the SNR was increased. At very low SNR values, the detector randomly guesses, but

when a decent SNR (5) is used, the detector is able to achieve a fairly high probability of detection for a low probability of error. These curves were generated using the Neyman Pearson detection scheme which involved comparing the likelihood ratio of the pdfs to a variable threshold, η.

Figure 1: ROC of the detector as SNR varies.

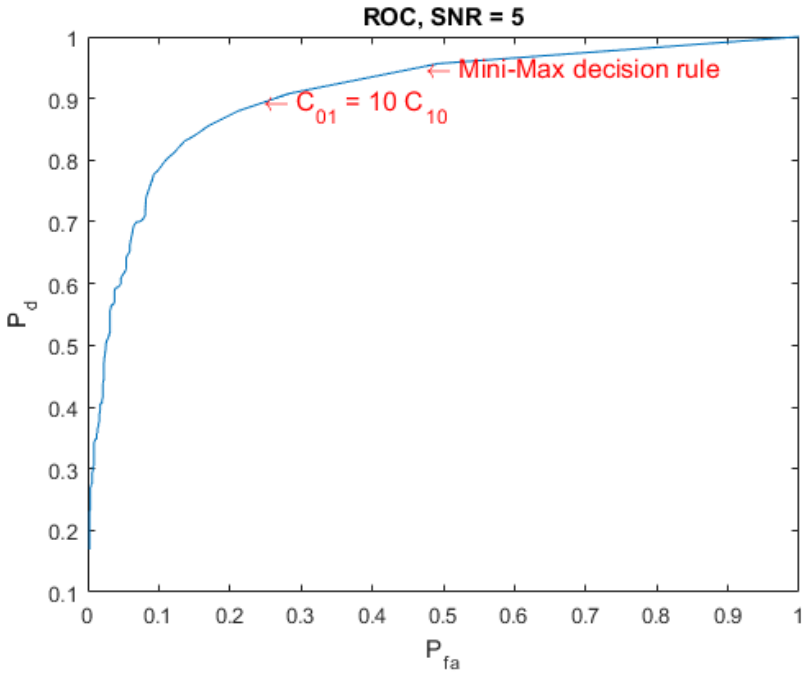


As a study, we then assumed that failing to detect the target should be penalized ten times more than falsely detecting a target. We were interested in the decision rule that would minimize the conditional risk. We first noted that the definition of eta, the threshold in the Neyman Pearson test was defined as follows

$$\eta = \frac{p_0(C_{10} - C_{00})}{p_1(C_{01} - C_{11}} \tag{7}$$

Here p_i refers to the prior probabilities and C_ij refers to the costs. We took $C_ii = 0$ for i = 1,2 which meant there was no penalty for a correct detection. Noting our aforementioned penalty condition, we took $C_{01} = 10C_{10}$ which led to a numerical value of $\eta = .4$. The resulting decision rule is labeled on the ROC in figure 2.

Figure 2: ROC of detector with SNR = 5. The points corresponding to the detectors which heavily penalize a missed detect and the minimax decision rule have been labeled.



Assuming that $C_{01} = 10C_{10}$ as before, we then analyzed the expected cost (risk) as a function of the prior probabilities. The formula for risk is (taking C_{ii} = 0 for i = 1,2)

$$R = E[C] = C_{10}P_0P_{fa} + C_{01}P_1P_{fd} \tag{8}$$

where P_{fd} refers to the probability of a failed detection. The P_{fa} and P_d were theoretically calculated similar to the methods used in (4) as follows. The risk in (8) was plotted in figure 3.

$$P_{fa} = 1 - G(\frac{2\sigma^2 ln(\eta) + A^2}{2A\sigma}) \tag{9}$$

$$P_{fd} = G(\frac{2\sigma^2 ln(\eta) + A^2}{2A\sigma} - A) \tag{10}$$

Another detection rule we studied in this scenario was the minimax decision rule. The minimax decision rule minimizes the maximum risk at some fixed SNR. It reduces to the following relation.

$$C_{01}P_{fd} = C_{10}P_{fa} \tag{11}$$

Plugging (11) into (8) gives the following expression for the risk associated with minimax decision rule. The risk associated with (12) was plotted in figure 3. The

value of η corresponding to the minimax decision rule could not be calculated, because the priors are not known. Instead, a simple manipulation of (11) gives the following relation.

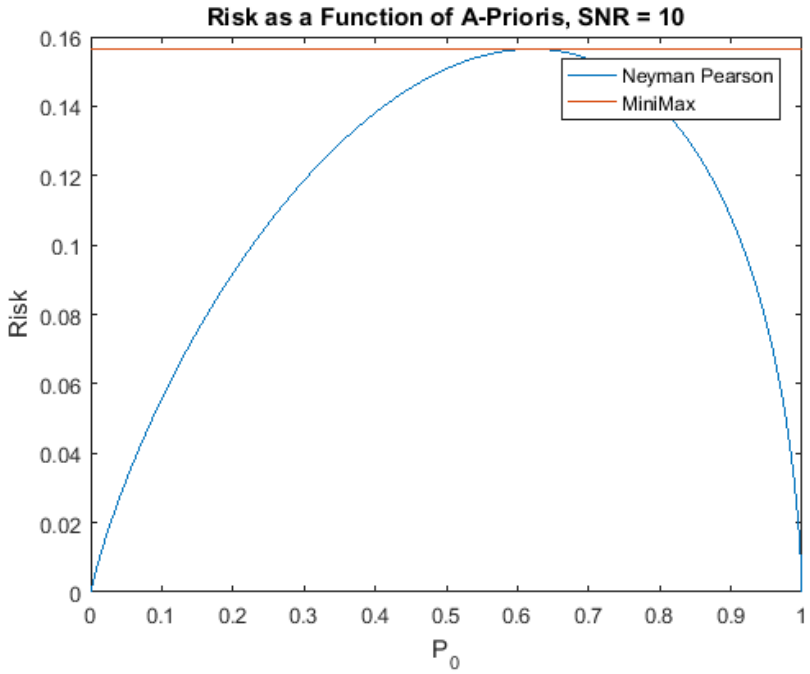
$$P_d = (1 - P_{fm}) = 1 - \frac{P_{fa}}{10} \tag{12}$$

The intersection of the line specified in (12) and the ROC in figure 2 was found and the minimax decision rule was labeled.

$$R = E[C] = C_{01} P_{fd} \tag{13}$$

Again, η is .4 and the SNR was chosen to be 10. C_{10} was taken arbitrarily to be 1, as we were only interested in the general trends of the risk as the priors were varied.
Figure 3 was generated by varying the priors and evaluating (8) and (13). Notably, (13) was calculated by finding the η at which the risk in (8) was maximized. The P_{fd} was then calculated using this value of eta.

Figure 3: Risk plotted as function of the prior probability of H_0 being true. The risk was plotted for both the minimax and Neyman Pearson decision rules.



We then slightly modified the problem and defined new a-priori conditions on the sent and received signals. We modeled the signal corresponding to a target present Y = A + X, where A is a constant and X $\sim$ N(0,σ_x^2), and the

signal corresponding to no target present Y = A + Z where Z $\sim$ N(0,σ_z^2). Here $\sigma_z > \sigma_x$. The same prior probabilities were used to generate the data.
To calculate the theoretical accuracy of a MAP detector, (3), modifications to (5) and (6) need to be made. Let us first calculate the P_{fa} for this detector. We assume that x $\sim$ N(0,σ_z^2) and note that if H_1 is said to be true by our MAP rule then from (2) we have

$$P_0 f_0(x) < P_1 f_1(x) \tag{14}$$

where f_0 and f_1 are Gaussian pdf's of mean A and variance σ_z^2 and σ_x^2 respectively. Simplifying and grouping terms yields the following relation between x and known variables.

$$(x-A)^2(\sigma_z^2 - \sigma_x^2) < 2\sigma_z^2\sigma_x^2 ln(\frac{P_0\sigma_x}{P_1\sigma_z}) \tag{15}$$

We then have the following two regions for x under which a false alarm could be issued by the detector.

$$-\frac{\sigma_x}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z}) < \frac{x-A}{\sigma_z} < \frac{\sigma_x}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z}) \tag{16}$$

Since x is N(0,σ_z^2), the probability of false alarm is then the probability it falls into the region specified by (16) which is given by the (17).

$$P_{fa} = G(\frac{\sigma_x}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z})) - G(-\frac{\sigma_x}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z})) \tag{17}$$

Similarly, the regions x must fall into which warrant a failed detection are given by

$$\frac{x-A}{\sigma_x} > \frac{\sigma_z}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z}) \tag{18}$$

$$\frac{x-A}{\sigma_x} < -\frac{\sigma_z}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z}) \tag{19}$$

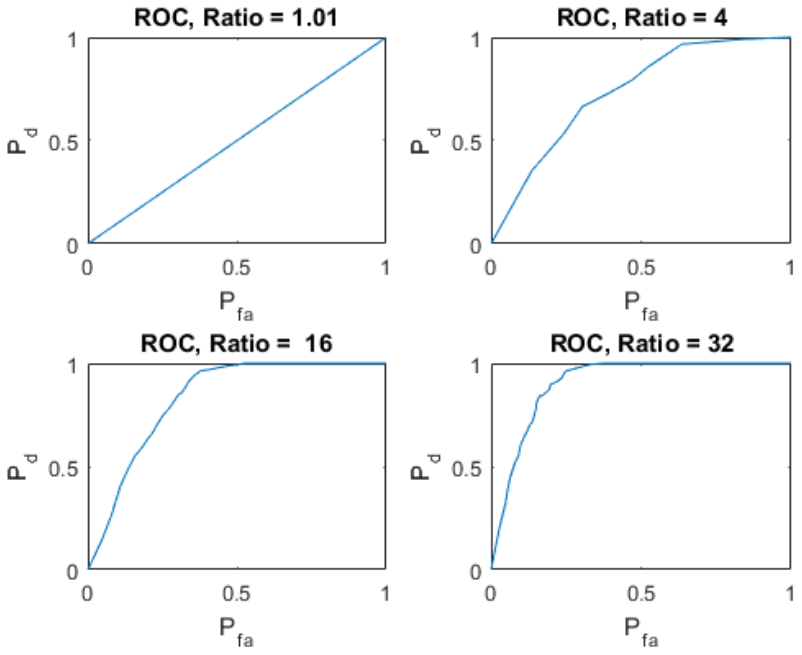
This results in a probability of failed detection of

$$P_{fd} = 1 - G(\frac{\sigma_z}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z})) + G(-\frac{\sigma_z}{\sqrt{2(\sigma_z^2-\sigma_x^2)}} ln(\frac{P_0\sigma_x}{P_1\sigma_z})) \tag{20}$$

We note that, in order for the regions in (16), (18) and (19) to not overlap (no negative signs), and therefore have the probability expressions in (17) and (20) be valid, the value inside the natural logarithm must stay above one. This results in the restriction that the ratio of σ_z^2 to σ_x^2 be less than 16 (for the purposes of accuracy calculations).
We let the ratio of σ_z^2 to σ_x^2 be 15, set A = 1, and $\sigma_x^2 = 1$. Numerically evaluating (3) for this scheme yielded a theoretical P_e of about 0.2001. This

simulated average probability of error was about 0.1996.
A few ROC curves were generated for different values of σ_z^2 / σ_x^2 and the results are shown in figure 4. As the discrepancy between the two noise variances increases, the detector is able to successfully achieve a high P_d while sacrificing a low P_{fa}.

Figure 4: ROC for different values of σ_z^2 to σ_x^2.

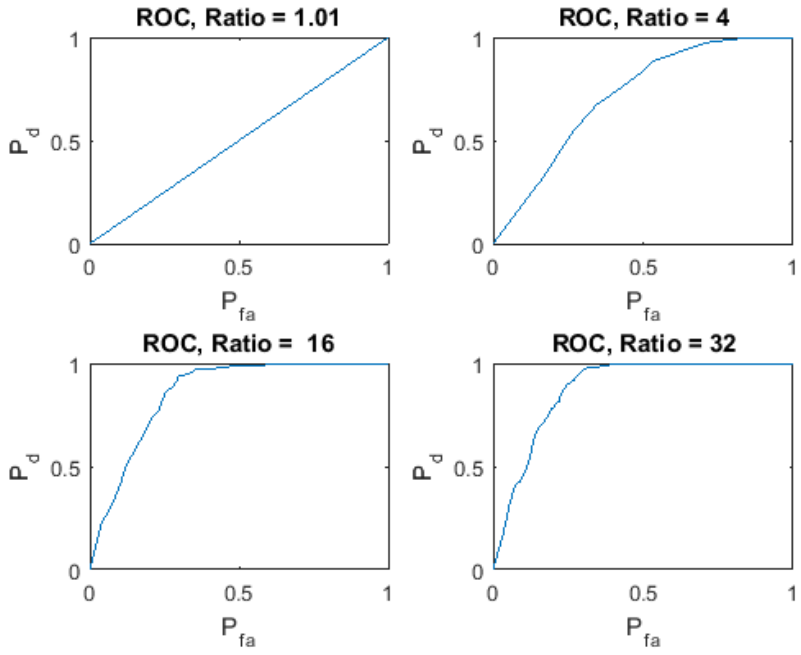


II - Photon Detection

A similar scenario was presented - we are tasked with detecting the presence of photons. Photon arrivals are typically modeled as exponential random variables. These exponential random variables are parameterized with a "rate", λ. In our scheme, we assumed that if photons were present, they would arrive at the detector at a significantly greater rate than if they were not. Therefore we modeled a signal which indicated photons were present as an exponential random variable with rate λ_1 and not present as an exponential random variable with rate $\lambda_0 < \lambda_1$.
In this scenario, it was assumed that as the discrepancy between the rates increased the detector would do a better job at classifying the signals correctly. Figure 5 shows ROC's of the photon detector for a variety of ratios of λ_1 to λ_0. The figure confirms our prediction that this increased ratio detected photons

better, as is evidenced by the larger area under the curve for larger ratios.

Figure 5: ROC of photon detector for different values of λ_1 / λ_0.



Part III - Introduction to Pattern Classification and Machine Learning

Satisfied with the performance in both radar and photon detection schemes, we decided to move to a multi-class problem. We were tasked with classifying the Iris dataset which consists of 150 instances. Each of these instances has four features and belongs to one of three classes.
The MAP rule for classifying the data was used. Since each class was assumed to be equally likely (and indeed it was), this rule was equivalent to the ML decision rule. So for each x, the distribution which maximized the probability was selected as the correct class to which x belonged. In order to calculate the distributions, it was assumed x belonged to one of three multi-variate Gaussian distributions such as the following

$$f(x) = (2\pi)^{-k/2}\Sigma^{-1/2} \exp \frac{-(x-\mu)'\Sigma^{-1}(x-\mu)}{2} \tag{21}$$

where x $\in R^k$ (k=4 in this example), μ = E[x] and Σ is the k x k covariance matrix of x. The mean and covariance matrices of x were not known for each

class, so they had to be estimated from the data.
Half of the data was used as a training set from which the mean and covariance matrices were estimated. The other half was used as a testing set over which the decision rule was evaluated. It was important to not use the test data to estimate the mean and covariance matrices, because the decision rule must not have any knowledge of the test data prior to classification. The accuracy and confusion matrices of the classifier were determined. A typical accuracy of 88% was achieved using this classifier.
Tests were also run with the dimensionality of the dataset reduced. When one feature was removed, a typical accuracy of the classifier was around 80% or more - little to no difference from the classifier presented with all the information. When two or more features were removed, however, the classifier struggled to produce prediction accuracies greater than 70% on the test set which shows that, with less information the classifier does not perform well. It is important to note that 70% accuracy is far better than random guessing.
The confusion matrix revealed that, no matter the dimensionality, the classifier seemed to confuse classes two and three the most. That is, the classifier incorrectly labeled many examples as class two which were actually class three.

Overall Conclusions

The utility of simple detection and classification algorithms was assessed in three scenarios: a radar detection problem, a photon detection problem, and a three-class classification problem. It was shown that basic detection rules, notably MAP, performed well.
It was notably shown that a good "SNR" was necessary for any of these classification problems to work. By "SNR" we generally mean a certain parameter that, when large, distinguishes two or more hypothesis with certainty. For the radar detection problem, this "SNR" was an actual SNR. For the classification problem, this SNR could be interpreted as the KL divergence between the underlying probability distributions of the two classes. In the latter case, one cannot easily vary the SNR which presents a more difficult classification problem.

Appendix

MATLAB Code

What follows is the MATLAB code used to simulate the environments discussed in this paper.

```
clear all; close all; clc;
% Koch Gaitan Nemati HW2 Stoch
%% Part 1 radar Detection

snr = 10;
```

```
not_present_prob = .8;
num_sims = 1e3;
A = 1;
var = A/snr;
format longG;

accuracys = zeros(1,num_sims); %get an average accuracy for MAP
for l = 1:num_sims
    x = zeros(2,num_sims);
    for i = 1:num_sims
        if rand > not_present_prob %is present
            x(1,i) = sqrt(var) * randn(1) + A;
            x(2,i) = 1;
        else %isnt present
            x(1,i) = sqrt(var) * randn(1);
            x(2,i) = 0;
        end
    end
    pdf_there = 1/(sqrt(2 * pi * var)) * exp(-(x(1,:) - A) .^ 2 / (2 * var))
        ;
    pdf_not_there = 1/(sqrt(2 * pi * var)) * exp(-x(1,:) .^2 / (2 * var));
    predictions = (pdf_there * (1 - not_present_prob) > pdf_not_there *
        not_present_prob);
    accuracys(l) = sum(predictions == x(2,:)) / num_sims; %error rate is 1
        - accuracy
end
accuracy = mean(accuracys);

%ROC Calculations
snrs = [.01 .1 1 5];
etas = linspace(-100,100,num_sims);
pds = zeros(length(snrs),length(etas)); %storage for P_d's
pfas = zeros(length(snrs),length(etas)); %storage for P_fa's
for i = 1:length(snrs)
    snr = snrs(i);
    var = A / snr;
    x = zeros(2,num_sims);
    for j = 1:num_sims %generate the data at this snr
        if rand > not_present_prob %is present
            x(1,j) = sqrt(var) * randn(1) + A;
            x(2,j) = 1;
        else %isnt present
            x(1,j) = sqrt(var) * randn(1);
            x(2,j) = 0;
        end
    end
```

```
    pdf_there = 1/(sqrt(2 * pi * var)) * exp(-(x(1,:) - A) .^ 2 / (2 * var))
        ;
    pdf_not_there = 1/(sqrt(2 * pi * var)) * exp(-x(1,:) .^2 / (2 * var));
    for j = 1:length(etas) %neyman pearson test
        eta = etas(j);
        predictions = (pdf_there ./ pdf_not_there > eta);
        pds(i,j) = sum((predictions + x(2,:)) == 2) / sum(x(2,:) == 1);
        pfas(i,j) = sum((predictions - x(2,:)) == 1) / sum(x(2,:) == 0);
    end
end

figure();
subplot(2,2,1); plot(pfas(1,:),pds(1,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, SNR = .01');
subplot(2,2,2);plot(pfas(2,:),pds(2,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, SNR = .1');
subplot(2,2,3); plot(pfas(3,:),pds(3,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, SNR = 1');
subplot(2,2,4); plot(pfas(4,:),pds(4,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, SNR = 5');

%assume eta = (.8 * 1) / (.2 * 10)
factor = 10;
eta = not_present_prob / ((1 - not_present_prob) * factor);
predictions = (pdf_there ./ pdf_not_there > eta);
pd = sum((predictions + x(2,:)) == 2) / sum(x(2,:) == 1);
pfa = sum((predictions - x(2,:)) == 1) / sum(x(2,:) == 0);
figure();
plot(pfas(4,:),pds(4,:)); xlabel('P_f_a'); ylabel('P_d'); title('ROC, SNR
    = 5');
text(pfa,pd,'\leftarrow C_0_1 = 10 C_1_0','Color','red','FontSize',12);

%minimax decision is pd = 1 - pfa/10
pfas_mm = linspace(0,1,num_sims);
pds_mm = 1 - pfas_mm/10;
[x,y] = intersections(pfas(4,:),pds(4,:),pfas_mm,pds_mm,1); %external
    function
text(x,y,'\leftarrow Mini-Max decision rule','Color','red','Fontsize',12);


%plot risk as a-prioris are varied at SNR = 5
P_0 = linspace(0,1,num_sims);
P_1 = ones(1,num_sims) - P_0;
snr = 10;
var = A / snr;
eta = P_0 ./ (factor * P_1);
```

```
pfa = 1 - normcdf((2 * var * log(eta) + A^2) / (2 * A * sqrt(var)));
pfd = normcdf(((2 * var * log(eta) + A^2) / (2 * A) - A) / sqrt(var));
risks = P_0 .* pfa + factor * P_1 .* pfd;

figure();
plot(P_0,risks); xlabel('P_0'); ylabel('Risk'); title('Risk as a Function of
    A-Prioris, SNR = 10');


% assume aprioris not known
% 'decision rule' : p_d = 1 - (p_f_a / 10) (figure 2 shows decision rule)
% risk = p_f_a at some eta which minimizes maximum risk
[biggest,i] = max(risks);
eta = eta(i);
pfa = 1 - normcdf((2 * var * log(eta) + A^2) / (2 * A * sqrt(var)));
hold on; plot(P_0,pfa * ones(1,num_sims));
legend('Neyman Pearson','MiniMax');


% assume not there -> Y = A + Z, Z is N(O,var_z)
% there -> Y = A + X, X is N(0,var_x), var_x < var_z

ratio_z_xs = [1.01 4 16 32];
x = zeros(2,num_sims);
A = 1;
var_x = 1;
var_z = 15 * var_x;
accuracys = zeros(1,num_sims);
for l = 1:num_sims
    for i = 1:num_sims
        if rand > not_present_prob %is present
            x(1,i) = sqrt(var_x) * randn(1) + A;
            x(2,i) = 1;
        else %isnt present
            x(1,i) = sqrt(var_z) * randn(1) + A;
            x(2,i) = 0;
        end
    end

    pdf_there = 1/(sqrt(2 * pi * var_x)) * exp(-(x(1,:) - A) .^ 2 / (2 *
        var_x));
    pdf_not_there = 1/(sqrt(2 * pi * var_z)) * exp(-(x(1,:) - A) .^2 / (2 *
        var_z));
    predictions = ((pdf_there * (1 - not_present_prob)) > (pdf_not_there
        * not_present_prob));
    accuracys(l) = sum(predictions == x(2,:)) / num_sims;
```

```
end
accuracy = 1-mean(accuracys); %error rate is 1 - accuracy
%ROC Calculations
etas = linspace(-100,100,num_sims);
pds = zeros(length(ratio_z_xs),length(etas)); %storage for P_d's
pfas = zeros(length(ratio_z_xs),length(etas)); %storage for P_fa's
for i = 1:length(ratio_z_xs)
    var_z = var_x * ratio_z_xs(i);
    x = zeros(2,num_sims);
    for j = 1:num_sims %generate the data at this snr
        if rand > not_present_prob %is present
            x(1,j) = sqrt(var_x) * randn(1) + A;
            x(2,j) = 1;
        else %isnt present
            x(1,j) = sqrt(var_z) * randn(1) + A;
            x(2,j) = 0;
        end
    end
    pdf_there = 1/(sqrt(2 * pi * var_x)) * exp(-(x(1,:) - A) .^ 2 / (2 *
        var_x));
    pdf_not_there = 1/(sqrt(2 * pi * var_z)) * exp(-(x(1,:) - A) .^2 / (2 *
        var_z));
    for j = 1:length(etas) %neyman pearson test
        eta = etas(j);
        predictions = (pdf_there ./ pdf_not_there > eta);
        pds(i,j) = sum((predictions + x(2,:)) == 2) / sum(x(2,:) == 1);
        pfas(i,j) = sum((predictions - x(2,:)) == 1) / sum(x(2,:) == 0);
    end
end

figure();
subplot(2,2,1); plot(pfas(1,:),pds(1,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, Ratio = 1.01');
subplot(2,2,2);plot(pfas(2,:),pds(2,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, Ratio = 4');
subplot(2,2,3); plot(pfas(3,:),pds(3,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, Ratio = 16');
subplot(2,2,4); plot(pfas(4,:),pds(4,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, Ratio = 32');



%% Part 2 Photon Detection

rate_0 = 5;
```

```
%ROC Calculations
rate_1s = [6 10 15 50];
etas = linspace(-100,100,num_sims);
pds = zeros(length(snrs),length(etas)); %storage for P_d's
pfas = zeros(length(snrs),length(etas)); %storage for P_fa's
for i = 1:length(rate_1s)
    rate_1 = rate_1s(i);
    x = zeros(2,num_sims);
    for j = 1:num_sims %generate the data at these rates
        if rand > .5 %rate 1
            x(1,j) = exprnd(1/rate_1);
            x(2,j) = 1;
        else %rate 0
            x(1,j) = exprnd(1/rate_0);
            x(2,j) = 0;
        end
    end
    pdf_there = exppdf(x(1,:),1/rate_1);
    pdf_not_there = exppdf(x(1,:),1/rate_0);
    for j = 1:length(etas) %neyman pearson test
        eta = etas(j);
        predictions = (pdf_there ./ pdf_not_there > eta);
        pds(i,j) = sum((predictions + x(2,:)) == 2) / sum(x(2,:) == 1);
        pfas(i,j) = sum((predictions - x(2,:)) == 1) / sum(x(2,:) == 0);
    end
end

figure();
subplot(2,2,1); plot(pfas(1,:),pds(1,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, \lambda_1 = 6, \lambda_0 = 5');
subplot(2,2,2);plot(pfas(2,:),pds(2,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, \lambda_1 = 10, \lambda_0 = 5');
subplot(2,2,3); plot(pfas(3,:),pds(3,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, \lambda_1 = 15, \lambda_0 = 5');
subplot(2,2,4); plot(pfas(4,:),pds(4,:)); xlabel('P_f_a'); ylabel('P_d');
    title('ROC, \lambda_1 = 20, \lambda_0 = 5');


%% Intro to Pattern Classification and ML

data = csvread('iris.csv');
%grab half of each class for train and test
train_ex = [data(1:24,1:4); data(51:75,1:4); data(101:125,1:4)];
train_lab = [data(1:24,5); data(51:75,5); data(101:125,5)];
test_ex = [data(25:50,1:4); data(76:100,1:4); data(126:150,1:4)];
test_lab = [data(25:50,5); data(76:100,5); data(126:150,5)];
```

```
% estimate parameters associated with each class assuming each class
% follows a gaussian distribution
mu_hat_1 = mean(train_ex(1:24,:));
cov_hat_1 = cov(train_ex(1:24,:));
mu_hat_2 = mean(train_ex(25:49,:));
cov_hat_2 = cov(train_ex(25:49,:));
mu_hat_3 = mean(train_ex(50:74,:));
cov_hat_3 = mean(train_ex(50:74,:));

predictions = zeros(length(test_ex),1);
%calculate the class according to the likelihood test with eta = 1
for i = 1:length(test_ex)
    pred = 1;
    tmp = test_ex(i,:);
    if mvnpdf(tmp,mu_hat_2,cov_hat_2) > mvnpdf(tmp,mu_hat_1,
        cov_hat_1)
        pred = 2;

        if mvnpdf(tmp,mu_hat_3,cov_hat_3) > mvnpdf(tmp,mu_hat_2,
            cov_hat_2)
            pred = 3;
        end
    elseif mvnpdf(tmp,mu_hat_3,cov_hat_3) > mvnpdf(tmp,mu_hat_1,
        cov_hat_1)
        pred = 3;
    end

    predictions(i) = pred;
end

%results
acc = sum(predictions == test_lab) / length(predictions);
c_mat = confusionmat(test_lab,predictions);

% reduce the dimensionality
train_ex = [data(1:24,1:2); data(51:75,1:2); data(101:125,1:2)];
train_lab = [data(1:24,5); data(51:75,5); data(101:125,5)];
test_ex = [data(25:50,1:2); data(76:100,1:2); data(126:150,1:2)];
test_lab = [data(25:50,5); data(76:100,5); data(126:150,5)];

% estimate parameters associated with each class assuming each class
% follows a gaussian distribution
mu_hat_1 = mean(train_ex(1:24,:));
cov_hat_1 = cov(train_ex(1:24,:));
mu_hat_2 = mean(train_ex(25:49,:));
```

```
cov_hat_2 = cov(train_ex(25:49,:));
mu_hat_3 = mean(train_ex(50:74,:));
cov_hat_3 = mean(train_ex(50:74,:));

predictions = zeros(length(test_ex),1);
%calculate the class according to the likelihood test with eta = 1
for i = 1:length(test_ex)
    pred = 1;
    tmp = test_ex(i,:);
    if mvnpdf(tmp,mu_hat_2,cov_hat_2) > mvnpdf(tmp,mu_hat_1,
        cov_hat_1)
        pred = 2;

        if mvnpdf(tmp,mu_hat_3,cov_hat_3) > mvnpdf(tmp,mu_hat_2,
            cov_hat_2)
            pred = 3;
        end
    elseif mvnpdf(tmp,mu_hat_3,cov_hat_3) > mvnpdf(tmp,mu_hat_1,
        cov_hat_1)
        pred = 3;
    end

    predictions(i) = pred;
end

%results -- reduced dimensionality
acc = sum(predictions == test_lab) / length(predictions)
c_mat = confusionmat(test_lab,predictions)
```