Problem Set 2

MAS 622J/1.126J: Pattern Recognition and Analysis

Due: 5:00 p.m. on October 12

[Note: All instructions to plot data or write a program should be carried out using Matlab. In order to maintain a reasonable level of consistency and simplicity we ask that you do not use other software tools.]

If you collaborated with other members of the class, please write their names at the end of the assignment. Moreover, you will need to write and sign the following statement: "In preparing my solutions, I did not look at any old homeworks, copy anybody's answers or let them copy mine."

Problem 1: Discriminability and ROC [20 points]

Please download the datasets (dataset1.txt, dataset2.txt, dataset3.txt, dataset4.txt) from the course website. Each dataset includes 1-D data samples from two classes w_1 and w_2. The first column and the second column of each dataset specify 1000 samples from class w_1 and 1000 samples from class w_2, respectively.

a. For each dataset, compute the discriminability $d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\sigma_1^2 + \sigma_2^2}}$ where μ_1 and σ_1 are the mean and standard deviation of the distribution of class w_1, and μ_2 and σ_2 are the mean and standard deviation of the distribution of class w_2.

   Solution: Please refer to the Matlab code for (b). Approximately, For dataset1: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 1$, $\sigma_2 \simeq 1$, $d' \simeq 1.39$
   For dataset2: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 4$, $\sigma_2 \simeq 1$, $d' \simeq 0.46$
   For dataset3: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 1$, $\sigma_2 \simeq 4$, $d' \simeq 0.45$
   For dataset4: $\mu_1 \simeq 10$, $\mu_2 \simeq 12$, $\sigma_1 \simeq 4$, $\sigma_2 \simeq 4$, $d' \simeq 0.31$

b. Now we compute the ROC curve for each dataset. Please plot four ROC curves in the same figure. To do this, we approximate $\mathbf{P_{TP}} = \mathbf{P}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_2)$ by $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_2)/1000$, and $\mathbf{P_{FP}} = \mathbf{P}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_1)$ by $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_1)/1000$. Here, for i =1 or 2, $\mathbf{N}(\mathbf{x} > \mathbf{x}^* | \mathbf{x} \in w_i)$ is denotes the number of samples in class w_i whose value is greater than $\mathbf{x}^*$. Note that $\mathbf{N}$ doesn't denote a normal distribution!

   Solution: Changing the classifier value ($\mathbf{x}^*$) from $-\infty$ to ∞, we can plot the ROC curve for each dataset. Note that when $\mathbf{x}^* = -\infty$, $\mathbf{P_{TP}} = \mathbf{1}$ and $\mathbf{P_{FP}} = \mathbf{1}$ (corresponding to the uppermost/rightmost position in the ROC curve), and when $\mathbf{x}^* = \infty$, $\mathbf{P_{TP}} = \mathbf{0}$ and $\mathbf{P_{FP}} = \mathbf{0}$ (corresponding to the lowermost/leftmost position in the ROC curve).

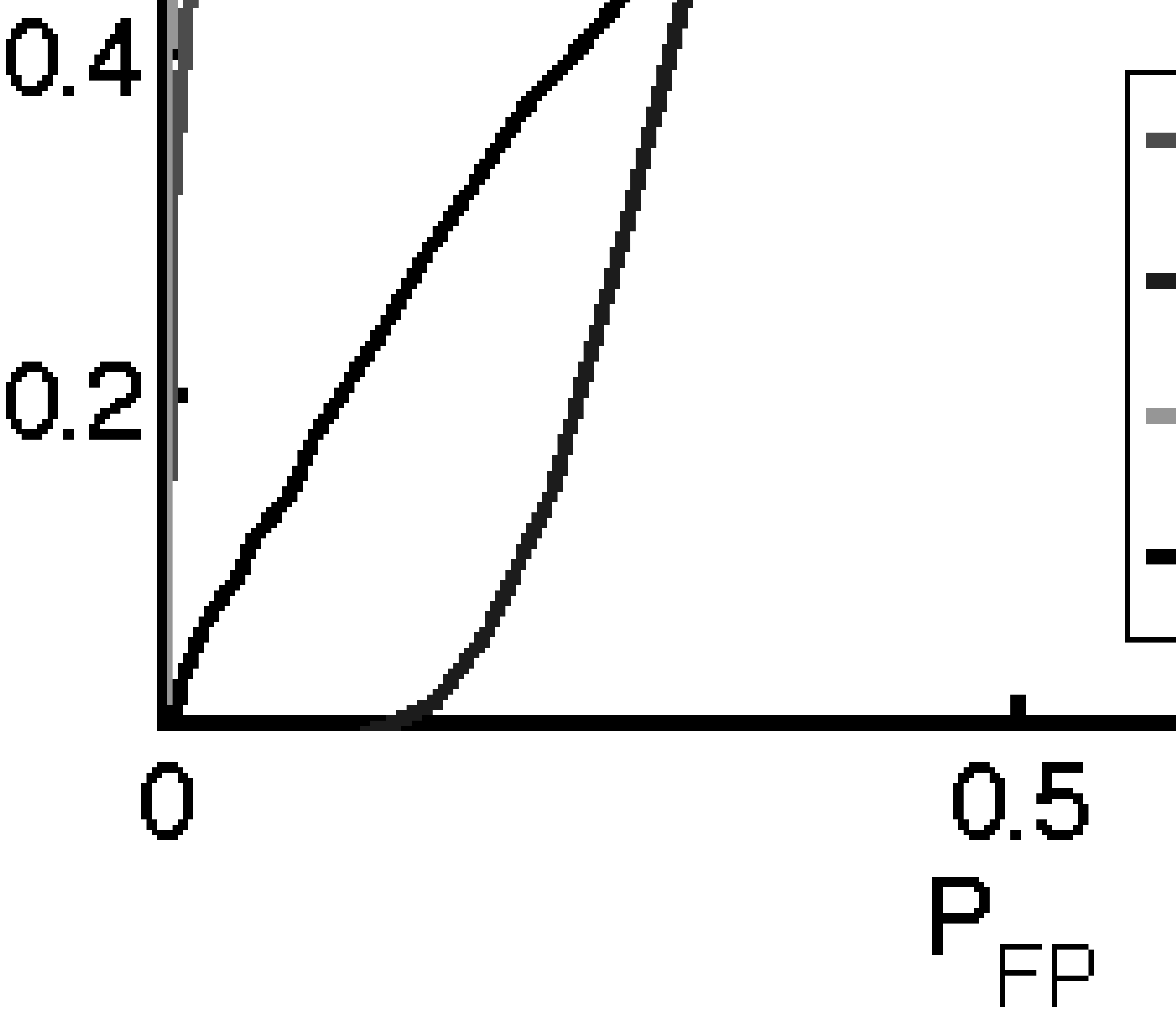
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MIT MAS622j/1.126J Fall 2008, Problem Set 3
```

```
%
% (for Problem 1(a),(b))
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all

num_samples = 1000;  % number of test samples from each class

sprintf('dataset1:')
dd = load('dataset1.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end


mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'
```

```
figure
hold on
plot(Pf_save, Pd_save, 'r')


sprintf('dataset2:')
dd = load('dataset2.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);


Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c =  -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'

plot(Pf_save, Pd_save, 'b')
```

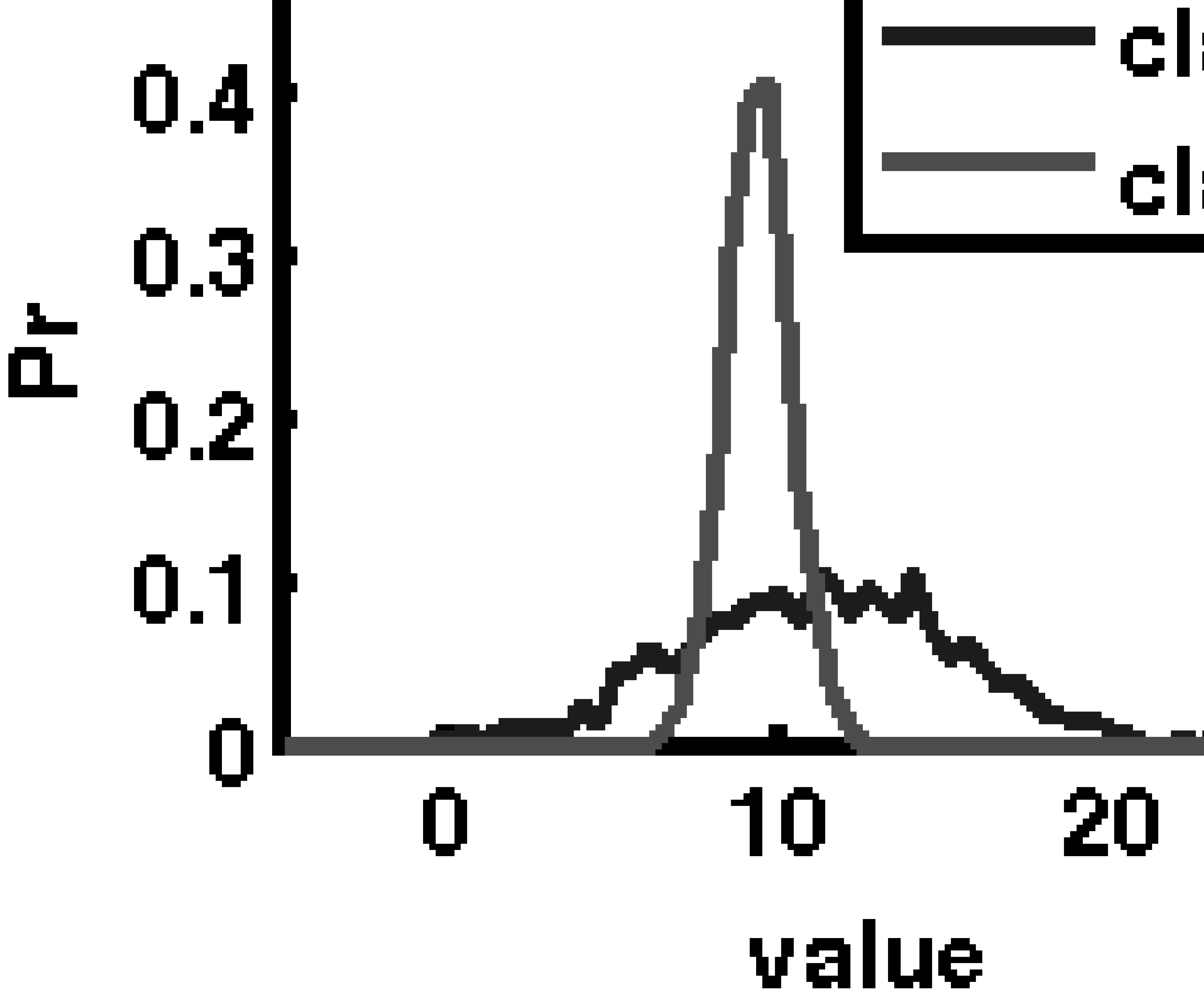
```
sprintf('dataset3:')
dd = load('dataset3.txt');

% test samples from class 1
x1 = dd(:,1);
% test samples from class 2
x2 = dd(:,2);

Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'

plot(Pf_save, Pd_save, 'g')

sprintf('dataset4:')
dd = load('dataset4.txt');

% test samples from class 1
x1 = dd(:,1);
```

```
% test samples from class 2
x2 = dd(:,2);


Pd_save = [];
Pf_save = [];

% x_c is a classifier
for x_c = -10:0.5:30

    % number of TP (True Positive) samples
    nTP = length(find(x2 > x_c));
    % approximate probability of hit, P(x > x_c | x in w=2)
    Pd = nTP / num_samples;

    % number of FP (False Positive) samples
    nFP = length(find(x1 > x_c));
    % approximate probability of false alarm, P(x > x_c | x in w=1)
    Pf = nFP / num_samples;

    Pd_save = [Pd_save; Pd];
    Pf_save = [Pf_save; Pf];

end

mean1 = mean(x1)
mean2 = mean(x2)
sig1 = std(x1)
sig2 = std(x2)
sig = sqrt(sig1^2 + sig2^2)

discrim = abs(mean1 - mean2)/sig % d'


plot(Pf_save, Pd_save, 'k')
xlabel('P_{FP}')
ylabel('P_{TP}')
axis equal
axis([0 1 0 1])

legend('dataset1','dataset2','dataset3','dataset4')

hold off
```



c. For each dataset, plot the two *approximated* probability density functions. Note that the probability density function is the derivative of the cumulative density function. (Do NOT just approximate distributions by Gaussians and draw those approximated Gaussians in this problem.) *Hint:* Use the same method we use in (b) to get the cumulative distribution. This is called *Monte Carlo* method. Also, note that the derivative (i.e., probability density function $p(x)$) relates to the increment of the cumulative density function $P_X(x)$. That is, $p(x) = \Delta P_X(x)/\Delta x$.

Solution:



d. How does the discriminability relate to the ROC curve?

(Solution)

First, comparing dataset1 ($d' = 1.39$) with dataset4 ($d' = 0.31$), we can see that the two distributions in dataset1 are more discriminable than the ones in dataset4 (See the distribution plot in (c)). Also, from the ROC plot in (b), we find that better performance (here, meaning higher $\mathbf{P_{TP}}$ and lower $\mathbf{P_{FP}}$ with the optimal classifier location) is achieved on dataset1, compared with dataset4. In general, for two gaussian distributions, as d' becomes bigger, $\mathbf{P_{TP}} - \mathbf{P_{FP}}$ becomes bigger (Refer to the problem 5 (a) in our Problem Set 2).

Second, comparing dataset2 ($d' = 0.46$) with dataset3 ($d' = 0.45$), we can see that the discriminability of the two distributions in dataset2 is almost the same as that of the ones in dataset3 (See the distribution plot in (c)). Also, from the ROC plot in (b), we find that ROC curves of dataset3 and dataset4 have very different characteristics although the two have similar d' values. Note that depending on the shapes of two distributions (in dataset2, the distribution with higher mean has lower variance, but in dataset3, the distribution with higher mean has higher variance), the characteristic of the corresponding ROC curve is determined. Note that for each dataset, when the change of the classifier value ($\mathbf{x}^*$) happens at around the mean of the distribution with lower variance, the ROC curve ($\mathbf{P_{TP}}$ for dataset2, $\mathbf{P_{FP}}$ for dataset3) radically changes.

Problem 2: (DHS 2.6) Optimal Decision Boundaries [20 points]

A Spanish campany called Goorrel has launched an application to recognize important e-mails (ω_1) vs unimportant e-mails (ω_2.) The company is using two secret features such that their training data is well approximated by two Gaussians:

$$p(\mathbf{x}|\omega_1) \sim \mathcal{N}(\mu_1, \Sigma_1)$$
$$p(\mathbf{x}|\omega_2) \sim \mathcal{N}(\mu_2, \Sigma_2)$$

where $\mu_1 = [0\ 0]^T$, $\mu_2 = [5\ 0]^T$, $\Sigma_1 = \mathrm{I}$, and $\Sigma_2 = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$, where I is the identity matrix.

a. Plot the one-sigma ellipses for these two classes in the place $\mathbf{x} = [x_1\ x_2]^T$.

Solution: The one sigma ellipses are circles of radii 1 and 2 respectively (see Figure 8.)

Figure 1: One sigma ellipses

(MATLAB code is at the end of the problem)

b. The company finds that choosing a threshold at $x_1 = 3$ perfectly separates the training examples they have; thus, they propose that this should be the best classifier. Show them an expression, in terms of $\mathbf{x}$, which can improve their classifier with respect to minimizing the Bayes probability of error. Assume that unimportant emails are three times as likely as important emails.

Solution: Using discriminant functions for each class, we have:

$$g_i(\mathbf{x}) = \mathbf{x}^T W_i \mathbf{x} + \mathbf{w}_i^T + w_i,$$

where

$$\begin{aligned}
W_i &= -\frac{1}{2}\Sigma_i^{-1}, \\
\mathbf{w}_i &= \Sigma_i^{-1}\boldsymbol{\mu}_i, \text{ and} \\
w_i &= \frac{1}{2}\boldsymbol{\mu}_i^T\Sigma_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i).
\end{aligned}$$

Because email from class ω_2 is three times as likely as email from class ω_1, we have:

$$\begin{aligned}
P(\omega_2) &= 3P(\omega_1), \text{ and} \\
P(\omega_1) + P(\omega_2) &= 1. \text{ Resulting in} \\
P(\omega_1) &= \frac{1}{4}, \text{ and} \\
P(\omega_2) &= \frac{3}{4}.
\end{aligned}$$

Then, plugging in our values to the discriminant equations, we have:

$$\begin{aligned}
g_1(\mathbf{x}) &= \mathbf{x}^T(-\frac{1}{2}\Sigma_1^{-1})\mathbf{x} + (\Sigma_1^{-1}\boldsymbol{\mu}_1)^T\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_1) \\
&= -\frac{1}{2}\mathbf{x}^T\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1}\mathbf{x} + \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)^T\mathbf{x} - \frac{1}{2}\begin{bmatrix} 0 \\ 0 \end{bmatrix}^T\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{2}\ln\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} + \ln\frac{1}{4} \\
&= -\frac{1}{2}(x_1^2 + x_2^2) + \ln\frac{1}{4}
\end{aligned}$$

and

$$\begin{aligned}
g_2(\mathbf{x}) &= \mathbf{x}^T(-\frac{1}{2}\Sigma_2^{-1})\mathbf{x} + (\Sigma_2^{-1}\boldsymbol{\mu}_2)^T\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu}_2 - \frac{1}{2}\ln|\Sigma_2| + \ln P(\omega_2) \\
&= -\frac{1}{2}\mathbf{x}^T\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}^{-1}\mathbf{x} + \left(\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}^{-1}\begin{bmatrix} 5 \\ 0 \end{bmatrix}\right)^T\mathbf{x} - \frac{1}{2}\begin{bmatrix} 5 \\ 0 \end{bmatrix}^T\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}^{-1}\begin{bmatrix} 5 \\ 0 \end{bmatrix} - \frac{1}{2}\ln\begin{vmatrix} 4 & 0 \\ 0 & 4 \end{vmatrix} + \ln\frac{3}{4} \\
&= -\frac{1}{8}(x_1^2 + x_2^2) + \frac{5}{4}x_1 - \frac{25}{8} - \frac{1}{2}\ln 16 + \ln\frac{3}{4}
\end{aligned}$$

Now, using the decision criteria, $g_1(\mathbf{x}) > g_2(\mathbf{x})$, to decide when to choose category ω_1, we get:

$$\begin{aligned}
-\frac{1}{2}(x_1^2 + x_2^2) + \ln\frac{1}{4} &> -\frac{1}{8}(x_1^2 + x_2^2) + \frac{5}{4}x_1 - \frac{25}{8} - \frac{1}{2}\ln 16 + \ln\frac{3}{4} \\
-\frac{3}{8}(x_1^2 + x_2^2) - \frac{5}{4}x_1 &> -\frac{25}{8} - \frac{1}{2}\ln 16 + \ln\frac{3}{4} - \ln\frac{1}{4} \\
x_1^2 + x_2^2 + \frac{10}{3}x_1 &< \frac{8}{3}(\frac{25}{8} + \frac{1}{2}\ln 16 - \ln\frac{3}{4} + \ln\frac{1}{4}) \\
(x_1 + \frac{5}{3})^2 + x_2^2 &< \frac{8}{3}(\frac{25}{8} + \frac{1}{2}\ln 16 - \ln\frac{3}{4} + \ln\frac{1}{4}) + \frac{25}{9} \\
(x_1 + \frac{5}{3})^2 + x_2^2 &< 3.45^2
\end{aligned}$$

So, if $\mathbf{x}$ is within the circle described by the above equation, then we should decide ω_1.

c. The shape of this optimal decision boundary is:

- a line
- a parabola
- a hyperbola
- a circle
- an ellipse
- none of the above (explain)

Be sure to justify your answer.

Solution: *a circle*: The decision region is a circle with mean, $\mu_d = [-\frac{5}{3}0]^T$, and radius, $r = 3.45$ (see Figure 2.) This makes sense because both of the Gaussian distributions are circular, so dividing the two distributions, such as in the following form of the discriminant function

$$g(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)} \tag{1}$$

must also be circular.

Figure 2: Decision Boundary ($\lambda_{21} = 1$)

d. If Gorrel accidently misses relevant e-mails, then this will be terrible for the users. The company estimates this error will cost them twenty times as much as the cost of missclasifying unimportant e-mails (there's no cost to choosing correctly in either case.) Describe qualitatively how this changes your result above. Make a sketch of the change (it doesn't have to be precisely plotted). Justify your answer.

Solution: Previously, we had implicitely assumed an equal cost for making a misclassification error, such that $\lambda_{12} = 1$, $\lambda_{21} = 1$, $\lambda_{11} = 0$, and $\lambda_{22} = 0$. Now, we must now minimize the cost function with, $\lambda_{12} = 1$, $\lambda_{21} = 20$, $\lambda_{11} = 0$, and $\lambda_{22} = 0$. Using the minimum-risk decision rule, we can see how things change when the risk is no longer equal for each type of misclassification:

$$(\lambda_{21} - \lambda_{11}) p(\mathbf{x}|\omega_1) P(\omega_1) > (\lambda_{12} - \lambda_{22}) p(\mathbf{x}|\omega_2) P(\omega_2)$$
$$20 p(\mathbf{x}|\omega_1) \frac{1}{4} > p(\mathbf{x}|\omega_2) \frac{3}{4}$$
$$\frac{20}{3} p(\mathbf{x}|\omega_1) > p(\mathbf{x}|\omega_2)$$

We can see that this new decision relationship makes decisions favor ω_1 much more than previously. The shape of the decision boundary will now be a larger circular decision region within which we should choose ω_1 (see Figure 3.)

The new decision boundary looks as follows:

Figure 3: New Decision Boundary ($\lambda_{21} = 20$)

The MATLAB code that generates all of the figures is as follows:

```
close all;
clear all;

%Initialize parameters
mu_1 = [0 0]';
mu_2 = [5 0]';
Sigma_1 = [1 0; 0 1];
Sigma_2 = [4 0; 0 4];
rad_1 = sqrt(Sigma_1(1,1));
rad_2 = sqrt(Sigma_2(1,1));

%Compute points
```

```
x_points_1 = [0:100]*2/100 - 1;
x_points_1 = rad_1*x_points_1 + mu_1(1);
y_points_1 = sqrt(rad_1^2 - (x_points_1 - mu_1(1)).^2);

x_points_2 = [0:100]*2/100 - 1;
x_points_2 = rad_2*x_points_2 + mu_2(1);
y_points_2 = sqrt(rad_2^2 - (x_points_2 - mu_2(1)).^2);


figure;
plot(x_points_1,y_points_1+ mu_1(2),'r-','LineWidth',2);
hold on;
plot(x_points_1,-y_points_1+ mu_1(2),'r-','LineWidth',2);
plot(x_points_2,y_points_2+ mu_2(2),'g-','LineWidth',2);
plot(x_points_2,-y_points_2+ mu_2(2),'g-','LineWidth',2);
axis equal;
xlabel('x_1'); ylabel('x_2');



g = inline('X'' *( -1/2 * iE) * X + (iE * m)'' * X - 1/2 * m''
   * iE * m - 1/2 * log(det(E)) + log(prior) ','X','m','E','iE','prior');
%Standard priors (b and c)
prior_1 = 1/4;
prior_2 = 1 - prior_1;
%Priors with missclasification costs (d)
% prior_1 = 20;
% prior_2 = 1;

%Obtain g_1 and g_2 for a grid of values
x_axis = get(gca,'xLim'); y_axis = get(gca,'yLim');
x_vect = linspace(x_axis(1),x_axis(2),60);
y_vect = linspace(y_axis(1),y_axis(2),60);
Z_1 = zeros(length(x_vect),length(y_vect));
Z_2 = Z_1;
iSigma_1 = inv(Sigma_1);
iSigma_2 = inv(Sigma_2);
for i = 1:length(x_vect)
    for j = 1:length(y_vect)
        Z_1(i,j) = g([x_vect(i) y_vect(j)]',mu_1,Sigma_1,iSigma_1,prior_1);
        Z_2(i,j) = g([x_vect(i) y_vect(j)]',mu_2,Sigma_2,iSigma_2,prior_2);
    end
end
```

```
%Visualize information
[X Y] = meshgrid(x_vect, y_vect);

figure;
imagesc(X(:),Y(:),abs(Z_2-Z_1)')
set(gca,'YDir','normal')
hold on;

[C,h] = contour(X,Y,(Z_2-Z_1)',[0 0]);
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2,'Edgecolor',[1 1 1]);
colormap cool

plot(x_points_1,y_points_1+ mu_1(2),'r-','LineWidth',2);
plot(x_points_1,-y_points_1+ mu_1(2),'r-','LineWidth',2);
plot(x_points_2,y_points_2+ mu_2(2),'g-','LineWidth',2);
plot(x_points_2,-y_points_2+ mu_2(2),'g-','LineWidth',2);
axis equal;
xlabel('x_1'); ylabel('x_2');
```

Problem 3: Principal Component Analysis [10 points]

Consider the covariance matrix for a Gaussian with mean = (0,0) and variance = $\sigma^2 \times I_2$ where σ^2 is a positive constant, and I_2 is a 2 × 2 identity matrix.

a. What are the two principle components for this matrix? What are their eigenvalues?

   Solution: In order to compute the principal components, we have to find the eigenvalues (λ) and eigenvectors ($\mathbf{v}$) of the covariance matrix ($\Sigma = \sigma^2 \times I_2$) after mean ($\mu$) centering the data for each attribute. Since $\mu = 0$, we can find the eigenvalues by solving:

$$\det(\Sigma - \lambda I) = \mathbf{0}$$

$$\det \begin{bmatrix} \sigma^2 - \lambda & 0 \\ 0 & \sigma^2 - \lambda \end{bmatrix} = (\sigma^2 - \lambda)^2 = 0$$

   Therefore, $\lambda_1 = \lambda_2 = \sigma^2$ and $\mathbf{v}$ will be any vector that satisfies:

$$\begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \times \mathbf{v} = \sigma^2 \times \mathbf{x}$$

   To ensure orthonormality of the components, we select the following vectors $\mathbf{v}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{v}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$.

b. Given a data point (x,y) from this distribution, what is the reconstructed data using the projection onto the first principal component of this matrix?

Solution: Given the properties of our covariance matrix, both components (i.e. $\mathbf{v}_1$ and $\mathbf{v}_2$) are equally important. If we project our data point $\mathbf{p}$ into $\mathbf{v}_1$ we obtain:

$$\mathbf{v}_1^{\mathrm{T}} \cdot (\mathbf{p} - \mu) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x$$

If we reconstruct our data point, we obtain:

$$\hat{\mathbf{p}} = \mathbf{v}_1 \times x + \mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \times x = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

c. For this reconstructed value, what is the expected value of the reconstruction error (squared error between the true value and reconstructed value).

Solution: The squared error between the true value (p) and the reconstructed value ($\hat{\mathbf{p}}$) is:

$$SE = \sum_{i=1}^{2} (\hat{\mathbf{p}}_i - \mathbf{p}_i)^2 \ = \mathbf{y}^2$$

Give the properties of our covariance matrix, the expected value of SE is:

$$\mathrm{E}(SE) = \mathrm{E}(\mathbf{y}^2) = \mathrm{Var}(\mathbf{y}) + \mathrm{E}(\mathbf{y})^2 = \sigma^2 + 0 = \sigma^2$$

Problem 4: EigenFaces [30 points]

In this exercise we provide a dataset of face images (450x400 pixels) to explore the concept of eigenfaces and some of its applications. Please include MATLAB code and images to support your answers.

a. Find the eigenfaces of the dataset. Show the first three components. The images can be loaded by using the helper function "[X] = load_imgs('training');" in MATLAB. (Hint: svd(X,0))

Solution: Figure 4 shows the first three principal components. As we can see, they represent the variance of the images (e.g. beard, glasses).

b. Reduce the dimensionality of the images by projecting them onto a lower dimensional space. How many basis are you using? Justify your answer.

Solution: In order to select the basis we can look at the energy associated to their eigenvalues. Figure 5 shows the normalized eigenvalues and a standard threshold of 0.95. Note that with these basis we reduce from 180000 (450 x 400) to just 24 dimensions.

c. We received a new image but we lost some of its information. How do you suggest to automatically fix it? Show your solution as well as the result. ("[X] = load_imgs('corrupted');")

Solution: As we saw in class, we can reduce the dimensionality of the data and the restore it to the original dimensionality. Figure 6 shows the image before and after reconstruction.

Figure 4: First three principal components

d. We received another image but this time we do not know the name of the person. Use the eigenfaces to perform face recognition. For the sake of simplicity, you can report the closest image in the training set instead of the name of the person. (“[X] = load_imgs(’testing’);”’)

   Solution: To recognize the face we can use find the closest match in the training data. To do so, we project the image onto a lower dimensional space and compare their coefficients. Figure 7 shows the closest match using Euclidean distance.

e. Suggest a different way to use eigenfaces.

   Solution: In class we saw that they are useful for outlier detection and data compression.

   The MATLAB code for all of the previous questions is as follows:

```
function run_experiment()

%Initialize variables
global w; global h;
w = 400; h = 450;
[X] = load_imgs('training');

%Reduce dimensionality while preserving .95 of the energy
[W mX mu] = getPCA(X,0.95,1);

%Load and recover corrupted image
[I] = load_imgs('corrupted');
recover(I,mu,W,1);

%Load and make prediction image
[I] = load_imgs('testing');
```

Figure 5: Normalized eigenvalues and 0.95 energy threshold

```
predict(I,mu,W,mX,1);



function [P] = predict(Itest,mu,W,mX, show_info)

%Subtract the mean and project
Itest = Itest(:) - mu;
Y = W’ * mX;
ynew = W’ * Itest;

%Compute Euclidean distance
Dist =  sqrt(sum((Y - repmat(ynew,1,size(Y,2))).^2));
[v ord] = sort(Dist);

%Closest Match
P = mX(:,ord(1)) + mu;

if show_info
    global w; global h;
    figure;
    subplot(121); imagesc(reshape(Itest+mu,[h w]));
    colormap gray; set(gca,’XTick’,[],’YTick’,[]);
    axis equal; title(’Testing Image’);

    subplot(122);imagesc(reshape(P,[h w]));
    colormap gray; set(gca,’XTick’,[],’YTick’,[]);
```

Figure 6: (Left) Original image, (right) reconstructed image

```
    title('Closest Image'); axis equal;
end



function [rI] = recover(I,mu,W,show_info)
%In order to fill the missing information, we project I onto a lower dimensional
%space and then get it back
ynew = W' * (I - mu);
rI = W * ynew;
%We add the mean back
rI = rI + mu;

if show_info
    global w; global h;
    figure;
    subplot(121); imagesc(reshape(I,[h w]));
    colormap gray; set(gca,'XTick',[],'YTick',[]);
    axis equal; title('Before');

    subplot(122);imagesc(reshape(rI,[h w]));
    colormap gray; set(gca,'XTick',[],'YTick',[]);
    title('After'); axis equal;
end
```

Figure 7: (Left) Testing image, (right) Closest match

```
%PCA
function [W mX mu] = getPCA(X, energy, show_info)
n = size(X,2);

%-Subtract the mean
mu = mean(X,2);
mX = X - repmat(mu,1,n);
%-There is no need to make unit variance
%because all of the features have the same scale
%-Compute projection matrix
%(eigenvalues are already sorted)
[U,S,V] = svd(mX,0);

%As we saw in class, we can the eigenvectors based on their energy
%(i.e. normalized value of its eigenvalue)
D = diag(S);
tmp = D./sum(D);
good_idx = find(cumsum(tmp)<=energy);

W = U(:,good_idx);

if show_info
    global w; global h;
    %Show first 3 eigenfaces
    figure;
    for i = 1:3
```

```
        subplot(1,3,i)
        imagesc(reshap
        title(sprintf(
        set(gca,'XTick
    end

    %Show energy for a
    figure;
    plot(tmp);
    hold on;
    plot([length(good_
    xlim([1 n]);
    ylabel('Energy');
    xlabel('Eigenvalue
end
```

Problem 5: Hidde

We have two 2-state Hidden Ma
B:

The output probabilities are gi

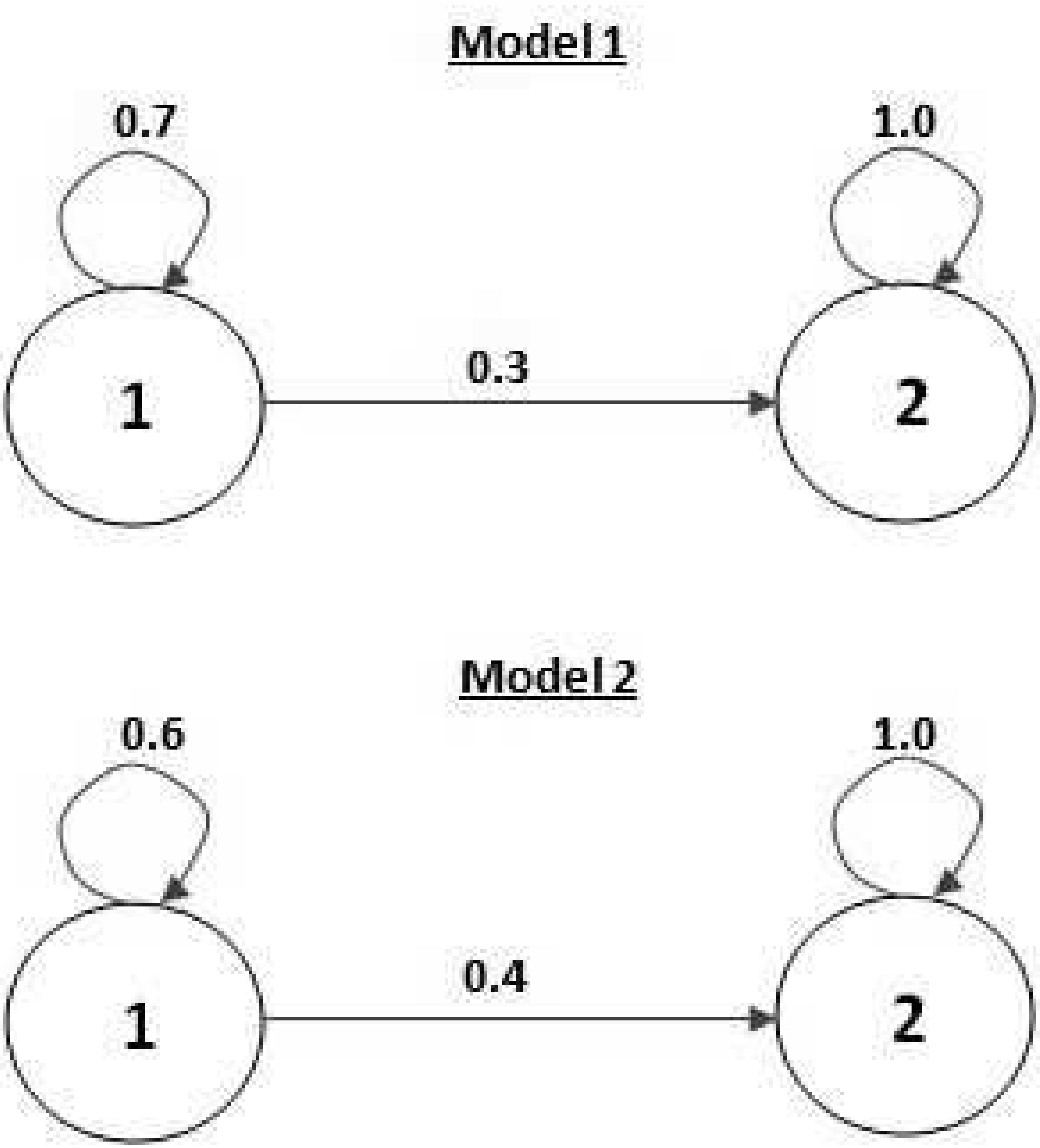


Figure 8: Two HMMs

Model 1: $b_1(A)=0.85$ $b_1(B)=0.15$ $b_2(A)=0.4$ $b_2(B)=0.6$

Model 2: $b_1(A)=0.2$ $b_1(B)=0.8$ $b_2(A)=0.1$ $b_2(B)=0.9$

The initial probabilities are given by:

Model 1: $\pi_1 = 0.75 \quad \pi_2 = 0.25$

Model 2: $\pi_1 = 0.5 \quad \pi_2 = 0.5$

a. For Model 1, what is the probability of an observation sequence { B A B } being generated?

   SOLUTION:

   P({ B A B } | M_1)

State Sequence	Probability Calculation for State Sequence	Total
1 1 1	0.75 * 0.15 * 0.7 * 0.85 * 0.7 * 0.15	0.007
1 1 2	0.75 * 0.15 * 0.7 * 0.85 * 0.3 * 0.6	0.012
1 2 2	0.75 * 0.15 * 0.3 * 0.4 * 1 * 0.6	0.0081
2 2 2	0.25 * 0.6 * 1 * 0.4 * 1 * 0.6	0.036
–	Sum	0.0631

   The probability of the observed sequence given model 1 is: 0.0631. I.e. the sum of the probabilities of all the ways that this sequence could have occurred in model 1.

b. For Model 1, given that this HMM produced an observation sequence { B A B }, what is the most likely sequence of hidden states that led to those observations?

   Solution:
   Look at for max row in the table above:

   2 2 2 is the most probable sequence of hidden states to produce the observed sequence given model 1.

c. Which model is more likely to produce the observation sequence { B A B }?

   Solution:

State Sequence	Probability Calculation for State Sequence	Total
1 1 1	0.5 * 0.8 * 0.6 * 0.2 * 0.6 * 0.8	0.023
1 1 2	0.5 * 0.8 * 0.6 * 0.2 * 0.4 * 0.9	0.0173
1 2 2	0.5 * 0.8 * 0.4 * 0.1 * 1 * 0.9	0.0144
2 2 2	0.5 * 0.9 * 1 * 0.1 * 1 * 0.9	0.0405
–	Sum	0.0952

   Model 2 is more likely to produce the observed sequence as the sum of probabilities of all the possible ways that the sequence could occur is greater for model 2.