Appendix A:
Problem 1

```
clear; close all;

%Got insprisation from 2020SUMMER2 code and
generateDataFromGMM.m

%% Generate  Samples with All Labels
%L = 0/1 ~ 0.65/0.35
%Lc = 00/01 ~ 0.325/0.325

n = 3; % number of feature dimensions
N = 10000; % number of iid samples
mu(:,1) = [3;0]; mu(:,2) = [0;3]; mu(:,3) = [2;2];
Sigma(:,:,1) = [2,0;0,1];%00
Sigma(:,:,2) = [1,0;0,2];%01
Sigma(:,:,3) = [1,0;0,1];%11
p = [0.65,0.35]; % class priors for labels 0 and 1
respectively
w = [0.5, 0.65];
priors = [w(1)*w(2), w(2)*(1-w(1)), 1-w(2)];

u = rand(1,N);
pthresholds = [cumsum(priors),1];%ideal
for i = 1:length(priors)
    indl = find(u <= pthresholds(i)); Nt =
length(indl);
    labels(1,indl) = i*ones(1,Nt);
    u(1,indl) = 1.1*ones(1,Nt);
    r(:,indl) = mvnrnd(mu(:,i),Sigma(:,:,i),Nt)';
end

% figure(1); plot(r(1,labels == 1),r(2,labels
==1),'.b'); axis equal,hold on;
% plot(r(1,labels ==2),r(2,labels ==2),'.g');axis
equal,hold on;
% plot(r(1,labels ==3),r(2,labels ==3),'.r');

%Ture labels
label(:,labels==1)=0;
label(:,labels==2)=0;
label(:,labels==3)=1;
```

```
figure(2);plot(r(1,label == 1),r(2,label == 1),'.r');
axis equal,hold on;
plot(r(1,label ==0),r(2,label ==0),'.b');
Nc = [length(find(label==0)),length(find(label==1))];%
number of samples from each class
x = r; % save up space

%% Part1. ERM 1.1 & 1.2


%Theoretical Minimum Error
lambda = [0 1;1 0]; % loss values
gamma = (lambda(2,1)-lambda(1,1))/(lambda(1,2)-
lambda(2,2)) * p(1)/p(2); %threshold
discriminantScore = -
(log(w(1)*evalGaussian(r,mu(:,1),Sigma(:,:,1))...
    + (1-w(1))*evalGaussian(r,mu(:,2),Sigma(:,:,2)))...
    - log(evalGaussian(r,mu(:,3),Sigma(:,:,3))));%
log(gamma);
decision_ideal = (discriminantScore >= log(gamma));

ind00 = find(decision_ideal==0 & label==0); p00 =
length(ind00)/Nc(1); % probability of true negative
ind10 = find(decision_ideal==1 & label==0); p10 =
length(ind10)/Nc(1); % probability of false positive
ind01 = find(decision_ideal==0 & label==1); p01 =
length(ind01)/Nc(2); % probability of false negative
ind11 = find(decision_ideal==1 & label==1); p11 =
length(ind11)/Nc(2); % probability of true positive

% if norm(lambda-[0,1;1,0])<eps % Using 0-1 loss
indicates intent to minimize P(error)
%     Perror_MAP = [p10,p01]*Nc'/N, % probability of
error, empirically estimated
% end

figure(3), % class 0 circle, class 1 +, correct green,
incorrect red
plot(x(1,ind00),x(2,ind00),'ob'); hold on,
plot(x(1,ind10),x(2,ind10),'or'); hold on,
plot(x(1,ind01),x(2,ind01),'+r'); hold on,
plot(x(1,ind11),x(2,ind11),'+b'); hold on,
axis equal,
```

```
legend('Correct decisions for data from Class 0','Wrong
decisions for data from Class 0',...
    'Wrong decisions for data from Class 1','Correct
decisions for data from Class 1'),
title('Data and their classifier decisions versus true
labels'),
xlabel('x_1'), ylabel('x_2'),

%Estimate Minimum Error

sortDS=sort(discriminantScore);
%Generate vector of gammas for parametric sweep
logGamma=[min(discriminantScore)-eps
sort(discriminantScore)+eps];
for ind=1:length(logGamma)
decision=discriminantScore> logGamma(ind);
Num_pos(ind)=sum(decision);
pFP(ind)=sum(decision==1 & label==0)/Nc(1);
pTP(ind)=sum(decision==1 & label==1)/Nc(2);
pFN(ind)=sum(decision==0 & label==1)/Nc(2);
pTN(ind)=sum(decision==0 & label==0)/Nc(1);
pFE(ind)=(sum(decision==0 & label==1) + sum(decision==1
& label==0))/N;
end

%If multiple minimums are found choose the one closest
to the theoretical
%minimum
[min_pFE, min_pFE_ind]=min(pFE);
if length(min_pFE_ind)>1

[~,minDistTheory_ind]=min(abs(logGamma(min_pFE_ind)-
logGamma_ideal));
    min_pFE_ind=min_pFE_ind(minDistTheory_ind);
end
%Find minimum gamma and corresponding false and true
positive rates
minGAMMA=exp(logGamma(min_pFE_ind));
min_FP=pFP(min_pFE_ind);
min_TP=pTP(min_pFE_ind);

figure(4);
plot(pFP,pTP,'DisplayName','ROC Curve','LineWidth',2);
hold all;
```

```
plot(min_FP,min_TP,'o','DisplayName','Estimated Min.
Error','LineWidth',2);
plot(p10,p11,'+','DisplayName',...
'Theoretical Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve');
legend 'show';
grid on; box on;
fprintf('Theoretical for ERM: Gamma=%1.2f,
Error=%1.2f%%\n',...
gamma,(length(ind10)+length(ind01))/100);
fprintf('Estimated for ERM: Gamma=%1.2f,
Error=%1.2f%%\n',...
    minGAMMA,100*min_pFE);

%% Part1.3
%Probability of Error vs. Gamma
figure(5);
plot(logGamma,pFE,'DisplayName','Errors','LineWidth',2)
;
hold on;
plot(logGamma(min_pFE_ind),pFE(min_pFE_ind),...
    'o','DisplayName','Estimated Minimum
Error','LineWidth',2);
plot(log(gamma),(length(ind10)+length(ind01))/N,...
    '+','DisplayName','Theoretical Minimum
Error','LineWidth',2);
xlabel('Gamma');
ylabel('Proportion of Errors');
title('Probability of Error vs. Gamma')
grid on;
legend 'show';

%% Part2. LDA
%Compute Sample Mean and covariances
mu_LDA(:,1)=mean(r(:,labels==1),2);
mu_LDA(:,2)=mean(r(:,labels==2),2);
mu_LDA(:,3)=mean(r(:,labels==3),2);
Sigma_LDA(:,:,1)=cov(r(:,labels==1)')';
Sigma_LDA(:,:,2)=cov(r(:,labels==2)')';
Sigma_LDA(:,:,3)=cov(r(:,labels==3)')';

% %Check mu/sigma
```

```
% mu0 = 0.5*mu(:,1) + 0.5*mu(:,2);
% sigma0 = 0.5*Sigma(:,:,1) + 0.5*Sigma(:,:,2)...
%           + 0.5*(mu(:,1)-mu0)*(mu(:,1)-
mu0)'+0.5*(mu(:,2)-mu0)*(mu(:,2)-mu0)';%Law of total
variance

%Compute scatter matrices
Sb = (mu_LDA(:,1)+mu_LDA(:,2)-
mu_LDA(:,3))*(mu_LDA(:,1)+mu_LDA(:,2)-mu_LDA(:,3))';
Sw = Sigma_LDA(:,:,1) + Sigma_LDA(:,:,2) +
Sigma_LDA(:,:,3);
[V,D] = eig(Sw\Sb);

[~,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(label==1))-
mean(yLDA(label==0)))*wLDA; % ensures class1 falls on
the + side of the axis
yLDA = sign(mean(yLDA(label==1))-
mean(yLDA(label==0)))*yLDA; % flip yLDA accordingly

%Evaluate for different taus
tau=[min(yLDA)-0.1 sort(yLDA)+0.1];
for ind=1:length(tau)
    decision=yLDA>tau(ind);
    %Num_pos_LDA(ind)=sum(decision);
    pFP_LDA(ind)=sum(decision==1 & label==0)/Nc(1);
    pTP_LDA(ind)=sum(decision==1 & label==1)/Nc(2);
    pFN_LDA(ind)=sum(decision==0 & label==1)/Nc(2);
    pTN_LDA(ind)=sum(decision==0 & label==0)/Nc(1);
    pFE_LDA(ind)=(sum(decision==0 & label==1) +
sum(decision==1 & label==0))/N;
end

%Estimated Minimum Error

[min_pFE_LDA, min_pFE_ind_LDA]=min(pFE_LDA);
if length(min_pFE_ind_LDA)>1

[~,minDistTheory_ind]=min(abs(logGamma(min_pFE_ind_LDA)
-logGamma_ideal));
    min_pFE_ind_LDA=min_pFE_ind_LDA(minDistTheory_ind);
```

```matlab
end
minTAU_LDA=tau(min_pFE_ind_LDA);
min_FP_LDA=pFP_LDA(min_pFE_ind_LDA);
min_TP_LDA=pTP_LDA(min_pFE_ind_LDA);
%Plot results
figure;
plot(yLDA(label==0),zeros(1,Nc(1)),'o','DisplayName','L
abel 0');
hold all;
plot(yLDA(label==1),ones(1,Nc(2)),'o','DisplayName','La
bel 1');
ylim([-1 2]);
plot(repmat(tau(min_pFE_ind_LDA),1,2),ylim,'m--',...
'DisplayName','Tau for Min. Error','LineWidth',2);
grid on;
xlabel('yLDA');
title('Fisher LDA Projection of Data');
legend 'show';

figure;
plot(pFP_LDA,pTP_LDA,'DisplayName','ROC
Curve','LineWidth',2);
hold all;
plot(min_FP_LDA,min_TP_LDA,'o','DisplayName',...
'Estimated Min. Error','LineWidth',2);
xlabel('Prob. False Positive');
ylabel('Prob. True Positive');
title('Mininimum Expected Risk ROC Curve');
legend 'show';
grid on; box on;

figure;
plot(tau,pFE_LDA,'DisplayName','Errors','LineWidth',2);
hold on;
plot(tau(min_pFE_ind_LDA),pFE_LDA(min_pFE_ind_LDA),'ro'
,...
'DisplayName','Minimum Error','LineWidth',2);
xlabel('Tau');
ylabel('Proportion of Errors');
title('Probability of Error vs. Tau for Fisher LDA')
grid on;
legend 'show';
fprintf('Estimated for LDA: Tau=%1.2f,
Error=%1.2f%%\n',...
```

```
minTAU_LDA,100*min_pFE_LDA);


%% Functions
%From generateDataFromGMM.m
function g = evalGaussian(x ,mu,Sigma)
%Evaluates the Gaussian pdf N(mu, Sigma ) at each
column of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2); %coefficient
E = -0.5*sum((x-repmat(mu,1,N)).*(Sigma\(x-
repmat(mu,1,N))),1);%exponent
g = C*exp(E); %finalgaussianevaluation
end
```

Appendix B:
Problem 2

```
clear; close all;

%Got inspriation from 2020SUMMER2,
generateDataFromGMM.m and
%ERMwithClabels.m

%% Generate  Samples with All Labels
%L = 1/2/3 ~ 0.3/0.3/0.4
%Lc = 30/31 ~ 0.2/0.2

p = [.3 .3 .2 .2]; % class priors for labels 0 and 1
respectively
n = length(p); % number of feature dimensions
N = 10000; % number of iid samples
mu(:,1) = [0;0;1]; mu(:,2) = [2;2;0]; mu(:,3) =
[0;3;3]; mu(:,4) = [1;2;2];
Sigma(:,:,1) = [2,0,0;0,1,0;0,0,1];%1
Sigma(:,:,2) = [1,0,0;0,2,0;0,0,2];%2
Sigma(:,:,3) = [1,0,0;0,1,0;0,0,1];%30
Sigma(:,:,4) = [9,0,0;0,1,0;0,0,1];%31

u = rand(1,N);
pthresholds = [cumsum(p),1];%ideal
for i = 1:length(p)
    indl = find(u <= pthresholds(i)); Nt =
length(indl);
    labels(1,indl) = i*ones(1,Nt);
    u(1,indl) = 1.1*ones(1,Nt);
    r(:,indl) = mvnrnd(mu(:,i),Sigma(:,:,i),Nt)';
end

figure(1); plot3(r(1,labels == 1),r(2,labels
==1),r(3,labels ==1),'.r'); axis equal,hold on;
plot3(r(1,labels ==2),r(2,labels ==2),r(3,labels
==2),'.g');axis equal,hold on;
plot3(r(1,labels ==3),r(2,labels ==3),r(3,labels
==3),'.b');axis equal,hold on;
plot3(r(1,labels ==4),r(2,labels ==4),r(3,labels
==4),'.k');
title('All Disturbution Map')
```

```
%Ture labels
n= n-1;
priors = [.3 .3 .4];
label(:,labels==1)=1;
label(:,labels==2)=2;
label(:,labels==3)=3;
label(:,labels==4)=3;
figure(2); plot3(r(1,label == 1),r(2,label
==1),r(3,label ==1),'.r'); axis equal,hold on;
plot3(r(1,label ==2),r(2,label ==2),r(3,label
==2),'.g');axis equal,hold on;
plot3(r(1,label ==3),r(2,label ==3),r(3,label
==3),'.b')
title('Label Map')
legend("Label 1","Label 2","Label 3")
Nc =
[length(find(label==1)),length(find(label==2)),length(f
ind(label==3))];% number of samples from each class
x = r; % save up space

%%
symbols='oxs';
lambda(:,:,1) = ones(n,n)-eye(n,n);
lambda(:,:,2) = [0 1 10; 1 0 10; 1 1 0];
lambda(:,:,3) = [0 1 100; 1 0 100; 1 1 0];

for ind = 1:n+1
    Nc(ind,1) = length(find(labels==ind));
end

for ind = 1:n+1
    pxgivenls(ind,:) =...
    evalGaussian(x,mu(:,ind),Sigma(:,:,ind)); %
Evaluate p(x|L=l)
end
%Ture pxgivenl
pxgivenl(1:2,:)=pxgivenls(1:2,:);
pxgivenl(3,:) = 0.5*pxgivenls(3,:)+0.5*pxgivenls(4,:);

px = priors*pxgivenl; % Total probability theorem
classPosteriors =
pxgivenl.*repmat(priors',1,N)./repmat(px,n,1); %
P(L=l|x)
```

```
for j = 1:3
    expectedRisks = lambda(:,:,j)*classPosteriors; %
Expected Risk for each label (rows) for each sample
(columns)
    [~,decisions] = min(expectedRisks,[],1); % Minimum
expected risk decision with 0-1 loss is the same as MAP

    figure(j+2);
    for i = 1:n
        plotLabelIndex = label == i;
        plotDecisionIndex = decisions == i;
        plot3(x(1,label == i&decisions == i),x(2,label
== i&decisions == i),x(3,label == i&decisions == i),...

'Marker',symbols(i),'MarkerEdgeColor','#3CB371',...

'MarkerFaceColor','none','LineStyle','none');axis
equal,hold on;
        plot3(x(1,label == i&decisions ~= i),x(2,label
== i&decisions ~= i),x(3,label == i&decisions ~= i),...

'Marker',symbols(i),'MarkerEdgeColor','#FF6347',...

'MarkerFaceColor','none','LineStyle','none');axis
equal,hold on;
        %pFEIndex(i) = sum(label == i & decisions ~=
i);
    end
    legend('Correct decisions for data from Class
1','Wrong decisions for data from Class 1',...
        'Correct decisions for data from Class
2','Wrong decisions for data from Class 2',...
        'Correct decisions for data from Class
3','Wrong decisions for data from Class 3', ...
        'Location','northeast');
    xlabel('x');
    ylabel('y');
    zlabel('z');
    title('Data and their classifier decisions versus
true labels')

    for d = 1:n % each decision option
        for l = 1:n % each class label
            ind_dl = find(decisions==d & labels==l);
```

```
            ConfusionMatrix(d,l) =
length(ind_dl)/length(find(labels==l));
        end
    end

   ExpRisk =
priors*mean(expectedRisks'*ConfusionMatrix,1)';
   fprintf('Expected Risk for Loss Matrix %1.0f=%1.2f
\n',...
        j,ExpRisk);
end

%% Functions
%From generateDataFromGMM.m
function g = evalGaussian(x ,mu,Sigma)
%Evaluates the Gaussian pdf N(mu, Sigma ) at each
column of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2); %coefficient
E = -0.5*sum((x-repmat(mu,1,N)).*(Sigma\(x-
repmat(mu,1,N))),1);%exponent
g = C*exp(E); %finalgaussianevaluation
end
```