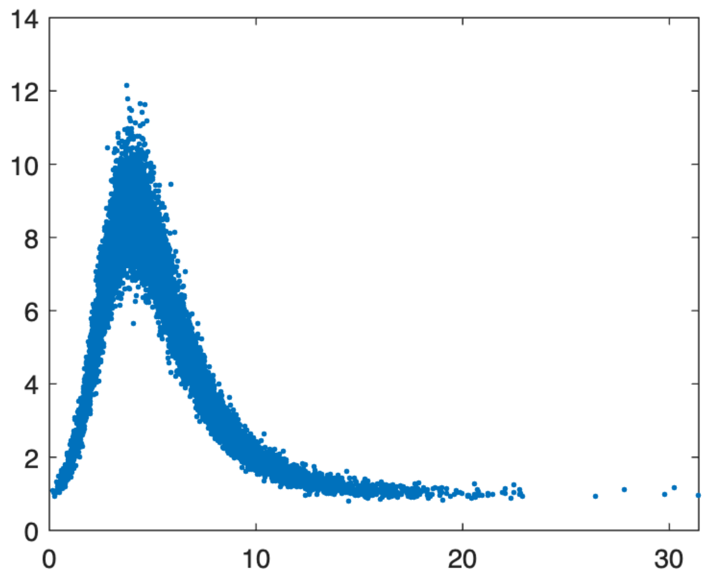
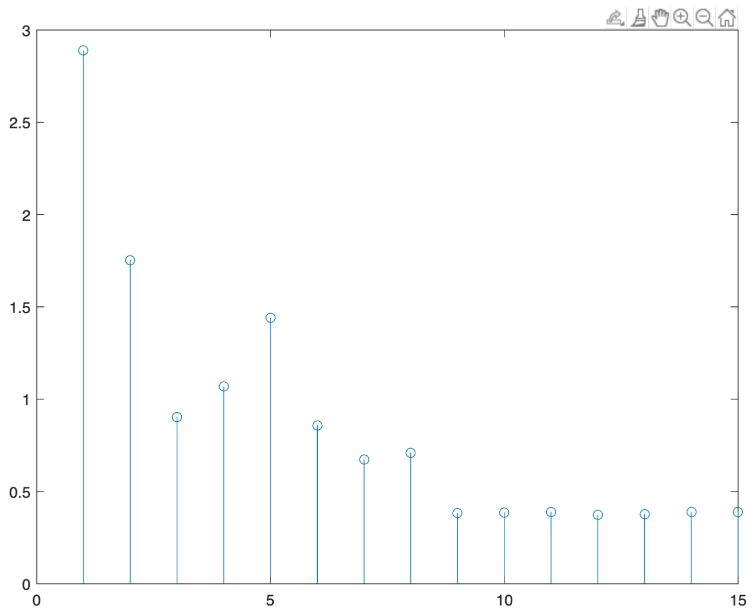
Homework4 by Jiabo Cheng

Question 1:

The generated data is plot below:

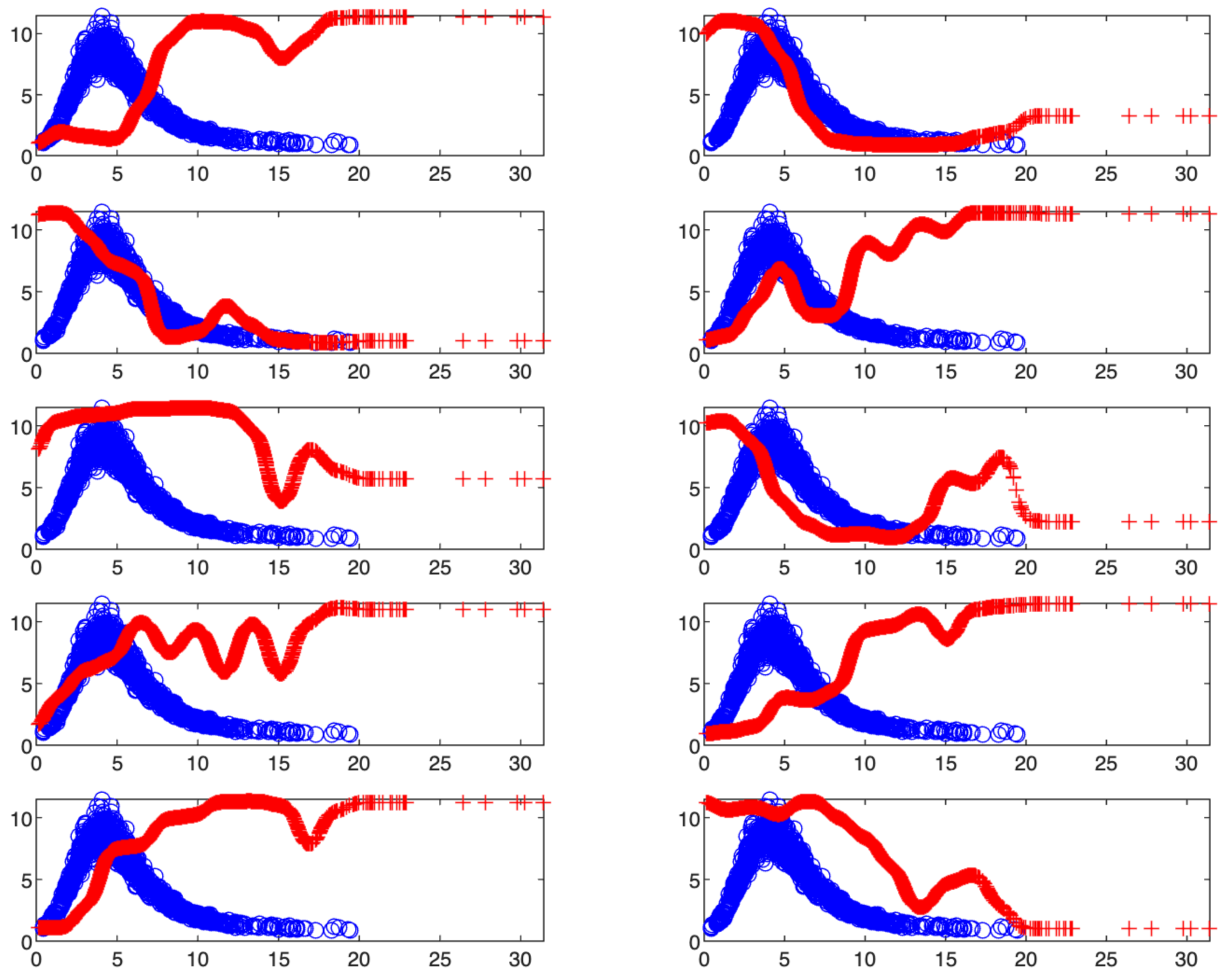


Next, choosing parameters with minimum mean square error using the maximum likelihood parameter estimation method and 10-fold cross validation to select the best number of perceptrons in the first layer.
The plot below shows the relationship between minimum square error vs. number of perceptrons.



Based on parameter estimation, we would select MLP with 12 perceptrons in the first layer to train.

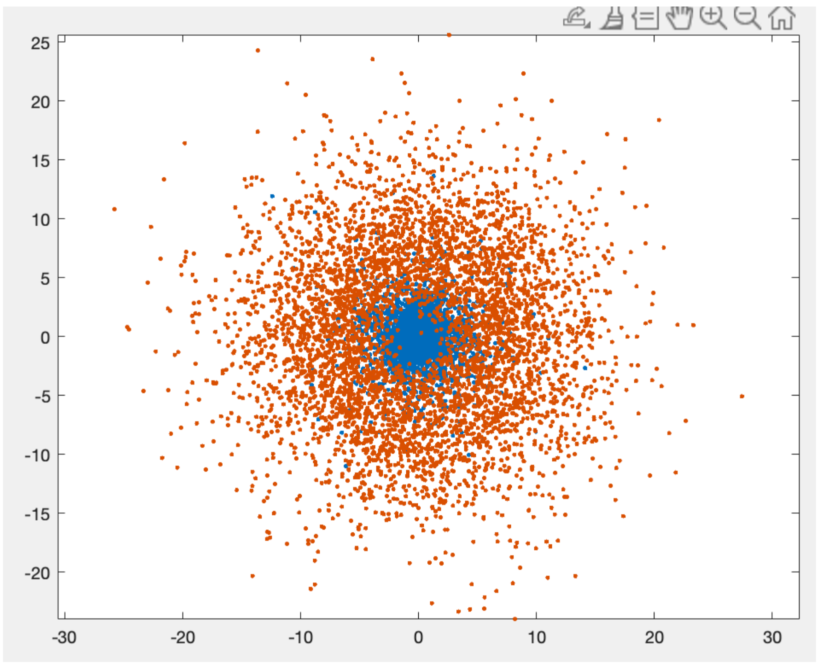
Finally, train the trained MLP to test set 10 times in order for avoiding to stop at local minimum square error. The visualization of trained model shows below:



Obviously, the 3rd time calculated value relatively perfect fit the training set, the **minimum square error is 9.442.**

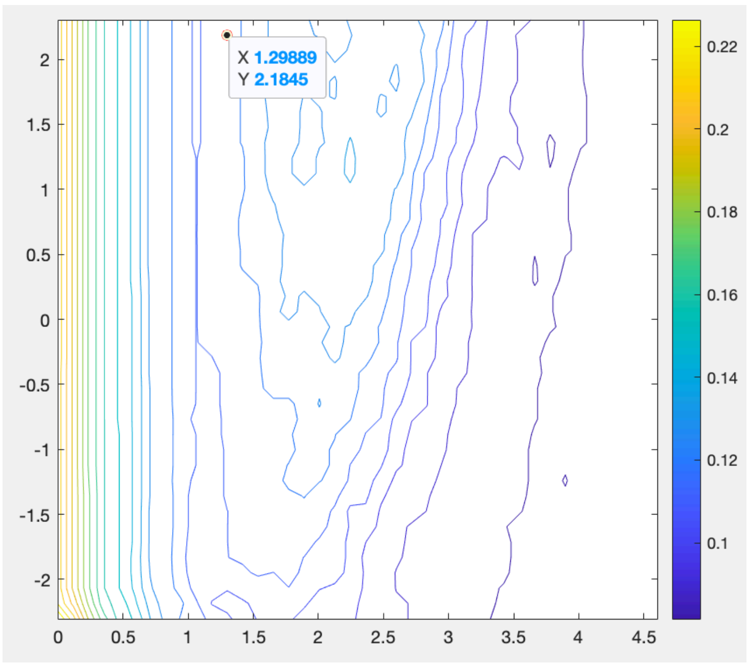
Question 2:

The data generated by *generateMultiringDataset.m* function is shown below:



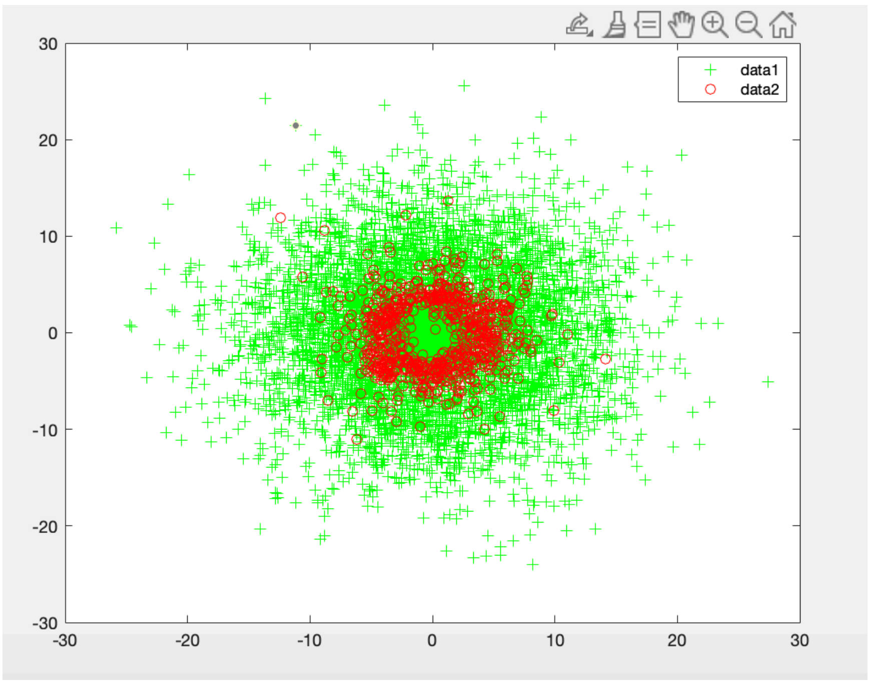
Using 10-fold cross validation to select the best box constraint hyper parameter C and Gaussian kernel with parameter σ.

The plot below shows minimum average probability of error with different C and σ.



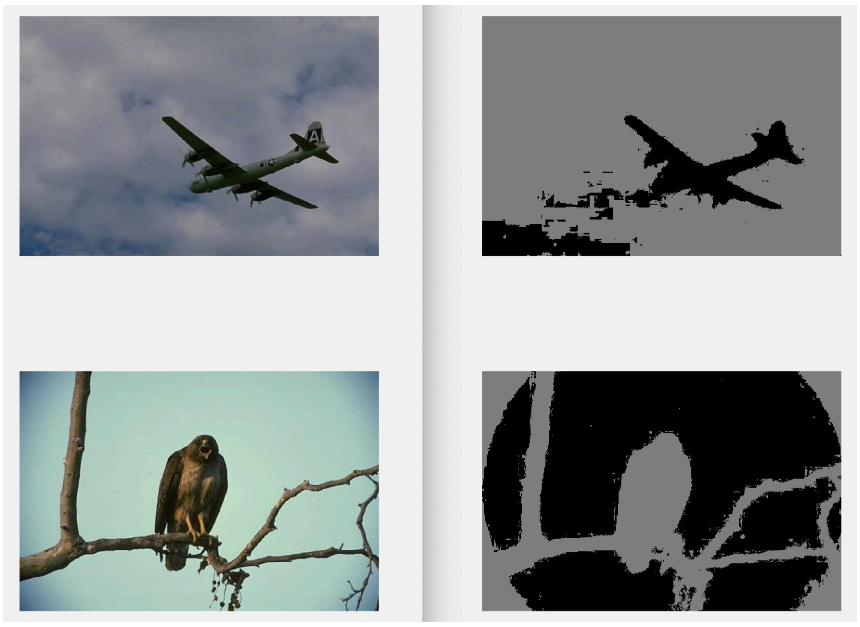
The selected best C =3.6652, best $\sigma = 8.8862$, using these hyper parameter to final train the train set, validate the result using validation dataset. Finally, the minimum probability error is 0.0714.

The visualization is shown below:

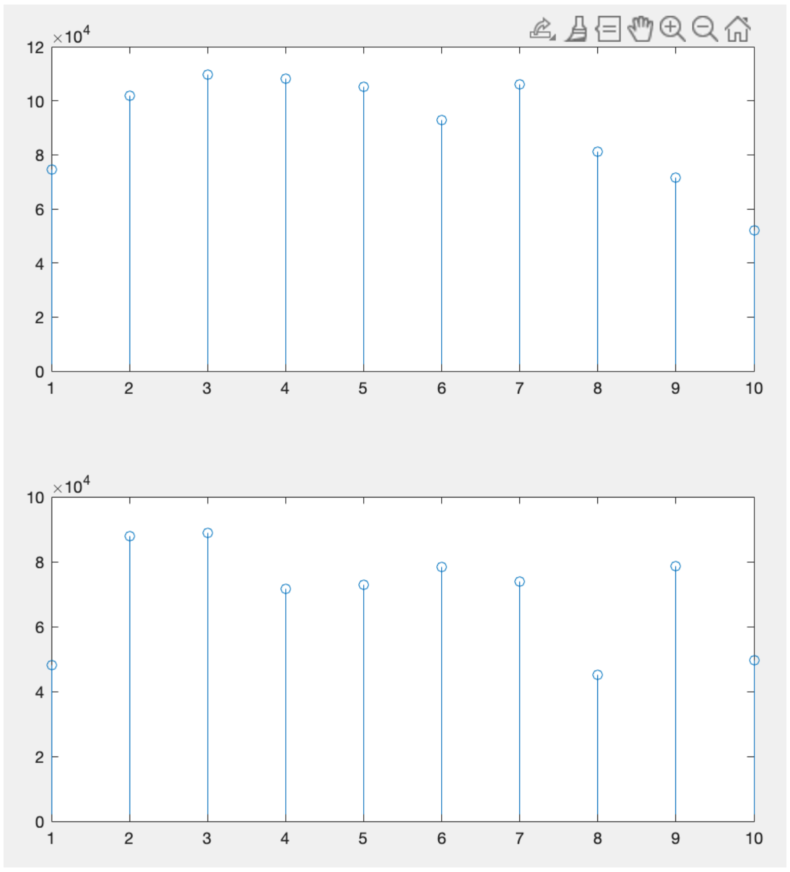


Question 3:

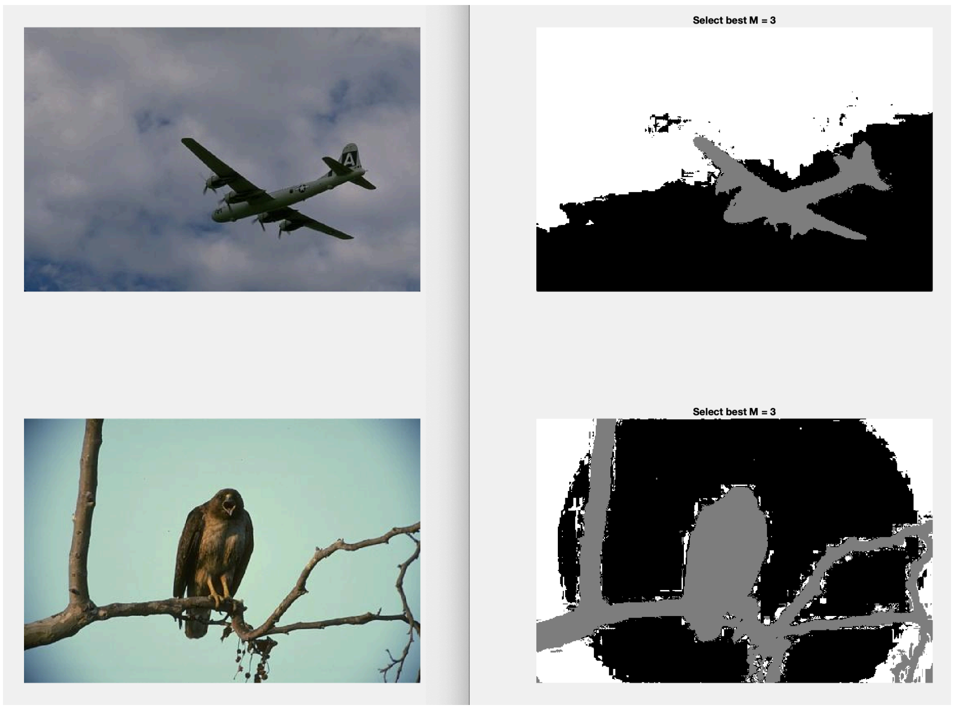
(1) Using maximum likelihood parameter estimation to fit a GMM with 2-component clusters, the result shows below:



(2) Using 10-fold cross validation and maximum average validation log-likelihood as objective, to select best number of Gaussian components



With the increase of number of components, the log likelihood increase at first, and then decrease.



The best selected number of components are both 3 for two photos.

Appendix
Question 1:

```
clear all; close all;


N_train = 1000;
N_valid = 10000;


[X_train,y_train] = exam4q1_generateData(N_train);
[X_valid,y_valid] = exam4q1_generateData(N_valid);


figure(1)
plot(X_valid,y_valid,'.')


numP = 15;
K = 10;
dummy = ceil(linspace(0,N_train,K+1));
for m = 1:numP

    for k = 1:K
        K_x_valid = X_train(:,[dummy(k)+1:dummy(k+1)]);
        K_y_valid = y_train([dummy(k)+1:dummy(k+1)]);
        train_ind = setdiff([1:N_train],[dummy(k)+1:dummy(k+1)]);
        K_x_train = X_train(:,train_ind);
        K_y_train = y_train(train_ind);

        net = feedforwardnet(m);
        net = train(net,K_x_train,K_y_train);
        y_cal = net(K_x_valid);

        MSE(k) = mean((y_cal - K_y_valid).^2);


    end
    figure(2)
    subplot(numP,3,m)
    stem(MSE)
    meanMSE(m) = mean(MSE);

end
figure(3)
stem(meanMSE)
[~,bestM] = min(meanMSE);


%final_training


for n = 1:10 % number of training
    net_final = patternnet(bestM);
    net_final = train(net_final,X_train,y_train);
    y_cal_final = net_final(X_valid);
    MSE_final(n) = mean((y_cal_final-y_valid).^2);


    figure(4)
    subplot(5,2,n)
plot(X_train,y_train,'bo',X_valid,y_cal_final,'r+')


end
[minMSE,idx] = min(MSE_final)






function [x,y] = exam4q1_generateData(N)
```

Question 2:

```
%Question2
clear all; close all;


N_train = 1000;
N_valid = 10000;
Num_L = 2;




[X_train, Label_train] = generateMultiringDataset(Num_L,N_train);
[X_valid, Label_valid] = generateMultiringDataset(Num_L,N_valid);


Label_train(Label_train==1) = -1;
Label_train(Label_train==2) = 1;
Label_valid(Label_valid==1) = -1;
Label_valid(Label_valid==2) = 1;


sigmalist = logspace(-1,1,40);
clist = logspace(0,2,40);
meanPE = zeros(length(clist),length(sigmalist));


K = 10;
dummy = ceil(linspace(0,N_train,K+1));
%for loop
for i = 1:length(clist)
for j=1:length(sigmalist)
[i,j]
PE = zeros(1,K);
parfor k = 1:K

K_x_valid = X_train(:,[dummy(k)+1:dummy(k+1)]);
K_y_valid = Label_train([dummy(k)+1:dummy(k+1)]);
```

```
train_ind = setdiff([1:N_train],[dummy(k)+1:dummy(k+1)]);
K_x_train = X_train(:,train_ind);
K_y_train = Label_train(train_ind);
%train svm
SVMK =
fitcsvm(K_x_train',K_y_train,'BoxConstraint',clist(i),'KernelFunction','gaussian','Kerne
lScale',sigmalist(j));
decisions = SVMK.predict(K_x_valid')';
PE(k) = length(find(K_y_valid~=decisions))/ length(K_x_valid);


        end
    meanPE(i,j) = mean(PE)

    end
end


figure(2)
contour(log(clist),log(sigmalist),meanPE,20)
axis equal;
hold all;
plot(log(bestc),log(bestsigma),'ro')


[~,minind] = min(meanPE(:))
[indc,indsigma]=ind2sub(size(meanPE),minind);
bestc = clist(indc);
bestsigma = sigmalist(indsigma);


%final training
SVM_final = fitcsvm(X_train',
Label_train','BoxConstraint',bestc,'KernelFunction','gaussian','KernelScale',bestsigma);
decisions_final = SVM_final.predict(X_valid')';
PEmin = length(find(Label_valid~=decisions_final))/ length(X_valid)


%plot decisions
figure(3)
plot(X_valid(1,find(Label_valid==decisions_final)),X_valid(2,find(Label_valid==decisions
_final)),'g+');
hold all;
plot(X_valid(1,find(Label_valid~=decisions_final)),X_valid(2,find(Label_valid~=decisions
_final)),'ro');




function [data,labels] = generateMultiringDataset(numberOfClasses,numberOfSamples)


C = numberOfClasses;
N = numberOfSamples;
% Generates N samples from C ring-shaped
% class-conditional pdfs with equal priors


% Randomly determine class labels for each sample
thr = linspace(0,1,C+1); % split [0,1] into C equal length intervals
u = rand(1,N); % generate N samples uniformly random in [0,1]
labels = zeros(1,N);
for l = 1:C
    ind_l = find(thr(l)<u & u<=thr(l+1));
    labels(ind_l) = repmat(l,1,length(ind_l));
end


a = [1:C].^2.5; b = repmat(1.7,1,C); % parameters of the Gamma pdf needed later
% Generate data from appropriate rings
% radius is drawn from Gamma(a,b), angle is uniform in [0,2pi]
angle = 2*pi*rand(1,N);
```

```
radius = zeros(1,N); % reserve space
for l = 1:C
    ind_l = find(labels==l);
    radius(ind_l) = gamrnd(a(l),b(l),1,length(ind_l));
end


data = [radius.*cos(angle);radius.*sin(angle)];


if 1
    colors = rand(C,3);
    figure(1), clf,
    for l = 1:C
        ind_l = find(labels==l);
        plot(data(1,ind_l),data(2,ind_l),'.','MarkerFaceColor',colors(l,:)); axis equal,
hold on,
    end
end
end
```

Question 3:

```
clear all; close all;


filenames ={'3096_color.jpg','42049_color.jpg'};
for i = 1:2
data = imread(filenames{i});
figure(1)
subplot(2,1,i);
imshow(data);
[R,C,D] = size(data);
N=R*C;
data = double(data);;
rowind = [1:R]'*ones(1,C);
colind = ones(R,1)*[1:C];
features = [rowind(:)';colind(:)'];

for d = 1:D
data_d = data(:,:,d);
features = [features; data_d(:)'];
end

    minf = min(features,[],2);
    maxf = max(features,[],2);
    ranges = maxf-minf;
    %norm
    x=(features-minf)./ranges;

    GMM2 = fitgmdist(x',2,'Replicates',10)
    post2 = posterior(GMM2,x')';
    decisions = post2(2,:)>post2(1,:);

    %plot gmm2
    label2 = reshape(decisions,R,C);
    figure(2)
    subplot(2,1,i);
    imshow(uint8(label2*255/2));

    %start k-fold
    M=10; %numebr of gmm
    K=10;
    N = length(x);
    dummy = ceil(linspace(0,N,K+1));
    for m = 1:M
        m
        parfor k = 1:K


            K_x_valid = x(:,[dummy(k)+1:dummy(k+1)]);
            train_ind = setdiff([1:N],[dummy(k)+1:dummy(k+1)]);
            K_x_train = x(:,train_ind);
            %fit
            GMMk = fitgmdist(K_x_train',m,'Replicates',5);
            if GMMk.Converged
                prob(k) = sum(log(pdf(GMMk,K_x_valid')));
            else
                prob(k) = 0

            end
        end

    meanProb(i,m)= mean(prob)

    end

    %plot prob vs m
    figure(3)
    subplot(2,1,i)
    stem(meanProb(i,:))

    [~,bestM] = max(meanProb(i,:));
    resultM(i) = bestM;
    GMM_final = fitgmdist(x',bestM,'Replicates',10);
    postk = posterior(GMM_final,x')';

    lossMatrix = ones(bestM,bestM)-eye(bestM);

    expectedRisk = lossMatrix * postk;

   [~,decisions] = min(expectedRisk,[],1);

   %Plot
   labelk = reshape(decisions-1,R,C);
   figure(4)
   subplot(2,1,i)
   imshow(uint8(labelk*255/2));
   title(strcat({'Select best M = '},num2str(bestM)));

end
```