EECE 5644

Assignment 4

Question 1

For the first part of Question 1, SVM was trained and tested. Gaussian kernel was used by the SVM. A training dataset with 1000 samples and a testing dataset with 10000 samples were generated.

Then, a 10-fold cross validation was performed using the training data to find the optimal hyperparameters, which are the box constraints and the Gaussian kernel width. The range of box constraint was set to be [0.1 2] and the range of kernel width was set to [0.1 4]. Figure 1 below shows the performance for the k-fold hyperparameter validation.

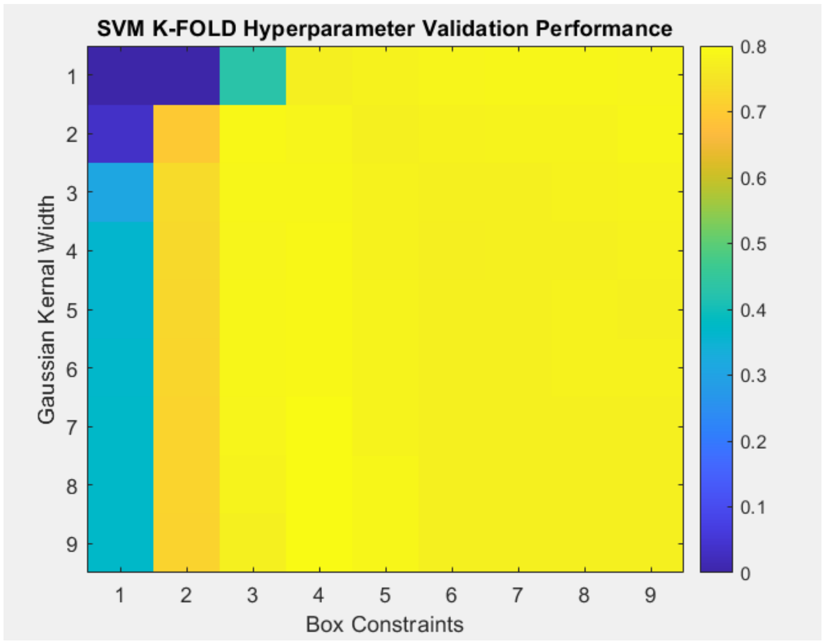

Figure 1: SVM k-fold Hyperparameter Validation Performance

From the cross-validation process, the optimal values of Gaussian kernel width and the box constraint were determined to be 2 and 0.1. And these parameters achieved an average accuracy of 0.896.

The optimal SVM model was then trained on the testing dataset with 10000 samples. The Figure 2 below shows the SVM classification performance.

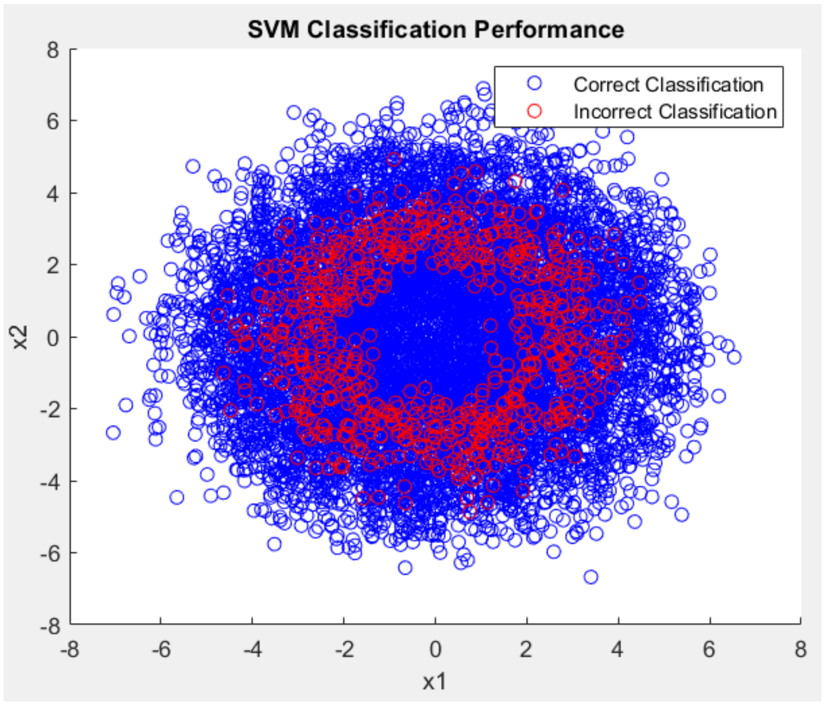

Figure 2: SVM Classification Performance

It can be seen on the figure, the classification boundary was roughly a circle between the two disks, which is as expected. The accuracy was determined to be 0.873.

For the second part of question 1, a MLP classifier was trained and tested. The same training and testing data from the first part of the question was used. The MLP uses a single-hidden layer model. During the cross-validation process, a max number of 15 perceptrons were tested to determine the best number of perceptrons. Figure 3 below shows the number of perceptrons and the corresponding accuracy.

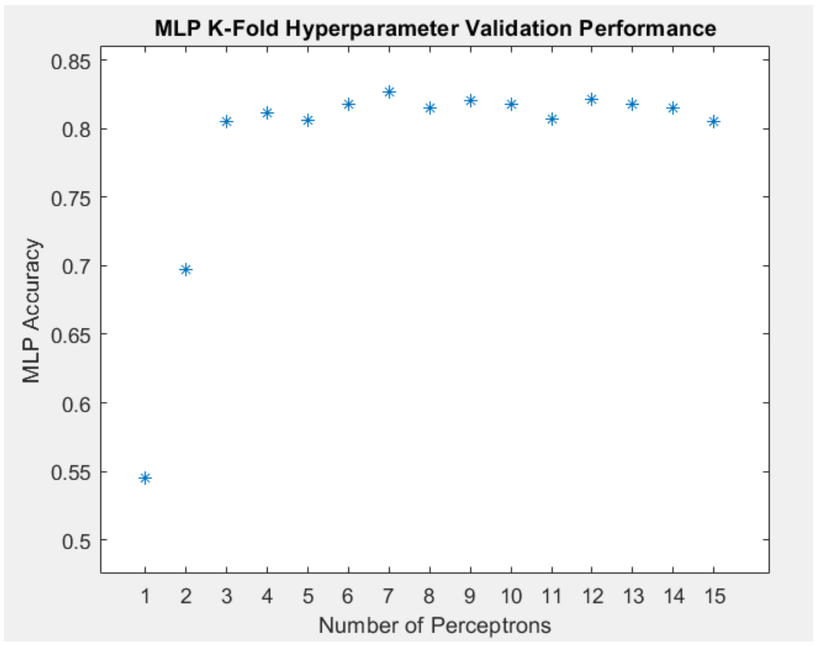

Figure 3: Number of Perceptrons and Accuracy

From the graph, it can be seen that the accuracy reached maximum when the number of perceptron is 7 and the accuracy is 0.822. Then, the optimal MLP model was performed on the entire testing dataset. Figure 4 below shows the classification performance.

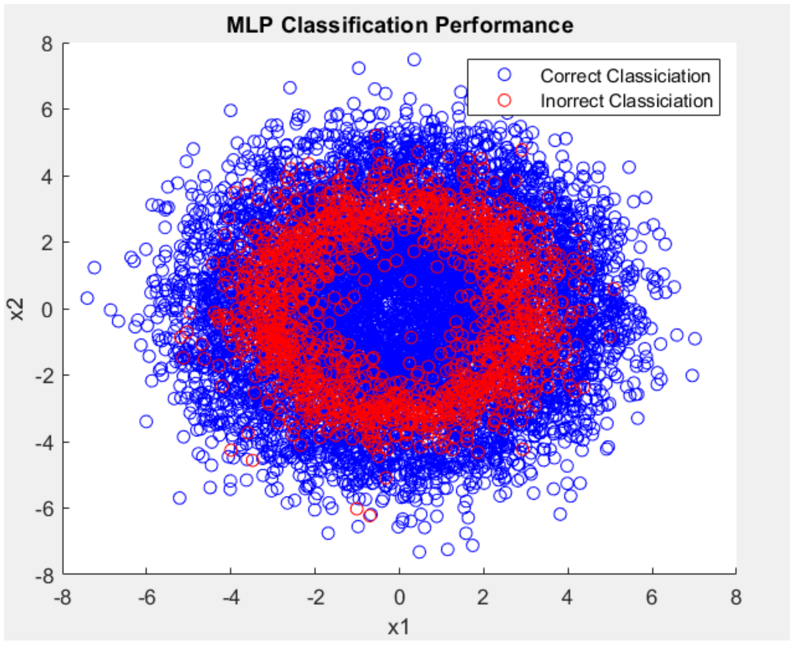

Figure 4: MLP Classification Performance

Similar to the SVM Classification Performance, the classification boundary is also roughly a circle. The accuracy achieved was 0.84, which is little less than the SVM classifier.

Question 2

For question 2, a color image was chosen to be segmented by using GMM based clustering. The image was first read into a 5-dimensional dataset and the data are then normalized so that they all have values between 0 and 1. The image that was used for testing is “3096_color.jpg”.

After the image data has been preprocessed, a 10-fold cross validation and maximum likelihood parameter estimation were used to fit a GMM to the processed data. By determining the maximum average validation-log-likelihood, the optimal model was selected and the best number of GMMs was 3. Figure 5 below shows the result of the cross validation. It can be seen that the log likelihood reached maximum when number of GMMs was 3. Then, it started to decrease as number of GMMs increase.

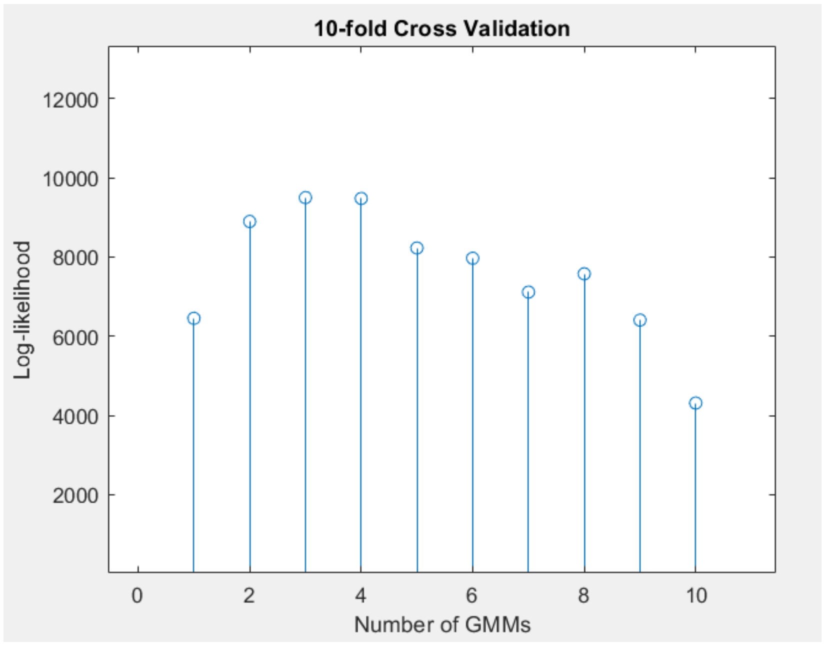

Figure 5: Cross Validation Result

The GMM model obtained was then used for the segmentation with the number of segments equals to the number of components in GMM. The original image is shown in Figure 6 and the GMM segmentation version of the image is shown in Figure 7.

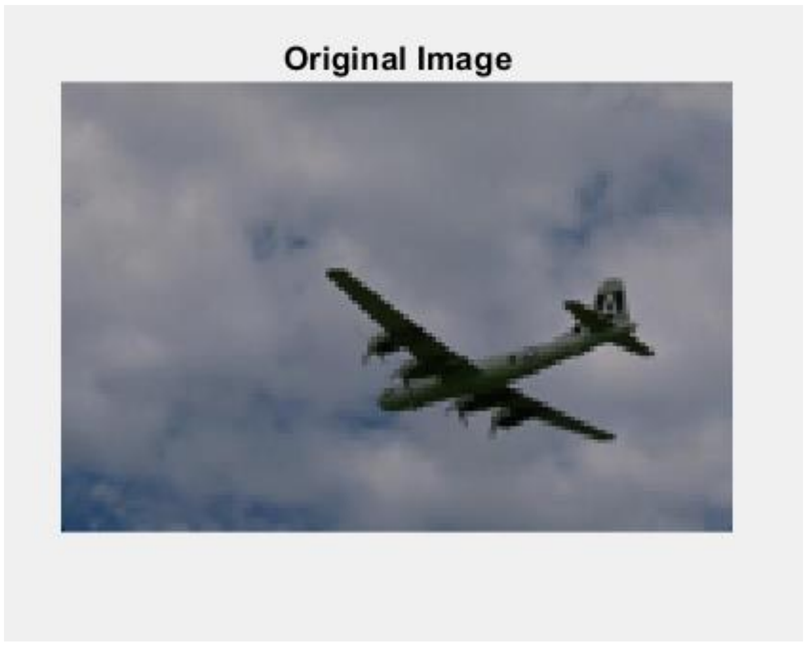

Figure 6: Original Image

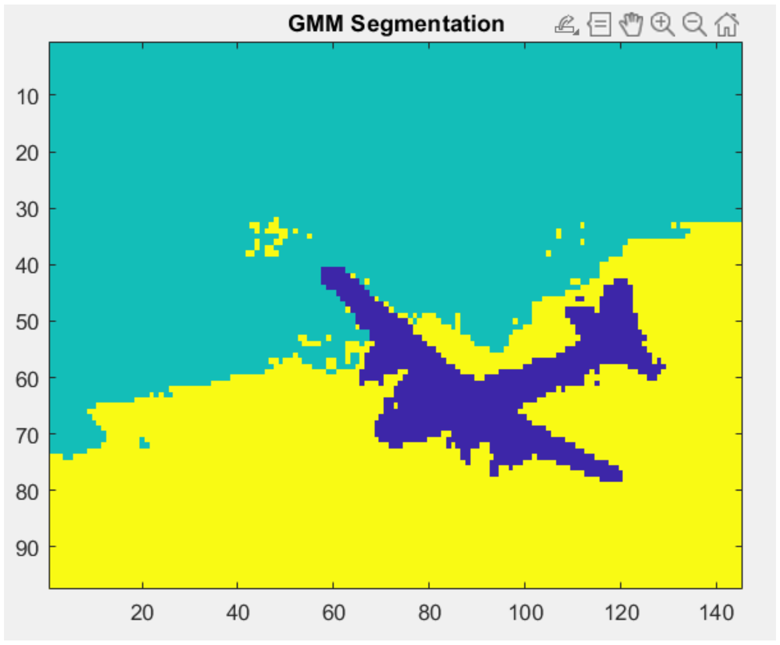

Figure 7: GMM Segmentation

Appendix A: Code for Question 1

```
%This code received significant help from
%2019Summer1 Exam4Q1_GenerateData.m
%2019Summer1 Exam 4 Solution

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SVM
clear;
close all;
% generate data
num = 1000;
N = randn(num,2);
theta = unifrnd(-pi,pi,num,1);
rl = [2*ones(num/2,1);4*ones(num/2,1)];
data = [rl.*cos(theta) rl.*sin(theta)] + N;
labels = [-ones(num/2,1);ones(num/2,1)];

figure,title('Training Dataset'),hold on
plot(data(labels==-1,1),data(labels==1,2),'ro');
plot(data(labels==1,1),data(labels==1,2),'bo');
xlabel("x1");
ylabel("x2");

%%%
KSs = [0.1 0.5 1 1.5 2 2.5 3 3.5 4]; % box constraints parameter
BCs = [0.1 0.25 0.5 0.75 1 1.25 1.5 1.75 2]; % Gaussian kernel width

for ii = 1:length(BCs)
    for jj = 1:length(KSs)
    fprintf('ii = %d / %d, jj = %d / %d ',ii, length(BCs),jj, length(KSs))
    % SVM
    BC = BCs(ii);
    KF = 'gaussian';
    KS = KSs(jj);
    PO = 3;
    OF = 0;
    T=templateSVM( 'BoxConstraint',BC,'KernelFunction',KF,'KernelScale',KS);

    rand('seed',0);
    index = randperm(num);
    for fold=1:10
        foldwd = round(num/10);
        idx1 = (fold-1)*foldwd + 1;

        %%%
        test_data = data([idx1:idx1+foldwd-1],:);
        test_label = labels([idx1:idx1+foldwd-1]);

        train_data = data;
        train_data([idx1:idx1+foldwd-1],:)=[];
        train_label = labels;
        train_label([idx1:idx1+foldwd-1],:)=[];
        % train svm
        svm = fitcecoc(train_data, train_label,'Learners',T);
        [predict_label,predict_scores] = predict(svm, test_data);

        % training dataset accuracy
        accuracy = sum(predict_label == test_label)/numel(test_label);
        fprintf('svm accuracy is %.02f \n',accuracy)
        acc_fold(fold) = accuracy;
    end
    accs(ii,jj) = mean(acc_fold);
    fprintf(' fold mean svm accuracy is %.02f \n',mean(acc_fold))
    end
end

figure, imagesc(accs), title(['SVM K-FOLD Hyperparameter Validation ' ...
    'Performance']), xlabel('Box Constraints'), ylabel(['Gaussian ' ...
    'Kernal Width'])
colorbar;

% generate test data
numT = 10000;
N = randn(numT,2);
theta = unifrnd(-pi,pi,numT,1);
rl = [2*ones(numT/2,1);4*ones(numT/2,1)];
dataT = [rl.*cos(theta) rl.*sin(theta)] + N;
labelsT = [-ones(numT/2,1);ones(numT/2,1)];

% select best svm model
[macc, midx] = max(accs(:));
[I,J] = ind2sub(length(BCs),length(KSs));
bestBC = BCs(I);
bestKS = KSs(J);
%
BC = bestBC;
KF = 'gaussian';
KS = bestKS;
PO = 3;
OF = 0;

T=templateSVM( 'BoxConstraint',BC,'KernelFunction',KF,'KernelScale',KS);


% Train svm
svm = fitcecoc(dataT, labelsT,'Learners',T);
[predict_label,predict_scores] = predict(svm, dataT);

% Testing dataset accuracy
accuracy = sum(predict_label == labelsT)/numel(labelsT);
fprintf('best BC = %f, best Ks = %f \n',bestBC, bestKS)
fprintf('svm accuracy is %.02f \n',accuracy)

figure,title('Testing Dataset'),hold on
plot(dataT(labelsT==-1,1),dataT(labelsT==-1,2),'ro');
plot(dataT(labelsT==1,1),dataT(labelsT==1,2),'bo');
```

```
figure,title('SVM Classification Performance'),hold on
plot(dataT(labelsT==1,1),dataT(labelsT==1,2),'bo');
plot(dataT(labelsT==-1,1),dataT(labelsT==-1,2),'bo');
plot(dataT(predict_label~=labelsT,1),dataT(predict_label~=labelsT,2),'ro')
xlabel("x1");
ylabel("x2");
legend('Correct Classification','','Incorrect Classification')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MLP
clear all;
close all;
clc;
% generate data
num = 1000;
N = randn(num,2);
theta = unifrnd(-pi,pi,num,1);
rl = [2*ones(num/2,1);4*ones(num/2,1)];
data = [rl.*cos(theta) rl.*sin(theta)] + N;
labels = [-ones(num/2,1);ones(num/2,1)];

figure,title('Training Dataset'),hold on
plot(data(labels==-1,1),data(labels==1,2),'ro');
plot(data(labels==1,1),data(labels==1,2),'bo');

labels(labels==-1)=2;

%%%

numPerc = 15;
for i = 1:numPerc
    trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.
    % Create a Pattern Recognition Network
    hiddenLayerSize = i;
    net.divideParam.trainRatio = 1;
    net.divideParam.valRatio = 0;
    net.divideParam.testRatio = 0;
    net = patternnet(hiddenLayerSize);

    rand('seed',0);
    index = randperm(num);
    for fold=1:10
        foldwd = round(num/10);
        idx1 = (fold-1)*foldwd + 1;

        %
        test_data = data([idx1:idx1+foldwd-1],:);
        test_label = labels([idx1:idx1+foldwd-1]);

        train_data = data;
        train_data([idx1:idx1+foldwd-1],:)=[];
        train_label = labels;
        train_label([idx1:idx1+foldwd-1],:)=[];
        train_y = zeros(length(train_label),2);
        for tt=1:length(train_label)
            train_y(tt,train_label(tt)) = 1;
        end

        % train MLP
        [net,tr] = train(net,train_data',train_y');
        predict_label = net(test_data');
        [~,predict_label] = max(predict_label,[],1);
        predict_label = predict_label';

        % accuracy
        accuracy = sum(predict_label == test_label)/numel(test_label);
        fprintf('mlp accuracy is %.02f \n',accuracy)
        acc_fold(fold) = accuracy;
    end
    accs(i) = mean(acc_fold);
    fprintf(' mean mlp accuracy is %.02f \n',mean(acc_fold))
end

figure;
plot(1:15,accs,'*')
xlabel('Number of Perceptrons')
ylabel('MLP Accuracy')
set(gca,'XTick',[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]);
title('MLP K-Fold Hyperparameter Validation Performance')
% generate test data
num = 10000;
N = randn(num,2);
theta = unifrnd(-pi,pi,num,1);
rl = [2*ones(num/2,1);4*ones(num/2,1)];
data = [rl.*cos(theta) rl.*sin(theta)] + N;
labels = [-ones(num/2,1);ones(num/2,1)];
labels(labels==-1)=2;

test_y = zeros(length(labels),2);
for tt=1:length(labels)
    test_y(tt,labels(tt)) = 1;
end

% select best mlp model
[macc, midx] = max(accs(:));
bestLS = midx;
%
trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.
% Create a Pattern Recognition Network
hiddenLayerSize = midx;
net.divideParam.trainRatio = 1;
net.divideParam.valRatio = 0;
net.divideParam.testRatio = 0;
net = patternnet(hiddenLayerSize);

% Train MLP
[net,tr] = train(net,data',test_y');
predict_label = net(data');
```

```
[~,predict_label] = max(predict_label,[],1);
predict_label = predict_label';

%
accuracy = sum(predict_label == labels)/numel(labels);
fprintf('best num of perceptrons = %f \n',bestLS)
fprintf('mlp accuracy is %.02f \n',accuracy)

figure,title('Testing Dataset'),hold on
plot(data(labels==2,1),data(labels==2,2),'ro');
plot(data(labels==1,1),data(labels==1,2),'bo');

figure,title('MLP Classification Performance'),hold on
plot(data(labels==2,1),data(labels==2,2),'bo');
plot(data(labels==1,1),data(labels==1,2),'bo');
plot(data(predict_label~=labels,1),data(predict_label~=labels,2),'ro')
xlabel("x1");
ylabel("x2");
legend('Correct Classiciation','','Inorrect Classiciation')
```

Appendix B: Code for Question 2

```
%This code received help from
%2020Summer2 HW4 Solution


clc;
clear all;
close all;
warning off;
%load test.mat;
%read image
im = imread('3096_color.jpg');

im = imresize(im,0.3);
%s=x;
[x,y,z] = size(im);
fea = reshape(im,[x*y,z]);
t=0;
for i=1:y
    for j=1:x
        t=t+1;
        coord(t,:) = [j,i];
    end
end

coord_fea = double([coord fea]);


%normalized data
data = coord_fea;
data = mapminmax(data',0,1)';

cluster_num = 3;


rand('seed',0);
%
Mu=0.5*rand(3,5);
temp = rand(5,5);
temp =(temp+temp')/2;
Sigma(:,:,1) = temp;
Sigma(:,:,2) = temp;
Sigma(:,:,3) = temp;
Pcom=[1/3 1/3 1/3];
S = struct('mu',Mu,'Sigma',Sigma,'ComponentPropotion',Pcom);

%10 fold cross validation
opt.MaxIter = 5000;
k_fold = 10;
idx = crossvalind('Kfold',size(data,1),k_fold);
ll = [];

%Gaussians for GMM model
N=length(data);
```

```
numValIters=10;
%Setup cross validation on training data
partSize=floor(N/k_fold);
partInd=[1:partSize:N length(data)];
%
for numComponents = 1:10
    %prob = [];
    for i = 1:k_fold
        val=partInd(i):partInd(i+1);
        train=setdiff(1:N,val);
        gm = fitgmdist(data(train,:),numComponents,'Replicates',5);
        if gm.Converged
            p=pdf(gm,data(val,:));
            prob(i)=sum(log(p));
        else
            prob(i)=0;
        end
    end
    ll(numComponents)=mean(prob);
    fprintf('NumGMM: %1.0f\n',numComponents);
end
for i = 1:length(ll)
    fprintf('Log likelohood %0.2f\n',ll(i));
end
figure;
stem(ll);
xlabel('Number of GMMs');
ylabel('Log-likelihood');
title('10-fold Cross Validation')
%set(gca,'XTick',[1 2 3 4 5 6 7 8 9 10]);
%}

GMModel = fitgmdist(data,cluster_num);

T1 = cluster(GMModel,data);

cen=[mean(data(T1==1,:));...
    mean(data(T1==2,:));...
    mean(data(T1==3,:))];
dist=sum(cen.^2,2);
[dump,sortind]=sort(dist,'ascend');
newT1=zeros(size(T1));
for i =1:3
    newT1(T1==i)=find(sortind==i);
end

figure,imshow(im), title('Original Image')
figure,imagesc(reshape(newT1,x,y)),title('GMM Segmentation')
```