# Homework 4

Zili Huang

## Question 1:

**Data Distribution and Generate Data:**

In this case, generating 1000 training sample pairs and 10000 test sample pairs from 'exam4q1_generateData.m'. Training a single hidden layer MLP by y = f (x) + v where v is assumed to be a zero-mean $\sigma^2$-variance additive Gaussian noise. Generating training and test data: figure 1 and figure 2.

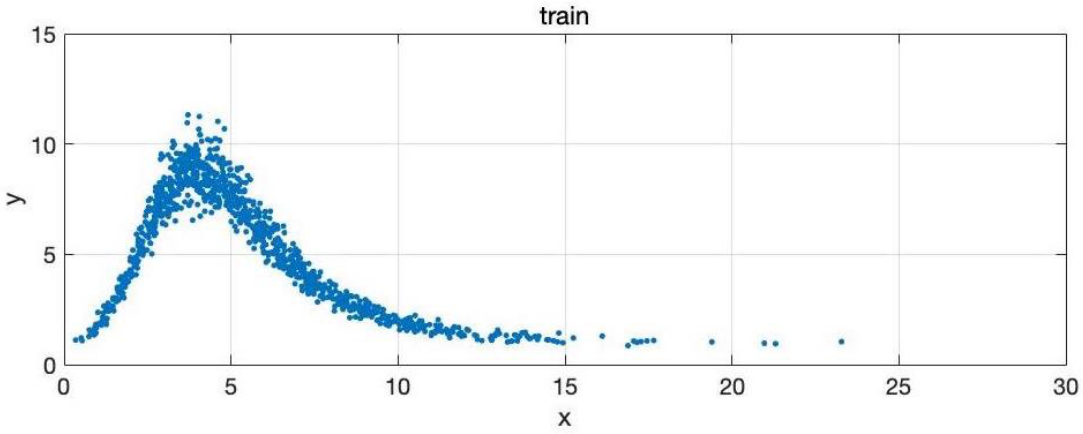


*Figure 1*

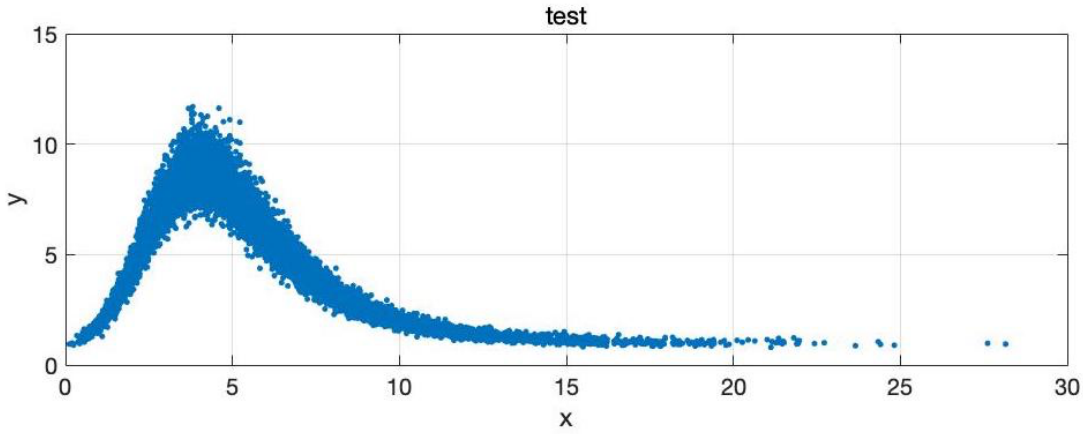


*Figure 2*

Selecting the minimum (best) average of min-squared error (MSE) in this 10-fold cross validation. Figure 3 showed the relationship between minimum squared errors and number of perceptrons. MLP with best MSE has 14 perceptrons.

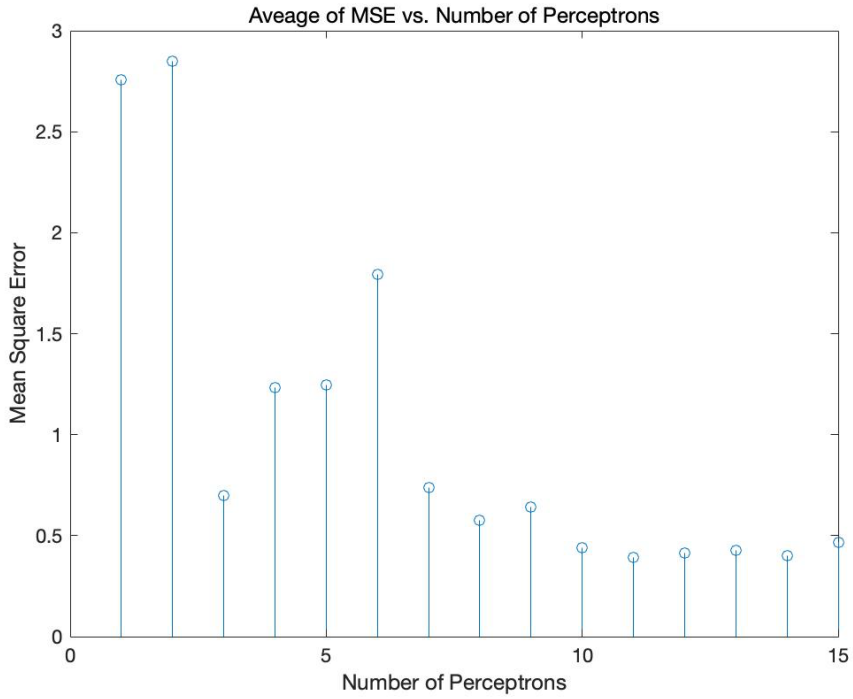


*Figure 3*

Comparing MLP has 5 perceptrons and 14 perceptrons from 10-cross validation (figure 4 and 5). The validation training data in MLP has 14 perceptrons fit much better than MLP has 5 perceptrons. Based on this example and selecting rules. MLP with 14 perceptrons be chose to train.

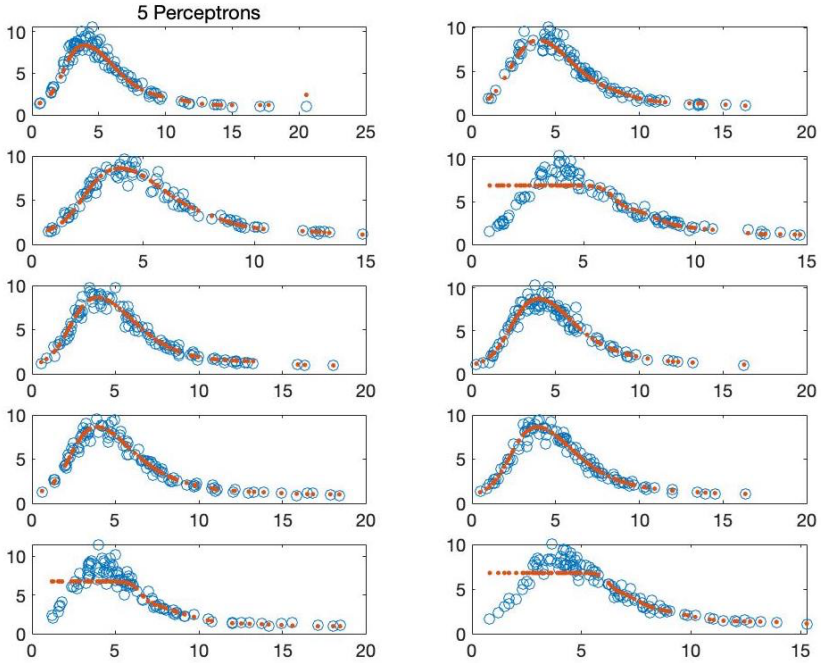


*Figure 4*

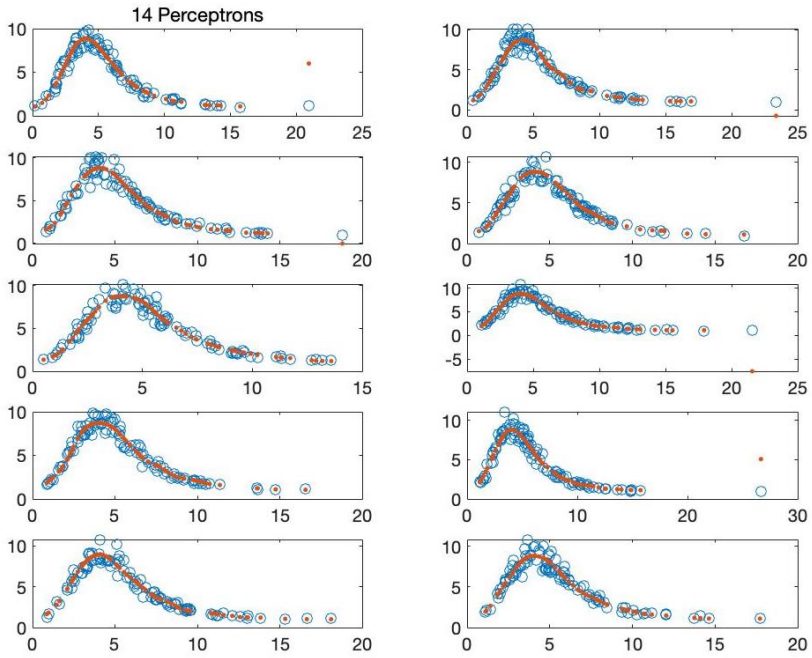


*Figure 5*

Using MLP has 14 perceptrons to train data, showed in figure 6. The MSE in this time equal to 0.34. In x axis range from 0 to 24, the model fit well. For range x greater than 24 the estimated line became deviated. The reason causes overfitting might be lack of test data in this range.

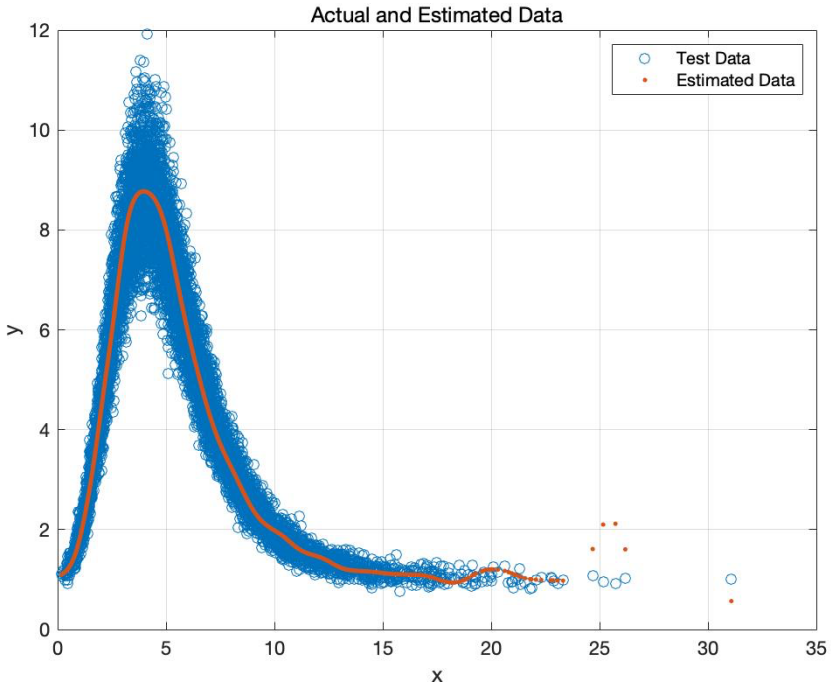


*Figure 6*

## Question 2:

In this case, generate a 2-class training set with 1000 samples and test data set with 10000 samplesby support vector machine (SVM) classifier with Gaussian kernel and ‘generateMultiRingDataset.m’ function. 10-fold cross validation was used to select the best hyperparameter C. Gaussian kernel used to determine hyperparameter Sigma. Figure 7 and figure 8:

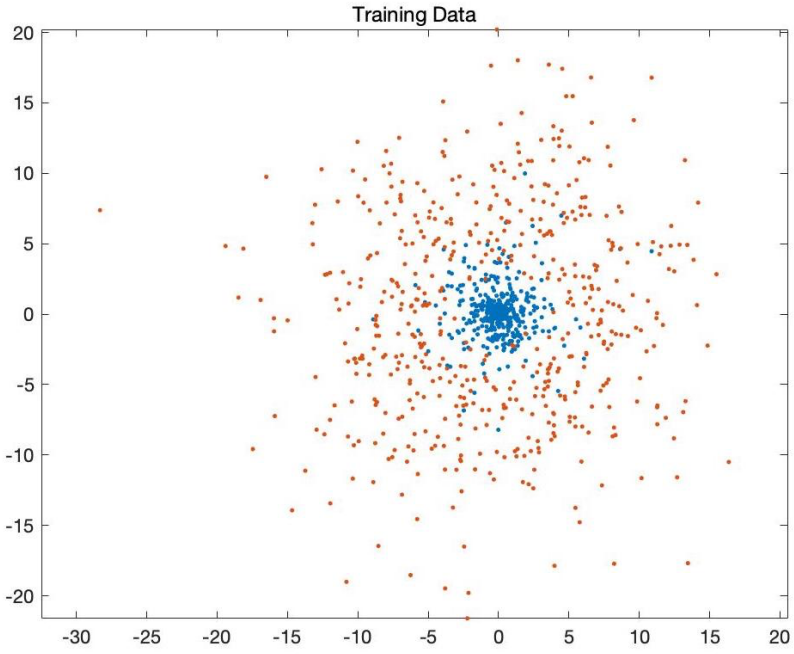


*Figure 7*

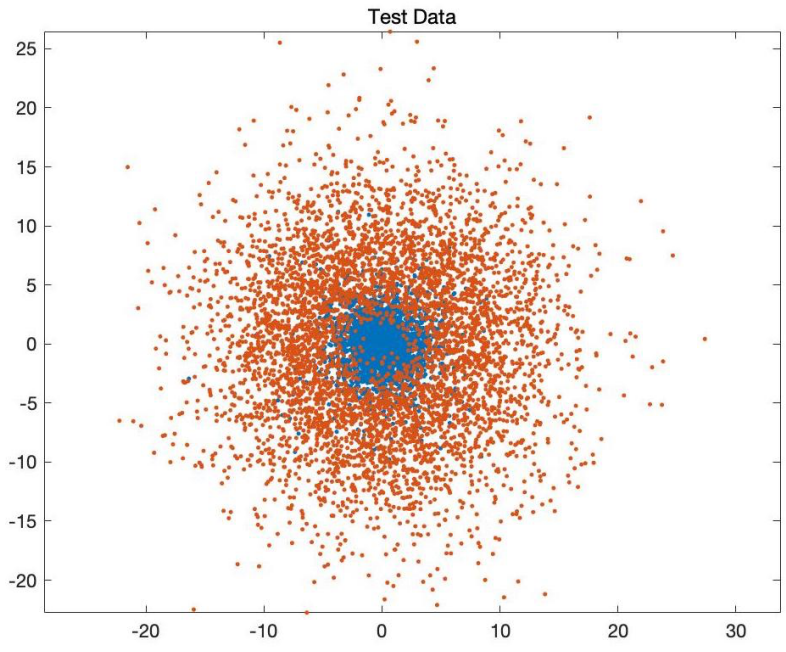


*Figure 8*

Shrinking the range of hyperparameter to find the optimal estimates.

| Run | Hyperparameter Range | | Probability of Error |
|---|---|---|---|
| | Log10(C) | Log(Sigma) | |
| 1 | 2 to 2 | -2 to 2 | 20.52% |
| 2 | 1.5 to 1.5 | -1 to 1 | 15.55% |

From table above, the probability of error decrease when range of hyperparameter shrink.

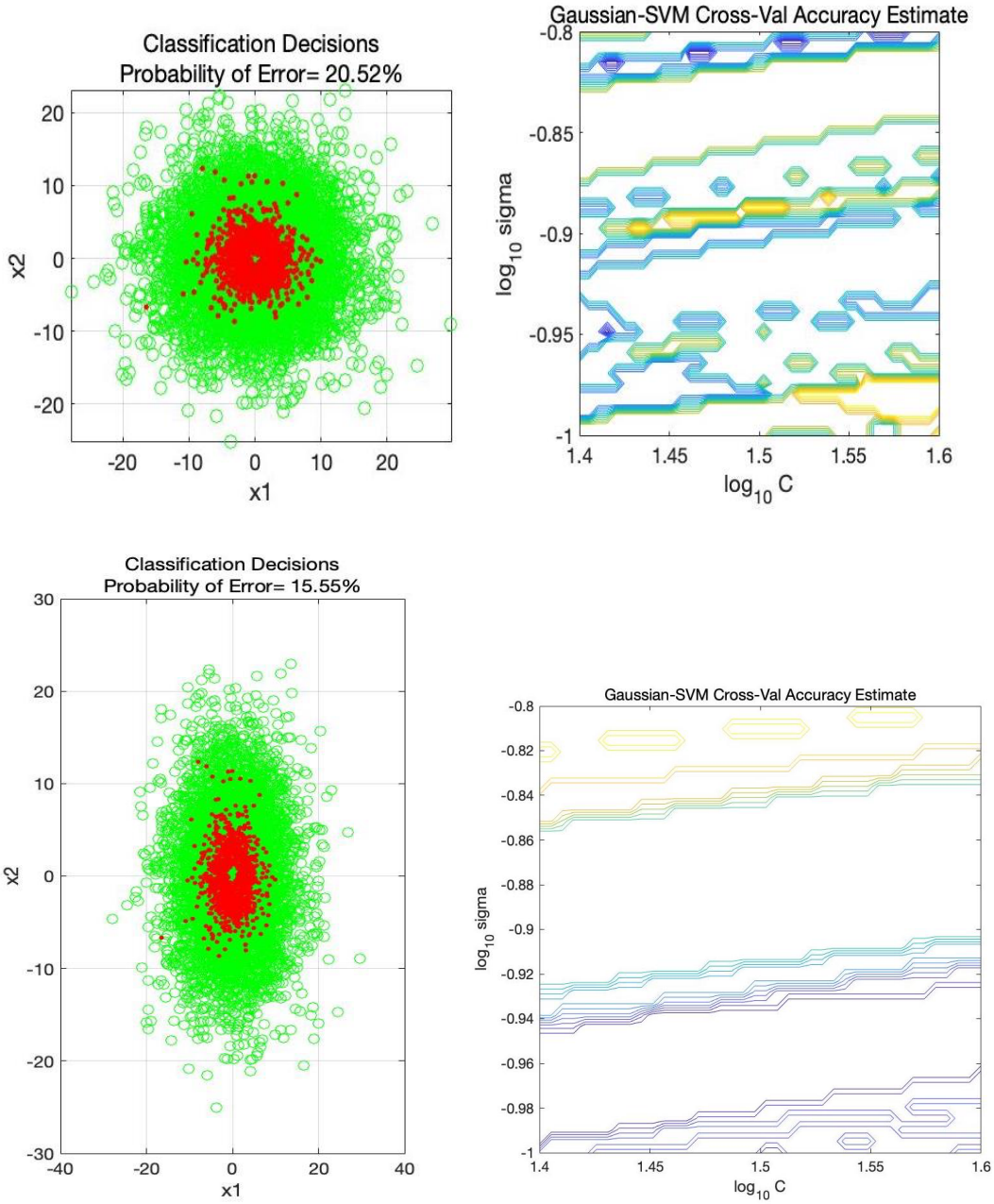


*Figure 9*

## Question 3:

In this question, using GMM-based clustering to segment the color images 3096 color.jpg plane and 42049 color.jpg bird.
Using maximum likelihood parameter estimation to fit a 2-GMM and segment the image into two parts. According to cross validation result, figure 10 and 11, the best GMM number for plane is three, and the best GMM number for bird is eight. The number of GMM first increase, then decrease and increase after decrease. Figure 12 is the image result after processing.

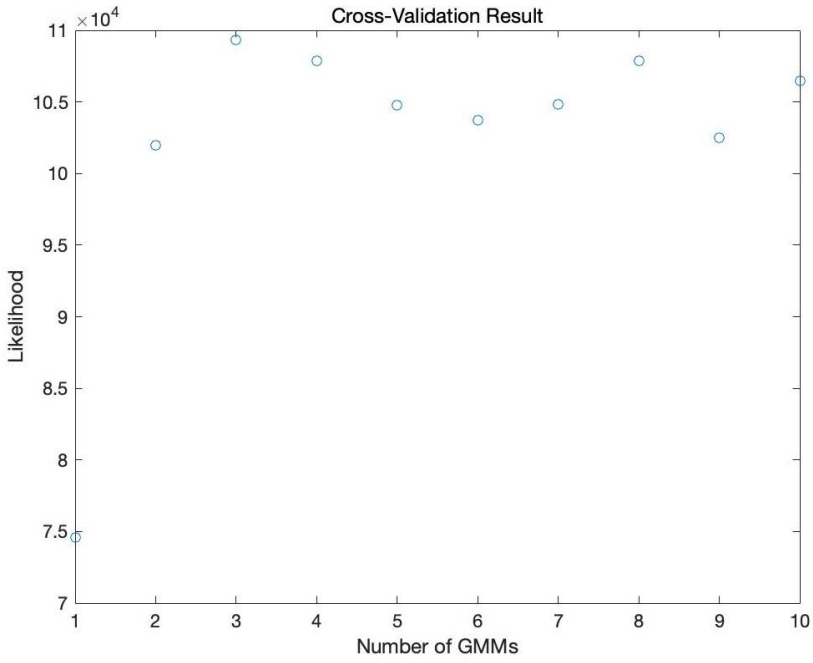


*Figure 10*

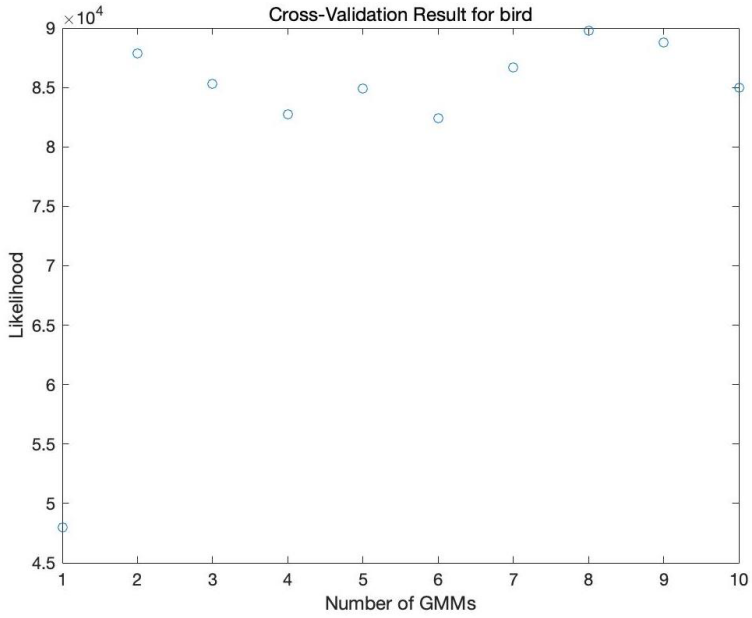


*Figure 11*

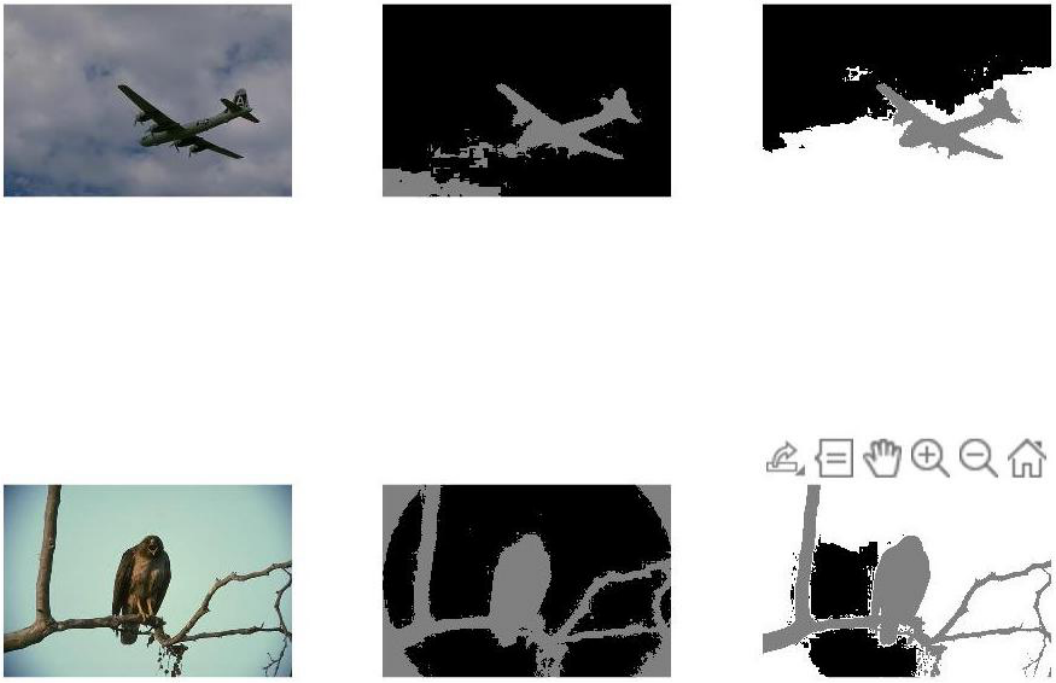

*Figure 12*

# Appendix

## Question1:

```
clear all;
close all; clc;

dTypes={'train'; 'test'};
D.train.N=1e3;
D.test.N=10e3;

%Generate Data
figure;
for ind=1:length(dTypes)
[D.(dTypes{ind}).x,D.(dTypes{ind}).y] =...
    exam4q1_generateData(D.(dTypes{ind}).N);
subplot(2,1,ind);
plot(D.(dTypes{ind}).x,D.(dTypes{ind}).y,'.');
xlabel('x');ylabel('y'); grid on;
xlim([0 30]); ylim([0 15]); title([dTypes{ind}]);
end

%Train and Validate Data
numPerc=15;
k=10;

%kfold validation is in this function
[D.train.net,D.train.MSEtrain,optM,stats]=...
    kfoldMLP_Fit(numPerc,k,D.train.x, D.train.y);
%Produce validation data from test dataset
yVal=D.train.net(D.test.x);
%Calculate MSE
MSEval=mean((yVal-D.test.y).^2);

%Plot number of perceptrons vs. pFE for the cross validation runs
for ind=1:length(dTypes)-1
    stem(stats.M,stats.avgMSE);
    xlabel('Number of Perceptrons');
    ylabel('Mean Square Error');
    title('Aveage of MSE vs. Number of Perceptrons');
end

%Print and plot results
fprintf('MSE=%1.2f%\n',MSEval);
```

```
figure;
plot(D.test.x,D.test.y,'o','DisplayName','Test Data'); hold all;
plot(D.test.x,yVal,'.','DisplayName','Estimated Data');
xlabel('x');ylabel('y');grid on;
title('Actual and Estimated Data');
legend 'show';


function [outputNet,outputMSE, optM,
stats]=kfoldMLP_Fit(numPerc,k,x,y)
N=length(x);
numValIters=10;
%Setup cross validation on training data
partSize=N/k;
partInd=[1:partSize:N length(x)];
%Perform cross validation to select number of perceptrons
for M=1:numPerc
    figure;
    for ind=1:k
        %Separate training and validation data
        index.val=partInd(ind):partInd(ind+1)-1;
        index.train=setdiff(1:N,index.val);
         net=feedforwardnet(M);
         net=train(net,x(:,index.train),y(:,index.train));
         %Validate with remaining data
         yVal=net(x(:,index.val));
         MSE(ind)=mean((yVal-y(:,index.val)).^2);
         %Plot overlay of model and validation data
         subplot(5,2,ind); plot(x(:,index.val),y(:,index.val),'o');
         hold all;
         plot(x(:,index.val),yVal,'.')
         if ind ==1
            title([num2str(M) ' Perceptrons']);
         end
    end
    avgMSE(M)=mean(MSE);
    stats.M=1:M;
    stats.avgMSE=avgMSE;
    stats.mMSE(:,M)=MSE;
end
[~,optM]=min(avgMSE);
    for ind=1:numValIters
    netName(ind)={['net' num2str(ind)]};
    finalnet.(netName{ind})=patternnet(optM);
```

```
    finalnet.(netName{ind})=train(net,x,y);
    yVal=finalnet.(netName{ind})(x);
    MSEFinal(ind)=mean((yVal-y).^2);
    end
[minMSE,outInd]=min(MSEFinal);
stats.finalMSE=MSEFinal;
outputMSE=minMSE;
outputNet=finalnet.(netName{outInd});
end
```

## Question 2:

```
if 1
    if 0
        clear all;
numL=2;
k=10;
D.train.N=1e3;
D.test.N=10e3;
[D.train.x,D.train.labels] =
generateMultiringDataset(numL,D.train.N);
D.train.labels(D.train.labels==1)=-1;
D.train.labels(D.train.labels==2)=1;
figure(1);title('Training Data');
[D.test.x,D.test.labels] = generateMultiringDataset(numL,D.test.N);
D.test.labels(D.test.labels==1)=-1;
D.test.labels(D.test.labels==2)=1;
figure(2);title('Test Data');
    end
%Cross Validation to select parameters
%Setup cross validation on training data
partSize=D.train.N/k;
partInd=[1:partSize:D.train.N length(D.train.x)+1];
%Hyperparameter search parameters
sigmaList=logspace(-1,1,40);
cList=logspace(1.5,1.5,40);

%Perform cross validation to select model parameters
avgPFE=zeros(length(cList),length(sigmaList));
for SigInd=1:length(sigmaList)
    for Cind=1:length(cList)
        for ind = 1:k
            index.val = partInd(ind):partInd(ind+1)-1;
```

```
            index.train = setdiff(1:D.train.N,index.val);
            SVMk =
fitcsvm(D.train.x(index.train)',D.train.labels(index.train),...
                'BoxConstraint',cList(Cind),'KernelFunction',...
                'gaussian','KernelScale',sigmaList(SigInd));
            decisions = SVMk.predict(D.train.x(index.val)')';
            indINCORRECT= D.train.labels(index.val).*decisions == -1;
            pFE=sum(indINCORRECT)/length(index.val);
        end
        %Determine average probability of error for a number of
perceptrons
        avgPFE(Cind,SigInd)=mean(pFE);
        fprintf('Sigma %1.0f/%1.0f,
C %1.0f/%1.0f\n',SigInd,length(sigmaList),Cind,length(cList));
    end
end
end

%Plot results
figure; contour(log10(cList),log10(sigmaList),1-avgPFE',20);
xlabel('log_{10} C');ylabel('log_{10} sigma');
title('Gaussian-SVM Cross-Val Accuracy Estimate'); axis equal;
%Determine hyperparameter values that minimize prob. of error
%Determine hyperparameter values that minimize prob. of error
[~,indMINpFE] = min(avgPFE(:));
[indOptC, indOptSigma] = ind2sub(size(avgPFE),indMINpFE); cOpt=
cList(indOptC);
sigmaOpt= sigmaList(indOptSigma);
%Train final model using entire training dataset
SVMopt = fitcsvm(D.train.x',D.train.labels','BoxConstraint',cOpt,...
    'KernelFunction','gaussian','KernelScale',sigmaOpt);
%Evaluate performance on test dataset
decisionsOpt=SVMopt.predict(D.test.x')';
decisionsEval=decisionsOpt.*D.test.labels;
dInc=decisionsEval==-1; dCorr=decisionsEval==1;
pFEopt=sum(dInc)/D.test.N;
fprintf('Probability of Error = %1.2f%%\n',pFEopt);


%Plot correct and incorrect decisions
plot(D.test.x(1,dCorr),D.test.x(2,dCorr),'go','DisplayName','Correct
Decisions');
hold all;
plot(D.test.x(1,dInc),D.test.x(2,dInc),'r.','DisplayName','Incorrect
```

```
Decisions');
grid on;
xlabel('x1'); ylabel('x2');
title(sprintf('Classification Decisions\nProbability of
Error= %1.2f%%',100*pFEopt));
Nx = 1001; Ny = 990;
xGrid = linspace(-10,10,Nx);
yGrid = linspace(-10,10,Ny);
[h,v] = meshgrid(xGrid,yGrid);
dGrid = SVMopt.predict([h(:),v(:)]);
zGrid = reshape(dGrid,Ny,Nx);
figure(1), subplot(1,2,2);
contour(xGrid,yGrid,zGrid,0);
xlabel('x1'), ylabel('x2'), axis equal;

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:

```
if 1
filenames = {'3096_color.jpg';'42049_color.jpg'};
dTypes={'c3096' 'c42049'};
%Cross validation parameters
NumGMMtoCheck=10;   %Number of Params
k=10;   %Number of folds
end

for ind=1:length(filenames)
imdata = imread(filenames{ind});
figure(1);
subplot(length(filenames),3,(ind-1)*3+1);
imshow(imdata);

if 1
    [R,C,D]=size(imdata);
    N=R*C;
    imdata=double(imdata);
    rows=(1:R)'*ones(1,C); columns=ones(R,1)*(1:C);
    featureData=[rows(:)';columns(:)'];
    for ind2 =1:D
        imdatad =imdata(:,:,ind2);
        featureData =[featureData; imdatad(:)'];
    end
    minf=min(featureData,[],2);
    maxf=max(featureData,[],2);
    ranges=maxf-minf;
```

```
    %Normalized data
    x=(featureData-minf)./ranges;
    %Assess for GMM with 2 Gaussians case
    GMM2=fitgmdist(x',2,'Replicates',10);
    post2=posterior(GMM2,x')';
    decisions=post2(2,:)>post2(1,:);
end
    labelImage2=reshape(decisions,R,C); %Plot GMM=2 CASE
    subplot(length(filenames),3,(ind-1)*3+2);
    imshow(uint8(labelImage2*255/2));
if 1
    N =length(x);
    numValIters =10;
    partSize=floor(N/k);
    partInd=[1:partSize:N length(x)];
    for NumGMMs=1:NumGMMtoCheck
          for NumKs=1:k
              index.val=partInd(NumKs):partInd(NumKs+1);
              index.train=setdiff(1:N,index.val);
%Create object with M perceptrons in hidden layer
              GMMk_loop=fitgmdist(x(:,index.train)',NumGMMs,...
                  'Replicates',5);
% net.layers{1}.transferFcn = 'softplus';%didn't work
              if GMMk_loop.Converged

probX(NumKs)=sum(log(pdf(GMMk_loop,x(:,index.val)')));
              else
              end
          end
          avgProb(ind,NumGMMs)=mean(probX);
stats(ind).NumGMMs=1:NumGMMtoCheck;
          stats(ind).avgProb=avgProb;
stats(ind).mProb(:,NumGMMs)=probX;
          fprintf('File: %1.0f, NumGMM: %1.0f\n',ind,NumGMMs);
    end
end
[~,optNumGMM]=max(avgProb(ind,:));

GMMk=fitgmdist(x',optNumGMM,'Replicates',10);
postk=posterior(GMMk,x')';
lossMatrix=ones(optNumGMM,optNumGMM)-eye(optNumGMM);
expectedRisks =lossMatrix*postk;
[~,decisions] = min(expectedRisks,[],1);
%Plot segmented image
```

```
labelImageK=reshape(decisions-1,R,C);
subplot(length(filenames),3,(ind-1)*3+3);
imshow(uint8(labelImageK*255/2));
end

figure(2);plot(avgProb(1,:),'o');
xlabel('Number of GMMs');ylabel('Likelihood');
title('Cross-Validation Result');
```