Forest Mapping Data Set:

```
%this is  libSvmOverForestData_v21
%in this code i combined the given testing and training data to form the
%file SPECTFlibsvm.combined. This will be used to separated out the 10%90%
%etc test-training combination
tic
clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');

% addpath to the data
dirData = './dataFromUCI/ForestTypes/';
addpath(dirData);



% read the data set
[heart_scale_label, heart_scale_inst] =
libsvmread(fullfile(dirData,'SPECTFlibsvm.combined'));
[N D] = size(heart_scale_inst)


NoOfRows=N;

%The value for testing and training index is presented in percentages
percentageTraining=10;
%fix removes all the decimal values
stopValue=fix(NoOfRows*percentageTraining/100)


% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
trainLabel = heart_scale_label(trainIndex==1,:);
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

totalData = heart_scale_inst;
size(totalData)
%return;
totalLabel = heart_scale_label;
```

```
%[N D] = size(totalData)
labelList = unique(heart_scale_label(:));

%total number of unique classes into which the classification will occur
NClass = length(labelList);



stepSize = 1;
log2c_list = -20:stepSize:20;
%size(log2c_list)
%ovrtrain was giving error that gamma values have to be positive. While
%looking at the iteration i found that best gamma comes around 5. So might
%as well change the ranges
log2g_list = -20:stepSize:20;
%size(log2g_list)
%return;
%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list)



%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);

%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = svmtrain(trainLabel, trainData, param);

        cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = get_cv_ac(trainLabel, trainData, cmd, 3);

        cvMatrix(i,j) = cv;
        %fprintf( 'value of i is %d\n', i );
        %fprintf( 'value of j is %d\n', j );
       % if (cv >= bestcv),
```

```
      %disp(log2g);
      %disp(cv);
       if (cv > bestcv),
       %     disp('************');
          bestcv = cv;
          bestLog2c = log2c ;
          bestLog2g = log2g;
          accuracy(i)=bestcv;
%             if(bestcv>95),
%                 disp(bestLog2g);
%                 disp(bestLog2c);
%
%             end
       end


   end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is
when G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
%figure;
% plot(log2g_list,accuracy);
% title('Accuracy vs G-value')
% xlabel('G Value')
% ylabel('Accuracy')




% ########################################################################
```

```
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
% ######################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);


cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = svmtrain(trainLabel, trainData, param);

        cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = get_cv_ac(trainLabel, trainData, cmd, 3);


        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
            accuracy2(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg,
bestcv);
    end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);
```

```
%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
%figure;
% plot(log2g_list,accuracy2);
% title('Accuracy2 vs G-value')
% xlabel('G Value')
% ylabel('Accuracy2')

% ########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = svmtrain(trainLabel, trainData, param);
        cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = get_cv_ac(trainLabel, trainData, cmd, 3);



        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
```

```
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
             accuracy3(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg,
bestcv);
    end
end

disp(['CV scale3: best c value is:',num2str(2^bestLog2c),' best g value
is:',num2str(2^bestLog2g),' accuracy:',num2str(bestcv*100),'%']);
disp(['also CV scale3: best value of bestLog2c  is:',num2str(bestLog2c),' best value of
bestLog2g:',num2str(bestLog2g),' accuracy:',num2str(bestcv*100),'%']);

% ########################
% Train the SVM in one-vs-rest (OVR) mode
% ########################
%for some reason ovrtrain is not accepting negative values of gamma. So
%converting it to positive values. This is Stupidand wrong, but no other
%go.
%update: figured out why ovrtrain doesnt accept negative values. Gamma
%value is 2^bestLog2g, not bestLog2g which was -10
bestParam = ['-c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g)];

model = ovrtrain(trainLabel, trainData, bestParam);
% ########################
% Classify samples using OVR model
% ########################
[predict_label, accuracy, prob_values] = ovrpredict(testLabel, testData, model);
fprintf('Accuracy = %g%%\n', accuracy * 100);



% ================================
% ===== Showing the results ======
% ================================

% Assign color for each class
%colorList = generateColorList(NClass); % This is my own way to assign the color...don't
worry about it
 %colorList = spring(100);
 colorList = prism(100);

% true (ground truth) class
trueClassIndex = zeros(N,1);
for i = 1:NClass
    trueClassIndex(totalLabel==labelList(i)) = i;
```

```
end
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
for i = 1:NClass
    resultClassIndex(predict_label==labelList(i)) = i;
end
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(totalData,'euclidean');
% newCoor = mdscale(distanceMatrix,2); % take longer time, but more beautiful
newCoor = cmdscale(distanceMatrix); %
% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
figure;
%scatter(x,y,patchSize,colorTrueClassPlot,'x');
scatter(x,y,patchSize,'blue','o');
title('whole data set');
xlabel('Forest Feature Scaled1')
ylabel('Forest Feature Scaled2')

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
patchSize = 60;% 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'o','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 60;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
title('classification results');
xlabel('Forest Feature Scaled1')
ylabel('Forest Feature Scaled2');
Toc
```

BreastCancer Dataset:

Code:

```
% this version libSvmOverBreastCancerData_v18
% is built on top of v18 of the basic code.
tic
% this code works on the data set of heart diseases given here:
%http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

%has 13 attributes
%
% 1. Sample code number: id number
% 2. Clump Thickness: 1 - 10
% 3. Uniformity of Cell Size: 1 - 10
% 4. Uniformity of Cell Shape: 1 - 10
% 5. Marginal Adhesion: 1 - 10
% 6. Single Epithelial Cell Size: 1 - 10
% 7. Bare Nuclei: 1 - 10
% 8. Bland Chromatin: 1 - 10
% 9. Normal Nucleoli: 1 - 10
% 10. Mitoses: 1 - 10
% 11. Class: (2 for benign, 4 for malignant)
%
%scaled version is found
here:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/breast-cancer_scale

% predicts 2 and 4 for class has/doesnt have Breast cancer-2 for benign, 4 for malignant

% This code just simply run the SVM on the example data set "heart_scale",
% which is scaled properly. The code divides the data into 2 parts
% train: 1 to 200
% test: 201:270
% Then plot the results vs their true class. In order to visualize the high
% dimensional data, we apply MDS to the 13D data and reduce the dimension
% to 2D

clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');
```

```
% addpath to the data
dirData = './dataFromUCI/breastcancer';
addpath(dirData);


% read the data set
[heart_scale_label, heart_scale_inst] = libsvmread(fullfile(dirData,'breast-cancer_scale'));
[N D] = size(heart_scale_inst)

NoOfRows=N;

%The value for testing and training index is presented in percentages
percentageTraining=10;
%fix removes all the decimal values
stopValue=fix(NoOfRows*percentageTraining/100)


% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
trainLabel = heart_scale_label(trainIndex==1,:);
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

% ###########################################################################
% From here on, we do 3-fold cross validation on the train data set
% ###########################################################################

% ###########################################################################
% cross validation scale 1
% run the scale from -20 to +20 with a stepsize of 1
% This is the big scale (rough)
% ###########################################################################
stepSize = 1;
log2c_list = -20:stepSize:20;
log2g_list = -20:stepSize:20;


%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);
```

```
%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);

%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = svmtrain(trainLabel, trainData, param);

        %cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = get_cv_ac(trainLabel, trainData, cmd, 3);



        cvMatrix(i,j) = cv;
        %fprintf( 'value of i is %d\n', i );
        %fprintf( 'value of j is %d\n', j );
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
             accuracy(i)=bestcv;
        end

        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg,
bestcv);
        %store the corresponding accuracy

    end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is
when G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
```

```
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
% figure;
% plot(log2g_list,accuracy);
% title('Accuracy vs G-value')
% xlabel('G Value')
% ylabel('Accuracy')



%ignoring the plotting below. It was way too complicated
% xlabel('Log_2\gamma');
% figure
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

% ########################################################################
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
% ########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);


cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
```

```
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
     cv = svmtrain(trainLabel, trainData, param);

      %cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = get_cv_ac(trainLabel, trainData, cmd, 3);



      cvMatrix(i,j) = cv;
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
         accuracy2(i)=bestcv;
      end
      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg,
bestcv);
   end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);

% % Plot the results
% figure;
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
% figure;
% plot(log2g_list,accuracy2);
```

```
% title('Accuracy2 vs G-value')
% xlabel('G Value')
% ylabel('Accuracy2')

% ###########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ###########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = svmtrain(trainLabel, trainData, param);

       % cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = get_cv_ac(trainLabel, trainData, cmd, 3);


        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
             accuracy3(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg,
bestcv);
    end
end
```

```
disp(['CV scale3: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv*100),'%']);

% Plot the results
% figure;
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy3);
title('Accuracy3 vs C-value')
xlabel('C Value')
ylabel('Accuracy3')

%plot accuracy against gamma value
% figure;
% plot(log2g_list,accuracy2);
% title('Accuracy3 vs G-value')
% xlabel('G Value')
% ylabel('Accuracy3')

%i.e by the time the code reaches here we have found a very good value for
% C and gamma.

% ###################################################################
% Test phase
% Use the parameters to classify the test set
% ###################################################################
param = ['-q -c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g), ' -b 1'];
bestModel = svmtrain(testLabel, testData, param);
[predict_label, accuracy, prob_values] = svmpredict(testLabel, testData, bestModel, '-b 1');
% test the training data

% ================================
% ===== Showing the results ======
% ================================

% Assign color for each class
```

```
%colorList = generateColorList(2); % This is my own way to assign the color...don't worry about it
 % prism is a type of color map. details about color map can be found here:http://www.mathworks.com/help/matlab/ref/colormap.html

 %parula,jet plots in blue and white
%colorList = parula(100);
%colorList = jet(100);
%colorList = winter(100);
colorList = prism(100);



% true (ground truth) class
trueClassIndex = zeros(N,1);
trueClassIndex(heart_scale_label==2) = 1;
trueClassIndex(heart_scale_label==4) = 2;
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
resultClassIndex(predict_label==2) = 1;
resultClassIndex(predict_label==4) = 2;
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(heart_scale_inst,'euclidean');

%mdscale was giving error for the breast cancer data saying :Points in the configuration have co-located.  Try a different
%starting point, or use a different criterion. colocated-so added teh
%criterion=sstress
%newCoor = mdscale(distanceMatrix,2);
newCoor = mdscale(distanceMatrix,2,'criterion', 'sstress') ;

% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
figure; scatter(x,y,patchSize,colorTrueClassPlot,'filled');
scatter(x,y,patchSize,'blue');
title('whole data set');

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
```

```
patchSize = 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'o','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 30;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
title('classification results');
toc
```

Handwriting Recognition Dataset:

```
%this is  libSvmOverForestData_v21
%in this code i combined the given testing and training data to form the
%file SPECTFlibsvm.combined. This will be used to separated out the 50%50%
%etc test-training combination
tic
clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');

% addpath to the data
dirData = './dataFromUCI/handwritingRecognition/';
addpath(dirData);



% read the data set
[heart_scale_label, heart_scale_inst] =
libsvmread(fullfile(dirData,'HandWritingRecognition.combined'));
[N D] = size(heart_scale_inst)


NoOfRows=N;

%The value for testing and training index is presented in percentages
percentageTraining=40;
%fix removes all the decimal values
stopValue=fix(NoOfRows*percentageTraining/100)


% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
trainLabel = heart_scale_label(trainIndex==1,:);
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

totalData = heart_scale_inst;
size(totalData)
%return;
totalLabel = heart_scale_label;

%[N D] = size(totalData)
labelList = unique(heart_scale_label(:));
```

```
%total number of unique classes into which the classification will occur
NClass = length(labelList);


% read the data set
% [dnaTrainLabel, dnaTrainData] = libsvmread(fullfile(dirData,'SPECTFlibsvm.combined'));
% NTrain = size(dnaTrainData,1);
% [dnaTrainLabel, permIndex] = sortrows(dnaTrainLabel);
% dnaTrainData = dnaTrainData(permIndex,:);
% [dnaTestLabel, dnaTestData] = libsvmread(fullfile(dirData,'SPECTFlibsvm.test'));
% NTest = size(dnaTestData,1);
% [dnaTestLabel, permIndex] = sortrows(dnaTestLabel);
% dnaTestData = dnaTestData(permIndex,:);
%
% % combine the data together just to fit my format
% totalData = [dnaTrainData; dnaTestData];
% totalLabel = [dnaTrainLabel; dnaTestLabel];
% %figure;
% figure('Name','Legend','NumberTitle','off')
% subplot(1,2,1); imagesc(totalLabel); title('class label');
% subplot(1,2,2); imagesc(totalData); title('features');
%
% [N D] = size(totalData);
% labelList = unique(totalLabel(:));
% NClass = length(labelList);
%
% % ########################
% % Determine the train and test index
% % ########################
% trainIndex = zeros(N,1); trainIndex(1:NTrain) = 1;
% testIndex = zeros(N,1); testIndex( (NTrain+1):N) = 1;
% trainData = totalData(trainIndex==1,:);
% trainLabel = totalLabel(trainIndex==1,:);
% testData = totalData(testIndex==1,:);
% testLabel = totalLabel(testIndex==1,:);
%
% ########################
% Parameter selection using 3-fold cross validation
% this part is replaced with the code from libSvmOverBreastCancerData_v18
% where we vary C and Gamma for 3 cycles.
% % ########################
% bestcv = 0;
% for log2c = -1:1:3,
%   for log2g = -4:1:2,
%     cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
%     cv = get_cv_ac(trainLabel, trainData, cmd, 3);
%     if (cv >= bestcv),
%       bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
%     end
%     fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
```

```
%   end
% end
stepSize = 2;
log2c_list = -10:stepSize:10;
%size(log2c_list)
%ovrtrain was giving error that gamma values have to be positive. While
%looking at the iteration i found that best gamma comes around 5. So might
%as well change the ranges
log2g_list = -10:stepSize:10;
%size(log2g_list)
%return;
%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list)


%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);

%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
       % param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = svmtrain(trainLabel, trainData, param);
         cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
         cv = get_cv_ac(trainLabel, trainData, cmd, 3);
        cvMatrix(i,j) = cv;
        %fprintf( 'value of i is %d\n', i );
        %fprintf( 'value of j is %d\n', j );
       % if (cv >= bestcv),
       %disp(log2g);
       fprintf('reaching here inside the first cvscale1 with i value %d and j value %d',i,j);
         if (cv > bestcv),
        %     disp('**********');
            bestcv = cv;
            bestLog2c = log2c ;
            bestLog2g = log2g;
            accuracy(i)=bestcv;
%             if(bestcv>95),
%                 disp(bestLog2g);
%                 disp(bestLog2c);
%
```

```
%            end
      end


   end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is when
G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy);
title('Accuracy vs G-value')
xlabel('G Value')
ylabel('Accuracy')




% ########################################################################
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
% ########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);
```

```
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
       % param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = svmtrain(trainLabel, trainData, param);
         cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
         cv = get_cv_ac(trainLabel, trainData, cmd, 3);
        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
            accuracy2(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
    end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);


%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy2);
title('Accuracy2 vs G-value')
xlabel('G Value')
ylabel('Accuracy2')

% ########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);
```

```
%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
     %  cv = svmtrain(trainLabel, trainData, param);
      %  cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
       cv = get_cv_ac(trainLabel, trainData, cmd, 3);
      cvMatrix(i,j) = cv;
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
          accuracy3(i)=bestcv;
      end
      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
   end
end

disp(['CV scale3: best c value is:',num2str(2^bestLog2c),' best g value is:',num2str(2^bestLog2g),'
accuracy:',num2str(bestcv),'%']);
disp(['also CV scale3: best value of bestLog2c  is:',num2str(bestLog2c),' best value of
bestLog2g:',num2str(bestLog2g),' accuracy:',num2str(bestcv),'%']);

% ########################
% Train the SVM in one-vs-rest (OVR) mode
% ########################
%for some reason ovrtrain is not accepting negative values of gamma. So
%converting it to positive values. This is Stupidand wrong, but no other
%go.
%update: figured out why ovrtrain doesnt accept negative values. Gamma
%value is 2^bestLog2g, not bestLog2g which was -10
bestParam = ['-c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g)];

model = ovrtrain(trainLabel, trainData, bestParam);
% ########################
% Classify samples using OVR model
% ########################
[predict_label, accuracy, prob_values] = ovrpredict(testLabel, testData, model);
fprintf('Accuracy = %g%%\n', accuracy * 100);


% ================================
% ===== Showing the results ======
% ================================
```

```
% Assign color for each class
%colorList = generateColorList(NClass); % This is my own way to assign the color...don't worry
about it
 colorList = prism(100);

% true (ground truth) class
trueClassIndex = zeros(N,1);
for i = 1:NClass
    trueClassIndex(totalLabel==labelList(i)) = i;
end
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
for i = 1:NClass
    resultClassIndex(predict_label==labelList(i)) = i;
end
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(totalData,'euclidean');
% newCoor = mdscale(distanceMatrix,2); % take longer time, but more beautiful
newCoor = cmdscale(distanceMatrix); %
% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
figure;
%scatter(x,y,patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,'blue');
title('whole data set');
xlabel('Handwriting Feature Scaled1')
ylabel('Handwriting Feature Scaled2')

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
patchSize = 30;% 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'O','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 30;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
title('classification results');
```

```
xlabel('Handwriting Feature Scaled1')
ylabel('Handwriting Feature Scaled2')

toc
```

Heart Disease Data set:

```
% this version libSvmOverHeartScaleData_v17
%1. separates testing and training data into 20% training and 80% testing
tic

% previous version libSvmOverHeartScaleData_v16 plots accuracy vs cValue graph
%also this version does 3 fold cross validation : -v 3
%param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];


% this code works on the data set of heart diseases given here:
%https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

%has 13 attributes
%
% Attribute Information:
% ------------------------
% -- 1. age
% -- 2. sex
% -- 3. chest pain type (4 values)
% -- 4. resting blood pressure
% -- 5. serum cholestoral in mg/dl
% -- 6. fasting blood sugar > 120 mg/dl
% -- 7. resting electrocardiographic results (values 0,1,2)
% -- 8. maximum heart rate achieved
% -- 9. exercise induced angina
% -- 10. oldpeak = ST depression induced by exercise relative to rest
% -- 11. the slope of the peak exercise ST segment
% -- 12. number of major vessels (0-3) colored by flourosopy
% -- 13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

% predicts +1 and -1 for has/doesnt have heart disease

% This code just simply run the SVM on the example data set "heart_scale",
% which is scaled properly. The code divides the data into 2 parts
% train: 1 to 200
% test: 201:270
% Then plot the results vs their true class. In order to visualize the high
% dimensional data, we apply MDS to the 13D data and reduce the dimension
% to 2D

clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');
```

```
% addpath to the data
dirData = './dataFromUCI/heart/';
addpath(dirData);


% read the data set
[heart_scale_label, heart_scale_inst] = libsvmread(fullfile(dirData,'heart_scale'));
[N D] = size(heart_scale_inst)

NoOfRows=N;

%The value for testing and training index is presented in percentages
percentageTraining=50;
stopValue=NoOfRows*percentageTraining/100


% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
trainLabel = heart_scale_label(trainIndex==1,:);
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

% ###########################################################################
% From here on, we do 3-fold cross validation on the train data set
% ###########################################################################

% ###########################################################################
% cross validation scale 1
% run the scale from -20 to +20 with a stepsize of 1
% This is the big scale (rough)
% ###########################################################################
stepSize = 1;
log2c_list = -20:stepSize:20;
log2g_list = -20:stepSize:20;


%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);


%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);
```

```
%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = svmtrain(trainLabel, trainData, param);
      cvMatrix(i,j) = cv;
      %fprintf( 'value of i is %d\n', i );
      %fprintf( 'value of j is %d\n', j );
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
          accuracy(i)=bestcv;
      end

      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
      %store the corresponding accuracy

   end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is when
G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy);
title('Accuracy vs G-value')
xlabel('G Value')
ylabel('Accuracy')



%ignoring the plotting below. It was way too complicated
% xlabel('Log_2\gamma');
```

```
% figure
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

% ###############################################################################
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
% ###############################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);


cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = svmtrain(trainLabel, trainData, param);
        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
            accuracy2(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
    end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);

% % Plot the results
% figure;
```

```
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy2);
title('Accuracy2 vs G-value')
xlabel('G Value')
ylabel('Accuracy2')

% ###########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ###########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        cv = svmtrain(trainLabel, trainData, param);
        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
```

```
            accuracy3(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
    end
end

disp(['CV scale3: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);

% Plot the results
% figure;
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy3);
title('Accuracy3 vs C-value')
xlabel('C Value')
ylabel('Accuracy3')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy2);
title('Accuracy3 vs G-value')
xlabel('G Value')
ylabel('Accuracy3')

%i.e by the time the code reaches here we have found a very good value for
% C and gamma.

% ###################################################################
% Test phase
% Use the parameters to classify the test set
% ###################################################################
param = ['-q -c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g), ' -b 1'];
bestModel = svmtrain(testLabel, testData, param);
[predict_label, accuracy, prob_values] = svmpredict(testLabel, testData, bestModel, '-b 1'); % test
the training data

% ================================
% ===== Showing the results ======
% ================================

% Assign color for each class
```

```
%colorList = generateColorList(2); % This is my own way to assign the color...don't worry about it
 % prism is a type of color map. details about color map can be found
here:http://www.mathworks.com/help/matlab/ref/colormap.html

 %parula,jet plots in blue and white
%colorList = parula(100);
%colorList = jet(100);
%colorList = winter(100);
colorList = prism(100);



% true (ground truth) class
trueClassIndex = zeros(N,1);
trueClassIndex(heart_scale_label==1) = 1;
trueClassIndex(heart_scale_label==-1) = 2;
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
resultClassIndex(predict_label==1) = 1;
resultClassIndex(predict_label==-1) = 2;
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(heart_scale_inst,'euclidean');
newCoor = mdscale(distanceMatrix,2);

% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
figure;
%scatter(x,y,patchSize,colorTrueClassPlot,'filled');
scatter(x,y,patchSize,'blue','o');
title('whole data set');

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
patchSize = 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'o','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 30;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
```

```
title('classification results');

toc
```

Tic Tac Toe Dataset:

```
tic
% this version is libSvmOverTicTacToeData1090_v1
%

% this code works on the data set of tic tac toe given here:
%http://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame
%
%scaled version is found
here:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/breast-cancer_scale

% predicts 2 and 4 for class has/doesnt have Breast cancer-2 for benign, 4 for malignant

% This code just simply run the SVM on the example data set "heart_scale",
% which is scaled properly. The code divides the data into 2 parts
% train: 1 to 200
% test: 201:270
% Then plot the results vs their true class. In order to visualize the high
% dimensional data, we apply MDS to the 13D data and reduce the dimension
% to 2D

clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');

% addpath to the data
dirData = './dataFromUCI/TicTacToe';
addpath(dirData);


% read the data set
[heart_scale_label, heart_scale_inst] = libsvmread(fullfile(dirData,'tictactoe2.combined'));
[N D] = size(heart_scale_inst)
[N D] = size(heart_scale_label)

mergedData=[heart_scale_label heart_scale_inst ];
size(mergedData);

%shuffling the contents of training data
trainDataSize=size(mergedData,1);
ix = randperm(trainDataSize);
mergedData = mergedData(ix,:);
trainDataSize=size(mergedData);
heart_scale_label = mergedData(:, 1);
heart_scale_inst=mergedData(:, 2:end);
```

```
[N D] = size(heart_scale_inst)
[N D] = size(heart_scale_label)


NoOfRows=N;

%The value for testing and training index is presented in percentages
percentageTraining=50;
%fix removes all the decimal values
stopValue=fix(NoOfRows*percentageTraining/100);


% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
trainLabel = heart_scale_label(trainIndex==1,:)

%return;
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

% ###########################################################################
% From here on, we do 3-fold cross validation on the train data set
% ###########################################################################

% ###########################################################################
% cross validation scale 1
% run the scale from -20 to +20 with a stepsize of 1
% This is the big scale (rough)
% ###########################################################################
stepSize = 1;
log2c_list = -20:stepSize:20;
log2g_list = -20:stepSize:20;


%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);


%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);

%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
```

```
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = svmtrain(trainLabel, trainData, param)



      %cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = get_cv_ac(trainLabel, trainData, cmd, 3);



      cvMatrix(i,j) = cv;
      %fprintf( 'value of i is %d\n', i );
      %fprintf( 'value of j is %d\n', j );
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
          accuracy(i)=bestcv;
      end

      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
      %store the corresponding accuracy

   end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is when
G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy);
title('Accuracy vs G-value')
xlabel('G Value')
ylabel('Accuracy')
```

```
%ignoring the plotting below. It was way too complicated
% xlabel('Log_2\gamma');
% figure
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

% ###############################################################################
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
% ###############################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);


cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
     cv = svmtrain(trainLabel, trainData, param);

      %cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = get_cv_ac(trainLabel, trainData, cmd, 3);


      cvMatrix(i,j) = cv;
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
         accuracy2(i)=bestcv;
      end
      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
```

```
    end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);

% % Plot the results
% figure;
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy2);
title('Accuracy2 vs G-value')
xlabel('G Value')
ylabel('Accuracy2')

% ###########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ###########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
    log2c = log2c_list(i);
```

```
    for j = 1:numLog2g
        log2g = log2g_list(j);
        % -v 3 --> 3-fold cross validation
        param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = svmtrain(trainLabel, trainData, param);

       % cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
        %cv = get_cv_ac(trainLabel, trainData, cmd, 3);



        cvMatrix(i,j) = cv;
        if (cv >= bestcv),
            bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
             accuracy3(i)=bestcv;
        end
        % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
    end
end

disp(['CV scale3: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);

% Plot the results
% figure;
% imagesc(cvMatrix); colormap('jet'); colorbar;
% set(gca,'XTick',1:numLog2g)
% set(gca,'XTickLabel',sprintf('%3.1f|',log2g_list))
% xlabel('Log_2\gamma');
% set(gca,'YTick',1:numLog2c)
% set(gca,'YTickLabel',sprintf('%3.1f|',log2c_list))
% ylabel('Log_2c');

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy3);
title('Accuracy3 vs C-value')
xlabel('C Value')
ylabel('Accuracy3')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy2);
title('Accuracy3 vs G-value')
xlabel('G Value')
ylabel('Accuracy3')

%i.e by the time the code reaches here we have found a very good value for
% C and gamma.

% ###################################################################
```

```
% Test phase
% Use the parameters to classify the test set
% ###################################################################
param = ['-q -c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g), ' -b 1'];
bestModel = svmtrain(testLabel, testData, param);
[predict_label, accuracy, prob_values] = svmpredict(testLabel, testData, bestModel, '-b 1'); % test
the training data

% ================================
% ===== Showing the results ======
% ================================

% Assign color for each class
%colorList = generateColorList(2); % This is my own way to assign the color...don't worry about it
 % prism is a type of color map. details about color map can be found
here:http://www.mathworks.com/help/matlab/ref/colormap.html

 %parula,jet plots in blue and white
%colorList = parula(100);
%colorList = jet(100);
%colorList = winter(100);
colorList = prism(100);



% true (ground truth) class
trueClassIndex = zeros(N,1);
trueClassIndex(heart_scale_label==2) = 1;
trueClassIndex(heart_scale_label==4) = 2;
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
resultClassIndex(predict_label==2) = 1;
resultClassIndex(predict_label==4) = 2;
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(heart_scale_inst,'euclidean');

%mdscale was giving error for the breast cancer data saying :Points in the configuration have
co-located.  Try a different
%starting point, or use a different criterion. colocated-so added teh
%criterion=sstress
newCoor = mdscale(distanceMatrix,2);
%newCoor = mdscale(distanceMatrix,2,'criterion', 'sstress') ;

% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
```

```
figure; scatter(x,y,patchSize,colorTrueClassPlot,'filled');
title('whole data set');

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
patchSize = 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'o','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 30;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
title('classification results');

toc
```

IRIS Dataset:

```
%this is  libSvmOverForestData_v21
%in this code i combined the given testing and training data to form the
%file SPECTFlibsvm.combined. This will be used to separated out the 10%90%
%etc test-training combination

clear
clc
close all

% addpath to the libsvm toolbox
%addpath('../libsvm-3.12/matlab');

% addpath to the data
dirData = './dataFromUCI/iris/';
addpath(dirData);



% read the data set
[heart_scale_label, heart_scale_inst] = libsvmread(fullfile(dirData,'iris.combined'));
[N D] = size(heart_scale_inst)
[N D] = size(heart_scale_label)


NoOfRows=N;

%The value for testing and training index is presented in percentages

percentageTraining=50;
%fix removes all the decimal values
stopValue=fix(NoOfRows*percentageTraining/100)

%shuffling the contents of training data
mergedData=[heart_scale_label heart_scale_inst ];
size(mergedData);
trainDataSize=size(mergedData,1);
ix = randperm(trainDataSize);
mergedData = mergedData(ix,:);
trainDataSize=size(mergedData);
heart_scale_label = mergedData(:, 1);
heart_scale_inst=mergedData(:, 2:end);
[N D] = size(heart_scale_inst)
[N D] = size(heart_scale_label)

% Determine the train and test index
trainIndex = zeros(N,1); trainIndex(1:stopValue) = 1;
testIndex = zeros(N,1); testIndex(stopValue:N) = 1;
trainData = heart_scale_inst(trainIndex==1,:);
```

```
trainLabel = heart_scale_label(trainIndex==1,:);
testData = heart_scale_inst(testIndex==1,:);
testLabel = heart_scale_label(testIndex==1,:);

totalData = heart_scale_inst;
size(totalData)
%return;
totalLabel = heart_scale_label;

%[N D] = size(totalData)
labelList = unique(heart_scale_label(:));

%total number of unique classes into which the classification will occur
NClass = length(labelList);


% read the data set
% [dnaTrainLabel, dnaTrainData] = libsvmread(fullfile(dirData,'SPECTFlibsvm.combined'));
% NTrain = size(dnaTrainData,1);
% [dnaTrainLabel, permIndex] = sortrows(dnaTrainLabel);
% dnaTrainData = dnaTrainData(permIndex,:);
% [dnaTestLabel, dnaTestData] = libsvmread(fullfile(dirData,'SPECTFlibsvm.test'));
% NTest = size(dnaTestData,1);
% [dnaTestLabel, permIndex] = sortrows(dnaTestLabel);
% dnaTestData = dnaTestData(permIndex,:);
%
% % combine the data together just to fit my format
% totalData = [dnaTrainData; dnaTestData];
% totalLabel = [dnaTrainLabel; dnaTestLabel];
% %figure;
% figure('Name','Legend','NumberTitle','off')
% subplot(1,2,1); imagesc(totalLabel); title('class label');
% subplot(1,2,2); imagesc(totalData); title('features');
%
% [N D] = size(totalData);
% labelList = unique(totalLabel(:));
% NClass = length(labelList);
%
% % ########################
% % Determine the train and test index
% % ########################
% trainIndex = zeros(N,1); trainIndex(1:NTrain) = 1;
% testIndex = zeros(N,1); testIndex( (NTrain+1):N) = 1;
% trainData = totalData(trainIndex==1,:);
% trainLabel = totalLabel(trainIndex==1,:);
% testData = totalData(testIndex==1,:);
% testLabel = totalLabel(testIndex==1,:);
%
% ########################
% Parameter selection using 3-fold cross validation
```

```
% this part is replaced with the code from libSvmOverBreastCancerData_v18
% where we vary C and Gamma for 3 cycles.
% % ########################
% bestcv = 0;
% for log2c = -1:1:3,
%   for log2g = -4:1:2,
%     cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
%     cv = get_cv_ac(trainLabel, trainData, cmd, 3);
%     if (cv >= bestcv),
%       bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
%     end
%     fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
%   end
% end
stepSize = 1;
log2c_list = -20:stepSize:20;
%size(log2c_list)
%ovrtrain was giving error that gamma values have to be positive. While
%looking at the iteration i found that best gamma comes around 5. So might
%as well change the ranges
log2g_list = -20:stepSize:20;
%size(log2g_list)
%return;
%create a matrix for storing each of the accuracy values
%accuracy = nan()
%N = NaN(n) is an n-by-n matrix of NaN values.

numLog2c = length(log2c_list);
numLog2g = length(log2g_list)


%create a matrix of 41 rows and 1 column to store accuracy values
accuracy = zeros(numLog2c,1);

%cvmatrix stores the accuracy
cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = svmtrain(trainLabel, trainData, param);

      cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = get_cv_ac(trainLabel, trainData, cmd, 3);

      cvMatrix(i,j) = cv;
      %fprintf( 'value of i is %d\n', i );
```

```
        %fprintf( 'value of j is %d\n', j );
       % if (cv >= bestcv),
       %disp(log2g);
       %disp(cv);
         if (cv > bestcv),
        %     disp('************');
            bestcv = cv;
            bestLog2c = log2c ;
            bestLog2g = log2g;
            accuracy(i)=bestcv;
%               if(bestcv>95),
%                   disp(bestLog2g);
%                   disp(bestLog2c);
%
%               end
       end


    end
end

%at the end of each for loop pick the c and gamma values that give the
%maximum accuracy
disp(['CV scale1: best C Value is when C:',num2str(bestLog2c),'and the best g value is when
G:',num2str(bestLog2g),' and the best accuracy is:',num2str(bestcv),'%']);

% Plot the results
%figure;

%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy);
title('Accuracy vs C-value')
xlabel('C Value')
ylabel('Accuracy')

%plot accuracy against gamma value
figure;
plot(log2g_list,accuracy);
title('Accuracy vs G-value')
xlabel('G Value')
ylabel('Accuracy')




% ######################################################################
% cross validation scale 2
% This is the medium scale
% here we run the 40x40 between the values found in the previous cycle
```

```
% ###################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;


numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy2 = zeros(numLog2c,1);


cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = svmtrain(trainLabel, trainData, param);

      cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = get_cv_ac(trainLabel, trainData, cmd, 3);


      cvMatrix(i,j) = cv;
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
         accuracy2(i)=bestcv;
      end
      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
   end
end

disp(['CV scale2: best log2c:',num2str(bestLog2c),' best log2g:',num2str(bestLog2g),'
accuracy:',num2str(bestcv),'%']);


%plot accuracy against c value- c value is stored in log2c_list
figure;
plot(log2c_list,accuracy2);
title('Accuracy2 vs C-value')
xlabel('C Value')
ylabel('Accuracy2')

%plot accuracy against gamma value
figure;
```

```
plot(log2g_list,accuracy2);
title('Accuracy2 vs G-value')
xlabel('G Value')
ylabel('Accuracy2')

% ###########################################################################
% cross validation scale 3
% This is the small scale- this is even smaller than the step2. This will
% be the most accurate
% ###########################################################################
prevStepSize = stepSize;
stepSize = prevStepSize/2;
log2c_list = bestLog2c-prevStepSize:stepSize:bestLog2c+prevStepSize;
log2g_list = bestLog2g-prevStepSize:stepSize:bestLog2g+prevStepSize;

numLog2c = length(log2c_list);
numLog2g = length(log2g_list);

%create a 1 column vector to store the accuracy
accuracy3 = zeros(numLog2c,1);

cvMatrix = zeros(numLog2c,numLog2g);
bestcv = 0;
for i = 1:numLog2c
   log2c = log2c_list(i);
   for j = 1:numLog2g
      log2g = log2g_list(j);
      % -v 3 --> 3-fold cross validation
      %param = ['-q -v 3 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      %cv = svmtrain(trainLabel, trainData, param);
      cmd = ['-q -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
      cv = get_cv_ac(trainLabel, trainData, cmd, 3);


      cvMatrix(i,j) = cv;
      if (cv >= bestcv),
         bestcv = cv; bestLog2c = log2c; bestLog2g = log2g;
          accuracy3(i)=bestcv;
      end
      % fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
   end
end

disp(['CV scale3: best c value is:',num2str(2^bestLog2c),' best g value is:',num2str(2^bestLog2g),'
accuracy:',num2str(bestcv*100),'%']);
disp(['also CV scale3: best value of bestLog2c  is:',num2str(bestLog2c),' best value of
bestLog2g:',num2str(bestLog2g),' accuracy:',num2str(bestcv*100),'%']);

% ########################
% Train the SVM in one-vs-rest (OVR) mode
```

```
% ########################
%for some reason ovrtrain is not accepting negative values of gamma. So
%converting it to positive values. This is Stupidand wrong, but no other
%go.
%update: figured out why ovrtrain doesnt accept negative values. Gamma
%value is 2^bestLog2g, not bestLog2g which was -10
bestParam = ['-c ', num2str(2^bestLog2c), ' -g ', num2str(2^bestLog2g)];

model = ovrtrain(trainLabel, trainData, bestParam);
% ########################
% Classify samples using OVR model
% ########################
[predict_label, accuracy, prob_values] = ovrpredict(testLabel, testData, model);
fprintf('Accuracy = %g%%\n', accuracy * 100);


% ================================
% ===== Showing the results ======
% ================================

% Assign color for each class
%colorList = generateColorList(NClass); % This is my own way to assign the color...don't worry
about it
 %colorList = spring(100);
 colorList = prism(100);

% true (ground truth) class
trueClassIndex = zeros(N,1);
for i = 1:NClass
    trueClassIndex(totalLabel==labelList(i)) = i;
end
colorTrueClass = colorList(trueClassIndex,:);
% result Class
resultClassIndex = zeros(length(predict_label),1);
for i = 1:NClass
    resultClassIndex(predict_label==labelList(i)) = i;
end
colorResultClass = colorList(resultClassIndex,:);

% Reduce the dimension from 13D to 2D
distanceMatrix = pdist(totalData,'euclidean');
% newCoor = mdscale(distanceMatrix,2); % take longer time, but more beautiful
newCoor = cmdscale(distanceMatrix); %
% Plot the whole data set
x = newCoor(:,1);
y = newCoor(:,2);
patchSize = 30; %max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass;
figure;
%scatter(x,y,patchSize,colorTrueClassPlot,'x');
```

```
scatter(x,y,patchSize,'blue','o');
title('whole data set');
xlabel('IRIS Feature Scaled1')
ylabel('IRIS Feature Scaled2')

% Plot the test data
x = newCoor(testIndex==1,1);
y = newCoor(testIndex==1,2);
patchSize = 60;% 80*max(prob_values,[],2);
colorTrueClassPlot = colorTrueClass(testIndex==1,:);
figure; hold on;
scatter(x,y,2*patchSize,colorTrueClassPlot,'o','filled');
scatter(x,y,patchSize,colorResultClass,'o','filled');
% Plot the training set
x = newCoor(trainIndex==1,1);
y = newCoor(trainIndex==1,2);
patchSize = 60;
colorTrueClassPlot = colorTrueClass(trainIndex==1,:);
scatter(x,y,patchSize,colorTrueClassPlot,'o');
title('classification results');
xlabel('IRIS Feature Scaled1')
ylabel('IRIS Feature Scaled2')
```