Lab 2

Thursday, July 01, 2010
4:11 PM

SETUP
Add /usr/ccrma/courses/mir2010/Toolboxes/stprtool with subfolders to your matlab path.

If you want to learn more about the STPRTool, a pattern recognition toolbox (not MIR-specific), see http://cmp.felk.cvut.cz/cmp/software/stprtool/.

ADABOOST WITH STPRTOOL
Run "help adaboost" for information about how the adaboost command works.
Note that you can run a model trained using adaboost using the "adaclass" command (similar to how we used knnfwd in Lab 1).
Run "help adaclass" for an overview.

BOOSTING AND THE DECISION BOUNDARY
Now we'll use some of the adaboost demo data to get a feel for how boosting evolves the decision boundary. Load the dataset by:

```
  data = load('riply_trn');
```

Next set some options that we'll use to train the classifier:

```
  options.learner = 'weaklearner'; % Sets adaboost to boost on a decision stump weak learner
  options.max_rules = 1;             % Sets the number of boosting rounds, or equivalently the number of stumps used, to just 1
  options.verb = 1;                  % verbose output
```

Train it and plot the data with the decision boundary:

```
  model = adaboost(data,options);                 % creates the model
  figure; ppatterns(data); pboundary(model); %stprtool-specific functions for plotting data and decision boundary in 2D
```

% NOTE: PBOUNDARY DOES NOT WORK WITH FEATURE DATA > 2 dimensions
% NOTE 2: If you put your kick/snare feature data into AdaBoost, it will converge after only 1 ROUND, since this is such an easy challenge.

Now change the number of max_rules in the options to 2, 3, 10, 100, and 1000. Notice how the decision boundary changes over time. Think about how it compares to the decision boundary that would be produced by a kNN with k=1.

BOOSTING ON MUSICAL DATA
Next, compare the performance of kNN on clarinet vs. sax classification to the performance of AdaBoost on the same data.
This example dataset is posted at /usr/ccrma/courses/mir2010/audio/toy data/.

In order to train AdaBoost, notice that you'll need to slightly change the formatting of your training dataset.

If variable myFeatures contains an N x F matrix of training data, like in Lab 1, where N is the number of datapoints and F is the number of features, assign it to the X field of a new data variable as follows:
data.X = myFeatures';

Then if variable myLabels contains an N x C matrix of 1-of-N labels, where N in the number of datapoints and C is the number of classes, assign the Y field of your data variable as follows:

```
for i=1:N
  for j=1:C
    if (myLabels(i,j) == 1)
      Y(i) = j;
    end
    end
end
data.Y=Y;
```

COMPARING ADABOOST TO KNN

Train an AdaBoost classifier with 100 rounds on the training dataset and evaluate testing accuracy on the testing dataset. Compare this to your kNN in lab 1.

BONUS

Try a different 2-class problem with new data. (Such as the data contained in the **audio** folder!)

NEW FEATURES

A function to calculate Spectral Flatness Measure and Temporal Centroidgm is provided for your enjoyment:

```
% x is your frame of audio
[ sfm ] = spectralFlatnessMeasure(x,fs,fftSize)
 [tc] = temporalCentroid(x);
```

CROSS VALIDATION

You'll need some of this code and information to calculate your accuracy rate on your classifiers.

EXAMPLE

Let's say we have 10-fold cross validation...

a. Divide test set into 10 random subsets.
b. 1 test set is tested using the classifier trained on the remaining 9.
c. We then do test/train on all of the other sets and average the percentages.

To achieve the first step (divide our training set into k disjoint subsets), use the function **crossvalind.m** (posted in the **Utilities**)

> INDICES = CROSSVALIND('Kfold',N,K) returns randomly generated indices for a K-fold cross-validation of N observations. INDICES contains equal (or approximately equal) proportions of the integers 1 through K that define a partition of the N observations into K disjoint subsets.
>
> You can type **help crossvalind** to look at all the other options.

This code is also posted as a template in /usr/ccrma/courses/mir2010/Toolboxes/ **crossValidation.m**

```
%
%  This code is provided as a template for your cross-validation
%  computation.  Replace the variables "features", "labels" with your own
%  data.
%  As well, you can replace the code in the "BUILD" and "EVALUATE" sections
%  to be useful with other types of Classifiers.
%

    %% CROSS VALIDATION
    numFolds = 10;                % how many cross-validation folds do you want - (default=10)
    numInstances = size(features,1);    % this is the total number of instances in our training set
    numFeatures = size(features,2);     % this is the total number of instances in our training set
    indices = crossvalind('Kfold',numInstances,numFolds)   % divide test set into 10 random subsets

    clear errors
        for i = 1:10
             % SEGMENT DATA INTO FOLDS
            disp(['fold: ' num2str(i)])
            test = (indices == i) ;    % which points are in the test set
            train = ~test;       % all points that are NOT in the test set

            % SCALE
            [trainingFeatures,mf,sf]=scale(features(train,:));

            % BUILD NEW MODEL - ADD YOUR MODEL BUILDING CODE HERE...
             model = knn(numFeatures,2,3,trainingFeatures,labels(train,:));
```

```
                    % RESCALE TEST DATA TO TRAINING SCALE SPACE
                    [testingFeatures]=rescale(features(test,:),mf,sf);

                    % EVALUATE WITH TEST DATA - ADD YOUR MODEL EVALUATION CODE HERE
                    [voting,model_output] = knnfwd(model ,testingFeatures);

                   % CONVERT labels(test,:) LABELS TO SAME FORMAT TO COMPUTE ERROR
                    labels_test = zeros(size(model_output,1),1); % create array of 0s
                    labels_test(find(labels(test,1)==1))=1;  % convert column 1 to class 1
                    labels_test(find(labels(test,2)==1))=2;  % convert column 2 to class 2

                    %  COUNT ERRORS
                    errors(i) = mean ( model_output ~= labels_test  )

                  end
        disp(['cross validation error: '  num2str(mean(errors))])
        disp(['cross validation accuracy: ' num2str(1-mean(errors))])
```

BONUS (ONLY IF YOU HAVE EXTRA TIME…)

1. For a slick experience, check out the commands uigetdir and uigetfile -- these allow your matlab scripts to present a GUI browser to query for file locations.

2. Create a new classifiers, using other audio samples.