Lab 7 - HMM / Cross validation

Tuesday, July 29, 2008
10:01 PM

PURPOSE

By the end of this lab, you should have a skeletal outline of where your project is going to go, what data you will use, what features, classifiers / techniques you will use, and what your metric of success / GOAL is.

PROJECTS OUTLINE

Still need Ideas?

Look at large audio collections and see what you can classify or categorize.
Examples for possible classifiers:
http://en.wikipedia.org/wiki/List_of_Music_Genome_Project_attributes_by_type
Other samples projects:
Instrument ID
Drum Loop searches
Transcription by classifier
Speaker ID

What you should do for the project:

Today

Explain your ideas and implementation, solicit key ideas and assistance from others.
Define specific, concrete project with measurable goals.
Define evaluation metrics. (error rates) (You might need to create a metric (distance from a prototype example)
Brainstorm approaches
Research previous approaches (ask Jay for papers, if you need some)
Specifically define your approach (features to try, classifiers to try)

Thursday + Friday

Implement!
Play with it AND Evaluate (domain knowledge adjustments to it)
Make improvements, adjustments, re-evaluate.
Try different features / classifiers / data sets.
Meticulously document your changes and how they effect the final results.

On Friday afternoon:

Prepare demonstrations and sound examples for class.
State how you did what you did.
How can you improve?

CROSS VALIDATION

In yesterday's lecture, we covered k-fold cross-validation.
You'll need some of this code and information to calculate your accuracy rate on your classifiers -- that is, if you chose to do this as your project.

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

1. Divide test set into 10 random subsets.
2. 1 test set is tested using the classifier trained on the remaining 9.
3. 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 Lab 7 folder)

```
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.

Here is an outline of how to perform cross-validation on a classifier:

```
% cross_validation
k = 10;                          % how many folds do you want?
N = size(features,1) ;           % this is the total number of observations or rows that we have
indices = crossvalind('Kfold',N,k)  % divide test set into 10 random subsets
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
     model = knn(numFeatures,1,1,trainingFeatures,labels(train,:));

    % EVALUATE WITH TEST DATA
    model_output = knnfwd(model ,features(test,:))

    %  COUNT ERRORS
    errors(i) = mean ( model_output ~= labels(test,:) )

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

ASIDE: HMM COMMANDS FOR FUTURE USE

NOTE: Due to the difficult in creating and annotating, labeling training data for sequence purposes, no training sets are available for this lab. The commands are provided for your future reference and later experimentation.

A good review of HMMs from Matlab's perspective can be found at :
http://www.mathworks.com/access/helpdesk/help/toolbox/stats/index.html?/access/helpdesk/help/toolbox/stats/f8368.html&http://www.google.com/search?q=As+an+example%2C+consider+a+Markov+model+with+two+states+and+six+possible+emissions.+The+model+uses%3A&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a

Statistics Toolbox includes five functions related to hidden Markov models:

hmmgenerate —Generates a sequence of states and emissions from a Markov
[seq,states] = hmmgenerate(len,TRANS,EMIS) takes a known Markov model, specified by transition probability matrix TRANS and emission probability matrix EMIS, and uses it to generate

* A random sequence seq of emission symbols
* A random sequence states of states

The length of both seq and states is len. TRANS(i,j) is the probability of transition from state i to state j. EMIS(k,l) is the probability that symbol l is emitted from state k.

modelhmmestimate —Calculates maximum likelihood estimates of transition and emission probabilities from a sequence of emissions and a known sequence of states

hmmestimate
hmmestimate requires that you know the sequence of states states that the model went through to generate seq.The following takes the emission and state sequences and returns estimates of the transition and emission matrices:[TRANS_EST, EMIS_EST] = hmmestimate(seq, states)

hmmtrain — Calculates maximum likelihood estimates of transition and emission probabilities from a sequence of emissions

"If you do not know the sequence of states states, but you have initial guesses for TRANS and EMIS, you can still estimate TRANS and EMIS using hmmtrain. Suppose you have the following initial guesses for TRANS and EMIS.

hmmtrain uses an iterative algorithm that alters the matrices TRANS_GUESS and EMIS_GUESS so that at each step the adjusted matrices are more likely to generate the observed sequence, seq. The algorithm halts when the matrices in two successive iterations are within a small tolerance of each other."

hmmviterbi — Calculates the most probable state path for a given hidden Markov model

STATES = hmmviterbi(seq,TRANS,EMIS) given a sequence, seq, calculates the most likely path through the hidden Markov model specified by transition probability matrix, TRANS, and emission probability matrix EMIS.

hmmdecode — Calculates the posterior state probabilities of a sequence of emissions

"The posterior state probabilities of an emission sequence seq arethe conditional probabilities that the model is in a particular state when it generates a symbol in seq, given that seq is emitted. You compute the posterior state probabilities with: "