EEL 6586: HW#6

Due Friday, April 4, 2003 in class. Late homework loses $e^{\#\ of\ days\ late} - 1$ percentage points. See the current late penalty at http://www.cnel.ufl.edu/hybrid/harris/latepoints.html

HMM-based Digit Recognition

In this part, you will investigate the implementation issues of a Hidden Markov Model (HMM) by using the HMM to recognize the English digits "zero" through "nine". You may implement your own HMM, but we highly recommend you download the H2M Matlab Toolbox by Olivier Capp http://www-sig.enst.fr/~cappe/h2m/h2m.html. Read the contents of this webpage to become familiar with H2M; its implementation is as straightforward as using any other Matlab Toolbox functions. Example files such as `ex_basic.m` and `ex_sprec.m` are handy.

[1] Draw the state diagram for a left-right HMM typically used for word recognition. What is a reasonable number of states to represent, say, the digits considered in this problem? Explain. Also include the state transition matrix A and initial state probability vector π. Include reasonable values for the A matrix and π vector. How did you find these values? (Hint: consider "hard" segmentation as used in `hmm_mint.m` in H2M).

2 In this problem, you will perform "clean" speech recognition on a digits database using an HMM. The database of raw speech available in http://www.cnel.ufl.edu/hybrid/courses/EEL6586/hw6.zip is separated into TEST and TRAIN sets. See the README.TXT file in hw6.zip for details on the database format. For your HMM, use the following parameters: States=6, `n_Iterations`=10, output observation pdfs are single multivariate Gaussian with diagonal covariance matrices. You can use the speech features you used in HW3, though we recommend mfcc features (Malcolm Slaney's mfcc.m code in Auditory Toolbox available at http://rvl4.ecn.purdue.edu/~malcolm/interval/1998-010. For parametric classifiers, it is typical to report two recognition scores: 1) recognition on the TRAIN data, and 2) recognition on TEST data. Since your HMM model parameters were estimated from the TRAIN data, recognition using the TRAIN data should be higher than recognition on unknown TEST data. The difference in these two scores is an indicator of how "generalized" your classifier is. Hand in both recognition results (Note: should be greater than 90% for this database) along with both confusion matrices (same as in HW3) and a description of the features you used.

3 From the results of part [2] above, you should find only a few (if any) incorrectly-classified utterances. Find the utterances misclassified when using the TEST data and report the utterance label (eg: `ti\_00F8ST01`) as well as the class (digit) each one classified as. Comment on possible explanations as to why these utterances were misclassified.

4 Repeat [2] above, but instead of using "clean" TEST utterances, use TEST utterances which have added white Gaussian noise (AWGN) at a global SNR of 20 dB. "Global" SNR is defined over an entire utterance, as opposed to "Local" SNR, where each frame of speech is set to the same SNR – high-energy frames have more noise added than low-energy frames (unrealistic since noise level is now speech-dependent which violates the assumption that the noise and utterance are independent). Report your recognition results on the noisy TEST data and clean TRAIN data as well as your confusion matrices. Below is code you can use to add AWGN to each utterance:

```
% Create random sequence of normal distribution (zero mean, unity variance):
noise = randn(size(x)); % x is the "clean" time-domain utterance (whole word)

% Find energy of each utterance and noise:
energyX = sum(x.^2); energyNoise = sum(noise.^2);

% Find amplitude for noise:
noiseAmp = sqrt(energyX/energyNoise*10^(-SNR/10)); % SNR is in dB

% Add noise to utterance:
x = x + noiseAmp*noise;
```

5 Repeat [4] above, but now you are free to modify your algorithm to improve recognition performance. Consider varying the number states in your HMM, the number of iterations used to train your HMM, the order in which you present your train data to the EM algorithm (see extra credit below). Report recognition results on the TEST and the TRAIN data as well as the confusion matrices. The TA reports results around 40% correct as a ballpark figure.

Extra Credit On unlabeled data found at http://www.cnel.ufl.edu/hybrid/courses/EEL6586/hw6EC.zip, you will classify the unlabeled data using your HMM from part [5]. The only thing you know about the unlabeled test data is that it is "clean" speech corrupted with AWGN at 20 dB SNR. Since you don't know the utterance of the data, you cannot tabulate a confusion matrix. So instead, you will report your results as a column vector. The elements of your results vector correspond to the rows of the character matrix testNames containing the names of the unlabeled TEST utterances. Each element of your results vector is the class 0-9 output by your HMM in float format (not character format!). WHAT TO TURN IN: to get extra credit, store your results vector in a Matlab variable called resultsEC, then save this variable to a .mat file with the following naming convention – first initial followed by last name (eg: jharris.mat, lgu.mat). Mail your .mat file to the TA (lygu@cnel.ufl.edu). Since grading will be done by a Windows PC, using the proper naming convention is CRITICAL! The students with the highest accuracy on the unlabeled set will receive bonus points.

Hints: To improve your recognition accuracy for noisy speech, consider including one of more of the following techniques in your code:

- endpoint detection to reduce silence regions at beginning and end of each utterance.
- noise-reduction techniques (cepstral mean subtraction or spectral subtraction are easy to implement).
- create separate sets of HMMs for male and female speakers. When testing your HMM on data (where you won't know if the speaker is male or female), you'll need to find a way to choose which models are correct (unless both report the same class).
- consider adding noise to your TRAIN data (why would this help?).
- are you using the Viterbi algorithm or the Forward algorithm to find the highest log-likelihood during evaluation of your HMM? Consider using the other.
- single multivariate gaussians vs. gaussian mixtures to represent the output observation pdfs.
- delta cepstrum and delta-delta cepstrum features.
- replacing "hard" segmentation used in `hmm_mint.m` to initialize your HMM parameters w/ some other heuristic.
- noise-reduction techniques like simultaneous and temporal auditory masking (used in mp3 coding). See http://www.dcs.shef.ac.uk/~jeremy/litrev.htm for a review of speech recognition in noisy environments.
- using a Wiener filter to reduce noise, given that the noise is AWGN at 20 dB SNR.

As usual, attach all of your code to the end of the assignment.