EEL 6586: HW#4

Parts A and B due Friday, March 10, 2006 in class. Part C due Monday, March 27. Late homework loses usual exponential penalty

PART A: Textbook problems (Use Matlab only to optionally check your work)

- A1 a) Compute the complex cepstrum of $H(z) = (1 2z^{-1})/(1 + .25z^{-2})$ b) Compute the real cepstrum of $H(z) = (1 - 2z^{-1})/(1 + .25z^{-2})$
- A2 Let $x_1(n)$ and $x_2(n)$ denote two sequences and $\hat{x}_1(n)$ and $\hat{x}_2(n)$ their corresponding complex cepstra. If $x_1(n) * x_2(n) = \delta(n)$ determine the relationship between $\hat{x}_1(n)$ and $\hat{x}_2(n)$.
- A3 Suppose the complex cepstrum of y(n) is $\hat{y}(n) = \hat{s}(n) + 2\delta(n)$. Determine y(n) in terms of s(n).
- A4 Euclidean distance in complex cepstral space can be related to a RMS log spectral distance measure. Assuming that

$$\log S(\omega) = \sum_{n=-\infty}^{n=+\infty} c_n e^{-jn\omega}$$

where $S(\omega)$ is the power spectrum (magnitude-squared Fourier transform), prove the following:

$$\sum_{n=-\infty}^{n=+\infty} (c_n - c'_n)^2 = \frac{1}{2\pi} \int |\log(S(\omega)) - \log(S'(\omega))|^2 d\omega$$

where $S(\omega)$ and $S'(\omega)$ are the power spectra for two different signals.

A5 (Extra Credit) Assuming that

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = \frac{G}{1 - \sum_{k=1}^{p} a(k) z^{-k}}$$

Prove that the complex cepstrum $\hat{h}(n)$ can be derived from the linear prediction coefficients a(k) using the following relation:

$$\hat{h}(n) = a(n) + \sum_{k=1}^{n-1} (k/n)\hat{h}(k)a(n-k)$$

for $n \ge 1$.

J.G. Harris

1

PART B: HMM Analysis

An HMM-based phoneme recognizer has been trained to recognize two phonemes e and o. These phonemes are modelled by a 2-state discrete-output HMM, with 3 output symbols in the output distribution (thus, each feature vector is quantized to one of three possible symbols). The three output symbols will be called x, y and z. The states will be referred to as state 1 and state 2. The models are left-to-right and must start in the initial state (state 1) and end in the final state (state 2). The two trained HMM models have the same transition matrix:

$$A_e = A_o = A = \{a_{ij}\} = \begin{bmatrix} .9 & .1 \\ 0 & 1 \end{bmatrix}$$

The two output probability matrices for phoneme 1 and 2 are given by:

$$B_e = \{b_i(k)\} = \begin{bmatrix} .8 & .7 \\ .1 & .1 \\ .1 & .2 \end{bmatrix} \qquad B_o = \{b_i(k)\} = \begin{bmatrix} .2 & .1 \\ .4 & .3 \\ .4 & .6 \end{bmatrix}$$

The columns of the B matrix denote the probabilities of the three symbols (x, y and z) for states 1 and 2. A sequence of observations is created as x-y-z-y.

- B1 How many valid state sequences are possible for a state sequence of 4 symbols (T=4)? List the sequences.
- B2 What is the probability that the e model produced the observed sequence? Do the same for the o model. This is evaluation but you do not need to simulate the forward algorithm to get the right answer.
- B3 For each phoneme model, what is the optimal state sequence? This is decoding but you do not need to simulate the Viterbi algorithm.
- B4 How will the system classify this phoneme, as an e or an o? Discuss.

PART C: 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 or use any available toolbox but we highly recommend you download the HMM toolbox written by Kevin Murphy http://www.cs.ubc.ca/murphyk/Software/HMM/hmm.html http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html.

C1 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?

- C2 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/hw4.zip is separated into TEST and TRAIN sets. See the README.TXT file in hw4.zip for details on the database format. You can use any speech features you like, 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 matrices and a description of the features you used.
- C3 From the results of part [C2] 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.
- C4 Repeat [C2] 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;
```

C5 Repeat [C4] 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. Results around 40% correct are a ballpark figure.

Extra Credit On unlabeled data found at

http://www.cnel.ufl.edu/hybrid/courses/EEL6586/hw4EC.zip, you will classify the unlabeled data using your HMM from part [C5]. 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 savya2u@ufl.edu (Savyasachi Singh). 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 (think about why would this help?).
- are you using the Viterbi algorithm or the Forward algorithm to find the highest loglikelihood 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.

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