EEL 6586: HW#5

Due Monday, March 24 in class, standard late penalties apply.

Remember that we will have an open-book exam on Tuesday, March 18 from 8:20pm to 10:10pm in FLG 220. HMMs will not be on the exam.

PART A: Short Answer, use a few sentences to answer each question.

A1 DTW and HMM speech recognition systems were developed for a vocabulary of 1000 words. In general terms, compare and contrast the systems in terms of the time required for training and the time required for evaluating an unknown word.

A2 Which requires more computation: the forward algorithm or the Viterbi algorithm, or are they about the same? Explain.

A3 How many different state sequences exist given an HMM with 3 states (N=3) and sequences that are 5 long (T=5)? Assume that the π vector is such that you must start in sequence 1 and the only nonzero transistion probabilities are moving to the next state or remaining in the current state. Each sequence needs to end in the final state.

A4 Estimate how many utterances are needed to train a single word model using the following crude assumptions. There are 6 possible observation symbols (M=6). Only consider the probability distribution of the first state where all the symbol probabilities are equal, i.e. all equal to 1/6. Finally assume that these values need to be approximately to within 1% of their true value. Make whatever other assumptions you feel you need.

PART B: HMM Analysis

An HMM-based phoneme recognizer has been trained to recognize two phonemes e and o. Each phoneme is modelled with a 2-state discrete-output HMM, with 2 output symbols in the output distribution (thus, each feature vector is quantized to one of two possible symbols). The two output symbols will be called y and n. The states will be referred to as state 1 and state 2.

The two trained HMM models have the same transition matrix:

$$A_e = A_o = A = \{a_{ij}\} = \begin{bmatrix} .5 & .5 \\ .5 & .5 \end{bmatrix}$$

and the same initial state probability vector:

$$\pi = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The two output probability matrices for phoneme e and o are given by:

$$B_e = \{b_i(k)\} = \begin{bmatrix} .5 & .5 \\ .5 & .5 \end{bmatrix} \qquad B_o = \{b_i(k)\} = \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}$$

The columns of the B matrix denote the probabilities of the two symbols y and n for states 1 and 2. To be clear, state 1 of phoneme o will generate a y with probability .8 and an n with probability 0.2. Note that these HMMs are not very practical.

B1 How many valid state sequences are possible for a state sequence of 3 symbols (T=3)? List the state sequences.

B2 A sequence of observations is created as n-y-y. Determine the most likely state sequence for each phoneme and its corresponding likelihood.

B3 Will the system classify this phoneme as an e or an o assuming that the likelihood of the optimal state sequence is used in the classification?

B4 How would the classification of the phonemes in question [B3] change if we used evaluation instead of decoding to classify? This means we sum the likelihood of all the possible state sequences to make the decision. 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.

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/hw5.zip is separated into TEST and TRAIN sets. See the README.TXT file in hw5.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 results (Note: should be greater than 90% for this database) along with both confusion 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/hw5EC.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, janderson.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 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.

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