Massachusetts Institute of Technology
Department of Electrical Engineering & Computer Science

6.345 Automatic Speech Recognition
Spring, 2003

Issued: 03/07/03
Due: 03/19/03

Assignment 5
Introduction to Hidden Markov Models

Introduction

Hidden Markov models (HMMs) are a class of statistical models useful for analyzing a discrete-time series of observations such as a stream of acoustic tokens extracted from a speech signal. HMMs are very well-suited for speech recognition due to their inherent ability to represent variable-duration acoustic events and the existence of efficient algorithms to automatically compute model parameters from training data.

Software

In this lab, you will use a program called hmm_tool. This program operates on discrete observation HMMs. It has the capability to generate random observation sequences from an HMM, perform the Viterbi search algorithm on an observation sequence for a particular HMM, and estimate an HMM's parameters using the Baum-Welch algorithm. For this lab you will utilize hmm_tool on artificial data in order to familiarize yourself with the properties of HMMs and their associated algorithms. The hmm_tool program operates directly from the UNIX command line. HMMs and observation sequences are stored in ASCII files and passed to hmm_tool through command line options. To list the complete set of command line options available with hmm_tool type:

```
% hmm_tool -help
```

Getting Started

To get started for this lab, type the following command at the UNIX prompt:

```
% start_lab5.cmd
```

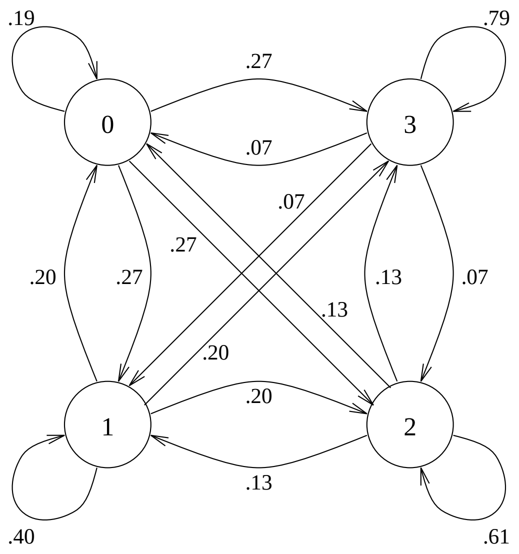
This will make a local copy (in your home directory) of the set of files needed for this lab. These files contain the hidden Markov models and observation sequences used during the lab. The usage for hmm_tool will also be displayed.

Part I: Symmetric HMMs

We will start with a class of HMMs called "symmetric" HMMs. The structure of symmetric HMMs is sufficiently simple to permit direct estimation of model parameters from observed data instead of iterative estimation using the Baum-Welch reestimation algorithm. An N-state symmetric HMM has the following properties:

1. The size of the output alphabet is equal to the number of states N.
2. Every state can output only one symbol and no two states can output the same symbol. More precisely, the B matrix is an N by N identity matrix. This basically means the model is not "hidden" anymore.
3. All transitions leaving a node are equally likely (except self-transitions); i.e., for any given state i, $a_{ij} = a_{ik} \forall j, k \neq i$ where $0 \leq i, j, k < N$.
4. An N-state symmetric HMM is completely determined by its N self-loop transition probabilities $a_{00}, a_{11}, \ldots a_{(N-1)(N-1)}$.

Here is an example of a 4-state symmetric HMM:



T1: Display the parameters of the two symmetric HMMs we will utilize and examine their properties with the commands:

```
% hmm_tool -hmm_in symmetric1.hmm -print_models
% hmm_tool -hmm_in symmetric2.hmm -print_models
```

Use `symmetric1.hmm` (which is actually the same as the model shown above) to generate a single observation sequence 100 symbols long with the following command:

```
% hmm_tool -hmm_in symmetric1.hmm -generate 1 100 \
           -print_obs -obs_out seq1.obs
```

Note that the observation sequence is written to the file `seq1.obs`. Compute and record the number of times all 16 possible output symbol bigrams have occurred in the observation sequence using the command:

```
% hmm_tool -hmm_in symmetric1.hmm -obs_in seq1.obs \
           -count_bigrams
```

Generate another observation sequence 10,000 symbols long and compute and record the bigram counts again (do not use the `-print_obs` option this time to avoid printing 10,000 observations to the screen).

Q1: Derive a formula which estimates the self-loop transition probabilities directly from the bigram counts which you computed. Use this formula together with the observed counts to estimate the self-loop transition probabilities from both observation sequences. Do the counts from the longer observation sequence yield more accurate estimates as expected?

T2: The second HMM-model, `symmetric2.hmm`, differs from the first only in the transition probability a_{33} which equals 1.0. Generate one observation sequence 5,000 symbols long using `symmetric2.hmm` and compute the bigram counts.

Q2: What happens when you try to estimate self-loop transition probabilities for this HMM and why? Would observing longer sequences help? What can you do to estimate the self-loop transitions for this model?

Part II: Training via the Baum-Welch Algorithm

In this part of the lab, we will examine some issues related to training HMMs using the Baum-Welch or Forward-Backward algorithm. We will be concerned with a 5-state left-to-right hidden Markov model defined by the following transition and output probability matrices:

$$A = \begin{bmatrix} .8 & .1 & .1 & 0 & 0 \\ 0 & .9 & .1 & 0 & 0 \\ 0 & 0 & .8 & .1 & .1 \\ 0 & 0 & 0 & .9 & .1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} .7 & .3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .8 & .2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .2 & .8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & .7 \end{bmatrix}$$

Note that A is upper-triangular, which makes the model left-to-right (backward transitions are prohibited). We will generate some observations using this model and then use the Baum-Welch training algorithm to train a new HMM model and examine how closely the estimates match those of the actual model. We will explore the effect of different amounts of training data and the effect of different initial A and B matrices.

T3: Examine the model `5-state-lr.hmm` to see that it matches the A and B matrices above with the command:

```
% hmm_tool -hmm_in 5-state-lr.hmm -print_models
```

Generate 10 observation sequences, each 50 symbols long, using this model:

```
% hmm_tool -hmm_in 5-state-lr.hmm -generate 10 50 \
           -print_obs -obs_out seq2.obs
```

Examine the HMM `5-state-init1.hmm`, which has the same zero elements in its A and B matrices (i.e., the same forbidden transitions and outputs) as the actual model but the non-zero elements are uniformly initialized. We will use `5-state-init1.hmm` as the initial model to the Baum-Welch reestimation algorithm for training an HMM from the observed data. To perform Baum-Welch training using the observation sequence you just generated, type the command:

```
% hmm_tool -hmm_in 5-state-init1.hmm -obs_in seq2.obs \
           -bw 20 -print_models
```

Observe the log-likelihood score of the training data after each iteration and note how quickly the Baum-Welch algorithm converges.

Next, generate 100 observation sequences of length 50 and again use `5-state-init1.hmm` as the initial model for reestimating the HMM parameters from the new set of 100 observation sequences. Compare the parameters of the models estimated from the 10 sequence data and the 100 sequence data to the parameters of the actual model which generated the sequences.

Q3: What is the ratio of the number of data points in the training data set to the number of free parameters being estimated in the two training runs that you performed above? Is there a noticeable difference in the quality of estimates of the A and B matrices between the two training runs?

T4: We will now estimate the HMM parameters using more general initial HMMs. Examine the three initial HMM models:

- `5-state-init2.hmm`
- `5-state-init3.hmm`
- `5-state-init4.hmm`

Notice that `5-state-init2.hmm` is a left-to-right model while `5-state-init3.hmm` and `5-state-init4.hmm` are not. Use the Baum-Welch algorithm to estimate the HMM parameters using the 100 sequence observation data generated in **T3** starting with the initial model `5-state-init2.hmm`. Record the log-likelihood values of the training data after each iteration. When training is finished, compare the parameter values of the estimated HMM to those of the actual model.

Repeat this task for `5-state-init3.hmm` and `5-state-init4.hmm`.

Q4: On the same set of axes, plot the log-likelihood as a function of the number of iterations observed in training `5-state-init2.hmm`, `5-state-init3.hmm`, and `5-state-init4.hmm`. What can you say about the relative difficulty of training more general models vs. more constrained ones? What can you conclude about the importance of initializing the A and B matrices before beginning training?

Q5: What property of the Baum-Welch algorithm would account for major discrepancies that you may have observed between the estimated and true parameters?

Part III: Trellis Representation and Viterbi Search

Both the Forward-Backward algorithm used in Baum-Welch reestimation and the Viterbi decoding algorithm rely on the trellis (or graph) representation of hidden Markov models. This section of the lab deals with the following 2-state HMM:

$$A = \begin{bmatrix} .4 & .6 \\ 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} .8 & .1 & .1 \\ .2 & .2 & .6 \end{bmatrix}$$

T5: Generate and write down a sequence of 5 observations from the model shown above with the command:

```
% hmm_tool -hmm_in 2-state-lr.hmm -generate 1 5 \
           -print_models -print_obs
```

Q6: Use the attached answer sheet to draw a trellis for the sequence that you observed. Compute all branch probabilities and determine the most likely path through the trellis.

Q7: Roughly estimate the length of an observation sequence which would cause underflow in a typical computer if logarithms are not used; i.e., how long an observation sequence will have a probability on the order of 10^{-100}?

Part IV: Recognition with HMMs

In a typical HMM-based word recognition system, the acoustics of each word in the vocabulary are represented by a separate hidden Markov model. Recognition can be accomplished by using the Viterbi algorithm to compute the most likely hidden state sequence in each of the candidate HMMs and then select the word corresponding to the HMM with the most likely state sequence.

T6: Each of these files:

- `testA.obs`
- `testB.obs`
- `testC.obs`

contains one sequence of 50 observations generated from one of the three HMMs:

- `recognition1.hmm`
- `recognition2.hmm`
- `recognition3.hmm`

The HMMs are 5-state left-to-right models which differ only in their B matrices. Use the Viterbi search algorithm to compute the log-probability score of the most likely hidden state sequence through each of the models for each of the observation sequences (make sure you compute all 9 probabilities). For example, to compute the score between `testA.obs` and `recognition1.hmm`, we can say:

```
% hmm_tool -hmm_in recognition1.hmm \
           -obs_in testA.obs -viterbi
```

Q8: Make a 3×3 table showing the computed Viterbi scores for each of the candidate models and each of the observation sequences. Which model was most likely to generate each test sequence based on these scores? What other method of scoring can be used to perform this recognition task?

Part V: Continuous Density HMMs

In class, we solved the three main HMM problems for the case when the observations were characterized as discrete symbols chosen from a finite alphabet, and therefore we could use a discrete probability density within each state of an HMM model. A more common class of HMMs is continuous density HMMs where the observations are continuous. The most general representation of the probability density function (pdf) for which a reestimation procedure has been formulated is the mixture of Gaussians of the form:

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} N(o_t, \mu_{jk}, U_{jk})$$

where o_t is the observation vector being modeled at time t, c_{jk} is the mixture coefficient for the kth mixture in state j and N is the Gaussian density with mean vector μ_{jk} and covariance matrix U_{jk} for the kth mixture component in state j.

In this problem, we will look at a simple approach for deriving the reestimation formula for the mean for the simplified case of a single Gaussian observation pdf, that is, when the number of mixtures $M = 1$:

$$b_j(o_t) = N(o_t, \mu_j, U_j)$$

Q9: In order to find the reestimation formula for μ_j, we quantize the continuous random variable using a set of bins derived from the continuous distribution. Let the values o_t be quantized so that each $o_t \approx o_k$ for some value k. Notice here that the o_k is now equivalent to an observation symbol from the discrete case. Express μ_j as a function of o_k and $\bar{b}_j(k)$ where $\bar{b}_j(k)$ is the reestimation formula for the observation of the kth symbol in the jth state for the discrete case and is given by:

$$\bar{b}_j(k) = \frac{\sum\limits_{\substack{t=1 \\ s.t.\ o_t = o_k}}^{T} \gamma_t(j)}{\sum\limits_{t=1}^{T} \gamma_t(j)}$$

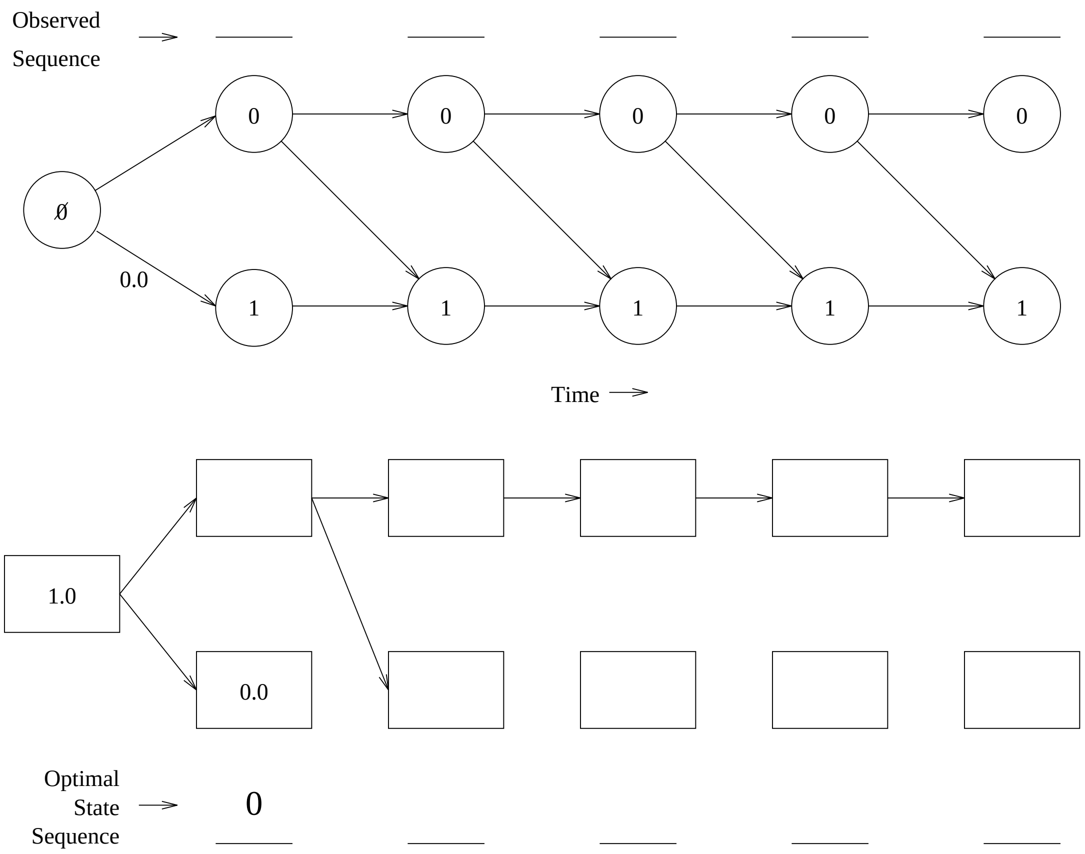
Remember that $\gamma_t(j)$ is the probability of being in state j at time t.

Hint: Start with the definition of the mean for a continuous random variable and approximate the integration with a sum over the quantized values.

Q10: Show that the reestimation formula for μ_j, the mean of the Gaussian of state j, is given by:

$$\mu_j = \frac{\sum\limits_{t=1}^{T} \gamma_t(j).o_t}{\sum\limits_{t=1}^{T} \gamma_t(j)}$$

Answer Sheet for Assignment5 Q6



Directions:

1. Enter the 5 symbols which you have observed (from T5) on the top line.
2. Label each arc in the top diagram with the joint probability of that transition being taken and the corresponding output being generated in the next state.
3. In the second diagram, enter in each box the probability of the most likely partial path through that state and draw arrows connecting the states in the best partial paths.
4. Read off the most likely hidden state sequence from the second diagram and write the result in the space provided.