

Fundamentals of Speech Recognition

Suggested Project

The Hidden Markov Model

1. Project Introduction

For this project, it is proposed that you design and implement a hidden Markov model (HMM) that optimally matches the behavior of a set of training sequences that will be provided to you as part of this project. The goal will be to use the standard set of forward-backward estimation algorithms to optimally determine the best (maximum likelihood) HMM that matches a given set of training data. You are also asked to use the Viterbi algorithm for estimating the model parameters and comparing and contrasting the results for the two methods. You may also want to investigate the effects of using subsets of the training data on the estimated models. Finally, if time permits, you are asked to design your own sequence generator and determine the effects of changing the training sequence characteristics on the estimated models.

2. HMM Generative Models

A total of four models were created with the following characteristics:

Model 1:

- number of states, $Q=5$
- type of density of observations: discrete
- number of observation possibilities, $K=5$
- type of model: ergodic
- state transition matrix density: random
- state observation matrix density: random
- state prior density: random

Model 2:

- number of states, $Q=5$
- type of density of observations: discrete
- number of observation possibilities, $K=5$
- type of model: ergodic
- state transition matrix density: skewed
- state observation matrix density: skewed
- state prior density: skewed

Model 3:

- number of states, $Q=5$
- type of density of observations: discrete
- number of observation possibilities, $K=5$
- type of model: left-right
- state transition matrix density: constrained

state observation matrix density: random
state prior density: constrained

Model 4:

number of states, $Q=5$
type of density of observations: discrete
number of observation possibilities, $K=5$
type of model: left-right
state transition matrix density: constrained
state observation matrix density: skewed
state prior density: constrained

A plot showing the generic structure of both the ergodic model (Figure 1) and the left-right model (Figure 2) is given below. (Note that only a subset of the state transitions are shown in the figure since it got very messy showing all possible state-state transition paths.)

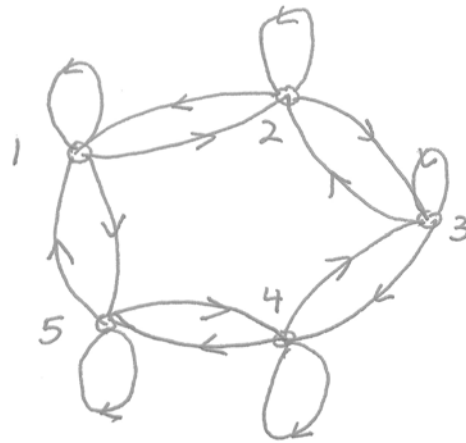


Figure 1—5 state ergodic model (only some of the actual state transitions shown in figure)



Figure 2—5 state left-right model

3. HMM Training Sequences

For this project there are four training sets (available from the course website), labeled:

1. hmm_observations_ergodic_random.mat,
2. hmm_observations_ergodic_skewed.mat,

3. hmm_observations_left-rt_random.mat
4. hmm_observations_left-rt_skewed.mat

The characteristics of these four datasets are as follows:

1. each mat file contains the following information:
 - a. an observation array, called $\text{data}(\text{nex}, T)$, where nex is 50 and T is 100. The elements of data are the observation values (between 1 and K) of each of the observation sequences as drawn from the discrete state observation matrix (see below)
 - b. a “hidden” states array, called $\text{states}(\text{nex}, T)$. The elements of states are the state indices (between 1 and Q) of each of the observation sequences as drawn from the discrete state prior density and the state transition matrix. The hidden states array cannot be used in any calculation, it is provided for sanity checks on the system.
 - c. a duration array, called $\text{duration}(\text{nex})$, which specifies the actual length of each of the nex training sequences. It should be noted that for the ergodic sequences, the duration of each sequence is exactly $T=100$ observations (with the actual observations given in data), but for the left-right models, the duration of each sequence is different and is specified in the duration array
 - d. a state prior array, called $\text{prior0}(Q)$, which specifies the probability of each training sequence starting in each of the Q states
 - e. a state transition matrix, called $\text{transmat0}(Q, Q)$, which specifies the probability, a_{ij} , of a state transition from state i to state j , where $1 \leq i, j \leq Q$
 - f. a state observation matrix, called $\text{obsmat0}(Q, K)$, which specifies the probability of symbol k appearing in state i , where $1 \leq k \leq K$, and $1 \leq i \leq Q$
2. the first two mat files contain training sequences for the ergodic model with either random entries for the state prior density, the state transition matrix, and the state observation matrix (the first mat file), or skewed entries for the state prior density, the state transition matrix, and the state observation matrix (the second mat file). The second two mat files contain training sequences for the left-right model with either random entries for the state observation matrix (the third mat file), or skewed entries for the state observation matrix (the fourth mat file). The model state prior density and state transition matrix are both highly skewed for both the third and fourth mat files. Again it is noted that for left-right models, the duration of the training sequences are variable and are specified in the duration array. For the ergodic models, the duration of each training sequence is fixed at $T=100$ observations.

4. Review of the Forward-Backward Algorithm

The goal of the forward-backward algorithm is to determine the values of the state prior density, the state transition matrix, and the state observation density that maximizes the

probability of the training set. This maximization is achieved in an iterative manner, namely by first postulating a (random or well educated) guess as to the values of the state prior density, the state transition matrix, and the state observation density, and then calculating the forward probabilities, $\alpha_t(i, n)$, for time t , state i , and training sequence n , the backward probabilities, $\beta_t(i, n)$, for time t , state i , and training sequence n , the scaling sequence $c(t, n)$, for time t , and training sequence n , and the scaled values of α and β , namely $\hat{\alpha}$ and $\hat{\beta}$ which are all computed as follows:

1. Obtain initial estimates of the state prior density, the state transition matrix, and the state observation matrix from either random guesses, or from skewed initial conditions.

Initial estimates of π , \mathbf{a} , and \mathbf{b} of the form:

$\pi = \{\pi_1, \pi_2, \dots, \pi_Q\}$; π_i = probability of being in state i at time 1

$$\mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & a_{1Q} \\ a_{21} & a_{22} & a_{2Q} \\ \vdots & \vdots & \vdots \\ a_{Q1} & a_{Q2} & a_{QQ} \end{bmatrix}$$

a_{ij} = probability of making a transition from state i to state j

$$\mathbf{b} = \begin{bmatrix} b_1(1) & b_1(2) & b_1(K) \\ b_2(1) & b_2(2) & b_2(k) \\ \vdots & \vdots & \vdots \\ b_Q(1) & b_Q(2) & b_Q(K) \end{bmatrix}$$

$b_i(k)$ = probability of observing the k^{th} symbol in state i

2. Use as the training set one of the set of observations read in earlier. We denote the complete training set of nex sequences of observations, with T_n being the duration of the n -th observation sequence, as:

O_t^n = value of the observation at time t and for sequence n ,

$1 \leq O_t^n \leq K, t = 1, 2, \dots, T_n, n = 1, 2, \dots, nex$

3. Compute the forward, backward, scale factor, and scaled estimates of α and β as:

Forward Loop--For each observation sequence $n = 1, 2, \dots, nex$ in the training set (with sequence duration T_n), compute the following:

Initialization Step ($t = 1$):

$$\alpha_1(i, n) = \pi_i b_i(O_1^n), \quad i = 1, 2, \dots, Q \quad (\text{unscaled } \alpha)$$

$$c(1, n) = \sum_{i=1}^Q \alpha_1(i, n) \quad (\text{scale factors})$$

$$\hat{\alpha}_1(i, n) = \alpha_1(i, n) / c(1, n) \quad (\text{scaled } \alpha)$$

Iteration Step:

for $t = 2, 3, \dots, T_n$

$$\alpha_t(i, n) = \left[\sum_{j=1}^Q \hat{\alpha}_{t-1}(j, n) a_{ji} \right] b_i(O_t^n), \quad i = 1, 2, \dots, Q$$

$$c(t, n) = \sum_{i=1}^Q \alpha_t(i, n)$$

$$\hat{\alpha}_t(i, n) = \alpha_t(i, n) / c(t, n), \quad i = 1, 2, \dots, Q$$

Backward Loop--For each observation sequence $n = 1, 2, \dots, nex$ in the training set (with sequence duration T_n), compute the following:

Initialization Step:

$$\beta_{T_n}(i, n) = 1; \quad i = 1, 2, \dots, Q \quad (\text{unscaled } \beta)$$

$$\hat{\beta}_{T_n}(i, n) = 1; \quad i = 1, 2, \dots, Q \quad (\text{scaled } \beta)$$

Iteration Step:

for $t = T_n - 1, T_n - 2, \dots, 1$

$$\beta_t(i, n) = \sum_{j=1}^Q a_{ij} \hat{\beta}_{t+1}(j, n) b_j(O_{t+1}^n), \quad i = 1, 2, \dots, Q$$

$$\hat{\beta}_t(i, n) = \beta_t(i, n) / c(t+1, n), \quad i = 1, 2, \dots, Q$$

4. Compute the state density, $\gamma_t(i, n)$, and the state transition density, $\xi_t(i, j, n)$, as:

$\gamma_t(i, n)$ = probability of being in state i at time t for sequence n

$$\gamma_t(i, n) = \hat{\alpha}_t(i, n) \cdot \hat{\beta}_t(i, n), \quad i = 1, 2, \dots, Q, \quad t = 1, 2, \dots, T_n, \quad n = 1, 2, \dots, nex$$

$\xi_t(i, j, n)$ = probability of being in state i at time t and making a transition to state j at time $t+1$ for sequence n

$$\xi_t(i, j, n) = \hat{\alpha}_t(i, n) \cdot a_{ij} \cdot b_j(O_{t+1}^n) \cdot \hat{\beta}_{t+1}(j, n) / c(t+1, n), \quad i = 1, 2, \dots, Q, \\ j = 1, 2, \dots, Q, \quad t = 1, 2, \dots, T_n, \quad n = 1, 2, \dots, nex$$

5. Re-estimate the state prior density, the state transition matrix and the state observation matrix, using the relations:

1. re-estimation of state prior density:

$$\bar{\pi}_i = \frac{\sum_{n=1}^{nex} \gamma_1(i, n)}{nex}, \quad i = 1, 2, \dots, Q$$

2. re-estimation of state transition matrix:

$$\bar{a}_{ij} = \frac{\sum_{n=1}^{nex} \sum_{t=1}^{T_n-1} \xi_t(i, j, n)}{\sum_{n=1}^{nex} \sum_{t=1}^{T_n-1} \gamma_t(i, n)}, \quad i = 1, 2, \dots, Q, \quad j = 1, 2, \dots, Q$$

3. re-estimation of state observation density matrix:

$$\bar{b}_j(k) = \frac{\sum_{n=1}^{nex} \sum_{t=1}^{T_n} \gamma_t(j, n)}{\sum_{n=1}^{nex} \sum_{t=1}^{T_n} \gamma_t(j, n)}, \quad j = 1, 2, \dots, Q, \quad k = 1, 2, \dots, K$$

6. Training sequence likelihood calculation. It is essential when doing re-estimation of HMM parameters to monitor the negative log probability of the total set of observation sequences to ensure that at each iteration the total negative log probability is decreasing,

and as a check as to how closely the likelihoods from the best estimate match or exceed the likelihood from the actual generator densities. This calculation of total negative log probability can be accomplished by keeping track, at each iteration, of the sum of the negative logs of all the scale factors, i.e.,

$$L_{training} = -\sum_{n=1}^{nex} \sum_{t=1}^{T_n} \log(c(t, n))$$

An interesting feature that can be used when the actual generator sequence is known is to do a single forward-backward pass using the actual generator state prior density, the actual state transition matrix, and the actual state observation matrix, thereby giving the training sequence likelihood for the actual generator. Interestingly, the best estimated model can have a training sequence likelihood that is actually higher than that of the generator sequence—especially when the actual sequences are generated from randomly generated state priors, state transition matrices, and state observation matrices. However, most of the time, the training sequence likelihoods are worse than that of the generator sequence—at least until the re-estimation procedure has reached a stable optimum and not gotten stuck at a local minimum.

5. Review of the Viterbi Algorithm

An alternative approach to the forward-backward method, for re-estimation of HMM model parameters, is the Viterbi algorithm which inherently finds the best matching path that best aligns each training sequence with the current model. By keeping track only of the best sequence, the computation can be reduced considerably. However, in cases where the best path is not much better than alternative paths, the Viterbi algorithm can lead to highly sub-optimal solutions. This occurs more often with randomly created HMM model parameters than with skewed model parameters, or with left-right models which have strong constraints on the state prior density and the state transition density.

The details of how the Viterbi algorithm operates are as follows:

1. Conversion to log probabilities--one time operation

$$\tilde{\pi}_i = \log(\pi_i), \quad i = 1, 2, \dots, Q$$

$$\tilde{a}_{ij} = \log(a_{ij}), \quad i = 1, 2, \dots, Q, \quad j = 1, 2, \dots, Q$$

$$\tilde{b}_i(k) = \log(b_i(k)), \quad i = 1, 2, \dots, Q, \quad k = 1, 2, \dots, K$$

2. For each sequence $n = 1, 2, \dots, nex$ in the training set (with sequence duration T_n), compute the following:

Initialization ($t = 1$):

$$\tilde{\delta}_1(i, n) = \tilde{\pi}_i + \tilde{b}_i(O_1^n), \quad i = 1, 2, \dots, Q$$

$$\psi_1(i, n) = 0; \quad i = 1, 2, \dots, Q$$

Recursion:

for $t = 2, 3, \dots, T_n$

$$\tilde{\delta}_t(j, n) = \max_{1 \leq i \leq Q} [\tilde{\delta}_{t-1}(i, n) + \tilde{a}_{ij}] + \tilde{b}_j(O_t^n), \quad j = 1, 2, \dots, Q$$

$$\psi_t(j, n) = \arg \max_{1 \leq i \leq Q} [\tilde{\delta}_{t-1}(i, n) + \tilde{a}_{ij}], \quad j = 1, 2, \dots, Q$$

Termination:

$$\tilde{P}^*(n) = \max_{1 \leq i \leq Q} [\tilde{\delta}_{T_n}(i, n)]$$

$$q_T^*(n) = \arg \max_{1 \leq i \leq Q} [\tilde{\delta}_{T_n}(i, n)]$$

Backtracking the best state sequence:

for $t = T_n - 1, T_n - 2, \dots, 1$

$$q_t^*(n) = \psi_{t+1}(q_{t+1}^*(n))$$

Once the Viterbi alignment of the training sequences with the current model has been obtained, we have essentially uncovered the hidden parts of the model, so we have a unique assignment of each observation to a distinct model state. Hence the re-estimation of the state prior density, the state transition matrix, and the state observation matrix is considerably simpler than for the forward-backward method, and can be stated simply as follows:

1. re-estimation of state prior density:

$$\bar{\pi}_i = \frac{\sum_{n=1}^{nex} c[q_1^*(n) = i]}{nex}, \quad i = 1, 2, \dots, Q$$

where $c[q_1^*(n) = i] = 1$ when $q_1^*(n) = i$ and it is zero otherwise

2. re-estimation of state transition matrix:

$$\bar{a}_{ij} = \frac{\sum_{n=1}^{nex} \sum_{t=1}^{T_n-1} c[q_t^*(n) = i, q_{t+1}^*(n) = j]}{\sum_{n=1}^{nex} \sum_{t=1}^{T_n-1} c[q_t^*(n) = i]}, \quad i = 1, 2, \dots, Q, \quad j = 1, 2, \dots, Q$$

where $c[q_t^*(n) = i, q_{t+1}^*(n) = j] = 1$ when $q_t^*(n) = i$ and $q_{t+1}^*(n) = j$ and it is zero otherwise

3. re-estimation of state observation density matrix:

$$\bar{b}_j(k) = \frac{\sum_{n=1}^{nex} \sum_{t=1}^{T_n} c[q_t^*(n) = j, O_t^n = k]}{\sum_{n=1}^{nex} \sum_{t=1}^{T_n-1} c[q_t^*(n) = j]}, \quad j = 1, 2, \dots, Q, \quad k = 1, 2, \dots, K$$

where $c[q_t^*(n) = j, O_t^n = k] = 1$ when $q_t^*(n) = j$ and $O_t^n = k$ and it is zero otherwise

6. HMM Estimation Project

The purpose of this project is to teach you how to realize both a forward-backward and a Viterbi procedure for estimating parameters of Hidden Markov Models. To that end you are asked to read in each of the four training sets, and to program HMM re-estimation algorithms using both a forward-backward approach and a Viterbi approach, and to compare and contrast the two procedures in terms of speed, accuracy, sensitivity to initial estimates, computational issues (taking logs of zero-valued quantities), and any other issue that arises during the course of this project.

It is suggested that you first write some simple code to generate a random state prior density, a random state transition density, and a random state observation density for the parameters of this exercise, namely $Q=5$ state models, with $K=5$ possible observations in each state. The actual generator densities are labeled `prior0`, `transmat0` and `obsmat0` so you might want to label your random selections as `prior1`, `transmat1`, and `obsmat1` to distinguish them from the actual sequence generators.

Once you read in each of the training sets, familiarize yourself with the data sequence which is of the form `data(nex,T)` where `nex` is 50 and `T` is 100. This data sequence is the observed set of outputs of the model, but, of course, there are no observed states as the states are hidden. Actually the array `states(nex,T)` has the generator state data but this cannot be used in any manner in your simulations. You should also remind yourself that the duration of each training sequence ($n=1,2,\dots,nex$) is specified in the array `duration(1:nex)`. (For the ergodic models that we will be using, the duration of each sequence is $T=100$. For the left-right models the duration is variable and is specified in `duration(1:nex)`.)

Once the training data has been read in, the first step you should do is score the training set on the generator model by running one full iteration of the forward-backward routine (which you have to write), and by noting the sum of the negative logs of the scaling sequences. This likelihood score represents a target value for your actual re-estimation routine—a target that you will sometimes surpass slightly (why is this the case) and mostly will miss hitting due to local optima or bad initial starts.

The next step is to create full forward-backward estimation and HMM model re-estimation routines, as well as a Viterbi estimation and HMM model re-estimation routine. Once these routines have been debugged, you can begin playing with the four training sets and determining how well you can estimate the model parameters. It is an interesting exercise to use the actual HMM densities (rather than random estimates) to determine how the total log likelihood score varies as you re-estimate the densities starting from the “ideal conditions”; however this exercise is just to show how good a solution can be obtained if you can get past local minima of the re-estimation routines.

In summary, you are asked to build two sets of HMM re-estimation routines, one based on the forward-backward algorithm, one based on the Viterbi state sequence, and to determine how well your algorithms work on two types of models, namely an ergodic model with 5 states and 5 possible observations in each state, and a left-right model with 5 states and 5 possible observations in each state. You should consider using several random starts to determine the best solution for each model and each training set. You are also given two versions of each model, one with randomly chosen values for the state prior density, the state transition matrix and the state observation matrix, and one with skewed values for the state prior density and the state transition matrix. Hence you should experiment with each of these training sets to understand which training methods work best, and why. Finally, you should investigate the effects of using highly constrained models (such as the left-right model) on the state prior density estimation and the state transition matrix estimation.

Other questions you should address as part of this project include the following:

1. for cases when the converged likelihood is comparable to the generator likelihood, how do the model densities match those of the generator?
2. for cases when the converged likelihood is comparable to the generator likelihood, how do the model states match those of the generator?
3. for cases when the converged likelihood is smaller than the generator likelihood (i.e., convergence at a local minimum but not the global minimum), what is happening with the states and how do the resulting models compare to the generator model?
4. how does the speed of convergence compare for the forward-backward re-estimation and the Viterbi re-estimation methods.
5. how do the computational requirements for the forward-backward and Viterbi methods compare?
6. how do the likelihood scores of the generator models compare when using forward-backward scoring with those when using Viterbi scoring? What accounts for these differences and how do they compare for random and skewed training sequences.
7. what do you think would be the effect of fewer training sequences (you can try this out by using less than the $n_{ex}=50$ sequences supplied)? What do you think would be the effect of more training sequences?

If you are successful in making the re-estimation methods work properly and efficiently, you might want to build your own HMM sequence generator and experiment with creating a range of model observations and see how well your re-estimation routines work on this new data. It is especially interesting to investigate the effect of increased numbers of observation sequences to determine how many are needed to make the convergence more rapid and less sensitive to the initial estimates of model parameters.