Hidden Markov Models

Lecturer: Shivani Agarwal

Disclaimer: These notes are designed to be a supplement to the lecture. They may or may not cover all the material discussed in the lecture (and vice versa).

Outline

• Introduction
• Finding the posterior distributions of individual hidden states (Forward-backward algorithm)
• Finding the most likely sequence of hidden states (Viterbi algorithm)
• Parameter estimation for HMMs

1 Introduction

Hidden Markov models (HMMs) are widely used to model sequence data in which the individual elements of a sequence cannot be assumed to be independent of each other. Examples of such sequence data include speech segments (sequences of acoustic signals), video segments (sequences of video frames), natural language sentences (sequences of words), biological sequences (sequences of nucleotides/amino acids), financial data (sequences of stock prices), etc.

HMMs are probabilistic models.\(^1\) They assume that for any sequence of observed signals \(x_{1:T} = (x_1, \ldots, x_T)\), there is a corresponding sequence of hidden states \(z_{1:T} = (z_1, \ldots, z_T)\) from which the observed signals are generated. For example, for speech data, the hidden states might be the words or phonemes being spoken, and the observed signals are the pronunciations of those words or phonemes. For financial data, the hidden states might be the market conditions on successive days, and the observed signals are the stock prices on those days. HMMs make a Markov assumption on the hidden states: namely, that given the state \(z_t\) at any time (position) \(t\), the next state \(z_{t+1}\) is conditionally independent of earlier states \(z_1, \ldots, z_{t-1}\) (and observations \(x_1, \ldots, x_t\)). They also assume that given the state \(z_t\) at any time (position) \(t\), the observed signal \(x_t\) is conditionally independent of all other states and observations. See Figure 1 for a pictorial representation.\(^2\)

Formally, let us define a sequence of random variables \(X_{1:T} = (X_1, \ldots, X_T)\) for the observed signals, and a sequence of random variables \(Z_{1:T} = (Z_1, \ldots, Z_T)\) for the hidden states. Then an HMM defines a joint probability distribution on the random variables \((X_{1:T}, Z_{1:T}) = (X_1, \ldots, X_T, Z_1, \ldots, Z_T)\). The hidden states \(Z_t\) are assumed to take one of a finite number of possible values, say \(K\) values that we will denote by \(\{1, \ldots, K\}\). The observations \(X_t\) can be discrete or continuous, depending on the application domain; for simplicity, we consider here only the discrete case, and assume that each \(X_t\) takes one of \(M\) possible values that we will denote by \(\{1, \ldots, M\}\). We will denote by \(x_t\) and \(z_t\) particular realizations of the random variables.

\(^1\)As we will see later, HMMs are a special case of the broad class of (directed) probabilistic graphical models.

\(^2\)We will discuss this type of representation in greater detail when we discuss probabilistic graphical models more generally.
Hidden Markov Models

Figure 1: Graphical representation of an HMM.

X_t and Z_t, respectively. Then, given the above conditional independence assertions, we have that the joint probability of any observation-state sequence \( (x_{1:T}, z_{1:T}) \) can be written as

\[
P(X_{1:T} = x_{1:T}, Z_{1:T} = z_{1:T}) = P(Z_1 = z_1) \left( \prod_{t=1}^{T-1} P(Z_{t+1} = z_{t+1} | Z_t = z_t) \right) \left( \prod_{t=1}^{T} P(X_t = x_t | Z_t = z_t) \right).
\]

We consider here only homogeneous HMMs, for which the state transition probabilities \( P(Z_{t+1} = k | Z_t = j) \) and observation/emission probabilities \( P(X_t = x | Z_t = k) \) are the same for all times (positions) \( t \). Such an HMM then has the following parameters:

- **Initial state probabilities**, given by a \( K \)-dimensional vector \( \pi \):
  \[
  \pi_k = P(Z_1 = k).
  \]
  Here \( \pi_k \geq 0 \) for all \( k \) and \( \sum_{k=1}^{K} \pi_k = 1 \).

- **State transition probabilities**, given by a \( K \times K \) matrix \( A \):
  \[
  A_{jk} = P(Z_{t+1} = k | Z_t = j).
  \]
  Here \( A_{jk} \geq 0 \) for all \( j, k \) and \( \sum_{k=1}^{K} A_{jk} = 1 \) for all \( j \).

- **Observation/emission probabilities**, given by a \( K \times M \) matrix \( \phi \):
  \[
  \phi_{kx} = P(X_t = x | Z_t = k).
  \]
  Here \( \phi_{kx} \geq 0 \) for all \( k, x \) and \( \sum_{x=1}^{M} \phi_{kx} = 1 \) for all \( k \).

Thus the joint probability of an observation-state sequence \( (x_{1:T}, z_{1:T}) \) under an HMM with parameters \( \theta = (\pi, A, \phi) \) is given by

\[
P(Z_{1:T} = z_{1:T}, X_{1:T} = x_{1:T}) = \pi_{z_1} \left( \prod_{t=1}^{T-1} A_{z_t, z_{t+1}} \right) \left( \prod_{t=1}^{T} \phi_{z_t, x_t} \right).
\]

There are several problems of interest associated with HMMs. For example, given an HMM with specified parameters \( \theta = (\pi, A, \phi) \), one is often interested in the following types of probabilistic inference problems:

- Given a partial observation sequence \( x_{1:t} \) up to time \( t \), find the posterior distribution of the hidden state \( z_t \) at time \( t \) (sometimes called the ‘filtering’ problem)
- Given an observation sequence \( x_{1:T} \), find the individually most likely hidden state \( z_t \) at each time \( t \) (\( 1 \leq t \leq T \))
• Given an observation sequence \( x_{1:T} \), find the jointly most likely sequence of hidden states \( z_{1:T} \)

In Section 2, we discuss the Forward-Backward algorithm, which given an observation sequence \( x_{1:T} \), finds the posterior distributions of all individual hidden states \( z_1, \ldots, z_T \); this can be used to solve the second problem above. As we will see, one of the recursions involved in the Forward-Backward algorithm also solves the first problem above. In Section 3, we discuss the Viterbi algorithm, which is used to solve the third problem.

Another set of problems involves estimating the parameters of an HMM from observed data. There are usually two cases of interest:

• Given observation-state sequences \( (x^i_{1:T}, z^i_{1:T}) \) \((i = 1, \ldots, m)\), estimate the parameters of an HMM

• Given only observation sequences \( (x^i_{1:T}) \) \((i = 1, \ldots, m)\), estimate the parameters of an HMM

The most common approach to parameter estimation in HMMs is via maximum likelihood estimation. In the first case above, this is straightforward; in the second case, the hidden states are unobserved, and the EM algorithm is used (in the context of HMMs, this is often also called the Baum-Welch algorithm). We discuss these in Section 4.

\section{Finding the Posterior Distributions of Individual Hidden States (Forward-Backward Algorithm)}

Suppose we are given an HMM with specified parameters \( \theta = (\pi, A, \phi) \).

We consider the following problem: Given an observation sequence \( x_{1:T} \), what is the posterior distribution of the hidden state at time \( t \) \((1 \leq t \leq T)\)? In particular, we will denote the posterior probability distribution of the random variable \( Z_t \) given \( x_{1:T} \) by \( \gamma_t \):

\[ \gamma_t(k) = \mathbb{P}(Z_t = k \mid X_{1:T} = x_{1:T}), \quad k = 1, \ldots, K. \]

Clearly, once we compute \( \gamma_t(k) \) for all \( k \), the individually most likely hidden state at time \( t \) (given the observation sequence \( x_{1:T} \)) is obtained simply by choosing the hidden state with maximal posterior probability:

\[ \tilde{z}_t \in \arg\max_k \gamma_t(k). \]

Now, by definition of conditional probability, we have

\[ \gamma_t(k) = \frac{\mathbb{P}(X_{1:T} = x_{1:T}, Z_t = k)}{\mathbb{P}(X_{1:T} = x_{1:T})}. \]

Note that each of the terms in the above ratio is a marginal probability over some of the variables in the full joint distribution of \((X_{1:T}, Z_{1:T})\). A naïve marginalization approach to computing each term would give

\[ \gamma_t(k) = \frac{\sum_{z_1, \ldots, z_t-1, z_{t+1}, \ldots, z_T} \mathbb{P}(X_{1:T} = x_{1:T}, Z_{1:T} = (z_1, \ldots, z_{t-1}, k, z_{t+1}, \ldots, z_T))}{\sum_{z_{1:T}} \mathbb{P}(X_{1:T} = x_{1:T}, Z_{1:T} = z_{1:T})}. \]

If these sums are computed naïvely, this would require \( O(K^T) \) computations. One can in fact compute the sums more cleverly by eliminating one variable at a time, and reduce the computation to \( O(K^2 T) \) for each \( t \). However, if we want to compute \( \gamma_t \) for all \( t \), then it turns out we can use a dynamic programming approach to re-use intermediate computations and do even better.
In particular, for each time \( t \in \{1, \ldots, T\} \) and state \( k \in \{1, \ldots, K\} \), let us define a forward variable \( \alpha_t(k) \) as follows:

\[
\alpha_t(k) = P(X_{1:t} = x_{1:t}, Z_t = k).
\]

This is the joint probability of seeing the partial observation sequence \( x_{1:t} \) up to time \( t \) and being in state \( k \) at time \( t \).

Similarly, for each time \( t \in \{1, \ldots, T - 1\} \) and state \( k \in \{1, \ldots, K\} \), we define a backward variable \( \beta_t(k) \) as follows:

\[
\beta_t(k) = P(X_{t+1:T} = x_{t+1:T} | Z_t = k)
\]

This is the conditional probability of seeing the partial observation sequence \( x_{t+1:T} \) in the future, given that the state at time \( t \) is \( k \). For convenience, we also define

\[
\beta_T(k) = 1.
\]

Now, we can write the posterior probabilities of the hidden states given the full observation sequence \( x_{1:T} \) in terms of these variables as follows:

\[
\gamma_t(k) = \frac{\alpha_t(k) \beta_t(k)}{\sum_{k'} \alpha_t(k') \beta_t(k')}
\]

To see this, note that we have

\[
P(X_{1:T} = x_{1:T}, Z_t = k) = P(X_{1:t} = x_{1:t}, Z_t = k) P(X_{t+1:T} = x_{t+1:T} | X_{1:t} = x_{1:t}, Z_t = k)
\]

\[
= P(X_{1:t} = x_{1:t}, Z_t = k) P(X_{t+1:T} = x_{t+1:T} | Z_t = k)
\]

\[
= \alpha_t(k) \beta_t(k).
\]

Therefore, our problem reduces to computing the forward and backward variables efficiently. This can be done by recursive use of the total probability theorem. In particular, for the forward variables, we have the following recursion:

\[
\alpha_1(k) = \pi_k \phi_{k,x_1}, \quad k = 1, \ldots, K
\]

\[
\alpha_{t+1}(k) = \left( \sum_{j=1}^{K} \alpha_t(j) A_{jk} \right) \phi_{k,x_{t+1}}, \quad t = 1, \ldots, T - 1; \ k = 1, \ldots, K
\]

Similarly, for the backward variables, we have the following backward recursion:

\[
\beta_T(k) = 1, \quad k = 1, \ldots, K
\]

\[
\beta_t(k) = \sum_{j=1}^{K} A_{kj} \phi_{j,x_{t+1}} \beta_{t+1}(j), \quad t = T - 1, \ldots, 1; \ k = 1, \ldots, K
\]

Thus, we can compute the variables \( \alpha_t(k), \beta_t(k) \) for all \( t \) and \( k \) using a total of \( (K^2T) \) computations! This is often called the Forward-Backward algorithm, and is summarized in Figure 2; the algorithm simply does a forward pass over \( t \) to compute all the forward variables, and then does a backward pass over \( t \) to compute all the backward variables. Once these variables are computed, computing the posterior distributions \( \gamma_t \) for all \( t \) takes only \( O(KT) \) additional computations. Thus, using the Forward-Backward algorithm, all the posterior distributions \( \gamma_t \) (for all \( 1 \leq t \leq T \)) can be computed in a total of \( O(K^2T) \) computations.

**Note on filtering problem.** In the filtering problem, one wants to compute ‘online’ posterior probabilities \( P(Z_t = k | X_{1:t} = x_{1:t}) \) at each time \( t \). Clearly, the forward variables above facilitate easy computation of these posteriors:

\[
P(Z_t = k | X_{1:t} = x_{1:t}) = \frac{\alpha_t(k)}{\sum_{k'=1}^{K} \alpha_t(k')}
\]

The forward recursion for the variables can therefore be used to solve the filtering problem as well; at each time \( t \), this requires \( O(K^2) \) computations.
Algorithm Forward-Backward

Inputs: HMM parameters \( \theta = (\pi, A, \phi) \); observation sequence \( x_{1:T} \)

Initialize:
For \( k = 1, \ldots, K \):
\[
\alpha_1(k) = \pi_k \phi_{k,x_1} \\
\beta_T(k) = 1
\]

Forward pass:
For \( t = 1, \ldots, T - 1 \)
For \( k = 1, \ldots, K \):
\[
\alpha_{t+1}(k) = \left( \sum_{j=1}^{K} \alpha_t(j) A_{jk} \right) \phi_{k,x_{t+1}}
\]

Backward pass:
For \( t = T - 1, \ldots, 1 \)
For \( k = 1, \ldots, K \):
\[
\beta_t(k) = \sum_{j=1}^{K} A_{kj} \phi_{j,x_{t+1}} \beta_{t+1}(j)
\]

Compute posterior probabilities of hidden states:
For \( t = 1, \ldots, T \)
For \( k = 1, \ldots, K \):
\[
\gamma_t(k) = \frac{\alpha_t(k) \beta_t(k)}{\sum_{k'}^{K} \alpha_t(k') \beta_t(k')}
\]

Outputs: Posterior probabilities \( \gamma_t(k) \) for \( t = 1, \ldots, T \) and \( k = 1, \ldots, K \)

Figure 2: The Forward-Backward algorithm.

3 Finding the Most Likely Sequence of Hidden States (Viterbi Algorithm)

Suppose again that we are given an HMM with specified parameters \( \theta = (\pi, A, \phi) \).

We now consider the following problem: Given an observation sequence \( x_{1:T} \), what is the most likely sequence of hidden states \( z_{1:T} \)? In other words, we would like to find
\[
z_{1:T}^* \in \operatorname{argmax}_{z_{1:T}} P(Z_{1:T} = z_{1:T} \mid X_{1:T} = x_{1:T}) .
\]

Since the conditioning on the observation sequence does not impact the maximization, we can also write this as
\[
z_{1:T}^* \in \operatorname{argmax}_{z_{1:T}} P(X_{1:T} = x_{1:T}, Z_{1:T} = z_{1:T}) .
\]

This is a maximum over \( K^T \) hidden state sequences, and a naïve brute-force approach would involve computing and comparing the joint probabilities for all \( O(K^T) \) sequences. We will again use a dynamic programming approach to compute the most likely sequence \( z_{1:T}^* \) – also called the maximum a posteriori (MAP) sequence – more efficiently.

For each time \( t \) and state \( k \), we will maintain the maximal probability of any partial observation-state sequence \((x_{1:t}, z_{1:t})\) that ends in hidden state \( z_t = k \) (and with \( x_{1:t} \) determined from the given observation sequence \( x_{1:T} \)). In particular, for each time \( t \in \{1, \ldots, T\} \) and state \( k \in \{1, \ldots, K\} \), we define
\[
\delta_t(k) = \max_{z_{1:t-1}} P(X_{1:t} = x_{1:t}, Z_{1:t} = (z_{1:t-1}, k)) .
\]
**Algorithm Viterbi**

**Inputs:** HMM parameters $\theta = (\pi, A, \phi)$; observation sequence $x_{1:T}$

**Initialize:**
For $k = 1, \ldots, K$:
$$\delta_1(k) = \pi_k \phi_{k,x_1}$$

Compute maximal joint probability over (partial) hidden state sequences (and maintain back-tracking variables):
For $t = 1, \ldots, T - 1$:
For $k = 1, \ldots, K$:
$$\delta_{t+1}(k) = \max_{1 \leq j \leq K} \left( \delta_t(j) A_{jk} \right) \phi_{k,x_{t+1}}$$
$$\psi_{t+1}(k) \in \arg\max_{1 \leq j \leq K} \left( \delta_t(j) A_{jk} \right)$$

Back-track to compute most likely hidden state sequence:
$$z^*_T \in \arg\max_{1 \leq k \leq K} \delta_T(k)$$
For $t = T - 1, \ldots, 1$:
$$z^*_t = \psi_{t+1}(z^*_{t+1})$$

**Output:** Most likely hidden state sequence $z^*_{1:T}$

Figure 3: The Viterbi algorithm.

It can be verified that these maximal probabilities can be computed recursively as follows:

$$\delta_1(k) = \pi_k \phi_{k,x_1}, \quad k = 1, \ldots, K$$
$$\delta_{t+1}(k) = \max_{1 \leq j \leq K} \left( \delta_t(j) A_{jk} \right) \phi_{k,x_{t+1}}, \quad t = 1, \ldots, T - 1; \quad k = 1, \ldots, K$$

This takes $O(K^2 T)$ computations. At the end of this process, clearly the maximal joint probability of any full observation-state sequence of the form $(x_{1:T}, z_{1:T})$ (with $x_{1:T}$ being the given observation sequence) is given by

$$\max_{1 \leq k \leq K} \delta_T(k),$$

and the last element of the most likely state sequence (given $x_{1:T}$) is given by

$$z^*_T \in \arg\max_{1 \leq k \leq K} \delta_T(k).$$

In order to retrieve the other elements of the most likely state sequence, we need to keep track of the states that achieve $\delta_t(k)$ for each $t$ and $k$. We do this by defining additional ‘back-tracking’ variables as follows:

$$\psi_{t+1}(k) \in \arg\max_{1 \leq j \leq K} \left( \delta_t(j) A_{jk} \right), \quad t = 1, \ldots, T - 1; \quad k = 1, \ldots, K$$

These can also be computed in $O(K^2 T)$ time. Then, working backwards, for each $t = T - 1, \ldots, 1$, the $t$-th element of the most likely state sequence is given by

$$z^*_t = \psi_{t+1}(z^*_{t+1}).$$

Clearly, once the $\delta_t(k)$ and $\psi_t(k)$ variables are computed, finding the most likely hidden sequence takes only $O(K T)$ additional computations. Thus the whole process requires only $O(K^2 T)$ computations. This is generally called the **Viterbi** algorithm and is summarized in Figure 3.
4 Parameter Estimation for HMMs

We now turn our attention to estimating the parameters of an HMM from training data. We will first consider the case where we are given complete observation-state sequences. In this case, we will see that HMM parameter estimation via maximum likelihood is fairly straightforward. We will then consider the case where we are given only observation sequences, and the states are hidden. In this case, we will see how to estimate HMM parameters using the EM algorithm.

To simplify notation below, when clear from context, we will abbreviate probabilities of events involving random variables by writing the corresponding probability mass functions, e.g. we will write \( P(X_{1:T} = x_{1:T}, Z_{1:T} = z_{1:T}) \) as \( p(x_{1:T}, z_{1:T}) \); \( P(Z_t = z_t) \) as \( p(z_t) \); \( P(Z_{t+1} = z_{t+1} \mid Z_t = z_t) \) as \( p(z_{t+1} \mid z_t) \); and so on. We will also make the dependence on model parameters \( \theta = (\pi, A, \phi) \) explicit, thus writing the joint probability of \( (X_{1:T}, Z_{1:T}) \) under parameters \( \theta \) as \( p(x_{1:T}, z_{1:T}; \theta) \), the probability of the first hidden state being \( z_1 \) under parameter vector \( \pi \) as \( p(z_1; \pi) \), and so on. Thus, in particular, we will write

\[
p(x_{1:T}, z_{1:T}; \theta) = p(z_1; \pi) \left( \prod_{t=1}^{T-1} p(z_{t+1} \mid z_t; A) \right) \left( \prod_{t=1}^{T} p(x_t \mid z_t; \phi) \right)
\]

\[
= \pi_{z_1} \left( \prod_{t=1}^{T-1} A_{z_t, z_{t+1}} \right) \left( \prod_{t=1}^{T} \phi_{z_t, x_t} \right).
\]

4.1 Parameter Estimation from Complete Data (Observation-State Sequences)

Say we are given \( m \) observation-state sequences \((x^i, z^i)\), \( i = 1, \ldots, m \), where the \( i \)-th sequence is of length \( T_i \), so that \((x^i, z^i) = (x_{1:T_i}^i, z_{1:T_i}^i) = (x_1^i, \ldots, x_{T_i}^i, z_1^i, \ldots, z_{T_i}^i)\). Denote by \((x, z)\) the full data: \((x, z) = (x^1, \ldots, x^m, z^1, \ldots, z^m)\). Then the log-likelihood of the complete data \((x, z)\), as a function of the HMM parameters \( \theta = (\pi, A, \phi) \) (assuming the \( m \) sequences are drawn independently from the same underlying HMM), is simply

\[
\ell(\theta; x, z) = \sum_{i=1}^{m} \ln p(x^i, z^i; \theta)
\]

\[
= \sum_{i=1}^{m} \left( \ln \pi_{z_1^i} + \sum_{t=1}^{T_i-1} \ln A_{z_t^i, z_{t+1}^i} + \sum_{t=1}^{T_i} \ln \phi_{z_t^i, x_t^i} \right).
\]

Maximizing this log-likelihood w.r.t. the individual parameters is then equivalent to estimating the individual multinomial distributions via maximum-likelihood, and gives the following parameter estimates:

\[
\hat{\pi}_k = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(z_1^i = k)
\]

\[
\hat{A}_{jk} = \frac{\sum_{i=1}^{m} \sum_{t=1}^{T_i-1} \mathbf{1}(z_t^i = j, z_{t+1}^i = k)}{\sum_{i=1}^{m} \sum_{t=1}^{T_i-1} \mathbf{1}(z_t^i = j)}
\]

\[
\hat{\phi}_{kx} = \frac{\sum_{i=1}^{m} \sum_{t=1}^{T_i} \mathbf{1}(z_t^i = k, x_t^i = x)}{\sum_{i=1}^{m} \sum_{t=1}^{T_i} \mathbf{1}(z_t^i = k)}.
\]
4.2 Parameter Estimation from Incomplete Data (Observation Sequences Only)

Now consider the case where we have only \( m \) observation sequences \( x_i = x_{i:T_i} = (x_{i1}, \ldots, x_{iT_i}), i = 1, \ldots, m, \) and the corresponding state sequences \( z^i \) are not known. Here the state sequences \( z^i = z_{i:T_i} = (z_{i1}, \ldots, z_{iT_i}) \) are hidden (latent) variables. The log-likelihood of the observed incomplete data \( x = (x^1, \ldots, x^m) \) is

\[
\ell(\theta; x) = \sum_{i=1}^{m} \ln p(x^i; \theta) = \sum_{i=1}^{m} \ln \left( \sum_{z^i} p(x^i, z^i; \theta) \right).
\]

One can in principle use gradient based methods to maximize the above objective function (accounting for the various probability constraints on the parameters by introducing Lagrange multipliers). An alternative and more widely used approach is to use the EM algorithm.

Recall that the EM algorithm is an iterative algorithm used for maximum likelihood estimation of parameters in the presence of hidden (latent) variables. In particular, the algorithm starts with an initial estimate of the parameters, \( \theta^0 \), and then iteratively updates these parameters by alternating between an E-step, in which it (implicitly) computes the posterior distribution of the hidden random variables \( Z \) under the current parameter estimates, and an M-step, in which it maximizes the expected (weighted) complete-data log-likelihood under the current posterior over \( Z \).

Since we are using \( t \) for time steps (positions) in sequences, let us use \( s \) to denote an EM iteration. For each iteration \( s \), denote by \( \theta^s = (\pi^s, A^s, \phi^s) \) the current parameter estimates. Denote by \( q^s(z^i) = p(z^i | x^i; \theta^s) \) the posterior distribution of the hidden state sequence \( z^i \) (given observation sequence \( x^i \)) under \( \theta^s \), and by \( q^s(z) = \prod_{i=1}^{m} q^s(z^i) \) the overall posterior of \( Z \) (given \( x \)) under \( \theta^s \). Then the expected complete-data log-likelihood under this posterior, as a function of \( \theta \), is given by

\[
\mathbb{E}_{Z \sim q^s}[\ell(\theta; x, Z)] = \mathbb{E}_{Z \sim q^s}\left[ \sum_{i=1}^{m} \ln p(x^i, Z^i; \theta) \right]
\]

\[
= \sum_{i=1}^{m} \mathbb{E}_{Z^i \sim q^s}[ \ln p(x^i, Z^i; \theta)]
\]

\[
= \sum_{i=1}^{m} \left( \sum_{k=1}^{K} P(Z^i_1 = k | X^i = x^i; \theta^s) \cdot \ln \pi_k 
+ \sum_{t=1}^{T_i-1} \sum_{j=1}^{K} \sum_{k=1}^{K} P(Z^i_t = j, Z^i_{t+1} = k | X^i = x^i; \theta^s) \cdot \ln A_{jk} 
+ \sum_{t=1}^{T_i} \sum_{k=1}^{K} P(Z^i_t = k | X^i = x^i; \theta^s) \cdot \ln \phi_{k,x^i_t} \right)
\]

\[
= \sum_{i=1}^{m} \left( \sum_{k=1}^{K} a^{i,s}_1(k) \cdot \ln \pi_k 
+ \sum_{t=1}^{T_i-1} \sum_{j=1}^{K} \sum_{k=1}^{K} a^{i,s}_{t+1}(j,k) \cdot \ln A_{jk} 
+ \sum_{t=1}^{T_i} \sum_{k=1}^{K} a^{i,s}_{t}(k) \cdot \ln \phi_{k,x^i_t} \right),
\]

\(^3\)Technically, in the E-step, we do not need the posterior distribution of \( Z \) itself, but rather the expected complete-data log-likelihood under this posterior.
where we have defined
\[
\gamma_{i,s}^{i,s}(k) = P(Z_i^t = k \mid X_i^t = x_i^t; \theta^s)
\]
\[
\xi_{i,s,t,t+1}^{i,s}(j,k) = P(Z_i^t = j, Z_{i+1}^t = k \mid X_i^t = x_i^t; \theta^s).
\]

Recall that the quantities \(\gamma_{i,s}^{i,s}(k)\) can be computed as
\[
\gamma_{i,s}^{i,s}(k) = \frac{\alpha_{i,s}^{i,s}(k) \beta_{i,s}^{i,s}(k)}{\sum_{k'=1}^K \alpha_{i,s}^{i,s}(k') \beta_{i,s}^{i,s}(k')},
\]
where \(\alpha_{i,s}^{i,s}(k), \beta_{i,s}^{i,s}(k)\) are the forward and backward variables for sequence \(i\) under the HMM parametrized by \(\theta^s\), as defined in Section 2. The quantities \(\xi_{i,s,t,t+1}^{i,s}(j,k)\) can be similarly computed:
\[
\xi_{i,s,t,t+1}^{i,s}(j,k) = \frac{\alpha_{i,s}^{i,s}(j) A_{j,k}^s \phi_{k,x_{i+1}}^s \beta_{i,s}^{i,s}(k)}{\sum_{j'=1}^K \sum_{k'=1}^K \alpha_{i,s}^{i,s}(j') A_{j',k'}^s \phi_{k',x_{i+1}}^s \beta_{i,s}^{i,s}(k')}.
\]

It can be seen that \(\gamma_{i,s}^{i,s}(j) = \sum_{k=1}^K \xi_{i,s,t,t+1}^{i,s}(j,k)\).

When maximizing the expected complete-data log-likelihood \(E_{Z \sim q_s}[\ell(\theta; x, Z)]\) over \(\theta\), the quantities \(\gamma_{i,s}^{i,s}(k)\) and \(\xi_{i,s,t,t+1}^{i,s}(j,k)\) can be treated as constants, since they depend only on the old parameters \(\theta^s\) and do not depend on the optimization variables \(\theta\). This gives the following update rules in the M-step:
\[
\pi_{k+1}^{s+1} = \frac{1}{m} \sum_{i=1}^m \gamma_{i,s}^{i,s}(k)
\]
\[
A_{jk}^{s+1} = \frac{\sum_{i=1}^m \sum_{t=1}^{T_i-1} \xi_{i,s,t,t+1}^{i,s}(j,k)}{\sum_{i=1}^m \sum_{t=1}^{T_i-1} \gamma_{i,s}^{i,s}(j)}
\]
\[
\phi_{kx}^{s+1} = \frac{\sum_{i=1}^m \sum_{t=1}^{T_i} \gamma_{i,s}^{i,s}(k) \cdot 1(x_i^t = x)}{\sum_{i=1}^m \sum_{t=1}^{T_i} \gamma_{i,s}^{i,s}(k)}
\]

Thus in the E-step, one computes the quantities \(\gamma_{i,s}^{i,s}(k)\) and \(\xi_{i,s,t,t+1}^{i,s}(j,k)\) (by running the Forward-Backward algorithm under the parameters \(\theta^s\)); in the M-step, one computes new parameters \(\theta^{s+1}\) as above. The E-step can be completed using \(O(K^2 (T_1 + \ldots + T_m))\) computations; the M-step can be completed using \(O((K^2 + KM)(T_1 + \ldots + T_m))\) computations. These steps are repeated until convergence (typically until the change in the incomplete-data log-likelihood, \(\ell(\theta^s; x) = \ln p(x; \theta^s)\), is less than some pre-specified tolerance threshold \(\epsilon\)).