Lecture 25: Hidden Markov Models-cont’d

ELE 525: Random Processes in Information Systems

Hisashi Kobayashi

Department of Electrical Engineering
Princeton University
January 13, 2014
20.4 Estimation Algorithms for State Sequence

Now consider a case where we wish to estimate the hidden state sequence $s(= s_0^T)$, based on the observation sequence $y(= y_0^T)$.

20.4.1 Forward Algorithm for MAP State Sequence Estimation

The MAP state sequence estimate $\hat{s}^*$ is defined by

$$\hat{s}^* = \arg \max_s \pi(s|y),$$  \hspace{1cm} (20.67)

which can be rewritten, using Bayes’ formula $\pi(s|y) = \frac{p(s,y)}{p(y)}$, as

$$\hat{s}^* = \arg \max_s p(s,y).$$  \hspace{1cm} (20.68)

We introduce the following auxiliary variables:

$$\tilde{\alpha}_t(j, y_0^t) \triangleq \max_{s_0^{t-1}} P[S_0^{t-1} = s_0^{t-1}, S_t = j, Y_0^t = y_0^t],$$  \hspace{1cm} (20.69)
The variable $\tilde{\alpha}_t(j, y_0^t)$ is similar to the forward variable $\alpha_t(j, y_0^t)$ of (20.54),

$$\alpha_t(j, y_0^t) = P[S_t = j, Y_0^t = y_0^t] = \sum_{s_0^{t-1}} P[S_0^{t-1} = s_0^{t-1}, S_t = j, Y_0^t = y_0^t]. \quad (20.70)$$

$\sum$ in (20.70) is replaced by $\max$ in (20.69).

Then analogous to the recursion formula (20.57) for the variable $\alpha_t(j, y_0^t)$, we obtain

$$\tilde{\alpha}_t(j, y_0^t) = \max_{i \in S} \{ \tilde{\alpha}_{t-1}(i, y_0^{t-1})c(i; j, y_t) \}, \quad j \in S, \ 1 \leq t \leq T, \quad (20.71)$$

$$\tilde{\alpha}_0(i, y_0) = \alpha_0(i, y_0) = P[S_0 = i, Y_0 = y_0], \quad i \in S. \quad (20.72)$$

We can show that the recursion formula leads to the MAP estimation sequence $\hat{s}^*$ of (20.68).
First we consider the case where we know the initial state \( s_0 \in S \).

Then, given an observation up to time \( t - 1 \) denoted as \( y_{0}^{t-1} \) suppose that we have found a most likely state sequence. We denote this most likely path on the trellis diagram as \( \hat{s}_{0}^{t-1}(i) \), and retain it as a **surviving sequence**.

Then we proceed to find a most likely state sequence that maximizes the RHS of (20.71) among all possible state sequences entering at state \( j \) at time \( t \), so there will be \( M (= |S|) \) surviving sequences, one per each state \( j \in S \).

Then there will be \( M \) surviving sequences when we reach \( t = T \).

The sequence that has the largest auxiliary variable \( \hat{\alpha}_{t}(j, y_{0}^{T}) \) is the MAP sequence estimate \( \hat{s} \).

As observed above, there are exactly \( M (= |S|) \) survivors at any time \( t \), but these \( M \) survivors may share a unique state subsequence up to time \( t' (< t) \). Then we know for sure that this sequence must be a part of the MAP sequence
The logarithmic conversion of probabilities:

By applying the logarithmic transformation to (20.71) and (20.72),

$$\tilde{\alpha}_t(j, y^t_0) = \max_{i \in S} \{ \tilde{\alpha}_{t-1}(i, y^{t-1}_0) + d(i; j, y_t) \}, \ j \in S, \ 1 \leq t \leq T,$$

where

$$\tilde{\alpha}_t(j, y^t_0) \triangleq \log \tilde{\alpha}_t(j, y^t_0),$$

$$d(i; j, y_t) \triangleq \log c(i; j, y_t) = \log P[S_t = j, Y_t = y_t | S_{t-1} = i],$$

with the initial value

$$\tilde{\alpha}_0(i, y_0) = \log (\alpha_0(i, y_0)) = d(\emptyset; i, y_0), \ i \in S.$$
20.4.2 The Viterbi Algorithm

Now consider the case where all possible state sequences are considered *equally likely*

\[ \pi(s) = \text{constant for all legitimate sequences } s \text{'s.} \quad (20.77) \]

Then, it is apparent that the MAP state sequence estimate also maximize the *likelihood function*

\[ L_y(s) \triangleq p(y|s), \quad (20.78) \]

where \( s = (s_0, s_1, \ldots, s_T) \). The solution is called the **maximum-likelihood sequence estimate** (MLSE).

The algorithms based on (20.73) is commonly known as the **Viterbi algorithm**
Algorithm 20.2  Forward Algorithm for MAP State Sequence Estimation: The Viterbi Algorithm

1: Compute the forward variables recursively:

$$\bar{\alpha}_0(i) = \log \alpha_0(i, y_0), \ i \in S,$$

$$\bar{\alpha}_t(j) = \max_{i \in S} \{\bar{\alpha}_{t-1}(i) + d(i, j; y_t)\}, \ j \in S, \ t = 1, 2, \ldots, T.$$ 

While computing the survivor’s score $\bar{\alpha}_t(j)$, keep a pointer to the state $\hat{s}^*_{t-1}$ from which the surviving path emanates, i.e.,

$$\hat{s}^*_{t-1} = \arg \max_{i \in S} \{\bar{\alpha}_{t-1}(i) + d(i, j; y_t)\}. $$

2: Find the surviving state at $t = T$, i.e.,

$$\arg \max_{j \in S} \bar{\alpha}_T(j) \triangleq \hat{s}^*_T.$$ 

3: Starting from $\hat{s}^*_T$, back-track the state sequence $(\hat{s}^*_T, \ldots, \hat{s}^*_t, \ldots, \hat{s}^*_1, \hat{s}^*_0)$, as the pointer to each surviving state indicates.
The state sequence

\[ \hat{s}^* = (\hat{s}_0^*, \hat{s}_1^*, \hat{s}_2^*, \ldots, \hat{s}_i^*, \ldots, \hat{s}_T^*) \]

thus obtained is the MAP state sequence estimate, which is also the MLSE when prior probabilities for all feasible state sequences are equal. i.e.,

\[ \hat{s}^* = \arg \max_{s} p(s, y). \]  \hspace{1cm} (20.79)

Note that if the initial state \( s_0 \) is known, the initial forward value should be set as

\[ \tilde{\alpha}_0(i) = \begin{cases} 0, & \text{for } i = s_0, \\ -\infty, & \text{for } i \neq s_0. \end{cases} \]
In a digital communications, the encoder output $o = T(s)$ is sent over a noisy channel.

If the channel is subject to additive white Gaussian noise (AWGN), the problem can be reduced to a **shortest path problem** on the trellis diagram.

If a noisy channel is characterized by a binary symmetric channel (BSC), then the distance metric becomes the Hamming distance.

Because of the assumption (20.77) we usually make, the Viterbi algorithm is often referred to as a computationally efficient algorithm for *maximum-likelihood sequence estimation* (MLSE), although it is more appropriate to call it a MAP sequence estimation algorithm.
The Viterbi algorithm has been successfully applied to convolutional decoders, to almost all digital recording systems, the so-called PRML (partial-response, maximum-likelihood) scheme and a channel with intersymbol interference.

The Viterbi algorithm [338, 339], originally devised by A. Viterbi in 1967 as an optimal decoding scheme for convolutional codes, is perhaps the most widely practiced algorithm in hidden Markov model applications. See Kobayashi [199] and Forney [108] for its early applications to digital communication channels with intersymbol interference, and Kobayashi [198, 197] for its application to high-density digital recording. See also Forney [107] and Hayes, Cover and Riera [147], Poor [271, 272] on the Viterbi and related algorithms.
20.5 The BCJR Algorithm

The BCJR algorithm was proposed [11] as an optimal scheme for the minimum symbol error-rate decoding of convolutional and linear codes.

Suppose that we wish to find a maximum a posteriori probability (MAP) estimate, of a hidden state $s_t$ at time $t$ (rather than the hidden state sequence $s$) on the basis of $y \triangleq (y_0, y_1, \ldots, y_T)$

$$\hat{s}_t^* = \arg \max_{i \in S} P[s_t = i | Y_0^T = y], \; t \in T \triangleq [0, 1, 2, \ldots, T].$$ \hspace{1cm} (20.80)

we consider the following a posteriori probability (APP) of state $S_t$

$$\gamma_t(i | y) \triangleq P[S_t = i | Y_0^T = y], \; i \in S, \; t \in T.$$ \hspace{1cm} (20.81)
we can interpret $\sum_{t \in T} \gamma_t(i|y)$ as the expected number of times that $S_t; \ t \in T$ enters state $i$, when $y$ is observed. This should be also equal to the expected number of transitions out of state $i$.

We will find it more convenient to use the following joint probability,

$$\lambda_t(i, y) \triangleq P[S_t = i, Y_0^T = y], \ i \in S, \ t \in T. \quad (20.82)$$

$$\gamma_t(i|y) = \frac{\lambda_t(i, y)}{p(y)}, \ i \in S, \ t \in T. \quad (20.83)$$

$\lambda_t(i, y)$ can be written in terms of the forward and backward variables

$$\lambda_t(i, y) = P[S_t = i, Y_0^t = y_0^t] P[Y_0^{T} = y_0^{T} | S_t = i, Y_0^t = y_0^t]$$

$$= \alpha_t(i, y_0^t) P[Y_{t+1}^T = y_0^{T} | S_t = i] = \alpha_t(i, y_0^t) \beta_t(i; y_0^{T})$$

$$= \alpha_t(i, y_0^t) \beta_t(i; y_0^{T}), \quad (20.84)$$
From (20.83) and (20.84), we find the following relation

\[
\gamma_t(i|y) = \frac{\alpha_t(i, y_0^t) \beta_t(i; y_{t+1}^T)}{p(y)}, \quad i \in S, \quad t \in T,
\]  

(20.85)

Once the APPs \( \gamma_t(i|y) \)'s are found for all states \( S_t = i \in S \) at given time \( t \), the MAP estimate (20.80) can then be expressed as

\[
\hat{s}_t^* = \arg \max_{i \in S} \left\{ \alpha_t(i, y_0^t) \beta_t(i; y_{t+1}^T) \right\}, \quad t \in T.
\]  

(20.86)
Algorithm 20.3  Forward-Backward Algorithm for MAP State Estimation: BCJR Algorithm

1: Compute and save the forward vector variables recursively:
\[ \alpha_0^\top = (\alpha_0(i, y_0), \ i \in S), \]
\[ \alpha_t^\top = \alpha_{t-1}^\top C(y_t), \ t = 1, 2, \ldots, T, \]

2: Compute the backward vector variables recursively and find the MAP state estimate:
\[ \beta_T = 1, \]
\[ \beta_t = C(y_{t+1}) \beta_{t+1}, \]
\[ \hat{s}_t^* = \arg \max_{i \in S} \alpha_t(i) \beta_t(i), \ t = T - 1, T - 2, \ldots, 1. \]

The BCJR algorithm can be also viewed as an extension of smoothing discussed in the context of Wiener filtering in Section 22.2 in the sense it provides an optimal estimate of \( s_t \) on the basis of the observations \( y_0^T \).
In Wiener filtering, \( Y_t \) and \( S_t \) are restricted to the case where \( Y_t = S_t + N_t \), and both \( S_t \) and \( N_t \) are WSS processes.
20.6 Maximum-Likelihood Estimation of Model Parameters

In Section 19.2.2 on the EM algorithm for missing data, we discussed an iterative method for an MLE of the parameter $\theta$ associated with $p_X(x; \theta)$, where only $Y$ of $X = (Y, Z)$ is observable. Our problem at hand exactly fits in that framework. The hidden Markov process $S$ corresponds to the latent variable $Z$ in that formulation.

20.6.1 Forward-Backward Algorithm for a Transition-Based HMM

In this section we focus on the EM algorithm for estimation and re-estimation of

$$\theta = (\alpha_0(j, k), c(i; j, k), \ i, j \in S, \ k \in Y).$$

Recall the general auxiliary function derived in (19.38)

$$Q(\theta | \theta^{(p)}) = E \left[ \log p(S, y; \theta) | y; \theta^{(p)} \right] = \sum_s p(s|y; \theta^{(p)}) \log p(s, y; \theta),$$

(20.87)
From (20.49) we have

$$p(s, y; \theta) = \alpha_0(s_0, y_0; \theta) \prod_{t=1}^{T} c(s_{t-1}; s_t, y_t),$$

(20.88)

By taking the logarithm

$$\log p(S, y; \theta) = \log \alpha_0(S_0, y_0; \theta) + \sum_{t=1}^{T} \log c(S_{t-1}; S_t, y_t).$$

(20.89)

Now take the expectation

$$Q(\theta|\theta^{(p)}) = E[\log \alpha_0(S_0, y_0; \theta)|y, \theta^{(p)}] + \sum_{t=1}^{T} E[\log p(S_t, y_t|S_{t-1}; \theta)|y, \theta^{(p)}]$$

$$= Q_0(\theta|\theta^{(p)}) + Q_1(\theta|\theta^{(p)}),$$

(20.90)

where we used the definition $c(s_{t-1}; s_t, y_t) = p(s_t, y_t|s_{t-1}; \theta)$. 

01/13/2014
Hisashi Kobayashi
The first term can be written as

\[ Q_0(\theta | \theta^{(p)}) = \sum_{i \in S} \log \alpha_0(i, y_0; \theta) \gamma_0(i | y; \theta^{(p)}), \quad (20.91) \]

where \( \gamma_0(i | y; \theta^{(p)}) = P[S_0 = i | y; \theta^{(p)}] \) is the \textit{a posteriori} probability (APP) of the initial hidden state \( S_0 \) defined by (20.85).

\[
\gamma_t(i | y) = \frac{\alpha_t(i, y_t^0) \beta_t(i; y_{t+1}^T)}{p(y)}, \quad i \in S, \ t \in T, \quad (20.85)
\]

Similarly,

\[ Q_1(\theta | \theta^{(p)}) = \sum_{t=1}^{T} \sum_{i \in S} \sum_{i \in S} \log p(j, y_t | i; \theta) \xi_{t-1}(i, j | y; \theta^{(p)}), \quad (20.92) \]

where \( \xi_{t-1}(i, j | y; \theta^{(p)}) = P[S_{t-1} = i, S_t = j | y; \theta^{(p)}] \) is the conditional joint probability of \((S_{t-1}, S_t)\).
\[ \xi_{t-1}^{(p)}(s_{t-1}, s_t | y) = \frac{\alpha_{t-1}^{(p)}(s_{t-1}, y_0^{t-1})c^{(p)}(s_{t-1}; s_t, y_t)\beta_t^{(p)}(s_t; y_{t+1}^T)}{L_y(\theta^{(p)})}, \] (20.93)

where we use the simplified notation

\[ \xi_{t-1}^{(p)}(s_{t-1}, s_t | y) \triangleq \xi_{t-1}(s_{t-1}, s_t | y; \theta^{(p)}), \]
\[ \alpha_{t-1}^{(p)}(s_{t-1}, y_0^{t-1}) \triangleq \alpha_{t-1}(s_{t-1}, y_0^{t-1}; \theta^{(p)}), \]
\[ \beta_t^{(p)}(s_t; y_{t+1}^T) \triangleq \beta_t(s_t; y_{t+1}^T; \theta^{(p)}). \]

Note that, because \( \gamma_t(i | y; \theta^{(p)}) = \sum_{j \in S} \xi_t(i, j | y; \theta^{(p)}) \), it is sufficient to calculate \( \xi_t(s_{t-1}, s_t | y; \theta^{(p)}) \) only.

Thus, the E-step can be performed using the forward-backward algorithm

In the forward part we compute and save the forward variables \( \alpha(y_0^t; \theta^{(p)}) \), and in the backward part we compute \( \beta(y_t^T; \theta^{(p)}) \)

and, using the saved forward variables, compute \( \xi_{t-1}(s_{t-1}, s_t | y; \theta^{(p)}) \)
The M-step of the EM algorithm for a transition-based HMM

Recall that in a transition-based HMM, the model parameter is $\theta = (\alpha, C)$, where

\[
\alpha = (\alpha_0(j, k); j \in S, k \in Y), \quad C = [c(i; j, k); i, j \in S, k \in Y],
\]

\[
\alpha_0(j, k) = P[S_0 = j, Y_0 = k], \quad j \in S, \quad k \in Y,
\]

\[
c(i; j, k) = P[S_t = j, Y_t = k|S_{t-1} = i], \quad i, j \in S, \quad k \in Y, \quad t = 1, 2, \ldots, T,
\]

We readily see from (20.91) and (20.92) that $Q_0(\theta|\theta^{(p)})$ depends only on $\alpha_0$ and $Q_1(\theta|\theta^{(p)})$ depends on $C$, but not on $\alpha_0$.

Recall the log-sum inequality (10.21):

\[
\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq a \log \frac{a}{b}, \tag{10.21}
\]

\[
\sum_{i=1}^{n} a_i \triangleq a \quad \text{and} \quad \sum_{i=1}^{n} b_i \triangleq b.
\]

where the equality holds when $\frac{b_i}{b} = \frac{a_i}{a}$, for all $i$. 
Then we find $Q_0(\theta | \theta^{(p)})$ is maximized at the $(p + 1)$-st step when $\alpha_0(j, y_0; \theta)$ is set

$$
\alpha_0^{(p+1)}(j, y_0) = \gamma_0(i | y; \theta^{(p)}) = \frac{\alpha_0^{(p)}(j, y_0) \beta_0^{(p)}(j; y_1^T)}{L_y(\theta^{(p)})}.
$$

(20.96)

Similarly, $Q_1(\theta | \theta^{(p)})$ can be maximized at

$$
c^{(p+1)}(i; j, k) = \frac{\sum_{t=1}^{T} \xi_{t-1}^{(p)}(i, j | y) \delta_{y_t, k}}{\sum_{j \in S} \sum_{t=1}^{T} \xi_{t-1}^{(p)}(i, j | y) \delta_{y_t, k}}.
$$

(20.97)
Algorithm 20.4  EM algorithm for a transition-based HMM

1: Set $p \leftarrow 0$, and denote the initial estimate of the model parameters as $\alpha_0^{(0)} = [\alpha_0^{(0)}(i, y_0), \ i \in \mathcal{S}]$ and $C^{(0)}(y_0) = [c^{(0)}(i; j, y_0); \ i, j \in \mathcal{S}, k \in \mathcal{Y}]$. 

2: The forward algorithm in the E-step: Compute and save the forward vector variables $\alpha_t^{(p)}$ recursively:

$$\alpha_t^{(p)} = \alpha_{t-1}^{(p)} C^{(p)}(y_t), \ t = 1, 2, \ldots, T,$$

3: Compute the likelihood function: $L^{(p)} = 1^T \alpha_T^{(p)}$.

4: The backward algorithm in the E-step: Compute the backward vector variables $\beta_t^{(p)}$ recursively. Compute and accumulate $\alpha_t^{(p)}(i) c^{(p)}(i; j, k) \beta_t^{(p)}(j)$.

1. Set $\beta_T^{(p)} = 1$, and $S^{(p)}(i, j, k) = 0$, $i, j \in \mathcal{S}, k \in \mathcal{Y}$.
2. For $t = T - 1, T - 2, \ldots, 0$:
   a. Compute $\beta_t^{(p)} = C^{(p)}(y_{t+1}) \beta_{t+1}^{(p)}$.
   b. Compute
      $$S^{(p)}(i, j, k) \leftarrow S^{(p)}(i, j, k) + \alpha_{t-1}^{(p)}(i) c^{(p)}(i; j, y_t) \beta_t^{(p)}(j) \delta_{k, y_t}.$$

5: The M-step: Update the model parameters:

$$\alpha_0^{(p+1)}(j) \leftarrow \frac{\alpha_0^{(p)}(j) \beta_0^{(p)}(j)}{L^{(p)}}, \ \text{for all} \ j \in \mathcal{S}$$

$$c^{(p+1)}(i; j, k) \leftarrow \frac{S^{(p)}(i, j, k)}{\sum_{j \in \mathcal{S}} S^{(p)}(i, j, k)} \ \text{for all} \ i, j \in \mathcal{S}, \ k \in \mathcal{Y}.$$

6: If any of the stopping conditions is met, stop the iteration and output the estimated $\alpha_0^{(p+1)}$ and $C^{(p+1)}$; else set $p \leftarrow p + 1$ and repeat the Steps 2 through 5.
20.6.2 The Baum-Welch Algorithm

Let us turn our attention to the state-based output model,

\[ \alpha_0(j, k) = \pi_0(j)b(j; k), \quad j \in \mathcal{S}, \quad k \in \mathcal{Y}, \]  

(20.100)

\[ c(i; j, k) = a(i, j)b(j; k), \quad i, j \in \mathcal{S}, \quad k \in \mathcal{Y}. \]  

(20.101)

Then the model parameter is \( \theta = (\pi_0, A, B) \), where

\[ \pi_0 = (\pi_0(i); i \in \mathcal{S}), \quad A = [a(i, j); i, j \in \mathcal{S}], \quad B = [b(j; k); j \in \mathcal{S}, k \in \mathcal{Y}]. \]

The EM algorithm for this type of HMM is known as the Baum-Welch algorithm [345].

Using (20.100), we can write \( Q_0 \) of (20.91) as

\[ Q_0(\theta|\theta^{(p)}) = \sum_{i \in \mathcal{S}} \log \pi_0(i) \gamma_i^{(p)}(i|y) + \sum_{i \in \mathcal{S}} \log b(i; y_0) \gamma_0^{(p)}(i|y), \]  

(20.102)

where

\[ \gamma_0^{(p)}(i|y) = \gamma_0(i|y; \theta^{(p)}). \]
Similarly, using (20.101), we write $Q_1$ of (20.92) as

$$Q_1(\theta|\theta^{(p)}) = \sum_{t=1}^{T} \sum_{i \in S, j \in S} \log a(i, j)\xi^{(p)}_{t-1}(i, j|y) + \sum_{t=1}^{T} \sum_{i \in S, j \in S} \log b(j; y_t)\xi^{(p)}_{t-1}(i, j|y)$$

$$= \sum_{t=1}^{T} \sum_{i \in S, j \in S} \log a(i, j)\xi^{(p)}_{t-1}(i, j|y) + \sum_{t=1}^{T} \sum_{j \in S} \log b(j; y_t)\gamma^{(p)}(j|y), \quad (20.103)$$

where we used the identity $\sum_{i \in S} \xi_{t-1}(i, j|y) = \gamma_t(j|y)$

Then

$$Q(\theta|\theta^{(p)}) = \sum_{i \in S} \log \pi_0(i)\gamma^{(p)}(i|y) + \sum_{t=1}^{T} \sum_{i \in S, j \in S} \log a(i, j)\xi^{(p)}_{t-1}(i, j|y)$$

$$+ \sum_{t=0}^{T} \sum_{j \in S} \log b(j; y_t)\gamma^{(p)}(j|y). \quad (20.104)$$
It is clear that we can maximize the three summed terms separately, and find the following $M$-step formula:

\[
\begin{align*}
\pi_0^{(p+1)}(i) &= \gamma_0^{(p)}(i \mid y), \quad i \in S, \\
a^{(p+1)}(i; j) &= \frac{\sum_{t=1}^{T} \xi_{t-1}(i, j \mid y)}{\sum_{i=1}^{T} \gamma_{t-1}^{(p)}(i \mid y)}, \quad i, j \in S, \\
b^{(p+1)}(j; k) &= \frac{\sum_{t=0}^{T} \gamma_{t}^{(p)}(j \mid y) \delta_{k, y_t}}{\sum_{t=0}^{T} \gamma_{t}^{(p)}(j \mid y)}, \quad j \in S.
\end{align*}
\] (20.105)

Algorithm 20.5 shows the EM algorithm for a state-based HMM, which is widely known as the Baum-Welch algorithm.
Algorithm 20.5  Baum-Welch Algorithm: The EM algorithm for a state-based HMM

1: Set \( p \leftarrow 0 \), and denote the initial estimate of the model parameters as \( \theta^{(0)} = (\pi_0^{(0)}, A^{(0)}, B^{(0)}(y_0)) \), and define an \( M \times M \) matrix \( C(y_0) = [a(i, j)b(j; y_0); i, j \in S] \)

2: The forward algorithm in the E-step: Compute and save the forward vector variables \( \alpha_t^{(p)} \) recursively:

\[
\alpha_0^{(p)^T} = (\alpha_0^{(p)}(i); i \in S), \text{ where } \alpha_0^{(p)}(i) = \pi_0^{(p)}(i)b^{(p)}(i; y_0)
\]

\[
\alpha_t^{(p)^T} = \alpha_{t-1}^{(p)^T}C^{(p)}(y_t), \quad t = 1, 2, \ldots, T,
\]

3: Compute the likelihood function: \( L^{(p)} = 1^T \alpha_T^{(p)} \).

4: The backward algorithm in the E-step: Compute the backward vector variables \( \beta_t^{(p)} \) recursively. Compute and accumulate the \( m_t^{(p)}(i, j) \) and \( n_t^{(p)}(j, k) \).

1. Set

\[
\beta_T^{(p)} = 1, \quad M^{(p)}(i, j) = 0 \text{ and }
\]

\[
N(j, k) = \alpha_T^{(p)}(j)\delta_{k, y_T}, \quad i, j \in S, k \in Y.
\]
2. For $t = T - 1, T - 2, \ldots, 1$:
   a. Compute $\beta_{t}^{(p)} = C^{(p)}(y_{t+1})\beta_{t+1}^{(p)}$
   b. Compute $m_{t}^{(p)}(i, j) = \alpha_{t-1}^{(p)}(i)\alpha^{(p)}(i, j)b^{(p)}(j, y_{t})\beta_{t}^{(p)}(j)$ and add to $M^{(p)}(i, j)$:
      $$M^{(p)}(i, j) \leftarrow M^{(p)}(i, j) + m_{t}^{(p)}(i, j), \quad i, j \in \mathcal{S}, k \in \mathcal{Y}.$$  
   c. Compute $n_{t}^{(p)}(j, k) = \alpha_{t}^{(p)}(j)\beta_{t}^{(p)}(j)\delta_{k, y_{t}}$ and add to $N^{(p)}(j, k)$:
      $$N^{(p)}(j, k) \leftarrow N^{(p)}(j, k) + n_{t}^{(p)}(j, y_{t}).$$

5. **The M-step:** Update the model parameters:
   $$\pi_{0}^{(p+1)}(i) \leftarrow \gamma_{0}^{(p)}(i), \quad i \in \mathcal{S},$$
   $$\alpha^{(p+1)}(i; j) \leftarrow \frac{M^{(p)}(i, j)}{\sum_{j \in \mathcal{S}} M^{(p)}(i, j)}, \quad i, j \in \mathcal{S},$$
   $$b^{(p+1)}(j; k) \leftarrow \frac{N^{(p)}(j, k)\delta_{k, y_{t}}}{N^{(p)}(j, y_{t})}, \quad j \in \mathcal{S}, \ y_{t} \in \mathcal{Y}.$$  

6. If any of the stopping conditions is met, stop the iteration and output the estimated the model parameters $\theta^{(p+1)} = (\pi_{0}^{(p+1)}, A^{(p+1)}, B^{(p+1)})(y)$; else set $p \leftarrow p + 1$, and repeat the Steps 2 through 5.
In order to simplify the notation, we define an $M \times M$ matrix $C(y_t)$

$$C(y_t) = [a(i, j)b(j; y_t); i, j \in S], \ y_t \in \mathcal{Y}.$$  \hspace{1cm} (20.108)

and the forward vector variables

$$\alpha_t = (\alpha_t(i); i \in S)^T. \hspace{1cm} (20.109)$$

For notational brevity, we drop the superscript $^{(p)}$.

$$m_t(i; j) \triangleq \alpha_{t-1}(i, y_{0}^{t-1})a(i; j)b(j; y_t)\beta_t(j; y_{t+1}^T), \ i, j \in S,$$

is the probability of a transition from state $S_{t-1} = i$ to $S_t = j$.

$$M(i; j) = \sum_{t=1}^{T} m_t(i, j), \ i, j \in S,$$

is the expected number of transitions from state $i$ into state $j$ in $S_0^T$. 
\[ n_t(j, k) \triangleq \alpha_t(j, y_0^t) \beta_t(j; y_{t+1}^T) \delta_{k, y_t}, \quad j \in S, y_t \in \mathcal{Y} \]

is the joint probability that \( S_t = [j], y_t = k \).

where \( \lambda_t(y, y) = \alpha_t(j, y_0^t) \beta_t(j; y_{t+1}^T) \), as defined in (20.82)

\[ N(j, k) = \sum_{t=0}^{T} n_t(j, y_t), \quad j \in S, k, y_t \in \mathcal{Y} \]

is the expected number of occurrences of \( (S_t = j, y_t = k) \) in \( (S_0^T, y_0^T) \).

\( \gamma_0(i) \) is the probability that the system is in state \( i \) at time \( t = 0 \).

Then

\[ a^{(p+1)}(i; j) \leftarrow \frac{M^{(p)}(i, j)}{\sum_{j \in S} M^{(p)}(i, j)}, \quad i, j \in S, \]

\[ b^{(p+1)}(j; k) \leftarrow \frac{N^{(p)}(j, k) \delta_{k, y_t}}{N^{(p)}(j, y_k)}, \quad j \in S, k, y_k \in \mathcal{Y}. \]