Hidden Markov Models (HMMs)

Handout Notes - Generative and Deep Learning (GDL)

Davide Bacciu - University of Pisa

Notation. Random variables are uppercase (e.g., Y_t, S_t) and observed values are lowercase (e.g., y_t, s_t). A dataset of sequences is denoted $\mathcal{D} = \{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}\}$ with $N = |\mathcal{D}|$. The n-th sequence has length T_n and is written $\mathbf{y}^{(n)} = (y_1^{(n)}, \dots, y_{T_n}^{(n)})$. Hidden states are $S_t \in \{1, \dots, C\}$ and observations are Y_t. HMM parameters are $\theta = (\boldsymbol{\pi}, A, B)$ where $\pi_i = p(S_1 = i)$, $A_{ij} = p(S_t = i \mid S_{t-1} = j)$, and $b_i(y) = p(Y_t = y \mid S_t = i)$ (emission probabilities); for discrete observations we also write $B_{hi} = p(Y_t = h \mid S_t = i)$. We use $p(\cdot)$ consistently for probabilities/mass functions (the model is discrete unless stated otherwise).

1 Sequential data and why HMMs

Many learning problems involve *sequences*: ordered collections of observations $\mathbf{y} = (y_1, \dots, y_T)$, where the index t denotes position (often time). A dataset typically contains multiple i.i.d. sequences $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)}$, possibly with different lengths $T_1, \dots, T_N$.

A first attempt at sequence modeling is a (first-order) Markov chain over observed variables:

$$p(y_1, \dots, y_T) = p(y_1) \prod_{t=2}^{T} p(y_t \mid y_{t-1}).$$

This is mathematically valid, but it may be semantically unsatisfying in many domains: we often believe there is an *unobserved* process driving the observations (e.g., a regime, a topic, a physical state), and that observations are noisy manifestations of that hidden process.

Hidden Markov Models (HMMs) formalize this idea by separating:

- a **hidden state dynamics** (a Markov chain over S_t), and
- an **observation model** (emissions Y_t generated from states S_t).

2 Markov chains as a building block

A first-order Markov chain over hidden states $S_1, \dots, S_T$ factorizes as

$$p(s_1, \dots, s_T) = p(s_1) \prod_{t=2}^{T} p(s_t \mid s_{t-1}).$$

The Markov assumption is

$$p(s_t \mid s_{1:t-1}) = p(s_t \mid s_{t-1}),$$

meaning the next state depends only on the previous state (extensions to higher order are possible, but less common).

3 Hidden Markov Models: definition and factorization

An HMM introduces a hidden state sequence $(S_1, \dots, S_T)$ and an observation sequence $(Y_1, \dots, Y_T)$ with two assumptions:

1. **Markovian dynamics:** $p(s_t \mid s_{1:t-1}) = p(s_t \mid s_{t-1})$.
2. **Conditional independence of emissions:** $p(y_t \mid s_{1:t}, y_{1:t-1}) = p(y_t \mid s_t)$.

With parameters $\theta = (\boldsymbol{\pi}, A, B)$, the joint distribution factorizes as

$$p(\mathbf{y}, \mathbf{s} \mid \theta) = p(s_1 \mid \boldsymbol{\pi})\, p(y_1 \mid s_1, B) \prod_{t=2}^{T} p(s_t \mid s_{t-1}, A)\, p(y_t \mid s_t, B).$$

For a discrete-state, discrete-observation HMM this is commonly written as

$$p(\mathbf{y}, \mathbf{s} \mid \theta) = \pi_{s_1}\, b_{s_1}(y_1) \prod_{t=2}^{T} A_{s_t s_{t-1}}\, b_{s_t}(y_t).$$

The observed-data likelihood is obtained by marginalizing out the hidden states:

$$p(\mathbf{y} \mid \theta) = \sum_{\mathbf{s}} p(\mathbf{y}, \mathbf{s} \mid \theta) = \sum_{s_1=1}^{C} \cdots \sum_{s_T=1}^{C} \pi_{s_1} b_{s_1}(y_1) \prod_{t=2}^{T} A_{s_t s_{t-1}} b_{s_t}(y_t).$$

Naively this sum is $O(C^T)$, which is infeasible for realistic T. The key contribution of HMM inference algorithms is to exploit the chain structure to compute these quantities in $O(TC^2)$.

4 Three central inference problems

Given a model θ and an observed sequence $\mathbf{y}$, three recurring tasks are:

Smoothing (state posterior). Compute the posterior distribution of the hidden state at time t:

$$p(S_t = i \mid \mathbf{y}, \theta).$$

Decoding (optimal state assignment). Compute an optimal state sequence $\mathbf{s}^\star = (s_1^\star, \ldots, s_T^\star)$ under some criterion. Two common criteria are:

- **Pointwise MAP:** $s_t^\star = \arg\max_i p(S_t = i \mid \mathbf{y}, \theta)$ (may not yield a globally consistent path).
- **Most likely path:** $\mathbf{s}^\star = \arg\max_{\mathbf{s}} p(\mathbf{y}, \mathbf{s} \mid \theta)$ (Viterbi).

Learning (parameter estimation). Given a dataset $\mathcal{D}$ of observed sequences and a fixed number of states C, estimate θ by maximum likelihood:

$$\theta_{\mathrm{ML}} = \arg\max_{\theta} \prod_{n=1}^{N} p(\mathbf{y}^{(n)} \mid \theta), \qquad \ell(\theta) = \sum_{n=1}^{N} \log p(\mathbf{y}^{(n)} \mid \theta).$$

Because S_t is unobserved, learning is an instance of maximum likelihood with latent variables and is classically solved by EM (Baum–Welch for HMMs).

5 Forward–Backward algorithm (sum–product on a chain)

The forward–backward algorithm computes smoothing posteriors efficiently via dynamic programming. The core idea is to decompose the contribution of the left and right parts of the chain around time t.

5.1 Forward messages

Define the forward quantity

$$\alpha_t(i) = p(S_t = i, Y_{1:t} = y_{1:t} \mid \theta).$$

Intuitively, $\alpha_t(i)$ is the joint probability of being in state i at time t and having generated the prefix $y_{1:t}$.

Worked example — Forward recursion

Initialization:

$$\alpha_1(i) = p(S_1 = i \mid \boldsymbol{\pi})\, p(Y_1 = y_1 \mid S_1 = i, B) = \pi_i\, b_i(y_1).$$

Recursion (for $t \geq 2$):

$$\alpha_t(i) = p(S_t = i, y_{1:t}) = p(y_t \mid S_t = i) \sum_{j=1}^{C} p(S_t = i \mid S_{t-1} = j)\, p(S_{t-1} = j, y_{1:t-1}),$$

hence

$$\boxed{\alpha_t(i) = b_i(y_t) \sum_{j=1}^{C} A_{ij}\, \alpha_{t-1}(j).}$$

Termination (likelihood):

$$p(\mathbf{y} \mid \theta) = \sum_{i=1}^{C} \alpha_T(i).$$

5.2 Backward messages

Define the backward quantity

$$\beta_t(i) = p(Y_{t+1:T} = y_{t+1:T} \mid S_t = i, \theta).$$

Intuitively, $\beta_t(i)$ is the probability of generating the *suffix* from $t+1$ onward given the current state at time t.

Worked example — Backward recursion

Initialization:

$$\beta_T(i) = 1 \quad \text{for all } i,$$

because there are no observations after time T.
Recursion (for $t = T-1, T-2, \ldots, 1$):

$$\beta_t(j) = p(y_{t+1:T} \mid S_t = j) = \sum_{i=1}^{C} p(S_{t+1} = i \mid S_t = j)\, p(y_{t+2:T} \mid S_{t+1} = i)\, p(y_{t+1} \mid S_{t+1} = i),$$

so

$$\boxed{\beta_t(j) = \sum_{i=1}^{C} A_{ij}\, b_i(y_{t+1})\, \beta_{t+1}(i).}$$

5.3 Smoothing: combining forward and backward

The state posterior factorizes as

$$p(S_t = i \mid \mathbf{y}, \theta) \propto p(S_t = i, y_{1:t})\, p(y_{t+1:T} \mid S_t = i) = \alpha_t(i)\, \beta_t(i).$$

Worked example — Smoothing posterior and pairwise posterior

Define

$$\gamma_t(i) \equiv p(S_t = i \mid \mathbf{y}, \theta).$$

Then

$$\boxed{\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{C} \alpha_t(j)\beta_t(j)}.}$$

For EM learning we also need the *pairwise* posterior for successive states:

$$\xi_t(i,j) \equiv p(S_t = i, S_{t-1} = j \mid \mathbf{y}, \theta) \quad (t \geq 2).$$

Using the chain factorization,

$$p(S_{t-1} = j, S_t = i, \mathbf{y}) = \alpha_{t-1}(j)\, A_{ij}\, b_i(y_t)\, \beta_t(i),$$

hence

$$\boxed{\xi_t(i,j) = \frac{\alpha_{t-1}(j)\, A_{ij}\, b_i(y_t)\, \beta_t(i)}{\sum_{m=1}^{C} \sum_{\ell=1}^{C} \alpha_{t-1}(m)\, A_{\ell m}\, b_\ell(y_t)\, \beta_t(\ell)}.}$$

Message passing perspective. Forward–backward is a special case of *sum–product message passing* on a chain: messages propagate forward and backward, and marginals are obtained by multiplying incoming messages and normalizing.

6 Viterbi algorithm (max–product decoding)

When the goal is a single most likely state sequence

$$\mathbf{s}^\star = \arg\max_{\mathbf{s}} p(\mathbf{y}, \mathbf{s} \mid \theta),$$

the sum in sum–product is replaced by a max. This is *max–product* message passing, implemented by the Viterbi algorithm.

Worked example — Viterbi recursion (most likely path)

Define the best path score ending in state i at time t:

$$\delta_t(i) = \max_{s_{1:t-1}} p(S_t = i, S_{1:t-1} = s_{1:t-1}, Y_{1:t} = y_{1:t} \mid \theta).$$

Initialization:

$$\delta_1(i) = \pi_i \, b_i(y_1).$$

Recursion (for $t \geq 2$):

$$\delta_t(i) = b_i(y_t) \max_{j \in \{1,\dots,C\}} A_{ij} \, \delta_{t-1}(j).$$

Store backpointers

$$\psi_t(i) = \arg\max_j A_{ij} \, \delta_{t-1}(j).$$

Termination:

$$s_T^\star = \arg\max_i \delta_T(i),$$

and backtrack for $t = T-1, \dots, 1$:

$$s_t^\star = \psi_{t+1}(s_{t+1}^\star).$$

In practice, compute in log space to avoid underflow:

$$\log \delta_t(i) = \log b_i(y_t) + \max_j \big(\log A_{ij} + \log \delta_{t-1}(j) \big).$$

7 Learning HMM parameters with EM (Baum–Welch)

We now consider maximum likelihood learning from $\mathcal{D} = \{\mathbf{y}^{(n)}\}_{n=1}^N$ when states are unobserved. EM alternates:

- **E-step:** compute posterior expectations of latent-state indicators given current parameters $\theta^{(k)}$;
- **M-step:** maximize the expected complete-data log-likelihood to obtain $\theta^{(k+1)}$.

7.1 Complete-data log-likelihood with indicators

For the n-th sequence, introduce indicator variables

$$z_{ti}^{(n)} = \mathbb{I}[S_t^{(n)} = i], \qquad z_{ti,\,t-1\,j}^{(n)} = \mathbb{I}[S_t^{(n)} = i, \; S_{t-1}^{(n)} = j].$$

The complete-data log-likelihood for the whole dataset can be written (up to terms not depending on θ) as:

$$\log p(\mathcal{D}, \mathcal{Z} \mid \theta) = \sum_{n=1}^N \left[\sum_{i=1}^C z_{1i}^{(n)} \log \pi_i + \sum_{t=2}^{T_n} \sum_{i=1}^C \sum_{j=1}^C z_{ti,\,t-1\,j}^{(n)} \log A_{ij} + \sum_{t=1}^{T_n} \sum_{i=1}^C z_{ti}^{(n)} \log b_i\big(y_t^{(n)}\big) \right].$$

In EM, we replace indicators by their posterior expectations:

$$\mathbb{E}[z_{ti}^{(n)} \mid \mathbf{y}^{(n)}, \theta^{(k)}] = \gamma_t^{(n)}(i), \qquad \mathbb{E}[z_{ti,\,t-1\,j}^{(n)} \mid \mathbf{y}^{(n)}, \theta^{(k)}] = \xi_t^{(n)}(i,j),$$

which are computed by forward–backward.

7.2 E-step

For each sequence n, run forward–backward under $\theta^{(k)}$ and compute:

$$\gamma_t^{(n)}(i) = p(S_t^{(n)} = i \mid \mathbf{y}^{(n)}, \theta^{(k)}), \qquad \xi_t^{(n)}(i,j) = p(S_t^{(n)} = i, S_{t-1}^{(n)} = j \mid \mathbf{y}^{(n)}, \theta^{(k)}).$$

These are the sufficient statistics needed for the M-step in discrete HMMs.

7.3 M-step (multinomial initial, transition, and emission)

Assume a discrete observation alphabet $\mathcal{V} = \{1, \dots, H\}$ and define emission parameters $B_{hi} = p(Y_t = h \mid S_t = i)$, with constraints $\sum_{h=1}^{H} B_{hi} = 1$ for each state i.

Expected complete-data objective. Define the EM auxiliary function:

$$Q(\theta \mid \theta^{(k)}) = \mathbb{E}_{\mathcal{Z}|\mathcal{D},\theta^{(k)}}\big[\log p(\mathcal{D}, \mathcal{Z} \mid \theta)\big].$$

Plugging the expectations yields

$$Q(\theta \mid \theta^{(k)}) = \sum_{n=1}^{N} \left[\sum_{i=1}^{C} \gamma_1^{(n)}(i) \log \pi_i + \sum_{t=2}^{T_n} \sum_{i=1}^{C} \sum_{j=1}^{C} \xi_t^{(n)}(i,j) \log A_{ij} + \sum_{t=1}^{T_n} \sum_{i=1}^{C} \gamma_t^{(n)}(i) \log b_i\big(y_t^{(n)}\big) \right].$$

When emissions are multinomial over $\mathcal{V}$, we write

$$\log b_i(y) = \sum_{h=1}^{H} \mathbb{I}[y = h] \log B_{hi}.$$

Worked example — M-step updates via Lagrange multipliers (discrete HMM)

1) Initial distribution $\boldsymbol{\pi}$. Maximize

$$Q_\pi(\boldsymbol{\pi}) = \sum_{n=1}^{N}\sum_{i=1}^{C}\gamma_1^{(n)}(i)\log\pi_i \quad \text{s.t.} \quad \sum_{i=1}^{C}\pi_i = 1.$$

Lagrangian:

$$\mathcal{J}(\boldsymbol{\pi},\lambda) = \sum_{n=1}^{N}\sum_{i=1}^{C}\gamma_1^{(n)}(i)\log\pi_i + \lambda\Big(\sum_{i=1}^{C}\pi_i - 1\Big).$$

Stationarity:

$$\frac{\partial\mathcal{J}}{\partial\pi_i} = \frac{\sum_{n=1}^{N}\gamma_1^{(n)}(i)}{\pi_i} + \lambda = 0 \;\Rightarrow\; \pi_i = -\frac{\sum_n\gamma_1^{(n)}(i)}{\lambda}.$$

Normalize using $\sum_i\pi_i = 1$:

$$\sum_{i=1}^{C}\pi_i = -\frac{1}{\lambda}\sum_{n=1}^{N}\sum_{i=1}^{C}\gamma_1^{(n)}(i) = -\frac{N}{\lambda} = 1 \;\Rightarrow\; \lambda = -N,$$

since $\sum_i\gamma_1^{(n)}(i) = 1$ for each sequence n. Thus

$$\boxed{\pi_i^{(k+1)} = \frac{1}{N}\sum_{n=1}^{N}\gamma_1^{(n)}(i).}$$

2) Transition matrix A. Maximize

$$Q_A(A) = \sum_{n=1}^{N}\sum_{t=2}^{T_n}\sum_{i=1}^{C}\sum_{j=1}^{C}\xi_t^{(n)}(i,j)\log A_{ij} \quad \text{s.t.} \quad \forall j:\ \sum_{i=1}^{C}A_{ij} = 1.$$

For each fixed previous-state j, define

$$N_{ij} \equiv \sum_{n=1}^{N}\sum_{t=2}^{T_n}\xi_t^{(n)}(i,j), \qquad N_{\cdot j} \equiv \sum_{i=1}^{C}N_{ij}.$$

Then the constrained objective for column j is $\sum_i N_{ij}\log A_{ij}$ with $\sum_i A_{ij} = 1$. Lagrangian (per j):

$$\mathcal{J}_j(\{A_{ij}\}_i,\lambda_j) = \sum_{i=1}^{C}N_{ij}\log A_{ij} + \lambda_j\Big(\sum_{i=1}^{C}A_{ij} - 1\Big).$$

Stationarity gives $A_{ij} = -N_{ij}/\lambda_j$. Enforcing $\sum_i A_{ij} = 1$ yields $\lambda_j = -N_{\cdot j}$, hence

$$\boxed{A_{ij}^{(k+1)} = \frac{\sum_{n=1}^{N}\sum_{t=2}^{T_n}\xi_t^{(n)}(i,j)}{\sum_{n=1}^{N}\sum_{t=2}^{T_n}\gamma_{t-1}^{(n)}(j)}}.$$

(Here we used $N_{\cdot j} = \sum_{n,t}\sum_i\xi_t^{(n)}(i,j) = \sum_{n,t}\gamma_{t-1}^{(n)}(j)$.)

Worked example — M-step updates (Continued)

3) Emission probabilities B. For multinomial emissions, maximize

$$Q_B(B) = \sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{C}\gamma_t^{(n)}(i)\log b_i(y_t^{(n)}) = \sum_{n,t}\sum_{i=1}^{C}\sum_{h=1}^{H}\gamma_t^{(n)}(i)\,\mathbb{I}[y_t^{(n)}=h]\log B_{hi}$$

subject to $\forall i:\ \sum_{h=1}^{H} B_{hi} = 1$. For each state i define expected emission counts

$$M_{hi} \equiv \sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_t^{(n)}(i)\,\mathbb{I}[y_t^{(n)}=h], \qquad M_{\cdot i} \equiv \sum_{h=1}^{H} M_{hi} = \sum_{n,t}\gamma_t^{(n)}(i).$$

Then for fixed i we maximize $\sum_h M_{hi}\log B_{hi}$ with $\sum_h B_{hi} = 1$. By the same Lagrange-multiplier argument as above,

$$\boxed{B_{hi}^{(k+1)} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_t^{(n)}(i)\,\mathbb{I}[y_t^{(n)}=h]}{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_t^{(n)}(i)}}.$$

These updates have a clear interpretation: they are normalized *expected counts* under the posterior distribution over hidden states and transitions. In other words, EM for HMMs learns the parameters as if the hidden sequence were observed, but with fractional (soft) membership given by γ and ξ.

8 Pseudocode: learning an HMM with EM

This section gives a practical, implementation-oriented summary of how EM learning for a discrete HMM is carried out in combination with the forward–backward algorithm. The forward–backward pass provides the posterior sufficient statistics needed by the M-step updates.

Worked example — EM for HMM: the Baum–Welch Algorithm

Input: dataset $\mathcal{D} = \{\mathbf{y}^{(n)}\}_{n=1}^{N}$, $\mathbf{y}^{(n)} = (y_1^{(n)}, \ldots, y_{T_n}^{(n)})$,
number of states C, alphabet size H, tolerance ε, max iterations $K_{\max}$.
Output: parameters $\theta = (\boldsymbol{\pi}, A, B)$.

Initialize: Choose $\boldsymbol{\pi}^{(0)}$ with $\sum_i \pi_i^{(0)} = 1$, $A^{(0)}$ column-stochastic ($\sum_i A_{ij}^{(0)} = 1$ for each j), and $B^{(0)}$ column-stochastic ($\sum_h B_{hi}^{(0)} = 1$ for each i). Set $k \leftarrow 0$.

Repeat until convergence or $k = K_{\max} - 1$:

E-step: compute posteriors with forward–backward.
Initialize accumulators:

$$\widehat{\pi}_i \leftarrow 0, \qquad \widehat{A}_{ij} \leftarrow 0, \qquad \widehat{A}_{\cdot j} \leftarrow 0, \qquad \widehat{B}_{hi} \leftarrow 0, \qquad \widehat{B}_{\cdot i} \leftarrow 0.$$

Also initialize total log-likelihood $\ell^{(k)} \leftarrow 0$.

For each sequence $n = 1, \ldots, N$:

(1) Forward pass (with optional scaling).
Compute $\alpha_t^{(n)}(i)$ for $t = 1, \ldots, T_n$:

$$\alpha_1^{(n)}(i) \leftarrow \pi_i^{(k)}\, b_i^{(k)}\left(y_1^{(n)}\right),$$

$$\alpha_t^{(n)}(i) \leftarrow b_i^{(k)}\left(y_t^{(n)}\right) \sum_{j=1}^{C} A_{ij}^{(k)}\, \alpha_{t-1}^{(n)}(j) \qquad (t = 2, \ldots, T_n).$$

Compute the sequence likelihood:

$$p(\mathbf{y}^{(n)} \mid \theta^{(k)}) = \sum_{i=1}^{C} \alpha_{T_n}^{(n)}(i), \qquad \ell^{(k)} \leftarrow \ell^{(k)} + \log p(\mathbf{y}^{(n)} \mid \theta^{(k)}).$$

(For long sequences, use scaling factors or log-space to avoid underflow.)

(2) Backward pass.
Compute $\beta_t^{(n)}(i)$ for $t = T_n, \ldots, 1$:

$$\beta_{T_n}^{(n)}(i) \leftarrow 1,$$

$$\beta_t^{(n)}(j) \leftarrow \sum_{i=1}^{C} A_{ij}^{(k)}\, b_i^{(k)}\left(y_{t+1}^{(n)}\right)\, \beta_{t+1}^{(n)}(i) \qquad (t = T_n - 1, \ldots, 1).$$

(3) Posteriors: γ and ξ.
For each $t = 1, \ldots, T_n$ and each $i = 1, \ldots, C$:

$$\gamma_t^{(n)}(i) \leftarrow \frac{\alpha_t^{(n)}(i)\beta_t^{(n)}(i)}{\sum_{m=1}^{C} \alpha_t^{(n)}(m)\beta_t^{(n)}(m)}.$$

For each $t = 2, \ldots, T_n$ and each (i, j):

$$\xi_t^{(n)}(i,j) \leftarrow \frac{\alpha_{t-1}^{(n)}(j)\, A_{ij}^{(k)}\, b_i^{(k)}\left(y_t^{(n)}\right)\, \beta_t^{(n)}(i)}{\sum_{m=1}^{C} \sum_{\ell=1}^{C} \alpha_{t-1}^{(n)}(m)\, A_{\ell m}^{(k)}\, b_\ell^{(k)}\left(y_t^{(n)}\right)\, \beta_t^{(n)}(\ell)}.$$

Worked example — EM for HMM: the Baum–Welch Algorithm (continued)

(4) Accumulate expected counts (sufficient statistics).
Initial-state expected counts:

$$\widehat{\pi}_i \leftarrow \widehat{\pi}_i + \gamma_1^{(n)}(i).$$

Transition expected counts:

$$\widehat{A}_{ij} \leftarrow \widehat{A}_{ij} + \sum_{t=2}^{T_n} \xi_t^{(n)}(i,j), \qquad \widehat{A}_{\cdot j} \leftarrow \widehat{A}_{\cdot j} + \sum_{t=2}^{T_n} \gamma_{t-1}^{(n)}(j).$$

Emission expected counts (multinomial emissions):

$$\widehat{B}_{hi} \leftarrow \widehat{B}_{hi} + \sum_{t=1}^{T_n} \gamma_t^{(n)}(i)\, \mathbb{I}\left[y_t^{(n)} = h\right], \qquad \widehat{B}_{\cdot i} \leftarrow \widehat{B}_{\cdot i} + \sum_{t=1}^{T_n} \gamma_t^{(n)}(i).$$

M-step: re-estimate parameters by normalized expected counts.
Update initial distribution:

$$\pi_i^{(k+1)} \leftarrow \frac{\widehat{\pi}_i}{\sum_{m=1}^{C} \widehat{\pi}_m} \quad (\text{note: } \sum_m \widehat{\pi}_m = N).$$

Update transitions (for each j):

$$A_{ij}^{(k+1)} \leftarrow \frac{\widehat{A}_{ij}}{\widehat{A}_{\cdot j}}.$$

Update emissions (for each i):

$$B_{hi}^{(k+1)} \leftarrow \frac{\widehat{B}_{hi}}{\widehat{B}_{\cdot i}}.$$

Convergence check.
Compute $\ell^{(k+1)}$ by re-running only the forward termination step (or store it if computed during the E-step after updates). If $\left|\ell^{(k+1)} - \ell^{(k)}\right| \le \varepsilon$, stop. Otherwise set $k \leftarrow k+1$ and continue.

Return $\theta^{(k+1)}$.

Implementation notes. For numerical stability, it is standard to use either (i) per-time-step scaling constants in the forward/backward recursions and accumulate the log-likelihood via the scaling factors, or (ii) log-space computation with log-sum-exp. Multiple random initializations are recommended because EM can converge to local optima.