Hidden Markov Models

Wha Sook Jeon

Mobile Computing & Communications Lab.

HMM Basics

- A hidden Markov model is a doubly stochastic process
  - An underlying Markov process
    - not observable
    - can only be observed through another observation process
  - An observation process that produced a sequence of observations
- A hidden Markov model is usually defined as five-tuples (S, Ω, P, Φ, Π)
  - $S = \{s_1, s_2, \cdots, s_N\}$ is a state space of the underlying process
  - $\Omega = \{o_1, o_2, \cdots, o_M\}$ is a set of possible observations
  - $P = [p_{ij}]$ where p_{ij} is the state transition probability from s_i to s_j
  - $\Phi = [\phi_j(o_k)]$ where $\phi_j(o_k)$ is the probability that o_k is produced in state s_j
  - $\Pi = [\pi_j]$ is the initial state distribution
- Parameter of an HMM: $\lambda = (P, \Phi, \Pi)$

HMM Assumptions

- q_t, o_t : the hidden state and the observation at time t
- Markov assumption
  - $P(q_{t+1} = j \mid q_t = i, q_{t-1} = l, \cdots, q_0 = n) = P(q_{t+1} = j \mid q_t = i)$
- Time-homogeneous assumption
  - $p_{ij} = P(q_{t+1} = j \mid q_t = i) = P(q_{m+1} = j \mid q_m = i)$
- Observation independence assumption
  - $P(o_1, o_2, \cdots o_T \mid q_1, q_2, \cdots q_T, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda)$

$$\Downarrow$$

- Joint Probability distribution

$$P(Q, O) = \prod_{t=1}^{T} P(q_t \mid q_{t-1}) P(o_t \mid q_t)$$

Fundamental Problems in HMM

- Evaluation problem (likelihood computation)
  - Given $\lambda = (P, \Phi, \Pi)$ and an observation sequence $O = (o_1, o_2, \cdots, o_T)$ how do we efficiently compute $P(O \mid \lambda)$?

- Decoding problem
  - Given $\lambda = (P, \Phi, \Pi)$, what is the most likely sequence of hidden states that could have generated a given observation sequence?
  - $Q^* = \arg\max_Q P(Q, O \mid \lambda)$

- Learning problem
  - Given an observation sequence, find the parameters of the HMM that maximize the probability of a given observation sequence
  - $\lambda^* = \arg\max_\lambda P(O \mid \lambda)$

Solution Methods

- Evaluation problem
  - Forward algorithm
  - Backward algorithm
- Decoding problem
  - Viterbi algorithm
- Learning problem
  - Baum-Welch algorithm (Backward-Forward algorithm)

Evaluation Problem (1)

- $$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(Q \mid \lambda)$$

  where $P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t) = \phi_{q_1}(o_1)\phi_{q_2}(o_2) \cdots \phi_{q_T}(o_T)$

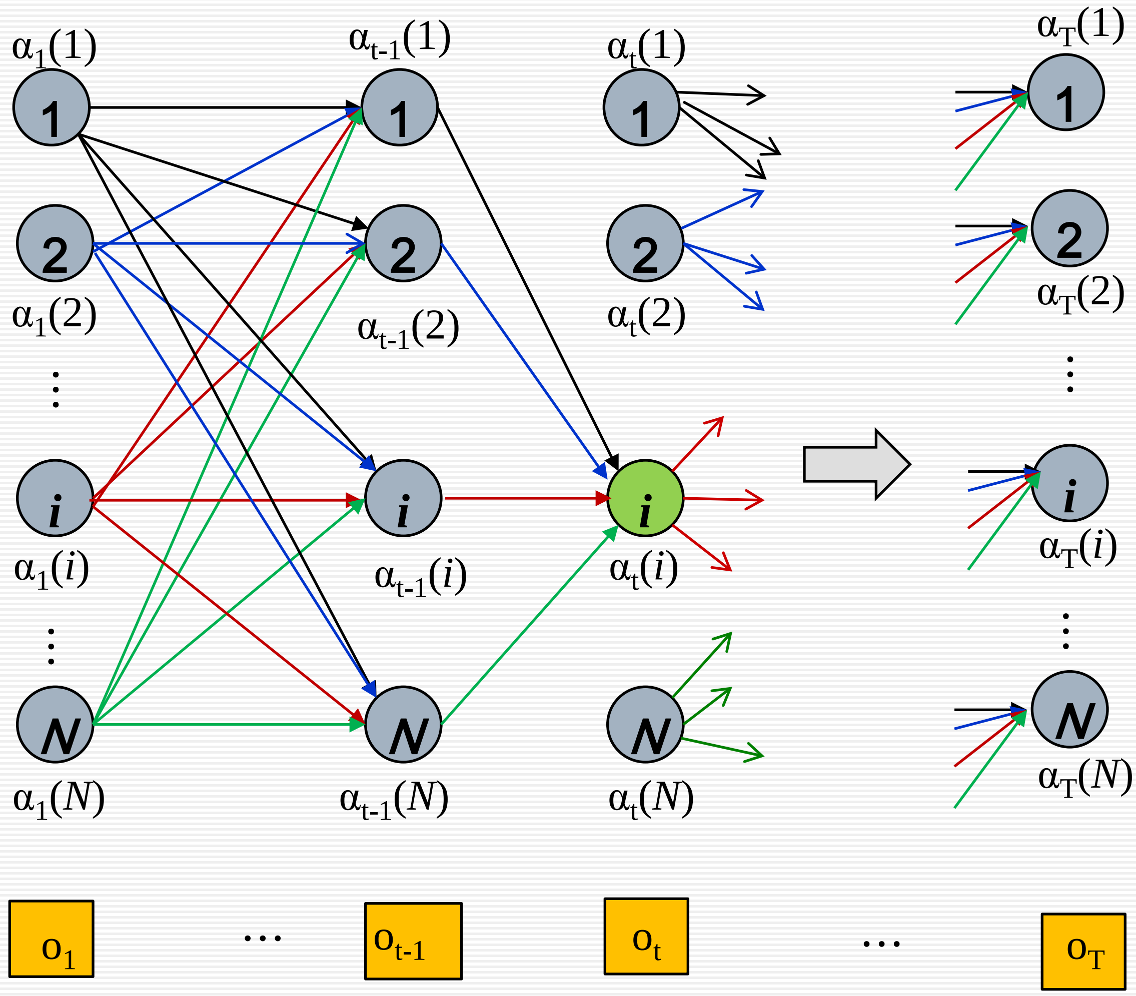
  $P(Q \mid \lambda) = \pi_{q_1} p_{q_1 q_2} p_{q_2 q_3} \cdots p_{q_{T-1} q_T}$

$$\boxed{P(O \mid \lambda) = \sum_{q_1 \cdots q_T} \pi_{q_1} \phi_{q_1}(o_1) p_{q_1 q_2} \phi_{q_2}(o_2) p_{q_2 q_3} \cdots p_{q_{T-1} q_T} \phi_{q_T}(o_T)}$$

- Forward Algorithm

$$\begin{aligned}
\alpha_t(i) &= P(o_1, o_2, \cdots, o_t, q_t = i) \\
&= P(o_t \mid o_1, o_2, \cdots, o_{t-1}, q_t = i) P(o_1, o_2, \cdots, o_{t-1}, q_t = i) \\
&= P(o_t \mid q_t = i) P(o_1, o_2, \cdots, o_{t-1}, q_t = i) \\
&= \phi_i(o_t) \sum_{j \in S} P(q_t = i \mid q_{t-1} = j) P(o_1, o_2, \cdots, o_{t-1}, q_{t-1} = j) \\
&= \phi_i(o_t) \sum_{j=1}^{N} p_{ji} \alpha_{t-1}(j)
\end{aligned}$$

Forward Algorithm



Evaluation Problem (2)

- Forward Algorithm

1. Initialization

$$\alpha_1(i) = \pi_i \, \phi_i(o_1) \qquad 1 \le i \le N$$

2. Induction

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} p_{ji}\alpha_t(j)\right)\phi_i(o_{t+1}) \qquad 1 \le i \le N, 1 \le t \le T-1$$

3. Set $t = t+1$. If $t < T$, go to step 2; otherwise go to step 4

4. Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_T = i) = \sum_{i=1}^{N} \alpha_T(i)$$

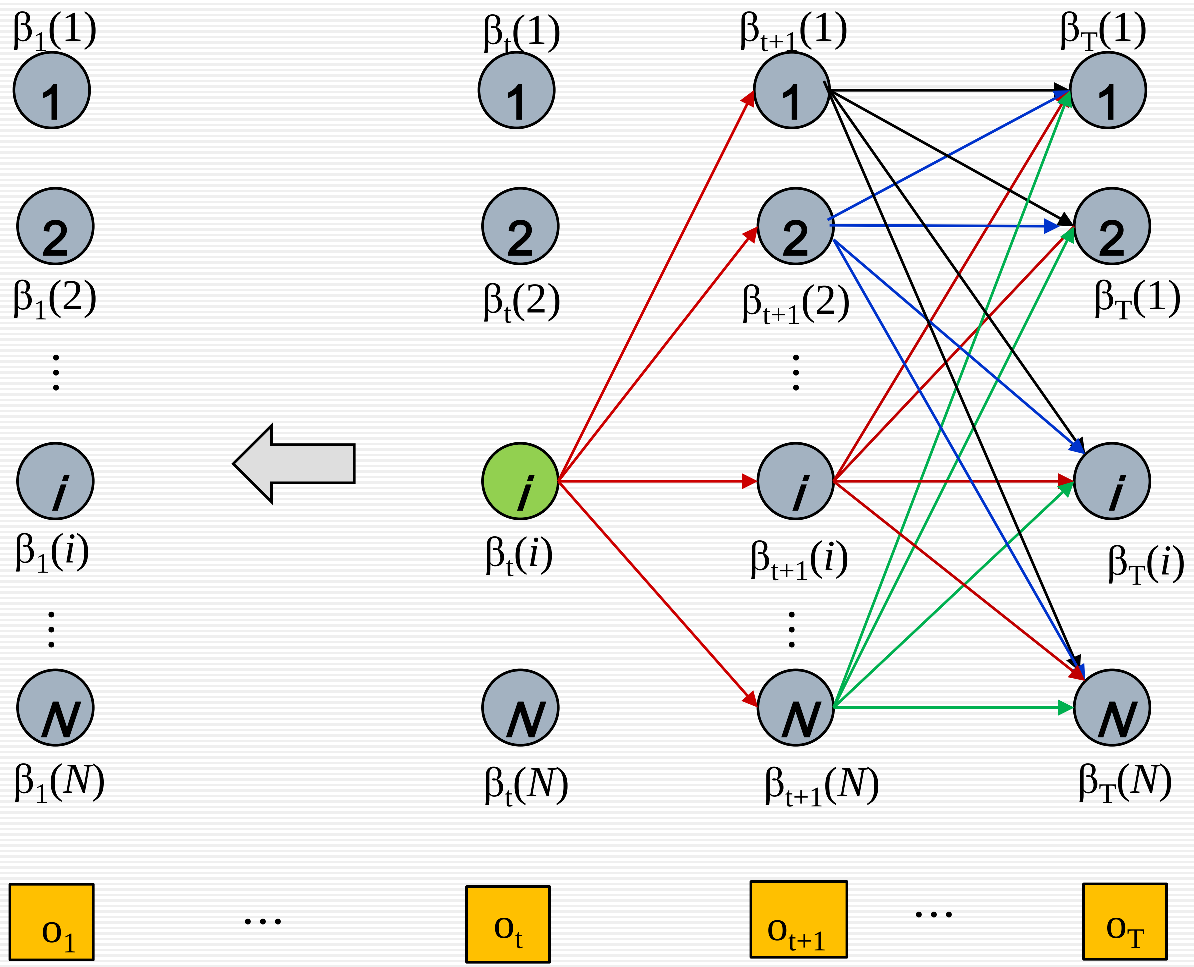
Evaluation Problem (3)

- Backward Algorithm
  - $\beta_t(i) = P(o_{t+1}, o_{t+2}, \cdots, o_T \mid q_t = i)$

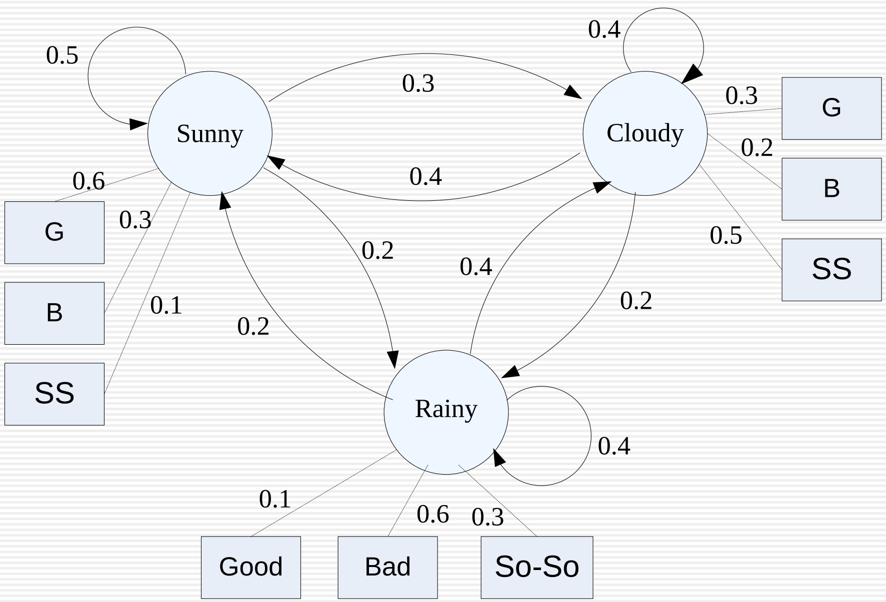
$$
\begin{aligned}
&= \sum_{j \in S} P(o_{t+1}, o_{t+2}, \cdots, o_T, q_{t+1} = j \mid q_t = i) \\
&= \sum_{j \in S} P(o_{t+1} \mid q_{t+1} = j) P(o_{t+2}, \cdots, o_T, q_{t+1} = j \mid q_t = i) \\
&= \sum_{j \in S} \phi_j(o_{t+1}) P(o_{t+2}, \cdots, o_T \mid q_{t+1} = j) P(q_{t+1} = j \mid q_t = i) \\
&= \sum_{j=1}^{N} \phi_j(o_{t+1}) \beta_{t+1}(j) p_{ij}
\end{aligned}
$$

1. Initialization: $\beta_T(i) = 1 \qquad 1 \le i \le N$
2. Induction: $\beta_t(i) = \sum_{j=1}^{N} p_{ij} \phi_j(o_{t+1}) \beta_{t+1}(j) \qquad 1 \le i \le N, \quad T-1 \ge t \ge 1$
3. Set $t = t$-1. If $t > 0$, go to step 2; otherwise, go to step 4
4. Termination: $P(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \pi_i \phi_i(o_1)$

Backward Algorithm



Example: Forward Algorithm (1)

- 

- $P(O = (G, G, SS, B, B) \mid \lambda)$
  $- \lambda: \quad \pi_S = \pi_C = \pi_R = 1/3,$ diagram

Example: Forward Algorithm (2)

- $\alpha_1(S) = \pi_S \phi_S(G) = 1/3 \times 0.6 = 0.2$
  $\alpha_1(C) = \pi_C \phi_C(G) = 1/3 \times 0.3 = 0.1$
  $\alpha_1(R) = \pi_R \phi_R(G) = 1/3 \times 0.1 = 0.033$
- $\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} p_{ji}\alpha_t(j)\right)\phi_i(o_{t+1}) \qquad 1 \le i \le N, \quad 1 \le t \le T-1$
  - $\alpha_2(S) = \left(p_{SS}\alpha_1(S) + p_{CS}\alpha_1(C) + p_{RS}\alpha_1(R)\right)\phi_S(G)$
    $= (0.5 \times 0.2 + 0.4 \times 0.1 + 0.2 \times 0.033) \times 0.6 = 0.088$
    $\alpha_2(C) = \left(p_{SC}\alpha_1(S) + p_{CC}\alpha_1(C) + p_{RC}\alpha_1(R)\right)\phi_C(G) = 0.034$
    $\alpha_2(R) = \left(p_{SR}\alpha_1(S) + p_{CR}\alpha_1(C) + p_{RR}\alpha_1(R)\right)\phi_R(G) = 0.007$
  - $\alpha_3(S) = \left(p_{SS}\alpha_2(S) + p_{CS}\alpha_2(C) + p_{RS}\alpha_2(R)\right)\phi_S(SS) = 0.018$
    $\alpha_3(C) = 0.021 \quad \alpha_3(R) = 0.008$
  - $\alpha_4(S) = 0.002 \quad \alpha_4(C) = 0.003 \quad \alpha_4(R) = 0.007$
  - $\alpha_5(S) = 0.0004 \quad \alpha_5(C) = 0.0009 \quad \alpha_5(R) = 0.0023$
- $P(O = (G, G, SS, B, B) \mid \lambda) = \alpha_5(S) + \alpha_5(C) + \alpha_5(R) = 0.0036$

Decoding Problem

- Viterbi Algorithm (Similar to Forward Algorithm)

1. Initialization

$$\alpha_1(i) = \pi_i \, \phi_i(o_1) \qquad 1 \le i \le N$$

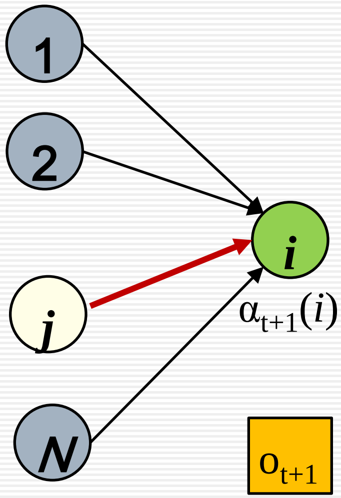
2. Induction $\quad (1 \le i \le N, \quad 1 \le t < T)$

$$\alpha_{t+1}(i) = \max_{\text{all } j} \alpha_t(j)\, p_{ji}\, \phi_i(o_{t+1})$$
$$b_{t+1}(i) = \underset{\text{all } j}{\text{argmax}}\, \alpha_t(j)\, p_{ji}\, \phi_i(o_{t+1})$$

3. Set $t = t+1$. If $t < T$, go to step 2; otherwise go to step 4

4. Termination

$$\alpha^*_T = \max_{\text{all } j} \alpha_T(j) \qquad b^*_T = \underset{\text{all } j}{\text{argmax}}\, \alpha_T(j)$$



Learning Problem

- $\lambda^* = \arg\max_\lambda P(O \mid \lambda)$
- There is no known method to analytically obtain λ that maximizes $P(O \mid \lambda)$
- **Baum-Welch Algorithm**
  - Iterative algorithm for choosing the model parameters in such a way that $P(O \mid \lambda)$ is locally maximized.
  - A special case of the Expectation Maximization method

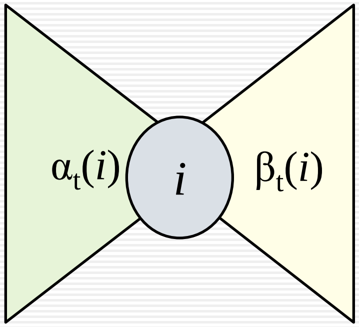
    - $p_{ij} = P(q_{t+1} = j \mid q_t = i) = \dfrac{P(q_t = i, q_{t+1} = j)}{P(q_t = i)}$

      $\Rightarrow \quad \bar{p}_{ij} = \dfrac{\sum_{t=0}^{T-1} P(q_t = i, q_{t+1} = j \mid O)}{\sum_{t=0}^{T-1} P(q_t = i \mid O)} = \dfrac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \gamma_t(i)}$

    - $\phi_j(k) = P(o_t = k \mid q_t = j) = \dfrac{P(o_t = k, q_t = j)}{P(q_t = j)} \quad \Rightarrow \quad \bar{\phi}_j(k) = \dfrac{\sum_{t=1, o_t = k}^{T-1} \gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(j)}$

    - We need $\xi_t(i,j)$ and $\gamma_t(i)$

Baum-Welch algorithm (1)



$$
\begin{aligned}
\gamma_t(i) &= P(q_t = i \mid O) \\
&= \frac{P(q_t = i, o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)}{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)} \\
&= \frac{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T \mid q_t = i) P(q_t = i)}{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)} \\
&= \frac{P(o_1, \cdots, o_t \mid o_{t+1}, \cdots, o_T, q_t = i) P(o_{t+1}, \cdots, o_T \mid q_t = i) P(q_t = i)}{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)} \\
&= \frac{P(o_1, \cdots, o_t \mid q_t = i) P(o_{t+1}, \cdots, o_T \mid q_t = i) P(q_t = i)}{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)} \\
&= \frac{P(o_1, \cdots, o_t, q_t = i) P(o_{t+1}, \cdots, o_T \mid q_t = i)}{P(o_1, \cdots, o_t, o_{t+1}, \cdots, o_T)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} P(o_1, \cdots, o_t, q_t = i) P(o_{t+1}, \cdots, o_T \mid q_t = i)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\end{aligned}
$$

Baum-Welch algorithm (2)

- $$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O) = \frac{P(q_t = i, q_{t+1} = j, O)}{P(O)} = \frac{\alpha_t(i)\, p_{ij}\phi_j(t+1)\beta_{t+1}(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} = \frac{\alpha_t(i)\, p_{ij}\phi_j(t+1)\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\, p_{ij}\phi_j(t+1)\beta_{t+1}(j)}$$

- $$\overline{p}_{ij} = \frac{\sum_{t=0}^{T-1} P(q_t = i, q_{t+1} = j \mid O)}{\sum_{t=0}^{T-1} P(q_t = i \mid O)} = \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \gamma_t(i)}$$

- $$\bar{\phi}_j(k) = \frac{\sum_{t=1}^{T} p(o_t = k, q_t = j \mid O)}{\sum_{t=1}^{T} p(q_t = j \mid O)} = \frac{\sum_{t=1, o_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Baum-Welch algorithm (3)

- Forward-backward algorithm

$$\gamma_t(i) = P(q_t = i \mid O) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j \mid O) = \frac{\alpha_t(i)\, p_{ij}\phi_j(t+1)\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

$$\alpha_1(i) = \pi_i\, \phi_i(o_1) \qquad 1 \le i \le N$$

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} p_{ji}\alpha_t(j)\right)\phi_i(o_{t+1}) \qquad 1 \le i \le N, \quad 1 \le t \le T-1$$

$$\beta_T(i) = 1 \qquad 1 \le i \le N$$

$$\beta_t(i) = \sum_{j=1}^{N} p_{ij}\phi_j(o_{t+1})\beta_{t+1}(j) \qquad 1 \le i \le N, \quad T-1 \ge t \ge 1$$

Baum-Welch algorithm (4)

- The algorithm starts by setting the parameters $\lambda = (P, \Phi, \Pi)$ to some initial values that can be chosen from some prior knowledge or from some uniform distribution
- Detailed Procedure

1. Setting an initial parameters: λ
   - Obtain the estimate of the initial state distribution for state i as the expected frequency with which state i is visited at time $t = 1$: $\bar{\pi}_i$
   - Obtain the estimates $\bar{p}_{ij}$ and $\bar{\phi}_j(k)$
2. Let the current model be $\lambda = (P, \Phi, \Pi)$ and compute $\bar{p}_{ij}$ and $\bar{\phi}_j(k)$

   Let the re-estimated model be $\bar{\lambda} = (\bar{P}, \bar{\Phi}, \bar{\Pi})$.
3. If $P(O \mid \bar{\lambda}) - P(O \mid \lambda) < \delta$, stop, where δ is a predefined threshold value. Otherwise, we go to step 2 (a new iteration) by using the updated model.