Multimodal Speech Recognition with Hidden Markov Model

ECE417 – Multimedia Signal Processing

Spring2015

Multimodal Digit Recognition with HMM

- Data: spoken digit video clips
  - You do NOT need to do feature extraction
  - Audio feature: MFCC
  - Visual feature: lips tracking
- Task:
  - Train HMMs for the spoken digits “2” and “5”
  - Use HMMs to do maximum likelihood recognition
    - Audio
    - Video
    - Audio-visual [concatenate the audio and visual feature]

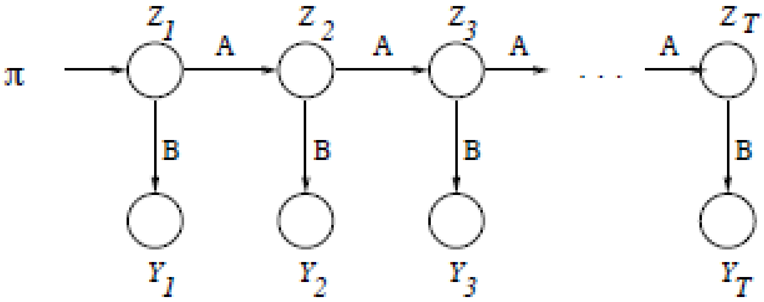
Hidden Markov Model (HMM)

$X = (Y, Z)$, where Z is unobserved data and Y is the observed data

$Z = (Z_1, \ldots, Z_T)$ is a time-homogeneous Markov process, with one-step transition probability matrix $A = (a_{i,j})$, and with Z_1 having the initial distribution π. Here, T, with $T \geq 1$, denotes the total number of observation times. The state-space of Z is denoted by $\mathcal{S}$, and the number of states of $\mathcal{S}$ is denoted by N_s.

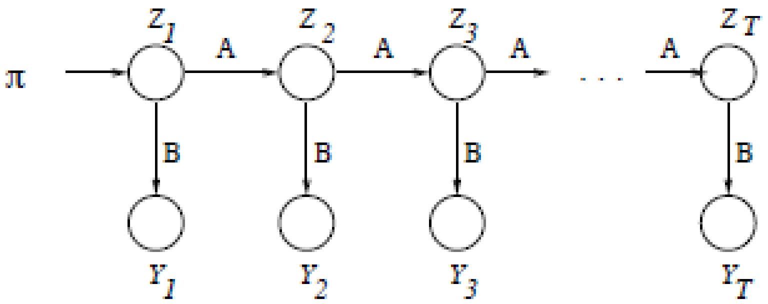
$Y = (Y_1, \ldots, Y_n)$ is the observed data. It is such that given $Z = z$, for some $z = (z_1, \ldots, z_n)$, the variables $Y_1, \cdots, Y_n$ are conditionally independent with $P(Y_t = l | Z = z) = b_{z_t,l}$, for a given observation generation matrix $B = (b_{i,l})$. The observations are assumed to take values in a set of size N_o, so that B is an $N_s \times N_o$ matrix and each row of B is a probability vector.

The parameter for this model is $\theta = (\pi, A, B)$.



Structure of hidden Markov model.

Estimation Tasks for HMM



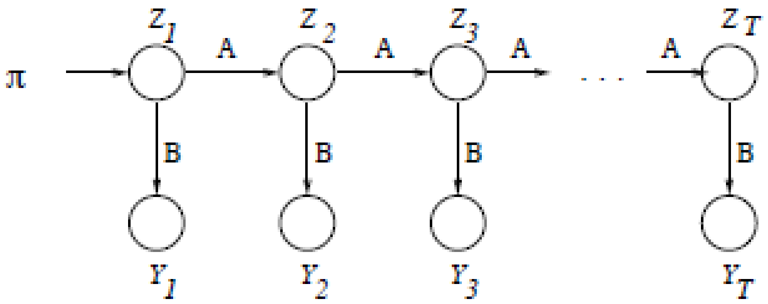
Structure of hidden Markov model.

1. Given the observed data and θ, compute the conditional distribution of the state (solved by the forward-backward algorithm)

2. Given the observed data and θ, compute the most likely sequence for hidden states (solved by the Viterbi algorithm)

3. Given the observed data, compute the maximum likelihood (ML) estimate of θ (solved by the Baum-Welch/EM algorithm).

Posterior State Probabilities



Structure of hidden Markov model.

$$
\begin{aligned}
\widehat{Z}_{t|t_{MAP}} &= \arg\max_{i \in \mathcal{S}} P(Z_t = i | Y_1 = y_1, \ldots, Y_t = y_t, \theta) \\
\widehat{Z}_{t|T_{MAP}} &= \arg\max_{i \in \mathcal{S}} P(Z_t = i | Y_1 = y_1, \ldots, Y_T = y_T, \theta) \\
(\widehat{Z_t, Z_{t+1}})_{|T_{MAP}} &= \arg\max_{(i,j) \in \mathcal{S} \times \mathcal{S}} P(Z_t = i, Z_{t+1} = j | Y_1 = y_1, \ldots, Y_T = y_T, \theta)
\end{aligned}
$$

Forward-Backward Algorithm (1)

$$\alpha_i(t) \overset{\triangle}{=} P(Y_1 = y_1, \cdots, Y_t = y_t, Z_t = i|\theta),$$

$$\begin{aligned}
\alpha_j(t+1) &= \sum_{i \in \mathcal{S}} P(Y_1 = y_1, \cdots, Y_{t+1} = y_{t+1}, Z_t = i, Z_{t+1} = j|\theta) \\
&= \sum_{i \in \mathcal{S}} P(Y_1 = y_1, \cdots, Y_t = y_t, Z_t = i|\theta) \\
&\qquad \cdot P(Z_{t+1} = j, Y_{t+1} = y_{t+1}|Y_1 = y_1, \cdots, Y_t = y_t, Z_t = i, \theta) \\
&= \sum_{i \in \mathcal{S}} \alpha_i(t) a_{ij} b_{j y_{t+1}}.
\end{aligned}$$

$$\begin{aligned}
P(Z_t = i|Y_1 = y_1, \ldots, Y_t = y_t, \theta) &= \frac{P(Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t|\theta)}{P(Y_1 = y_1, \ldots, Y_t = y_t|\theta)} \\
&= \frac{\alpha_i(t)}{\sum_{j \in \mathcal{S}} \alpha_j(t)}
\end{aligned}$$

Forward-Backward Algorithm (2)

$$\beta_i(t) \overset{\triangle}{=} P(Y_{t+1} = y_{t+1}, \cdots, Y_T = y_T | Z_t = i, \theta),$$

$$\begin{aligned}
\beta_i(t-1) &= \sum_{j \in \mathcal{S}} P(Y_t = y_t, \cdots, Y_T = y_T, Z_t = j | Z_{t-1} = i, \theta) \\
&= \sum_{j \in \mathcal{S}} P(Y_t = y_t, Z_t = j | Z_{t-1} = i, \theta) \\
&\qquad \cdot P(Y_{t+1} = y_t, \cdots, Y_T = y_T, | Z_t = j, Y_t = y_t, Z_{t-1} = i, \theta) \\
&= \sum_{j \in \mathcal{S}} a_{ij} b_{j y_t} \beta_j(t).
\end{aligned}$$

$$\begin{aligned}
P(Z_t = i, Y_1 = y_1, \ldots, Y_T = y_T | \theta) &= P(Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t | \theta) \\
&\qquad \cdot P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T | \theta, Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t) \\
&= P(Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t | \theta) \\
&\qquad \cdot P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T | \theta, Z_t = i) \\
&= \alpha_i(t) \beta_i(t)
\end{aligned}$$

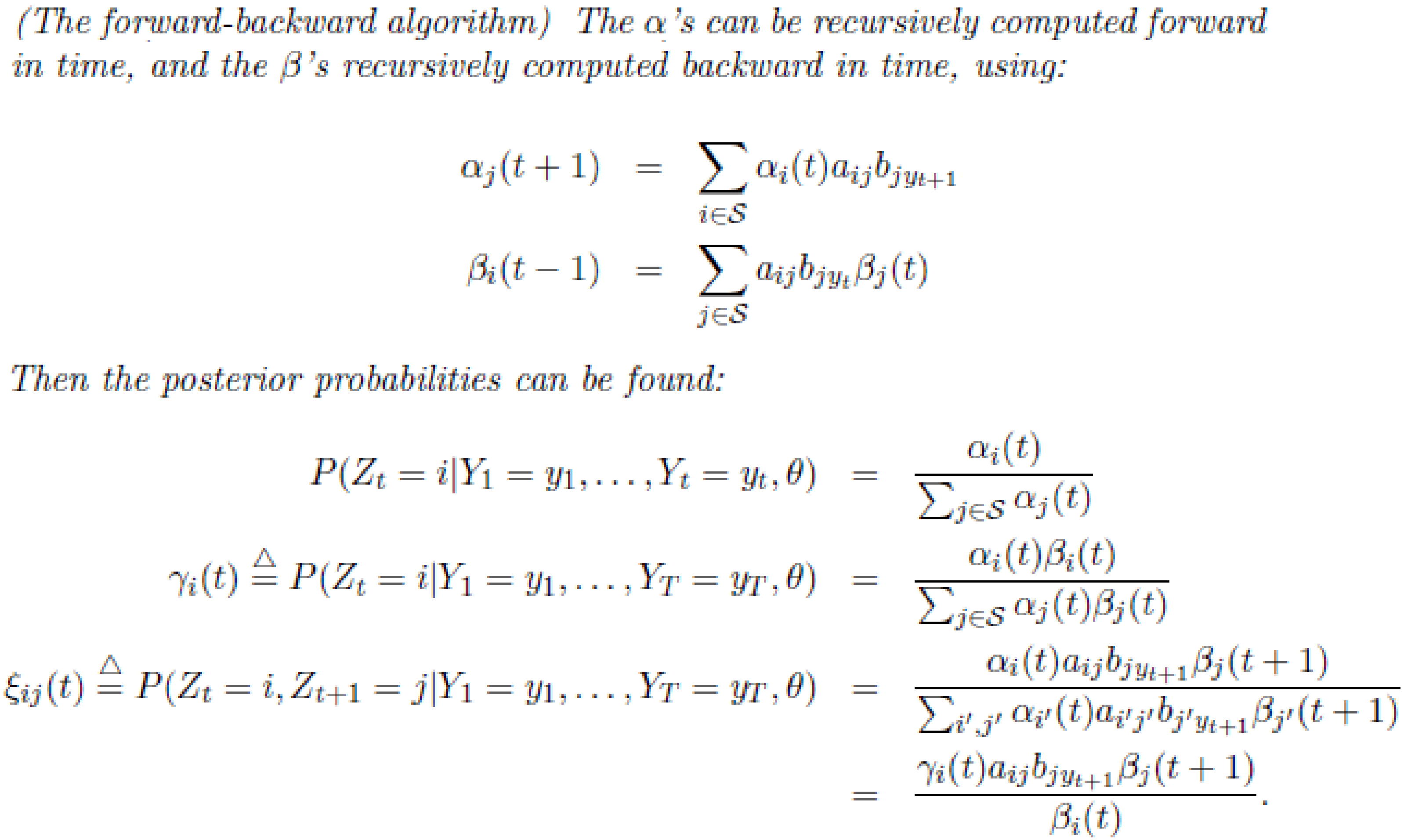
$$\begin{aligned}
\gamma_i(t) &\overset{\triangle}{=} P(Z_t = i | Y_1 = y_1, \ldots, Y_T = y_T, \theta) \\
&= \frac{P(Z_t = i, Y_1 = y_1, \ldots, Y_T = y_T | \theta)}{P(Y_1 = y_1, \ldots, Y_T = y_T | \theta)} \\
&= \frac{\alpha_i(t) \beta_i(t)}{\sum_{j \in \mathcal{S}} \alpha_j(t) \beta_j(t)}
\end{aligned}$$

Forward-Backward Algorithm (3)

$$\begin{aligned}
&P(Z_t = i, Z_{t+1} = j, Y_1 = y_1, \ldots, Y_T = y_T | \theta) \\
&\quad = \quad P(Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t | \theta) \\
&\qquad\qquad \cdot P(Z_{t+1} = j, Y_{t+1} = y_{t+1} | \theta, Z_t = i, Y_1 = y_1, \ldots, Y_t = y_t) \\
&\qquad\qquad\quad \cdot P(Y_{t+2} = y_{t+2}, \ldots, Y_T = y_T | \theta, Z_t = i, Z_{t+1} = j, Y_1 = y_1, \ldots, Y_{t+1} = y_{t+1}) \\
&\quad = \quad \alpha_i(t) a_{ij} b_{j y_{t+1}} \beta_j(t+1),
\end{aligned}$$

$$\begin{aligned}
\xi_{ij}(t) \quad &\overset{\triangle}{=} \quad P(Z_t = i, Z_{t+1} = j | Y_1 = y_1, \ldots, Y_T = y_T, \theta) \\
&= \quad \frac{P(Z_t = i, Z_{t+1} = j, Y_1 = y_1, \ldots, Y_T = y_T | \theta)}{P(Y_1 = y_1, \ldots, Y_T = y_T | \theta)} \\
&= \quad \frac{\alpha_i(t) a_{ij} b_{j y_{t+1}} \beta_j(t+1)}{\sum_{i', j'} \alpha_{i'}(t) a_{i' j'} b_{j' y_{t+1}} \beta_{j'}(t+1)} \\
&= \quad \frac{\gamma_i(t) a_{ij} b_{j y_{t+1}} \beta_j(t+1)}{\beta_i(t)}
\end{aligned}$$

Forward-Backward Algorithm



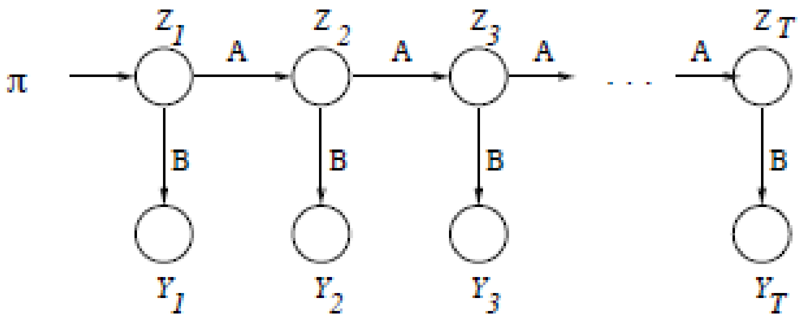
(The forward-backward algorithm) The α's can be recursively computed forward in time, and the β's recursively computed backward in time, using:

$$\begin{aligned}
\alpha_j(t+1) &= \sum_{i \in \mathcal{S}} \alpha_i(t) a_{ij} b_{j y_{t+1}} \\
\beta_i(t-1) &= \sum_{j \in \mathcal{S}} a_{ij} b_{j y_t} \beta_j(t)
\end{aligned}$$

Then the posterior probabilities can be found:

$$\begin{aligned}
P(Z_t = i | Y_1 = y_1, \ldots, Y_t = y_t, \theta) &= \frac{\alpha_i(t)}{\sum_{j \in \mathcal{S}} \alpha_j(t)} \\
\gamma_i(t) \triangleq P(Z_t = i | Y_1 = y_1, \ldots, Y_T = y_T, \theta) &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j \in \mathcal{S}} \alpha_j(t)\beta_j(t)} \\
\xi_{ij}(t) \triangleq P(Z_t = i, Z_{t+1} = j | Y_1 = y_1, \ldots, Y_T = y_T, \theta) &= \frac{\alpha_i(t) a_{ij} b_{j y_{t+1}} \beta_j(t+1)}{\sum_{i',j'} \alpha_{i'}(t) a_{i'j'} b_{j' y_{t+1}} \beta_{j'}(t+1)} \\
&= \frac{\gamma_i(t) a_{ij} b_{j y_{t+1}} \beta_j(t+1)}{\beta_i(t)}.
\end{aligned}$$

Most Likely State Sequence



Structure of hidden Markov model.

$$\widehat{Z}_{MAP}(y,\theta) = \arg\max_{z} p_{cd}(y,z|\theta)$$

$$\delta_i(t) \overset{\triangle}{=} \max_{(z_1,\dots,z_{t-1})\in\mathcal{S}^{t-1}} P(Z_1=z_1,\dots,Z_{t-1}=z_{t-1},Z_t=i,Y_1=y_1,\cdots,Y_t=y_t|\theta).$$

Viterbi Algorithm

$$
\begin{aligned}
\delta_j(t) &= \max_i \max_{\{z_1,\ldots,z_{t-2}\}} P(Z_1 = z_1, \ldots, Z_{t-2} = z_{t-2}, Z_{t-1} = i, Z_t = j, Y_1 = y_1, \cdots, Y_t = y_t|\theta) \\
&= \max_i \max_{\{z_1,\ldots,z_{t-2}\}} P(Z_1 = z_1, \ldots, Z_{t-2} = z_{t-2}, Z_{t-1} = i, Y_1 = y_1, \cdots, Y_{t-1} = y_{t-1}|\theta) a_{i,j} b_{jy_t} \\
&= \max_i \{\delta_i(t-1) a_{i,j} b_{jy_t}\}
\end{aligned}
$$

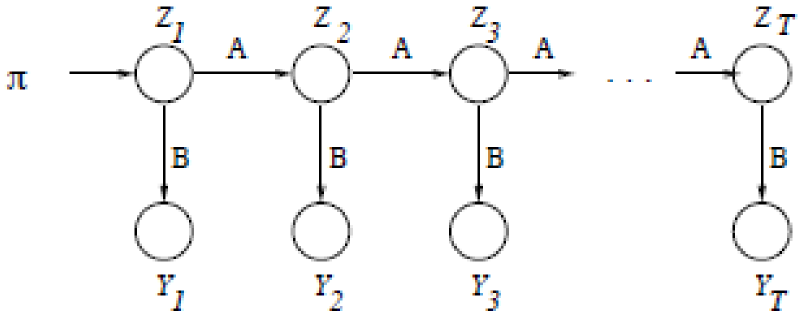
(Viterbi algorithm) Compute the δ's and associated back pointers by a recursion forward in time:

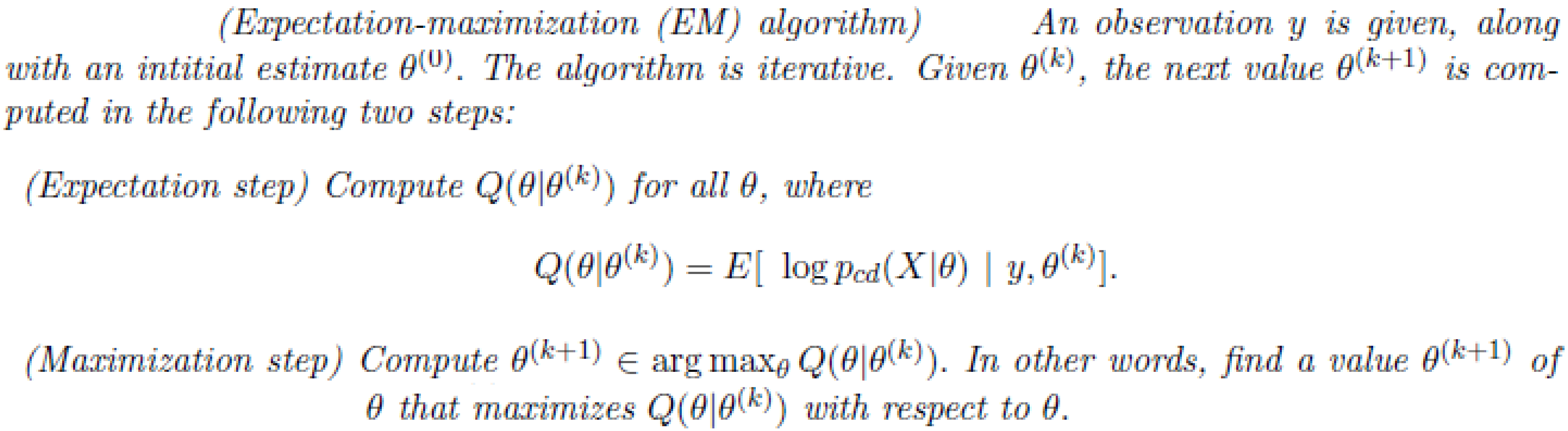
$$
\begin{aligned}
\textit{(initial condition)} \quad \delta_i(1) &= \pi(i) b_{iy_1} \\
\textit{(recursive step)} \quad \delta_j(t) &= \max_i \{\delta_i(t-1) a_{ij} b_{j,y_t}\} \\
\textit{(storage of back pointers)} \quad \phi_j(t) &\overset{\triangle}{=} \arg\max_i \{\delta_i(t-1) a_{i,j} b_{j,y_t}\}
\end{aligned}
$$

Then $z^* = \widehat{Z}_{MAP}(y, \theta)$ *satisfies* $p_{cd}(y, z^*|\theta) = \max_i \delta_i(T)$, *and* z^* *is given by tracing backward in time:*

$$
z^*_T = \arg\max_i \delta_i(T) \quad \textit{and} \quad z^*_{t-1} = \phi_{z^*_t}(t) \ \textit{for} \ 2 \le t \le T.
$$

EM Algorithm





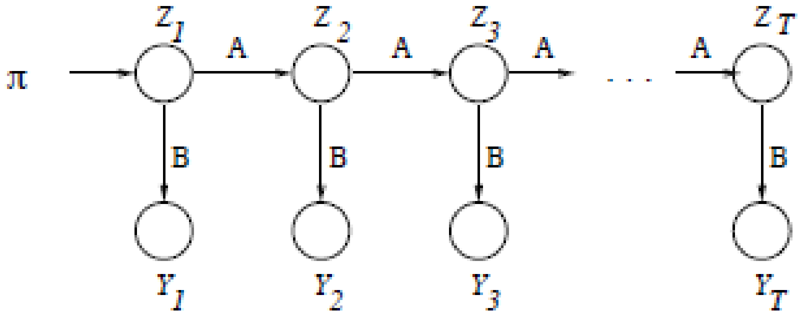
(Expectation-maximization (EM) algorithm) *An observation y is given, along with an intitial estimate $\theta^{(0)}$. The algorithm is iterative. Given $\theta^{(k)}$, the next value $\theta^{(k+1)}$ is computed in the following two steps:*

(Expectation step) Compute $Q(\theta|\theta^{(k)})$ for all θ, where

$$Q(\theta|\theta^{(k)}) = E[\ \log p_{cd}(X|\theta)\ |\ y, \theta^{(k)}].$$

(Maximization step) Compute $\theta^{(k+1)} \in \arg\max_\theta Q(\theta|\theta^{(k)})$. In other words, find a value $\theta^{(k+1)}$ of θ that maximizes $Q(\theta|\theta^{(k)})$ with respect to θ.

EM Algorithm for HMM

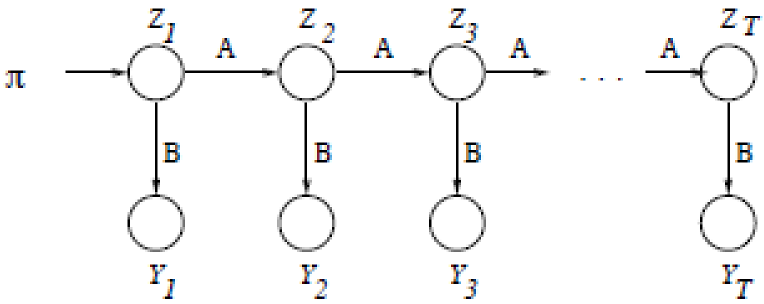


$$p_{cd}(y, z|\theta) = \pi_{z_1} \prod_{t=1}^{T-1} a_{z_t, z_{t+1}} \prod_{t=1}^{T} b_{z_t, y_t}.$$

$$\log p_{cd}(y, z|\theta) \quad = \quad \log \pi_{z_1} + \sum_{t=1}^{T-1} \log a_{z_t, z_{t+1}} + \sum_{t=1}^{T} \log b_{z_t, y_t}$$

$$\begin{aligned} Q(\theta|\theta^{(k)}) &= E[\log p_{cd}(y, Z|\theta)|y, \theta^{(k)}] \\ &= \sum_{i \in \mathcal{S}} \gamma_i(1) \log \pi_i + \sum_{t=1}^{T-1} \sum_{i,j} \xi_{ij}(t) \log a_{i,j} + \sum_{t=1}^{T} \sum_{i \in \mathcal{S}} \gamma_i(t) \log b_{i, y_t}, \end{aligned}$$

Estimation Tasks for HMM



Structure of hidden Markov model.

1. Given the observed data and θ, compute the conditional distribution of the state (solved by the forward-backward algorithm)
2. Given the observed data and θ, compute the most likely sequence for hidden states (solved by the Viterbi algorithm)
3. Given the observed data, compute the maximum likelihood (ML) estimate of θ (solved by the Baum-Welch/EM algorithm).

Evaluation: leave-one-out

- There are 10 sequences for “2” and 10 for “5”
- Using leave-one-out scheme
  - Each time you exclude one sequence from the training set for testing
  - Repeat this for 20 times, get the average accuracy

Matlab functions

- Train HMM ghmm_learn.m
    - [P0,A,mu,sigma] = ghmm_learn(Yseq,N,Ainit)
- Forward algorithm gmhmm_fwd.m
    - [alpha,scale] = gmihmm_fwd(Y,A,P0,mu,sigma)
- Backward algorithm ghmm_bwd .m
    - beta = ghmm_bwd(Y,A,P0,mu,sigma,scale)

1. DON'T HARD-CODE
2. Use Matlab's logical indexing functionality to it's fullest extent
3. Use meaningful variable names
4. COMMENT YOUR CODE
5. When doing arithmatic calculations, use Matlab's built in functions, and do operations as matrix vector calculations (MATLAB = MATrix LABoratory)
6. If you need to do a lot of nested, iterative calculations, it might be easier to do in C...look into MEX. That being said, you should not have to do this in this MP or any of the others we have done thus far.

Reference

- Rabiner, L., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. *77, no.* 2, pp. 257-286, 1989