

T-61.184

Automatic Speech Recognition: From Theory to Practice

`http://www.cis.hut.fi/Opinnot/T-61.184/`
October 4, 2004

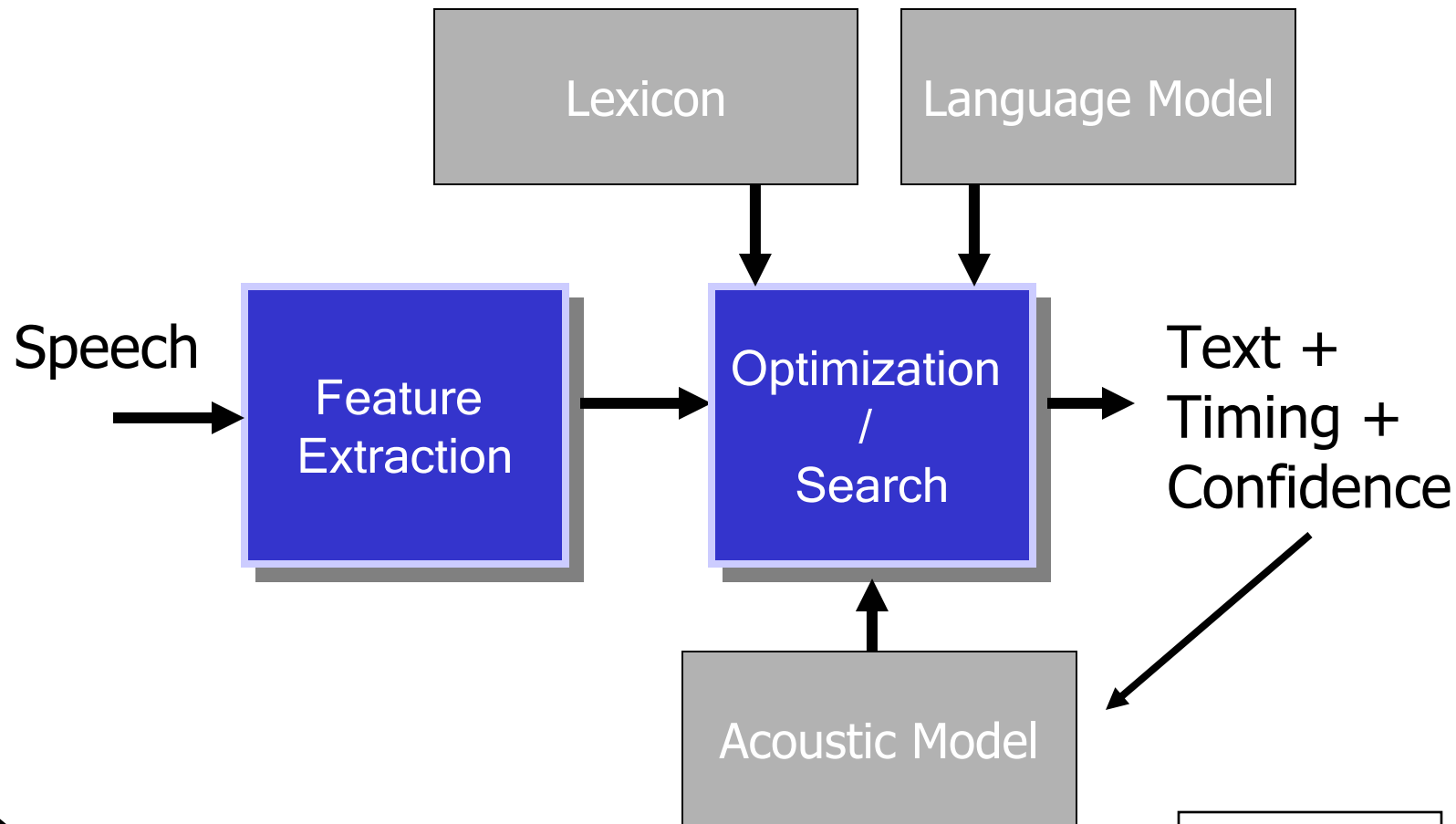
Prof. Bryan Pellom

Department of Computer Science
Center for Spoken Language Research
University of Colorado

`pellom@cslr.colorado.edu`

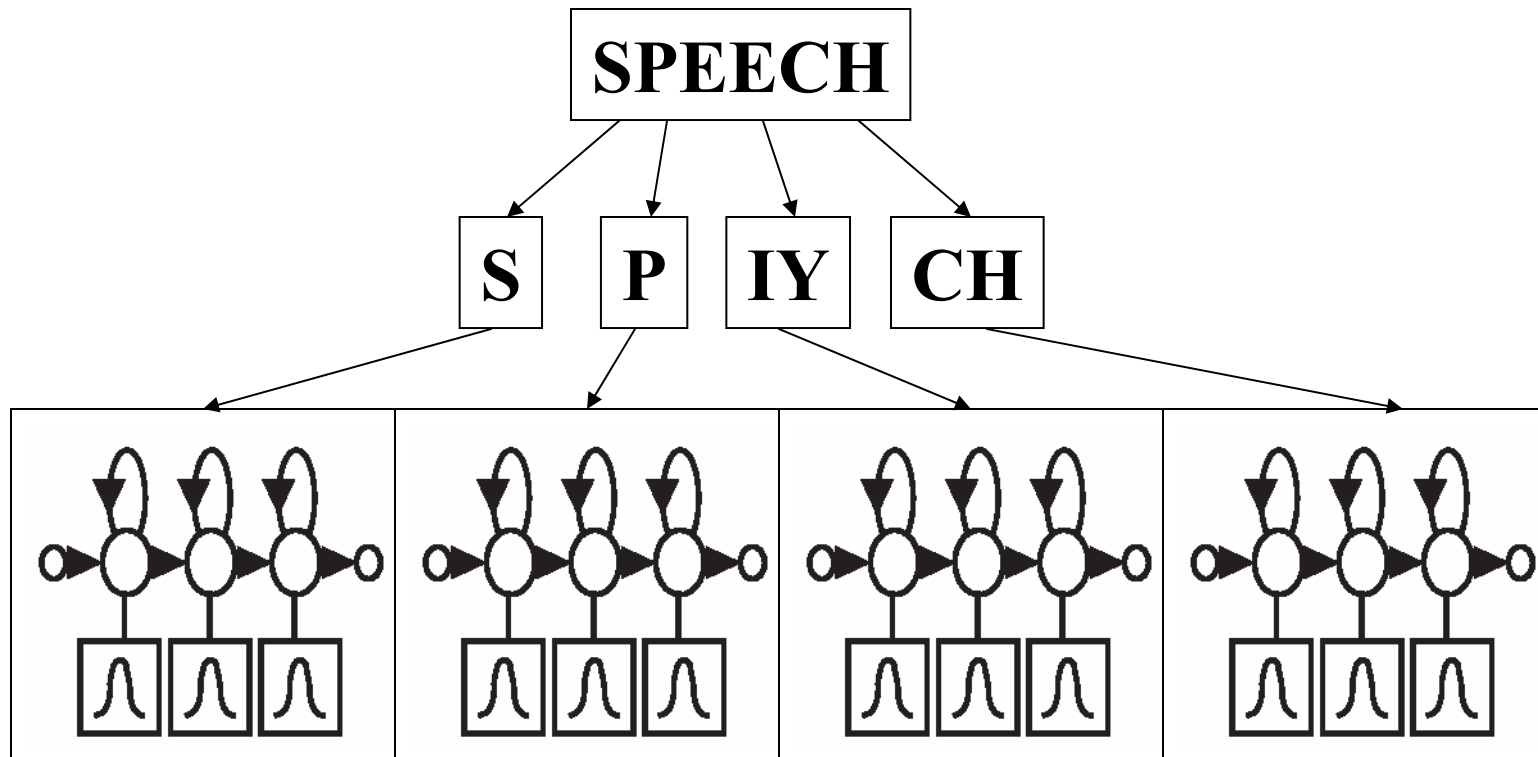
T-61.184

Role of Hidden Markov Models in Speech Recognition



T-61.184

Motivation for Today's Topic



T-61.184

Expectations for this course

- Today to present a basic idea of Hidden Markov Models (HMMs)
- Next week we'll discuss how HMMs are used in practice to model acoustics of speech
- Don't be lost in the math today: I don't expect you'll get the theory from all these slides! If you want to learn about HMMs it's important to take a look at the resources listed on the next slide.

Resources for Today's Topic

- L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” Proc. IEEE, Vol. 77, No. 2, pp. 257-286, February, 1989.
- L.R. Rabiner & B. W. Juang, Fundamentals of Speech Recognition, Prentice-Hall, ISBN 0-13 015157-2, 1993 (see chapter 6)
- The Cambridge HTK Toolkit has a user manual called the “HTK Book”. This is an excellent resource and covers many of the equations related to HMM modeling

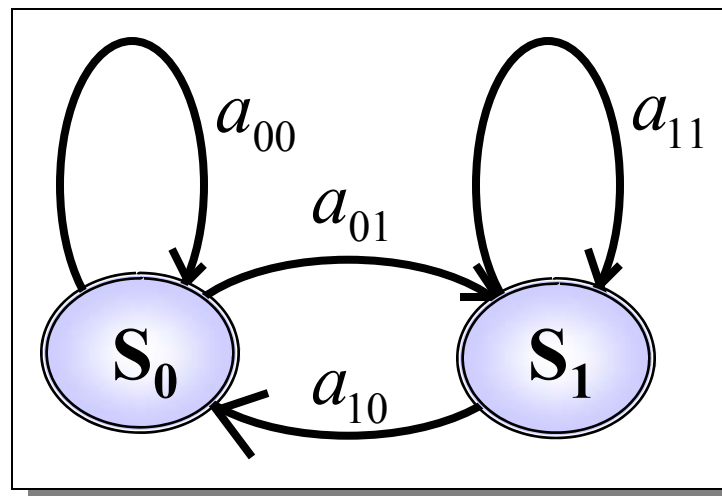
Discrete-Time Markov Process

■ Characterized by:

- A set of N states

$$S = \{S_0, S_1, \dots, S_N\}$$

- Transition Probabilities a_{ij}



■ First-order Markov Chain

- Current system state only depends on previous state
- Let q_t be the system state at time t.
- Transition probability depends only on previous state

$$P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k \dots) = P(q_t = S_j | q_{t-1} = S_i)$$

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$$

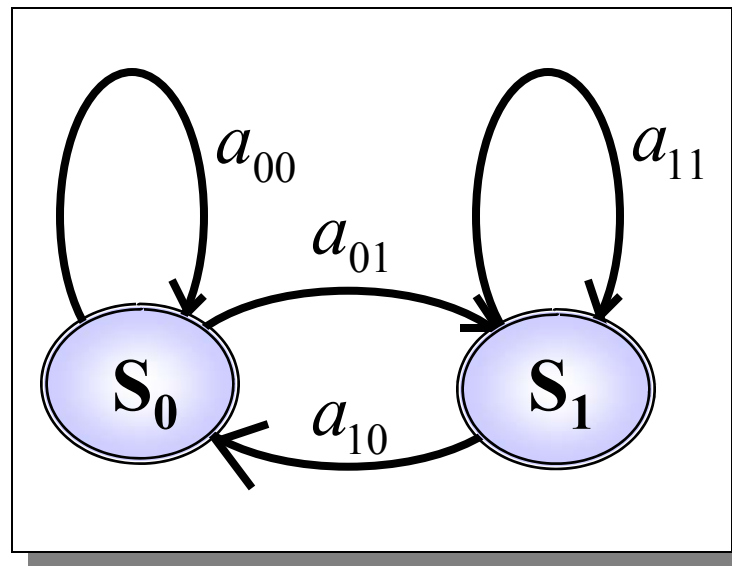
Discrete-Time Markov Process

- **Properties:**

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$$

$$a_{ij} \geq 0 \quad \forall i, j$$

$$\sum_{j=0}^N a_{ij} = 1 \quad \forall i$$



- **“Observable Markov Model”** : output of the process is a set of states.

Example 1: Single Fair Coin Observable Markov Process

- **Outcomes**

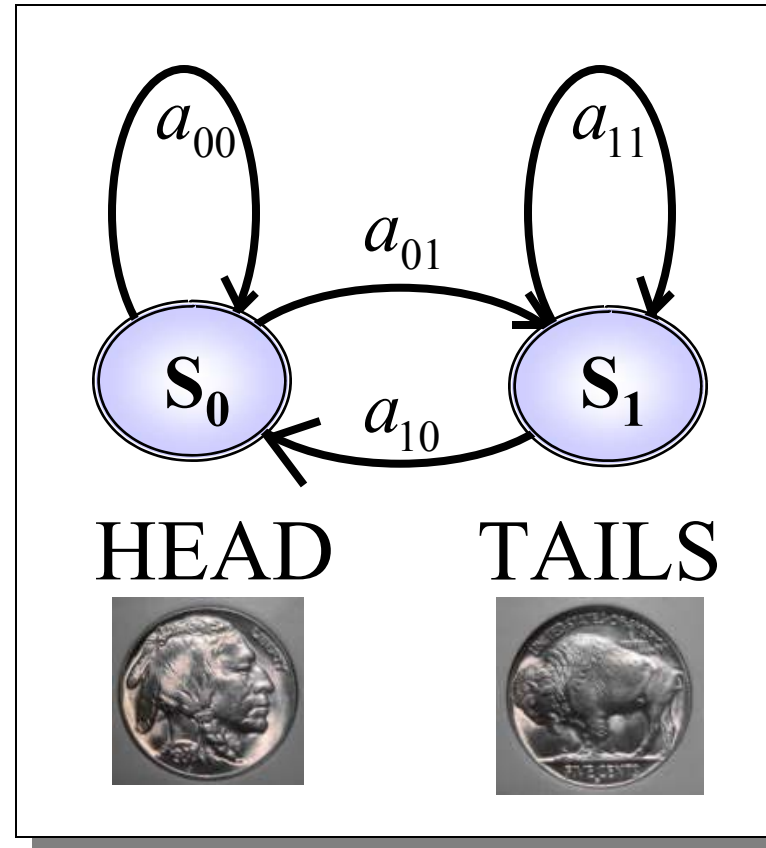
- Head (State 0)
- Tail (State 1)

- **Observed outcomes uniquely define state sequence**

- e.g., HHHTTTTHHTT →
S=0001110011

- **Transition Probabilities,**

$$A = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$



T-61.184

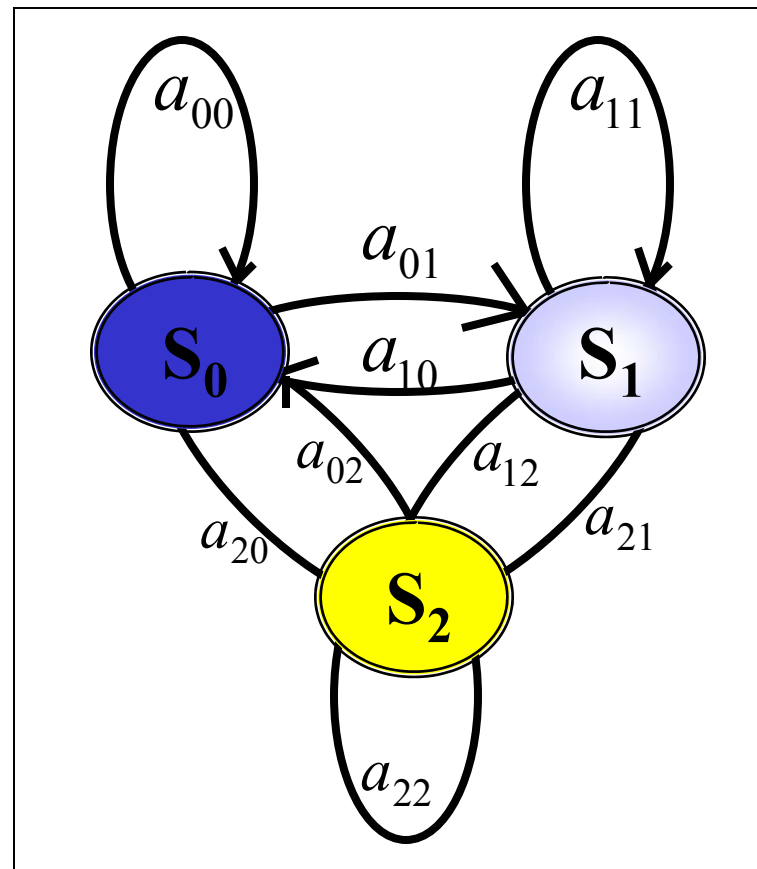
Example 2: Observable Markov Model of Weather

- **States:**

- S_0 : rainy
- S_1 : cloudy
- S_2 : sunny

- **With state transition probabilities,**

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$



T-61.184

Observable Markov Model of Weather

- **What is the probability that the weather for 8 consecutive days is “sun, sun, sun, rain, rain, sun, cloudy, sun”?**

- **Solution**

- Observation sequence:

$O = \{\text{sun, sun, sun, rain, rain, sun, cloudy, sun}\}$

- Corresponds to state sequence, $S = \{2, 2, 2, 0, 0, 2, 1, 2\}$

- Want to determine, $P(O \mid \text{model})$

- $P(O \mid \text{model}) = P (S=\{2,2,2,0,0,2,1,2\} \mid \text{model})$

Observable Markov Model of Weather

π_i : initial state probability

$$\pi_i = P(q_1 = i)$$

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$\begin{aligned} P(O \mid \text{model}) &= P(S = \{2, 2, 2, 0, 0, 2, 1, 2\} \mid \text{model}) \\ &= P(q_1 = 2)P(q_2 = 2 \mid q_1 = 2) \cdots P(q_8 = 2 \mid q_7 = 1) \\ &= \pi_2 \cdot a_{22} \cdot a_{22} \cdot a_{20} \cdot a_{00} \cdot a_{02} \cdot a_{21} \cdot a_{12} \\ &= \pi_2 \cdot (0.8)^2 (0.1) \cdot (0.4) \cdot (0.3) \cdot (0.1) \cdot (0.2) \end{aligned}$$

“Hidden” Markov Models

- **Observations are a probabilistic function of state**
- **Underlying state sequence is not observable (hidden)**
- **First-order assumption**
- **Output independence: observations are dependent only on the state that generated them, not on each other.**

Example: 2-Coins, Observable Markov Process

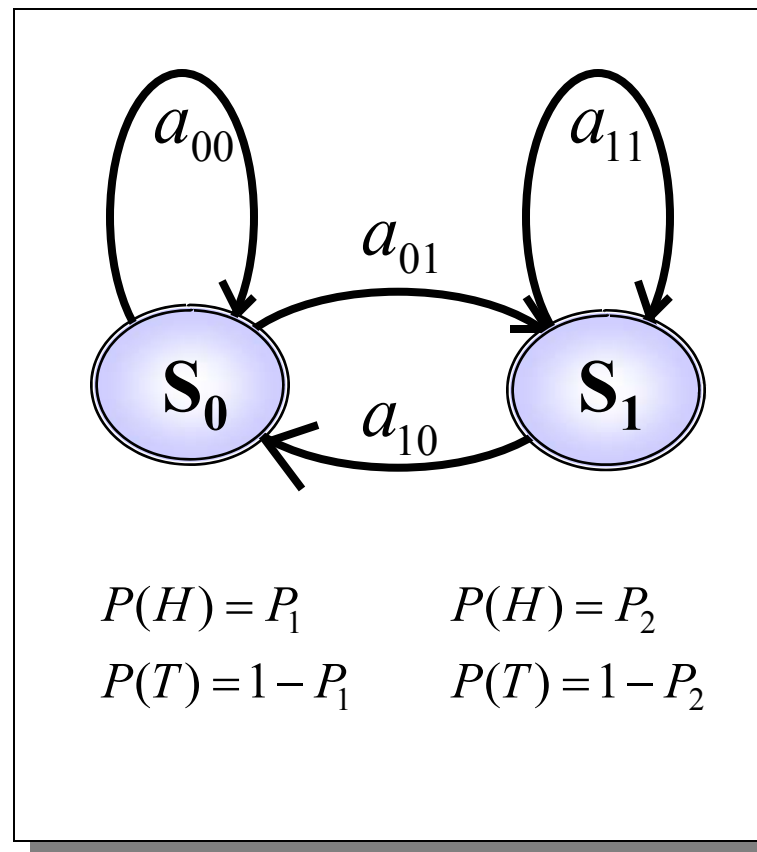
- **Observations**

- Head
- Tail

- **Observed outcomes do not uniquely define state sequence**

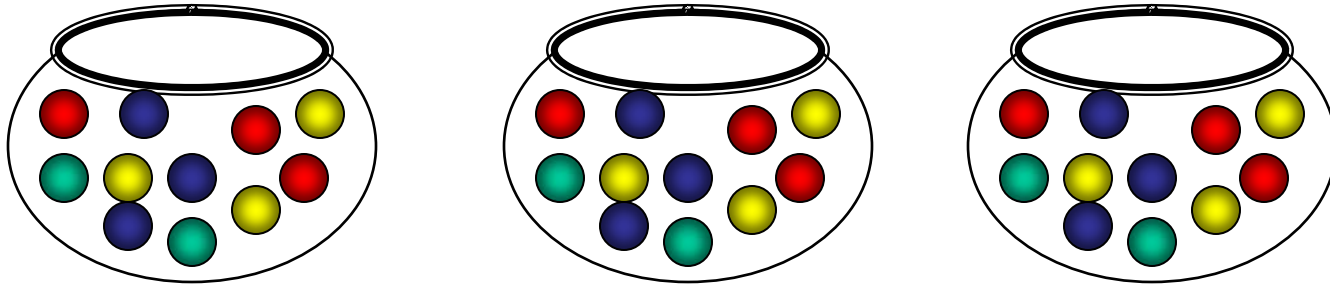
- **Transition Probabilities,**

$$A = \begin{bmatrix} a_{00} & 1 - a_{00} \\ 1 - a_{11} & a_{11} \end{bmatrix}$$



T-61.184

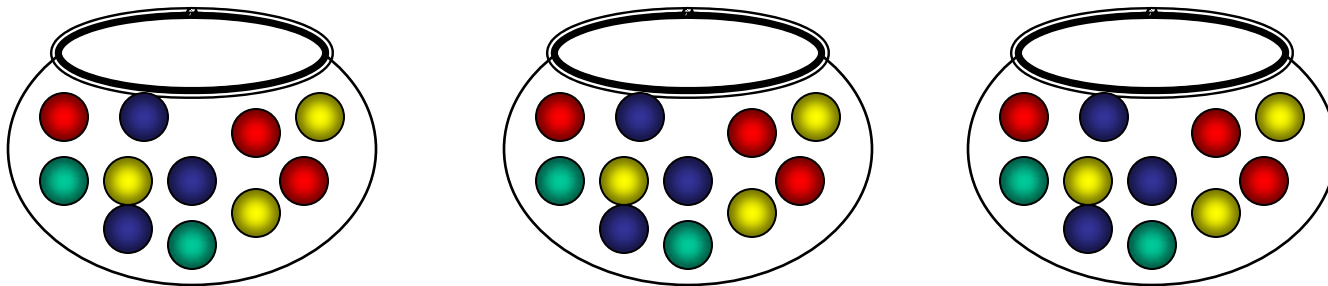
Urn-and-Ball Illustration



A Genie plays a game with a blind observer

1. Genie initially selects an Urn at random
2. Genie picks a colored ball, tells blind observer the color
3. Genie puts ball back in Urn
4. Genie moves to next Urn based on random selection procedure from current Urn.
5. Steps 2-4 repeated

Urn-and-Ball Illustration



- **Observations:**

- The color sequence of the balls

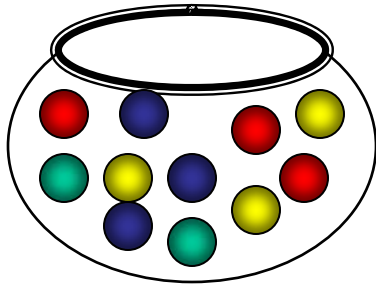
- **States:**

- The identity of the urn

- **State-Transition:**

- The process of selecting the urns

Urn-and-Ball Illustration

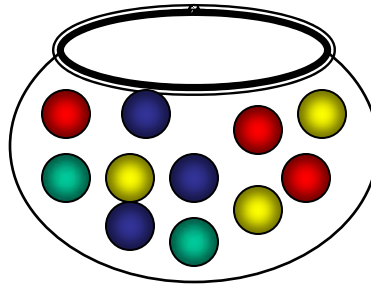


$$P(R) = 0.20$$

$$P(G) = 0.30$$

$$P(B) = 0.10$$

$$P(Y) = 0.60$$

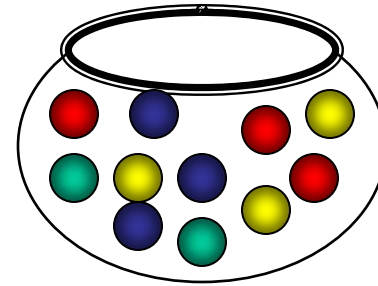


$$P(R) = 0.45$$

$$P(G) = 0.15$$

$$P(B) = 0.20$$

$$P(Y) = 0.20$$



$$P(R) = 0.15$$

$$P(G) = 0.70$$

$$P(B) = 0.10$$

$$P(Y) = 0.05$$

■ **Observation sequence: R B Y Y G B Y G R ...**

□ Individual colors (observations) don't reveal urn (state)

Discrete Symbol Observation HMM

- A set of N states

$$S = \{S_0, S_1, \dots, S_N\}$$

- Transition Probabilities

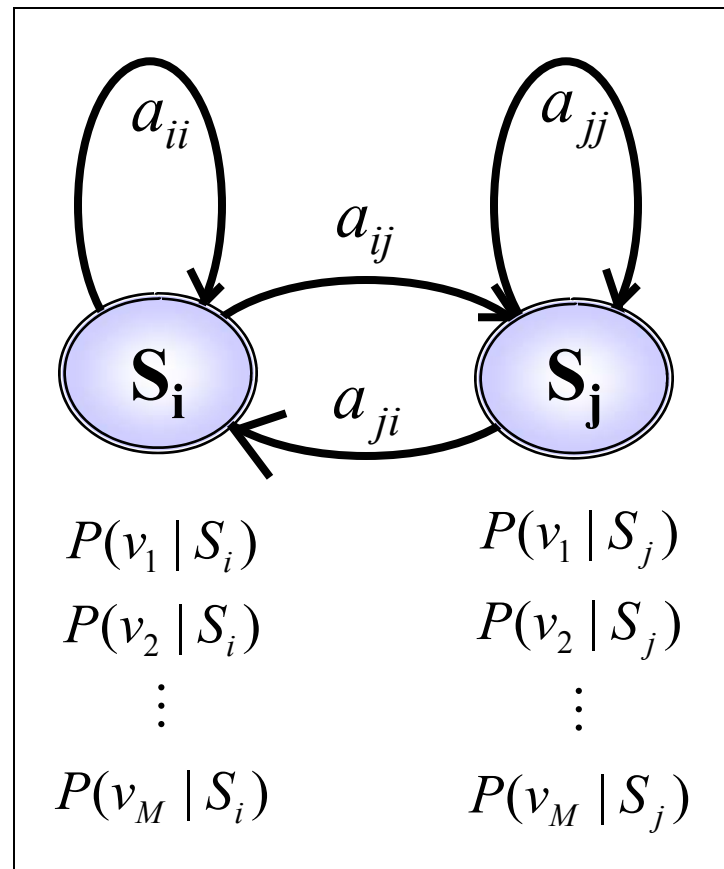
$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$$

- A set of M observation symbols

$$V = \{v_1, v_2, \dots, v_M\}$$

- Probability Distribution (for state j , symbol k)

$$b_j(k) = P(o_t = v_k | q_t = j)$$



T-61.184

Discrete Symbol Observation HMM

- **Also characterized by:**

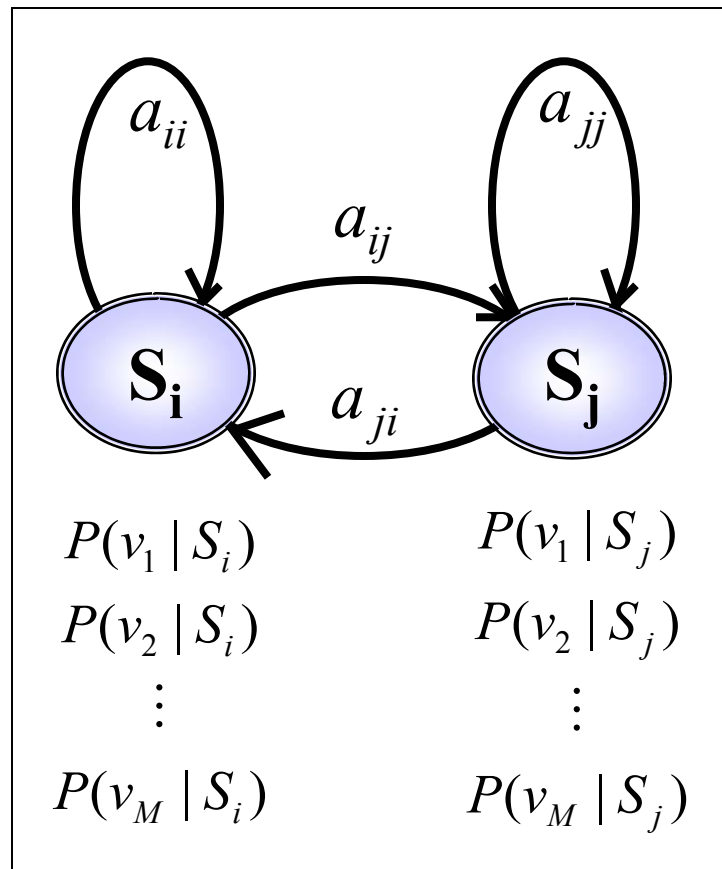
- An initial state distribution

$$\pi = \{\pi_i\} = P(q_1 = i)$$

- **Specification thus requires**

- 2 model parameters, N and M
- Specification of M symbols
- 3 probability measures A,B, π

$$\lambda = (A, B, \pi)$$

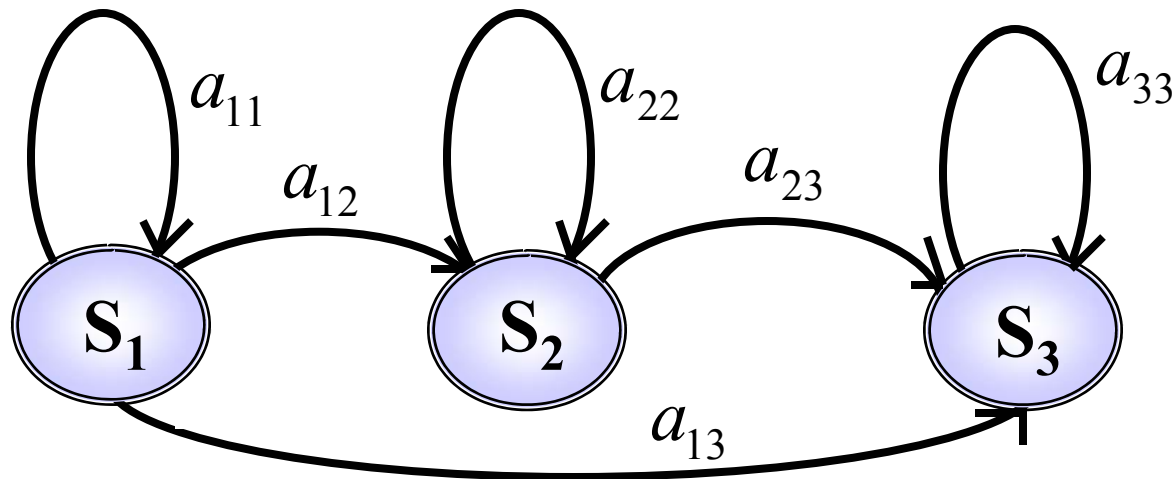


Left-to-Right HMMs

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$a_{ij} = 0 \quad j < i$$

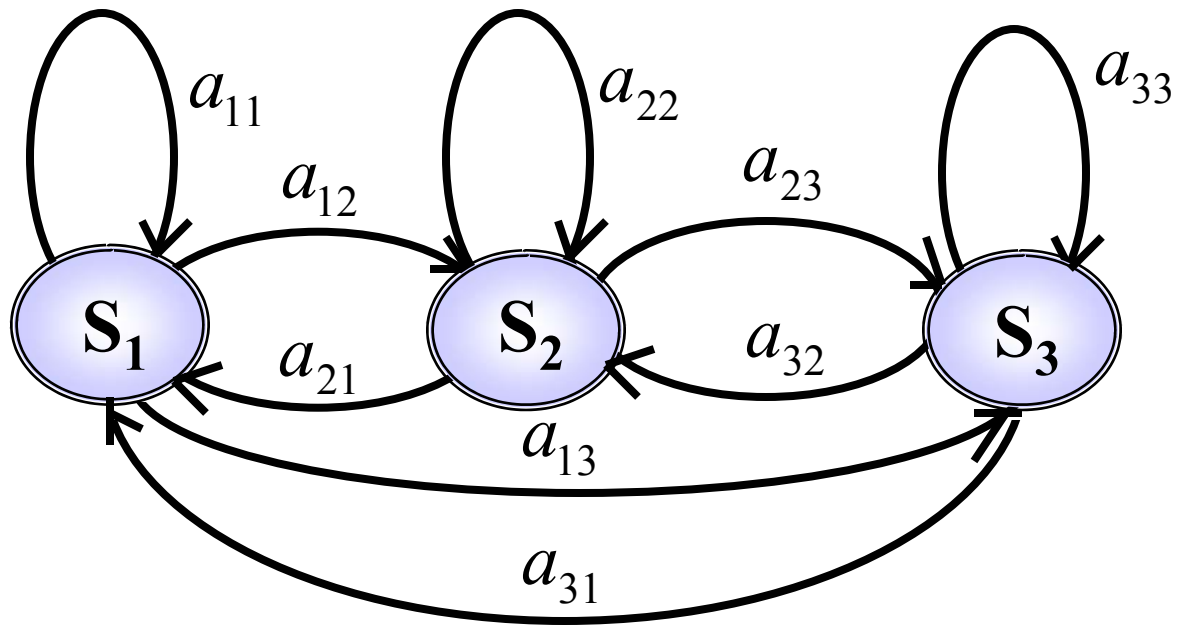
$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases}$$



T-61.184

Ergodic HMMs

$$a_{ij} > 0 \quad \forall i, \forall j$$



HMM as a Symbol Generator

- Given parameters $N, M, A, B,$ and $\pi,$ HMMs can generate an observation sequence,

$$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$$

- 1. Choose initial state $q_1 = i$ from init. state distribution π
- 2. Set $t = 1$
- 3. Choose $\mathbf{O}_t = \mathbf{V}_k$ according to distribution $b_i(k)$
- 4. Transition to set state $q_{t+1} = j$ according to state-transition probability distribution a_{ij}
- 5. Set $t = t + 1$ go to step 3

HMM as a Symbol Generator

- Example output from symbol generation

Time, t	1	2	3	4	5	6	...	T
State	q_1	q_2	q_3	q_4	q_5	q_6		q_T
Observation	\mathbf{o}_1	\mathbf{o}_2	\mathbf{o}_3	\mathbf{o}_4	\mathbf{o}_5	\mathbf{o}_6		\mathbf{o}_T

- We can think of the HMM as generating the observation sequence as it transitions from state to state.

Three Interesting Problems

■ Problem 1: Scoring & Evaluation

- How to efficiently compute the probability of an observation sequence (O) given a model (λ)? $\rightarrow P(O | \lambda)$

■ Problem 2: Decoding

- Given an observation sequence (O) and a model (λ), how do we determine the corresponding state-sequence (q) that “best explains” how the observations were generated?

■ Problem 3: Training

- How to adjust model parameters ($\lambda = \{A, B, \pi\}$) to maximize probability of generating a given observation sequence? \rightarrow maximize $P(O | \lambda)$.

Problem 1: Scoring and Evaluation

- Given an observation sequence,

$$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$$

- Want to compute probability of generating it:

$$P(\mathbf{O} | \lambda)$$

- Let's assume a particular sequence of states,

$$q = \{q_1, q_2, \dots, q_T\}$$

Problem 1: Scoring and Evaluation

- Using the chain rule we can decompose the problem by summing over all possible state sequences,

$$P(\mathbf{O} \mid \lambda) = \sum_{\text{all } q} P(\mathbf{O} \mid q, \lambda)P(q \mid \lambda)$$

- The first term relates to the likelihood of generating the observed symbol sequence given the assumed state sequence.
- The second term relates to how likely the system is to step through those sequence of states.

Problem 1: Scoring and Evaluation

- **Probability of the observation sequence given the state sequence,**

$$\begin{aligned} P(\mathbf{O} | q, \lambda) &= \prod_{t=1}^T p(\mathbf{o}_t | q_t, \lambda) \\ &= b_{q_1}(\mathbf{o}_1) \cdot b_{q_2}(\mathbf{o}_2) \cdots b_{q_T}(\mathbf{o}_T) \end{aligned}$$

- **Probability of the state sequence,**

$$P(q | \lambda) = \pi_{q_1}(a_{q_1 q_2}) \cdot (a_{q_2 q_3}) \cdots (a_{q_{T-1} q_T})$$

Problem 1: Scoring and Evaluation

- Using the chain rule,

$$\begin{aligned} P(\mathbf{O} \mid \lambda) &= \sum_{\text{all } q} P(\mathbf{O} \mid q, \lambda) P(q \mid \lambda) \\ &= \sum_{\text{all } q} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T) \end{aligned}$$

- This is not practical to compute. For N states, T observations, the number of state sequences is:

$$O(2T * N^T)$$

Forward Algorithm

- **Definition:** $\alpha_t(i) = P(\mathbf{o}_1 \mathbf{o}_2 \dots \mathbf{o}_t, q_t = i \mid \lambda)$

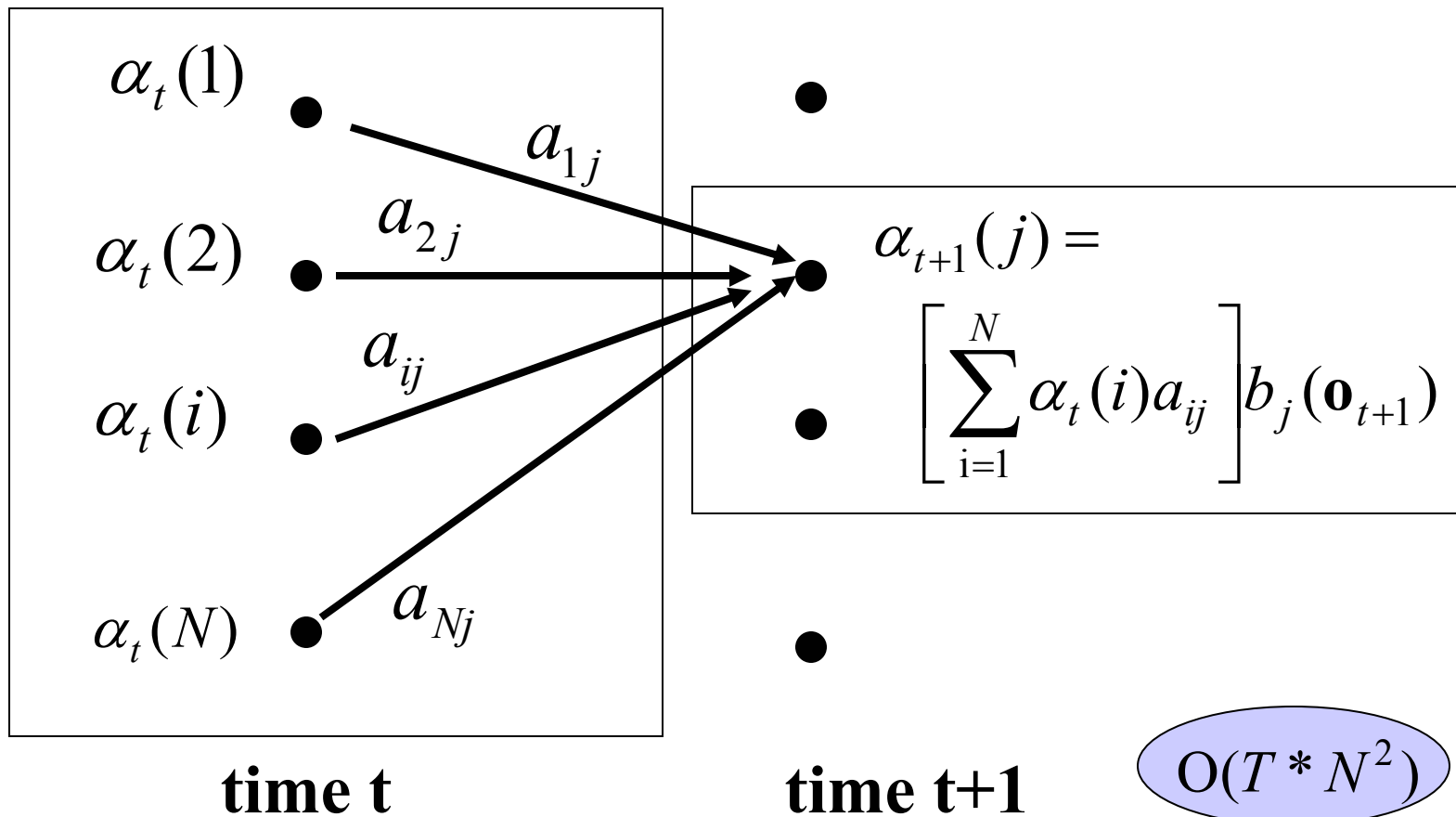
(Probability of seeing observations \mathbf{o}_1 to \mathbf{o}_t and ending at state i given HMM λ)

1. Initialization $\alpha_0(i) = \pi_i$

2. Induction $\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(\mathbf{o}_{t+1})$

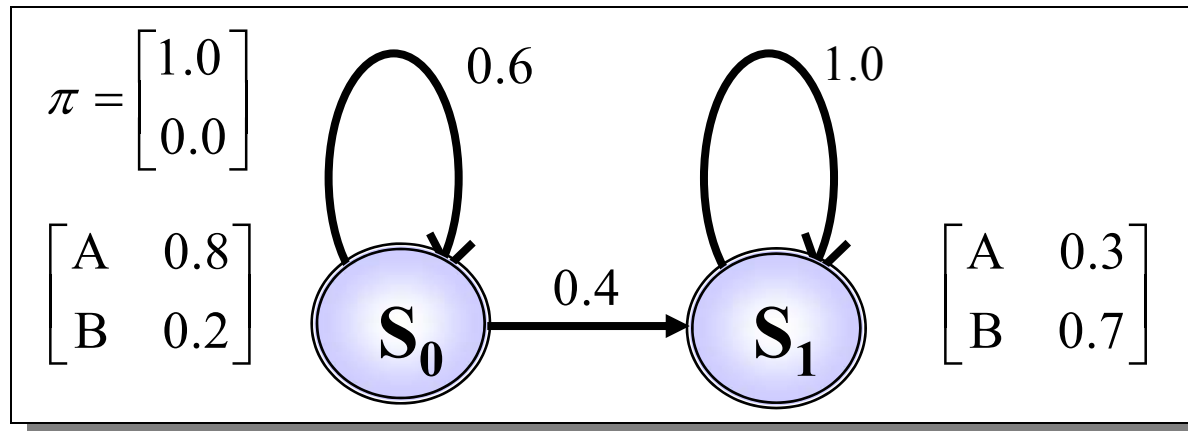
3. Termination $P(\mathbf{O} \mid \lambda) = \sum_{i=1}^N \alpha_T(i)$

Forward Algorithm Illustration



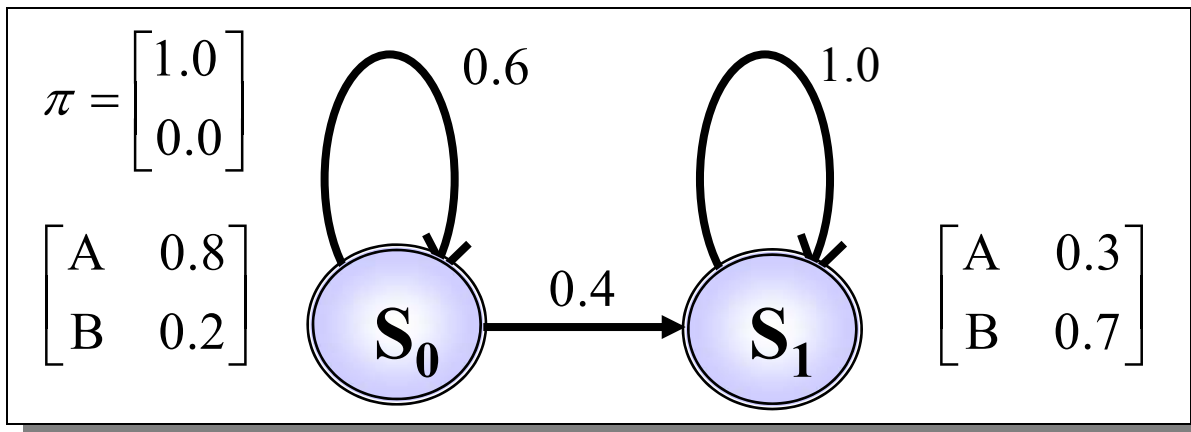
T-61.184

Forward Algorithm Example

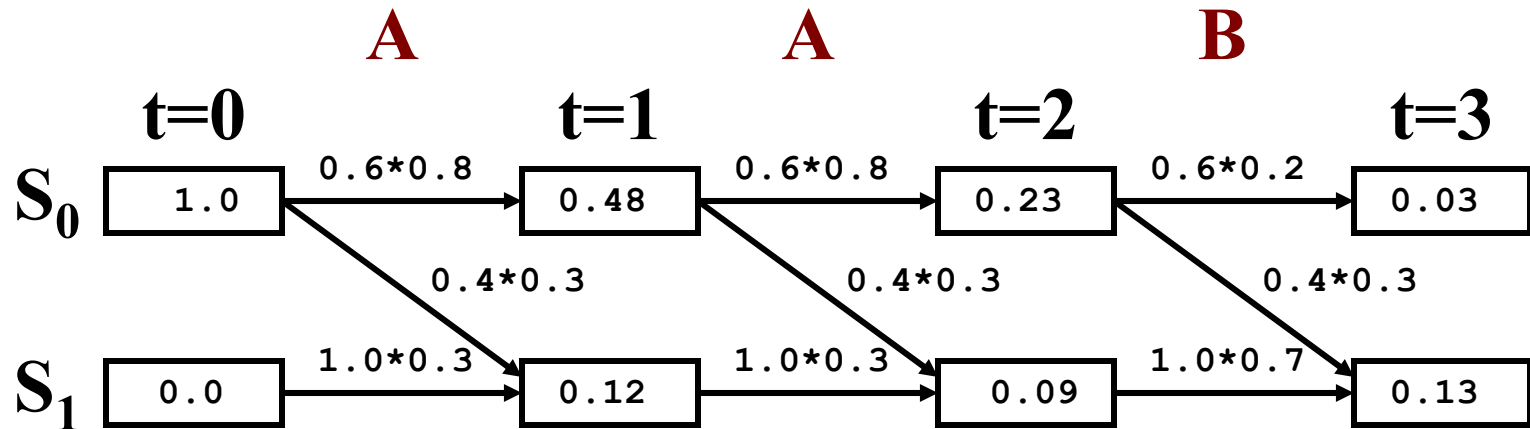


- Given the above HMM with discrete observations “A” and “B”, what is the probability of generating the sequence “ $O = \{A, A, B\}$ ”?
- In other words, find $P(O = \{A, A, B\} | \lambda)$

Forward Algorithm Example



Answer:
 $P(O|\lambda) = 0.03 + 0.13 = 0.16$



T-61.184

Backward Algorithm

■ **Definition:** $\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2} \dots \mathbf{o}_T, q_t = i \mid \lambda)$

■ **Probability of observation sequence \mathbf{o}_{t+1} to \mathbf{o}_T given state i at time t and HMM λ**

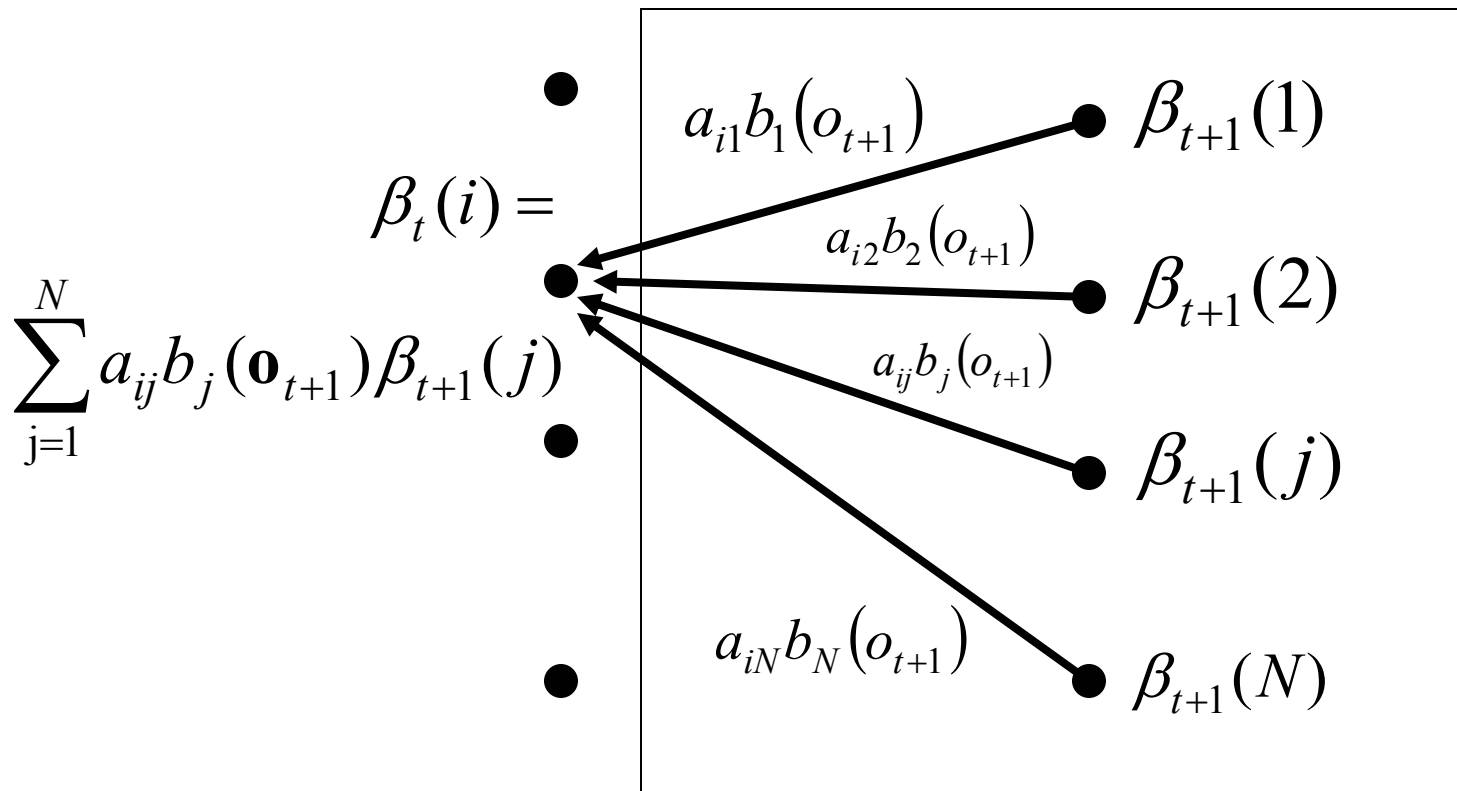
■ **Initialization** $\beta_T(i) = 1$

■ **Induction**
$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)$$

$t = T-1, T-2, \dots, 1$

$1 \leq i \leq N$

Backward Algorithm Illustration



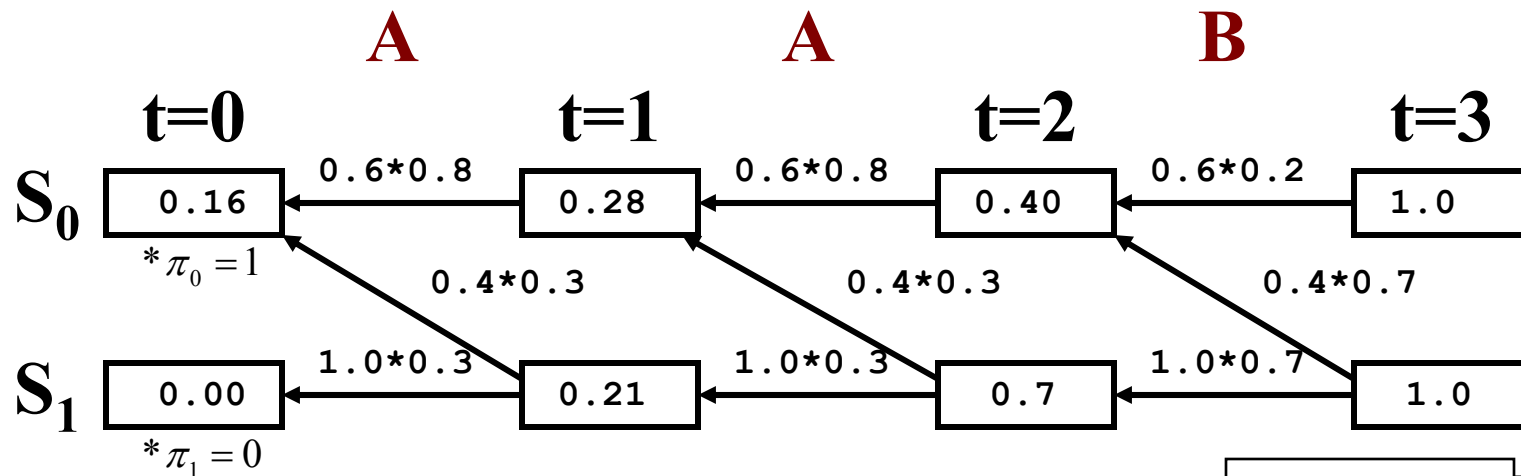
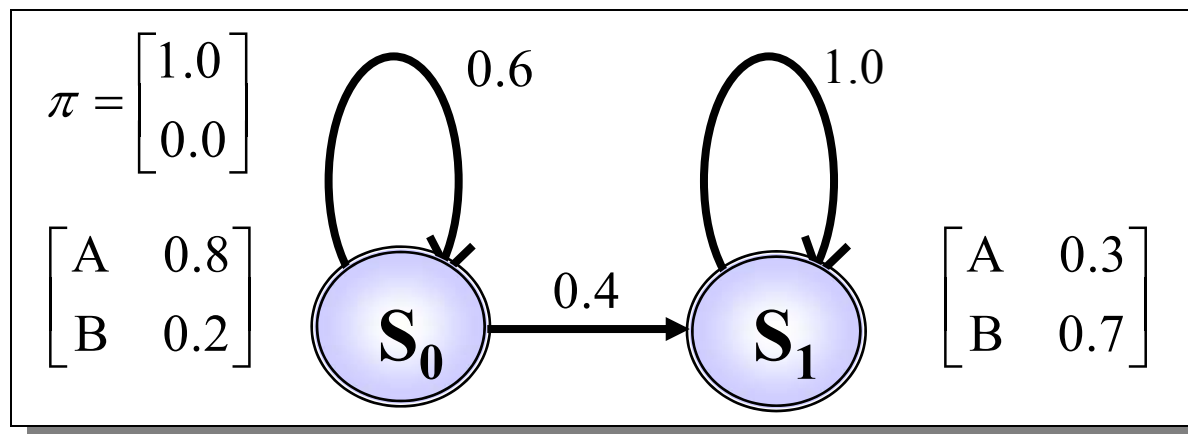
time t

time t+1

$O(T * N^2)$

T-61.184

Backward Algorithm Example



T-61.184

Problem 1: Scoring and Evaluation

- In fact, there are 2 ways to compute $P(\mathbf{O} | \lambda)$

$$P(\mathbf{O} | \lambda) = \sum_{i=1}^N \alpha_T(i) \quad \text{Forward Algorithm}$$

$$P(\mathbf{O} | \lambda) = \sum_{i=1}^N \pi_i \beta_0(i) \quad \text{Backward Algorithm}$$

- Computation of each term is $O(N^2T)$

Problem 2: Decoding

- Given an observation sequence,

$$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$$

- Find the single best sequence of states,

$$q = \{q_1, q_2, \dots, q_T\}$$

- Which maximizes,

$$P(\mathbf{O}, q | \lambda)$$

Solution: Viterbi Algorithm

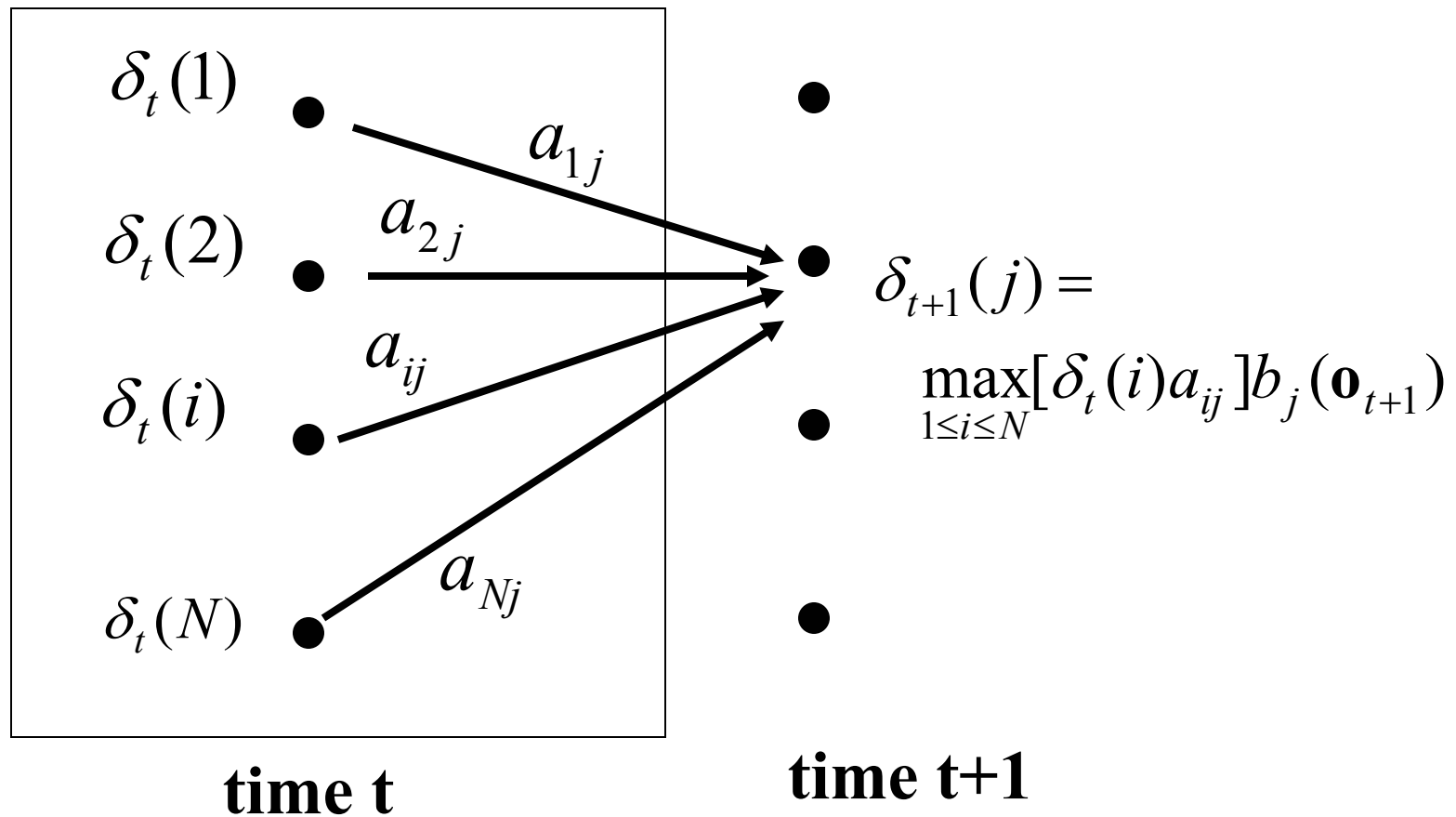
■ Define:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \cdots q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 \cdots \mathbf{o}_t \mid \lambda)$$

“Highest probable state sequence that accounts for observations 1 through $t-1$ and ends in state i and time t ”.

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(o_{t+1})$$

Viterbi Algorithm Illustration



Viterbi Algorithm

1. **Initialization** $\delta_1(i) = \pi_i b_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

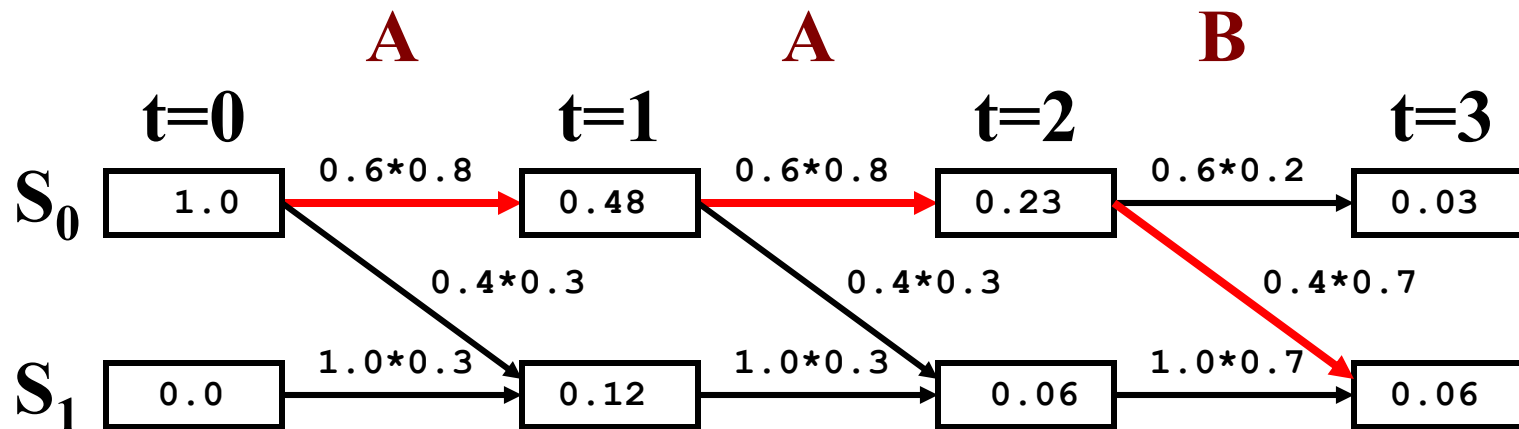
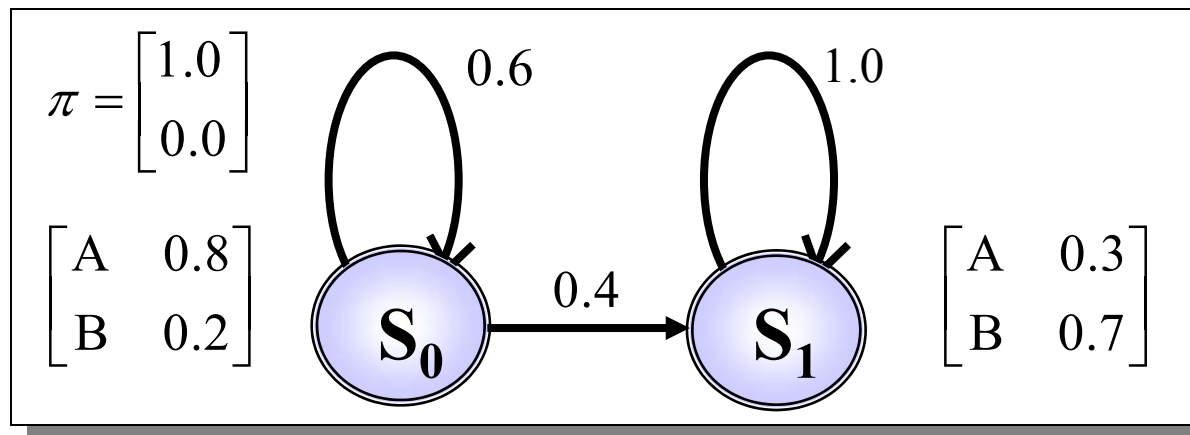
2. **Recursion**

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t)$$
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}]$$

3. **Termination** $P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$

4. **Path Back trace** $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Viterbi Algorithm Illustration



T-61.184

Viterbi Algorithm in Log-Domain

- Due to numerical underflow issues, it is often common to conduct the Viterbi algorithm in the log-domain,

$$\tilde{\pi}_i = \log(\pi_i)$$

$$\tilde{b}_j(o_t) = \log(b_j(o_t))$$

$$\tilde{a}_{ij} = \log(a_{ij})$$

- Viterbi search in speech recognition systems is implemented in the log-domain for example.

Viterbi Algorithm in Log-Domain

1. Initialization $\tilde{\delta}_1(i) = \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

2. Recursion
$$\delta_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}]$$

3. Termination $\tilde{P}^* = \max_{1 \leq i \leq N} [\tilde{\delta}_T(i)] \quad q_T^* = \arg \max_{1 \leq i \leq N} [\tilde{\delta}_T(i)]$

4. Path Back trace $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Problem 3: Training

- **Train parameters of HMM**
 - Tune λ to maximize $P(O|\lambda)$
 - No efficient algorithm for global optimum
 - Efficient iterative algorithm finds a local optimum
- **(Baum-Welch) Forward-Backward Algorithm**
 - Compute probabilities using current model λ
 - Refine $\lambda \rightarrow \bar{\lambda}$ based on computed values
 - Uses α and β from forward and backward algorithms

Forward-Backward Algorithm

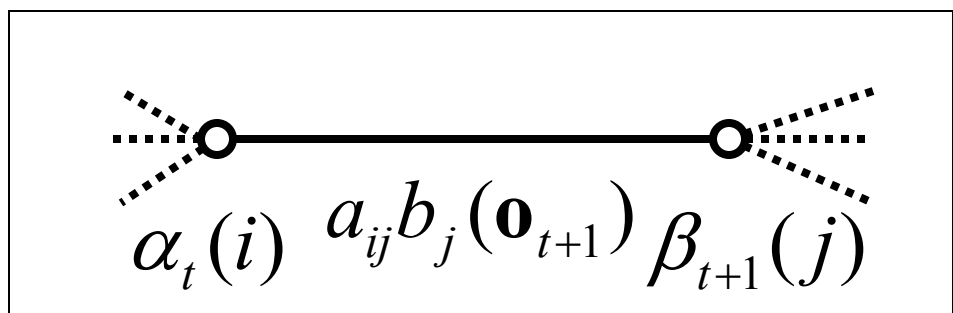
- Define:

$$\begin{aligned}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{O}, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, \mathbf{O} \mid \lambda)}{P(\mathbf{O} \mid \lambda)}\end{aligned}$$

“Probability of being in state i at time t and state j at time $t+1$ given the model and observation sequence”

Forward-Backward Algorithm

$$\begin{aligned}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{O}, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \lambda)}\end{aligned}$$



Forward-Backward Algorithm

$$\begin{aligned}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{O}, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)}\end{aligned}$$

Forward-Backward Algorithm

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j)$$

Probability of being in state i at time t

$$\sum_{t=1}^{T-1} \gamma_t(i) =$$

Expected number of transitions from state i in O .

$$\sum_{t=1}^{T-1} \xi_t(i, j) =$$

Expected number of transitions from state i to state j in O .

Forward-Backward Algorithm

- Using these definitions, we can compute the initial state occupancy probability,

$$\begin{aligned}\bar{\pi}_i &= \text{expected number of times in state } i \text{ at time } t = 1 \\ &= \gamma_1(i)\end{aligned}$$

Forward-Backward Algorithm

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\begin{aligned} & \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^N \xi_t(i, j)} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned}$$

Forward-Backward Algorithm

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing } k\text{th symbol}}{\text{expected number times in state } j}$$

$$\begin{aligned} & \frac{\sum_{t=1}^T \sum_{j=1}^N \xi_t(j, j)}{\sum_{t=1}^T \sum_{j=1}^N \xi_t(j, j)} = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} = \frac{\sum_{t=1}^T \alpha_t(j) \beta_t(j)}{\sum_{t=1}^T \alpha_t(j) \beta_t(j)} \\ & \stackrel{\mathbf{o}_t = v_k}{=} \frac{\sum_{t=1}^T \sum_{j=1}^N \xi_t(j, j)}{\sum_{t=1}^T \sum_{j=1}^N \xi_t(j, j)} = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} = \frac{\sum_{t=1}^T \alpha_t(j) \beta_t(j)}{\sum_{t=1}^T \alpha_t(j) \beta_t(j)} \end{aligned}$$

Forward-Backward Algorithm

1. Initialize $\lambda=(\mathbf{A},\mathbf{B},\pi)$
2. Compute α , β , and ξ
3. Estimate $\bar{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\pi})$
4. Replace λ with $\bar{\lambda}$
5. If not converged go to 2

It can be shown
That

$$P(\mathbf{O}|\bar{\lambda}) > P(\mathbf{O}|\lambda)$$

Unless
 $\bar{\lambda} = \lambda$

FB Algorithm Satisfies Constraints:

$$\sum_{i=1}^N \bar{\pi}_i = 1$$

$$\sum_{j=1}^N \bar{a}_{ij} = 1 \quad 1 \leq i \leq N$$

$$\sum_{k=1}^M \bar{b}_j(k) = 1 \quad 1 \leq j \leq N$$

Continuous Observation Densities

- **Discussions up until this point have considered observations characterized by discrete symbols**
- **Often, we work with continuous valued data**
- **Can convert continuous variables to discrete symbols using a Vector Quantizer (VQ) codebook [with information loss]**
- **How to model continuous observations directly?**

Mixture Gaussian PDFs

- **Mixture Gaussian PDF with M components,**

$$b_j(\mathbf{o}_t) = \sum_{k=1}^M c_{jk} \mathbf{N}(\mathbf{o}_t, \mu_{jk}, \Sigma_{jk})$$

- **Constraints on the mixture weights,**

$$\sum_{k=1}^M c_{jk} = 1$$

$$c_{jk} \geq 0, \quad 1 \leq k \leq M$$

Definition

- Probability of being in state j at time t with the k th mixture component accounting for \mathbf{o}_t :

$$\gamma_t(j, k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jk} \mathbf{N}(\mathbf{o}_t, \mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^M c_{jm} \mathbf{N}(\mathbf{o}_t, \mu_{jm}, \Sigma_{jm})} \right]$$

Resulting Equations for Mixture Weight & Mean Update

$$\bar{c}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)}$$

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k) \cdot \mathbf{o}_t}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)}$$

Resulting Equations for Transition Matrix & Variance Update

- Transition probability a_{ij} same as case for discrete symbols. Covariance Matrix:

$$\bar{\Sigma}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k) \cdot (\mathbf{o}_t - \bar{\mu}_{jk})(\mathbf{o}_t - \bar{\mu}_{jk})'}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)}$$

Estimation from Multiple Observation Sequences

- **We estimate HMM parameters from multiple examples of speech:**
 - Collected from different speakers
 - In different contexts
- **We attempt to model variability in producing each sound unit by estimating parameters from large corpora of speech data.**

Multiple Observations

- Assume K training patterns (observation sequences),

$$\mathbf{O} = [\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \dots, \mathbf{O}^{(K)}]$$

- Where,

$$\mathbf{O}^{(k)} = \{\mathbf{o}_1^{(k)}, \mathbf{o}_1^{(k)}, \dots, \mathbf{o}_{T_k}^{(k)}\}$$

- And,

$$P_k = P(\mathbf{O}^{(k)} | \lambda)$$

Transition Probabilities with Multiple Observations

$$\bar{a}_{ij} = \frac{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(\mathbf{o}_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}$$

Observation Probabilities with Multiple Observations

$$\bar{b}_j(l) = \frac{\sum_{k=1}^K \frac{1}{P_k} \sum_{\substack{t=1 \\ s.t. O_t=v_l}}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^K \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}$$

Single State HMM Example

- How to estimate the parameters of a single-state HMM model with M component Gaussians?

$$b(\mathbf{o}_t) = \sum_{k=1}^M w_k b_k(\mathbf{o}_t, \mu_k, \Sigma_k)$$
$$= \sum_{k=1}^M \frac{w_k}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{o}_t - \mathbf{u}_k)' \Sigma_k^{-1} (\mathbf{o}_t - \mathbf{u}_k)\right)$$

Single State HMM Example

- Define probability of t th observation being generated by the k th mixture component,

$$P(k | o_t, \lambda) = \frac{w_k b_k(o_t)}{\sum_{k=1}^M w_k b_k(o_t)}$$

- Note that “ k ” (b_k) here refers to mixture component, not HMM state. We are assuming just 1 HMM state.

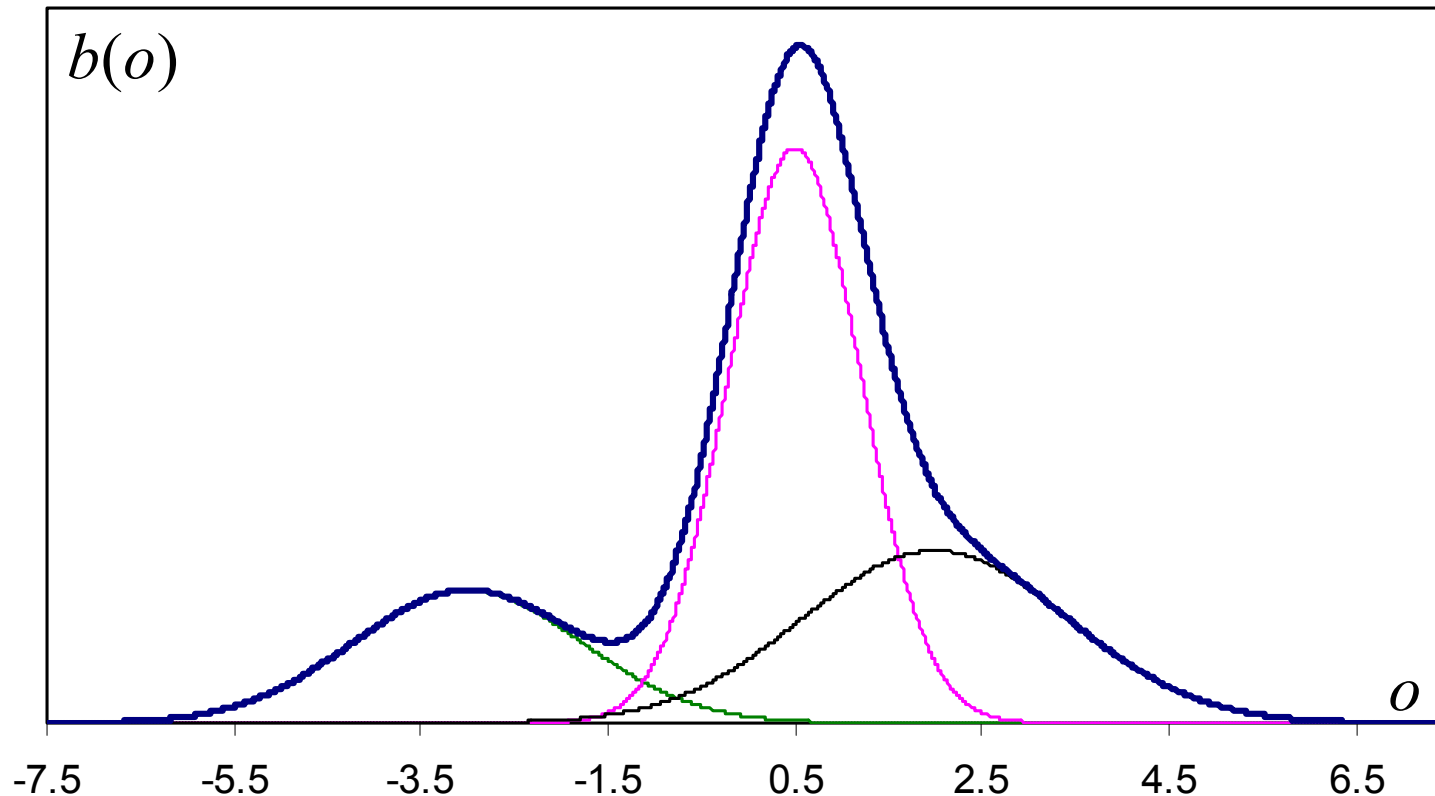
Single State HMM Update Equations

**Updated
(new)
parameter
estimates**

$$\bar{w}_k = \frac{1}{T} \sum_{t=1}^T P(k | o_t, \lambda)$$
$$\bar{u}_k = \frac{\sum_{t=1}^T P(k | o_t, \lambda) \cdot o_t}{\sum_{t=1}^T P(k | o_t, \lambda)}$$
$$\bar{\Sigma}_k = \frac{\sum_{t=1}^T P(k | o_t, \lambda) \cdot (o_t)^2}{\sum_{t=1}^T P(k | o_t, \lambda)} - (\bar{u}_k)^2$$

**This term
is computed
using model
parameters
from previous
algorithm
iteration.**

Mixture Gaussian (1-D case)



Example: 3 mixtures used to model underlying random process of 3 Gaussians

T-61.184

HMM Assumptions

- **First-Order Markov Chain**
 - Probability of transitioning to a state only depends on the current state
- **Time-independence of State Transitions**
 - Transitioning from state A to state B is independent of time
- **Observation Independence**
 - Observations don't depend on each other, just on the states that generated them
- **(sometimes) Left-to-Right Topology**
 - As we will see, it is generally assumed that a left-to-right HMM is used to model speech units (phonemes)

Important Concepts from Today

- **Do you understand the basic idea of an HMM?**
- **Could you implement the Viterbi Algorithm if you had to?**
- **Can you estimate at least the parameters of a single-state HMM (mixture-Gaussians)?**
- **What assumptions do HMMs impose on the data being modeled?**

Next Week

- **How to use HMMs for modeling speech units?**
- **Consider major strategies for acoustic training**
- **How to model context dependencies using HMMs**
- **Some practical notes on HMM training for speech recognition**