

**Dr. HMM**

***How I Learned to Stop Worrying and  
Love Hidden Markov Models***

Tracy Westeyn

Contextual Computing Group  
GVU, College of Computing  
Georgia Institute of Technology



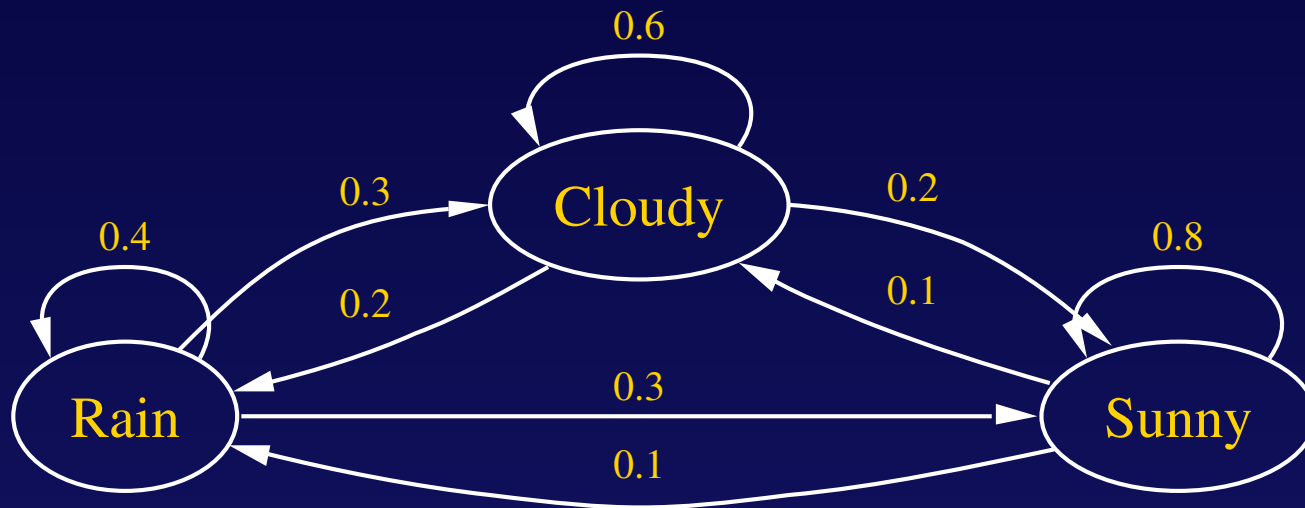
# Introduction

- Markov Models
- Hidden Processes
- Hidden Markov Models
- Three Problems for HMMs
- Implementation Issues



# Markov Models

- Each state of the model corresponds to an observable event.
- Model representing the weather observed at noon.

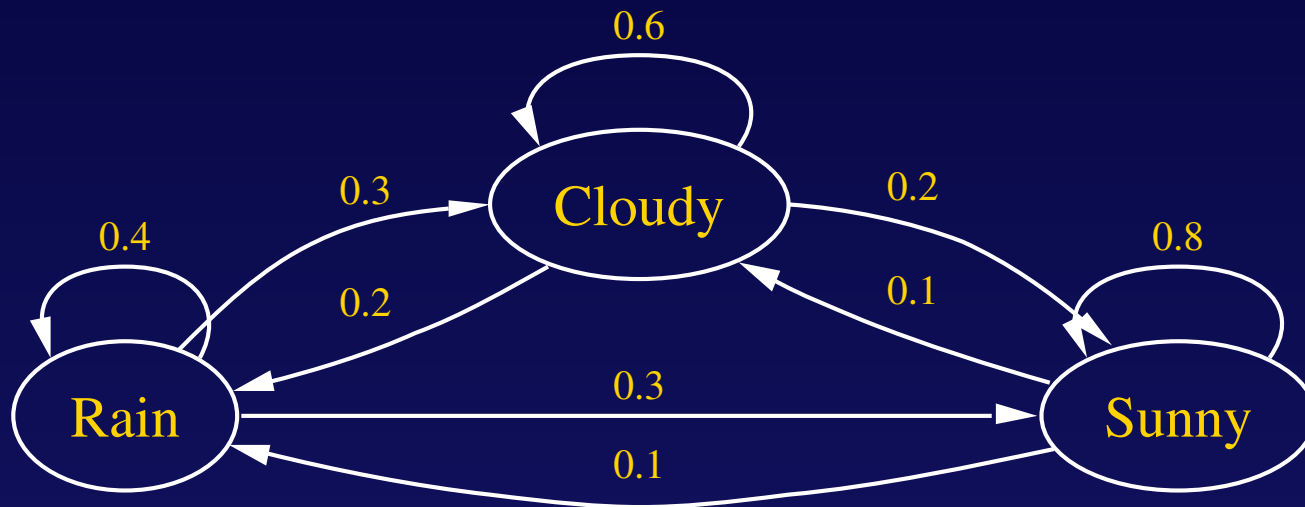


$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$



# Markov Models

- What sorts of questions can we address using these types of models?



- ▶ What is the probability of observing a specific sequence given the model?  $P(O|Model)$
- ▶ Given that the model is in a known state, what is the probability that the model stays in that state for exactly  $d$  time steps.



# Markov Models

- Specific Sequence is determined by a joint probability of the state sequence given the model:

$$P(O|model) = P(Q|model) = \pi_3 * a_{33} * a_{31}$$

$$P(O|model) = P(Q|model) = 1 * .80 * .10$$

- Expected number of observations in a state:

$$\sum_{d=1}^{\infty} d(a_{ij})^{(d-1)}(1 - a_{ij}) = \frac{1}{1 - a_{ij}}$$



# Hidden Markov Models

- We cannot directly observe the process we are trying to model
- We can observe events that are a probabilistic functions of the state of the hidden process



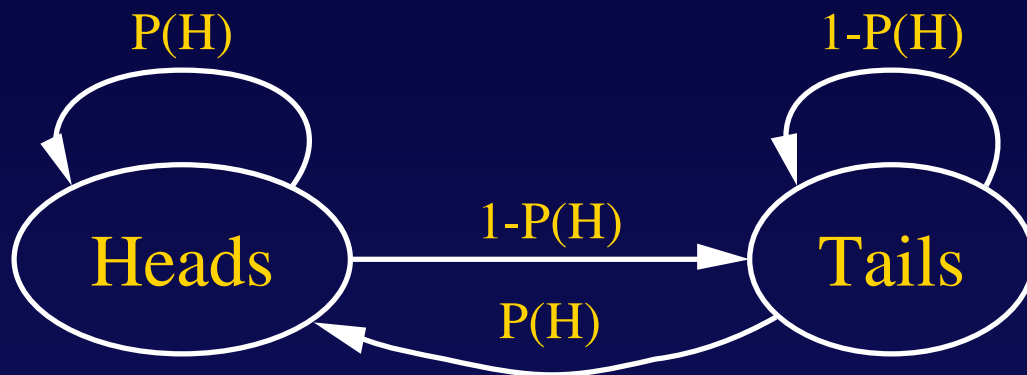
# Hidden Process: Coin Flip Example

- You are divided from a person by a curtain
- The person on the other side of the curtain is reading off an observation sequence of  $H$  and  $T$  corresponding to the result of coin flips:
  - ▶  $O = \text{“HHTTTHTTH ...”}$
- You would like to model the process that is generating the series of heads and tails.
  - ▶ What should the states correspond to?
  - ▶ How many states do we need?
- Answers depend on how many coins were used to generate the sequence



# Hidden Process: Coin Flip Example

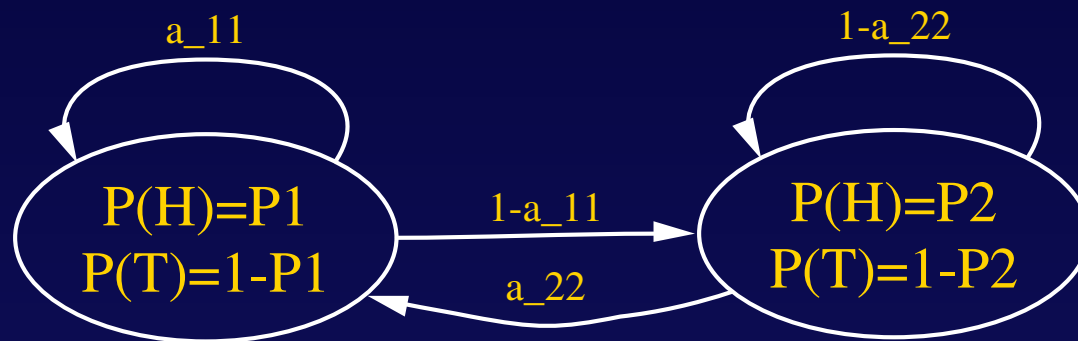
- Assume that only one biased coin was flipped:



- ▶ 2-state model (one for each face of the coin)
- ▶ observable model
- ▶ determine the bias of the coin

# Hidden Process: Coin Flip Example

- Assume that two biased coin were flipped:



- ▶ 2-state model (one for each coin)
  - Each state is characterized by a probability distribution of heads and tails
- ▶ Transitions between the coins is characterized by a transition matrix
  - physical mechanism
  - set of independent coin tosses
  - some other probabilistic event



# Data generation: Ball and Urns Model

- $N$  urns each with a large number of  $M$  distinctly colored balls
- Choose an urn based on only the previously chosen urn
- Choose a ball from the urn, observe the color and place it back into the urn
- Repeat the process to generate a finite observation sequence of colors.
- Generative process:
  1. generate a sequence of hidden urn choices
  2. generate a sequence of observed ball choices



# Elements of an HMM: $\lambda = \{A, B, \pi\}$

1. Number of states,  $N$  in the model
2. Number of distinct observation symbols,  $M$ , per state
3. State transition probability distribution:

$$A = a_{ij} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & a_{n(n-1)} & a_{nn} \end{bmatrix}$$

4. Observation symbol probability distribution for each state,  
 $B = b_j(k) = P(v_k \text{ at } t | q_t = S_j)$
5. Initial state distribution,  $\pi$



# Three Basic Problems for HMMs

1. How do we compute the probability of an observed sequence being generated by a specific model:  $P(O|\lambda)$
2. Given observations and a model, what is path (state sequence) that best explains the observation sequence
3. How do we train the models (adjust the parameters  $\lambda$ ) to maximize  $P(O|\lambda)$



# Solution for Problem 1: $P(O|\lambda)$

- Enumerate every possible state sequence with length equivalent to the number of observations.

$$P(O|\lambda) = \sum_{all Q} P(O|Q, \lambda)P(Q|\lambda)$$

- Note there is an error in the derivation (eq 15)  
The last term should be  $P(Q|\lambda)$  not  $P(Q, \lambda)$
- This is inefficient:  $2T * N^T$  calculations
  - ▶  $N^T$  possible state transitions
  - ▶  $(2T - 1) * N^T$  multiplications and  $(N^T - 1) * N^T$  additions



# Forward Algorithm

- The forward variable,  $\alpha$ , defines the probability of observing the sequence thus far and residing in a specific state given the model

$$\alpha_t(i) = P(O_1 O_2 \dots O_t, q_t = S_i | \lambda)$$

- We solve for  $P(O|\lambda)$  by computing  $\alpha_T$  for all states and summing over these values

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

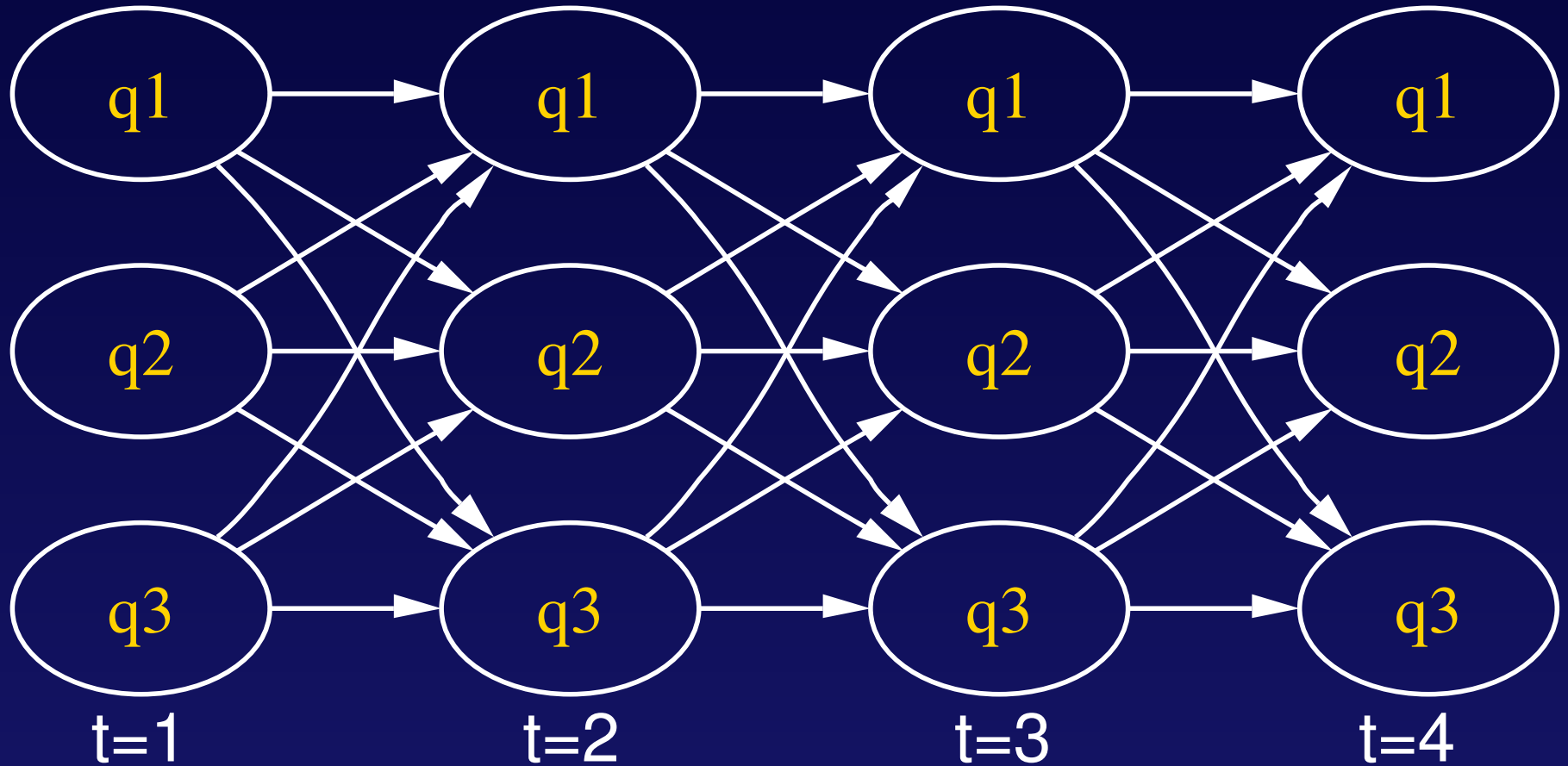
- $\alpha_T$  is inductively computed:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad 1 \leq t \leq T - 1$$

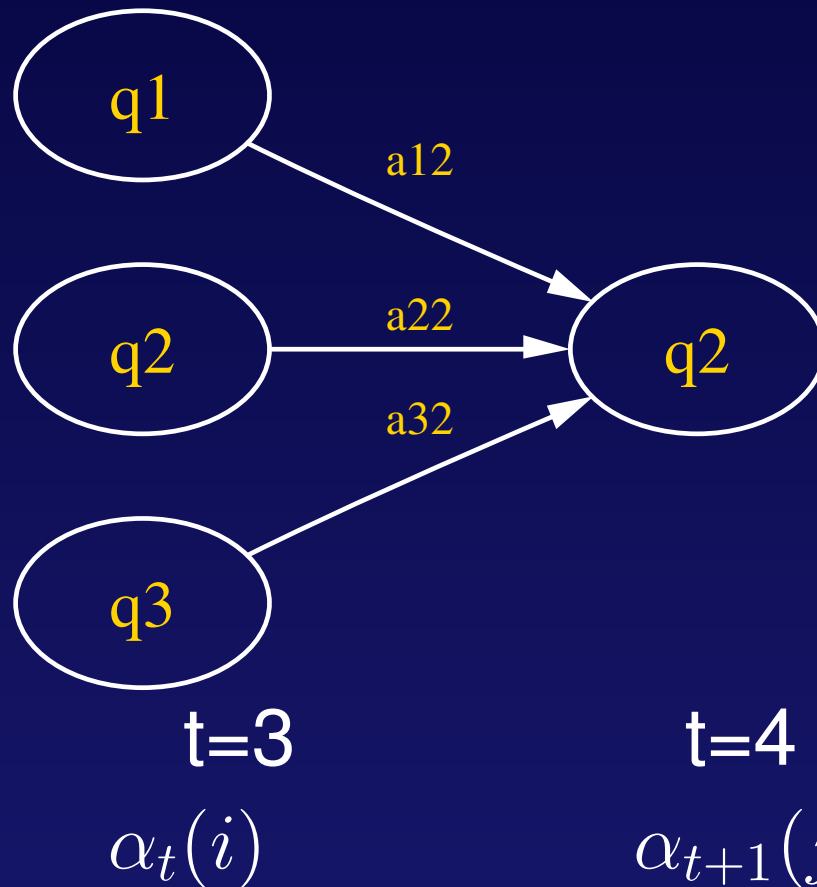


# Lattice of Observations



# Lattice of Observations

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad 1 \leq t \leq T - 1$$



# Forward Algorithm

- Key to efficiency is the lattice
- There are only  $N$  slots at each time slice of the lattice
- Each of the  $N$  slots is reached from the same  $N$  slots at the previous time slice
- Calculating  $\alpha_t(j)$  only involves the  $N$  previous values  $\alpha_{t-1}(j)$
- $O(N^2T)$  calculations vs brute force  $O(2TN^T)$  calculations



# Backward Variable

- The backward variable,  $\beta$ , defines the probability that we will observe a specific sequence in future  $(t + 1) \dots T$  observations

$$\beta_t(i) = P(O_{t+1}O_{t+2} \dots O_T | q_t = S_i, \lambda)$$

- $\beta_T$  is inductively computed:

$$\beta_T(i) = 1, 1 \leq i \leq N \text{ (arbitrary)}$$

$$\beta_t(j) = \sum_{i=1}^N \beta_{t+1}(i) a_{ij} b_j(O_{t+1}) \quad t = T - 1, t = T - 1, \dots, 1$$



# Solution to Problem 2

- Find the “optimal” state sequence associated with the given observation sequence.
- Let us maximize the expected number of correct individual states
- Let  $\gamma$  define the probability of being in a specific state at a specific time given an observation sequence and a model

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)}$$

- Solve for the individually most likely state  $q_t$  at time  $t$

$$q_t = \operatorname{argmax}[\gamma_t(i)], 1 \leq t \leq T$$



# Solution to Problem 2

- Constructing an “optimal” solution in this manner can result in sequences that do not exist
- Must factor in transitions to avoid constructing an invalid path
- To address this problem revise the optimality criteria
- Find the single best state sequence to maximize  $P(Q, O|\lambda)$
- Dynamic programming solution



# Viterbi Algorithm

- Find the *single* best state sequence given the observation
- The highest likelihood,  $\delta_t(i)$ , along a *single* path among all paths ending in state  $S_i$  at time  $t$  that accounts for the first  $t$  observations

$$\delta_t(i) = \max_{(q_1, q_2, \dots, q_{i-1})} P(q_1, q_2, \dots, q_t = i, O_1, O_2, \dots, O_t | \lambda)$$

- The variable  $\psi_t(j)$  is used to keep track of the “best path” ending in state  $j$  at time  $t$  by storing the state and observation that maximizes  $\delta_{t+1}(j)$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad 2 \leq t \leq T \quad 1 \leq j \leq N$$



# Viterbi Algorithm

- The path that leads to a maximum likelihood is computed by using the best path from the previous time step, the transition from it, and the likelihood of the observation

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t) \quad 2 \leq t \leq T \quad 1 \leq j \leq N$$

- The single most likely path has probability

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

- The single most likely path is reconstructed by backtracking through the variable  $\psi$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$



# Viterbi Algorithm vs Forward Algorithm

- Calculation of  $\delta$  and *alpha* are very similar
- Summations are replaced by maximizations
- Viterbi can be computed in fewer calculations but the two algorithms don't always return the same result in terms of which model best describes the given sequence of observations
  - ▶ Probability of paths: 0.05 0.05 0.05 0.4 0.05
  - ▶ Viterbi Algorithm:  $\max_{1 \leq i \leq N} [\delta_T(i)] = 0.4$
  - ▶ Forward Algorithm:  $\sum_{1 \leq i \leq N} [\alpha_T(i)] = 0.6$
- Despite this, Viterbi is “close enough” for most speech applications



# Solution to Problem 3

- How to adjust the model parameters  $\lambda = (A, B, \pi)$  to maximize the probability of the observation sequence given the model
- Cannot analytically solve for the model which maximizes the probability of the observation sequence
- Given an finite sequence of observations there is no optimal way to maximize the model parameters
- Locally maximize  $P(O|\lambda)$  using Expectation Maximization



# Baum–Welch Re–Estimation

- Let  $\xi_t(i, j)$ , the probability of being in state  $S_i$  at time  $t$  and  $S_j$  at time  $t + 1$  given the observation sequence

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

- $\xi$  can be written in terms of  $\alpha$  and  $\beta$  or in terms of  $\gamma$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1} \beta_{t+1}(j))}{P(O | \lambda)}$$

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

- These relations can be used to count even occurrences and provide a method of re–estimation for the model parameters



# Baum–Welch Re–Estimation

- Expected number of transitions made form state  $S_i$  (also the number of times state  $S_i$  was visited)

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

- Expected number of transitions made form  $S_i$  to  $S_j$

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



# Baum–Welch Re–Estimation

- Re–estimation of initial state distribution

$$\bar{\pi}_i = \gamma_1(i) = \text{expected number of times in } S_i \text{ at } t = 1$$

- Re–estimation of transition probability

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{expected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from } S_i}$$

- Re–estimation of observation probability

$$\bar{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j) O_{t=v_k}}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{expected visits to state } S_i \text{ and seeing symbol } v_k}{\text{expected number of times in } S_j}$$

.....

# Baum–Welch Re–Estimation

- Iterative process to calculate the maximum likelihood estimate of the HMM
- use initial/current model  $\lambda = (A, B, \pi)$  in the re–estimation equations to produce  $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$
- after re–estimation, two conditions can hold
  1. the model  $\lambda$  defines a critical point and  $\bar{\lambda} = \lambda$
  2. the model  $\bar{\lambda} = \lambda$  is more likely to have produced the observation sequence than the model  $\lambda$

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

- Iteratively using  $\bar{\lambda}$  in place of  $\lambda$  and repeating the re–estimation procedure allows us to improve the probability of  $O$  being observed from the model until some limiting point is reached.

# Implementation Issues

This talk has discussed theoretical issues of HMMs. Performing these calculations, even with trivial examples, can exceed the precision range of the computer. Perform calculations in  $\log$  land.

