

Statistical NLP: Hidden Markov Models



Updated 4/09



Markov Models

- Markov models are statistical tools that are useful for NLP because they can be used for part-of-speech-tagging applications
- Their first use was in modeling the letter sequences in works of Russian literature
- They were later developed as a general statistical tool
- More specifically, they model a sequence (perhaps through time) of random variables that are not necessarily independent
- They rely on two assumptions: **Limited Horizon** and **Time Invariant**



Markov Assumptions

- Let $X=(X_1, \dots, X_t)$ be a sequence of random variables taking values from some finite set $S=\{s_1, \dots, s_n\}$, the state space, the Markov properties are:
 - **Limited Horizon**: $P(X_t=s_k|X_1, \dots, X_{t-1})=P(X_t = s_k |X_{t-1})$ i.e., a state at position t only depends on the previous state.
 - **Time Invariant**:
 $P(X_{t+1}=s_k|X_t)=P(X_2 =s_k|X_1)$ i.e., the dependency does not change over time.
- If X possesses these properties, then X is said to be a Markov Chain



Markov Model parameters

- A Markov chain is described by:
 - Stochastic transition matrix
$$a_{ij} = p(X_{t+1} = s_j \mid X_t = s_i)$$
 - Probabilities of initial states of chain
$$\pi_i = p(X_1 = s_i)$$

there are obvious normalization constraints on \mathbf{a} and on π .

Finally a Markov Model is formally specified by a three-tuple (S, Π, A) where S is the set of states, and Π, A are the probabilities for the initial state and state transitions.

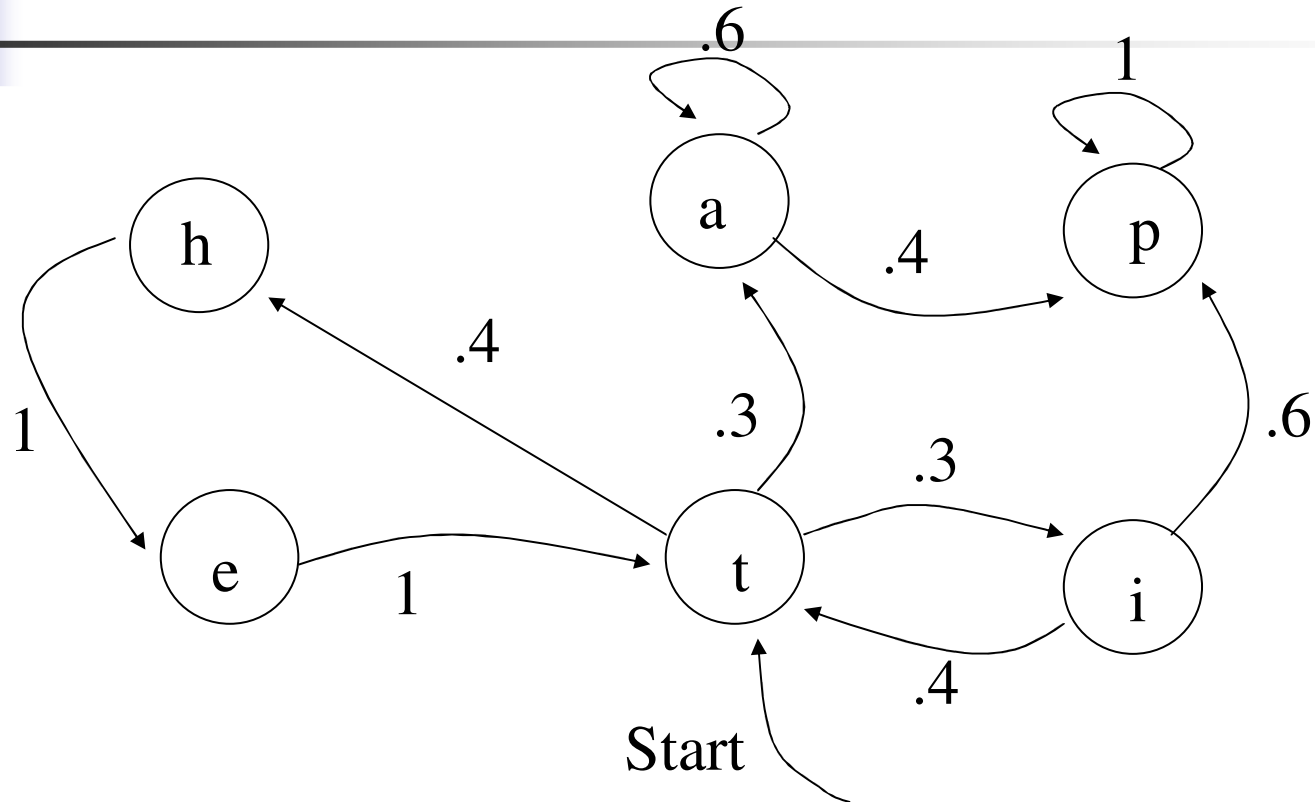


Probability of sequence of states

By Markov assumption

$$\begin{aligned} P(X_1, X_2, \dots, X_T) &= p(X_1) p(X_2|X_1) \dots p(X_T|X_{T-1}) \\ &= \pi_{X_1} \prod_{t=1}^{T-1} \mathbf{a}_{X_t, X_{t+1}} \end{aligned}$$

Example of a Markov Chain



Probability of sequence of states:

$$\begin{aligned} P(t, i, p) &= p(X_1 = t) p(X_2=i | X_1=t) p(X_3=p | X_2=i) \\ &= 1.0 * 0.3 * 0.6 = 0.18 \end{aligned}$$



Are n-gram models Markov Models?

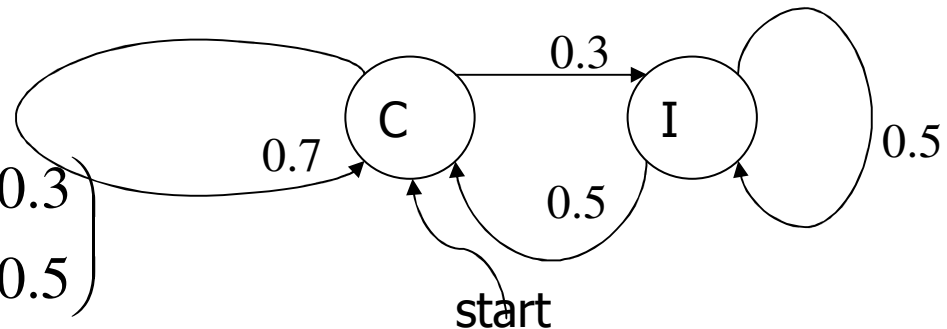
- Bi-gram models are obviously Markov Models.
- What about tri-gram, four-gram models etc.?

Stationary distribution of Markov Models

- What is the distribution of a Markov state $p(x_n)$ for very large n .
- Obviously, the initial state is forgotten.
- Stationary distribution $p^*A = p$ this is an eigenvalue problem $A^*p' = 1^*p'$

- Example:

- Here $A = \begin{pmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{pmatrix}$



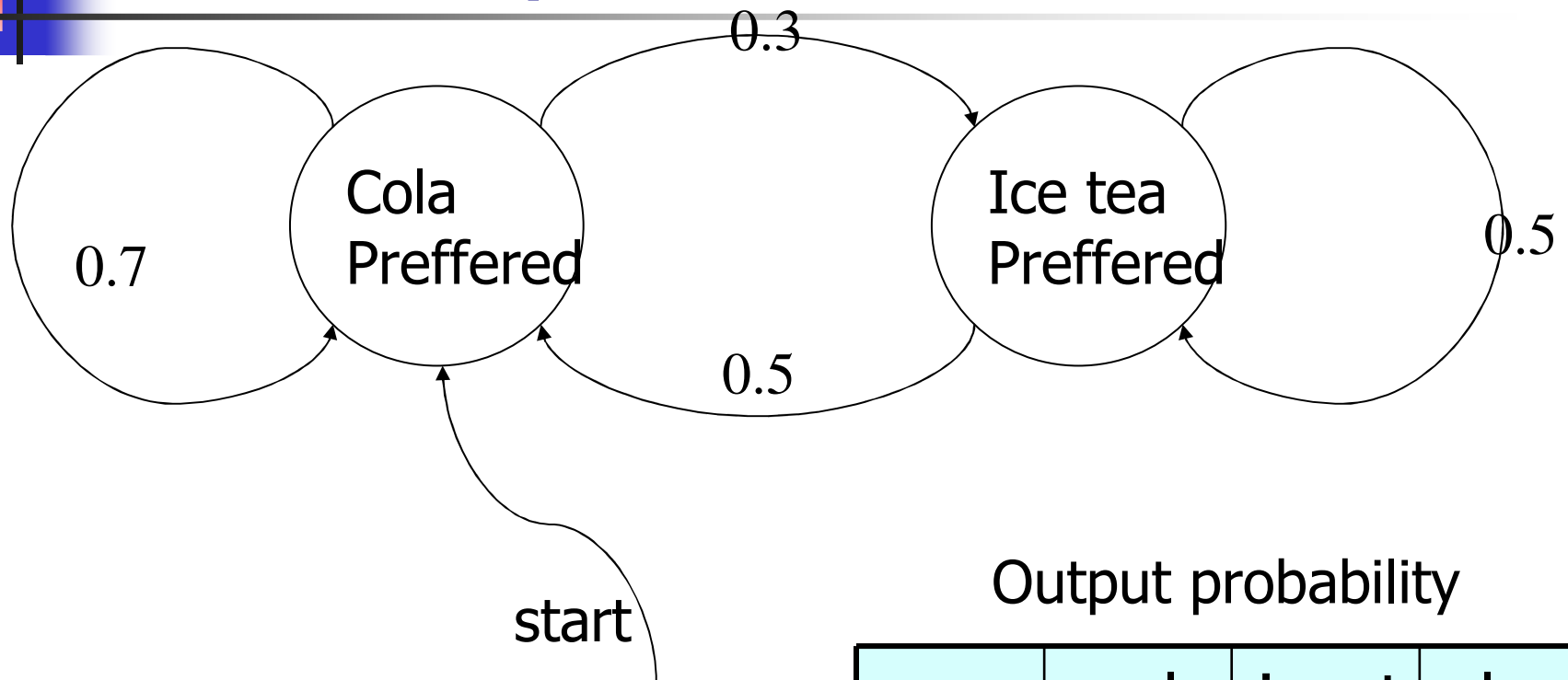
- Solution of eigenvalue problem: $p = (5/8 \quad 3/8)$



Hidden Markov Models (HMM)

- Here states are hidden.
- We have only ‘clues’ about states by the symbols each state outputs:
- $P(O_t = k \mid X_t = s_i, X_{t+1} = s_j) = b_{i,j,k}$

The crazy soft drink machine



Comment: for this machine the output really depends only on s_i namely $b_{ijk} = b_{ik}$

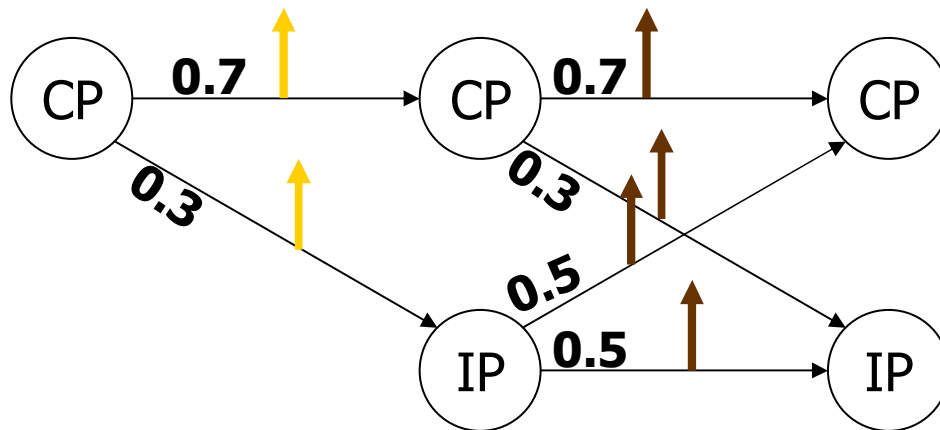
Output probability

	cola	ice_t	lem
CP	0.6	0.1	0.3
IP	0.1	0.7	0.2

The crazy soft drink machine – cont.

- What is the probability of observing {**lem**, **ice_t**}?
- Need to sum over all 4 possible paths that might be taken through the HMM:

$$0.7*0.3*0.7*0.1 + 0.7*0.3*0.3*0.1 + 0.3*0.3*0.5*0.7 + 0.3*0.3*0.5*0.7 = 0.084$$



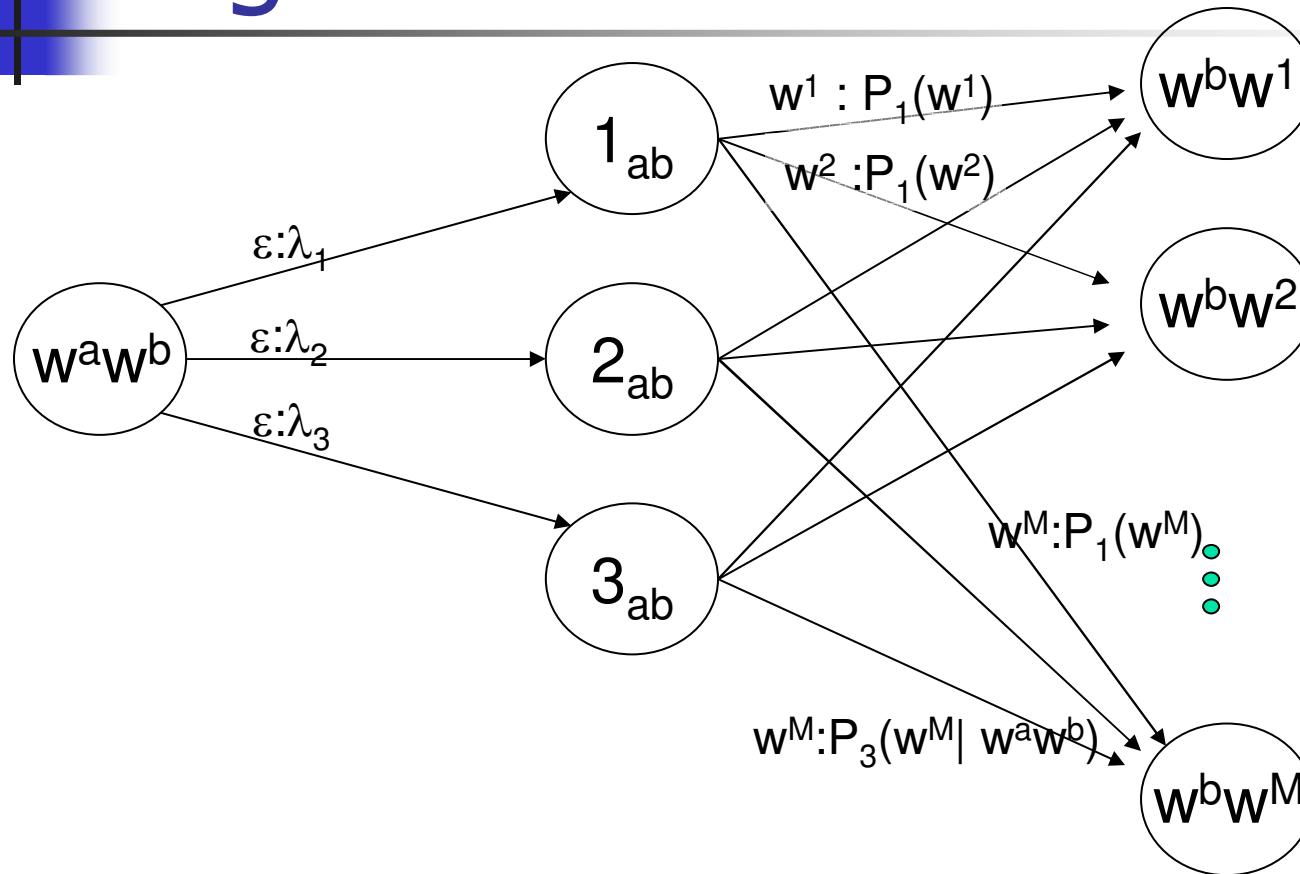
	cola	ice_t	lem
CP	0.6	0.1	0.3
IP	0.1	0.7	0.2



Why Use Hidden Markov Models?

- HMMs are useful when one can think of underlying events probabilistically generating surface events. Example: Part-of-Speech-Tagging or speech.
- HMMs can efficiently be trained using the EM Algorithm.
- Another example where HMMs are useful is in generating parameters for linear interpolation of n-gram models.

interpolating parameters for n-gram models





interpolating parameters for n-gram models – comments

- the HMM calculation of observing the sequence (w_{n-2}, w_{n-1}, w_n) is equivalent to
$$P_{\text{lin}}(w_n | w_{n-2}, w_{n-1}) = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n | w_{n-1}) + \lambda_3 P_3(w_n | w_{n-1}, w_{n-2})$$
- λ Transitions are special transitions that produce no output symbol
- In the above model each word pair (a, b) has a different HMM. This is relaxed by using *tied states*.



General Form of an HMM

- An HMM is specified by a five-tuple (S, K, Π, A, B) where S and K are the set of states and the output alphabet, and Π, A, B are the probabilities for the initial state, state transitions, and symbol emissions, respectively.
- When the sets S and K are obvious, HMM will be represented by a three-tuple (Π, A, B) .
- Given a specification of an HMM, we can simulate the running of a Markov process and produce an output sequence using the algorithm shown on the next page.
- More interesting than a simulation, however, is assuming that some set of data was generated by a HMM, and then being able to calculate probabilities and probable underlying state sequences.



A Program for a Markov Process modeled by HMM

```
t:= 1;  
Start in state  $s_i$  with probability  $\pi_i$  (i.e.,  $X_1=i$ )  
Forever do  
  Move from state  $s_i$  to state  $s_j$  with  
  probability  $a_{ij}$  (i.e.,  $X_{t+1} = j$ )  
  Emit observation symbol  $o_t = k$  with  
  probability  $b_{ijk}$   
  t:= t+1  
End
```

The Three Fundamental Questions for HMMs

- Given a model $\mu=(A, B, \Pi)$, how do we efficiently compute how likely a certain observation is, that is, $P(O | \mu)$
- Given the observation sequence O and a model μ , how do we choose a state sequence (X_1, \dots, X_{T+1}) that best explains the observations?
- Given an observation sequence O , and a space of possible models found by varying the model parameters $\mu = (A, B, \pi)$, how do we find the model that best explains the observed data?



Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu= (A, B, \Pi)$, we wish to know how to efficiently compute $P(O | \mu)$.
- For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$\begin{aligned} P(O | \mu) &= \sum_X P(O | X, \mu) P(X | \mu) \\ &= \sum_{X_1 \cdots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t} \end{aligned}$$

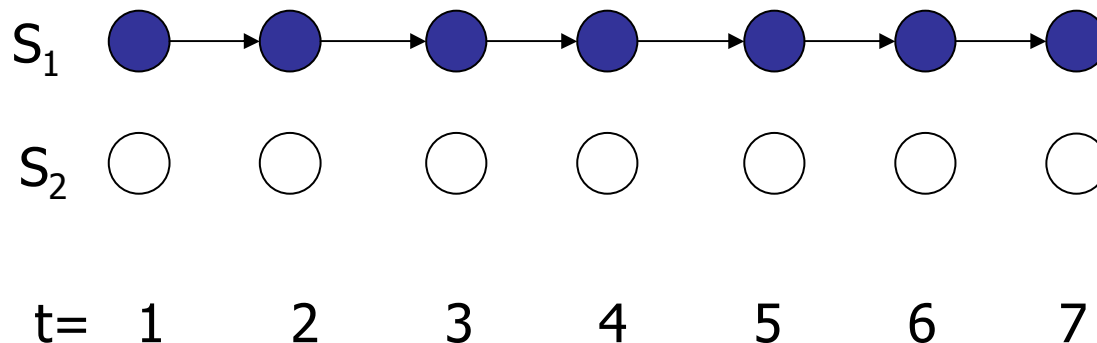
- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$.
- For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

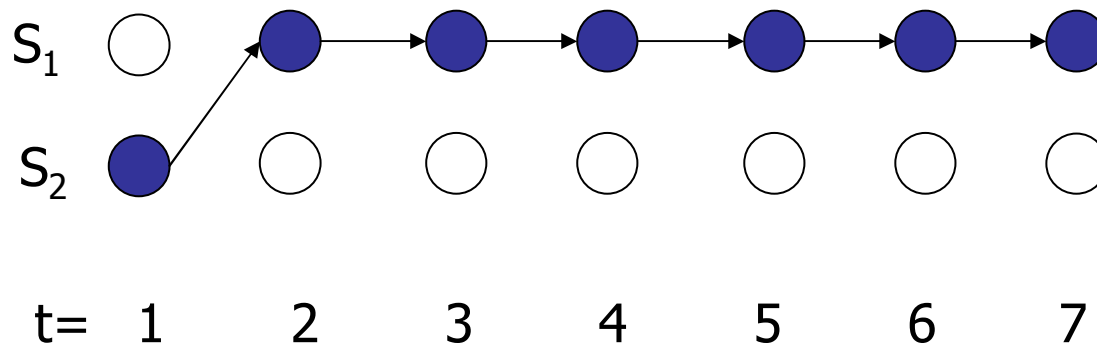


Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$. For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

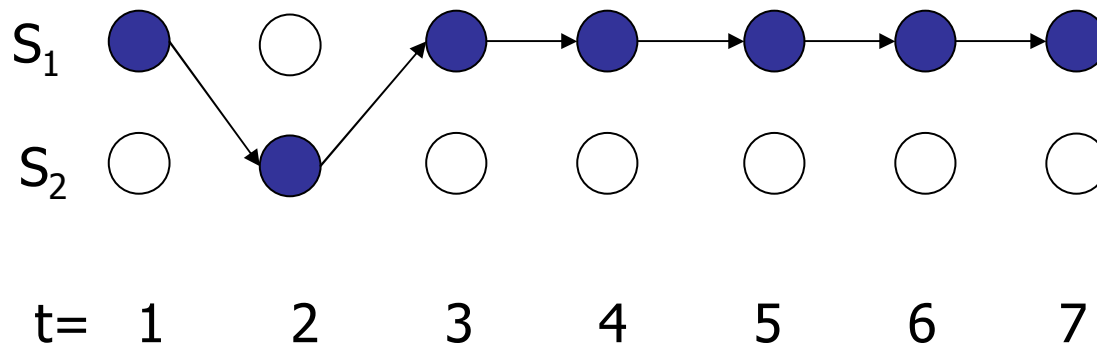


Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$. For any state sequence $X=(X_1, \dots, X_{T+1})$, we find

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

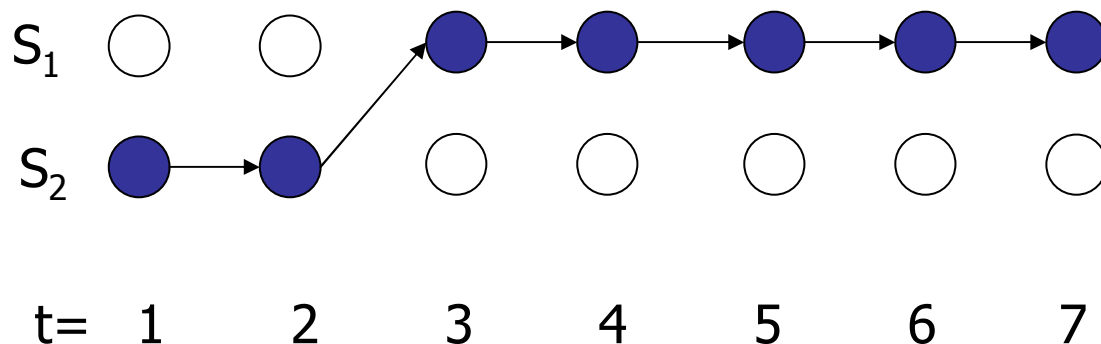


Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$. For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

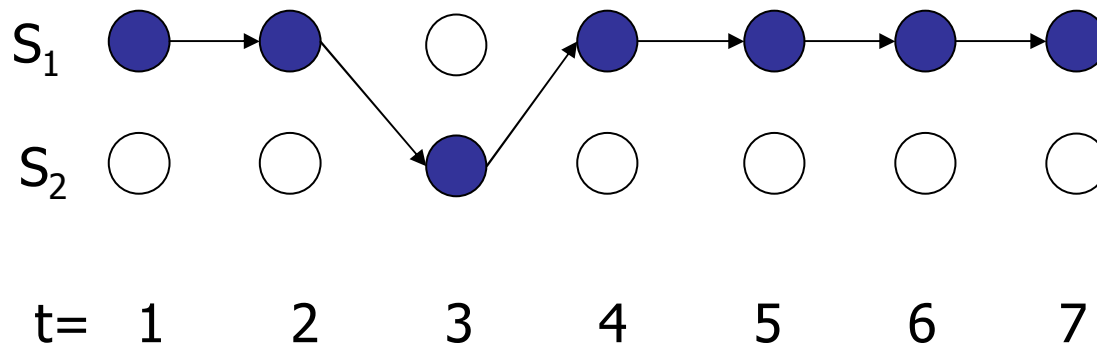


Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$. For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.

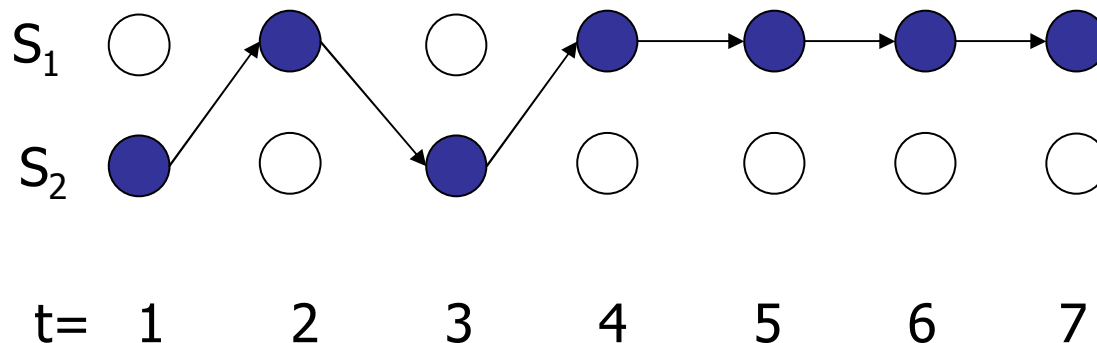


Finding the probability of an observation I

- Given the observation sequence $O=(o_1, \dots, o_T)$ and a model $\mu=(A, B, \Pi)$, we wish to know how to efficiently compute $P(O|\mu)$. For any state sequence $X=(X_1, \dots, X_{T+1})$, we find:

$$P(O|\mu) = \sum_X P(O|X, \mu)P(X|\mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

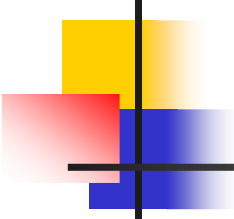
- This is simply the sum of the probability of the observation occurring according to each possible state sequence.
- Direct evaluation of this expression, however, is extremely inefficient.





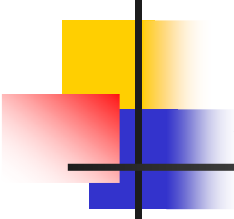
Finding the probability of an observation II

- In order to avoid this complexity, we can use *dynamic programming* or *memorization* techniques.
- In particular, we use *trellis* algorithms.
- We make a square array of states versus time and compute the probabilities of being at each state at each time in terms of the probabilities for being in each state at the preceding time.
- A trellis can record the probability of all initial subpaths of the HMM that end in a certain state at a certain time. The probability of longer subpaths can then be worked out in terms of the shorter subpaths.



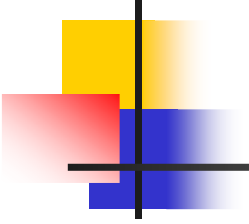
Finding the probability of an observation III: The *forward procedure*

- A *forward variable*, $\alpha_i(t) = P(o_1 o_2 \dots o_{t-1} X_t = i | \mu)$ is stored at (s_i, t) in the trellis and expresses the total probability of ending up in state s_i at time t .
- Forward variables are calculated as follows:
- *Initialization*: $\alpha_i(\mathbf{1}) = \pi_i, 1 \leq i \leq N$
- *Induction*: $\alpha_j(\mathbf{t+1}) = \sum_{i=1}^N \alpha_i(\mathbf{t}) a_{ij} b_{ij|o_t}, 1 \leq \mathbf{t} \leq T, 1 \leq j \leq N$
- *Total*: $P(\mathbf{O} | \mu) = \sum_{i=1}^N \alpha_i(\mathbf{T+1})$
- This algorithm requires $2N^2T$ multiplications (much less than the direct method which takes $(2T+1)N^{T+1}$)



Finding the probability of an observation IV: The ***backward procedure***

- The backward procedure computes ***backward variables*** which are the total probability of seeing the rest of the observation sequence given that we were in state s_i at time t .
- Backward variables are useful for the problem of parameter estimation.



Finding the probability of an observation V : The **backward procedure**

- Let $\beta_i(t) = P(o_t \dots o_T \mid X_t = i, \mu)$ be the backward variables.
- Backward variables can be calculated working backward through the trellis as follows:
- **Initialization**: $\beta_i(T+1) = 1, 1 \leq i \leq N$
- **Induction**: $\beta_i(t) = \sum_{j=1}^N a_{ij} b_{ijot} \beta_j(t+1), 1 \leq t \leq T, 1 \leq i \leq N$
- **Total**: $P(O \mid \mu) = \sum_{i=1}^N \pi_i \beta_i(1)$



Finding the probability of an observation VI: Combining *Forward* and *backward*

- $P(\mathbf{O}, X_t = i \mid \mu)$
 - $= P(o_1 \dots o_T, X_t = i \mid \mu)$
 - $= P(o_1 \dots o_{t-1}, X_t = i, o_t \dots o_T \mid \mu)$
 - $= P(o_1 \dots o_{t-1}, X_t = i \mid \mu) P(o_t \dots o_T \mid o_1 \dots o_{t-1}, X_t = i, \mu)$
 - $= P(o_1 \dots o_{t-1}, X_t = i \mid \mu) P(o_t \dots o_T \mid X_t = i, \mu)$
 - $= \alpha_i(\mathbf{t}) \beta_i(\mathbf{t}),$
- Total:
 $P(\mathbf{O} \mid \mu) = \sum_{i=1}^N \alpha_i(\mathbf{t}) \beta_i(\mathbf{t}), \mathbf{1} \leq \mathbf{t} \leq \mathbf{T}+1$



Finding the Best State Sequence I

- One method consists of finding the states individually:
- For each t , $1 \leq t \leq T+1$, we would like to find X_t that maximizes $P(X_t | O, \mu)$.
- Let $\gamma_i(t) = P(X_t = i | O, \mu) = P(X_t = i, O | \mu) / P(O | \mu) = (\alpha_i(t)\beta_i(t) / \sum_{j=1}^N \alpha_j(t)\beta_j(t))$
- The individually most likely state is

$$\mathbf{X}_t = \mathbf{argmax}_{1 \leq i \leq N} \gamma_i(t), \quad 1 \leq t \leq T+1$$

- This quantity maximizes the expected number of states that will be guessed correctly. However, it may yield a quite unlikely state sequence.

Finding the Best State Sequence

II: The Viterbi Algorithm

- The **Viterbi algorithm** efficiently computes the most likely state sequence.
- Commonly, we want to find the most likely complete path, that is: **$\mathit{argmax}_x P(X/O, \mu)$**
- To do this, it is sufficient to maximize for a fixed O: **$\mathit{argmax}_x P(X, O / \mu)$**
- We define
 $\delta_j(t) = \mathit{max}_{x_1 \dots x_{t-1}} P(X_1 \dots X_{t-1}, o_1 \dots o_{t-1}, X_t = j / \mu)$
- **$\psi_j(t)$** records the node of the incoming arc that led to this most probable path.



Finding the Best State Sequence III: The Viterbi Algorithm

The Viterbi Algorithm works as follows:

- **Initialization:** $\delta_j(\mathbf{1}) = \pi_j \quad \mathbf{1} \leq j \leq N$
- **Induction:** $\delta_j(\mathbf{t}+1) = \max_{\mathbf{1} \leq i \leq N} \delta_i(\mathbf{t}) a_{ij} b_{ijot} \quad \mathbf{1} \leq j \leq N$

Store backtrace:

$$\psi_j(\mathbf{t}+1) = \operatorname{argmax}_{\mathbf{1} \leq i \leq N} \delta_i(\mathbf{t}) a_{ij} b_{ijot} \quad \mathbf{1} \leq j \leq N$$

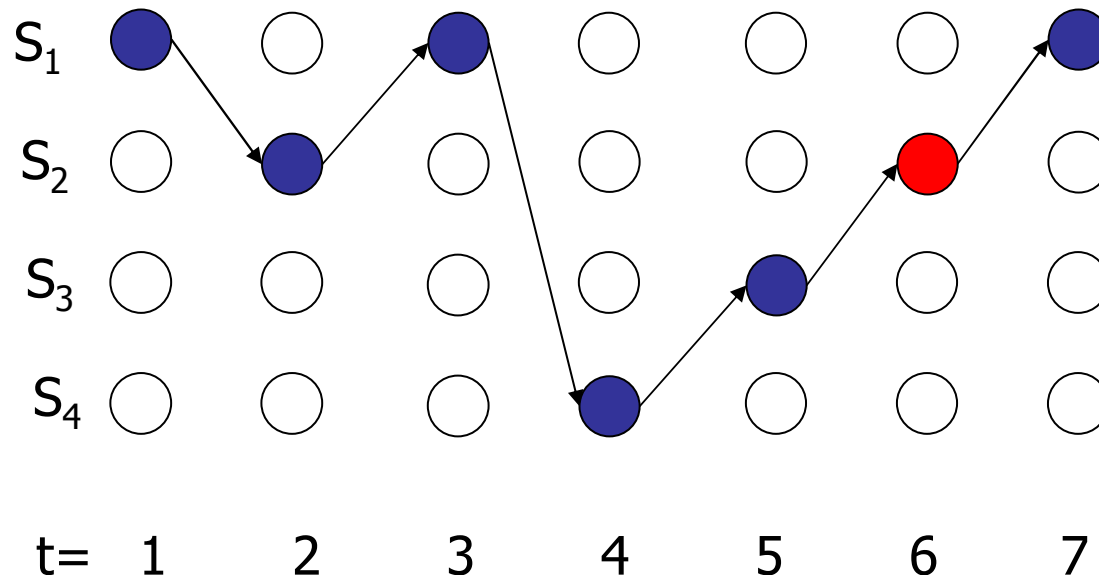
- **Termination and path readout:**

$$\hat{X}_{T+1} = \operatorname{argmax}_{\mathbf{1} \leq i \leq N} \delta_i(T+1)$$

$$X_t = \psi_{X_{t+1}}(\mathbf{t}+1)$$

$$\hat{P}(X) = \max_{\mathbf{1} \leq i \leq N} \delta_i(T+1)$$

Finding the Best State Sequence IV: Visualization



- $\delta_2(\mathbf{t=6})$ = probability of reaching state 2 at $t=6$ by the most probable path (marked) through state 2 at $t=6$
- $\psi_2(\mathbf{t=6}) = \mathbf{3}$ is the state from preceding state 2 at $t=6$ on the most probable path through state 2 at $t=6$



Parameter Estimation I

- Given a certain observation sequence, we want to find the values of the model parameters $\mu=(A, B, \pi)$ which best explain what we observed.
- Using Maximum Likelihood Estimation, we can find the values that maximize $P(\mathbf{O} / \mu)$, i.e. ***argmax*** $\mu P(\mathbf{O}_{training} / \mu)$
- There is no known analytic method to choose μ to maximize $P(\mathbf{O} / \mu)$. However, we can locally maximize it by an iterative hill-climbing algorithm known as **Baum-Welch** or **Forward-Backward algorithm**. (special case of the EM Algorithm)



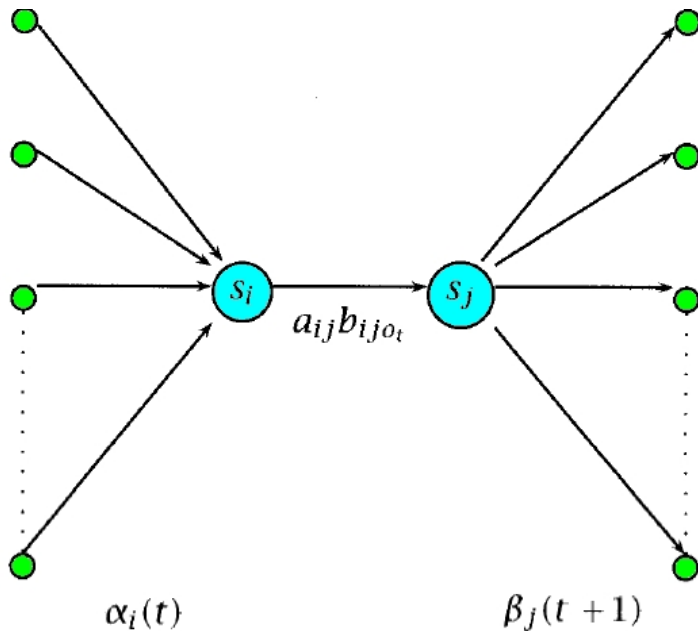
Parameter Estimation II

- We don't know what the model is, but we can work out the probability of the observation sequence using some (perhaps randomly chosen) model.
- Looking at that calculation, we can see which state transitions and symbol emissions were probably used the most.
- By increasing the probability of those, we can choose a revised model which gives a higher probability to the observation sequence.

Parameter Estimation III

- Define $P_t(i, j)$ as the probability of traversing a certain arc at time t given observation sequence:

$$\begin{aligned}
 p_t(i, j) &= P(X_t = i, X_{t+1} = j | O, \mu) \\
 &= \frac{P(X_t = i, X_{t+1} = j, O | \mu)}{P(O | \mu)}
 \end{aligned}$$





Parameter Estimation IV

- The probability of traversing a certain arc at time t given observation sequence can be expressed as:

$$p_t(i, j) = \frac{\alpha_i(t) a_{ij} b_{ij} o_t \beta_j(t+1)}{\sum_{m=1}^N \alpha_m(t) \beta_m(t)}$$

- The probability of leaving state i at time t

$$\gamma_i(t) = \sum_{j=1 \dots N} p_t(i, j)$$



Parameter Estimation V

$$\hat{\pi}_i = \gamma_i(1)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T p_t(i, j)}{\sum_{t=1}^T \gamma_i(t)}$$

$$\hat{b}_{ijk} = \frac{\sum_{\{t: o_t=k, 1 \leq t \leq T\}} p_t(i, j)}{\sum_{t=1}^T p_t(i, j)}$$



Parameter Estimation VI

- Namely, given an HMM model $\mu=(A, B, \pi)$ the Baum Welsh algorithm gives new values for the model parameters $\hat{\mu}=(\hat{A}, \hat{B}, \hat{\pi})$.
- One may prove that $P(O | \mu) \geq P(O | \hat{\mu})$ which is a general property of EM.
- The process is repeated until convergence.
- It is prone to local maxima, another property of EM.