CSE 150A-250A AI: Probabilistic Models

Lecture 11

Fall 2025

Trevor Bonjour
Department of Computer Science and Engineering
University of California, San Diego

Slides adapted from previous versions of the course (Prof. Lawrence, Prof. Alvarado, Prof Berg-Kirkpatrick)

Agenda

Review

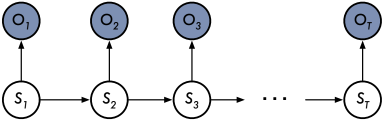
Forward Algorithm

Viterbi algorithm

Review

Hidden Markov models

- **Belief network**



observations $o_t \in \{1, 2, \ldots, m\}$

states $s_t \in \{1, 2, \ldots, n\}$

- **Parameters**

$$a_{ij} = P(S_{t+1} = j | S_t = i) \qquad \boxed{n \times n \text{ transition matrix}}$$

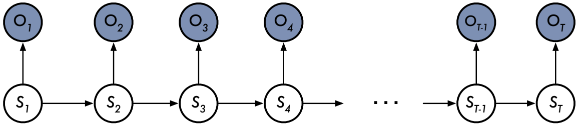
$$b_{ik} = P(O_t = k | S_t = i) \qquad \boxed{n \times m \text{ emission matrix}}$$

$$\pi_i = P(S_1 = i) \qquad \boxed{n \times 1 \text{ initial state distribution}}$$

- **Notation**

Sometimes we'll write $b_i(k) = b_{ik}$ to avoid double subscripts.

Key computations in HMMs



Inference

1. How to compute the likelihood $P(o_1, o_2, \ldots, o_T)$?
2. How to compute the most likely hidden states $\text{argmax}_{\vec{s}} P(\vec{s}|\vec{o})$?
3. How to update beliefs by computing $P(s_t|o_1, o_2, \ldots, o_t)$?

Learning

How to estimate parameters $\{\pi_i, a_{ij}, b_{ik}\}$ that maximize the log-likelihood of observed sequences?

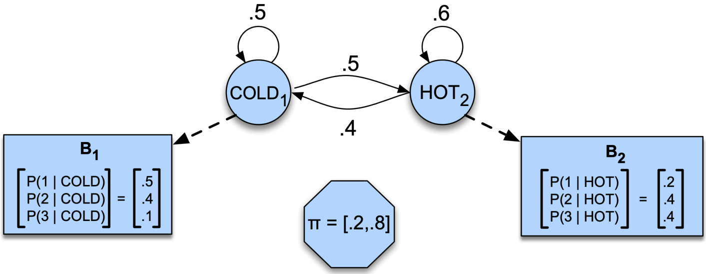
Example[1] - Set Up

Imagine that you are a climatologist in the year 2799 studying the history of global warming. You cannot find any records of the weather in Baltimore, Maryland, for the summer of 2020, but you do find Jason Eisner's diary, which lists how many ice creams (1, 2 or 3) Jason ate every day that summer. Our goal is to use these observations to estimate the temperature every day. We'll simplify this weather task by assuming there are only two kinds of days: cold (C) and hot (H).

[1] Eisner, J. 2002. An interactive spreadsheet for teaching the forward-backward algorithm.

Example

- Two hidden states (Weather): $\{H, C\}$
- Observations (Ice creams): $\{1, 2, 3\}$



Computing likelihood via enumeration

Sequence of observations: $o_1 = 3, o_2 = 1, o_3 = 3$

$$P(\vec{o}) = \sum_{s_1,s_2,s_3} P(o_1, o_2, o_3, s_1, s_2, s_3)$$
$$= \sum_{s_1,s_2,s_3} P(s_1) \cdot P(o_1|s_1) \cdot P(s_2|s_1) \cdot P(o_2|s_2) \cdot P(s_3|s_2) \cdot P(o_3|s_3)$$

Q. What is the number of all possible state sequences for the three observations (3, 1, 3)?

A. 2 **B.** 4 **C.** 8 **D.** 16 **E.** 32

Complexity: $\mathcal{O}(N^T)$ where N = number of states, T = sequence length

Computing likelihood via enumeration

Possible state sequences:

$$C \to C \to C : P(C) \cdot P(3|C) \cdot P(C|C) \cdot P(1|C) \cdot P(C|C) \cdot P(3|C)$$

$$C \to C \to H : P(C) \cdot P(3|C) \cdot P(C|C) \cdot P(1|C) \cdot P(H|C) \cdot P(3|H)$$

$$C \to H \to C : P(C) \cdot P(3|C) \cdot P(H|C) \cdot P(1|H) \cdot P(C|H) \cdot P(3|C)$$

$$C \to H \to H : P(C) \cdot P(3|C) \cdot P(H|C) \cdot P(1|H) \cdot P(H|H) \cdot P(3|H)$$

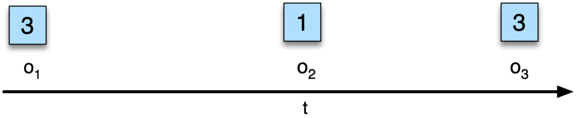
$$H \to C \to C : P(H) \cdot P(3|H) \cdot P(C|H) \cdot P(1|C) \cdot P(C|C) \cdot P(3|C)$$

$$H \to C \to H : P(H) \cdot P(3|H) \cdot P(C|H) \cdot P(1|C) \cdot P(H|C) \cdot P(3|H)$$

$$H \to H \to C : P(H) \cdot P(3|H) \cdot P(H|H) \cdot P(1|H) \cdot P(C|H) \cdot P(3|C)$$

$$H \to H \to H : P(H) \cdot P(3|H) \cdot P(H|H) \cdot P(1|H) \cdot P(H|H) \cdot P(3|H)$$

Example



Forward Algorithm

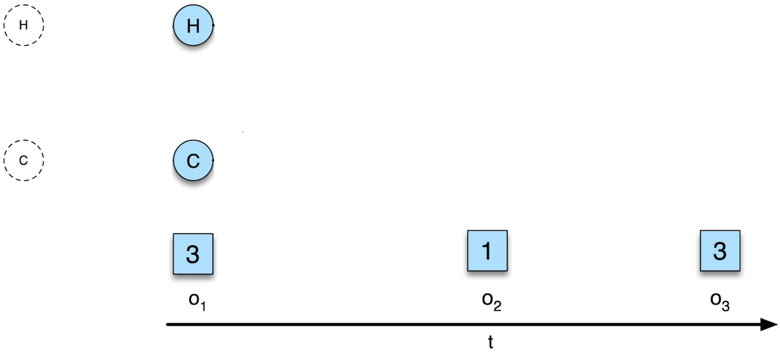
Computing $P(o_1, o_2, \ldots, o_t, S_t = i)$

Definition

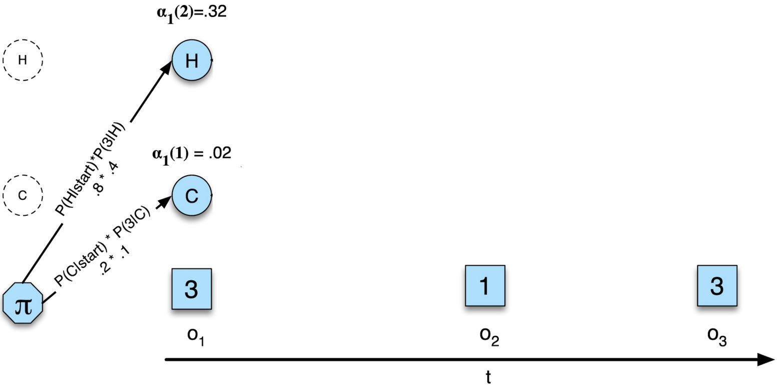
For a particular sequence of observations $\{o_1, o_2, \ldots, o_T\}$, define the matrix with elements:

$$\alpha_{it} = P(o_1, o_2, \ldots, o_t, S_t = i)$$

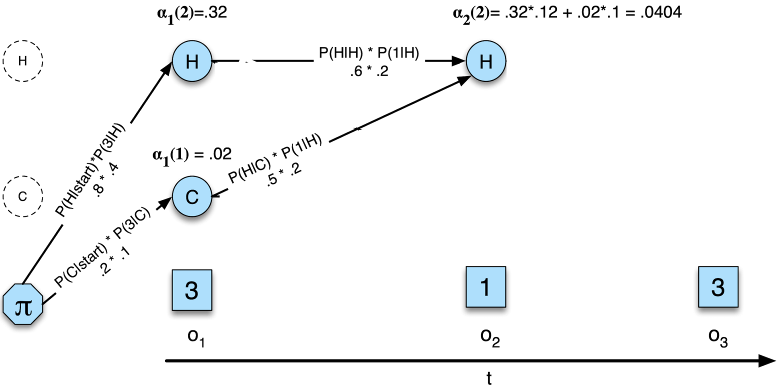
Example - Forward Algorithm



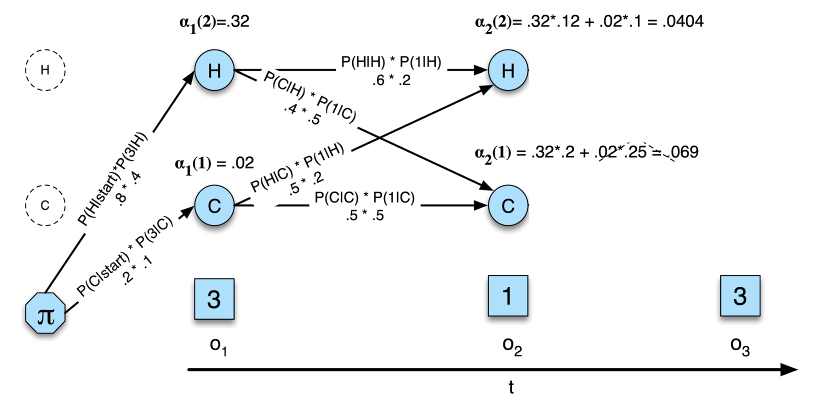
Example - Forward Algorithm



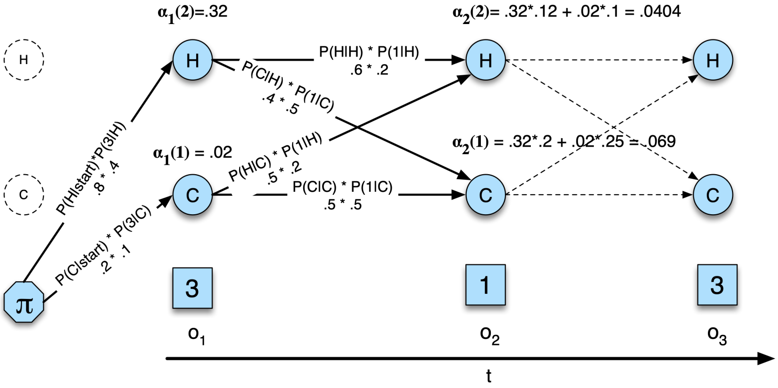
Example - Forward Algorithm



Example - Forward Algorithm



Example - Forward Algorithm



Computing $P(o_1, o_2, \ldots, o_t, S_t = i)$

- **Definition**

  For a particular sequence of observations $\{o_1, o_2, \ldots, o_T\}$, define the matrix with elements:

$$\alpha_{it} = P(o_1, o_2, \ldots, o_t, S_t = i) \qquad n \text{ rows} \left\{ \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1,T-1} & \alpha_{1T} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2,T-1} & \alpha_{2T} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n,T-1} & \alpha_{nT} \end{bmatrix} \right.$$

  α_{it} represents the probability of being in state **i** after seeing the first **t** observations.

- **First column ($t = 1$)**

$$\begin{aligned} \alpha_{i1} &= P(o_1, S_1 = i) \\ &= P(S_1 = i)\, P(o_1 | S_1 = i) && \boxed{\text{product rule}} \\ &= \pi_i\, b_i(o_1) && \boxed{\text{CPTs}} \end{aligned}$$

Computing $\alpha_{i2} = P(o_1, o_2, S_2 = i)$

- Next column ($t = 2$)

$$
\begin{aligned}
\alpha_{j,2} &= P(o_1, o_2, S_2 = j) \\
&= \sum_{i=1}^{n} P(o_1, o_2, S_1 = i, S_2 = j) \quad \boxed{\text{marginalization}} \\
&= \sum_{i=1}^{n} \Big[P(o_1, S_1 = i) \cdot \\
&\qquad P(S_2 = j | o_1, S_1 = i) \cdot \\
&\qquad P(o_2 | o_1, S_1 = i, S_2 = j) \Big] \quad \boxed{\text{product rule}} \\
&= \sum_{i=1}^{n} \Big[P(o_1, S_1 = i)\, P(S_2 = j | S_1 = i)\, P(o_2 | S_2 = j) \Big] \quad \boxed{\text{CI}} \\
&= \sum_{i=1}^{n} \alpha_{i1}\, a_{ij}\, b_j(o_2) \quad \boxed{\text{CPTs}}
\end{aligned}
$$

Computing $\alpha_{it} = P(o_1, o_2, \ldots, o_t, S_t = i)$

- Next columns ($t > 1$)

$$
\begin{aligned}
\alpha_{j,t+1} &= P(o_1, o_2, \ldots, o_{t+1}, S_{t+1} = j) \\
&= \sum_{i=1}^{n} P(o_1, o_2, \ldots, o_{t+1}, S_t = i, S_{t+1} = j) \quad \boxed{\text{marginalization}} \\
&= \sum_{i=1}^{n} \Big[P(o_1, o_2, \ldots, o_t, S_t = i) \cdot \\
&\qquad P(S_{t+1} = j | o_1, o_2, \ldots, o_t, S_t = i) \cdot \\
&\qquad P(o_{t+1} | o_1, o_2, \ldots, o_t, S_t = i, S_{t+1} = j) \Big] \quad \boxed{\text{product rule}} \\
&= \sum_{i=1}^{n} \Big[P(o_1, o_2, \ldots, o_t, S_t = i) \, P(S_{t+1} = j | S_t = i) \, P(o_{t+1} | S_{t+1} = j) \Big] \quad \boxed{\text{CI}} \\
&= \sum_{i=1}^{n} \alpha_{it} \, a_{ij} \, b_j(o_{t+1}) \quad \boxed{\text{CPTs}}
\end{aligned}
$$

Forward algorithm

The forward algorithm fills in the matrix of α_{it} elements one column at a time:

$$\boxed{\begin{aligned} \alpha_{i1} &= \pi_i \, b_i(o_1) \\ \alpha_{j,t+1} &= \sum_{i=1}^{n} \alpha_{it} \, a_{ij} \, b_j(o_{t+1}) \end{aligned}}$$

Computing the likelihood $P(o_1, o_2, \ldots, o_T)$

$$
n \text{ rows} \left\{ \begin{bmatrix}
\alpha_{11} & \alpha_{12} & \cdots & \alpha_{1,T-1} & \alpha_{1T} \\
\alpha_{21} & \alpha_{22} & \cdots & \alpha_{2,T-1} & \alpha_{2T} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n,T-1} & \alpha_{nT}
\end{bmatrix} \right.
$$

$$
\begin{aligned}
& P(o_1, o_2, \ldots, o_T) \\
&= \sum_{i=1}^{n} P(o_1, o_2, \ldots, o_T, s_T = i) \qquad \boxed{\text{marginalization}} \\
&= \sum_{i=1}^{n} \alpha_{iT} \qquad \boxed{\text{sum of last column}}
\end{aligned}
$$

Computing the likelihood $P(o_1, o_2, \ldots, o_T)$

Sum!

$$n \text{ rows} \left\{ \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1,T-1} & \alpha_{1T} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2,T-1} & \alpha_{2T} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n,T-1} & \alpha_{nT} \end{bmatrix} \right.$$

$$\begin{aligned} &P(o_1, o_2, \ldots, o_T) \\ &\quad = \sum_{i=1}^{n} P(o_1, o_2, \ldots, o_T, s_T = i) && \boxed{\text{marginalization}} \\ &\quad = \sum_{i=1}^{n} \alpha_{iT} && \boxed{\text{sum of last column}} \end{aligned}$$

Computing the likelihood $P(o_1, o_2, \ldots, o_T)$

Sum!

$$n \text{ rows} \left\{ \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1,T-1} & \alpha_{1T} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2,T-1} & \alpha_{2T} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n,T-1} & \alpha_{nT} \end{bmatrix} \right.$$

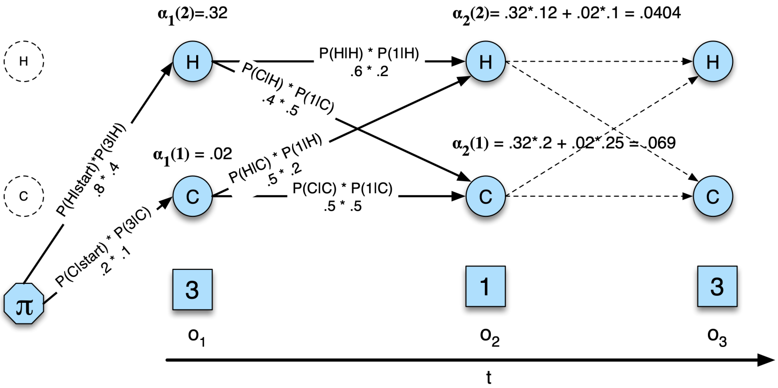
$$\begin{aligned} & P(o_1, o_2, \ldots, o_T) \\ & \quad = \sum_{i=1}^{n} P(o_1, o_2, \ldots, o_T, s_T = i) \\ & \quad = \sum_{i=1}^{n} \alpha_{iT} \end{aligned}$$

marginalization

sum of last column

Example - Forward Algorithm

$$
\begin{aligned}
\alpha_{i1} &= \pi_i \, b_i(o_1) \\
\alpha_{j,t+1} &= \sum_{i=1}^{n} \alpha_{it} \, a_{ij} \, b_j(o_{t+1})
\end{aligned}
$$



Question

What is the running time for Forward algorithm?

A. $O(n)$

B. $O(n^2)$

C. $O(Tn^2)$

D. $O(T^2n^4)$

E. $O(n^T)$

Pseudocode

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$\quad forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

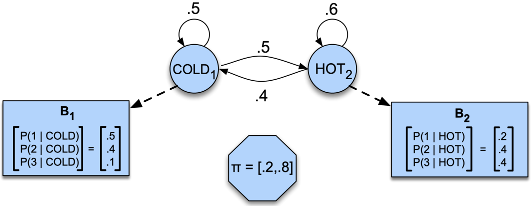
$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T] \qquad \text{; termination step}$$

return *forwardprob*

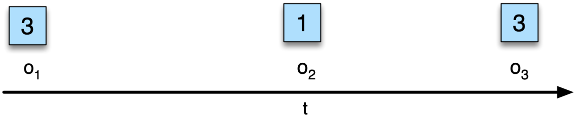
Figure A.7 The forward algorithm, where $forward[s,t]$ represents $\alpha_t(s)$.

Viterbi algorithm

Viterbi Algorithm



There are many paths through the hidden states (H and C) that lead to the given sequence, but they do not all have the same probability.



The most likely sequence of hidden states

The **Viterbi algorithm** allows us to efficiently compute the most probable path using *dynamic programming*.

$$\{s_1^*, s_2^*, \ldots, s_T^*\}$$

$$= \quad \text{argmax}_{s_1, s_2, \ldots, s_T} P(s_1, s_2, \ldots, s_T | o_1, o_2, \ldots, o_T)$$

The matrix ℓ^*

For a particular sequence of observations $\{o_1, o_2, \dots, o_T\}$, we define the following matrix:

$$\ell^*_{it} = \max_{s_1, s_2, \dots, s_{t-1}} P(s_1, s_2, \dots, s_{t-1}, S_t = i, o_1, o_2, \dots, o_t)$$

Q. What does ℓ^* mean, in English?

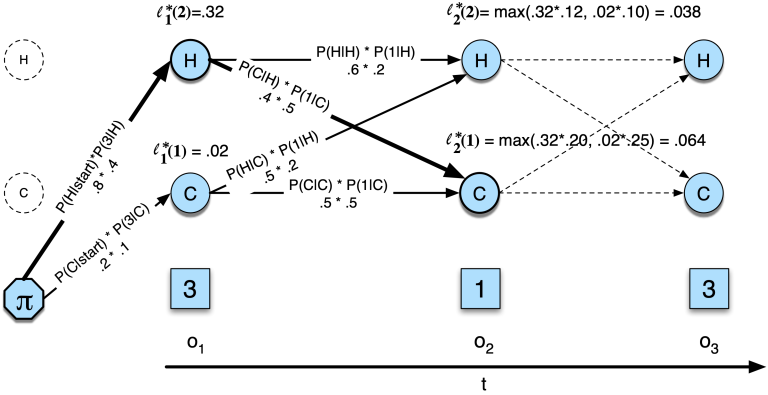
A. The probability of the most likely $s_1, \cdots, s_t$ given $o_1, \cdots, o_t$

B. The most likely state at time t

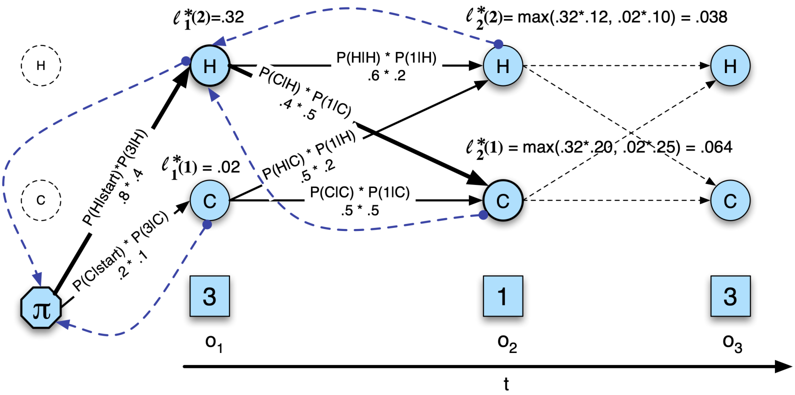
C. The probability of the most likely $s_1, \cdots, s_t$ that ends in state $s_t = i$ and explains $o_1, \cdots, o_t$

D. The probability of most likely states $s_1, \cdots, s_t$ that explains observations $o_1, \cdots, o_t$

Example - Viterbi (Fill ℓ^*)



Example - Viterbi (Backtrack through ℓ^*)



Viterbi Algorithm

In practice, we switch to log probabilities:

- Optimization stays the same (doesn't change our outcome)
- Allows us to compute sums instead of products
- Prevent underflow

That’s all folks!