CS 486/686
Hidden Markov Models II

Yuntian Deng

Lecture 11

RN 14.2.2

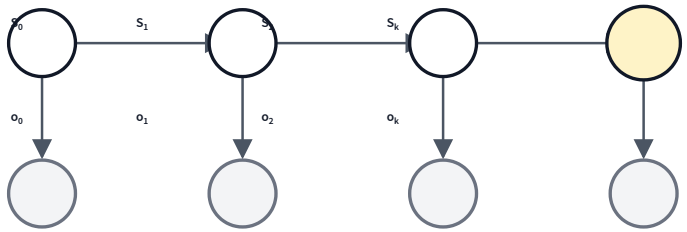
Learning goals

- Compute the **smoothing** probability for a time step in an HMM.
- **Justify** each step in the derivation of the smoothing formula.
- Describe the **forward-backward** algorithm.
- Describe the **Viterbi** algorithm.

Recap: filtering (from L10)

From past evidence $o_{0:k}$, filtering gives the posterior over the *current* state:

$$f_{0:k} = P(S_k|o_{0:k}) = \alpha\, P(o_k|S_k) \sum_{s_{k-1}} P(S_k|s_{k-1})\, f_{0:(k-1)}$$

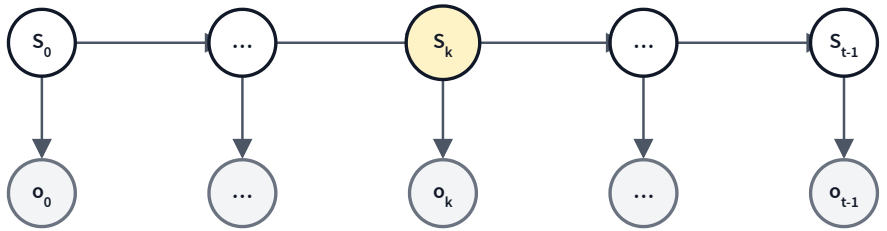


- Yellow = query state S_k (the *present*).
- Gray = observed evidence.
- Cost: $O(k)$ ops via the recursive predict-update.

Today: a query about a **past** state, given all the evidence we have.

Smoothing: query a past state

$$P(S_k | o_{0:(t-1)}), \quad 0 \le k \le t-1$$



Filtering (L10)

$P(S_k | o_{0:k})$ — the present state from past evidence.

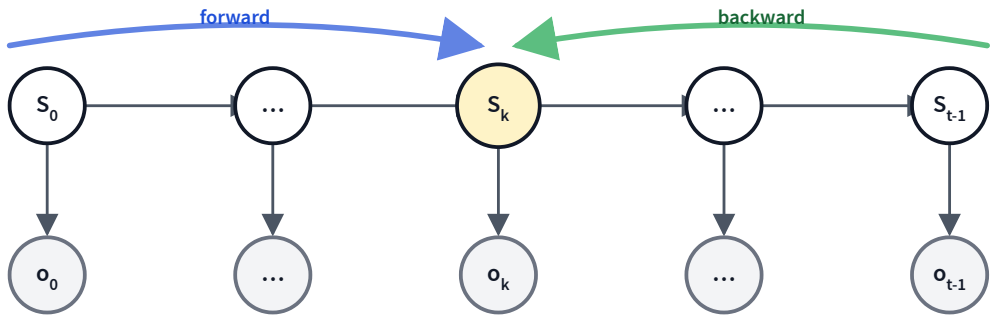
Smoothing (today)

$P(S_k | o_{0:(t-1)}),\ k < t-1$ — a past state from all evidence (past *and* future).

Smoothing = forward × backward

Split the evidence around S_k into past and future:

$$P(S_k|o_{0:(t-1)}) = \alpha \underbrace{P(S_k|o_{0:k})}_{\textbf{forward } f_{0:k}} \underbrace{P(o_{(k+1):(t-1)}|S_k)}_{\textbf{backward } b_{(k+1):(t-1)}}$$



Two sweeps meet at S_k; multiply pointwise and normalize.

Backward recursion

Compute $b_{(k+1):(t-1)} \triangleq P(o_{(k+1):(t-1)} | S_k)$ right-to-left.

Base case (k = t − 1, no future observations)

$$b_{t:(t-1)} = \langle 1, 1 \rangle$$

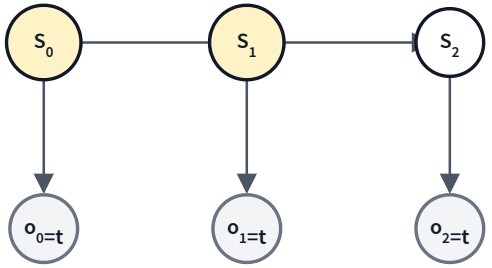
Recursive case (k < t − 1)

$$b_{(k+1):(t-1)} = \sum_{s_{k+1}} P(o_{k+1} \mid s_{k+1})\, b_{(k+2):(t-1)}\, P(s_{k+1} \mid S_k)$$

Same factor sizes as forward recursion — total cost $O(t)$ for the whole backward pass.

No normalization here: b is a likelihood, not a probability. Normalize *after* multiplying $f \times b$.

A smoothing example: setup



Umbrella story, given $O_0 = O_1 = O_2 = t$. Compute the *smoothed* beliefs:

- $P(S_0|o_{0:2})$: did it rain on **day 0**?
- $P(S_1|o_{0:2})$: did it rain on **day 1**?

From L10 we already have $f_{0:0} = \langle 0.818, 0.182 \rangle$ and $f_{0:1} = \langle 0.883, 0.117 \rangle$.

Backward pass: compute $b_{2:2}$, then $b_{1:2}$

$$b_{2:2} = P(o_2|S_1) = \sum_{s_2} P(o_2|s_2)\, b_{3:2}\, P(s_2|S_1) \text{ with } b_{3:2} = \langle 1, 1 \rangle\text{:}$$

$$= 0.9 \cdot 1 \cdot \langle 0.7, 0.3 \rangle + 0.2 \cdot 1 \cdot \langle 0.3, 0.7 \rangle$$

$$= \langle 0.63, 0.27 \rangle + \langle 0.06, 0.14 \rangle = \langle \mathbf{0.69}, \mathbf{0.41} \rangle$$

$$b_{1:2} = P(o_{1:2}|S_0) = \sum_{s_1} P(o_1|s_1)\, b_{2:2}\, P(s_1|S_0)\mathbf{:}$$

$$= 0.9 \cdot 0.69 \cdot \langle 0.7, 0.3 \rangle + 0.2 \cdot 0.41 \cdot \langle 0.3, 0.7 \rangle$$

$$= \langle 0.4347, 0.1863 \rangle + \langle 0.0246, 0.0574 \rangle = \langle \mathbf{0.4593}, \mathbf{0.2437} \rangle$$

Combine: $P(S_k|o_{0:2}) = \alpha\, f_{0:k}\, b_{(k+1):2}$

$$P(S_1|o_{0:2}) = \alpha\, f_{0:1} \cdot b_{2:2}:$$

$$= \alpha\, \langle 0.883, 0.117\rangle \cdot \langle 0.69, 0.41\rangle = \alpha\, \langle 0.6093, 0.0480\rangle = \langle \mathbf{0.927}, \mathbf{0.073}\rangle$$

$$P(S_0|o_{0:2}) = \alpha\, f_{0:0} \cdot b_{1:2}:$$

$$= \alpha\, \langle 0.818, 0.182\rangle \cdot \langle 0.4593, 0.2437\rangle = \alpha\, \langle 0.3757, 0.0444\rangle = \langle \mathbf{0.894}, \mathbf{0.106}\rangle$$

	$P(S_1 = T \mid \ldots)$
Filtering with $o_{0:1}$ only	0.883
Smoothing with $o_{0:2}$	**0.927**

The extra "umbrella on day 2" makes "rain on day 1" more likely — future evidence helps.

Deriving the smoothing formula

Four small steps from $P(S_k|o_{0:(t-1)})$ to the forward-backward product:

$$
\begin{aligned}
& P(S_k|o_{0:(t-1)}) \\
&= P(S_k|o_{(k+1):(t-1)} \wedge o_{0:k}) && \textit{rewrite} \\
&= \alpha\, P(S_k|o_{0:k})\, P(o_{(k+1):(t-1)}|S_k \wedge o_{0:k}) && \textit{Bayes' rule} \\
&= \alpha\, P(S_k|o_{0:k})\, P(o_{(k+1):(t-1)}|S_k) && \textit{Markov} \\
&= \alpha\, f_{0:k}\, b_{(k+1):(t-1)} && \textit{definitions}
\end{aligned}
$$

Try to identify the rule before the label appears.

Deriving the backward recursion

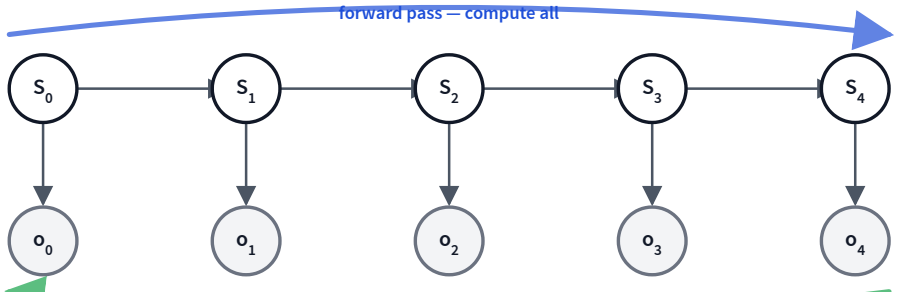
Five steps from $P(o_{(k+1):(t-1)} | S_k)$ to the recursive form:

$$
\begin{aligned}
& P(o_{(k+1):(t-1)} | S_k) \\
&= \sum_{s_{k+1}} P(o_{(k+1):(t-1)} \wedge s_{k+1} | S_k) && \textit{sum rule} \\
&= \sum_{s_{k+1}} P(o_{(k+1):(t-1)} | s_{k+1} \wedge S_k)\, P(s_{k+1} | S_k) && \textit{chain rule} \\
&= \sum_{s_{k+1}} P(o_{(k+1):(t-1)} | s_{k+1})\, P(s_{k+1} | S_k) && \textit{Markov} \\
&= \sum_{s_{k+1}} P(o_{k+1} \wedge o_{(k+2):(t-1)} | s_{k+1})\, P(s_{k+1} | S_k) && \textit{rewrite} \\
&= \sum_{s_{k+1}} P(o_{k+1} | s_{k+1})\, P(o_{(k+2):(t-1)} | s_{k+1})\, P(s_{k+1} | S_k) && \textit{Markov}
\end{aligned}
$$

The final factor is exactly $P(o_{k+1} | s_{k+1})\, b_{(k+2):(t-1)}\, P(s_{k+1} | S_k)$ — the backward recursion.

The forward-backward algorithm

Compute the smoothed posterior at *every* time step k with one forward sweep + one backward sweep.



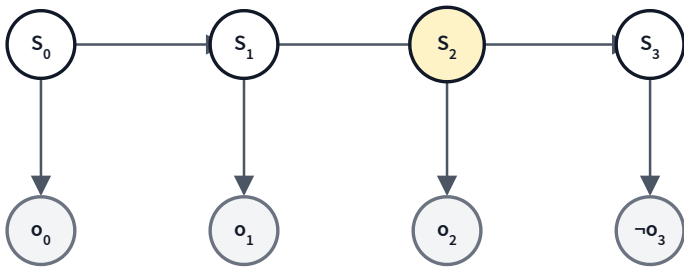
Combine the two sweeps at every k

$$P(S_k \mid o_{0:(t-1)}) = \alpha\, f_{0:k} \cdot b_{(k+1):(t-1)}$$

Forward gives every $f_{0:k}$ (recap of L10), backward gives every $b_{(k+1):(t-1)}$ — total cost $O(t)$, one sweep each direction.

Try it: smoothing on a 4-step HMM

Compute $P(S_2 \mid o_0 \wedge o_1 \wedge o_2 \wedge \neg o_3)$ – here $k = 2,\ t = 4$.



Numbers

$$P(s_0) = 0.4$$
$$P(s_t|s_{t-1}) = 0.7$$
$$P(s_t|\neg s_{t-1}) = 0.2$$
$$P(o_t|s_t) = 0.9$$
$$P(o_t|\neg s_t) = 0.2$$

Forward (from L10): $f_{0:2} \approx \langle 0.884, 0.116 \rangle$

Backward: $b_{3:3} = 0.1\langle 0.7, 0.2 \rangle + 0.8\langle 0.3, 0.8 \rangle = \langle 0.31, 0.66 \rangle$

Smoothed: $\alpha\langle 0.884, 0.116 \rangle \cdot \langle 0.31, 0.66 \rangle = \alpha\langle 0.274, 0.0766 \rangle = \boxed{\langle 0.782, 0.218 \rangle}$

The most-likely explanation

Find the single **best sequence of hidden states** that explains the observations:

$$\hat{s}_{0:t-1} = \arg\max_{s_{0:t-1}} P\left(s_{0:t-1} \middle| o_{0:t-1}\right)$$

- Generalize: $S_t \in \{0, 1, \ldots, n-1\}$ – not just binary.
- Transition matrix $A \in \mathbb{R}^{n \times n}$, emission matrix $O \in \mathbb{R}^{n \times m}$.

Brute force: enumerate every state sequence.

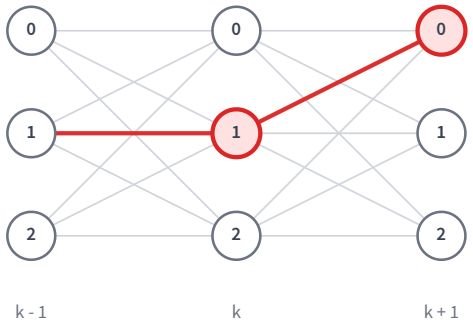
n^t sequences — intractable.

Dynamic programming to the rescue: the **Viterbi algorithm**.

Dynamic programming: define $\pi_k(j)$

$$\pi_k(j) \;=\; \max_{s_{0:(k-1)}} P\left(s_{0:(k-1)},\, S_k = j \mid o_{0:k}\right)$$

The highest probability of any state sequence $s_{0:k}$ ending with $S_k = j$, given observations $o_{0:k}$.



Trellis: one column per time step, one node per state value. Red = best path entering each node.

Viterbi recursion

Base case

$$\pi_0(j) = \alpha\, P(S_0 = j)\, P(o_0 | S_0 = j)$$

Recursive case

$$\pi_k(j) = P(o_k | S_k = j) \cdot \max_z \left[\pi_{k-1}(z)\, P(S_k = j | S_{k-1} = z) \right]$$

Backpointer

$$\phi_k(j) = \arg\max_z \left[\pi_{k-1}(z)\, P(S_k = j | S_{k-1} = z) \right]$$

$\pi_k(j)$ = best probability of any path ending at state j at time k. $\phi_k(j)$ remembers which predecessor produced it — we'll use it to trace the best sequence back.

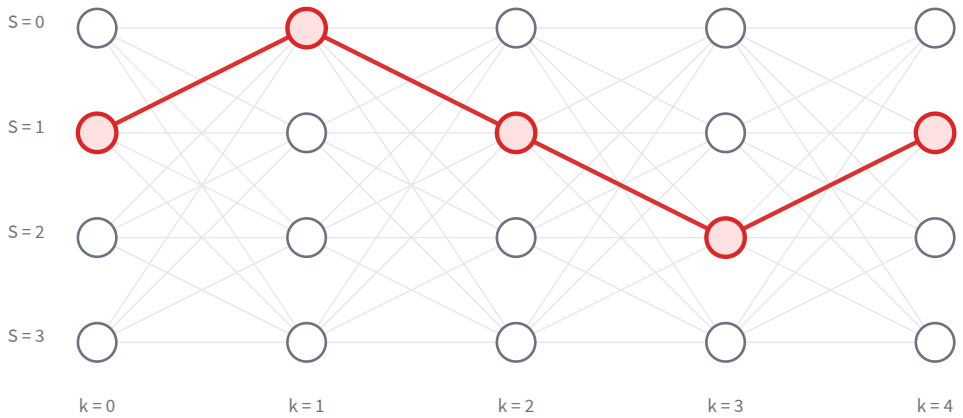
Viterbi: the algorithm

`viterbi(observations, transition, emission)`

1. Initialize $\pi_0(j) = \alpha\, P(S_0 = j)\, P(o_0 | S_0 = j)$ for each state j; $\phi_0(j) = \text{none}$.
2. **Forward sweep**: for $k = 1, \ldots, t-1$ and each j, compute $\pi_k(j)$ and record the argmax in $\phi_k(j)$.
3. **Best terminal**: $\hat{s}_{t-1} = \arg\max_j \pi_{t-1}(j)$.
4. **Traceback**: for $k = t-1, \ldots, 1$, set $\hat{s}_{k-1} = \phi_k(\hat{s}_k)$.
5. Return the sequence $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{t-1})$.

One sweep fills the trellis; one walk back along the backpointers extracts the MAP sequence.

Viterbi: trellis walk-through + complexity



Best path: $\hat{s} = (1, 0, 1, 2, 1)$.

Complexity: $O(t \cdot n^2)$ — one trellis edge per cell per step. Linear in time, quadratic in #states.

Learning goals (recap)

✓ Compute the **smoothing** probability for a time step in an HMM.

✓ **Justify** each step in the smoothing & backward derivations.

✓ Describe the **forward-backward** algorithm.

✓ Describe the **Viterbi** algorithm.

Next module: Decision Making

So far we have *beliefs*. Next we add *preferences* — how should an agent **act** when outcomes are uncertain?

- **L12** — decision theory: utility, MEU, decision networks, value of information.
- **L13–L14** — Markov decision processes: Bellman equation, value & policy iteration.
- **L15** — reinforcement learning: ADP, Q-learning, SARSA.