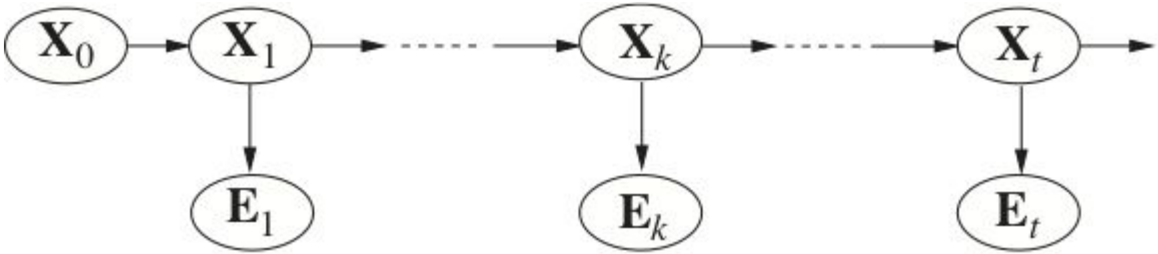
Smoothing

CS 580
Intro to Artificial Intelligence

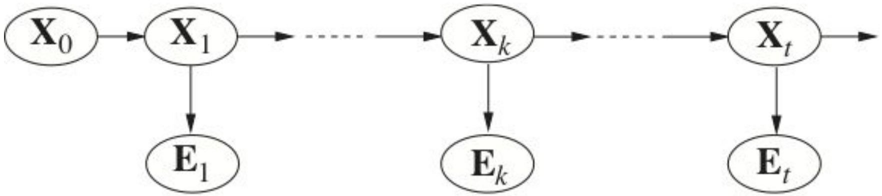
Smoothing

What if we want to know the probability of the state variable at a given point in time in the past?



If we want to know about X_k and we have evidence from $E_{1:t}$, we should incorporate that (rather than just using $E_{1:k}$)

Forward-Backward (1)



The new probability we care about is $p(X_k|e_{1:t})$ which we can split into two pieces

$$p(X_k \mid e_{1:t}) = p(X_k \mid e_{1:k}, e_{k+1:t})$$

Bayes rule

$$= \alpha \cdot p(e_{k+1:t} \mid X_k, e_{1:k}) \cdot p(X_k \mid e_{1:k})$$

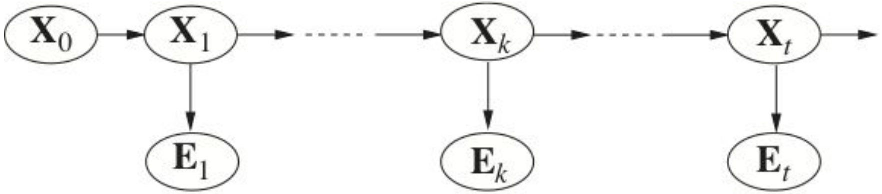
Conditional Independence

$$= \alpha \cdot p(e_{k+1:t} \mid X_k) \cdot p(X_k \mid e_{1:k})$$

Let’s take a closer look at this term

We already know how to compute this!

Forward-Backward (2)



$$p(e_{k+1:t} \mid X_k) = \sum_h p(e_{k+1:t}, X_{k+1} = h \mid X_k)$$

Marginalize out X_{k+1}

Definition of conditional prob

$$= \sum_h p(e_{k+1:t} \mid X_k, X_{k+1} = h) \cdot p(X_{k+1} = h \mid X_k)$$

Cond. Indep.

$$= \sum_h p(e_{k+1:t} \mid X_{k+1} = h) \cdot p(X_{k+1} = h \mid X_k)$$

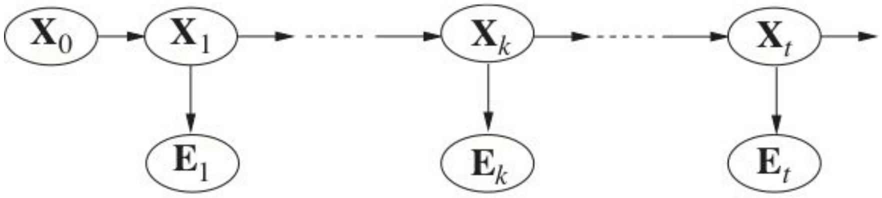
Split $e_{k+1:t}$ into e_{k+1} and $e_{k+2:t}$

$$= \sum_h p(e_{k+1}, e_{k+2:t} \mid X_{k+1} = h) \cdot p(X_{k+1} = h \mid X_k)$$

Cond. Indep.

$$= \sum_h \underbrace{p(e_{k+1} \mid X_{k+1} = h)}_{\text{Sensor Model}} \cdot \underbrace{p(e_{k+2:t} \mid X_{k+1} = h)}_{\text{Recurrence!}} \cdot \underbrace{p(X_{k+1} = h \mid X_k)}_{\text{Transition Model}}$$

Forward-Backward (3)



So our equation for smoothing is

$$\begin{aligned} p(X_k \mid e_{1:t}) &= \alpha \cdot p(X_k \mid e_{1:k}) \cdot p(e_{k+1:t} \mid X_k) \\ &= \alpha \cdot \mathbf{f}_{1:k} \odot \mathbf{b}_{k+1:t} \end{aligned}$$

Where **f** is the “forward” variable

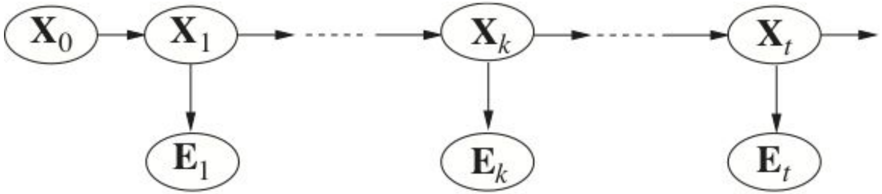
$$\begin{aligned} \mathbf{f}_{1:t} &= p(X_t \mid e_{1:t}) \\ &= \alpha \cdot p(e_t \mid X_t) \sum_h p(X_t \mid X_{t-1} = h) \cdot p(X_{t-1} = h \mid e_{1:t-1}) \end{aligned}$$

And **b** is the “backward” variable

$$\begin{aligned} \mathbf{b}_{k+1:t} &= p(e_{k+1:t} \mid X_k) \\ &= \sum_h p(e_{k+1} \mid X_{k+1} = h) \cdot p(e_{k+2:t} \mid X_{k+1} = h) \cdot p(X_{k+1} = h \mid X_k) \end{aligned}$$

Base case: $\mathbf{b}_{t+1:t} = p(e_{t+1:t} | X_t) = \mathbf{1}$

Forward-Backward (4)



So we have an equation for how to smooth a sequence of evidence for a **single** timestep, how do we do this for **all** the timesteps?

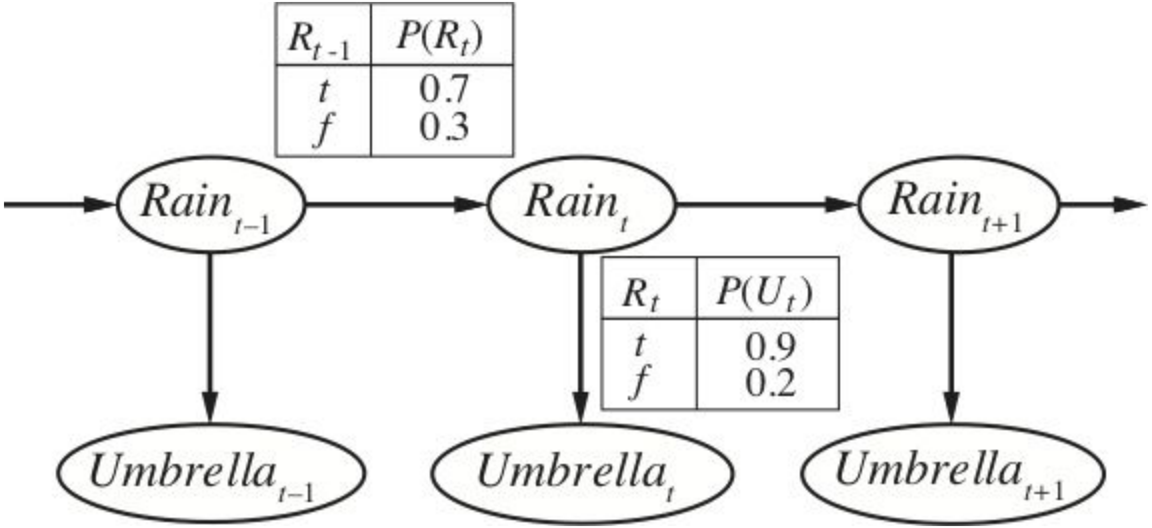
$$p(X_k \mid e_{1:t}) = \alpha \cdot p(X_k \mid e_{1:k}) \cdot p(e_{k+1:t} \mid X_k)$$
$$= \alpha \cdot \mathbf{f}_{1:k} \odot \mathbf{b}_{k+1:t}$$

```
def forward_backward(sensor_m, transition_m, prior, evidence):
      fv[0] = prior
      b = numpy.ones(len(prior))
      for i in range(1,t+1):
            fv[i] = forward(fv[i-1],evidence[i],sensor_m,transition_m)
      for i in range(t,0,-1):
            smoothed[i] = normalize(fv[i]*b)
            b = backward(b,evidence[i],sensor_m,transition_m)
      return smoothed
```

Key Idea: save the forward pass computations for use during the backward pass

Weather example (1)

Example problem: a security guard would like to know about the weather. They can see people entering/leaving with umbrellas, but can’t see directly whether it’s raining or not.



Weather example (2)

$$\mathbf{f}_{1:t} = \alpha \cdot p(e_t \mid X_t) \sum_h p(X_t \mid X_{t-1} = h) \cdot p(X_{t-1} = h \mid e_{1:t-1})$$

$$\mathbf{b}_{k+1:t} = \sum_h p(e_{k+1} \mid X_{k+1} = h) \cdot p(e_{k+2:t} \mid X_{k+1} = h) \cdot p(X_{k+1} = h \mid X_k)$$

Observations: (U_1=True, U_2=True)

Forward pass

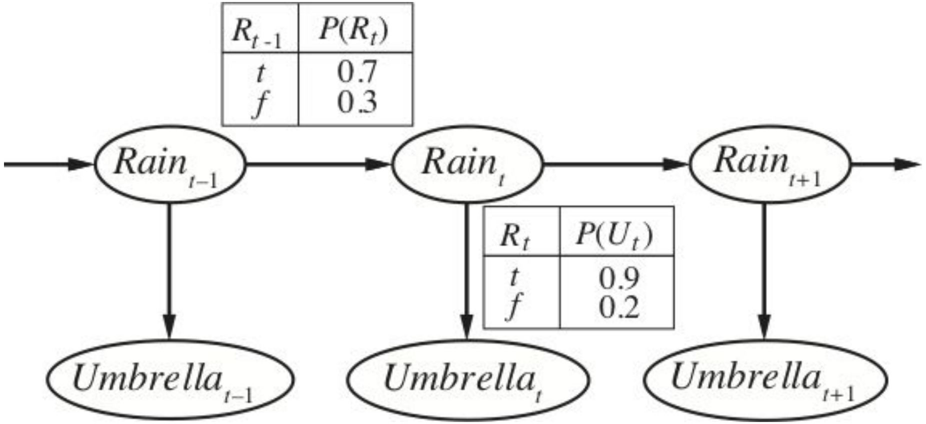
$\mathbf{f}_{1:0}$ = p(R_0) = <0.5, 0.5>

$\mathbf{f}_{1:1}$ = α <0.9,0.2>*(<0.7,0.3>*0.5 +<0.3,0.7>*0.5)

= α <0.45,0.1> = <0.818, 0.182>

$\mathbf{f}_{1:2}$ = α <0.9,0.2>*(<0.7,0.3>*.818+<0.3,0.7>*.182)

= α <0.565, 0.075> = <0.883, 0.117>



Backward pass

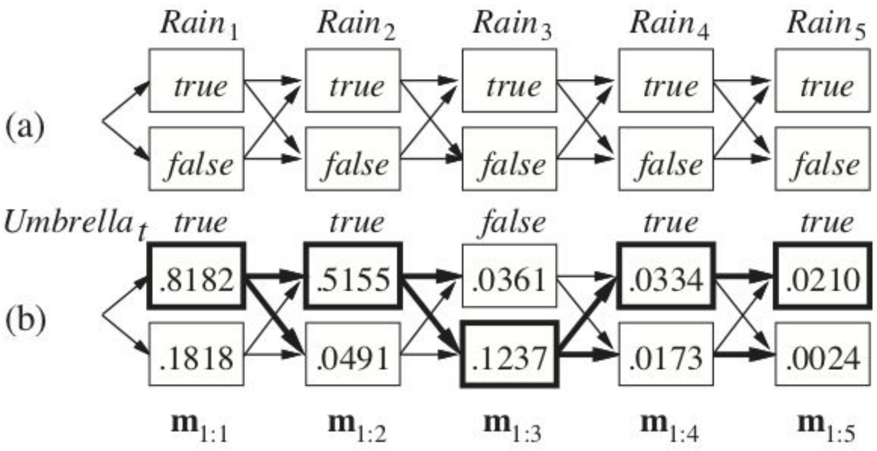
$\mathbf{b}_{3:2}$=1

$\mathbf{b}_{2:2}$=(.9*1*<.7,.3> + .2*1*<.3,.7>) = <0.69,0.41>, **smoothed** = <0.927,0.073>

$\mathbf{b}_{1:2}$=(.9*.69*<.7,.3> + .2*.41*<.3,.7>) = <.459,.243>, **smoothed** = <0.894,0.106>

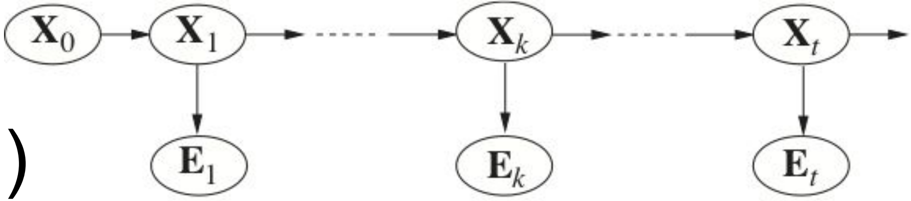
The most likely sequence (1)

What is the most likely sequence of states?



Note: not the same as the most likely state at each step!

The most likely sequence (2)



Rephrase: what is the **probability** of the last state X_t in the most likely sequence?

Bayes' Rule on e_t and $x_{1:t-1}$, X_t

$$\max_{x_{1:t-1}} p(x_{1:t-1}, X_t \mid e_{1:t}) = \max_{x_{1:t-1}} \alpha \cdot p(e_t \mid x_{1:t-1}, X_t, e_{1:t-1}) \cdot p(x_{1:t-1}, X_t \mid e_{1:t-1})$$

Sensor Markov

$$= \max_{x_{1:t-1}} \alpha \cdot p(e_t \mid X_t) \cdot p(x_{1:t-1}, X_t \mid e_{1:t-1})$$

Product rule on $x_{1:t-1}$, X_t

$$= \max_{x_{1:t-1}} \alpha \cdot p(e_t \mid X_t) \cdot p(X_t \mid x_{1:t-1}, e_{1:t-1}) \cdot p(x_{1:t-1} \mid e_{1:t-1})$$

Transition Markov

$$= \max_{x_{1:t-1}} \alpha \cdot p(e_t \mid X_t) \cdot p(X_t \mid x_{t-1}) \cdot p(x_{1:t-1} \mid e_{1:t-1})$$

$$= \alpha \cdot \underbrace{p(e_t \mid X_t)}_{\text{Sensor Model}} \cdot \max_{x_{t-1}} \underbrace{p(X_t \mid x_{t-1})}_{\text{Transition Model}} \cdot \underbrace{\max_{x_{1:t-2}} p(x_{1:t-2}, X_{t-1} = x_{t-1} \mid e_{1:t-1})}_{\text{Recurrence!}}$$

The most likely sequence (3)

Define the **max** variable

$$\mathbf{m}_{1:t} = \max_{x_{1:t-1}} p(x_{1:t-1}, X_t \mid e_{1:t})$$

$$= \alpha \cdot p(e_t \mid X_t) \cdot \max_{x_{t-1}} p(X_t \mid x_{t-1}) \odot \mathbf{m}_{1:t-1}$$

Compare with the **forward** variable

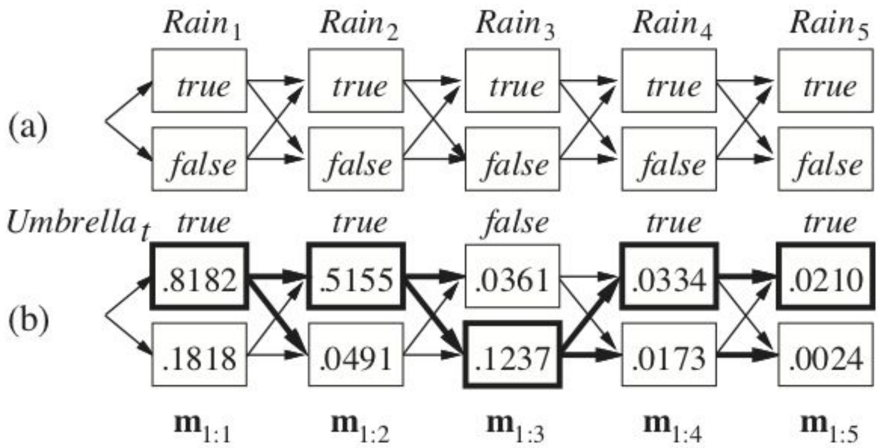
$$\mathbf{f}_{1:t} = p(X_t \mid e_{1:t})$$

$$= \alpha \cdot p(e_t \mid X_t) \sum_h p(X_t \mid X_{t-1} = h) \odot \mathbf{f}_{1:t-1}$$

Swapped **sum** for **max**

The Viterbi algorithm

1. Init with $\mathbf{m}_{1:0} = p(X_0)$ (prior)
2. For each i in 1:t
   a. Compute $\mathbf{m}_{1:i}$
   b. Store the best state that leads to X_i (bold arrows)
3. max($\mathbf{m}_{1:t}$) is the probability of the most likely sequence
4. The actual sequence can be recovered by following backpointers from the most likely final state



Filtering, smoothing, Viterbi

Exact Filtering

$\mathbf{f}_{1:T}$: space O(|S|), time O(|S|*T), Online

Smoothing (forward-backward)

$\mathbf{f}_{1:T}$: space O(|S|*T), time O(|S|*T)
$\mathbf{b}_{1:T}$: space O(|S|), time O(|S|*T)
Offline (fixed-lag smoothing online version)

Most Likely Sequence (Viterbi)

$\mathbf{m}_{1:T}$: space O(|S|*T), time O(|S|*T), Offline

Summary and preview

Wrapping up

- Two more inference algorithms: Smoothing, and Viterbi
- All of these inference algorithms can be modified to work with Bayes nets with different structures
- Additionally, for some Bayes nets, we can actually **learn** the parameters given sequences of observations (Expectation-Maximization)

Next time

- Search, as an interlude to ML