Temporal probability models

Chapter 15

Outline

- ♢ Time and uncertainty
- ♢ Inference: filtering, prediction, smoothing
- ♢ Hidden Markov models
- ♢ Kalman filters (a brief mention)
- ♢ Dynamic Bayesian networks
- ♢ Particle filtering

Time and uncertainty

The world changes; we need to track and predict it

Diabetes management vs vehicle diagnosis

Basic idea: copy state and evidence variables for each time step

$\mathbf{X}_t$ = set of unobservable state variables at time t
e.g., $BloodSugar_t$, $StomachContents_t$, etc.

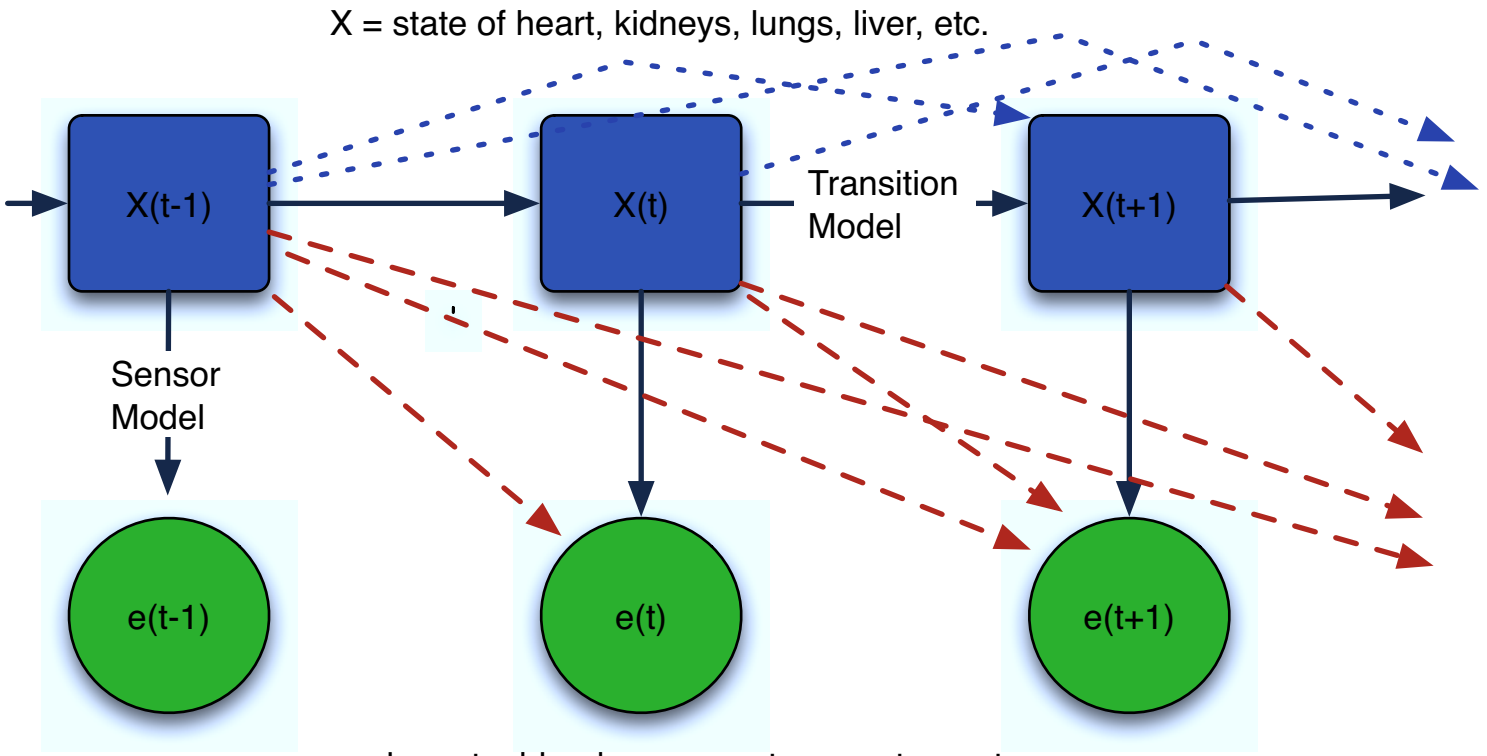
$\mathbf{E}_t$ = set of observable evidence variables at time t
e.g., $MeasuredBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$

This assumes **discrete time**; step size depends on problem

Notation: $\mathbf{X}_{a:b} = \mathbf{X}_a, \mathbf{X}_{a+1}, \ldots, \mathbf{X}_{b-1}, \mathbf{X}_b$

Reducing Causal Dependencies

- Only the essential causal relationships should be in our models. Else, too complex.
- Keys to reducing number of variable dependencies:
  - Sensor Model - evidence is totally dependent upon current state. Removes all red links.
  - Markov Assump - Current state = f(**finite** history of earlier states). Reduces blue links.
  - Stationary Process Assump - Same causal mechs at each step.

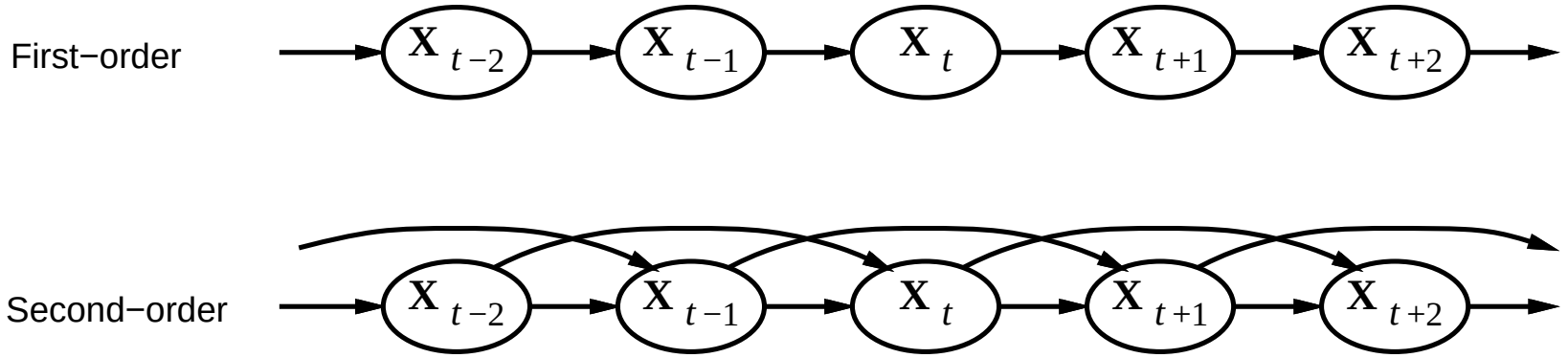


Markov processes (Markov chains)

Construct a Bayes net from these variables: parents?

Markov assumption: $\mathbf{X}_t$ depends on **bounded** subset of $\mathbf{X}_{0:t-1}$

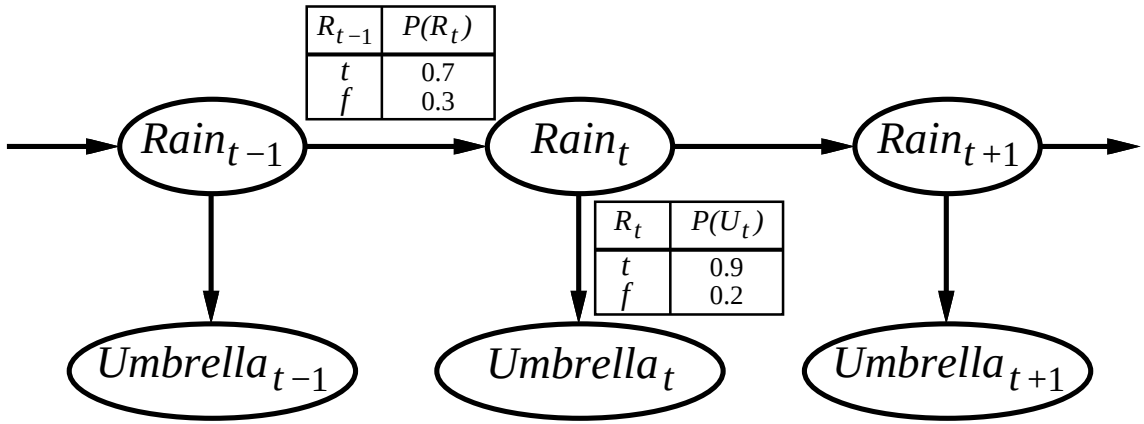
First-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
Second-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1})$



Sensor Markov assumption: $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

Stationary process: transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ and
sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ fixed for all t

Example



First-order Markov assumption not exactly true in real world!

Possible fixes:

1. **Increase order** of Markov process
2. **Augment state**, e.g., add $Temp_t$, $Pressure_t$

Example: robot motion.

Augment position and velocity with $Battery_t$

Inference tasks

Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$

belief state—input to the decision process of a rational agent

Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$

evaluation of possible action sequences;
like filtering without the evidence

Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$

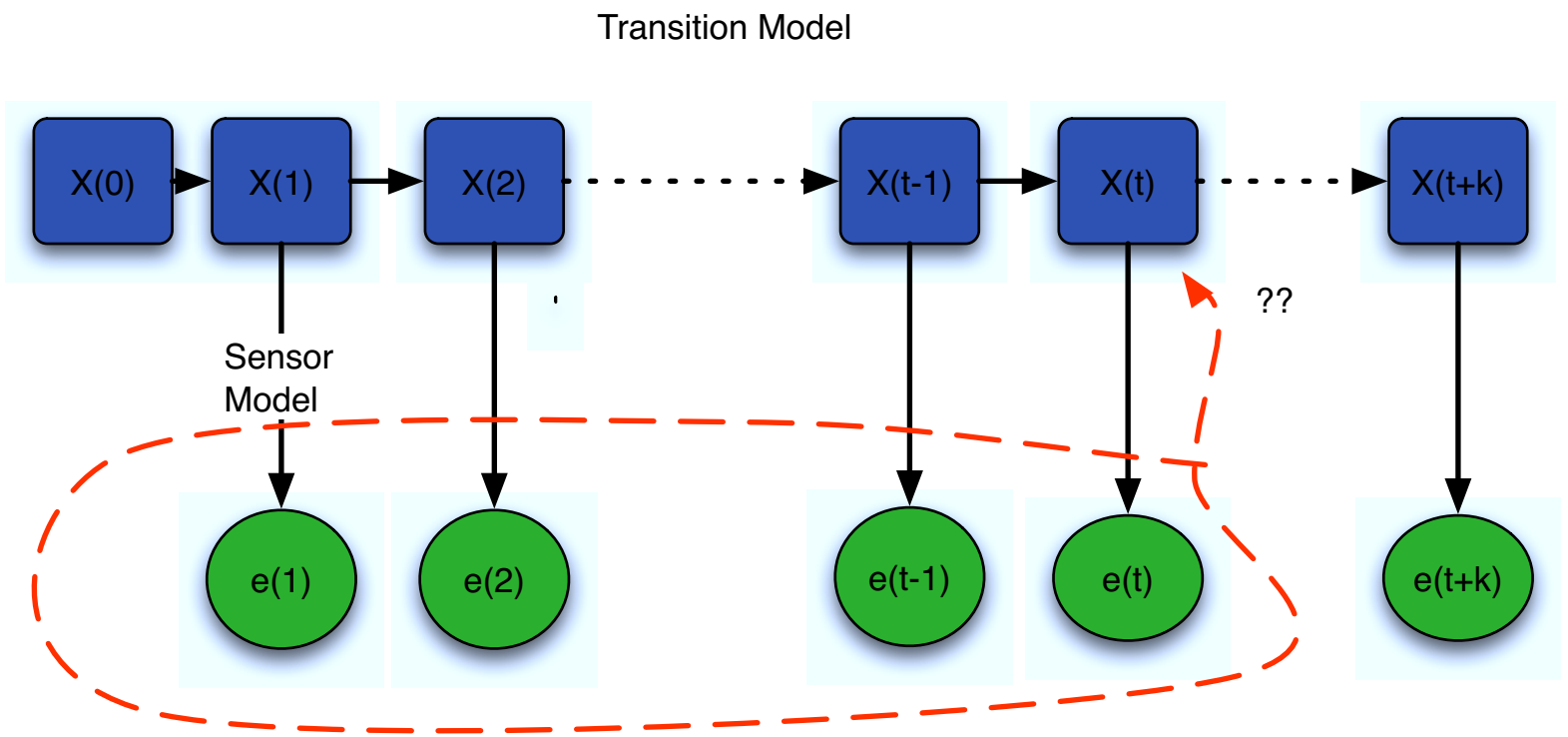
better estimate of past states, essential for learning

Most likely explanation: $\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$

speech recognition, decoding with a noisy channel

- All are **Abductive** tasks: Given evidence, calc probs of the causes (states).
- All use Bayes Rule: Calc p(X) from E (evidence) using $p(x \rightarrow e)$ (sensor model).
- All use the transition model: $p(x_t \rightarrow x_{t+1})$.

Filtering Overview



Given: Evidence from time 0 to present. (E.g., tiredness, fever, runny nose)
Compute: Probability of a given CURRENT state (E.g. Influenza?)

Filtering

Aim: devise a **recursive** state estimation algorithm:

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t}))$$

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) \\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})\end{aligned}$$

I.e., prediction + estimation. Prediction by summing out $\mathbf{X}_t$:

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\Sigma_{\mathbf{x}_t}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t, \mathbf{e}_{1:t})P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\Sigma_{\mathbf{x}_t}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{e}_{1:t})\end{aligned}$$

$\mathbf{f}_{1:t+1} = \textsc{Forward}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$ where $\mathbf{f}_{1:t} = \mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$
Time and space **constant** (independent of t)

Generalized Bayes' Rule

$$P(Y \mid X, e) = \frac{P(X \mid Y, e)P(Y \mid e)}{P(X \mid e)}$$

Proof:

$$\begin{aligned} P(Y \mid X, e) &= \frac{P(X, Y, e)}{P(X, e)} \\ &= \frac{P(X \mid Y, e)P(Y, e)}{P(X, e)} \\ &= \frac{P(X \mid Y, e)P(Y \mid e)P(e)}{P(X \mid e)P(e)} \\ &= \frac{P(X \mid Y, e)P(Y \mid e)}{P(X \mid e)} \end{aligned}$$

Generalized Bayes' Rule in Filtering

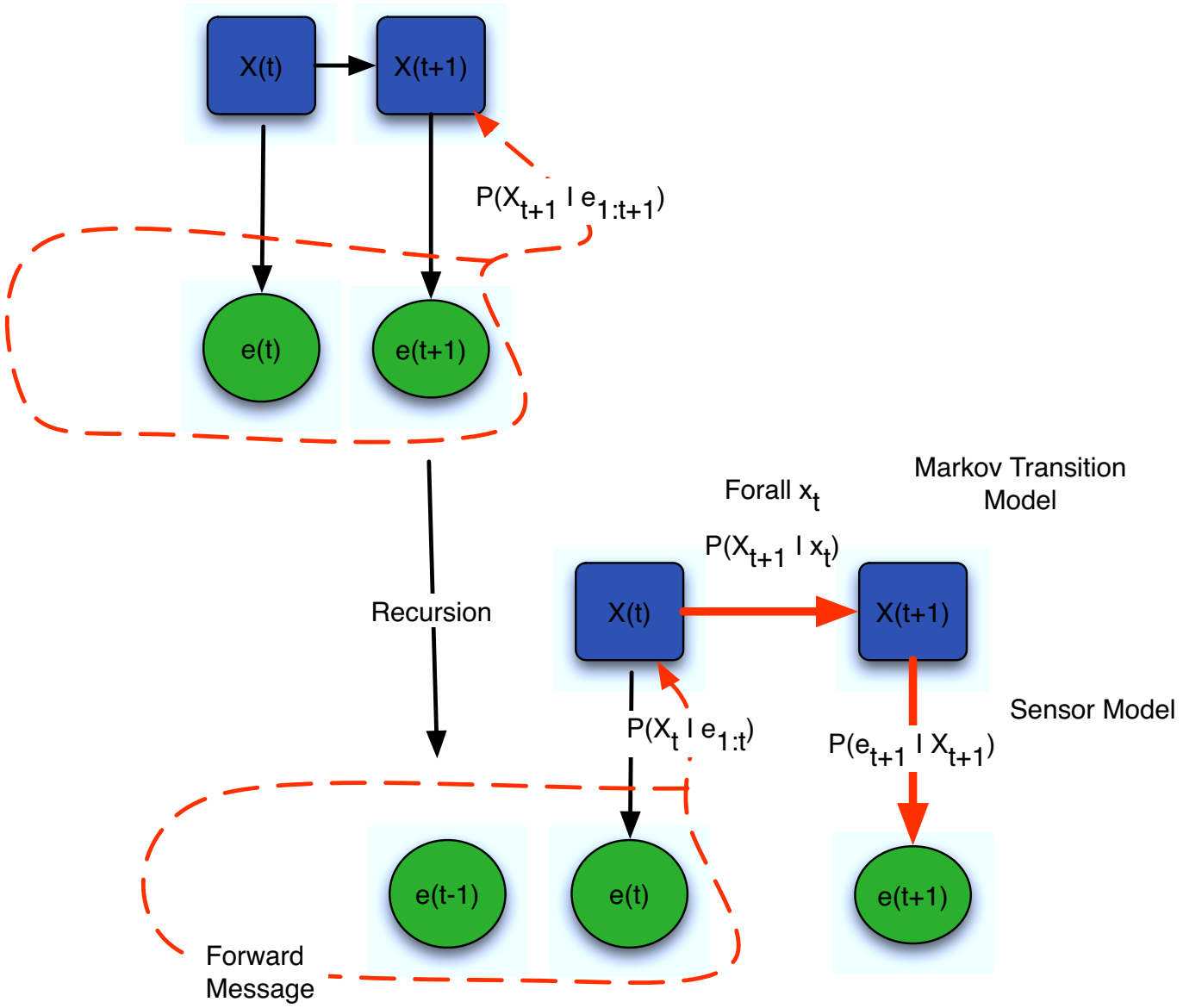
$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1},\mathbf{e}_t) &= \frac{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1},\mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})}{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{e}_{1:t})} \\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1},\mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})\end{aligned}$$

$\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{e}_{1:t})$ is the correct normalizing constant because:

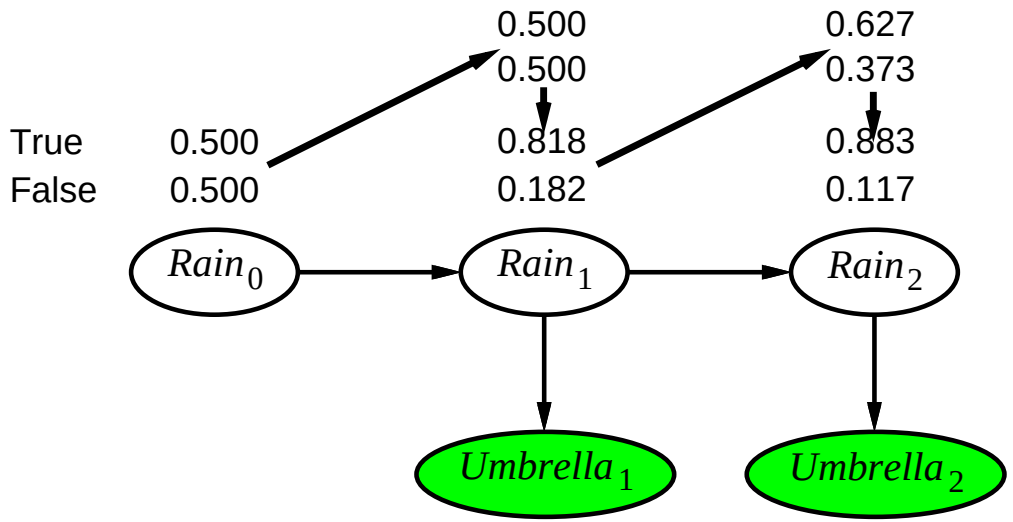
- In the course of generating the distribution $\forall\, x_{t+1} \in \mathbf{X}_{t+1}$,
- we go through all possible routes from $\mathbf{e}_{1:t}$ to $\mathbf{e}_{t+1}$, via the use of all possible x_{t+1} in the numerator of the General Bayes' Equation.
- So the sum of these numerators for all possible x_{t+1} is $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{e}_{1:t})$.

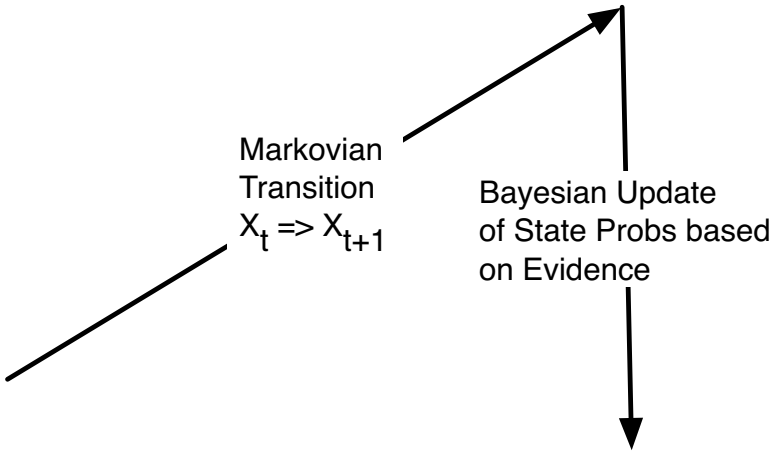
Big win: Now we are using the causal (sensor) model for $\mathbf{e}_{t+1}$ (via $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1},\mathbf{e}_{1:t})$ which reduces to $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$)
Dealing with $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ is easy, since it is reduces to Markov transitions and a recursive call.

Recursive Nature of Filtering

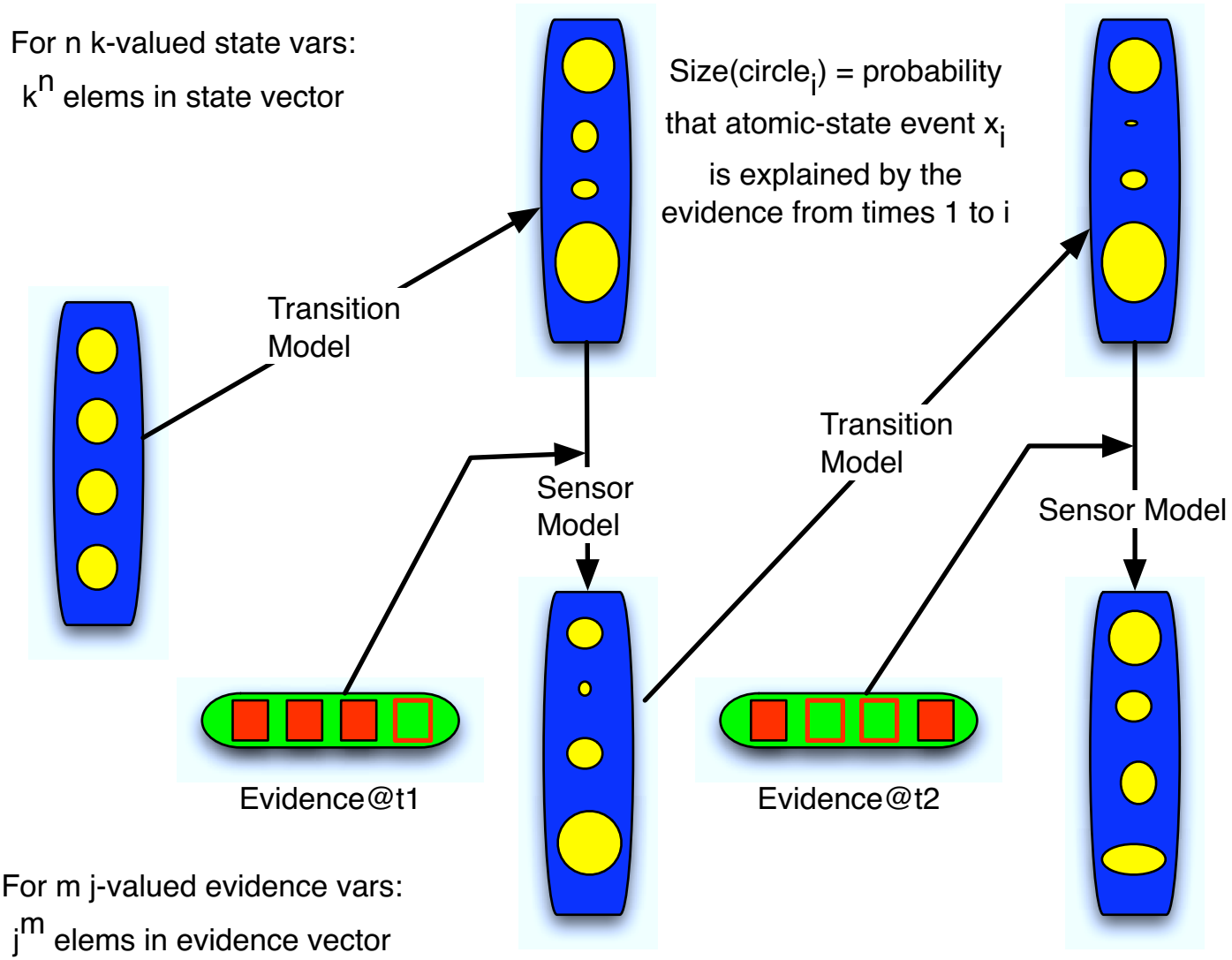


Filtering example

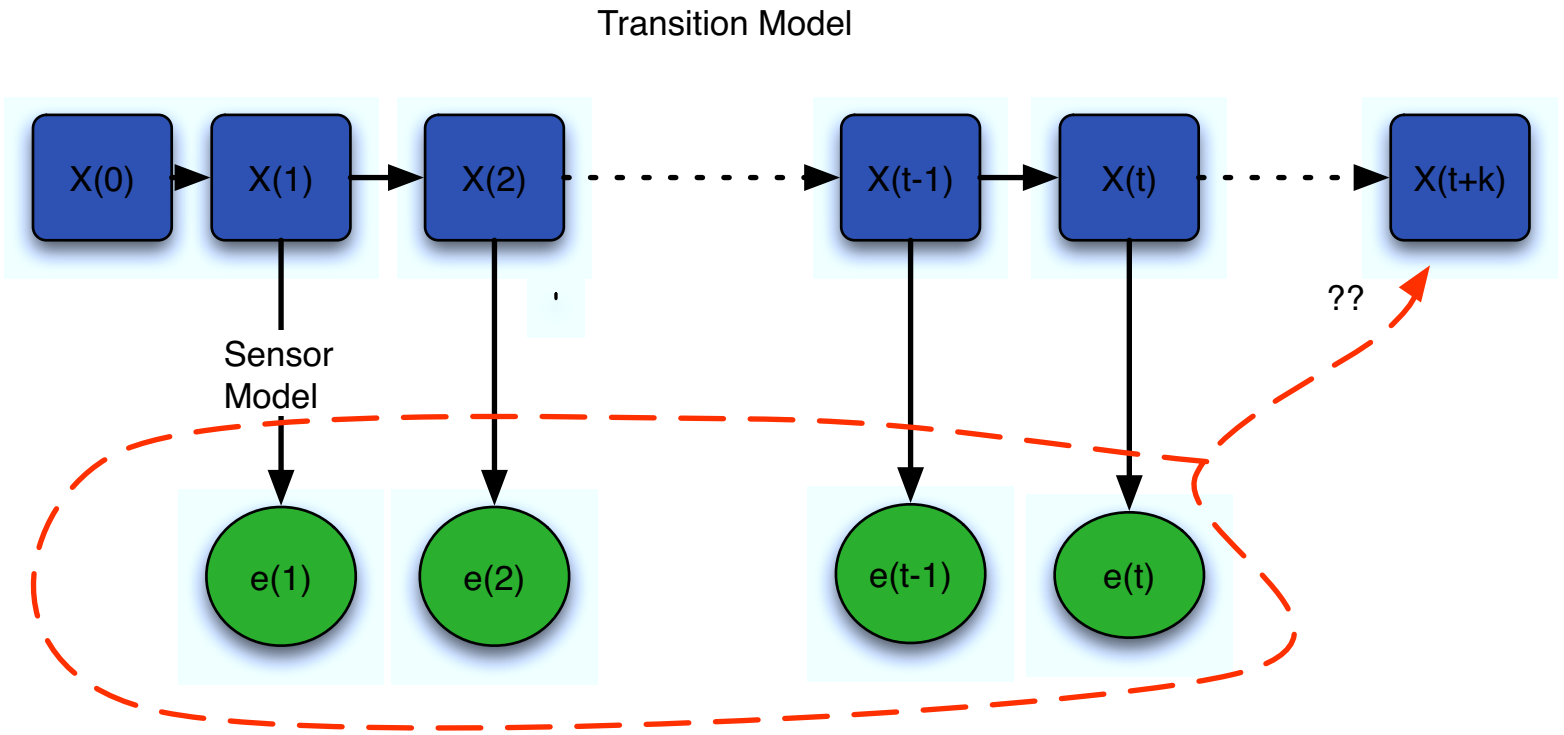




Forward Messaging

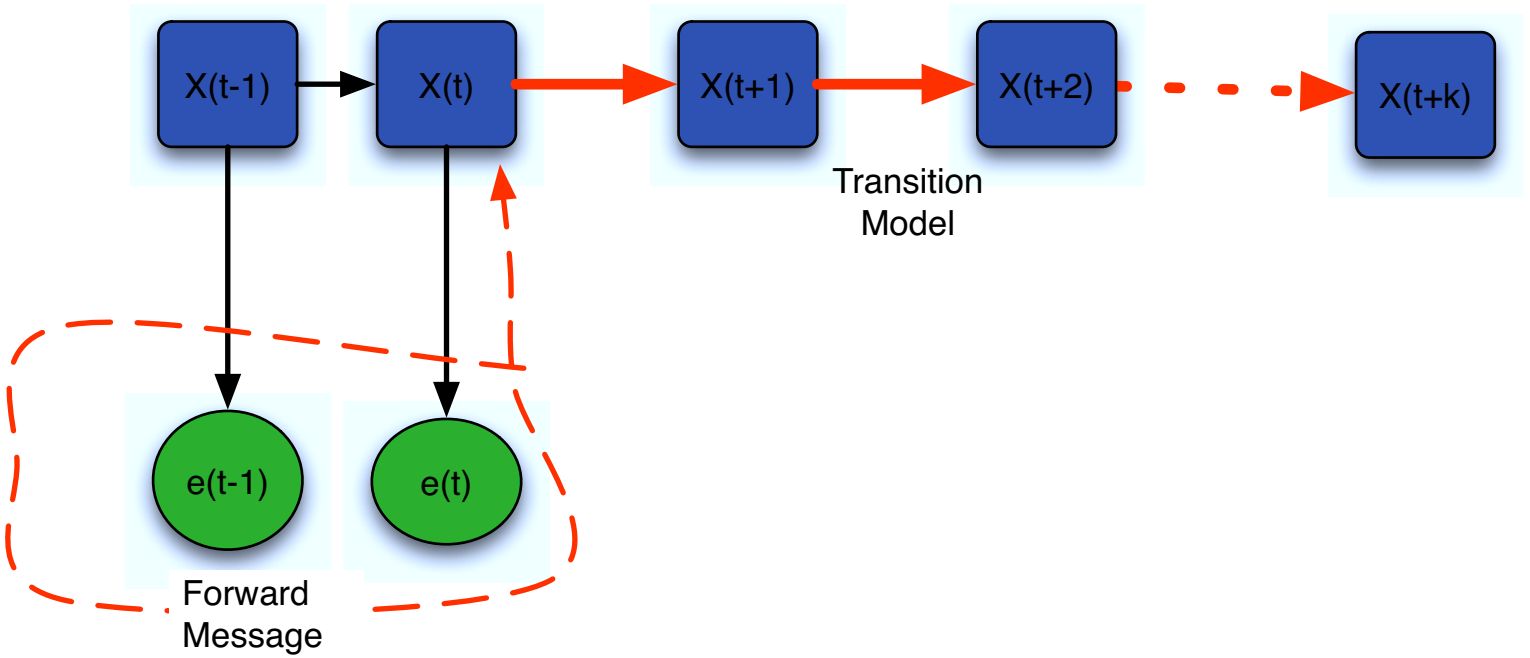


Prediction Overview



Given: Evidence (E.g., high sugar intake, sedentary lifestyle)
Compute: Probability of a FUTURE state (e.g. diabetes)

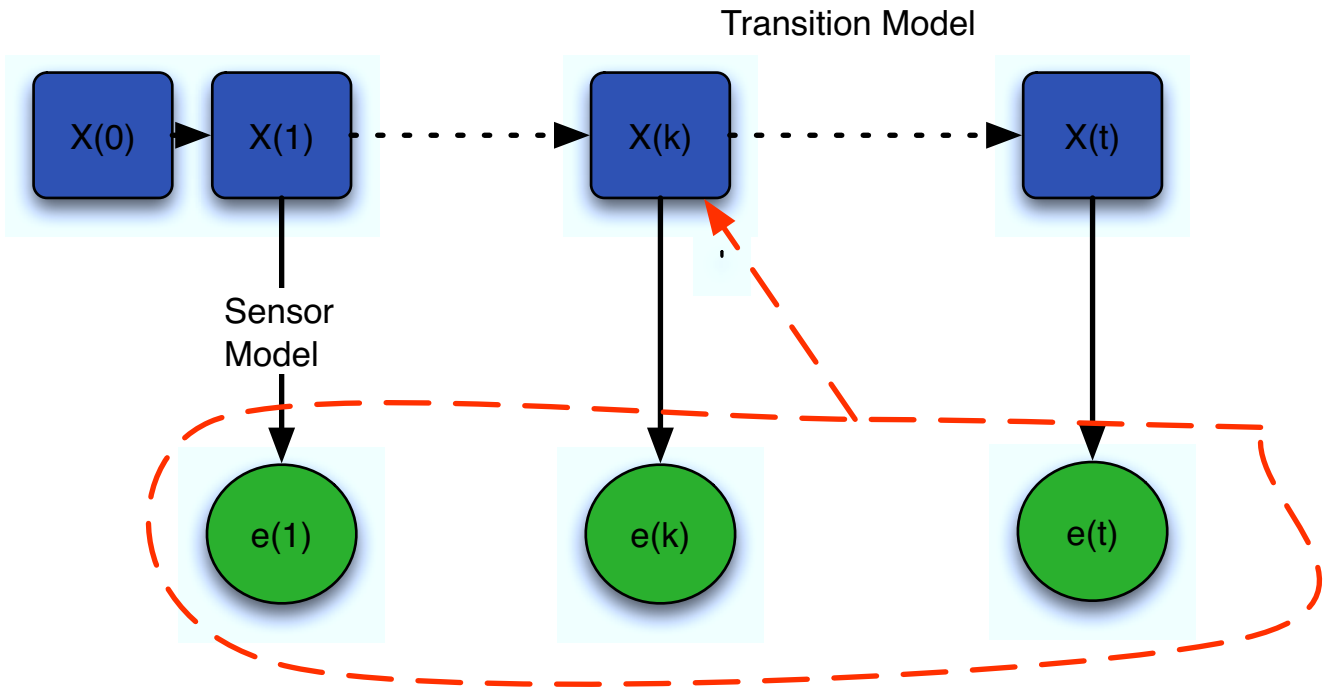
Prediction Recursion



$$\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \Sigma_{\mathbf{x}_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k}) \mathbf{P}(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t})$$

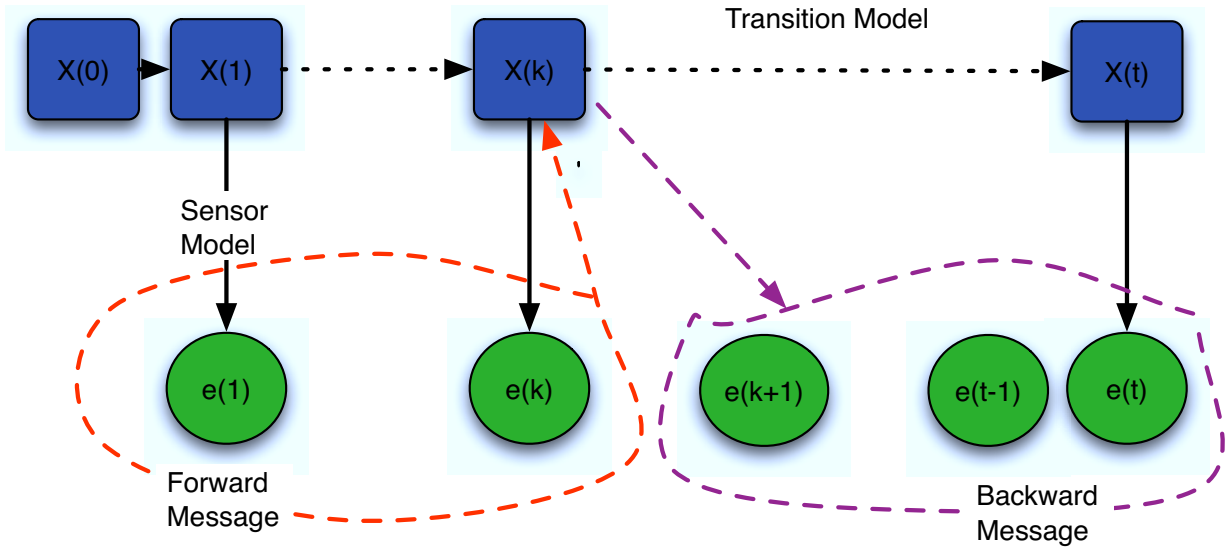
- 2nd term of the summation = Recursion
- Base case = forward message: $\mathbf{P}(\mathbf{x}_t \mid \mathbf{e}_{1:t})$

Smoothing Overview



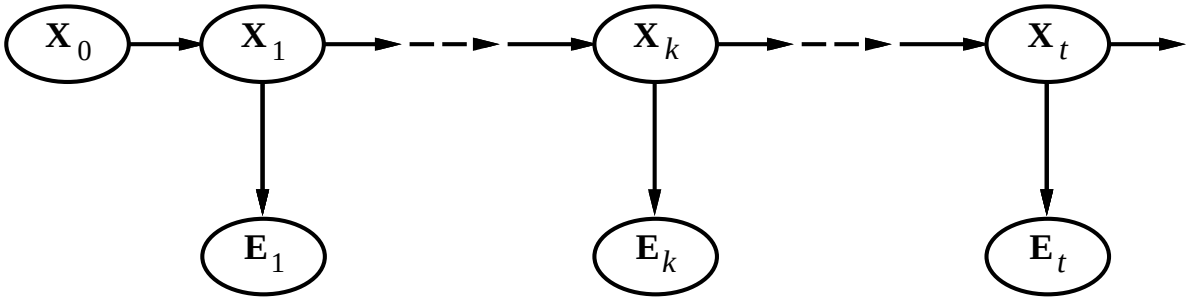
Given: Evidence (E.g., memory-loss today, loss of balance yesterday)
Compute: Probability of an EARLIER state (e.g. stroke 2 days ago)

Smoothing and Bi-Directional Recursion



- *Forward* actually means that we recurse backwards in time, but then the base-case probability is propagated forward from time 0 and modified by the transition and sensor models. Here, the base case is $\mathbf{P}(\mathbf{X}_0 \mid \mathbf{e}_0) = \mathbf{P}(X_0)$, since there is no evidence (by convention) at time 0.
- *Backward* involves forward recursion in time, with the eventual base-case probability propagating backwards from time = t. Here, the base case is $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_t) = 1$, since the evidence at time t+1 is the empty set.

Smoothing



Divide evidence $\mathbf{e}_{1:t}$ into $\mathbf{e}_{1:k}$, $\mathbf{e}_{k+1:t}$:

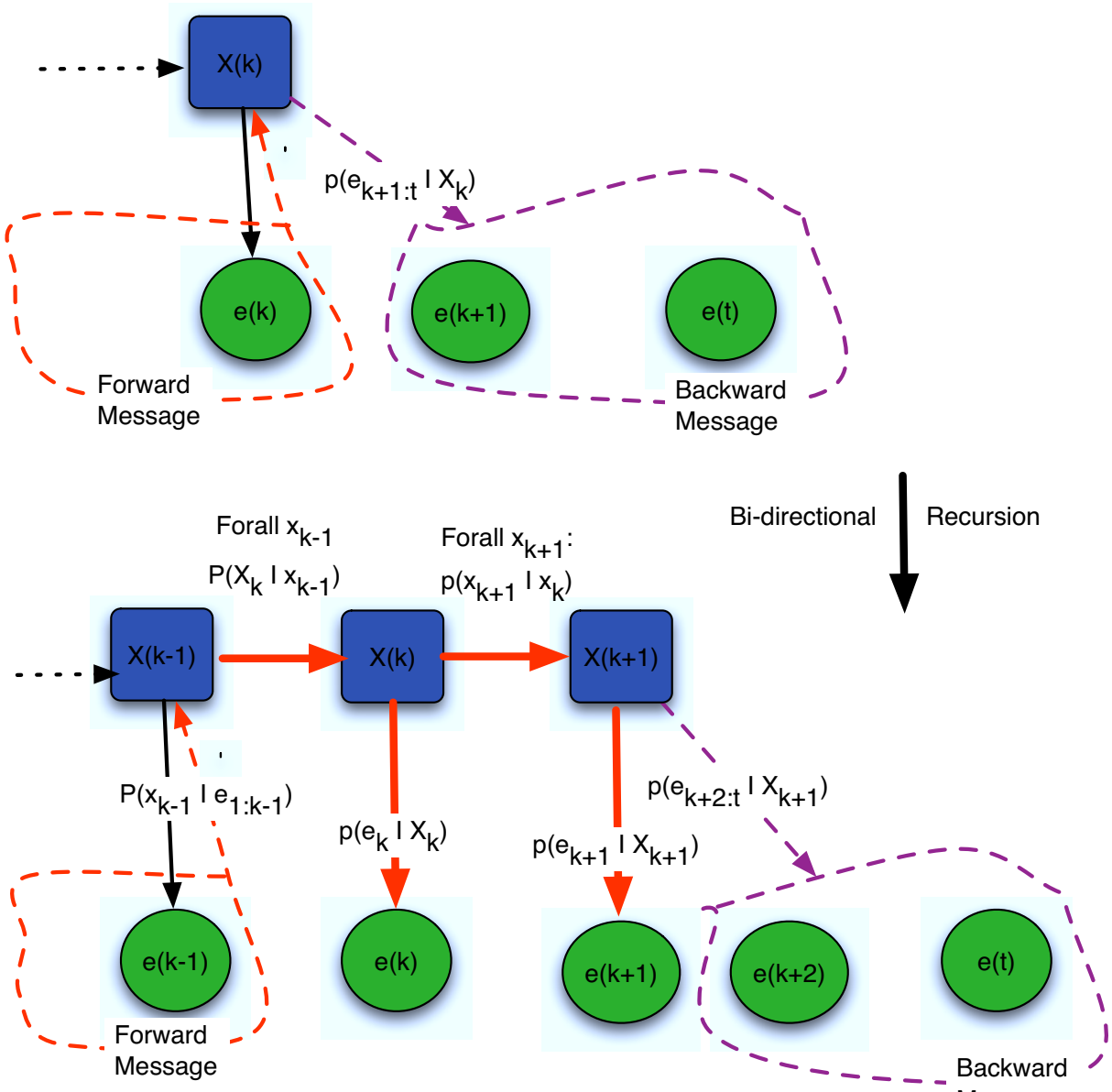
$$\begin{aligned}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k},\mathbf{e}_{k+1:t}) \\ &= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{e}_{1:k}) \\ &= \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \\ &= \alpha\mathbf{f}_{1:k}\mathbf{b}_{k+1:t}\end{aligned}$$

Backward message computed by a backwards recursion:

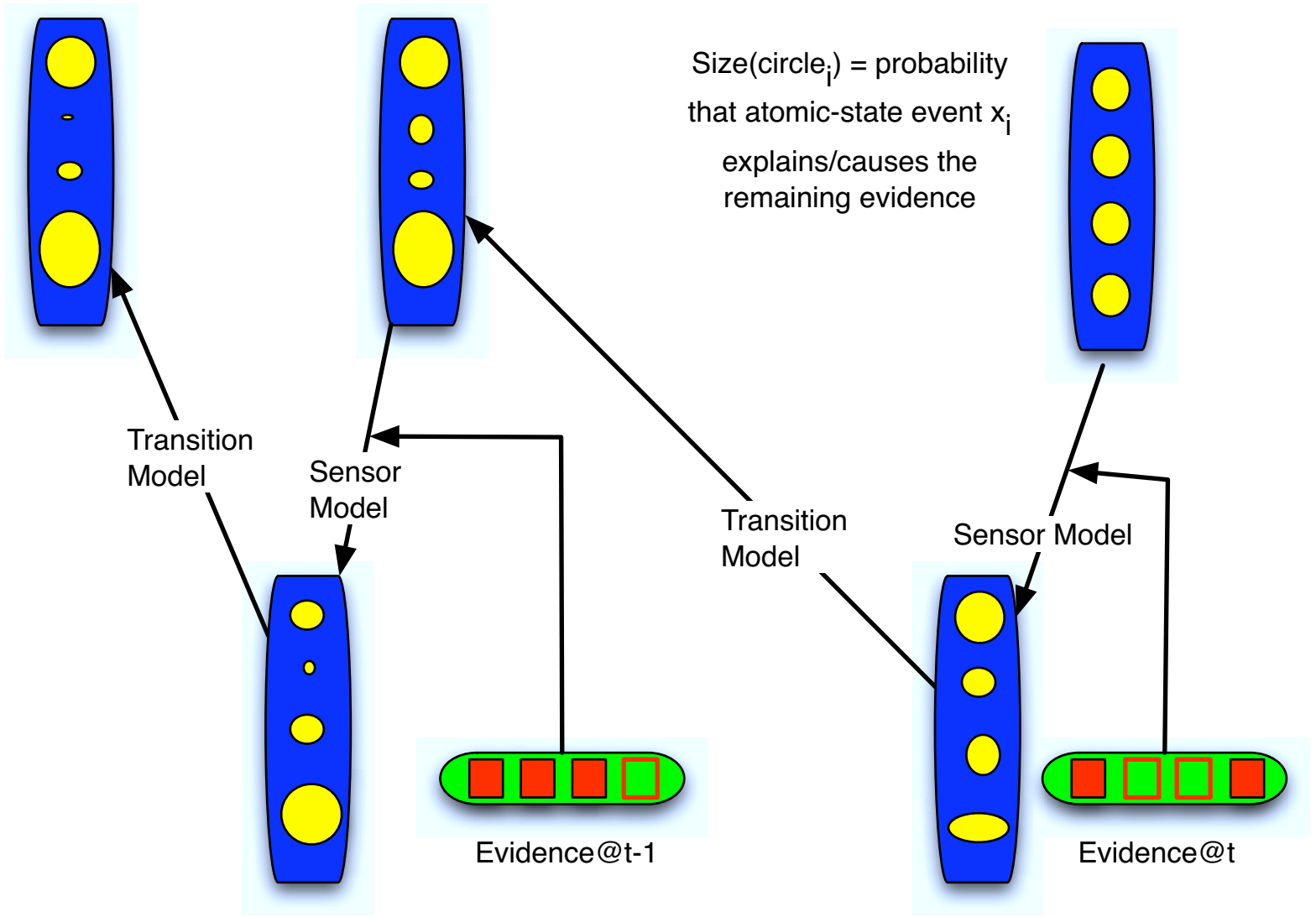
$$\begin{aligned}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \textstyle\sum_{\mathbf{x}_{k+1}}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\ &= \textstyle\sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\ &= \textstyle\sum_{\mathbf{x}_{k+1}}P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)\end{aligned}$$

3 parts of the final sum: Sensor Model, Recursive Step, Transition Model

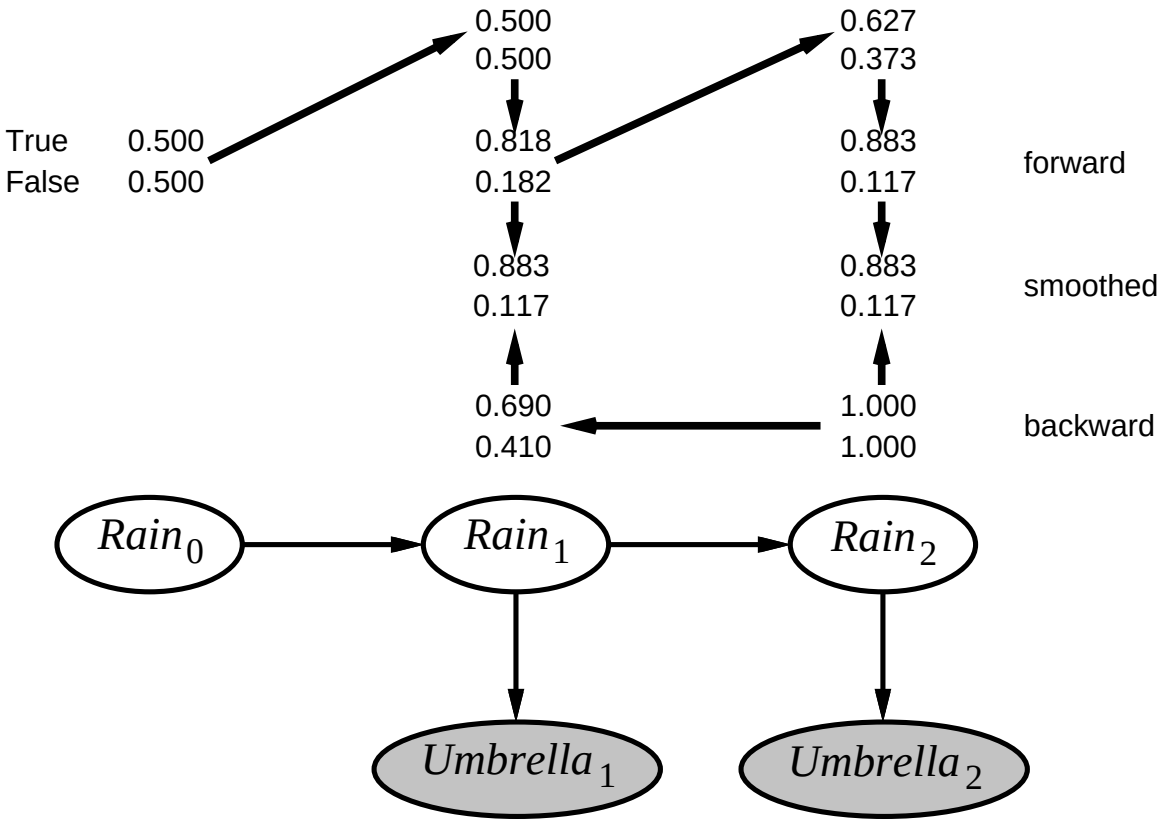
Smoothing Recursion



Backward Messaging

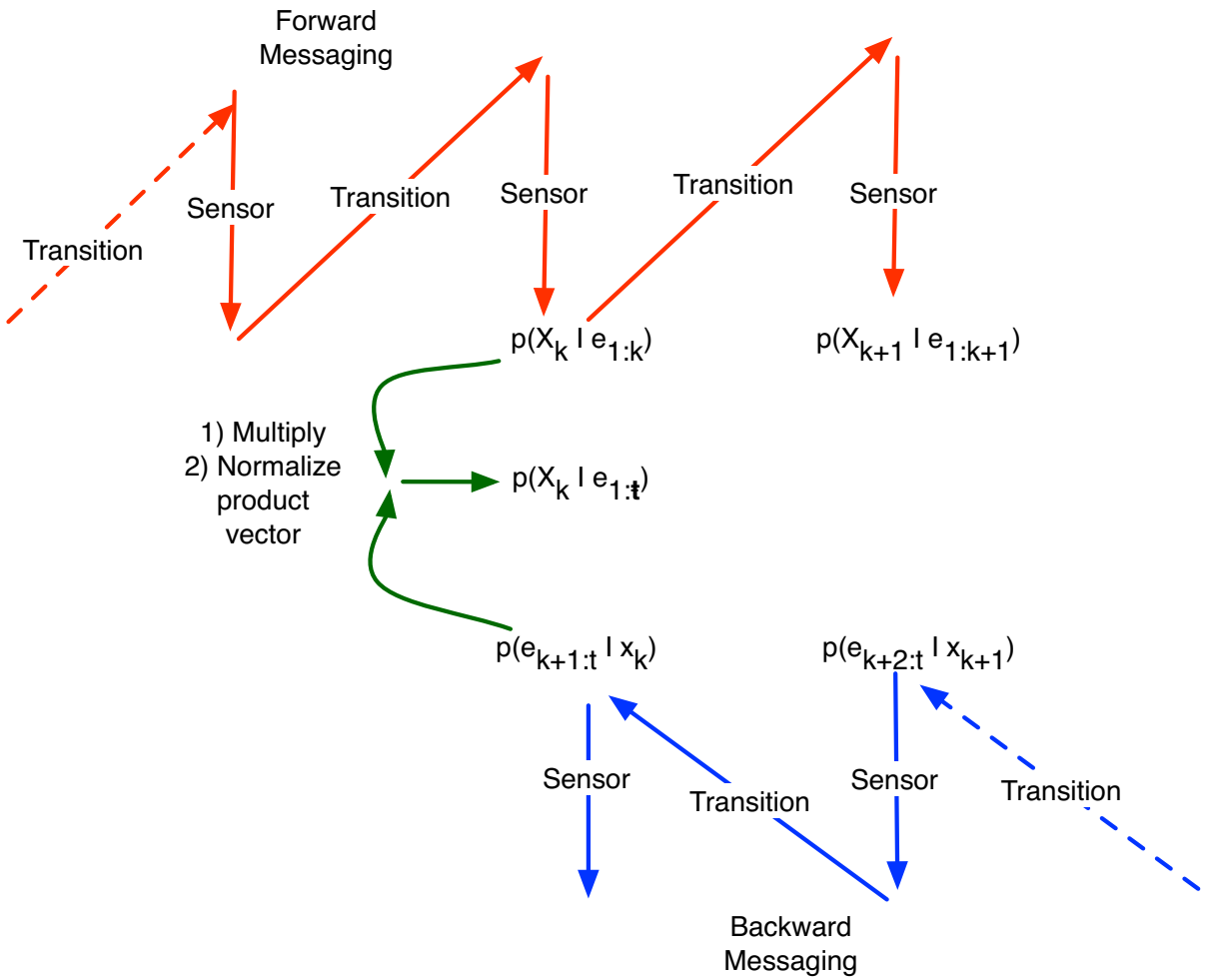


Smoothing example



Forward–backward algorithm: cache forward messages along the way
Time linear in t (polytree inference), space $O(t|\mathbf{f}|)$

Combining Forward and Backward Messages



Most likely explanation

Most likely sequence $\neq$ sequence of most likely states!!!!

Most likely path to each $\mathbf{x}_{t+1}$
= most likely path to **some** $\mathbf{x}_t$ plus one more step

$$\max_{\mathbf{x}_1 \ldots \mathbf{x}_t} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{X}_{t+1} | \mathbf{e}_{1:t+1})$$
$$= \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \left(\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{e}_{1:t}) \right)$$

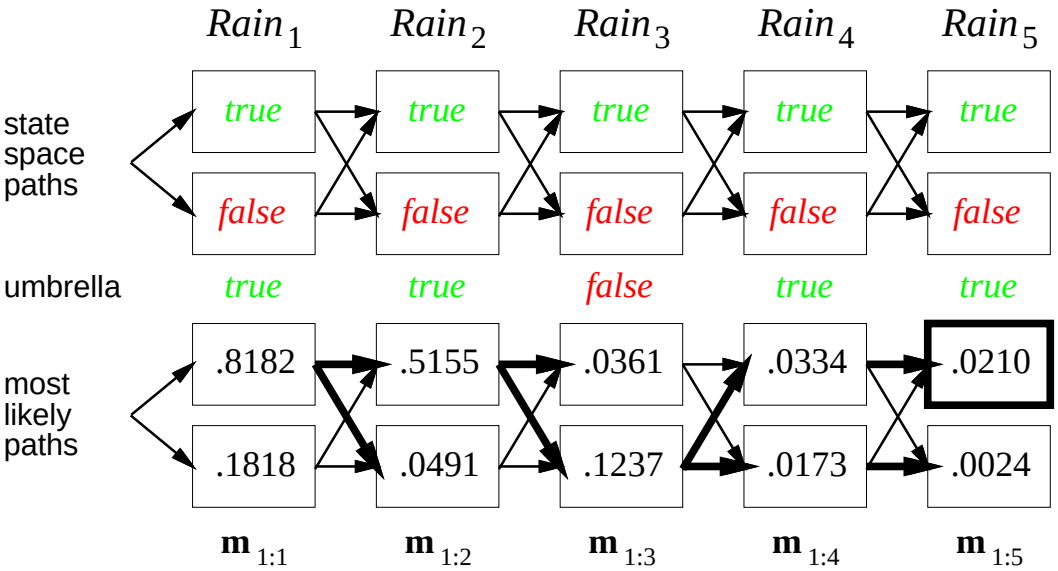
Identical to filtering, except $\mathbf{f}_{1:t}$ replaced by

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{X}_t | \mathbf{e}_{1:t}),$$

I.e., $\mathbf{m}_{1:t}(i)$ gives the probability of the most likely path to state i.
Update has sum replaced by max, giving the Viterbi algorithm:

$$\mathbf{m}_{1:t+1} = \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} (\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \mathbf{m}_{1:t})$$

Viterbi example



Hidden Markov models

$\mathbf{X}_t$ is a single, discrete variable (usually $\mathbf{E}_t$ is too)
Domain of X_t is $\{1, \ldots, S\}$

Transition matrix $\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i)$, e.g., $\begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

Sensor matrix $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t | X_t = i)$
e.g., with $U_1 = true$, $\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$

Forward and backward messages as column vectors:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{b}_{k+1:t} &= \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t} \end{aligned}$$

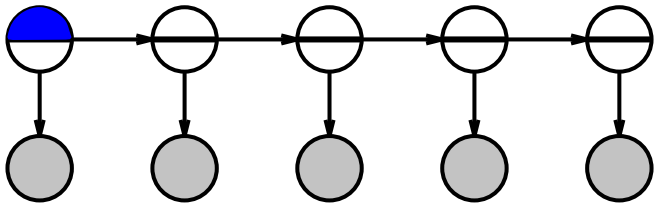
Forward-backward algorithm needs time $O(S^2 t)$ and space $O(St)$

Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

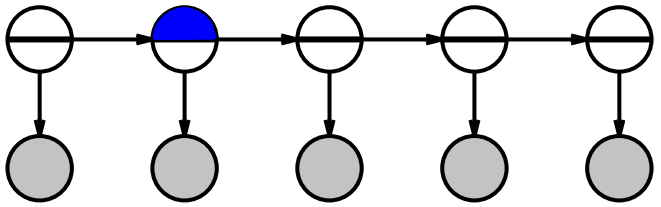


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

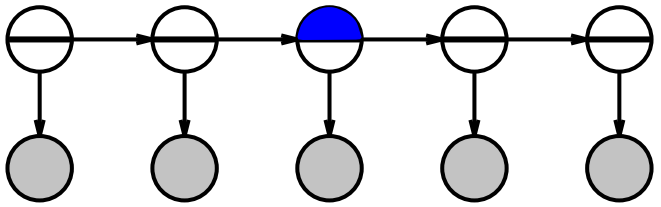


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

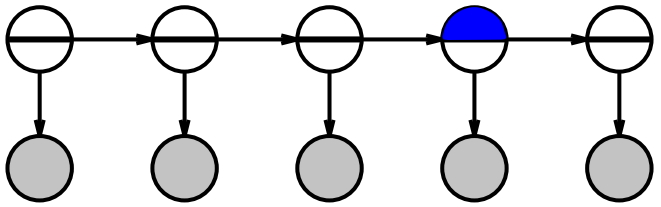


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

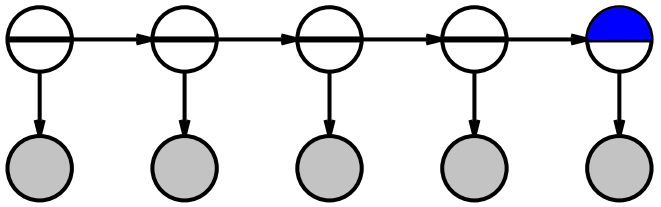


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

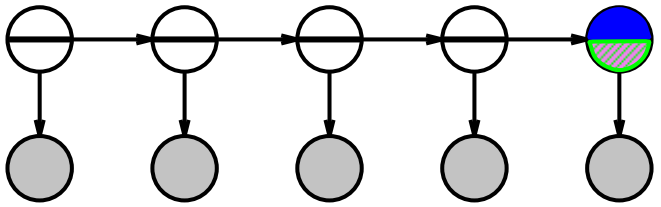


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

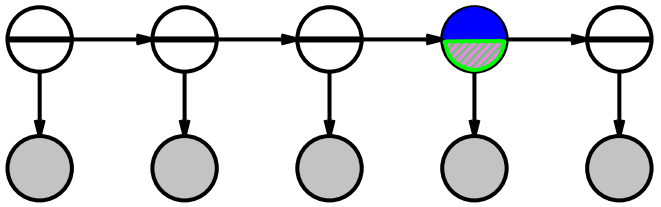


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha' (\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

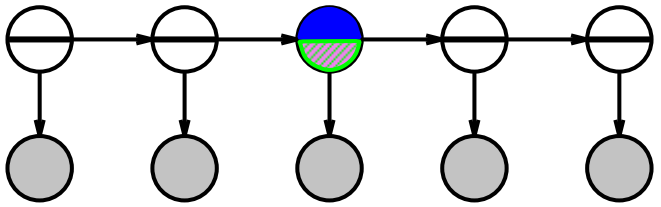


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

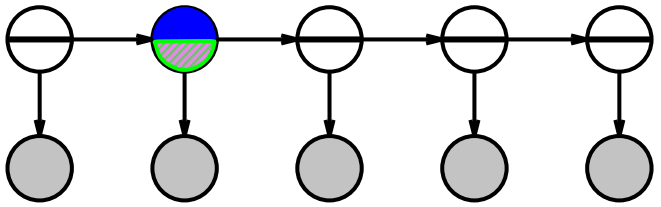


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$

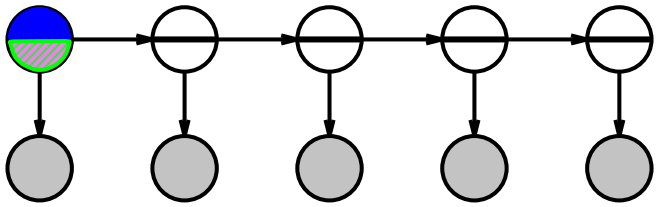


Country dance algorithm

Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

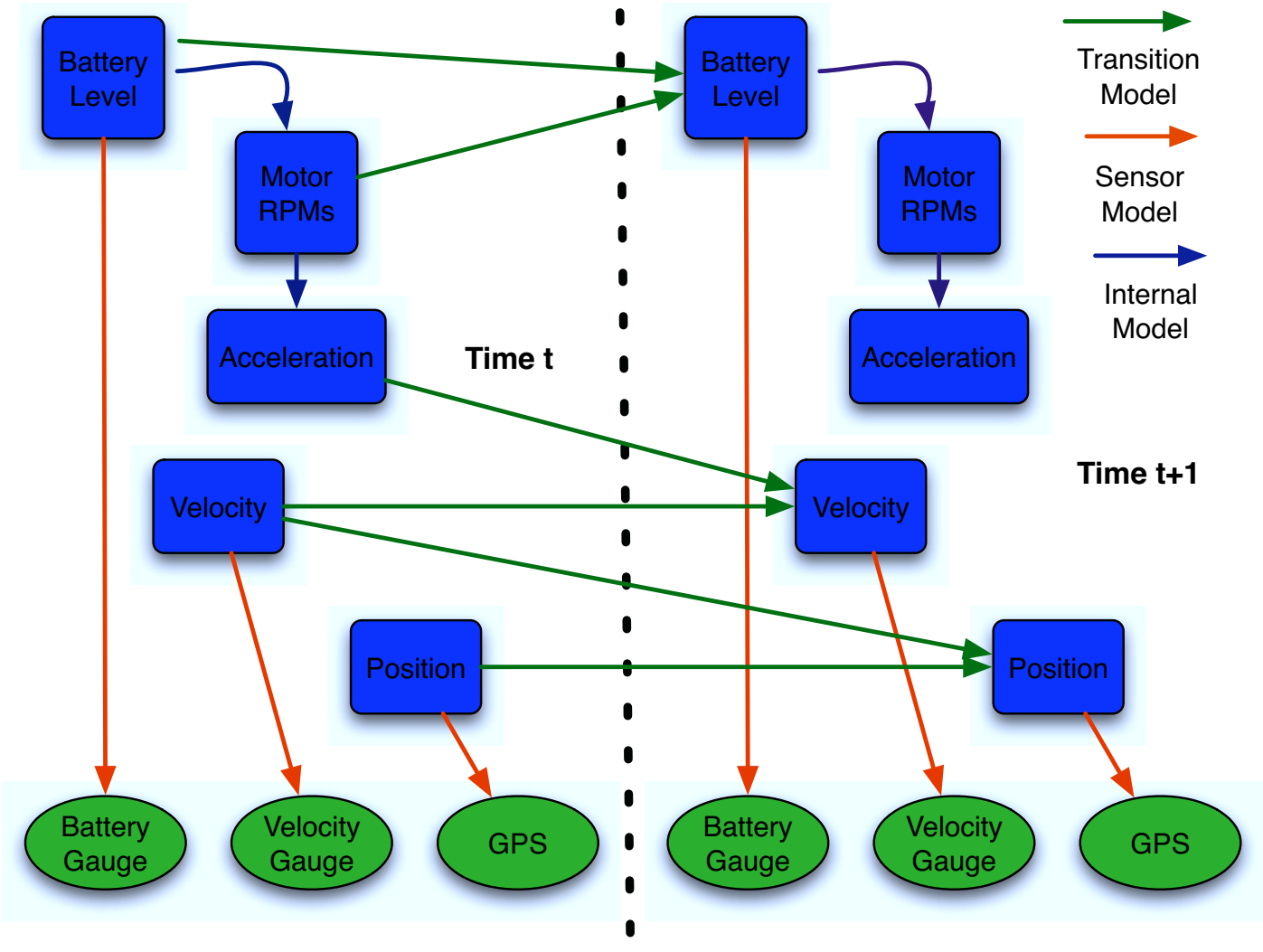
$$\begin{aligned} \mathbf{f}_{1:t+1} &= \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \\ \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \alpha \mathbf{T}^\top \mathbf{f}_{1:t} \\ \alpha'(\mathbf{T}^\top)^{-1} \mathbf{O}_{t+1}^{-1} \mathbf{f}_{1:t+1} &= \mathbf{f}_{1:t} \end{aligned}$$

Algorithm: forward pass computes $\mathbf{f}_t$, backward pass does $\mathbf{f}_i$, $\mathbf{b}_i$



Dynamic Bayesian networks (DBNs)

$\mathbf{X}_t$, $\mathbf{E}_t$ contain arbitrarily many variables in a replicated Bayes net



DBN Requirements

To build a DBN, you need:

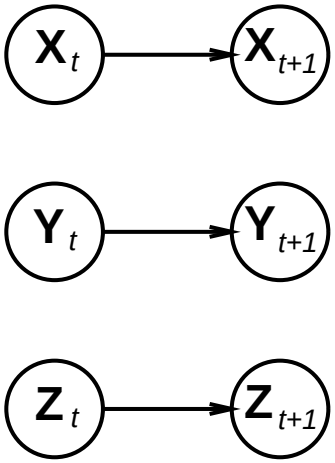
- Prior probabilities for the internal state variables: $\mathbf{X}_0$ values.
- Transition model: $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t)$.
- Sensor Model: $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$.

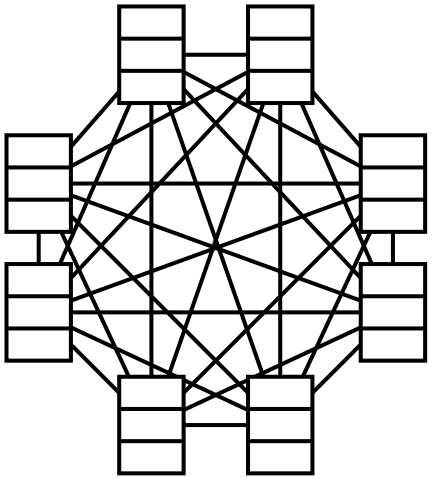
Given this, we can do filtering, predictive, smoothing and best-explanation tasks.
However, this can be expensive.
Each conditional prob query invokes the Bayesian Reasoner, which may:

- Generate an **exact** answer via enumeration.
- Generate an **approximate** answer via sampling.

DBNs vs. HMMs

Every HMM is a single-variable DBN; every discrete DBN is an HMM





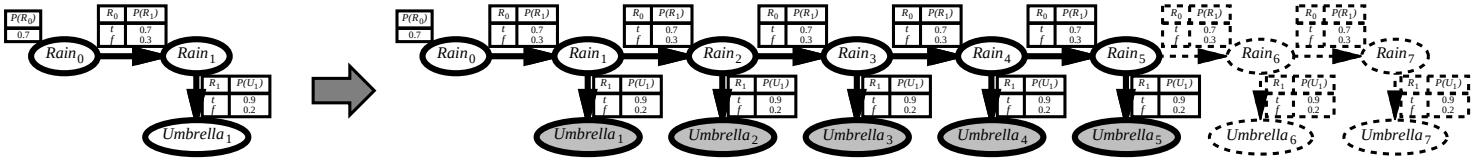
Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

e.g., 20 state variables, three parents each

DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$

Exact inference in DBNs

Naive method: unroll the network and run any exact algorithm



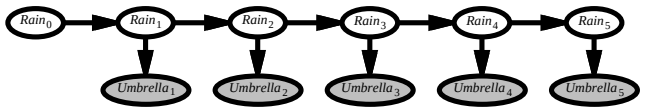
Problem: inference cost for each update grows with t

Rollup filtering: add slice $t+1$, "sum out" slice t using variable elimination

Largest factor is $O(d^{n+1})$, update cost $O(d^{n+2})$
(cf. HMM update cost $O(d^{2n})$)

Likelihood weighting for DBNs

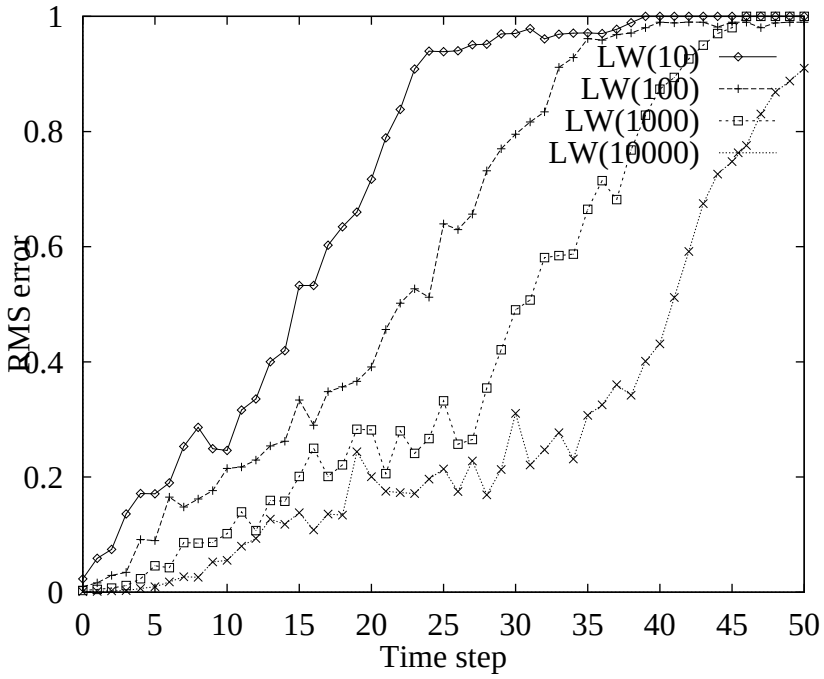
Set of weighted samples approximates the belief state



LW samples pay no attention to the evidence!

$\Rightarrow$ fraction "agreeing" falls exponentially with t

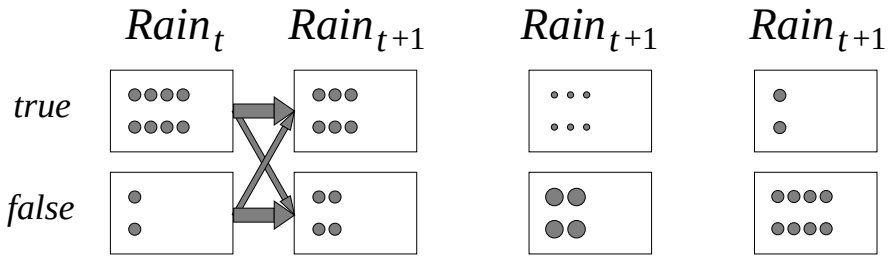
$\Rightarrow$ number of samples required grows exponentially with t



Particle filtering

Basic idea: ensure that the population of *particles* (i.e., sample events covering all state variables) tracks the high-likelihood regions of the state-space

Replicate particles proportional to likelihood for $\mathbf{e}_t$: how well the states explain the evidence.



(a) Propagate (b) Weight (c) Resample

Here, evidence at t+1 is not(umbrella)
Widely used for tracking nonlinear systems, esp. in vision
Also used for simultaneous localization and mapping in mobile robots 10^5-dimensional state space

Particle filtering Algorithm

1. Generate N sample events over all the **state** variables, using prior probs $\mathbf{X}_0$ as basis.
2. t = 0;
3. For each sample, x_t, propagate it forward to x_{t+1} using the Transition Model, $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t)$.
4. weight(x_{t+1}) = $\mathbf{P}(\mathbf{e}_{t+1} \mid x_{t+1})$. How well does the sample agree with evidence?
5. Resample the population based on the weights of samples at t+1:
   - Choose N new sample events from the current pool of N samples.
   - Highly-weighted samples will be chosen (hence replicated) many times.
   - Low-weight samples may not be chosen much (if at all).
6. If at final final time step: Stop, Base state probs on particle ratios.
7. ELSE: t = t + 1, goto step 3

Particle filtering contd.

Assume consistent at time t: $N(\mathbf{x}_t|\mathbf{e}_{1:t})/N = P(\mathbf{x}_t|\mathbf{e}_{1:t})$

Propagate forward: populations of $\mathbf{x}_{t+1}$ are

$$N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t})$$

Weight samples by their likelihood for $\mathbf{e}_{t+1}$:

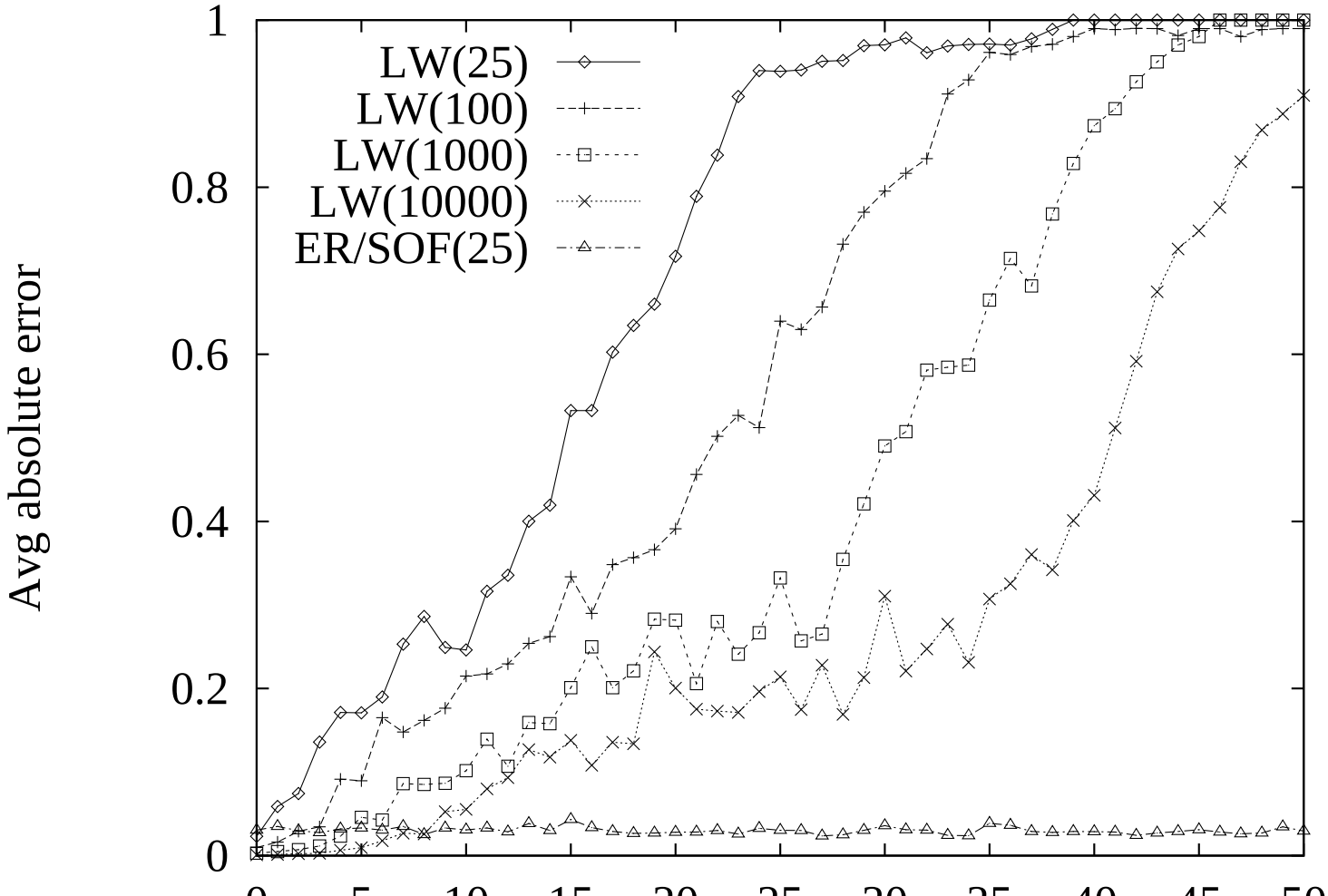
$$W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t})$$

Resample to obtain populations proportional to W:

$$\begin{aligned} N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1})/N &= \alpha W(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) N(\mathbf{x}_{t+1}|\mathbf{e}_{1:t}) \\ &= \alpha P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) N(\mathbf{x}_t|\mathbf{e}_{1:t}) \\ &= \alpha' P(\mathbf{e}_{t+1}|\mathbf{x}_{t+1}) \sum_{\mathbf{x}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{e}_{1:t}) \\ &= P(\mathbf{x}_{t+1}|\mathbf{e}_{1:t+1}) \end{aligned}$$

Particle filtering performance

Approximation error of particle filtering remains bounded over time, at least empirically—theoretical analysis is difficult



Summary

Temporal models use state and sensor variables replicated over time

Markov assumptions and stationarity assumption, so we need
- transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$
- sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$

Tasks are filtering, prediction, smoothing, most likely sequence;
all done recursively with constant cost per time step

Hidden Markov models have a single discrete state variable; used for speech recognition

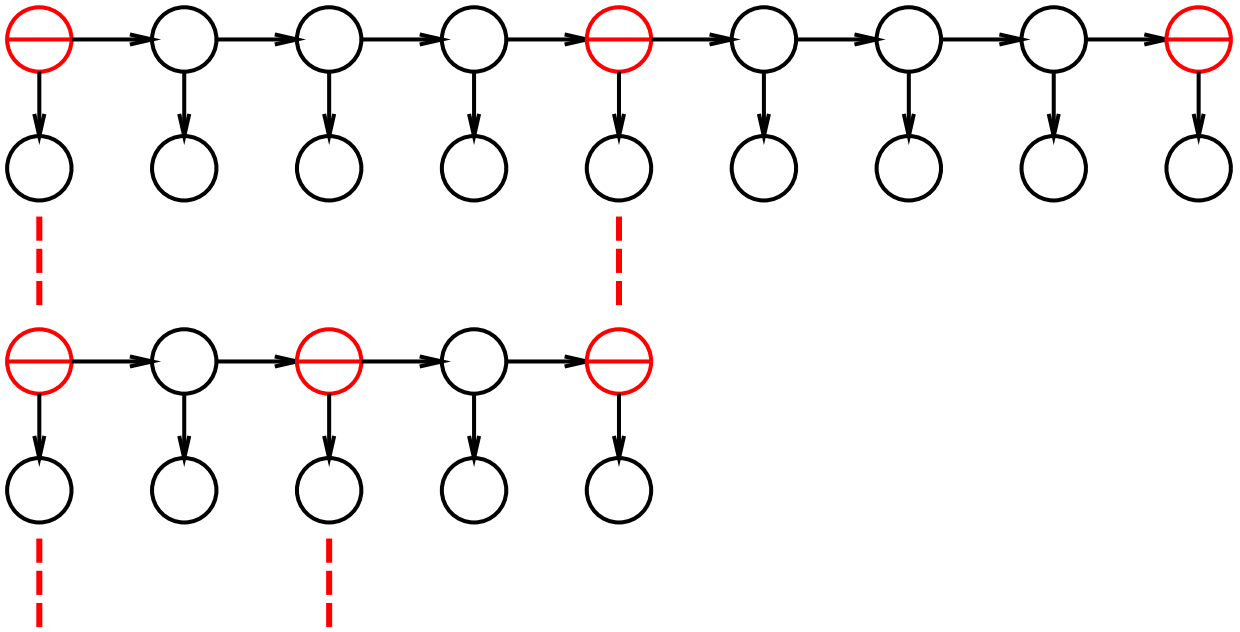
Kalman filters allow n state variables, linear Gaussian, $O(n^3)$ update

Dynamic Bayes nets subsume HMMs, Kalman filters; exact update intractable

Particle filtering is a good approximate filtering algorithm for DBNs

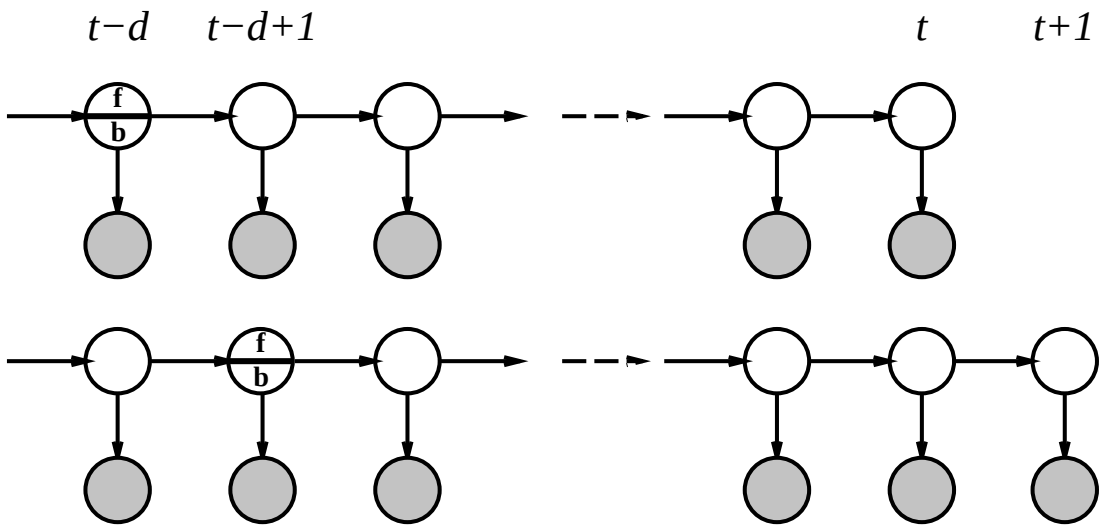
Island algorithm

Idea: run forward-backward storing $\mathbf{f}_t$, $\mathbf{b}_t$ at only $k-1$ points
Call recursively (depth-first) on k subtasks



$O(k|\mathbf{f}| \log_k t)$ space, $O(k \log_k t)$ more time

Online fixed-lag smoothing



Obvious method runs forward–backward for d steps each time

Recursively compute $\mathbf{f}_{1:t-d+1}$, $\mathbf{b}_{t-d+2:t+1}$ from $\mathbf{f}_{1:t-d}$, $\mathbf{b}_{t-d+1:t}$?

Forward message OK, backward message not directly obtainable

Online fixed-lag smoothing contd.

Define $\mathbf{B}_{j:k} = \prod_{i=j}^{k} \mathbf{T}\mathbf{O}_i$, so

$$\begin{aligned} \mathbf{b}_{t-d+1:t} &= \mathbf{B}_{t-d+1:t}\mathbf{1} \\ \mathbf{b}_{t-d+2:t+1} &= \mathbf{B}_{t-d+2:t+1}\mathbf{1} \end{aligned}$$

Now we can get a recursive update for $\mathbf{B}$:

$$\mathbf{B}_{t-d+2:t+1} = \mathbf{O}_{t-d+1}^{-1}\mathbf{T}^{-1}\mathbf{B}_{t-d+1:t}\mathbf{T}\mathbf{O}_{t+1}$$

Hence update cost is constant, independent of lag d

Approximate inference in DBNs

Particle filtering (Gordon, 1994; Kanazawa, Koller, and Russell, 1995; Blake and Isard, 1996)

Factored approximation (Boyen and Koller, 1999)

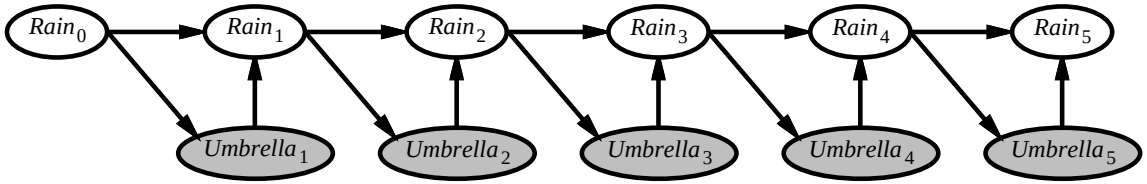
Loopy propagation (Pearl, 1988; Yedidia, Freeman, and Weiss, 2000)

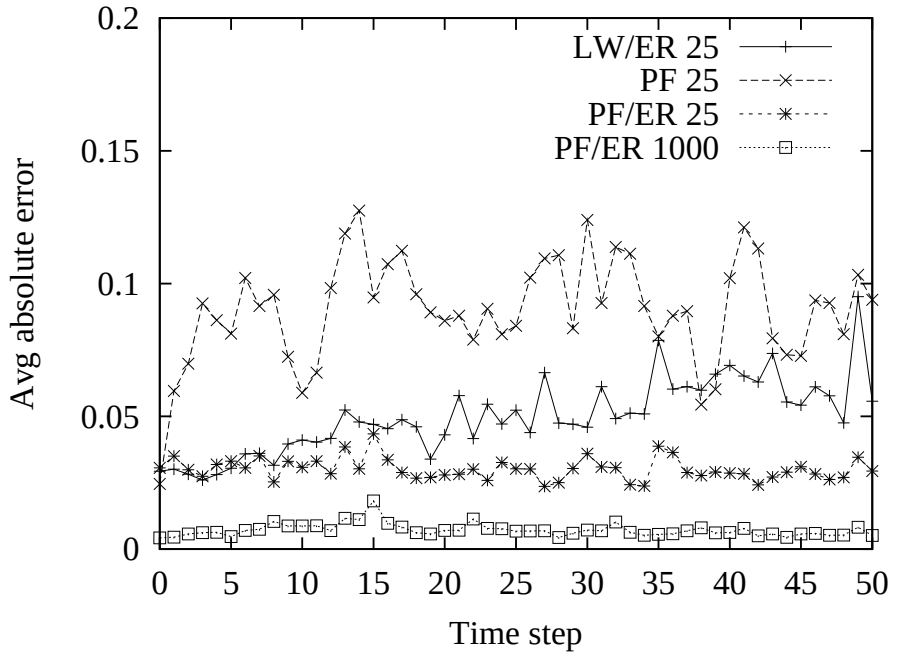
Variational approximation (Ghahramani and Jordan, 1997)

Decayed MCMC (unpublished)

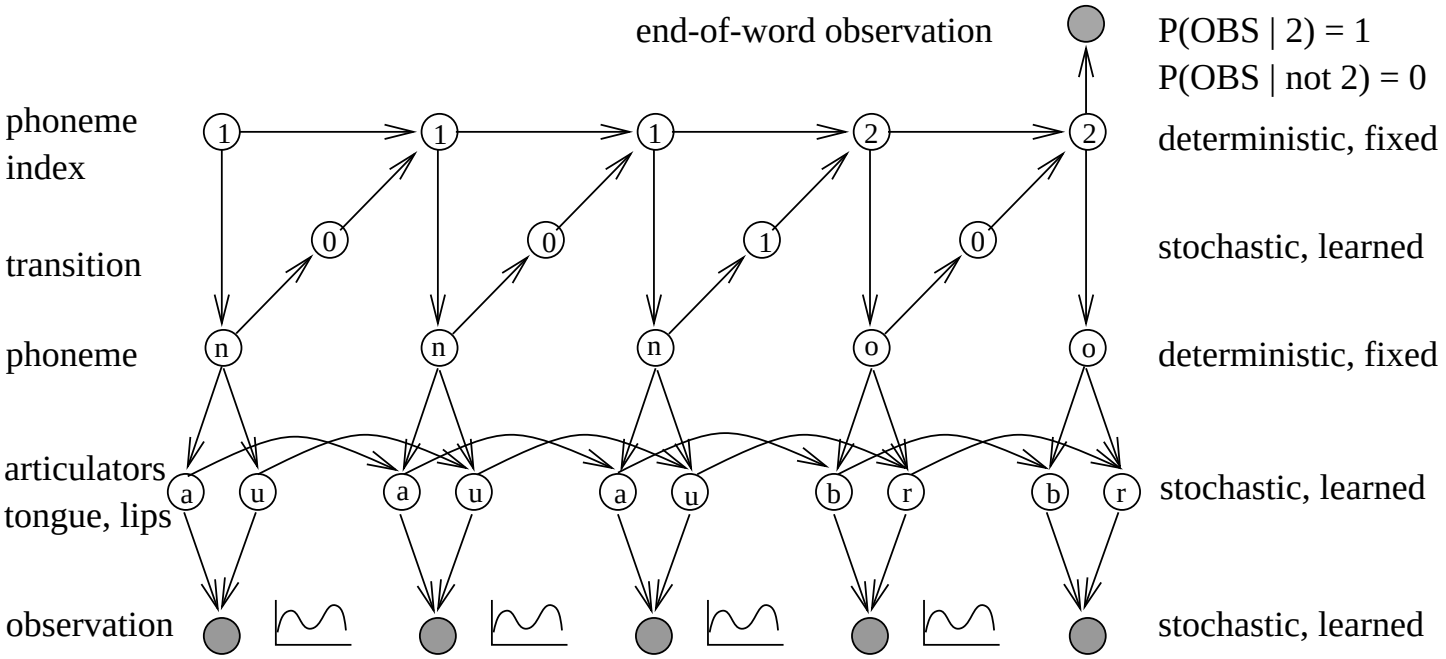
Evidence reversal

Better to propose new samples conditioned on the new evidence
Minimizes the variance of the posterior estimates (Kong & Liu, 1996)





Example: DBN for speech recognition



Also easy to add variables for, e.g., gender, accent, speed.

Zweig and Russell (1998) show up to 40% error reduction over HMMs