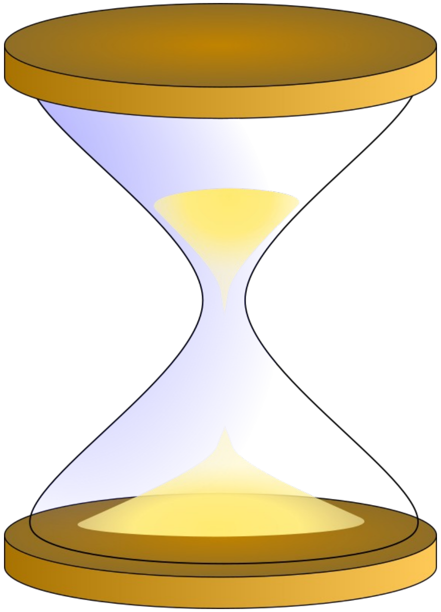
Probabilistic Artificial Intelligence

Reasoning over time



2023-2024

Department of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

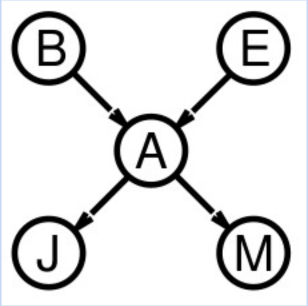
Approximate Inference

Recap

- Probability: useful for representing beliefs of our agents
  - ▹ Bayes' rule to update beliefs
- Compactly representing
  - ▹ Bayesian networks
  - ▹ exploits conditional independence
- Inference intractable in general…

  → approximate inference

P(B|j,m)?



TU Delft

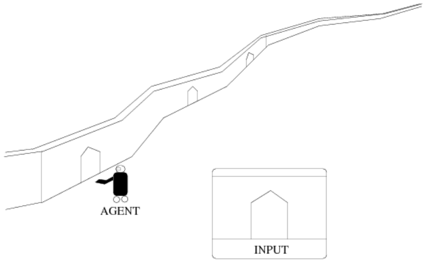
It allows to *maintain* a belief

- Given conditional independent observations
  $\mathbf{P}(o_1,o_2|State) = \mathbf{P}(o_1|State)\mathbf{P}(o_2|State)$

- ...we can also sequentially update:

  $\mathbf{P}'(State) := \mathbf{P}(o_1|State)\mathbf{P}(State) / P(o_1)$
  $\mathbf{P}''(State) := \mathbf{P}(o_2|State)\mathbf{P}'(State) / P(o_2)$

  - then $\mathbf{P}''(State)=\mathbf{P}(State\ |o_1,o_2)$
  - Exercise...?



- "We will see later how to incorporate robot movement over time"
  → the time has come!

It allows to *maintain* a belief

- Given conditional independent observations
  P$(o_1,o_2|State)$ = ***P***$(o_1|State)$***P***$(o_2|State)$

- …we can also sequentially update:

  P'$(State) :=$ ***P***$(o_1|State)$***P***$(State) / P(o_1)$
  P''$(State):=$ ***P***$(o_2|State)$***P'***$(State) / P(o_2)$

  - then ***P''***$(State)$=***P***$(State\ |o_1,o_2)$
  - Exercise…?

$$\boldsymbol{P}(S|o_1,o_2) = \frac{P(o_1,o_2|S)P(S)}{\sum_s P(o_1,o_2|s)P(s)}$$

$$\{\text{conditional independence}\} = \frac{P(o_2|S)P(o_1|S)P(S)}{\sum_s P(o_2|s)P(o_1|s)P(s)}$$

$$\{\text{multiply with 1}\} = \frac{1/P(o_1)}{1/P(o_1)} \times \frac{P(o_2|S)P(o_1|S)P(S)}{\sum_s P(o_2|s)P(o_1|s)P(s)}$$

$$\{\text{push inward}\} = \frac{P(o_2|S)\frac{P(o_1|S)P(S)}{P(o_1)}}{\sum_s P(o_2|s)\frac{P(o_1|s)P(s)}{P(o_1)}}$$

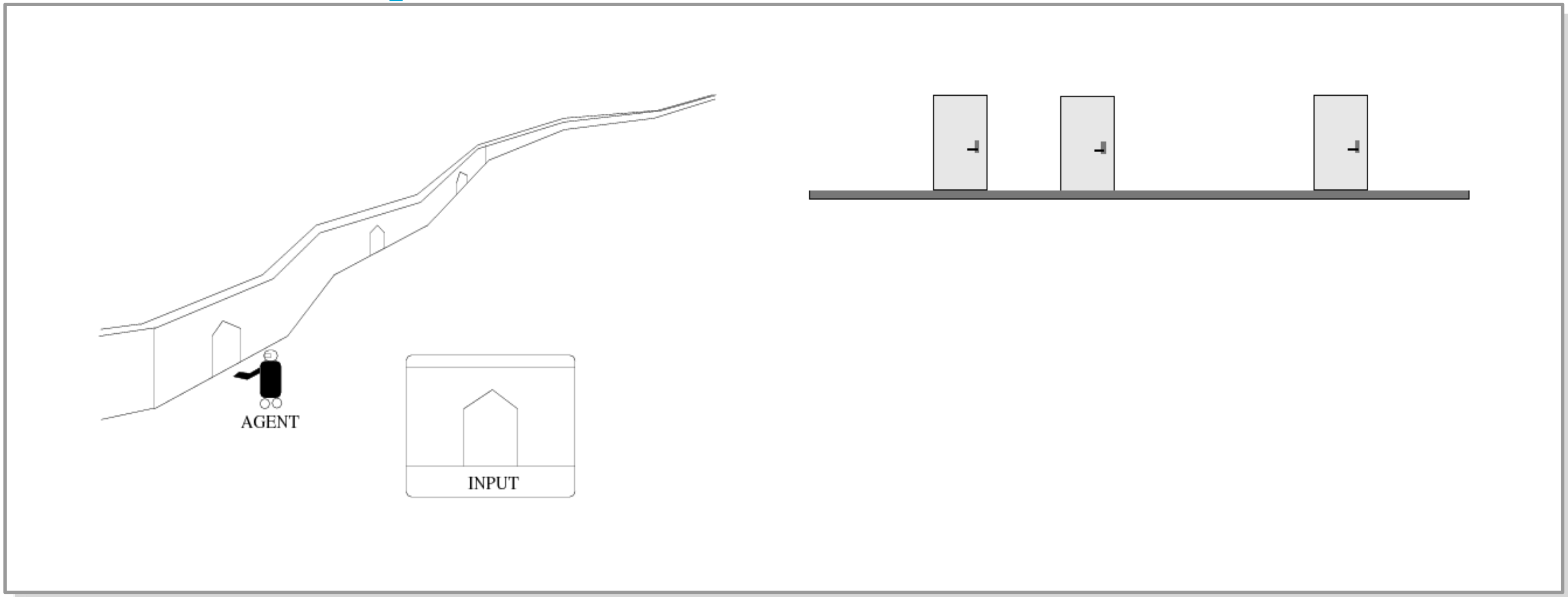
$$= \frac{P(o_2|S)P'(S)}{\sum_s P(o_2|s)P'(s)} = P''(\text{state})$$

with

$$P'(S) \triangleq \frac{P(o_1|S)P(S)}{P(o_1)}$$

the updated belief after observing o_1.

Warm Up Exercise



- Assume the history is:
  (see nothing, move right, see door, move right, see door)
  - where are we now?
  - where did we start?

Motivation

Why do we need special methods for time?

- Bayes rule is awesome...
  - but perhaps not (directly) sufficient to deal with time...?
  - $\boldsymbol{P}(State|o_1) := \boldsymbol{P}(o_1|State)\boldsymbol{P}(State) / P(o_1)$

- Problems?

Why do we need special methods for time?

- Bayes rule is awesome…
  - but perhaps not (directly) sufficient to deal with time…?
  - $\boldsymbol{P}(State|o_1) := \boldsymbol{P}(o_1|State)\boldsymbol{P}(State) / P(o_1)$

- Problems?

E.g.: a self-built drone that determines its position based on images it takes with its camera…

- to make it easier the designers place a landmark
- accuracy is important… the measurement is noisy… so designers decide to average all measurements over the course of 20s…

Will it work?

The World Changes

- … position could vary substantially over 20s interval

The World Changes

- … position could vary substantially over 20s interval
- Another example: treating a diabetic patient
  - need to decide on food intake and insulin dose
  - need to estimate: current blood sugar and insulin levels – these can vary quickly

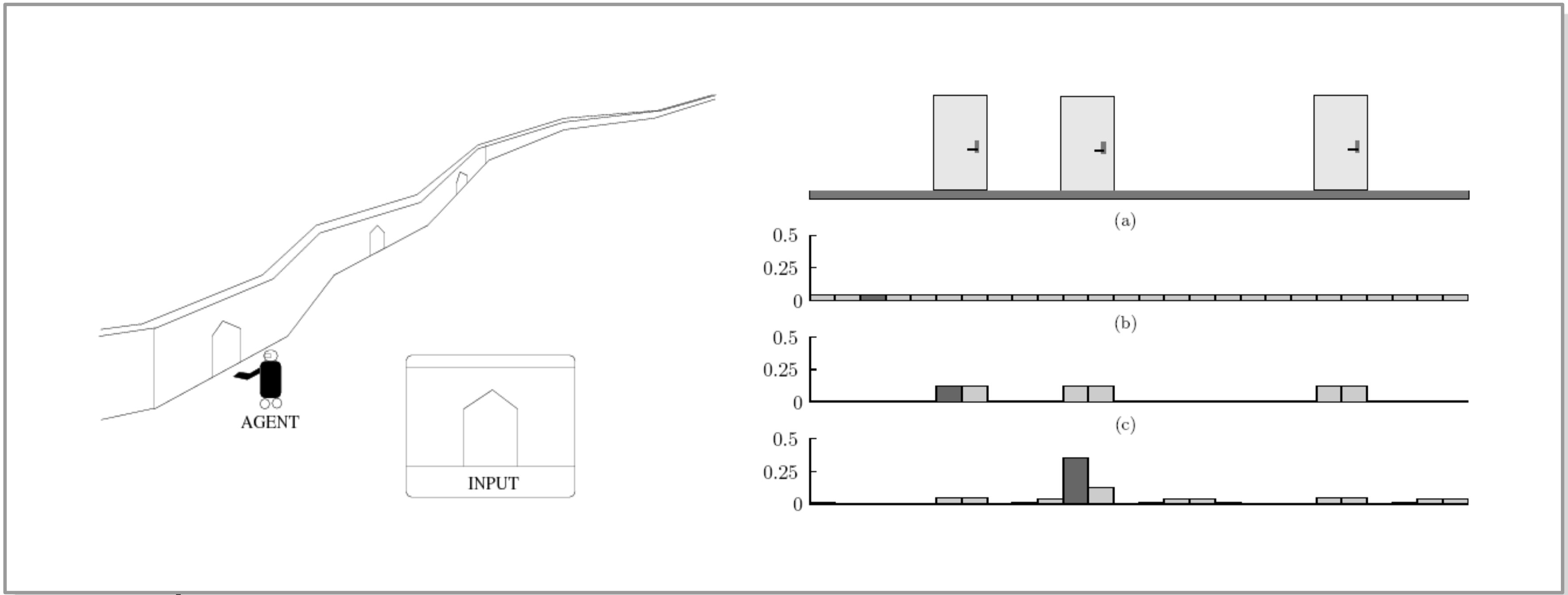
The World Changes

- … position could vary substantially over 20s interval
- Another example: treating a diabetic patient
  - need to decide on food intake and insulin dose
  - need to estimate: current blood sugar and insulin levels – these can vary quickly

- Upshot:
  - need not only consider “sensor model” $\boldsymbol{P}(o_1 | State)$
  - but also a “transition model” that predicts how the state changes over time.

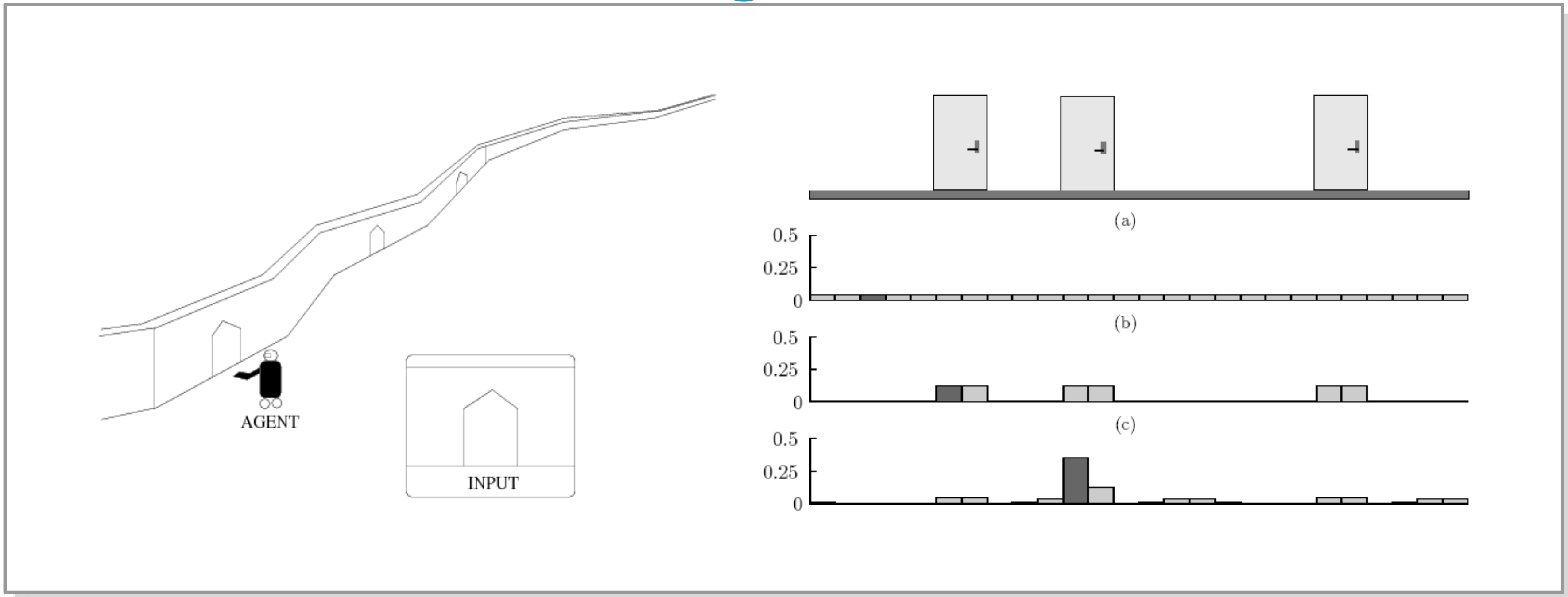
TU Delft

The World Changes



- Upshot:
  - need not only consider “sensor model” $\boldsymbol{P}(o_1|State)$
  - but also a “transition model” that predicts how the state changes over time.

The World Changes



- Upshot:
  - need not only consider "sensor model" $\boldsymbol{P}(o_1|State)$
  - but also a "transition model" that predicts how the state changes over time.

Representing Time

Representing Time

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

Representing Time

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

Question:
How can we compactly represent a probability distribution over the evolution of state X_t ?

Markov Processes (Markov Chains)

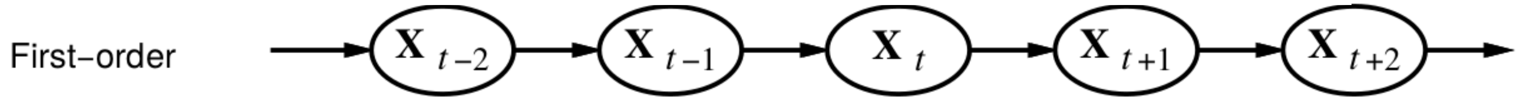
Construct a Bayes net from these variables: parents?

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: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ ‘transition model’



Markov Processes (Markov Chains)

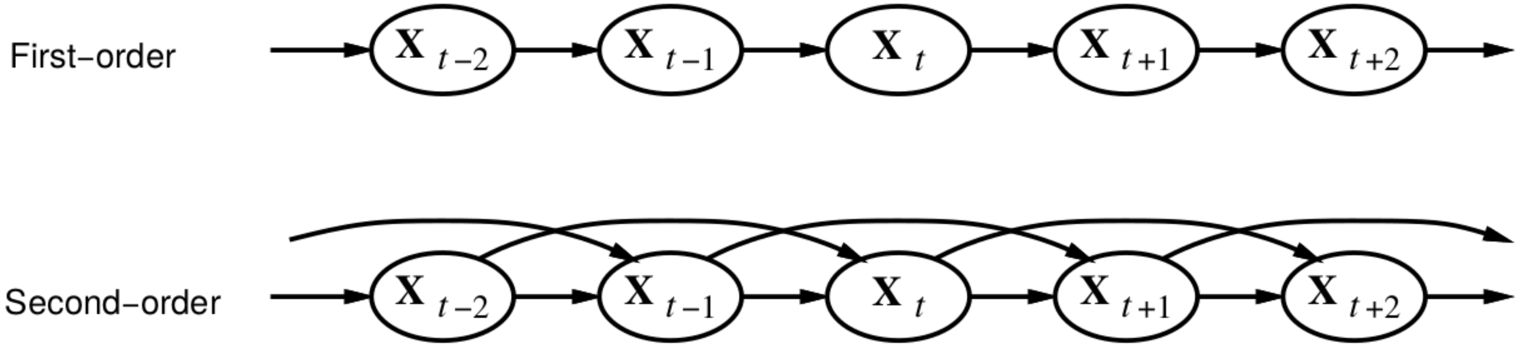
Construct a Bayes net from these variables: parents?

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

'transition model'

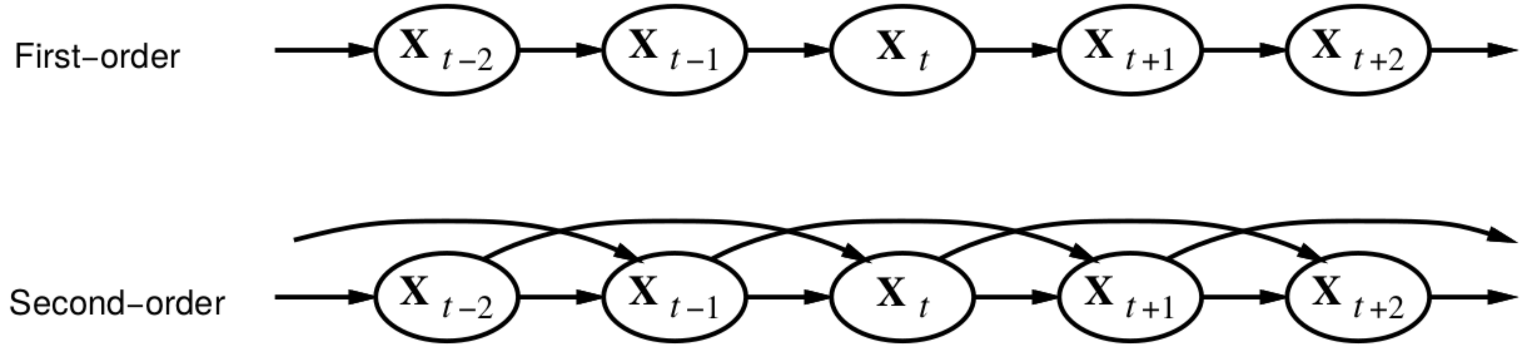


Markov Processes (Markov Chains)

Construct a Bayes net from these variables: parents?

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

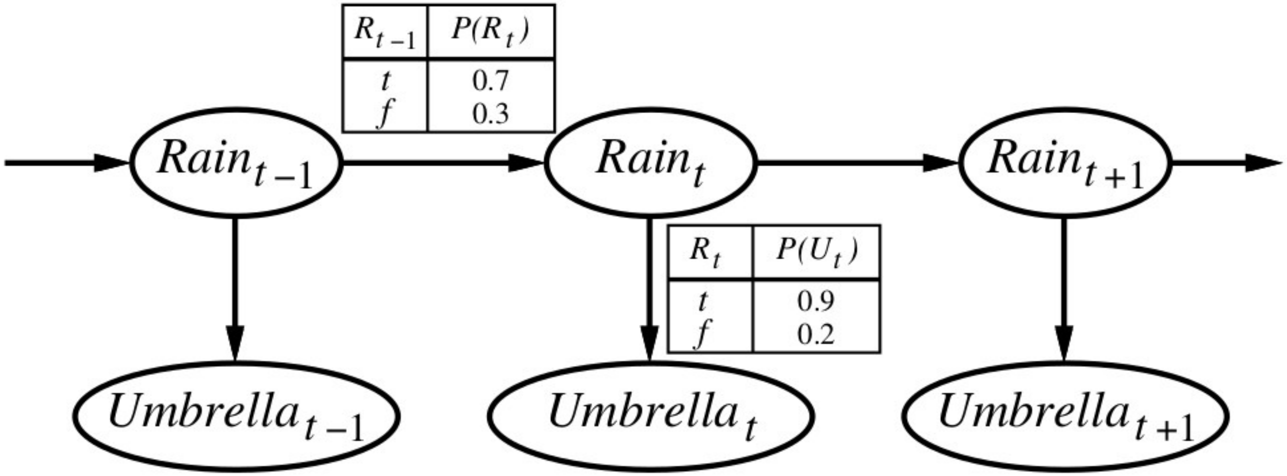
First-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ ‘transition model’
Second-order Markov process: $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1})$



Sensor Markov assumption: $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ ‘observation model’

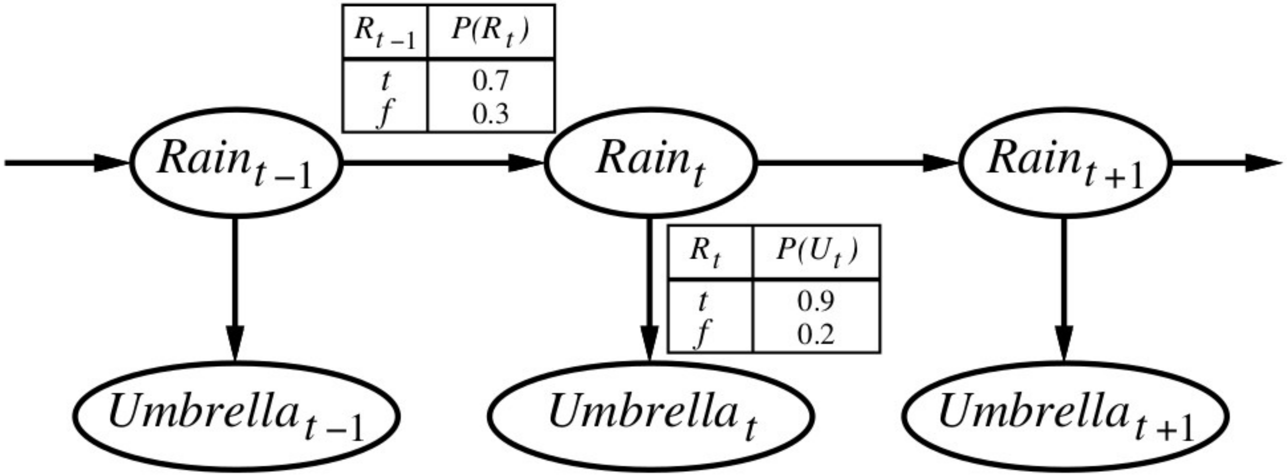
Stationary process: transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ and sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ fixed for all t

Example



R_{t-1} | $P(R_t)$
t 0.7
f 0.3
$Rain_{t-1}$
$Rain_t$
$Rain_{t+1}$
R_t | $P(U_t)$
t 0.9
f 0.2
$Umbrella_{t-1}$
$Umbrella_t$
$Umbrella_{t+1}$

Example



This is called a **hidden Markov model (HMM)**

- state consists of a single discrete variable

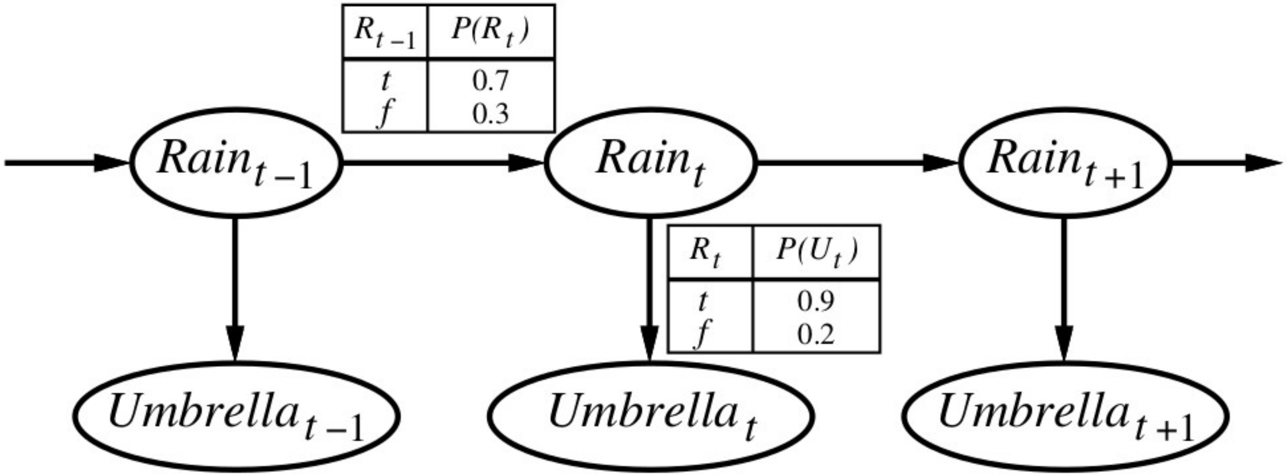
Semantics of Markov Chains

- Transition model + observation model
  + initial state distribution $\boldsymbol{P(X_0)}$ = joint PD

  $$\boldsymbol{P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1:t} P(X_i \mid X_{i-1})\, P(E_i \mid X_i)}$$

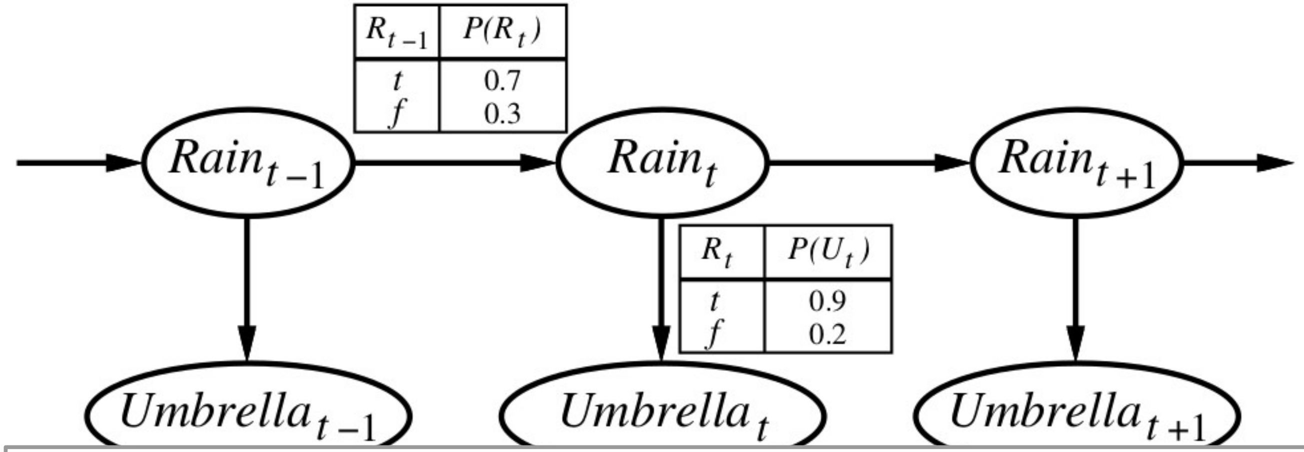
- "Unrolled over time" it is just a Bayesian network
  - note: we typically assume that $\boldsymbol{P(X_0)}$ captures all our knowledge at t=0, and hence there is no observation $\boldsymbol{E_0}$
    (as a convention)

The Markov Assumption…?



Is it appropriate?

The Markov Assumption…?



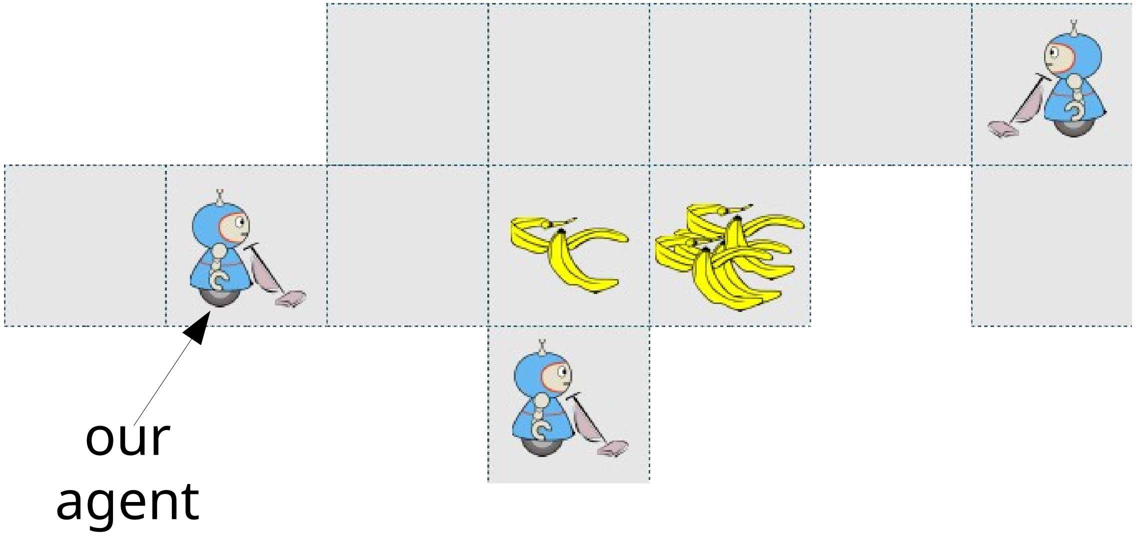
First-order Markov assumption not exactly true in real world!

Possible fixes:

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

Markov Assumption – 2

- Or how about "Spatial Task Allocation Problems"?
  - Need to decide where we go to clean...
  - How will the state change?
  - What information do we need?

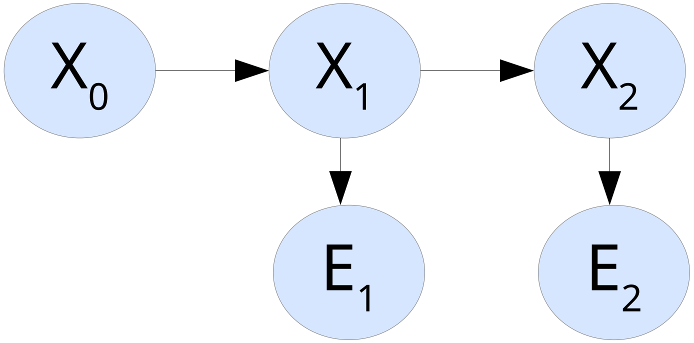


Aleksander Czechowski and Frans A. Oliehoek. Decentralized MCTS via Learned Teammate Models. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI), pp. 81–88, July 2020.

TU Delft

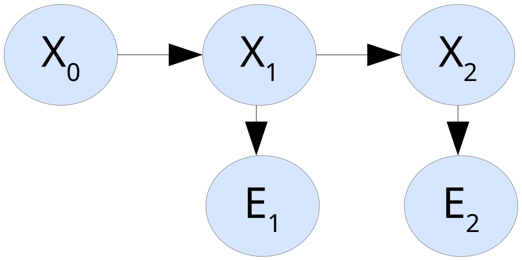
Inference in Hidden Markov Models

Inference on Markov Chains



- Unrolled we have a BN...
  - So we can use VE, rejection sampling, likelihood weighting etc...
- ...but what can we ask?

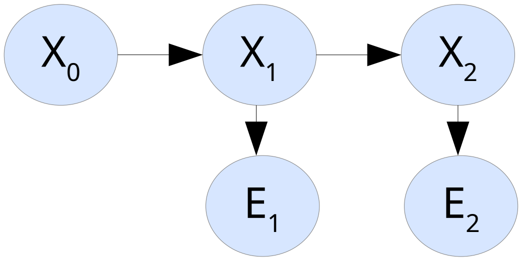
Types of Queries



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

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

Types of Queries



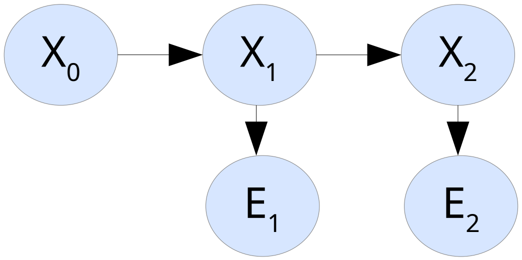
Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$

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

Types of Queries



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

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

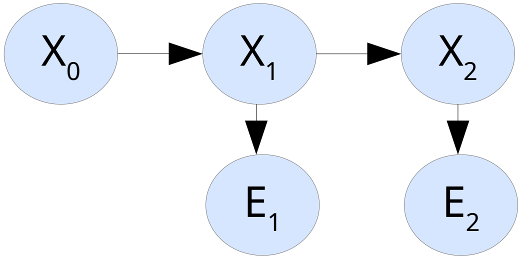
Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$

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

Types of Queries



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

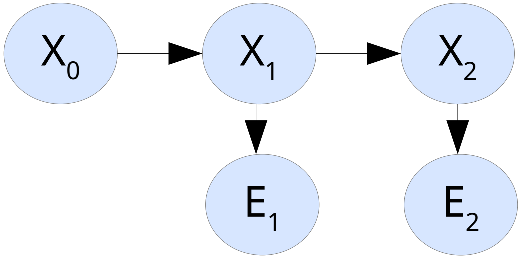
Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$

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

Types of Queries



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 $0 \leq k < t$

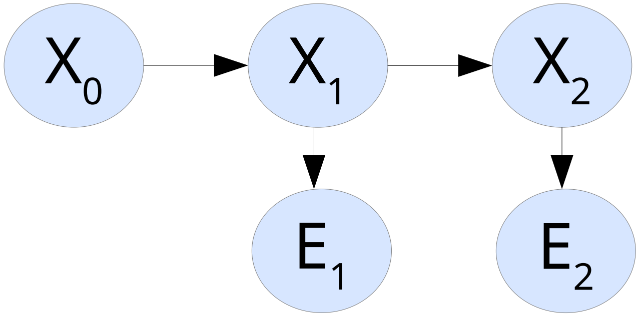
better estimate of past states, essential for learning

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

many names:
belief tracking, state estimation, recursive Bayesian estimation, etc.

Focus: Inference in HMMs

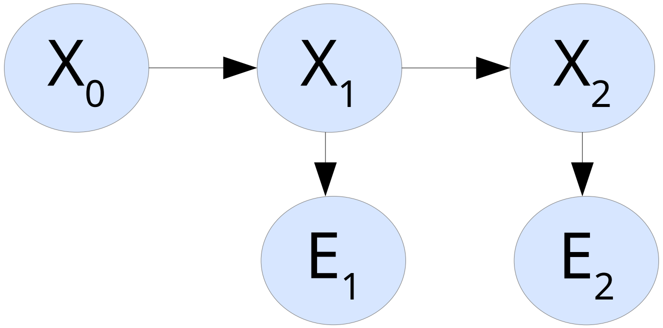


- We focus on HMMs, but...
  - ...can be generalized (cf. R&N)
  - e.g., continuous states: Kalman filter

- **Inference in HMMs**: answering queries *given* the HMM parameters
  - If we are trying to infer those parameters themselves: **learning HMMs**

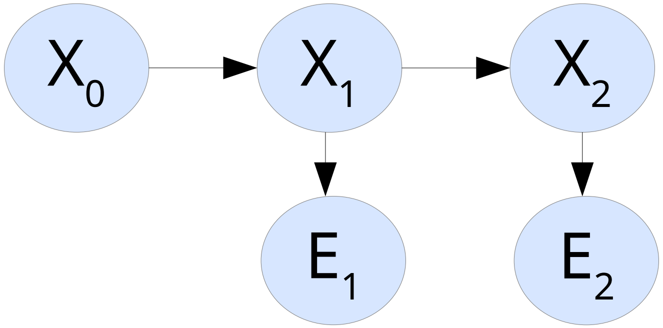
Filtering: $b_t(X_t) = P(X_t | e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



the term ‘belief’ and notation b(X) (or b(s)) is very common in reinforcement learning and planning under uncertainty

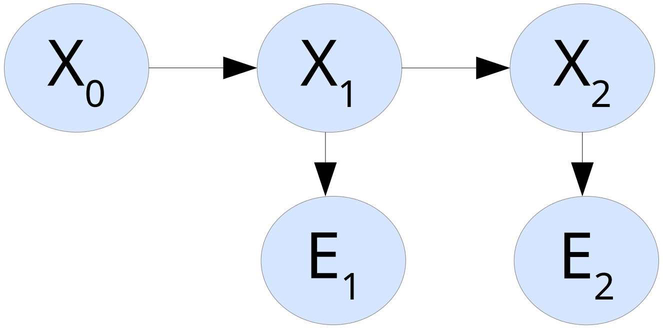
Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$



- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$
- Why?

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$

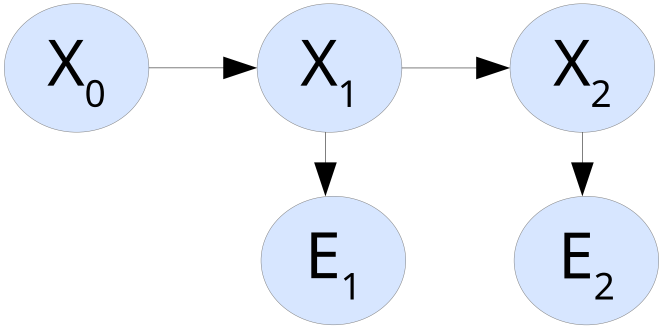
- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



- How?

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$

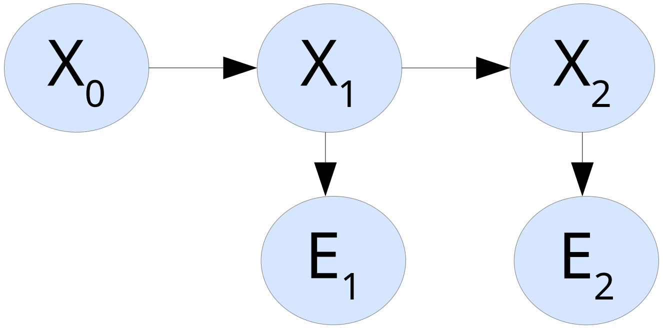
- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



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

Filtering: $\mathbf{b}_t(X_t)=\mathbf{P}(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$

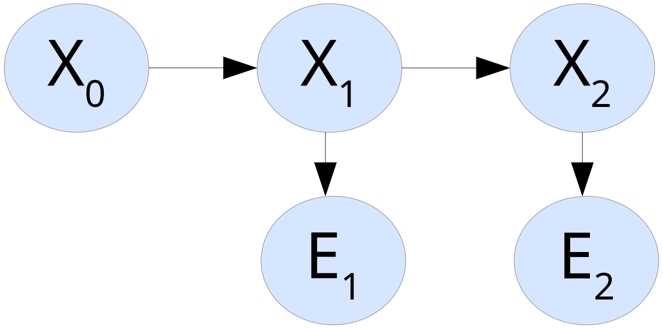


$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1})$$
$$= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$
$$= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

?

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$

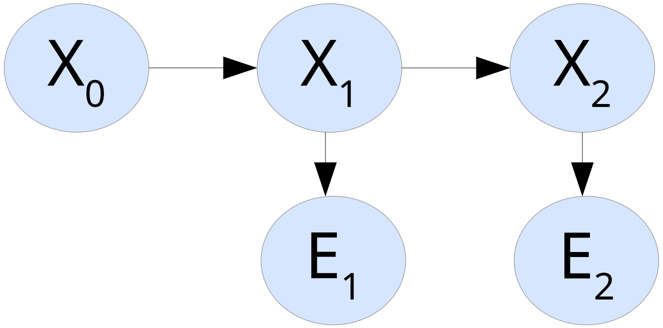


$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1})\\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})\\ &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})\end{aligne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

$P(X_{t+1}|e_{1:t},e_{t+1})=$
$P(e_{t+1}|X_{t+1},e_{1:t})P(X_{t+1}|e_{1:t}) \,/\, P(e_{t+1}|e_{1:t})$

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$

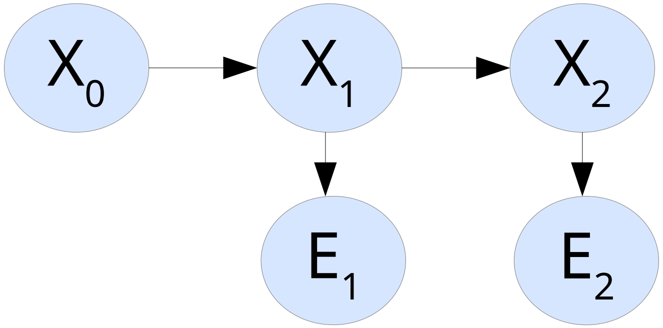


$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1})$$
$$= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$
$$= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

?

Filtering: $b_t(X_t)=P(X_t|e_{1:t})$

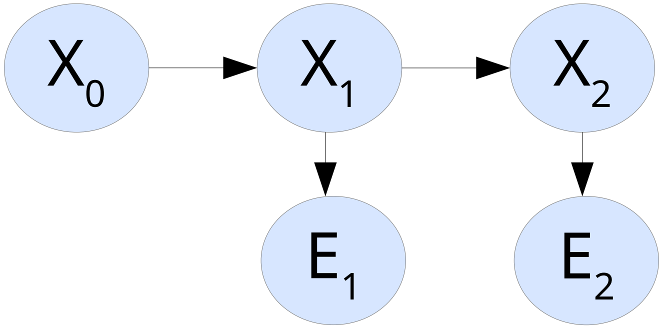
- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



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

sensor Markov assumption!

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$



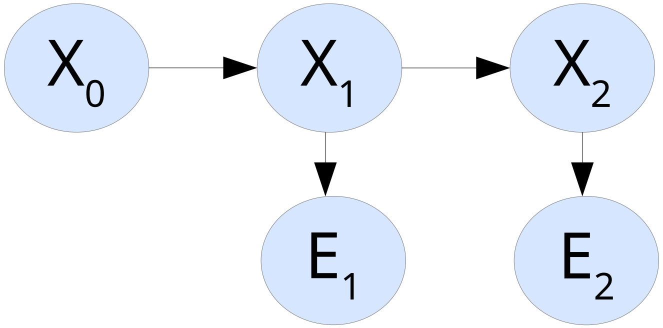
- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$

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

I.e., prediction + estimation.

Filtering: $b_t(X_t) = P(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$

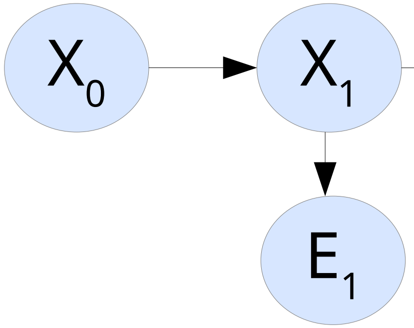


$$
\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})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

I.e., prediction + estimation.

Filtering: $b_t(X_t) = P(X_t|e_{1:t})$

Prediction:

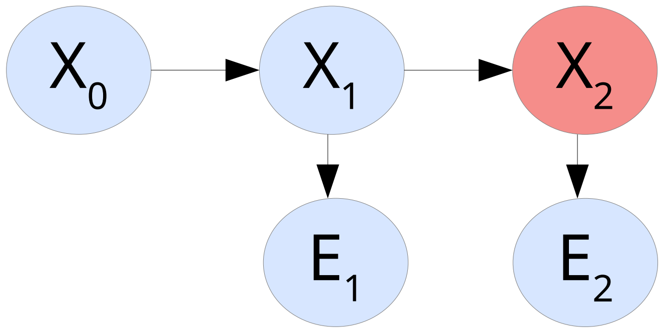


$$= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

I.e., prediction + estimation.

Filtering: $b_t(X_t) = P(X_t|e_{1:t})$

and estimation:

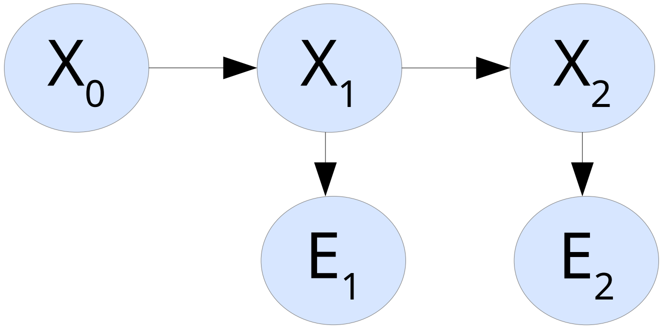


$$= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

I.e., prediction + estimation.

Filtering: $b_t(X_t)=P(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



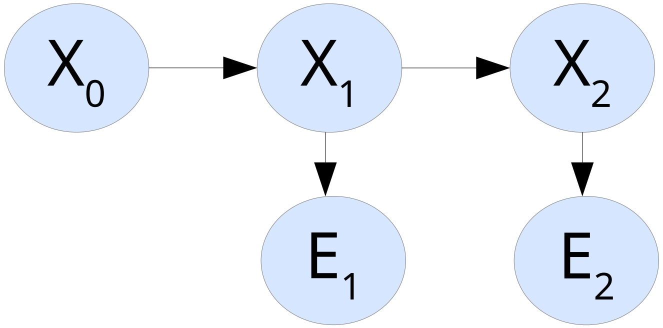
$$\begin{aligned}
\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})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

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

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

Filtering: $\boldsymbol{b}_t(X_t)=\boldsymbol{P}(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - new belief $\mathbf{b}_{t+1}(X_{t+1})$
  - from old belief $\mathbf{b}_t(X_t)$



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

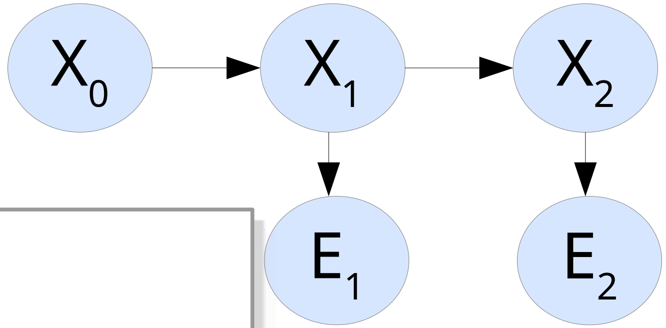
I.e., prediction + estimation. Prediction by summing out $\mathbf{X}_t$:

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

Markov assumption

Filtering: $b_t(X_t)=P(X_t|e_{1:t})$

- Ideal: a recursive way to compute
  - ▷ new belief $\mathbf{b}_{t+1}(X_{t+1})$



So now, we have this expression of the form

$$\boldsymbol{b}_{t+1}(X_{t+1}) = \alpha\, \boldsymbol{P}(e_{t+1} \,|\, X_{t+1})\, \Sigma_x\, \boldsymbol{P}(X_{t+1} \,|\, x_t)\, \boldsymbol{b}_t(x_t)$$

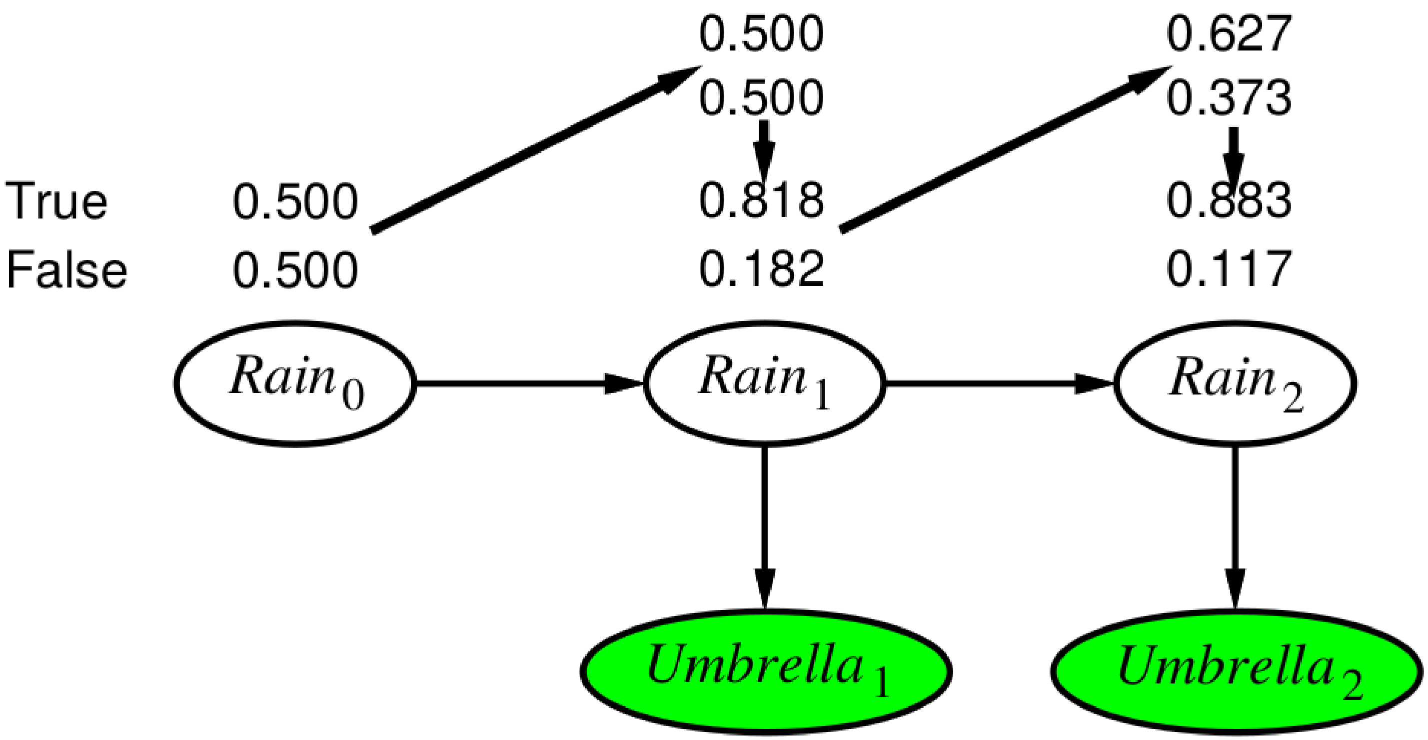
$$= \alpha \text{ Forward}(\, \boldsymbol{b}_t(X_t), e_{t+1})$$

- the $\boldsymbol{b}_t(X_t)$ are also called **forward messages**, $\mathbf{f}_{1:t}$
- initialize: $\mathbf{f}_{1:0} = \boldsymbol{P}(X_0)$

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

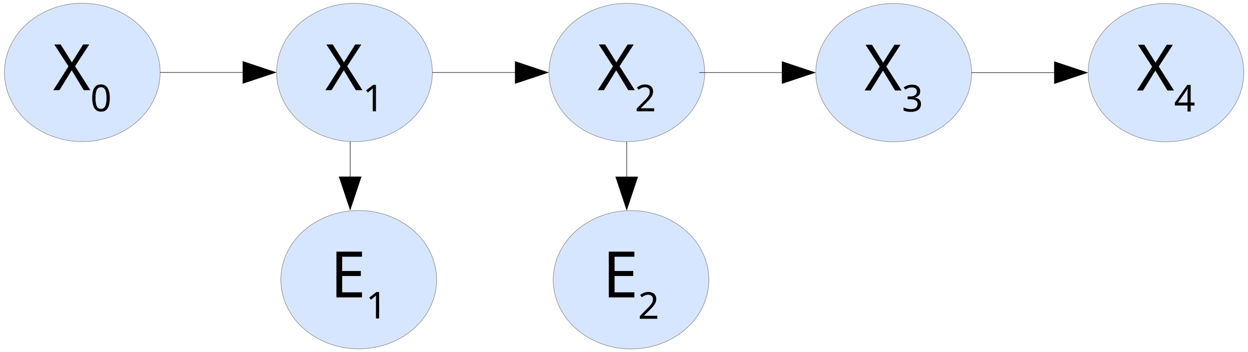
Filtering Example

Numbers?
→ Exercise!



Prediction

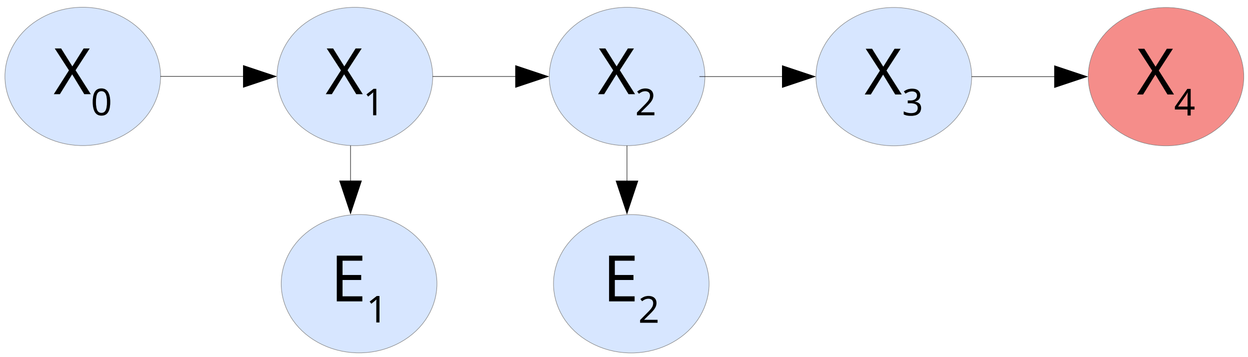
- Query: $\boldsymbol{P(}X_{t+K} \mid e_{1:t})$?



Prediction

$$\mathbf{b}_{t+1}(X_{t+1}) = \alpha\, \mathbf{P}(e_{t+1} | X_{t+1})\, \Sigma_x\, \mathbf{P}(X_{t+1} | x_t)\, \mathbf{b}_t(x_t)$$

- Query: $\mathbf{P}(X_{t+K} \mid e_{1:t})$?



use normal filtering

use only prediction

Prediction… limits?

- How well can we predict “Rain” 10 steps into the future…?

Prediction… limits?

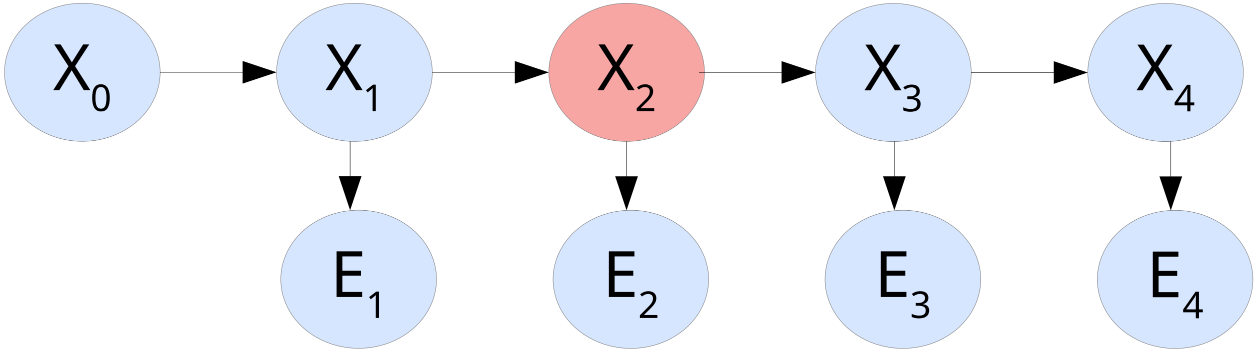
- How well can we predict “Rain” 10 steps into the future…?

- Belief will converge to <0.5, 0.5> quite fast…
  - called the **stationary distribution**
  - the more stochastic the process, the shorter the **mixing time**
  - predicting beyond a fraction of the mixing time will not work

TU Delft

‘Smoothing’

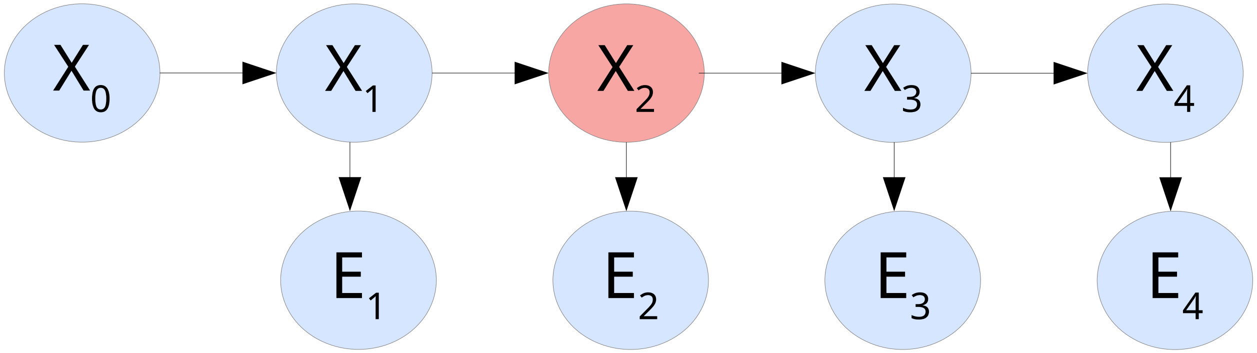
- Query: $\boldsymbol{P}(X_t \mid \boldsymbol{e}_{1:t+K})$



  - more information available than with filtering → less variance in answer → `smoother result’

- Now: also need to take into account information (passed back) from the future... **forward-backward algorithm**

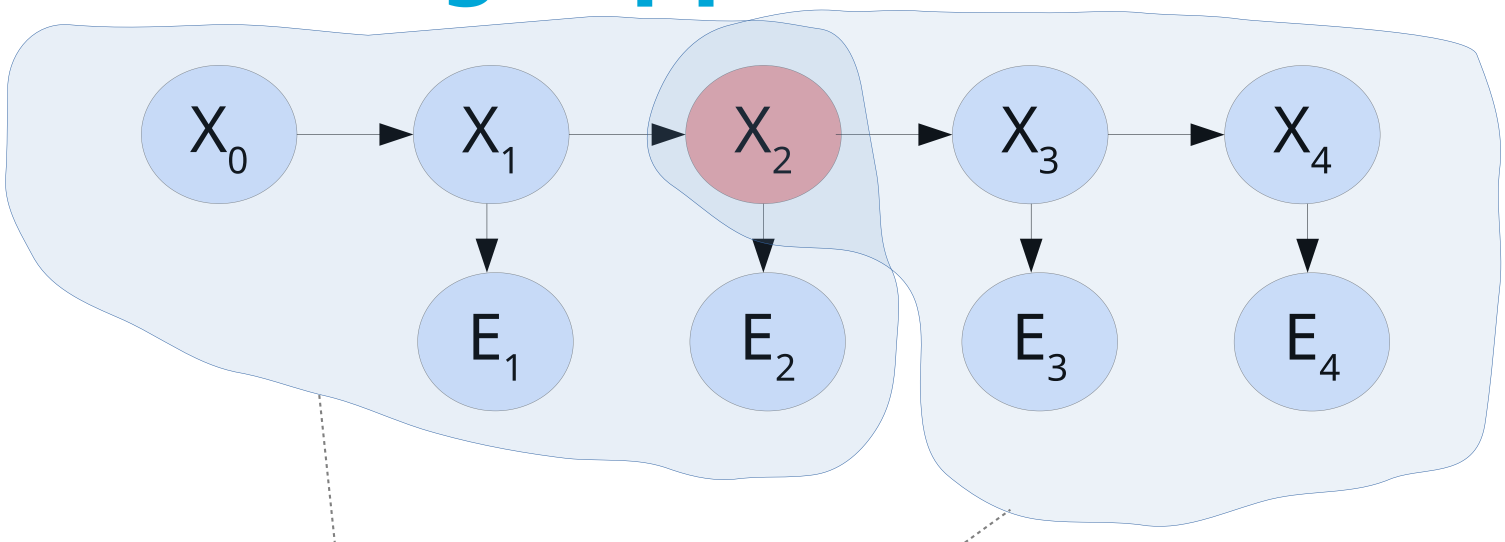
Smoothing: Approach



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

?

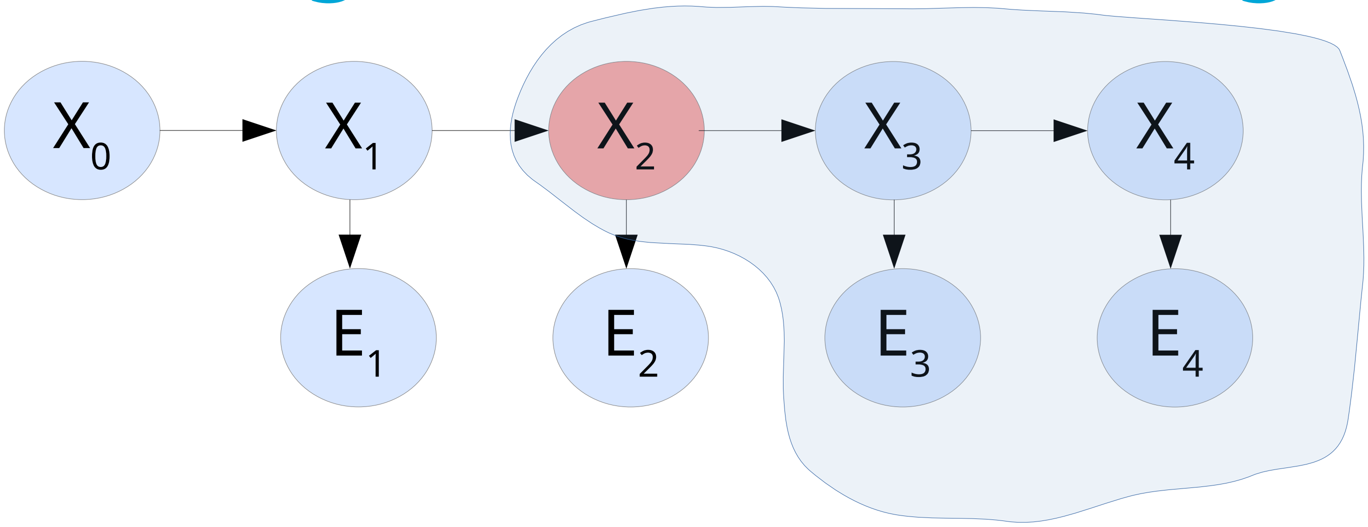
Smoothing: Approach



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

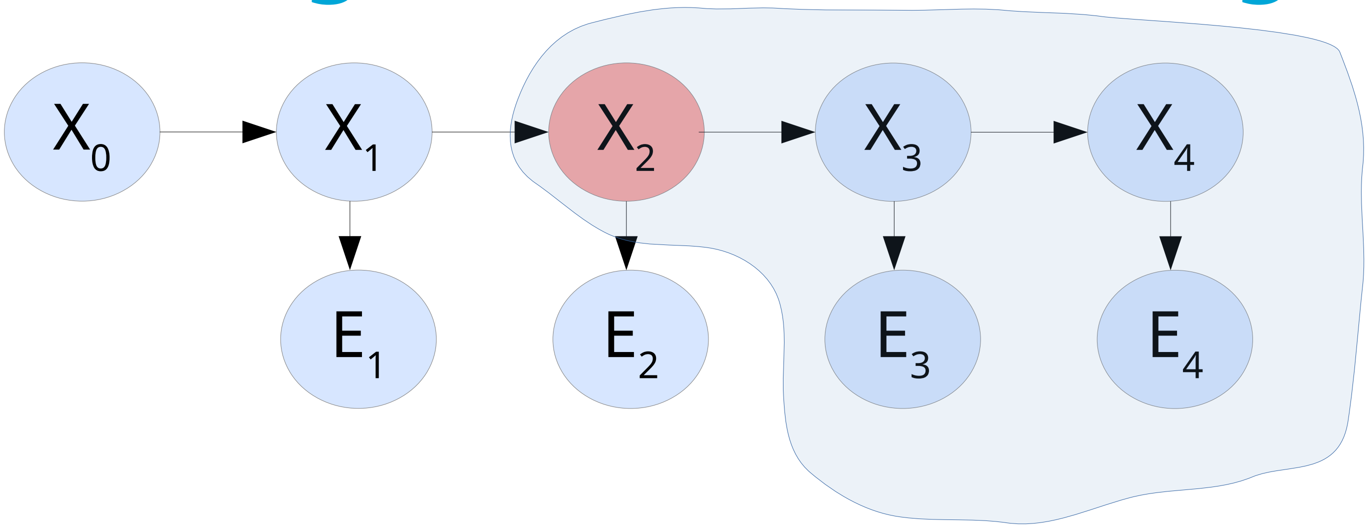
product of **forward** and **backward** messages

Smoothing: Backward messages



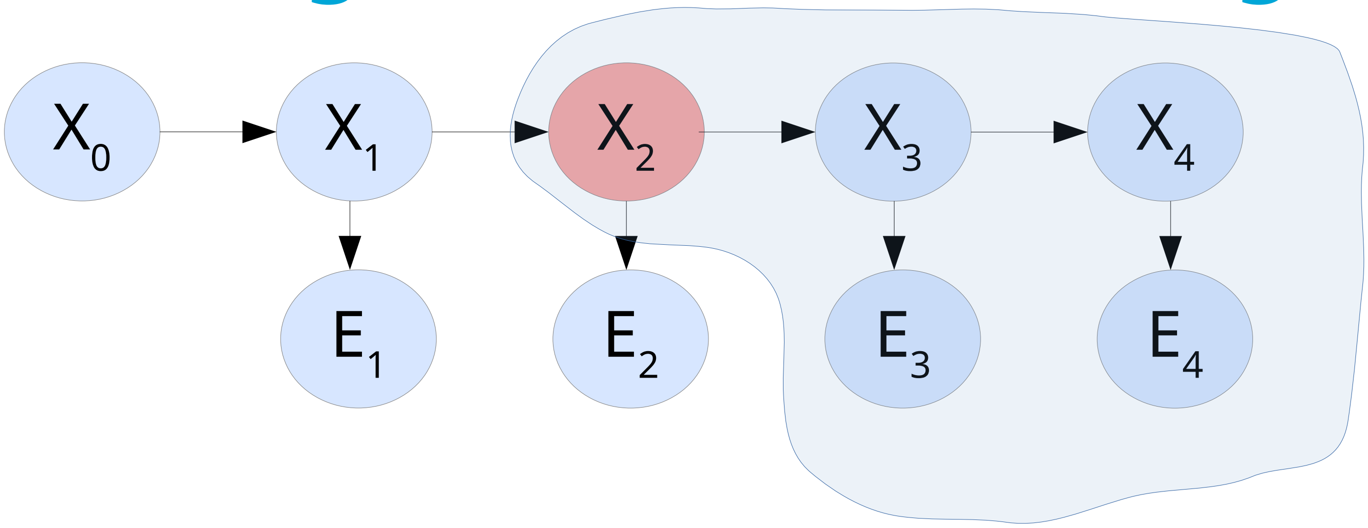
$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \Sigma_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$

Smoothing: Backward messages



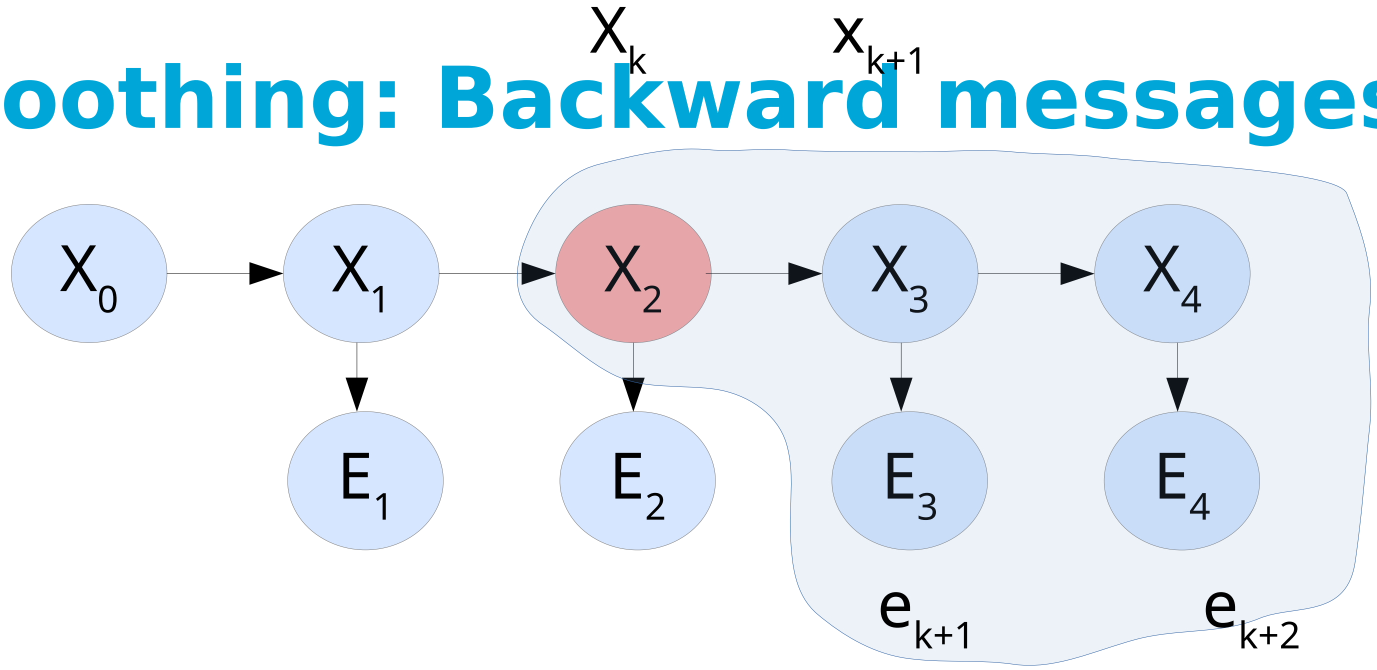
$$
\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \Sigma_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \Sigma_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)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

Smoothing: Backward messages



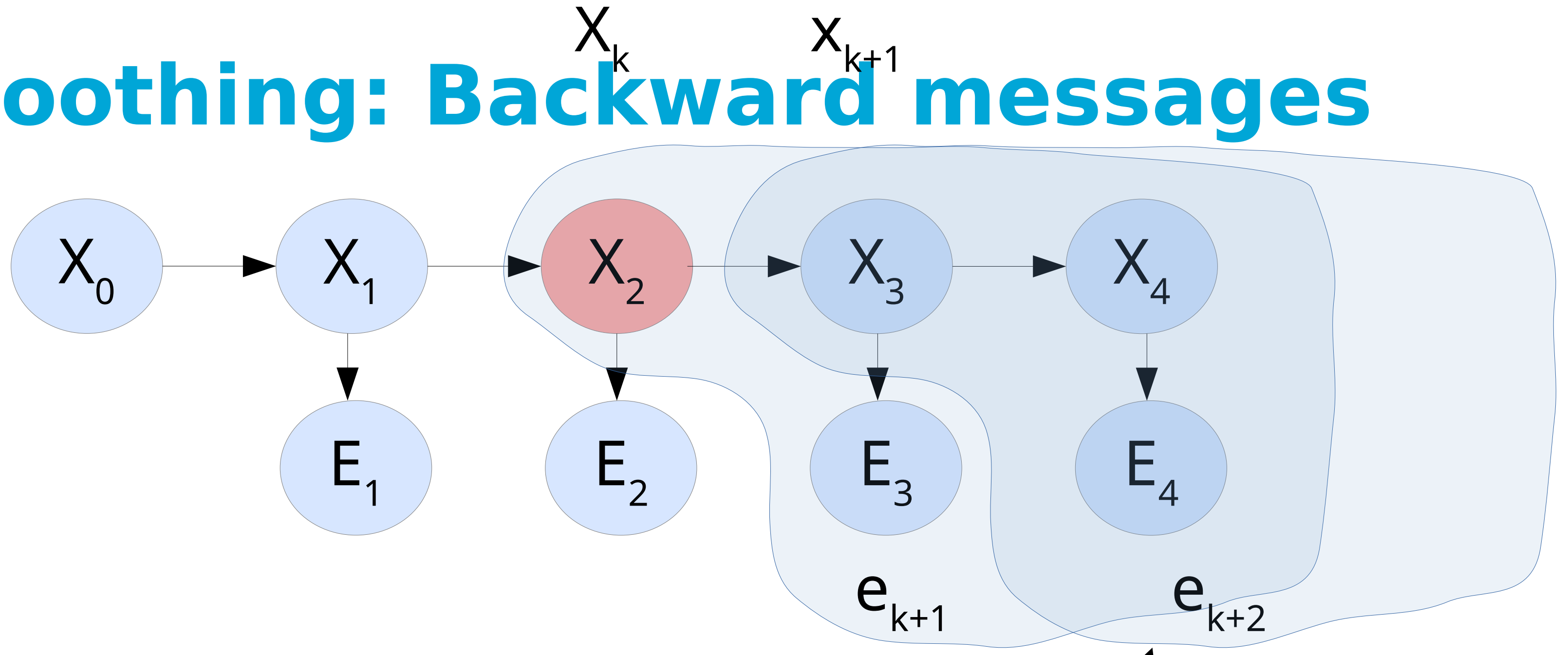
$$
\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \Sigma_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \Sigma_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)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

Smoothing: Backward messages



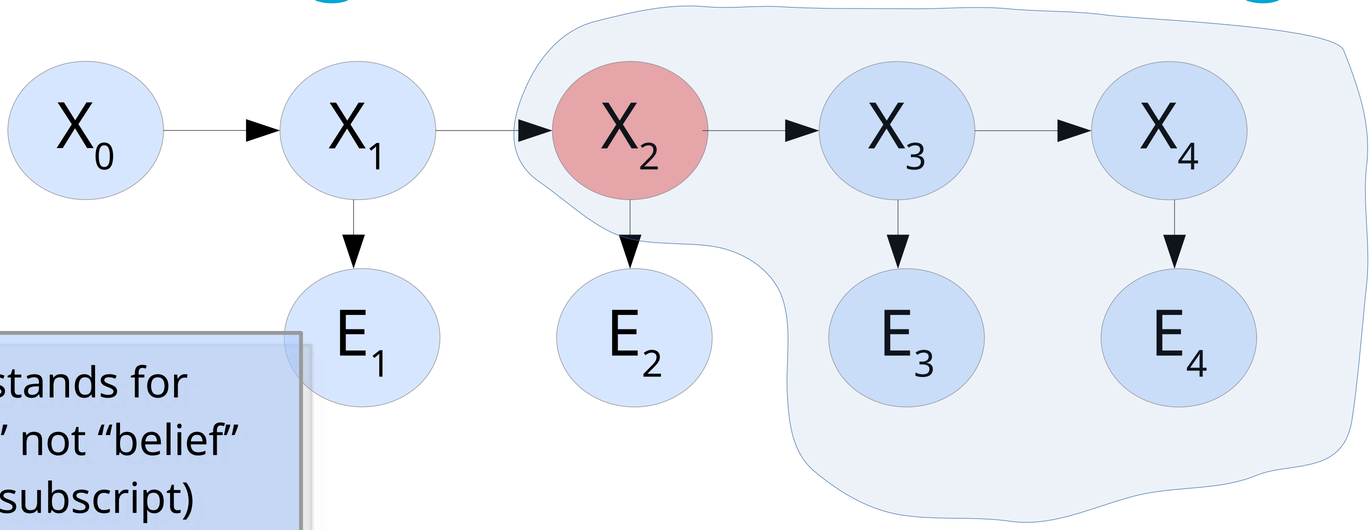
$$
\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \Sigma_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \Sigma_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)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

Smoothing: Backward messages



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

Smoothing: Backward messages



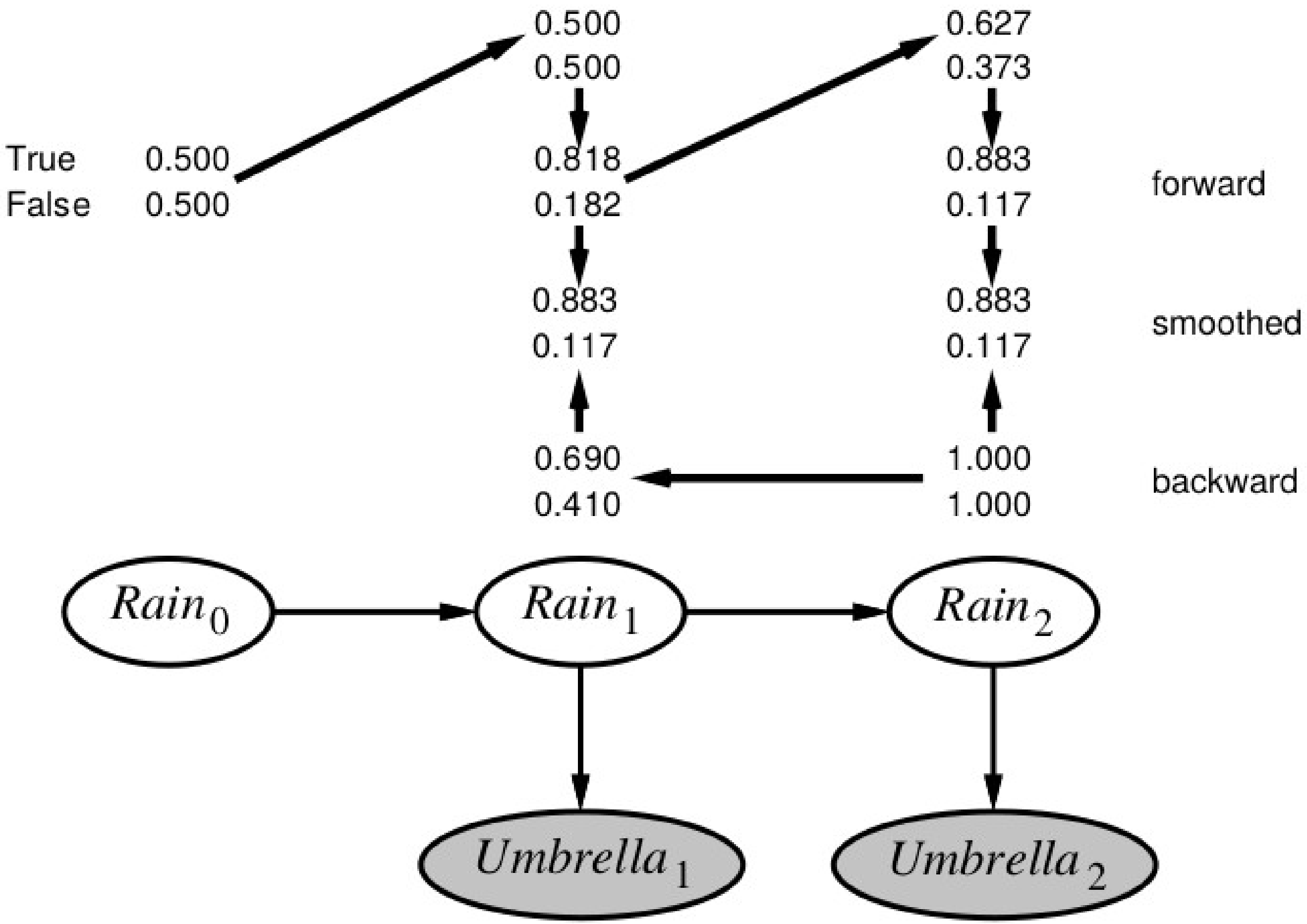
here **b** stands for "backward" not "belief" (look at subscript)

$$\mathbf{b}_{k+1:t}(X_k) = \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$

$\mathbf{b}_{k+2:t}(X_{k+1})$

initialize: $\mathbf{b}_{t+1:t} = \mathbf{1}$ (i.e., vector)

Forward-Backward Illustrated



Smoothing vs most-likely sequences

- Smoothing can compute
  $\{P(X_0 | E_{1:t}),\ P(X_1 | E_{1:t}), \ldots,\ P(X_t | E_{1:t})\}$

- But does not give most likely sequence!
  $\max_{(x0,x1,\ldots,xt)} P(x_0,x_1,\ldots,x_t | E_{1:t})$
  $!=$
  $\{\max_{x0} P(x_0 | E_{1:t}), \max_{x1} P(x_1 | E_{1:t}), \ldots, \max_{xt} P(x_t | E_{1:t})\}$

- (Need a different algo: Viterby – see book)

Dynamic Bayesian Networks

Complex worlds

- E.g., the world has many aspects

BOUTILIER, DEAN, & HANKS

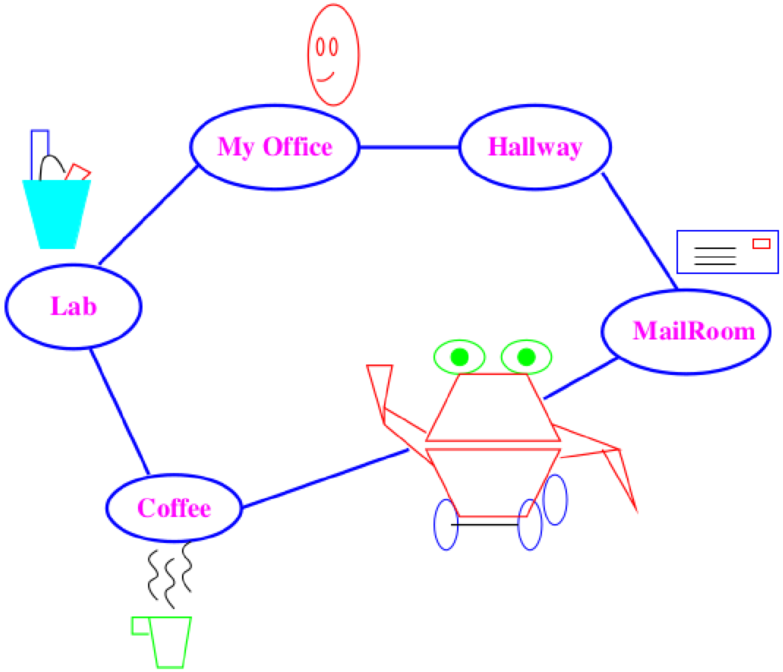


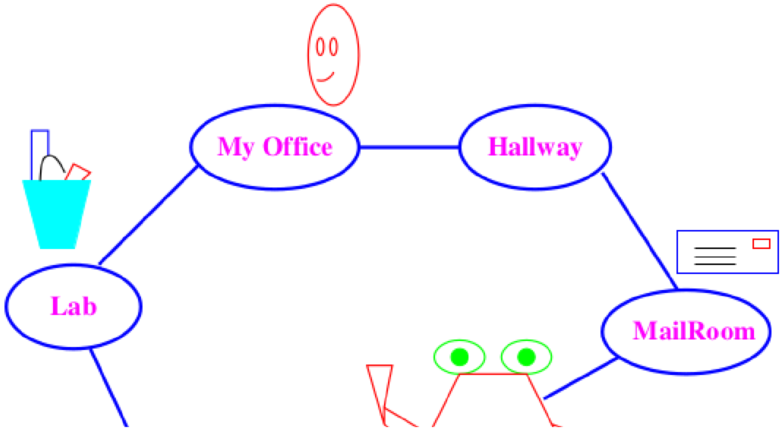
Figure 1: A decision-theoretic planning problem

Boutilier, Craig, Thomas Dean, and Steve Hanks. "Decision-theoretic planning: Structural assumptions and computational leverage." Journal of Artificial Intelligence Research 11 (1999): 1-94.

Complex worlds

Boutilier, Dean, & Hanks

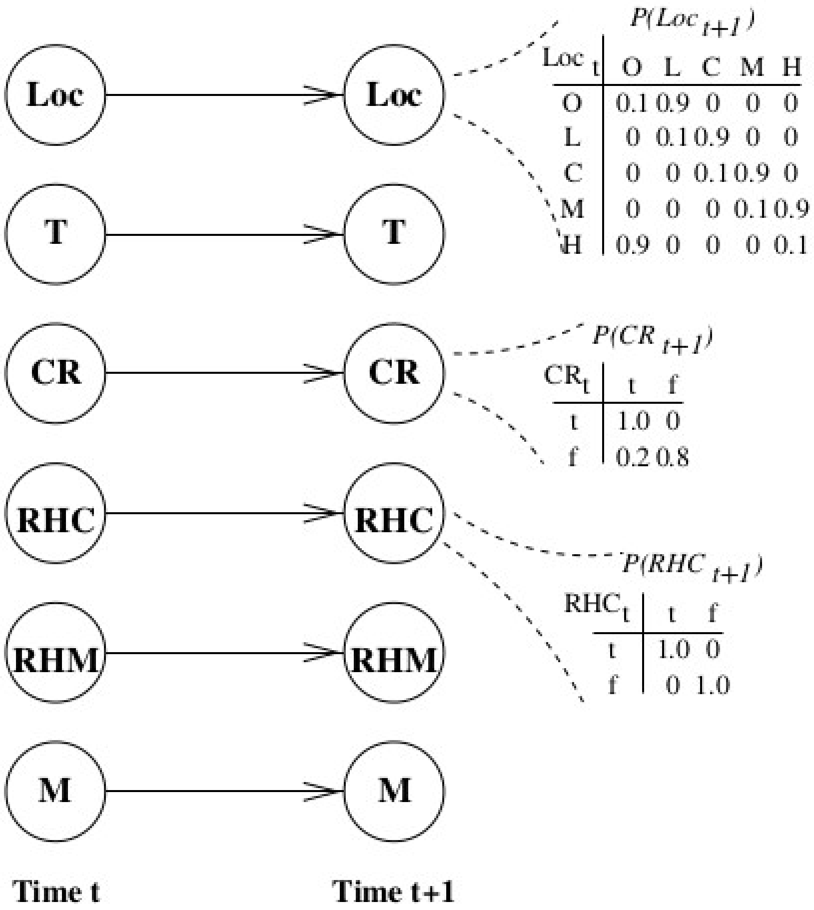
- E.g., the world has many aspects



Features	Denoted	Description
Location	$Loc(M)$, etc.	Location of robot. Five possible locations: mailroom (M), coffee room (C), user's office (O), hallway (H), laboratory (L)
Tidiness	$T(0)$, etc.	Degree of lab tidiness. Five possible values: from 0 (messiest) to 4 (tidiest)
Mail present	$M, \overline{M}$	Is there mail is user's mail box? True (M) or False ($\overline{M}$)
Robot has mail	$RHM, \overline{RHM}$	Does the robot have mail in its possession?
Coffee request	$CR, \overline{CR}$	Is there an outstanding (unfulfilled) request for coffee by the user?
Robot has coffee	$RHC, \overline{RHC}$	Does the robot have coffee in its possession?

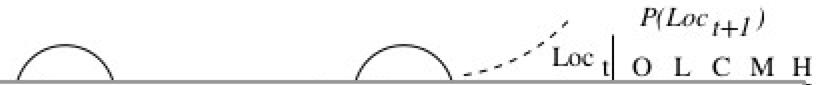
Again: compact representations

- E.g.
  “move counter-clockwise”

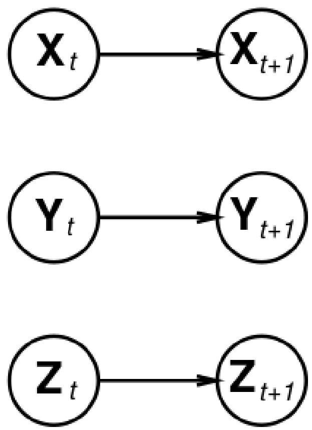


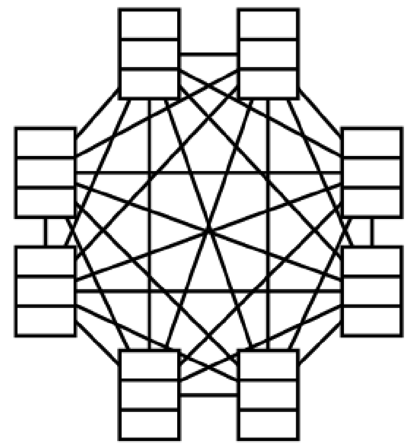
TU Delft

Again: compact representations



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



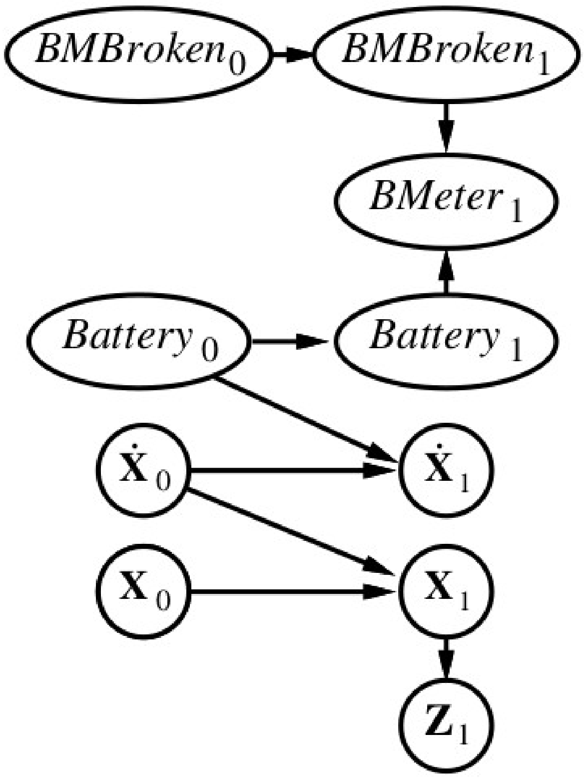


Sparse dependencies $\Rightarrow$ exponentially fewer parameters;

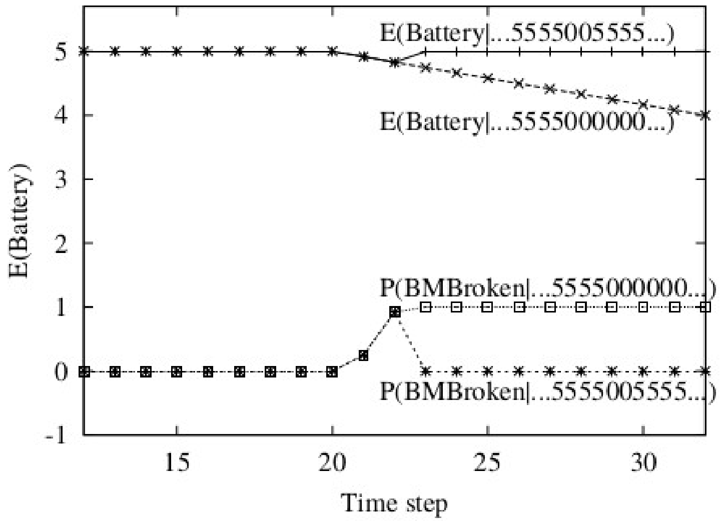
e.g., 20 state variables, three parents each

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

Example, dealing with failing sensors

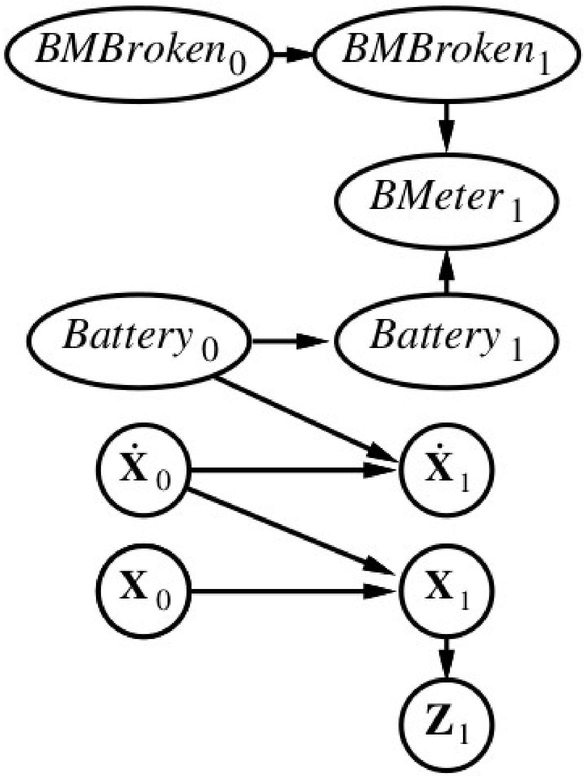


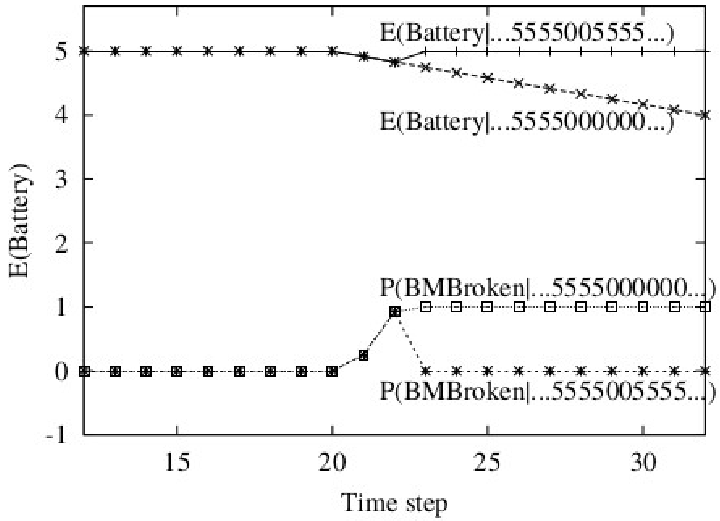
- Can explicitly represent prob. of sensors failing



Example, dealing with failing sensors

- battery meter was '5' for 20 steps
- complete discharge unlikely
  (according to transition model)
- meter can have fluke
- meter might be broken





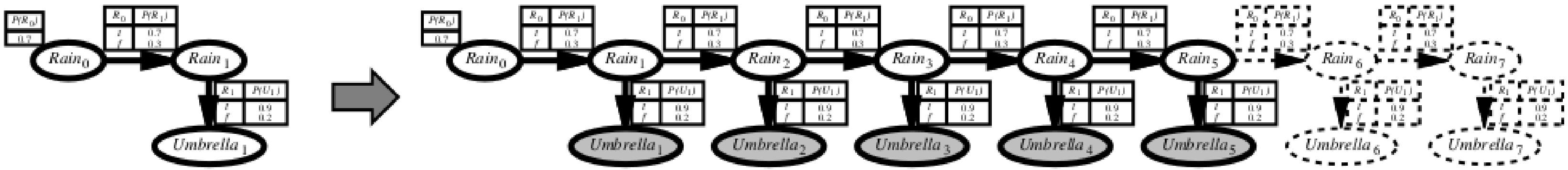
Inference in DBNs…

Suggestions?

Difficulties?

Inference in DBNs...

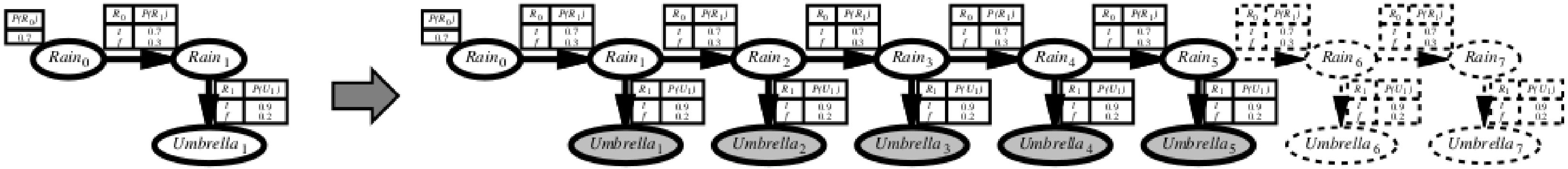
Naive method: unroll the network and run any exact algorithm



Problem: inference cost for each update grows with t

Inference in DBNs…

Naive method: unroll the network and run any exact algorithm



Problem: inference cost for each update grows with t

- But filtering can be done recursively…

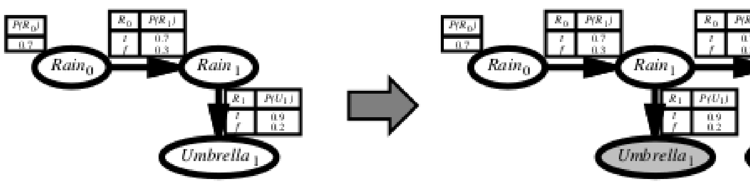
$$\mathbf{b}_{t+1}(X_{t+1}) = \alpha\, \mathbf{P}(e_{t+1} | X_{t+1})\, \Sigma_x\, \mathbf{P}(X_{t+1} | x_t)\, \mathbf{b}_t(x_t)$$

… so perhaps can run variable elimination per time step?

TU Delft

Inference in DBNs

Naive method: unroll the network



Problem: inference cost for each u

Yes... but...

- for a DBN, the factors that you will construct will be huge.
- will include all variables that have parents in previous stage...

→ **approximate inference**

- But filtering can be done recursively...

$$\mathbf{b}_{t+1}(X_{t+1}) = \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1})\, \Sigma_x\, \mathbf{P}(X_{t+1} \mid x_t)\, \mathbf{b}_t(x_t)$$

... so perhaps can run variable elimination per time step?

Ns

“entanglement”

→ try this!

- draw a simple DBN with 3 variables.
- try and compute b’ assuming that b is completely factored:

$$\mathbf{b}(X_1,X_2,X_3)=b(X_1)b(X_2)b(X_3)$$

Yes… but…

- for a DBN, the factors that you will construct will be huge.
- will include all variables that have parents in previous stage…

→ approximate inference

recursively…

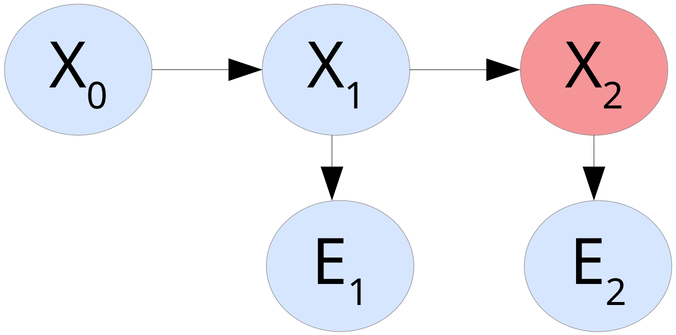
$$\boldsymbol{b}_{t+1}(X_{t+1}) = \alpha\, \boldsymbol{P}(e_{t+1} | X_{t+1})\, \Sigma_x\, \boldsymbol{P}(X_{t+1} | x_t)\, \boldsymbol{b}_t(x_t)$$

… so perhaps can run variable elimination per time step?

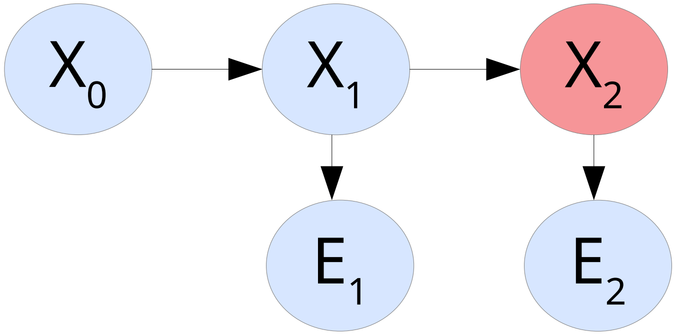
Particle Filters

Approximate inference for DBNs

- Why not Likelihood Weighting?
- For $\mathbf{P}(X_2 \mid \mathbf{e}_{1:2})$ this would be:
  - For i=1:num_samples
    - sample states x_0, x_1, and x_2 (from transition model)
    - form data point $(x_0,x_1,x_2,\mathbf{e}_{1:2})$
    - compute 'weight' w
    - set w[x_2] += w
  - renormalize weights
  - $P(x_2 \mid \mathbf{e}_{1:2}) = W[x_2]$

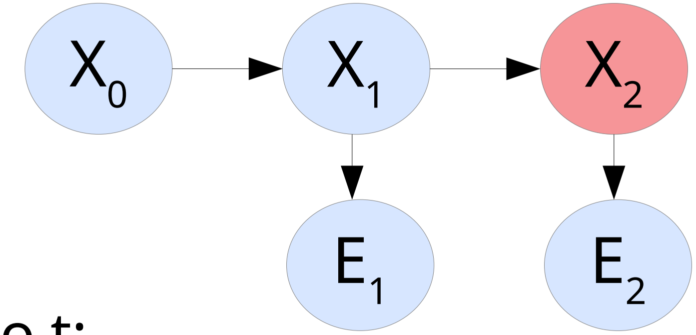


LW… Problems for DBNs



- Two main problems….

LW... Problems for DBNs

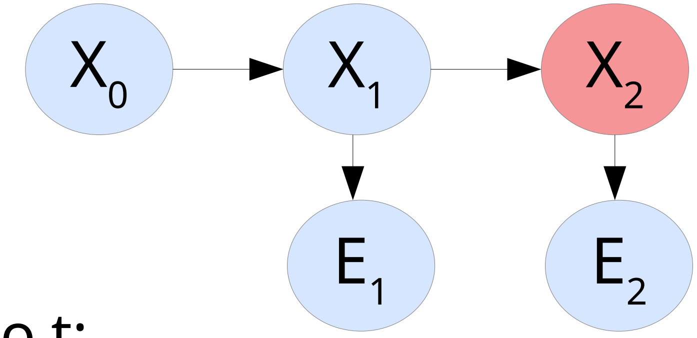


- Two main problems....

1) running each sample from step 1 to t:
time needed grows over time...

2)

LW… Problems for DBNs



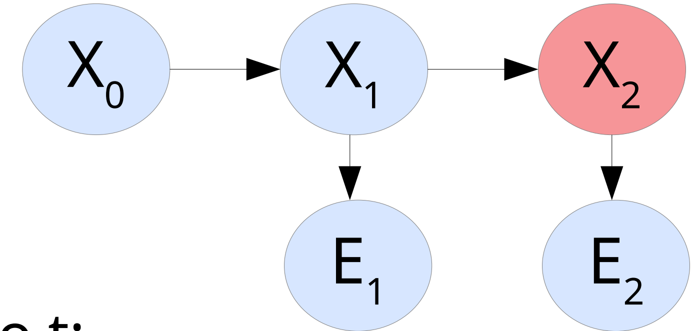
- Two main problems….

1) running each sample from step 1 to t:
   time needed grows over time…

2) states are sampled **independently** of the evidence
   - most samples are completely wrong
     → very few data points data points (x_0,x_1,x_2,**e**) will be likely
     → get all the (renormalized) weight

TU Delft

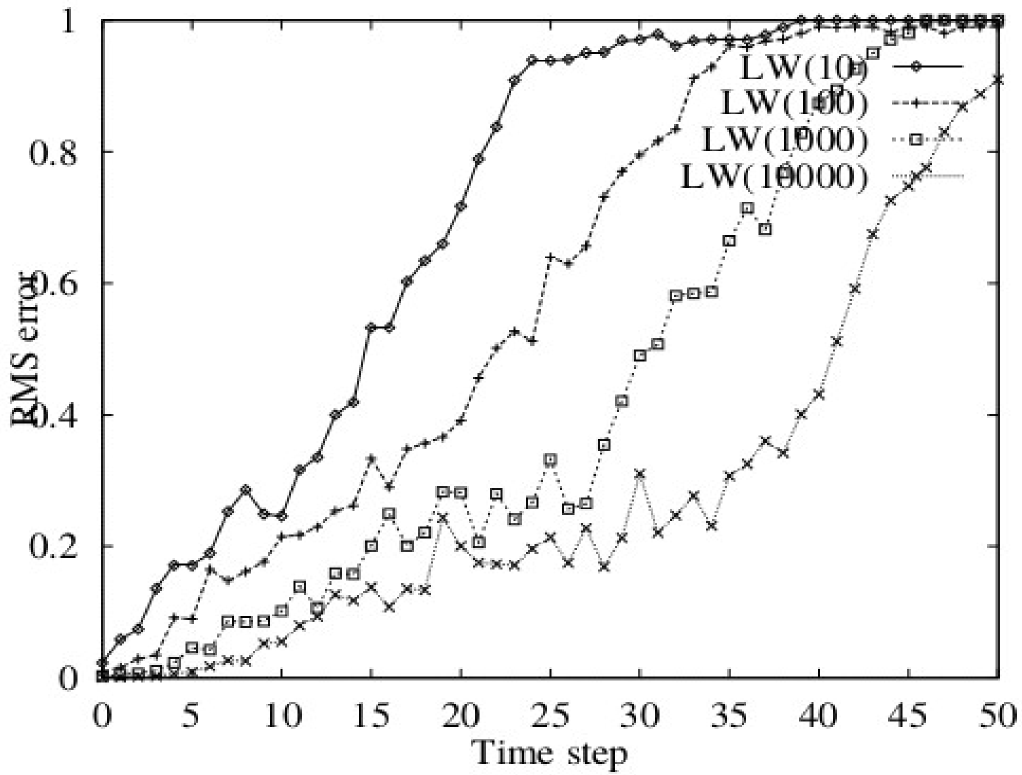
LW… Problems for DBNs



o t:

/ of the evidence

x_0,x_1,x_2,**e**) will be likely



Fixing these: Particle Filtering

run all N samples at the same time
→ 'particles' themselves represent belief

- Two main problems….

1) running each sample from step 1 to t:
   time needed grows over time…

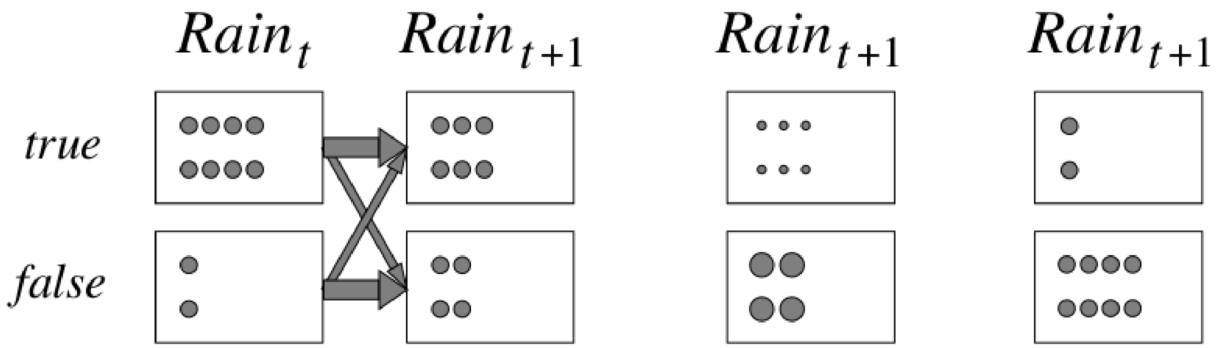
2) states are sampled **independently** of the evidence
   - most samples are completely wrong
     → very few data points data points (x_0,x_1,x_2,**e**) will be likely
     → get all the (renormalized) weight

resampling
→ focus attention on parts of state space
with large prob. under the evidence

Particle Filtering: Intuition

Basic idea: ensure that the population of samples ("particles") tracks the high-likelihood regions of the state-space

Replicate particles proportional to likelihood for $\mathbf{e}_t$



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

TUDelft

Particle Filtering: Updates

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

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

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:

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

Particle Filtering: Updates

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

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

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

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

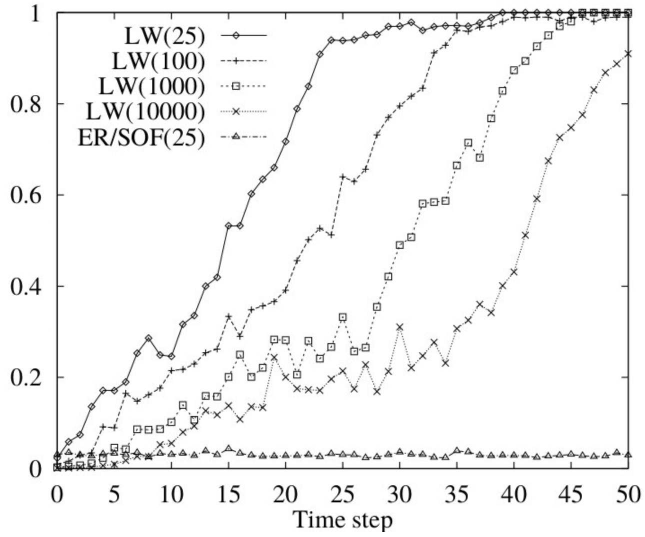
Resample to obtain populations proportional to W:

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

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

Particle Filtering in Practice

- In practice this works very well



- Many video explanations -E.g.,
  - https://www.youtube.com/watch?v=sz7cJuMgKFg
  - https://youtu.be/eAqAFSrTGGY?t=1117

- Many real world applications
  - http://stanford.edu/~cpiech/cs221/apps/driverlessCar.html

Stanley

Reasoning over Time: Summary

- Agents need to reason over time: time-slice based Bayesian networks
  - Hidden Markov Models – HMMs – 1 discrete state
  - Dynamic Bayesian networks – DBNs

- Inference over time
  - Filtering, prediction, smoothing
  - Scaling to large DBNs: intractable
    → approximate inference: particle filter

TUDelft