

Introduction to Artificial Intelligence

Lecture 7: Reasoning over time

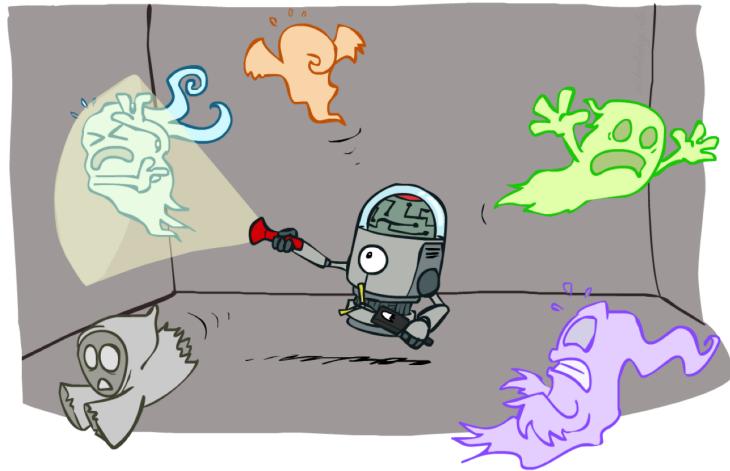
Prof. Gilles Louppe
g.louppe@uliege.be

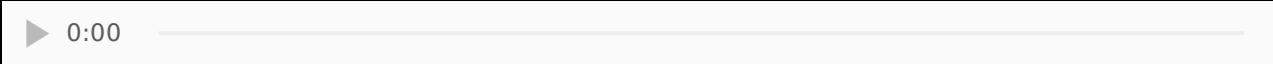


Today

Maintain a **belief state** about the world, and update it as time passes and evidence is collected.

- Markov models
 - Markov processes
 - Inference tasks
 - Prediction
 - Filtering
 - Smoothing
 - Most likely explanation
 - Hidden Markov models
- Filters
 - Kalman filter
 - Particle filter





Pacman revenge: How to make good use of the sonar readings?

Markov models

Reasoning over time

Often, we want to **reason about a sequence** of observations.

- Speech recognition
- Robot localization
- User attention
- Medical monitoring.

For this reason, we need to introduce **time** (or space) in our model.

Modelling time

Consider the world as a **discrete** series of **time slices**, each of which contains a set of random variables:

- \mathbf{X}_t denotes the set of **unobservable** state variables at time t .
- \mathbf{E}_t denotes the set of **observable** evidence variables at time t .

We specify:

- a prior $P(\mathbf{X}_0)$ that defines our initial belief state over hidden state variables.
- a **transition model** $P(\mathbf{X}_t | \mathbf{X}_{0:t-1})$ (for $t > 0$) that defines the probability distribution over the latest state variables, given the previous (unobserved) values.
- a **sensor model** $P(\mathbf{E}_t | \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1})$ (for $t > 0$) that defines the probability distribution over the latest evidence variables, given all previous (observed and unobserved) values.

Markov processes

Markov assumption

- The current state of the world depends only on its immediate previous state(s).
- i.e., \mathbf{X}_t depends on only a bounded subset of $\mathbf{X}_{0:t-1}$.

Random processes that satisfy this assumption are called **Markov processes**.

First-order Markov processes

- Markov processes such that

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

- i.e., \mathbf{X}_t and $\mathbf{X}_{0:t-2}$ are conditionally independent given \mathbf{X}_{t-1} .

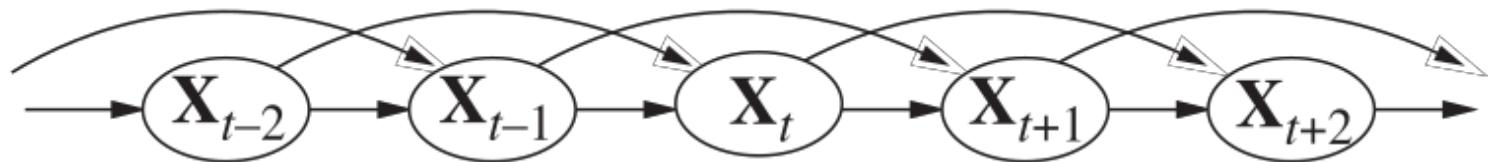


Second-order Markov processes

- Markov processes such that

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

- i.e., \mathbf{X}_t and $\mathbf{X}_{0:t-3}$ are conditionally independent given \mathbf{X}_{t-1} and \mathbf{X}_{t-2} .



Sensor Markov assumption

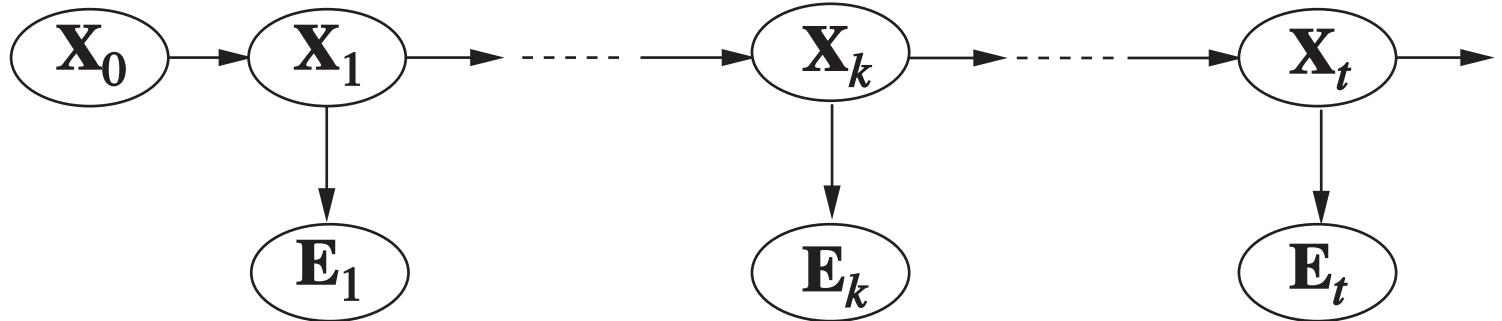
- Additionally, we make a (first-order) **sensor Markov assumption**:

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

Stationarity assumption

- The transition and the sensor models are the same for all t (i.e., the laws of physics do not change with time).

Joint distribution

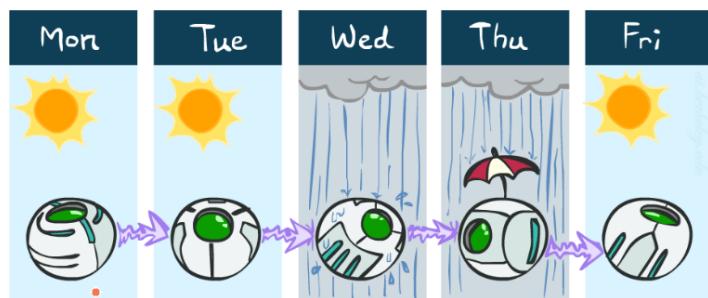
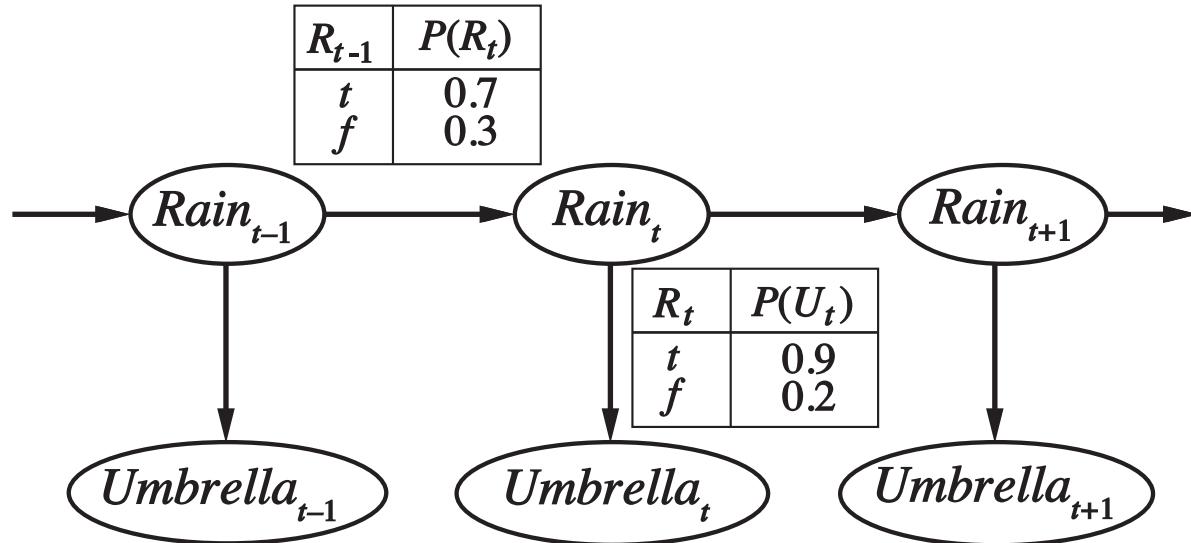


A Markov process can be described as a **growable** Bayesian network, unrolled infinitely through time, with a specified **restricted structure** between time steps.

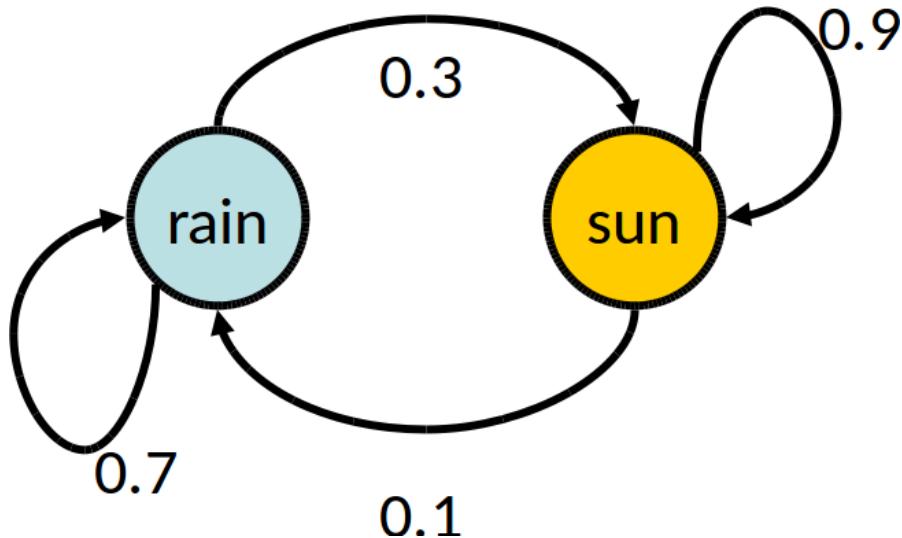
Therefore, the **joint distribution** of all variables up to t in a (first-order) Markov process is

$$P(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = P(\mathbf{X}_0) \prod_{i=1}^t P(\mathbf{X}_i | \mathbf{X}_{i-1}) P(\mathbf{E}_i | \mathbf{X}_i).$$

Example: Will you take your umbrella today?



- $P(Umbrella_t | Rain_t)?$
- $P(Rain_t | Umbrella_{0:t-1})?$
- $P(Rain_{t+2} | Rain_t)?$

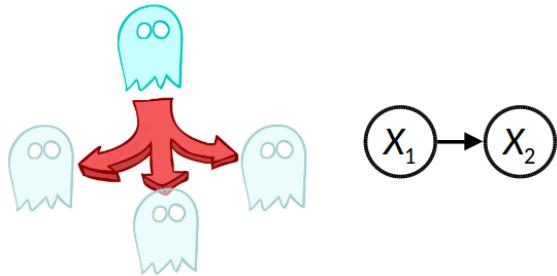


The transition model $P(\text{Rain}_t | \text{Rain}_{t-1})$ can equivalently be represented by a state transition diagram.

Inference tasks

- **Prediction:** $P(\mathbf{X}_{t+k} | \mathbf{e}_{1:t})$ for $k > 0$
 - Computing the posterior distribution over future states.
 - Used for evaluation of possible action sequences.
- **Filtering:** $P(\mathbf{X}_t | \mathbf{e}_{1:t})$
 - Filtering is what a rational agent does to keep track of the current hidden state \mathbf{X}_t , its **belief state**, so that rational decisions can be made.
- **Smoothing:** $P(\mathbf{X}_k | \mathbf{e}_{1:t})$ for $0 \leq k < t$
 - Computing the posterior distribution over past states.
 - Used for building better estimates, since it incorporates more evidence.
 - Essential for learning.
- **Most likely explanation:** $\arg \max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t} | \mathbf{e}_{1:t})$
 - Decoding with a noisy channel, speech recognition, etc.

Base cases



$$\begin{aligned} P(\mathbf{X}_2) &= \sum_{\mathbf{x}_1} P(\mathbf{X}_2, \mathbf{x}_1) \\ &= \sum_{\mathbf{x}_1} P(\mathbf{x}_1)P(\mathbf{X}_2|\mathbf{x}_1) \end{aligned}$$

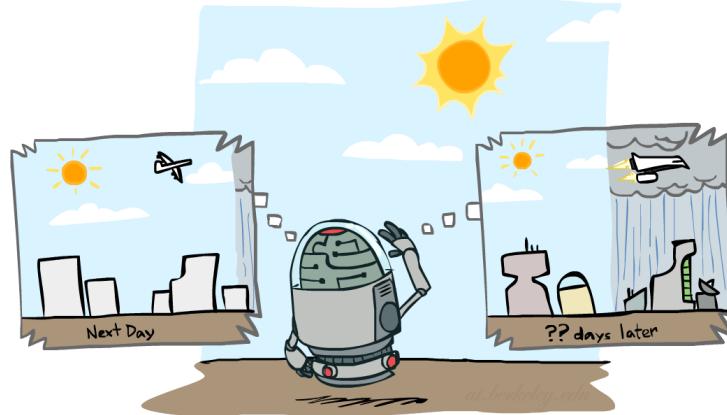
(Predict) Push $P(\mathbf{X}_1)$ forward through the transition model.



$$\begin{aligned} P(\mathbf{X}_1|\mathbf{e}_1) &= \frac{P(\mathbf{X}_1, \mathbf{e}_1)}{P(\mathbf{e}_1)} \\ &\propto P(\mathbf{X}_1, \mathbf{e}_1) \\ &= P(\mathbf{X}_1)P(\mathbf{e}_1|\mathbf{X}_1) \end{aligned}$$

(Update) Update $P(\mathbf{X}_1)$ with the evidence \mathbf{e}_1 , given the sensor model.

Prediction

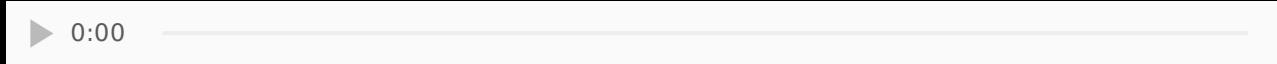


To predict the future $P(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$:

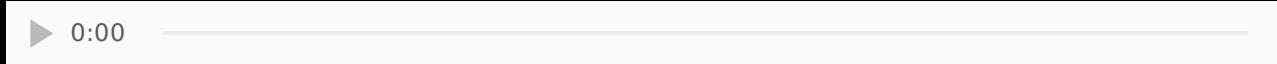
- Push the prior belief state $P(\mathbf{X}_t|\mathbf{e}_{1:t})$ through the transition model:

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

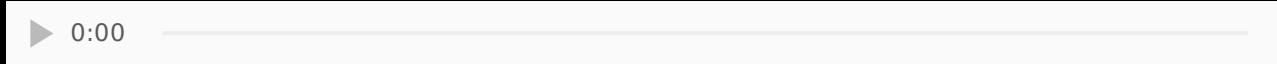
- Repeat up to $t + k$, using $P(\mathbf{X}_{t+k-1}|\mathbf{e}_{1:t})$ to compute $P(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$.



Basic dynamics (Ghostbusters)



Circular dynamics (Ghostbusters)



Whirlpool dynamics (Ghostbusters)

<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	1.00	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

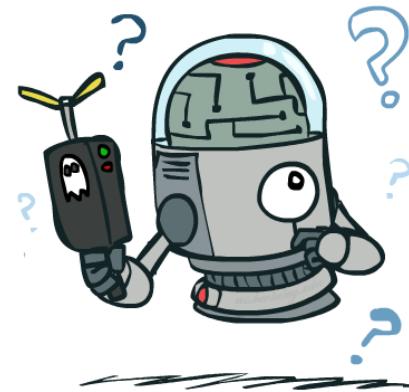
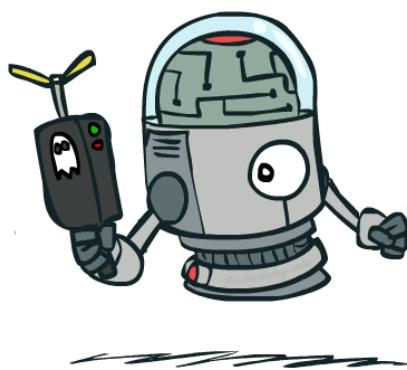
$T = 1$

<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

$T = 2$

0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

$T = 5$



As time passes, uncertainty "accumulates" if we do not accumulate new evidence.

Stationary distributions

What if $t \rightarrow \infty$?

- For most chains, the influence of the initial distribution gets lesser and lesser over time.
- Eventually, the distribution converges to a fixed point, called the **stationary distribution**.
- This distribution is such that

$$P(\mathbf{X}_\infty) = P(\mathbf{X}_{\infty+1}) = \sum_{\mathbf{x}_\infty} P(\mathbf{X}_{\infty+1} | \mathbf{x}_\infty) P(\mathbf{x}_\infty)$$

Example

$$\begin{aligned} P(\mathbf{X}_\infty = \text{sun}) &= P(\mathbf{X}_{\infty+1} = \text{sun}) \\ &= P(\mathbf{X}_{\infty+1} = \text{sun} | \mathbf{X}_\infty = \text{sun})P(\mathbf{X}_\infty = \text{sun}) \\ &\quad + P(\mathbf{X}_{\infty+1} = \text{sun} | \mathbf{X}_\infty = \text{rain})P(\mathbf{X}_\infty = \text{rain}) \\ &= 0.9P(\mathbf{X}_\infty = \text{sun}) + 0.3P(\mathbf{X}_\infty = \text{rain}) \end{aligned}$$

Therefore, $P(\mathbf{X}_\infty = \text{sun}) = 3P(\mathbf{X}_\infty = \text{rain})$.

Which implies that $P(\mathbf{X}_\infty = \text{sun}) = \frac{3}{4}$ and $P(\mathbf{X}_\infty = \text{rain}) = \frac{1}{4}$.

Filtering

0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

<0.01	<0.01	<0.01	<0.01	0.02	<0.01
<0.01	<0.01	<0.01	0.83	0.02	<0.01
<0.01	<0.01	0.11	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

After observation

What if we collect new observations? Beliefs get reweighted, and uncertainty "decreases":

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

Bayes filter

An agent maintains a **belief state** estimate $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ and updates it as new evidences \mathbf{e}_{t+1} are collected.

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

- (Predict step): Project the current belief state forward from t to $t + 1$ through the transition model.
- (Update step): Update this new state using the evidence \mathbf{e}_{t+1} .

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

where

- the normalization constant

$$\alpha = \frac{1}{P(\mathbf{e}_{t+1} | \mathbf{e}_{1:t})} = 1 / \sum_{\mathbf{x}_{t+1}} P(\mathbf{e}_{t+1} | \mathbf{x}_{t+1}) P(\mathbf{x}_{t+1} | \mathbf{e}_{1:t})$$

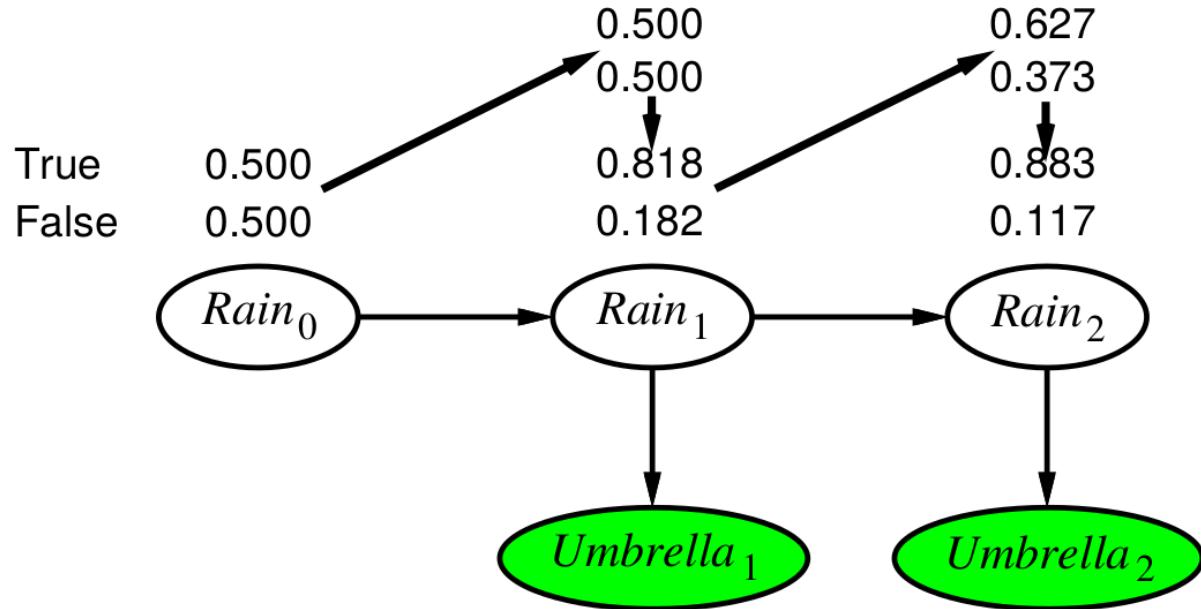
is used to make probabilities sum to 1;

- in the last expression, the first and second terms are given by the model while the third is obtained recursively.

We can think of $P(\mathbf{X}_t | \mathbf{e}_{1:t})$ as a message $\mathbf{f}_{1:t}$ that is propagated forward along the sequence, modified by each transition and updated by each new observation.

- Thus, the process can be implemented as $\mathbf{f}_{1:t+1} = \alpha \text{forward}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$.
- The complexity of a forward update is constant (in time and space) with t .

Example

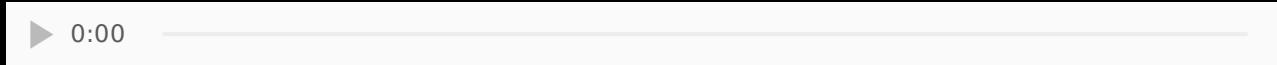


$$R_{t-1} \quad P(R_t)$$

$$\begin{array}{ll} true & 0.7 \\ false & 0.3 \end{array}$$

$$R_t \quad P(U_t)$$

$$\begin{array}{ll} true & 0.9 \\ false & 0.2 \end{array}$$



Ghostbusters with a Bayes filter

Smoothing

We want to compute $P(\mathbf{X}_k | \mathbf{e}_{1:t})$ for $0 \leq k < t$.

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

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

Let the **backward** message $\mathbf{b}_{k+1:t}$ correspond to $P(\mathbf{e}_{k+1:t} | \mathbf{X}_k)$. Then,

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

where \times is a pointwise multiplication of vectors.

This backward message can be computed using backwards recursion:

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

The first and last factors are given by the model. The second factor is obtained recursively. Therefore,

$$\mathbf{b}_{k+1:t} = \text{backward}(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1}).$$

Forward-backward algorithm

function FORWARD-BACKWARD(\mathbf{ev} , $prior$) **returns** a vector of probability distributions

inputs: \mathbf{ev} , a vector of evidence values for steps $1, \dots, t$

prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$

local variables: \mathbf{fv} , a vector of forward messages for steps $0, \dots, t$

\mathbf{b} , a representation of the backward message, initially all 1s

\mathbf{sv} , a vector of smoothed estimates for steps $1, \dots, t$

$\mathbf{fv}[0] \leftarrow prior$

for $i = 1$ **to** t **do**

$\mathbf{fv}[i] \leftarrow \text{FORWARD}(\mathbf{fv}[i - 1], \mathbf{ev}[i])$

for $i = t$ **downto** 1 **do**

$\mathbf{sv}[i] \leftarrow \text{NORMALIZE}(\mathbf{fv}[i] \times \mathbf{b})$

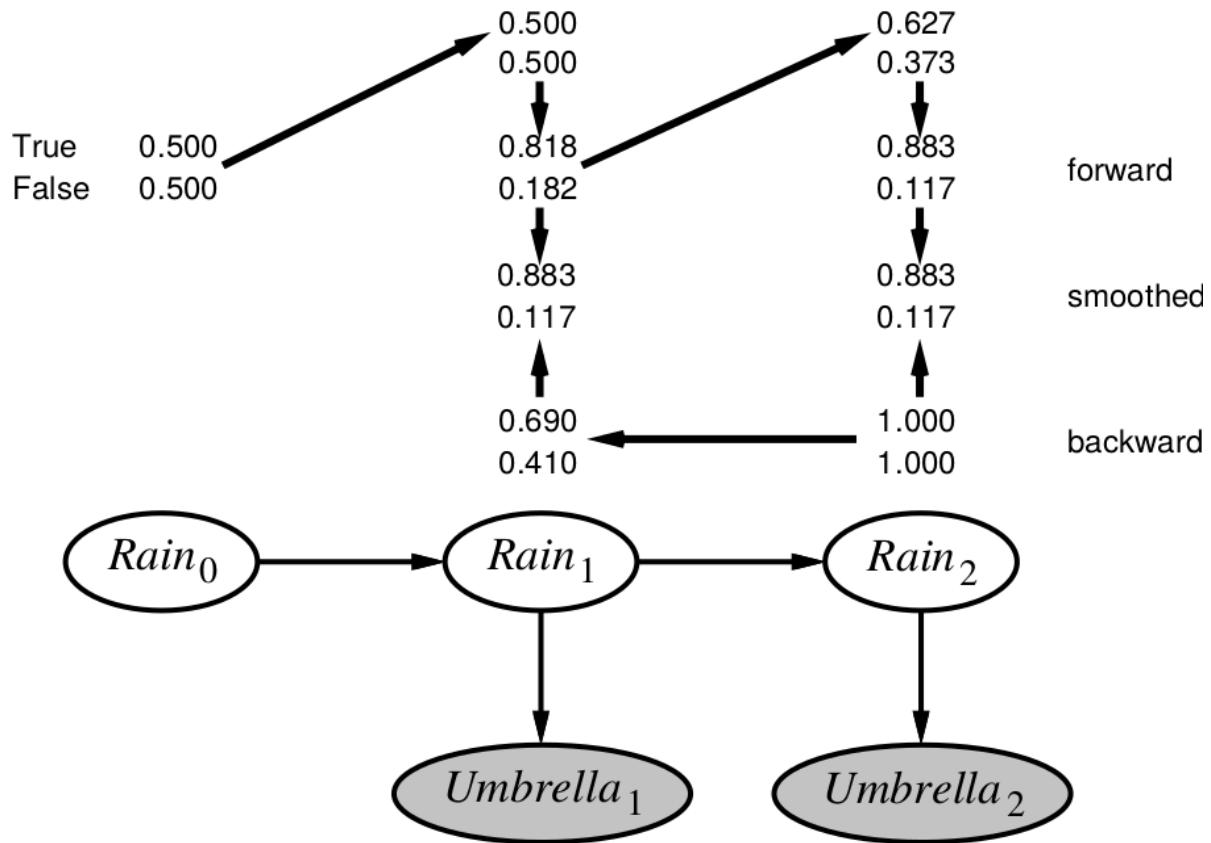
$\mathbf{b} \leftarrow \text{BACKWARD}(\mathbf{b}, \mathbf{ev}[i])$

return \mathbf{sv}

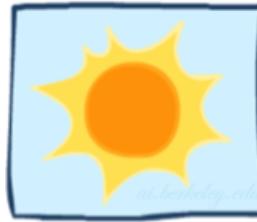
Complexity:

- Smoothing for a particular time step k takes: $O(t)$
- Smoothing a whole sequence (because of caching): $O(t)$

Example



Most likely explanation



Suppose that $[true, true, false, true, true]$ is the umbrella sequence.

What is the weather sequence that is the most likely to explain this?

- Does the absence of umbrella at day 3 means it wasn't raining?
- Or did the director forget to bring it?
- If it didn't rain on day 3, perhaps it didn't rain on day 4 either, but the director brought the umbrella just in case?

Among all 2^5 sequences, is there an (efficient) way to find the most likely one?

- The most likely sequence **is not** the sequence of the most likely states!
- The most likely path to each \mathbf{x}_{t+1} , is the most likely path to **some** \mathbf{x}_t plus one more step. Therefore,

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

- Identical to filtering, except that the forward message $\mathbf{f}_{1:t} = P(\mathbf{X}_t | \mathbf{e}_{1:t})$ is replaced with

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

where $\mathbf{m}_{1:t}(i)$ gives the probability of the most likely path to state i .

- The update has its sum replaced by max, resulting in the **Viterbi algorithm**:

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

Example

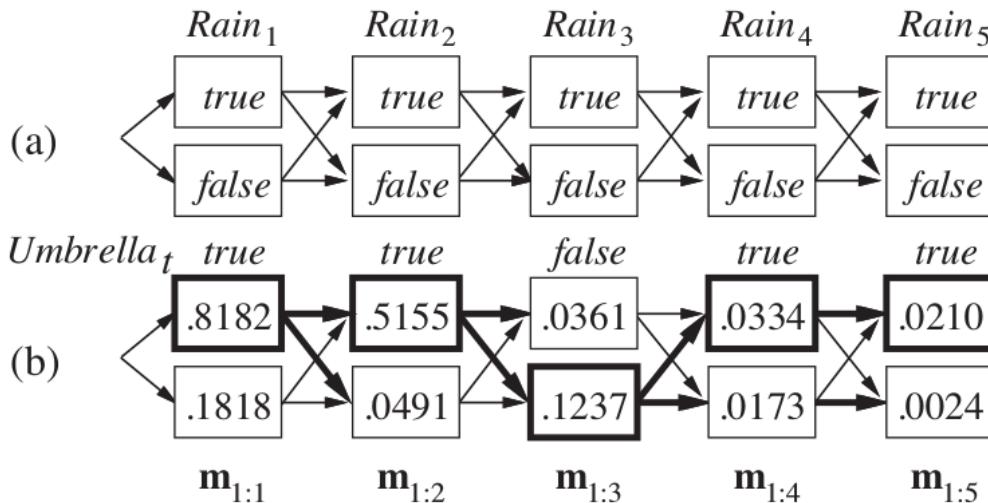


Figure 15.5 (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence [true, true, false, true, true]. For each t , we have shown the values of the message $\mathbf{m}_{1:t}$, which gives the probability of the best sequence reaching each state at time t . Also, for each state, the bold arrow leading into it indicates its best predecessor as measured by the product of the preceding sequence probability and the transition probability. Following the bold arrows back from the most likely state in $\mathbf{m}_{1:5}$ gives the most likely sequence.

Hidden Markov models

So far, we described Markov processes over arbitrary sets of state variables \mathbf{X}_t and evidence variables \mathbf{E}_t .

- A **hidden Markov model** (HMM) is a Markov process in which the state \mathbf{X}_t and the evidence \mathbf{E}_t are both **single discrete** random variables.
 - $\mathbf{X}_t = X_t$, with domain $D_{X_t} = \{1, \dots, S\}$
 - $\mathbf{E}_t = E_t$, with domain $D_{E_t} = \{1, \dots, R\}$
- This restricted structure allows for a reformulation of the forward-backward algorithm in terms of matrix-vector operations.

Note on terminology

Some authors instead divide Markov models into two classes, depending on the observability of the system state:

- Observable system state: Markov chains
- Partially-observable system state: Hidden Markov models.

We follow here the terminology of the textbook.

Simplified matrix algorithms

- The prior $P(X_0)$ becomes a (normalized) column vector $\mathbf{f}_0 \in \mathbb{R}_+^S$.
- The transition model $P(X_t|X_{t-1})$ becomes an $S \times S$ transition matrix \mathbf{T} , such that

$$\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i).$$

- The sensor model $P(E_t|X_t)$ is defined as an $S \times R$ sensor matrix \mathbf{B} , such that

$$\mathbf{B}_{ij} = P(E_t = j | X_t = i).$$

- Let the observation matrix \mathbf{O}_t be a diagonal matrix whose elements corresponds to the column e_t of the sensor matrix \mathbf{B} .
- If we use column vectors to represent forward and backward messages, then we have

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^T \mathbf{f}_{1:t}$$

$$\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t},$$

where $\mathbf{b}_{t:t}$ is an all-one vector of size S .

- Therefore the forward-backward algorithm needs time $O(S^2 t)$ and space $O(St)$.

Example

Suppose that $[true, true, false, true, true]$ is the umbrella sequence.

$$\mathbf{f}_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$\mathbf{T} = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix}$$

$$\mathbf{O}_1 = \mathbf{O}_2 = \mathbf{O}_4 = \mathbf{O}_5 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}$$

$$\mathbf{O}_3 = \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.8 \end{pmatrix}$$

See code/lecture7-forward-backward.ipynb for the execution.

Stationary distribution

The stationary distribution \mathbf{f} of a HMM is a distribution such that

$$\mathbf{f} = \mathbf{T}^T \mathbf{f}.$$

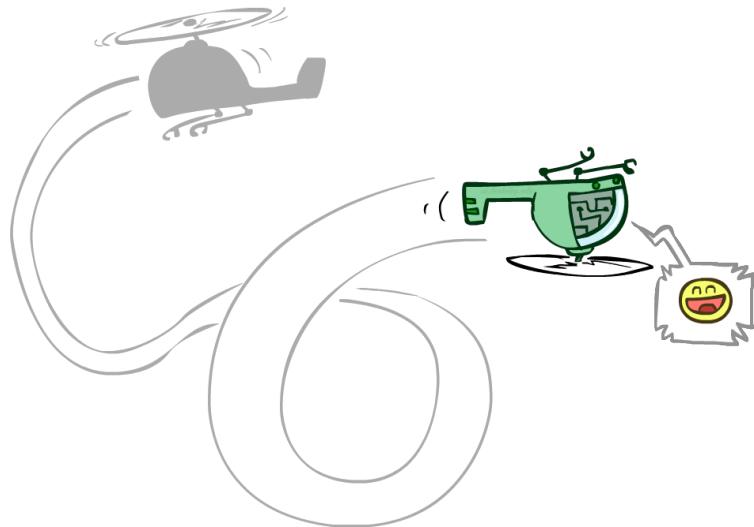
Therefore, the stationary distribution corresponds to the (normalized) eigenvector of the transposed transition matrix with an eigenvalue of 1 .

Applications

HMMs are used in many fields where the goal is to recover a data sequence that is not immediately observable, but other data that depend on the sequence are.

- Speech recognition (see Lecture 10)
- Speech synthesis
- Part-of-speech tagging
- Machine translation
- Robotics
- Computational finance
- Handwriting recognition
- Time series analysis
- Activity recognition

Filters



Suppose we want track the position and velocity of a robot from noisy observations collected over time.

Formally, we want to estimate **continuous** state variables such as

- the position \mathbf{X}_t of the robot at time t ,
- the velocity $\dot{\mathbf{X}}_t$ of the robot at time t .

We assume **discrete** time steps.

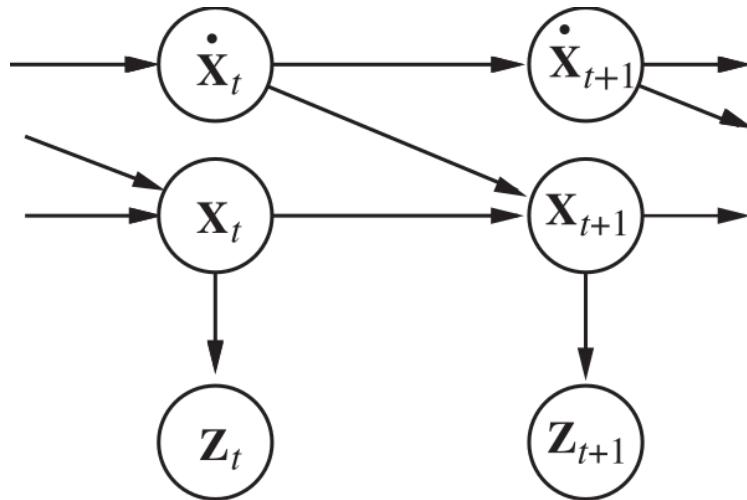
The Bayes filter similarly applies to **continuous** state and evidence variables \mathbf{X}_t and \mathbf{E}_t , in which case summations are replaced with integrals:

$$p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t+1}) = \alpha p(\mathbf{e}_{t+1} | \mathbf{x}_{t+1}) \int p(\mathbf{x}_{t+1} | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{e}_{1:t}) d\mathbf{x}_t$$

where the normalization constant is

$$\alpha = 1 / \int p(\mathbf{e}_{t+1} | \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t}) d\mathbf{x}_{t+1}.$$

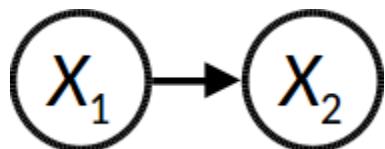
Kalman filter



The **Kalman filter** is a special case of the Bayes filter, which assumes:

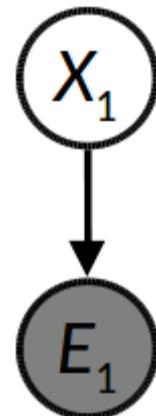
- Gaussian prior
- Linear Gaussian transition model
- Linear Gaussian sensor model

Linear Gaussian models



$$p(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{Ax}_t + \mathbf{b}, \Sigma_{\mathbf{x}})$$

Transition model



$$p(\mathbf{e}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{e}_t|\mathbf{Cx}_t + \mathbf{d}, \Sigma_{\mathbf{e}})$$

Sensor model

Cheat sheet for Gaussian models (Bishop, 2006)

Given a marginal Gaussian distribution for \mathbf{x} and a linear Gaussian distribution for \mathbf{y} given \mathbf{x} in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1})$$

the marginal distribution of \mathbf{y} and the conditional distribution of \mathbf{x} given \mathbf{y} are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T)$$
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}(\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}), \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A})^{-1}.$$

Filtering Gaussian distributions

- *Prediction step:*

If the distribution $p(\mathbf{x}_t | \mathbf{e}_{1:t})$ is Gaussian and the transition model $p(\mathbf{x}_{t+1} | \mathbf{x}_t)$ is linear Gaussian, then the one-step predicted distribution given by

$$p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t}) = \int p(\mathbf{x}_{t+1} | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{e}_{1:t}) d\mathbf{x}_t$$

is also a Gaussian distribution.

- *Update step:*

If the prediction $p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t})$ is Gaussian and the sensor model $p(\mathbf{e}_{t+1} | \mathbf{x}_{t+1})$ is linear Gaussian, then after conditioning on new evidence, the updated distribution

$$p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t+1}) \propto p(\mathbf{e}_{t+1} | \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} | \mathbf{e}_{1:t})$$

is also a Gaussian distribution.

Therefore, for the Kalman filter, $p(\mathbf{x}_t | \mathbf{e}_{1:t})$ is a multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ for all t .

- Filtering reduces to the computation of the parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$.
- By contrast, for general (nonlinear, non-Gaussian) processes, the description of the posterior grows **unboundedly** as $t \rightarrow \infty$.

1D example

Gaussian random walk:

- Gaussian prior:

$$p(x_0) = \mathcal{N}(x_0 | \mu_0, \sigma_0^2)$$

- The transition model adds random perturbations of constant variance:

$$p(x_{t+1} | x_t) = \mathcal{N}(x_{t+1} | x_t, \sigma_x^2)$$

- The sensor model yields measurements with Gaussian noise of constant variance:

$$p(e_t | x_t) = \mathcal{N}(e_t | x_t, \sigma_e^2)$$

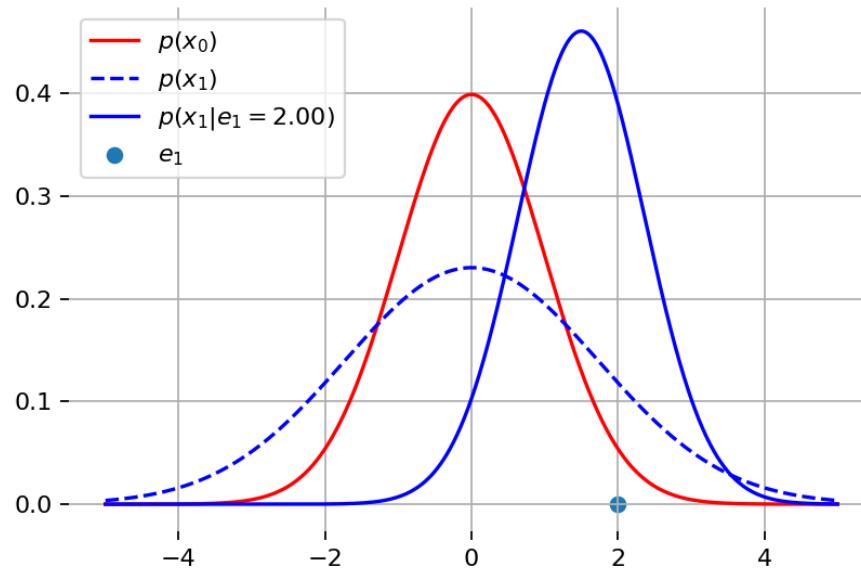
The one-step predicted distribution is given by

$$\begin{aligned} p(x_1) &= \int p(x_1|x_0)p(x_0)dx_0 \\ &= \alpha \int \exp\left(-\frac{1}{2}\frac{(x_1 - x_0)^2}{\sigma_x^2}\right) \exp\left(-\frac{1}{2}\frac{(x_0 - \mu_0)^2}{\sigma_0^2}\right) dx_0 \\ &= \alpha \int \exp\left(-\frac{1}{2}\frac{\sigma_0^2(x_1 - x_0)^2 + \sigma_x^2(x_0 - \mu_0)^2}{\sigma_0^2\sigma_x^2}\right) dx_0 \\ &\dots \text{ (simplify by completing the square)} \\ &= \alpha \exp\left(-\frac{1}{2}\frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2}\right) \\ &= \mathcal{N}(x_1|\mu_0, \sigma_0^2 + \sigma_x^2) \end{aligned}$$

Note that the same result can be obtained by using instead the Gaussian models identities.

For the update step, we need to condition on the observation at the first time step:

$$\begin{aligned}
 p(x_1|e_1) &= \alpha p(e_1|x_1)p(x_1) \\
 &= \alpha \exp\left(-\frac{1}{2} \frac{(e_1 - x_1)^2}{\sigma_e^2}\right) \exp\left(-\frac{1}{2} \frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2}\right) \\
 &= \alpha \exp\left(-\frac{1}{2} \frac{\left(x_1 - \frac{(\sigma_0^2 + \sigma_x^2)e_1 + \sigma_e^2\mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}\right)^2}{\frac{(\sigma_0^2 + \sigma_x^2)\sigma_e^2}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}}\right) \\
 &= \mathcal{N}\left(x_1 \middle| \frac{(\sigma_0^2 + \sigma_x^2)e_1 + \sigma_e^2\mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}, \frac{(\sigma_0^2 + \sigma_x^2)\sigma_e^2}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}\right)
 \end{aligned}$$



In summary, the update equations given a new evidence e_{t+1} are:

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)e_{t+1} + \sigma_e^2\mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}$$

$$\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_e^2}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}$$

General Kalman update

The same derivations generalize to multivariate normal distributions.

Assuming the transition and sensor models

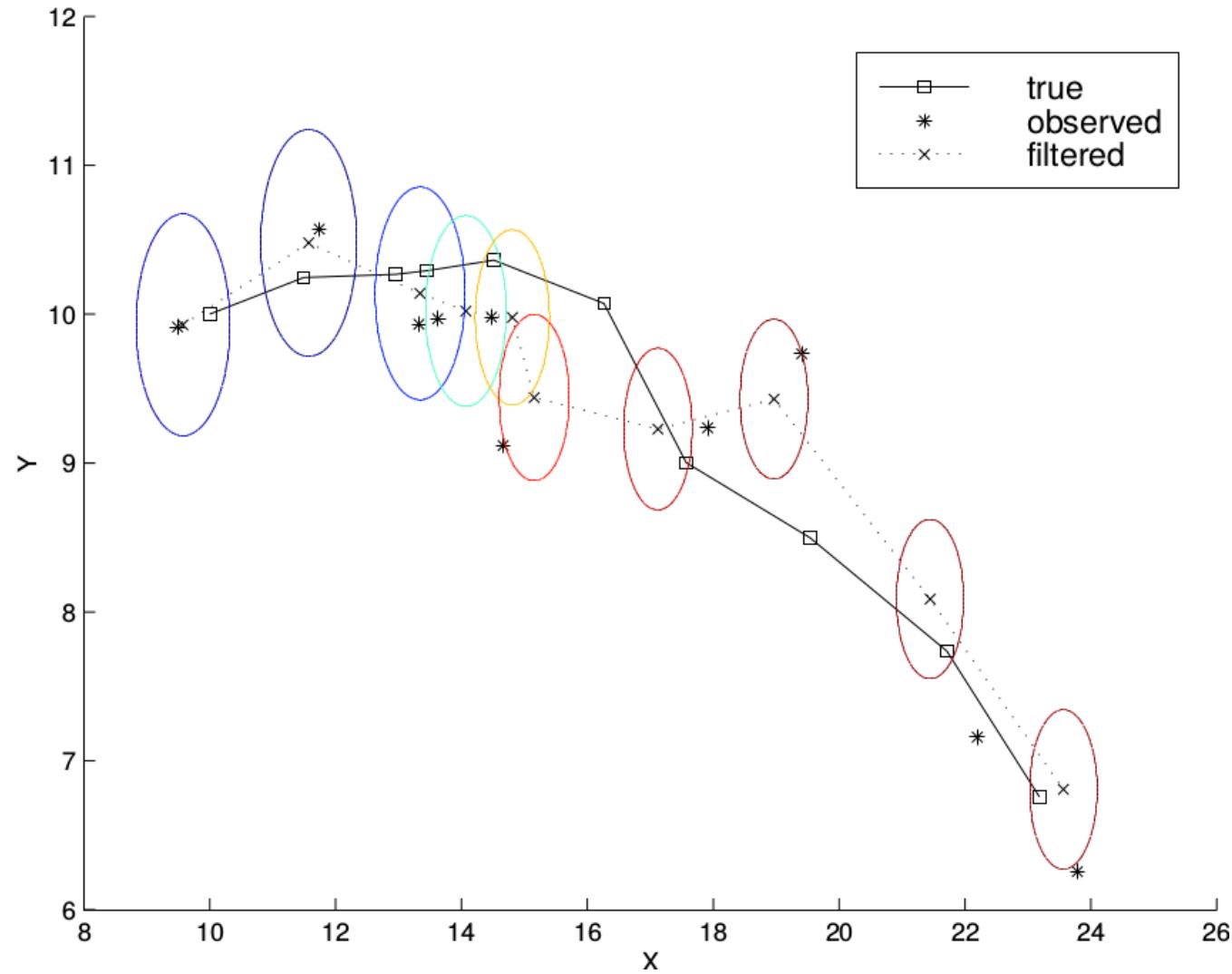
$$p(\mathbf{x}_{t+1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1} | \mathbf{F}\mathbf{x}_t, \boldsymbol{\Sigma}_x)$$
$$p(\mathbf{e}_t | \mathbf{x}_t) = \mathcal{N}(\mathbf{e}_t | \mathbf{H}\mathbf{x}_t, \boldsymbol{\Sigma}_e),$$

we arrive at the following general update equations:

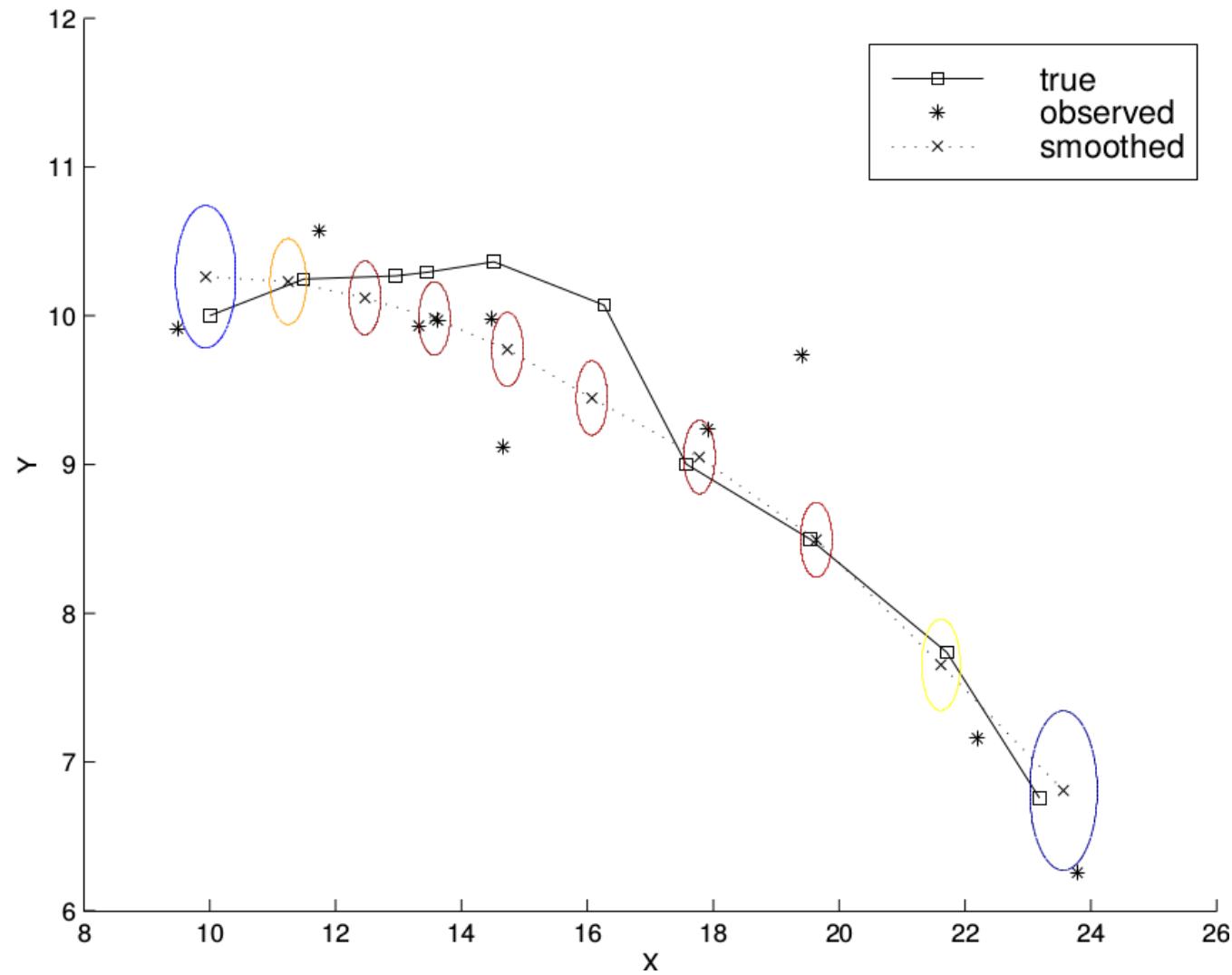
$$\mu_{t+1} = \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{e}_{t+1} - \mathbf{H}\mathbf{F}\mu_t)$$
$$\boldsymbol{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^T + \boldsymbol{\Sigma}_x)$$
$$\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^T + \boldsymbol{\Sigma}_x)\mathbf{H}^T(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^T + \boldsymbol{\Sigma}_x)\mathbf{H}^T + \boldsymbol{\Sigma}_e)^{-1}$$

where \mathbf{K}_{t+1} is the Kalman gain matrix.

2D tracking: filtering



2D tracking: smoothing



Apollo guidance computer

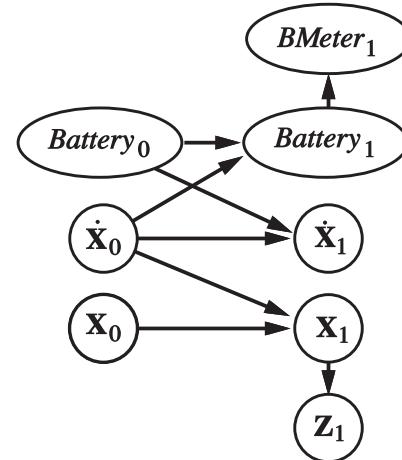
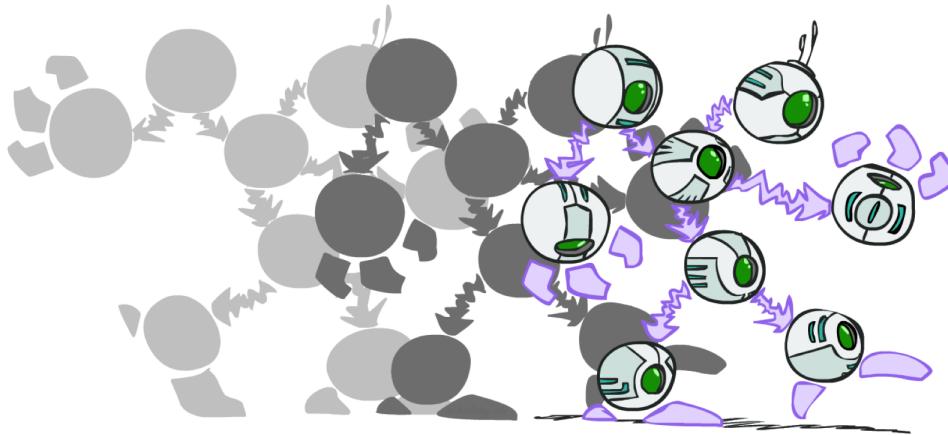
- The Kalman filter put man on the Moon, literally!
- The onboard guidance software of Saturn-V used a Kalman filter to merge new data with past position measurements to produce an optimal position estimate of the spacecraft.



```
# INCORP2 -- INCORPORATES THE COMPUTED STATE VECTOR DEVIATIONS INTO THE
# ESTIMATED STATE VECTOR.  THE STATE VECTOR UPDATED MAY BE FOR EITHER THE
# LEM OR THE CSM.  DETERMINED BY FLAG VEHUPFLG.  (ZERO = LEM) (1 = CSM)
#
# INPUT
#   PERMANENT STATE VECTOR FOR EITHER THE LEM OR CSM
#   VEHUPFLG = UPDATE VEHICLE C=LEM 1=CSM
#   W = ERROR TRANSITION MATRIX
#   DELTAX = COMPUTED STATE VECTOR DEVIATIONS
#   DMENFLG = SIZE OF W MATRIX (ZERO=6X6) (1=9X9)
#   GAMMA = SCALAR FOR INCORPORATION
#   ZI = VECTOR USED IN INCORPORATION
#   OMEGA = WEIGHTING VECTOR
#
# OUTPUT
#   UPDATED PERMANENT STATE VECTOR
```



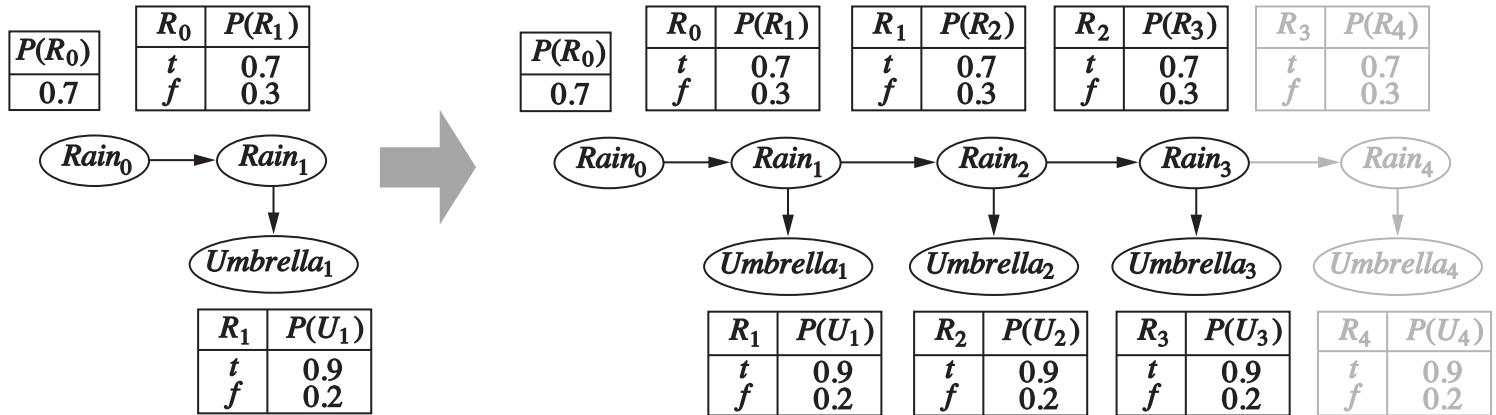
Dynamic Bayesian networks



Dynamics Bayesian networks (DBNs) can be used for tracking multiple variables over time, using multiple sources of evidence.

- Idea: Repeat a fixed Bayes net structure at each time t .
- Variables from time t condition on those from $t - 1$.
- DBNs are a generalization of HMMs and of the Kalman filter.

Exact inference



Unroll the network through time and run any exact inference algorithm (e.g., variable elimination)

- 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 $O(d^{n+k})$ and the total update cost per step is $O(nd^{n+k})$.
 - Better than HMMs, which is $O(d^{2n})$, but still **infeasible** for large numbers of variables.

Likelihood weighting

If exact inference in DBNs intractable, then let's use [approximate inference](#) instead.

- Likelihood weighting? Generated LW samples **pay no attention** to the evidence!
- The fraction of samples that remain close to the actual series of events drops exponentially with t .
- Therefore, the number of required samples for inference grows exponentially with t .

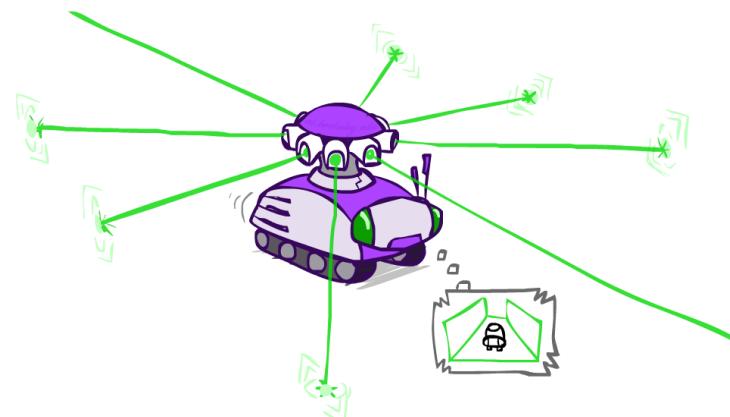
⇒ We need a better solution!

Particle filter

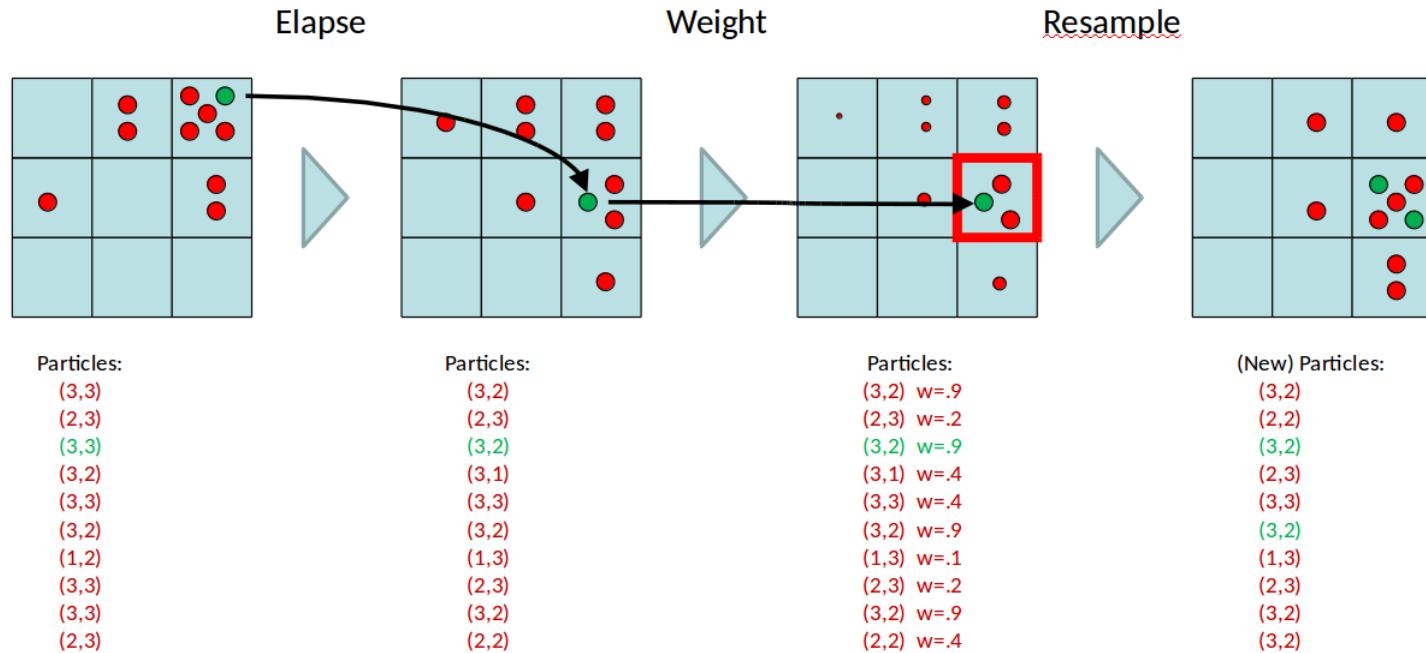
Basic idea:

- Maintain a finite population of samples, called **particles**.
 - The representation of our beliefs is a list of N particles.
- Ensure the particles track the high-likelihood regions of the state space.
- Throw away samples that have very low weight, according to the evidence.
- Replicate those that have high weight.

This scales to high dimensions!



Update cycle



function PARTICLE-FILTERING(\mathbf{e}, N, dbn) **returns** a set of samples for the next time step

inputs: \mathbf{e} , the new incoming evidence
 N , the number of samples to be maintained
 dbn , a DBN with prior $\mathbf{P}(\mathbf{X}_0)$, transition model $\mathbf{P}(\mathbf{X}_1|\mathbf{X}_0)$, sensor model $\mathbf{P}(\mathbf{E}_1|\mathbf{X}_1)$

persistent: S , a vector of samples of size N , initially generated from $\mathbf{P}(\mathbf{X}_0)$
local variables: W , a vector of weights of size N

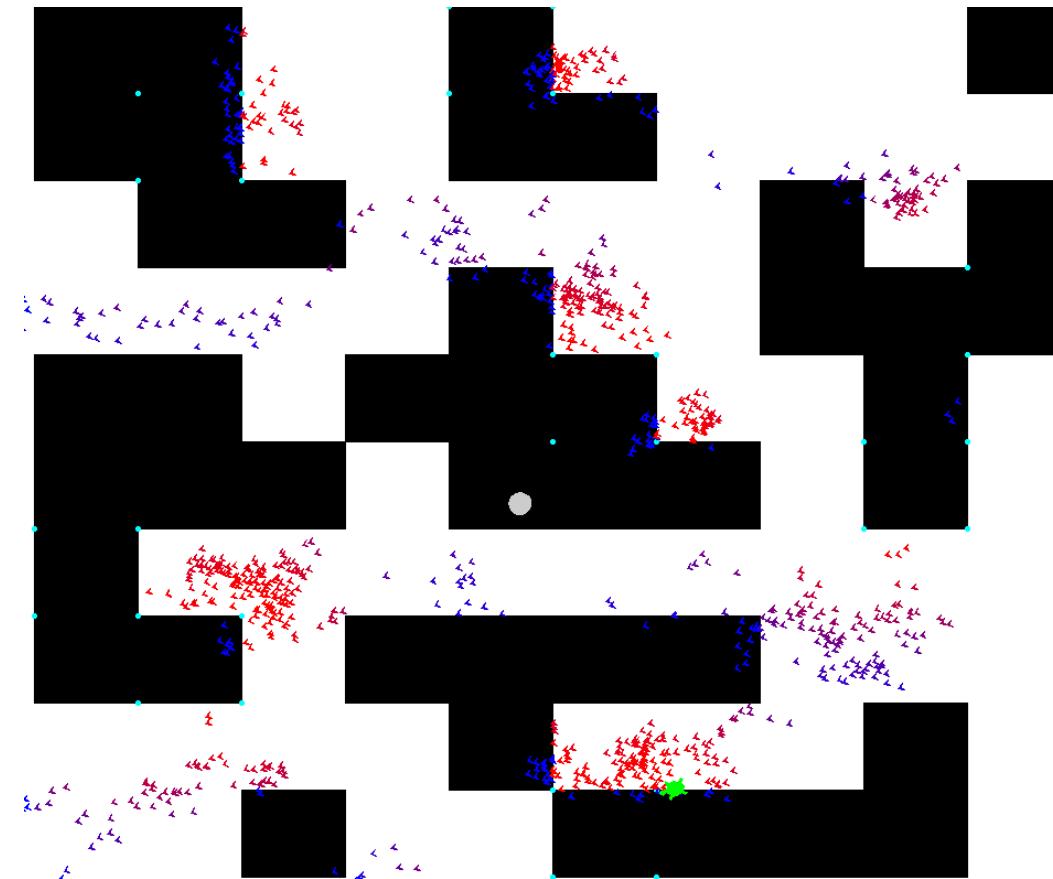
for $i = 1$ to N **do**

- $S[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0 = S[i])$ /* step 1 */
- $W[i] \leftarrow \mathbf{P}(\mathbf{e} \mid \mathbf{X}_1 = S[i])$ /* step 2 */

$S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W) /* step 3 */

return S

Robot localization



(See demo)

Summary

- Temporal models use state and sensor variables replicated over time.
 - Their purpose is to maintain a belief state as time passes and as more evidence is collected.
- The Markov and stationarity assumptions imply that we only need to specify
 - a transition model $P(\mathbf{X}_{t+1}|\mathbf{X}_t)$,
 - a sensor model $P(\mathbf{E}_t|\mathbf{X}_t)$.
- Inference tasks include filtering, prediction, smoothing and finding the most likely sequence.
- Filter algorithms are all based on the core of idea of
 - projecting the current belief state through the transition model,
 - updating the prediction according to the new evidence.

References

- Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." *Journal of basic Engineering* 82.1 (1960): 35-45.
- Bishop, Christopher "Pattern Recognition and Machine Learning" (2006).