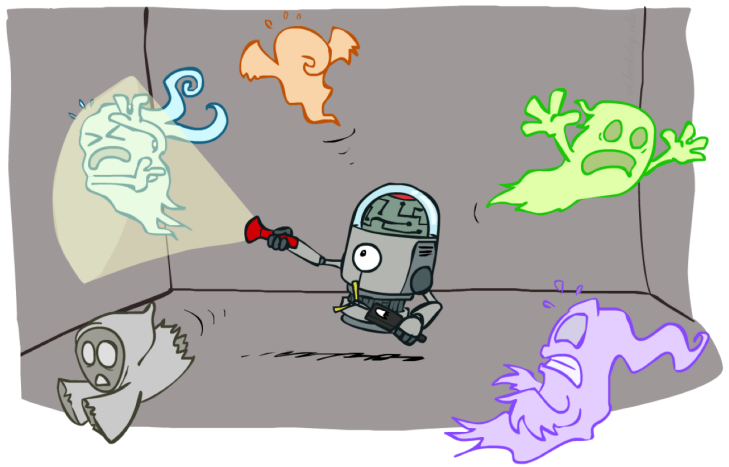
Introduction to Artificial Intelligence

Lecture 7: Reasoning over time

LIÈGE université

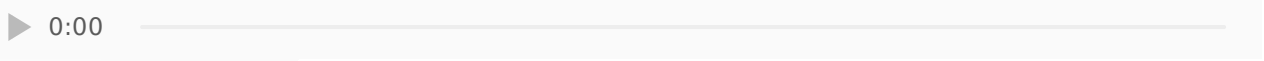
Today

- Markov models
  - Markov processes
  - Inference tasks
    - Prediction
    - Filtering
    - Smoothing
    - Most likely explanation
  - Hidden Markov models
- Filtering
  - Kalman filter
  - Dynamic Bayesian networks
  - Particle filters



Credits: UC Berkeley, CS188

Pacman revenge



[Q] How to make good use of the sonar readings?

Credits: UC Berkeley, CS188

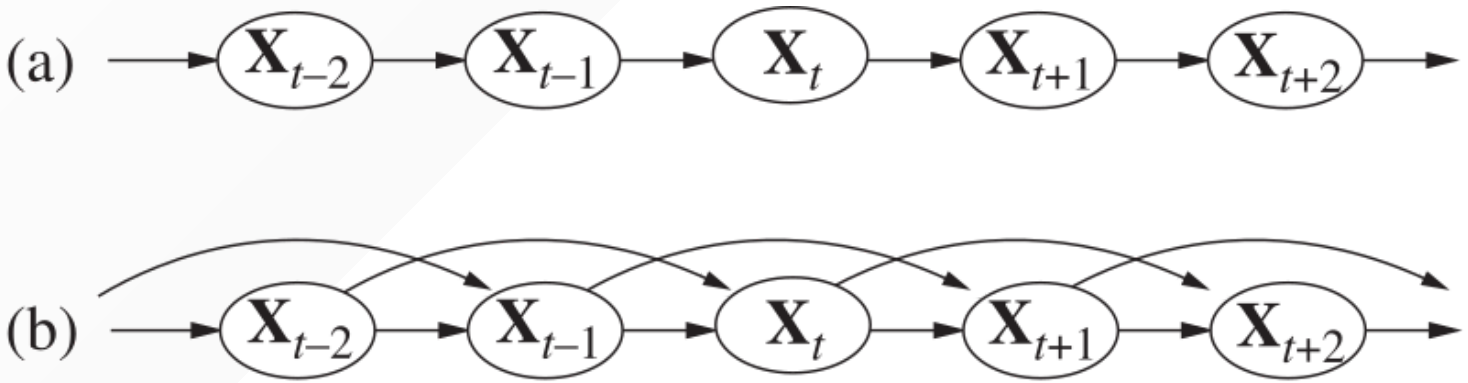
Markov models

Reasoning over time

- Often, we want to reason about a sequence of observations.
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring.
- Therefore, we need to introduce time (or space) in our model.
- 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 transition model $P(\mathbf{X}_t | \mathbf{X}_{0:t-1})$ that defines the probability distribution over the latest state variables, given the previous values.
- Similarly, we define a sensor model $P(\mathbf{E}_t | \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1})$.

Markov processes (1)

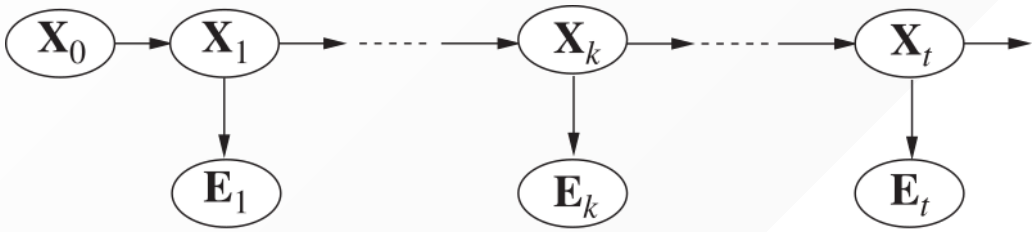
- Markov assumption: $\mathbf{X}_t$ depends on only a bounded subset of $\mathbf{X}_{0:t-1}$.
  - Processes that satisfy this assumption are called Markov processes.
- First-order Markov processes: $P(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = P(\mathbf{X}_t|\mathbf{X}_{t-1})$.
  - i.e., $\mathbf{X}_t$ and $\mathbf{X}_{0:t-2}$ are conditionally independent given $\mathbf{X}_{t-1}$.
- Second-order Markov processes:
  $P(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = P(\mathbf{X}_t|\mathbf{X}_{t-2},\mathbf{X}_{t-1})$.



Markov processes (2)

- Additionally, we make a sensor Markov assumption:
  $P(\mathbf{E}_t|\mathbf{X}_{0:t},\mathbf{E}_{0:t-1}) = P(\mathbf{E}_t|\mathbf{X}_t)$
- Stationary process: 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

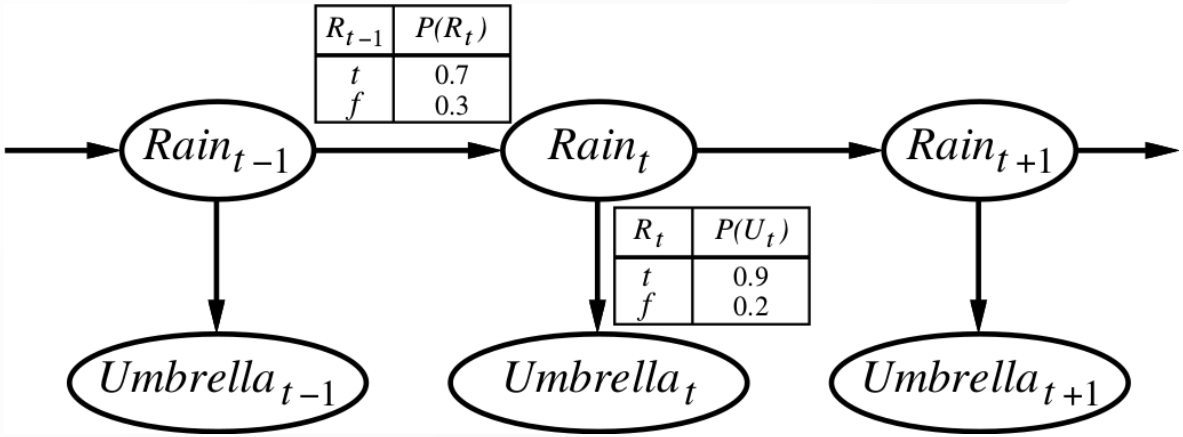


- A Markov process can be described as a growable Bayesian network, unrolled infinitely through time, with a specified restricted structure between time steps.
  - i.e., we can use standard Bayesian network reasoning when truncating the sequence.
- Therefore, the joint distribution of all variables up to t in a (first-order) Markov process is:

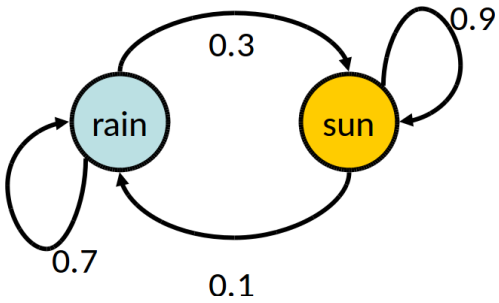
$$P(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = P(\mathbf{X}_0) \prod_{i=1}^{t} P(\mathbf{X}_i | \mathbf{X}_{i-1}) P(\mathbf{E}_i | \mathbf{X}_i)$$

[Q] Why is this true? Don't the variables $\mathbf{X}_{t+1:\infty}$ and $\mathbf{E}_{t+1:\infty}$ also characterize the joint distribution of the former?

Example: Weather



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



Transition model

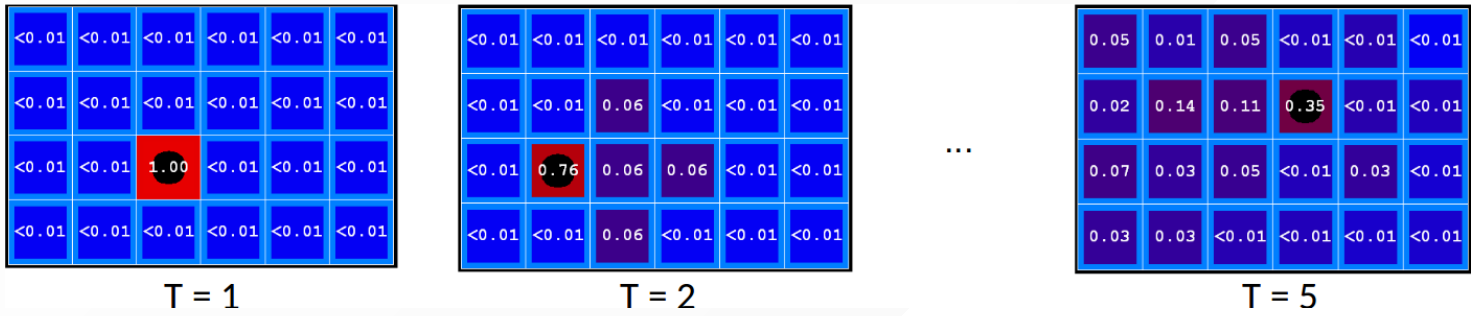
$P(Rain_t|Rain_{t-1})$

[Q] How else can you represent the transition model?

Inference tasks

- 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.
- 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.
- 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.

Prediction



- To predict the future, the current belief state $P(\mathbf{X}_t|\mathbf{e}_{1:t})$ get pushed through the transition model.

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

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

Ghostbusters: Basic dynamics

Credits: UC Berkeley, CS188

Ghostbusters: Circular dynamics

Credits: UC Berkeley, CS188

Ghostbusters: Whirlpool

Credits: UC Berkeley, CS188

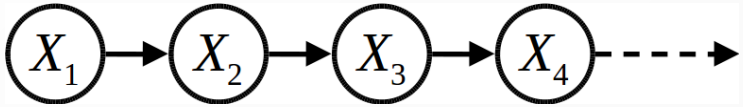
Stationary distributions

What if $t \to \infty$?

- 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.
- It satisfies:

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

Example



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

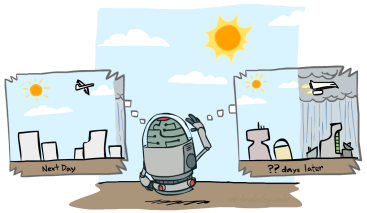
$\mathbf{X}_{t-1}$	$\mathbf{X}_t$	P
sun	sun	0.9
sun	$rain$	0.1
$rain$	sun	0.3
$rain$	$rain$	0.7

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

Which implies that:

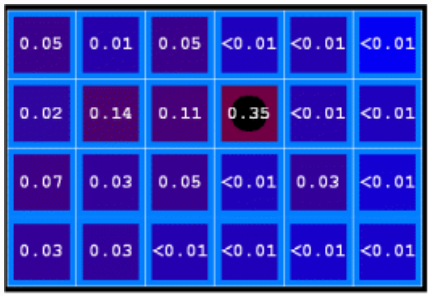
$P(\mathbf{X}_\infty = sun) = \frac{3}{4}$

$P(\mathbf{X}_\infty = rain) = \frac{1}{4}$

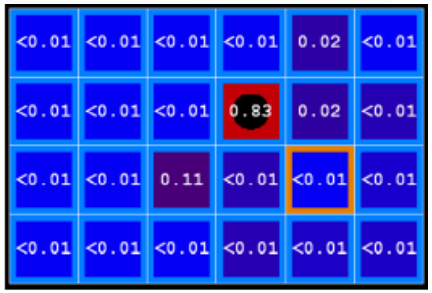


Credits: UC Berkeley, CS188

Observation



Before observation



After observation

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

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

Filtering (1)

- An agent should maintain a current belief state estimate $P(\mathbf{X}_t|\mathbf{e}_{1:t})$ and update it as new evidences $\mathbf{e}_{t+1}$ are collected.
  - Rather than going back over the entire history of percepts for each update.
- Recursive estimation: $P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, P(\mathbf{X}_t|\mathbf{e}_{1:t}))$
  - Project the current state belief forward from t to $t+1$
  - Update this new state using the evidence $\mathbf{e}_{t+1}$.

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

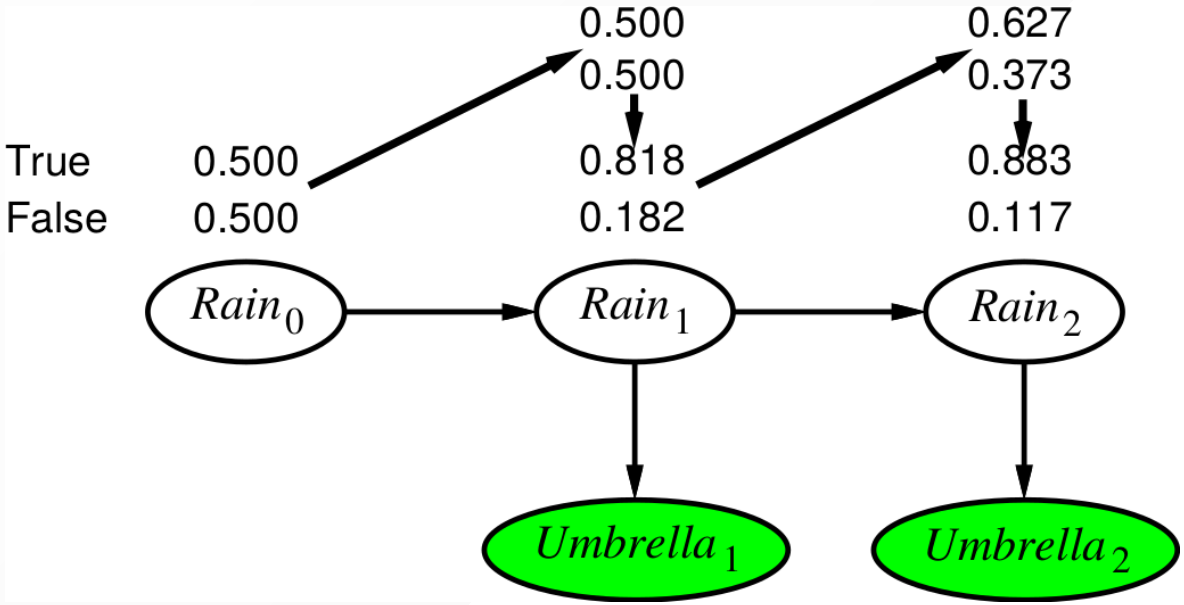
The first and second terms are given by the model. The third is obtained recursively.

Filtering (2)

- 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 $\mathbf{f}_{1:t+1} = \alpha\,\mathrm{forward}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$.
- The complexity of a forward update is constant (in time and space) with t.

[Q] What is the explicit form of the normalization constant α?

Example



R_{t-1}	$P(R_t)$
$true$	0.7
$false$	0.3

R_t	$P(U_t)$
$true$	0.9
$false$	0.2

Smoothing

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

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

The backward message $\mathbf{b}_{k+1:t}$ can be computed using backwards recursion:

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

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

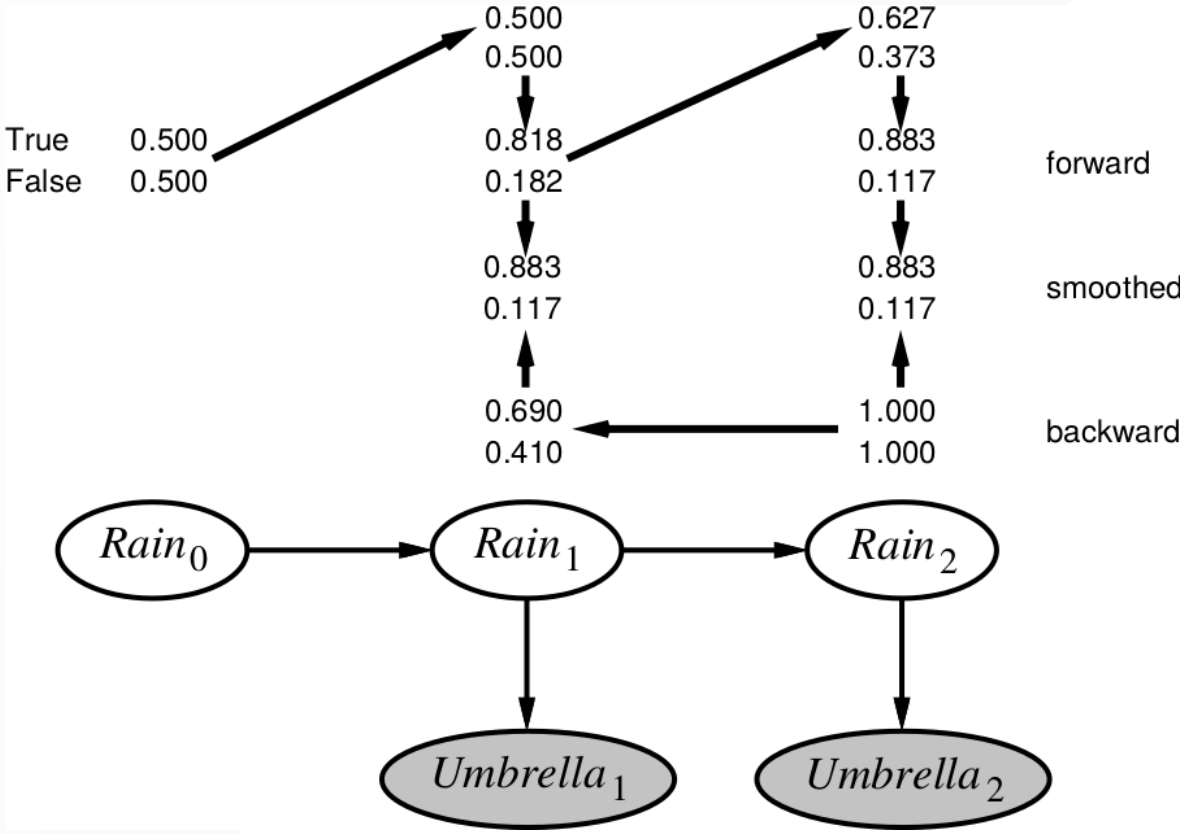
function FORWARD-BACKWARD(**ev**, $prior$) **returns** a vector of probability distributions
  inputs: **ev**, a vector of evidence values for steps $1, \ldots, t$
    $prior$, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$
  local variables: **fv**, a vector of forward messages for steps $0, \ldots, t$
    b, a representation of the backward message, initially all 1s
    sv, a vector of smoothed estimates for steps $1, \ldots, t$

  fv$[0] \leftarrow prior$
  for $i = 1$ **to** t **do**
    fv$[i] \leftarrow$ FORWARD(**fv**$[i-1]$, **ev**$[i]$)
  for $i = t$ **downto** 1 **do**
    sv$[i] \leftarrow$ NORMALIZE(**fv**$[i] \times$ **b**)
    b $\leftarrow$ BACKWARD(**b**, **ev**$[i]$)
  return sv

Complexity:

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

Example


0.500
0.500
0.627
0.373
True
False
0.500
0.500
0.818
0.182
0.883
0.117
forward
0.883
0.117
0.883
0.117
smoothed
0.690
0.410
1.000
1.000
backward
$Rain_0$
$Rain_1$
$Rain_2$
$Umbrella_1$
$Umbrella_2$

Most likely explanation

- The most likely sequence is not the sequence of 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,

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

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

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

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

- The update has its sum replaced by max, giving 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

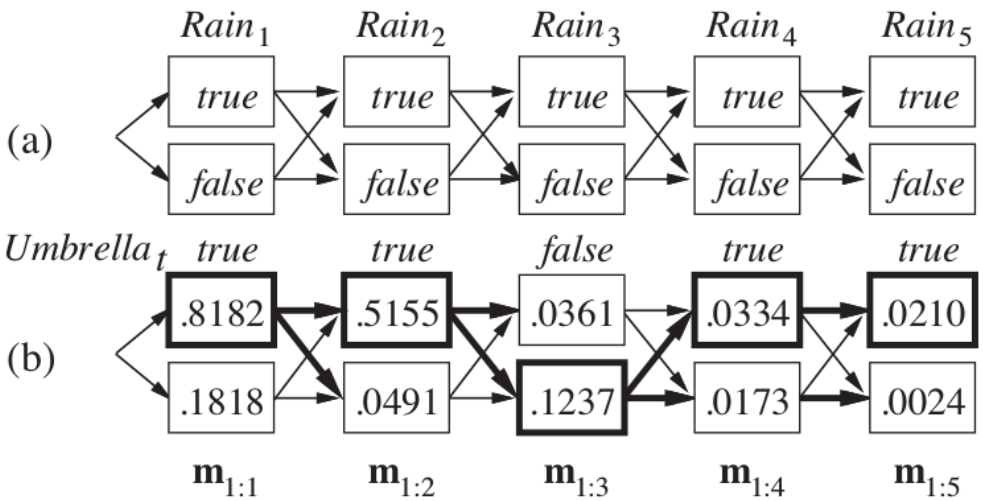


Figure 15.5 (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence $[true, true, false, true, true]$. 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.

[Q] How do you retrieve the path, in addition to its likelihood?

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.
  - i.e., $\mathbf{X}_t = X_t$, with domain $\{1, ..., S\}$.
- This restricted structure allows for a simple matrix implementation of the inference algorithms.

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 transition model $P(X_t|X_{t-1})$ becomes an $S \times S$ transition matrix $\mathbf{T}$, where $\mathbf{T}_{ij} = P(X_t = j|X_{t-1} = i)$.
- The sensor model $P(E_t|X_t)$ is defined, for convenience, as an $S \times S$ sensor matrix $\mathbf{O}_t$ whose i-th diagonal element is $P(e_t|X_t = i)$ and whose other entries are 0.
- If we use column vectors to represent forward and backward messages, then we have:

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

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

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

Applications

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

- Computational finance
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Machine translation
- Handwriting recognition
- Time series analysis
- Activity recognition
- ...

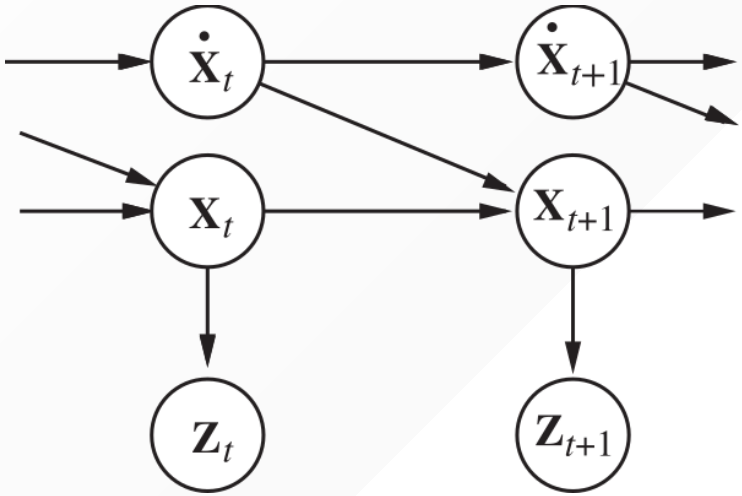
Filters

Continuous state variables

- From noisy observations collected over time, we want to estimate continuous state variables, e.g.
  - position $\mathbf{X}_t$,
  - velocity $\dot{\mathbf{X}}_t$.
- We still assume discrete time steps.
- Applications:
  - tracking
  - robots
  - guidance
  - planet motion
  - financial time series
  - ...

[Q] How can we model this system to make filtering efficient and accurate?

Kalman filters



A Kalman filter assumes:

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

Updating Gaussian distributions

- Prediction step:
  - If $P(\mathbf{X}_t|\mathbf{e}_{1:t})$ is Gaussian and the transition model $P(\mathbf{X}_{t+1}|\mathbf{x}_t)$ is linear Gaussian, then

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

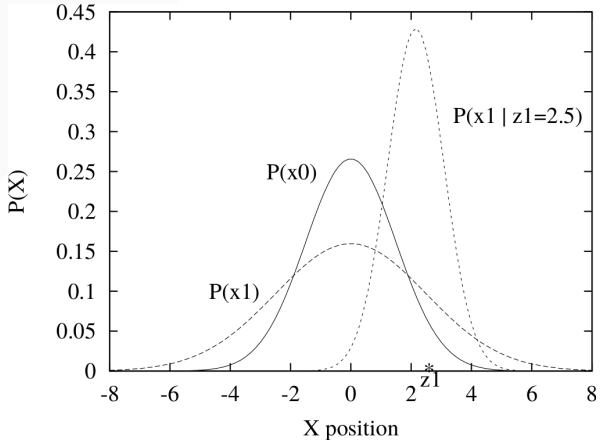
    is Gaussian.
- Update step:
  - If $P(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$ is Gaussian and the sensor model $P(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ is linear Gaussian, then

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

    is also Gaussian.
- Hence, for a Kalman filter, $P(\mathbf{X}_t|\mathbf{e}_{1:t})$ is a multivariate Gaussian $\mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$ for all t.
- General (nonlinear, non-Gaussian) process: the description of the posterior grows unboundedly as $t \to \infty$.

1D example

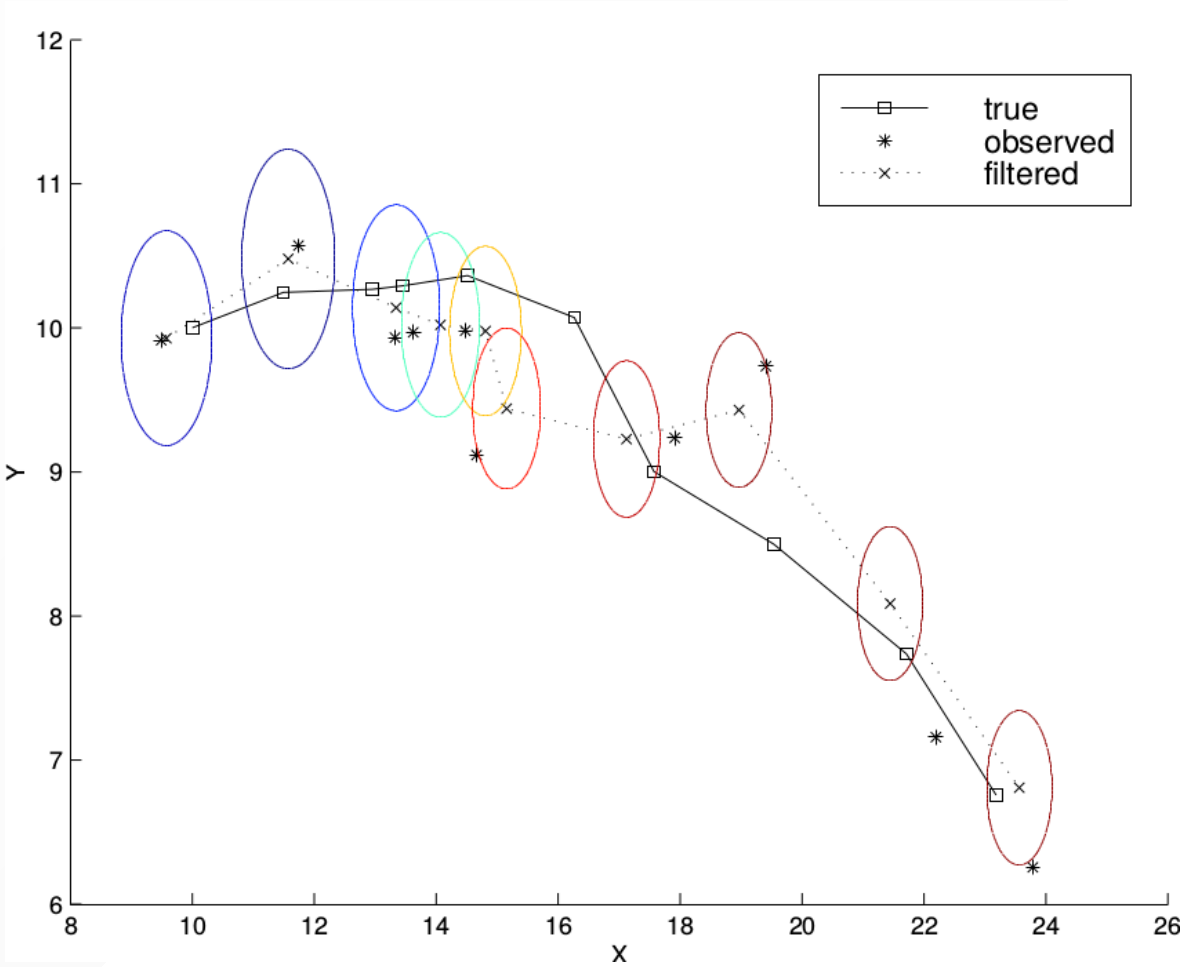
- Gaussian random walk on X-axis:
  - Gaussian prior with variance σ_0^2.
  - The transition model adds random perturbations of constant variance σ_x^2.
  - The sensor model yields measurements with Gaussian noise with variance σ_z^2.
- Then the update equations given a new evidence z_{t+1} are:
  - $\mu_{t+1} = \frac{(\sigma_t^2+\sigma_x^2)z_{t+1}+\sigma_z^2\mu_t}{\sigma_t^2+\sigma_x^2+\sigma_z^2}$
  - $\sigma_{t+1}^2 = \frac{(\sigma_t^2+\sigma_x^2)\sigma_z^2}{\sigma_t^2+\sigma_x^2+\sigma_z^2}$



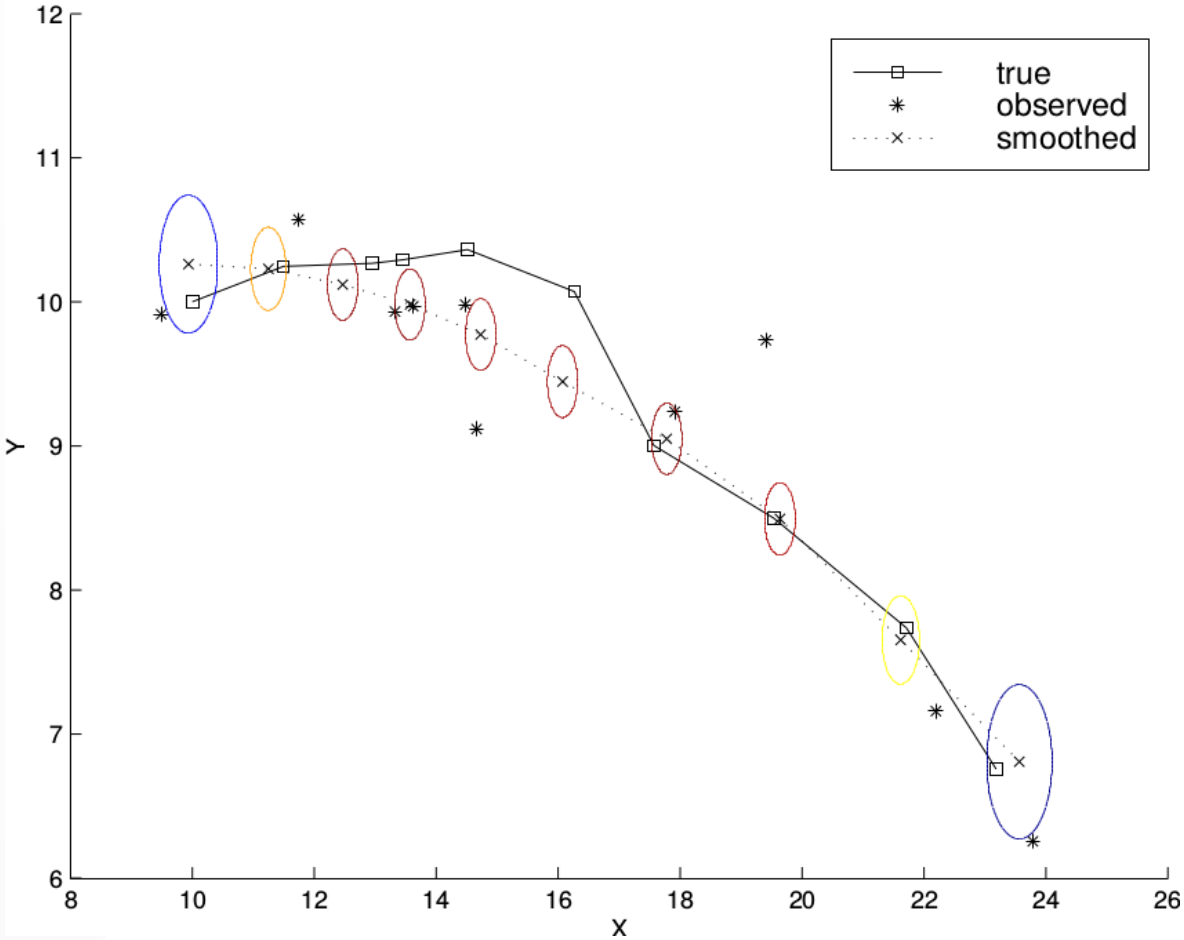
General Kalman update

- Transition and sensor models:
  - $P(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{F}\mathbf{x}_t, \mathbf{\Sigma}_x)(\mathbf{x}_{t+1})$
  - $P(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{H}\mathbf{x}_t, \mathbf{\Sigma}_z)(\mathbf{z}_t)$
- $\mathbf{F}$ and $\mathbf{\Sigma}_x$ are matrices describing the linear transition model and transition noise covariance.
- $\mathbf{H}$ and $\mathbf{\Sigma}_z$ are the corresponding matrices for the sensor model.
- The filter computes the following update:
  - $\mu_{t+1} = \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t)$
  - $\mathbf{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\mathbf{\Sigma}_t\mathbf{F}^T + \mathbf{\Sigma}_x)$
    - where $\mathbf{K}_{t+1} = (\mathbf{F}\mathbf{\Sigma}_t\mathbf{F}^T + \mathbf{\Sigma}_x)\mathbf{H}^T(\mathbf{H}(\mathbf{F}\mathbf{\Sigma}_t\mathbf{F}^T + \mathbf{\Sigma}_x)\mathbf{H}^T + \mathbf{\Sigma}_z)^{-1}$ is the Kalman gain matrix.
- Note that $\mathbf{\Sigma}_t$ and $\mathbf{K}_t$ are independent of the evidence. Therefore, they can be computed offline.

2D tracking: filtering



2D tracking: smoothing



Apollo Guidance Computer

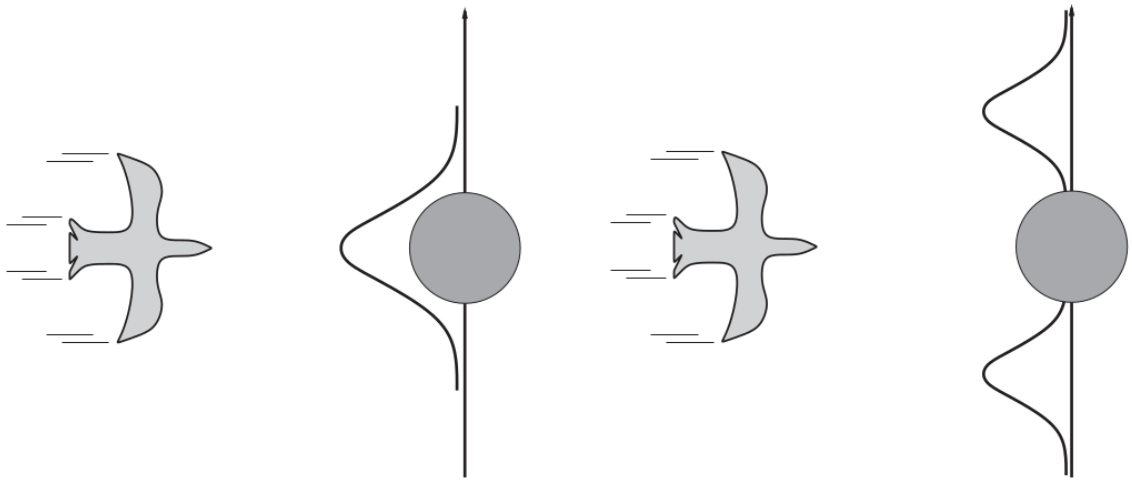
- The Kalman filter put man on the Moon, literally!
- The onboard guidance software of Saturn-V used a Kalman filter.
    - Used to merge new data with past position measurements to produce an optimal position estimate of the spacecraft.

```
# 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
```

Credits: Apollo-11 source code

Limitations

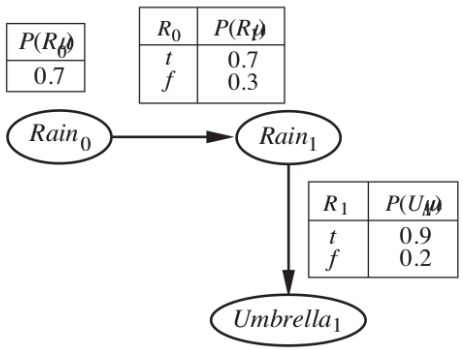
- The Kalman filter cannot be applied if the transition model is non-linear.
- The Extended Kalman Filter models transitions as locally linear around $\mathbf{x}_t = \mu_t$.
  - Still fails if the system is locally unsmooth.

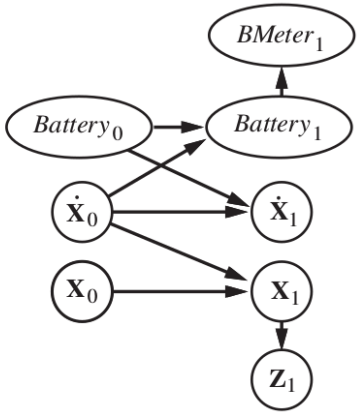


(a) (b)

Dynamic Bayesian networks

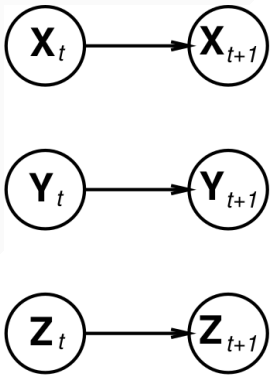
- A dynamic Bayesian network (DBN) is a Bayesian network that represents a temporal probability model.
  - Over infinitely many time steps.
- Each slice of a DBN can have any number of state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$.
- Nodes can be arbitrarily connected
  - to nodes from the same slice
  - to nodes from previous slices
- The Markov assumption need not to be satisfied.

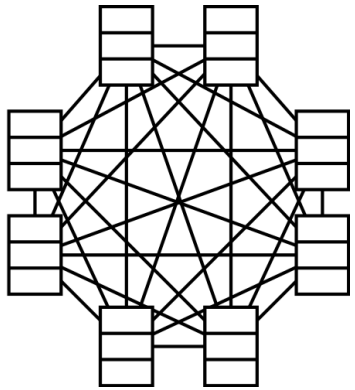




DBNs vs HMMs

- Every HMM is a single-variable DBN.
- Every discrete DBN can be transformed into an HMM.
  - Group state (resp. evidence) variables into a single variable whose values are all possible tuples of values of the individual state variables.
- By decomposing a system state into its constituent variables, DBNs can take advantage of the sparseness of the temporal probability model.
  - 20 boolean state variables, each of which has 3 parents in the preceding slice.
  - The DBN transition model counts $20 \times 2^3 = 160$ probabilities.
  - The corresponding HMM has 2^{20} state values and therefore 2^{40} probabilities in the transition matrix!

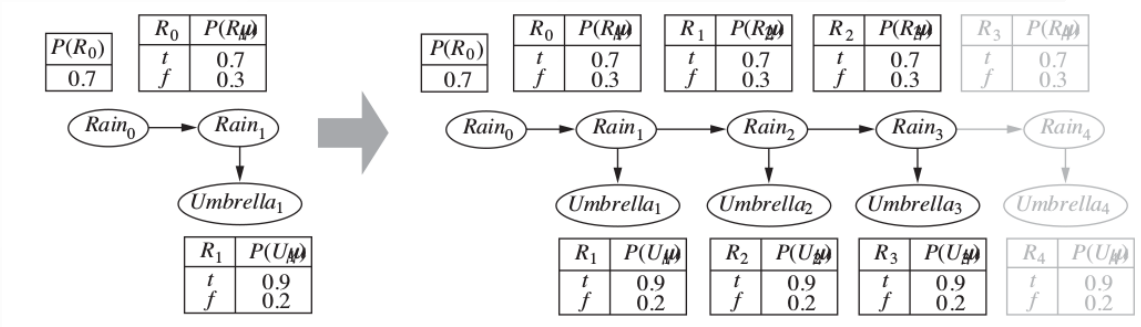




DBNs vs Kalman filters

- Every Kalman filter model is a DBN.
- Few DBNs are Kalman filter models.
  - The real world requires non-Gaussian posteriors.
  - DBNs can model arbitrary distributions.

Exact inference in DBNs

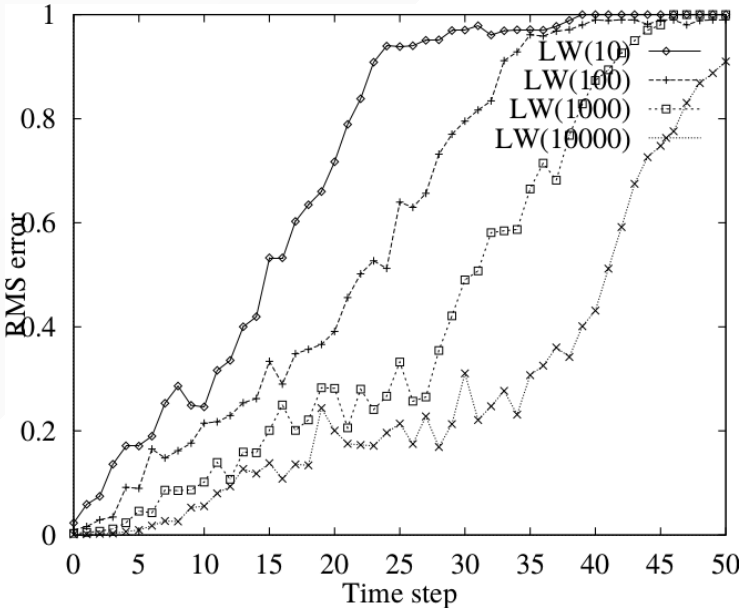


- Straightforward method for exact inference: unroll the network through time and run any exact algorithm.
  - e.g., variable elimination.
- Problem: inference cost for each update grows with t.
- Rollup filtering: add slice $t+1$, sum out slice t using variable elimination.
  - Largest factor is $O(d^{n+k})$ 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.

[Q] Compare rollup filtering to forward-backward in Markov processes.

Likelihood weighting for DBNs

- If exact inference is intractable, then let's use instead approximate inference. What about 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.



Particle filtering

- Basic idea:
  - Maintain a finite population of samples, called 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.
- Scale to high-dimensional state spaces ($n > 10^5$).
- Can be shown to be consistent.

---

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

Update cycle

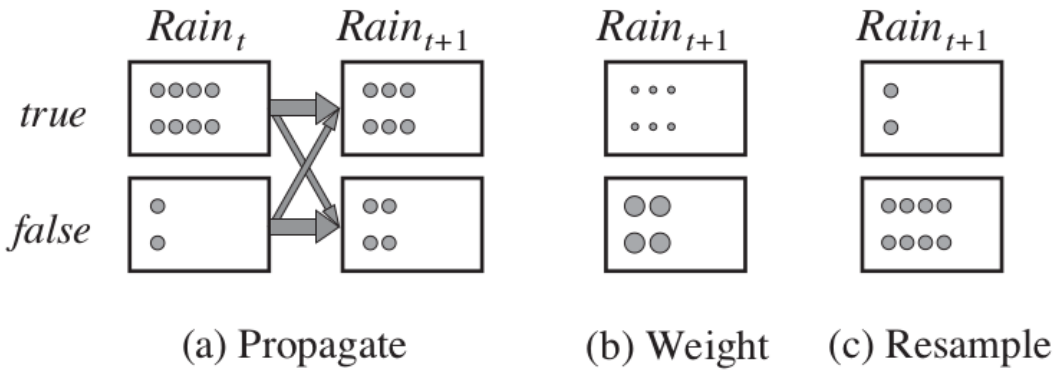
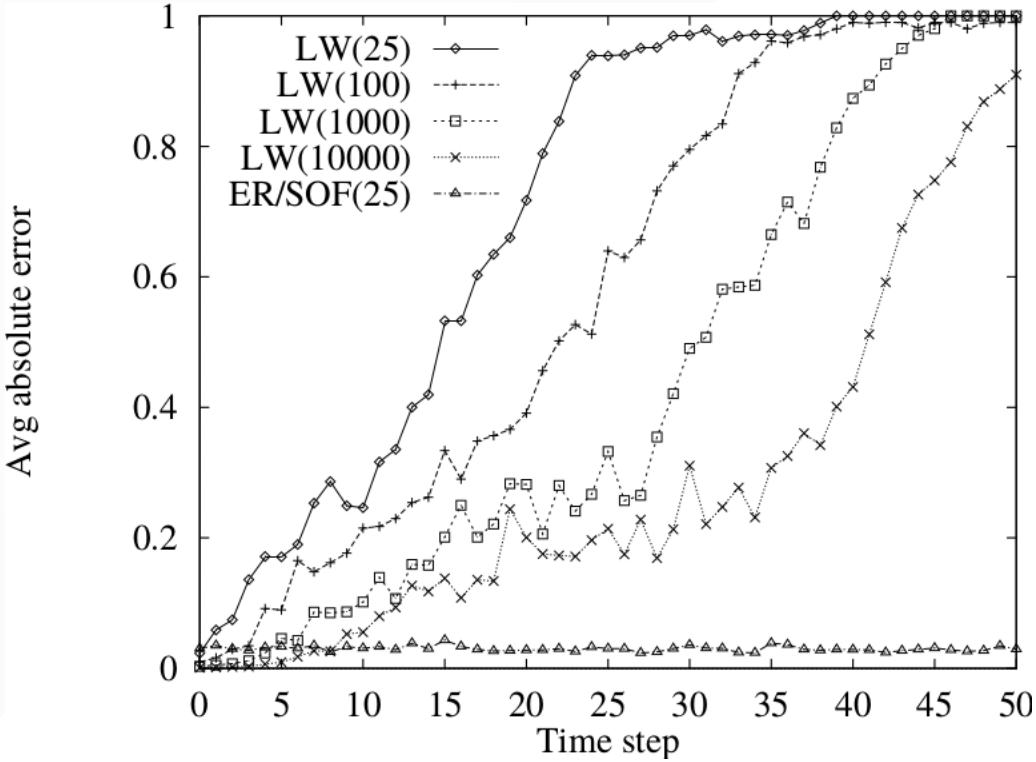


Figure 15.18 The particle filtering update cycle for the umbrella DBN with $N = 10$, showing the sample populations of each state. (a) At time t, 8 samples indicate *rain* and 2 indicate $\neg rain$. Each is propagated forward by sampling the next state through the transition model. At time $t + 1$, 6 samples indicate *rain* and 4 indicate $\neg rain$. (b) $\neg umbrella$ is observed at $t + 1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles. (c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples that indicate *rain* and 8 that indicate $\neg rain$.

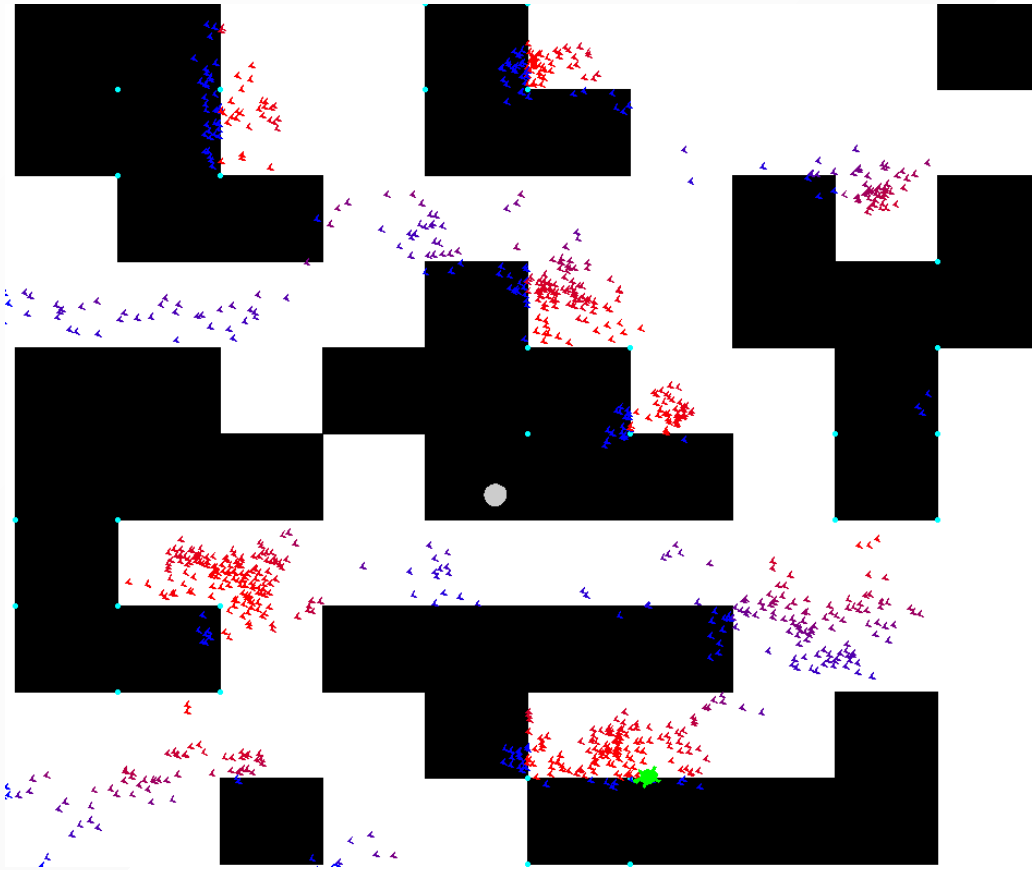
Performance

Approximation error of particle filtering remains bounded over time.

- At least empirically.
- Theoretical analysis is difficult.



Robot localization



Run demo.

Pacman, revisited

Credits: UC Berkeley, CS188

Summary

- Temporal models use state and sensor variables replicated over time.
- Markov assumptions and stationarity assumption. So we only need:
  - transition model $P(\mathbf{X}_{t+1}|\mathbf{X}_t)$
  - sensor model $P(\mathbf{E}_t|\mathbf{X}_t)$
- Inference tasks include filtering, prediction, smoothing and most likely sequence.
  - All can be done recursively with constant cost per time step.
- HMMs have a signel discrete state variable.
- Kalman filters allow n continuous state variables, assume a linear Gaussian model.
- DBNs generalize HMMs and Kalman filters.
  - Exact inference is usually intractable.
  - Particle filtering is a good approximate filtering algorithm for DBNs.