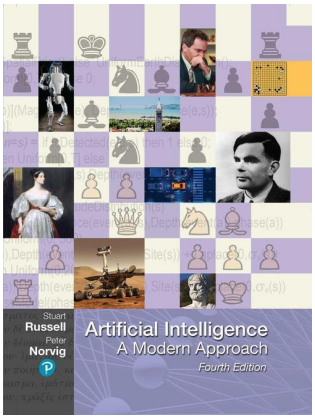
Artificial Intelligence: A Modern Approach

Fourth Edition



Chapter 14

Probabilistic Reasoning Over Time

Outline

- Time and Uncertainty
- Inference in Temporal Models
- Hidden Markov Models
- Kalman Filters
- Dynamic Bayesian Networks

Time and Uncertainty

States and observations

- discrete-time models, in which the world is viewed as a series of snapshots or time slices
- the time interval Δ between slices is assumed to be the same for every interval.
- $\mathbf{X}_t$: denotes the set of state variables at time t, which are assumed to be unobservable
- $\mathbf{E}_t$: denotes the set of observable evidence variables.
- The observation at time t is $\mathbf{E}_t = \mathbf{e}_t$

Transition and sensor models

- The transition model specifies the probability distribution over the latest state variables, given the previous values: $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$.
- Problem: the set $\mathbf{X}_{0:t-1}$ is unbounded in size as t increases.
- Solution: **Markov assumption** [the current state depends on only a finite fixed number of previous states]
- First order, $\boldsymbol{P}(\boldsymbol{X}_t \mid \mathbf{X}_{t-1})$; Second order, $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$;
- $\boldsymbol{P}(\boldsymbol{E}_t \mid \mathbf{X}_t)$ is our sensor model, sensor Markov assumption:

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

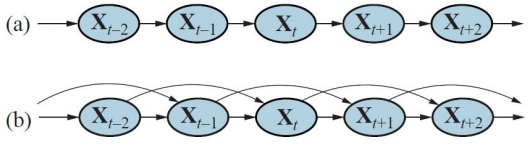
Time and Uncertainty

- the prior probability distribution at time 0, $\mathbf{P}(\mathbf{X}_0)$.

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

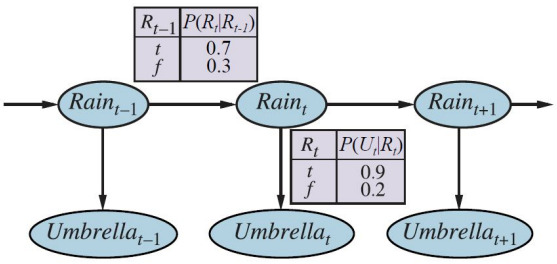
- Umbrella World: first-order Markov process—the probability of rain is assumed to depend only on whether it rained the previous day
- The first-order Markov assumption says that the state variables contain all the information needed to characterize the probability distribution for the next time slice.
- Ways to improve the accuracy of the approximation
  - Increasing the order of the Markov process mode
  - Increasing the set of state variables

Time and Uncertainty



(a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$
(b) A second-order Markov process.

Time and Uncertainty



Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $\mathbf{P}(Rain_t \mid Rain_{t-1})$ and the sensor model is $\mathbf{P}(Umbrella_t \mid Rain_t)$.

Inference in Temporal Models

- formulate the basic inference tasks that must be solved:
  - **Filtering** or **state estimation** is the task of computing the **belief state** $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$
  - **Prediction**: This is the task of computing the posterior distribution over the future state, given all evidence to date.
  - **Smoothing**: This is the task of computing the posterior distribution over a past state, given all evidence up to the present
  - **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations
- Besides inference tasks:
  - **Learning:** The transition and sensor models, if not yet known, can be learned from observations

Inference in Temporal Models

- Filtering and prediction

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

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \quad \text{(dividing up the evidence)} \\ &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(using Bayes' rule, given } \mathbf{e}_{1:t}) \\ &= \alpha\, \underbrace{\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\text{update}}\, \underbrace{\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})}_{\text{prediction}} \quad \text{(by the sensor Markov assumption).}\end{aligned}$$

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \\ &= \alpha\, \underbrace{\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{x}_t} \underbrace{\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)}_{\text{transition model}}\, \underbrace{P(\mathbf{x}_t \mid \mathbf{e}_{1:t})}_{\text{recursion}} \quad \text{(Markov assumption).}\end{aligned}$$

Inference in Temporal Models

- Smoothing

$$\begin{aligned}\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule, given } \mathbf{e}_{1:k}) \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\, \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)} \\ &= \alpha\, \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}\,.\end{aligned}$$

- where "×" represents pointwise multiplication of vectors.
- backward message $\mathbf{b}_{k+1:t}$ can be computed by recursive process that runs backward from t

$$\begin{aligned}\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(conditioning on } \mathbf{X}_{k+1}) \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(by conditional independence)} \\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\, \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \\ &= \sum_{\mathbf{x}_{k+1}} \underbrace{P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})}_{\text{sensor model}}\, \underbrace{P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})}_{\text{recursion}}\, \underbrace{\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)}_{\text{transition model}}\,,\end{aligned}$$

Inference in Temporal Models

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 $i = t$ **down to** 1 **do**
    sv$[i] \leftarrow$ NORMALIZE(**fv**$[i] \times$ **b**)
    b $\leftarrow$ BACKWARD(**b**, **ev**$[i]$)
  return sv

The forward–backward algorithm for smoothing: computing posterior probabilities of a sequence of states given a sequence of observations

Inference in Temporal Models

Finding the most likely sequence

- There is a linear-time algorithm for finding the most likely sequence
- It relies on the same Markov property that yielded efficient algorithms for filtering and smoothing
- view each sequence as a path through a graph whose nodes are the possible states at each time step.
- likelihood of any path is the product of the transition probabilities along the path and the probabilities of the given observations at each state
- there is a recursive relationship between most likely paths to each state $\mathbf{x}_{t+1}$ and most likely paths to each state $\mathbf{x}_t$

Inference in Temporal Models

Finding the most likely sequence

- Recursively computed message $m_{1:t}$

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

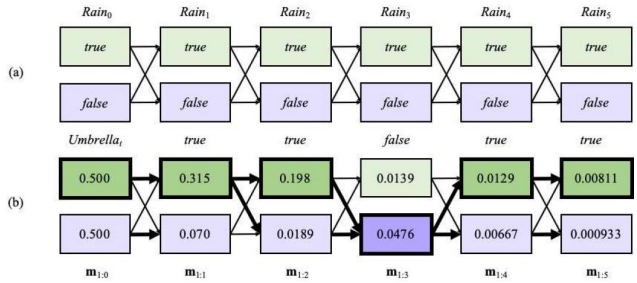
$$\begin{aligned}\mathbf{m}_{1:t+1} &= \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t+1}) = \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}, e_{t+1}) \\ &= \max_{\mathbf{x}_{1:t}} \mathbf{P}(e_{t+1} \mid \mathbf{x}_{1:t},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

- $\mathbf{m}_{1:t}$ will contain the probability for the most likely sequence reaching *each* of the final states.

Viterbi algorithm:

- select the final state of the most likely sequence overall. In order to identify the actual sequence, as opposed to just computing its probability, the algorithm will also need to record, for each state, the best state that leads to it

Inference in Temporal Models



(a) Possible state sequences for *Rain* can be viewed as paths through a graph of the possible states at each time step.
(b) Operation of the Viterbi algorithm for the umbrella observation sequence [*true*; *true*; *false*; *true*; *true*], where the evidence starts at time 1.

Hidden Markov Models

- Hidden Markov model, or HMM is a temporal probabilistic model in which the state of the process is described by a single, discrete random variable
- No restriction on the evidence variables. There can be many evidence variables, both discrete and continuous

Simplified matrix algorithms

transition model $\mathbf{P}(X_t \mid X_{t-1})$ becomes an $S \times S$ matrix $\mathbf{T}$ where:

$$T_{ij} = P(X_t = j \mid X_{t-1} = i)$$

T_{ij} is the probability of a transition from state i to state j.

HMMs, the matrix formulation reveals opportunities for improved algorithms
- simple variation on the forward–backward algorithm that allows smoothing to be carried out in constant space, independently of the length of the sequence
- Online smoothing with a fixed lag.

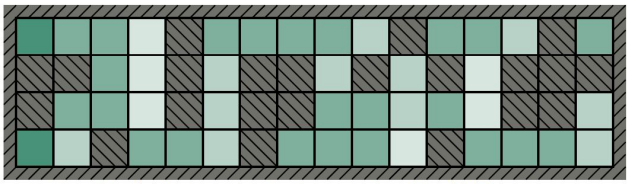
Hidden Markov Models

function FIXED-LAG-SMOOTHING(e_t, hmm, d) **returns** a distribution over $\mathbf{X}_{t-d}$
  inputs: e_t, the current evidence for time step t
    hmm, a hidden Markov model with $S \times S$ transition matrix $\mathbf{T}$
    d, the length of the lag for smoothing
  persistent: t, the current time, initially 1
    $\mathbf{f}$, the forward message $\mathbf{P}(X_t \mid e_{1:t})$, initially hmm.PRIOR
    $\mathbf{B}$, the d-step backward transformation matrix, initially the identity matrix
    $e_{t-d:t}$, double-ended list of evidence from $t-d$ to t, initially empty
  local variables: $\mathbf{O}_{t-d}$, $\mathbf{O}_t$, diagonal matrices containing the sensor model information

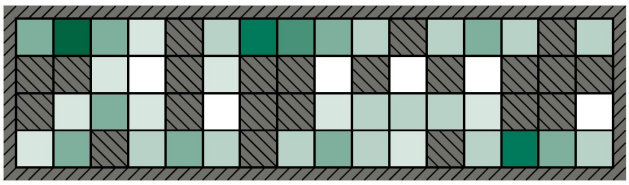
  add e_t to the end of $e_{t-d:t}$
  $\mathbf{O}_t \leftarrow$ diagonal matrix containing $\mathbf{P}(e_t \mid X_t)$
  if $t > d$ **then**
    $\mathbf{f} \leftarrow$ FORWARD($\mathbf{f}, e_{t-d}$)
    remove e_{t-d-1} from the beginning of $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d} \mid X_{t-d})$
    $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t+1$
  if $t > d+1$ **then return** NORMALIZE($\mathbf{f} \times \mathbf{B}1$) **else return** null

An algorithm for smoothing with a fixed time lag of d steps, implemented as an online algorithm that outputs the new smoothed estimate given the observation for a new time step.

Hidden Markov Models

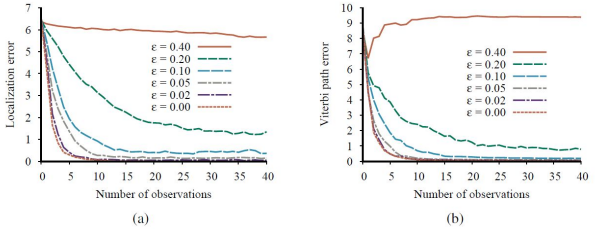


(a) Posterior distribution over robot location after $E_1 = 1011$



(b) Posterior distribution over robot location after $E_1 = 1011$, $E_2 = 1010$

Hidden Markov Models



Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs.
(a) The localization error, defined as the Manhattan distance from the true location.
(b) The Viterbi path error, defined as the average Manhattan distance of states on the Viterbi path from corresponding states on the true path

Kalman Filters

- handling continuous variables
- current distribution $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ is Gaussian and the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$ is linear–Gaussian, then the one-step predicted distribution given by

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

- If the prediction $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ is Gaussian and the sensor model $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})$ is linear– Gaussian, then, after conditioning on the new evidence, the updated distribution

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

Thus, the FORWARD operator for Kalman filtering takes a Gaussian forward message $\mathbf{f}_{1:t}$, specified by a mean μ_t and covariance Σ_t, and produces a new multivariate Gaussian forward message $\mathbf{f}_{1:t+1}$, specified by a mean μ_{t+1} and covariance Σ_{t+1}.

Kalman Filters

full multivariate Gaussian distribution

$$\mathcal{N}(\mathbf{x};\mu,\Sigma) = \alpha e^{-\frac{1}{2}\left((\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right)}.$$

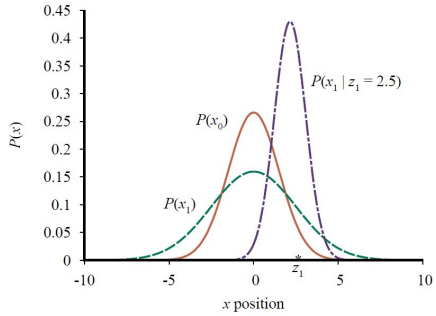
Mean And Covariance

$$\begin{aligned}\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\ \Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x),\end{aligned}$$

Kalman Gain Matrix

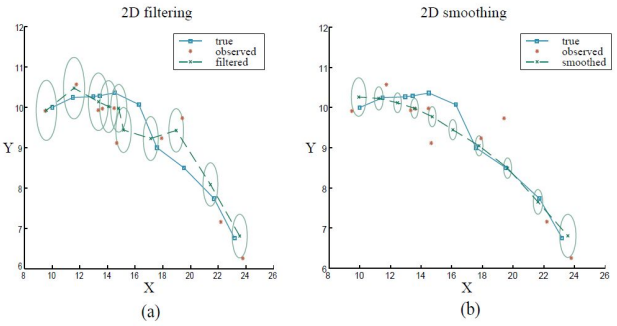
$$\mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}$$

Kalman Filters



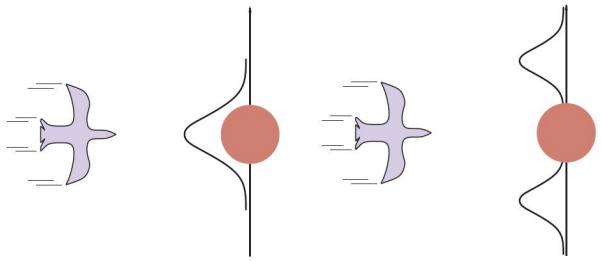
Stages in the Kalman filter update cycle for a random walk with a prior given by $\mu_0 = 0.0$ and $\sigma_0 = 1.5$, transition noise given by $\sigma_x = 2.0$, sensor noise given by $\sigma_z = 1.0$, and a first observation $z_1 = 2.5$ (marked on the x-axis).

Kalman Filters



(a) Results of Kalman filtering for an object moving on the $X-Y$ plane, showing the true trajectory (left to right), a series of noisy observations, and the trajectory estimated by Kalman filtering. Variance in the position estimate is indicated by the ovals.
(b) The results of Kalman smoothing for the same observation sequence.

Kalman Filters



A bird flying toward a tree (top views).
(a) A Kalman filter will predict the location of the bird using a single Gaussian centered on the obstacle.
(b) A more realistic model allows for the bird's evasive action, predicting that it will fly to one side or the other

Dynamic Bayesian Networks

- **Dynamic Bayesian networks**, or **DBNs**, extend the semantics of standard Bayesian networks to handle temporal probability models
- Each slice of a DBN can have any number of state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$
- every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable
- Every HMM is a DBN and every DBN can be translated into an HMM
- By decomposing the state of a complex system into its constituent variables, we can take advantage of **sparseness in the temporal probability model.**
- In a Kalman filter, the current state distribution is always a single multivariate Gaussian distribution. DBNs can model **arbitrary distributions**

Dynamic Bayesian Networks

- **Dynamic Bayesian networks**, or **DBNs**, extend the semantics of standard Bayesian networks to handle temporal probability models
- Each slice of a DBN can have any number of state variables $\mathbf{X}_t$ and evidence variables $\mathbf{E}_t$
- DBN representation size $O(nd^k)$ if the number of parents of each variable is bounded by k.
- every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable
- Every HMM is a DBN and every DBN can be translated into an HMM
- By decomposing the state of a complex system into its constituent variables, we can take advantage of **sparseness in the temporal probability model.**
- In a Kalman filter, the current state distribution is always a single multivariate Gaussian distribution. DBNs can model **arbitrary distributions**

Dynamic Bayesian Networks

- **To construct a DBN, one must specify three kinds of information:**
  - the prior distribution over the state variables, $\mathbf{P}(\mathbf{X}_0)$
  - the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t)$
  - sensor model $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$.
- **Transient failure:** the sensor occasionally decides to send some nonsense
- for a system to handle sensor failure properly, the sensor model must include the possibility of failure
- The simplest kind of failure model for a sensor allows a certain probability that the sensor will return some completely incorrect value, regardless of the true state of the world
- **Persistent failure model**: describes how the sensor behaves under normal conditions and after failure
- This persistence arc has a CPT that gives a small probability of failure in any given time step,

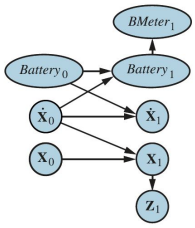
Dynamic Bayesian Networks

Example:
monitoring a battery-powered robot moving in the X–Y plane

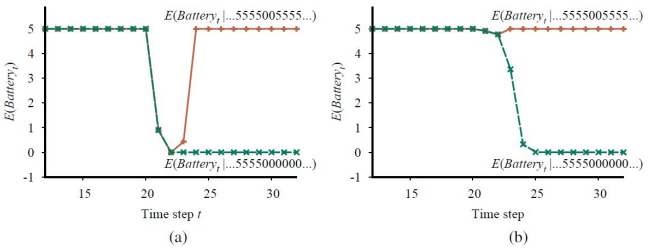
State variables, which will include both $\mathbf{X}_t = (X_t, Y_t)$ for position and $\dot{\mathbf{X}}_t = (\dot{X}_t, \dot{Y}_t)$ for velocity.

assume some method of measuring position yielding measurements $\mathbf{Z}_t$

$Battery_t$: battery level
$BMeter_t$: measures the battery charge level

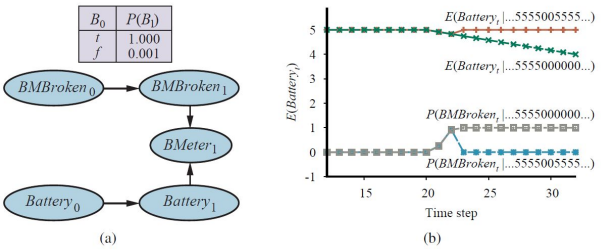


Dynamic Bayesian Networks



(a) Upper curve: trajectory of the expected value of $Battery_t$
(b) The same experiment run with the transient failure model

Dynamic Bayesian Networks



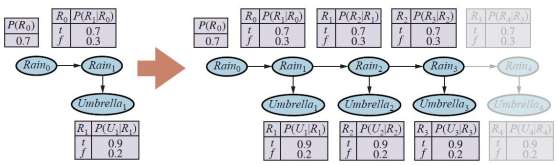
(a) A DBN fragment showing the sensor status variable required for modeling persistent failure of the battery sensor.
(b) Upper curves: trajectories of the expected value of $Battery_t$ for the "transient failure" and "permanent failure" observations sequences. Lower curves: probability trajectories for *BMBroken* given the two observation sequences.

Dynamic Bayesian Networks

Exact inference in DBNs

- **Unrolling**: construct the full Bayesian network representation of a DBN by replicating slices until the network is large enough to accommodate the observations
- Once the DBN is unrolled, one can use any of the inference algorithms—variable elimination, clustering methods
- the filtering update: works by summing out the state variables of the previous time step to get the distribution for the new time step (variable elimination algorithm)
- The maximum factor size is $O(d^{n+k})$ and the total update cost per step is $O(nd^{n+k})$, where d is the domain size of the variables and k is the maximum number of parents of any state variable.
- even though we can use DBNs to represent very complex temporal processes with many sparsely connected variables we cannot reason efficiently and exactly about those processes.

Dynamic Bayesian Networks



Unrolling a dynamic Bayesian network

Dynamic Bayesian Networks

Approximate inference in DBNs

Sequential importance sampling or SIS
- use the samples themselves as an approximate representation of the current state distribution.
- "constant" time per update
- No need to unroll the DBN
- to maintain a given level of accuracy, we need to increase the number of samples exponentially with t
- Even with 100,000 samples, the SIS approximation fails completely after about 20 steps

Particle filtering
- focus the set of samples on the high-probability regions of the state space
- sequential importance sampling with resampling

Dynamic Bayesian Networks

Particle filtering
- $N(\mathbf{x}_t \mid \mathbf{e}_{1:t})$: the number of samples occupying state $\mathbf{x}_t$ after observations $\mathbf{e}_{1:t}$ have been processed, we therefore have

$$N(\mathbf{x}_t \mid \mathbf{e}_{1:t})/N = P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

- propagate each sample forward by sampling the state variables at $t+1$,

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

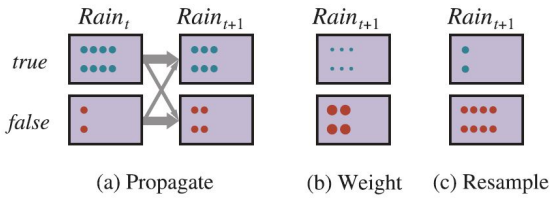
Dynamic Bayesian Networks

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 defined by $\mathbf{P}(\mathbf{X}_0)$, $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0)$, and $\mathbf{P}(\mathbf{E}_1 \mid \mathbf{X}_1)$
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

The **particle filtering algorithm** implemented as a recursive update operation with state (the set of samples). Each of the sampling operations involves sampling the relevant slice variables in topological order, much as in PRIOR-SAMPLE. The WEIGHTED-SAMPLE-WITH-REPLACEMENT operation can be implemented to run in $O(N)$ expected time.

Dynamic Bayesian Networks



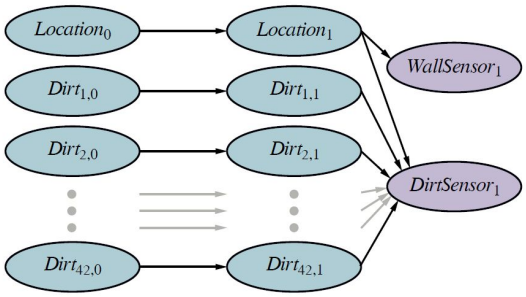
The particle filtering update cycle for the umbrella DBN with N=10, showing the sample populations of each state.

Dynamic Bayesian Networks

The **particle filtering algorithm** weakness:
- Possible the initial guesses in each particle are never updated by the evidence.
- Best particles will come to dominate the total likelihood as time progresses and the diversity of the population of particles will collapse.
- all the particles agree on a single, incorrect map, the algorithm becomes convinced that that map is correct and never changes its mind

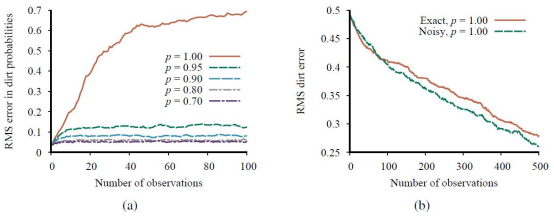
Rao-Blackwellization particle filter
- Exact inference is always more accurate than sampling, even if it's only for a subset of the variables
- For the SLAM problem, we run particle filtering on the robot location and then, for each particle, we run exact HMM inference for each dirt square independently, conditioned on the location sequence in that particle

Dynamic Bayesian Networks



- A dynamic Bayes net for simultaneous localization and mapping in the stochastic-dirt vacuum world. Dirty squares persist with probability p, and clean squares become dirty with probability $1-p$. The local dirt sensor is 90% accurate, for the square in which the robot is currently located.

Dynamic Bayesian Networks



(a) Performance of the standard particle filtering algorithm with 1,000 particles, showing RMS error in marginal dirt probabilities compared to exact inference for different values of the dirt persistence p.
(b) Performance of Rao-Blackwellized particle filtering (100 particles) compared to ground truth, for both exact location sensing and noisy wall sensing and with deterministic dirt. Data averaged over 20 runs.

Summary

- The changing state of the world is handled by using a set of random variables to represent the state at each point in time.
- Representations can be designed to (roughly) satisfy the Markov property, so that the future is independent of the past given the present. Combined with the assumption that the process is time-homogeneous, this greatly simplifies the representation.
- A temporal probability model can be thought of as containing a transition model describing the state evolution and a sensor model describing the observation process.
- The principal inference tasks in temporal models are filtering (state estimation), prediction, smoothing, and computing the most likely explanation.
- In practice, the particle filtering algorithm and its descendants are an effective family of approximation algorithms