Chap. 15

Temporal Probability Reasoning

In which we try to interpret the present, understand the past, and perhaps predict the future, under the uncertainty.

Outline

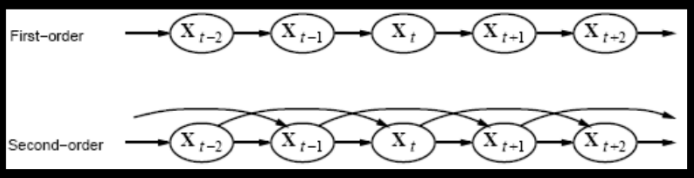
- Time and uncertainty
- Inference: filtering, prediction, smoothing
- Hidden Markov Models
- Kalman Filters
- Dynamic Bayesian Networks
- Particle Filtering

Time and uncertainty

- The world changes; we need to track and predict it.
- Diabetes management vs. vehicle diagnosis
- Basic idea:
  copy state and evidence variables for each time step
- X_t = set of *unobservable* state variables at time ***t***
  e.g.) $BloodSugarLevel_t$, $StomachContents_t$, etc.
- E_t = set of *observable evidence* variables at time *t*
  e.g.) $MeasurerdBloodSugar_t$, $PulseRate_t$, $FoodEaten_t$.
- This assumes discrete time; step size depends on problem
- Notation: $X_{a:b} = X_a, X_{a+1}, \ldots, X_{b-1}, X_b$.

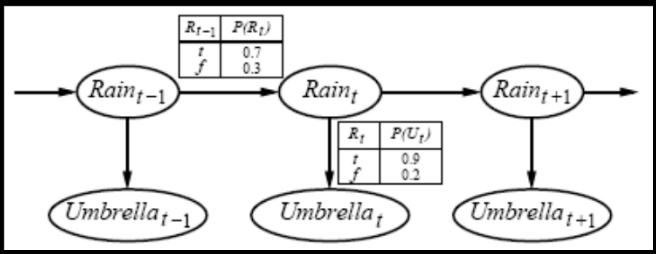
Markov processes (Markov Chains)

- Construct a Bayes net from these variables: parents?
- X_t: (belief) *state variable* at *t*, E_t: observable *evidence variable* at *t*.
- **Markov assumption**: X_t depends on bounded subset of $X_{0:t-1}$ (*finite history* of previous states)
  - **1st -order Markov process**: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$
  - **2nd -order Markov process**: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-2}, X_{t-1})$



- **Sensor Markov assumption**: $P(E_t \mid X_{0:t}, E_{0:t-1}) = P(E_t \mid X_t)$
  - **Stationary** process: *Transition Model* $P(X_t \mid X_{t-1})$ and *Sensor Model* (or observational model) $P(E_t \mid X_t)$ fixed for all *t*.
- $P(X_0, X_1, \ldots, X_t, E_1, \ldots E_t) = P(X_0) \prod_{i=1..t} P(X_i \mid X_{i-1}) P(E_i \mid X_i)$

Example



- 1st-order Markov assumption not exactly true in real world!
- Possible fixes:
  1. Increase order of Markov process
  2. Increase the set of state variables, e.g., add $Temp_t$, $Pressure_t$, $Season_t$
     - increase the prediction requirements as well as improve the system's prediction power. – need to predict the new variables.
- Example: robot motion.
  Augment position and velocity with $Battery_t$.

Inference tasks

- **Filtering (monitoring, state estimation)**: $P(X_t \mid e_{1:t})$
  belief state -- input to the decision process of a rational agent
- **Prediction**: $P(X_{t+k} \mid e_{1:t})$ for $k>0$
  future state -- evaluation of possible action sequences;
  like filtering without the new evidence
- **Smoothing (hindsight)**: $P(X_k \mid e_{1:t})$ for $0 \leq k < t$
  better estimate of past states, essential for learning
- **Most likely explanation**: $\arg\max_{X1:t} P(X_{1:t} \mid e_{1:t})$
  the sequence of states that is most likely to have generated those observations.
  speech recognition, decoding with a noisy channel
- **Learning**:
  learning of the transition/sensor models from observation.
  Use the new estimates to update the models.

Filtering

- Maintain a current state estimate and update it.
- Aim: devise a recursive state estimation algorithm:

$P(X_{t+1} \mid e_{1:t+1}) = f(e_{t+1}, P(X_t \mid e_{1:t}))$ - given the result of filtering up to t.

$P(X_{t+1} \mid e_{1:t+1}) = P(X_{t+1} \mid e_{1:t}, e_{t+1})$: dividing up the evidence

$= \alpha\, P(e_{t+1} \mid X_{t+1}, e_{1:t})\, P(X_{t+1} \mid e_{1:t})$: using Bayes' rule

$= \alpha\, P(e_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid e_{1:t})$: by the Markov sensor assumption

i.e. prediction + estimation. Prediction by summing out X_t:

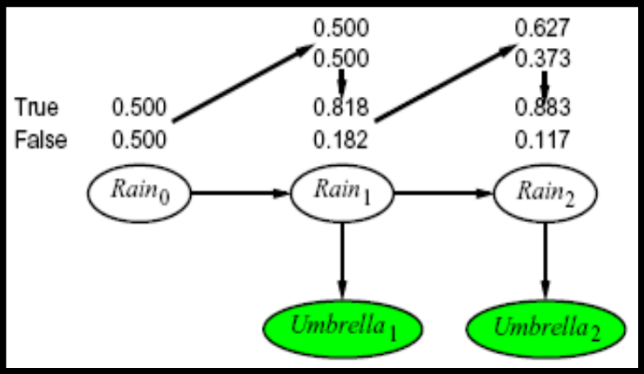
$P(X_{t+1} \mid e_{1:t+1}) = \alpha\, P(e_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t, e_{1:t})\, P(x_t \mid e_{1:t})$

$= \alpha\, P(e_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t)\, P(x_t \mid e_{1:t})$

$f_{1:t+1} = \alpha \text{FORWARD}(f_{1:t}, e_{t+1})$ where $f_{1:t} = P(X_t \mid e_{1:t})$ $f_{1:0} = P(X_0)$

- Time and space requirements for updating are O(1) : independent of t.

Filtering example



Prediction

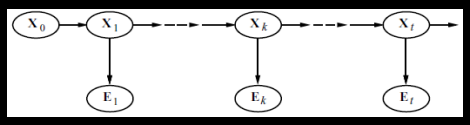
- Filtering without the addition of new evidence.

  $P(X_{t+k+1} \mid e_{1:t+1}) = \sum_{x_{t+k}} P(X_{t+k+1} \mid x_{t+k})\, P(x_{t+k} \mid e_{1:t})$

  - transition model only
- The predicted distribution converges to a fixed point – the stationary distribution of the Markov process defined by the transition model.
- Mixing time: the time taken to reach the fixed point.
- Fail to predict the actual state for a number of steps > the mixing time.
- The more uncertainty there is in the transition model, the shorter will be the mixing time and the more the future is obscured.
- Use a forward recursion to compute the *likelihood of the evidence sequence*, $P(e_{1:t})$. – useful for comparison of different temporal models with same evidence sequence.
  - $l_{1:t+1} = FORWARD\,(l_{1:t}, e_{1:t+1})$ where $l_{1:t} = P(X_t, e_{1:t})$
  - So, $l_{1:t} = P(e_{1:t}) = \sum_{x_t} l_{1:t}(x_t)$: the actual likelihood

Smoothing

- The process of computing the distribution over *past states* given evidence up to the present: $P(X_k \mid e_{1:t})$ for $0 \le k < t$



- Divide evidence $e_{1:t}$ into $e_{1:k}$, $e_{k+1:t}$:

  $P(X_k|e_{1:t}) = P(X_t| e_{1:k}, e_{k+1:t})$: dividing up the evidence

  $= \alpha\, P(X_k |e_{1:k})\, P(e_{k+1:t}|X_k, e_{1:k})$: using Bayes' rule

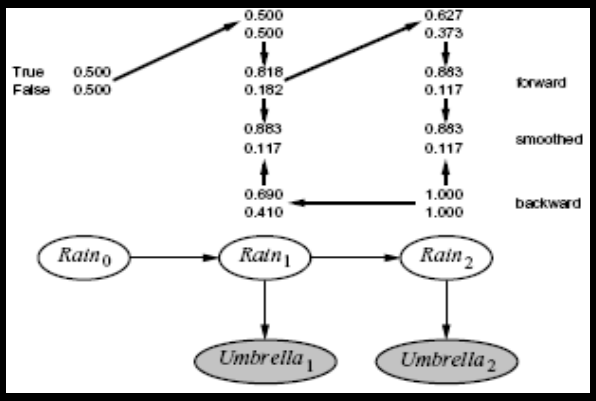
  $= \alpha\, P(X_k |e_{1:k})\, P(e_{k+1:t}|X_k)$: using conditional independence

  $= \alpha\, f_{1:k}\, b_{k+1:t}$
- Backward message computed by a backwards recursion:

$$\begin{aligned} P(e_{k+1:t}|X_k) &= \sum_{x_{k+1}} P(e_{k+1:t}|X_k, x_{k+1}) P(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1}) P(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1}) P(e_{k+2:t}|x_{k+1}) P(x_{k+1}|X_k) \end{aligned}$$

$b_{k+1:t} = \text{BACKWARD}(b_{k+2:t}, e_{k+1:t})$ where $b_{k+2:t} = P(e_{k+2:t} \mid x_{k+1})$

Smoothing



- **Forward-backward** algorithm: cache forward messages along the way
  $P(X_k|e_{1:t}) = \alpha\, f_{1:k}\, b_{k+1:t}$: the smoothed estimate at *k* from $b_{k+1:t}$ and the stored $f_{1:k}$

Smoothing

- Forward-backward algorithm: cache forward messages along the way
- Time *O(t)* : polytree inference with each forward/backward recursion of O(1).
- Space *O (t·|f|)* where *|f|* is the size of representation of forward message.

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
  inputs: ev, a vector of evidence values for steps 1, ..., t
          prior, the prior distribution on the initial state, P(X_0)
  local variables: fv, a vector of forward messages for steps 0, ..., t
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps 1, ..., t

  fv[0] ← prior
  for i = 1 to t do
      fv[i] ← FORWARD(fv[i − 1], ev[i])
  for i = t downto 1 do
      sv[i] ← NORMALIZE(fv[i] × b)
      b ← BACKWARD(b, ev[i])
  return sv
```

Smoothing

- Drawbacks:
  - a high space complexity when the state space is large and the sequences are long.
  - The need of modification for an online setting where smoothed estimates must be computed for earlier time slices as new observations are continuously added to the end of the sequence.
- The requirement is for *fixed-lag smoothing*, which requires computing the smoothed estimate $P(X_{t-d}|e_{1:t})$ for fixed d. -- i.e. smoothing is done for the time slice d steps behind the current time t; as t increases, the smoothing has to keep up. Run the forward-backward algorithm over the d-step window as each new observation is needed -- inefficient => modified fixed-lag smoothing algorithm in O(1) per update, independent of d.

Most likely explanation

- Most likely sequence (of hidden states) $\neq$ sequence of most likely states!
  (need to consider joint probabilities over all the time steps
  vs. posterior distribution over single time step)
- Mostly likely (state) sequence
  $Q^* = \text{argmax}_Q P(X_{1:t} \mid e_{1:t})$ where $Q = (X_1, X_2, \ldots, X_t) = X_{1:t}$
- Sequence of mostly likely states
  $(X_1^*, X_2^*, \ldots, X_t^*)$ where $X_i^* = \text{argmax}_{Xi} P(X_i \mid e_{1:l})$
- Cf) Learning:
  learning of the transition/sensor models from observation.
  Given an observational sequence $e_{1:t}$, find the optimal parameters λ^* in transition model or in sensor model s.t.
  $\lambda^* = \text{argmax}_\lambda P(E_t \mid X_t, \lambda)$ or $\text{argmax}_\lambda P(X_{t+1} \mid X_t, \lambda)$

Most likely explanation

- Most likely sequence (of hidden states) ≠ sequence of most likely states!

(need to consider joint probabilities over all the time steps vs. posterior distribution over single time step)

- Most likely path to each x_{t+1}

  = most likely path to some x_t plus one more step

$$\max_{x_1 \ldots x_t} \mathbf{P}(x_1, \ldots, x_t, \mathbf{X}_{t+1} | \mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{e}_{t+1} | \mathbf{X}_{t+1}) \max_{x_t} \left(\mathbf{P}(\mathbf{X}_{t+1} | x_t) \max_{x_1 \ldots x_{t-1}} P(x_1, \ldots, x_{t-1}, x_t | \mathbf{e}_{1:t}) \right)$$

  i.e. a recursive relationship b/t most likely paths to each state x_{t+1} and most likely paths to each state x_t.

- Identical to filtering, except $f_{1:t}$ replaced by

$$\mathbf{m}_{1:t} = \max_{x_1 \ldots x_{t-1}} \mathbf{P}(x_1, \ldots, x_{t-1}, \mathbf{X}_t | \mathbf{e}_{1:t})$$

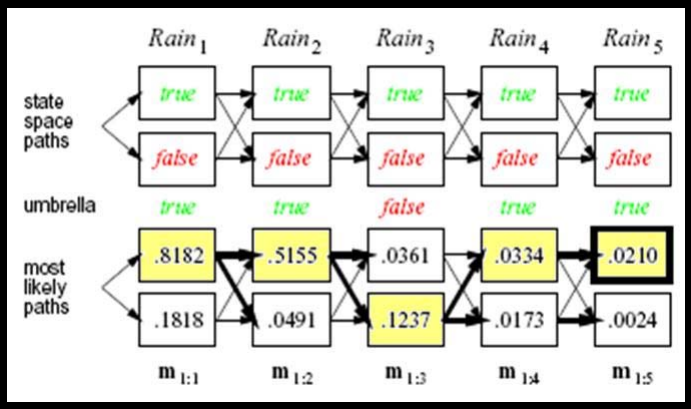
  i.e. $\mathbf{m}_{1:t}$ gives *the probability of the most likely path to state* x_t.

- Update has sum replaced by max, giving the Viterbi algorithm:

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

- Time *O(t)*, (*t* is the length of sequence), Space *O(t)* because of keeping the pointers.

Viterbi Path Example



Hidden Markov Models (HMM)

- A temporal probabilistic model in which the state of the process, X_t is a *single*, *discrete* variable (usually E_t is too)
- Domain of X_t is $\{1, \ldots, S\}$: a value of the variable is a state of the world.
- Transition matrix (S×S): $T_{ij} = P(X_t = j \mid X_{t-1}=i)$, e.g. $\begin{matrix} & P(x_t|x_{t-1}) & P(\sim x_t|x_{t-1}) \\ x_{t-1} & 0.7 & 0.3 \\ \sim x_{t-1} & 0.3 & 0.7 \end{matrix}$
  S ×S matrix, a prob. of transition from state *i* to state *j*.
- Sensor matrix (S×S): O_t for each time step, *diagonal* elements $P(e_t| X_t=i)$, e.g., with $U_1 = true$, $O_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$: the probability that e_t appears.
  cf) $U_3 = false$, $O_3 = ?$
- Forward and backward messages as column vectors:
  $\mathbf{f}_{1:t+1} = \alpha\, \mathbf{O}_{t+1}\mathbf{T}^T\mathbf{f}_{1::t}$ **
  $\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}$
- Forward-backward algorithm needs time $O(S^2t)$ and space $O(St)$

Country Dance Algorithm - Improved Forward-backward alg.

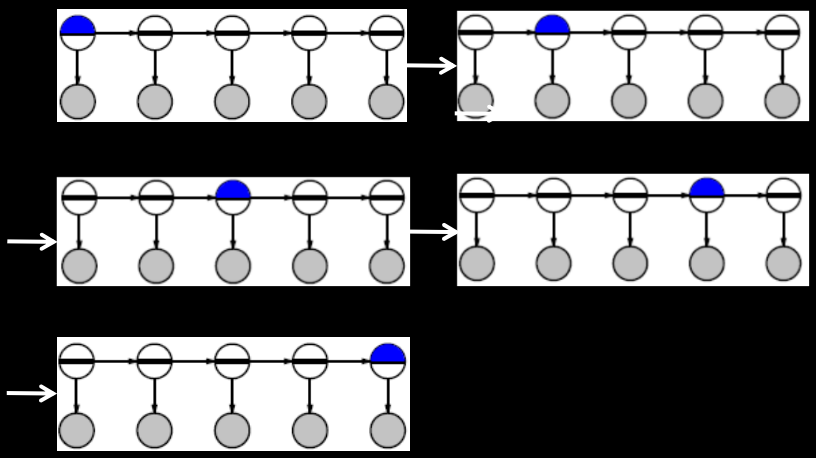
- Can avoid storing all forward messages in smoothing by running forward algorithm backwards:

$$\mathbf{f}_{1:t+1} = \alpha\, \mathbf{O}_{t+1}\mathbf{T}^T\mathbf{f}_{1::t}$$
$$\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} = \alpha\, \mathbf{T}^T\mathbf{f}_{1::t}$$
$$\alpha'\, (\mathbf{T}^T)^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1} = \mathbf{f}_{1::t}$$

  - Algorithm: forward pass computes f_t , backward pass does f_i , b_i
  - Requirement: T is invertible and a sensor model O have no zeros.
- Improvements of forward-backward algorithm:
  - 1. Space complexity: O(1)
    – constant because only one copy of each message is needed.

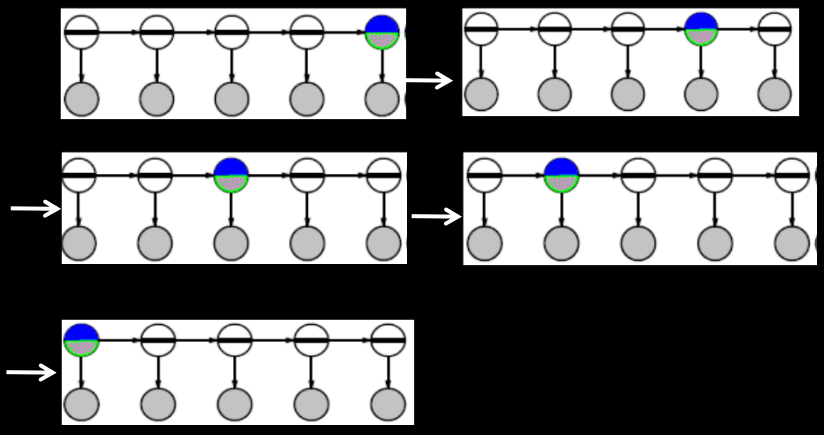
Country dance algorithm

- Algorithm: forward pass computes f_t , backward pass does f_i , b_i



Country dance algorithm

- Algorithm: forward pass computes f_t , backward pass does f_i , b_i



Fixed-Lag Smoothing algorithm

- Improvements of forward-backward algorithm:
  - 2. An efficient online smoothing with a fixed lag whose time complexity is O(1), independent of the length of the lag(d).
- Fixed-Lag Smoothing Algorithm
  - Need to compute $\alpha \mathbf{f}_{1:t-d}\mathbf{b}_{t-d+1:t}$ for slice *t-d*.
  - Also need to compute $\alpha \mathbf{f}_{1:t-d+1}\mathbf{b}_{t-d+2:t+1}$ for slice *t-d+1* when a new observation arrives.
  - First, compute $\mathbf{f}_{1:t-d+1}$ from using $\mathbf{f}_{1:t-d}$ the standard filtering process.
  - To compute the backward message incrementally, apply $\mathbf{b}_{k+1:t} = \mathbf{TO}_{k+1}\mathbf{b}_{k+2:t}$ *d* times

$$\mathbf{b}_{t-d+1:t} = \left(\prod_{i=t-d+1}^{t} \mathbf{TO}_i\right)\mathbf{b}_{t+1:t} = \mathbf{B}_{t-d+1:t}\mathbf{1} \qquad \mathbf{b}_{t-d+2:t+1} = \left(\prod_{i=t-d+2}^{t+1} \mathbf{TO}_i\right)\mathbf{b}_{t+2:t+1} = \mathbf{B}_{t-d+2:t+1}\mathbf{1}$$

  where $\mathbf{B}_{t-d+1:t}$ is the product of seq. of T & O: a transformation operator that transforms a later backward message into an earlier one.
  - $\mathbf{B}_{t-d+2:t+1} = \mathbf{O}^{-1}_{t-d+1}\mathbf{T}^{-1}\mathbf{B}_{t-d+1:t}\mathbf{TO}_{t+1}$
  - Thus, $\mathbf{b}_{t-d+2:t+1} = \mathbf{B}_{t-d+2:t+1}\mathbf{1} = \mathbf{O}^{-1}_{t-d+1}\mathbf{T}^{-1}\mathbf{B}_{t-d+1:t}\mathbf{TO}_{t+1}\mathbf{1}$

Fixed-Lag Smoothing algorithm

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 $\mathbf{P}(X_t|e_{1:t})$, 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 $\mathbf{P}(e_t|X_t)$
  if $t > d$ **then**
    $\mathbf{f} \leftarrow$ FORWARD($\mathbf{f}, e_t$)
    remove e_{t-d-1} from the beginning of $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d}|X_{t-d})$
    $\mathbf{B} \leftarrow \mathbf{O}^{-1}_{t-d}\mathbf{T}^{-1}\mathbf{BTO}_t$
  else $\mathbf{B} \leftarrow \mathbf{BTO}_t$
  $t \leftarrow t+1$
  if $t > d$ **then return** NORMALIZE($\mathbf{f} \times \mathbf{B1}$) **else return** null

HMM Example : Localization

- The localization problem for the vacuum world
- A single *nondeterministic* Move action
- Sensor's report: obstacle to N/S/E/W
- The robot's belief state is the set of possible locations it could be in.
- X_t : location of the robot on the discrete grid $\{s_1, s_2, \ldots, s_n\}$
- NEIGHBORS(s): the set of adjacent empty squares to s whose size is $N(s)$.
- Transition model: the robot is likely to be at any neighboring square.

$$P(X_{t+1} = j \mid X_t = i) = \boldsymbol{T}_{ij} = \begin{cases} 1/N(i) & \text{if } j \in \text{NEIGHBORS}(i) \\ 0 & \text{else} \end{cases}$$

where $P(X_0 = i) = 1/n$.

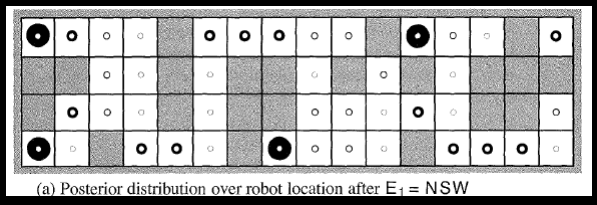
- Sensor model: E_t = (obstacle at N, obstacle at S, obstacle at E, obstacle at W)
  Each sensor's error rate = ε. Errors occur independently for each direction.

$$P(E_t = e_t \mid X_t = i) = \mathbf{O}_{t_{ii}} = (1-\epsilon)^{4-d_{it}} \epsilon^{d_{it}}$$

  if d_{it} is the discrepancy b/t the true values for square i & actual reading e_t.
- Compute the posterior distribution over locations: i.e. where is the robot?

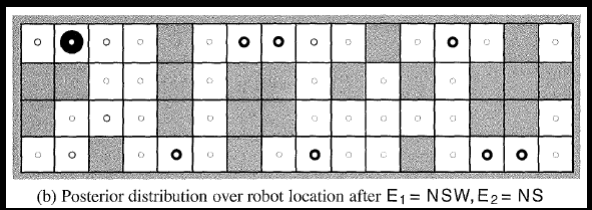
HMM Example : ..continued

- $P(E_t = NSE \mid X_t = NS) = (1-\varepsilon)^3 \cdot \varepsilon^1$
- $P(X_1 \mid E_1 = NSW) = ?$ with $\varepsilon = .2$



(a) Posterior distribution over robot location after E_1 = NSW

- $P(X_2 \mid E_1 = NSW, E_2 = NS) = ?$



(b) Posterior distribution over robot location after E_1 = NSW, E_2 = NS

HMM Example : .. continued

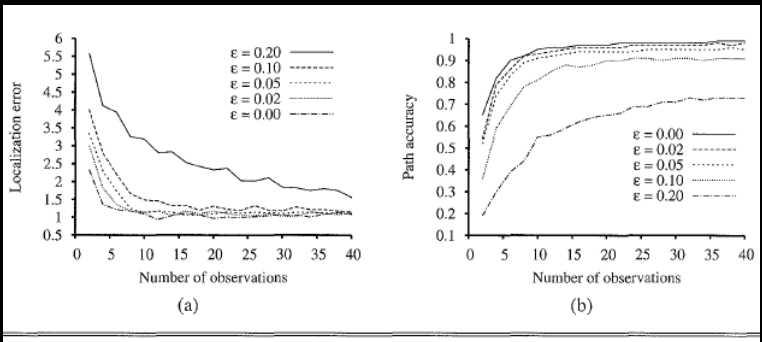
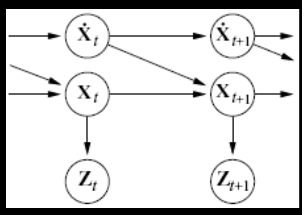


Figure 15.8 Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path accuracy, defined as the fraction of correct states on the Viterbi path.

Kalman filters

- Modeling systems described by a set of *continuous variables*,
  e.g., tracking a bird flying, Airplanes, robots, ecosystems, economies, chemical plants, planets, etc.
- Estimate the state of physical system from noisy observations over time.
  - An inference in temporal probability model where the *transition model* describes the physics of *motion* and the *sensor model* does the *measurement* process.
- The specification of bird's flight by 6 continuous variables:
  (X_t, Y_t, Z_t) for position & $(\dot{X}_t, \dot{Y}_t, \dot{Z}_t)$ for velocity.
  - Gaussian prior, *linear* Gaussian for transition and sensor model:

$$P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_t + \dot{x}_t \Delta, \sigma^2)(x_{t+\Delta})$$



Kalman filters

- The specification of bird's flight: (X_t, Y_t, Z_t) for position & $(\dot{X}_t, \dot{Y}_t, \dot{Z}_t)$ for velocity, Z_t for position measurement with noise.
  - Gaussian prior, *linear* Gaussian for transition and sensor model:
    - Gaussian Prior with noise (σ^2_0): $P(x_0) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}$
    - X_{t+1} is a linear f^n of X_t + Gaussian noise (σ^2_x):

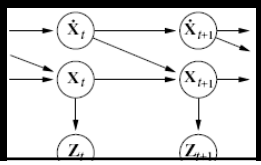
$$P(x_{t+1} \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}$$

    - Z_t is a linear f^n of X_t + Gaussian noise (σ_z^2):

$$P(z_t \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)}$$

    - Position update: $X_{t+\Delta} = X_t + \dot{X}\,\Delta$ where Δ is time interval b/t observations.

$$P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_t + \dot{x}_t\,\Delta, \sigma^2)(x_{t+\Delta})$$



Updating Gaussian distributions

- Prediction step: If $P(X_t|e_{1:t})$ is Gaussian and a linear Gaussian transition model, then prediction is Gaussian.

$$P(X_{t+1}|e_{1:t}) = \int_{x_t}^{\infty} P(X_{t+1}|x_t)P(x_t\,|e_{1:t})dx_t$$

- If $P(X_{t+1}|e_{1:t})$ is Gaussian and a linear Gaussian sensor model, then the updated distribution with e_{t+1} is Gaussian.

$$P(X_{t+1} \mid e_{1:t+1}) = \alpha\, P(e_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid e_{1:t})$$

- Hence, $P(X_t|e_{1:t})$ is multivariate Gaussian $N(\mu_t, \Sigma_t)$ for all t;
  $P(X_{t+1} \mid e_{1:t+1}) = f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1}) = N(\mu_{t+1}, \Sigma_{t+1})$
  where $f_{1:0} = P(X_0) = N(\mu_0, \Sigma_0)$.
- General (continuous or hybrid, nonlinear, non-Gaussian) process: state distribution grows unboundedly as $t \to \infty$

Updating Gaussian distributions: 1-D example

- Prior distribution: Gaussian with variance σ_0^2 :

$$P(x_0) = \alpha e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}$$

- A linear Gaussian transition model with variance σ_x^2 :

$$P(x_{t+1}|x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}$$

- A sensor model with Gaussian noise with variance σ_z^2 :

$$P(z_t|x_t) = \alpha e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)}$$

- 1-step predicted distribution:

$$P(x_1) = \int_{-\infty}^{\infty} P(x_1|x_0)\,P(x_0)dx_0 = \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2(x_1-x_0)^2+\sigma_x^2(x_0-\mu_0)^2}{\sigma_0^2\sigma_x^2}\right)} dx_0 = \alpha e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)}$$

- Update:

$$P(x_1|z_1) = \alpha P(z_1|x_1)P(x_1) = \alpha e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_x^2}\right)} \cdot e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)}$$

$$= \alpha e^{-\frac{1}{2}\left(\frac{(x_1-\frac{(\sigma_0^2+\sigma_x^2)z_1+\sigma_x^2\mu_0)^2}{\sigma_0^2+\sigma_x^2+\sigma_z^2}}{(\sigma_0^2+\sigma_x^2)\sigma_z^2/(\sigma_0^2+\sigma_x^2+\sigma_z^2)}\right)}.$$

-- a new Gaussian distribution variable.

The prior distribution is assumed to be Gaussian with variance σ_0^2:

$$P(x_0) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)}.$$

(For simplicity, we use the same symbol α for all normalizing constants in this section.) The transition model adds a Gaussian perturbation of constant variance σ_x^2 to the current state:

$$P(x_{t+1} \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)}.$$

The sensor model assumes Gaussian noise with variance σ_z^2:

$$P(z_t \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)}.$$

Now, given the prior $\mathbf{P}(X_0)$, the one-step predicted distribution comes from Equation (15.17):

$$P(x_1) = \int_{-\infty}^{\infty} P(x_1 \mid x_0)P(x_0)\,dx_0 = \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(x_1-x_0)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)} dx_0$$

$$= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2(x_1-x_0)^2+\sigma_x^2(x_0-\mu_0)^2}{\sigma_0^2\sigma_x^2}\right)} dx_0.$$

COMPLETING THE SQUARE

This integral looks rather complicated. The key to progress is to notice that the exponent is the sum of two expressions that are *quadratic* in x_0 and hence is itself a quadratic in x_0. A simple trick known as **completing the square** allows the rewriting of any quadratic $ax_0^2 + bx_0 + c$ as the sum of a squared term $a(x_0 - \frac{-b}{2a})^2$ and a residual term $c - \frac{b^2}{4a}$ that is independent of x_0. The residual term can be taken outside the integral, giving us

$$P(x_1) = \alpha\, e^{-\frac{1}{2}\left(c-\frac{b^2}{4a}\right)} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(a(x_0-\frac{-b}{2a})^2\right)} dx_0.$$

Now the integral is just the integral of a Gaussian over its full range, which is simply 1. Thus, we are left with only the residual term from the quadratic. Then, we notice that the residual term is a quadratic in x_1; in fact, after simplification, we obtain

$$P(x_1) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)}.$$

That is, the one-step predicted distribution is a Gaussian with the same mean μ_0 and a variance equal to the sum of the original variance σ_0^2 and the transition variance σ_x^2.

To complete the update step, we need to condition on the observation at the first time step, namely, z_1. From Equation (15.18), this is given by

$$P(x_1 \mid z_1) = \alpha P(z_1 \mid x_1)P(x_1)$$

$$= \alpha\, e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)}.$$

Once again, we combine the exponents and complete the square (Exercise 15.11), obtaining

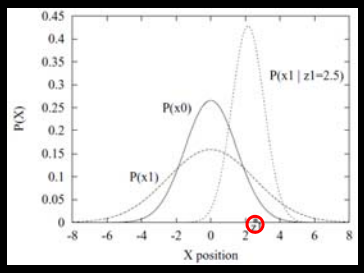
$$P(x_1 \mid z_1) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_1-\frac{(\sigma_0^2+\sigma_x^2)z_1+\sigma_x^2\mu_0)^2}{\sigma_0^2+\sigma_x^2+\sigma_z^2}}{(\sigma_0^2+\sigma_x^2)\sigma_z^2/(\sigma_0^2+\sigma_x^2+\sigma_z^2)}\right)}. \quad (15.19)$$

Cont.. Simple 1-D example

- Gaussian random walk on X_t –axis with a noisy observation Z_t, with variance σ^2_x, sensor variance σ^2_z, and σ^2_t

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2\mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \sigma^2_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$

- μ_{t+1} : a weighted average of the new observation z_{t+1} and the old mean μ_t .
- σ_{t+1} is independent of the observation.



$\mu_0 = 0$, σ_0=1.0, $\sigma_x = 2.0$, $\sigma_z = 1.0$, $z_1 = 2.5$. μ_1 < z1

General Kalman update

- The full multivariate Gaussian distribution: $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})(\mathbf{x}) = \alpha\, e^{-\frac{1}{2}((\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))}$
- Transition and models:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = N(\mathbf{F}\mathbf{x}_t, \boldsymbol{\Sigma}_x)(\mathbf{x}_{t+1})$$
$$P(\mathbf{z}_t|\mathbf{x}_t) = N(\mathbf{H}\mathbf{x}_t, \boldsymbol{\Sigma}_z)(\mathbf{z}_t)$$

- F is the matrix for the linear transition; Σ_x, the transition noise covariance
- H is the matrix for the sensors; Σ_z, sensor noise covariance
- Filter computes the following update:

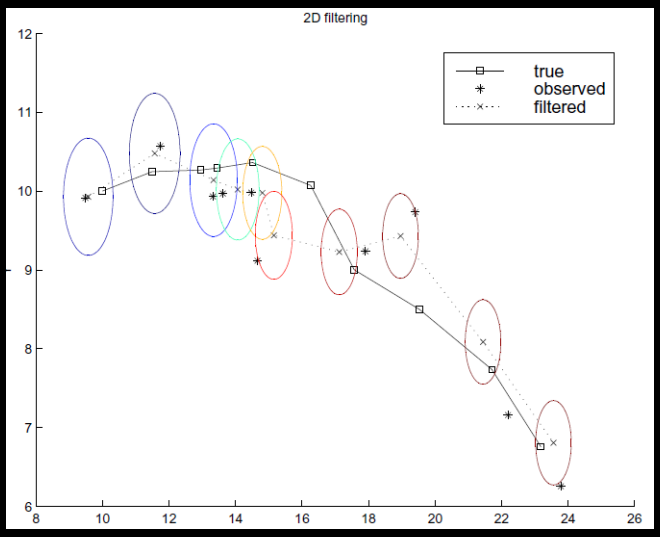
$$\boldsymbol{\mu}_{t+1} = \mathbf{F}\boldsymbol{\mu}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{HF}\boldsymbol{\mu}_t)$$
$$\boldsymbol{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)$$

where $\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top + \boldsymbol{\Sigma}_z)^{-1}$ is the Kalman gain matrix which is a measure of how seriously to take the new observation relative to the prediction;

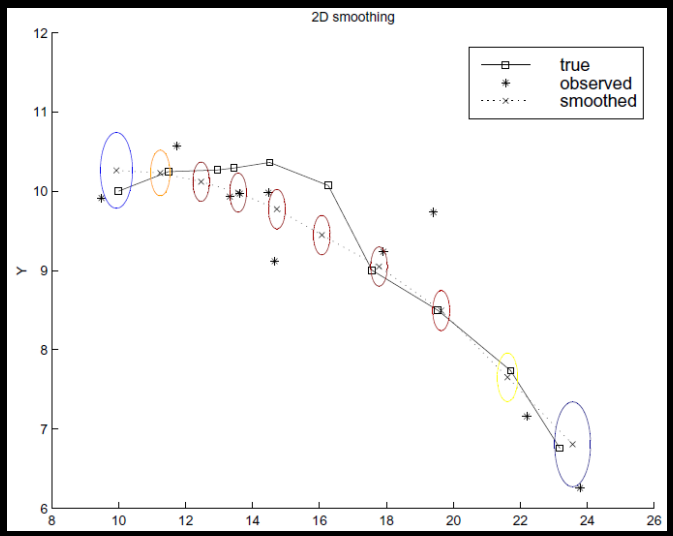
$\mathbf{F}\boldsymbol{\mu}_t$: the predicted state at t+1, $\mathbf{HF}\boldsymbol{\mu}_t$: the predicted observation, so z_{t+1}-$\mathbf{HF}\boldsymbol{\mu}_t$ the error in the predicted observation; $\mathbf{K}_{t+1}(z_{t+1}$-$\mathbf{HF}\boldsymbol{\mu}_t)$: correct the predicted state.

- Σ_t and $\mathbf{K}_t$ are independent of observation sequence, so compute offline.

2-D tracking example: filtering

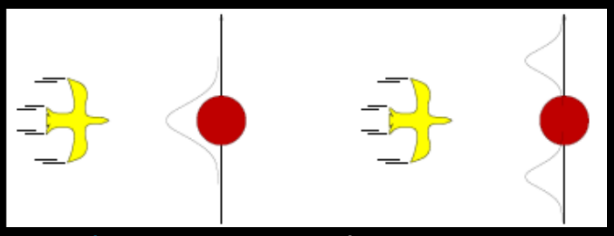


2-D tracking example: smoothing



Where it breaks

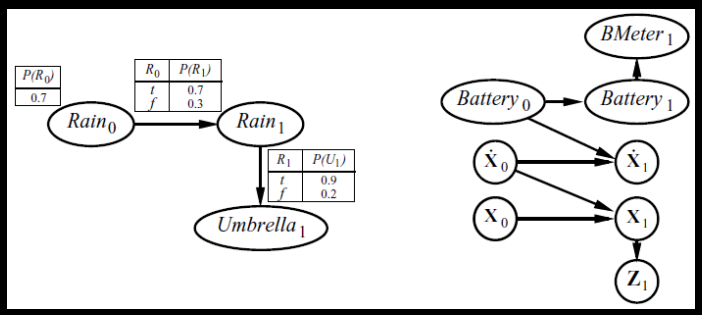
- Cannot be applied if the transition model is nonlinear
- Extended Kalman Filter models transition as locally linear around $x_t = \mu_t$
- Fails if systems is locally unsmooth or poorly-behaved -- evasive action.
  - there is significant nonlinearity in system response w/i the region that is close to μ_t.



- Switching Kalman filter: multiple Kalman filters run in parallel, each using a different model of the system. A weighted sum of predictions is used where the weight depends on how well each filter fits the current data.

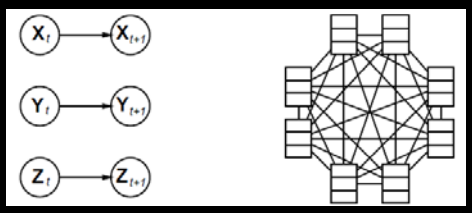
Dynamic Bayesian Networks(DBN)

- DBN is a Bayesian network that represent a sequence of state variables in transition and evidence variables in temporal probability model:
  E.g.) HMM, Kalman Filter.
- X_t , E_t contain arbitrarily many variables in a *replicated* Bayesian network
  Assumption: a replicated BN, 1st-order Markov process ⇒ parents from its own slice or from the immediately preceding slice.



DBNs vs. HMMs

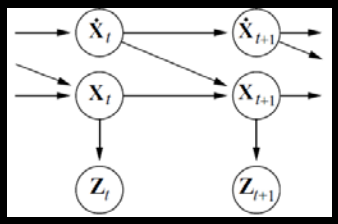
- Every HMM is a single-variable DBN; every discrete DBN is an HMM
  - Variables $V_1, .. V_n$ in DBN $\Rightarrow (V_1, .., V_n)$ where $v_i \in \text{Dom } D_i$



- Difference: By decomposing the state of a complex system into its constituent variables, the DBN is able to take advantage of sparseness in the temporal probability model: Spare dependencies $\Rightarrow$ exponentially fewer parameters;
  E.g., 20 boolean state variables, three parents each;
  DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$
  HMM requires a more space/expensive inference with huge Transition mat. of HMM, due to learning a huge # of parameters $\Rightarrow$unsuitable for large problem

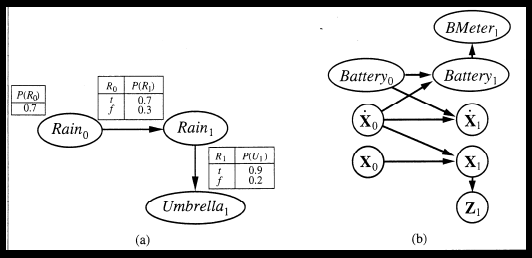
DBNs vs. Kalman filters

- Every Kalman filter model is
  a DBN with *continuous* variables + *linear* Gaussian distribution,
  but few DBNs are KFs with a *single* multivariate Gaussian dist.
- Real world requires *nonlinear*-Gaussian posteriors
  that require combination of discrete/continuous variables.



Construction of DBNs

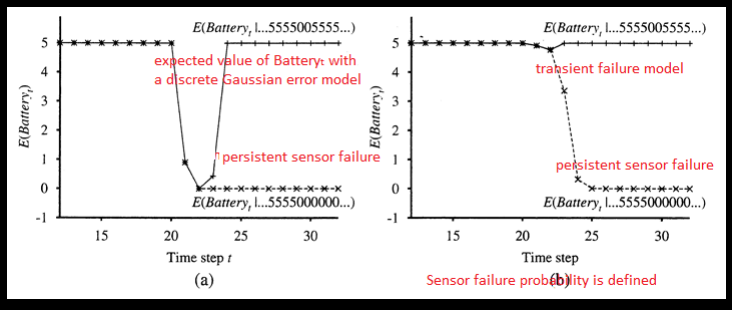
- Need $P(X_0)$, $P(X_{t+1}|X_t)$, $P(E_t|X_t)$: a prior distribution over the state variables, transition model, sensor model, respectively – a need of the topology of the connections b/t successive slices and b/t the state & evidence variables.
- Transition & sensor model: stationary $\Rightarrow$ a copy of the 1st slice.
- Example: monitoring a battery-powered robot moving in the X-Y plane.
  - Variables: $\mathbf{X}_t = (X_t, Y_t)$ for position, $\dot{\mathbf{X}}_t = (\dot{X}_t, \dot{Y}_t)$ for velocity, $Battery_t$; $\mathbf{Z}_t$ for *measurements* of position, $BMeter_t$ of *measurement* of battery charge level.
  - $BMeter_t$ would be a diagonal matrix with 1 if the meter is accurate; o.w. noise.



Construction of DBNs

- Handling of Noise:
  - The sensor model must include the possibility of failure to handle sensor failure properly.
  - *Gaussian error model* with transient failure of sensor
  - *Transient failure model* with probability of sensor's failure

    e.g.) P($BMeter_t$ =0| $Battery_t$ =5) = 0.03



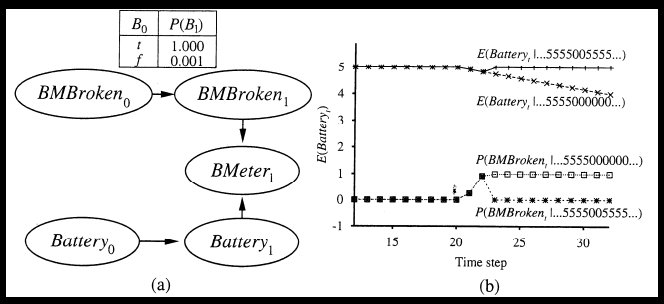
Construction of DBNs

- Handling of Noise:
  - Persistent failure model:

    - how the sensor behaves under normal conditions and after failure.

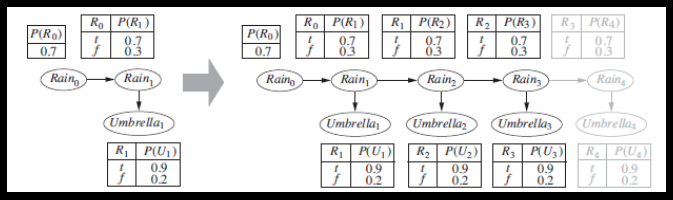
    - augment the hidden *state* of the system with an additional variable, *BMBroken*.

    A persistence arc with a probability of failure at *t*; sensor stays broken.



Exact inference in DBNs

- Naive method: *unrolling* the network and run any exact algorithm: Enumeration, VE, Joint-tree methods, etc.



- 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 $O(d^{n+k})$, update cost $O(nd^{n+k})$ - cf.) HMM update cost $O(d^{2n})$
- Even though we can use DBNs to represent very complex temporal processes with many sparsely connected variables, we can't reason efficiently and exactly a/b those processes. The DBN model is factorable into its constituent CPTs, but the posterior joint distribution conditioned on an observation sequence (i.e. the forward message) is generally not factorable $\Rightarrow$ a need of approximate method.

Approximate Inference in DBN: Particle Filtering (= Sequential Monte Carlo Method, Bootstrap Filter)

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

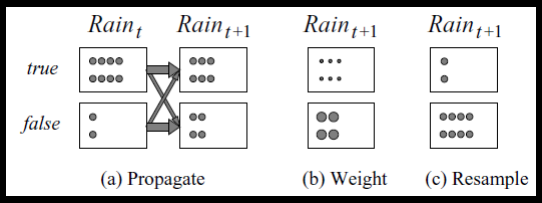
  for $i = 1$ to N **do**
    $S[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0 = S[i])$ /* step 1 */
    $W[i] \leftarrow \mathbf{P}(\mathbf{e} \mid \mathbf{X}_1 = S[i])$ /* step 2 */
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W) /* step 3 */
  return S

Approximate Inference in DBN: Particle filtering

- Approximate the distribution using the population of samples ("particles").
- As the number of particles N increases toward ∞, it converges to the actual distribution.
- Basic idea: Ensure that the population of samples ("particles") tracks the high-likelihood regions of the state-space.
  - Use the samples themselves as an approximate representation of the current state distribution.
  - Focus the set of samples on the high-probability regions of the state space – done by throwing away samples with low weight and replicating those with high weight.

Approximate Inference in DBN: Particle filtering

- Replicate particles proportional to likelihood for e_t.



- Widely used for tracking *nonlinear systems*, esp. in vision
- Also used for simultaneous localization and mapping in mobile robots

10^5–dimensional state space.

Approximate Inference in DBN: Particle filtering

Algorithm

1. Population of N initial-state samples is created by sampling from $P(X_0)$.
2. Update Cycle for each t:

1. Each sample is propagated forward by sampling the next state value $\mathbf{x}_{t+1}$ given the current value $\mathbf{x}_t$ for the sample, based on the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$.
2. Each sample is weighted by the likelihood it assigns to the new evidence, $P(\mathbf{e}_{t+1} \mid \mathbf{x}_{t+1})$.
3. The population is *resampled* to generate a new population of N samples. Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.

Particle filtering cont.

- Assume consistent at time t:
- $N(x_t | e_{1:t})$: the # of samples occupying state x_t after observations $e_{1:t}$:

  $N(x_t|e_{1:t}) / N = P(x_t|e_{1:t})$ for large N.
- Propagate forward: populations of x_{t+1} are

  $N(x_{t+1} | e_{1:t}) = \sum_{x_t} P(x_{t+1} | x_t) N(x_t | e_{1:t})$
- Weight samples by their likelihood for e_{t+1}:

  $W(x_{t+1} | e_{1:t+1}) = P(e_{t+1} | x_{t+1}) N(x_{t+1} | e_{1:t})$
- Resample to obtain populations proportional to W :

  $N(x_{t+1} | e_{1:t+1}) / N = \alpha W(x_{t+1}|e_{1:t+1})$
  $= \alpha P(e_{t+1}|x_{t+1}) N(x_{t+1}|e_{1:t})$
  $= \alpha P(e_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t) N(x_t|e_{1:t})$
  $= \alpha' P(e_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t) P(x_t|e_{1:t})$
  $= P(x_{t+1}|e_{1:t+1})$
- Performance: Approximation error of particle filtering remains bounded over time, at least empirically – theoretical analysis is difficult.

Summary

- Temporal models use state and sensor variables replicated over time
- Markov assumptions and stationay assumption, so we need
  - Transition model $P(X_t | X_{t-1})$
  - Sensor model $P(E_t | X_t)$
- Tasks are filtering, prediction, smoothing, most likely sequence; all done recursively with constant cost per time step
- Hidden Markov models have a single discrete state variable; used for speech recognition
- Kalman filters allow n state variables, linear Gaussian, $O(n^3)$ update.
- Dynamic Bayes nets subsume HMMs, Kalman filters; exact update intractable.
- Particle filtering is a good approximate filtering algorithm for DBNs.