Lecture 20: April 6, 2015
cs 573: Probabilistic Reasoning
Professor Nevatia
Spring 2015

Review

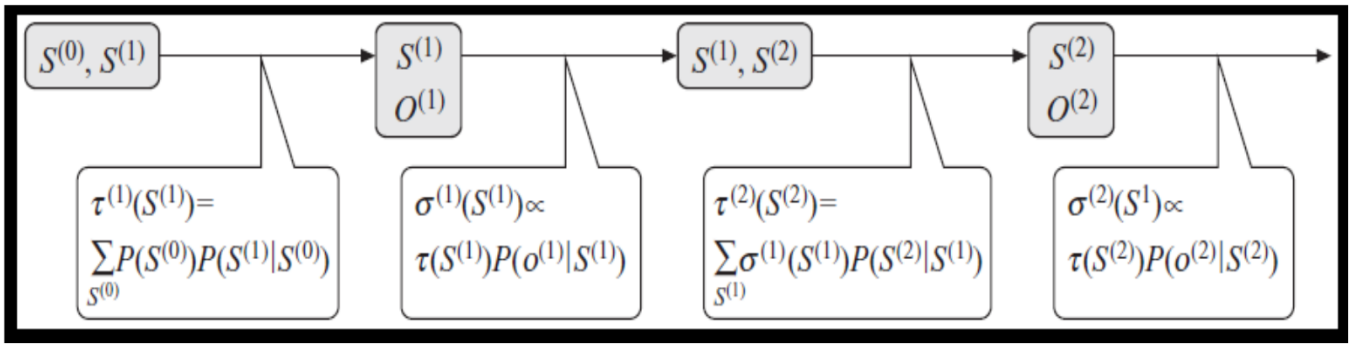
- HW #6A posted, due 4/8/15
- HW 6B: to be posted separately
- Previous Lecture
  - Intro to temporal models
- Today's objective
  - Inference in temporal models

Inference Tasks in Temporal Models

- *Filtering*: distribution of current state, given all the observations so far: $\mathbf{P}\left(\boldsymbol{X}^{(t)} \mid O^{(1:t)} \right)$
- *Prediction*: probability distribution over X at time t' > t, given $O^{(1:t)}$.
- *Smoothing*: Compute probabilities of a state given all the evidence: $\mathbf{P}\left(\boldsymbol{X}^{(t)} \mid O^{(1:T)} \right)$
- *Most likely trajectory* (sequence of states) given all the observations: arg max $_{\xi(0:T)}$ $\mathbf{P}\left(\boldsymbol{\xi}^{(0:T)} \mid O^{(1:T)} \right)$
  - Viterbi algorithm solves the last query

Exact Inference: State-Observation Models

- Can view unrolled model as any other graph: construct a clique tree for it and perform inference on it



- Filtering requires just the forward pass: normalize at each step
- Prediction is inference without evidence node (prediction can be from the start or after some time t)
- Smoothing requires a backward pass also: multiply messages and normalize
- Trajectory: solve the MAP problem
- Book derives recursive formula for forward pass but omits details of others. Instead, we look at more details from the Russell-Norvig book (slides should be self-contained)

Filtering Equations

- Notation: belief state $\sigma^{(t)}(\boldsymbol{X}^{(t)}) = \mathbf{P}(\boldsymbol{X}^{(t)} \mid O^{(1:t)})$
- Goal: compute $\sigma^{(t+1)}$ from $\sigma^{(t)}$ (recursive algorithm)
- Notation: $\sigma^{(.t+1)}(\boldsymbol{X}^{(t+1)}) = \mathbf{P}(\boldsymbol{X}^{(t+1)} \mid O^{(1:t)})$; .t says evidence up to time *t* only (prior belief state)
- $\sigma^{(.t+1)}(\boldsymbol{X}^{(t+1)}) = \mathbf{P}(\boldsymbol{X}^{(t+1)} \mid O^{(1:t)}) =$

$$= \Sigma_{xt}\, \mathbf{P}(\boldsymbol{X}^{(t+1)} \mid \boldsymbol{X}^{(t)}, O^{(1:t)})\, \mathbf{P}(\boldsymbol{X}^{(t)} \mid O^{(1:t)})$$

$$= \Sigma_{xt}\, \mathbf{P}(\boldsymbol{X}^{(t+1)} \mid \boldsymbol{X}^{(t)})\, \sigma^{(t)}(\boldsymbol{X}^{(t)})$$

- Add the effect of $O^{(t+1)}$:

$$\begin{aligned}
\sigma^{(t+1)}(\boldsymbol{X}^{(t+1)}) &= P(\boldsymbol{X}^{(t+1)} \mid o^{(1:t)}, o^{(t+1)}) \\
&= \frac{P(o^{(t+1)} \mid \boldsymbol{X}^{(t+1)}, o^{(1:t)}) P(\boldsymbol{X}^{(t+1)} \mid o^{(1:t)})}{P(o^{(t+1)} \mid o^{(1:t)})} \\
&= \frac{P(o^{(t+1)} \mid \boldsymbol{X}^{(t+1)}) \sigma^{(\cdot t+1)}(\boldsymbol{X}^{(t+1)})}{P(o^{(t+1)} \mid o^{(1:t)})}.
\end{aligned}$$

- Basically, multiply by observation probability and normalize
- We can derive same equations from clique tree, normalized belief propagation

Smoothing

- Estimate $\mathbf{P}\ (\ \boldsymbol{X}^{(t)} \mid O^{(1:T)}\ )$
- Probability of state at time *t* given all the observation variables
- Can be computed directly from the belief propagation algorithm on the clique tree
- Alternative formulations of backward/forward passes exist
  - They compute different probabilities but, in the end, compute a smoothed estimate of probabilities.
  - A derivation from Bilmes is given in following slides

Alternative Forward-Backward Algorithms (from Bilmes Tutorial)

- Define $\alpha_i(t) = p(O_1 = o_1, \ldots, O_t = o_t, Q_t = i|\lambda)$

  1. $\alpha_i(1) = \pi_i b_i(o_1)$
  2. $\alpha_j(t+1) = \left[\sum_{i=1}^{N} \alpha_i(t) a_{ij}\right] b_j(o_{t+1})$
  3. $p(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T)$

- Define $\beta_i(t) = p(O_{t+1} = o_{t+1}, \ldots, O_T = o_T|Q_t = i, \lambda)$

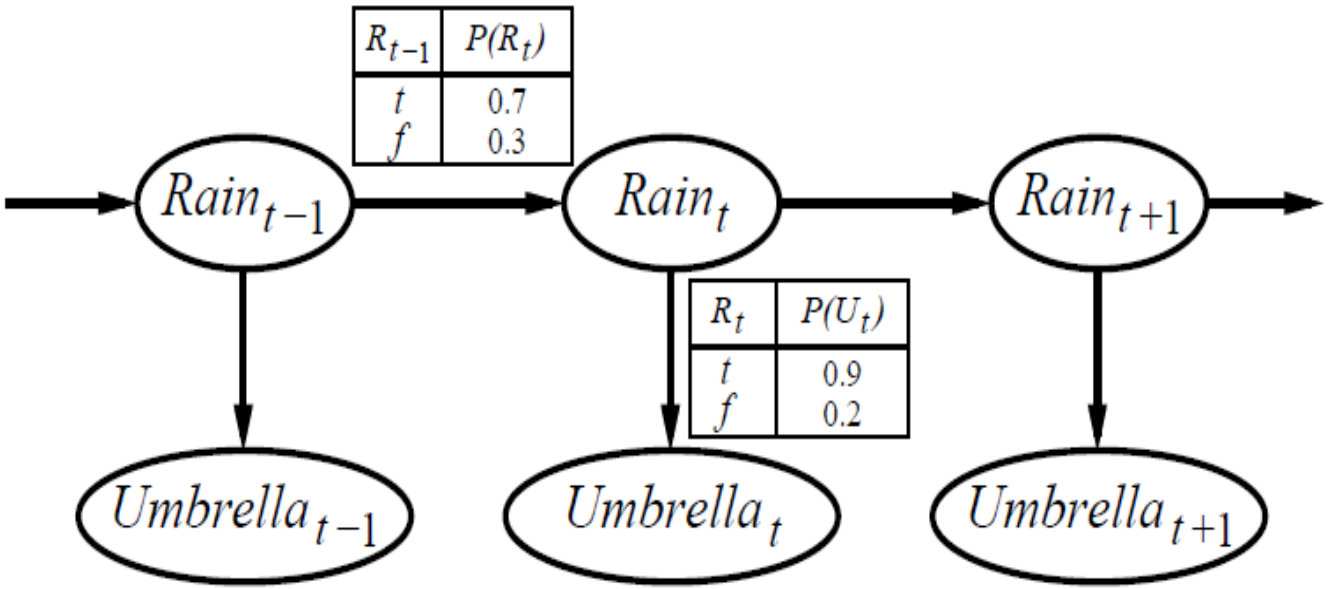
  1. $\beta_i(T) = 1$
  2. $\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1)$
  3. $p(O|\lambda) = \sum_{i=1}^{N} \beta_i(1) \pi_i b_i(o_1)$

- Define $\gamma_i(t) = p(Q_t = i|O, \lambda)$

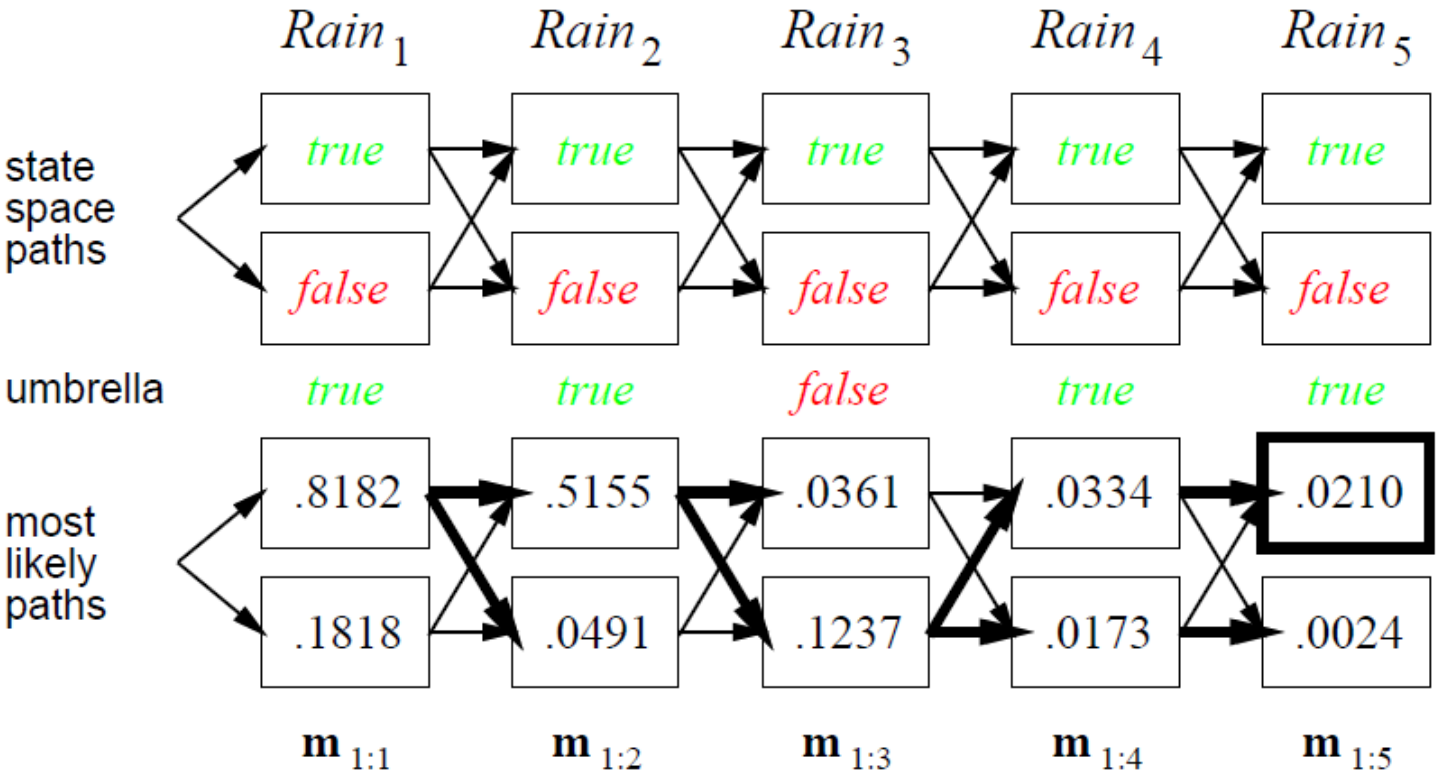
$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

Example (from Russell-Norvig Book)

- Security guard in basement can't see whether it is raining outside or not but can only observe if people come in with umbrella or not.



Viterbi Example



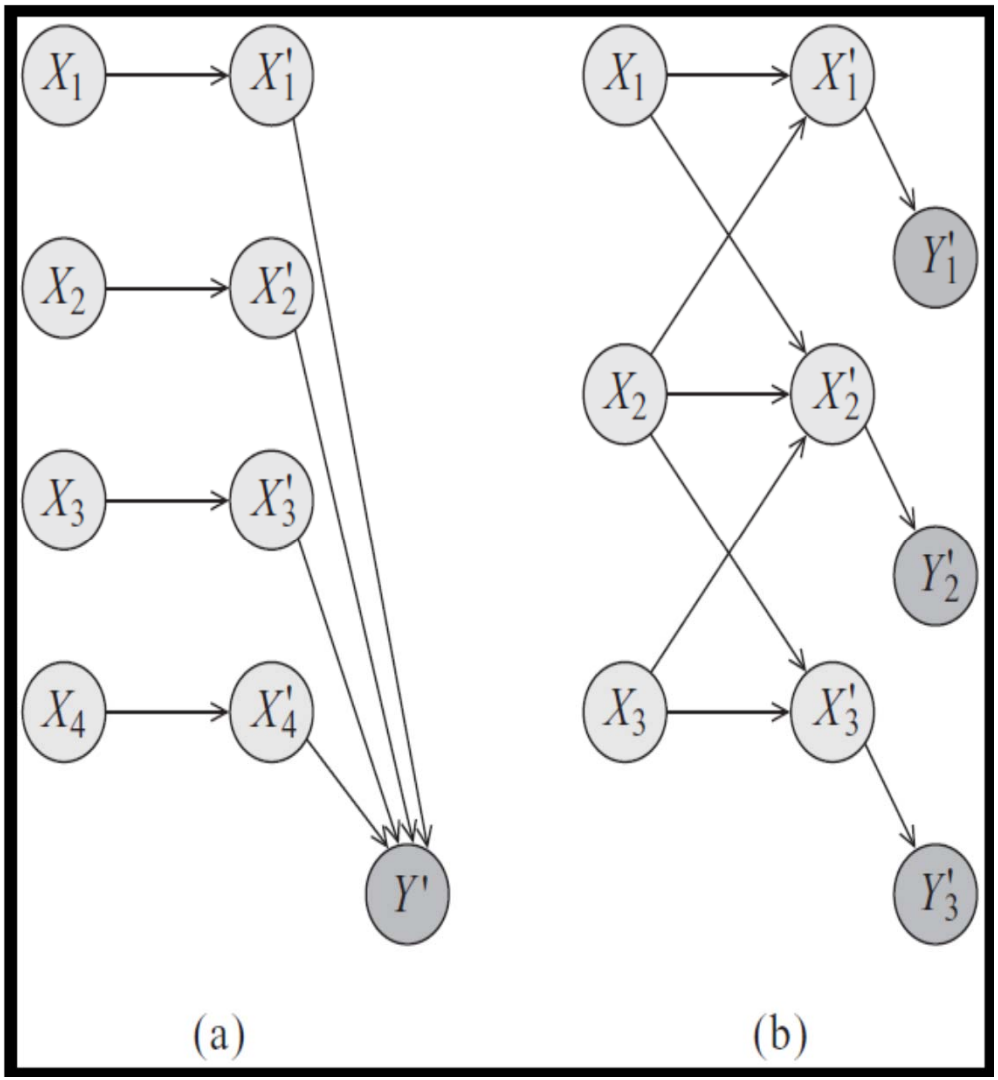
Note: **bold box** indicates the probability of the best sequence to that node. **Bold arrows** show the previous state on this best path. At each step, we multiply by transition probability and observation probability. Actually, the diagram ignores multiplication by observation probability as it is the same for each path and hence not relevant to finding the maximum

Viterbi Complexity

- Time complexity: $O(N^2T)$: N is the number of states, T is time
- Space Complexity: O(NT)
- History:
  - Originally designed to decode convolution codes
  - Stated to be an approximation by Dr. Viterbi
  - Later (soon after) recognized to be an exact, optimal algorithm not only for convolution codes but for any HMM
  - Generalizations to more complex networks that use dynamic programming still often referred to as the Viterbi algorithm

More Specialized Models

Factorial HMM



Coupled HMM

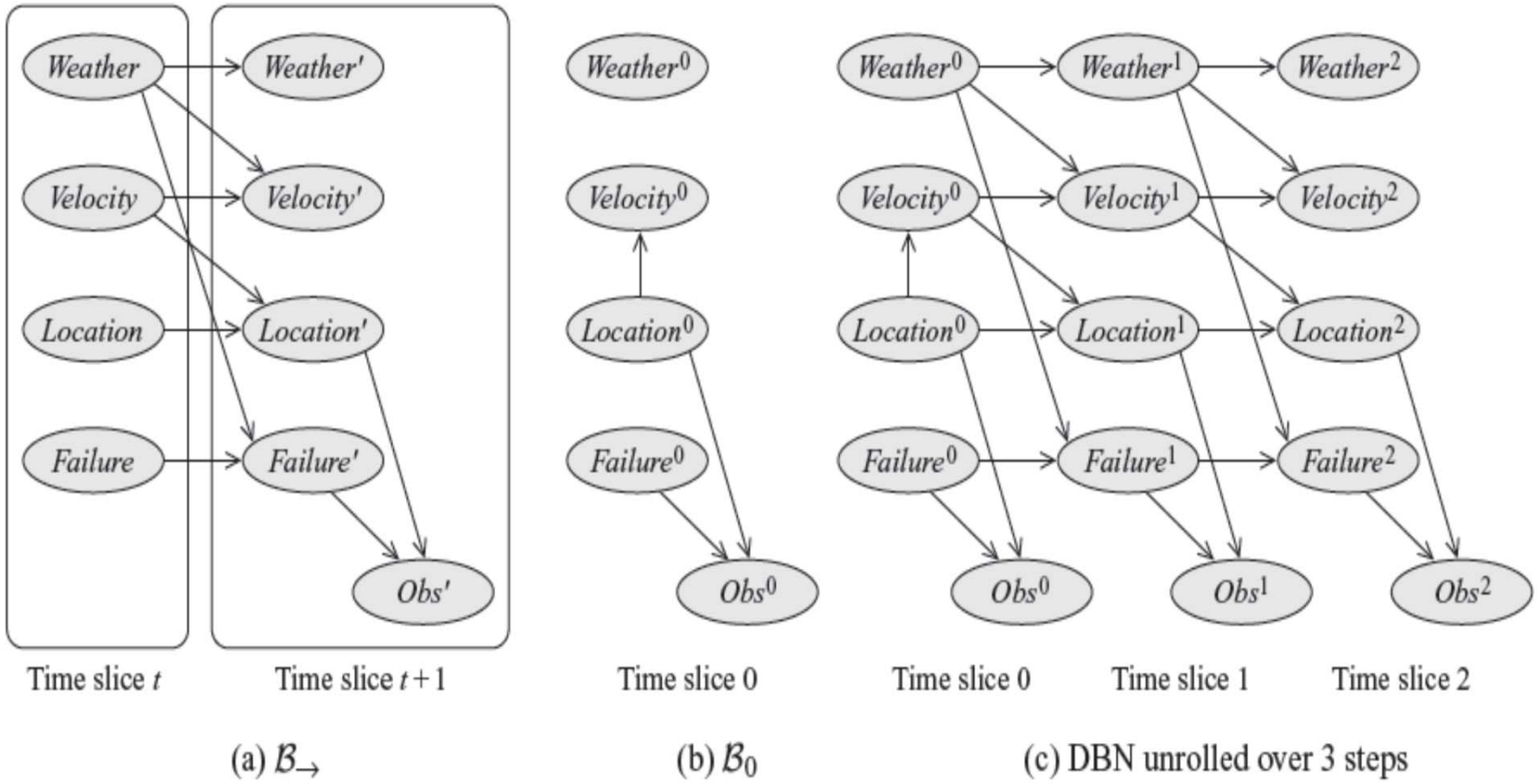
Semi-Markov Models

- In an HMM, probability of staying in one state declines exponentially with time.
- Semi-Markov HMMs (HSMM) allow transition probability to be a function of the time spent in that state, allowing us to design arbitrary duration models (uniform, Gaussian…)
  - However, "decoding", *i.e.* finding the most likely sequence becomes more complex and is no longer linear in number of time frames.

Inference in General DBNs

- Construct a clique tree and do inference over it
  - In HMMs, though the clique tree can be unboundedly large, only the tree over two neighboring slices is needed at a time
  - This observation also generalizes to general DBNs
- However, belief state in this algorithm is a joint distribution over $X_I^{(t)}$ (X_I are the interface variables)
  - Exponential number of entries in this joint
- A compact representation of the belief state in a DBN is not always possible as some intuitive conditional independence properties do not hold
- Example: Weather$^{(2)}$ and Location$^{(2)}$ are connected via Velocity$^{(1)}$ and Weather$^{(0)}$ with no evidence nodes in between (next slide)
  - Such connections are termed *entanglement*
  - Correspondingly, needed cliques may be large

Dynamic Bayesian Networks (DBNs)



Approximate Inference in DBNs

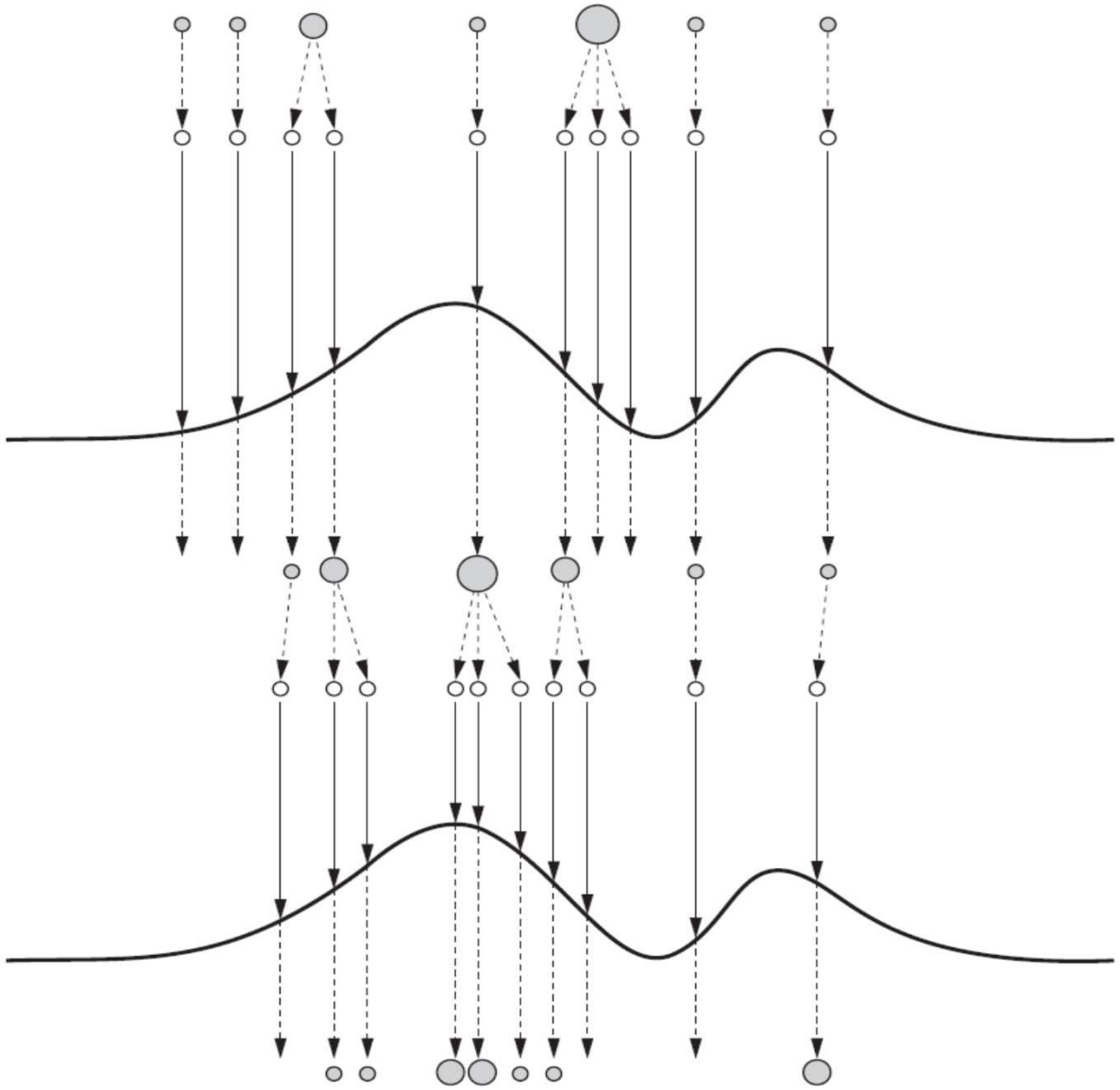
- Entanglement argues for need of approximate inference
- Loopy BP and variational methods are also problematic as the size of the graph can be very large in typical temporal problems and each slice needs to be visited repeatedly
- Bounded History: Even though state at time *t* may be influenced by states at much earlier time slots, such influence can be expected to decay with time so limited history might still give reasonable approximations.
- Particle Filtering
  - Forward sampling to generate samples (particles)
  - For long trajectories, rejection sampling will reject most samples
  - Usual LW sampling starts from the root and continues till the leaf nodes are reached; we want "on-line" answers

Naïve Likelihood Weighting

- Maintain a set of samples (each represents a trajectory) for each time step *t*
  - Let samples be $\xi^{(t)}[1], \ldots, \xi^{(t)}[M]$
  - w[m] is the weight associated with the m^{th} sample
- At each time slice, propagate each sample forward and adjust the weight according to the evidence
- Alg 15.2 applies to a 2-TBN
- Performance of LW sampling likely to be not very good as trajectories get long.
  - Particles are generated mostly based on prior distributions and matching with evidence is by chance so most trajectories will be of low weight.
- Solution: only propagate the "good" (high probability) samples forward with time

Bootstrap Particle Filtering

- Let $D^{(t)}$ be a set of *M* trajectories at time *t*
- Each trajectory is denoted as $\mathbf{x}^{-(0:t)}[m]$ with weight $w^{(t)}[m]$
- Distribution generated by set is: $\hat{P}_{\mathcal{D}^{(t)}}(\boldsymbol{x}^{(0:t)}) \propto \sum_{m=1}^{M} w^{(t)}[m] \boldsymbol{I}\{\bar{\boldsymbol{x}}^{(0:t)}[m] = \boldsymbol{x}^{(0:t)}\}$
  - Sum of delta distribution functions
- Generate *M* new samples at time *t* +1
  - Select a sample to propagate according to $P_{D(t)}$ above (*i.e.* proportional to the weight of the sample)
  - Propagate as in the LW-2TBN process: compute sample according to the transition function; compute weight from the evidence
- Note: higher weight particles may be propagated multiple times, lower weighted ones may not be selected at all
  - May need strategy for regenerating new particles (e.g. MCMC or exact inference on a time slice); we omit details given in sec 15.3.3.3 to 15.3.3.5
- Illustration in Fig 15.5, results in Fig 15.6



Particle Filtering Results

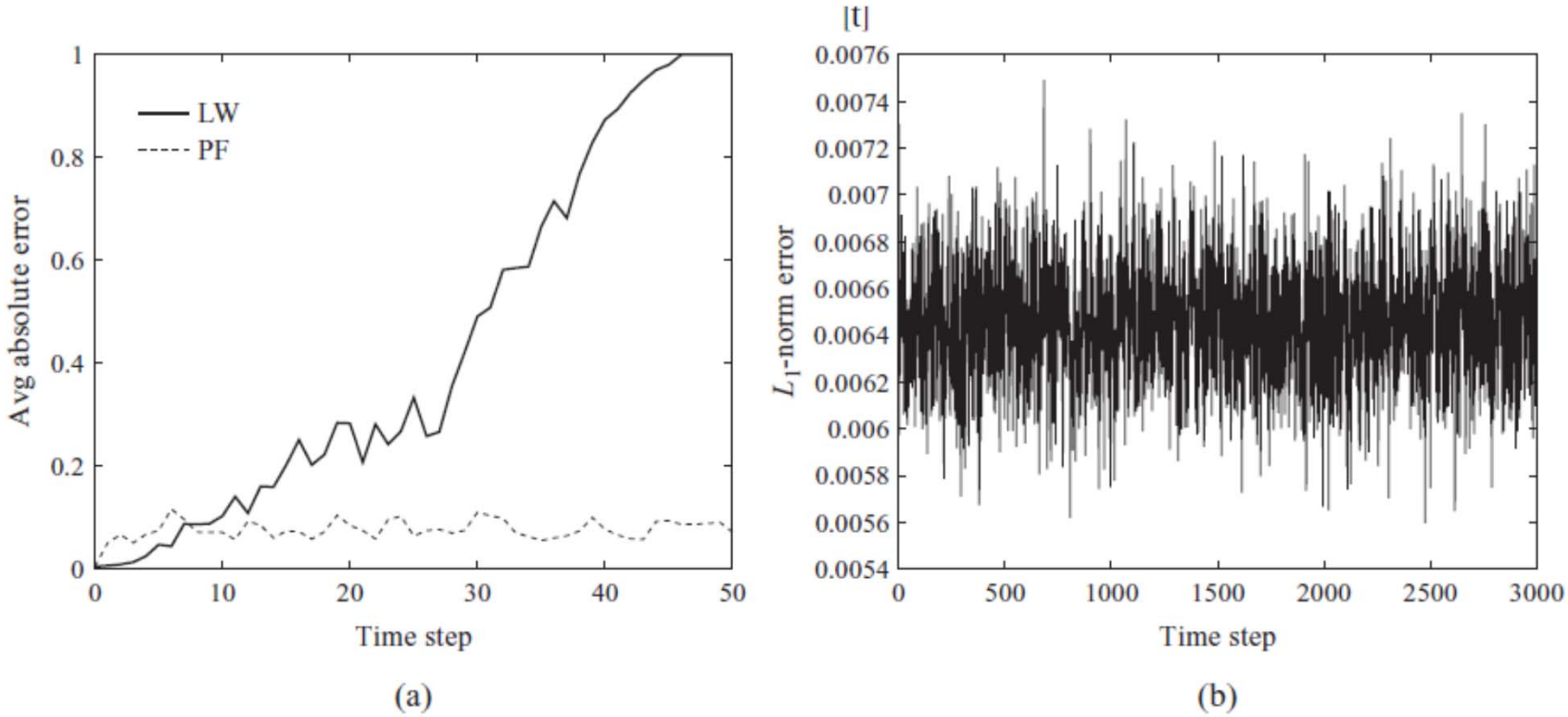


Figure 15.6 Likelihood weighting and particle filtering over time. (a) A comparison for 1,000 time slices. (b) A very long run of particle filtering.

Video Tracking Examples

- From paper by Isard-Blake

Continuous DBNs: Kalman-Bucy Filter

- Linear Dynamical system: all variables are continuous, dependencies are linear Gaussian

  $P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}) = N(A\mathbf{X}^{(t-1)}; Q)$; $P(O^{(t)} \mid \mathbf{X}^{(t)}) = N(H\mathbf{X}^{(t)}; R)$

- Solve as for the discrete case: propagate according to transition function by marginalizing, update by multiplying by observation function
- Transition Update

  $$\mu^{(.t+1)} = A\,\mu^{(t)}; \quad \Sigma^{(.t+1)} = A\,\Sigma^{(t)}\,A^T + Q$$

- Observation update

$$\begin{aligned} K^{(t+1)} &= \Sigma^{(\cdot t+1)} H^T (H\Sigma^{(\cdot t+1)} H^T + R)^{-1} \\ \mu^{(t+1)} &= \mu^{(\cdot t+1)} + K^{(t+1)}(o^{(t+1)} - H\mu^{(\cdot t+1)}) \\ \Sigma^{(t+1)} &= (I - K^{(t+1)} H)\Sigma^{(\cdot t+1)}. \end{aligned}$$

Continuous DBNs: Kalman-Bucy Filter

- Note: K is called Kalman Gain,
  - If R tends to 0, K tends to H^{-1}; we trust observation completely; co-variance tends to 0
  - If co-variance tends to 0, K tends to 0; observation is essentially ignored
  - Interestingly, $\Sigma^{(t+1)}$ tends to a stable value, independent of observations
- We can also derive Kalman filter equations in canonical (information) form
  - Observation update will have simpler equation, transition update will be more complex.

Non-linear Systems

- Let $P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}) = f(\mathbf{X}^{(t-1)}; \mathbf{U}^{(t-1)})$;

  $P(O^{(t)} \mid \mathbf{X}^{(t)}) = g(\mathbf{X}^{(t)}; \mathbf{W}^{(t)})$

  f and g are deterministic, possibly non-linear functions,

  $\mathbf{U}^{(t)}$, $\mathbf{W}^{(t)}$) are Gaussian random variables
- Extended Kalman Filter (EKF)
  - Approximate f and g by expanding in Taylor series and maintaining first order gradient terms only
- Unscented Kalman Filter (UKF)s
  - Maintain f and g as non-linear functions but approximate resulting distributions by a Gaussian
  - Sample some points and fit a Gaussian to result
- Details of non-linear systems described in sec 15.4.1.2 of KF book; we omit for CSCI 573.

Next Class

- Read sections 17.1 and 17.2 (excluding 17.2.4 and 17.2.5) of the KF book