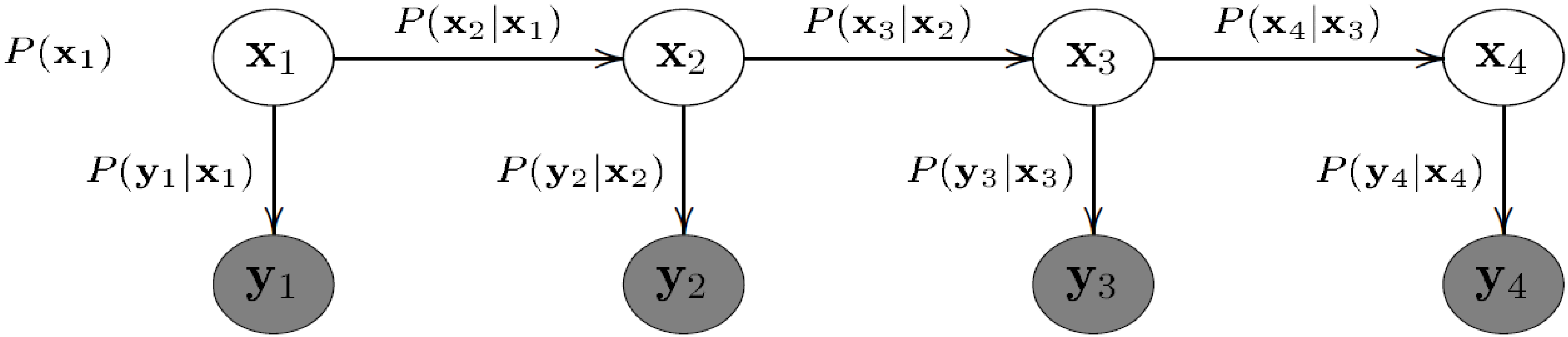
Expectation Propagation for Approximate Inference in Dynamic Bayesian Networks

Tom Heskes and Onno Zoeter

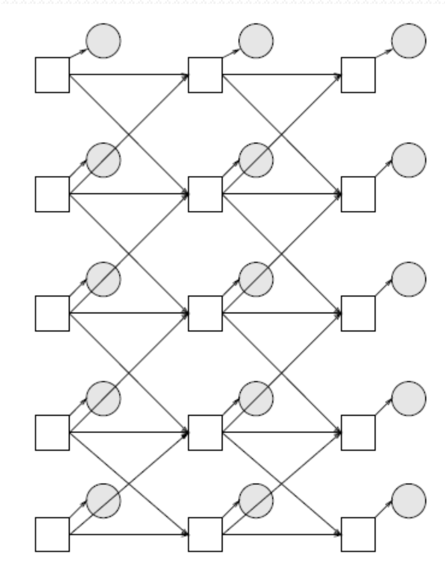
Presented by Mark Buller

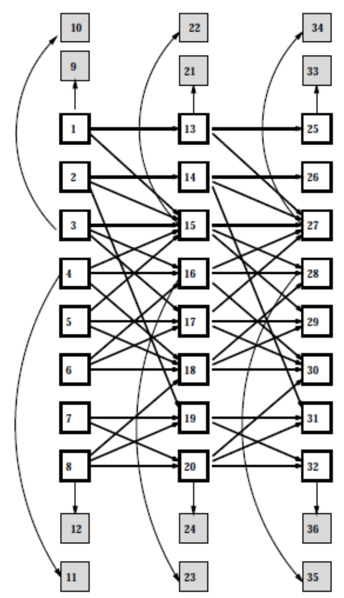
Dynamic Bayesian Networks



- Directed graphical models of stochastic processes
- Represent hidden and observed variables with different dependencies
- Generalize Hidden Markov Models (HMM)

Goal is Inference





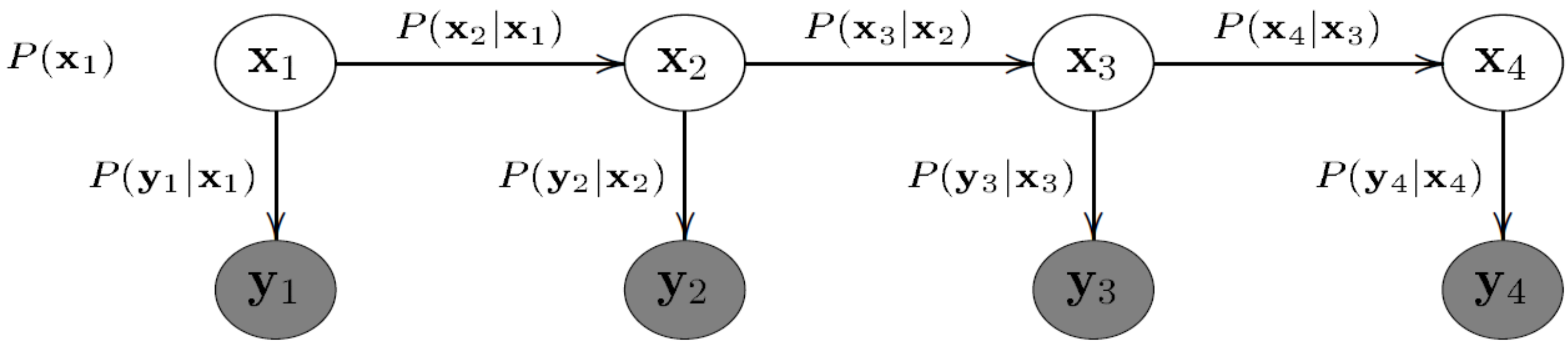
- Fart Left coupled HMM with 5 chains
- Left DBN to monitor waste water treatment plant.
- Murphy and Weiss 2001

- Will generally like to perform inference: $P(\mathbf{x}_t \mid \mathbf{y}_{1:T})$
- Why not discretize and use the "Forward-Backward" algorithm for exact inference?
- Very quickly can become untenable.

Approximate Inference

- Sampling
  - Particle Filters
- Variational
  - (Ghahramani and Hinton 1998) Switching Linear Dynamical System
  - (Ghahramani and Jordan 1997) Factorial Hidden Markov Models
- Variational Subset
  - Greedy projection algorithms
    - Where projection provides a simpler approximate belief
    - Expectation Propagation

Problem Setup



$$P(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = \prod_{t=1}^{T} \psi_t(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{y}_t)$$

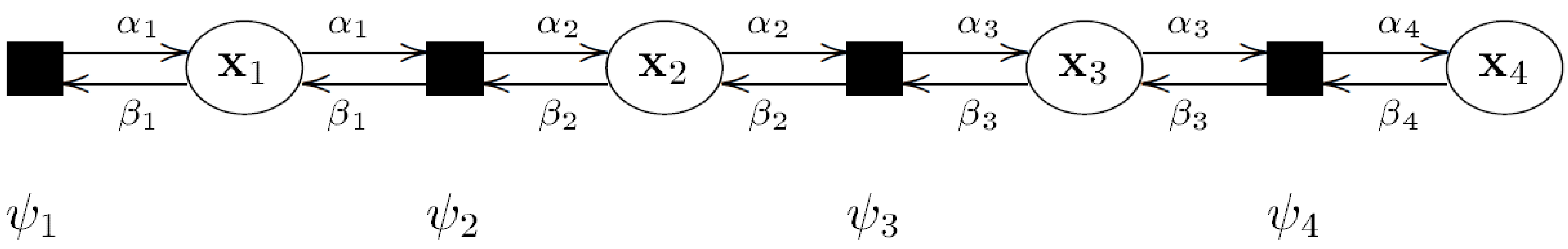
$$\psi_t(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{y}_t) = P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{y}_t|\mathbf{x}_t)$$

- $\mathbf{x}_t$ – super node that contains all latent variables at a time point.
- $\mathbf{y}_{1:T}$ – fixed and is included in the definition of the potentials: $\psi_t(x_{t-1,t}) \equiv \psi_t(x_{t-1}, x_t, y_t)$

Goal: Infer $P(\mathbf{x}_t \mid \mathbf{y}_{1:T})$

- Find the marginal "beliefs" or the probability distributions of the latent variables at a given time given all the evidence.
- Pearl's Belief Propagation (1988)
- Specific case of the sum-product rule in factor graphs (Kschischang et al., 2001)
- Note: In chain factor graphs variable nodes simply pass received messages on to the next function node.

Message Propagation



1. Compute estimate of distribution at local function node:

$$\hat{P}(\mathbf{x}_{t-1,t}) \propto \alpha_{t-1}(\mathbf{x}_{t-1})\psi_t(\mathbf{x}_{t-1,t})\beta_t(\mathbf{x}_t)$$

2. Integrate out all variables except $\mathbf{x}_{t'}$ ($\mathbf{x}_{t'}$ the node to which the message is sent) to get current estimate of the belief $\hat{P}(\mathbf{x}_{t'})$ and project this belief onto a distribution in the exponential family:

$$q_{t'}(\mathbf{x}_{t'})$$

3. Conditionalize, i.e. divide by message from $X_{t'}$ to ψ_t

Belief Approximation

- Project belief takes an exponential family form:

$$q_t(\mathbf{x}_t) \propto e^{\boldsymbol{\gamma}_t^T \mathbf{f}(\mathbf{x}_t)}$$

  - Where γ_t = canonical parameters and $f(x_t)$ the sufficient statistics.
  - If the forward and backward messages are initialized as:

$$\alpha_t(\mathbf{x}_t) \propto e^{\boldsymbol{\alpha}_t^T \mathbf{f}(\mathbf{x}_t)} \qquad \beta_t(\mathbf{x}_t) \propto e^{\boldsymbol{\beta}_t^T \mathbf{f}(\mathbf{x}_t)}$$

  - With $\boldsymbol{\alpha}_t = \boldsymbol{\beta}_t = \mathbf{0}$ then the canonical parameters α_t and β_t will fully specify the messages $\alpha_t(x_t)$ and $\beta_t(x_t)$.
  - Thus the belief can be specified as a combination of the messages

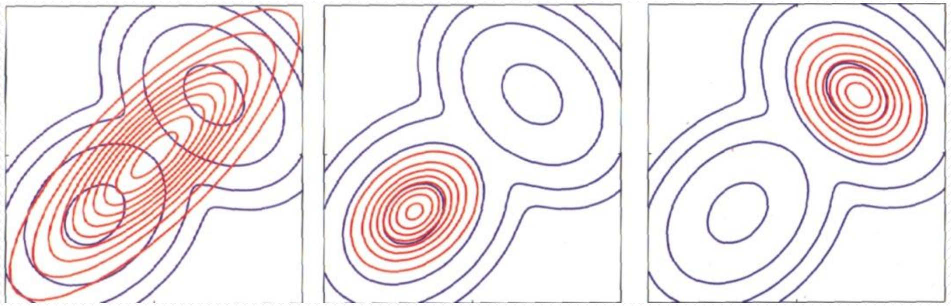
$$\boldsymbol{\gamma}_t = \boldsymbol{\alpha}_t + \boldsymbol{\beta}_t$$

Moment Matching

- To project the belief $\hat{P}(\mathbf{x}_{t'})$ to the best exponential family approximation is found when the Kullback-Leibler (KL) divergence is minimized:

$$\mathrm{KL}(\hat{P}|q) = \int d\mathbf{x}\, \hat{P}(\mathbf{x}) \log \left[\frac{\hat{P}(\mathbf{x})}{q(\mathbf{x})} \right]$$

- Minima is found when the moments of P(x) and q(x) are matched.



Bishop 2006

KL(p|q) KL(q|p) KL(q|p)

- Function **g** converts from canonical form to moments

$$\mathbf{g}(\boldsymbol{\gamma}) \equiv \langle \mathbf{f}(\mathbf{x}) \rangle_q \equiv \int d\mathbf{x}\, q(\mathbf{x}) \mathbf{f}(\mathbf{x}) = \int d\mathbf{x}\, \hat{P}(\mathbf{x}) \mathbf{f}(\mathbf{x})$$

Computing Forward and Backward Messages

- Compute α_t such that:

$$\langle \mathbf{f}(\mathbf{x}_t) \rangle_{\hat{p}_t} = \langle \mathbf{f}(\mathbf{x}_t) \rangle_{q_t} = \mathbf{g}(\boldsymbol{\alpha}_t + \boldsymbol{\beta}_t)$$

- With β_t kept fixed:

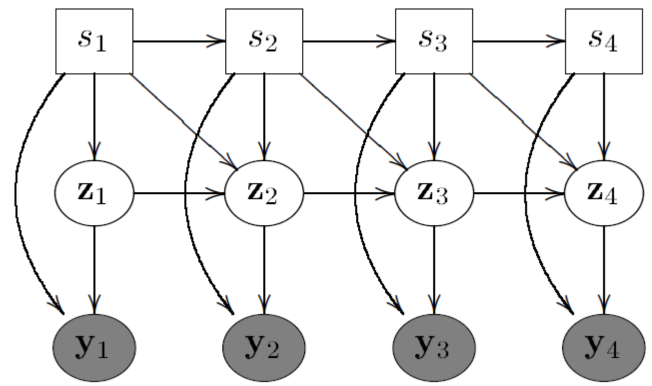
$$\boldsymbol{\alpha}_t = g^{-1}(\langle \mathbf{f}(\mathbf{x}_t) \rangle_{\hat{p}_t}) - \boldsymbol{\beta}_t$$

- Similarly Compute β_{t-1} such that:

$$\langle \mathbf{f}(\mathbf{x}_{t-1}) \rangle_{\hat{p}_t} = \langle \mathbf{f}(\mathbf{x}_{t-1}) \rangle_{q_{t-1}} = \mathbf{g}(\boldsymbol{\alpha}_{t-1} + \boldsymbol{\beta}_{t-1})$$

- Note: without the projection to the exponential family this is basically the standard forward backward algorithm.
- Order of message updating is free

Example: Switching Linear Dynamical System



- Potentials:

$$\psi_t(s_{t-1,t}^{i,j}, \mathbf{z}_{t-1,t}) = p_\psi(s_t^j | s_{t-1}^i) \Phi(\mathbf{z}_t; A_{ij}\mathbf{z}_{t-1}, Q_{ij}) \Phi(\mathbf{y}_t; C_j \mathbf{z}_t, R_j)$$

- Messages are taken to be conditional Gaussian potentials:

$$\begin{aligned} \alpha_{t-1}(s_{t-1}^i, \mathbf{z}_{t-1}) &\propto p_\alpha(s_{t-1}^i) \Psi(\mathbf{z}_{t-1}; \mathbf{m}_{i,t-1}^\alpha, V_{i,t-1}^\alpha) \\ \beta_t(s_t^j, \mathbf{z}_t) &\propto p_\beta(s_t^j) \Psi(\mathbf{z}_t; \mathbf{m}_{j,t}^\beta, V_{j,t}^\beta) , \end{aligned}$$

Example: Step 1

- Compute estimate of distribution at local function node :

$$\hat{P}(s^{i,j}_{t-1,t}, \mathbf{z}_{t-1,t}) \propto \alpha_{t-1}(s^{i}_{t-1}, \mathbf{z}_{t-1})\psi_t(s^{i,j}_{t-1,t}, \mathbf{z}_{t-1,t})\beta_t(s^{j}_{t}, \mathbf{z}_{t})$$

- Messages are combinations of M Gaussian potentials one for each switch state *i*. *Transform to a representation with moments*

$$\hat{P}(s^{i,j}_{t-1,t}, \mathbf{z}_{t-1,t}) \propto \hat{p}_{ij}\Phi(\mathbf{z}_{t-1,t}; \hat{\mathbf{m}}_{ij}, \hat{V}_{ij})$$

Example: Step 2

- Integrate and sum out components $\mathbf{z}_{t-1}$ and $\mathbf{s}_{t-1}$:
- Integration over $\mathbf{z}_{t-1}$ can be done directly:

$$\hat{P}(s_{t-1,t}^{i,j}, \mathbf{z}_t) \propto \hat{p}_{ij} \Phi(\mathbf{z}_t; \hat{\mathbf{m}}_{ij}, \hat{V}_{ij})$$

- Summation over $\mathbf{s}_{t-1}$ yields a mixture of Gaussians and must be approximated using moment matching:

$$q_t(s_t^j, \mathbf{z}_t) = \hat{p}_j \Phi(\mathbf{z}_t; \hat{\mathbf{m}}_j, \hat{V}_j)$$

Example: Step 3

- Forward message is found by dividing the approximate belief by the backward message :

$$\alpha_t(s_t, \mathbf{z}_t) = \frac{\text{Convert to Canonical form } q_t(s_t, \mathbf{z}_t)}{\beta_t(s_t, \mathbf{z}_t)}$$

Observations

- Backward pass is symmetric to the forward pass.
- Forward filtering pass is equivalent to a popular inference algorithm for switching linear dynamical system (GPB2 – Bar-Shalom and Li 1993)
- Backward smoothing pass improves upon current algorithms because no additional approximations were required.
- Forward and Backward passes can be iterated until convergence.
- Expectation propagation can be used to iteratively improve other methods for inference in DBNs (e.g. Murphy and Weiss 2001)
- But this algorithm does not always converge

Bethe Free Energy

- Fixed points of expectation propagation correspond to fixed points of the "Bethe free energy" (Minka, 2001)

$$F(\hat{p}, q) = -\sum_{t=1}^{T-1} \int d\mathbf{x}_t \, q_t(\mathbf{x}_t) \log q_t(\mathbf{x}_t) + \sum_{t=1}^{T} \int d\mathbf{x}_{t-1,t} \, \hat{p}_t(\mathbf{x}_{t-1,t}) \log \left[\frac{\hat{p}_t(\mathbf{x}_{t-1,t})}{\psi_t(\mathbf{x}_{t-1,t})} \right]$$

- Expectation constraints

$$\langle \mathbf{f}(\mathbf{x}_t) \rangle_{\hat{p}_t} = \langle \mathbf{f}(\mathbf{x}_t) \rangle_{q_t} = \langle \mathbf{f}(\mathbf{x}_t) \rangle_{\hat{p}_{t+1}}$$

- Under these constraints the free energy function may not be convex. i.e. Can have local fixed points.

Double Loop Algorithm

- Linearly bound concave part:

$$F_{\text{bound}}(\hat{p}, q, q^{\text{old}}) = -\sum_{t=1}^{T-1} \int d\mathbf{x}_t \, q_t(\mathbf{x}_t) \log q_t^{\text{old}}(\mathbf{x}_t)$$

$$+ \sum_{t=1}^{T} \int d\mathbf{x}_{t-1,t} \, \hat{p}_t(\mathbf{x}_{t-1,t}) \log \left[\frac{\hat{p}_t(\mathbf{x}_{t-1,t})}{\psi_t(\mathbf{x}_{t-1,t})} \right] .$$

- For each outer loop step reset the bound:

$$F_{\text{bound}}(\hat{p}, q, q^{\text{old}}) = F(\hat{p}, q)$$

- For inner loop solve convex constrained minimization problem, guaranteeing:

$$F(\hat{p}^{\text{new}}, q^{\text{new}}) \leq F_{\text{bound}}(\hat{p}^{\text{new}}, q^{\text{new}}, q^{\text{old}}) \leq F_{\text{bound}}(\hat{p}, q, q^{\text{old}}) = F(\hat{p}, q)$$

Inner Loop

- Change to a constrained maximization problem over Lagrange multipliers δ_t:

$$F_1(\boldsymbol{\gamma}, \boldsymbol{\delta}) = -\sum_{t=1}^{T} \log Z_t \text{ with}$$

$$Z_t = \int d\mathbf{x}_{t-1,t}\, \mathrm{e}^{\boldsymbol{\alpha}_{t-1}^T \mathbf{f}(\mathbf{x}_{t-1})} \psi_t(\mathbf{x}_{t-1,t}) \mathrm{e}^{\boldsymbol{\beta}_t^T \mathbf{f}(\mathbf{x}_t)}$$

- With: $\log q^{old}(\mathbf{x}_t) \equiv \gamma_t \mathbf{f}(\mathbf{x}_t)$ and substituting:

$$\boldsymbol{\alpha}_t = \frac{1}{2}(\boldsymbol{\gamma}_t + \boldsymbol{\delta}_t) \text{ and } \boldsymbol{\beta}_t = \frac{1}{2}(\boldsymbol{\gamma}_t - \boldsymbol{\delta}_t)$$

- “That is, $\boldsymbol{\delta}$ can be interpreted as the difference between the forward and backward messages, γ as their sum”.

Inner Loop Maximization

- In terms of: $\tilde{\boldsymbol{\alpha}}_t \equiv \tilde{\boldsymbol{\alpha}}_t(\boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t)$ and $\tilde{\boldsymbol{\beta}}_t \equiv \tilde{\boldsymbol{\beta}}_t(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_{t+1})$ gradient with respect to δ_t:

$$\frac{\partial F_1(\boldsymbol{\gamma}, \boldsymbol{\delta})}{\partial \boldsymbol{\delta}_t} = \frac{1}{2}\left[\mathbf{g}(\tilde{\boldsymbol{\alpha}}_t + \boldsymbol{\beta}_t) - \mathbf{g}(\boldsymbol{\alpha}_t + \tilde{\boldsymbol{\beta}}_t)\right]$$

- Set to 0: $\boldsymbol{\delta}_t^{\text{new}} = \tilde{\boldsymbol{\delta}}_t \equiv \tilde{\boldsymbol{\alpha}}_t - \tilde{\boldsymbol{\beta}}_t$
- Damp update: $\boldsymbol{\delta}_t^{\text{new}} = \boldsymbol{\delta}_t + \epsilon_\delta(\tilde{\boldsymbol{\delta}}_t - \boldsymbol{\delta}_t)$
- Outer-loop can be re-written as the update:

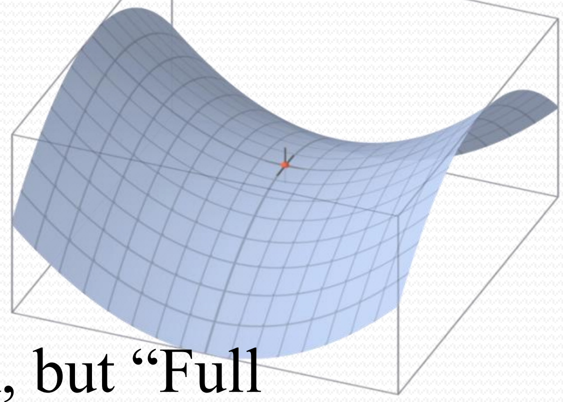
$$\boldsymbol{\gamma}_t^{\text{new}} = \mathbf{g}^{-1}\left(\frac{1}{2}\left[\mathbf{g}(\boldsymbol{\alpha}_t + \tilde{\boldsymbol{\beta}}_t) + \mathbf{g}(\tilde{\boldsymbol{\alpha}}_t + \boldsymbol{\beta}_t)\right]\right)$$

Damped Expectation Propagation

- Minimization of the free energy under the expectation constraints is equivalent to "Saddle Point" problem.

$$\min_{\boldsymbol{\gamma}} \max_{\boldsymbol{\delta}} F(\boldsymbol{\gamma}, \boldsymbol{\delta}) \ \text{with} \ F(\boldsymbol{\gamma}, \boldsymbol{\delta}) \equiv F_0(\boldsymbol{\gamma}) + F_1(\boldsymbol{\gamma}, \boldsymbol{\delta})$$

$$\text{and } F_0(\boldsymbol{\gamma}) = \sum_{t=1}^{T-1} \log \int d\mathbf{x}_t \, e^{\boldsymbol{\gamma}_t^T \mathbf{f}(\mathbf{x}_t)} \, .$$



- Double-loop algorithm solves this problem, but "Full completion in the inner loop is required to guarantee convergence"
- Gradient descent-ascent behavior can be achieved by damping the full updates in EP: $\boldsymbol{\alpha}_t = \tilde{\boldsymbol{\alpha}}_t$ $\quad \boldsymbol{\beta}_t = \tilde{\boldsymbol{\beta}}_t$
- Stable fixed points of damped EP must be at least local minima of Bethe free energy

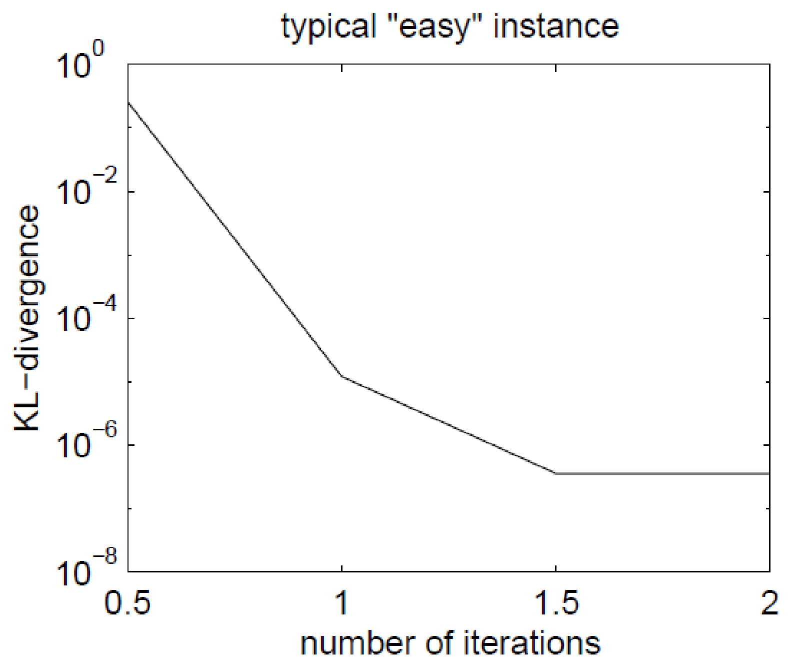
Simulations

- Randomly generated switching linear dynamical systems.
  - T varied between 2 and 5, number of switches between 2 and 4

- "Exact" beliefs calculated using an algorithm by (Lauritzen, 1992) using a strong junction tree.
  - Compared approximate algorithm beliefs to exact beliefs using KL divergence.

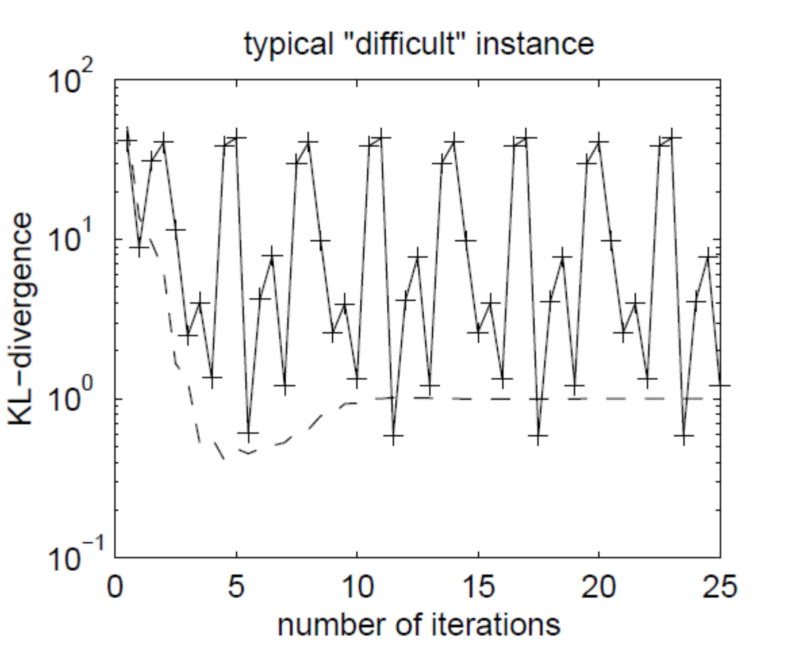
$$\sum_{t=1}^{T} \mathrm{KL}(P_t | \hat{P}_t)$$

Simulation Results



- Undamped EP
  - One forward pass yields acceptable results
  - KL drops after 1 to 2 more passes
  - Double-loop and damped EP converge to same point

Simulation Results



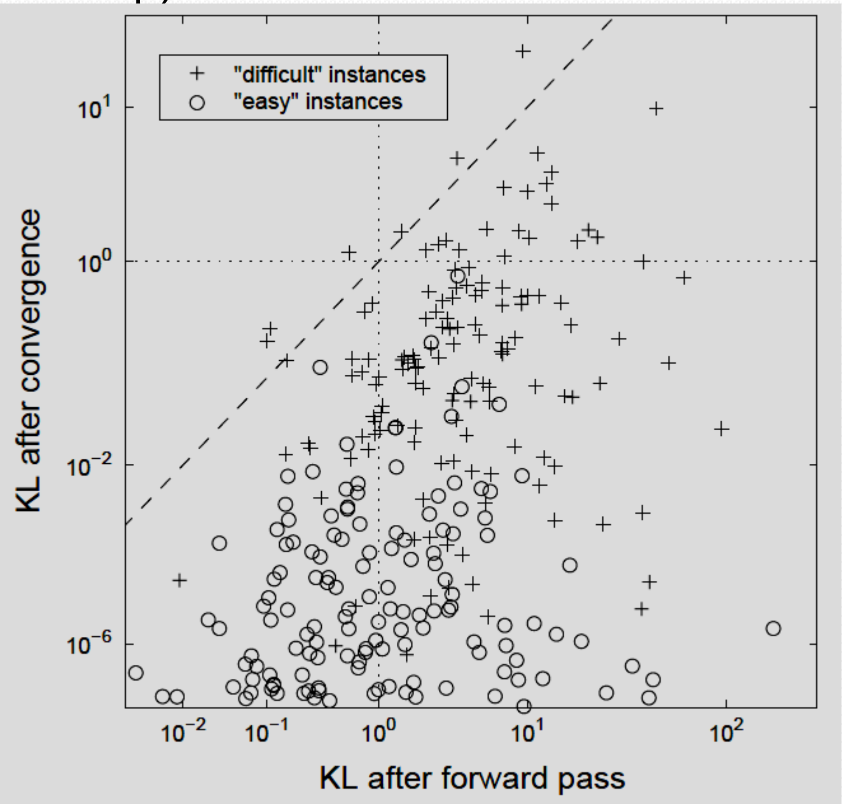
- “Difficult Instance”
  - Undamped stuck in a limit cycle (solid line)
  - Damped EP ($\varepsilon = 0.5$), allows stable convergence
  - Double-loop converges but usually takes longer

Non Convergence

- One Instance where damped EP did not converge
  - Does it make sense to force convergence using double-loop?
  - Compared KL divergence after a single forward pass and after convergence

For “easy” (damped EP) and “difficult” (double-loop)

- Conclude:
  - It makes sense to search for the minimum of the free energy using more exhaustive means.
  - Convergence of undamped belief propagation is an indication of the quality of an approximation



Conclusion

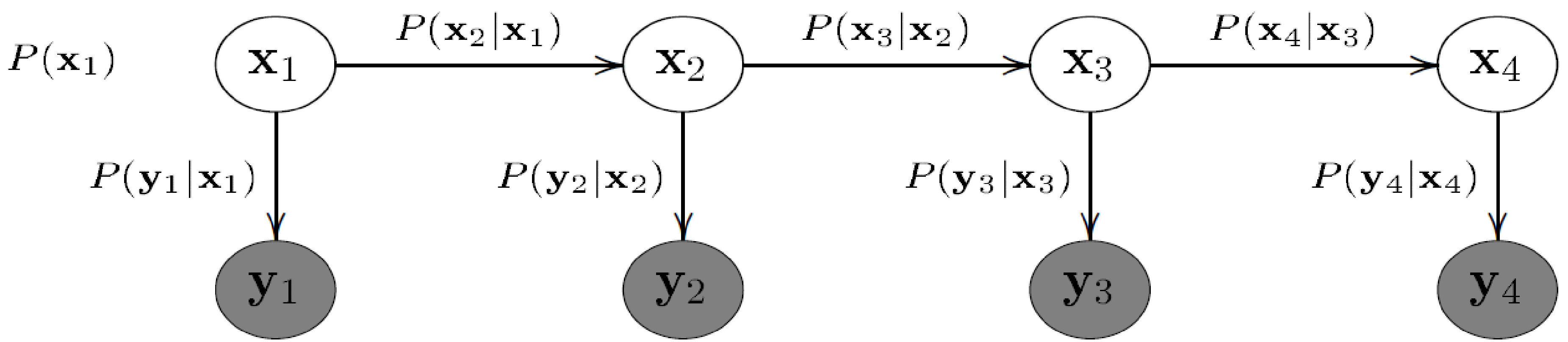
- Introduced a belief propagation algorithm for DBN that is symmetric for both forwards and backward messages
- Project beliefs and derive messages from approximate beliefs rather than approximate messages
- Derived double-loop algorithm guaranteed to converge
- Derived damped EP as a single-loop version
  - Property that when it converges this must be a minimum of Bethe free energy.
  - Thus minimum KL divergence for approximation
- Undamped EP works well in many cases
  - When it fails could be due to:
  - Need for damping
  - Need for “more tedious” double-loop algorithm

The Factored Frontier Algorithm for Approximate Inference in DBNs

Kevin Murphy and Yair Weiss

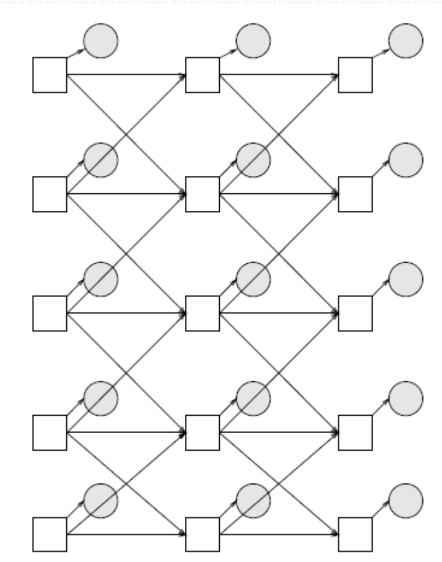
Presented by Mark Buller

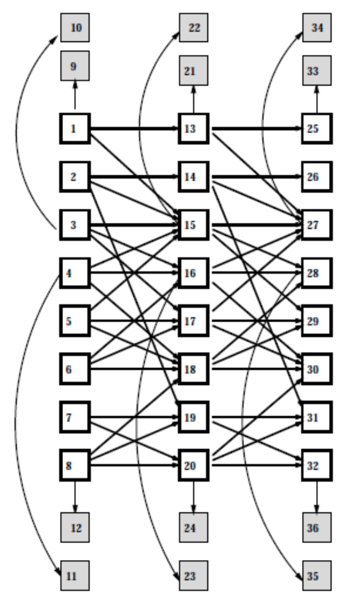
Dynamic Bayesian Networks



- Directed graphical models of stochastic processes
- Represent hidden and observed variables with different dependencies
- Generalize Hidden Markov Models (HMM)

Goal is Inference





- Fart Left coupled HMM with 5 chains
- Left DBN to monitor waste water treatment plant.
- Murphy and Weiss 2001

- Will generally like to perform inference: $P(\mathbf{x}_t \mid \mathbf{y}_{1:T})$
- Why not discretize and use the "Forward-Backward" algorithm?
- $O(TS^2)$, S=num states

Forwards Backward Algorithm

$$\alpha_t^i \stackrel{def}{=} P(X_t = i \mid y_{1:t})$$

$$\beta_t^i \stackrel{def}{=} P(X_t = i \mid y_{t+1:T})$$

$$\gamma_t^i \stackrel{def}{=} P(X_t = i \mid y_{1:T}) \propto \alpha_t^i \beta_t^i$$

Transition Matrix

$$M(i,j) \stackrel{def}{=} P(X_{t+1} = j \mid X_t = i)$$

Diagonal Evidence Matrix

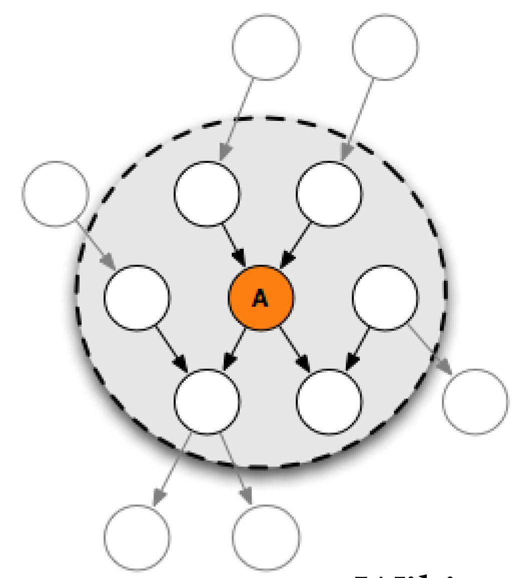
$$W_t(i,i) \stackrel{def}{=} P(y_t \mid X_t = i)$$

$$\alpha_t \propto W_t M^T \alpha_{t-1}$$

$$\beta_t \propto W_{t+1} M \beta_{t+1}$$

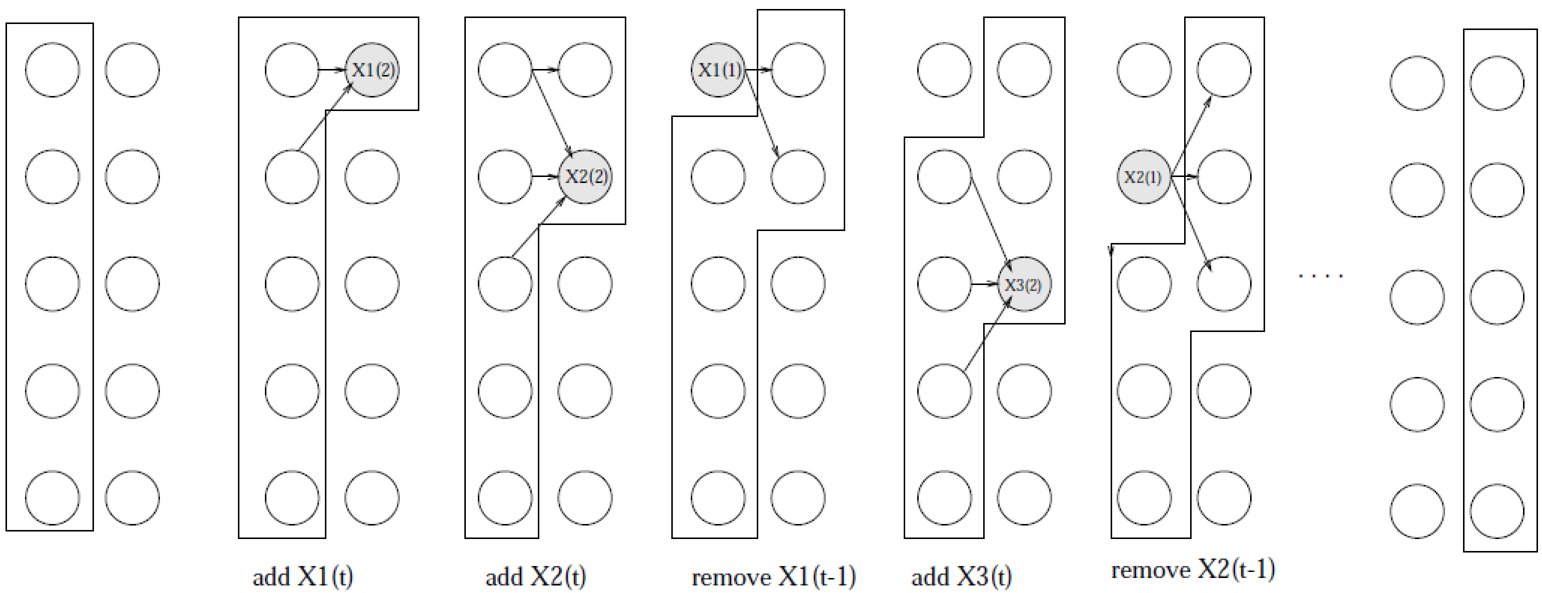
Frontier Algorithm

- Method to compute α_t and β_ts without the need to form the Q^N x Q^N transition matrix:
  - N = number of hidden nodes
  - Q = number possible states of a node
- "Sweep" a Markov Blanket forwards then backwards across the DBN.
  - The set of nodes composed of a node's the parents, children, and children's other parents.
  - Every other node is conditionally independent of A when conditioned on A's Markov blanket.



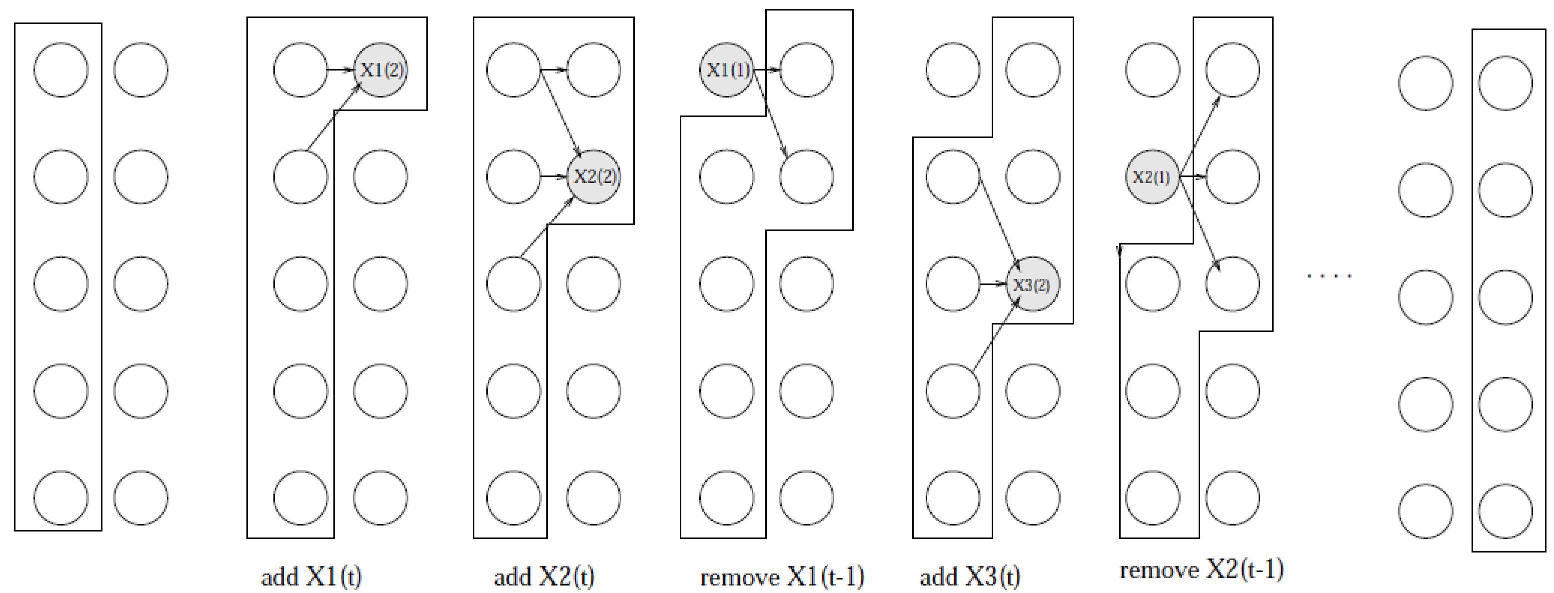
Wikipedia

Frontier Algorithm



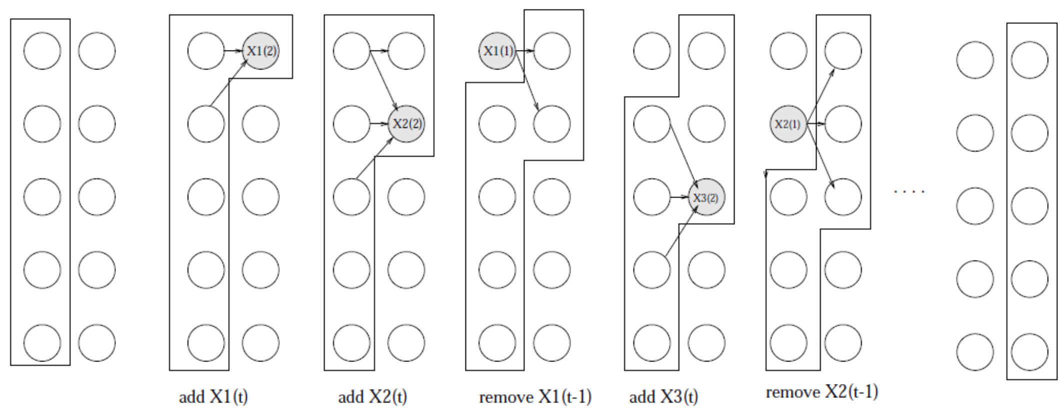
- *F* "Frontier Set" = Nodes in Markov Blanket, Nodes to left = *L*, Nodes to right = *R*.
- At every step *F "d-separates" L* and *R*.
- A joint distribution over nodes in *F* is maintained.

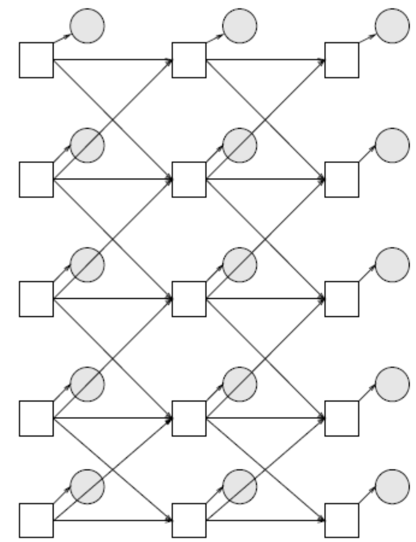
Frontier Algorithm



- A node is added from *R* to *F* as soon as all parents are in *F*
  - To add a node multiply by conditional probability table (CPT)
- A node is moved from *F* to *L* as soon as all children are in *F*
  - To remove a marginalize by the removed node.

Frontier Algorithm





Add X(1)t

$$F_{t,0} \stackrel{\text{def}}{=} \alpha_{t-1} = P(X_{t-1}^{1:N} | y_{1:t-1})$$

$$F_{t,1} = P(X_t^1, X_{t-1}^{1:N} | y_{1:t-1}) = P(X_t^1 | X_{t-1}^1, X_{t-1}^2) \times F_{t,0}$$

Add X(2)t

$$\begin{aligned} F_{t,2} &= P(X_t^{1:2}, X_{t-1}^{1:N} | y_{1:t-1}) \\ &= P(X_t^2 | X_{t-1}^1, X_{t-1}^2, X_{t-1}^3) \times F_{t,1} \end{aligned}$$

Rem X(1)t-1

$$F_{t,3} = P(X_t^{1:2}, X_{t-1}^{2:N} | y_{1:t-1}) = \sum_{X_{t-1}^1} F_{t,2}$$

$$F_{t,N} = P(X_t^{1:N} | y_{1:t-1})$$

Forward Message

$$\alpha_t = P(X_t^{1:N} | y_{1:t}) \propto P(y_t | X_t^{1:N}) \times F_{t,N}$$

Frontier Algorithm (Observations)

- Exact Inference takes $O(TNQ^{N+2})$ time and space:
  - N = number of hidden nodes
  - Q = number possible states of a node
- Exponential in the size of the largest frontier
  - Optimal ordering of additions and removals to minimize *F* is NP-Hard.
- For regular DBNs when unrolled, the frontier algorithm is equivalent to the junction tree algorithm.
  - Frontier sets correspond to: maximal cliques in the moralized triangulated graph.

Factored Frontier Algorithm

- Approximate the belief state with a product of marginals:

$$P(X_t \mid y_{1:t}) \approx \prod_{i=1}^{N} P(X_t^i \mid y_{1:t})$$

- When a node is added the node's CPT is multiplied by the product of factors corresponding to its parents.
  - Joint distribution for the family
  - Parent nodes are immediately marginalized out
  - Can be done for any node in any order as long as parents are added first.
- Joint distribution over frontier nodes is maintained in factored form.
- Takes $O(TNQ^{F+1})$

Boyen-Koller Algorithm

- Belief state with a product of marginals over C clusters:

$$P(X_t \mid y_{1:t}) \approx \prod_{c=1}^{C} P(X_t^c \mid y_{1:t})$$

- Where X_t^c is a subset of the variables $\{X_t^i\}$
  - Accuracy depends on size of clusters used to approximate belief state
  - Exact inference corresponds to using a single cluster with all hidden variables at a time slice
  - Most aggressive approximation uses N clusters one per variable
    - very similar to FF

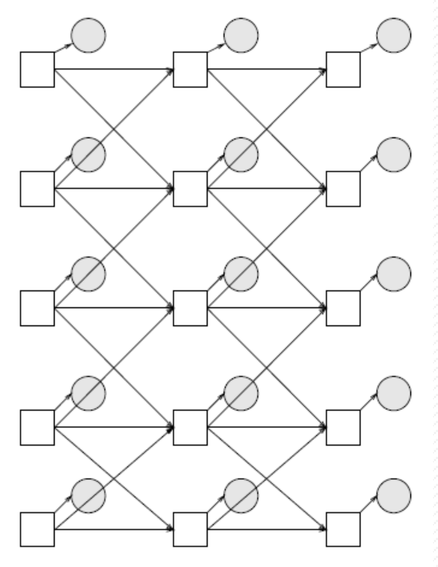
BK and FF as Special Cases of Loopy Belief Propagation

- Pearl's belief propagation algorithm computes exact marginal posterior probabilities in graphs without cycles
- Generalizes the forward-backward algorithm to trees.
- Assumes messages coming into a node are independent.
  - FF makes the same assumption
  - Both algorithms are equivalent if the order of messages in LBP is specified
    - Normally LBP every node computes λ and π messages in parallel and then sends out to all of the neighbors
    - However, messages can be computed in a forwards backward approach. First send π (α) from left to right, then send λ (β) messages from right to left.
  - FF and BK are equivalent to one iteration LBP, thus they can be improved by iterating more than once.

Experiments

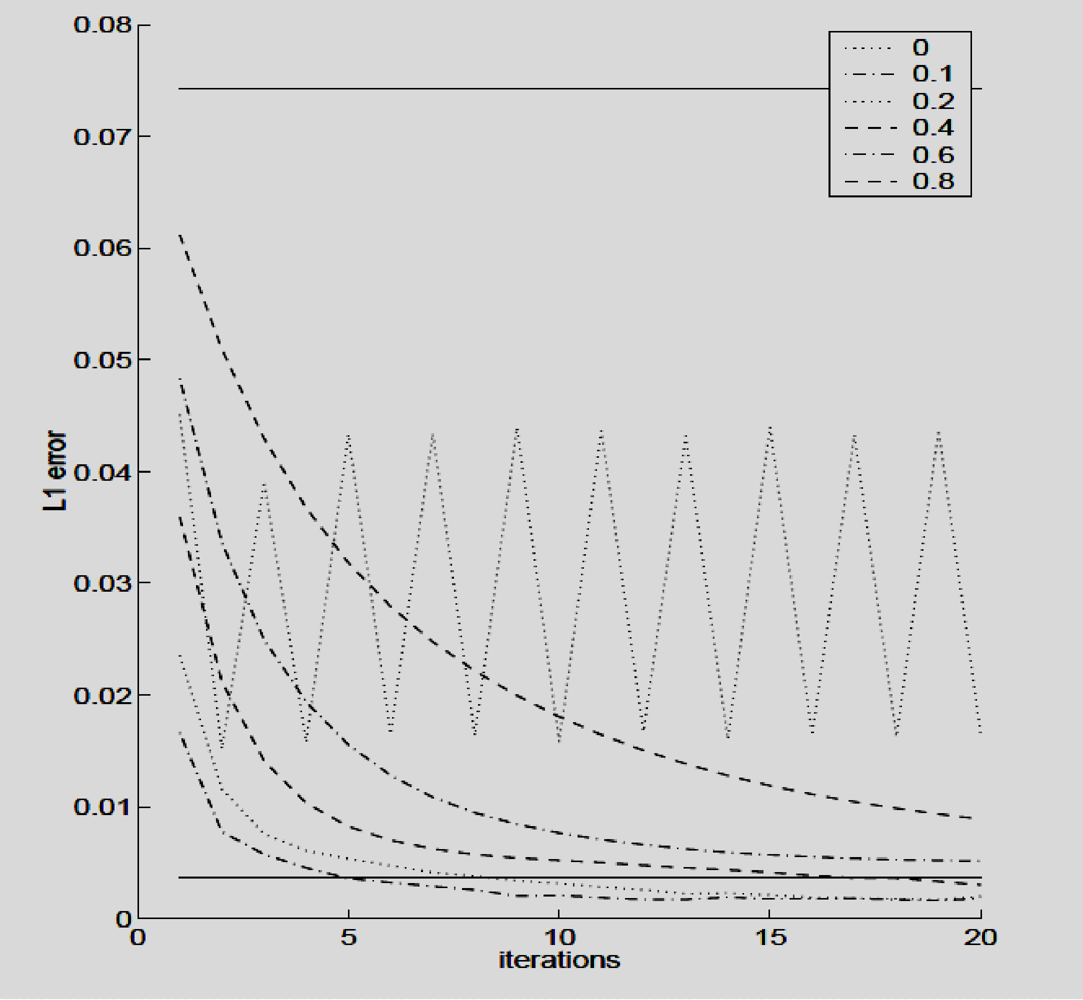
- Used a coupled HMM (CHMM) with 10 chains trained with real highway data.
- Define L1 error as:

$$\Delta_t = \sum_{i=1}^{N} \sum_{s=1}^{Q} | P(X_i^t = s \mid y_{1:T}) - \hat{P}(X_i^t = s \mid y_{1:T}) |$$

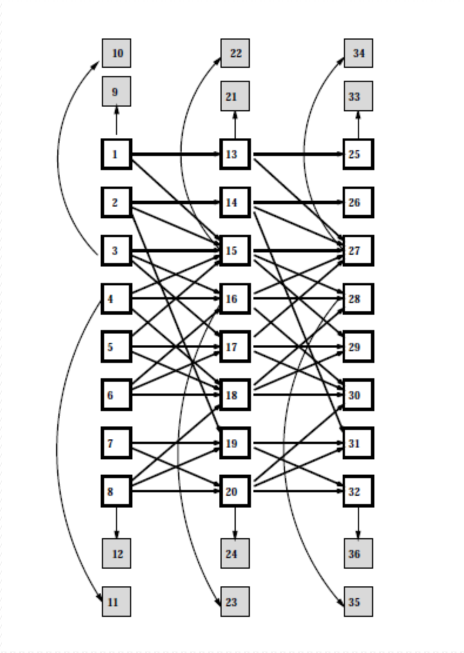


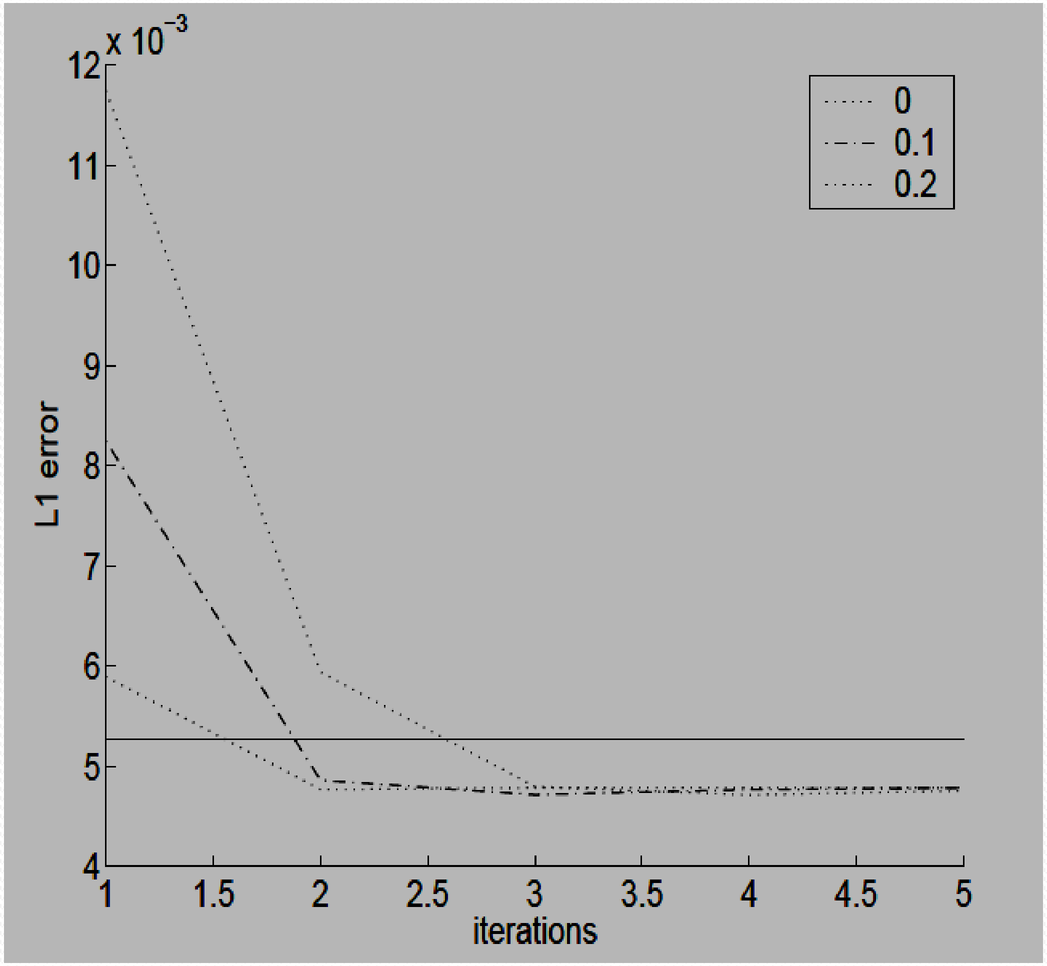
Results

- Damping was necessary with LBP.
- Iterating with damped LBP improves just a single run of BK



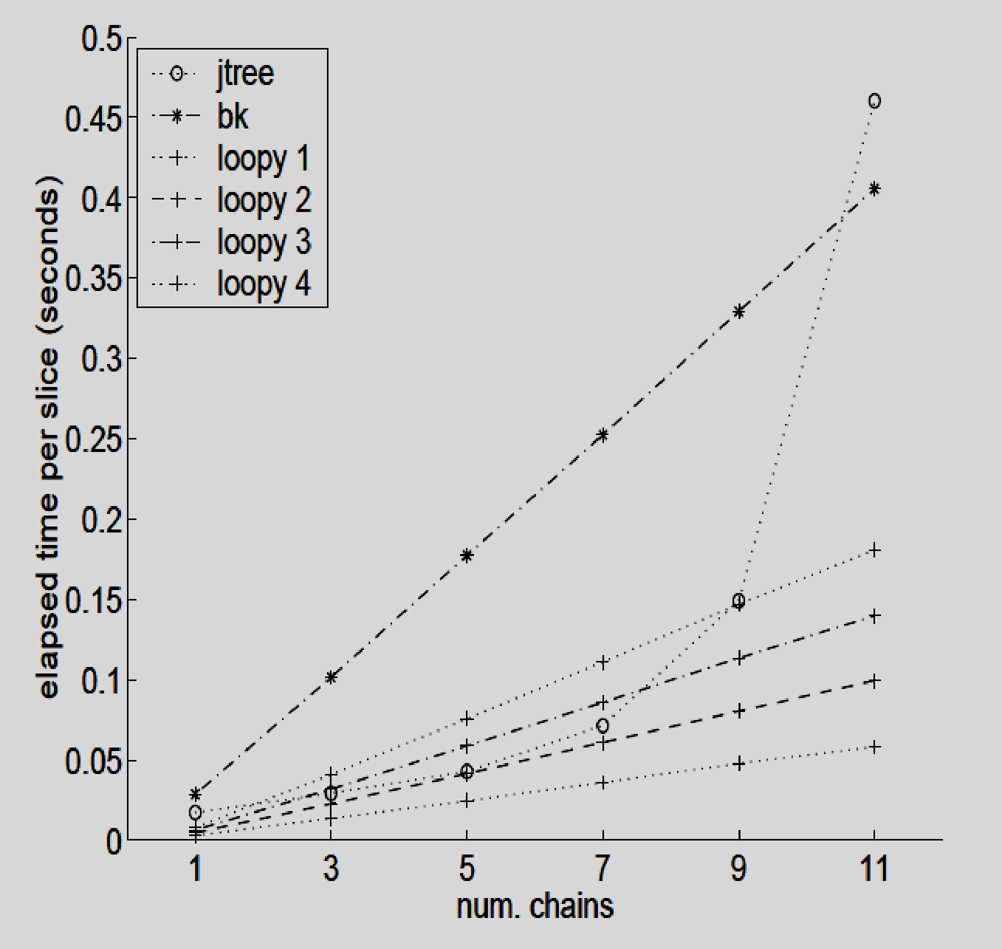
Results Water Network





Results Speed

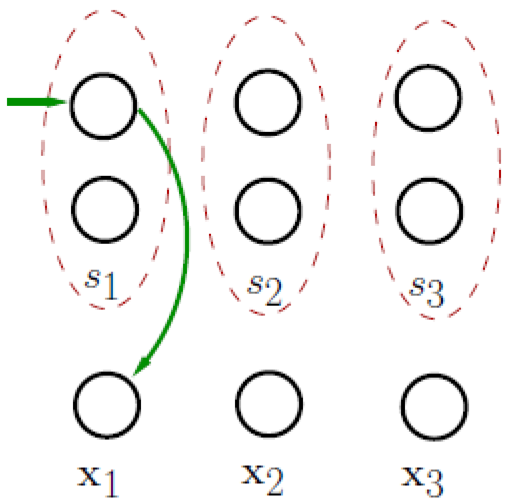
- BK and FF / LBP have a running time linear in N
- BK is slower because of repeated marginalizations
  - When N<11 BK slower than exact inference



Conclusions

- Described a simple approximate inference algorithm for DBNs and shown equivalence to LBP
- Shown a connection between BK and LBP
- Showed empirically that LBP can improve FF and BK.

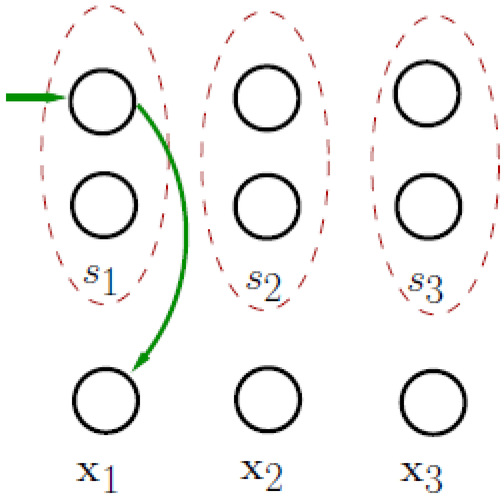
Computing forward probabilities: $t = 1$



$$\alpha_1(1) = p(\mathbf{x}_1, s_1 = 1)$$

Shakhnarovich 1996,CS195-5

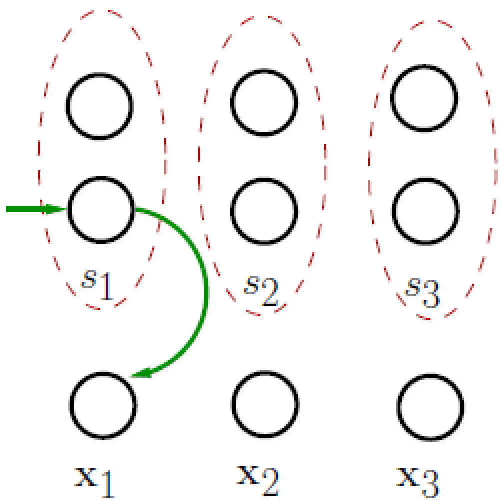
Computing forward probabilities: $t = 1$



$$\alpha_1(1) = p(\mathbf{x}_1, s_1 = 1) = p_0(1)p(\mathbf{x}_1 \mid s_1 = 1);$$

Shakhnarovich 1996,CS195-5

Computing forward probabilities: $t = 1$

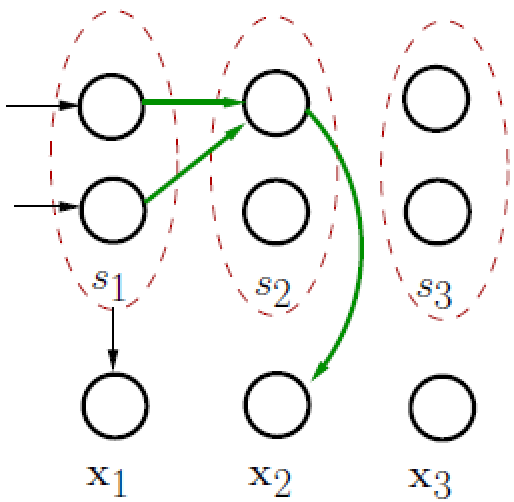


$$\alpha_1(1) = p(\mathbf{x}_1, s_1 = 1) = p_0(1) p(\mathbf{x}_1 \mid s_1 = 1);$$

$$\alpha_1(2) = p(\mathbf{x}_1, s_1 = 2) = p_0(2) p(\mathbf{x}_1 \mid s_1 = 2);$$

Shakhnarovich 1996,CS195-5

Computing forward probabilities: $t = 2$

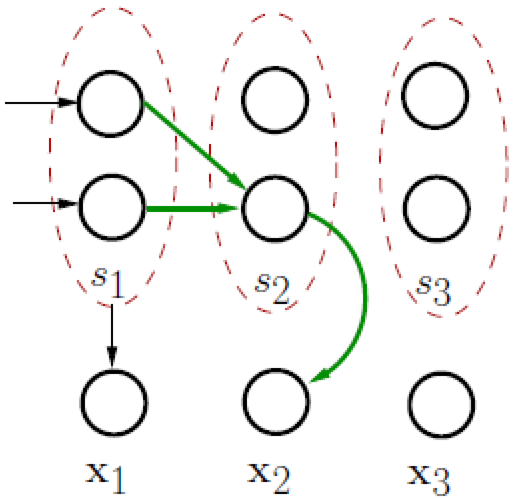


$$\alpha_1(1) = p(x_1, s_1 = 1) = p_0(1)p\left(x_1 \,|\, s_1 = 1\right),$$
$$\alpha_1(2) = p(x_1, s_1 = 2) = p_0(2)p\left(x_1 \,|\, s_1 = 2\right)$$

$$\alpha_2(1) = p(x_1, x_2, s_2 = 1)$$

Shakhnarovich 1996,CS195-5

Computing forward probabilities: $t = 2$



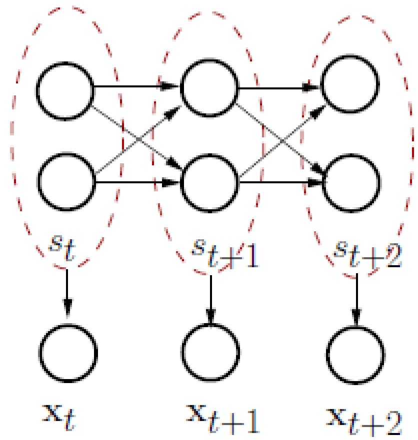
$$\alpha_1(1) = p(\mathbf{x}_1, s_1 = 1) = p_0(1)p(\mathbf{x}_1 \mid s_1 = 1),$$
$$\alpha_1(2) = p(\mathbf{x}_1, s_1 = 2) = p_0(2)p(\mathbf{x}_1 \mid s_1 = 2)$$

$$\begin{aligned}\alpha_2(1) &= p(\mathbf{x}_1, \mathbf{x}_2, s_2 = 1) = p(\mathbf{x}_1, s_2 = 1)p(\mathbf{x}_2 \mid s_2 = 1) \\ &= [\alpha_1(1)p(1 \to 1) + \alpha_1(2)p(2 \to 1)]\, p(\mathbf{x}_2 \mid s_2 = 1) \\ \alpha_2(2) &= [\alpha_1(2)p(1 \to 2) + \alpha_1(2)p(2 \to 2)]\, p(\mathbf{x}_2 \mid s_2 = 2)\end{aligned}$$

Shakhnarovich 1996,CS195-5

Forward probabilities: recursion

$$\alpha_t(s) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t,\, s_t = s)$$



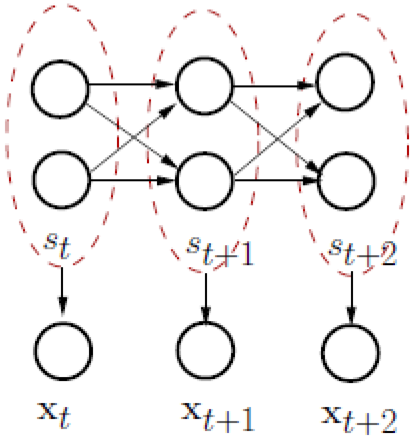
$$\alpha_1(s) = p_0(s)p\left(\mathbf{x}_1 \,|\, s_1 = s\right)$$

$$\alpha_t(s) = \left[\sum_{s'} \alpha_{t-1}(s')p(s' \rightarrow s)\right] p\left(\mathbf{x}_t \,|\, s_t = s\right)$$

Shakhnarovich 1996,CS195-5

Backward probabilities

$$\beta_t(s) \triangleq p\left(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_N \mid s_t = s\right)$$

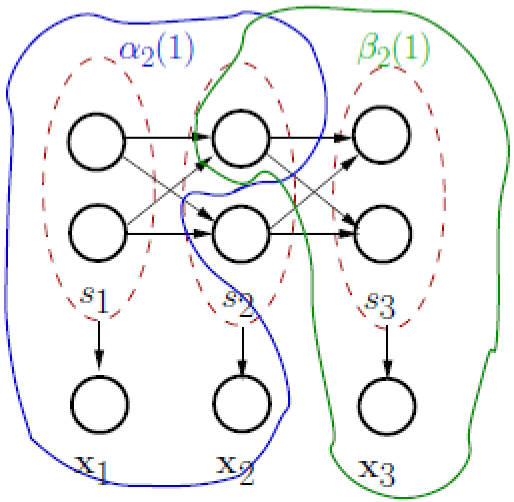


$$\beta_N(s) = p\left(\varnothing \mid s_N = s\right) \triangleq 1$$

$$\beta_t(s) = \sum_{s'} \left[p(s \rightarrow s') p\left(\mathbf{x}_{t+1} \mid s_{t+1} = s'\right) \beta_{t+1}(s')\right]$$

Shakhnarovich 1996,CS195-5

State posterior probability



$$\gamma_t(s) \triangleq p\left(s_t = s \mid \mathbf{x}_1, \ldots, \mathbf{x}_N\right) = \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_N,\, s_t = s)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_N)}$$

$$= \frac{\overbrace{p(\mathbf{x}_1, \ldots, \mathbf{x}_t,\, s_t = s)}^{\alpha_t(s)}\overbrace{p\left(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_N \mid s_t = s\right)}^{\beta_t(s)}}{p(\mathbf{x}_1, \ldots, \mathbf{x}_N)}$$

$$= \frac{\alpha_t(s)\beta_t(s)}{\sum_s' \alpha_t(s')\beta_t(s')}$$

Shakhnarovich 1996,CS195-5