Probabilistic Graphical Models

Lecture 13 – Loopy Belief Propagation

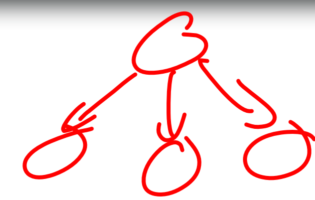
CS/CNS/EE 155
Andreas Krause

Announcements

- Homework 3 out
  - Lighter problem set to allow more time for project
- Next Monday: Guest lecture by **Dr. Baback Moghaddam** from the **JPL Machine Learning Group**
- PLEASE fill out feedback forms
  - This is a new course
  - Your feedback can have major impact in future offerings!!

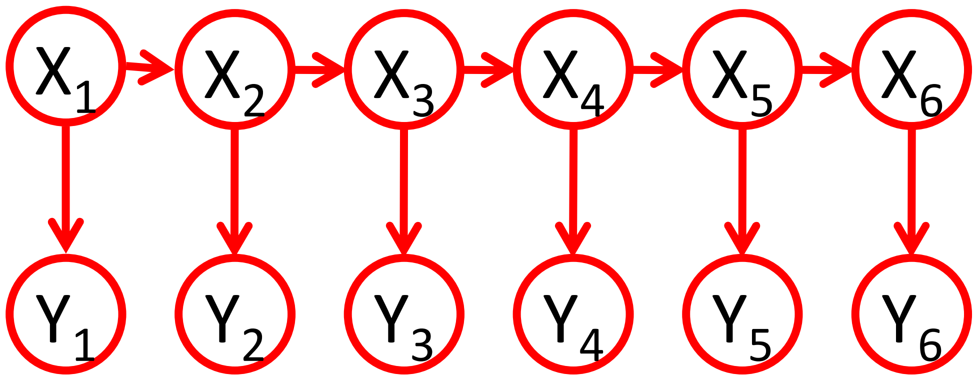
HMMs / Kalman Filters

- Most famous Graphical models:
  - Naïve Bayes model
  - Hidden Markov model
  - Kalman Filter



- Hidden Markov models
  - Speech recognition
  - Sequence analysis in comp. bio
- Kalman Filters control
  - Cruise control in cars
  - GPS navigation devices
  - Tracking missiles..
- Very simple models but very powerful!!

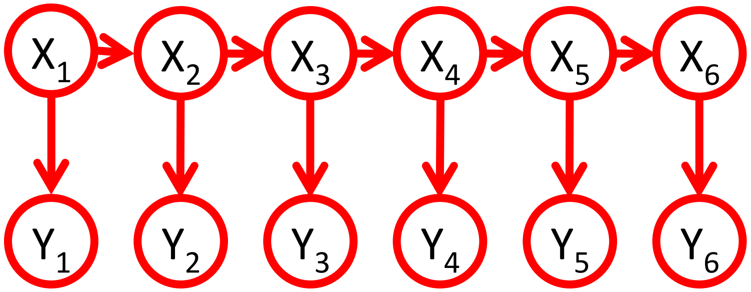
HMMs / Kalman Filters



- $X_1,\ldots,X_T$: Unobserved (hidden) variables
- $Y_1,\ldots,Y_T$: Observations
- HMMs: X_i Multinomial, Y_i arbitrary
- Kalman Filters: X_i, Y_i Gaussian distributions
  - Non-linear KF: X_i Gaussian, Y_i arbitrary

Hidden Markov Models

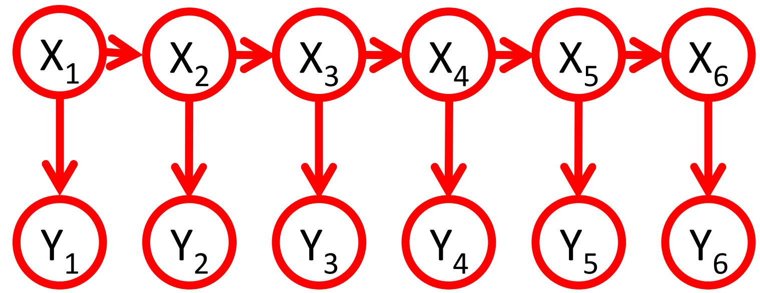
- Inference:
  - In principle, can use VE, JT etc.
  - New variables X_t, Y_t at each time step ➔ need to rerun



- Bayesian Filtering:
  - Suppose we already have computed $P(X_t \mid y_{1,\ldots,t})$
  - Want to efficiently compute $P(X_{t+1} \mid y_{1,\ldots,t+1})$

Bayesian filtering

- Start with $P(X_1)$
- At time t
  - Assume we have $P(X_t \mid y_{1...t-1})$
  - Condition: $P(X_t \mid y_{1...t})$



$$P(X_t \mid y_{1...t}) \propto P(X_t \mid y_{1..t-1}) \underbrace{P(Y_t \mid X_t, y_{1..t-1})}_{\text{cond. ind. } P(Y_t \mid X_t)}$$

  - Prediction: $P(X_{t+1}, X_t \mid y_{1...t})$

$$P(X_{t+1}, X_t \mid y_{1..t}) = P(X_t \mid y_{1..t}) \cdot \underbrace{P(X_{t+1} \mid X_t, y_{1..t})}_{= P(X_{t+1} \mid X_t)}$$

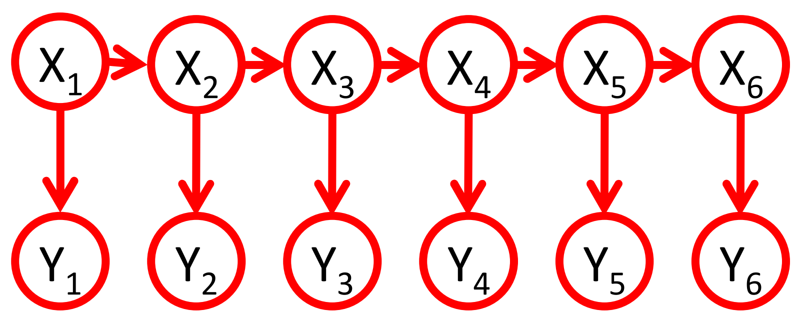
"Rollup"

  - Marginalization: $P(X_{t+1} \mid y_{1...t})$

$$P(X_{t+1} \mid y_{1..t}) = \sum_{x_t} P(X_{t+1}, x_t \mid y_{1..t})$$

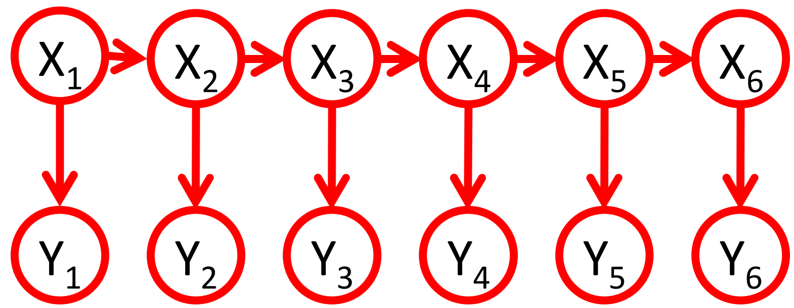
Kalman Filters (Gaussian HMMs)

- $X_1,\ldots,X_T$: Location of object being tracked
- $Y_1,\ldots,Y_T$: Observations
- $P(X_1)$: Prior belief about location at time 1
- $P(X_{t+1}|X_t)$: "Motion model"
  - How do I expect my target to move in the environment?
  - Represented as CLG: $X_{t+1} = A X_t + N(0, \Sigma_M)$
- $P(Y_t \mid X_t)$: "Sensor model"
  - What do I observe if target is at location X_t?
  - Represented as CLG: $Y_t = H X_t + N(0, \Sigma_O)$



Bayesian Filtering for KFs

- Can use Gaussian elimination to perform inference in "unrolled" model



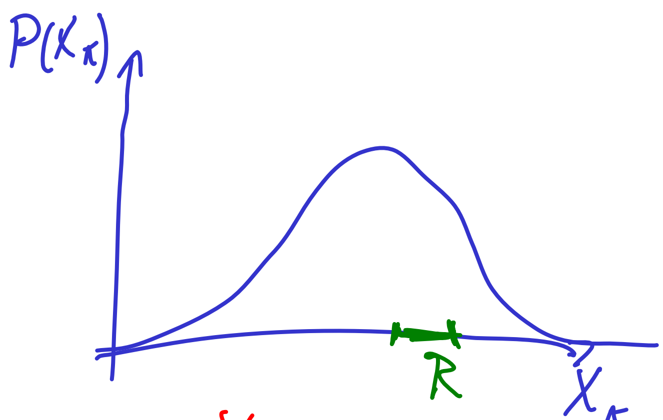
- Start with prior belief $P(X_1)$ $\mathcal{N}(x_t; \eta_t, \Lambda_t)$
- At every timestep have belief $P(X_t \mid y_{1:t-1})$ "sensor model"
  - Condition on observation: $P(X_t \mid y_{1:t})$ Multiply likelihood
  - Predict (multiply motion model): $P(X_{t+1}, X_t \mid y_{1:t})$ Multiply motion model
  - "Roll-up" (marginalize prev. time): $P(X_{t+1} \mid y_{1:t})$

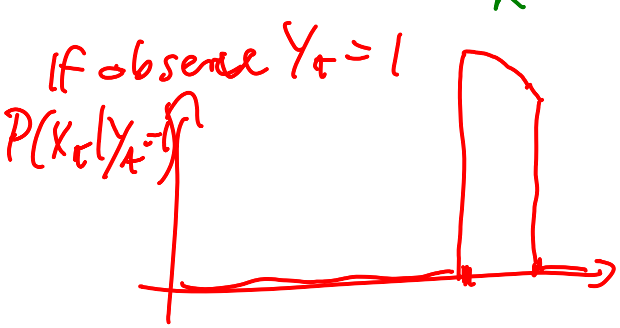
What if observations not “linear”?

- Linear observations:
  - $Y_t = H X_t + \text{noise}$
- Nonlinear observations:

“Motion detector”: $Y_t = 1$ if $X_t \in R$

$= 0$ otherwise



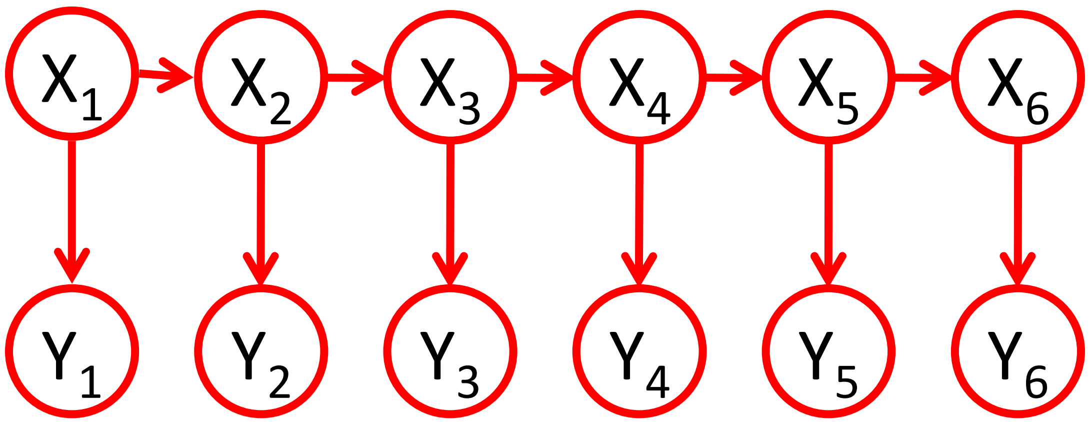


Incorporating Non-gaussian observations

- Nonlinear observation → $P(Y_t \mid X_t)$ not Gaussian ☹
- Make it Gaussian! ☺
- First approach: Approximate $P(Y_t \mid X_t)$ as CLG
  - Linearize $P(Y_t \mid X_t)$ around current estimate $E[X_t \mid y_{1..t-1}]$
  - Known as Extended Kalman Filter (EKF)
  - Can perform poorly if $P(Y_t \mid X_t)$ highly nonlinear
- Second approach: Approximate $P(Y_t, X_t)$ as Gaussian
  - Takes correlation in X_t into account
  - After obtaining approximation, condition on $Y_t=y_t$ (now a "linear" observation)

Factored dynamical models

- So far: HMMs and Kalman filters

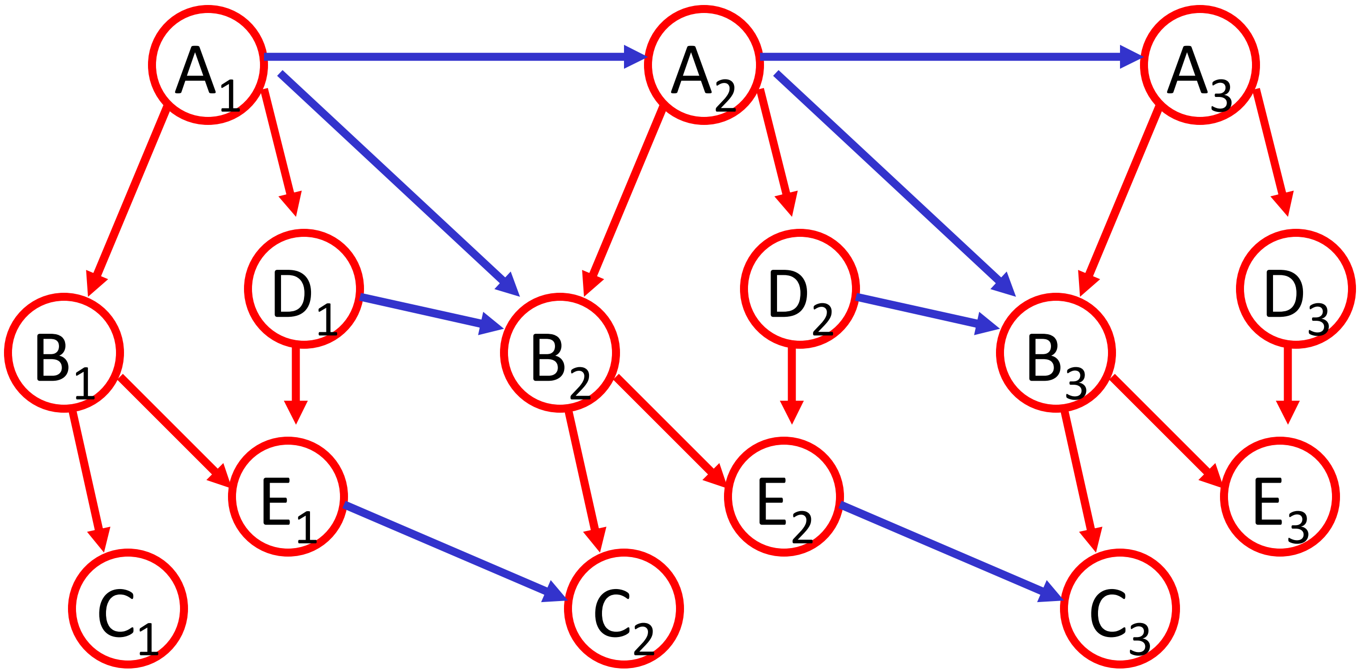


- What if we have more than one variable at each time step?
  - E.g., temperature at different locations, or road conditions in a road network?

  ➔ Spatio-temporal models

Dynamic Bayesian Networks

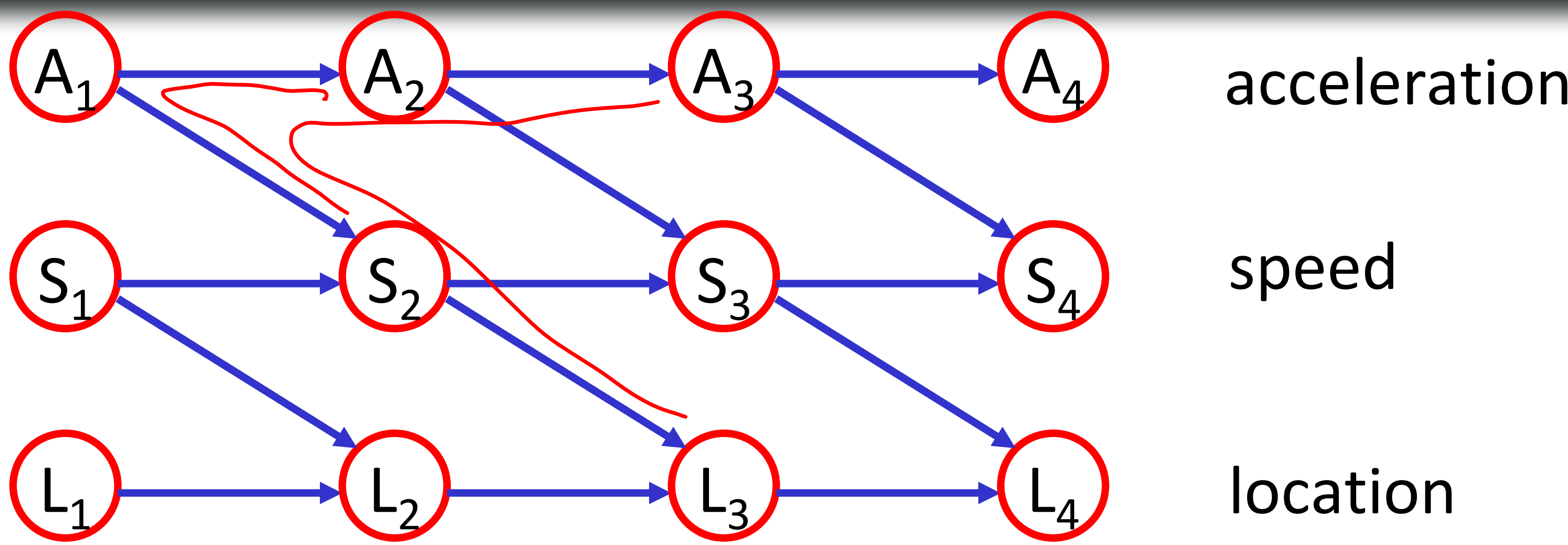
- At every timestep have a Bayesian Network



$S_1 = \{A_1, B_1, \ldots, E_1\}$

- Variables at each time step t called a "slice" $\mathbf{S_t}$
- "Temporal" edges connecting S_{t+1} with S_t

Flow of influence in DBNs



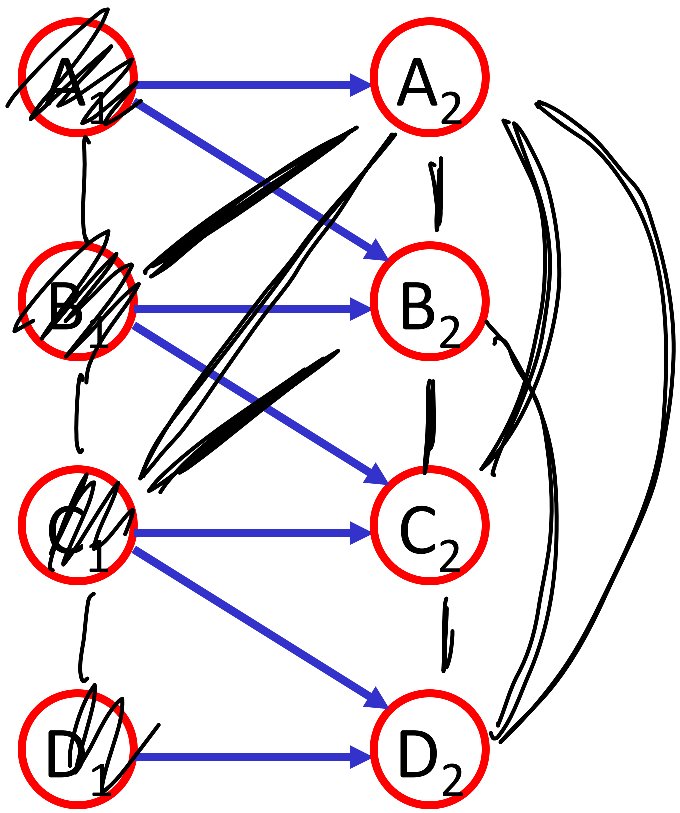
$A_1 \perp S_1$ ✓ $\quad$ $A_2 \perp S_2$ ✗

$A_1 \perp L_1$ ✓ $\quad$ $A_2 \perp L_2$ ✓ $\quad$ $A_3 \perp L_3$ ✗

- Can we do efficient filtering in BNs?

Efficient inference in DBNs?

DBN

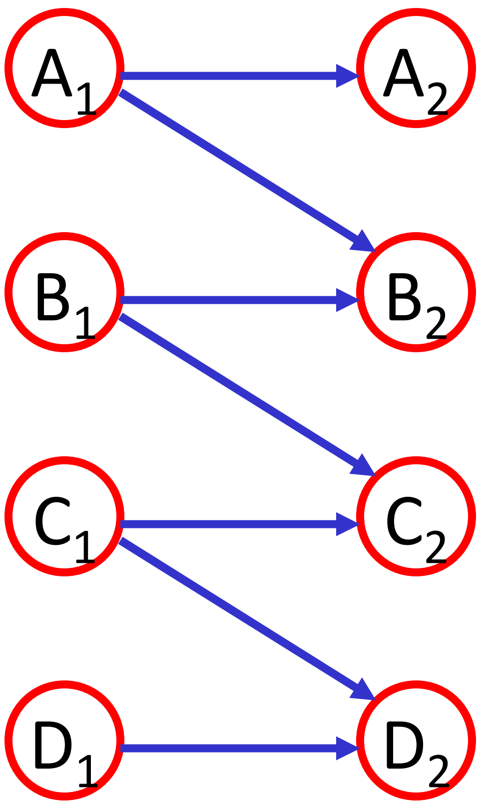


$P(A_2, B_2, C_2, D_2)$

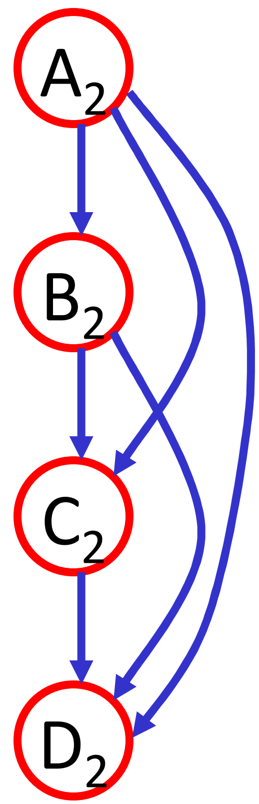
fully connected :(

Approximate inference in DBNs?

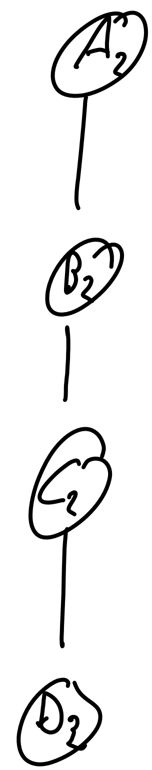
DBN



Marginals at time 2



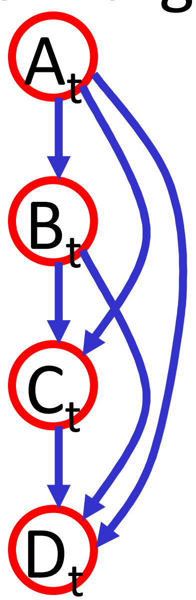
Approximate marginal



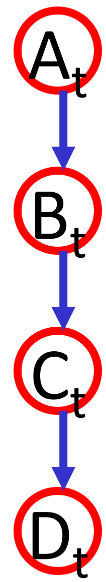
How can we find principled approximations that still allow efficient inference??

Assumed Density Filtering

True marginal



Approximate marginal



Formally:

$$Q^* = \arg\min_Q KL(P \| Q)$$

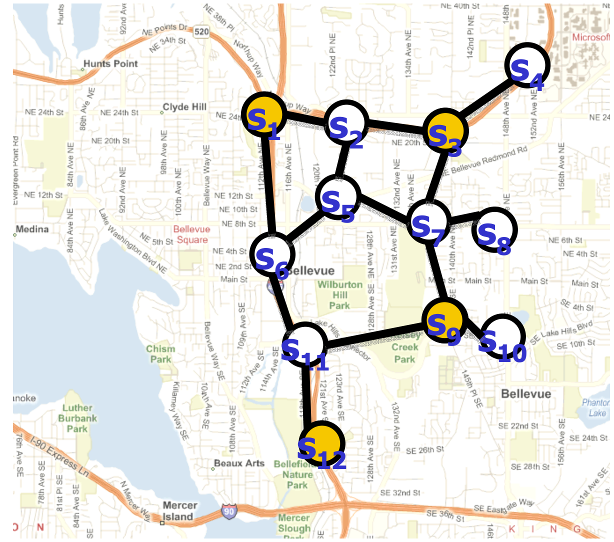
more later

- True marginal $P(X_t)$ fully connected
- Want to find "simpler" distribution $Q(X_t)$ such that $P(X_t) \approx Q(X_t)$
- Optimize over parameters of Q to make Q as "close" to P as possible
- Similar to incorporating non-linear observations in KF!
- More details later (variational inference)!

Big picture summary


States of the world,
sensor measurements, …

represent



Graphical model

- Want to choose a model that …
  - **represents** relevant statistical dependencies between variables
  - we can use to make **inferences** (make predictions, etc.)
  - we can **learn** from training data

What you have learned so far

- **Representation**
  - Bayesian Networks
  - Markov Networks
  - Conditional independence is key
- **Inference**
  - Variable Elimination and Junction tree inference
  - Exact inference possible if graph has low treewidth
- **Learning**
  - **Parameters**: Can do MLE and Bayesian learning in Bayes Nets and Markov Nets if data fully observed
  - **Structure**: Can find optimal tree

Representation

- Conditional independence = Factorization
- Represent factorization/independence as graph
  - Directed graphs: Bayesian networks
  - Undirected graphs: Markov networks
- Typically, assume factors in exponential family (e.g., Multinomial, Gaussian, …)

- So far, we assumed **all variables** in the model are **known**
  - In practice
    - Existence of variables can depend on data
    - Number of variables can grow over time
  - We might have hidden (unobserved variables)!

Inference

- **Key idea**: Exploit factorization (distributivity)
- Complexity of inference depends on treewidth of underlying model
  - Junction tree inference "only" exponential in treewidth
- In practice, often have high treewidth
  - Always high treewidth in DBNs
  - ➔ Need approximate inference

Learning

- Maximum likelihood estimation
  - **In BNs**: independent optimization for each CPT (decomposable score)
  - **In MNs**: Partition function couples parameters, but can do gradient ascent (no local optima!)
- Bayesian parameter estimation
  - Conjugate priors convenient to work with
- Structure learning
  - NP-hard in general
  - Can find optimal tree (Chow Liu)

- So far: Assumed all variables are observed
  - In practice: often have missing data

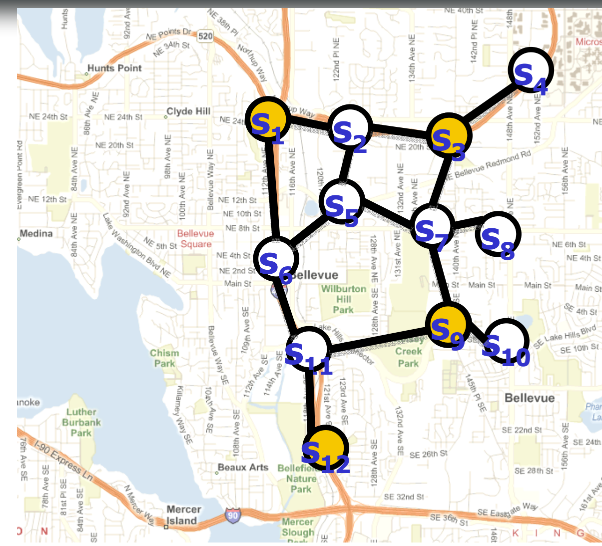
The "light" side

- Assumed
  - everything fully observable
  - low treewidth
  - no hidden variables

- Then everything is nice ☺
  - Efficient exact inference in large models
  - Optimal parameter estimation without local minima
  - Can even solve some structure learning tasks exactly

The "dark" side


represent



States of the world,
sensor measurements, …

Graphical model

- In the real world, these assumptions are often violated..
- Still want to use graphical models to solve interesting problems..

Remaining Challenges

- Representation:
  - Dealing with **hidden variables**
- **Approximate inference** for high-treewidth models
- Dealing with **missing data**

- This will be focus of remaining part of the course!

Recall: Hardness of inference

- Computing conditional distributions:
  - Exact solution: **#P-complete**
  - Approximate solution: **NP-hard**

- Maximization:
  - MPE: **NP-complete**
  - MAP: **NP^{PP}-complete**

Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations
  - Whenever the graph is low treewidth
  - Whenever there is **context-specific independence**
  - Several other special cases
- For BNs where exact inference is not possible, can use algorithms for **approximate inference**
  - Coming up now!

Approximate inference

- Three major classes of general-purpose approaches

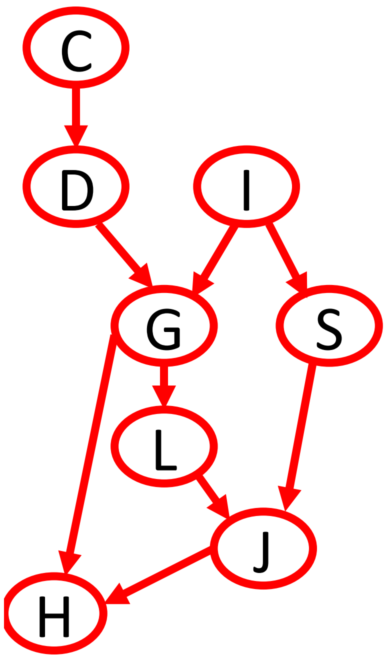
- **Message passing**
  - E.g.: Loopy Belief Propagation (today!)

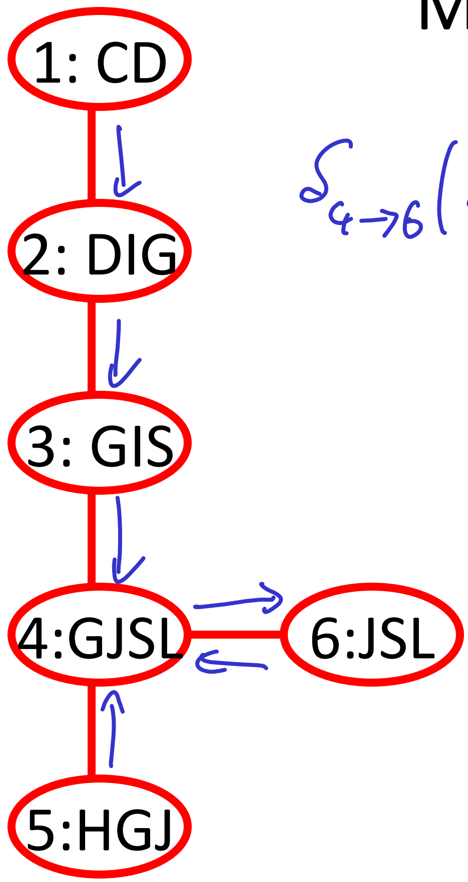
- **Inference as optimization**
  - Approximate posterior distribution by simple distribution
  - Mean field / structured mean field

- **Sampling based inference**
  - Importance sampling, particle filtering
  - Gibbs sampling, MCMC

- Many other alternatives (often for special cases)

Recall: Message passing in Junction trees

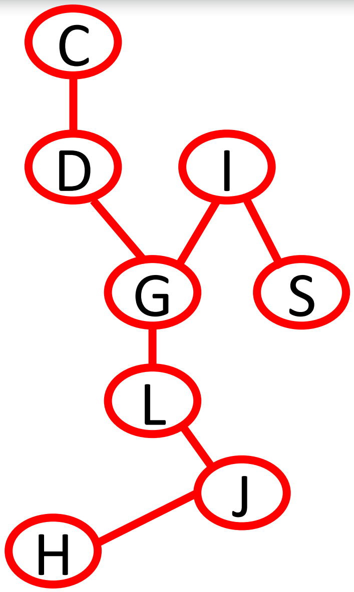




Messages between clusters:

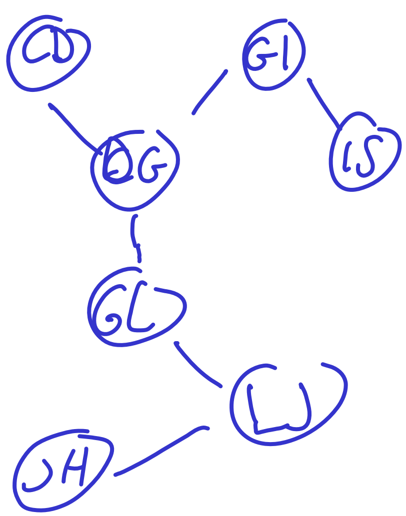
$$\delta_{4\to6}(J,S,L) = \sum_{g} \pi(g,J,S,L)\cdot\delta_{3\to4}(g,S)\cdot\delta_{5\to4}(g,J)$$

BP on Tree Pairwise Markov Nets



- Suppose graph is given as tree pairwise Markov net
- Don't need a junction tree!
  - Graph is already a tree!
- Example message:

$$\delta_{G\to L}(L) = \sum_{g} \pi_{GL}(g,L)\cdot\pi_G(g)\,\delta_{D\to G}(g)\,\delta_{I\to G}(g)$$

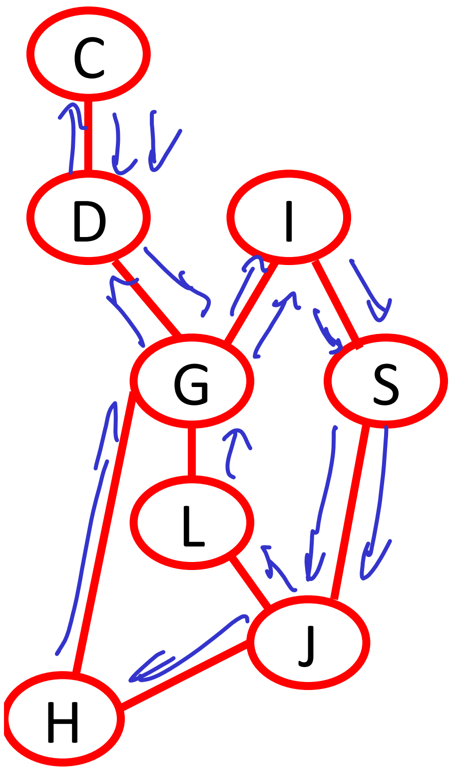


- More generally:

$$\delta_{i\to j}(x_j) = \sum_{x_i} \pi_{ij}(x_i,x_j)\,\pi_i(x_i) \prod_{s\in N(i)\setminus\{j\}} \delta_{s\to i}(x_i)$$

- **Theorem**: For trees, get correct answer!

Loopy BP on arbitrary pairwise MNs



- What if we apply BP to a graph with loops?
  - Apply BP and hope for the best..

$$\delta_{i\to j}(X_j) = \sum_{x_i} \pi_i(x_i)\pi_{i,j}(x_i, X_j) \prod_{s\in N(i)\setminus\{j\}} \delta_{s\to i}(x_i)$$

- Will not generally converge..
- If it converges, will not necessarily get correct marginals

$$\hat{P}(X_i) \propto \prod_{s\in N(i)} \delta_{s\to i}(x_i)$$

- However, in practice, answers often still useful!

Practical aspects of Loopy BP

- Messages product of numbers ≤ 1

$$\delta_{i \to j}(X_j) = \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \to i}(x_i)$$

- On loopy graphs, repeatedly multiply same factors
➔ products converge to 0 (numerical problems)
- Solution:
  - Renormalize!

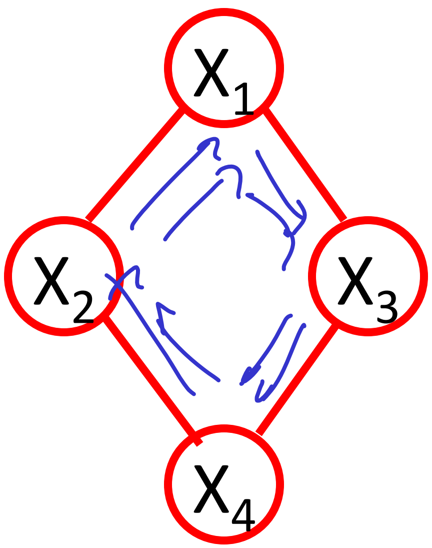
$$\delta_{i \to j}(X_j) = \frac{1}{Z_{i \to j}} \sum_{x_i} \pi_i(x_i) \pi_{i,j}(x_i, X_j) \prod_{s \in N(i) \setminus \{j\}} \delta_{s \to i}(x_i)$$

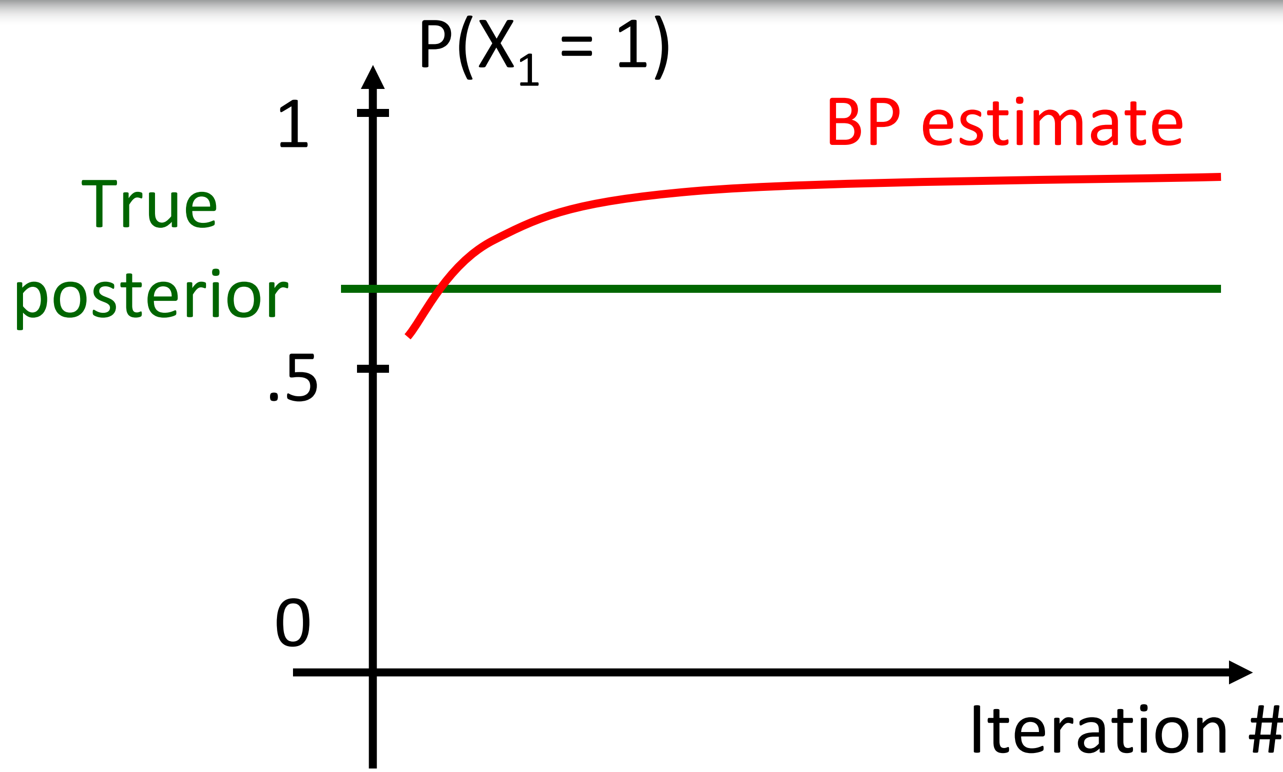
  - Does not affect outcome:

Normalization doesn't matter

$$\hat{P}(X_i) \propto \prod_{s \in N(i)} \delta_{s \to i}(X_i) \cdot (Z_{i \to j})$$

Behavior of BP





- Loopy BP multiplies same potentials multiple times

  ➔ BP often overconfident

When do we stop?

- Messages

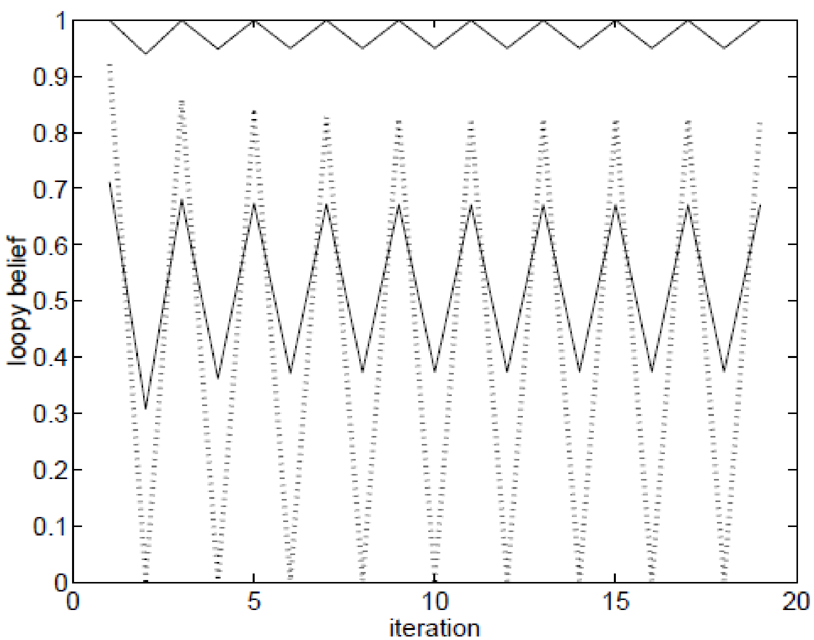
$$\delta_{i\to j}^{(t+1)}(X_j) = \frac{1}{Z_{i\to j}} \sum_{x_i} \pi_i(x_i)\pi_{i,j}(x_i, X_j) \prod_{s\in N(i)\setminus\{j\}} \delta_{s\to i}^{(t)}(x_i)$$

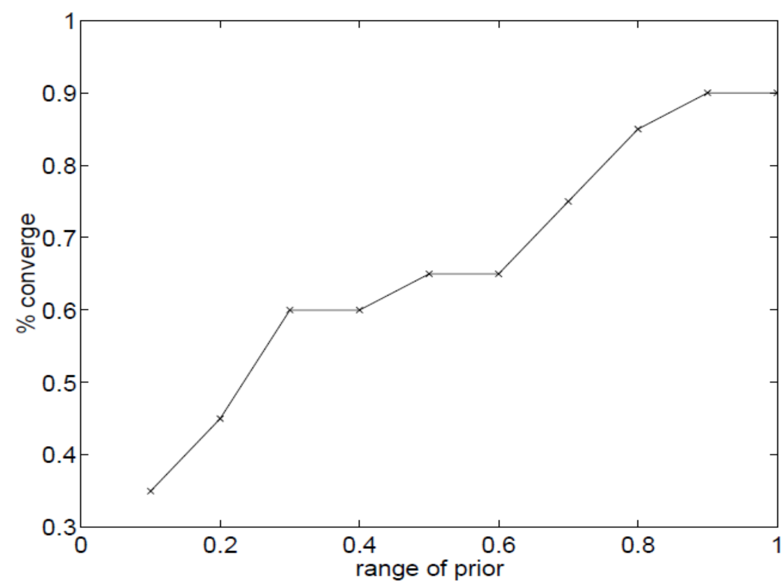
Stop if messages "don't change much"

$$\left| \delta_{i\to j}^{(t+1)} - \delta_{i\to j}^{(t)} \right| \leq \varepsilon \quad \forall i,j$$

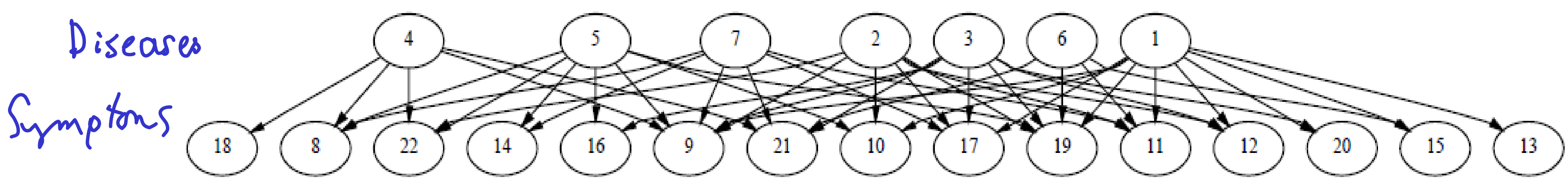
Does Loopy BP always converge?

- No! Can oscillate!
- Typically, oscillation the more severe the more "deterministic" the potentials





Graphs from K. Murphy UAI '99



What can we do to make BP converge?

Damping:

$$\hat{\delta}_{i\to j}^{(t+1)} = (1-\alpha)\,\delta_{i\to j}^{(t+1)} + \alpha\,\hat{\delta}_{i\to j}^{(t)}$$

What I said

"Correct" BP message

What I sent last time

If we need to dampen, answer will most likely be bad

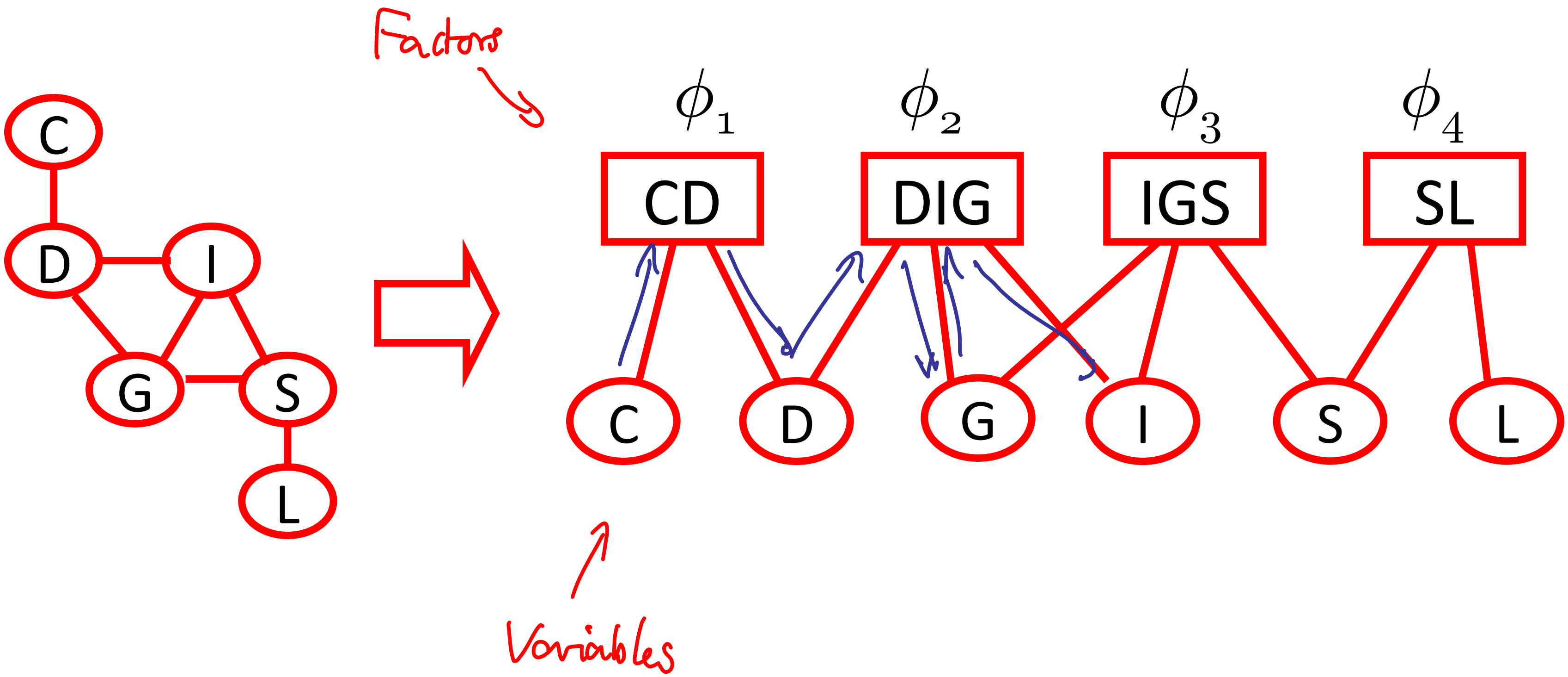
Can we prove convergence of BP?

- Yes, for special types of graphs (e.g., random graphs arising in coding)

- Sometimes can prove that message update "contracts"

What if we have non-pairwise MNs?

- Two approaches:
  - Convert to pairwise MN (possibly exponential blowup)
  - Perform BP on **factor graph**



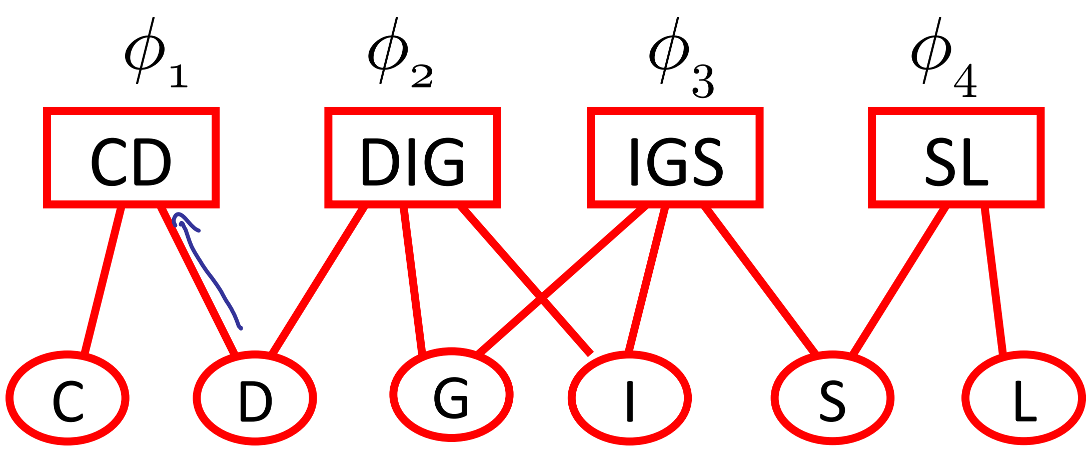
BP on factor graphs

- Messages from nodes to factors

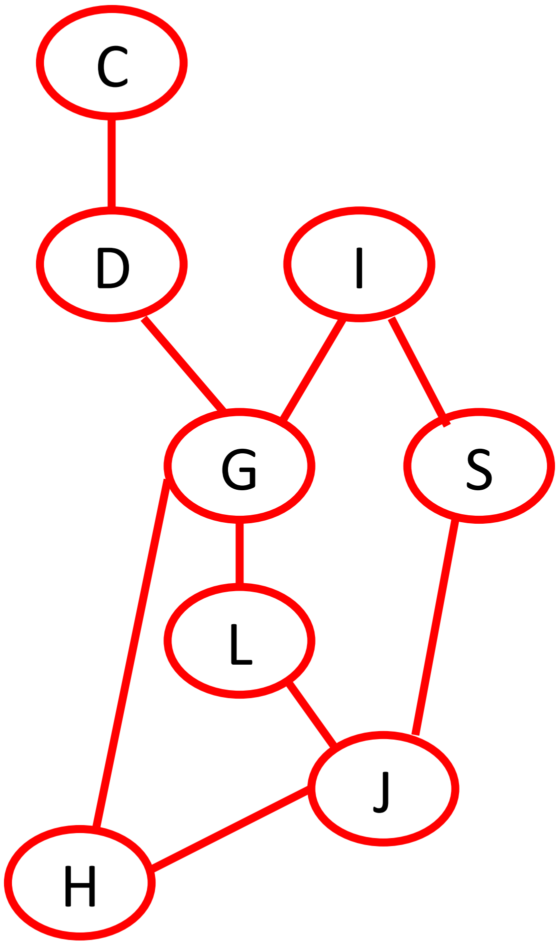
$$\delta_{x \to \phi}(x) = \frac{1}{Z} \prod_{\phi' \in N(x) \setminus \{\phi\}} \delta_{\phi' \to x}(x)$$

- Messages from factors to nodes

$$\delta_{\phi \to x}(x) = \frac{1}{Z} \sum_{x_\phi \sim x} \phi(x_\phi) \prod_{x' \in N(\phi) \setminus \{x\}} \delta_{x' \to \phi}(x)$$

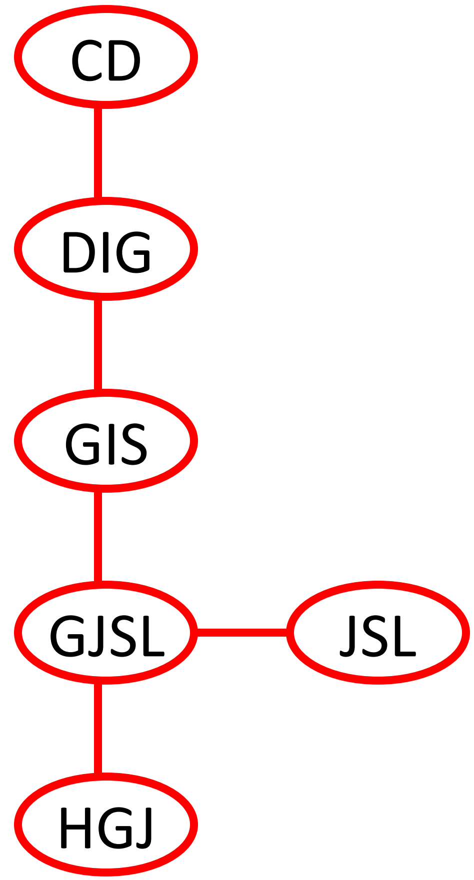


Loopy BP vs Junction tree



Some nice algos live here

vs.



Both BP and JT inference are “ends of a spectrum”

Other message passing algorithms

- Gaussian Belief propagation
- BP based on particle filters (see sampling)
- Expectation propagation
- …