CSC535: Probabilistic Graphical Models

Midterm Review

Prof. Jason Pacheco

Administrative Items

- Midterm out (obviously)
  - Due Friday @ 11:59pm
  - 4 questions (15 points) + 1 Extra Credit (2 points)
  - **You may provide handwritten responses (scanned PDF)**
  - Make sure handwriting is clear and easy-to-read

- No office hours Friday (I will be traveling)

Midterm

Problem 1 (4 points)

- Provide Bayes Net and Factor graphs for a model
- Give formula for sum-product messages in model
- Show dependence / independence

Extra Credit (2 points)

- Derive Poisson maximum likelihood estimate
- Derive MAP estimate with Gamma prior

Problem 2 (3 points)

- Two player game, best of 7 rounds wins
- Compute probability of winning conditioned on current score

Problem 3 (4 points)

- Show variable elimination for two different elimination orderings
- Bound on maximal clique size

Problem 4 (4 points)

- Show variable elimination for two different elimination orderings
- Bound on maximal clique size

Topics

- Probability and Statistics
- Probabilistic Graphical Models
- Message Passing Inference
- Parameter Learning

Topics

- Probability and Statistics
- Probabilistic Graphical Models
- Message Passing Inference
- Parameter Learning

Probability and Random Events

Fundamental Rules of Probability

- Conditional: $p(X \mid Y) = \frac{p(X,Y)}{p(Y)} = \frac{p(X,Y)}{\sum_x p(X=x,Y)}$
- Law of total probability: $p(Y) = \sum_x p(Y, X = x)$
- Probability chain rule: $p(X, Y) = p(Y)p(X \mid Y)$

Independence of RVs

- Two RVs X & Y are <u>independent</u> iff: $p(X \mid Y) = p(X)$
- Equivalently: $p(X, Y) = p(X)p(Y)$
- X & Y are <u>conditionally independent</u> given Z iff: $p(X \mid Y, Z) = p(X \mid Z)$
- Equivalently: $p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z)$

Tabular Method

Let X, Y be binary RVs with the joint probability table

For Binomial use K-by-K probability table.

		Y		
		y_1	y_2	
X	x_1	0.04	0.36	0.4
	x_2	0.30	0.30	0.6
	P(y)	0.34	0.66	P(x)

$P(x_1)$ $P(x_2)$ $P(y_1)$ $P(y_2)$

$P(y_1)=P(x_1,y_1)+P(x_2,y_1)$
$P(y_2)=P(x_1,y_2)+P(x_2,y_2)$
[i.e., sum down columns]

$P(x_1)=P(x_1,y_1)+P(x_1,y_2)$
$P(x_2)=P(x_2,y_1)+P(x_2,y_2)$
[i.e., sum across rows]

Tabular Method

We don't care about event $Y=y_2$

		Y	
		y_1	y_2
X	x_1	0.04	Censored!
	x_2	0.30	
		0.34	

$P(y_1)$

$P(x|y_1)=?$

Tabular Method

		$Y=y_1$		
X	x_1	0.04	$\Rightarrow$	$0.04 / 0.34$
	x_2	0.30	$\Rightarrow$	$0.30 / 0.34$

$P(x|y_1)$

0.34

$P(y_1)$

These sum to one:
A conditional probability distribution is still a probability distribution

Bayes' Rule

Posterior represents all uncertainty after observing data…

prior probability

likelihood function
for the parameters

$$p(\theta \mid y) = \frac{p(\theta)p(y \mid \theta)}{p(y)}$$

posterior probability

marginal likelihood

Bayesian Inference Example

About 29% of American adults have high blood pressure (BP). Home test has 30% false positive rate and no false negative error.

A recent home test states that you have high BP. Should you start medication?

An Assessment of the Accuracy of Home Blood Pressure Monitors When Used in Device Owners

Jennifer S. Ringrose,[1] Gina Polley,[1] Donna McLean,[2–4] Ann Thompson,[1,5] Fraulein Morales,[1] and Raj Padwal[1,4,6]

Bayesian Inference Example

About 29% of American adults have high blood pressure (BP). Home test has 30% false positive rate and no false negative error.

- Latent quantity of interest is hypertension: $\theta \in \{true, false\}$
- Measurement of hypertension: $y \in \{true, false\}$
- Prior: $p(\theta = true) = 0.29$
- Likelihood: $p(y = true \mid \theta = false) = 0.30$

$$p(y = true \mid \theta = true) = 1.00$$

Bayesian Inference Example

About 29% of American adults have high blood pressure (BP). Home test has 30% false positive rate and no false negative error.

Suppose we get a positive measurement, then posterior is:

$$p(\theta = true \mid y = true) = \frac{p(\theta = true)p(y = true \mid \theta = true)}{p(y = true)}$$

$$= \frac{0.29 * 1.00}{0.29 * 1.00 + 0.71 * 0.30} \approx 0.58$$

Bayesian Estimation

Task: *produce an estimate* $\hat{\theta}$ *of* θ *after observing data* y

Bayes estimators minimize expected **loss function**:

$$\mathbb{E}[L(\theta, \hat{\theta}) \mid y] = \int p(\theta \mid y) L(\theta, \hat{\theta})\, d\theta$$

Example: Minimum mean squared error (MMSE):

$$\hat{\theta}^{\mathrm{MMSE}} = \arg\min \mathbb{E}[(\hat{\theta} - \theta)^2 \mid y] = E[\theta \mid y]$$

Posterior mean always minimizes squared error.

Topics

- Probability and Statistics
- Probabilistic Graphical Models
- Message Passing Inference
- Parameter Learning

Directed Graphical Models

- Distribution factorized as product of conditionals via chain rule

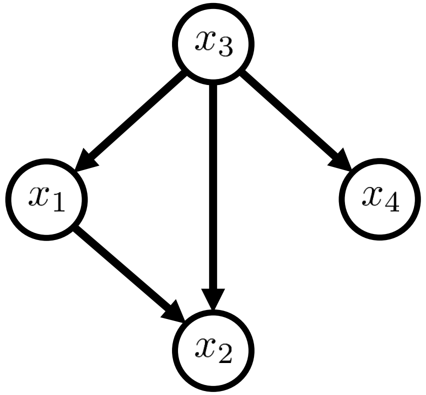
$$p(x_1, x_2, x_3, x_4) = p(x_3)p(x_1 \mid x_3)p(x_4 \mid x_1, x_3)p(x_2 \mid x_1, x_3, x_4)$$

- Choose ordering where terms simplify due to conditional independence

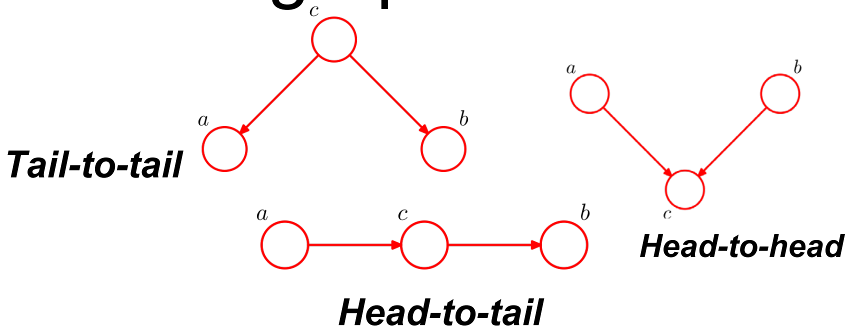
Eg. Suppose $x_4 \perp x_1 \mid x_3$ and $x_2 \perp x_4 \mid x_1$ then:

$$p(x) = p(x_3)p(x_1 \mid x_3)p(x_4 \mid x_3)p(x_2 \mid x_1, x_3)$$

- Directed graph encodes factorized distribution via conditional independence properties

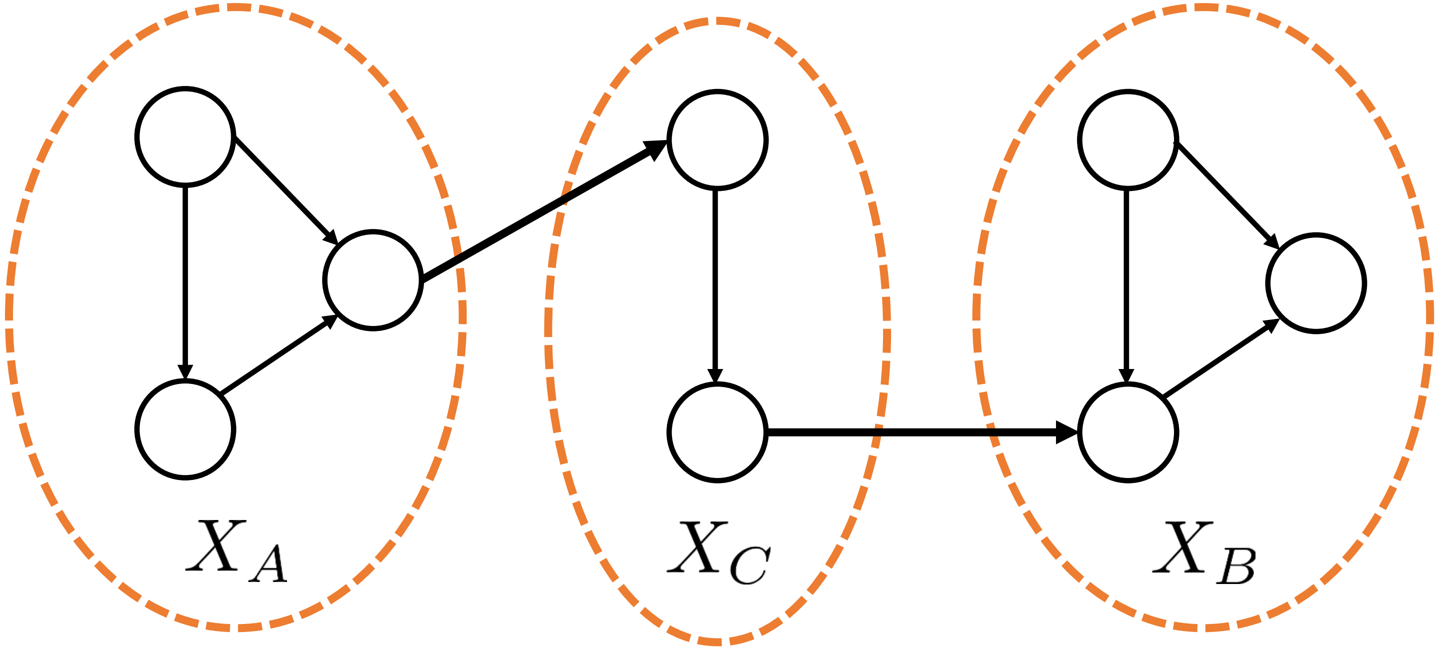


- Test independence using canonical subgraphs:
- Straightforward simulation via **ancestral sampling**



Bayes Ball Algorithm

To test if $X_A \perp X_B \mid X_C$ roll ball from *every node in* X_A …



If *any* ball reaches *any node* in X_B then

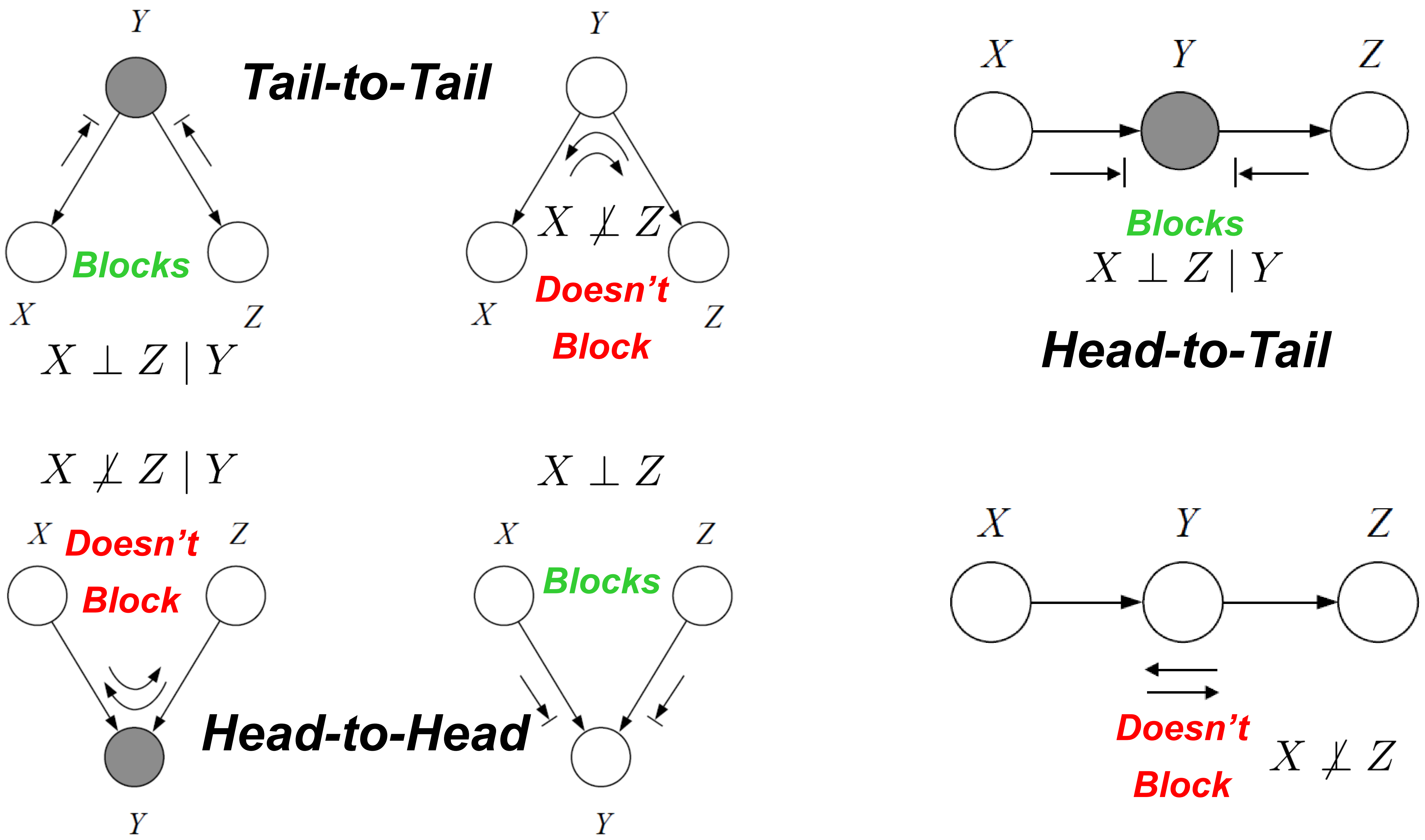
$$X_A \not\perp X_B \mid X_C$$

Otherwise:

$$X_A \perp X_B \mid X_C$$

Tests for property of ***directed separation*** **(d-separation)**: if X_C separates / blocks X_A from X_B then $X_A \perp X_B \mid X_C$

Bayes Ball Algorithm

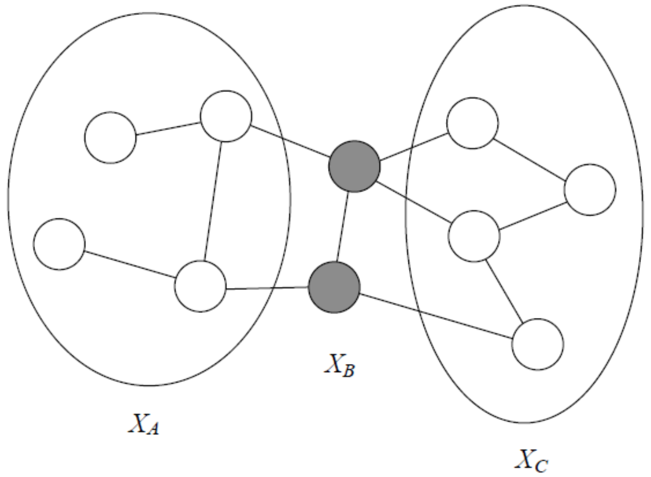


Undirected Graphical Models

- Joint factorization as nonnegative factors (potentials) over subsets:

$$p(x) \propto \prod_{f \in \mathcal{F}} \psi_f(x_f)$$

- Easier to specify models compared to Bayes nets since:
  - Factors do not need to be normalized conditional probabilities
  - May specify up to unknown normalization constant
- Easier to verify Markov independence via *separating sets*
- Factorization ambiguous in MRFs, but explicit in factor graphs (factor graphs generally preferred)
- Simulation is not easy in general. Can't do ancestral sampling.

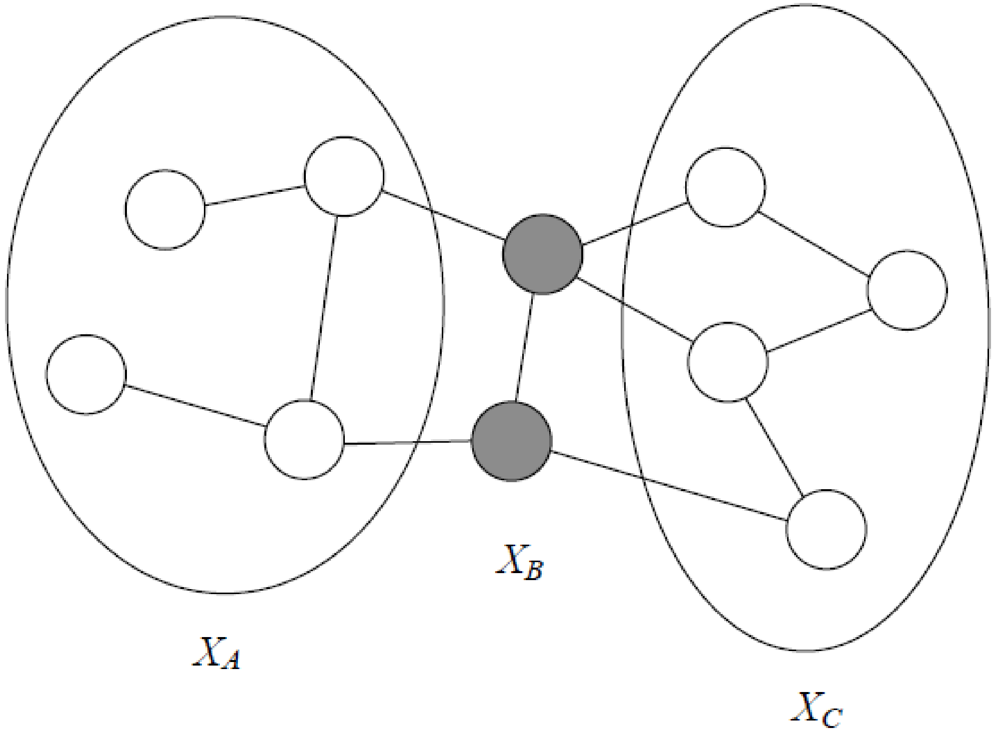


Conditional Independence (Undirected)

We say x_A and x_C are conditionally *independent* $x_A \perp\!\!\!\perp x_C \mid x_B$ *given variables* x_B *iff,*

$$p(x_A, x_C \mid x_B) = p(x_A \mid x_B)p(x_C \mid x_B)$$

Def. We say $p(x)$ is *globally Markov* w.r.t. $\mathcal{G}$ if $x_A \perp\!\!\!\perp x_C \mid x_B$ in any separating set of $\mathcal{G}$.

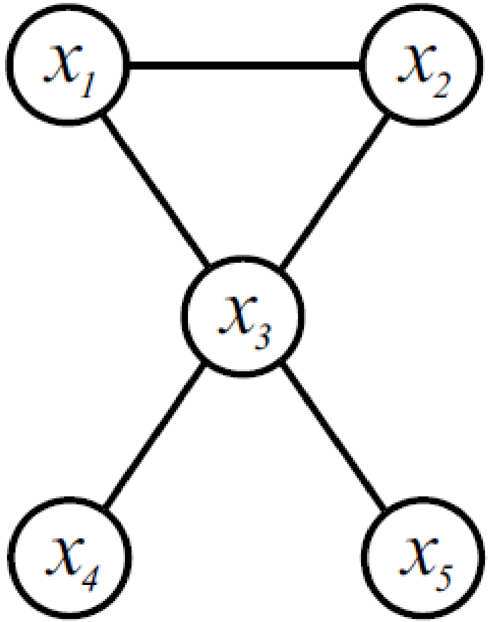


Conditional independence in undirected graphical models is defined by separating sets

[Source: Michael I. Jordan]

Markov Random Fields (MRFs)

A factor $\psi_c(x_c)$ corresponds to a clique $c \in \mathcal{C}$ (fully connected subgraph) in the MRF



An MRF does not imply a unique factorization, for example either of the following are "*valid*":

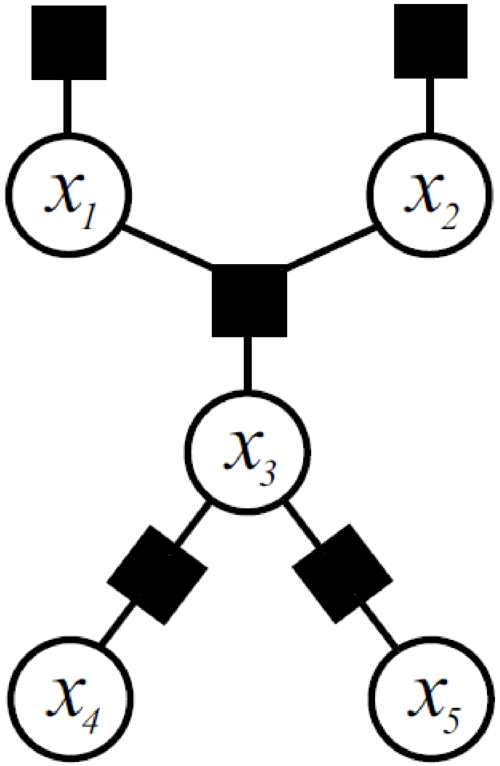
$$\psi(x_1, x_2, x_3)\psi(x_3, x_4)\psi(x_3, x_5)$$

$$\psi(x_1, x_2)\psi(x_2, x_3)\psi(x_1, x_3)\psi(x_3, x_4)\psi(x_3, x_5)$$

A factorization is *valid* if it satisfies the *Global Markov property*, defined by conditional independencies

Factor Graphs

Factor graphs make factorization explicit…



Factor node for each product term in the joint factorization:

$$p(x) \propto \prod_{f \in \mathcal{F}} \psi_f(x_f)$$

where $x_f = \{x_i : i \in f\}$ the set of variables in factor *f*. For example:

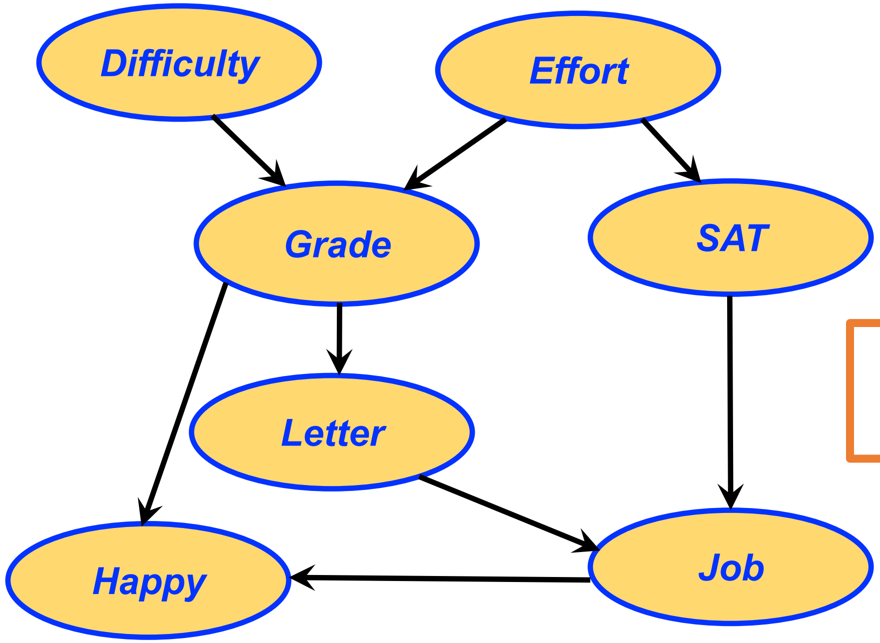
$$\psi(x_1)\psi(x_2)\psi(x_1, x_2, x_3)\psi(x_3, x_4)\psi(x_3, x_5)$$

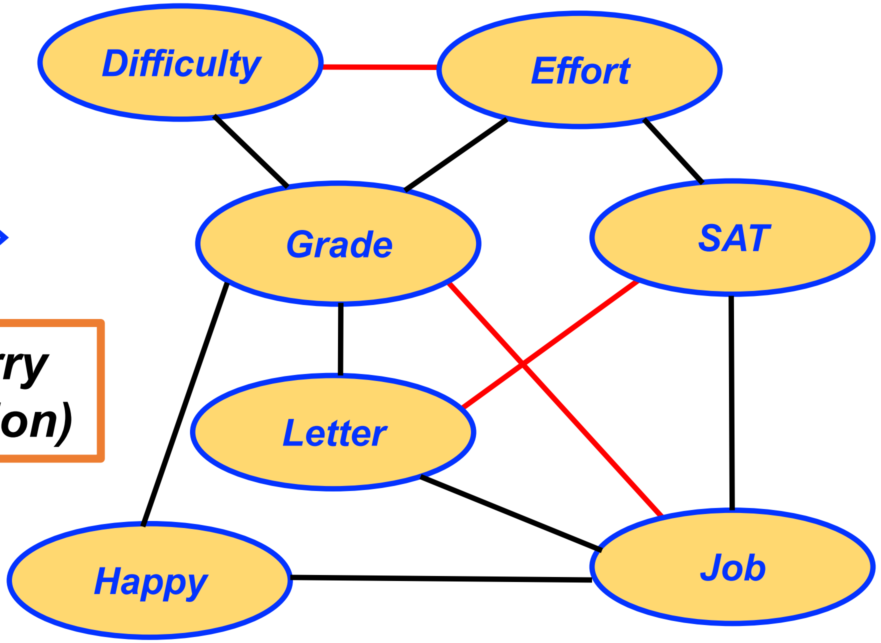
Factor nodes correspond to MRF cliques

Topics

- Probability and Statistics
- Probabilistic Graphical Models
- Message Passing Inference
- Parameter Learning

Bayes Net → MRF





Added edges *marry* parents (*moralization*)

$$P(\cdot) = P(D)P(E)P(G \mid D, E)P(S \mid E)P(L \mid G)P(J \mid S, L)P(H \mid J, G)$$

Drop local normalization

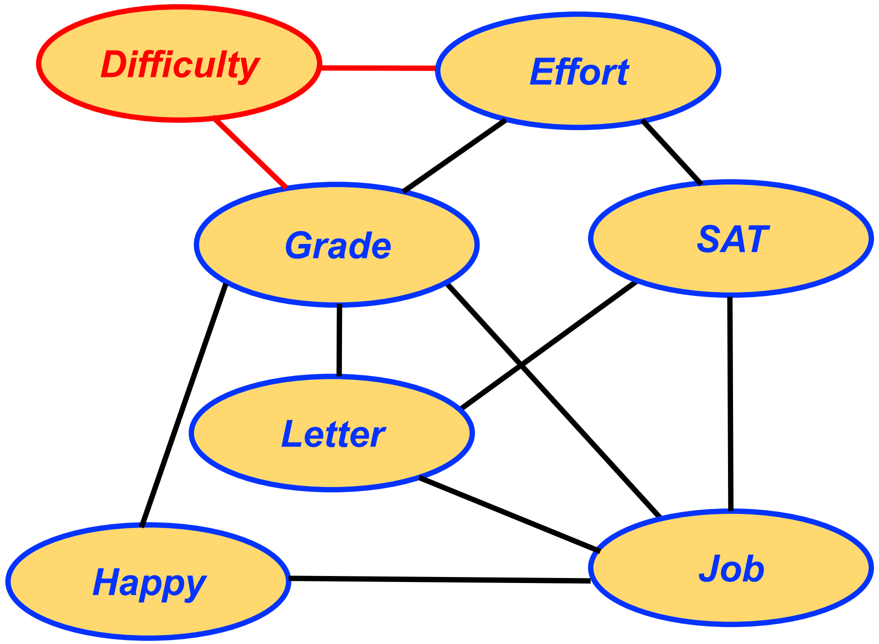
$$P(\cdot) \propto \psi(D)\psi(E)\psi(G, D, E)\psi(S, E)\psi(L, G)\psi(J, S, L)\psi(H, J, G)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$\mathcal{O}(K^3)$



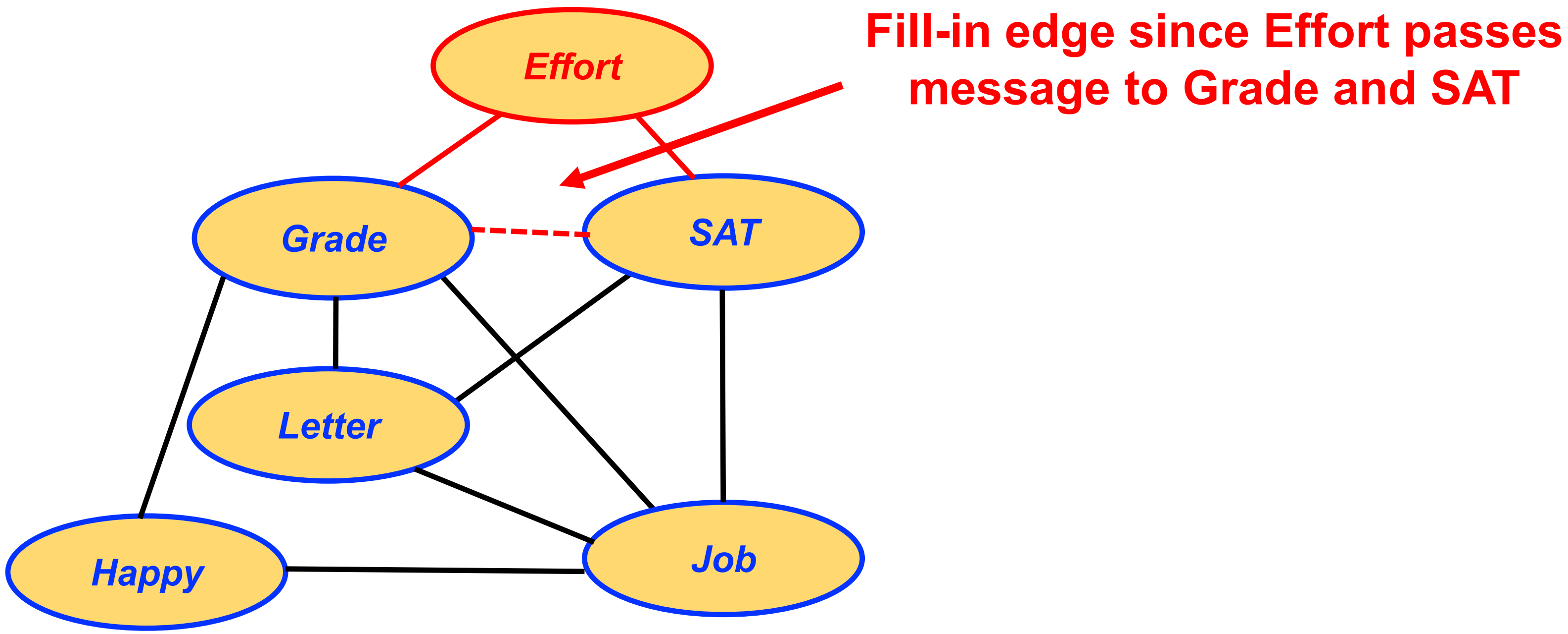
$$\phi(D, E, G) = \mathcal{O}(K^3)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$\mathcal{O}(K^3)$



$$\phi(E, G, S) = \mathcal{O}(K^3)$$

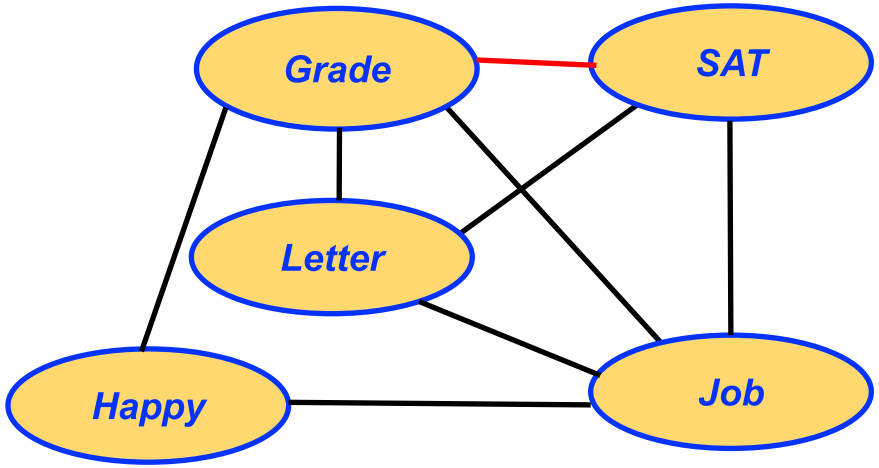
Variable Elimination

Elimination order D, E, H, G, S, L

Fill-in Edge

Worst-case Complexity:

$\mathcal{O}(K^3)$



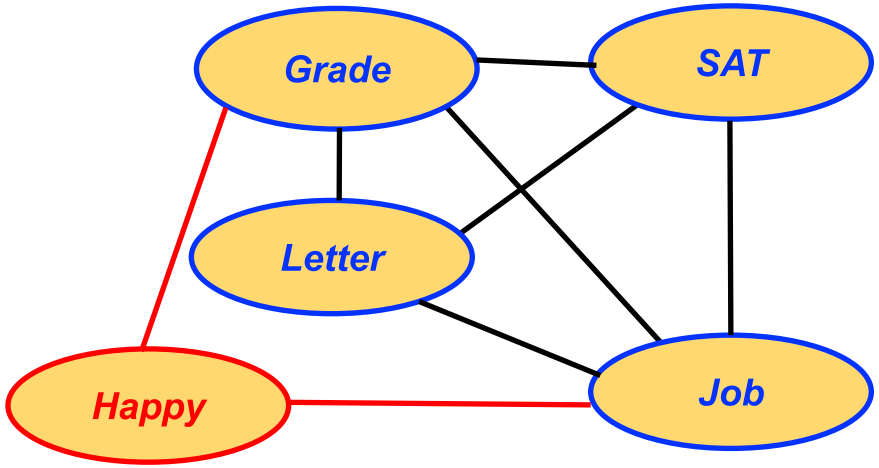
$$\phi(E, G, S) = \mathcal{O}(K^3)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$$\mathcal{O}(K^3)$$



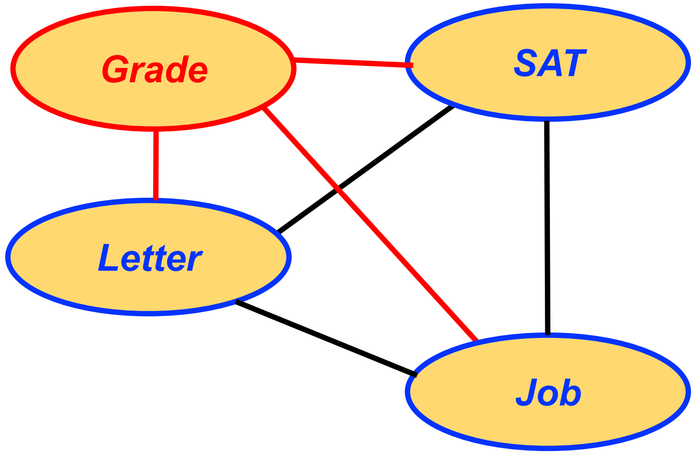
$$\phi(H, G, J) = \mathcal{O}(K^3)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$\mathcal{O}(K^4)$



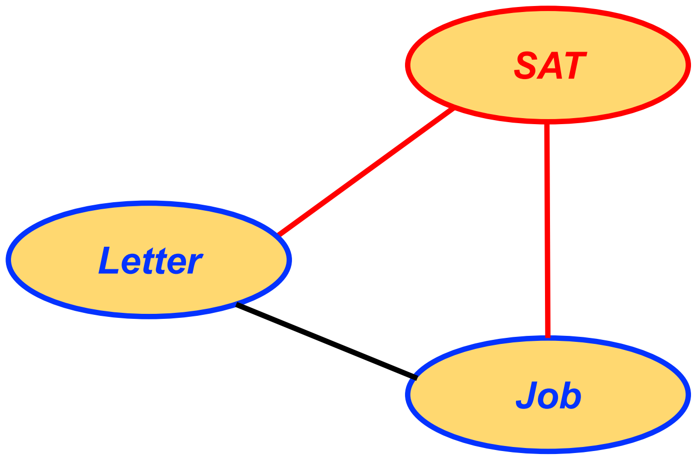
$$\phi(G, S, L, J) = \mathcal{O}(K^4)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$$\mathcal{O}(K^4)$$



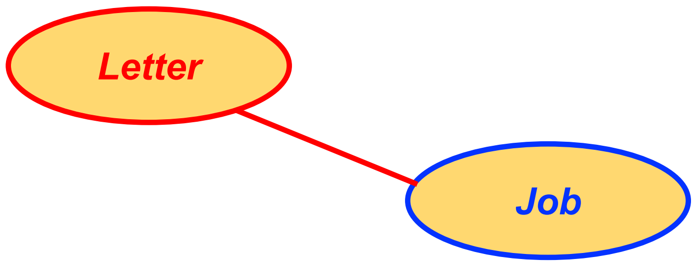
$$\phi(S, L, J) = \mathcal{O}(K^3)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$$\mathcal{O}(K^4)$$



$$\phi(L, J) = \mathcal{O}(K^2)$$

Variable Elimination

Elimination order D, E, H, G, S, L

Worst-case Complexity:

$$\mathcal{O}(K^4)$$

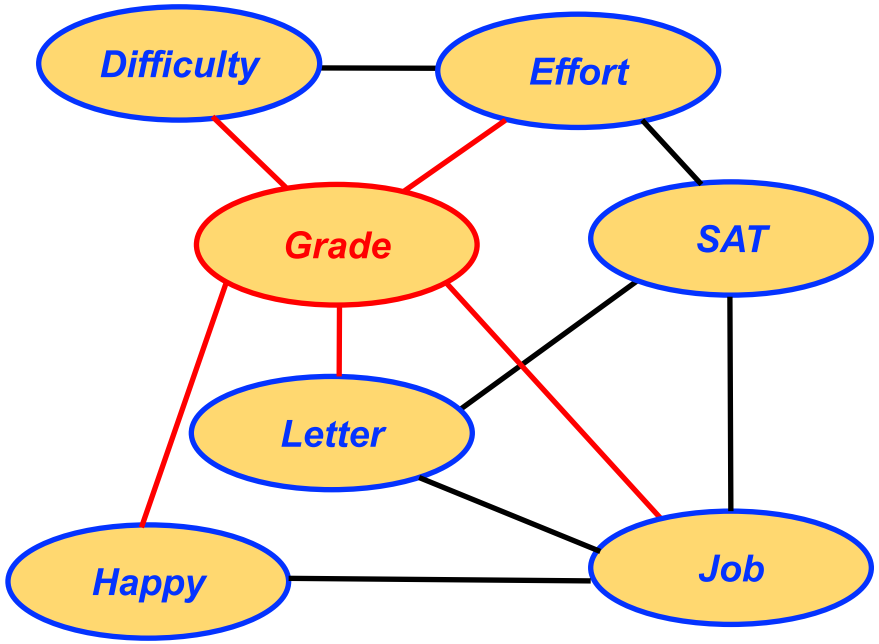
What if we choose a different elimination order?

$$\phi(L, J) = \mathcal{O}(K^2)$$

Computational Complexity

Eliminate G first…

Add fill-in edges to connect all neighbors



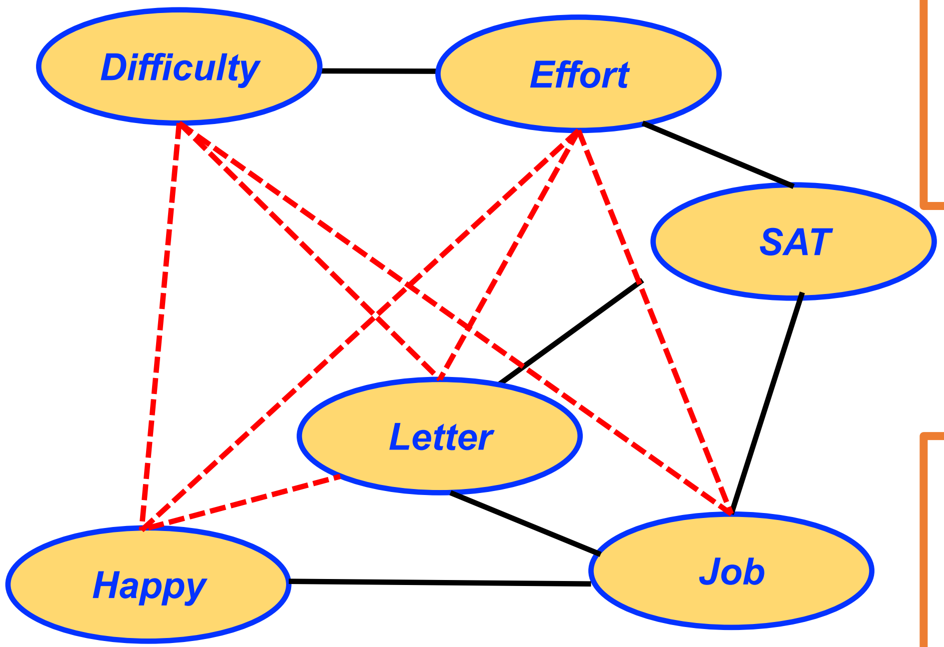
$$\phi(G, D, E, L, H, J) = \mathcal{O}(K^6)$$

Complexity depends on elimination order…

For N variables worst case is: $\mathcal{O}(K^N)$

Computational Complexity

Eliminate G first…



$$\phi(G, D, E, L, H, J) = \mathcal{O}(K^6)$$

Worst-case Complexity:

$\mathcal{O}(K^6)$

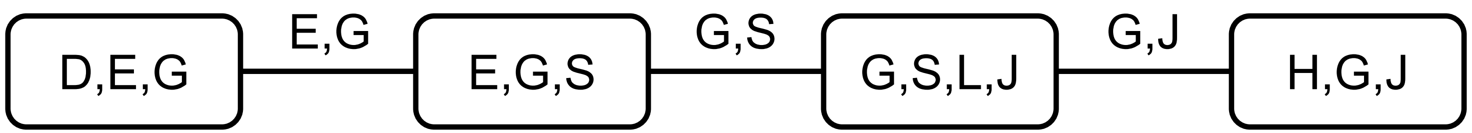
Complexity depends on elimination order…

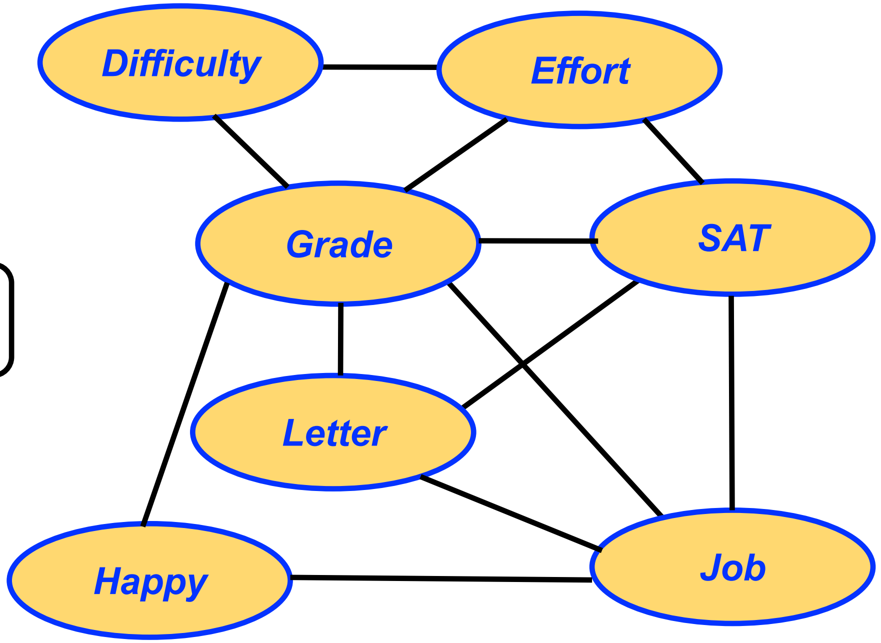
For N variables worst case is:

$\mathcal{O}(K^N)$

Computational Complexity

Clique Tree





Elimination order $\prec$ induces graph with maximal cliques $\mathcal{C}(\prec)$ and *width:*

$$w(\prec) = \max_{c \in \mathcal{C}(\prec)} |c| - 1$$

- Complexity of variable elimination is $\mathcal{O}(K^{w(\prec)+1})$
- Lowest complexity given by the *treewidth:*

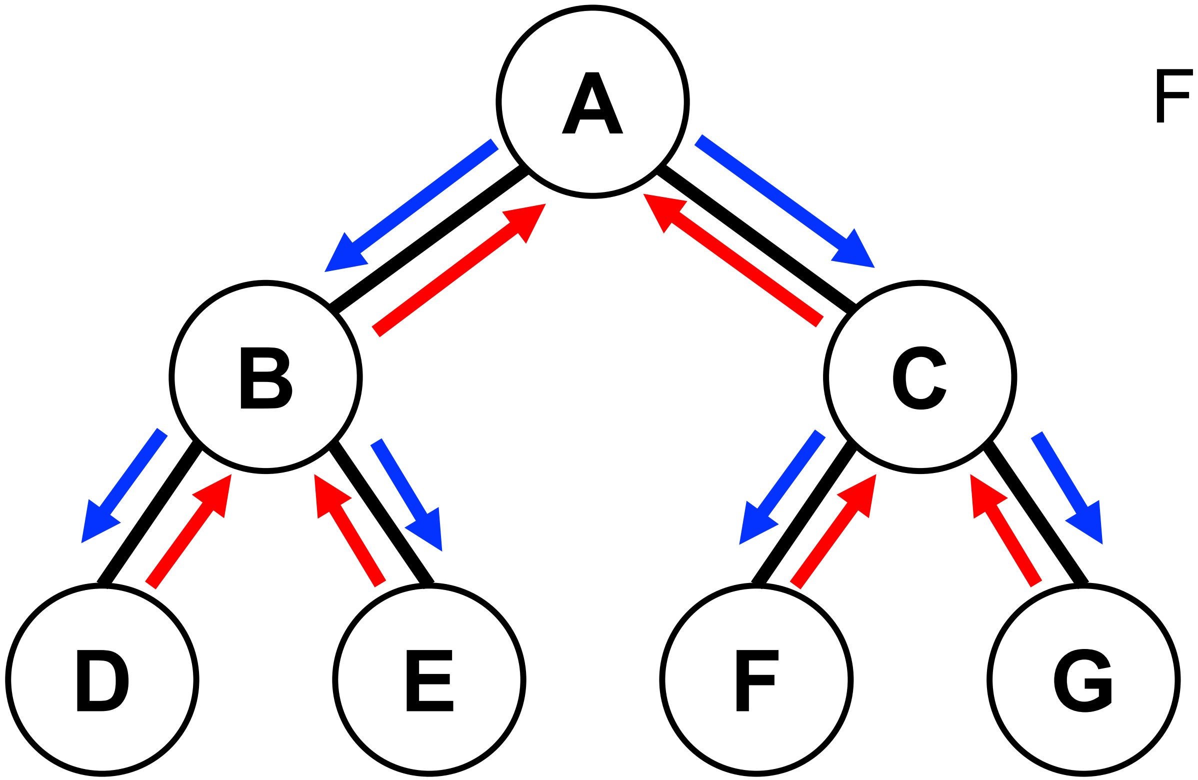
$$w^* = \min_{\prec} \max_{c \in \mathcal{C}(\prec)} |c| - 1$$

It is NP-hard to compute treewidth, and therefore an optimal elimination order (of course…)

Variable Elimination Summary

- Variable elimination allows computation of marginals / conditionals
- Algorithm is valid for **any graphical model**
- Suffices to show variable elimination for MRFs, since Bayes nets → MRFs by *moralization*
- Worst-case complexity is dependent on elimination order, and is **exponential** in number of variables
- Optimal ordering = treewidth, is NP-hard to compute

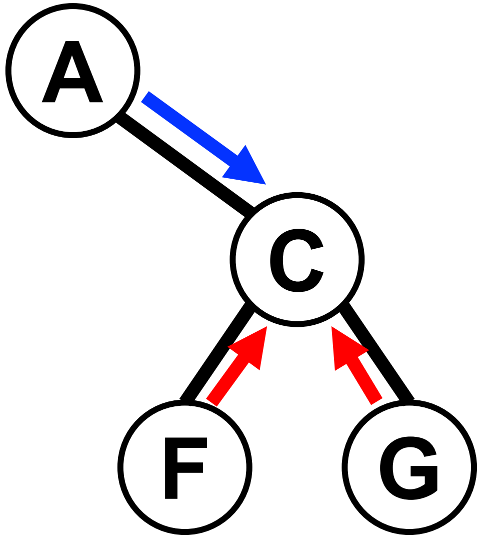
Sum-Product Belief Propagation



Pass messages from leaves-to-root, then root-to-leaves

Forward-Backward extends to *any tree-structured pairwise MRF*

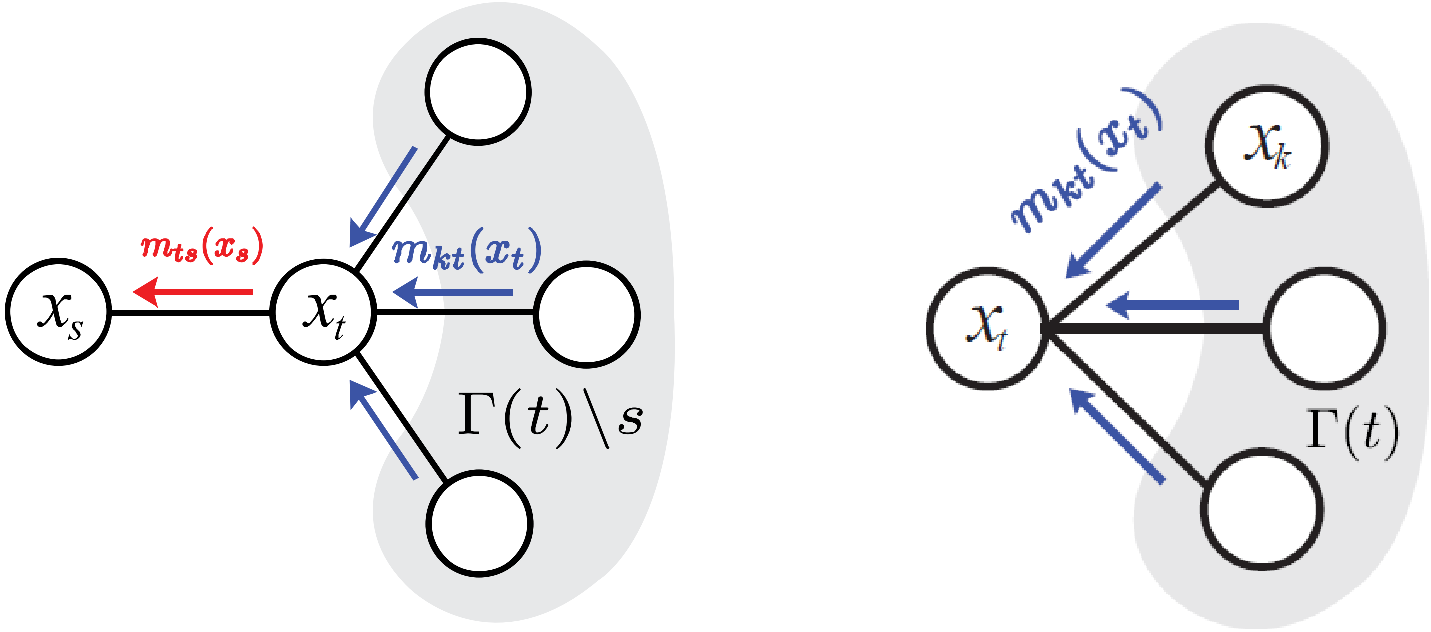
Marginal given by *incoming* messages (e.g. node C):



$$p(C) \propto \psi(C)m_A(C)m_F(C)m_G(C)$$

Pairwise MRF Sum-Product Belief Propagation

Message updates depend only on Markov blanket…

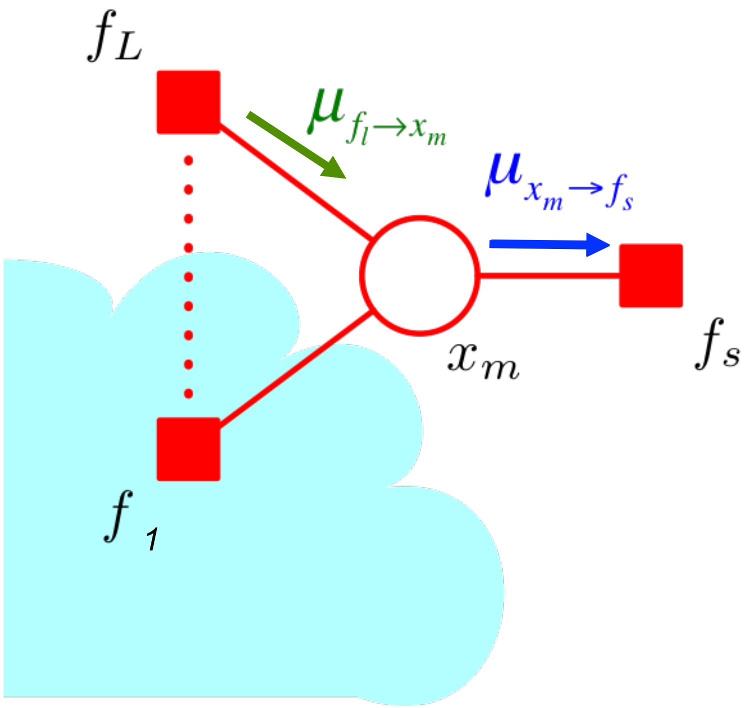


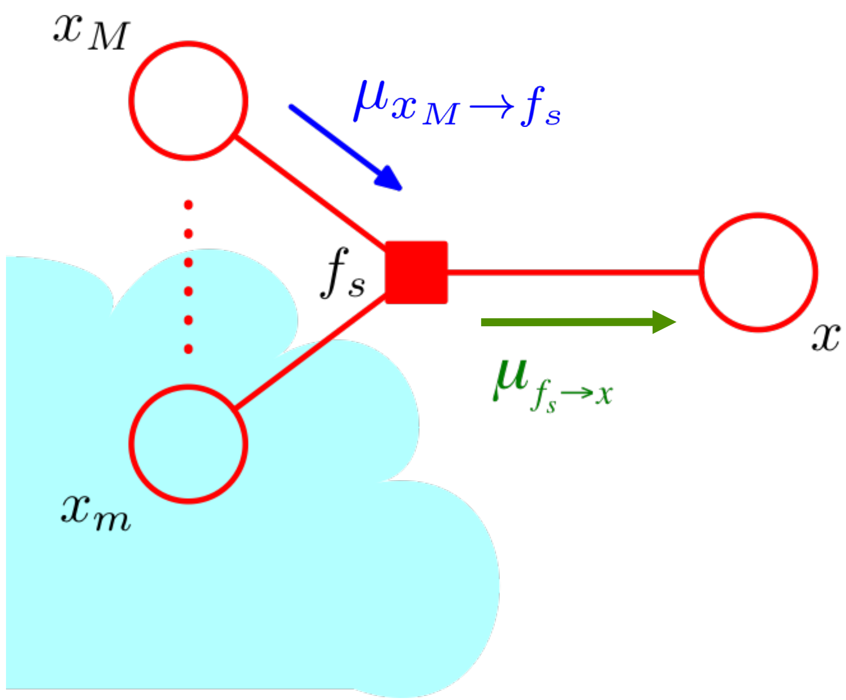
Message $m_{ts}(x_s) = \sum_{x_t} \psi_{st}(x_s, x_t)\psi_t(x_t) \prod_{k\in\Gamma(t)\backslash s} m_{kt}(x_t)$

Marginal $p(x_t) \propto \psi_t(x_t) \prod_{k\in\Gamma(t)} m_{kt}(x_t)$

Messages involve a **sum** over **products**, hence the name "sum-product algorithm"

Factor Graph Sum-Product Belief Propagation





Variable node x_m gathers messages, $\mu_{f_l \to x_m}$, and sends

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \ni f_l \in n(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

Factor f_s gathers messages $\mu_{x_m \to f_s}(x_m)$, and sends

$$\mu_{f_s \to x}(x) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_M} f_s(x, x_1, x_2, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

Marginal is product of incoming factor-to-variable messages:

$$p(x_m) \propto \prod_{f_l \in ne(x_m)} \mu_{f_l \to x_m}(x_m)$$

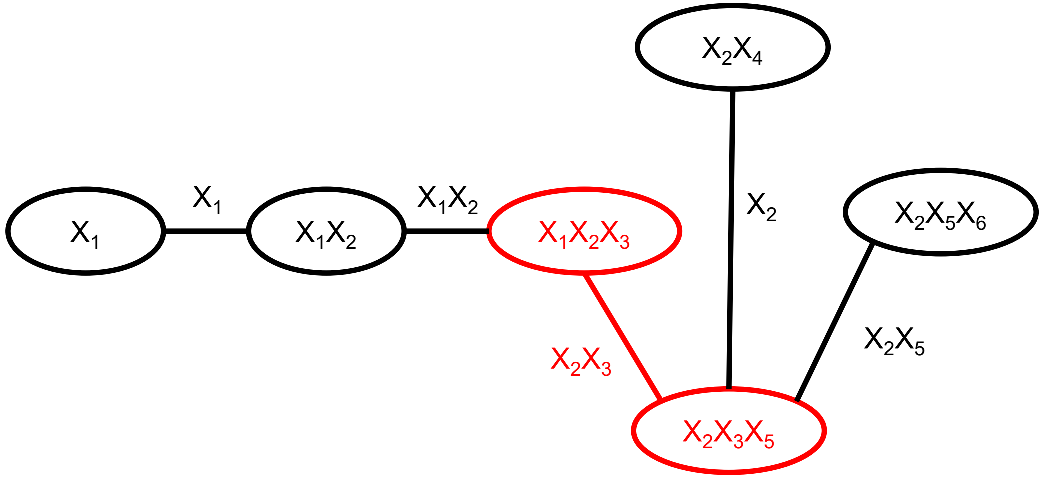
Marginal Inference Algorithms

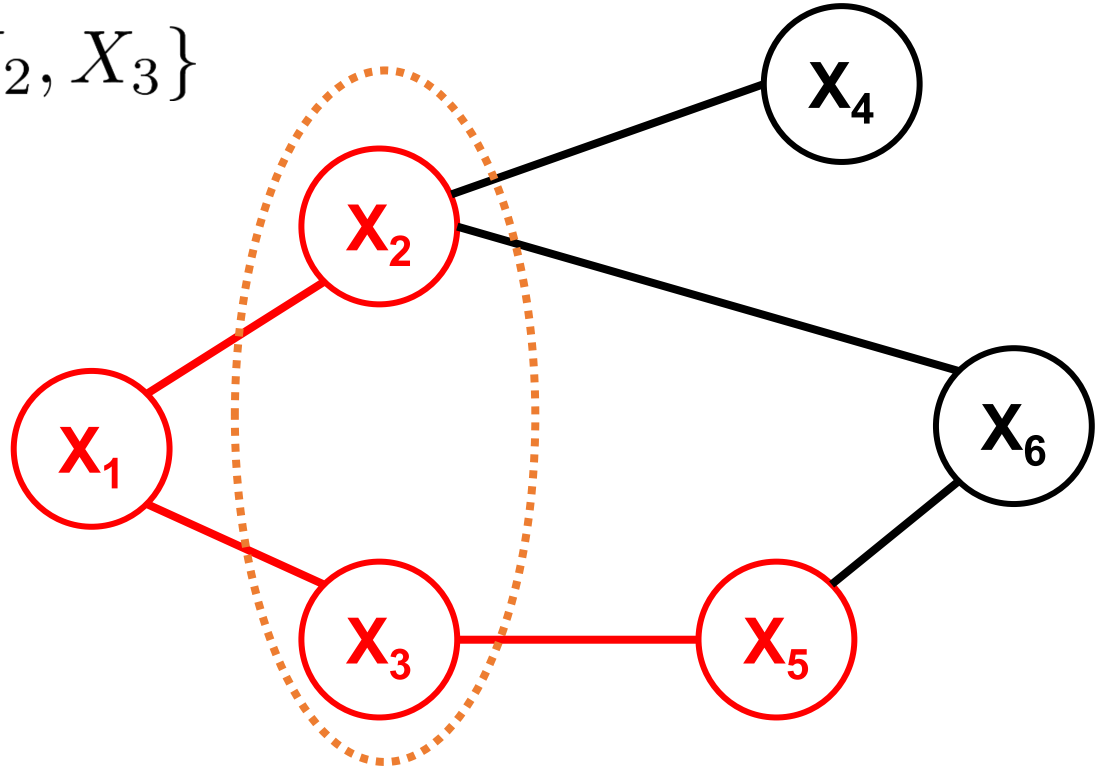
	One Marginal	*All Marginals*
Tree	Elimination applied to leaves of tree	Belief Propagation (BP) or sum-product algorithm
Graph	Variable Elimination	Junction Tree Algorithm BP on a junction tree (special clique tree)

Junction Tree

Clique tree edges are separator sets in original MRF…so clique tree encodes conditional independencies

$$X_1 \perp X_5 \mid \{X_2, X_3\}$$



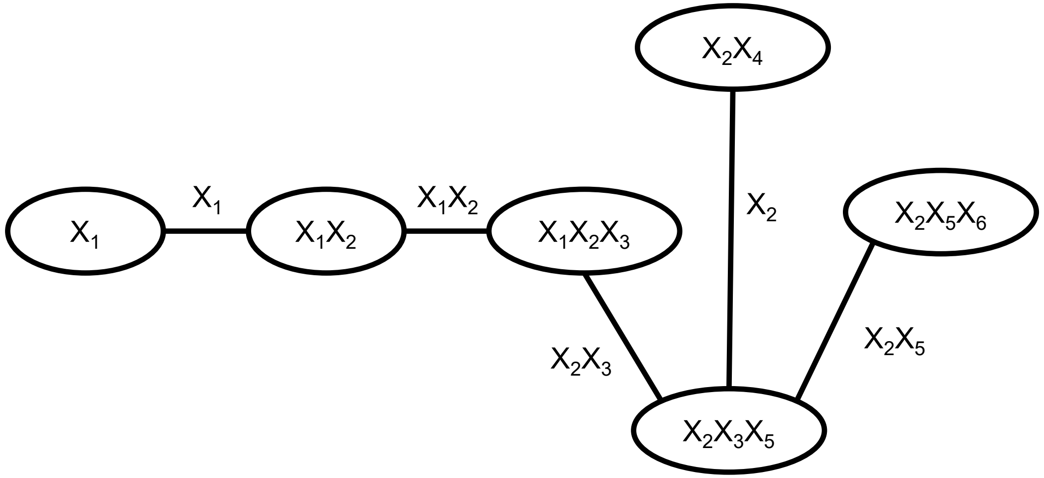


Theorem A clique tree resulting from variable elimination **satisfies the running intersection property** and is thus **a junction tree**

Junction Tree

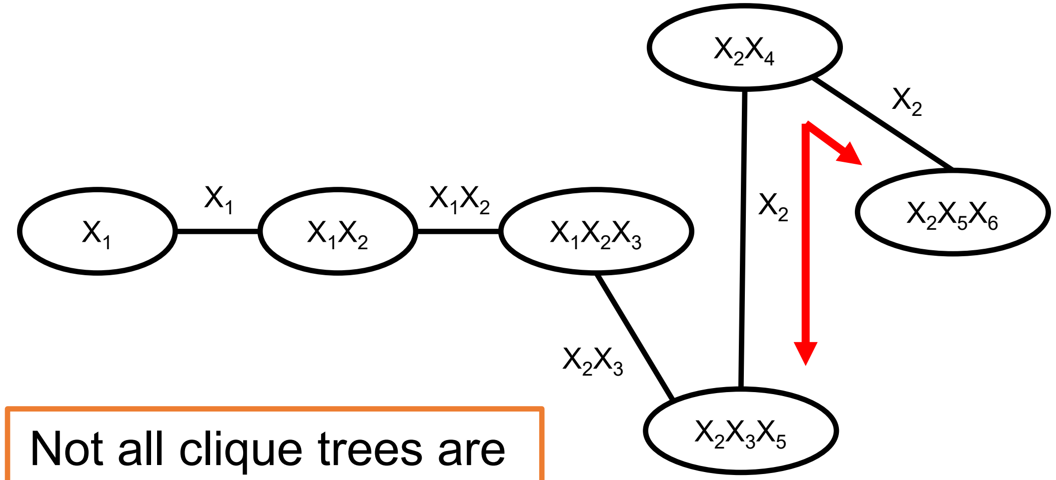
Definition (Running intersection) For any pair of clique nodes V,W all cliques on the *unique path* between V and W contain shared variables

Junction Tree



Not A Junction Tree

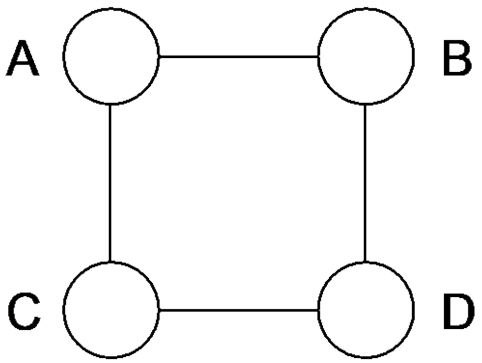
$$\{X_2, X_3, X_5\} \cap \{X_2, X_5, X_6\} = \{X_2, X_5\}$$

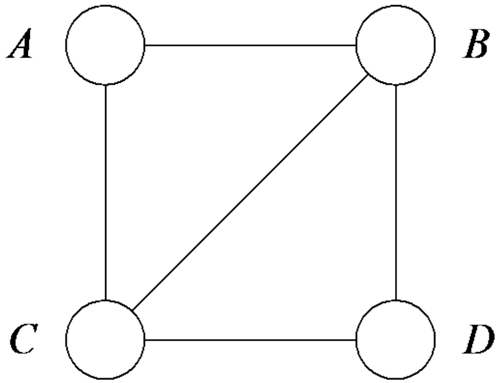


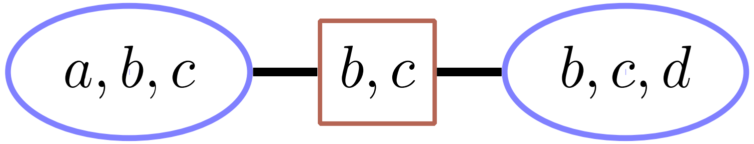
Not all clique trees are junction trees

Theorem A clique tree resulting from variable elimination **satisfies the running intersection property** and is thus **a junction tree**

Junction Trees and Triangulation







- A *chord* is an edge connecting two non-adjacent nodes in some *cycle*
- A cycle is *chordless* if it contains no chords
- A graph is *triangulated (chordal)* if it contains no chordless cycles of length 4 or more

Theorem: The maximal cliques of a graph have a corresponding junction tree *if and only if* that undirected graph is triangulated

Lemma: *For a non-complete triangulated graph with at least 3 nodes, there is a decomposition of the nodes into disjoint sets A, B, S such that S separates A from B, and S is complete.*

- ➢ Key induction argument in constructing junction tree from triangulation
- ➢ Implies existence of *elimination ordering which introduces no new edges*

Induced Graph

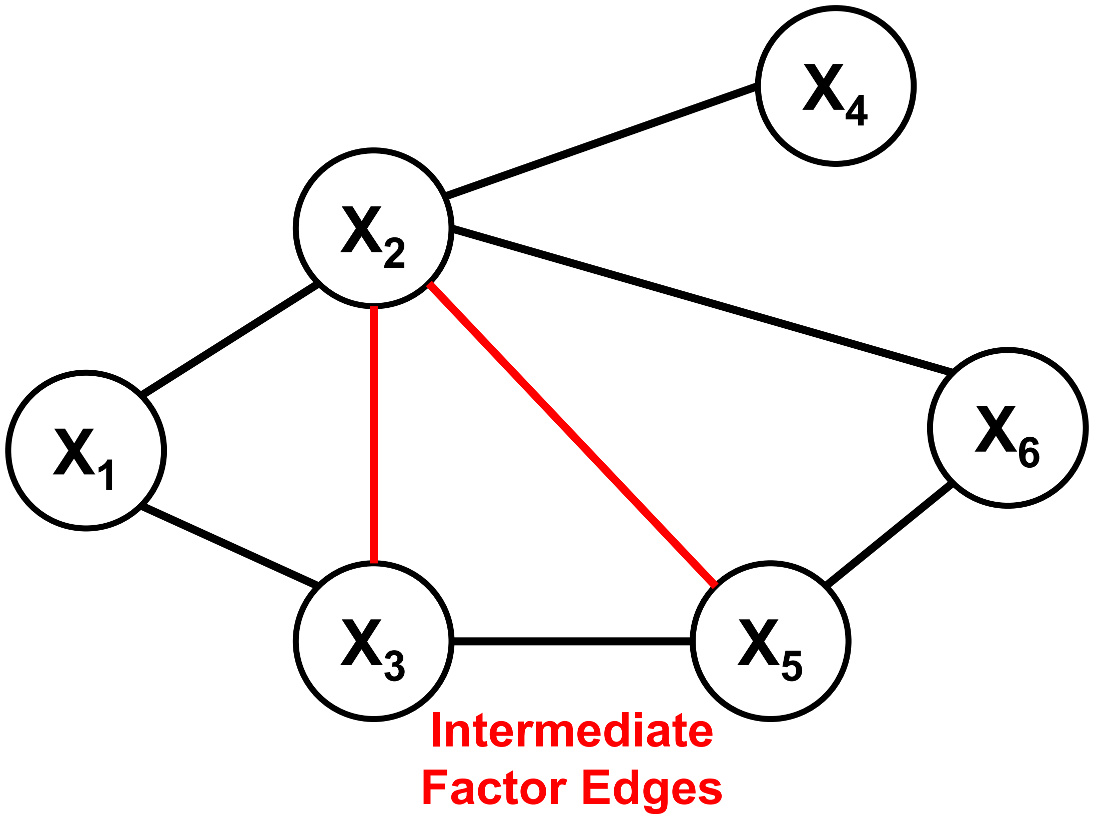
Recall the **induced graph** is the union over intermediate graphs from running variable elimination

The induced graph **is chordal** thus:

- Maximal cliques of the induced graph form a junction tree
- It admits an elimination ordering that introduces *no new edges*

Logic of junction tree algorithm:

1. Triangulate the graph
   a. Implies a junction tree
   b. Induces an elimination order
2. Run sum-product BP on junction tree to compute **all clique marginals**



Loopy Belief Propagation (sum-product)

Initialize Messages

Constant: $m^0_{st}(x_t) = \text{const.}$

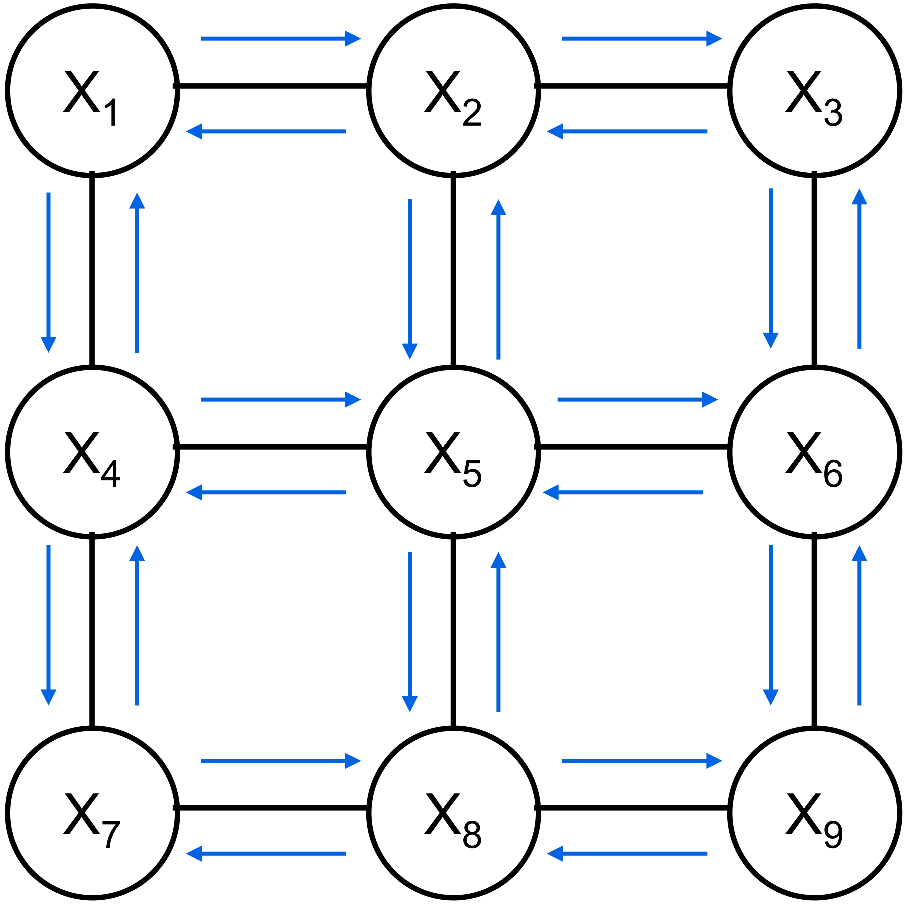
Random: $m^0_{st}(x_t) \sim U([0,1])$

Parallel (Synchronous) Updates

At iteration i update *all messages in parallel* using current messages m^{i-1} from previous iteration:

$$m^i_{st}(x_t) = \sum_{x_s} \psi_{st}(x_s, x_t) \prod_{k \in \Gamma(s) \setminus t} m^{i-1}_{ks}(x_s)$$

- Store, both, the *previous* messages (from iteration i-1) and *current* messages (from iteration i)
- Many convergence results assume parallel updates



Loopy Belief Propagation (sum-product)

Initialize Messages

Constant: $m^0_{st}(x_t) = \text{const.}$

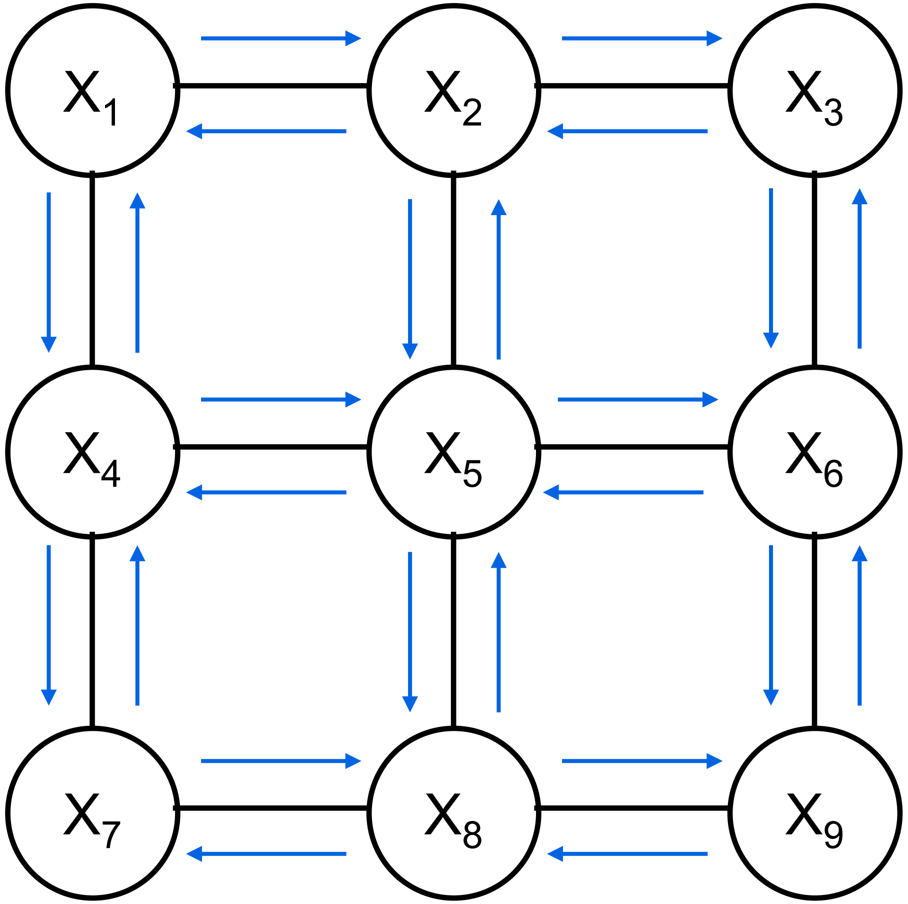
Random: $m^0_{st}(x_t) \sim U([0,1])$

Asynchronous (Sequential) Updates

Choose an ordering of nodes and update using the latest available messages:

$$m_{st}(x_t) = \sum_{x_s} \psi_{st}(x_s, x_t) \prod_{k \in \Gamma(s) \setminus t} m_{ks}(x_s)$$

- Simplifies updates since only need to keep track of one copy of messages
- Makes parallel processing trickier



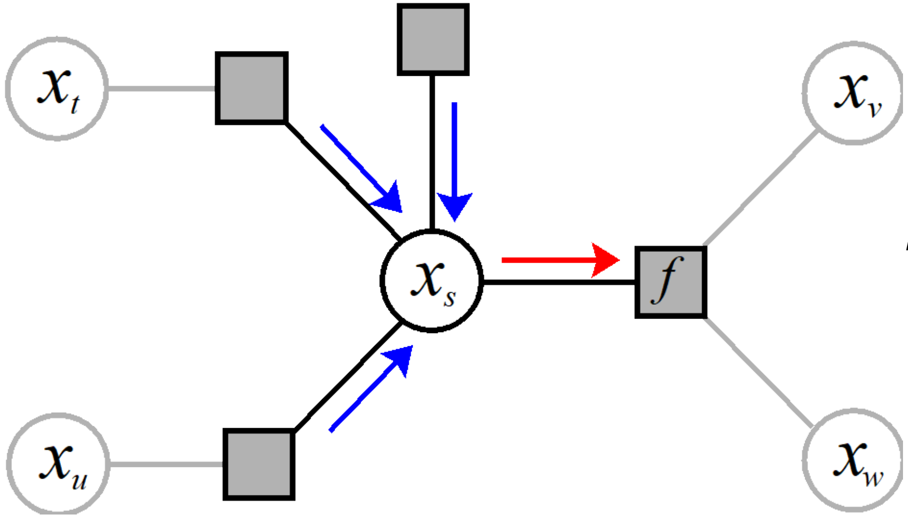
Pseudocode from Murphy's Textbook

Algorithm 22.1: Loopy belief propagation for a pairwise MRF

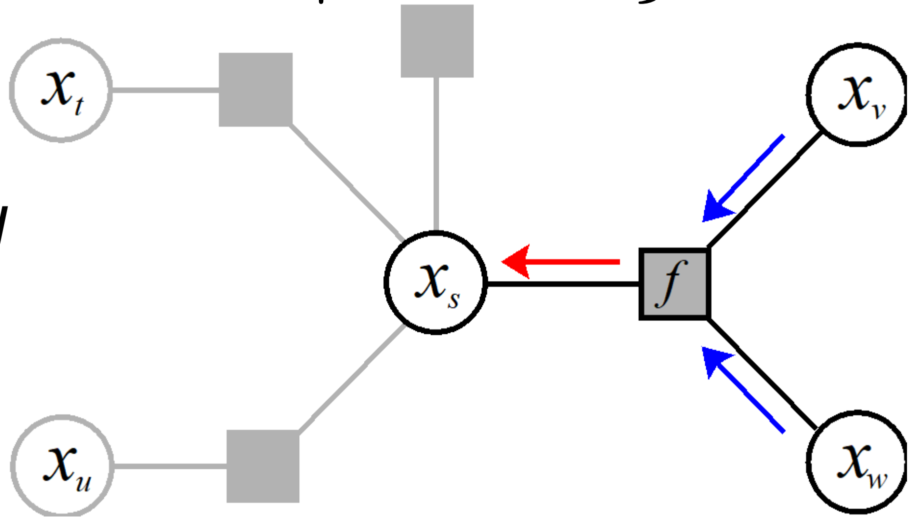
Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
Initialize messages $m_{s\to t}(x_t) = 1$ for all edges $s - t$;
Initialize beliefs $\mathrm{bel}_s(x_s) = 1$ for all nodes s;
repeat
    Send message on each edge
    $m_{s\to t}(x_t) = \sum_{x_s} \left(\psi_s(x_s)\psi_{st}(x_s, x_t) \prod_{u \in \mathrm{nbr}_s \setminus t} m_{u\to s}(x_s) \right)$;
    Update belief of each node $\mathrm{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \mathrm{nbr}_s} m_{t\to s}(x_s)$;
until *beliefs don't change significantly*;
Return marginal beliefs $\mathrm{bel}_s(x_s)$;

Loopy BP on Factor Graphs

Set of *neighbors* of node *s*: $\Gamma(s) = \{f \in \mathcal{F} \mid s \in f\}$



Loopy BP:
Message updates can be iteratively computed on graphs with cycles.
But marginals not guaranteed correct!



$$\bar{m}_{sf}(x_s) = \prod_{g \in \Gamma(s) \setminus f} m_{gs}(x_s) \propto \frac{p_s(x_s)}{m_{fs}(x_s)}$$

$$m_{fs}(x_s) = \sum_{x_{f \setminus s}} \psi_f(x_f) \prod_{t \in f \setminus s} \bar{m}_{tf}(x_t)$$

Marginal Distribution of Each Variable:

$$p_s(x_s) \propto \prod_{f \in \Gamma(s)} m_{fs}(x_s)$$

Marginal Distribution of Each Factor:
Clique of variables linked by factor.

$$p_f(x_f) \propto \psi_f(x_f) \prod_{s \in f} \bar{m}_{sf}(x_s)$$

Message Passing Inference Summary

- Brute-force enumeration exponential regardless of graph
- Sum-Product BP
  - Exact inference in tree-structure graphs in $O(TK^2)$ time for T nodes, each taking K states
  - Reduces to Forward-Backward in HMMs
  - Same for Max-Product BP (reduces to Viterbi in HMMs)
- Variable elimination
  - Exact marginals in general graphs
  - Worst-case complexity exponential in size of largest clique
  - Need to rerun from scratch for each marginal
  - Complexity dependent on elimination order (NP-hard to optimize)

Message Passing Inference Summary

- Junction Tree Algorithm
  - Exact marginals in general graphs
  - Caches messages to compute all marginals
  - Worst-case complexity exponential in size of largest clique
  - Optimizing Jtree is NP-hard (corresponds to finding treewidth)

- Loopy BP
  - BP updates only depend on tree-structured Markov blanket
  - **Approximate** inference in loopy graphs
  - No guarantees, but works well empirically in many instances
  - Some techniques to improve convergence
    - Message damping
    - Asynchronous message update schedules

Topics

- Probability and Statistics
- Probabilistic Graphical Models
- Message Passing Inference
- Parameter Learning

Maximum Likelihood Estimation

$$\theta^{\mathrm{MLE}} = \arg\max_{\theta} p(\mathcal{Y} \mid \theta) = \arg\max_{\theta} \log p(\mathcal{Y} \mid \theta)$$

If concave then just solve for zero-gradient solution,

Log-Likelihood Function doesn't change argmax since log is monotonic

$$\mathcal{L}(\theta) \equiv \log p(\mathcal{Y} \mid \theta) \qquad \nabla_\theta \mathcal{L}(\theta^{\mathrm{MLE}}) = 0$$

Logarithm serves a couple of practical purposes:

1) Simplifies derivatives for conditionally independent data

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{i=1}^{N} \nabla_\theta \log p(y_i \mid \theta)$$

2) Avoids numerical under/overflow

MLE of Gaussian Mean

Assume data are i.i.d. univariate Gaussian,

Variance is known

$$p(\mathcal{Y} \mid \theta) = \prod_{i=1}^{N} \mathcal{N}(y_i \mid \theta, \sigma^2)$$

Log-likelihood function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{1}{2}(y_i - \theta)^2 \sigma^{-2} \right) \right)$$

Constant doesn't depend on mean

$$= \text{const.} - \frac{1}{2} \sum_{i=1}^{N} \left((y_i - \theta)^2 \sigma^{-2} \right)$$

MLE doesn't change when we:
1) Drop constant terms (in θ)
2) Minimize negative log-likelihood

MLE estimate is *least squares estimator*:

$$\theta^{\text{MLE}} = -\frac{1}{2\sigma^2} \arg\max_{\theta} \sum_{i=1}^{N} (y_i - \theta)^2 = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - \theta)^2$$

Maximum A Posteriori (MAP) Estimation

Recall the MAP estimator maximizes posterior probability,

$$
\begin{aligned}
\theta^{\mathrm{MAP}} &= \arg\max_{\theta} p(\theta \mid \mathcal{Y}) \\
&= \arg\max_{\theta} p(\theta, \mathcal{Y}) && \textbf{(Bayes' rule)} \\
&= \arg\max_{\theta} p(\mathcal{Y} \mid \theta) p(\theta) && \textbf{(Probability Chain Rule)} \\
&= \arg\max_{\theta} \log p(\mathcal{Y} \mid \theta) + \log p(\theta) && \textbf{(Monotonicity of Logarithm)}
\end{aligned}
$$

Prior serves as regularizer in regularized MLE:

$$
\theta^{\mathrm{MLE}} = \arg\max_{\theta} \mathcal{L}(\theta) - \lambda R(\theta)
$$

Learning Summary

Maximum a posteriori (MAP) maximizes posterior probability,

$$\theta^{\mathrm{MAP}} = \arg\max_{\theta} \log p(\theta \mid \mathcal{Y}) = \arg\max_{\theta} \mathcal{L}(\theta) + \log p(\theta)$$

Parameters are *random* quantities with prior $p(\theta)$.

Corresponds to regularized MLE for specific prior/regularizer pair,

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) - \lambda \mathcal{R}(\theta)$$

Gaussian prior=L2, Laplacian prior=L1

Straightforward sequential updating, e.g. Bayesian linear regression