[70240413 Statistical Machine Learning, Spring, 2018]

Probabilistic Graphical Models (II) Inference & Leaning

Jun Zhu

dcszj@mail.tsinghua.edu.cn

http://bigml.cs.tsinghua.edu.cn/~jun

State Key Lab of Intelligent Technology & Systems

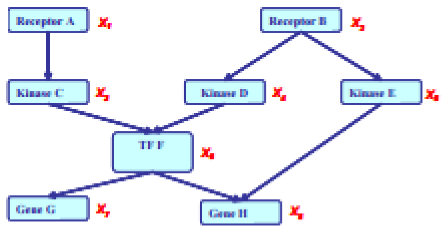
Tsinghua University

May 22, 2018

Two Types of PGMs

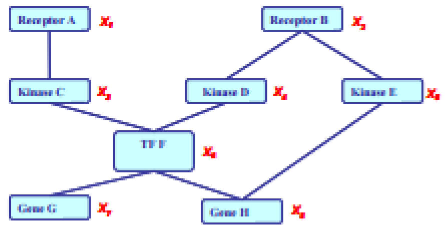
- Directed edges give causality relationships (Bayesian Network or Directed Graphical Models)

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$
$$= P(X_1)\, P(X_2)\, P(X_3 | X_1)\, P(X_4 | X_2)\, P(X_5 | X_2)\, P(X_6 | X_3, X_4)\, P(X_7 | X_6)\, P(X_8 | X_5, X_6)$$



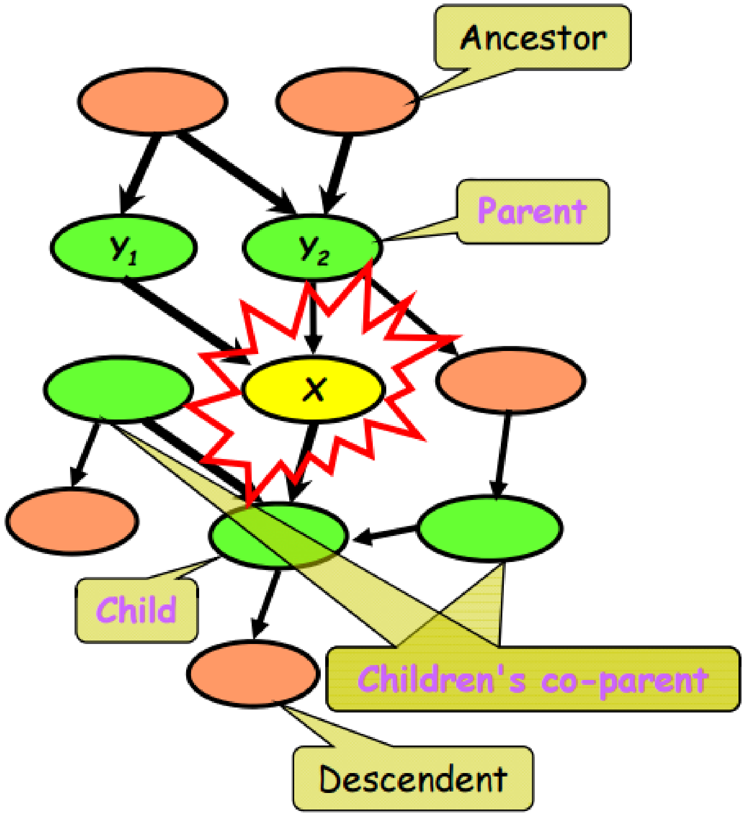
- Undirected edges give correlations between variables (Markov Random Field or Undirected Graphical Models)

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$
$$= 1/Z \exp\{E(X_1)+E(X_2)+E(X_3, X_1)+E(X_4, X_2)+E(X_5, X_2) + E(X_6, X_3, X_4)+E(X_7, X_6)+E(X_8, X_5, X_6)\}$$



Bayesian Networks

- Structure: ***DAG***
- Meaning: a node is **conditionally independent** of every other node in the network outside its **Markov blanket**
- Local conditional distributions (**CPD**) and the **DAG** completely determine the **joint** distribution

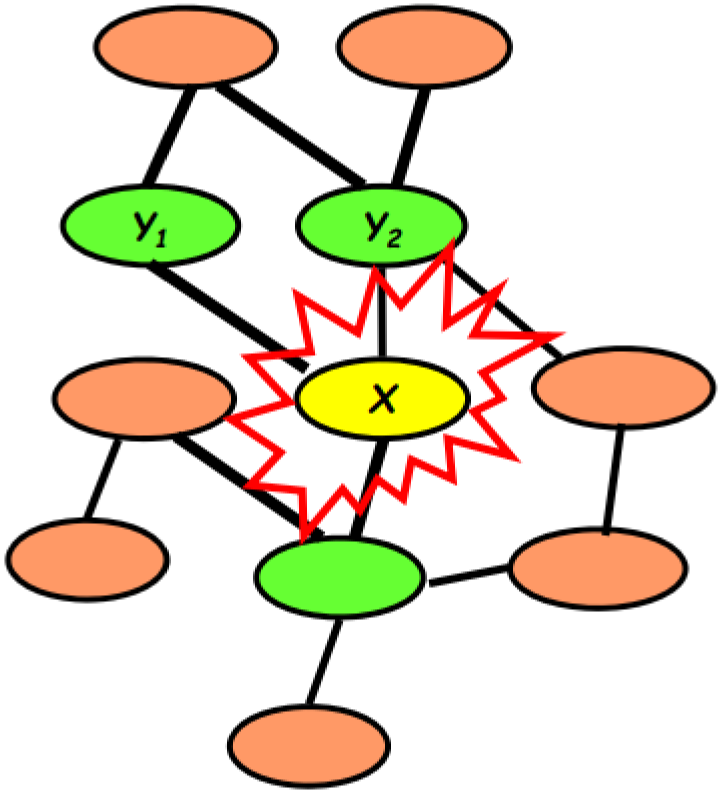


Markov Random Fields

- Structure: ***undirected graph***

- Meaning: a node is **conditionally independent** of every other node in the network given its **Direct Neighbors**

- Local contingency functions (**potentials**) and the **cliques** in the graph completely determine the **joint** distribution



Three Fundamental Questions

- We now have compact representations of probability distributions: Graphical Models
- A GM M describes a unique probability distribution P

- **Typical tasks:**
  - **Inference**
    - How do I answer questions/queries according to my model and/or based on given data?

      e.g.: $P(X_i \mid \mathbf{D})$
  - **Learning**
    - What model is "right" for my data? e.g.: $\mathcal{M} = \arg\max_{\mathcal{M} \in M} F(\mathbf{D}; \mathcal{M})$
    - Note: for Bayesian, they seek $p(M \mid D)$, which is actually an inference problem

Query 1: Likelihood

- Most of the queries one may ask involve **evidence**
  - Evidence e is an assignment of values to a set E variables
  - Without loss of generality

$$\mathbf{E} = \{ X_{k+1}, \ldots, X_n \}$$

- Simplest query: compute probability of evidence

$$P(\mathbf{e}) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, \ldots, x_k, \mathbf{e})$$

Query 2: Conditional Probability

- Often we are interested in the conditional probability distribution of a variable given the evidence

$$P(X \mid \mathbf{e}) = \frac{P(X, \mathbf{e})}{P(\mathbf{e})} = \frac{P(X, \mathbf{e})}{\sum_x P(X = x, \mathbf{e})}$$

  - This is the a posterior belief in X, given evidence e

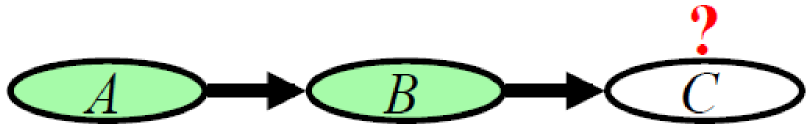
- We usually query a subset of Y of all domain variables X={Y,Z} and "don't care" about the remaining Z:

$$P(\mathbf{Y} \mid e) = \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} \mid \mathbf{e})$$

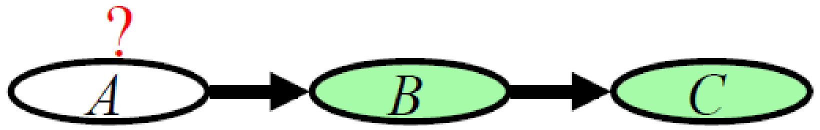
  - The resulting $p(\mathrm{Y} \mid \mathrm{e})$ is called a marginal prob.

Applications of a posterior belief

- **Prediction**: what is the probability of an outcome given the starting condition



  - The query node is a descendent of the evidence
- **Diagnosis**: what is the prob of disease/fault given symptoms

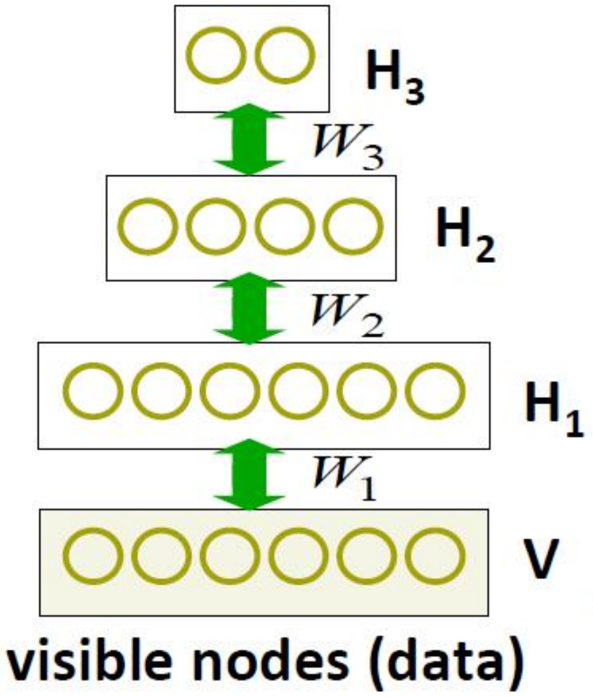


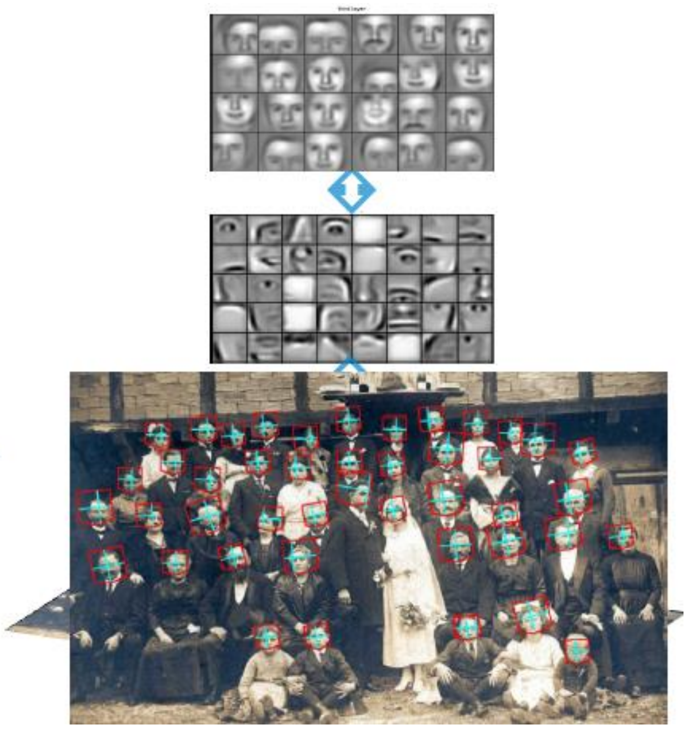
  - The query node is an ancestor of the evidence
- **Learning** under partial observations
  - Fill in the unobserved values under an "EM" setting

- The directionality of info flow between variables is not restricted by the directionality of edges in a GM
  - Posterior inference can combine evidence from all parts of the network

Example: Deep Belief Network

- Deep belief network (DBN) [Hinton et al., 2006]
  - Generative model or RBM with multiple hidden layers
  - Successful applications: OCR, collaborative filtering, multimodal learning





Query 3: Most Probable Assignment

- In this query we want to find the most probably joint assignment (MPA) for some variables of interest

- Such reasoning is usually performed under some given evidence e and ignoring other variables Z:

$$\mathrm{MPA}(\mathbf{Y} \mid \mathbf{e}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y} \mid \mathbf{e}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{\mathbf{z}} P(\mathbf{y}, \mathbf{z} \mid \mathbf{e})$$

  - This is the maximum a posterior configuration of y.

Applications of MPA

- **Classification**:
  - Find most likely label, given the evidence (input features)
- **Explanation**:
  - What is the most likely scenario, given the evidence

- Cautionary note:
  - The MPA of a variable depends on its "context" – the set of variables been jointly queried
  - Example:
    - MPA of Y_1 ?
    - MPA of $(Y_1\ Y_2)$?

y_1	y_2	$P(y_1,y_2)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Complexity of Inference

- Theorem:
  - Computing $P(X=x|e)$ in a GM is NP-hard

- Hardness does not mean we cannot solve inference
  - It implies that we cannot find a general procedure that works efficiently for arbitrary GMs
  - For particular families of GMs, we can have provably efficient procedures.

Approaches to Inference

- <u>**Exact inference algorithms**</u>
  - The elimination algorithm
  - Message-passing algorithm (sum-product, belief propagation)
  - The junction tree algorithms

- <u>**Approximate inference algorithms**</u>
  - Markov chain Monte Carlo methods
  - Variational methods

Marginalization and Elimination

- A signal transduction pathway

$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

  - What's the likelihood that protein E is active?

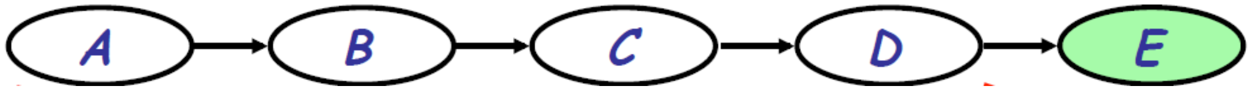
- Query: *P*(e)

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a,b,c,d,e)$$

  - A naive summation needs to enumerate over an exponential # of terms
  - By chain decomposition, we get

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d)$$

Elimination on Chains



- Rearranging terms …

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

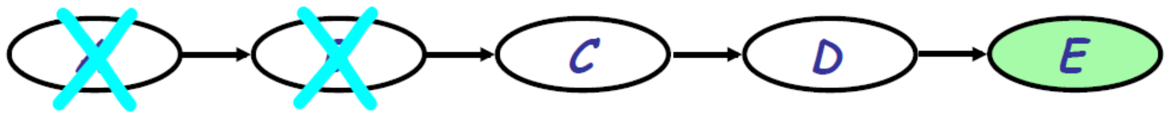
$$= \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)\boxed{\sum_a P(a)P(b \mid a)}$$

- Now, we can perform the innermost summation

$$P(e) = \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)\,p(b)$$

  - This summation "**eliminates**" one variable from our summation argument at a "**local cost**"

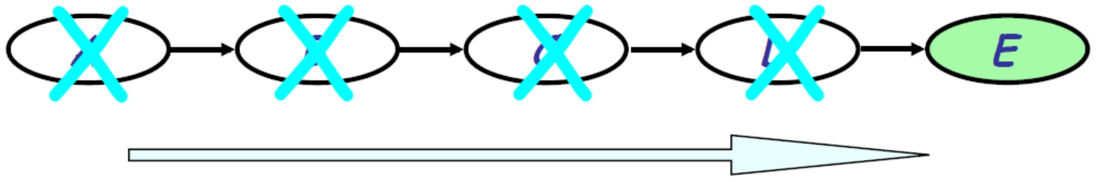
Elimination on Chains



- Rearranging and then summing again, we get

$$
\begin{aligned}
P(e) &= \sum_d \sum_c \sum_b P(c \mid b)\overbrace{P(d \mid c)P(e \mid d)}p(b) \\
&= \sum_d \sum_c P(d \mid c)P(e \mid d)\underbrace{\sum_b P(c \mid b)p(b)} \\
&= \sum_d \sum_c P(d \mid c)P(e \mid d)p(c)
\end{aligned}
$$

Elimination on Chains



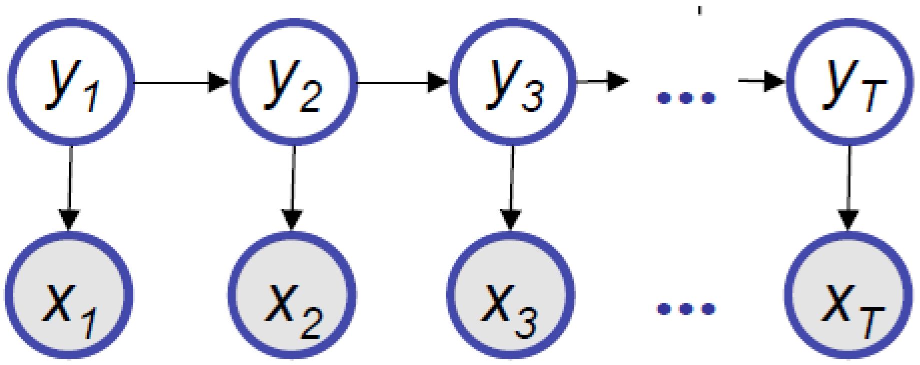
- Eliminate nodes one by one all the way to the end, we get

$$P(e) = \sum_d P(e \mid d)p(d)$$

- Complexity:
  - Each step costs $O(|Val(X_i)| \times |Val(X_{i+1})|)$ operations: $O(nk^2)$
  - Compare to naive evaluation that sums over joint values of n-1 variables $O(k^n)$

Hidden Markov Model

- Now, you can do the marginal inference for HMM:

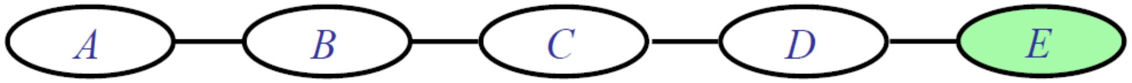


$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \\ &= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T) \end{aligned}$$

- Answer the query:

$$p(y_1 | x_1, \ldots, x_T)$$

Undirected Chains



- Rearranging terms …

$$P(e) = \sum_d \sum_c \sum_b \sum_a \frac{1}{Z} \phi(b,a)\phi(c,b)\phi(d,c)\phi(e,d)$$

$$= \frac{1}{Z} \sum_d \sum_c \sum_b \phi(c,b)\phi(d,c)\phi(e,d) \sum_a \phi(b,a)$$

$$= \cdots$$

The Sum-Product Operation

- In general, we can view the task at hand as that of computing the value of an expression of the form:

$$\sum_{\mathbf{Z}} \prod_{\phi \in \mathcal{F}} \phi$$

  - where is a set of factors

- We call this task the *sum-product* inference task

Inference on General GM via VE

- General idea of Variable Elimination (VE):
  - Write query in the form

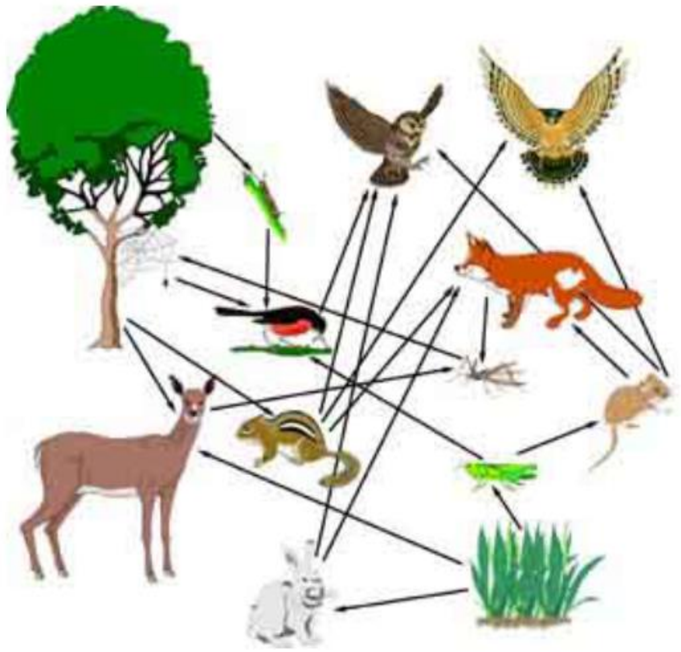
$$P(X_1, \mathbf{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

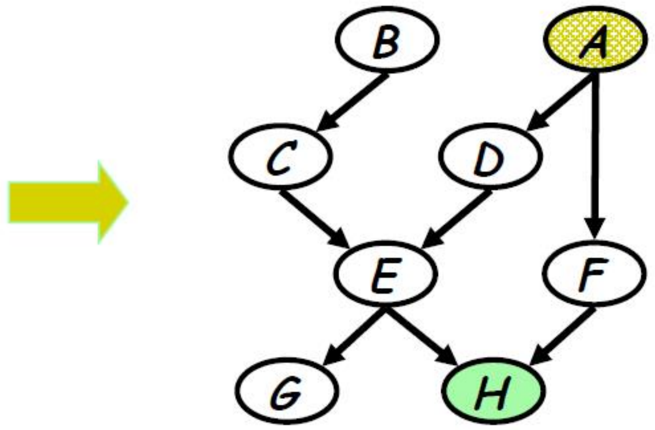
    - This suggests an "**elimination order**" of latent variables

  - Iteratively:
    - Move all irrelevant terms outside of **innermost sum**
    - Perform innermost sum, getting a **new term**
    - Insert the new term into the product

A more complex network

- A food web





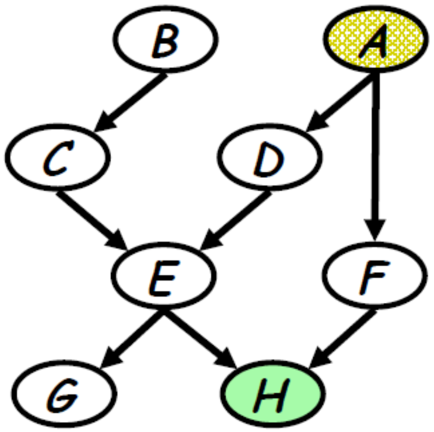
- What is the prob that hawks are leaving given that the grass condition is poor?

Example: VE

- Query: *P(A |h)*
  - Need to eliminate: *B,C,D,E,F,G,H*
- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$

- Choose an elimination order: *H,G,F,E,D,C,B*

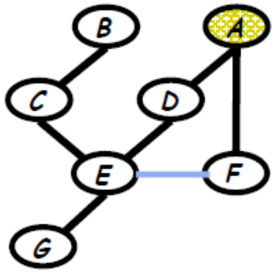


- Step 1:
  - **Conditioning** (fix the evidence node (i.e., *h*) on its observed value (i.e., $\tilde{h}$)):

$$m_h(e,f) = p(h = \tilde{h} \mid e,f)$$

  - This step is isomorphic to a marginalization step:

$$m_h(e,f) = \sum_h p(h \mid e,f)\delta(h = \tilde{h})$$

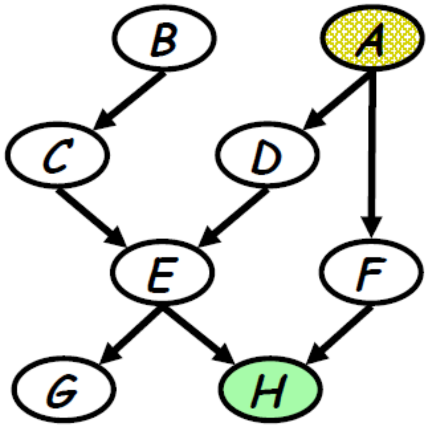


Example: VE

- Query: $P(A \mid h)$
    - Need to eliminate: B,C,D,E,F,G
- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)\underline{P(g \mid e)}m_h(e,f)$$

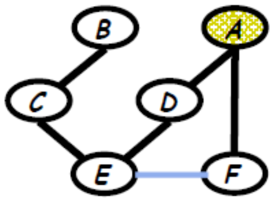


- Step 2: Eliminate G
- compute

$$m_g(e) = \sum_g p(g \mid e) = 1$$

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e)m_h(e,f)$$

$$= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)\underline{m_h(e,f)}$$

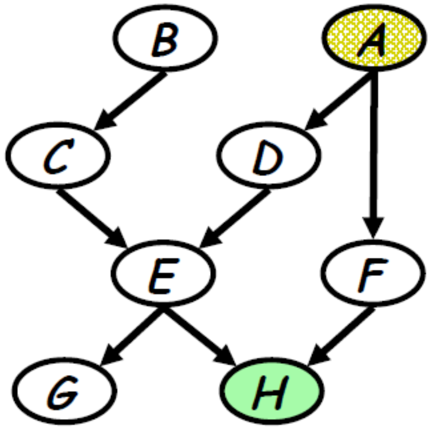


- Keep eliminating F,E,D,C,B in order

Example: VE

- Query: $P(A \mid h)$
  - Need to elimінate: B
- Initial factors:

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$
$$\Rightarrow P(a)m_b(a)$$

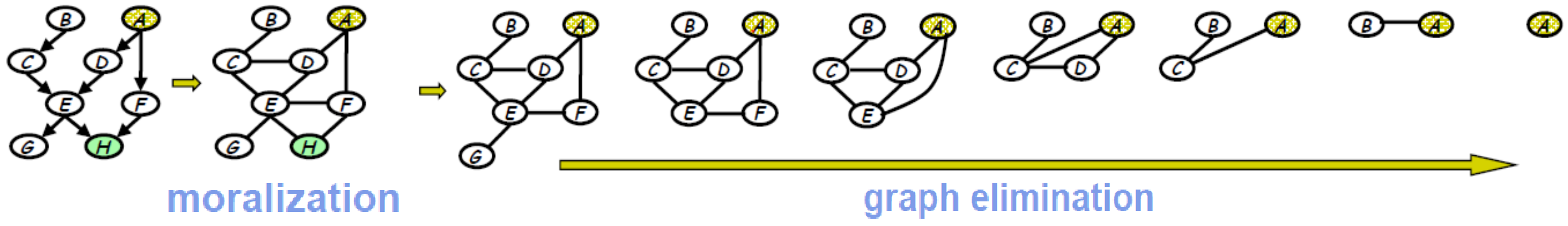


- Final Step: Wrap-up $p(a,\tilde{h}) = p(a)m_b(a)$, $\quad p(\tilde{h}) = \sum_a p(a)m_b(a)$

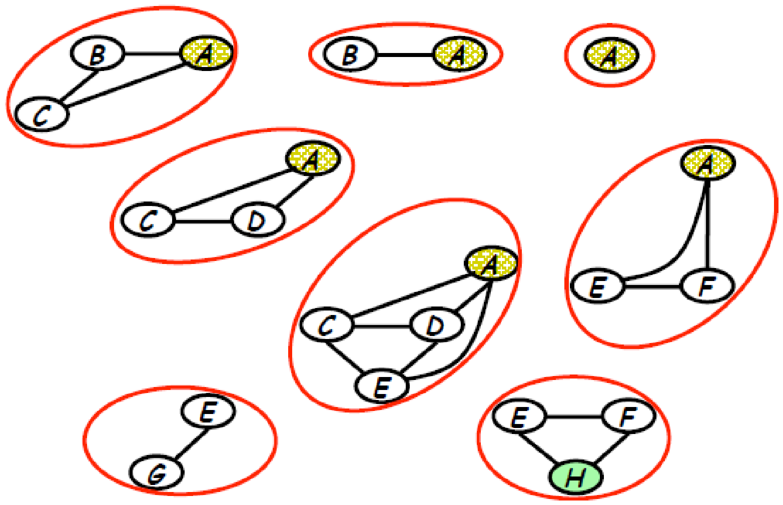
$$\Rightarrow P(a|\tilde{h}) = \frac{p(a)m_b(a)}{\sum p(a)m_b(a)}$$

Understanding VE

- A graph elimination algorithm



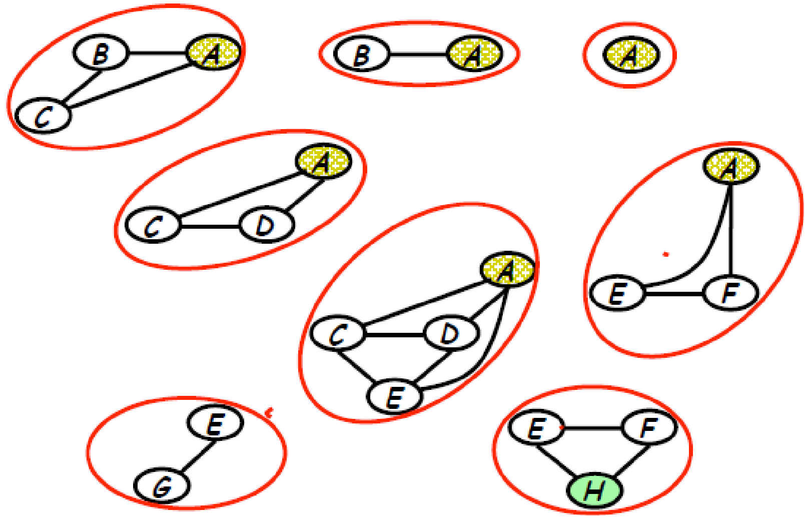
- Intermediate terms correspond to the cliques resulted from elimination



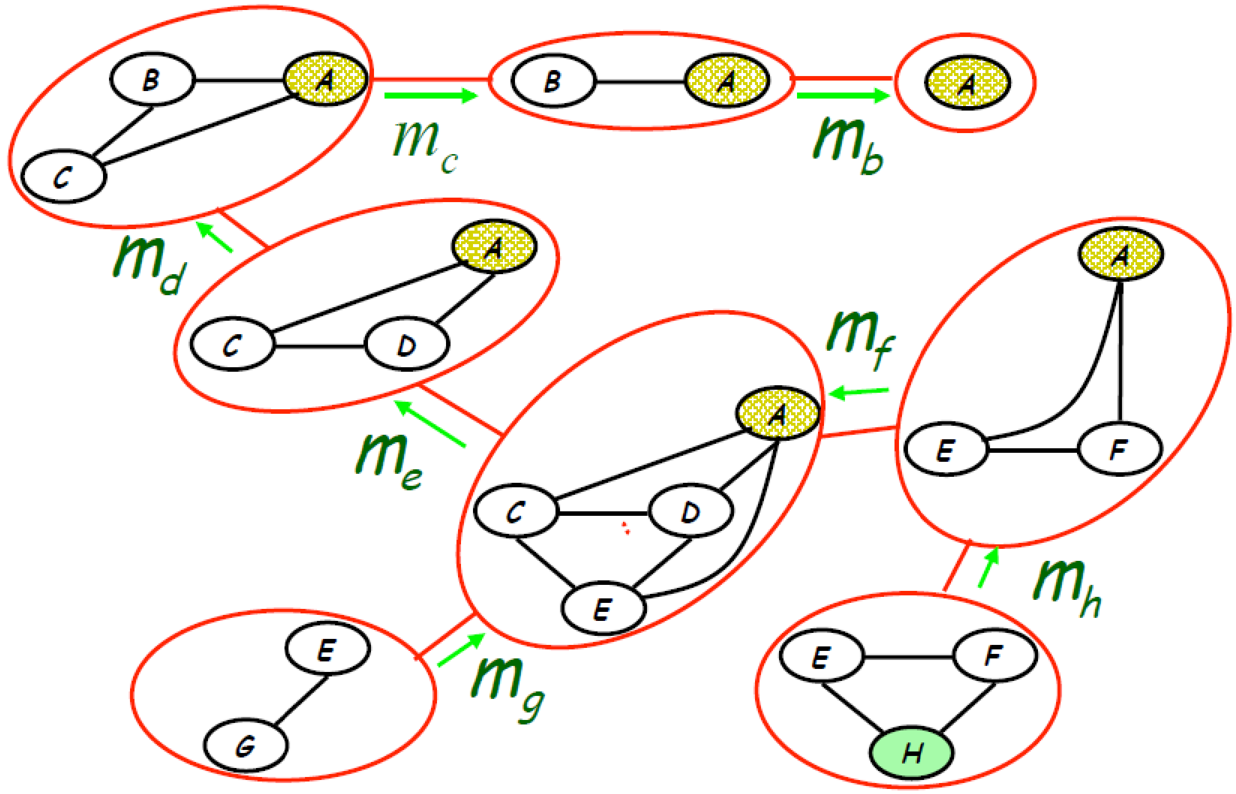
Graph elimination and marginalization

- Induced dependency during marginalization
  - summation ⇔ elimination
  - intermediate term ⇔ elimination clique

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$
$$\Rightarrow P(a)m_b(a)$$



A clique tree



$$m_e(a,c,d) = \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

Complexity

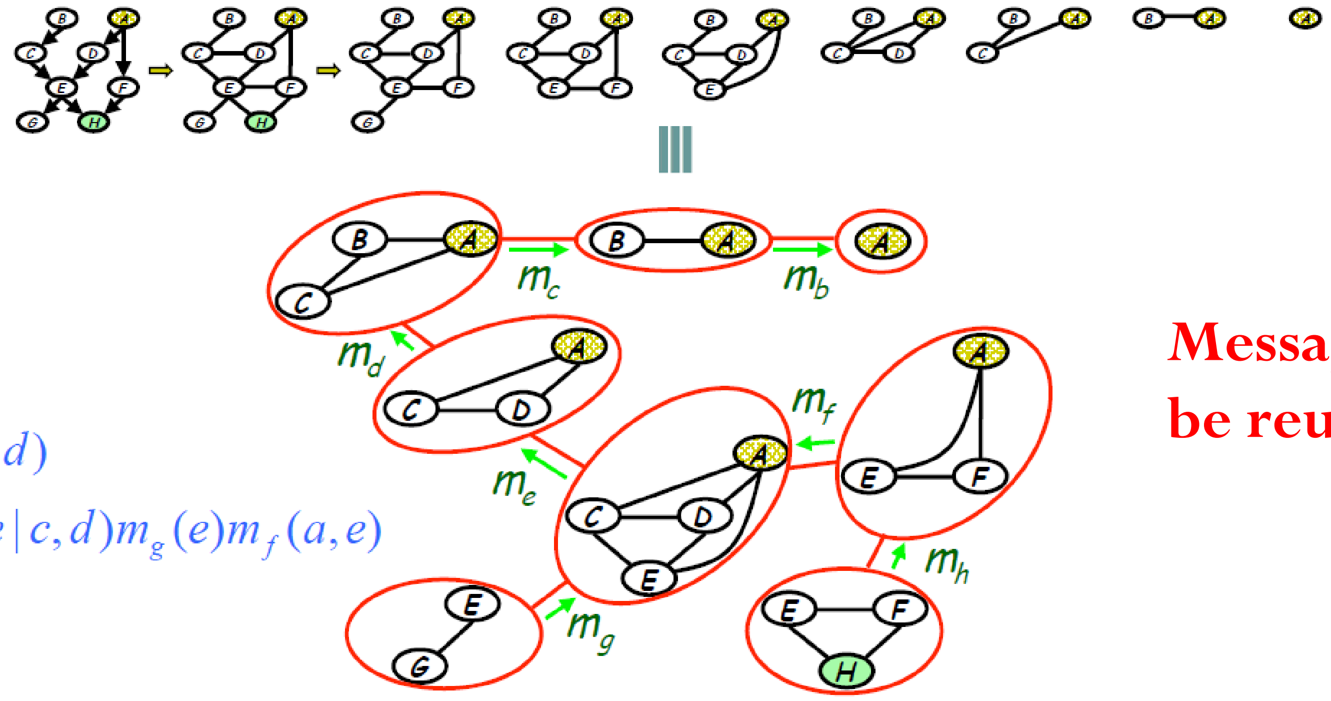
- The overall complexity is determined by the number of largest elimination clique
  - What is the largest elimination clique? – a pure graph theory question
  - "good" elimination orderings lead to small cliques and hence reduce complexity
    - What if we eliminate "e" first in the above graph?
  - Find the best elimination ordering of a graph – NP-hard

    ➔ inference is NP-hard!
  - But there often exist "obvious" optimal or near-opt elimination ordering

From Elimination to Message Passing

- VE answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?

- Elimination ⇔ message passing on a clique tree



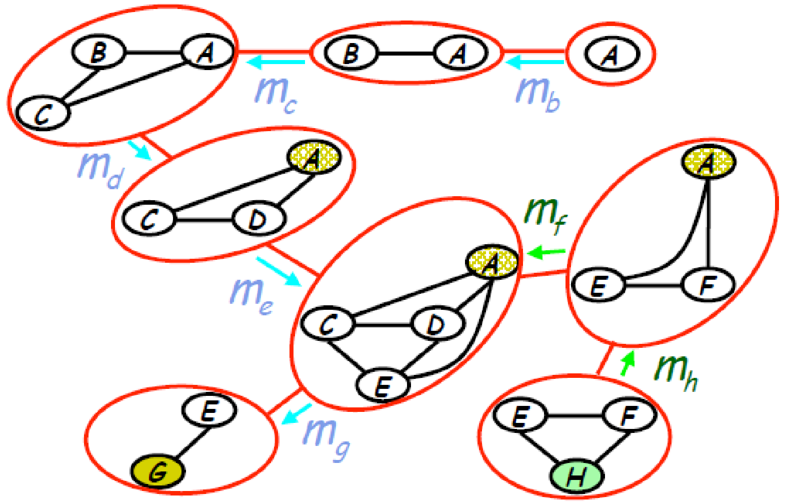
$$m_e(a,c,d) = \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

Messages can be reused!

From Elimination to Message Passing

- VE answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?

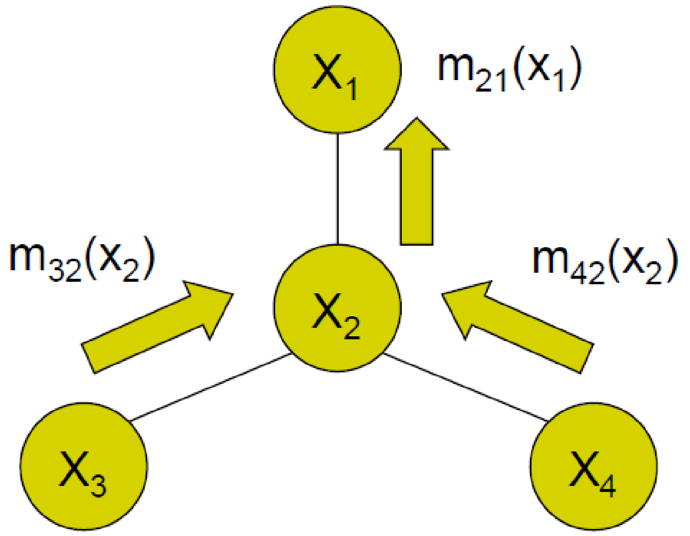
- Elimination ⇔ message passing on a clique tree
  - Another query …



- Messages m_f and m_h are reused, others need to be recomputed

The Message Passing Protocol

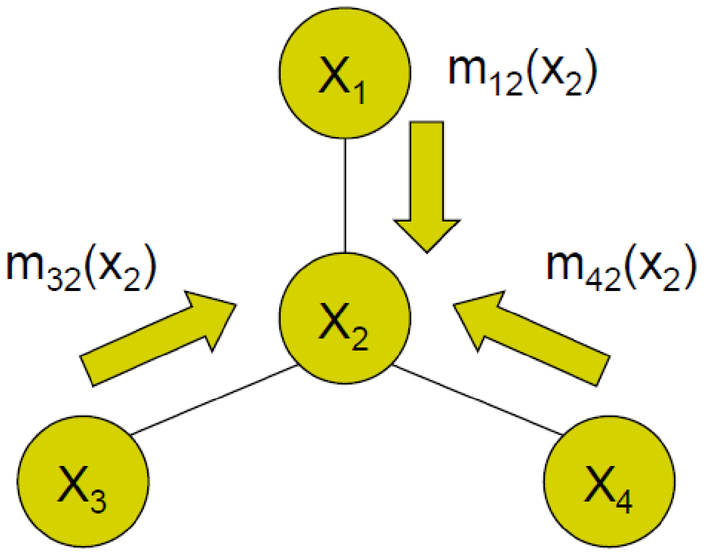
- A node can send a message to its neighbors when (and only when) it has received messages from all its other neighbors
- Computing node marginal:
  - Naive approach: consider each node as the root and execute message passing



Computing $P(X_1)$

The Message Passing Protocol

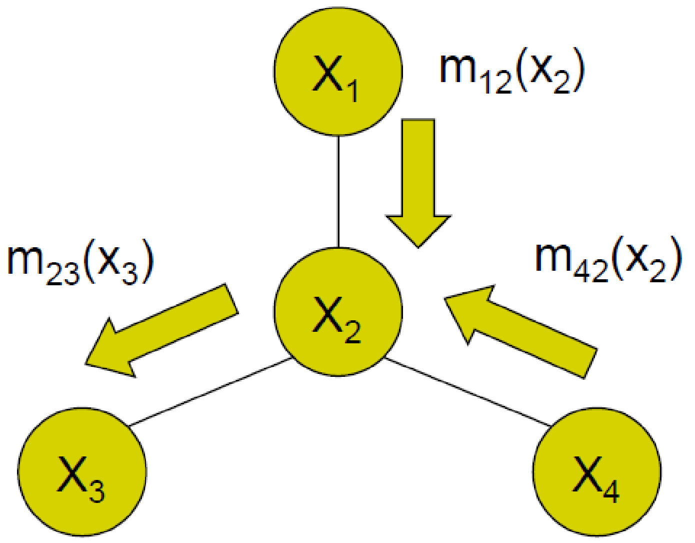
- A node can send a message to its neighbors when (and only when) it has received messages from all its other neighbors
- Computing node marginal:
    - Naive approach: consider each node as the root and execute message passing



Computing $P(X_2)$

The Message Passing Protocol

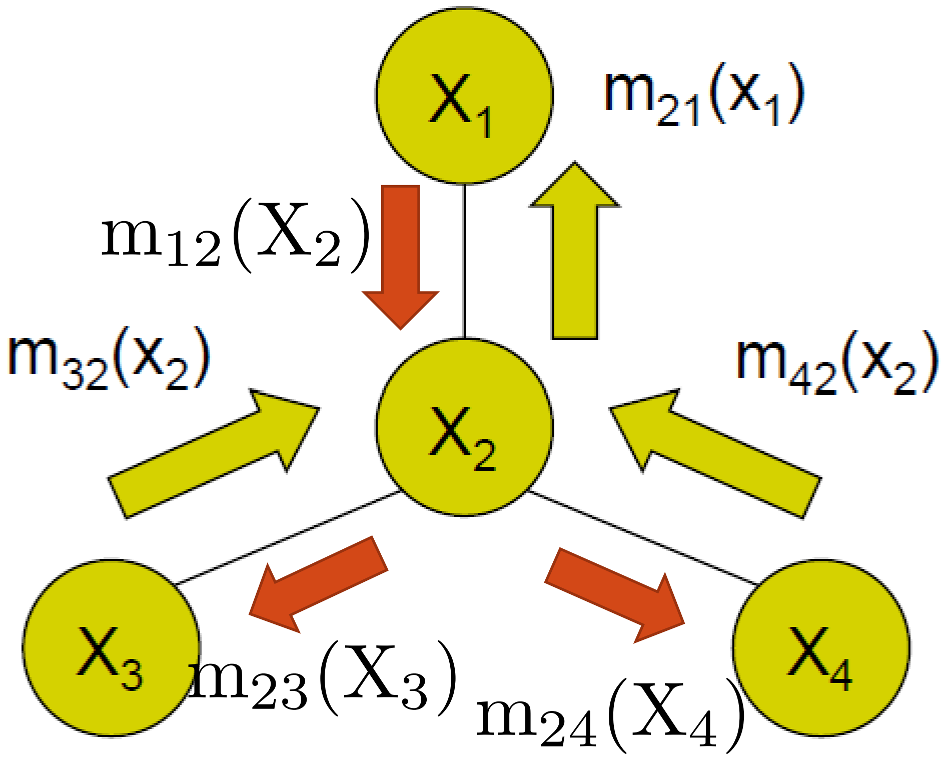
- A node can send a message to its neighbors when (and only when) it has received messages from all its other neighbors
- Computing node marginal:
  - Naive approach: consider each node as the root and execute message passing



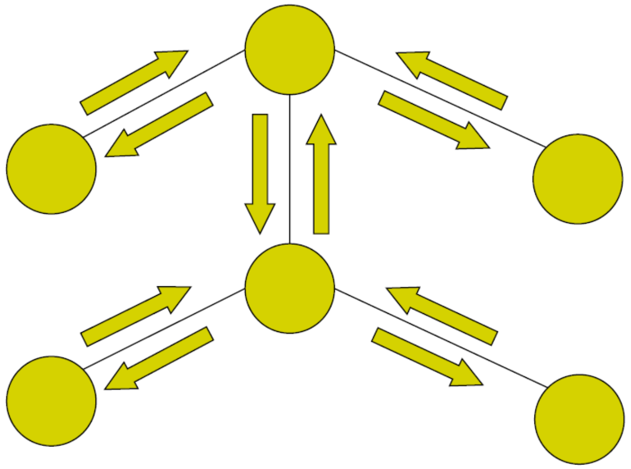
Computing $P(X_3)$

The message passing protocol

- A two-pass algorithm



Belief Propagation: parallel synchronous implementation



$$m_{ji}(x_i) \quad = \quad \sum_{x_j} \left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \right)$$

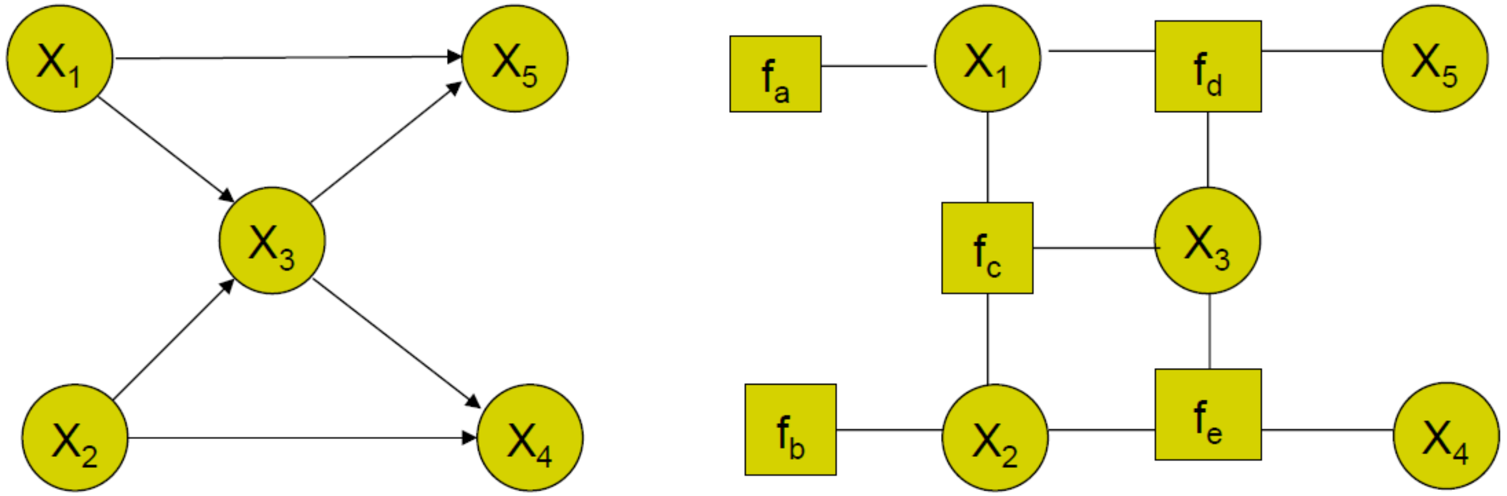
- For a node of degree d, whenever messages have arrived on any subset of d-1 nodes, compute the message for the remaining edge and send!
  - A pair of messages have been computed for each edge, one per direction
  - All incoming messages are eventually computed for each node

Correctness of BP for tree

- **Theorem**: the message passing algorithm guarantees obtaining all marginals in the tree

Another view of M-P: Factor Graph

- Example 1:



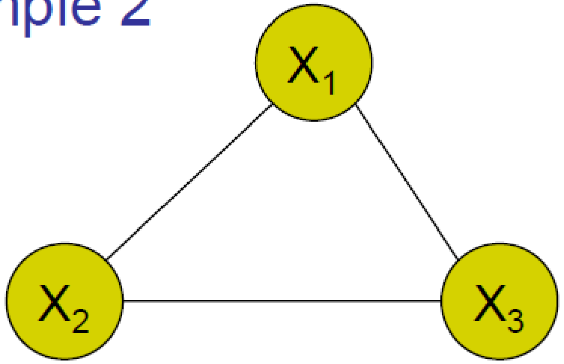
$P(X_1)$ $P(X_2)$ $P(X_3|X_1,X_2)$ $P(X_5|X_1,X_3)$ $P(X_4|X_2,X_3)$

$\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$

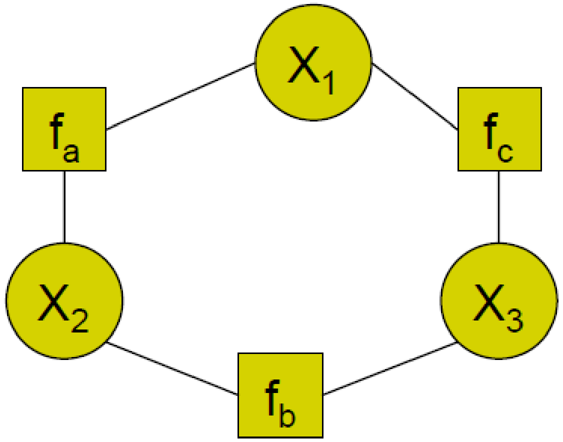
$f_a(X_1)$ $f_b(X_2)$ $f_c(X_3,X_1,X_2)$ $f_d(X_5,X_1,X_3)$ $f_e(X_4,X_2,X_3)$

Factor Graphs

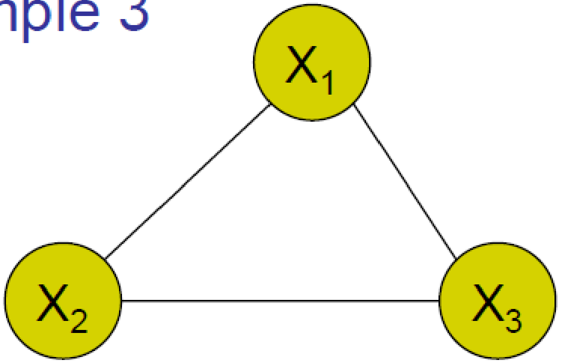
- Example 2



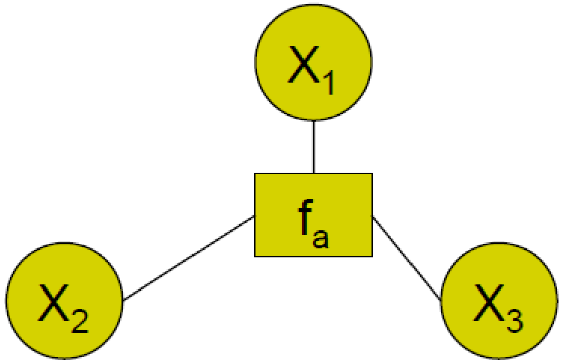
$\psi(x_1,x_2,x_3) = f_a(x_1,x_2)f_b(x_2,x_3)f_c(x_3,x_1)$



- Example 3



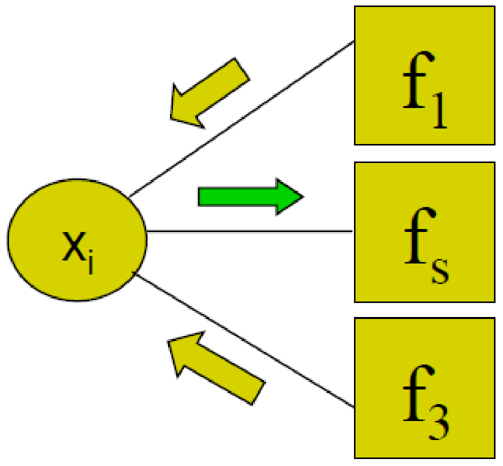
$\psi(x_1,x_2,x_3) = f_a(x_1,x_2,x_3)$



Message Passing on a Factor Tree

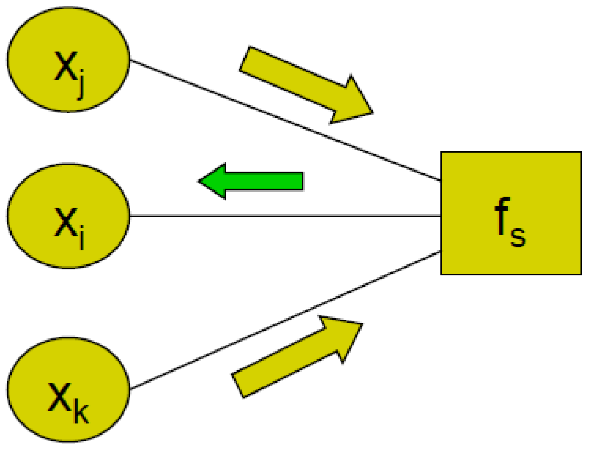
- Two kinds of messages

From variables to factors



$$\nu_{is}(x_i) = \prod_{t \in \mathcal{N}(i) \setminus s} \mu_{ti}(x_i)$$

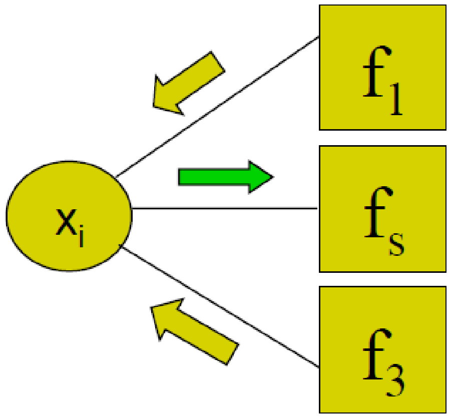
From factors to variables

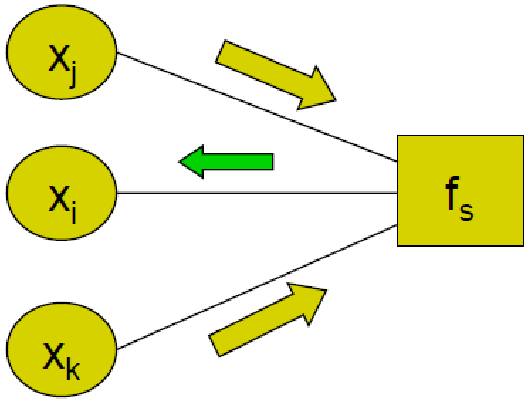


$$\mu_{si}(x_i) = \sum_{x_{\mathcal{N}(s) \setminus i}} \Big(f_s(x_{\mathcal{N}(s)}) \prod_{j \in \mathcal{N}(s) \setminus i} \nu_{js}(x_j) \Big)$$

Message Passing on a Factor Tree

- Message passing protocol:
  - A node can send a message to a neighboring node only when it has received messages from all its other neighbors
- Marginal probability of nodes

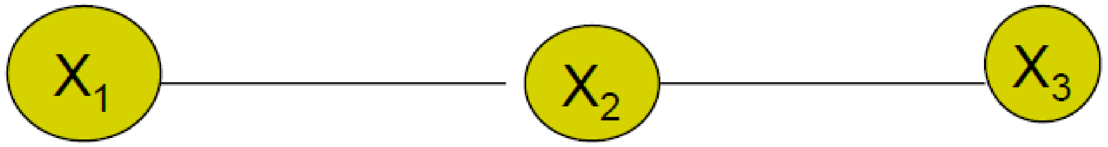


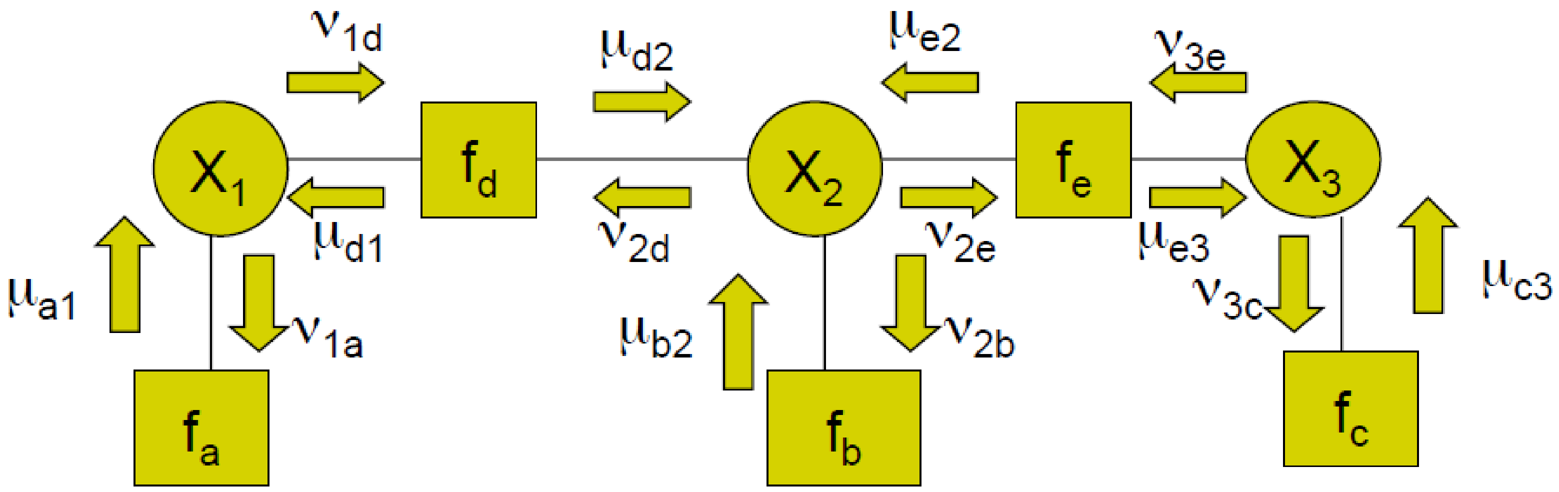


$$P(x_i) \propto \prod_{s \in N(i)} \mu_{si}(x_i)$$
$$\propto \nu_{is}(x_i)\mu_{si}(x_i)$$

BP on a Factor Tree

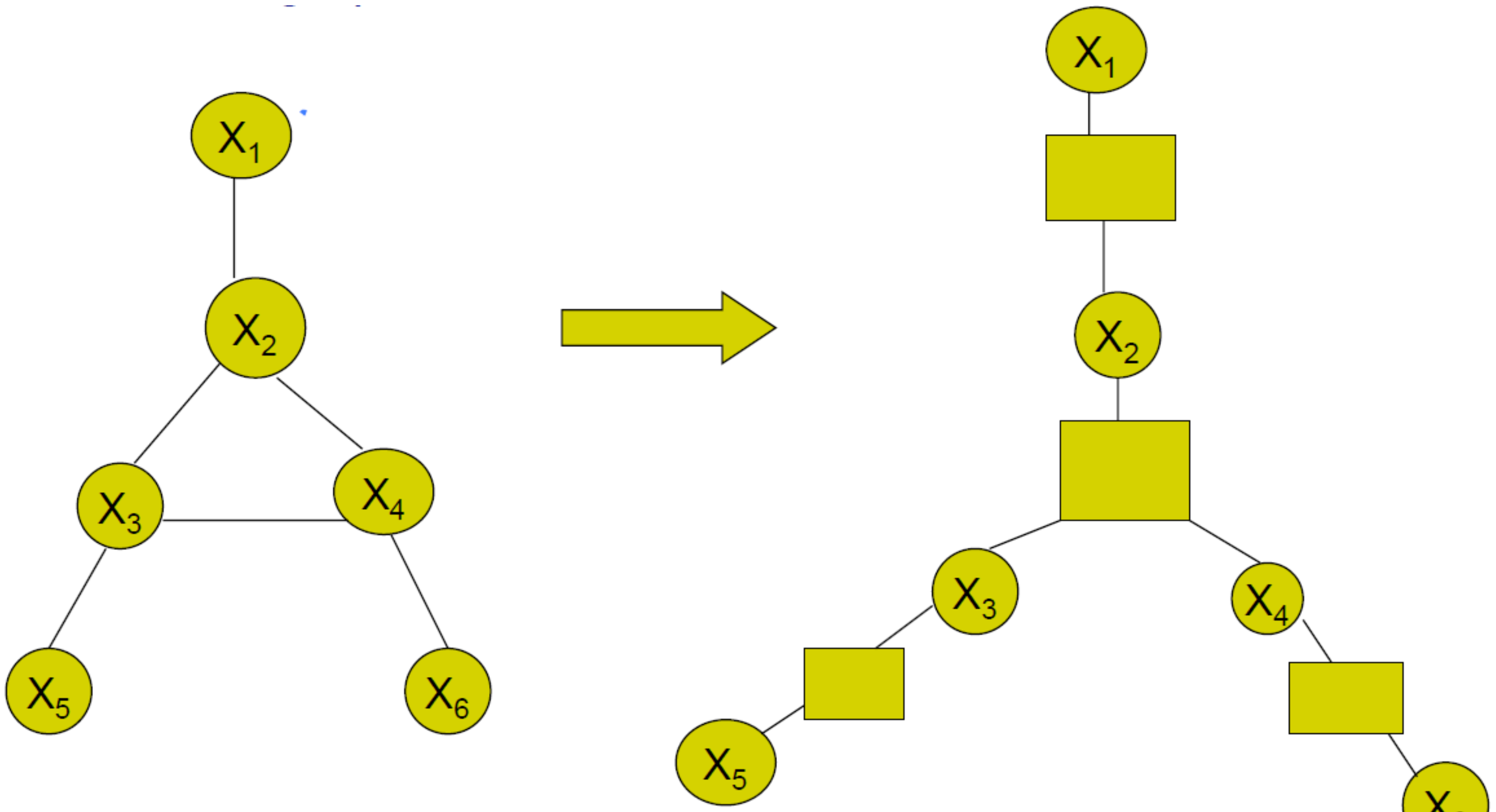
- Two-pass algorithm:





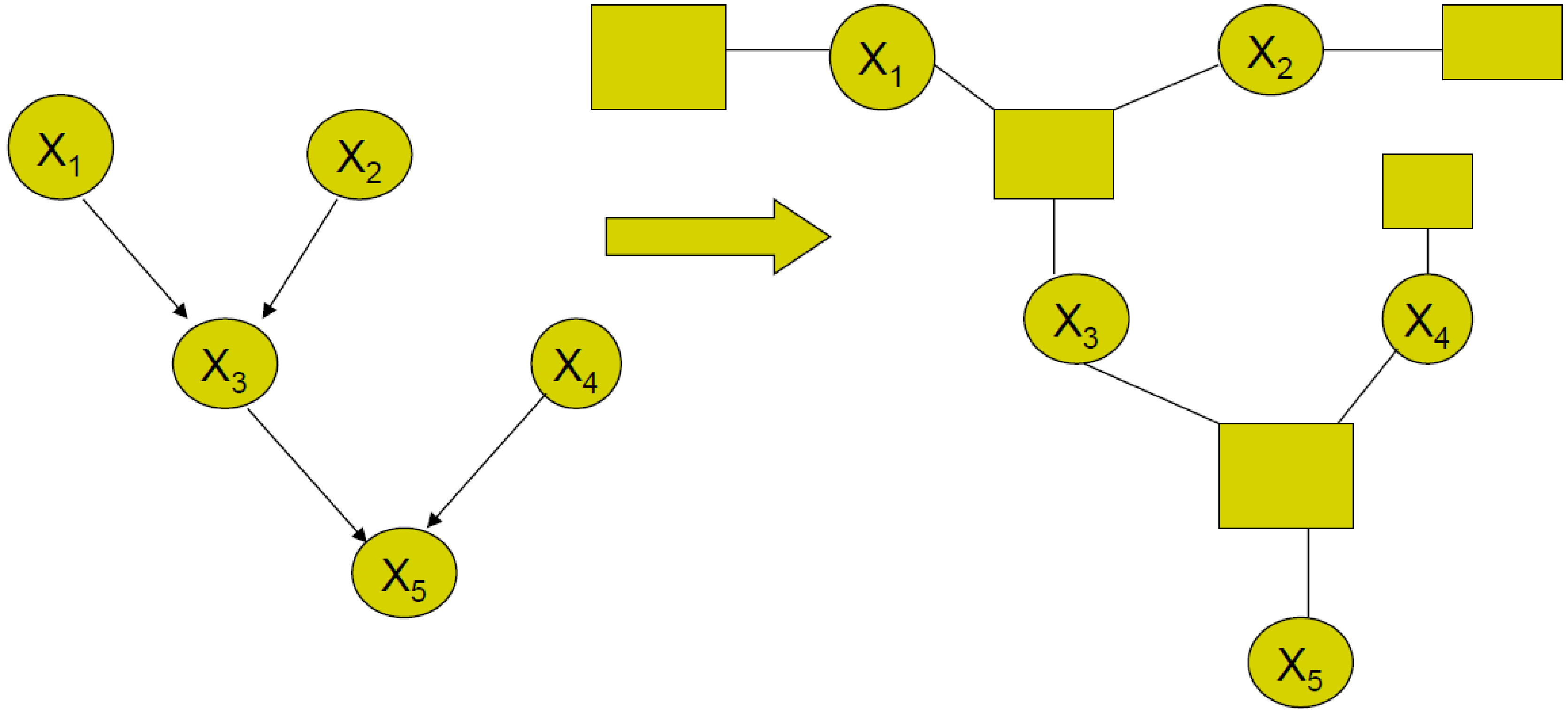
Why factor graph?

- Turn tree-like graphs to factor trees



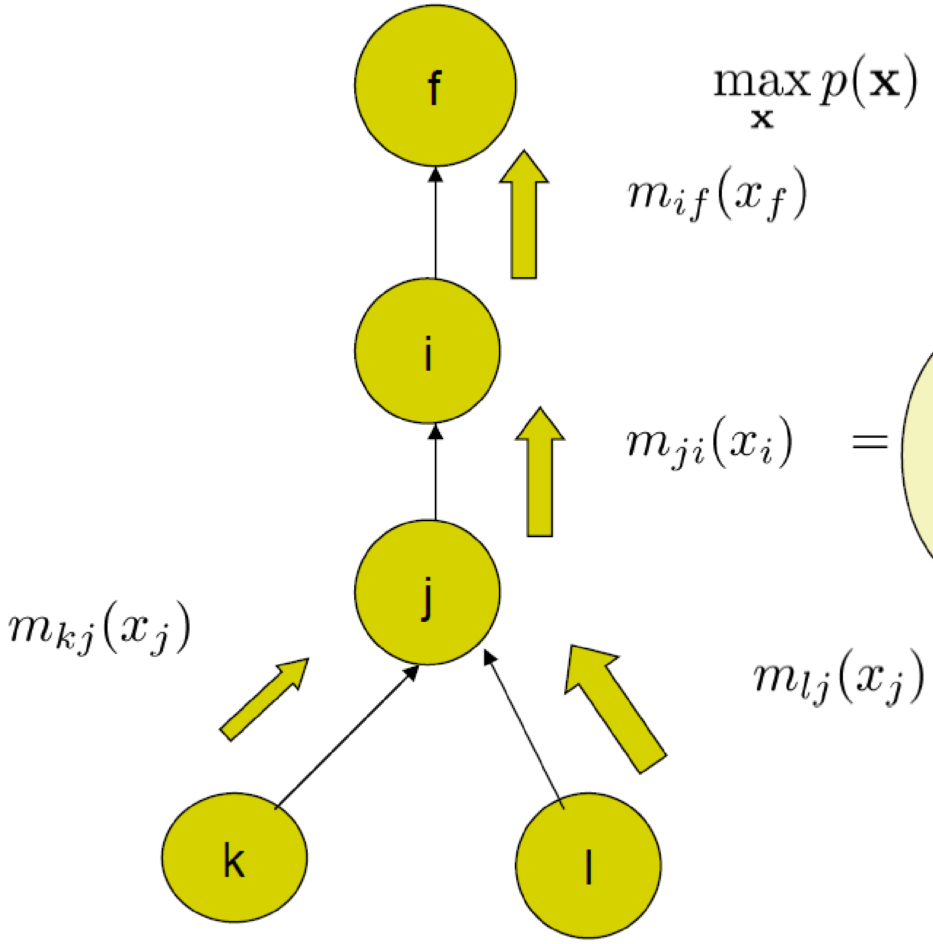
Why factor graph?

- Turn tree-like graphs to factor trees



Trees are a data-structure that guarantees correctness of M-P!

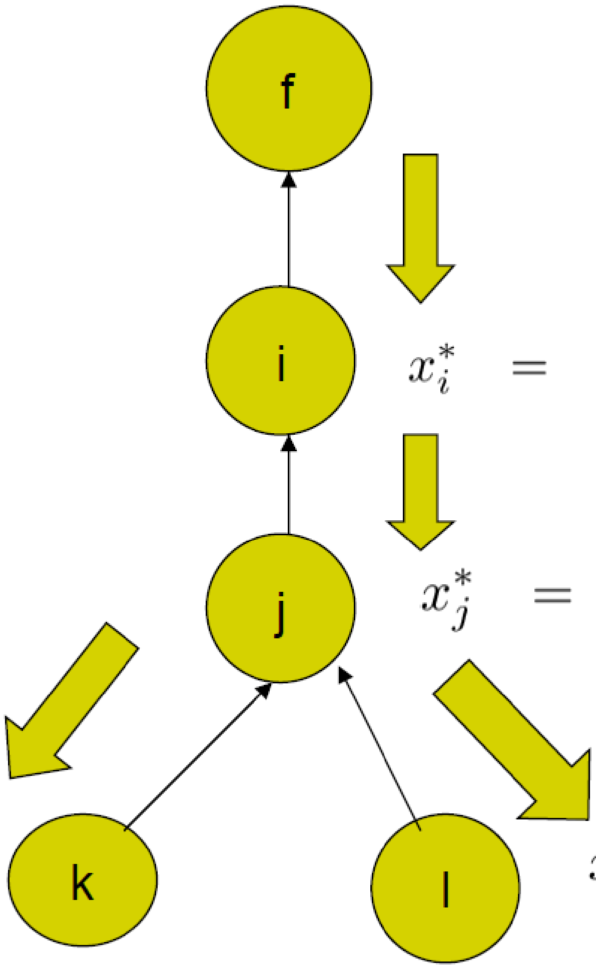
Max-product Algorithm: computing MAP assignment



$$\max_{\mathbf{x}} p(\mathbf{x}) \quad = \quad \max_{x_f}(\psi(x_f) m_{if}(x_f))$$

$$m_{ji}(x_i) \quad = \max_{x_j}\left(\psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j)-i} m_{kj}(x_j)\right)$$

Max-product Algorithm: computing MAP configurations using a final bookkeeping backward pass



$$x_f^* = \arg\max_{x_f}(\psi(x_f)m_{if}(x_f))$$

$$x_i^* = \arg\max_{x_i}\left(\psi(x_i)\psi(x_f^*, x_i)m_{ji}(x_i)\right)$$

$$x_j^* = \arg\max_{x_j}\left(\psi(x_j)\psi(x_i^*, x_j)m_{kj}(x_j)m_{lj}(x_j)\right)$$

$$x_l^* = \arg\max_{x_l}\left(\psi(x_l)\psi(x_l, x_j^*)\right)$$

$$x_k^* = \arg\max_{x_k}\left(\psi(x_k)\psi(x_k, x_j^*)\right)$$

Inference on general GM

- Now, what if the GM is not a tree-like graph?

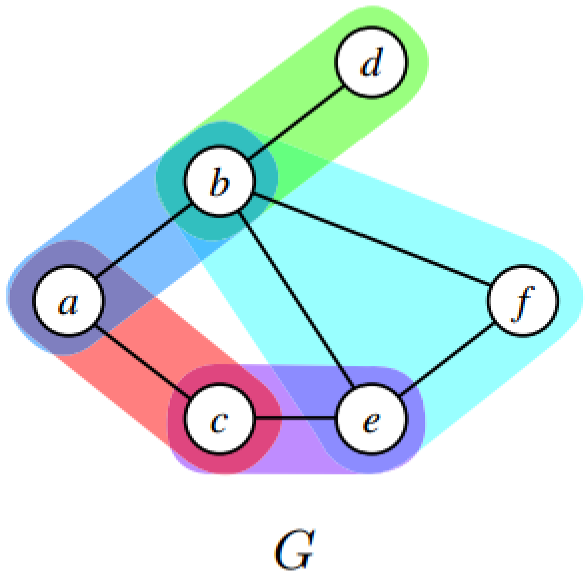
- Can we still directly run message-passing protocol along its edges?

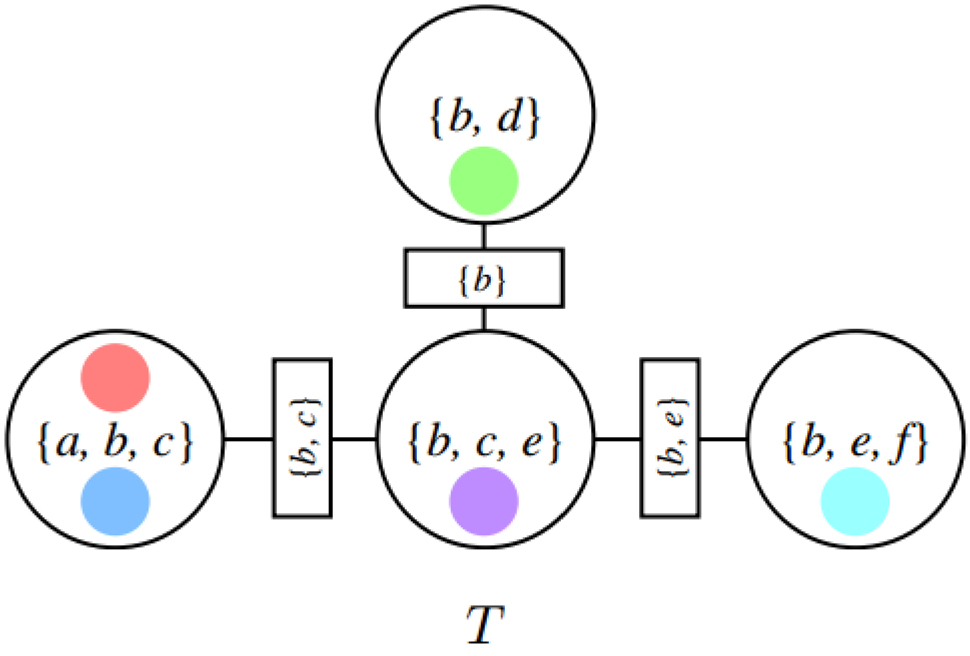
- For non-trees, we do not have the guarantee that message-passing will be consistent

- Then what?
  - Construct a graph data-structure from P that has a tree structure, and run message-passing on it!

  - Junction tree algorithm

Junction Tree





A cluster graph T is a **junction tree** for G if it has these three properties:

1. **singly connected**: there is exactly one path between each pair of clusters.
2. **covering**: for each clique A of G there is some cluster C such that $A \subseteq C$.
3. **running intersection**: for each pair of clusters B and C that contain i, each cluster on the unique path between B and C also contains i.

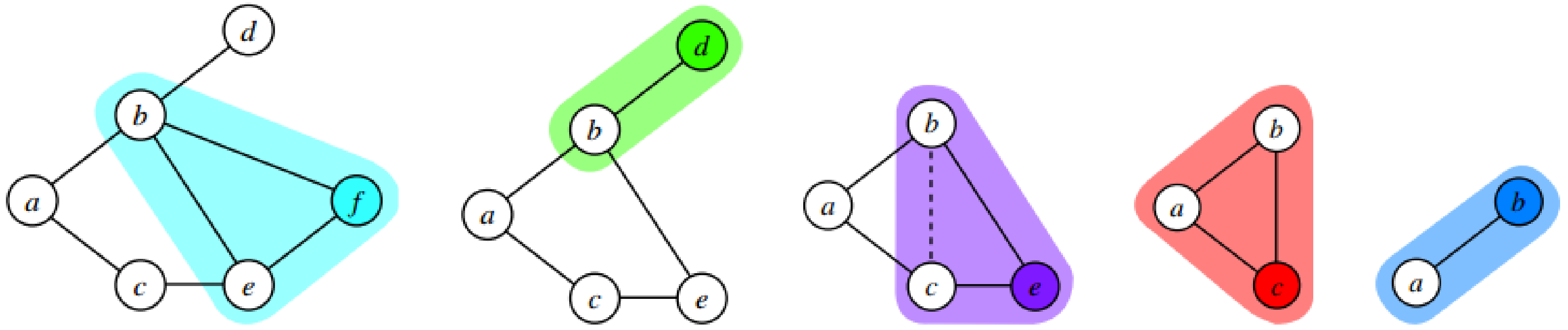
Building Junction Tree

- To build a junction tree:
  1. Choose an ordering of the nodes and use Node Elimination to obtain a set of elimination cliques.
  2. Build a **complete** cluster graph over the **maximal** elimination cliques.
  3. Weight each edge $\{B, C\}$ by $|B \cap C|$ and compute a maximum-weight spanning tree.

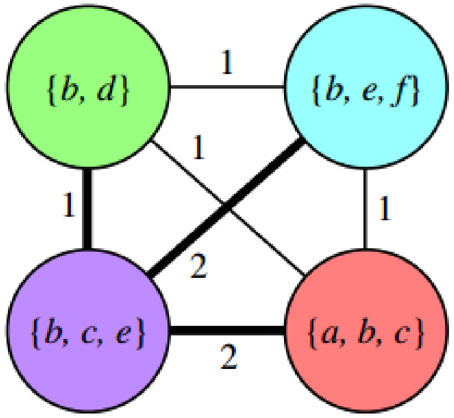
  This spanning tree is a junction tree for G (see Cowell *et al.*, 1999).
- Different junction trees are obtained with different elimination orders and different maximum-weight spanning trees.
- Finding the junction tree with the smallest clusters is an NP-hard problem.

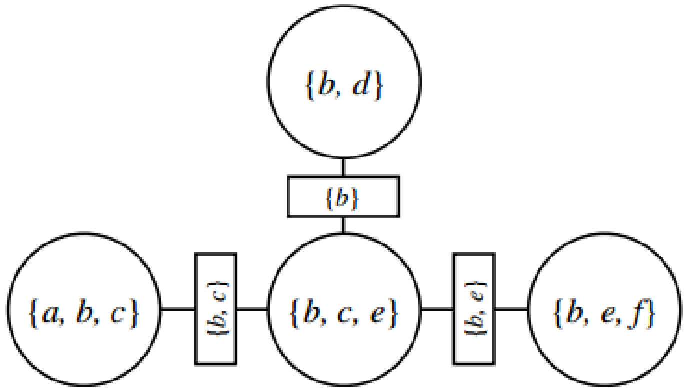
An Example

1. Compute the elimination cliques (the order here is f, d, e, c, b, a).



2. Form the complete cluster graph over the maximal elimination cliques and find a maximum-weight spanning tree.





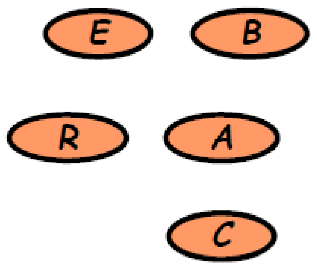
Summary

- Sum-product algorithm computes singleton marginal probabilities on
  - Trees
  - Tree-like graphs
- Maximum a posterior configurations can be computed by replacing sum with max in the sum-product algorithm
- Junction tree data-structure for exact inference on general graphs

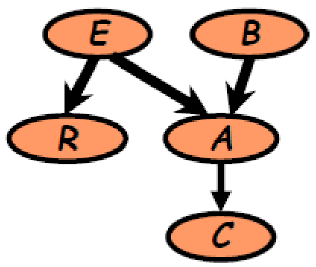
Learning Graphical Models

The goal:

Given set of independent samples (***assignments*** of random variables), find the ***best*** (the most likely?) Bayesian Network (both DAG and CPDs)



(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
....
(B,E,A,C,R)=(F,T,T,T,F)



Structural learning

E	B	P(A | E,B)	
e	$\underline{b}$	0.9	0.1
$\underline{e}$	b	0.2	0.8
$\underline{e}$	$\underline{b}$	0.9	0.1
e	b	0.01	0.99

Parameter learning

Learning Graphical Models

- Scenarios:
  - completely observed GMs
    - directed
    - undirected
  - partially or unobserved GMs
    - directed
    - undirected (an open research topic)
- Estimation principles:
  - Maximal likelihood estimation (MLE)
  - Bayesian estimation
  - Maximal conditional likelihood
  - Maximal "Margin"
  - Maximum entropy
- We use **learning** as a name for the process of estimating the parameters, and in some cases, the topology of the network, from data.

ML Structure Learning for Fully Observed Networks

- Two optimal approaches:
  - "Optimal" here means the employed algorithms guarantee to return a structure that maximizes the objectives (e.g., LogLik)
    - Many heuristics used to be popular, but they provide no guarantee on attaining optimality, interpretability, or even do not have an explicit objective
    - E.g.: structured EM, Module network, greedy structural search, etc.
  - We will learn two classes of algorithms for guaranteed structure learning, which are likely to be the only known methods enjoying such guarantee, but they only apply to certain families of graphs:
    - Trees: The Chow-Liu algorithm
    - Pairwise MRFs: covariance selection, neighborhood-selection (later)

ML Parameter Est. for fully observed Bayesian Networks of given structure

Parameter Learning

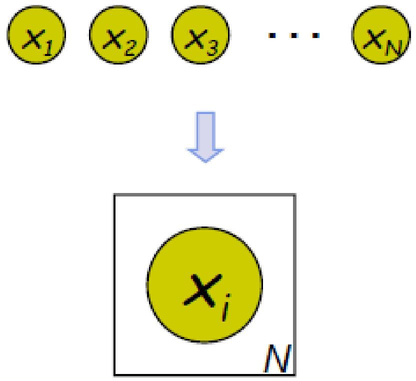
- Assume G is known and fixed,
  - from expert design
  - from an intermediate outcome of iterative structure learning
- Goal: estimate from a dataset of N independent, identically distributed (*iid*) training cases $D = \{x_1, \ldots, x_N\}$.
- In general, each training case $\mathbf{x}_n=(x_{n,1}, \ldots, x_{n,M})$ is a vector of M values, one per node,
  - the model can be completely observable, i.e., every element in x_n is known (no missing values, no hidden variables),
  - or, partially observable, i.e., $\exists i$, s.t. $x_{n,i}$ is not observed.
- **In this lecture we consider learning parameters for a BN with given structure and is completely observable**

$$\ell(\theta; D) = \log p(D \mid \theta) = \log \prod_n \left(\prod_i p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_i \left(\sum_n \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$

Recall Density Estimation

- Can be viewed as a single-node graphical model

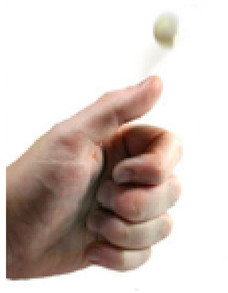
- Instances of exponential family dist.
- Building block of general GM
- MLE and Bayesian estimate



- Recall the example of Bernoulli distribution

$$P(x) = \begin{cases} 1-p & \text{for } x = 0 \\ p & \text{for } x = 1 \end{cases} \quad \Rightarrow \quad P(x) = p^x (1-p)^{1-x}$$

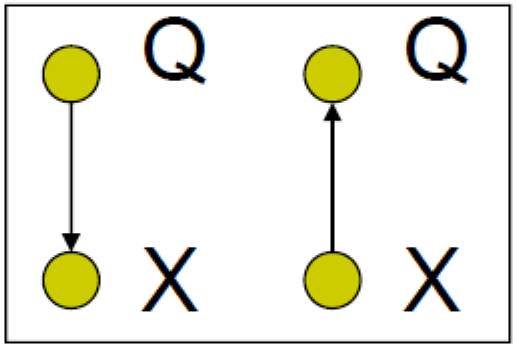
  - MLE gives count frequency
  - Bayes introduces pseudo-counts



Recall Conditional Density Estimation

- Can be viewed as two-node graphical models

- Instances of GLIM
- Building blocks of general GM
- MLE and Bayesian estimate

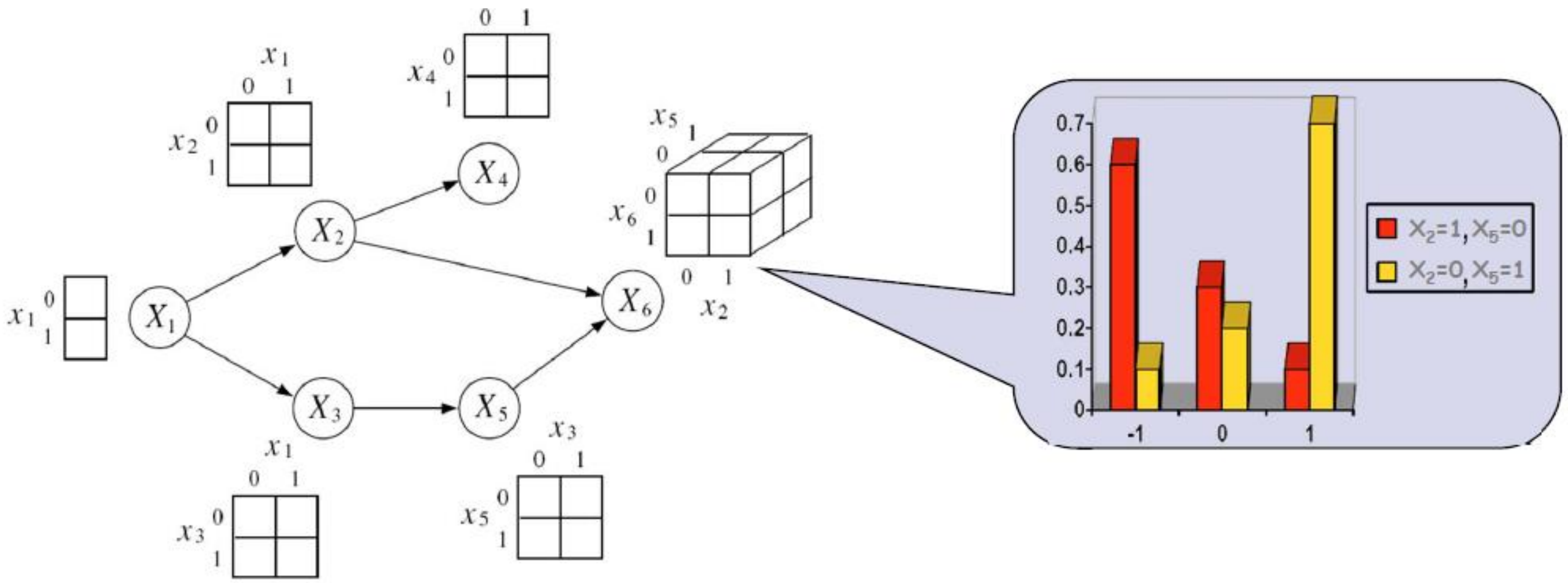


- Recall example of logistic regression
  - We talked about the MLE
  - Bayesian estimate is a bit involved (due to non-conjugacy). We'll come to it in GPs

MLE for general BNs

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood decomposes into a sum of local terms, one per node:

$$\ell(\theta;D)=\log p(D\mid\theta)=\log\prod_n\left(\prod_i p(x_{n,i}\mid \mathbf{x}_{n,\pi_i},\theta_i)\right)=\sum_i\left(\sum_n \log p(x_{n,i}\mid \mathbf{x}_{n,\pi_i},\theta_i)\right)$$

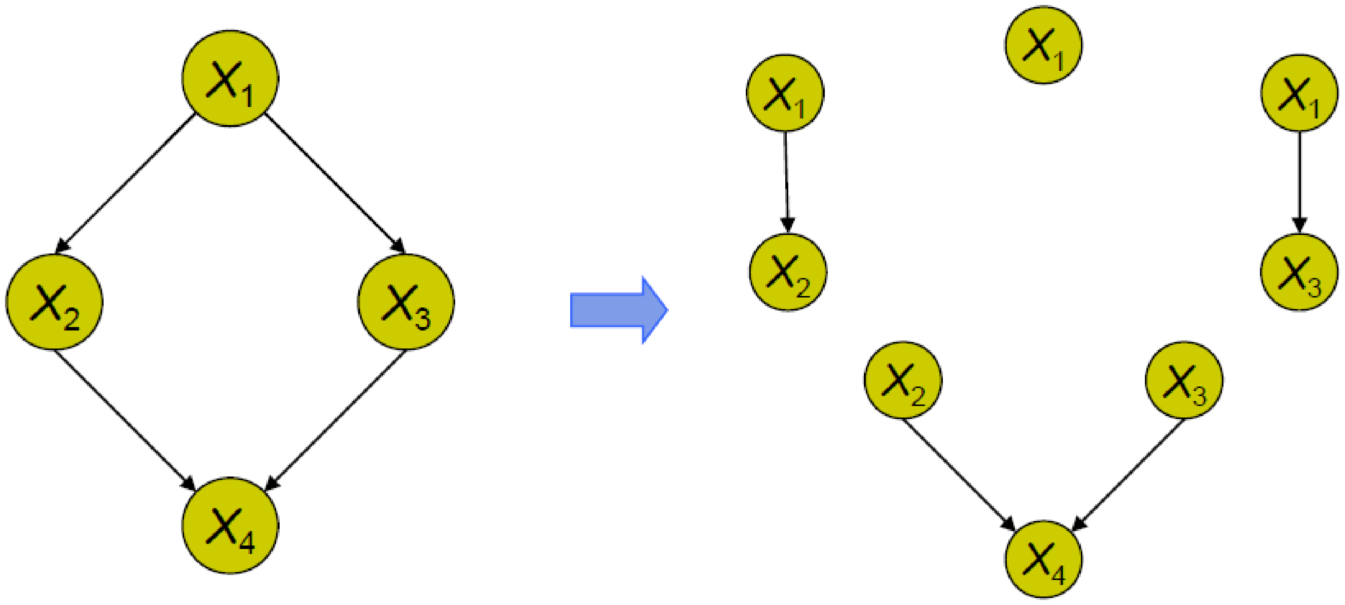


Decomposable likelihood of a BN

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_2) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_4)$$

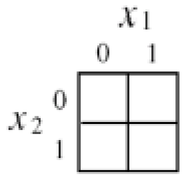
- This is exactly like learning four separate small BNs, each of which consists of a node and its parents

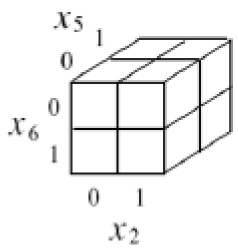


MLE for BNs with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

$$\theta_{ijk} \overset{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$





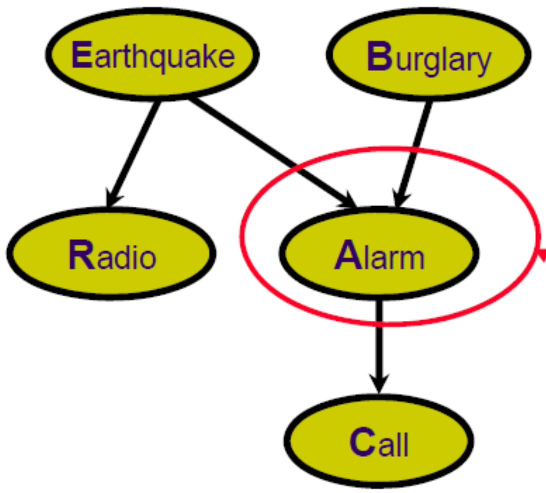
  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

$$n_{ijk} \overset{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is

$$\ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk} \quad \Rightarrow \quad \theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{j'} n_{ij'k}}$$

Bayesian Estimate for BNs



Factorization: $p(\mathbf{X}=\mathbf{x}) = \prod_{i=1}^{M} p(x_i \mid \mathbf{x}_{\pi_i})$

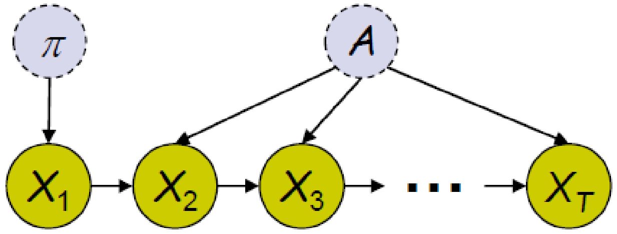
Local Distributions
defined by, e.g., multinomial parameters:

$$p(x_i^k \mid \mathbf{x}_{\pi_i}^j) = \theta_{x_i^k \mid \mathbf{x}_{\pi_i}^j}$$

- How to define a parameter prior? $p(\theta|G)$?
- Assumptions (Geiger & Hecherman, 1997)
  - Global parameter independence [illegible]
  - Local parameter independence [illegible]

Parameter Sharing

- Consider a time-invariant (stationary) 1st-order Markov model



  - Initial state probability vector $\pi_k \overset{\text{def}}{=} p(X_1^k = 1)$
  - State transition probability matrix $A_{ij} \overset{\text{def}}{=} p(X_t^j = 1 \mid X_{t-1}^i = 1)$
- The joint distribution:

$$p(X_{1:T} \mid \theta) = p(x_1 \mid \pi) \prod_{t=2}^{T} \prod_{t=2} p(X_t \mid X_{t-1})$$

- Log-likelihood

$$\ell(\theta; D) = \sum_n \log p(x_{n,1} \mid \pi) + \sum_n \sum_{t=2}^{T} \log p(x_{n,t} \mid x_{n,t-1}, A)$$

- Again, we optimize each parameter separately
  - We have seen how to estimate . What about A?

Learning a Markov chain transition matrix

- A is a stochastic matrix
- Each row of A is multinomial distribution

$$\sum_j A_{ij} = 1$$

- So, MLE of is the fraction of transitions from i to j:

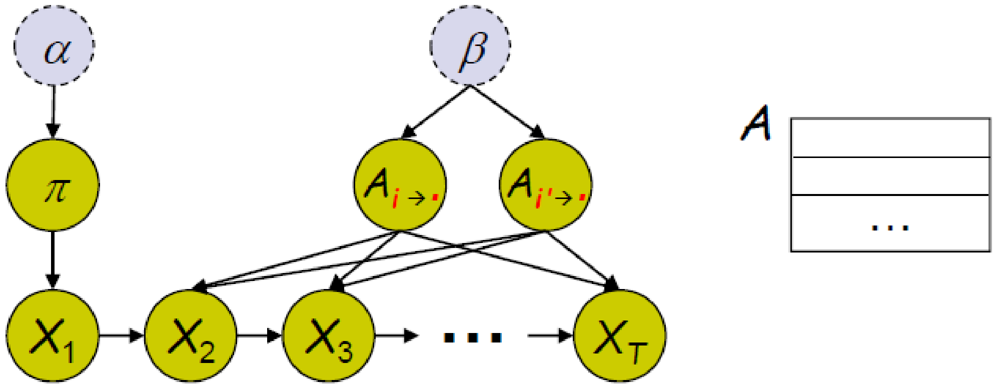
$$A_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=2}^{T} x_{n,t-1}^{i} x_{n,t}^{j}}{\sum_n \sum_{t=2}^{T} x_{n,t-1}^{i}}$$

- Application:
  - If the states represent words, this is called a bigram language model
- Data sparsity problem:
  - If didn't occur in data, we have , then any future sequence with word pair will have zero probability
  - A standard hack: *backoff smoothing*

$$\tilde{A}_{i\rightarrow\cdot} = \lambda\eta + (1-\lambda)A_{i\rightarrow\cdot}^{ML}$$

Bayesian language model

- Interpreted as a Bayesian language model



- If assign a Dirichlet prior to each row of the transition matrix
- We have

$$A_{ij}^{Bayes} \overset{\text{def}}{=} p(j \mid i, D, \beta_i) = \frac{\#(i \to j) + \beta_{i,k}}{\#(i \to \bullet) + |\beta_i|} = \lambda_i \beta'_{i,k} + (1 - \lambda_i) A_{ij}^{ML}, \text{ where } \lambda_i = \frac{|\beta_i|}{|\beta_i| + \#(i \to \bullet)}$$

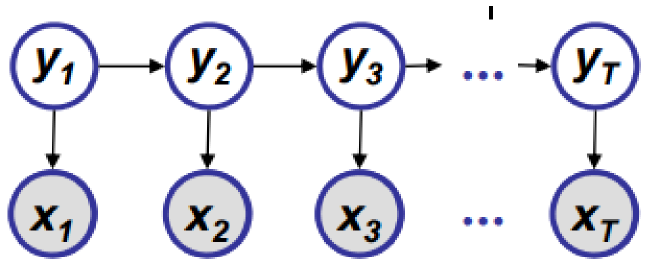
Example: HMMs

- **Supervised learning**: estimation when the "right answer" is known
  - Example:
    - the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- **Unsupervised learning**: estimation when the "right answer" is unknown
  - Example:
    - 10,000 rolls of the casino player, but we don't see when he changes dice

- **Question**: update the parameters of the model to maximize likelihood

Definition of HMM



- Observation space

  Alphabetic set: $\mathrm{C} = \{c_1, c_2, \cdots, c_K\}$

  Euclidean space: $\mathbb{R}^d$

- Index set of hidden states

  $\mathrm{I} = \{1, 2, \cdots, M\}$

- Transition probabilities between any two states

  $p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j}$,

  or $p(y_t \mid y_{t-1}^i = 1) \sim \mathrm{Multinomial}(a_{i,1}, a_{i,1}, \ldots, a_{i,M}), \forall i \in \mathrm{I}.$

- Start probabilities

  $p(y_1) \sim \mathrm{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$

- Emission probabilities associated with each state

  $p(x_t \mid y_t^i = 1) \sim \mathrm{Multinomial}(b_{i,1}, b_{i,1}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$

  or in general:

  $p(x_t \mid y_t^i = 1) \sim \mathrm{f}(\cdot \mid \theta_i), \forall i \in \mathrm{I}.$

Supervised MLE

- Given $x = x_1 \dots x_N$ for which the true state path $y = y_1 \dots y_N$ is known,
  - **Define:**

    A_{ij} = # times state transition $i \rightarrow j$ occurs in **y**

    B_{ik} = # times state i in **y** emits k in **x**

  - **We can show that the maximum likelihood parameters θ are:**

$$a_{ij}^{ML} = \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^{i} y_{n,t}^{j}}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^{i}} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

$$b_{ik}^{ML} = \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^{i} x_{n,t}^{k}}{\sum_n \sum_{t=1}^{T} y_{n,t}^{i}} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

  - **What if x is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian …**

Be aware of the zero-count problem!

Summary: Learning BNs

- For fully observed BN, the log-likelihood function decomposes into a sum of local terms, one per node; thus learning is also factored
  - Structure learning
    - Chow-Liu;
    - Neighborhood selection (later)
  - Learning single-node GM – density estimation: exponential family distribution
  - Learning two-node BN: GLIM
  - Learning BN with more nodes
    - Local operations

ML Parameter Est. for fully observed Markov Random Fields of given structure

MLE for Undirected Graphical Models

- What we have known
  - For directed GMs, the log-likelihood decomposes into a sum of terms, one per family (node plus parents)

- However, for undirected GMs, the log-likelihood does NOT decompose!

$$P(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \psi_c(\mathbf{x}_c) \qquad Z = \sum_{x_1, \ldots, x_n} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

- In general, we will need to do inference (i.e., marginalization) to learn parameters for undirected GMs, even in the fully observed case

Log-likelihood for UGMs with tabular clique potentials

- **Sufficient statistics**: for a UGM (V, E), the number of times that a configuration is observed in a dataset $D=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$

$$m(\mathbf{x}) \overset{\text{def}}{=} \sum_n \delta(\mathbf{x},\mathbf{x}_n) \quad \text{(total count)}, \qquad \text{and} \qquad m(\mathbf{x}_c) \overset{\text{def}}{=} \sum_{\mathbf{x}_{V\setminus c}} m(\mathbf{x}) \quad \text{(clique count)}$$

- In terms of counts, the log-likelihood is

$$p(D|\theta) = \prod_n \prod_{\mathbf{x}} p(\mathbf{x}\mid\theta)^{\delta(\mathbf{x},\mathbf{x}_n)}$$

$$\log p(D|\theta) = \sum_n \sum_{\mathbf{x}} \delta(\mathbf{x},\mathbf{x}_n) \log p(\mathbf{x}\mid\theta) = \sum_{\mathbf{x}} \sum_n \delta(\mathbf{x},\mathbf{x}_n) \log p(\mathbf{x}\mid\theta)$$

$$\ell = \sum_{\mathbf{x}} m(\mathbf{x}) \log\left(\frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c)\right)$$

$$= \sum_c \sum_{\mathbf{x}_c} m(\mathbf{x}_c) \log \psi_c(\mathbf{x}_c) - N\boxed{\log Z}$$

A nasty term!

Derivative of Log-likelihood

- Log-likelihood $\ell = \sum_c \sum_{\mathbf{x}_c} m(\mathbf{x}_c) \log \psi_c(\mathbf{x}_c) - N \log Z$
- First term: $\frac{\partial \ell_1}{\partial \psi_c(\mathbf{x}_c)} = m(\mathbf{x}_c) \Big/ \psi_c(\mathbf{x}_c)$
- Second term: $\frac{\partial \log Z}{\partial \psi_c(\mathbf{x}_c)} = \frac{1}{Z} \frac{\partial}{\partial \psi_c(\mathbf{x}_c)} \left(\sum_{\tilde{\mathbf{x}}} \prod_d \psi_d(\tilde{\mathbf{x}}_d) \right)$

Set the value of variables to $\mathbf{x}$

$$= \frac{1}{Z} \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) \frac{\partial}{\partial \psi_c(\mathbf{x}_c)} \left(\prod_d \psi_d(\tilde{\mathbf{x}}_d) \right)$$

$$= \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) \frac{1}{\psi_c(\tilde{\mathbf{x}}_c)} \frac{1}{Z} \prod_d \psi_d(\tilde{\mathbf{x}}_d)$$

$$= \frac{1}{\psi_c(\mathbf{x}_c)} \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) p(\tilde{\mathbf{x}}) = \frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

Conditions on Clique Marginals

- Derivative of log-likelihood

$$\frac{\partial \ell}{\partial \psi_c(\mathbf{x}_c)} = \frac{m(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)} - N\frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

- Hence, for the ML parameters, we know that

$$p^*_{MLE}(\mathbf{x}_c) = \frac{m(\mathbf{x}_c)}{N} \overset{\text{def}}{=} \tilde{p}(\mathbf{x}_c)$$

- In other words, at the ML setting of the parameters, for each clique, the model marginal must be equal to the observed marginal (empirical counts)

- Note: this condition doesn't tell us how to get the ML parameters!

MLE for decomposable UGMs

- **Decomposable models**
  - G is decomposable $\Leftrightarrow$ G is triangulated $\Leftrightarrow$ G has a junction tree
  - Potential based representation: $p(\mathbf{x}) = \frac{\prod_c \psi_c(\mathbf{x}_c)}{\prod_s \varphi_s(\mathbf{x}_s)}$
- Consider a chain $X_1 - X_2 - X_3$
  - The cliques are (X_1,X_2) (X_2,X_3) ; the separator is X_2
  - The empirical marginal must equal the model marginal
- Let's guess that

$$\hat{p}_{MLE}(x_1,x_2,x_3) = \frac{\tilde{p}(x_1,x_2)\tilde{p}(x_2,x_3)}{\tilde{p}(x_2)}$$

  - We can verify that such a guess satisfies the condition

$$\hat{p}_{MLE}(x_2,x_3) = \tilde{p}(x_2,x_3)$$

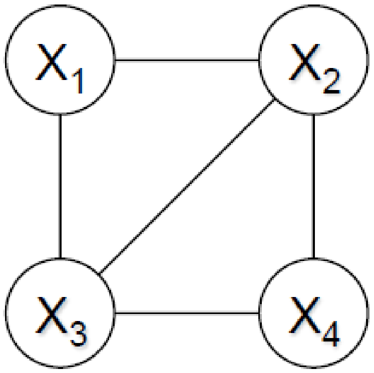
  - Similar for (x_1,x_2)

MLE for decomposable UGMs (cont.)

- Let's guess that $\hat{p}_{MLE}(x_1,x_2,x_3) = \frac{\tilde{p}(x_1,x_2)\tilde{p}(x_2,x_3)}{\tilde{p}(x_2)}$
- To compute clique potentials, just equate them to the empirical marginal (or conditionals). Then Z=1:

$$\psi_{12}^{MLE}(x_1,x_2) = \tilde{p}(x_1,x_2) \qquad \psi_{23}^{MLE}(x_2,x_3) = \frac{\tilde{p}(x_2,x_3)}{\tilde{p}(x_2)} = \tilde{p}(x_3 \mid x_2)$$

- One more example:



$$\hat{p}_{MLE}(x_1,x_2,x_3,x_4) = \frac{\tilde{p}(x_1,x_2,x_3)\tilde{p}(x_2,x_3,x_4)}{\tilde{p}(x_2,x_3)}$$

$$\psi_{123}^{MLE}(x_1,x_2,x_3) = \frac{\tilde{p}(x_1,x_2,x_3)}{\tilde{p}(x_2,x_3)} = \tilde{p}(x_1 \mid x_2,x_3)$$

$$\psi_{234}^{MLE}(x_2,x_3,x_4) = \tilde{p}(x_2,x_3,x_4)$$

Iterative Proportional Fitting (IPF)

- From the derivative of log-likelihood

$$\frac{\partial \ell}{\partial \psi_c(\mathbf{x}_c)} = \frac{m(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)} - N\frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

- We derive another relationship:

$$\frac{\tilde{p}(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)} = \frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

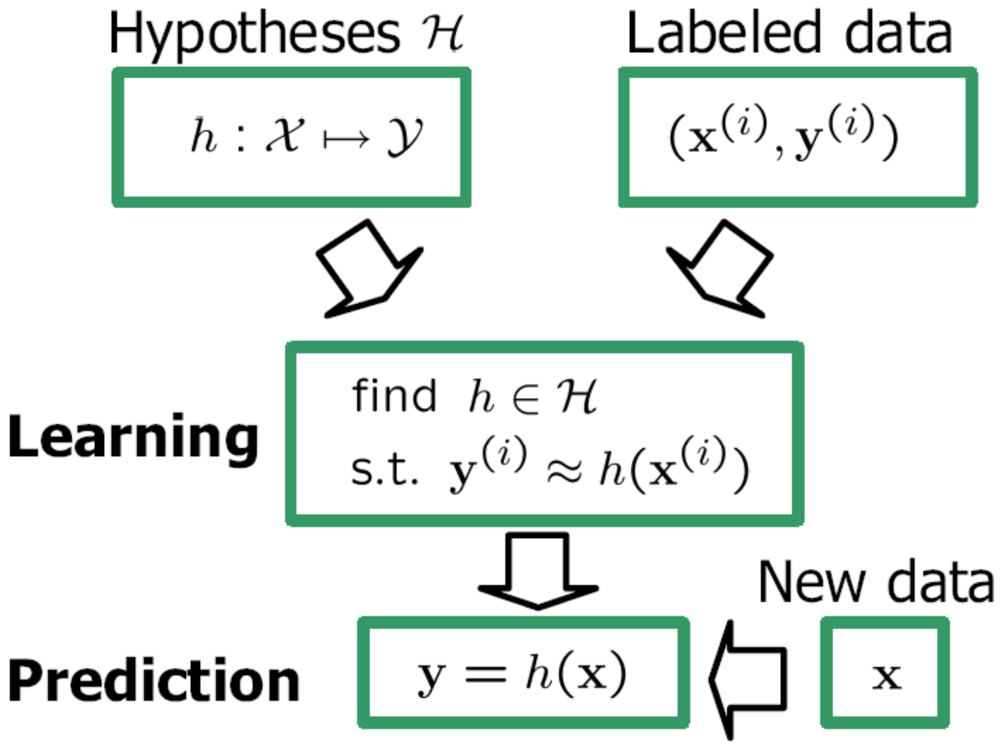
  - Note that ψ_c appears implicitly in the model marginal
- This is therefore a fixed-point equation for ψ_c
- The idea of IPF is to hold ψ_c fixed on the R.H.S and solve it on the L.H.S. We cycle through all cliques and iterate:

$$\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c)\frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$$

Feature-based Clique Potentials

- We use CRFs as an example to explain!

Classical Supervised Learning


Hypotheses $\mathcal{H}$
$h : \mathcal{X} \mapsto \mathcal{Y}$
Labeled data
$(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$
Learning
find $h \in \mathcal{H}$
s.t. $\mathbf{y}^{(i)} \approx h(\mathbf{x}^{(i)})$
New data
$\mathbf{x}$
Prediction
$\mathbf{y} = h(\mathbf{x})$

$$\mathrm{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \mathrm{L}\ , x_d^{(i)} \right]^T$$

$$\mathrm{y}^{(i)} \in \{ c_1, c_2, \mathrm{L}\ , c_M \}$$

Supervised Setting (figure from Taskar'05)

Sequential Labeling

- Example: POS (part-of-speech) tagging
  - "Do you want fries with that?"
  - <verb pron verb noun prep pron>

  - x_i – sequence of words
  - y_i – sequence of part of speech

Sequential Labeling

- Example: Web Information Extraction

<dl><dt><b>Srinivasan Seshan</b> (Carnegie Mellon University) <dt><a href=…><i>Making Virtual Worlds Real</i></a><dt>Tuesday, June 4, 2002<dd>2:00 PM , 322 Sieg<dd>Research Seminar

•* * name name * * affiliation affiliation affiliation * * * * title title title title * * * date date date date * time time * location location * event-type event-type

x_i – sequence of tokens

y_i – sequence of field labels {name, …}

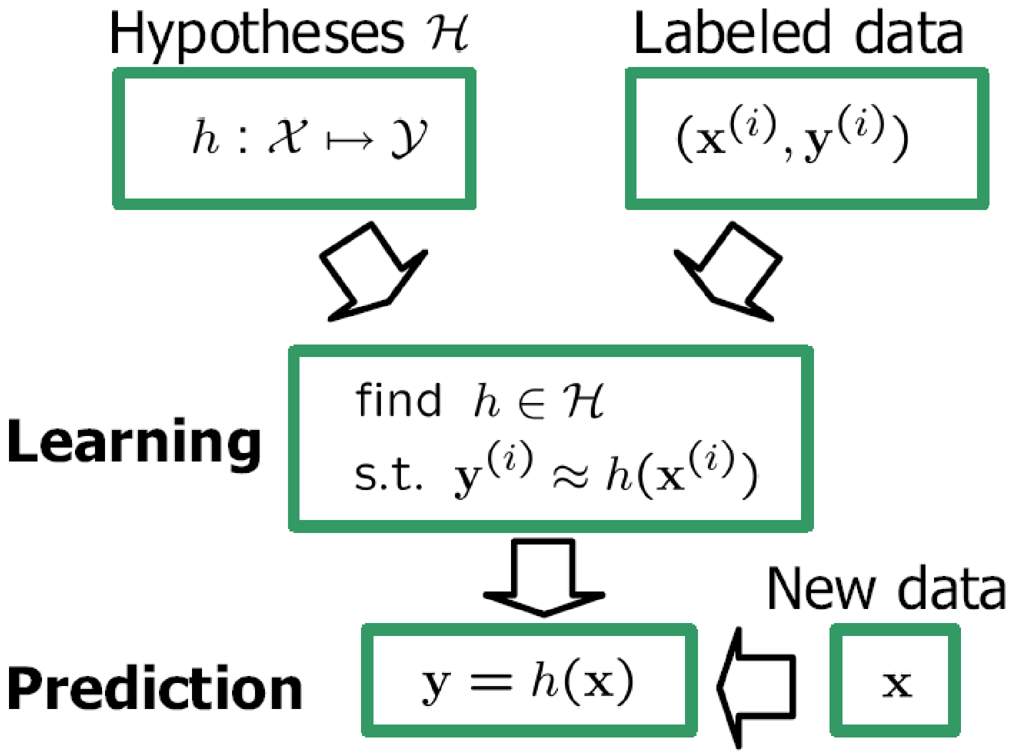
Two kinds of Relationships

- Relationships between the x_t and y_t
  - Example: “Friday” is usually a “date”

- How about the relationships among the y_t
  - Example: “name” is usually followed by “affiliation”

- *Classical models fail to consider the second kind of relationships.*

Sequential Supervised Learning (SSL)


Hypotheses $\mathcal{H}$
$h : \mathcal{X} \mapsto \mathcal{Y}$
Labeled data
$(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$
Learning
find $h \in \mathcal{H}$
s.t. $\mathbf{y}^{(i)} \approx h(\mathbf{x}^{(i)})$
New data
Prediction
$\mathbf{y} = h(\mathbf{x})$
$\mathbf{x}$

$$\mathrm{x}^{(i)} = \left\langle \mathrm{x}_1^{(i)}, \mathrm{x}_2^{(i)}, \cdots, \mathrm{x}_{T_i}^{(i)} \right\rangle$$

$$\mathrm{y}^{(i)} = \left\langle \mathrm{y}_1^{(i)}, \mathrm{y}_2^{(i)}, \cdots, \mathrm{y}_{T_i}^{(i)} \right\rangle$$

$$\mathrm{x}_j^{(i)} = \left[x_{j1}^{(i)}, x_{j2}^{(i)}, \cdots, x_{jd}^{(i)} \right]^T$$

$$\mathrm{y}_j^{(i)} \in \left\{ c_1, c_2, \cdots, c_M \right\}$$

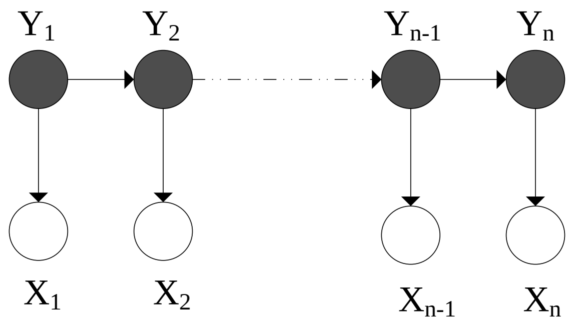
Supervised Setting (figure from Taskar'05)

Graphical Models for SSL

- Hidden Markov Models
- Maximum Entropy Markov Models
- Conditional Random Fields
- Structural SVMs / Max-margin Markov Networks
- Maximum Entropy Discrimination Markov Networks
- …

Hidden Markov Models

- Define a joint probability of paired observation and label sequences



$$p(\mathrm{x},\mathrm{y}) = p(\mathrm{y})\,p(\mathrm{x}\,|\,\mathrm{y})$$
$$= \left(\prod_{i=1:n} p(y_i\,|\,y_{i-1})\right)\left(\prod_{i=1:n} p(x_i\,|\,y_i)\right)$$

- y_t form a Markov chain
- x_t are generated independently (as in naÏve Bayes or Gaussian classifiers).

Hidden Markov Models

- Learning:
  - MLE
    - Count and divide for complete case
    - EM for incomplete case, **forward-backward algorithm** to compute marginal probabilities
  - Discriminative Learning
    - Voted Perceptron
- Labeling
  - Given an observation sequence, choose label sequence s.t.

$$y^* = \arg\max_{y} p(x, y)$$

  - Viterbi algorithm (i.e., max-product)

Hidden Markov Models

- Models both the x_t and y_t relationships and the y_t and y_{t+1} relationships.

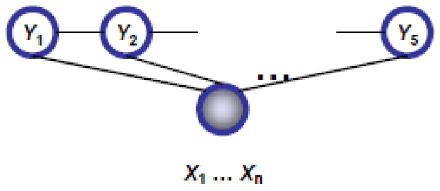
- Does not handle **long-distance dependency**
  - Everything must be captured by the current label y_t .
  - Photograph: /ˈfəʊtəˌɡrɑːf/; Photography: /fəˈtɒɡrəfɪ/

- Does not permit rich y_t and x relationships
  - We can't use several x_t to predict y_t .
  - For computational tractability, strong independence assumption about the observations

Recall Definition of CRFs

- If the graph $G = (V, E)$ of $\mathbf{Y}$ is a tree, the conditional distribution over the label sequence $\mathbf{Y} = \mathbf{y}$, given $\mathbf{X} = \mathbf{x}$, by the Hammersley Clifford theorem of random fields is:

$$p_\theta(\mathrm{y} \mid \mathrm{x}) \propto \exp\left(\sum_{e \in E, k} \lambda_k f_k(e, \mathrm{y}|_e, \mathrm{x}) + \sum_{v \in V, k} \mu_k g_k(v, \mathrm{y}|_v, \mathrm{x}) \right)$$



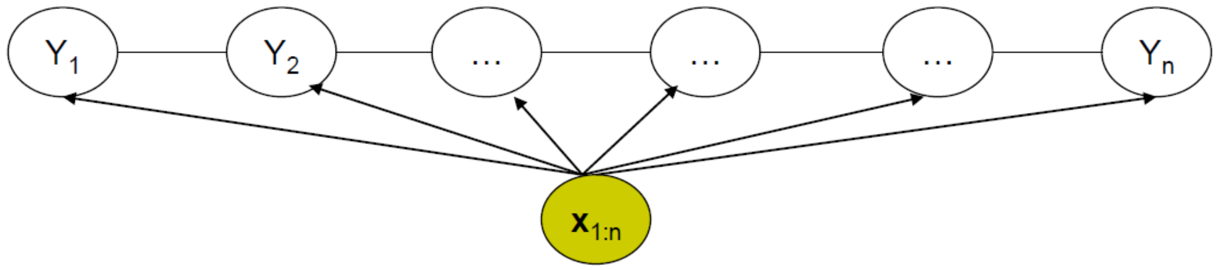
  - x is a data sequence
  - y is a label sequence
  - *v* is a vertex from vertex set V = set of label random variables
  - *e* is an edge from edge set E over V
  - f_k and g_k are given and fixed. g_k is a Boolean vertex feature; f_k is a Boolean edge feature
  - *k* is the number of features
  - $\theta = (\lambda_1, \lambda_2, \cdots, \lambda_n; \mu_1, \mu_2, \cdots, \mu_n); \lambda_k$ and μ_k are parameters to be estimated
  - $\mathrm{y}|_e$ is the set of components of y defined by edge *e*
  - $\mathrm{y}|_v$ is the set of components of y defined by vertex *v*

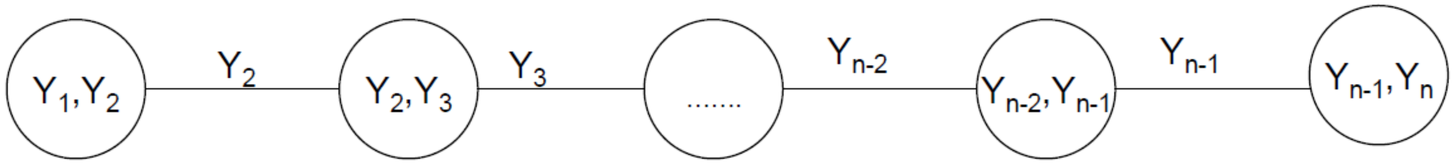
CRFs: Inference

- Given CRF parameters (λ, μ), find the that maximizes p(y|x):

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \exp\left(\sum_{i=1}^{n} (\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\right)$$

  - Can ignore Z(x) because it is not a function of y.
- Run the max-product algorithm on the junction-tree of CRF:





CRF Learning

- Given $\{(\mathbf{x}_d, \mathbf{y}_d)\}_{d=1}^{N}$, find λ^*, μ^* such that

$$\begin{aligned}
\lambda*, \mu* &= \arg\max_{\lambda,\mu} L(\lambda,\mu) = \arg\max_{\lambda,\mu} \prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \prod_{d=1}^{N} \frac{1}{Z(\mathbf{x}_d, \lambda, \mu)} \exp(\sum_{i=1}^{n} (\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d))) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N} (\sum_{i=1}^{n} (\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d, \lambda, \mu))
\end{aligned}$$

- Computing the gradient w.r.t λ:

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N} (\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}} (P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d)))$$

CRF Learning

$$\nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^{N} \left(\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \boxed{\sum_{\mathbf{y}} \left(P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) \right)} \right)$$

- Computing the model expectations:
  - Requires exponentially large number of summations: Is it intractable?

$$\sum_{\mathbf{y}} \left(P(\mathbf{y}|\mathbf{x}_d) \sum_{i=1}^{n} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) \right) = \sum_{i=1}^{n} \left(\sum_{\mathbf{y}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(\mathbf{y}|\mathbf{x}_d) \right)$$

$$= \sum_{i=1}^{n} \sum_{y_i, y_{i-1}} \boxed{\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1}|\mathbf{x}_d)}$$

Expectation of f over the marginal prob of neighboring nodes!

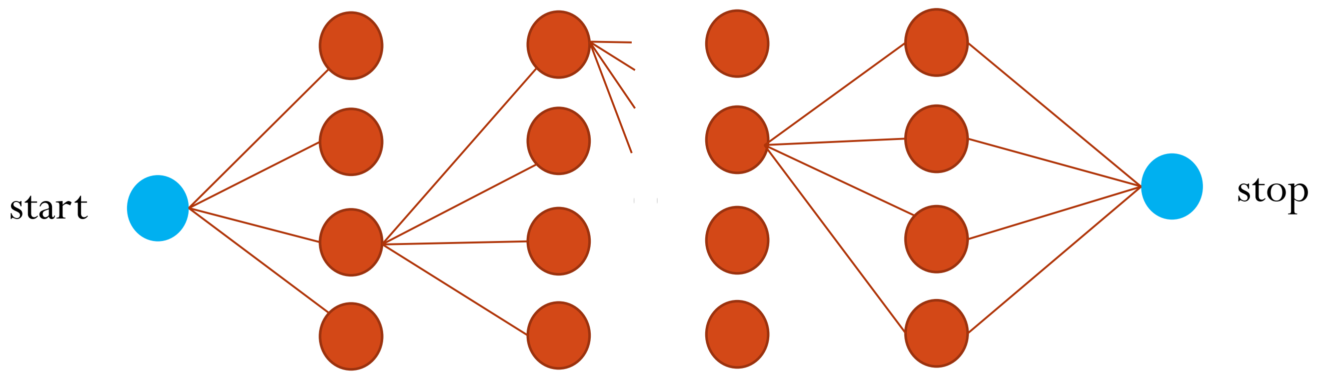
  - Tractable!
    - Can compute marginal using the sum-product algorithm on the chain!

CRF Learning

- Computing marginal forward-backward message passing
  - Represented in matrix form

$$M_i(\mathbf{x}) = [M_i(y_{i-1}, y_i|\mathbf{x})]$$

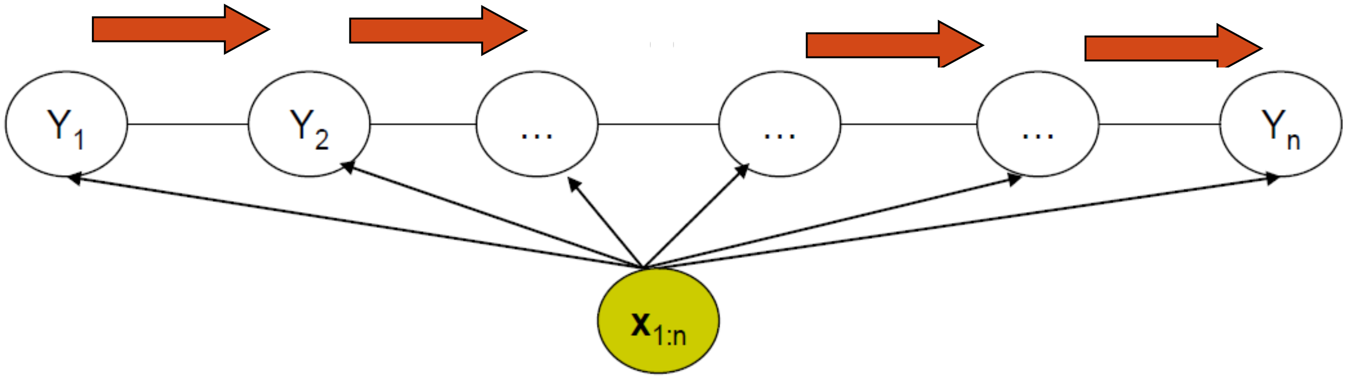
$$M_i(y_{i-1}, y_i|\mathbf{x}) = \exp\left(\lambda^\top \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T(y_i, \mathbf{x})\right)$$



CRF Learning

- Computing marginal forward-backward message passing
  - Forward pass:

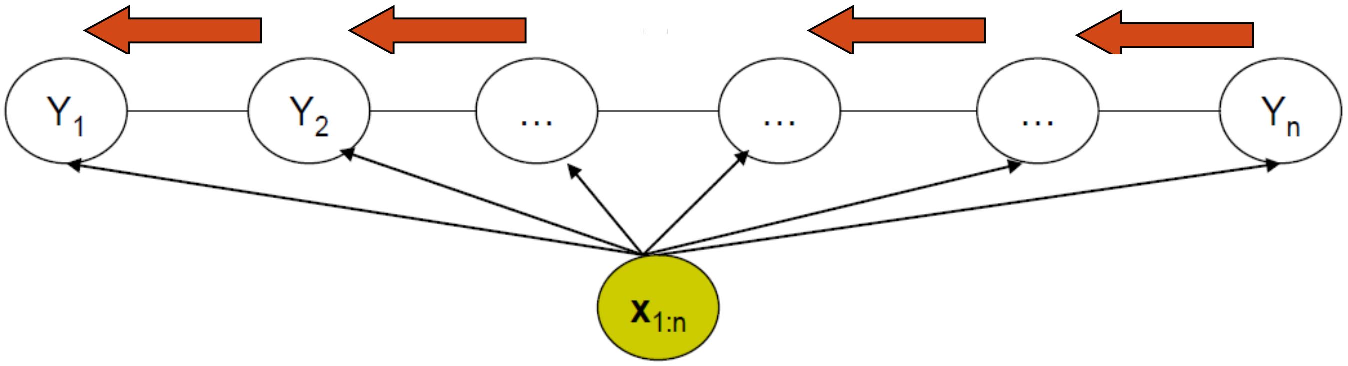
$$\alpha_0(y_0 \mid \mathrm{x}) = \begin{cases} 1 & if\ \ y_0 = start \\ 0 & otherwise \end{cases} \qquad \alpha_t(\mathrm{x}) = \alpha_{t-1}(\mathrm{x}) M_t(\mathrm{x})$$



CRF Learning

- Computing marginal forward-backward message passing
  - Forward pass:

$$\beta_{n+1}(y \mid \mathrm{x}) = \begin{cases} 1 & if\ \ y = stop \\ 0 & otherwise \end{cases} \qquad \beta_t(\mathrm{x})^{\mathrm{T}} = M_{t+1}(\mathrm{x})\beta_{t+1}(\mathrm{x})$$



CRF Learning

- Computing marginal forward-backward message passing
  - Normalization to get marginal probabilities:

Single Variable:

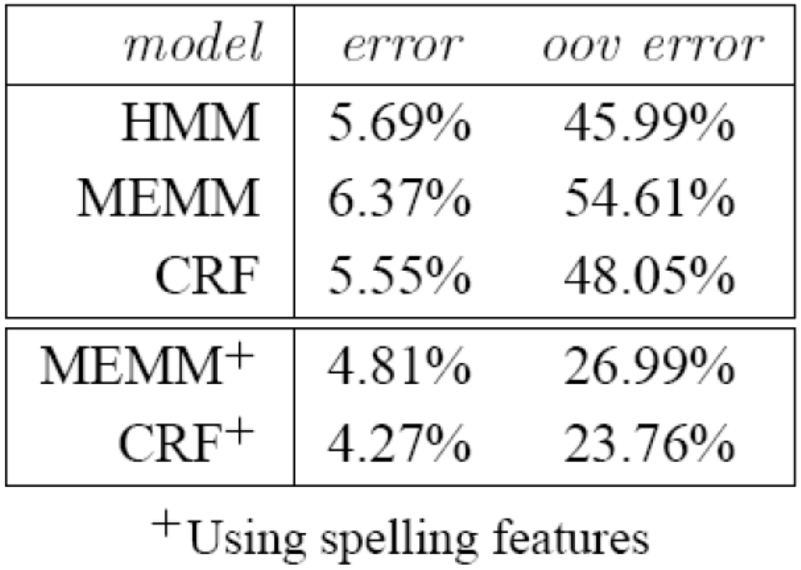
$$p(y_t \mid \mathrm{x}) = \frac{\alpha_t(y_t \mid \mathrm{x})\beta_t(y_t \mid \mathrm{x})}{Z(\mathrm{x})} = \frac{\alpha_t(y_t \mid \mathrm{x})\beta_t(y_t \mid \mathrm{x})}{\alpha_t(\mathrm{x})\beta_t(\mathrm{x})}$$

Neighboring Variables:

$$p(y_{t-1}, y_t \mid \mathrm{x}) = \frac{\alpha_{t-1}(y_{t-1} \mid \mathrm{x})M_t(y_{t-1}, y_t \mid \mathrm{x})\beta_t(y_t \mid \mathrm{x})}{Z(\mathrm{x})} = \frac{\alpha_{t-1}(y_{t-1} \mid \mathrm{x})M_t(y_{t-1}, y_t \mid \mathrm{x})\beta_t(y_t \mid \mathrm{x})}{\alpha_t(\mathrm{x})\beta_t(\mathrm{x})}$$

Some Empirical Results

- Part-of-Speech tagging



model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

- Using same set of features: HMM >=< CRF > MEMM
- Using additional overlapping features: CRF$^+$ > MEMM$^+$ >> HMM

Beyond Linear-Chains

- CRFs can be defined over arbitrary undirected graphs, not limited to sequences
  - Grid-like CRFs
    - 2D CRFs for Web information extraction (Zhu et al., 2005)
  - Tree structured CRFs with/without inner-layer connections
    - Multi-scale CRFs for image labeling
    - HCRFs for simultaneous record detection and labeling (Zhu et al., 2006)
  - Dynamic CRFs (in terms of time)
    - Factorial CRFs for joint POS tagging and chunking
  - Semi-Markov Random Fields
    - Model uncertainty of segment boundary for joint segmentation and labeling

- General mechanisms for specifying templates of graphical structure
  - Relational Markov Networks
  - Markov Logic Networks

Summary

- Parameter learning for Undirected graphical models
- Conditional random fields

References

- Chap. 8 of PRML
- Chap. 17 of ESL (undirected graphical models)