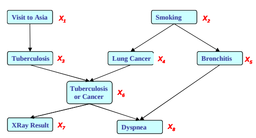
Machine Learning

10-701/15-781, Fall 2006

Graphical Models II
Inference

Eric Xing



Lecture 13, October 26, 2006

Reading: Chap. 8, C.B book

Recap of Basic Prob. Concepts

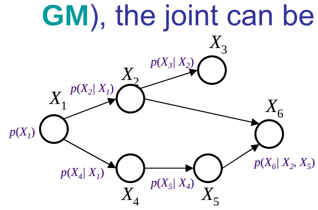
- Joint probability dist. on multiple variables:

$$P(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1, X_2)P(X_4 \mid X_1, X_2, X_3)P(X_5 \mid X_1, X_2, X_3, X_4)P(X_6 \mid X_1, X_2, X_3, X_4, X_5)$$

- If X_i's are **independent**: ($P(X_i|\cdot)$= $P(X_i)$)

$$P(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$= P(X_1)P(X_2)P(X_3)P(X_4)P(X_5)P(X_6) = \prod_i P(X_i)$$

- If X_i's are **conditionally independent** (as described by a **GM**), the joint can be factored to simpler products, e.g.,

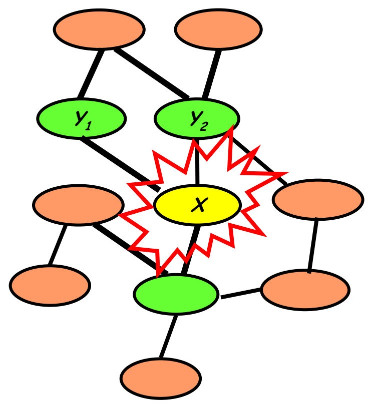


$$P(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$= P(X_1)\, P(X_2| X_1)\, P(X_3| X_2)\, P(X_4| X_1)\, P(X_5| X_4)\, P(X_6| X_2, X_5)$$

Markov Random Fields

Structure: an ***undirected graph***

- Meaning: a node is **conditionally independent** of every other node in the network given its **Directed neighbors**
- Local contingency functions (**potentials**) and the **cliques** in the graph completely determine the **joint** dist.
- Give **correlations** between variables, but no explicit way to generate samples



Representation

- Defn: an undirected graphical model represents a distribution $P(X_1, \ldots, X_n)$ defined by an undirected graph H, and a set of positive ***potential functions*** y_c associated with cliques of H, s.t.

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

where Z is known as the partition function:

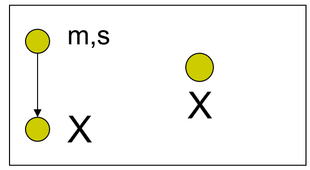
$$Z = \sum_{x_1, \ldots, x_n} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

- Also known as Markov Random Fields, Markov networks …
- The ***potential function*** can be understood as an contingency function of its arguments assigning "pre-probabilistic" score of their joint configuration.

GMs are your old friends

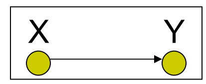
Density estimation

Parametric and nonparametric methods



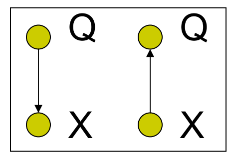
Regression

Linear, conditional mixture, nonparametric

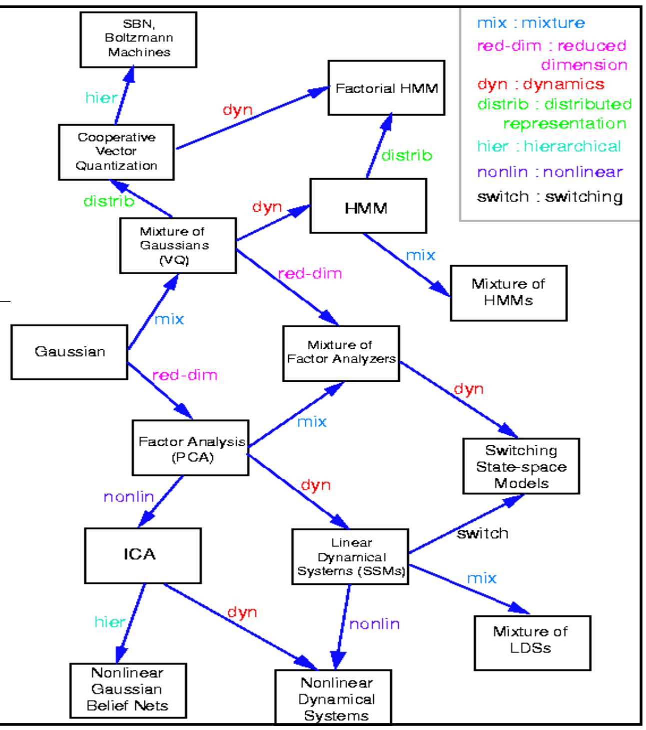


Classification

Generative and discriminative approach



An (incomplete) genealogy of graphical models



(Picture by Zoubin Ghahramani and Sam Roweis)

Probabilistic Inference

- We now have compact representations of probability distributions: **Graphical Models**
- A GM *M* describes a unique probability distribution *P*
- How do we answer **queries** about *P*?
- We use **inference** as a name for the process of computing answers to such queries

Eric Xing 7

Query 1: Likelihood

- Most of the queries one may ask involve **evidence**
  - Evidence ***e*** is an assignment of values to a set ***E*** variables in the domain
  - Without loss of generality $\mathbf{E} = \{ X_{k+1}, \dots, X_n \}$
- Simplest query: compute probability of evidence

$$P(e) = \sum_{x_1} \dots \sum_{x_k} P(x_1, \dots, x_k, e)$$

  - this is often referred to as computing the **likelihood** of ***e***

Eric Xing 8

Query 2: Conditional Probability

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

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

  - this is the ***a posteriori* belief** in X, given evidence e

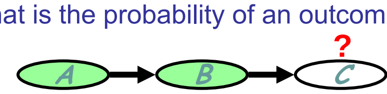
- We usually query a subset Y of all domain variables $X=\{Y,Z\}$ and "don't care" about the remaining, Z:

$$P(Y \mid e) = \sum_z P(Y, Z = z \mid e)$$

  - the process of summing out the "don't care" variables z is called **marginalization**, and the resulting $P(y|e)$ is called a **marginal** prob.

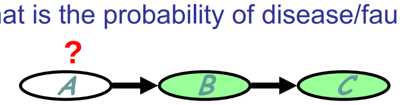
Applications of *a posteriori* Belief

- **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 probability of disease/fault given symptoms



  - the query node an ancestor of the evidence

- **Learning** under partial observation
  - fill in the unobserved values under an "EM" setting (more later)

- The directionality of information flow between variables is not restricted by the directionality of the edges in a GM
  - probabilistic inference can combine evidence form all parts of the network

Query 3: Most Probable Assignment

- In this query we want to find the **most probable joint assignment** (MPA) for ***some*** variables of interest
- Such reasoning is usually performed under some given evidence ***e***, and ignoring (the values of) other variables ***z*** :

$$\mathrm{MPA}(Y \mid e) = \arg\max_y P(y \mid e) = \arg\max_y \sum_z P(y, z \mid e)$$

  - this is the **maximum *a posteriori*** configuration of ***y***.

Eric Xing 11

Applications of MPA

- Classification
  - find most likely label, given the evidence
- 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 *X*?
  - MPA of *(X, Y)*?

x	y	P(x,y)
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Eric Xing 12

Complexity of Inference

Thm:

Computing $P(X = x \mid e)$ in a GM is NP-hard

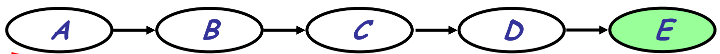
- Hardness does not mean we cannot solve inference
  - It implies that we cannot find a general procedure that works efficiently for arbitrary GMs
  - For particular families of GMs, we can have provably efficient procedures

Approaches to inference

- Exact inference algorithms
  - The elimination algorithm √
  - The junction tree algorithms √ (but will not cover in detail here)
- Approximate inference techniques
  - Stochastic simulation / sampling methods √
  - Markov chain Monte Carlo methods √
  - Variational algorithms (will be covered in advanced ML courses)

Marginalization and Elimination

- A signal transduction pathway:



What is the likelihood that protein E is active?

- Query: $P(e)$

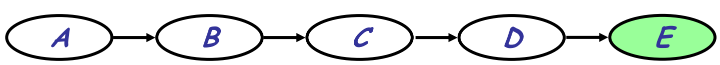
$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a,b,c,d,e)$$

a naïve summation needs to enumerate over an exponential number of terms

- By chain decomposition, we get

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

Elimination on Chains

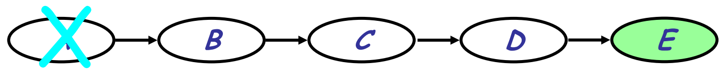


- Rearranging terms ...

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

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

Elimination on Chains

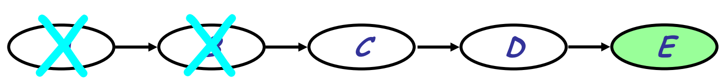


- Now we can perform innermost summation

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

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

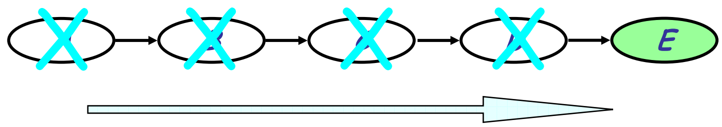
Elimination in Chains



- Rearranging and then summing again, we get

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

Elimination in 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 $O(|Val(X_i)|*|Val(X_{i+1})|)$ operations: $O(kn^2)$
  - Compare to naïve evaluation that sums over joint values of n-1 variables $O(n^k)$

Inference on General GM via Variable Elimination

General idea:

- Write query in the form

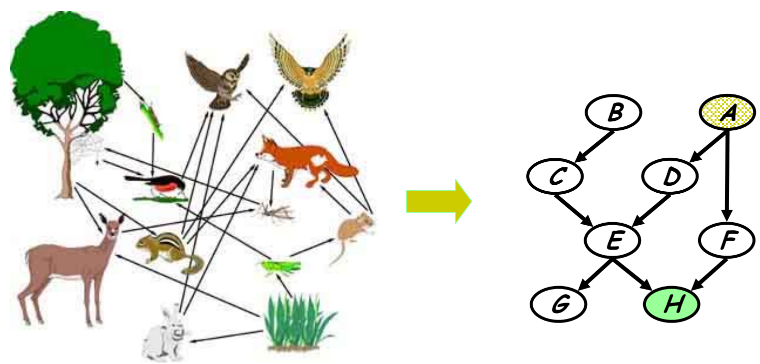
$$P(X_1, \boldsymbol{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

  - this suggests an "elimination order" of latent variables to be marginalized
- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product
- wrap-up

$$P(X_1 \mid \boldsymbol{e}) = \frac{P(X_1, \boldsymbol{e})}{P(\boldsymbol{e})}$$

A more complex network

A food web



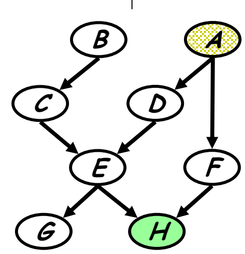
What is the probability that hawks are leaving given that the grass condition is poor?

Example: Variable Elimination

- Query: $P(A \mid h)$
  - 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



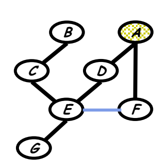
A regulatory network

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

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

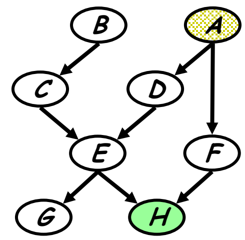
  - This step is isomorphic to a marginalization step:

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



Example: Variable Elimination

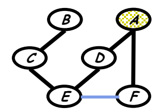
- Query: $P(B \mid h)$
  - Need to eliminate: B,C,D,E,F,G
- Initial factors:



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

- Step 2: Eliminate G
  - compute

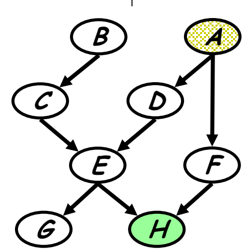
$$m_g(e) = \sum_g p(g \mid e) = 1$$



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

Example: Variable Elimination

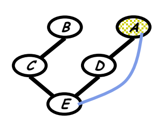
- Query: $P(B \mid h)$
  - Need to eliminate: B,C,D,E,F
- Initial factors:



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

- Step 3: Eliminate F
  - compute

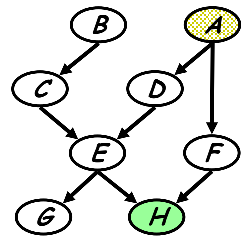
$$m_f(e,a) = \sum_f p(f \mid a)m_h(e,f)$$



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

Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B,C,D,E*

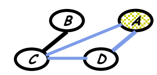


- Initial factors:

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

- Step 4: Eliminate *E*
  - compute

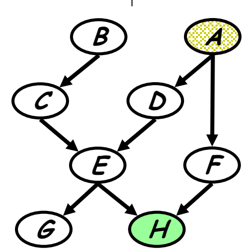
$$m_e(a,c,d) = \sum_e p(e|c,d)m_f(a,e)$$



$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{m_e(a,c,d)}$$

Example: Variable Elimination

- Query: *P(B | h)*
  - Need to eliminate: *B,C,D*

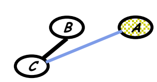


- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|b)\underline{P(d|a)m_e(a,c,d)}$$

- Step 5: Eliminate *D*
  - compute

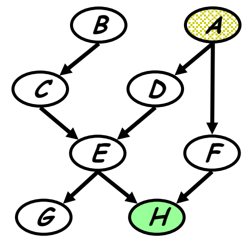
$$m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$$



$$\Rightarrow P(a)P(b)P(c|d)\underline{m_d(a,c)}$$

Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: B, C
- Initial factors:



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

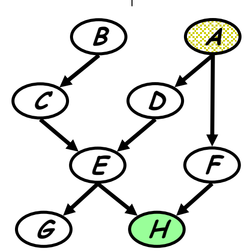
- Step 6: Eliminate C

  - compute $$m_c(a,b) = \sum_c p(c \mid b)m_d(a,c)$$

$$\Rightarrow P(a)P(b)\underline{P(c \mid d)m_d(a,c)}$$

Example: Variable Elimination

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



$$P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)P(e \mid c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c \mid d)P(d \mid a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c \mid d)m_d(a,c)$$
$$\Rightarrow P(a)\underline{P(b)m_c(a,b)}$$

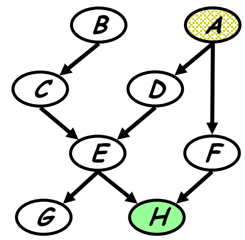
- Step 7: Eliminate B

  - compute $$m_b(a) = \sum_b p(b)m_c(a,b)$$

$$\Rightarrow P(a)\underline{m_b(a)}$$

Example: Variable Elimination

- Query: $P(B\,|\,h)$
  - Need to eliminate: B



- Initial factors:

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

- Step 8: Wrap-up $\quad p(a,\tilde{h}) = p(a)m_b(a)\,, \quad p(\tilde{h}) = \sum_a p(a)m_b(a)$

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

Complexity of variable elimination

- Suppose in one elimination step we compute

$$m_x(y_1,\ldots,y_k) = \sum_x m'_x(x,y_1,\ldots,y_k)$$
$$m'_x(x,y_1,\ldots,y_k) = \prod_{i=1}^{k} m_i(x,\mathbf{y}_{c_i})$$

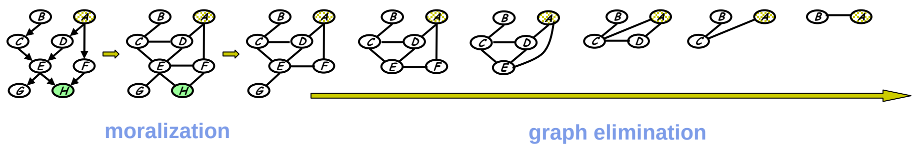
This requires

- $k \bullet |\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ multiplications
  - For each value of $x, y_1, \ldots, y_k$, we do k multiplications
- $|\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ additions
  - For each value of $y_1, \ldots, y_k$, we do $|Val(X)|$ additions

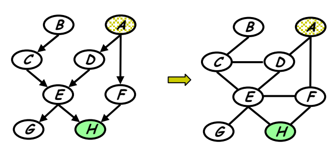
Complexity is **exponential** in number of variables in the intermediate factor

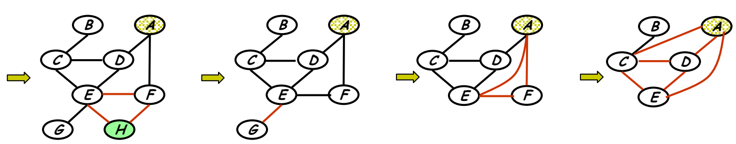
Understanding Variable Elimination

- A graph elimination algorithm

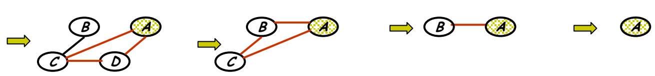


Elimination Cliques





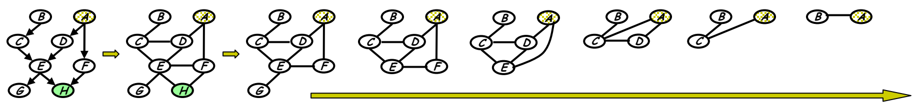
$m_h(e,f)$ $m_g(e)$ $m_f(e,a)$ $m_e(a,c,d)$



$m_d(a,c)$ $m_c(a,b)$ $m_b(a)$

Understanding Variable Elimination

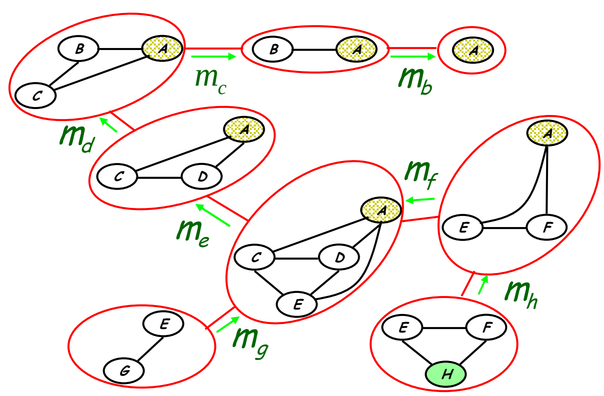
- A graph elimination algorithm



moralization graph elimination

- Intermediate terms correspond to the cliques resulted from elimination
  - "good" elimination orderings lead to **small cliques** and hence reduce complexity
    (what will happen if we eliminate "e" first in the above graph?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found
- Applies to undirected GMs

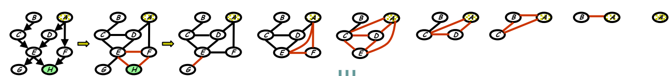
A clique tree



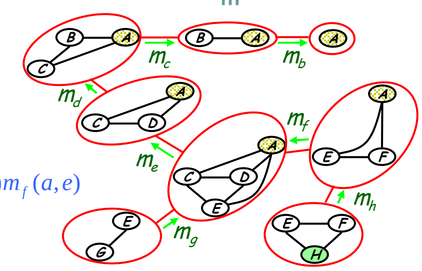
$$m_e(a,c,d) = \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

From Elimination to Message Passing

- Our algorithm so far 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**



≡

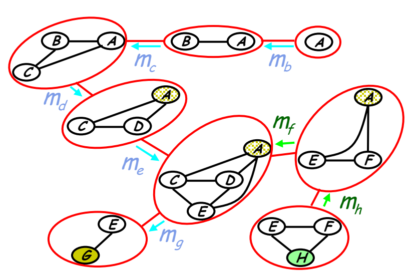


$$m_e(a,c,d) = \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

- Messages can be reused

From Elimination to Message Passing

- Our algorithm so far 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

A Sketch of the Junction Tree Algorithm

- **The algorithm**
  - Construction of junction trees --- a special **clique tree**
  - Propagation of probabilities --- a message-passing protocol
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT
- Many well-known algorithms are special cases of JT
  - Forward-backward, Kalman filter, Peeling, Sum-Product ...

Approaches to inference

- Exact inference algorithms
  - The elimination algorithm √
  - The junction tree algorithms √ (but will not cover in detail here)
- Approximate inference techniques
  - Stochastic simulation / sampling methods √
  - Markov chain Monte Carlo methods √
  - Variational algorithms (later lectures)

Monte Carlo methods

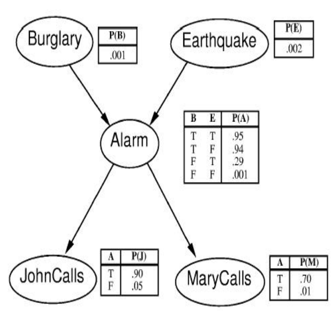
- Draw random samples from the desired distribution
- Yield a stochastic representation of a complex distribution
  - marginals and other expections can be approximated using sample-based averages

$$E[f(x)] = \frac{1}{N}\sum_{t=1}^{N} f(x^{(t)})$$

- **Asymptotically** exact and easy to apply to arbitrary models
- Challenges:
  - how to draw samples from a given dist. (not all distributions can be trivially sampled)?
  - how to make better use of the samples (not all sample are useful, or eqally useful, see an example later)?
  - how to know we've sampled enough?

Example: naive sampling

- Sampling: Construct samples according to probabilities given in a BN.



E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E1	B0	A1	M1	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0

Alarm example: (Choose the right sampling sequence)
1) Sampling:P(B)=<0.001, 0.999> suppose it is false, B0. Same for E0. P(A|B0, E0)=<0.001, 0.999> suppose it is false...
2) Frequency counting: In the samples right,
P(J|A0)=P(J,A0)/P(A0)=<1/9, 8/9>.

Example: naive sampling

- Sampling: Construct samples according to probabilities given in a BN.

Alarm example: (Choose the right sampling sequence)

3) what if we want to compute P(J|A1) ?
we have only one sample ...
P(J|A1)=P(J,A1)/P(A1)=<0, 1>.

4) what if we want to compute P(J|B1) ?
No such sample available!
P(J|A1)=P(J,B1)/P(B1) can not be defined.

For a model with hundreds or more variables, rare events will be very hard to garner evough samples even after a long time or sampling ...

E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E1	B0	A1	M1	J1
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0
E0	B0	A0	M0	J0

Monte Carlo methods (cond.)

- Direct Sampling
  - We have seen it.
  - Very difficult to populate a high-dimensional state space
- Rejection Sampling
  - Create samples like direct sampling, only count samples which is consistent with given evidences.
- ....
- Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo

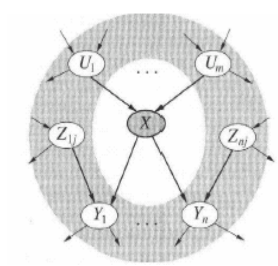
- Samples are obtained from a **Markov chain** (of sequentially evolving distributions) whose **stationary distribution** is the desired $p(x)$
- Gibbs sampling
  - we have variable set to $\mathbf{X}=\{x_1, x_2, x_3, \dots x_N\}$
  - at each step one of the variables X_i is selected (at random or according to some fixed sequences)
  - the conditonal distribution $p(X_i | \mathbf{X}_{-i})$ is computed
  - a value x_i is sampled from this distribution
  - the sample x_i replaces the previous of X_i in $\mathbf{X}$.

Eric Xing 43

MCMC

- Markov-Blanket
  - A variable is independent from others, given its parents, children and children's parents. d-separation.

    $\Rightarrow p(X_i | \mathbf{X}_{-i}) = p(X_i | \text{MB}(X_i))$

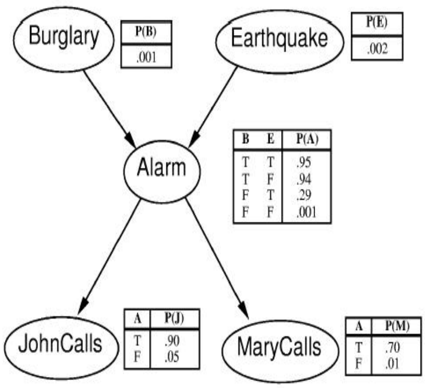


- Gibbs sampling
  - Create a random sample. Every step, choose one variable and sample it by P(X|MB(X)) based on previous sample.

MB(A)={B, E, J, M}
MB(E)={A, B}

Eric Xing 44

MCMC



- To calculate P(J|B1,M1)
- Choose (B1,E0,A1,M1,J1) as a start
- **Evidences** are B1, M1, **variables** are A, E, J.
- Choose next variable as A
- Sample A by P(A|MB(A))=P(A|B1, E0, M1, J1) suppose to be false.
- (B1, E0, A0, M1, J1)
- Choose next random variable as E, sample E~P(E|B1,A0)
- ...

Complexity for Approximate Inference

- Approximate Inference will not reach the exact probability distribution in finite time, but only close to the value.
- Often much faster than exact inference when BN is big and complex enough. In MCMC, only consider P(X|MB(X)) but not the whole network.