Elimination Algorithm

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, IN, USA

Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

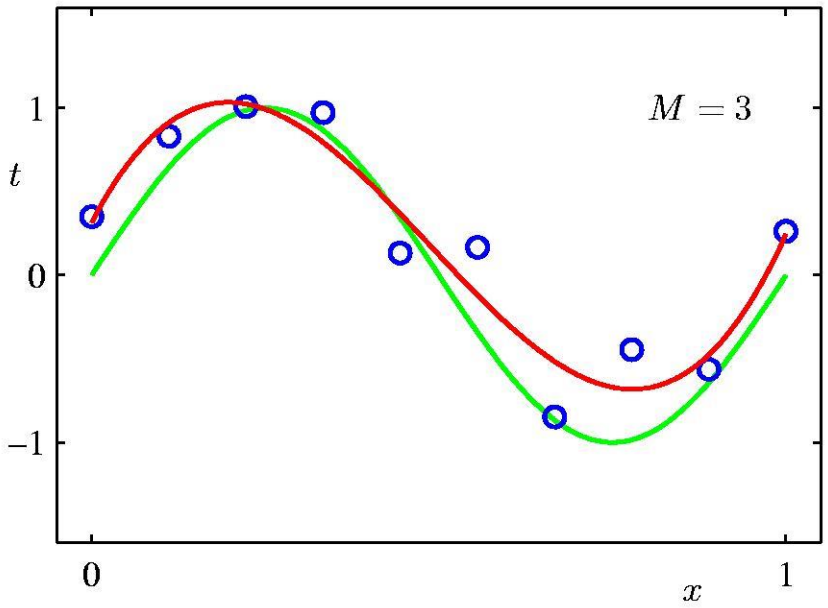
January 30, 2018

Contents

- Plate Notation: Polynomial Regression, Relevance Vector Machine, State Space Models, Learning from Complete Data
- Probabilistic Inference in General Graphs, Distributive Law, Performing Summations
- Clamped Variables, Inner Summations and New Tables
- Evidence Potentials
- Elinination Algorithm for Directed Graphs, Elimination Algorithm for Undirected Graphs
- Graph Elimination, Graph Elimination in Undirected Graphs, Induced Dependencies, Graph Elimination in Directed Graphs
- Computational Complexity, Elimination in Trees, Treewidth, Inference in Undirected Graphs

- Kevin Murphy, Machine Learning: A probabilistic Perspective, Chapter 20
- Chris Bishop, Pattern Recognition and Machine Learning, Chapter 8
- Jordan, M. I. (2007). An introduction to probabilistic graphical models. In preparation (Chapter 3).
- Video Lectures on Machine Learning, Z. Gahramani, C. Bishop and others.

Example: Polynomial Regression



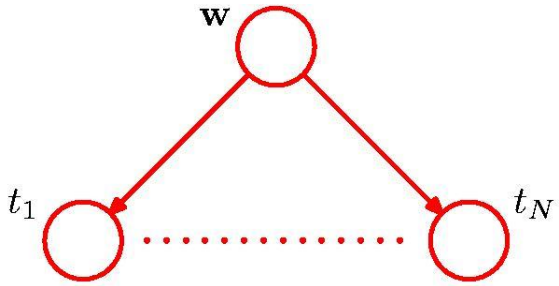
Polynomial regression

$$t = y(x, w) = \sum_{i=0}^{M} w_i x^i$$

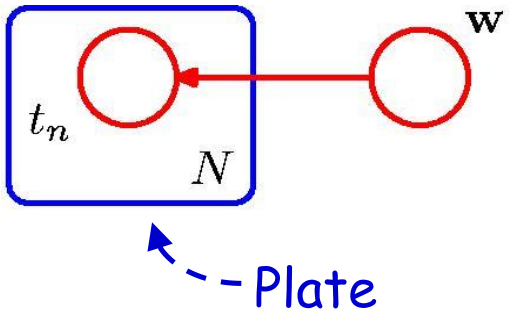
$$p(t, w) = p(w) \prod_{n=1}^{N} p(t_n | w)$$

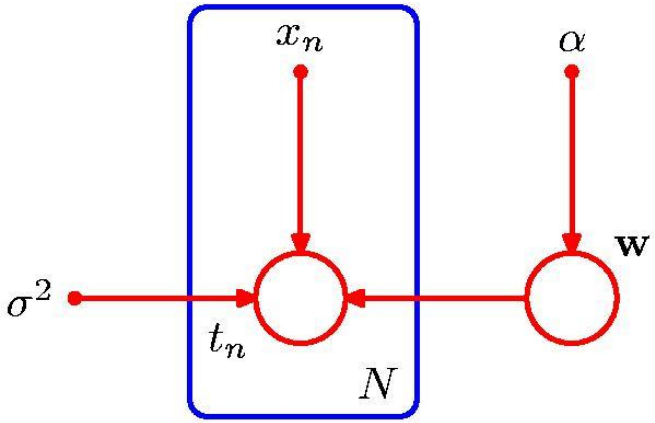
Example: Polynomial Regression

- Graphical model for the polynomial regression (left). A more compact graphical representation using plate for the polynomial regression (right)



$$p(\boldsymbol{t},\boldsymbol{w}) = p(\boldsymbol{w})\prod_{n=1}^{N} p(t_n|\boldsymbol{w})$$





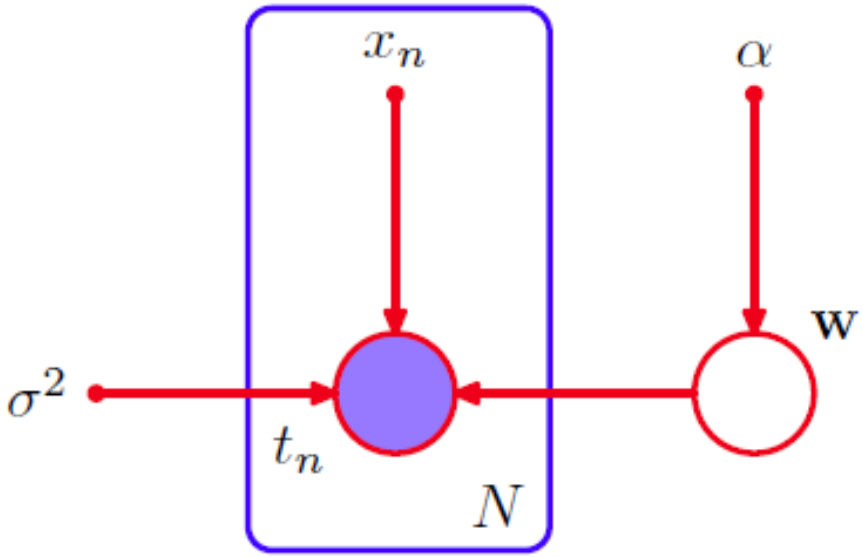
Express the model in an explicit form:

$$p\left(\boldsymbol{t},\boldsymbol{w}|\boldsymbol{x},\alpha,\sigma^2\right) = p\left(\boldsymbol{w}|\alpha\right)\prod_{n=1}^{N} p\left(t_n|\boldsymbol{w},x_n,\sigma^2\right)$$

Example: Polynomial Regression

- Evidence nodes are shaded (observed values). For our regression problem this is shown as follows:

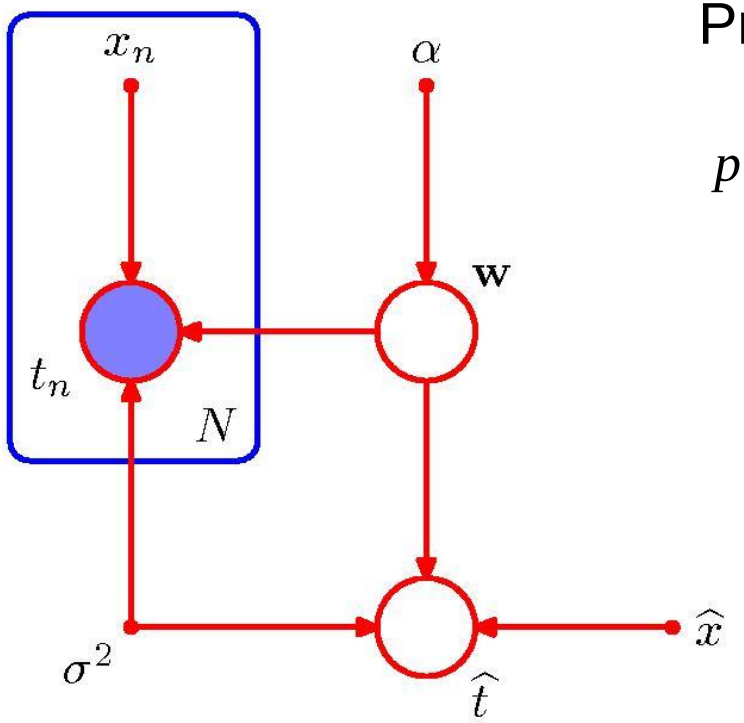
- ***w*** is an example of a latent variable (unobserved)



$$p\left(\boldsymbol{t}, \boldsymbol{w} \middle| \boldsymbol{x}, \alpha, \sigma^2\right) = p\left(\boldsymbol{w} \middle| \alpha\right) \prod_{n=1}^{N} p\left(t_n \middle| \boldsymbol{w}, x_n, \sigma^2\right)$$

Example: Polynomial Regression

- In Bayesian regression, our goal is to make predictions for new input values.



Predictive distribution:

$$p\left(\hat{t}\middle|x,\mathbf{x},\mathbf{t},\alpha,\sigma^2\right)\propto\int p\left(\hat{t},\mathbf{t},\mathbf{w}\middle|x,\mathbf{x},\alpha,\sigma^2\right)d\mathbf{w}$$

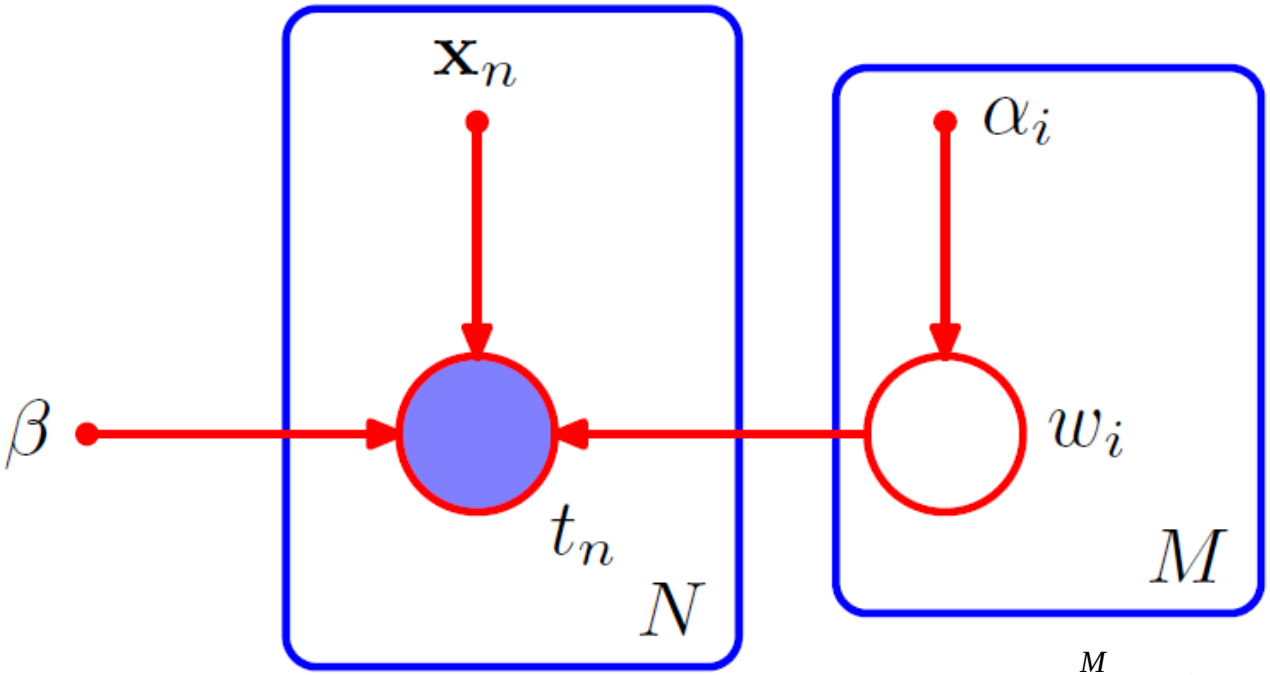
where

$$p\left(\hat{t},\mathbf{t},\mathbf{w}\middle|x,\mathbf{x},\alpha,\sigma^2\right)=$$

$$\left[\prod_{n=1}^{N}p\left(t_n\middle|\mathbf{w},x_n,\sigma^2\right)\right]p\left(\mathbf{w}\middle|\alpha\right)p\left(\hat{t}\middle|x,\mathbf{w},\sigma^2\right)$$

Example: Relevance Vector Machine

- Recall the RVM (relevance vector machine) framework for regression.
- The graphical model is shown below.

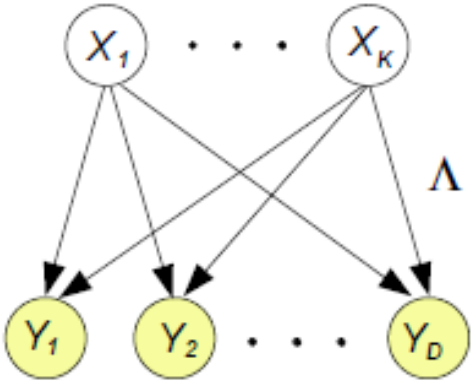


$$p\left(\boldsymbol{t}\middle|\boldsymbol{X},\boldsymbol{w},\beta\right)=\prod_{n=1}^{N}p\left(t_n\mid\boldsymbol{x}_n,\boldsymbol{w},\beta\right)$$

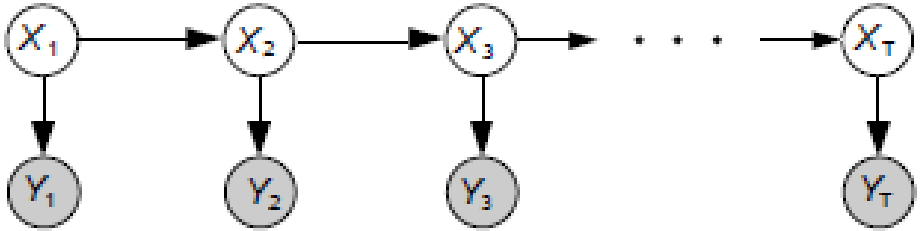
$$p\left(\boldsymbol{w}\middle|\boldsymbol{\alpha}\right)=\prod_{n=1}^{M}p\left(w_i\middle|0,\alpha_i^{-1}\right)$$

State Space Models

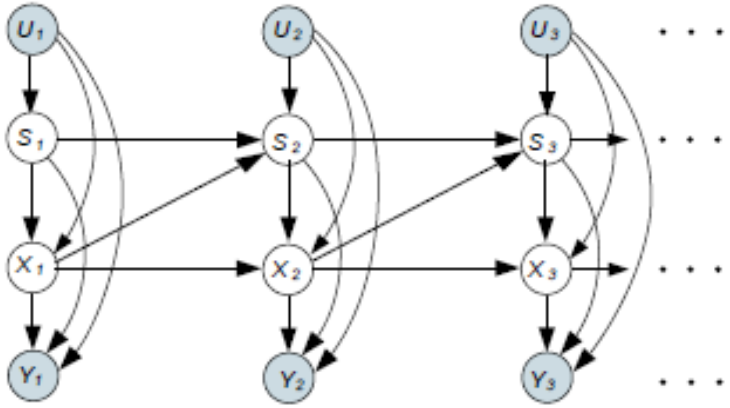
- Probabilistic Principal Component Analysis (PCA)



- Hidden Markov Models

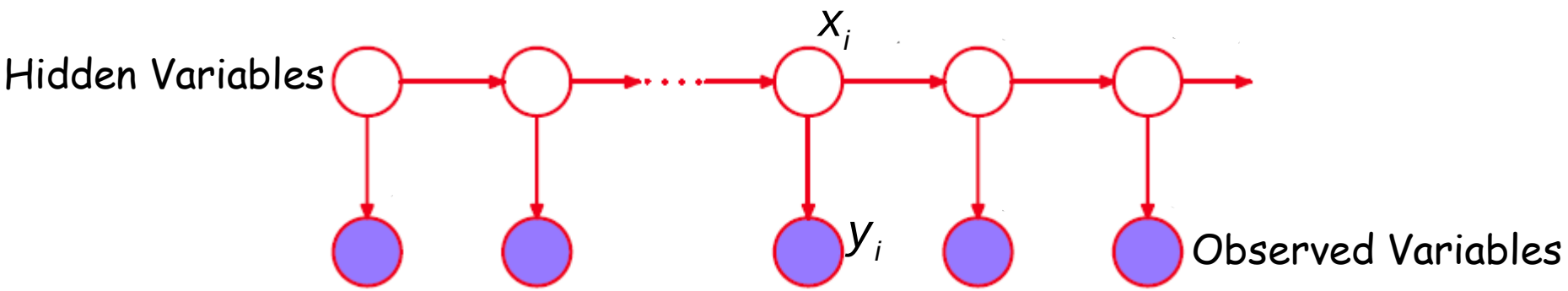


- Switching State-Space Models



Example in State Space Models

- Consider a Hidden Markov Model (discrete states) or a Gaussian Filter (linear Gaussian Model)

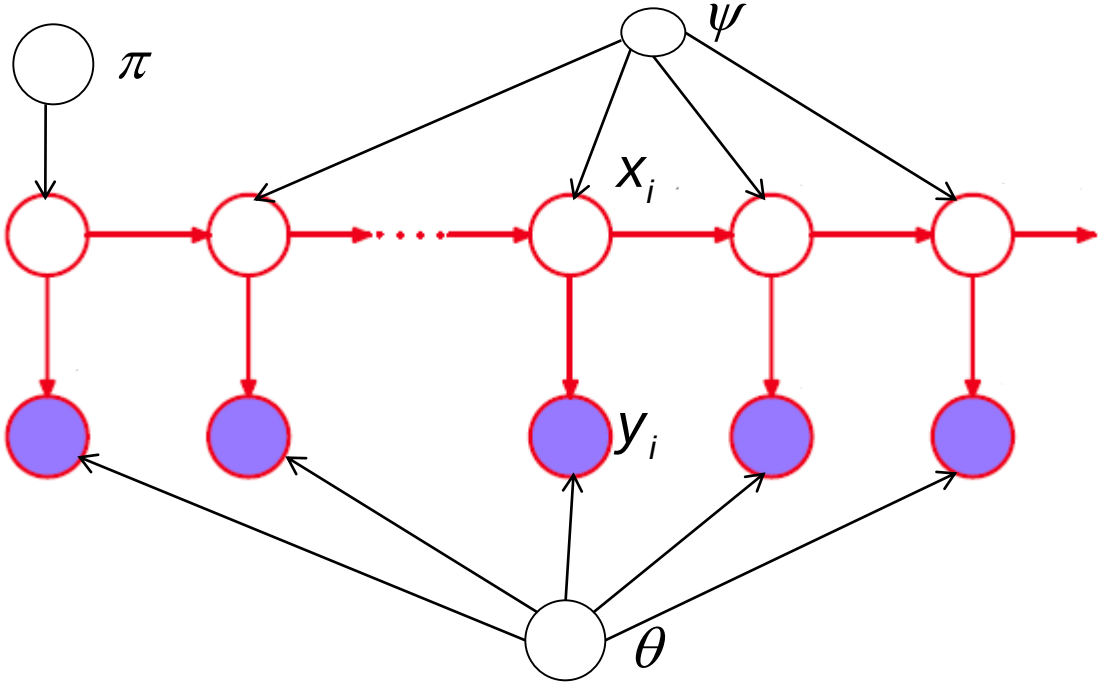


$$p(\boldsymbol{x},\boldsymbol{y}) = p(x_1)p(y_1 \mid x_1)p(x_2 \mid x_1)....p(x_i \mid x_{i-1})p(y_i \mid x_i)....$$

- For linear Gaussian model for the conditional distributions, i.e. $p(x \mid z) = \mathcal{N}\left(Az + b, \Sigma\right)$ the model is a Kalman Filter. In this case, the joint distribution over all variables is also a highly structured Gaussian.

Example: Bayesian State Space Models

- Introduce a prior for x_1, and parametric models (parameters θ and ψ - each with its own prior – are the same for all models):

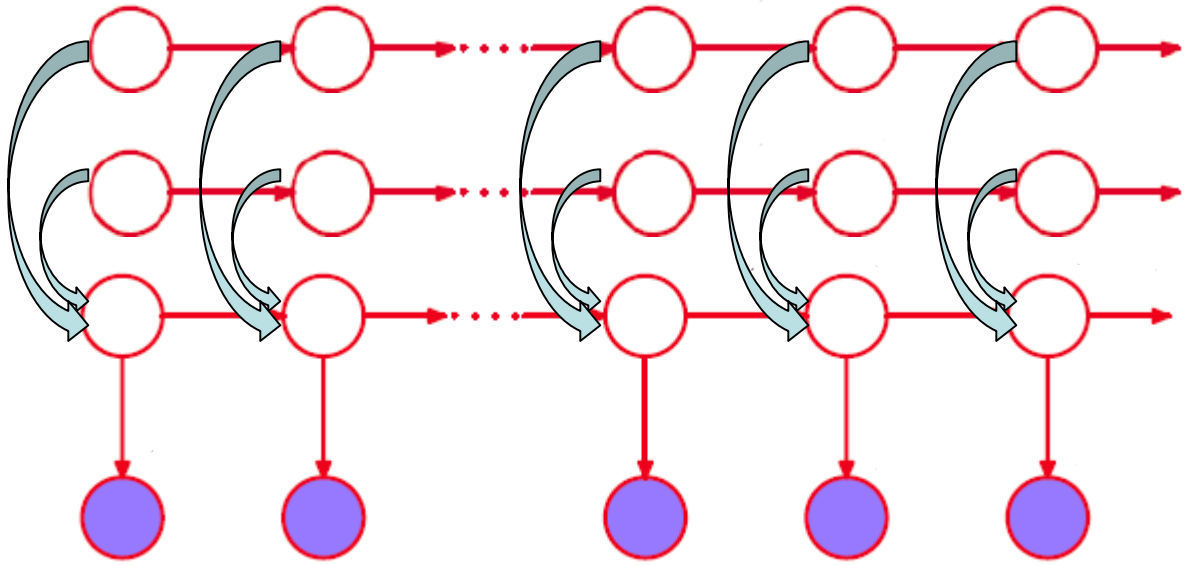


$$....p(x_i \mid x_{i-1}; \psi)p(y_i \mid x_i; \theta)....$$

- *In this graph we have loops (not like the tree structure of the HMM model shown earlier). This makes approximate (rather than exact) inference the only option.*

Example: Factorial State Space Model

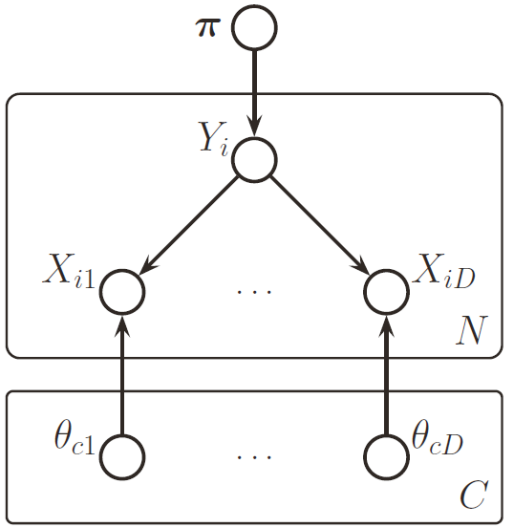
- Consider multiple Hidden Sequences (if each state has m configurations, then a total of m^3 configurations at each time step)

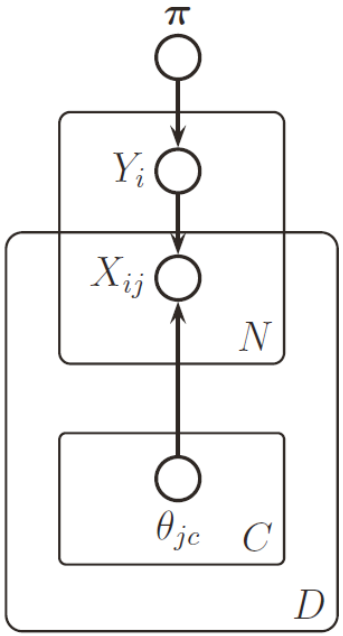


- *This model is more tractable than a single hidden chain with m^3 configurations at each time.*

Context Specific Independence

- On the right a naive Bayes classifier has been unrolled for D features and uses a plate notation over data $i = 1 : N$. The Fig. on the bottom right shows a *nested plate* notation for the same model.
- A variable is inside two plates has two sub-indices (e.g. θ_{jc} is the parameter for feature j in class-conditional density c).
- Note that *plates can be nested or crossing.*
- Note that θ_{jc} is used to generate x_{ij} iff $y_i = c$, otherwise it is ignored (this is certainly not clear from the nested notation).
- This is context specific independence, since the CI $x_{ij} \perp \theta_{jc}$ only holds if $y_i \neq c$.





- Heckerman, D., C. Meek, and D. Koller (2004). Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research.

Learning from Complete Data

- *If all the variables are fully observed in each data case (no missing data and no hidden variables) we say the data is complete.*
- For a DGM with complete data, the likelihood is given by

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(\mathbf{x}_i|\theta) = \prod_{i=1}^{N}\prod_{t=1}^{V} p(x_{it}|\mathbf{x}_{i,pa(t)},\theta_t) = \prod_{t=1}^{V} p(\mathcal{D}_t|\theta_t)$$

- $\mathcal{D}_t$ is the data associated with node *t* and its parents (t'th family). This is a *product of terms, one per CPD.* ***The likelihood decomposes according to the graph structure.***
- Assume a prior that factorizes as well: $p(\theta) = \prod_{t=1}^{V} p(\theta_t)$
- Then clearly *the posterior also factorizes:*

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) = \prod_{t=1}^{V} p(\mathcal{D}_t|\theta_t)p(\theta_t)$$

- *Thus we* ***can compute the posterior of each CPD independently.*** More details will be provided in a forthcoming lecture.

Learning from Complete Data

- As an example consider that all CPDs are tabular. We have a separate row (i.e., a separate multinoulli distribution) for each conditioning case.

- The t'th CPT is $x_t \mid \boldsymbol{x}_{pa(t)} = c \sim \mathcal{C}at\ \theta_{tc}$, where $\sum_k \theta_{tck} = 1$ with:

$$\theta_{tck} = p\ \ x_t = k \mid \boldsymbol{x}_{pa(t)} = c\ \ , k = 1{:}K_t, c = 1{:}C_t, t = 1{:}T \ with\ C_t = \prod_{s \in pa(t)} K_s$$

- Assume a separate Dirichlet prior on each row of each CPT, i.e., $\boldsymbol{\theta}_{tc} \sim \mathcal{D}ir(\boldsymbol{\alpha}_{tc})$. We can compute the posterior by simply adding the pseudo counts to the empirical counts $\boldsymbol{\theta}_{tc} | \mathcal{D} \sim \mathcal{D}ir\ (\mathbf{N}_{tc} + \boldsymbol{\alpha}_{tc})$, where N_{tck} is the number of times that node t is in state k while its parents are in state c:

$$\mathrm{N}_{tck} \triangleq \sum_{i=1}^{N} \mathbb{I}\left(x_{i,t} = k, x_{i,pa(t)} = c\right)$$

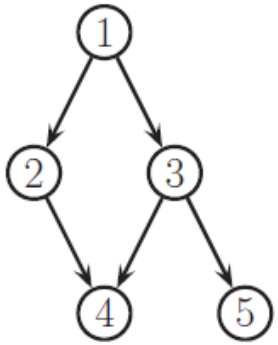
- The mean of this distribution is given by the following:

$$\bar{\theta}_{tck} = \frac{\mathrm{N}_{tck} + \boldsymbol{\alpha}_{tck}}{\sum_{k'} \mathrm{N}_{tck'} + \boldsymbol{\alpha}_{tck'}}$$

Learning from Complete Data

- Consider the DGM shown, Suppose the training data is:

x_1	x_2	x_3	x_4	x_5
0	0	1	0	0
0	1	1	1	1
1	1	0	1	0
0	1	1	0	0
0	1	1	1	0



- Below we list all the sufficient statistics N_{tck}, and *the posterior mean parameters* $\bar{\theta}_{tck}$ *under a Dirichlet prior with* $\alpha_{tck} = 1$ *(add-one smoothing) for the t = 4 node:*

x_2	x_3	$N_{tck=1}$	$N_{tck=0}$	$\bar{\theta}_{tck=1}$	$\bar{\theta}_{tck=0}$
0	0	0	0	1/2	1/2
1	0	1	0	2/3	1/3
0	1	0	1	1/3	2/3
1	1	2	1	3/5	2/5

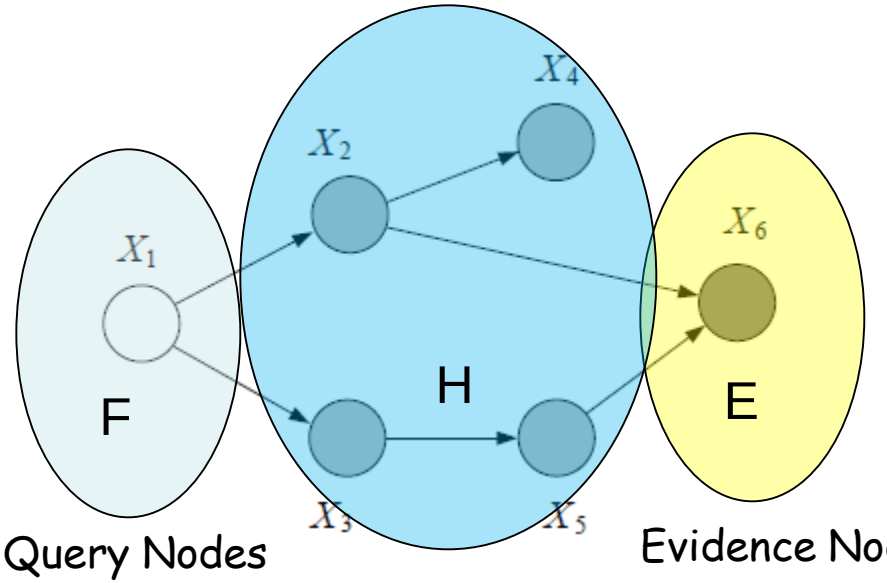
- The *MLE* has the same form without the α_{tck} terms:

$$\hat{\theta}_{tck} = \frac{N_{tck}}{\sum_{k'} N_{tck'}}$$

- The MLE suffers from the zero-count problem so it is important to use a prior.

Probabilistic Inference in General Graphs

- Let us now discuss the problem of computing conditional and marginal probabilities in general graphical models.
- We introduce the "elimination algorithm" for probabilistic inference. This algorithm applies to directed and undirected graphs.
- Let (*E,F,H*) be a partitioning of the indices/random variables of a graphical model into disjoint subsets. Our goal is to calculate $p(x_F \mid x_E)$ for arbitrary subsets *E*, *F* and *H*.



$$p(x_E, x_F) = \sum_{x_H} p(x_E, x_F, x_H)$$

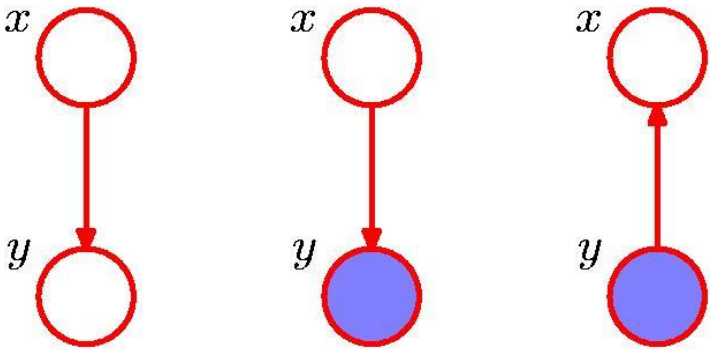
$$p(x_F \mid x_E) = \frac{\sum_{x_H} p(x_E, x_F, x_H)}{\sum_{x_F, x_H} p(x_E, x_F, x_H)}$$

Simplest Example: Bayes' Theorem

- Our goal is to compute the posterior distributions of one or more subsets of other nodes when some of the nodes in a graph are clamped to observed values (evidence).

- Graphic representation of Bayes' theorem (y is clamped)

$$p(x,y) = p(x)\,p(y|x)$$



$$p(x,y) = p(y)\,p(x|y)$$

- We infer the posterior of x given y and a prior p(x)

$$p(x|y) = \frac{p(y|x)\,p(x)}{p(y)}$$

$$p(y) = \sum_{x'} p(y|x')\,p(x')$$

Distributive Law

- Considering the following summation with many variables and one constant:

$$\sum_i a x_i = a \sum_i x_i$$

- We can generalize with summation (marginalization) over many variables:

$$\sum_{x_1,\ldots,x_N} \prod_{i=1}^{N} a_i x_i = \sum_{x_1} a_1 x_1 \sum_{x_2} a_2 x_2 \ldots \sum_{x_N} a_N x_N$$

- By *pushing the summations inwards we can do this marginalization efficiently.*

- *These operations and variable elimination in general are applicable to any commutative semi-ring (a set that together with the + and × (associative and commutative) operations satisfy the distributed law (a × b)+(a × c)=a ×(b+c).*

  - Bistarelli, S., U. Montanari, and F. Rossi (1997). Semiring-based constraint satisfaction and optimization. *J. of the ACM 44*(2), 201–236.

Performing The Summations

- Recall that the local conditional distributions in a directed graph can be thought of as tables (CPT)
- An inner sum over a latent variable x_i can be thought of as a new potential function
- This function defines a new table that no longer depends on x_i
- *Efficient algorithms for inference depend on exploiting these intermediate tables*

- Zhang, N. and D. Poole (1996). Exploiting causal independence in Bayesian network inference. *J. of AI Research*, 301–328.
- Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.
- Cannings, C., E. A. Thompson, and M. H. Skolnick (1978). Probability functions in complex pedigrees. *Advances in Applied Probability 10*, 26–61.
- Arnborg, S., D. G. Corneil, and A. Proskurowski (1987). Complexity of finding embeddings in a ktree. *SIAM J. on Algebraic and Discrete Methods 8*, 277–284.
- Kjaerulff, U. (1990). Triangulation of graphs – algorithms giving small total state space. Technical Report R-90-09, Dept. of Math. and Comp. Sci., Aalborg Univ., Denmark.
- Bertele, U. and F. Brioschi (1972). *Nonserial Dynamic Programming*. Academic Press.
- Bistarelli, S., U. Montanari, and F. Rossi (1997). Semiring-based constraint satisfaction and optimization. *J. of the ACM 44*(2), 201–236.

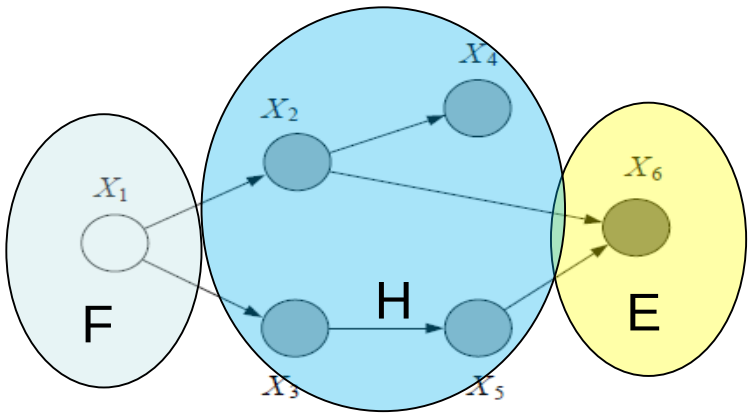
Performing the Summations

- As we already have seen, the needed summations for inference need attention.

$$p(x_E, x_F) = \sum_{x_H} p(x_E, x_F, x_H)$$

- The summation $\sum_{x_H}$ expands into a sequence of summations, one for each of the random variables indexed by *H.*

- *If each such random variable can take on r values, and there are |H| variables, we obtain $r^{|H|}$ terms in our summation which is generally infeasible.*

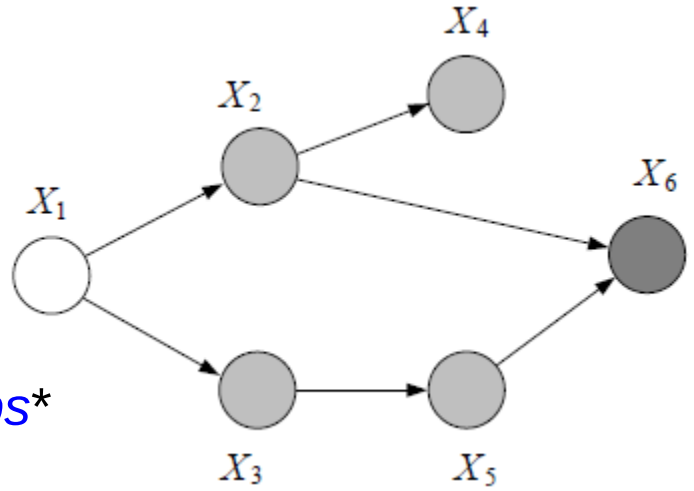


Performing the Summations

- Consider the graph below and assume that we want to compute

$$\sum_{x_6} p(x_1, x_2, x_3, x_4, x_5, x_6)$$

where the joint distribution is given as a table r^6.



- *A naïve calculation requires r^6 calculations* (we need to update the Table of size r^5 – each update takes r summations).

- But note *a calculation that takes r^3 steps**

$$\sum_{x_6} p(x_1, x_2, x_3, x_4, x_5, x_6) =$$

$$p(x_1)\,p(x_2 \mid x_1)\,p(x_3 \mid x_1)\,p(x_4 \mid x_2)\,p(x_5 \mid x_3) \underbrace{\sum_{x_6} p(x_6 \mid x_2, x_5)}_{\phi_{x_6}(x_2, x_5)\,(=1)}$$

* In this case, no computation needs to be performed as marginalization of the conditional $p(x_6|x_2,x_5)$ gives 1!

Performing the Summations

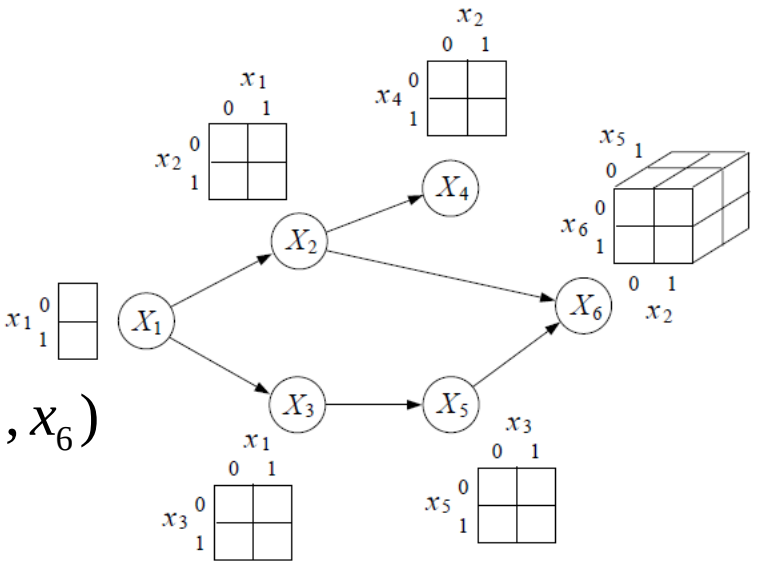
- Similarly consider performing

$$\sum_{x_5} p(x_1, x_2, x_3, x_4, x_5, x_6)$$

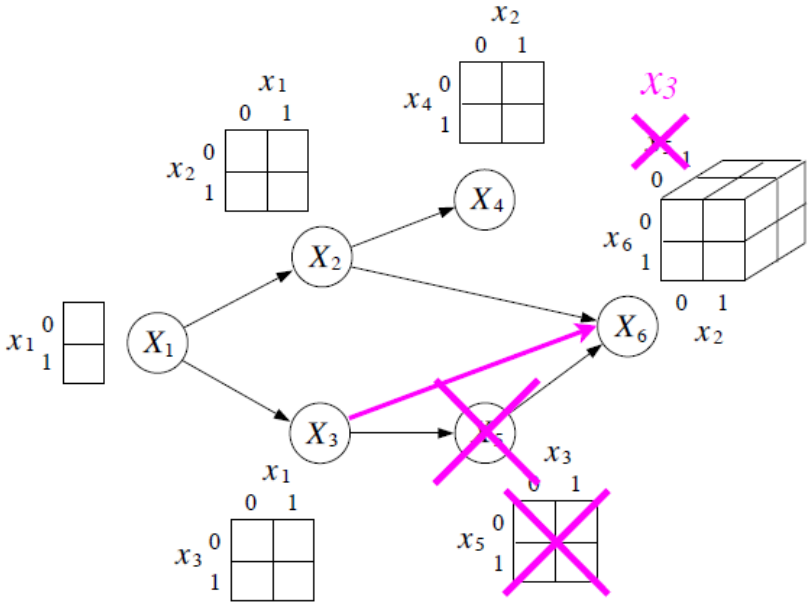
$$p(x_1, x_2, x_3, x_4, x_6) = \sum_{x_5} p(x_1, x_2, x_3, x_4, x_5, x_6)$$

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

$$p(x_4 \mid x_2) \underbrace{\sum_{x_5} p(x_5 \mid x_3)\, p(x_6 \mid x_2, x_5)}_{\phi_{x_5}(x_2, x_3, x_6)}$$

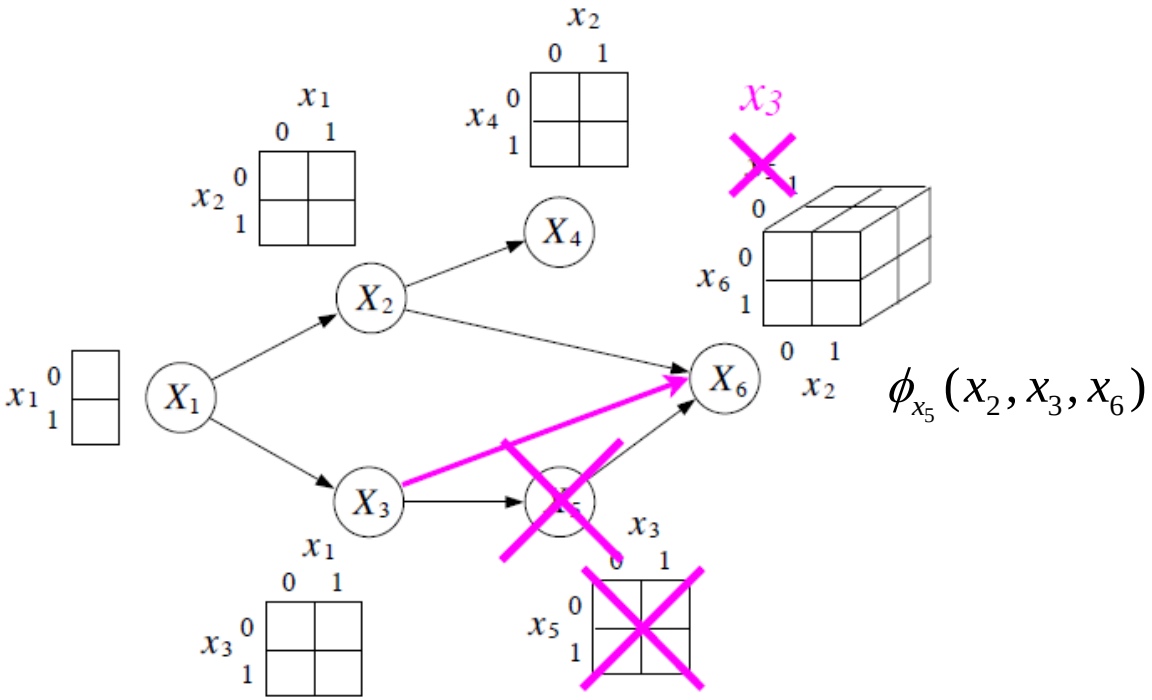


- The new modified CPT Tables are:



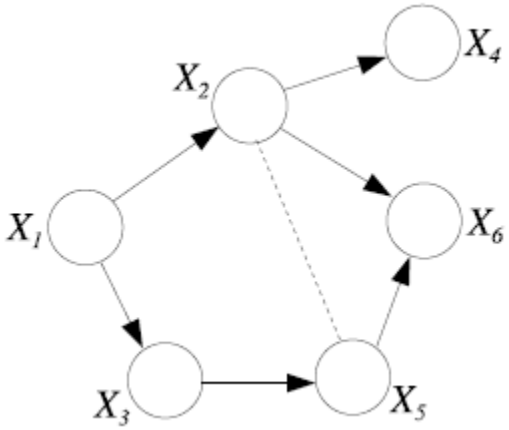
Performing the Summations

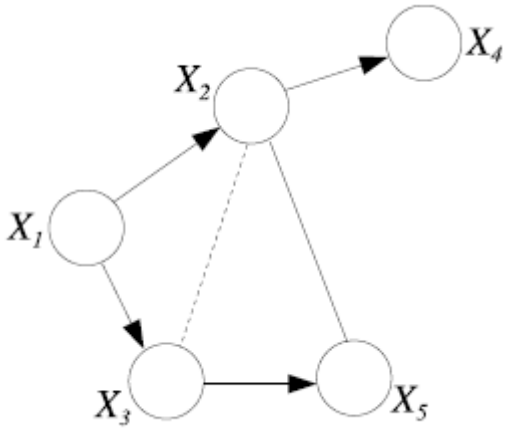
- In the general case, a series of new potential functions will be introduced, and they need to be distinguished
- Let the potential function that arises when summing over x_i, with dependent variables x_{Si}, be denoted $\phi_i(x_{Si})$
- Thus, the function in the previous example would be denoted as $\phi_{x_5}(x_2, x_3, x_6)$



Elimination Algorithm & Graph Elimination

- Consider computing $p(x_1,x_2)$. We first eliminate X_6 – which in the created intermediate message links nodes X_2 & X_5.



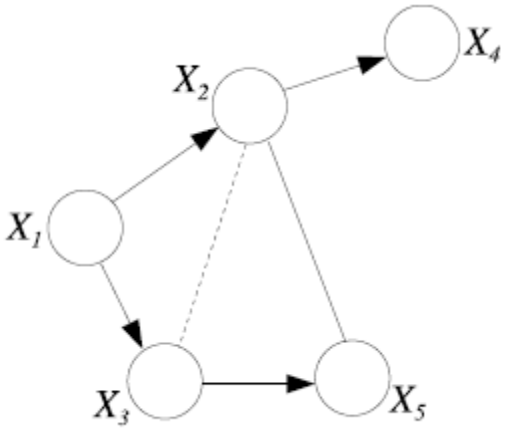


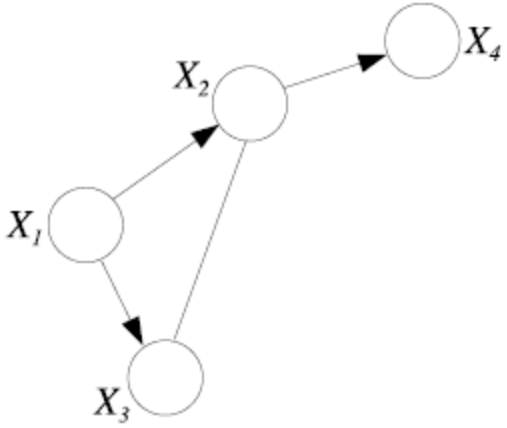
$$p(x_1,x_2) = \sum_{x_3,\ldots,x_6} p(x_1,x_2,x_3,x_4,x_5,x_6)$$

$$= \sum_{x_3,x_4,x_5} p(x_1)p(x_2\mid x_1)p(x_3\mid x_1)p(x_4\mid x_2)p(x_5\mid x_3)\underbrace{\sum_{x_6} p(x_6\mid x_2,x_5)}_{\phi_{x_6}(x_2,x_5)}$$

Elimination Algorithm & Graph Elimination

- We next eliminate x_5 – which in the created intermediate message links nodes X_2 and X_3.

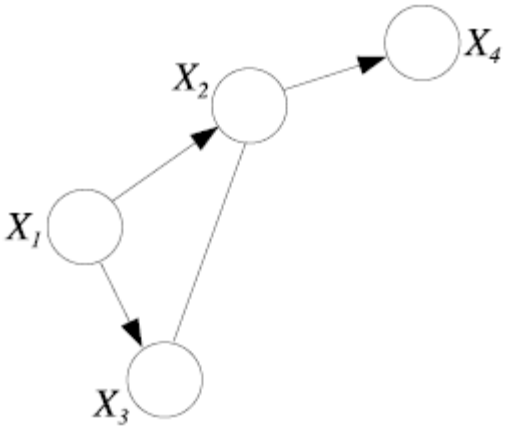


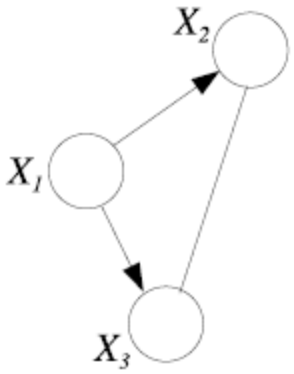


$$p(x_1, x_2) = \sum_{x_3, x_4} p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1)\, p(x_4 \mid x_2) \underbrace{\sum_{x_5} \phi_{x_6}(x_2, x_5)\, p(x_5 \mid x_3)}_{\phi_{x_5}(x_2, x_3)}$$

Elimination Algorithm & Graph Elimination

- We next eliminate x_4 – no links are generated in this step.



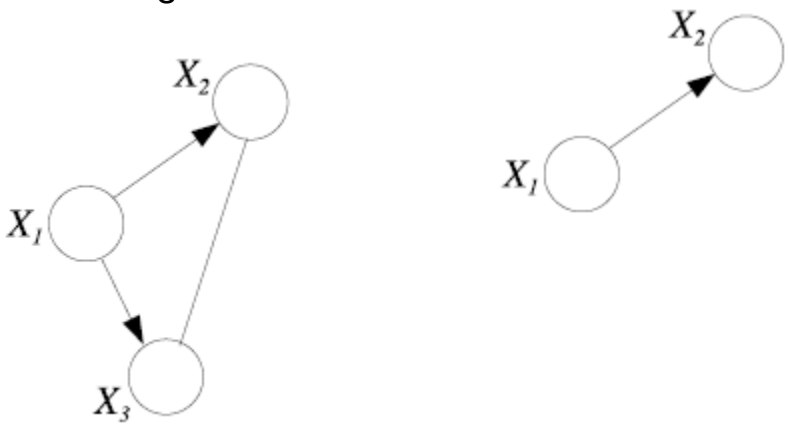


$$p(x_1, x_2) = \sum_{x_3} p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1)\, \phi_{x_5}(x_2, x_3) \underbrace{\sum_{x_4} p(x_4 \mid x_2)}_{\phi_{x_4}(x_2)=1}$$

Run VariableElimination.m from PMTK3

Elimination Algorithm & Graph Elimination

- We next eliminate x_3.



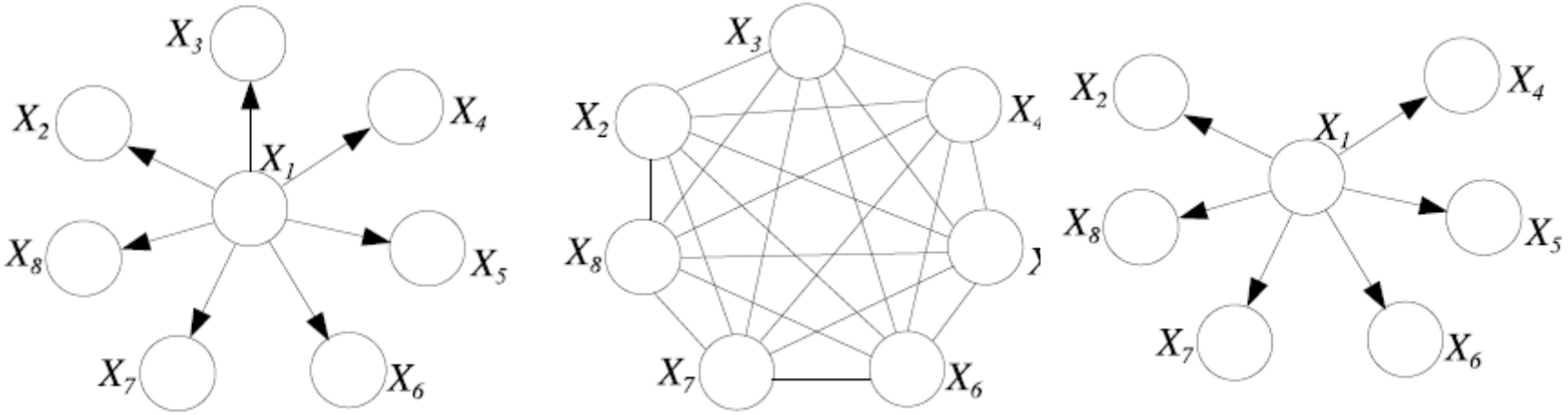
$$p(x_1, x_2) = p(x_1)\, p(x_2 \mid x_1)\, \phi_{x_4}(x_2) \underbrace{\sum_{x_3} p(x_3 \mid x_1)\, \phi_{x_5}(x_2, x_3)}_{\phi_{x_3}(x_1, x_2)}$$

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

- *If the elimination of nodes is in the same order as the summation over variables, the size of the largest elimination clique is the same as the max number of variables involved in the summation.*

Elimination Algorithm & Graph Elimination

- *The elimination order is critical.* In the graph below, if we eliminate X_1 first, then all other nodes need to be fully connected. However, if X_3 is eliminated first, no additional links are needed.



$$p(x_2, x_3, x_4, x_5, x_6, x_7, x_8) = \underbrace{\sum_{x_1} p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_1) \ldots p(x_8 \mid x_1)}_{\phi_{x_1}(x_2, \ldots, x_8)}$$

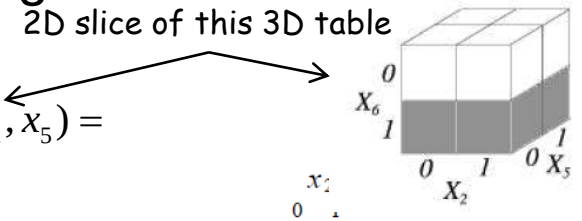
$$p(x_1, x_2, x_4, x_5, x_6, x_7, x_8) = p(x_1)\, p(x_2 \mid x_1)\, p(x_4 \mid x_1) \ldots p(x_8 \mid x_1) \underbrace{\sum_{x_3} p(x_3 \mid x_1)}_{\phi_{x_3}(x_1)=1}$$

Clamped Variables

- The evidence variables X_E are fixed constants in these calculations
- They are said to be clamped at their observed values.
- We distinguish a possible value of a variable x_i from a clamped value by denoting the later as $\bar{x}_i$
- The same circumstance arises when computing marginal probabilities (e.g. the likelihood of a model where the latent variables are integrated out)

Inner Summations and New Tables

- We want to compute $p(x_1|\bar{x}_6)$. An inner sum over a latent variable x_i can be thought of as a new potential function that defines a new table that no longer depends on x_i . Efficient algorithms for inference exploit these intermediate tables.



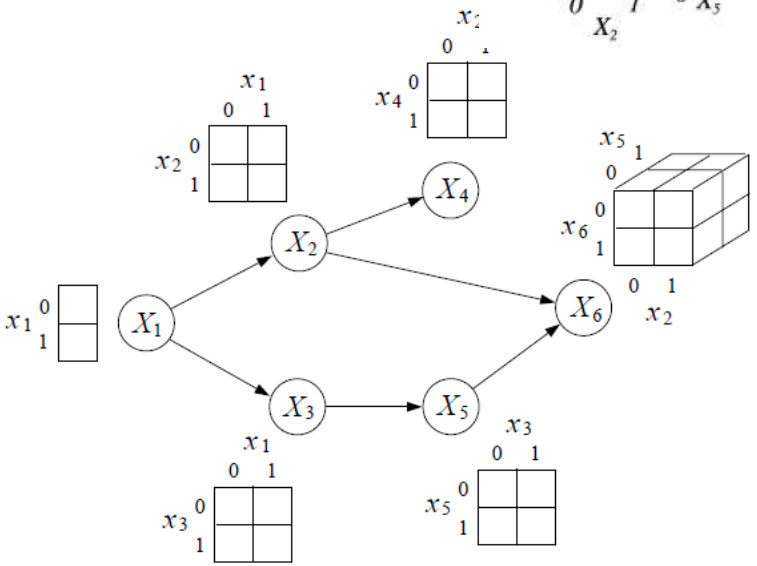
$$p(x_1,\bar{x}_6) = p(x_1)\sum_{x_2} p(x_2\mid x_1)\sum_{x_3} p(x_3\mid x_1)\sum_{x_4} p(x_4\mid x_2)\sum_{x_5} p(x_5\mid x_3)\,p(\bar{x}_6\mid x_2,x_5) =$$

$$p(x_1)\sum_{x_2} p(x_2\mid x_1)\sum_{x_3} p(x_3\mid x_1)\underbrace{\sum_{x_4} p(x_4\mid x_2)\phi_{x_5}(x_2,x_3)}_{\phi_{x_4}(x_2)\,(=1)} =$$

$$p(x_1)\sum_{x_2} p(x_2\mid x_1)\phi_{x_4}(x_2)\underbrace{\sum_{x_3} p(x_3\mid x_1)\phi_{x_5}(x_2,x_3)}_{\phi_{x_3}(x_1,x_2)} =$$

$$\underbrace{p(x_1)\sum_{x_2} p(x_2\mid x_1)\phi_{x_4}(x_2)\phi_{x_3}(x_1,x_2)}_{\phi_{x_2}(x_1)} = p(x_1)\phi_{x_2}(x_1)$$

$$p(x_1\mid\bar{x}_6) = \frac{p(x_1)\phi_{x_2}(x_1)}{\sum_{x_1} p(x_1)\phi_{x_2}(x_1)}$$



- *What is hidden in this algorithm is the set of variables on which each summation operates.* The graph elimination algorithm provides that.

Inner Summations and New Tables

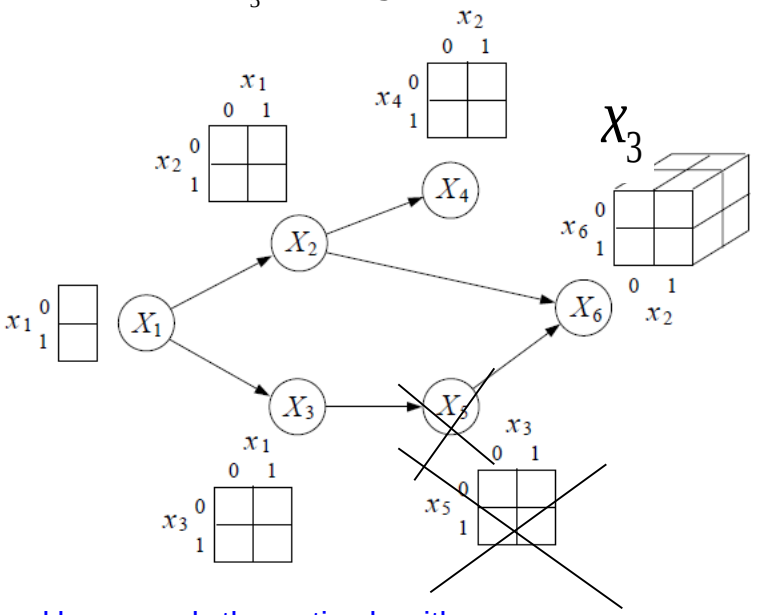
- An inner sum over x_5 leads to a new potential function that defines a new table that no longer depends on x_5 . *These summations are of reduced complexity $\mathcal{O}(r^k)$ (for some k<<V, V=number of nodes) leading to an overall complexity of $\mathcal{O}(Vr^k)$ instead of $\mathcal{O}(r^V)$.**

$$p(x_1, x_2, x_3, x_4, \bar{x}_6) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)\underbrace{\sum_{x_5} p(x_5 \mid x_3)p(\bar{x}_6 \mid x_2, x_5)}_{\phi_{x_5}(x_2, x_3, \bar{x}_6)} =$$

$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2)\phi_{x_5}(x_2, x_3, \bar{x}_6)$$

The evidence variables X_E *are fixed* constants in these calculations.

We denote the fixed value of the variable X_6 as $\bar{x}_6$.



New Table of x_2, x_3, x_6

* The parameter k can be determined by a graph-theoretic algorithm.

Evidence Potentials

- Let X_i be an evidence node whose observed value is $\bar{x}_i$. We use a trick to *view conditioning on the evidence as a summation process:*

$$g\left(\bar{x}_i\right) = \sum_{x_i} g(x_i)\delta(x_i, \bar{x}_i), \quad where \quad \delta(x_i, \bar{x}_i) = \begin{cases} 1 & if \ x_i = \bar{x}_i \\ 0 & otherwise \end{cases}$$

We refer to the δ functions as *evidencial functions.*

- For example the evaluation of $p(x_6|x_2,x_5)$ based on the evidence $\bar{x}_6$ can be written as:

$$\phi_{X_6}(x_2, x_5)\left(\equiv p\left(\bar{x}_6 \mid x_2, x_5\right)\right) = \sum_{x_6} p\left(x_6 \mid x_2, x_5\right)\delta(x_6, \bar{x}_6)$$

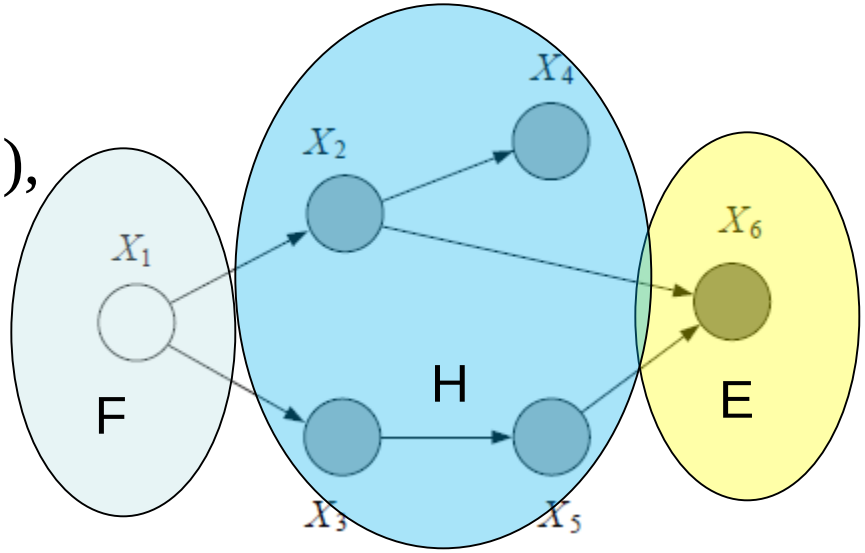
- This is only a trick – we don't really perform the summation but rather take a slice from the distribution $p\left(x_6 \mid x_2, x_5\right)$

Evidence Potentials

- Thus for our earlier inference problem

$$p(x_F, \bar{x}_E) = \sum_{x_E} p(x_F, x_E)\delta(x_E, \bar{x}_E),$$

$$p(\bar{x}_E) = \sum_{x_E, x_F} p(x_F, x_E)\delta(x_E, \bar{x}_E)$$



- In some cases, it is useful to work with the unnormalized distribution:

$$p^E(x) = p(x)\delta(x_E, \bar{x}_E)$$

- This notation eliminates special case handling of evidence notes and allows for a more general formulation of the elimination algorithm to be considered shortly.

Elimination Algorithmin Directed Graphs

- Choose an *elimination ordering I* such that the query node *F* appears last
- Maintain active list of *potential functions*
- For each node x_i in *I*, remove all potentials that reference it from the active list, take their product, and sum over x_i
- Add this new potential function to the active list and repeat
- Normalize for final conditional probability

Elimination Algorithm on Directed Graphs

Elimination(*G*, *E*, *F*)
- Initialize(*G*, *F*)
- Evidence(*E*)
- Update(*G*)
- Normalize(*F*)

Initialize(*G*, *F*)

choose an ordering *I* such that X_F appears last

for each node X_i in *G*

place $p(x_i \mid x_{\pi i})$ on the active list

end

Evidence(*E*)

for each node X_i in X_E

place $\delta(x_i, \bar{x}_i)$ on the active list

end

Elimination Algorithm on Directed Graphs

Elimination(*G*, *E*, *F*)
- Initialize(*G*, *F*)
- Evidence(*E*)
- Update(*G*)
- Normalize(*F*)

Update(*G*)

for each X_i *in I*

find all potentials from the active list that reference X_i *and remove them from the active list*

let ψ_{Xi} *be the product of these potentials*

let $\phi_{X_i} = \sum \psi_{X_i}$

place $\phi_{Xi}^{x_i}$ *on the active list*

end

Normalize(*F*)

$p(x_F \mid x_E) \leftarrow \psi_{X_F}(x_F)/\phi_{X_F}$

Active List of Potential Functions

- Throughout the algorithm we maintain an active list of potential functions. The active list is initialized to hold the local conditional probabilities, $p(x_i \mid x_{\pi_i})$, *for* $X_i \in \mathcal{I}$, *and the evidence functions,* $\delta(x_i, \bar{x}_i)$, *for* $X_i \in X_E$.

- At each step of the algorithm, we find all those potentials on the active list that reference the next node (call it X_i) in the elimination ordering I. These potential functions are removed from the active list. We take the product of these functions and sum this product with respect to x_i.

- This defines a new intermediate term, ϕ_{X_i}, that we add to the active list.

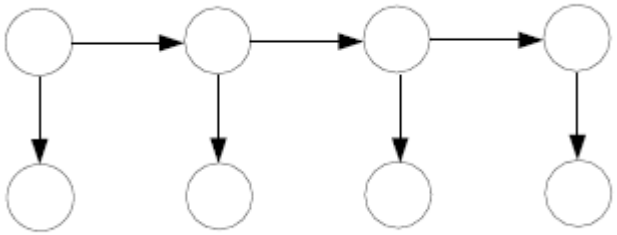
$$\phi_{X_i}(x_{S_i}) = \sum_{x_i} \psi_{X_i}(x_{C_i}),\ where\ x_{S_i} \triangleq x_{c_i} \backslash x_i,\ x_{C_i} = elimination\ clique$$

- *We then proceed to the next* node in the elimination ordering.

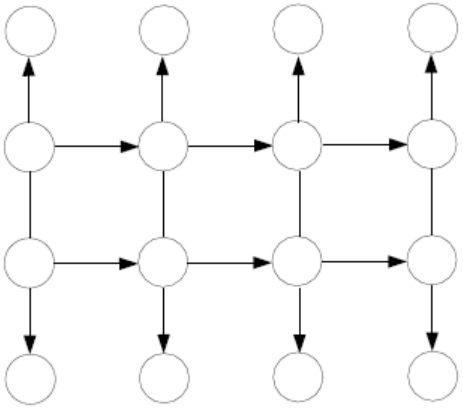
- The algorithm terminates when we arrive at the query node X_F .

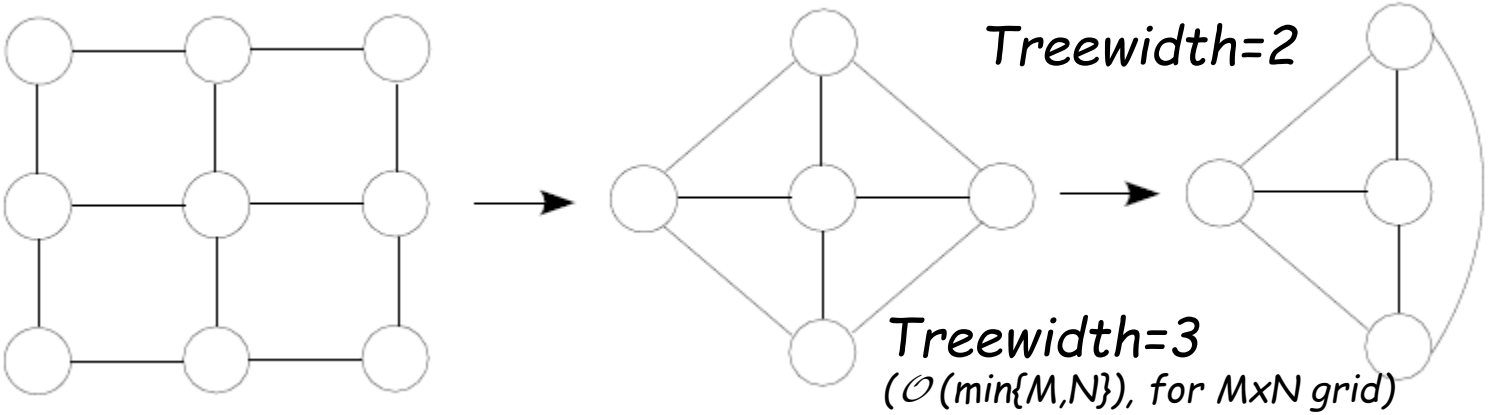
Treewidth

- We define the treewidth as the minimum size (over all possible eliminations) of the maximal cliques created during graph elimination MINUS one.



Treewidth=1





Treewidth=2

Treewidth=3

($\mathcal{O}$(min{M,N}), for MxN grid)

- For variables with K states each, the cost of the VE algorithm is $\mathcal{O}$ (VK^{w+1}). Finding an elimination order to minimize the induced width is NP-hard.

Elimination Algorithm for Undirected Graphs

- The elimination algorithm for undirected graphs remains practically as for DGs.

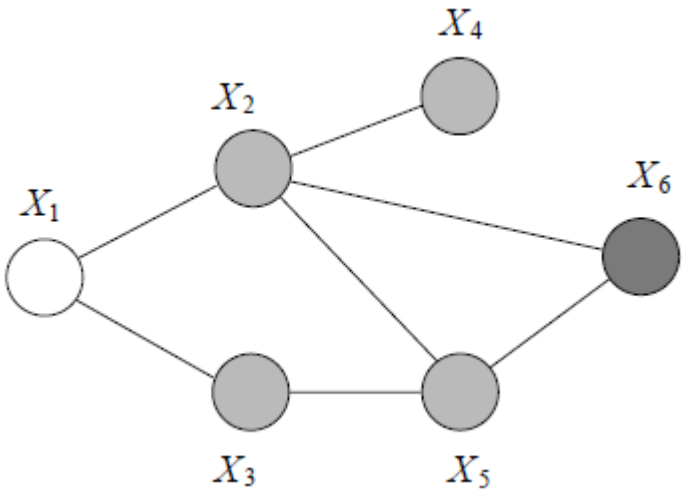
- Minor differences include:

  - We initialize with *local potentials rather than conditional probabilities.*

  - We need to *consider the normalization constant* (this has implications for termination and nodes without children)

Elimination Algorithm for Undirected Graphs

- The entire Elimination algorithm also applies to the undirected case.
- The only change needed is in the Initialize procedure, where instead of using local conditional probabilities we initialize the active list to contain the potentials $\psi_{XC}(x_C)$.



- In this example, we represent the joint probability on the graph via potential functions on the cliques $\{X_1,X_2\}$ $\{X_1,X_3\}$, $\{X_2,X_4\}$, $\{X_3,X_5\}$, and $\{X_2,X_5,X_6\}$. Let us calculate the potential $p(x_1, x_6)$

Inference in Undirected Graphs

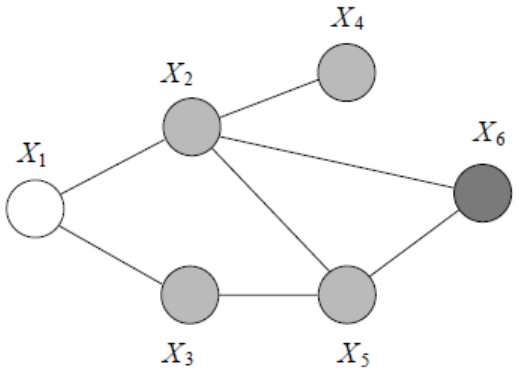
$$p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5)\psi(x_2, x_5, \bar{x}_6)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3)\phi_{X_5}(x_2, x_3) \sum_{x_4} \psi(x_2, x_4)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2)\phi_{X_4}(x_2) \sum_{x_3} \psi(x_1, x_3)\phi_{X_5}(x_2, x_3)$$

$$= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2)\phi_{X_4}(x_2)\phi_{X_3}(x_1, x_2) = \frac{1}{Z}\phi_{X_2}(x_1)$$



$$p(x_1 \mid \bar{x}_6) = \frac{\phi_{X_2}(x_1)}{\sum_{x_1} \phi_{X_2}(x_1)}$$

- $\phi_{X4}(x_2)$, which earlier could be omitted, no longer necessarily sums to one and must be explicitly carried along in the calculation.
- Note that we don't need to compute Z explicitly. Instead we can use the final message to quickly compute Z for all marginal distrib.

Graph Elimination

- While the ELIMINATE algorithm provides a solution to our inference problem (by performing summations over a product of potential functions), it does not address
  - How to control the size of the summands of the potentials
  - How to compute and improve the computational complexity.
- We proceed next to define a fully graph-theoretic approach to the inference problem that overcomes the limitations of the ELIMINATE algorithm.
- *Questions regarding the computational complexity of the algorithm can be answered on purely graph theoretic arguments.*

Graph Elimination

- We describe a simple procedure that eliminates nodes in a graph based *solely on graph-theoretic manipulations*.

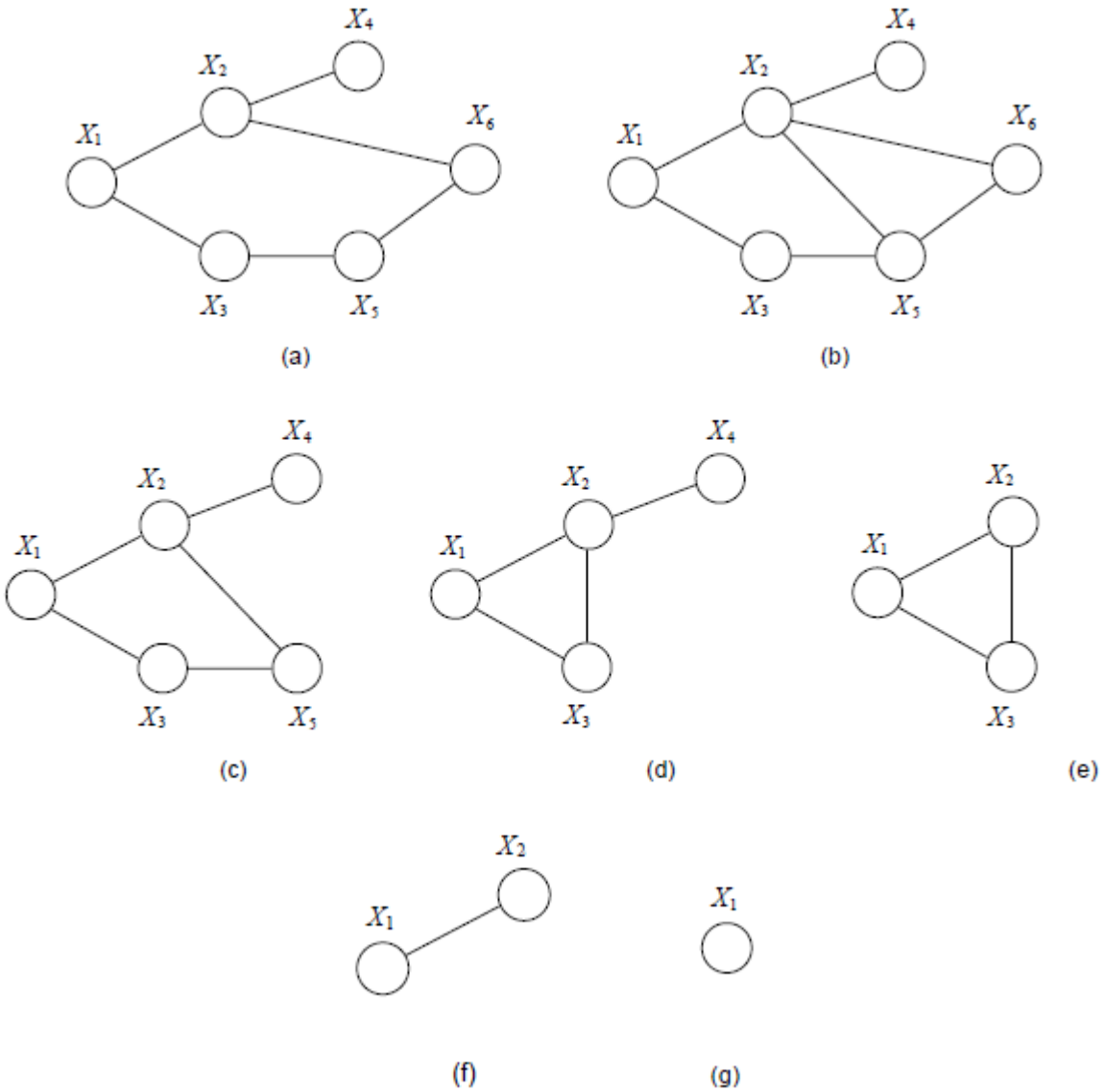
- Given an undirected graph G, we first choose an ordering I of the nodes. This will be our elimination ordering. We then eliminate the nodes in sequence, connecting the (remaining) neighbors of each node.

UndirectedGraphEliminate(*G, I)*
  for each node X_i *in I*
    connect all of the neighbors of X_i
    remove X_i *from the graph*
  end

- Let us see how this algorithm works in our earlier example.

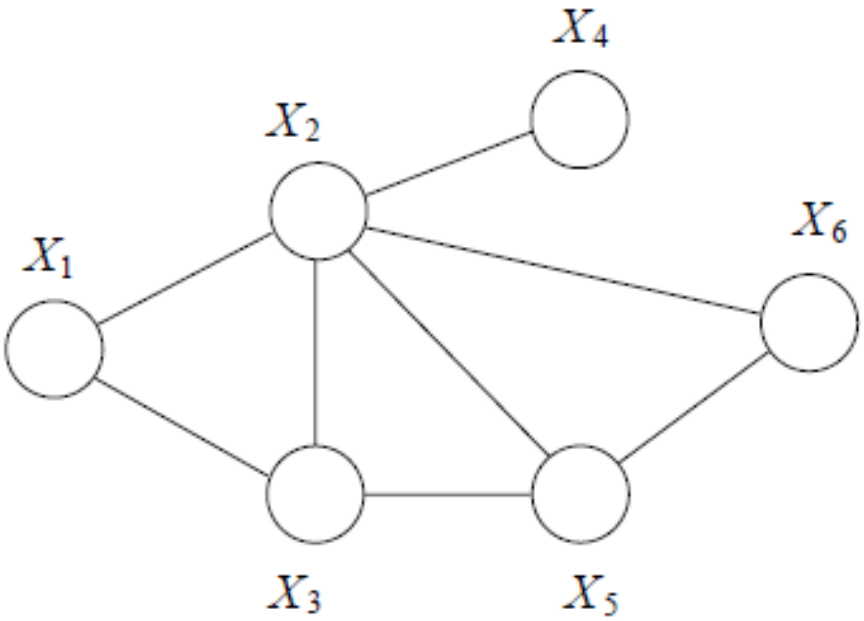
A Run of the Graph Elimination Algorithm

- Consider the elimination sequence $(X_6,X_5,X_4,X_3,X_2,X_1)$.
- Starting with node X_6 we first connect its neighbors, adding an edge between X_2 and X_5.
- We then remove X_6. Moving to X_5, we connect its neighbors, X_2 and X_3, and remove X_5, etc.
- *Each time a node is eliminated, all of its neighbors are connected forming a clique.*



Graph Elimination in Undirected Graphs

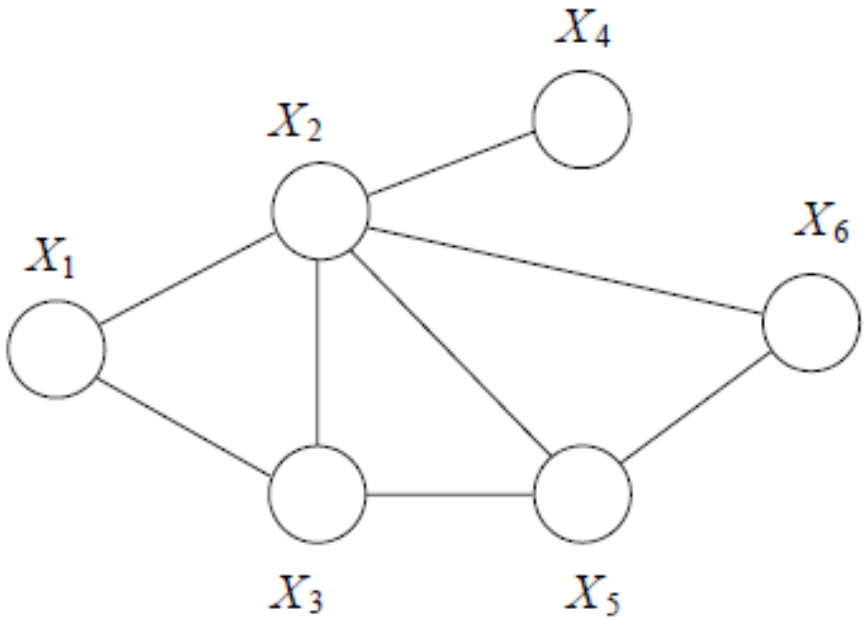
- We show the *reconstituted graph* (it is a triangulated graph) with all edges that were added during the elimination.
- The relevant properties of the graph can be captured by recording the *elimination cliques of the graph.*
- *Each time we remove a node X_i in step (2) of* the algorithm, we record the collection of nodes that are the neighbors of X_i at that moment, including X_i itself.
- These nodes form a fully-connected subset of nodes by virtue of step (1); that is, they form a clique.



Here, C_6 = {2, 5, 6} and C_5 = {2, 3, 5}.

Graph Elimination in Undirected Graphs

- *The elimination cliques define the sets of variables on which summations operate during marginalization.*
- *The largest such clique determines the overall complexity (k) of marginalization.*
- *k* is referred to as the *tree-width of the graph.*
- When removing a random variable from a joint distribution, we perform a sum over the product of all factors that depend on that random variable.
- This couples all other random variables (neighbors of the node in the graph) that appear in those factors.

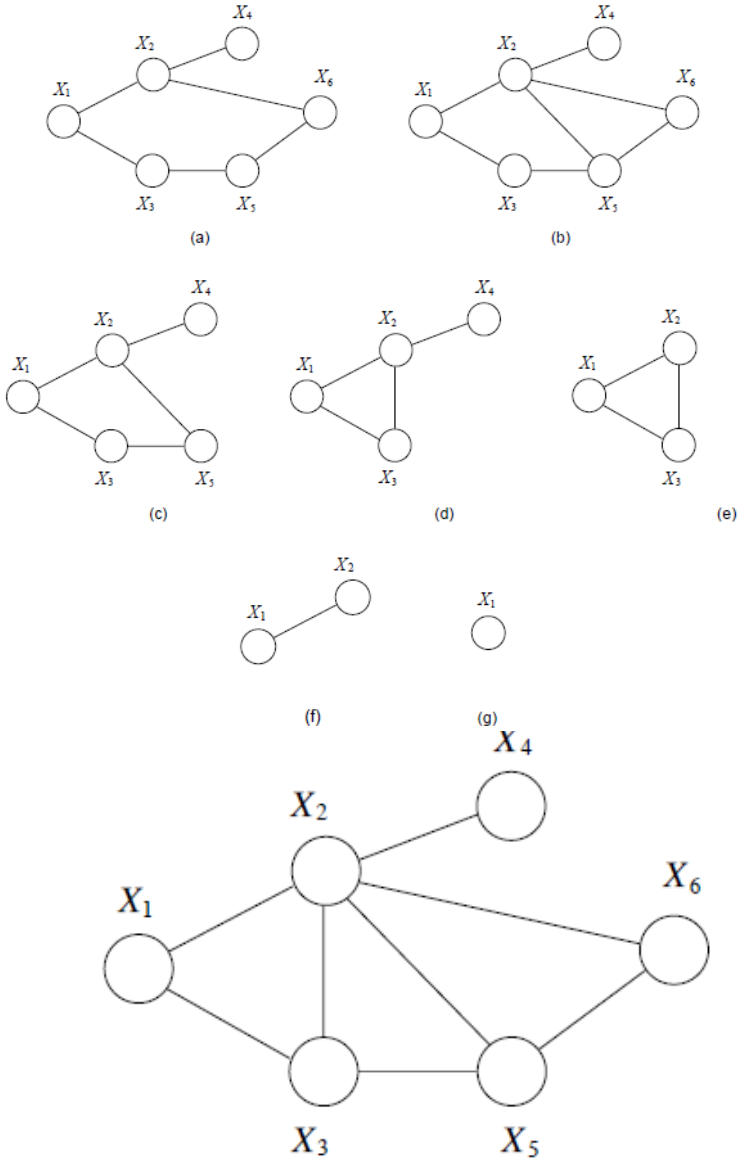


Inference in Undirected Graphs

- The undirected case is not affected by moralization as in directed graphs.
- *The elimination cliques created by UndirectedGraphEliminate are directly the arguments to the potentials created during the Elimination algorithm.*
- The calculation of Z is a summation over the unnormalized representation of the joint probability (summation over all of the variables).

Induced Dependencies

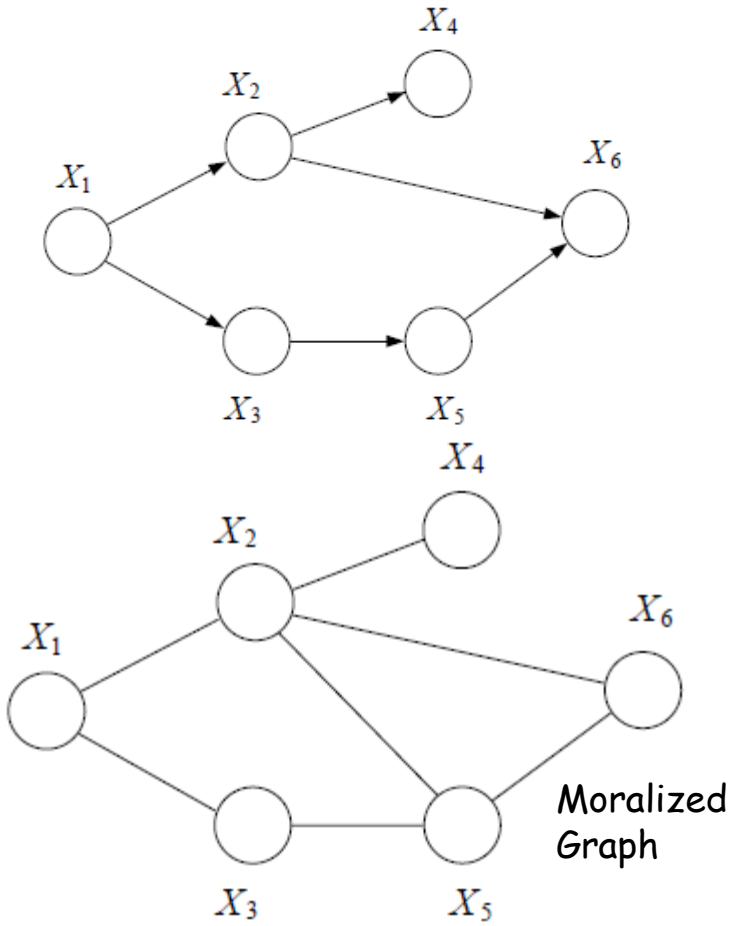
- As nodes are eliminated, new (unfactorable) potentials are introduced that involve all neighboring nodes
- These potentials correspond to new tables of dimension equal to the number of neighbors of the node
- *The elimination of a node has the effect of creating new dependencies (edges) between all pairs of neighbors, thus introducing a new clique in the graph*
- This is known as ***triangulation***



Graph Elimination in a Directed Graph

DirectedGraphEliminate(*G, I)*
    G^m = *Moralize(G)*
    UndirectedGraphEliminate(G^m, *I)*

Moralize(*G)*
    for each node X_i *in I*
        connect all of the parents of X_i
    end drop the orientation of all edges
    return *G*



- Before applying the Elimination algorithm, we need to moralize (marry the parents and convert to an undirected graph)

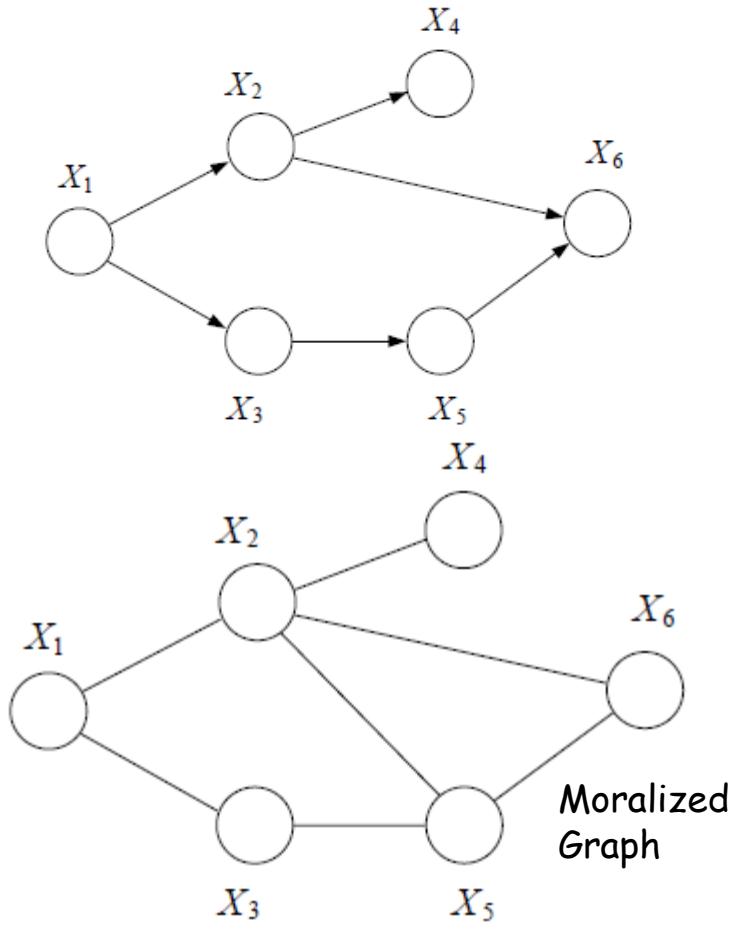
- The moralized graph of the directed graph on the top is shown.

Graph Elimination in a Directed Graph

- If we eliminate X_6 first (before its parents), then moralization is not needed as the graph elimination will link for us the neighbors X_2 and X_5 of X_6.
- However, if we eliminate X_5 first, then X_2 is not included within the elimination clique of X_5. This fails to capture the fact that summing over X_5 creates an intermediate term that refers to X_2:

$$\sum_{x_5} p(x_5 \mid x_3)\, p(x_6 \mid x_2, x_5)$$

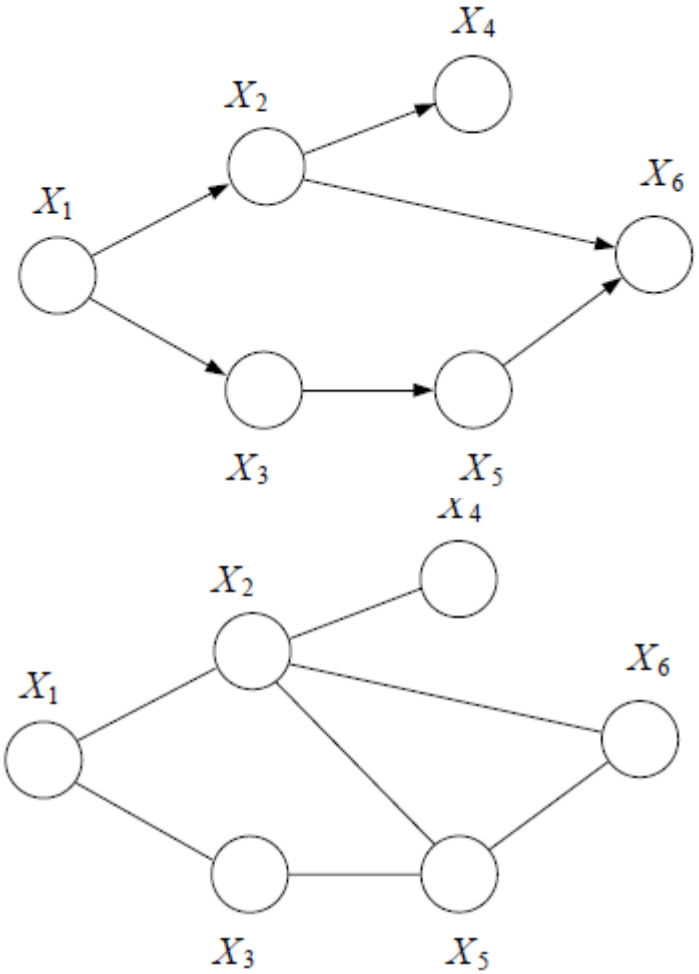
- However, if we moralize as shown, then the elimination algorithm will link the neighbors X_2 and X_3 before it eliminates X_5 (X_2 and X_3 become neighbors after we marry X_2 and X_5)



Graph Elimination in a Directed Graph

DirectedGraphEliminate(*G, I)*
    G^m = *Moralize(G)*
    UndirectedGraphEliminate(G^m, *I)*

Moralize(*G)*
    for each node X_i *in I*
        connect all of the parents of X_i
    end drop the orientation of all edges
    return *G*

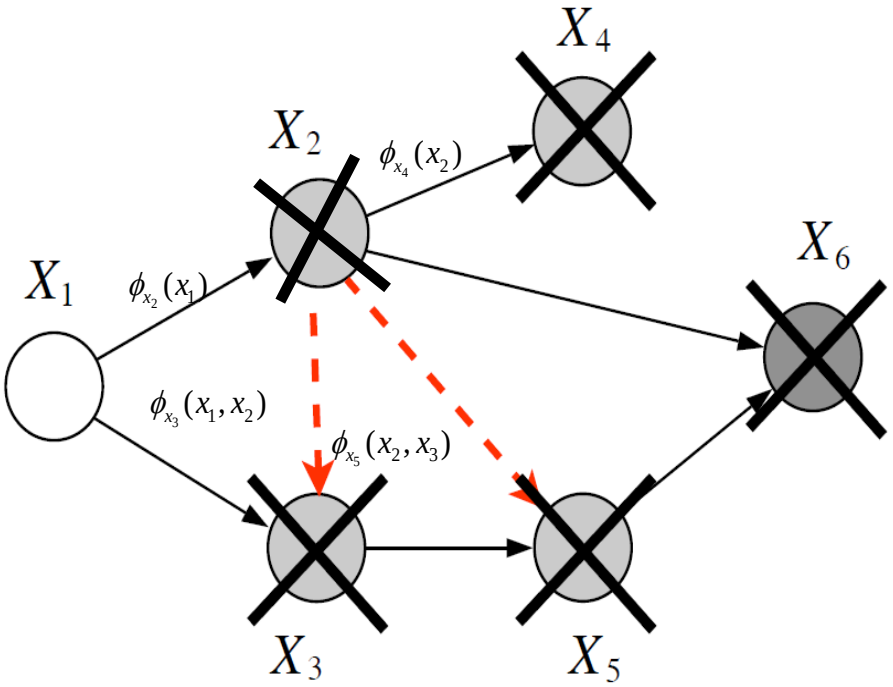


- Thus summing $p(x_5 \mid x_3)p(x_6 \mid x_2, x_5)$ wrt x_5 creates an intermediate factor that involves x_2 and x_3 and *we have an induced dependency between x_2 and x_3.*
- UndirectedGraphEliminate links the neighbors of the node being summed over to make this coupling explicit. *Elimination cliques are the graph-theoretic counterpart of the sets of variables on which summations operate.*

Inner Summations and New Tables

- Note that in computing $p(x_1 \mid \bar{x}_6)$, *we first moralize the graph and then transform it to an undirected graph before we start the Elimination process.* Note the two red links (one from the moralization process (X_2-X_5), the other from connecting neighbors (X_2-X_3) in the Graph Elimination Algorithm).

- A graphical representation of the operation is shown below.



Induced Dependencies in Directed Graphs

- In directed graphs, the potential functions (conditional distributions) implicitly create dependencies between the parents of a child node.

- These dependencies come into play in the algorithm and need to be considered in assessing computational complexity.

- For this purpose*, it is sufficient to moralize the graph, then proceed with the resulting undirected graph*.

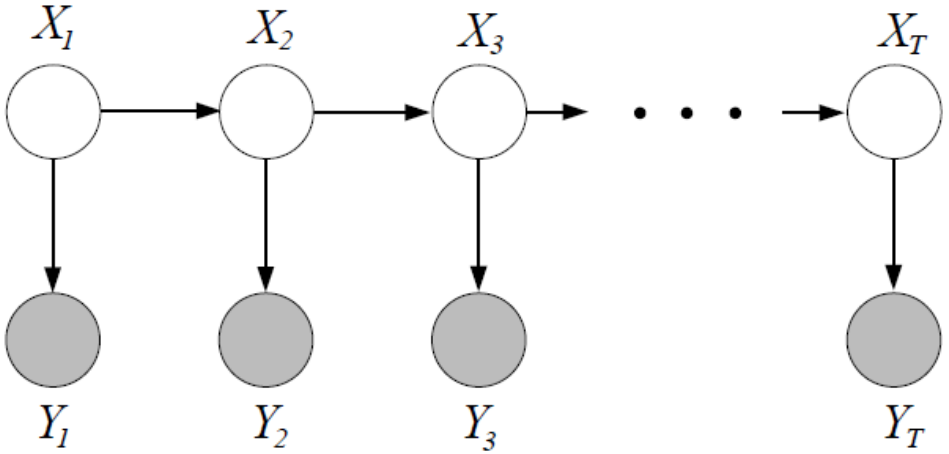
Computational Complexity

- The limiting step in the algorithm is the computation of each new potential
- This requires the generation of a new table of dimension $|S_i|$ (set of variables in the elimination clique omiting x_i itself), with $r^{|S_i|}$ cells (where *r* is the cardinality)
- The summation requires another factor of *r* for a total complexity of $\mathcal{O}(r^{|T_i|})$, $T_i = \{i\} \cup S_i$ (all variables that appear in the operand $\sum_{x_i}$
- Notice *that $|T_i|$ is also the size of the elimination clique*
- *Thus the algorithm has a running time that is exponential in the size of the largest elimination clique*

- Arnborg, S., D. G. Corneil, and A. Proskurowski (1987). Complexity of finding embeddings in a ktree. *SIAM J. on Algebraic and Discrete Methods 8*, 277–284.
- Kjaerulff, U. (1990). Triangulation of graphs – algorithms giving small total state space. Technical Report R-90-09, Dept. of Math. and Comp. Sci., Aalborg Univ., Denmark.

Elimination Algorithm Details

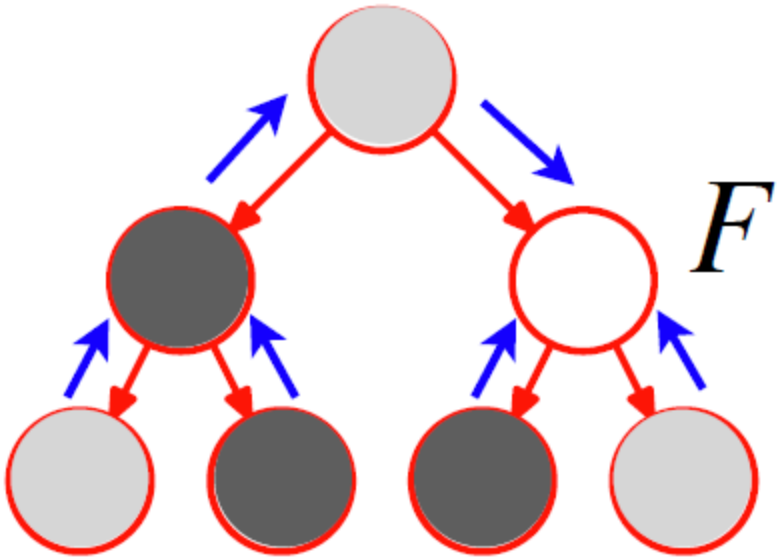
- Need data structure to avoid linear search of active list
- *The elimination algorithm only handles single query node; extension to multiple nodes nontrivial* – A general procedure is needed for avoiding redundant calculations (e.g. in the chain below running a single forward and backward pass)



- The Sum-Product allows computing all marginal probabilities in one run – but it is exact only for trees not arbitrary graphs.
- The Junction-Tree computes all marginals and it is exact for arbitrary graphs but computationally expensive. Runtime can be determined beforehand for each graph.

Elimination on Trees

- Let F be an arbitrary node in a tree. The optimal ordering proceeds inwards from the leaves.
- No new edges are created during elimination
- Inference takes $\mathcal{O}(Nk^2)$ time where k is the node cardinality.



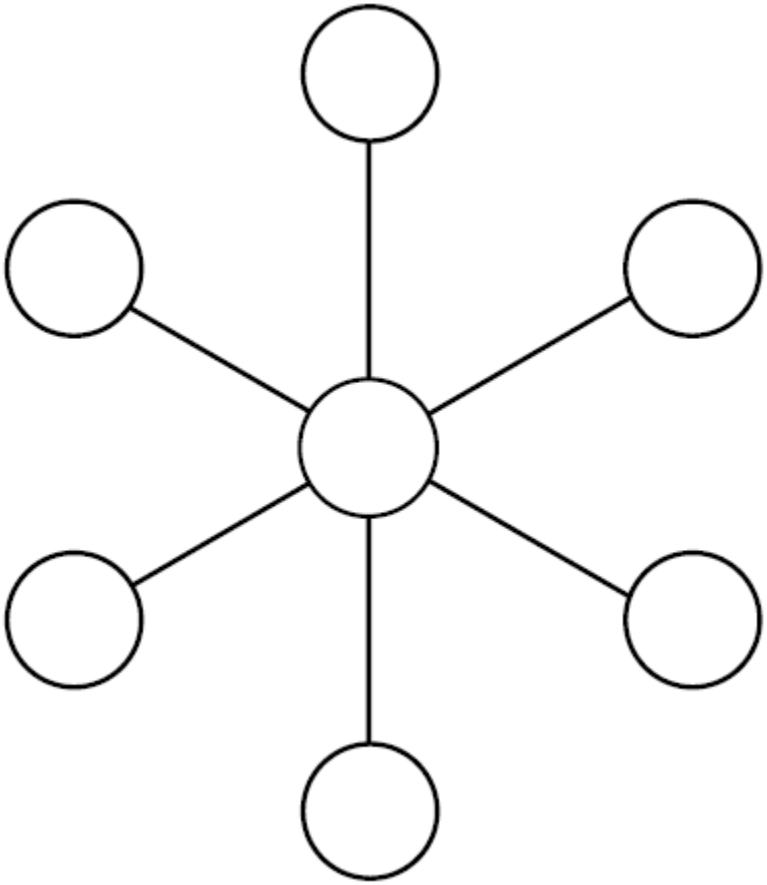
Elimination Orderings-Treewidth

- Notice that the elimination cliques, and hence the running time, are a function of the elimination ordering
- We desire an *ordering that minimizes the size of the largest elimination clique*
- *The minimum such size over all elimination orderings (minus 1) is called the **treewidth** of the graph*
- *Finding the treewidth is in general NP-hard, but good heuristics are available*
- In many cases of practical interest, the optimal elimination ordering is obvious

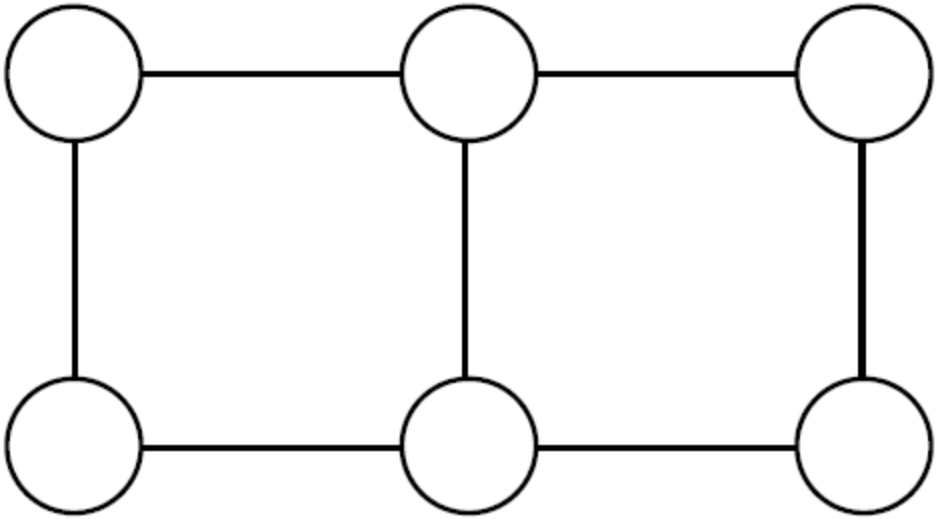
- Larranaga, P., C. M. H. Kuijpers, M. Poza, and R. H. Murga (1997). Decomposing bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing (UK) 7* (1), 19–34.
- Amir, E. (2010). Approximation Algorithms for Treewidth. *Algorithmica 56*(4), 448.
- Lipton, R. J. and R. E. Tarjan (1979). A separator theorem for planar graphs. *SIAM Journal of Applied Math 36*, 177–189.

Treewidth

- We define the treewidth as the size of the largest clique during (the best possible) elimination ordering minus one.



Treewidth=1



Treewidth=2

Inference in Undirected Graphs

- To obtain a single marginal, e.g., $p(x_1)$, define an elimination ordering in which x_1 is the final variable, and then normalize the result to calculate Z and obtain the marginal.

- In the directed case, a variable that is parentless has its marginal represented directly in the graph and no calculation is needed.

  Also, nodes that are downstream from a target node can simply be deleted, and *marginalization involves an inference calculation involving the ancestors of the node.* The worst case is a leaf node.

- In the undirected case, there is no notion of "ancestor," and essentially all nodes are worst case.

  Once *Z is calculated from a particular* elimination ordering (possibly one that yields small elimination cliques), it can be used to normalize other marginal probabilities.