Data analysis and stochastic modeling

Lecture 8 – Graphical modals and Bayesian networks

Guillaume Gravier

`guillaume.gravier@irisa.fr`

Joint probability, conditionals and marginals

In general, statistical learning is based on

$$\widehat{y} = \arg\max_y p(y|\mathbf{x}) = \arg\max_y p(\mathbf{x}|y)p(y)$$

which requires obtaining a model of either $p(\mathbf{x}|c)$ or $p(c|\mathbf{x})$, where $\mathbf{x}$ might be a complex collection of random variables.

Need for simple, tractable models $\Rightarrow$ assumptions must be made

naive Bayes	$p(x_1, \ldots, x_n|c) = \prod_i p(x_i|c)$
Markov property	$p(x_1, \ldots, x_n) = p(x_1) \prod_i p(x_i|x_{i-1})$
HMMs	$p(x_1, y_1, \ldots, x_n, y_n) = p(x_1)p(y_1|x_1) \prod_i p(x_i|x_{i-1})p(y_i|x_i)$

... to the expense of accuracy!

How can we handle more complex (and thus better) models ... without requiring a new theory everytime?

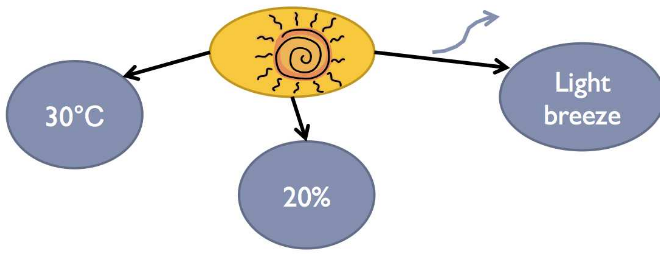
The key idea of graphical models

represent variable dependence (and thus independence) as a graph that enables factorization of the joint probability and graph-theoretic generic inference algorithms

A graphical model is a family of probability distributions defined in terms of a directed or undirected graph. The nodes in the graph are identified with random variables, and joint probability distributions are defined by taking products over functions defined on connected subsets of nodes.

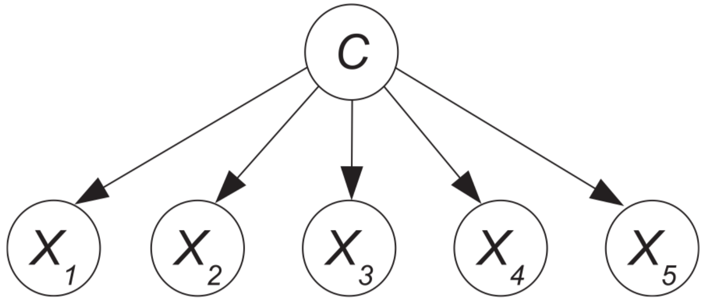
[Michael I. Jordan, *Graphical models*, Statistical Science, 2004]

$$p(\mathbf{x}, y) = p(y) \prod_{i=1}^{n} p(x_i|y)$$



Directed *vs.* Undirected

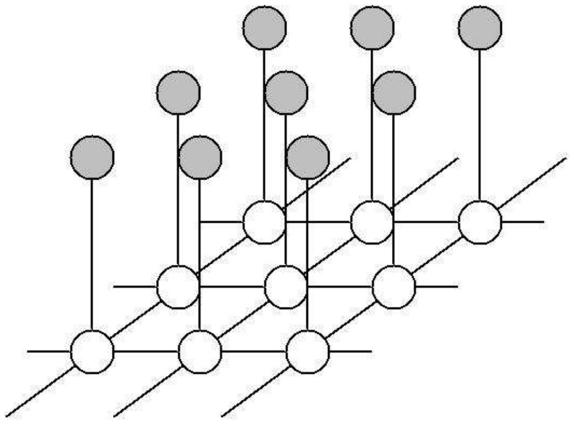
Directed graphs



$$p(\mathbf{x}) = \prod_{v \in \mathcal{V}} p(x_v | x_{\pi_v})$$

- used to encode “causal” relations
- graph must be acyclic (DAG)

Undirected graphs



$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c)$$

- used to encode spatial relations
- known as random fields

x_A represents the set $\{x_i \quad \forall i \in A\}$ and π_v the set of parent vertices of v

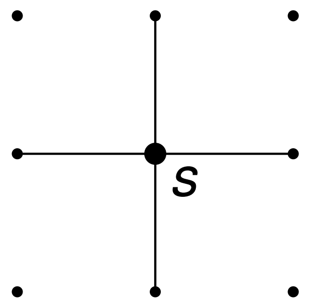
$\mathcal{V}$ is the set of vertices, $\mathcal{C}$ is the set of cliques on the graph

Undirected graph: Markov random fields

Markov property in random fields

$$P[X_s = x_s | X_{S\backslash s} = x_{S\backslash s}] = P[X_s = x_s | X_{V_s} = x_{V_s}]$$

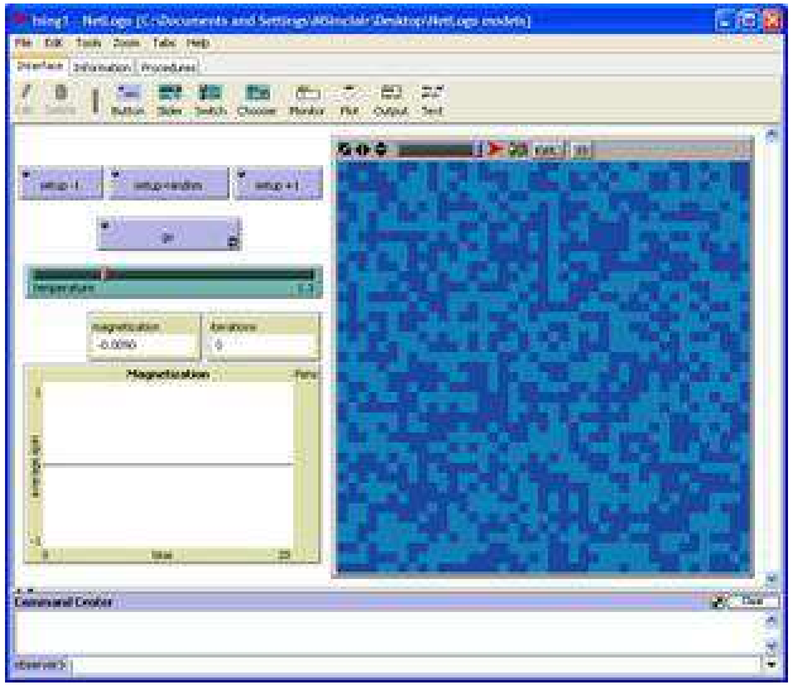
- $\forall s \in S \quad x_s \in E$
- sample space: $\Omega = E^{|S|}$
- neighborhood system: V
- set of cliques : $c \in \mathcal{C}$

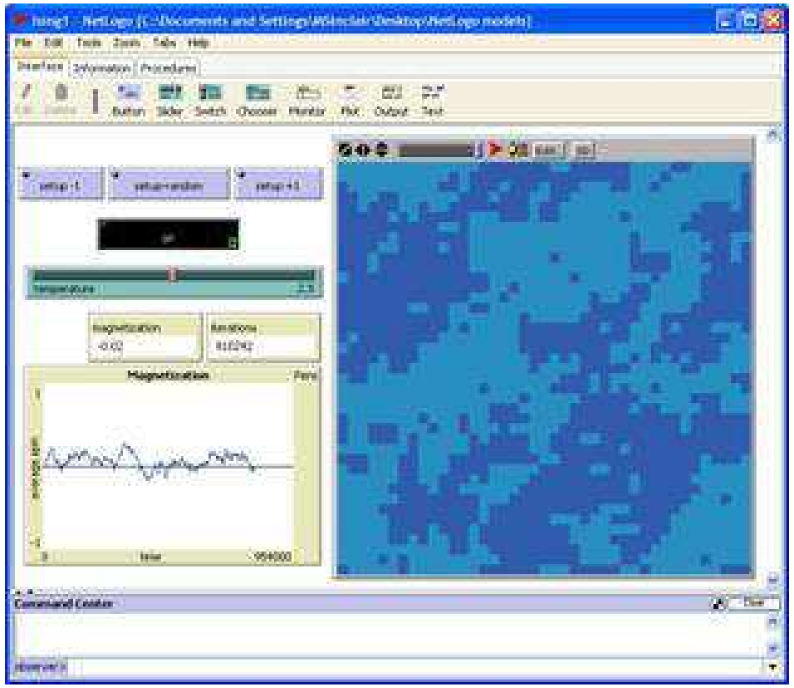


$$P[X = x] = \frac{1}{Z} \exp - \underbrace{\sum_{c \in \mathcal{C}} U_c(x)}_{U(x)} \qquad \text{with } Z = \sum_{x \in \Omega} \exp - U(x)$$

Markov random fields: an example

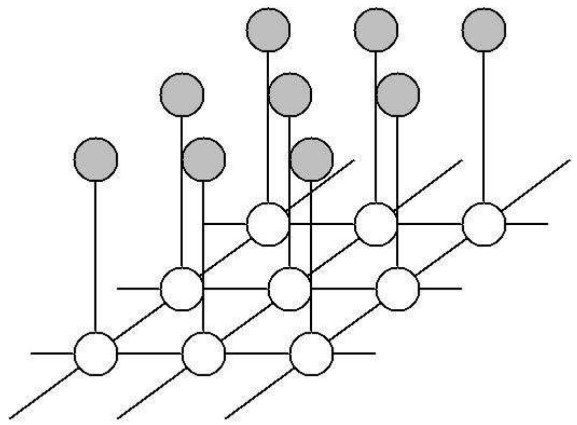
$$P[X_s|X_{V_s}] = \frac{\exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}{1+\exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}$$





Hidden Markov random fields

$X \longmapsto Y$, a noisy version of X



As we assumed conditional independence of the observations, we have

$$P[Y = y|X = x] = \prod_{s \in S} b_{x_s}(y_s) = \exp - \underbrace{\sum_{s \in S} - \ln b_{x_s}(y_s)}_{U(y|x)}$$

$P[X = x|Y = y]$: Gibbs distribution $U(x) + U(y|x)$

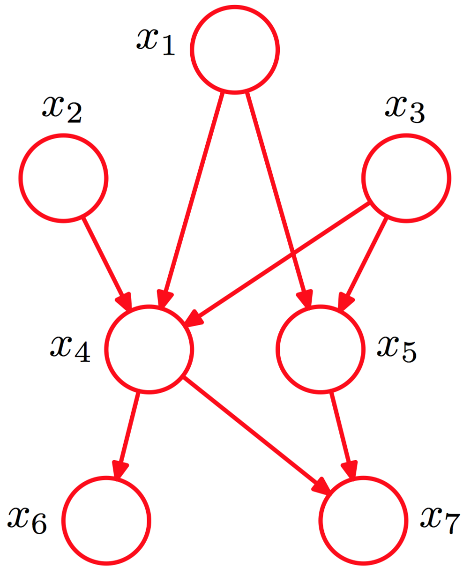
Directed graph: Bayesian networks

Bayesian network

=

directed acyclic graph over a collection of (discrete) random variables

$$\begin{aligned} P[X_1,\ldots,X_7] &= P[X_1]P[X_2|X_1]P[X_3|X_2,X_1]P[X_4|X_1,X_2,X_3] \\ &\quad P[X_5|X_1,\ldots,X_4]P[X_6|X_1,\ldots,X_5] \\ &\quad P[X_7|X_1,\ldots,X_6] \\ \\ &= P[X_1]P[X_2]P[X_3]P[X_4|X_1,X_2,X_3] \\ &\quad P[X_5|X_1,X_3]P[x_6|X_4]P[X_7|X_4,X_5] \end{aligned}$$



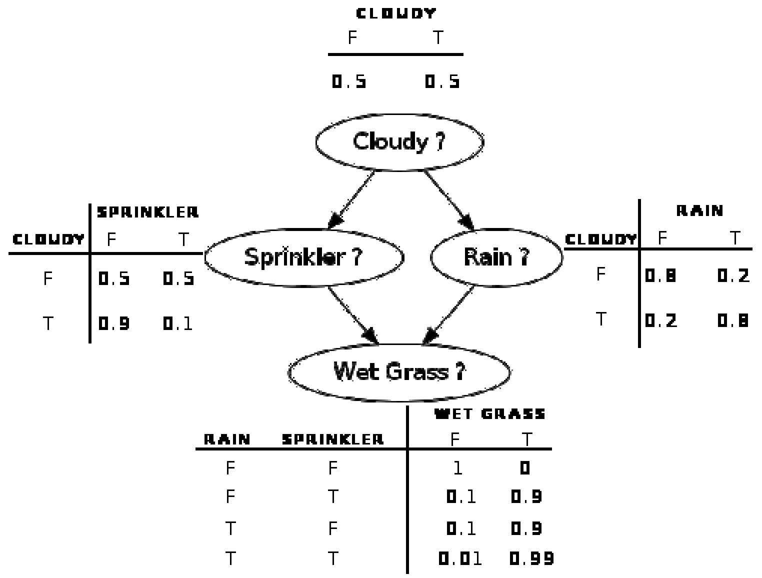
[from Bishop, 2006]

Interest:

- enables factorization of the joint likelihood
- enables the use of generic algorithms for inference

Bayesian networks: a parametric model

The sprinkler DAG encodes a family of distributions that verifies $P[C,S,R,W] = P[C]P[S|C]P[R|C]P[W|S,R]$ (we say the distribution *factorizes* on the graph), an instance of which is specified by the parameters in the figure.



CLOUDY F	CLOUDY T
0.5	0.5

CLOUDY	SPRINKLER F	SPRINKLER T
F	0.5	0.5
T	0.9	0.1

CLOUDY	RAIN F	RAIN T
F	0.8	0.2
T	0.2	0.8

RAIN	SPRINKLER	WET GRASS F	WET GRASS T
F	F	1	0
F	T	0.1	0.9
T	F	0.1	0.9
T	T	0.01	0.99

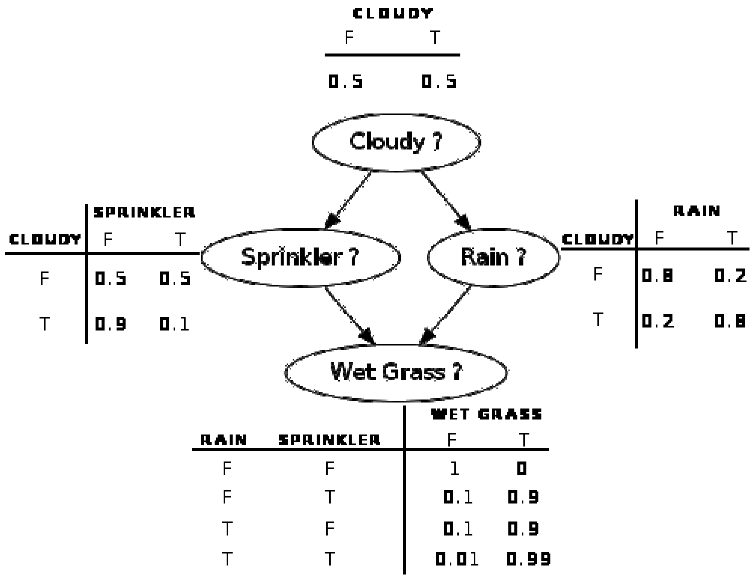
Factorization enables different types of reasoning:

- Bottom-up: given observations, what are the most likely causes
  $\rightarrow$ i.e., $P[S=1|W=1]$ and $P[R=1|W=1]$
- Top-down or causal: what is the probability of an event?
  $\rightarrow$ i.e., $P[W=1|C=1]$

Bottom-up *inference* in the spinkler example

$$P[S=1|W=1] = \frac{\sum_{c,r} P[S=1,W=1,C=c,R=r]}{P[W=1]} = 0.2781/0.6471$$

$$P[R=1|W=1] = \frac{\sum_{c,s} P[R=1,W=1,C=c,S=s]}{P[W=1]} = 0.4581/0.6471$$



CLOUDY F	CLOUDY T
0.5	0.5

CLOUDY	SPRINKLER F	SPRINKLER T
F	0.5	0.5
T	0.9	0.1

CLOUDY	RAIN F	RAIN T
F	0.8	0.2
T	0.2	0.8

RAIN	SPRINKLER	WET GRASS F	WET GRASS T
F	F	1	0
F	T	0.1	0.9
T	F	0.1	0.9
T	T	0.01	0.99

Rain is more likely to have caused wet grass than sprinkler! Interstingly, if we know it's raining, we have

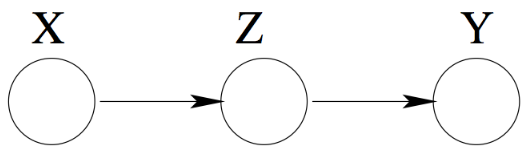
$$P[S=1|W=1,R=1] = \frac{\sum_{c} P[S=1,W=1,C=c,R=1]}{P[W=1,R=1]} = 0.1945$$

i.e., the probability that the sprinkler explains the wet grass diminishes (aka *explaining away* for two competing variables).

Basic (3 vertices) graphs and independence

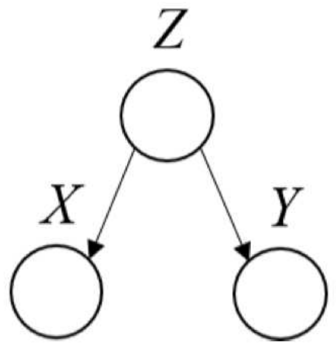
Chain graph

$$X \perp Y|Z \quad \text{and} \quad P[Y|Z,X] = P[Y|Z]$$



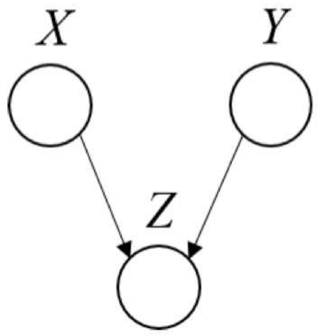
Latent cause graph

$$X \perp Y|Z \quad \text{and} \quad P[X,Y|Z] = P[X|Z]P[Y|Z]$$



Explain away graph (aka v-shape)

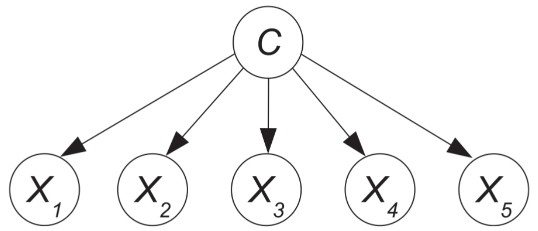
$$X \perp Y \quad \text{and} \quad P[X,Y] = P[X]P[Y]$$



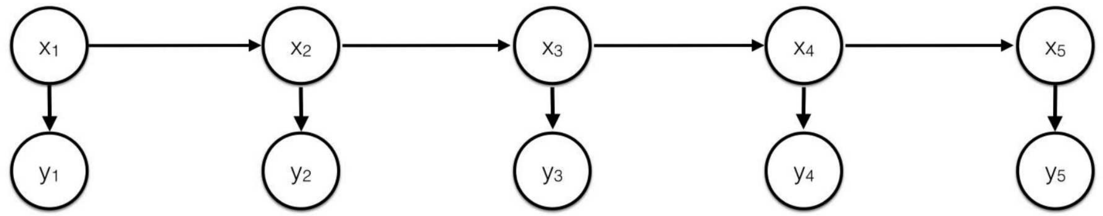
$X \perp Y|Z$ does not hold in the explain away graph!

Bayesian networks we've already seen

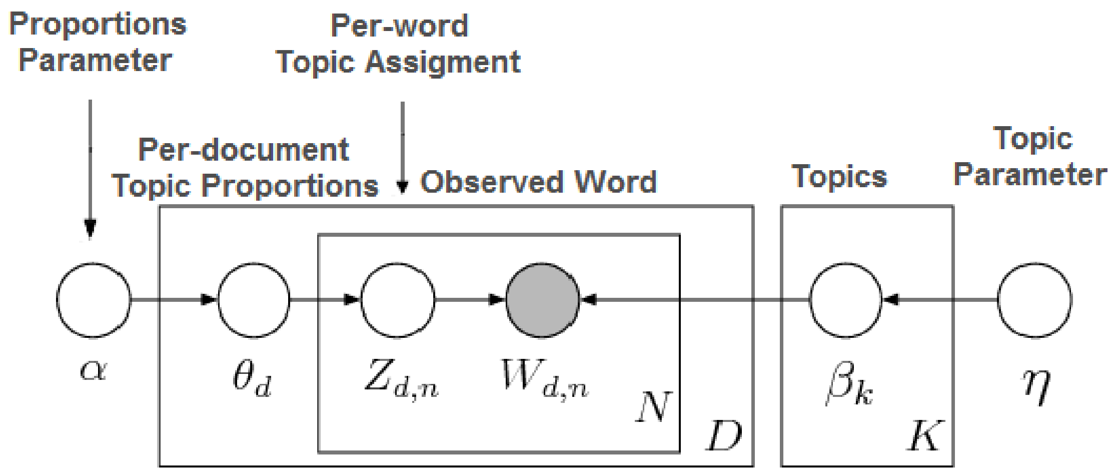
naive Bayes



HMM

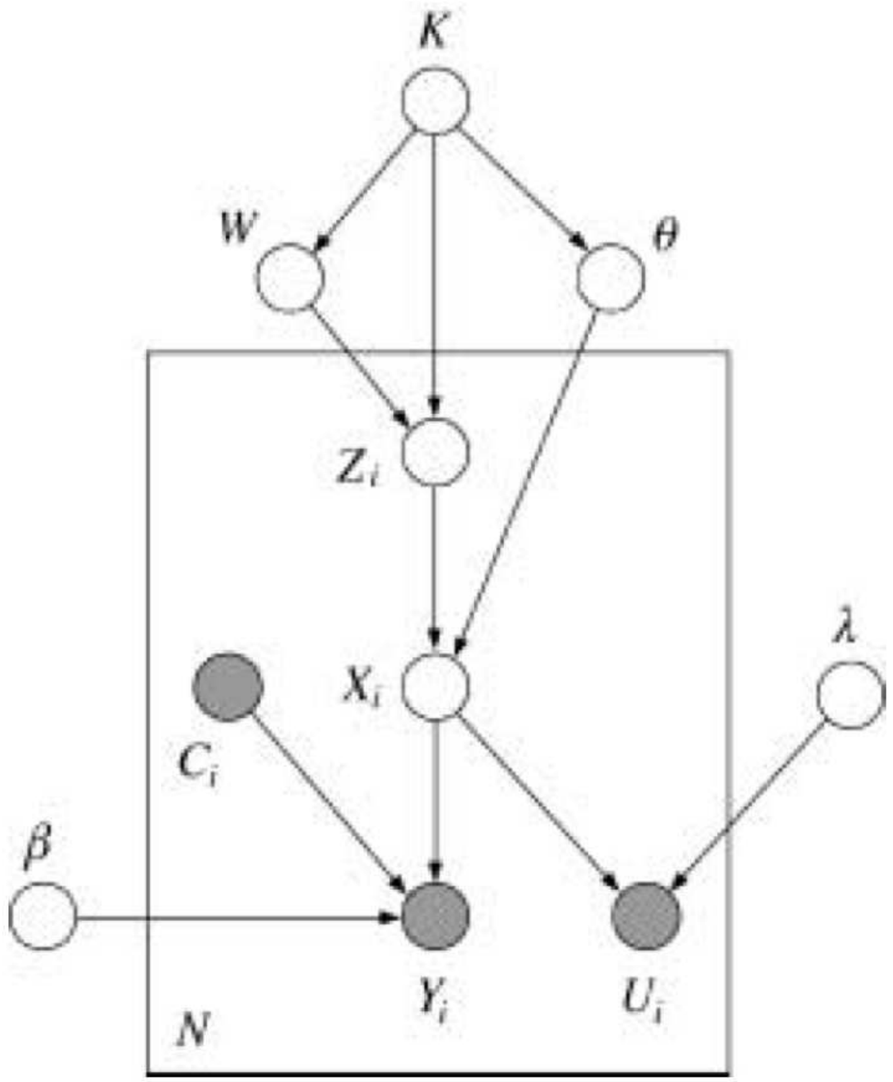


LDA



More complex networks

[...] The errors-in-covariates logistic regression model of Richardson, Leblond, Jaussent and Green (2002). The core of this model is a logistic regression of Y_i on X_i . The covariate X_i is not observed (in general), but noisy measurements U_i of X_i are available, as are additional observed covariates C_i. The density model for X_i is taken to be a mixture model, where K is the number of components, W are the mixing proportions, Z_i are the allocations and θ parameterizes the mixture components. [Michael I. Jordan, *Graphical models*, Statistical Science, 2004]



Common vision for directed/undirected graphs

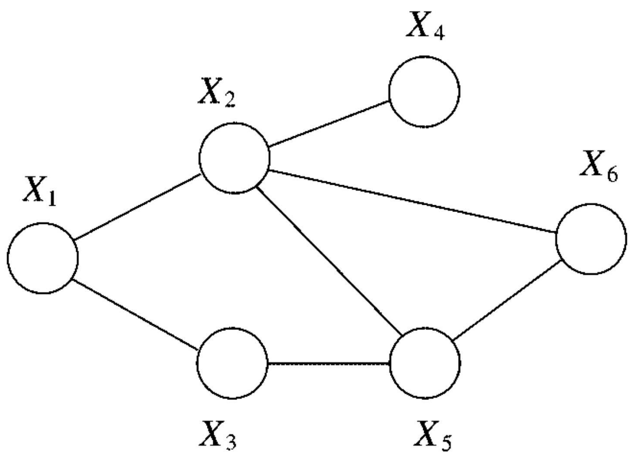
Directed graphs

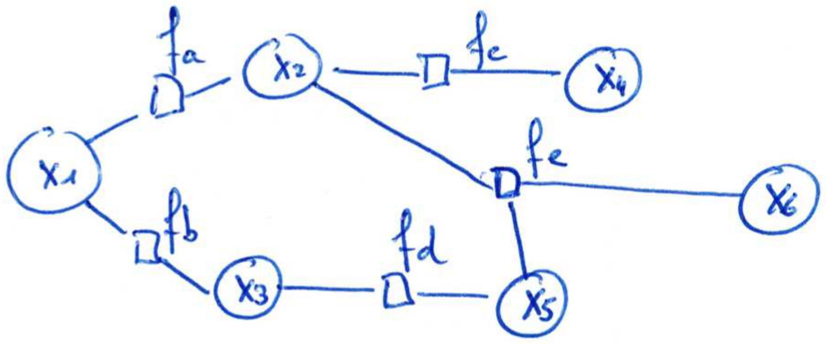
$$p(\mathbf{x}) = \prod_{v \in \mathcal{V}} p(x_v | x_{\pi_v})$$

Undirected graphs

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c)$$

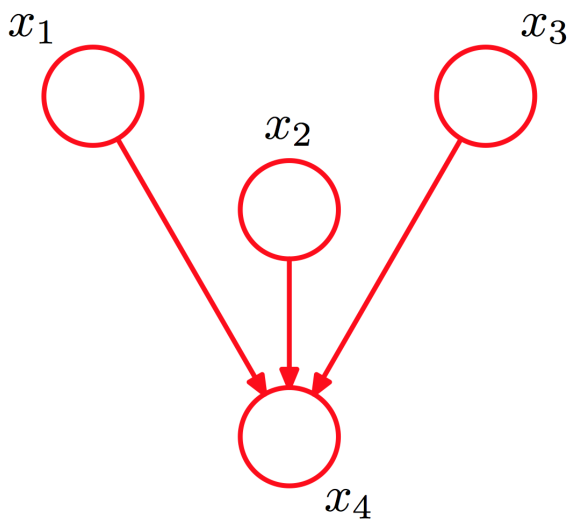
can all be cast into factor graphs: $p(\mathbf{x}) = \dfrac{1}{Z} \displaystyle\prod_{i \in \mathcal{I}} f_i(x_{C_i})$

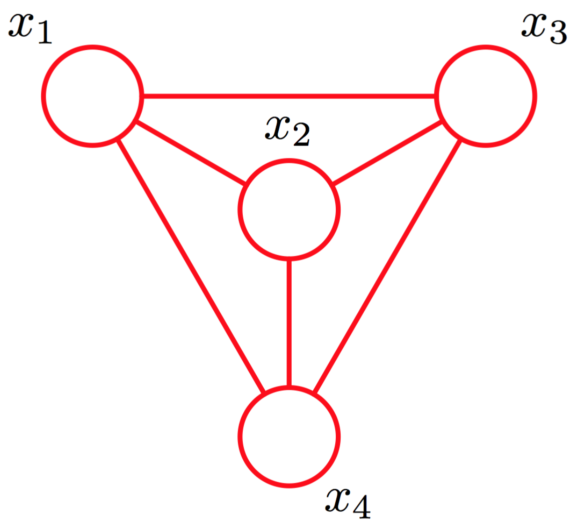




$$P[\mathbf{x}] = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_3) f_c(x_2, x_4) f_d(x_3, x_5) f_e(x_2, x_5, x_6)$$

Moralization of directed graphs





$$P[\mathbf{x}] = P[x_4|x_1, x_2, x_3]P[x_1]P[x_2]P[x_3] = \frac{1}{Z} f_a(x_1, x_2, x_3, x_4)$$

For chain graphs, moralization boils down to “removing” the direction of the edges

$$\begin{aligned} P[\mathbf{x}] &= P[x_1]P[x_2|x_1]P[x_3|x_2] \ldots P[x_n|x_{n-1}] \\ &= f_1(x_1) f_2(x_1, x_2) f_3(x_2, x_3) \ldots f_n(x_{n-1}, x_n) \end{aligned}$$

[From Chritopher M. Bishop, *Pattern Recognition and Machine Learning*, 2006]

Towards generic inference algorithms

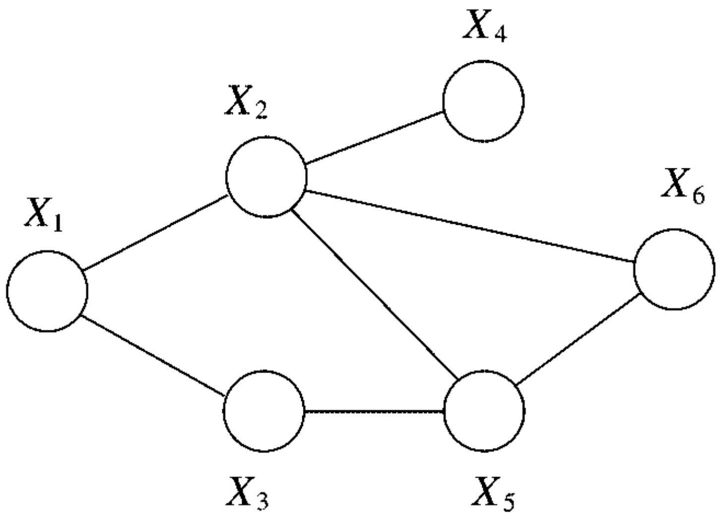
Inference = find posteriors for (all) unobserved variables given the observed ones

The idea is to define generic algorithms that take advantage of the (undirected) graph structure

Three families of approaches

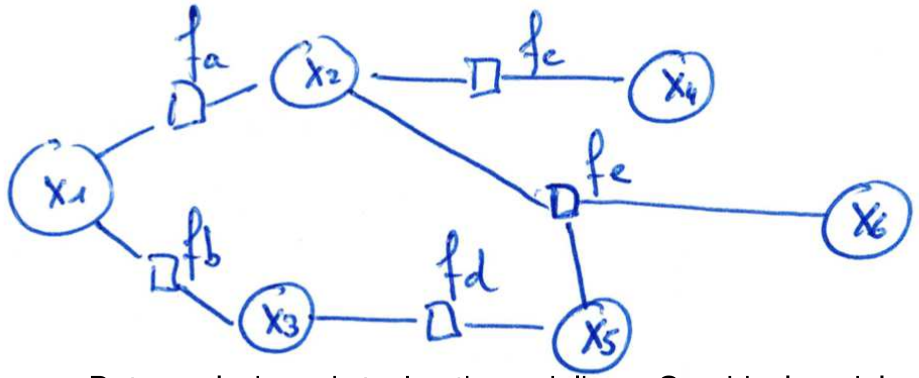
- exact inference algorithms
  - $\rightarrow$ works well for particular cases or fairly simple structures
- Monte Carlo approaches
  - $\rightarrow$ use sampling to compute posteriors and marginalization
  - $\rightarrow$ Gibbs sampling commonly used in this case
- Variational methods
  - $\rightarrow$ make simplifications of the model to get an approximate solution

The variable elimination principle

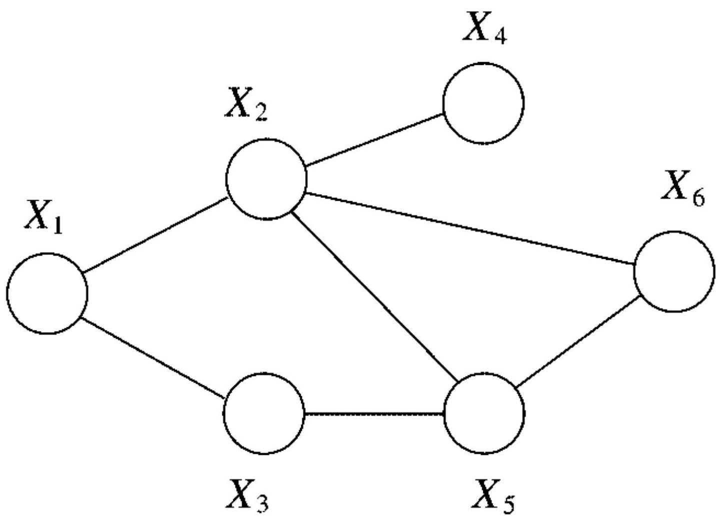


factorizing according to

$$P[\mathbf{x}] = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_3) f_c(x_2, x_4) f_d(x_3, x_5) f_e(x_2, x_5, x_6)$$



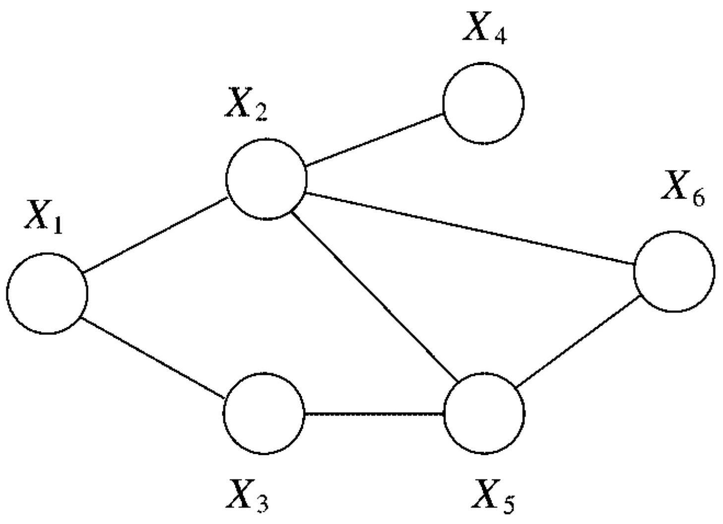
The variable elimination principle (cont'd)



$$\begin{aligned}
P[x_1] &\propto \sum_{x_2} \dots \sum_{x_6} f_a(x_1,x_2) f_b(x_1,x_3) f_c(x_2,x_4) f_d(x_3,x_5) f_e(x_2,x_5,x_6) \\
&\propto \sum_{x_2} f_a(x_1,x_2) \sum_{x_3} f_b(x_1,x_3) \sum_{x_4} f_c(x_2,x_4) \sum_{x_5} f_d(x_3,x_5) \underbrace{\sum_{x_6} f_e(x_2,x_5,x_6)}_{m_6(x_2,x_5)} \\
&\propto \sum_{x_2} f_a(x_1,x_2) \sum_{x_3} f_b(x_1,x_3) \sum_{x_4} f_c(x_2,x_4) \sum_{x_5} f_d(x_3,x_5) m_6(x_2,x_5)
\end{aligned}$$

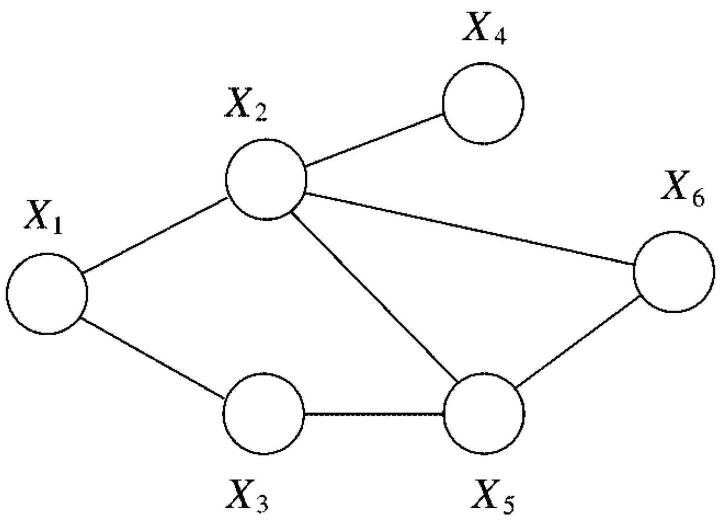
UNIVERSITÉ DE RENNES 1

The variable elimination principle (cont'd)



$$
\begin{aligned}
P[x_1] &\propto \sum_{x_2} f_a(x_1,x_2) \sum_{x_3} f_b(x_1,x_3) \sum_{x_4} f_c(x_2,x_4) \underbrace{\sum_{x_5} f_d(x_3,x_5) m_6(x_2,x_5)}_{m_5(x_2,x_3)} \\
&\propto \sum_{x_2} f_a(x_1,x_2) \sum_{x_3} f_b(x_1,x_3) \sum_{x_4} f_c(x_2,x_4) \underbrace{\sum_{x_5} f_d(x_3,x_5) m_6(x_2,x_5)}_{m_5(x_2,x_3)} \\
&\propto \sum_{x_2} f_a(x_1,x_2) \sum_{x_3} f_b(x_1,x_3) \sum_{x_4} f_c(x_2,x_4) m_5(x_2,x_3)
\end{aligned}
$$

The variable elimination principle (cont'd)



$$
\begin{aligned}
P[x_1] &\propto \sum_{x_2} f_a(x_1,x_2) \underbrace{\sum_{x_4} f_c(x_2,x_4)}_{m_4(x_2)} \sum_{x_3} f_b(x_1,x_3) m_5(x_2,x_3) \\
&\propto \sum_{x_2} f_a(x_1,x_2) m_4(x_2) \underbrace{\sum_{x_3} f_b(x_1,x_3) m_5(x_2,x_3)}_{m_3(x_1,x_2)} \\
&\propto \sum_{x_2} f_a(x_1,x_2) m_4(x_2) m_3(x_1,x_2) \quad \propto \mathbf{m_2(x_1)}
\end{aligned}
$$

UNIVERSITÉ DE RENNES 1

The variable elimination principle (almost last)

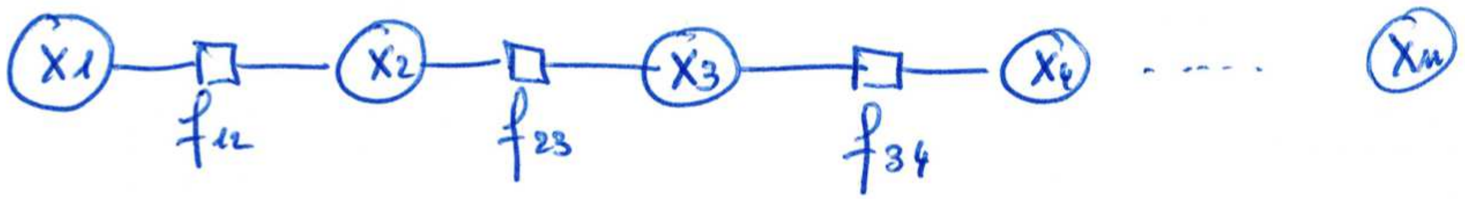
To sum up, the idea is to progressively "eliminate variables" by iteratively making partial sums.

The iterative process breaks down the complexity but key questions/issues remain:

- choosing an adequate elimination order is far from trivial in general
  - it's in fact known as a NP-hard problem for general graphs!
- need to run a specific elimination procedure foreach variable
  - that's bad!

If variables are observed, the algorithm remains valid by replacing the corresponding sum by a single value

Variable elimination on chain graphs



$$P[\mathbf{x}] \propto f_{1,2}(x_1, x_2) f_{2,3}(x_2, x_3) f_{3,4}(x_3, x_4) \ldots f_{n-1,n}(x_{n-1}, x_n) \ .$$

To compute efficiently compute $P[x_i] = \sum_{x_1} \ldots \sum_{x_{i-1}} \sum_{x_{i+1}} \ldots \sum_{x_n} P[\mathbf{x}]$, we can take advantage of the fact that the summation on

- x_n only depends on $f_{n-1,n}(x_{n-1}, x_n) = \beta_n(x_{n-1})$
- x_{n-1} only depends on $f_{n-2,n-1}(x_{n-2}, x_{n-1})$ and $\beta_n(x_{n-1})$

or, similarly, that the summation on

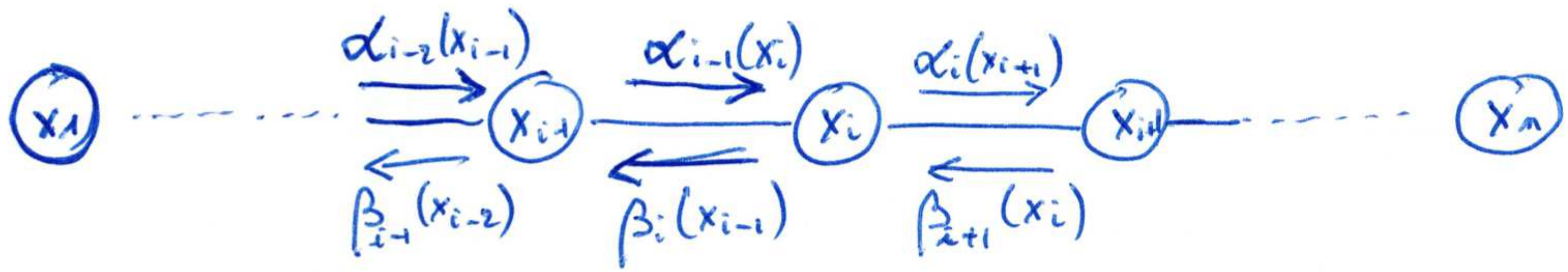
- x_1 only depends on $f_{1,2}(x_1, x_2) = \alpha_1(x_2)$
- x_2 only depends on $f_{2,3}(x_2, x_3)$ and $\alpha_1(x_2)$

Variable elimination and message passing

$$\sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} P[\mathbf{x}] \propto$$

$$\underbrace{\left(\sum_{x_{i-1}} f_{i-1,i}(x_{i-1}, x_i) \ldots \left(\sum_{x_2} f_{2,3}(x_2, x_3) \left(\sum_{x_1} f_{1,2}(x_1, x_2) \right) \right) \ldots \right)}_{\alpha_{i-1}(x_i)}$$

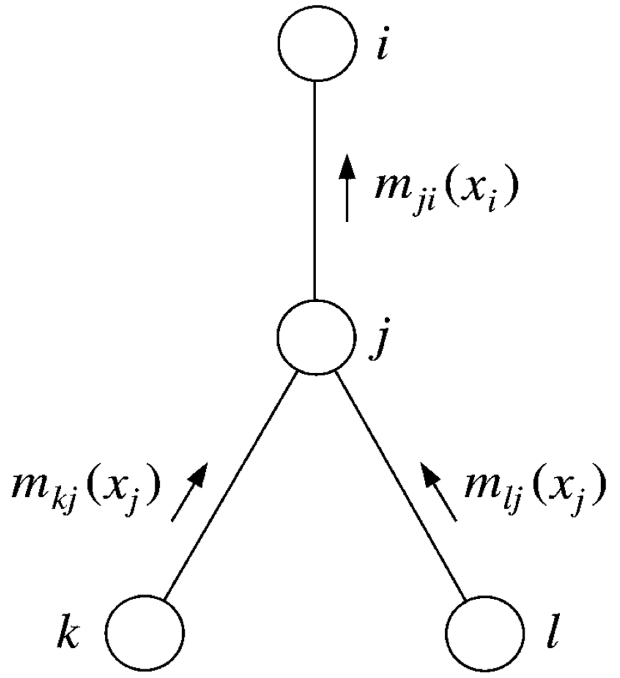
$$\underbrace{\left(\sum_{x_{i+1}} f_{i,i+1}(x_i, x_{i+1}) \ldots \left(\sum_{x_{n-1}} f_{n-2,n-1}(x_{n-2}, x_{n-1}) \left(\sum_{x_n} f_{n-1,n}(x_{n-1}, x_n) \right) \right) \ldots \right)}_{\beta_{i+1}(x_i)}$$

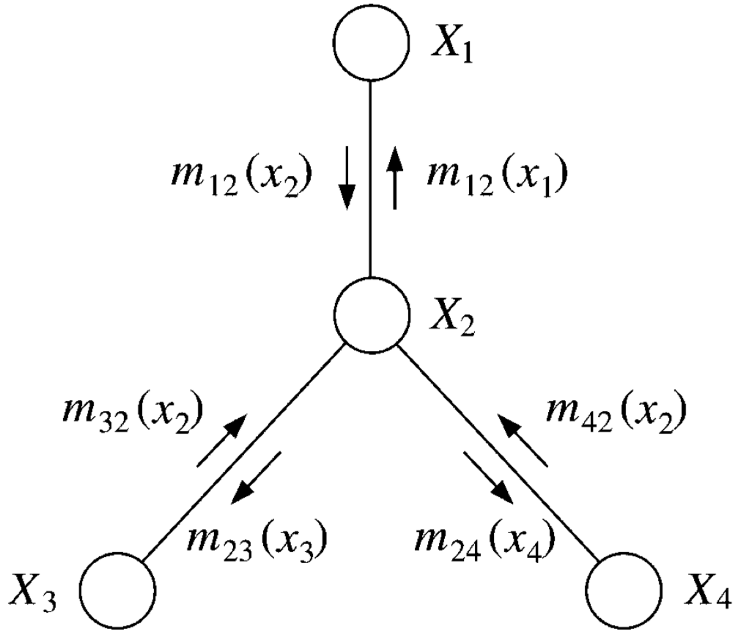


$$\alpha_{i-1}(x_i) = \sum_{x_{i-1}} f_{i-1,i}(x_{i-1}, x_i)\alpha_{i-2}(x_{i-1}) \qquad \beta_i(x_{i-1}) = \sum_{x_i} f_{i-1,i}(x_{i-1}, x_i))\beta_{i+1}(x_i)$$

Generalization to trees (sum-product)

The message passing principle illustrated with chain graphs straightforwardly generalizes to trees (where a node has a single parent) and is know as the *sum-product* algorithm





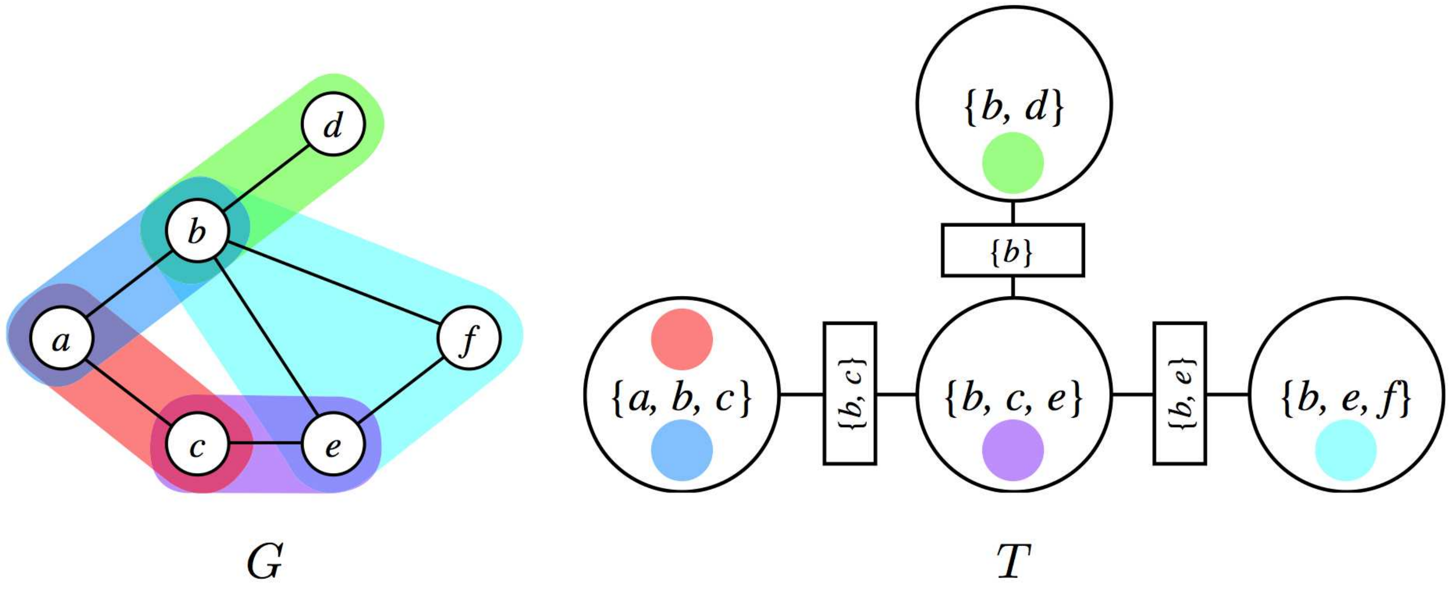
Many variants of the sum-product principle do exist like

- replace *sum* with *max* to get the most likely configuration
- exploit the bipartite factor graph

Generalization with junction trees

A *junction tree* $\mathcal{T}$ for a graph $\mathcal{G}$ is a tree where nodes are clusters of nodes from $\mathcal{G}$ related to the cliques in $\mathcal{G}$ and that verifies the following properties:

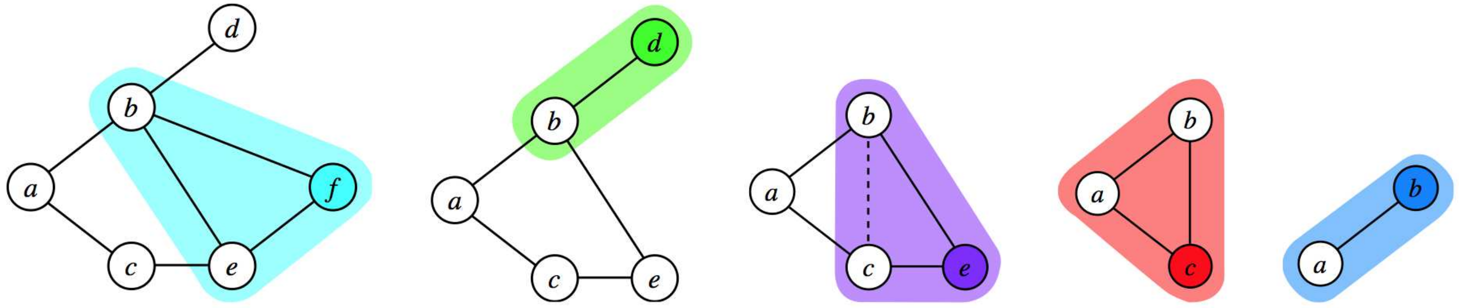
- only one path between each pair of clusters (it's a tree!)
- each clique in $\mathcal{G}$ is included in a cluster
- for each pair of cluster A and B that contains i, each cluster on the (unique) path between A and B contains i



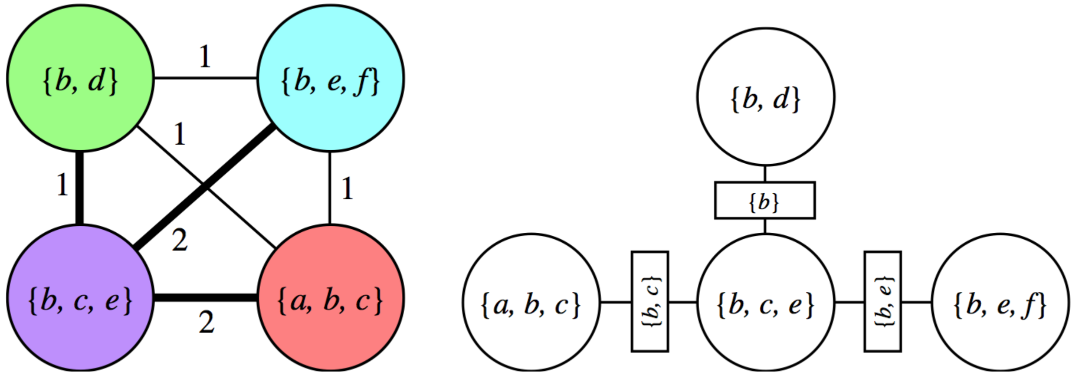
[borrowed from Mark Paskin's course]

Building junction trees

1. choose a node elimination order
2. run node elimination to get the set of elimination cliques



3. build a complete graph over the elimination maximal cliques
4. weight edge $A \to B$ with $|A \cup B|$
5. build maximum weight spanning tree



Inference with the junction tree algorithm

apply sum-product on the junction tree of $\mathcal{G}$ to perform inference on $\mathcal{G}$

The junction tree inference algorithm boils down to

1. compute moral graph (if input is directed) – will change topology!
2. perform graph triangulation to remove chord-less cycles – will change topology!
3. create junction tree on triangulated graph
4. run sum-product or max-product message passing on the junction tree
5. compute marginals on $\mathcal{G}$ from message passing on the junction tree

Generic and efficient algorithm for reasonably well-formed Bayes nets or graphs but rapidly becomes intractable when graph topology gets complex (because width of junction tree explodes)!

A quick word in estimation

Parameter estimation

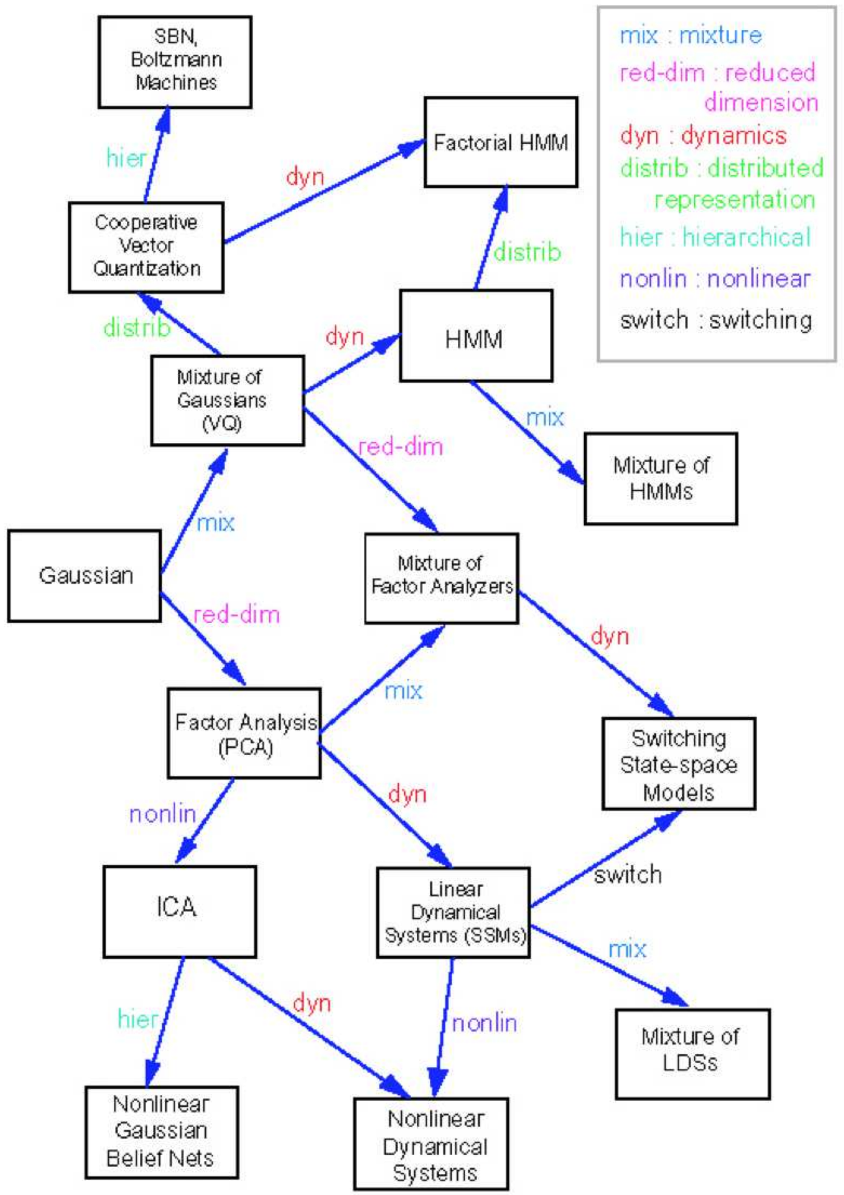
- ◦ Standard maximum likelihood approaches for complete data
- ◦ Possible maximum a posteriori regularization
- ◦ EM and EM-like algorithms for incomplete data

  $\longrightarrow$ E-step benefits from sum-product algorithm!

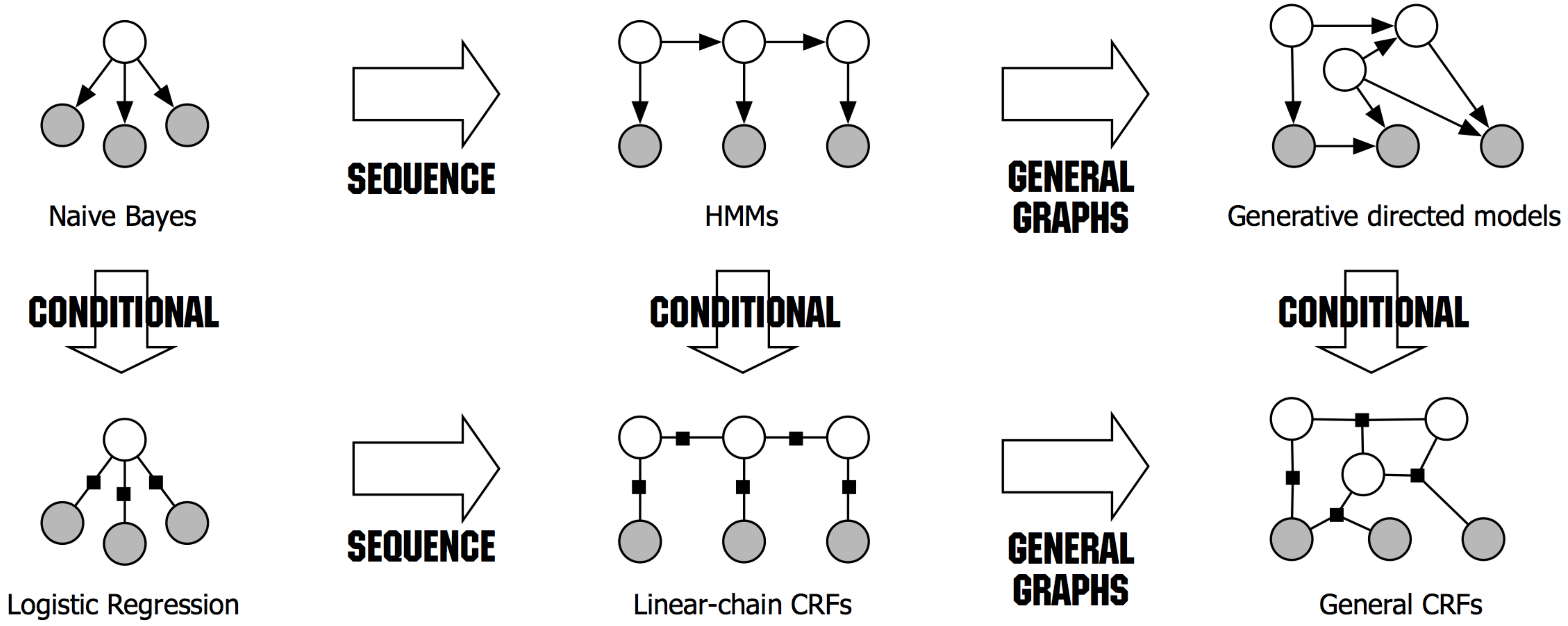
Model selection: a large body of work on learning the BN structure

- ◦ Bayesian information criteria
- ◦ Augmented networks: tree augmented, forest augmented, etc.
- ◦ The K2 algorithm
- ◦ etc.

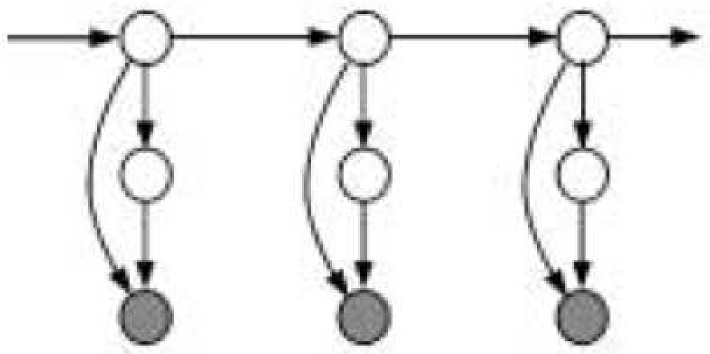
Revisiting the bestiary of models

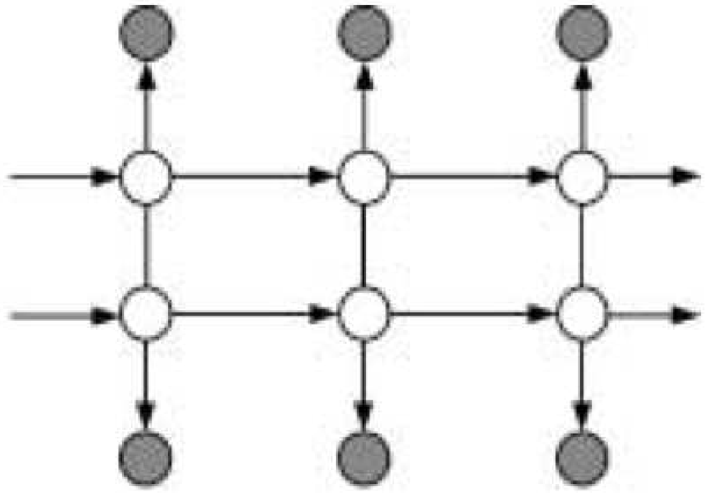


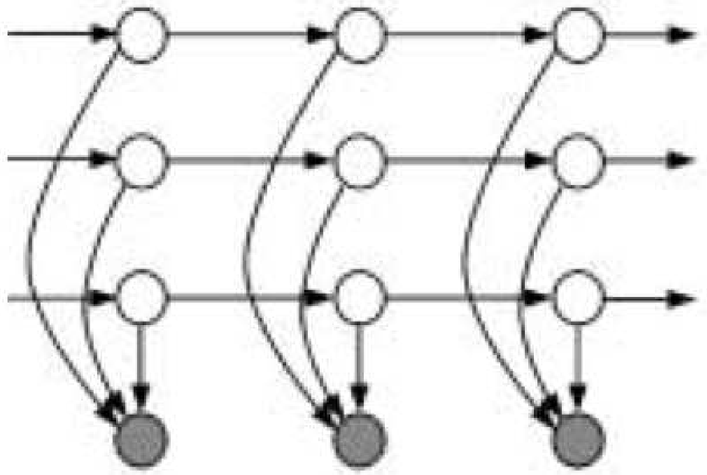
Revisiting the bestiary of models

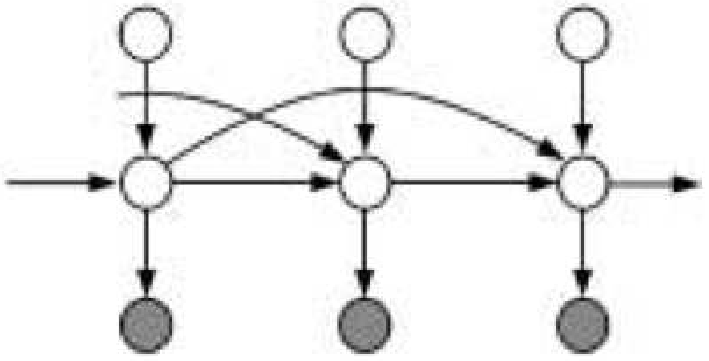


Bayesian network extensions of HMMs


(a)


(b)


(c)


(d)

Key references

Full books

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

Michael I Jordan. *Learning in graphical models*. MIT Press, 1999.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)* – Chap. 8: Graphical models. Springer-Verlag New York, 2006.

Overview journal papers

Michael I. Jordan. Graphical models. Statistical Science, 19(1):140–155, 2004.

Concha Bielza and Pedro Larragaña. Discrete Bayesian network classifiers: a survey. Computing Surveys, 47(1), 2014.