Introduction to Graphical Models and its Applications

Hsing-Kuo Kenneth Pao, Ph.D.

National Taiwan University of Science and Technology

Outline

- Introduction to graphical models
- Directed and undirected models
- Probabilistic inference
- Building the models
- Applications in text mining, bioinformatics, and computer vision, etc.

Introduction to Graphical Models, Directed and Undirected Models

Probabilistic Graphical Models

- A way to summarize a model, a distribution (family) or a knowledge system
- Trying the **divide and conquer** approach
- Ingredient: graph theory and probability theory
- More formally,
  - Graph structure encodes parts of system that are directly dependent or the set of conditional dependence/independence assumptions
  - Local functions specify how parts interact or factors in the joint probability distribution
- E.g. 1: Bayesian networks = PGMs based on DAGs
- E.g. 2: Markov networks (Markov random fields) = PGMs with undirected graphs
- Others: Discrete models, continuous models, Gaussian graphical models, hybrid Bayesian networks, etc

Probabilistic Graphical Models (con't)

- In structure, more general than neural networks, finite state machines
- In structure, different from the social networks, random graphs
  - We may not need "identity" in each node of the social networks or random graphs
  - They do share some common things: e.g., belief propagation
- In ability, explainable for other skills including Hidden Markov Model, dimension reduction, etc.
  - Forward-backward algorithm $\rightarrow$ node elimination
  - Finding hidden states for mixture models $\rightarrow$ probabilistic inference
- In ability, more powerful in structural model learning, more powerful for some computation, e.g., diffusion/random walk

Why Probability?

- Why probabilistic graphical models?
- Why probabilistic approach can help us in machine learning or understanding?
- *Guess* 1: world is nondeterministic or
- *Guess* 2: world is deterministic, but lacking of relevant facts/attributes to decide the answer

$$P(x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n; y)$$

$$\boxed{\epsilon(h) = \int_{(\mathbf{x},y)} C(h(\mathbf{x}), y) dp(\mathbf{x}, y)}$$

$\Rightarrow$ probabilistic models can make inference with the missing inputs

$\Rightarrow$ probabilistic models can make inference about the missing inputs: classification is one example (missing is “y”)!

- Either way, probabilistic approach can make decisions which minimize expected loss

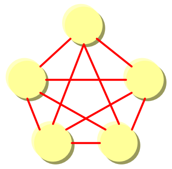
Why Graphical Models?

- We discuss this issue along the way…
- Basically, graphical models help us to deal with complex problem modularized and in a visualized way

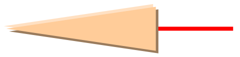
  $\Rightarrow$ e.g., some independence rule can be read from the graph structure or by some simple calculations
- That is, breaking problems to subproblems (recursively) until we can solve them

Between Simple and Complex Models

- Given n r.v.'s $X_1, X_2, \ldots, X_n$, with state space = $\{1, 2, 3\}$, let us consider the degree of freedom (or complexity) of the model to describe the joint probabilities
- most general
  - df = $3^n - 1$
  - not efficient in time and space

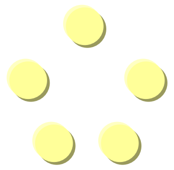


- ???



graphical models

- independent
  - df = $2n$ ($P(X_i=1)$, $P(X_i=2)$)



- i.i.d.
  - df = 2 ($P(X_1=1)$, $P(X_1=2)$)

Factorable Joint Probability

- *Goal 1*: represent a joint distribution $P(\mathbf{X} = \mathbf{x}) = P(X_1=x_1, \ldots, X_n=x_n)$ compactly (independent of n or « n) even when there are many variables
- *Goal 2*: efficiently calculate marginals and conditionals of such compactly represented joint distribution
- The independence assumption is too restrictive. So we make conditional independence assumption instead!

Directed Models: Bayesian Networks

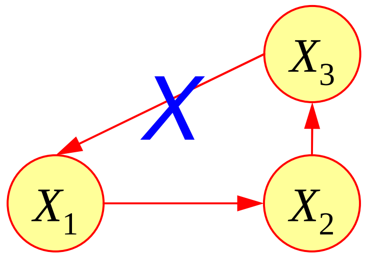
- A.K.A. belief network, directed graphical model
- Definition
  - a set of nodes representing (random) variables and
  - a set of directed edges between variables
  - each variable has a finite set of mutually exclusive states
  - to each variable X with parents $Y_1, Y_2, \ldots, Y_n$, there is attached the conditional probability table $P(X \mid Y_1, Y_2, \ldots, Y_n)$
- Informally, edges represent "causality"
- The variables together with the directed edges form a Directed Acyclic Graph (no cycles allowed)

$$\begin{tikzcd} X & Y \end{tikzcd}$$

$p(X, Y) = p(X)\, p(Y)$

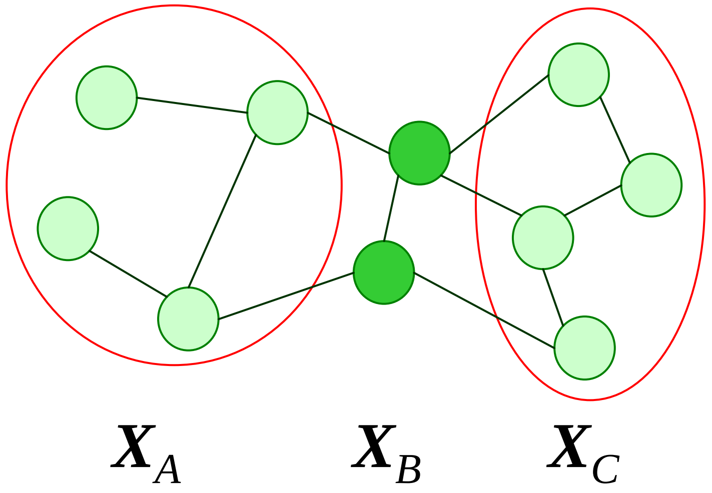
$$\begin{tikzcd} X \arrow[r] & Y \end{tikzcd}$$

$p(X, Y) = p(X)\, p(Y \mid X)$



Undirected Graphical Models: Markov Networks

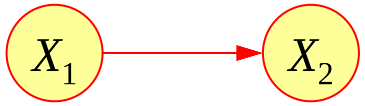
- A.K.A Markov Random Fields
- Also graphs with one node per random variable and edges that connect pairs of nodes, but now the edges are undirected
- Semantics: every node set is conditionally independent from its non-neighbors given its neighbors, i.e. $\boldsymbol{X}_A \perp \boldsymbol{X}_C \mid \boldsymbol{X}_B$ if every path between $\boldsymbol{X}_A$ and $\boldsymbol{X}_C$ goes through $\boldsymbol{X}_B$
- Can model symmetric interactions that directed models cannot!



Common Misunderstandings

- Directions in Bayesian network represent something about *causality*? **No**

$D = (x_1, x_2) = \{(0, 0), (1, 0), (1, 1)\}$



- Bayesian network can represent all kinds of distributions? **No**
- Markov random field can represent all kinds of distribution? **No**
- The network representation for a given distribution is unique? **No**
- Bayesian network is easier than Markov random field, or the other way around?
- How about the distribution is strictly positive or not?

Factorization

- Consider *directed acyclic graphs* over n variables.
- Each node has (possibly empty) set of parents π_i.
- Each node maintains a function $f_i(X_i; \mathbf{X}_{\pi_i})$ such that

$$f_i > 0 \text{ and } \sum_{x_i} f_i(X_i = x_i; \mathbf{X}_{\pi_i}) = 1 \ \forall \pi_i$$

- Define the joint probability to be:

$$P(X_1, X_2, \ldots, X_n) = \prod_i f_i(X_i; \mathbf{X}_{\pi_i})$$

- Even with no further restriction on the the f_i, it is always true that

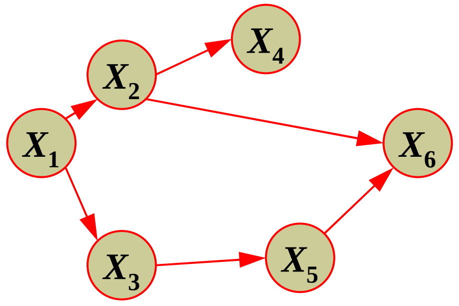
$$f_i(X_i; \mathbf{X}_{\pi_i}) = P(X_i | \mathbf{X}_{\pi_i})$$

- so we will just write

$$P(X_1, X_2, \ldots, X_n) = \prod_i P(X_i | \mathbf{X}_{\pi_i})$$

- Factorization of the joint in terms of local conditional probabilities. Exponential in “fan-in” of each node instead of in total variables n.

So by the Alternative Definition…



$$P(X_1,X_2,X_3,X_4,X_5,X_6)=\prod_i f_i(X_i|\mathbf{X}_{\pi_i})$$

$$\begin{aligned}&P(X_1,X_2,X_3,X_4,X_5,X_6)\\ =\;&P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3)P(X_6|X_2,X_5)\end{aligned}$$

$$\begin{aligned}P(X_2,X_5,X_6) &= \sum_{X_1,X_3,X_4} f_1(X_1)f_2(X_2;X_1)f_3(X_3;X_1)f_4(X_4;X_2)f_5(X_5;X_3)f_6(X_6;X_2,X_5)\\ &= f_6(X_6;X_2,X_5)\sum_{X_1,X_3,X_4} f_1(X_1)f_2(X_2;X_1)f_3(X_3;X_1)f_4(X_4;X_2)f_5(X_5;X_3)\end{aligned}$$

$$\begin{aligned}P(X_2,X_5) &= \sum_{X_1,X_3,X_4} f_1(X_1)f_2(X_2;X_1)f_3(X_3;X_1)f_4(X_4;X_2)f_5(X_5;X_3)\sum_{X_6} f_6(X_6;X_2,X_5)\\ &= \sum_{X_1,X_3,X_4} f_1(X_1)f_2(X_2;X_1)f_3(X_3;X_1)f_4(X_4;X_2)f_5(X_5;X_3)\end{aligned}$$

$$\Rightarrow P(X_6|X_2,X_5)=P(X_2,X_5,X_6)/P(X_2,X_5)=f_6(X_6;X_2,X_5)$$

Conditional Independence

- Notation: $\boldsymbol{X}_A \perp \boldsymbol{X}_B \mid \boldsymbol{X}_C$

  Definition: two (sets of) variables $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are conditionally independent given a third $\boldsymbol{X}_C$ if:

  $$P(\boldsymbol{X}_A , \boldsymbol{X}_B \mid \boldsymbol{X}_C) = P(\boldsymbol{X}_A \mid \boldsymbol{X}_C)\, P(\boldsymbol{X}_B \mid \boldsymbol{X}_C) \qquad \forall \boldsymbol{X}_C = \boldsymbol{x}_C$$

  which is equivalent to saying

  $$P(\boldsymbol{X}_A \mid \boldsymbol{X}_B , \boldsymbol{X}_C) = P(\boldsymbol{X}_A \mid \boldsymbol{X}_C) \qquad \forall \boldsymbol{X}_C = \boldsymbol{x}_C$$

- Only a subset of all distribution respect any given (nontrivial) conditional independence statement. The subset of distributions that respect all the CI assumptions we make is the family of distributions consistent with our assumptions

$$G \cong \text{set of C.I. rules} \cong \text{family of distributions}$$

- Probabilistic graphical models are a powerful, elegant and simple way to specify such a family

Conditional Independence in DAGs

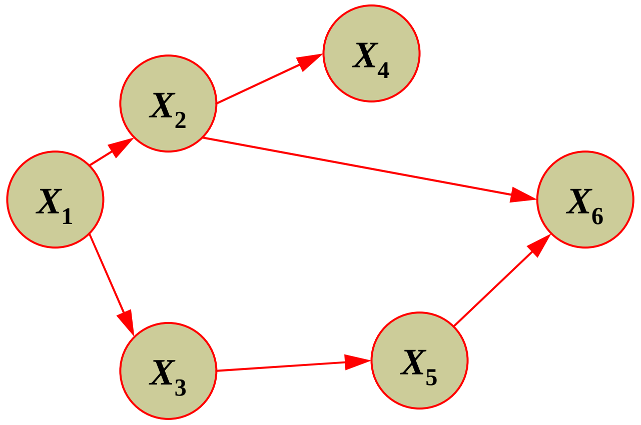
- If we order the nodes in a directed graphical model so that parents always come before their children in the ordering then the graphical model implies the following about the distribution:

$$\{X_i \perp \mathbf{X}_{\tilde{\pi}_i} | \mathbf{X}_{\pi_i}\} \ \forall i$$

where $\mathbf{X}_{\tilde{\pi}_i}$ are the nodes coming before X_i that are not its parents.

- In other words, the DAG is telling us that each variable is conditionally independent of its non-descendants given its parents, this is called local Markov property
- Such an ordering is called a "topological" ordering

From the Definition!



Prove: $X_4 \perp X_1 \mid X_2$

$$\begin{aligned} P(X_{[1,2,4]}) &= P(X_1)P(X_2|X_1)P(X_4|X_2)\sum_{X_3} P(X_3|X_1)\sum_{X_5} P(X_5|X_3)\sum_{X_6} P(X_6|X_2,X_5) \\ &= P(X_1)P(X_2|X_1)P(X_4|X_2) \end{aligned}$$

$$\begin{aligned} P(X_{[1,2]}) &= P(X_1)P(X_2|X_1)\sum_{X_3} P(X_3|X_1)\sum_{X_4} P(X_4|X_2)\sum_{X_5} P(X_5|X_3)\sum_{X_6} P(X_6|X_2,X_5) \\ &= P(X_1)P(X_2|X_1) \end{aligned}$$

$\Rightarrow P(X_4|X_1,X_2) = P(X_4|X_2)$

Bayes Rule

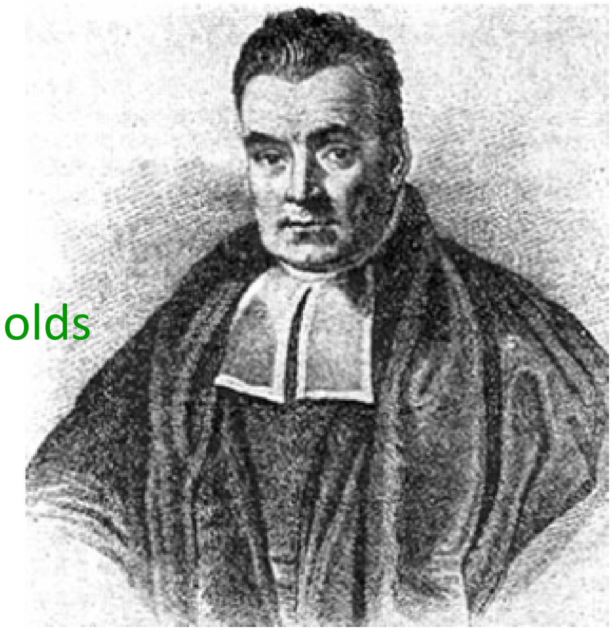
$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- **Thomas Bayes (1763)** “An essay towards solving a problem in the doctrine of chances”. *Philosophical Transactions of the Royal Society of London,* **53**: pp. 370-418.
- Best h in H given D
  - best = most probable
  - BT: direct method of computing it
- Notations
  - $P(h)$: prob. h (hypothesis) holds
  - $P(D)$: prob. D (data) is
  - $\mathcal{L}(D \mid h) = P(D \mid h)$: D observed when h holds
  - $P(h \mid D)$: h holds when D observed (!)

prior probability

likehood

posterior probability



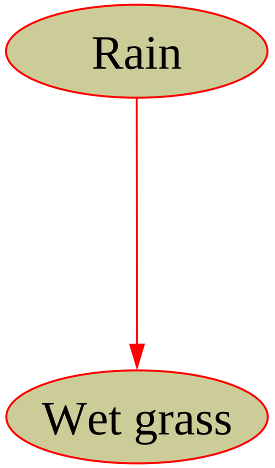
1:30 上午

Bayes Rule (cont.)

- Bayes rule: $$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

  the realm of density estimation
- Generally, best h maximizes $P(h \mid D)$
  - MAP: *Maximum A Posteriori*
  - $h_{MAP} = \mathrm{argmax}_h\, P(h \mid D)$
- Especially, if every h equally likely
  - ML: *Maximum Likelihood*
  - $h_{ML} = \mathrm{argmax}_h\, P(D \mid h)$
- Note: applicable to general h & D

Example: Causal Graph



***P*(*R*) = 0.4**

***P*(*W* | *R*) = 0.9**
***P*(*W* | ~*R*) = 0.2**

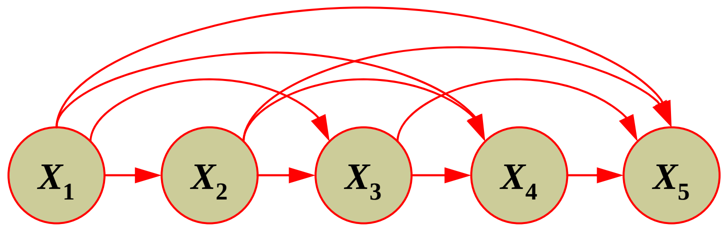
$$P(R|W) = \frac{P(W|R)P(R)}{P(W)}$$

$$= \frac{0.9 \times 0.4}{0.9 \times 0.4 + 0.2 \times 0.6} = 0.75$$

- In general, fewer values need to be specified for the whole distribution than that for the unconstrained case
- $O(2^n) \rightarrow O(n2^k)$, where k is the maximum fan-in of a node

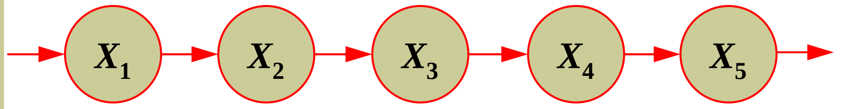
Example: Complete Graph

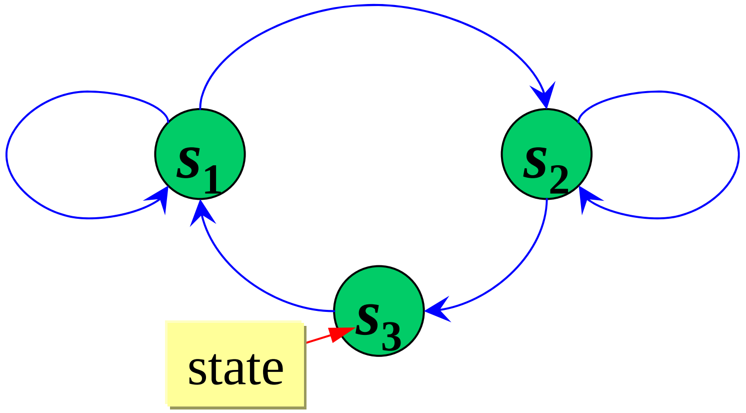
$$\begin{aligned} & P(X_1, X_2, X_3, X_4, X_5) \\ = \; & P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \\ & P(X_4|X_1, X_2, X_3)P(X_5|X_1, X_2, X_3, X_4) \end{aligned}$$



- A network represents a relationship between r.v.'s, however, the representation to give this relationship is not unique; e.g., different orders of X_i can produce different networks

Example: Markov Chain

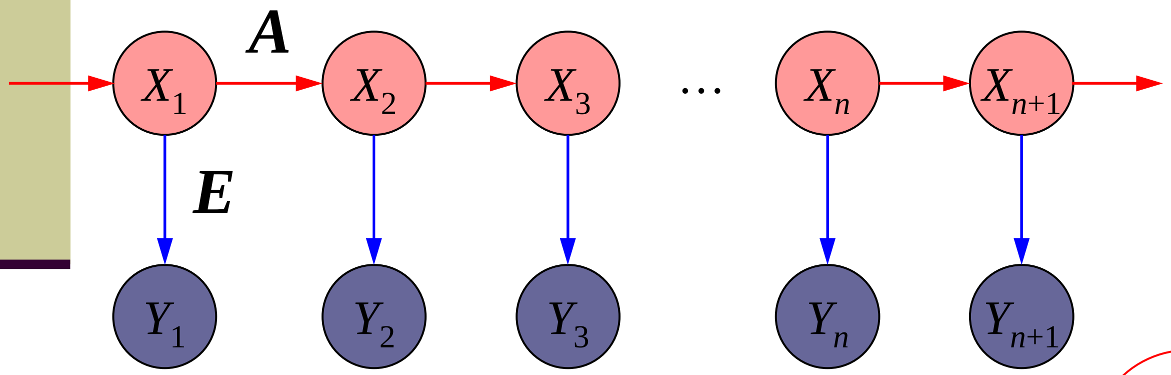


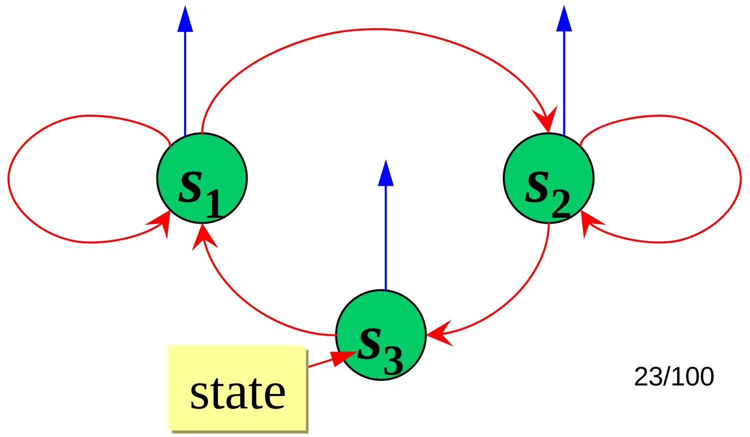


$$
\begin{aligned}
P(X) &= P(X_1, X_2, \ldots, X_n) \\
&= P(X_n|X_{n-1}, \ldots, X_1)P(X_{n-1}|X_{n-2}, \ldots, X_1)\cdots P(X_2|X_1)P(X_1) \\
&= P(X_n|X_{n-1})P(X_{n-1}|X_{n-2})\cdots P(X_2|X_1)P(X_1) \\
&= P(X_1)\prod_{i=2}^{n} P(X_i|X_{i-1})
\end{aligned}
$$

Example: Hidden Markov Model

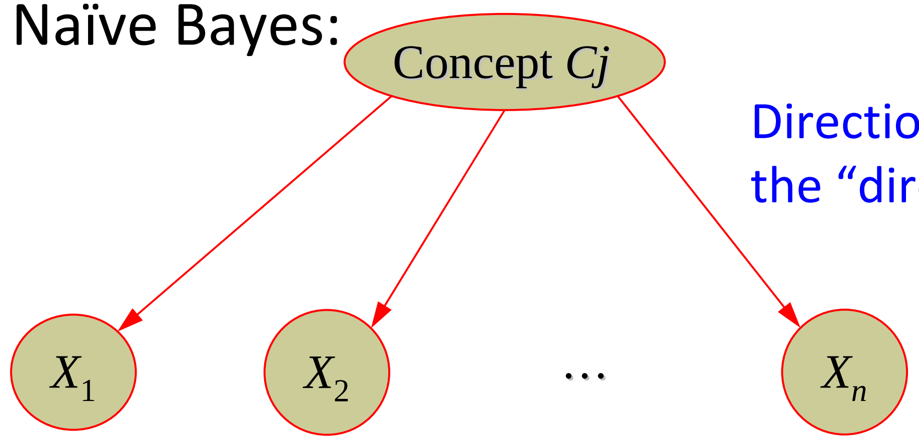
- A hidden Markov model consists a hidden stochastic process that satisfies the property of Markov chain and a series of observable variables that is generated by the process. The graphical representation of the hidden Markov model is as follows





Example: Naïve Bayes

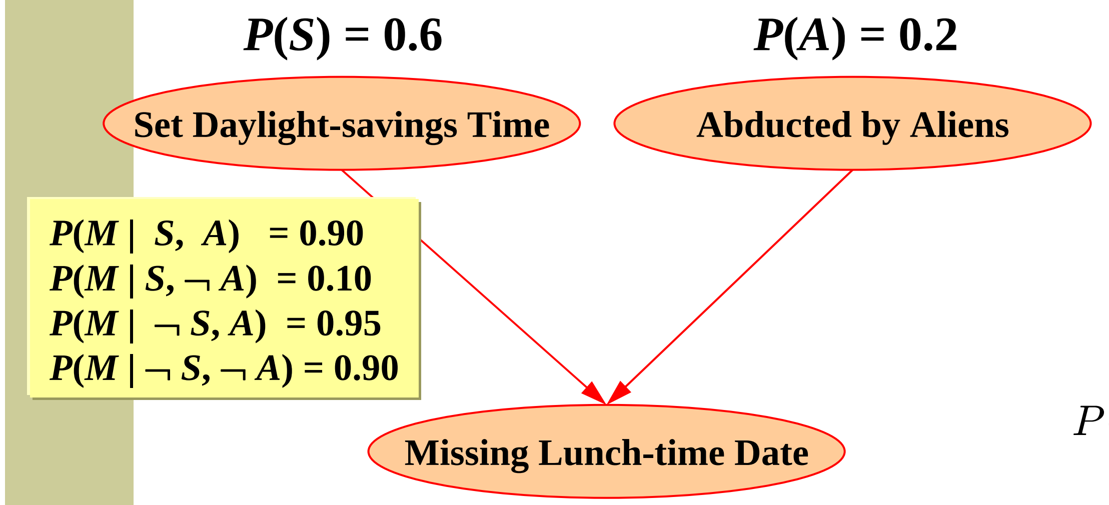
- What is the connection between a BN and classification?
- Suppose one of the variables is the target variable. Can we compute the probability of the target variable given the other variables?
- In Naïve Bayes:



Directions are not the same with the "direction of classification"!

$$P(X_1, X_2, \ldots, X_n, Cj) = P(Cj)\, P(X_1 \mid C_j)\, P(X_2 \mid C_j) \ldots P(X_n \mid C_j)$$

Explaining Away

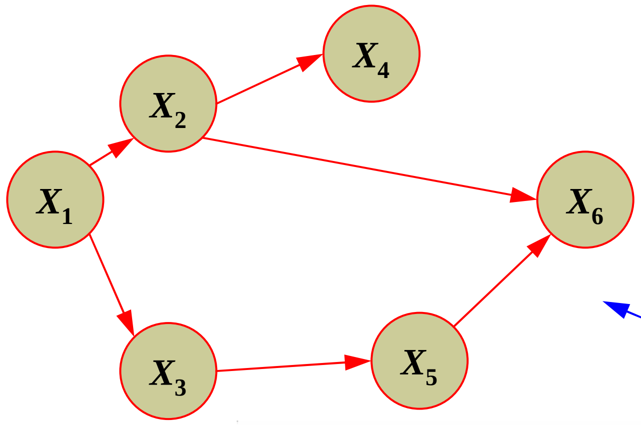


$P(M \mid S, A) = 0.90$
$P(M \mid S, \neg A) = 0.10$
$P(M \mid \neg S, A) = 0.95$
$P(M \mid \neg S, \neg A) = 0.90$

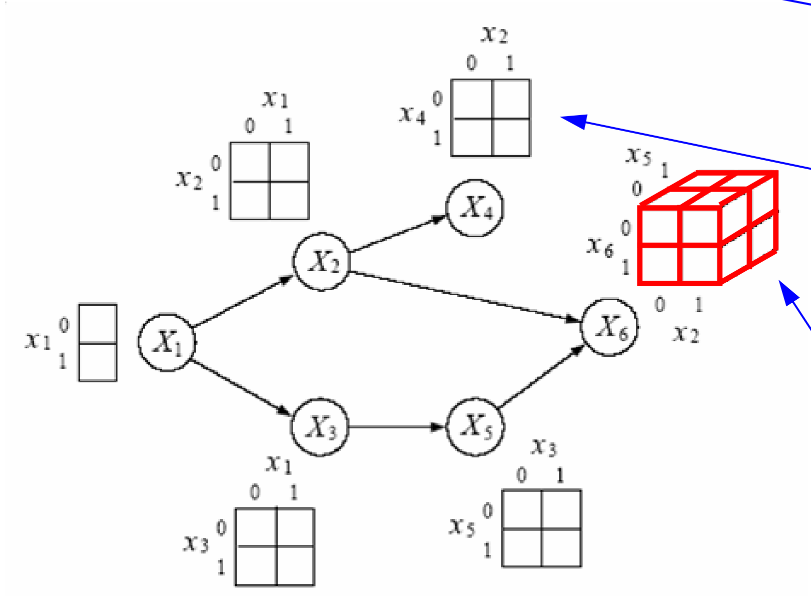
$$\begin{aligned} P(A) &= 0.2 \\ P(A|M) &= \frac{P(M|A)P(A)}{P(M)} \\ &= \frac{0.92 \cdot 0.2}{0.52} = 0.35 > P(A) \\ P(A|\neg S, M) &= \frac{P(M|\neg S, A)P(A|\neg S)}{P(M|\neg S)} \\ &= \frac{P(M|\neg S, A)P(A)}{P(M|\neg S)} \\ &= \frac{0.95 \cdot 0.2}{0.91} \approx 0.21 \\ &< P(A|W) \end{aligned}$$

- *Explaining "Explaining Away"* by M. P. Wellman and M. Henrion, PAMI, 1993.
- Knowing Bob forgets to set the daylight–savings time, the probability that Alice is abducted by Aliens (to cause that Alice missed the date with Bob) decreases, knowing that they missed the date
- Berkson's paradox!

Saving in Time & Space



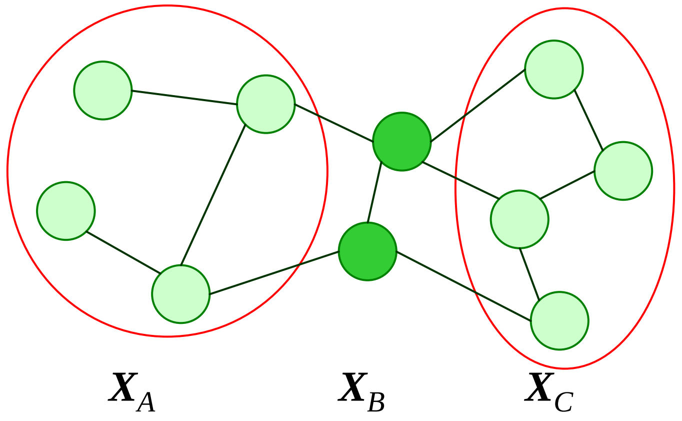
$$
\begin{aligned}
& P(X_1, X_2, X_3, X_4, X_5, X_6) \\
= \; & P(X_1)P(X_2|X_1)P(X_3|X_1) \\
& P(X_4|X_2)P(X_5|X_3)P(X_6|X_2, X_5)
\end{aligned}
$$



- Missing edges imply (cond.) independence
- Each entry/cell represents a cond. prob. combination
- The biggest table, the bottleneck of the computation

Conditional Independence in Undirected Models

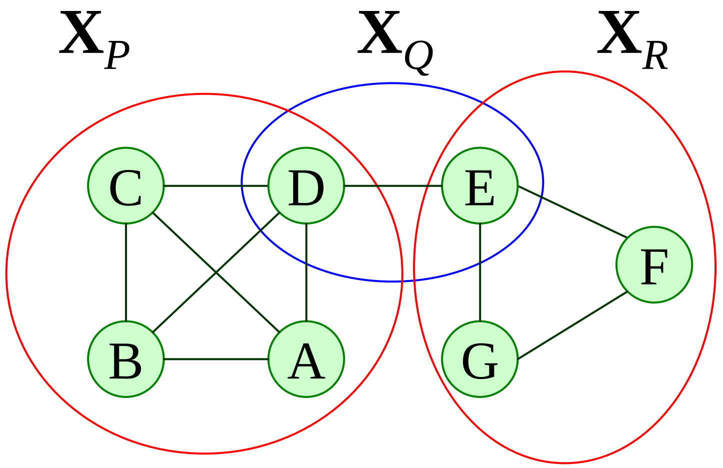
- In undirected models, simple graph separation (as opposed to d-separation) tells us about conditional independencies
- $\boldsymbol{X}_A \perp \boldsymbol{X}_C \mid \boldsymbol{X}_B$ if every path between $\boldsymbol{X}_A$ and $\boldsymbol{X}_C$ is blocked by some node in $\boldsymbol{X}_B$



- "Markov Ball" algorithm:

  remove $\boldsymbol{X}_B$ and see if there is any path from $\boldsymbol{X}_A$ to $\boldsymbol{X}_C$

Conditional Independence and Graph Separation

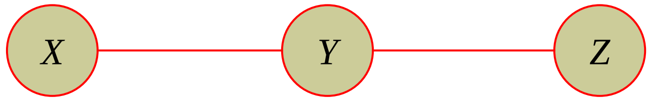


ψ: potential functions

$\mathbf{X}_P, \mathbf{X}_Q, \mathbf{X}_R$: maximal cliques

$$\begin{aligned}
P(A,B,C,D,E,F,G) &= P(A,B,C,D)P(E,F,G|A,B,C,D)\\
&= P(A,B,C,D)P(E,F,G|D)\\
&= P(A,B,C,D)P(E|D)P(F,G|D,E)\\
&= P(A,B,C,D)P(E|D)P(F,G|E)\\
&= \psi(A,B,C,D)\psi(D,E)\psi(E,F,G)\\
&= \psi(\mathbf{X}_P)\psi(\mathbf{X}_Q)\psi(\mathbf{X}_R)
\end{aligned}$$

Interpretation of Clique Potentials



- The model implies $X \perp Z \mid Y$

  $$P(X, Y, Z) = P(Y)\, P(X \mid Y)\, P(Z \mid Y)$$

- We can write this as:

  $$P(X, Y, Z) = P(X, Y)\, P(Z \mid Y) = \psi_{xy}(X, Y)\, \psi_{yz}(Y, Z)$$

  $$P(X, Y, Z) = P(X \mid Y)\, P(Z, Y) = \psi_{xy}(X, Y)\, \psi_{yz}(Y, Z)$$

  cannot have all potentials be marginals

  cannot have all potentials be conditionals

- The positive clique potentials can only be thought of as general “compatibility”, “goodness” or “happiness” functions over their variables, but not as probability distributions.

Undirected Models with Potential Functions

- Whatever factorization we pick, we know that only connected nodes can be arguments of a single local function.
- A *clique* is a fully connected subset of nodes.
- Thus, consider using a product of positive clique potentials:

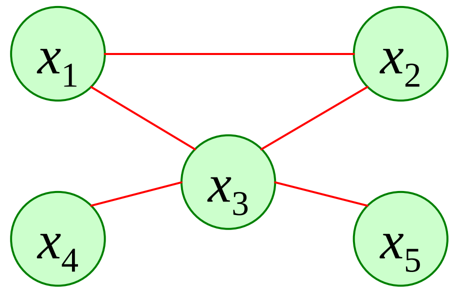
$$P(\mathbf{X}) = \frac{1}{Z} \prod_{cliques\ c} \psi_c(\mathbf{x}_c) \qquad Z = \sum_{\mathbf{X}} \prod_{cliques\ c} \psi_c(\mathbf{x}_c)$$

- The product of functions that don't need to agree with each other.
- Still factors in the way that the graph semantics demand.

Potential functions

- $P(X_{[1:5]}) = \frac{1}{Z}\psi(X_1, X_2, X_3)\psi(X_3, X_4)\psi(X_3, X_5)$

Partition function



Partition Function

- Normalizer $Z(\mathbf{X})$ above is called the “partition function”.
- Computing the normalizer and its derivatives can often be the hardest part of inference and learning in undirected models.
- Often the factored structure of the distribution makes it possible to efficiently do the sums/integrals required to compute Z.
- Don’t always have to compute Z, e.g. for conditional probabilities.

Directed vs. Undirected Models

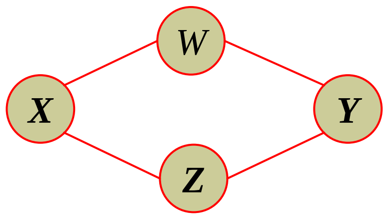
- Directed models
  - using conditional prob. for each local substructure
  - called Bayesian networks
  - may describe some distributions which can not be described by undirected models
  - mainly used in A.I., diagnostic, decision making, etc.
- Undirected models
  - using potential functions in each local substructure
  - called Markov random fields or Markov networks
  - may describe some distributions which can not be described by directed models
  - for problems with little causal structure to guide the graph construction: image restoration, certain optimization problems, models of physical systems

Directed vs. Undirected Models (con’t)

- Directed can’t do it!

  $X \perp Y \mid \{ W, Z \}$

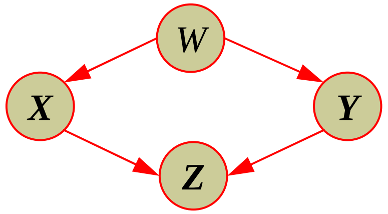
  $W \perp Z \mid \{ X, Y \}$



- Must be acyclic, will have at least one V structure and Bayes ball goes through

  $X \perp Y \mid W$

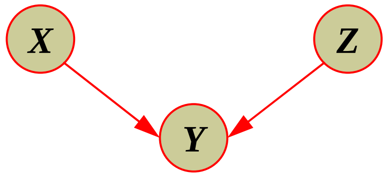
  $X \cancel{\perp} Y \mid \{ W, Z \}$



- Undirected can’t do it!

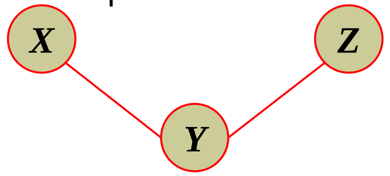
  $X \perp Z$

  $X \cancel{\perp} Z \mid Y$

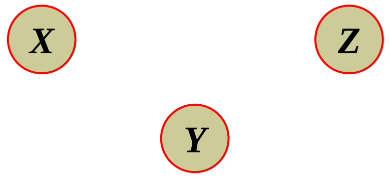


- Undirected can’t do it!

  $X \perp Z \mid Y$



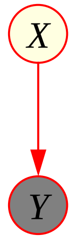
$X \perp Z$



Using Graphical Models: Probabilistic Inference

Probabilistic Inference

- How about statistical inference?
- Task
  - Infer value of some r.v.'s
  - Given values of (some) other r.v.'s in the network
- Facts
  - Some cases are easy, just a series of products for each evaluation of a r.v.
  - General case: NP-hard (Cooper, 1990)
  - Monte-Carlo (Pradham & Dagum, 1996)
- Using Bayes Rule to "reverse" the arrows
- Taking advantage of graph structure (conditional independence)

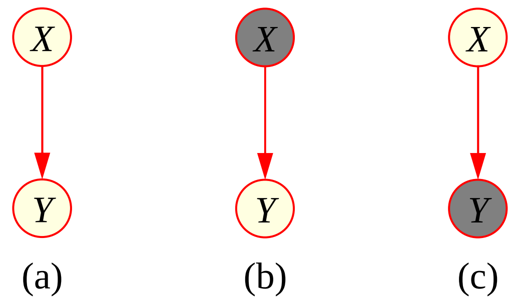


Probabilistic Inference II

- Partition the random variables in a domain $\mathbf{X}$ into three disjoint subsets X_E, X_F, X_R. The general *probabilistic inference* problem is to compute the posterior $P(X_F \mid X_E)$ over the *query nodes* X_F.
- This involves conditioning on evidence nodes X_E and *integrating (summing) out marginal nodes* X_R
- If the joint distribution is represented as a huge table, this is trivial: just select the appropriate indices in the columns corresponding to X_E based on the values, sum over the columns corresponding to X_R , and renormalize the resulting table over X_F

Recall Bayes Rule

- For simple models, we can derive the inference formulae by hand using Bayes rule



(a) $p(x, y) = p(x)p(y \mid x)$

(b) $p(y \mid x)$

(c) $p(x \mid y) = \dfrac{p(x)p(y \mid x)}{\sum_x p(x)p(y \mid x)}$

This is called "reversing the arrow"

- In general, the calculation we want to do is:

$$p(X_F \mid X_E) = \frac{\sum_{x_R} p(X_E, X_F, X_R = x_R)}{\sum_{x_F, x_R} p(X_E, X_F = x_F, X_R = x_R)}$$

- **Q**: Can we do these sums efficiently?

  Q: Can we avoid repeating unnecessary work each time we do inference?

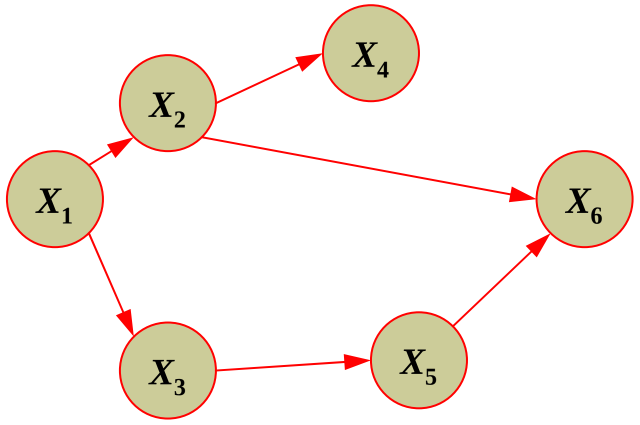
  A: Yes, if we exploit the factorization of the joint distribution

The (Generalized) Distributed Law

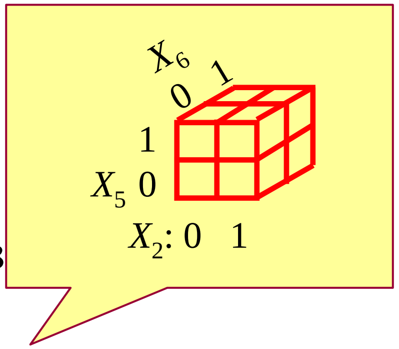
$$a\,b + a\,c = a\,(b + c)$$

left: 2 “$\times$” 1 “$+$”, right: 1 “$\times$”, 1 “$+$”

Take Advantage of Distributed Law



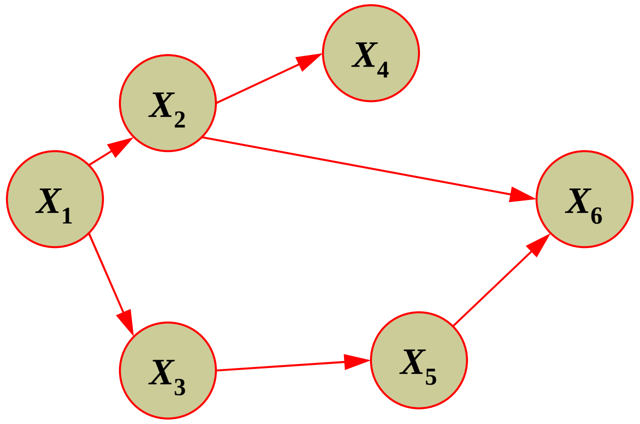
- compute $P(X_{[1..5]})$?



$$
\begin{aligned}
P(X_{[1..5]}) &= \sum_{X_6} P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3)P(X_6|X_2,X_5) \\
&= P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3)\sum_{X_6} P(X_6|X_2,X_5) \\
&= P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3)
\end{aligned}
$$

5×2 multiplications, 1 additions, 5 multiplications, 1 additions!

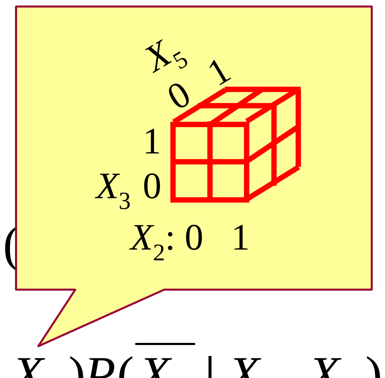
An Example of Inference in Directed Networks



- Compute $P(X_1 \mid \overline{X_6}) = P(X_1, \overline{X_6}) / P(\overline{X_6})$?
- $P(\overline{X_6}) = \sum_{x_1'} P(X_1', \overline{X_6})$

$$P(X_1, \overline{X_6}) = \sum_{X_2,X_3,X_4,X_5} P(X_1,X_2,X_3,X_4,X_5,\overline{X_6})$$

$$= \sum_{X_2,X_3,X_4,X_5} P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)P(X_4 \mid X_2)P(X_5 \mid X_3)P($$

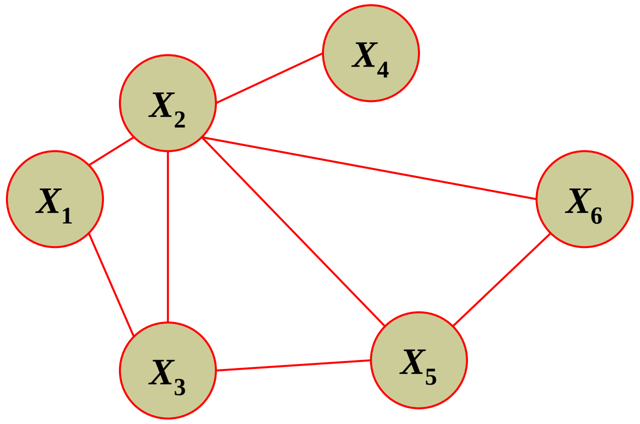


$$= P(X_1)\sum_{X_2} P(X_2 \mid X_1)\sum_{X_4} P(X_4 \mid X_2)\sum_{X_3} P(X_3 \mid X_1)\sum_{X_5} P(X_5 \mid X_3)P(\overline{X_6} \mid X_2, X_5)$$

$$= P(X_1)\sum_{X_2} P(X_2 \mid X_1)\sum_{X_4} P(X_4 \mid X_2)\sum_{X_3} P(X_3 \mid X_1)\Phi_5(X_2, X_3)$$

$$= P(X_1)\sum_{X_2} P(X_2 \mid X_1)\Phi_4(X_2)\Phi_3(X_1, X_2) = P(X_1)\Phi_2(X_1)$$

An Example of Inference in Undirected Networks



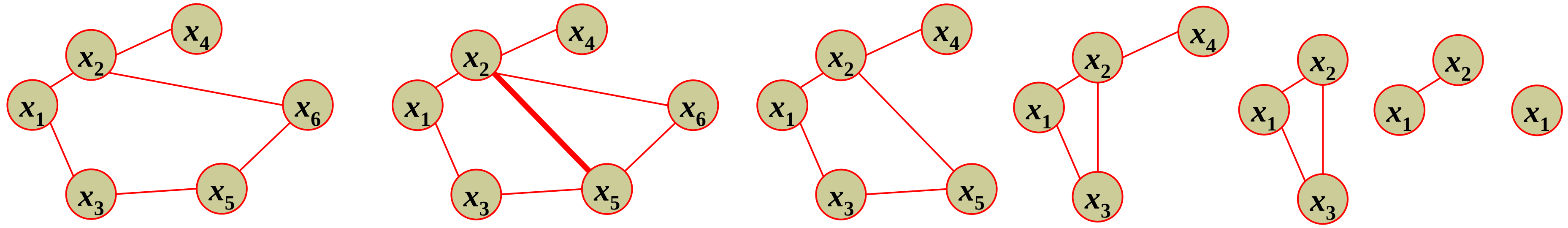
$$
\begin{aligned}
P(X_1,\overline{X_6}) &= \frac{1}{Z}\sum_{X_{[2..5]}} \psi(X_1,X_2)\psi(X_1,X_3)\psi(X_2,X_4)\psi(X_3,X_5)\psi(X_2,X_6)\psi(X_5,X_6) \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\sum_{X_4}\psi(X_2,X_4)\sum_{X_3}\psi(X_1,X_3)\sum_{X_5}\psi(X_3,X_5)\boxed{\psi(X_2,\overline{X_6})\psi(X_5,\overline{X_6})} \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\sum_{X_4}\psi(X_2,X_4)\sum_{X_3}\psi(X_1,X_3)\boxed{\sum_{X_5}\psi(X_3,X_5)\boxed{\Phi(X_2,X_5)}} \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\boxed{\sum_{X_4}\psi(X_2,X_4)}\boxed{\sum_{X_3}\psi(X_1,X_3)\boxed{\Phi(X_2,X_3)}} \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\boxed{\Phi(X_2)}\boxed{\Phi(X_1,X_2)} = \frac{1}{Z}\boxed{\Phi(X_1)}
\end{aligned}
$$

Some Hints…

- During the computation, given the order (which we do not know how to choose it yet), we do the following:
  - Push the sums in as far as possible
  - After performing the innermost sum, we create a new term which does not depend on the term summed out
  - The process is like a node elimination process in the graph
  - Each time a node X_i is eliminated, all the neighbors of X_i are connected
  - Continue to do summations until only the query node left
  - After the elimination process, the graph becomes triangulated
  - There are some other details, …
- Not much difference between directed and undirected models; actually, undirected model is easier than the other because we got symmetry!

Single Node Posteriors

- For a single node posterior (i.e. X_F is a single node), there is a simple, efficient algorithm based on eliminating nodes.
- Notation: $\overline{X_i}$ is the value of evidence node X_i .
- The algorithm, called ELIMINATION, requires a *node ordering* to be given, which tells it which order to do the summations in.
- In this ordering, the query node must appear last. Graphically, we'll remove a node from the graph once we sum it out.



Evidence Potentials

- Elimination also uses a bookkeeping trick, called evidential functions:

$$g(\overline{X_i}) = \sum_{X_i} g(X_i)\delta(X_i, \overline{X_i})$$

where $\delta(X_i, \overline{X_i})$ is 1 if $X_i = \overline{X_i}$ and 0 otherwise.

- This trick allows us to treat conditioning in the same way as we treat marginalization. So everything boils down to doing sums:

$$P(X_F|\overline{X_E}) = P(X_F, \overline{X_E}) / P(\overline{X_E})$$

$$P(X_F, \overline{X_E}) = \sum_{X_R}\sum_{X_E} P(X_F, X_E, X_R)\delta(X_E, \overline{X_E})$$

$$P(\overline{X_E}) = \sum_{X_R}\sum_{X_E}\sum_{X_F} P(X_F, X_E, X_R)\delta(X_E, \overline{X_E})$$

- We just pick an ordering and go for it...

Elimination Algorithm I

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$: graph, E : index set for evidences, F : index for the query node

```
ELIMINATE($\mathcal{G}$, $E$, $F$)
   INITIALIZE($\mathcal{G}$, $F$)
   EVIDENCE($E$)
   UPDATE($\mathcal{G}$)
   NORMALIZE($F$)

INITIALIZE($\mathcal{G}$, $F$)
   choose an ordering $I$ such that $F$ appears last
   for each node $X_i$ in $\mathcal{V}$
         place $P(X_i|X_{\pi_i})$ on the active list
   end

EVIDENCE(E)
   for each $i$ in $E$
         place $\delta(X_i, \overline{X_i})$ on the active list
   end
```

Elimination Algorithm II

```
UPDATE($\mathcal{G}$)
   for each $i$ in $I$
          find all potentials from the active list
               that reference $X_i$ and remove them
               from the active list
          let $\phi_i(X_{T_i})$ denote the product of these potentials
          let $m_i(X_{S_i}) = \sum_{x_i} \phi_i(X_{T_i})$
          place $m_i(X_{S_i})$ on the active list
   end

NORMALIZE($F$)
   $P(X_F|\overline{X_E}) \leftarrow \phi_F(X_F) / \sum_{x_F} \phi_F(X_F)$
```

1. T_i denotes i plus all other nodes referenced by potentials on i
2. S_i denotes all T_i except i

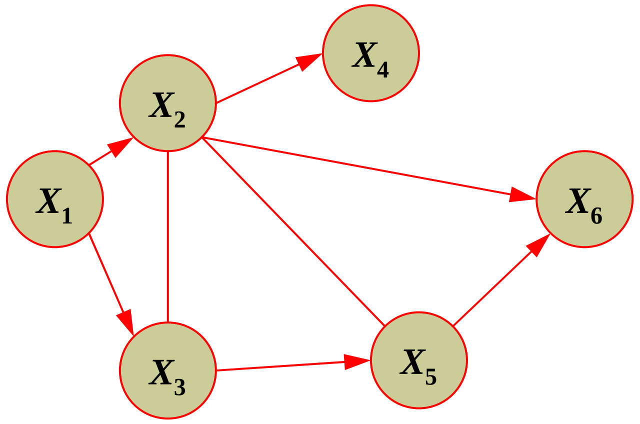
Node Elimination

- The algorithm we presented is really a way of eliminating nodes from a graph one by one. For undirected graphs:

```
foreach node $X_i$ in ordering $I$:
connect all the neighbors of $X_i$
remove $X_i$ from the graph
end
```

- The removal operation requires summing out X_i (or conditioning on observed evidence for X_i).
- Summing out X_i leaves a function involving all its previous neighbors and thus they become connected by this step.
- The original graph, augmented by all the added edges is now a triangulated graph. (Reminder: triangulated means that every cycle of length >3 contains a chord, i.e. an edge not on the cycle but between two nodes in the cycle.)

Added Edges = Triangulation



- It is easy to check if a graph is triangulated in linear time
- It is easy to triangulate a non-triangulated graph
- But it is very hard to do so in a way that induces small clique sizes

$$P(X_1,\overline{X_6}) = \sum_{X_2,X_3,X_4,X_5} P(X_1,X_2,X_3,X_4,X_5,\overline{X_6})$$

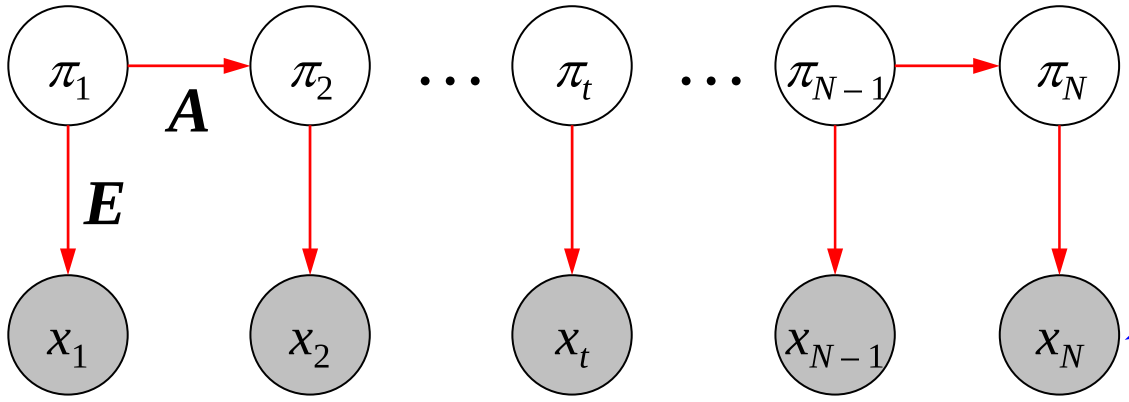
$$= \sum_{X_2,X_3,X_4,X_5} P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1)P(X_4 \mid X_2)P(X_5 \mid X_3)P(\overline{X_6} \mid X_2,X_5)$$

$$= P(X_1)\sum_{X_2} P(X_2 \mid X_1)\sum_{X_4} P(X_4 \mid X_2)\sum_{X_3} P(X_3 \mid X_1)\sum_{X_5} P(X_5 \mid X_3)\boxed{P(\overline{X_6} \mid X_2,X_5)}$$

$$= P(X_1)\sum_{x_2} P(X_2 \mid X_1)\sum_{x_4} P(X_4 \mid X_2)\sum_{x_3} P(X_3 \mid X_1)\boxed{\Phi_5(X_2,X_3)}$$

$$= P(X_1)\sum_{x_2} P(X_2 \mid X_1)\boxed{\Phi_4(X_2)}\boxed{\Phi_3(X_1,X_2)} = P(X_1)\boxed{\Phi_2(X_1)}$$

Node Elimination as Probabilistic Inference



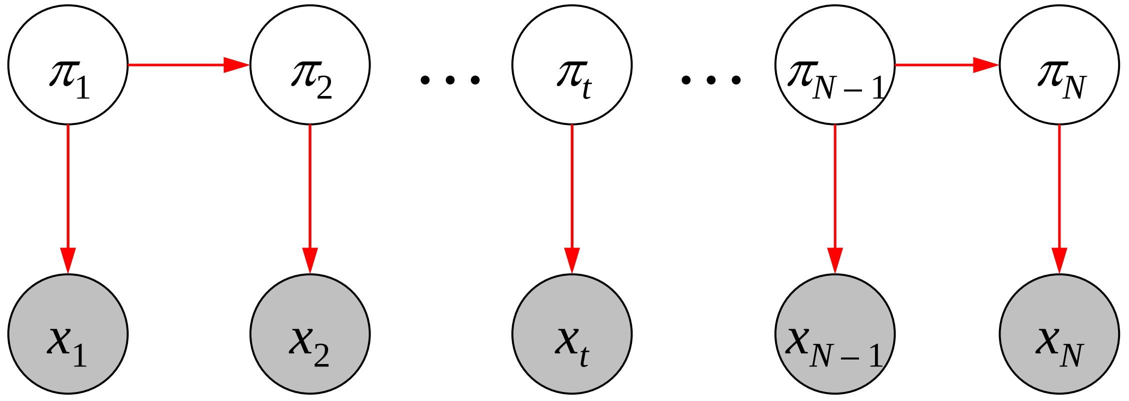
Shaded nodes: with evidence

$$
\begin{aligned}
P(\mathbf{x}, \pi_t) &= \sum_{\pi_1,\ldots,\pi_{t-1},\pi_{t+1},\ldots,\pi_N} P(\mathbf{x}, \pi_1, \ldots, \pi_N) \\
&= \sum_{\pi_1,\ldots,\pi_{t-1},\pi_{t+1},\ldots,\pi_N} P(\pi_1) \prod_{s=1}^{N} P(x_s|\pi_s) \prod_{s=1}^{N-1} P(\pi_{s+1}|\pi_s)
\end{aligned}
$$

$$\boxed{P(\mathbf{x}, \pi_t) = f_i(t) b_i(t)}$$

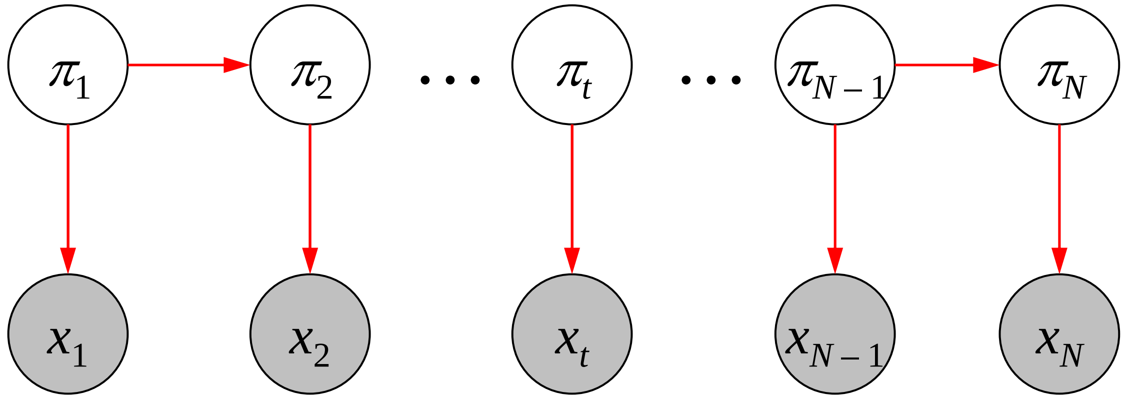
**assuming no begin & end!*

Forward Algorithm



$$\begin{aligned}
f(t) &= P(x_1, \ldots, x_t, \pi_t) \\
&= \sum_{\pi_1,\ldots,\pi_{t-1}} P(x_1, \ldots, x_t, \pi_1, \ldots, \pi_t) \\
&= \sum_{\pi_1,\ldots,\pi_{t-1}} P(\pi_1) \prod_{s=1}^{t} P(x_s|\pi_s) \prod_{s=1}^{t-1} P(\pi_{s+1}|\pi_s) \\
&= P(x_t|\pi_t) \sum_{\pi_{t-1}} P(\pi_t|\pi_{t-1}) P(x_{t-1}|\pi_{t-1}) \cdots \sum_{\pi_1} P(\pi_1) P(\pi_2|\pi_1) P(x_1|\pi_1)
\end{aligned}$$

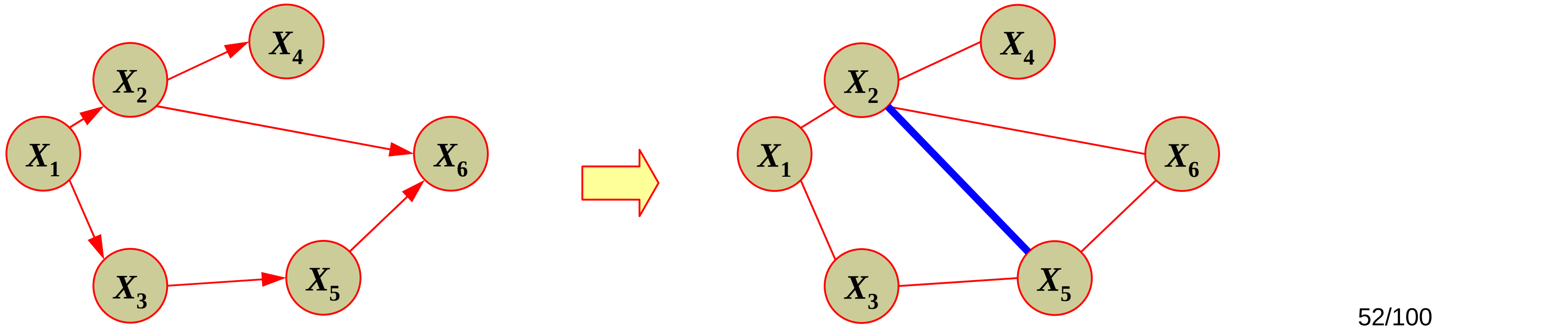
Backward Algorithm



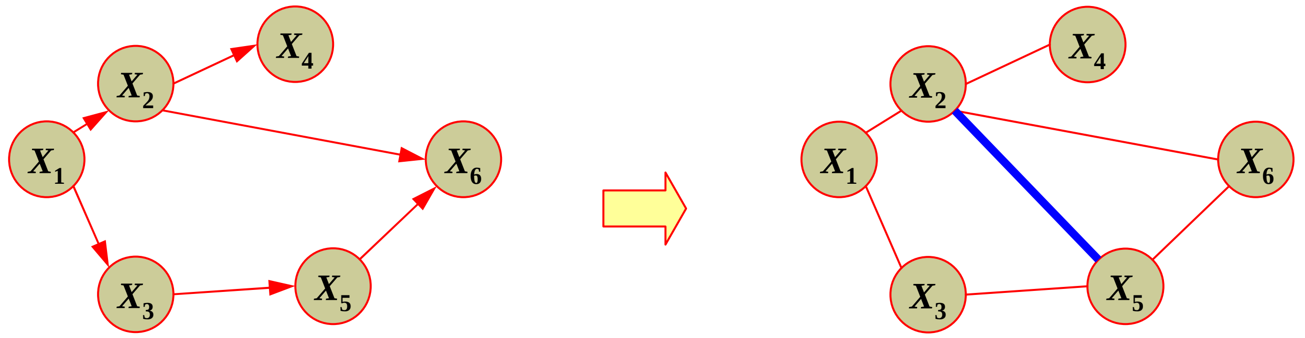
$$
\begin{aligned}
b(t) &= P(x_{t+1},\ldots,x_N|\pi_t) \\
&= \sum_{\pi_{t+1},\ldots,\pi_N} P(x_{t+1},\ldots,x_N,\pi_{t+1},\ldots,\pi_N|\pi_t) \\
&= \sum_{\pi_{t+1},\ldots,\pi_N} \prod_{s=t+1}^{N} P(x_s|\pi_s) \prod_{s=t}^{N-1} P(\pi_{s+1}|\pi_s) \\
&= \sum_{\pi_{t+1}} P(\pi_{t+1}|\pi_t)P(x_{t+1}|\pi_{t+1}) \cdots \sum_{\pi_N} P(\pi_N|\pi_{N-1})P(x_N|\pi_N)
\end{aligned}
$$

Moralization

- For directed graphs, the parents may not be explicitly connected, but they are involved in the same potential function $P(X_i \mid X_{\pi_i})$
- Thus to think of ELIMINATION as a node removal algorithm, we first must connect all the parents of every node and drop the directions on the links
- This step is known as "Moralization" and it is essential: since conditioning couples parents in directed models ("explaining away") we need a mechanism for respecting this when we do inference.



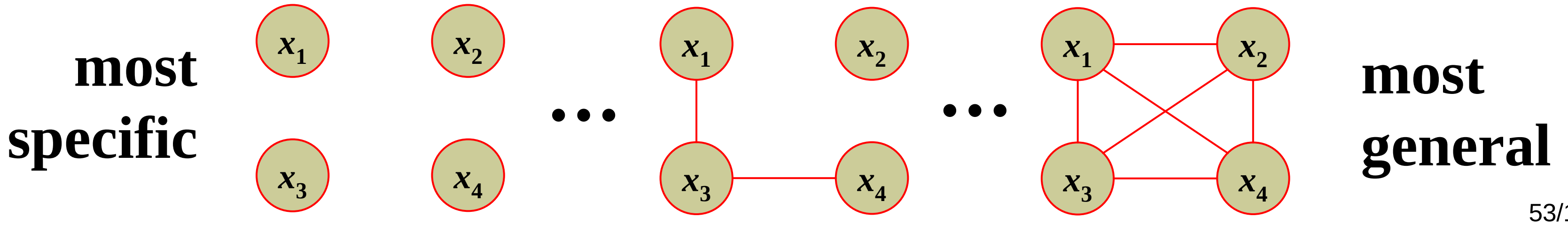
Moralization (cont.)



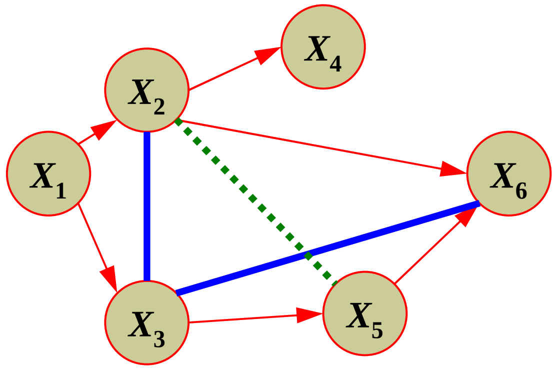
$$P(X_1)(X_2 \mid X_1)P(X_3 \mid X_1)P(X_4 \mid X_2)P(X_5 \mid X_3)P(\overline{X_6} \mid X_2, X_5)$$

$$\frac{1}{Z}\psi(X_1, X_2)\psi(X_1, X_3)\psi(X_2, X_4)\psi(X_3, X_5)\psi(X_2, X_5, \overline{X_6})$$

- The graph after moralization looks more general!
- What do we lose? e.g., $X_2 \perp X_5 \mid X_1 , X_3$
- Moral graph is more general (loses some independencies)



Markov Blanket



- Elimination order 1:
  $X_2 \leftarrow X_4 \leftarrow X_3 \leftarrow X_5 \leftarrow X_6$
- Elimination order 2:
  $X_2 \leftarrow X_4 \leftarrow X_3 \leftarrow X_6 \leftarrow X_5$
  For each eliminated node X_i $(= X_5)$, we need to take care (1) the child $(= X_6)$, (2) the parent(s) $(= X_3)$, and (3) the parent(s) of the child $(= X_2)$.

Order 1:

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

$$= P(X_1)\sum_{X_2} P(X_2 | X_1)\sum_{X_4} P(X_4 | X_2)\sum_{X_3} P(X_3 | X_1)\sum_{X_5} P(X_5 | X_3)\Phi(X_2, X_5)$$

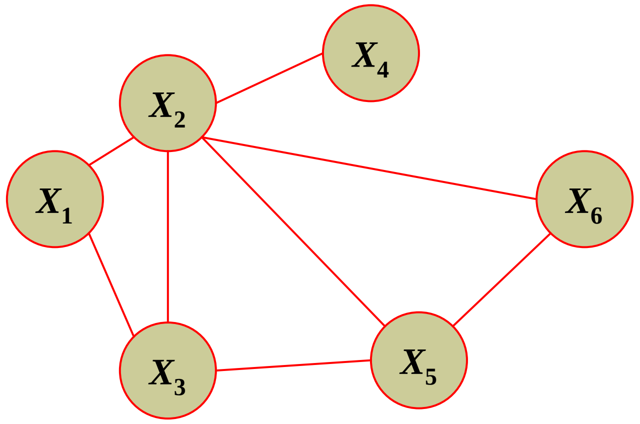
Order 2:

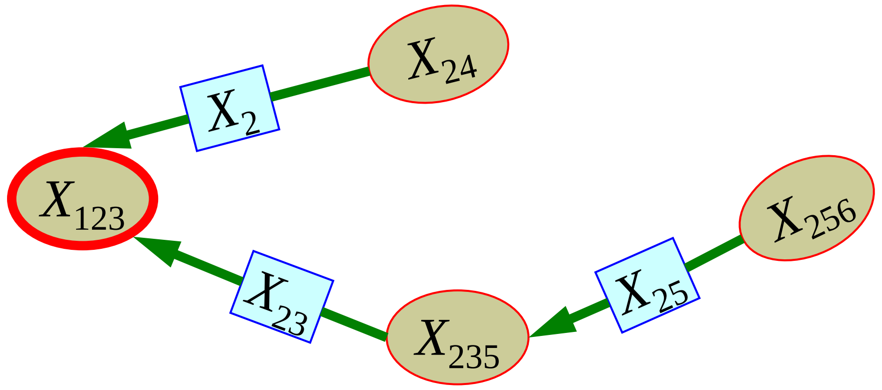
$$P(X_1) = P(X_1)\sum_{X_2} P(X_2 | X_1)\sum_{X_4} P(X_4 | X_2)\sum_{X_3} P(X_3 | X_1)\sum_{X_6}\sum_{X_5} P(X_5 | X_3)P(X_6 | X_2, X_5)$$

$$= P(X_1)\sum_{X_2} P(X_2 | X_1)\sum_{X_4} P(X_4 | X_2)\sum_{X_3} P(X_3 | X_1)\sum_{X_6} \Phi(X_2, X_3, X_6)$$

*Here we use conditional probabilities. It may not sum to 1 if some evidences are involved!

Viewed by Junction Tree Process





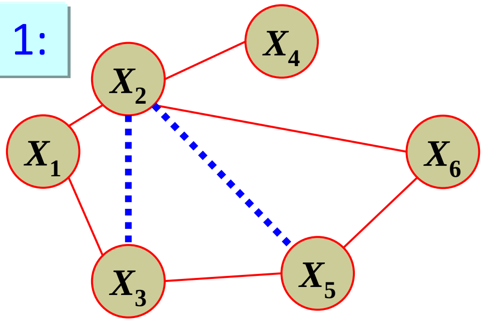
$$
\begin{aligned}
P(X_1,\overline{X_6}) &= \frac{1}{Z}\sum_{X_{[2..5]}} \psi(X_1,X_2)\psi(X_1,X_3)\psi(X_2,X_4)\psi(X_3,X_5)\psi(X_2,X_6)\psi(X_5,X_6) \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\sum_{X_4}\psi(X_2,X_4)\sum_{X_3}\psi(X_1,X_3)\sum_{X_5}\psi(X_3,X_5)\psi(X_2,\overline{X_6})\psi(X_5,\overline{X_6}) \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\sum_{X_4}\psi(X_2,X_4)\sum_{X_3}\psi(X_1,X_3)\sum_{X_5}\psi(X_3,X_5)\boxed{\Phi(X_2,X_5)} \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\sum_{X_4}\psi(X_2,X_4)\sum_{X_3}\psi(X_1,X_3)\boxed{\Phi(X_2,X_3)} \\
&= \frac{1}{Z}\sum_{X_2}\psi(X_1,X_2)\boxed{\Phi(X_2)}\Phi(X_1,X_2) = \frac{1}{Z}\Phi(X_1)
\end{aligned}
$$

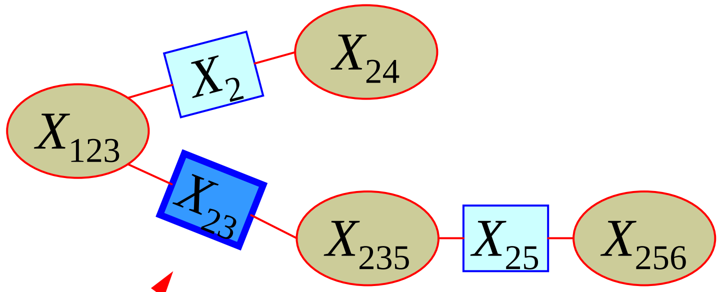
Node Elimination: A Special Case of Junction Tree Algorithm

- The ELIMINATION algorithm we described was query based: given the single node marginal to compute (the last item in the ordering), it efficiently summed out or conditioned on all other variables?
- But what if we want to do
  1. pairwise queries?
  2. multiple inferences?

Motivation 1: Pairwise Queries

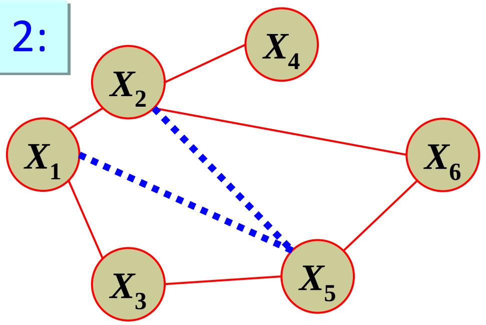
Query 1:



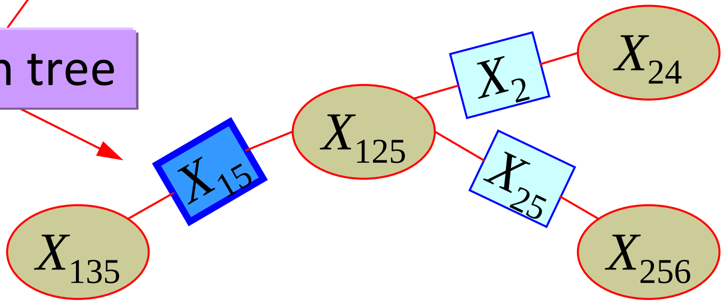


$$P(X_2, X_3) = \frac{1}{Z} \sum_{X_1} \psi(X_2, X_1)\psi(X_3, X_1) \sum_{X_4} \psi(X_4, X_2) \sum_{X_5} \psi(X_5, X_3) \sum_{X_6} \psi(X_2, X_6)\psi(X_5, X_6)$$

Query 2:



junction tree



$$P(X_1, X_5) = \frac{1}{Z} \sum_{X_3} \psi(X_3, X_1)\psi(X_5, X_3) \sum_{X_2} \psi(X_2, X_1) \sum_{X_6} \psi(X_6, X_2)\psi(X_6, X_5) \sum_{X_4} \psi(X_4, X_2)$$

How to compute those pairwise (or even more) posterior probability systematically?

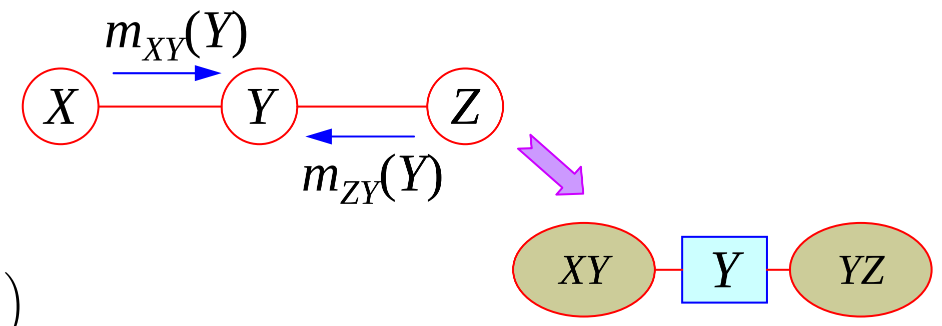
Motivation 2: Multiple Queries - Efficient Multi-Elimination

- If we want to do multiple inferences? For example, during learning, constraint satisfaction, planning
- We could run `ELIMINATION` once for each marginal, but this would be *extremely* inefficient since most of the calculations would be duplicated
- We want an algorithm that reuses work efficiently to compute all marginals given evidence
- We do not want to duplicate work unnecessarily
- We want to reuse calculations as best as possible
- This needs:
  1) A plan for which intermediate factors to compute in what order
  2) Some storage for these intermediate factors.

Motivation 2: Multiple Queries - Message Passing

Case 1:

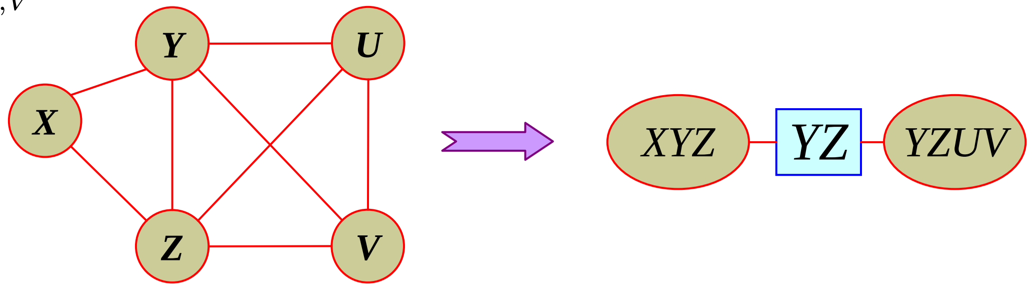
$$\begin{aligned} P(Y) &= \textstyle\sum_{X,Z} P(X,Y,Z) \\ &= \frac{1}{Z}\textstyle\sum_{X,Z}\psi(X,Y)\psi(Y,Z) \\ &= \frac{1}{Z}\textstyle\sum_{X}\psi(X,Y)\sum_{Z}\psi(Z,Y) \\ &= \frac{1}{Z}\left(\textstyle\sum_{X}\psi(X,Y)\right)\left(\textstyle\sum_{Z}\psi(Z,Y)\right) \\ &= \frac{1}{Z} m_{XY}(Y)\, m_{ZY}(Y) \end{aligned}$$



this two parts can be computed independently!

Case 2:

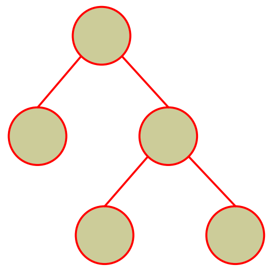
$$\begin{aligned} P(Y,Z) &= \frac{1}{Z}\textstyle\sum_{X,U,V}\psi(X,Y,Z)\psi(V,Y,Z,U) \\ &= \frac{1}{Z}\textstyle\sum_{X}\psi(X,Y,Z)\sum_{U,V}\psi(V,Y,Z,U) \\ &= \frac{1}{Z} m_1(Y,Z)\, m_2(Y,Z) \end{aligned}$$

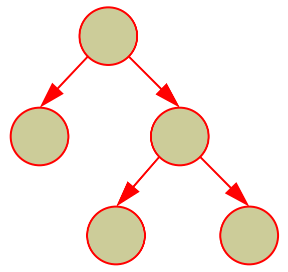


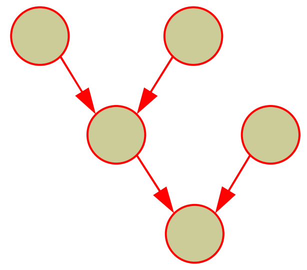
- The final joint probability is given by collecting “message” coming from different directions!

Tree-Structured Graphical Models

- For now, we will focus on tree-structured graphical models
- Trees are an important class; they incorporate all chains (e.g. HMMs) as well
- Exact inference on trees is the basis for the junction tree algorithm which solves the general exact inference problem for directed acyclic graphs and for many approximate algorithms which can work on intractable or cyclic graphs
- Directed and undirected trees make exactly same conditional independence assumptions, so we cover them together
  - (a) & (b) are trees, (c) is not a tree!


(a)


(b)

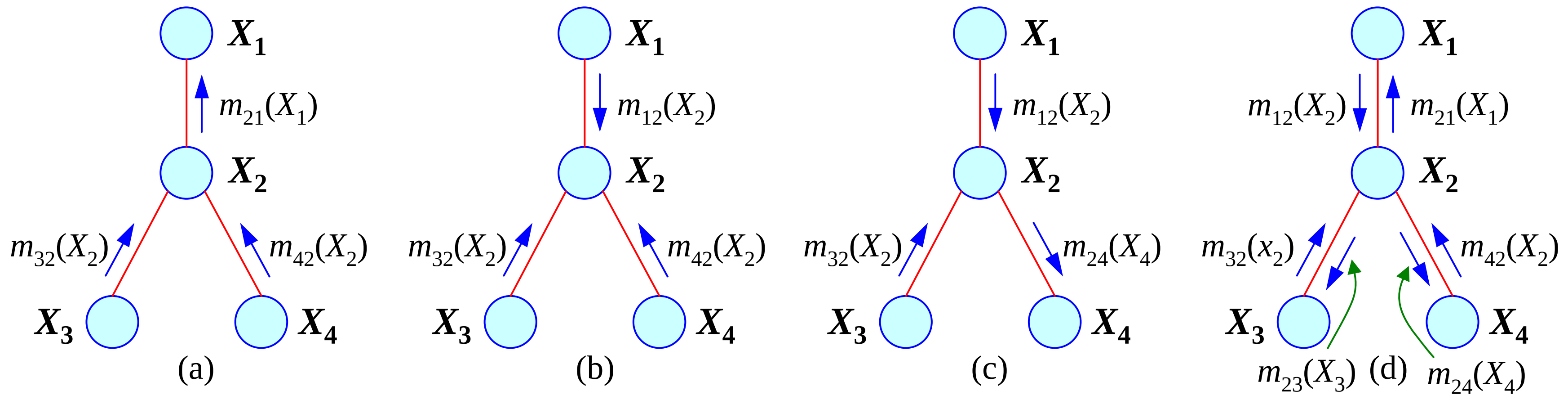

(c)

Elimination on Trees

- Recall basic structure of `ELIMINATION`:
  1. Convert directed graph to undirected by moralization
  2. Chose elimination ordering with query node last
  3. Place all potentials on active list
  4. Eliminate nodes by removing all relevant potentials, taking product, summing out node and placing resulting factor back onto potential list
- What happens when the original graph is a tree?
  1. No moralization is necessary (because only one parent for each node)
  2. There is a natural elimination ordering with query node as root (any *depth first search* order)
  3. For directed models, all subtrees with no evidence nodes can be ignored (since they will leave a potential of unity once they are eliminated)

Message are Reused in MultiElimination

- Consider querying X_1, X_2, X_3 and X_4 in the graph below.
- The messages needed for X_1, X_2, X_4 individually are shown (a-c).
- Also shown in (d) is the set of messages needed to compute all possible marginals over single query nodes.



- Key insight: even though the naive approach (rerun `ELIMINATION`) needs to compute N^2 messages to find marginals for all N query nodes, there are only $2N$ possible messages.
- We can compute all possible messages in only double the amount of work it takes to do one query.
- Then we take the product of relevant messages to get marginals.

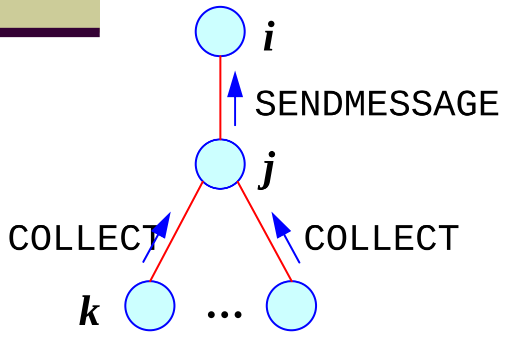
Belief Propagation (Sum-Product Algorithm)

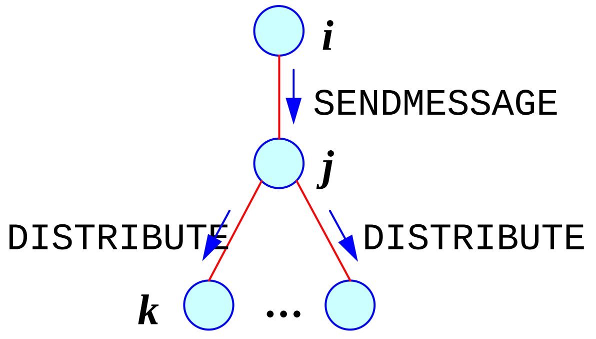
- Choose a root node (arbitrarily or as first query node)
- If *j* is an evidence node, $\psi^E(X_j) = \delta(X_j, \overline{X}_j)$, else $\psi^E(X_j) = 1$
- Pass messages from leaves up to root and then back down using:

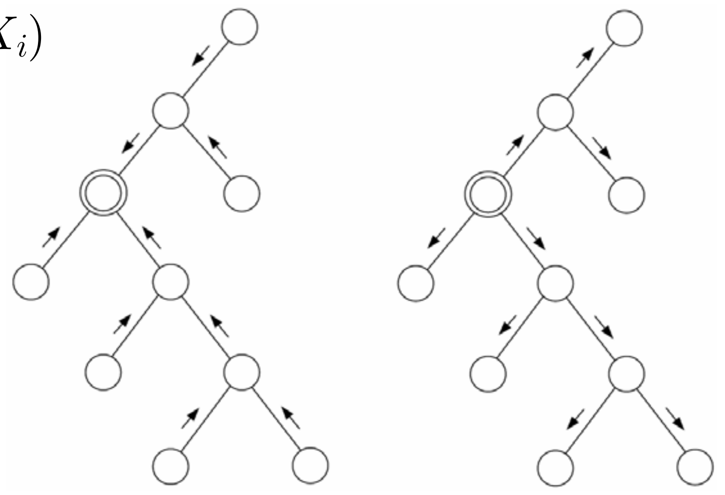
$$m_{ji}(X_i) = \sum_{x_j} \left(\psi^E(X_j)\psi(X_i, X_j) \prod_{k \in c(j)} m_{kj}(X_j) \right)$$

- Given messages, compute marginals using:

$$P(X_i | \overline{\mathbf{X}_E}) \propto \psi^E(X_i) \prod_{k \in c(i)} m_{ki}(X_i)$$







Sum-product Algorithm II

$\mathcal{T}$: tree, $\mathcal{V}$: nodes, E : index set for evidences

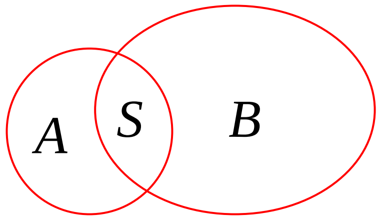
```
SUM-PRODUCT($\mathcal{T}$, $E$)
    EVIDENCE($E$)
    $f$ = CHOOSEROOT($\mathcal{V}$)
    for $c \in \mathcal{N}(f)$
        COLLECT($f$, $e$)
    for $e \in \mathcal{N}(f)$
        DISTRIBUTE($f$, $e$)
    for $i \in \mathcal{V}$
        COMPUTEMARGINAL($i$)

EVIDENCE($E$)
    for $i \in E$
        $\psi^E(X_i) = \psi(X_i)\delta(X_i, \overline{X_i})$
    for $i \notin E$
        $\psi^E(X_i) = \psi(X_i)$

COLLECT($i$, $j$)
    for $k \in \mathcal{N}(j)\backslash i$
        COLLECT($j$, $k$)
    SENDMESSAGE($j$, $i$)

DISTRIBUTE($i$, $j$)
    SENDMESSAGE($i$, $j$)
    for $k \in \mathcal{N}(j)\backslash i$
        DISTRIBUTE($j$, $k$)

SENDMESSAGE($j$, $i$)
    $m_{ji}(X_i) = \sum_{x_j} (\psi^E(X_j)\psi(X_i, X_j) \prod_{k \in \mathcal{N}(j)\backslash i} m_{kj}(X_j))$

COMPUTEMARGINAL($i$)
    $P(X_i) \propto \psi^E(X_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(X_i)$
```

Decomposition & Decomposability

- Def: A triple (A, B, S) of disjoint subsets of the vertex set V of an undirected graph $G \equiv (V, E)$ is said to form a decomposition of G if $V = A \cup B \cup S$ and

  (1) S separates A from B;

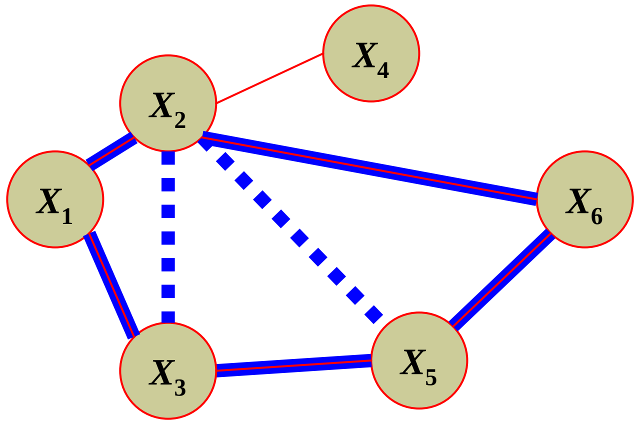
  (2) S is a complete subset of V



- Def (a recursive definition): An undirected graph G is said to be decomposable if it is complete, or if there exists a proper decomposition (A, B, S) into decomposable subgraphs $G_{A\cup S}$ and $G_{B\cup S}$

Triangulated Graph

- A triangulated graph is an undirected graph with the property that every cycle of length $n \geq 4$ possesses a chord; i.e., two non-consecutive vertices are neighbors. A definition such as this is a so-called “forbidden path” definition
- **Prop**: If $G \equiv (V, E)$ is triangulated and $A \subseteq V$, then G_A is triangulated



Junction Tree Property

- A tree of cliques possesses the junction tree property if for every pair of clique V and W, all cliques on the (unique) path between V and W contain $V \cap W$. A tree of cliques that possesses the junction tree property is referred to as a junction tree
- If a node A appears in two cliques in a junction tree, then A is contained in every clique along the path between these two cliques

Decomposable ≡ Triangulated ≡ Having Junction Tree

- The following five characterizations of graph G are equivalent

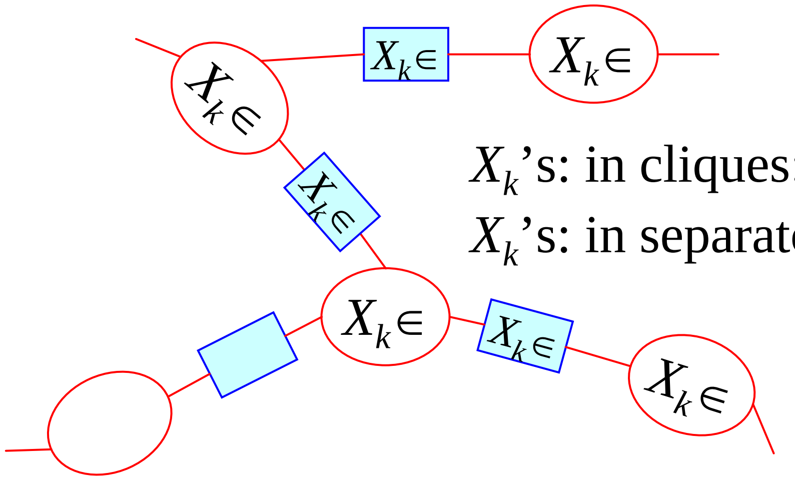
1. G is decomposable
2. G is triangulated
3. Every minimal (α, β)-separator in G is complete
4. G has a junction tree
5. In G, there exists an ordering such that elimination using that ordering introduces no new edges

Maximal Spanning Tree

- **Def**: $w(T)$: sum of cardinalities of the separator sets in the tree T
- **Theorem**: A clique tree T is a junction tree if and only if it is a maximal spanning tree

Proof:

$$\begin{aligned} w(T) &= \sum_{j=1}^{M-1} |S_j| \\ &= \sum_{j=1}^{M-1} \sum_{k=1}^{N} 1(X_k \in S_j) \\ &= \sum_{k=1}^{N} \sum_{j=1}^{M-1} 1(X_k \in S_j) \\ &\le \sum_{k=1}^{N} \left[\sum_{i=1}^{M} 1(X_k \in C_i) - 1 \right] \\ &= \sum_{i=1}^{M} \sum_{k=1}^{N} 1(X_k \in C_i) - N = \sum_{i=1}^{M} |C_i| - N \end{aligned}$$



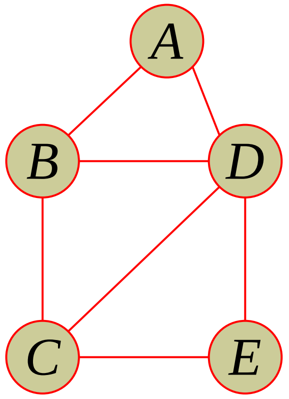
X_k's: in cliques: 4

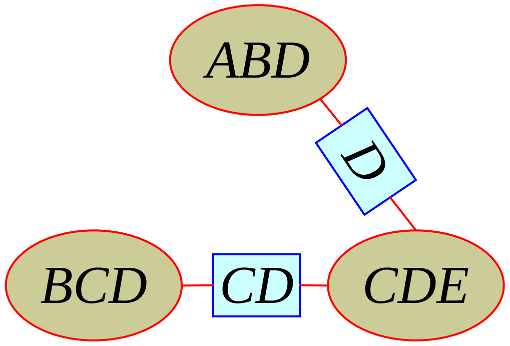
X_k's: in separators: 3

$\Rightarrow$ The equality holds iff T is a junction tree!

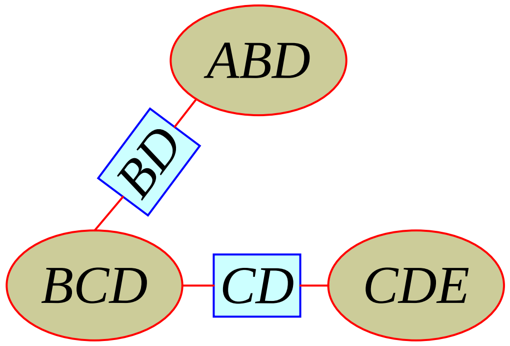
Construct Maximal Spanning Tree

- **Step 1**: Build (max-)cliquese
- **Step 2**: Connect cliques by edges with the weight equal to the cardinalities of their separators
- **Step 3**: Construct the maximum spanning tree by either Kruskal or Prim's algorithms
  - Kruskal: choose edges from the maximal weighted edge without forming any cycles
  - Prim's: start from a maximal spanning subtree and add the maximal nearby edges without forming any cycles





$w(T) = 3$



$w(T) = 4$

The Whole Junction Tree Process

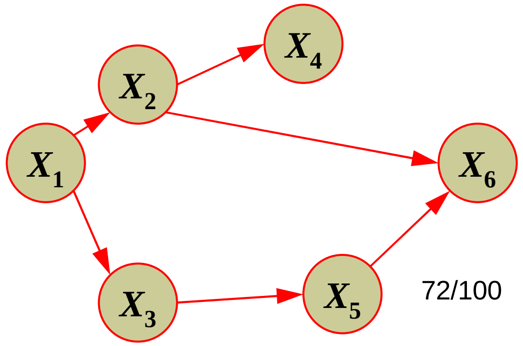
- Moralization (if for directed models)
- Introduction of evidence
- Triangulation
  - if the graph is not yet triangulated
  - no guarantee to find the order minimizing the max-clique!
- Construct junction tree
  - find the max-cliques from the triangulated graph
  - find the maximal spanning tree in the clique tree
- Propagate probabilities

Maximizing instead of Summing

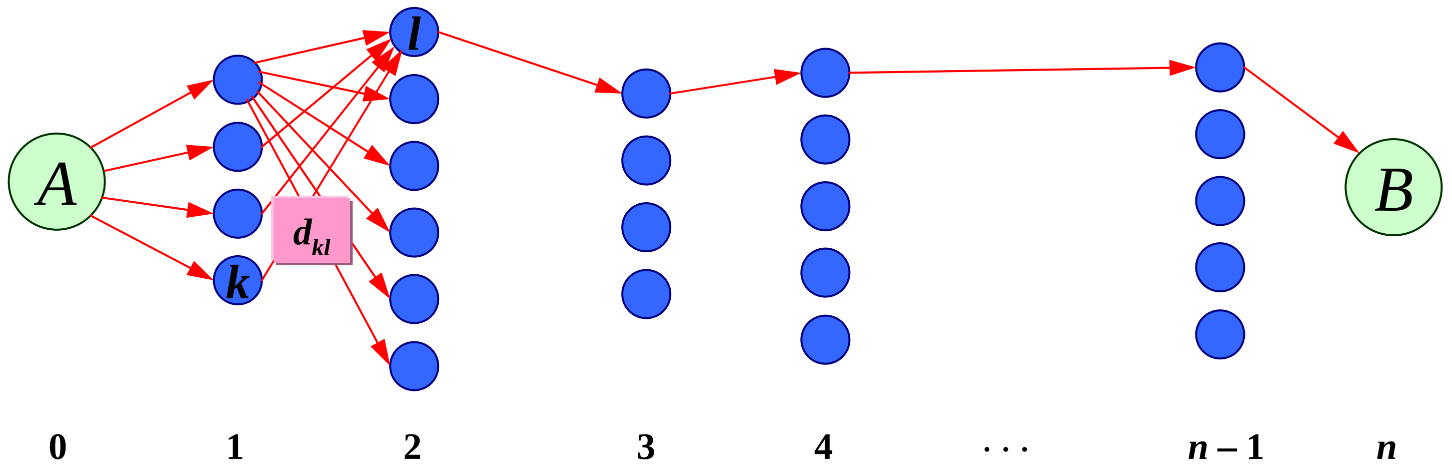
- Elimination algorithm and Belief Propagation both summed over all possible values of the marginal (non-query, non-evidence) nodes to get a marginal probability
- What if we wanted to maximize over the non-query, non-evidence nodes to find the probability of the single best setting consistent with any query and evidence?

$$\begin{aligned}\max_x p(\mathbf{x}) &= \max_{x_1}\max_{x_2}\max_{x_3}\max_{x_4}\max_{x_5} P(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2,x_5)\\ &= \max_{x_1} p(x_1)\max_{x_2} p(x_2|x_1)\max_{x_3} p(x_3|x_1)\max_{x_4} P(x_4|x_2)\max_{x_5} p(x_5|x_3)p(x_6|x_2,x_5)\end{aligned}$$

- This is known as the maximum a-posteriori or MAP configuration
- It turns out that (on trees), we can use an algorithm exactly like belief-propagation to solve this problem



The Shortest Path



- Given: a multilayer network, with distance d_{kl} shown on edges between nodes in neighboring layers.
- Problem: find the shortest path from A to B

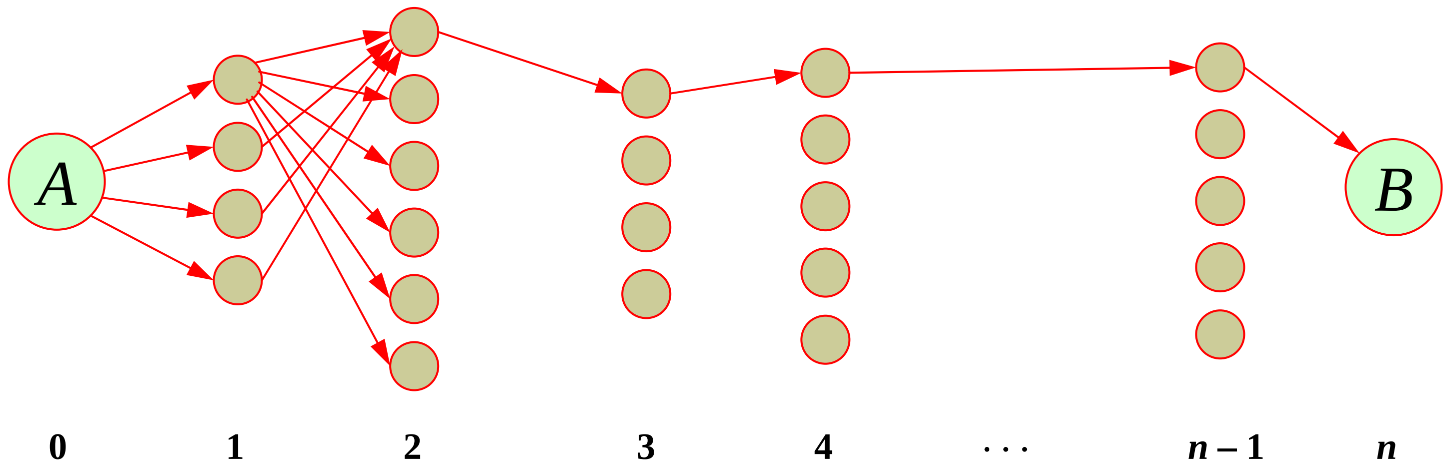
A Dynamic Programming Approach

- All paths have the same start state A, so $v_k(0) = 0$. By keeping pointers backwards, the actual path can be found by backtracking. The full algorithm:

$$\begin{aligned}
&\text{Init } (i=0): && v_k(0) = 0 \text{ for } k \geq 0 \\
&\text{Recursion}(i=1..n): && v_l(i) = \min_k (v_k(i-1) + d_{kl}) \\
& && ptr_i(l) = \arg\min_k (v_k(i-1) + d_{kl}) \\
&\text{Termination}: && P(Y, \pi^*) = \min_k (v_k(n) + d_{k0}) \\
& && \pi_n^* = \arg\min_k (v_k(n) + a_{k0}) \\
&\text{Traceback}(i=n..1): && \pi_{i-1}^* = ptr_i(\pi_i^*)
\end{aligned}$$

(Note : end state is assumed; also,
the begin and end states are numbered zero)

The Most Probable Path



- Given: a multilayer network, with transition probability a_{kl} shown on edges.
- Problem: find the most probable path from A to B
- The solutions can be given by Viterbi algorithm, using dynamic programming

The Viterbi Algorithm

- All paths have the same start state A, so $v_0(0) = 1$. By keeping pointers backwards, the actual path can be found by backtracking. The full algorithm:

$$\begin{aligned}
&\text{Init } (i=0): && v_0(0)=1, v_k(0)=0 \text{ for } k>0 \\
&\text{Recursion}(i=1..n): && v_l(i)=\max_k (v_k(i-1)a_{kl}) \\
& && ptr_i(l)=\arg\max_k (v_k(i-1)a_{kl}) \\
&\text{Termination}: && P(Y,\pi^*)=\max_k (v_k(n)a_{k0}) \\
& && \pi_n^* = \arg\max_k (v_k(n)a_{k0}) \\
&\text{Traceback}(i=n..1): && \pi_{i-1}^* = ptr_i(\pi_i^*)
\end{aligned}$$

(Note : end state is assumed; also,

the begin and end states are numbered zero)

Commutative Semi-ring

- A set K, together with two binary operations called “+” and “·”, which satisfy the following three axioms:

1. The operations “+” is associative and commutative, and there is an additive identity element called “0” s.t. $k + 0 = k$, $\forall\ k$
2. The operation “·” is also associative and commutative, and there is a multiplicative identity element called “1” s.t. $k \cdot 1 = k$, $\forall\ k$
3. The distributive law holds, i.e.

   $(a \cdot b) + (a \cdot c) = a \cdot (b + c)$, $\forall\ a, b, c$ from K

- A semi-ring is a commutative ring without the additive inverse

Example: max-product

- $K = \boldsymbol{R}^+ = [0, +\infty)$, "+": max, "·": usual multiplication

1. Checking the operations "+"

   $\max(k, 0) = k, \forall\ k$; i.e., identity: 0

2. Checking the operation "·"

   $k \cdot 1 = k, \forall\ k$; i.e., identity: 1

3. The distributive law holds, i.e.

   $\max\{a \cdot b, a \cdot c\} = a \cdot \max(b, c)$, $\forall\ a, b, c$ from K

- Other examples: min-product, min-sum, max-sum, etc.

Building Graphical Models

Building Graphical Models

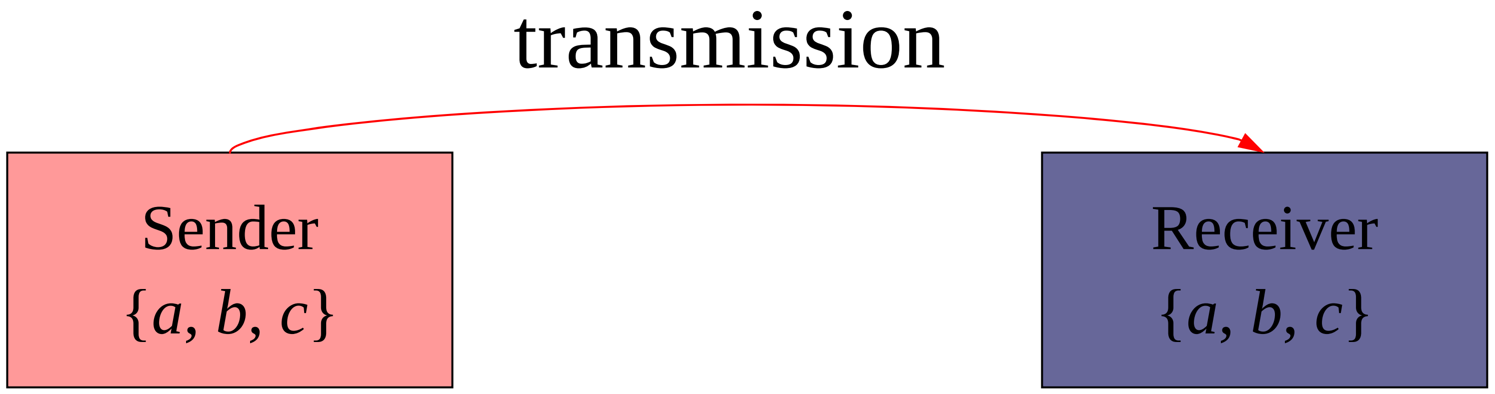
- Tasks of building models
  - Catching the model structure
  - Determining the parameters (conditional probabilities or potential functions)
- What deciding the models?
  - Theoretical considerations
    - e.g.: mixed data, overfitting, etc.
  - A set of data
  - Subjective views from experts (can be partially provided!)

Some Issues in Learning Structure & Parameters

- Nodes as variables
  - Introducing latent variables?
  - How to deal with partially observable variables?
- How many edges?
  - We can always start from a complete graph, but it may not be a good idea!
  - Edge missing can be treated as an edge of probability zero
- Undirected, directed or hybrid, and how to decide the direction for the directed case?
  - Direction may not reflect causality
- Complexity vs. training error: the principle of Minimum Description Length

Example: Strings of Symbols

- A language L over 2 symbols $\{a, b\}$ is transmitted through a channel. Each word is surrounded by the delimiter symbol c. In the transmission some characters may be corrupted by noise and be confused with others.



Modeling by HMM

- A model M_{Simp} for symbol transmission: T_i are the symbols transmitted and R_i are the symbols received.

$$\begin{tikzcd}
T_1 \arrow[r, "A"] \arrow[d, "E"] & T_2 \arrow[r] \arrow[d] & T_3 \arrow[r] \arrow[d] & T_4 \arrow[r] \arrow[d] & T_5 \arrow[d] \\
R_1 & R_2 & R_3 & R_4 & R_5
\end{tikzcd}$$

	$T = a$	$T = b$
$R = a$	0.80	0.15
$R = b$	0.10	0.80
$R = c$	0.10	0.05

$P(R \mid T)$: under transmission

	$T_1 = a$	$T_1 = b$
$T_2 = a$	0.6	0.4
$T_2 = b$	0.4	0.6

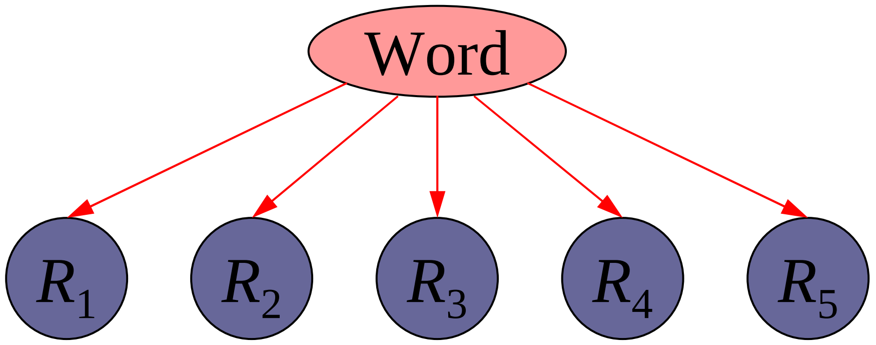
$P(T_2 \mid T_1)$

	$T_2 = a$	$T_2 = b$
$T_3 = a$	0.24	0.74
$T_3 = b$	0.76	0.26

$P(T_3 \mid T_2)$

A More Complicated Design

- An alternative model for symbol transmission:



First 2 letters	Last 3 letters							
	aaa	*aab*	*aba*	*abb*	*baa*	*bab*	*bba*	*bbb*
aa	0.017	0.021	0.019	0.019	0.045	0.068	0.045	0.068
ab	0.033	0.040	0.037	0.038	0.011	0.016	0.010	0.015
ba	0.011	0.014	0.010	0.010	0.031	0.046	0.031	0.045
bb	0.050	0.060	0.056	0.057	0.016	0.023	0.015	0.023

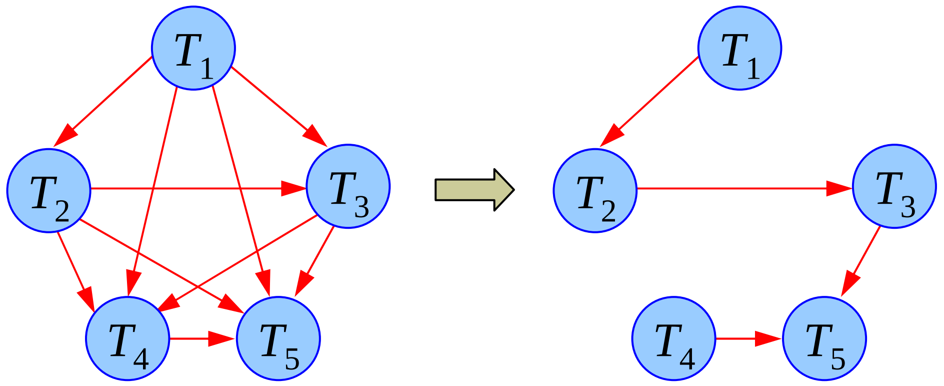
$P(T_1, T_2, \ldots, T_5)$: join probabilities of five-letter words in L

Acceptance Measure of a Model

- $Acc(P, M^*) = size(M^*) + k \cdot dist(P, P^*)$
  - $size(M^*)$: size of a model
  - $dist(P, P^*)$: a distance measure
  - k : weight of the distance measure

Search for Possible Structures

- We cannot investigate all possible DAGs.
  - For an m-variable distribution, there are more than $2^{m(m-1)/2}$ possible DAGs
- Therefore, we can only search in a subset of possible DAGs



- Approach I:
  1. Starting from complete DAG (most complex)
  2. Repeatedly remove links until $dist(P, P^*) > dist\_threshold$
- Approach II:
  1. Starting from the graph with only singletons
  2. Repeartely add links until $size(M^*) > size\_threshold$

Practical Issues

- Impossible to start from the complete DAG because the model grows exponentially with the number of random variables
  - Start from a simpler model
  - Allow add, remove, and reverse a link
- Predefined node ordering to reduce the search space
- KL divergence is more suitable for large models.
- How to handle missing values?
- …

Rules of Thumb

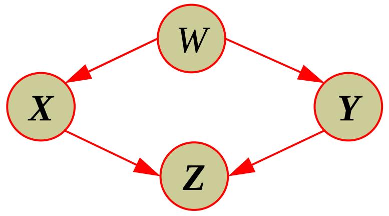
- Parameter estimation
  - For completely observed model, we can use Maximum Likelihood criteria to estimate the parameters: counting the frequencies
  - For models with latent variable, the common approach is EM algorithm
- Structure learning
  - In many cases, model structure is ready for us. The only task then is parameter estimation. E.g., regular Markov random field for computer vision problems
  - Generative models (more biased) vs. discriminative models (less biased, but harder to learn)
- Undirected models are hard to learn compared to directed models
  - Because we need to deal with the partition function

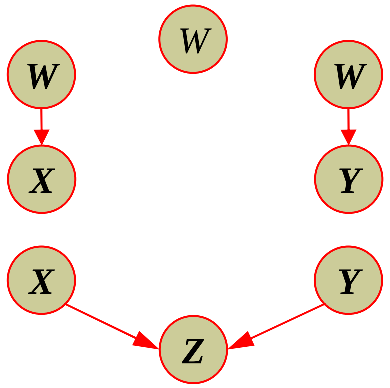
MLE for Directed GMs

- For a directed GM, the likelihood function has a nice form:

$$\log P(D|\theta) = \log \prod_m \prod_i P(X_i^m|\mathbf{X}_{\pi_i}, \theta_i) = \sum_m \sum_i \log P(X_i^m|\mathbf{X}_{\pi_i}, \theta_i)$$

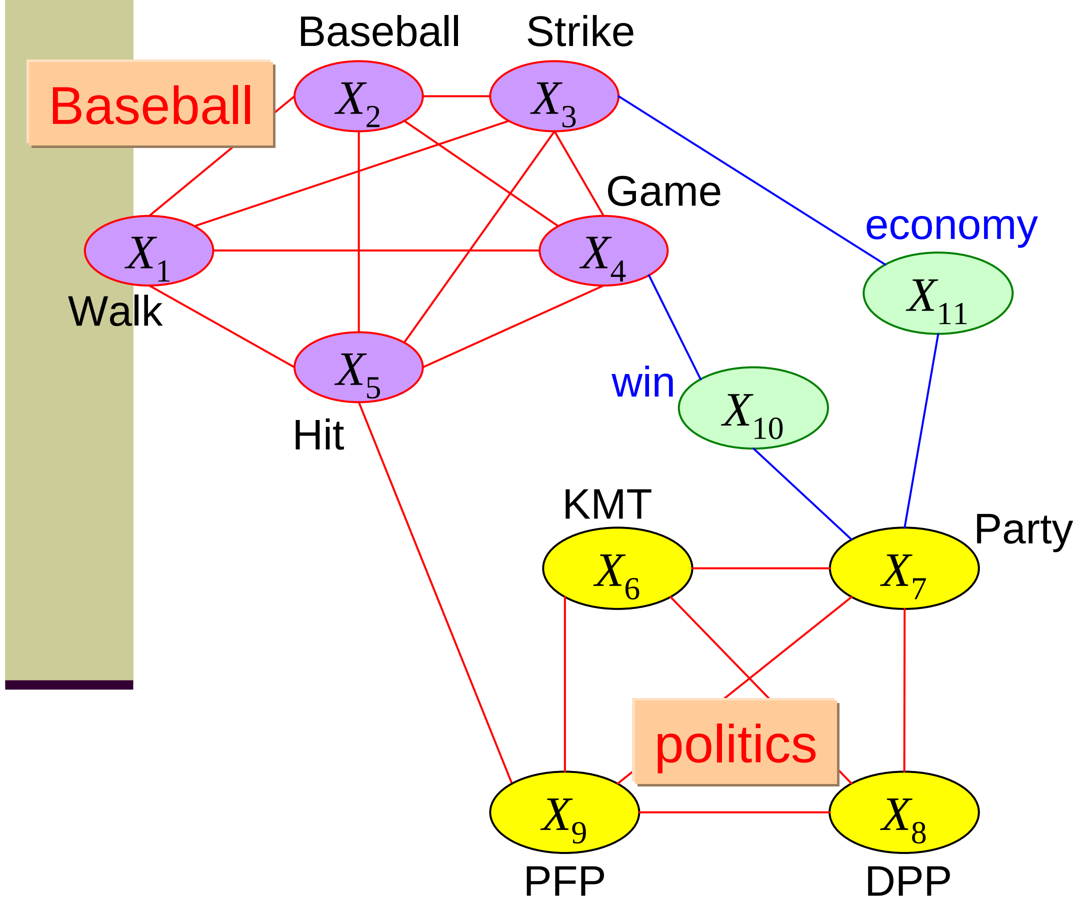
- The parameters decouple; so we can maximize likelihood independently for each node's function by setting θ_i
- Only need the values of x_i and its parents in order to estimate θ_i
- Furthermore, if $X^m, \mathbf{X}_{\pi_i}$ have sufficient statistics only need those.
- In general, for fully observed data if we know how to estimate parameters at a single node we can do it for the whole network.





The Applications of Graphical Models

Graphical Models Applied in Text Mining



- Each maximal clique can be treated as one concept
- Separators between maximal cliques act as filters, once they are observed, (conditional) independence between cliques is established!
- The junction tree algorithm can help us to do inference

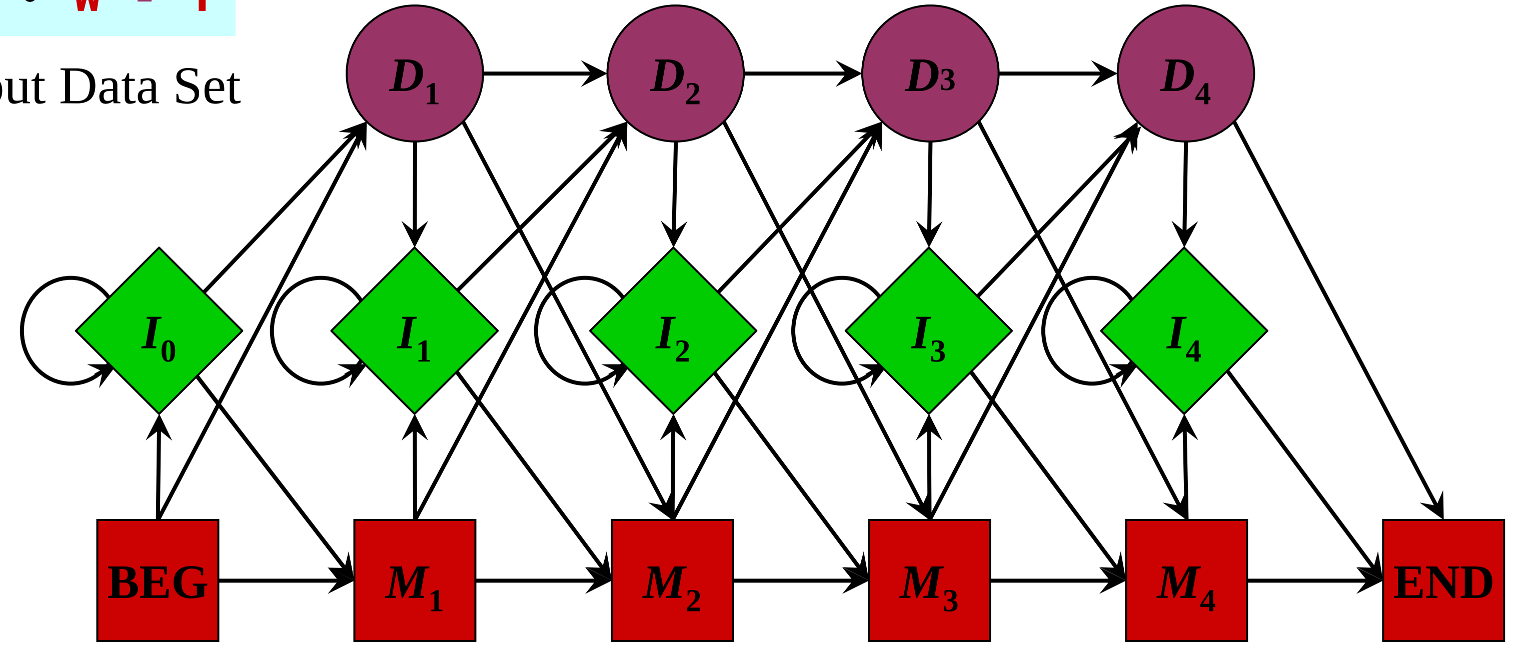
  (The graph is inspired by Lexical FreeNet)

HMM for Sequence Alignment

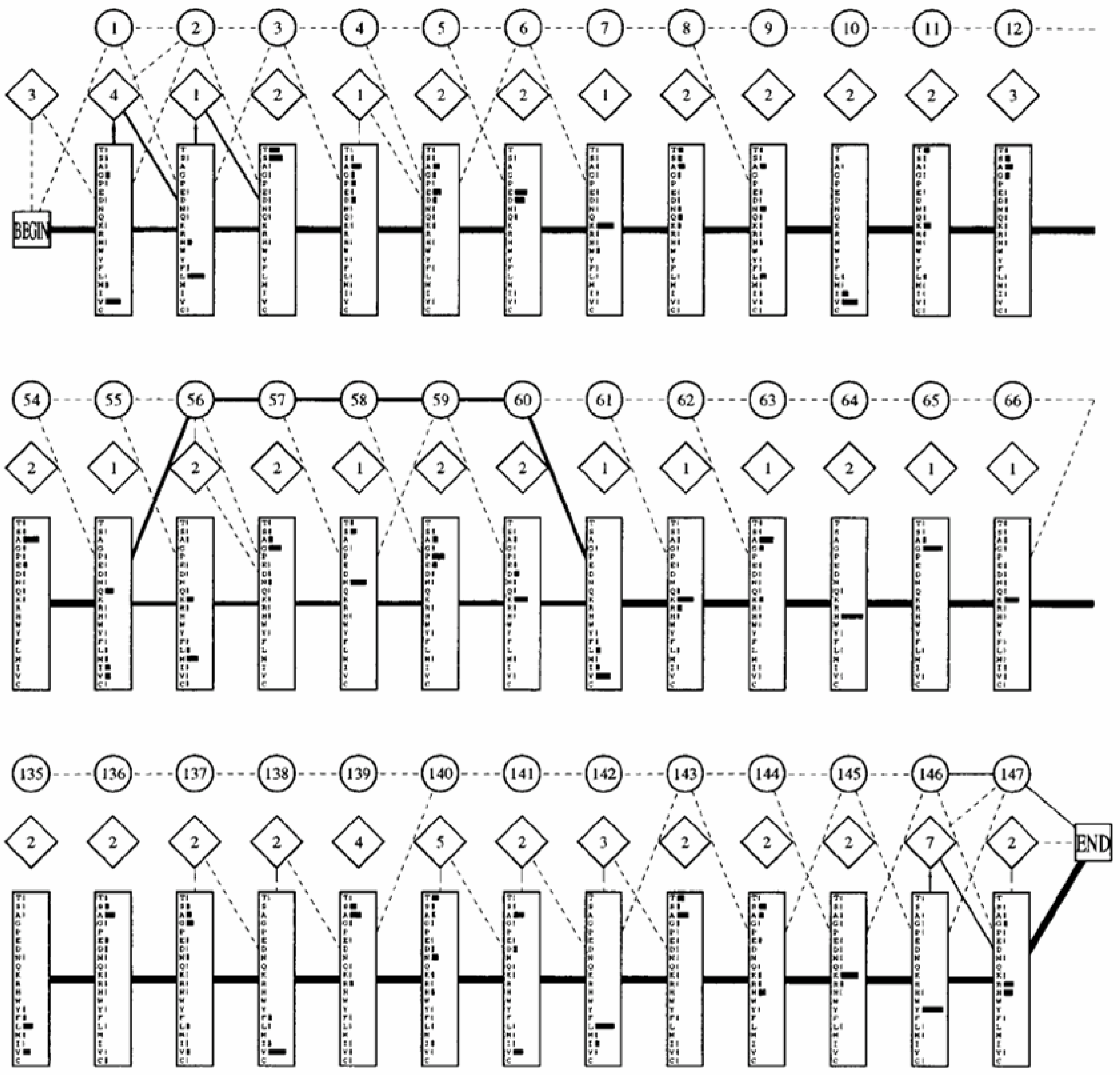
N	•	F	L	S
N	•	F	L	S
N	K	Y	L	T
Q	•	W	-	T

Input Data Set

Red position represents alignment in column
Green position represents insert in column
Purple position represents delete in column

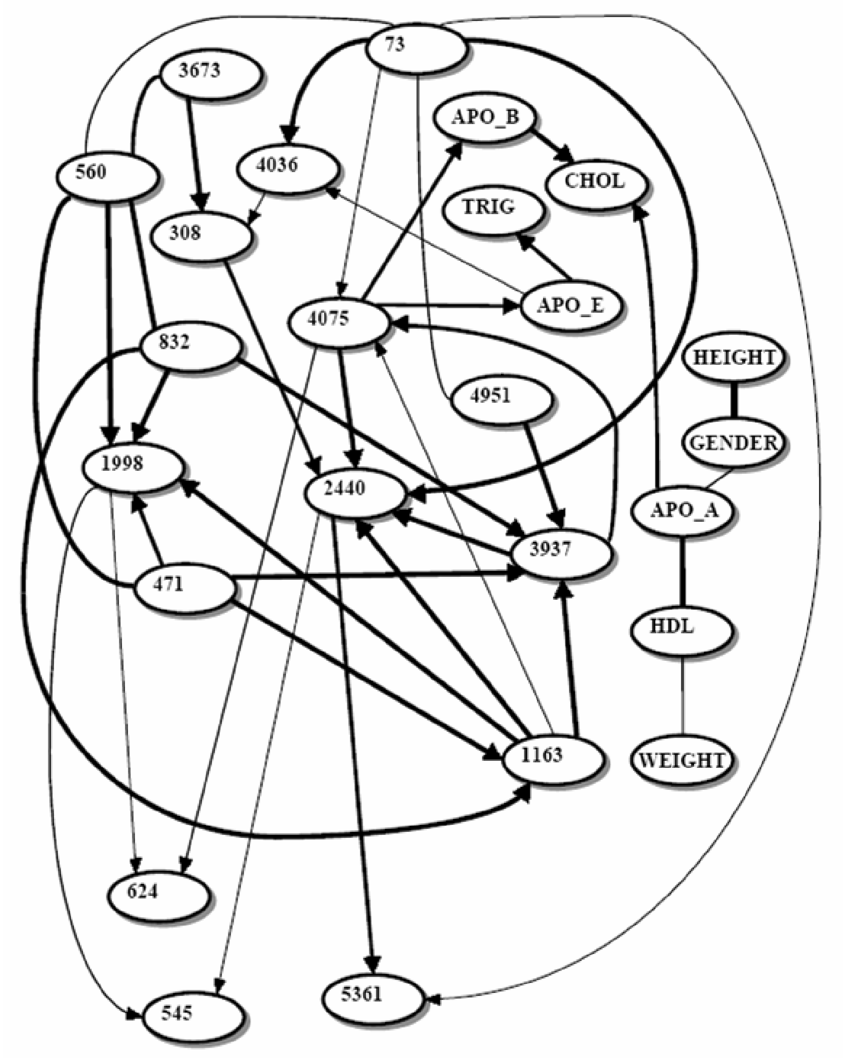


🟥 match state 🟩 insert state ● delete state → transition probability



BEGIN
END

Bayesian Networks to Resolve Genotype-to-phenotype Relations

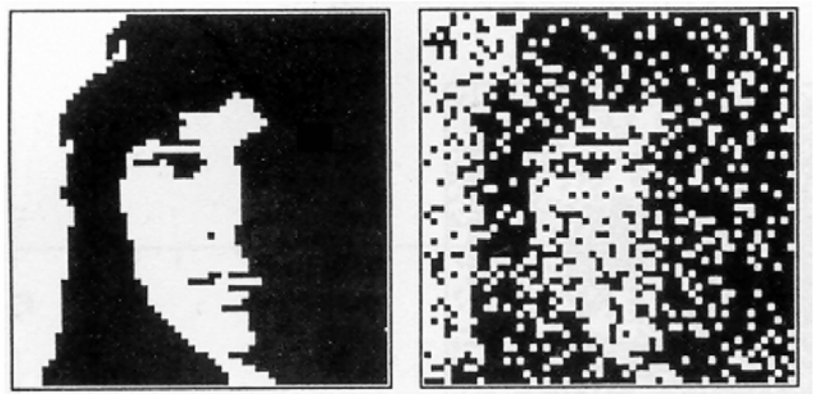


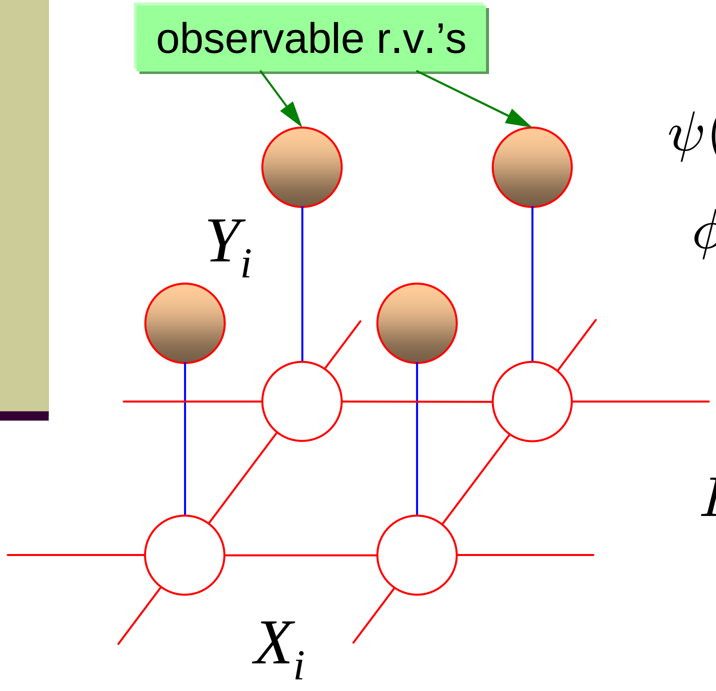
- Learned BN relating apolipoprotein E gene SNPs to plasma **apoE** levels in Jackson, MS. Node legends: numbers refer to corresponding SNPs (see Figure 1 in Nickerson *et al*. (2000) for an SNP map.) APO_E, APO_A, APO_B, TRIG, CHOL and HDL stand for levels of polipoproteins E, AI and B, triglycerides, cholesterol and HDL cholesterol, respectively
- Undirected edges indicate dependencies, directed edges indicate possible causations
- Line thickness corresponds to the relative edge strength

(Andrei S. Rodin and Eric Boerwinkle, bioinformatics, 2005)

(Hidden) Markov Random Field

- (Hidden) Markov Random Field can apply to image labeling, for instance, segmentation, figure/ground separation, etc.



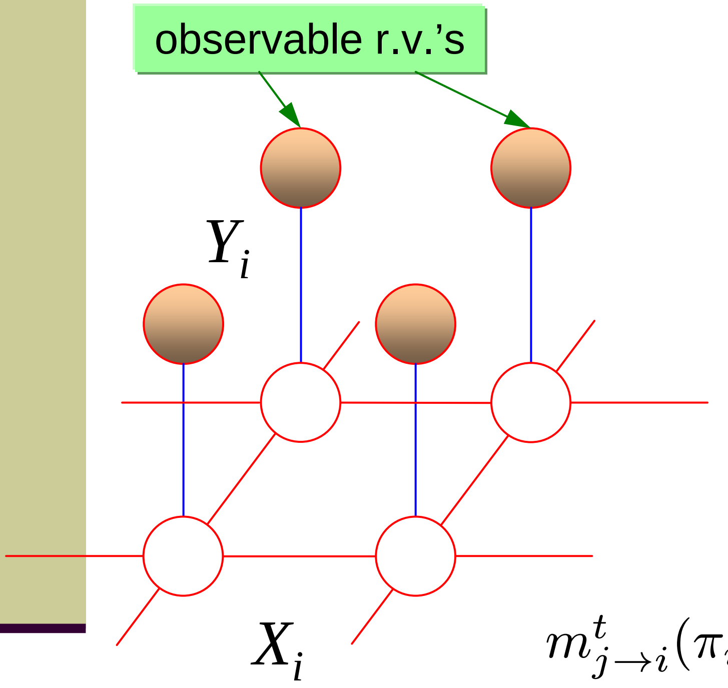


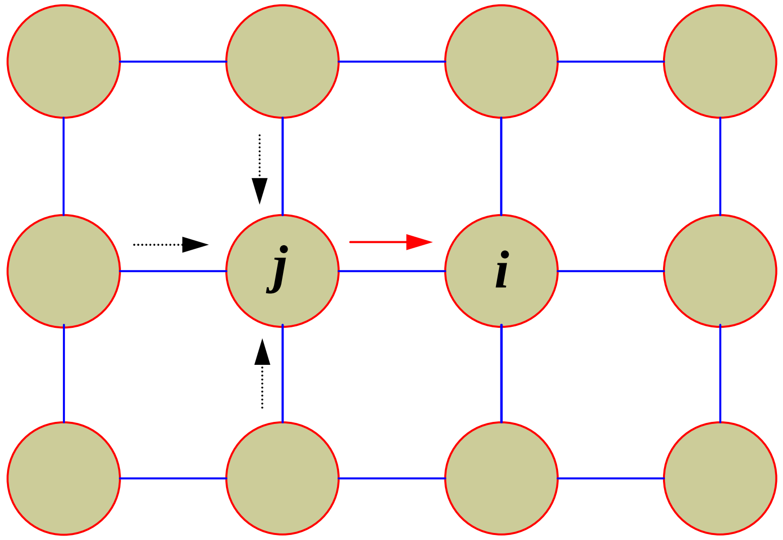
$$\psi(X_i, X_j) = \exp^{-|d_i - d_j|^2/2\sigma^2}$$

$$\phi(X_i, Y_i) = \exp^{-|y_i - y(x_i)|^2/2\sigma^2}$$

$$P(\underset{\text{scene}}{\mathbf{X}}, \underset{\text{image}}{\mathbf{Y}}) = \frac{1}{Z} \prod_i \underbrace{\phi_i}_{\text{image-scene compatibility function}}(\underbrace{X_i, Y_i}_{\text{local observations}}) \prod_{i,j} \underbrace{\psi_{ij}}_{\text{scene-scene compatibility function}}(\overbrace{X_i, X_j}^{\text{neighboring scene nodes}})$$

Loopy Belief Propagation



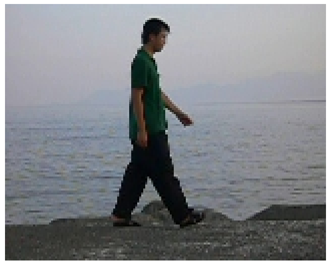
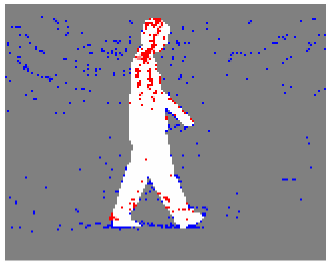


$$m^t_{j\to i}(\pi_i) = \min_{\pi_j}\left(V(\pi_j - \pi_i) + \sum_{k\in\mathcal{N}(j)\backslash i} m^{t-1}_{k\to j}(\pi_j)\right)$$

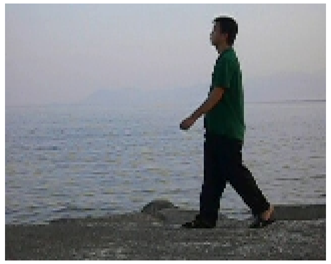
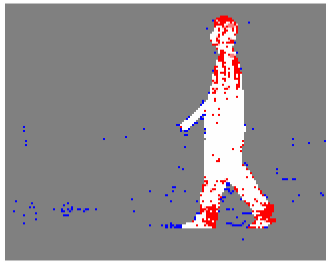
$$b_i(\pi_i) = \sum_{j\in\mathcal{N}(i)} m^T_{j\to i}(\pi_i)$$

Segmentation of Moving Objects

	TP rate	TN rate	Error rate
Training	90.75%±1.32%	98.86%±0.44%	1.86%±0.43%
Test	70.64%±7.74%	99.17%±0.10%	3.38%±0.80%

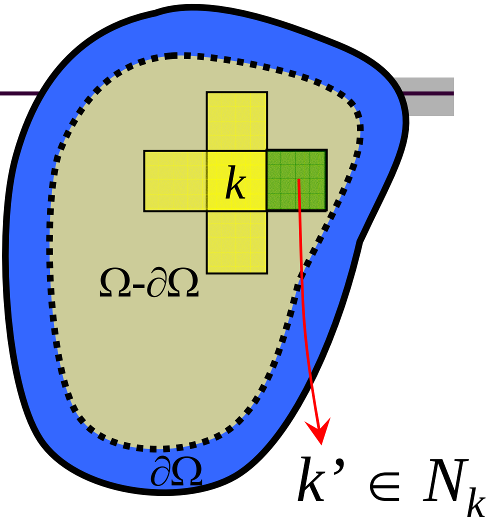



Training

Test

Energy Minimization



- Given σ_0 we define the functional as

$$E(\sigma|\sigma_0) = E_{data}(\sigma|\sigma_0) + E_{smooth}(\sigma)$$

where

$$E_{data}(\sigma|\sigma_0) = \sum_{k \in \partial\Omega} \frac{1}{\epsilon}(\sigma(k) - \sigma_0(k))^2 + \sum_{k \notin \partial\Omega} \lambda \cdot \sigma^2(k)$$

and

$$E_{smooth}(\sigma) = \sum_{k=1}^{K} \sum_{k' \in N_k} (\sigma(k) - \sigma(k'))^2$$

Energy Minimization vs. MLE

- By choosing the Gaussian function as the potentials, the solution to energy minimization is exactly the solution to MLE
- We can freely choose among:
  - Diffusion process / random walk
  - Belief propagation

  to obtain the solution
- However, for tree structured model (not for regular MRF which has loops involved), belief propagation can converge in finite steps, which can not be done via random walk or other numerical methods (e.g., conjugate gradient descent)

Conclusions

- Graphical models provide an tractable approach which is at the same time complicated enough to describe the real world phenomenon
- Graphical models provide a unified framework for many machine learning approaches, including Naïve Bayes, HMM, Markov random field, PCA, etc
- Learning and Inference can be the key step to decide whether or not our built models are successful!
- Graphical models are broadly used in many fields including bioinformatics, text mining, computer vision, etc