



Bayesian networks

鮑興國 Ph.D.

National Taiwan University of
Science and Technology

Outline

- Introduction to Bayesian networks
- Bayesian networks: definition, d-separation, equivalence of networks
- Examples: causal graphs, explaining away, Markov chains, Naïve Bayes, etc
- Undirected models
- Probabilistic inference
 - node elimination
 - junction tree
- Building the networks

Probabilistic Graphical Models

- Combination of **graph theory** and **probability theory**
- Informally,
 - Graph structure specifies which parts of system are directly **dependent**
 - **Local functions** at each node specify how parts interact
- More formally,
 - Graph encodes **conditional independence** assumptions
 - Local functions at each node are **factors in the joint probability distribution**
- E.g. 1: **Bayesian networks** = PGMs based on **directed acyclic graphs**
- E.g. 2: **Markov networks (Markov random fields)** = PGM with **undirected graph**
- Others: Discrete models, continuous models, Gaussian graphical models, hybrid Bayesian networks, CG models, etc

Applications of PGMs

- Machine learning
- Statistics
- Speech recognition
- Natural language processing
- Computer vision
- Error-control codes
- Bioinformatics
- Medical diagnosis
- Financial predictions
- etc

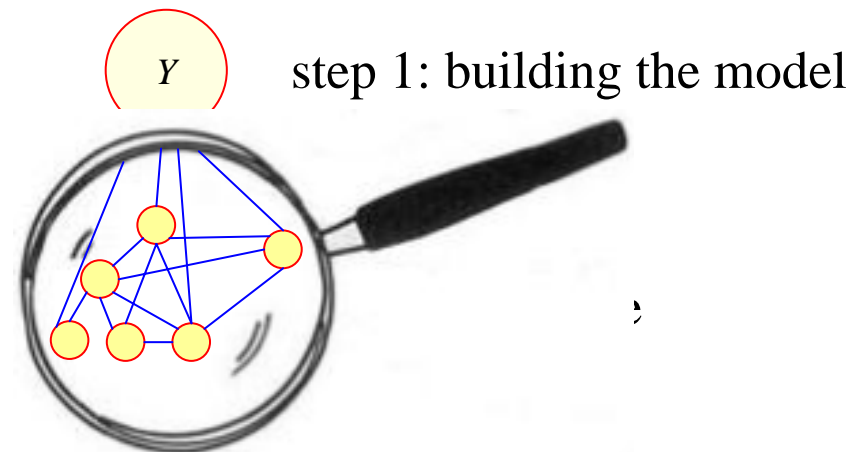
Why Probability?

- Why probabilistic graphical models?
- Why probabilistic approach can help us in machine learning?
- *Guess 1*: world is nondeterministic or
- *Guess 2*: world is deterministic, but **lacking of relevant facts/attributes** to decide the answer
 - ⇒ probabilistic model can make **inferences about missing inputs**
 - ⇒ **classification is one example (missing is “y”)**!
- Either way, **probabilistic approach can make decisions which minimize expected loss**

Probabilistic Approach

- Classification and regression: conditional density estimation $P(Y | X)$
 - then, by Bayes rule...
- Unsupervised learning: density estimation $P(X)$
- Imaging Y as a latent variable, classification is no more than missing value filling

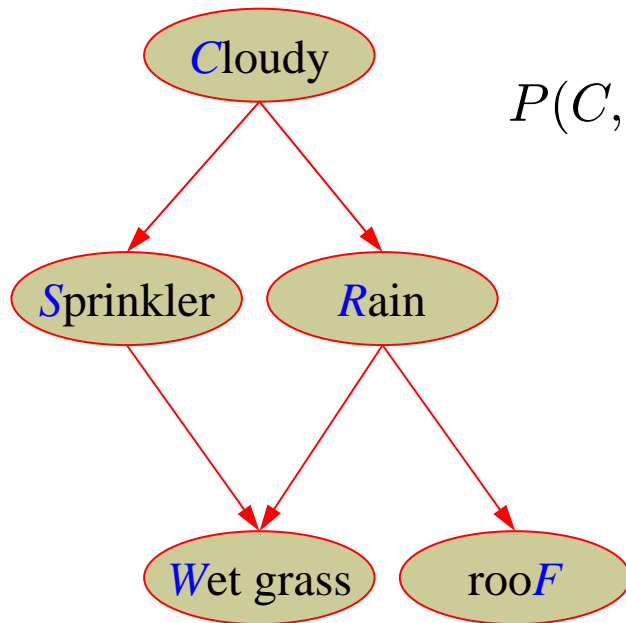
$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_N^{(1)} & y \\ x_1^{(2)} & x_2^{(2)} & \dots & x_N^{(2)} & y \\ \vdots & \vdots & & \vdots & \\ x_1^{(d)} & x_2^{(d)} & \dots & x_N^{(d)} & y \\ x_1^{(d+1)} & x_2^{(d+1)} & \dots & x_N^{(d+1)} & - \\ \vdots & \vdots & & \vdots & \\ x_1^{(d+k)} & x_2^{(d+k)} & \dots & x_N^{(d+k)} & - \end{bmatrix}$$



Why Graphical Models?

- We discuss this issue later on...
- Basically, graphical models help us to deal with complicated problem modularized and in a visualized way
- That is, breaking problems to subproblems recursively until we can solve them

Welcome to Hotel California!



$$\begin{aligned}
 P(C, S, R, W, F) &= P(C)P(S|C)P(R|C, S) \\
 &\quad P(W|C, S, R)P(F|C, S, R, W) \\
 &= P(C)P(S|C)P(R|C) \\
 &\quad P(W|S, R)P(F|R)
 \end{aligned}$$

actually...

Conditional Probability Table

$P(W R, S) = 0.95$
$P(W R, \sim S) = 0.90$
$P(W \sim R, S) = 0.90$
$P(W \sim R, \sim S) = 0.10$

$$\begin{aligned}
 P(R|C, S) &= P(R|C) \\
 P(W|C, S, R) &= P(W|S, R) \\
 P(F|C, S, R, W) &= P(F|R)
 \end{aligned}$$

Joint Probability

- *Goal 1:* **represent** a joint distribution $P(\mathbf{X} = \mathbf{x}) = P(X_1=x_1, \dots, X_n=x_n)$ **compactly** even when there are many variables
- *Goal 2:* efficiently **calculate** marginal and conditionals of such compactly represented joint distribution
- For n discrete variables of arity k , the naïve (table) presentation is HUGE: it requires k^n entries
- We need to make some assumptions about the distribution
 - One simple assumption: independence = complete factorization

$$P(\mathbf{X}) = \prod_i P(X_i)$$

- But the independence assumption is too restrictive. So we make **conditional independence** assumption instead

Conditional Independence

- Notation: $X_A \perp X_B \mid X_C$

Definition: two (sets of) variables X_A and X_B are conditionally independent given a third X_C if:


$$P(X_A, X_B \mid X_C) = P(X_A \mid X_C) P(X_B \mid X_C) \quad \forall X_C$$

which is equivalent to saying

$$P(X_A \mid X_B, X_C) = P(X_A \mid X_C) \quad \forall X_C$$

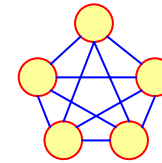
- Only a subset of all distribution respect any given (nontrivial) conditional independence statement. The subset of distributions that respect all the CI assumptions we make is **the family of distributions consistent with our assumptions**
- Bayesian networks (probabilistic graphical models) are a powerful, elegant and simple way to specify such a family

Between Simple and Complex Models

- Given n r.v.'s X_1, X_2, \dots, 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
- ???  Bayesian networks
- independent
 - $df = 2n (P(X_i=1), P(X_i=2))$
- i.i.d.
 - $df = 2 (P(X_1=1), P(X_1=2))$

Bayesian Belief networks

- Bayesian networks (graphical models) is an intermediate approach
 - ○ ■ i.i.d. assumption too restrictive
 - ○ ■ the most general cases ineffective
- BN (GM) represent large joint distributions compactly using a set of “local” relationships specified by a graph. The joint probability then can be factorized into such local relationships
- BN describes
 - *conditional independence* among subsets of variables
 - conditional probabilities \leftrightarrow joint probabilities (Bayes theorem)
- **Bayesian networks** (or directed models) vs. **Markov random fields** (MRF, or undirected models)
 - Belief networks using *conditional pr.*, while
 - Markov random fields using *potential function*





Part I



Bayesian Networks: Introduction to
Directed Models and Undirected Models

Outline

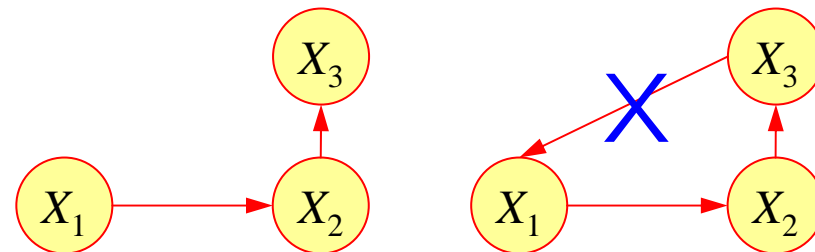
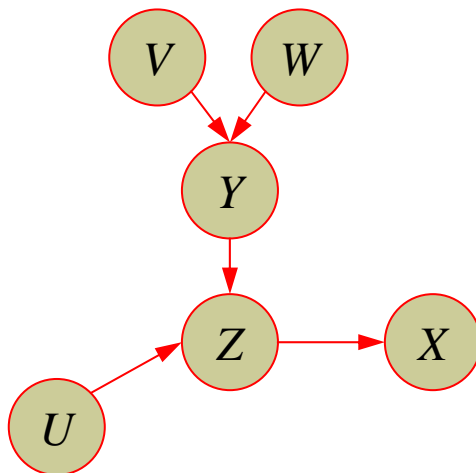
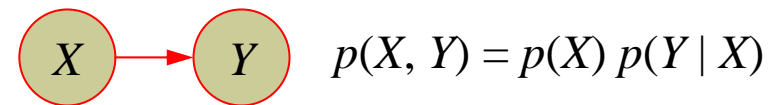
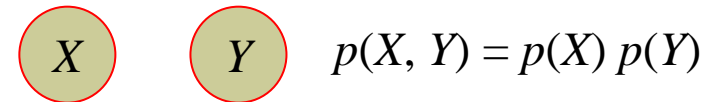
- Introduction to Bayesian networks
- Bayesian networks: definition, d-separation, equivalence of networks
- Examples: causal graphs, explaining away, Markov chains, Naïve Bayes, etc
- Undirected models
- Probabilistic inference
 - node elimination
 - junction tree
- Building the networks

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, \dots, Y_n , there is attached the **conditional probability table** $P(X | Y_1, Y_2, \dots, Y_n)$

Bayesian Networks (cont.)

- Informally, **edges** represent “**causation**”
- The variables together with the directed edges form a **Directed Acyclic Graph** (no cycles allowed)



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

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

- Define the joint probability to be:

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

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

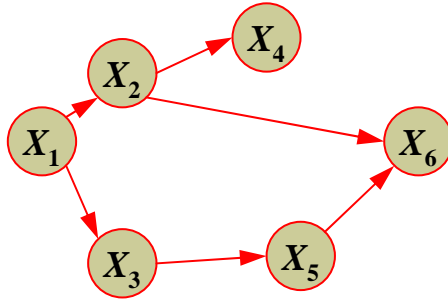
$$f_i(X_i; \mathbf{X}_{\pi_i}) = P(X_i | \mathbf{X}_{\pi_i})$$

- so we will just write

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

- 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...



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

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

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

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

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

$$= \sum_{X_1, X_3, X_4} f_1(X_1)f_2(X_2; X_1)f_3(X_3; X_1)f_4(X_4; X_2)f_5(X_5; X_3)$$

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

Conditional Independence in DAGs

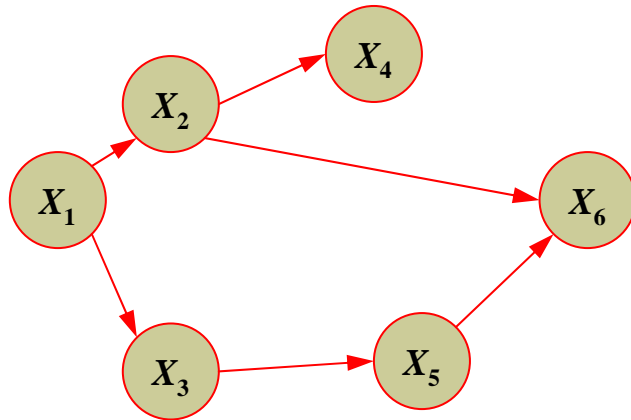
- If we order the nodes in a directed graphical model so that parents always come before their children in the ordering then the graphical model implies the following about the distribution:

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

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 again!



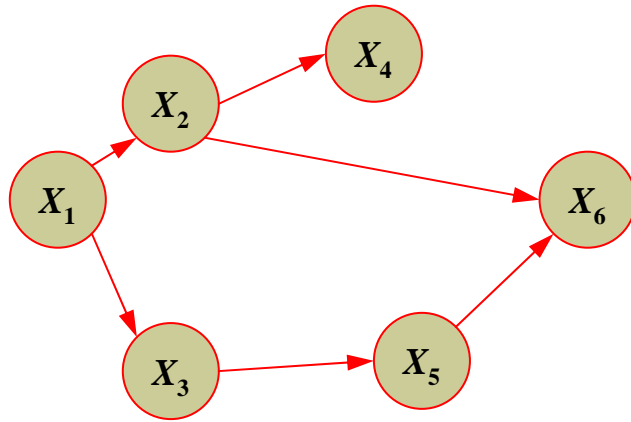
Prove: $X_4 \perp X_1 \mid X_2$

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

$$\begin{aligned} P(X_{[1..3]}) &= P(X_1)P(X_2|X_1)P(X_3|X_1) \sum_{X_4} P(X_4|X_2) \sum_{X_5} P(X_5|X_3) \sum_{X_6} P(X_6|X_2, X_5) \\ &= P(X_1)P(X_2|X_1)P(X_3|X_1) \end{aligned}$$

$\Rightarrow P(X_4|X_{[1..3]}) = P(X_4|X_2)$ and $P(X_4|X_1, X_2) = P(X_4|X_2)$ why?

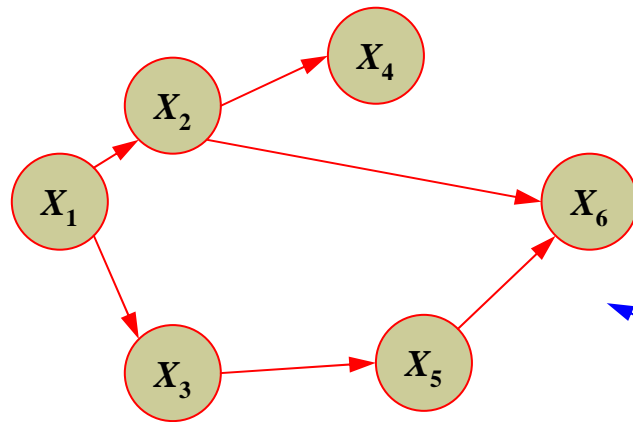
Saving in Time & Space



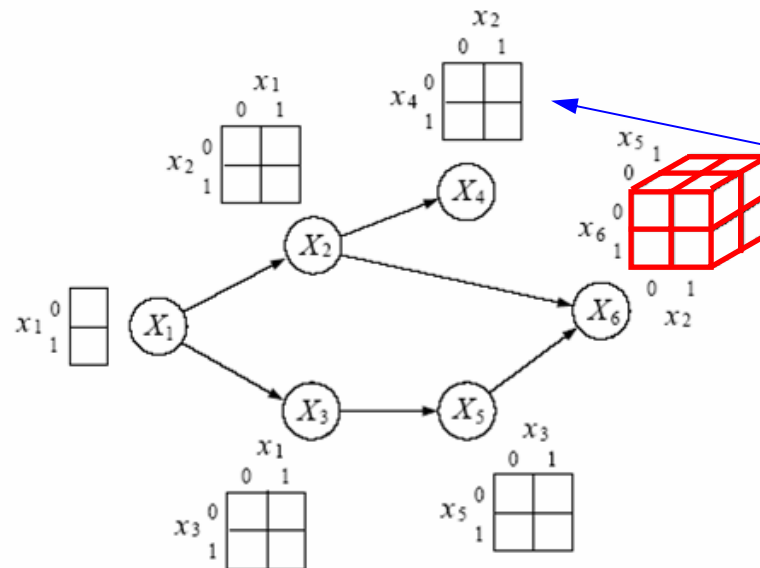
- Conditional independence:
E.g.1a: $X_4 \perp X_1 \mid X_2$
E.g.1b: $X_5 \perp X_2 \mid X_3$
E.g.1c: $X_5 \not\perp X_6 \mid X_3$
- Idea can be extended to sets:
E.g.2a: $X_6 \perp X_3 \mid X_2, X_5$
E.g.2b: $X_5, X_6 \perp X_1 \mid X_2, X_3$

$$\begin{aligned} P(X_{[1..6]}) &= \sum_{X_{[1..6]}} P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_{[1..3]})P(X_5|X_{[1..4]})P(X_6|X_{[1..5]}) \\ &= \sum_{X_{[1..6]}} P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3)P(X_6|X_2, X_5) \end{aligned}$$

Saving in Time & Space (cont.)

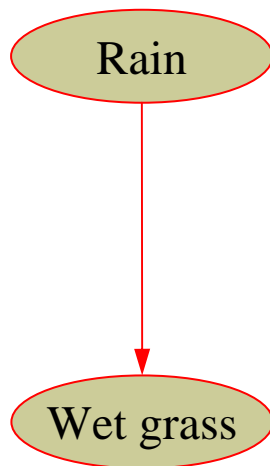


$$\begin{aligned}
 &P(X_1, X_2, X_3, X_4, X_5, X_6) \\
 &= P(X_1)P(X_2|X_1)P(X_3|X_1) \\
 &P(X_4|X_2)P(X_5|X_3)P(X_6|X_2, X_5)
 \end{aligned}$$



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

Example: Causal Graphs



$$P(R) = 0.4$$

$$P(W | R) = 0.9$$
$$P(W | \sim R) = 0.2$$

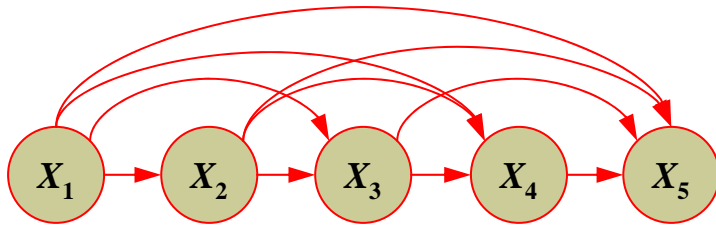
$$P(R|W) = \frac{P(W|R)P(R)}{P(W)}$$

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

- 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

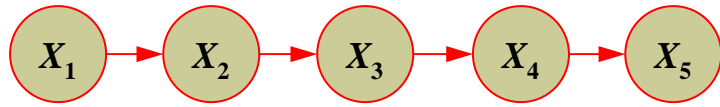
Example: Complete Graph

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



- 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

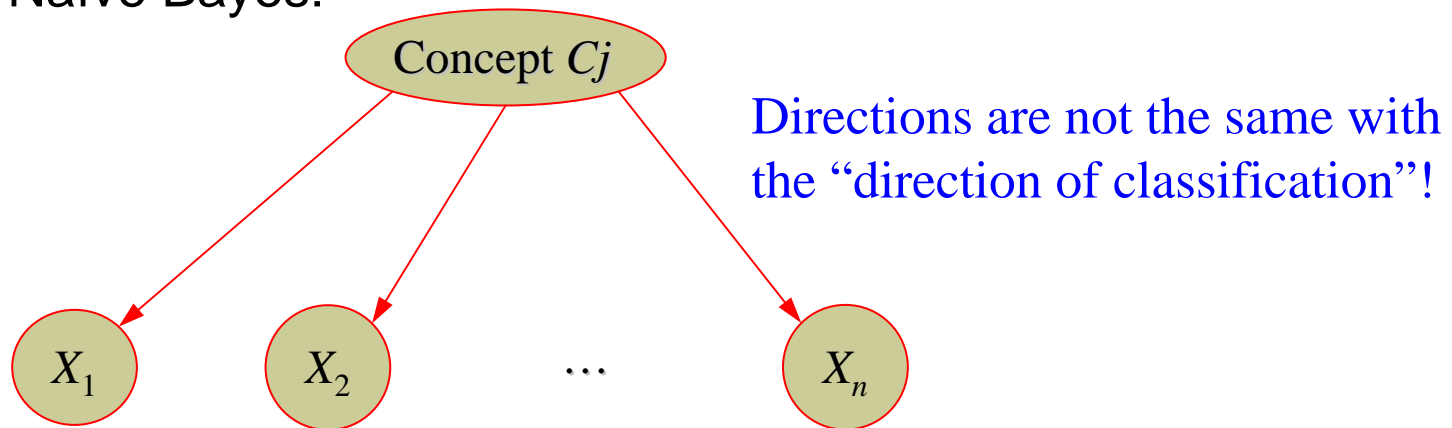
Example: Markov Chain



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

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:

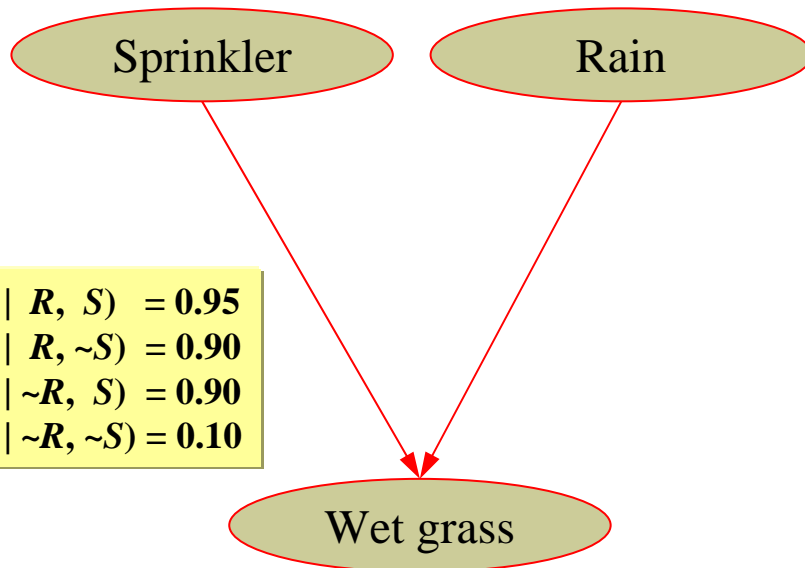


$$P(X_1, X_2, \dots, X_n, C_j) = P(C_j) P(X_1 | C_j) P(X_2 | C_j) \dots P(X_n | C_j)$$

Explaining Away

$$P(S) = 0.2$$

$$P(R) = 0.4$$



$$\begin{aligned} P(W | R, S) &= 0.95 \\ P(W | R, \sim S) &= 0.90 \\ P(W | \sim R, S) &= 0.90 \\ P(W | \sim R, \sim S) &= 0.10 \end{aligned}$$

$$\begin{aligned} P(S|W) &= \frac{P(W|S)P(S)}{P(W)} \\ &= \frac{0.92 \times 0.2}{0.52} = 0.35 \end{aligned}$$

$$\begin{aligned} P(S|R, W) &= \frac{P(W|R,S)P(S|R)}{P(W|R)} \\ &= \frac{P(W|R,S)P(S)}{P(W|R)} = 0.21 \\ &< P(S|W) \end{aligned}$$

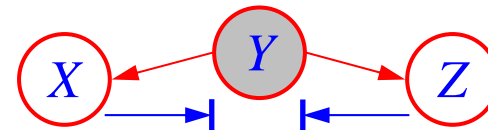
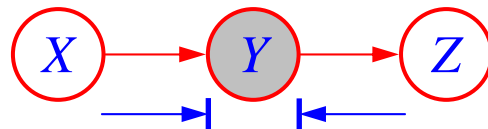
- **Knowing** it rained, the probability that sprinkler is on (to cause the web grass) decreases, **knowing** that the grass is wet
- Berkson's paradox!

Directed-separation: Definition

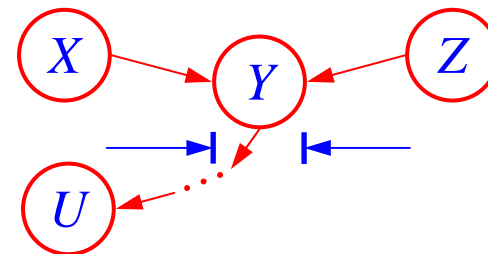
■ Definition (**d-Separation**)

A path p is said to be **d-separated** (or **blocked**) if and only if

- p contains a chain $X \rightarrow Y \rightarrow Z$ or a fork $X \leftarrow Y \rightarrow Z$ and Y is instantiated (shaded)



- p contains an inverted fork (or **collider**) $X \rightarrow Y \leftarrow Z$, and neither Y nor any **descendants** of Y have received evidence



Directed-separation: Definition (cont.)

- A set A is said to be d-separated from a set B if every path from a node in A to a node in B is d-separated
- A and B connected if they are not separated!
- A set A is said to be d-separated from a set B **given a set C** if every path from a node in A to a node in B is d-separated, **given all nodes in C are instantiated!**
- A and B connected given C if they are not separated given C !

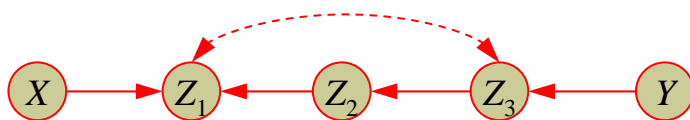
D-separation and (Conditional) Independence

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

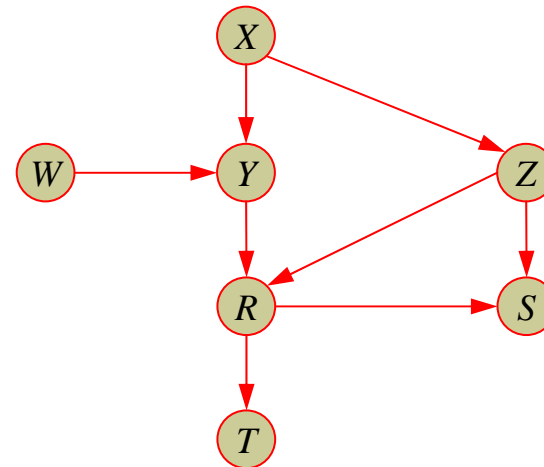
- **(Probabilistic Implications of d-Separation)**

If sets A and B are d-separated given C in a DAG G , then A is independent of B conditional on C in **every** distribution compatible with G . Conversely, if A and B are **not** d-separated given C in a DAG G , then A and B are dependent conditional on C in **at least one** distribution compatible with G .

D-separation: example I



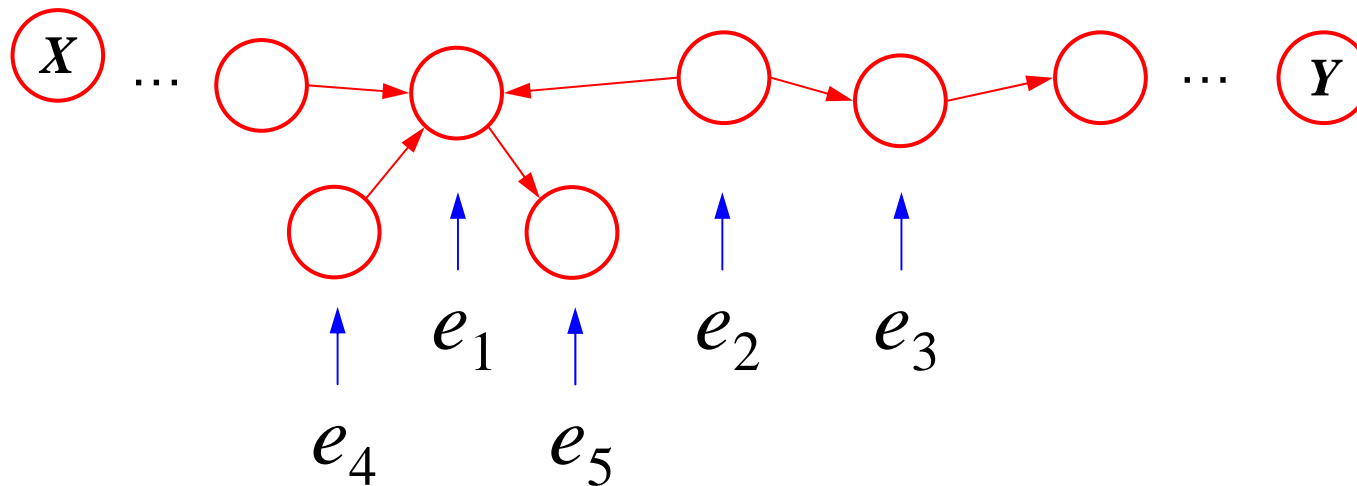
- X and Y are d-separated given Z_2 and d-connected given Z_1



- X and R are d-separated by $\{Y, Z\}$
- X and T are d-separated by $\{Y, Z\}$
- W and T are d-separated by $\{R\}$
- W and X are *not* d-separated by Y
- W and X are d-separated by \emptyset

D-separation: example II

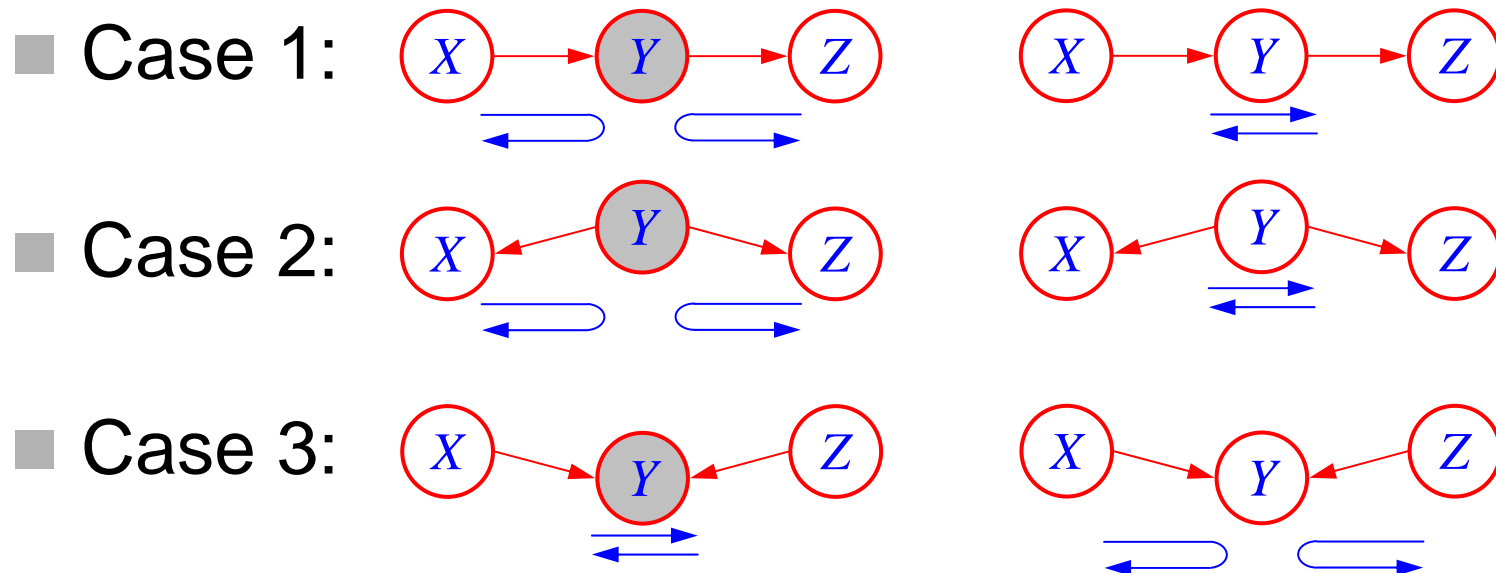
- Is X d-separated from Y ?



- d-separated: given no evidence or e_2 or e_3 or **only** e_4
- d-connected: given **only** e_1 or **only** e_5 (provided the rest path is connected)

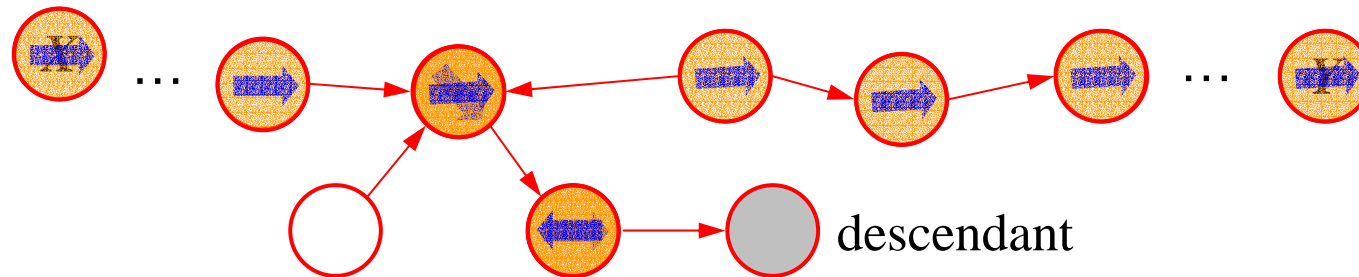
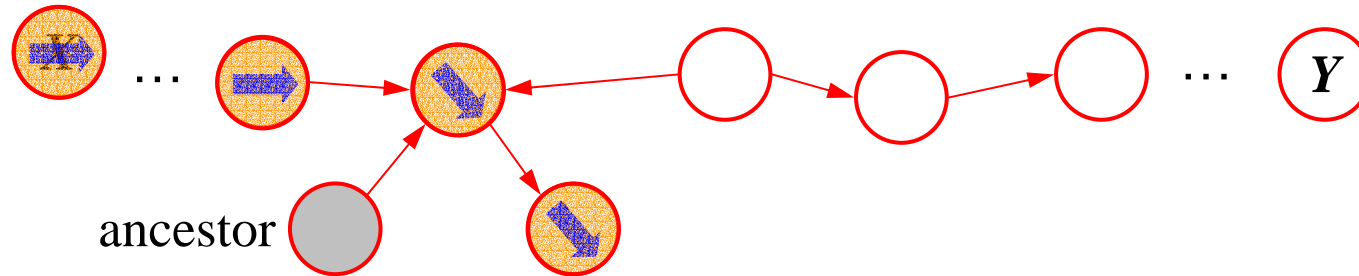
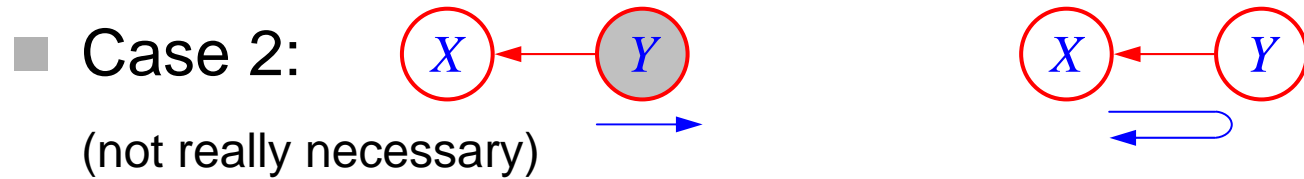
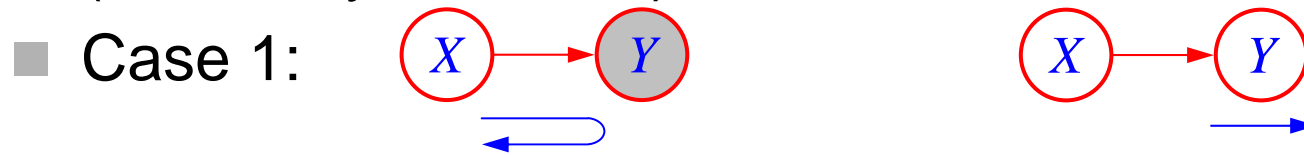
Bayes (Bouncing) Ball Rules

- A is d-separated from B given C if we cannot send a ball from any node in A to any node in B according to the rules below, where shaded nodes are in C



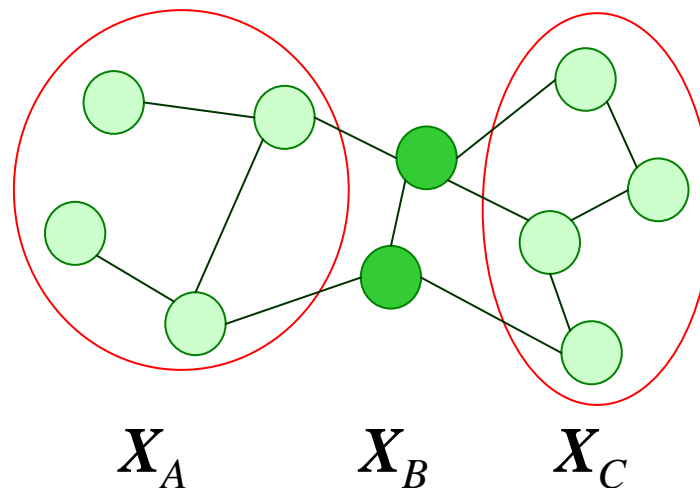
Bayes Ball Rules (cont.)

(Boundary condition)



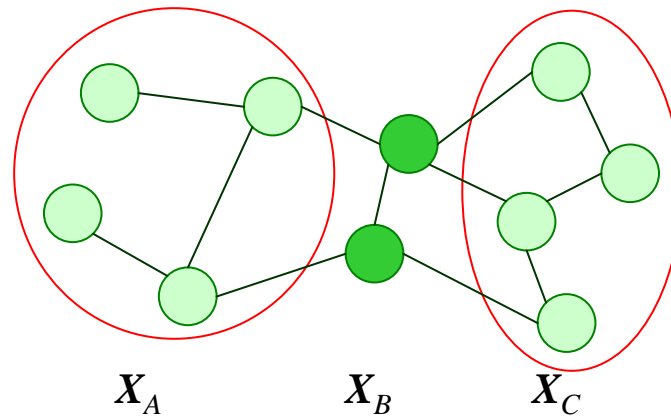
Undirected Graphical Models

- A.K.A **Markov Random Fields, Markov Networks**
- 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-neighbours given its neighbours, i.e. $X_A \perp X_C \mid X_B$ if every path between X_A and X_C goes through X_B**
- **Can model symmetric interactions that directed models cannot!**



Simple Graph Separation

- In undirected models, simple graph separation (as opposed to d-separation) tells us about conditional independencies
- $X_A \perp X_C \mid X_B$ if every path between X_A and X_C is blocked by some node in X_B



- “Markov Ball” algorithm:
remove X_B and see if there is any path from X_A to X_C

Conditional Parameterization?

- In directed models, we started with $p(X) = \prod_i p(x_i | \mathbf{x}_{\pi_i})$ and we derived the d-separation semantics from that.
- Undirected models: have the semantics, need parametrization.
- What about this “conditional parameterization”?

$$p(X) = \prod_i p(x_i | \mathbf{x}_{\text{neighbors}(i)})$$

- Good: product of local functions.
Good: each one has a simple conditional interpretation.
Bad: local functions cannot be arbitrary, but must agree properly in order to define a valid distribution.

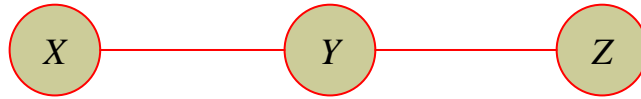
Marginal Parameterization?

- OK, what about this “marginal parameterization”?

$$p(X) = \prod_i p(x_i, \mathbf{x}_{\text{neighbors}(i)})$$

- Good: product of local functions.
Good: each one has a simple marginal interpretation.
Bad: only very few pathological marginals on overlapping nodes can be multiplied to give a valid joint.

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:

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

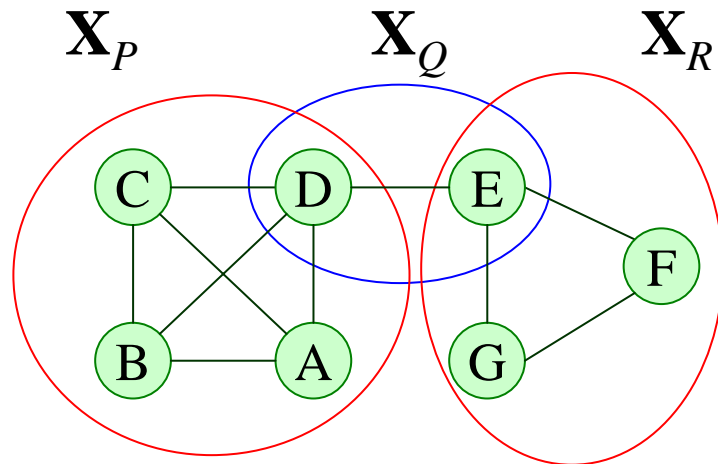
$$P(X, Y, Z) = P(X \mid Y) P(Z, Y) = \psi_{xy}(X, Y) \psi_{yz}(Y, Z)$$

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.

When Causality not Playing a Role!



ψ : potential functions

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

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

An example of Undirected Model

- 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:

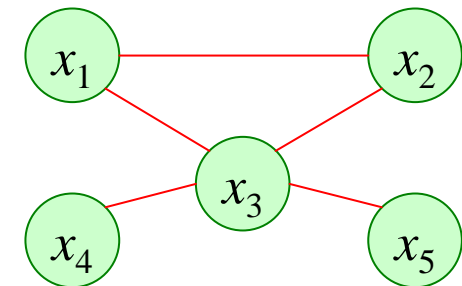
$$P(\mathbf{X}) = \frac{1}{Z} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c) \quad Z = \sum_{\mathbf{X}} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

- The product of functions that don't need to agree with each other.
- Still factors in the way that the graph semantics demand.
- Without loss of generality we can restrict ourselves to maximal cliques. (Why?)

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.

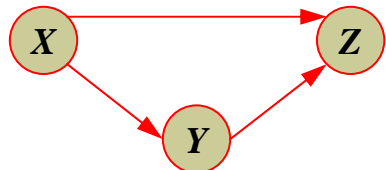
Directed vs. Undirected Models

- Directed models
 - using **conditional prob.** for each local substructure
 - called Bayesian network
 - **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 field or Markov network
 - **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

A Directed Model of 3 r.v.

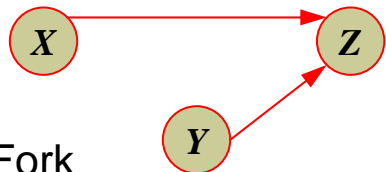
- A general dist.

$$P(X, Y, Z) = P(X) P(Y | X) P(Z | X, Y)$$



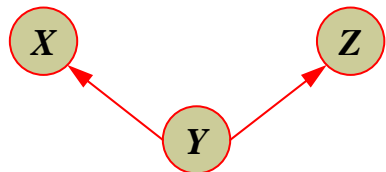
- Explaining away

$$P(X, Y, Z) = P(X) P(Y) P(Z | X, Y)$$



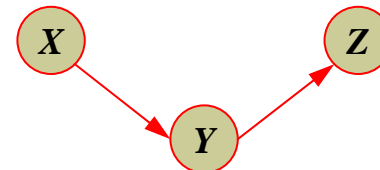
- Fork

$$P(X, Y, Z) = P(Y) P(X | Y) P(Z | Y)$$



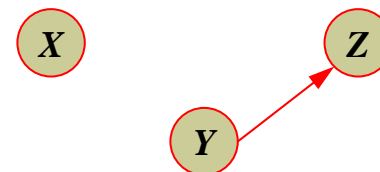
- Chain

$$P(X, Y, Z) = P(X) P(Y | X) P(Z | Y)$$



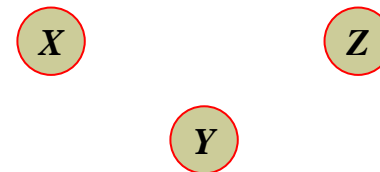
- One link

$$P(X, Y, Z) = P(X) P(Y) P(Z | Y)$$



- All independence

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

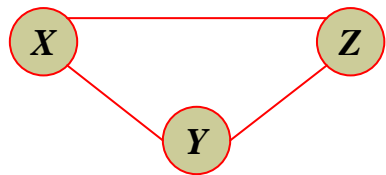


the same group

An Undirected Model of 3 r.v.

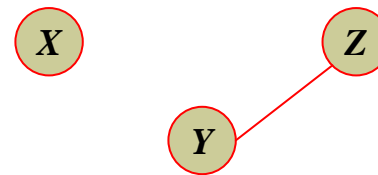
- A general dist.

$$\begin{aligned} P(X, Y, Z) &= P(X) P(Y | X) P(Z | X, Y) \\ &= \psi(X, Y, Z) \end{aligned}$$



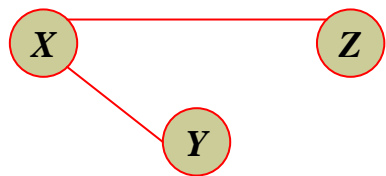
- One link (3 cases)

$$\begin{aligned} P(X, Y, Z) &= P(X) P(Y) P(Z | Y) \\ &= \psi(X) \psi(Y, Z) \end{aligned}$$



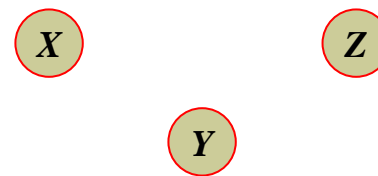
- Two links (3 cases)

$$\begin{aligned} P(X, Y, Z) &= P(Y, Z | X) P(X) \\ &= P(X) P(Y | X) P(Z | X) \\ &= \psi(X, Y) \psi(X, Z) \end{aligned}$$



- All independence

$$\begin{aligned} P(X, Y, Z) &= P(X) P(Y) P(Z) \\ &= \psi(X) \psi(Y) \psi(Z) \end{aligned}$$

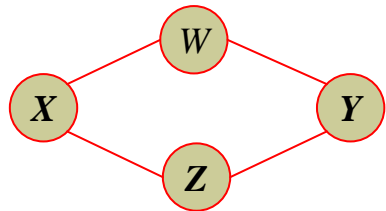


Between Directed and Undirected Models

- Directed can't do it!

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

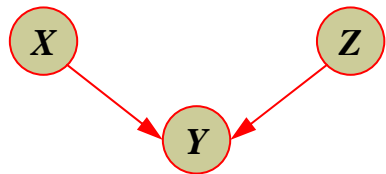
$$W \perp Z / \{ X, Y \}$$



- Undirected can't do it!

$$X \perp Z$$

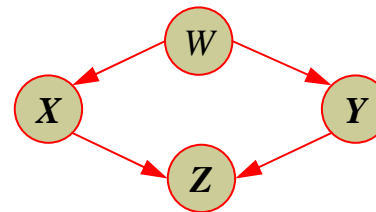
$$X \not\perp Z / Y$$



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

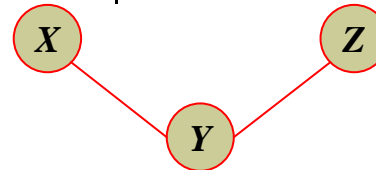
$$X \perp Y / W$$

$$X \not\perp Y / \{ W, Z \}$$

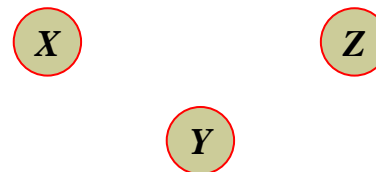


- Undirected can't do it!

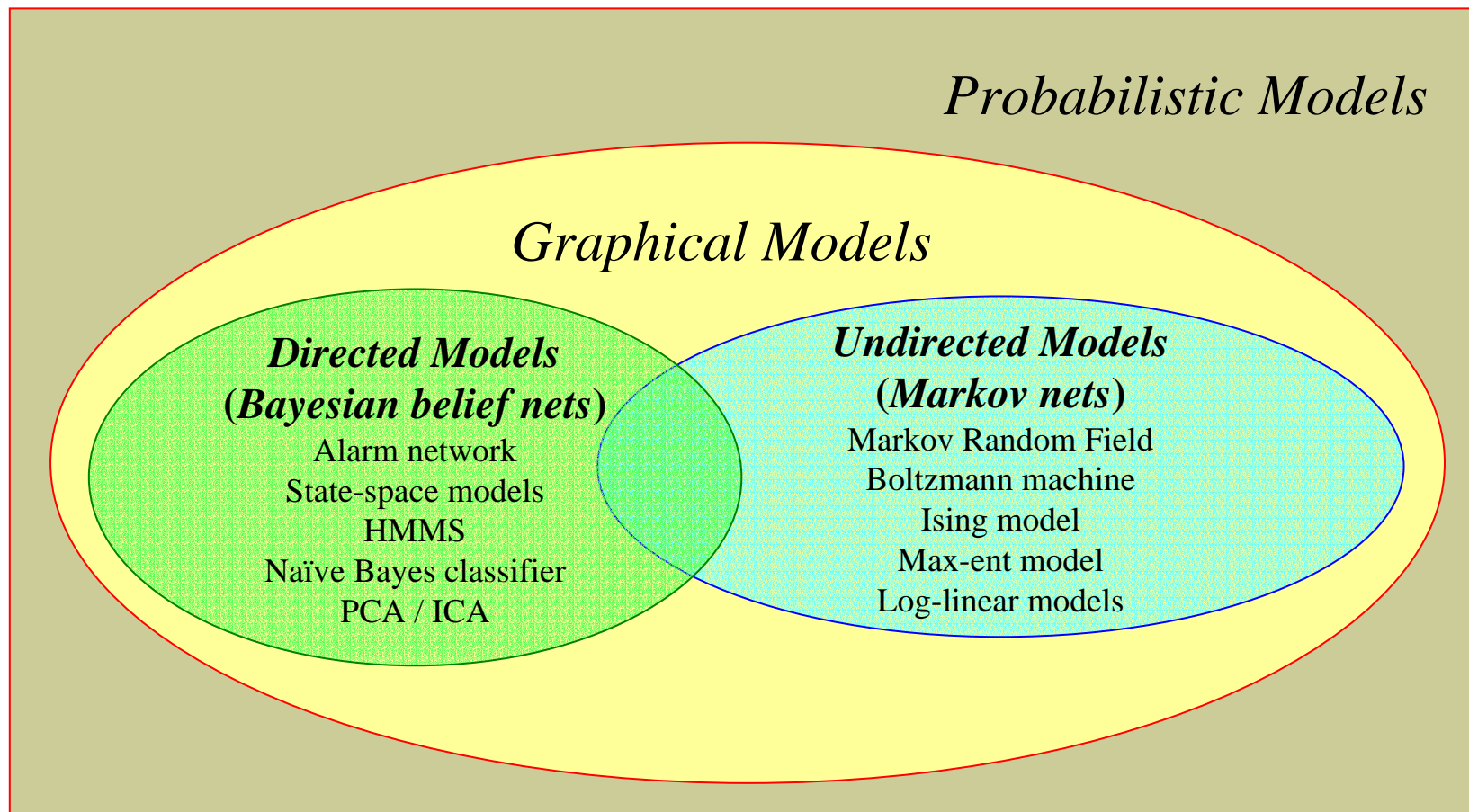
$$X \perp Z | Y$$



$$X \perp Z$$



Probabilistic Graphical Models





Part II



Probabilistic Inference

Outline

- Introduction to Bayesian networks
- Bayesian networks: definition, d-separation, equivalence of networks
- Examples: causal graphs, explaining away, Markov chains, Naïve Bayes, etc
- Undirected models
- **Probabilistic inference**
 - node elimination
 - junction tree
- Building the networks

Probabilistic Inference

- Task
 - infer value of some X_i
 - given values of (some) other X_j in the network
- Facts
 - all other attributes known: easy
 - general case: **NP-hard** (Cooper, 1990)
(proving 3-SAT easier than probabilistic inference!)
 - approximated inference: still NP-hard (Dagum & Luby, 1993)
 - Monte-Carlo (Pradham & Dagum, 1996)

Probabilistic Inference II

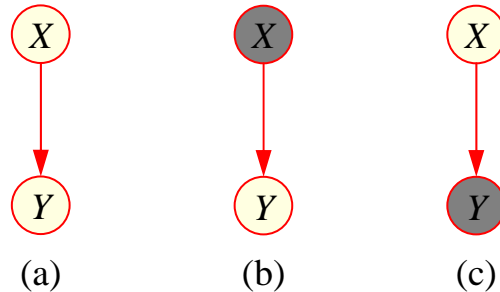
- Partition the random variables in a domain \mathbf{X} into three disjoint subsets X_E, X_F, X_R . The general *probabilistic inference* problem is to compute the posterior $p(X_F | X_E)$ over the *query nodes* X_F .
- This involves conditioning on evidence nodes X_E and *integrating (summing) out marginal 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

Probabilistic Inference III

- If the joint is a known *continuous* function this can sometimes be done analytically. (e.g. Gaussian: eliminate rows/cols corresponding to X_R ; apply conditioning formulas for $p(X_F | X_E)$)
- But what if the joint distribution over \mathbf{X} is represented by a *directed or undirected graphical model*?

Recall Bayes Rule

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



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

$$(b) p(y | x)$$

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

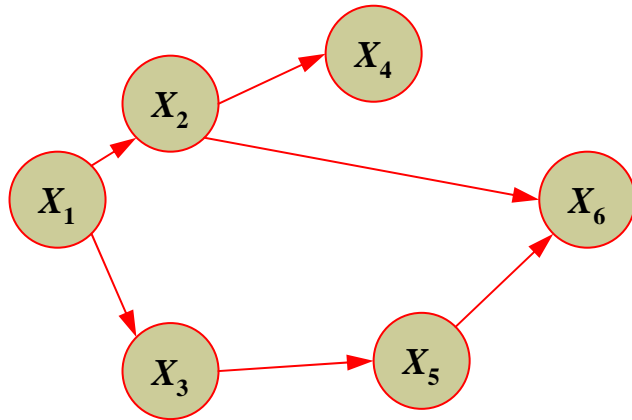
This is called “reversing the arrow”

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

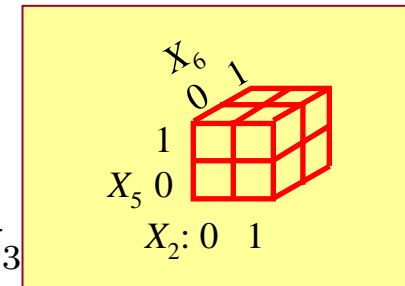
$$p(x_F | x_E) = \frac{\sum_{x_R} p(x_E, x_F, x_R)}{\sum_{x_F, x_R} p(x_E, x_F, x_R)}$$

- **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

Take Advantage of Distributed Law



■ Compute $P(X_{[1..5]})$?



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

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

The Generalized Distributed Law

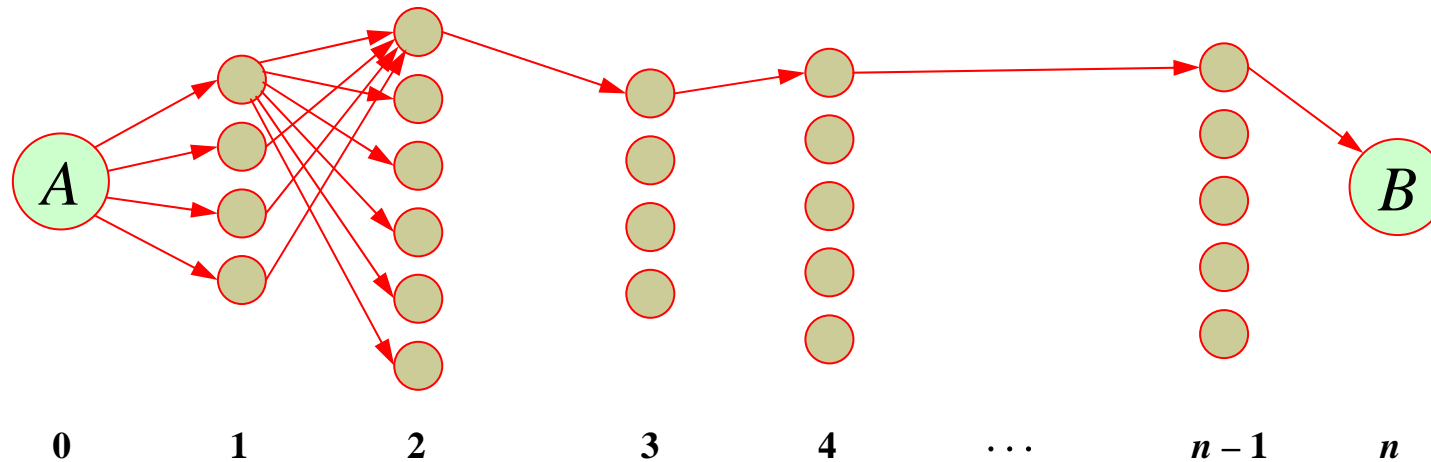
$$a b + a c = a (b + c)$$

left: 2 “×” 1 “+”, right: 1 “×”, 1 “+”



一場遊戲一場夢?!

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
- (One of the) solutions is 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:

Init ($i = 0$): $v_0(0) = 1, v_k(0) = 0$ for $k > 0$

Recursion ($i = 1..n$): $v_l(i) = \max_k (v_k(i-1)a_{kl})$
 $ptr_i(l) = \arg \max_k (v_k(i-1)a_{kl})$

Termination: $P(Y, \pi^*) = \max_k (v_k(n)a_{k0})$
 $\pi_n^* = \arg \max_k (v_k(n)a_{k0})$

Traceback ($i = n..1$): $\pi_{i-1}^* = ptr_i(\pi_i^*)$

(Note : end state is assumed)

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.
$$(a \cdot b) + (a \cdot c) = a \cdot (b + c), \forall a, b, c \text{ from } K$$
- A semi-ring is a commutative ring without the additive inverse

Example: max-product

- $K = \mathbf{R}^+ = [0, +\infty)$, “+”: 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.

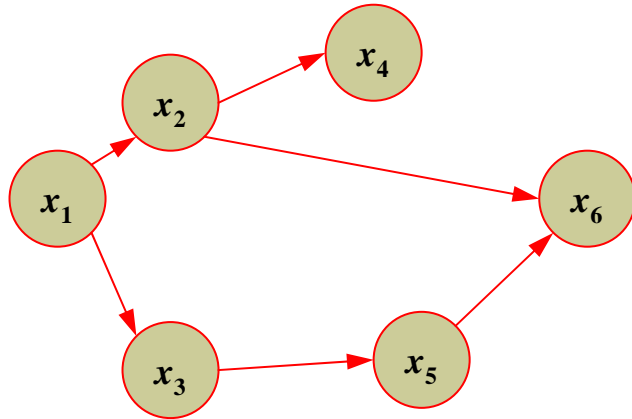
$\max\{a \cdot b, a \cdot c\} = a \cdot \max(b, c), \forall a, b, c$ from K

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

Viterbi vs. GDL

- Viterbi is just a form of **GDL**, by choosing an appropriate **semi-ring**
- In fact, many **dynamic programming** processes can be interpreted by GDL
- Other examples: Baum-Welch algorithm, FFT(fast Fourier transform) on any finite Abelian group, Gallager-Tanner-Wiberg decoding algorithm, BCJR algorithm, Pearl's "belief propagation", Shafer-Shenoy probability propagation algorithm, turbo decoding algorithm, etc.

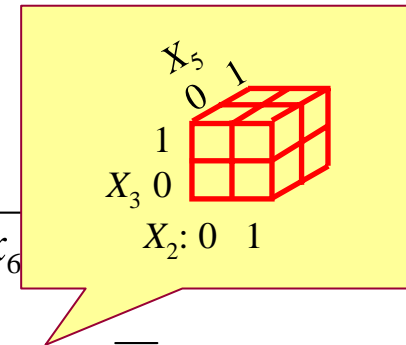
Example



■ Compute $p(x_1 | \bar{x}_6) = p(x_1, \bar{x}_6) / p(\bar{x}_6)$?

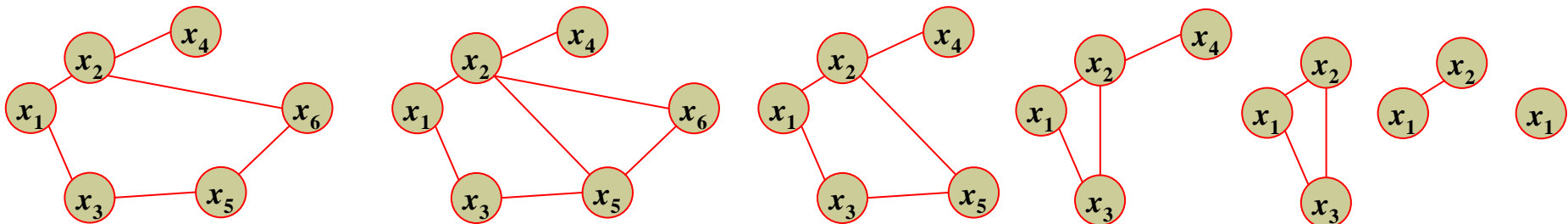
■ $p(\bar{x}_6) = \sum_{x_1'} p(x_1', \bar{x}_6)$

$$\begin{aligned}
 p(x_1, \bar{x}_6) &= \sum_{x_2, x_3, x_4, x_5} p(x_1, x_2, x_3, x_4, x_5, \bar{x}_6) \\
 &= \sum_{x_2, x_3, x_4, x_5} p(x_1) p(x_2 | x_1) p(x_3 | x_1) p(x_4 | x_2) p(x_5 | x_3) p(\bar{x}_6 | x_2, x_5) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_5} p(x_5 | x_3) p(\bar{x}_6 | x_2, x_5) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \Phi_5(x_2, x_3) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \Phi_4(x_2) \Phi_3(x_1, x_2) = p(x_1) \Phi_2(x_1)
 \end{aligned}$$



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

ELIMINATE(G)

place all $P(X_i | \mathbf{X}_{\pi_i})$ and $\delta(X_i, \overline{X_i})$ on the active list

choose an ordering I such that F appears last

for each X_i in I

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

define a new potential as the sum (with respect to X_i) of the product of these potentials

place the new potential on the active list

end

return the product of the remaining potentials

Algorithm Details

- At each step we are trying to **remove the current variable in the elimination ordering from the distribution**
For marginal nodes the sums them out, for evidence nodes this conditions on their observed values using the **evidential functions**
- Each step performs a sum over a product of potential functions
- The algorithm terminates when we reach the query node, which **always appears last in the ordering**
- We renormalize what we have left to get the final result $P(\mathbf{X}_F | \mathbf{X}_E)$
- For undirected models, everything is the same except that initialization phase uses the clique potentials instead of the parent-conditionals

Marginalization without Evidence

- Marginalization of joint distributions represented by graphical models is a special case of probabilistic inference
- To compute the marginal $P(X_i)$ of a single node, we set it to be the query node and set the evidence set to be empty
- In directed models, we can ignore **all nodes downstream from the query node, and marginalize only the part of the graph before it**
- **If the node has no parents, we can read off its marginal directly**
- In undirected models, we need to do the full computation: compute $P(X_i) / Z$ using elimination and then normalize in the last step of elimination to get Z . (We can reuse Z later if we want to save work)

Efficiency Trick in Directed Elimination

- In directed models, we often know that a certain sum must evaluate to unity, since it is a conditional probability.
- For example, consider the term $\Phi_4(\mathbf{x}_2)$ in our six node example:

$$\Phi_4(\mathbf{x}_2) = \sum_{x_4} p(\mathbf{x}_4 | \mathbf{x}_2) \equiv 1$$

- We can't use this trick in undirected models, because there are no guarantees about what clique potentials sum to.

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 :

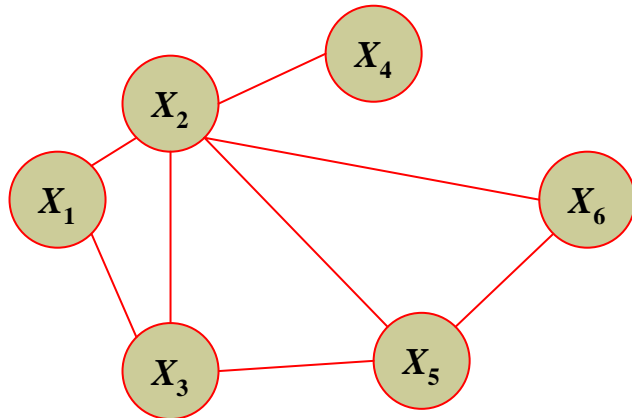
connect all the neighbors of x_i

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.)

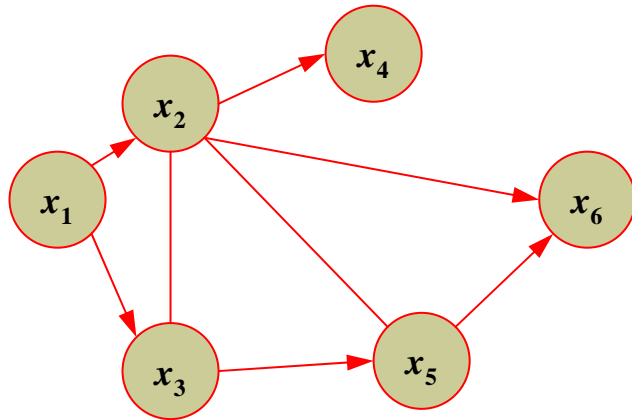
Example: Node/Variable Elimination



- 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
- Continue to do summations, ...

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

Added Edges = Triangulation

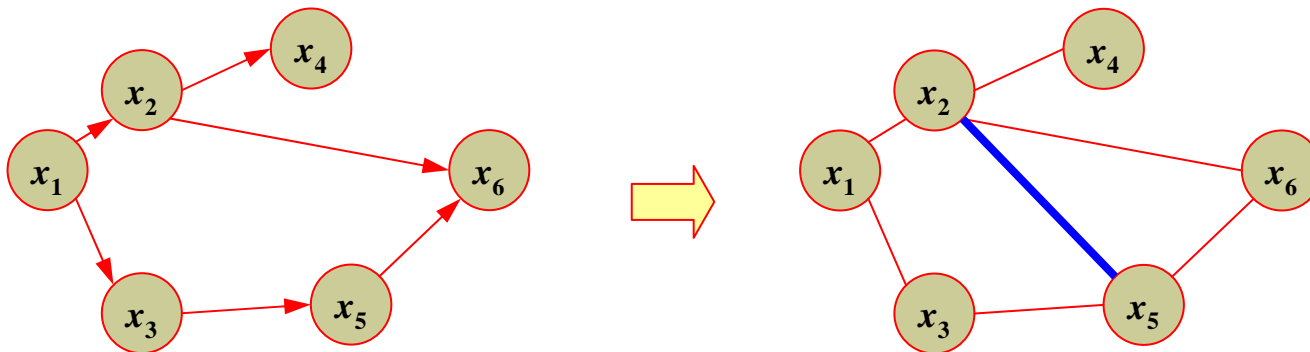


- It is easy to check if a graph is triangulated in linear time
- It is easy to triangulate a non-triangulated graph
- **But it is very hard to do so in a way that induces small clique sizes**

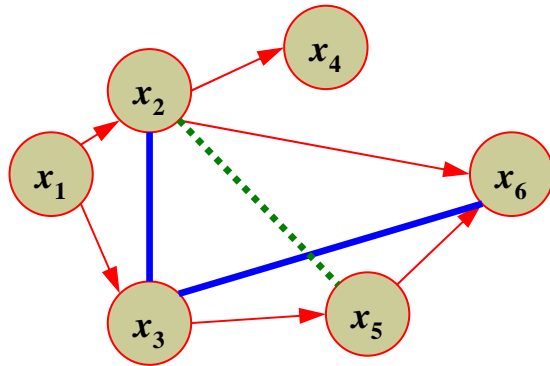
$$\begin{aligned}
 p(x_1, \bar{x}_6) &= \sum_{x_2, x_3, x_4, x_5} p(x_1, x_2, x_3, x_4, x_5, \bar{x}_6) \\
 &= \sum_{x_2, x_3, x_4, x_5} p(x_1) p(x_2 | x_1) p(x_3 | x_1) p(x_4 | x_2) p(x_5 | x_3) p(\bar{x}_6 | x_2, x_5) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_5} p(x_5 | x_3) p(\bar{x}_6 | x_2, x_5) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \Phi_5(x_2, x_3) \\
 &= p(x_1) \sum_{x_2} p(x_2 | x_1) \Phi_4(x_2) \Phi_3(x_1, x_2) = p(x_1) \Phi_2(x_1)
 \end{aligned}$$

Moralization

- For directed graphs, the parents may not be explicitly connected, but they are involved in the same potential function $P(x_i | x_{\pi_i})$
- 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.



Markov Blanket



- Elimination order 1:

$$x_2 \leftarrow x_4 \leftarrow x_3 \leftarrow x_5 \leftarrow x_6$$

- Elimination order 2:

$$x_2 \leftarrow x_4 \leftarrow x_3 \leftarrow x_5 \leftarrow x_5$$

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 ($= X_2$). This three is called the Markov blanket (of X_5)

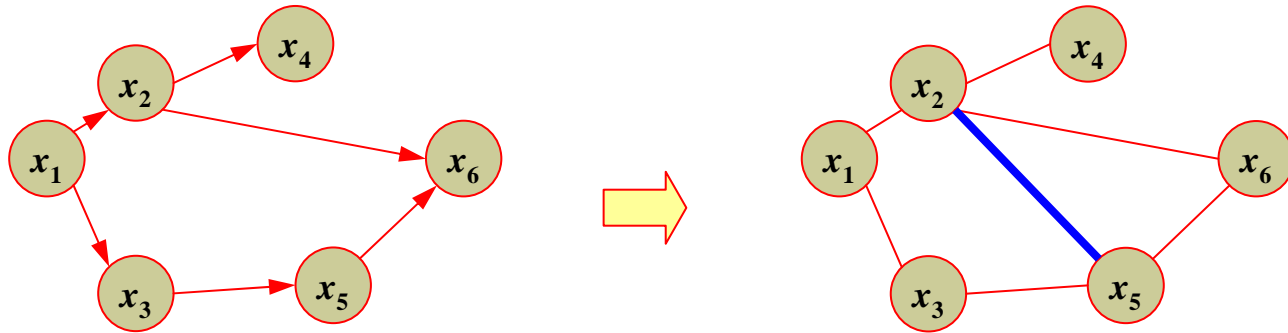
Order 1:

$$\begin{aligned} p(x_1) &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_5} p(x_5 | x_3) \sum_{x_6} p(x_6 | x_2, x_5) \\ &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_5} p(x_5 | x_3) \Phi(x_2, x_5) \end{aligned}$$

Order 2:

$$\begin{aligned} p(x_1) &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_6} \sum_{x_5} p(x_5 | x_3) p(x_6 | x_2, x_5) \\ &= p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_3} p(x_3 | x_1) \sum_{x_6} \Phi(x_2, x_3, x_6) \end{aligned}$$

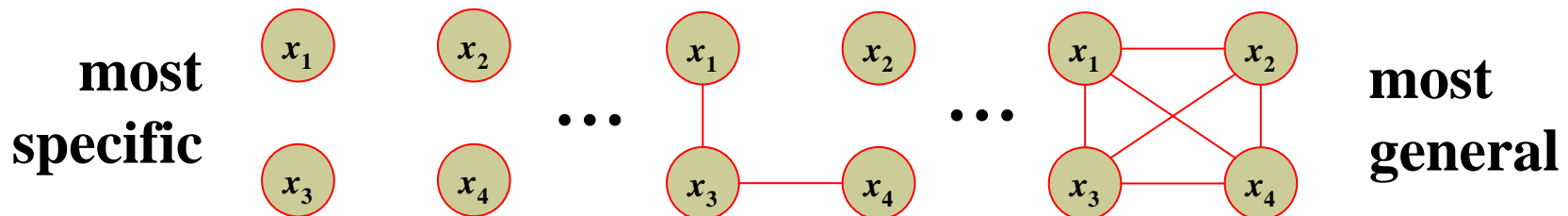
Moral Graph



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

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

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

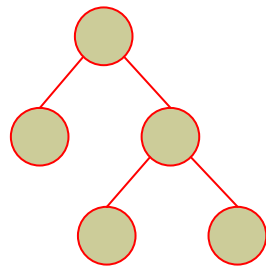


Junction Tree

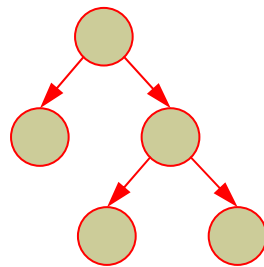
- Moralization
- Triangulation
- Construct Junction Tree
- Propagate Probabilities

Tree-Structured Graphical Models

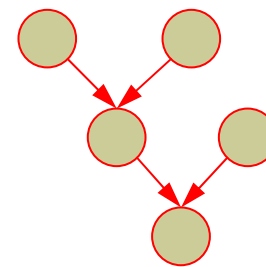
- For now, we will focus on tree-structured graphical models.
- Trees are an important class; they incorporate all chains (e.g. HMMs) as well.
- Exact inference on trees is the basis for the [junction tree algorithm](#) which solves the general exact inference problem for directed acyclic graphs and 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)



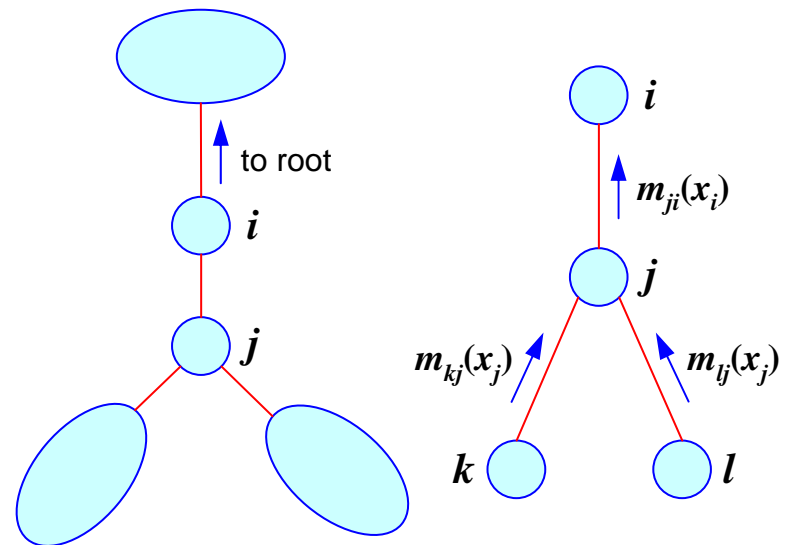
(c)

Elimination on Trees

- Recall basic structure of Eliminate:
 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.
 2. There is a natural elimination ordering with query node as root. (Any *depth first search* order.)
 3. All subtrees with no evidence nodes can be ignored (since they will leave a potential of unity once they are eliminated).

Elimination on Trees

- Now consider eliminating node j which is followed by i in the order.
- Which nodes appear in the potential created after summing over j ?
 - nothing in the subtree below j (already eliminated)
 - nothing from other subtrees, since the graph is a tree
 - only i , from ψ_{ij} which relates i and j
- Call the factor that is created $m_{ji}(x_i)$, and think of it as a message that j passes to i when j is eliminated.
- This message is created by summing over j the product of all earlier messages $m_{kj}(x_j)$ sent to j as well as $\psi_j^E(x_j)$ (if j is an evidence node).



Eliminate = Message Passing

- On a tree, ELIMINATE can be thought of as passing messages up to the query node at the root from the other nodes at the leaves or interior. Since we ignore subtrees with no evidence, observed (evidence) nodes are always at the leaves.

- The message $m_{ji}(x_i)$ is created when we sum over x_j

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

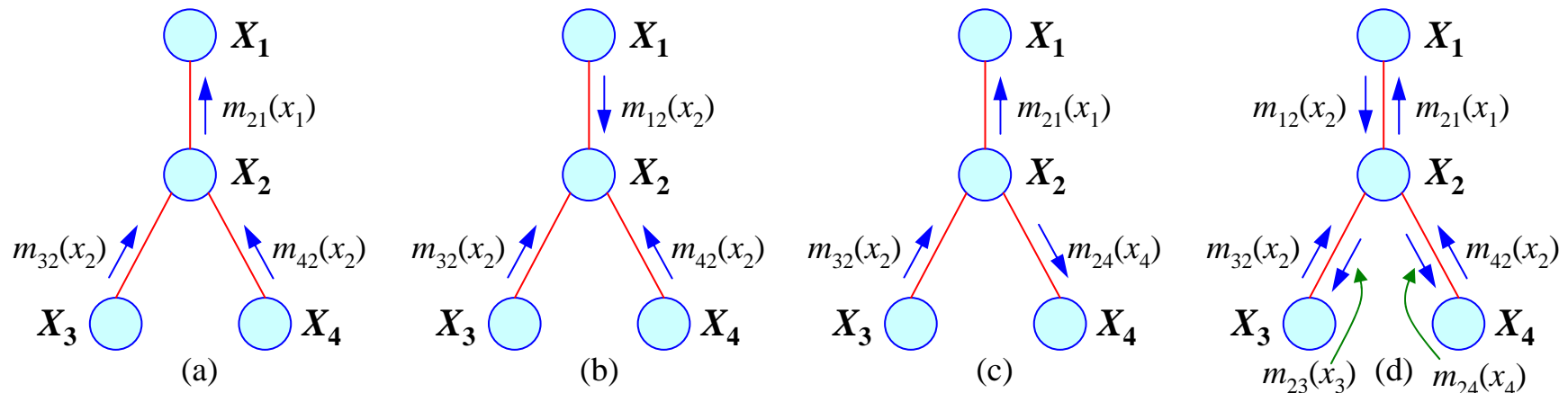
- At the final node x_f , we obtain the answer:

$$p(x_f | \bar{\mathbf{x}}_E) \propto \psi^E(x_f) \prod_{k \in c(f)} m_{kf}(x_f)$$

- If j is an evidence node, $\psi^E(x_j) = \delta(x_j, \bar{x}_j)$ else $\psi^E(x_j) = 1$.
- If j is a leaf node in the ordering, $c(j)$ is empty, otherwise $c(j)$ are the children of j in the ordering.

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.

Computing All Possible Messages

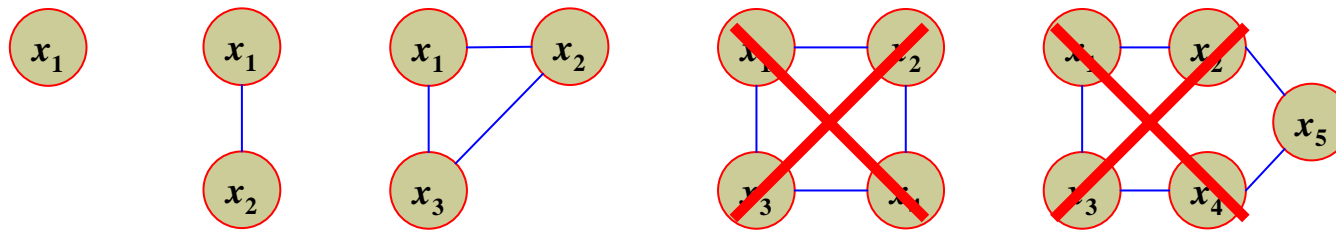
- How can we compute all possible messages efficiently?
- Idea: respect the following Message-Passing-Protocol: A node can send a message to a neighbor only when it has received messages from all its other neighbors.
- Protocol is realizable: designate one node (arbitrarily) as the root. Collect messages inward to root then distribute back out to leaves.
- Once we have the messages, we can compute marginals using:

$$p(x_i | \bar{\mathbf{x}}_E) \propto \psi^E(x_i) \prod_{k \in c(i)} m_{ki}(x_i)$$

- Remember that the directed tree on which we pass messages might not be same directed tree we started with.
- We can also consider “synchronous” or “asynchronous” message passing nodes that respect the protocol but don’t use the Collect-Distribute schedule above. (Must prove this terminates.)

Triangulation

- Triangulation: connect nodes in moral graph such that no cycle of 4 or more nodes remains in graph



- So, add links, but many possible choices...
- HINT: Try to keep largest clique size small (makes junction tree algorithm more efficient)
- Sub-optimal triangulations of moral graph are polynomial
- Triangulation that minimizes largest clique size is NP
- But, OK to use a suboptimal triangulation (slower JTA)



Part III



Learning Belief Networks

Outline

- Introduction to Bayesian networks
- Bayesian networks: definition, d-separation, equivalence of networks
- Examples: causal graphs, explaining away, Markov chains, Naïve Bayes, etc
- Undirected models
- Probabilistic inference
 - node elimination
 - junction tree
- **Building the networks**

Building Bayesian Network Models

- Tasks of building models
 - Catching the model structure
 - Determining the conditional probabilities
- 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
- 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

Maximum Likelihood

- For IID data:

$$P(D|\theta) = \prod_m P(\mathbf{X}^m|\theta)$$

$$\ell(\theta; D) = \sum_m \log P(\mathbf{X}^m|\theta)$$

- Idea of maximum likelihood estimation (MLE): pick the setting of parameters most likely to have generated the data we saw:

$$\theta_{ML}^* = \operatorname{argmax}_{\theta} \ell(\theta; D)$$

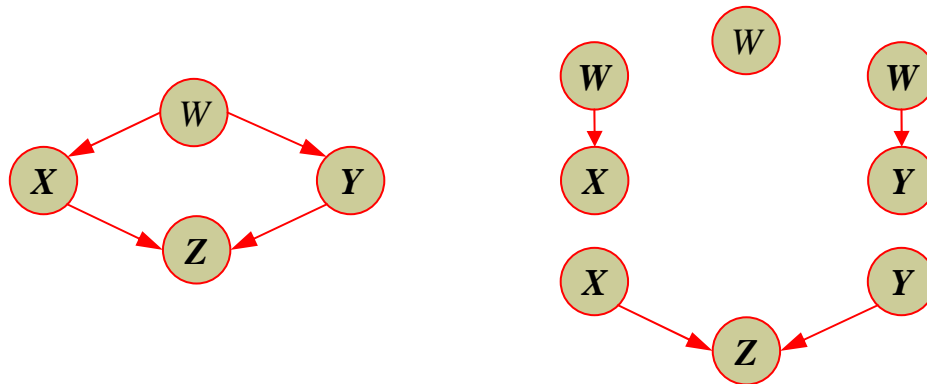
- Very commonly used in statistics.
Often leads to “intuitive”, “appealing”, or “natural” estimators.
- For a start, the IID assumption makes the log likelihood into a sum, so its derivative can be easily taken term by term.

MLE for Directed GMs

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

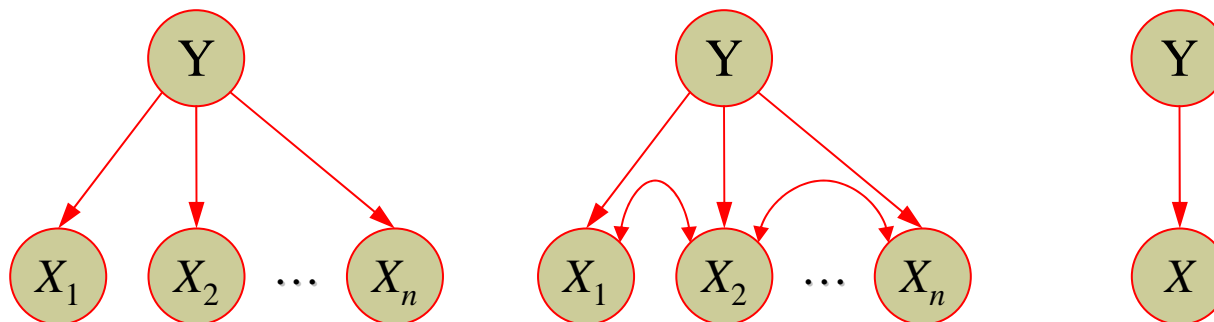
$$\log P(D|\theta) = \log \prod_m \prod_i P(X_i^m | \mathbf{X}_{\pi_i}, \theta_i) = \sum_m \sum_i \log P(X_i^m | \mathbf{X}_{\pi_i}, \theta_i)$$

- 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}_{π_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.



Three Key Regularization Ideas

- To avoid overfitting, we can put priors on the parameters of the class and class conditional feature distributions
- We can also tie some parameters together so that fewer of them are estimated using more data
- Finally, we can make factorization or independence assumptions about the distributions. In particular, for the class conditional distributions we can assume the features are fully dependent, partly dependent, or independent (!).



Discrete (Multinomial) Naive Bayes

- Discrete features x_i , assumed independent given the class label y

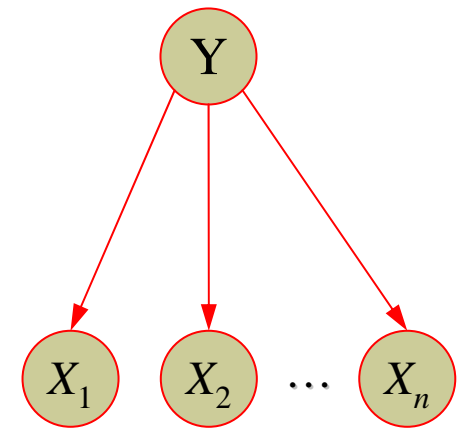
$$P(X_i = j | y = k) = \eta_{ijk}$$
$$P(\mathbf{X} | y = k, \eta) = \prod_i \prod_j \eta_{ijk}^{[X_i=j]}$$

- Classification rule:

$$P(y = k | \mathbf{X}, \eta) = \frac{\pi_k \prod_i \prod_j \eta_{ijk}^{[x_i=j]}}{\sum_q \pi_q \prod_i \prod_j \eta_{ijq}^{[x_i=j]}}$$
$$= \frac{e^{\beta_k^T \mathbf{x}}}{\sum_q e^{\beta_q^T \mathbf{x}}}$$

$$\beta_k = \log[\eta_{11k} \cdots \eta_{1jk} \cdots \eta_{ijk} \cdots \log \pi_k]$$

$$\mathbf{x} = [x_1 = 1; x_1 = 2; \dots; x_i = j; \dots; 1]$$



Fitting Discrete Naive Bayes

- ML parameters are class-conditional frequency counts:

$$\eta_{ijk}^* = \frac{\sum_m [x_i^m = j][y^m = k]}{\sum_m [y^m = k]}$$

- How do we know? Write down the likelihood:

$$\ell(\theta; D) = \sum_m \log P(y^m | \pi) + \sum_{mi} \log P(x_i^m | y^m, \eta)$$

and optimize it by setting its derivative to zero (careful! enforce normalization with Lagrange multipliers):

$$\ell(\eta; D) = \sum_m \sum_{ijk} [x_i^m = j][y^m = k] \log \eta_{ijk} + \sum_{ik} \lambda_{ik} (1 - \sum_j \eta_{ijk})$$

$$\frac{\partial \ell}{\partial \eta_{ijk}} = \frac{\sum_m [x_i^m = j][y^m = k]}{\eta_{ijk}} - \lambda_{ik}$$

$$\frac{\partial \ell}{\partial \eta_{ijk}} = 0 \Rightarrow \lambda_{ik} = \sum_m [y^m = k] \Rightarrow \eta_{ijk}^* = \text{above}$$

Learning Markov Models

- The ML parameter estimates for a simple Markov model are easy:

$$P(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T) = P(\mathbf{y}_1, \dots, \mathbf{y}_k) \prod_{t=k+1}^T P(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots, \mathbf{y}_{t-k})$$

$$\log P(\{\mathbf{y}\}) = \log P(\mathbf{y}_1, \dots, \mathbf{y}_k) + \sum_{t=k+1}^T \log P(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots, \mathbf{y}_{t-k})$$

- Each window of $k + 1$ outputs is a training case for the model

$$P(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots, \mathbf{y}_{t-k})$$

- Example: for discrete outputs (symbols) and a 2nd-order Markov model we can use the multinomial model:

$$P(y_t = m | y_{t-1} = a, y_{t-2} = b) = \alpha_{mab}$$

- The maximum likelihood values for α are:

$$\alpha_{mab}^* = \frac{|t \text{ s.t. } y_t = m, y_{t-1} = a, y_{t-2} = b|}{|t \text{ s.t. } y_{t-1} = a, y_{t-2} = b|}$$

Summary

- Introduction to Bayesian networks
- Bayesian networks: definition, d-separation, equivalence of networks
- Examples: causal graphs, explaining away, Markov chains, Naïve Bayes, etc
- Undirected models
- Probabilistic inference
 - node elimination
 - junction tree
- Building the networks

What Else ...

- Bayesian networks with continuous r.v.'s
 - PCA, ICA, other dimension reduction methods
 - will be discussed in coming lectures!
- Models with unobservable r.v.'s
 - ⇒ learning by Expectation-Maximization
 - ⇒ will also be delivered!