Probabilistic Graphical Models

Probabilistic Modelling

1. **Represent** the world as a collection of random variables $X_1, \ldots, X_n$ with joint distribution $p(X_1, \ldots, X_n)$
2. **Learn** the distribution from data
3. Perform "**inference**" (compute conditional distributions $p(X_i \mid X_1 = x_1, \ldots, X_m = x_m)$)

Motivating Example (Credit: Chris Bishop)

A murder mystery

A fiendish murder has been committed

Whodunit?

There are two suspects:

- the **Butler**
- the **Cook**

There are three possible murder weapons:

- a butcher's **Knife**
- a **Pistol**
- a fireplace **Poker**

Prior distribution

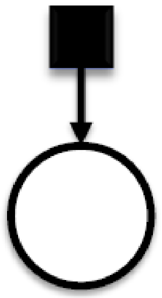
Butler has served family well for many years

Cook hired recently, rumours of dodgy history

P(Culprit = **Butler**) = 20%

P(Culprit = **Cook**) = 80%

Probabilities add to 100%



This is called a *factor graph* (we'll see why later)

Conditional distribution

Butler is ex-army, keeps a gun in a locked drawer

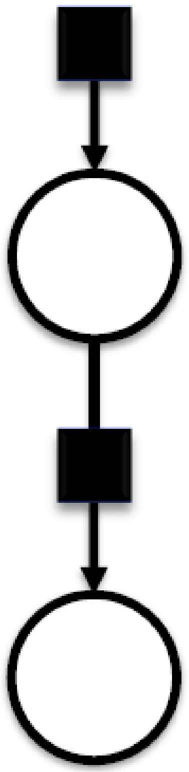
Cook has access to lots of knives

Butler is older and getting frail

	Pistol	Knife	Poker	
Cook	5%	65%	30%	**= 100%**
Butler	80%	10%	10%	**= 100%**

P(Weapon | Culprit)

Factor graph



Prior distribution

$P(\text{Culprit})$

Culprit = {**Butler**, **Cook**}

Conditional distribution

$P(\text{Weapon} \mid \text{Culprit})$

Weapon = {**Pistol**, **Knife**, **Poker**}

Joint distribution

What is the probability that the **Cook** committed the murder using the **Pistol**?

P(Culprit = **Cook**) = 80%

P(Weapon = **Pistol** | Culprit = **Cook**) = 5%

P(Weapon = **Pistol** , Culprit = **Cook**) = 80% x 5% = 4%

Likewise for the other five combinations of Culprit and Weapon

Joint distribution

	Pistol	Knife	Poker
Cook	4%	52%	24%
Butler	16%	2%	2%

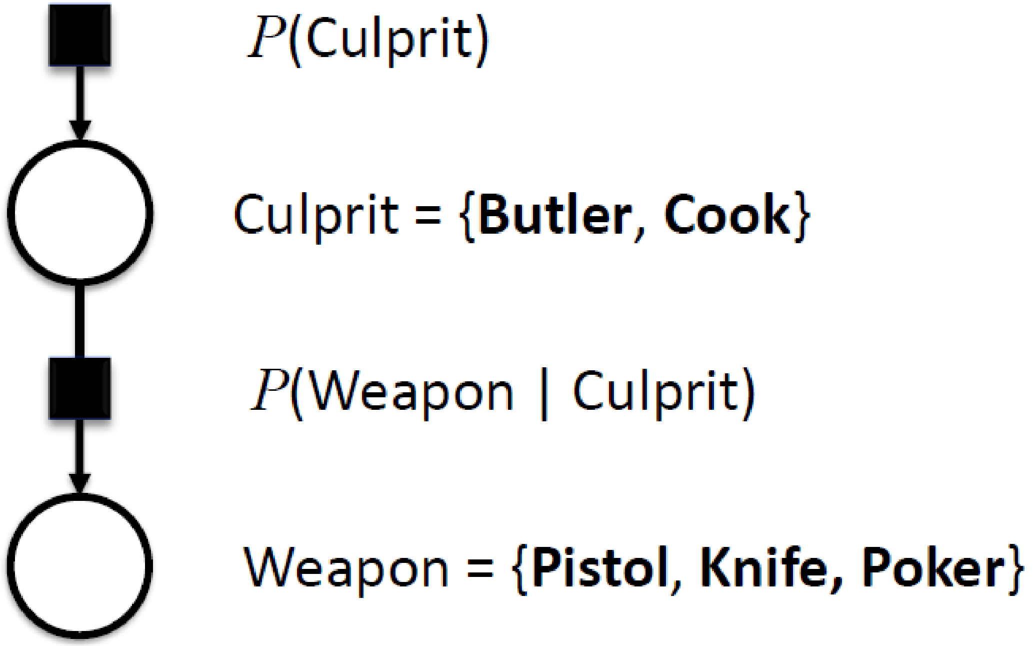
= 100%

P(Weapon, Culprit) = P(Weapon | Culprit) P(Culprit)

$$\boxed{P(x,y) = P(y|x)P(x)}$$

Product rule

Factor graphs


P(Culprit)
Culprit = {**Butler**, **Cook**}
P(Weapon | Culprit)
Weapon = {**Pistol**, **Knife**, **Poker**}

P(Weapon, Culprit) = P(Weapon | Culprit) P(Culprit)

Generative viewpoint

Murderer	Weapon
Cook	Knife
Butler	Knife
Cook	Pistol
Cook	Poker
Cook	Knife
Butler	Pistol
Cook	Poker
Cook	Knife
Butler	Pistol
Cook	Knife
…	…

Marginal distribution of Culprit

	Pistol	Knife	Poker	
Cook	4%	52%	24%	**= 80%**
Butler	16%	2%	2%	**= 20%**

$$\boxed{P(x) = \sum_y P(x,y)}$$

Sum rule

Marginal distribution of Weapon

	Pistol	Knife	Poker
Cook	4%	52%	24%
Butler	16%	2%	2%
	= 20%	**= 54%**	**= 26%**

$$\boxed{P(x) = \sum_{y} P(x, y)}$$

Sum rule

Posterior distribution

We discover a **Pistol** at the scene of the crime

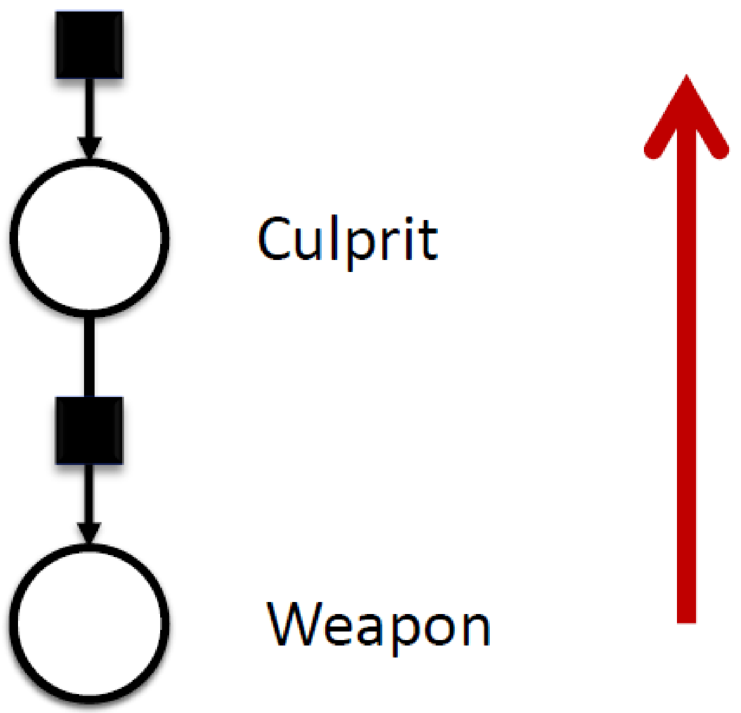
	Pistol	**Knife**	**Poker**	
Cook	4%	~~52%~~	~~24%~~	= 20%
Butler	16%	~~2%~~	~~2%~~	= 80%

This looks bad for the Butler!

Generative viewpoint

Murderer	Weapon
~~Cook~~	~~Knife~~
~~Butler~~	~~Knife~~
Cook	Pistol
~~Cook~~	~~Poker~~
~~Cook~~	~~Knife~~
Butler	Pistol
~~Cook~~	~~Poker~~
~~Cook~~	~~Knife~~
Butler	Pistol
~~Cook~~	~~Knife~~
…	…

Reasoning backwards



The Rules of Probability

Sum rule

$$P(x) = \sum_{y} P(x,y)$$

Product rule

$$P(x,y) = P(y|x)P(x)$$

Bayes' theorem

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Denominator

$$P(x) = \sum_{y} P(x|y)P(y)$$

Probabilistic Modelling in Practice

- Large number of variables with relations between them
- Complex analytic calculations during the inferencing procedure
- Problem of estimating high dimensional probability distributions
- How do we incorporate domain knowledge
- How do we update models

Probabilistic Graphical Models

Combine probability theory with graphs

- ✓ new insights into existing models
- ✓ framework for designing new models
- ✓ Graph-based algorithms for calculation

Advantages

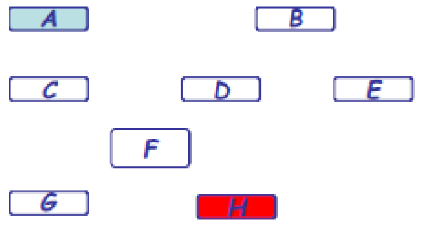
- Common semantics
- Compact representation
- Fast Computation
- Ease of Visualization
- Ease of incorporating domain knowledge

Tasks

- Representation: what is the joint probability dist. on multiple variables?

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

  - How many state configurations in total? --- 2^8
  - Are they all needed to be represented?
  - **Do we get any scientific/medical insight?**



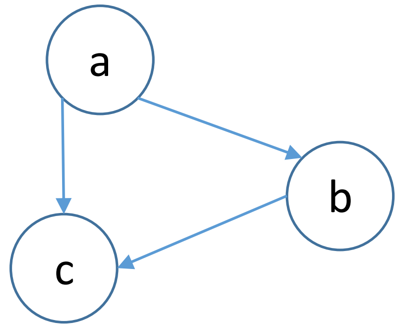
- Learning: where do we get all this probabilities?
  - Maximal-likelihood estimation? but how many data do we need?
  - Are there other est. principles?
  - Where do we put domain knowledge in terms of plausible relationships between variables, and plausible values of the probabilities?

- Inference: If not all variables are observable, how to compute the conditional distribution of latent variables given evidence?

Types of Graphical Models

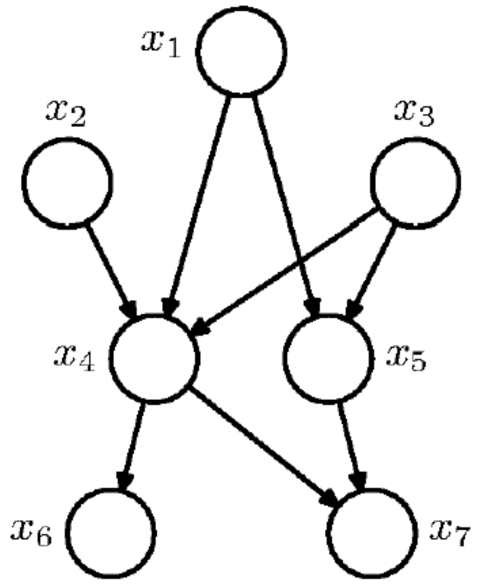
- Directed Graphical Models – Nodes: Variables, Edges: Causality
  - Bayesian Network
- Undirected Graphical Models – Nodes: Variables, Edges: Correlation
  - Markov Random Fields
- Factor Graphs – Combines above two in a general form
- Many others!

Directed Graphical Model



- Decomposition of the joint distribution:
- p(a, b, c) = p(c|b, a)p(b|a)p(a) [not unique]
- Every factorization of the joint distribution is a directed graph
- In the directed graph representation
  - Every variable is a node
  - The conditionals in the factor corresponding to the variable are represented by directed edges
- Just a compact representation. No additional information
- Any distribution can be represented as the graph above. If we drop edges we get more restricted distributions.

DGM



$$p(x_1, \dots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

p(x1, x2, x3, x4, x5, x6, x7) = ?

Arrows may represent causal relationship.

Directed Graphical Models (Bayes Net)

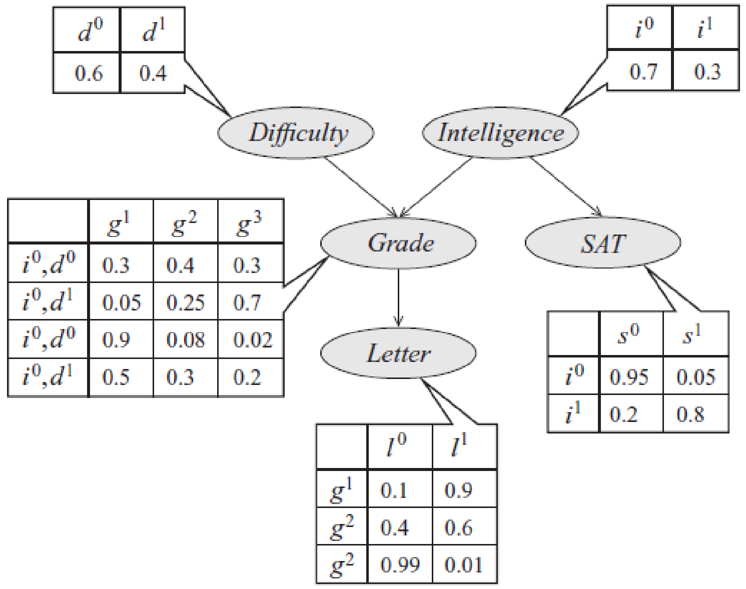
- A **Bayesian network** is specified by a directed *acyclic* graph $G = (V, E)$ with:
  1. One node $i \in V$ for each random variable X_i
  2. One conditional probability distribution (CPD) per node, $p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$, specifying the variable's probability conditioned on its parents' values
- Corresponds 1-1 with a particular factorization of the joint distribution:

$$p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

- Powerful framework for designing *algorithms* to perform probability computations

Example: Bayesian Network

- Consider the following Bayesian network:



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^0,d^0	0.9	0.08	0.02
i^0,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

- What is its joint distribution?

$$
\begin{aligned}
p(x_1, \ldots x_n) &= \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)}) \\
p(d, i, g, s, l) &= p(d)p(i)p(g \mid i, d)p(s \mid i)p(l \mid g)
\end{aligned}
$$

Conditional Independence

Conditional Independence:

$$X \perp\!\!\!\perp Y | V \quad \Leftrightarrow \quad p(X|Y,V) = p(X|V)$$

when $p(Y,V) > 0$. Also

$$X \perp\!\!\!\perp Y | V \quad \Leftrightarrow \quad p(X,Y|V) = p(X|V)\, p(Y|V)$$

In general we can think of conditional independence between **sets of variables**:

$$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{V} \quad \Leftrightarrow \quad p(\mathcal{X},\mathcal{Y}|\mathcal{V}) = p(\mathcal{X}|\mathcal{V})\, p(\mathcal{Y}|\mathcal{V})$$

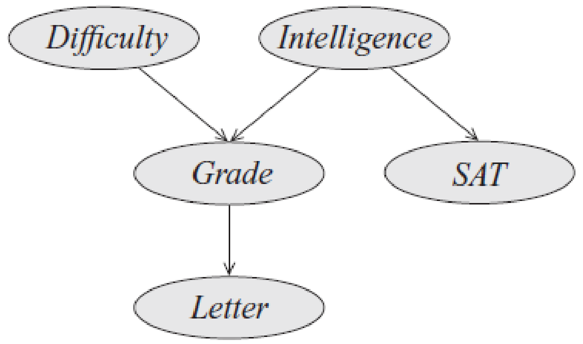
Marginal Independence:

$$X \perp\!\!\!\perp Y \quad \Leftrightarrow \quad X \perp\!\!\!\perp Y | \emptyset \quad \Leftrightarrow \quad p(X,Y) = p(X)\, p(Y)$$

Conditional and Marginal Independence (Examples)

- Amount of Speeding Fine $\perp\!\!\!\perp$ Type of Car | Speed
- Lung Cancer $\perp\!\!\!\perp$ Yellow Teeth | Smoking
- (Position, Velocity)$_{t+1}$ $\perp\!\!\!\perp$ (Position, Velocity)$_{t-1}$ | (Position, Velocity)$_t$, Acceleration$_t$
- Child's Genes $\perp\!\!\!\perp$ Grandparents' Genes | Parents' Genes
- Ability of Team A $\perp\!\!\!\perp$ Ability of Team B
- not (Ability of Team A $\perp\!\!\!\perp$ Ability of Team B | Outcome of A vs B Game)

BayesNet Structure Implies Conditional Independences



A node is conditionally independent of its non-descendents given its parent.

- The joint distribution corresponding to the above BN factors as

$$p(d,i,g,s,l) = p(d)p(i)p(g \mid i,d)p(s \mid i)p(l \mid g)$$

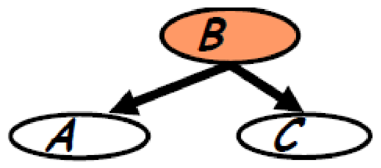
- However, by the chain rule, *any* distribution can be written as

$$p(d,i,g,s,l) = p(d)p(i \mid d)p(g \mid i,d)p(s \mid i,d,g)p(l \mid g,d,i,g,s)$$

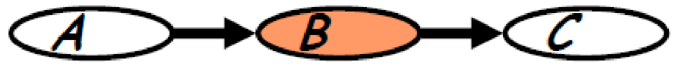
- Thus, we are assuming the following additional independencies:
  $D \perp I, \qquad S \perp \{D,G\} \mid I, \qquad L \perp \{I,D,S\} \mid G.$ What else?

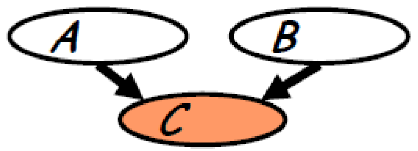
Local Conditional Independence Structures

- Generalizing the above arguments, we obtain that a variable is independent from its non-descendants given its parents



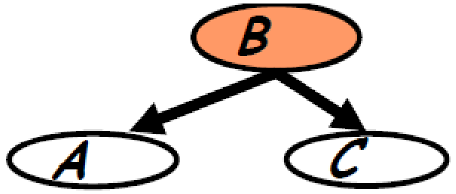
- **Common parent** – fixing B *decouples* A and C
- **Cascade** – knowing B *decouples* A and C





- **V-structure** – Knowing C *couples* A and B
  - This important phenomona is called **explaining away** and is what makes Bayesian networks so powerful

Justification



We'll show that $p(A, C \mid B) = p(A \mid B)p(C \mid B)$ for *any* distribution $p(A, B, C)$ that factors according to this graph structure, i.e.

$$p(A, B, C) = p(B)p(A \mid B)p(C \mid B)$$

Proof.

$$p(A, C \mid B) = \frac{p(A, B, C)}{p(B)} = p(A \mid B)p(C \mid B)$$

□

D-Separation (Global Independences)

Semantics: $X \perp\!\!\!\perp Y | \mathcal{V}$ if $\mathcal{V}$ d-separates X from Y

Definition: $\mathcal{V}$ d-separates X from Y if *every* undirected path[3] between X and Y is **blocked** by $\mathcal{V}$. A path is blocked by $\mathcal{V}$ if there is a node W on the path such that either:

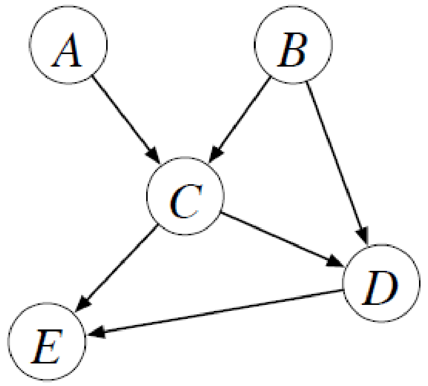
1. W has converging arrows along the path $(\rightarrow W \leftarrow)$[4] and neither W nor its descendants are observed (in $\mathcal{V}$), or
2. W does not have converging arrows along the path $(\rightarrow W \rightarrow$ or $\leftarrow W \rightarrow)$ and W is observed $(W \in \mathcal{V})$.

Corollary: Markov Boundary for X: $\{\text{parents}(X) \cup \text{children}(X) \cup \text{parents-of-children}(X)\}$.

[3] An undirected path ignores the direction of the edges.

[4] Note that converging arrows *along the path* only refers to what happens on that path. Also called a *collider*.

Examples of D-Separation in DAGs



Examples:

- $A \perp\!\!\!\perp B$ since $A \rightarrow C \leftarrow B$ is blocked$_1$ by C, $A \rightarrow C \rightarrow D \leftarrow B$ is blocked$_1$ by D, etc.

- not $(A \perp\!\!\!\perp B | C\ )$ since $A \rightarrow C \leftarrow B$ is not blocked.

- $A \perp\!\!\!\perp D | \{B, C\}$ since $A \rightarrow C \rightarrow D$ is blocked$_2$ by C, $A \rightarrow C \leftarrow B \rightarrow D$ is blocked$_2$ by B, and $A \rightarrow C \rightarrow E \leftarrow D$ is blocked$_2$ by C.

- not $(A \perp\!\!\!\perp B | E)$ since $A \rightarrow C \leftarrow B$ is not blocked.

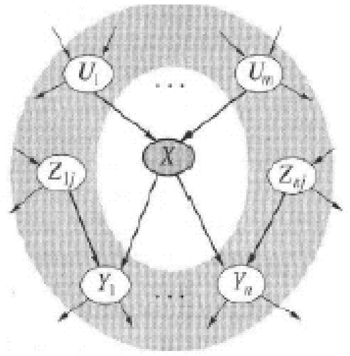
Note that it is the *absence of edges* that conveys conditional independence.

Markov Blankets

- The conditional P(x_i | x_1, …, x_{i-1}, x_{i+1}, …, x_n) looks intimidating, but recall Markov Blankets:
  - Let MB(x_i) be the Markov Blanket of x_i, then

$$P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid MB(x_i))$$

- For a BN, the Markov Blanket of x_i is the set containing its parents, children, and co-parents

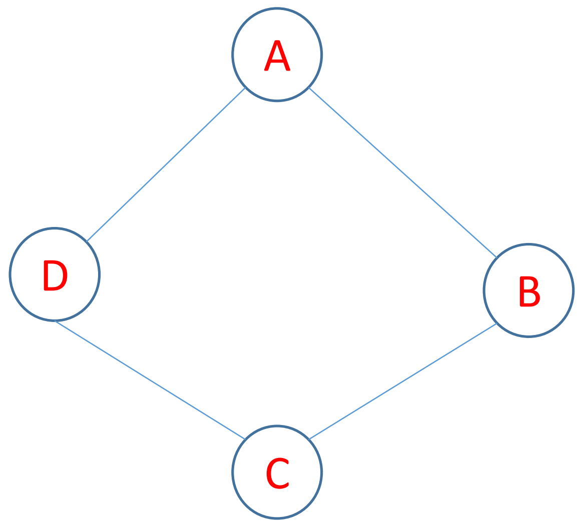


- For an MRF, the Markov Blanket of x_i is its immediate neighbors

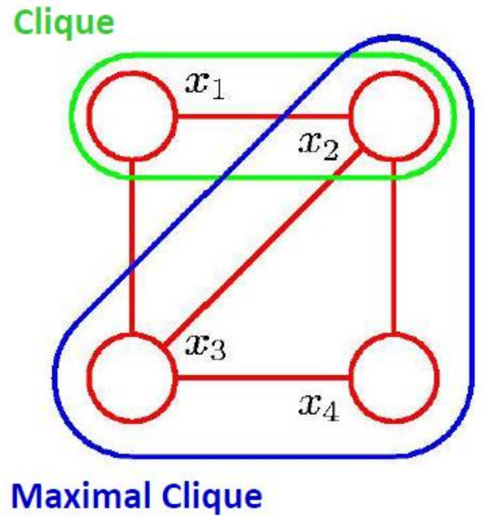
Independence Maps

- Let $I(G)$ be the set of all conditional independencies implied by the directed acyclic graph (DAG) G
- Let $I(p)$ denote the set of all conditional independencies that hold for the joint distribution p.
- A DAG G is an **I-map** (independence map) of a distribution p if $I(G) \subseteq I(p)$
    - A fully connected DAG G is an I-map for *any* distribution, since $I(G) = \emptyset \subseteq I(p)$ for all p
- G is a **minimal I-map** for p if the removal of even a single edge makes it not an I-map
    - A distribution may have several minimal I-maps
    - Each corresponds to a specific node-ordering
- G is a **perfect map** (P-map) for distribution p if $I(G) = I(p)$

Undirected Graphical Models



Undirected Graphical Models



$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

M K-state variables $\rightarrow K^M$ terms in Z

Markov Random Fields

Undirected Graphical Model

In an Undirected Graphical Model, the joint probability over all variables can be written in a factored form:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_j g_j(\mathbf{x}_{C_j})$$

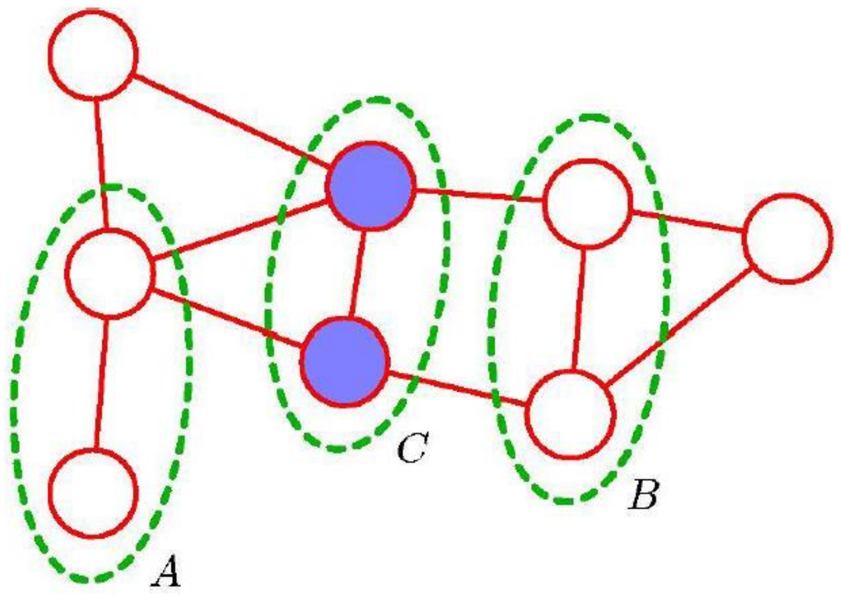
where $\mathbf{x} = (x_1, \ldots, x_K)$, and

$$C_j \subseteq \{1, \ldots, K\}$$

are subsets of the set of all variables, and $\mathbf{x}_S \equiv (x_k : k \in S)$.

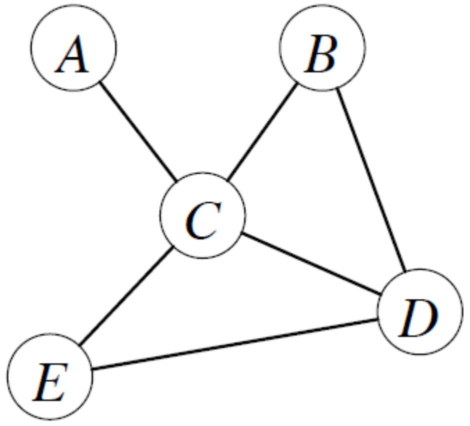
Graph Specification: Create a node for each variable. Connect nodes i and k if there exists a set C_j such that both $i \in C_j$ and $k \in C_j$. These sets form the *cliques* of the graph (fully connected subgraphs).

Independences: Undirected Graphical Models



$$A \perp\!\!\!\perp B \mid C$$

Independences in Undirected Models



$$p(A,B,C,D,E) = \frac{1}{Z} g_1(A,C) g_2(B,C,D) g_3(C,D,E)$$

Fact: $X \perp\!\!\!\perp Y | \mathcal{V}$ if every path between X and Y contains some node $V \in \mathcal{V}$

Corollary: Given the neighbors of X, the variable X is conditionally independent of all other variables: $X \perp\!\!\!\perp Y \,|\, \mathrm{ne}(X), \quad \forall Y \notin \{X\} \cup \mathrm{ne}(X)$

Markov Blanket: $\mathcal{V}$ is a Markov Blanket for X iff $X \perp\!\!\!\perp Y | \mathcal{V}$ for all $Y \notin \{X \cup \mathcal{V}\}$.

Soundness

- $p(\mathbf{x})$ is a *Gibbs distribution* over G if it can be written as

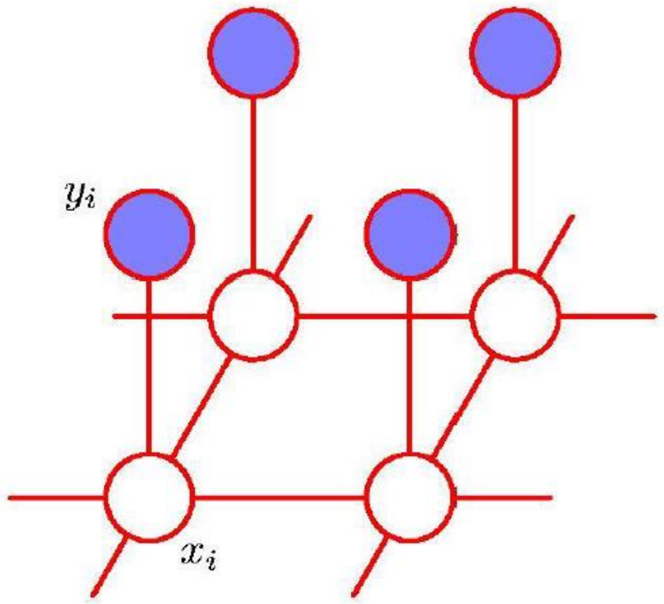
$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(\mathbf{x}_c),$$

  where the variables in each potential $c \in C$ form a clique in G

- Theorem (**soundness of separation**): If $p(\mathbf{x})$ is a Gibbs distribution for G, then G is an I-map for $p(\mathbf{x})$, i.e. $I(G) \subseteq I(p)$

- A distribution is **positive** if $p(\mathbf{x}) > 0$ for all $\mathbf{x}$
- Theorem (**Hammersley-Clifford**, 1971): If $p(\mathbf{x})$ is a positive distribution and G is an I-map for $p(\mathbf{x})$, then $p(\mathbf{x})$ is a Gibbs distribution that factorizes over G

Image Denoising using MRF



$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}$$

$$\begin{aligned} E(\mathbf{x}, \mathbf{y}) &= h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j \\ &\quad - \eta \sum_i x_i y_i \end{aligned}$$

Conditional Random Fields

- **Conditional random fields** are undirected graphical models of conditional distributions $p(\mathbf{Y} \mid \mathbf{X})$
  - $\mathbf{Y}$ is a set of **target variables**
  - $\mathbf{X}$ is a set of **observed variables**
- We typically show the graphical model using just the $\mathbf{Y}$ variables
- Potentials are a function of $\mathbf{X}$ and $\mathbf{Y}$

CRF

- A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies the conditional distribution

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$$

  with partition function

$$Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}_c, \hat{\mathbf{y}}_c).$$

- As before, two variables in the graph are connected with an undirected edge if they appear together in the scope of some factor
- The only difference with a standard Markov network is the normalization term – before marginalized over $\mathbf{X}$ and $\mathbf{Y}$, now only over $\mathbf{Y}$

CRF

- Factors may depend on a large number of variables
- We typically parameterize each factor as a log-linear function,

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\{\mathbf{w} \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\}$$

- $\mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)$ is a feature vector

  - Undirected graphical models very popular in applications such as computer vision: segmentation, stereo, de-noising
  - Grids are particularly popular, e.g., pixels in an image with 4-connectivity

Factorization

Directed graphs:

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

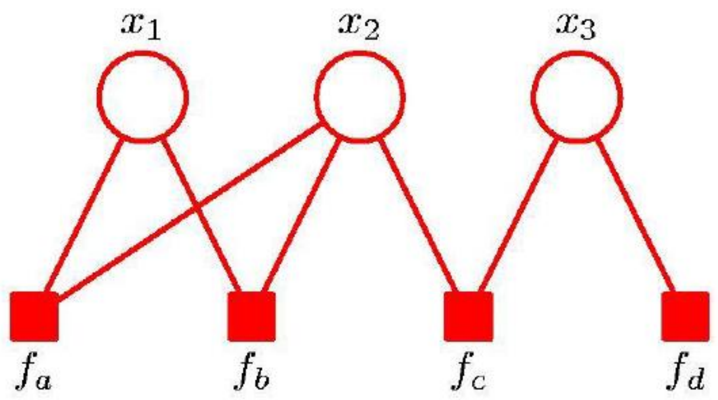
Undirected graphs:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C} \psi_C(\mathbf{x}_C)$$

Both have the form of products of factors:

$$p(\mathbf{x}) = \prod_{s} f_s(\mathbf{x}_s)$$

Factor Graphs

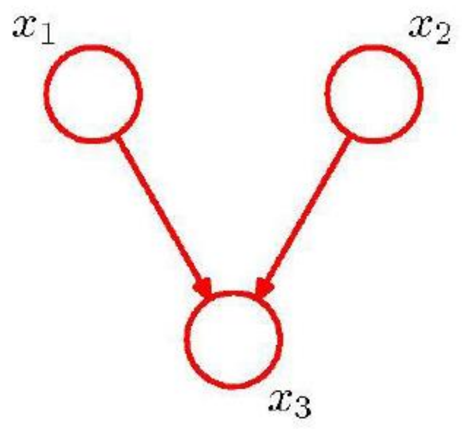


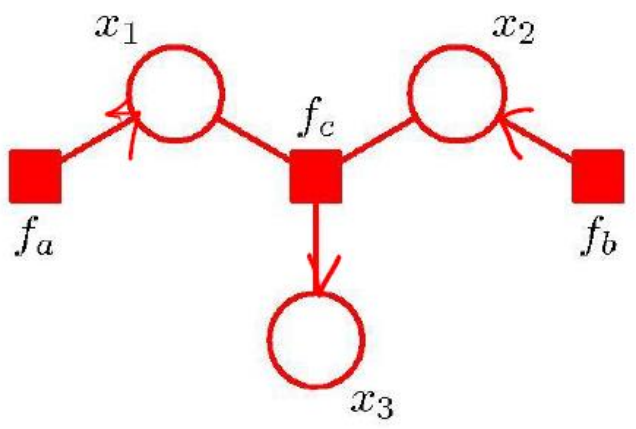
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

Directed Graphs to Factor Graphs

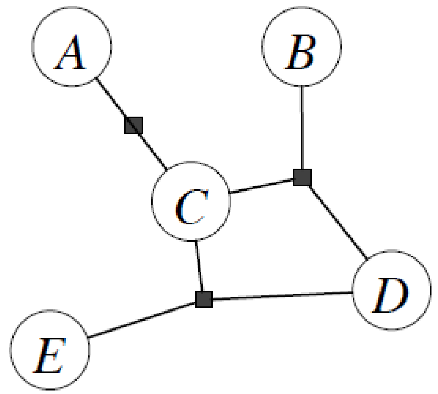
$$p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$$



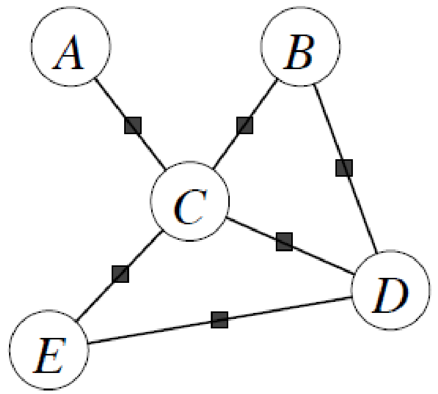


$$\begin{aligned} f_a(x_1) &= p(x_1) \\ f_b(x_2) &= p(x_2) \\ f_c(x_1, x_2, x_3) &= p(x_3|x_1, x_2) \end{aligned}$$

Factor Graphs



(a)



(b)

Two types of nodes:

- The circles in a factor graph represent random variables (e.g. A).
- The filled dots represent factors in the joint distribution (e.g. $g_1(\cdot)$).

(a) $p(A, B, C, D, E) = \frac{1}{Z} g_1(A, C) g_2(B, C, D) g_3(C, D, E)$

(b) $p(A, B, C, D, E) = \frac{1}{Z} g_1(A, C) g_2(B, C) g_3(C, D) g_4(B, D) g_5(C, E) g_6(D, E)$

The g_i are non-negative functions of their arguments, and Z is a normalization constant. E.g. in (a), if all variables are discrete and take values in $\mathcal{A} \times \mathcal{B} \times \mathcal{C} \times \mathcal{D} \times \mathcal{E}$:

$$Z = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} \sum_{c \in \mathcal{C}} \sum_{d \in \mathcal{D}} \sum_{e \in \mathcal{E}} g_1(A = a, C = c) g_2(B = b, C = c, D = d) g_3(C = c, D = d, E = e)$$

Conditional Independences in Factor Graphs

Two nodes are neighbors if they share a common factor.

Definition: A *path* is a sequence of neighboring nodes.

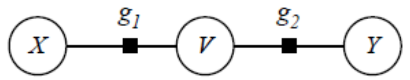
Fact: $X \perp\!\!\!\perp Y | \mathcal{V}$ if every path between X and Y contains some node $V \in \mathcal{V}$

Corollary: Given the neighbors of X, the variable X is conditionally independent of all other variables: $X \perp\!\!\!\perp Y \,|\, \mathrm{ne}(X), \quad \forall Y \notin \{X\} \cup \mathrm{ne}(X)$

Justification of Independences

Assume:

$$p(X,Y,V) = \frac{1}{Z} g_1(X,V) g_2(Y,V), \tag{1}$$



We want to show conditional independence:

$$X \perp\!\!\!\perp Y | V \quad \Leftrightarrow \quad p(X|Y,V) = p(X|V) \tag{2}$$

Summing (1) over X we get:

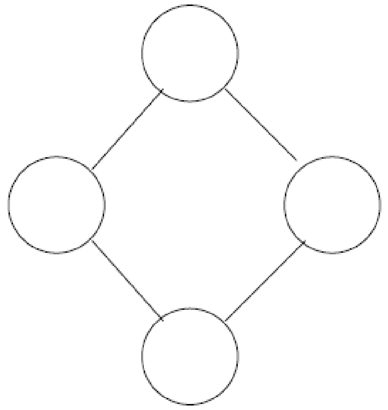
$$p(Y,V) = \frac{1}{Z} \left[\sum_X g_1(X,V) \right] g_2(Y,V) \tag{3}$$

Dividing (1) by (3) we get:

$$p(X|Y,V) = \frac{g_1(X,V)}{\sum_X g_1(X,V)} \tag{4}$$

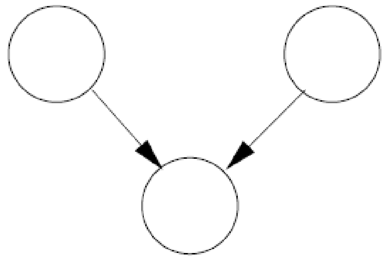
Since the rhs. of (4) doesn't depend on Y, it follows that X is independent of Y given V. Therefore factorizaton (1) implies conditional independence (2).

Expressive Powers of Directed and Undirected Graphs



No Directed Graph (Bayesian network) can represent these and only these independencies

No matter how we direct the arrows there will always be two non-adjacent parents sharing a common child $\Longrightarrow$ dependence in Directed Graph but independence in Undirected Graph.

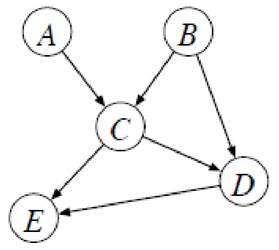


No Undirected Graph or Factor Graph can represent these and only these independencies

Directed graphs are better at expressing causal generative models, undirected graphs are better at representing soft constraints between variables.

Inference in Graphical Models

Inference in a Graphical Model

Consider the following graph:  which represents:

$$p(A,B,C,D,E) = p(A)p(B)p(C|A,B)p(D|B,C)p(E|C,D)$$

Inference: evaluate the probability distribution over some set of variables, given the values of another set of variables.

For example, how can we compute $p(A|C=c)$? Assume each variable is binary.

Naive method:

$$p(A, C=c) = \sum_{B,D,E} p(A,B,C=c,D,E) \qquad [16 \text{ terms}]$$

$$p(C=c) = \sum_{A} p(A, C=c) \qquad [2 \text{ terms}]$$

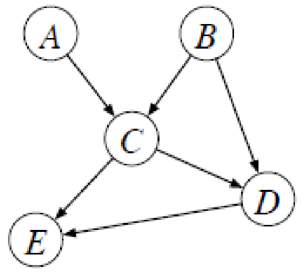
$$p(A|C=c) = \frac{p(A,C=c)}{p(C=c)} \qquad [2 \text{ terms}]$$

Total: 16+2+2 = 20 terms have to be computed and summed

Efficient Computation by Reordering Operations

$$\begin{aligned}\sum_x \sum_y xy &= x_1y_1 + x_2y_1 + x_1y_2 + x_2y_2 \\ &= (x_1 + x_2)(y_1 + y_2)\end{aligned}$$

Efficient Computation

Consider the following graph: 

which represents:

$$p(A,B,C,D,E) = p(A)p(B)p(C|A,B)p(D|B,C)p(E|C,D)$$

Computing $p(A|C=c)$.

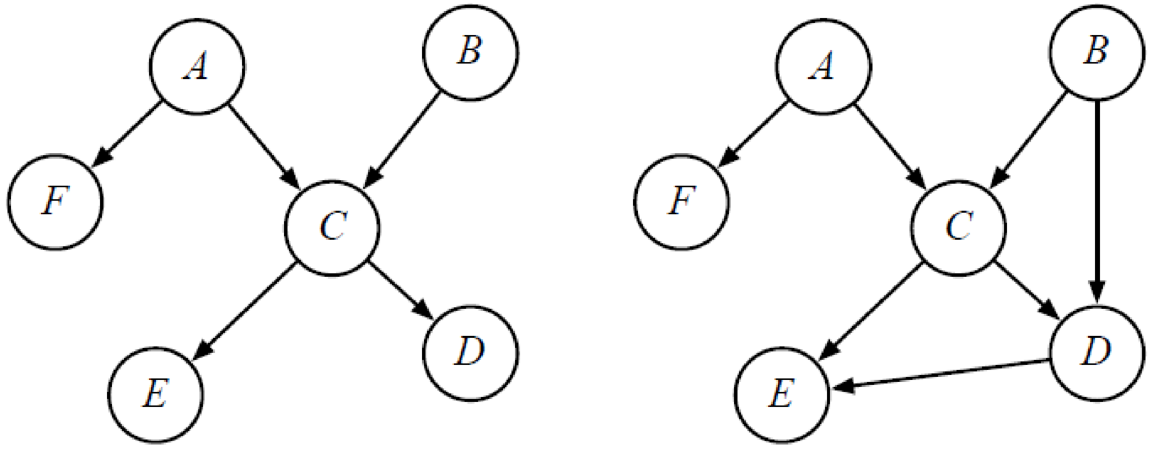
More efficient method:

$$\begin{aligned}
p(A, C=c) &= \sum_{B,D,E} p(A)p(B)p(C=c|A,B)p(D|B,C=c)p(E|C=c,D) \\
&= \sum_{B} p(A)p(B)p(C=c|A,B) \sum_{D} p(D|B,C=c) \sum_{E} p(E|C=c,D) \\
&= \sum_{B} p(A)p(B)p(C=c|A,B) \qquad \text{[4 terms]}
\end{aligned}$$

Total: $4+2+2 = 8$ terms

Singly Connected DAGs

Definition: A DAG is *singly connected* if its underlying undirected graph is a tree, *ie* there is only one undirected path between any two nodes.

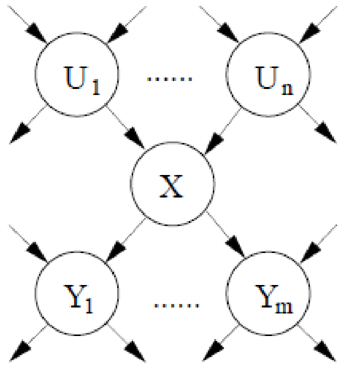


Goal: For some node X we want to compute conditional $p(X|e)$ given evidence (i.e. observed, visible variables) e.

Since we are considering singly connected graphs:

- every node X divides the evidence into upstream e_X^+ and downstream e_X^-
- every edge $X \rightarrow Y$ divides the evidence into upstream e_{XY}^+ and downstream e_{XY}^-.

Belief Propagation (Message Passing)



Idea 1: The probability of a variable X can be found by combining upstream and downstream evidence:

$$p(X|e) = \frac{p(X,e)}{p(e)} = \frac{p(X,e_X^+,e_X^-)}{p(e_X^+,e_X^-)} \propto p(X|e_X^+) \quad \times \quad \underbrace{p(e_X^-|X,e_X^+)}_{X \text{ d-separates } e_X^- \text{ from } e_X^+}$$

$$= p(X|e_X^+)p(e_X^-|X) = \pi(X)\lambda(X)$$

Idea 2: The upstream and downstream evidence can be computed via a local message passing algorithm between the nodes in the graph.

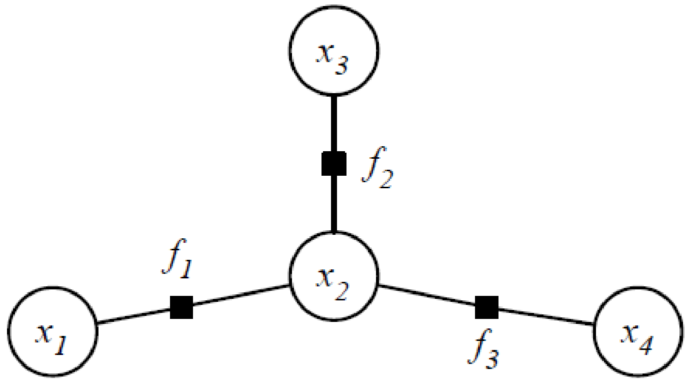
Idea 3: "Don't send back to a node (any part of) the message it sent to you!"

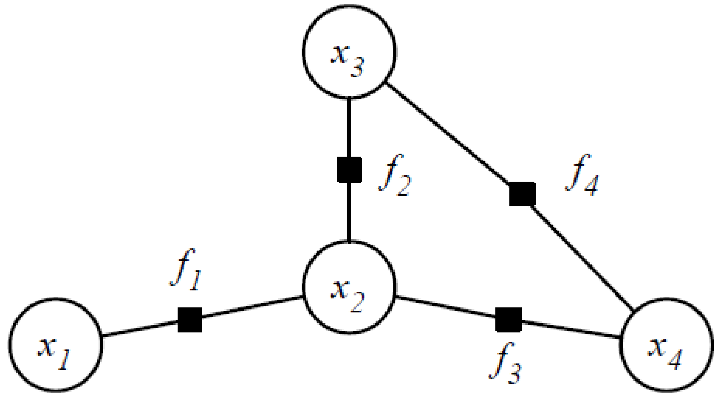
Factor Graph Propagation

Algorithmically and implementationally, it's often easier to convert directed and undirected graphs into factor graphs, and run *factor graph propagation*.

$$\begin{aligned} p(\mathbf{x}) &= p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2) \\ &\equiv f_1(x_1, x_2)f_2(x_2, x_3)f_3(x_2, x_4) \end{aligned}$$

Singly connected vs Multiply connected factor graphs:





Propagation in Factor Graphs

Let $\mathrm{n}(x)$ denote the set of factor nodes that are neighbors of x.
Let $\mathrm{n}(f)$ denote the set of variable nodes that are neighbors of f.

We can compute probabilities in a factor graph by propagating messages from variable nodes to factor nodes and viceversa.

message from variable x to factor f:

$$\mu_{x\to f}(x) = \prod_{h\in \mathrm{n}(x)\setminus\{f\}} \mu_{h\to x}(x)$$

message from factor f to variable x:

$$\mu_{f\to x}(x) = \sum_{\mathbf{x}\setminus x}\left(f(\mathbf{x}) \prod_{y\in \mathrm{n}(f)\setminus\{x\}} \mu_{y\to f}(y)\right)$$

where $\mathbf{x}$ are the variables that factor f depends on, and $\sum_{\mathbf{x}\setminus x}$ is a sum over all variables neighboring factor f except x.

$\mathrm{n}(x)$ denotes the set of factor nodes that are neighbors of x.
$\mathrm{n}(f)$ denotes the set of variable nodes that are neighbors of f.

message from variable x to factor f:

$$\mu_{x \to f}(x) = \prod_{h \in \mathrm{n}(x) \setminus \{f\}} \mu_{h \to x}(x)$$

message from factor f to variable x:

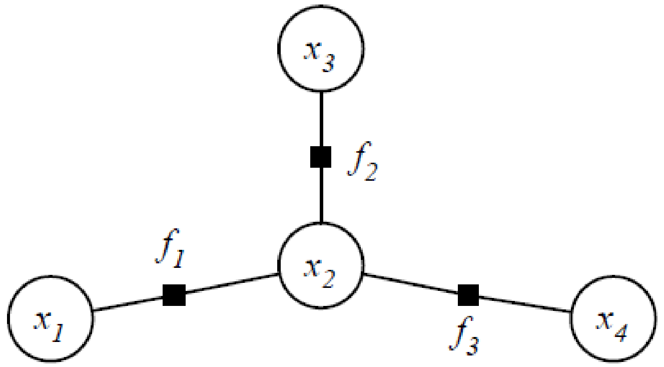
$$\mu_{f \to x}(x) = \sum_{\mathbf{x} \setminus x} \left(f(\mathbf{x}) \prod_{y \in \mathrm{n}(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

If a variable has only one factor as a neighbor, it can initiate message propagation.

Once a variable has received all messages from its neighboring factor nodes, one can compute the probability of that variable by multiplying all the messages and renormalising:

$$p(x) \propto \prod_{h \in \mathrm{n}(x)} \mu_{h \to x}(x)$$

Example

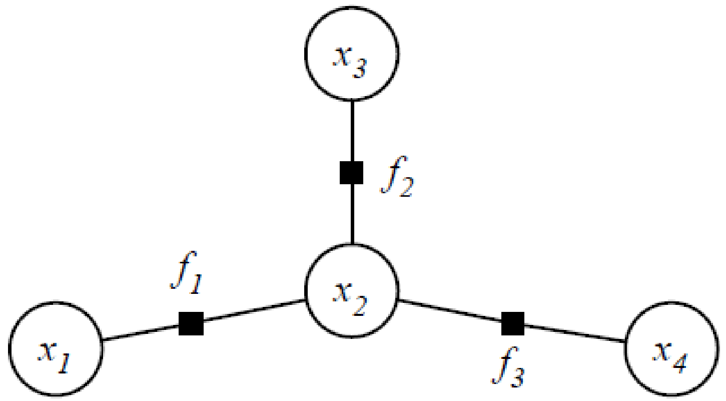


initialise all messages to be constant functions

an example schedule of messages resulting in computing $p(x_4)$:

message direction	message value
$x_1 \to f_1$	$1(x_1)$
$x_3 \to f_2$	$1(x_3)$
$f_1 \to x_2$	$\sum_{x_1} f_1(x_1, x_2)1(x_1)$
$f_2 \to x_2$	$\sum_{x_3} f_2(x_3, x_2)1(x_3)$
$x_2 \to f_3$	$\left(\sum_{x_1} f_1(x_1, x_2)\right)\left(\sum_{x_3} f_2(x_3, x_2)\right)$
$f_3 \to x_4$	$\sum_{x_2} f_3(x_2, x_4)\left(\sum_{x_1} f_1(x_1, x_2)\right)\left(\sum_{x_3} f_2(x_3, x_2)\right)$

where $1(x)$ is a constant uniform function of x

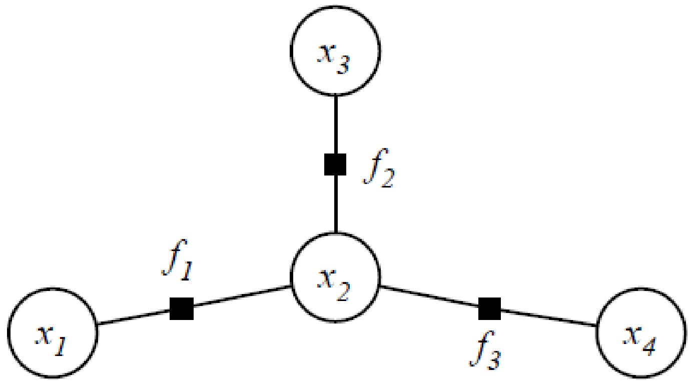


an example schedule of messages resulting in computing $p(x_4|x_1 = a)$:

message direction	message value
$x_1 \to f_1$	$\delta(x_1 = a)$
$x_3 \to f_2$	$1(x_3)$
$f_1 \to x_2$	$\sum_{x_1} f_1(x_1, x_2)\delta(x_1 = a) \;=\; f_1(x_1 = a, x_2)$
$f_2 \to x_2$	$\sum_{x_3} f_2(x_3, x_2)1(x_3)$
$x_2 \to f_3$	$f_1(x_1 = a, x_2)\left(\sum_{x_3} f_2(x_3, x_2)\right)$
$f_3 \to x_4$	$\sum_{x_2} f_3(x_2, x_4) f_1(x_1 = a, x_2)\left(\sum_{x_3} f_2(x_3, x_2)\right)$

where $\delta(x = a)$ is a delta function

Eliminating Nodes



eliminating observed variables

If a variable x_i is **observed**, i.e. its value is given, then it is a *constant* in all factor that include x_i.

We can **eliminate** x_i from the graph by removing the corresponding node and modifying all neighboring factors to treat it as a constant.

- **eliminating hidden variables**

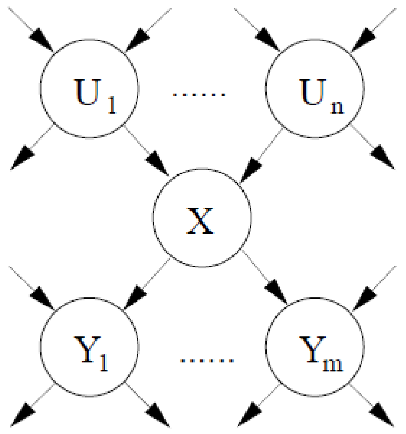
  If a variable x_i is **hidden** and we are not interested in it we can eliminate it from the graph by summing over all its values.

$$\begin{aligned}
\sum_{x_i} p(\mathbf{x}) &= \frac{1}{Z} \sum_{x_i} \prod_{j} f_j(\mathbf{x}_{S_j}) \\
&= \frac{1}{Z} \prod_{j \notin \mathrm{n}(x_i)} f_j(\mathbf{x}_{S_j}) \left(\sum_{x_i} \prod_{k \in \mathrm{n}(x_i)} f_k(\mathbf{x}_{S_k}) \right) \\
&= \frac{1}{Z} \prod_{j \notin \mathrm{n}(x_i)} f_j(\mathbf{x}_{S_j}) \quad f_{\mathrm{new}}(\mathbf{x}_{S_{\mathrm{new}}})
\end{aligned}$$

where $f_{\mathrm{new}}(\mathbf{x}_{S_{\mathrm{new}}}) = \sum_{x_i} \prod_{k \in \mathrm{n}(x_i)} f_k(\mathbf{x}_{S_k})$ and $S_{\mathrm{new}} = \bigcup_{k \in \mathrm{n}(x_i)} S_k \setminus \{i\}$.

This causes all its neighboring factor nodes to merge into one new factor node.

Belief Propagation in Directed Graphs



top-down upstream evidence:
(message U_i sends to X)

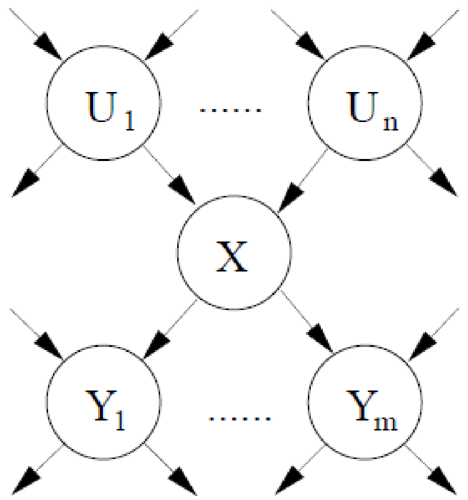
$\pi_X(U_i) = p(U_i|e^+_{U_iX})$

bottom-up downstream evidence:
(message Y_j sends to X)

$\lambda_{Y_j}(X) = p(e^-_{XY_j}|X)$

To update the probability of X given the evidence:

$$\begin{aligned}
\mathrm{BEL}(X) = p(X|e) &= \frac{1}{Z}\lambda(X)\,\pi(X) \\
\lambda(X) &= \prod_j \lambda_{Y_j}(X) \\
\pi(X) &= \sum_{U_1\cdots U_n} p(X|U_1,\ldots,U_n)\prod_i \pi_X(U_i)
\end{aligned}$$



top-down upstream evidence:
(message U_i sends to X)

$\pi_X(U_i) = p(U_i|e^+_{U_iX})$

bottom-up downstream evidence:
(message Y_j sends to X)

$\lambda_{Y_j}(X) = p(e^-_{XY_j}|X)$

Bottom-up propagation, message X sends to U_i:

$$\lambda_X(U_i) \;=\; \sum_X \lambda(X) \sum_{U_k:k\neq i} p(X|U_1,\ldots,U_n) \prod_{k\neq i} \pi_X(U_k)$$

Top-down propagation, message X sends to Y_j:

$$\pi_{Y_j}(X) \;=\; \frac{1}{Z}\Big[\prod_{k\neq j} \lambda_{Y_k}(X)\Big] \sum_{U_1\cdots U_n} p(X|U_1,\ldots,U_n) \prod_i \pi_X(U_i) \;=\; \frac{1}{Z}\frac{\mathrm{BEL}(X)}{\lambda_{Y_j}(X)}$$

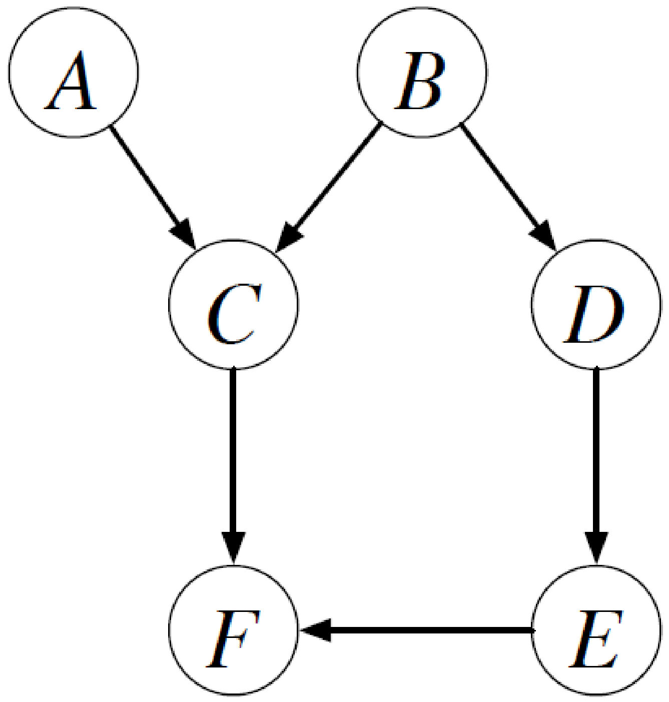
Inference in Multiply Connected DAGs

The Junction Tree algorithm: Form an undirected graph from your directed graph such that no additional conditional independence relationships have been created (this step is called "moralization"). Lump variables in cliques together and form a tree of cliques—this may require a nasty step called "triangulation". Do inference in this tree of cliques.

Cutset Conditioning: or "reasoning by assumptions". Find a small set of variables which, if they were given (i.e. known) would render the remaining graph singly connected. For each value of these variables run belief propagation on the singly connected network. Average the resulting beliefs with the appropriate weights (given by normalizing constants).

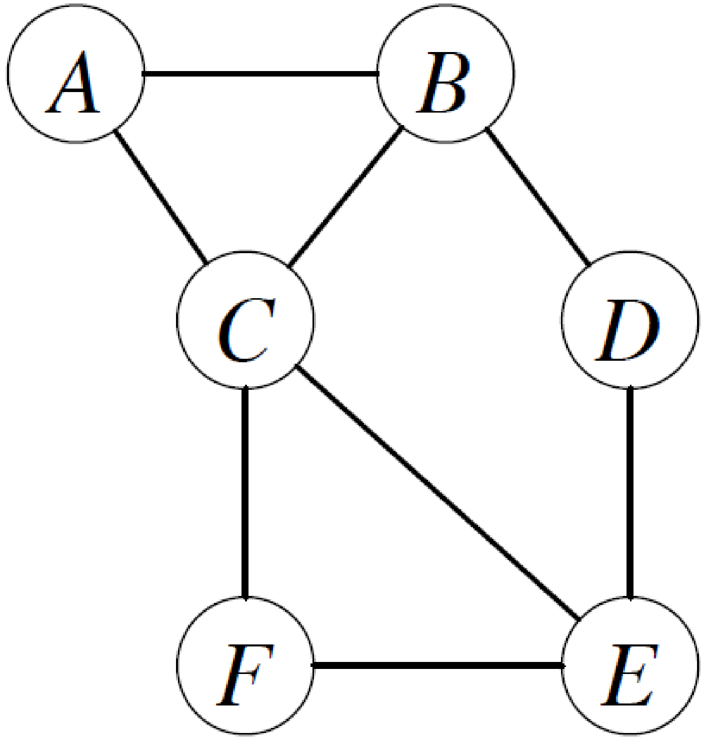
Loopy Belief Propagation: just use BP although there are loops. In this case the terms "upstream" and "downstream" are not clearly defined. No guarantee of convergence, except for certain special graphs, but often works well in practice (c.f. "turbo-decoding" for error-correcting codes).

Junction Tree Algorithm: Step 1



starting with a DAG...

Junction Tree Algorithm: Step 2

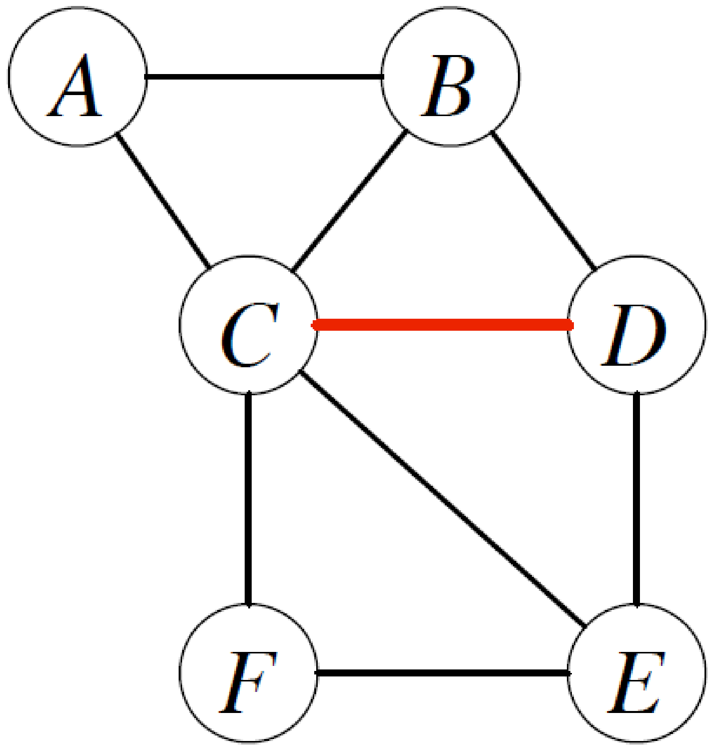


moralize by marrying the parents of each node

remove edge directions

this results in an undirected graph with no additional conditional independence relations

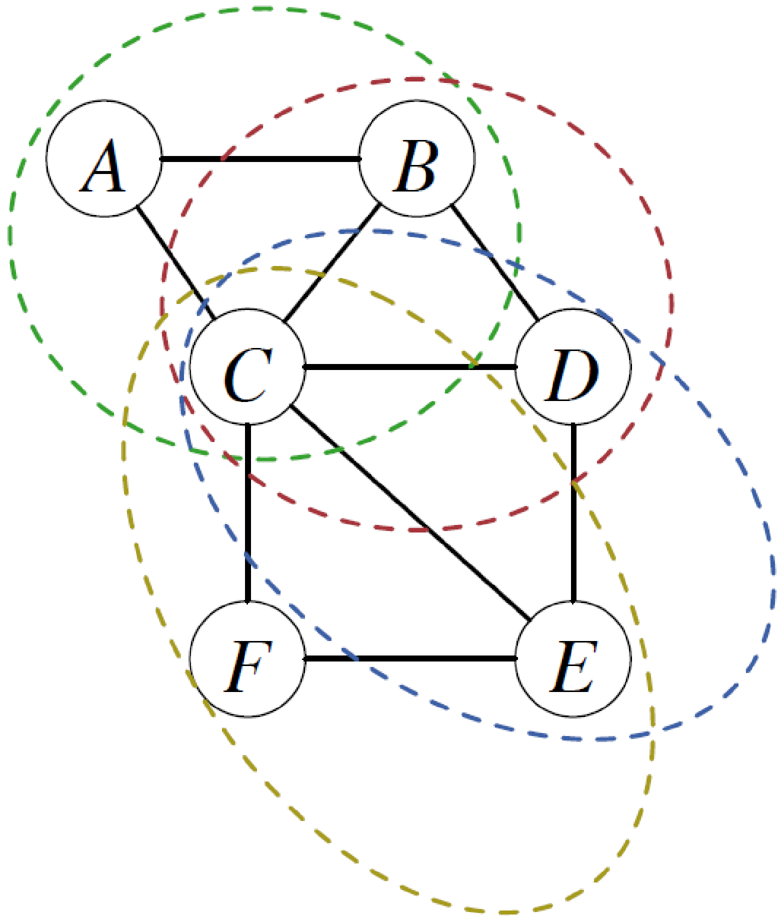
Junction Tree Algorithms: Step 3



triangulate so that there is no loop of length $>$ 3 without a chord

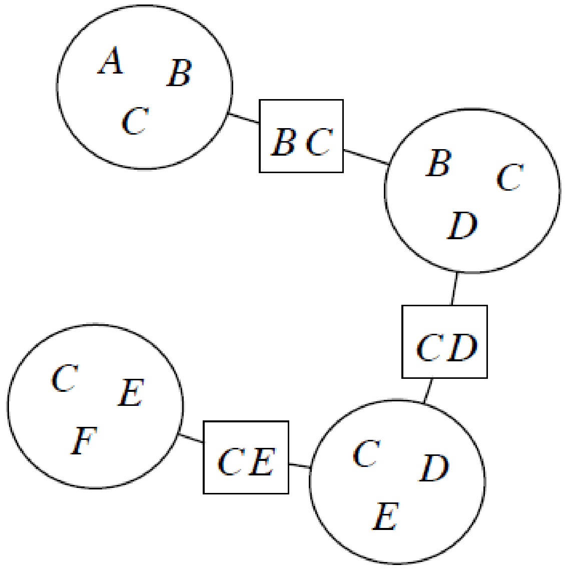
this is necessary so that the final junction tree satisfies the running intersection property

Junction Tree Algorithm: Step 4



find cliques of the moralized, triangulated graph

Junction Tree Algorithm: Step 5



- form **junction tree**: tree of (overlapping) sets of variables
- the **running intersection property** means that if a variable appears in more than one clique (e.g. C), it appears in all intermediate cliques in the tree.
- the junction tree propagation algorithm ensures that neighboring cliques have consistent probability distribution
- local consistency $\rightarrow$ global consistency

Approximate Inference

- Gibbs Sampling

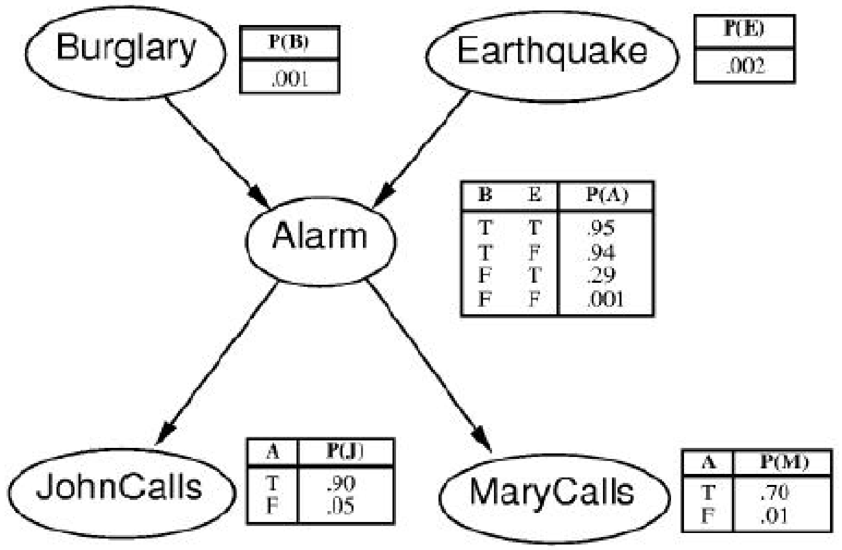
- Variational Inference
    - Mean field approximation
    - Loopy belief propagation

Gibbs Sampling for Graphical Models

- The GS algorithm:
  1. Suppose the graphical model contains variables $x_1,\ldots,x_n$
  2. Initialize starting values for $x_1,\ldots,x_n$
  3. Do until convergence:
     1. Pick an ordering of the n variables (can be fixed or random)
     2. For each variable x_i in order:
        1. Sample $x \sim P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, i.e. the conditional distribution of x_i given the current values of all other variables
        2. Update $x_i \leftarrow x$

- When we update x_i, we <u>immediately</u> use its new value for sampling other variables x_j

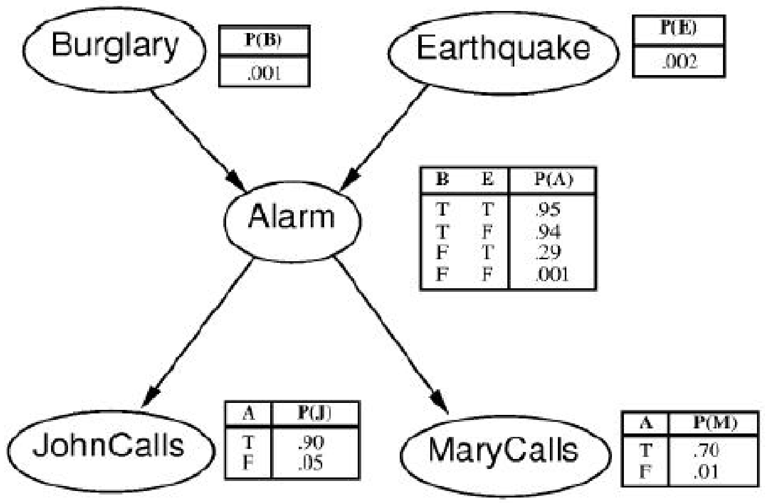
Gibbs Sampling Example



t	B	E	A	J	M
0	F	F	F	F	F
1					
2					
3					
4					

- Consider the alarm network
  - Assume we sample variables in the order B,E,A,J,M
  - Initialize all variables at t = 0 to False

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F				
2					
3					
4					

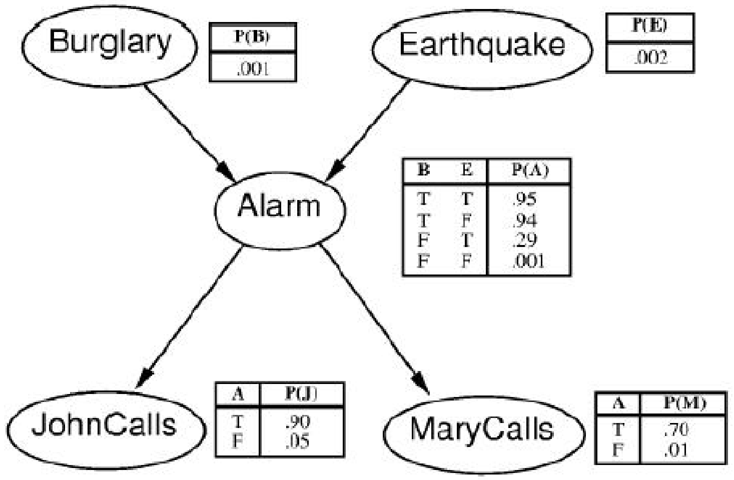
- Sampling P(B|A,E) at t = 1: Using Bayes Rule,

$$P(B \mid A, E) \propto P(A \mid B, E)P(B)$$

- A=false, E=false, so we compute:

$$P(B = T \mid A = F, E = F) \propto (0.06)(0.01) = 0.0006$$
$$P(B = F \mid A = F, E = F) \propto (0.999)(0.999) = 0.9980$$

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T			
2					
3					
4					

- Sampling P(E|A,B): Using Bayes Rule,

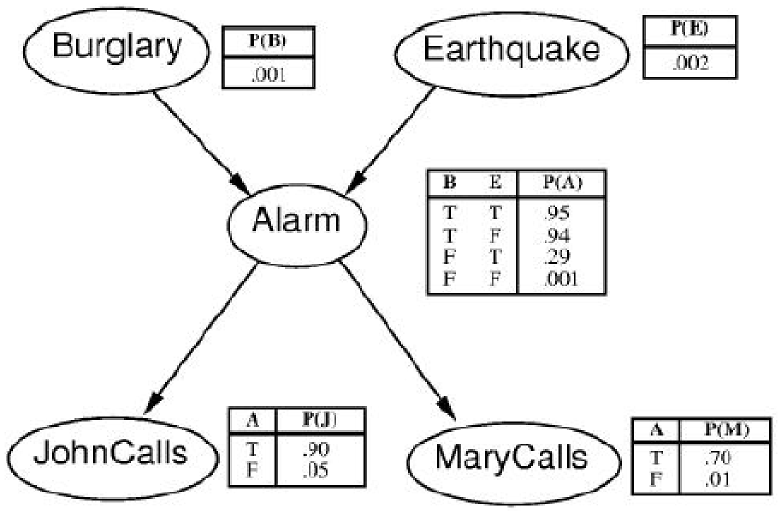
$$P(E \mid A, B) \propto P(A \mid B, E)P(E)$$

- (A,B) = (F,F), so we compute the following,

$$P(E = T \mid A = F, B = F) \propto (0.71)(0.02) = 0.0142$$

$$P(E = F \mid A = F, B = F) \propto (0.999)(0.998) = 0.9970$$

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F		
2					
3					
4					

- Sampling P(A|B,E,J,M): Using Bayes Rule,

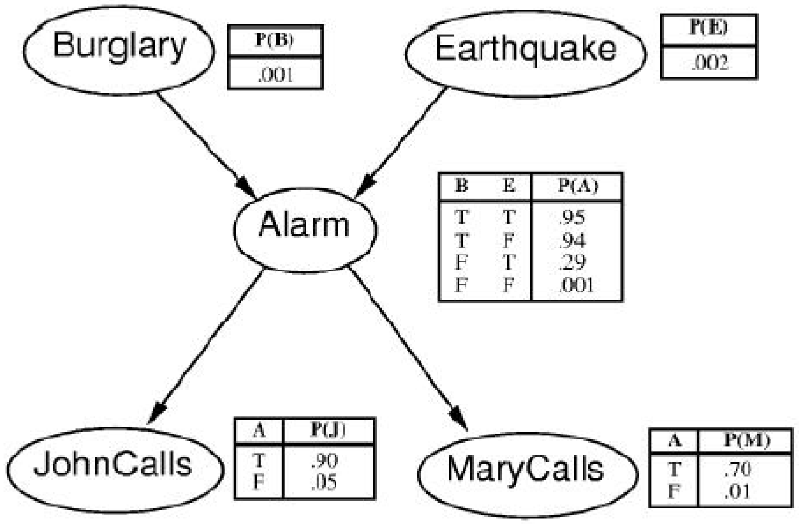
$$P(A \mid B, E, J, M) \propto P(J \mid A) P(M \mid A) P(A \mid B, E)$$

- (B,E,J,M) = (F,T,F,F), so we compute:

$$P(A = T \mid B = F, E = T, J = F, M = F) \propto (0.1)(0.3)(0.29) = 0.0087$$

$$P(A = F \mid B = F, E = T, J = F, M = F) \propto (0.95)(0.99)(0.71) = 0.6678$$

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	
2					
3					
4					

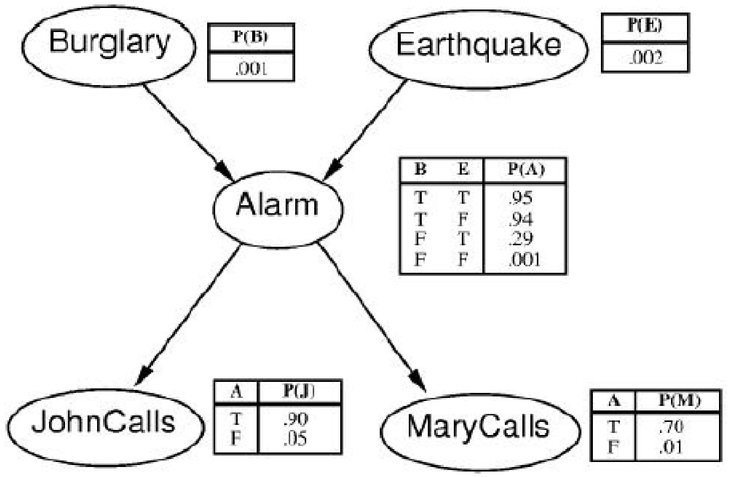
- Sampling P(J|A): No need to apply Bayes Rule

- A = F, so we compute the following, and sample

$$P(J = T \mid A = F) \propto 0.05$$

$$P(J = F \mid A = F) \propto 0.95$$

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	F
2					
3					
4					

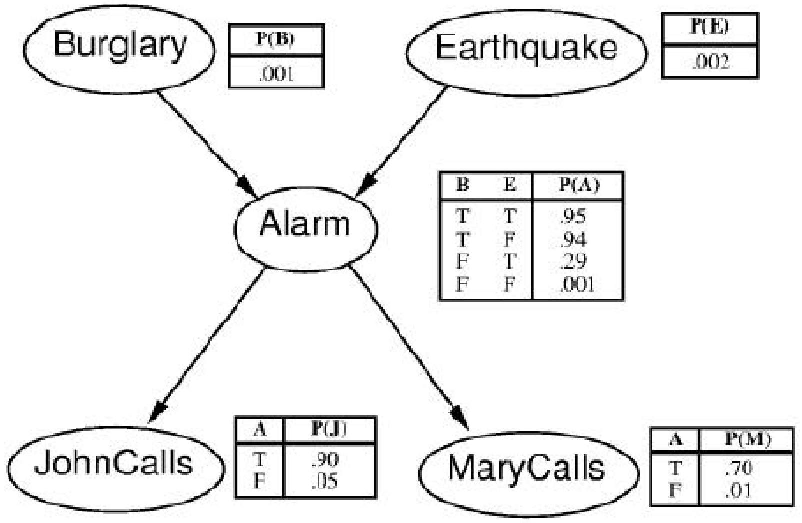
- Sampling P(M|A): No need to apply Bayes Rule

- A = F, so we compute the following, and sample

$$P(M=T \mid A=F) \propto 0.01$$

$$P(M=F \mid A=F) \propto 0.99$$

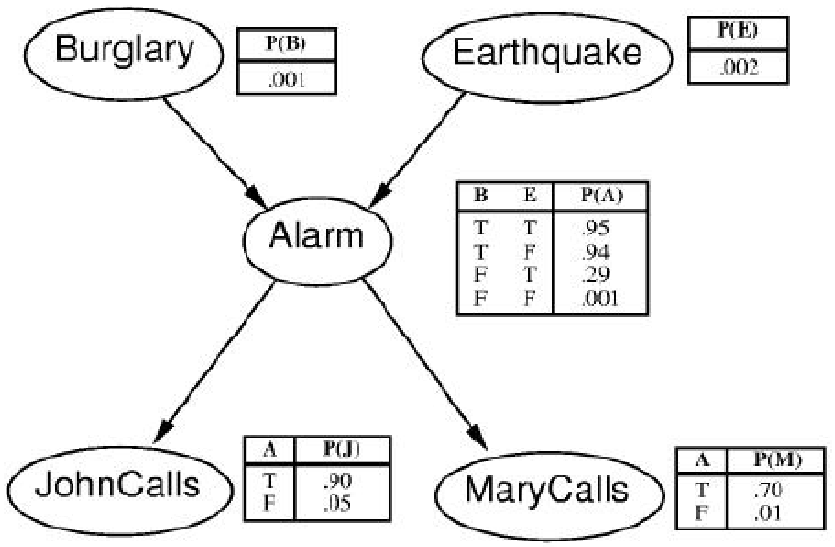
Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	F
2	F	T	T	T	T
3					
4					

- Now t = 2, and we repeat the procedure to sample new values of B,E,A,J,M …

Contd.



t	B	E	A	J	M
0	F	F	F	F	F
1	F	T	F	T	F
2	F	T	T	T	T
3	T	F	T	F	T
4	T	F	T	F	F

- Now t = 2, and we repeat the procedure to sample new values of B,E,A,J,M …

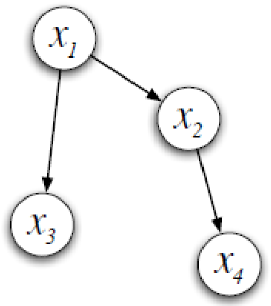
- And similarly for t = 3, 4, etc.

Summary

- Elegant method for representing probabilistic models
- Factorization and Independences
- Exact Inference
- Approximate Inference

Learning Probabilistic Graphical Models

Learning parameters



$$p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)$$

θ_2	x_2		
x_1	0.2	0.3	0.5
	0.1	0.6	0.3

Assume each variable x_i is discrete and can take on K_i values.

The parameters of this model can be represented as 4 tables: θ_1 has K_1 entries, θ_2 has $K_1 \times K_2$ entries, etc.

These are called **conditional probability tables** (CPTs) with the following semantics:

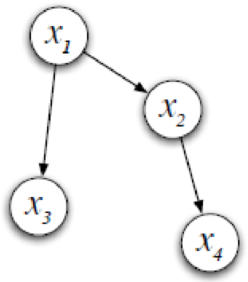
$$p(x_1 = k) = \theta_{1,k} \qquad p(x_2 = k'|x_1 = k) = \theta_{2,k,k'}$$

If node i has M parents, θ_i can be represented either as an $M+1$ dimensional table, or as a 2-dimensional table with $\left(\prod_{j \in \mathrm{pa}(i)} K_j\right) \times K_i$ entries by collapsing all the states of the parents of node i. Note that $\sum_{k'} \theta_{i,k,k'} = 1$.

Assume a data set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$.

How do we learn θ from $\mathcal{D}$?

Learning parameters



Assume a data set $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$. How do we learn $\boldsymbol{\theta}$ from $\mathcal{D}$?

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(x_1|\theta_1)p(x_2|x_1,\theta_2)p(x_3|x_1,\theta_3)p(x_4|x_2,\theta_4)$$

Likelihood:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{x}^{(n)}|\boldsymbol{\theta})$$

Log Likelihood:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N}\sum_{i} \log p(x_i^{(n)}|x_{\mathrm{pa}(i)}^{(n)},\theta_i)$$

This decomposes into sum of functions of θ_i. Each θ_i can be optimized separately:

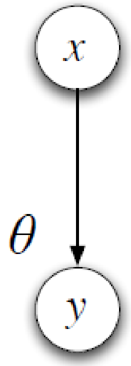
$$\hat{\theta}_{i,k,k'} = \frac{n_{i,k,k'}}{\sum_{k''} n_{i,k,k''}}$$

where $n_{i,k,k'}$ is the number of times in $\mathcal{D}$ where $x_i = k'$ and $x_{\mathrm{pa}(i)} = k$, where k represents a joint configuration of all the parents of i (i.e. takes on one of $\prod_{j\in\mathrm{pa}(i)} K_j$ values)

$$n_2 \quad \begin{array}{c|ccc} & & x_2 & \\ \hline x_1 & 2 & 3 & 0 \\ & 3 & 1 & 6 \end{array} \quad \Rightarrow \quad \theta_2 \quad \begin{array}{c|ccc} & & x_2 & \\ \hline x_1 & 0.4 & 0.6 & 0 \\ & 0.3 & 0.1 & 0.6 \end{array}$$

ML solution: Simply calculate frequencies!

Deriving the Maximum Likelihood Estimate



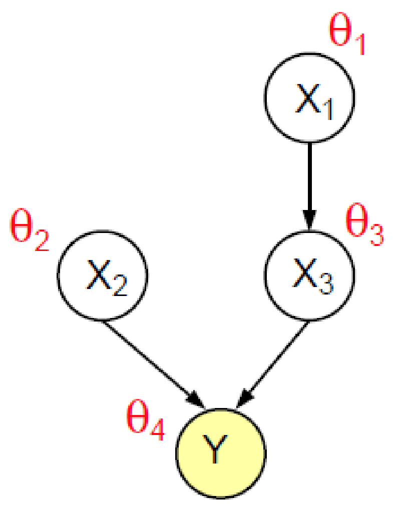
$$p(y|x,\theta) = \prod_{k,\ell} \theta_{k,\ell}^{\delta(x,k)\delta(y,\ell)}$$

Dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)}) : n = 1 \ldots, N\}$

$$\begin{aligned}
\mathcal{L}(\theta) &= \log \prod_n p(y^{(n)}|x^{(n)},\theta) \\
&= \log \prod_n \prod_{k,\ell} \theta_{k,\ell}^{\delta(x^{(n)},k)\delta(y^{(n)},\ell)} \\
&= \sum_{n,k,\ell} \delta(x^{(n)},k)\delta(y^{(n)},\ell) \log \theta_{k,\ell} \\
&= \sum_{k,\ell} \left(\sum_n \delta(x^{(n)},k)\delta(y^{(n)},\ell) \right) \log \theta_{k,\ell} \quad = \sum_{k,\ell} n_{k,\ell} \log \theta_{k,\ell}
\end{aligned}$$

Maximize $\mathcal{L}(\theta)$ w.r.t. θ subject to $\sum_\ell \theta_{k,\ell} = 1$ for all k.

Maximum Likelihood Learning with Hidden Variables

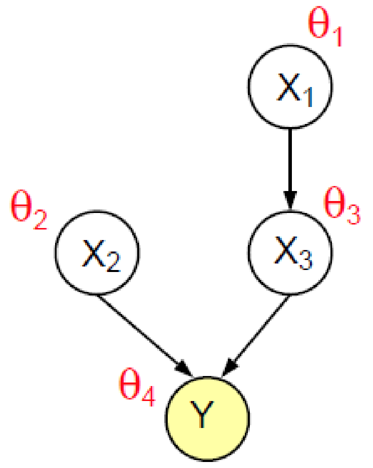


Assume a model parameterised by θ with observable variables Y and hidden variables X

Goal: maximize parameter log likelihood given observed data.

$$\mathcal{L}(\theta) = \log p(Y|\theta) = \log \sum_X p(Y, X|\theta)$$

Maximum Likelihood Learning with Hidden Variables: The EM Algorithm



Goal: maximise parameter log likelihood given observables.

$$\mathcal{L}(\theta) = \log p(Y|\theta) = \log \sum_X p(Y, X|\theta)$$

The Expectation Maximization (EM) algorithm (intuition):

Iterate between applying the following two steps:

- **The E step:** fill-in the hidden/missing variables
- **The M step:** apply complete data learning to filled-in data.

Maximum Likelihood Learning with Hidden Variables: The EM Algorithm

Goal: maximise parameter log likelihood given observables.

$$\mathcal{L}(\theta) = \log p(Y|\theta) = \log \sum_X p(Y, X|\theta)$$

The EM algorithm (derivation):

$$\mathcal{L}(\theta) \quad = \quad \log \sum_X q(X) \frac{p(Y, X|\theta)}{q(X)} \geq \sum_X q(X) \log \frac{p(Y, X|\theta)}{q(X)} = \mathcal{F}(q(X), \theta)$$

- **The E step:** maximize $\mathcal{F}(q(X), \theta^{[t]})$ wrt $q(X)$ holding $\theta^{[t]}$ fixed:
$$q(X) = p(X|Y, \theta^{[t]})$$

- **The M step:** maximize $\mathcal{F}(q(X), \theta)$ wrt θ holding $q(X)$ fixed:
$$\theta^{[t+1]} \leftarrow \text{argmax}_\theta \sum_X q(X) \log p(Y, X|\theta)$$

The E-step requires solving the *inference* problem, finding the distribution over the hidden variables $p(X|Y, \theta^{[t]})$ given the current model parameters. This can be done using **belief propagation** or the **junction tree algorithm**.

Maximum Likelihood Learning without and with Hidden Variables

ML Learning with Complete Data (No Hidden Variables)

Log likelihood decomposes into sum of functions of θ_i. Each θ_i can be optimized separately:

$$\hat{\theta}_{ijk} \leftarrow \frac{n_{ijk}}{\sum_{k'} n_{ijk'}}$$

where n_{ijk} is the number of times in $\mathcal{D}$ where $x_i = k$ and $x_{\mathrm{pa}(i)} = j$.

Maximum likelihood solution: Simply calculate frequencies!

ML Learning with Incomplete Data (i.e. with Hidden Variables)

Iterative EM algorithm

E step: compute expected counts given previous settings of parameters $E[n_{ijk}|\mathcal{D}, \boldsymbol{\theta}^{[t]}]$.

M step: re-estimate parameters using these expected counts

$$\theta_{ijk}^{[t+1]} \leftarrow \frac{E[n_{ijk}|\mathcal{D}, \boldsymbol{\theta}^{[t]}]}{\sum_{k'} E[n_{ijk'}|\mathcal{D}, \boldsymbol{\theta}^{[t]}]}$$

Maximum Entropy (MaxEnt)

- We can approach the modeling task from an entirely different point of view
- Suppose we know some expectations with respect to a (fully general) distribution $p(\mathbf{x})$:

$$\text{(true)} \sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}), \qquad \text{(empirical)} \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} f_i(\mathbf{x}) = \alpha_i$$

- Assuming that the expectations are consistent with one another, there may exist **many** distributions which satisfy them. Which one should we select?

  The most uncertain or flexible one, i.e., the one with maximum entropy.

- This yields a new optimization problem:

$$\max_p H(p(\mathbf{x})) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

$$\text{s.t.} \qquad \sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}) = \alpha_i$$

$$\sum p(\mathbf{x}) = 1 \quad \text{(strictly concave w.r.t. } p(\mathbf{x})\text{)}$$

MaxEnt

- To solve the MaxEnt problem, we form the Lagrangian:

$$L = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) - \sum_i \lambda_i \left(\sum_{\mathbf{x}} p(\mathbf{x}) f_i(\mathbf{x}) - \alpha_i \right) - \mu \left(\sum_{\mathbf{x}} p(\mathbf{x}) - 1 \right)$$

- Then, taking the derivative of the Lagrangian,

$$\frac{\partial L}{\partial p(\mathbf{x})} = -1 - \log p(\mathbf{x}) - \sum_i \lambda_i f_i(\mathbf{x}) - \mu$$

- And setting to zero, we obtain:

$$p^*(\mathbf{x}) = \exp\left(-1 - \mu - \sum_i \lambda_i f_i(\mathbf{x}) \right) = e^{-1-\mu} e^{-\sum_i \lambda_i f_i(\mathbf{x})}$$

- From the constraint $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$ we obtain $e^{1+\mu} = \sum_{\mathbf{x}} e^{-\sum_i \lambda_i f_i(\mathbf{x})} = Z(\lambda)$
- We conclude that the maximum entropy distribution has the form (substituting $w_i = -\lambda_i$)

$$p^*(\mathbf{x}) = \frac{1}{Z(\mathbf{w})} \exp(\sum_i w_i f_i(\mathbf{x}))$$

Bayesian Learning

Apply the basic rules of probability to learning from data.

Data set: $\mathcal{D} = \{x_1, \ldots, x_n\}$ Models: m, m' etc. Model parameters: θ

Prior probability of models: $P(m)$, $P(m')$ etc.
Prior probabilities of model parameters: $P(\theta|m)$
Model of data given parameters (likelihood model): $P(x|\theta, m)$

If the data are independently and identically distributed then:

$$P(\mathcal{D}|\theta, m) = \prod_{i=1}^{n} P(x_i|\theta, m)$$

Posterior probability of model parameters:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

Posterior probability of models:

$$P(m|\mathcal{D}) = \frac{P(m)P(\mathcal{D}|m)}{P(\mathcal{D})}$$

Bayesian parameter learning with no hidden variables

Let n_{ijk} be the number of times $(x_i^{(n)} = k$ and $x_{\mathrm{pa}(i)}^{(n)} = j)$ in $\mathcal{D}$.
For each i and j, $\boldsymbol{\theta}_{ij\cdot}$ is a probability vector of length $K_i \times 1$.

Since x_i is a discrete variable with probabilities given by $\boldsymbol{\theta}_{i,j,\cdot}$, the likelihood is:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_n \prod_i p(x_i^{(n)}|x_{\mathrm{pa}(i)}^{(n)}, \boldsymbol{\theta}) = \prod_i \prod_j \prod_k \theta_{ijk}^{n_{ijk}}$$

If we choose a prior on $\boldsymbol{\theta}$ of the form:

$$p(\boldsymbol{\theta}) = c \prod_i \prod_j \prod_k \theta_{ijk}^{\alpha_{ijk}-1}$$

where c is a normalization constant, and $\sum_k \theta_{ijk} = 1 \ \forall i, j$, then the posterior distribution also has the same form:

$$p(\boldsymbol{\theta}|\mathcal{D}) = c' \prod_i \prod_j \prod_k \theta_{ijk}^{\tilde{\alpha}_{ijk}-1}$$

Dirichlet distribution.

where $\tilde{\alpha}_{ijk} = \alpha_{ijk} + n_{ijk}$.

Bayesian parameter learning with hidden variables

Notation: let $\mathcal{D}$ be the observed data set, $\mathcal{X}$ be hidden variables, and $\boldsymbol{\theta}$ be model parameters. Assume discrete variables and Dirichlet priors on $\boldsymbol{\theta}$

Goal: to infer $p(\boldsymbol{\theta}|\mathcal{D}) = \sum_{\mathcal{X}} p(\mathcal{X}, \boldsymbol{\theta}|\mathcal{D})$

Problem: since (a)

$$p(\boldsymbol{\theta}|\mathcal{D}) = \sum_{\mathcal{X}} p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{D}) p(\mathcal{X}|\mathcal{D}),$$

and (b) for every way of filling in the missing data, $p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{D})$ is a Dirichlet distribution, and (c) there are exponentially many ways of filling in $\mathcal{X}$, it follows that $p(\boldsymbol{\theta}|\mathcal{D})$ is a mixture of Dirichlets with exponentially many terms!

Solutions:

- Find a single best ("Viterbi") completion of $\mathcal{X}$ (Stolcke and Omohundro, 1993)
- Markov chain Monte Carlo methods
- Variational Bayesian (VB) methods (Beal and Ghahramani, 2003)

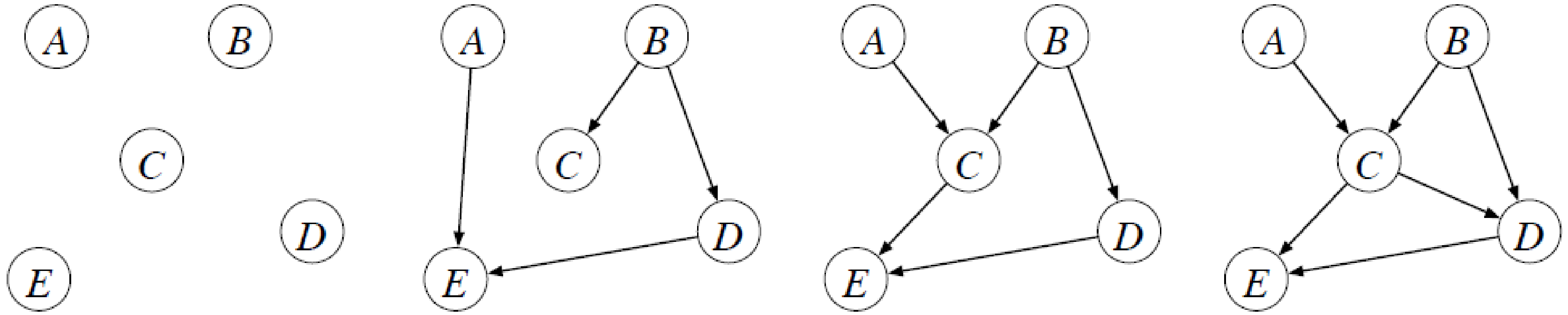
Summary of parameter learning

	Complete (fully observed) data	Incomplete (hidden /missing) data
ML	calculate frequencies	EM
Bayesian	update Dirichlet distributions	MCMC / Viterbi / VB

- For complete data Bayesian learning is not more costly than ML
- For incomplete data VB $\approx$ EM time complexity
- Other parameter priors are possible but Dirichlet is pretty flexible and intuitive.
- For non-discrete data, similar ideas but generally harder inference and learning.

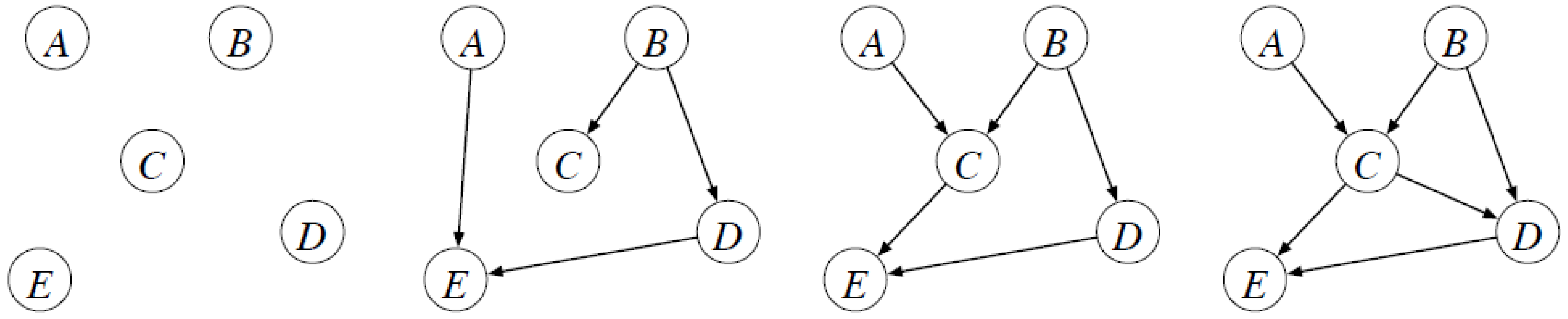
Structure learning

Given a data set of observations of (A, B, C, D, E) can we learn the structure of the graphical model?



Let m denote the graph structure = the set of edges.

Structure learning



Constraint-Based Learning: Use statistical tests of marginal and conditional independence. Find the set of DAGs whose d-separation relations match the results of conditional independence tests.

Score-Based Learning: Use a global score such as the BIC score or Bayesian marginal likelihood. Find the structures that maximize this score.

Score-based structure learning for complete data

Consider a graphical model with structure m, discrete observed data $\mathcal{D}$, and parameters θ. Assume Dirichlet priors.

The Bayesian marginal likelihood score is easy to compute:

$$\text{score}(m) = \log p(\mathcal{D}|m) = \log \int p(\mathcal{D}|\theta, m) p(\theta|m) d\theta$$

$$\text{score}(m) = \sum_i \sum_j \left[\log \Gamma(\sum_k \alpha_{ijk}) - \sum_k \log \Gamma(\alpha_{ijk}) - \log \Gamma(\sum_k \tilde{\alpha}_{ijk}) + \sum_k \log \Gamma(\tilde{\alpha}_{ijk}) \right]$$

where $\tilde{\alpha}_{ijk} = \alpha_{ijk} + n_{ijk}$. **Note that the score decomposes over i.**

One can incorporate structure prior information $p(m)$ as well:

$$\text{score}(m) = \log p(\mathcal{D}|m) + \log p(m)$$

Greedy search algorithm: Start with m. Consider modifications $m \to m'$ (edge deletions, additions, reversals). Accept m' if $\text{score}(m') > \text{score}(m)$. Repeat.

Bayesian inference of model structure: Run MCMC on m.

Bayesian Structural EM for *in*complete data

Consider a graphical model with structure m, observed data $\mathcal{D}$, hidden variables $\mathcal{X}$ and parameters θ

The Bayesian score is generally intractable to compute:

$$\text{score}(m) = p(\mathcal{D}|m) = \int \sum_{\mathcal{X}} p(\mathcal{X}, \theta, \mathcal{D}|m) d\theta$$

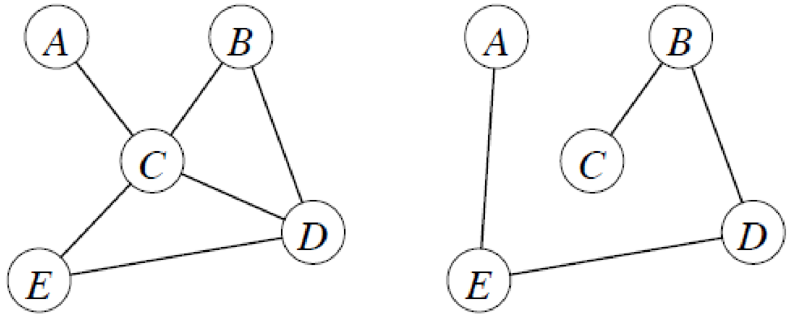
Bayesian Structure EM (Friedman, 1998):

1. compute MAP parameters $\hat{\theta}$ for current model m using EM
2. find hidden variable distribution $p(\mathcal{X}|\mathcal{D}, \hat{\theta})$
3. for a small set of candidate structures compute or approximate

$$\text{score}(m') = \sum_{\mathcal{X}} p(\mathcal{X}|\mathcal{D}, \hat{\theta}) \log p(\mathcal{D}, \mathcal{X}|m')$$

4. $m \leftarrow m'$ with highest score

Learning parameters and structure in undirected graphs



$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_j g_j(\mathbf{x}_{C_j}; \boldsymbol{\theta}_j)$ where $Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \prod_j g_j(\mathbf{x}_{C_j}; \boldsymbol{\theta}_j)$.

Problem: computing $Z(\boldsymbol{\theta})$ is computationally intractable for general (non-tree-structured) undirected models. Therefore, maximum-likelihood learning of parameters is generally intractable, Bayesian scoring of structures is intractable, etc.

Solutions:

- directly approximate $Z(\boldsymbol{\theta})$ and/or its derivatives (cf. Boltzmann machine learning; contrastive divergence; pseudo-likelihood)
- use approx inference methods (e.g. loopy belief propagation, bounding methods, EP).

See: (Murray and Ghahramani, 2004; Murray et al, 2006) for Bayesian learning in undirected models.

Summary

- Graphical models provide a powerful and intuitive framework for modelling and inference.
- Directed, undirected and factor graphs.
- Inference by message passing.
- Parameter and structure learning.