Lecture 9: Undirected Graphical Models
Machine Learning

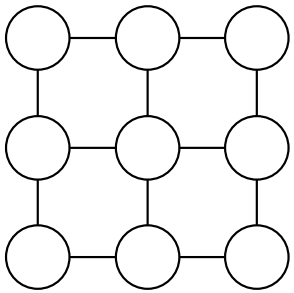
Andrew Rosenberg

March 5, 2010

Today

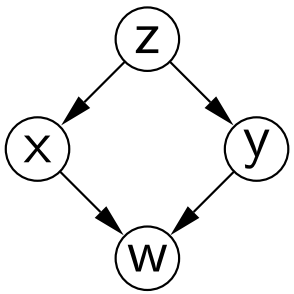
- Graphical Models
  - Probabilities in Undirected Graphs

Undirected Graphs

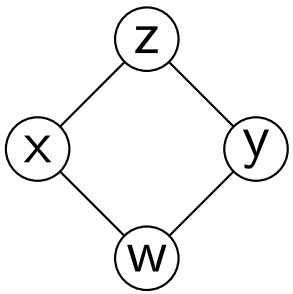


- What if we allow undirected graphs?
- What do they correspond to?
- It's not cause/effect, or trigger/response, rather, general dependence.
- Example: Image pixels, where each pixel is a bernouli.
- Can have a probability over all pixels $p(x_{11}, x_{1M}, x_{M1}, x_{MM})$
- Bright pixels have bright neighbors.
- No parents, just probabilities.
- Grid models are called **Markov Random Fields**.

Undirected Graphs



$x \perp\!\!\!\perp y | \{w\}$



$x \perp\!\!\!\perp y | \{w, z\}$

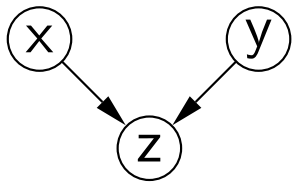
cannot represent

$x \perp\!\!\!\perp y | \{w, z\}$

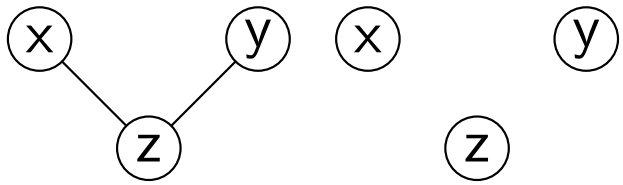
$w \perp\!\!\!\perp z | \{x, y\}$

- Undirected separation is easy.
- To check $x_a \perp\!\!\!\perp x_b | x_c$, check Graph reachability of x_a and x_b without going through nodes in x_c.

Undirected Graphs



$x \perp\!\!\!\perp y$



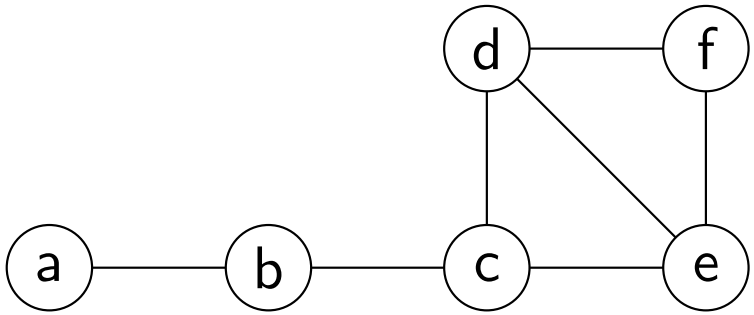
$x \perp\!\!\!\perp y|z$ **OR** $x \perp\!\!\!\perp z$

$x \neg\perp\!\!\!\perp y|z$

- Undirected separation is easy.
- To check $x_a \perp\!\!\!\perp x_b|x_c$, check Graph reachability of x_a and x_b without going through nodes in x_c.

Probabilities in Undirected Graphs

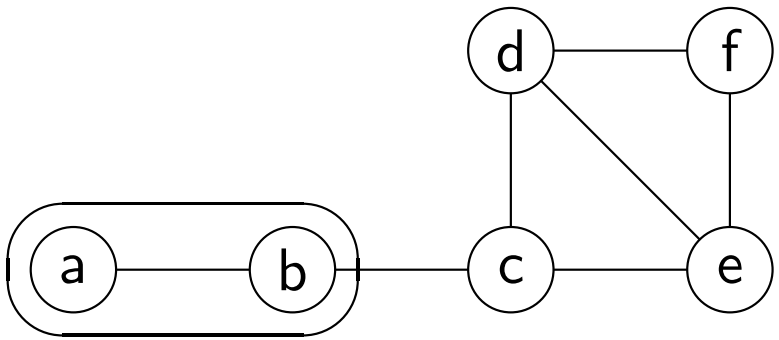
Graph cliques define clusters of dependent variables



- **Clique**: a set of nodes such there is an edge between every pair of members of the set.
- We define probability as a product of functions defined over cliques

Probabilities in Undirected Graphs

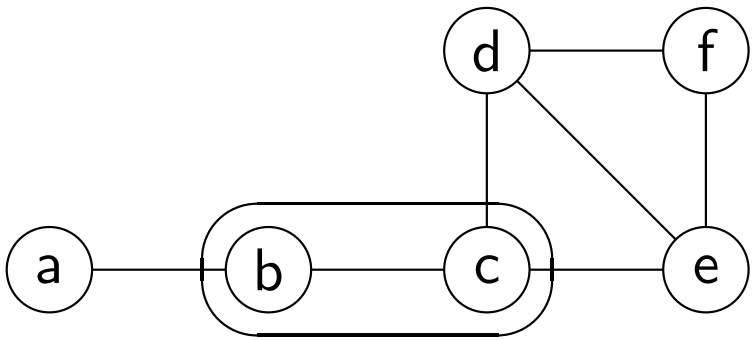
Graph cliques define clusters of dependent variables



- **Clique**: a set of nodes such there is an edge between every pair of members of the set.
- We define probability as a product of functions defined over cliques

Probabilities in Undirected Graphs

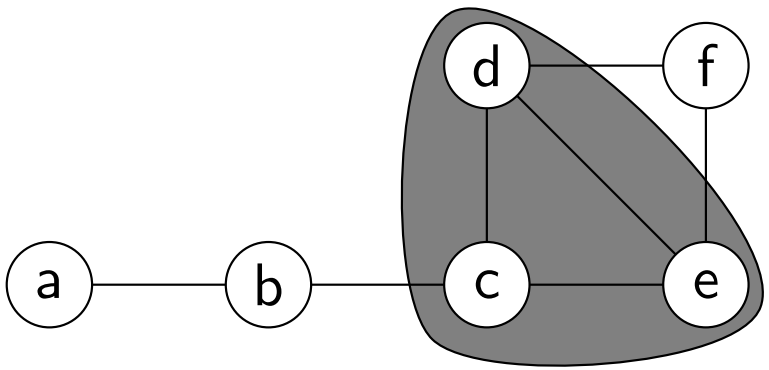
Graph cliques define clusters of dependent variables



- **Clique**: a set of nodes such there is an edge between every pair of members of the set.
- We define probability as a product of functions defined over cliques

Probabilities in Undirected Graphs

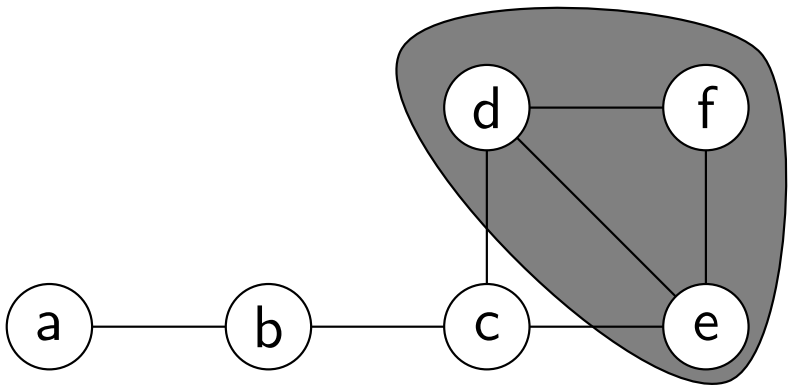
Graph cliques define clusters of dependent variables



- **Clique**: a set of nodes such there is an edge between every pair of members of the set.
- We define probability as a product of functions defined over cliques

Probabilities in Undirected Graphs

Graph cliques define clusters of dependent variables



- **Clique**: a set of nodes such there is an edge between every pair of members of the set.
- We define probability as a product of functions defined over cliques

Representing Probabilities

Potential Functions over cliques

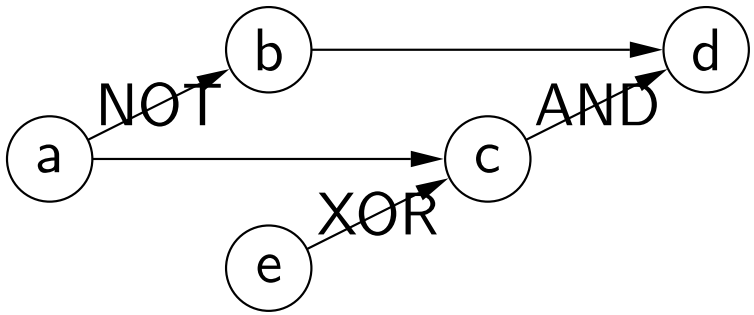
$$p(\mathbf{x}) = p(x_0, \ldots, x_{n-1}) = \frac{1}{Z} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

Normalizing Term guarantees that $p(\mathbf{x})$ sums to 1

$$Z = \sum_{\mathbf{x}} \prod_{c \in C} \psi_c(\mathbf{x}_c)$$

- **Potential Functions** are positive functions over groups of connected variables.
- Use only **maximal cliques**.
  e.g. $\psi(x_1, x_2, x_3)\psi(x_2, x_3) \rightarrow \psi(x_1, x_2, x_3)$

Logical Inference

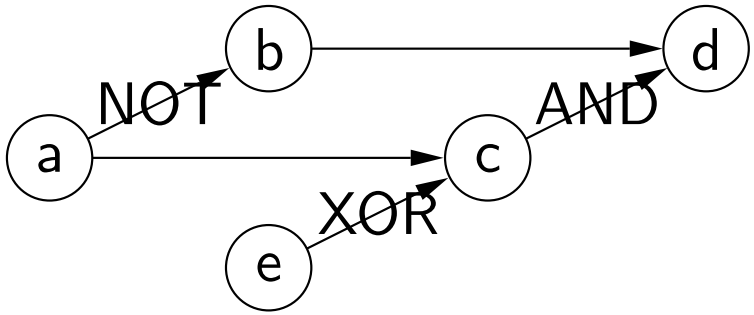


- In Logic Networks, nodes are binary, and edges represent gates
- Gates: AND, OR, XOR, NAND, NOR, NOT, etc.
- **Inference**: given observed variables, predict others.
- **Problems**: Uncertainty, conflicts and inconsistency

Rather than saying a variable is TRUE or FALSE, let's say it is .8 TRUE and .2 FALSE.

Probabilistic Inference

Probabilistic Inference

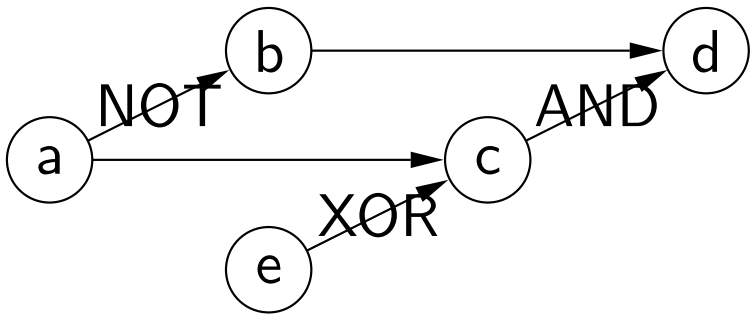


- Replace the logic network with a **Bayesian Network**
- **Probabilistic Inference**: given observed variables, predict marginals over others.

NOT

	b=t	b=f
a=t	0	1
a=f	1	0

Probabilistic Inference



- Replace the logic network with a **Bayesian Network**
- **Probabilistic Inference**: given observed variables, predict marginals over others.

Soft NOT

	b=t	b=f
a=t	.1	.9
a=f	.9	.1

Inference

General Problem

- Given a graph and probabilities, for any subset of variables, find

$$p(\mathbf{x}_e|\mathbf{x}_o) = \frac{p(\mathbf{x}_e, \mathbf{x}_o)}{p(\mathbf{x}_o)}$$

- Compute both marginals and divide.
- But this can be exponential...(Based on the number of parent each node has, or the size of the cliques)

$$\begin{aligned} p(\mathbf{x}_j, \mathbf{x}_k) &= \sum_{\mathbf{x}_0} \sum_{\mathbf{x}_1} \cdots \sum_{\mathbf{x}_{M-1}} \prod_{i=0}^{M-1} p(\mathbf{x}_i|\pi_i) \\ p(\mathbf{x}_j, \mathbf{x}_k) &= \sum_{\mathbf{x}_0} \sum_{\mathbf{x}_1} \cdots \sum_{\mathbf{x}_{M-1}} \prod_{c \in C} \psi(\mathbf{x}_c) \end{aligned}$$

- Have efficient learning and storage in Graphical Models, now inference.

Inefficient Marginals

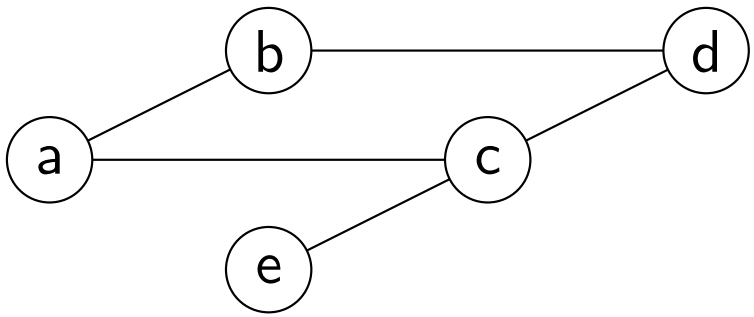
Brute Force.

- Given CPTs and a graph structure we can compute arbitrary marginals by brute force, but it's inefficient.

For Example

$$
\begin{aligned}
p(\mathbf{x}) &= p(\mathbf{x}_0)p(\mathbf{x}_1|\mathbf{x}_0)p(\mathbf{x}_2|\mathbf{x}_0)p(\mathbf{x}_3|\mathbf{x}_1)p(\mathbf{x}_4|\mathbf{x}_2)p(\mathbf{x}_5|\mathbf{x}_2,\mathbf{x}_5) \\
p(\mathbf{x}_0,\mathbf{x}_2) &= p(\mathbf{x}_0)p(\mathbf{x}_2|\mathbf{x}_0) \\
p(\mathbf{x}_0,\mathbf{x}_5) &= \sum_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} p(\mathbf{x}_0)p(\mathbf{x}_1|\mathbf{x}_0)p(\mathbf{x}_2|\mathbf{x}_0)p(\mathbf{x}_3|\mathbf{x}_1)p(\mathbf{x}_4|\mathbf{x}_2)p(\mathbf{x}_5|\mathbf{x}_2,\mathbf{x}_5) \\
p(\mathbf{x}_0|\mathbf{x}_5) &= \frac{\sum_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} p(\mathbf{x})}{\sum_{\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} p(\mathbf{x})} \\
p(\mathbf{x}_0|\mathbf{x}_5 = TRUE) &= \frac{\sum_{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} p(\mathbf{x}_{U\setminus 5}|\mathbf{x}_5 = TRUE)}{\sum_{\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4} p(\mathbf{x}_{U\setminus 5}|\mathbf{x}_5 = TRUE)}
\end{aligned}
$$

Efficient Computation of Marginals



- Pass **messages** (small tables) around the graph.
- The **messages** will be small functions that propagate potentials around an undirected graphical model.
- The inference technique is the **Junction Tree Algorithm**

Junction Tree Algorithm

- Efficient Message Passing on Undirected Graphs.
- For Directed Graphs, first convert to an Undirected Graph (**Moralization).**

Junction Tree Algorithm

- Moralization
- Introduce Evidence
- Triangulate
- Construct Junction Tree
- Propagate Probabilities

Bye

- Next
  - Junction Tree Algorithm