Bayesian Belief Networks

Belief Networks

Let $X_1, \ldots, X_n$ be discrete random variables.

A *belief network (or Bayesian network)* for $X_1, \ldots, X_n$ is a graph with m nodes such that

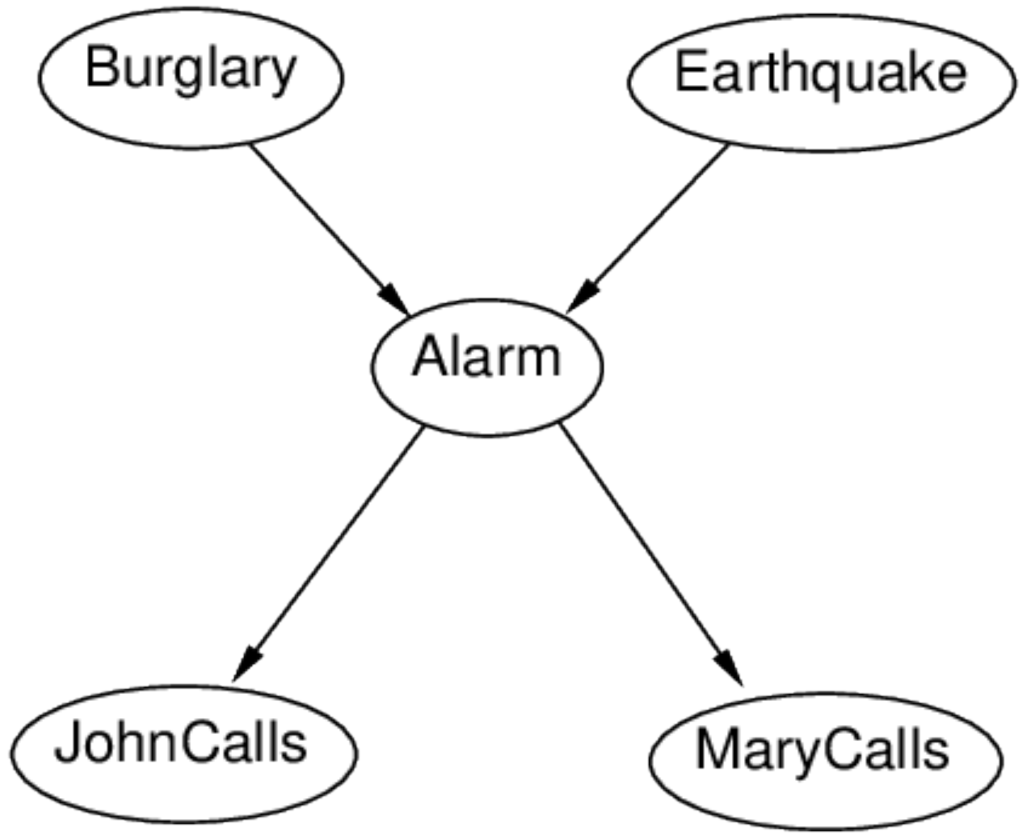
- there is a node for each X_i
- all the edges between two nodes are directed
- there are no cycles
- each node has a conditional probability table (CPT), given in terms of its parents

The intuitive meaning of an edge from a node X_i to a node X_j is that X_i has a direct influence on X_j

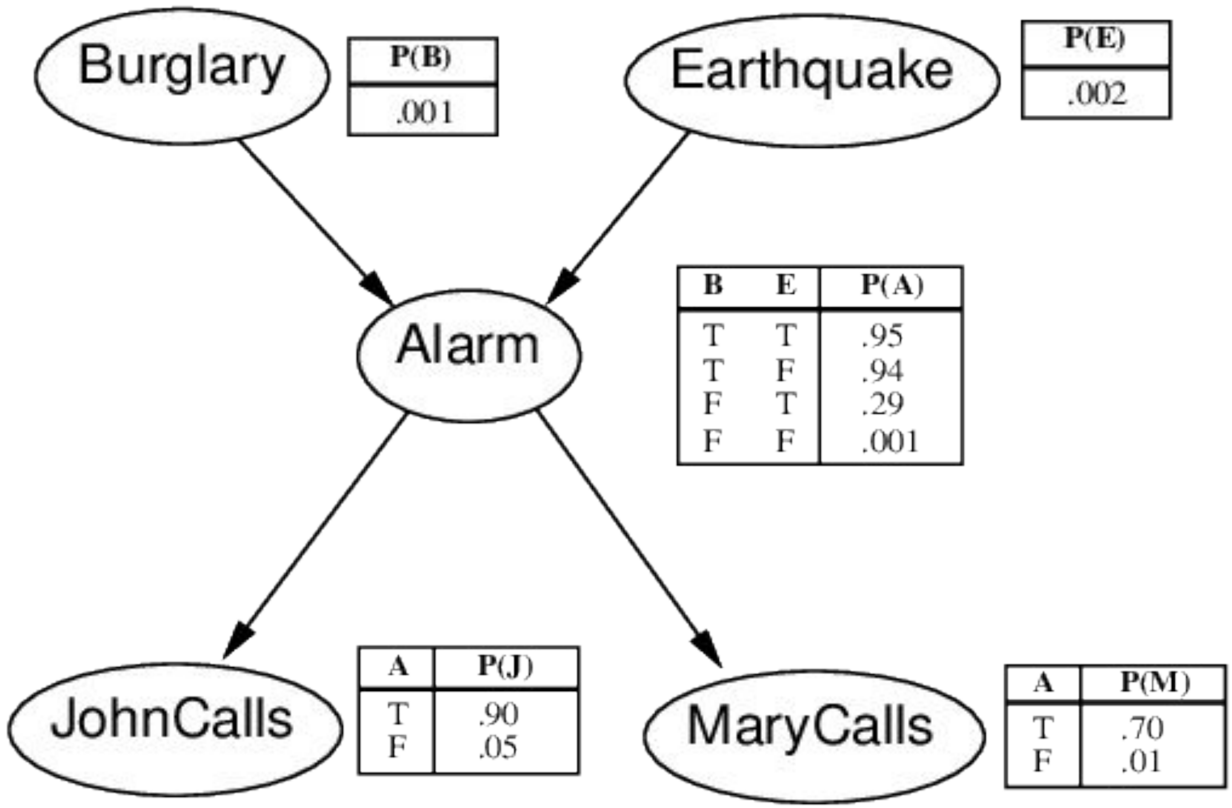
Example

- You have a new burglar alarm installed at home.
- It is fairly reliable at detecting a burglary, but is occasionally set off by minor earthquakes.
- You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm.
- John nearly always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too.
- Mary, on the other hand, likes rather loud music and often misses the alarm altogether.
- Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

A Belief Network



A Belief Network with CPTs



P(B)
.001

P(E)
.002

B	E	P(A)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J)
T	.90
F	.05

A	P(M)
T	.70
F	.01

Note: The tables only show $P(X = true)$ here because $P(X = false) = 1 - P(X = true)$

The Semantics of Belief Networks

There are two equivalent ways to interpret a belief network for the variables $X_1, \ldots, X_n$:

1. The network is a representation of the JPD $\mathbf{P}(X_1, \ldots, X_n)$
2. The network is a collection of conditional independence statements about $X_1, \ldots, X_n$

Interpretation 1 is helpful when constructing belief networks

Interpretation 2 is helpful in designing inference procedures based on them

Belief Network as JPDs

The whole JPD $\mathbf{P}(X_1, \ldots, X_n)$ can be computed from a belief network for $X_1, \ldots, X_n$ and its CPTs

For each tuple $\langle x_1, \ldots, x_n \rangle$ of possible values for $\langle X_1, \ldots, X_n \rangle$,

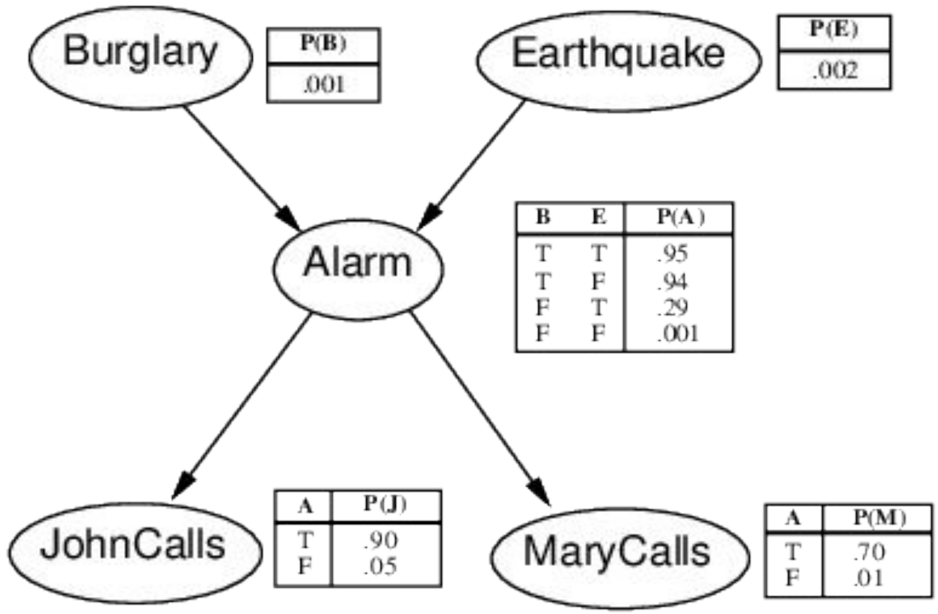
$$P(X_1 = x_1 \wedge \cdots \wedge X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$

where

$$Parents(X_i) = \{X_j = x_j \mid 1 \leq j \leq n \text{ and } X_j \text{ is a parent of } X_i\}$$

Belief Network as JPDs

$$P(X_1 = x_1 \wedge \cdots \wedge X_n = x_n) \;=\; \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$



$$\begin{aligned} & P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) \\ &= P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b \wedge \neg e)\, P(\neg b)\, P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 = 0.00062 \end{aligned}$$

Belief Networks and Cond. Independence

Let $\{X_1, X_2, \ldots, X_n\}$ be any set of nodes in the network such that

- all the parents of X_1 are in $\{X_2, \ldots, X_n\}$
- no node in $\{X_2, \ldots, X_n\}$ is a descendant of X_1

Let $\langle x_1, \ldots, x_n \rangle$ be a value assignment for $\langle X_1, \ldots, X_n \rangle$

From the equation

$$P(X_1 = x_1 \wedge \cdots \wedge X_n = x_n) \;=\; \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$

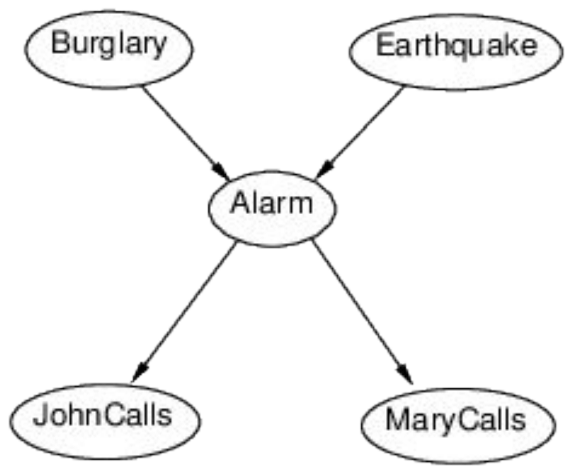
we can show that

$$P(X_1 = x_1 \mid X_2 = x_2 \wedge \cdots \wedge X_n = x_n) \;=\; P(X_1 = x_1 \mid Parents(X_1))$$

Belief Networks and Cond. Independence

$$P(X_1 = x_1 \mid X_2 = x_2 \wedge \cdots \wedge X_n = x_n) \;=\; P(X_1 = x_1 \mid Parents(X_1))$$

Examples:



$$P(b \mid e) = P(b)$$
$$P(j \mid m \wedge a) = P(j \mid a)$$
$$P(j \mid a \wedge e) = P(j \mid a)$$
$$P(j \mid a \wedge b \wedge e) = P(j \mid a)$$
$$P(j \mid m \wedge a \wedge b \wedge e) = P(j \mid a)$$

Exercise: Find all the conditional independences

Constructing Belief Networks

General Procedure

1. Identify a set of random variables $\{X_i\}_i$ that describe the domain
2. Choose an ordering $X_1, \ldots, X_n$ of the variables
3. Start with an empty network
4. For $i = 1 \ldots n$:
   (a) add X_i to the network
   (b) select as parents of X_i nodes from $X_1, \ldots, X_{i-1}$ such that $\mathbf{P}(X_i \mid Parents(X_i)) = \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1})$
   (c) fill in the CPT for X_i

Constructing Belief Networks

General Procedure

1. Identify a set of random variables $\{X_i\}_i$ that describe the domain
2. Choose an ordering $X_1, \ldots, X_n$ of the variables
3. Start with an empty network
4. For $i = 1 \ldots n$:
   (a) add X_i to the network
   (b) select as parents of X_i nodes from $X_1, \ldots, X_{i-1}$ such that $\mathbf{P}(X_i \mid Parents(X_i)) = \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1})$
   (c) fill in the CPT for X_i

This choice of parents guarantees the network semantics:

$$\begin{aligned} \mathbf{P}(X_1, \ldots, X_n) &= \prod_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \ldots, X_{i-1}) && \text{(chain rule)} \\ &= \prod_{i=1}^{n} \mathbf{P}(X_i \mid Parents(X_i)) && \text{(by construction)} \end{aligned}$$

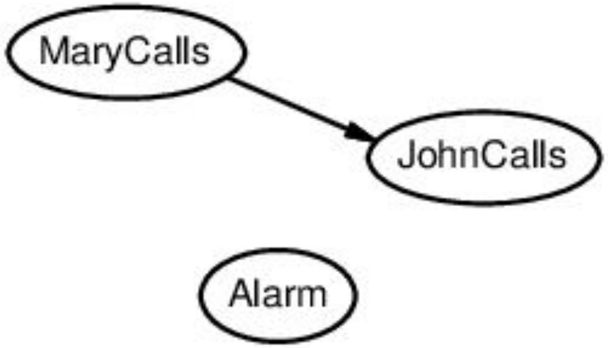
Example

Suppose we choose the ordering $MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$

$P(j \mid m) = P(j)$?

Example

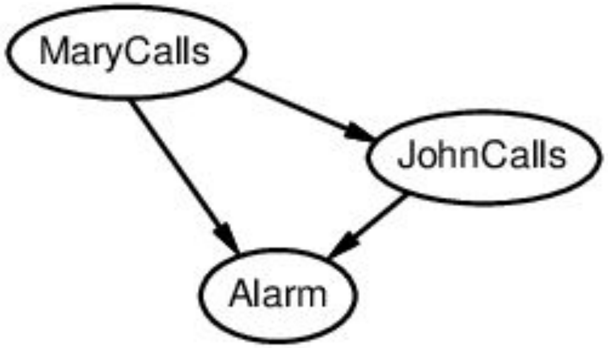
Suppose we choose the ordering $MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$



$P(j \mid m) = P(j)$? No
$P(a \mid j, m) = P(a \mid j)$?
$P(a \mid j, m) = P(a)$?

Example

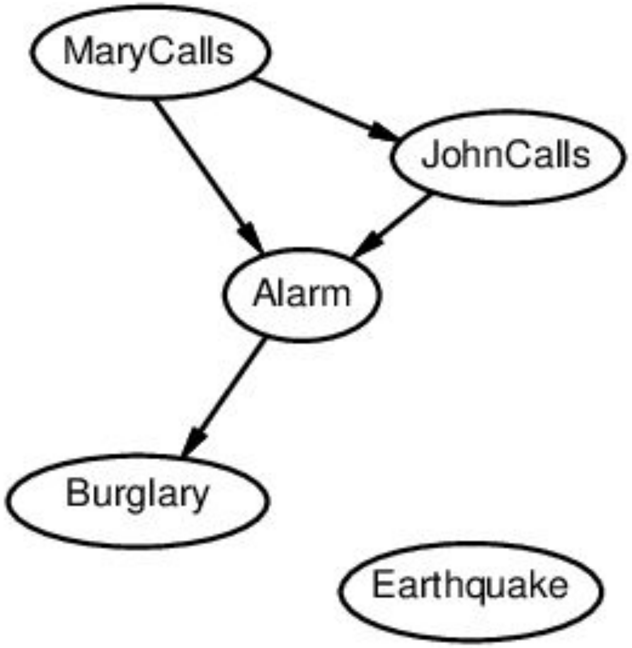
Suppose we choose the ordering $MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$



$P(j \mid m) = P(j)$? No
$P(a \mid j, m) = P(a \mid j)$? No
$P(a \mid j, m) = P(a)$? No
$P(b \mid a, j, m) = P(b \mid a)$?
$P(b \mid a, j, m) = P(b)$?

Example

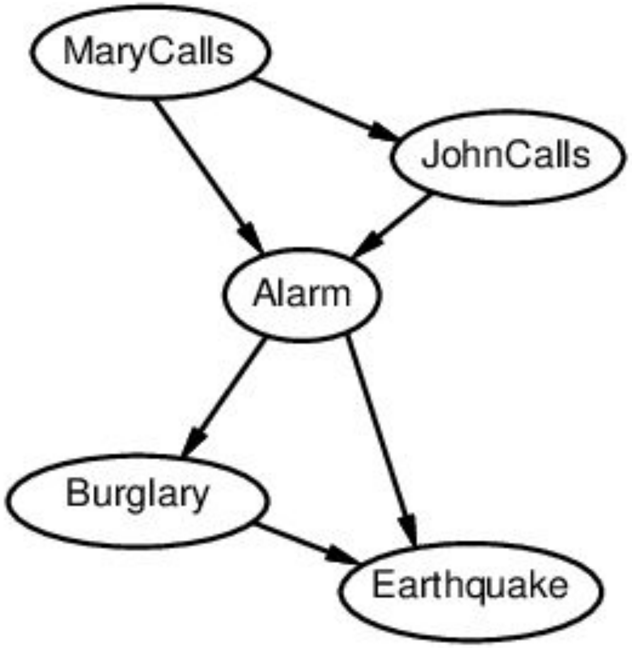
Suppose we choose the ordering $MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$



$P(j \mid m) = P(j)$? No
$P(a \mid j, m) = P(a \mid j)$? No
$P(a \mid j, m) = P(a)$? No
$P(b \mid a, j, m) = P(b \mid a)$? Yes
$P(b \mid a, j, m) = P(b)$? No
$P(e \mid b, a, j, m) = P(e \mid a)$?
$P(e \mid b, a, j, m) = P(e \mid a, b)$?

Example

Suppose we choose the ordering $MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$



$P(j \mid m) = P(j)$? No
$P(a \mid j, m) = P(a \mid j)$? No
$P(a \mid j, m) = P(a)$? No
$P(b \mid a, j, m) = P(b \mid a)$? Yes
$P(b \mid a, j, m) = P(b)$? No
$P(e \mid b, a, j, m) = P(e \mid a)$? No
$P(e \mid b, a, j, m) = P(e \mid a, b)$? Yes

Example

Deciding conditional independence is hard in non-causal directions

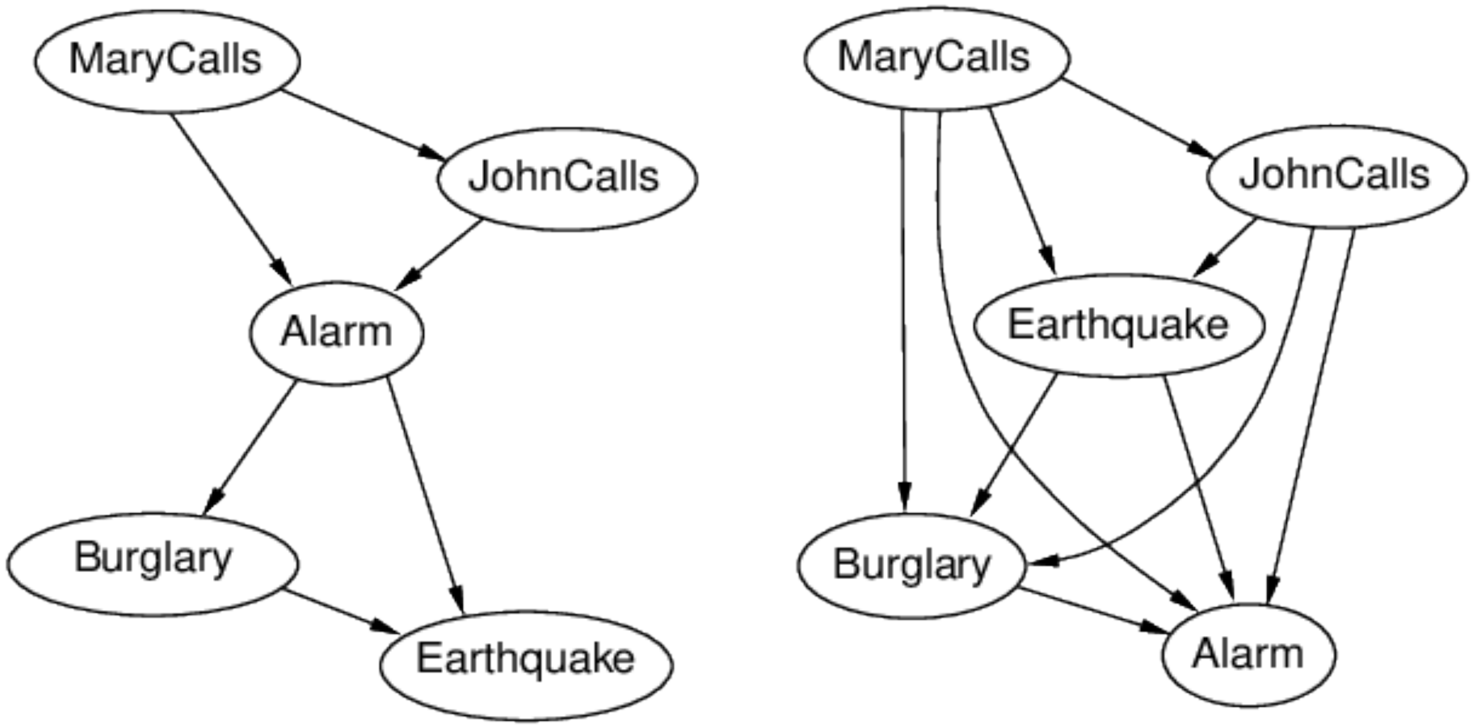
Causal models and conditional independence seem hardwired for humans!

Assessing conditional probabilities is hard in non-causal directions

Network is less compact: $1 + 2 + 4 + 2 + 4 = 13$ probabilities needed

Ordering the Variables

The order in which add the variables to the network is important

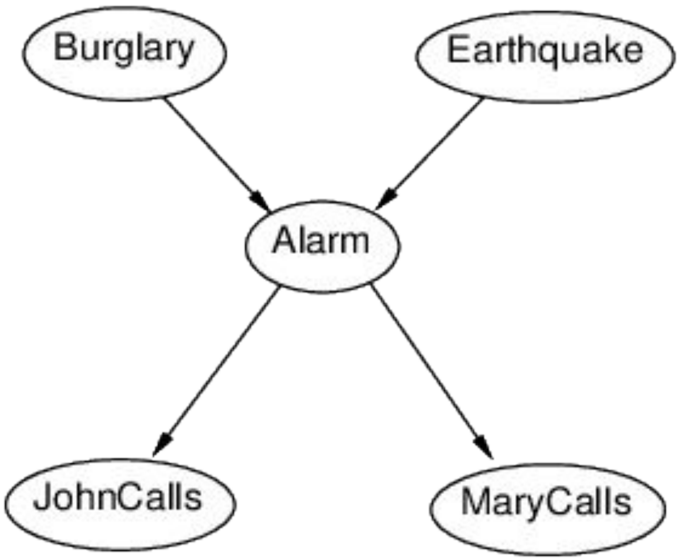


"Wrong" orderings produces more complex networks

Ordering the Variables Right

A general, effective heuristic for constructing simpler belief networks is to exploit causal links between random variables whenever possible

This is done by adding variables to the network so that causes get added before effects



Inference in Belief Networks

Main task of a belief network: Compute the conditional probability of a set of query variables, given exact values for some evidence variables

$$P(Query \mid Evidence)$$

Belief networks are flexible enough so that any node can serve as either a query or an evidence variable

In general, to decide what actions to take, an agent

1. first gets values for some variables from its percepts, or from its own reasoning
2. then asks the network about the possible values of the other variables

Probabilistic Inference with BNs

Belief networks are a very flexible tool for probabilistic inference because they allow several kinds of inference:

Diagnostic inference (from effects to causes)

E.g. $P(Burglary \mid JohnCalls)$

Causal inference (from causes to effects)

E.g. $P(JohnCalls \mid Burglary)$

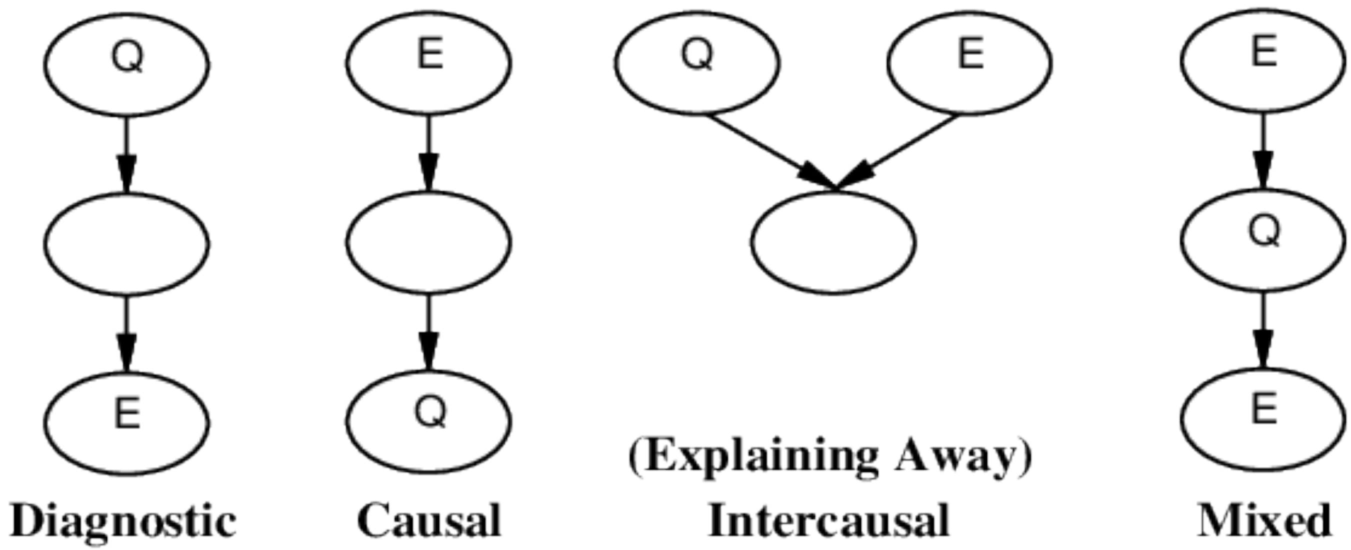
Intercausal inference (between causes of a common effect)

$P(Burglary \mid Alarm \land Earthquake)$

Mixed inference (combination of the above)

$P(Alarm \mid JohnCalls \land \neg Earthquake)$

Types of Inference in Belief Networks



Q = query variable

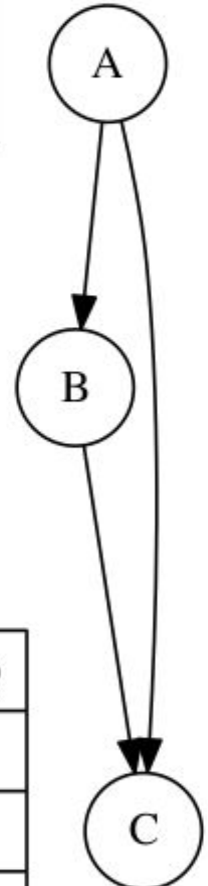
E = evidence variable

Example

P(A=T)
0.6

A	P(B=T|A)
T	0.3333
F	0.5

A	B	P(C=T|A,B)
T	T	0.25
T	F	0.125
F	T	0.25
F	F	0.25

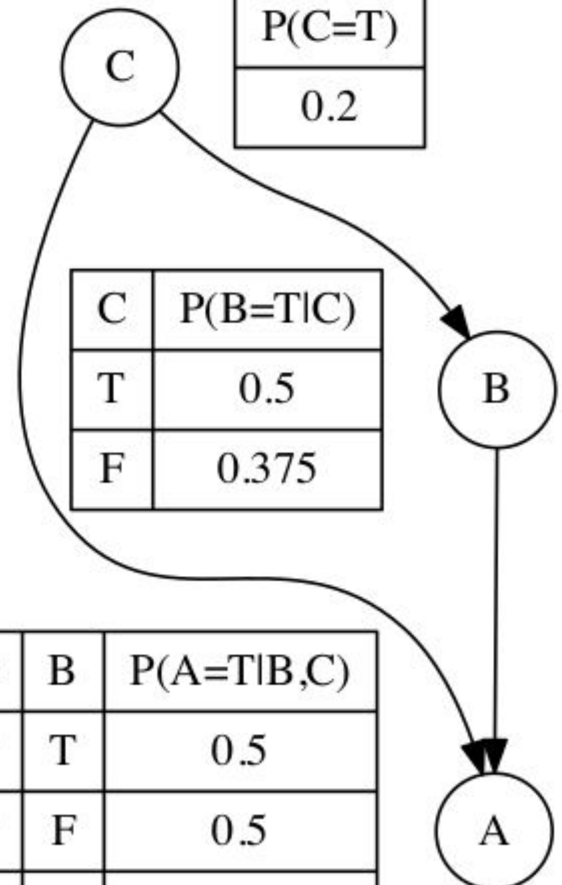


(a)

P(C=T)
0.2

C	P(B=T|C)
T	0.5
F	0.375

C	B	P(A=T|B,C)
T	T	0.5
T	F	0.5
F	T	0.5
F	F	0.7



(b)

Two different Bayesian networks, each defining same full joint probability distribution.

Example

We can illustrate that these two networks encode the same joint probability distribution by using each network to compute $P(\neg a, b, c)$

Using network (a) we get:

$$
\begin{aligned}
P(\neg a, b, c) &= P(c|\neg a, b) \times P(b|\neg a) \times P(\neg a) \\
&= 0.25 \times 0.5 \times 0.4 = 0.05
\end{aligned}
$$

Using network (b) we get:

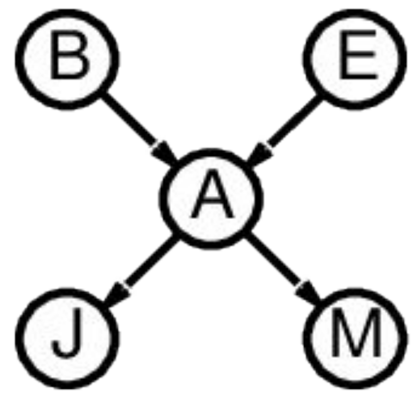
$$
\begin{aligned}
P(\neg a, b, c) &= P(\neg a|c, b) \times P(b|c) \times P(c) \\
&= 0.5 \times 0.5 \times 0.2 = 0.05
\end{aligned}
$$

Inference by enumeration

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

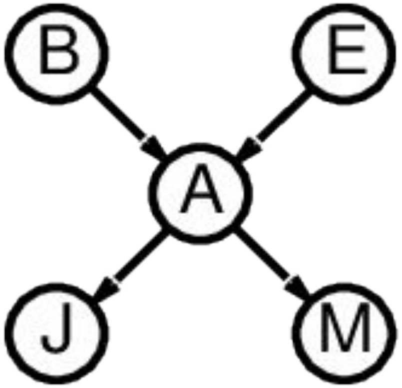


$\mathbf{P}(B \mid j, m)$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

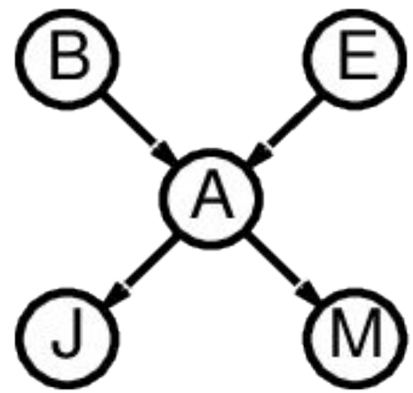


$$\begin{aligned}\mathbf{P}(B \mid j, m)& \\ &= \mathbf{P}(B, j, m)/P(j, m) \\ &= \alpha\mathbf{P}(B, j, m)\end{aligned}$$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

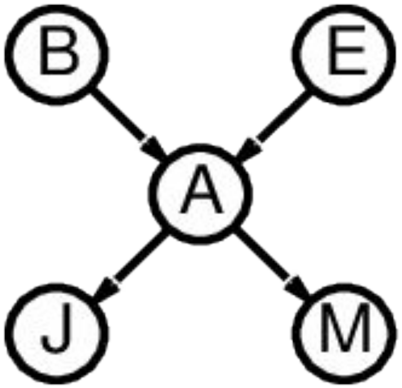


$$\mathbf{P}(B \mid j, m)$$
$$= \mathbf{P}(B, j, m)/P(j, m)$$
$$= \alpha \mathbf{P}(B, j, m)$$
$$= \alpha \sum_e \sum_a \mathbf{P}(B, e, a, j, m)$$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



$$
\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \mathbf{P}(B, j, m) / P(j, m) \\
&\quad = \alpha \mathbf{P}(B, j, m) \\
&\quad = \alpha \sum_e \sum_a \mathbf{P}(B, e, a, j, m)
\end{aligned}
$$

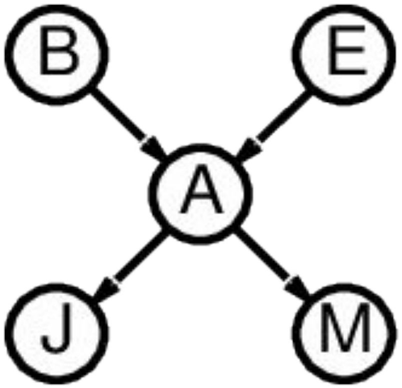
Rewrite full joint entries using product of CPT entries:

$$
\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \alpha \sum_e \sum_a
\end{aligned}
$$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



$$\begin{aligned}&\mathbf{P}(B \mid j, m)\\&\quad = \mathbf{P}(B, j, m)/P(j, m)\\&\quad = \alpha\mathbf{P}(B, j, m)\\&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B, e, a, j, m)\end{aligned}$$

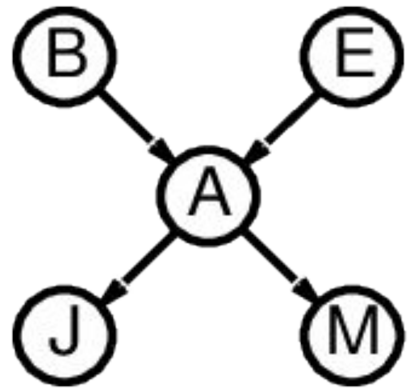
Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}&\mathbf{P}(B \mid j, m)\\&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a \mid B, e)P(j \mid a)P(m \mid a)\end{aligned}$$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



$$\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \mathbf{P}(B, j, m)/P(j, m) \\
&\quad = \alpha \mathbf{P}(B, j, m) \\
&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B, e, a, j, m)
\end{aligned}$$

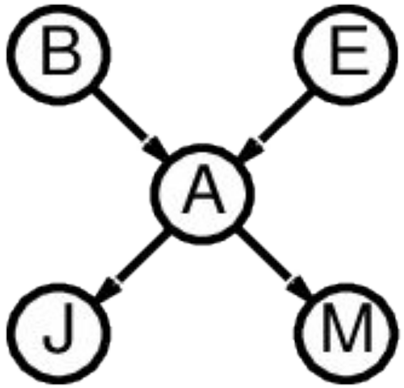
Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B) P(e) \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a) \\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum \qquad\qquad \sum
\end{aligned}$$

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



$$
\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \mathbf{P}(B, j, m)/P(j, m) \\
&\quad = \alpha \mathbf{P}(B, j, m) \\
&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B, e, a, j, m)
\end{aligned}
$$

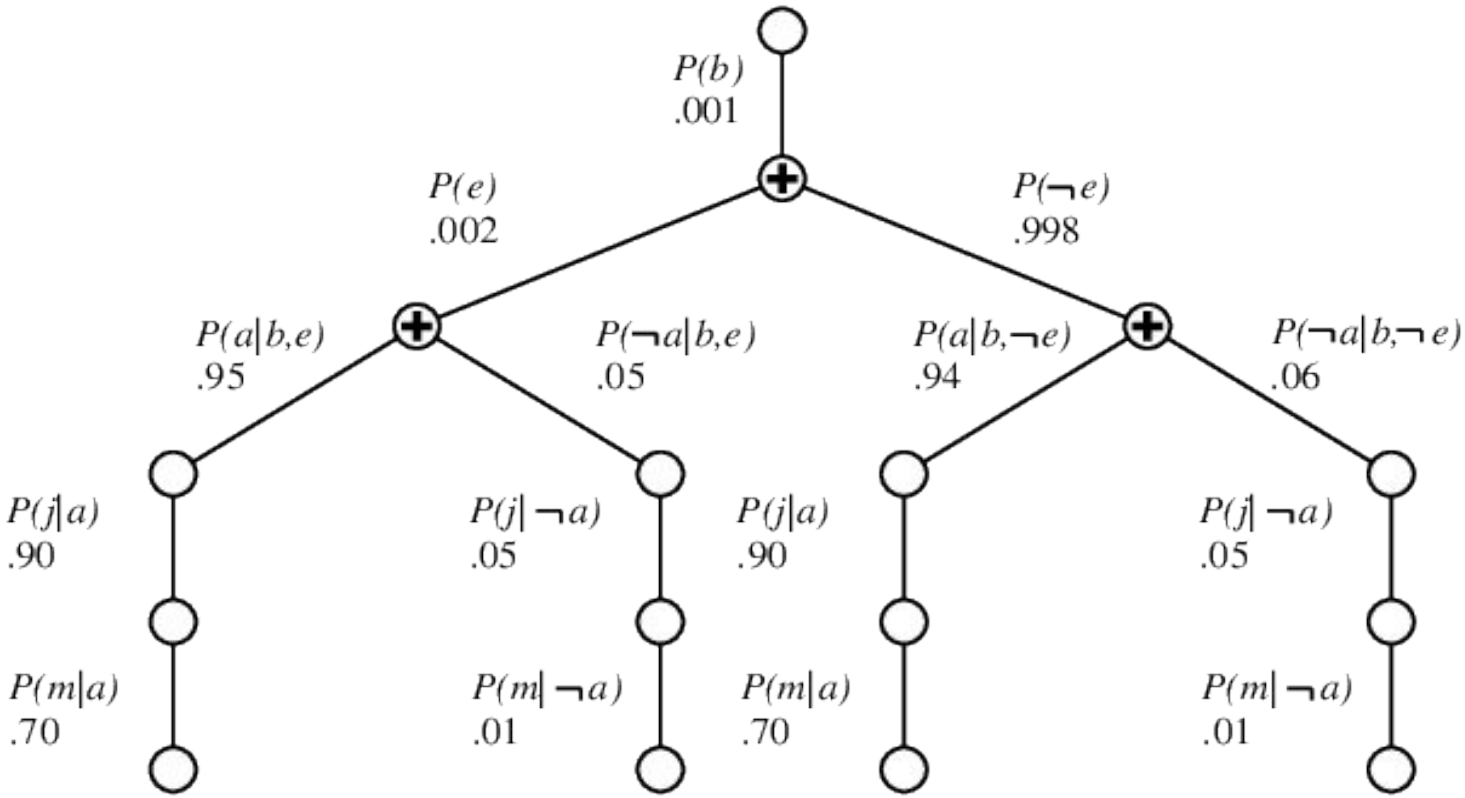
Rewrite full joint entries using product of CPT entries:

$$
\begin{aligned}
&\mathbf{P}(B \mid j, m) \\
&\quad = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B) P(e) \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a) \\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum_e P(e) \sum_a \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a)
\end{aligned}
$$

Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

Evaluation tree

$\alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a \mid B, e) P(j \mid a) P(m \mid a)$



Enumeration is inefficient: repeated computation. E.g., computes $P(j \mid a)P(m \mid a)$ for each value of e

Inference by variable elimination

14.4.2

An illustrative example

Suppose for simplicity that we are given a chain Bayesian network, i.e., a probability of the form

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$$

We are interested in computing the marginal probability $p(x_n)$.

The naive way of calculating this is to sum the probability over all k^{n-1} assignments to $x_1, \ldots, x_{n-1}$:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1, \ldots, x_n)$$

However, we can do much better by leveraging the factorization of our probability distribution?

An illustrative example

$$
\begin{aligned}
p(x_n) &= \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}) \\
&= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1)\, p(x_1)
\end{aligned}
$$

- We sum the inner terms first, starting from x_1 and ending with x_{n-1}.
- Concretely, we start by computing an intermediary factor $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1)\, p(x_1)$ by summing out x_1.
- This takes $O\left(k^2\right)$ time because we must sum over x_1 for each assignment to x_2.
- The resulting factor $\tau(x_2)$ can be thought of as a table of k values (though not necessarily probabilities), with one entry for each assignment to x_2 (just as factor $p(x_1)$ can be represented as a table).

An illustrative example

We may then rewrite the marginal probability using τ as

$$p\left(x_n\right)=\sum_{x_{n-1}} p\left(x_n \mid x_{n-1}\right) \sum_{x_{n-2}} p\left(x_{n-1} \mid x_{n-2}\right) \cdots \sum_{x_2} p\left(x_3 \mid x_2\right) \tau\left(x_2\right)$$

- Note that this has the same form as the initial expression, except that we are summing over one fewer variable (dynamic programming).
- We may therefore compute another factor $\tau\left(x_3\right)=\sum_{x_2} p\left(x_3 \mid x_2\right) \tau\left(x_2\right)$, and repeat the process until we are only left with x_n.
- Since each step takes $O\left(k^2\right)$ time, and we perform $O(n)$ steps, inference now takes $O\left(nk^2\right)$ time, which is much better than our naive $O\left(k^n\right)$ solution.
- Also, at each time, we are eliminating a variable, which gives the algorithm its name.

Factors and Factor Graph

$$f(x) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

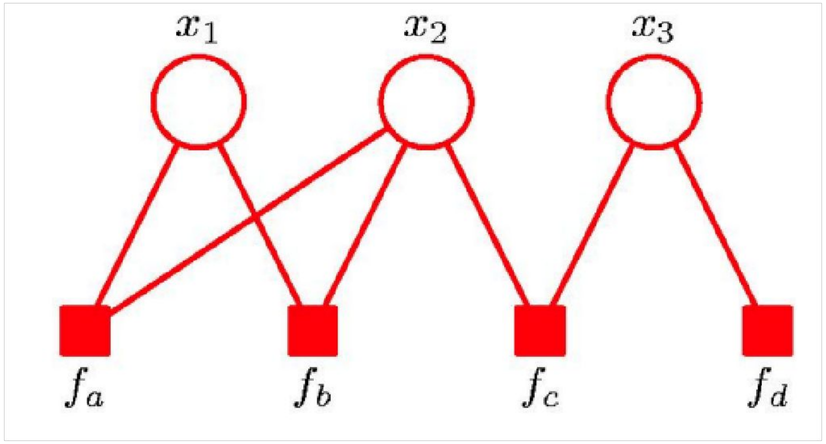


Figure 1: Two factors f_a and f_b of the same variables

They are bipartite since

- Two types of nodes
- All links go between nodes of opposite type

Factors

We assume that we are given a graphical model as a product of factors

$$p\left(x_1, \ldots, x_n\right) = \prod_{c \in C} \phi_c\left(x_c\right)$$

- We can view a factor as a multi-dimensional table assigning a value to each assignment of a set of variables x_c.
- In a Bayesian network, the factors correspond to conditional probability distributions.
- In a Markov Random Field, the factors encode an unnormalized distribution; to compute marginals.