Probabilistic Reasoning

- 2.1 Representing Knowledge in an Uncertain Domain
- 2.2 The Semantics of Bayesian Networks
  - Representing the full joint distribution
  - A method for constructing Bayesian networks
  - Compactness and node ordering
- 2.3 Efficient Representation of Conditional Distributions
- 2.4 Exact Inference in Bayesian Networks
  - Inference by enumeration
  - The variable elimination algorithm
  - The complexity of exact inference
  - Clustering algorithms
- 2.5 Approximate Inference in Bayesian Networks
  - Direct sampling methods
  - Rejection sampling in Bayesian networks
  - Likelihood weighting
  - Inference by Markov chain simulation

2.1 Representing Knowledge in an Uncertain Domain

- full joint probability distribution can answer any question about the domain, but can become intractably large
- independence and conditional independence relationships among variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution.
- Bayesian networks (BN): systematic way to represent relationships explicitly
- BN represent dependencies among variables and to give a concise specification of any full joint probability distribution.

Example: Bayesian Network (BN)

- Bayesian network: directed graph in which each node is annotated with quantitative probability information

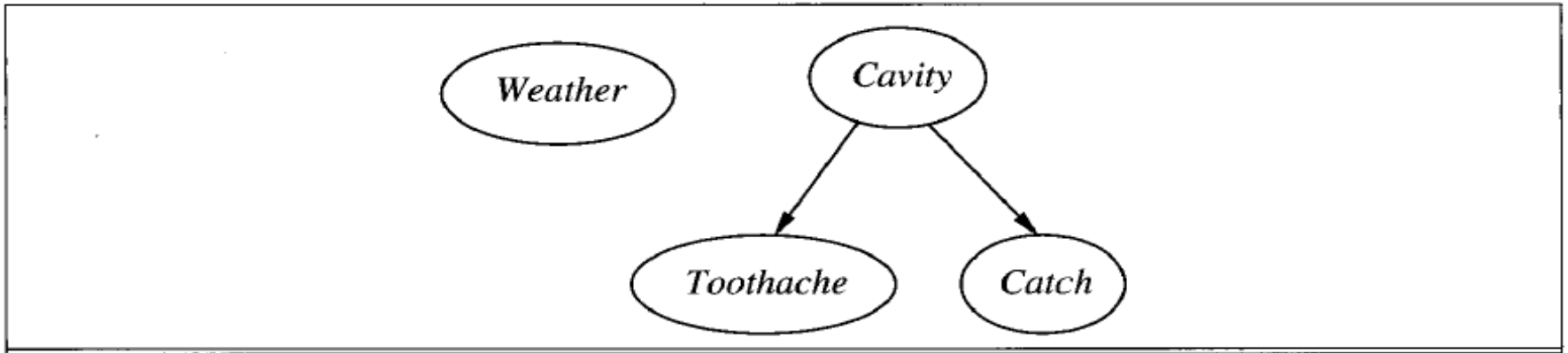


Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given *Cavity*.

- *Weather* is independent of the other variables; *Toothache* and *Catch* are conditionally independent, given *Cavity*.
- formally, conditional independence of *Toothache* and *Catch* given *Cavity* is indicated by the absence of a link between *Toothache* and *Catch*.

Definition: Bayesian Network (BN)

- Bayesian network: directed graph in which each node is annotated with quantitative probability information
- Full specification:

1. A set of random variables (discrete or continuous) are the nodes of the network.
2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.
3. Each node X_i has a conditional probability distribution

$$P(X_i \mid Parents(X_i))$$

that quantifies the effect of the parents on the node.

4. The graph has no directed cycles (and hence is a directed, acyclic graph, or DAG).

Definition: Bayesian Network (BN)

- Topology of the network - the set of nodes and links - specifies the conditional independence relationships that hold in the domain
- Intuitive meaning of an arrow in a properly constructed network is usually that X has a direct influence on Y.
- Once the topology of the Bayesian network is laid out, we have to specify a conditional probability distribution for each variable, given its parents.

Combination of the topology and the conditional distributions suffices to specify (implicitly) the full joint distribution for all the variables.

Another Example: burglar alarm

- new burglar alarm, fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes
- two neighbors, John and Mary, who have promised to call you at work when they hear the alarm
- John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too
- Mary, likes rather loud music and sometimes misses the alarm altogether.

Another Example: burglar alarm

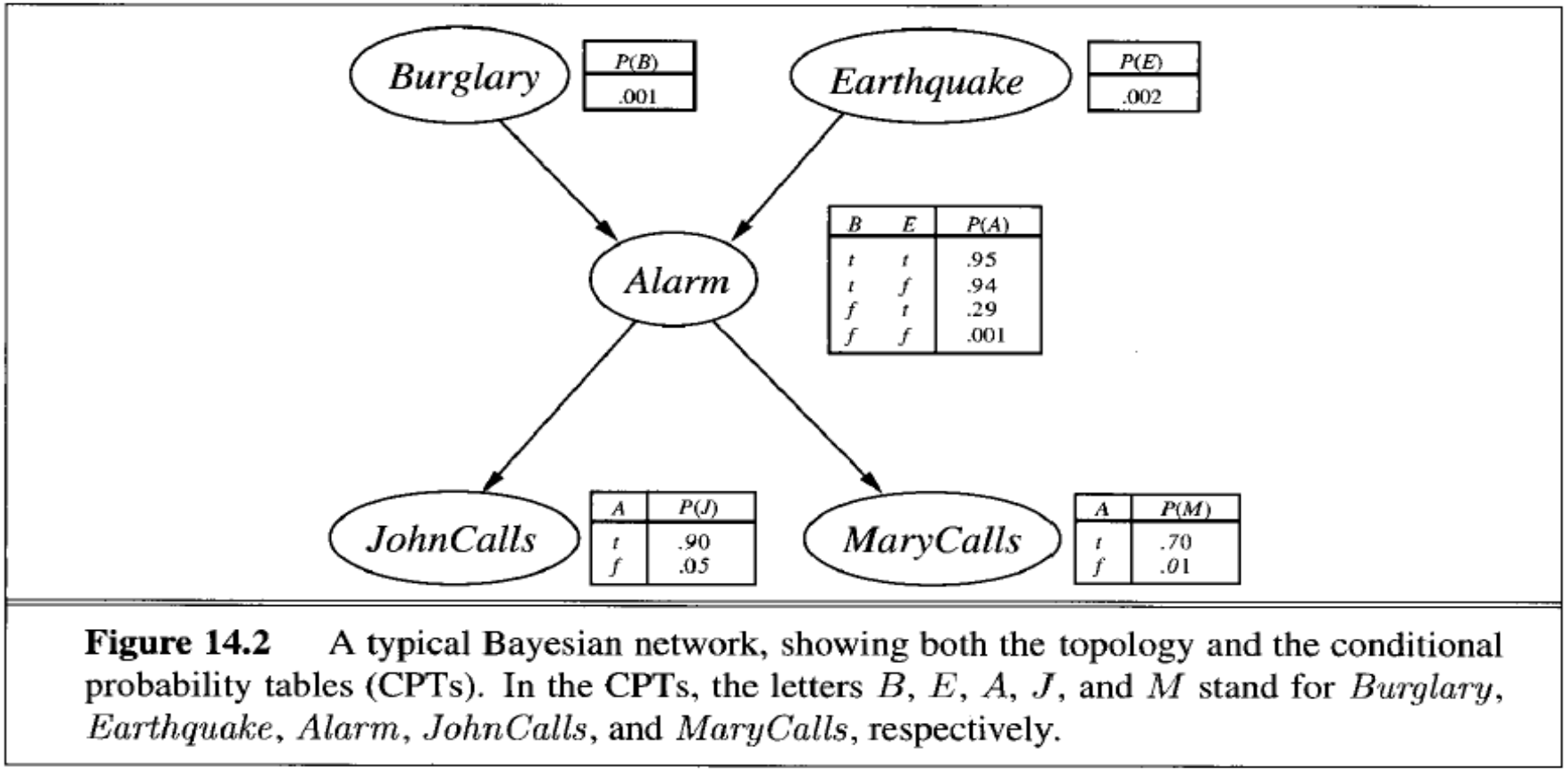


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

<u>Task</u>

Given evidence of who has or has not called, we would like to estimate the probability of a burglary.

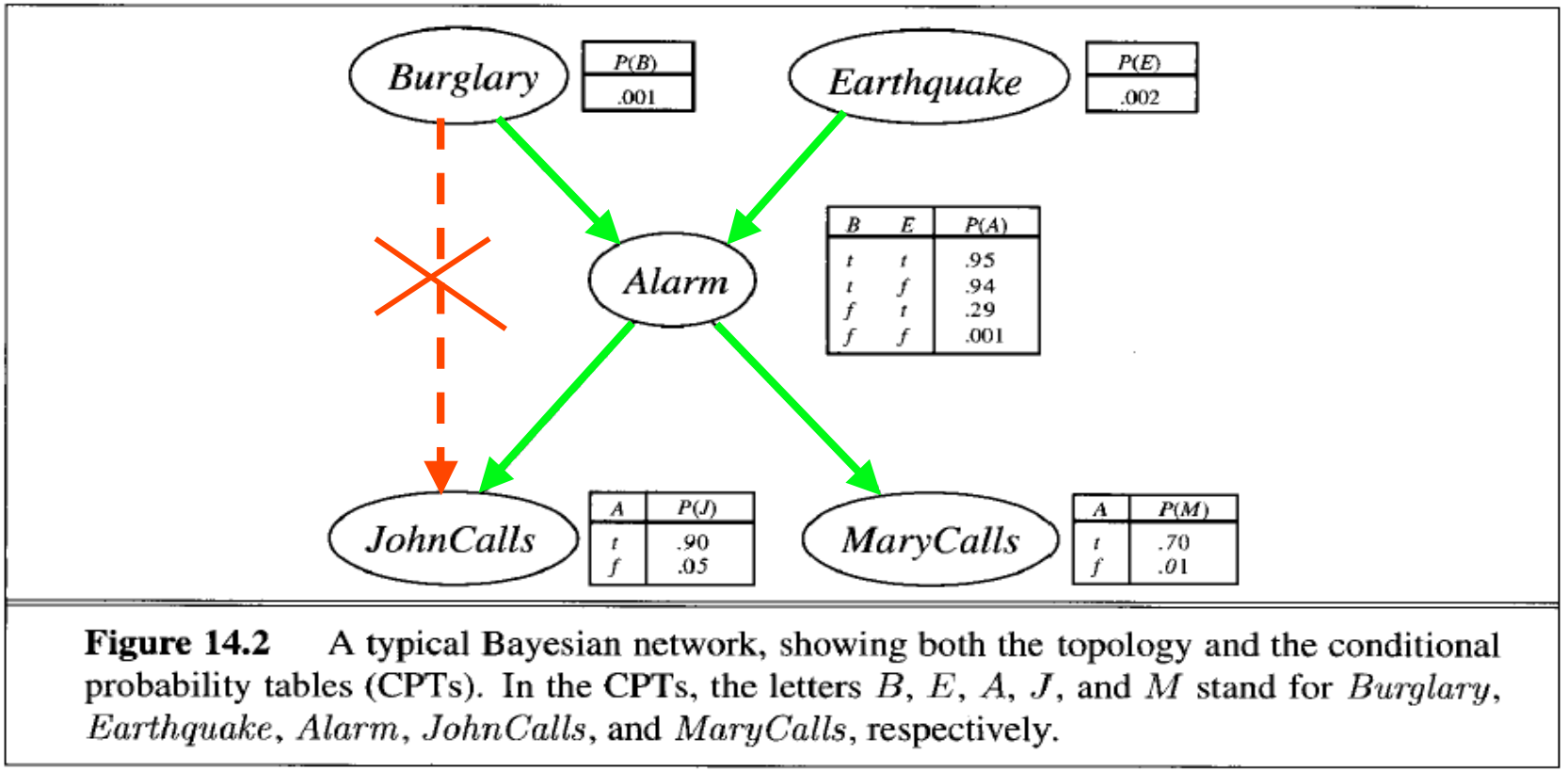


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

- topology shows that burglary and earthquakes directly affect the probability of the alarm's going off
- whether John and Mary call depends only on the alarm.
- network thus represents our assumptions that they do not perceive any burglaries directly
- they do not notice the minor earthquakes, and they do not confer before calling.

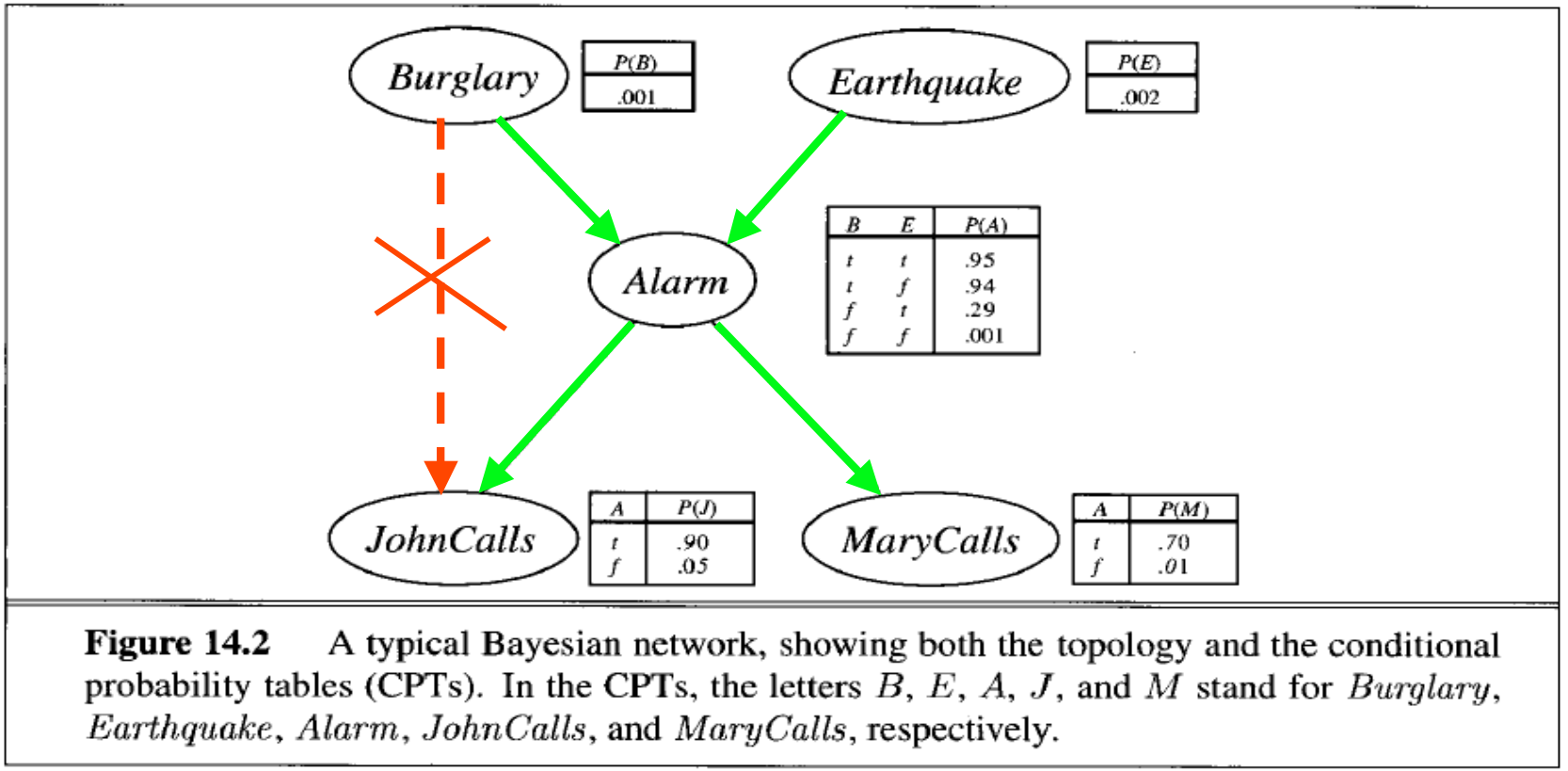


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

- Notice: network does not have nodes corresponding to Mary's currently listening to loud music or to the telephone ringing and confusing John
- These factors are summarized in the uncertainty associated with the links from *Alarm* to *JohnCalls* and *MaryCalls*.
- Probabilities actually summarize a potentially infinite set of circumstances in which the alarm might fail to go off.

Conditional Probability Table

- in figure each distribution is shown as a conditional probability table, or CPT.
- each row in a CPT contains the conditional probability of each node value for a conditioning case
- conditioning case is just a possible combination of values for the parent nodes (miniature atomic event)
- each row must sum to 1, because the entries represent an exhaustive set of cases for the variable. *)
- in general, a table for a Boolean variable with k Boolean parents contains 2^k independently specifiable probabilities.
- node with no parents has only one row, representing the prior probabilities of each possible value of the variable.

---

*) for Boolean variables, once you know that the probability of a true value is p, the probability of false must be 1 - p $\rightarrow$ second number omitted

2.2 The Semantics of Bayesian Networks

Two ways in which one can understand the semantics of Bayesian networks:

- see the network as a representation of the joint probability distribution.
- view it as an encoding of a collection of conditional independence statements.
- two views are equivalent, but the first turns out to be helpful in understanding how to construct networks, whereas the second is helpful in designing inference procedures.

Representing the full joint distribution

- Bayesian network provides a complete description of the domain
- every entry in the full joint probability distribution can be calculated from the information in the network**!!!**
- generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, such as $P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$ abbreviated as $P(x_1, \ldots, x_n)$ as an abbreviation for this.
- value of this entry is given by the formula

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)) \qquad (2.1)$$

  where $parents(X_i)$ denotes the specific values of the variables in $Parents(X_i)$

- each entry in joint distribution is represented by the product of the appropriate elements of the conditional probability tables (CPTs) in the Bayesian network
- CPTs therefore provide a decomposed representation of the joint distribution.

Example

probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call

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

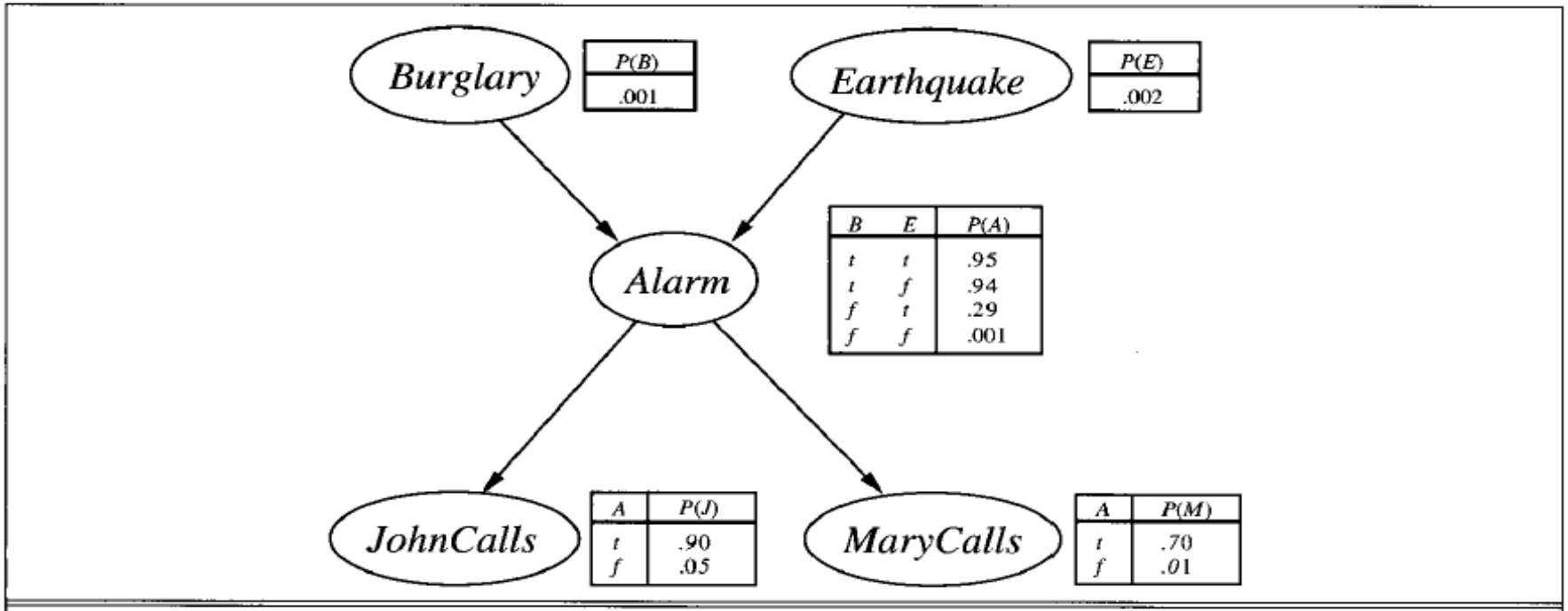


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Example

probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call

$$
\begin{aligned}
P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) &= \\
&= P(j \mid a) P(m \mid a) P(a \mid \neg b \wedge \neg e) P(\neg b) P(\neg e) \\
&= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062
\end{aligned}
$$

A method for constructing Bayesian networks

- How to construct a Bayesian network in such a way that the resulting joint distribution is a good representation of a given domain?
- Equation (2.1) implies certain conditional independence relationships that can be used to guide the knowledge engineer in constructing the topology of the network.
- Rewrite the joint distribution in terms of a conditional probability, using the product rule

product rule!!!

$$P(x_1,\ldots,x_n) = P(x_n \mid x_{n-1},\ldots,x_1)P(x_{n-1},\ldots,x_1)$$

- Repeat the process, reducing each conjunctive probability to a conditional probability and a smaller conjunction

$$P(x_1,\ldots,x_n) = P(x_n \mid x_{n-1},\ldots,x_1)P(x_{n-1} \mid x_{n-2}\ldots,x_1)\ldots P(x_2 \mid x_1)P(x_1)$$
$$= \prod_{i=1}^{n} P(x_i \mid x_{i-1},\ldots,x_1)$$

- Identity holds true for any set of random variables and is called the **chain rule**.

Equivalence

- Comparing it with Equation 2.1, we see that the specification of the joint distribution is equivalent to the general assertion that, for every variable X_i in the network,

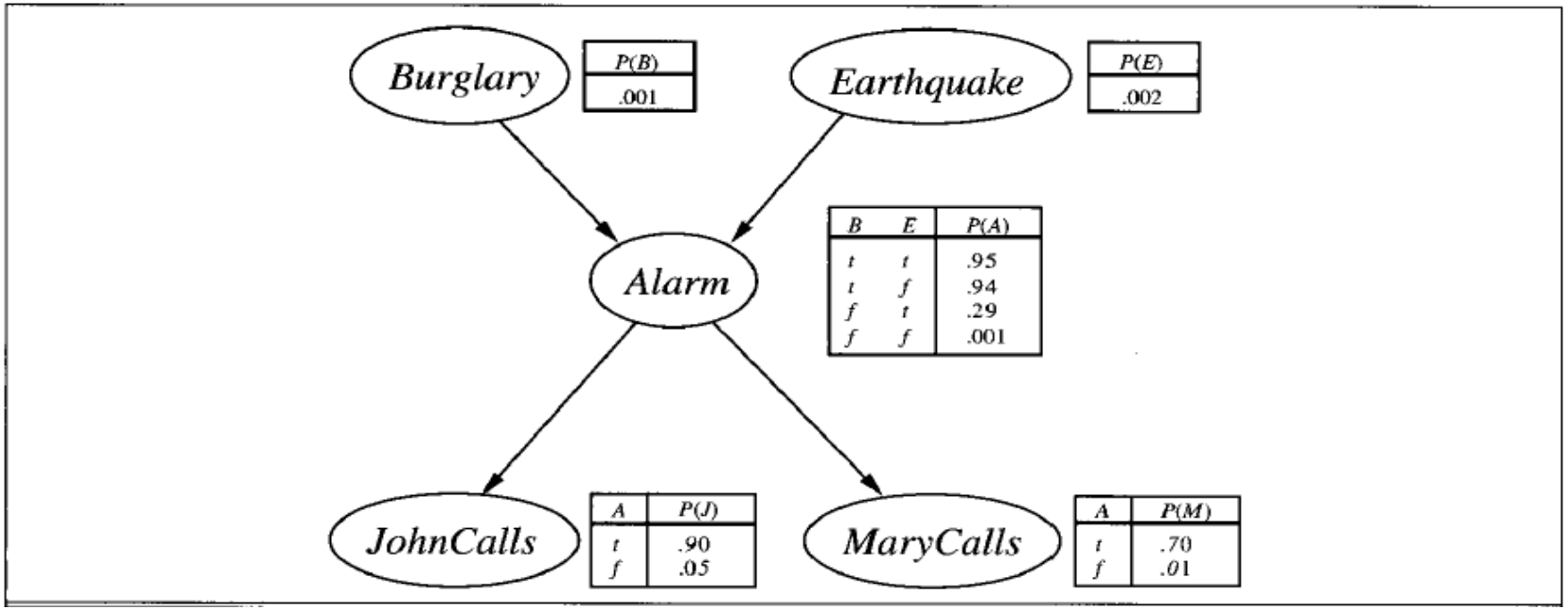
$$\mathbf{P}(X_i \mid X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i \mid Parents(X_i)) \qquad (2.2)$$

  provided that $Parents(X_i) \subset \{X_{i-1}, \ldots, X_1\}$

- Equation 2.2 says that Bayesian network is a correct representation of the domain only if each node is conditionally independent of its predecessors in the node ordering, given its parents.

- Intuitively, the parents of node X_i should contain all those nodes in X_1, ..., X_{i-1} that directly influence X_i.

Example



- suppose we have completed the network in Figure 14.2 except for the choice of parents for *MaryCalls*
- *MaryCalls* is certainly influenced by whether there is a *Burglary* or an *Earthquake*, but not directly influenced → no arc!
- events influence Mary's calling behavior only through their effect on the alarm

$$\mathbf{P}(MaryCalls \mid JohnCalls, Alarm, Earthquake, Burglary) = \mathbf{P}(MaryCalls \mid Alarm)$$

Compactness and node ordering

Compactness

- Bayesian network can often be far more compact than the full joint distribution.
- in case of Bayesian networks, it is reasonable to suppose that in most domains each random variable is directly influenced by at most k others
- if we assume n Boolean variables for simplicity, then the amount of information needed to specify each conditional probability table will be at most 2^k numbers,
  and the complete network can be specified by $n2^k$ numbers

  in contrast, the joint distribution contains 2^n numbers

Example

suppose we have $n = 30$ nodes, each with five parents ($k = 5$); Bayesian network requires 960 numbers, but the full joint distribution requires over a billion.

Node ordering

Correct order in which to add nodes:

> add the "root causes" first, then the variables they influence, and so on, until we reach the "leaves," which have no direct causal influence on the other variables.

What happens if we happen to choose the wrong order?

Burglary example:

Suppose we decide to add the nodes in the order *MaryCalls, JohnCalls, Alarm, Burglary, Earthquake;* then process goes as follows:

- Adding *MaryCalls*: No parents

Node ordering

Correct order in which to add nodes:

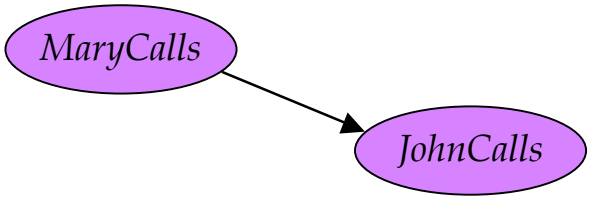
> add the "root causes" first, then the variables they influence, and so on, until we reach the "leaves," which have no direct causal influence on the other variables.

What happens if we happen to choose the wrong order?

Burglary example:

Suppose we decide to add the nodes in the order *MaryCalls, JohnCalls, Alarm, Burglary, Earthquake;* then process goes as follows:

- Adding *MaryCalls*: No parents
- Adding *JohnCalls*: if Mary calls this probably means alarm has gone off, which of course would make it more likely that John calls; therefore, *JohnCalls* needs *MaryCalls* as a parent



Node ordering

Correct order in which to add nodes:

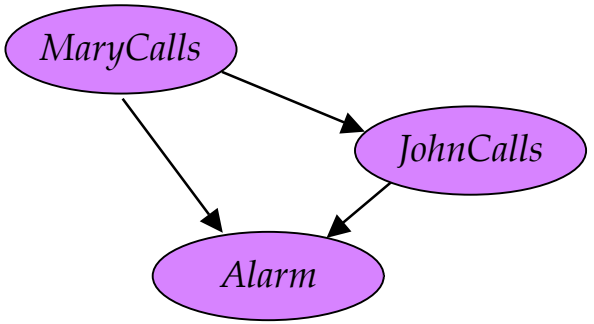
> add the "root causes" first, then the variables they influence, and so on, until we reach the "leaves," which have no direct causal influence on the other variables.

What happens if we happen to choose the wrong order?

Burglary example:

Suppose we decide to add the nodes in the order *MaryCalls, JohnCalls, Alarm, Burglary, Earthquake;* then process goes as follows:

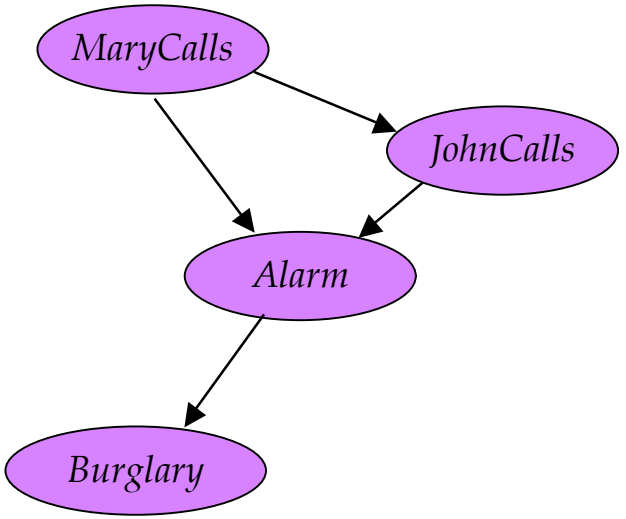
- Adding *MaryCalls*: No parents
- Adding *JohnCalls*: if Mary calls this probably means alarm has gone off, which of course would make it more likely that John calls; therefore, *JohnCalls* needs *MaryCalls* as a parent
- Adding *Alarm*: if both call, it is more likely that the alarm has gone off than if just one or neither call, so we need both *MaryCalls* and *JohnCalls* as parents.



- Adding $Burglary$: If we know alarm state, then call from John or Mary might give us information about our phone ringing or Mary's music, but not about burglary:

  $\mathbf{P}(Burglary \mid Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary \mid Alarm)$

  $\rightarrow$ only $Alarm$ as parent



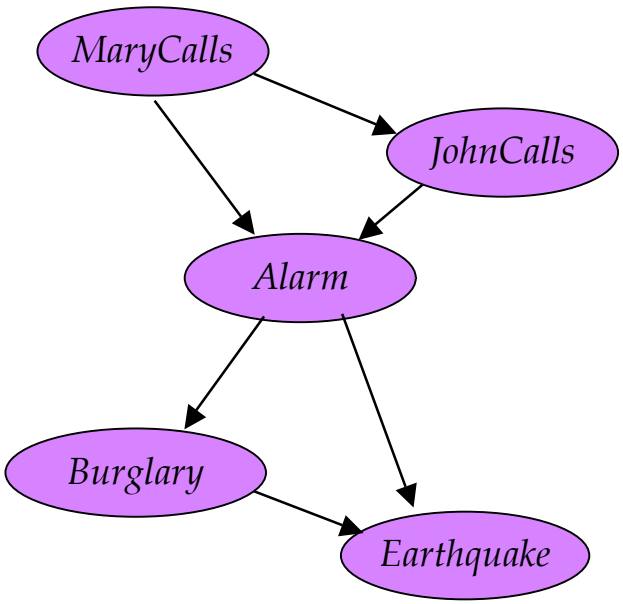
- Adding $Burglary$: If we know alarm state, then call from John or Mary might give us information about our phone ringing or Mary's music, but not about burglary:

  $\mathbf{P}(Burglary \mid Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary \mid Alarm)$

  $\rightarrow$ only $Alarm$ as parent

- Adding $Earthquake$: if alarm is on, it is more likely that there has been an earthquake; but if we know there has been a burglary, this explains the alarm, and probability of earthquake would be only slightly above normal.

  $\rightarrow$ we need both $Alarm$ and $Burglary$ as parent



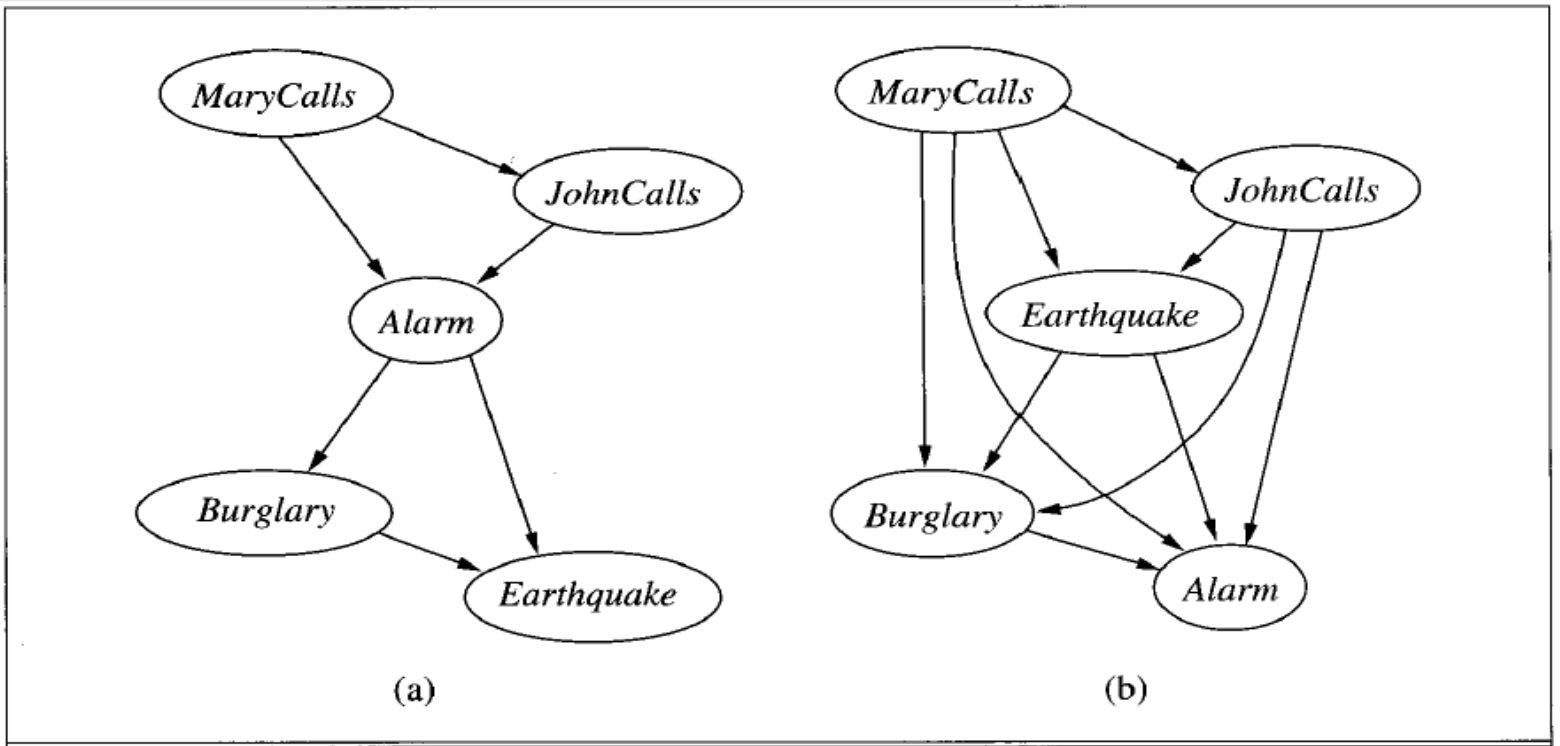


Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

- resulting network has two more links than the original network and requires three more probabilities to be specified.
- what is worse: some of the links represent tenuous relationships that require difficult and unnatural probability judgments, such as assessing the probability of *Earthquake*, given *Burglary* and *Alarm*.

Diagnostic vs. causal

- phenomenon is quite general and is related to the distinction between causal and diagnostic models
- if we try to build a diagnostic model with links from symptoms to causes (as from *MaryCalls* to *Alarm* or *Alarm* to *Burglary*), we end up having to specify additional dependencies between otherwise independent causes
- **if we stick to a causal model, we end up having to specify fewer numbers, and the numbers will often be easier to come up with.**
- figure (b) shows a very bad node ordering: *MaryCalls*, *JohnCalls*, *Earthquake*, *Burglary*, *Alarm*; network requires 31 distinct probabilities to be specified - exactly the same as the full joint distribution.

It is important to realize, that any of the three networks can represent exactly the same joint distribution; last two versions simply fail to represent all the conditional independence relationships and hence end up specifying a lot of unnecessary numbers instead.

Conditional independence relations in Bayesian networks

1. A node is conditionally independent of its non-descendants, given its parents.

   For example, in Figure 14.2, *JohnCalls* is independent of *Burglary* and *Earthquake*, given the value of *Alarm*.

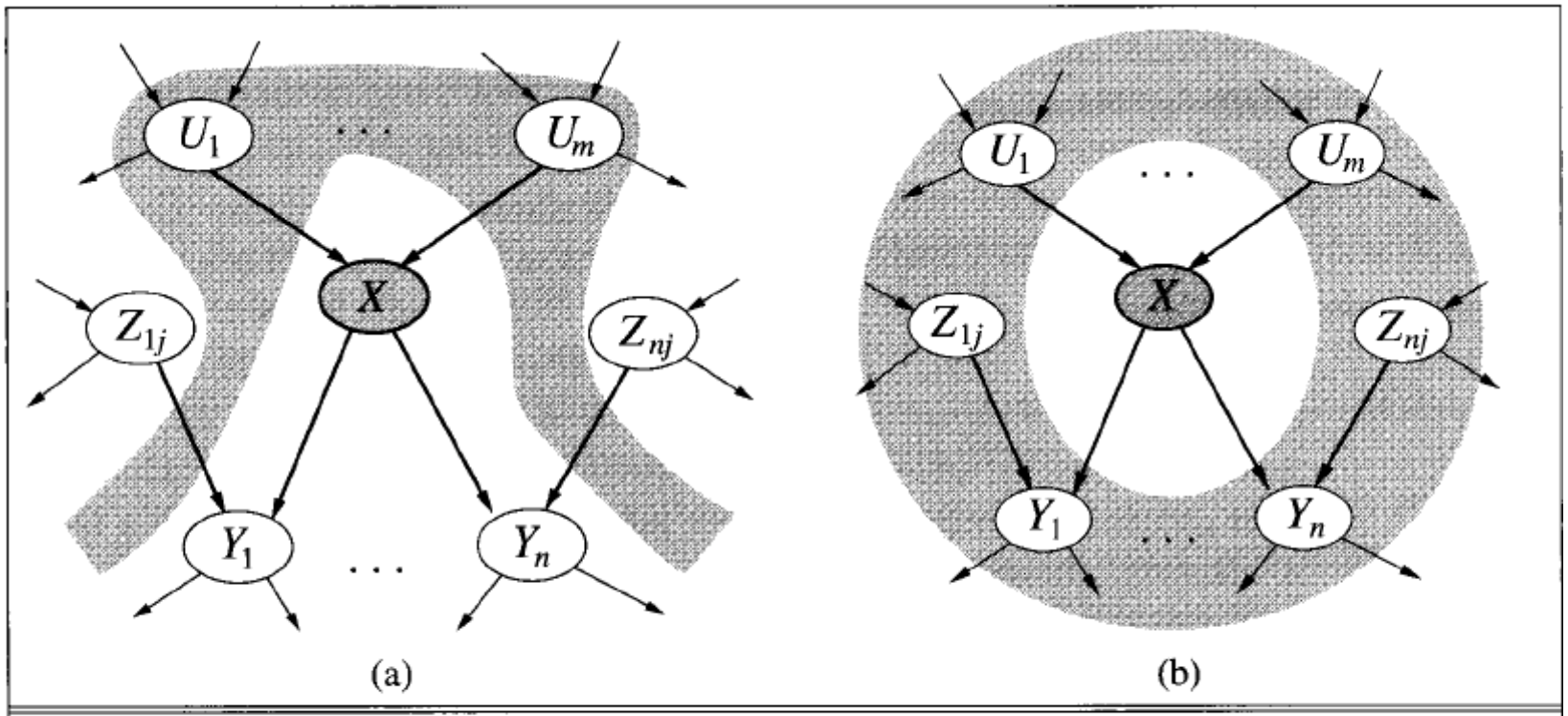


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

2. A node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents-that is, given its Markov blanket.

   For example, Burglary is independent of *JohnCalls* and *MaryCalls* , given *Alarm* and *Earthquake*.

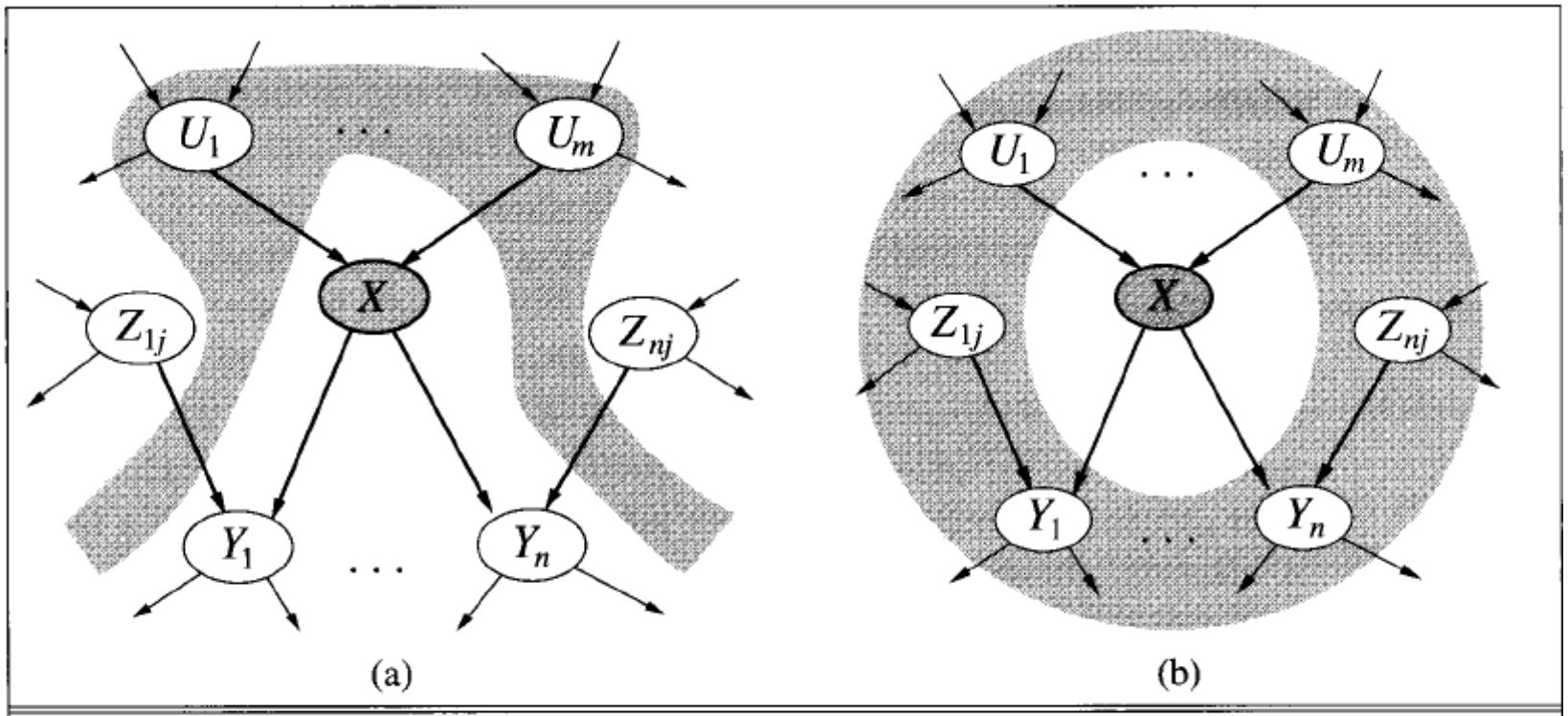


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

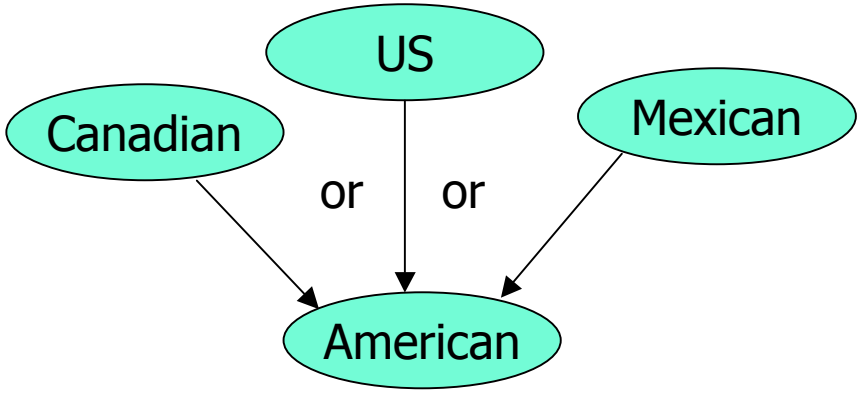
2.3 Efficient Representation of Cond. Distributions

Remark

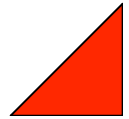
even if maximum number of parents k is smallish, filling in the CPT for a node requires up to $O(2^k)$ numbers

Certain relationships - deterministic nodes

- deterministic node has its value specified exactly by the values of its parents, with no uncertainty
- relationship can be a logical one: for example, the relationship between the parent nodes $Canadian$, US, $Mexican$ and the child node $NorthAmerican$ is simply that the child is the disjunction of the parents.



- relationship can also be numerical: for example, if the parent nodes are the prices of a particular model of car at several dealers, and child node is the price that a bargain hunter ends up paying, then the child node is the minimum of the parent values

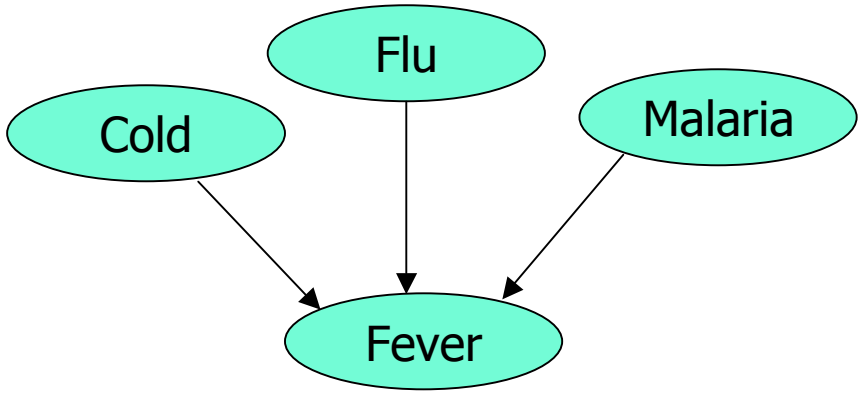


Uncertain relationships

- uncertain relationships often characterized by so-called "noisy" logical relationships
- standard example: noisy-OR relation (generalization of the logical OR)
- in propositional logic: $Fever$ is true if and only if $Cold$, Flu, or $Malaria$ is true.
- noisy-OR model allows for uncertainty about the ability of each parent to cause the child to be true
- the causal relationship between parent and child may be inhibited, and so a patient could have a cold, but not exhibit a fever

- individual inhibition probabilities:

$$P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$$



- model makes two assumptions.
  - all possible causes are listed (not as strict as it seems, because we can always add a so-called leak node that covers "miscellaneous causes.")
  - inhibition of each parent is independent of inhibition of any other parents: for example, whatever inhibits $Malaria$ from causing a fever is independent of whatever inhibits Flu from causing a fever.

Given these assumptions, $Fever$ is false if and only if all its *true parents* are inhibited, and the probability of $Fever = false$ is the product of the inhibition probabilities for earch parent.

- individual inhibition probabilities:

$$P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$$

- from this information and the noisy-OR assumptions, the entire CPT can be built

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T		
T	F	F	0.4	**0.6**
T	F	T		
T	T	F		
T	T	T		

- individual inhibition probabilities:

$$P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$$

- from this information and the noisy-OR assumptions, the entire CPT can be built

$Cold$	Flu	$Malaria$	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T		
T	T	F		
T	T	T		

- individual inhibition probabilities:

$$P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$$

- from this information and the noisy-OR assumptions, the entire CPT can be built

$Cold$	Flu	$Malaria$	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

- from this information and the noisy-OR assumptions, the entire CPT can be built

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

- in general, noisy logical relationships in which a variable depends on k parents can be described using $O(k)$ parameters instead of $O(2^k)$ for the full conditional probability table; makes assessment and learning much easier.

Bayesian networks with continuous variables

Discretization

- Many real-world problems involve continuous quantities, such as height, mass, temperature, and money
- Continuous variables have an infinite number of possible values, so it is impossible to specify conditional probabilities explicitly for each value
- One possible way to handle continuous variables is using discretization, i.e. dividing up the possible values into a fixed set of intervals:

  example: < 0°C , 0°C — 100°C and > 100°C

- Discretization is sometimes an adequate solution, but often results in a considerable loss of accuracy and very large CPTs.
- Other solution is to define standard families of probability density functions that are specified by a finite number of parameters.

  Example: Gaussian (or normal) distribution $N(\mu,\sigma^2)(x)$

Hybrid Bayesian networks

- Network with both discrete and continuous variables is called a hybrid Bayesian network.
- To specify a hybrid network, we have to specify two new kinds of distributions:
  - the conditional distribution for a continuous variable given discrete or continuous parents and
  - the conditional distribution for a discrete variable given continuous parents.

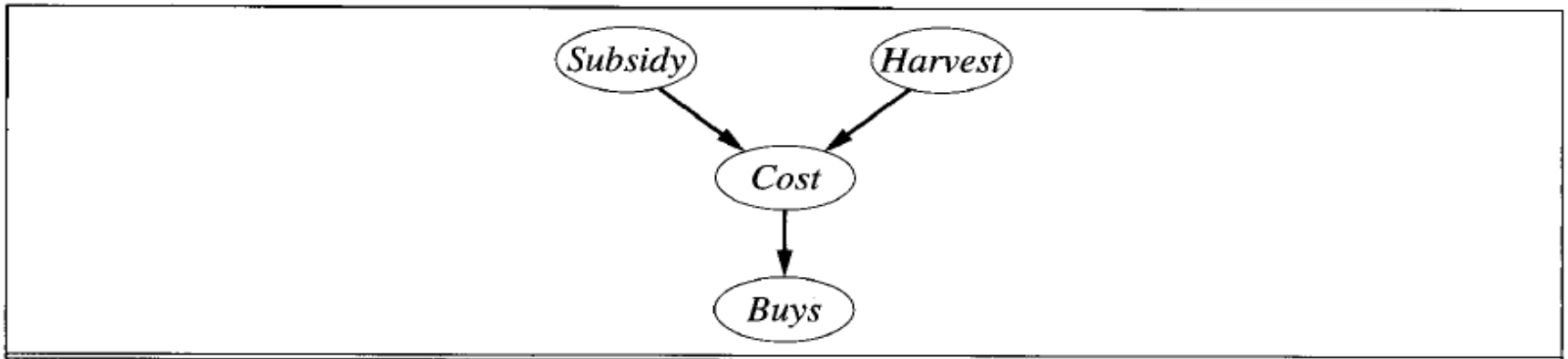


Figure 14.5 A simple network with discrete variables (*Subsidy* and *Buys*) and continuous variables (*Harvest* and *Cost*).

Example: Hybrid Bayesian networks

- Customer buys some fruit depending on its cost, which depends in turn on the size of the harvest and whether the government's subsidy scheme is operating.
- Variable $Cost$ is continuous and has continuous and discrete parents; variable $Buys$ is discrete and has a continuous parent
- Task: specify $\mathbf{P}(Cost \mid Harvest, Subsidy)$
- Discrete parent is handled by explicit enumeration: specifying both

  $\mathbf{P}(Cost \mid Harvest, subsidy)$ and $\mathbf{P}(Cost \mid Harvest, \neg subsidy)$
- To handle $Harvest$, specify how the distribution over the cost c depends on the continuous value h of $Harvest$

  ➔ linear Gaussian distribution

Linear Gaussian distribution

- Child has a Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed
- We need two distributions, one for $subsidy$ and one for $\neg\, subsidy$, with different parameters:

$$P(c \mid h, subsidy) = N(a_t h + b_t, \sigma_t^2)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_t h+b_t)}{\sigma_t}\right)^2}$$

$$P(c \mid h, \neg subsidy) = N(a_f h + b_f, \sigma_f^2)(c) = \frac{1}{\sigma_f \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_f h+b_f)}{\sigma_f}\right)^2}$$

The conditional distribution for $Cost$ is specified by naming the linear Gaussian distribution and providing the parameters a_t, b_t, σ_t, a_f, b_f, and σ_f

Linear Gaussian distribution

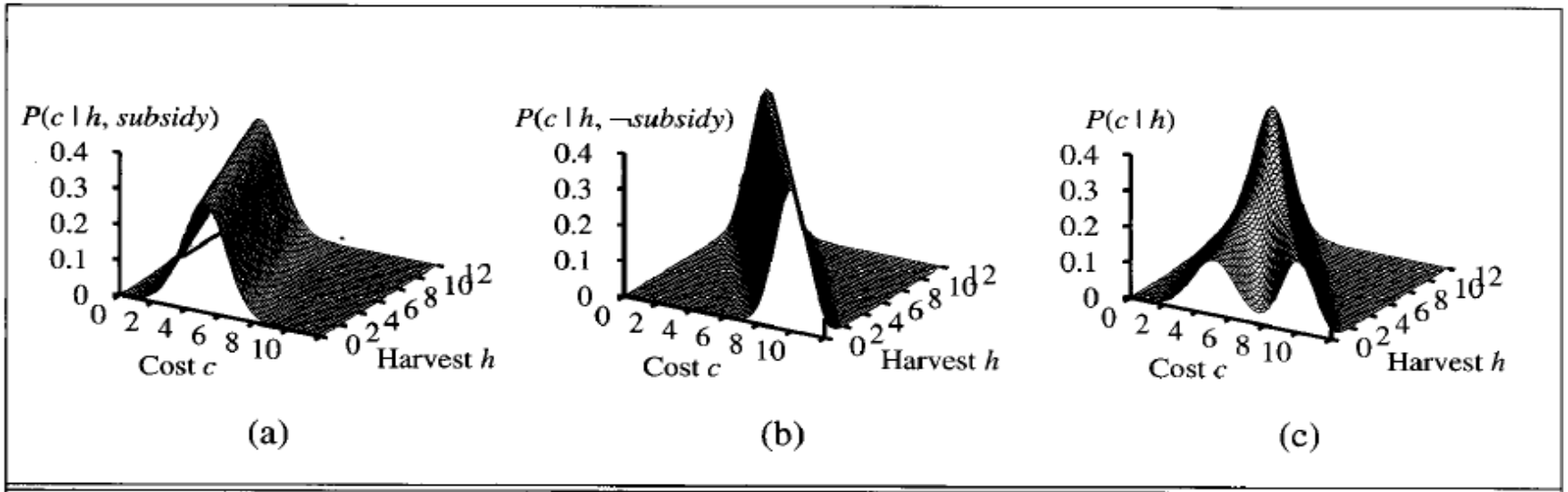


Figure 14.6 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false respectively. Graph (c) shows the distribution $P(Cost|Harvest)$, obtained by summing over the two subsidy cases.

- Notice: in each case the slope is negative, because price decreases as supply increases
- Figure 14.6 (c) shows the distribution $P(c \mid h)$, averaging over the two possible values of $Subsidy$ and assuming that each has prior probability 0.5

Distributions for discrete variables with continuous parents

Example: $Buys$ node in Figure 14.5.

- Customer will buy if the cost is low and will not buy if it is high; probability of buying varies smoothly in some intermediate region
- Conditional distribution is like a "soft" threshold function
- One way to make soft thresholds is to use the *integral* of the standard normal distribution:

$$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$$

- Then the probability of $Buys$ given $Cost$ might be

$$P(buys \mid Cost = c) = \Phi((-c + \mu)/\sigma)$$

Which means that the cost threshold occurs around μ, the width of the threshold region is proportional to σ, and the probability of buying decreases as cost

Distributions for discrete variables with continuous parents

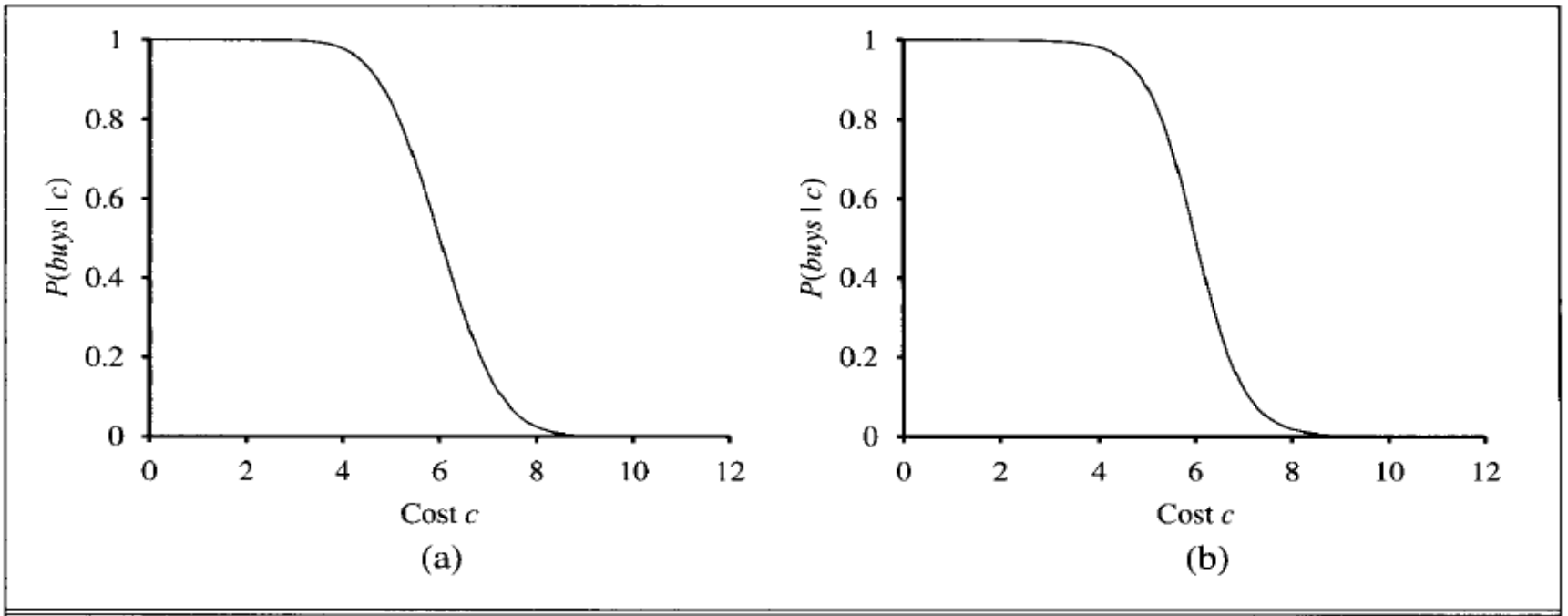


Figure 14.7 (a) A probit distribution for the probability of *Buys* given *Cost*, with $\mu = 6.0$ and $\sigma = 1.0$. (b) A logit distribution with the same parameters.

2.4 Exact Inference in Bayesian Networks

Basic task for any probabilistic inference system:

Compute the posterior probability distribution for a set of query variables, given some observed event (i.e., some assignment of values to a set of evidence variables)

- Notation:

  X denotes the query variable
  $\mathbf{E}$ denotes the set of evidence variables $E_1, \ldots, E_m$, and
  $\mathbf{e}$ is a particular observed event
  $\mathbf{Y}$ will denote the non-evidence variables $Y_1, \ldots, Y_l$, (sometimes called the hidden variables).

  complete set of variables: $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$.

- Typical query asks for the posterior probability distribution: $\mathbf{P}(X \mid \mathbf{e})$

- Example: burglary network

  observe the event in which $JohnCalls = true$ and $MaryCalls = true$;
  query the probability that a burglary has occurred:

  $$\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true) = \langle 0.284, 0.716 \rangle$$

Two questions:

- exact algorithms for computing posterior probabilities?
- complexity of this algorithm?

General case is intractable!!!

➔ Section 14.5 covers methods for approximate inference.

Inference by enumeration

- Recall:
  any conditional probability can be computed by summing terms from the full joint distribution.

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

- Eq. 2.1 showed that the terms $P(x, \mathbf{e}, \mathbf{y})$ in the joint distribution can be written as products of conditional probabilities from the network

  ➔ a query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network

- Example: $\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true)$.

  hidden variables: $Earthquake$ and $Alarm$.

$$\mathbf{P}(B \mid j,m) \;=\; \alpha\mathbf{P}(B,j,m) \;=\; \alpha\sum_{e}\sum_{\mathrm{a}}\mathbf{P}(B,e,a,j,m)$$

  semantics of Bayesian networks (Eq 2.1) gives us an expression in terms of CPT entries; for $Burglary = true$:

$$P(b \mid j,m) \;=\; \alpha\sum_{e}\sum_{\mathrm{a}}P(b)P(e)P(a \mid b,e)P(j \mid a)P(m \mid a) \qquad (2.3)$$

  How many terms?

- In the worst case, where we have to sum out almost all the variables, the complexity of the algorithm for a network with n Boolean variables is $O(n2^n)$.

- Improvement: the $P(b)$ term is a constant and can be moved outside the summations over a and e, and the $P(e)$ term can be moved outside the summation over a.

$$P(b \mid j,m) = \alpha P(b) \sum_{e} P(e) \sum_{\mathrm{a}} P(a \mid b,e) P(j \mid a) P(m \mid a)$$

- Improvement: the $P(b)$ term is a constant and can be moved outside the summations over a and e, and the $P(e)$ term can be moved outside the summation over a.

$$P(b \mid j,m) = \alpha P(b) \sum_{e} P(e) \sum_{a} P(a \mid b,e) P(j \mid a) P(m \mid a)$$

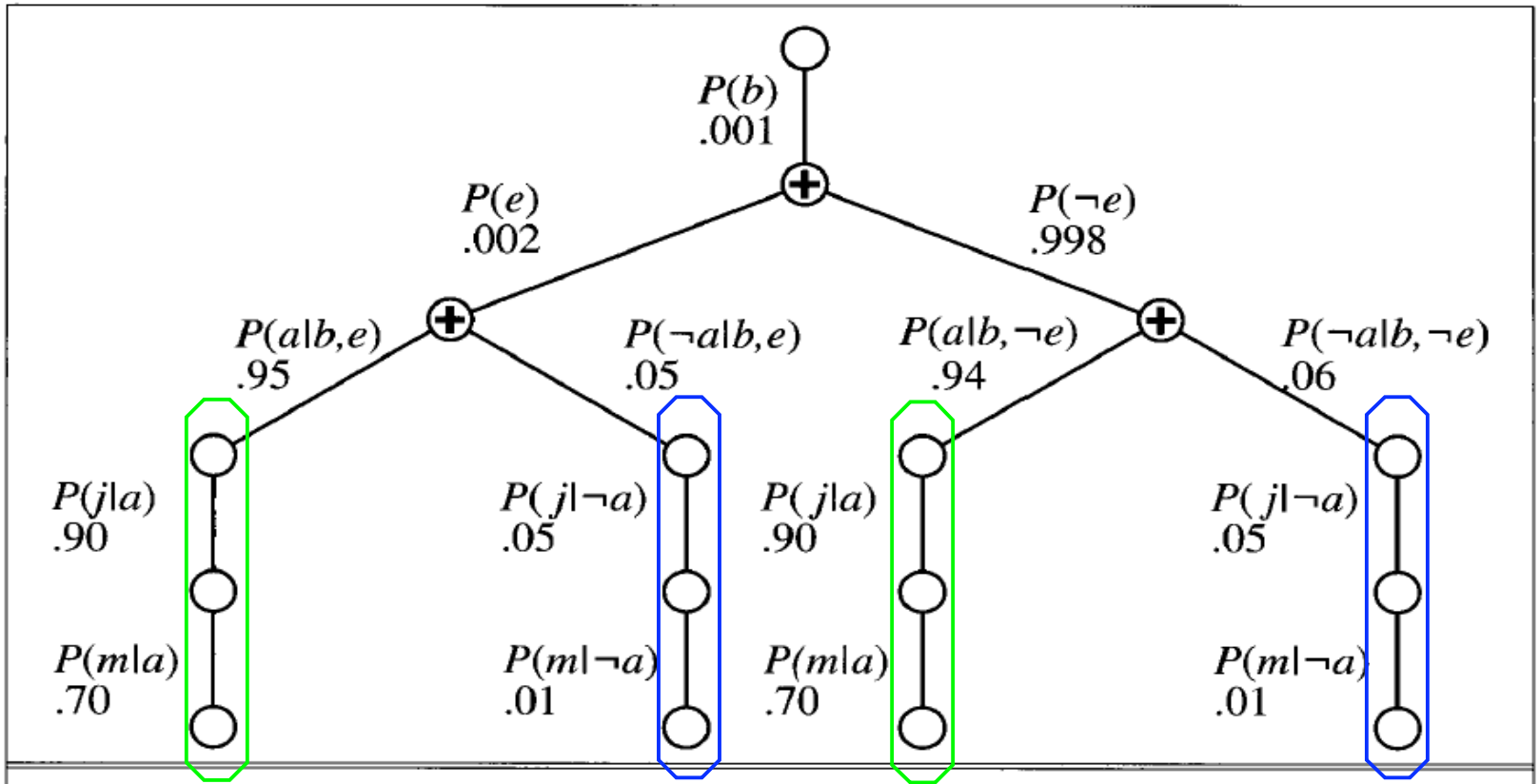


Figure 14.8 The structure of the expression shown in Equation (14.3). The evaluation proceeds top-down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for j and m.

- Improvement: the $P(b)$ term is a constant and can be moved outside the summations over a and e, and the $P(e)$ term can be moved outside the summation over a.

$$P(b \mid j,m) = \alpha P(b) \sum_{e} P(e) \sum_{a} P(a \mid b,e) P(j \mid a) P(m \mid a)$$

function ENUMERATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$ /* **Y** = *hidden variables* */

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    extend **e** with value x_i for X
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], **e**)
  return NORMALIZE($\mathbf{Q}(X)$)

---

function ENUMERATE-ALL($vars$, **e**) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in **e**
    then return $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), **e**)
    else return $\sum_y P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is **e** extended with $Y = y$

Figure 14.9 The enumeration algorithm for answering queries on Bayesian networks.

- Using the numbers from Figure 14.2, we obtain

$$\left.\begin{aligned} P(b \mid j,\ m) &= \alpha \times 0.00059224 \\ P(\text{-}b \mid j,\ m) &= \alpha \times 0.0014919 \end{aligned}\right\} \Rightarrow \mathbf{P}(B \mid j,m) = \alpha \langle 0.00059224, 0.0014919 \rangle$$

$$= \langle 0.284, 0.716 \rangle$$

- ENUMERATION-ASK algorithm in Figure 14.9 evaluates trees using depth-first recursion
- Space complexity of ENUMERATION-ASK is only linear in the number of variables- effectively, the algorithm sums over the full joint distribution without ever constructing it explicitly
- Time complexity for a network with n Boolean variables is always $O(2^n)$

Note:
Products $P(j \mid a)P(m \mid a)$ and $P(j \mid \neg a)P(m \mid \neg a)$ are computed twice, once for each value of e.

Desirable:
General method that avoids such wasted computations

Variable elimination algorithm

- Enumeration algorithm can be improved substantially by eliminating repeated calculations:

> Do the calculation once and save the results for later use!

(form of dynamic programming)

- Variable elimination works by
  - evaluating expressions such as Equation 2.3 in right-to-left order (bottom-up in Figure 14.8)
  - intermediate results are stored, and summations over each variable are done only for those portions of the expression that depend on the variable

- Example: burglary network - evaluate the expression

$$\mathbf{P}(B \mid j,m) = \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_{e} \underbrace{P(e)}_{E} \sum_{a} \underbrace{\mathbf{P}(a \mid B,e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M}$$

Parts B, E, A, J, M are called factors.

Variable elimination algorithm

$$\mathbf{P}(B \mid j,m) = \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_{e} \underbrace{P(e)}_{E} \sum_{a} \underbrace{\mathbf{P}(a \mid B,e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M}$$

- Factor for M, $P(m \mid a)$, store the probability, given each value of a, in a two-element vector,

$$\mathbf{f}_M(A) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} .70 \\ .01 \end{pmatrix}$$

- Similarly, we store the factor for J as the two-element vector $\mathbf{f}_J(A)$.
- Factor for A is $P(a \mid B,e)$, which will be a 2 x 2 x 2 matrix $\mathbf{f}_A(A,B,E)$.
- Next sum out A from the product of these three factors. will give us a 2 x 2 matrix whose indices range over just B and E.

pointwise multiplication

$$\mathbf{f}_{\bar{A}JM}(B,E) = \sum_{a} \mathbf{f}_A(a,B,E) \times \mathbf{f}_J(a) \times \mathbf{f}_M(a)$$
$$= \mathbf{f}_A(a,B,E) \times \mathbf{f}_J(a) \times \mathbf{f}_M(a) \;+\; \mathbf{f}_A(\neg a,B,E) \times \mathbf{f}_J(\neg a) \times \mathbf{f}_M(\neg a)$$

---

bar over A indicates that A has been summed out

- Pointwise product of two factors $\mathbf{f}_1$ and $\mathbf{f}_2$ yields a new factor $\mathbf{f}$ whose variables are the union of the variables in $\mathbf{f}_1$ and $\mathbf{f}_2$

$$\mathbf{f}(X_1 \ldots X_j, Y_1 \ldots Y_k, Z_1 \ldots Z_n) = \mathbf{f}_1(X_1 \ldots X_j, Y_1 \ldots Y_k)\mathbf{f}_2(Y_1 \ldots Y_k, Z_1 \ldots Z_n)$$

- If all the variables binary, then $\mathbf{f}_1$ and $\mathbf{f}_2$ have 2^{j+k} and 2^{k+l} entries respectively, and the pointwise product has 2^{j+k+l} entries
- Example:
given factors $\mathbf{f}_1(A,B)$ and $\mathbf{f}_2(B,C)$, pointwise product $\mathbf{f}_1 \times \mathbf{f}_2$ is given as $\mathbf{f}_3$ (A, B, C)

A	B	$\mathbf{f}_1(A,B)$	B	C	$\mathbf{f}_2(B,C)$	A	B	C	$\mathbf{f}_3(A,B,C)$
T	T	.3	T	T	.2	T	T	T	$.3 \times .2$
T	F	.7	T	F	.8	T	T	F	$.3 \times .8$
F	T	.9	F	T	.6	T	F	T	$.7 \times .6$
F	F	.1	F	F	.4	T	F	F	$.7 \times .4$
						F	T	T	$.9 \times .2$
						F	T	F	$.9 \times .8$
						F	F	T	$.1 \times .6$
						F	F	F	$.1 \times .4$

Variable elimination algorithm

$$\mathbf{P}(B \mid j,m) = \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_{e} \underbrace{P(e)}_{E} \sum_{a} \underbrace{\mathbf{P}(a \mid B,e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M}$$

- Process E in the same way: sum out E from the product of $\mathbf{f}_E(E)$ and $\mathbf{f}_{AJM}$(B,E)

$$\mathbf{f}_{\bar{E}\bar{A}JM}(B) = \mathbf{f}_E(e) \times \mathbf{f}_{\bar{A}JM}(B,e) + \mathbf{f}_E(\neg e) \times \mathbf{f}_{\bar{A}JM}(B,\neg e)$$

- Now we can compute the answer simply by multiplying the factor for B (i.e., $\mathbf{f}_B(B) = \mathbf{P}(B)$) by the accumulated matrix $\mathbf{f}_{\bar{E}\bar{A}JM}(B)$

$$\mathrm{P}(B \mid j,m) = \alpha \mathrm{f}_B(B) \times \mathrm{f}_{\bar{E}\bar{A}JM}(B)$$

Variable elimination algorithm

$$\mathbf{P}(B \mid j,m) = \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_{e} \underbrace{P(e)}_{E} \sum_{a} \underbrace{\mathbf{P}(a \mid B,e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M}$$

$$\underbrace{\mathbf{f}_A(A,B,E) \times \mathbf{f}_J(A) \times \mathbf{f}_M(A)}$$

$$\mathbf{f}_{\bar{A}JM}(B,E) = \underbrace{\mathbf{f}_A(a,B,E) \times \mathbf{f}_J(a) \times \mathbf{f}_M(a) + \mathbf{f}_A(\neg a,B,E) \times \mathbf{f}_J(\neg a) \times \mathbf{f}_M(\neg a)}$$

$$\mathbf{f}_{\bar{E}\bar{A}JM}(B) = \underbrace{\mathbf{f}_E(e) \times \mathbf{f}_{\bar{A}JM}(B,e) + \mathbf{f}_E(\neg e) \times \mathbf{f}_{\bar{A}JM}(B,\neg e)}$$

$$\underbrace{\mathbf{f}_B(B) \times \mathbf{f}_{\bar{E}\bar{A}JM}(B)}$$

- Altogether: two basic computational operations are required

  - pointwise product of factors
  - summing out a variable from a product of factors

1. Summing out a variable from a product of factors:
   any factor that does not depend on the variable to be summed out can be moved outside the summation process

$$\sum_e \mathbf{f}_E(e) \times \mathbf{f}_A(A,B,e) \times \mathbf{f}_J(A) \times \mathbf{f}_M(A) =$$
$$\mathbf{f}_J(A) \times \mathbf{f}_M(A) \times \sum_e \mathbf{f}_E(e) \times \mathbf{f}_A(A,B,e)$$

2. Pointwise product inside the summation is computed, and the variable is summed out of the resulting matrix

$$\mathbf{f}_J(A) \times \mathbf{f}_M(A) \times \sum_e \mathbf{f}_E(e) \times \mathbf{f}_A(A,B,e) = \mathbf{f}_J(A) \times \mathbf{f}_M(A) \times \mathbf{f}_{\bar{E},A}(A)$$

The complexity of exact inference

- variable elimination is more efficient than enumeration because it avoids repeated computations (as well as dropping irrelevant variables)
- time and space requirements of variable elimination are dominated by the size of the largest factor constructed during the operation of the algorithm; this in turn is determined by the order of elimination of variables and by the structure of the network
- burglary network of Figure 14.2 belongs to family of networks in which there is at most one undirected path between any two nodes in the network these are called singly connected networks or poly trees

  nice property: time and space complexity of exact inference in poly trees is linear in the size of the network.

- However:
  complexity in multiply connected networks can have exponential time and space complexity

polytree?

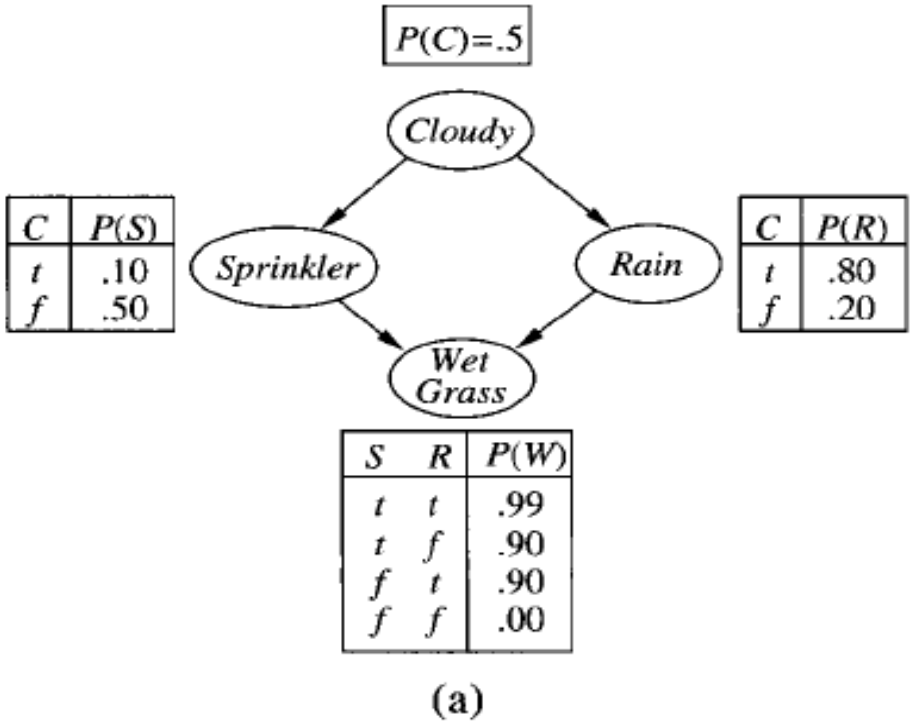


Figure 14.11 (a) A multiply connected network with conditional probability tables.

Clustering algorithms

- basic idea of clustering is to join individual nodes of the network to form cluster nodes in such a way that the resulting network is a polytree
- example: the multiply connected network shown in Figure 14.11 (a) can be converted into a polytree by combining the *Sprinkler* and *Rain* node into a cluster node called *Sprinkler*+*Rain*, as shown in Figure 14.11 (b).

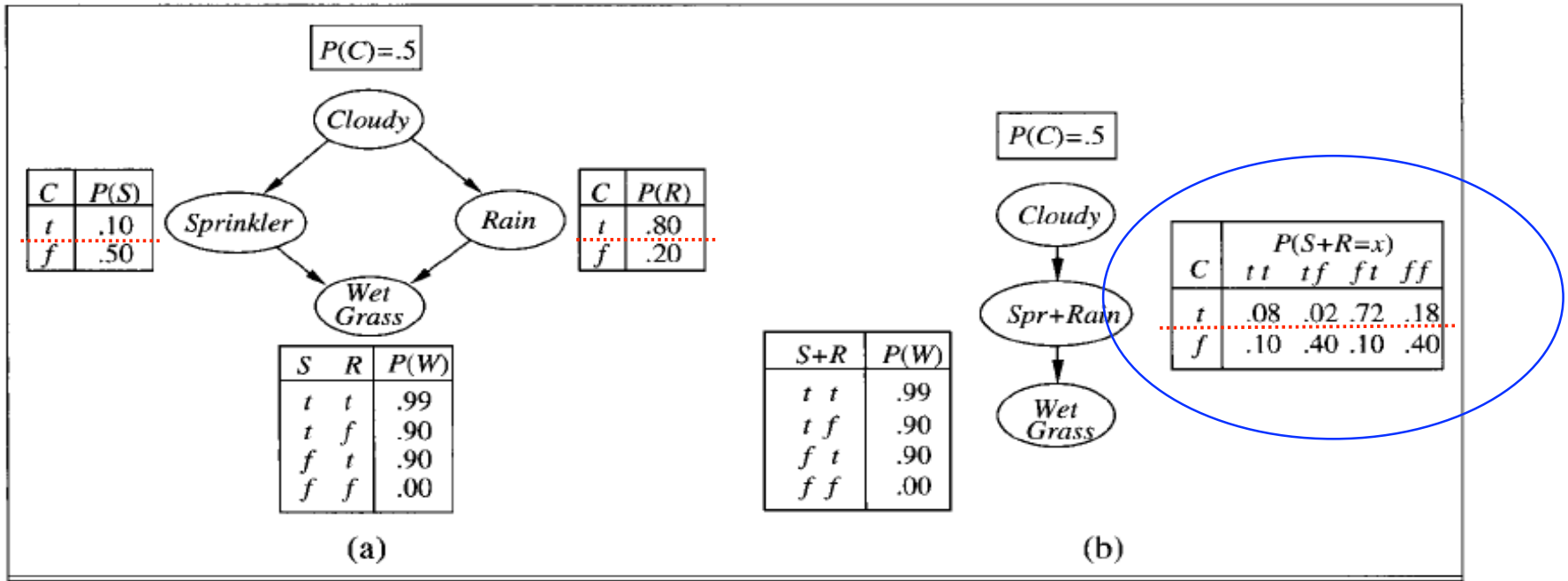


Figure 14.11 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

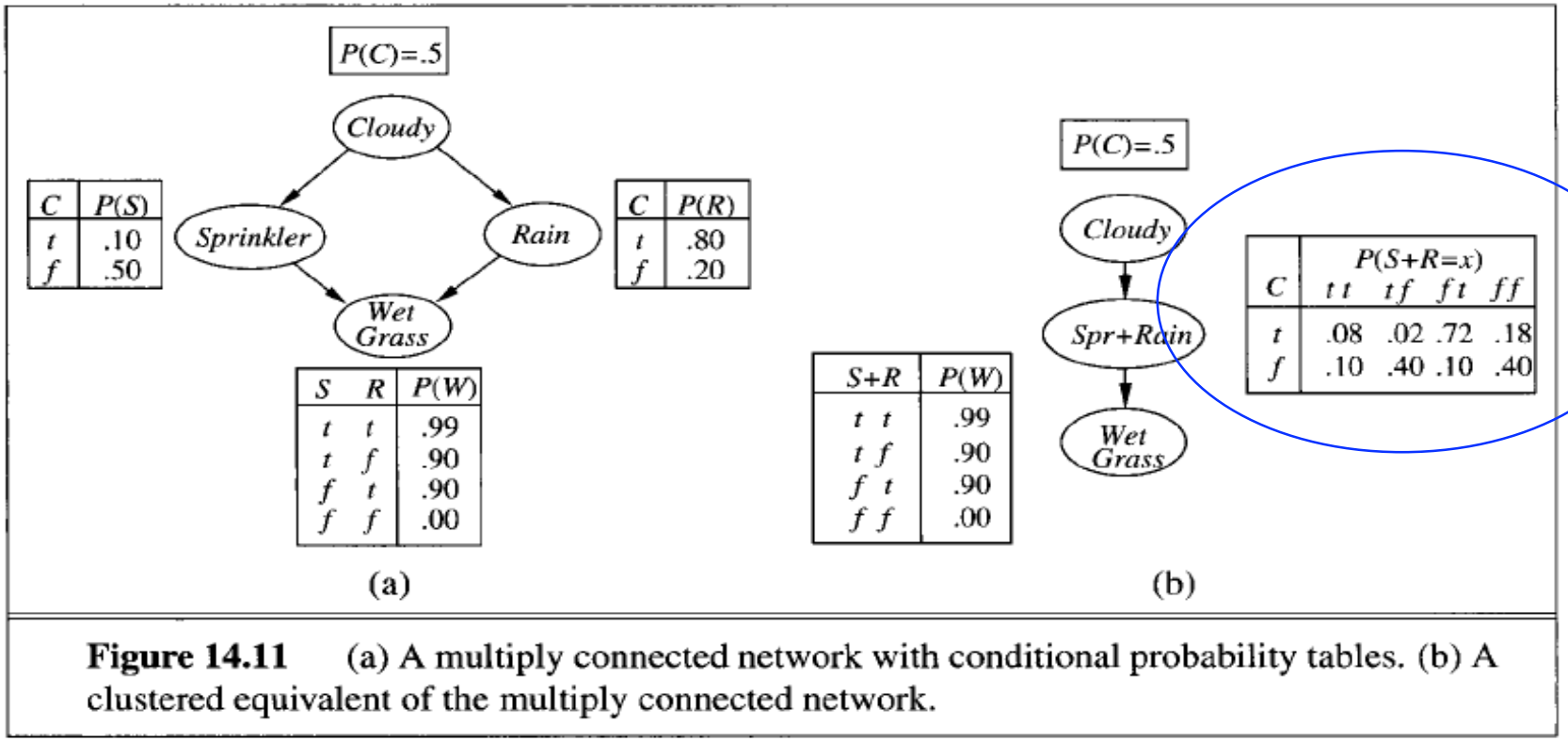


Figure 14.11 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

- The two Boolean nodes are replaced by a meganode that takes on four possible values: TT, TF, FT, and FF; meganode has only one parent, the Boolean variable *Cloudy*, so there are two conditioning cases.

2.5 Approximate Inference in Bayesian Networks

Monte Carlo algorithms

- Given intractability of exact inference in large, multiply connected networks, it is essential to consider approximate inference methods
- Monte Carlo algorithms: randomized sampling algorithms, that provide approximate answers whose accuracy depends on the number of samples generated
- Widely used in computer science to estimate quantities that are difficult to calculate exactly
- We are interested in sampling applied to the computation of posterior probabilities
- Two families of algorithms: direct sampling and Markov chain sampling

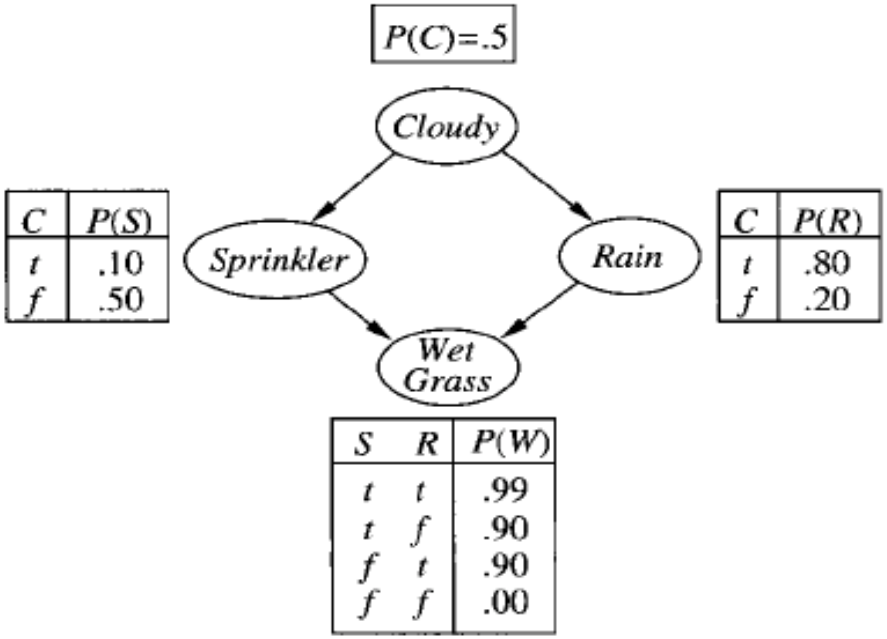
Direct sampling methods

- Basic element in any sampling algorithm is the generation of samples from known probability distribution
- Example:

  unbiased coin can be thought of as random variable $Coin$ with values $\langle heads, tails \rangle$ and a prior distribution $\mathbf{P}(Coin) = \langle 0.5, 0.5 \rangle$

  sampling from this distribution is exactly like flipping the coin: with probability 0.5 it will return $heads$, and with probability 0.5 it will return $tails$
- Simplest kind of random sampling process for Bayesian networks generates events from a network that has no evidence associated with it
- Idea is to sample each variable in turn, in topological order;
- Probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents

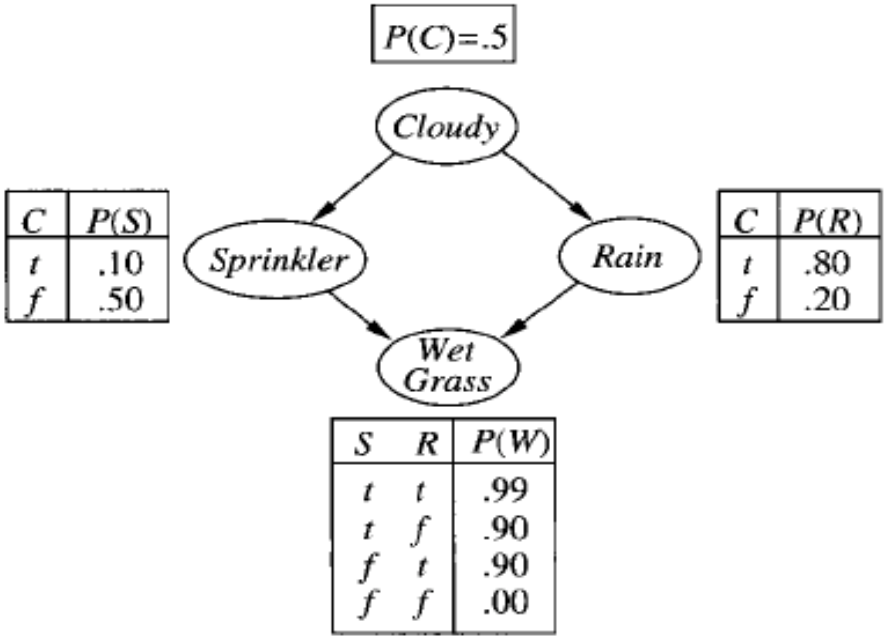
Example: Sampling the *Sprinkler-Rain-WettGrass* network



- Ordering [*Cloudy, Sprinkler, Rain, WetGrass*]
  1. sample from $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$; suppose this returns *true*
  2. sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*
  3. sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*
  4. sample from $\mathbf{P}(WetGrass \mid Sprinkler{=}false, Rain = true) = \langle 0.9, 0.1 \rangle$; suppose this returns *true*

  ➔ sample algorithm returns the event [*true, false, true, true*]

Example: Sampling the *Sprinkler-Rain-WettGrass* network



- Ordering [*Cloudy, Sprinkler, Rain, WetGrass*]
  1. sample from $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$; suppose this returns *true*
  2. sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*
  3. sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*
  4. sample from $\mathbf{P}(WetGrass \mid Sprinkler{=}false, Rain = true) = \langle 0.9, 0.1 \rangle$; suppose this returns *true*

  ➔ sample algorithm returns the event $[true, false, true, true] = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324$

Algorithm: PRIOR-SAMPLE

function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

x ← an event with n elements
for $i = 1$ **to** n **do**
  $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
return x

Figure 14.12 A sampling algorithm that generates events from a Bayesian network.

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by the PRIOR-SAMPLE algorithm.
- Just looking at the sampling process, we have

$$S_{PS}(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i))$$

because each sampling step depends only on the parent values

- Looks familiar, because it is also the probability of the event according to the BN representation of the joint distribution

$$S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n)$$

because each sampling step depends only on the parent values

- Allows us to answer queries by using samples: the answers are computed by counting the actual samples generated
- Suppose there are N total samples, and let $N(x_1,\ldots,x_n)$ be the frequency of the specific event $x_1,\ldots,x_n$.
- We expect this frequency to converge, in the limit, to its expected value according to the sampling probability:

$$\lim_{N\to\infty} \frac{N_{PS}(x_1,\ldots,x_n)}{N} = S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n) \qquad (2.4)$$

- Consider the event produced earlier: $[true, false, true, true]$.

  Sampling probability for this event is

$$S_{PS}(true, false, true, true) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324$$

$\rightarrow$ in the limit of large N, we expect 32.4% of the samples to be of this event.

Whenever we use an approximate equality ("≈") in what follows, we mean it in exactly this sense - that the estimated probability becomes exact in the large-sample limit. Such an estimate is called consistent.

$$P(x_1, \ldots, x_m) \approx N_{PS}(x_1, \ldots, x_m) / N \qquad (2.5)$$

That is, the probability of the event can be estimated as the fraction of all complete events generated by the sampling process that match the partially specified event.

- Example: if we generate 1000 samples from the sprinkler network, and 511 of them have $Rain = true$, then the estimated probability of rain,

$$\hat{P}_{PS}(Rain = true) = 0.511$$

Rejection sampling in Bayesian networks

- Can be used to compute conditional probabilities - that is, to determine $P(X \mid \mathbf{e})$.
- Algorithm
  1. generates samples from the prior distribution specified by the network
  2. rejects all those that do not match the evidence
  3. estimate $P(X=x \mid \mathbf{e})$ is obtained by counting how often $X = x$ occurs in the remaining samples
- Let $\hat{\mathrm{P}}(X \mid \mathrm{e})$ be the estimated distribution that the algorithm returns. From the definition of the algorithm, we have

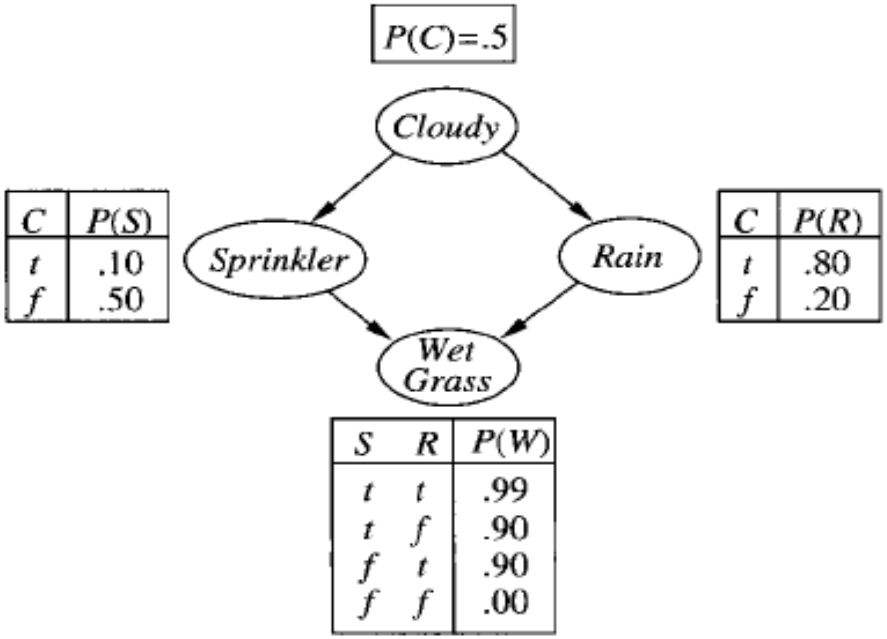
$$\hat{\mathrm{P}}(X \mid \mathrm{e}) = \alpha \mathrm{N}_{PS}(X, \mathrm{e}) = \frac{\mathrm{N}_{PS}(X, \mathrm{e})}{\mathrm{N}_{PS}(\mathrm{e})}$$

From (2.5) this becomes

$$\hat{\mathrm{P}}(X \mid \mathrm{e}) \approx \frac{\mathrm{P}(X, \mathrm{e})}{P(\mathrm{e})} = \mathrm{P}(X \mid \mathrm{e})$$

That is, rejection sampling produces a consistent estimate of the true probability.

Example: Sampling the *Sprinkler-Rain-WettGrass* network



- We wish to estimate $\mathbf{P}(Rain \mid Sprinkler = true)$, using 100 samples
- Of these 100 samples, suppose that 73 have $Sprinkler = false$ and are rejected, while 27 have $Sprinkler = true$; of the 27, 8 have $Rain = true$ and 19 have $Rain = false$.
- Hence, $\mathbf{P}(Rain \mid Sprinkler = true) \approx \text{NORMALIZE}(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle$.
- True answer: $\langle 0.3, 0.7 \rangle$

Rejection sampling

- As more samples are collected, the estimate will converge to the true answer.
- Standard deviation of the error in each probability will be proportional to $1/\sqrt{n}$ where n is the number of samples used in the estimate
- Biggest problem with rejection sampling is that it rejects so many samples!
- Fraction of samples consistent with the evidence $\mathbf{e}$ drops exponentially as the number of evidence variables grows

  ➔ procedure is simply unusable for complex problems

Remark

- Rejection sampling is very similar to the estimation of conditional probabilities directly from the real world.

  Example: $\mathbf{P}(\mathit{Rain} \mid \mathit{RedSkyAtNight} = \mathit{true})$

  Simply count how often it rains after a red sky is observed the previous evening, ignoring those events, when the sky is not red

Likelihood weighting

- Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence $\mathbf{e}$ [1)]
- Fixes the values for the evidence variables $\mathbf{E}$ and samples only the remaining variables X and $\mathbf{Y}$
- This guarantees that each event generated is consistent with the evidence
- Before tallying counts in the distribution for the query variable, each event is weighted by the likelihood that the event accords to the evidence
- Intuitively, events in which the actual evidence appears unlikely should be given less weight.

---

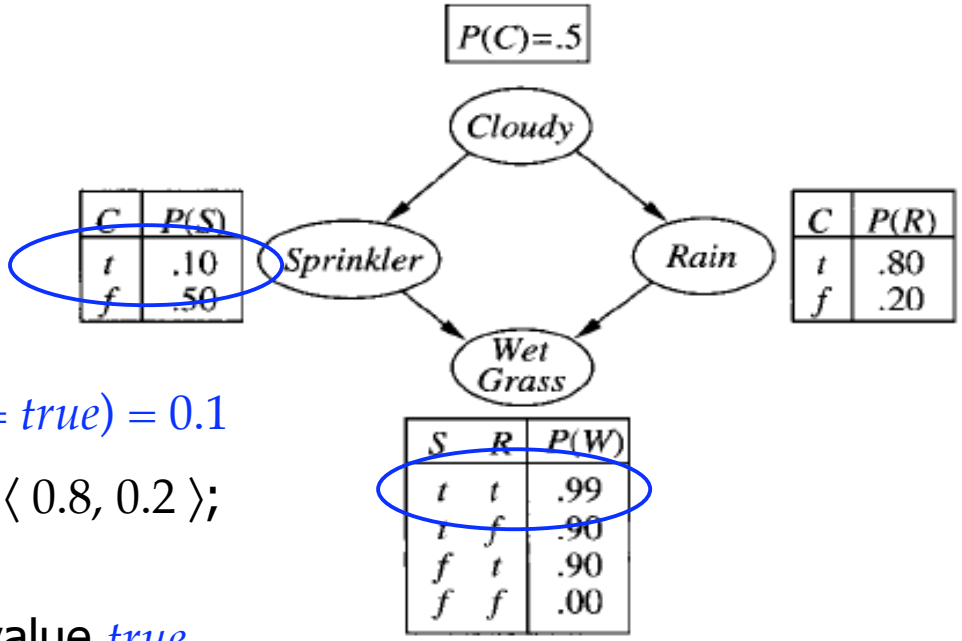
1) as measured by the product of the conditional probabilities for each evidence variable, given its parents

Example: sampling the *Sprinkler-Rain-WettGrass* network

- Query $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$ in *Sprinker-Rain-WetGrass* network
- First, the weight w is set to 1.0.
  Then an event is generated:
  1. sample from $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$; suppose this returns *true*.
  2. *Sprinkler* is an evidence variable with value *true*. Therefore, we set
     $w \leftarrow w \times P(Sprinkler = true \mid Cloudy = true) = 0.1$
  3. sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*.
  4. *WetGrass* is an evidence variable with value *true*.

     Therefore, we set

     $w \leftarrow w \times P(WettGrass = true \mid Sprinkler = true, Rain = true) = 0.099$



- Returns the event $[true, true, true, true]$ with weight 0.099; this is tallied under $Rain = true$
- Weight is low because the event describes a cloudy day, which makes the sprinkler unlikely to be on

Likelihood weighting

- Because likelihood weighting uses all the samples generated, it can be much more efficient than rejection sampling
- It will, however, suffer a degradation in performance as the number of evidence variables increases

Inference by Markov chain simulation

Markov chain Monte Carlo (MCMC) algorithm

- Unlike other sampling algorithms, which generate each event from scratch, MCMC generates each event by making random change to preceding event/state
- Next event/state is generated by randomly sampling a value for one of the nonevidence variables X_i, conditioned on the current values of the variables in the Markov blanket
- MCMC therefore wanders randomly around the state space – the space of possible complete assignments – flipping one variable at a time, but keeping the evidence variables fixed.

Markov blanket of a variable consists of its parents, children, and children's parents

Example: query $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$

- Evidence variables $Sprinkler$ and $WetGrass$ are fixed to their observed values
- Hidden variables $Cloudy$ and $Rain$ initialized randomly: $true$ and $false$ respectively
- Initial state is $[true, true, false, true]$
- Following steps are executed repeatedly:
  1. $Cloudy$ is sampled, given current values of its Markov blanket variables: in this case, we sample from $\mathbf{P}(Cloudy \mid Sprinkler = true, Rain{=}false)$.

     Suppose result is $Cloudy = false$; new current state is $[false, true, false, true]$.
  2. $Rain$ is sampled, given the current values of its Markov blanket variables: we sample from $\mathbf{P}(Rain \mid Cloudy = false, Sprinkler = true, WetGrass = false)$.

     Suppose this yields $Rain = true$; new current state is $[false, true, true, true]$
- If the process visits 20 states where $Rain$ is true and 60 states where $Rain$ is false, then answer to the query is $\text{NORMALIZE}(\langle 20{,}60 \rangle) = \langle\, 0.25, 0.75 \,\rangle$

2.6 Summary

- A Bayesian network is a directed acyclic graph whose nodes correspond to random variables; each node has a conditional distribution for the node, given its parents.
- Bayesian networks provide a concise way to represent conditional independence relationships in the domain.
- A Bayesian network specifies a full joint distribution; each joint entry is defined as the product of the corresponding entries in the local conditional distributions. A Bayesian network is often exponentially smaller than the full joint distribution.
- Many conditional distributions can be represented compactly by canonical families of distributions. Hybrid Bayesian networks, which include both discrete and continuous variables, use a variety of canonical distributions.
- Inference in Bayesian networks means computing the probability distribution of a set of query variables, given a set of evidence variables. Exact inference algorithms, such as variable elimination, evaluate sums of products of conditional probabilities as efficiently as possible.

- In poly trees (singly connected networks), exact inference takes time linear in the size of the network. In the general case, the problem is intractable.
- Stochastic approximation techniques such as likelihood weighting and Markov chain Monte Carlo can give reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.