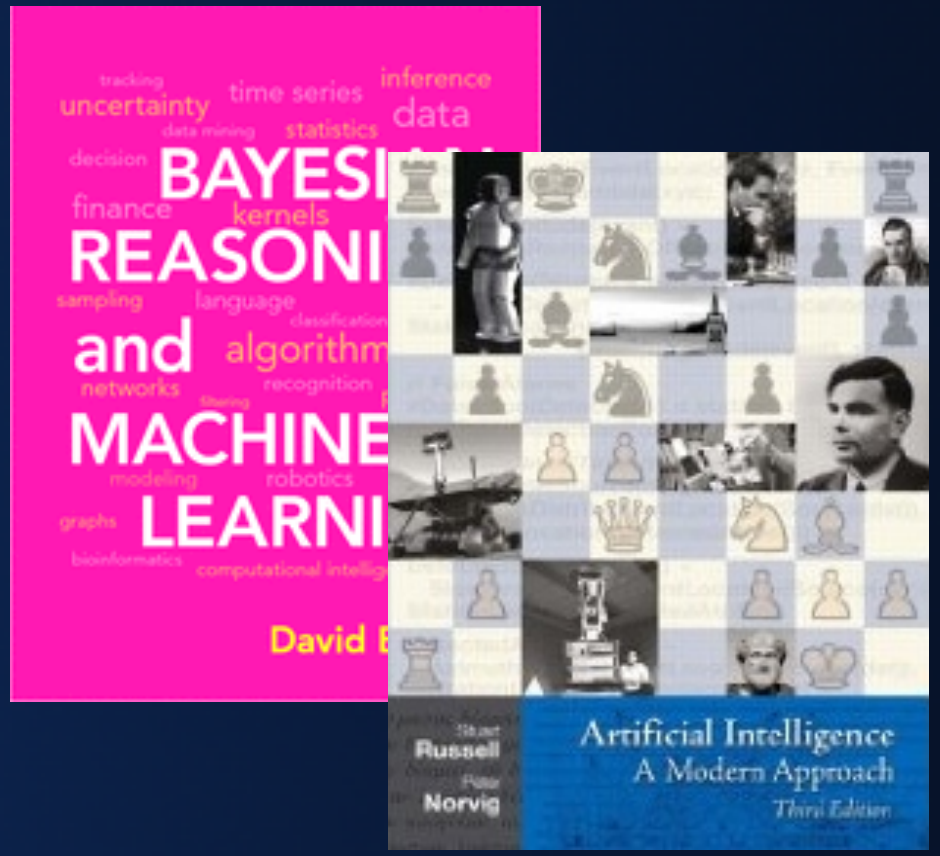


AI Fundamentals: uncertain reasoning

Maria Simi

Probabilistic reasoning
AIMA Chapter 14

LESSON 2: BELIEFS NETWORKS – SEMANTICS – EXACT INFERENCE

Summary

- Definition of Bayesian networks
- Semantics of Bayesian networks
- Efficient representation of conditional distribution
- Exact inference in Bayesian networks
- Two algorithms for computing on Bayesian networks:
  - ✓ Enumeration-Ask
  - ✓ Variable-Elimination

Bayesian networks/belief networks

Bayesian networks (also called **belief networks**) are graphs for representing dependencies among variables.

The network makes explicit *conditional dependencies*: the intuitive meaning of an arc from X to Y is typically that **X has a direct influence on Y.**

Bayesian networks are directed acyclic graphs (DAG) so defined:

1. Each node corresponds to a random variable, which may be discrete or continuous.
2. Directed links or arcs connect pairs of nodes. If there is an arc from node X to node Y, X is said to be a parent of Y. $Parents(Y)$ is the set of variables directly influencing Y.
3. Each node X has an associated **conditional probability distribution table P**$(X \mid Parents(X))$ that quantifies the effect of the parents on the node.

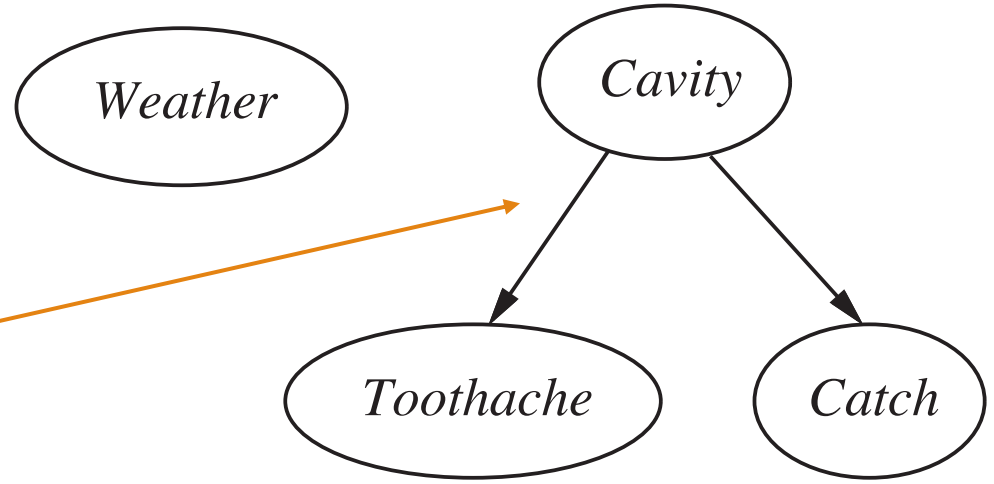
Easier for domain experts to decide what direct influences exist in the domain than actually specifying the probabilities themselves.

The *Cavity* & *Weather* example

The network naturally represents independence and conditional independence:

1. *Weather* is **independent** from the other variables (*Cavity, Toothache, Catch*).
2. *Toothache* and *Catch* are **conditionally independent**, given *Cavity*.

The lack of an arc connection between two variables is interpreted as **independence**.



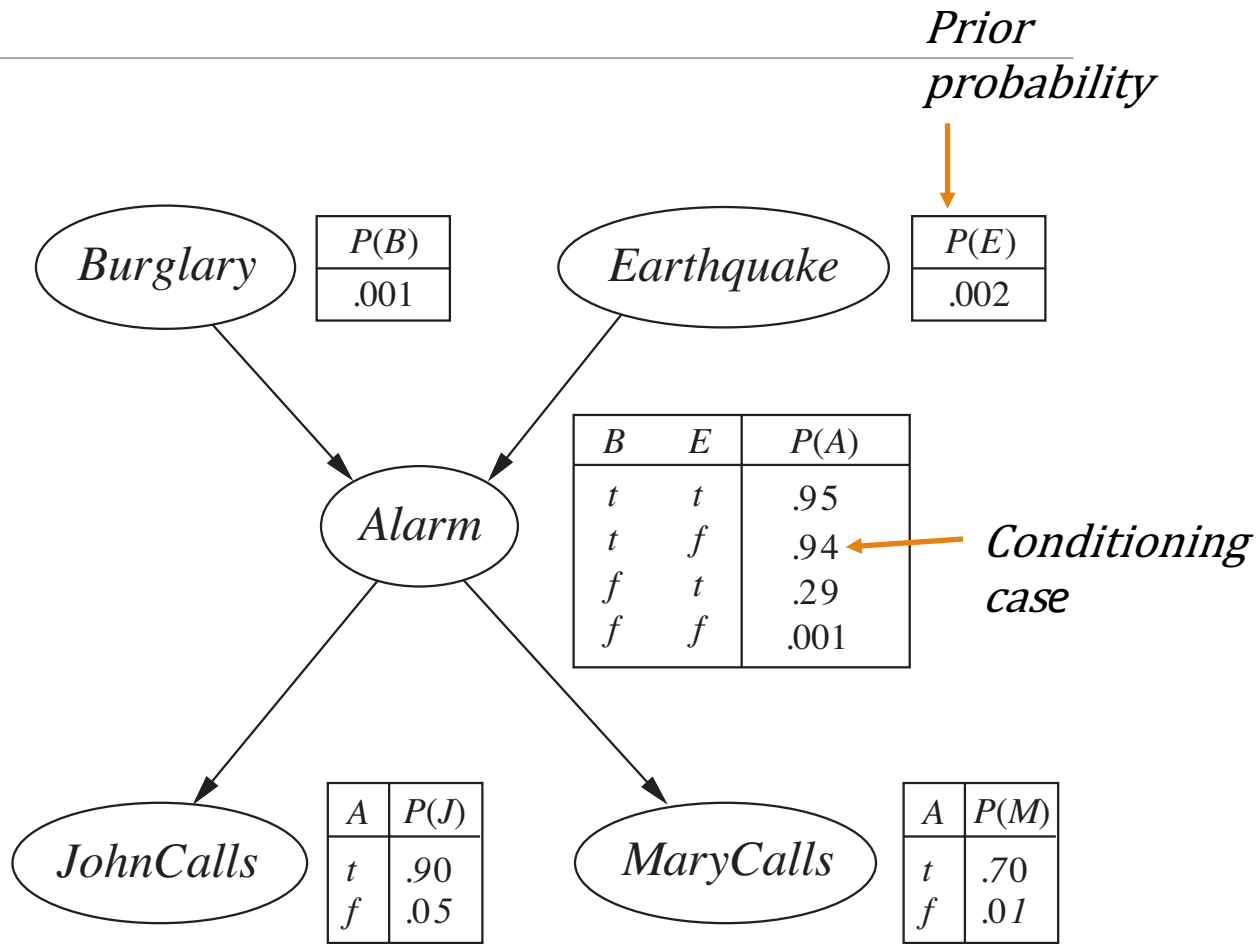
The *alarm* example

The burglary alarm goes off very likely on burglary and occasionally on earthquakes. John and Mary are neighbors who agreed to call when the alarm goes off. Their reliability is different …

A typical Bayesian network, showing both the topology and the **Conditional Probability Tables** (CPTs).

In the CPTs this abbreviations are used:

- *B: Burglary*
- *E: Earthquake*
- *A: Alarm*
- *J: JohnCalls*
- *M: MaryCalls*



Semantics of Bayesian networks

Semantics of Bayesian networks

Two alternative and equivalent ways to look at the semantics of Bayesian networks:

1. Distributed representation of the full joint probability distribution
   - Suggests a methodology for constructing networks
2. An encoding of a collection of conditional independence statements
   - Helps in designing inference procedures

Semantics of the networks

A Bayesian network can be seen as a **representation of the full joint distribution**.

A joint probability distribution can be expressed in terms of conditional probabilities:

$$P(x_1, \dots, x_n) = P(x_n | x_{n-1}, \dots, x_1)\, P(x_{n-1} \dots x_1) \qquad \textit{by the product rule}$$

By iterating the process we get the so-called **chain rule**:

$$P(x_1, \dots, x_n) = P(x_n | x_{n-1}, \dots, x_1)\, P(x_{n-1} | x_{n-2}, \dots, x_1) \dots P(x_2 | x_1)\, P(x_1) \qquad (\textit{chain rule})$$

$$= \prod_{i=1}^{n} P(x_i | x_{i-1}, \dots, x_1)$$

This amounts to:

1. assuming an ordering of the variables
2. computing the posterior probabilities of each variable, given all previous variables
3. Taking the product of these posteriors.

Representing the joint probability distribution

Given that $P(x_1, \dots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_{i-1}, \dots, x_1)$

Assuming:

1. Each variable appears after its parents in the ordering, i.e. $Parents(Xi) \subseteq \{X_{i-1}, \dots, X_1\}$
2. We simplify the computation by conditioning the computation only to values of the parent variables (assuming the others are independent)
3. The numbers in the CPT's are actually conditional probabilities.

Each entry in the joint distribution can be computed as the product of the appropriate entries in the conditional probability tables (CPTs) in the Bayesian network.

We get the simplified rule for computing joint distributions:

$$P(x_1, \dots, x_n) = \prod_{i=1}^{n} P(xi \mid parents(Xi))$$

where $parents(Xi)$ denotes the set of values $x_1, \dots, x_n$ for $parents(Xi)$

A Bayesian network is a **distributed representation of the full joint distribution**.

An example

Example. Given the *Alarm* network problem and a suitable ordering of the variables (J, M, A, B, E) we can compute, by the *chain rule*:

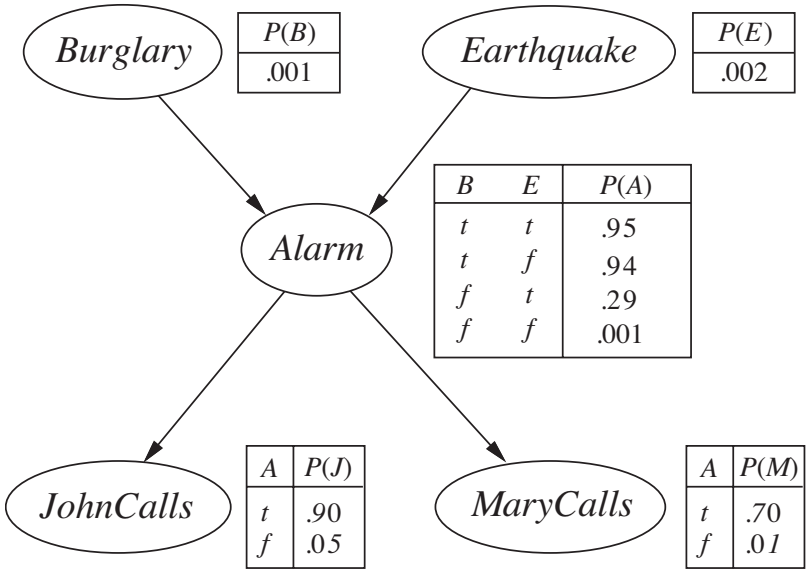
$$P(j, m, a, \neg b, \neg e) = P(j \mid m, a, \neg b, \neg e)\, P(m \mid a, \neg b, \neg e)\, P(a \mid \neg b, \neg e)\, P(\neg b \mid \neg e)\, P(\neg e)$$

and taking into account only direct dependencies, we can simplify as follows:

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

and use the probability values from the CPTs.

$$P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e) =$$
$$0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628$$



Procedure for building the network

A procedure for building a Bayesian network which is a good representation of a domain:

1. *Nodes*: Determine the set of variables required to model the domain. Order them: $\{X_1, \dots, X_n\}$

   Ordering influences the result: the resulting network will be more compact if the variables are ordered such that **causes precede effects**.

2. *Links*: For *i* = 1 to *n* do:
   - ✓ Choose, from $X_1, \dots, X_{i-1}$, a minimal set of parents for X_i, such that equation $\mathbf{P}(X_i \mid X_{i-1}, \dots, X_1) = \mathbf{P}(Xi \mid Parents(Xi))$ is satisfied
   - ✓ For each parent insert an arc from the parent to X_i.
   - ✓ CPTs: write down the conditional probability table, $\mathbf{P}(X_i \mid Parents(Xi))$

Note 1: the parents of node X_i should contain all those nodes in $X_1, \dots, X_{i-1}$ that **directly** influence X_i. Example: $\mathbf{P}(M \mid J, A, E, B) = \mathbf{P}(M \mid A)$

Note 2: The network is a DAG by construction and contains no redundant probability values.

Node ordering

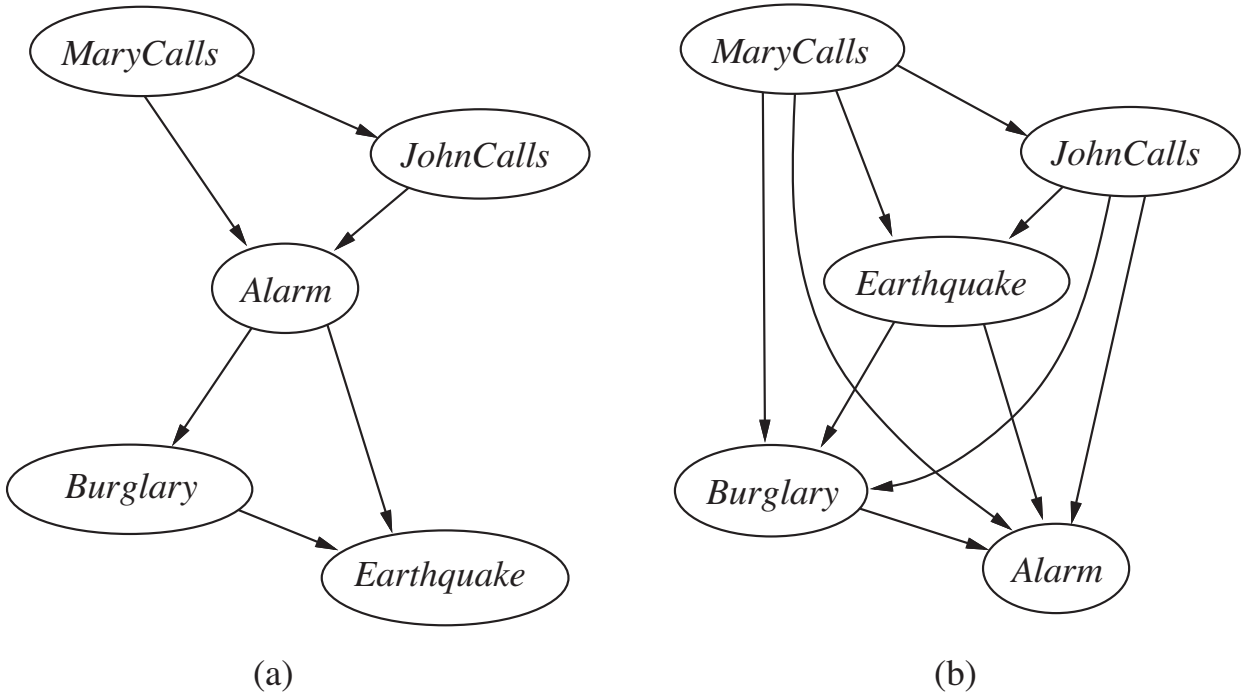
Ordering of variables influences the result:

a. *MaryCalls, JohnCalls, Alarm, Burglary, Earthquake*
b. *MaryCalls, JohnCalls, Earthquake, Burglary, Alarm*

If we use a causal model (with **links from causes to effect**) we obtain a better network, with less connections (more **compact**), fewer probabilities to specify and the numbers will often be easier to obtain.

In **locally structured systems**, each subcomponent interacts directly with only a bounded number of other components.

Huge savings wrt full joint distribution tables



Reduction in complexity (compactness)

- **Locally structured domains** are usually associated with linear rather than exponential growth in complexity.
- In the case of Bayesian networks, it is reasonable to suppose that in most domains each random variable is directly influenced by at most k others, for some constant k.
- If we assume n Boolean variables, then each conditional probability table will have at most 2^k numbers, and the complete network can be specified by $n\,2^k$ numbers. In contrast, the joint distribution contains 2^n numbers.
- Concrete example: n =30 nodes, each with five parents (k=5). The Bayesian network requires 960 numbers, but the full joint distribution requires over a billion.

Conditional independence - 1

We can also extract the **independence assumptions** encoded in the graph structure to do inference.

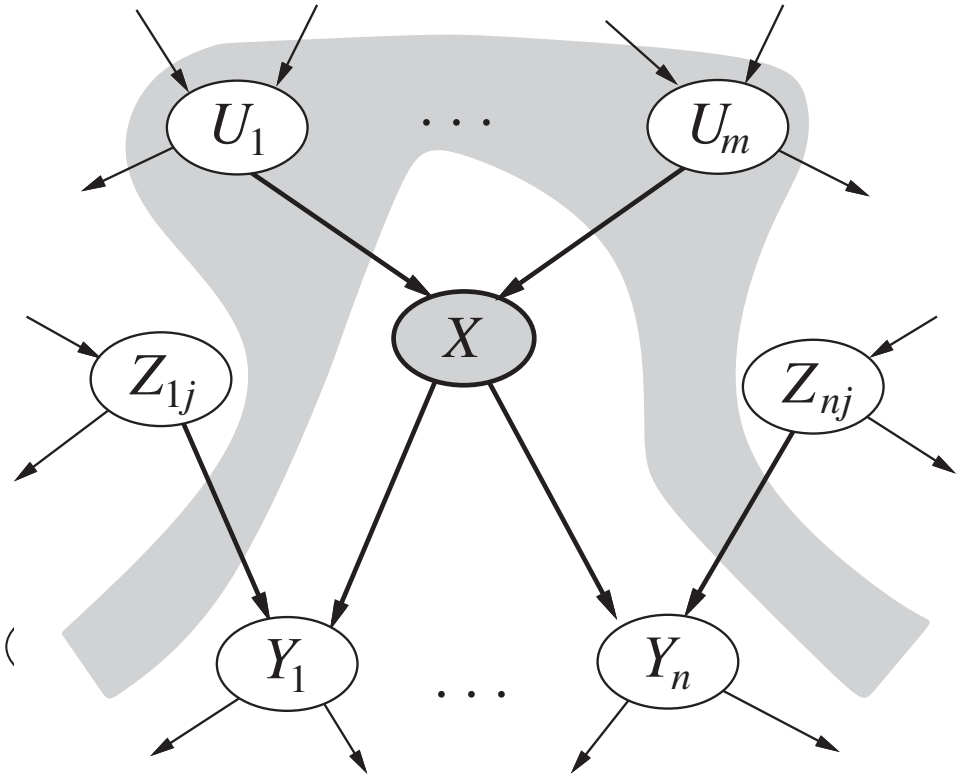
The **topological semantics** specifies that each variable is conditionally independent of its **non-descendants**, given its parents.

In our example, the network tells us that *JohnCalls* is independent of *Burglary, Earthquake,* and *MaryCalls* given the value of *Alarm.*

This authorizes the following simplification.

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

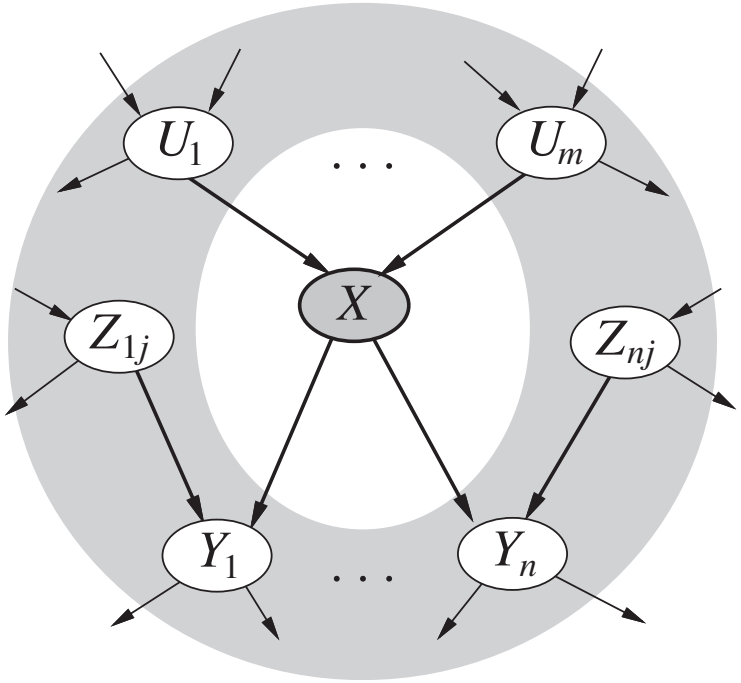
In general see figure.



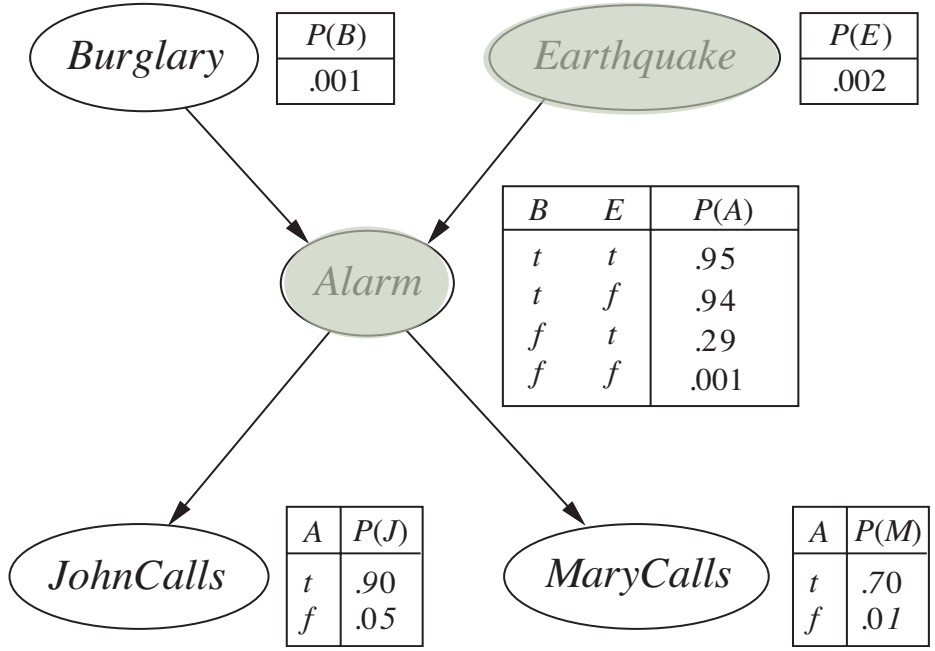
A node X is conditionally independent of its non-descendants (e.g., the Z_{ij} s) given its parents (the U_i s shown in the gray area).

Conditional independence - 2

As a consequence …



A node X is conditionally independent of all other nodes in the network **given** its parents, children, and children's parents, i.e. its **Markov blanket** (the gray area).



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

Efficient construction of CPTs

Often the relationship between a node and its parent follows a **canonical distribution** and the construction of the CPT's can be simplified.

Two examples:

1. **Deterministic nodes**: nodes whose value is specified exactly by the values of their parents

   Example 1. *Canadian, US, Mexican* (the parents) determine *NorthAmerican* (the child).

   This can be seen as a logical OR of the parent. There is no uncertainty.

   Example 2. $Price_1$, $Price_1$, …, $Price_k$ (the parents) determine *min*($Price_1$, $Price_1$, …, $Price_k$) the best price

Efficient construction of CPTs (cnt.)

2. **Noisy-OR relations**: a generalization of the logical OR

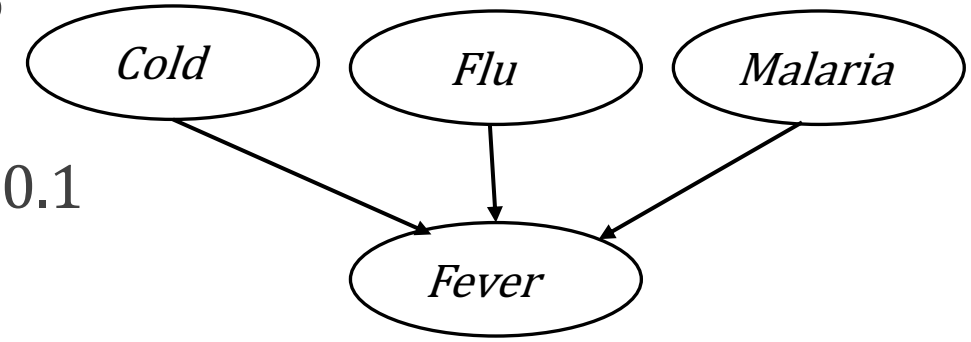
   $Cold \vee Flu \vee Malaria \Leftrightarrow Fever$ but not always …

   You can specify *inhibiting factors* for *Cold* , *Flu*, *Malaria*: e.g. a patient with a cold, could have no fever with a given probability

$q_{\text{cold}} = P(\neg fever \mid \textbf{\textit{cold}}, \neg flu, \neg malaria) = 0.6$

$q_{\text{flu}} = P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$

$q_{\text{malaria}} = P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$



Two assumptions:

- ✓ all the possible causes are listed; if not, we can add a **leak** condition/node.
- ✓ the *inhibiting factor* of each parent is independent of any other parent

Example of *noisy-or*

The 8 probabilities in the CPT of *Fever* can be computed from 3 values as follows.

The probability of $\neg fever$ is the product of the inhibition probabilities q for each *true* parent. It is 1 when none of the causes is true.

The rule is:

$$P(x_i \mid Parents(X_i)) = 1 - \prod_{\{j:\, X_j = \text{true}\}} q_j$$

O(k) instead of O(2^k)

Inhibiting factors:

$q_{\text{cold}} = P(\neg fever \mid cold, \neg flu, \neg malaria) = 0.6$

$q_{\text{flu}} = P(\neg fever \mid \neg cold, flu, \neg malaria) = 0.2$

$q_{\text{malaria}} = P(\neg fever \mid \neg cold, \neg flu, malaria) = 0.1$

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	$0.012 = 0.6 \times 0.2 \times 0.1$

Continuous variables

1. One possible way to handle continuous variables (such as temperature) is to avoid them by using **discrete intervals.**

   Example for temperature: $(<0^{o}\mathrm{C})$, $(0^{o}\mathrm{C}-100^{o}\mathrm{C})$, $(>100^{o}\mathrm{C})$.

2. The most common solution is to define **standard families of probability density functions** that are specified by a **finite number** of parameters.

   Example: a Gaussian (or normal) distribution $N(\mu, \sigma^2)(x)$ with parameters the mean μ and variance σ^2

A network with both discrete and continuous variables is called a **hybrid** Bayesian network.

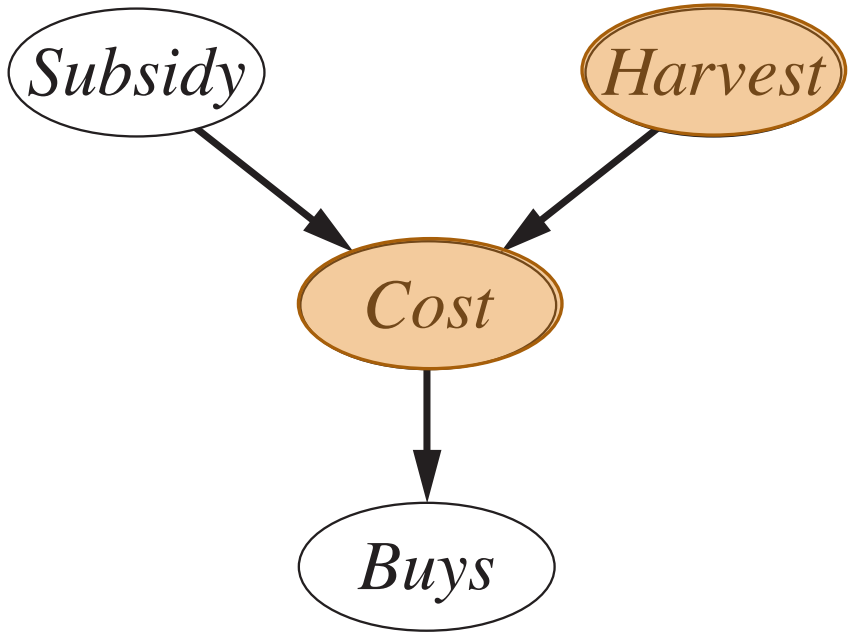
Mixing continuous and discrete variables

Hybrid Bayesian networks combine discrete (*Subsidy, Buys*) and continuous variables (*Harvest, Cost*).

Two new kinds of distributions:

1. the conditional distribution for a continuous variable (*Cost*) given discrete and continuous parents;
2. the conditional distribution for a discrete variable (*Buys*) given continuous parents.

A customer buys some fruit depending on its cost, which depends in turn on the size of the harvest (continuous) and whether the government's subsidy scheme is operating (discrete).



Exact inference in Bayesian networks

INFERENCE BY ENUMERATION – ENUMERATION-ASK – VARIABLE-ELIMINATION

Exact inference in Bayesian networks

The basic task of probabilistic inference. Given:

X: the query variable (we assume one)

E: the set of evidence variables $\{E_1, \ldots, E_m\}$

Y: the set of unknown variables, i.e. the **hidden** (non-evidence) variables $\{Y_1, \ldots, Y_l\}$

Thus **X** $=\{X\} \cup$ **E** $\cup$ **Y**

A typical query asks for the posterior probability distribution, given some evidence:

$\mathbf{P}(X \mid \mathbf{e})$

In the *Alarm* example, the query could be:

$P(Burglary \mid JohnCalls = true, MaryCalls = true)$

Evidence variables: $\mathbf{E} = \{JohnCalls, MaryCalls\}$

Hidden variables: $\mathbf{Y} = \{Earthquake, Alarm\}$

Inference by enumeration

We defined a procedure for the task as:

$\mathbf{P}(X|\mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_y \mathbf{P}(X, \mathbf{e}, \mathbf{y})$ where α is a normalization factor

The query can be answered using a Bayesian network by computing **sums of products** of conditional probabilities from the network. In the example:

$\mathbf{P}(B|j, m) = \alpha\, \mathbf{P}(B, j, m) = \alpha \sum_e \sum_a \mathbf{P}(B, j, m, e, a)$ *summing up over* $\mathbf{y}=\{E, A\}$

We can now use the CPT's tables for computing $P(b, j, m, e, a)$ i.e. the case of $Burglary = true$:

$$P(b|j, m) = \alpha \sum_e \sum_a P(b, j\, m, e, a) = \alpha \sum_e \sum_a P(b)\, P(e)\, P(a|b, e)\, P(j|a)\, P(m|a)$$

We can simplify by factorization to:

$$P(b|j, m) = \alpha\, P(b) \sum_e P(e) \sum_a P(a|b, e)\, P(j|a)\, P(m|a)$$

Expression tree of the computation

$$P(b \mid j, m) = \alpha\, P(b) \sum_e P(e) \sum_a P(a \mid b, e)\, P(j \mid a)\, P(m \mid a)$$

$$= \alpha \times 0.00059224$$

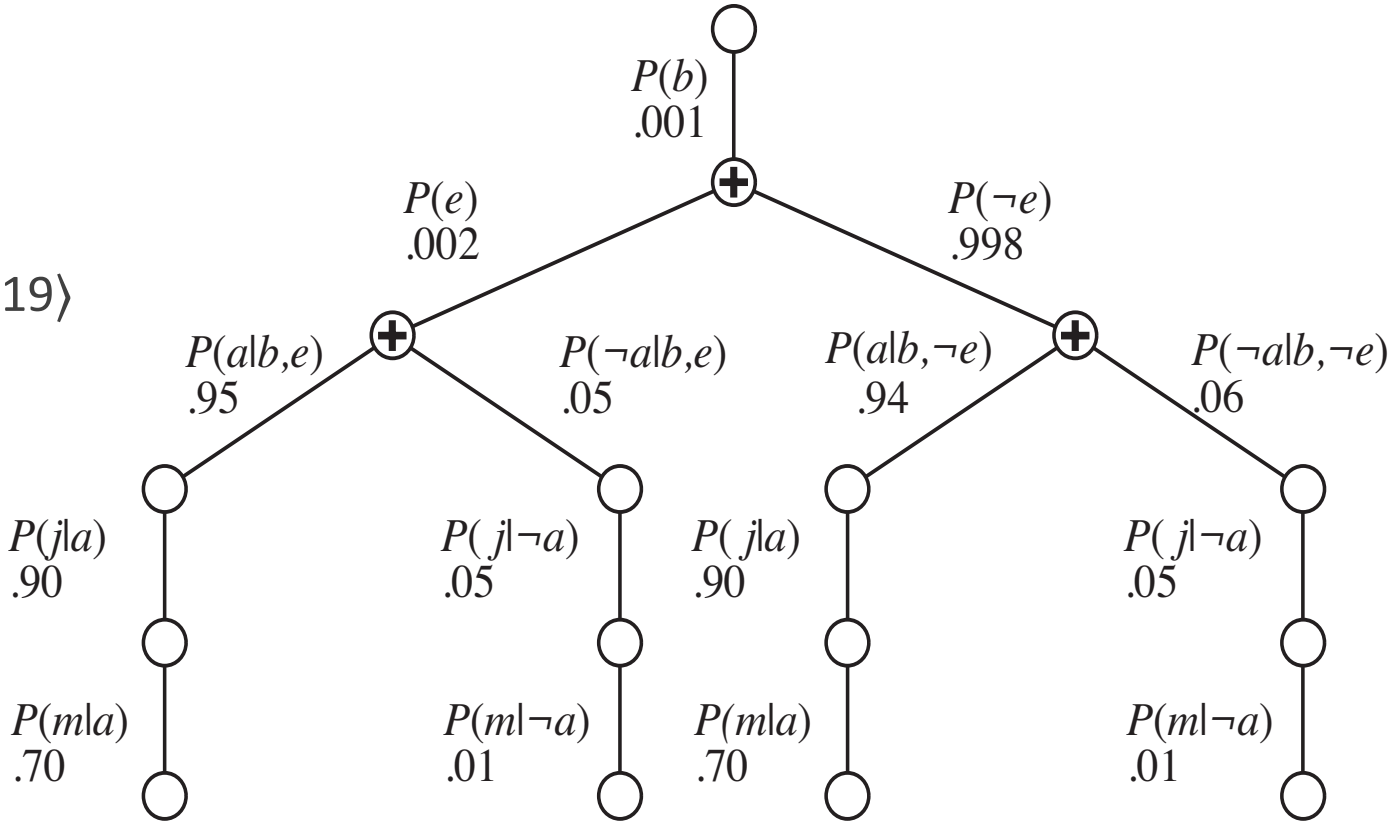
$$P(\neg b \mid j, m) = \alpha \times 0.0014919$$

$$\mathbf{P}(B \mid j, m) = \alpha \langle 0.00059224, 0.0014919 \rangle$$

$$\mathbf{P}(B \mid j, m) \approx \langle 0.284, 0.716 \rangle$$

The tree is evaluated depth first

With *n* Boolean variables,
the time complexity is $O(2^n)$
Space complexity is linear in n



Enumeration-Ask

function ENUMERATION-ASK(X, **e**, bn) **returns** a distribution over X — *Computes* **P**(X | **e**)

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* */

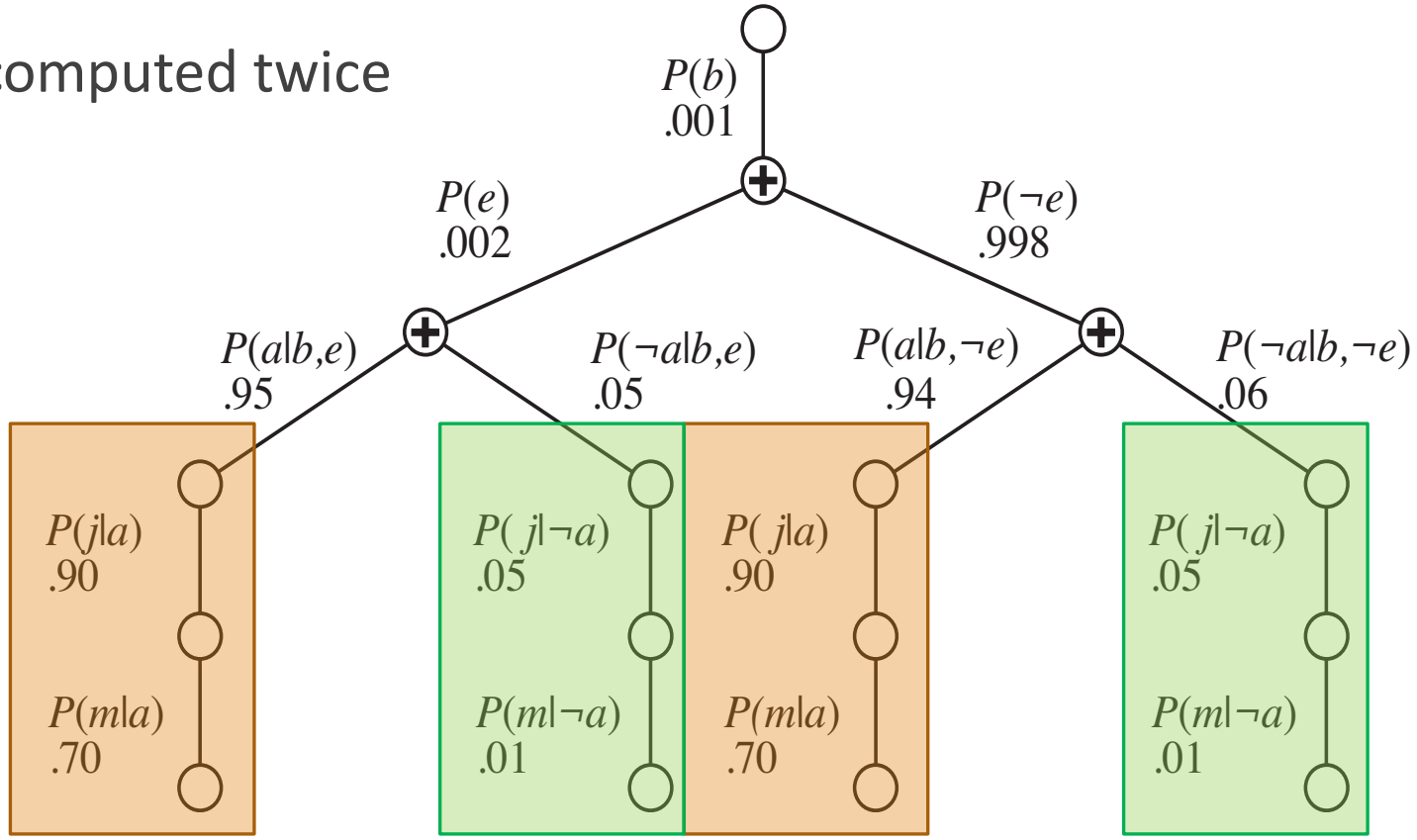
Q(X) ← a distribution over X, initially empty
for each value x_i of X **do**
Q(x_i) ← ENUMERATE-ALL(bn.VARS, $\mathbf{e}_{x_i}$) — *Computes P(xi,* **Y, e** *)*
where $\mathbf{e}_{x_i}$ is **e** extended with $X = x_i$ — *a probability value*
return NORMALIZE(**Q**(X))

---

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

Improving the computation

- $P(j|a)$ $P(m|a)$ is computed twice
- $P(j|\neg a)$ $P(m|\neg a)$ is computed twice



Variable elimination: the idea

The enumeration algorithm can be improved substantially, by storing and reusing the results of repeated computations (a kind of *dynamic programming*).

The **variable elimination algorithm**, proceeds right-to-left (bottom-up).

Conditional probabilities are represented as **factors,** i.e. matrices resulting from conditional probabilities; the names make explicit the argument.

For the *Alarm* network:

$$P(B \mid j, m) = \alpha\, P(b) \sum_e P(e) \sum_a P(a \mid b, e)\; P(j \mid a)\; P(m \mid a)$$

$$P(B \mid j, m) = \alpha\, f_1(B) \times \sum_e f_2(E) \times \sum_a f_3(A, B, E) \times f_4(\mathrm{A}) \times f_5(\mathrm{A})$$

1. How factors are constructed?
2. Two operations *on factors*: **pointwise-product** (×) and **summing out** variables.

Constructing *factors* from CPT's and evidence

Given a variable and some evidence, a factor can be built by looking at the variable part of CPTS's in the network: MAKE-FACTOR(var, e)

Examples:

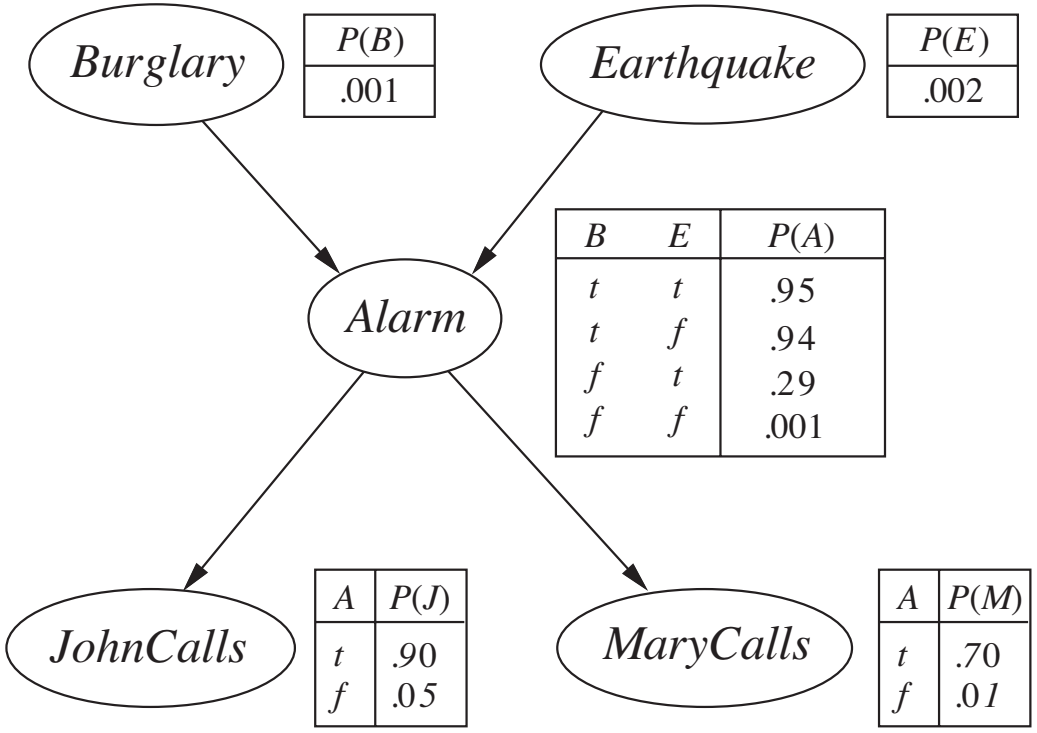
- MAKE-FACTOR$(M, \{j, m\}) = f_5(A)$ because M only depends on A, m is an evidence.

  $f_5(A) = \langle P(m|\, a), P(m\,|\neg a)\rangle = \langle 0.7, 0.01\rangle$
- MAKE-FACTOR $(J, \{j, m\}) = \langle P(j|\, a), P(j\,|\neg a)\rangle =$

  $= f_4(A)$ only depends on $A = \langle 0.9, 0.05\rangle$
- MAKE-FACTOR$(A, \{j, m\}) = f_3(A, B, E)$

  = CPT of A, a 2x2x2 matrix

  …



Pointwise product of factors

The pointwise product of two factors f_1 and f_2 yields a new factor f_3 such that:

- variables are the union of the variables in f_1 and f_2 and
- elements are given by the product of the corresponding values in the two factors.

Computation of the pointwise product $f_1(A, B) \times f_2(B, C)$ gives $f_3(A, B, C)$

Note: The result is not necessarily a probability distribution

A	B	$\mathbf{f}_1(A,B)$
T	T	.3
T	F	.7
F	T	.9
F	F	.1

B	C	$\mathbf{f}_2(B,C)$
T	T	.2
T	F	.8
F	T	.6
F	F	.4

A	B	C	$\mathbf{f}_3(A,B,C)$
T	T	T	$.3 \times .2 = .06$
T	T	F	$.3 \times .8 = .24$
T	F	T	$.7 \times .6 = .42$
T	F	F	$.7 \times .4 = .28$
F	T	T	$.9 \times .2 = .18$
F	T	F	$.9 \times .8 = .72$
F	F	T	$.1 \times .6 = .06$
F	F	F	$.1 \times .4 = .04$

Summing out variables

Summing out variable A in factor f_3:

$$\sum_a f_3(A, B, C) = f_3(a, B, C) + f_3(\neg a, B, C) = \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix}$$

Computational saving comes from the realization that any factor that does not depend on the variable to be summed out can be moved outside the summation.

For example:

$$\sum_e f_2(E) \times f_3(A, B, E) \times f_4(A) \times f_5(A) = f_4(A) \times f_5(A) \times \sum_e f_2(E) \times f_3(A, B, E)$$

This operation of "distributing out" common factors is part of the operation, which in the general case returns a set of factors.

The computation

Let's go back to the problem of computing:

$$P(B \mid j, m) = \alpha f_1(B) \times \sum_e f_2(E) \times \sum_a f_3(A, B, E) \times f_4(\mathrm{A}) \times f_5(\mathrm{A})$$

1. We compute the product of f_3, f_4, and f_5. Summing out on A gives us a new 2 x 2 factor $f_6(B, E)$ whose indices range over just B and E:

   $$f_6(B, E) = (f_3(a, B, E) \times f_4(a) \times f_5(a)) + (f_3(\neg a, B, E) \times f_4(\neg a) \times f_5(\neg a))$$

   $$P(B \mid j, m) = \alpha f_1(B) \times \sum_e f_2(E) \times f_6(B, E)$$

2. Next, we sum out E from the product of f_2 and f_6:

   $$P(B \mid j, m) = \alpha f_1(B) \times f_7(B)$$

   Take the pointwise product and normalize the result.

Variable elimination: the algorithm

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $factors \leftarrow []$
  for each var **in** ORDER(bn.VARS) **do**
    $factors \leftarrow [\text{MAKE-FACTOR}(var, \mathbf{e}) | factors]$
    if var is a hidden variable **then** $factors \leftarrow$ SUM-OUT($var, factors$)
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

Computes **P**(X | **e**)

Elimination ordering is important

Add new factor to factors

- MAKE-FACTOR creates a factor for each variable, given the evidence;
- SUM-OUT sums out over the possible values of a hidden variable, producing a set of factors; it takes care of distributing out factors which do not depend on the variable

Variable ordering

Every choice of ordering yields a valid algorithm, but **different orderings of variables can produce differences in efficiency**. For example we could have decided to change the order in which A and E are eliminated (inverting the order of summations):

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

$$P(b \mid j, m) = \alpha\, f_1(B) \times \sum_{a} f_4(A) \times f_5(A) \sum_{e} f_2(E) \times f_3(A, B, E)$$

$$P(b \mid j, m) = \alpha\, f_1(B) \times \sum_{a} f_4(A) \times f_5(A) \times f_6(A, B)$$

Determining the optimal ordering is intractable, but several good heuristics are available, for example eliminate whichever variable minimizes the size of the next factor to be constructed.

Variable relevance

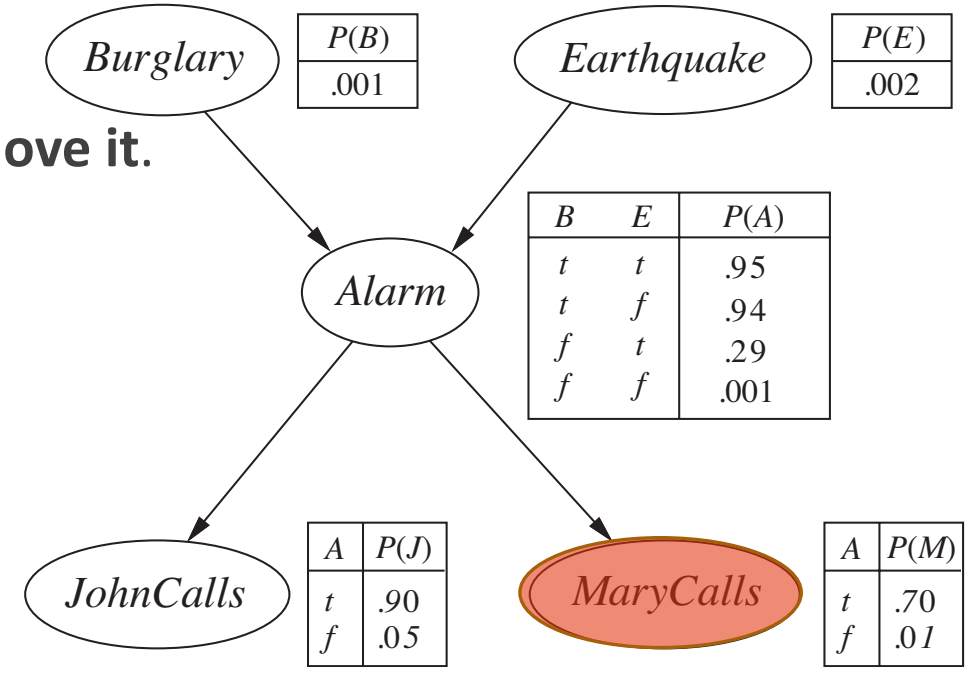
Let's compute the probability distribution of "John calls" given that there was a burglary.

$$P(j|\, b) = \alpha\, P(b) \sum_e P(e) \sum_a P(a|\, b, e)\, P(j|a) \sum_m P(m|a)$$

Note: $\sum_m P(m|a) = P(m|a) + P(\neg m|a) = 1$

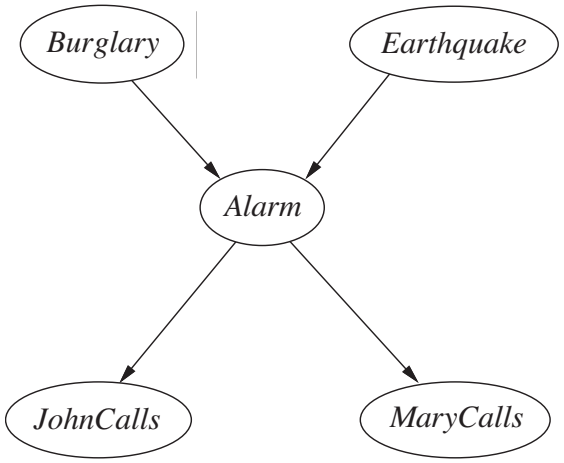
In fact M is **irrelevant** to the query and **we can remove it**.

- ✓ We can remove any leaf node that is not a query variable or an evidence variable.
- ✓ Continuing this process, *we can remove any variable that is not an ancestor of a query variable or evidence variable*.

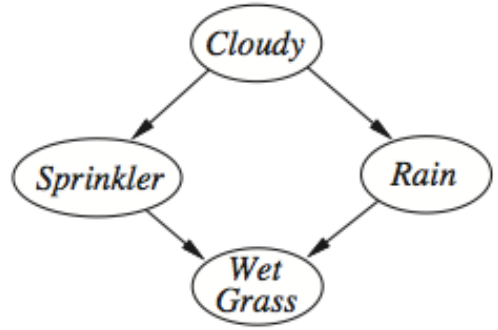


The complexity of exact inference

- The complexity of exact inference in Bayesian networks depends strongly on the structure of the network.
- **Singly connected networks** or **polytrees** are such that there is at most one undirected path between any two nodes. The *Alarm* network is a polytree.
- The time and space complexity of exact inference in polytrees is **linear** in the size of the network (the number of CPT entries).
- For **multiply connected networks** variable elimination can have exponential time and space complexity in the worst case.
- **In fact, i**nference in Bayesian networks includes as a special case propositional inference, which is NP-complete.

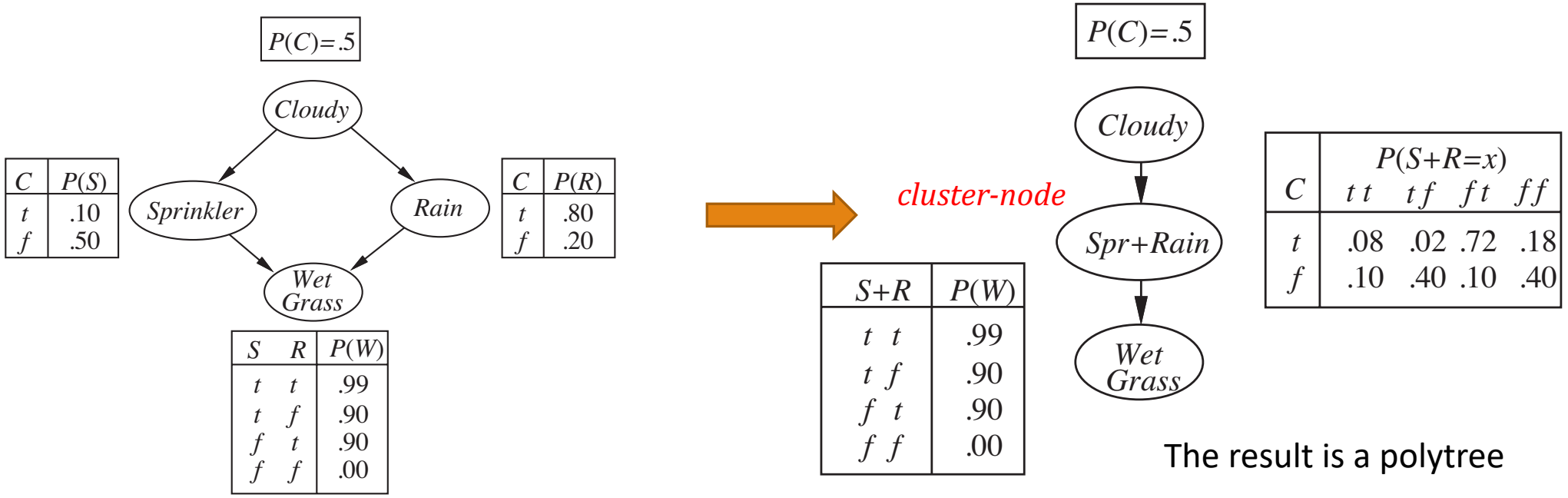


Singly connected network



Multiply connected network

Clustering/join trees algorithms



Mega-nodes correspond to variables (*Spr+Rain*) with combined values (*tt, tf, ft, ff*)

Probability values are obtained as follows from the *Sprinkler* and *Rain* CPT's:

1^{st} *row*: $P(S{+}R = xy \mid c) = P(S{=}x \mid c) \times P(R{=}y \mid c)$ for $x, y \in \{t, f\}$.

2^{nd} *row*: $P(S{+}R{=}xy \mid \neg c) = P(S{=}x \mid \neg c) \times P(R{=}y \mid \neg c)$

Caveat: mega-nodes may share variables, and this makes the algorithms more complex.

Conclusions

- ✓ We have explored **Bayesian networks** as a natural and concise way to represent conditional independence assumptions.
- ✓ Their semantics in terms of join probability distribution.
- ✓ How to perform efficient probabilistic inference with this networks: two specific algorithms for exact inference: **Enumeration-Ask** and **Elimination-Ask**.
- ✓ Given the intractability of exact inference in large, multiply connected networks, it is essential to consider alternative methods.
- ✓ We leave to other courses:
  - stochastic approximate inference methods (logic sampling, likelihood weighting, …)
  - dealing with continuous distributions
  - Bayesian learning

References

- ✓ Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach* (3rd edition). Pearson Education 2010 [Cap 14 – Probabilistic reasoning]
- ✓ David Barber, *Bayesian Reasoning and Machine Learning,* Online version February 2017 (Ch. 2)