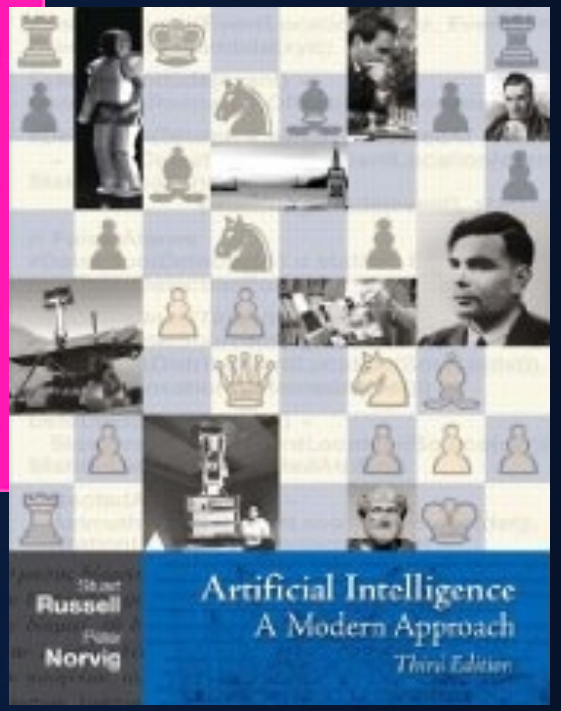


AI Fundamentals: uncertain reasoning

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
- Other approaches to uncertainty

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. A set of directed links or arcs connects pairs of nodes. If there is an arrow 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_i has an associated conditional probability distribution $\mathbf{P}(X_i \mid Parents(X_i))$ that quantifies the effect of the parents on the node.

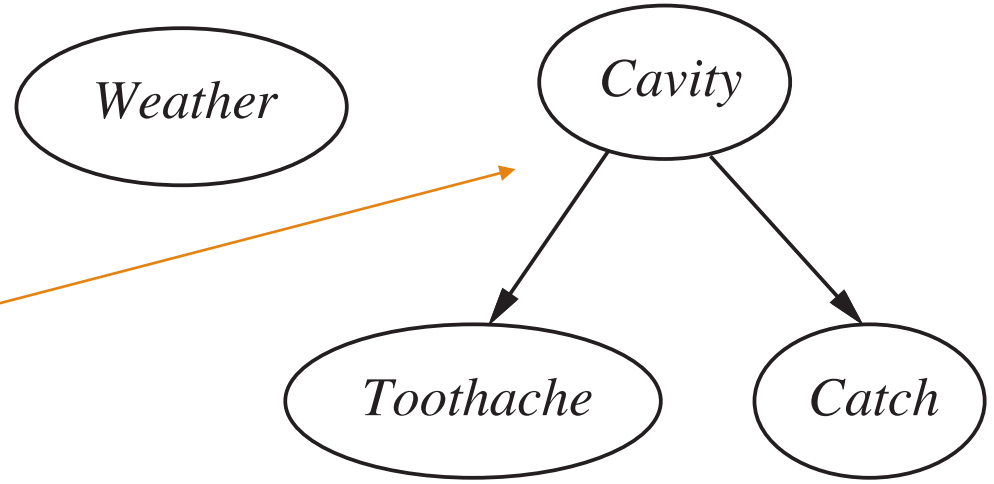
Easier for domain experts to decide what direct influences exist in the domain than actually specifying the probabilities themselves.

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**.



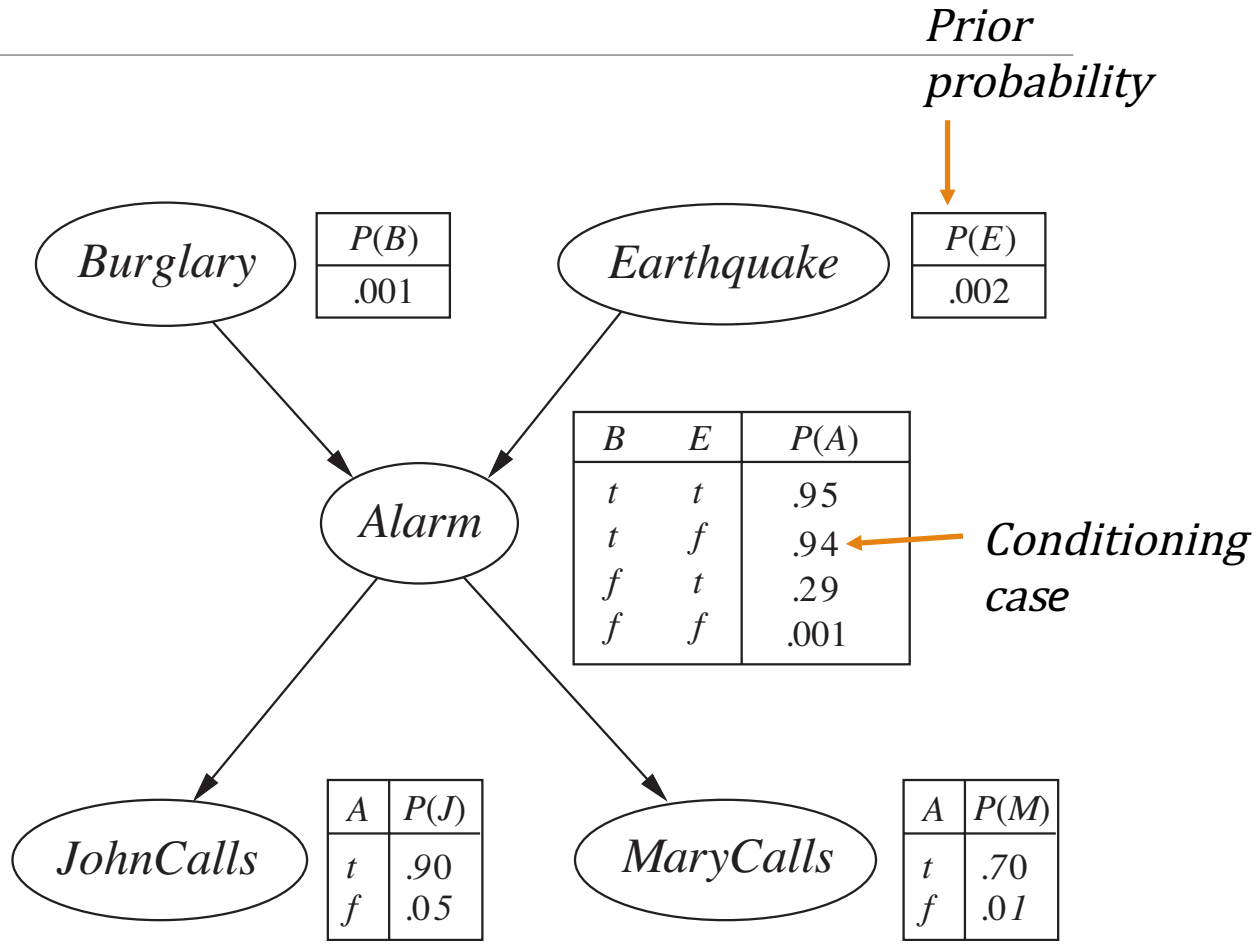
The *alarm* example

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

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.

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 equivalent ways to look at the semantics of Bayesian networks:

1. 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 - 1

We explain why 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**:

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

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

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.

Semantics of the networks - 1

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

Computing the following quantity from the network

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

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

is like computing $P(x_1, \dots, x_n)$ provided:

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

Under these assumptions 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.

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

Representing the joint probability distribution

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

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

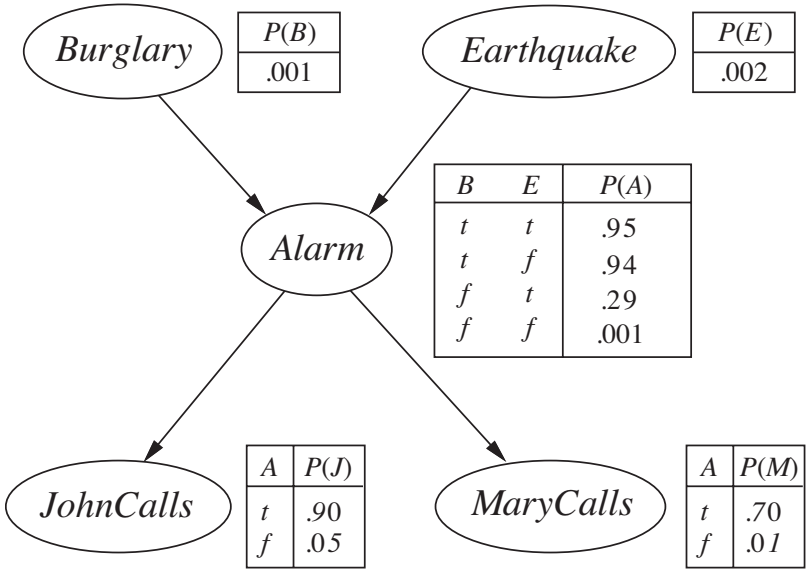
where $parents(Xi)$ denotes the values of $Parents(Xi)$ that appear in $x_1, \ldots, x_n$.

Example. Given the *Alarm* network we can compute:

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

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

$0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628$



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: $\{X_1, \ldots, X_n\}$

   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 $X_1, \ldots, X_{i-1}$, a minimal set of parents for X_i, such that equation

     $\mathbf{P}(X_i \mid X_{i-1}, \ldots, X_1) = \mathbf{P}(Xi \mid Parents(Xi))$ is satisfied
   - For each parent insert a link from the parent to X_i.
   - CPTs: Write down the conditional probability table, $\mathbf{P}(X_i \mid Parents(Xi))$

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

Node ordering and compactness

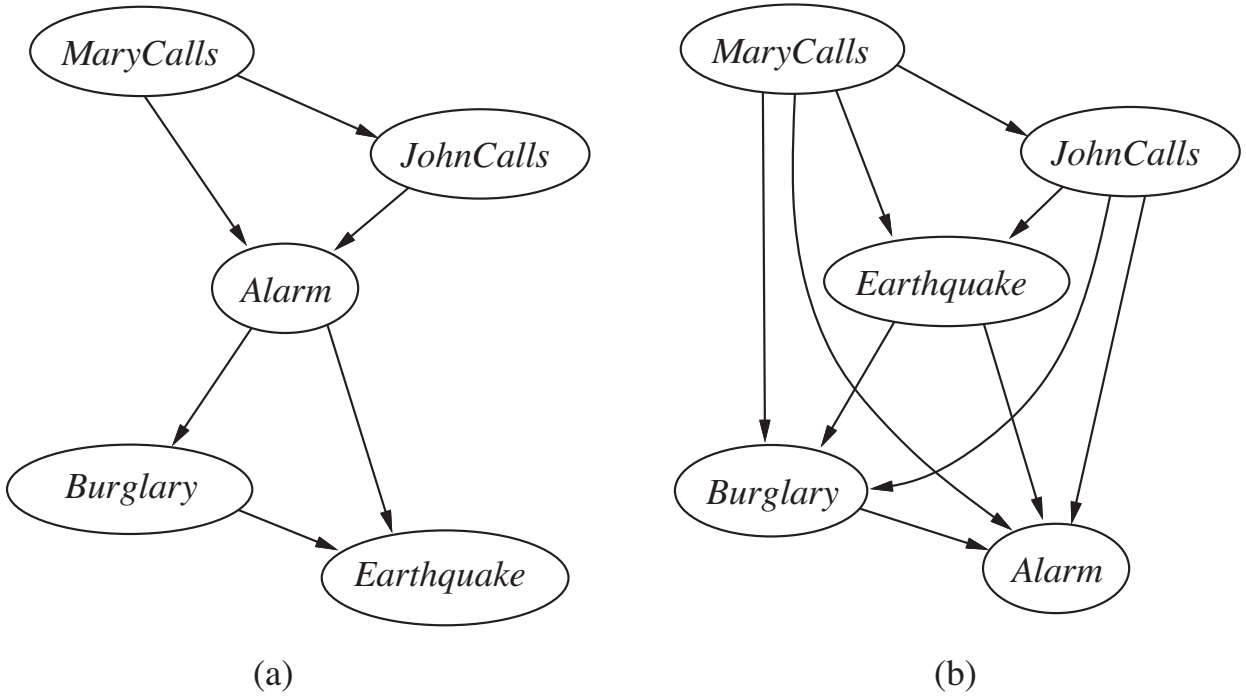
Ordering influences the result:

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

- Local structure is 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.

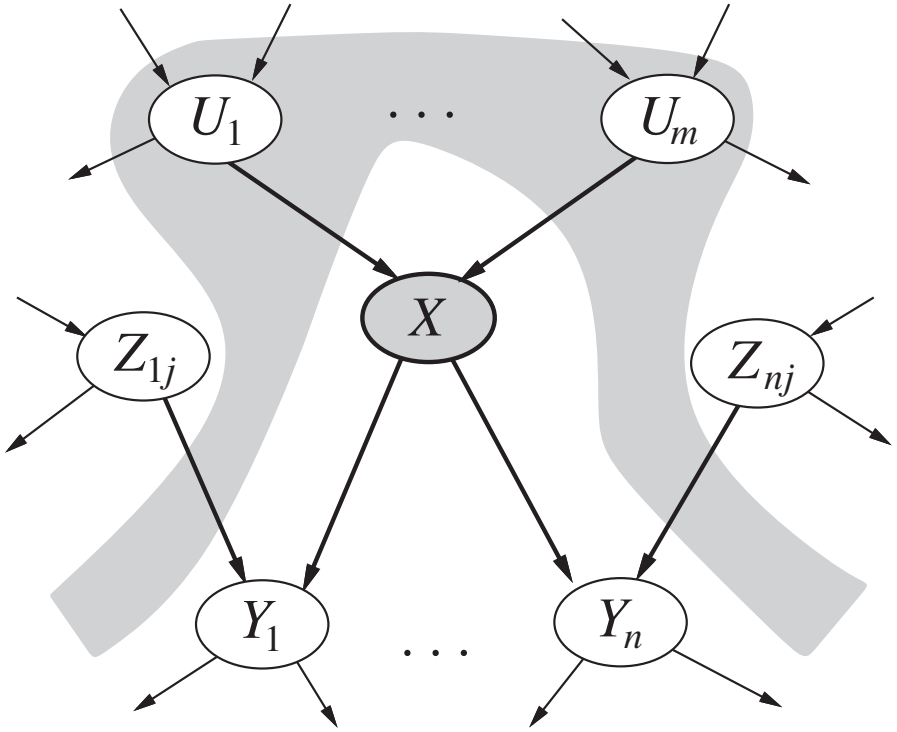
Conditional independence - 1

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.

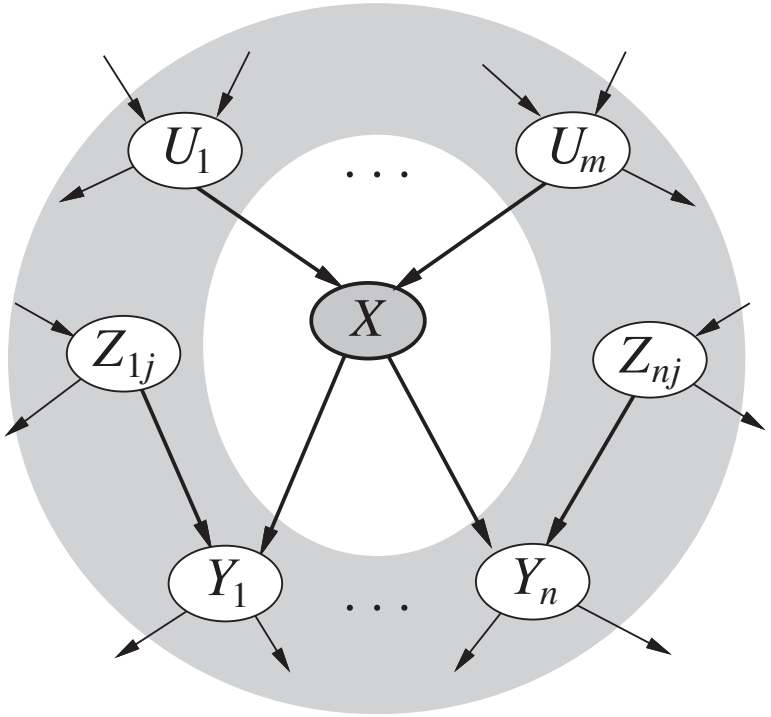
In our examples example, *JohnCalls* is independent of *Burglary, Earthquake,* and *MaryCalls* given the value of *Alarm.*

In general see figure.

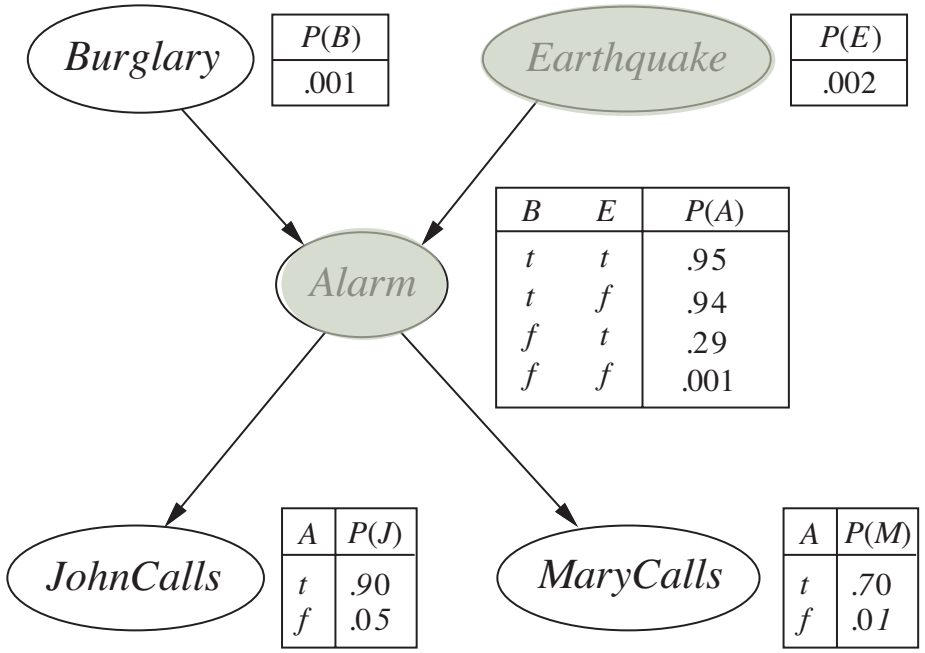


A node X is conditionally independent of its non-descendants (e.g., the Z_{ij} s) given its parents (the U_i s shown in the gray area).

Conditional independence - 2



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 representation of CPTs

The idea: Instead of filling CPT's use "canonical" distributions.

Two examples:

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

   $Canadian, US, Mexican$ (the parents) determine $\underline{NorthAmerican}$ (the child)

   $Price_1, Price_1, \ldots, Price_k$ (the parents) determine $\underline{min}(Price_1, Price_1, \ldots, Price_k)$

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

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

   The causal relationship between parent and child may be **inhibited,** with a given P.
   Example: A patient could have a cold, but not a fever with probability P=0.6

   Two assumptions:
   - We assume all the possible causes are listed; if not, we can add a **leak** condition/node.
   - We assume that inhibition factor of each parent is independent of any other parent

Example of *noisy-or*

Assume the following inhibiting P's:

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

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

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

fever is *false* iff all its ***true*** **parents** are inhibited, and the probability of this is the product of the inhibition probabilities q for each *true* parent.

The general rule is:

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

The 8 probabilities in the CPT are computed from 3 values.

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

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 **discretization**.

   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

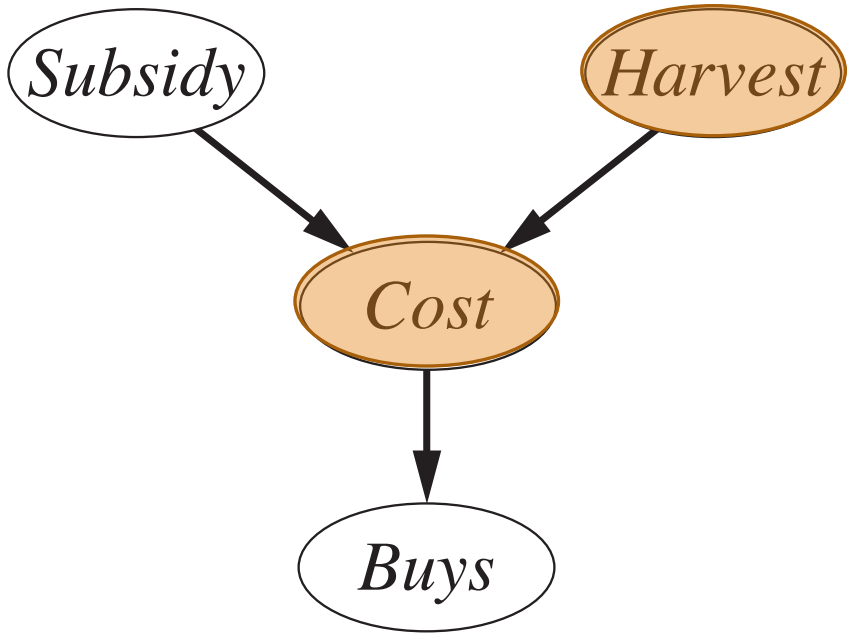
A network with both discrete and continuous variables is called a **hybrid** Bayesian network.

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 given discrete or continuous parents;
2. the conditional distribution for a discrete variable given continuous parents.



A 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

Exact inference in Bayesian networks

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 $\mathbf{X} = \{X\} \cup \mathbf{E} \cup \mathbf{Y}$

A typical query asks for the posterior probability distribution:

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

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

$P(Burglary \mid JohnCalls, MaryCalls)$

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

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

Inference by enumeration

We defined 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* **y**

We can now use the CPT's tables for computing $P(b\,|j, m)$, 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:

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

Computing with the network

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

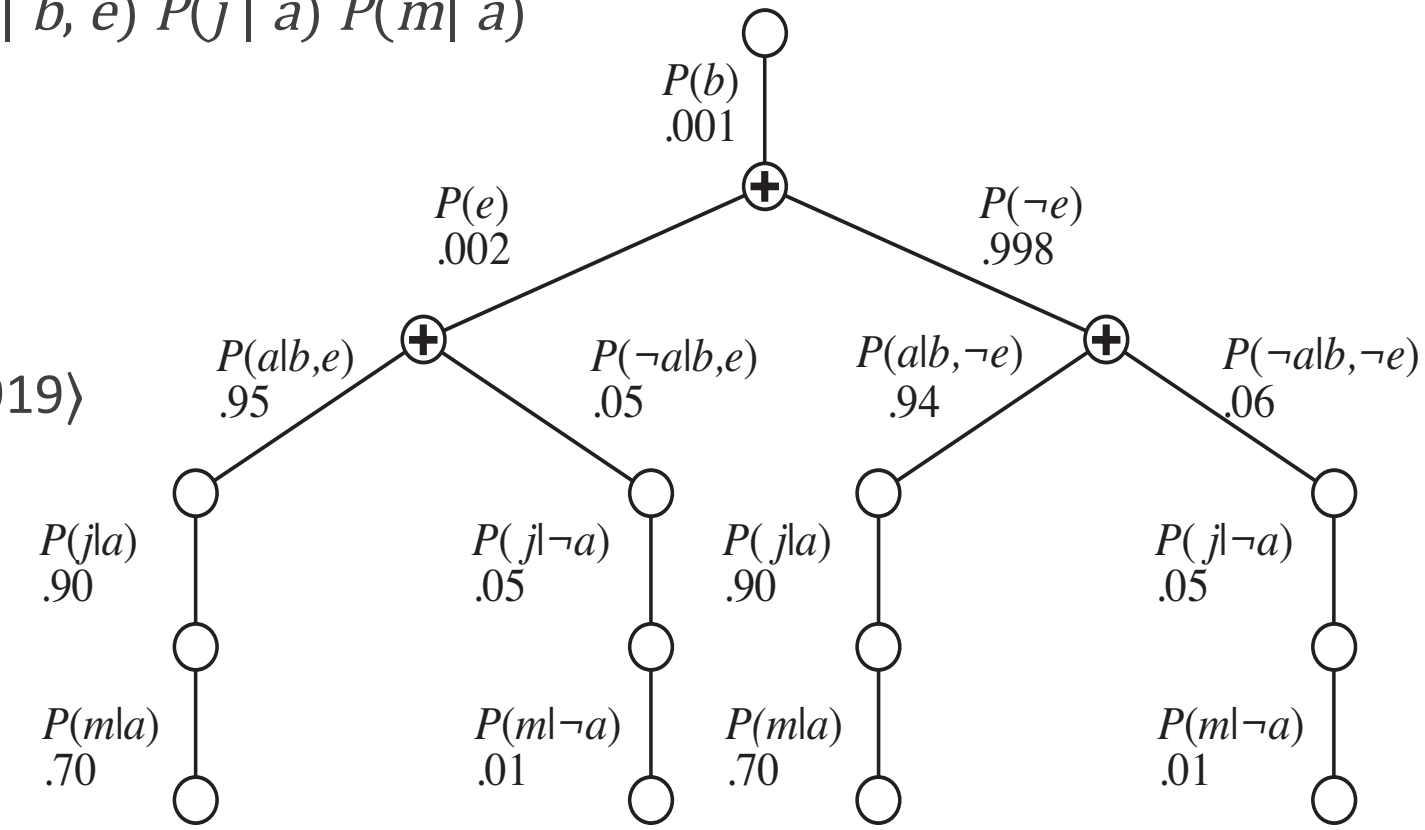
$$= \alpha \times 0.00059224$$

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

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

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

Time complexity is $O(2^n)$.



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))

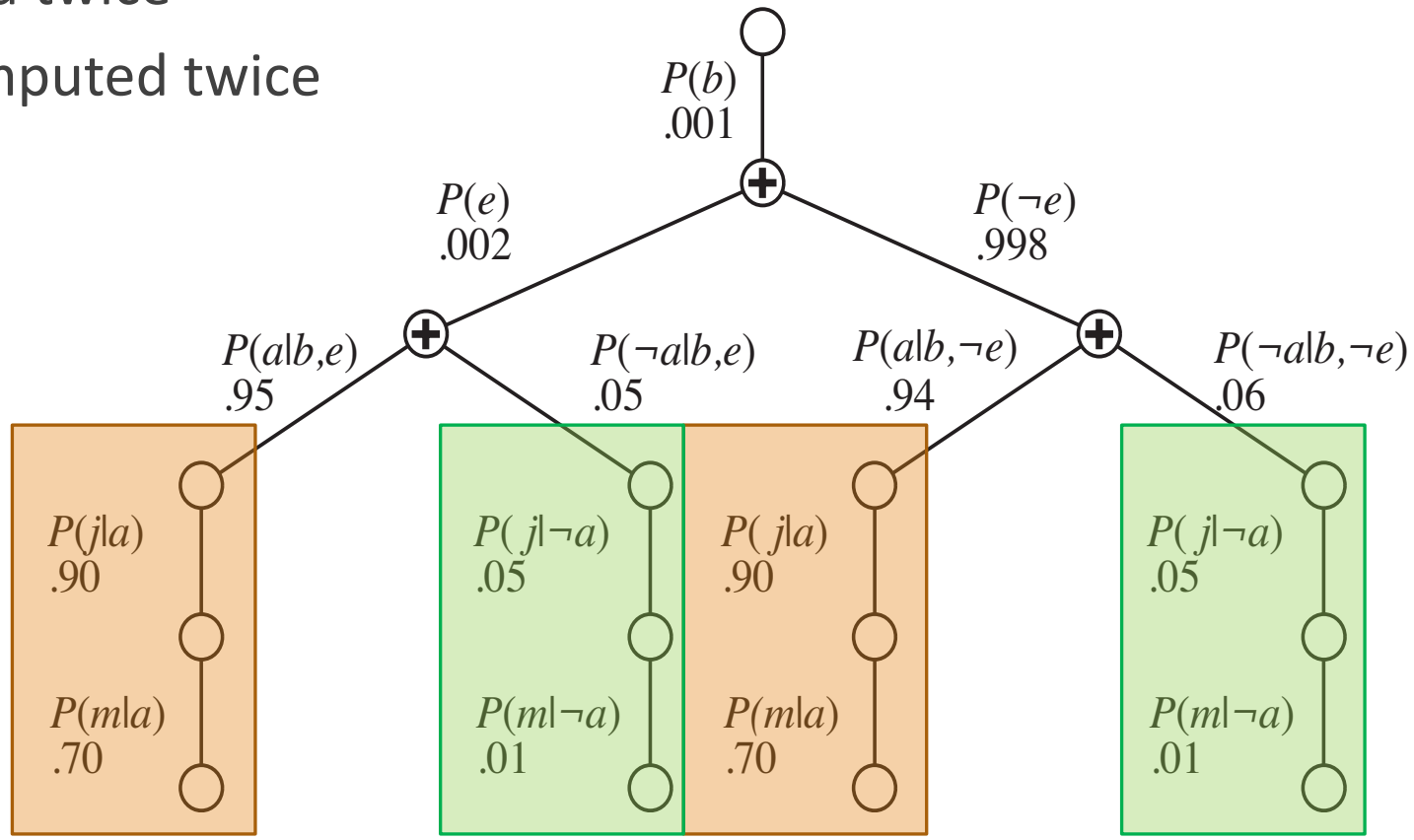
---

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**
then return $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), **e**) — *evidence variable or X*
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).

For the *Alarm* network:

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

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

Each factor *f* is a matrix indexed by the values of its argument variables

j and *m* are given, thus $f_4(\mathrm{A}) = \langle P(j \mid a), P(j \mid \neg a) \rangle$ and $f_5(\mathrm{A}) = \langle P(m \mid a), P(m \mid \neg a) \rangle$

We need now to define two operations *on factors*: **pointwise-product** (indicated by ×) and **summing out** variables.

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 $f_3(A, B, C)$

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	$.3 \times .2 = .06$
T	F	.7	T	F	.8	T	T	F	$.3 \times .8 = .24$
F	T	.9	F	T	.6	T	F	T	$.7 \times .6 = .42$
F	F	.1	F	F	.4	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:

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

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 sum out A from the product of f_3, f_4, and f_5. This gives us a new 2 x 2 factor $f_6(B, E)$ whose indices range over just B and E:

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

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

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

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

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($factors$))

Ordering is important

Add new factor to factors

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 invert summations.

Changing the order in which A and E are eliminated:

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

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

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

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.

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

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

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

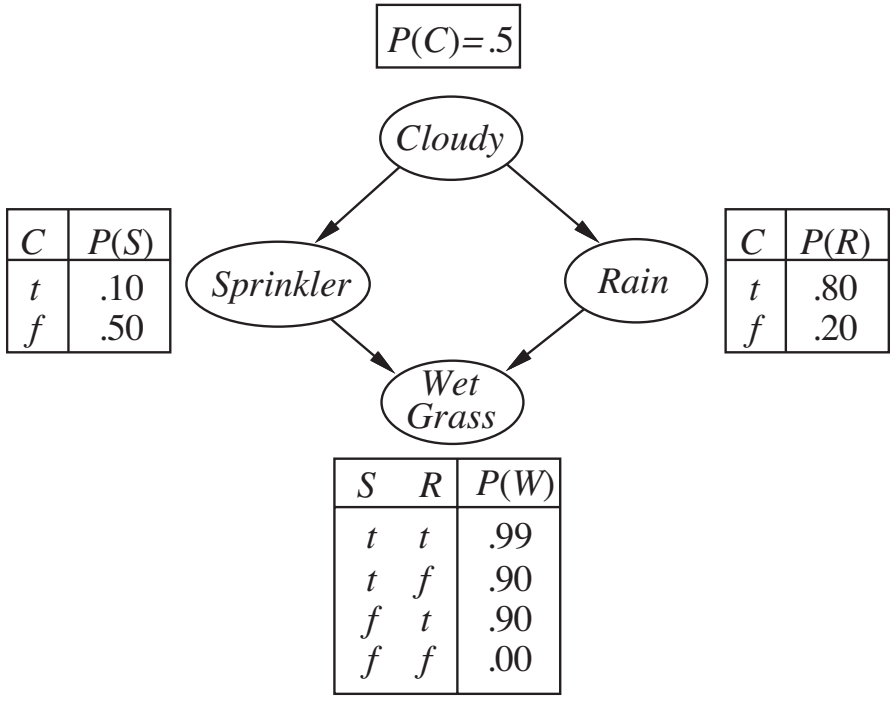
Singly connected networks or **polytrees** are such that there is at most one undirected path between any two nodes

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).

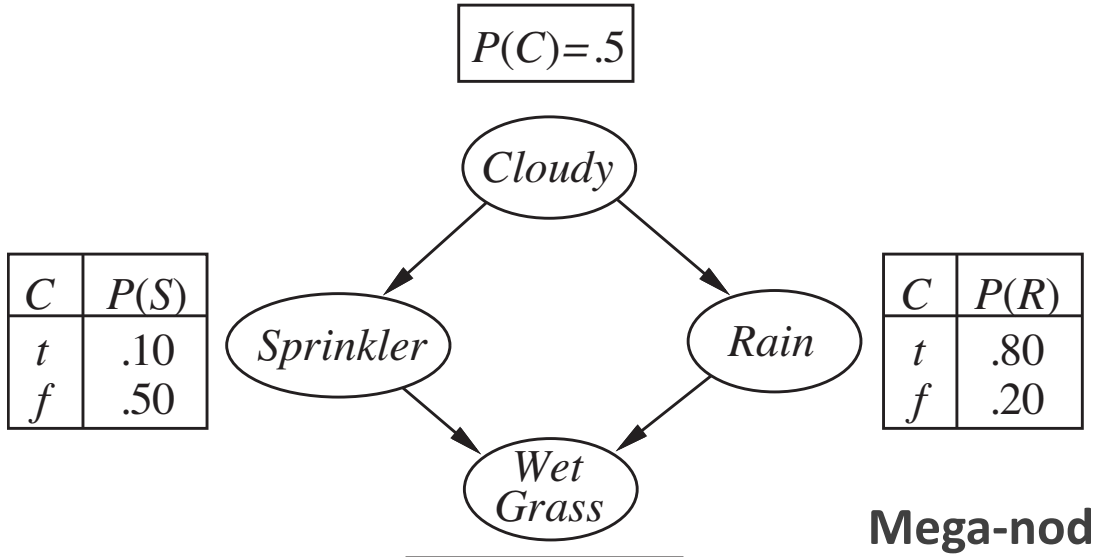
For **multiply connected networks** variable elimination can have exponential time and space complexity in the worst case.

Note: Inference in Bayesian networks includes as a special case propositional inference, which is NP-complete.



Multiply connected network

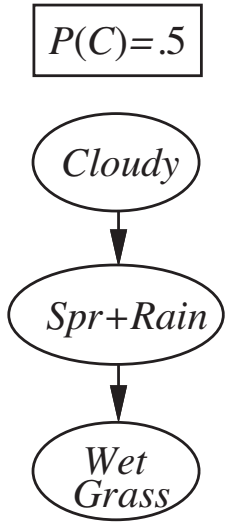
Clustering/joint trees



C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00



S+R	P(W)
t t	.99
t f	.90
f t	.90
f f	.00

C	P(S+R=x)			
	t t	t f	f t	f f
t	.08	.02	.72	.18
f	.10	.40	.10	.40

The result is a polytree

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

Probability values are obtained this way from the S & R CPT's:

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

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

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

Other approaches to uncertain reasoning

An overview of other approaches

Probability theory and Bayesian networks are the dominant approaches today, despite the disadvantage of having to specify many probability values (lots of numbers).

Humans reason with numbers?

Other approaches have been used:

1. Rule-based methods for uncertain reasoning
2. Representing ignorance: Dempster–Shafer theory
3. Representing vagueness: fuzzy sets and fuzzy logic

Rule-based methods for uncertain reasoning

Three good properties of classical logic based rules:

1. **Locality**: In logical systems, from A and A $\Rightarrow$ B, we can conclude B, without worrying about any other rules. In probabilistic systems, we need to consider all the evidence.
2. **Detachment**: Once B is proved, it can be used regardless of how it was derived. It can be *detached* from its justification.
3. **Truth-functionality**: the truth of complex sentences can be computed from the truth of the components. Probability combination does not work this way.

The idea is to attach **degree of belief** to facts and rules and to combine and propagate them

The most famous example is the **certainty factors model**, which was developed for the MYCIN medical diagnosis program. The system was carefully engineered to avoid mixing different kind of rules (diagnostic vs causal) leading to non plausible results.

Representing ignorance: Dempster–Shafer theory

Introduces the distinction between **uncertainty** and **ignorance**.

Instead of computing the probability of an event uses a function $Bel(\mathrm{p})$ to represent the probability that the evidence supports p.

Example:

Not knowing whether a coin is fair $Bel(\mathrm{head}) = 0$ and $Bel(\mathrm{tail}) = 0$.
Knowing that the coin is fair with probability 0.9 then $Bel(\mathrm{head}) = 0.9 \text{ x } 0.5 = 0.45$, similarly for tail

Instead of assigning probabilities to **possible worlds**, the theory assigns masses to **sets of possible worlds**, that is, to events.

Representing vagueness: fuzzy sets and fuzzy logic

Fuzzy set theory is a means of specifying how well an object satisfies a vague (non categorical) property, like “being tall”.

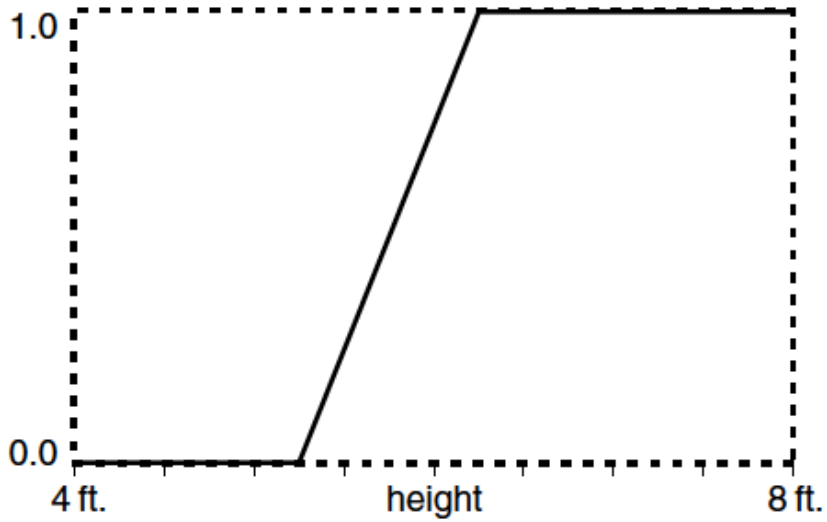
Tall is a fuzzy predicate and says that the truth value of $Tall(john)$ is a number between 0 and 1, rather than being just *true* or *false.*

A fuzzy predicate implicitly defines a set that does not have sharp boundaries (a **fuzzy set**).

Fuzzy logic is a method for reasoning with logical expressions describing membership in fuzzy sets.

Example. $\mathrm{T}(Tall(john) \wedge Heavy(john)) =$

$$= \mathrm{MIN}(\mathrm{T}(Tall(john)), \mathrm{T}(Heavy(john))$$



Relation between degree of ‘tallness’ and height

Conclusions

- ✓ We have explored Bayesian networks as a natural an concise way to represent conditional independence assumptions.
- ✓ Their semantics in terms of join probability distribution.
- ✓ How to perform efficient probabilistic inference with this networks.
- ✓ A specific algorithm for exact inference: **variable elimination**.
- ✓ 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

Your turn

- ✓ Build a Bayesian network for a new example and compute some inference. See for example exercises at the end of AIMA, ch 14.
- ✓ Study the AIMA-code for the Enumeration algorithm, run it on an example and report.
- ✓ Study the AIMA-code for the Variable Elimination algorithms, run it on an example and report.
- ✓ First order probability models (AIMA)
- ✓ Fuzzy sets and fuzzy logic
- ✓ Dempster–Shafer theory

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)