

# PROBABILISTIC REASONING SYSTEMS

*In which we explain how to build reasoning systems that use network models to reason with uncertainty according to the laws of probability theory.*

## Outline

- Knowledge in uncertain domains
- Probabilistic Networks
- Semantic of Bayesian Networks
  - Global Semantic
  - Local Semantic
- Summary

2

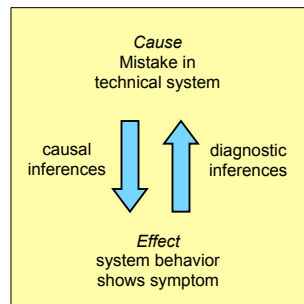
## Bayes'-Rule

Why is this rule useful?

- Causal experiences  
C: cause, E: effect
- Diagnostic Inference

$$P(C | E) = \frac{P(E | C) P(C)}{P(E)}$$

This simple equation underlies all modern AI systems for probabilistic inference



Bayes' Rule

3

## Knowledge in uncertain domains

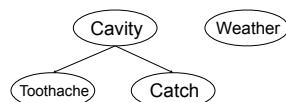
- Joint probability distribution
  - delivers answers to questions that exists in domain
  - Problem: intractable with large number of variables
  - Specification Probabilities difficult for atom events
- Complexity
  - Independence and conditional dependence reduce complexity
- Bayesian Networks
  - Data structure represents dependencies between variables
  - Specification of joint distribution

Knowledge in uncertain domains

4

## Syntax

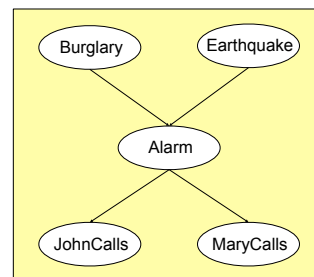
- Graph-theoretical structure
  - Set of variables as nodes (discrete, continuous)
  - One node per variable
  - Directed acyclic graph (DAG), links express causal dependencies between variables
- Conditional probability tables
  - For each node a table for conditional probabilities
  - Table consists of distribution of probabilities given their parents  $P(X_i | \text{Parents}(X_i))$



Probabilistic Networks

5

## Simple Bayesian Network • Example Alarm

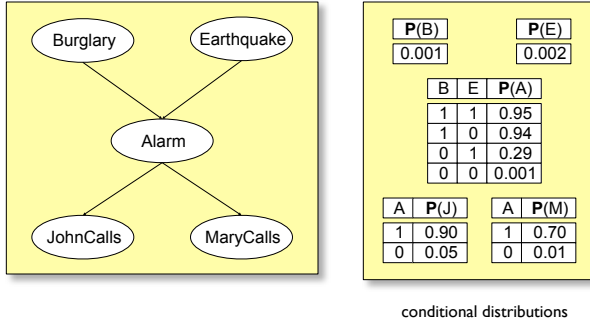


- new burglar alarm fairly reliable at detecting a burglary
- also responds on occasion to minor earthquake
- two neighbors, John and Mary have promised to call when they hear the alarm
- John always calls when he hears alarm, but sometimes confuses the telephone ringing with the alarm and calls then too
- Mary likes loud music and sometimes misses the alarm altogether
- Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

Probabilistic Networks

6

## Simple Bayesian Network



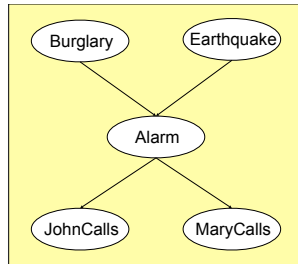
## Semantics of Bayesian Networks

- Two views on semantics
  1. Global Semantics: The first is to see the network as a representation of the joint probability distribution
  2. Local Semantics: The second is to view it as an encoding of a collection of conditional independence statements
- Views are equivalent
  - first helpful in understanding how to construct networks
  - second helpful in designing inference procedures

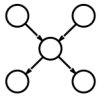
## Representing the full joint distribution

- General idea
  - Joint distribution can be expressed as product of local conditional probabilities
  - Every entry in the joint probability distribution can be calculated from the information in the network.
  - Generic entry

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(X_i))$$



## Representing the full joint distribution



- Example
  - Alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Mary call
  - $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$   
 $= P(j|a) P(m|a) P(a|\neg b, \neg e) P(\neg b) P(\neg e)$   
 $= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$   
 $\approx 0.00063$

<b>P(B)</b>		<b>P(E)</b>	
0.001		0.002	
<b>B</b>	<b>E</b>	<b>P(A)</b>	
1	1	0.95	
1	0	0.94	
0	1	0.29	
0	0	0.001	
<b>A</b>	<b>P(J)</b>	<b>A</b>	<b>P(M)</b>
1	0.90	1	0.70
0	0.05	0	0.01

## Method for Constructing Bayesian Networks

- Generic Rule
  - Semantic but: not how to construct a network
  - Implicitly: conditional independence  
→ Help for Knowledge Engineer
- Reformulate the rule
  - Use of conditional probabilities  
→ Product rule
- Repeat process
  - Reduction of conjunctive probabilities to a conditional dependency and a smaller conjunction
  - Final: a big product

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

## Chain rule

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

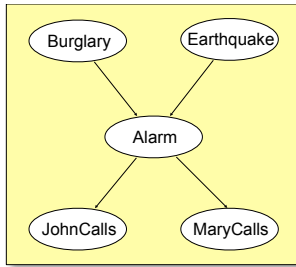
- Compare with
  - $P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(X_i))$
  - reveals that specification is equivalent to general assertion
- I.e.:
  - This last condition is satisfied by labeling the nodes in any order that is consistent with the partial order implicit in the graph structure.
  - The 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.

$$P(X_i | X_{1, \dots, i-1}) = P(X_i | \text{Parents}(X_i))$$

$$(\text{as long as } \text{Parents}(X_i) \subseteq \{X_1, \dots, X_{i-1}\})$$

## Construction of Bayesian Networks

- Important while constructing
  - We need to choose parents for each node such that this property holds.
- Intuitive
  - Parents of node  $X_i$  should contain all those nodes in  $X_1, \dots, X_{i-1}$  that directly influence  $X_i$
  - Example.:
    - M (is influenced by B or E but not directly)
    - Influenced by A, and J calls are not evident



$$P(M|J,A,E,B) = P(M|A)$$

## General Procedure

- Choose the set of relevant variables  $X_i$  that describe the domain.
- Choose an ordering for the variables.  
(Any ordering works, but some orderings work better than others, as we will see.)
- While there are variables left:
  - Pick a variable  $X_i$  and add a node to the network for it.
  - Set  $Parents(X_i)$  to some minimal set of nodes already in the net such that the conditional independence property is satisfied.
  - Define the conditional distribution  $P(X_i|Parents(X_i))$

## Notes

- Construction method guarantees that the network is acyclic
  - Because each node is connected only to earlier nodes, this.
- Redundancies
  - No redundant probability values
  - Exception: for one entry in each row of each conditional probability table, if  $P(x_2|x_1)$   $P(\neg x_2|x_1)$  is redundant
- This means that it is impossible for the knowledge engineer or domain expert to create a Bayesian network that violates the axioms of probability!*

## Compactness

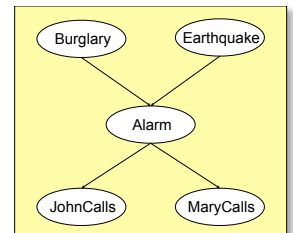
- Compactness
  - A Bayesian Network is a complete and not-redundant representation of a domain
  - Can be more compact as a joint distribution
  - This is important in practice
  - Compactness is an example for property that we call in local structure (our sparse coded) in general
- Local Structures
  - Each sub-component is connected to a limited number of other components
  - Complexity: linear instead of exponential
  - With BN: in most domains one variable is influenced by  $k$  others, with  $n$  variables  $2^k$  conditional probabilities, the whole network  $n2^k$
  - In contrast, the joint contains  $2^n$  numbers

## Node Ordering

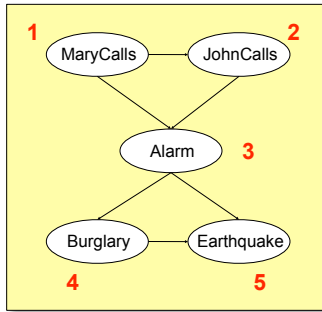
- Local structures (example)
  - 30 nodes, each max. 5 parents
  - 960 for BN, > 1 billion with joint distribution
- Construction
  - not trivial
  - Variable directly influenced only from a few others
  - set parent node "appropriately"
  - Network topology
  - "direct influencers" first
  - Thus: correct order important
- Order
  - Add:
    - root first
    - then direct influencers
    - then down to leaves
  - What happens with "wrong" order?

## Example ordering

- Let us consider the burglary example again.
- Suppose we decide to add the nodes in the order
  - M, J, A, B, E
  - M, J, E, B, A



## Example ordering



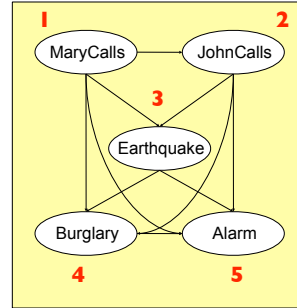
Better: stick with causal model, less numbers!

Global Semantics

- Order
  - M,J,A,B,E
  - add M: no parents
  - add J: if Mary calls, that probably means the alarm has gone off, which would make it more likely that John calls.  
→ dependency: J needs M as parent
  - add A: both as parents
  - add B: only Alarm
  - add E: A and B
- Network
  - two more links, three more probabilities
  - “weak, unnatural” relations (e.g. E given A and B)
  - Reason: causal vs diagnostic model

19

## Example ordering (2)



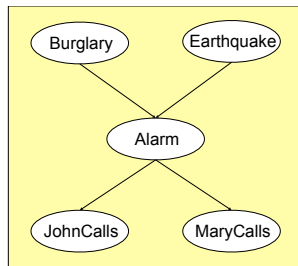
Global Semantics

- Order
  - M,J,E,B,A
- Network
  - 31 probabilities
  - like joint distribution
  - thus: bad choice
- All three networks represent same probability distribution
- Last two versions
  - simply fail to represent all the conditional independence relationships
  - end up specifying a lot of unnecessary numbers instead.

20

## Conditional independence relations in Bayesian networks

- Before
  - “numerical (global) semantics with probability distribution
  - from this derive conditional independencies
- Idea now
  - opposite direction: topological (local) semantics
  - specifies conditional independencies
  - from this derive numerical semantics



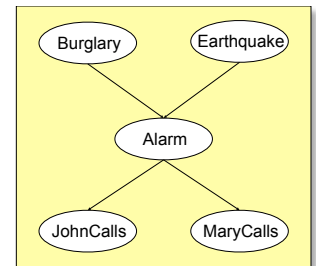
- Example
  - J is independent of M given A, i.e.
  - $P(J|M,A) = P(J|A)$

Local Semantics

21

## Conditional independence relations in Bayesian networks

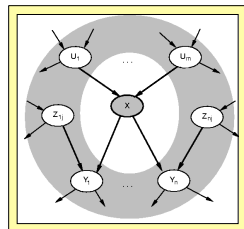
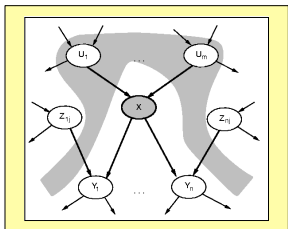
- General idea
  - A node is conditionally independent of its non-**descendants** given its parents
  - A node is conditionally independent of all other nodes in the network given its parents, children, and children's parents—that is, given its **Markov blanket**
- Example
  - B is independent of J and M given A and E, i.e.
  - $P(B|A,E,J,M) = P(B|A,E)$



Local Semantics

22

## Conditional independence relations in Bayesian networks



- Node X is conditionally independent of its non-descendants (e.g., the  $Z_{ij}$ s) given its parents (the  $U_{ij}$ s)
- A node X is conditionally independent of all other nodes in the network given its Markov blanket.

Local Semantics

23

## Compact conditional distributions

CPT grows exponentially with number of parents  
CPT becomes infinite with continuous-valued parent or child

Solution: **canonical** distributions that are defined compactly

**Deterministic** nodes are the simplest case:

$$X = f(\text{Parents}(X)) \text{ for some function } f$$

E.g., Boolean functions

$$\text{NorthAmerican} \Leftrightarrow \text{Canadian} \vee \text{US} \vee \text{Mexican}$$

E.g., numerical relationships among continuous variables

$$\frac{\partial \text{Level}}{\partial t} = \text{inflow} + \text{precipitation} - \text{outflow} - \text{evaporation}$$

Efficient Representation of conditional distributions

24

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

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

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

## Compact conditional distributions

Noisy-OR distributions model multiple noninteracting causes

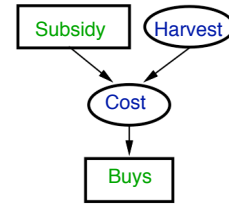
- 1) Parents  $U_1 \dots U_k$  include all causes (can add **leak node**)
- 2) Independent failure probability  $q_i$  for each cause alone  
 $\Rightarrow P(X|U_1 \dots U_j, \neg U_{j+1} \dots \neg U_k) = 1 - \prod_{i=1}^j q_i$

Cold	Flu	Malaria	$P(\text{Fever})$	$P(\neg \text{Fever})$
F	F	F	<b>0.0</b>	1.0
F	F	T	0.9	<b>0.1</b>
F	T	F	0.8	<b>0.2</b>
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	<b>0.6</b>
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	$0.012 = 0.6 \times 0.2 \times 0.1$

Number of parameters **linear** in number of parents

## Bayesian nets with continuous variables

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



- Option 1: discretization—possibly large errors, large CPTs  
 Option 2: finitely parameterized canonical families

- 1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
- 2) Discrete variable, continuous parents (e.g., *Buys?*)

## Continuous child variables

Need one **conditional density** function for child variable given continuous parents, for each possible assignment to discrete parents

Most common is the **linear Gaussian** model, e.g.,:

$$P(\text{Cost} = c | \text{Harvest} = h, \text{Subsidy?} = \text{true})$$

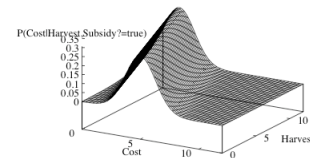
$$= N(a_t h + b_t, \sigma_t)(c)$$

$$= \frac{1}{\sigma_t \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right)$$

Mean *Cost* varies linearly with *Harvest*, variance is fixed

Linear variation is unreasonable over the full range  
 but works OK if the **likely** range of *Harvest* is narrow

## Continuous child variables

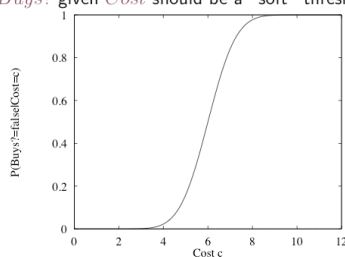


All-continuous network with LG distributions  
 $\Rightarrow$  full joint distribution is a multivariate Gaussian

Discrete+continuous LG network is a **conditional Gaussian** network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

## Discrete variable w/ continuous parents

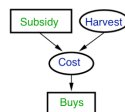
Probability of *Buys?* given *Cost* should be a “soft” threshold:



Probit distribution uses integral of Gaussian:

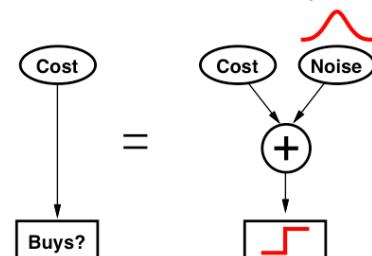
$$\Phi(x) = \int_{-\infty}^x N(0, 1)(x) dx$$

$$P(\text{Buys?} = \text{true} | \text{Cost} = c) = \Phi((-c + \mu)/\sigma)$$



## Discrete variable w/ continuous parents

1. It's sort of the right shape
2. Can view as hard threshold whose location is subject to noise

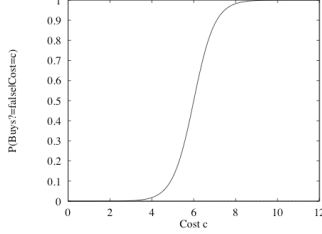


## Discrete variable w/ continuous parents

Sigmoid (or logit) distribution also used in neural networks:

$$P(\text{Buys?} = \text{true} \mid \text{Cost} = c) = \frac{1}{1 + \exp(-2\frac{-c+\mu}{\sigma})}$$

Sigmoid has similar shape to probit but much longer tails:



Efficient Representation of conditional distributions

31

## Inference tasks

**Simple queries:** compute posterior marginal  $P(X_i | E = e)$

e.g.,  $P(\text{NoGas} | \text{Gauge} = \text{empty}, \text{Lights} = \text{on}, \text{Starts} = \text{false})$

**Conjunctive queries:**  $P(X_i, X_j | E = e) = P(X_i | E = e)P(X_j | X_i, E = e)$

**Optimal decisions:** decision networks include utility information; probabilistic inference required for  $P(\text{outcome} | \text{action}, \text{evidence})$

**Value of information:** which evidence to seek next?

**Sensitivity analysis:** which probability values are most critical?

**Explanation:** why do I need a new starter motor?

Exact inference by enumeration

32

## Enumeration algorithm

**function ENUMERATION-ASK**( $X, e, bn$ ) **returns** a distribution over  $X$

**inputs:**  $X$ , the query variable  
 $e$ , observed values for variables  $E$   
 $bn$ , a Bayesian network with variables  $\{X\} \cup E \cup Y$

$Q(X) \leftarrow$  a distribution over  $X$ , initially empty

**for each** value  $x_i$  of  $X$  **do**

    extend  $e$  with value  $x_i$  for  $X$

$Q(x_i) \leftarrow \text{ENUMERATE-ALL}(\text{VARS}[bn], e)$

**return**  $\text{NORMALIZE}(Q(X))$

**function ENUMERATE-ALL**( $\text{vars}, e$ ) **returns** a real number

**if**  $\text{EMPTY?}(\text{vars})$  **then return** 1.0

$Y \leftarrow \text{FIRST}(\text{vars})$

**if**  $Y$  has value  $y$  in  $e$

**then return**  $P(y | Pa(Y)) \times \text{ENUMERATE-ALL}(\text{REST}(\text{vars}), e)$

**else return**  $\sum_y P(y | Pa(Y)) \times \text{ENUMERATE-ALL}(\text{REST}(\text{vars}), e_y)$

        where  $e_y$  is  $e$  extended with  $Y = y$

Exact inference by enumeration

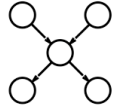
33

## Inference by enumeration

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

Simple query on the burglary network:

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



Rewrite full joint entries using product of CPT entries:

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

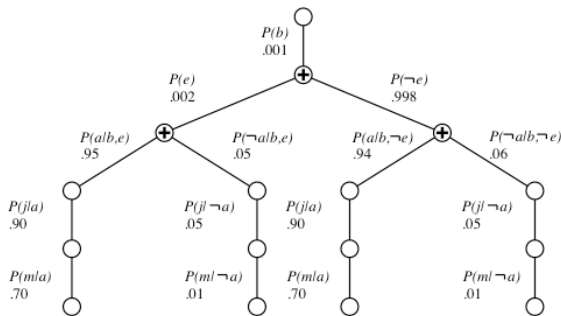
Recursive depth-first enumeration:  $O(n)$  space,  $O(d^n)$  time

$$P(B|j, m) = \alpha(0.00059224, 0.0014919) \approx (0.284, 0.716)$$

Exact inference by enumeration

34

## Evaluation tree



Exact inference by enumeration

35

## Inference by variable elimination

Variable elimination: carry out summations right-to-left, storing intermediate results (**factors**) to avoid recomputation

$$\begin{aligned} P(B|j, m) &= \alpha \underbrace{P(B)}_B \underbrace{\sum_e P(e)}_E \underbrace{\sum_a P(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M \\ &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a) \\ &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a) \\ &= \alpha P(B) \sum_e P(e) \sum_a f_{AJM}(a, b, e) f_J(a) f_M(a) \\ &= \alpha P(B) \sum_e P(e) f_{AJM}(b, e) \text{ (sum out } A) \\ &= \alpha P(B) f_{EAJM}(b) \text{ (sum out } E) \\ &= \alpha f_B(b) \times f_{EAJM}(b) \end{aligned}$$

Exact inference by enumeration

36

## Factors

$$\mathbf{P}(B|j, m) = \alpha \underbrace{\mathbf{P}(B)}_B \sum_e \underbrace{P(e)}_E \sum_a \underbrace{\mathbf{P}(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M$$

Notice that we have annotated each part of the expression with the name of the associated variable; these parts are called **factors**. The steps are as follows:

- The factor for  $M$ ,  $P(m|a)$ , does not require summing over  $M$  (because  $M$ 's value is already fixed). We simply store the probability given each value of  $a$  in a 2-element vector, which we call  $\mathbf{f}_M(A)$ :

$$\mathbf{f}_M(A) = \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix}$$

(In general, each factor we compute will be subscripted with the names of the variables that have been processed to produce it.)

- Similarly, we store the factor for  $J$  as a two-element vector  $\mathbf{f}_J(A)$ .
- The factor for  $A$  is  $\mathbf{P}(a|B, e)$ , which will be a  $2 \times 2 \times 2$  matrix  $\mathbf{f}_A(A, B, E)$ .

Exact inference by enumeration

37

## Factors

$$\mathbf{P}(B|j, m) = \alpha \underbrace{\mathbf{P}(B)}_B \sum_e \underbrace{P(e)}_E \sum_a \underbrace{\mathbf{P}(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M$$

- Now we must sum out  $A$  from the product of these three factors. This will give us a  $2 \times 2$  matrix whose indices range over just  $B$  and  $E$ . We put a bar over  $A$  in the name of the matrix to indicate that  $A$  has been summed out:

$$\begin{aligned} \mathbf{f}_{\bar{A}JM}(B, E) &= \sum_a \mathbf{f}_A(A, B, E) \times \mathbf{f}_J(A) \times \mathbf{f}_M(A) \\ &= \mathbf{f}_A(a, B, E) \times \mathbf{f}_J(a) \times \mathbf{f}_M(a) \\ &\quad + \mathbf{f}_A(\neg a, B, E) \times \mathbf{f}_J(\neg a) \times \mathbf{f}_M(\neg a) \end{aligned}$$

Notice that the multiplication process is a **pointwise product**—that is, multiplication of corresponding entries—rather than standard matrix multiplication.

- We process  $E$  in the same way: sum out  $E$  from the product of  $\mathbf{f}_E(E)$  and  $\mathbf{f}_{\bar{A}JM}(B, E)$ :

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

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

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

Exact inference by enumeration

38

## Variable elimination: Basic operations

**Summing out** a variable from a product of factors:

move any constant factors outside the summation

add up submatrices in pointwise product of remaining factors

$$\sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_{\bar{x}}$$

assuming  $f_1, \dots, f_i$  do not depend on  $X$

**Pointwise product** of factors  $f_1$  and  $f_2$ :

$$\begin{aligned} f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l) \\ = f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l) \end{aligned}$$

E.g.,  $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Exact inference by enumeration

39

## Variable elimination algorithm

```
function ELIMINATION-ASK( $X, e, bn$ ) returns a distribution over  $X$ 
inputs:  $X$ , the query variable
        $e$ , evidence specified as an event
        $bn$ , a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$ 

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

Exact inference by enumeration

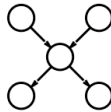
40

## Irrelevant variables

Consider the query  $P(\text{JohnCalls} | \text{Burglary} = \text{true})$

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

Sum over  $m$  is identically 1;  $M$  is **irrelevant** to the query



Thm 1:  $Y$  is irrelevant unless  $Y \in \text{Ancestors}(\{X\} \cup \mathbf{E})$

Here,  $X = \text{JohnCalls}$ ,  $\mathbf{E} = \{\text{Burglary}\}$ , and  $\text{Ancestors}(\{X\} \cup \mathbf{E}) = \{\text{Alarm}, \text{Earthquake}\}$  so  $\text{MaryCalls}$  is irrelevant

(Compare this to backward chaining from the query in Horn clause KBs)

Exact inference by enumeration

41

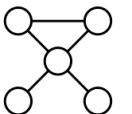
## Irrelevant variables contd.

Defn: **moral graph** of Bayes net: marry all parents and drop arrows

Defn:  $\mathbf{A}$  is **m-separated** from  $\mathbf{B}$  by  $\mathbf{C}$  iff separated by  $\mathbf{C}$  in the moral graph

Thm 2:  $Y$  is irrelevant if m-separated from  $X$  by  $\mathbf{E}$

For  $P(\text{JohnCalls} | \text{Alarm} = \text{true})$ , both  $\text{Burglary}$  and  $\text{Earthquake}$  are irrelevant



Exact inference by enumeration

42

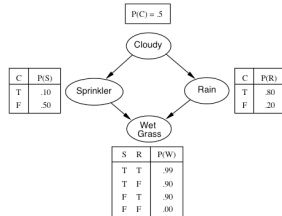
## Complexity of exact inference

**Singly connected** networks (or **polytrees**):

- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are  $O(d^k n)$

**Multiply connected** networks:

- can reduce 3SAT to exact inference  $\Rightarrow$  NP-hard
- equivalent to **counting** 3SAT models  $\Rightarrow$  #P-complete



Exact inference by enumeration

43

## Clustering algorithms

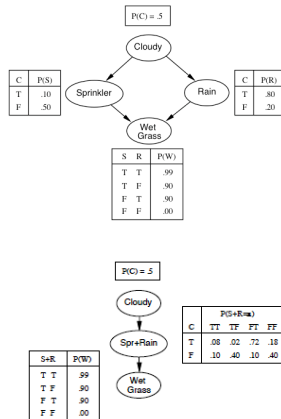
- Variable elimination algorithm is simple and efficient for answering individual queries
- For computation of posterior probabilities for all the variables in a network it can be less efficient:  $O(n^2)$
- Using clustering algorithms (also known as join tree algorithms), this can be reduced to  $O(n)$ .
- The 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

Exact inference by enumeration

44

## Clustering algorithms

- Multiply connected network can be converted into a polytree by combining *Sprinkler* and *Rain* node into cluster node called *Sprinkler + Rain*.
- Two Boolean nodes replaced by a meganode that takes on four possible values: *TT, TF, FT, FF*. The meganode has only one parent, the Boolean variable *Cloudy*, so there are two conditioning cases.



Exact inference by enumeration

45

## Summary

This chapter has described **Bayesian networks**, a well-developed representation for uncertain knowledge. Bayesian networks play a role roughly analogous to that of propositional logic for definite knowledge.

- 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.

46

## Summary (2)

- 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 polytrees (singly connected networks), exact inference takes time linear in the size of the network. In the general case, the problem is intractable.

47