LU33-Exact Inference in Bayesian-Networks

LU Objectives	
To infer by enumeration using Bayesian network	
LU Outcomes	CO : 5
Able to infer from Bayesian networks	

Resource Acknowledgment:
Alex Lascarides - alex@inf.ed.ac.uk

Inference in BNs

- Basic task: compute posterior distribution for set of **query variables** given some observed **event** (i.e. assignment of values to **evidence variables**)
- Formally: determine $\mathbf{P}(X|\mathbf{e})$ given query variables $\mathbf{X}$, evidence variables $\mathbf{E}$ (and non-evidence or **hidden** variables $\mathbf{Y}$)
- Example:
  $\mathbf{P}(Burglary|JohnCalls = true, MaryCalls = true) = \langle 0.284, 0.716 \rangle$
- First we will discuss exact algorithms for computing posterior probabilities then approximate methods later

Inference by enumeration

- We have seen that any conditional probability can be computed from a full JPD by summing terms
- $\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$
- Since BN gives complete representation of full JPD, we must be able to answer a query by computing sums of products of conditional probabilities from the BN
- Consider query $\mathbf{P}(Burglary|JohnCalls = true, MaryCalls = true) = \mathbf{P}(B|j, m)$
- $\mathbf{P}(B|j, m) = \alpha\mathbf{P}(B, j, m) = \alpha \sum_{e} \sum_{a} \mathbf{P}(B, e, a, j, m)$

Enumeration algorithm for answering queries on Bayesian networks

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**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(bn.VARS, $\mathbf{e}_{x_i}$)
      where $\mathbf{e}_{x_i}$ is **e** extended with $X = x_i$
  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 $Y = y$

Inference by enumeration

- Recall $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$
- We can use CPTs to simplify this exploiting BN structure
- For $Burglary = true$:

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

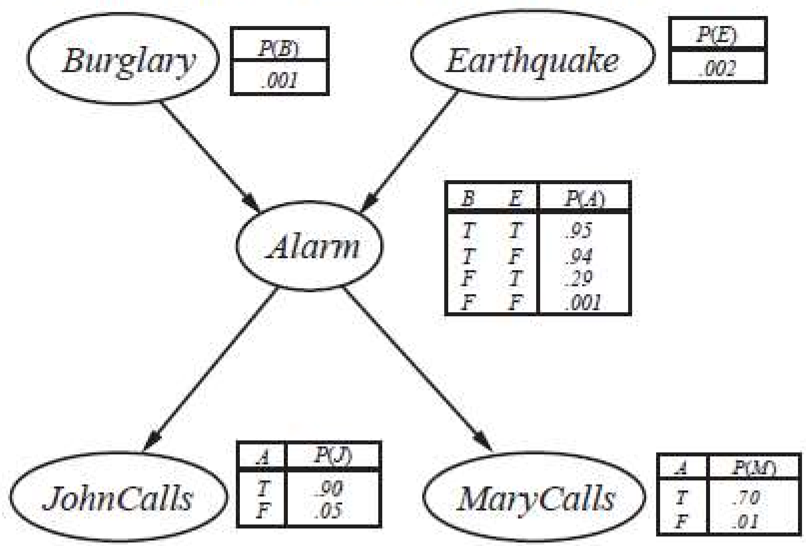
- But we can improve efficiency of this by moving terms outside that don't depend on sums

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

- To compute this, we need to loop through variables in order and multiply CPT entries; for each summation we need to loop over variable's possible values

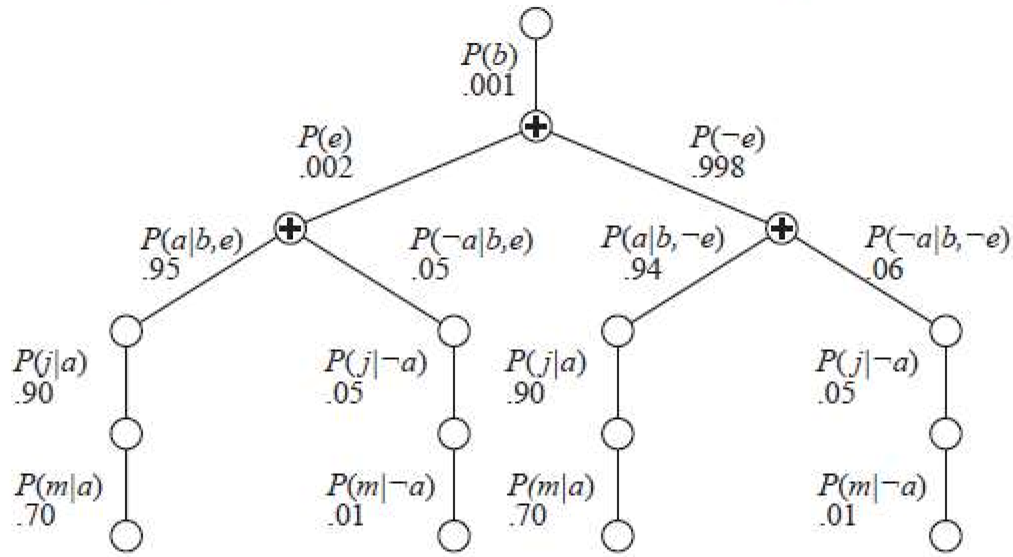
Example

- New burglar alarm has been fitted, fairly reliable but sometimes reacts to earthquakes
- Neighbours John and Mary promise to call when they hear alarm
- John sometimes mistakes phone for alarm, and Mary listens to loud music and sometimes doesn't hear alarm



The variable elimination algorithm

- Enumeration method is computationally quite hard.
- You often compute the same thing several times; e.g. $P(j|a)P(m|a)$ and $P(j|\neg a)P(m|\neg a)$ for each value of e
- Evaluation of expression shown in the following tree:



Variable elimination algorithm for inference in Bayesian networks

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

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

The variable elimination algorithm

- Idea of **variable elimination**: avoid repeated calculations
- Basic idea: store results after doing calculation once
- Works bottom-up by evaluating subexpressions
- Assume we want to evaluate

$$\mathbf{P}(B|j,m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathsf{f}_1(B)} \sum_e \underbrace{\mathbf{P}(e)}_{\mathsf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a|B,e)}_{\mathsf{f}_3(A,B,E)} \underbrace{P(j|a)}_{\mathsf{f}_4(A)} \underbrace{P(m|a)}_{\mathsf{f}_5(A)}$$

- We've annotated each part with a **factor**.
- A factor is a **matrix**, indexed with its argument variables. E.g:
  - Factor $\mathbf{f}_5(A)$ corresponds to $P(m|a)$ and depends just on A because m is fixed (it's a 2×1 matrix).

$$\mathbf{f}_5(A) = \langle P(m|a), P(m|\neg a) \quad \mathbf{f}_5(A) = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.70 \\ 0.01 \end{pmatrix}$$

  - $\mathbf{f}_3(A,B,E)$ is a $2 \times 2 \times 2$ matrix for $\mathbf{P}(a|B,e)$

The variable elimination algorithm

$\mathbf{P}(B|j,m) = \alpha \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \sum_a \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$

- Summing out A produces a 2×2 matrix (via **pointwise product**):
$$\begin{aligned} \mathbf{f}_6(B,E) = & \sum_a \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A) \\ = & (\mathbf{f}_3(a,B,E) \times \mathbf{f}_4(a) \times \mathbf{f}_5(a)) + \\ & (\mathbf{f}_3(\neg a,B,E) \times \mathbf{f}_4(\neg a) \times \mathbf{f}_5(\neg a)) \end{aligned}$$
- So now we have
$\mathbf{P}(B|j,m) = \alpha \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B,E)$
- Sum out E in the same way:
$\mathbf{f}_7(B) = (\mathbf{f}_2(e) \times \mathbf{f}_6(B,e)) + (\mathbf{f}_2(\neg e) \times \mathbf{f}_6(B,\neg e))$
- Using $\mathbf{f}_1(B) = \mathbf{P}(B)$, we can finally compute
$$\mathbf{P}(B|j,m) = \alpha \mathbf{f}_1(B) \times \mathbf{f}_7(B)$$
- Remains to define pointwise product and summing out

An example

- Pointwise product yields product for union of variables in its arguments:

$$\mathbf{f}(X_1 \ldots X_i, Y_1 \ldots Y_j, Z_1 \ldots Z_k) = \mathbf{f}_1(X_1 \ldots X_i, Y_1 \ldots Y_j)\mathbf{f}_2(Y_1 \ldots Y_j, Z_1 \ldots Z_k)$$

A	B	$\mathbf{f}_1(A, B)$	B	C	$\mathbf{f}_2(B, C)$	A	B	C	$\mathbf{f}(A, B, C)$
T	T	0.3	T	T	0.2	T	T	T	0.3×0.2
T	F	0.7	T	F	0.8	T	T	F	0.3×0.8
F	T	0.9	F	T	0.6	T	F	T	0.7×0.6
F	F	0.1	F	F	0.4	T	F	F	0.7×0.4
						F	T	T	0.9×0.2
						F	T	F	0.9×0.8
						F	F	T	0.1×0.6
						F	F	F	0.1×0.4

- For example $\mathbf{f}(T, T, F) = \mathbf{f}_1(T, T) \times \mathbf{f}_2(T, F)$

$$\begin{aligned}\mathbf{f}(B,C) &= \sum_a \mathbf{f}_3(A,B,C) = \mathbf{f}_3(a,B,C) + \mathbf{f}_3(\neg a,B,C) \\ &= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix} .\end{aligned}$$

An example

- Summing out is similarly straightforward
- Trick: any factor that does not depend on the variable to be summed out can be moved outside the summation process
- For example

$$
\begin{aligned}
&\sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A) \\
&\quad = \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A, B, E)
\end{aligned}
$$

- Matrices are only multiplied when we need to sum out a variable from the accumulated product

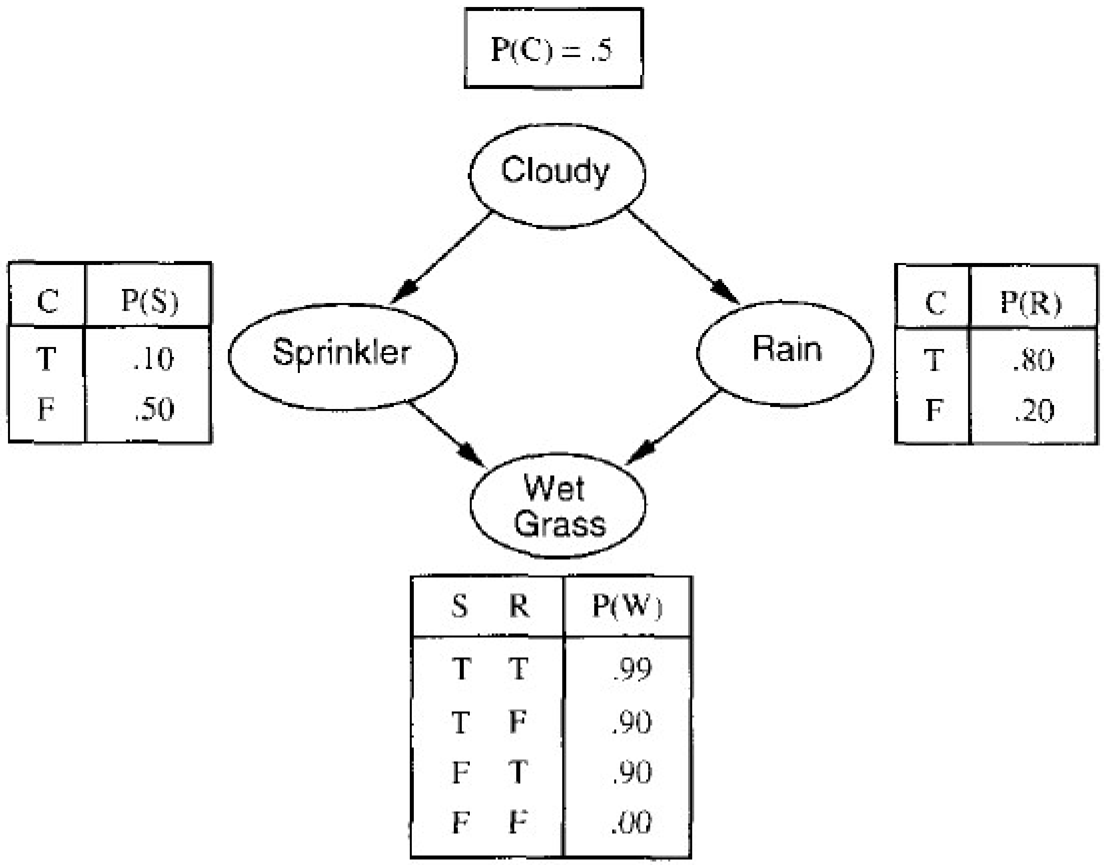
- Let us consider one more query: P(JohnCalls | Burglary =true).
- The first step is to write out the nested summation:

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

- Evaluating this expression from right to left, wkt P(m| a) is equal to 1 by definition.
- That is the variable M is irrelevant to this query.
- In general, we can remove any leaf node that is not a query variable or an evidence variable.
- After its removal, there may be some more leaf nodes, and these too may be irrelevant.
- Continuing this process, we eventually find that every variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query.
- A variable elimination algorithm can therefore remove all these variables before evaluating the query.

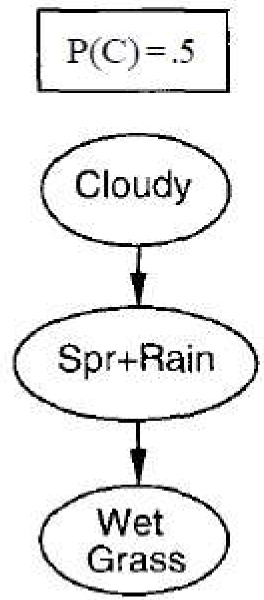
- A network in which there is at most one undirected path between any two nodes in the network is called singly connected networks or polytrees.
- The time and space complexity of exact inference in polytrees is linear in the size of the network. (size =the number of CPT entries)
- If the number of parents of each node is bounded by a constant, then the complexity will also be linear in the number of nodes.
- For multiply connected networks, variable elimination can have exponential time and space complexity in the worst case, even when the number of parents per node is bounded.
- That is, inference in Bayesian networks is NP-hard.

Clustering Algorithm (Join Tree)



A multiply connected network with conditional probability tables

A clustered equivalent of the multiply connected network



	P(S+R=x)			
C	TT	TF	FT	FF
T	.08	.02	.72	.18
F	.40	.10	.40	.10

S+R	P(W)
T T	.99
T F	.90
F T	.90
F F	.00