8 Probabilistic Reasoning

Syllabus

Representing Knowledge in an Uncertain Domain, The Semantics of Bayesian Networks, Efficient Representation of Conditional Distribution, Exact Inference in Bayesian Networks, Approximate Inference in Bayesian Networks.

Contents

8.1 Probabilistic Reasoning

GTU : Summer-18,19, Winter-19

Representing knowledge in uncertain domain :

Independence and conditional independence relationships among variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution. There is technique called as Bayesian Network which can be used to represent the dependencies among variables and to give a concise specification of any full joint probability distribution.

8.1.1 Representing Knowledge in an Uncertain Domain and Bayesian Network

8.1.1.1 Bayesian Networks Definition

It is a data structure which is a graph, in which each node is annotated with quantitative probability information.

The nodes and edges in the graph are specified as follows :-

1) A set of random variables makes up the nodes of the network. Variables may be discrete or continuous.
2) A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, then X is said to be a parent of Y.
3) Each node X_i, has a conditional probability distribution $P(X_i \mid Parents(X_i))$ that quantifies the effect of the parents on the node.
4) The graph has no directed cycles (and hence is a directed, acyclic graph, or DAG).

The set of nodes and links is called as **topology of the network**. The topology of the network specifies the conditional independence relationships that hold in the domain.

The intutive meaning of an arrow between two nodes X and Y, in a properly constructed network, is that "X has direct influence on Y". The domain expert is a person who can identify such influence associations.

Once Bayesian network topology is specified then conditional probability distribution for each variable is specified. For specifying continuous probability distribution for a variable, its parent information is required.

The combination of topology and the conditional distributions is sufficient to specify the full joint distribution for all the variables.

Consider our example of simple world consisting of variables toothache, cavity, catch and weather. Weather is surly independent of other variables. Toothache and Catch are conditionally independent if given the information of cavity.

This relationships are represented by Bayesian network as shown below :

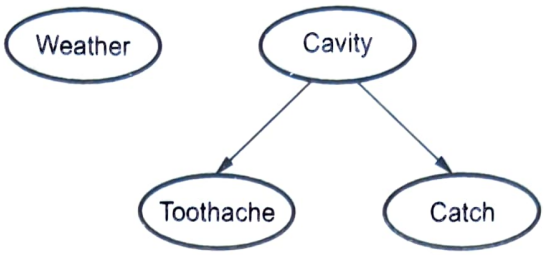


Fig. 8.1.1 A simple Bayesian network in which Weather is independent of other three variables and Toothache and Catch are conditionally independent, given the cavity

Note

1) The conditional independence of toothache and catch given the cavity is indicated by the absence of a link between toothache and catch.
2) The network represents the fact that cavity is a direct cause of toothache and catch.
3) No direct causal relationship exists between toothache and catch.

Consider another example of burglar alarm installed at A's home. This alarm notifies in case of burglary and can occasionally respond to minor earthquakes. There are two neighbours to A, M and J. M and J promised to call A in office when they hear alarm. J always calls A when he hears alarm, but many times he is confused between telephone ring and burglar alarm. M many a times misses alarm because he keeps on listening to loud music. Given the evidence of who has or has not called, we are to estimate the probability of burglary.

The Bayesian network for above problem along with the associated probabilities is shown below :

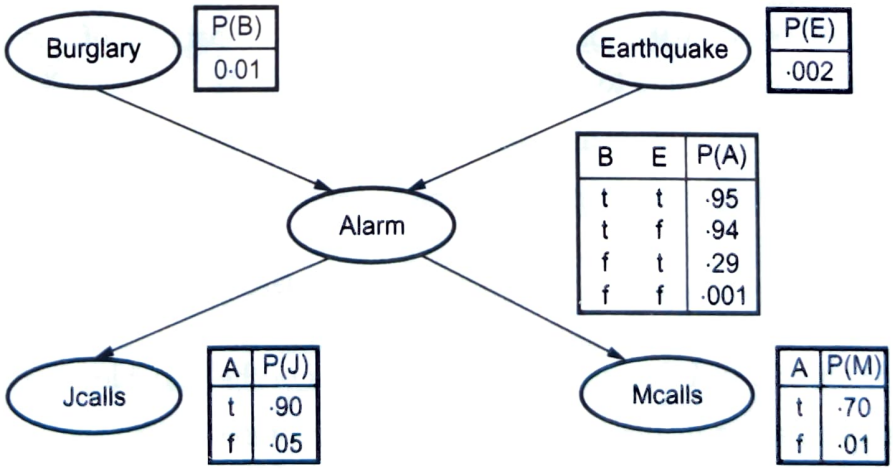


Fig. 8.1.2 A Typical Bayesian network, showing both the topology and the conditional probability tables (CPTs).

Note *In the figure, in the CPTs, the letters B, E, A, J and M stand for Burglary, Earthquake, Alarm, Johncalls, Marycalls respectively.*

1) In the figure, each distribution is shown as conditional probability table. Each row in the table contains the conditional probability of each node value for a conditioning case. A conditioning case is just a possible combination of values for the parent nodes. Each row must sum upto 1 because the entries represent an exhaustive set of cases for the variable.

2) For boolean variable if it is true and its probability is know to be 'P' then when it is false its probability is 1–P and it is ommited from the table (because it can be deduced from 'P').

3) A table for a boolean variable with k boolean parents contains 2^k independently specifiable probabilities. A node with no parents has only one row, representing the prior probabilities of each possible value of the variable.

8.1.1.2 The Semantics of Bayesian Networks

There are two ways through which semantic (meaning) of Bayesian network can be understood.

One way is to view network as a representation of the joint probability distribution. This view helps to constructs networks. Second way is to view a network as an encoding of a collection of conditional independence statements. This view helps in designing inference procedure semantically both views are equivalent.

Understanding semantic of Bayesian network method 1

- Representing the full joint distribution :
  1) Every entry in the full joint probability distribution (hereafter abbreviated as "joint") can be calculated from the information in the network.
  2) A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, such as $P(X_1 = x_1 \wedge ..... \wedge X_n = x_n)$
  3) The notation $P(x_1, ..., x_n)$ is used as an abbrevation for this.
  4) The value of this entry is given by the formula.

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i \mid \text{parents } (X_i)) \quad \ldots (8.1.1)$$

  Where parents (x_i) denotes the specific values of the variables in parents (x_i).

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

We can calculate the probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred and both J and M call. We use single letter names for the variables :

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

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$

6) Remember that the full joint distribution can be used to answer any query about the domain.

If a Bayesian network is a representation of the joint distribution, then it too can be used to answer any query, by summing all the relevant joint entries.

- A method for constructing Bayesian network :

1) We rewrite the joint distribution in terms of a conditional probability, using the product rule.

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

2) Then the process is represented, reducing each conjunctive probability to a conditional probability and a smaller conjunction. We end up with one big product.

$$P(x_1, \ldots, x_n) = P(x_n | x_{n-1}, \ldots, x_1) P(x_{n-1} | x_{n-2}, \ldots, x_1) \ldots P(x_2 | x_1)\, P(x_1)$$

$$= \prod_{i=1}^{n} P(x_i | x_{i-1}, \ldots x_1) \qquad \ldots (8.1.2)$$

3) Above identity holds true for any set of random variables and is called the **Chain rule.** Comparing it with equation (8.1.1). We see that the specification of the joint distribution is equivalent to the general assertion that, for every variable X_i in the network,

$$P(X_i | X_{i-1}, \ldots X_1) = P(X_i | \text{Parents}(X_i)),$$

provided that parents $(X_i) \subseteq \{X_{i-1}, \ldots X_1\}$. This last condition is satisfied by labeling the nodes in any order that is consistent with the partial order implicit in the graph structure.

4) Bayesian network is correct representation of the domain only if each node is conditionally independent of its predecessors in the node ordering, given its parents.

5) In order to construct a Bayesian network with the correct structure for the domain, we need to choose parents for each node such that this property holds. Intuitively, the parents of node X_i should contain all those nodes in $X_i, \ldots X_{i-1}$ that directly influence X_i.

For example : Suppose we have completed the network in Fig. 8.1.2 except for the choice of parents for M calls, M calls is certainly influenced by

whether there is a burglary or an earthquake, but not directly influenced. Intitutively, our knowledge of the domain tells us that these events influence M's calling behaviour only through their effect on the alarm. Also given the state of the alarm, whether J calls has no influence on M's calling. Formally speaking, we believe that the following conditional independence statement holds :

P(M Calls | J Calls, Alarm, Earthquake, Burglary) = **P**(M calls | Alarm).

- **Compactness and node ordering :**

  Bayesian network are compact and they possess a property of being locally structured (also called as **sparse systems**). In a locally structured system, each sub component interacts directly with only a bounded number of other components, regardless of the total number of components.

⇒ Local structure is usually associated with linear rather than exponential growth in complexity.

⇒ We assume 'n' boolean variables for simplicity, then the amount of information needed to specify each conditional probability table will be at most 2^k numbers. Where each random variable is influenced by 'k' other variables.

⇒ The complete network can be specified by $n2^k$ numbers.

⇒ With some values of 'n' the joint distribution contains 2^n numbers.

⇒ To make this concrete, suppose we have n = 30 nodes, each with five parents (k=5). Then the Bayesian network requires 960 numbers, but the full joint distribution requires over a billion.

Ordering of nodes in Bayesian network :

1) The correct order in which to add nodes is to add the "root causes" first, then the variables they influence, and so on, until we reach the "leaves", which have no direct causal influence on the other variables.
2) If wrong order is choosen we get more complicated network.

   For example : Consider network shown in following diagram.

The network construction process goes as follows :

i) Adding M calls : No parent.

ii) Adding J calls : If M calls, that probably means the alarm has gone off, which ofcourse would make it more likely that J calls. Therefore J calls needs M calls as a parent.

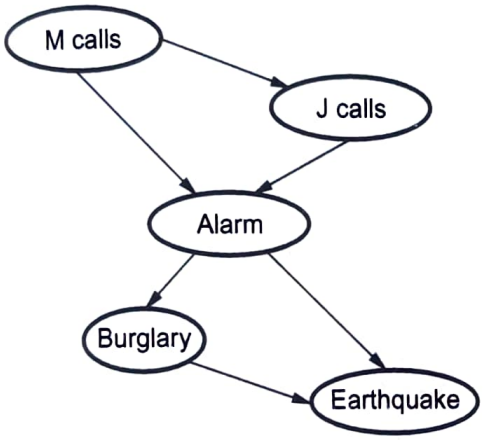
iii) Adding alarm : Clearly, if both call, it is more likely that the alarm has gone off than if just one or neither call, so we need both M calls and J calls as parent.

iv) Adding burglary : If we know the alarm state, then the call from J or M might give us information about phone ringing or M's music, but not about burglary : P(Burglary | Alarm, J calls, M calls) = P(Burglary | Alarm).

Hence we need just alarm as parent.

v) Adding earthquake : If the alarm is on, it is more likely that there has been an earthquake. (The alarm is an earthquake detector of sorts). But if we know that there has been a burglary, then that explains the alarm, and the probability of an earthquake would be only slightly above normal. Hence, we need both Alarm and Burglary as parents. The network is shown below.

- Consider example of a very bad node ordering as shown in the diagram 8.1.4 :



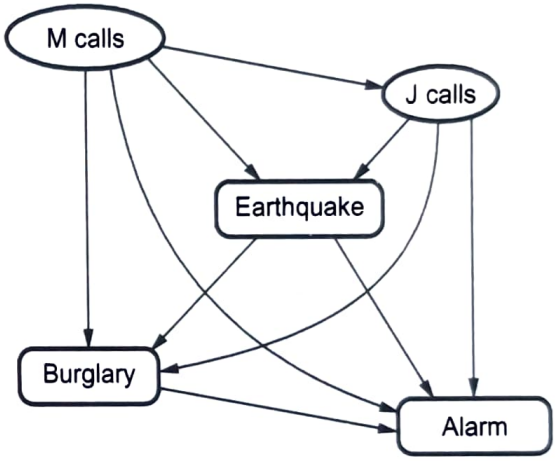


Fig. 8.1.3 and 8.1.4 Network structure depends on order of introduction. In each network nodes are added from top to bottom

M calls, J calls, Earthquake, Burglary, Alarm are nodes. This network requires 31 distinct probabilities to be specified exactly the same as the full joint distribution. It is important to realize, however, that any of the three networks can represent exactly the same joint distribution. The last two versions simply fail to represent all conditional independence relationships and hence end up specifying a lot of unnecessary numbers instead.

8.1.1.3 Efficient Representation of Conditional Distribution

One can start from a "topological" semantics that specifies the conditional independence relationships encoded by the graph structure, and from these we can derive the "numerical" semantics. The topological semantics is given by either of the following specifications, which are equivalent.

1) A node is conditionally independent of its **non-descendants**, given its parents. For example, in Fig. 8.1.2 J calls is independent of Burglary and Earthquake, given the value of Alarm.

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

For example : Burglary is independent of J calls and M calls given Alarm and Earthquake.

This specifications are illustrated in following figures. From these conditional independence assertions and the CPTs, the full joint distribution can be reconstructed; thus the "numerical" semantics and the "topological" semantics are equivalent.

A node X is conditionally independent of its non-descendants (example - The Z_{1j}s) given its parents (the U_i is shown in the gray area).

A node X is conditionally independent of all other nodes in the network given its Markove blanket (the gray area).

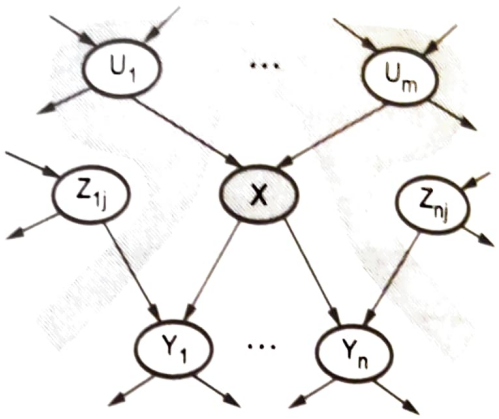


Fig. 8.1.5 (a) A node x is conditionally independent of its non-descendants. (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area).

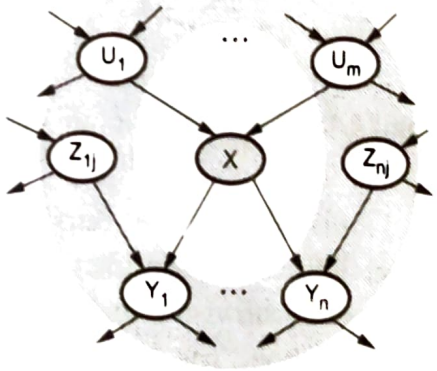


Fig. 8.1.5 (b) A node x is conditionally independent of all other nodes in the network given its markov blanket (the gray area)

Efficient representation of conditional representation :

If the maximum number of parents k is smallish, filling in the CPT for a node, would require up to $O(2^k)$ numbers.

To avoid this big relationship number canconical distribution is used. In this a complete table is specified by naming the patterns supplied with parameters, which can describe relationship which are necessary. A deterministic node represents canonical distribution. A deterministic node is a node whose value is exactly specified by the value of its parents with no uncertainity.

The uncertain relationships are often represented by noisy-OR relationships. [Noisy-OR is generalization of logical OR]. The noisy-OR model allows uncertainity about the parent to cause the child to be true i.e. the causal relationship between parent and child would be inhibited.

For example : A patient could have cold (parent) but not exhibit a fever (child).

For noisy-OR model it is required that all the possible effects of parent must be listed. It should be clear that inhibition of one parent is independent of other parent.

8.1.2 The Hybrid Bayesian Network

A network with both discrete and continuous variables is called as hybrid Bayesian network. For representing continuous variable its discretization is done in terms of intervals (because it can have infinite values).

To specify a hybrid network, we have to specify two new kinds of distributions. The conditional distribution for a continuous variable given discrete or continuous parents; and the conditional distribution for a discrete variable given continuous parents.

Consider the simple example in following diagram, in which 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. The variable Cost is continuous and has continuous and discrete parents; the variable Buys is discrete and has a continuous parent.

For the cost variable, we need to specify P(Cost | Harvest, Subsidy). The discrete parent is handled by explicit enumeration that is, specifying both P(Cost | Harvest, Subsidy) and P(Cost | Harvest, ¬ Subsidy). To handle Harvest, we specify how the distribution over the cost 'c' depends on the continuous value 'h', of Harvest. In other words, we specify the parameter of the cost distribution as a function of 'h'.

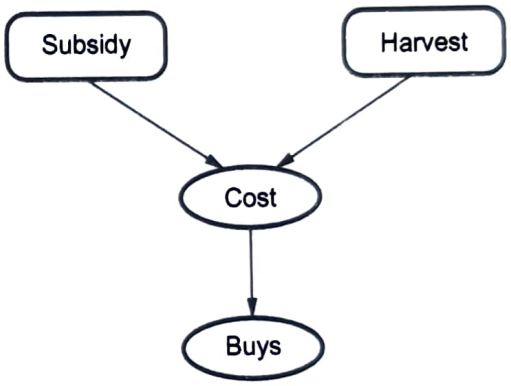


Fig. 8.1.6 A simple network with discrete variables (Subsidy and Buys) and continuous variables (Harvest and Cost).

8.1.3 Inferencing in Bayesian Networks

8.1.3.1 Exact Inference in Bayesian Networks

For inferencing in probabilistic system, it is required to calculate posterior probability distribution for a set of query variables, where some observed events are given. [That is we have some values attached to evidence variables].

- Notation Revisited :

The notation used in inferencing is same as the one used in probability theory.

X : Query variable.

E : The set of evidence variables E_1,....., E_m and 'e' is the perticular observed event.

Y : The set of non-evidence variables $Y_1, Y_2, \ldots Y_k$ [Non-evidence variables are also called as hidden variables].

X : It the complete set of all the types of variables, where $\mathbf{X} = \{X\} \cup E \cup Y$.

- Generally the query requires the posterior probability distribution $P(X|e)$ [assuming that query variable is not among the evidence variables, if it is, then posterior distribution for X simply gives probability 1 to the observed value].
  [Note that query can contain more than one variable. For study purpose we are assuming single variable].
- Example : In the burglary case, if the observed event is Jcalls = true and Mcalls = true.
  The query is 'Has burglary occured ?'
  The probability distribution for this situation would be,
  P(Burglary | J calls = true, M calls = true) = < 0.284, 0.716 >

8.1.3.2 Inference by Enumeration

A Bayesian network gives a complete representation of the full joint distribution. These full joint distributions can be written as product of conditional probabilities from the Bayesian network.

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

- The algorithm
  The algorithm ENUMERATE-JOINT-ASK gives inference by enumerating on full joint distribution.

Characteristics of algorithm :

1) It takes input a full joint distribution **P** and looks up values in it. [The same algorithm can be modified to take input as Bayesian network and looking up in joint entries by multiplying the corresponding conditional probability table entries from Bayesian network.
2) The ENUMERATION-JOINT-ASK uses ENUMERATION-ASK (EA) algorithm which evaluate expression using depth-first recursions. Therefore, the **space complexity** of EA is only linear in the number of variables. The algorithm sums over the full joint distribution without ever constructing it explicitely. The time complexity for network with 'n' boolean variables is always $O(2^n)$ which is better than the $O(n\,2^n)$ required in simple inferencing approach (using posterior probability).
3) The drawback of the algorithm is, it keeps on evaluating repeated subexpression which results in wastage of computation time.

The enumeration algorithm for answering queries on Bayesian network.

- The 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 Bayes net with variables {X} $\cup$ E $\cup$ Y/* y = Hidden variable */

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.

$Q(x_i) \leftarrow$ ENUMERATE-ALL (VARS[bn] e) return NORMALIZE (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 | parents(Y)) X ENUMERATE-ALL (REST(vars), e)

else return $\sum_y$ P(y | parents(Y)) X ENUMERATE-ALL (REST(vars), e_y)

where e_y is e extended with Y = y.

Example :

Consider query,

P(Burglary | J calls = true, M calls = true)

Hidden variables in the queries are $\rightarrow$ Earthquake and Alarm.

using the query equation.

$$\mathbf{P}(\text{Burglary} \mid j, m) = \alpha\ \mathbf{P}(\text{Burglary}, J, M) = \alpha \sum_e \sum_a \mathbf{P}(\text{Burglary}, e, a, j, m)$$

The semantics of Bayesian networks (equation 8.1.1) then gives us an expression in terms of CPT entries. For simplicity, we will do this just for Burglary = true.

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

- To compute this expression, we have to add four terms, each computed by multiplying five numbers.
- **Worst case**, where we have to sum out almost all the variables, the complexity of the algorithm for a network with n boolean variables is $O(n2^n)$.

- An **improvement** can be obtained from the following simple observations. The P(b) term is a constant and can be moved outside the summations over a and e, and the P(e) term can be moved outside the summation over a. Hence, we have

$$P(b \mid j, m) = \alpha P(b) \sum_{e} P(e) \sum_{a} P(a \mid b, e) P(j \mid a) P(m, a) \quad \ldots (8.1.3)$$

This expression can be evaluated by looping through the variables in order, multiplying CPT entries as we go. For each summation, we also need to loop over the variable's possible values. The structure of this computation is shown in following diagram. Using the numbers from Fig. 8.1.2, we obtain $P(b \mid j, m) = \alpha \times 0.00059224$. The corresponding computation for $\neg b$ yields $\alpha \times 0.0014919$;

Hence,

$$P(B \mid j, m) = \alpha < 0.00059224, 0.0014919 >$$

$$\approx < 0.284, 0.716 >$$

That is, the chance of burglary, given calls from both neighbours is about 28 %.

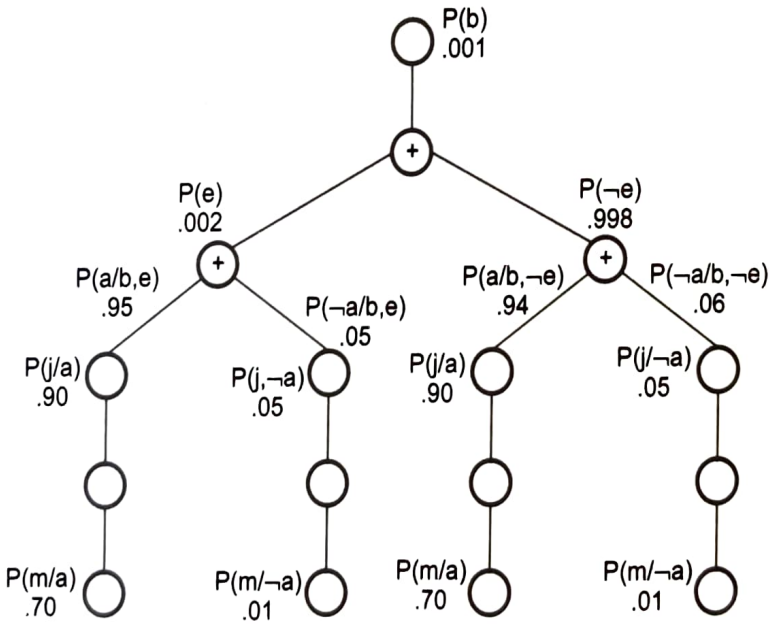


Fig. 8.1.7 The structure of the expression shown in equation 8.1.3

Note *In the Fig. 8.1.7, the evalution proceeds top to down, multiplying values along each path and summing at the "t" nodes. Observe that there is repetition of paths for j and m.*

8.1.3.3 The Variable Elimination Algorithm

The enumeration algorithm can be improved substantially by eliminating calculations of repeated sub expression in tree. Calculation can be done once and save the results for later use. This is a form of dynamic programming.

Working of variable elimination algorithm

1) It works by evaluating expressions such as $[P(b|j, m) = \alpha\ P(b) \sum_e P(e) \sum_a P(a|b, e)\ P(j|a)\ P(m|a)]$ in right-to-left order.

2) Intermediate results are stored and summations over each variable are done only for those portions of the expression that depends on the variable.

For example : Consider the Burglary network.

We evaluate the expression :

$$P(B|j, m) = \underbrace{\alpha\ P(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum \underbrace{P(a|B, e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}$$

3) Factors : Each part of the expression is annotated with the name of the associated variable, these parts are called factors.

Steps in algorithm :

i) The factor for M, $P(m|a)$, does not require summing over M. Probability is stored, given each value of a, in a two-element factor,

$$f_M(A) = \begin{bmatrix} P(m|a) \\ P(m|\neg a) \end{bmatrix}$$

Note : f_M means that M was used to produce f.

ii) Store the factor for J as the two-element vector $f_J(A)$.

iii) The factor for A is $P(a|B, e)$ which will be a $2\times2\times2$ matrix $f_A(A, B, E)$.

iv) Summing out A from the product of these tree factors. This will give 2×2 matrix whose indices range over just B and E. We put bar over A in the name of the matrix to indicate that A has been summed out.

$$f_{\bar{A}JM}(B, E) = \sum_a f_A(a, B, E) \times f_J(a) \times f_M(a)$$

$$= f_A(a, B, E) \times f_J(a) \times f_M(a) + f_A(\neg a, B, E) \times f_J(\neg a) \times f_M(\neg a)$$

The multiplication process used is called a printwise product.

v) Processing E in the same way (i.e.) sum out E from the product of $f_E(E)$ and $f_{\bar{A}JM}(B, E)$:

$$f_{\bar{A}JM}(B) = f_E(e) \times f_{\bar{A}JM}(B, e) + f_E(\neg e) \times f_{\bar{A}JM}(B, \neg e).$$

vi) Compute the answer simply by multiplying the factor for B. (i.e.) $(f_B|B) = P(B)$, by the accumulated matrix (B) :

$$P(B|j, m) = \alpha\ f_B(B) \times f_{\bar{E}\bar{A}JM}(B).$$

From the above sequence of steps it can be noticed that two computational operations are required.

a) Pointwise product of a pair of factors.

b) Summing out a variable from a product of factors.

a) **Pointwise product of a pair of factors** : The pointwise product of two factors f_1 and f_2 yields a new factor f, those variables are the union of the variables in f_1 and f_2. Suppose the two factors have variables $Y_1, \ldots, Y_k$. Then we have

$$f(X_1, \ldots, X_j, Y_1, \ldots Y_k, Z_1, \ldots Z_l) = f_1(X_1, \ldots X_j, Y_1 \ldots Y_k)\, f_2(Y_1, \ldots Y_k, Z_1, \ldots Z_l).$$

If all the variables are binary, then f_1 and f_2 have 2^{j+k} and 2^{k+l} entries and the pointwise product has 2^{j+k+l} entries.

For example : Given two factors $f_1(A, B)$ and $f_2(B, C)$ with probability distributions shown below, the pointwise product $f_1 \times f_2$ is given as $f_1(A, B, C)$.

A	B	$f_1(A, B)$	B	C	$f_2(B, C)$	A	B	C	$f_3(A, B, C)$
T	T	.3	T	T	.2	T	T	T	.3 × .2
T	F	.7	T	F	.8	T	T	F	.3 × .8
F	T	.9	F	T	.6	T	F	T	.7 × .6
F	F	.1	F	F	.4	T	F	F	.7 × .4
						F	T	T	.9 × .2
						F	T	F	.9 × .8
						F	F	T	.1 × .6
						F	F	F	.1 × .4

b) **Summing out a variable from a product of factors** : It is a straight forward computation. Any factor that does not depend on the variable to be summed out can be moved outside the summation process.

For example :

$$\sum_e f_E(e) \times f_A(A, B, e) \times f_J(A) \times f_M(A) = f_J(A) \times f_M(A) \times \sum_e f_E(e) \times f_A(A, B, e).$$

Now, the pointwise product inside the summation is computed and the variable is summed out of the resulting matrix.

$$f_J(A) \times f_M(A) \times \sum_e f_E(e) \times f_A(A, B, e) = f_J(A) \times f_M(A) \times f_{\bar{E}A}(A, B).$$

Matrices are not multiplied until we need to sum but a variable from the accumulated product. At that point, multiply those matrices that include the variable to be summed out.

The procedure for pointwise product and summing is given below, the variable elimination algorithm is shown below :

```
Function ELIMINATION-ASK (X, e, bn) returns a distribution over X
Inputs : X, the query variable
         e, evidence specified as an event
         bn, a Bayesian network specifying joint distribution P(X1,...., Xn).
Factors ← [ ] : vars ← REVERSE (VARS [bn])
for each var in vars do
Factors ← [MAKE-FACTOR (var, e) factors]
if var is a hidden variable then
Factors ← SUM-OUT (var, factors)
returns NORMALIZE (POINTWISE-PRODUCT (factors)).
```

P(J calls, Burglary = True)

The first step is to write out the nested summation.

$$P(J|b) = \alpha\ P(b) \sum_{e} P(e) \sum_{a} P(a|b,e)\ P(J|a) \sum_{m} P(M|a).$$

Evaluating this expression from right to left,

$\sum_{m} P(M|a)$ is equal to 1.

Note *The variable M is irrelevant to this query. Result of the query P(J calls/Burglary = True) is unchanged if we remove M calls from the network. We can remove any leaf node which is not a query variable or an evidence variable. After its removal, there may be more leaf nodes and they may be irrelevant. Eventually we 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 remove all these variables before evaluating the query.*

8.1.3.4 The Complexity Involved in Exact Inferencing

The variable elimination algorithm is more efficient than enumeration algorithm because it avoids repeated computations as well as drops irrelevant variables.

The variable elimination algorithm constructs the factor, deriving its operation. The space and time complexity of variable elimination is directly dependant on size of the largest factor constructed during the operation. Basically the factor construction is determined by the order of elimination of variables and by the structure of the network; which affects both space and time complexity.

For developing more efficient process we can construct **singly connected networks** which are also called as **polytrees**. In singly connected network, there is at most one undirected path between any two nodes in the networks. The singly connected networks have property that, the time and space complexity of exact inference in polytrees is linear in the size of the network. Here the size is defined as 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 example : The Burglary network shown in the Fig. 8.1.2 is a polytrees.

[Note that every problem may not be represented as polytrees].

In multiply networks [In this, their can be multiple undirected paths between any two nodes and more than one directed path between some pair of nodes], variable elimination takes exponential time and space complexity in the worst case, even when the number of parents per node is bounded. It should be noted that variable elimination includes inference in propositional logic as a special case and **inference in Bayesian network is NP-hard**. In fact it is strictly harder than NP-complete problem.

Clustering algorithm :

1) Clustering algorithm (known as joint tree algorithms) in which inferencing time can be reduced to O(n). In clustering individual nodes of the network are joint to form cluster nodes to such a way that the resulting network is a polytree.

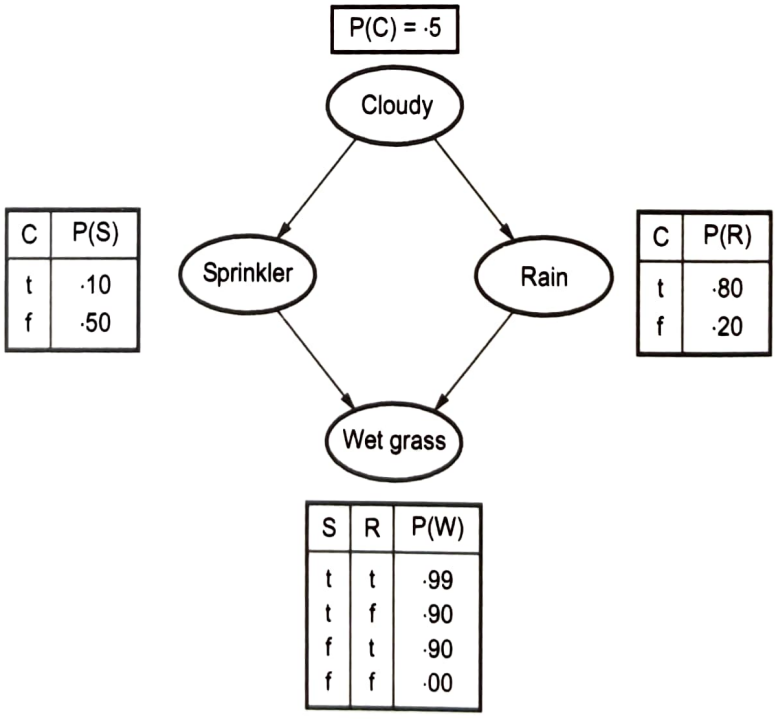


Fig. 8.1.8 (a) A multiply connected network with conditional probability tables

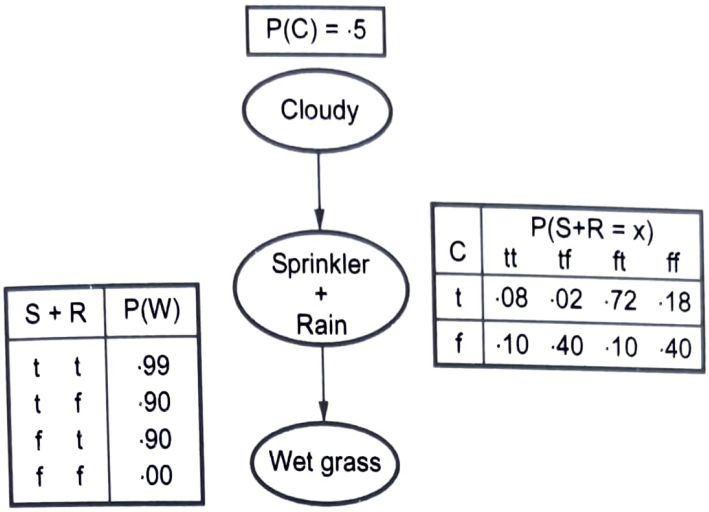


Fig. 8.1.8 (b) A clustered equivalent of the multiply connected network

2) The variable elimination algorithm is efficient algorithm for answering individual queries. Posterior probabilities are computed for all the variables in the network. It can be less efficient, in polytree network because it needs to issue O(n) queries costing O(n) each, for a total of $O(n^2)$ time, clustering algorithm, improves over it.

For example : The multiply connected network shown in following diagram 8.1.8 (a) can be converted into a polytree by combining the Sprinkler and Rain node into a cluster node called Sprinkler + Rain, as shown in diagram 8.1.8 (b). The two Boolean nodes are 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.

Peculiarities of algorithm :

1) Once the network is in polytree form, a special-purpose inference algorithm is applied. Essentially, the algorithm is a form of constraint propagation where the constraints ensure that neighbouring clusters agree on the posterior probability of any variables that they have in common.

2) With careful book keeping, this algorithm is able to compute posterior probabilities for all the non-evidence nodes in the network in time O(n), where n is now the size of the modified network.

3) However, the NP-hardness of the problem continues : If a network requires exponential time and space with variable elimination, then the CPTs in the clustered network will require exponential time and space to construct.

8.1.4 Approximate Inference in Bayesian Network

Because of the fact that exact inference in multiply connected network is intractable problem, we go for approximate inferencing.

For approximate inferencing randomize sampling algorithm (Monte Carlo Algorithm) is used. Its accuracy depends on number of samples generated. Monte Carlo Algorithm is widely used in problem areas where it is difficult to calculate exact quantities.

There are two **families** of Monte Carlo algorithm :

1) Direct sampling algorithm.
2) Markov chain sampling algorithm.

8.1.4.1 Direct Sampling Algorithm

1) The basic element in any sampling algorithm is the generation of samples from a known probability distribution.

   For example : An unbiased coin can be thought of as a random variable coin with values (heads, tails) and a prior distribution P(coin) = <0.5, 0.5>. Sampling from this distribution is exactly like flipping the coin : with probability 0.5 it will return heads, and with probability 0.5 it will return tails. Given a source of random numbers in the range [0, 1], it is a simple matter to sample any distribution on a single variable.

2) The simplest kind of random sampling process for Bayesian networks generates events from a network that has no evidence associated with it. The idea is to sample each variable in turn, in topological order.

3) The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents.

4) The sampling algorithm :

   Function PRIOR-SAMPLE (bn) returns an event sampled from the prior specified by bn.
   inputs : bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
   $X \leftarrow$ an event with n elements.
   for i = 1 to n do
   $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid \text{parent}(X_i))$
   return x.

5) **Applying operations of algorithm on the network in diagram 8.1.8 (a)** assuming an ordering [Cloudy, Sprinkler, Rain, WetGrass] :

   i) Sample from **P**(Cloudy) = <0.5, 0.5>; suppose this returns true.

   ii) Sample from **P**(Sprinkler | Cloudy = true) = <0.1, 0.9>; suppose this returns false.

   iii) Sample from **P**(Rain | Cloudy = true) = <0.8, 0.2>; suppose this returns true.

iv) Sample from P(WetGrass | Sprinkler = false, Rain = true) = <0.9, 0.1>; suppose this returns true.

In this case PRIOR-SAMPLE returns the **event [true, false, true, true].**

6) PRIOR-SAMPLE generates samples from the prior joint distribution specified by the nework. First, let $S_{ps}(x_1,....,x_n)$ be the probability that a specific event is generated by the PRIOR-SAMPLE algorithm. Just looking at the sampling process, we have,

$$S_{ps}(x_1,....,x_n) = \prod_{i=1}^{n} P(x_i | \text{parents}(X_i)).$$

Because each sampling step depends only on the parent values. This expression is also the probability of the event according to the Bayesian net's representation of the joint distribution. We have,

$$S_{ps}(x_1,....,x_n) = P(x_1,....,x_n)$$

7) In any sampling algorithm, the answers are computed by counting the actual samples generated. Suppose there are N total samples, and let $N(x_1,....,x_n)$ be the frequency of the specific event $x_1,....,x_n$. We expect this frequency to converge in the limit, to its expected value according to the sampling probability :-

$$\lim_{N \to \infty} \frac{N_{ps}(x_1,....,x_n)}{N} = S_{ps}(x_1,....,x_n) = P(x_1,....,x_n) \quad \ldots (8.1.4)$$

For example : Consider the event produced earlier : [true, false, true, true]. The sampling probability for this event is,

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

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

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

For example : One can produce a consistent estimate of the probability of any partially specified event $x_1,....,x_n$, where $m \leq n$, as follows : -

$$P(x_1,....,x_m) \approx N_{ps}(x_1,....,x_m) / N \quad \ldots (8.1.5)$$

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

For example : If we generate 1000 samples from the sprinkler network, and 511 of them have, Rain = true, then the estimated probability of rain, written as P(Rain = true), is 0.511.

8.1.4.2 Rejection Sampling in Bayesian Networks

1) **Rejection sampling is a method for producing samples from a hard-to-sample distribution given an easy-to-simple distribution.**

2) It can be used to compute conditional probabilities ; that is, to determine $P(X \mid e)$.

3) The Rejection Sampling algorithm is,

function REJECTION-SAMPLING (x, e, bn, N) returns an estimate of $P(X \mid e)$

inputs : X, the query variable

e, evidence specified as an event.

bn, a Bayesian network.

N, the total number of samples to be generated.

local variables : N, a vector of counts over X, initially zero.

for j = 1 to N do

$X \leftarrow$ PRIOR-SAMPLE (bn)

if X is consistent with e is $N[x] \leftarrow N[x] + 1$ where x is the value of X in x.

return NORMALIZE (N[x]).

The rejection sampling algorithm for answering queries given evidence in a Bayesian network.

- **Working of algorithm :**
  - i) It generates samples from the prior distribution specified by the network.
  - ii) It rejects all those samples that do not match the evidence.
  - iii) Finally, the estimate $P(X = x \mid e)$ is obtained by counting how often $X = x$ occurs in the remaining samples.

4) Let $\mathbf{P}(X \mid e)$ be the estimated distribution that the algorithm returns. From the definition of the algorithm, we have,

$$P(X \mid e) = \alpha N_{ps}(X, e) = \frac{N_{ps}(X, e)}{N_{ps}(e)}$$

from equation (8.1.5) this becomes,

$$P(X \mid e) \approx \frac{P(x, e)}{P(e)} = P(x, e).$$

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

5) Applying operations of algorithm on the network in the diagram 8.1.8(a), let us assume that we wish to estimate **P**(Rain/Sprinkler = true), using 100 samples. Of the 100 that we generate, suppose 8 have Rain = true and 19 have Rain = false.

Hence,

P(Rain | Sprinkler = true) ≈ NORMALIZE

(<8.19>) = <0.296, 0.704 >

The true answer is <0.3, 0.7>

6) As more samples are collected, the estimate will converges to the true answer. The standard deviation of the error in each probability will be proportional to $1/\sqrt{n}$, where 'n' is the number of samples used in the estimate.

7) The biggest problem with rejection sampling is that it rejects so many samples ! The fraction of samples consistent with the evidence 'e' drops exponentially as the number of evidence variables grows, so the procedure is simply unusable for complex problems.

8.1.4.3 Likelihood Weighing in Bayesian Network

1) **Likelihood weighing avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence 'e'.**

2) The Likelihood Weighing algorithm is,

Function LIKELIHOOD-WEIGHING (X, e, bn, N) returns an estimate of P(X | e)

inputs : X, the query variable

e, evidence specified as an event

bn, a Bayesian network.

N, the total number of samples to be generated.

Local variables : W, a vector of weighted counts over X, initially zero.

for j = 1 to N do

X, w ← WEIGHTED-SAMPLE (bn)

W[X] ← W[x] + w where x is the value of X in x.

return NORMALIZE (W[X]).

Function WEIGHTED-SAMPLE (bn, e) returns an event and a weight

x ← an event with n elements; w ← 1.

for i = 1 to n do

if X, has a value x_i in e

then w ← wX $P(X_i = x_i | \text{parents}(X_i))$

else x_i ← a random sample from $P(X_i | \text{parents}(X_i))$

return X, w.

Note on likelihood weighing algorithm -

i) It fixes the values for the evidence variables E and samples only the remaining variables X and Y. This guarantees that each event generated is consistent with the evidence.

ii) Not all events are equal, however, before tallying the counts in the distribution for the query variable, each event is weighted by the likelihood that the event accords to the evidence, as measured by the product of the conditional probabilities for each evidence variable, given its parents.

iii) Intuitively, events in which the actual evidence appears unlikely should be given less weight.

3) Applying operations of algorithm on the network Fig. 8.1.8 (a), with the query **P**(Rain/Sprinkler = true, WetGrass = true). The process goes as follows :

i) The weight w is set to 1.0

ii) Now, an event is generated.

- Sample from **P**(Cloudy) = <0.5, 0.5>; suppose this returns true.
- Sprinkler is an evidence variable with value true. Therefore, we set
  $w \leftarrow w \times \mathbf{P}$ (Sprinkler = true | Cloudy = true) = 0.1
  Sample from **P**(Rain | Cloudy = true) = <0.8, 0.2>; suppose this returns true.
- WetGrass is an evidence variable with value true. Therefore, we set
  $w \leftarrow w \times$ P(Wet Grass = true | Sprinkler = true, Rain = true) = 0.099.

iii) Here WEIGHTED-SAMPLE returns the event [true, true, true, true] with weight 0.099, and this is tallied under Rain = true.

iv) The weight is low because the event describes a cloudy day, which makes the sprinkler unlikely to be on.

4) **Likelihood Weighting working :**

i) Examine the sampling distribution S_{ws} for WEIGHTED-SAMPLE.

ii) The evidence variables E are fixed with values 'e'.

iii) Call the other variables Z, that is, $Z = \{X\} \cup Y$.

iv) The algorithm samples each variable in Z, in given its parent values :

$$S_{ws}(Z, e) = \prod_{i=1}^{l} P(Z_i | \text{parents}(Z_i)) \quad \ldots (8.1.6)$$

Notice that Parents (Z_i) can include both hidden variables and evidence variables. Unlike the prior distribution P(Z), the distribution S_{ws} pays some attention to the evidence the sampled values for each Z_i will be influenced by evidence among Z_i's ancestors.

v) On the other hand, S_{ws} pays less attention to the evidence than does the true posterior distribution $P(Z|e)$, because the sampled values for each Z_i ignore evidence among Z_i's non ancestors.

vi) The likelihood weight w makes up for the difference between the actual and desired sampling distribution. The weight for a given sample 'x', composed from Z and 'e', is the product of the likelihood for each evidence variable given its parents (some or all of which may be among the Z_{is}) :

$$w(Z, e) = \prod_{i=1}^{m} P(e_i / \text{parents}(E_i)) \quad \ldots (8.1.7)$$

vii) Multiplying equations (8.1.4) and (8.1.5), we see that the weighted probability of a sample has the particularly convenient form,

$$S_{ws}(Z, e)\, W(Z, e) = \prod_{i=1}^{l} P(y_i | \text{parents}(Y_i))$$

$$\prod_{i=1}^{m} P(e_i | \text{parents}(E_i)) = P(y, e) \quad \ldots (8.1.8)$$

Because the two products cover all the variables in the network, allowing us to use equation (8.1.1) for the joint probability.

viii) It can be easy to show that likelihood weighting estimates are consistent. For any particular value x of X, the estimated posterior probability can be calculated as follows :

$$P(x|e) = \alpha \sum_{y} N_{ws}(x, y, e)\; w(x, y, e)$$

from LIKELIHOOD-WEIGHTING.

$$\approx \alpha' \sum_{y} S_{ws}(x, y, e)\; w(x, y, e)$$

from large N.

$$= \alpha' \sum_{y} P(x, y, e)$$ by equation (8.1.8)

$$= \alpha' P(x, e) = P(x|e)$$

Hence, likelihood weighting returns consistent estimates.

5) Performance of algorithm :

i) Likelihood weighting uses all the samples generated therefore it can be much more efficient than rejection sampling.

ii) It will, suffer a degradation in performance as the number of evidence variables increases.

iii) Because most samples will have very low weights and hence the weighted estimate will be dominated by the tiny fraction of samples that accord more than an infinitesimal likelihood to the evidence.

iv) The problem is exacerbated if the evidence variables occur late in the variable ordering, because then the samples will be the simulations that bear little resemblance to the reality suggested by the evidence.

8.1.4.4 Markov Chain Monte Carlo (MCMC) Algorithm for Inference in Bayesian Networks

Working of MCMC algorithm

1) MCMC generates each event by making a random change to the preceding event. It is therefore helpful to think of the network as being in a particular current state specifying a value for every variable.

2) The next state is generated by randomly sampling a value for one of the non - evidence variables X_i, conditioned on the current values of the variables in the Markov blanket of X_i.

3) MCMC therefore wonders randomly around the state space - the space of possible complete assignments flipping one variable at a time, but keeping the evidence variables fixed.

4) For example : Consider the query **P**(rain | Sprinkler = true, WetGrass = true) applied to the network of Fig. 8.1.8 (a). The evidence variables 'Sprinkler' and 'WetGrass' are fixed to their observed values and the hidden variables 'Cloudy' and 'Rain' are initialized randomly. Let us say to true and false respectively. Thus, the initial state is [true, true, false, true]. Now the following steps are executed repeatedly :

   a) Cloudy is sampled, given the current values of its Markov blanket variables : in this case, we sample from **P**(Cloudy | Sprinkler = true, Rain = false), (Shortly we will show how to calculate this distribution). Suppose the result is Cloudy = false. Then the new current state is [false, true, false, true].

   b) Rain is sampled, given the current values of its Markov blanket variables : In this case we sample from **P**(Rain | Cloudy = false, Sprinkler = true, WetGrass = true). Suppose this yields Rain = true. The new current state is [false, true, true, true].

5) Each state visited during this process is a sample that contributes to the estimate, for the query variable Rain. If the process visits 20 states where Rain is true and 60 states where Rain is false, then the answer to the query is NORMALIZE (<20, 60>) = <0.25, 0.75>).

6) The complete algorithm is shown as follows :-

Function MCMC-ASK (X, e, bn, N) returns an estimate of $P(X|e)$

local variables : N[X], a vector of counts over X, initially zero.

Z, the nonevidence variables in bn.

X, the current state of the network, initially copied from e.

initialize X with random values for the variables in Z.

for j = 1 to N do

$N[x] \leftarrow N[x] + 1$ where x is the value of X in x.

for each Z_i in Z do

sample the value of Z_i in X from $\mathbf{P}(Z_i|mb(Z_i))$ given the value of MB (Zi) in X return NORMALIZE (N[X]).

Why MCMC works

1) It should be noticed that the sampling process settles into a "dynamic equilibrium" in which the long-run fraction of time spent in each state is exactly proporitional to its posterior probability.

2) This remarkable property follows from the specific **transition probability** with which the process moves from one state to another, as defined by the conditional distribution given the Markov blanket of the variable being sampled.

3) Let $q(X \rightarrow X')$ be the probability that the process makes a transition from state X to X′. This transition probability defines what is called a **Markov chain** on the state space.

4) Now suppose that we run the Markov chain for 't' steps, and let $\pi_t(X)$ be the probability that the system is in state X at time 't'.

   - Similarly, let $\pi_{t+1}(X')$ be the probability of being in state X′ at time 't+1.'
   - Given $\pi_t(X)$
   - We can calculate $\pi_{t+1}(X')$ by summing for all states the system could be in at time 't'.
   - The probability of being in that state is the times the probability of making the transition to $X' \rightarrow \pi_{t+1}(X') = \sum_X \pi_t(X)\, q(X \rightarrow X')$

5) We will say that the chain has reached its **stationary distribution** if $\pi_t = \pi_{t+1}$.

   Call this stationary distribution π ; its defining equation is therefore,

$$\pi(X') = \sum_X \pi(X) \quad q(X \rightarrow X') \text{ for all } X' \qquad \ldots (8.1.9)$$

Under certain standard assumptions about the transition probability distribution q, there is exactly one distribution π satisfying this equation for any given q.

6) Above equation (8.1.8) can be read as saying that the expected "outflow" from each state (i.e., its current "population") is equal to the expected "inflow" from all the state.

One way to satisfy this relationship is, the expected flow between any pair of states is the same in both direction. This is the property of **detailed balance** :

$$\pi(X)\ q(X \to X') = \pi(X')\ q\ (X' \to X) \text{ for all } X, X'. \quad \ldots (8.1.10)$$

7) It can be shown that detailed balance implies stationarily simply by summing over X in equation (8.1.9). We have,

$$\sum_x \pi(X)\ q\ (X \to X') = \sum_x \pi(X')\ q\ (X' \to X) = \pi(X') \sum_x q\ (X' \to X) = \pi(X').$$

8) To show that the transition probability $q(X \to X')$ defined by the sampling step in MCMC-ASK satisfies the detailed balance equation with a stationary distribution equal to $P(X|e)$. (The true posterior distribution on the hidden variables).

This is done in two steps -

a) First, we will define a Markov chain in which each variable is sampled conditionally on the current values of all the other variables, and we will show that this satisfies detailed balance.

b) We will simply observe that, for Bayesian networks, doing that is equivalent to sampling conditionally on the variable's Markov blanket.

- Let X_i be the variable to be sampled, and let $\overline{X}_i$ be all the hidden variables other than X_i. Their values in the current state are x_i and $\overline{x}_i$. If we sample a new x_i' for X_i conditionally on all the other variables, including the evidence, we have ,
  $q(X \to X') = q(x_i, \overline{X}_i) \to (x_i', \overline{x}_i) = P(x_i' | \overline{x}_i, e)$.
- This transition probability is called **Gibbs sampler** and is a particularly convenient form of MCMC. Now we show that the Gibbs sampler is, in detailed balance with the true posterior.
  $\pi(X)\ q(X \to X') = P(X|e)\ P(x_i'|\overline{x}_i, e) = P(x_i, \overline{X}_i|e)\ P(x_i'|\overline{X}_i, e)$
  $= P(x_i|\overline{X}_i, e)\ P(\overline{X}_i|e)\ P(x_i'|\overline{X}_i, e)$ (using the chain rule of first term).
  $= P(x_i|\overline{X}_i, e)\ P(x_i', \overline{X}_i|e)$ (using the chain rule backwards)
  $= \pi(X')\ q\ (X' \to X)$.
- Note that, a variable is independent of all other variables given its Markov blanket; hence
  $P(x_i' + \overline{X}_i, e) = P(x' | mb(X_i))$
  where mb (X_i) denotes the values of the variables in X_i's Markov blanket, MB (X_i).

The probability of a variable, given its Markov blanket, is proportional to the probability of the variable given its parents times the probability of each child given its respective parents :-

$$P(x' \mid mb(X_i)) = \alpha\ P(x' \mid Parents(X_i)) \times \prod_{Y_j \in children(x_i)} P(y_j \mid Parents(Y_j)) \quad \ldots (8.1.11)$$

- Hence, to flip each variable X_i, the number of multiplications required is equal to the number of X_i's children.

9) We have discussed here simple variant of MCMC, namely the Gibbs sampler. In its most general form, MCMC is a powerful method for computing with probability models and many variants have been developed, including the simulated annealing algorithm presented.

8.2 Other Techniques for Uncertain Reasoning

GTU : Summer-18,19,20, Winter-19

There are two basic reasons why probability can fail :-

1) One common view is that probability theory is essentially numerical, whereas human judgemental reasoning is more "qualitative".
2) Probabilistic Reasoning did not scale up because of the exponential number of probabilities required in the full joint distribution.

8.2.1 Default Reasoning

- We can do qualitative reasoning using technique like default reasoning.
- Default reasoning treats conclusions not as "believed to a certain degree", but as "believed until a better reason is found to believe something else".
- We have discussed default reasoning in detail in chapter 7.

8.2.2 Rule-based

1) This approach hope to build on the success of logical rule-based systems, but add a sort of "Fudge factor" to each rule to accommodate uncertainty. These methods were developed in the mid-1970s and formed the basis for a large number of expert systems in medicine and other areas.
2) Rule-based systems emerged from early work on practical and intuitive systems for logical inference.
3) Logical systems in general, and logical rule-based systems in particular have three desirable properties :
   i) **Locality :** In logical systems, whenever we have a rule of the form $A \Rightarrow B$, we conclude B, given evidence A, without worrying about any other rules. In probabilistic systems, we need to consider all the evidence in the Markov blanket.

ii) **Detachment** : Once a logical proof is found for a proposition B, the proposition can be used regardless of how it was derived. That is, it can be detached from its justification. In dealing with probabilities, on the other hand, the source of the evidence for a belief is important for subsequent reasoning.

iii) **Truth functionality** : In logic, the truth of complex sentences can be computed from the truth of the components. Probability combination does not work this way, except under strong global independence assumptions.

4) There have been several attempts to devise uncertain reasoning schemes that retain these advantages. The idea is to attach degrees of belief to propositions and rules and to devise purely local schemes for combining and propagating those degrees of belief. The schemes are also truth functional.

For example : The degree of belief in $A \vee B$ is a function of the belief in A and the belief in B.

5) Problem associated with rule based methods :-

i) The properties of locality, detachment, and truth-functionality are simply not appropriate for uncertain reasoning.

ii) Let us look at truth functionality first. Let H_1, be the event that a fair coin flip comes up heads. Let T_1 be the event that the coin comes up tails on that same flip, and let H_2 be the event that the coin comes up heads on a second flip. Clearly, all three events have the same probability, 0.5, and so a truth functional system must assign the same belief to the disjunction of any two of them. But we can see that the probability of the disjunction depends on the events themselves and not just on their probabilities.

P (A)	P (B)	P (A ∨ B)
$P(H_1) = 0.5$	$P(H_1) = 0.5$	$P(H_1 \vee H_1) = 0.50$
	$P(T_1) = 0.5$	$P(H_1 \vee T_1) = 1.00$
	$P(H_2) = 0.5$	$P(H_1 \vee H_2) = 0.75$

It gets worse when we chain evidence together. Truth-functional systems have rules of the form $A \rightarrow B$ that allow us to compute the belief in B as a function of the belief in the rule and the belief in A. Both forward and backward-chaining systems can be devised. **The belief in the rule is assumed to be constant and is usually specified by the knowledge engineer.**

For example : as $A \rightarrow_{0.9} B$

Consider the WetGrass situation. If we wanted to be able to do both causal and diagnostic reasoning, we would need the two rules,

Rain → WetGrass and WetGrass → Rain

These two rules form a feedback loop : Evidence for Rain increases the belief in WetGrass, which in turn increases the belief in Rain even more. Clearly, uncertain reasoning systems need to keep track of the paths along which evidence is propagated.

Inter-causal reasoning (for explaining away) is also tricky. Consider what happens when we have the two rules,

Sprinkler → WetGrass and WetGrass → Rain.

Suppose we see that the Sprinkler is on. Chaining forward through our rules, this increases the belief that the grass will be wet, which in turn increases the belief that it is raining. But this is ridiculous; the fact that the sprinkler is on explains away the wet grass and should reduce the belief in rain. A truth functional system acts as if it also believes Sprinkler → Rain.

6) If task is restricted and rules are engineered carefully then truth-functional systems work well.

8.2.3 Ignorance

1) One area that we have not addressed so far is the question of ignorance, as opposed to uncertainty. Consider the flipping of a coin. If we know that the coin is fair, then a probability of 0.5 for heads is reasonable. If we know that the coin is biased, but we do not know which way, then 0.5 is the only reasonable probability. The two cases are different, yet probability seems not to distinguish them. The Dempster-Shafer theory uses interval-valued degrees of belief to represent an agent's knowledge of the probability of a proposition.

2) Representing ignorance using Dempster-Shafer theory :-

   i) The Dempster-Shafer theory is designed to deal with the distinction between **uncertainty** and **ignorance**.

   ii) Rather than computing the probability of a proposition, it computes the probability that the evidence supports the propositions.

   iii) This measure of belief is called a **belief function**, written Bel(X).

   iv) We return to coin flipping for an example of belief functions. Suppose a shady character comes up to you and offers to bet you ₹ 10 that this coin will come up heads on the next flip. Given that the coin might or might not be fair, what belief should you ascribe to the event that it comes up heads? Dempster-Shafer theory says that because you have no evidence either way,

you have to say that the belief Bel(Heads) = 0 and also that Bel(¬ Heads) = 0. This make Dempster-Shafer reasoning systems skeptical in a way that has some intuitive appeal.

v) Now suppose you have an expert at your disposal who testifies with 90 % certainty that the coin is fair (i.e he is 90 % sure that P(Heads) = 0.5). Then Dempster-Shafer theory gives Bel(Heads) = 0.9×0.5 = 0.45 and likewise Bel (¬ Heads) = 0.45. There is still a 10 percentage point "gap" that is not accounted for by the evidence.

vi) 'Dempster's rule' (Dempster, 1968) shows how to combine evidence to give new values for Bel, and Shafer's work extends this into a complete computational model.

vii) Problems associated with Dempster-Shafer theory :-

1. There is a problem in connecting beliefs to actions.
2. With probabilities, decision theory says that if P(Heads) = P(¬ Heads) = 0.5 then (assuming that winning ₹ 10 and losing ₹ 10 are considered equal magnitude opposites). The reasoner will be indifferent between the action of accepting and declining the bet.
3. The Dempster-Shafer reasoner has Bel(¬ Heads) = 0 and thus no reason to accept the bet, but then it also has Bel(Heads) = 0 and thus no reason to decline it. Thus, it seems that the Dempster-Shafer reasoner comes to the same conclusion about how to act in this case.
4. Unfortunately, Dempster-Shafer theory allows no definite decision in many other cases where probabilistic inference does yield a specific choice.

viii) One inter-pretation of Dempster-Shafer theory is that it defines a probability interval, the interval for Heads is [0,1] before our expert testimony and [0.45, 0.55] after. The width of the interval might help in deciding when we need to acquire more evidence. It can tell you that the expert's testimony will help you if you do not know whether the coin is fair, but will not help you if you have already learned that the coin is fair. However, there are no clear guidelines for how to do this, because there is no clear meaning for what the width of an interval means.

ix) In the Bayesian approach, this kind of reasoning can be done easily by examining how much one's belief would change if one were to acquire more evidence.

For example : Knowing whether the coin is fair would have a significant impact on the belief that it will come up heads, and detecting an asymmetric weight would have an impact on the belief that the coin is fair. A complete Bayesian model would include probability estimates for factors such as these,

allowing us to express our "ignorance" in terms of how our beliefs would change in the face of future information gathering.

8.2.4 Vagueness

1) Probability makes the same ontological commitment as logic :- That events are true or false in the world, even if the agent is uncertain as to which is the case. Researchers in fuzzy logic have proposed an ontology that allows vagueness :- That an event can be "Sort of" true. Vagueness and uncertainty are in fact orthogonal issues.

2) Representing vagueness : **(Fuzzy sets and fuzzy logic)** :

   i) Fuzzy set theory is used for specifying how well an object satisfies a vague description.

   ii) For example : Consider the proposition "Anil is tall". Is this true, if Anil is 5'10"? most people would hesitate to answer "true" or "false", preferring to say, "sort of".

   iii) Note that this is not a question of uncertainty about the external world. We are sure of Anil's height. The issue is that the linguistic term "tall" does not refer to a sharp demarcation of objects into two classes and there are degrees of tallness.

   iv) Due to this reason, fuzzy set theory is not a method for uncertain reasoning at all. Rather, fuzzy set theory treats Tall as a fuzzy predicate and says that the truth value of Tall(Anil) is a number between 0 and 1, rather than being just true or false.

   v) The name "fuzzy set" derives from the interpretation of the predicate as implicitly defining a set of its members - a set that does not have sharp boundaries.

   vi) **Fuzzy logic :**

   1. Fuzzy logic is a method for reasoning with logical expressions describing membership in fuzzy sets.
   2. For example : The complex sentence Tall(Anil) $\wedge$ Heavy(Anil) has a fuzzy truth value that is a function of the truth values of its components.
   3. The standard rules for evaluating the fuzzy truth, T, of a complex sentence are

      $T(A \wedge B) = \min(T(A), T(B))$

      $T(A \vee B) = \max(T(A), T(B))$

      $T(\neg A) = 1 - T(A)$

4. Fuzzy logic is therefore a truth-functional system - a fact that causes serious difficulties.
5. For example : Suppose that T (Tall(Anil)) = 0.6 and T (Heavy(Anil)) = 0.4. Then we have T (Tall(Anil)) $\wedge$ T (Heavy(Anil)) = 0.4. Which seems reasonable but we also get the result T (Tall(Anil)) $\wedge \neg$ (Tall(Anil)) = 0.4 which does not.
6. The problem arises from the inability of a truth-functional approach to take into account the correlations or anti-correlations among the component propositions.

vii) **Fuzzy control :**

1. Fuzzy control is methodology for constructing control systems in which the mapping between real-valued input and output parameters is represented by fuzzy rules.
2. Fuzzy control has been very successful in commercial products such as automatic transmissions, video cameras, and electric shavers.
3. These applications enjoy success may be because they have small rule bases, no chaining of inferences and tunable parameters that can be adjusted to improve the system's performance. The fact that they are implemented with fuzzy operators might be incidental to their success; the key is simply to provide a concise and intuitive way to specify a smoothly interpolated, real-world function.

viii) fuzzy logic can be represented in terms of probability theory. One idea is to view assertions such as "Anil is Tall" as discrete observations, made concerning a continuous hidden variable, Anil's actual height. The probability model specifies P(observer says Anil is tall/Height), using a probit distribution. A posterior distribution over Anil's height can then be calculated in the usual way, for example if the model is part of a hybrid Bayesian Network.

ix) Fuzzy predicates can also be given a probabilistic interpretation in terms of random sets - that is, random variables whose possible values are sets of objects.

For example : Tall is random set whose possible values are sets of people. The probability P(Tall = S_1), where S_1 is some particular set of people, is the probability that exactly the set would be identified as "tall" by an observer. Then the probability that "Anil is tall" is the sum of the probabilities of all the sets of which Anil is a member.

x) Both the hybrid Bayesian network approach and the random sets approach appear to capture aspects of fuzziness without introducing degrees of truth. But they can not handle proper representation of linguistic observations and continuous quantities that have been neglected by most outside the fuzzy community.

Answer in Brief

1. *Explain in detail probabilistic reasoning system with example.* **(Refer section 8.1.1)**
2. *Explain method of handling approximate inference in Bayesian networks.* **(Refer section 8.1.4)**
3. *Explain the need of fuzzy set and fuzzy logic with example.* **(Refer section 8.2.1)**
4. *Define Dempster-Shafer theory.*

Ans. : The Dempster-Shafer theory is designed to deal with the distinction between uncertainty and ignorance.

Rather than computing the probability of a proposition, it computes the probability the evidence that supports the proposition.

5. *Define : Baye's theorem.*

Ans. : In probability theory and applications, Baye's theorem (alternatively called as Baye's law or Bayes rule) links a conditional probability to its inverse.

$$P(b \mid a) = \frac{P(a \mid b)\, P(b)}{P(a)}$$

This equation is called as Baye's Rule or Baye's Theorem.

6. *What is reasoning by default ?*

Ans. : We can do qualitative reasoning using technique like default reasoning.

Default reasoning treats conclusions not as "believed to a certain degree", but as "believed until a better reason is found to believe something else".

7. *What are the logics used in reasoning with uncertain information ?*

Ans. : There are two approaches that can be taken for reasoning with uncertain information in which logic is used.

Non-monotonic logic is used in default reasoning process. Default reasoning also uses other type or logic called as default logic.

The second approach towards reasoning is vagueness which uses fuzzy logic. Fuzzy logic is a method for reasoning with logical expressions describing membership in fuzzy sets.

8. *Define prior probability.*

Ans. : The prior (unconditional) probability is associated with a proposition 'a'.

The prior probability is the degree of belief accorded to a proposition in the absence of any other information.

It is written as P(a). For example, the probability that, Ram has cavity = 0.1, then the prior priobility is written as,

P(Cavity = true) = 1 or P(cavity) = 0.1

9 State the types of approximation methods.

Ans. : For approximate inferencing randomize sampling algorithm (Monte Carlo Algorithm) is used. There are two approximation methods that are used in randomize sampling algorithm which are 1) Direct sampling algorithm and 2) Markov chain sampling algorithm.

In direct sampling algorithm samples are generated from known probability distribution. In Markov chain sampling each event is generated by making a random change to the preceding event.

10. What do you mean by hybrid Bayesian network ?

Ans. : A network with both discrete and continuous variables is called as hybrid Bayesian network. In hybrid Bayesian network, for representing the continuous variable its discretization is done in terms of intervals because it can have infinite values.

For specifying the hybrid network two kinds of distribution are specified. The conditional distribution for a continuous variable given discrete or continuous parents and the conditional distribution for a discrete variable given continuous parent.

11. Define computational learning theory.

Ans. : The computational learning theory is a mathematical field related to the analysis of machine learning algorithms.

The computational learning theory is used in the evaluation of sample complexity and computational complexity. Sample complexity targets the issue that, how many training examples are needed to learn a successful hypothesis ? The computational complexity evaluates that how much computational effort is needed to learn a successful hypothesis ?

In addition to performance bounds, computational learning theory also deals with the time complexity and feasibility of learning.

12. Give the full specification of Bayesian network.

Ans. : Bayesian network : Definition : It is a data structure which is a graph, in which each node is annotated with quantitative probability information.

The nodes and edges in the graph are specified as follows :-

1) A set of random variables makes up the nodes of the network. Variables may be discrete or continuous.
2) A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, then X is said to be a parent of Y.
3) Each node X_i, has a conditional probability distribution $P(X_i \mid \text{Parents}(X_i))$ that quantifies the effect of the parents on the node.
4) The graph has no directed cycles (and hence is a directed, acyclic graph, or DAG).

The set of nodes and links is called as **topology of the network**.

8.3 University Questions with Answers

Summer - 18

Q.1 *Differentiate Fuzzy logic and Crisp logic.* **(Refer sections 8.2)** [3]

Q.2 *Discuss various defuzzification methods.* **(Refer section 8.2)** [4]

Q.3 *Discuss Bayesian network and its application.* **(Refer sections 8.1)** [4]

Summer - 19

Q.4 *Differentiate fuzzy logic and crisp logic. Also describe set operations on fuzzy and crisp logic.* **(Refer section 8.2)** [7]

Q.5 *Discuss various defuzzification methods.* **(Refer section 8.2)** [4]

Q.6 *Discuss Bayesian network and its application.* **(Refer section 8.1)** [4]

Winter - 19

Q.7 *Explain probabilistic inference in Bayesian networks with the help of a suitable example.* **(Refer section 8.1)** [4]

Q.8 *Explain the difference between Boolean and fuzzy set membership using a suitable example.* **(Refer section 8.2)** [3]

Summer - 20

Q.9 *What is the importance of fuzzy logic ? How do you perform union, intersection and complement operation on the fuzzy sets ?* **(Refer section 8.2)** [4]

❑❑❑