Inferences in Bayesian Networks — Junction Tree Algorithms

Yuqing Tang

Doctoral Program in Computer Science
The Graduate Center
City University of New York
ytang@cs.gc.cuny.edu

October 20th, 2010

Outline

The query

$$
\begin{aligned}
P(X|\mathbf{E}) &= \frac{P(X,\mathbf{E})}{P(\mathbf{E})} \\
&= \alpha\, P(X,\mathbf{E}) \\
P(X,\mathbf{E}) &= \Sigma_{\mathbf{Y}} P(X,\mathbf{Y},\mathbf{E}) \\
\alpha &= \frac{1}{P(\mathbf{E})} = \frac{1}{\Sigma_X P(X,\mathbf{E})}
\end{aligned}
$$

- X is the query variable
- **E** is the set of evidence variables
- **Y** is the set of hidden variables

Outline

1. Introduction
2. Junction Algorithm
3. Stochastic Approximation
4. Summary

An Algebra of Potentials

A potential ϕ is a real-valued function over the domain of finite variables:

$$\phi : Domain(Variables(\phi)) \to \mathcal{R}$$[1]

Multiplication over two potentials ϕ_1 and ϕ_2 is defined as

$$\phi = \phi_1 \cdot \phi_2$$

such that

- $\phi : Domain(Variables(\phi)) \to \mathcal{R}$
- $Variables(\phi) = Variables(\phi_1) \cup Variables(\phi_2)$
- $\phi(\mathbf{x}) = \phi_1(\mathbf{x}_1) \cdot \phi_2(\mathbf{x}_2)$ where
  - $\mathbf{x} \in Domain(Variables(\phi))$ is an value assignment to the variables in $Varaibles(\phi)$
  - $\mathbf{x}_1 \in Domain(Variables(\phi_1))$ is a sub-assignment of $\mathbf{x}$
  - $\mathbf{x}_2 \in Domain(Variables(\phi_2))$ is a sub-assignment of $\mathbf{x}$
- Distributed law: If $X \notin Variables(\phi_1)$, then $\Sigma_X \phi_1 \phi_2 = \phi_1 \Sigma_X \phi_2$.

[1] The finite set of variables, $Variables(\phi)$, is also called the domain of ϕ in [Jensen and Nielsen, 2007, Chapter 1].

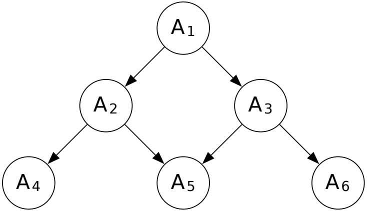
Intermediate Probability Calculations in Potentials

- Probabilities in potential $P(\mathbf{x}) : Domain(\mathbf{x}) \rightarrow [0, 1]$
- Conditional probabilities in potential $P(\mathbf{x}|\mathbf{y}) : Domain(\mathbf{x} \cup \mathbf{y}) \rightarrow [0, 1]$
- Multiplying probabilities and conditional probabilities into potentials:

$$\phi = \Pi_i P(\mathbf{X}_i) \cdot \Pi_j P(\mathbf{Z}_j|\mathbf{Y}_j)$$

  Note:
  - The variable set of ϕ is the union of the variable sets of all involved probabilities
  - The resulted value $\phi(\mathbf{x})$ (not a necessary probability) for each assignment $\mathbf{x} \in Domain(Variables(\phi))$ is obtained by multiplying the probabilities for $\mathbf{x}$'s corresponding sub-assignments defined by the probability functions
- Working on potentials afterwards

A Simple Bayesian Network

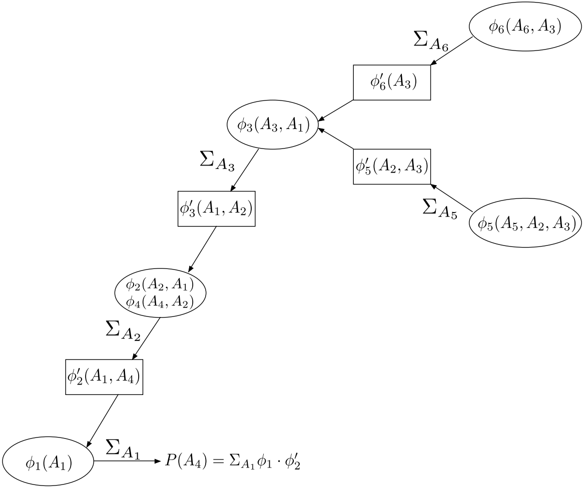


Let $\phi_1 = P(A_1)$, $\phi_2 = P(A_2|A_1)$, $\phi_3 = P(A_3|A_1)$, $\phi_4 = P(A_4|A_2)$, $\phi_5 = P(A_5|A_2, A_3)$ and $\phi_6 = P(A_6|A_3)$.

$$\begin{aligned}
P(\mathcal{U}) &= \phi_1\phi_2\phi_3\phi_4\phi_5 \\
P(A_4) &= \Sigma_{A_1,A_2,A_3,A_5,A_6}\phi_1\phi_2\phi_3\phi_4\phi_5 \\
&= \Sigma_{A_1}\phi_1(A_1)\Sigma_{A_2}\phi_2(A_2, A_1)\phi_4(A_4, A_2)\Sigma_{A_3}\phi_3(A_3, A_1) \\
&\quad \Sigma_{A_5}\phi_5(A_5, A_2.A_3)\Sigma_{A_6}\phi_6(A_6, A_3)
\end{aligned}$$

By the distributed law: If $X \notin \mathit{Variables}(\phi_1)$, then $\Sigma_X\phi_1\phi_2 = \phi_1\Sigma_X\phi_2$.

Variable Elimination Order



$$
\begin{aligned}
P(A_4) \quad = \quad & \Sigma_{A_1}\phi_1(A_1)\Sigma_{A_2}\phi_2(A_2, A_1)\phi_4(A_4, A_2)\Sigma_{A_3}\phi_3(A_3, A_1) \\
& \Sigma_{A_5}\phi_5(A_5, A_2.A_3)\Sigma_{A_6}\phi_6(A_6, A_3)
\end{aligned}
$$

Evidences in Potentials

Let $e \in Domain(X)$ be an evidence on variable X. We can define this evidence as a potential in the following form

$$e : Domain(X) \rightarrow \{0, 1\}$$

such that

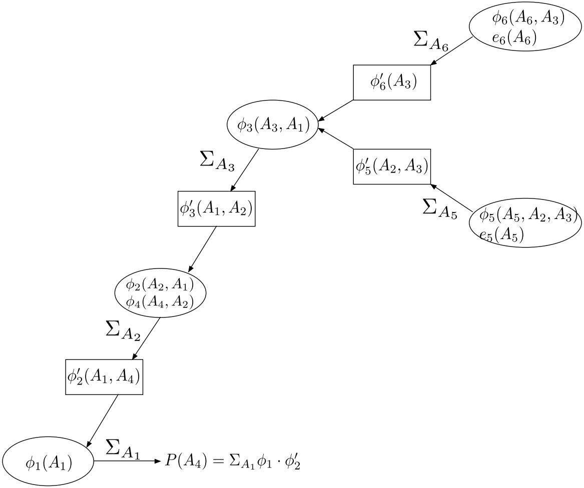
- $e(x) = 1$ if $x = e$, and
- $e(x) = 0$ if $x \neq e$.

Example

Let $P(Pollution = high) = 0.6$, $P(Pollution = low) = 0.4)$, and the evidence $e(Pollution)$ be *high*. We will have $e(high) = 1$, and $e(low) = 0$. The multiplication of the probability potential and the evidence potential becomes

$$\begin{aligned} P(Pollution = high) \cdot e(Pollution = high) &= P(Pollution = high) \\ P(Pollution = low) \cdot e(Pollution = low) &= 0 \end{aligned}$$

Variable Elimination Order with Evidences



$$
\begin{aligned}
P(A_4) \quad = \quad & \Sigma_{A_1}\phi_1(A_1)\Sigma_{A_2}\phi_2(A_2, A_1)\phi_4(A_4, A_2)\Sigma_{A_3}\phi_3(A_3, A_1) \\
& \Sigma_{A_5}\phi_5(A_5, A_2.A_3)e_5\Sigma_{A_6}\phi_6(A_6, A_3)e_6
\end{aligned}
$$

Inferences as Variable Eliminations in Potentials

Let $\Phi = \{\phi_1, \ldots, \phi_n, e_1, \ldots, e_k\}$ be the set of potentials correspond to the CPTs in a Bayesian net and the incoming events, then

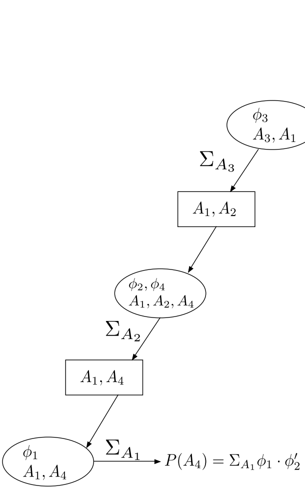
$$P(X|e_1, \ldots, e_k) = \alpha \Sigma_{\mathbf{Y}} \phi_1 \phi_2 \ldots \phi_n e_1 \ldots e_k$$

1. Looking for an elimination ordering on the hidden variables $Y_1, Y_2, \ldots, Y_M$ so that
   - the total size of the intermediate potentials' domains $\Sigma_i |Domain(Variables(\phi_i))|$ can be as small as possible,
   - namely, the total number of entries in the intermediate tables is as small as possible
2. Compute

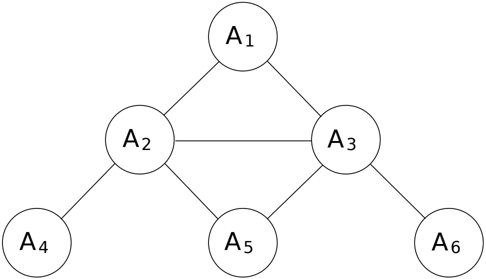
$$\Sigma_{\mathbf{Y}} \phi_1 \phi_2 \ldots \phi_n e_1 \ldots e_k =$$
$$\Sigma_{Y_1} \Pi \Phi(Y_1 \setminus Y_2, \ldots, Y_M) \Sigma_{Y_2} \Pi \Phi(Y_2 \setminus Y_3, \ldots, Y_M) \ldots \Sigma_{Y_M} \Pi \Phi(Y_M)$$

where
   - $\Phi(Y_i) = \{\phi_j | \phi_j \in \Phi \text{ and } Y_i \in Variables(\phi_j)\}$
   - $\Phi(Y_i \setminus Y_{i+1}, \ldots, Y_M) = \Phi(Y_i) \setminus \cup_{j=i+1}^{M} \Phi(Y_j)$

Domains for the Variable Elimination



Domain Graphs

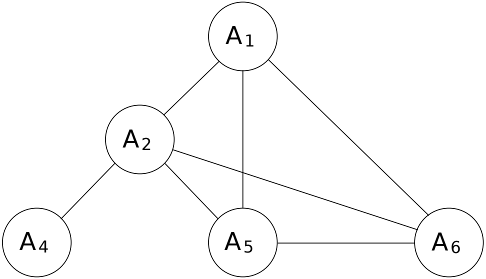


A domain graph $G(\Phi)$ for a set of potentials $\Phi = \{\phi_1, \ldots, \phi_n\}$ is an undirected graph such that

- Nodes are variables of the potentials
- There is a link between each pair of variables of the same potential $\phi_i \in \Phi$

A BN can be converted into a domain graph by inserting a moral link between every pair of nodes with a common child

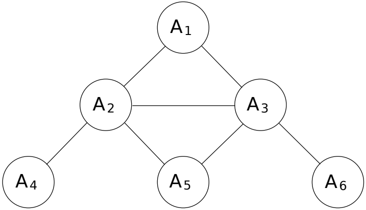
Elimination Variables in Domain Graphs



Given a domain graph $G = \langle V, E \rangle$

- To eliminate a variable Y_i, we need to work on all the potentials ϕ_js with Y_i in its variable set, denoted by $\phi_j \in \Phi(Y_i)$
- After eliminate Y_i, we obtain a new potential $\phi'_i = \Sigma_{Y_i} \Pi \Phi(Y_i)$ with all Y_i's neighbors $nb(Y_i)$ in its variable set
- Correspondingly we transform G into G' where
  - we remove Y_i and edges that involve in Y_i, and
  - we connect every pair of variables in $neb(Y_i)$

Perfect Eliminations

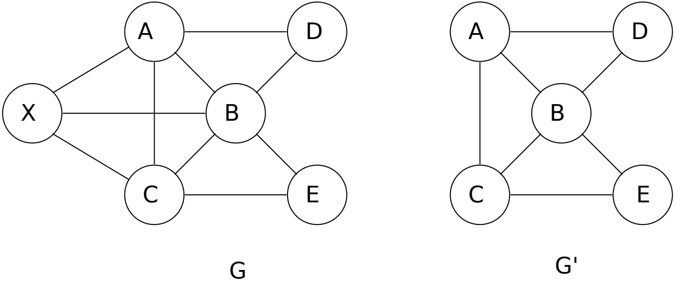


- A perfect elimination sequence is a variable order in which no new edges will be added to the domain graph during the elimination process
  - New edges that are added during the elimination process are called fill-ins
  - e.g. $A_5, A_6, A_3, A_1, A_2, A_4$, as well as $A_1, A_5, A_6, A_3, A_2, A_4$, and $A_6, A_1, A_3, A_5, A_2, A_4$ are all perfect elimination sequences

Triangulated Graphs

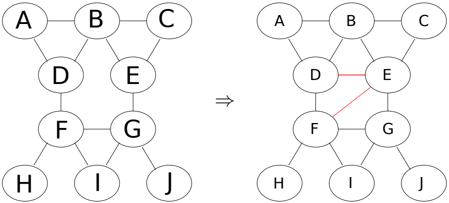
- A domain graph that has a perfect elimination is called a triangulated graph
- The domain set of an elimination sequence is the set of domains of potentials produced during the elimination in which potentials that are subsets of other potentials are removed
- All perfect elimination sequences produce the same domain set, namely the set of cliques of the domain graph
  - A set of nodes is complete if all nodes are pairwise linked
  - A complete set is a clique if it is not a subset of another complete set (a maximal complete set)

Simplicial nodes



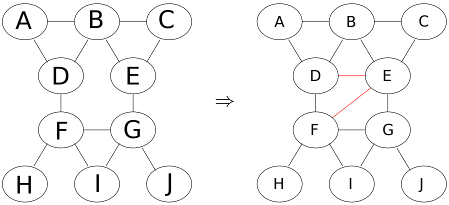
- Nodes with a complete neighbor set are called simplicial nodes
  - let $fa(X) = \{X\} \cup nb(X)$ be the family of X
  - X is a simpolicial node iff $fa(X)$ is a clique
- Elimination as domain graph transformation:
  If G is a triangulated graph, and X is a simplicial node in G, then eliminating X from G (removing all the edges connecting to X as well) results in a new triangulated graph G'.

Triangulation – Example



- Heuristics: Eliminate a node X with minimum number of family states, ie. $sz(fa(X))$
  - recall that $fa(X) = \{X\} \cup nb(X)$
  - $sz(V) = |Domain(V)|$
  - $sz(\{V_1, ..., V_n\}) = \Pi_i sz(V_i)$
- Let A, B, C, H, I, J have 2 states, D have 4 states, E have 5 states, F have 6 states, G have 7 states. Assume we have removed A, C, H, I, J already, then $sz(fa(B)) = 40$, $sz(fa(D)) = 48$, $sz(fa(E)) = 70$, $sz(fa(F)) = 168$ and $sz(fa(G)) = 210$.

Triangulation – Example (cont.)



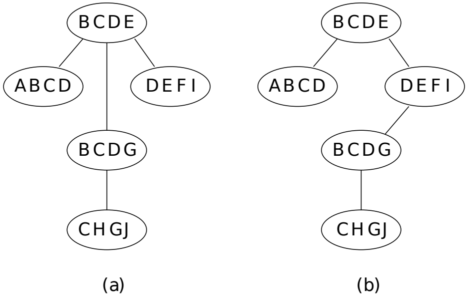
- A, C, H, I, J can be eliminiated without introducing new edges (fill-ins)
- B has the smallest family state size so choose B to eliminate, we add edge (D, E).
- Update size: $sz(fa(D)) = 120$ and $sz(fa(E)) = 140$.
- Choose D, we add (E, F)
- Now the graph is triangulated, we can eliminate the nodes without adding new edges: e.g. E, F, G in order

Triangulation Algorithm

Given a graph G

1. For each node $X \in G$
   - If there is simplicial node X in G, eliminate X
2. Compute the family size of each node G, select a node X with minimum family size and eliminate X
3. If there are no nodes left in G then terminate; otherwise go to step 1

Join Tree

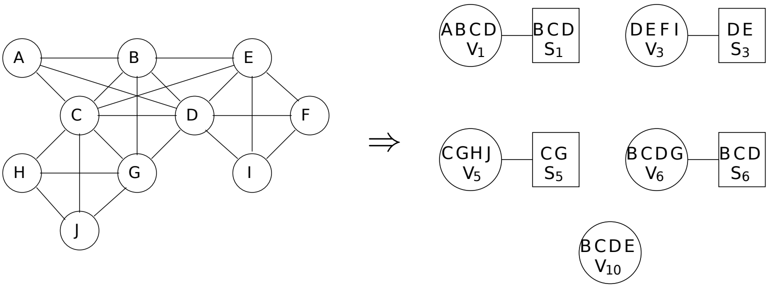


(a) A join tree; (b) not a join tree.

Definition

Let $\mathcal{G} \subseteq 2^G$ be the set of cliques from an undirected graph G, and let the cliques of $\mathcal{G}$ be organized in a tree T. Then T is a join tree if for any pair of nodes $V, W \in T$, all nodes on the path between V and W contain the intersection $V \cap W$.

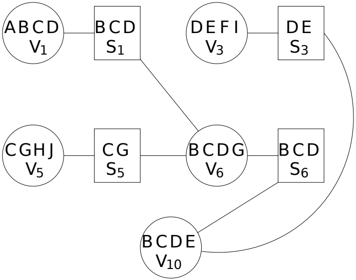
Transformation of a triangulated graph into a join tree I



1. Establish an elimination sequence for the nodes in G and add fill-ins to G if neccessary (e.g. according to a triangulation algorithm)
2. $i \leftarrow 0$
3. For each node $X \in G$, if X is a simplicial node then $fa(X)$ is a clique
   - Remove nodes $Y_{x,1}, \ldots Y_{x,K}$ from $fa(X)$ with $nb(Y_{x,k}) \subseteq fa(X)$,

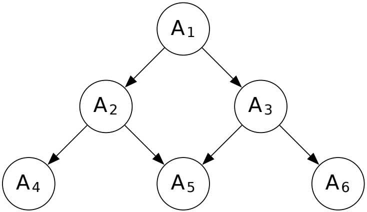
Transformation of a triangulated graph into a join tree II

- $i \leftarrow i + K$ (i is the number nodes removed so far)
- Construct a clique $V_i \leftarrow fa(X)$
- Construct a separator $S_i \leftarrow \{Y_{x,1}, \ldots Y_{x,K}\}$
- Connect V_i and S_i in the join tree T



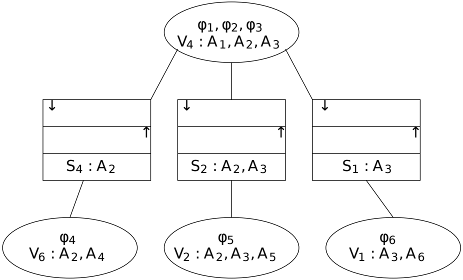
4. For each separator S_i
   - Select a V_j such that $j > i$ and $S_i \subset V_j$
   - Connect S_i to V_j in the joint tree T

Return to The Simple Bayesian Network



Let $\phi_1 = P(A_1)$, $\phi_2 = P(A_2|A_1)$, $\phi_3 = P(A_3|A_1)$, $\phi_4 = P(A_4|A_2)$, $\phi_5 = P(A_5|A_2, A_3)$ and $\phi_6 = P(A_6|A_3)$.

Junction tree

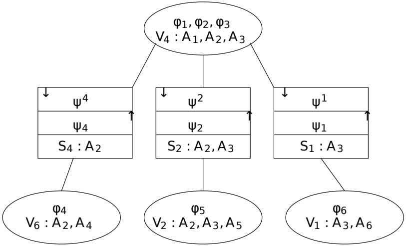


Definition

Let Φ be a set of potentials with a triangulated domain graph G. A junction tree for Φ is a join tree for G with the following addition:

- Each potential $\phi \in \Phi$ is attached to a clique containing $Variables(\phi)$
- Each link has the appropriate separator attached
- Each separator contains two mailboxes, one for each direction: $\pi \downarrow$ and $\lambda \uparrow$

Junction tree – full propagation



Now we can apply the poly-tree message-passing algorithm:

$$
\begin{aligned}
\downarrow \psi^1 &= \Sigma_{A_1,A_2}\phi_1\phi_2\phi_3 \text{ //project to } A_3 \\
\uparrow \psi_1 &= \Sigma_{A_6}\phi_6 \text{ //project to } A_3 \\
\downarrow \psi^2 &= \Sigma_{A_1}\phi_1\phi_2\phi_3 \text{ //project to } A_2, A_3 \\
\uparrow \psi_2 &= \Sigma_{A_5}\phi_5 \text{ //project to } A_2, A_3 \\
\downarrow \psi^4 &= \Sigma_{A_1,A_3}\phi_1\phi_2\phi_3 \text{ //project to } A_2 \\
\uparrow \psi_4 &= \Sigma_{A_4}\phi_4 \text{ //project to } A_2
\end{aligned}
$$

Outline

Probabilistic logical sampling

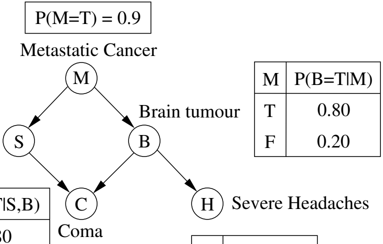
Query $P(X|\mathbf{E} = \mathbf{e})$

1. Choose a topological order $X_1, \ldots, X_N$ of the variables
2. Initialize $Count(X_i) \leftarrow 0$ for each X_i in order
3. Loop until a given number M of samples are obtained
   - For $i = 1$ to N
     - Choose a value x_i for X_i randomly w.r.t $P(X_i = x_i|Parents(X_i) = \pi)$ where π is a value vector consistent with the values chosen
   - If $x_1, \ldots, x_n$ is consistent with $\mathbf{e}$, $Count(X_i = x_i) \leftarrow Count(X_i = x_i) + 1$
4. Return $P(X = x|E = \mathbf{e}) \approx \frac{Count(X=x)}{\sum_{v \in Domain(X)} Count(X=v)}$

Some notes about probabilistic logical sampling

- A topological order satisfies that for any variables X_i and X_j in the order, if there is a path between X_i and X_j in the BN then $i < j$
- Choose a value x_k for X_i randomly
  - Generate a random number $\beta \in [0, 1]$
  - If $\Sigma_{j=0,\ldots k-1} P(X_i = v_j | X_1, \ldots, X_{i-1}) \leq \beta < \Sigma_{j=1,\ldots k} P(X_i = v_j | Parents(X_i) = \pi)$, then $X_i = v_k$ is chosen ($P(X_i = v_0 | Parents(X_i) = \pi)$ is set to 0)

Logical sampling example I



M	P(S=T|M)
T	0.20
F	0.05

M	P(B=T|M)
T	0.80
F	0.20

S	B	P(C=T|S,B)
T	T	0.80
T	F	0.80
F	T	0.80
F	F	0.05

B	P(H=T|B)
T	0.80
F	0.60

- Choose the order M, S, B, C, H
- $P(M = T) = 0.2$, a random number greater than 0.2 is generated, then F is chosen

Logical sampling example II

- Next, generate another two random numbers (assume they are less than $P(S = T|M = F)$ and $P(B = T|M = F)$ respectively), so we choose $S = T$ and $B = T$
- In a similar way, we choose $C = F$ and $H = T$ randomly
- Suppose we want to compute $P(M|H = T)$, the above case will be counted into $Count(M = F, H = T)$ but not count to $Count(M = T, H = T)$.

Likelihood Weighting

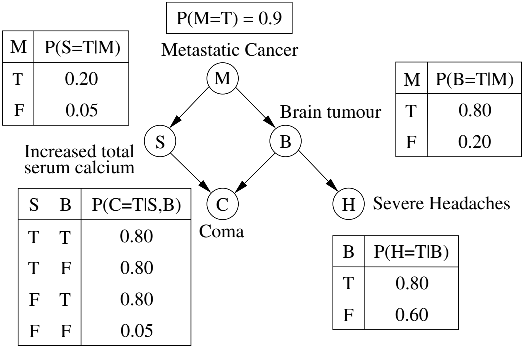
In probabilistic logical sampling, the number of samples consistent with the incoming evidences will be very small if their probability is small. We can use likelihood weighting:

Query

$$P(X|\mathbf{E} = \mathbf{e})$$

1. Choose a topological order $X_1, \ldots, X_N$ of the variables
2. Initialize $Count(X_i) \leftarrow 0$ for each X_i in order
3. Loop until a given number M of samples are obtained
   - $w \leftarrow 1$
   - For $i = 1$ to N
     - If $X_i \notin \mathbf{E}$, choose a value x_i for X_i randomly w.r.t $P(X_i = x_i|Parents(X_i) = \pi)$ where π is a value vector consistent with the values chosen
     - If $X_i \in \mathbf{E}$, $w \leftarrow w \cdot P(X_i = e_i|Parents(X_i) = \pi)$
     - $Count(X_i = x_i) \leftarrow Count(X_i = x_i) + w$
4. Return $P(X = x|E = \mathbf{e}) \approx \frac{Count(X=x)}{\Sigma_{v \in Domain(X)} Count(X=v)}$

Likelihood weighting example I



M	P(S=T|M)
T	0.20
F	0.05

M	P(B=T|M)
T	0.80
F	0.20

S	B	P(C=T|S,B)
T	T	0.80
T	F	0.80
F	T	0.80
F	F	0.05

B	P(H=T|B)
T	0.80
F	0.60

Assume the query is $P(C = T|B = T)$

- Choose the order M, S, B, C, H
- $P(M = T) = 0.2$, a random number greater than 0.2 is generated, then F is chosen

Likelihood weighting example II

- Next, we choose a value for S randomly according to $P(S = T|M = F) = 0.2$; suppose $S = T$ is chosen
- Since $P(B = T|M = T) = 0.05$, we set the likelihood weight $w \leftarrow 0.05$
- Then, assume we choose $C = T$ and $H = T$ randomly
- In this case, $Count(C = T)$ is increased by 0.05

The idea behind likelihood weighting

$$P(X = x, \mathbf{E} = \mathbf{e}) = \\ \Sigma_{Y \notin \{X\} \cup \mathbf{E}} \Pi_{X \notin \mathbf{E}} P(X | Parents(X) = \pi) \Pi_{X \in \mathbf{E}} P(X | Parents(X) = \pi)$$

- The random sample step corresponds to $\Pi_{X \notin \mathbf{E}} P(X | Parents(X) = \pi)$,
- The weighting step corresponds to $\Pi_{X \in \mathbf{E}} P(X | Parents(X) = \pi)$
- The counting step corresponds to the marginalization

Outline

Summary

- Probabilistic inference: compute the probability distribution for query variables, given evidence variables

$$P(X|\mathbf{E})$$

- BN Inference is very flexible: can enter evidence about any node and update beliefs in any other nodes.
- The speed of inference in practice depends on the structure of the network: how many loops; numbers of parents; location of evidence and query nodes.
- Inference methods
  - Exact inference by enumeration
  - Polytree message passing
  - Junction-tree algorithms for general BNs
  - Approximation inference with stochastic simulation

Acknowledgments

Lecture 6 is composed the instructor's own understanding of the subject, and materials from [Korb and Nicholson, 2003, Chapter 3] and [Jensen and Nielsen, 2007, Chapter 4] with the instructor's own interpretations. The instructor takes full responsibility of any mistakes in the slides.

References I

Finn V. Jensen and Thomas D. Nielsen.
Bayesian Networks and Decision Graphs.
Springer Publishing Company, Incorporated, 2007.

K. Korb and A. E. Nicholson.
Bayesian Artificial Intelligence.
Chapman & Hall /CRC, 2003.