CSE 250A. Principles of AI

Probabilistic Reasoning and Decision-Making

Lecture 6 – Inference in loopy BNs

Lawrence Saul
Department of Computer Science and Engineering
University of California, San Diego

Fall 2021

Outline

Inference

- **Problem**

  Given a set E of evidence nodes, and a set Q of query nodes, how to compute the posterior distribution $P(Q|E)$?

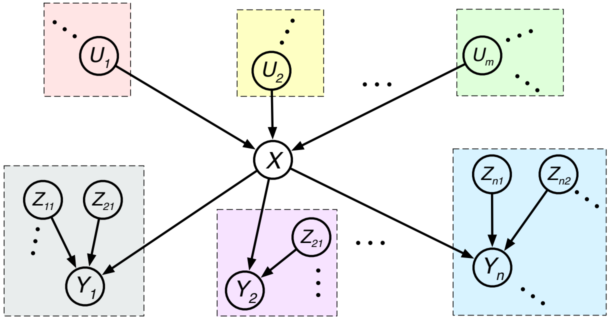
- **More precisely**

  How to express $P(Q|E)$ in terms of the CPTs $P(X_i|pa(X_i))$ of the BN, which are assumed to be given?

- **Tools at our disposal**

Bayes rule	marginal independence
marginalization	conditional independence
product rule	

Polytree algorithm

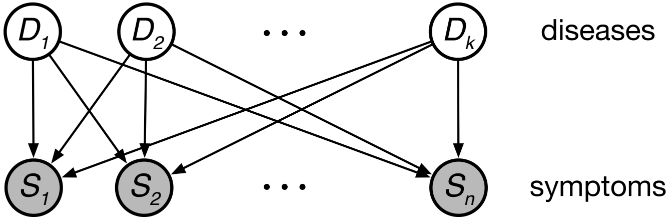


- **Polytrees** are singly connected networks — with no loops and at most one path between any two nodes.
- Exact inference is tractable in polytrees, scaling **linearly** in the size of the DAG and CPTs.

Questions?

Today

But many interesting BNs are not polytrees!



How to compute $P(D_i = 1 | S_1, S_2, \ldots, S_n)$**?**

What are general strategies for inference in these BNs?

Outline

1. **Review**
2. **Exact inference in loopy BNs**
3. **Approximate inference in loopy BNs**

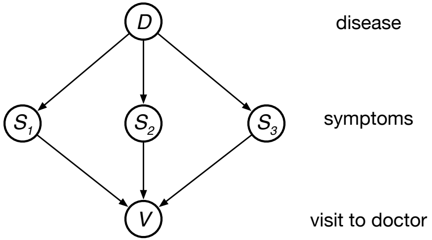
Exact inference in loopy BNs

- **Main idea**

  Can we transform a loopy BN into a polytree?
  If so, then we can run the polytree algorithm.

- **Example**

  We'll use a simple BN with binary variables to illustrate two different ways of doing this.

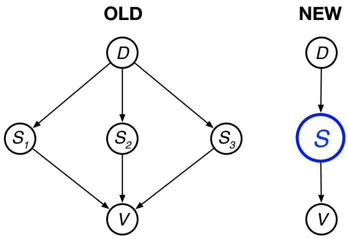


1. Node clustering

- **Key idea**

  Merge (well-chosen) nodes in the DAG to remove loops, so that what remains is a polytree.

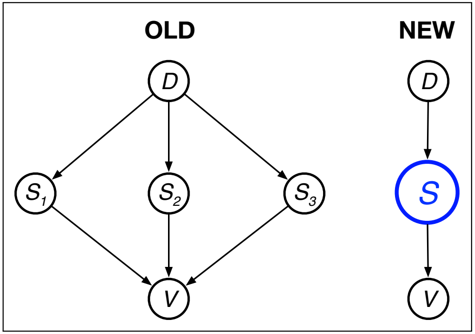
- **Example**



Cluster nodes $\{S_1, S_2, S_3\}$ into mega-node S.

Merge CPTs at these nodes into mega-CPT $P(S|D)$.

Old versus new nodes



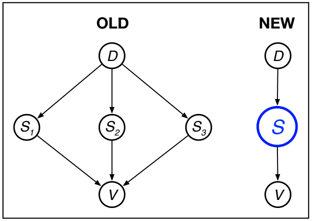
S_3	S_2	S_1	S
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**

Pro	The graph simplifies to a polytree.
Con	The node becomes (exponentially) more complex:

$$|S| = |S_1| \cdot |S_2| \cdot |S_3| = 2^3 = 8$$

Old versus new CPTs

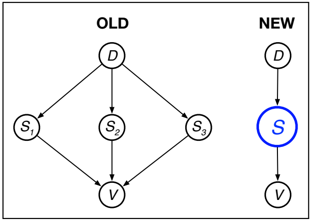
S_3	S_2	S_1	S
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**



OLD	**NEW**
$P(S_1\|D)$ $P(S_2\|D)$ $P(S_3\|D)$	$P(S\|D) = P(S_1, S_2, S_3\|D) = \prod_{i=1}^{3} P(S_i\|D)$
$P(V\|S_1, S_2, S_3)$	$P(V\|S) = P(V\|S_1, S_2, S_3)$

Worked example

S_3	S_2	S_1	S
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	**0**	**1**	**5**
1	1	0	**6**
1	1	1	**7**

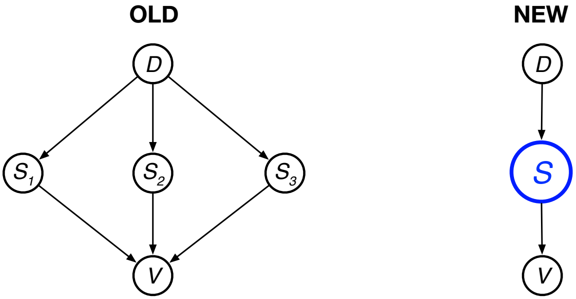


To calculate the new CPTs:

$$
\begin{aligned}
P(S=5|D=0) &= P(S_1=1, S_2=0, S_3=1|D=0) \\
&= P(S_1=1|D=0)\,P(S_2=0|D=0)\,P(S_3=1|D=0)
\end{aligned}
$$

$$
P(V=1|S=5) = P(V|S_1=1, S_2=0, S_3=1)
$$

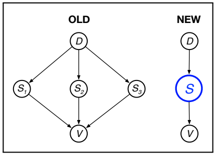
Which nodes to cluster?



In this BN, we can eyeball the right nodes to cluster.

What about in larger BNs?

General case



- **It seems simple enough:**

  Cluster nodes as needed to remove loops.
  Apply polytree algorithm to the resulting BN.

- **But there are tradeoffs:**

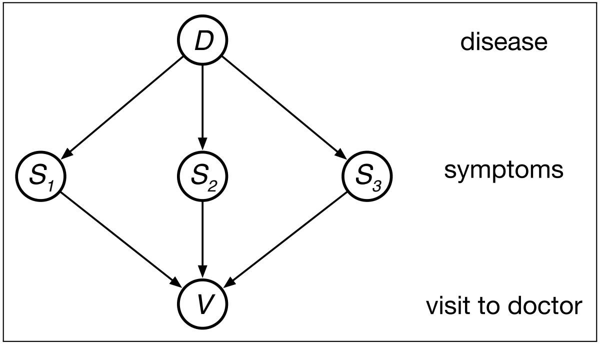
  The polytree algorithm scales linearly in the size of CPTs.
  CPTs grow exponentially when nodes are clustered.

- **Can we optimize this tradeoff?**

  Which clustering leads to maximally efficient inference?
  There is no efficient algorithm to find this!

Questions?

A different approach?



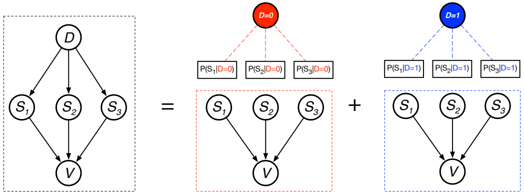
What if, instead of merging nodes, we remove them?

2. Cutset conditioning

- **Key idea**

  Remove one or more nodes by instantiating them as evidence.
  Call the polytree algorithm for each possible instantiation.

- **Example**

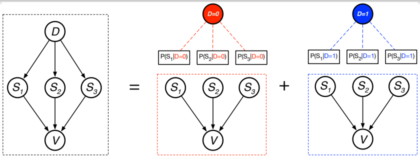


- **Definition**

  The set of instantiated nodes is called the **cutset**.
  By removing them, we cut the BN into polytrees.

Worked example

How to calculate $P(V=1)$?



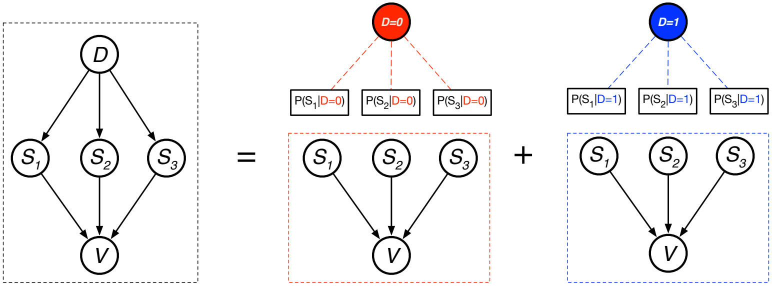
- **Run the polytree algorithm twice:**

  (1) Compute $P(V=1|D=0)$ from the left polytree.
  (2) Compute $P(V=1|D=1)$ from the right polytree.

- **Combine the results:**

$$\begin{aligned} P(V=1) &= \sum_d P(D=d, V=1) & \boxed{\textbf{marginalization}} \\ &= \sum_d P(D=d)\, P(V=1|D=d) & \boxed{\textbf{product rule}} \\ &= P(D=0)P(V=1|D=0) + P(D=1)P(V=1|D=1) \end{aligned}$$

How to choose the cutset?



In this BN, we can eyeball the right node to instantiate.

What about in larger BNs?

General case

- **It seems simple enough:**

  Instantiate nodes as needed to remove loops.
  Apply polytree algorithm to the resulting BNs.

- **But there are tradeoffs:**

  How many times must we run the polytree algorithm?
  This number grows exponentially with the size of the cutset.

- **Can we optimize this tradeoff?**

  What is the minimal cutset for maximally efficient inference?
  There is no efficient algorithm to compute this!

Questions?

Outline

1. **Review**
2. **Exact inference in loopy BNs**
3. **Approximate inference in loopy BNs**

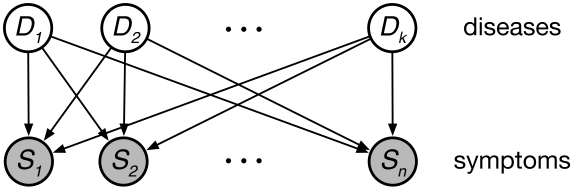
Motivation

- **Formal results**

  Exact inference in belief networks is **NP-hard**.
  Actually it is **#P-hard** (even worse).

- **Practical tools**

  But many large loopy BNs remain useful models.
  In these BNs, we must resort to approximate methods.



Sampling a discrete random variable $X \sim P(X)$

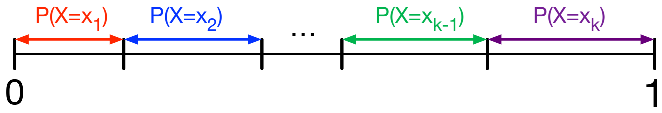
Let X be a discrete random variable with probabilities $P(X=x_i)$. Suppose we can generate random numbers uniformly in $[0, 1]$.

- **Problem**

  How to sample values of X so that repeated samples are distributed according to $P(X)$?

- **Solution**

  Note that $P(X=x_i)$ defines a partition of unity, which maps a random number $r \in [0, 1]$ into a discrete value of X.



Sampling from the joint distribution in a discrete BN

Let $\{X_1, X_2, \ldots, X_m\}$ be discrete random variables in a BN.

- **Problem**

  How to sample values of $\{X_1, X_2, \ldots, X_m\}$ so that repeated samples are distributed according to $P(X_1, X_2, \ldots, X_m)$?

- **Solution**

  The BN defines a **generative model** with joint distribution

$$P(X_1, X_2, \ldots, X_m) = \prod_{i=1}^{m} P(X_i|\mathrm{pa}(X_i)).$$

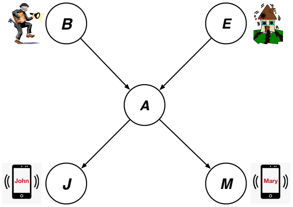
  To draw samples, we can simply take $X_i \sim P(X_i|\mathrm{pa}(X_i))$. This is best illustrated by example.

Alarm example

- **Joint sample**

  To draw a joint sample $\{b, e, a, j, m\}$ from $P(B, E, A, J, M)$, it is enough to draw the individual samples:

$$\begin{aligned} b &\sim P(B) \\ e &\sim P(E) \\ a &\sim P(A|B=b, E=e) \\ j &\sim P(J|A=a) \\ m &\sim P(M|A=a) \end{aligned}$$



- **Convergence in the limit**

$$P(b, e, a, j, m) = \lim_{N\to\infty} \frac{\mathrm{count}(B=b, E=e, A=a, J=j, M=m)}{N}.$$

Questions?

Approximate inference

- **Problem**

  Let Q denote a set of query nodes.
  Let E denote a set of evidence nodes.
  How can we ~~calculate~~ **estimate** $P(Q|E)$?

- **Challenge**

  While easy to sample from the BN's joint distribution,
  it may be much harder to sample directly from $P(Q|E)$.

- **Solutions**

  1. rejection sampling (very inefficient)
  2. likelihood weighting (more efficient)
  3. MCMC (most efficient)

1. Rejection sampling (in general)

- **Problem**

  How to estimate the relative area of the bullseye to the board?

- **One simple approach**

  1. Throw N_0 darts randomly at the wall.
  2. Ignore (or **reject**) those that miss the board.
  3. Count the number N_1 that hit the board.
  4. Count the number N_2 that hit the bullseye.
  5. Take the ratio of these counts:

$$\text{relative area} = \lim_{N_0 \to \infty} \left(\frac{N_2}{N_1} \right) \quad \text{where} \quad N_2 \leq N_1 \leq N_0$$

Rejection sampling in BNs

- **Problem**

  Let Q denote a set of query nodes.
  Let E denote a set of evidence nodes.
  How to estimate $P(Q\!=\!q|E\!=\!e)$?

- **Solution**

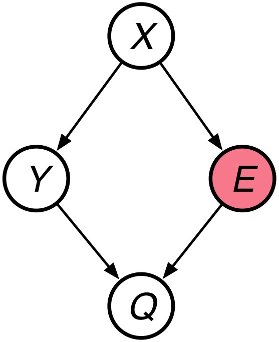
  Generate N samples from the BN's joint distribution.
  Discard (or **reject**) the samples where $E \neq e$.
  Count the samples $N(q,e)$ where $Q\!=\!q$ and $E\!=\!e$.
  Count the samples $N(e)$ where $E\!=\!e$.
  Take the ratio of these counts:

$$\boxed{P(Q\!=\!q|E\!=\!e) \;\approx\; \frac{N(q,e)}{N(e)}} \quad \textbf{where} \quad \boxed{N(q,e) \;\leq\; N(e) \leq N}$$

Example for rejection sampling

- **Sample N times:**

$$
\begin{aligned}
x_i &\sim P(X)\\
y_i &\sim P(Y|X=x_i)\\
e_i &\sim P(E|X=x_i)\\
q_i &\sim P(Q|Y=y_i, E=e_i)
\end{aligned}
$$



- **Define the indicator function:**

$$
I(z,z') = \begin{cases} 1 & \text{if } z=z' \\ 0 & \text{if } z\neq z' \end{cases}
$$

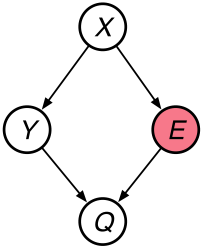
- **Estimate from ratio:**

$$
P(Q=q|E=e) \approx \frac{N(q,e)}{N(e)} = \frac{\sum_{i=1}^{N} I(q,q_i)I(e,e_i)}{\sum_{i=1}^{N} I(e,e_i)}
$$

Properties of rejection sampling

- **It converges in the limit:**

$$\lim_{N\to\infty} \frac{N(q,e)}{N(e)} = P(Q=q|E=e)$$



- **But it is extremely inefficient:**

  It discards all samples without $E = e$.
  It converges **very slowly** for rare evidence.

How can we do better?

Questions?

2. Likelihood weighting

- **Key idea**

  Instantiate evidence nodes instead of sampling them.
  Weight each sample using CPTs at evidence nodes.

- **Intuition**

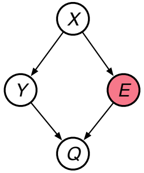
  "Cheat" by fixing the evidence nodes to their desired values.
  "Correct" for cheating by penalizing especially unlikely values.

- **Benefits**

  No discarding of uninformative samples.
  No wasted computation.
  Faster convergence.

Example for likelihood weighting

How to estimate $P(Q=q|E=e)$?



- **Sample N times:**

$$\begin{aligned} x_i &\sim P(X) \\ y_i &\sim P(Y|X=x_i) \\ q_i &\sim P(Q|Y=y_i, E=e) \end{aligned}$$

$\boxed{\textbf{Note: } E \textbf{ is fixed to } e.}$

- **Estimate from ratio:**

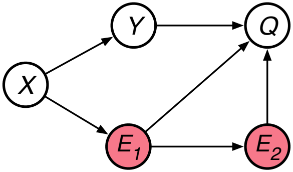
$$P(Q=q|E=e) \approx \frac{\sum_{i=1}^N I(q,q_i)\,\overbrace{P(E=e|X=x_i)}^{\text{likelihood weight}}}{\sum_{i=1}^N P(E=e|X=x_i)}$$

- **Compare to rejection sampling:**

$$P(Q=q|E=e) \approx \frac{\sum_{i=1}^N I(q,q_i) I(e,e_i)}{\sum_{i=1}^N I(e,e_i)}$$

Example with multiple evidence nodes

How to estimate $P(Q=q|E_1=e, E_2=e')$?



- **Sample N times:**

$$\begin{aligned} x_i &\sim P(X) \\ y_i &\sim P(Y|X=x_i) \\ q_i &\sim P(Q|Y=y_i, E_1=e, E_2=e') \end{aligned}$$

$$\boxed{\textbf{Note: } (E_1, E_2) \textbf{ fixed to } (e, e')}$$

- **Estimate from ratio:**

$$P(Q=q|E_1=e, E_2=e') \approx \frac{\sum_{i=1}^{N} I(q, q_i) \overbrace{P(E_1=e|X=x_i)\, P(E_2=e'|E_1=e)}^{\text{product of likelihood weights}}}{\sum_{i=1}^{N} P(E_1=e|X=x_i)\, P(E_2=e'|E_1=e)}$$

- **Compare to rejection sampling:**

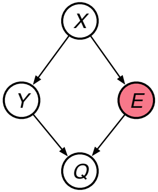
$$P(Q=q|E=e) \approx \frac{\sum_{i=1}^{N} I(q, q_i)\, I(e, e_{1i})\, I(e', e_{2i})}{\sum_{i=1}^{N} I(e, e_{1i})\, I(e', e_{2i})}$$

Questions?

Properties of likelihood weighting

- **It converges in the limit:**

$$\lim_{N\to\infty} \frac{\sum_{i=1}^{N} I(q, q_i)\, P(E=e|X=x_i)}{\sum_{i=1}^{N} P(E=e|X=x_i)} = P(Q=q|E=e)$$



- **It's more efficient than rejection sampling:**

  No samples need to discarded.
  Descendants of evidence nodes are conditioned on evidence.

- **But it can still be very slow:**

$$\begin{tikzcd} Q \arrow[r] & X \arrow[r] & E \end{tikzcd}$$

The worst case for likelihood weighting is when rare evidence is descended from query nodes.

What next?

What do rejection sampling and likelihood weighting have in common?

They both sample via purely forward passes through the BN.
This makes them very simple to implement (HW 3).

But it also makes them slow in certain cases.
To handle those cases, we need to sample nodes differently.

Next lecture: Markov chain Monte Carlo simulation ...