Note Title 3/30/2009

Marginalization in Lazy Propagation

Alg. 3.1.1 [Madsen's dissertation]

Let $\Phi = \{\varphi_1, \ldots \varphi_n\}$ be a set of potentials.

If marginalization of X is invoked on Φ, then

1. set $\Phi_X = \{\varphi \in \Phi \mid X \in dom(\varphi)\}$

2. $\varphi_X^* = \sum_X \prod_{\varphi \in \Phi_X} \varphi$

3. $\Phi^* = \{\varphi_X^*\} \cup \Phi \setminus \Phi_X$

Φ^* is the set of potentials that results from eliminating X from Φ. $\square$

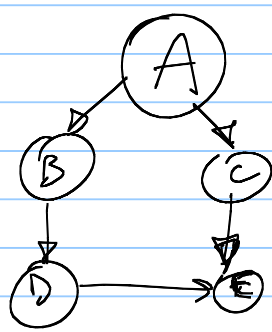
This is Defn. 4.1 on p. 116 with different terminology.

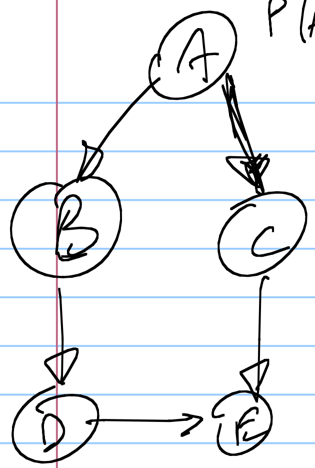
elimination for marginalization

Φ^{-X} for X^{*}.

4.8 Stochastic Simulation in Bayesian Networks

$$P(E=y) \approx \frac{N(E=y)}{N} = \frac{\text{number of cases in which } E=y}{\text{total number of cases}}$$





$P(A) = (.4, .6)$

B \ A	y	n
y	.3	.8
n	.7	.2

C \ A	y	n
y	.7	.4
n	.3	.6

D \ B	y	n
y	.5	.1
n	.5	.9

D \ C	y	n
y	(.9, .1)	(.999, .001)
n	(.999, .001)	(.999, .001)

$P(E \mid C, D)$

AB \ CDE	yyy	yyn	yny	ynn	nyy	nyn	nny	nnn
yy	4	0	5	0	1	0	2	0
yn	2	0	16	0	1	0	8	0
ny	9	1	10	0	14	0	16	0
nn	0	0	4	0	0	0	7	0

$$P(B=y) \approx \frac{N(B=y)}{N} = \frac{62}{100} = .62$$

$$P(E) \approx \left(\frac{N(E=y)}{N}, \frac{N(E=n)}{N}\right) = \left(\frac{99}{100}, \frac{1}{100}\right) = .99, .01$$

The algorithm (p. 448 [J07]):

1. Let $\langle X_1, \ldots, X_n \rangle$ be a topological ordering of the variables (e.g. $\langle A, B, C, D, E \rangle$)
2. For $j = 1$ to N:
   a) For $i = 1$ to n:
      - sample a state x_i of X_i using $P(X_i \mid pa(X_i) = \pi)$, where π is the configuration already sampled for $pa(X_i)$.

   b) If $\underline{x} = (x_1, \ldots, x_n)$ is consistent with $\underline{e}$, then

$N(X_k = x_k) := N(X_k = x_k) + 1$, where

x_k is the state that was sampled for X_k

[else: discard x – because it is inconsistent with the evidence]

3. Return

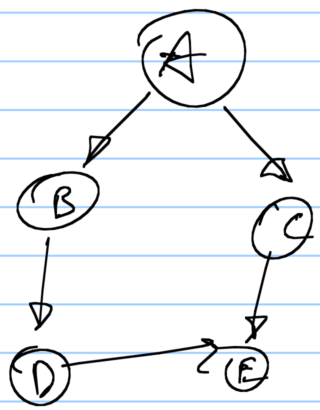
$$P(X_k = x_k \mid \underline{e}) \approx \frac{N(X_k = x_k)}{\sum_{x \in sp(X_k)} N(X_k = x_k)}$$

$$P(X_k = x_k \mid \underline{e}) = \frac{P(X_k = x_k, \underline{e})}{P(\underline{e})}$$

The alg. described requires generating a lot of irrelevant samples when the evidence has low probability.

4.8.2 Likelihood weighting was designed to overcome this problem.

Idea: to avoid flipping a biased coin for each variable on which there is evidence. Wrong results!



$$P(A \mid B=n, E=w)$$

By following the "idea," you estimate $P(A)$, not $P(A \mid \underline{e})$, b/c when the state of A is determined, only $P(A)$ is used

The idea can be salvaged by carefully writing down

$$P(U, \underline{e}) = \prod_{X \in U \setminus \mathcal{E}} P(X \mid pa(X)', pa(X)'' = e) \times \qquad (4.4)$$

$$\prod_{X \in \mathcal{E}} P(X = e \mid pa(X)', pa(X)'' = e)$$

$\mathcal{E}$ is the set of variables that have received evidence

The mistaken (naive) algorithm (based on the "idea") ignores the second part!

So the fix is to weigh each sample by

$$\prod_{X \in \mathcal{E}} P(X = e \mid pa(X)', pa(X)'' = e)$$

$$w(\underline{x}, \underline{e}) = \prod_{E \in \mathcal{E}} P(E = e \mid pa(E) = \pi)$$

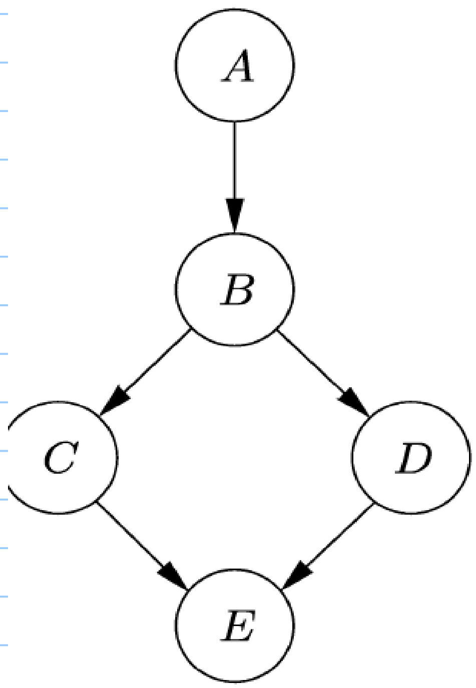
configuration in the sample point

In practice, once a sample point (case) has been produced, $w(\underline{x}, \underline{e})$ is calculated and it is added to $N(\,?\,)$ instead of 1.

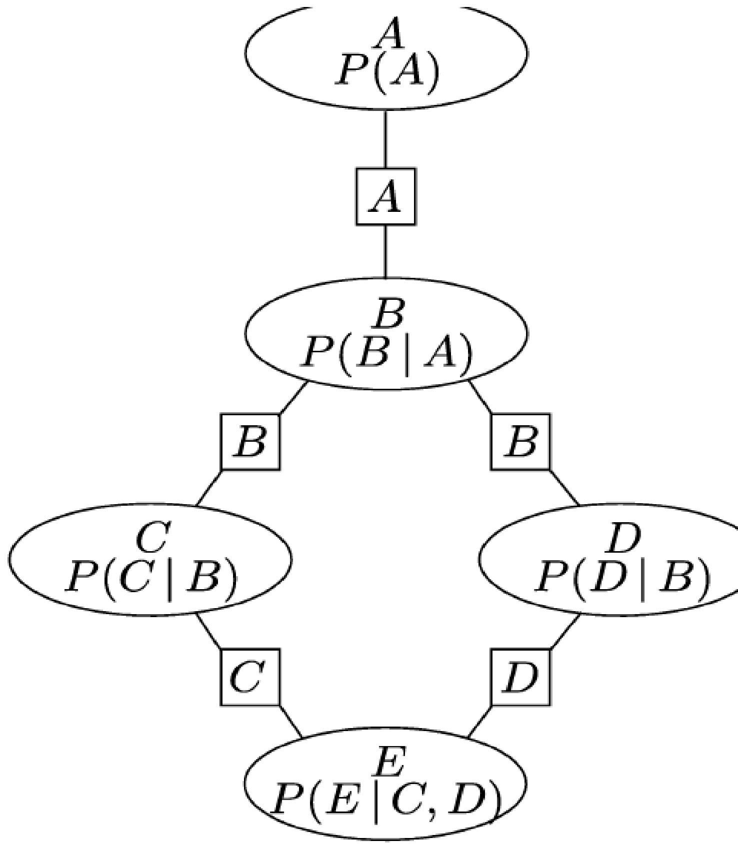
4.8.3 Gibbs Sampling (used Nielsen slides, unpublished)

4.9 Loopy Belief Propagation [~ 1998 ?]

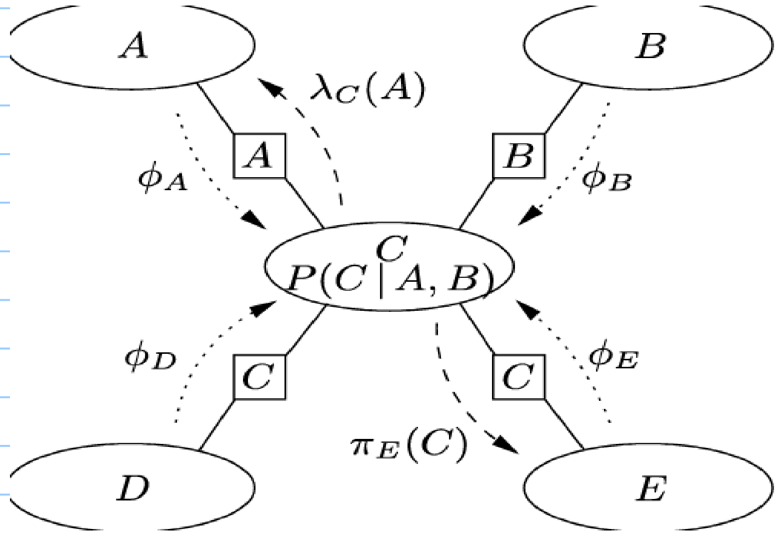
Used in "turbo coding" (McEliece)



(a)



(b)

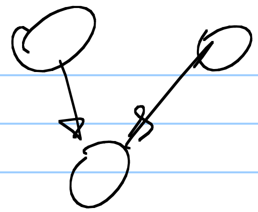
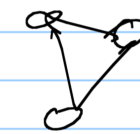


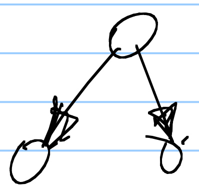
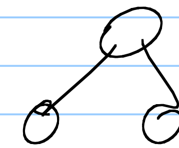
$$\lambda_C(A) = \textstyle\sum_{B,C} P(C \mid A, B)\phi_B\phi_D\phi$$

$$\pi_E(C) = \phi_D \textstyle\sum_{A,B} P(C \mid A, B)\phi_A\phi$$

In the special case of BNs that are trees, this algorithm is the same as the junction tree algorithm.

This alg, in the case of trees is due to Pearl (& Kim) [1982]
The alg also works for polytrees — BNs whose moral graph is a singly connected graph.

 ~~not~~ a polytree, b/c its moral graph is 

 a polytree, b/c its moral graph is 

By the observation that each clique in a j.tree for a polytree is small, you may show that belief update in polytrees can be done in linear time.