Bayesian networks: approximate inference
Machine Intelligence

Thomas D. Nielsen

September 2008
Motivation

Because of the (worst-case) intractability of exact inference in Bayesian networks, try to find more efficient approximate inference techniques:

Instead of computing exact posterior

\[ P(A \mid E = e) \]

compute approximation

\[ \hat{P}(A \mid E = e) \]

with

\[ \hat{P}(A \mid E = e) \sim P(A \mid E = e) \]
Approximate Inference

Absolute/Relative Error

For $p, \hat{p} \in [0, 1]$: $\hat{p}$ is approximation for $p$ with *absolute error* $\leq \epsilon$, if

$$| p - \hat{p} | \leq \epsilon,$$

i.e. $\hat{p} \in [p - \epsilon, p + \epsilon]$. 
Approximate Inference

Absolute/Relative Error

For $p, \hat{p} \in [0, 1]$: $\hat{p}$ is approximation for $p$ with absolute error $\leq \epsilon$, if

$$| p - \hat{p} | \leq \epsilon, \text{ i.e. } \hat{p} \in [p - \epsilon, p + \epsilon].$$

$\hat{p}$ is approximation for $p$ with relative error $\leq \epsilon$, if

$$| 1 - \hat{p}/p | \leq \epsilon, \text{ i.e. } \hat{p} \in [p(1 - \epsilon), p(1 + \epsilon)].$$
Absolute/Relative Error

For $p, \hat{p} \in [0, 1]$: $\hat{p}$ is approximation for $p$ with absolute error $\leq \epsilon$, if

$$|p - \hat{p}| \leq \epsilon, \text{ i.e. } \hat{p} \in [p - \epsilon, p + \epsilon].$$

$\hat{p}$ is approximation for $p$ with relative error $\leq \epsilon$, if

$$|1 - \hat{p}/p| \leq \epsilon, \text{ i.e. } \hat{p} \in [p(1 - \epsilon), p(1 + \epsilon)].$$

This definition is not always fully satisfactory, because it is not symmetric in $p$ and $\hat{p}$ and not invariant under the transition $p \rightarrow (1 - p), \hat{p} \rightarrow (1 - \hat{p})$. Use with care!

When $\hat{p}_1, \hat{p}_2$ are approximations for $p_1, p_2$ with absolute error $\leq \epsilon$, then no error bounds follow for $\hat{p}_1/\hat{p}_2$ as an approximation for $p_1/p_2$.

When $\hat{p}_1, \hat{p}_2$ are approximations for $p_1, p_2$ with relative error $\leq \epsilon$, then $\hat{p}_1/\hat{p}_2$ approximates $p_1/p_2$ with relative error $\leq (2\epsilon)/(1 + \epsilon)$. 
Approximate Inference

Randomized Methods

Most methods for approximate inference are randomized algorithms that compute approximations \( \hat{P} \) from random samples of instantiations.

We shall consider:

- Forward sampling
- Likelihood weighting
- Gibbs sampling
- Metropolis Hastings algorithm
Forward Sampling

Observation: can use Bayesian network as random generator that produces full instantiations \( V = v \) according to distribution \( P(V) \).

Example:

- Generate random numbers \( r_1, r_2 \) uniformly from \([0,1]\).
- Set \( A = t \) if \( r_1 \leq .2 \) and \( A = f \) else.
- Depending on the value of \( A \) and \( r_2 \) set \( B \) to \( t \) or \( f \).

Generation of one random instantiation: linear in size of network.
Approximate Inference

Sampling Algorithm

Thus, we have a randomized algorithm $S$ that produces possible outputs from $sp(V)$ according to the distribution $P(V)$.

Define

$$\hat{P}(A = a | E = e) := \frac{|\{i \in 1, \ldots, N | E = e, A = a \text{ in } S_i\}|}{|\{i \in 1, \ldots, N | E = e \text{ in } S_i\}|}$$
Approximate Inference

Forward Sampling: Illustration

Sample with
not $E = e$
$E = e, A \neq a$
$E = e, A = a$

Approximation for $P(A = a \mid E = e)$:

\[
\frac{#}{#} \frac{\#}{\#} \cup \bullet \bullet
\]
Approximate Inference

Sampling from the conditional distribution

Problem of forward sampling: samples with $E \neq e$ are useless!

Idea: find sampling algorithm $S_c$ that produces outputs from $s_p(V)$ according to the distribution $P(V \mid E = e)$. 
Approximate Inference

Sampling from the conditional distribution

Problem of forward sampling: samples with $E \neq e$ are useless!

Idea: find sampling algorithm $S_c$ that produces outputs from $s_p(V)$ according to the distribution $P(V \mid E = e)$.

**A tempting approach:** Fix the variables in $E$ to $e$ and sample from the nonevidence variables only!
Approximate Inference

Sampling from the conditional distribution

Problem of forward sampling: samples with $E \neq e$ are useless!

Idea: find sampling algorithm $S_c$ that produces outputs from $sp(V)$ according to the distribution $P(V \mid E = e)$.

A tempting approach: Fix the variables in $E$ to $e$ and sample from the nonevidence variables only!

Problem: Only evidence from the ancestors are taken into account!
Likelihood weighting

We would like to sample from \((\text{pa}(X))''\) are the parents in \(E\)

\[
P(U, e) = \prod_{X \in U \setminus E} P(X \mid \text{pa}(X)', \text{pa}(X)'' = e) \times \prod_{X \in E} P(X = e \mid \text{pa}(X)', \text{pa}(X)'' = e),
\]

but by applying forward sampling with fixed \(E\) we actually sample from:

\[
\text{Sampling distribution} = \prod_{X \in U \setminus E} P(X \mid \text{pa}(X)', \text{pa}(X)'' = e).
\]
Approximate Inference

Likelihood weighting

We would like to sample from \((\text{pa}(X))'' \) are the parents in \(E\)

\[
P(\mathcal{U}, e) = \prod_{X \in \mathcal{U} \setminus E} P(X | \text{pa}(X)', \text{pa}(X)'' = e) \times \prod_{X \in E} P(X = e | \text{pa}(X)', \text{pa}(X)'' = e),
\]

but by applying forward sampling with fixed \(E\) we actually sample from:

Sampling distribution = \[
\prod_{X \in \mathcal{U} \setminus E} P(X | \text{pa}(X)', \text{pa}(X)'' = e).
\]

Solution: Instead of letting each sample count as 1, use

\[
w(x, e) = \prod_{X \in E} P(X = e | \text{pa}(X)', \text{pa}(X)'' = e).
\]
Likelihood weighting: example

- Assume evidence $B = t$.
- Generate a random number $r$ uniformly from $[0,1]$.
- Set $A = t$ if $r \leq 0.2$ and $A = f$ else.
- If $A = t$ then let the sample count as $w(t, t) = 0.7$; otherwise $w(f, t) = 0.4$. 
Approximate Inference

Likelihood weighting: example

- Assume evidence $B = t$.
- Generate a random number $r$ uniformly from $[0,1]$.
- Set $A = t$ if $r \leq 0.2$ and $A = f$ else.
- If $A = t$ then let the sample count as $w(t, t) = 0.7$; otherwise $w(f, t) = 0.4$.

With $N$ samples $(a_1, \ldots, a_N)$ we get

$$\hat{P}(A = t \mid B = t) = \frac{\sum_{i=1}^{N} w(a_i = t, e)}{\sum_{i=1}^{N} (w(a_i = t, e) + w(a_i = f, e))}.$$
Gibbs Sampling

For notational convenience assume from now on that for some \( l \): \( E = V_{l+1}, V_{l+2}, \ldots, V_n \). Write \( W \) for \( V_1, \ldots, V_l \).

Principle: obtain new sample from previous sample by randomly changing the value of only one selected variable.

```
Procedure Gibbs sampling
\( v_0 = (v_{0,1}, \ldots, v_{0,l}) := \text{arbitrary instantiation of } W \)
\( i := 1 \)
repeat forever
   choose \( V_k \in W \) # deterministic or randomized
   generate randomly \( v_{i,k} \) according to distribution
   \[
P(V_k | V_1 = v_{i-1,1}, \ldots, V_{k-1} = v_{i-1,k-1},
   V_{k+1} = v_{i-1,k+1}, \ldots, V_l = v_{i-1,l}, E = e)\]
   set \( v_i = (v_{i-1,1}, \ldots, v_{i-1,k-1}, v_{i,k}, v_{i-1,k+1}, \ldots, v_{i-1,l}) \)
   \( i := i + 1 \)
```

Approximate Inference
The process of Gibbs sampling can be understood as a \textit{random walk} in the space of all instantiations with $E = e$:

Reachable in one step: instantiations that differ from current one by value assignment to at most one variable (assume randomized choice of variable $V_k$).
Implementation of Sampling Step

The sampling step

generate randomly $v_{i,k}$ according to distribution

$$P(V_k \mid V_1 = v_{i-1,1}, \ldots, V_{k-1} = v_{i-1,k-1},$$
$$V_{k+1} = v_{i-1,k+1}, \ldots, V_l = v_{i-1,l}, E = e)$$

requires sampling from a conditional distribution. In this special case (all but one variables are instantiated) this is easy: just need to compute for each $v \in \text{sp}(V_k)$ the probability

$$P(V_1 = v_{i-1,1}, \ldots, V_{k-1} = v_{i-1,k-1}, V_k = v, V_{k+1} = v_{i-1,k+1}, \ldots, V_l = v_{i-1,l}, E = e)$$

(linear in network size), and choose $v_{i,k}$ according to these probabilities (normalized).

This can be further simplified by computing the distribution on $\text{sp}(V_k)$ only in the Markov blanket of $V_k$, i.e. the subnetwork consisting of $V_k$, its parents, its children, and the parents of its children.
Convergence of Gibbs Sampling

Under certain conditions: the distribution of samples converges to the posterior distribution $P(W \mid E = e)$:

$$\lim_{i \to \infty} P(v_i = v) = P(W = v \mid E = e) \quad (v \in sp(W)).$$

Sufficient conditions are:

- in the repeat loop of the Gibbs sampler, variable $V_k$ is randomly selected (with non-zero selection probability for all $V_k \in W$), and
- the Bayesian network has no zero-entries in its cpt’s
Approximate Inference using Gibbs Sampling

1. Start Gibbs sampling with some starting configuration $\mathbf{v}_0$.
2. Run the sampler for $N$ steps (“Burn in”)
3. Run the sampler for $M$ additional steps; use the relative frequency of state $\mathbf{v}$ in these $M$ samples as an estimate for $P(\mathbf{W} = \mathbf{v} | \mathbf{E} = \mathbf{e})$.

Problems:
- How large must $N$ be chosen? Difficult to say how long it takes for Gibbs sampler to converge!
- Even when sampling is from the stationary distribution, samples are not independent. Result: error cannot be bounded as function of $M$ using Chebyshev’s inequality (or related methods).
Effect of dependence

\[ P(v_N = v) \] close to \( P(W = v \mid E = e) \): probability that \( v_N \) is in the red region is close to \( P(A = a \mid E = e) \).

This does not guarantee that the fraction of samples in \( v_N, v_{N+1}, \ldots, v_{N+M} \) that are in the red region yields a good approximation to \( P(A = a \mid E = e) \)!
Multiple starting points

In practice, one tries to counteract these difficulties by restarting the Gibbs sampling several times (often with different starting points):
Approximate Inference

Metropolis Hastings Algorithm

Another way of constructing a random walk on $sp(W)$:

Let

$$\{ q(v, v') \mid v, v' \in sp(W) \}$$

be a set of transition probabilities over $sp(W)$, i.e. $q(v, \cdot)$ is a probability distribution for each $v \in sp(W)$. The $q(v, v')$ are called proposal probabilities.

Define

$$\alpha(v, v') := \min \left\{ 1, \frac{P(W = v' \mid E = e)q(v', v)}{P(W = v \mid E = e)q(v, v')} \right\}$$

$$:= \min \left\{ 1, \frac{P(W = v', E = e)q(v', v)}{P(W = v, E = e)q(v, v')} \right\}$$

$\alpha(v, v')$ is called the acceptance probability for the transition from $v$ to $v'$. 
Procedure Metropolis Hastings sampling

\[ v_0 = (v_{0,1}, \ldots, v_{0,l}) := \text{arbitrary instantiation of } W \]

\[ i := 1 \]

repeat forever

sample \( v' \) according to distribution \( q(v_{i-1}, \cdot) \)

set \textit{accept} to \textit{true} with probability \( \alpha(v, v') \)

if \textit{accept}

\[ v_i := v' \]

else

\[ v_i := v_{i-1} \]

\[ i := i + 1 \]
Approximate Inference

Convergence of Metropolis Hastings Sampling

Under certain conditions: the distribution of samples converges to the posterior distribution $P(W \mid E = e)$.

A sufficient condition is:
- $q(v, v') > 0$ for all $v, v'$.

To obtain a good performance, $q$ should be chosen so as to obtain high acceptance probabilities, i.e. quotients

$$\frac{P(W = v' \mid E = e)q(v', v)}{P(W = v \mid E = e)q(v, v')}$$

should be close to 1. Optimal (but usually not feasible): $q(v, v') = P(W = v' \mid E = e)$. Generally: try to approximate with $q$ the target distribution $P(W \mid E = e)$. 
Loopy belief propagation

- Message passing algorithm like junction tree propagation
- Works directly on the Bayesian network structure (rather than the junction tree)
Loopy belief propagation

Message passing

A node sends a message to a neighbor by

- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.
Loopy belief propagation

Message passing

A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.
Loopy belief propagation

Message passing

A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.

$$P(C|A, B)$$

Diagram:

- Nodes: A, B, C, D, E
- Messages: $\phi_A$, $\phi_B$
Loopy belief propagation

Message passing

A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.
Loopy belief propagation

Message passing

A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.
A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.

\[ \pi_E(C) = \phi_D \sum_{A,B} P(C|A,B)\phi_A\phi_B \]
Loopy belief propagation

Message passing

A node sends a message to a neighbor by
- multiplying the incoming messages from all other neighbors to the potential it holds.
- marginalizing the result down to the separator.

\[
\pi_E(C) = \phi_D \sum_{A,B} P(C \mid A, B) \phi_A \phi_B \\
\lambda_C(A) = \sum_{B,C} P(C \mid A, B) \phi_B \phi_D \phi_E
\]
A few observations:

- When calculating $P(E)$ we treat $C$ and $D$ as being independent (in the junction tree $C$ and $D$ would appear in the same separator).
- Evidence on a converging connection may cause the error to cycle.
Loopy belief propagation

In general

- There is no guarantee of convergence, nor that in case of convergence, it will converge to the correct distribution. However, the method converges to the correct distribution surprisingly often!
- If the network is singly connected, convergence is guaranteed.
Literature
