Probabilistic Programming

Lecture #18: Bayesian Networks

Joost-Pieter Katoen

RWTH Lecture Series on Probabilistic Programming 2018

Overview

1. Motivation
2. Inference by program verification
3. How long to sample a Bayesian network?

Overview

1. Motivation
2. Inference by program verification
3. How long to sample a Bayesian network?

Inference

Inference (of conditional probabilities)

Let B be a BN with set V of vertices and the evidence $\mathbf{E} \subseteq V$ and the questions $\mathbf{Q} \subseteq V$.

The probabilistic inference problem is to determine the conditional probability:

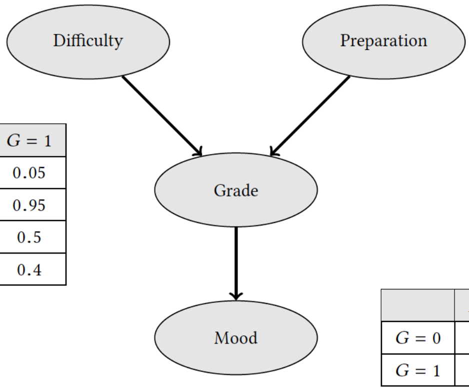
$$Pr(\mathbf{Q} = \mathbf{q} \mid \mathbf{E} = \mathbf{e}) \ = \ \frac{Pr(\mathbf{Q} = \mathbf{q} \wedge \mathbf{E} = \mathbf{e})}{Pr(\mathbf{E} = \mathbf{e})}.$$

Inference is the main focus when reasoning about Bayesian networks.

Inference example: student exam's mood

$D=0$	$D=1$
0.6	0.4

$P=0$	$P=1$
0.7	0.3

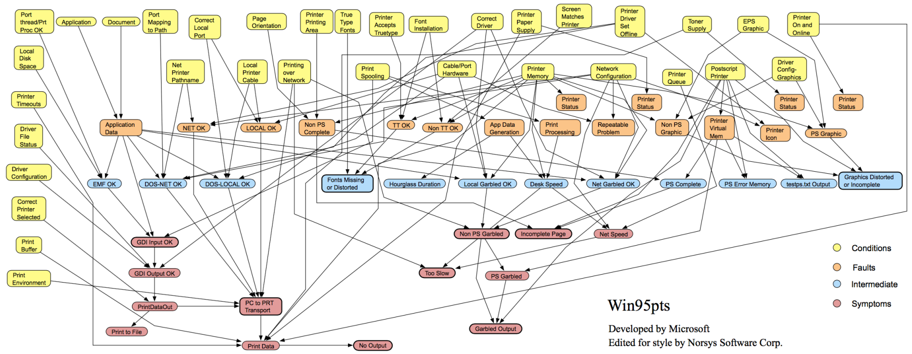


	$G=0$	$G=1$
$D=0, P=0$	0.95	0.05
$D=1, P=1$	0.05	0.95
$D=0, P=1$	0.5	0.5
$D=1, P=0$	0.6	0.4

	$M=0$	$M=1$
$G=0$	0.9	0.1
$G=1$	0.3	0.7

How likely does a student end up with a bad mood after getting a bad grade for an easy exam, **given that** she is well prepared?

Inference example: Printer troubleshooting in Windows 95



How likely is it that your print is garbled **given that** the ps-file is not and the page orientation is portrait?

Overview

Bayesian inference

- Exact inference of Bayesian networks is NP-hard
  - Variable Elimination
  - Join-tree Algorithm
  - ......
- Approximate inference of BNs is NP-hard too
- Typical fallback: use simulation
  - Rejection Sampling
  - Markov Chain Monte Carlo (MCMC)
  - Metropolis-Hastings
  - Gibbs Sampling
  - Importance Sampling
  - ......

Most sampling-based methods exploit rejection sampling.

Rejection sampling

For a given Bayesian network and some evidence:

1. Sample from the joint distribution described by the BN
2. If the sample complies with the evidence, accept the sample and halt
3. If not, repeat sampling (that is: go back to step 1.)

If this procedure is applied N times, N iid-samples result.

Potential problem: What happens if the evidence has low probability? E.g., zero.

Removal of conditioning = rejection sampling

Recall conditioning removal:

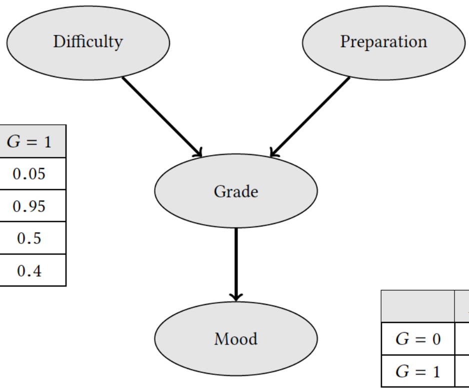
```
x := 0 [p] x := 1;
y := 0 [p] y := 1;
observe(x != y)
```

```
sx, sy := x, y; flag := true;
while(flag) {
  x, y := sx, sy; flag := false;
  x := 0 [p] x := 1;
  y := 0 [p] y := 1;
  flag := (x = y)
}
```

This program transformation replaces observe-statements by loops.
The resulting loopy programs represent rejection sampling.

Student exam's mood example

$D = 0$	$D = 1$
0.6	0.4



$P = 0$	$P = 1$
0.7	0.3

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

	$M = 0$	$M = 1$
$G = 0$	0.9	0.1
$G = 1$	0.3	0.7

How likely does a student end up with a bad mood after getting a bad grade for an easy exam, **given that** she is well prepared?

Bayesian networks as programs

- Take a topological sort of the BN's vertices, e.g., $D; P; G; M$
- Map each conditional probability table (aka: node) to a program, e.g.:

```
if (xD = 0 && xP = 0) {
   xG := 0 [0.95] xG := 1
   } else if (xD = 1 && xP = 1) {
   xG := 0 [0.05] xG := 1
   } else if (xD = 0 && xP = 1) {
   xG := 0 [0.5] xG := 1
   } else if (xD = 1 && xP = 0) {
   xG := 0 [0.6] xG := 1
}
```

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

- Condition on the evidence, e.g., for $P = 1$ ("well-prepared"):

```
repeat { progD ; progP; progG ; progM } until (P=1)
```

Properties of BN programs

```
repeat { progD ; progP; progG ; progM } until (P=1)
```

1. Every BN-program naturally represents rejection sampling
2. The loop in a BN-program is simple
   - No "data-flow" between successive loop iterations
   - Loop invariants are not needed (as we will see)
3. BN-programs may be not positively a.s.-terminating

   Such BNs are ill-conditioned' their evidence has probability zero
4. A BN-program's size is linear in the BN's size

Soundness

For BN B with evidence $E \subseteq V$ and value $\underline{v}$ for vertex v:

$$\underbrace{wp(\mathtt{prog}(B, \mathbf{e}), \bigwedge_{v \in V \setminus E} x_v = \underline{v})}_{\text{wp of the BN program of } B} \quad = \quad \underbrace{Pr\left(\bigwedge_{v \in V \setminus E} v = \underline{v} \;\middle|\; \bigwedge_{e \in E} e = \underline{e}\right)}_{\text{joint distribution of BN } B}$$

where $\mathtt{prog}(B, \mathbf{e})$ equals $\mathtt{repeat}\ progB\ \mathtt{until}\ (\bigwedge_{e \in E} x_e = \underline{e})$.

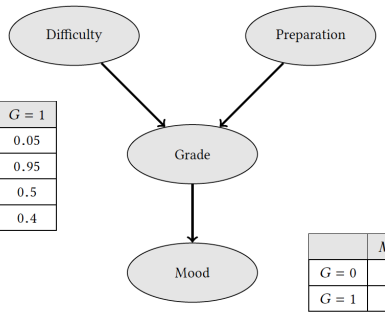
Thus: inference of BNs can be done using wp-reasoning

Inference by wp-reasoning

$D = 0$	$D = 1$
0.6	0.4

$P = 0$	$P = 1$
0.7	0.3

	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4



	$M = 0$	$M = 1$
$G = 0$	0.9	0.1
$G = 1$	0.3	0.7

Ergo: exact Bayesian inference can be done by wp-reasoning, e.g.,

$$wp(P_{mood}, [x_D = 0 \wedge x_G = 0 \wedge x_M = 0]) \quad = \quad \frac{Pr(D = 0, G = 0, M = 0, P = 1)}{Pr(P = 1)} \quad = \quad 0.27$$

Reasoning about loops

Reasoning about loops is hard.
Typically, loop invariants are used to capture the effect of loops.
Finding such loop invariants in general is undecidable.

Bayesian networks correspond to "simple" probabilistic programs.
Loops in such programs are "data-flow" free.
Their effect can be given as closed-form solution.
This can be done algorithmically.

I.i.d-loops

Loop `while(G)P` is iid wrt. expectation f whenever:

both $wp(P, [G])$ and $wp(P, [\neg G] \cdot f)$ are unaffected by P.

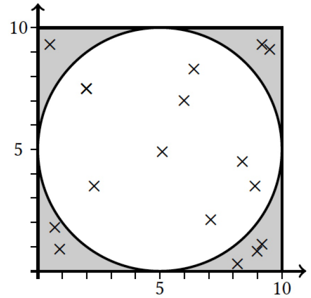
f is *unaffected* by P if none of f's variables are modified by P:

x is a variable of f iff $\exists s. \exists v, u: \quad f(s[x = v]) \neq f(s[x = u])$

If g is unaffected by program P, then: $wp(P, g \cdot f) \;=\; g \cdot wp(P, f)$

Example: sampling within a circle

```
while ((x-5)**2 + (y-5)**2 >= 25){
    x := uniform(0..10);
    y := uniform(0..10)
}
```



This program is iid for every f, as both are unaffected by P's body:

$$wp(P, [G]) \;=\; \frac{48}{121} \quad \text{and}$$

$$wp(P, [\neg G]{\cdot}f) \;=\; \frac{1}{121} \sum_{i=0}^{10p} \sum_{j=0}^{10p} [(i/p{-}5)^2 + (j/p{-}5)^2 < 25] \cdot f\big(x/(i/p), y/(j/p)\big)$$

Weakest precondition of iid-loops

If `while(G)P` is iid for expectation f, it holds for every state s:

$$wp(\texttt{while(G)P}, f)(s) \;=\; [G](s) \cdot \frac{wp(P, [\neg G]{\cdot}f)(s)}{1 - wp(P, [G])(s)} + [\neg G](s) \cdot f(s)$$

where we let $\frac{0}{0} = 0$.

Proof: use $wp(while_n(G)P, f) = [G] \cdot wp(P, [\neg G]{\cdot}f) \cdot \sum_{i=0}^{n-2} \left(wp(P, [G])^i\right) + [\neg G] \cdot f$

No loop invariant or martingale needed. Fully automatable.

Overview

1. Motivation
2. Inference by program verification
3. How long to sample a Bayesian network?

How long to sample a BN?

[Gordon, Nori, Henzinger, Rajamani, 2014]

"the main challenge in this setting [sampling-based approaches] is that many samples that are generated during execution are ultimately rejected for not satisfying the observations."

Recall: Rejection sampling

For a given Bayesian network and some evidence:

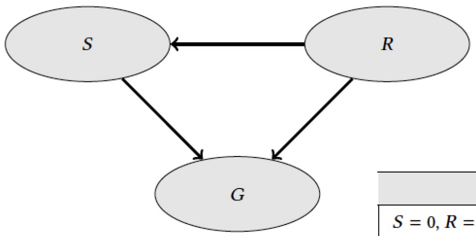
1. Sample from the joint distribution described by the BN
2. If the sample complies with the evidence, accept the sample and halt
3. If not, repeat sampling (that is: go back to step 1.)

If this procedure is applied N times, N iid-samples result.

Q: How many samples do we need on average for a **single** iid-sample?

Sampling time of a toy Bayesian network

	$S=0$	$S=1$
$R=0$	a	$1-a$
$R=1$	0.2	0.8



$R=0$	$R=1$
a	$1-a$

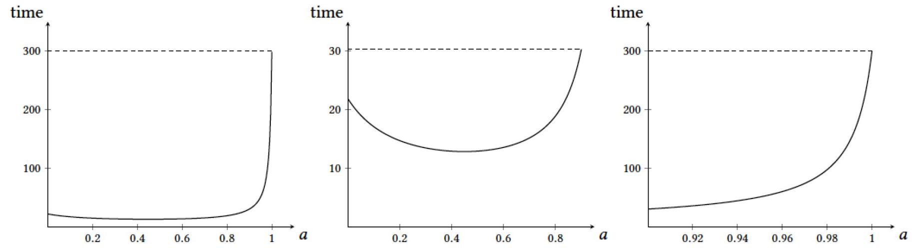
	$G=0$	$G=1$
$S=0, R=0$	0.01	0.99
$S=0, R=1$	0.25	0.75
$S=1, R=0$	0.9	0.1
$S=1, R=1$	0.2	0.8

This BN is parametric (in a)

How many samples are needed on average for a **single** iid-sample for evidence $G=0$?

Sampling time for example BN

Rejection sampling for $G=0$ requires $\dfrac{200a^2-40a-460}{89a^2-69a-21}$ samples:



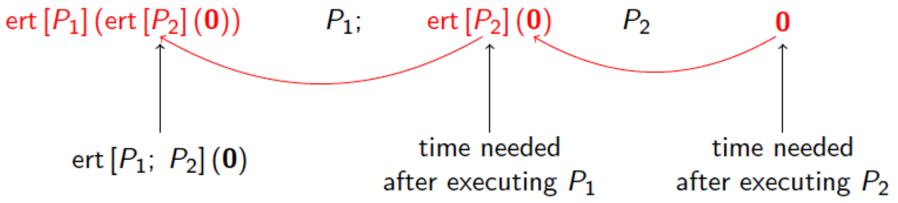
For $a \in [0.1, 0.78]$, < 18 samples; for $a \geq 0.98$, 100 samples are needed

For real-life BNs, one may exceed 10^{15} (or more) samples

Deriving sampling times via expected runtimes

Let $ert() : \text{pGCL} \rightarrow (\mathbb{T} \rightarrow \mathbb{T})$ where:

- $ert(P, t)(s)$ is the expected runtime of P on input state s if t captures the runtime of computation following P.
- $ert(P, \mathbf{0})(s)$ is the expected runtime of P on input state s.



Expected runtime transformer

Syntax	Semantics $ert(P, t)$
`skip`	$\mathbf{1}+t$
`diverge`	∞
`x := E`	$\mathbf{1} + t[x := E]$
`P1 ; P2`	$ert(P_1, ert(P_2, t))$
`if (G) P1 else P2`	$\mathbf{1} + [G] \cdot ert(P_1, t) + [\neg G] \cdot ert(P_2, t)$
`P1 [p] P2`	$\mathbf{1} + p \cdot ert(P_1, t) + (1-p) \cdot ert(P_2, t)$
`while(G)P`	$\text{lfp}\, X.\, \mathbf{1} + ([G] \cdot ert(P, X) + [\neg G] \cdot t)$

lfp is the least fixed point operator wrt. the ordering $\leq$ on runtimes

Decomposition

Decomposition theorem

For every pGCL program P and expectation f:

$$ert(P, f) \;=\; ert(P, \mathbf{0}) + wp(P, f)$$

Expected runtime of iid-loops

For a.s.-terminating iid-loop `while(G)P` for which every iteration runs in the same expected time, we have:

$$ert(\texttt{while(G)P}, t) \;=\; \mathbf{1} + [G] \cdot \frac{\mathbf{1} + ert(P, [\neg G] \cdot t)}{1 - wp(P, [G])} + [\neg G](s) \cdot t$$

where $0/0 := 0$ and $a/0 := \infty$ for $a \neq 0$.

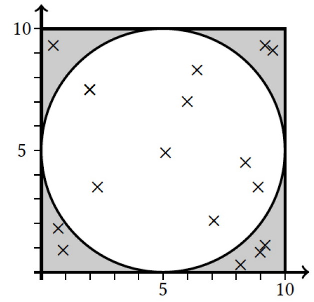
Proof: similar as for the inference (wp) using the decomposition result:

$$ert(P, t) \;=\; ert(P, \mathbf{0}) + wp(P, t)$$

No loop invariant needed. Fully automatable.

Example: sampling within a circle

```
while ((x-5)**2 + (y-5)**2 >= 25){
    x := uniform(0..10);
    y := uniform(0..10)
}
```



This iid-loop is a.s.-terminating, and every iteration has same expected time.

$$\text{Then:} \quad ert(P_{circle}, \mathbf{0}) \;=\; \mathbf{1} + [(x{-}5)^2 + (y{-}5)^2 \geq 25] \cdot \frac{363}{73}$$

So: $1 + {}^{363}/_{73} \approx 5.97$ operations are required on average using rejection sampling

Sample times of BN programs

Every BN-program is iid for every f, is almost surely terminating, and every loop-iteration takes on average equally long.

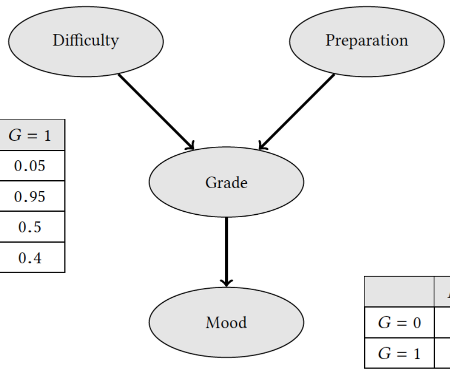
This enables determining the exact expected sampling times of BNs in a fully automated manner.

But: BN-programs may be not positively a.s.-terminating
This holds for ill-conditioned BNs.
The evidence(s) in such BNs are occurring with probability zero.

The student's mood example

$D = 0$	$D = 1$
0.6	0.4

$P = 0$	$P = 1$
0.7	0.3



	$G = 0$	$G = 1$
$D = 0, P = 0$	0.95	0.05
$D = 1, P = 1$	0.05	0.95
$D = 0, P = 1$	0.5	0.5
$D = 1, P = 0$	0.6	0.4

	$M = 0$	$M = 1$
$G = 0$	0.9	0.1
$G = 1$	0.3	0.7

$$ert(\underbrace{\texttt{repeat D; P; G; M until (P=1)}}_{\text{program of student mood's BN}}, \mathbf{0}) \;=\; \frac{\mathbf{1} + ert(D; P; G; M, \mathbf{0})}{wp(D; P; G; M, [P=1])} \;\approx\; \mathbf{23.46}$$

Experimental results

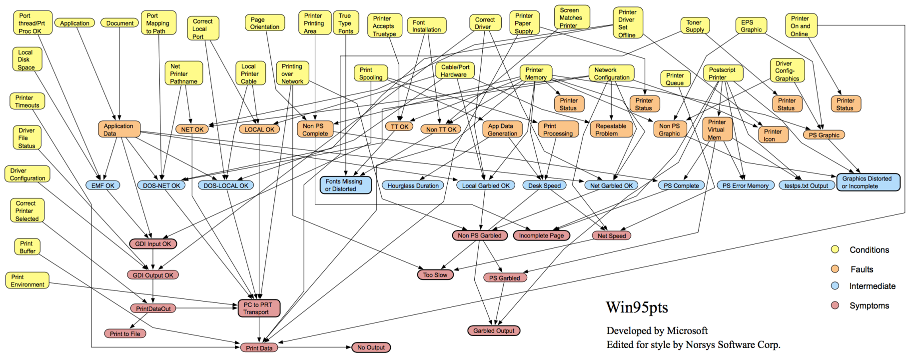
Benchmark BNs from `www.bnlearn.com`

BN	$\lvert V\rvert$	$\lvert E\rvert$	aMB	**|E|**	**EST**	**time** (s)	**|E|**	**EST**	**time** (s)
`hailfinder`	56	66	3.54	5	$5\,10^5$	0.63	9	$9\,10^6$	0.46
`hepar2`	70	123	4.51	1	$1.5\,10^2$	1.84	2	—	MO
`win95pts`	76	112	5.92	3	$4.3\,10^5$	0.36	12	$4\,10^7$	0.42
`pathfinder`	135	200	3.04	3	$2.9\,10^4$	31	7	∞	5.44
`andes`	223	338	5.61	3	$5.2\,10^3$	1.66	7	$9\,10^4$	0.99
`pigs`	441	592	3.92	1	$2.9\,10^3$	0.74	7	$1.5\,10^6$	1.02
`munin`	1041	1397	3.54	5	∞	1.43	10	$1.2\,10^{18}$	65

aMB = *average Markov Blanket size*, a measure of independence in BNs

The last column is the analysis time of ert-analysis of the BN-program.

Printer troubleshooting in Windows 95



Java implementation executes about 10^7 steps in a single second

For $|\mathbf{E}| = 17$, an EST of 10^{15} yields **3.6 years simulation for a single iid-sample**

Take-home messages

- Bayesian networks are directed acyclic graphs of random variables
- Inference of conditional probabilities on BNs is NP-hard
- BNs are probabilistic programs with “data-flow”-free loops
- No loop invariants are needed to reason about BN programs
- Wp-reasoning can do inference and determines sampling times
- . . . . . . in a fully automated manner.

Written exam: February 25, 2019 (10–12:00) and March 27, 2019 (10–12:00)