Fundamentals of AI and KR - Module 3

5. Approximate inference

Paolo Torroni

Fall 2023

Notice

Table of Contents

+ scholarships

About the exam

Exam formats

Choose one:

- Written exam
- Project

Exam formats

Choose one:

- Written exam
  1. [*8 points*] one **exercise** in the style of those seen in class or proposed by Russel & Norvig, Koller & Friedman; structured in several points/questions (see Virtuale for past exam exercises)
  2. [*3 points*] one or more **open questions** about theory, material covered in classes
  - Lasts about 1 hour
  - How to sit exam: **sign up via AlmaEsami** (be mindful of deadlines)
  - 4 dates each year. In 2024:
    - 19 January, 9am-noon
    - 16 February, 9am-noon
    - June or July
    - September
- Project

Exam formats

Choose one:

- Written exam
- Project
  - Usually two types: vertical case study, or domain-independent investigation of one or more aspects of Bayes nets
  - **Case study**: implement an (interesting) example in a domain of your choice using pgmpy or other libraries, explore the model to some extent, for example by studying structural properties, developing queries, making experiments to see what happens if you change parameter values, evidence, etc
  - **Investigation**: choose one or more aspects of Bayes nets you wish to study, make hypotheses, run experiments to validate

Exam formats

Choose one:

- Written exam
- Project
  - Usually two types: vertical case study, or domain-independent investigation of one or more aspects of Bayes nets
  - **Case study**
    - Basic project: existing network (online library, tutorial, textbook, ...), run experiments following examples seen in class
    - Advanced project: original network (own idea, inspired from scientific article, ...), pose interesting questions, develop experiments to answer them, draw motivated conclusions
  - **Investigation**
    - Examples: modeling aspects, theoretical properties, inference methods, learning, software/libraries, ...

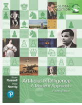
Exam formats

Choose one:

- Written exam
- Project
  - Must deliver **code** and 2-page **report** showing aim, model, methods, results (see guidelines & template)
  - Report must follow a fixed format; using LaTeX is encouraged
  - Oral discussion: 10 minutes presentation + 10 minutes Q&A, may include questions on topics seen in class
  - Evaluation based on work done [*4 points*], report [*2 pts*], presentation [*2 pts*] and Q&A [*2 pts*], plus *1 point* for outstanding work (honours)
  - Capacity limited to 7 project discussions each date
  - How to book a date: **sign up via AlmaEsami after uploading project+report** via link (be mindful of **limited capacity**)
  - Next dates: 9-Jan, 16-Jan, 6-Feb, 13-Feb 9am to noon (tentative)
  - More dates in June, July, and September

Approximate inference

Car insurance network



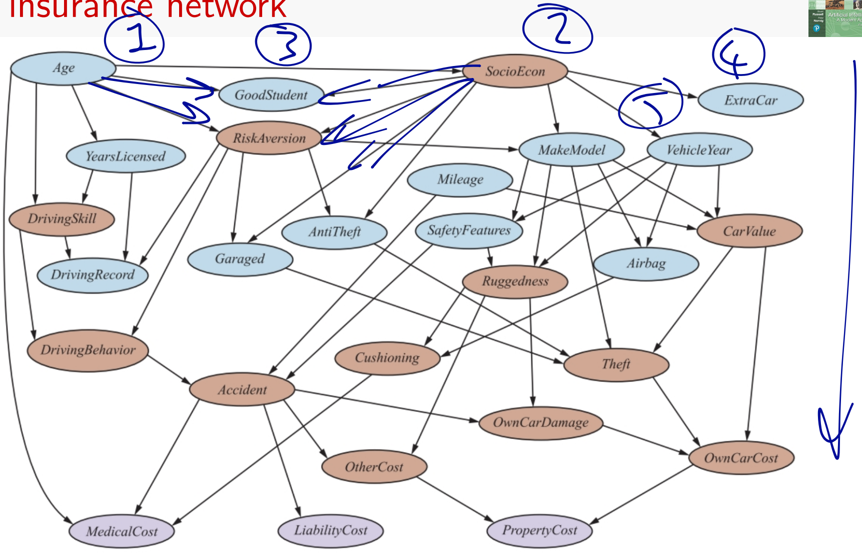
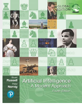


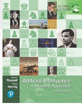
Figure 13.9 A Bayesian network for evaluating car insurance applications.



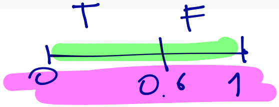
Car insurance network

Some observations

- Introduction of hidden variables (brown) yields sparse model with reasonable number of parameters
- What causal factors lead where is important question driving modeling choices (topology and hidden variables)
- Another modeling decision is continuous vs discrete ranges
  - Continuous: more precision but exact inference generally impossible
  - Discrete: many values generally yield higher inference cost, unless variable is observed (example: MakeModel)
- Cost of inference with exact enumeration: $\approx 10^8$ operations
- With variable elimination and good ordering: $\approx 10^5$ operations



Inference by stochastic simulation

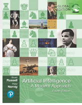


Basic idea:

1. Draw N samples from a sampling distribution S
2. Compute an approximate posterior probability $\hat{P}$
3. Show this converges to the true probability P

Outline:

- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior



Performance of approximation algorithms

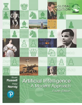
Absolute approximation: $|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})| \leq \epsilon$

Relative approximation: $\frac{|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})|}{P(X|\mathbf{e})} \leq \epsilon$

- Relative $\Rightarrow$ absolute since $0 \leq P \leq 1$
- Randomized algorithms may fail with probability at most δ

Theorem (Dagum and Luby, 1993)

Both absolute and relative approximation for either deterministic or randomized algorithms are NP-hard for any $\epsilon, \delta < 0.5$



Sampling from an empty network

```
function PRIOR-SAMPLE($bn$) returns an event sampled from $bn$
    inputs: $bn$, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
    $\mathbf{x} \leftarrow$ an event with $n$ elements
    for $i = 1$ to $n$ do
        $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i | parents(X_i))$
        given the values of $Parents(X_i)$ in $\mathbf{x}$
    return $\mathbf{x}$
```

Idea: sample each variable in turn, in topological order

Example

0.24, 0.7, 0.5, 0.18

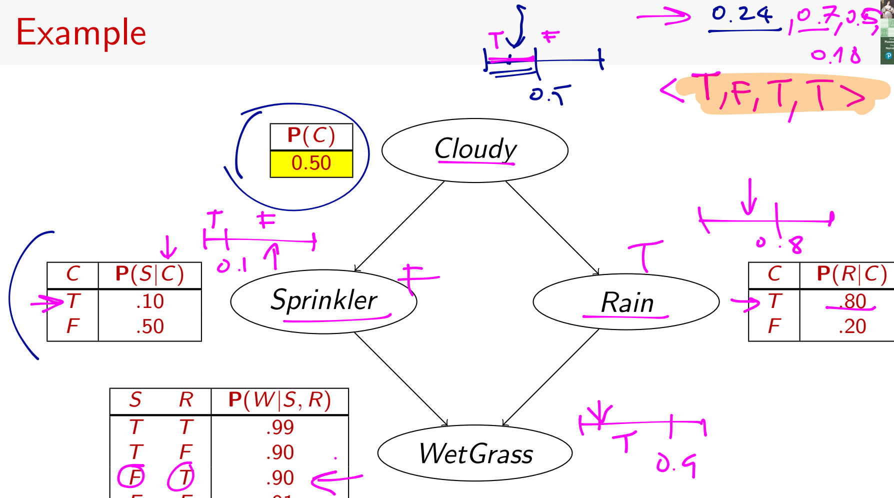
<T, F, T, T>

P(C)
0.50

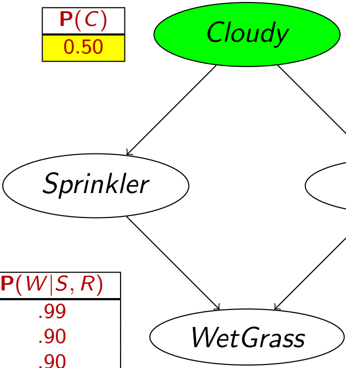
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S, R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01



Example



$\mathbf{P}(C)$
0.50

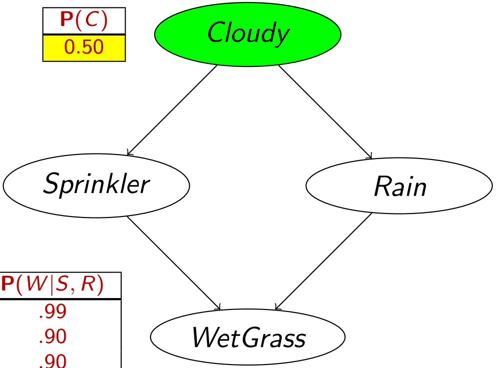
C	$\mathbf{P}(S\mid C)$
T	.10
F	.50

C	$\mathbf{P}(R\mid C)$
T	.80
F	.20

S	R	$\mathbf{P}(W\mid S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

$\mathbf{P}(C)$
0.50



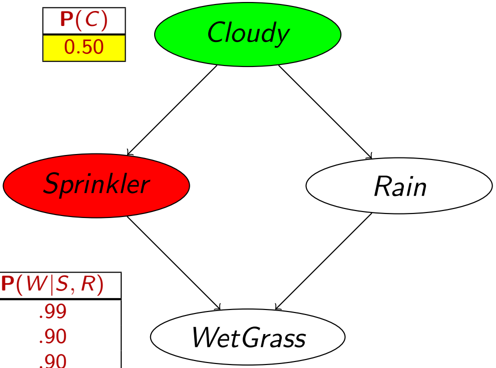
C	$\mathbf{P}(S\|C)$
T	.10
F	.50

C	$\mathbf{P}(R\|C)$
T	.80
F	.20

S	R	$\mathbf{P}(W\|S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

$\mathbf{P}(C)$
0.50



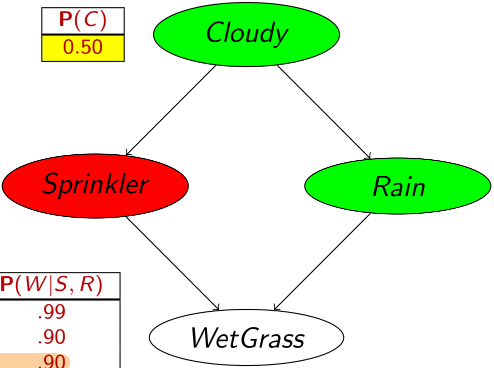
C	$\mathbf{P}(S \mid C)$
T	.10
F	.50

C	$\mathbf{P}(R \mid C)$
T	.80
F	.20

S	R	$\mathbf{P}(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

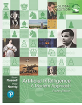
P(C)
0.50



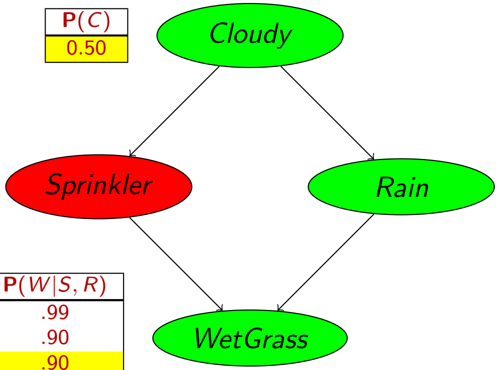
C	**P**$(S \mid C)$
T	.10
F	.50

C	**P**$(R \mid C)$
T	.80
F	.20

S	R	**P**$(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01



Example

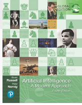


$\mathbf{P}(C)$
0.50

C	$\mathbf{P}(S \mid C)$
T	.10
F	.50

C	$\mathbf{P}(R \mid C)$
T	.80
F	.20

S	R	$\mathbf{P}(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

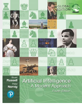


Sampling from an empty network

Probability that PRIORSAMPLE generates a particular event
$\boxed{S_{PS}}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i)) = P(x_1 \ldots x_n)$
i.e., the true prior probability
E.g., $S_{PS}(t,f,t,t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t,f,t,t)$
Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$
Then we have

$$\begin{aligned} \lim_{N\to\infty} \hat{P}(x_1,\ldots,x_n) &= \boxed{\lim_{N\to\infty} N_{PS}(x_1,\ldots,x_n)/N} \\ &= \boxed{S_{PS}}(x_1,\ldots,x_n) \\ &= \boxed{P}(x_1 \ldots x_n) \longleftarrow \end{aligned}$$

That is, estimates derived from PRIORSAMPLE are consistent
Shorthand: $\underline{\hat{P}(x_1,\ldots,x_n) \approx P(x_1 \ldots x_n)}$



Rejection sampling

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

```
function REJECTION-SAMPLING($X$,$\mathbf{e}$,$bn$,$N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
    local variables: $\mathbf{N}$, a vector of counts over $X$, initially zero
    for $j = 1$ to $N$ do
        $\mathbf{x} \leftarrow$ PRIOR-SAMPLE($bn$)
        if $\mathbf{x}$ is consistent with $\mathbf{e}$ then
            $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where $x$ is the value of $X$ in $\mathbf{x}$
    return NORMALIZE($\mathbf{N}$)
```

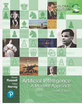
E.g., estimate $\mathbf{P}(Rain|Sprinkler = true)$ using 100 samples

27 samples have $Sprinkler = true$

Of these, 8 have $Rain = true$ and 19 have $Rain = false$.

$\hat{\mathbf{P}}(Rain|Sprinkler = true) = \text{NORMALIZE}(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle$

Similar to a basic real-world empirical estimation procedure



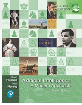
Analysis of rejection sampling

$$\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X,\mathbf{e}) \qquad \text{(algorithm defn.)}$$
$$= \mathbf{N}_{PS}(X,\mathbf{e})/N_{PS}(\mathbf{e}) \qquad \text{(normalized by } N_{PS}(\mathbf{e})\text{)}$$
$$\approx \boxed{\mathbf{P}(X,\mathbf{e})/P(\mathbf{e})} \qquad \text{(property of PriorSample)}$$
$$= \underline{\mathbf{P}(X|\mathbf{e})} \qquad \text{(defn. of conditional probability)}$$

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if $P(\mathbf{e})$ is small

$P(\mathbf{e})$ drops off exponentially with number of evidence variables!



Likelihood weighting

Idea: fix evidence variables, sample only nonevidence variables and weight each sample by the likelihood it accords the evidence

```
function LIKELIHOOD-WEIGHTING($X$,$\mathbf{e}$,$bn$,$N$) returns an estimate of $P(X|\mathbf{e})$
    local variables: $\mathbf{W}$, a vector of weighted counts over $X$, initially zero
    for $j = 1$ to $N$ do
        $\mathbf{x}$,$w$ $\leftarrow$ WEIGHTED-SAMPLE($bn$,$\mathbf{e}$)
        $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in $\mathbf{x}$
    return NORMALIZE($\mathbf{W}[X]$)
```

```
function WEIGHTED-SAMPLE($bn$,$\mathbf{e}$) returns an event and a weight
    $\mathbf{x} \leftarrow$ an event with $n$ elements initialized from $\mathbf{e}$
    $w \leftarrow 1$
    for $i = 1$ to $n$ do
        if $X_i$ is an evidence variable with value $x_i$ in $\mathbf{e}$ then
            $w \leftarrow w \times P(X_i = x_i | parents(X_i))$
        else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i | parents(X_i))$
    return $\mathbf{x}$, $w$
```

Likelihood weighting example

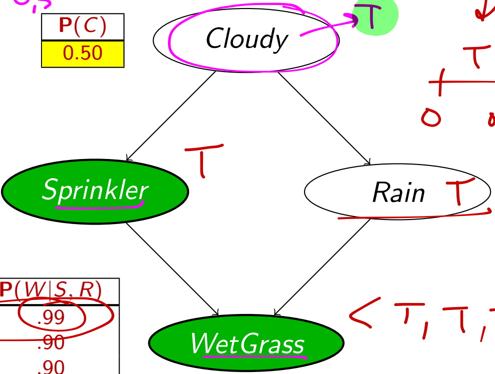
$\rightarrow$ 0.4, $\boxed{0.7}$, 0.1, 0.9

T 0.5

P(*C*)
0.50

T

0 0.8



C	**P**(*S*|*C*)
T	.10
F	.50

C	**P**(*R*|*C*)
T	.80
F	.20

S	*R*	**P**(*W*|*S*, *R*)
T	*T*	.99
T	*F*	.90
F	*T*	.90
F	*F*	.01

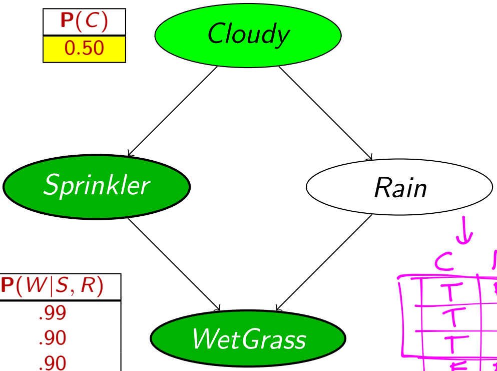
< T, T, T, T >

$w = 0.1 \times 0.99$

$w = 1.0$

Likelihood weighting example

P(*C*)
0.50



C	**P**(*S*|*C*)
T	.10
F	.50

C	**P**(*R*|*C*)
T	.80
F	.20

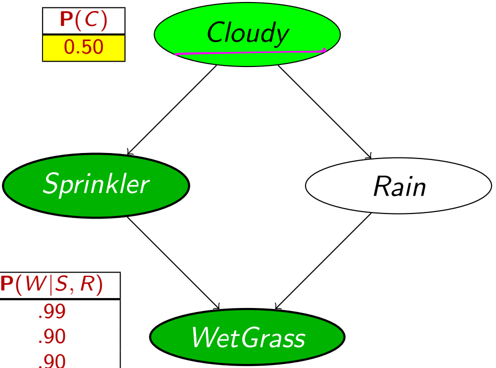
S	*R*	**P**(*W*|*S*, *R*)
T	*T*	.99
T	*F*	.90
F	*T*	.90
F	*F*	.01

$w = 1.0$

C	R	W
T	F	.1
T	T	.99
T	T	.2
F	F	.1
F	T	.5

$$P(\underset{T}{C} \mid s, g) = \frac{.1 + .99 + .2}{\Sigma w_i}$$

Likelihood weighting example



$\mathbf{P}(C)$
0.50

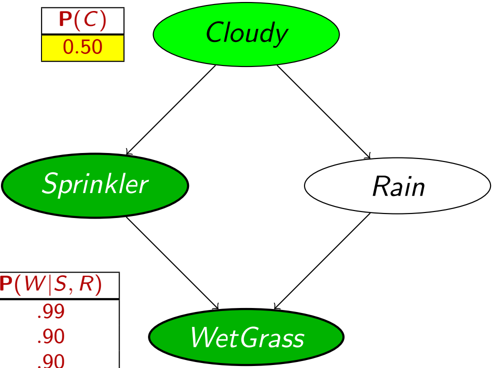
C	$\mathbf{P}(S\|C)$
T	.10
F	.50

C	$\mathbf{P}(R\|C)$
T	.80
F	.20

S	R	$\mathbf{P}(W\|S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$ $\times$ 0.1

Likelihood weighting example



$\mathbf{P}(C)$
0.50

C	$\mathbf{P}(S\|C)$
T	.10
F	.50

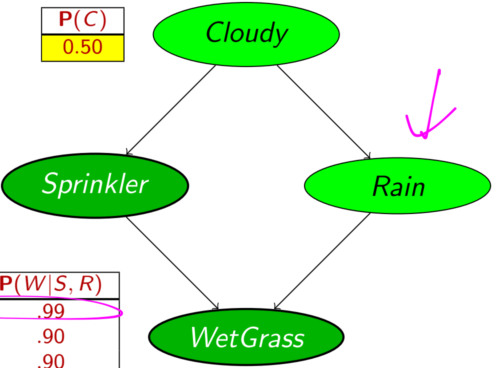
C	$\mathbf{P}(R\|C)$
T	.80
F	.20

S	R	$\mathbf{P}(W\|S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

$w = 1.0 \times 0.1$

Likelihood weighting example



P(*C*)
0.50

C	**P**(*S*|*C*)
T	.10
F	.50

C	**P**(*R*|*C*)
T	.80
F	.20

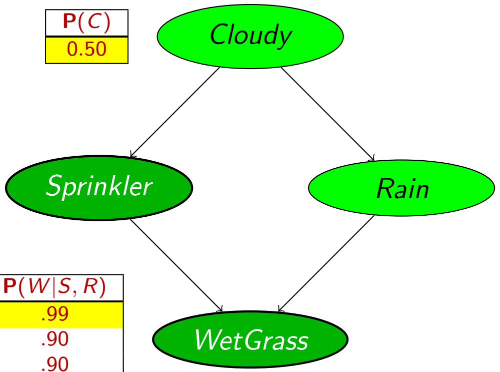
S	*R*	**P**(*W*|*S*, *R*)
T	*T*	.99
T	*F*	.90
F	*T*	.90
F	*F*	.01

$w = 1.0$

$\boxed{w = 1.0 \times 0.1}$

Likelihood weighting example

P(*C*)
0.50



C	**P**(*S*|*C*)
T	.10
F	.50

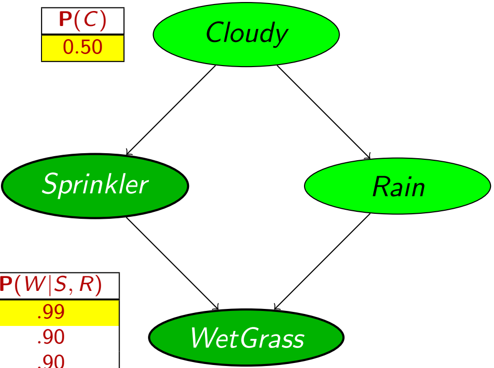
C	**P**(*R*|*C*)
T	.80
F	.20

S	*R*	**P**(*W*|*S*, *R*)
T	*T*	.99
T	*F*	.90
F	*T*	.90
F	*F*	.01

$w = 1.0$

$w = 1.0 \times 0.1$

Likelihood weighting example



$\mathbf{P}(C)$
0.50

C	$\mathbf{P}(S\|C)$
T	.10
F	.50

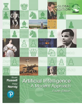
C	$\mathbf{P}(R\|C)$
T	.80
F	.20

S	R	$\mathbf{P}(W\|S,R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

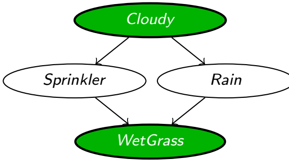
$w = 1.0 \times 0.1$

$w = 1.0 \times 0.1 \times 0.99 = 0.099$ ✓



Analysis of likelihood weighting

Sampling probability for WeightedSample is
$S_{WS}(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i|parents(Z_i))$
Note: pays attention to evidence in **ancestors** only
$\Rightarrow$ somewhere "in between" prior and posterior distribution



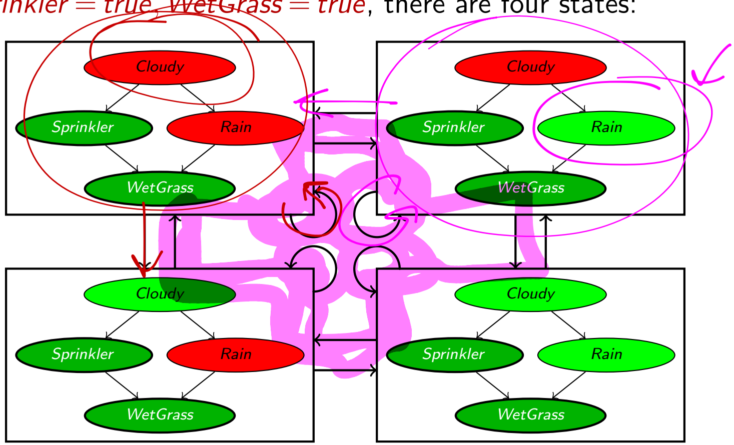
Weight for a given sample $\mathbf{z},\mathbf{e}$ is $w(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{m} P(e_i|parents(E_i))$

Weighted sampling probability is $S_{WS}(\mathbf{z},\mathbf{e})w(\mathbf{z},\mathbf{e})$
$= \prod_{i=1}^{l} P(z_i|parents(Z_i)) \quad \prod_{i=1}^{m} P(e_i|parents(E_i))$
$= P(\mathbf{z},\mathbf{e})$ (by standard global semantics of network)

Hence likelihood weighting returns consistent estimates but performance still degrades with many evidence variables because a few samples have nearly all the total weight

Markov chain Monte Carlo example

With *Sprinkler* = *true*, *WetGrass* = *true*, there are four states:

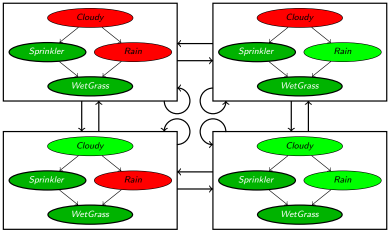


Wander about for a while, average what you see

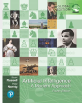
Markov chain Monte Carlo example

Estimate $\mathbf{P}(Rain|Sprinkler = true, WetGrass = true)$
Sample *Cloudy* or *Rain* given its Markov blanket, repeat.
Count number of times *Rain* is true and false in the samples.
E.g., visit 100 states, 31 have $Rain = true$, 69 have $Rain = false$

$$\hat{\mathbf{P}}(Rain|Sprinkler = true, WetGrass = true)$$
$$= \text{NORMALIZE}(\langle 31, 69\rangle) = \langle 0.31, 0.69\rangle$$



The Markov chain

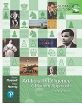


"State" of network = current assignment to all variables.
Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

Theorem

Theorem: chain approaches stationary distribution: long-run fraction of time spent in each state is exactly proportional to its posterior probability

Markov chain Monte Carlo



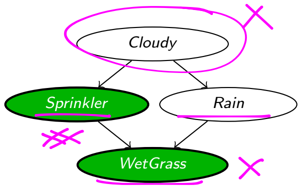
```
function MCMC-ASK($X$,$\mathbf{e}$,$bn$,$N$) returns an estimate of $\mathbf{P}(X|\mathbf{e})$
    local variables: $\mathbf{N}$, a vector of counts for each value of $X$, initially zero
                     $\mathbf{Z}$, the nonevidence variables in $bn$
                     $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$
    initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
    for $j = 1$ to $N$ do
        for each $Z_i$ in $\mathbf{Z}$ do
            set the value of $Z_i$ in $\mathbf{x}$ by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
            $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in $\mathbf{x}$
    return NORMALIZE($\mathbf{N}$)
```

Can also choose a variable to sample at random each time

Markov blanket sampling

Markov blanket of *Cloudy* is . . . S, R

Markov blanket of *Rain* is . . .



Probability given the Markov blanket is calculated as follows:

$$P(x_i'|mb(X_i)) = P(x_i'|parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$$

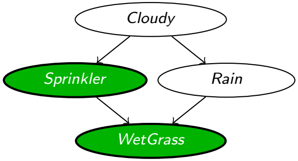
Main computational problems:

1. Difficult to tell if convergence has been achieved
2. Can be wasteful if Markov blanket is large: $P(X_i|mb(X_i))$ won't change much (law of large numbers)

Markov blanket sampling

Markov blanket of *Cloudy* is
$\{Sprinkler, Rain\}$

Markov blanket of *Rain* is
$\{Cloudy, Sprinkler, WetGrass\}$



Probability given the Markov blanket is calculated as follows:
$P(x_i'|mb(X_i)) = P(x_i'|parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$

Main computational problems:

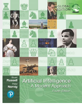
1. Difficult to tell if convergence has been achieved
2. Can be wasteful if Markov blanket is large:
   $P(X_i|mb(X_i))$ won't change much (law of large numbers)

MCMC analysis: Outline

- Transition probability $q(\mathbf{x} \rightarrow \mathbf{x}')$
- Occupancy probability $\pi_t(\mathbf{x})$ at time t
- Equilibrium condition on π_t defines stationary distribution $\pi(\mathbf{x})$

  Note: stationary distribution depends on choice of $q(\mathbf{x} \rightarrow \mathbf{x}')$
- Pairwise detailed balance on states guarantees equilibrium
- Gibbs sampling transition probability: sample each variable given current values of all others

  $\Rightarrow$ detailed balance with the true posterior
- For Bayesian networks, Gibbs sampling reduces to sampling conditioned on each variable's Markov blanket



Stationary distribution

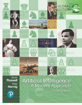
$\pi_t(\mathbf{x})$ = probability in state $\mathbf{x}$ at time t
$\pi_{t+1}(\mathbf{x}')$ = probability in state $\mathbf{x}'$ at time $t+1$
π_{t+1} in terms of π_t and $q(\mathbf{x} \rightarrow \mathbf{x}')$

$$\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$$

Stationary distribution: $\pi_t = \pi_{t+1} = \pi$

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') \qquad \text{for all } \mathbf{x}'$$

If π exists, it is unique (specific to $q(\mathbf{x} \rightarrow \mathbf{x}')$)
In equilibrium, expected "outflow" = expected "inflow"



Detailed balance

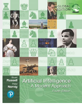
"Outflow" = "inflow" for each pair of states:

$$\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \qquad \text{for all } \mathbf{x},\ \mathbf{x}'$$

Detailed balance $\Rightarrow$ stationarity:

$$\begin{aligned} \sum_{\mathbf{x}} \pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') &= \sum_{\mathbf{x}} \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \\ &= \pi(\mathbf{x}') \sum_{\mathbf{x}} q(\mathbf{x}' \to \mathbf{x}) \\ &= \pi(\mathbf{x}') \end{aligned}$$

MCMC algorithms typically constructed by designing a transition probability q that is in detailed balance with desired π



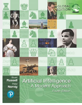
Gibbs sampling

Sample each variable in turn, given **all other variables**
Sampling X_i, let $\bar{\mathbf{X}}_i$ be all other nonevidence variables
Current values are x_i and $\bar{\mathbf{x}}_i$; $\mathbf{e}$ is fixed
Transition probability is given by

$$q(\mathbf{x} \to \mathbf{x}') = q(x_i, \bar{\mathbf{x}}_i \to x_i', \bar{\mathbf{x}}_i) = P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e})$$

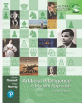
This gives detailed balance with true posterior $P(\mathbf{x}|\mathbf{e})$:

$$\begin{aligned}
\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') &= P(\mathbf{x}|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) = P(x_i, \bar{\mathbf{x}}_i|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) \\
&= P(x_i|\bar{\mathbf{x}}_i, \mathbf{e})P(\bar{\mathbf{x}}_i|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) \quad \text{(chain rule)} \\
&= P(x_i|\bar{\mathbf{x}}_i, \mathbf{e})P(x_i', \bar{\mathbf{x}}_i|\mathbf{e}) \quad \text{(chain rule backwards)} \\
&= q(\mathbf{x}' \to \mathbf{x})\pi(\mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x})
\end{aligned}$$



Compiling approximate inference

- Sampling operates on Bayes net represented as data structure
- Naive implementation would yield huge amounts of repeated operations (example: to find a node's parents)
- Repetition completely unnecessary
- For example, in the alarm network, an operation required by MCMC-Ask is sampling from $\mathbf{P}(Z_i | mb(Z_i))$
- Sampling from $\mathbf{P}(Earthquake | mb(Earthquake))$ requires
  - looking up children and parents of Earthquake in the network structure
  - looking up their current values
  - using these values to look up into CPT
  - computing products to obtain distribution
  - sample from distribution
- Solution: *compile* the network into model-specific inference code

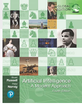


Compiling approximate inference

For example:

```
$r \leftarrow$ a uniform random sample from [0,1]
**if** $Alarm = true$
    **then if** $Burglary = true$
        **then return** $[r < 0.0020212]$
        **else return** $[r < 0.36755]$
    **else if** $Burglary = true$
        **then return** $[r < 0.0016672]$
        **else return** $[r < 0.0014222]$
```

Code not pretty but 2/3 orders of magnitude faster than MCMC-Ask



Summary

Exact inference by variable elimination

- only for discrete random variables and very special cases of continuous variables with canonical distributions
- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by LW, MCMC:

- Can handle arbitrary combinations of discrete and continuous variables
- LW does poorly when there is lots of (downstream) evidence
- LW, MCMC generally insensitive to topology
- Convergence can be very slow with probabilities close to 1 or 0
- Compiling could be crucial

Questions?