# Probabilistic Reasoning

Bùi Tiến Lên

2022

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Contents

# Notation

| symbol | meaning |
|---|---|
| $a, b, c, N \ldots$ | scalar number |
| $\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$ | column vector |
| $\boldsymbol{X}, \boldsymbol{Y} \ldots$ | matrix |
| $\mathbb{R}$ | set of real numbers |
| $\mathbb{Z}$ | set of integer numbers |
| $\mathbb{N}$ | set of natural numbers |
| $\mathbb{R}^D$ | set of vectors |
| $\mathcal{D}, \mathcal{X}, \mathcal{Y}, \ldots$ | set |
| $\mathcal{A}$ | algorithm |

| symbol | meaning |
|---|---|
| $X, Y \ldots$ | random variable |
| $\boldsymbol{X}, \boldsymbol{Y} \ldots$ | multivariate random variable |
| $x, y \ldots$ | value |
| $\boldsymbol{x}, \boldsymbol{y} \ldots$ | vector |
| $p, pr, P, Pr$ | probability |

# Inference tasks

- **Simple queries**:

$$P(X \mid E = e) \tag{1}$$

- **Conjunctive queries**:

$$P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e}) \tag{2}$$

- **Optimal decisions**: decision networks include utility information; probabilistic inference required for

$$P(outcome \mid action, evidence)$$

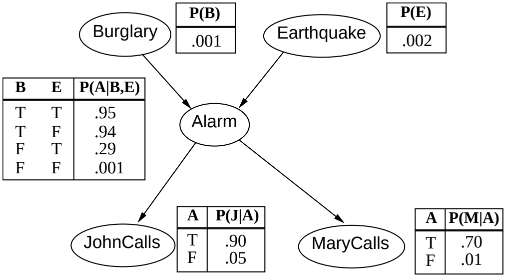
- **Value of information**: “which evidence to seek next?”
- **Sensitivity analysis**: “which probability values are most critical?”
- **Explanation**: “why do I need a new starter motor?”

# Exact inference by enumeration

# Inference by enumeration

## Idea

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation



# Inference by enumeration (cont.)

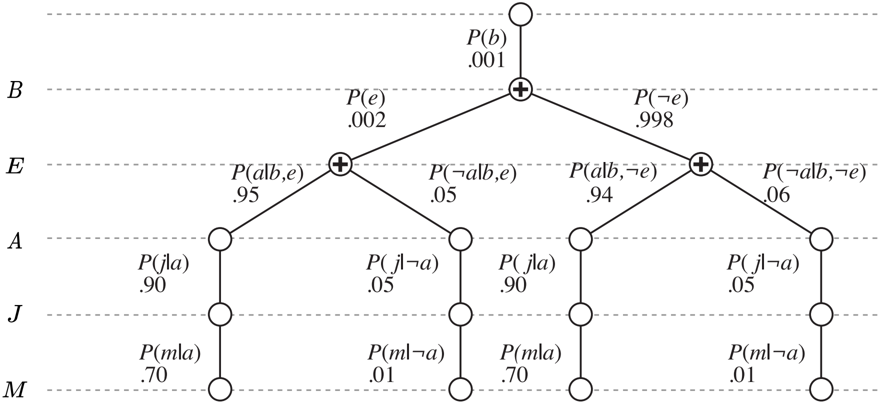
- Simple query on the burglary network:

$$\begin{aligned} P(B \mid j, m) &= P(B, j, m)/P(j, m) \\ &= \alpha P(B, j, m) \\ &= \alpha \sum_e \sum_a P(B, e, a, j, m) \end{aligned}$$

- Rewrite full joint entries using product of CPT entries:

$$\begin{aligned} &P(B \mid j, m) \\ &= \alpha \sum_e \sum_a P(B)P(e)P(a \mid B, e)P(j \mid a)P(m \mid a) \\ &= \alpha\, P(B) \sum_e P(e) \sum_a P(a \mid B, e)P(j \mid a)P(m \mid a) \\ &= \alpha\, \langle 0.00059224, 0.0014919 \rangle \\ &= \langle 0.284, 0.716 \rangle \end{aligned}$$

# Evaluation tree




- Enumeration is inefficient: repeated computation; e.g., computes $P(j \mid a)P(m \mid a)$ for each value of $e$

# Enumeration algorithm

- Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
inputs: X, the query variable
        e, observed values for variables E
        bn, a Bayes net with variables {X} ∪ E ∪ Y # Y:hidden variables
  Q(X) ← a distribution over X, initially empty
  for each value x_i of X do
    Q(x_i) ← ENUMERATE-ALL(bn.vars, e_{x_i})
      where e_{x_i} is e extended with X = x_i
return NORMALIZE(Q(X))

function ENUMERATE-ALL(vars, e) returns a real number
  if EMPTY?(vars) then return 1.0
  Y ← FIRST(vars)
  if Y has value y in e
  then return P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e)
  else return ∑_y P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e_y)
                where e_y is e extended with Y = y
```

# Exact inference by variable elimination

# Inference by variable elimination

## Idea

Carry out summations right-to-left, storing intermediate results (**factors**) to avoid recomputation

$$
\begin{aligned}
P(B \mid j, m) &= \alpha \underbrace{P(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum_a \underbrace{P(a \mid B, e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M} \\
&= \alpha\, P(B) \sum_e P(e) \sum_a P(a \mid B, e) P(j \mid a) f_M(a) \\
&= \alpha\, P(B) \sum_e P(e) \sum_a P(a \mid B, e) f_J(a) f_M(a) \\
&= \alpha\, P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
&= \alpha\, P(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \ \text{(sum out A)} \\
&= \alpha\, P(B) f_{\bar{E}\bar{A}JM}(b) \ \text{(sum out E)} \\
&= \alpha\, f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

# Variable elimination: Basic operations

- **Summing out** a variable from a product of factors:
  - move any constant factors outside the summation
  - add up submatrices in pointwise product of remaining factors

$$\begin{aligned}\sum_x f_1 \times \cdots \times f_k &= f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k \\ &= f_1 \times \cdots \times f_i \times f_{\bar{X}} \end{aligned} \tag{3}$$

  assuming $f_1, \ldots, f_i$ do not depend on $X$

- **Pointwise product** of factors $f_1$ and $f_2$:

$$\begin{aligned} &f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l) \\ &= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l) \end{aligned} \tag{4}$$

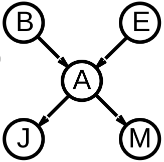
# Variable elimination algorithm

**function** $\textsc{Elimination-Ask}(X, \mathbf{e}, bn)$
**returns** a distribution over $X$
**inputs**: $X$, the query variable
        $\mathbf{e}$, observed values for variables $\mathbf{E}$
        $bn$, a Bayesian network specifying
            joint distribution $P(X_1, \ldots, X_n)$
  $factors \leftarrow \emptyset$
  **for each** $var$ **in** $\textsc{Order}(bn.\textsc{Vars})$ **do**
    $factors \leftarrow [\textsc{Make-Factor}(var, \mathbf{e}) \mid factors]$
    **if** $var$ is a hidden variable **then**
      $factors \leftarrow \textsc{Sum-Out}(var, factors)$
**return** $\textsc{Normalize}(\textsc{Pointwise-Product}(factors))$

# Irrelevant variables

- Consider the query $P(JohnCalls \mid Burglary = true)$

$$P(J \mid b) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(J \mid a) \sum_m P(m \mid a)$$




Sum over $m$ is identically 1; $M$ is *irrelevant* to the query

- Here, $X = JohnCalls$, $\mathbf{E} = \{Burglary\}$, and $Ancestors(\{X\} \cup \mathbf{E}) = \{Alarm, Earthquake\}$ so $MaryCalls$ is irrelevant (Compare this to backward chaining from the query in Horn clause KBs)

**Theorem 1**

*$Y$ is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathbf{E})$*

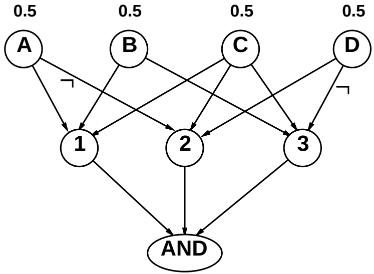
# Complexity of exact inference

- **Singly connected** networks (or **polytrees**):
  - any two nodes are connected by at most one (undirected) path
  - time and space cost of variable elimination are $O(d^k n)$
- **Multiply connected** networks:
  - can reduce 3SAT to exact inference $\implies$ NP-hard
  - equivalent to *counting* 3SAT models $\implies$ P-complete

**1. A v B v C**

**2. C v D v ¬A**

**3. B v C v ¬D**



# Approximate inference by stochastic simulation

- Random Number
- Direct Sampling
- Rejection Sampling
- Likelihood Weighting

# Uniform Random Numbers

- The basis of all of these simulation methods is in the generation of random numbers
- The simplest method is the linear **congruential generator** to generate **uniformly distributed random numbers**

$$x_{n+1} = (ax_n + c) \bmod m \quad (5)$$

where $a$,$c$, and $m$ are parameters that have to be chosen carefully, the initial input $x_0$ is known as the **seed**

- For example, we can use $m = 2^{32}$, $a = 1,664,525$ and $c = 1,013,904,223$

# Gaussian Random Numbers

**The Box–Muller algorithm**

1. Pick two uniformly distributed random numbers $0 \leq U_1, U_2 \leq 1$
2. Set $\theta = 2\pi U_1$ and $r = \sqrt{-2\ln(U_2)}$
3. Then $x = r\cos(\theta)$ and $y = r\sin(\theta)$ are **independent Gaussian-distributed variables** with zero mean ($\mu = 0$) and unit variance ($\sigma = 1$)

# Sampling for Single Categorical Variable

- Want to sample values of a random variable $X$ whose domain is {true, false}, with probability distribution

| true | false |
|---|---|
| 0.4 | 0.6 |

- Simple approach

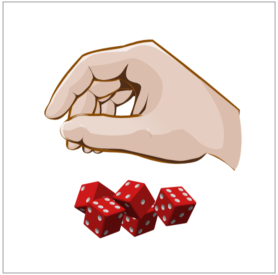
```
r = uniform_random([0, 1])
if r < 0.4:
  sample = true
else:
  sample = false
```

# Inference by stochastic simulation

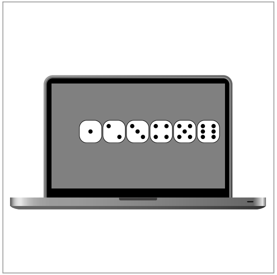
## Basic idea

1. **Draw** $N$ samples from a **sampling distribution** $S$ (**sampling** is a lot like **repeated simulation**)
2. **Compute** an approximate posterior probability $\hat{P}$
3. Show this converges to the true probability $P$

Experiment



Stochastic simulation



# Inference by stochastic simulation (cont.)

**Why sample**?

- **Learning**: get samples from a distribution you don't know
- **Inference**: getting a sample is faster than computing the right answer (e.g. with variable elimination)

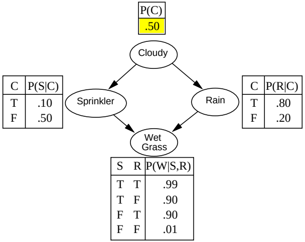
**Approaches**:

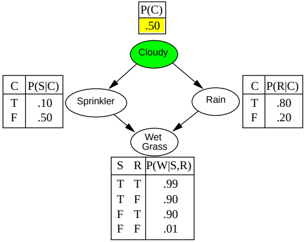
1. Sampling from an empty network
2. Rejection sampling: reject samples disagreeing with evidence
3. Likelihood weighting: use evidence to weight samples
4. Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

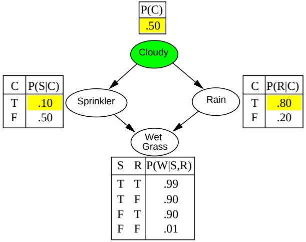
# Sampling from an empty network

```
function PRIOR-SAMPLE(bn)
returns an event sampled from the prior specified by bn
inputs: bn, a Bayesian network specifying
              joint distribution P(X_1, ..., X_n)
  x ← an event with n elements
  foreach variable X_i in X_1, …, X_n do
    x[i] ← a random sample from P(X_i | parents(X_i))
return x
```

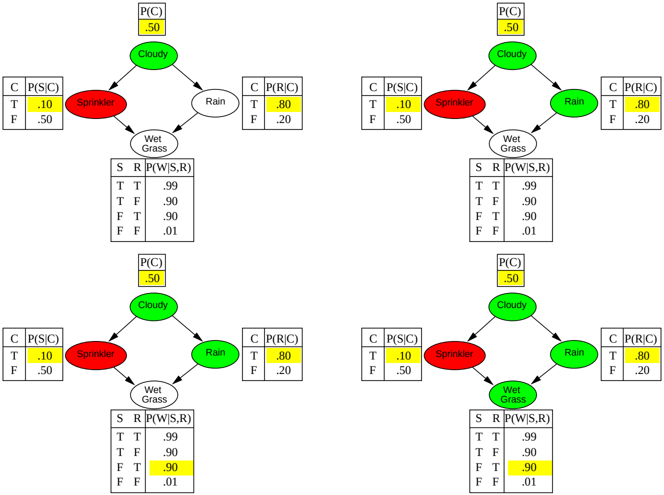
# Example


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

# Example (cont.)



# Example (cont.)

- **Estimate** $P(Cloudy, Sprinkler, Rain, WetGrass)$ using 100 samples

| # | Cloudy | Sprinkler | Rain | Wet Grass |
|---|---|---|---|---|
| 1 | T | F | T | T |
| 2 | T | T | T | T |
| 3 | F | T | T | F |
| 4 | T | F | T | T |
| 5 | F | F | F | T |
| ... | ... | ... | ... | ... |
| 100 | T | F | T | F |

# Analysis

- Probability that PriorSample generates a particular event

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(x_i)) = P(x_1 \ldots x_n)$$

- Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$
- Then we have

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

That is, estimates derived from PriorSample are **consistent**

# Rejection sampling

- $\hat{P}(X \mid e)$ estimated from samples agreeing with $e$

```
function REJECTION-SAMPLING(X, e, bn, N)
returns an estimate of P(X | e)
inputs: X, the query variable
        e, observed values for variables E
        bn, a Bayesian network
        N, the total number of samples to be generated
local variables: N, a vector of counts for each value of X,
                    initially zero
  for j = 1 to N do
    x ← PRIOR-SAMPLE(bn)
    if x is consistent with e then
      N[x] ← N[x] + 1 where x is the value of X in x
  return NORMALIZE(N)
```

# Exampe

- **Estimate** $P(Rain \mid Sprinkler = true)$ using 100 samples
- **Results** 27 samples have $Sprinkler = true$; of these, 8 have $Rain = true$ and 19 have $Rain = false$.

| # | Cloudy | Sprinkler | Rain | Wet Grass |
|---|---|---|---|---|
| 1 | T | F | T | T |
| 2 | T | T | T | T |
| 3 | F | T | T | F |
| 4 | T | F | T | T |
| 5 | F | F | F | T |
| ... | ... | ... | ... | ... |
| 100 | T | F | T | F |

$\rightarrow$

| # | Sprinkler | Rain |
|---|---|---|
| 1 | T | T |
| 2 | T | T |
| 3 | T | T |
| 4 | T | F |
| 5 | T | T |
| ... | ... | ... |
| 27 | T | F |

# Analysis of rejection sampling

$$
\begin{aligned}
\hat{P}(X \mid e) &= \alpha N_{PS}(X, e) && \text{(algorithm defn.)} \\
&= N_{PS}(X, e)/N_{PS}(e) && \text{(normalized by } N_{PS}(e)\text{)} \\
&\approx P(X, e)/P(e) && \text{(property of } \textsc{PriorSample}\text{)} \\
&= P(X \mid e) && \text{(defn. of conditional probability)}
\end{aligned}
$$

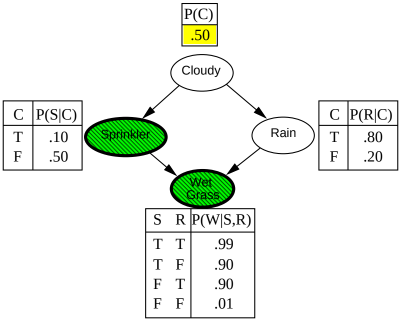
- Hence rejection sampling returns consistent posterior estimates
- Problem: hopelessly expensive if $P(e)$ is small. $P(e)$ drops off exponentially with number of evidence variables!

# Likelihood weighting

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

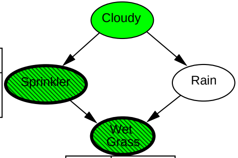
```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns an estimate of P(X | e)
inputs: X, the query variable
        e, observed values for variables E
        bn, a Bayesian network specifying joint distribution P(X_1, ..., X_n)
        N, the total number of samples to be generated
local variables: W, a vector of weighted counts for each value of X, initially zero
  for j = 1 to N do
    x, w ← WEIGHTED-SAMPLE(bn, e)
    W[x] ← W[x] + w where x is the value of X in x
  return NORMALIZE(W)

function WEIGHTED-SAMPLE(bn, e) returns an event and a weight
  w ← 1
  x ← an event with n elements initialized from e
  foreach variable X_i in X_1, ..., X_n do
    if X_i is an evidence variable with value x_i in e
    then w ← w × P(X_i = x_i | parents(X_i))
    else x[i] ← a random sample from P(X_i | parents(X_i))
  return x, w
```

# Likelihood weighting example




$$w = 1.0$$

# Likelihood weighting example (cont.)




| P(C) |
|---|
| .50 |

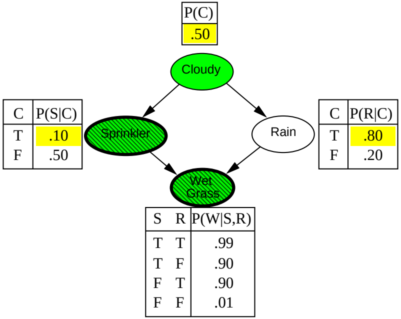
| C | P(S\|C) |
|---|---|
| T | .10 |
| F | .50 |

| 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 |

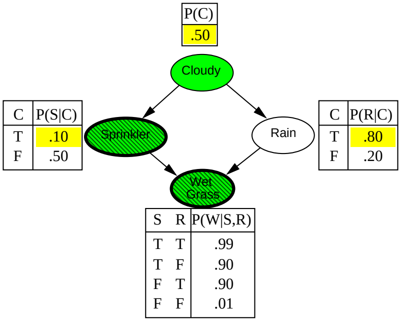
$w = 1.0$

# Likelihood weighting example (cont.)




$w = 1.0$

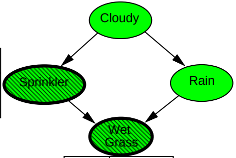
# Likelihood weighting example (cont.)




$$w = 1.0 \times 0.1$$

# Likelihood weighting example (cont.)

| P(C) |
|---|
| .50 |



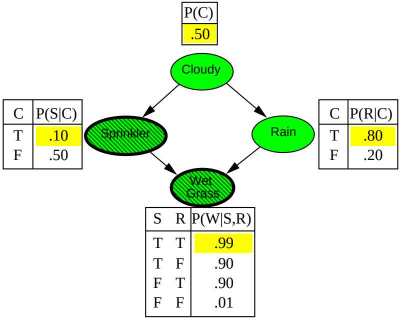

| C | P(S\|C) |
|---|---|
| T | .10 |
| F | .50 |

| 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 |

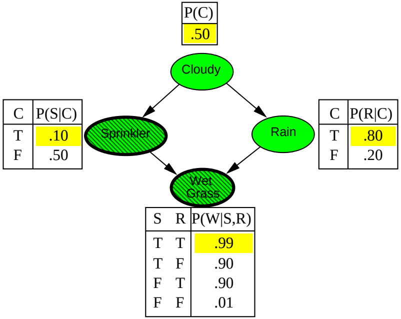
$$w = 1.0 \times 0.1$$

# Likelihood weighting example (cont.)




$$w = 1.0 \times 0.1$$

# Likelihood weighting example (cont.)




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

# Likelihood weighting example (cont.)

- **Estimate** $P(Rain \mid Sprinkler = true, WetGrass = true)$ using 100 samples

| # | Cloudy | Sprinkler | Rain | Wet Grass | weight $w$ |
|---|---|---|---|---|---|
| 1 | T | T | T | T | 0.099 |
| 2 | F | T | T | T | ... |
| 3 | T | T | T | T | ... |
| 4 | F | T | T | T | ... |
| 5 | T | T | F | T | ... |
| ... | ... | ... | ... | ... | ... |
| 100 | F | T | T | T | ... |

# Likelihood weighting analysis

- Sampling probability for WeightedSample is

$$S_{WS}(z, e) = \prod_{i=1}^{l} P(z_i \mid parents(Z_i))$$

- Note: pays attention to evidence in *ancestors* only $\implies$ somewhere "in between" prior and posterior distribution
- Weight for a given sample $z, e$ is

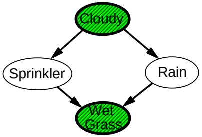
$$w(z, e) = \prod_{i=1}^{m} P(e_i \mid parents(E_i))$$

# Likelihood weighting analysis (cont.)

- Weighted sampling probability is

$$
\begin{aligned}
& S_{WS}(z, e) w(z, e) \\
& = \prod_{i=1}^{l} P(z_i \mid parents(Z_i)) \prod_{i=1}^{m} P(e_i \mid parents(E_i)) \\
& = P(z, e) \text{ (by standard global semantics of network)}
\end{aligned}
$$

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



# Approximate inference by Markov chain Monte Carlo (MCMC)

- Gibbs Sampling
- MCMC for Bayesian Network

# State

## Concept 1

**State** $\boldsymbol{x}$ is current assignment to all variables.

## Concept 2

- **Transition probability** $q(\boldsymbol{x} \rightarrow \boldsymbol{x}')$ is the probability to change from state $\boldsymbol{x}$ to $\boldsymbol{x}'$
- **Occupancy probability** $\pi_t(\boldsymbol{x})$ is the probability in state $\boldsymbol{x}$ at time $t$

# 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 $t+1$
- $\pi_{t+1}$ in terms of $\pi_t$ and $q(\mathbf{x} \to \mathbf{x}')$

$$\pi_{t+1}(q(\mathbf{x} \to \mathbf{x}')') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \to \mathbf{x}') \tag{6}$$

## Concept 3

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

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \to \mathbf{x}') \text{ for all } \mathbf{x}' \tag{7}$$

- If $\pi$ exists, it is unique (specific to $q(\mathbf{x} \to \mathbf{x}')$)

# Detailed balance

## Concept 4

**Detailed balance**: "Outflow" = "inflow" for each pair of states:

$$\pi(\boldsymbol{x})q(\boldsymbol{x} \rightarrow \boldsymbol{x}') = \pi(\boldsymbol{x}')q(\boldsymbol{x}' \rightarrow \boldsymbol{x}) \text{ for all } \boldsymbol{x}, \boldsymbol{x}' \tag{8}$$

## Theorem 2

*Detailed balance $\Longrightarrow$ stationarity:*

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

# Gibbs sampling

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

1. Generate next state by sampling one variable given *all other variables*
2. Sample each variable in turn, keeping evidence fixed

$$\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2 \to \ldots \to \mathbf{x}_n$$

# Gibbs sampling (cont.)

- 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

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

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

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

# Analysis

## Theorem 3

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

- **Gibbs sampling** transition probability: sample each variable given current values of all others $\implies$ detailed balance with the true posterior
- For Bayesian networks, Gibbs sampling reduces to sampling conditioned on each variable's Markov blanket

# Performance of approximation algorithms

- **Absolute approximation**:

$$\mid P(X \mid \mathbf{e}) - \hat{P}(X \mid \mathbf{e}) \mid \leq \epsilon$$

- **Relative approximation**:

$$\frac{\mid P(X \mid \mathbf{e}) - \hat{P}(X \mid \mathbf{e}) \mid}{P(X \mid \mathbf{e})} \leq \epsilon$$

Relative $\implies$ absolute since $0 \leq P \leq 1$ (may be $O(2^{-n})$)
Randomized algorithms may fail with probability at most $\delta$
Polytime approximation: $\text{poly}(n, \epsilon^{-1}, \log \delta^{-1})$

**Theorem 4 (Dagum and Luby (1993))**

*Both absolute and relative approximation for either deterministic or randomized algorithms are NP-hard for any $\epsilon, \delta < 0.5$ (absolute approximation polytime with no evidence—Chernoff bounds)*

# Approximate inference using MCMC

## Concept 5

"State" of network = current assignment to all variables.

1. Generate next state by sampling one variable given Markov blanket
2. Sample each variable in turn, keeping evidence fixed

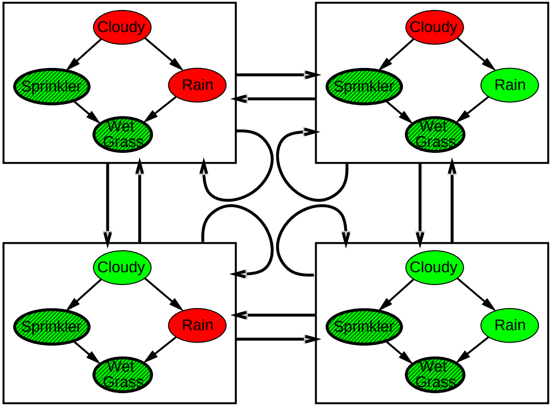
**Note**: We can also choose a variable to sample at random each time

# Approximate inference using MCMC (cont.)

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

# The Markov chain

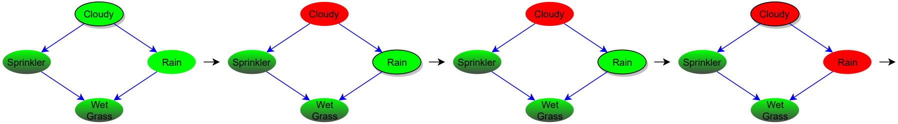
- With *Sprinkler* $=$ *true*, *WetGrass* $=$ *true*, there are four states:




- Wander about for a while, average what you see

# MCMC example

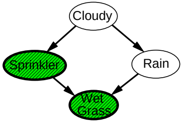
- **Estimate** $P(Rain \mid Sprinkler = true, WetGrass = true)$
  - **Sample** *Cloudy* or *Rain* given its Markov blanket, repeat.
  - **Count** number of times *Rain* is **true** and **false** in the samples.
  - **Result**: visit 100 states, 31 have $Rain = true$, 69 have $Rain = false$




| # | Cloudy | Sprinkler | Rain | Wet Grass |
|---|---|---|---|---|
| 1 | T | T | T | T |
| 2 | F | T | T | T |
| 3 | F | T | T | T |
| 4 | F | T | F | T |
| ... | ... | ... | ... | ... |
| 100 | T | T | F | T |

# Markov blanket sampling

- Markov blanket of *Cloudy* is *Sprinkler* and *Rain*
- Markov blanket of *Rain* is *Cloudy*, *Sprinkler*, and *WetGrass*




- Probability given the Markov blanket is calculated as follows:

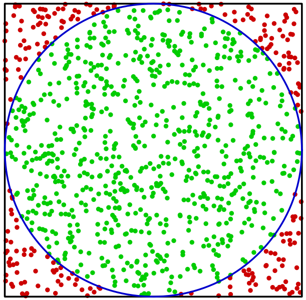
$$P(x_i' \mid mb(X_i)) = P(x_i' \mid parents(X_i)) \prod_{Z_j \in children(X_i)} P(z_j \mid parents(Z_j))$$

- Easily implemented in message-passing parallel systems
- Main computational problems:
  1. Difficult to tell if convergence has been achieved
  2. Can be wasteful if Markov blanket is large: $P(X_i \mid mb(X_i))$ won't change much (law of large numbers)

# Sampling for Continous Variables

- Reject Sampling
- Metropolis-Hastings

# Reject Sampling

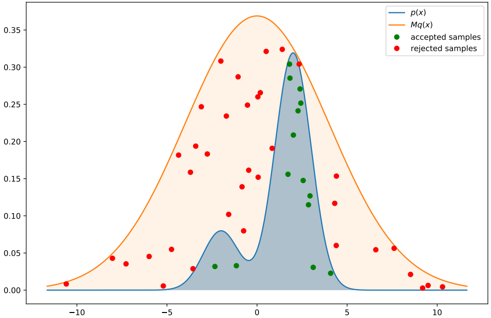


$$\pi \approx 4 \times \frac{\text{number of points inside circle}}{\text{number of points inside square}}$$

# Reject Sampling (cont.)

$p(x)$ is a **target distribution**, $q(x)$ is an an **easy-to-sample distribution** and $M$ is a constant such that $\forall x \in \mathcal{X}, p(x) \leq Mq(x)$

1. Sample $x$ from a known $q(x)$
2. Sample $u$ from a $\mathcal{U}[0, Mq(x)]$
3. If $u < p(x)$ accept the sample $x$; otherwise reject the sample



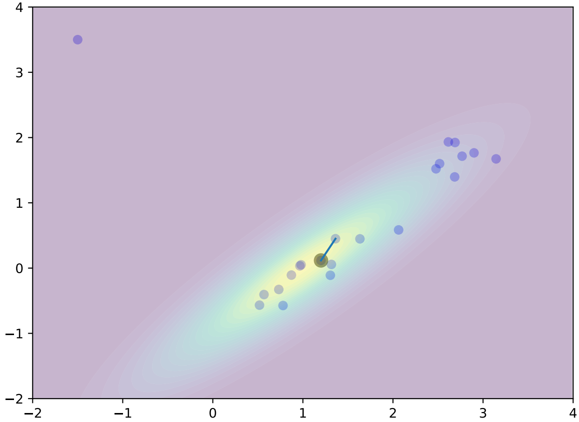
# Metropolis-Hastings algorithm

1. Choose an initial value for a sample $x_0$
2. Propose a new sample value $x_{i+1}$ given $x_i$ from $q(x \mid x_i)$
3. Compute the probability of accepting a new parameter value by using the Metropolis-Hastings criteria:

$$\rho = \min\left(1, \frac{p(x_{i+1})}{p(x_i)}\right) \tag{9}$$

4. Sample $u$ from a $\mathcal{U}[0,1]$
5. If $u < \rho$ we accept the new value $x_{i+1}$; otherwise, we stay in the old value $x_i$

# Example



# References


Goodfellow, I., Bengio, Y., and Courville, A. (2016).
*Deep learning*.
MIT press.

Lê, B. and Tô, V. (2014).
*Cở sở trí tuệ nhân tạo*.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
*Artificial intelligence: a modern approach*.
Pearson Education Limited.