

# Inference in belief networks

## CHAPTER 15.3-4 + NEW

# Outline

- ◇ Exact inference by enumeration
- ◇ Exact inference by variable elimination
- ◇ Approximate inference by stochastic simulation
- ◇ Approximate inference by Markov chain Monte Carlo

## Inference tasks

Simple queries: compute posterior marginal  $\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$

e.g.,  $P(NoGas|Gauge = empty, Lights = on, Starts = false)$

Conjunctive queries:  $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e}) = \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

Optimal decisions: decision networks include utility information;  
probabilistic inference required for  $P(outcome|action, evidence)$

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?

## Inference by enumeration

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

Simple query on the burglary network:

$$\begin{aligned} & \mathbf{P}(B|J = true, M = true) \\ &= \mathbf{P}(B, J = true, M = true) / P(J = true, M = true) \\ &= \alpha \mathbf{P}(B, J = true, M = true) \\ &= \alpha \sum_e \sum_a \mathbf{P}(B, e, a, J = true, M = true) \end{aligned}$$

Rewrite full joint entries using product of CPT entries:

$$\begin{aligned} & P(B = true | J = true, M = true) \\ &= \alpha \sum_e \sum_a P(B = true) P(e) P(a|B = true, e) P(J = true|a) P(M = true|a) \\ &= \alpha P(B = true) \sum_e P(e) \sum_a P(a|B = true, e) P(J = true|a) P(M = true|a) \end{aligned}$$

# Enumeration algorithm

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

```
ENUMERATIONASK( $X, \mathbf{e}, bn$ ) returns a distribution over  $X$ 
inputs:  $X$ , the query variable
          $\mathbf{e}$ , evidence specified as an event
          $bn$ , a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$ 

 $\mathbf{Q}(X) \leftarrow$  a distribution over  $X$ 
for each value  $x_i$  of  $X$  do
    extend  $\mathbf{e}$  with value  $x_i$  for  $X$ 
     $\mathbf{Q}(x_i) \leftarrow$  ENUMERATEALL(VARS[ $bn$ ],  $\mathbf{e}$ )
return NORMALIZE( $\mathbf{Q}(X)$ )
```

---

```
ENUMERATEALL( $vars, \mathbf{e}$ ) returns a real number
if EMPTY?( $vars$ ) then return 1.0
else do
     $Y \leftarrow$  FIRST( $vars$ )
    if  $Y$  has value  $y$  in  $\mathbf{e}$ 
        then return  $P(y \mid Pa(Y)) \times$  ENUMERATEALL(REST( $vars$ ),  $\mathbf{e}$ )
    else return  $\sum_y P(y \mid Pa(Y)) \times$  ENUMERATEALL(REST( $vars$ ),  $\mathbf{e}_y$ )
    where  $\mathbf{e}_y$  is  $\mathbf{e}$  extended with  $Y = y$ 
```

## Inference by variable elimination

Enumeration is inefficient: repeated computation

e.g., computes  $P(J = true|a)P(M = true|a)$  for each value of  $e$

Variable elimination: carry out summations right-to-left,  
storing intermediate results (factors) to avoid recomputation

$$\begin{aligned}
 & \mathbf{P}(B|J = true, M = true) \\
 &= \alpha \underbrace{\mathbf{P}(B)}_B \underbrace{\sum_e P(e)}_E \underbrace{\sum_a \mathbf{P}(a|B, e)}_A \underbrace{P(J = true|a)}_J \underbrace{P(M = true|a)}_M \\
 &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B, e) P(J = true|a) f_M(a) \\
 &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B, e) f_J(a) f_M(a) \\
 &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
 &= \alpha \mathbf{P}(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \text{ (sum out } A) \\
 &= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E) \\
 &= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
 \end{aligned}$$

## Variable elimination: Basic operations

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

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

E.g.,  $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Summing out a variable from a product of factors: move any constant factors outside the summation:

$$\sum_x f_1 \times \dots \times f_k = f_1 \times \dots \times f_i \sum_x f_{i+1} \times \dots \times f_k = f_1 \times \dots \times f_i \times f_{\bar{X}}$$

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

# Variable elimination algorithm

```
function ELIMINATIONASK( $X, \mathbf{e}, bn$ ) returns a distribution over  $X$ 
inputs:  $X$ , the query variable
          $\mathbf{e}$ , evidence specified as an event
          $bn$ , a belief network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$ 

if  $X \in \mathbf{e}$  then return observed point distribution for  $X$ 
 $factors \leftarrow []$ ;  $vars \leftarrow \text{REVERSE}(\text{VARS}[bn])$ 
for each  $var$  in  $vars$  do
     $factors \leftarrow [\text{MAKEFACTOR}(var, \mathbf{e}) | factors]$ 
    if  $var$  is a hidden variable then  $factors \leftarrow \text{SUMOUT}(var, factors)$ 
return NORMALIZE(PPOINTWISEPRODUCT( $factors$ ))
```



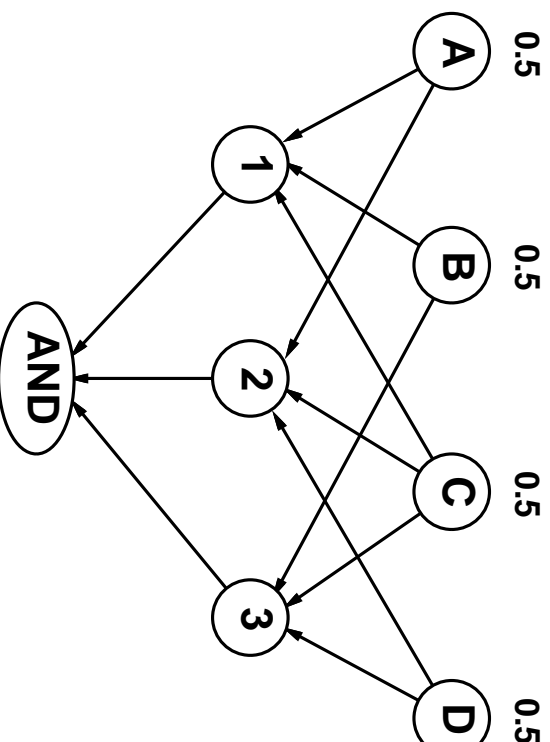
# Complexity of exact inference

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  $\Rightarrow$  NP-hard
- equivalent to *counting* 3SAT models  $\Rightarrow$  #P-complete



1.  $A \vee B \vee C$
2.  $C \vee D \vee \sim A$
3.  $B \vee C \vee \sim D$

# 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
- MCMC: sample from a stochastic process whose stationary distribution is the true posterior

# Sampling from an empty network

```

function PRIORSAMPLE(bn) returns an event sampled from  $\mathbf{P}(X_1, \dots, X_n)$  specified by bn
  x  $\leftarrow$  an event with n elements
  for i = 1 to n do
    xi  $\leftarrow$  a random sample from  $\mathbf{P}(X_i \mid \text{Parents}(X_i))$ 
  return x

```

$$\mathbf{P}(\textit{Cloudy}) = \langle 0.5, 0.5 \rangle$$

sample  $\rightarrow$  *true*

$$\mathbf{P}(\textit{Sprinkler} \mid \textit{Cloudy}) = \langle 0.1, 0.9 \rangle$$

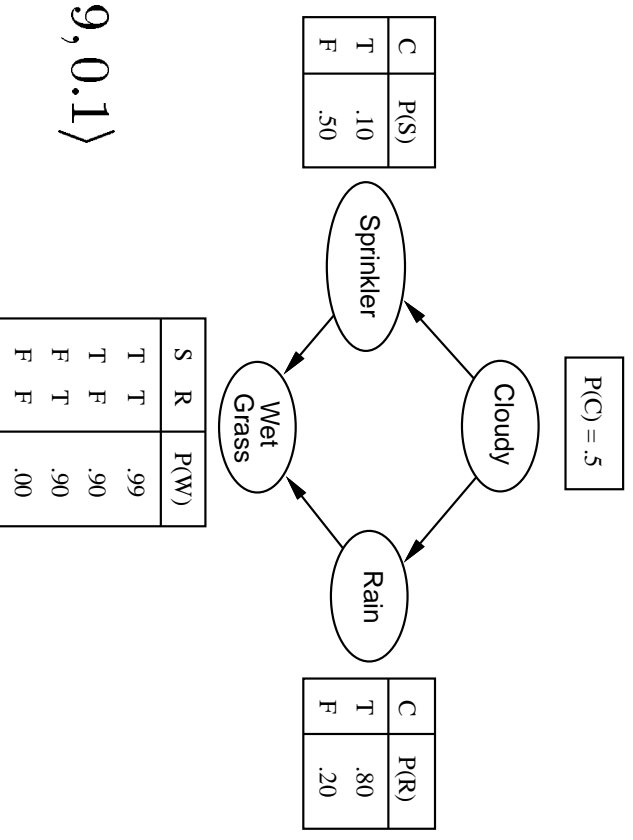
sample  $\rightarrow$  *false*

$$\mathbf{P}(\textit{Rain} \mid \textit{Cloudy}) = \langle 0.8, 0.2 \rangle$$

sample  $\rightarrow$  *true*

$$\mathbf{P}(\textit{WetGrass} \mid \neg \textit{Sprinkler}, \textit{Rain}) = \langle 0.9, 0.1 \rangle$$

sample  $\rightarrow$  *true*



## Sampling from an empty network contd.

Probability that `PRIORSAMPLE` generates a particular event

$$S_{PS}(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(X_i)) = P(x_1 \dots x_n)$$

i.e., the true prior probability

Let  $N_{PS}(\mathbf{Y} = \mathbf{y})$  be the number of samples generated for which  $\mathbf{Y} = \mathbf{y}$ , for any set of variables  $\mathbf{Y}$ .

Then  $\hat{P}(\mathbf{Y} = \mathbf{y}) = N_{PS}(\mathbf{Y} = \mathbf{y})/N$  and

$$\begin{aligned} \lim_{N \rightarrow \infty} \hat{P}(\mathbf{Y} = \mathbf{y}) &= \sum_{\mathbf{h}} S_{PS}(\mathbf{Y} = \mathbf{y}, \mathbf{H} = \mathbf{h}) \\ &= \sum_{\mathbf{h}} P(\mathbf{Y} = \mathbf{y}, \mathbf{H} = \mathbf{h}) \\ &= P(\mathbf{Y} = \mathbf{y}) \end{aligned}$$

That is, estimates derived from `PRIORSAMPLE` are consistent

## Rejection sampling

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

```
function REJECTION_SAMPLING( $X, \mathbf{e}, bn, N$ ) returns an approximation to  $P(X|\mathbf{e})$ 
   $\mathbf{N}[X] \leftarrow$  a vector of counts over  $X$ , initially zero
  for  $j = 1$  to  $N$  do
     $\mathbf{x} \leftarrow \text{PRIOR\_SAMPLE}(bn)$ 
    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( $\mathbf{N}[X]$ )
```

E.g., estimate  $P(Rain|Sprinkler = true)$  using 100 samples  
27 samples have *Sprinkler = true*

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

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

Similar to a basic real-world empirical estimation procedure

## Analysis of rejection sampling

$$\begin{aligned}\hat{\mathbf{P}}(X|\mathbf{e}) &= \frac{\mathbf{N}_{PS}(X, \mathbf{e})}{\mathbf{N}_{PS}(\mathbf{e})} && \text{(algorithm defn.)} \\ &= \mathbf{N}_{PS}(X, \mathbf{e}) / N_{PS}(\mathbf{e}) && \text{(normalized by } N_{PS}(\mathbf{e})) \\ &\approx \mathbf{P}(X, \mathbf{e}) / P(\mathbf{e}) && \text{(property of } \mathbf{P}_{PRIORSAMPLE}) \\ &= \mathbf{P}(X|\mathbf{e}) && \text{(defn. of conditional probability)}\end{aligned}$$

Hence rejection sampling returns consistent posterior estimates

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

## Likelihood weighting

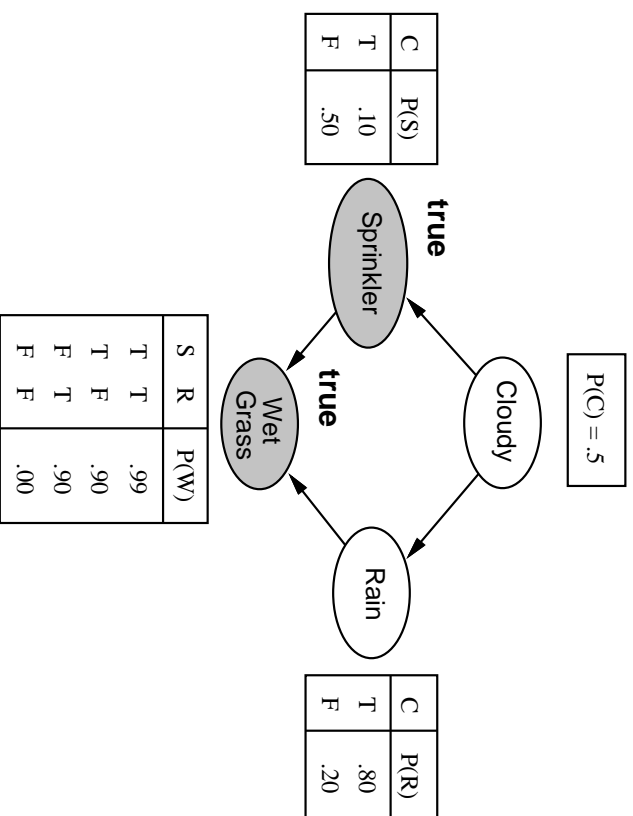
Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```
function WEIGHTEDSAMPLE( $bn, e$ ) returns an event and a weight
 $\mathbf{x} \leftarrow$  an event with  $n$  elements;  $w \leftarrow 1$ 
for  $i = 1$  to  $n$  do
    if  $X_i$  has a value  $x_i$  in  $e$ 
        then  $w \leftarrow w \times P(X_i = x_i \mid Parents(X_i))$ 
        else  $x_i \leftarrow$  a random sample from  $\mathbf{P}(X_i \mid Parents(X_i))$ 
    return  $\mathbf{x}, w$ 

function LIKELIHOODWEIGHTING( $X, e, bn, N$ ) returns an approximation to  $P(X|e)$ 
 $\mathbf{W}[X] \leftarrow$  a vector of weighted counts over  $X$ , initially zero
for  $j = 1$  to  $N$  do
     $\mathbf{x}, w \leftarrow$  WEIGHTEDSAMPLE( $bn$ )
     $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$  where  $x$  is the value of  $X$  in  $\mathbf{x}$ 
return NORMALIZE( $\mathbf{W}[X]$ )
```

# Likelihood weighting example

Estimate  $P(Rain | Sprinkler = true, WetGrass = true)$





## LW example contd.

Sample generation process:

1.  $w \leftarrow 1.0$
2. Sample  $\mathbf{P}(\textit{Cloudy}) = \langle 0.5, 0.5 \rangle$ ; say *true*
3. *Sprinkler* has value *true*, so  
 $w \leftarrow w \times P(\textit{Sprinkler} = \textit{true} | \textit{Cloudy} = \textit{true}) = 0.1$
4. Sample  $\mathbf{P}(\textit{Rain} | \textit{Cloudy} = \textit{true}) = \langle 0.8, 0.2 \rangle$ ; say *true*
5. *WetGrass* has value *true*, so  
 $w \leftarrow w \times P(\textit{WetGrass} = \textit{true} | \textit{Sprinkler} = \textit{true}, \textit{Rain} = \textit{true}) = 0.099$

## Likelihood weighting analysis

Sampling probability for WEIGHTEDSAMPLE is

$$S_{WS}(\mathbf{y}, \mathbf{e}) = \prod_{i=1}^l P(y_i | Parents(Y_i))$$

Note: pays attention to evidence in *ancestors* only

$\Rightarrow$  somewhere “in between” prior and posterior distribution

Weight for a given sample  $\mathbf{y}, \mathbf{e}$  is

$$w(\mathbf{y}, \mathbf{e}) = \prod_{i=1}^m P(e_i | Parents(E_i))$$

Weighted sampling probability is

$$\begin{aligned} S_{WS}(\mathbf{y}, \mathbf{e}) w(\mathbf{y}, \mathbf{e}) \\ &= \prod_{i=1}^l P(y_i | Parents(Y_i)) \prod_{i=1}^m P(e_i | Parents(E_i)) \\ &= P(\mathbf{y}, \mathbf{e}) \text{ (by standard global semantics of network)} \end{aligned}$$

Hence likelihood weighting returns consistent estimates  
but performance still degrades with many evidence variables

## Approximate inference using MCMC

“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

```
function MCMC-ASK( $X, \mathbf{e}, bn, N$ ) returns an approximation to  $P(X|\mathbf{e})$ 
  local variables:  $N[X]$ , a vector of counts over  $X$ , initially zero
                   $\mathbf{Y}$ , 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  $\mathbf{Y}$ 
  for  $j = 1$  to  $N$  do
     $N[x] \leftarrow N[x] + 1$  where  $x$  is the value of  $X$  in  $\mathbf{x}$ 
    for each  $Y_i$  in  $\mathbf{Y}$  do
      sample the value of  $Y_i$  in  $\mathbf{x}$  from  $\mathbf{P}(Y_i|MB(Y_i))$  given the values of  $MB(Y_i)$  in  $\mathbf{x}$ 
  return NORMALIZE( $N[X]$ )
```

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

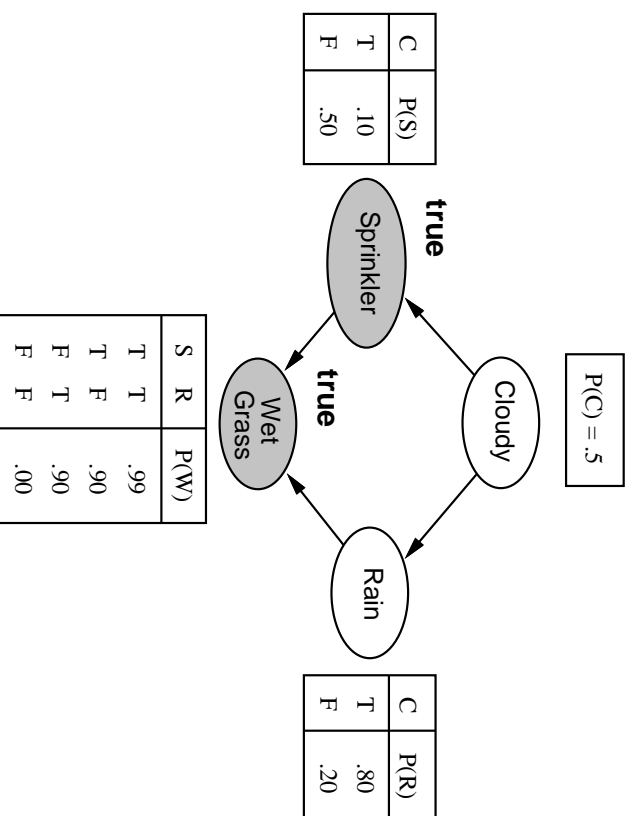
# MCMC Example

Estimate  $P(Rain|Sprinkler = true, WetGrass = true)$

Sample *Cloudy* then *Rain*, repeat.

Count number of times *Rain* is true and false in the samples.

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



## MCMC example contd.

Random initial state: *Cloudy* = *true* and *Rain* = *false*

1.  $\mathbf{P}(\textit{Cloudy}|MB(\textit{Cloudy})) = \mathbf{P}(\textit{Cloudy}|\textit{Sprinkler}, \neg \textit{Rain})$   
sample  $\rightarrow$  *false*
2.  $\mathbf{P}(\textit{Rain}|MB(\textit{Rain})) = \mathbf{P}(\textit{Rain}|\neg \textit{Cloudy}, \textit{Sprinkler}, \textit{WetGrass})$   
sample  $\rightarrow$  *true*

Visit 100 states

31 have *Rain* = *true*, 69 have *Rain* = *false*

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

# MCMC analysis: Outline

Transition probability  $q(\mathbf{y} \rightarrow \mathbf{y}')$

**D**occupancy probability  $\pi_t(\mathbf{y})$  at time  $t$

Equilibrium condition on  $\pi_t$  defines stationary distribution  $\pi(\mathbf{y})$

Note: stationary distribution depends on choice of  $q(\mathbf{y} \rightarrow \mathbf{y}')$

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

$\pi_t(\mathbf{y})$  = probability in state  $\mathbf{y}$  at time  $t$

$\pi_{t+1}(\mathbf{y}') =$  probability in state  $\mathbf{y}'$  at time  $t + 1$

$\pi_{t+1}$  in terms of  $\pi_t$  and  $q(\mathbf{y} \rightarrow \mathbf{y}')$

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

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

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

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

In equilibrium, expected “outflow” = expected “inflow”

## Detailed balance

“Outflow” = “inflow” for each pair of states:

$$\pi(y)q(y \rightarrow y') = \pi(y')q(y' \rightarrow y) \quad \text{for all } y, y'$$

Detailed balance  $\Rightarrow$  stationarity:

$$\begin{aligned}\sum_y \pi(y)q(y \rightarrow y') &= \sum_y \pi(y')q(y' \rightarrow y) \\ &= \pi(y') \sum_y q(y' \rightarrow y) \\ &= \pi(y')\end{aligned}$$

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



## Gibbs sampling

Sample each variable in turn, given *all other variables*

Sampling  $Y_i$ , let  $\bar{\mathbf{Y}}_i$  be all other nonevidence variables

Current values are  $y_i$  and  $\bar{y}_i$ ;  $\mathbf{e}$  is fixed

Transition probability is given by

$$q(\mathbf{y} \rightarrow \mathbf{y}') = q(y_i, \bar{\mathbf{y}}_i \rightarrow y'_i, \bar{\mathbf{y}}_i) = P(y'_i | \bar{\mathbf{y}}_i, \mathbf{e})$$

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

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

## Markov blanket sampling

A variable is independent of all others given its Markov blanket:

$$P(y'_i | \bar{Y}_i, \mathbf{e}) = P(y'_i | MB(Y_i))$$

Probability given the Markov blanket is calculated as follows:

$$P(y'_i | MB(Y_i)) = P(y'_i | Parents(Y_i)) \prod_{Z_j \in Children(Y_i)} P(z_j | Parents(Z_j))$$

Hence computing the sampling distribution over  $Y_i$  for each flip requires just  $cd$  multiplications if  $Y_i$  has  $c$  children and  $d$  values; can cache it if  $c$  not too large.

Main computational problems:

- 1) Difficult to tell if convergence has been achieved
- 2) Can be wasteful if Markov blanket is large:

$P(Y_i | MB(Y_i))$  won't change much (law of large numbers)

# 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  $0 \leq P \leq 1$  (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 (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)