Bayesian networks: Inference and learning

CS194-10 Fall 2011 Lecture 22

Outline

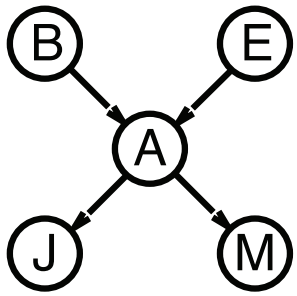
- ♢ Exact inference (briefly)
- ♢ Approximate inference (rejection sampling, MCMC)
- ♢ Parameter learning

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:

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

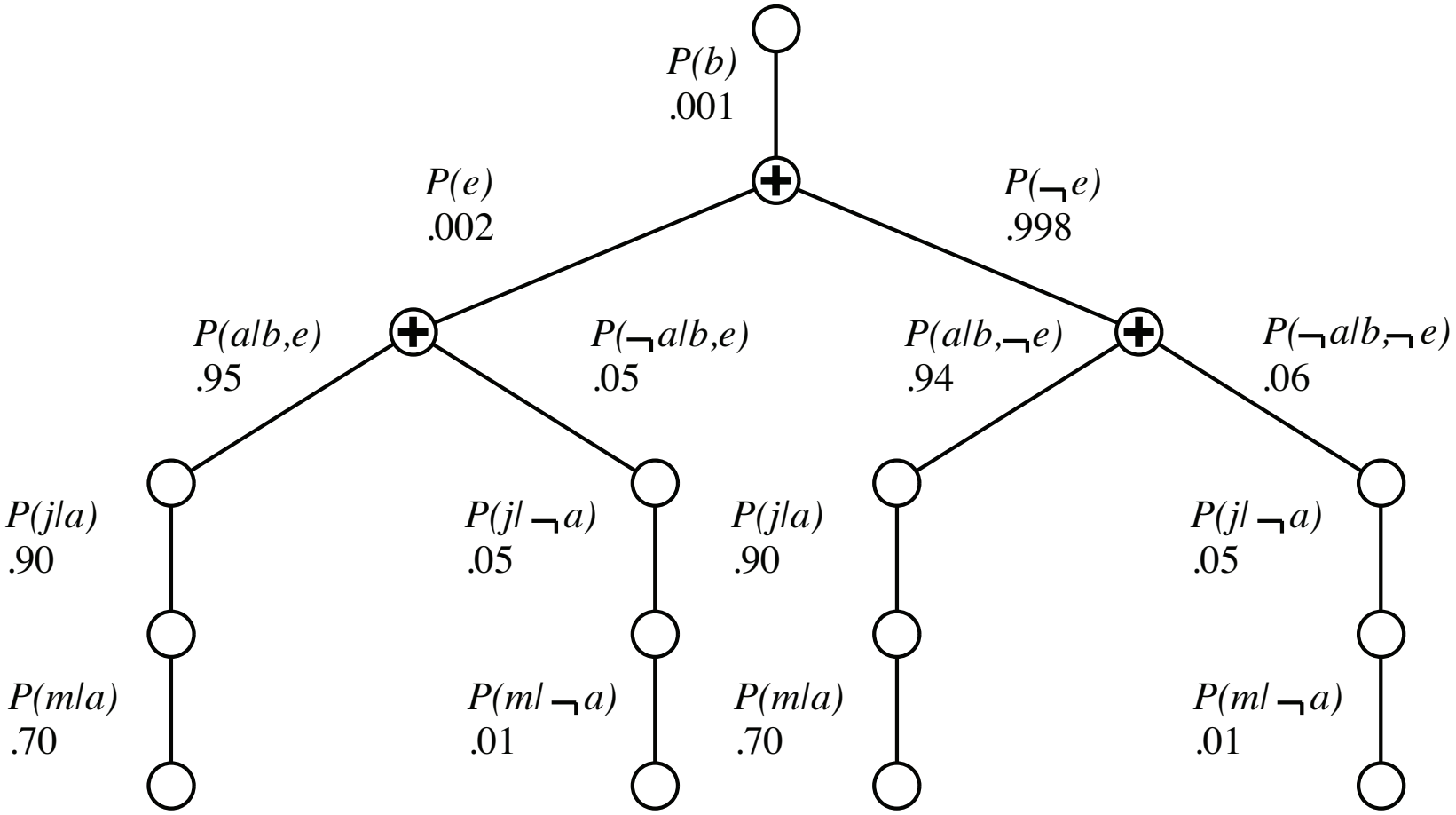


Rewrite full joint entries using product of CPT entries:

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

Recursive depth-first enumeration: $O(D)$ space, $O(K^D)$ time

Evaluation tree



Enumeration is inefficient: repeated computation

e.g., computes $P(j|a)P(m|a)$ for each value of e

Efficient exact inference

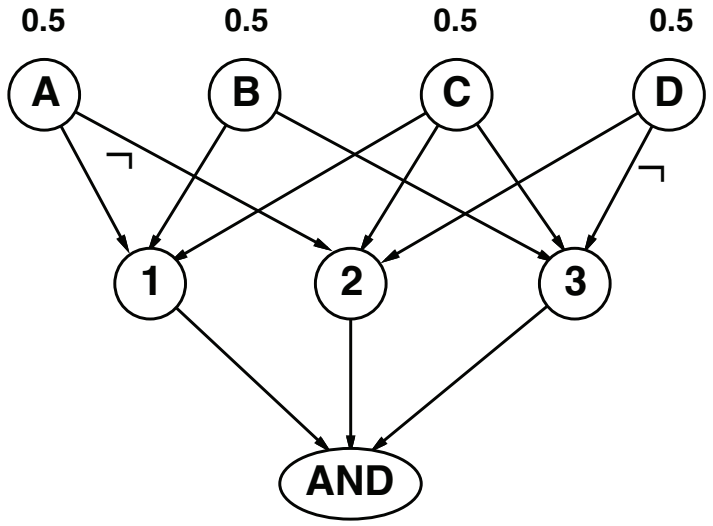
Junction tree and variable elimination algorithms
avoid repeated computation (generalized from of dynamic programming)

- ♢ Singly connected networks (or polytrees):
  - – any two nodes are connected by at most one (undirected) path
  - – time and space cost of exact inference are $O(K^L D)$
- ♢ Multiply connected networks:
  - – can reduce 3SAT to exact inference ⇒ NP-hard
  - – equivalent to **counting** 3SAT models ⇒ #P-complete

1. A v B v C

2. C v D v ¬A

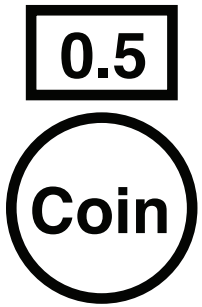
3. B v C v ¬D



Inference by stochastic simulation

Idea: replace **sum** over hidden-variable assignments with a **random sample**.

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
– 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 prior specified by bn

inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_D)$

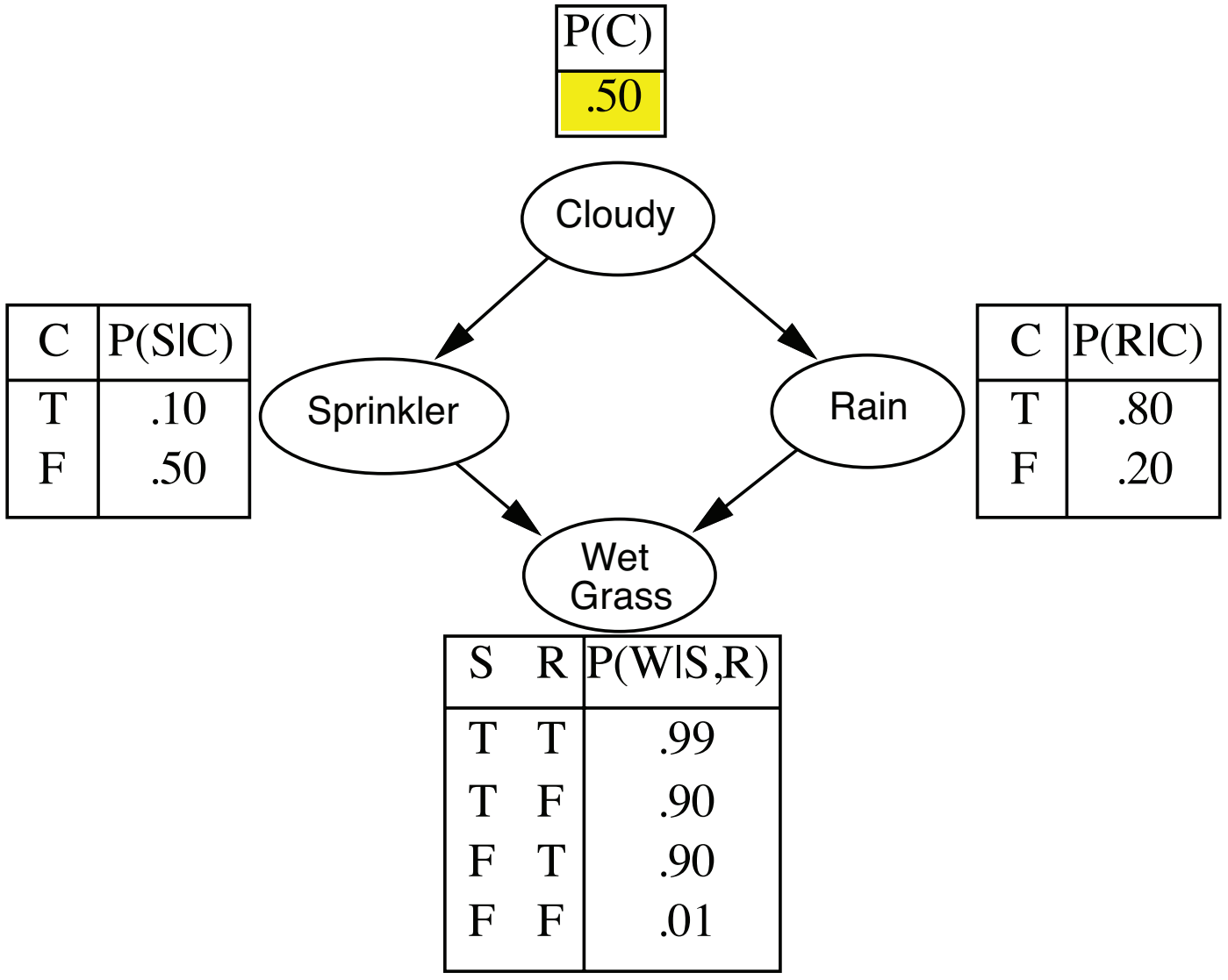
$\mathbf{x} \leftarrow$ an event with D elements

for $j = 1, \ldots, D$ **do**

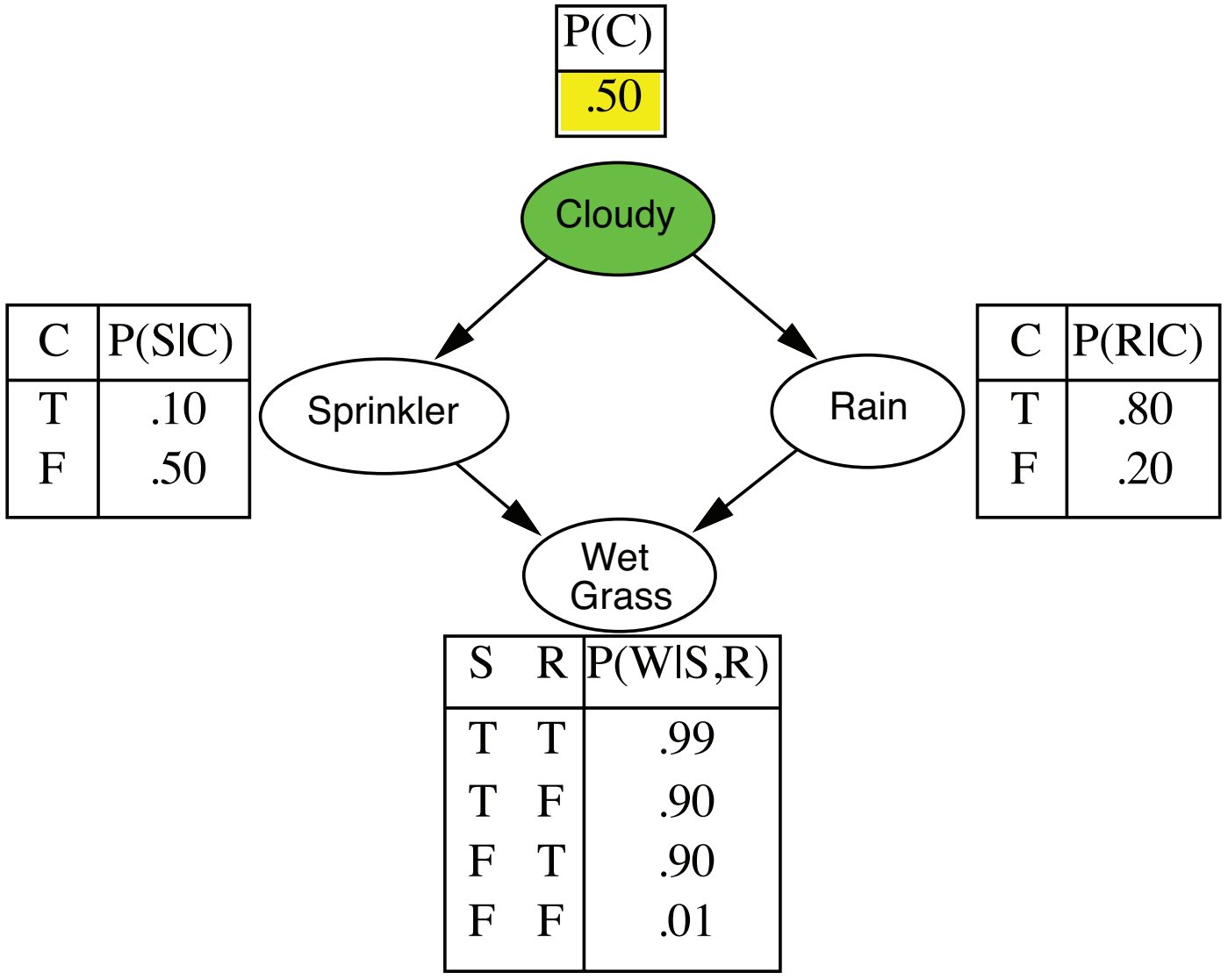
$\mathbf{x}[j] \leftarrow$ a random sample from $\mathbf{P}(X_j \mid$ values of $parents(X_j)$ in $\mathbf{x})$

return $\mathbf{x}$

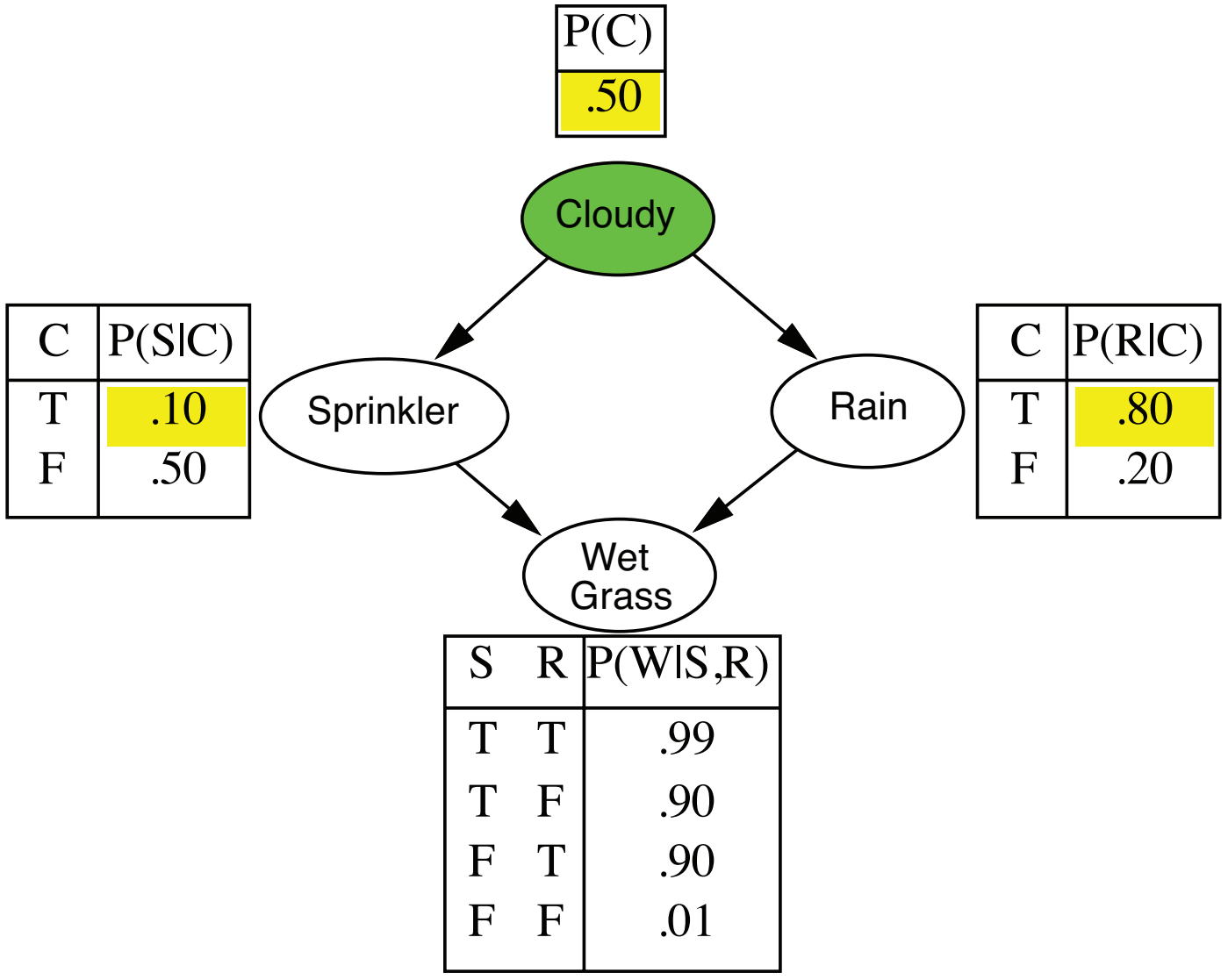
Example


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

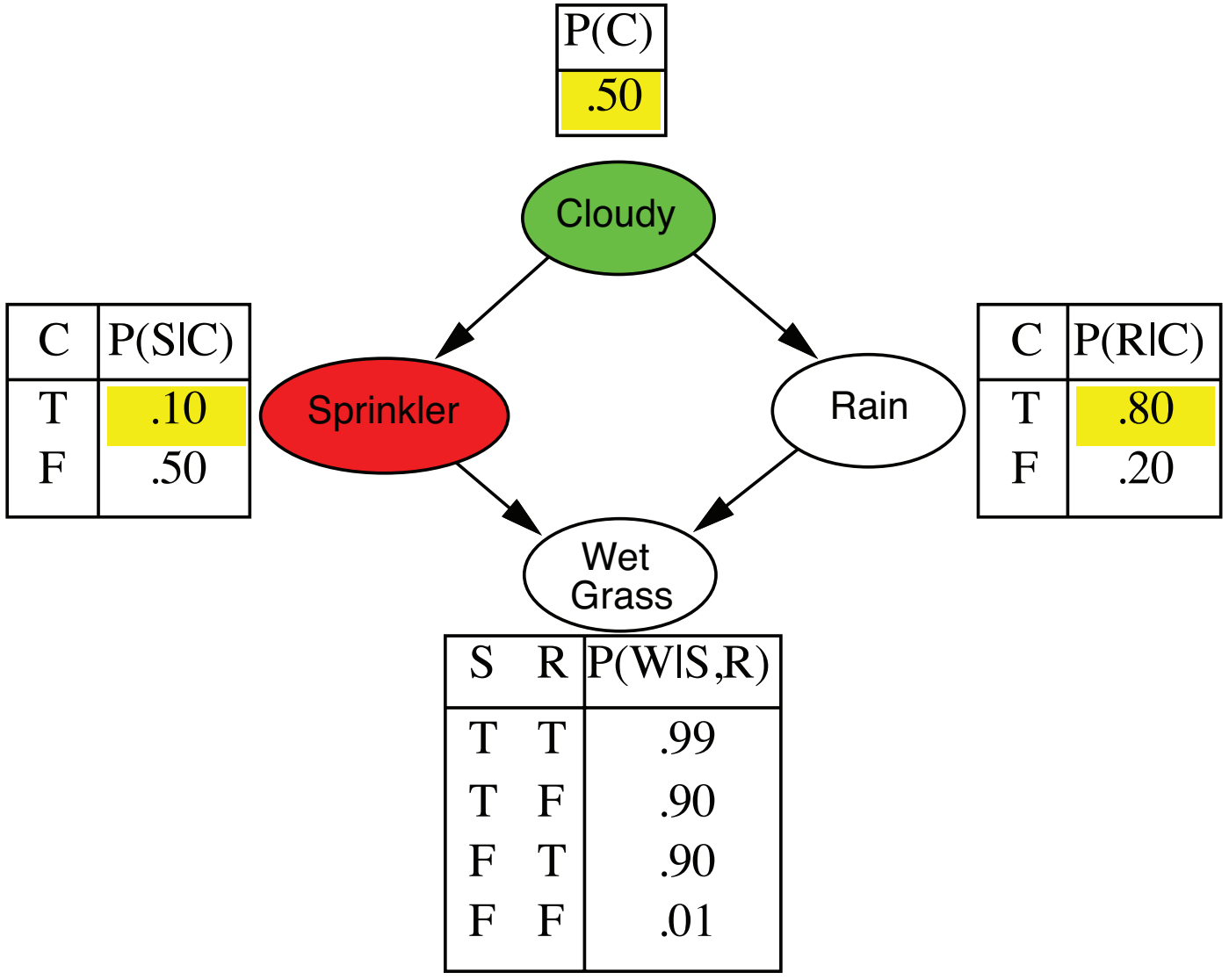
Example


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

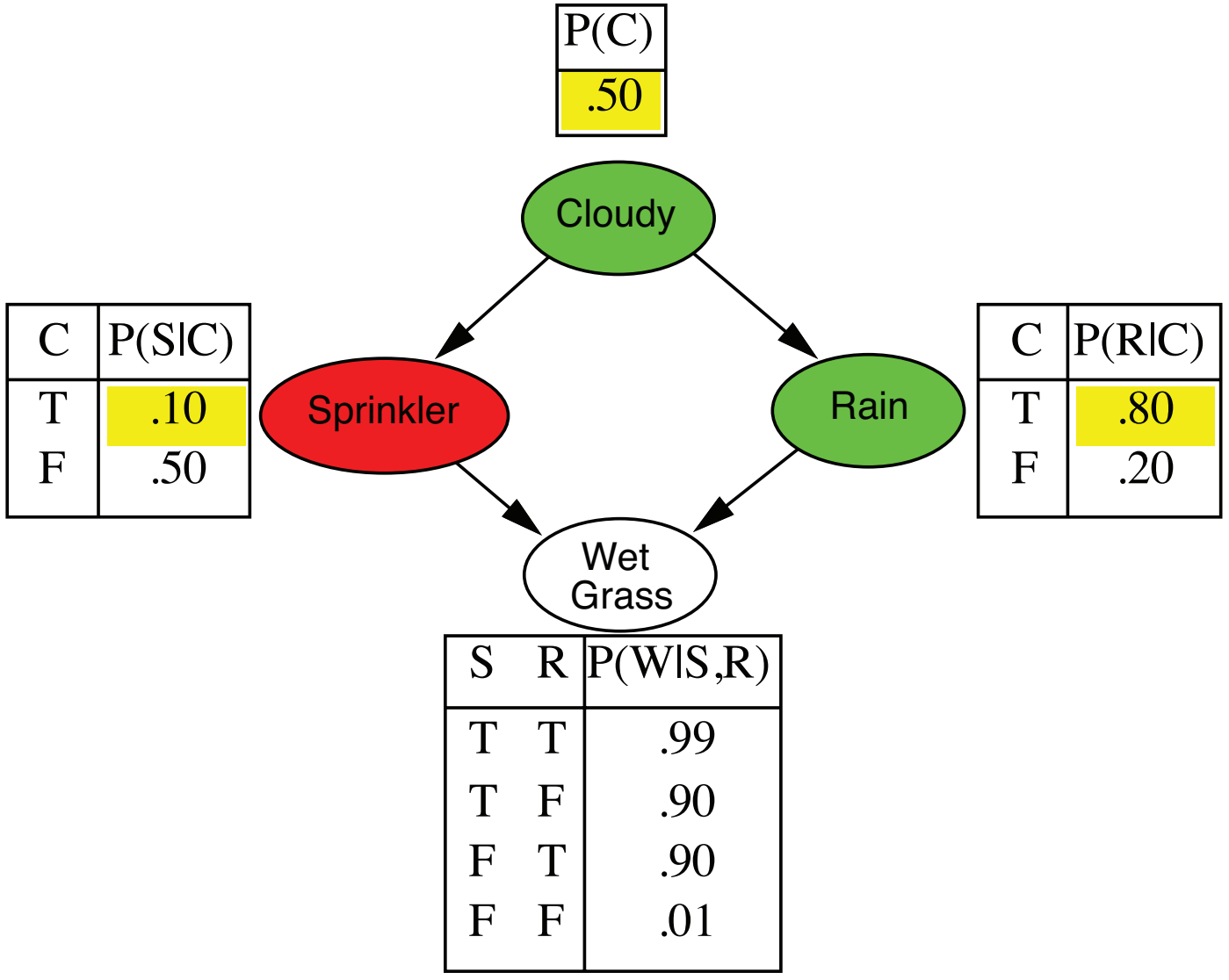
Example


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

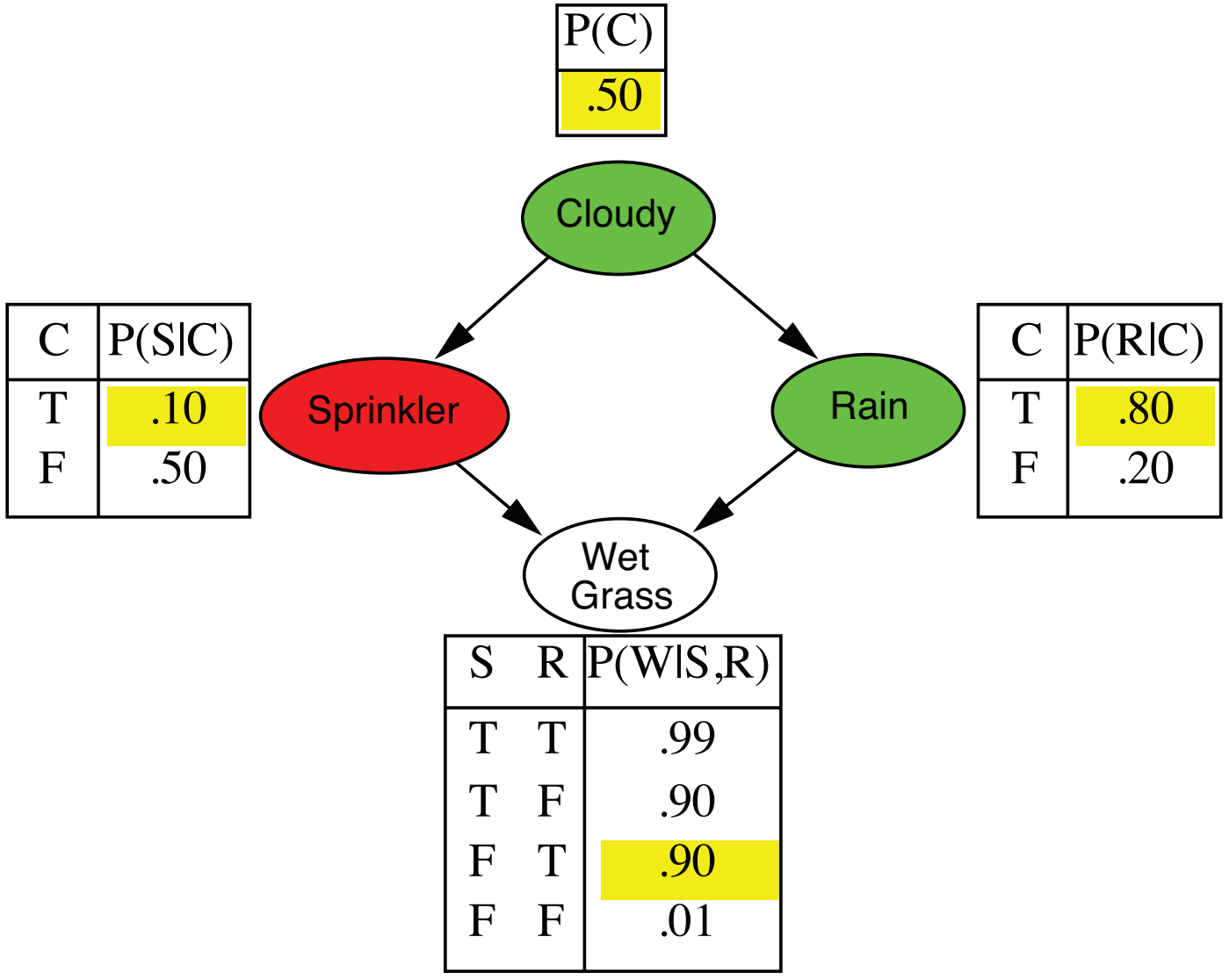
Example


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

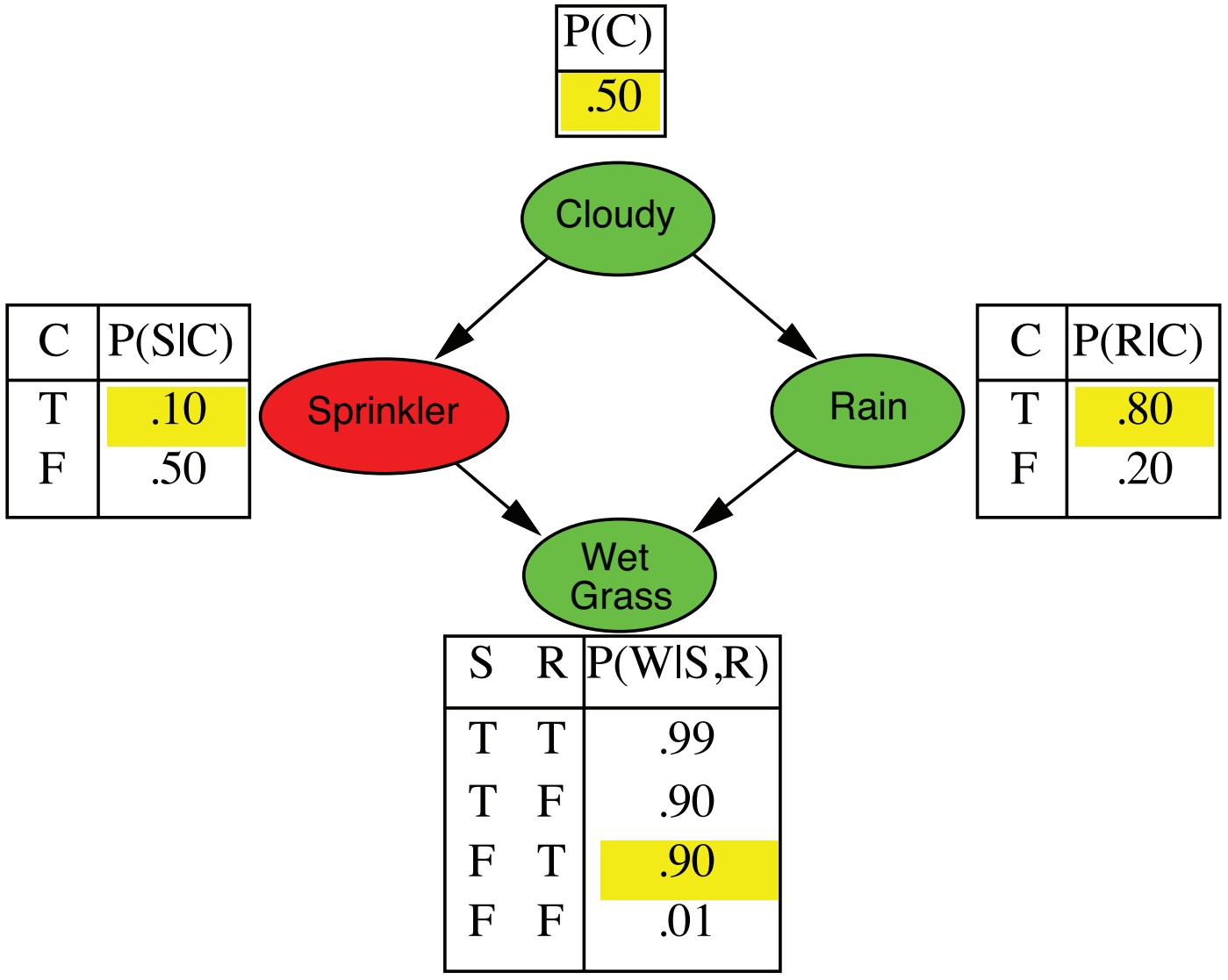
Example


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

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

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

Sampling from an empty network contd.

Probability that PriorSample generates a particular event

$$S_{PS}(x_1 \ldots x_D) = \prod_{j=1}^{D} P(x_j | parents(X_j)) = P(x_1 \ldots x_D)$$

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 $N_{PS}(x_1 \ldots x_D)$ be the number of samples generated for event $x_1, \ldots, x_D$

Then we have

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

That is, estimates derived from PriorSample are consistent

Shorthand: $\hat{P}(x_1, \ldots, x_D) \approx P(x_1 \ldots x_D)$

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 for each value of X, initially zero

  for $i = 1$ 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

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

Approximate inference using MCMC

General idea of Markov chain Monte Carlo

♢ Sample space Ω, probability $\pi(\omega)$ (e.g., posterior given $\mathbf{e}$)

♢ Would like to sample directly from $\pi(\omega)$, but it's hard

♢ Instead, wander around Ω randomly, collecting samples

♢ Random wandering is controlled by transition kernel $\phi(\omega \longrightarrow \omega')$
specifying the probability of moving to ω' from ω
(so the random state sequence $\omega_0, \omega_1, \ldots, \omega_t$ is a Markov chain)

♢ If ϕ is defined appropriately, the stationary distribution is $\pi(\omega)$
so that, after a while (mixing time) the collected samples are drawn from π

Gibbs sampling in Bayes nets

Markov chain state ω_t = current assignment $\mathbf{x}_t$ to all variables

Transition kernel: pick a variable X_j, sample it conditioned on all others

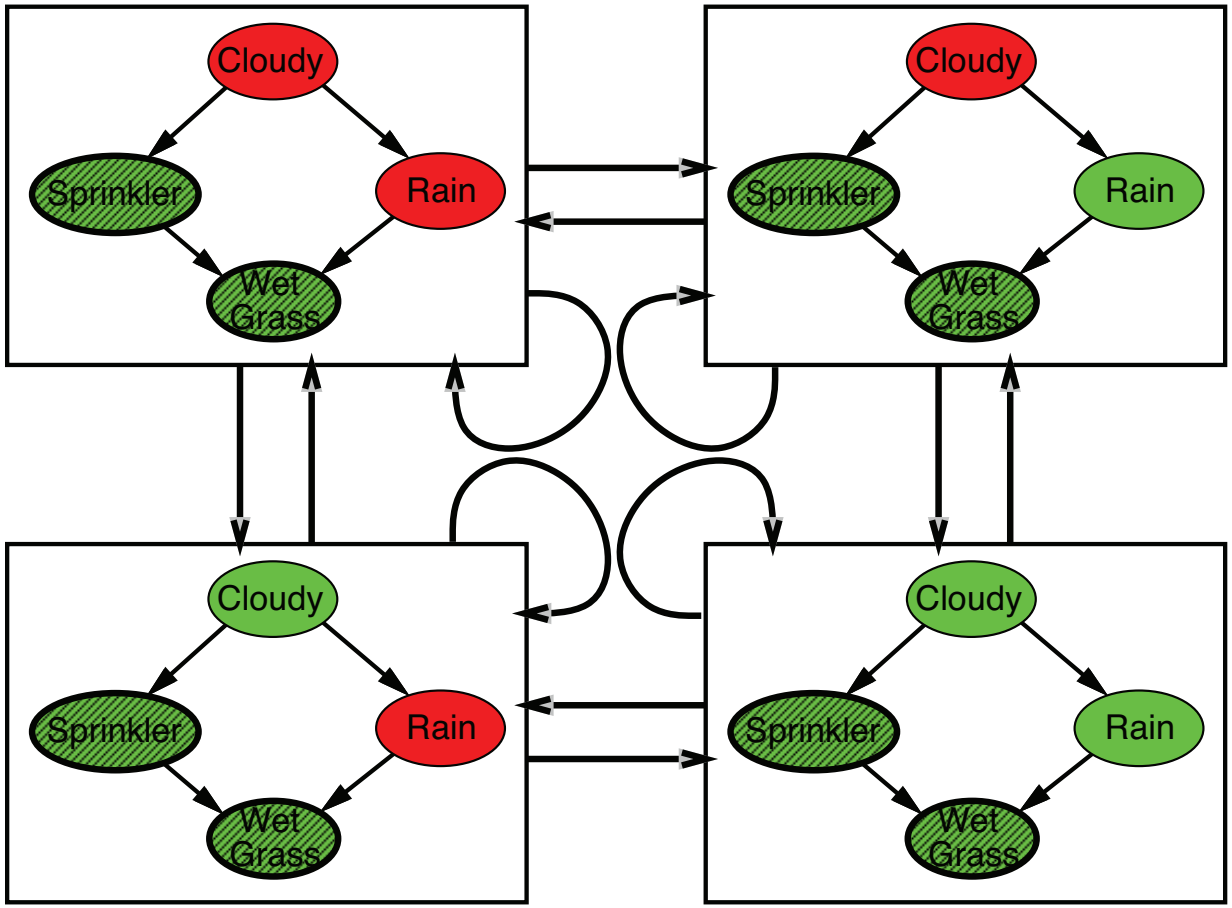
Markov blanket property: $P(X_j \mid \text{all other variables}) = P(X_j \mid mb(X_j))$
so generate next state by sampling a variable given its Markov blanket

function GIBBS-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
    $\mathbf{z}$, the current state of variables $\mathbf{Z}$, initially random

  for $i = 1$ to N **do**
    choose Z_j in $\mathbf{Z}$ uniformly at random
    set the value of Z_j in $\mathbf{z}$ by sampling from $\mathbf{P}(Z_j|mb(Z_j))$
    $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{z}$
  return NORMALIZE($\mathbf{N}$)

The Markov chain

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



MCMC example contd.

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.

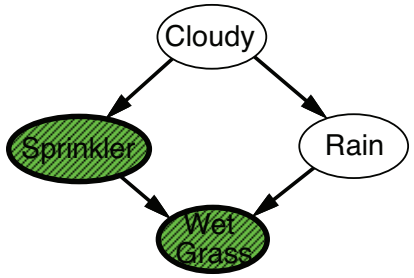
E.g., visit 100 states

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

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

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:

$$P(x'_j|mb(X_j)) = \alpha P(x'_j|parents(X_j))\Pi_{Z_\ell \in Children(X_j)} P(z_\ell|parents(Z_\ell))$$

E.g., $\phi(\neg cloudy, rain \rightarrow cloudy, rain)$
$= 0.5 \times \alpha P(cloudy)P(\ |\ cloudy)P(\ |\ cloudy) = 0.5 \times \alpha \times 0.5 \times 0.1 \times 0.8$
$= 0.5 \times \frac{0.040}{0.040+0.050} = 0.2222$

(Easy for discrete variables; continuous case requires mathematical analysis for each combination of distribution types.)

Easily implemented in message-passing parallel systems, brains

Can converge slowly, especially for near-deterministic models

Theory for Gibbs sampling

Theorem: stationary distribution for Gibbs transition kernel is $P(\mathbf{z} \mid \mathbf{e})$;
i.e., long-run fraction of time spent in each state is exactly proportional to its posterior probability

Proof sketch:

- The Gibbs transition kernel satisfies detailed balance for $P(\mathbf{z} \mid \mathbf{e})$
  i.e., for all $\mathbf{z}, \mathbf{z}'$ the "flow" from $\mathbf{z}$ to $\mathbf{z}'$ is the same as from $\mathbf{z}'$ to $\mathbf{z}$
- π is the unique stationary distribution for any ergodic transition kernel satisfying detailed balance for π

Detailed balance

Let $\pi_t(\mathbf{z})$ be the probability the chain is in state $\mathbf{z}$ at time t

Detailed balance condition: "outflow" = "inflow" for each pair of states:

$$\pi_t(\mathbf{z})\phi(\mathbf{z} \to \mathbf{z}') = \pi_t(\mathbf{z}')\phi(\mathbf{z}' \to \mathbf{z}) \qquad \text{for all } \mathbf{z},\ \mathbf{z}'$$

Detailed balance $\Rightarrow$ stationarity:

$$\begin{aligned} \pi_{t+1}(\mathbf{z}) = \textstyle\sum_{\mathbf{z}'} \pi_t(\mathbf{z}')\phi(\mathbf{z}' \to \mathbf{z}) &= \textstyle\sum_{\mathbf{z}'} \pi_t(\mathbf{z})\phi(\mathbf{z} \to \mathbf{z}') \\ &= \pi_t(\mathbf{z}) \textstyle\sum_{\mathbf{z}'} \phi(\mathbf{z} \to \mathbf{z}') \\ &= \pi_t(\mathbf{z}') \end{aligned}$$

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

Gibbs sampling transition kernel

Probability of choosing variable Z_j to sample is $1/(D - |\mathbf{E}|)$

Let $\bar{\mathbf{Z}}_j$ be all other nonevidence variables, i.e., $\mathbf{Z} - \{Z_j\}$
Current values are z_j and $\bar{\mathbf{z}}_j$; $\mathbf{e}$ is fixed; transition probability is given by

$$\phi(\mathbf{z} \rightarrow \mathbf{z}') = \phi(z_j, \bar{\mathbf{z}}_j \rightarrow z'_j, \bar{\mathbf{z}}_j) = P(z'_j | \bar{\mathbf{z}}_j, \mathbf{e})/(D - |\mathbf{E}|)$$

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

$$\begin{aligned}
\pi(\mathbf{z})\phi(\mathbf{z} \rightarrow \mathbf{z}') &= \frac{1}{D - |\mathbf{E}|} P(\mathbf{z} \mid \mathbf{e}) P(z'_j | \bar{\mathbf{z}}_j, \mathbf{e}) = \frac{1}{D - |\mathbf{E}|} P(z_j, \bar{\mathbf{z}}_j \mid \mathbf{e}) P(z'_j | \bar{\mathbf{z}}_j, \mathbf{e}) \\
&= \frac{1}{D - |\mathbf{E}|} P(z_j | \bar{\mathbf{z}}_j, \mathbf{e}) P(\bar{\mathbf{z}}_j \mid \mathbf{e}) P(z'_j | \bar{\mathbf{z}}_j, \mathbf{e}) \quad \text{(chain rule)} \\
&= \frac{1}{D - |\mathbf{E}|} P(z_j | \bar{\mathbf{z}}_j, \mathbf{e}) P(z'_j, \bar{\mathbf{z}}_j \mid \mathbf{e}) \quad \text{(chain rule backwards)} \\
&= \phi(\mathbf{z}' \rightarrow \mathbf{z}) \pi(\mathbf{z}') = \pi(\mathbf{z}') \phi(\mathbf{z}' \rightarrow \mathbf{z})
\end{aligned}$$

Summary (inference)

Exact inference:

- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by MCMC:

- Generally insensitive to topology
- Convergence can be very slow with probabilities close to 1 or 0
- Can handle arbitrary combinations of discrete and continuous variables

Parameter learning: Complete data

$\theta_{jk\ell} = P(X_j = k \mid Parents(X_j) = \ell)$

Let $x_j^{(i)}$ = value of X_j in example i; assume Boolean for simplicity

Log likelihood

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \sum_{j=1}^{D} \log P(x_j^{(i)} \mid parents(X_j)^{(i)} \\
&= \sum_{i=1}^{N} \sum_{j=1}^{D} \log \theta_{j1\ell^{(i)}}^{x_j^{(i)}} (1 - \theta_{j1\ell^{(i)}})^{1-x_j^{(i)}} \\
\frac{\partial L}{\partial \theta_{j1\ell}} &= \frac{N_{j1\ell}}{\theta_{j1\ell}} - \frac{N_{j0\ell}}{1 - \theta_{j1\ell}} = 0 \qquad \text{gives} \\
\theta_{j1\ell} &= \frac{N_{j1\ell}}{N_{j0\ell} + N_{j1\ell}} = \frac{N_{j1\ell}}{N_{j\ell}} \, .
\end{aligned}
$$

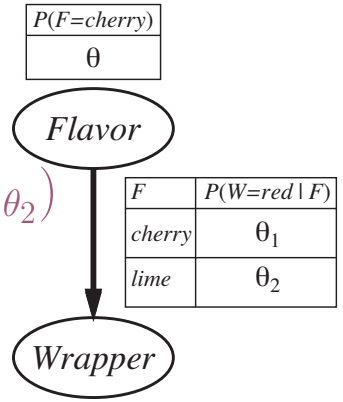
I.e., learning is completely decomposed; MLE = observed condition frequency

Example

Red/green wrapper depends probabilistically on flavor:

Likelihood for, e.g., cherry candy in green wrapper:

$$\begin{aligned} & P(F = cherry,\, W = green | h_{\theta,\theta_1,\theta_2}) \\ &= P(F = cherry | h_{\theta,\theta_1,\theta_2}) P(W = green | F = cherry, h_{\theta,\theta_1,\theta_2}) \\ &= \theta \cdot (1 - \theta_1) \end{aligned}$$



N candies, r_c red-wrapped cherry candies, etc.:

$$P(\mathbf{X} | h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^\ell \cdot \theta_1^{r_c} (1-\theta_1)^{g_c} \cdot \theta_2^{r_\ell} (1-\theta_2)^{g_\ell}$$

$$\begin{aligned} L &= [c \log \theta + \ell \log(1-\theta)] \\ &+ [r_c \log \theta_1 + g_c \log(1-\theta_1)] \\ &+ [r_\ell \log \theta_2 + g_\ell \log(1-\theta_2)] \end{aligned}$$

Example contd.

Derivatives of L contain only the relevant parameter:

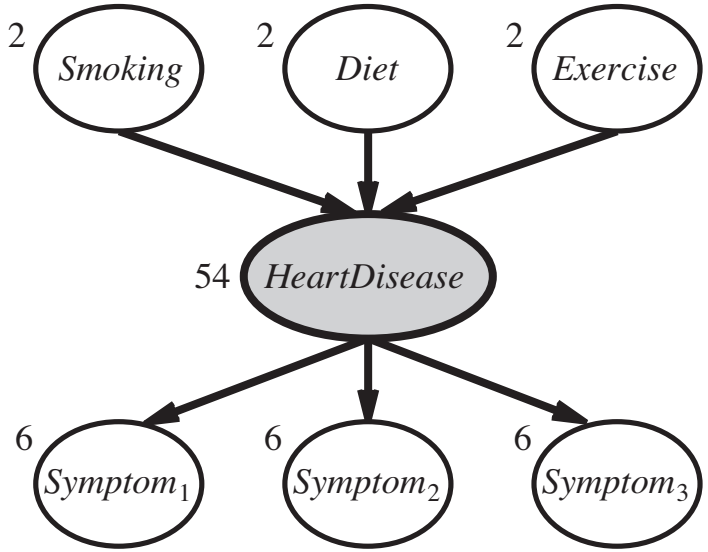
$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \qquad \Rightarrow \quad \theta = \frac{c}{c+\ell}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \qquad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c+g_c}$$

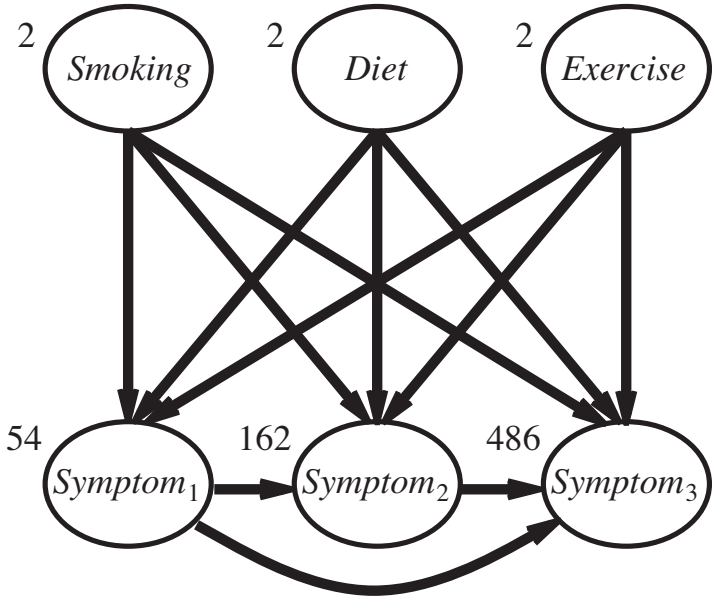
$$\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \qquad \Rightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell}$$

Hidden variables

Why learn models with hidden variables?



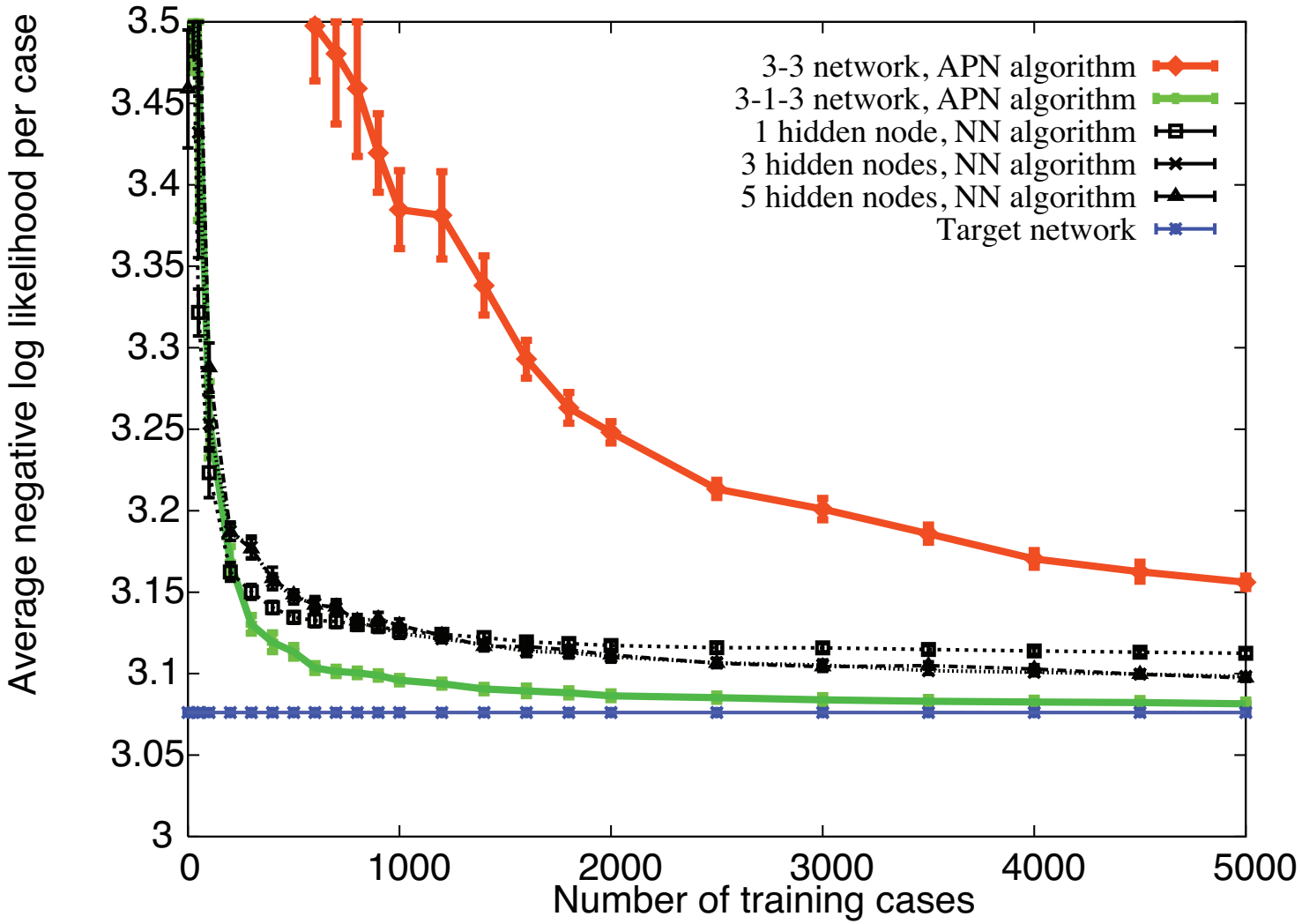
(a)



(b)

Hidden variables $\Rightarrow$ simplified structure, fewer parameters
$\Rightarrow$ faster learning

Learning with and without hidden variables



EM for Bayes nets

For $t = 0$ to ∞ (until convergence) do

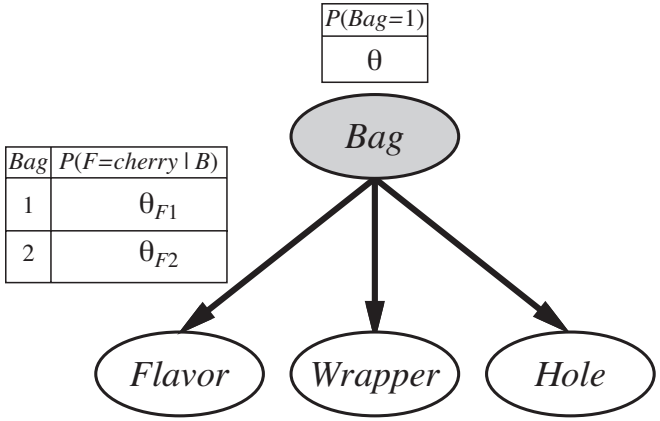
E step: Compute all $p_{ijk\ell} = P(X_j = k, Parents(X_j) = \ell \mid \mathbf{e}^{(i)}, \boldsymbol{\theta}^{(t)})$

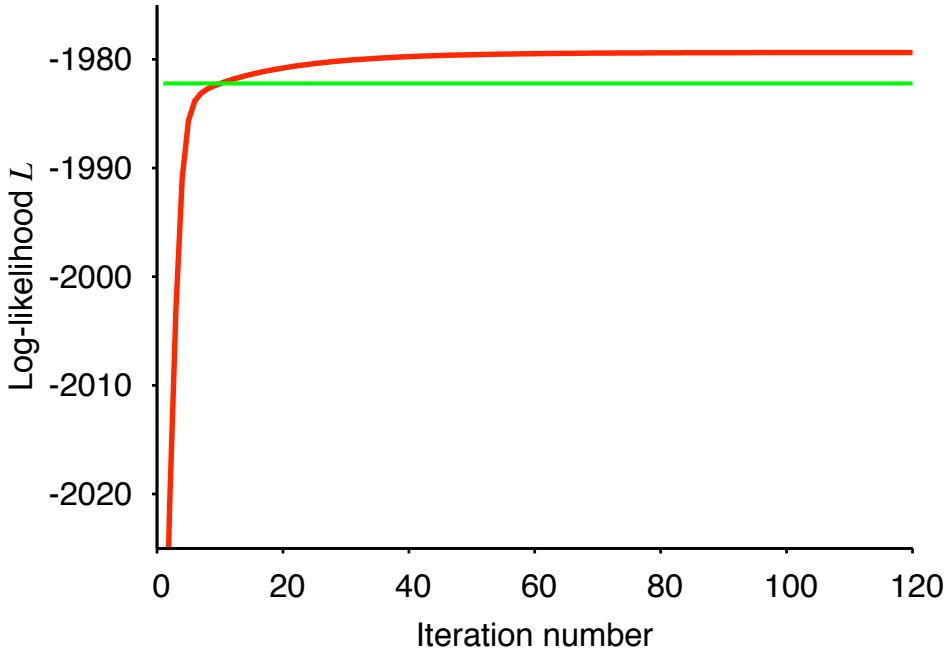
M step: $\theta_{jk\ell}^{(t+1)} = \dfrac{\hat{N}_{jk\ell}}{\sum_{k'} \hat{N}_{jk'\ell}} = \dfrac{\sum_i p_{ijk\ell}}{\sum_i \sum_{k'} p_{ijk'\ell}}$

E step can be any exact or approximate inference algorithm

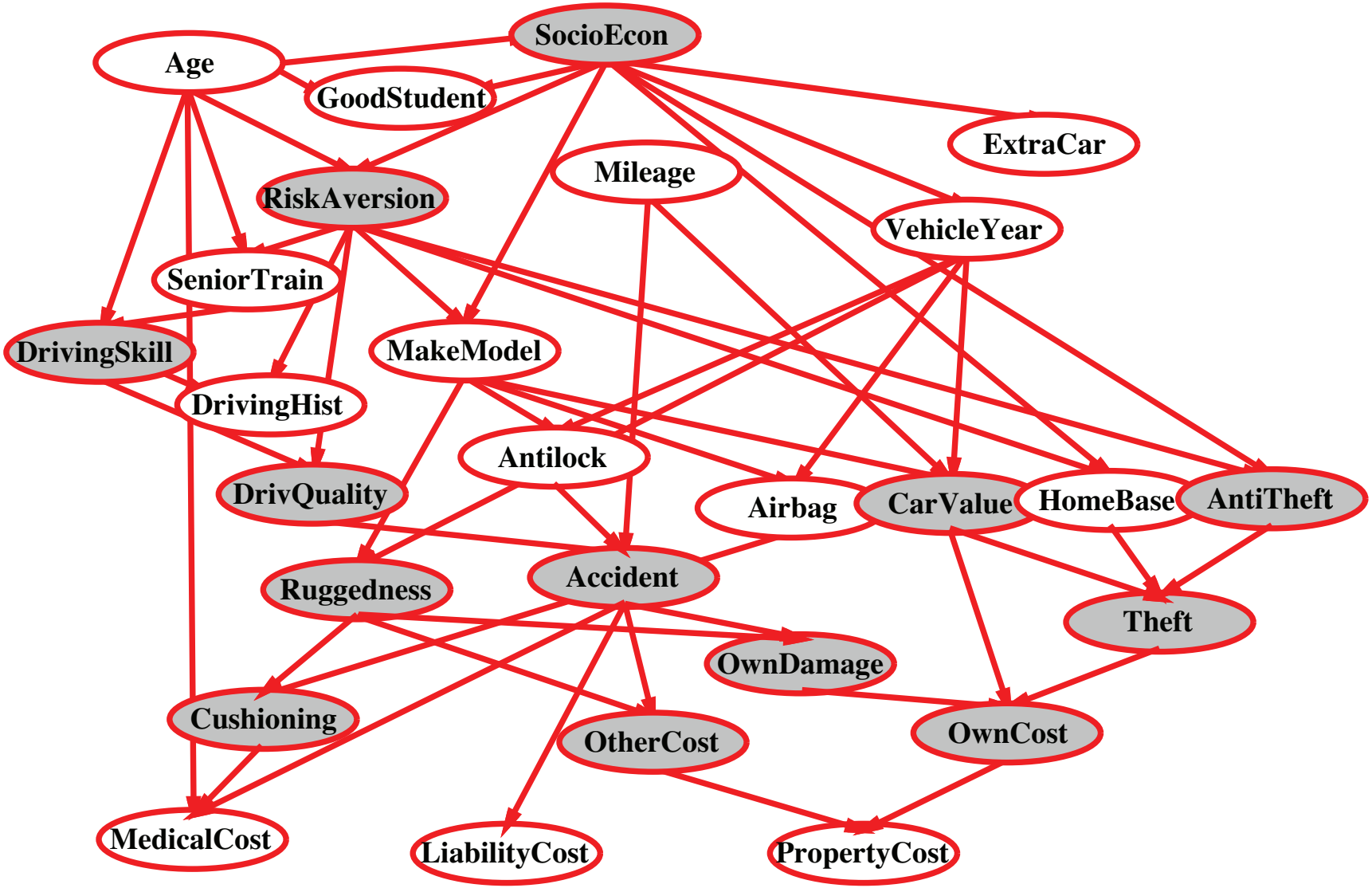
With MCMC, can treat each sample as a complete-data example

Candy example

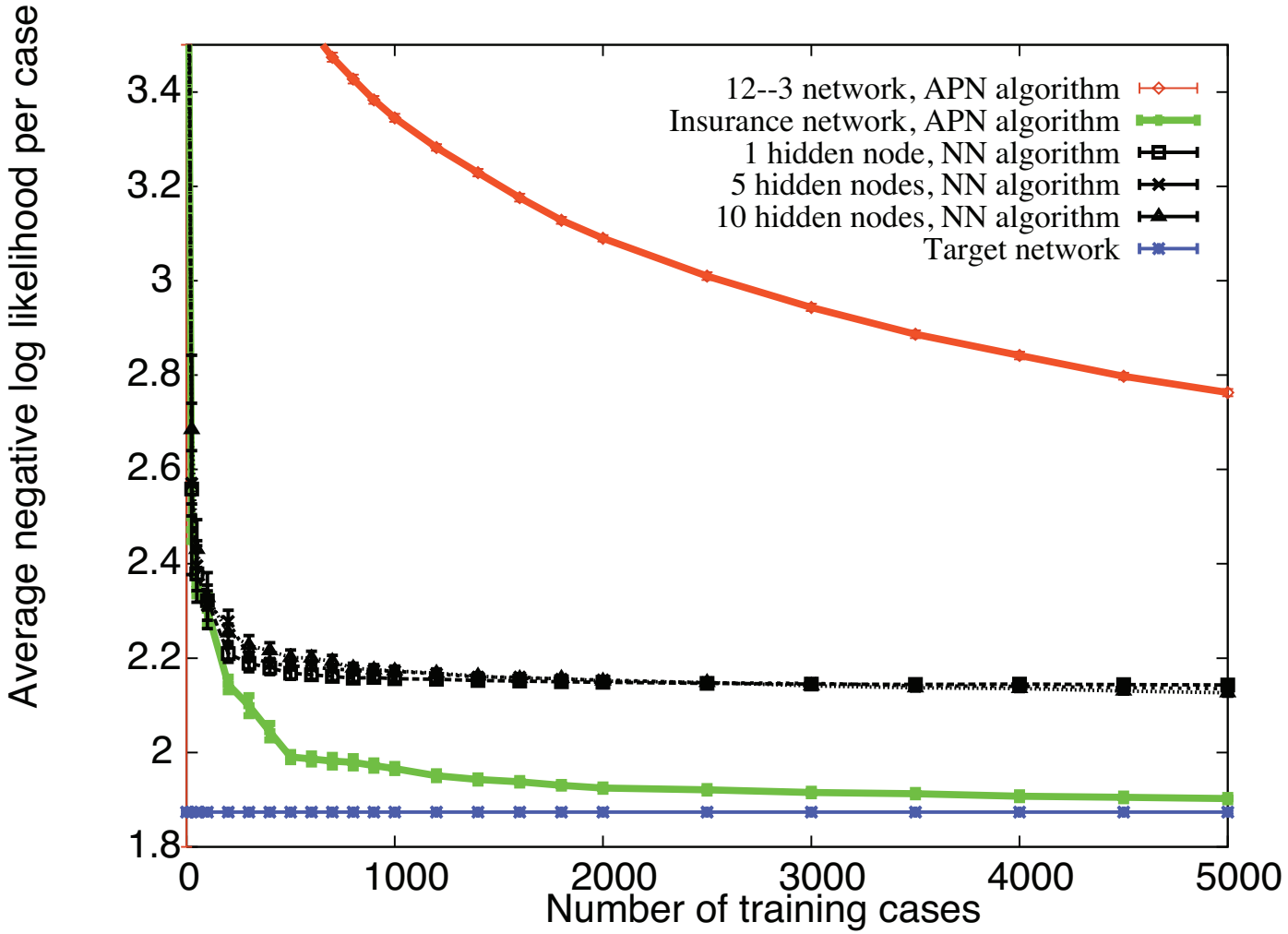




Example: Car insurance



Example: Car insurance contd.

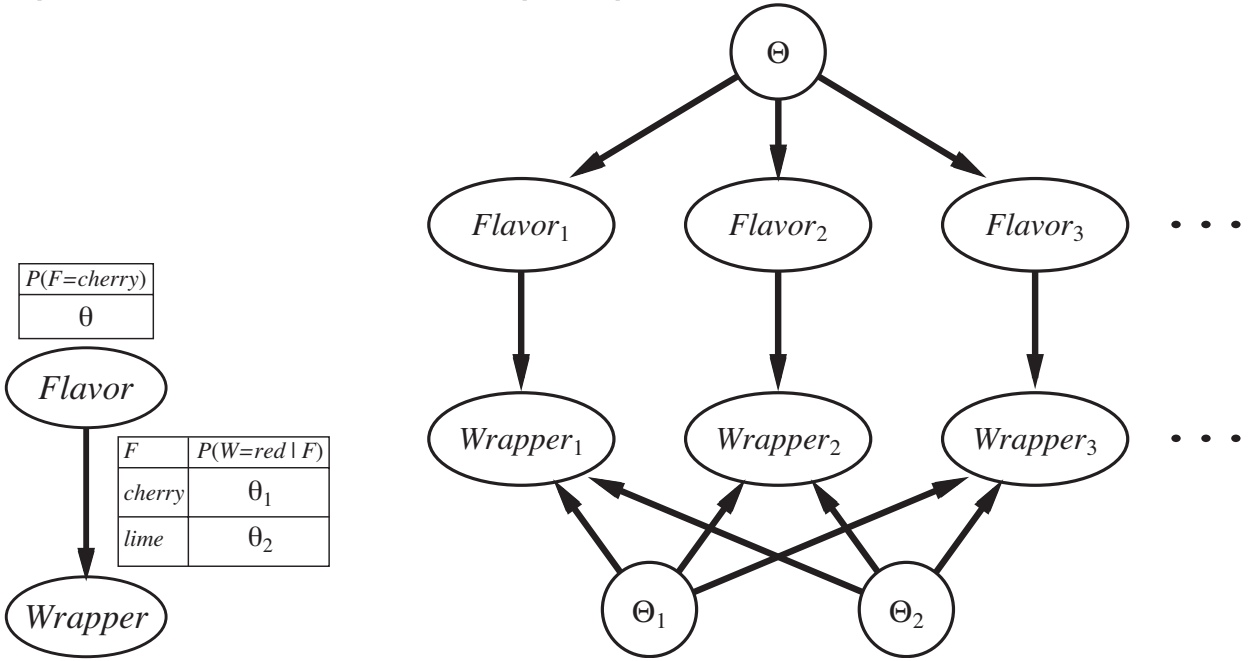


Bayesian learning in Bayes nets

Parameters become variables (*parents* of their previous owners):

$$P(\,Wrapper = red \mid Flavor = cherry, \Theta_1 = \theta_1, \Theta_2 = \theta_2) = \theta_1 \ .$$

network replicated for each example, parameters shared across all examples:



Bayesian learning contd.

Priors for parameter variables: Beta, Dirichlet, Gamma, Gaussian, etc.

With independent Beta or Dirichlet priors,
MAP EM learning = pseudocounts + expected counts:

$$\text{M step: } \theta_{jk\ell}^{(t+1)} = a_{jk\ell} + \frac{\hat{N}_{jk\ell}}{\sum_{k'} \hat{N}_{jk'\ell}}$$

Implemented in EM training mode for Hugin and other Bayes net packages

Summary (parameter learning)

Complete data: likelihood factorizes; each parameter $\theta_{jk\ell}$ learned separately from the observed counts for conditional frequency $N_{jk\ell}/N_{j\ell}$

Incomplete data: likelihood is a summation over all values of hidden variables; can apply EM by computing "expected counts"

Bayesian learning: parameters become variables in a replicated model with their own prior distributions defined by hyperparameters; then **Bayesian learning is just ordinary inference in the model**