COMS W4701: Artificial Intelligence

Lecture 11: Inference and Sampling

Tony Dear, Ph.D.
Department of Computer Science
School of Engineering and Applied Sciences

Today

- Exact inference in Bayes nets
- Variable elimination
- Direct sampling methods: Prior, rejection, importance
- MCMC methods: Gibbs sampling

Inference in Bayes Nets

- General task: Find the posterior distribution of a set of **query** variables X given a set of observed **evidence** e
- There may also be **hidden** variables Y interacting with X and E

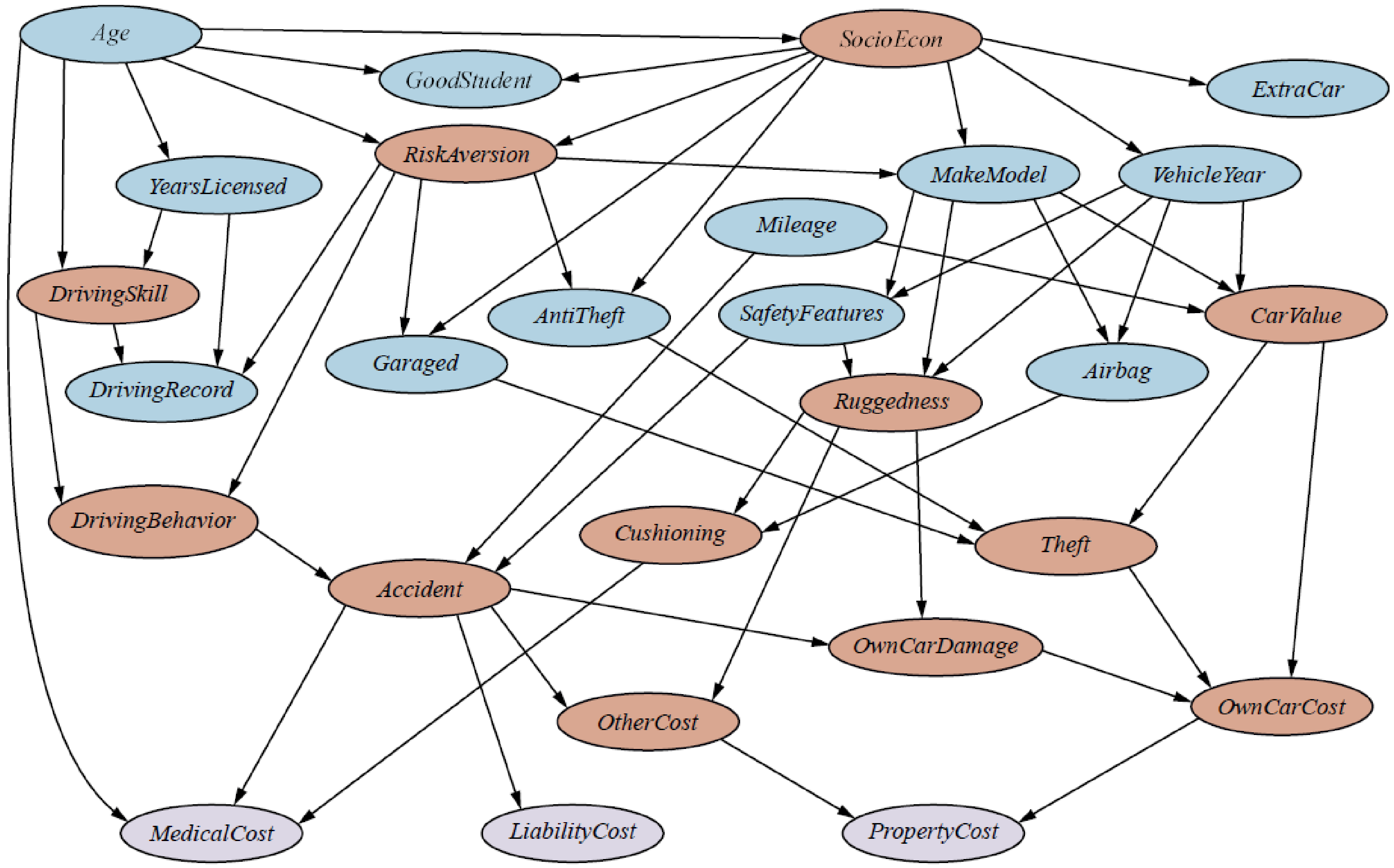
- General strategy: Construct joint distributions via chain rule and remove hidden variables via marginalization

$$P(\boldsymbol{X} \mid \boldsymbol{e}) = \alpha P(\boldsymbol{X}, \boldsymbol{e}) = \alpha \sum_{\boldsymbol{y}} P(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{e})$$

- Can be computationally heavy; conditional independences can help

Example: Car Insurance

- Queries: Costs of insurance (purple)
- Evidence: Entries requested by insurance company (blue)
- Hidden variables: Not observed, but may play role in determining insurance costs (red)



Example: Alarm Network

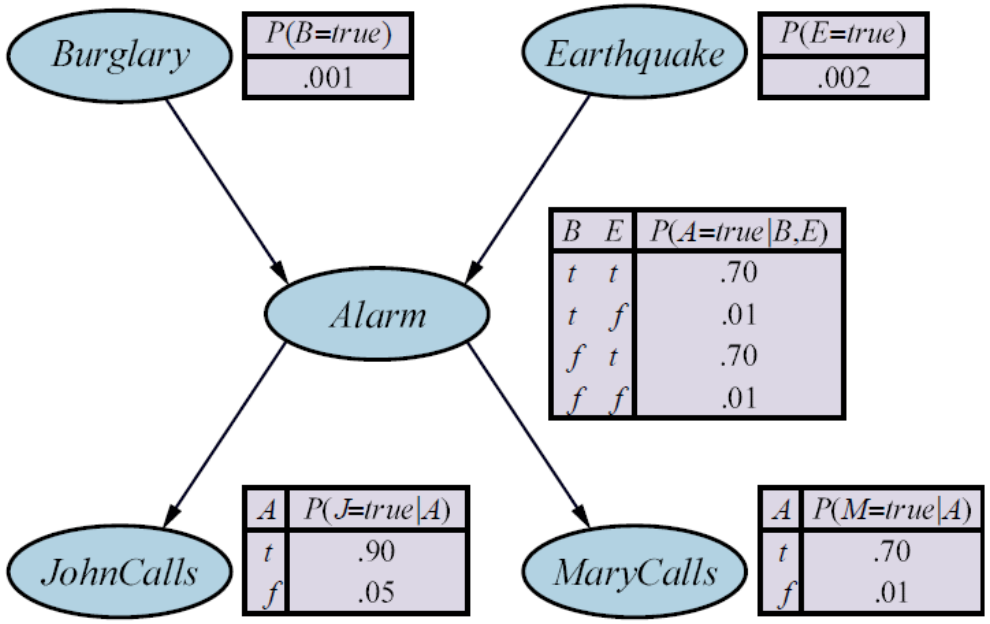
- Let's query individual probabilities first, instead of entire distributions
- **Query** variables $\boldsymbol{X}$; **evidence** variables $\boldsymbol{e}$; **hidden** variables $\boldsymbol{Y}$

$$P(+b,-e,+a) = P(+b)P(-e)P(+a|+b,-e)$$
$$= (.001)(.998)(.01) = 9.98 \times 10^{-6}$$

$$P(+j|-a,+e) = P(+j|-a) = 0.05$$

$$P(+j,+m|-a) = P(+j|-a)P(+m|-a)$$
$$= (0.05)(0.01) = 0.0005$$

$$P(+a) = \sum_{b,e} P(b,e,+a) = \sum_{b,e} P(b)P(e)P(+a|b,e)$$
$$= (.001)(.002)(.7) + (.001)(.998)(.01) + (.999)(.002)(.7) + (.999)(.998)(.01) = .01138$$



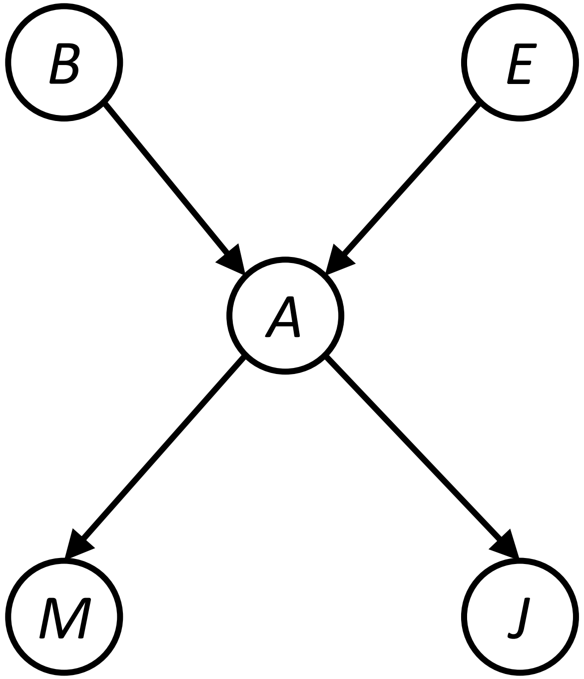
Example: Alarm Network

$$\sum_{e} P(b)P(e)P(a \mid b,e)$$

$$\sum_{b,e,a} P(b)P(e)P(a \mid b,e)P(m \mid a)P(j \mid a)$$

$$\frac{P(b)P(e)P(a \mid b,e)}{\sum_{b,e} P(b)P(e)P(a \mid b,e)}$$

$$\frac{\sum_{b,e} P(b)P(e)P(a \mid b,e)\, P(m \mid a)}{\sum_{b,e,m} P(b)P(e)P(a \mid b,e)P(m \mid a)}$$



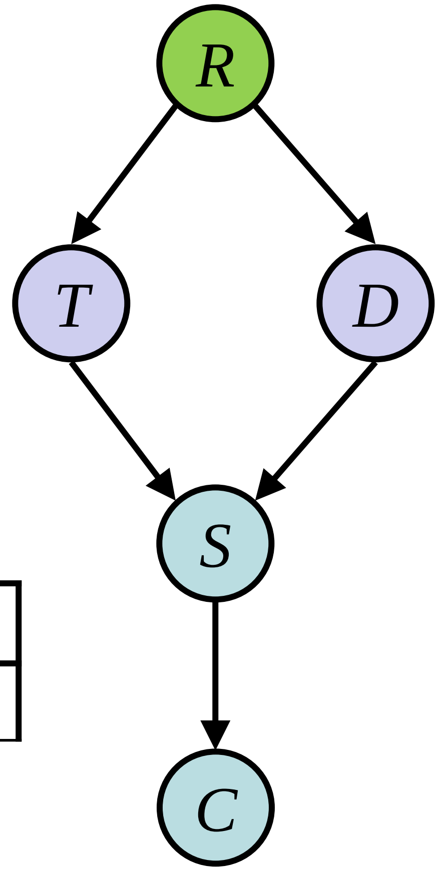
Querying Distributions

- The chain rule can be used to form an entire joint distribution all at once by *pointwise multiplying* matching rows

$$P(R|s,c) = P(R|s) \propto P(R,s) = \sum_{t,d} P(R,t,d,s)$$

Conditional independence

$$= \sum_{t,d} P(R)P(t|R)P(d|R)P(s|t,d)$$



R	P(R)
2 rows	

$\times$

R	T	P(T|R)
4 rows		

$\times$

R	D	P(D|R)
4 rows		

$\times$

T	D	P(s|T,D)
4 rows		

$\Rightarrow$

R	T	D	P(R,T,D,s)
8 rows			

$\sum_{t,d}$ $\Rightarrow$

R	P(R,s)
2 rows	

Example 1

$$P(R|+s,+c) \propto \sum_{t,d} P(R)P(t|R)P(d|R)P(+s|t,d)$$

$$= \overbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.1}^{P(R,+t,+d,+s,+c)} + \overbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * 0.4}^{P(R,+t,-d,+s,+c)}$$

$$+ \underbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.2}_{P(R,-t,+d,+s,+c)} + \underbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * 0.9}_{P(R,-t,-d,+s,+c)}$$

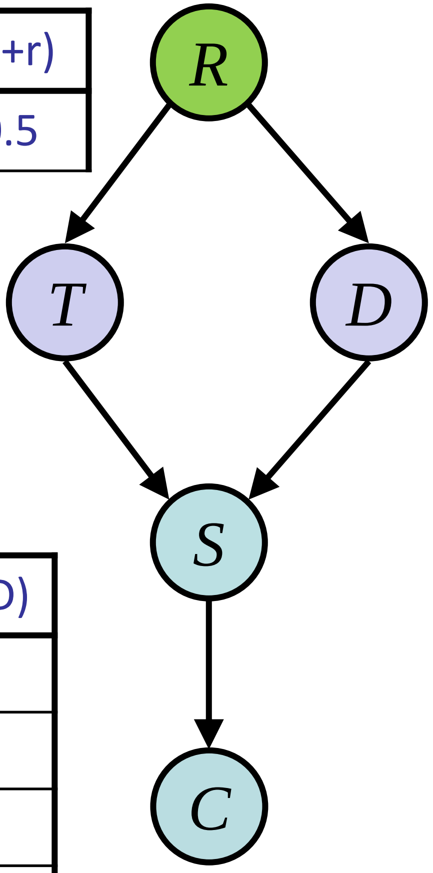
$$= \underbrace{\begin{pmatrix}0.128\\0.162\end{pmatrix}}_{P(R,+s,+c)} \propto \begin{pmatrix}0.441\\0.559\end{pmatrix} = P(R \mid +s,+c)$$

Prior to summing, we are building a distribution over R, T, D (3 variables)

P(+r)
0.5

R	P(+t|R)= P(+d|R)
+r	0.7
-r	0.6

T	D	P(+s|T,D)
+t	+d	0.1
+t	-d	0.4
-t	+d	0.2
-t	-d	0.9



S	P(+c|S)
+s	0.8
-s	0.3

Example 2

$$P(R \mid +c, +d) \propto \sum_{t,s} P(R)P(t \mid R)P(+d \mid R)P(s \mid t,+d)P(+c \mid s)$$

$$= \overbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.1 * 0.8}^{P(R,+t,+s,+d,+c)} + \overbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.9 * 0.3}^{P(R,+t,-s,+d,+c)}$$

$$+ \underbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.2 * 0.8}_{P(R,-t,+s,+d,+c)} + \underbrace{\begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.3\\0.4\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.8 * 0.3}_{P(R,-t,-s,+d,+c)}$$

$$= \underbrace{\begin{pmatrix}0.1278\\0.111\end{pmatrix}}_{P(R,+c,+d)} \propto \begin{pmatrix}0.535\\0.465\end{pmatrix} = P(R \mid +c, +d)$$

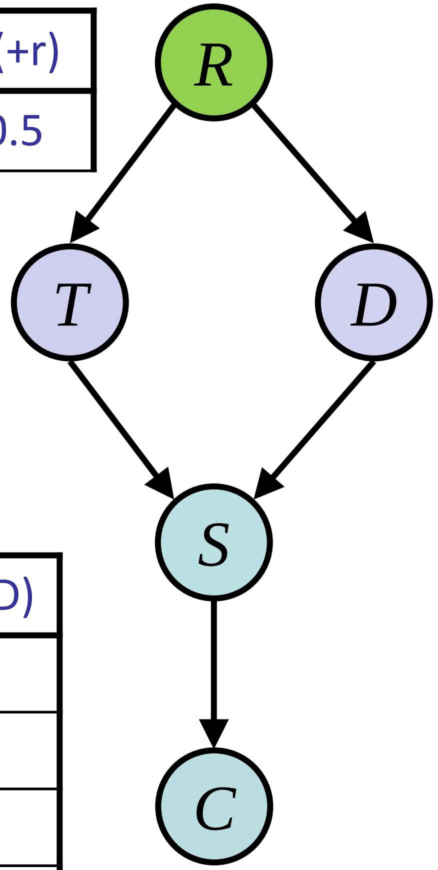
Prior to summing, we are building a distribution over R, T, S (3 variables)

P(+r)
0.5

R	P(+t|R)= P(+d|R)
+r	0.7
-r	0.6

T	D	P(+s|T,D)
+t	+d	0.1
+t	-d	0.4
-t	+d	0.2
-t	-d	0.9

S	P(+c|S)
+s	0.8
-s	0.3



Complexity of Inference

- Inference process involves building up a joint distribution encompassing all relevant variables, followed by marginalization down to the original query

- Worst case: Joint distribution over the entire Bayes net!

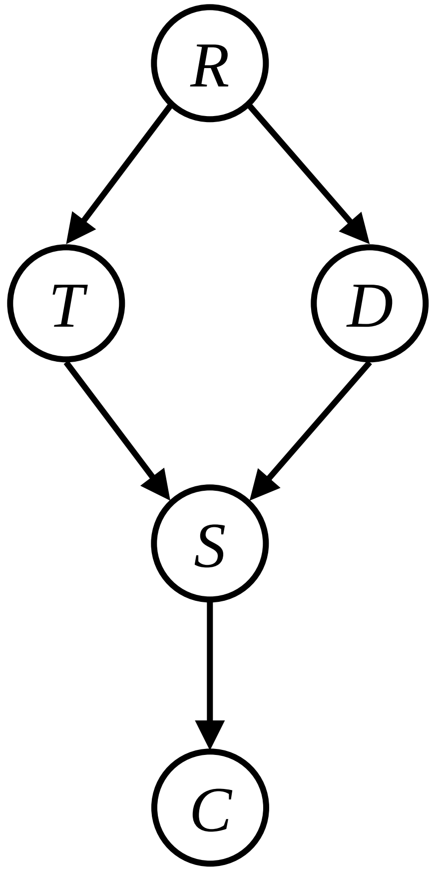
- Inference is NP-hard in general
- We can try to make process more efficient by marginalizing early and often
- Alternate between building up and summing out

Variable Elimination

- Idea: Alternate between building up and marginalizing

$$P(S|r) \propto P(r,S) = \sum_{t,d} P(r)P(t|r)P(d|r)P(S|t,d)$$
$$= P(r)\sum_t P(t|r)\sum_d P(d|r)P(S|t,d)$$

$$P(S|c) \propto P(S,c) = \sum_{r,t,d} P(r)P(t|r)P(d|r)P(S|t,d)P(c|S)$$
$$= P(c|S)\sum_r P(r)\sum_t P(t|r)\sum_d P(d|r)P(S|t,d)$$



Example: Variabl

the equation is the same: multiply and sum from top to bottom. if the variable is given, use the value (+d); if the variable is not given, use the sum of its all possible values (t).

$$P(R|+c,+d) \propto P(R)P(+d|R)\sum_t P(t|R)\sum_s P(s|t,+d)P(+c|s)$$

$P(R,+c,+d)$

$$\begin{pmatrix}0.12775\\0.111\end{pmatrix} = \begin{pmatrix}0.5\\0.5\end{pmatrix} * \begin{pmatrix}0.7\\0.6\end{pmatrix} * \begin{pmatrix}0.365\\0.37\end{pmatrix}$$

$P(+c,-t|R,+d)$

$$\begin{pmatrix}0.7\\0.6\end{pmatrix} * 0.35 + \begin{pmatrix}0.3\\0.4\end{pmatrix} * 0.4$$

$P(+c,+t|R,+d)$

$P(+c,-s|T,+d)$

$$\begin{pmatrix}0.1\\0.2\end{pmatrix} * 0.8 + \begin{pmatrix}0.9\\0.8\end{pmatrix} * 0.3$$

$P(+c,+s|T,+d)$

$\propto$

$$\begin{pmatrix}0.441\\0.559\end{pmatrix} \quad P(R|+c,+d)$$

$$\begin{pmatrix}0.365\\0.37\end{pmatrix}$$

$P(+c|R,+d)$

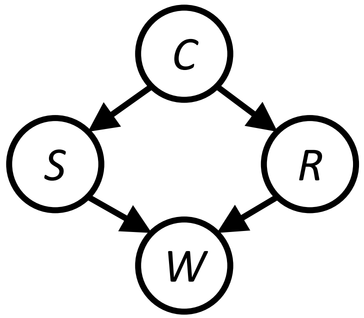
$$\begin{pmatrix}0.35\\0.4\end{pmatrix}$$

$P(+c|T,+d)$

Largest distribution at any point is over 2 variables

Approximate Inference: Sampling

- Exact inference becomes impossible when we have hundreds of variables
- **Monte Carlo**: Sampling from *known* probability distribution to estimate *unknown* distribution
- The more samples we get, the better the accuracy
- We can sample the variables in topological order according to each conditional probability table



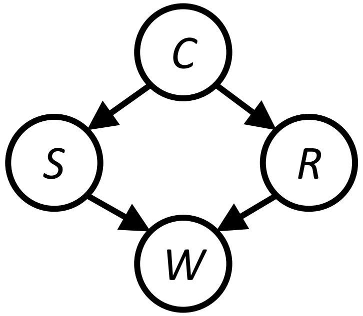
Ordering: C, S, R, W

1. Sample C using $P(C)$
2. Sample S using $P(S|c)$
3. Sample R using $P(R|c)$
4. Sample W using $P(W|s, r)$

Ordering C, R, S, W also works

Prior Sampling

- Inferences can be computed by counting samples corresponding to the query



Suppose we get 5 samples:

- (+c, -s, +r, +w)
- (+c, +s, +r, +w)
- (-c, +s, +r, -w)
- (+c, -s, +r, +w)
- (-c, -s, -r, +w)

- Prior sampling is **consistent**
- Probability that an event is generated equal to the true probability

$$\prod_{i=1}^{n} P(X_i|parents(X_i))$$

- Proportion of an event in samples approaches true probability in large-sample limit

$\hat{P}(R)$

+r	0.8
-r	0.2

$\hat{P}(C,W)$

+c	+w	0.6
	-w	0
-c	+w	0.2
	-w	0.2

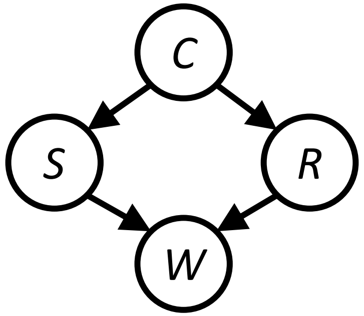
$\hat{P}(S|W)$

+w	+s	0.25
	-s	0.75
-w	+s	1
	-s	0

Rejection Sampling

- Counting samples can be done online instead of all at the end
- If query contains evidence, many samples may be irrelevant
- E.g., want $P(A|+b)$, all samples with $-b$ are useless to us!

- Idea: Discard irrelevant samples as they come and only count *consistent* ones



1. (+c, -s, +r, +w)
2. (+c, +s, +r, +w)
3. (-c, +s, +r, -w)
4. (+c, -s, +r, +w)
5. (-c, -s, -r, +w)

$P(C|+s)$

+c	0.5
-c	0.5

Reject 1, 4, 5

$P(R|-c)$

+r	0.5
-r	0.5

Reject 1, 2, 4

$P(S|+r,+w)$

+s	0.3
-s	0.7

Reject 3, 5

Rejection Sampling

initialize $C = 0$, vector of counts for values of query variable X
for $i = 1\!:N$ (number of samples requested)
    sample s via prior sampling from the Bayes net
    if s is consistent with evidence e:
        $C[j] \leftarrow C[j] + 1$ where $X = j$ in s
return normalize(C)

- Problem: Lots of potentially wasted work due to rejected samples!

- Fraction of accepted samples = probability of evidence
- With more evidence variables, fraction of consistent samples drops *exponentially*
- Need to wait a long time for rare evidence to occur

Likelihood Weighting

initialize $W = 0$, vector of counts for values of query variable X
for $i = 1{:}N$
    sample s *while* *fixing evidence* *variables e*
    $w \leftarrow \prod_{e_i} P(e_i|parents(E_i))$ in s
    $W[j] \leftarrow W[j] + w$ where $X = j$ in s
return normalize(C)

- Idea: **Fix** evidence variables to the values that we want

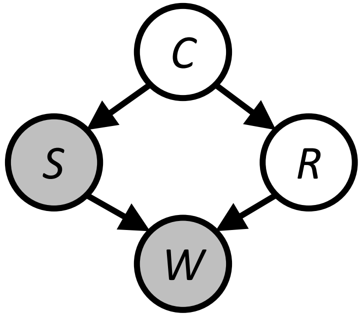
- Compensate by **weighting** each sample using probability of evidence given parents
- Weights are *cumulative products* for each evidence variable

Example: Likelihood Weighting

- We want $P(C, R \mid +s, +w)$
- Fix $+s$ and $+w$; sample all other variables

C	P(+s|C)
+c	0.1
-c	0.5

R	P(+w|+s,R)
+r	0.99
-r	0.90



- Each sample has a weight given by

$$P(+s|parents(S))P(+w|parents(W)) = P(+s|c)P(+w|+s,r)$$

- When counting, **sum up the weights of each sample**, and then normalize

- (+c, +s, +r, +w) $0.1 \times .99 = .099$
- (+c, +s, +r, +w) $0.1 \times .99 = .099$
- (+c, +s, -r, +w) $0.1 \times 0.9 = 0.09$
- (-c, +s, -r, +w) $0.5 \times 0.9 = 0.45$

$\hat{P}(C, R, +s, +w)$

+c	+r	0.198
	-r	0.09
-c	+r	0
	-r	0.45

$\propto$

$\hat{P}(C, R \mid +s, +w)$

+c	+r	0.268
	-r	0.122
-c	+r	0
	-r	0.610

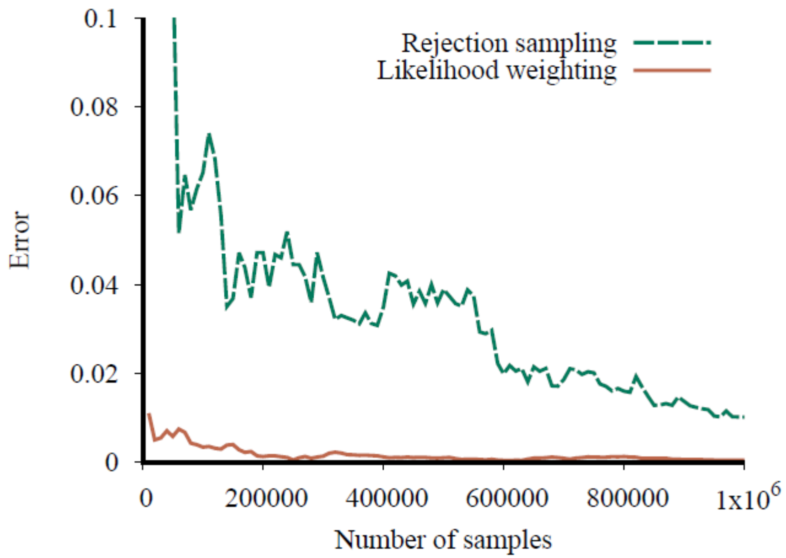
Importance Sampling*

- Likelihood weighting is an example of **importance sampling**: Use *sampling distribution* g and a correction factor to simulate sampling from *target distribution* f

- f is too hard to sample from, so use g instead
- For a sample x, correction factor is the *weight* $f(x)/g(x)$
- This works no matter what g we use!

- Much more efficient than rejection sampling

- Can get good accuracies with fewer samples



Likelihood Weighting Consistency*

- When performing inference in a Bayes net, the target distribution is $f(\boldsymbol{z}) = P(\boldsymbol{z}|\boldsymbol{e})$, where $\boldsymbol{Z}$ are nonevidence variables
- Sampling distribution is $Q(\boldsymbol{z}) = \prod_i P(z_i|parents(Z_i))$ (how we generate samples)

- Sample weight is as follows:

$$w(\boldsymbol{z}) = \frac{f(\boldsymbol{z})}{g(\boldsymbol{z})} = \frac{P(\boldsymbol{z}|\boldsymbol{e})}{Q(\boldsymbol{z})} \propto \frac{P(\boldsymbol{z},\boldsymbol{e})}{Q(\boldsymbol{z})} = \frac{\prod_i P(z_i|parents(Z_i)) \prod_j P(e_j|parents(E_j))}{\prod_i P(z_i|parents(Z_i))}$$

- Likelihood weights each sample by $\prod_j P(e_j|parents(E_j))$ and normalizes the counts at the end, so our estimates are consistent!

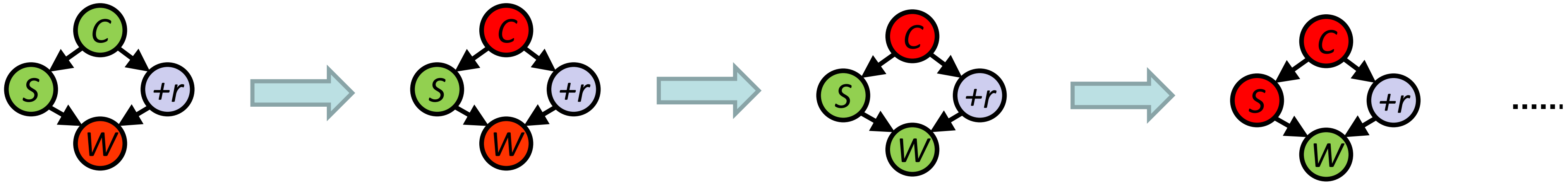
Sampling as Local Search

- Drawback of likelihood weighting: With lots of evidence, weights become small and tallies are dominated by a few samples with larger weights
- Another problem: Evidence variables "downstream" from their parents cannot influence the generation of their values
- How can we "condition" on both ancestors as well as descendants?
- Idea: Instead of generating each new sample from scratch, make small change to current one (just like local search!)

Gibbs Sampling

- **Gibbs sampling**: Fix evidence $\boldsymbol{e}$ and randomly sample non-evidence variables $\boldsymbol{Z}$. Then repeatedly choose and sample a variable Z_i conditioned on the *current* sample.

- Example: Evidence $+r$. Start (randomly) with $(+c, -w, +r)$ and sample S.



Sample from $P(S \mid +c, +r, -w)$ and obtain $+s$

Sample from $P(C \mid +s, +r, -w)$ and obtain $-c$

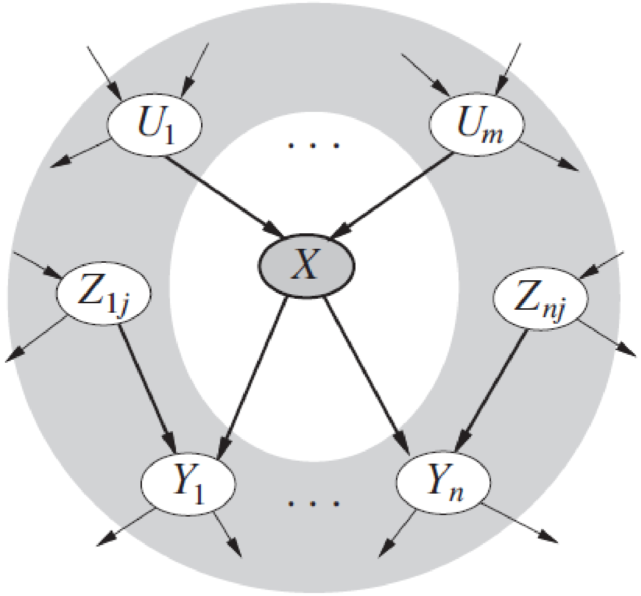
Sample from $P(W \mid -c, +s, +r)$ and obtain $+w$

Sample from $P(S \mid -c, +r, +w)$ and obtain $-s$

Markov Blanket

- Problem: How do we sample from $P(X_i \mid all\ other\ nodes\ in\ the\ BN)$?
- We actually only have to worry about a smaller subset of nodes

- A RV is conditionally independent of all other nodes given its **Markov blanket**: parents, children, children's parents



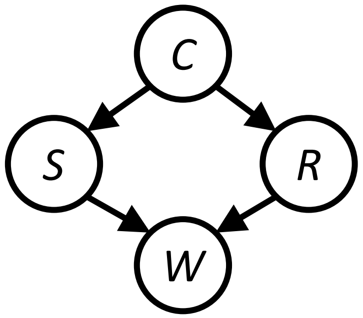
- Parents(X) *block* causal chains and common causes
- Children(X) *enable* common effects, but…
- Parents(Y_i) again block causal chains and common causes

Gibbs Sampling

- To sample from $P(X_i|mb(X_i))$, find the joint distribution and normalize
- All variables in $mb(X_i)$ are fixed, so easy to compute analytically
- Size is $O(|X_i|)$

$$P(x_i' \mid mb(X_i)) = \alpha \, P(x_i' \mid parents(X_i)) \times \prod_{Y_j \in Children(X_i)} P(y_j \mid parents(Y_j))$$

- Examples:



$$P(C \mid s, r, w) = P(C \mid s, r) \propto P(C)P(s|C)P(r|C)$$

$$P(S \mid c, r, w) \propto P(c)P(S|c)P(r|c)P(w|S,r) \propto P(S|c)P(w|S,r)$$

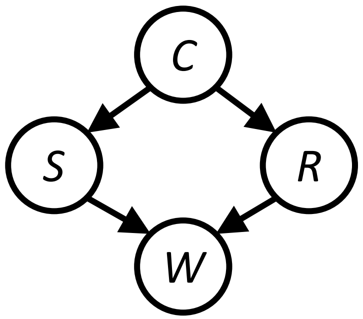
$$P(R \mid c, s, w) \propto P(c)P(s|c)P(R|c)P(w|s,R) \propto P(R|c)P(w|s,R)$$

$$P(W \mid c, s, r) = P(W \mid s, r)$$

Example: Gibbs Sampling

C	P(+s|C)
+c	0.1
-c	0.5

C	P(C)
+c	0.5



C	P(+r|C)
+c	0.8
-c	0.2

S	R	P(+w|S,R)
+s	+r	0.99
+s	-r	0.90
-s	+r	0.90
-s	-r	0

$$P(S \mid +c,+r,-w) \propto P(S|+c)P(-w|S,+r)$$

S	P(S,+c+r,-w)
+s	0.001
-s	0.09

=

S	P(S|+c)
+s	0.1
-s	0.9

×

S	P(-w|S,+r)
+s	0.01
-s	0.1

$$P(C \mid +s,+r,-w) \propto P(C)P(+s|C)P(+r|C)$$

C	P(C,+s,+r)
+c	0.04
-c	0.05

=

C	P(C)
+c	0.5
-c	0.5

×

C	P(+s|C)
+c	0.1
-c	0.5

×

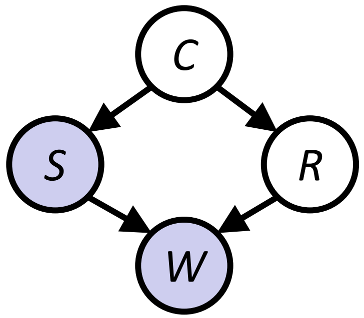
C	P(+r|C)
+c	0.8
-c	0.2

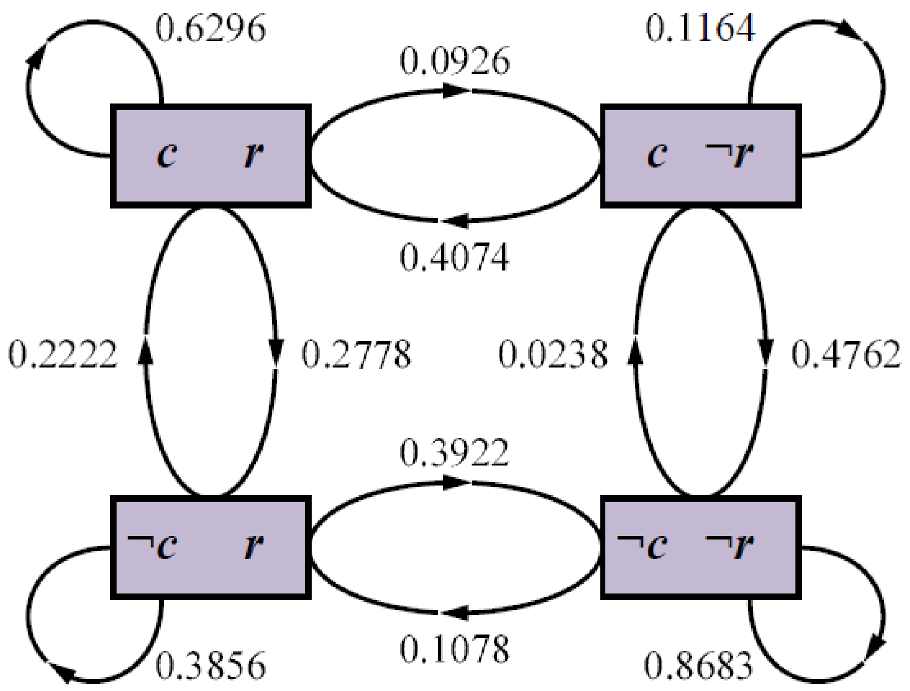
$$P(W \mid -c,+s,+r) = P(W \mid +s,+r)$$

W	P(W|+s,+r)
+w	0.99
-w	0.01

Markov Chain Monte Carlo

- Gibbs sampling is a **Markov chain Monte Carlo (MCMC)** method
- Traverses a Markov chain in the space of RVs
- Transition probabilities are the *likelihoods* of obtaining a sample given its predecessor
- The posterior distribution of the BN, conditioned on the evidence, is the *stationary distribution* of this Markov chain
- This is exactly what we want!





Gibbs Sampling Performance

- Each sampling step only depends on a node's immediate neighbors
- Good news: Independent of network size

- Generally performs better than likelihood weighting when evidence is "downstream"
- Information from evidence propagates outward in all directions

- However, convergence (**mixing rate**) is sensitive to the relationships among the RVs

- If certain states are hard to reach (low transition probabilities), then convergence can take a long time—same issues as in local search!

Metropolis-Hastings Sampling*

- Idea: As in local search algorithms like simulated annealing, sample locally *most* of the time, but occasionally allow for *jumps* to other part of the state space

- This can be specified by a **proposal distribution** $q(x'|\, x)$
- Ex: With small probability ε, generate sample x' using likelihood weighting (jump); otherwise, generate x' via Gibbs sampling

- But not all samples are good candidates, especially when jumping around
- Should only accept samples from the proposal according to their likelihood
- Otherwise, reject it (and stay put)

Acceptance Probability*

- Although we can use any arbitrary proposal $q(x'|x)$, let's restrict ourselves to *symmetric* distributions only: $q(x'|x) = q(x|x')$
- Equally likely to go from x' to x as it is to go from x to x', e.g. uniform distributions

- Suppose current sample is x; we sample x' according to proposal q and compute the **acceptance probability:**

$$a(x'|x) = \min\left(1, \frac{P(x', e)}{P(x, e)}\right)$$

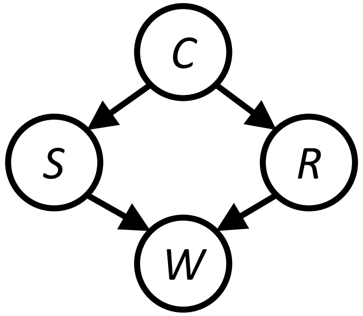
- If x' is more likely than x, accept it
- If x' is less likely than x, accept it with probability given by their likelihood ratios
- If x' is rejected, then new sample is x again (stay put)

Example: MH Sampling*

C	P(+s|C)
+c	0.1
-c	0.5

C	P(C)
+c	0.5

C	P(+r|C)
+c	0.8
-c	0.2



S	R	P(+w|S,R)
+s	+r	0.99
+s	-r	0.90
-s	+r	0.90
-s	-r	0

- Current sample: $x = (+c, +s, +r, -w)$
- Proposed sample: $x' = (+c, -s, +r, -w)$
- Acceptance ratio: $\frac{P(x')}{P(x)} = \frac{(0.5)(0.9)(0.8)(0.1)}{(0.5)(0.1)(0.8)(0.01)} = 90$
- x' is much more likely than x, so accept

- Current sample: $x = (+c, -s, +r, -w)$
- Proposed sample: $x' = (-c, +s, +r, -w)$
- Acceptance ratio: $\frac{P(x')}{P(x)} = \frac{(0.5)(0.5)(0.2)(0.01)}{(0.5)(0.9)(0.8)(0.1)} = 0.035$
- x' is very unlikely to occur; most likely reject

MH Properties*

- The form of the acceptance probability ensures that the underlying Markov chain has a stationary distribution (just like in Gibbs)

- Convergence is guaranteed for any choice of (symmetric) proposal distribution
- Gibbs is just special case of MH in which proposals are always accepted

- Computation of acceptance probability can be optimized
- Since samples usually only change locally, most of the terms can be reused
- Ex: $P(x')$ becomes $P(x)$ in the next acceptance probability

More Sampling Applications*

- Where else have we seen Monte Carlo methods before?
- Monte Carlo tree search for games
- Reinforcement learning
- Many problems in which solving for exact solutions is too hard!

Sampling in Games*

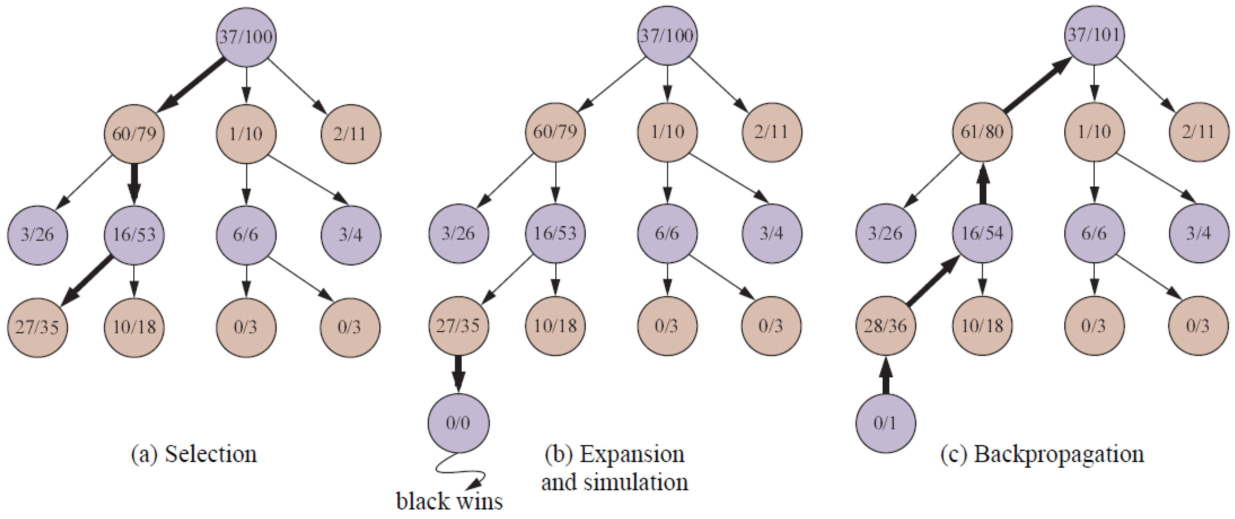
- Recall some of the challenges of performing search in game trees
- Huge branching factors, e.g. in games like Go
- Difficult to define good eval functions; most info at the end of the game

- **Monte Carlo tree search**: Instead of using an eval function, a state value is estimated through many simulated playouts from the state to the end of the game

- A *playout policy* may be learned from experience or follow some weak heuristics
- A *selection policy* chooses which states to simulate and play
- As in RL, need to balance exploitation and exploration, e.g. using UCB

Monte Carlo Tree Search*

- Given a current game tree, perform the following steps:
- 1. Follow selection policy down to a leaf of the tree
- 2. Expand the leaf and simulate a playout to the end of the game
- 3. Record and backpropagate the result to all nodes in the simulated path



- After a number of iterations, choose the move with the most playouts

Sampling in Reinforcement Learning*

- We have already discussed using Monte Carlo methods in RL for both prediction (estimate values for a fixed policy) and control (find an optimal policy)

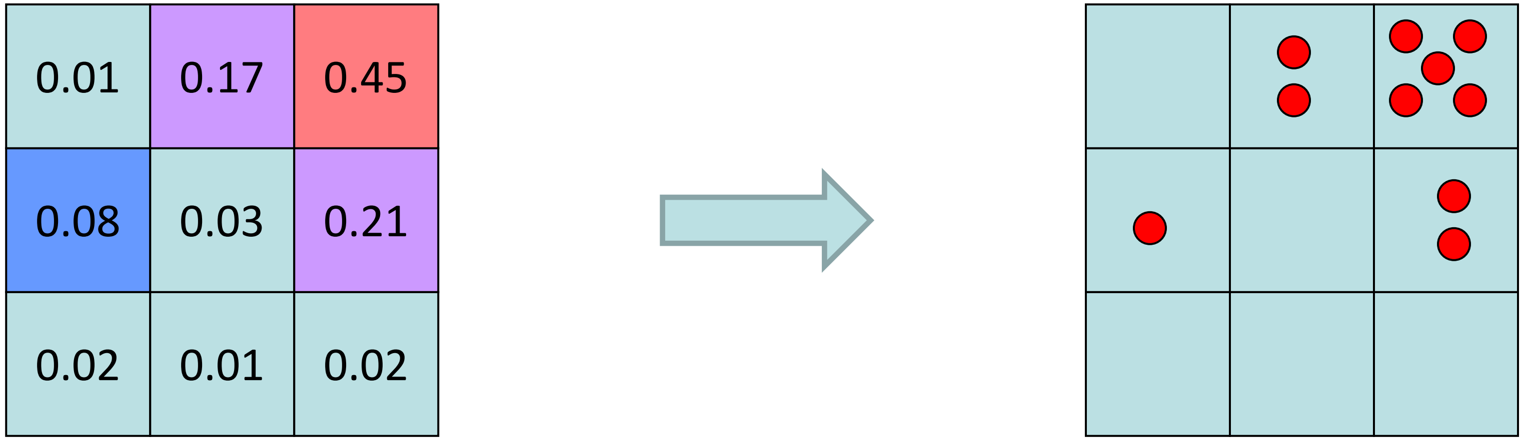
- Control example: Generate many episodes in the MDP following a $\varepsilon-$greedy policy
- If we do not decrease ε to 0, we may not actually learn a purely greedy optimal policy!

- We have a different behavior ($\varepsilon-$greedy) and target (greedy) policy
- How to address this discrepancy?
- Use importance sampling: Weight each sample by the relative likelihoods according to the target and behavior policies

Sampling in HMMs*

- We know how to perform exact state estimation for a general HMM
- This may be computationally intractable for large problems, e.g. robot localization

- **Particle filtering**: Instead of keeping track of exact distributions of belief states, keep track of a number of particles (samples) that estimate the belief state
- Each particle evolves according to transition and sensor models

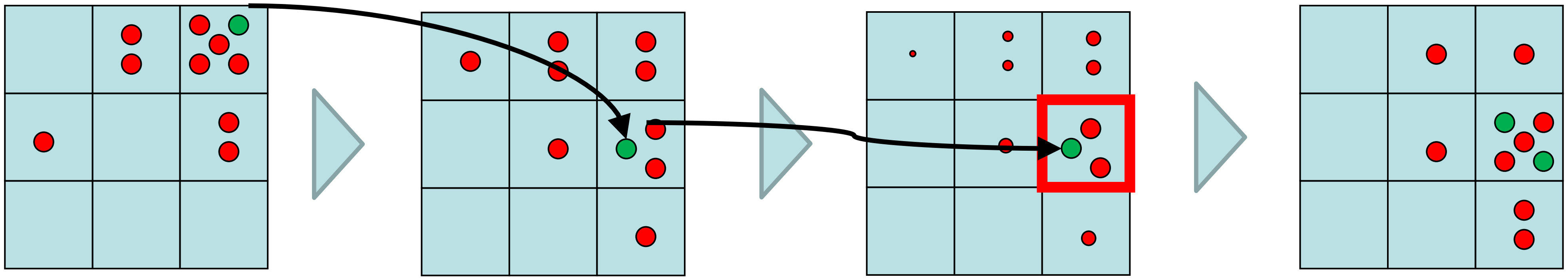


Particle Filter*

sample $p(\boldsymbol{x}_t|\boldsymbol{x}^j_{t-1})$

weight $w_t^j = p(\boldsymbol{z}_t|\boldsymbol{x}_t^j)$

Resample (renormalize):



Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,2)	(2,2) w=.4	(3,2)

Summary

- Performing inference in Bayes nets involves querying distributions given evidence
- Computationally heavy in large networks with many hidden variables

- Monte Carlo sampling allows us to *estimate* probability distributions
- Direct sampling methods draw samples independently
- Reject inconsistent samples or enforce consistency through likelihood weighting

- MCMC methods (e.g., Gibbs) treat sampling as local search
- Transitions follow a Markov chain; stationary distribution gives us posterior