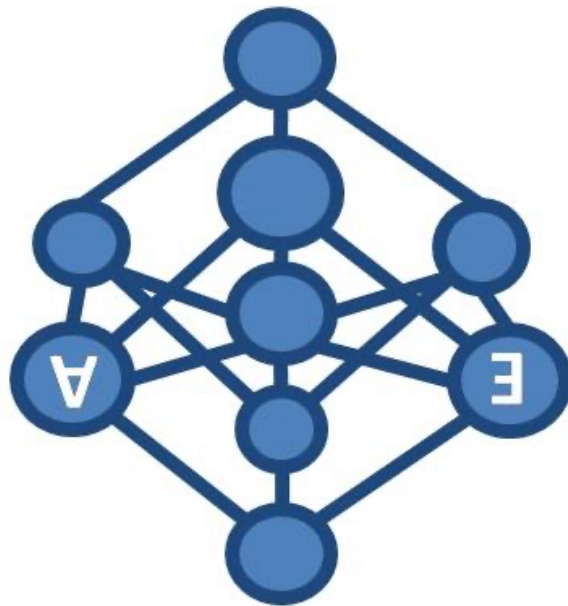


# Probabilistic Graphical Models

## Approximate Inference



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



\*Thanks to Carlos Guestrin,  
Pedro Domingos and many  
others for making their slides  
publically available



# Today

- From class, we know that computing the aposteriori belief of a variable in a general BN is NP-hard
- In particular, exact inference for DBNs is intractable even for simple cases
- Solution: approximate inference
  - Loopy Belief Propagation
  - Sampling

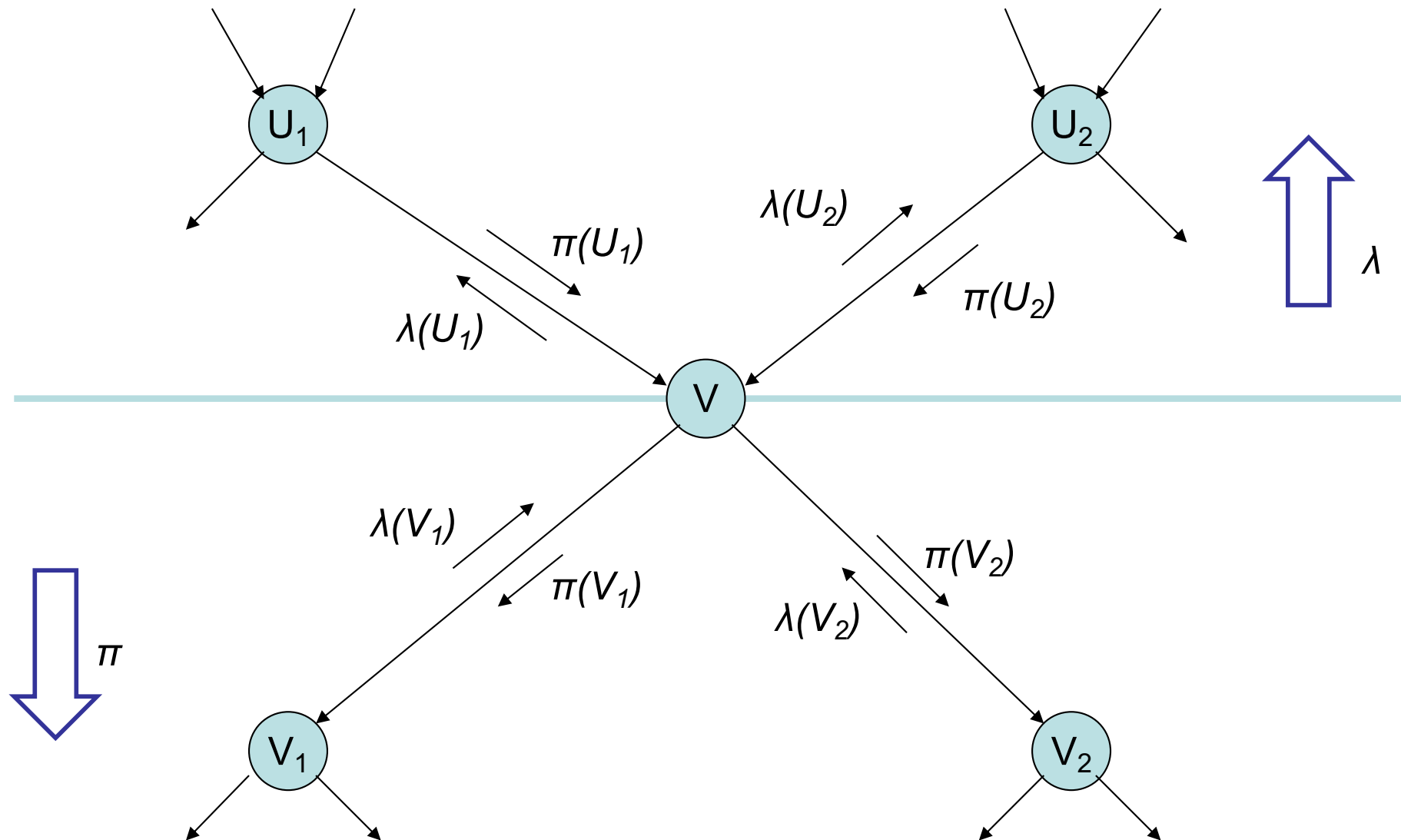


# Pearl's belief propagation

- We have the evidence  $E$  and want to compute all the single node marginals  $P(V_i|E)$  for all  $i$  using
- Local computation for one node  $V$  desired
- Information flows through the links of  $G$ 
  - flows as messages of two types –  $\lambda$  and  $\pi$
- $V$  splits network into two *disjoint* parts
  - **Strong independence assumptions induced – crucial!**
- Denote  $E_V^+$  the part of evidence accessible through the parents of  $V$  (*causal*)
  - passed downward in  $\pi$  messages
- Analogously, let  $E_V^-$  be the *diagnostic* evidence
  - passed upwards in  $\lambda$  messages

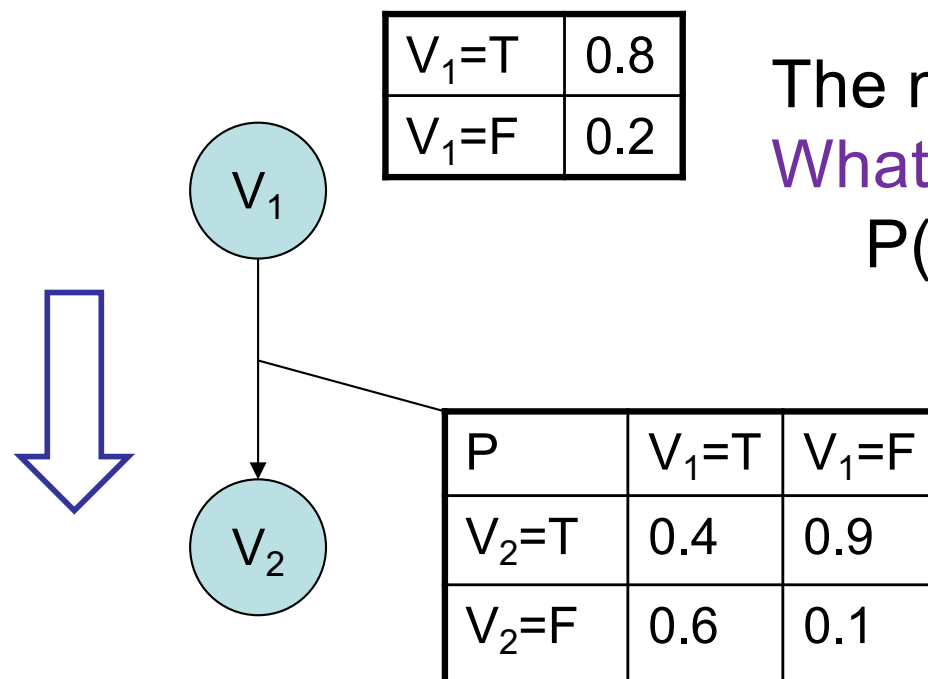


# Pearl's Belief Propagation (assuming polytree)



# The $\pi$ (downward) Messages

- What are the  $\pi$  (downward) messages?
- For simplicity, let the nodes be binary. Marginal  $P(V_1)$  is easy to compute locally at  $V_1$ . What about  $V_2$



The message passes on information.

**What information?** We can rewrite:

$$P(V_2) = P(V_2 | V_1=T)P(V_1=T) + P(V_2 | V_1=F)P(V_1=F)$$

**Thus, the information that  $V_2$  requires is the CPT of  $V_1 = \pi_V(V_1)$**

**$\pi$  Messages capture information passed from parents to children**



# Evidence easy to incorporate

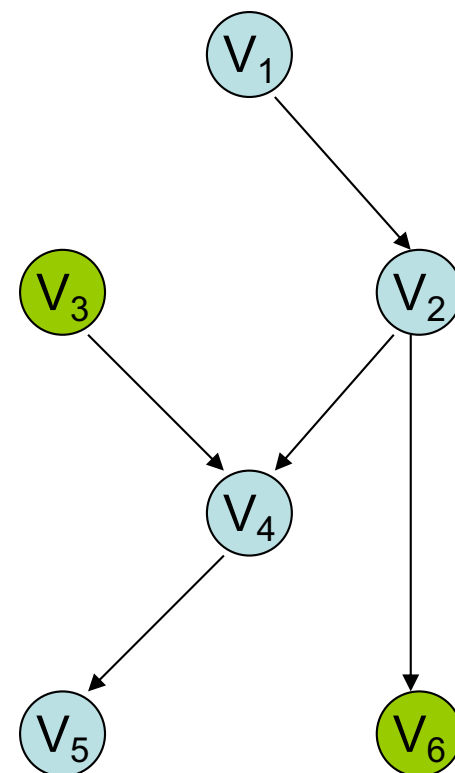
- Evidence – values of observed nodes
  - $V_3 = T, V_6 = 3$
- Our belief in what the value of  $V_i$  'should' be changed.
- This belief is propagated
- As if the CPTs became

$V_3=T$	1.0
$V_3=F$	0.0

P	$V_2=T$	$V_2=F$
$V_6=1$	0.0	0.0
$V_6=2$	0.0	0.0
$V_6=3$	1.0	1.0

In that sense

$$\begin{aligned}
 P(V_2 | V_1) &= P(V_2 | V_1=T)P(V_1=T) \\
 &\quad + P(V_2 | V_1=F)P(V_1=F)
 \end{aligned}$$

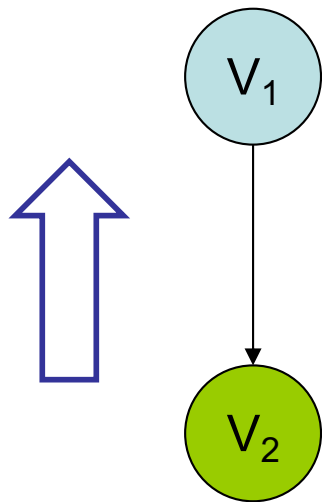


# Now, the $\lambda$ (upward) Messages

- We know what the  $\pi$  (downward) messages are
- What about  $\lambda$ , i.e., the messages going upwards?

Assume  $E = \{ V_2 \}$  (otherwise “boring”) and compute by Bayes rule:

$$P(V_1 | V_2) = \frac{P(V_1)P(V_2 | V_1)}{P(V_2)} = \alpha P(V_1)P(V_2 | V_1)$$



The information not available at  $V_1$  is  $P(V_2 | V_1)$ , to be passed upwards by a  $\lambda$ -message. Again, this is not exactly the CPT, but the belief based on evidence down the tree, and we have just seen how to compute this.

- To sum up, the messages are  $\pi(V)=P(V|E^+)$  and  $\lambda(V)=P(E^-|V)$



# Combination of evidence

- So, we can send messages locally. More precisely, recall that  $E_V = E_V^+ \cup E_V^-$  and let us compute

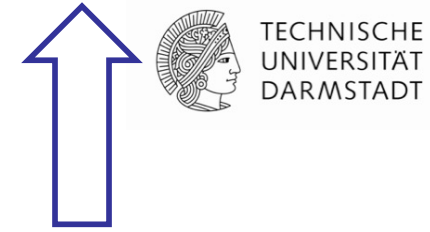
$$\begin{aligned} P(V \mid E) &= P(V \mid E_V^+, E_V^-) = \alpha' P(E_V^+, E_V^- \mid V) P(V) = \\ &\alpha' P(E_V^- \mid V) P(E_V^+ \mid V) P(V) = \alpha P(E_V^- \mid V) P(V \mid E_V^+) = \\ &\alpha \lambda(V) \pi(V) = BEL(V) \end{aligned}$$

- $\alpha$  is the normalization constant
  - normalization is not necessary (can do it at the end), but may prevent numerical underflow problems
- **In other words, we can maintain an approx. belief locally. In general, it is approx. due to the strong independence assumption we made.**





# Upward Belief



- Assume  $X$  received  $\lambda$ -messages from neighbors
- How to compute  $\lambda(x) = p(e^-|x)$ ?
- Let  $Y_1, \dots, Y_c$  be the children of  $X$
- $\lambda_{XY}(x)$  denotes the  $\lambda$ -message sent from  $Y$  to  $X$

$$\lambda(x) = \prod_{j=1}^c \lambda_{Y_j X_i}(x)$$

Derivation clear: Assume two children  $Y, Z$  of  $X$ ,  
then  $P(e^-|x) = P(e^-_Y, e^-_Z|x) = P(e^-_Y|x) * P(e^-_Z|x)$   
(siblings independent given parent)



# Downward Belief



- Assume  $X$  received  $\pi$ -messages from neighbors
- How to compute  $\pi(x) = p(x|e^+)$  ?
- Let  $U_1, \dots, U_p$  be the parents of  $X$
- $\pi_{XY}(x)$  denotes the  $\pi$ -message sent from  $Y$  to  $X$
- Summation over the CPT

$$\pi(x) = \sum_{u_1, \dots, u_p} P(x | u_1, \dots, u_p) \prod_{j=1}^p \pi_{U_j X_i}(u_j)$$

Derivation: We have to sum over each state of  $X$ , now given its parents. That is, we condition on all  $U_i$  and note that each pair (parent  $U_i$ , evidence  $e^+_i$ ) is independent of the other pairs ( $U_j, e^+_j$ )



# The messages to pass

- We need to compute  $\pi_{XY_j}(x)$

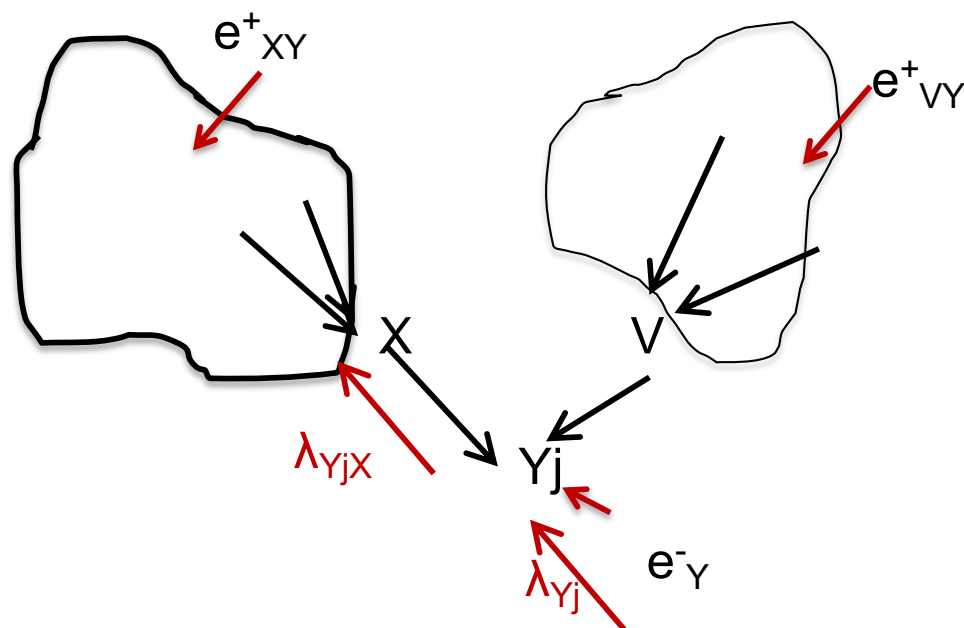
$$\pi_{XY_j}(x) = \alpha \pi_X(x) \prod_{k \neq j} \lambda_{Y_k X}(x)$$

Derivation

$$\begin{aligned} \pi_{XY_j} &= P(x \mid e^+_{XY_j}) \\ &= P(x \mid e - e^-_{XY_j}) = \\ &= [\text{Bel}(x) \text{ when evidence } e^-_{XY_j} \text{ is suppressed}] \\ &= \text{Bel}(x) \text{ setting } \lambda_{XY_j}(x) = 1 \\ &= \text{formula above} \end{aligned}$$



# Messages to pass



## Derivation hint

- $\lambda_{Y_j X} = P(e^-_{XY} | x)$   
 $= P(e^+_{VY_j}, e^-_{Y_j} | X)$
- then condition on  $Y_j$  and  $V$ ,
- And use independences
  - $Y_j$  separates  $e^+_{VY_j}$  from  $e^-_{Y_j}$
  - $V$  separates  $e^+_{VY_j}$  from  $X$

- Symbolically, group other ( $\neq X$ ) parents of  $Y_j$  into single complex  $V = V_1, \dots, V_q$  and distinguish link  $X \rightarrow Y_j$  vs.  $V \rightarrow Y_j$

$$\lambda_{Y_j X}(x) = \sum_{y_j} \lambda_{Y_j}(y_j) \sum_{v_1, \dots, v_q} p(y | v_1, \dots, v_q) \prod_{k=1}^q \pi_{V_k Y_j}(v_k)$$



# Summary of the messages

- So, we need to compute  $\pi_{XY}(x)$

$$\pi_{XY_j}(x) = \alpha \pi_X(x) \prod_{k \neq j} \lambda_{Y_k X}(x)$$

Evidential support from parents

Evidential support from children

- Similarly,  $\lambda_{XY}(x)$
- Group all parents of Y into  $V = V_1, \dots, V_q$

Evidential support from parents

$$\lambda_{Y_j X}(v_j) = \sum_x \lambda_{Y_j}(x) \sum_{v_k: k \neq j} p(x | v_1, \dots, v_n) \prod_{k \neq j} \pi_{V_k Y_j}(v_k)$$

Evidential support from children



# The Pearl Belief Propagation Algorithm

- We can summarize the algorithm now:
  - Initialization step
    - For all  $V_i = e_i$  in  $E$ :
      - $\lambda(x_i) = 1$  wherever  $x_i = e_i$ ; 0 otherwise
      - $\pi(x_i) = 1$  wherever  $x_i = e_i$ ; 0 otherwise
    - For nodes without parents
      - $\pi(x_i) = p(x_i)$  - prior probabilities
    - For nodes without children
      - $\lambda(x_i) = 1$  uniformly (normalize at end)



# The Pearl Belief Propagation Algorithm

- Iterate until no change occurs
  - (For each node  $X$ ) if  $X$  has received all the  $\pi$  messages from its parents, calculate  $\pi(x)$
  - (For each node  $X$ ) if  $X$  has received all the  $\lambda$  messages from its children, calculate  $\lambda(x)$
  - (For each node  $X$ ) if  $\pi(x)$  has been calculated and  $X$  received all the  $\lambda$ -messages from all its children (except  $Y$ ), calculate  $\pi_{XY}(x)$  and send it to  $Y$ .
  - (For each node  $X$ ) if  $\lambda(x)$  has been calculated and  $X$  received all the  $\pi$ -messages from all parents (except  $U$ ), calculate  $\lambda_{XU}(x)$  and send it to  $U$ .
- Compute  $\text{BEL}(X) = \lambda(x)\pi(x)$  and normalize



# Complexity

- On a polytree, the BP algorithm converges in time proportional to diameter of network – at most linear
- Work done in a node is proportional to the size of CPT
- Hence BP is linear in number of network parameters





# Most Graphs are not Polytrees

- Cutset conditioning
  - Instantiate a node in cycle, absorb the value in child's CPT.
  - Do it with all possible values and run belief propagation.
  - Sum over obtained conditionals
  - Hard to do
    - Need to compute  $P(c)$
    - Exponential explosion - minimal cutset desirable (also NP-complete)
- Clustering algorithm
- Approximate inference
  - **Loopy BP**, **Sampling Approaches**



# Loopy Belief Propagation

- If BP is used on graphs with loops, messages may circulate indefinitely
- **Empirically, a good approximation is still achievable**
  - Stop after fixed # of iterations
  - Stop when no significant change in beliefs
  - If solution is not oscillatory but converges, it usually is a good approximation
- Example: Turbo Codes, **Lifted Inference**



# Sampling

- Input: Bayesian network with set of nodes  $X$
- Sample = a tuple with assigned values
$$s = (X_1 = x_1, X_2 = x_2, \dots, X_k = x_k)$$
- Tuple may include all variables (except evidence) or a subset
- Sampling schemas dictate how to generate samples
- Ideally, samples are distributed according to  $P(X|E)$



# Sampling Fundamentals

Given a set of variables  $X = \{X_1, X_2, \dots, X_n\}$  that represent a joint probability distribution  $\pi(\mathbf{X})$  and some function  $g(X)$ , we can compute the expected value of  $g(\mathbf{X})$  as follows:

$$E_{\pi} g = \int g(x) \pi(X) dx$$



# Sampling From $\pi(\mathbf{X})$

A sample  $\mathbf{S}^t$  is an instantiation:

$$\mathbf{S}^t = \{x_1^t, x_2^t, \dots, x_n^t\}$$

Given independent, identically distributed samples  
(iid)  $\mathbf{S}^1, \mathbf{S}^2, \dots, \mathbf{S}^T$  from  $\pi(\mathbf{X})$ , it follows from **Strong Law of Large Numbers**:

$$\bar{g} = \frac{1}{T} \sum_{t=1}^T g(\mathbf{S}^t)$$



# Sampling Basics

- Given random variable  $X$  over  $D(X)=\{0, 1\}$
- Given  $P(X) = \{0.3, 0.7\}$
- Generate  $k$  samples: 0,1,1,1,0,1,1,0,1
- Approximate  $P'(X)$ :

$$P'(X = 0) = \frac{\#samples(X = 0)}{\#samples} = \frac{4}{10} = 0.4$$

$$P'(X = 1) = \frac{\#samples(X = 1)}{\#samples} = \frac{6}{10} = 0.6$$

$$P'(X) == \{0.4, 0.6\}$$



# How to draw a sample ?

- Given random variable  $X$ ,  $D(X)=\{0, 1\}$
- Given  $P(X) = \{0.3, 0.7\}$
- Sample  $X \leftarrow P(X)$ 
  - draw random number  $r \in [0, 1]$
  - If  $(r < 0.3)$  then set  $X=0$
  - Else set  $X=1$
- Can generalize for any domain size



# Sampling in BN

- Same Idea: generate a set of samples  $T$
- Estimate  $P(X_i|E)$  from samples
- Challenge:  $X$  is a vector and  $P(X)$  is a huge distribution represented by BN
- Need to know:
  - How to generate a new sample ?
  - How many samples  $T$  do we need ?
  - How to estimate  $P(E=e)$  and  $P(X_i|e)$  ?





# Sampling Algorithms

- Forward Sampling
- Gibbs Sampling (MCMC)
  - Blocking
  - Rao-Blackwellised
- Likelihood Weighting
- Importance Sampling
- Sequential Monte-Carlo (Particle Filtering) in Dynamic Bayesian Networks



# Forward Sampling - No Evidence (Henrion 1988)

Input: Bayesian network

$X = \{X_1, \dots, X_N\}$ ,  $N$  - #nodes,  $T$  - # samples

Output:  $T$  samples

*Process nodes in topological order – first process the ancestors of a node, then the node itself:*

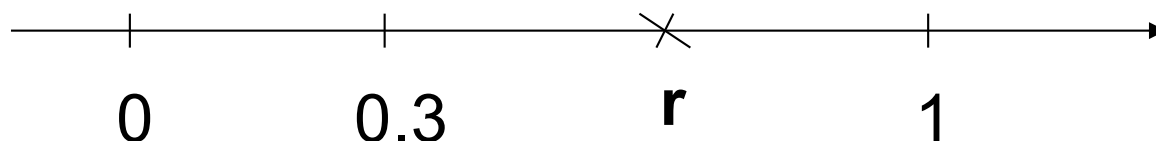
1. For  $t = 0$  to  $T$
2.     For  $i = 0$  to  $N$
3.          $X_i \leftarrow \text{sample } x_i^t \text{ from } P(x_i \mid \text{pa}_i)$



# Sampling a Value

What does it mean to sample  $x_i^t$  from  $P(X_i \mid \text{pa}_i)$  ?

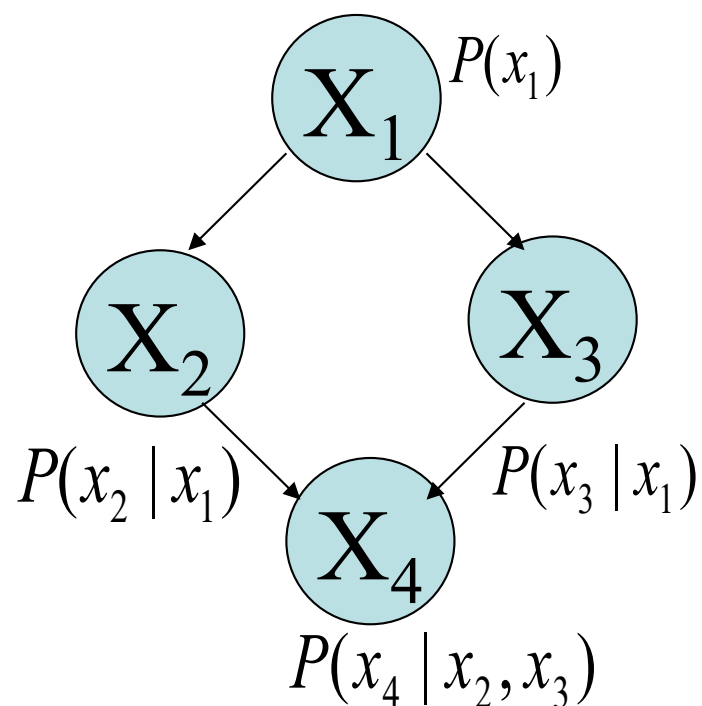
- Assume  $D(X_i) = \{0, 1\}$
- Assume  $P(X_i \mid \text{pa}_i) = (0.3, 0.7)$



- Draw a random number  $r$  from  $[0, 1]$   
If  $r$  falls in  $[0, 0.3]$ , set  $X_i = 0$   
If  $r$  falls in  $[0.3, 1]$ , set  $X_i = 1$



# Forward sampling (example)



No Evidence

// generate sample  $k$

1. Sample  $x_1$  from  $P(x_1)$
2. Sample  $x_2$  from  $P(x_2 | x_1)$
3. Sample  $x_3$  from  $P(x_3 | x_1)$
4. Sample  $x_4$  from  $P(x_4 | x_2, x_3)$



# Forward Sampling-Answering Queries

## Task:

Given  $T$  samples  $\{S^1, S^2, \dots, S^n\}$ , estimate  $P(X_i = x_i)$  :

$$\bar{P}(X_i = x_i) = \frac{\#samples(X_i = x_i)}{T}$$

**Basically, count the proportion of samples where  $X_i = x_i$**



# Forward Sampling w/ Evidence

Input: Bayesian network

$X = \{X_1, \dots, X_N\}$ ,  $N$  - #nodes

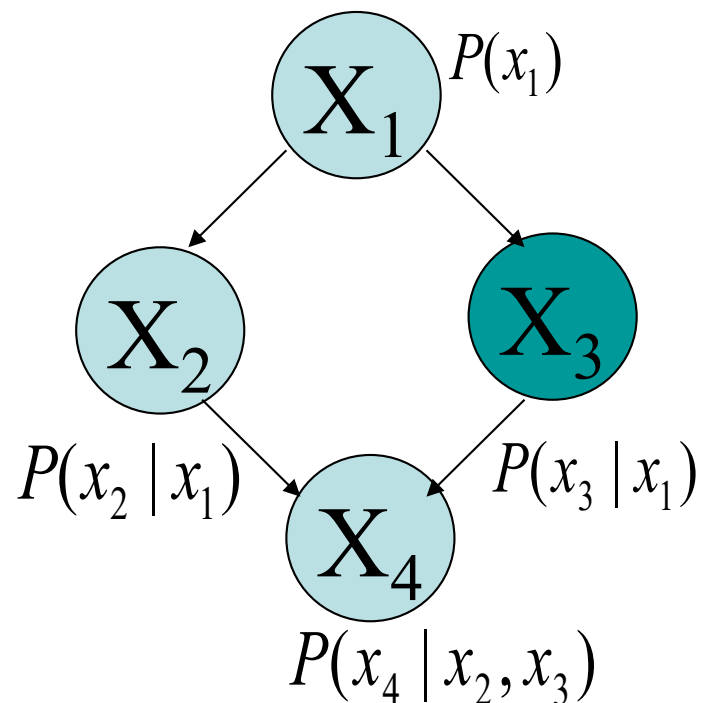
$E$  – evidence,  $T$  - # samples

Output:  $T$  samples consistent with  $E$

1. For  $t=1$  to  $T$
2.     For  $i=1$  to  $N$
3.          $X_i \leftarrow$  sample  $x_i^t$  from  $P(x_i \mid \text{pa}_i)$
4.         **If  $X_i$  in  $E$  and  $X_i \neq x_i$ , reject sample:**
5.              **$i = 1$  and go to step 2**



# Forward sampling (example)



Evidence:  $X_3 = 0$

// generate sample  $k$

1. Sample  $x_1$  from  $P(x_1)$

2. Sample  $x_2$  from  $P(x_2 | x_1)$

3. Sample  $x_3$  from  $P(x_3 | x_1)$

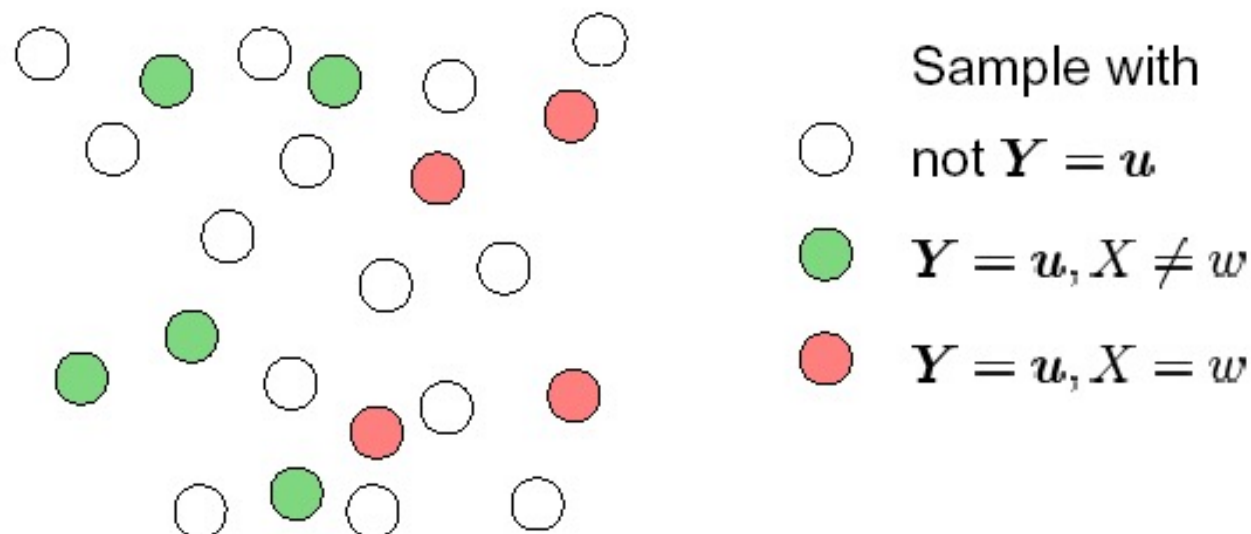
4. If  $x_3 \neq 0$ , reject sample  
and start from 1, otherwise




5. Sample  $x_4$  from  $P(x_4 | x_2, x_3)$



# Forward Sampling: Illustration

Let  $Y$  be a subset of evidence nodes s.t.  $Y = u$



Approximation for  $P^X(X = w \mid Y = u)$ :  $\frac{\# \text{  } }{\# \text{  } \cup \text{  } }$



# Forward Sampling: Performance

## Advantages:

- $P(x_i \mid \text{pa}(x_i))$  is readily available
- Samples are independent !

## Drawbacks:

- If evidence **E** is rare ( $P(e)$  is low), then we will reject most of the samples!
- Since **P(y)** in estimate of **T** is unknown, must estimate **P(y)** from samples themselves!
- If **P(e)** is small, **T** will become very big!



# Problem: Evidence

- Forward Sampling
  - High Rejection Rate
- Fix evidence values
  - Gibbs sampling (MCMC)
  - Likelihood Weighting
  - Importance Sampling



# Gibbs Sampling

- Markov Chain Monte Carlo method  
(Gelfand and Smith, 1990, Smith and Roberts, 1993, Tierney, 1994)
- Samples are **dependent**, form Markov Chain
- Sample from  $P'(X|e)$  which **converges** to  $P(X|e)$
- Guaranteed to converge when all  $P > 0$
- Methods to improve convergence:
  - Blocking
  - Rao-Blackwellised



# Gibbs Sampling (Pearl, 1988)

- A sample  $\mathbf{t} \in [1, 2, \dots]$ , is an instantiation of all variables in the network:

$$\mathbf{x}^t = \{X_1 = x_1^t, X_2 = x_2^t, \dots, X_N = x_N^t\}$$

- Sampling process
  - Fix values of observed variables  $\mathbf{e}$
  - Instantiate node values in sample  $\mathbf{x}^0$  at random
  - Generate samples  $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^T$  from  $P(\mathbf{x}|\mathbf{e})$
  - As before, compute posteriors from samples



# Ordered Gibbs Sampler

Generate sample  $x^{t+1}$  from  $x^t$  :

Process  
All  
Variables  
In Some  
Order

$$X_1 = x_1^{t+1} \leftarrow P(x_1 \mid x_2^t, x_3^t, \dots, x_N^t, e)$$

$$X_2 = x_2^{t+1} \leftarrow P(x_2 \mid x_1^{t+1}, x_3^t, \dots, x_N^t, e)$$

...

$$X_N = x_N^{t+1} \leftarrow P(x_N \mid x_1^{t+1}, x_2^{t+1}, \dots, x_{N-1}^{t+1}, e)$$

In short, for  $i=1$  to  $N$ :

$$X_i = x_i^{t+1} \leftarrow \text{sampled from } P(x_i \mid x^t \setminus x_i, e)$$

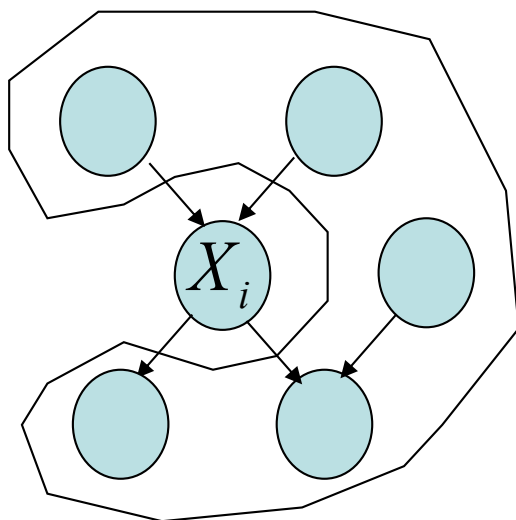


# Gibbs Sampling (cont'd)

(Pearl, 1988)

Important:  $P(x_i | x^t \setminus x_i) = P(x_i | \text{markov}^t \setminus x_i)$ :

$$P(x_i | x^t \setminus x_i) \propto P(x_i | pa_i) \prod_{X_j \in ch_i} P(x_j | pa_j)$$



Markov blanket:

$$M(X_i) = pa_i \cup ch_i \cup \left( \bigcup_{X_j \in ch_j} pa_j \right)$$

Given *Markov blanket*

(parents, children, and their parents),

$X_i$  is independent of all other nodes

# Ordered Gibbs Sampling Algorithm

Input:  $X, E$

Output:  $T$  samples  $\{x^t\}$

- Fix evidence  $E$
- Generate samples from  $P(X \mid E)$
- 1. For  $t = 1$  to  $T$  (compute samples)
- 2.   For  $i = 1$  to  $N$  (loop through variables)
- 3.      $X_i \leftarrow$  sample  $x_i^t$  from  $P(X_i \mid \text{markov}^t \setminus X_i)$



- **Query:**  $P(x_i | e) = ?$
- **Method 1:** count #of samples where  $X_i = x_i$ :

$$\bar{P}(X_i = x_i) = \frac{\#samples(X_i = x_i)}{T}$$

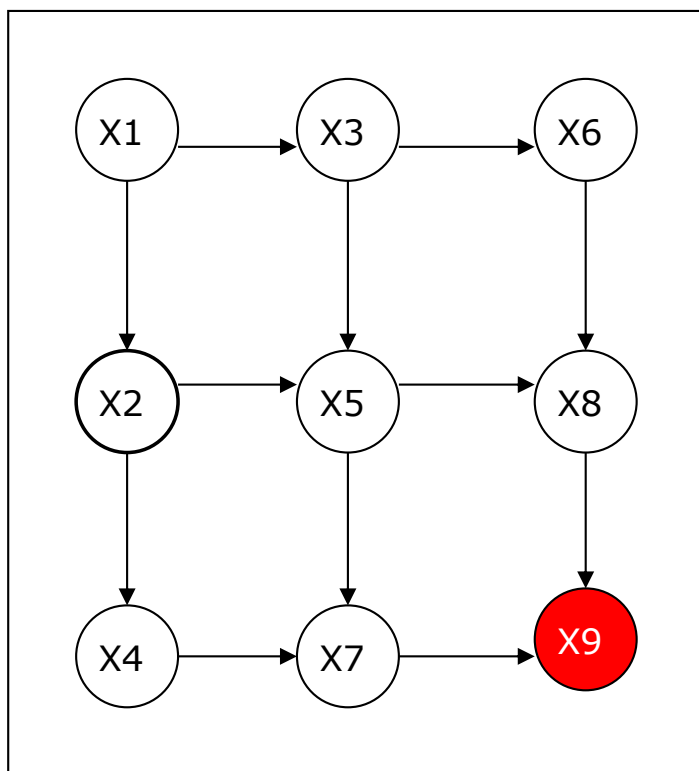
**Method 2:** average probability (mixture estimator):

$$\bar{P}(X_i = x_i) = \frac{1}{T} \sum_{t=1}^n P(X_i = x_i | markov^t \setminus X_i)$$





# Gibbs Sampling Example - BN

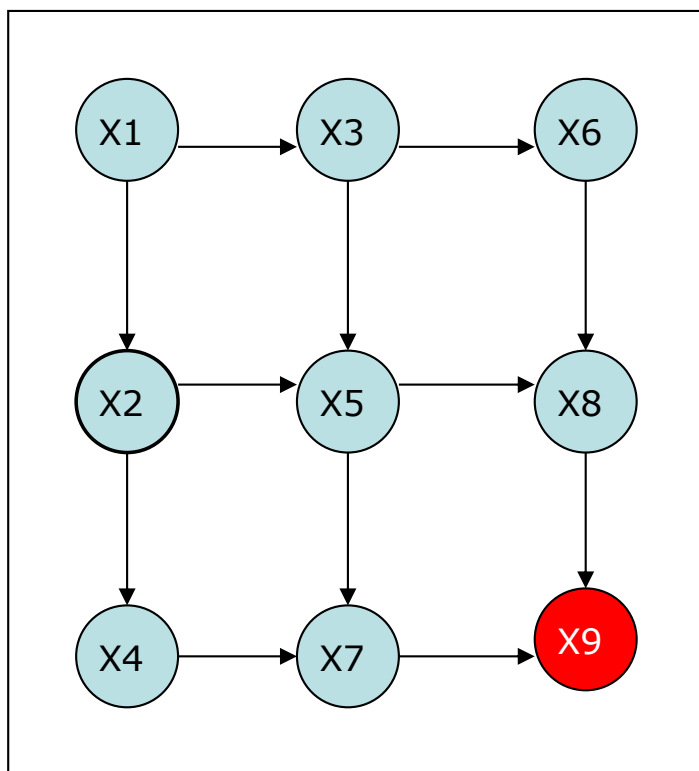


$$X = \{X_1, X_2, \dots, X_9\}$$

$$E = \{X_9\}$$



# Gibbs Sampling Example - BN



$$X_1 = x_1^0$$

$$X_2 = x_2^0$$

$$X_3 = x_3^0$$

$$X_4 = x_4^0$$

$$X_5 = x_5^0$$

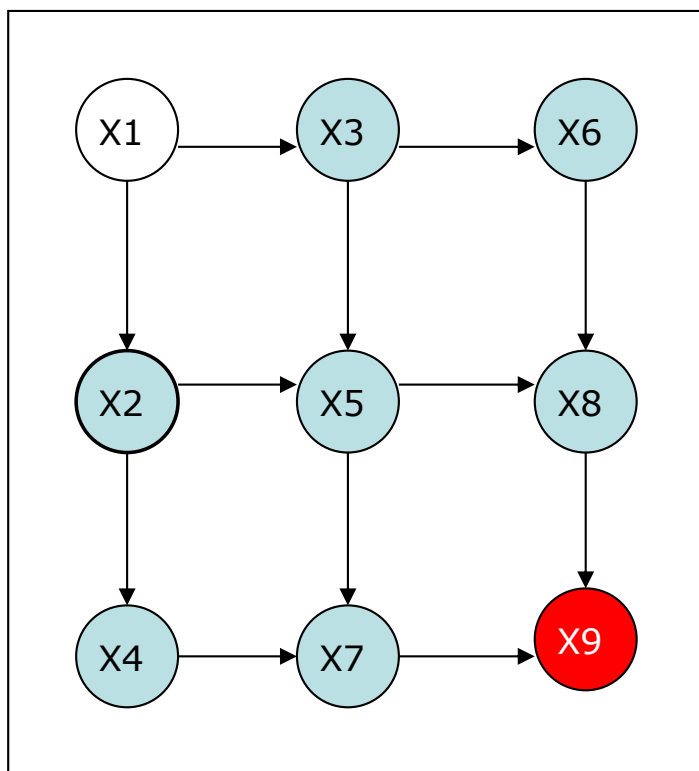
$$X_6 = x_6^0$$

$$X_7 = x_7^0$$

$$X_8 = x_8^0$$



# Gibbs Sampling Example - BN

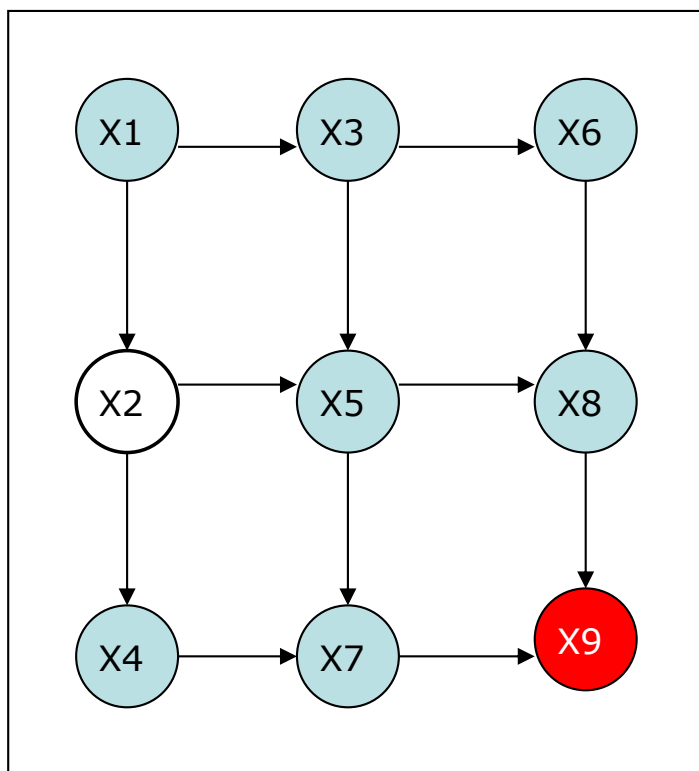


$$X_1 \leftarrow P(X_1 \mid X_2^0, \dots, X_8^0, X_9)$$

$$E = \{X_9\}$$



# Gibbs Sampling Example - BN



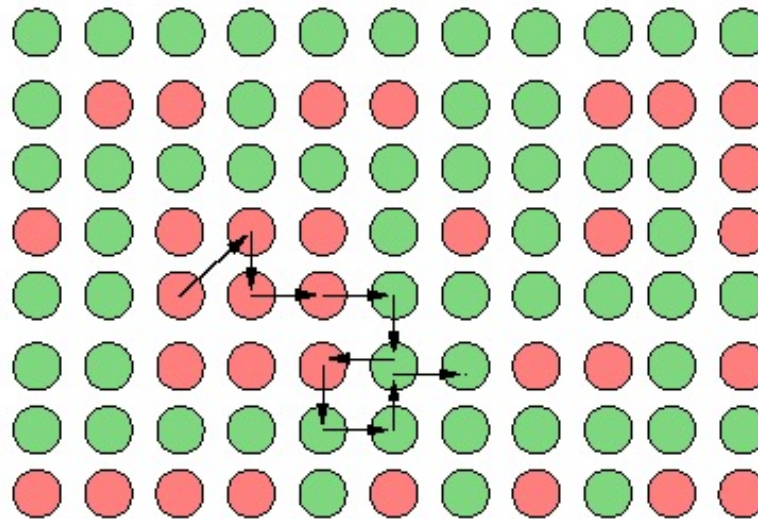
$$X_2 \leftarrow P(X_2 \mid X_1^1, \dots, X_8^0, X_9)$$

$$E = \{X_9\}$$



# Gibbs Sampling: Illustration

The process of Gibbs sampling can be understood as a *random walk* in the space of all instantiations with  $\mathbf{Y} = \mathbf{u}$ :



Reachable in one step: instantiations that differ from current one by value assignment to at most one variable (assume randomized choice of variable  $X_k$ ).

# Gibbs Sampling: Burn-In

- We want to sample from  $P(X | E)$
- But...starting point is **random**
- Solution: throw away first  $K$  samples
- Known As “Burn-In”
- What is  $K$  ? Hard to tell. Use intuition.
- Alternatives: sample first samples from approximate  $P(x|e)$  (for example, run Loopy Belief Propagation first)



# Gibbs Sampling: Convergence

- Converge to stationary distribution  $\pi^*$  :

$$\pi^* = \pi^* P$$

where  $P$  is a transition kernel

$$p_{ij} = P(X^i \rightarrow X^j)$$

- **Guaranteed to converge iff chain is :**
  - irreducible
  - aperiodic
  - ergodic (  $\forall i,j \ p_{ij} > 0$  )



# Background: Irreducible

- A Markov chain (or its probability transition matrix) is said to be *irreducible* if it is possible to reach every state from every other state (not necessarily in one step).
- In other words,  $\forall i, j \exists k : P^{(k)}_{ij} > 0$  where  $k$  is the number of steps taken to get to state  $j$  from state  $i$ .





# Background: Aperiodic

- Define  $d(i) = \text{g.c.d.}\{n > 0 \mid \text{it is possible to go from } i \text{ to } i \text{ in } n \text{ steps}\}$
- Here, g.c.d. means the greatest common divisor of the integers in the set.
- If  $d(i)=1$  for  $\forall i$ , then chain is *aperiodic*, i.e., returns to state  $i$  can occur at **irregular times**



# Background: Ergodicity

- A *recurrent* state is a state to which the chain returns with probability 1:

$$\sum_n P^{(n)}_{ij} = \infty$$

- **Recurrent, aperiodic states are *ergodic*.**

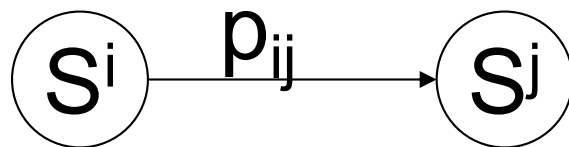
Note: an extra condition for ergodicity is that expected recurrence time is finite. This holds for recurrent states in a finite state chain.



# Background:

## Gibbs Sampling and Ergodicity

- Convergence to the correct distribution is guaranteed only if network is **ergodic**: transition from any state  $S^i$  to any state  $S^j$  has non-zero probability:



$$p_{ij} > 0$$

**Intuition: if  $\exists i, j$  such that  $p_{ij} = 0$ , then we will not be able to explore full sampling space !**



# Gibbs Convergence

- Gibbs convergence is generally guaranteed as long as all probabilities are positive!
  - Intuition for **ergodicity** requirement:  
if nodes  $X$  and  $Y$  are correlated s.t.  $X=0 \leftrightarrow Y=0$ ,  
then:
    - once we sample and assign  $X=0$ , then we are forced to assign  $Y=0$ ;
    - once we sample and assign  $Y=0$ , then we are forced to assign  $X=0$ ;
- ➔ we will never be able to change their values again!
- **Another problem: it can take a very long time to converge!**



# Gibbs Sampling: Performance

- +Advantage: guaranteed to converge to  $P(X|E)$
- Disadvantage: convergence may be slow

## Problems:

- Samples are **dependent** !
- Statistical variance is too big in high-dimensional problems



# Gibbs: Speeding Convergence

Objectives:

1. Reduce dependence between samples (autocorrelation)
  - Skip samples
  - Randomize Variable Sampling Order
2. Reduce variance
  - Blocking Gibbs Sampling
  - Rao-Blackwellisation



# Skipping Samples

- Pick only every  $k$ -th sample (Gayer, 1992)

Can reduce dependence between samples !

Increases variance ! Waists samples !



# Randomized Variable Order

## Random Scan Gibbs Sampler

Pick each next variable  $X_i$  for update at random with probability  $p_i$ ,  $\sum_i p_i = 1$ .

(In the simplest case,  $p_i$  are distributed uniformly)

In some instances, reduces variance

(MacEachern, Peruggia, 1999 “Subsampling the Gibbs Sampler: Variance Reduction”)





# Blocking

- Sample several variables **together, as a block**
- **Example:** Given three variables  $X, Y, Z$ , with domains of size 2, group  $Y$  and  $Z$  together to form a variable  $W = \{Y, Z\}$  with domain size 4. Then, given sample  $(x^t, y^t, z^t)$ , compute next sample:

$$X^{t+1} \leftarrow P(y^t, z^t) = P(w^t)$$

$$(y^{t+1}, z^{t+1}) = W^{t+1} \leftarrow P(x^{t+1})$$

- + Can improve convergence greatly when two variables are strongly correlated!
- Domain of the block variable grows exponentially with the #variables in a block!



# Rao-Blackwellisation

- Do not sample all variables!
- Sample a subset!
- **Example:** Given three variables  $X, Y, Z$ , sample only  $X$  and  $Y$ , sum out  $Z$ . Given sample  $(x^t, y^t)$ , compute next sample:

$$x^{t+1} \leftarrow P(y^t)$$

$$y^{t+1} \leftarrow P(x^{t+1})$$



# Rao-Blackwell Theorem

**Rao Blackwell Theorem:** *Let a DBN have two groups of variables,  $\mathbf{R}$  and  $\mathbf{L}$ . Then, for the joint distribution  $\pi(\mathbf{R}, \mathbf{L})$ , the following result applies*

$$\text{Var}_{\pi} [\mathbb{E}_{\pi} \{f(\mathbf{R}) | \mathbf{L}\}] \leq \text{Var}_{\pi} [f(\mathbf{R})]$$

*for a function of interest  $f$ , e.g. the mean or covariance (Casella & Robert, 1996, Liu et. al. 1995).*

**Bottom line: reducing number of variables in a sample reduce variance!**



# Gibbs: Multiple Chains

- Generate  $M$  chains of size  $K$
- Each chain produces independent estimate  $P_m$ :

$$P_m = P(x_i | e) = \frac{1}{K} \sum_{t=1}^K P(x_i | x^t \setminus x_i)$$

Estimate  $P(x_i|e)$  as average of  $P_m(x_i|e)$  :

$$\bar{P} = \frac{1}{M} \sum_{m=1}^M P_m$$

Treat  $P_m$  as independent random variables.



# Likelihood Weighting

(Fung and Chang, 1990; Shachter and Peot, 1990)

“Clamping” evidence

+ forward sampling

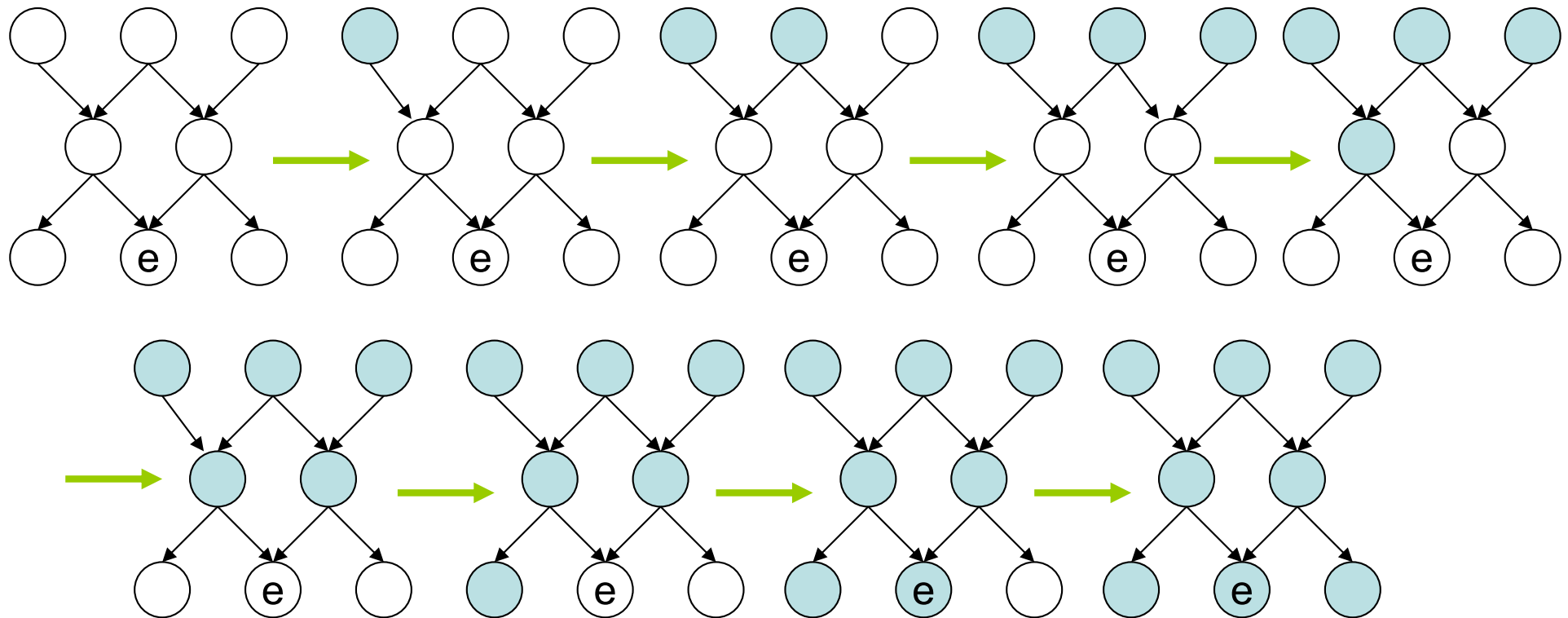
+ weighing samples by evidence likelihood

Works well for *likely evidence*!



# Likelihood Weighting

Sample in topological order over **X** !



$$x_i \leftarrow P(X_i | pa_i)$$

$P(X_i | pa_i)$  is a look-up in CPT!



# Likelihood Weighting

**For** *sample*  $\#k = 1$  to  $T$

For each  $X_i$  in topological order  $o = (X_1, \dots, X_n)$ :

$$w_k = 1$$

**if**  $X_i \notin E$

$X_i \leftarrow$  sample  $x_i$  from  $P(x_i \mid pa_i)$

**else**

assign  $X_i = e_i$

$$w_k = w_k \bullet P(e_i \mid pa_i)$$



# Likelihood Weighting

*Estimate Posterior Marginals:*

$$\hat{P}(x_i | e) = \frac{\hat{P}(x_i, e)}{\hat{P}(e)} = \frac{\sum_{t=1}^T w^{(t)} \delta(x_i, x^{(t)})}{\sum_{t=1}^T w^{(t)}}$$

$$w^{(t)} = \frac{P(x^{(t)})}{Q(x^{(t)})} = \prod_j P(e_j | pa_j^{(t)}) \text{ since } Q(e_j | pa_j) = 1$$





# Likelihood Weighting

- Converges to exact posterior marginals
  - Generates Samples Fast
  - Sampling distribution is close to prior (especially if  $E \subset \text{Leaf Nodes}$ )
  - Increasing sampling variance
- ⇒ Convergence may be slow
- ⇒ Many samples with  $P(x^{(t)})=0$  rejected

