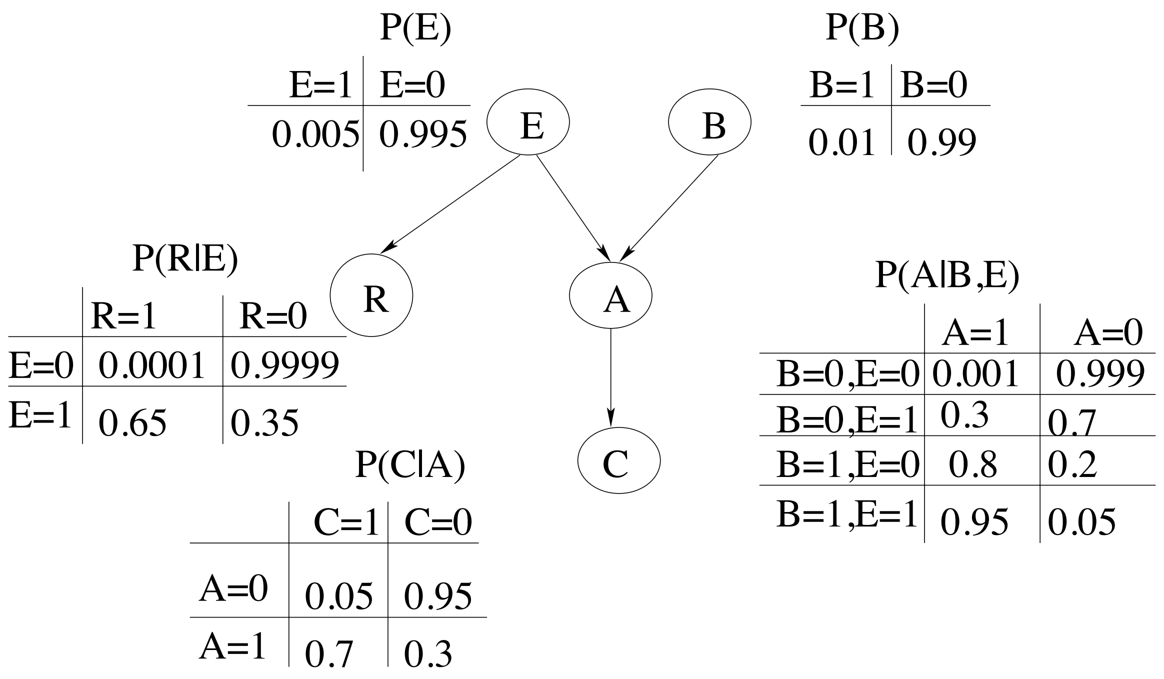
Lecture 8: Bayesian Networks

- Bayesian Networks
- Inference in Bayesian Networks

Bayes nets



P(E)

E=1	E=0
0.005	0.995

P(B)

B=1	B=0
0.01	0.99

P(R|E)

	R=1	R=0
E=0	0.0001	0.9999
E=1	0.65	0.35

P(A|B,E)

	A=1	A=0
B=0,E=0	0.001	0.999
B=0,E=1	0.3	0.7
B=1,E=0	0.8	0.2
B=1,E=1	0.95	0.05

P(C|A)

	C=1	C=0
A=0	0.05	0.95
A=1	0.7	0.3

- The nodes represent random variables
- The arcs represent "influences"
- At each node, we have a conditional probability distribution (CPD) for the corresponding variable *given its parents*

Factorization

- Let G be a DAG over random variables variables $X_1, \ldots, X_n$
- We say that a joint probability distribution P *factorizes* according to G if it can be expressed as a product:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_{\pi_i}), \forall x_i \in Dom(X_i)$$

  where X_{π_i} denotes the parents of X_i in the graph
- The individual factors $P(X_i | X_{\pi_i})$ are called *local probabilistic models* or *conditional probability distributions (CPD)*
- Naive Bayes and Gaussian discriminant models are special cases of this type of model

Complexity of a factorized representation

- If k is the maximum number of ancestors for any node in the graph, and we have binary variables, then every conditional probability distribution will require $\leq 2^k$ numbers to specify
- The whole joint distribution can then be specified with $\leq n \cdot 2^k$ numbers, instead of 2^n
- The savings are big if the graph is sparse ($k \ll n$).

Types of local parametrization

- Tables (if both parents and the node are discrete)
- Tree-structured CPDs (if all nodes are discrete, but the conditional is not always dependent on all the parents)
- Gaussians (for continuous variables)
- Mixture of Gaussians
- ....

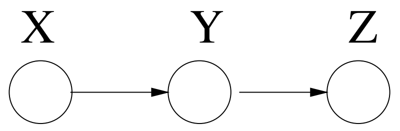
Applications of Bayes nets

- Medical diagnosis
- Bioinformatics (data integration)
- Risk assessment
- Environmental science (e.g., wildlife habitat viability, risk of foreign species invasion)
- Analysis of demographic data
- In general, diagnosis and causal reasoning tasks
- Many commercial packages available (e.g. Netica, Hugin, WinMine, ...)
- Sometimes Bayes net technology is incorporated in business software

Example: Pathfinder (Heckermann, 1991)

- Medical diagnostic system for lymph node diseases
- Large net! 60 diseases, 100 symptoms and test results, 14000 probabilities
- Network built by medical experts
  - 8 hours to determine the variables
  - 35 hours for network topology
  - 40 hours for probability table values
- Experts found it easy to invent causal links and probabilities
- Pathfinder is now *outperforming world experts* in diagnosis
- Commercialized by Intellipath and Chapman Hall Publishing; extended to other medical domains

Implied independencies in Bayes nets: "Markov chain"



- Think of X as the past, Y as the present and Z as the future
- This is a simple **Markov chain**
- Based on the graph structure, we have:

$$p(X,Y,Z) = p(X)p(Y|X)p(Z|Y)$$

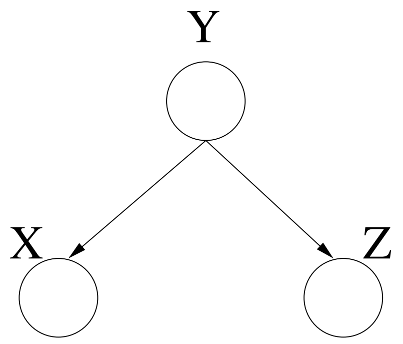
- Hence, we have:

$$p(Z|X,Y) = \frac{p(X,Y,Z)}{p(X,Y)} = \frac{p(X)p(Y|X)p(Z|Y)}{p(X)p(Y|X)} = p(Z|Y)$$

  Hence, $X \perp\!\!\!\perp Z|Y$

- Note that the edges that are *present* do *not* imply dependence. But the edges that are *missing* do imply independence.

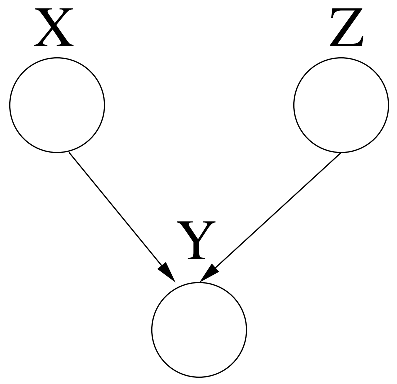
Implied independencies in Bayes nets: "Common cause"



$$p(X|Y,Z) = \frac{p(X,Y,Z)}{p(Y,Z)} = \frac{p(Y)p(X|Y)p(Z|Y)}{p(Y)p(Z|Y)} = p(X|Y)$$

- Based on this, $X \perp\!\!\!\perp Z | Y$.
- This is a "hidden variable" scenario: if Y is unknown, then X and Z could appear to be dependent on each other

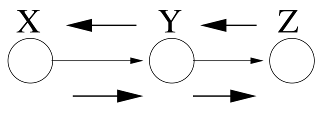
Implied independencies in Bayes nets: "V-structure"



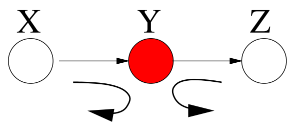
- In this case, the lacking edge between X and Z is a statement of *marginal independence*: $X \perp\!\!\!\perp Z$.
- In this case, once we know the value of Y, X and Z might depend on each other.
- E.g., suppose X and Z are independent coin flips, and Y is true if and only if both X and Z come up heads.
- Note that in this case, X is *not* independent of Z given Y!
- This is the case of "explaining away".

Rules for information propagation

- *Head-to-tail*

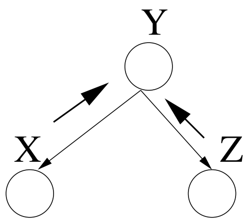


Y unknown, path unblocked

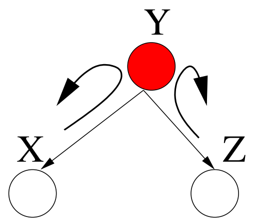


Y known, path blocked

- *Tail-to-tail*

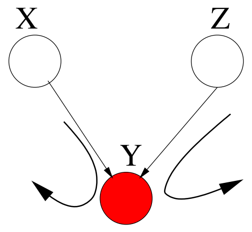


Y unknown, path unblocked

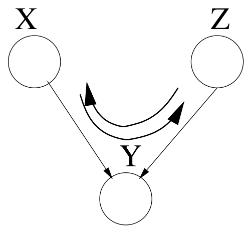


Y known, path blocked

- *Head-to-head*



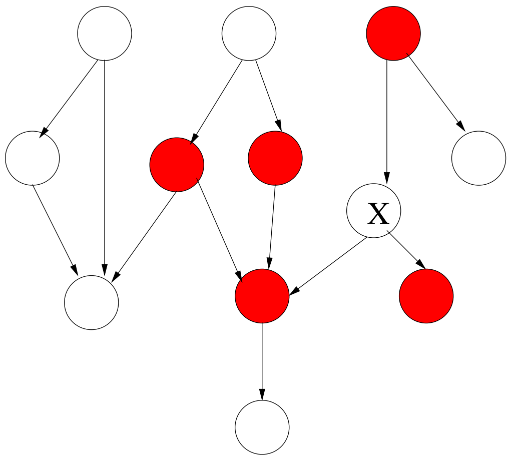
Y unknown, path BLOCKED



Y known, path UNBLOCKED

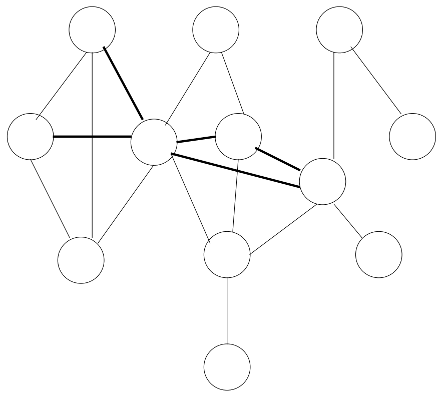
Markov blanket of a node

- Suppose we want the smallest set of nodes U such that X is independent of all other nodes in the network given U. What should U be?
- Clearly, at least X's parents and children should be in U
- But this is not enough if there are v-structures; U will also have to include X's "spouses" - i.e. the other parents of X's children
- The set U consisting of X's parents, children and other parents of its children is called the **Markov blanket** of X.



Moral graphs

- Given a DAG G, we define the **moral graph of G** to be an undirected graph U over the same set of vertices, such that the edge (X, Y) is in U if X is in Y's Markov blanket
- Many independencies are lost in the moral graph
- Moral graphs will prove to be useful when we talk about inference.



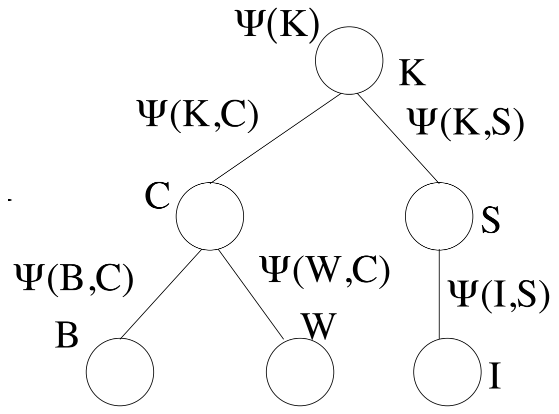
Inference in Bayes nets

- Exact inference
  - Variable elimination
  - Belief propagation
- Approximate inference
  - Loopy belief propagation
  - Rejection sampling
  - Likelihood weighting (a for of importance sampling)
  - Gibbs sampling
  - Variational methods

Applying belief propagation to Bayes nets

1. Construct an undirected model corresponding to the Bayes net, by "moralizing" the graph
2. Potential functions can be constructed based on the original probabilities
3. Use belief propagation (on the junction tree, or the loopy version) on the undirected graph
4. Alternatively, use the information propagation rules to transmit messages

Belief propagation on trees



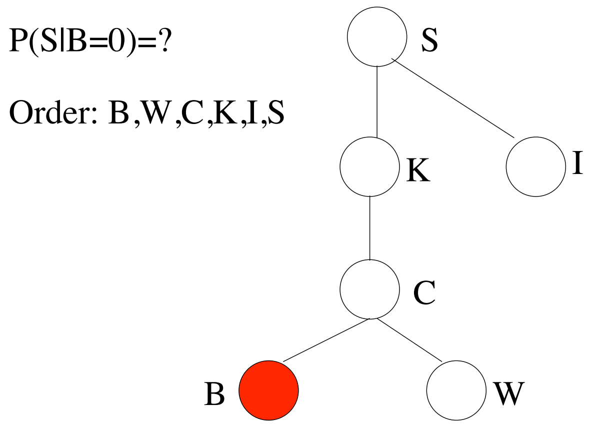
Undirected model

We can parameterize the corresponding undirected model by:

$$\Psi(\text{root}) = p(\text{root}) \text{ and } \Psi(x_j, x_i)$$

for any nodes such that X_i is the parent of X_j

Introducing evidence

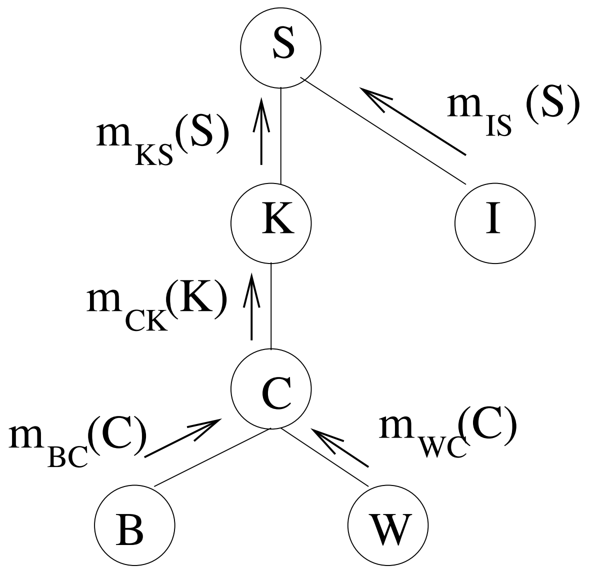


- If we want to compute $p(Y|E)$, we introduce the evidence potential $\delta(x_i, \hat{x}_i)$, for all evidence variables $X_i \in E$
- The potentials now become:

$$\psi^E(x_i) = \begin{cases} \psi(x_i)\delta(x_i, \hat{x}_i) & \text{if } X_i \in E \\ \psi(x_i), & \text{otherwise} \end{cases}$$

- Traverse the tree starting at the leaves, propagating evidence

What are the messages?



The message passed by C to K will be:

$$m_{CK}(K) = \sum_{c} \left(\psi^E(c)\psi(c,k)m_{BC}(c)m_{WC}(c) \right)$$

where $m_{BC}(C)$ and $m_{WC}(C)$ are the messages from B and C.

Belief propagation updates

- Node X_j computes the following message for node X_i

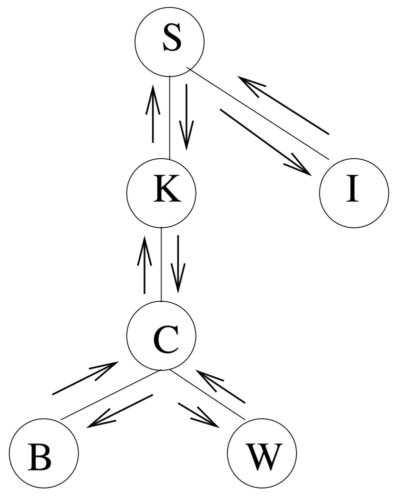
$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \text{neighbors}(x_j) - \{x_i\}} m_{kj}(x_j) \right)$$

  Note that it aggregates the evidence of X_j itself, the evidence received from all neighbors except X_i, and "modulates" it by the potential between X_j and X_i.

- The desired query probability is computed as:

$$p(y|\hat{x}_E) \propto \psi^E(y) \prod_{k \in \text{neighbors}(Y)} m_{ky}(y)$$

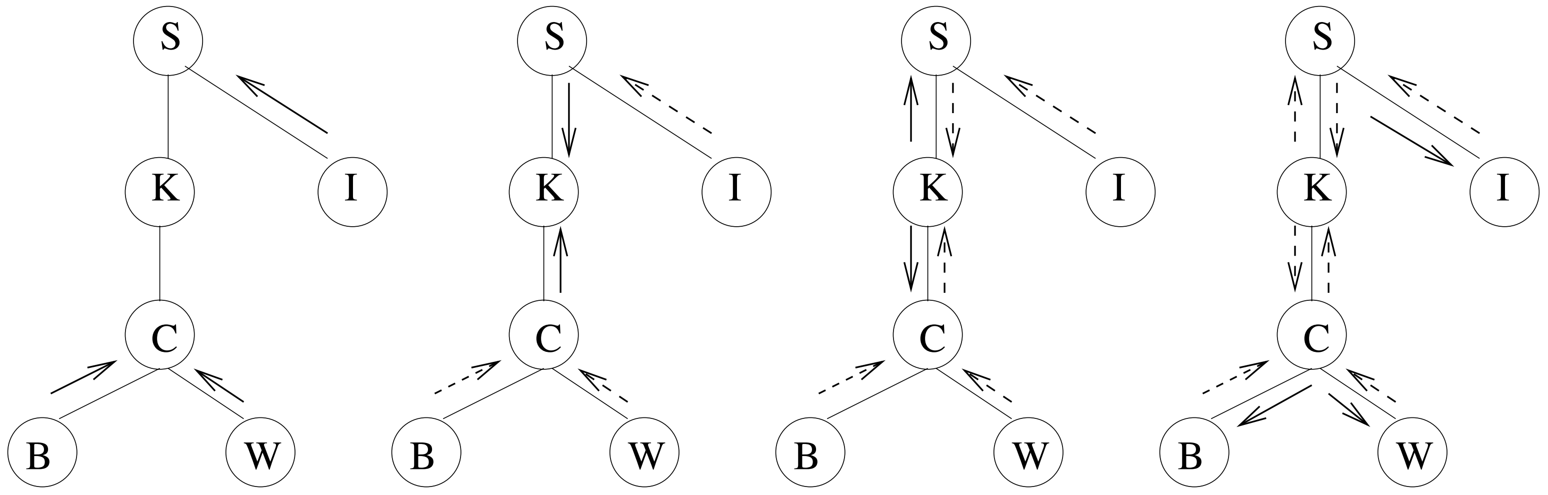
Computing all probabilities simultaneously



- Because messages can be re-used, we can compute *all* conditional probabilities by computing all messages!
- Note that the number of messages is not too big
- We can use our previous equations to compute messages, but we need a protocol for when to compute them

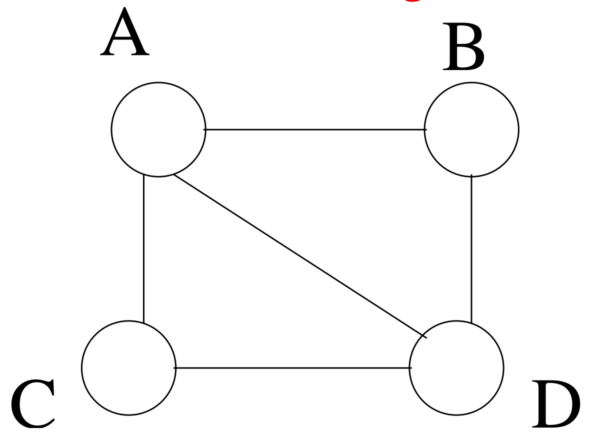
Message-passing protocol

- A node can send a message to a neighbor *after it has received* the messages from *all its other neighbors*.
- Synchronous parallel implementation: any node with d neighbors sends a message after receiving messages on $d-1$ edges
- Solid edges are current messages, dashed are already sent
- This is called the *sum-product algorithm for trees*



Using the tree algorithm - take 1

- Turn the undirected moral graph into a tree!
- Then just use the tree algorithm
- The catch: the graph needs to be *triangulated*



- Finding an optimal triangulation in general is NP-hard
- Usually not the preferred option...

Using the tree algorithm - take 2

- Simply keep computing and propagating messages!
- At node X_j, re-compute the message as:

$$m_{ij}^{new}(x_j) = \sum_{x_i} \psi(X_i, X_j)\psi(X_i) \prod_{k \in \text{Neighbors}(i), k \neq j} m_{ki}^{old}(x_i)$$

- Note that the last two factors are only a function of X_i
- This is called loopy belief propagation

A different idea

- Generate *one sample*, with the given evidence variables instantiated correctly; the sample need not be from the same distribution
- Then *change the values of the variable in the sample* in such a way that it ends up being from the correct distribution
- Repeat until a desired number of samples

Gibbs sampling for Bayes nets

1. Initialization
   - Set evidence variables E, to the observed values e
   - Set all other variables to random values (e.g. by forward sampling)

   This gives us a sample $x_1, \dots, x_n$.
2. Repeat (as much as wanted)
   - Pick a non-evidence variable X_i uniformly randomly)
   - Sample x'_i from $P(X_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$.
   - Keep all other values: $x'_j = x_j, \forall j \neq i$
   - The new sample is $x'_1, \dots, x'_n$
3. Alternatively, you can march through the variables in some predefined order

Why Gibbs works in Bayes nets

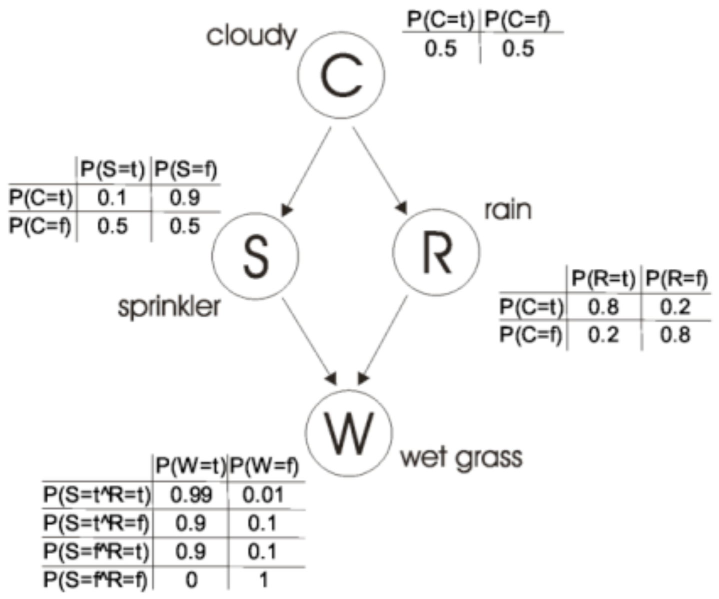
- The key step is sampling according to $P(X_i|x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$. How do we compute this?
- In Bayes nets, we know that a variable is conditionally independent of all others given its Markov blanket (parents, children, spouses)

$$P(X_i|x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = P(X_i|\text{MarkovBlanket}(X_i))$$

- So we need to sample from $P(X_i|\text{MarkovBlanket}(X_i))$
- Let $Y_j, j=1,\ldots,k$ be the children of X_i. It is easy to show that:

$$\begin{aligned} P(X_i = x_i|\text{MarkovBlanket}(X_i)) \quad &\propto \quad P(X_i = x_i|\text{Parents}(X_i)) \cdot \\ &\cdot \quad \prod_{j=1}^{k} P(Y_j = y_j|\text{Parents}(Y_j)) \end{aligned}$$

Example



1. Generate a first sample: $C = 0, R = 0, S = 0, W = 1$.
2. Pick R, sample it from $P(R|C = 0, W = 1, S = 0)$. Suppose we get $R = 1$.
3. Our new sample is $C = 0, R = 1, S = 0, W = 1$
4. ....

Analyzing Gibbs sampling

- Consider the variables $X_1, \ldots, X_n$. Each possible assignment of values to these variables is a state of the world, $\langle x_1, \ldots, x_n \rangle$.
- In Gibbs sampling, we start from a given state $s = \langle x_1, \ldots, x_n \rangle$. Based on this, we generate a new state, $s' = \langle x'_1, \ldots, x'_n \rangle$.
- *s' depends only on s*!
- There is a well-defined probability of going from s to s'.
- Gibbs sampling constructs a *Markov chain* over the Bayes net

Recall: Markov chains

- Suppose you have a system which evolves through time:
$$s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_t \rightarrow s_{t+1} \rightarrow \dots$$
- A *Markov chain* is a special case of such a system, defined by:
  - A set of states S
  - A starting distribution over the set of states $p_0(s) = P(s_0 = s)$. If the state space is discrete, this can be represented as a column vector $\mathbf{p}_0$
  - A stationary transition probability, specifying $\forall s, s' \in S$, $p_{ss'} = P(s_{t+1} = s' | s_t = s)$. The *Markov property* here means that $P(s_{t+1}|s_t) = P(s_{t+1}|s_0, \dots s_t)$.
- For convenience, we put these probabilities in a $|S| \times |S|$ *transition matrix* $\mathbf{T}$.

How does the chain evolve over time?

- Where will the chain be on the first time step, $t = 1$?

$$P(s_{t+1} = s') = \sum_s P(s_0 = s)P(s_1 = s'|s_0 = s)$$

by using the graphical model for the first time step: $s_0 \rightarrow s_1$.

- We can put this in matrix form as follows:

$$\mathbf{p}_1' = \mathbf{p}_0'\mathbf{T} \longrightarrow \mathbf{p}_1 = \mathbf{T}'\mathbf{p}_0$$

where $\mathbf{T}'$ denotes the transpose of $\mathbf{T}$

- Similarly, at $t = 2$, we have:

$$\mathbf{p}_2 = \mathbf{T}'\mathbf{p}_1 = (\mathbf{T}')^2\mathbf{p}_0$$

Steady-state (stationary) distribution

- By induction, the probability distribution over possible states at time step t can be computed as:

$$\mathbf{p}_t = \mathbf{T}'\mathbf{p}_{t-1} = (\mathbf{T}')^t\mathbf{p}_0$$

- If $\lim_{t\to\infty}\mathbf{p}_t$ exists, it is called the *stationary or steady-state distribution* of the chain.
- If the limit exists, $\pi = \lim_{t\to\infty}\mathbf{p}_t$, then we have:

$$\pi = \mathbf{T}'\pi, \sum_{s\in S}\pi_s = 1$$

- Under what conditions does a chain have a stationary distribution?
- Does the equation $\pi = \mathbf{T}'\pi$ always have a unique solution?

Not all chains have a stationary distribution

- If the chain has a purely periodic cycle, the stationary distribution does not exist
- E.g. the system is always in one state on odd time steps and the other state on even time steps, so the probability vector $\mathbf{p}_t$ oscillates between 2 values
- For the limit to exist, the chain must be *aperiodic*
- A standard trick for breaking periodicity is to add self-loops with small probability

Limit distribution may depend on the initial transition

- If the chain has multiple "components", the limit distribution may exist, but depend on a few initial steps
- Such a chain is called *reducible*
- To eliminate this, every state must be able to reach every other state:

$$\forall s, s', \exists k > 0 \text{ s.t. } P(s_{t+k} = s' | s_t = s) > 0$$

Ergodicity

- An *ergodic* Markov chain is one in which any state is reachable from any other state, and there are no strictly periodic cycles (in other words, the chain is irreducible and aperiodic)
- In such a chain, there is a unique stationary distribution π, which can be obtained as:

  $$\pi = \lim_{t \to \infty} \mathbf{p}_t$$

  This is also called the *equilibrium* distribution
- The chain reaches the equilibrium distribution regardless of $\mathbf{p}_0$
- The distribution can be computed by solving:

$$\pi = \mathbf{T}'\pi, \sum_s \pi_s = 1$$

Balance in Markov chains

- Consider the steady-state equation for a system of n states:

$$[\pi_1\pi_2\dots\pi_n] = [\pi_1\pi_2\dots\pi_n]\begin{bmatrix} 1-\sum_{i\neq 1} p_{1i} & p_{12} & \dots & p_{1n} \\ p_{21} & 1-\sum_{i\neq 2} p_{2i} & \dots & p_{2n} \\ \dots & \dots & \dots & \dots \\ p_{n1} & p_{n2} & \dots & 1-\sum_{i\neq n} p_{ni} \end{bmatrix}$$

- By doing the multiplication, for any state s, we get:

$$\pi_s = \pi_s\left(1-\sum_{i\neq s} p_{si}\right) + \sum_{i\neq s}\pi_i p_{is} \Longrightarrow \pi_s\sum_{i\neq s} p_{si} = \sum_{i\neq s}\pi_i p_{is}$$

This can be viewed as a "flow" property: the flow out of s has to be equal to the flow coming into s from all other states

Detailed balance

- Suppose we were designing a Markov chain, and we wanted to ensure a stationary distribution
- This means that the flow equilibrium at every state must be achieved.
- One way to ensure this is to *make the flow equal between any pair of states*:
$$\pi_s p_{ss'} = \pi_{s'} p_{s's}, \forall s, s'$$
This gives us a sufficient condition for stationarity, called *detailed balance*
- A Markov chain with this property is called *reversible*

Gibbs sampling as MCMC

- We have a set of random variables $X = \{x_1 \ldots x_n\}$, with evidence variables $E = e$. We want to sample from $P(X - E|E = e)$.
- Let X_i be the variable to be sampled, currently set to x_i, and $\bar{x}_i$ be the values for all other variables in $X - E - \{X_i\}$
- The transition probability for the chain is: $p_{ss'} = P(x_i'|\bar{x}_i, e)$
- Under mild assumptions on the original graphical model, the chain is ergodic
- We want to show that $P(X - E|e)$ is the stationary distribution

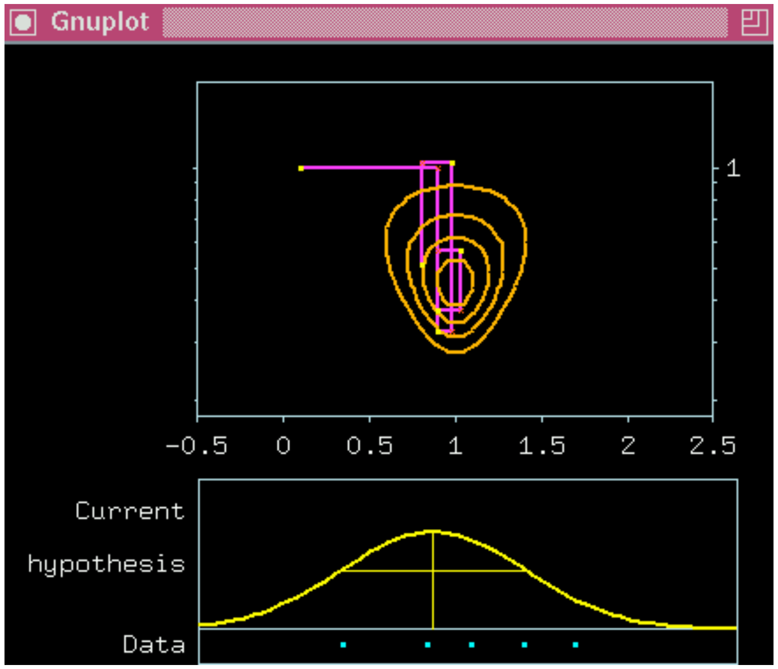
Gibbs satisfies detailed balance

- We show that if we plug in $P(X - E|e)$ as the stationary distribution, detailed balance is satisfied

$$\begin{aligned}
\pi_s p_{ss'} &= P(X - E|e)P(x_i'|\bar{x}_i, e) \\
&= P(x_i, \bar{x}_i|e)P(x_i'|\bar{x}_i, e) \\
&= P(x_i|\bar{x}_i, e)P(\bar{x}_i|e)P(x_i'|\bar{x}_i, e) \text{ (by chain rule)} \\
&= P(x_i|\bar{x}_i, e)P(x_i', \bar{x}_i|e) \text{ (backwards chain rule)} \\
&= p_{s's}\pi_{s'}
\end{aligned}$$

- If the chain is ergodic, there is a unique stationary distribution, and since $P(X - E|e)$ satisfies the balance equation, it must be the stationary distribution.

Example: Gibbs for sampling from a joint distribution (MacKay)



- Start with an initial point
- Draw alternatively from $P(X_2|X_1)$ and $P(X_1|X_2)$
- Line shows the evolution of the samples

Mixing time

- Gibbs sampling can be viewed as a method for *sampling from the stationary distribution* of a Markov chain
- This is done by sampling successively from a sequence $\mathbf{p}_t$ of distributions
- Hence, it is useful to know for what value of t we have $\mathbf{p}_t \approx \pi$
- This is called the *mixing time* of the chain
- If the mixing time is "small" (e.g. compared to the number of states in the chain) we say the chain is *rapidly mixing*
- Unfortunately, the mixing time is hard to estimate (it depends on the gap between the first two eigenvalues of the transition matrix)
- In practice, it is best to wait as long as possible to generate a sample (this is called the *burn-in time*)

Markov Chain Monte Carlo (MCMC) methods

- Suppose you want to generate samples from some distribution, but it is hard to get samples directly

  E.g., We want to sample uniformly the space of graphs with certain properties
- We set up a Markov chain such that its *stationary distribution* is the *desired distribution*
- Note that the 'states" of this chain can be fairly complicated!
- Start at some state, wait for some number of transitions, then take samples
- For this to work we need to ensure that:
  - the chain has a unique stationary distribution
  - the stationary distribution is equal to the desired distribution
  - we reach the stationary distribution quickly

Sampling the equilibrium distribution

- We can sample π just by running the chain a long time:
  - Set $s_0 = i$ for some arbitrary i
  - For $t = 1, \ldots, M$, if $s_t = s$, sample a value s' for s_{t+1} based on $p_{ss'}$
  - Return s_M.

  If M is large enough, this will be a sample from π
- In practice, we would like to have a rapidly mixing chain, i.e. one that reaches the equilibrium quickly

Example: Random graphs

- Suppose you want to sample uniformly from the space of graphs with v vertices and certain properties (e.g. certain in-degree and out-degree bounds, cycle properties...)
- You set up a chain whose states are graphs with v vertices
- Transitions consist of adding or removing an arc (reversal too, if the graphs are directed), with a certain probability
- We start with a graph satisfying the desired property.
- The probabilities are devised based on the distribution that we want to reach in the limit.

Implementation issues

- The initial samples are influenced by the starting distribution, so they need to be thrown away. This is called the *burn-in stage*
- Because burn-in can take a while, we would like to draw several samples from the same chain
- However, if we take samples t, $t+1$, $t+2$..., they will be highly correlated
- Usually we wait for burn-in, then take every nth sample, for some n sufficiently large. This will ensure that the samples are (for all practical purposes) uncorrelated

Learning in Bayesian networks

Given data, in the form of **instances**, e.g.:

Earthquake	Burglary	Alarm	Call	Radio
No	No	No	No	No
No	Yes	Yes	Yes	No
...	...	...	...	...

Create a complete graphical model (graph structure + CPDs)

Two learning problems

1. Known vs. unknown structure of the network

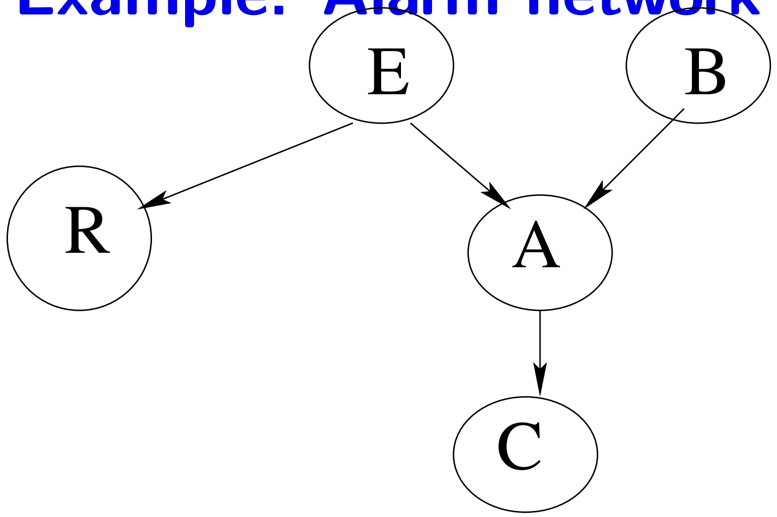
   I.e. do we know the arcs or do we have to find them too?

2. Complete vs. incomplete data

   In real data sets, values for certain variables may be missing, and we need to "fill in"

Today we focus on the case of known network structure with complete data. This problem is called **parameter estimation**

Example: Alarm network



- Instances are of the form $\langle r_j, e_j, b_j, a_j, c_j \rangle, j = 1, \dots m$

- What are the parameters we are trying to estimate?

$$
\begin{aligned}
L(\theta|D) &= \prod_{j=1}^{m} p(r_j, e_j, b_j, c_j, a_j|\theta) \text{ (from i.i.d)} \\
&= \prod_{j=1}^{m} p(e_j)p(r_j|e_j)p(b_j)p(a_j|e_j, b_j)p(c_j|e_j) \text{ (factorization)} \\
&= (\prod_{j=1}^{m} p(e_j))(\prod_{j=1}^{m} p(r_j|e_j))(\prod_{j=1}^{m} p(b_j))(\prod_{j=1}^{m} p(a_j|e_j, b_j))(\prod_{j=1}^{m} p(c_j|e_j)) \\
&= \prod_{i=1}^{n} L(\theta_i|D)
\end{aligned}
$$

where θ_i are the parameters associated with node i.

MLE for Bayes nets

Generalizing, for any Bayes net with variables $X_1, \ldots X_n$, we have:

$$\begin{aligned}
L(\theta|D) &= \prod_{j=1}^{m} p(x_1(j), \ldots x_n(j)|\theta) \text{ (from i.i.d)} \\
&= \prod_{j=1}^{m} \prod_{i=1}^{n} p(x_i(j)|x_{\pi_i}(j)), \theta) \text{ (factorization)} \\
&= \prod_{i=1}^{n} \prod_{j=1}^{m} p(x_i(j)|x_{\pi_i}(j)) \\
&= \prod_{i=1}^{n} L(\theta_i|D)
\end{aligned}$$

The likelihood function **decomposes** according to the structure of the network, which creates **independent estimation problems**.

Missing data

- Use EM!
- We start with an initial set of parameters
- Use inference to fill in the missing values of all variables
- Re-compute the parameters (which now can be done locally at each node)
- These steps are iterated until parameters converge
- Theoretical properties are identical to general EM