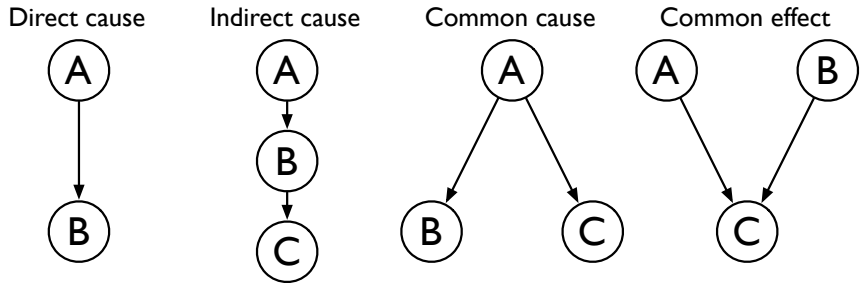
Artificial Intelligence
15-381

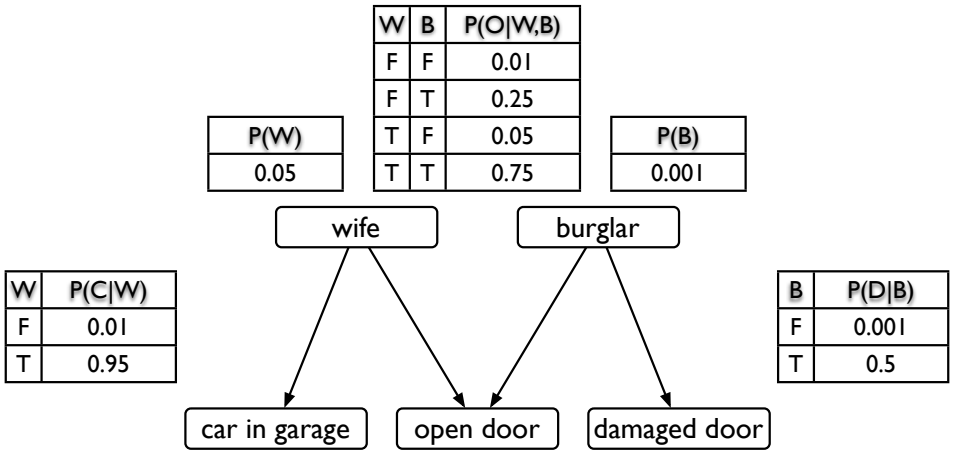
Mar 29, 2007

Bayesian Networks 2

Recap of last lecture: Modeling causal relationships with Bayes nets



Recap (cont'd)

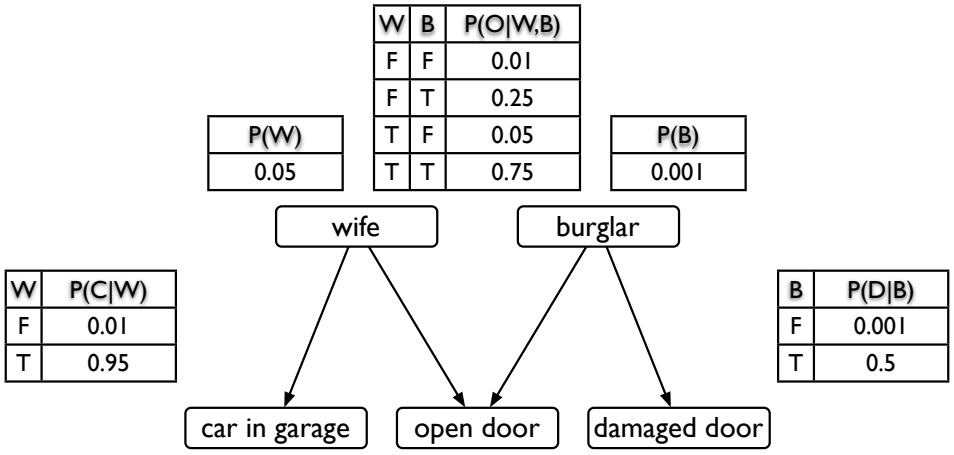


- The structure of this model allows a simple expression for the joint probability

$$P(x_1, \ldots, x_n) \equiv P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$$
$$= \prod_{i=1}^{n} P(x_i | \text{parents}(X_i))$$
$$\Rightarrow P(o, c, d, w, b) = P(c|w)P(o|w, b)P(d|b)P(w)P(b)$$

Recall how we calculated the joint probability on the burglar network:

- P(o,w,¬b,c,¬d) = P(o|w,¬b)P(c|w)P(¬d|¬b)P(w)P(¬b)
  = 0.05 × 0.95 × 0.999 × 0.05 × 0.999 = 0.0024
- How do we calculate P(b|o), i.e. the probability of a burglar given we see the open door?
- This is not an entry in the joint distribution.



Computing probabilities of propositions

- How do we compute P(o|b)?
  - Bayes rule
  - marginalize joint distribution

Variable elimination on the burglary network

- As we mentioned in the last lecture, we could do straight summation:

$$\begin{aligned} p(b|o) &= \alpha\, p(o, w, b, c, d) \\ &= \alpha \sum_{w,c,d} p(o|w,b)p(c|w)p(d|b)p(w)p(b) \end{aligned}$$

- But: the number of terms in the sum is *exponential* in the non-evidence variables.
- This is bad, and we can do much better.
- We start by observing that we can pull out many terms from the summation.

Variable elimination

- When we've pulled out all the redundant terms we get:

$$p(b|o) = \alpha\, p(b) \sum_d p(d|b) \sum_w p(w) p(o|w,b) \sum_c p(c|w)$$

- We can also note the last term sums to one. In fact, every variable that is not an ancestor of a query variable or evidence variable is *irrelevant* to the query, so we get

$$p(b|o) = \alpha\, p(b) \sum_d p(d|b) \sum_w p(w) p(o|w,b)$$

  which contains far fewer terms: In general, *complexity is **linear** in the # of CPT entries.*

- This method is called *variable elimination.*
  - if # of parents is bounded, also linear in the number of nodes.
  - the expressions are evaluated in right-to-left order (bottom-up in the network)
  - intermediate results are stored
  - sums over each are done only for those expressions that depend on the variable

- Note: for multiply connected networks, variable elimination can have exponential complexity in the worst case.
- This is similar to CSPs, where the complexity was bounded by the hypertree width.

Inference in Bayesian networks

- For queries in Bayesian networks, we divide variables into three classes:
  - evidence variables: $e = \{e_1, \ldots, e_m\}$ what you know
  - query variables: $x = \{x_1, \ldots, x_n\}$ what you want to know
  - non-evidence variables: $y = \{y_1, \ldots, y_l\}$ what you don't care about
- The complete set of variables in the network is $\{e \cup x \cup y\}$.
- Inferences in Bayesian networks consist of computing p(x|e), the posterior probability of the query given the evidence:

$$p(x|e) = \frac{p(x,e)}{p(e)} = \alpha\, p(x,e) = \alpha \sum_y p(x,e,y)$$

- This computes the marginal distribution p(x,e) by suming the joint over all values of y.
- Recall that the joint distribution is defined by the product of the conditional pdfs:

$$p(z) = \prod_{i=1} P(z_i | \mathrm{parents}(z_i))$$

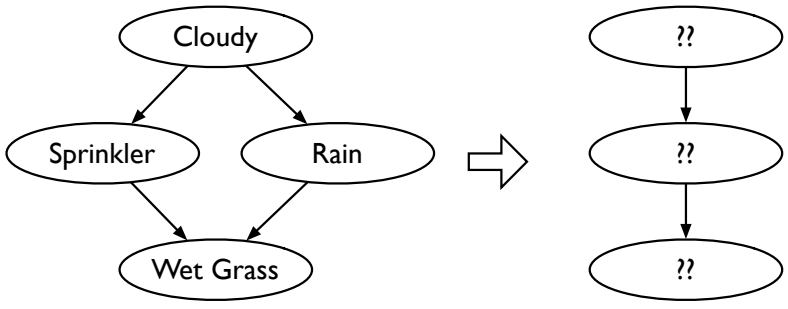
  where the product is taken over all variables in the network.

The generalized distributive law

- The variable elimination algorithm is an instance of the generalized distributive law.
- This is the basis for many common algorithms including:
  - the fast Fourier transform (FFT)
  - the Viterbi algorithm (for computing the optimal state sequence in an HMM)
  - the Baum-Welch algorithm (for computing the optimal parameters in an HMM)
- We'll come back to some of these in a future lecture

Clustering algorithms

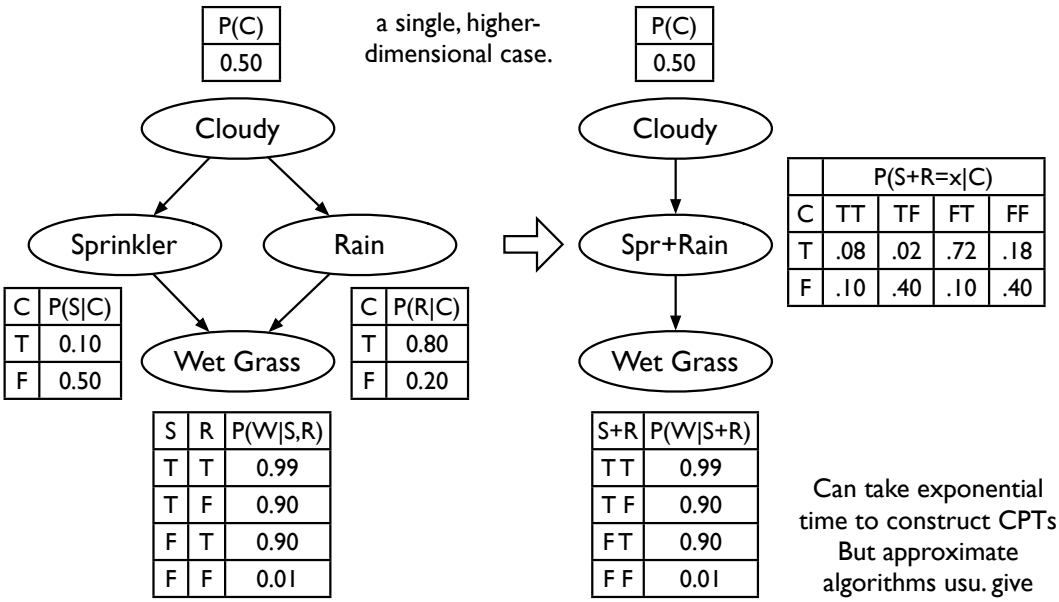
- Inference is efficient if you have a *polytree*, ie a singly connected network.
- But what if you don't?
- Idea: Convert a non-singly connected network to an equivalent singly connected network.



What should go into the nodes?

Clustering or join tree algorithms

Idea: merge multiply connected nodes into a single, higher-dimensional case.



Can take exponential time to construct CPTs
But approximate algorithms usu. give reasonable solutions.

Example: Pathfinder

Pathfinder system. (Heckerman, Probabilistic Similarity Networks, MIT Press, Cambridge MA).

- Diagnostic system for lymph-node diseases.
- 60 diseases and 100 symptoms and test-results.
- 14,000 probabilities.
- Expert consulted to make net.
  - 8 hours to determine variables.
  - 35 hours for net topology.
  - 40 hours for probability table values.
- Apparently, the experts found it quite easy to invent the causal links and probabilities.

Pathfinder is now outperforming the world experts in diagnosis. Being extended to several dozen other medical domains.

Another approach: Inference by stochastic simulation

Basic idea:

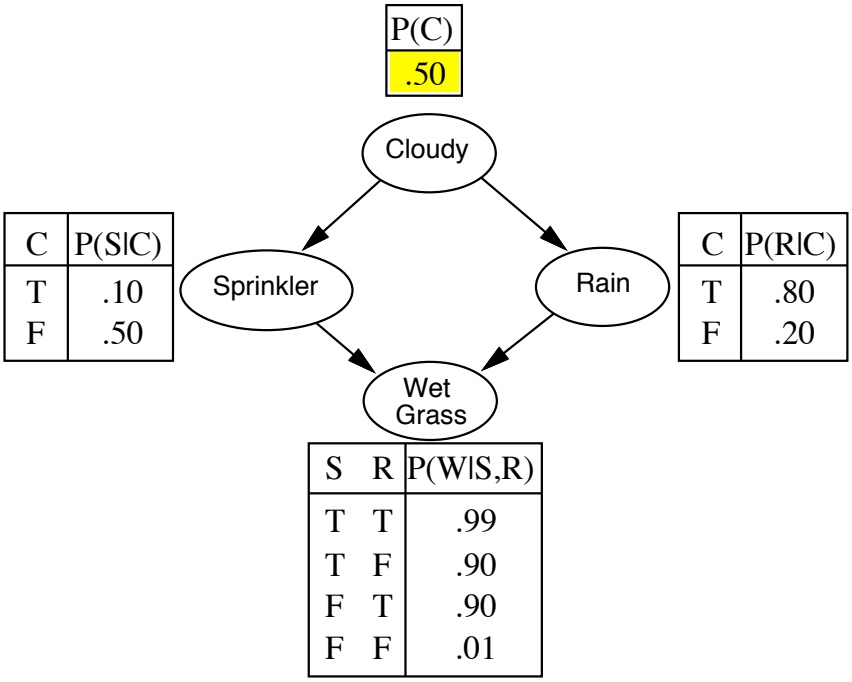
1. Draw N samples from a sampling distribution S
2. Compute an approximate posterior probability
3. Show this converges to the true probability

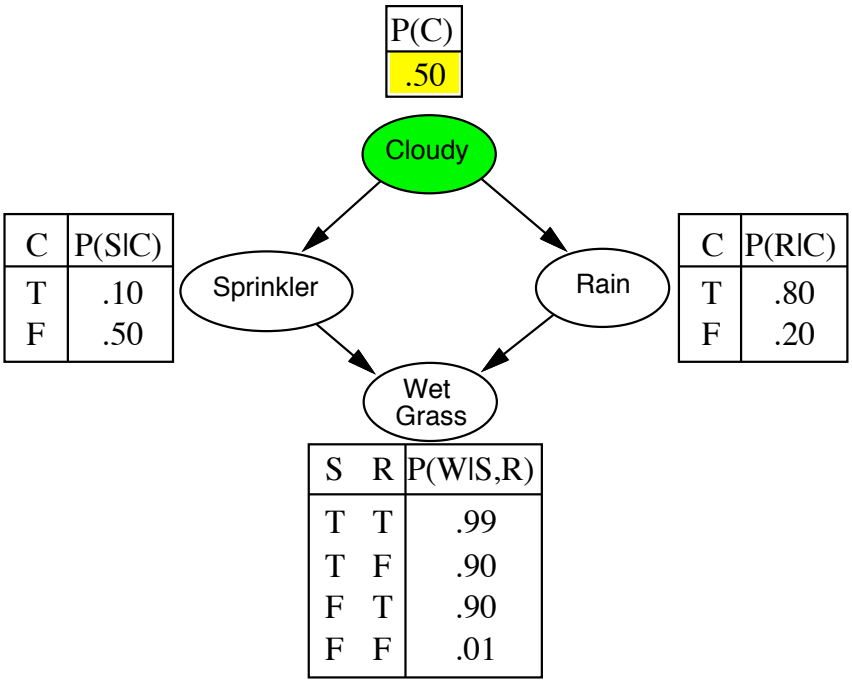
Sampling with no evidence (from the prior)

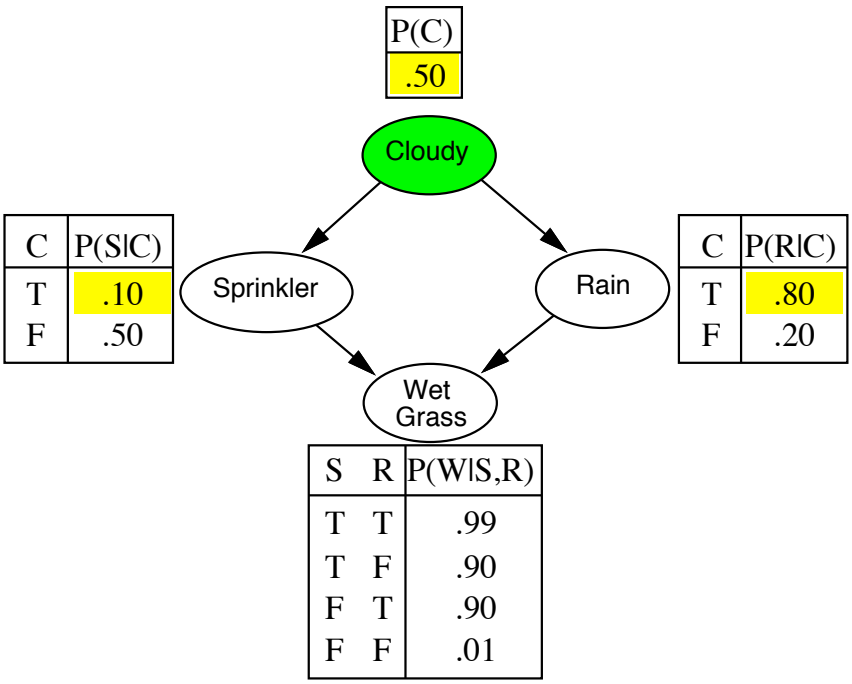
function PRIOR-SAMPLE(bn) **returns** an event sampled from bn
  inputs: bn, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

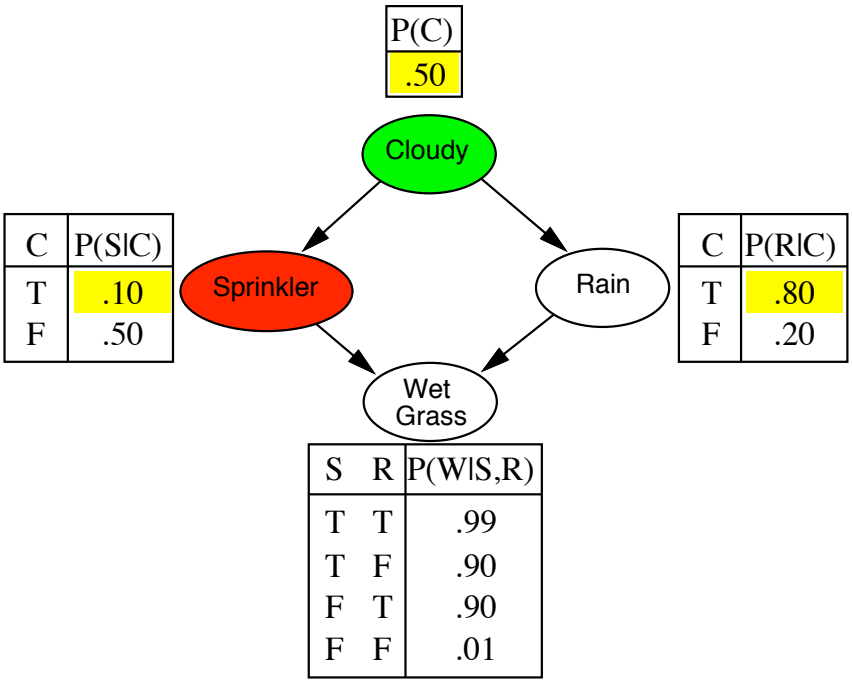
  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
      given the values of $Parents(X_i)$ in $\mathbf{x}$
  return $\mathbf{x}$

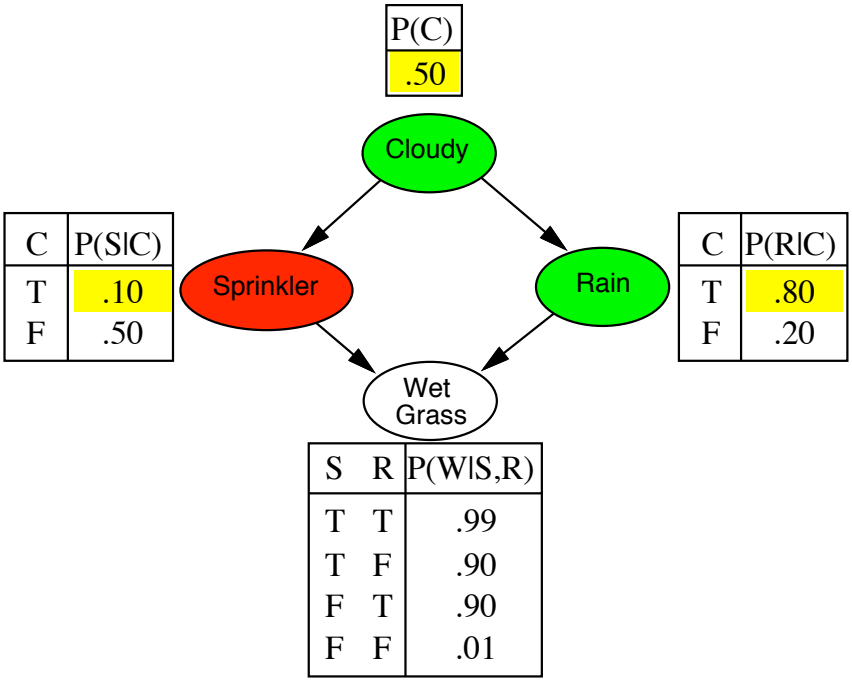
(the following slide series is from our textbook authors.)

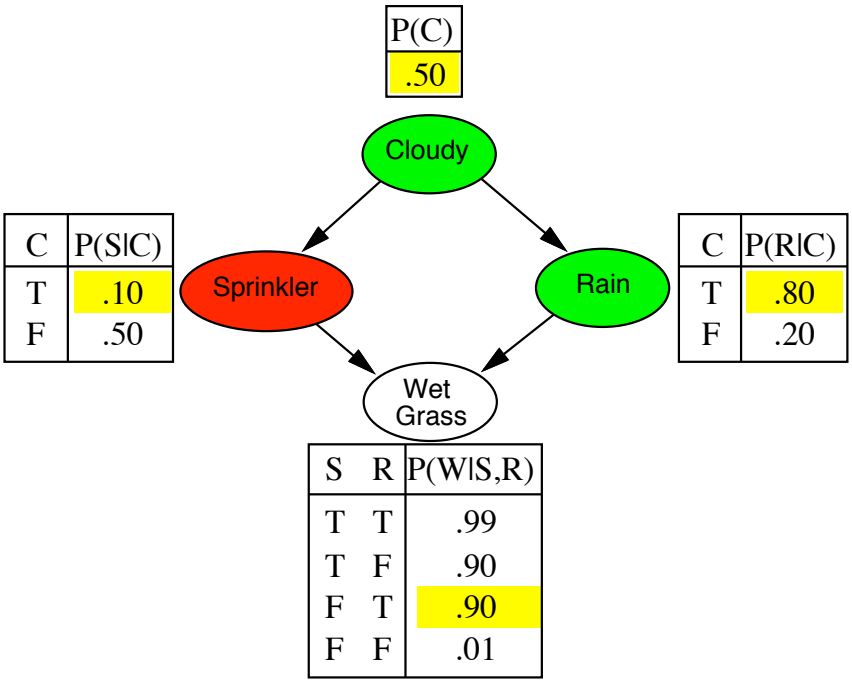

P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

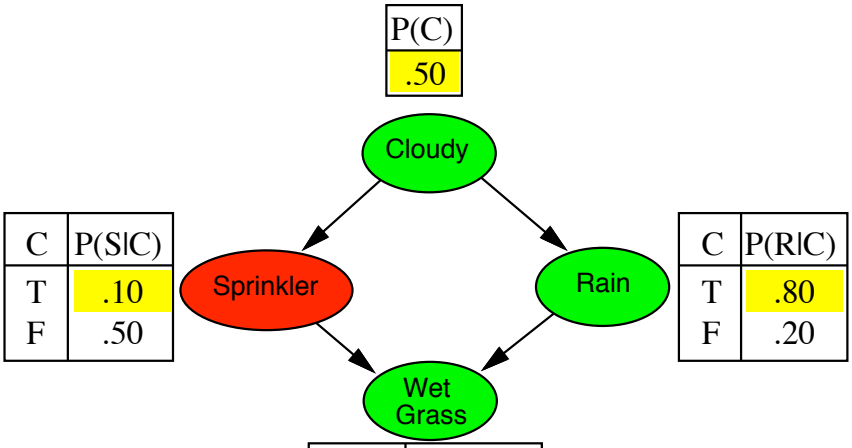

P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01


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


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


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


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



Keep sampling to estimate joint probabilities of interest.

What if we do have some evidence? 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 $P(X|\mathbf{e})$
  local variables: $\mathbf{N}$, a vector of counts over X, initially zero
  for $j = 1$ 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($\mathbf{N}[X]$)

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 Markov Chain Monte Carlo (MCMC)

"State" of network = current assignment to all variables.

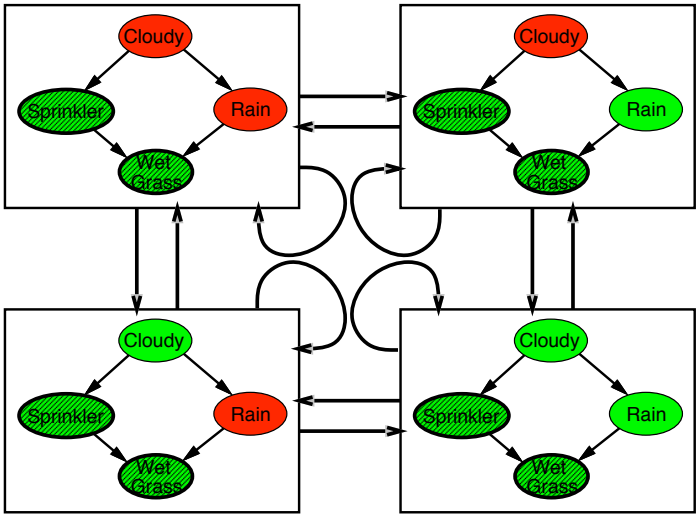
Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

function MCMC-ASK($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
    $\mathbf{Z}$, the nonevidence variables in bn
    $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$
  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Y}$
  for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
      sample the value of Z_i in $\mathbf{x}$ from $\mathbf{P}(Z_i|mb(Z_i))$
        given the values of $MB(Z_i)$ in $\mathbf{x}$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}[X]$)

Can also choose a variable to sample at random each time

The Markov chain

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



Wander about for a while, average what you see

After obtaining the MCMC samples

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 $Rain = false$

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

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

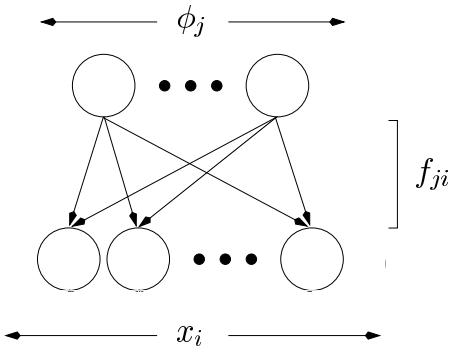
But:
1. Difficult to tell when samples have converged. Theorem only applies in limit, and it could take time to "settle in".
2. Can also be inefficient if each state depends on many other variables.

Gibbs sampling (back to the example from last lecture)

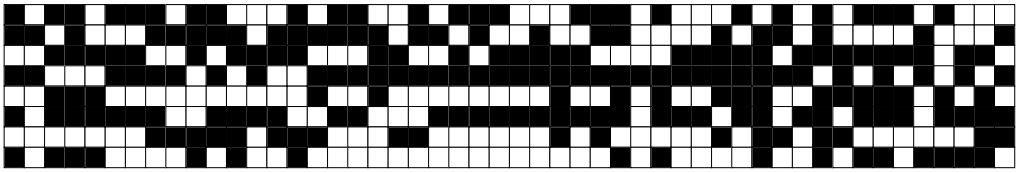
- Model represents stochastic binary features.
- Each input x_i encodes the probability that the ith binary input feature is present.
- The set of features represented by ϕ_j is defined by weights f_{ij} which encode the probability that feature i is an instance of ϕ_j.
- Trick: It's easier to adapt weights in an unbounded space, so use the transformation:

$$f = 1/(1 + \exp(-w))$$

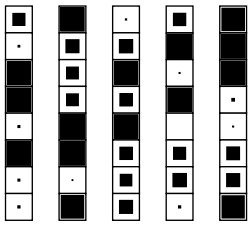
- optimize in w-space.



The data: a set of stochastic binary patterns



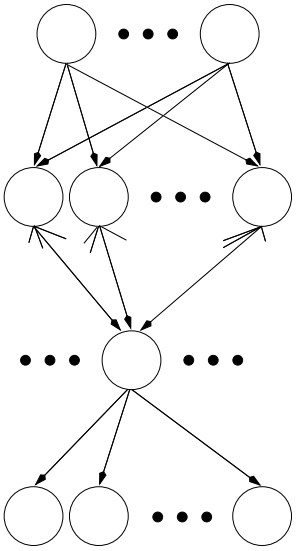
Each column is a distinct eight-dimensional binary feature.



true hidden causes of the data

Hierarchical Statistical Models

A Bayesian belief network:



The joint probability of binary states is

$$P(\mathbf{S}|\mathbf{W}) = \prod_i P(S_i|\mathrm{pa}(S_i), \mathbf{W})$$

The probability S_i depends only on its parents:

$$P(S_i|\mathrm{pa}(S_i), \mathbf{W}) = \begin{cases} h(\sum_j S_j w_{ji}) & \text{if } S_i = 1 \\ 1 - h(\sum_j S_j w_{ji}) & \text{if } S_i = 0 \end{cases}$$

The function h specifies how causes are combined, $h(u) = 1 - \exp(-u)$, $u > 0$.

Main points:

- hierarchical structure allows model to form high order representations
- upper states are priors for lower states
- weights encode higher order features

Inferring the best representation of the observed variables

- Given on the input **D**, the is no simple way to determine which states are the input's most likely causes.
  - Computing the most probable network state is an *inference* process
  - we want to find the explanation of the data with highest probability
  - this can be done efficiently with *Gibbs sampling*
- Gibbs sampling is another example of an MCMC method
- Key idea:

The samples are guaranteed to converge to the

true posterior probability distribution

Gibbs Sampling

Gibbs sampling is a way to select an ensemble of states that are representative of the posterior distribution $P(\mathbf{S}|\mathbf{D}, \mathbf{W})$.

- Each state of the network is updated iteratively according to the probability of S_i given the remaining states.
- this conditional probability can be computed using

$$P(S_i = a|S_j : j \neq i, \mathbf{W}) \propto P(S_i = a|\mathrm{pa}(S_i), \mathbf{W}) \prod_{j \in \mathrm{ch}(S_i)} P(S_j|\mathrm{pa}(S_j), S_i = a, \mathbf{W})$$

- limiting ensemble of states will be typical samples from $P(\mathbf{S}|\mathbf{D}, \mathbf{W})$
- also works if any subset of states are fixed and the rest are sampled

The Gibbs sampling equations (derivation omitted)

The probability of S_i changing state given the remaining states is

$$P(S_i = 1 - S_i|S_j : j \neq i, \mathbf{W}) = \frac{1}{1 + \exp(-\Delta x_i)}$$

Δx_i indicates how much changing the state S_i changes the probability of the whole network state

$$\begin{aligned}\Delta x_i = &\log h(u_i; 1 - S_i) - \log h(u_i; S_i) \\ &+ \sum_{j \in \mathrm{ch}(S_i)} \log h(u_j + \delta_{ij}; S_j) - \log h(u_j; S_j)\end{aligned}$$

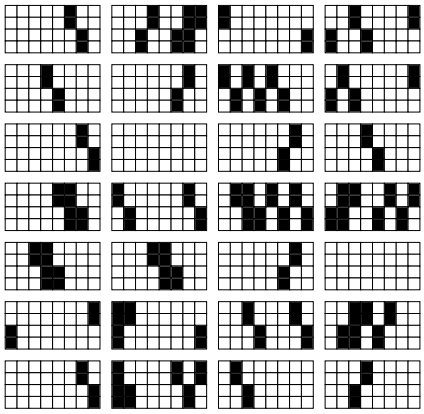
- u_i is the causal input to S_i, $u_i = \sum_k S_k w_{ki}$
- δ_j specifies the change in u_j for a change in S_i, $\delta_{ij} = +S_j w_{ij}$ if $S_i = 0$, or $-S_j w_{ij}$ if $S_i = 1$

Interpretation of Gibbs sampling equation

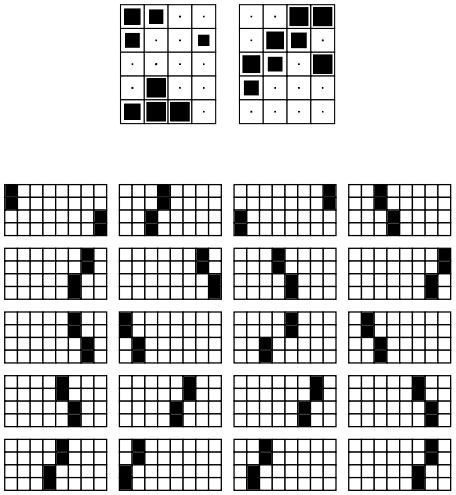
Gibbs equation can be interpreted as: *feedback* $+ \sum$ *feedforward*.

- *feedback*: how consistent is S_i with current causes?
- $\sum$ *feedforward*: how likely is S_i a cause of its children?
- feedback allows the lower level units to use information only computable at higher levels
- feedback determines state when the feedforward input is ambiguous

The Shifter Problem

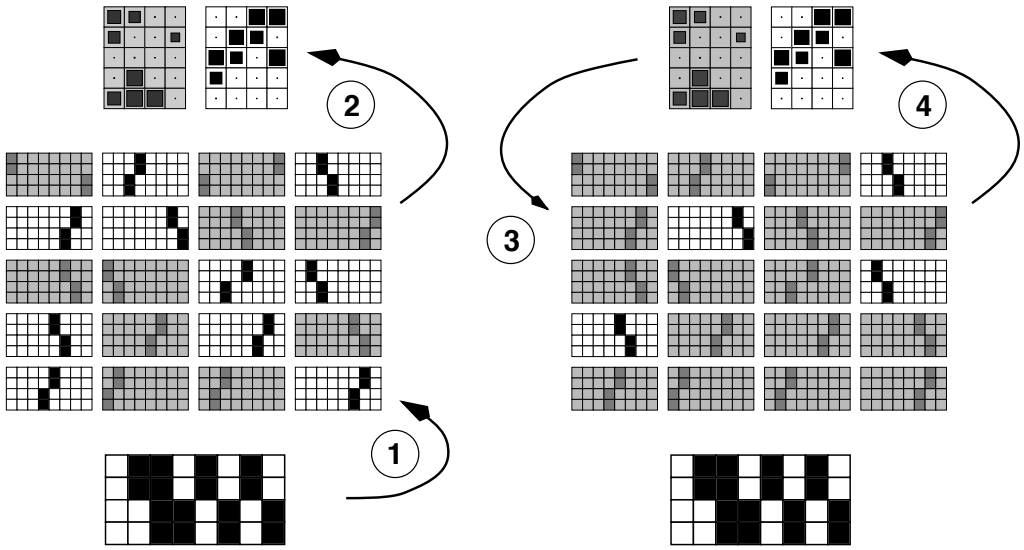


Shift patterns



weights of a 32-20-2 network

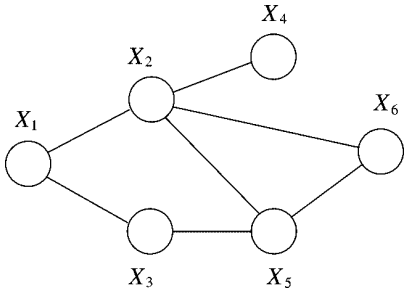
Gibbs sampling: feedback disambiguates lower-level states



One the structure learned, the Gibbs updating convergences in two sweeps.

Undirected graphical models: Markov nets

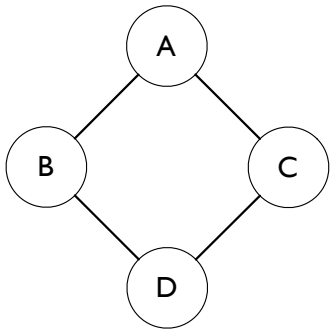
- *Undirected graphical models* are a different way of factorizing a joint probability density.
- Unlike Bayesian belief networks, undirected graphical models are useful when there is no intrinsic causality, e.g. relationships among pixels in images, language models.
- Absence of a links between nodes indicates *independence* of those two variables.
- This general approach is to represent the structure of a joint probability distribution in terms of independent factors represented by *potential functions*.



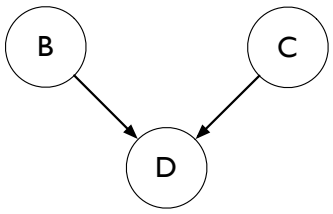
$$p(x_{\mathcal{V}}) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)$$

Directed and undirected graphical models

- DGs and UGs define different dependence relationships.
- There are families of probability distributions that are captured by DGs but not any UG and vice versa.



No directed graph can represent only:
$A \perp D \mid \{B,C\}$
$B \perp C \mid \{A,D\}$



No undirected graph can represent only:
$B \perp C$

Why?

How do we parameterize the relationships?

- Why can't we use a simple conditional parameterization, where the joint probability is a product of the conditional probability of each node given its neighbors:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_i p(\mathbf{x}_i | \mathrm{neigh}(\mathbf{x}_i))$$

- This is a product of functions, which factorizes the distribution, but...
- Multiplying conditional densities does not, in general, yield valid joint probability distributions.
- What about products of marginals?

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_i p(\mathbf{x}_i, \mathrm{neigh}(\mathbf{x}_i))$$

- This too does not yield valid probability distributions.
- Only directed graphs have this property because p(a,b) = p(a|b) p(b).
- Therefore we assume for undirected graphs that the joint distribution factorizes into arbitrarily defined *potential* functions, $\Psi(x)$.

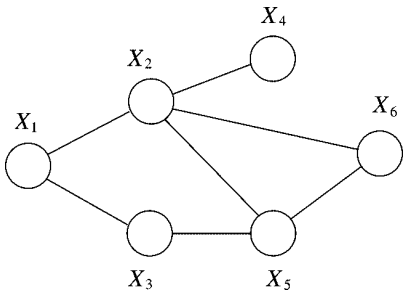
The joint pdf for a Markov net is a product of potential functions

- The joint probability is a product of *clique* potential functions:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{1}{Z} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

- Where each $\psi_c(\mathbf{x}_c)$ is an arbitrary positive function of it's arguments.
- The set of cliques is the set of maximal complete subgraphs.
- Z is a normalization constant that defines a valid joint pdf, and is sometimes called the *partition function*.

$$Z = \sum_{\mathbf{x}} \prod_{\text{cliques } c} \psi_c(\mathbf{x}_c)$$

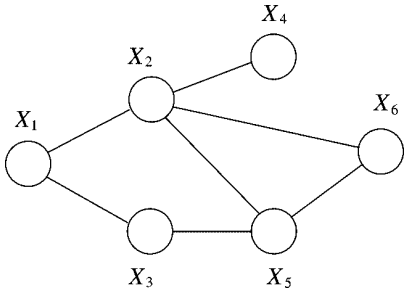


This is a greatly reduced representation.

$$p(x_{\mathcal{V}}) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)$$

Multiple definitions are possible

- It is not necessary to restrict the definition to maximal cliques
- The following definition is also valid:



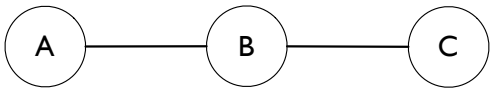
$$p(X) = \frac{1}{Z}\psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5)\psi(x_2, x_6)\psi(x_5, x_6)$$

- Here, we have assumed a factorization in terms of 2D densities.
- Why can we do this?

This is equivalent to assuming that $\psi(x_2, x_5, x_6)$ factors.

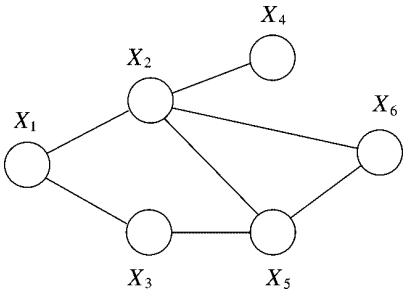
What are the clique potential functions?

- Consider the following model:



- The model specifies that A ⊥ C | B.
- The joint distribution can then be written as
  - p(A,B,C) = p(B) p(A|B) p(C|B)
- This can be written in two ways:
  - p(A,B,C) = p(A,B) p(C|B) = ψ_1(A,B) ψ_2(B,C)
  - p(A,B,C) = p(A|B) p(B,C) = ψ_3(A,B) ψ_4(B,C)
- This shows that the potential functions cannot both be marginals or both be conditionals.
- In general, the clique potential functions *do not* represent probability distributions. They are simply factors in the joint pdf.

Exact inference on undirected graphs by elimination



- How do we compute a marginal distribution, eg $p(x_1)$, on an undirected graph?
- Want to compute marginal, $p(x_1)$, sum over remaining vars:

$$p(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\frac{1}{Z}\psi(x_1,x_2)\psi(x_1,x_3) \cdot \psi(x_2,x_4)\psi(x_3,x_5) \cdot \psi(x_2,x_5,x_6).$$

- Like before, the naive sum is r^6, but we can push the sums in the products to obtain:

$$p(x_1) = \frac{1}{Z}\sum_{x_2}\psi(x_1,x_2)\sum_{x_3}\psi(x_1,x_3)\sum_{x_4}\psi(x_2,x_4) \cdot \sum_{x_5}\psi(x_3,x_5)\sum_{x_6}\psi(x_2,x_5,x_6)$$

Next time

- Markov decision processes