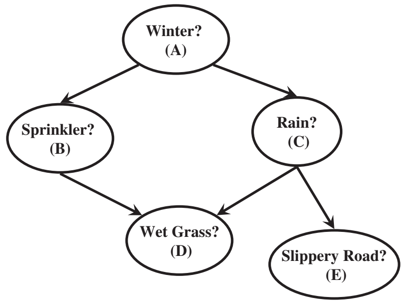
Inference in Bayesian networks

Outline

◇ Exact inference by variable elimination

◇ Approximate inference by stochastic simulation

A Bayesian Network



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Inference in BNs

$$Pr(D, E) \ ?$$

- Constructing the joint probability distribution and then summing out variables A, B, and C

$$Pr(d, e) = \sum_{a,b,c} Pr(a, b, c, d, e)$$

- One can sum out variables without having to construct the joint probability distribution explicitly.

$$Pr(d, e) = \sum_{a,b,c} \theta_{e|c}\theta_{d|bc}\theta_{c|a}\theta_{b|a}\theta_a$$

Variable Elimination

- Avoid recomputation

$$Pr(d,e) = \sum_c \theta_{e|c} \sum_b \theta_{d|bc} \sum_a \theta_{c|a}\theta_{b|a}\theta_a$$

- Variable elimination: carry out summations right-to-left, storing intermediate results (as tables) to avoid recomputation: e.g. $\sum_a \theta_{c|a}\theta_{b|a}\theta_a$ need to be computed for each d, e value; $\theta_{b|a}\theta_a$ is computed for each value of c

$$Pr(D,E) = \sum_C \Theta_{E|C} \sum_B \Theta_{D|BC} \sum_A \Theta_{C|A}\Theta_{B|A}\Theta_A$$

Tables

Definition A **(probability) table** T over variables X is a function which maps each instantiation x of variables X to a non-negative number, denoted $T(x)$.

B	C	D	T_1
true	true	true	$.95$
true	true	false	$.05$
true	false	true	$.9$
true	false	false	$.1$
false	true	true	$.8$
false	true	false	$.2$
false	false	true	0
false	false	false	1

D	E	T_2
true	true	0.448
true	false	0.192
false	true	0.112
false	false	0.248

Table Operations

Definition Let T be a probability table over variables S and let X be a variable in S. The result of **summing out** variable X from table T is another table over variables $Y = S - \{X\}$, which is denoted by $\sum_X T$ and defined as follows:

$$(\sum_X T)(y) = \sum_x T(y, x)$$

Summing out any number of variables from a probability table T can be done in time which is linear in the size of table T

Table Operations

Definition The result of **multiplying** tables $T_1(X)$ and $T_2(Y)$ is another table over variables $Z = X \cup Y$, which is denoted by T_1T_2 and defined as follows:

$$(T_1T_2)(z) = T_1(x)T_2(y)$$

where x and y are consistent with z.

E.g., $F_1(a, b) \times F_2(b, c) = F(a, b, c)$

The complexity of table multiplication is linear in the size of resulting table, $O(d^{|Z|})$

Variable Elimination

$$Pr(D,E) = \sum_{A,B,C} \Theta_{E|C}\Theta_{D|BC}\Theta_{C|A}\Theta_{B|A}\Theta_{A}$$

Theorem If T_1 and T_2 are tables, and if variable X appears only in T_2, then

$$\sum_X T_1 T_2 = T_1 \sum_X T_2$$

$$\begin{aligned} Pr(D,E) &= \sum_C \Theta_{E|C} \sum_B \Theta_{D|BC} \sum_A \Theta_{C|A}\Theta_{B|A}\Theta_A \\ &= \sum_C \Theta_{E|C} \sum_B \Theta_{D|BC} F_1(B,C) \\ &= \sum_C \Theta_{E|C} F_2(C,D) \end{aligned}$$

Variable Elimination Algorithm

Algorithm 1 VE-PR-I($\mathcal{N}$: a Bayesian network, $\mathbf{Q}$: some variables in network $\mathcal{N}$, π: an ordering of the n variables not in $\mathbf{Q}$): returns the prior marginal $\mathsf{Pr}(\mathbf{Q})$.

```
S ← CPTs of network N
for i = 1 to n do
    T ← ∏_k T_k, where T_k belongs to S and mentions variable π(i)
    T_i ← ∑_{π(i)} T
    S ← S − {T_k} ∪ {T_i}
Return ∏_{T∈S} T
```

- Complexity: linear in the size of constructed table T_i
- **Width** w of the order π: the number of variables in the largest table we ever construct
- The time and space complexity is $O(nd^w)$
- The elimination order is significant computationally.
- Computing an optimal order is an NP-hard problem. Use heuristics

Computing Posterior Marginals

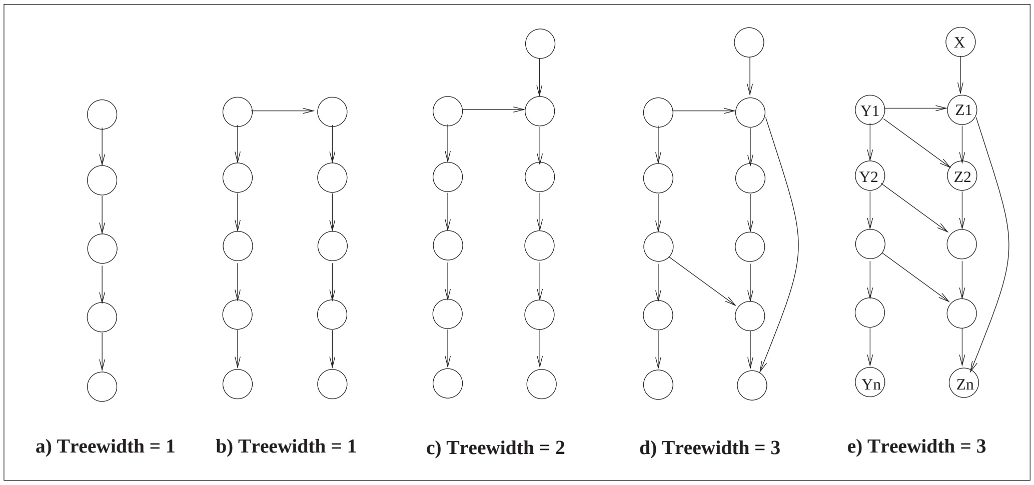
- $Pr(Q|e)$?
- Compute joint marginals $Pr(Q, e)$
- We can compute the posterior marginal $Pr(Q|e)$ by simply normalizing the corresponding joint marginal $Pr(Q, e)$
- **zeroing out**: given table $T(X)$ and evidence e, a table T^e is obtained by replacing the number $T(x)$ by a zero for every instantiation x that is inconsistent with evidence e
- We will omit the zeroed out rows

$$Pr(D, E, e) = \sum_{A,B,C} \Theta^e_{E|C}\Theta^e_{D|BC}\Theta^e_{C|A}\Theta^e_{B|A}\Theta^e_{A}$$

- Computing the probability of evidence $Pr(e)$ by eliminating all variables

Complexity of exact inference

tree-width of the network: the width of the best elimination order (eliminating all variables)



The Polytree Algorithm

- **Tree networks**: each node has at most one parent. The treewidth is 1.
- **Polytree networks (singly-connected networks)**: there is at most one undirected path between any two nodes. The treewidth is the maximum number of parents that any node may have.
- The polytree algorithm
  - Message passing algorithm (aka belief propagation algorithm)
  - Time and space complexity is linear in the size of the network $O(nd^k)$, where k is the maximum number of parents

Complexity of exact inference

- Multiply connected networks: NP-hard
  - The number of nodes has no genuine effect on treewidth.
  - The number of parents per node has a direct effect on treewidth. If the number of parents per node is k, treewidth is no less than k.
  - Cycles have a genuine effect on treewidth.

Cutset Conditioning

- In general, we can compute the query $Pr(Q, e)$ as
$$Pr(q, e) = \sum_c Pr(q, e, c)$$
- We choose the nodes in C so that deleting their outgoing edges leads to a singly-connected network
- A set of nodes C which satisfies the above property is known as a loop-cutset (cycle cutset) for the network.
- The method is known as loop-cutset conditioning.

Cutset Conditioning

- Given a loop cutset C of size s, the method of loop-cutset conditioning requires $O(d^s)$ invocations to the polytree algorithm, each taking $O(nd^k)$ time
- Therefore, loop-cutset conditioning takes $O(nd^{k+s})$ time, which is exponential in the size of used cutset.
- Computing a loop-cutset of minimal size is hence an important task in the context of cutset conditioning, but such computation is known to be NP-hard.
- The space complexity of loop-cutset conditioning is only $O(nd^k)$, which is clearly not exponential in the cutset size and is quite important as the variable elimination and jointree algorithms have time and space complexities which are both exponential in the treewidth.

Jointree algorithm

- Suppose that our goal is to compute the posterior marginal for each network variable given evidence e
- We can run variable elimination $O(n)$ times: the total complexity will be $O(n^2 d^w)$
- We can compute all of the above marginals in only $O(nd^w)$ time and space using the **jointree algorithm (junction tree, clique-tree algorithm, or tree-clustering algorithm)**
  - Join individual nodes to form cluster nodes in such a way that the resulting network is a tree $\rightarrow$ jointree
  - Message passing algorithm
  - Can compute all $P(X_i, PA_i|e)$ marginals simultaneously in $O(nd^w)$ time and space

Inference in BN

- Exact inference is NP-hard
  - Variable elimination
  - Jointree
  - Cutset Conditioning
  - Recursive conditioning
  - Inference with Local Structures
- Approximate inference is NP-hard
  - Stochastic sampling

Inference by stochastic simulation

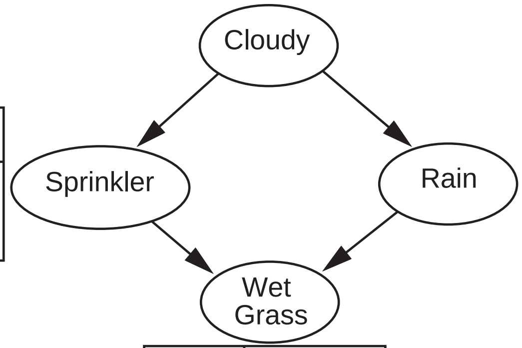
- **Approximate inference algorithms** based on the principle of stochastic sampling
- Draw random samples according to a given distribution, and then estimate the probabilities of events based on the frequency of their occurrence in the samples.
- Accuracy of the results depends on the size of the sample
- Produce samples for a binary variable X: choosing a random number r in the interval $[0, 1)$, and choosing the value 0 if $r < Pr(0)$, and the value 1 otherwise.
- If a variable X has multiple values $x_1, \ldots, x_m$, we can generate a sample from $Pr(X)$ as follows: generate a random number p in the interval $[0, 1)$, and then choose the value x_k if $\sum_{i=1}^{k-1} Pr(x_i) \leq p < \sum_{i=1}^{k} Pr(x_i)$

Inference by stochastic simulation

- How to produce samples from a distribution that is induced by a Bayesian network?
- Traverse the network in topological order, visiting parents before children, and generating a value for each visited node according to the probability of that node

Example

P(C)
.50



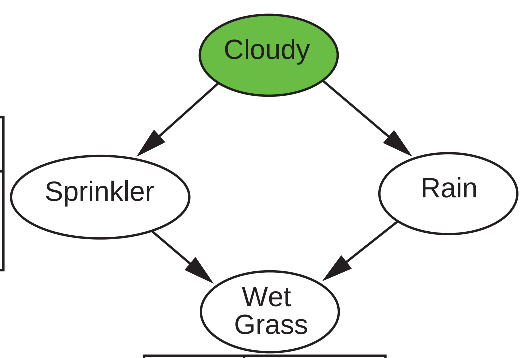
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



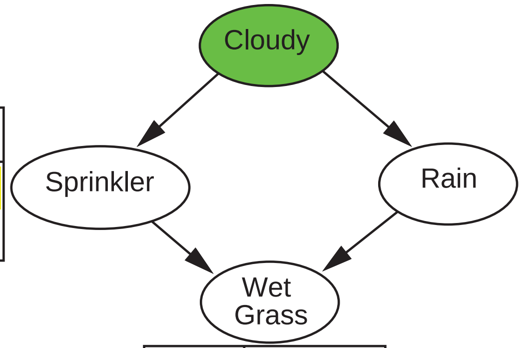
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



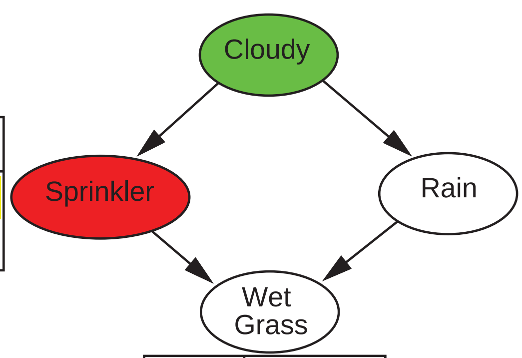
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



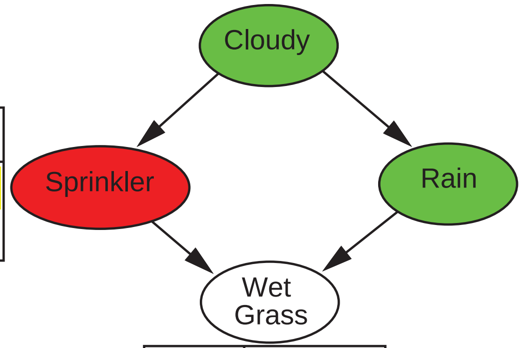
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



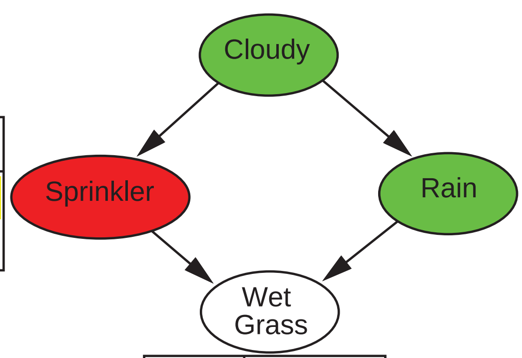
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



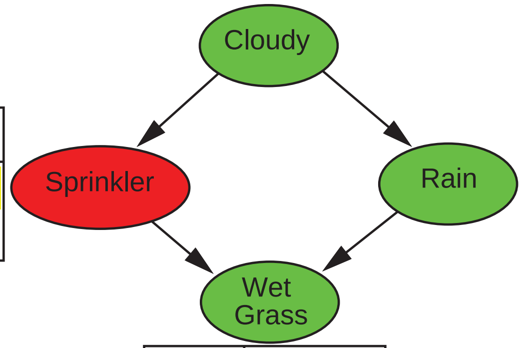
C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Example

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Inference by stochastic simulation

◇ Monte Carlo approximation: the probability is approximated using sample frequencies

$$P(S = T) = N_{S=T}/N$$

◇ How to provide guarantees on the quality of estimates?

◇ Estimate $P(X|e)$? What if $P(e)$ is small?

Inference in BN

- Exact inference is NP-hard
  - Variable elimination
  - Jointree
  - Cutset Conditioning
  - Recursive conditioning
  - Inference with Local Structures
- Approximate inference is NP-hard
  - Stochastic sampling: importance sampling, Markov chain Monte Carlo (MCMC)
  - Loopy belief propagation
  - Variational methods

Software Packages for BNs

- SamIam from UCLA
  http://reasoning.cs.ucla.edu/samiam/
- Graphical Model Algorithms at UC Irvine
  http://graphmod.ics.uci.edu/group
- GeNIe/SMILE from the University of UPitt
  http://genie.sis.pitt.edu/
- Hugin lite from Hugin:
  http://www.hugin.com
- **Software Packages for Graphical Models / Bayesian Networks**
  http://www.cs.ubc.ca/∼murphyk/Software/bnsoft.html

Other Probabilistic Graphical Models

- Markov networks/Markov Random Fields: use undirected graphs
- Dynamic Bayesian networks: probabilistic reasoning over time
- Probabilistic Relational Models: BN over relational database

Combining the expressive power of first-order logic with probability theory and graphical models? $\rightarrow$ Statistical relational learning (SRL)

SRL Alphabet Soup