Spezielle Themen der Künstlichen Intelligenz

10. Termin:

Bayesian Networks Inference Algorithms

Modeling with Bayesian networks

Using Bayesian networks for real-world problems requires two steps:

- constructing an appropriate Bayesian network
- solve the problems by applying one of the possible queries

How to construct a Bayesian network?

1. define network variables and their values
   - distinguish between *query*, *evidence*, and *intermediary* variables
2. define network structure
   - for each var *X*, answer the question: what set of var's are direct causes of *X*?
3. define network parameters (CPTs)
   - difficulty and objectivity depend on problem

Reasoning with Bayesian networks

Four general types of queries one can pose:

- probability of evidence: how likely is a variable instantiation **e** ➔ $Pr(\mathbf{e})=?$
- prior and posterior marginals: how probable is an instantiation of a limited set of variables ➔ $Pr(x_1,...,x_m)=?$ or $Pr(x_1,...,x_m|\mathbf{e})=?$
- most probable explanation (MPE): what is the most probable instantiation of all network var's given some evidence **e** ➔ **x** *with* $Pr(x_1,...,x_n|\mathbf{e})=max?$
- maximum a posteriori hypothesis (MAP): what is the most probable instantiation of a subset of var's given some evidence **e** ➔ **x** *with* $Pr(x_1,...,x_m|\mathbf{e})=max?$

All these queries can be computed from a Bayesian network.

But, how?

Inference algorithms

- variable elimination (see last lecture)
- jointtree algorithm
- recursive conditioning
- belief propagation
- Monte Carlo Markov Chain

Variable elimination

Variable elimination:

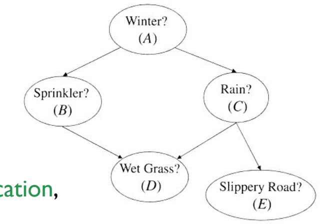
- given a distribution *Pr(A,B,C,D,E)*, variable *A* with values a_i can be summed out by

$$Pr(B,C,D,E) = \sum_{a_i} Pr(a_i, B, C, D, E)$$

Definition: factor f over var's **X** is a function that maps each instantiation **x** of **X** to a number f(**x**)≥0

- can represent any marginal or conditional distribution
- Summing out a variable from a factor: $(\sum_X f)(\mathbf{y}) := \sum_x f(x, \mathbf{y})$
  - marginalizing *X*, projecting on *Y*
- Multiplying factors: $(f_1 f_2)(\mathbf{z}) := f_1(\mathbf{x}) f_2(\mathbf{y})$ with $\mathbf{x} \sim \mathbf{z}, \mathbf{y} \sim \mathbf{z}$

Variable elimination



Use for computing prior marginals:

- express joint distribution as factor multiplication, viewing CPTs as factors, e.g. *Pr(a,b,c,d,e):*

$$Pr(a,b,c,d,e) = \Theta_{E|C}\Theta_{D|BC}\Theta_{C|A}\Theta_{B|A}\Theta_A$$

- compute marginal distribution by summing out variables from these factors, e.g. *Pr(D,E):*

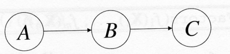
$$Pr(D,E) = \sum_{A,B,C} \Theta_{E|C}\Theta_{D|BC}\Theta_{C|A}\Theta_{B|A}\Theta_A$$

Can (and should) be simplified according to:

$$\sum_X f_1 f_2 = f_1 \sum_X f_2 \text{ if X appears only in } f_2$$

Example: compute prior marginal *Pr(C)* by eliminating first *A*, then *B*

- $Pr(C) = \underbrace{\sum_B \Theta_{C|B} \underbrace{\sum_A \Theta_A \Theta_{B|A}}}$



A	Θ_A
true	.6
false	.4

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

B	C	$\Theta_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

A	B	$\Theta_A\Theta_{B\|A}$
true	true	.54
true	false	.06
false	true	.08
false	false	.32

B	$\sum_A \Theta_A\Theta_{B\|A}$
true	$.62 = .54 + .08$
false	$.38 = .06 + .32$

B	C	$\Theta_{C\|B}\sum_A \Theta_A\Theta_{B\|A}$
true	true	.186
true	false	.434
false	true	.190
false	false	.190

C	$\sum_B \Theta_{C\|B}\sum_A \Theta_A\Theta_{B\|A}$
true	.376
false	.624

Order of elimination irrelevant for result, but not for computational costs !!

Other possibility: first *B* then *A*

- $Pr(C) = \sum_A \Theta_A \underbrace{\sum_B \Theta_{B|A}\Theta_{C|B}}$

largest factor with 2 variables

Best order: smallest width = number of var's in the largest factor constructed

- can be computed offline, but NP-hard
- heuristics

Variable elimination

Use for computing posterior marginals:

- need to computer the factor *Pr(**Q**|**e**)*

Better to compute joint marginal *Pr(**Q**, **e**)* and normalize to get *Pr(**Q**|**e**)*

- also gives *Pr(**e**)* for free as $Pr(\mathrm{e}) = \sum_q Pr(\mathrm{q}, \mathrm{e})$

Example:

D	*E*	*Pr(**Q**|**e**)*
true	true	.448
true	false	.192
false	true	.112
false	false	.248

D	*E*	*Pr(**Q**,**e**)*
true	true	.21504
true	false	.09216
false	true	.05376
false	false	.11904

=.11904 / .48

*∑=.48 Pr(**e**)*

Variable elimination

Use elimination for computing joint marginals:

- zero out all rows that are not compatible with evidence **e**

Definition: reduction of factor *f*(**X**) given evidence **e** is another factor over **X** denoted by f^e, defined by

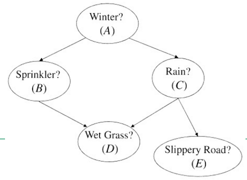
$$f^e(\mathbf{x}) := \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \sim \mathbf{e} \\ 0 & \text{otherwise} \end{cases}$$

- distributivity with factor multiplication: $(f_1 f_2)^{\mathbf{e}} = f_1^{\mathbf{e}} f_2^{\mathbf{e}}$

Joint marginal *Pr*(**Q**,**e**) can hence be computed as follows:

- *Example*: **Q**={*D*,*E*}

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

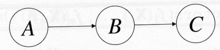


Example: compute posterior marginal *Pr*(**Q**={*C*},**e**:*A*=*true*) by eliminating first *A*, then *B*

- $Pr(\mathbf{Q},\mathbf{e}) = \sum_B \sum_A \Theta^e_A \Theta^e_{B|A} \Theta^e_{C|B}$
$$= \sum_B \Theta^e_{C|B} \Theta^e_A \Theta^e_{B|A}$$

Therefore:

- *Pr*(*C*=*true*,*A*=*true*)=.192
- *Pr*(*C*=*false*,*A*=*true*)=.408
- *Pr*(*A*=*true*)=.6
- *Pr*(*C*=*true*|*A*=*true*)=.192/.6=.32



A	Θ_A
true	.6
false	.4

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

B	C	$\Theta_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

A	Θ^e_A
true	.6

A	B	$\Theta^e_{B\|A}$
true	true	.9
true	false	.1

B	C	$\Theta^e_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

A	B	$\Theta^e_A\Theta^e_{B\|A}$
true	true	.54
true	false	.06

B	$\sum_A \Theta^e_A\Theta^e_{B\|A}$
true	.54
false	.06

B	C	$\Theta^e_{C\|B}\sum_A \Theta^e_A\Theta^e_{B\|A}$
true	true	.162
true	false	.378
false	true	.030
false	false	.030

C	$\sum_B \Theta^e_{C\|B}\sum_A \Theta^e_A\Theta^e_{B\|A}$
true	.192
false	.408

Variable elimination

Summary

- combines factor multiplication, reduction & projecting
- useful for computing prior and posterior marginals and probability of evidence (joint marginals)
- structure-based algorithm, performance depends on network structure (e.g. no. parents per node, loops, paths between nodes)

Usually combined with pruning for a given query *Pr*(**Q**,**e**)

- *iteratively* remove any leaf node *not* in **Q** or **E**
- remove edge $U \rightarrow X$ from any node U in E and $\Theta_{X|\mathbf{U}} \leftarrow \sum_U \Theta^u_{X|\mathbf{U}}$

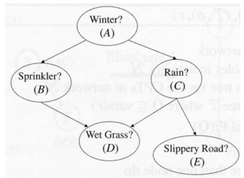
Factor elimination

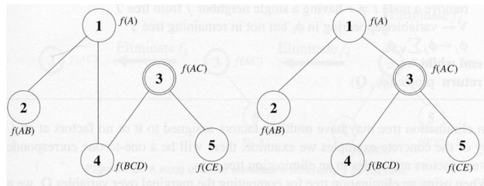
Generalization of variable elimination to factor elimination, i.e. elimination of *sets of variables* (Lauritzen & Spiegelhalter 1988)

- elimination order ➔ elimination trees

Definition: An elimination tree (T, Θ) for a set of factors ***S*** is a tree T, in which each node Θ_i is assigned exactly one factor in ***S***

- factors are the CPTs in the Bayesian network
- different tree structures are possible:





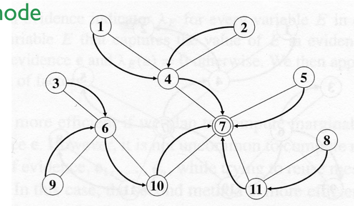
Factor elimination

Factor elimination in elimination trees:

- eliminate a node (factor) if all its neighbors, except the one closer to the root, have been eliminated
- when a node *i* is about to be eliminated, it will have a single neighbor *j* and *i*'s factor is projected and multiplied into factor of *j*
  - viewed as passing a „message" from *i* to *j*

Using factor elimination for computing marginal over **Q**

- pick one node *r* with **Q** ⊆ *vars(r)* as root node
- push messages toward the root
- when all messages are available in root, multiply with factor *r* and project to **Q**



Jointtree algorithm

Definition: A jointtree $(T, \mathbf{C})$ for a DAG G is a tree T in which each node has a label $\mathbf{C}_i$ (called cluster), satisfying the properties:

- each cluster is a set of nodes from G
- each *family** in G appears in some cluster
- if a node appears in two clusters $\mathbf{C}_i, \mathbf{C}_j$, it must appear in *every* cluster on the path connecting nodes *i* and *j* in the jointtree

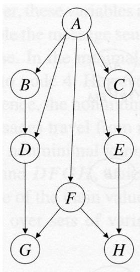
the separator of edge *i-j* is defined as $S_{ij} := C_i \cap C_j$

**family = a node along with its parents*

Also known as junction trees, clique trees, Markov trees, hypertrees

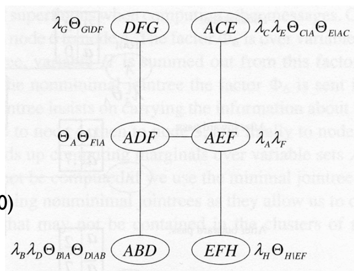
An evidence indicator is a factor over variable *X* that captures the value of *X* in evidence **e**: $\lambda_X(x) = 1 \text{ if x consistent with } \mathbf{e}, 0 \text{ otherwise}$

Jointtree algorithm



Algorithm:

- construct jointtree for a given Bayesian network
- assign each CPT $\Theta_{X|\mathbf{U}}$ to a cluster that contains X and **U**
- assign each evidence indicator λ_X to a cluster that contains *X*
- select a root node that contains the query **Q**
- start eliminating factors (using projecting and multiplying) inwards/outwards*
- finally project cluster in the root node onto **Q**



**different propagation strategies with different space and time complexities*

- Shenoy-Shafer architecture (Shenoy & Shafer 1990)
- Hugin architecture (Jensen et al. 1990)

Recursive conditioning

Idea: simplify a problem by solving a number of cases and combining the results to a solution to the original problem (case analysis)

$$Pr(x) = \sum_c Pr(x, c)$$

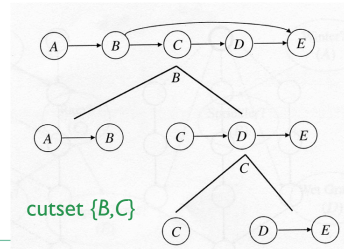
Approach: reduce query on a network into a queries on simpler networks

- if var *E* given as evidence, the network can be pruned (see above)
- *in general*: any query *Pr*(**q,e**) leads to decomposition into networks N_e^r and N_e^l such that

„cutset" **e**

$$Pr(q) = \sum_e Pr(q, e)$$

$$= \sum_e Pr_e^l(q^l, e^l) Pr_e^r(q^r, e^r)$$



Recursive conditioning

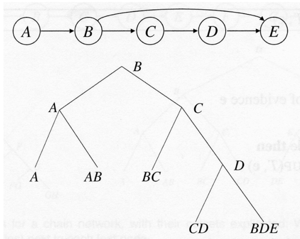
Recursive condition algorithm:

- decompose network in a divide-and-conquer fashion, following an appropriate cutset
- when at leaf node, look up the conditioned CPT
- propagate value back according to $\sum_e Pr_e^l(q^l, e^l) Pr_e^r(q^r, e^r)$

Question (again): what is an appropriate cutset (order)?

Answer: all are valid, some lead to less work

- need to minimize total number of considered cases
- use decomposition trees: full binary trees, leaves are CPTs in the network
- useful to employ caching techniques (Darwiche, chapt. 8)



Belief propagation

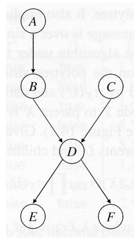
Approximative inference algorithms

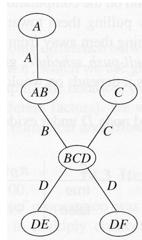
- originially for exact inference in polytree* networks, then generalized to approximative solution for arbitrary networks
- spectrum of approximations: trade-off quality with computational costs

*Belief propagation algorithm for computing joint marginals Pr(X,**e**):*

- identical to jointtree algorithm for jointtrees that coincide with the polytree network structure
- *Example*:

**polytree = network with only one path between any two nodes*





Belief propagation

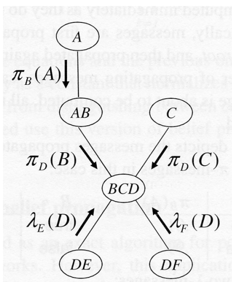
*Computing joint marginals Pr(X,**e**):*

- node *i* in jointtree has cluster $C_i = X\mathbf{U}$ (with $\mathbf{U}$ parents of *X*)
- edge *i-j* in jointtree corresp. to edge *X-Y* in network has „separator" $\mathbf{S}_{ij} = X$
- „messages" to eliminate factors:
  - from U to X: causal support $\pi_X(U)$
  - from Y to parent X: diagnostic support $\lambda_Y(X)$
- messages are sent by a node, when it has received messages from all other nodes
  - start with those that do not depend on others

Example:

- Belief propagation toward node *D*, evidence *E=true*

$$Pr(BCD, \mathbf{e}) = \Theta_{D|BC} \pi_D(B) \pi_D(C) \lambda_E(D) \lambda_F(D)$$



Belief propagation

problem: leads to „deadlocks" in some network, when each message is dependent on any other

solution: iterative belief propagation

- assume initial values to each message in the network
- propagate beliefs and re-iterate
- converge to a „fixed point" solution
  - may generally have mutiple fixed points on a given network
  - may oscillate on some networks, loop forever

Stochastic sampling

Idea: simulate an event according to some probability of occurrence, estimate the prob. of this event from its frequency in these simulations

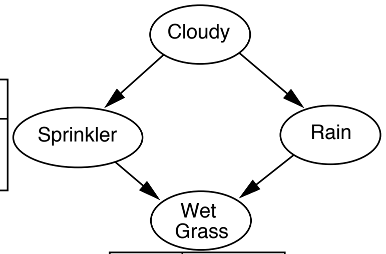
Simulating a Bayesian network:

A Bayesian network induces a distribution $Pr(\mathbf{X})$

- visit each node in topological order
- generate value for each node according to $Pr(x|\mathbf{u})$
- end with a sample $\{\mathbf{x}^1,\ldots,\mathbf{x}^n\}$ of n events
- estimate probability $\hat{Pr}(\mathbf{x})$ of value $\mathbf{x}$ from its frequency in this sample
- show that $\hat{Pr}(\mathbf{x})$ converges against $Pr(\mathbf{x})$ with increasing n

Example

P(C)
.50



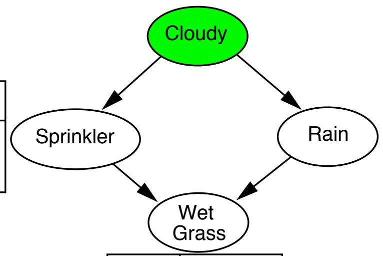
C	P(S|C)
T	.10
F	.50

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



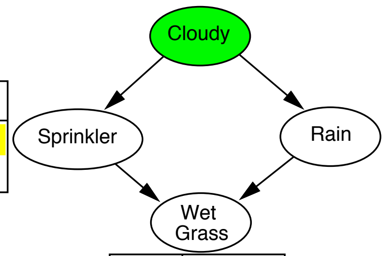
C	P(S|C)
T	.10
F	.50

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



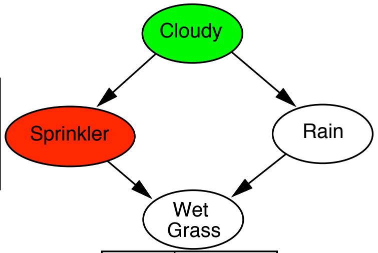
C	P(S|C)
T	.10
F	.50

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



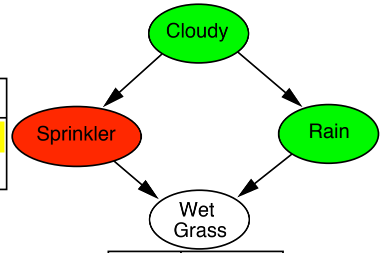
C	P(S|C)
T	.10
F	.50

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



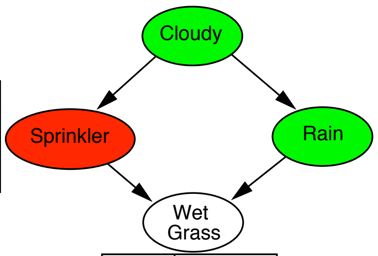
C	P(S|C)
T	.10
F	.50

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



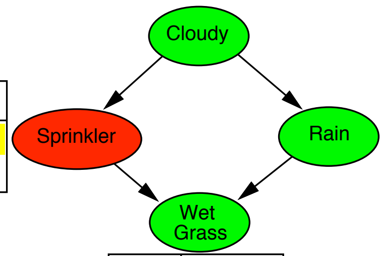
C	P(S|C)
T	.10
F	.50

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

returned event:
$\mathbf{x}^i$ = <T,F,T,T>

Stochastic sampling

Sampling relies on taking probability as expectation about a function

- expectation value of a function *f(**X**)*: $Ex(f) := \sum_x f(x) \cdot Pr(x)$
  μ
- variance of a function *f(**X**)*: $Var(f) := \sum_x (f(x) - Ex(f))^2 \cdot Pr(x)$
  σ^2

Direct sampling function:

- let $\hat{\alpha}(x) := 1$ if α true at $\mathbf{x}$, 0 otherwise
- then: $Ex(\hat{\alpha}) = Pr(\alpha)$

$$Var(\hat{\alpha}) = Pr(\alpha)Pr(\neg\alpha) = Pr(\alpha) - Pr(\alpha)^2$$

That is, approximating *Pr* boils down to estimating the expectation
How?

Monte Carlo simulation

Principle:

- simulate random sample $\mathbf{x}^1, ..., \mathbf{x}^n$ from sampling distribution *Pr(**X**)*
- evaluate function at each instantiation $f(\mathbf{x}^1), ..., f(\mathbf{x}^n)$
- compute arithemtic average of attained values: sample mean

$$Av_n(f) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}^i)$$

- works because of *law of large numbers*: for function *f* with expectation μ and every ϵ >0: $\lim_{n \to \inf} P(|Av_n(f) - \mu| \leq \epsilon) = 1$

Monte Carlo simulation using $\hat{\alpha}(x)$ gives direct sampling:

- simulate sample $\mathbf{x}^1,..., \mathbf{x}^2$ from Bayesian network
- compute values $\hat{\alpha}(x^1), ..., \hat{\alpha}(x^n)$
- estimate $Pr(\alpha)$ using sample mean $Av_n(\hat{\alpha})$

Rejection sampling

Goal: Calculate conditional prob. *Pr(a|b)* with *Pr(.)* induced by network

Approach:

- calculate estimate for *Pr(a∧b)* and *Pr(b)*: $Av_n(\hat{\gamma}), Av_n(\hat{\beta})$ with $\gamma = \alpha \wedge \beta$
- take ratio as estimate for *Pr(a|b)*: $Av_n(\hat{\gamma})/Av_n(\hat{\beta})$
  - *c_1=#samples with a∧b=true, c_2=#samples with b=true → (c_1/n)/(c_2/n)=c_1/c_2*
- reject all samples in which *b* is false: rejection sampling

Example: estimate *P(Rain|Sprinkler=true)* from 100 samples; 27 have *Sprinkler=true*, of these 8 have *Rain=true*, 19 have *Rain=false*

$$\hat{\mathbf{P}}(Rain|Sprinkler = true) = \text{NORMALIZE}(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle$$

True answer: <0.3,0.7>

Importance sampling

Idea: reduce variance due to rare events by sampling from an importance distribution *Pr*' emphasizing instantiations consistent with rare event

Monte Carlo simulation using the importance sampling function:

$$\tilde{\alpha}(x) = Pr(x)/Pr'(x) \text{ if } \alpha \text{ true at instantiation } x, 0 \text{ otherwise}$$

Improves on direct sampling only when *Pr*' emphasizes important events no less than *Pr*

Finding ideal distribution generally not feasible, but some other weaker conditions can be ensured easier and still improve on variance

Importance sampling

Likelihood weighting: given evidence **e**, what is *Pr(**x**|**e**)*?

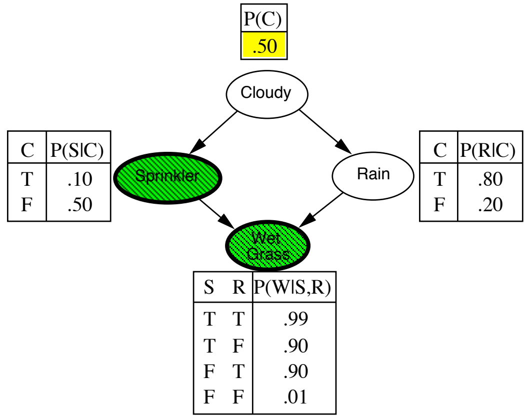
- generate only samples that are consistent with **e**
- fix evidence variables, sample non-evidence var‘s and weight sample by likelihood it accords the evidence
- consistent estimate, but performance gets worse with growing evidence because few samples have ~all total weight

function WEIGHTED-SAMPLE(bn, $\mathbf{e}$) **returns** an event and a weight

  $\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$
  for $i = 1$ **to** n **do**
    if X_i has a value x_i in $\mathbf{e}$
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$, w

Example

Query: **P**(Rain | Sprinkler=true, WetGrass=true) = ??



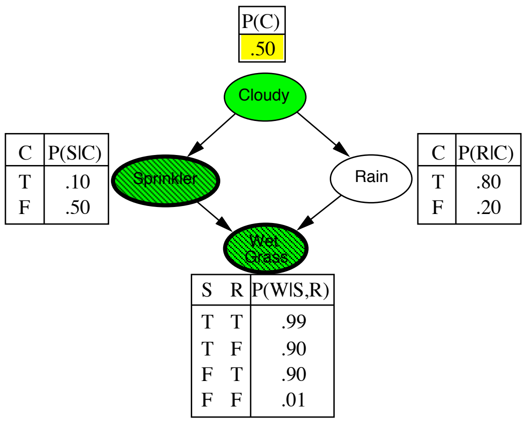
P(C)
.50

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

$w = 1.0$



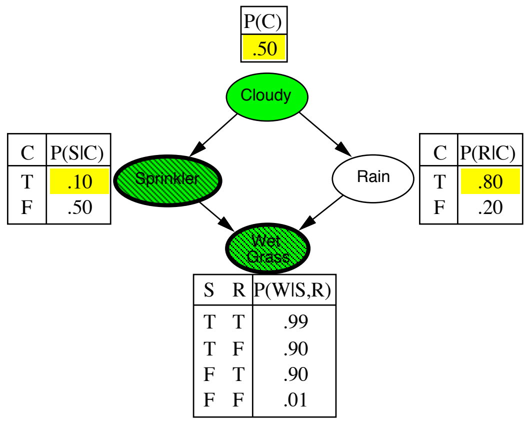
P(C)
.50

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

$w = 1.0$



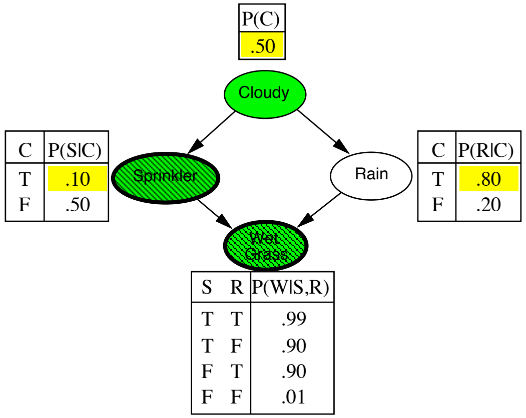
P(C)
.50

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

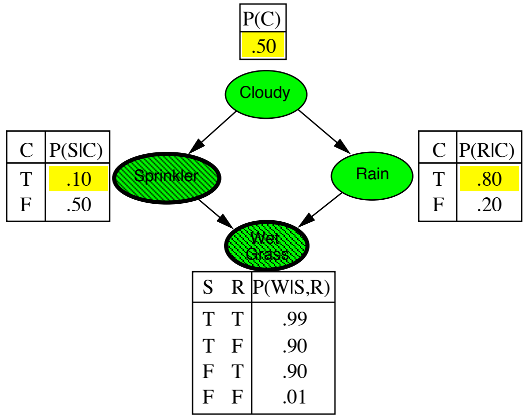
C	P(R|C)
T	.80
F	.20

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

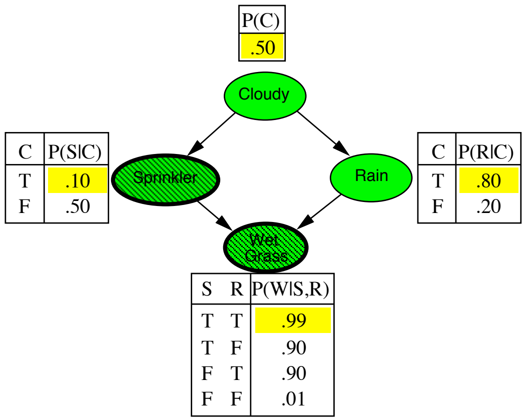
$w = 1.0$



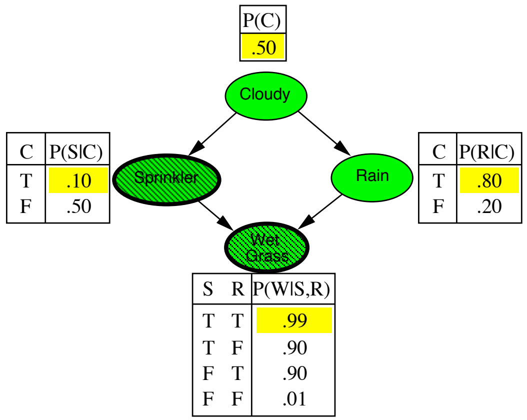
$w = 1.0 \times 0.1$



$w = 1.0 \times 0.1$



$w = 1.0 \times 0.1$



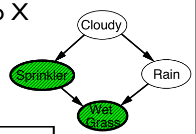
$w = 1.0 \times 0.1 \times 0.99 = 0.099$ = weight for event <t,t,t,t>

Markov Chain Monte Carlo (MCMC)

Network is in a state = current assignment to variables

next state: sample *non-evidence variable X* given its Markov blanket
= variables that, when known, make other variables irrelevant to X

- Markov blanket of *Cloudy* is *Sprinkler* and *Rain*
- Markov blanket of *Rain* is *Sprinkler*, *Cloudy*, *WetGrass*



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 $\mathbf{e}$

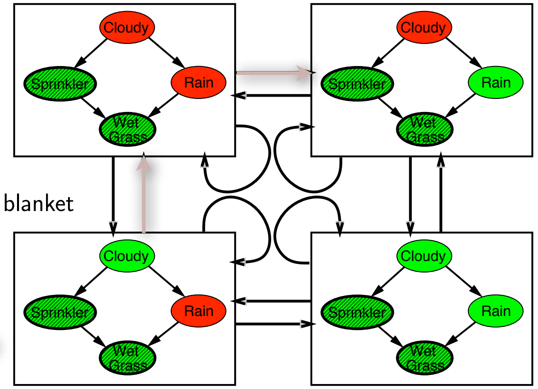
  initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Y}$
  for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
      sample the value of Z_i in $\mathbf{x}$ from $\mathbf{P}(Z_i|mb(Z_i))$
        given the values of $MB(Z_i)$ in $\mathbf{x}$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{N}[X]$)

„transition prob.“ of moving into new state

Estimate $\mathbf{P}(Rain|Sprinkler = true, WetGrass = true)$

Sample *Cloudy* given its Markov blanket

Initial state (random)



Count number of times *Rain* is true and false in the samples.

E.g.: visit 100 states: $\mathbf{P}(Rain|Sprinkler = true, WetGrass = true)$
$= \text{NORMALIZE}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$

MCMC - Markov blanket sampling

Because of the transition probabilites, sampling runs into an „equilibrium“ in which time spent in each state is proportional to its posterior probability

Transition probability (given the Markov blanket) is:

$$P(x_i'|mb(X_i)) = P(x_i'|parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$$

- easily implemented in parallel systems

Main difficulties:

- difficult to tell if and when convergence has been achieved
- can be wasteful if Markov blanket large, prob doesn‘t change much

Bayes nets inference algorithms - summary

Exact algorithms

- Variable Elimination and Factor Elimination
- Jointtree algorithm
- Recursive conditioning

Approximative algorithms

- Belief propagation
- Stochastic sampling (Monte Carlo simulation)
  - direct sampling
  - importance sampling, likelihood weighting
- Monte Carlo Markov Chain