Introduction to Artificial Intelligence

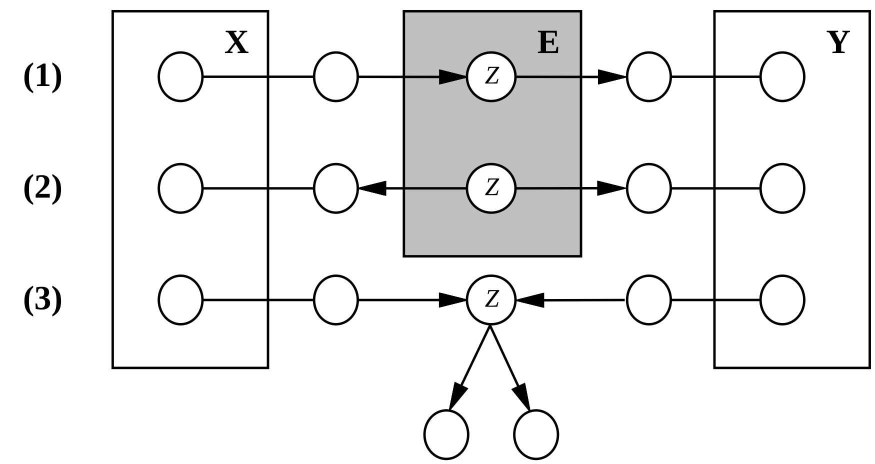
Inference in belief networks

Chapter 15.2–4 + new

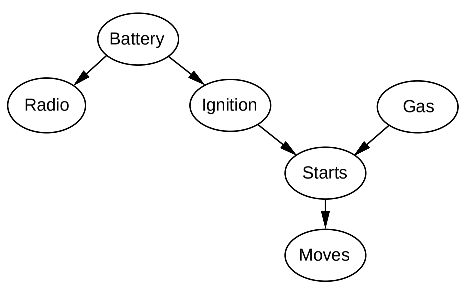
Dieter Fox

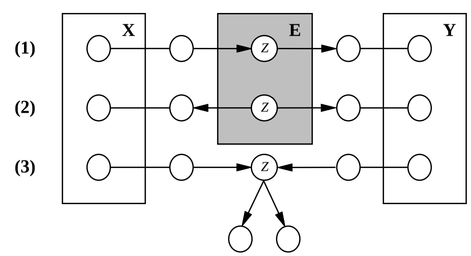
D-Separation

Nodes X are **independent** of nodes Y given E, when every undirected path from a node in X to a node in Y is d-separated by E.



Example





1. $E =$ *Ignition* d-separates *Gas* and *Radio*
2. $E =$ *Battery* d-separates *Gas* and *Radio*
3. *Gas* and *Radio* are independent given no evidence, but *Gas* and *Radio* are dependent given $E =$ *Starts* or $E =$ *Moves*.

Inference, outline

◊ Exact inference by enumeration

◊ Exact inference by variable elimination

◊ Approximate inference by stochastic simulation

Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$\mathbf{P}(B|J=true, M=true)$
$= \mathbf{P}(B, J=true, M=true)/P(J=true, M=true)$
$= \alpha\mathbf{P}(B, J=true, M=true)$
$= \alpha\sum_e\sum_a \mathbf{P}(B, e, a, J=true, M=true)$

Rewrite full joint entries using product of CPT entries:

$P(B=true|J=true, M=true)$
$= \alpha\sum_e\sum_a P(B=true)P(e)P(a|B=true, e)P(J=true|a)P(M=true|a)$
$= \alpha P(B=true)\sum_e P(e)\sum_a P(a|B=true, e)P(J=true|a)P(M=true|a)$

Enumeration algorithm

Exhaustive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

```
EnumerationAsk(X,e,bn) returns a distribution over X
inputs: X, the query variable
          e, evidence specified as an event
          bn, a belief network specifying joint distribution P(X_1,...,X_n)
   Q(X) ← a distribution over X
   for each value x_i of X do
        extend e with value x_i for X
        Q(x_i) ← EnumerateAll(Vars[bn],e)
   return Normalize(Q(X))
EnumerateAll(vars,e) returns a real number
   if Empty?(vars) then return 1.0
   else do
        Y ← First(vars)
        if Y has value y in e
            then return P(y | Pa(Y)) × EnumerateAll(Rest(vars),e)
            else return Σ_y P(y | Pa(Y)) × EnumerateAll(Rest(vars),e_y)
                where e_y is e extended with Y = y
```


Inference by variable elimination

Enumeration is inefficient: repeated computation
e.g., computes $P(J=true|a)P(M=true|a)$ for each value of e

Variable elimination: carry out summations right-to-left, storing intermediate results (**factors**) to avoid recomputation

$\mathbf{P}(B|J=true, M=true)$

$$= \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(J=true|a)}_{J} \underbrace{P(M=true|a)}_{M}$$
$$= \alpha\mathbf{P}(B)\sum_e P(e)\sum_a \mathbf{P}(a|B,e)P(J=true|a)f_M(a)$$
$$= \alpha\mathbf{P}(B)\sum_e P(e)\sum_a \mathbf{P}(a|B,e)f_J(a)f_M(a)$$
$$= \alpha\mathbf{P}(B)\sum_e P(e)\sum_a f_A(a,b,e)f_J(a)f_M(a)$$
$= \alpha\mathbf{P}(B)\sum_e P(e)f_{\bar{A}JM}(b,e)$ (sum out A)
$= \alpha\mathbf{P}(B)f_{\bar{E}\bar{A}JM}(b)$ (sum out E)
$= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)$

Complexity of exact inference

Singly connected networks (or **polytrees**):
– any two nodes are connected by at most one (undirected) path
– time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:
– can reduce 3SAT to exact inference $\Rightarrow$ NP-hard
– equivalent to *counting* 3SAT models $\Rightarrow$ #P-complete

Inference by stochastic simulation

Basic idea:

1) Draw N samples from a sampling distribution S
2) Compute an approximate posterior probability $\hat{P}$
3) Show this converges to the true probability P

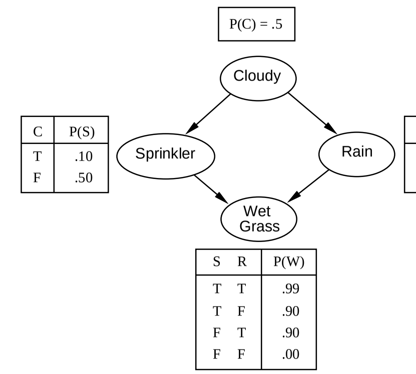
Outline:

– Sampling from an empty network
– Rejection sampling: reject samples disagreeing with evidence
– Likelihood weighting: use evidence to weight samples
– MCMC: sample from a stochastic process whose stationary distribution is the true posterior

Sampling from an empty network

function PRIORSAMPLE(bn) **returns** an event sampled from $\mathbf{P}(X_1, \ldots, X_n)$ specified
  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid Parents(X_i))$
  return x

$\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$
  sample $\rightarrow true$
$\mathbf{P}(Sprinkler|Cloudy) = \langle 0.1, 0.9 \rangle$
  sample $\rightarrow false$
$\mathbf{P}(Rain|Cloudy) = \langle 0.8, 0.2 \rangle$
  sample $\rightarrow true$
$\mathbf{P}(WetGrass|\neg Sprinkler, Rain) = \langle 0.9, 0.1 \rangle$
  sample $\rightarrow true$



Sampling from an empty network contd.

Probability that PRIORSAMPLE generates a particular event

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|Parents(X_i)) = P(x_1 \ldots x_n)$$

i.e., the true prior probability

Let $N_{PS}(\mathbf{Y}=\mathbf{y})$ be the number of samples generated for which $\mathbf{Y}=\mathbf{y}$, for any set of variables $\mathbf{Y}$.

Then $\hat{P}(\mathbf{Y}=\mathbf{y}) = N_{PS}(\mathbf{Y}=\mathbf{y})/N$ and

$$\begin{aligned} \lim_{N\to\infty} \hat{P}(\mathbf{Y}=\mathbf{y}) &= \sum_{h} S_{PS}(\mathbf{Y}=\mathbf{y}, \mathbf{H}=\mathbf{h}) \\ &= \sum_{h} P(\mathbf{Y}=\mathbf{y}, \mathbf{H}=\mathbf{h}) \\ &= P(\mathbf{Y}=\mathbf{y}) \end{aligned}$$

That is, estimates derived from PRIORSAMPLE are **consistent**

Rejection sampling

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

function REJECTIONSAMPLING(X,**e**,bn,N) **returns** an approximation to $P(X|\mathbf{e})$
  N[X] $\leftarrow$ a vector of counts over X, initially zero
  for j = 1 to N **do**
    x $\leftarrow$ PRIORSAMPLE(bn)
    if x is consistent with **e then**
      N[x] $\leftarrow$ **N**[x]+1 where x is the value of X in **x**
  return NORMALIZE(**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

$$\begin{aligned}\hat{\mathbf{P}}(X|\mathbf{e}) &= \alpha \mathbf{N}_{PS}(X,\mathbf{e}) && \text{(algorithm defn.)}\\ &= \mathbf{N}_{PS}(X,\mathbf{e})/N_{PS}(\mathbf{e}) && \text{(normalized by } N_{PS}(\mathbf{e})\text{)}\\ &\approx \mathbf{P}(X,\mathbf{e})/P(\mathbf{e}) && \text{(property of PriorSample)}\\ &= \mathbf{P}(X|\mathbf{e}) && \text{(defn. of conditional probability)}\end{aligned}$$

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if $P(\mathbf{e})$ is small

Likelihood weighting

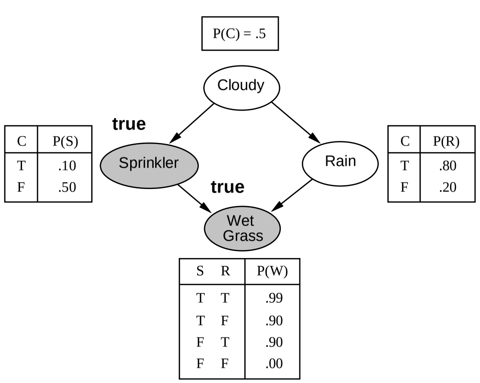
Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```
function WeightedSample(bn,e) returns an event and a weight
   x ← an event with n elements; w ← 1
   for i = 1 to n do
      if X_i has a value x_i in e
         then w ← w × P(X_i = x_i | Parents(X_i))
         else x_i ← a random sample from P(X_i | Parents(X_i))
   return x, w

function LikelihoodWeighting(X,e,bn,N) returns an approximation to P(X|e)
   W[X] ← a vector of weighted counts over X, initially zero
   for j = 1 to N do
      x,w ← WeightedSample(bn)
      W[x] ← W[x] + w where x is the value of X in x
   return Normalize(W[X])
```


Likelihood weighting example

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



LW example contd.

Sample generation process:

1. $w \leftarrow 1.0$
2. Sample $\mathbf{P}(Cloudy) = \langle 0.5, 0.5\rangle$; say $true$
3. $Sprinkler$ has value $true$, so
   $w \leftarrow w \times P(Sprinkler = true|Cloudy = true) = 0.1$
4. Sample $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2\rangle$; say $true$
5. $WetGrass$ has value $true$, so
   $w \leftarrow w \times P(WetGrass = true|Sprinkler = true, Rain = true) = 0.099$

Approximate inference using MCMC

“State” of network = current assignment to all variables

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

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

Main computational problems:
1) Difficult to tell if convergence has been achieved
2) Can be wasteful if Markov blanket is large:
$P(Y_i|MB(Y_i))$ won’t change much (law of large numbers)

Performance of approximation algorithms

Absolute approximation: $|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})| \leq \epsilon$

Relative approximation: $\frac{|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})|}{P(X|\mathbf{e})} \leq \epsilon$

Relative $\Rightarrow$ absolute since $0 \leq P \leq 1$

Randomized algorithms may fail with probability at most δ

Polytime approximation: poly$(n, \epsilon^{-1}, \log \delta^{-1})$

Theorem (Dagum and Luby, 1993): both absolute and relative approximation for either deterministic or randomized algorithms are NP-hard for any $\epsilon, \delta < 0.5$

Case study: Pathfinder IV

Diagnostic expert system for lymph-node diseases.

Deciding on vocabulary: 8 hours

Design topology of network: 35 hours

Make 14,000 probability assessments: 40 hours

Pathfinder now outperforms experts who were consulted during its creation!