Lecture of 11/16

Two parts:

Part 1: Practical issues in constructing Bayes networks

Part 2. Inference in Bayes networks

Part 1. Issues in constructing Bayes Nets

Constructing Belief Networks: Summary

- [[Decide on what sorts of queries you are interested in answering
  - This in turn dictates what factors to model in the network
- Decide on a vocabulary of the variables and their domains for the problem
  - Introduce "Hidden" variables into the network as needed to make the network "sparse"
- Decide on an order of introduction of variables into the network
  - Introducing variables in causal direction leads to fewer connections (sparse structure) **AND** easier to assess probabilities
- Try to use canonical distributions to specify the CPTs
  - Noisy-OR
  - Parameterized discrete/continuous distributions
    - Such as Poisson, Normal (Gaussian) etc

chapter15a - GSview

File Edit Options View Orientation Media Help

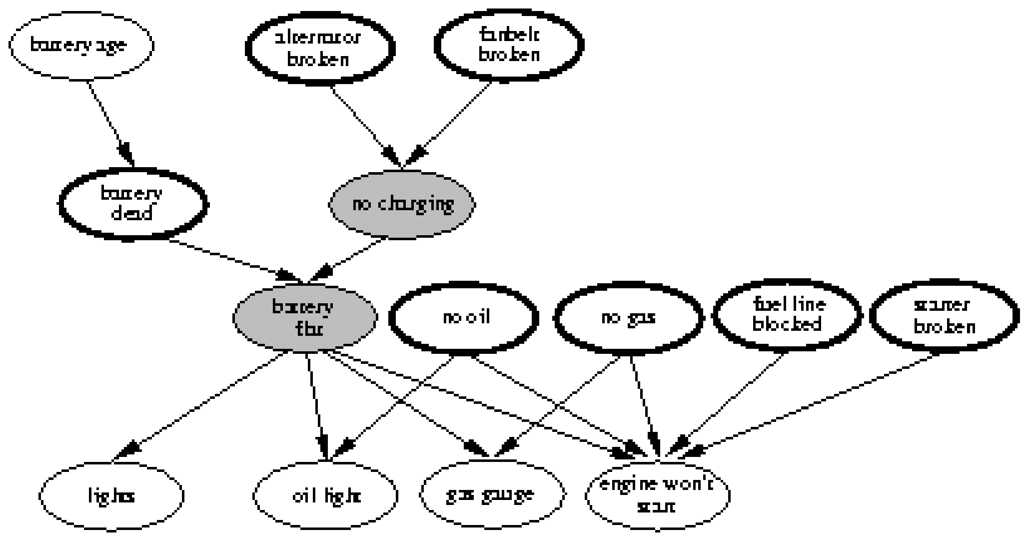
File: chapter15a 192, 127pt Page: "18" 18 of 26

Example: Car diagnosis

Initial evidence: engine won't start

Testable variables (thin ovals), diagnosis variables (thick ovals)

Hidden variables (shaded) ensure sparse structure, reduce parameters



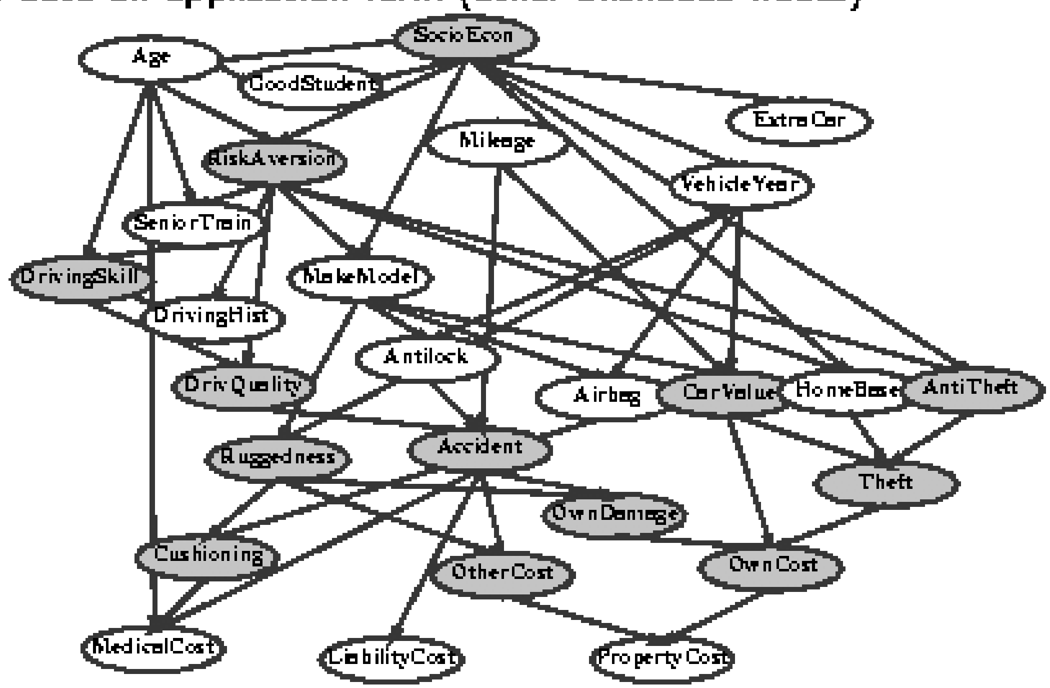
chapter15a - GSview

File Edit Options View Orientation Media Help

File: chapter15a 192, 127pt Page: "19" 19 of 26

Example: Car insurance

Predict claim costs (medical, liability, property)
given data on application form (other unshaded nodes)



Compact conditional distributions

CPT grows exponentially with no. of parents
CPT becomes infinite with continuous-valued parent or child

Solution: canonical distributions that are defined compactly

Deterministic nodes are the simplest case:
$X = f(Parents(X))$ for some function f

E.g., Boolean functions
$NorthAmerican \Leftrightarrow Canadian \vee US \vee Mexican$

E.g., numerical relationships among continuous variables

$$\frac{\partial Level}{\partial t} = \text{inflow + precipation - outflow - evaporation}$$

Compact conditional distributions contd.

Noisy-OR distributions model multiple noninteracting causes
1) Parents $U_1 \ldots U_k$ include all causes (can add leak node)
2) Independent failure probability q_i for each cause alone

$$\Rightarrow P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \Pi_{i=1}^{j} q_i$$

$Cold$	Flu	$Malaria$	$P(Fever)$	$P(\neg Fever)$
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Number of parameters linear in number of parents

Constructing Belief Networks: Summary

- [[Decide on what sorts of queries you are interested in answering
  - This in turn dictates what factors to model in the network
- Decide on a vocabulary of the variables and their domains for the problem
  - Introduce “Hidden” variables into the network as needed to make the network “sparse”
- Decide on an order of introduction of variables into the network
  - Introducing variables in causal direction leads to fewer connections (sparse structure) **AND** easier to assess probabilities
- Try to use canonical distributions to specify the CPTs
  - Noisy-OR
  - Parameterized discrete/continuous distributions
    - Such as Poisson, Normal (Gaussian) etc

Case Study: Pathfinder System

- Domain: Lymph node diseases
  - Deals with 60 diseases and 100 disease findings
- Versions:
  - Pathfinder I: A rule-based system with logical reasoning
  - Pathfinder II: Tried a variety of approaches for uncertainity
    - Simple bayes reasoning outperformed
  - Pathfinder III: Simple bayes reasoning, but reassessed probabilities
  - Parthfinder IV: Bayesian network was used to handle a variety of conditional dependencies.
    - Deciding vocabulary: 8 hours
    - Devising the topology of the network: 35 hours
    - Assessing the (14,000) probabilities: 40 hours
      - Physician experts liked assessing causal probabilites
- Evaluation: 53 “referral” cases
  - Pathfinder III: 7.9/10
  - Pathfinder IV: 8.9/10 [Saves one additional life in every 1000 cases!]
  - A more recent comparison shows that Pathfinder now *outperforms* experts who helped design it!!

Part II. Inference in Bayes Nets

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "3" 3 of 27

Inference tasks

Simple queries: compute posterior marginal $\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$
e.g., $P(NoGas|Gauge=empty, Lights=on, Starts=false)$

Conjunctive queries: $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e}) = \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

Optimal decisions: decision networks include utility information;
probabilistic inference required for $P(outcome|action, evidence)$

Value of information: which evidence to seek next?

Sensitivity analysis: which probability values are most critical?

Explanation: why do I need a new starter motor?

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "9" 9 of 27

Complexity of exact inference

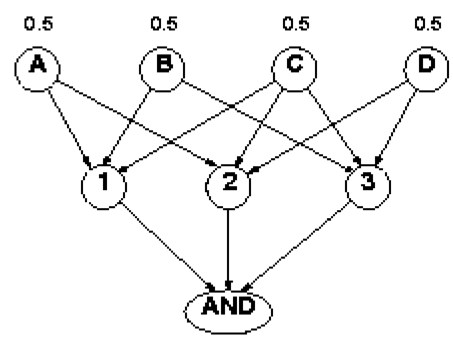
Singly connected networks (or polytrees):
- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

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

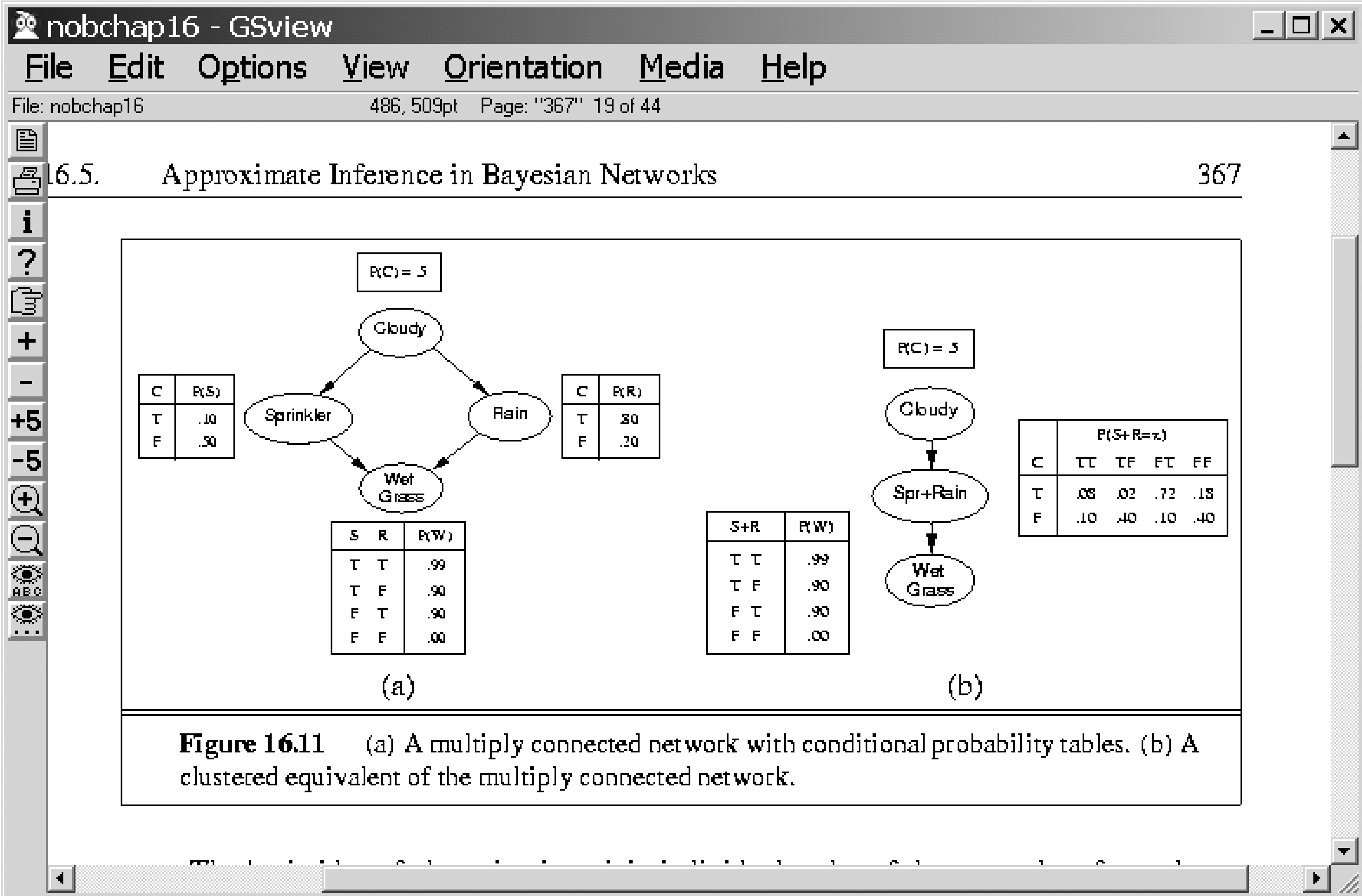
1. A v B v C

2. C v D v ¬A

3. B v C v ¬D



Converting Multi-connected trees into Singly connected trees



Conversion will take exponential time

-Still worth doing if conversion is done off-line

and the cost is amortized over many potential queries

Summary of BN Inference Algorithms

TONS OF APPROACHES

Exact Inference

- Complexity
  - NP-hard (actually #P-Complete; since we “count” models)
    - Polynomial for “Singly connected” networks (one path between each pair of nodes)
- Algorithms
  - Enumeration
  - Variable elimination
    - Avoids the redundant computations of Enumeration
  - [Many others such as “message passing” algorithms, Constraint-propagation based algorithms etc.]

Approximate Inference

- Complexity
  - NP-Hard for both absolute and relative approximation
- Algorithms
  - Based on Stochastic Simulation
    - Sampling from empty networks
    - Rejection sampling
    - Likelyhood weighting
    - [And many more]

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "27" 27 of 27

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 $0 \leq P \leq 1$ (may be $O(2^{-n})$)

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 $\epsilon, \delta < 0.5$

(Absolute approximation polytime with no evidence—Chernoff bounds)

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "4" 4 of 27

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:

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

Rewrite full joint entries using product of CPT entries:

$P(B=true|J=true, M=true)$
$= \alpha\Sigma_e\Sigma_a P(B=true)P(e)P(a|B=true, e)P(J=true|a)P(M=true|a)$
$= \alpha P(B=true)\Sigma_e P(e)\Sigma_a P(a|B=true, e)P(J=true|a)P(M=true|a)$

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "5" 5 of 27

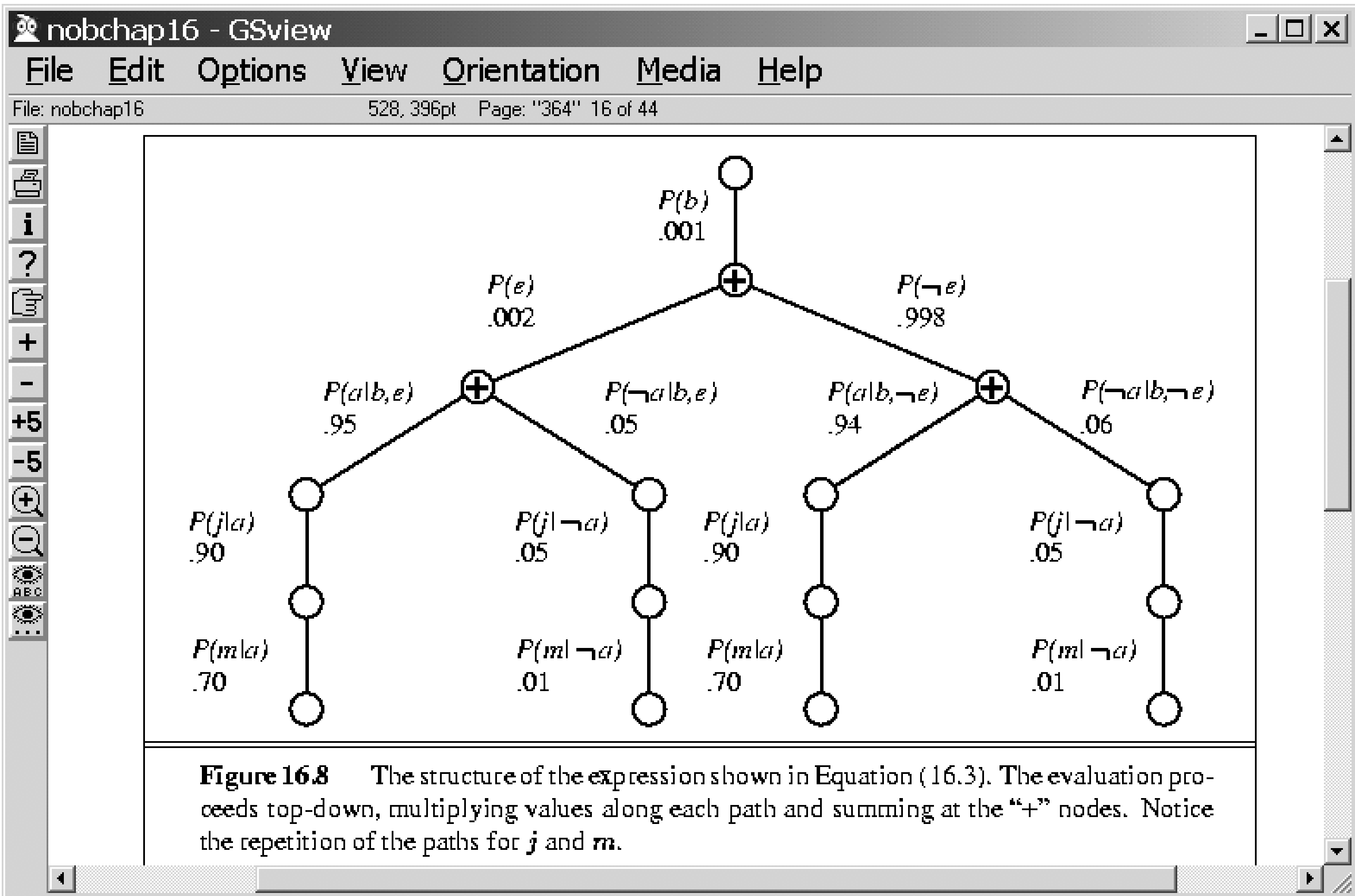
Enumeration algorithm

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

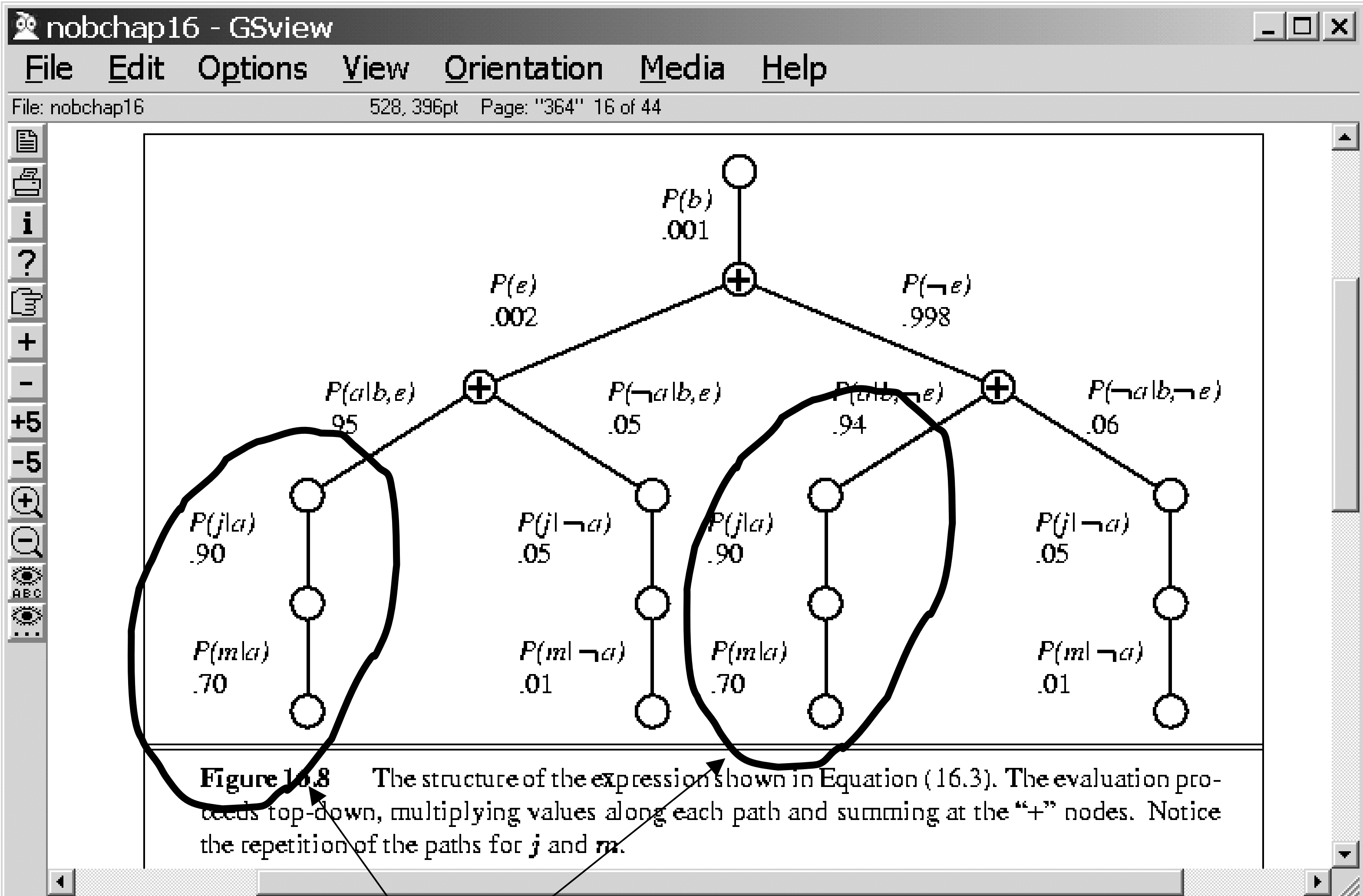
```
EnumerationAsk(X,e,bn) returns a distribution over X
inputs: X, the query variable
        e, evidence specified as an event
        bn, a belief network specifying joint distribution P(X_1, ..., X_n)

  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
```

Inefficient (redundant) computations in Enumeration



Inefficient (redundant) computations in Enumeration



Repeated multiplications

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

$$
\begin{aligned}
\mathbf{P}(B|J=true, M=true) \\
&= \alpha \underbrace{\mathbf{P}(B)}_{B} \Sigma_e \underbrace{P(e)}_{E} \Sigma_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(J=true|a)}_{J} \underbrace{P(M=true|a)}_{M} \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(J=true|a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a) \\
&= \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A\text{)} \\
&= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)} \\
&= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "7" 7 of 27

Variable elimination: Basic operations

Pointwise product of factors f_1 and f_2:

$$f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)$$
$$= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)$$

E.g., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Summing out a variable from a product of factors: move any constant factors outside the summation:

$$\Sigma_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \, \Sigma_x \, f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_{\bar{X}}$$

assuming $f_1, \ldots, f_i$ do not depend on X

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "8" 8 of 27

Variable elimination algorithm

```
function ELIMINATIONASK(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)

   if X ∈ e then return observed point distribution for X
   factors ← []; vars ← REVERSE(VARS[bn])
   for each var in vars do
       factors ← [MAKEFACTOR(var, e)|factors]
       if var is a hidden variable then factors ← SUMOUT(var, factors)
   return NORMALIZE(POINTWISEPRODUCT(factors))
```

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "10" 10 of 27

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

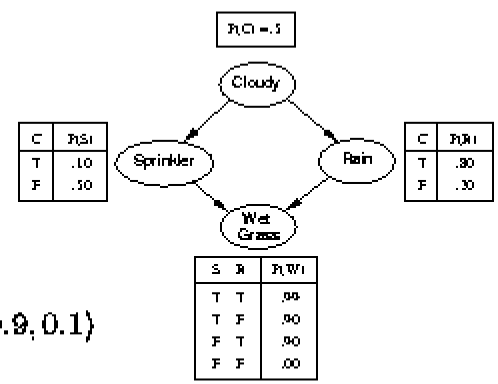
chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "11" 11 of 27

Sampling from an empty network

function PRIORSAMPLE(bn) **returns** an event sampled from $\mathbf{P}(X_1, \ldots, X_n)$ specified by bn
  x ← an event with n elements
  for i = 1 **to** n **do**
    x_i ← 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
$S_{PS}(x_1 \ldots x_n) = \Pi_{i=1}^{n} P(x_i | Parents(X_i)) = P(x_1 \ldots x_n)$
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

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

That is, estimates derived from PRIORSAMPLE are consistent

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "13" 13 of 27

Rejection sampling

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

```
function REJECTION-SAMPLING(X, e, bn, N) returns an approximation to P(X|e)
   N[X] ← a vector of counts over X, initially zero
   for j = 1 to N do
       x ← PRIOR-SAMPLE(bn)
       if x is consistent with e then
          N[x] ← 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

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

Hence rejection sampling returns consistent posterior estimates

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

chapter15b - GSview

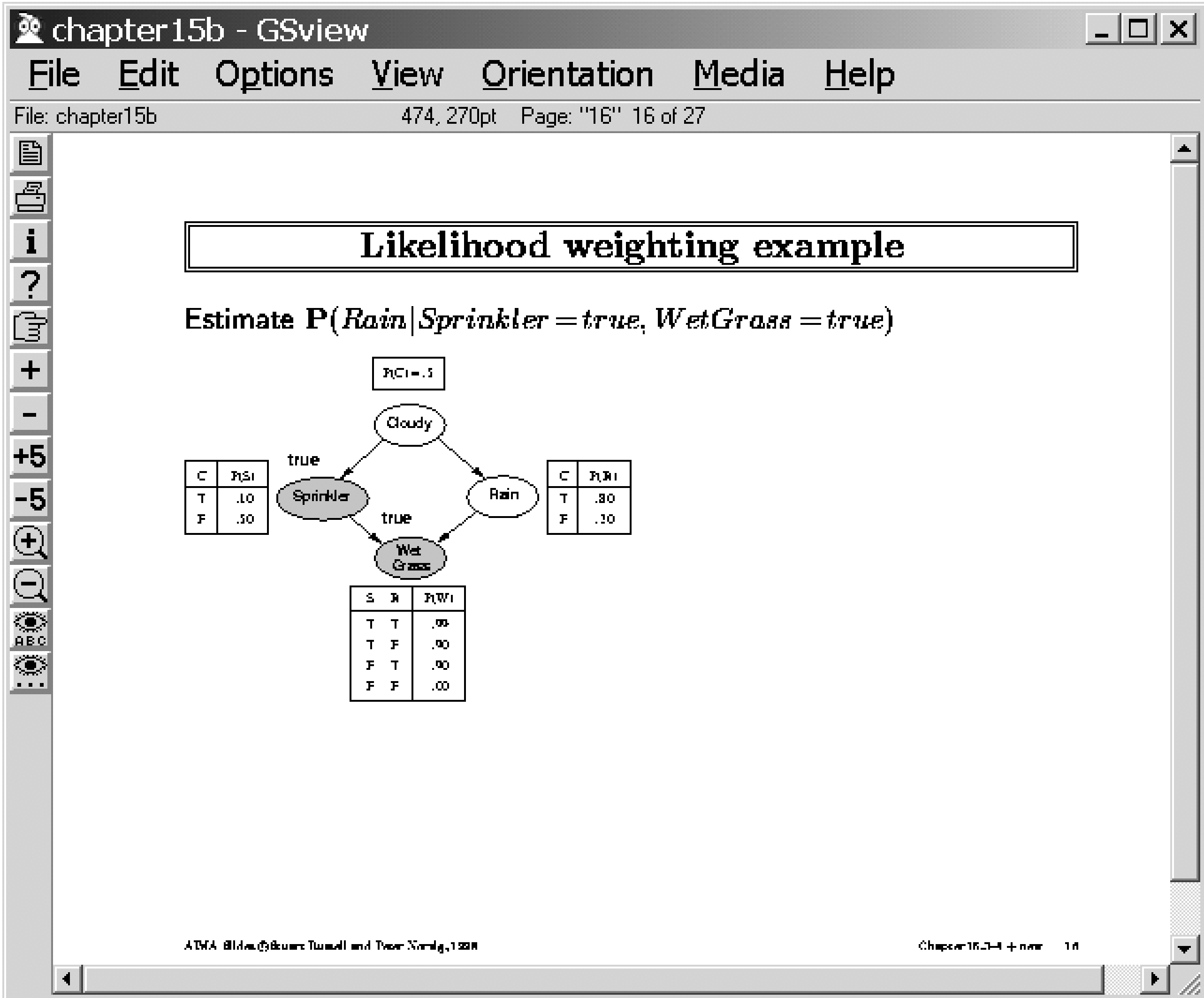
File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "15" 15 of 27

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])
```



chapter15b - GSview
File Edit Options View Orientation Media Help
File: chapter15b
474, 270pt Page: "16" 16 of 27
Likelihood weighting example
Estimate $\mathbf{P}(Rain|Sprinkler = true, WetGrass = true)$
P(C) = .5
Cloudy
Sprinkler
Rain
Wet Grass
true
true
C P(S)
T .10
F .50
C P(R)
T .80
F .20
S R P(W)
T T .99
T F .90
F T .90
F F .00
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 15.3–4 + new 16

chapter15b - GSview

File Edit Options View Orientation Media Help

File: chapter15b 474, 270pt Page: "17" 17 of 27

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
   $w \leftarrow w \times P(WetGrass = true | Sprinkler = true, Rain = true) = 0.099$

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 $0 \leq P \leq 1$ (may be $O(2^{-n})$)

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 $\epsilon, \delta < 0.5$

(Absolute approximation polytime with no evidence—Chernoff bounds)

Summary of BN Inference Algorithms

TONS OF APPROACHES

Exact Inference

- Complexity
  - NP-hard (actually #P-Complete; since we "count" models)
    - Polynomial for "Singly connected" networks (one path between each pair of nodes)
- Algorithms
  - Enumeration
  - Variable elimination
    - Avoids the redundant computations of Enumeration
  - [Many others such as "message passing" algorithms, Constraint-propagation based algorithms etc.]

Approximate Inference

- Complexity
  - NP-Hard for both absolute and relative approximation
- Algorithms
  - Based on Stochastic Simulation
    - Sampling from empty networks
    - Rejection sampling
    - Likelyhood weighting
    - [And many more]