Artificial Intelligence

Chapter 14
Probabilistic reasoning

Stuart RUSSEL

reorganized by L. Aszalós

May 16, 2016

Outline

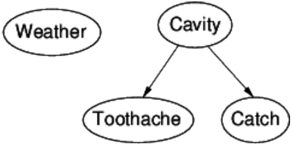
- Syntax
- Semantics
- Exact inference by enumeration
- Exact inference by variable elimination

Bayesian networks

- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link $\approx$ "directly influences")
  - a conditional distribution for each node given its parents: $\mathcal{P}(X_i|Parents(X_i))$
- In the simplest case, conditional distribution represented as a **conditional probability table** (CPT) giving the distribution over X_i for each combination of parent values

Example

- Topology of network encodes conditional independence assertions:

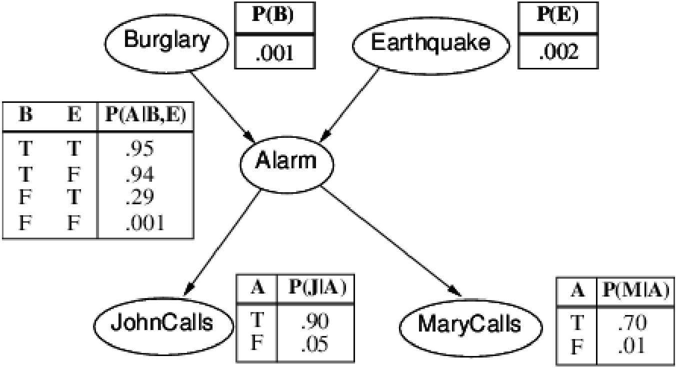


- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

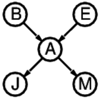
Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?
- Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

Example contd.

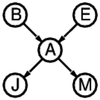


Compactness



- *A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values*
- *Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1 - p$)*
- *If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers*
- *I.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution*
- *For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)*

Global semantics



- **Global** *semantics defines the full joint distribution as the product of the local conditional distributions*
- $$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$
- *e.g.,* $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

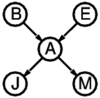
$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \approx 0.00063$$

Inference tasks

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

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:* $\mathcal{P}(B|j,m)$

$$= \mathcal{P}(B,j,m)/P(j,m) = \alpha\mathcal{P}(B,j,m) = \alpha \sum_e \sum_a \mathcal{P}(B,e,a,j,m)$$

- *Rewrite full joint entries using product of CPT entries:*

$$\mathcal{P}(B|j,m) = \alpha \sum_e \sum_a \mathcal{P}(B)P(e)\mathcal{P}(a|B,e)P(j|a)P(m|a) =$$

$$\alpha\mathcal{P}(B) \sum_e P(e) \sum_a \mathcal{P}(a|B,e)P(j|a)P(m|a)$$

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

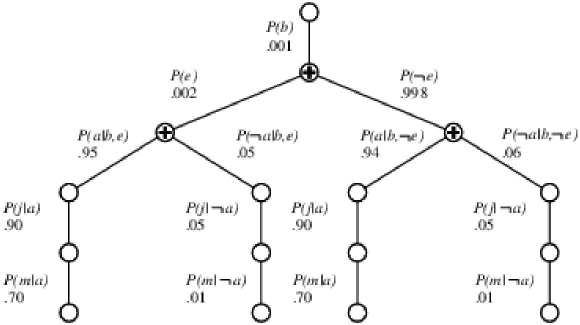
Enumeration algorithm

```
function Enumeration-Ask(X, e, bn): returns a distribution over X
    inputs: X, the query variable
            e, observed values for variables E
            bn, a Bayesian network with variables X U E U Y

    Q(X) := a distribution over X, initially empty
    for each value x_i of X do
        extend e with value x_i for X
        Q(x_i) := Enumerate-All(Vars[bn], e)
    return Normalize(Q(X))

function Enumerate-All(vars, e): a real number
    if Empty?(vars) then return 1.0
    Y := First(vars)
    if Y has value y in e
        then return P(y|Parent(Y))*Enumerate-All(Rest(vars), e)
        else return sum_y P(y|Parent(Y))*Enumerate-All(Rest(vars), e_y)
                where e_y is e extended with Y=y
```

Evaluation tree



- Enumeration is inefficient: repeated computation
  - e.g., computes $P(j|a)P(m|a)$ for each value of e

Inference by variable elimination

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

$$\mathcal{P}(B|j,m) = \alpha \underbrace{\mathcal{P}(B)}_{B} \sum_{e} \underbrace{P(e)}_{E} \sum_{a} \underbrace{\mathcal{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M} =$$

$$\alpha \mathcal{P}(B) \sum_{e} P(e) \sum_{a} \mathcal{P}(a|B,e) P(j|a) f_M(a)$$

$$= \alpha \mathcal{P}(B) \sum_{e} P(e) \sum_{a} \mathcal{P}(a|B,e) f_J(a) f_M(a)$$

$$= \alpha \mathcal{P}(B) \sum_{e} P(e) \sum_{a} f_A(a,b,e) f_J(a) f_M(a)$$

$$= \alpha \mathcal{P}(B) \sum_{e} P(e) f_{\bar{A}JM}(b,e) (\text{sum out } A)$$

$$= \alpha \mathcal{P}(B) f_{\bar{E}\bar{A}JM}(b) (\text{sum out } E)$$

$$= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)$$

Variable elimination: Basic operations

- **Summing out** a variable from a product of factors:
  - move any constant factors outside the summation
  - add up submatrices in pointwise product of remaining factors

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

- assuming $f_1, \ldots, f_i$ do not depend on X
- **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., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Variable elimination algorithm

```
function Elimination-Ask(X, e, bn): a distribution over X
    inputs: X, the query variable
        e, evidence specified as an event
        bn, a belief network specifying joint distribution P(X1,...,Xn)

    factors:=[], vars:= Reverse(Vars[bn])
    for each var in vars do
        factors:= [Make-Factor(var, e)| factors]
        if var is a hidden variable then factors:= Sum-Out(var, factors)
    return Normalize(Pointwise-Product(factors))
```