Algorithms Part II: Current Directions

Bruce D’Ambrosio and Robert Fung
Introduction

1. Computational breakthrough of Bayes Nets is exploitation of conditional independence captured graphically.

1. Research directions
   - exploit additional structure
   - push expressivity

1. Structure exploitation
   - Noisy-Ors
   - Asymmetries (e.g., Similarity Networks)
   - Entropy

1. Expressivity
   - Continuous variables
Outline

1. **Representation**
   - Noisy Or
   - Asymmetries
   - Continuous Vars

1. **Approximation**
   - Simulation
   - Search
Noisy Or

1. Distribution size is exponential in number of parents
   - Difficult to acquire
   - Expensive to compute

1. Noisy Or interaction model
   - Finding absent if all parents absent
   - parent contributions independent
   - widely used for multi-parent interactions.

1. Representations
   - ladder model
   - local expressions

```
F1  F2  F3  F4  F5
  |   |   |   |
  D1  D2  D3  D4
```

Summer Institute on Probability in AI 1994

Inference 2 4
Noisy Or II

1 Model implies:

- $P(\neg f_3|D_1,D_2,D_3) = C(\neg f_3|D_1)*C(\neg f_3|D_2)*C(\neg f_3|D_3)$
  - where $C(\neg f_1|D_1) = P(\neg f_1|D_1,\neg d_2,\neg d_3)$
  - $= 1, D_1=\neg d_1; (1-P(f_1|d_1,\neg d_2,\neg d_3)), D_1=d_1$
- $P(f_3|D_1,D_2,D_3) = 1-P(\neg f_3|D_1,D_2,D_3)$
Quickscoring

1. Basic expression is exponential in causes
2. Rearrange posterior expression for efficient evaluation (Heckerman 89):
   - Linear in causes
   - Linear in negative findings
   - Exponential in positive findings
3. Doesn’t take advantage of topological structure
Quickscore - Negative Findings

1. \[ P(D1) = \sum_{D2} P(F1|D1,D2) \times P(F2|D1,D2) \times P(D1) \times P(D2) \]
2. but \( F1 \) negative, so: \[ P(F1|D1,D2) = C(\neg f1|D1) \times C(\neg f1|D2) \]
3. therefore
   - \[ P(D1) = \sum_{D2} C(\neg f1|D1) \times C(\neg f1|D2) \times C(\neg f2|D1) \times C(\neg f2|D2) \times P(D1) \times P(D2) \]
4. rearranging:
   - \[ P(D1) = (C(\neg f1|D1) \times C(\neg f2|D1) \times P(D1)) \times (\sum_{D2} C(\neg f1|D2) \times C(\neg f2|D2) \times P(D2)) \]
5. Linear in negative findings and diseases
Quickscore - Positive Findings

1 \[ P(D1) = \sum_{D2} P(F1|D1,D2)P(F2|D1,D2)P(D1)P(D2) \]
1 But F1 Positive, so: \[ P(F1|D1,D2) = 1 - C(\neg f1|D1)C(\neg f1|D2) \]
1 Substituting:
   • \[ P(D1) = \sum_{D2} (1 - C(\neg f1|D1)C(\neg f1|D2)) \times (1 - C(\neg f2|D1)C(\neg f2|D2)) \times P(D1)P(D2) \]
1 distributing over F1,F2:
   • \[ \sum_{D2} (1 \times 1 \times P(D1) \times P(D2) - C(\neg f1|D1)C(\neg f1|D2) \times 1 \times P(D1) \times P(D2) \]
   • \[ -1 \times C(\neg f2|D1)C(\neg f2|D2) \times P(D1) \times P(D2) \]
   • \[ + C(\neg f1|D1)C(\neg f1|D2) \times C(\neg f2|D1)C(\neg f2|D2) \times P(D1) \times P(D2) \]
1 rearranging:
   • \[ P(D1) \times \sum_{D2} P(D2) - C(\neg f1|D1) \times P(D1) \times \sum_{D2} (C(\neg f1|D2) \times P(D2)) \]
   • - \[ C(\neg f2|D1) \times P(D1) \times \sum_{D2} (C(\neg f2|D2) \times P(D2)) \]
   • + \[ C(\neg f1|D1)C(\neg f2|D1) \times P(D1) \times \sum_{D2} (C(\neg f1|D2) \times C(\neg f2|D2) \times P(D2)) \]
Ladder Model

1. Heckerman 93
2. \( P(F_1^2) = P(F_1|D_1 \text{ only}) \)
3. \( P(F_1^1|F_1^2, D_2): \)
   - \( 1 - F_1^2 \) present
   - \( P(F_1|D_2 \text{ only}) \) - \( F_1^2 \) absent
Ladder model, negative evidence

1. Negative evidence is asserted at each F\textsuperscript{x} node!
   - note this means re-writing child distributions
   - \( P'(F_{1*} \mid D2) = P(F_{1*} \mid D2, F_{1*}^2) \)

1. Clearly shows negative finding decouples causes.

```
<table>
<thead>
<tr>
<th>D2</th>
<th>F1\textsuperscript{2}</th>
<th>P(F_{11} \mid D2, F_{12})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>Abs</td>
<td>1.0</td>
</tr>
<tr>
<td>Pres</td>
<td>Abs</td>
<td>.4</td>
</tr>
<tr>
<td>Abs</td>
<td>Pres</td>
<td>.0</td>
</tr>
<tr>
<td>Pres</td>
<td>Pres</td>
<td>1.0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>D2</th>
<th>P'(F_{1*} \mid D2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>1.0</td>
</tr>
<tr>
<td>Pres</td>
<td>.6</td>
</tr>
</tbody>
</table>
```
Ladder model, Positive Evidence

1. Post positive evidence only at terminal nodes
2. Clearly linear in diseases
Local Expressions

1. Symbolic algebra can factor dynamically
2. Order in which we distribute over evidence matters

<table>
<thead>
<tr>
<th>Case</th>
<th>Pos</th>
<th>Ev</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>23</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Research in Noisy Or

1. Generalized noisy or (Srinivas 93)
2. Multi-value
3. Multi-level networks (CPCS, extended set factoring)
4. Embedding noisy or in general networks
Asymmetries

1. **Representation:**
   - Value dependent independence
     - \( P(Z|x,Y) = .3 \)
     - Contingent existence

2. **Inference:**
   - Efficient exploitation of asymmetry

3. **D’Ambrosio 91, Heckerman and Geiger 93, Poole 94, Shenoy ??, Fung and Shachter, Barlow, DA related**

4. **Inferential cost/benefit unclear**

Diagram:
- A circle labeled 'Return' connected to a diamond labeled 'Value'
- A square labeled 'Bet?' connected to 'Return' and 'Value'
Continuous Variables

1. Signal to symbol gap: a major embarrassment for AI
2. Some success in mixed discrete/continuous belief nets
3. General models/simulation-based (predictive)
   - Demos (Henrion)
4. Exact, linear with discrete predecessors (diagnostic)
   - Hugin (Jensen, Olesen)
   - SPI (Chang and Fung)
5. Continuous Influence Diagrams (Kenley, Poland)
Approximation

1. Exact Inference is Hard

1. Approximation techniques:
   • Approximate representation
     • Qualitative (Goldschmidt, Pearl, Wellman, ...)
     • Reduced model (Breese, Heckerman, Horvitz)
     • Ignoring weak dependencies (Jensen)
   • Approximate inference
     • Simulation (Pearl, Fung, Peot, ...)
     • Search (D’Ambrosio, Henrion, Poole)
     • Local evaluation (Draper)
Backward Simulation

- A new algorithm family for probabilistic inference in Bayesian Networks.
- Addresses two limitations of current simulation methods (forward, markov blanket)
  - slow convergence when faced with low-likelihood evidence
  - slow convergence with deterministic relationships
- No magic with respect to complexity results.
Forward Sampling Example

Possible Order
- D1, D2, D3, D4

Weight
- $P(E_1|d_1)*P(E_2|d_3)$
Backward Simulation: Simple Example

\[ \frac{P(X=x_0)}{P(X=x_1)} = 10^{-n} \]

\[ \frac{P(E|X=x_0)}{P(E|X=x_1)} = 10^m \]

\[ m >> n \]

\[ \frac{P(X=x_0|E)}{P(X=x_1|E)} = 10^{m-n} \]
Backward Simulation: Algorithm

1 Ordering
   • A node must be instantiated to be sampled
   • The union of the predecessors of the nodes in the order must cover all the nodes in the network

1 Sampling
   • Takes place from normalized likelihoods and sets the predecessor values
   • Forward sampling is used for nodes with no downstream evidence

1 Weighting
   • Importance sampling
   • the ratio of the prior probability and the sampling probability
Backward Simulation: Weighting Equation

\[ Z(x) = \frac{P(x)}{P_s(x)} \]

\[ P(x) = \prod_{i \in N} P(x_i | x_{Pa(i)}) \]

\[ P_s(x) = \prod_{j \in N_{bo}} \frac{P(x_j | x_{Pa(j)})}{\sum_{x_{Pa(j)} \in X_{Pa(j)}} P(x_j | x_{Pa(j)})} \]

\[ Z(x) = \frac{\prod_{i \in N} P(x_i | x_{Pa(i)})}{\prod_{j \in N_{bo}} \frac{P(x_j | x_{Pa(j)})}{\sum_{x_{Pa(j)} \in X_{Pa(j)}} P(x_j | x_{Pa(j)})}} \]
Backward Simulation: Example

1. Simulation out from the evidence
2. Sampling sets a node's predecessors

Possible Orders
- E2, E1, D2', D4'
- E1, D2', E2, D4'
Discussion

1. Improved convergence in low-likelihood situations
2. Computational costs of normalization
   - costs can be reduced through: precomputation and/or caching
3. Integration of backwards and forwards sampling
4. Grouping of nodes for sampling
5. Dynamic node ordering
6. Handling continuous deterministic relations through function inversion.
Backward Simulation: Status

1. Have “vanilla” implementation working in IDEAL
2. Have worked out QMR (e.g., noisy-or) formulation
3. Currently working on QMR implementation
Instantiation Control

1. Why strictly forward or backward?
2. Partial instantiation makes distribution “informative”
3. Entropy and other information-theoretic measures
4. Static vs Dynamic selection
Search

1 Goal: estimate query by computing mass of large instantiations:
   • find instantiation with maximum a-posteriori probability.
   • repeat until sufficient mass computed.

1 Method: Incrementally extend an instantiation using local search.

1 D’Ambrosio: incremental probabilistic inference

1 Henrion: search in large BN2O nets

1 Poole: use of conflict sets in search
Search - Simple Example

\[ P(D) = \sum_{B,C} P(D|B,C) \sum_A P(C|A) P(B|A) P(A) \]

\[
P(D) = 0.58 * 0.6 * 0.8 * 0.9 \ (0.25056 \rightarrow D=t)
\]

\[
P(D) = 0.56 * 0.8 * 0.6 * 1 \ (0.02688 \rightarrow D=
\]

\[
\sum_{B,C} P(D|B,C) \sum_A P(C|A) P(B|A) P(A)
\]

\[
P(D) = \sum_{B,C} P(D|B,C) P(C|A) P(B|A) P(A)
\]

\[
P(D) = 0.58 * 0.6 * 0.8 * 0.9 \ (0.25056 \rightarrow D=t)
\]

\[
P(D) = 0.56 * 0.8 * 0.6 * 1 \ (0.02688 \rightarrow D=
\]

\[
\sum_{B,C} P(D|B,C) \sum_A P(C|A) P(B|A) P(A)
\]

\[
P(D) = \sum_{B,C} P(D|B,C) P(C|A) P(B|A) P(A)
\]

\[
P(D) = 0.58 * 0.6 * 0.8 * 0.9 \ (0.25056 \rightarrow D=t)
\]

\[
P(D) = 0.56 * 0.8 * 0.6 * 1 \ (0.02688 \rightarrow D=
\]
Search

1. Like simulation, instantiate the variables in one distribution at a time
2. Use heuristic search techniques to guide instantiation process

Why?
- Direct solution of MLCH (admissible search required!)
- Few large terms might be informative
  - skewed -> n+1 scenarios contain $2/e$ mass!

How?
- $f(i) = g(i) \times h(i)$
  - $g(i)$ - likelihood of partial scenario
  - $h(i)$ - heuristic estimate of max prob of remaining vars given instantiation so far.

But: performance can be poor
- Large search space
- Extensive backtracking
Search - first example

Model: A, C, A', C', A'', C'', low entropy

Suppose we know I - I'' all zero, O, O' zero, O'' 1

Search starts with A, C, A', C', A'', C'' all in high probability state (ok)

Now what?

• Poor search architecture can yield exponential time
Poole - Conflicts in Search

1. Can derive a conflict involving \( A'' \) from first search.
2. \( P(V|O) := \text{Maximum Probability of any assignment to a conflict containing vars } V. \)
3. \( h(i) = \prod P(V|O) \) over all subsequent independent conflicts.
4. Quickly reveals \( A, A' \) not worth working on.
5. Poole - UAI 92, UAI-93
Conflicts in Search

1. Conflict set: \{C’, A’’\}
2. \(h\) for any partial term not including C’ or A’’ is max P(c’, a’’) consistent with obs.
3. \(h\) for any partial term including C’ or A’’ is 1.0
D’Ambrosio - factoring and caching

2. Top down, left to right search, but cache results
3. After first fail, ANY instantiation of left subtree with same C’ has same h.
4. Unlike Poole, feeds back all info, not just conflicts (0s)
5. But doesn’t generalize over conflict set
6. D’Ambrosio: DX-92, UAI-93
What is search good for?

1. MLCH
2. Posterior estimation
3. Policy estimation

Assumptions:
- looking for a few good instantiations
- easy to find
- more is better
Experiments

Terms Computed

(MLCH, System OK)
Experiments III

(MLCH, Single Fault)
Search in Noisy OR

1. TopN (Henrion, 89)

1. Again the crucial assumption:
   - “...only a tiny fraction of them account for most of the probability mass.”

1. $R(h,F) = \frac{P(h|F)}{P(h_0|F)}$

1. $\text{MEP}(d,H) = \frac{R(H_\text{ud}, F)}{R(H,F)}$
   - if $\text{MEP}(d,H) < 1$, don’t add d yet

1. $\text{MEP}(d,H) \geq \text{MEP}(d,H^+)$
   - if it isn’t worth it to add d now, it never will be

1. Algorithm:
   - start with null hypothesis
   - consider all d with $\text{MEP} > 1$
   - pick best $\text{MEP}$, add, expand resulting hypothesis
   - loop
Search - Open issues

1. Probability community ahead in exploiting network structure
2. AI automated reasoning community ahead in supporting dynamic restructuring.
3. Mixed discrete/continuous networks?
4. Instantiation order?
Summary

1 Representation
   • Noisy Or
   • Asymmetries
   • Continuous

1 Approximation
   • Search
   • Simulation

1 Are we done?
   • broad outline seems understood
   • lots of cleanup
   • hybrid algorithms/architectures?
   • engineering
   • experimentation
   • mining real applications (e.g., QMR)