Exact Lifted Inference on Relational Models

Statistical Relational AI

Tutorial at BTW 2019

Tanya Braun, University of Lübeck
Propositional Models: Worlds

- Characterise world by random variables
  - E.g., $Epid$
  - Possible values (range):
    $$r(Epid) = \{true, false\}$$
- 1 world = specific values for variables
- Probability per world
- Joint probability distribution $P_F$ over all possible worlds

$$2^9 = 512 \text{ possible worlds}$$
Propositional Models: Factors

• Full joint $P_F$ as product of factors
  Sparse encoding!
  • Model $F$
  • $rv(F), rv(f)$
random variables in $F, f$

$$P_F = \frac{1}{Z} \prod_{i=1}^{n} f_i$$

$$Z = \sum_{v \in r(rv(F))} \prod_{i=1}^{n} f_i(\pi_{rv(f_i)}(v))$$

$7 \cdot 2^3 = 56$ entries
Propositional Models: Factors

• Factors
  • Potentials
  • $f^1_2(\text{Travel.eve}, \text{Sick.eve}, \text{Epid})$

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Epid</th>
<th>Sick.eve</th>
<th>$f^1_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>9</td>
</tr>
</tbody>
</table>
Query Answering (QA): Queries

- **Marginal distribution**
  - $P(\text{Sick.eve})$
  - $P(\text{Travel.eve, Treat.eve.m}_1)$

- **Conditional distribution**
  - $P(\text{Sick.eve}|\text{Epid})$
  - $P(\text{Epid}|\text{Sick.eve} = \text{true})$

- **Most probable assignment**

\[
\text{argmax}_{v \in r(R)} \sum_{v \in r(R)} P_F(v), \quad R \subseteq rv(F)
\]

- **MPE:** $R = rv(F)$
- **MAP:** $R = \{\text{Travel.eve, Man.virus}\}$
QA: Variable Elimination (VE)

- Eliminate all variables not appearing in query
  - Through summing out
- E.g., marginal
  - \( P(\text{Travel.eve}) \)

\[
P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} \sum_{s \in r(\text{Sick.eve})} \sum_{m_1 \in r(\text{Treat.eve.m}_1)} \sum_{m_2 \in r(\text{Treat.eve.m}_2)} \sum_{o \in r(\text{Nat.flood})} \sum_{i \in r(\text{Nat.fire})} \sum_{w \in r(\text{Man.war})} \sum_{v \in r(\text{Man.virus})} P_F
\]
QA: Variable Elimination (VE)

P(Travel.eve) ∝ \sum_{e \in r(Epid)} \sum_{s \in r(Sick.eve)} f_2^1(Travel.eve, e, s) \sum_{m_1 \in r(Treat.eve.m_1)} f_3^1(e, s, m_1) \\
\sum_{m_2 \in r(Treat.eve.m_2)} f_3^2(e, s, m_2)

f_1'(e) \sum_{o \in r(Nat.flood)} \sum_{i \in r(Nat.fire)} \sum_{w \in r(Man_war)} f_1^1(o, w, e)f_1^2(i, w, e)

f_1'(q, i, e) \sum_{v \in r(Man.virus)} f_1^4(i, v, e)

Zhang and Poole (1994)
QA: Variable Elimination (VE)

\[ P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'_2(\text{Travel.eve}, e, s) f'_3(e, s) f''(e, s) \]

\[ = \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'(\text{Travel.eve}, e, s) \]

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Epid</th>
<th>Sick.eve</th>
<th>( f' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>50</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>10</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>34</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>16</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>54</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>36</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>12</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>69</td>
</tr>
</tbody>
</table>

Zhang and Poole (1994)
QA: Variable Elimination (VE)

\[ P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} f_1'(e) \sum_{s \in r(\text{Sick.eve})} f_2'(\text{Travel.eve}, e, s) f_3'(e, s) f_3''(e, s) \]

\[ = \sum_{e \in r(\text{Epid})} f_1'(e) \sum_{s \in r(\text{Sick.eve})} f'(\text{Travel.eve}, e, s) \]

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Epid</th>
<th>Sick.eve</th>
<th>f'</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>34</td>
<td>50</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>54</td>
<td>90</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>12</td>
<td>81</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>
QA: Variable Elimination (VE)

\[ P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'_1(\text{Travel.eve}, e, s) f'_3(e, s) f''_3(e, s) \]

\[ = \sum_{e \in r(\text{Epid})} f'_1(e) f''(\text{Travel.eve}, e) \]

\[ = \sum_{e \in r(\text{Epid})} f'''(\text{Travel.eve}, e) \]

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Epid</th>
<th>f'''</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>90</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>100</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>135</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>162</td>
</tr>
</tbody>
</table>

Zhang and Poole (1994)
QA: Variable Elimination (VE)

\[
P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f^1_2(\text{Travel.eve}, e, s)f'_3(e, s)f''_3(e, s)
\]

\[
= \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'(\text{Travel.eve}, e, s)
\]

\[
= \sum_{e \in r(\text{Epid})} f'_1(e)f''(\text{Travel.eve}, e)
\]

\[
= \sum_{e \in r(\text{Epid})} f'''(\text{Travel.eve}, e)
\]

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Epid</th>
<th>f'''</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>90</td>
<td>190</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>135</td>
<td>297</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>162</td>
<td></td>
</tr>
</tbody>
</table>

Zhang and Poole (1994)
QA: Variable Elimination (VE)

Zhang and Poole (1994)

\[ P(Travel.\ eve) \propto \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'_2(Travel.\ eve, e, s) f'_3(e, s) f''_3(e, s) \]

\[ = \sum_{e \in r(\text{Epid})} f'_1(e) \sum_{s \in r(\text{Sick.eve})} f'(Travel.\ eve, e, s) \]

\[ = \sum_{e \in r(\text{Epid})} f'_1(e) f''(Travel.\ eve, e) \]

\[ = \sum_{e \in r(\text{Epid})} f'''(Travel.\ eve, e) = f(Travel.\ eve) = f_n(Travel.\ eve) \]

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>190</td>
</tr>
<tr>
<td>true</td>
<td>297</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>f_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0.39</td>
</tr>
<tr>
<td>true</td>
<td>0.61</td>
</tr>
</tbody>
</table>
QA: Variable Elimination (VE)

• Eliminate all variables not appearing in query
  • Through summing out
• E.g., conditional
  • $P(\text{Travel.eve}|\text{Epid} = \text{true})$
  • Absorb $\text{Epid} = \text{true}$ in all factors
• Sum out remaining variables

Zhang and Poole (1994)
QA: Evidence Absorption

• Absorb $Epid = true$ in all factors, e.g., $f^1_2$

• Possibly eliminate variable

<table>
<thead>
<tr>
<th>Travel.eve</th>
<th>Sick.eve</th>
<th>$f^1_2$</th>
<th>$e$</th>
<th>$f^1_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Zhang and Poole (1994)
Problem: Models Explode

\[ 7 \cdot 2^3 = 56 \text{ entries in 7 factors, 9 variables} \]
Problem: Models Explode

13 \cdot 2^3 = 104 entries in 13 factors, 17 variables
Solution: Lifting

• Parameterised random variables = PRV
  • With logical variables
  • E.g., $X$
  • Possible values (domain):
    $\mathcal{D}(X) = \{alice, eve, bob\}$

• Inputs to factors!

Poole (2003)
Solution: Lifting

- Factors with PRVs = \textit{parfactors}
  - (Graphical) Model G
  - E.g., $g_2$

<table>
<thead>
<tr>
<th>Travel(X)</th>
<th>Epid</th>
<th>Sick(X)</th>
<th>$g_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>9</td>
</tr>
</tbody>
</table>

$3 \cdot 2^3 = 24$ entries in 3 parfactors, 6 PRVs

Poole (2003)
Solution: Lifting

*Grounding*

- E.g., $gr(g_2) = \{f_2^1, f_2^2, f_2^3\}$

<table>
<thead>
<tr>
<th>$Travel(eve)$</th>
<th>Epid</th>
<th>$Sick(eve)$</th>
<th>$f_2^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>5</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>4</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>6</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>9</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$Travel(bob)$</th>
<th>Epid</th>
<th>$Sick(bob)$</th>
<th>$f_2^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>5</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>6</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>9</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>9</td>
</tr>
</tbody>
</table>
Solution: Lifting

- Joint probability distribution $P_G$ by grounding

$$P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$$

$$Z = \sum_{v \in rv(gr(G))} \prod_{f \in gr(G)} f_i(\pi_{rv(f_i)}(v))$$
QA: Queries

- **Marginal distribution**
  - $P(\text{Sick}(\text{eve}))$
  - $P(\text{Travel}(\text{eve},) \text{ Treat}(\text{eve}, m_1))$

- **Conditional distribution**
  - $P(\text{Sick}(\text{eve})|\text{Epid})$
  - $P(\text{Epid}|\text{Sick}(\text{eve}) = \text{true})$

- **Most probable assignment**

\[
\arg\max_{v \in r(R)} \sum_{v \in r(R)} P_F(v), R \subseteq \text{rv(gr(G))}
\]

- **MPE:** $R = \text{rv(gr(G))}$
- **MAP:** $R = \{\text{Travel}(\text{eve}), \text{Man(virus)}\}$

Avoid groundings!
QA: Lifted VE (LVE)

- Eliminate all variables not appearing in query
- **Lifted summing out**
  - Sum out representative instance
  - Exponentiate result for isomorphic instances
- Preconditions exist!
- Count conversion (not part of this tutorial)

Avoid groundings!

Poole (2003), de Salvo Braz et al. (2005, 2006), Milch et al. (2008), Taghipour et al. (2012, 2013)
QA: LVE in Detail

- E.g., marginal
  - \(P(\text{Travel(eve)})\)
  - Split w.r.t. \textit{Travel(eve)}
  - Eliminate all non-query variables
  - Normalise

Poole (2003), de Salvo Braz et al. (2005, 2006), Milch et al. (2008), Taghipour et al. (2013, 2013a)
Problem: Many Queries

- **Set of queries**
  - $P(\text{Travel}(\text{eve}))$
  - $P(\text{Sick}(\text{bob}))$
  - $P(\text{Treat}(\text{eve}, m_1))$
  - $P(\text{Epid})$
  - $P(\text{Nat}(\text{flood}))$
  - $P(\text{Man}(\text{virus}))$
  - Combinations of variables

- **Under evidence**
  - $\text{Sick}(X') = \text{true}$
  - $X' \in \{\text{alice, eve}\}$

- (L)VE starts with complete model for QA
Solution: Junction Tree Algorithm

- Identify clusters in model to form junction tree

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990)
Junction Tree: Data Structure

- **Clusters**: sets of variables from underlying model

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990)
Junction Tree: Definition

• Junction tree $J = (V, E)$ for model $F$
  • $V$ set of nodes, a node = cluster $C$
  • $E$ set of edges
  1. A cluster $C_i$ is a set of variables from $F$
  2. For every factor $f$ in $F$, its arguments appear in some cluster $C_i$
  3. If a variable from $F$ appears in clusters $C_i$ and $C_j$, it must appear in every cluster $C_k$ on the path between nodes $i$ and $j$.

• Each cluster has a local model $F_i :$ set of factors
  • factor arguments subset of cluster
  • $\bigcup_{i \in V} F_i = F$  \hspace{0.5cm} (\text{$F_i$ partition $F$})

• Separator: shared variables of edges $(i, j) \in E$
Junction Tree: Components

- Clusters sufficient for QA after some preprocessing
Junction Tree: Message Passing

Lauritzen and Spiegelhalter (1988), Shenoy and Shafer (1990)

- Distribute local information by messages
  1. If a cluster received messages from all neighbours but one, it sends message to remaining neighbour
  2. If a cluster received all messages, it send messages to all neighbours that have not received a message yet

- **Message**: VE on local model and other messages with query over separator

- Local computations correctness:
  - Valid junction tree (w.r.t. properties)
  - Combination & marginalisation (in the form of multiplication & summing out)

- QA on local models and messages
  - Use cluster that contains query variable
Junction Tree: Messages

• From periphery to centre and back

Shafer and Shenoy (1989)
Junction Tree: Messages

- Message: eliminate all non-separator variables
- E.g., cluster with $f_3^3$ in local model

<table>
<thead>
<tr>
<th>Sick. alice</th>
<th>Epid</th>
<th>$\sum$</th>
<th>$\text{eat.,alice.m}_1$</th>
<th>$f_3^3$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>11</td>
<td>false</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>7</td>
<td>true</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>8</td>
<td>false</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>15</td>
<td>true</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td>false</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td>true</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td></td>
<td>false</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td></td>
<td>true</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Shafer and Shenoy (1989)
Junction Tree: Messages

- From periphery to centre and back

Shafer and Shenoy (1989)
QA: Query Answering with JT

- Eliminate all variables not appearing in query
  - Through summing out
- E.g., marginal
  - $P(\text{Travel.eve})$

$$P(\text{Travel.eve}) \propto \sum_{e \in r(\text{Epid})} m_{e,\text{back}}(e) \sum_{s \in r(\text{Sick.eve})} f_2^1(\text{Travel.eve}, e, s)m_{m_1}(e, s)m_{m_1}(e, s)$$

Shafer and Shenoy (1989)

2 sums

Diagram:
- Epid
  - $f_2^1$ $m_{m_1}$ $m_{e,\text{back}}$
  - Epid Sick.eve
  - $f_2^1$ $m_{m_1}$ $m_{m_2}$
- Travel.eve
- Sick.eve
Junction Tree: Symmetry→ Inefficiency

- Identical messages incoming
- Information already present
- Calculating identical messages + sending information partially present

\[ m_{a,\text{back}} \]

\[ m_{\text{eve}} : \text{Eliminate } \text{Travel.eve, Sick.eve} \]
from \( f_2^2, m_{p_1}, m_{p_2} \)

\[ m_{m_1} : \text{Eliminate } \text{Treat.eve}.m_1 \]
from \( f_3^1 \)

\[ m_{m_{1,\text{back}}} \]

\[ f_3^2 \]

\[ m_{m_{1,\text{back}}} \]

\[ f_3^5 \]

\[ m_{m_{2,\text{back}}} \]

\[ m_{m_2} : \text{Eliminate } \text{Treat.eve}.m_2 \]
from \( f_3^2 \)

\[ f_1^3 f_1^4 m_{\text{eve}} m_{\text{bob}} m_{\text{alice}} m_{\text{war}} \]
First-order Junction Tree: FO jtree

- Parameterised clusters = parclusters
  - Nodes of FO jtree
  - Shared PRVs between neighbours = separators
  - Parclusters have local models of parfactors

- FO jtree for parfactor model $G$:
  - Analogous definition, junction tree properties apply
  - Message passing, QA analogous

\[
\begin{align*}
\text{Epid Nat}(D) & \quad \text{Man}(W) \quad \text{Epid Sick}(X) \\
& \quad \text{Travel}(X) \quad \text{Treat}(X, M)
\end{align*}
\]

Sparse encoding!
Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017)

- **Input**
  - Model $G$
  - Evidence $E$
  - Queries $Q$

- **Algorithm**
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $E$ into $J$
  3. Pass messages in $J$
     - Inbound
     - Outbound
  4. Answer queries $Q$

Queries on grounded PRVs, e.g.,
- $Travel(eve)$, $Treat(eve, p_1)$, $Epid$

Diagram:
- $Nat(D)$
- $Man(W)$
- $Epid$
- $Travel(X)$
- $Treat(X, M)$
- $Sick(X)$

Nodes and edges with $g_1$, $g_2$, $g_3$.
LJT: Example Input

- Model $G = \{g_i\}_{i=1}^{3}$
  - $g_1(Epid, Nat(D), Man(W))$
  - $g_2(Travel(X), Epid, Sick(X))$
  - $g_3(Epid, Sick(X), Treat(X, M))$
  → Including function specification

- Evidence $E = \{Sick(alice) = true, Sick(eve) = true\}$

- Queries $Q = \{Travel(eve), Epid\}$

- Algorithm
  1. Build FO jtree $J$ for $G$
FO Jtree Construction

• Propositional junction tree construction
  • Triangulation, compute maximum spanning tree, ...
  • Hypergraph partitioning
  • Decomposition tree (dtree), clusters, ...

• First-order: logical variables
  • First-order decomposition trees (FO dtrees)
  • FO dtrees have node properties (cutset, context, cluster)
  • (FO) dtree + clusters = (FO) jtree
  • Heuristic to build an FO dtree
    (logical variables guide the construction)

Taghipour et al. (2013b)
Lifted Junction Tree Algorithm: LJT

• Input
  • Model $G$
  • Evidence $E$
  • Queries $Q$

• Algorithm
  1. Build FO jtree $J$ for $G$

  2. Enter evidence $E$ into $J$
LJT: Enter Evidence

- Evidence as a set of events
  - \( E = \{\text{Sick}(\text{eve}) = \text{true}, \text{Sick}(\text{alice}) = \text{true}\} \)
- Evidence as a parfactor
  - \( g_E(\text{Sick}(X')) \)
  - \( X' \in \{\text{eve, alice}\} \)
- Function specification

<table>
<thead>
<tr>
<th>Sick((X'))</th>
<th>(g_E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>1</td>
</tr>
</tbody>
</table>

- At every parcluster that contains evidence variables, enter evidence
LJT: Enter Evidence

• At every parcluster that contains evidence variables
  • \( g_E(Sick(X')) \), \( X' \in \{eve, alice\} \)
  
• Parclusters
  • \( Sick(X') \not\subseteq C_1 \)
  • \( Sick(X') \subseteq C_2 \)
  • \( Sick(X') \subseteq C_3 \)

• Enter evidence at \( C_2 \) and \( C_3 \)
LJT: Enter Evidence

- At every parcluster that contains evidence variables
  - \( g_E(Sick(X')) \), \( X' \in \{ \text{eve, alice} \} \)
  - Parclusters \( C_2 \) and \( C_3 \)

  \[
  \begin{array}{c}
  \text{C}_1 \\
  \text{Epid Nat}(D) \\
  \text{Man}(W) \\
  \end{array}
  \begin{array}{c}
  \text{C}_2 \\
  \text{Epid Sick}(X) \\
  \text{Travel}(X) \\
  \end{array}
  \begin{array}{c}
  \text{C}_3 \\
  \text{Epid Sick}(X) \\
  \text{Treat}(X, M) \\
  \end{array}
  \]

  \( g_1 \)  \( g_2 \)  \( g_3 \)

- Enter evidence at \( C_2 \) (\( C_3 \) analogous)
  - Split local model
    - \( X \in \{ \text{bob, ...} \} \)
  - Absorb evidence in \( g'_2 \)
Lifted Junction Tree Algorithm: LJT

- **Input**
  - Model $G$
  - Evidence $E$
  - Queries $Q$

- **Algorithm**
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $E$ into $J$
  3. Pass messages in $J$

Braun and Möller (2017)
LJT: Pass Messages

- **Separators**
- **Messages**
  - **Inbound**
    - $m_{12}$ from $C_1$ to $C_2$ over $Epid$
    - $m_{32}$ from $C_3$ to $C_2$ over $Epid, Sick(X)$
  - **Outbound**
    - $m_{21}$ from $C_1$ to $C_2$ over $Epid$
    - $m_{23}$ from $C_3$ to $C_2$ over $Epid, Sick(X)$
LJT: Example Message Inbound

- $m_{32}$ from $C_3$ to $C_2$
  - Eliminate $Treat(X, P), Treat(X', P)$

\[ m_{32} = \{\hat{g}_3, \hat{g}'_3\} \]
LJT: Messages at $C_2$

- After $m_{12}$ and $m_{32}$ arrived

$m_{12} = \{\hat{g}_1\}$

$m_{32} = \{\hat{g}_3, \hat{g}_3'\}$

\[ C_1 \]
\begin{align*}
Epid Nat(D) Man(W) \\
g_1
\end{align*}

\[ C_2 \]
\begin{align*}
Epid Sick(X) Travel(X) \\
g_2, g_2' \\
m_{12}, m_{32}
\end{align*}

\[ C_3 \]
\begin{align*}
Epid Sick(X) Treat(X, M) \\
g_3, g_3'
\end{align*}
LJT: Example Message Outbound

- $m_{21}$ from $C_2$ to $C_1$
  - Eliminate $\textit{Sick}(X), \textit{Travel}(X), \textit{Travel}(X')$ from $g_2, g_2', m_{32}$
LJT: Example Message Outbound

- $m_{23}$ from $C_2$ to $C_3$
  - Eliminate $\text{Travel}(X), \text{Travel}(X')$ from $g_2, g'_2, m_{12}$
Lifted Junction Tree Algorithm: LJT

- **Input**
  - Model $G$
  - Evidence $E$
  - Queries $Q$

- **Algorithm**
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $E$ into $J$
  3. Pass messages in $J$
  4. Answer queries $Q$

Braun and Möller (2017)
LJT: Answer Queries

- Queries $Q = \{Travel(eve), Epid\}$
- For each query $Q$
  - Find parcluster that contains $Q$
  - Extract submodel of local model and messages
  - Use LVE to answer $Q$
LJT: Answer Queries

- $Q_1 = \text{Travel(eve)}$
  - Find parcluster: $C_2$
  - Extract submodel: $G' = \{g_2, g_2', m_{12}, m_{32}\}$
  - Answer \text{Travel(eve)} with LVE
LJT: Answer Queries

- $Q_2 = \textit{Epid}$
  - Find parcluster: $C_1$ (any of the three parclusters)
  - Extract submodel: $G' = \{g_1, m_{21}\}$
  - Answer $\textit{Epid}$ with LVE

![Diagram of LJT model with parclusters and submodels]
LJT: Analysis

• Static overhead
  • Construction
  • Evidence entering
  • Message passing

• Payoff during QA
  • Multiple queries
  • Complexity of LVE for one query = Complexity of message pass in LJT

Queries all under the same evidence

\[ E = \{\text{Sick(eve)} = \text{true}, \text{Sick(alice)} = \text{true}\} \]
Extending LVE and LJT

Probability queries, based on LVE

• Liftable models (fusion)

• Conjunctive queries
  • \( P(Epid, \text{Travel} (eve)) \)

• Isomorphic query terms (parameterised queries)
  • \( P(\text{Sick} (eve), \text{Sick} (alice), \text{Sick} (bob)) \equiv P(\text{Sick} (X)) \)
Isomorphic Conjunctive Queries

- $Q = \text{Sick}(\text{alice}) \land \text{Sick}(\text{eve}) \land \text{Sick}(\text{bob})$

- Same parcluster, easy!

- But
  - Split model: grounding for constants in query
  - Eliminate non-query variables: many identical sums, large intermediate factors
  - Normalise: large result with symmetries

| Implicit Parcluster | Grounding | | | g |
|---|---|---|---|
| false | false | false | 1 |
| false | false | true | 2 |
| false | true | false | 2 |
| false | true | true | 3 |
| true | false | false | 2 |
| true | false | true | 3 |
| true | true | false | 3 |
| true | true | true | 4 |

Braun and Möller (2018a)
Parameterised Queries

- Compact query representation
- Lifted computations during LVE
- Compact result representation

\[ Q = \textit{Sick}(X) \]

Result: \( \#_x[Sick(X)] \)

<table>
<thead>
<tr>
<th>( #_x[Sick(X)] )</th>
<th>( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,3]</td>
<td>1</td>
</tr>
<tr>
<td>[1,2]</td>
<td>2</td>
</tr>
<tr>
<td>[2,1]</td>
<td>3</td>
</tr>
<tr>
<td>[3,0]</td>
<td>4</td>
</tr>
</tbody>
</table>
Grounding Parameterised Queries

- \( g(\text{Sick}(X), \text{Treat}(X, P), \text{Effectiveness}(P)) \)

- \( Q = \text{Treat}(X, M) \)
  - LVE grounds one logical variable

- \( Q = \text{Sick}(X), \text{Treat}(X, M), \text{Effectiveness}(M) \)
  - LVE grounds one logical variable
  - Compare forbidden queries in PDBs (Dalvi & Suciu 12)
Extending LVE and LJT

Additional inference targets

• Most probable assignment (MPE, MAP)
  • New argmax operators

• Dynamic inference (filtering, prediction, hindsight)
  • Lifted dynamic junction tree algorithm

• Adaptive inference (incremental changes)
  • Evidence, model structure, parfactors
    → Adaptive steps of LJT
What about time?

• Based on interface algorithm, dynamic Bayes nets

• Dynamic parfactor graphs
  • Another application area: Health care

• Lifted Dynamic Junction Algorithm
  • Answer filtering, prediction, hindsight queries

Murphy (2002), Gehrke et al. (2018, 2019)
Does it have to be LVE in LJT?

LJT with **LVE** &

**First-order Knowledge Compilation (FOKC)** to solve a WFOMC problem

- LVE for evidence entering and message passing
- FOKC for query answering
Does it work in practice?

• Prototype implementation
  • LVE by Taghipour
    https://dta.cs.kuleuven.be/software/gcfove
  • LJT
    (in preparation)

MISSING
Scalable implementation with DB backing
Message passing = Propagation in advance
as in TensorLog (Cohen 2016, Cohen et al. 2017)
Grounded vs. Lifted Inference

Runtimes in milliseconds, one query, domain sizes between 2 and 1000, evidence at 0%
LJT: Payoff versus LVE

Accumulated runtimes in milliseconds, seven queries, grounded model size: 100,000
Evidence at 0% and 20% on PRVs with one parameter
LJT: Evidence

Runtimes in milliseconds for each LJT step
Evidence from 0% to 100% in 10% steps on PRVs with one parameter
LJT: Parameterised Queries

Run times in milliseconds, one query, grounded model size between 10 and 100,000
Outlook

• Continue optimising
  • Parallelisation
  • Caching

• From discrete over interval to continuous ranges

• Learning?
  • Structure
  • Potentials
  • Symmetries
  • *Transfer learning*

• Open world?
  • Unknown domains
  • Unknown behaviour
Wrap-up Exact Lifted Inference

• Parfactor models for **sparse encoding**
  • Factorisation of full joint distribution
  • Logical variables to model objects
• Algorithms for exact query answering
  • LVE for single inference
  • LJT for repeated inference
• Extensions possible
  • Parameterised, conjunctive queries
  • Temporal models
  • Incremental changes
  • Assignment queries

Next: Approximate Lifted Inference
Mission and Schedule of the Tutorial*

Providing an overview and a synthesis of StaR AI

- Introduction
  - StaR AI

- Overview: Probabilistic relational modeling
  - Semantics (grounded-distributional, maximum entropy)
  - Inference problems and their applications
  - Algorithms and systems
  - Scalability (limited expressivity, knowledge compilation, approximation)

- Scalability by lifting
  - Exact lifted inference
  - Approximate lifted inference

- Summary

10 min  ✓
40 min ✓
40+50 min ✓
30 min
10 min

*Thank you to the SRL/StaRAI crowd for all their exciting contributions! The tutorial is necessarily incomplete. Apologies to anyone whose work is not cited.
References

• **Apsel and Brafman (2012)**

• **Cohen (2016)**

• **Cohen et al. (2017)**

• **Dalvi and Suciu (2012)**

• **Dawid (1992)**
References

• Dechter (1999)

• De Salvo Braz et al. (2005)

• De Salvo Braz et al. (2006)

• Jensen et al. (1990)
References

• **Koller and Friedman (2009)**

• **Lauritzen and Spiegelhalter (1988)**

• **Milch et al. (2008)**

• **Murphy (2002)**
References

• **Poole (2003)**

• **Shafer and Shenoy (1989)**

• **Sharma et al. (2018)**

• **Shenoy and Shafer (1990)**
References

• Taghipour et al. (2013)

• Taghipour et al. (2013a)

• Taghipour et al. (2013b)

• Zhang and Poole (1994)
Work @ IFIS

• Braun and Möller (2016)

• Braun and Möller (2017)

• Braun and Möller (2017a)
Work @ IFIS

• **Braun and Möller (2018)**

• **Braun and Möller (2018a)**

• **Braun and Möller (2018b)**

• **Braun and Möller (2018c)**
Work @ IFIS

• Gehrke et al. (2018)

• Gehrke et al. (2019)
Appendix

Extensions to LVE and LJT
New Evidence? Back to Step 2

• Input
  • Model $G$
  • Evidence $E = \{\text{Epid} = \text{true}\}$
  • Queries $Q$

• Algorithm
  1. Build FO jtree $J$ for $G$

  2. Enter evidence $E$ into $J$
And continue...

- **Input**
  - Model $G$
  - Evidence $E = \{\text{Epid} = \text{true}\}$
  - Queries $Q$

- **Algorithm**
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $E$ into $J$
  3. Pass messages

![Diagram showing the FO jtree structure with nodes and edges labeled with actions such as Epid Nat(D), Man(W), Epid Sick(X), Travel(X), Treat(X, M), and messages $g'_1$, $g'_2$, $g'_3$.]
What about incremental changes?

• Input
  - Model $G$
  - Evidence $E = \{Epid = true\} \cup \{Travel(eve) = true\}$
  - Queries $Q$

• Algorithm
  1. Build FO jtree $J$ for $G$
     
     $C_1$
     
     Epid Nat($D$) Man($W$)
     
     $g_1'$

     $C_2$
     
     Epid Sick($X$) Travel($X$)
     
     $g_2'$

     $C_3$
     
     Epid Sick($X$) Treat($X, M$)
     
     $g_3'$

  2. Enter evidence $E$ into $J$ incrementally

Braun and Möller (2018)
What about incremental changes?

- **Input**
  - Model $G$
  - Evidence $E = \{Epid = true\} \cup \{Travel(eve) = true\}$
  - Queries $Q$

- **Algorithm**
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $E$ into $J$ adaptively

```
C_1

Epid Nat(D)
Man(W)

g'_1
```

```
C_2

Epid Sick(X)
Travel(X)

g'_2, g''_2
```

```
C_3

Epid Sick(X)
Treat(X, M)

g'_3
```

3. Pass messages adaptively

Braun and Möller (2018)
Adaptive Messages

• Calculate new message only if
  • Local model changed
  • Incoming message changed
  • Existing messages invalid

• Else
  • Send an empty message
  • Existing messages still valid

\[ C_3 \text{ calculates new messages } m_{21}, m_{23} \]

\[ C_1, C_2 \text{ send empty messages} \]

\( C_1 \)
\[ \text{Epid Nat}(D) \]
\[ \text{Man}(W) \]
\[ g_1', m_{21} \]

\( C_2 \)
\[ \text{Epid Sick}(X) \]
\[ \text{Travel}(X) \]
\[ g_2', g_2'', m_{12}, m_{32} \]

\( C_3 \)
\[ \text{Epid Sick}(X) \]
\[ \text{Treat}(X, M) \]
\[ g_3', m_{23} \]
Changes: Adaptive Inference

Braun and Möller (2018)

• Changes
  • Domains
  • Parfactors
    • Changing the structure: new/removed argument
    • Without changing the model structure: new range
  • Evidence (additional, retracted, new)

• Adaptive LJT
  • Adapt FO jtree $J$ to changes in $G$
  • Adaptively enter evidence
    • Changes in evidence $E$
    • Changes in $J$
  • Adaptively pass messages
  • Answer queries

$J$ may require reconstruction after a while
LJT: Conjunctive queries?

Koller and Friedman (2009), Braun and Möller (2017a)

- **Conjunctive query**: $Epid \land Travel(eve)$
  - QA as before: Eliminate all non-query variables

- **Conjunctive query**: $Travel(eve) \land Treat(eve, m_1)$
  - Query variables not in one parcluster!
  - Duplicate information in messages!
LJT: Answer Conjunctive Queries

Koller and Friedman (2009), Braun and Möller (2017a)

• Queries $Q = \{Q_1, Q_2, Q_3, Q_4\}$
  • $Q_1 = Travel(eve) \land Treat(eve, m_1)$
  • $Q_2 = Epid \land Travel(eve)$
  • $Q_3 = Travel(eve)$
  • $Q_4 = Epid$

• For each query $Q$
  • Find *subtree* that covers the query variables
  • Extract *submodel* without duplicate information
  • Use LVE to answer query

\[ C_1 \]
\[ \text{Epid Nat}(D) \]
\[ \text{Man}(W) \]
\[ g_1, m_{21} \]

\[ C_2 \]
\[ \text{Epid Sick}(X) \]
\[ \text{Travel}(X) \]
\[ g_2, g'_2, m_{12}, m_{32} \]

\[ C_3 \]
\[ \text{Epid Sick}(X) \]
\[ \text{Treat}(X, M) \]
\[ g_3, g'_3, m_{23} \]
LJT: Answer Conjunctive Queries

• $Q_1 = \text{Travel}(\text{eve}) \land \text{Treat}(\text{eve}, m_1)$
  • Find subtree that covers the query variables
  • Extract submodel without duplicate information
  • Use LVE to answer query

Braun and Möller (2017a)
LJT: Answer Conjunctive Queries

- Queries \( Q = \{Q_1, Q_2, Q_3, Q_4\} \)
  - \( Q_1 = Travel(eve) \land Treat(eve, m_1) \)
  - \( Q_2 = Epid \land Travel(eve) \)
  - \( Q_3 = Travel(eve) \)
  - \( Q_4 = Epid \)

- For each query \( Q \)
  - Find **subtree** that covers the query variables
  - Extract **submodel** without duplicate information
  - Use LVE to answer query

Subtree = Single parcluster

Braun and Möller (2017a)
QA: Most probable assignment

Dawid (1992), Dechter (1999), de Salvo Braz et al. (2006), Apsel and Brafman (2012), Braun and Möller (2018b)

- To all variables: most probable explanation (MPE)

\[
\text{argmax}_{rv(G)} P_G
\]

- (L)VE: maxing out (instead of summing out)
- (L)JT: message calculation by maxing out
- Isomorphic instances: identical most probable assignment

- As before: construction, evidence entering

- Message passing
  - Only one message pass (from periphery to centre)
  - Max out remaining variables at centre
LJT: MPE

- Epid = false
- \( \forall D, W : \text{Nat}(D) = false, \text{Man}(W) = false \)
- \( \forall X' \in \{alice, eve\}, P : \text{Travel}(X') = true, \text{Treat}(X', P) = true \)
- \( \forall X, M : \text{Travel}(X) = true, \text{Sick}(X) = true, \text{Treat}(X, M) = true \)
QA: Most probable assignment

• To subset of variables: maximum a posteriori (MAP)

\[
\arg\max_{Travel.eve,Epid} \sum_{r(Nat(D), Man(W), Sick(X), Treat(X,P)), r(Travel(X)), X \neq eve} P_G
\]

• MAP a more general case of MPE

• \(\sum\) and \(\max\) not commutative!

• As before: construction, evidence entering, message passing (with summing out)

• Answer MAP query: get submodel (as before)
  • Sum out non-query variables, max out query variables

• Assignment to complete parclusters safe

Dechter (1999), Sharma et al. (2018)
Braun and Möller (2018b)
Does LJT lift what is liftable?

- Original LJT had avoidable groundings

- After fusion, answer is YES!

Braun and Möller (2017)
Appendix

FO jtree construction
FO Jtree Construction

• Propositional junction tree construction
  • Triangulation, compute maximum spanning tree, ...
  • Hypergraph partitioning
  • Decomposition tree (dtree), clusters, ...

• First-order: logical variables
  • First-order decomposition trees (FO dtrees)
  • FO dtrees have node properties (cutset, context, cluster)
  • (FO) dtree + clusters = (FO) jtree
  • Heuristic to build an FO dtree
  (logical variables guide the construction)

Taghipour et al. (2013b)
FO Dtree: Data Structure

- (L)VE decomposes model into subproblems
- Represent as tree

Taghipour et al. (2013b)
FO Dtree: Cutset, Context, Cluster

Taghipour et al. (2013b)
FO Dtreet to FO Jtree

• Compute clusters per node and minimise

Braun and Möller (2016)
Minimising an FO Jtree

- **FO jtree is minimal**
  - If by removing a variable from any parcluster, the FO jtree stops being an FO jtree

- **Merge parclusters**
  - If neighbouring parclusters are subsets of each other

---

Braun and Möller (2016)