Relevance-based Incremental Belief Updating in Bayesian Network

by

Yan Lin & Marek J. Druzdzel
Relevance Based Incr. Belief Update

- **Belief Updating**
  - Exact Infrence = NP
  - Complexity of Algorithmes are exponential to the size of the BN

- **Relevance Based Reasoning**
  - Pruning the BN → Improved efficiency
Relevance Based Incr. Belief Update

- Relevance Reasoning
  - Evidence → Target Nodes

- Ex. Diabeties.

- Techniques
  - d- separation
  - Barren Nodes
  - Relevance based Decomposition
Barren Node

• Barran Node
  • Neither target nor evidence node.
  • Have no descendans
    – If they do, then all their children are Barren Nodes
Barren Node

- \( Ev = \{posX, \text{Smoking}\} \)
- \( TN = \{\text{TuberC}, \text{LungC}\} \)
• Eq. in respect to compute postr.probability of TN given Ev
Relevance Based Decomposition

- Given Evidence Nodes
- Given Target Nodes
- Divide the BN into SubNetworks, by only focusing on parts of the target nodes.
Relevance based Decomposition

- $E_v = \{\text{PosX}\}$
- $T_N = \{\text{TripToAsia, Bronchitis, Dyspnea}\}$
T = \{\text{TripAisia}\}
$T = \{\text{Bronchitis}\}$
$T = \{ \text{Dyspnea} \}$
Relevance based incremental belief updating

- Incremental Belief gathering
  - Recompute CPT each time.
  - Can take too long time.
  - Ex. Robot.

- Use relevance reasoning to minimize complexity.
Relevance based incremental belief updating

- Relevance based incremental belief updating
  - Proposed technique
  - New evidence → invalidate posterior belief of nodes probabilistic dependent on the new evidence.
  - Update invalidated nodes, Keep the rest of BN intact.
  - Use relevance reasoning → calculate CPT
Relevance based incremental belief updating

• Idea
  • All nodes equipped with flags
    – Flag: Valid or Invalid
  • Evidence:
    – Nodes, not d-separated from the evidence node, is marked invalid.
• Use Relevance reasoning recalculate CPT.
  – Not all invalid nodes are updated.
Relevance based incremental belief updating

- Given: BN net, TargetN. T, evidence E

Void New_Ev(net, E)

For all nodes n in net, not d-separated from evidence e in E → set n.valid = false

Void Inc_update(net, T, E)

Remove from T, nodes t where t.valid= true
Use Relevance reasoning to discard from net, all nodes not relevant for updating T given E

Main(net, E)

Create junction tree J from net
New_Ev(net, E)
Inc_update(net, T, E)
if( cost.inf(J) > cost.inf(pruned net)) {Do inf(pruned net) } else {Do inf(J)}
Void New_Ev(\textit{net, E})

For all nodes \(n\) in \(\textit{net}\), not \textit{d}-seperated from evidence \(e\) in \(E\) \n\rightarrow set \(n.\texttt{valid} = \texttt{false}\)

Void Inc_update(\textit{net, T, E})

Remove from \(T\), nodes \(t\) where \(t.\texttt{valid} = \texttt{true}\)

Use Relevance reasoning to discard from \(\textit{net}\), all nodes not relevant for updating \(T\) given \(E\)

Main(\textit{net, E})

Create junction tree \(J\) from \(\textit{net}\)

New_Ev(\textit{net, E})

Inc_update(\textit{net, T, E})

if(cost.\texttt{inf}(J)>cost.\texttt{inf} (\textit{pruned net})) \{\textbf{Do} cost.\texttt{inf} (\textit{pruned net}) \} else
\{\textbf{Do} cost.\texttt{inf}(J)\}
Void New_Ev($net, E$)

For all nodes $n$ in $net$, not $d$-seperated from evidence $e$ in $E$ → set $n.valid = false$

Void Inc_update($net, T, E$)

Remove from $T$, nodes $t$ where $t.valid = true$

Use Relevance reasoning to discard from $net$, all nodes not relevant for updating $T$ given $E$

Main($net, T\{A, M\} , E = \{D, F\}$

Create junction tree $J$ from $net$

New_Ev($net, E$)

Inc_update($net, T, E$)

if($\text{cost.inf}(J) > \text{cost.inf( pruned } net))$
{Do $\text{inf( pruned } net)$ } else {Do $\text{inf}(J)$}
Void New_Ev(net, E = \{D, F\})

For all nodes \(n\) in \(net\), not \(d\)-seperated from evidence \(e\) in \(E\) → set \(n.valid = false\)

Void Inc_update(net, T, E)

Remove from \(T\), nodes \(t\) where \(t.valid = true\)

Use Relevance reasoning to discard from \(net\), all nodes not relevant for updating \(T\) given \(E\)

Main(net, \(T=\{A, M\}\), \(E = \{D, F\}\))

Create junction tree \(J\) from \(net\)

New_Ev(net, E = \{D, F\})

Inc_update(net, T, E)

if(cost.inf(J)>cost.inf(pruned net))
{Do inf(pruned net) } else {Do inf(J)}
Void New_Ev($net, E$)

For all nodes $n$ in $net$, not d-separated from evidence $e$ in $E \rightarrow$ set $n.valid = false$

Void Inc_update($net, T=\{A, M\}, E = \{D, F\}$)

Remove from $T$, nodes $t$ where $t.valid = true$

Use Relevance reasoning to discard from $net$, all nodes not relevant for updating $T$ given $E$

Main($net, T=\{A, M\} , E = \{D, F\}$)

Create junction tree $J$ from $net$

New_Ev($net, E$)

Inc_update($net, T=\{A, M\}, E = \{D, F\}$)

if(cost.inf($J$)>cost.inf(pruned $net$)) {Do inf(pruned $net$) } else {Do inf($J$)}
Void New_Ev(net, E)

For all nodes n in net, not d-separated from evidence e in E → set n.valid = false

Void Inc_update(net, T={A, M}, E = {D, F})

Remove from T, nodes t where t.valid= true

Use Relevance reasoning to discard from net, all nodes not relevant for updating T={M} given E

Main(net, T={A, M} , E = {D, F})

Create junction tree J from net

New_Ev(net, E)

Inc_update(net, T={A, M}, E = {D, F})

if(cost.inf(J)>cost.inf(pruned net)) {Do inf(pruned net) } else {Do inf(J)}
Void New_Ev(\textit{net}, \textit{E})

For all nodes \textit{n} in \textit{net}, not d-separated from evidence \textit{e} in \textit{E} \rightarrow \textit{set n.valid = false}

Void Inc_update(\textit{net}, \textit{T}=%1\{A, M\}, \textit{E} = \{D, F\})

Remove from \textit{T}, nodes \textit{t} where \textit{t.valid= true}

Use Relevance reasoning to discard from \textit{net}, all nodes not relevant for updating \textit{T}=%1\{M\} given \textit{E}

Main(\textit{net}, \textit{T}=%1\{A, M\}, \textit{E} = \{D, F\})

Create junction tree \textit{J} from \textit{net}

New_Ev(\textit{net}, \textit{E})

Inc_update(\textit{net}, \textit{T}=%1\{A, M\}, \textit{E} = \{D, F\})

\textbf{if} (\textit{cost.inf(J)} > \textit{cost.inf(pruned net)})

\{Do \textit{inf(pruned net)} \} \textbf{else} \{Do \textit{inf(J)}\}

\textbf{We can predict the complexity of the pruned sub-network by the number of potentials generated from triangulation.}
Junction Trees
Incrementality

- Query Incrementality
- Evidence Incrementality
- Representation Incrementality
  - Model Revision
Representation Incrementality

- Adding/Removing Nodes
  - Remove arcs

- Adding/Removing Arcs
  - Remove and adding arcs → change the conditional probability of the node it points to.
  - Idea: Mark the node and relevant nodes as invalid and update CPT.
Void Remove_Arc(BN Net, Ev E, arc a)

n:=node which a points to.
n.valid=false
Change_CPT(net, n, E)
remove a

Void Change_CPT(BN net, node n, Ev E)

Each child c of n
  c.valid=false
  each node m in net, relevant to c given E
  m.valid=false
if(n or n.decedant = E)
  for each parent p of n
    p.valid = false
    each node m in net relevant to p given E
    m.valid = false
Change CPT
Void Remove_Arc(BN Net, Ev E, arc a)

n:=node which a points to.
n.valid=false

Change_CPT(net, n, E)
remove a

Void Change_CPT(BN net, node n, Ev E)

Each child c of n

c.valid=false

each node m in net, relevant to c given E

m.valid=false

if(n or n.decedant = E)

for each parent p of n

p.valid = false

each node m in net relevant to p given E

m.valid = false

Change CPT
Discussion

• Empirical results
  • BN with 422 nodes.
  • Machine
    – 2 UltraSPARC cpu's running at 168 Mhz
    – L2 cache: 0.5 MB
    – Ram: 384

• Not efficient on small BN

• Genie