Local Structure Discovery in Bayesian Networks

Teppo Niinimäki, Pekka Parviainen
August 18, 2012

University of Helsinki
Department of Computer Science
Outline

Introduction

Local learning

From local to global

Summary
Outline

Introduction

Local learning

From local to global

Summary
Bayesian network

- Structure: directed acyclic graph $A$

- Conditional probabilities

$$p(x) = \prod_{v \in N} p(x_v | x_{A_v})$$
Global structure learning

Data $D$
- $n$ variables
- $m$ samples from distribution $p$

**Task:** Given data $D$ learn the structure $A$. 

<table>
<thead>
<tr>
<th>sample</th>
<th>variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 ... 8</td>
</tr>
<tr>
<td>1</td>
<td>2 1 0 ... 1</td>
</tr>
<tr>
<td>2</td>
<td>2 0 2 ... 0</td>
</tr>
<tr>
<td>3</td>
<td>2 0 1 ... 0</td>
</tr>
<tr>
<td>4</td>
<td>1 0 2 ... 1</td>
</tr>
<tr>
<td>...</td>
<td>... ... ...</td>
</tr>
<tr>
<td>5000</td>
<td>2 2 1 ... 1</td>
</tr>
</tbody>
</table>
Global structure learning

Data $D$
- $n$ variables
- $m$ samples from distribution $p$

Task: Given data $D$ learn the structure $A$.

Approaches:
- constraint-based: (pairwise) independency tests
- score-based: (global) score for each structure
Score-based structure learning

Structure learning is NP-hard.

Dynamic programming algorithm

- \( O(n^22^n) \) time
- \( O(n2^n) \) space

\( \Rightarrow \) Exact learning infeasible for large networks.
Score-based structure learning

Structure learning is NP-hard.

Dynamic programming algorithm

- $O(n^22^n)$ time
- $O(n2^n)$ space

⇒ Exact learning infeasible for large networks.

Note: Linear programming based approaches can scale to very large networks.

- However, no good worst case guarantees in general case.
Local structure learning

- interested in target node(s)
- learn local structure near target(s)
- constraint-based HITON, GLL framework (Aliferis, Statnikov, Tsamardinos, Mani and Koutsoukos, 2010)
Local structure learning

- interested in target node(s)
- learn local structure near target(s)
- constraint-based HITON, GLL framework (Aliferis, Statnikov, Tsamardinos, Mani and Koutsoukos, 2010)

**our contribution:** score-based variant: SLL
Local to global learning

- learn local structure for each node
- combine results to global structure
- MMHC, LGL framework (Tsamardinos et al., 2006; Aliferis et al., 2010)
Local to global learning

- learn local structure for each node
- combine results to global structure
- MMHC, LGL framework (Tsamardinos et al., 2006; Aliferis et al., 2010)
- **our contribution**: two SLL-based algorithms: SLL+C, SLL+G
Outline

Introduction

Local learning

From local to global

Summary
Local learning problem

Markov blanket = **neighbors** + **spouses**

- Given its Markov blanket a node is independent of all other nodes in the network.
- For example, if predicting the value of target node $t$ the rest of the nodes can be ignored.
Local learning problem

Markov blanket $= \text{neighbors} + \text{spouses}$

**Task:** Given data $D$ and a target node $t$ learn the Markov blanket of $t$. 
1. Learn neighbors:
   - Find potential neighbors (constraint-based)
   - Symmetry correction

2. Learn spouses:
   - Find potential spouses (constraint-based)
   - Symmetry correction
SLL algorithm outline

1. Learn neighbors:
   - Find potential neighbors *(score-based)*
   - Symmetry correction

2. Learn spouses:
   - Find potential spouses *(score-based)*
   - Symmetry correction
SLL algorithm outline

1. Learn neighbors:
   - Find potential neighbors (score-based)
   - Symmetry correction

2. Learn spouses:
   - Find potential spouses (score-based)
   - Symmetry correction

Assume: Procedure \texttt{OPTIMALNETWORK}(Z) returns an optimal DAG on subset \(Z\) of nodes.
   - if \(Z\) small \(\Rightarrow\) exact computation possible
   - for example: dynamic programming algorithm
Finding potential neighbors

Algorithm: FINDPOTENTIALNEIGHBORS

5  4  7  1  3
Finding potential neighbors

**Algorithm:** FIND_POTENTIAL_NEIGHBORS
Finding potential neighbors

**Algorithm:** FINDPOTENTIALNEIGHBORS
Finding potential neighbors

**Algorithm:** `FIND_POTENTIAL_NEIGHBORS`

```
5  4  7
```

```
3

1

Z

```

```
t
```
Finding potential neighbors

**Algorithm:** \texttt{FINDPOTENTIALNEIGHBORS}

Call \texttt{OPTIMALNETWORK}(Z).
Finding potential neighbors

Algorithm: FIND POTENTIAL NEIGHBORS
Finding potential neighbors

**Algorithm:** FINDPOTENTIALNEIGHBORS
Finding potential neighbors

Algorithm: FIND_POTENTIAL_NEIGHBORS

5 4

3 7
t
Z
Finding potential neighbors

**Algorithm:** FIND POTENTIAL NEIGHBORS
Finding potential neighbors

Algorithm: FIND>POTENTIAL>NEIGHBORS

The remaining nodes are potential neighbors of target node $t$. 
Problem

True structure:

The algorithm:

\[
\begin{align*}
3 & \quad 2 & \quad 4 \\
Z & \quad t & \quad 2 & \quad 3 & \quad 4
\end{align*}
\]
Problem

True structure:

The algorithm:

\[ \frac{14}{45} \]
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Problem

True structure:

The algorithm:
Symmetry correction for neighbors

**Algorithm:** FINDNEIGHBORS

1. Find potential neighbors $H_t$ of $t$
2. Remove nodes $v \in H_t$ which do not have $t$ as a potential neighbor.
3. The nodes remaining in the end are the neighbors of $t$.

Note: Equivalent to symmetry correction in GLL.
Performance

- Experiments: returns good results in practise.

- Theoretical guarantees?
Asymptotic correctness

Does FINDNEIGHBORS always return the correct set of neighbors under certain assumptions when the sample size tends to infinity?

Compare:

- the generalized local learning (GLL) framework (Aliferis et al., 2010)
- greedy equivalent search (GES) (Chickering, 2002)
Asymptotic correctness

Does \texttt{FINDNEIGHBORS} always return the correct set of neighbors under certain assumptions when the sample size tends to infinity?

Compare:

- the generalized local learning (GLL) framework (Aliferis et al., 2010)
- greedy equivalent search (GES) (Chickering, 2002)

We don’t know for sure, but can say something.
Assumptions

- The data distribution is **faithful** to a Bayesian network. (i.e. there exists a **perfect map** of the data distribution)

- **OPTIMAL NETWORK** uses a **locally consistent** and **score equivalent** scoring criterion (e.g. BDeu)
Network structure properties

Definition

A structure $A$ **contains** a distribution $p$ if $p$ can be described using a Bayesian network with structure $A$ (local Markov condition holds).
Network structure properties

Definition
A structure $A$ **contains** a distribution $p$ if $p$ can be described using a Bayesian network with structure $A$ (local Markov condition holds).

Definition
A distribution $p$ is **faithful** to a structure $A$ if all conditional independencies in $p$ are implied by $A$. 
Network structure properties

Definition

A structure $A$ is a **perfect map** of $p$ if

- $A$ contains $p$ and
- $p$ is faithful to $A$
Network structure properties

Definition

A structure $A$ is a **perfect map** of $p$ if

- $A$ contains $p$ and
- $p$ is faithful to $A$

$\Rightarrow$ no missing or extra independencies

Note: The property of having a perfect map is not closed under marginalization.
Equivalent structures

Definition

Two structures are **Markov equivalent** if they contain the same set of distributions.

A scoring criterion is **score equivalent** if equivalent DAGs always have the same score. (e.g. BDeu)
Equivalent structures

Definition

Two structures are **Markov equivalent** if they contain the same set of distributions.

A scoring criterion is **score equivalent** if equivalent DAGs always have the same score. (e.g. BDeu)

⇒ learning possible up to an equivalence class
Consistent scoring criterion

Definition (Chickering, 2002)

Let $A'$ be the DAG that results from adding the arc $uv$ to $A$. A scoring criterion $f$ is **locally consistent** if in the limit as data size grows, for all $A$, $u$ and $v$ hold:

- If $u \not\perp \! \! \! \perp v | A_v$, then $f(A', D) > f(A, D)$.
- If $u \perp \! \! \! \perp v | A_v$, then $f(A', D) < f(A, D)$.

$\Rightarrow$ relevant arcs increase score, irrelevant decrease
Finding neighbors: theory

In the limit of large sample size, we get the following result:

**Lemma**

\[ \text{FIND\_POTENTIAL\_NEIGHBORS returns a set that includes all true neighbors of target node } t. \]

**Conjecture**

\[ \text{FIND\_NEIGHBORS returns exactly the true neighbors of target node } t. \]
Finding neighbors: Proof of the lemma

In the limit of large sample size:

**Lemma**

$\text{FIND\ POTENTIAL\ NEIGHBORS}$ returns a set that includes all true neighbors of target node $t$.

**Proof:**

- Assume $v$ a neighbor of $t$ in $A$
- Faithfulness $\Rightarrow t$ and $v$ not independent given any subset of nodes
- Local consistency $\Rightarrow \text{OPTIMAL\ NETWORK}$ on $Z$ including $t$ and $v$ always returns a structure with arc between $t$ and $v$
- Thus $v$ kept in $Z$ once added
SLL algorithm outline

1. Learn neighbors:
   - Find potential neighbors (score-based)
   - Symmetry correction

2. Learn spouses:
   - Find potential spouses (score-based)
   - Symmetry correction
Finding potential spouses

Algorithm: FIND_POTENTIAL_SPOUSES

6  2  5  4  1

3  7
t
8

Z
Finding potential spouses

**Algorithm:** FIND_POTENTIAL_SPOUSES

6  2  5  4

\[
Z = \begin{pmatrix}
3 & t & 8 \\
1 & & \\
7 & &
\end{pmatrix}
\]
Finding potential spouses

**Algorithm:** \textsc{FindPotentialSpouses}

\begin{center}
\begin{tikzpicture}
\node[draw,circle,fill=blue!20] at (0,0) (1) {1};
\node[draw,circle,fill=blue!20] at (1,1) (2) {2};
\node[draw,circle,fill=blue!20] at (1,-1) (3) {3};
\node[draw,circle,fill=blue!20] at (2,0) (4) {4};
\node[draw,circle,fill=blue!20] at (-1,1) (5) {5};
\node[draw,circle,fill=blue!20] at (-1,-1) (6) {6};
\node[draw,circle,fill=green!20] at (0,1.5) (t) {\textit{t}};
\node[draw,circle,fill=green!20] at (-0.5,2) (7) {7};
\node[draw,circle,fill=green!20] at (0.5,2) (8) {8};
\node[draw,circle,fill=green!20] at (0,2) (Z) {Z};
\draw (1) -- (2) -- (3) -- (4) -- (5) -- (6);
\end{tikzpicture}
\end{center}

Call \textsc{OptimalNetwork}(Z).
Finding potential spouses

**Algorithm:** FINDPOTENTIALSPOUSES
Finding potential spouses

Algorithm: FINDPOTENTIALSPOUSES
Finding potential spouses

**Algorithm:** FIND_POTENTIAL_SPOUSES
Finding potential spouses

**Algorithm:** FIND_POTENTIAL_SPOUSES

The remaining new nodes are potential spouses of target node $t$. 
Symmetry correction for spouses

**Algorithm:** FINDSPOUSES

1. Find potential spouses $S_t$ of $t$.
2. Add nodes which do have $t$ as a potential spouses.
3. The nodes gathered in the end are the spouses of $t$.

Note: Equivalent to symmetry correction in GLL.
Finding spouses: theory

Again, in the limit of large sample size:

**Lemma**

\[
\text{FINDPOTENTIALSPOUSES} \text{ returns a set that contains only true spouses of target node } t.
\]

**Conjecture**

\[
\text{FINDSPOUSES} \text{ returns exactly the true spouses of target node } t.
\]
Summarising previous lemmas and conjectures:

**Conjecture**

*If the data is faithful to a Bayesian network and one uses a locally consistent and score equivalent scoring criterion then in the limit of large sample size SLL always returns the correct Markov blanket.*
**Time and space requirements**

**OPTIMAL NETWORK**($Z$) dominates:
- $O(|Z|^2 2^{|Z|})$ time
- $O(|Z| 2^{|Z|})$ space

Worst case: $|Z| = O(n)$  
⇒ total time at most $O(n^4 2^n)$

**In practise:** Networks are relatively sparse.  
⇒ significantly lower running times
SLL implementation

A C++ implementation.

**OPTIMAL NETWORK**($Z$) procedure:
- dynamic programming algorithm
- BDeu score
- fallback to GES for $|Z| > 20$

The implementation available at
Experiments

- Generated data from different Bayesian networks. \((\text{ALARM5} = 185 \text{ nodes}, \text{CHILD10} = 200 \text{ nodes})\)

- Compared score-based SLL to constraint-based HITON by Aliferis et al. (2003, 2010).

- Measured SLHD: Sum of Local Hamming Distances between the returned neighbors (Markov blankets) and the true neighbors (Markov blankets).
Experimental results: neighbors

Average SLHD for neighbors.
Experimental results: Markov blankets

Average SLHD for Markov blankets.
Experimental results: time consumption

Average running times (s).

![Graph showing average running times for Alarm5 and Child10 with data points for SLL and HITON.]
Outline

Introduction

Local learning

From local to global

Summary
Local to global learning

**Idea:** Construct the global network structure based on local results.

Implemented two algorithms:
- Constraint-based: SLL+C
- Score-based: SLL+G

Compared to:
- max-min hill-climbing (MMHC) by Tsamardinos et al. (2006)
- greedy search (GS)
- greedy equivalence search (GES) by Chickering (2002)
**Algorithm: SLL+C**

(constraint-based local-to-global)

1. Run SLL for all nodes.

2. Based on the neighbor sets construct the skeleton of the DAG.

3. Direct the edges consistently (if possible) with the v-structures learned during the spouse-phase of SLL. (see Pearl, 2000)
SLL+G algorithm

**Algorithm: SLL+G**
(greedy search based local-to-global)

1. Run the first phase of the local neighbor search of SLL for all nodes.
2. Construct the DAG by greedy search but allow adding arc between two nodes only if both are each others potential neighbors.

⇒ Similar to MMHC algorithm.
Experimental results: SHD

Average SHD.
Experimental results: scores

Average normalized score.
(1 = true network, lower = better)
Experimental results: time consumption

Average running times (s).
Outline

Introduction

Local learning

From local to global

Summary
Summary

Score-based local learning algorithm: SLL
- Conjecture: same theoretical guarantees as GES and constraint-based local learning
- Experiments: competitive alternative to constraint-based local learning

Score-based local-to-global algorithms
- Experiments: mixed results

Future directions
- Prove the guarantees (or lack thereof)
- Speed improvements
References


