LEARNING CHORDAL MARKOV NETWORKS BY DYNAMIC PROGRAMMING
Kustaa Kangas, Teppo Niinimäki, Mikko Koivisto
Department of Computer Science

Motivation: Structure learning in Markov networks asks for an undirected graph that maximizes a given decomposable scoring function. A special interest is in learning graphs that are chordal, since chordal Markov networks (CMNs) of low width admit efficient inference.

Our contribution: We present a dynamic programming algorithm that finds optimal CMNs on $n$ variables in $O(4^n)$ time. Experiments demonstrate our implementation is competitive with recent algorithms based on constraint satisfaction [1] and linear programming [2].

INTRODUCTION

CHORDAL MARKOV NETWORKS

A Markov network:
- Undirected graph $G = (V, E)$ on $V = \{1, \ldots, n\}$
- Represents a joint distribution $p(x_1, \ldots, x_n) = \prod_{C \subseteq V} \psi_C(x_C)$, where $C$ is the set of (maximal) cliques of $G$ and $\psi_C$ are mappings to positive reals.

A chordal Markov network:
- Every cycle of length $\geq 4$ has an edge between two non-consecutive vertices.
- Admits a clique tree decomposition.

STRUCTURE LEARNING PROBLEM

Given data $D$, i.e., samples on $x_1, \ldots, x_n$, a scoring criterion $s_D(G)$ measures how well a chordal graph $G$ fits $D$.

Common scores (e.g. maximum likelihood, Bayesian Dirichlet) decompose as $s_D(G) = \prod_{C \in V} s_D(C)$, where $C$ is the (multi)set of separators. Separator: intersection of adjacent cliques in a clique tree decomposition of $G$.

RECURRENCE

Chordal graphs admit a recursive characterization of the problem.

For $S \subseteq V$ and $\emptyset \subset R \subset V \setminus S$, let $f(S, R)$ be the maximum $s_D(G)$ over chordal $G$ on $S \cup R$ s.t. $S$ is a proper subset of a clique.

Then, the solution is given by $f(\emptyset, V)$ and

$$f(S, R) = \max_{S \subset C \subset S \cup R} s_D(C) g(C, R \setminus C)$$

Dynamic programming runs in $O(4^n)$ time and $O(3^n)$ space on simplified recurrences:

1. Given sets $S$ and $R$, choose a clique $C$ within $R$ that contains $S$ completely:

$$f(S, R) = \max_{S \subset C \subset S \cup R} s_D(C) g(C, R \setminus C)$$

2. Partition the remaining vertices into $R_1, \ldots, R_k$:

$$g(C, U) = \max_{\emptyset \subset R \subset U} h(C, R) g(C, U \setminus R)$$

3. For each part $R$ choose a separator $S$ contained in $C$. Recurse back to step 1:

$$h(C, R) = \max_{S \subset C} f(S, R) / s_D(S)$$

EXPERIMENTS

Compared to a recent constraint satisfaction based algorithm [1], our C++ implementation, Junctor (**), appears to be faster by several orders of magnitude.

We also compared against the freely available GOBNILP (**) on several instances.

SYNTHETIC INSTANCES

The running times (median for GOBNILP) as a function of $n$, on sparse (top) and dense (bottom) instances with 100 (“small”), 1000 (“medium”), and 10,000 (“large”) data samples, bound clique size by $w$. The top red line indicates timeout or memout.

BENCHMARK INSTANCES

The running times on the following benchmark instances from the UCI repository.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Abbr.</th>
<th>$n$</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tic-tac-toe</td>
<td>X</td>
<td>10</td>
<td>958</td>
</tr>
<tr>
<td>Fisher</td>
<td>F</td>
<td>11</td>
<td>1000</td>
</tr>
<tr>
<td>Bridges</td>
<td>B</td>
<td>12</td>
<td>108</td>
</tr>
<tr>
<td>Flare</td>
<td>F</td>
<td>13</td>
<td>1066</td>
</tr>
<tr>
<td>Zoo</td>
<td>Z</td>
<td>17</td>
<td>101</td>
</tr>
<tr>
<td>Voting</td>
<td>V</td>
<td>17</td>
<td>435</td>
</tr>
<tr>
<td>Tumor</td>
<td>T</td>
<td>18</td>
<td>339</td>
</tr>
<tr>
<td>Lymph</td>
<td>L</td>
<td>19</td>
<td>148</td>
</tr>
</tbody>
</table>

Junctor can solve instances of up to 22 variables within a few days for $w = 4$.

(*) Junctor is publicly available to download at www.cs.helsinki.fi/u/jekangas/junctor/.

(**) GOBNILP by Bartlett and Cussens [2] uses integer linear programming for learning optimal Bayesian networks, but can also be restricted to learning chordal Markov networks.
