University at Buffalo The State University of New York

SCoRe group

High Performance Algorithms for Learning Exact Bayesian Networks

Subhadeep **KARAN**, Jaroslaw **ZOLA**
{skaran,jzola}@buffalo.edu

SCoRe – Scalable Computing Research Group
Department of Computer Science and Engineering
University at Buffalo

Abstract

Bayesian Networks (BN) are probabilistic graphical models often used in biomedical applications. In these applications, BN structure is a critical component learned from data (e.g. medical records, gene markers, environmental factors). The problem of BN exact structure learning is to find a network structure that is optimal under certain scoring criteria. This problem has been shown to be NP-hard and the existing solutions are computationally and memory intensive.
We investigate new sequential and parallel BN exact learning strategies that leverage relationships between a partial network structure and the remaining variables to constraint the number of ways in which the partial network can be optimally extended. Via experimental results, we show that our current method provides up to three times improvement in runtime, and orders of magnitude reduction in memory consumption compared to the best known algorithms.

Why Bayesian Networks

- Clear and intuitive interpretation
- Explicit support to handle uncertainty in data
- Ability to answer complex and speculative queries
- Ability to incorporate expert knowledge

Challenges

- Irrespective of the scoring function, the search space of possible BN structures is superexponential
- Existing approaches based on dynamic programming and memoization become infeasible even for relatively small input data

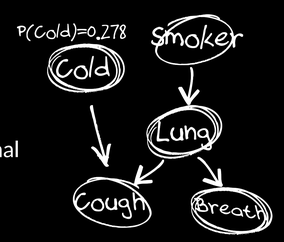
References

1. S. Karan, J. Zola, Exact Structure Learning of Bayesian Networks by Optimal Path Extension, In Proc. of IEEE BigData, 2016.

Bayesian Network

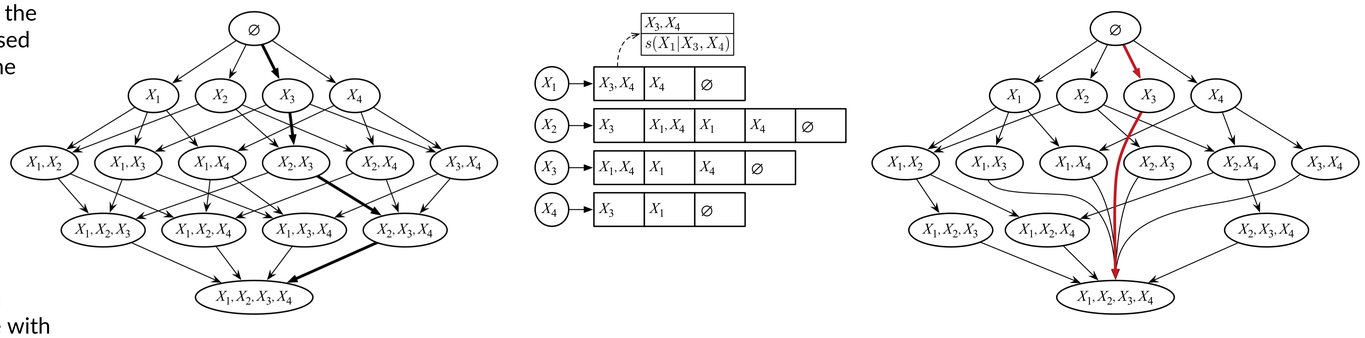
Pair (G,P) where G is a DAG over a set of random variables, and P is the corresponding probability distribution. G encodes efficient factorization of the joint probability P.

BN provides compact and easy to interpret representation of conditional independencies.



Proposed Approach

- Given n random variables and their observations, the search for optimal BN structure can be decomposed into two subproblems: i) finding and evaluating the minimal set of all possible parents for each variable, ii) finding an optimal variables ordering by exploring dynamic programming lattice.
- The first subproblem requires efficient indexing strategies to quickly estimate conditional probabilities from the input data.
- The second subproblem is equivalent to finding a shortest path in the dynamic programming lattice with 2^n nodes: node U represents partial network, and edge $(U, U \cup \{X_i\})$ corresponds to extending partial network with a new variable. The cost of adding a new variable is prescribed by the scoring function d.



Example dynamic programming lattice for problem with four variables and the resulting compacted lattice obtained via OPE.

A-star With Optimal Path Extension

```
1:  s.g ← 0
2:  s.h ← 0
3:  for X_i ∈ 𝒳 do
4:      s.h ← s.h + d(X_i, 𝒳 − {X_i})
5:  s.f ← s.h
6:  s.set ← φ
7:  s.p ← φ
8:  Q.push(s)
9:  while Q ≠ φ do
10:     v ← Q.pop()
11:     C.push(v)
12:     if v.set = 𝒳 then
13:         return Backtrack(v, C)
14:     for X_i ∈ 𝒳 − v.set do
15:         u.g ← v.g + d(X_i, v.set)
16:         u.h ← v.h − d(X_i, 𝒳 − {X_i})
17:         u.f ← u.g + u.h
18:         u.set ← u.set ∪ {X_i}
19:         u.p ← v.set
20:         u ← PathExtension(u)
21:         if u ∉ C then
22:             if u ∈ Q then
23:                 pu ← Q.handle(u)
24:                 if pu.f > u.f then
25:                     pu ← u
26:                     Q.update(pu)
27:             else
28:                 Q.push(u)
```

PathExtension

```
1:  repeat
2:      extended ← false
3:      for X_i ∈ 𝒳 − u.set do
4:          if d(X_i, u.set) = d(X_i, 𝒳 − {X_i}) then
5:              u.g ← u.g + d(X_i, 𝒳 − {X_i})
6:              u.h ← u.h − d(X_i, 𝒳 − {X_i})
7:              u.set ← u.set ∪ {X_i}
8:              extended ← true
9:  until not extended
10: return u
```

Optimal Path Extension (OPE)

- We propose a novel strategy to constraint the number of ways in which partial network can be optimally extended, thus compacting the search lattice.
- Node U at level k in the dynamic programming lattice can be extended in $(n - k)$ ways. However, in many cases we can immediately identify the only extensions that can lead to an optimal path.
- Theorem 1 [1]: Let U be a superset of the optimal parents set for $X_i \in \mathcal{X} - \{X_i\}$. In the optimal path from U to the sink of the dynamic programming lattice U must be followed by $U \cup \{X_i\}$.
- Our solution imposes negligible worst case $O(n - k)$ overhead to check if OPE can be applied.
- The technique can be seamlessly combined with different shortest path solvers (BFS, IDA, etc.), and is very easy to implement.

SABNA Toolkit

- We are developing SABNA – Scalable Accelerated Bayesian Network Analytics Toolkit available from: https://gitlab.com/SCoRe-Group/SABNA-Release
- The toolkit provides our optimized search strategies and supports different scoring functions for Bayesian Networks learning.
- Using standard benchmarks we compared performance of SABNA to the currently best tools for exact structure learning.

Dataset	n	BFS	URLearning	**SABNA**
Mushroom	23	2m12s	1m29s	1m
Autos	26	3m54s	37s	13s
Insurance	27	8m14s	7m25s	2m28s
Water	32	M	M	2m8s
Soybean	36	M	M	1h36m
Alarm	37	M	T	1h3m
Bands	39	M	T	1h10m

- Currently, we are expanding SABNA with parallel algorithms to leverage massively parallel and heterogeneous multicore processors to further accelerate exact structure learning.

Acknowledgment

Authors wish to acknowledge support provided by the Center for Computational Research at the University at Buffalo.