Globally Optimal Structure Learning of Bayesian Networks from Data

Kobra Etminani[1], Mahmoud Naghibzadeh[1], and Amir Reza Razavi[2]

[1] Dept. of Computer Engineering, Ferdowsi University of Mashhad, Iran
[2] Dept. of Medical Informatics, Medical University of Mashhad, Iran
etminani@wali.um.ac.ir, naghibzadeh@um.ac.ir,
razaviar@mums.ac.ir

Abstract. The problem of finding a Bayesian network structure which maximizes a score function is known as Bayesian network structure learning from data. We study this problem in this paper with respect to a decomposable score function. Solving this problem is known to be NP-hard. Several algorithms are proposed to overcome this problem such as hill-climbing, dynamic programming, branch and bound, and so on. We propose a new branch and bound algorithm that tries to find the globally optimal network structure with respect to the score function. It is an any-time algorithm, i.e., if stopped, it gives the best solution found. Some pruning strategies are applied to the proposed algorithm and drastically reduce the search space. The performance of the proposed algorithm is compared with the latest algorithm which showed better performance to the others, within several data sets. We showed that the new algorithm outperforms the previously best one.

Keywords: Bayesian networks, structure learning, discrete variables.

1 Introduction

A Bayesian network or Belief Network (BN) is a directed acyclic graph (DAG) where nodes stand for random variables and edges stand for conditional dependencies. The random variables can be discrete or continuous. In this paper, learning Bayesian network structure for discrete variables is studied.

Bayesian network structure learning from data has attracted a great deal of research in recent years. Finding the best structure for Bayesian network is known to be NP-Hard [1, 2]. Consequently, much of the research has focused on methods that find suboptimal solutions. Generally, there are several approaches to learn a structure. Some methods are based on scoring functions that depend on the data and some approaches are based on statistical similarities among variables. We focus on those methods that are based on scoring functions and try to find the optimal solution according to this function.

Buntine in [3] proposes a hill-climbing method that performs a stochastic local search. However, this algorithm may get stuck in a local maximum. Although the approach is simple and applicable to small networks, since it is an exhaustive method

K. Diamantaras, W. Duch, L.S. Iliadis (Eds.): ICANN 2010, Part I, LNCS 6352, pp. 101–106, 2010.

it cannot be applied to networks with large number of variables due to large number of possible edge modifications. Most exact methods that guarantee to find the optimal structure according to a scoring function, are based on dynamic programming [4, 5, 6], and branch and bound [7, 8] techniques. The time and/or space complexities of the methods that use dynamic programming approach forbid the application of those approaches to networks with large number of variables ($n > 30$). Campos et al. in [7] propose a branch and bound algorithm that extremely reduces the search space of possible structures, while guaranteeing to obtain the optimal solution. The procedure runs at most $\prod_i |C(i)|$ steps, where $C(i)$ is the size of the cache for variable i, which is the needed space to store the required local scores for variable i. Memory requirement for storing this cache is $\sum_i C(i)$ (in the worst case it is $O(n2^n)$). It is an anytime algorithm and gives the current best solution whenever stopped.

We present a new branch and bound (B&B) algorithm that guarantees to find the global optimal Bayesian network structure in less time and memory requirements comparing to the best exact previous methods. It uses a decomposable scoring function and drastically reduces the search space for possible structures.

The rest of the paper is organized as follows: In Section 2, the Bayesian networks and scoring functions (i.e. problem description) are introduced. The proposed algorithm is explained in Section 3. The performance of the new method through experiments is showed in Section 4 and the paper ends with new ideas for future work in Section 5.

2 Problem Description

A BN is a DAG, composed of n random variables, that represents joint probability densities over these variables. Usually, a BN can be defined as a triple ($\mathcal{G}$, $\mathcal{V}$, $\mathcal{P}$), where $\mathcal{G}$ denotes the DAG, $\mathcal{V}$ denotes the set of random variables $\{V_1,...,V_n\}$ (nodes in $\mathcal{G}$) and $\mathcal{P}$ is a set of conditional probability densities $p(V_i \mid \pi_i)$ where π_i denotes the parents of V_i in the graph. Each discrete variable V_i has a finite number of values, r_i and the number of configurations of the parent set, q_i, that is $q_i = \prod_{V_i \in \pi_i} r_i$.

The goal of Bayesian network structure learning is to find the globally optimal structure from data. We assume data is complete with no missing values and consisted of discrete variables. If not, the data is cleaned by removing rows with missing values and discretizing the continuous variables. Given the complete discrete data D with N instances, we want to find a network structure that maximizes a scoring function:

$$\max_G \ score(G) . \tag{1}$$

In this study, we assume a scoring function that is decomposable, i.e.:

$$score(G) = \sum_i localScore(V_i \mid \pi_i) . \tag{2}$$

where $localScore(V_i \mid \pi_i)$ is the local score for node V_i given its parents π_i. Many common scores such as BDeu [3], BIC [8], BD [9], BDe [10] and AIC [11] are decomposable.

3 The Proposed Branch and Bound Algorithm

In this section, the proposed branch & bound algorithm used to find global optimum structure of the Bayesian networks and its complexity is explained.

We define each node of the branch & bound tree, as follows:

Definition 1. A tree node *tn* of the branch & bound tree is defined by *tn(NS, s)*, where *NS* is the network structure that this node stores and *s* is the score of this network structure.

The main idea of our proposed method is:

1. calculate the needed local scores of each node considering Lemmas in [7]
2. create the root of the branch & bound tree by considering the best local score for each node in the network structure
3. if the network structure store in the node is a DAG then algorithm is finished and global best network structure is found, otherwise let $d = 1$
4. create the children of this node by replacing the parent set of variable V_d (node in the network) with its 1th, 2nd, … best parent configurations (the parent sets that provide 1th, 2nd, … best local score for V_d)
5. while creating a child, if a DAG is found, update the best network found. In addition, check the new child and create the other children if needed
6. choose the next leaf node (if existed) and assign *d* with the depth of this node
7. if the next node is null the algorithm is finished, otherwise go to step 3.

The proved Lemmas in [7] are used here in the first step to reduce the cache size. Therefore, the size of cache memory (required to keep local scores for the next step) is equal to [7]. In the proposed algorithm, it is supposed to replace the parent set of each node V_d with its all possible parent configurations (which are sequentially chosen based on the computed local scores sorted in a descending order) in the *d*th depth of the tree with a pre-set order. We consider a fixed ordering of variables when the algorithm begins and we keep it to the end of the algorithm. Therefore, node V_i always denotes the *i*th variable in the ordering. However, we try not to create all the tree nodes. We prune this tree using the criteria explained later according to Bayesian networks properties. The pruning, in practice, extremely decreases the number of nodes to be created to find the global best network structure, in comparison with [7].

The bounding rules considered are as follows:

1. If *NS* is a DAG, the best net score found for the DAG until that point is updated (if NS leads to a better score). Here, there is no need to add further nodes to the tree, otherwise:
2. If the new node obtains worse score than the best score found for a DAG until that point, then this node cannot lead to a better result. We do not add this node to the tree.
3. Suppose we are at *d*th depth of the tree (i.e. we want to replace the parent set of V_d with its 1th, 2nd, … best parent configuration). If there is a cycle that contains only $\{V_1,...,V_{d-1}\}$, this cycle cannot be removed. The reason is that

the parent sets of $V_d, V_{d+1}, ..., V_n$ are replaced with their other best parent sets from depth d to the maximum depth (depth n). However, the cycle contains the nodes ($\{V_1, ..., V_{d-1}\}$) that never change from depth d to the maximum depth n and no DAG can be found from this new node in the tree.

4. If the new node is in the depth n, there is no need to add it to the tree because it cannot lead to a result since the network nodes are finished and it is not a DAG.

The new child (if one of the above criteria is satisfied we put null in new child) is added to the parent node.

Fig. 1 shows the new branch & bound algorithm. The input to this algorithm is the local scores that are computed and stored in a file, previously and the output is the global best network structure. After creating all possible children nodes for *parentNode*, the next node to be expanded is selected. We select a leaf with the maximum score to be the *parentNode* next time.

Algorithm 1. *EtminaniB&B(localScores)*

```
rootNode(NS_r, s_r) = create a node by considering the best parent config for each node
parentNode = rootNode
d = 1      // d shows depth of tree
while (parentNode) do
/* expand nodes for each kth maximum local score for node V_d (if needed)*/
   for each k ∈ | localScore(V_d) |
      childNode = CreateNode(k)
      if (NotNeeded(childNode))
         exit for
   end for
   parentNode = MaxPossibleLeafNode()  //continue from the next maximum leaf(if existed)
end while
return best network found
```

Fig. 1. The proposed branch & bound algorithm

The required memory for the proposed algorithm for storing the needed local scores is the same as [7] and it is data dependent. In the worst case, all tree nodes should be created and none of the criteria satisfies to prune the tree and reduces the search space (this case practically never happens but we consider it as the worst case), so the maximum number of nodes is $\sum_{i=1}^{n} \prod_{j=1}^{i} | C(j) |$.

4 Experiments

In this study, datasets available at the UCI repository were used [12], to demonstrate the capability of our method. Some of these datasets contain missing data which were removed. The properties of the datasets, the size of the generated cache (the reduced cached based on proven Lemmas in [7]) and the size of the cache if all local scores were computed for the BIC scoring function are presented in Table 1.

Table 2 presents the results of two distinct algorithms: the proposed branch & bound algorithm described in Section 4 (EtminaniB&B), and the branch & bound algorithm presented in [7] (CassioB&B)[1] . Number of nodes in the branch & bound tree are shown to facilitate a better comparison between the two algorithms. The time column represents only the time of running the algorithms. The time required to compute the needed local scores is not included, because it is similar for both algorithms.

Table 1. Dataset properties and cache size for B&B algorithm and total cache size without reduction

name	n	N	Reduced Cache size	Total cache size ($n2^n$)
Abalone	9	4177	65	$2^{11.2}$
Tic-tac-toe	10	958	28	$2^{13.3}$
Bc-wisconsin	11	669	97	$2^{14.5}$
Wine	14	178	98	$2^{17.8}$
Heart-hungerian	14	294	8246	$2^{17.8}$
Heart-cleveland	14	303	45	$2^{17.8}$
Mushroom	22	5644	260	$2^{26.4}$
Spect-heart data	23	80	275	$2^{26.5}$
King rock	36	3196	524	$2^{41.2}$

Table 2. Comparison of BIC scores, number of created tree nodes and running time among EtminaniB&B and CassioB&B

Name	EtminaniB&B			CassioB&B		
	Score	No. tree nodes	Time (s)	Score	No. tree nodes	Time (s)
Abalone	-16571.7	76	< 1	-16571.7	1205	< 1
Tic-tac-toe	-9656.2	33	< 1	-9656.2	65	< 1
Bc-wisconsin	-3359.6	43	< 1	-3359.6	3909	9
Wine	-1889.8	155	< 1	-1889.8	259	1.2
Heart-hungerian	-2368.0	18	< 1	-2368.0	170	< 1
Heart-cleveland	-3410.8	21	< 1	-3410.8	18	< 1
Mushroom	-691.0	1	< 1	-691.0	1	< 1
Spect-heart data	-761.0	343	4.5	-761.0	288	2.8
King rock	-290.5	1	< 1	-290.5	1	< 1

In mushroom and king rock datasets, both of them create only one node. The reason is that the root of the B&B tree was already a DAG in this data set (best parent set for each node leads to DAG) and no additional node is created.

Our method (EtminaniB&B) most of the time creates smaller number of nodes in the branch & bound tree and leads to the global optimal result in less running time than the previous B&B (CassioB&B).

The new algorithm can result into the same result in almost less running time and smaller number of nodes in the B&B tree. In [7], Cassio et al. show the superiority of

[1] We have run the dynamic programming idea of [4], (which is worse than the two branch & bound algorithm in time and space). We omit it because of lack of space.

their algorithm compared to the previous methods [4, 5]. Now, we show the advantages and the performance of our new method to [7].

5 Conclusion and Future Work

In this study, learning the Bayesian network structure from data for discrete variables is studied. A new branch and bound algorithm is presented that guarantees global optimality with respect to a decomposable scoring function. It is an any-time method, i.e. if stopped it provides the best solution found so far. We made use of two previously proven lemmas to reduce the search space and the required memory.

Several common datasets are used to demonstrate the benefits of the proposed method. The experiments show that the proposed approach provides better results, i.e. resulting in smaller number of nodes and less running time, most of the cases.

Several aspects remain for the future work such as applying other criteria to forbid unnecessary computing for local scores and creating unneeded B&B tree nodes which can reduce the search space and the required memory.

References

1. Chickering, D.M., Meek, C., Heckerman, D.: Large-sample learning of Bayesian networks is NP-Hard. In: Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence, pp. 124–133. Morgan Kaufmann Publishers, San Francisco (2003)
2. Chickering, D.M.: Learning Bayesian networks is NP-complete. Learning from Data: Artificial Intelligence and Statistics V, 121–130 (1996)
3. Buntine, W.: Theory refinement on Bayesian Networks. In: Proceedings of the Seventh Conference on Uncertainty in Artificial intelligence, Los Angeles, pp. 52–60 (1991)
4. Silander, T., Myllymaki, P.: A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. In: Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence, pp. 445–452 (2006)
5. Singh, A.P., Moore, A.W.: Finding Optimal Bayesian Networks by Dynamic Programming. Technical Report, Carnegie Mellon University CALD-05-106 (2005)
6. Koivisto, M., Sood, K., Chickering, D.M.: Exact Bayesian structure discovery in Bayesian networks. Journal of Machine Learning Research 5, 549–573 (2004)
7. Campos, C.P., Zeng, Z., Ji, Q.: Structure Learning of Bayesian Networks using Constraints. In: Proceedings of the 26th International Conference on Machine Learning, Canada (2009)
8. Schwartz, G.: Estimating the dimensions of a model. Annals of Statistics 6, 461–464 (1978)
9. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992)
10. Hecherman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20, 197–243 (1995)
11. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19, 716–723 (1974)
12. Asuncion, A., Newman, D.: UCI machine learning repository, `http://archive.ics.uci.edu/ml/datasets.html`