Initialization Heuristics for Greedy Bayesian Network Structure Learning

Walter Perez Urcia
and
Denis Deratani Mauá

Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil
wperez@ime.usp.br,denis.maua@usp.br

Abstract. A popular and effective method for learning Bayesian network structures is to perform a greedy search on the space of variable orderings followed by an exhaustive search over the restricted space of compatible parent sets. Usually, the greedy search is initialized with a randomly sampled order. In this article we develop heuristics for producing informed initial solutions to order-based search motivated by the Feedback Arc Set Problem.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

Keywords: Bayesian networks, machine learning, local search

1. INTRODUCTION

Bayesian Networks are space-efficient representations of multivariate probability distributions over the atributtes of a data set. They are defined by two components: (i) a directed acyclic graph (DAG), where the nodes are the atributes of the data set and the edges encode the (in)dependence relationships among the atributes; and (ii) a collection of local conditional probability distributions of each atribute given its parents. More formally, a Bayesian network specification contains a DAG $G = (V, E)$, where $V = \{X_1, X_2, \ldots, X_n\}$ is the set of (discrete) attributes, and a collection of conditional probability distributions $P(X_i \mid Pa_G(X_i))$, $i = 1, \ldots, n$, where $Pa_G(X_i)$ is the set of atributes that are parents of X_i in G. This definition shows that the number of numerical parameters (i.e., local conditional probability values) grows exponentially with the number of parents (in-degree) of a node (assuming the values are organized in tables). A Bayesian network induces a joint probability distribution over all the attributes through the equation

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_G(X_i)). \tag{1}$$

Hence, Bayesian networks with sparse DAGs succinctly represent joint probability distributions over many attributes.

A common approach to learning Bayesian networks from a given data set is the search-and-score approach, which consists of associating every graph structure with a polynomial-time computable score value and searching for structures with high score values. The score value of a structure usually rewards structures that assign high probability of observing the data set (i.e., the data likelihood) and penalizes the complexity of the model (i.e., the number of parameters). Some examples are the Bayesian Information Criterion (BIC) [Cover and Thomas 1991], the Minimum Description Length

(MDL) [Lam and Bacchus 1994] and the Bayesian Dirichlet score (BD) [Heckerman et al. 1995].

Score-based Bayesian network structure learning from data is a NP-hard problem [Chickering et al. 2004], even when the in-degree (i.e., maximum number of parents) of the graph is bounded. For this reason, the commonest approach to solve the problem is to use local search methods such as searching over the space of structures, searching over Markov equivalence classes, and searching over the space of topological orders. Local search methods are well-known to be vulnerable to poor local maxima unless a strategy for avoinding low score regions is used. One such strategy is the use of non-greedy heuristics that allow moving for lower value solutions during search in order to escape local maxima. Another stragegy is to use *good initialization heuristics* that attempt to start the search in regions of high local maxima. Most often, a simple uniformed random generation of initial solutions is adopted.

In this article we propose new heuristics for generating good initial solutions to be fed into local search methods for Bayesian network structure learning. We focus on order-based methods, but our technique can be exploited by any local search procedure for learning Bayesian network structures. Also, one of the heuristics is motivated by solutions of the Feedback Arc Set Problem (FASP), which is the problem of transforming a cyclic direct graph into a DAG. Our experiments show that using these new methods improves the quality of order-based local search.

The article is structured as follows: We begin in Section 2 explaining the Greedy Search algorithm. In Section 3, we describe the new algorithms for generating initial solutions. Section 4 shows the experiments using both approaches and comparing them (in scoring and number of iterations needed) with multiple data sets. Finally, in Section 5 we give some conclusions about the new methods.

2. LEARNING BAYESIAN NETWORKS

In this section, we define the search-and-score approach learning Bayesian networks, and review some of its most popular techniques.

2.1 Definition of the problem

Given a data set D and a scoring function $sc(G)$,[1] the Bayesian network structure learning problem is to select a score-maximizing DAG, that is to find

$$G^* = \arg \max_{G : G \text{ is a DAG}} sc(G) \,. \tag{2}$$

Most scoring functions can be rewritten in the form

$$sc(G) = F(G) - \varphi(N) \times P(G), \tag{3}$$

where N is the number of records in the data set D, $F(G)$ is a data fitness function (i.e., how well the model represents the observed data), $\varphi(N)$ is a non-decreasing function of data size and $P(G)$ measures the model complexity of G. One example is the Bayesian information criterion (BIC) defined as

$$BIC(G) = LL(G) - \frac{\log N}{2} size(G), \tag{4}$$

where

$$LL(G) = \sum_{i=1}^{n} \sum_{k} \sum_{j} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \tag{5}$$

[1]The dependence of the scoring function on the data set is usually left implicitly, as for most of this explanation we can assume a fixed data set.

is the data loglikelihood,

$$size(G) = \sum_{i=1}^{n} (|\Omega_i| - 1) \prod_{X_j \in Pa(X_i)} |\Omega_j|, \tag{6}$$

is the "size" of a model with structure G, n is the number of atributes, N_{ijk} the number of instances where attribute X_i takes its kth value, and its parents take the jth configuration (for some arbitrary fixed ordering of the configurations of the parents' values), and similarly for N_{ij}, and Ω_i is the set of possible values for the attribute X_i. Most commonly used scoring functions, BIC included, are ***decomposable***, meaning that they can be written as a sum of local scoring functions: $sc(G) = \sum_i sc(X_i, Pa(X_i))$. Provided the scoring function is decomposable, we can obtain an upper bound on the value of $sc(G^*)$ by computing $sc(\overline{G})$, where

$$\overline{G} = \arg \sum_i \max_{Pa(X_i)} sc(x_i, Pa(X_i)) \,. \tag{7}$$

Another property often satisfied by scoring functions is ***likelihood equivalence***, which asserts that two structures with same loglikelihood also have the same score [Chickering and Meek 2004]. Likelihood equivalence is justified as a desirable property, since two structures that assign the same loglikelihood to data cannot be distinguished by the data alone. The BIC scoring function satisfies likelihood equivalence.

Finding a score-maximizing structure is a NP-hard problem, and practitioners often resort to greedy search algorithms, which we review next.

2.2 Greedy Search Algorithm

The Greedy Search algorithm is a popular heuristic method used to find an approximate solution for the Bayesian network learning. The method relies on the definition of a neighborhood space among solutions, and on local moves that search for an improving solution in the neighborhood of an incumbent solution. Different neighborhoods and local moves give rise to different methods such as Equivalence-based, Structure-based, and Order-based methods. Algorithm 1 shows a general pseudocode for this algorithm.

Algorithm 1: Greedy Search

```
1   $G = Initial\_Solution(X_1, \ldots, X_n, conf)$
2   For a number of iterations $K$
3           $best\_neighbor = find\_best\_neighbor(G)$
4           if $score(best\_neighbor) > score(G)$ then
5                   $G = best\_neighbor$
6   Return $G$
```

The main idea of the algorithm is to generate an initial solution based on the $conf$ parameter (for instance, $conf$ can specify that the solution has to be random generated). After that, for a number of iterations K the algorithm explores the search space and selects the best neighbor of the best solution until that moment. It then updates the lower bound on the score inc case an improving solutions has been found. Additionally, an early stop condition can be added to verify whether the algorithm has reached a local optimum (i.e. if no local move can improve the lower bound). The algorithm terminates by returning the best solution found. Several methods can be obtained by varying the implementation of lines 1, 3 and 4, which specify how to generate an initial solution, what the search space is and how the scoring function used, respectively.

2.2.1 *Structure-based.* One of earliest approaches to learning Bayesian networks was to perform a greedy search over the space of DAGs, with local moves being the operations of adding, removing or reverting an edge, followed by the verification of acyclicity in the case of edge addition. The initial solution is usually obtained by randomly generating a DAG, using one of the many methods available in the literature.

2.2.2 *Equivalence-based.* An alternative approach is to search within the class of score-equivalent DAGs. This can be efficiently achieved when the scoring function is likelihood equivalent by using pDAGs, which are graphs that contain both undirected and directed edges (but no directed cycles). In this case, greedy search operates on the space of pDAGs, and the neighborhood is defined by addition, removal and reversal of edges, just as in structure-based search.

2.2.3 *Order-based.* Given a topological ordering $<$ of the attributes, the problem of learning a Bayesian network simplifies to

$$G^* = \arg \max_{G \text{ consistent with } <} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) = \arg \sum_{i=1}^{n} \max_{P \subseteq \{X_j < X_i\}} sc(X_i, P), \tag{8}$$

This means the if an optimal ordering over the attributes is known, an optimal DAG can be found by maximizing the local scores independently. Since an exhaustive approach would take $O(2^n)$ time, the local optimization is usually constrained to the space of parents of cardinality at most a certain parameter d. For the BIC scoring function (as for others), limiting the cardinality of parent sets does not affect the optimality, as the number of maximal parents in the optimal BIC-maximizing DAG can be shown to be at most $\log N$ (so it suffices to select $d \geq \log N$).

Order-based Greedy Search is a popular and effective solution to the problem of learning the structure of a Bayesian network, which consists of searching the spaces of topological oderings of variables. The method starts with a topological ordering L, and greedily moves to an improving ordering by swapping two adjacent attributes in L if any exists. Algorithm 2 shows a pseudocode for the method. The function *swap* in line 5 swaps the values $L[i]$ and $L[i+1]$ in the order L in order to obtain a neighbor of the current solution.

Algorithm 2: Order-based Greedy Search

```
1    $L = Get\_Order(X_1, \ldots, X_n, conf)$
2    For a number of iterations $K$
3        $current\_sol = L$
4        For each $i = 1$ to $n-1$ do
5            $L_i = swap(L, i, i+1)$
6            if $score(L_i) > score(current\_sol)$
7                $current\_sol = L_i$
8        if $score(current\_sol) > score(L)$ then
9            $L = current\_sol$
10   Return $network(L)$
```

The standard approach to generate initial solutions is to sample a permutation of the attributes uniformly at random by some efficient procedure such as the Fisher-Yates algorithm [Knuth 1998]. In the next section, we propose new strategies to informed generation of topological orderings to be used as initial solutions in Order-Based search.

3. GENERATING INFORMED INITIAL SOLUTIONS

As with most local search approaches, the selection of a good initial solution that avoids poor local maxima is crucial for finding good solutions. Traditionally, this is attempted by randomly generating initial solutions (DAGs, pDAGs or orderings) in order to cover as much as possible of the space. In this section, we devise methods that take advantage of the structure of the problem to produce better initial solutions. Although we focus on Order-Based search, our methods could be used for any of the other greedy approaches discussed.

3.1 DFS-based solution

Generating random solutions means having a huge search space ($n!$, where n is the number of attributes in the dataset). In order to reduce this search space we can use a graph G^* (in Figure 1 that tell us some information about the relationships between the attributes.

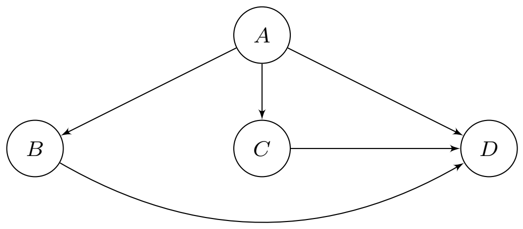

Fig. 1: Graph $\overline{G}$ built using equation 0??

From $\overline{G}$, we know that some orders does not generate some of the parent sets showed there. For instance, we could have attribute C before D, or A before B,C or D in the order generated. Having in consideration all these relations we have the following possible orders:

ABDC	ADBC	BDAC	CDAB	DABC	DBAC	DCAB
ACDB	ADCB	BDCA	CDBA	DACB	DBCA	DCBA

As can be noticed we have 14 possible orders given $\overline{G}$, but $4! = 24$ possible orders (the number of permutations using 4 nodes) using a random approach. So we reduce the search space and also improve the possibilities that getting a good initial order because the relations are the best parent sets for each attribute. Also, the difference could be greater in problems with more attributes. Taking into consideration the previous analysis, we propose the following algorithm to generate initial solutions:

```
(1) Find the best parent set $Pa(X_i)$ for each attribute $X_i$
(2) Build a graph $\overline{G}$ using the relations between each attribute $X_i$ and $Pa(X_i)$ as edges
(3) Start with an empty list $L$
(4) While there are nodes not visited do
    (a) Choose a random node $X_k$ not visited
    (b) Perform a DFS on graph $\overline{G}$ using $X_k$ as root and add each node $X_v$ visited to the order $L$
(5) Return $L$
```

3.2 FAS-based solution

In this subsection, we first explained the generic problem and one of the variants of the Feedback Arc Set (FAS) Problem. After that, a reduction of the learning structure bayesian network problem will be explained in order to propose a new algorithm for generating initial solutions.

3.2.1 *Feedback Arc Set Problem.* The generic problem is stated as: Given a graph $G = (V, E)$, a set $F \subseteq E$ is called the Feedback Arc Set (FAS) if every cycle of G has at least one edge in F. In other words, F is the edge set that if you put if off the graph G, it becomes a directed acyclic graph (DAG).
There are some variants of this problem, but we only focus on finding the minimum cost FAS F from a weighted directed graph G defined in equation 9.

$$F_G = \min_{G-F \text{ is a DAG}} \sum_{(u,v)\in E} W_{uv}, \tag{9}$$

where W_{uv} is the weight of the edge from u to v on G^*. Although this variant becomes NP-hard, there are some approximation algorithms like the one that is below.

(1) Start with $F = \emptyset$
(2) While there is a cycle C in G do
    (a) Find the lowest weight W_{min} of all the edges $(u, v) \in C$
    (b) For each edge $(u, v) \in C$, do $W_{uv} = W_{uv} - W_{min}$
    (c) If some $W_{uv} = 0$ for some edge $(u, v) \in C$, then add (u, v) to F
(3) For each edge $(u, v) \in F$, add it to G only if it does not build a cycle on the graph
(4) Return F

3.2.2 *Algorithm for initial solutions.* First, we have the following relation for the problem:

$$G^* = \arg \max_{G \text{ is acyclic}} \sum_{i=1}^{n} sc(X_i, Pa(X_i)) \leq sc(\overline{G}) \tag{10}$$

The left part is the original definition of the problem for learning structure of bayesian networks and the right part is the upper bound graph $\overline{G}$ relaxing the acyclicity constraint. In order to prove that the relation is true, we need to define the weight edges as follows:

$$W_{ji} = sc(X_i, P) - sc(X_i, P \setminus \{X_j\}), \tag{11}$$

where W_{ji} is the weight of the edge that goes from X_j to X_i and P is the best parents set for X_i. This formula represents the cost of getting X_j out of the set P and it is always a positive number. When the value is so small, it means the parent X_j could be more important. On the contrary if the weight is so big, this means that X_j is not so important in P.
Then we can prove that the problem can be reduce with the following equations:

$$\min \sum_{i=0}^{n} W_{uv} = \min \sum_{i=0}^{n} (sc(X_i, P) - sc(X_i, P \setminus X_j)) \tag{12}$$

$$\min \sum_{i=0}^{n} sc(X_i, P) - \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \tag{13}$$

$$\max \sum_{i=0}^{n} sc(X_i, P) + \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \tag{14}$$

$$sc(G) + \min \sum_{i=0}^{n} sc(X_i, P \setminus X_j) \approx sc(G^*) \tag{15}$$

This means that the score of the DAG G plus the score of the deleted arcs from $\overline{G}$ give an approximation for G^*, and also that G^* could be a subgraph of $\overline{G}$. We can now describe our second heuristic for generating initial solutions, based on the minimum cost FAS problem:

(1) Find the best parent set $Pa(X_i)$ for each attribute X_i
(2) Build the graph $\overline{G}$ using all the relationships $X_j \implies X_i, X_j \in Pa(X_i)$ and put its score as W_{ji}
(3) Find the minimum cost FAS F
(4) Delete all edges $(u, v) \in F$ from $\overline{G}$ to obtain a network G
(5) Return a topological order of G

These new methods for generating initial solutions will be used in next section to learn Bayesian networks with multiple data sets.

4. EXPERIMENTS AND RESULTS

In order to evaluate the quality of our approaches, we learned Bayesian networks using Order-based greedy search and different initialization strategies from several data sets commonly used for benchmarking. The names and relevant characteristics of the data sets used are shown in Table I. The

Dataset	n (#attributes)	N (#instances)
Census	15	30168
Letter	17	20000
Image	20	2310
Mushroom	23	8124
Sensors	25	5456
SteelPlates	28	1941
Epigenetics	30	72228

Table I: Data sets characteristics

greedy search was ran with a maximum number of parents ($d = 3$), a limit of 100 iterations ($K = 100$) and 100 restarts ($S = 100$). We used the BIC scoring in all experiments. The choice of the parameter S was adopted so as to have enough data to produce statistically significant results while still being able to ran all the experiments in a restricted amount of time.

We evaluated three different initialization strategies: Random, which randomly generates a topological ordering, DFS-based, which uses the DFS heuristic, and FAS-based, which uses the heuristic based on the minimum cost FAS. For each approach, we compared the score of the best network found, the percentage of solutions that converged to the best score and the average number of iterations that local search took to converge. The results are shown in Table II.

5. CONCLUSIONS AND FUTURE WORK

Learning Bayesian networks from data is notably difficult problem, and practitioners often resort to approximate solutions such as greedy search. The quality of the solutions produced by greedy approaches strongly depends on the initial solution. In this work, we proposed new heuristics for producing good initial solutions to be fed into greedy Bayesian network structure search methods.

Experiments with benchmark data sets showed that our heuristics occasionally lead to better solutions, and that in most cases it leads to faster convergence with a small overhead (compared to the commonly used methods of generating initial solutions). The advantage of our heuristics grows with the size of the data set.

Our heuristics can also be exploited by branch-and-bound solvers that find optimal solutions. This is left for future work.

Dataset	Approach	Best Score	% Solutions	Avg. It.
Census	Random	-223893.767	**9.00**	6.03 ± 2.55
	DFS-based	-223893.767	2.00	4.87 ± 2.67
	FAS-based	-223893.767	3.00	**3.07 ± 2.09**
Letter	Random	-162332.183	5.00	4.30 ± 1.57
	DFS-based	-162332.183	11.00	4.38 ± 1.82
	FAS-based	-162332.183	**51.00**	**2.25 ± 0.83**
Image	Random	-23749.089	3.00	7.02 ± 2.74
	DFS-based	-23749.089	6.00	6.96 ± 3.16
	FAS-based	-23749.089	**100.00**	**1.00 ± 0.00**
Mushroom	Random	-80384.174	3.00	5.30 ± 2.56
	DFS-based	**-80384.107**	3.00	5.42 ± 2.31
	FAS-based	-80384.174	**74.00**	**2.11 ± 0.62**
Sensors	Random	-80545.820	1.00	6.89 ± 2.53
	DFS-based	**-80542.640**	1.00	6.10 ± 2.55
	FAS-based	-80545.820	1.00	**5.58 ± 2.26**
SteelPlates	Random	-25159.708	3.00	4.54 ± 1.68
	DFS-based	-25159.706	5.00	4.64 ± 1.64
	FAS-based	**-25157.926**	**100.00**	**1.00 ± 0.00**
Epigenetics	Random	-215563.134	94.00	3.58 ± 1.19
	DFS-based	-215563.134	90.00	3.48 ± 1.18
	FAS-based	-215563.134	**100.00**	**1.00 ± 0.00**

Table II: Best score obtained, Percentage of solutions that converge to best score (% Solutions), and Average number of iterations (Avg. It.) using each approach (best values in bold)

REFERENCES

Chickering, D. M., Heckerman, D., and Meek, C. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research* 5 (1): 1287–1330, 2004.

Chickering, D. M. and Meek, C. Finding Optimal Bayesian Networks. *Journal of Machine Learning Research*, 2004.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory.* Wiley-Interscience, 1991.

Heckerman, D., Geiger, D., and Chickering, D. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Journal of Machine Learning Research* 20 (MSR-TR-94-09): 197–243, 1995.

Knuth. *The Art of Computer Programming 2.* Boston: Adison-Wesley, 1998.

Lam, W. and Bacchus, F. Learning Bayesian Belief Networks. An approach based on the MDL principle. *Computational Intelligence* 10 (4): 31, 1994.