Curriculum Learning of Bayesian Network Structures

Yanpeng Zhao[1] ZHAOYP1@SHANGHAITECH.EDU.CN
Yetian Chen[2] YETIANC@IASTATE.EDU
Kewei Tu[1] TUKW@SHANGHAITECH.EDU.CN
Jin Tian[2] JTIAN@IASTATE.EDU
[1]*School of Information Science and Technology, ShanghaiTech University, Shanghai, China*
[2]*Department of Computer Science, Iowa State University, Ames, IA, USA*

Abstract

Bayesian networks (BNs) are directed graphical models that have been widely used in various tasks for probabilistic reasoning and causal modeling. One major challenge in these tasks is to learn the BN structures from data. In this paper, we propose a novel heuristic algorithm for BN structure learning that takes advantage of the idea of *curriculum learning*. Our algorithm learns the BN structure by stages. At each stage a subnet is learned over a selected subset of the random variables conditioned on fixed values of the rest of the variables. The selected subset grows with stages and eventually includes all the variables. We prove theoretical advantages of our algorithm and also empirically show that it outperformed the state-of-the-art heuristic approach in learning BN structures.

Keywords: Bayesian networks, structure learning, curriculum learning.

1. Introduction

A Bayesian network (BN) is a directed acyclic graph (DAG) where nodes are random variables and directed edges represent probability dependencies among variables. Each node and its parents are associated with a conditional probability distribution (CPD), which quantifies the effect of the parents on the node. A BN provides a compact representation of a joint probability distribution over the set of random variables. These models are also called belief networks, or causal networks, because the directed edges are sometimes interpreted as causal relations. BNs have been widely used in various tasks for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2001). A major challenge in these tasks is to learn the structure and the associated CPDs from data.

Learning a BN is usually conducted in two phases. In the first phase, one manages to construct the topology (structure) of the network. In the second phase, one then estimates the parameters of the CPDs given the fixed structure. Parameter estimation in the second phase is considered a well-studied problem. The learning of the topology, or in other words, structure learning, is more challenging.

1.1. Previous Work on Bayesian Network Structure Learning

There has been an enormous amount of work on learning BNs from data. Methods for this learning problem fall into two categories: constraint-based and score-based.

Constraint-based algorithms look for conditional independencies (CIs) in the data and build the DAG consistent to these CIs. Well-known examples are the Peter-Clark (PC) algorithm (Spirtes et al., 2001) and the Inductive-Causation (IC) algorithm (Pearl, 2000). Other later work includes the Grow-Shrink (GS) and Total-Conditioning (TC) algorithms (Margaritis and Thrun, 2000; Pellet and Elisseeff, 2008). PC or IC algorithms are guaranteed to return a DAG that exactly represents the CI relationships implied by the target distribution if all CI tests are perfect. Such assumption certainly does not hold in reality since any kind of statistical test will have some probability of making errors given limited data samples. Even worse, an error of a statistical test can result in propagated errors in the subsequent learning process.

Score-based approaches convert the problem to an optimization problem, where a *decomposable score* is used to measure the fitness of a DAG to the observed data, then a search strategy is employed to maximize (or minimize) the score over the space of possible DAGs (Cooper and Herskovits, 1992; Heckerman et al., 1995). However, finding an optimal BN structure is NP-hard (Chickering, 1996). Algorithms in this category include exact algorithms that are able to find an optimal solution or heuristic algorithms that often return sub-optimal models. The research on exact algorithms started with a family of algorithms using dynamic programming (DP) (Koivisto and Sood, 2004; Silander and Myllymäki, 2006). These DP algorithms require exponential time and space, thus are only applicable to problems with up to 30 variables. Latest work in this area used A* or other techniques to prune the search space so that both time and space complexities were greatly reduced (Yuan et al., 2011; Malone et al., 2011; Yuan and Malone, 2012). Recently, a set of algorithms using Integer Linear Programming (ILP) are showed being competitive with the A* algorithm (Jaakkola et al., 2010; Bartlett and Cussens, 2013). In particular, GOBNILP (Globally Optimal BN learning using ILP) was demonstrated to be able to handle problems with up to a few hundred variables (Bartlett and Cussens, 2013). However, it assumes the *indegree* of each node is upper-bounded by a small constant.

Heuristic search method encompasses a broad class of algorithms, varying in the scoring functions being used, the search strategies being employed, and assumptions being made. The general search strategy is, given a starting point, i.e., any DAG, by adding, deleting or reversing one or a few edges, the algorithm manages to traverse the DAG space to find a high-scoring model. However, there are super-exponential number of possible DAGs. Thus, local search strategies such as greedy or more sophisticated search algorithms are often used. The searches will often stuck in local maxima.

Finally, ideas combining both constraint-based and score-based approaches have also been explored. A well-known algorithm is Max-Min Hill-Climbing (MMHC) algorithm (Tsamardinos et al., 2006). MMHC first estimates the parents and children (PC) set of each variable using a local discovery algorithm called *MMPC* (Tsamardinos et al., 2003). It then performs a Bayesian-scoring greedy hill-climbing search with the constraint that the neighbors of each variable must be in the variable's PC set. Extensive empirical evaluation showed that MMHC outperformed on average other heuristic algorithms thus it was claimed to be the current state-of-the-art. MMHC is a two-stage algorithm. In the first stage, the identification of the PC sets involves a large number of CI tests, which are very sensitive to noises in data. Thus, the problem of constraint-based method may also trap MMHC.

An empirical evaluation of the impact of learning strategies on the quality of BNs can be found in (Malone et al., 2015).

1.2. Curriculum Learning

Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which starts from relatively simple concepts and gradually introduces more complex ones. This idea has been formalized in the context of machine learning as *curriculum learning* (Bengio et al., 2009). A curriculum is a sequence of weighting schemes of the training data, denoted by $(W_1, W_2, ..., W_m)$. The first scheme W_1 assigns more weight to "easier" training samples, and each next scheme assigns slightly more weight to "harder" examples, until the last scheme W_m that assigns uniform weight to all examples. Ideally, the information entropy of the weighting scheme shall increase monotonically, i.e., $\forall i < j$, $H(W_i) < H(W_j)$. How to measure the "easiness" or complexity of training samples may vary depending on the learning problems, and no general measurement has been proposed. Learning is done iteratively, each time from the training data weighted by the current weighting scheme and initialized with the learning result from the previous iteration.

Curriculum learning has been successfully applied to many problems, such as learning language models and grammars (Elman, 1993; Spitkovsky et al., 2010; Tu and Honavar, 2011) and object recognition and localization (Kumar et al., 2010). There have also been attempts to explain the advantages of curriculum learning. Bengio et al. (2009) proposed that a well chosen curriculum strategy can act as a continuation method (Allgower and Georg, 1990), which first optimizes a highly smoothed objective and then gradually considers less smoothing. Tu and Honavar (2011) contended that in learning structures such as grammars, an ideal curriculum decomposes the structure into multiple components and guides the learner to incrementally construct the target structure. More recently, a few extensions of the original idea of curriculum learning have been proposed (Kumar et al., 2010; Jiang et al., 2015).

In BN structure learning, one needs to discover the conditional independence/dependence between variables. Most of the current approaches try to learn these relations between all the variables by looking at all the training samples at once. In contrast, human rarely look at all the variables and samples at once during learning; instead, they learn in a more organized way, starting with more common data samples that involve dependency relations between only a small subset of variables (typically with the other variables fixed at certain values), and only turning to less common data samples involving dependency relations with additional variables when some knowledge (i.e., a partial model) is obtained. In this way, learning could be made more accurate and even more efficient. This learning strategy can be seen as a type of curriculum learning, related to the idea of incremental construction (Tu and Honavar, 2011) mentioned above. In this paper, we design a novel heuristic algorithm for BN structure learning that takes advantage of this idea. Our algorithm learns the BN structure by stages. At each stage a subnet is learned over a selected subset of the random variables conditioned on fixed values of the rest of the variables. The selected subset grows with stages until it includes all the variables at the final stage. Figure 1 shows an illustrative example of our algorithm. We prove theoretical advantages of our algorithm and also

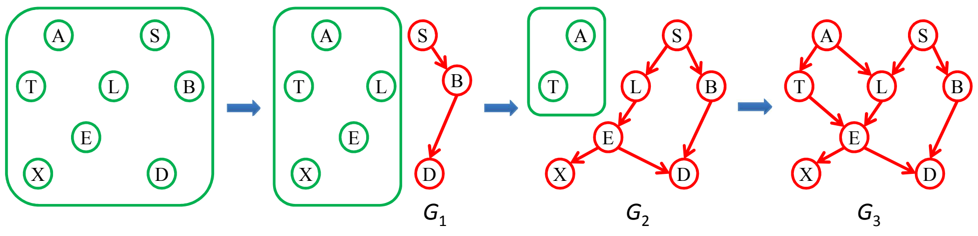


Figure 1: An illustrative example of curriculum learning of a BN structure. Given a curriculum $(\{S, B, D\}, \{S, B, D, L, E, X\}, \{S, B, D, L, E, X, A, T\})$ and step size $t = 3$, we learn the BN structure in three stages: (1) learn a subnet G_1 over $\{S, B, D\}$ from scratch; (2) learn a larger subnet G_2 over $\{S, B, D, L, E, X\}$ with G_1 as the start point of search; (3) learn a full network with G_2 as the start point. Each subnet (in red) is conditioned on the rest of the variables (in green).

empirically show that it outperformed the state-of-the-art heuristic approach in learning BN structures.

2. Preliminaries

Formally, a Bayesian network is a pair $B = (G, P)$, where G is a DAG that encodes a joint probability distribution P over a vector of random variables $\mathbf{X} = (X_1, ..., X_n)$ with each node of the graph representing a variable in $\mathbf{X}$. For convenience we typically work on the index set $V = \{1, ..., n\}$ and represent a variable X_i by its index i. The DAG can be represented as a vector $G = (Pa_1, ..., Pa_n)$ where each Pa_i is a subset of the index set V and specifies the parents of i in the graph.

Definition 1 ***(I-equivalence)*** *(Verma and Pearl, 1990) Two DAG G_1 and G_2 over the same set of variables $\mathbf{X}$ are I-equivalent if they represent the same set of conditional independence relations.*

Two I-equivalent DAGs are statistically indistinguishable. That is, given observational data, it is impossible to identify a unique data-generating DAG unless there is only one DAG in the corresponding equivalence class (EC). All BNs in an EC have the same skeleton and the same v-structures[1] (Verma and Pearl, 1990). Thus, we can represent an EC using a complete partially DAG (CPDAG) which consist of a directed edge for every irreversible edge and an undirected edge for every reversible edge.[2]

In the problem of learning BNs from data, we assume that a *complete* data D are generated $i.i.d$ from an underlying distribution $P^*(\mathbf{X})$ which is induced by some BN $B^* = (G^*, P^*)$. Due to the so called I-equivalence, the best we can hope is to recover G^*'s equivalence class. That is, we target any G that is I-equivalent to G^*.

In this paper, we focus on score-based search. The first component of the score-and-search method is a scoring criterion measuring the fitness of a DAG G to the data D. Several commonly used scoring functions are MDL, AIC, BIC and Bayesian score. In this

1. A v-structure in a DAG G is an ordered triple of nodes (u, v, w) such that G contains the directed edges $u \to v$ and $w \to v$ and u and w are not adjacent in G.
2. A CPDAG is also called a *pattern*. Each equivalence class has a unique CPDAG.

work, we use Bayesian score, defined as follows.

$$score(G : D) = \log P(D|G) + \log P(G), \tag{1}$$

where $P(D|G)$ is the likelihood of the data given the DAG G and $P(G)$ is the prior over the DAG structures. In this work, we assume discrete random variables which follow a Dirichlet-Multinomial distribution. That is, each variable X_i follows a multinomial distribution with parameter vector Θ_i conditioning on its parents, and the parameter vector Θ_i follows a Dirichlet distribution with parameter vector α_i as the prior. Assuming global and local parameter independence, parameter modularity, and uniform structure prior $P(G)$, the score is decomposable (Heckerman et al., 1995):

$$score(G : D) = \sum_{i=1}^{n} score_i(Pa_i : D), \tag{2}$$

where $score_i(Pa_i : D)$ is called the local score or family score measuring how well a set of variables Pa_i serves as parents of X_i. $score_i(Pa_i : D)$ can be computed efficiently from the sufficient statistics of the data D. Further we can show for any two I-equivalent DAGs G_1 and G_2, $score(G_1 : D) = score(G_2 : D)$. This is called score equivalence. The commonly used scoring functions such as MDL, AIC, BIC and BDe all satisfy score decomposability and equivalence.

3. Curriculum Learning of Bayesian Network Structures

The basic idea to apply curriculum learning and incremental construction in BN structure learning is that we can define a sequence of intermediate learning targets $(G_1, ..., G_m)$, where each G_i is a subnet of the target BN over a subset of variables $\mathbf{X}_{(i)}$ conditioned on certain fixed values $\mathbf{x}'_{(i)}$ of the rest of the variables $\mathbf{X}'_{(i)}$, where $\mathbf{X}_{(i)} \subseteq \mathbf{X}$, $\mathbf{X}'_{(i)} = \mathbf{X} \setminus \mathbf{X}_{(i)}$ and $\mathbf{X}_{(i)} \subset \mathbf{X}_{(i+1)}$; at stage i of curriculum learning, we try to learn G_i from a subset of data samples with $\mathbf{X}'_{(i)} = \mathbf{x}'_{(i)}$. In terms of the sample weighting scheme $(W_1, W_2, ..., W_m)$, each W_i assigns 1 to those samples with $\mathbf{X}'_{(i)} = \mathbf{x}'_{(i)}$ and 0 to the other samples.

However, training samples are often very limited in practice and thus the subset of samples with $\mathbf{X}'_{(i)} = \mathbf{x}'_{(i)}$ would typically be very small. Learning from such small-sized training sample is deemed unreliable. A key observation is that when we fix $\mathbf{X}'_{(i)}$ to different values, our learning target is actually the same DAG structure G_i but with different parameters (CPDs). Thus, we can make use of all the training samples in learning G_i at each stage by revising the scoring function to take into account multiple versions of parameters. This strategy extends the original curriculum learning framework.

Note that in the ideal case, the subnet learned in each stage would have only one type of discrepancy from the truth BN: it would contain extra edges between variables in $\mathbf{X}_{(i)}$ due to conditioning on $\mathbf{X}'_{(i)}$. More specifically, such variables in $\mathbf{X}_{(i)}$ must share a child node that is in, or has a descendant in, $\mathbf{X}'_{(i)}$ such that they are not d-separated when conditioning on $\mathbf{X}'_{(i)}$.

3.1. Scoring Function

In this paper, we use Bayesian score to design a scoring function that uses all training samples. Assume the domain for $\mathbf{X}'_{(i)}$ is $\{\mathbf{x}'_{(i),1}, ..., \mathbf{x}'_{(i),q}\}$. Then we can have a set of data segments $D_i = \{D_{i,1}, ..., D_{i,q}\}$ by grouping samples based on the values of $\mathbf{X}'_{(i)}$ and then projecting on $\mathbf{X}_{(i)}$. Assuming $D_{i,1}, ..., D_{i,q}$ are generated by the same DAG G_i but with "independent" CPDs, we can derive

$$P(G_i, D_i) = P(G_i) \prod_{j=1}^{q} P(D_{i,j}|G_i) = P(G_i)^{1-q} \prod_{j=1}^{q} P(G_i, D_{i,j}). \tag{3}$$

If we take logarithm for both sides, we obtain

$$\log P(G_i, D_i) = (1-q) \log P(G_i) + \sum_{j=1}^{q} \log P(G_i, D_{i,j}). \tag{4}$$

If we set uniform prior for G_i, i.e., $P(G_i) \propto 1$, we then have

$$\log P(G_i, D_i) = C + \sum_{j=1}^{q} \log P(G_i, D_{i,j}), \tag{5}$$

where C is a constant. We use BDe score (Heckerman et al., 1995) for discrete variables, i.e., $\log P(G_i, D_{i,j}) = score_{BDe}(G_i, D_{i,j})$, so we have the following scoring function

$$score(G_i, D_i) = \sum_{j=1}^{q} score_{BDe}(G_i, D_{i,j}), \tag{6}$$

i.e., the sum of BDe scores which are individually evaluated on each of the data segments $D_{i,1}, ..., D_{i,q}$.

One common problem with curriculum learning is that the learner may overfit the intermediate learning targets, especially when the number of variables is large and thus we have to divide learning into many stages. Overfitting also occurs when the sample size is small. Therefore, we introduce a penalty function that penalizes the size of the network especially when the number of variables is large or the sample size is small:

$$Penalty(G_i : D_i) = \left(\frac{a}{SS} + \frac{V(G_i)}{b} \right) E(G_i), \tag{7}$$

where SS is the sample size, $V(G_i)$ and $E(G_i)$ denote the number of variables and number of edges in G_i respectively, and a and b are positive constants. Combined with the penalty function, the scoring function becomes

$$score(G_i : D_i) = \sum_{j=1}^{q} score_{BDe}(G_i, D_{i,j}) - \left(\frac{a}{SS} + \frac{V(G_i)}{b} \right) E(G_i). \tag{8}$$

3.2. Curriculum

A remaining fundamental question is: what *curriculum*, i.e., the sequence of variable sets $(\mathbf{X}_{(1)}, ..., \mathbf{X}_{(m)})$, shall we use? Or equivalently, from stage $i-1$ to i, which variables $\mathbf{X}_{(i-1,i)}$ should we select to produce $\mathbf{X}_{(i)} = \mathbf{X}_{(i-1)} \cup \mathbf{X}_{(i-1,i)}$?

Intuitively, we should select the variables that are most connected to the current set of variables $\mathbf{X}_{(i-1)}$, because otherwise we may learn more edges that do not exist in the target BN. The problem is that the true structure of the target BN is unknown. Nonetheless, we can measure the strength of the dependency (e.g., using mutual information) with the current set of variables to heuristically estimate the connectivity.

Thus, in stage i, we compute pairwise mutual information $MI(X, Y)$ between any node X in $\mathbf{X}_{(i-1)}$ and node Y in $\mathbf{X} \setminus \mathbf{X}_{(i-1)}$. Then for any node Y in $\mathbf{X} \setminus \mathbf{X}_{(i-1)}$, compute the average pairwise mutual information by

$$AveMI(Y, \mathbf{X}_{(i-1)}) = \sum_{X \in \mathbf{X}_{(i-1)}} MI(X, Y)/|\mathbf{X}_{(i-1)}|. \tag{9}$$

We then pick the variables in a sequential way: we first pick variable Y_1 with the largest $AveMI(Y_1, \mathbf{X}_{(i-1)})$; we then pick the second variable Y_2 with the largest $AveMI(Y_2, \mathbf{X}_{(i-1)} \cup \{Y_1\})$; so on and so forth. The number of variables selected, $|\mathbf{X}_{(i-1,i)}|$, is called the *step size* and is a parameter of our algorithm. The step size can be a constant, meaning that we add the same number of variables in each stage. Or it can be different among stages. Intuitively, the smaller the step size is, the more cautious and less time-efficient the algorithm is, and also the more likely the algorithm would overfit the intermediate BNs.

We initialize $\mathbf{X}_{(1)}$ with a small number of variables where the first variable has the largest $AveMI$ with all the other variables in X and the rest are selected in the sequential way as described above.

3.3. Algorithm

Given the training data D and step size t, we first construct the curriculum $(\mathbf{X}_{(1)}, ..., \mathbf{X}_{(m)})$. In each learning stage of the curriculum, we use score-based search to find a good partial network with the partial network learned from the previous stage plus the new variables with no edge attached as the start point. Algorithm 1 sketches our algorithm, in which $search(D_i, \mathbf{X}_{(i)}, S, G_{i-1})$ can be any search algorithm that starts from G_{i-1} and searches the space of DAGs over variables $\mathbf{X}_{(i)}$ to optimize our scoring function with training data D_i; S is the parents and children (PC) set generated from the MMPC algorithm (Tsamardinos et al., 2003) that is used to constrain the search, i.e., only edges included in the PC set are considered during search. In our implementation, we simply use greedy hill climbing as the search algorithm.

In some stages, the number of data segments does not change although additional variables are selected. In this case, it can be shown that the subnet learned from the previous stage plus the new variables with no edge attached is already a local optimum, and therefore we can skip the stage without changing the learning result. In practice, we skip a stage whenever the number of data segments has no change or very small change from that of the previous stage.

Algorithm 1 Curriculum Learning of BN structure

```
require: variable set $\mathbf{X}$, training data $D$, PC set $S$, step size $t$, curriculum $(\mathbf{X}_{(1)}, ..., \mathbf{X}_{(m)})$.
Initialize $G_0$ to a network containing variables in $\mathbf{X}_{(1)}$ with no edge.
$i \leftarrow 1$
for $i \leq m$ do
    Generate the set of data segments $D_i = \{D_{i,1}, ..., D_{i,q}\}$ based on the values of $\mathbf{X} \setminus \mathbf{X}_{(i)}$
    $G_i \leftarrow search(D_i, \mathbf{X}_{(i)}, S, G_{i-1})$
    $i \leftarrow i + 1$
end
return $G_m$
```

4. Theoretical Analysis

Curriculum learning specifies a sequence of intermediate learning targets. Ideally, each intermediate target should be closer to the subsequent targets than any of its predecessors in the sequence. In this section we show that our curriculum learning approach to learning BNs satisfies this desired property.

With Bayesian networks as our learning targets, there are two different ways to measure the distance between them. The first is to measure the distance between the structures of two BNs. One such distance measure is the structural Hamming distance (*SHD*) (Tsamardinos et al., 2006), which measures the number of extra, missing or differently oriented edges between the two CPDAGs that respectively represent the equivalence classes of two BNs. The second is to measure the distance between the probabilistic distributions defined by two BNs. One such distance measure is the total variation distance (Csisz et al., 1967). With discrete random variables, the total variation distance between two distributions can be defined as:

$$d_{TV}(P, Q) = \frac{1}{2} \sum_{\mathbf{X}} |P(\mathbf{X}) - Q(\mathbf{X})|$$

Below we analyze our curriculum learning approaches based on these two types of distance measures respectively and show that our approach satisfies the desired property based on both distance measures.

4.1. Analysis Based on Distance between Structures

Suppose $\mathbf{X}_{(i)}$ is the set of variables selected in curriculum stage i and $\mathbf{X}'_{(i)} = \mathbf{X} \setminus \mathbf{X}_{(i)}$ is the rest of the variables. Recall that we try to learn a subnet of the true BN over variables in $\mathbf{X}_{(i)}$ that is *conditioned on* fixed values of variables in $\mathbf{X}'_{(i)}$. Therefore, the actual learning target at stage i is a BN G_i such that: (a) between variables in $\mathbf{X}_{(i)}$, the edges are connected in accordance with the true BN except that there might be extra edges between variables that share one or more descendants in $\mathbf{X}'_{(i)}$ (recall that the values of the variables in $\mathbf{X}'_{(i)}$ are fixed at stage i); (b) the variables in $\mathbf{X}'_{(i)}$ are fully connected with each other (because at stage i we regard the joint assignments to the variables in $\mathbf{X}'_{(i)}$ as the conditions and do not model any conditional independence between them); (c) there is an edge between each variable in $\mathbf{X}_{(i)}$ and each variable in $\mathbf{X}'_{(i)}$ (because the subnet over $\mathbf{X}_{(i)}$ is conditioned on all the variables in $\mathbf{X}'_{(i)}$). The orientation of the edges described in (b) and (c) can be arbitrary since it is not actually to be learned at stage i , but if we assume that these edges are oriented in a way that is consistent with the true BN then we have the following theorem.

Theorem 2 *For any i, j, k s.t. $1 \leq i < j < k \leq n$, we have*

$$d_H(G_i, G_k) \geq d_H(G_j, G_k)$$

where $d_H(G_i, G_j)$ is the structural Hamming distance (SHD) between the structures of two BNs G_i and G_j.

Proof At each stage of the curriculum, a set of variables $\mathbf{V} = \mathbf{X}_{(i)} \backslash \mathbf{X}_{(i-1)}$ become selected. This leads to two changes to the intermediate target BN: first, some extra edges between variables in $\mathbf{X}_{(i-1)}$ that share descendants in $\mathbf{V}$ are removed because their descendants no longer have fixed values; second, some edges connected to variables in V are removed to make the subnet of the variables in $\mathbf{X}_{(i)}$ consistent with the true BN. In other words, we always remove edges and never add or re-orient any edge of the BN at each stage of the curriculum. Since the corresponding CPDAG has the same structure as the BN except for some edges becoming undirected, it can also be shown that only edge-removal occurs to the CPDAG at each stage of the curriculum. Therefore, the structural Hamming distance $d_H(G_i, G_j)$ is simply the number of edges removed during stages $i+1$ to j. Since $i < j < k$, the set of edges removed during stages $i+1$ to k is a superset of the set of edges removed during stages $j+1$ to k. Therefore, we have $d_H(G_i, G_k) \geq d_H(G_j, G_k)$. ■

4.2. Analysis Based on Distance between Distributions

Based on the discussion in the previous subsection, it can be seen that the intermediate learning target G_i of stage i represents a probabilistic distribution $P(\mathbf{X}_{(i)}|\mathbf{X}'_{(i)})Q(\mathbf{X}'_{(i)})$, where P denotes the true conditional distribution of $\mathbf{X}_{(i)}$ given $\mathbf{X}'_{(i)}$ as represented by the target BN, and Q denotes an estimated distribution over $\mathbf{X}'_{(i)}$ (e.g., simply estimated based on the histogram built from the training data). We can prove the following theorem.

Theorem 3 *For any i, j, k s.t. $1 \leq i < j < k \leq n$, we have*

$$d_{TV}(G_i, G_k) \geq d_{TV}(G_j, G_k)$$

where $d_{TV}(G_i, G_j)$ is the total variation distance between the two distributions defined by the two BNs G_i and G_j.

Proof For any $i < j$, let $\mathbf{Y}_{ij} = \mathbf{X}_{(j)} \setminus \mathbf{X}_{(i)}$. We have

$$\begin{aligned} d_{TV}(G_i, G_j) &= \frac{1}{2} \sum_{\mathbf{X}} \left| P(\mathbf{X}_{(i)}|\mathbf{X}'_{(i)})Q(\mathbf{X}'_{(i)}) - P(\mathbf{X}_{(j)}|\mathbf{X}'_{(j)})Q(\mathbf{X}'_{(j)}) \right| \\ &= \frac{1}{2} \sum_{\mathbf{X}} P(\mathbf{X}_{(i)}|\mathbf{X}'_{(i)}) \left| Q(\mathbf{X}'_{(i)}) - P(\mathbf{Y}_{ij}|\mathbf{X}'_{(j)})Q(\mathbf{X}'_{(j)}) \right| \\ &= \frac{1}{2} \sum_{\mathbf{X}'_{(i)}} \left| Q(\mathbf{X}'_{(i)}) - P(\mathbf{Y}_{ij}|\mathbf{X}'_{(j)})Q(\mathbf{X}'_{(j)}) \right| \end{aligned}$$

Therefore, we have

$$d_{TV}(G_i, G_k) = \frac{1}{2} \sum_{\mathbf{X}'_{(j)}} \sum_{\mathbf{Y}_{ij}} \left| Q(\mathbf{X}'_{(i)}) - P(\mathbf{Y}_{ik}|\mathbf{X}'_{(k)}) Q(\mathbf{X}'_{(k)}) \right|$$

and

$$\begin{aligned} d_{TV}(G_j, G_k) &= \frac{1}{2} \sum_{\mathbf{X}'_{(j)}} \left| Q(\mathbf{X}'_{(j)}) - P(\mathbf{Y}_{jk}|\mathbf{X}'_{(k)}) Q(\mathbf{X}'_{(k)}) \right| \\ &= \frac{1}{2} \sum_{\mathbf{X}'_{(j)}} \left| \sum_{\mathbf{Y}_{ij}} Q(\mathbf{X}'_{(i)}) - \sum_{\mathbf{Y}_{ij}} P(\mathbf{Y}_{ik}|\mathbf{X}'_{(k)}) Q(\mathbf{X}'_{(k)}) \right| \end{aligned}$$

Because the absolute value is subadditive, we have

$$\sum_{\mathbf{Y}_{ij}} \left| Q(\mathbf{X}'_{(i)}) - P(\mathbf{Y}_{ik}|\mathbf{X}'_{(k)}) Q(\mathbf{X}'_{(k)}) \right| \geq \left| \sum_{\mathbf{Y}_{ij}} \left(Q(\mathbf{X}'_{(i)}) - P(\mathbf{Y}_{ik}|\mathbf{X}'_{(k)}) Q(\mathbf{X}'_{(k)}) \right) \right|$$

Therefore,

$$d_{TV}(G_i, G_k) \geq d_{TV}(G_j, G_k)$$

■

5. Experiments

In this section, we empirically evaluate our algorithm and compare it with MMHC (Tsamardinos et al., 2006), the current state-of-the-art heuristic algorithm in BN structure learning. For both algorithms, we used BDeu score (Heckerman et al., 1995) with the equivalent sample size 10 as the scoring function and used the MMPC module included in Causal Explorer (Aliferis et al., 2003) with the default setting to generate the PC set. For MMHC, we used the settings mentioned by Tsamardinos et al. (2006).

We collected 10 benchmark BNs from the `bnlearn` repository[3]. The statistics of these BNs are shown in Table 1. From each of these BNs, we generated datasets of various sample sizes ($SS = 100, 500, 1000, 5000, 10000, 50000$). For each sample size, we randomly generated 5 datasets and reported the algorithm performance averaged over these 5 datasets.

When running our algorithm on each dataset, we set the step size (introduced in section 3.2) to 1, 2 and 3 and learned three BNs; we also learned a BN by hill climbing with no curriculum. We then picked the BN with the largest BDeu score as the final output. In our experiments, we find that only on a small fraction (50 out of 300) of the datasets did hill climbing with no curriculum produce the final output. More detailed statistics regarding the effect of step size on final output can be found in Table S1 of the supplemental material. We tuned the parameter a and b of the penalty function (Equation 7) on a separate validation set and fixed them to 1000 and 100 respectively.

3. http://www.bnlearn.com/bnrepository/

Table 1: Bayesian networks used in experiments.

Network	Num. vars	Num. edges	Max in/out-degree	Cardinality range	Average cardinality
alarm	37	46	4/5	2-4	2.84
andes	223	338	6/12	2-2	2.00
asia	8	8	2/2	2-2	2.00
child	20	25	2/7	2-6	3.00
hailfinder	56	66	4/16	2-11	3.98
hepar2	70	123	6/17	2-4	2.31
insurance	27	52	3/7	2-5	3.30
sachs	11	17	3/6	3-3	3.00
water	32	66	5/3	3-4	3.63
win95pts	76	112	7/10	2-2	2.00

Cardinality denotes the number of values that a variable can take.

We used four metrics to evaluate the learned BNs: BDeu, BIC, KL and SHD. The first three metrics were evaluated on a separate test dataset of 5000 samples for each BN. The BDeu score, the scoring function used in our learning algorithms, measures how likely the network is given the data. BIC (Bayesian information criterion) can be regarded as the likelihood of the learned structure after having seen the data with a penalty term of model complexity measured by the number of parameters:

$$BIC(G:D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log(N)\sum_{i=1}^{n}(r_i-1)q_i, \tag{10}$$

where n denotes the number of variables, r_i denotes the number of values that X_i can take, $q_i = \prod_{X_l \in Pa_i} r_l$ denotes the number of values that the parent set Pa_i of X_i can take, N_{ijk} is the number of samples in D where $X_i = k$ and $Pa_i = j$, and N_{ij} is the number of samples with $Pa_i = j$ in D.

Both BDeu and BIC have the limitation that they are only reasonable under certain assumptions. To directly measure how close the gold-standard network and the learned network are, we used Kullback-Leibler divergence (KL) between the joint probability distributions associated respectively with the true network(P_T) and the learned network(P_L):

$$KL(P_T, P_L) = \sum_{\mathbf{X}} P_T(\mathbf{X})\log\left(\frac{P_T(\mathbf{X})}{P_L(\mathbf{X})}\right), \tag{11}$$

For the convenience of estimation, we used an equivalent form of Equation 11 by Acid and de Campos (2001):

$$KL(P_T, P_L) = -H_{P_T}(\mathbf{X}) + \sum_{X_i\in\mathbf{X}} H_{P_T}(X_i) - \sum_{X_i\in\mathbf{X}, Pa_L(X_i)\neq\emptyset} MI_{P_T}(X_i, Pa_L(X_i)), \tag{12}$$

where H_{P_T} denotes the Shannon entropy with respect to P_T. In Equation 12, the first two terms are not dependent on the learned network, so following Tsamardinos et al. (2006),

Table 2: Comparison between CL and MMHC on four metrics

		Sample Size (SS)					
Metric	Algorithm	100	500	1000	5000	10000	50000
BDeu	CL	1(0)	1(10)	1(9)	1(8)	1(10)	1(8)
	MMHC	0.89(10)	1.06(0)	1.02(1)	1.01(2)	1.02(0)	1.01(2)
BIC	CL	1(0)	1(9)	1(9)	1(6)	1(8)	1(8)
	MMHC	0.88(10)	1.07(1)	1.02(1)	1.02(4)	1.02(2)	1.01(2)
KL	CL	1(0)	1(10)	1(9)	1(7)	1(9)	1(9)
	MMHC	1.71(10)	0.82(0)	0.96(1)	0.96(2)	0.97(0)	0.97(0)
SHD	CL	1(7)	1(9)	1(7)	1(7)	1(8)	1(6)
	MMHC	1.06(3)	1.26(1)	1.29(3)	1.07(2)	1.21(1)	1.24(3)

Each number is an average normalized scores, i.e., the average of the ratios between the raw scores and the corresponding scores of CL (the ratios are averaged over 10 networks and 5 runs with randomly sampled training datasets on each network). For BDeu, BIC and SHD, smaller ratios indicate better learning results; for KL, larger numbers indicate better learning results. Each number in parentheses indicates the number of winning networks among the 10 networks, i.e., on how many networks the algorithm produced better results than its competitor. The number of draws (networks with equal scores) are not counted.

we only calculate and report the last term of the equation. Note that the last term appears with a negative sign, and hence the higher its value is, the smaller the KL-divergence is and the closer the learned network is to the true network.

Structural Hamming distance (SHD) is another distance metric, which directly measures the difference between the structures of the two networks as explained in section 4.

Table 2 shows the comparison between our algorithm (CL) and MMHC. Note that we choose to show the average ratios between the raw scores and the corresponding scores of CL. This is because the raw scores from different datasets vary significantly in order of magnitude, and the average of raw scores would be dominated by those from a small subset of the datasets. It can be seen that CL outperforms MMHC in almost all the cases, in terms of both the scores and the number of winning networks. A notable exception is that when $SS = 100$, CL under-performs MMHC on all the networks for three of the four metrics. We find that it is mainly because the penalty term (Equation 7) becomes too large when SS is very small, which drives the learner to produce a network with few edges. For example, on the Andes network with $SS = 100$, the learned network contains only around 50 edges while the number of edges in the true network is 338.

Since SHD is one of the most widely used evaluation metrics for BN structure learning, we further investigate the SHD scores of the two algorithms under different settings. Figure 2 plots the SHD averaged over five runs on the Andes, Hailfinder, Hepar2 and Win95pts networks (other plots are presented in Figure S1 of the supplemental material). It again shows that CL outperforms MMHC in almost all the cases.

In section 4 we have proved that each intermediate BN in our curriculum is closer to the subsequent BNs (including the target BN) than any of its predecessors. Here we would like

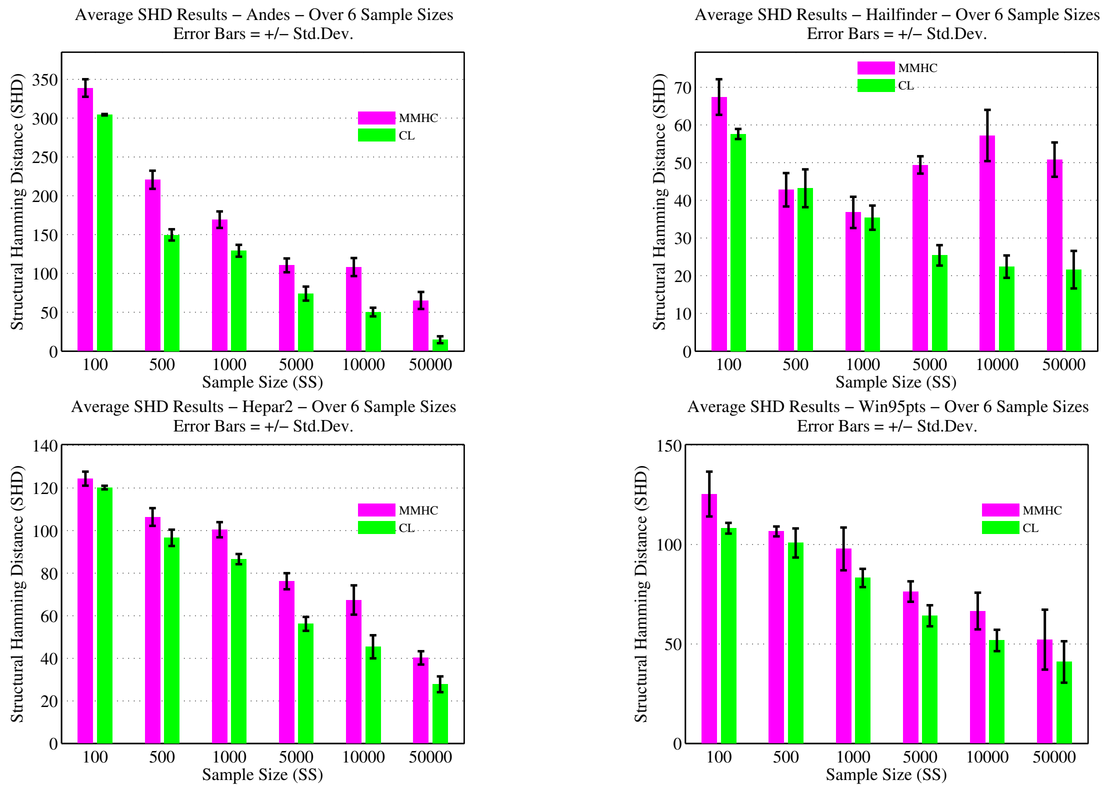
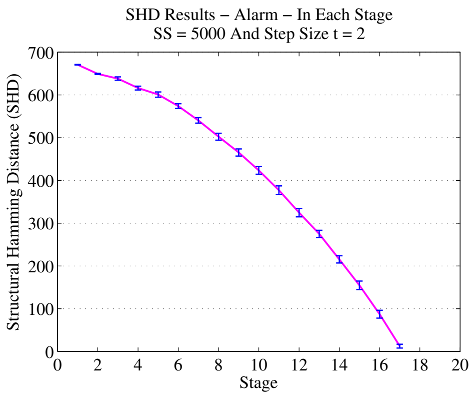


Figure 2: Comparison of the average SHD on the Andes, Hailfinder, Hepar2 and Win95pts networks between CL and MMHC

to empirically demonstrate that the learner is indeed guided by these intermediate target BNs to produce intermediate learning results that become increasingly closer to the target BN with more curriculum stages. Note that while at stage i of the curriculum we learn a subnet over the selected variables $\mathbf{X}_{(i)}$, this subnet is conditioned on fixed values of the rest of the variables $\mathbf{X}'_{(i)} = \mathbf{X} \setminus \mathbf{X}_{(i)}$. Hence we can view the intermediate learning result at stage i as a BN over all the variables consisting of three parts: (a) the learned subnet over $\mathbf{X}_{(i)}$; (b) a fully connected subnet over $\mathbf{X}'_{(i)}$; (c) a fully connected bipartite network between $\mathbf{X}_{(i)}$ and $\mathbf{X}'_{(i)}$. In order to correctly measure the distances between the intermediate learning results and the target BN, we first randomly generated a fully connected BN over all the variables, and then at each stage i we replaced the local structure over $\mathbf{X}_{(i)}$ with the subnet that we have learned. Figure 3 plots the SHD between the intermediate learning result at each stage and the target BN on two different networks. It can be seen that the intermediate learning results indeed become closer to the learning target with more curriculum stages.

With respect to running-time, our algorithm is in general slower than MMHC, on average taking 2.7 times as much time. One reason is that our algorithm has to perform hill climbing for multiple times, once at each stage, and the number of stages is proportional to the number of variables. Another reason is that our scoring function takes more time to compute: we have to compute a separate score for each data segment, which becomes slow when the data is partitioned into too many segments. The number of segments is



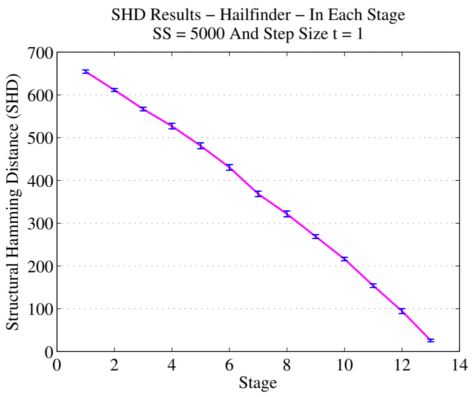


Figure 3: Changes of SHD from the target BN during curriculum learning with $SS = 5000$ on the Alarm and Hailfinder networks

determined by the number of variables as well as the cardinality of each variable. Our experiments show that the average cardinality of variables has a larger impact to the running time of our algorithm than the number of variables. With $SS = 5000$, the Andes network (223 variables with average cardinality of 2) takes only a few minutes for our algorithm to run, while the Mildew network (35 variables with average cardinality of 17.6) takes a few hours. To verify that the good performance of our algorithm does not come from the extra running time, we ran TABU search[4] of BN structures on each dataset with the same amount of time as used by our algorithm and found that our algorithm still has significantly better performance.

6. Conclusion and Discussion

In this paper, we propose a novel heuristic algorithm for BN structure learning that takes advantage of the idea of curriculum learning. Our algorithm learns the BN structure by stages. At each stage a subnet is learned over a selected subset of the random variables conditioned on fixed values of the rest of the variables. The selected subset grows with stages and eventually includes all the variables. We have tailored the scoring function for our algorithm and discussed an approach to order variables for curriculum construction. We have proved two theorems that show theoretical properties of our algorithm. Finally, we have empirically shown that our algorithm outperformed the state-of-the-art MMHC algorithm in learning BN structures.

While at each curriculum stage our algorithm tries to learn a network over a subset of the variables $\mathbf{X}_{(i)}$ conditioned on fixed values of the rest of the variables $\mathbf{X}'_{(i)}$, there is an obvious alternative approach which learns a network over $\mathbf{X}_{(i)}$ while ignoring $\mathbf{X}'_{(i)}$. In the ideal case, the subnet learned by this approach would have exactly one type of discrepancy from the true BN: it would contain extra edges between variables in $\mathbf{X}_{(i)}$ that cannot be d-separated in the true BN without fixing certain variables in $\mathbf{X}'_{(i)}$. In this alternative approach, the scoring function of the subnet can be computed much faster than in our

4. TABU search augments greedy hill-climbing by allowing worsening moves and using a *tabu* list to keep track of and avoid recently visited solutions.

original algorithm because no data partition is involved and hence we only need to compute a single score. However, the theoretical guarantees given in Theorem 2 and 3 are no longer true because counter-examples exist. Our experiments also showed that this approach in general resulted in worse learning accuracy than our original algorithm.

References

Silvia Acid and Luis M de Campos. A hybrid methodology for learning belief networks: Benedict. *International Journal of Approximate Reasoning*, 27(3):235–262, 2001.

Constantin F Aliferis, Ioannis Tsamardinos, Alexander R Statnikov, and Laura E Brown. Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery. In *METMBS*, volume 3, pages 371–376, 2003.

Eugene L Allgower and Kurt Georg. *Numerical continuation methods*, volume 13. Springer-Verlag Berlin, 1990.

Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13)*, pages 182–191, 2013.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from data*, pages 121–130. Springer, 1996.

Gregory F Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

I Csisz et al. Information-type measures of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.

Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99, 1993.

David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using lp relaxations. In *International Conference on Artificial Intelligence and Statistics*, pages 358–365, 2010.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *The Journal of Machine Learning Research*, 5:549–573, 2004.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23*. 2010.

Brandon Malone, Matti Järvisalo, and Petri Myllymäki. Impact of learning strategies on the quality of Bayesian networks: An empirical evaluation. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*, 2015.

Brandon M Malone, Changhe Yuan, and Eric A Hansen. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *AAAI*, 2011.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 2000.

Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press, 2000.

Jean-Philippe Pellet and André Elisseeff. Using markov blankets for causal structure learning. *The Journal of Machine Learning Research*, 9:1295–1342, 2008.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22th Conference on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search.* MIT Press, 2001.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. From baby steps to leapfrog: How “less is more” in unsupervised dependency parsing. In *NAACL*, 2010.

Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678. ACM, 2003.

Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

Kewei Tu and Vasant Honavar. On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1523, 2011.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, 1990.

Changhe Yuan and Brandon Malone. An improved admissible heuristic for learning optimal Bayesian networks. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12)*, 2012.

Changhe Yuan, Brandon Malone, and Xiaojian Wu. Learning optimal Bayesian networks using a* search. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, pages 2186–2191, 2011.