University of California

Los Angeles

Concave Penalized Estimation of Causal Gaussian Networks with Intervention

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Dacheng Zhang

2016

Abstract of the Thesis

Concave Penalized Estimation of Causal Gaussian Networks with Intervention

by

Dacheng Zhang

Master of Science in Statistics

University of California, Los Angeles, 2016

Professor Qing Zhou, Chair

We propose a penalized log-likelihood method for estimating causal Bayesian networks using a mix of intervention data and observation data, based on Aragam and Zhou's previous algorithm [AZ15]. A causal Bayesian network is represented as a directed acyclic graph (DAG). With intervention data, we can distinguish the true graph from DAG equivalence classes and learn about special structures in the graph. Tests on simulated data show that this method is consistent and is superior to other methods in multiple aspects such as true positive rate and timing. A two-stage approach to estimate big networks is proposed as an application of our method.

The thesis of Dacheng Zhang is approved.

Chad Hazlett

Arash Ali Amini

Qing Zhou, Committee Chair

University of California, Los Angeles

2016

To my grandmother . . .

who discovered my gift in mathematics

through a variety of board games and puzzles

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Bayesian network has seen its wide application in recent years in various modern areas such as gene regulatory networks, medicine, and text classification, following the growing demand of machine learning techniques. Bayesian network is useful for the fact that its graphic representation, which is a directed acyclic graph (DAG), can be used to represent conditional dependence relationships among variables. This encourages us to use Bayesian network to do causal inference on the data and phenomena we observed from these variables, in order to learn the mechanism behind them. In most cases, the primary task of causal inference is learning the structure of a causal Bayesian network, or equivalently, its DAG representation. Though it has been shown that structure learning of a Bayesian network is an NP-hard problem [CGH94], numerous estimating methods have been developed. Apart from using prior domain knowledge to facilitate the learning process, most of these methods can be categorized as constraint-based and score-based, as well as some hybrids of these two categories.

Constraint-based methods run repeated tests to detect the existence of edges between nodes. Edges are deleted if the connecting nodes are shown to be conditionally independent, so in the end the remaining edges will represent causal relationships between edges. PC algorithm [SGS00, KB07] and the Maximum Minimum Parents and Children (MMPC) algorithm [TBA06] are two classic examples among many others adopting this idea.

A score-based method, on the other hand, sets up a scoring function, usually consisting of likelihood and penalty terms, over all possible DAGs, and then search for one DAG that optimizes said scoring function. Classic choices of the scoring function include likelihood-equivalence Bayesian Dirichlet score metric [HGC95], minimum description length [LB94], and other types of metrics. Recent development of methods in this category are often accompanied with focus on sparse penalty. For instance, van de Geer and Bühlmann [GB13] used ℓ_0 penalized maximum likelihood, while Meinshausen and Bühlmann [MB06] and Fu and Zhou [FZ13] adopted ℓ_1 penalty.

Hybrid methods take advantages of both categories, as they first use constraint-methods to limit possible DAGs to a much smaller space, and then optimize for some scoring functions. The Max-Min Hill-Climbing (MMHC) algorithm [TBA06] is one famous example; another method under this category integrated an information-theory-based approach and a scoring-function-based approach [CAL08].

The basis algorithm of our new algorithm from Aragam and Zhou's work [AZ15] is a score-based method, as it requires neither prior domain knowledge nor conditional dependence tests; it adopts a concave penalty in a more generalized and flexible form. As a step forward, our algorithm presented in this article makes use of intervention data to better distinguish the true underlying DAG from its equivalence classes. Related works on using intervention include He and Geng's work [HG08] finding an optimal experiment design for intervention to correct the directions of edges; Hauser and Bühlmann [HB15] establishes maximum likelihood estimator for the DAG on a mixture of both observational and interventional data.

The rest of this article is organized into five sections. Section 2 shows the importance of intervention to help identify the true model among equivalence classes on general Bayesian network models, and then establish a formulation of the structure learning problem specific for causal Gaussian networks by translating it into an concave penalized optimization problem. Section 3 highlights core details of as well as supplements to the original algorithm, including several changes made to utilize intervention data. Section 4 summarizes numerical results of simulation studies on networks of different sizes, and compares our new algorithm with other methods. Section 5 proposes a two-stage approach, as an application of our new algorithm, to estimate Bayesian networks of even bigger sizes. Section 6 concludes the article with remarks and discussions.

CHAPTER 2

Preliminaries

Before we propose a detailed setup for our estimating method, we first present a general discussion on the necessity of intervention data in estimating Bayesian networks. A causal Bayesian network is often represented in two common ways. One way is the joint probability distribution: for a causal Bayesian network with p random variables $X_1, \ldots, X_p$, the joint probability distribution can be factorized as

$$P(X_1, \ldots, X_p) = \prod_{j=1}^{p} P(X_j | \Pi_j^G), \tag{2.1}$$

where Π_j^G is the set of "parents" of X_j; in other words, a change in one or some nodes in Π_j^G will directly "cause" a change in the distribution of X_j, eventually affecting observed outcomes of X_j. When X_j does not have any parents, we set $\Pi_j^G = \emptyset$. This is how the causal relationships are reflected in the joint distribution factorization.

On the other hand, the structure of a causal Bayesian network can also be represented by a DAG $\mathcal{G} = (V, E)$, with $V = \{1, \ldots, p\}$ as the set of nodes in the graph, and $E = \{(i, j) : X_i \in \Pi_j^{\mathcal{G}}, 1 \leq i, j \leq p\}$ as the set of directed edges. The acyclicity of $\mathcal{G}$ is essential for (2.1) to be well-defined, while the directions of edges will explicitly represent the causal relationships between nodes. The weights of edges are less relevant to the structure of a DAG so they are neglected in our method, since our primary task is inference on the structure of the causal Bayesian network.

To see how intervention would help determine a causal Bayesian network, let us consider the simplest case, with only two nodes, X_1 and X_2. We can factorize the joint probability in two ways:

$$P(X_1, X_2) = P(X_1)P(X_2|X_1) = P(X_2)P(X_1|X_2). \tag{2.2}$$

Each in this equivalence class represents a DAG $\mathcal{G}_1 : X_1 \to X_2$ and $\mathcal{G}_2 : X_2 \to X_1$, respectively, and can serve as a reasonable explanation of the casual relationship between these two nodes. With purely observational data only and no prior domain knowledge, we do not have any evidence to give preference to one DAG over the other one. Naturally we shall consider using experimental intervention as a test to gain evidence to distinguish the true causal relationships from others in its equivalence class.

Suppose during the data generation process we can fix X_2 with some known distribution $P(X_2|\cdot)$ which is independent of the DAG, while letting X_1 generated under the true mechanism. We will immediately see there is a difference between the new joint distribution assuming either $\mathcal{G}_1$ or $\mathcal{G}_2$ is the true DAG:

$$\tilde{P}(X_1, X_2) = \begin{cases} P(X_1)P(X_2|\cdot) & \text{, if } \mathcal{G}_1 : X_1 \to X_2 \text{ is true;} \\ P(X_2|\cdot)P(X_1|X_2) & \text{, if } \mathcal{G}_2 : X_2 \to X_1 \text{ is true.} \end{cases} \tag{2.3}$$

As a result of such difference, the log-likelihood of these two DAGs will also be different; we may then infer the true causal relationships by choosing the DAG that maximizes the log-likelihood function. This is how intervention will play a critical role in our proposed estimating method.

Note that in (2.3), the interventional distribution $P(X_2|\cdot)$ appears in both cases. It is also independent of the DAG, so it can be dropped when maximizing the log-likelihood to simplify the calculations.

To apply this general idea to a more specific setting, from now on we will focus on Gaussian networks; we assume the data, without any intervention, come from a p-variate Gaussian distribution:

$$(X_1, \ldots, X_p) \sim \mathcal{N}(0, \Sigma), \tag{2.4}$$

This can be rewritten as a system of Gaussian structural equations as follows:

$$X_j = \sum_{i=1}^{p} \beta_{ij} X_i + \varepsilon_j, \quad j = 1, \ldots, p. \tag{2.5}$$

Here ε_j's are mutually independent noises with $\varepsilon_j \sim \mathcal{N}(0, \omega_j^2)$ and independent of Π_j^G as well; let $\Omega = (\omega_1^2, \ldots, \omega_p^2) \in \mathbb{R}_+^p$ be a vector collecting the variances. Coefficients β_{ij}^0 take non-zero values if and only if $i \in \Pi_j^{\mathcal{G}}$. Let $B = (\beta_{ij})_{p \times p}$ be the coefficient matrix. It is possible to construct a DAG from B, denoted as $\mathcal{G}_B$, containing all edges $i \to j$ whenever $\beta_{ij} \neq 0$. We refer to $\mathcal{G}_B$ as the graph induced by B.

As for interventions, we assume the interventions on each X_j follow $\mathcal{N}(0, \hat{\sigma}^2)$ i.i.d. for all j. A node free from intervention still follows the distribution described in (2.5), though at the time some or all of its parent nodes may be under intervention. It is advised that all the nodes of a DAG should be topologically sorted, i.e. $i < j$ for all j that is a parent of i, so that this system (2.5) can be used as an efficient mechanism to sequentially generate samples following the desired distribution.

Having specified the distributions, we now present how we organize the sample data generated in an experiment into an $n \times p$ data matrix X. The rows of X are independent samples which shall be grouped into p blocks, so that the j-th group X^j contains all n_j rows where X_j is under intervention. We allow multiple interventions in each row, so these p groups may overlap. When a mixture of observational data and intervention data is presented, we will allow an additional $(p+1)$-th group X^{p+1} for the observational samples; in other words, this group consists of the rows where none of the nodes is put under the intervention distribution.

Let $\mathcal{O}_j$ be the collection of row indices where X_j is not under intervention, and $n_{-j} = |\mathcal{O}_j| = n - n_j$. By plugging in the density of Gaussian distributions defined in (2.5), we get the negative log-likelihood function of B and Ω as

$$L(B, \Omega | X) = \sum_{j=1}^{p} \left[\frac{n_{-j}}{2} \log(\omega_j^2) + \frac{1}{2\omega_j^2} \left\| X_{[\mathcal{O}_j, j]} - X_{[\mathcal{O}_j, -j]} B_{[-j, j]} \right\|^2 \right], \tag{2.6}$$

where $M_{[I_a, I_b]}$ denotes a submatrix of M by only selecting rows in the set I_a and columns in the set I_b.

Since we are estimating a sparse DAG, we would like to include penalty terms along with the negative log-likelihood into the score function. We impose a penalty function $p_\lambda(\cdot)$ on each β_{ij}, so as to discourage estimates with too many edges, thus avoiding overfitting and ensuring sparsity. λ will be the tuning parameter and we will introduce additional shape parameters for more flexibility.

To summarize, so far we have successfully translated our causal Gaussian network structure learning problem into seeking a solution the following minimizing problem:

$$\begin{aligned} \underset{B,\Omega}{\text{minimize}} \quad & L(B,\Omega|X) + \sum_{i,j} \alpha_j p_\lambda(|\beta_{ij}|) \\ \text{subject to} \quad & \omega_1^2, \ldots, \omega_p^2 > 0,\ \mathcal{G}_B \text{ is an DAG.} \end{aligned} \tag{2.7}$$

Here $\alpha_j = n_{-j}/n$ is a weight added to the penalty function to reflect the fact that β_{ij} only appears n_{-j} times out of total n samples.

By minimizing this score function, we will get a sparse estimate of the true underlying Bayesian network. In the next chapter we will provide details as well as new features in the minimizing algorithm as an supplement to Aragam and Zhou's work [AZ15].

CHAPTER 3

Algorithms

3.1 Non-convex optimization

It is easy to check that the negative log-likelihood function (2.6) is non-convex, which makes our optimization work harder than conventional convex optimization. However, we can overcome this by a reparameterization:

$$\begin{aligned} &\rho_j = 1/\omega_j, \quad \phi_{ij} = \beta_{ij}/\omega_j, \quad i, j = 1, \ldots, p. \\ &R = (\rho_1^2, \ldots, \rho_p^2), \quad \Phi = (\phi_{ij})_{p\times p}. \end{aligned} \tag{3.1}$$

Under the new reparameterization we can rewrite the negative log-likelihood function as

$$L(\Phi, R|X) = \sum_{j=1}^{n} \Big[- n_{-j} \log(\rho_j) + \frac{1}{2} \big\| \rho_j X_{[\mathcal{O}_j, j]} - X_{[\mathcal{O}_j, -j]} B_{[-j,j]} \big\|^2 \Big]. \tag{3.2}$$

The new loss function is now convex, and Φ induces a graph of the same structure as B does, so the constraint on $\mathcal{G}_B$ being acyclic is exactly translated to $\mathcal{G}_\Phi$ being acyclic. It should be clear that this DAG constraint is still non-convex.

In our work we use the minimax concave penalty (MCP) [Zha10] as the penalty on all ϕ_{ij}. It is defined for $t \geq 0$ as

$$p_\lambda(t;\gamma) := \lambda\Big(t - \frac{t^2}{2\lambda\gamma}\Big) 1(t < \lambda\gamma) + \frac{\lambda^2\gamma}{2} 1(t \geq \lambda\gamma) \tag{3.3}$$

It can be checked that $\alpha p_\lambda(t;\gamma) = p_{\alpha\lambda}(t; \frac{1}{\alpha}\gamma)$. This is useful when rescaling the penalty terms and will allow us to absorb the weights into the penalty terms to get a more unified formula for minimizers, as we will see in (3.11) in the next section.

3.2 Coordinate descent

The non-convex constraint on $\mathcal{G}_\Phi$ being acyclic has made it impossible to minimize the objective function over the entire parameter space of all ϕ_{ij}'s simultaneously since no global minimum point is garunteed. Instead we will use coordinate descent; that is, at each time we only minimize the objective function over one variable while holding the others fixed, and then we cycle through all variables until convergence.

Moreover, the acyclicity constraint inspires us to minimize over the pair $\{\phi_{ij}, \phi_{ji}\}$ simultaneously, as they cannot be both be non-zero at any time. Therefore, to ensure acyclicity, when minimizing, we first check whether adding the edge $i \to j$ induces a cycle; if so, we set $\phi_{ij} = 0$ and minimize over ϕ_{ji}. Similarly, if adding $j \to i$ induces a cycle, we just set $\phi_{ji} = 0$ and minimize over ϕ_{ij}. If neither of them induces a cycle, we minimize over them separately and chooses whichever achieves a smaller value in the score function.

For each single variable, the contributing terms in the score function are:

$$\begin{aligned} Q_1(\phi_{ij}) &= \frac{1}{2}\big\|\rho_j X_{[\mathcal{O}_j,j]} - X_{[\mathcal{O}_j,-j]} B_{[-j,j]}\big\|^2 + \alpha_j p_\lambda(|\phi_{ij}|;\gamma), \\ Q_2(\rho_j) &= -n_{-j} \log \rho_j + \frac{1}{2}\big\|\rho_j X_{[\mathcal{O}_j,j]} - X_{[\mathcal{O}_j,-j]} B_{[-j,j]}\big\|^2 \end{aligned} \tag{3.4}$$

In these functions all other variables other than ϕ_{ij} or ρ_j are considered fixed. Without loss of generality, we may assume the columns in the partial data matrix $X_{[\mathcal{O}_j,\cdot]}$ are normalized to have unit length and center at 0, since such a rescaling will not change the minimizer.

To find the minimizer for Q_1, we need to rewrite Q_1 :

$$Q_1(\phi_{ij}) = \frac{1}{2}\sum_{h\in\mathcal{O}_j}\Big(\rho_j x_{hj} - \sum_{k\neq i}\phi_{kj}x_{hk} - \phi_{ij}x_{hi}\Big)^2 + \alpha_j p_\lambda(|\phi_{ij}|;\gamma) \tag{3.5}$$

$$= \frac{1}{2}\sum_{h\in\mathcal{O}_j} x_{hi}^2\Big(\frac{1}{x_{hi}} r_{ij}^{(h)} - \phi_{ij}\Big)^2 + \alpha_j p_\lambda(|\phi_{ij}|;\gamma) \tag{3.6}$$

$$= \frac{1}{2}\Big(\sum_{h\in\mathcal{O}_j}(r_{ij}^{(h)})^2 - 2\phi_{ij}\sum_{h\in\mathcal{O}_j} x_{hi} r_{ij}^{(h)} + \phi_{ij}^2\Big) + \alpha_j p_\lambda(|\phi_{ij}|;\gamma) \tag{3.7}$$

$$= \frac{1}{2}\Big(\phi_{ij}^2 - \sum_{h\in\mathcal{O}_j} x_{hi} r_{ij}^{(h)}\Big)^2 + p_{\alpha_j\lambda}(\phi_{ij};\gamma/\alpha_j) + \text{const.} \tag{3.8}$$

Here $r_{ij}^{(h)} := \rho_j x_{hj} - \sum_{k\neq i}\phi_{kj}x_{hk}$. From (3.6) to (3.7) we expand the squares and make use of the assumption that $\sum_{h\in\mathcal{O}_j} x_{hi}^2 = 1$ to simplify the expressions.

The constant term on the last line does not involve ϕ_{ij} so it can be dropped in the minimization process. This allows us to find the minimizer by solving a minimizing problem in a more general form:

$$\arg\min_{\beta} Q^1(\beta) = \frac{1}{2}(\beta - \tilde{\beta})^2 + p_\lambda(|\beta|) \tag{3.9}$$

When we choose MCP as the penalty term, the solution to the minimizing problem above is given by the following cases depending on γ: [Zha10]

$$\text{if } \gamma > 1, \quad S_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \le \lambda, \\ \operatorname{sgn}(\tilde{\beta})\left(\frac{|\tilde{\beta}|-\lambda}{1-1/\gamma}\right), & \lambda < |\tilde{\beta}| \le \lambda\gamma, \\ \tilde{\beta}, & |\tilde{\beta}| > \lambda\gamma, \end{cases} \tag{3.10}$$

$$\text{if } 0 < \gamma \le 1, \quad S_\gamma(\tilde{\beta}, \lambda) = \begin{cases} 0, & |\tilde{\beta}| \le \lambda, \\ \tilde{\beta}, & |\tilde{\beta}| > \lambda, \end{cases}$$

Now back to (3.8), clearly the minimizer will be

$$\phi_{ij}^* = \arg\min Q_1(\phi_{ij}) = S_{\gamma/\alpha_j}\Big(\sum_{h\in\mathcal{O}_j} x_{hi} r_{ij}^{(h)}, \alpha_j\lambda\Big). \tag{3.11}$$

On the other hand, minimizing $Q_2(\rho_j)$ is much more direct as we do not have to go into cases:

$$\rho_j^* = \arg\min Q_2(\rho_j) = \frac{c + \sqrt{c^2 + 4n_{-j}}}{2}, \quad \text{where } c = \sum_{k\neq j} \phi_{kj} \sum_{h\in\mathcal{O}_j} x_{hk} x_{hj}. \tag{3.12}$$

3.3 Algorithm

Below is a summary of the entire algorithm.

It should be pointed out that coordinate descent is heavily affected by the order of nodes. Edges between last few nodes in the graph may be dropped simply due to acyclicity induced by edges estimated earlier in this cycle, even though they may be the true minimizer. It is suggested that we either permute the columns of the data matrix before running the algorithm, or permute the nodes each time we cycle through all the pairs when running the algorithm.

Algorithm 1: CCDri Algorithm for mixed observation and intervention data

```
Input: Data matrix $X$ of size $n \times p$; a list of $n$ vectors where each vector contains
       labels indicating the nodes under intervention in its corresponding row in $X$;
       initial estimates $(\Phi^{(0)}, R^{(0)})$; penalty parameters $(\lambda, \gamma)$; error tolerance $\varepsilon > 0$;
       maximum number of iterations $M$.
  1. Normalize the columns of $X_{[\mathcal{O}_j,\cdot]}$ $(j = 1, \ldots, p)$ to unit length for each $j$ separately, so
     a normalization for $X_{[\mathcal{O}_j,\cdot]}$ does not affect the normalization for $X_{[\mathcal{O}_k,\cdot]}$ for $k \neq j$.
  2. Cycle through $\rho_j$ for $j = 1, \ldots, p$, minimizing $Q_2$ over each $\rho_j$ at a time.
  3. Cycle through the $p(p-1)/2$ pairs $\{\phi_{ij}, \phi_{ji}\}$ for $j, k = 1, \ldots, p$ and $k < j$, minimizing
     $Q_1$ over each pair according to the following cases:
       (a) If $\phi_{ij}$ (or $\phi_{ji}$) must be 0 due to acyclicity, then we minimize $Q_1$ over the other
           one in the pair, and update the pair as $\{0, \phi^*_{ji}\}$ (or $\{\phi^*_{ij}, 0\}$).
       (b) If there is no acyclicity constraint, then we try to find a minimizer to
           $Q_1(\phi_{ij}) + Q_1(\phi_{ji})$ while setting $\phi_{ij}$ and $\phi_{ji}$ to 0 respectively, and choose the
           better minimizer of the two as the update for the pair.
  4. Repeat Step 2 and Step 3, until either the difference between two updates
     $\max_{i,j} |\phi^{(l-1)}_{ij} - \phi^{(l)}_{ij}|$ is less than $\varepsilon$, or iteration times exceeds $M$.
  5. Convert the final result $(\hat{\Phi}, \hat{R})$ back to the original parameters $(\hat{B}, \hat{\Omega})$ as output
     estimates.
```

3.4 Solution path

When estimating the Bayesian network from samples, we feed the algorithm with a series of λ to generate a solution path, from which we will choose the best one based on criteria involving some performance metrics.

The main factor we evaluated is the Structure Hamming Distance (SHD) [TBA06] between each estimate and the true graph. It is defined as the minimal number of the following three types of operators required to make the two DAGs match: add, delete, and reverse the direction of an directed edge.

We choose the best estimate to be the one in the solution path with smallest SHD from the true graph; in the event of a tie, the one with more estimated edges is preferred, as we do not want an underfitting model even though we know the true Bayesian network should be sparse.

CHAPTER 4

Simulation results

4.1 Performance in small networks

In the following simulations, we generate samples with interventions, and apply this algorithm (CCDri), the original algorithm without considering intervention effects (CCDr), along with CCDAG algorithm from Fu and Zhou (2013). For the last one, the best estimate is selected from the solution path using the same criteria as our algorithm. The number of nodes in the graph ranges from $p \in \{10, 20, 50, 100\}$, and the number of expected edges in the graph ranges from $n \in \{0.5p, p, 2p\}$ for each p. For each pair of n and p, we run simulations twice, where each node will get intervention twice and 5 times respectively.

The performance will be based on the following three factors: SHD; truth positive rate (TPR), which is defined as the percentage of correctly estimated edges in the true graph; and false discover rate (FDR), which is defined as the percentage of incorrectly estimated edges (estimated in the reverse direction, or not existing in the true graph) in the estimated graph. We will report each factor averaged over simulations repeated 50 times for each setting.

			Intervention times = 2			Intervention times = 5		
p	n	Algorithm	TPR	FDR	SHD	TPR	FDR	SHD
10	5	CCDri	0.4676	0.3296	3.46	0.7963	0.2008	1.542
		CCDr	0.3459	0.4339	3.98	0.4686	0.4779	3.208
		CCDAG	0.3310	0.2378	4.20	0.8094	0.1615	1.583
	10	CCDri	0.3959	0.3923	6.98	0.6993	0.2586	4.00
		CCDr	0.2832	0.4854	7.96	0.4175	0.5097	6.46

Continued on next page

Table 4.1 – *Continued from previous page*

			Intervention times = 2			Intervention times = 5		
p	n	Algorithm	TPR	FDR	SHD	TPR	FDR	SHD
10	10	CCDAG	0.4365	0.3620	7.44	0.7730	0.2406	3.66
	20	CCDri	0.3085	0.4710	16.48	0.4708	0.3952	13.20
		CCDr	0.2180	0.6018	18.32	0.2694	0.5736	16.68
		CCDAG	0.5821	0.3718	13.76	0.7740	0.2604	8.18
20	10	CCDri	0.5646	0.3169	4.64	0.7725	0.2059	2.94
		CCDr	0.3394	0.5625	6.74	0.5540	0.4304	5.12
		CCDAG	0.5213	0.3440	5.72	0.8287	0.1865	2.72
	20	CCDri	0.5546	0.3429	11.38	0.7174	0.2337	7.16
		CCDr	0.3446	0.5285	15.16	0.5166	0.4401	11.72
		CCDAG	0.4902	0.3407	13.42	0.7984	0.1920	6.16
	40	CCDri	0.2915	0.4297	32.76	0.4206	0.4263	29.78
		CCDr	0.2160	0.5214	35.24	0.2594	0.5305	34.54
		CCDAG	0.4220	0.4159	32.22	0.7016	0.3177	20.82
50	25	CCDri	0.6689	0.3069	8.82	0.8411	0.1570	4.46
		CCDr	0.5114	0.4550	12.82	0.5532	0.4390	11.88
		CCDAG	0.6376	0.3120	9.86	0.8306	0.1645	4.84
	50	CCDri	0.6510	0.2902	20.48	0.7793	0.2096	13.42
		CCDr	0.5197	0.4357	28.32	0.5905	0.3960	24.36
		CCDAG	0.6098	0.3006	23.96	0.8112	0.2078	13.68
	100	CCDri	0.4533	0.4110	68.82	0.3349	0.5190	79.66
		CCDr	0.3349	0.5190	79.66	0.4038	0.4882	74.28
		CCDAG	0.4790	0.4291	77.86	0.6888	0.3142	51.20
100	50	CCDri	0.7407	0.2597	13.92	0.8446	0.1567	8.42
		CCDr	0.5723	0.4272	22.64	0.5758	0.4232	22.84

Continued on next page

Table 4.1 – *Continued from previous page*

			Intervention times = 2			Intervention times = 5		
p	n	Algorithm	TPR	FDR	SHD	TPR	FDR	SHD
100	50	CCDAG	0.7127	0.2847	16.14	0.7931	0.2165	11.80
	100	CCDri	0.7274	0.2527	31.84	0.8322	0.1618	18.76
		CCDr	0.5798	0.4022	50.26	0.5910	0.3926	45.68
		CCDAG	0.6989	0.3106	40.66	0.7919	0.2257	25.60
	200	CCDri	0.5922	0.3426	107.08	0.6692	0.2812	90.14
		CCDr	0.4858	0.4401	132.60	0.5122	0.4220	127.52
		CCDAG	0.5470	0.3552	127.00	0.7432	0.2908	85.36

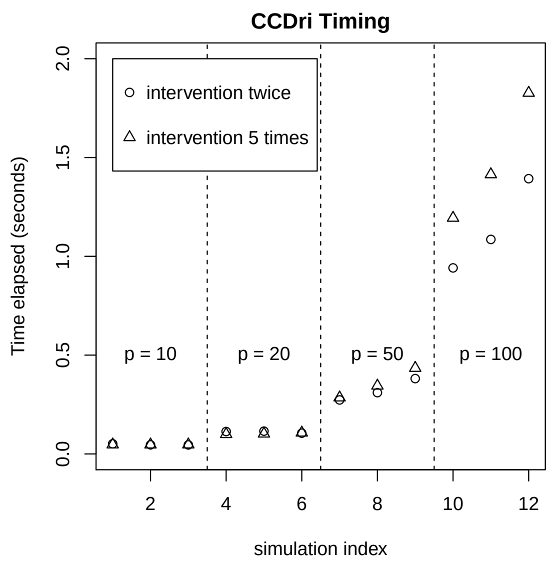
Table 4.1: Performance in small networks compared with other methods

Table 4.1 summarizes the result of the simulations. Overall CCDri outperforms CCDr, which shows that this new algorithm is indeed effective in estimating Bayesian network by taking advantage of the intervention introduced to the data. From the table we also see that our algorithm works better when the graph is sparse (i.e. when n/p^2 is small).

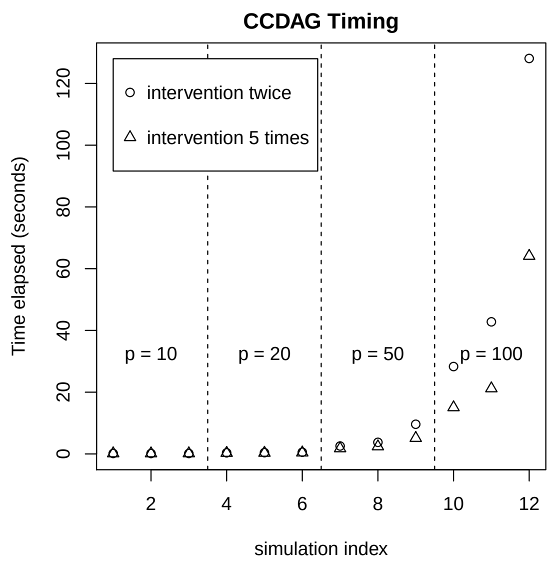
We may observe that for fixed p, when n grows, the accuracy of all algorithms drops, as TPR decreases and FDR increases. This is due to the fact the increasing number of edges in a graph means a more complex acyclic constraint on optimization. When doing coordinate descent in the last few pairs, edges are more likely to be chosen because they do not induce any cycle in the graph rather than because they minimize the score function better their reversed edges.

As for comparing CCDri with CCDAG, we can see that when each node gets less intervention (twice), the performance of CCDri and CCDAG are close; when each node gets more intervention (5 times), CCDAG slightly outperforms, at the cost of taking much longer time. This will be shown in the figure 4.1 below. It can be seen that CCDAG becomes much slower when the Bayesian network grows bigger, while accuracy does not improve very significantly.

Lastly we will take one particular setting $n = p = 100$ and investigate the effect or using



(a) Average time for CCDri

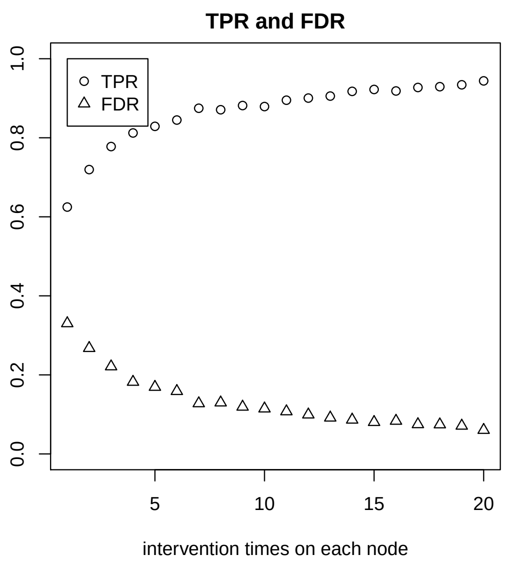


(b) Average time for CCDAG

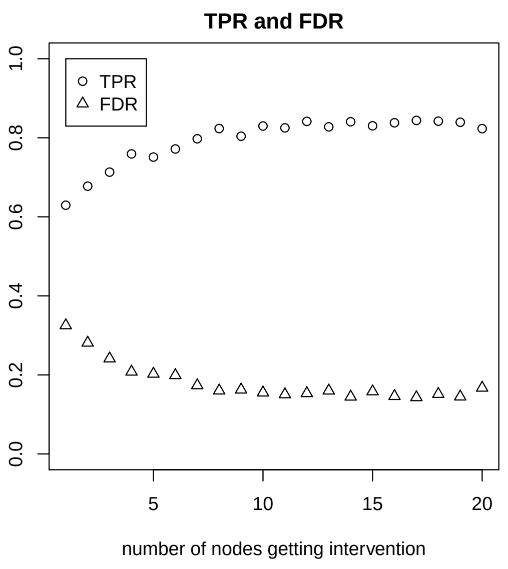
Figure 4.1: Comparing time used by CCDri and CCDAG

more intervention data. First we add intervention data step by step to see verify that our algorithm gives consistent estimators. As the figure on the left in Figure 4.2 shows, this is indeed the case; the average timing increases as well, steadily from 1.022 seconds to 3.676 seconds. In practice we may put intervention on each node 5 to 10 times in order to achieve high accuracy at an affordable cost.

Another way to use more intervention data is to put intervention on multiple nodes simultaneously in each sample. In this experiment we increase the number of nodes getting intervention from 1 to 20 in each step. As the figure on the right in Figure 4.2 shows, this is a working method as well, though not as effective as using more samples; however, average timing for this method does not increase too much, merely from 0.946 seconds to 1.076 seconds. This method can be used when we can ensure that the intervention we put on different nodes simultaneously are independent and do not interact with each other.



(a) TPR and FDR as more intervention added



(b) Average time for CCDAG

Figure 4.2: The effect of using more intervention

4.2 Performance in larger networks

We also run simulations for graphs with $n > 100$ nodes. The number of nodes in the graph ranges from $p \in \{200, 500\}$, and the number of expected edges in the graph ranges from $n \in \{0.5p, p, 2p\}$ for each p. CCDAG is dropped in these settings, because it takes much longer time only to give estimates at similar accuracy as our algorithm does, as we have seen in the pervious simulations. Instead, we compare our algorithm with MMHC from `bnlearn` package. We also considered PC algorithm form `pcalg` package in the preparation of the simulations. However, it is not included because the estimated graph of PC algorithm contains edges with both directions, so it is not a DAG, and thus not suitable to calculate SHD as well as other performance metrics.

Again, the results show that CCDri outperforms CCDr. However, in these simulations the difference of time used by CCDri and CCDr becomes more significant, even in the cases where each node get intervention only twice. This is primarily due to the amount of extra computation procedures introduced by intervention. It is also surprising to see that, while MMHC performs poorly in sparse Bayesian networks as its FDR is unusually high, it performs even better than CCDri when $n = 2p$ so the network is not too sparse. This suggests that MMHC might tend to overfit.

			Intervention times = 2			Intervention times = 5		
p	n	Algorithm	TPR	FDR	SHD	TPR	FDR	SHD
200	100	CCDri	0.7560	0.2486	25.84	0.8423	0.1563	16.22
		CCDr	0.5784	0.4200	44.52	0.5760	0.4197	44.04
		CCDAG	0.6776	0.7179	171.96	0.6692	0.6750	138.34
	200	CCDri	0.7538	0.2355	54.96	0.8311	0.1645	38.56
		CCDr	0.5918	0.3897	92.50	0.6072	0.3761	88.86
		CCDAG	0.7749	0.4087	108.46	0.7741	0.3829	97.00
	400	CCDri	0.6796	0.2859	170.82	0.7620	0.2180	131.50
		CCDr	0.5732	0.3720	222.18	0.5920	0.3515	211.1
		CCDAG	0.6949	0.1955	141.90	0.7702	0.1622	108.58
500	250	CCDri	0.7766	0.2165	60.09	0.8254	0.1553	37.28
		CCDr	0.5585	0.4348	119.5	0.5811	0.3902	78.41
		CCDAG	0.6546	0.7839	627.5	0.7544	0.7007	560.3
	500	CCDri	0.7672	0.2328	132.0	0.8039	0.1674	96.15
		CCDr	0.6210	0.3756	220.1	0.6808	0.3064	204.5
		CCDAG	0.7665	0.5074	396.5	0.7974	0.4835	340.6
	1000	CCDri	0.7365	0.2269	336.8	0.8128	0.1633	274.3
		CCDr	0.6043	0.3101	488.2	0.6803	0.2417	390.1
		CCDAG	0.8661	0.1493	203.4	0.8909	0.1221	175.2

Table 4.2: Performance in large networks compared with other methods

CHAPTER 5

Application on big networks

We have simulation results in the previous section showing that when the Bayesian network is too big, global intervention on all nodes may require much longer time for running algorithm. Therefore, we would want to avoid putting intervention directly on the entire causal Bayesian network.

Instead, we propose a two-stage approach which allow us to put a smaller amount of intervention on a group of local nodes to focus on estimating the causal structure among these nodes. The main idea is to break the big network into smaller components where we may afford more intervention experiments, so as to get more detailed estimates on local structures.

In the first stage, we will use observational data only to get a rough estimate of the big network. It does not matter much if the directions of edges are reversed. Instead, we would rather focus on the estimated structure of the network, to see if it can be decomposed into several smaller subgraphs. Characteristic structures such as $1 \to 2 \to 3$ and $1 \to 2 \leftarrow 3$ can also be identified and to be investigated later as well.

In the second stage, we still generate samples based on the whole true graph, but at the same time we are adding intervention to the nodes in a subgraph that we want to learn better about the causal relationships inside. Estimates will be based on both observational data from first stage and experiment data just generated. When comparing SHD between the solution path and the true graph, we will focus on the subgraph rather then the entire graph. If we want an estimate for the entire graph, we may repeat this process for each subgraph, and then combine estimates for each one into a larger graph.

Besides estimating the components, we may also investigate on several characteristic structures in a Bayesian network. Suppose we have identified edges among nodes 1, 2, 3 as 1—2—3 in the first stage, but we are unsure of the directions yet. Then we can just put intervention on these three nodes to distinguish the true causal relationships from its equivalence classes, whether $1 \to 2 \to 3$, $1 \to 2 \leftarrow 3$ or $1 \leftarrow 2 \to 3$.

The two-stage approach is effective when the true graph can indeed be partitioned into several subgraphs. If this is not the case, then we may have to force a partition of the graph; edges connecting these partitions in the true graphs are then lost and will not be estimated in the second stage. We will need additional experiments if we want to detect and recover such structures.

In simulation studies, by merging estimates of each component in the second stage, we will indeed get a better estimate of the network than first stage alone. However, improvement is not very significant; true positive rate (TPR) rises from around 50% in the first stage, to just around 60% in the second stage; on the other hand, the drop in false discovery rate (FDR) is not too much as well. Another disadvantage of this approach would be that even if we just focus on a local group of nodes in the second stage, we still generate samples from the whole network, which may take more time than expected. A summary of simulation results can be found from Table 5.1.

			First stage			Second stage		
p	n	partitions	TPR	FDR	SHD	TPR	FDR	SHD
50	50	5	0.4606	0.2814	22.84	0.5499	0.2605	17.13
100	100	5	0.5849	0.2986	35.12	0.6523	0.2577	31.34
100	100	10	0.5096	0.3311	41.81	0.6080	0.3115	39.22
200	200	5	0.6834	0.2749	62.29	0.7879	0.2301	56.18
200	200	10	0.6202	0.3122	67.05	0.7014	0.3004	62.50

Table 5.1: Performance of the two-stage approach

The success of this two-stage approach requires an rough estimate of the network in the first stage that a) is sparse so we can decompose the graph into multiple partitions, and b) contains most of the edges, so few nodes are singled out and edges connecting them to larger components may be identified in the second stage. This requires a tuning of the parameters. Another factor that we need to consider is that, the amount of intervention we need to flip back the edges which are reversely estimated from purely observational data in the first stage. As we can see, this two-stage approach is still under development and is open to improvements in multiple directions.

CHAPTER 6

Discussions

In this article we have made improvements to Aragam and Zhou's CCDr algorithm to estimate sparse Gaussian Bayesian network, by taking advantage of intervention data to distinguish the true causal structure from its equivalence classes. The new method has been shown in simulation experiments to be effective as it has a significantly higher accuracy than the original method, in different Bayesian network models of different sizes and sparsity. This algorithm will be included in an `R` package, which is currently still under construction, for causal Bayesian network learning.

With the introduction of intervention data along with observational data, it has indeed improved accuracy; meanwhile, it also requires more resource to meet the computational needs. To solve this, we have proposed a two-stage approach to use intervention locally to get better local estimates, and then combine them together for a global estimate. This still needs to be polished and will be one of our future research directions. Another solution is to identify the minimal set of nodes that we need to put intervention on in order to completely identify the true Bayesian network model; this is less statistically related, yet it is still one of our interests. We are also looking for ways to improve the efficiency of our algorithm by fitting to the input sample data structure more closely.

References

[AZ15] Bryon Aragam and Qing Zhou. "Concave penalized estimation of sparse Gaussian Bayesian networks." *Journal of Machine Learning Research*, **16**:2273–2328, 2015.

[CAL08] X. W. Chen, G. Anantha, and X. Lin. "Improving Bayesian Network Structure Learning with Mutual Information-Based Node Ordering in the K2 Algorithm." *IEEE Transactions on Knowledge and Data Engineering*, **20**(5):628–640, May 2008.

[CGH94] David M Chickering, Dan Geiger, David Heckerman, et al. "Learning Bayesian networks is NP-hard." Technical report, Citeseer, 1994.

[FZ13] Fei Fu and Qing Zhou. "Learning sparse causal Gaussian networks with experimental intervention: regularization and coordinate descent." *Journal of the American Statistical Association*, **108**(501):288–300, 2013.

[GB13] Sara Van de Geer, Peter Bühlmann, et al. "ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs." *The Annals of Statistics*, **41**(2):536–567, 2013.

[HB15] Alain Hauser and Peter Bühlmann. "Jointly interventional and observational data: estimation of interventional Markov equivalence classes of directed acyclic graphs." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **77**(1):291–318, 2015.

[HG08] Yang-Bo He and Zhi Geng. "Active learning of causal networks with intervention experiments and optimal designs." *Journal of Machine Learning Research*, **9**(Nov):2523–2547, 2008.

[HGC95] 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.

[KB07] Markus Kalisch and Peter Bühlmann. "Estimating high-dimensional directed acyclic graphs with the PC-algorithm." *Journal of Machine Learning Research*, **8**(Mar):613–636, 2007.

[LB94] Wai Lam and Fahiem Bacchus. "Learning Bayesian belief networks: An approach based on the MDL principle." *Computational intelligence*, **10**(3):269–293, 1994.

[MB06] Nicolai Meinshausen and Peter Bühlmann. "High-dimensional graphs and variable selection with the lasso." *The annals of statistics*, pp. 1436–1462, 2006.

[SGS00] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT press, 2nd edition, 2000.

[TBA06] 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.

[Zha10] Cun-Hui Zhang. "Nearly unbiased variable selection under minimax concave penalty." *The Annals of statistics*, pp. 894–942, 2010.