Learning DAGs with Continuous Optimization

Xun Zheng

August 2020
CMU-ML-20-110

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Eric Xing, Co-Chair
Pradeep Ravikumar, Co-Chair
Clark Glymour
Kun Zhang
Raquel Urtasun (University of Toronto)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under award numbers IIS1218282, IIS1563887 and IIS1617583; the National Institutes of Health award numbers P30DA035778 and R01GM117594; the Defense Advanced Research Projects Agency award numbers FA8721-05-C-0003 and FA8702-15-D-0002; the Air Force Research Laboratory award number FA87501720152; a CURE grant from the Pennsylvania Department of Health (Big Data for Better Health); a grant from the University of Pittsburgh Medical Center; and a fellowship from the Siebel Scholars Program.

Keywords: bayesian network, structure learning, causal discovery

To my grandparents.

Abstract

Learning the structure of directed acyclic graphs (DAGs, also known as Bayesian networks) from data is an important and classical problem in machine learning, with prominent applications in causal inference, fairness, interpretability, and biology, etc. This is a challenging problem since the search space of DAGs is combinatorial and scales superexponentially with the number of nodes. Existing approaches often rely on various local heuristics for enforcing the acyclicity constraint. By contrast, structure learning for undirected graphical models (e.g. Gaussian MRF) is recognized as a tractable optimization problem nowadays, and achieved huge success in various practical domains such as bioinformatics.

In this thesis, we take a first step towards bridging this gap between directed and undirected graphical models. We begin by introducing a fundamentally different strategy for Bayesian network structure learning: We formulate the problem as a purely *continuous* optimization program over real matrices that avoids the combinatorial constraint entirely. This is achieved by a novel characterization of acyclicity that is not only smooth but also exact. The resulting problem can be efficiently solved by standard numerical algorithms, without imposing any structural assumptions on the graph such as bounded treewidth or in-degree.

We then study the generalization of the above continuous algorithm to learning *nonparametric* DAGs. We extend the algebraic characterization of acyclicity to nonparametric structural equation model (SEM) by leveraging nonparametric sparsity based on partial derivatives, resulting in a continuous optimization problem that can be applied to a variety of nonparametric and semiparametric models including GLMs, additive noise models, and index models as special cases.

Lastly, we introduce a unified view of score-based and ICA-based methods based on the proposed continuous optimization framework. In particular, we show that the popular ICA-based methods that exploits non-Gaussianity of the independent noise distribution can be handled by the continuous optimization framework, which is conceptually clearer and easier to incorporate prior knowledge, and has the potential to be generalized to allow for models with hidden confounders and feedback loops.

Acknowledgments

Working towards a Ph.D. can be a tough journey. I was fortunate to have many people extending help throughout.

First and foremost, I would like to thank my two awesome advisors, Eric Xing and Pradeep Ravikumar, who have always been tremendously supportive and understanding. I am grateful for the exceptional freedom they allowed, as well as the guidance they provided when I was facing difficulties. I would also like to thank the committee members Clark Glymour, Kun Zhang, and Raquel Urtasun, for providing excellent feedback and encouraging me to pursue this direction.

Many thanks to the great people I met at CMU. The biggest one of course goes to Diane Stidle, who makes the department such a lovely place. Also thanks to Amy Protos and Sharon Cavlovich for all the support. Big shout-out to past and current members of SAILING and RAIL for constantly providing inspirations. It was a privilege to work with such a talented group of people with diverse interests. I learned a lot from them both academically and personally. Special thanks to former postdoc Yaoliang Yu and Bryon Aragam, who influenced me in many ways. This thesis could not have been written without their help. Also thanks to Chen Dan, a good friend and collaborator who is always delightful to talk to. I also appreciate Biwei Huang for introducing me to her group, and Joseph Ramsey for the hot coffee. Thanks to Yichong Xu for generously helping with the moving, and Mr. and Mrs. Hovy for the Italian food.

Thanks as well to my officemates for sparkling endless distraction and joy; to the "sam's blue gpu" group for existing; to the "morons" group for the funny random chats; to the Chinese and Korean friends for hunting for good Asian food. Special thanks to Jisu Kim for inviting me to the Wednesday dinner, and Seojin Bang for helping with the job search. Thanks to the entire 8th floor people for our random encounters in the corridor. As we are entering the 6th month of remote working, I increasingly miss every one of you. Hopefully, *for now we see only a reflection as in a mirror; then we shall see face to face.*

Lastly and most importantly, I would like to thank my family and Dr. Hong for the unconditional love and support. Thank you for being there throughout the whole time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Directed acyclic graphical models (DAGs, also known as Bayesian networks) are one of the foundational ideas in probabilistic models, with numerous applications in machine learning (Koller and Friedman, 2009), causal inference (Spirtes et al., 2000; Pearl, 2000), medicine (Heckerman et al., 1992), biology (Sachs et al., 2005), genetics (Zhang et al., 2013), fairness and accountability (Chiappa and Isaac, 2019), and finance (Sanford and Moosa, 2012), just to list a few. In addition to their undirected counterparts, DAG models offer a parsimonious, interpretable representation of a joint distribution that is useful in practice.

Bayesian networks are often designed to *encode* domain knowledge. In fact, many probabilistic expert systems are built using the language of Bayesian networks. One classic example is the ALARM network (Beinlich et al., 1989), where nodes represent 37 variables measured in an intensive care unit (ICU), such as breathing rate and blood pressure, and edges are designed by human experts to reflect direct causal relationship between variables. Even outside expert systems, many commonly used Bayesian networks in machine learning are about explicitly expressing conditional independence assumptions. For instance the hidden Markov model (HMM) is designed so that the hidden states only directly depend on the previous state and the observations are independent conditioned on the hidden states.

However, in many other cases, we are interested in *extracting* knowledge from the data, rather than encoding them. For instance, we would like to understand how different regions of brains interact with each other, by analyzing the patterns of fMRIs. In bioinformatics, estimating gene-regulatory networks from gene expression data is another important topic that requires extracting knowledge. In a machine learning language, given data drawn from a distribution induced by a DAG model, we are interested in recovering the underlying DAG from the observations. This is known as the *Bayesian network structure learning* (BNSL) problem.

Unfortunately, the BNSL problem is challenging since the search space of DAGs is combinatorial and scales super-exponentially with the number of nodes (Robinson, 1977), and it should not be a surprise that BNSL turns out to be NP-hard (Chickering, 1996; Chickering et al., 2004). Although for moderately sized problems one can still afford an exact search such as GOBNILP (Cussens, 2011; Cussens et al., 2017), many interesting real world problems are larger than what they can handle. For instance, the gene networks can have thousands of

variables, if not more, making the exact solvers impractical. The main difficulty comes from the acyclicity constraint, and many existing approaches rely on various local heuristics to satisfy this combinatorial constraint.

In this thesis, we propose a *continuous optimization* framework for score-based learning of Bayesian networks. We begin by introducing a fundamentally different strategy for Bayesian network structure learning: We formulate the structure learning problem as a purely *continuous* optimization problem, circumventing the combinatorial constraint entirely. This is achieved by a novel characterization of acyclicity that is not only smooth but also exact. The resulting problem can be efficiently solved by standard numerical algorithms, which also makes implementation effortless. We also study the generalization of the above continuous algorithm to learning *nonparametric* DAGs. We extend the algebraic characterization of acyclicity to nonparametric structural equation model (SEM) by leveraging nonparametric sparsity based on partial derivatives, resulting in a continuous optimization problem that can be applied to a variety of nonparametric and semiparametric models including GLMs, additive noise models, and index models as special cases. We explore the use of neural networks and orthogonal basis expansions to model nonlinearities for general nonparametric models. We further explore the special model class of linear non-Gaussian SEM, which is one of the basic cases the graph is uniquely identifiable from the observational data. We extend the continuous optimization algorithm to this model class to avoid the use of ICA in traditional algorithms. Overall, this work serves as a bridge between optimization and Bayesian networks literature, and we believe it can open up possibility for new directions.

1.1 Background

Consider a d-dimensional random vector $\mathbf{x} = (x_1, \dots, x_d) \sim P(\mathbf{x})$, whose distribution factorizes according to a Bayesian network structure G:

$$P(\mathbf{x}; G) = \prod_{j=1}^{d} P(x_j | \mathbf{x}_{\mathrm{pa}(j)}) \tag{1.1}$$

where $\mathrm{pa}(j)$ is the set of parent nodes of the j-th node in G. The conditional distributions can be equivalently expressed through the language of *structural equation model* (SEM):

$$x_j \sim P(x_j | \mathbf{x}_{\mathrm{pa}(j)}) \quad \Longleftrightarrow \quad x_j = f_j(\mathbf{x}_{\mathrm{pa}(j)}, z_j) \tag{1.2}$$

where f_j is a deterministic transformation and z_j is a random noise independent of $\mathbf{x}_{\mathrm{pa}(j)}$. In the most basic setting, when the transformation is linear, the noise is additive, the resulting set of equations defines a linear SEM:

$$x_j = \sum_{i \in \mathrm{pa}(j)} w_{ij} x_i + z_j, \quad z_j \sim p_j(z_j), \quad j = 1, \dots, d. \tag{1.3}$$

Let X be an $n \times d$ matrix that contains n realizations of the d-dimensional random vector $\mathbf{x}$. We are interested in the *Bayesian network structure learning* (BNSL) problem: Given observational data X, how can we recover the graph G?

1.2 Previous works

It is instructive to compare the historic development of structure learning in Bayesian networks with their undirected counterparts, Markov networks. Although there are a number of different approaches to graphical model structure learning, here we focus on the two major directions: *constraint-based* methods and *score-based* methods.

Constraint-based methods use repeated conditional independence tests to decide whether to include an edge in the graph. The idea is to construct a graph whose induced conditional independence set is compatible with what is detected from the data. The use of constraint-based method in Markov networks can be traced back to at least Dempster (1972) for Gaussian graphical models. The pioneers of constraint-based method in Bayesian network include the PC algorithm (Spirtes and Glymour, 1991; Kalisch and Bühlmann, 2007) and the MMPC algorithm (Tsamardinos et al., 2006). Although it is often computationally efficient, the multiple testing problem makes constraint-based methods sensitive to individual failures of tests. Moreover, the algorithm relies on the *faithfulness* assumption: the set of conditional independencies induced by the graph is the same as the conditional independencies induced by the distribution. This tend to be a strong assumption in practice as finite data may exhibit many spurious conditional independence that the underlying graph does not contain.

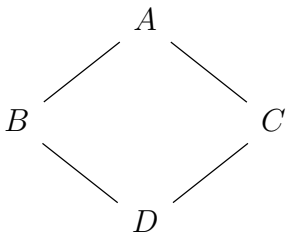
Score-based methods optimize a score function that measures the model fit of the candidate graph to the data. Early form of score-based method performs local search: at each step, a single edge that increases the score most is included in the graph, or conversely a single best edge is chosen to be removed until the score does not improve. In Markov networks, Pietra et al. (1997) proposed to use single-feature gain as a criterion for feature selection for structure learning. For Bayesian networks, greedy hill-climbing (Heckerman et al., 1995) and the later greedy equivalence search (GES) (Chickering, 2002; Ramsey, 2015; Ramsey et al., 2017) are prominent examples of this category. Popular score functions include BDe(u) (Heckerman et al., 1995), BGe (Kuipers et al., 2014), BIC (Chickering and Heckerman, 1997), and MDL (Bouckaert, 1993). One should not forget that for local search in Bayesian networks, edges and parent sets are added sequentially, while checking the DAG constraint. This is efficient as long as each node has only a few parents, but as the number of possible parents grows, local search rapidly becomes intractable. Furthermore, such strategies typically rely on severe structural assumptions such as bounded in-degree, bounded tree-width, or edge constraints.

In 2000s, a breakthrough took place in structure learning for Markov networks. It began by the observation that the problem of score-based learning for some classes of graphs can be written as a continuous (even convex) optimization programs, such as penalized log-determinant or least-squares (Meinshausen and Bühlmann, 2006; Friedman et al., 2008; Banerjee et al., 2008; Ravikumar et al., 2010, 2011). The significance of this realization is that one can now leverage the entire arsenal of convex and numerical optimization in solving the structure learning problem efficiently. The invention of these more efficient algorithms contributed to a huge success of undirected graphical models in various fields like bioinformatics. Unlike greedy local search, these methods optimize over the entire parameter space at once, hence they perform global search.

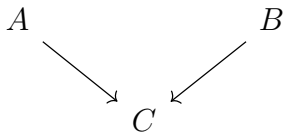
One natural question is whether this breakthrough in Markov networks can be applied to Bayesian networks. There are two immediate challenges:

1. We are dealing with directed graphs, whose adjacency matrices are asymmetric. It is known that symmetric matrices are more structured and possess nice properties such as having real eigenvalues. This is a challenge to the optimization.
2. We have to satisfy the acyclicity constraint, which is a combinatorial by nature. How can we optimize a smooth function over a combinatorial constraint using continuous optimization methods?

In this thesis, we would like to take the first step towards addressing these challenges. Table 1.1 summarizes the above review of the historical development, and locates this thesis in the literature.



(a) A diamond Markov network



(b) A v-structure

Figure 1.1: Examples of conditional independence relations that cannot be expressed by each other.

1.3 Why Bayesian networks?

As we will see shortly, the problem of Bayesian network structure learning is substantially harder than their undirected counterpart. What is the reward then, if there is any, given the increased difficulty? Why are we not satisfied by Markov networks alone?

First, Markov networks and Bayesian networks in general represent different set of conditional independence relations. For instance, there is no Bayesian network that can represent the same set of conditional independence as a diamond Markov network. Similarly, there is no Markov network that can represent the same conditional independence relations as a v-structure. See Figure 1.1 for an illustration. Only for chordal (triangular) graphs both Markov network and Bayesian network can have the equal representation power (Koller and Friedman, 2009). Therefore, Bayesian networks provides a different model class than Markov networks.

Second, with the help of additional assumptions one can interpret Bayesian networks as causal graphs. This gives one the opportunity to answer a much richer set of queries, such as interventional queries (e.g. distribution of X when intervening on Y) and counterfactual queries (e.g. What could have happened if this variable took another value for this individual?), in addition to the usual statistical queries based on correlation. Answering these causal

	Markov networks	Bayesian networks
constraint-based	Dempster (1972)	Spirtes and Glymour (1991)
score-based (combinatorial, local search)	Pietra et al. (1997)	Heckerman et al. (1995)
score-based (continuous, global search)	Meinshausen and Bühlmann (2006) Friedman et al. (2008) Banerjee et al. (2008) Ravikumar et al. (2010)	**This Thesis**

Table 1.1: A non-exhaustive history of structure learning.

queries has tremendous importance in e.g. policy making, medical diagnosis and treatment, and scientific discovery. Typical statistical methods such as regression-based variable selection will only recover the *Markov blanket* (the smallest set that can separate a variable from the rest), however this has no implication in answering the causal queries.

Of course, one should always pay attention to the assumptions when claiming something ambitious, such as causality. For instance, in the presence of unobserved confounders, many methods developed for fully observed scenarios will lead to incorrect answers. Therefore, it is crucial to be clear about the assumptions before applying to real problems.

For interested readers we refer to excellent textbooks such as Spirtes et al. (2000); Pearl (2000); Peters et al. (2017) for extensive review on causal inference and discovery.

1.4 Overview

The rest of the document is organized as follows (see also Figure 1.2):

- In Chapter 2, we start with the simplest setting: score-based learning of *linear* DAGs using continuous optimization. We show how to convert a combinatorial constraint on the graph into an smooth constraint on the adjacency matrix, while retaining their equivalence. This chapter is based on Zheng et al. (2018).
- In Chapter 3, we move beyond the linear case to extend the framework to general nonparametric DAGs. We show how to use flexible function classes such as neural networks and orthogonal basis expansions to detect nonlinear dependencies. This chapter is based on Zheng et al. (2020).
- In Chapter 4, we extend the differentiable causal discovery to ICA-based approaches. We show how ICA-based algorithms can be expressed as a single optimization problem instead of separate stages, and how this new formulation benefits the search result.

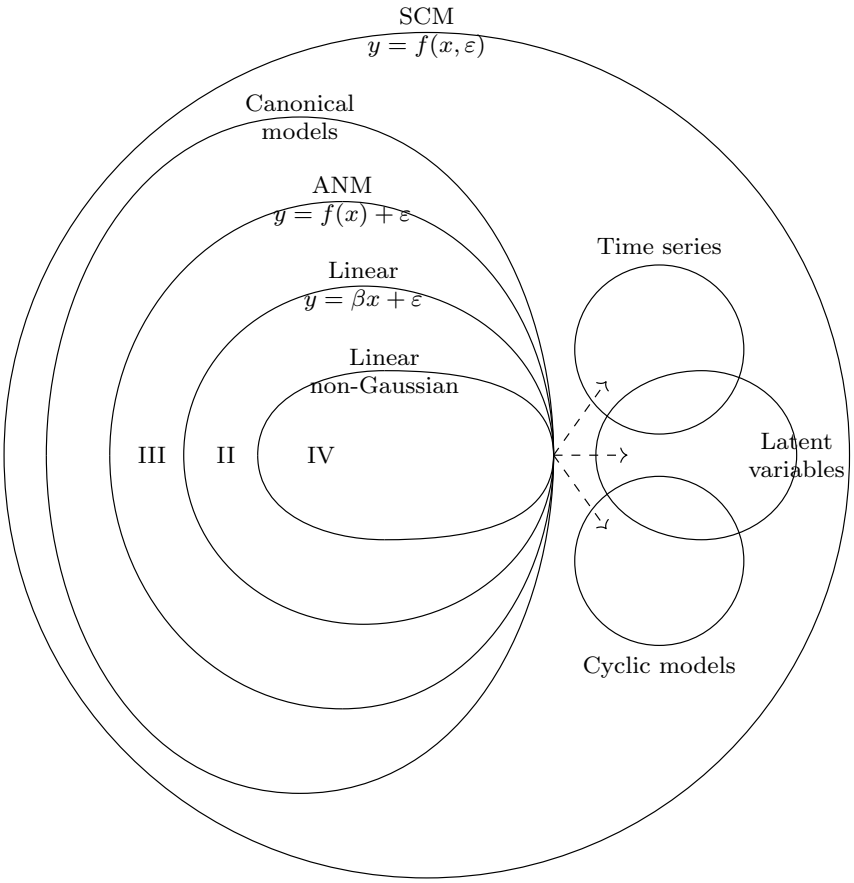


Figure 1.2: An overview of the model classes this thesis focuses on. For simplicity the bivariate case $x \to y$ is illustrated.

Chapter 2

Learning Linear DAGs with Continuous Optimization

Learning directed acyclic graphs (DAGs) from data is an NP-hard problem (Chickering, 1996; Chickering et al., 2004), owing mainly to the combinatorial acyclicity constraint that is difficult to enforce efficiently. At the same time, DAGs are popular models in practice, with applications in biology (Sachs et al., 2005), genetics (Zhang et al., 2013), machine learning (Koller and Friedman, 2009), and causal inference (Spirtes et al., 2000). For this reason, the development of new methods for learning DAGs remains a central challenge in machine learning and statistics.

In this work, we propose a new approach for score-based learning of DAGs by converting the traditional *combinatorial* optimization problem (left) into a *continuous* program (right):

$$\begin{array}{ll} \max\limits_{G} & \mathsf{score}(G; X) \\ \text{s.t.} & G \in \mathsf{DAGs} \end{array} \quad \Longleftrightarrow \quad \begin{array}{ll} \max\limits_{W} & \mathsf{score}(W; X) \\ \text{s.t.} & h(W) = 0 \end{array} \tag{2.1}$$

where G is the d-node graph and W is the corresponding $d \times d$ weighted adjacency matrix, the score function evaluates the model fit of a graph for a dataset X (see Section 2.2.1 for details), and our key technical device $h : \mathbb{R}^{d \times d} \to \mathbb{R}$ is a smooth function over real matrices, whose level set at zero exactly characterizes acyclic graphs. Although the two problems are equivalent, the continuous program on the right eliminates the need for specialized algorithms that are tailored to search over the combinatorial space of DAGs. Instead, we are able to leverage standard numerical algorithms for constrained problems, which makes implementation particularly easy, not requiring any knowledge about graphical models. This is similar in spirit to the situation for undirected graphical models, in which the formulation of a continuous log-det program (Banerjee et al., 2008) sparked a series of remarkable advances in structure learning for undirected graphs (Section 2.2.2). Unlike undirected models, which can be reduced to a convex program, however, the program (2.1) is *nonconvex*. Nonetheless, as we will show, even naive solutions to this program yield state-of-the-art results for learning DAGs.

2.1 Our contributions

The main thrust of this work is to re-formulate score-based learning of DAGs so that standard smooth optimization schemes such as L-BFGS (Nocedal and Wright, 2006) can be leveraged. To accomplish this, we make the following specific contributions:

- We explicitly construct a smooth function over $\mathbb{R}^{d\times d}$ with computable derivatives that encodes the acyclicity constraint. This allows us to replace the combinatorial constraint $G \in \mathsf{DAGs}$ in (2.4) with a smooth equality constraint.
- We develop an equality-constrained program for simultaneously estimating the structure and parameters of a sparse DAG from possibly high-dimensional data, and show how standard numerical solvers can be used to find stationary points.
- We demonstrate the effectiveness of the resulting method in empirical evaluations against existing state-of-the-arts.
- We compare our output to the exact global minimizer (Cussens, 2011), and show that our method attains scores that are comparable to the globally optimal score in practice, although our methods are only guaranteed to find stationary points.

Most interestingly, our approach is very simple and can be implemented in about 50 lines of Python code. As a result of its simplicity and effortlessness in its implementation, we call the resulting method NOTEARS: *Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning.* The implementation is publicly available at https://github.com/xunzheng/notears.

2.2 Background

The basic DAG learning problem is formulated as follows: Let $X \in \mathbb{R}^{n\times d}$ be a data matrix consisting of n i.i.d. observations of the random vector $\mathbf{x} = (x_1, \ldots, x_d)$ and let $\mathbb{G}^d$ denote the (discrete) space of directed acyclic graphs $G = (V, E)$ on d nodes. Given X, we seek to learn a DAG $G \in \mathbb{G}^d$ (also called a *Bayesian network*) for the joint distribution $\mathbb{P}(\mathbf{x})$ (Spirtes et al., 2000; Koller and Friedman, 2009). We model $\mathbf{x}$ via a structural equation model (SEM) defined by a weighted adjacency matrix $W \in \mathbb{R}^{d\times d}$. Thus, instead of operating on the discrete space $\mathbb{G}^d$, we will operate on $\mathbb{R}^{d\times d}$, the continuous space of $d \times d$ real matrices.

2.2.1 Score functions and SEM

Any $W \in \mathbb{R}^{d\times d}$ defines a graph on d nodes in the following way: Let $A(W) \in \{0, 1\}^{d\times d}$ be the binary matrix such that $[A(W)]_{ij} = 1 \iff w_{ij} \neq 0$ and zero otherwise; then $A(W)$ defines the adjacency matrix of a directed graph $G(W)$. In a slight abuse of notation, we will thus treat W as if it were a (weighted) graph. In addition to the graph $G(W)$, $W = [\,\mathbf{w}_1|\cdots|\mathbf{w}_d\,]$ defines a linear SEM by $x_j = \mathbf{w}_j^T\mathbf{x} + z_j$, where $\mathbf{x} = (x_1, \ldots, x_d)$ is a random vector and $\mathbf{z} = (z_1, \ldots, z_d)$ is a random noise vector. We do *not* assume that $\mathbf{z}$ is Gaussian. More generally, we can model x_j via a generalized linear model (GLM) $\mathbb{E}[x_j|\mathbf{x}_{\mathrm{pa}(j)}] = g(\mathbf{w}_j^T\mathbf{x})$,

where g is the link function. For example, if $x_j \in \{0, 1\}$, we can model the conditional distribution of x_j given its parents via logistic regression.

In this chapter, we focus on linear SEM and the least-squares (LS) loss $\ell(W; X) = \frac{1}{2n} \|X - XW\|_F^2$, although everything in the sequel applies to any smooth loss function ℓ defined over $\mathbb{R}^{d \times d}$. The statistical properties of the LS loss in scoring DAGs have been extensively studied: The minimizer of the LS loss provably recovers a true DAG with high probability on finite-samples and in high-dimensions ($n \ll d$), and hence is consistent for both Gaussian SEM (van de Geer and Bühlmann, 2013; Aragam et al., 2015) and non-Gaussian SEM (Loh and Bühlmann, 2014).[1] Note also that these results imply that the faithfulness assumption is not required in this set-up. Given this extensive previous work on statistical issues, our focus for now is entirely on the computational problem of finding an SEM that minimizes the LS loss.

This translation between graphs and SEM is central to our approach. Since we are interested in learning a *sparse* DAG, we add ℓ_1-regularization $\|W\|_1 = \|\mathrm{vec}(W)\|_1$ resulting in the regularized (negative) score function

$$F(W) = \ell(W; X) + \lambda \|W\|_1 = \frac{1}{2n} \|X - XW\|_F^2 + \lambda \|W\|_1 . \tag{2.2}$$

Thus we seek to solve

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & F(W) \\ \text{s.t.} \quad & G(W) \in \mathbb{G}^d. \end{aligned} \tag{2.3}$$

Unfortunately, although $F(W)$ is continuous, the DAG constraint $G(W) \in \mathbb{G}^d$ remains a challenge to enforce. In Section 2.3, we show how this discrete constraint can be replaced by a smooth equality constraint.

2.2.2 Previous work

Traditionally, score-based learning seeks to optimize a *discrete score* $Q : \mathbb{G} \to \mathbb{R}$ over the set of DAGs $\mathbb{G}$; note that this is distinct from our score $F(W)$ whose domain is $\mathbb{R}^{d \times d}$ instead of $\mathbb{G}^d$. This can be written as the following combinatorial optimization problem:

$$\begin{aligned} \max_{G} \quad & Q(G) \\ \text{s.t.} \quad & G \in \mathbb{G} \end{aligned} \tag{2.4}$$

Popular score functions include BDe(u) (Heckerman et al., 1995), BGe (Kuipers et al., 2014), BIC (Chickering and Heckerman, 1997), and MDL (Bouckaert, 1993). Unfortunately, (2.4) is NP-hard to solve Chickering (1996); Chickering et al. (2004) owing mainly to the nonconvex, combinatorial nature of the optimization problem. This is the main drawback of

[1] Due to nonconvexity, there may be more than one minimizer: These and other technical issues such as parameter identifiability are addressed in detail in the cited references.

existing approaches for solving (2.4): The acyclicity constraint is a combinatorial constraint with the number of acyclic structures increasing super-exponentially in d (Robinson, 1977). Notwithstanding, there are algorithms for solving (2.4) to global optimality for small problems (Ott and Miyano, 2003; Singh and Moore, 2005; Silander and Myllymäki, 2006; Xiang and Kim, 2013; Cussens, 2011; Cussens et al., 2017). There is also a wide literature on approximate algorithms based on order search (Teyssier and Koller, 2005; Schmidt et al., 2007; Scanagatta et al., 2015, 2016), greedy search (Heckerman et al., 1995; Chickering, 2002; Ramsey et al., 2017), and coordinate descent (Fu and Zhou, 2013; Aragam et al., 2015; Gu et al., 2018). By searching over the space of topological orderings, the former order-based methods trade-off the difficult problem of enforcing acyclicity with a search over $d!$ orderings, whereas the latter methods enforce acyclicity one edge at a time, explicitly checking for acyclicity violations each time an edge is added. Other approaches that avoid optimizing (2.4) directly include constraint-based methods (Spirtes and Glymour, 1991; Spirtes et al., 2000), hybrid methods (Tsamardinos et al., 2006; Gámez et al., 2011), and Bayesian methods (Ellis and Wong, 2008; Zhou, 2011; Niinimäki et al., 2016).

The intractable form of the program (2.4) has led to a host of heuristic methods, often borrowing tools from the optimization literature, but always resorting to clever heuristics to accelerate algorithms. Here we briefly discuss some of the pros and cons of existing methods. While not all methods suffer from *all* of the problems highlighted below, we are not aware of any methods that simultaneously avoid all of them.

Exact vs. approximate. Broadly speaking, there are two camps: *Approximate* algorithms and *exact* algorithms, the latter of which are guaranteed to return a globally optimal solution. Exact algorithms form an intriguing class of methods, but as they are based around an NP-hard combinatorial optimization problem, these methods remain computationally intractable in general. For example, recent state-of-the-art work (Cussens et al., 2017; Chen et al., 2016) only scale to problems with a few dozen nodes (van Beek and Hoffmann, 2015).[2] Older methods based on dynamic programming methods (Ott and Miyano, 2003; Singh and Moore, 2005; Silander and Myllymäki, 2006; Xiang and Kim, 2013; Loh and Bühlmann, 2014) also scale to roughly a few dozen nodes. By contrast, state-of-the-art approximate methods can scale to thousands of nodes (Ramsey et al., 2017; Aragam et al., 2015; Scanagatta et al., 2015, 2016).

Local vs. global search. Arguably the most popular approaches to optimizing (2.4) involve *local* search, wherein edges and parent sets are added sequentially, one node at a time. This is efficient as long as each node has only a few parents, but as the number of possible parents grows, local search rapidly becomes intractable. Furthermore, such strategies typically rely on severe structural assumptions such as bounded in-degree, bounded tree-width, or edge constraints. Since real-world networks often exhibit scale-free and small-world topologies (Watts and Strogatz, 1998; Barabási and Albert, 1999) with highly connected hub nodes, these kinds of structural assumptions are not only difficult to satisfy, but impossible to check.

[2] Cussens (2011) reports experiments with $d > 60$ under a constraint on the maximum parent size.

We note here promising work towards relaxing this assumption for discrete data (Scanagatta et al., 2015). By contrast, our method uses *global* search wherein the entire matrix W is updated in each step.

Model assumptions. The literature on DAG learning tends to be split between methods that operate on discrete data vs. methods that operate on continuous data. When viewed from the lens of (2.3), the reasons for this are not clear since both discrete and continuous data can be considered as special cases of the general score-based learning framework. Nonetheless, many (but not all) of the methods cited already only work under very specific assumptions on the data, the most common of which are categorical (discrete) and Gaussian (continuous). Since (2.3) is agnostic to the form of the data and loss function, there is significant interest in finding general methods that are not tied to specific model assumptions.

Conceptual clarity. Finally, on a higher level, a significant drawback of existing methods is their conceptual complexity: They are not straightforward to implement, require deep knowledge of concepts from the graphical modeling literature, and accelerating them involves many clever tricks. By contrast, the method we propose in this work is conceptually very simple, requires no background on graphical models, and can be implemented in just a few lines of code using existing black-box solvers.

2.2.3 Comparison

It is instructive to compare existing methods for learning DAGs against other methods in the machine learning literature. We focus here on two popular models: Undirected graphical models and deep neural networks. Undirected graphical models, also known as Markov networks, is recognized as a convex problem (Yuan and Lin, 2007; Banerjee et al., 2008) nowadays, and hence can be solved using black-box convex optimizers such as CVX (Grant and Boyd, 2014). However, one should not forget score-based methods based on discrete scores similar to (2.4) proliferated in the early days for learning undirected graphs (Pietra et al., 1997). More recently, extremely efficient algorithms have been developed for this problem using coordinate descent (Friedman et al., 2008) and Newton methods (Hsieh et al., 2014; Schmidt et al., 2009). As another example, deep neural networks are often learned using various descendants of stochastic gradient descent (SGD) (Bottou and Bousquet, 2008; Kingma and Ba, 2015; Bottou et al., 2016), although recent work has proposed other techniques such as ADMM (Taylor et al., 2016) and Gauss-Newton (Botev et al., 2017). One of the keys to the success of both of these models—and many other models in machine learning—was having a closed-form, tractable program for which existing techniques from the extensive optimization literature could be applied. In both cases, the application of principled optimization techniques led to significant breakthroughs. For undirected graphical models the major technical tool was convex optimization, and for deep networks the major technical tool was SGD.

Unfortunately, the general problem of DAG learning has not benefited in this way, and

one of our main goals in the current work is to formulate score-based learning similarly as a closed-form, continuous program. Arguably, the challenges with existing approaches stem from the intractable form of the program (2.4). One of our main goals in the current work is to formulate score-based learning via a similar closed-form, continuous program. The key device in accomplishing this is a smooth characterization of acyclicity that will be introduced in the next section.

2.3 A new characterization of acyclicity

In order to make (2.3) amenable to black-box optimization, we propose to replace the combinatorial acyclicity constraint $G(W) \in \mathbb{G}^d$ in (2.3) with a single smooth equality constraint $h(W) = 0$. Ideally, we would like a function $h : \mathbb{R}^{d\times d} \to \mathbb{R}$ that satisfies the following desiderata:

(a) $h(W) = 0$ if and only if W is acyclic (i.e. $G(W) \in \mathbb{G}^d$);

(b) The values of h quantify the "DAG-ness" of the graph;

(c) h is smooth;

(d) h and its derivatives are easy to compute.

Property (b) is useful in practice for diagnostics. By "DAG-ness", we mean some quantification of how severe violations from acyclicity become as W moves further from $\mathbb{G}$. Although there are many ways to satisfy (b) by measuring some notion of "distance" to $\mathbb{G}$, typical approaches would violate (c) and (d). For example, h might be the minimum ℓ_2 distance to $\mathbb{G}$ or it might be the sum of edge weights along all cyclic paths of W, however, these are either non-smooth (violating (c)) or hard to compute (violating (d)). If a function that satisfies desiderata (a)-(d) exists, we can hope to apply existing machinery for constrained optimization such as Lagrange multipliers. Consequently, the DAG learning problem becomes equivalent to solving a numerical optimization problem, which is agnostic about the graph structure.

We proceed in two steps: First, we consider the simpler case of binary adjacency matrices $B \in \{0, 1\}^{d\times d}$ (Section 2.3.1). Note that since $\{0, 1\}^{d\times d}$ is a discrete space, we cannot take gradients or do continuous optimization. For this we need the second step, in which we relax the function we originally define on binary matrices to real matrices (Section 2.3.2).

2.3.1 Special case: Binary adjacency matrices

When does a matrix $B \in \{0, 1\}^{d\times d}$ correspond to an acyclic graph? Recall the *spectral radius* $\rho(B)$ of a matrix B is the largest absolute eigenvalue of B. One simple characterization of acyclicity is the following:

Proposition 1 (Infinite series) *Suppose $B \in \{0, 1\}^{d\times d}$ and $\rho(B) < 1$. Then B is a DAG if and only if*

$$\operatorname{tr}(I - B)^{-1} = d. \tag{2.5}$$

Proof: It essentially boils down to the fact that $\operatorname{tr} B^k$ counts the number of length-k closed walks in a directed graph. Clearly an acyclic graph will have $\operatorname{tr} B^k = 0$ for all $k = 1, \ldots, \infty$. In other words, B has no cycles if and only if $f(B) = \sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii} = 0$, then

$$\operatorname{tr}(I - B)^{-1} = \operatorname{tr} \sum_{k=0}^{\infty} B^k = \operatorname{tr} I + \sum_{k=1}^{\infty} \operatorname{tr} B^k = d + \sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii} = d + f(B).$$

The desired result follows. ■

Unfortunately, the condition that $\rho(B) < 1$ is strong: although it is automatically satisfied when B is a DAG, it is generally not true otherwise, and furthermore the projection is nontrivial. Alternatively, instead of the infinite series, one could consider the characterization based on *finite* series $\sum_{k=1}^{d} \operatorname{tr} B^k = 0$, which does not require $\rho(B) < 1$. However, this is impractical for numerical reasons: The entries of B^k can easily exceed machine precision for even small values of d, which makes both function and gradient evaluations highly unstable. Therefore it remains to find a characterization that not only holds for all possible B, but also has numerical stability. Luckily, such function exists.

Proposition 2 (Matrix exponential) *A binary matrix $B \in \{0, 1\}^{d \times d}$ is a DAG if and only if*

$$\operatorname{tr} e^B = d. \tag{2.6}$$

Proof: Similar to Proposition 1 by noting that B has no cycles if and only if $(B^k)_{ii} = 0$ for all $k \geq 1$ and all i, which is true if and only if $\sum_{k=1}^{\infty} \sum_{i=1}^{d} (B^k)_{ii}/k! = \operatorname{tr} e^B - d = 0$. ■

It is worth pointing out that matrix exponential is well-defined for all square matrices. In addition to everywhere convergence, this characterization has an added bonus: As the number of edges in B increases along with the number of nodes d, the number of possible closed walks grows rapidly, so the trace characterization $\operatorname{tr}(I - B)^{-1}$ rapidly becomes ill-conditioned and difficult to manage. By re-weighting the number of length-k closed walks by $k!$, this becomes much easier to manage. While this is a useful characterization, it does not satisfy all of our desiderata since—being defined over a discrete space—it is not a smooth function. The final step is to extend Proposition 2 to all of $\mathbb{R}^{d \times d}$.

2.3.2 General case: Weighted adjacency matrices

Unfortunately, the characterization (2.6) fails if we replace B with an arbitrary weighted matrix W. However, we can replace B with any *nonnegative* weighted matrix, and the same argument use to prove Proposition 2 shows that (2.6) will still characterize acyclicity. Thus, to extend this to matrices with both positive and negative values, we can simply use the Hadamard product $W \circ W$, which leads to our main result.

Theorem 1 *A matrix $W \in \mathbb{R}^{d \times d}$ is a DAG if and only if*

$$h(W) = \operatorname{tr}\left(e^{W \circ W}\right) - d = 0, \tag{2.7}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential of A. Moreover, $h(W)$ has a simple gradient

$$\nabla h(W) = \left(e^{W \circ W}\right)^T \circ 2W, \tag{2.8}$$

and satisfies all of the desiderata (a)-(d).

The proof of (2.7) is similar to (2.6), and desiderata (c)-(d) follow from (2.8). To see why desiderata (b) holds, note that the proof of Proposition 1 shows that the power series $\operatorname{tr}(B + B^2 + \cdots)$ simply counts the number of closed walks in B, and the matrix exponential simply re-weights these counts. Replacing B with $W \circ W$ amounts to counting *weighted* closed walks, where the weight of each edge is w_{ij}^2. Thus, larger $h(W) > h(W')$ means either (a) W has more cycles than W' or (b) The cycles in W are more heavily weighted than in W'.

Moreover, notice that $h(W) \geq 0$ for all W since each term in the series is nonnegative. This gives another interesting perspective of the space of DAGs as the set of global minima of $h(W)$.

A key conclusion from Theorem 1 is that h and its gradient only involve evaluating the matrix exponential, which is a well-studied function in numerical analysis, and whose $O(d^3)$ algorithm (Al-Mohy and Higham, 2009) is readily available in many scientific computing libraries. Although the connection between trace of matrix power and number of cycles in the graph is well-known Harary and Manvel (1971), to the best of our knowledge, this characterization of acyclicity has not appeared in the DAG learning literature previously. We defer the discussion of other possible characterizations in the appendix. In the next section, we apply Theorem 1 to solve the program (2.3) to stationarity by treating it as an equality constrained program.

2.4 Optimization

Theorem 1 establishes a smooth, algebraic characterization of acyclicity that is also computable. As a consequence, the following equality-constrained program (ECP) is equivalent to (2.3):

$$\text{(ECP)} \qquad \begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & F(W) \\ \text{s.t.} \quad & h(W) = 0. \end{aligned} \tag{2.9}$$

The main advantage of (2.9) compared to both (2.3) and (2.4) is its amenability to classical techniques from the mathematical optimization literature. Nonetheless, since $\{W \mid h(W) = 0\}$ is a nonconvex set, (2.9) is a nonconvex program, hence we still inherit the difficulties associated with nonconvex optimization. In particular, we will be content to find stationary points of (2.9);

In the follows, we outline the algorithm for solving (2.9). It consists of three steps: (i) converting the *constrained* problem into a sequence of *unconstrained* subproblems, (ii) optimizing the unconstrained subproblems, and (iii) thresholding. The full algorithm is outlined in Algorithm 1.

Algorithm 1 NOTEARS algorithm

1. Input: Initial guess (W_0, α_0), progress rate $c \in (0, 1)$, tolerance $\epsilon > 0$, threshold $\omega > 0$.
2. For $t = 0, 1, 2, \ldots$:
   (a) Solve primal $W_{t+1} \leftarrow \text{argmin}_W L^\rho(W, \alpha_t)$ with ρ such that $h(W_{t+1}) < ch(W_t)$.
   (b) Dual ascent $\alpha_{t+1} \leftarrow \alpha_t + \rho h(W_{t+1})$.
   (c) If $h(W_{t+1}) < \epsilon$, set $\widetilde{W}_{\mathsf{ECP}} = W_{t+1}$ and break.
3. Return the thresholded matrix $\widehat{W} := \widetilde{W}_{\mathsf{ECP}} \circ 1(|\widetilde{W}_{\mathsf{ECP}}| > \omega)$.

2.4.1 Solving the ECP with augmented Lagrangian

We will use the augmented Lagrangian method (e.g. Nemirovski, 1999) to solve (ECP), which solves the original problem augmented by a quadratic penalty:

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & F(W) + \frac{\rho}{2}|h(W)|^2 \\ \text{s.t.} \quad & h(W) = 0 \end{aligned} \tag{2.10}$$

with a penalty parameter $\rho > 0$. A nice property of the augmented Lagrangian method is that it approximates well the solution of a *constrained* problem by the solution of *unconstrained* problems *without* increasing the penalty parameter ρ to infinity (Nemirovski, 1999). The algorithm is essentially a dual ascent method for (2.10). To begin with, the dual function with Lagrange multiplier α is given by

$$D(\alpha) = \min_{W \in \mathbb{R}^{d \times d}} L^\rho(W, \alpha), \tag{2.11}$$

$$\text{where} \quad L^\rho(W, \alpha) = F(W) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{2.12}$$

is the augmented Lagrangian. The goal is to find a local solution to the dual problem

$$\max_{\alpha \in \mathbb{R}} \quad D(\alpha). \tag{2.13}$$

Let $W_\alpha^\star$ be the local minimizer of the Lagrangian (2.11) at α, i.e. $D(\alpha) = L^\rho(W_\alpha^\star, \alpha)$. Since the dual objective $D(\alpha)$ is linear in α, the derivative is simply given by $\nabla D(\alpha) = h(W_\alpha^\star)$. Therefore one can perform dual gradient ascent to optimize (2.13):

$$\alpha \leftarrow \alpha + \rho h(W_\alpha^\star), \tag{2.14}$$

where the choice of step size ρ comes with the following convergence rate:

Proposition 3 (Corollary 11.2.1, Nemirovski, 1999) *For ρ large enough and the starting point α_0 near the solution $\alpha^\star$, the update (2.14) converges to $\alpha^\star$ linearly.*

In our experiments, typically fewer than 10 steps of the augmented Lagrangian scheme are required.

2.4.2 Solving the unconstrained subproblem

The augmented Lagrangian converts a *constrained* problem (2.10) into a sequence of *unconstrained* problems (2.11). We now discuss how to solve these subproblems efficiently. Let $\boldsymbol{w} = \mathrm{vec}(W) \in \mathbb{R}^p$, with $p = d^2$. The unconstrained subproblem (2.11) can be considered as a typical minimization problem over real vectors:

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} f(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 , \tag{2.15}$$

$$\text{where} \quad f(\boldsymbol{w}) = \ell(W; X) + \frac{\rho}{2}|h(W)|^2 + \alpha h(W) \tag{2.16}$$

is the smooth part of the objective. Our goal is to solve the above problem to high accuracy so that $h(W)$ can be sufficiently suppressed.

In the special case of $\lambda = 0$, the nonsmooth term vanishes and the problem simply becomes an unconstrained smooth minimization, for which a number of efficient numerical algorithms are available, for instance the L-BFGS (Byrd et al., 1995). To handle the nonconvexity, a slight modification (Nocedal and Wright, 2006, Procedure 18.2) needs to be applied.

When $\lambda > 0$, the problem becomes composite minimization, which can also be efficiently solved by the proximal quasi-Newton (PQN) method (Zhong et al., 2014). At each step k, the key idea is to find the descent direction through a quadratic approximation of the smooth term:

$$\boldsymbol{d}_k = \operatorname*{argmin}_{\boldsymbol{d}\in\mathbb{R}^p} \ \boldsymbol{g}_k^T \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T B_k \boldsymbol{d} + \lambda \|\boldsymbol{w}_k + \boldsymbol{d}\|_1 , \tag{2.17}$$

where $\boldsymbol{g}_k$ is the gradient of $f(\boldsymbol{w})$ and B_k is the L-BFGS approximation of the Hessian. Note that for each coordinate j, problem (2.17) has a closed form update $\boldsymbol{d} \leftarrow \boldsymbol{d} + z^\star e_j$ given by

$$z^\star = \operatorname*{argmin}_z \ \frac{1}{2}\underbrace{B_{jj}}_{a} z^2 + (\underbrace{\boldsymbol{g}_j + (B\boldsymbol{d})_j}_{b})z + \lambda|\underbrace{\boldsymbol{w}_j + \boldsymbol{d}_j}_{c} + z| = -c + S\left(c - \frac{b}{a}, \frac{\lambda}{a}\right). \tag{2.18}$$

Moreover, the low-rank structure of B_k enables fast computation for coordinate update. The precomputation time is only $O(m^2p + m^3)$ where $m \ll p$ is the memory size of L-BFGS, and each coordinate update is $O(m)$. Furthermore, since we are using sparsity regularization, we can further speed up the algorithm by aggressively shrinking the active set of coordinates based on their subgradients (Zhong et al., 2014), and exclude the remaining dimensions from being updated. With the updates restricted to the active set $\mathcal{S}$, all dependencies of the complexity on $O(p)$ becomes $O(|\mathcal{S}|)$, which is substantially smaller. Hence the overall complexity of L-BFGS update is $O(m^2|\mathcal{S}| + m^3 + m|\mathcal{S}|T)$, where T is the number of inner iterations, typically $T = 10$.

2.4.3 Details of proximal quasi-Newton

Recall $B_k \in \mathbb{R}^{p\times p}$ is the low-rank approximation of the Hessian matrix given by L-BFGS updates. Let the memory size of L-BFGS be m, which is taken to be $m \ll p$. The compact

Algorithm 2 Proximal Quasi-Newton for unconstrained problem (Zhong et al., 2014)

1. Input: $\boldsymbol{w}_0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{w}_0)$, active set $\mathcal{S} = [p]$.
2. For $k = 0, 1, 2, \ldots$:
   (a) Shrink $\mathcal{S}$ to rule out j with $w_j = 0$ or small subgradient $|\partial_j L(\boldsymbol{w})|$
   (b) If shrinking stopping criteria is satisfied
      i. Reset $\mathcal{S} = [p]$ and L-BFGS memory
      ii. Update shrinking stopping criteria and continue
   (c) Solve (2.17) for descent direction $\boldsymbol{d}_k$ using coordinate update (2.18) on active set
   (d) Line search for step size $\eta \in (0, 1]$ until Armijo rule is satisfied:

$$f(\boldsymbol{w}_k + \eta \boldsymbol{d}_k) \leq f(\boldsymbol{w}_k) + \eta c_1 (\lambda \|\boldsymbol{w}_k + \boldsymbol{d}_k\|_1 - \lambda \|\boldsymbol{w}_k\| + \boldsymbol{g}_k^T \boldsymbol{d}_k), \tag{2.21}$$

   where c_1 is some small constant, typically set to 10^{-3} or 10^{-4}.
   (e) Generate new iterate $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + \eta \boldsymbol{d}_k$
   (f) Update $\boldsymbol{g}, \boldsymbol{s}, \boldsymbol{y}, Q, R, \widehat{Q}$ restricted to $\mathcal{S}$

form of L-BFGS update can be written as

$$B_k = \gamma_k I - Q \widehat{Q}, \tag{2.19}$$

where

$$\begin{gathered}
Q = \begin{bmatrix} \gamma_k S_k & Y_k \end{bmatrix}, \; R = \begin{bmatrix} \gamma_k S_k^T S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1}, \; \widehat{Q} = R Q^T, \\
S_k = \begin{bmatrix} \boldsymbol{s}_{k-m} & \cdots & \boldsymbol{s}_{k-1} \end{bmatrix}, \; Y_k = \begin{bmatrix} \boldsymbol{y}_{k-m} & \cdots & \boldsymbol{y}_{k-1} \end{bmatrix}, \\
\boldsymbol{s}_k = \boldsymbol{w}_{k+1} - \boldsymbol{w}_k, \; \boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k, \; \gamma_k = \boldsymbol{y}_{k-1}^T \boldsymbol{y}_{k-1} / \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1}, \\
D_k = \operatorname{diag} \begin{bmatrix} \boldsymbol{s}_{k-m}^T \boldsymbol{y}_{k-m} & \cdots & \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}, \\
(L_k)_{ij} = \begin{cases} \boldsymbol{s}_{k-m+i-1}^T \boldsymbol{y}_{k-m+j-1} & \text{if } i > j \\ 0 & \text{otherwise} \end{cases}.
\end{gathered} \tag{2.20}$$

The low rank structure of B_k enables fast computation of subsequent coordinate descent procedure. Specifically, notice that all $Q, R, \widehat{Q}$, and $\operatorname{diag}(B)$ can be precomputed in $O(m^2 p + m^3)$ time, which is significantly smaller than naive Hessian inversion $O(p^3)$. After precomputation, in each coordinate update, both a and c in (2.18) can be computed and updated in $O(1)$ time. Moreover, let $\widehat{\boldsymbol{d}} = \widehat{Q} \boldsymbol{d} \in \mathbb{R}^{2m}$, we have $(B\boldsymbol{d})_j = \gamma \boldsymbol{d}_j - Q_{j,:} \widehat{\boldsymbol{d}}$, which suggests b in (2.18) only requires $O(m)$ to compute and update. Therefore each coordinate update is $O(m)$.

The detailed procedure of PQN is outlined in Algorithm 2.

2.4.4 Thresholding

In regression problems, it is known that post-processing estimates of coefficients via hard thresholding provably reduces the number of false discoveries (Zhou, 2009; Wang et al., 2016). Motivated by these encouraging results, we threshold the edge weights as follows: After obtaining a stationary point $\widetilde{W}_{\mathsf{ECP}}$ of (2.10), given a fixed threshold $\omega > 0$, set any weights smaller than ω in absolute value to zero. This strategy also has the important effect of "rounding" the numerical solution of the augmented Lagrangian (2.10), since due to numerical precisions the solution satisfies $h(\widetilde{W}_{\mathsf{ECP}}) \leq \epsilon$ for some small tolerance ϵ near machine precision (e.g. $\epsilon = 10^{-8}$), rather than $h(\widetilde{W}_{\mathsf{ECP}}) = 0$ strictly. However, since $h(\widetilde{W}_{\mathsf{ECP}})$ explicitly *quantifies* the "DAG-ness" of $\widetilde{W}_{\mathsf{ECP}}$ (see desiderata (b), Section 2.3), a small threshold ω suffices to rule out cycle-inducing edges.

2.5 Experiments

We compared our method against greedy equivalent search (GES) (Chickering, 2002; Ramsey et al., 2017), the PC algorithm (Spirtes and Glymour, 1991), and LiNGAM (Shimizu et al., 2006). For GES, we used the fast greedy search (FGS) implementation from Ramsey et al. (2017). Since the accuracy of PC and LiNGAM was significantly lower than either FGS or NOTEARS, we only report the results against FGS here. This is consistent with previous work on score-based learning (Aragam et al., 2015), which also indicates that FGS outperforms other techniques such as hill-climbing and MMHC (Tsamardinos et al., 2006). FGS was chosen since it is a state-of-the-art algorithm that scales to large problems.

Datasets We used simulated graphs from two well-known ensembles of random graphs:

- *Erdös-Rényi (ER).* Random graphs whose edges are added independently with equal probability p. We simulated models with d, $2d$, and $4d$ edges (in expectation) each, denoted by ER-1, ER-2, and ER-4, respectively.
- *Scale-free networks (SF).* Networks simulated according to the preferential attachment process described in Barabási and Albert (1999). We simulated scale-free networks with $4d$ edges and $\beta = 1$, where β is the exponent used in the preferential attachment process.

Scale-free graphs are popular since they exhibit topological properties similar to real-world networks such as gene networks, social networks, and the Internet. Given a random acyclic graph $B \in \{0,1\}^{d\times d}$ from one of these two ensembles, we assigned edge weights independently from $\mathsf{Unif}\big([-2,-0.5]\cup[0.5,2]\big)$ to obtain a weight matrix $W = [\mathbf{w}_1 \mid \cdots \mid \mathbf{w}_d] \in \mathbb{R}^{d\times d}$. Given W, we sampled $\mathbf{x} = W^T\mathbf{x} + \mathbf{z} \in \mathbb{R}^d$ according to the following three noise models:

- *Gaussian noise* (Gauss). $\mathbf{z} \sim \mathsf{N}(0, I_{d\times d})$.
- *Exponential noise* (Exp). $z_j \sim \mathsf{Exp}(1)$, $j = 1,\dots,d$.
- *Gumbel noise* (Gumbel). $z_j \sim \mathsf{Gumbel}(0,1)$, $j = 1,\dots,d$.

Based on these models, we generated random datasets $X \in \mathbb{R}^{n \times d}$ by generating the rows i.i.d. according to one of the models above. For each simulation, we generated n samples for graphs with $d \in \{10, 20, 50, 100\}$ nodes. To study both high- and low-dimensional settings, we used $n \in \{20, 1000\}$.

Methods For each dataset, we ran FGS, PC, and LinGAM and NOTEARS to compare the performance in reconstructing the DAG B. We used the following implementations:

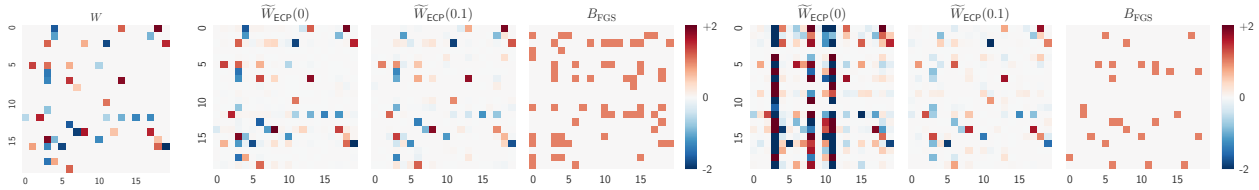
- FGS and PC were implemented through the `py-causal` package, available at https://github.com/bd2kccd/py-causal. Both of these methods are written in highly optimized Java code.
- LinGAM was implemented using the author's Python code: https://sites.google.com/site/sshimizu06/lingam.

Since the accuracy of PC and LiNGAM was significantly lower than either FGS or NOTEARS, we only report the results against FGS. A few comments on FGS are in order: 1) FGS estimates a graph, so it does not output any parameter estimates; 2) Instead of returning a DAG, FGS returns a CPDAG (Chickering, 2002), which contains undirected edges; 3) FGS has a single tuning parameter that controls the strength of regularization. Thus, in our evaluations, we treated FGS favourably by treating undirected edges as true positives as long as the true graph had a directed edge in place of the undirected edge. For tuning parameters, we used the values suggested by the authors of the FGS code.

Denote the estimate returned by FGS by B_{FGS}. We fix the threshold at $\omega = 0.3$ (see below for sensitivity analysis). Having fixed ω, when there is no regularization, NOTEARS requires no tuning. With ℓ_1-regularization, NOTEARS-ℓ_1 requires a choice of λ which was selected as follows: Based on the estimate returned by FGS, we tuned λ so that the selected graph (after thresholding) had the same number of edges as B_{FGS} (or as close as possible). This ensures that the results are not influenced by hyperparameter tuning, and fairly compares each method on graphs of roughly the same complexity. Denote this estimate by $\widehat{W}$ and the resulting adjacency matrix by $\widehat{B} = A(\widehat{W})$.

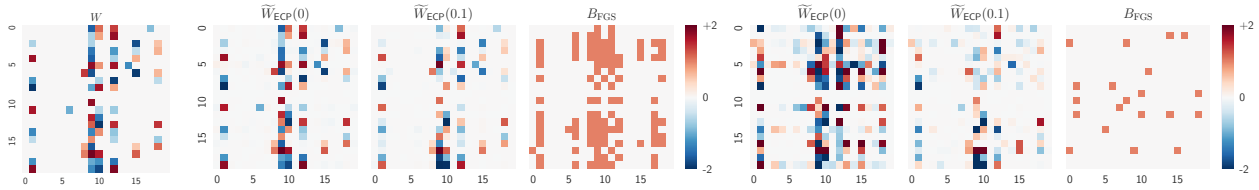
Metrics We evaluated the learned graphs on four common graph metrics: 1) False discovery rate (FDR), 2) True positive rate (TPR), 3) False positive rate (FPR), and 4) Structural Hamming distance (SHD). Recall that SHD is the total number of edge additions, deletions, and reversals needed to convert the estimated DAG into the true DAG. Since we consider directed graphs, a distinction between True Positives (TP) and Reversed edges (R) is needed: the former is estimated with correct direction whereas the latter is not. Likewise, a False Positive (FP) is an edge that is not in the undirected skeleton of the true graph. In addition, Positive (P) is the set of estimated edges, True (T) is the set of true edges, False (F) is the set of non-edges in the ground truth graph. Finally, let (E) be the extra edges from the skeleton, (M) be the missing edges from the skeleton. The four metrics are then given by:

1. FDR $= (R + FP)/P$
2. TPR $= TP/T$



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 2.1: Visual comparison of the learned weighted adjacency matrix on a 20-node graph with $n = 1000$ (large samples) and $n = 20$ (insufficient samples): $\widetilde{W}_{\mathsf{ECP}}(\lambda)$ is the proposed NOTEARS algorithm with ℓ_1-regularization λ, and B_{FGS} is the binary estimate of the baseline (Ramsey et al., 2017). The proposed algorithms perform well on large samples, and remains accurate on small n with ℓ_1 regularization.



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

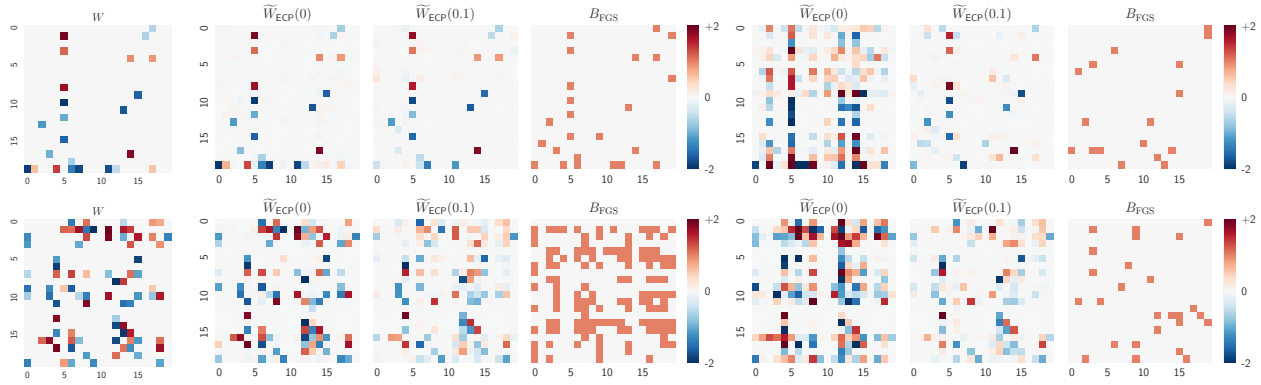
Figure 2.2: Parameter estimates of $\widetilde{W}_{\mathsf{ECP}}$ on a scale-free graph. Without the additional thresholding step in Algorithm 1, NOTEARS still produces consistent estimates of the true graph. The proposed method estimates the weights very well with large samples even without regularization, and remains accurate on insufficient samples when ℓ_1-regularization is introduced. See also Figure 2.1.

3. FPR $= (R + FP)/F$
4. SHD $= E + M + R$.

2.5.1 Parameter estimation

We first performed a qualitative study of the solutions obtained by NOTEARS *without thresholding* by visualizing the weight matrix $\widetilde{W}_{\mathsf{ECP}}$ obtained by solving (ECP) (i.e. $\omega = 0$). This is illustrated in Figures 2.1 (ER-2) and 2.2 (SF-4). The key takeaway is that our method provides (empirically) consistent parameter estimates of the true weight matrix W. The final thresholding step in Algorithm 1 is only needed to ensure accuracy in structure learning. It also shows how effective is ℓ_1-regularization in small n regime.

Figure 2.3 shows learned weighted adjacency matrices for ER1 and ER4. One can observe the same trend: with large n, both regularized and unregularized NOTEARS works well compared to FGS, and with small n, due to identifiability, the unregularized NOTEARS suffers significantly, yet with the help of ℓ_1-regularization we can still accurately recover the true underlying graph.



(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 2.3: Visual comparison of the learned weighted adjacency matrix on a 20-node graph with $n = 1000$ (large samples) and $n = 20$ (insufficient samples): $\widetilde{W}_{\mathsf{ECP}}(\lambda)$ is the proposed NOTEARS algorithm with ℓ_1-regularization λ, and B_{FGS} is the binary estimate of the baseline (Ramsey et al., 2017). Top row: ER1, bottom row: ER4.

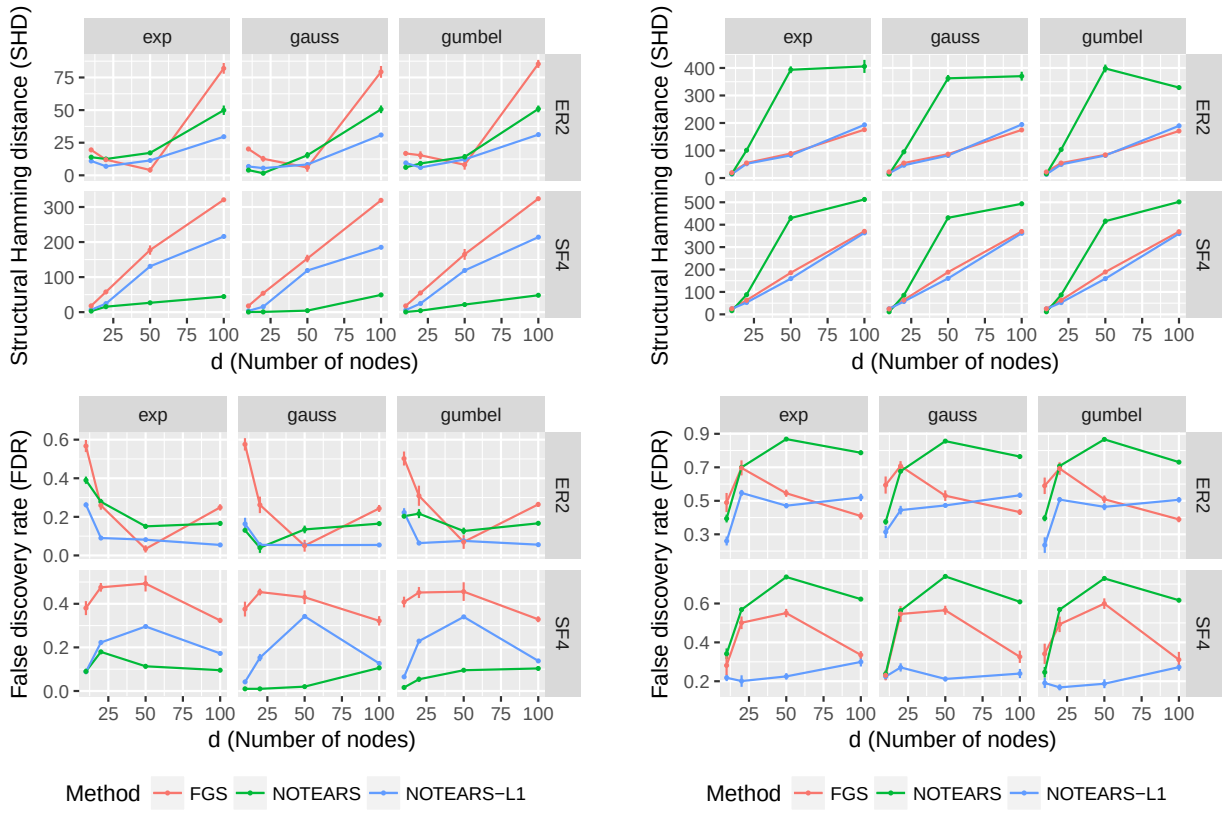
2.5.2 Structure learning

We now examine our method for structure recovery, which is shown in Figure 3.1. For brevity, we only report the numbers for the structural Hamming distance (SHD) here, but complete figures and tables for additional metrics can be found in the supplement. Consistent with previous work on greedy methods, FGS is very competitive when the number of edges is small (ER-2), but rapidly deteriorates for even modest numbers of edges (SF-4). In the latter regime, NOTEARS shows significant improvements. This is consistent across each metric we evaluated, and the difference grows as the number of nodes d gets larger. Also notice that our algorithm performs uniformly better for each noise model (Exp, Gauss, and Gumbel), without leveraging any specific knowledge about the noise type. Again, ℓ_1-regularizer helps significantly in the small n setting.

Figure 2.5 and Figure 2.6 shows structure recovery results for $n = 1000$ and $n = 20$ for various random graphs and SEM noise types. Other than fixed ω, we also included the optimal choice of thresholding, marked as "best". The trend is consistent with the main text: our method in general outperforms FGS, without tuning ω to the optimum for each setting.

2.5.3 Sensitivity of threshold

We demonstrate the effect of threshold in Figure 2.7. For each setting, we computed the "ROC" curve for FDR and TPR with varying level of threshold, while ensuring the resulting graph is indeed a DAG. On the right, we also present the estimated edge weights of $\widetilde{W}_{\mathsf{ECP}}$ in decreasing order. One can first observe that in all cases most of the edge weights are equal or close to zero as expected. The remaining question is how to choose a threshold that separates out these (near zero) from signals (away from zero) so that best performance can be achieved. With enough samples, one can often notice a sudden change in the weight distribution as



(a) SHD with $n = 1000$

(b) SHD with $n = 20$

Figure 2.4: Structure recovery in terms of SHD and FDR to the true graph (lower is better). Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.

in Figure 2.7(a)(c). With insufficient samples, the breakpoint is less clear, and the optimal choice that balances between TPR and FDR is depends on the specific settings. Nonetheless, the predictive performance is less sensitive to threshold value as one can see from the slope of the decrease in the weights before getting close to zero. Indeed, in our experiments, we found a fixed threshold $\omega = 0.3$ is a suboptimal yet reasonable choice across many different settings.

2.5.4 Sensitivity of weight scale

We investigate the effect of weight scaling to the NOTEARS algorithm in Figure 2.8. In particular, we run experiments with $w_{ij} \in \alpha \cdot [0.5, 2] \cup -\alpha \cdot [0.5, 2]$ with $\alpha \in \{1.0, 0.9, 0.8, \ldots, 0.1\}$. On the left, we plot the smallest threshold ω required to obtain a DAG (see Section 4.3) for different scale α. Overall, across different values of α, the variation in the smallest ω required is minimal. We also hasten to point out that this also decreases the signal to noise ratio (SNR), which more directly affects the accuracy. Indeed, in the figure on the right, we can observe (as expected) some performance drop when using smaller value of α.

2.5.5 Real-data

We also compared FGS and NOTEARS on a real dataset provided by Sachs et al. (2005). This dataset consists of continuous measurements of expression levels of proteins and phospholipids in human immune system cells ($n = 7466$ $d = 11$, 20 edges). This dataset is a common benchmark in graphical models since it comes with a known *consensus network*, that is, a gold standard network based on experimental annotations that is widely accepted by the biological community. In our experiments, FGS estimated 17 total edges with an SHD of 22, compared to 16 for NOTEARS with an SHD of 22.

2.6 Discussion

We have proposed a new method for learning DAGs from data based on a continuous optimization program. This represents a significant departure from existing approaches that search over the discrete space of DAGs, resulting in a difficult optimization program. We also proposed two optimization schemes for solving the resulting program to stationarity, and illustrated its advantages over existing methods such as greedy equivalence search. Crucially, by performing global updates (e.g. all parameters at once) instead of local updates (e.g. one edge at a time) in each iteration, our method is able to avoid relying on assumptions about the local structure of the graph. To conclude, let us discuss some of the limitations of our method and possible directions for future work.

First, it is worth emphasizing once more that the equality constrained program (2.9) is a nonconvex program. Thus, although we overcome the difficulties of *combinatorial* optimization, our formulation still inherits the difficulties associated with *nonconvex* optimization. In particular, black-box solvers can at best find stationary points of (2.9). With the exception of exact methods, however, existing methods suffer from this drawback as well.[3] The main advantage of NOTEARS then is *smooth*, *global search*, as opposed to combinatorial, local search; and furthermore the search is delegated to standard numerical solvers.

Second, the current work relies on the smoothness of the score function, in order to make use of gradient-based numerical solvers to guide the graph search. However it is also interesting to consider non-smooth, even discrete scores such as BDe (Heckerman et al., 1995). Off-the-shelf techniques such as Nesterov's smoothing (Nesterov, 2005) could be useful, however more thorough investigation is left for future work.

Third, since the evaluation of the matrix exponential is $O(d^3)$, the computational complexity of our method is cubic in the number of nodes, although the constant is small for sparse matrices. In fact, this is one of the key motivations for our use of second-order methods (as opposed to first-order), i.e. to reduce the number of matrix exponential computations. By using second-order methods, each iteration make significantly more progress than first-order methods. Furthermore, although in practice not many iterations ($t \sim 10$) are required, we have not established any worst-case iteration complexity results. Notwithstanding, NOTEARS

[3]GES (Chickering, 2002) is known to find the global minimizer in the limit $n \to \infty$ under certain assumptions, but this is not guaranteed for finite samples.

already outperforms existing methods when the in-degree is large, which is known difficult spot for existing methods. We leave it to future work to study these cases in more depth.

Lastly, in our experiments, we chose a fixed, suboptimal value of $\omega > 0$ for thresholding (Section 2.4.4). Clearly, it would be preferable to find a data-driven choice of ω that adapts to different noise-to-signal ratios and graph types. It is an interesting direction for future to study such choices.

The code is publicly available at https://github.com/xunzheng/notears.

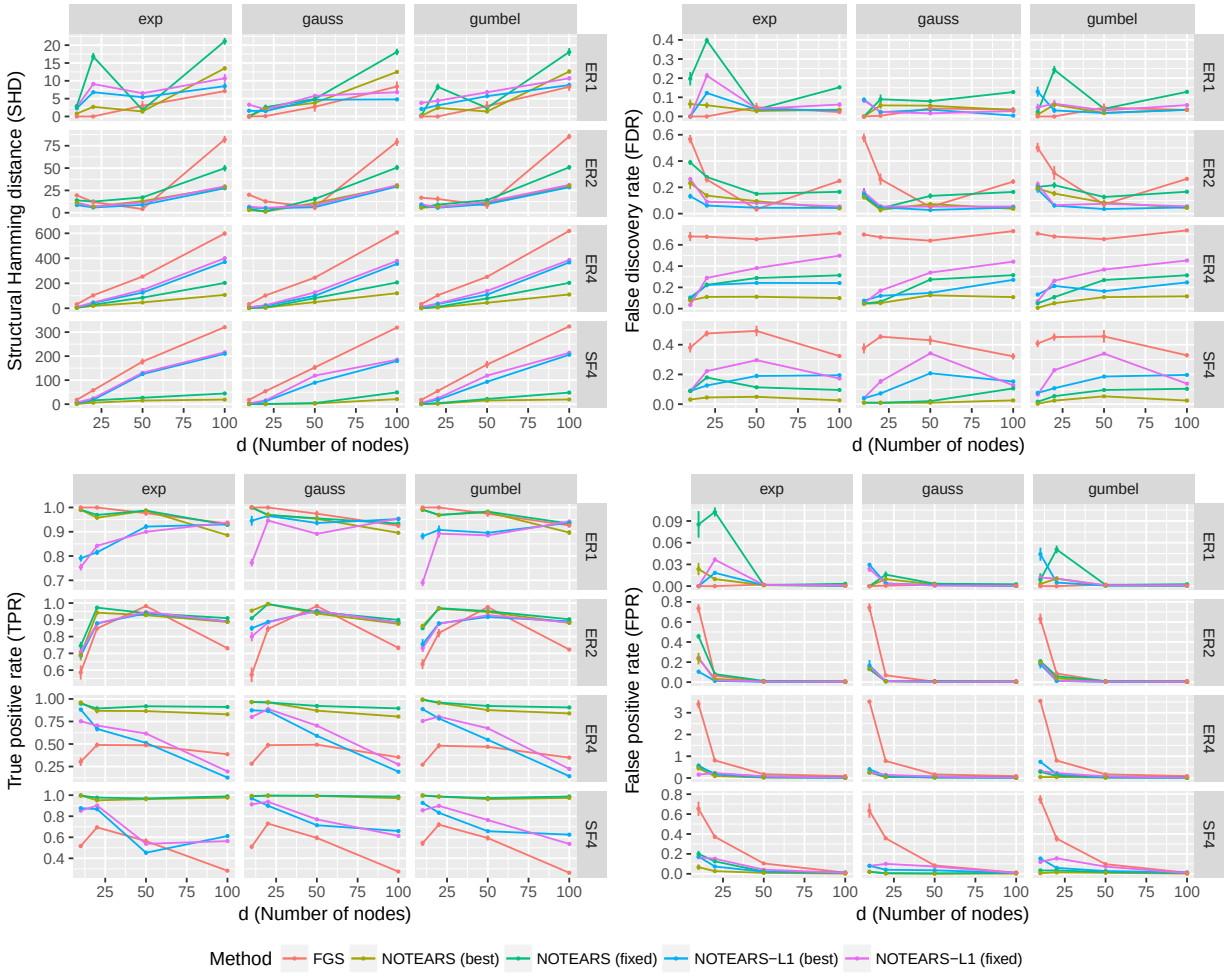


Figure 2.5: Structure recovery results for $n = 1000$. Lower is better, except for TPR (lower left), for which higher is better. Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.

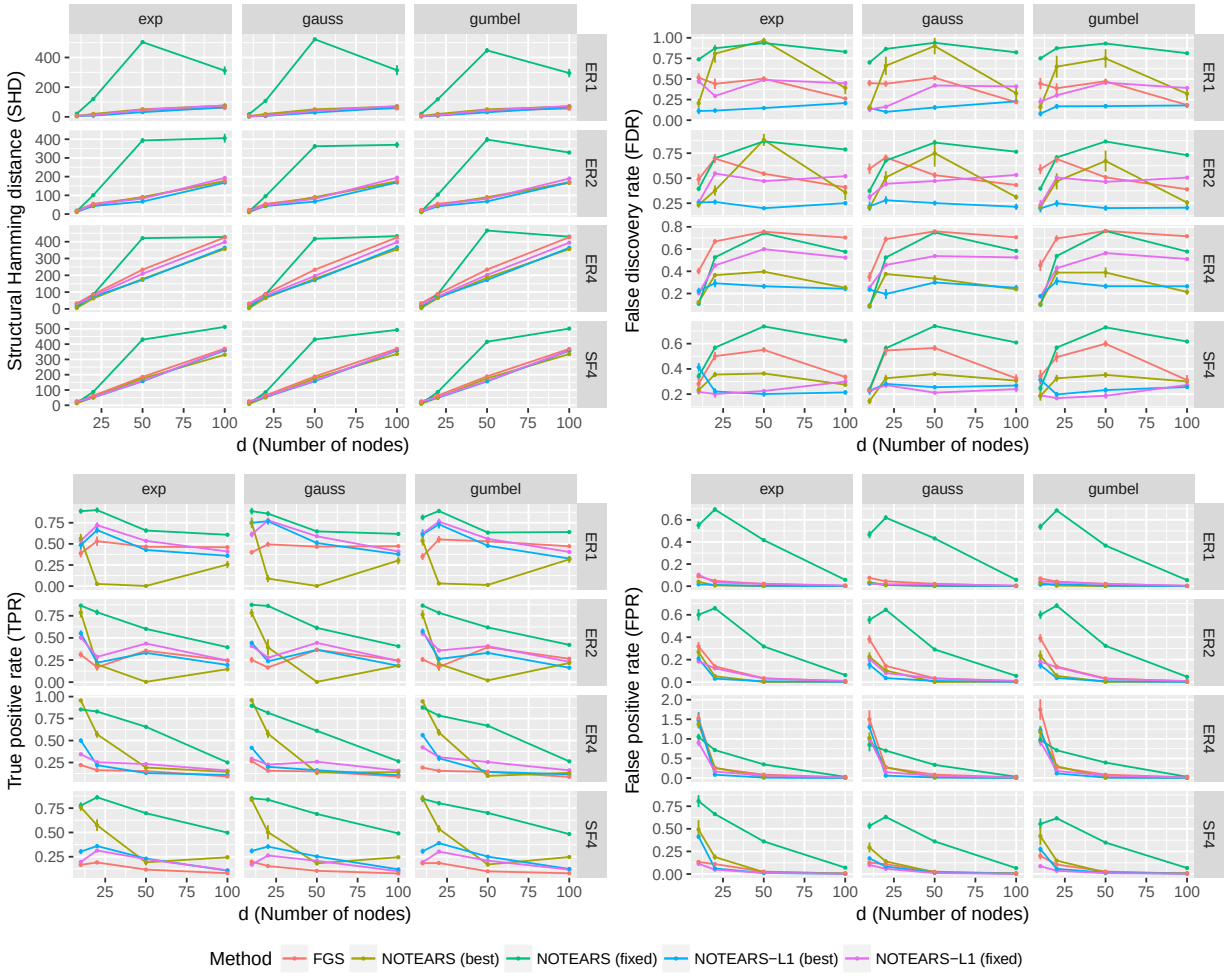
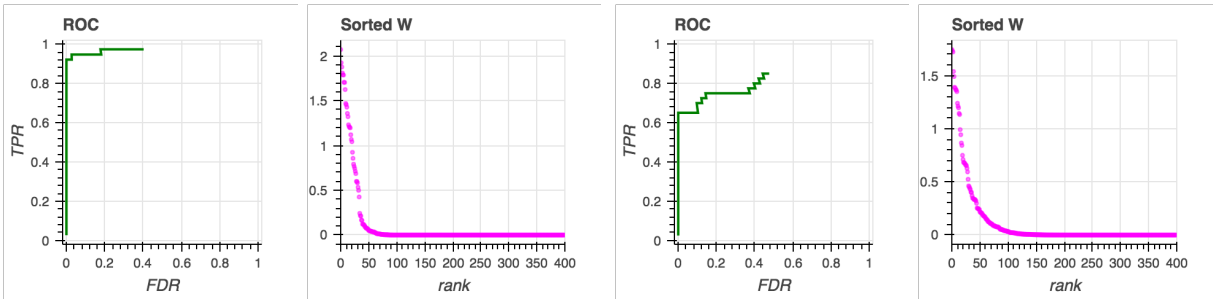
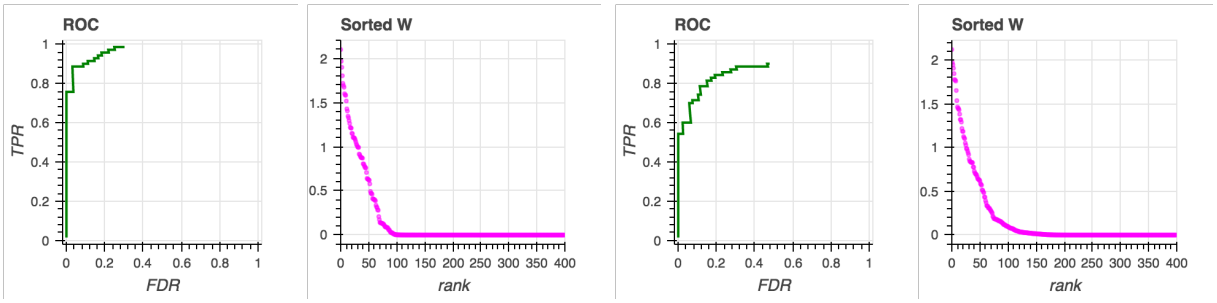


Figure 2.6: Structure recovery results for $n = 20$. Lower is better, except for TPR (lower left), for which higher is better. Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.



(a) ER2, $n = 1000$

(b) ER2, $n = 20$



(c) SF4, $n = 1000$

(d) SF4, $n = 20$

Figure 2.7: Illustration of the effect of the threshold with $d = 20$ and $\lambda = 0.1$. For each subfigure, ROC curve (left) shows FDR and TPR with varying level of threshold, and sorted weights (right) plots the entries of $\widetilde{W}_{\mathsf{ECP}}$ in decreasing order.

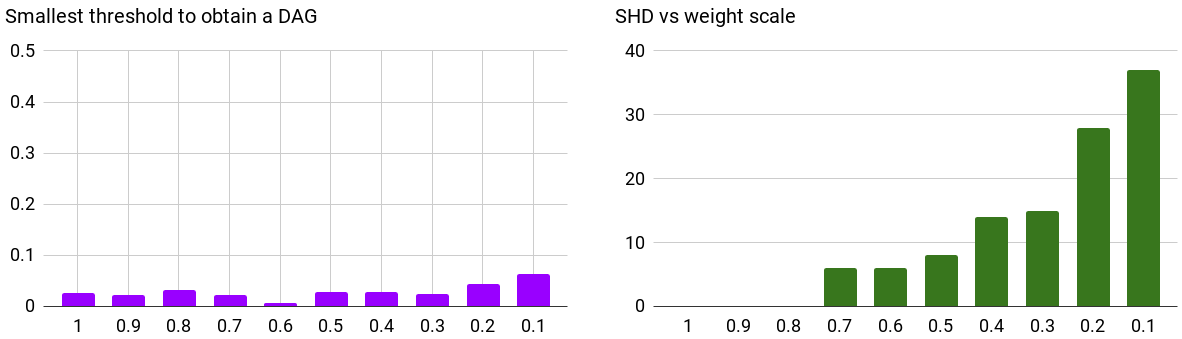


Figure 2.8: Varying weight scale $\alpha \in \{1.0, \ldots, 0.1\}$ with $d = 20$ and $n = 1000$ on an ER-2 graph. (Left) Smallest threshold ω such that $\widehat{W}$ is a DAG. (Right) SHD between ground truth and NOTEARS, lower the better. The minimum ω remains stable, while the accuracy of NOTEARS drops as expected since the SNR decreases with α.

Chapter 3

Learning Nonparametric DAGs with Continuous Optimization

In the previous chapter we established continuous optimization for *linear* structural equation models (SEM). However, linearity in practice can be a strong assumption, and can potentially lead to incorrect structure recovery. In fact, many existing methods for learning DAGs typically rely on specific model assumptions (e.g. linear or additive) and specialized algorithms (e.g. constraint-based or greedy optimization) that are not broadly applicable to different data. As a result, the burden is on the user to choose amongst many possible models and algorithms, which requires significant expertise. Thus, there is a need for a general framework for learning different DAG models—subsuming, for example, linear, parametric, and nonparametric—that does not require specialized algorithms. Ideally, the problem could be formulated as a conventional optimization problem that can be tackled with general purpose solvers, much like the current state-of-the-art for undirected graphical models (e.g. Suggala et al., 2017; Yang et al., 2015; Liu et al., 2009; Hsieh et al., 2014; Banerjee et al., 2008).

In this chapter, we develop such a general algorithmic framework for score-based learning of DAG models. This framework is flexible enough to learn general nonparametric dependence while also easily adapting to parametric and semiparametric models, including nonlinear models. The framework follows our approach for the *linear* SEM where we recast the score-based optimization problem as a *continuous* problem, instead of the traditional combinatorial approach. This allows generic optimization routines to be used in minimizing the score, providing a clean conceptual formulation of the problem that can be approached using any of the well-known algorithms from the optimization literature. However, the previous approach developed for the *linear SEM* relies heavily on the parametrization in terms of a weighted adjacency matrix $W \in \mathbb{R}^{d \times d}$ of the problem, which can be seen to be a stringent restriction on the class of models, and one of the key technical contributions of this chapter is extending this to general nonparametric problems, where no such parametrization in terms of a weighted adjacency matrix exists.

3.1 Our contributions

Our main contributions can be summarized as follows:

- We develop a generic optimization problem that can be applied to nonlinear and nonparametric SEM and discuss various special cases including additive models and index models. In contrast to existing work, we show how this optimization problem can be solved to stationarity with generic solvers, eliminating the necessity for specialized algorithms and models.
- We extend the existing smooth characterization of acyclicity to general nonparametric models, and apply this to several popular examples for modeling nonlinear dependencies (Section 3.3).
- We consider in detail two classes of nonparametric estimators defined through 1) Neural networks and 2) Orthogonal basis expansions, and study their properties (Section 3.4).
- We run extensive empirical evaluations on a variety of nonparametric and semiparametric models against recent state-of-the-art methods in order to demonstrate the effectiveness and generality of our framework (Section 3.5).

As with all score-based approaches to learning DAGs, ours relies on a nonconvex optimization problem. Despite this, we show that off-the-shelf solvers return stationary points that outperform other state-of-the-art methods. Finally, the algorithm itself can be implemented in standard machine learning libraries such as PyTorch, which should help the community to extend our approach to richer models moving forward.

3.2 Background

We briefly review the notations. Norms will always be explicitly subscripted to avoid confusion: $\|\cdot\|_p$ is the ℓ_p-norm on vectors, $\|\cdot\|_{L^p}$ is the L^p-norm on functions, $\|\cdot\|_{p,q}$ is the (p,q)-norm on matrices, and $\|\cdot\|_F = \|\cdot\|_{2,2}$ is the matrix Frobenius norm. For functions $f : \mathbb{R}^s \to \mathbb{R}$ and a matrix $A \in \mathbb{R}^{n\times s}$, we adopt the convention that $f(A) \in \mathbb{R}^n$ is the vector whose ith element is $f(a^i)$, where a^i is the ith row of A. We use $\mathbb{G}$ to denote the set of directed acyclic graphs (DAGs).

3.2.1 Score-based learning of nonparametric SEM

Our approach is based on (acyclic) structural equation models as follows. Let $\mathbf{x} = (x_1, \dots, x_d)$ be a random vector and $G = (V, E)$ a DAG with $V = \mathbf{x}$. We assume that there exist functions $f_j : \mathbb{R}^d \to \mathbb{R}$ [1] and $g_j : \mathbb{R} \to \mathbb{R}$ such that

$$\begin{aligned} &\mathbb{E}[x_j|\mathbf{x}_{\mathrm{pa}(j)}] = g_j(f_j(\mathbf{x})),\ \mathbb{E}f_j(\mathbf{x}) = 0,\ \text{and} \\ &f_j(u_1, \dots, u_d) \text{ does not depend on } u_k \text{ if } k \notin \mathrm{pa}(j). \end{aligned} \tag{3.1}$$

[1] The reason for writing $f_j(\mathbf{x})$ instead of $f_j(\mathbf{x}_{\mathrm{pa}(j)})$ is to simplify notation by ensuring each f_j is defined on the same space.

Here, $\text{pa}(j)$ denotes the parents of x_j in G. Formally, the second line in (3.1) means that for any $k \notin \text{pa}(j)$, the function $a(u) := f_j(x_1, \ldots, x_{k-1}, u, x_{k+1}, \ldots, x_d)$ is constant for all $u \in \mathbb{R}$. Thus, G encodes the conditional independence structure of $\mathbf{x}$. The functions g_j, which are typically known, allow for possible non-additive errors such as in generalized linear models (GLMs). The model (3.1) is quite general and includes additive noise models, linear and generalized linear models, and additive models as special cases (Section 3.3.2).

In this setting, the DAG learning problem can be stated as follows: Given a data matrix $X = [X_{\cdot 1}|\cdots|X_{\cdot d}] \in \mathbb{R}^{n \times d}$ consisting of n i.i.d. observations of the model (3.1), we seek to learn the DAG $G(\mathbf{x})$ that encodes the dependency between the variables in $\mathbf{x}$. Our approach is to learn $f = (f_1, \ldots, f_d)$ such that $G(f) = G(\mathbf{x})$ using a score-based approach. Given a loss function $\ell(y, \hat{y})$ such as least squares or the negative log-likelihood, we consider the following program:

$$\min_f \; L(f) \quad \text{s.t. } G(f) \in \mathbb{G}, \quad \text{where } L(f) = \frac{1}{n} \sum_{j=1}^{d} \ell(X_{\cdot j}, f_j(X)). \tag{3.2}$$

There are two challenges in this formulation: 1) How to enforce the acyclicity constraint that $G(f) \in \mathbb{G}$, and 2) How to enforce sparsity in the learned DAG $G(f)$? Previous work using linear and generalized linear models rely on a parametric representation of G via a weighted adjacency matrix $W \in \mathbb{R}^{d \times d}$, which is no longer well-defined in the model (3.1). To address this, we develop a suitable surrogate of W defined for general nonparametric models, to which we can apply the previous trace exponential regularizer.

3.2.2 Identifiability

Existing papers approach this problem as follows: 1) Assume a specific model for (3.1), 2) Prove identifiability for this specific model, and 3) Develop a specialized algorithm for learning this specific model. By contrast, our approach is generic: We do not assume any particular model form or algorithm, and instead develop a general framework that applies to any model that is identifiable. By now, there is a well-cataloged list of identifiability results for various linear, parametric, and nonlinear models, which we review briefly below (see also Section 3.3.2).

When the model (3.1) holds, the graph G is not necessarily uniquely defined: A well-known example is when X is jointly normally distributed, in which case the f_j are linear functions, and where it can be shown that the graph G is not uniquely specified. Fortunately, it is known that this case is somewhat exceptional: Assuming additive noise, as long as the f_j are linear with non-Gaussian errors (Kagan et al., 1973; Shimizu et al., 2006; Loh and Bühlmann, 2014) or the functions f_j are nonlinear (Hoyer et al., 2008a; Zhang and Hyvärinen, 2009; Peters et al., 2014), then the graph G is generally identifiable. We refer the reader to Peters et al. (2014) for details. Another example are so-called *quadratic variance function* models, which are parametric models that subsume many generalized linear models (Park and Raskutti, 2017; Park and Park, 2018). In the sequel, we assume that the model is chosen such that the graph G is uniquely defined from (3.1), and this dependence will be emphasized by writing

$G = G(X)$. Similarly, any collection of functions $f = (f_1, \ldots, f_d)$ defines a graph $G(f)$ in the obvious way. See Section 3.3.2 for specific examples with discussion on identifiability.

3.2.3 Related works

The problem of learning nonlinear and nonparametric DAGs from data has generated significant interest in recent years, including additive models (Bühlmann et al., 2014; Voorman et al., 2014; Rothenhäusler et al., 2018), generalized linear models (Park and Park, 2018; Park and Raskutti, 2017; Park and Park, 2019; Gu et al., 2018), additive noise models (Hoyer et al., 2008a; Peters et al., 2014; Blöbaum et al., 2018; Mooij et al., 2016), post-nonlinear models (Zhang and Hyvärinen, 2009; Zhang et al., 2016) and general nonlinear SEM (Monti et al., 2019; Goudet et al., 2018; Kalainathan et al., 2018; Sgouritsa et al., 2015). Recently, Yu et al. (2019) proposed to use graph neural networks for nonlinear measurement models and Huang et al. (2018) proposed a generalized score function for general SEM. The latter work is based on recent work in kernel-based measures of dependence (Gretton et al., 2005; Fukumizu et al., 2007; Zhang et al., 2011). Another line of work uses quantile scoring (Tagasovska et al., 2018). Also of relevance is the literature on nonparametric variable selection (Bertin and Lecué, 2008; Lafferty and Wasserman, 2008; Miller and Hall, 2010; Rosasco et al., 2013; Gregorová et al., 2018) and approaches based on neural networks (Feng and Simon, 2017; Ye and Sun, 2018; Abid et al., 2019). The main distinction between our work and previous work is that our framework is not tied to a specific model—as in Yu et al. (2019); Bühlmann et al. (2014); Park and Park (2018)—as our focus is on a *generic* formulation of an optimization problem that can be solved with *generic* solvers (see Section 3.2 for a more detailed comparison). This also distinguishes this work from concurrent work by Lachapelle et al. (2019) that focuses on neural network-based nonlinearities in the local conditional probabilities. Furthermore, compared to Huang et al. (2018) and Yu et al. (2019), our approach can be much more efficient. As such, we hope that this work is able to spur future work using more sophisticated nonparametric estimators and optimization schemes.

3.2.4 Comparison to existing approaches

It is instructive at this point to highlight the main distinction between our approach and existing approaches. A common approach is to assume the f_j are easily parametrized (e.g. linearity) (Zheng et al., 2018; Aragam et al., 2015; Gu et al., 2018; Park and Park, 2018; Park and Raskutti, 2017; Chen et al., 2019; Ghoshal and Honorio, 2017). In this case, one can easily encode the structure of G via, e.g. a weighted adjacency matrix, and learning G reduces to a parametric estimation problem. Nonparametric extensions of this approach include additive models (Bühlmann et al., 2014; Voorman et al., 2014), where the graph structure is easily deduced from the additive structure of the f_j. More recent work (Lachapelle et al., 2019; Yu et al., 2019) uses specific parametrizations via neural networks to encode G. An alternative approach relies on exploiting the conditional independence structure of $\mathbf{x}$, such as the post-nonlinear model (Zhang and Hyvärinen, 2009; Yu et al., 2019), the additive noise model (Peters et al., 2014), and kernel-based measures of conditional independence (Huang

et al., 2018). Our framework can be viewed as a substantial generalization of these approaches: We use partial derivatives to measure dependence in the *general* nonparametric model (3.1) without assuming a particular form or parametrization, and do not explicitly require any of the machinery of nonparametric conditional independence (although we note in some places this machinery is implicit). This allows us to use nonparametric estimators such as multilayer perceptrons and basis expansions, for which these derivatives are easily computed. As a result, the score-based learning problem is reduced to an optimization problem that can be tackled using existing techniques, making our approach easily accessible.

3.3 Characterizing acyclicity in nonparametric SEM

In this section, we discuss how to extend the trace exponential regularizer beyond the linear setting, and then discuss several special cases.

3.3.1 A notion of nonparametric acyclicity

Recall in the linear case, i.e. $g_j(s) = s$ and $f_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x}$ for some $\mathbf{w}_j \in \mathbb{R}^d$, we had the weighted adjacency matrix $W = [\mathbf{w}_1 | \cdots | \mathbf{w}_d] \in \mathbb{R}^{d \times d}$ of graph $G(f)$. Then we can formulate the entire problem in terms of W: If $L(W) = \|X - XW\|_F^2 / (2n)$, then optimizing $L(W)$ is equivalent to optimizing $L(f)$ over linear functions. Define the function $h(W) = \operatorname{tr} e^{W \circ W} - d$, where $[W \circ W]_{kj} = w_{kj}^2$, then (3.2) is equivalent to

$$\min_{W \in \mathbb{R}^{d \times d}} L(W) \quad \text{s.t. } h(W) = 0, \tag{3.3}$$

Our goal is to define a suitable surrogate of W for general nonparametric models, so that the same continuous program can be used to optimize (3.2).

Unfortunately, for general models of the form (3.1), there is no W, and hence the trace exponential formulation seems to break down. To remedy this, we use partial derivatives to measure the dependence of f_j on the kth variable, an idea that dates back to at least Rosasco et al. (2013). First, we need to make precise the spaces we are working on: Let $H^1(\mathbb{R}^d) \subset L^2(\mathbb{R}^d)$ denote the usual Sobolev space of square-integrable functions whose derivatives are also square integrable (for background on Sobolev spaces see Tsybakov (2003)). Assume hereafter that $f_j \in H^1(\mathbb{R}^d)$ and denote the partial derivative with respect to x_k by $\partial_k f_j$. It is then easy to show that f_j is independent of x_k if and only if $\|\partial_k f_j\|_{L^2} = 0$, where $\|\cdot\|_{L^2}$ is the usual L^2-norm. This observation implies that the following matrix precisely encodes the dependency structure amongst the x_j:

$$W(f) = W(f_1, \ldots, f_d) \in \mathbb{R}^{d \times d}, \quad [W(f)]_{kj} := \|\partial_k f_j\|_{L^2}. \tag{3.4}$$

Thus the program (3.2) is equivalent to

$$\min_{f : f_j \in H^1(\mathbb{R}^d), \forall j \in [d]} L(f) \quad \text{s.t. } h(W(f)) = 0. \tag{3.5}$$

This implies an equivalent continuous formulation of the program (3.2). Moreover, when the functions f_j are all linear, $W(f)$ is the same as the weighted adjacency matrix W. Thus, (3.5) is a genuine generalization of the linear case (3.3).

3.3.2 Special cases

In addition to applying to general nonparametric models of the form (3.2) and linear models, the program (3.5) applies to a variety of parametric and semiparametric models including additive noise models, generalized linear models, additive models, and index models. In this section we discuss these examples along with identifiability results for each case.

Additive noise models The nonparametric additive noise model (ANM) (Hoyer et al., 2008a; Peters et al., 2014) assumes that

$$x_j = f_j(\mathbf{x}) + z_j, \;\; z_j \sim \mathbb{P}_j, \;\; \mathbb{E}f_j(\mathbf{x}) = 0, \;\; z_j \perp f_j(\mathbf{x}). \tag{3.6}$$

Clearly this is a special case of (3.1) with $g_j(s) = s$. In contrast to the remaining examples below, without additional assumptions, it is not possible to simplify the condition for $[W(f)]_{kj} = 0$ in (3.4). Assuming the f_j are three times differentiable and not linear in any of its arguments, this model is identifiable (Peters et al., 2014, Corollary 31).

Generalized linear models A traditional GLM assumes that $\mathbb{E}[x_j|\mathbf{x}_{\mathrm{pa}(j)}] = g_j(\mathbf{w}_j^T\mathbf{x})$ for some known link functions $g_j : \mathbb{R} \to \mathbb{R}$ and $\mathbf{w}_j \in \mathbb{R}^d$. For example, we can use logistic regression for $x_j \in \{0, 1\}$ with $g_j(s) = e^s/(1+e^s)$. This is easily generalized to nonparametric mean functions $f_j \in H^1(\mathbb{R}^d)$ by setting

$$\mathbb{E}[x_j|\mathbf{x}_{\mathrm{pa}(j)}] = g_j(f_j(\mathbf{x})). \tag{3.7}$$

Clearly, (3.6) is a special case of (3.7). Furthermore, for linear mean functions, $[W(f)]_{kj} = 0$ if and only if $w_{jk} = 0$, recovering the parametric approach in the previous chapter. Several special cases of GLMs are known to be identifiable: Linear Gaussian with equal variances (Peters and Bühlmann, 2014), linear non-Gaussian models (Shimizu et al., 2006), Poisson models (Park and Park, 2019), and quadratic variance function models (Park and Raskutti, 2017).

Polynomial regression In polynomial regression, we assume that $f_j(\mathbf{x})$ is a polynomial in $x_1, \ldots, x_d$. More generally, given a known dictionary of functions $\eta_\ell(u_1, \ldots, u_d)$, we require that $f_j(\mathbf{x}) = \sum_\ell \beta_{j\ell}\, \eta_\ell(\mathbf{x})$. Then it is easy to check that $[W(f)]_{kj} = 0$ if and only if $\beta_{j\ell} = 0$ whenever η_ℓ depends on u_k. For each k, define $a_{jk}(u) := f_j(x_1, \ldots, x_{k-1}, u, x_{k+1}, \ldots, x_d)$. As long as $a_{jk}(u)$ is not a linear function (i.e. each f_j is a degree-2 polynomial or higher in x_k) for all k and j, then Corollary 31 in Peters et al. (2014) implies identifiability of this model.

Additive models In an additive model (Hastie and Tibshirani, 1987; Ravikumar et al., 2009), we assume that $f_j(\mathbf{x}) = \sum_{k \neq j} f_{jk}(x_k)$ for some $f_{jk} \in H^1(\mathbb{R})$. Then it is straightforward to show that $\|\partial_k f_j\|_{L^2} = 0$ if and only if $f_{jk} = 0$. In other words, $[W(f)]_{kj} = 0$ if and only if $\|f_{jk}\|_{L^2} = 0$. Assuming the f_{jk} are three times differentiable and not linear in any of its arguments, this model is identifiable (Peters et al., 2014, Corollary 31, see also Bühlmann et al., 2014).

Index models The multiple index model (Alquier and Biau, 2013; Yuan, 2011) assumes $f_j(\mathbf{x}) = \sum_{m=1}^{M} h_{jm}(\boldsymbol{\beta}_{jm}^T \mathbf{x})$ for some $h_{jm} \in H^1(\mathbb{R})$ and $\boldsymbol{\beta}_{jm} \in \mathbb{R}^d$. As long as M is sufficiently large, these functions are universal approximators (Diaconis and Shahshahani, 1984). When $M = 1$, this is known as a single-index model. As long as the functions h_{jm} $(m = 1, \ldots, M)$ are linearly independent, it is straightforward to show that $\|\partial_k f_j\|_{L^2} = 0$ if and only if $\beta_{jmk} = 0$ for each m. In other words, $[W(f)]_{kj} = 0$ if and only if $\sum_{m=1}^{M} \beta_{jmk}^2 = 0$. Once again, assuming three-times differentiability and nonlinearity of h_{jm}, Corollary 31 in Peters et al. (2014) implies identifiability of this model.

Among these examples, both polynomial regression and GLMs with linear mean function are nonlinear but finite-dimensional, and hence the problem (3.5) is straightforward to solve (see Section 3.4.3).

3.4 Optimization

In general, the program (3.5) is infinite-dimensional. In this section we discuss different ways to reduce this to a tractable, finite-dimensional optimization problem. One of the advantages of encoding dependence via $W(f)$ is that it provides a plug-and-play framework for plugging in various nonparametric estimators whose derivatives can be computed. We will illustrate two examples using multilayer perceptrons and orthogonal basis expansions, however, we emphasize that it is straightforward to implement other differentiable models for the f_j. These flexible nonparametric estimators will help reduce (3.5) to a straightforward optimization problem, as we discuss at the end of this section.

The basic recipe is the following:

1. Choose a model family for the conditional expectations $\mathbb{E}[x_j | \mathbf{x}_{\mathrm{pa}(j)}]$ (e.g. general nonparametric, additive, index, etc.);
2. Choose a suitable family of approximations (e.g. neural networks, orthogonal series, etc.);
3. Translate the loss function $L(f)$ and constraint $W(f)$ into parametric forms $L(\theta)$ and $W(\theta)$ using the approximating family;
4. Solve the resulting finite-dimensional problem.

Step 3 above is the key step that enables transforming (3.5) into a tractable optimization problem. By approximating the f_j with a flexible family of functions parametrized by θ, we can replace the infinite-dimensional quantity $W(f)$ with the simpler $W(\theta)$. As is standard in

the literature on nonparametric estimation, the dimension of θ is allowed to depend on n, although this dependence will be suppressed.

3.4.1 Multilayer perceptrons

We first consider the use of neural networks to approximate the f_j, as in an ANM (3.6) or GLM (3.7). Consider a multilayer perceptron (MLP) with h hidden layers and a single activation $\sigma : \mathbb{R} \to \mathbb{R}$, given by

$$\begin{aligned} &\mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)}) = \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(A^{(1)}u))), \\ &\quad A^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}, \quad m_0 = d. \end{aligned}$$

By increasing the capacity of the MLP (e.g. increasing the number of layers h or the number of hidden units m_ℓ in each layer), we can approximate any $f_j \in H^1(\mathbb{R}^d)$ arbitrarily well.

First, we must determine under what conditions $\mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)})$ is independent of u_k—this is important both for enforcing acyclicity and sparsity. It is not hard to see that if the kth column of $A^{(1)}$ consists of all zeros (i.e. $A^{(1)}_{bk} = 0$ for all $b = 1, \ldots, m_1$), then $\mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)})$ will be independent of u_k. In fact, we have the following proposition, which implies that this constraint precisely identifies the set of MLPs that are independent of u_k:

Proposition 4 *Consider the function class $\mathcal{F}$ of all MLPs that are independent of u_k and the function class $\mathcal{F}_0$ of all MLPs such that the kth column of $A^{(1)}$ consists of all zeros. Then $\mathcal{F} = \mathcal{F}_0$.*

Proof: For completeness, note that

$$\mathcal{F} = \{f \mid f(u) = \mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)}), f \text{ independent of } u_k\}$$

and

$$\mathcal{F}_0 = \{f \mid f(u) = \mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)}), A^{(1)}_{bk} = 0, \forall b = 1, \ldots, m_1\}.$$

We omit the bias terms in each layer as it does not affect the statement.

We will show that $\mathcal{F} \subseteq \mathcal{F}_0$ and $\mathcal{F}_0 \subseteq \mathcal{F}$.

(1) $\mathcal{F}_0 \subseteq \mathcal{F}$: for any $f_0 \in \mathcal{F}_0$, we have $f_0(u) = \mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)})$, where $A^{(1)}_{bk} = 0$ for all $b = 1, \ldots, m_1$. Hence the linear function $A^{(1)}u$ is independent of u_k. Therefore,

$$\begin{aligned} f_0(u) &= \mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)}) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(A^{(1)}u))) \end{aligned}$$

is also independent of u_k, which means $f_0 \in \mathcal{F}$.

(2) $\mathcal{F} \subseteq \mathcal{F}_0$: for any $f \in \mathcal{F}$, we have $f(u) = \mathsf{MLP}(u; A^{(1)}, \ldots, A^{(h)})$ and f is independent of u_k. We will show that $f \in \mathcal{F}_0$ by constructing a matrix $\tilde{A}^{(1)}$, such that

$$f(u) = \mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \ldots, A^{(h)}) \tag{3.8}$$

and $\tilde{A}^{(1)}_{bk} = 0$ for all $b = 1, \ldots, m_1$.

Let $\tilde{u}$ be the vector such that $\tilde{u}_k = 0$ and $\tilde{u}_{k'} = u_k$ for all $k' \neq k$. Since $\tilde{u}$ and u differ only on the kth dimension, and f is independent of u_k, we have

$$f(u) = f(\tilde{u}) = \mathsf{MLP}(\tilde{u}; A^{(1)}, \ldots, A^{(h)}). \tag{3.9}$$

Now define $\tilde{A}^{(1)}$ be the matrix such that $\tilde{A}^{(1)}_{bk} = 0$ and $\tilde{A}^{(1)}_{bk'} = A^{(1)}_{bk}$ for all $k' \neq k$. Then we have the following observation: for each entry $s \in \{1, \ldots, m_1\}$,

$$(\tilde{A}^{(1)}u)_s = \sum_{k'=1}^{d} \tilde{A}_{sk'}u_{k'} = \sum_{k' \neq k} A_{sk'}u_{k'} = \sum_{k'=1}^{d} A_{sk'}\tilde{u}_{k'} = (A^{(1)}\tilde{u})_s. \tag{3.10}$$

Hence,

$$\tilde{A}^{(1)}u = A^{(1)}\tilde{u}. \tag{3.11}$$

Therefore, by (3.9)

$$\begin{aligned} f(u) &= f(\tilde{u}) \\ &= \mathsf{MLP}(\tilde{u}; A^{(1)}, \ldots, A^{(h)}) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(A^{(1)}\tilde{u}))) \\ &= \sigma(A^{(h)}\sigma(\cdots A^{(2)}\sigma(\tilde{A}^{(1)}u))) \\ &= \mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \ldots, A^{(h)}) \end{aligned}$$

By definition of $\mathcal{F}_0$, we know that $\mathsf{MLP}(u; \tilde{A}^{(1)}, A^{(2)}, \ldots, A^{(h)}) \in \mathcal{F}_0$. Thus, $f \in \mathcal{F}_0$ and we have completed the proof. ■

This important proposition provides a rigorous way to enforce that an MLP approximation depends only on a few coordinates. Indeed, it is clear that constraining $A^{(1)}_{bk} = 0$ for each b will remove the dependence on k, however, there is a concern that we could lose the expressivity of multiple hidden layers in doing so. Fortunately, this proposition implies that there is in fact no loss of expressivity or approximating power. Furthermore, it follows that $[W(f)]_{kj} = 0$ if $\sum_b (A^{(1)}_{j,bk})^2 = 0$. This result enables us to characterize acyclicity independent of the depth of the neural network, as opposed to handling individual paths through the entire neural network as in Lachapelle et al. (2019), which depends linearly on the depth.

Let $\theta_j = (A^{(1)}_j, \ldots, A^{(h)}_j)$ denote the parameters for the jth MLP and $\theta = (\theta_1, \ldots, \theta_d)$. Define $[W(\theta)]_{kj} = [\sum_b (A^{(1)}_{j,bk})^2]^{1/2}$. The problem (3.2) is now reduced to

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{n}\sum_{j=1}^{d} \ell(X_{\cdot j}, \mathsf{MLP}(X; \theta_j)) + \lambda\|\theta_{j,1}\|_{1,1} \\ \text{s.t.} \quad & h(W(\theta)) = 0. \end{aligned} \tag{3.12}$$

3.4.2 Basis expansions

As an alternative to neural networks, we also consider the use of orthogonal basis expansions (Schwartz, 1967; Wahba, 1981; Hall, 1987; Efromovich, 1999). While many techniques are valid here, we adopt an approach based on Ravikumar et al. (2009). Let $\{\phi_r\}_{r=1}^{\infty}$ be an orthonormal basis of $H^1(\mathbb{R}^d)$ such that $\mathbb{E}\phi_r(X) = 0$ for each r. Then any $f \in H^1(\mathbb{R}^d)$ can be written uniquely

$$f(u) = \sum_{r=1}^{\infty} \alpha_r \phi_r(u), \quad \alpha_r = \int_{\mathbb{R}^d} \phi_r(u) f(u)\, du. \tag{3.13}$$

As long as the coefficients α_r decay sufficiently fast, f can be well-approximated by the finite series $\widehat{f}^R := \sum_{r=1}^{R} \alpha_r \phi_r$. Similar claims are true for one-dimensional Sobolev functions, which applies to both additive (i.e. for f_{jk}) and index (i.e. for h_{jm}) models.

We illustrate here an application with additive models and one-dimensional expansions. It is straightforward to extend these ideas to more general models using a tensor product basis, though this quickly becomes computationally infeasible. For more on high-dimensional orthogonal series, see Lee and Izbicki (2016). Thus,

$$f_j(u_1, \ldots, u_d) = \sum_{k \neq j} f_{jk}(u_k) = \sum_{k \neq j} \sum_{r=1}^{\infty} \alpha_{jkr} \phi_r(u_k). \tag{3.14}$$

Given integers R_k and assuming f_{jk} is sufficiently smooth, we have $\|f_{jk} - \widehat{f}_{jk}^{R_k}\|_{L^2} = O(1/R_k)$ (Efromovich, 1999), so that the overall approximation error is on the order $O(d/\min_k R_k)$. Furthermore, $[W(f)]_{kj} = 0 \iff \|f_{jk}\|_{L^2} = 0 \iff \alpha_{jkr} = 0$ for all r. Since we are discarding terms for $r > R_k$, in practice it suffices to check that $\alpha_{jkr} = 0$ for $r = 1 \ldots, R_k$, or $\sum_{r=1}^{R_k} \alpha_{jkr}^2 = 0$.

Letting θ denote the parameters α_{jkr} for all j, k, r, it thus suffices to define $[W(\theta)]_{kj} = [\sum_{r=1}^{R_k} \alpha_{jkr}^2]^{1/2}$ for the purposes of checking acyclicity. Let Φ_k be the matrix $[\Phi_k]_{ir} = \phi_r(X_k^{(i)})$. To estimate the coefficients α_{jkr}, we solve

$$\begin{aligned} \min_{\substack{1 \leq j,k \leq d \\ k \neq j}} \min_{a_{jk} \in \mathbb{R}^{R_k}} \quad & \frac{1}{n} \sum_{j=1}^{d} \ell\Big(X_{\cdot j}, \sum_{k \neq j} \Phi_k a_{jk}\Big) + \lambda_1 \sum_{k \neq j} \frac{1}{n} a_{jk}^T \Phi_k^T \Phi_k a_{jk} + \lambda_2 \sum_{k \neq j} \|a_{jk}\|_1 \\ \text{s.t.} \quad & h(W(\theta)) = 0. \end{aligned} \tag{3.15}$$

This is similar to Ravikumar et al. (2009) with the addition of an explicit ℓ_1 constraint.

3.4.3 Solving the continuous program

Having converted $L(f)$ and $W(f)$ to their finite-dimensional counterparts, we are now ready to solve (3.5) by applying standard optimization techniques. We emphasize that the hard work went into formulating the programs (3.12) and (3.15) as generic problems for which

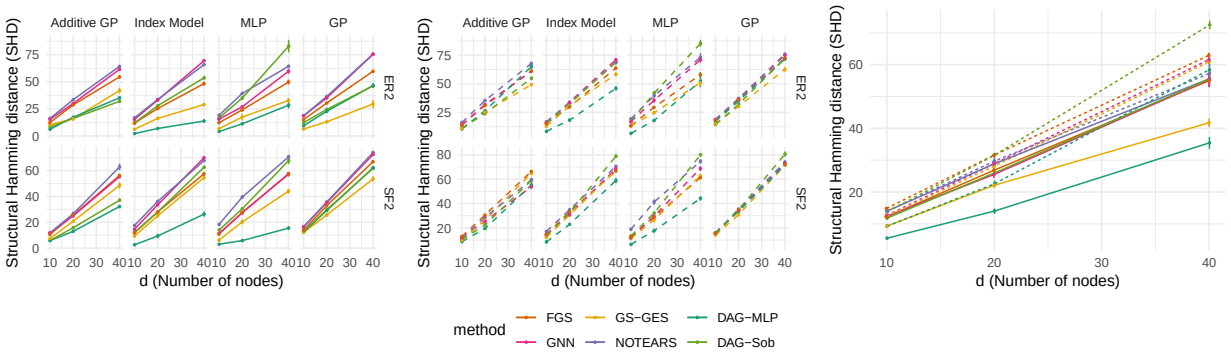


Figure 3.1: Structure recovery measured by SHD (lower is better) to ground truth. Left: $n = 1000$. Middle: $n = 200$. Right: Average over all configurations. Rows: random graph model (Erdos-Renyi and scale-free). Columns: different types of SEM. DAG-MLP performs well on a wide range of settings, while DAG-Sob shows good accuracy on additive models.

off-the-shelf solvers can be used. Importantly, since in both (3.12) and (3.15) the term $W(\theta)$ is differentiable w.r.t. θ, the optimization program is an ℓ_1-penalized smooth minimization under a differentiable equality constraint. As before, the standard machinery of augmented Lagrangian can be applied, resulting in a series of unconstrained problems:

$$\min_{\theta} \; F(\theta) + \lambda \left\|\theta\right\|_1, \quad F(\theta) = L(\theta) + \frac{\rho}{2}|h(W(\theta))|^2 + \alpha h(W(\theta)) \tag{3.16}$$

where ρ is a penalty parameter and α is a dual variable.

A number of optimization algorithms can be applied to the above *unconstrained* ℓ_1-penalized smooth minimization problem. A natural choice is the L-BFGS-B algorithm (Byrd et al., 1995), which can be directly applied by casting (3.16) into a box-constrained form:

$$\min_{\theta} \; F(\theta) + \lambda \left\|\theta\right\|_1 \quad \Longleftrightarrow \quad \min_{\theta^+ \geq 0, \theta^- \geq 0} F(\theta^+ - \theta^-) + \lambda \mathbf{1}^T(\theta^+ + \theta^-) \tag{3.17}$$

where $\mathbf{1}$ is a vector of all ones. We note that as in the linear case, (3.16) is a nonconvex program, and at best can be solved to stationarity. Our experiments indicate that this nonetheless leads to competitive and often superior performance in practice.

3.5 Experiments

We study the empirical performance of two instances of the general framework: MLP (3.4.1) and Sobolev basis expansion (3.4.2), denoted as DAG-MLP and DAG-Sob respectively. For DAG-MLP we use an MLP with one hidden layer with 10 hidden units and sigmoid activation function. For DAG-Sob we use Sobolev basis $\phi_r(u) = s_r \sin(u/s_r)$, $s_r = 2/((2r-1)\pi)$ for $r = 1, \ldots, 10$.

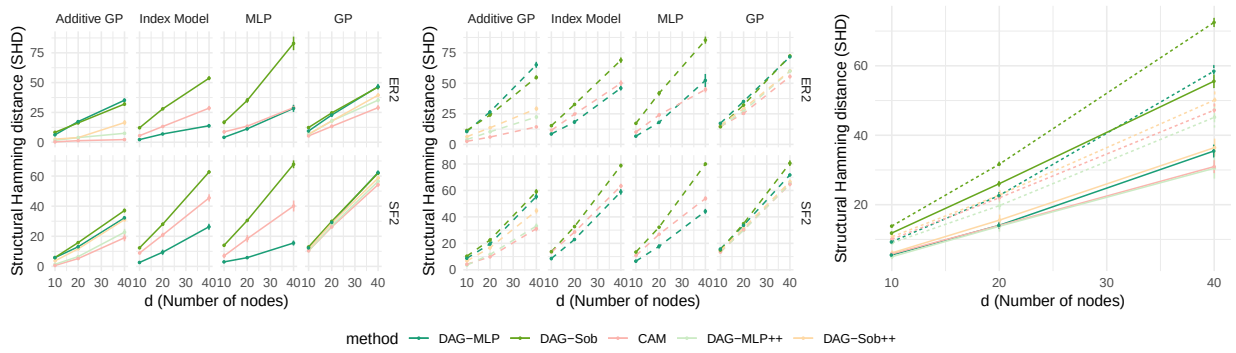


Figure 3.2: Structure recovery measured by SHD (lower is better) to ground truth. Left: $n = 1000$. Middle: $n = 200$. Right: Average over all configurations. Rows: random graph model (Erdos-Renyi and scale-free). Columns: different types of SEM. Either DAG-MLP or DAG-MLP++ (i.e. DAG-MLP with neighborhood selection and pruning) achieves competitive accuracy compared to CAM.

Baselines We consider the following baselines.

- Fast greedy equivalence search (FGS)[2] (Ramsey et al., 2017) is based on greedy search and assumes linear dependency between variables.
- Greedy equivalence search with generalized scores (GSGES)[3] (Huang et al., 2018) is also based on greedy search, but uses generalized scores without assuming a particular model class.
- DAG-GNN (GNN)[4] (Yu et al., 2019) learns a (noisy) nonlinear transformation of a linear SEM using neural networks.
- NOTEARS (Linear)[5] (Chapter 2) learns a linear SEM using continuous optimization.
- Causal additive model (CAM)[6] (Bühlmann et al., 2014) learns an additive SEM by leveraging efficient nonparametric regression techniques and greedy search over edges.

For all experiments, default parameter settings are used, except for CAM where both preliminary neighborhood selection and pruning are applied.

Simulation Given the graph G, we simulate the SEM $x_j = f_j(\mathbf{x}_{\mathrm{pa}(j)}) + z_j$ for all $j \in [d]$ in the topological order induced by G. We consider the following instances of f_j:

- Additive GP: $f_j(\mathbf{x}_{\mathrm{pa}(j)}) = \sum_{k \in \mathrm{pa}(j)} f_{jk}(x_k)$, where each f_{jk} is a draw from Gaussian process with RBF kernel with length-scale one.
- Index model: $f_j(\mathbf{x}_{\mathrm{pa}(j)}) = \sum_{m=1}^{3} h_m(\sum_{k \in \mathrm{pa}(j)} \theta_{jmk} x_k)$, where $h_1 = \tanh$, $h_2 = \cos$, $h_3 = \sin$, and each θ_{jmk} is drawn uniformly from range $[-2, -0.5] \cup [0.5, 2]$.

[2] https://github.com/bd2kccd/py-causal
[3] https://github.com/Biwei-Huang/Generalized-Score-Functions-for-Causal-Discovery/
[4] https://github.com/fishmoon1234/DAG-GNN
[5] https://github.com/xunzheng/notears
[6] https://cran.r-project.org/package=CAM

- MLP: f_j is a randomly initialized MLP with one hidden layer of size 100 and sigmoid activation.
- GP: f_j is a draw from Gaussian process with RBF kernel with length-scale one.

In all settings, z_j is i.i.d. standard Gaussian noise.

Metrics We evaluate the estimated DAG structure using the following common metrics: false discovery rate (FDR), true positive rate (TPR), false positive rate (FPR), and structural Hamming distance (SHD). Note that both FGS and GSGES return a CPDAG that may contain undirected edges, in which case we evaluate them favorably by assuming correct orientation for undirected edges whenever possible.

3.5.1 Structure learning

In this experiment we examine the structure recovery of different methods by comparing the DAG estimates against the ground truth. We simulate {ER1, ER2, ER4, SF1, SF2, SF4} graphs with $d = \{10, 20, 40\}$ nodes. For each graph, $n = \{1000, 200\}$ data samples are generated. The above process is repeated 10 times and we report the mean and standard deviations of the results. For DAG-MLP and DAG-Sob, $\lambda = \{0.01, 0.03\}$ are used for $n = \{1000, 200\}$ respectively.

Figure 3.1 shows the SHD in various settings; the complete set of results for the remaining metrics are deferred to the supplement. Overall, the proposed DAG-MLP method attains the best SHD (lower the better) across a wide range of settings, particularly when the data generating mechanism is an MLP or an index model. One can also observe that the performance of DAG-MLP stays stable for different graph types with varying density and degree distribution, as it does not make explicit assumptions on the topological properties of the graph such as density or degree distribution. Not surprisingly, DAG-Sob performs well when the underlying SEM is additive GP. On the other hand, when the ground truth is not an additive model, the performance of DAG-Sob degrades as expected. Finally, we observe that GSGES outperforms DAG-MLP and DAG-Sob on GP, which is a nonparametric setting in which a kernel-based dependency measure can excel, however, we note that the kernel-based approach accompanies an $O(n^3)$ time complexity, compared to linear dependency on n in DAG-MLP and DAG-Sob. For instance, the average runtime of GSGES on ER2 with $d = 40$, $n = 1000$ is over 90 minutes, whereas DAG-MLP takes less than five minutes on average. Also, with by properly tuning the regularization parameter, the performance of DAG-MLP for each individual setting can be improved considerably, for example in the GP setting. Since such hyperparameter tuning is not the main focus of this work, we fix a reasonable λ for all settings (see Appendix 3.5.4 for more discussion on runtime and hyperparameters).

Figure 3.2 shows the SHD compared with CAM. We first observe that DAG-MLP outperforms CAM in multiple index models and MLP models, on the other hand, CAM achieves better accuracy on additive GP and the full GP setting. Recall that the CAM algorithm involves three steps: 1) Preliminary neighborhood search (PNS), 2) Order search by greedy optimization of the likelihood, and 3) Edge pruning. By comparison, our methods effectively

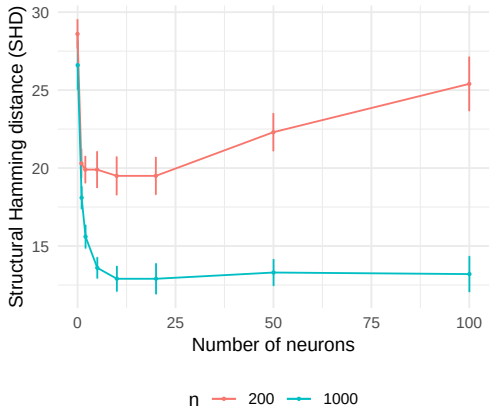


Figure 3.3: SHD (lower is better) with varying hidden layer size in DAG-MLP.

only perform the second step, and can easily be pre- and post-processed with the first (PNS) and third (edge pruning) steps. To further investigate the efficacy of these additional steps, we applied both preliminary neighborhood selection and edge pruning to DAG-MLP and DAG-Sob on additive GP and GP settings, denoted as DAG-MLP++ and DAG-Sob++. Noticeably, the output from PNS simply translates to a set of constraints in the form of $\theta_j = 0$ that can be easily incorporated into the L-BFGS-B algorithm for (3.17), demonstrating the flexibility of the proposed approach. The performance improves in both settings, matching or improving upon CAM.

3.5.2 Sensitivity to number of hidden units

We also investigated the effect of number of hidden units in the DAG-MLP estimate. It is well-known that as the size of the hidden layer increases, the functions representable by an MLP become more flexible. On the other hand, larger networks require more samples to estimate the parameters. Indeed, Figure 3.3 confirms this intuition. We plot the SHD with varying number of hidden units ranging from zero (*i.e.* linear function) to 100 units, using $n = 1000$ and $n = 200$ samples generated from the additive GP model on SF2 graph with $d = 20$ nodes. One can first observe a sharp phase transition between zero and very few hidden units, which suggests the power of nonlinearity. Moreover, as the number of hidden units increases to 20, the performance for both $n = 1000$ and $n = 200$ steadily improves, in which case the increased flexibility brings benefit. However, as we further increase the number of hidden units, while SHD for $n = 1000$ remains similar, the SHD for $n = 200$ deteriorates, hinting at the lack of samples to take advantage of the increased flexibility.

3.5.3 Real data

Finally, we evaluated DAG-MLP on a real dataset from Sachs et al. (2005) that is commonly used as a benchmark as it comes with a *consensus network* that is accepted by the biological

	DAG-MLP	DAG-Sob	FGS	Linear	GNN	GSGES
$d = 20$	92.12 ± 22.51	62.90 ± 16.83	0.55 ± 0.43	10.95 ± 4.52	498.32 ± 43.72	1547.42 ± 109.83
$d = 40$	282.64 ± 67.46	321.88 ± 57.33	0.59 ± 0.17	43.15 ± 12.43	706.35 ± 64.49	6379.98 ± 359.67

Table 3.1: Runtime (in seconds) of various algorithms on ER2 graph with $n = 1000$ samples.

community. The dataset consists of $n = 7466$ continuous measurements of expression levels of proteins and phospholipids in human immune system cells for $d = 11$ cell types. We report an SHD of 16 with 13 edges estimated by DAG-MLP. In comparison, NOTEARS predicts 16 edges with SHD of 22 and GNN predicts 18 edges that attains SHD of 19. (Due to the large number of samples, we could not run GSGES on this dataset.) Among the 13 edges predicted by DAG-MLP, 7 edges agree with the consensus network: raf $\to$ mek, mek $\to$ erk, PLCg $\to$ PIP2, PIP3 $\to$ PLCg, PIP3 $\to$ PIP2, PKC $\to$ mek, PKC $\to$ jnk; and 3 edges are predicted but in a reversed direction: raf $\leftarrow$ PKC, akt $\leftarrow$ erk, p38 $\leftarrow$ PKC. Among the true positives, 3 edges are not found by other methods: mek $\to$ erk, PIP3 $\to$ PLCg, PKC $\to$ mek.

3.5.4 Additional results

Full comparison We show {SHD, FDR, TPR, FPR} results on all {ER1, ER2, ER4, SF1, SF2, SF4} graphs in Figure 3.4, 3.5, 3.6, 3.7 respectively. Similarly, see Figure 3.8, 3.9, 3.10, 3.11 for full comparison with CAM. As in Figure 3.1, each row is a random graph model, each column is a type of SEM. Overall DAG-MLP has low FDR/FPR and high TPR, and same for DAG-Sob on additive GP. Also observe that in most settings GNN has low FDR as well as low TPR, which is a consequence of only predicting a small number of edges.

Runtime comparison Table 3.1 contains runtime comparison of different algorithms on ER2 graph with $n = 1000$ samples. Recall that the kernel-based approach of GSGES comes with a $O(n^3)$ computational complexity, whereas DAG-MLP and DAG-Sob has $O(n)$ dependency on n. This can be confirmed from the table, which shows GSGES has a significantly longer runtime.

Comments on hyperparameter tuning The experiments presented in this work were conducted under a fixed (and therefore suboptimal) value of λ and weight threshold across all graph types, sparsity levels, and SEM types, despite the fact that each configuration may prefer different regularization strengths. Indeed, we observe substantially improved performance by choosing different values of hyperparameters in some settings. As our focus is not on attaining the best possible accuracy in all settings by carefully tuning the hyperparameters, we omit these results in the main text and only include here as a supplement. For instance, for ER4 graph with $d = 40$ variables and $n = 200$ samples, when the SEM is additive GP and MLP, setting $\lambda = 0.03$ and threshold $= 0.5$ gives results summarized in Table 3.2.

SEM	Method	SHD	FDR	TPR	FPR	Predicted #
Additive-GP	DAG-MLP	124.3 ± 6.65	0.30 ± 0.07	0.35 ± 0.04	0.04 ± 0.01	81.70 ± 10.49
	GSGES	121.3 ± 5.02	0.36 ± 0.05	0.28 ± 0.03	0.04 ± 0.00	69.30 ± 5.01
MLP	DAG-MLP	88.40 ± 11.29	0.18 ± 0.08	0.57 ± 0.06	0.03 ± 0.02	111.70 ± 15.97
	GSGES	121.60 ± 11.95	0.33 ± 0.09	0.33 ± 0.06	0.04 ± 0.01	77.10 ± 7.13

Table 3.2: ER4, $d = 40$, $n = 200$ with $\lambda = 0.03$ and threshold $= 0.5$.

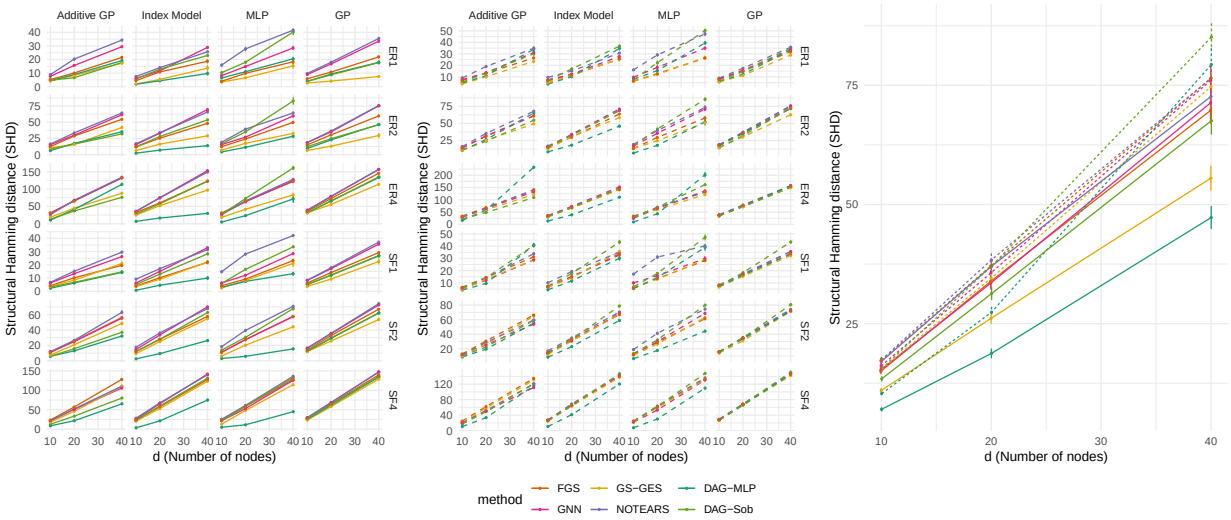


Figure 3.4: Structure recovery measured by SHD (lower is better) to ground truth.

3.6 Discussion

We present a framework for score-based learning of sparse directed acyclic graphical models that subsumes many popular parametric, semiparametric, and nonparametric models as special cases. The key technical device is a notion of nonparametric acyclicity that leverages partial derivatives in the algebraic characterization of DAGs. With a suitable choice of the approximation family, the estimation problem becomes a finite-dimensional differentiable program that can be solved by standard optimization algorithms. The resulting continuous optimization algorithm updates the entire graph (i.e. all edges simultaneously) in each iteration using global information about the current state of the network, as opposed to traditional local search methods that update one edge at a time based on local information. Notably, our approach is generally more efficient and more accurate than existing approaches, despite relying on generic algorithms. This out-of-the-box performance is desirable, especially when noting that future improvements and specializations can be expected to improve the approach substantially.

We would like to report a practical issue as well: When choosing MLP as the approximation class, it is possible that even if the k-th column of the first layer $A^{(1)}$ is close to zero, the subsequent layers of MLP can be highly non-smooth to the extent that the dependence between x_k and $f(\mathbf{x})$ is amplified away from zero. Therefore in practice it is often desired to

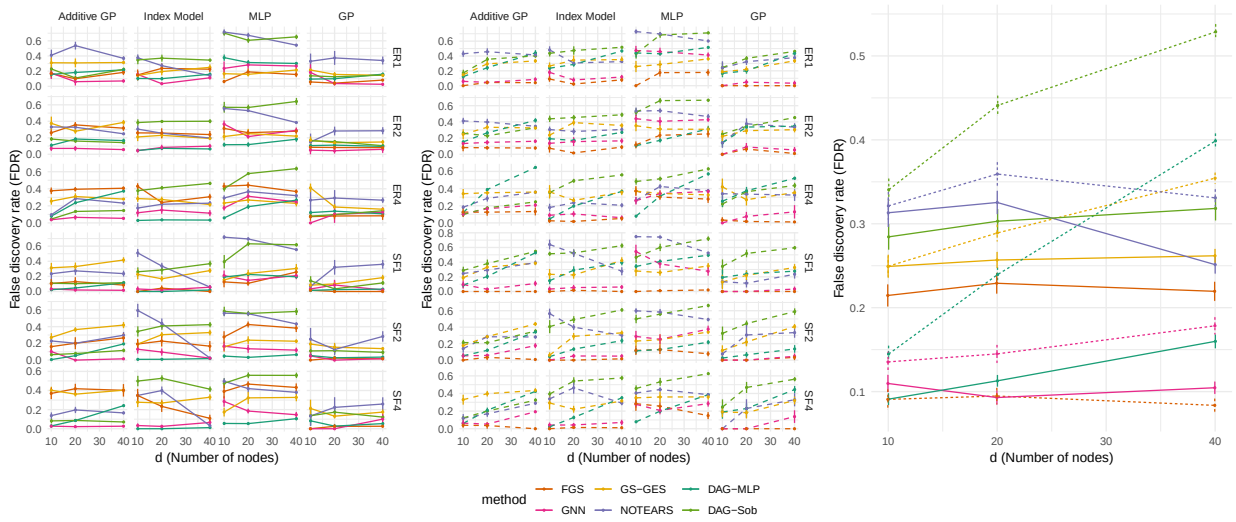


Figure 3.5: Structure recovery measured by FDR (lower is better) to ground truth.

restrict the function class further, for instance by restricting the depth of MLP to be small, or having strong regularization parameters.

An interesting direction for future work is to study the nonconvex landscape of (3.16) in order to provide rigorous guarantees on the optimality of the solutions found by our approach. It would also be interesting to study alternative loss functions (e.g. the maximum mean discrepancy loss as in (Goudet et al., 2018)) as well as more specialized optimization algorithms for solving (3.16).

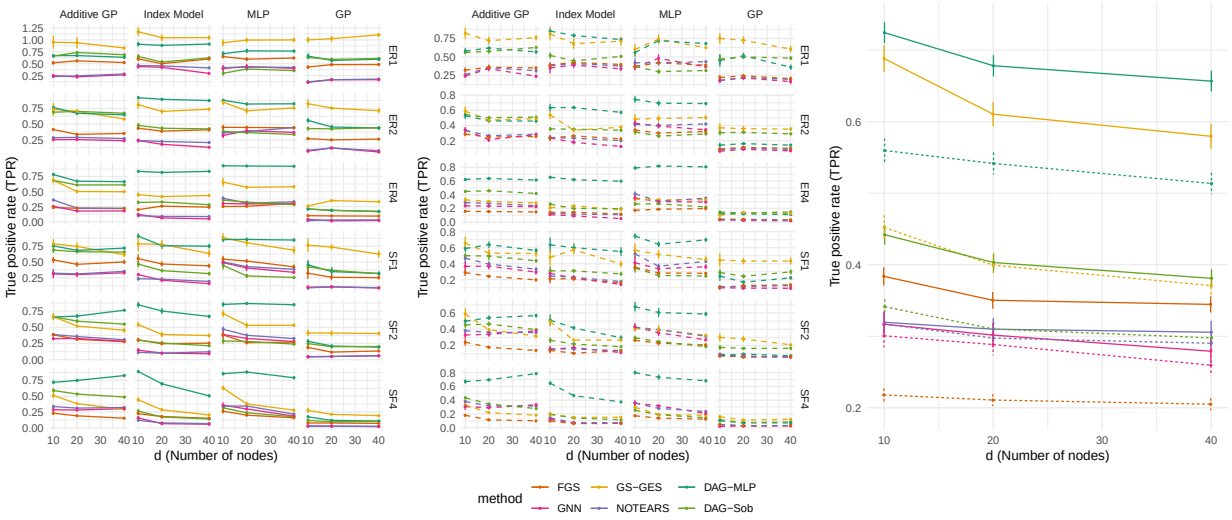


Figure 3.6: Structure recovery measured by TPR (higher is better) to ground truth.

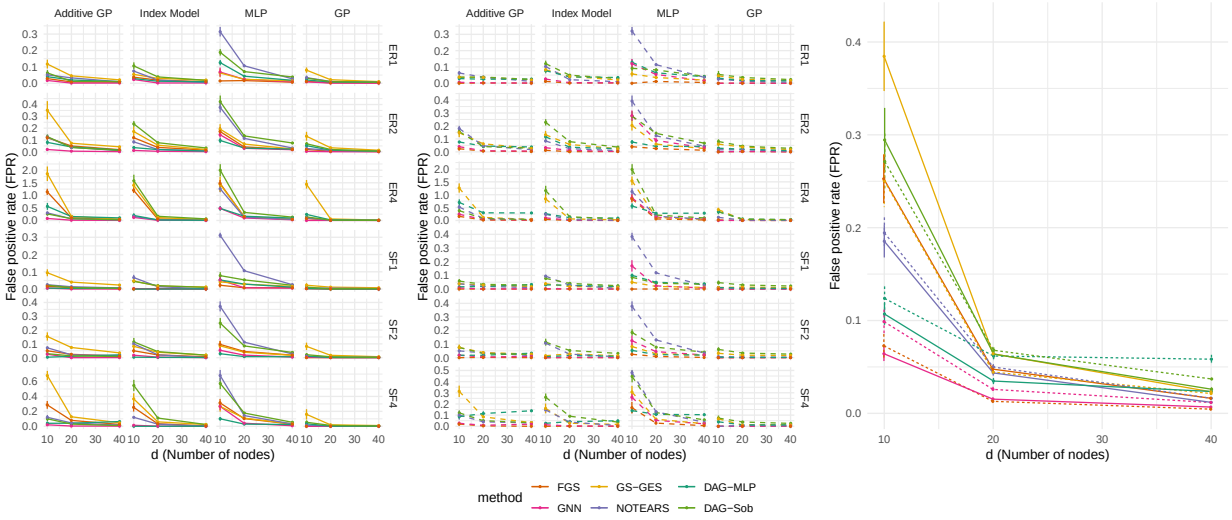


Figure 3.7: Structure recovery measured by FPR (lower is better) to ground truth.

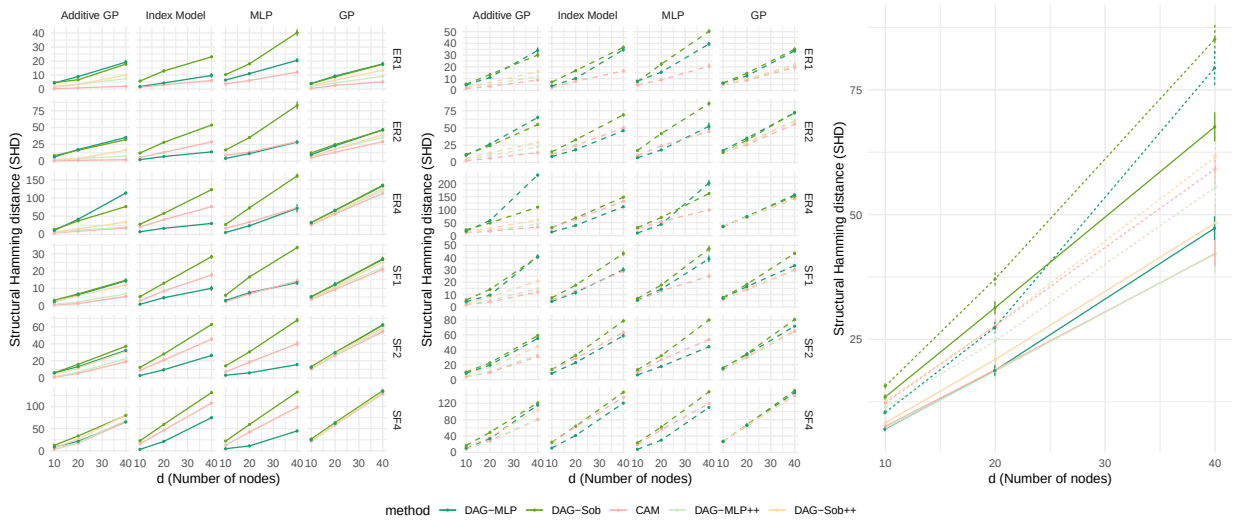


Figure 3.8: Structure recovery measured by SHD (lower is better) to ground truth, compared with CAM.

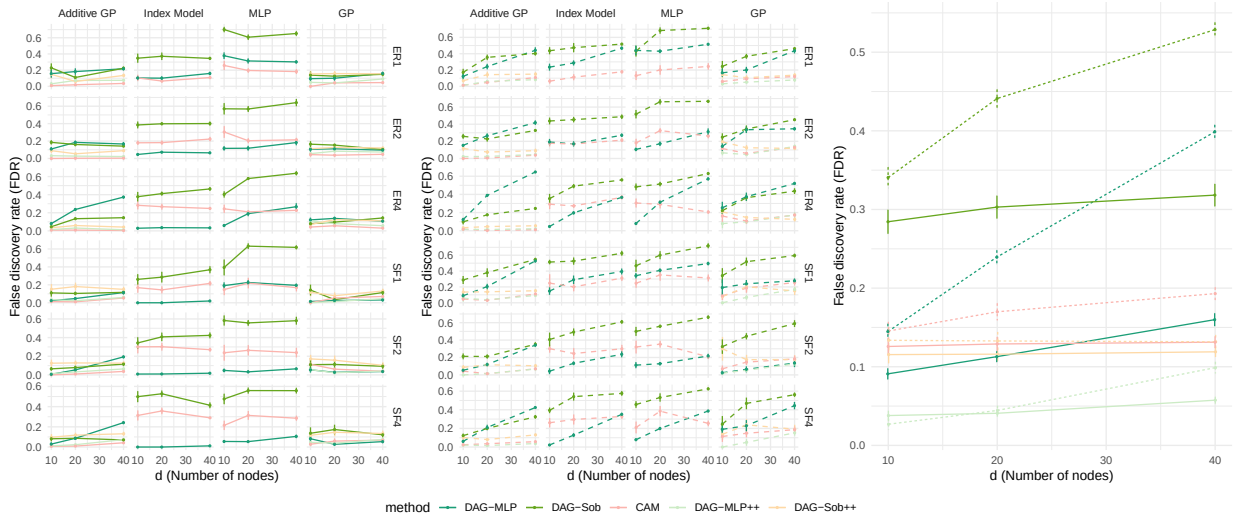


Figure 3.9: Structure recovery measured by FDR (lower is better) to ground truth, compared with CAM.

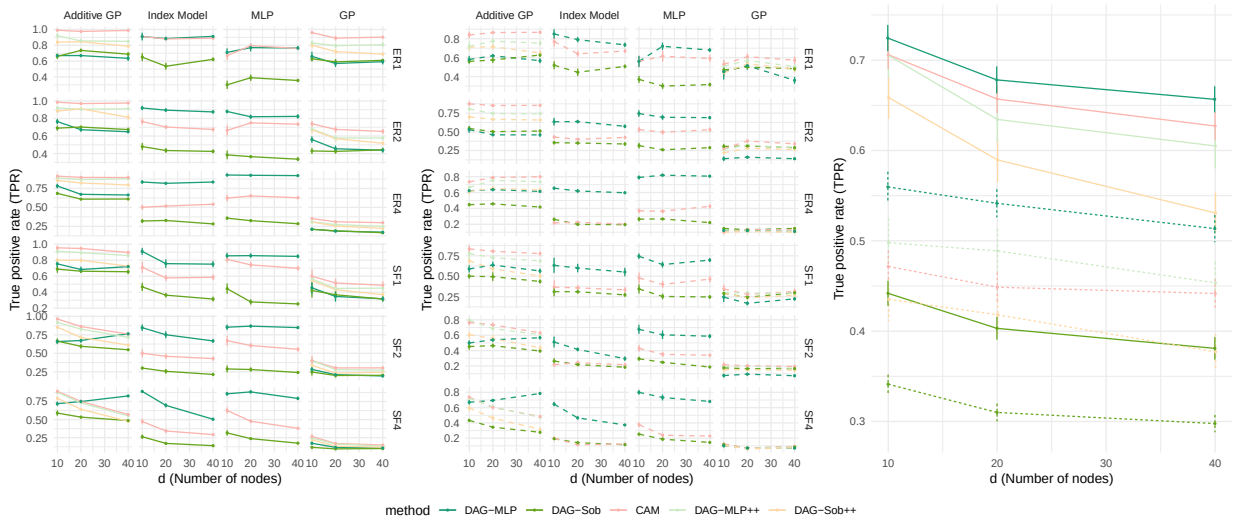


Figure 3.10: Structure recovery measured by TPR (higher is better) to ground truth, compared with CAM.

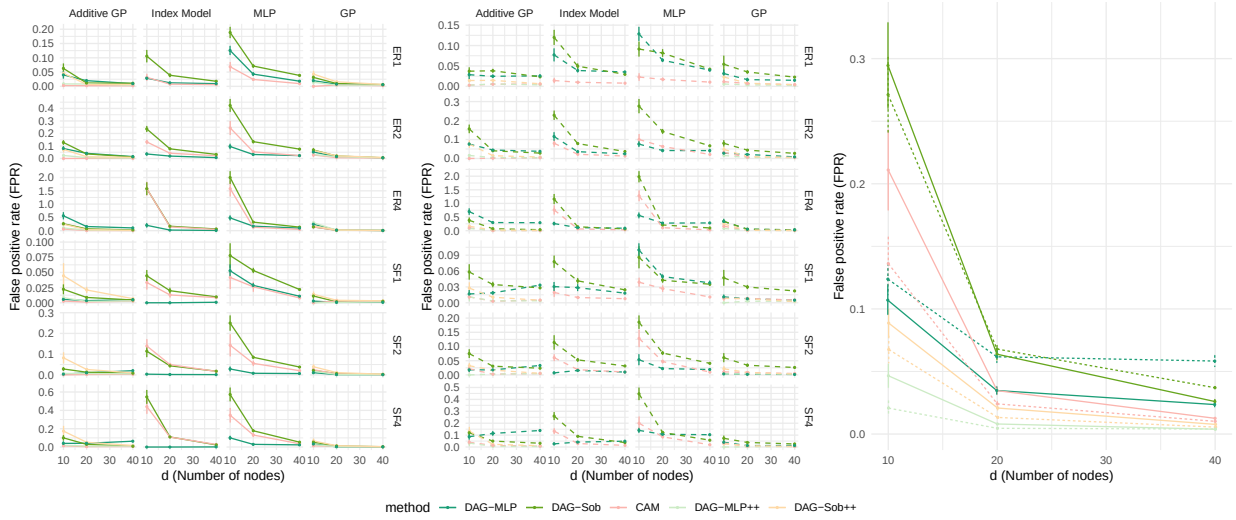


Figure 3.11: Structure recovery measured by FPR (lower is better) to ground truth, compared with CAM.

Chapter 4

Learning Non-Gaussian DAGs with Continuous Optimization

So far we have discussed linear Bayesian networks and nonlinear Bayesian networks. In this chapter, let's zoom into a special subclass of linear Bayesian networks, namely the linear non-Gaussian acyclic models (LiNGAM). Unlike general linear Bayesian networks which typically can only identified up to Markov equivalence class, LiNGAM can be uniquely identified from observations without assuming faithfulness as in PC (Spirtes and Glymour, 1991) or GES (Chickering, 2002). Because of this nice property, it has been extended to many other scenarios such as latent variable models (Hoyer et al., 2008b), cyclic Bayesian networks (Lacerda et al., 2008), and time series models (Hyvärinen et al., 2010).

In addition to the the identifiability result, Shimizu et al. (2006) introduces a two-step algorithm ICA-LiNGAM that consistently recovers the true graph given infinite samples. However, in practice, its performance with finite data can be far from being optimal. The initial step, ICA (Hyvärinen et al., 2001) may suffer from multiple sources of error: 1) statistical error from finite data; 2) suboptimality of local solution in nonconvex problems; and 3) numerical error in iterative algorithms. As we discuss in detail below, the error in ICA can potentially mislead the subsequent steps, resulting in suboptimal solution.

Motivated by the recent development in continuous optimization for DAG learning, we explore a single-step optimization algorithm for the ICA-LiNGAM, and show that it has certain advantages over the original two-step approach.

4.1 Our contributions

- We frame ICA-LiNGAM as a single continuous optimization problem, hence jointly performing ICA and permutation search.
- This algorithm falls into the NOTEARS framework, requiring only minimal change.
- The proposed algorithm can be easily extended to have structural regularization (e.g. sparsity) and can incorporate prior knowledge.

4.2 Background

We introduce some background on the non-Gaussian models and related literature.

4.2.1 Two variables: How to distinguish cause from effect?

The linear SEM for two variables (x, y) can be conveniently expressed in one line:

$$y = \beta x + e \tag{4.1}$$

where β is the weight coefficient on x and e is the noise term with an important property $x \perp\!\!\!\perp e$. The problem of structure learning is then essentially reduced to distinguishing between the two models $x \to y$ and $x \leftarrow y$, given i.i.d. samples $(x, y) \sim p(x, y)$.

Noticing the familiar linear regression form, one might conjecture that the correct regression direction will lead to an estimate of the noise $\hat{e}$ that is independent of the corresponding input variable x. Indeed, this intuition holds true under non-Gaussianity assumption. More precisely, if at most one of x and e is Gaussian, only in the true direction the regression residual will be statistically independent of the input variable.

This can be more easily seen from examples. Figure 4.1 and Figure 4.2 plots two examples with the same underlying graph: $x \to y$. The only difference is that Figure 4.1 has non-Gaussian x and e, whereas in Figure 4.2 both x and e are drawn from standard Gaussian distribution. Notice that in any ground truth model, in any regression direction, the regression residual is *uncorrelated* with the input variable by definition. However, in the non-Gaussian case (Figure 4.1), the regression residual is *independent* of the input variable only in the correct direction. For the incorrect direction, it is clear that the value of y is indicative of the range of the residual r_{yx}, hence they are statistically dependent. On the other hand, recall that under Gaussianity, uncorrelatedness implies independence. Hence in the Gaussian case (Figure 4.2), both directions will conclude that the residual is independent of the input, making them indistinguishable from this standpoint.

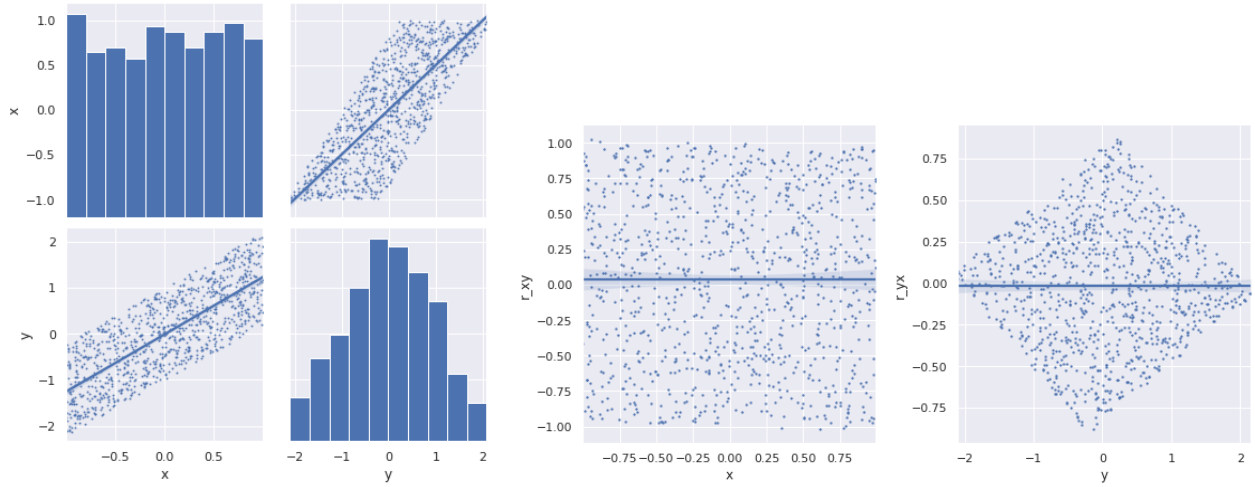
4.2.2 Beyond two variables: Linear non-Gaussian acyclic model (LiNGAM)

More generally, the linear SEM for d variables $\mathbf{x} \in \mathbb{R}^d$ is given by:

$$\mathbf{x} = B\mathbf{x} + \mathbf{e} \tag{4.2}$$

where B is the adjacency matrix[1] of the underlying DAG, whose (j, i)-th entry β_{ji} represents the coefficient from x_i to x_j, and importantly the noise variables $\mathbf{e} = (e_1, \ldots, e_d)$ are mutually independent.

[1]Note that we use matrix B to denote the adjacency matrix in this chapter instead of W in previous chapters. Moreover the edge direction has changed from $j \to i$ to $j \leftarrow i$, i.e. $B = W^T$. This change of notation is to follow the convention in the ICA and LiNGAM literature.



(a) Scatter plot of (x, y) (b) Correct: $x \to y$ (c) Incorrect: $x \leftarrow y$

Figure 4.1: Data generated from linear SEM $y = 1.2x + e$, where $x \sim \text{Uniform}(-1, 1)$ and $e \sim \text{Uniform}(-1, 1)$. In both regression directions, regression residual is uncorrelated with the input variable. However, only the true direction has regression residual *independent* of the input variable.

Rearranging the above equation, we can express the random variables $\mathbf{x}$ as a linear transformation of independent noise terms $\mathbf{e}$:

$$\mathbf{x} = (I - B)^{-1}\mathbf{e} = A\mathbf{e} \tag{4.3}$$

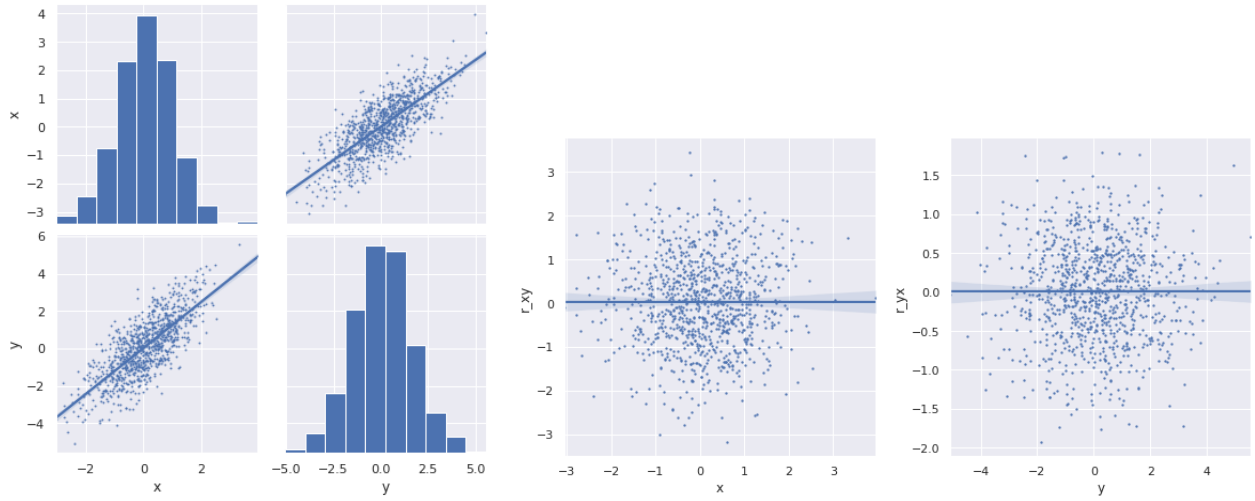
where $A = (I - B)^{-1}$ is also known as the total effect matrix, whose (j, i)-th entry a_{ji} represents the accumulated effect size of the noise variable e_i to the random variable x_j.

Notice that the system $\mathbf{x} = A\mathbf{e}$ forms an instance of the independent component analysis (ICA) (Hyvärinen et al., 2001) model, if the noise terms follow non-Gaussian distributions. Based on this observation, the identifiability of the linear non-Gaussian acyclic model (LiNGAM) can be established:

Theorem 2 (Identifiability of LiNGAM (Shimizu et al., 2006)) *If* $\mathbf{x}$ *follows linear acyclic SEM* (4.2) *where at most one of the independent noise variables* $\mathbf{e} = (e_1, \ldots, e_d)$ *is Gaussian, then* B *is uniquely identifiable from the joint distribution* $p(\mathbf{x})$.

The identifiability result is a direct consequence of the identifiability of ICA (Hyvärinen et al., 2001), which is in turn based on the Darmois-Skitovich theorem (Darmois, 1953; Skitovich, 1953) on the characterization of independence of linear combinations of random variables. Noticeably, it does not require faithfulness assumption as in many previous methods (Spirtes and Glymour, 1991; Chickering, 2002).

Not only it is interesting from a theoretical perspective, the identifiability result also suggests an ICA-based algorithm to recover the structure of B, known as ICA-LiNGAM (Shimizu et al., 2006), described in Algorithm 3. The algorithm proceeds in two steps. First, given samples $\mathbf{x}^{(1)}, \ldots \mathbf{x}^{(n)}$, it estimates the mixing matrix $\hat{A}$ (or equivalently the demixing matrix



(a) Scatter plot of (x, y) (b) Correct: $x \to y$ (c) Incorrect: $x \leftarrow y$

Figure 4.2: Data generated from linear SEM $y = 1.2x + e$, where $x \sim N(0, 1)$ and $e \sim N(0, 1)$. In both regression directions, regression residual is uncorrelated with the input variable. However, by Gaussianity, both directions have residual independent of the input variable, hence they are indistinguishable.

Algorithm 3 ICA-LiNGAM algorithm (Shimizu et al., 2006)

1. Estimate demixing matrix $\hat{W} = \hat{A}^{-1}$ for ICA model $\mathbf{x} = A\mathbf{e}$.
2. Find the unique row permutation of $\hat{W}$ such that $\text{diag}(\text{perm}(\hat{W})) \neq 0$.
3. Rescale each row of $\hat{W}$ such that $\text{diag}(\text{scale}(\text{perm}(\hat{W}))) = I$.
4. Return $\hat{B} = I - \text{scale}(\text{perm}(\hat{W}))$.

$\hat{W} = \hat{A}^{-1}$) for the ICA model $\mathbf{x} = A\mathbf{e}$. Since $W = I - B$ by definition, one might be tempted to directly estimate B by $\hat{B} = I - \hat{W}$, however due to permutation and scaling indeterminacy in ICA, W is only known up to row permutation and scaling. Fortunately, Shimizu et al. (2006) shows that there is a unique row permutation of $\hat{W}$ such that all diagonal elements are nonzero, which is necessary for $\hat{B}$ to be a DAG. Hence in the second step we find such permutation and corresponding row scaling of $\hat{W}$, and finally estimate the adjacency matrix $\hat{B}$. This procedure is shown to be consistent in the population setting.

4.2.3 Practical issues with finite data

With finite data, however, each step above may suffer from statistical or numerical error. Therefore in practice Algorithm 3 requires some modifications and additional steps in order to be applicable. For an illustration, consider the following toy example with three variables:

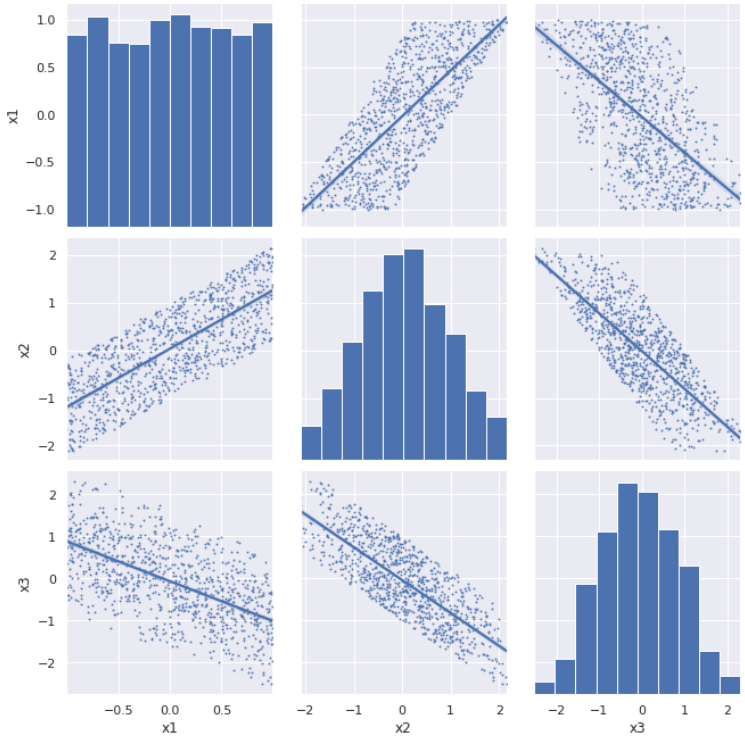


Figure 4.3: Data generated from linear SEM $x_1 = e_1$, $x_2 = 1.2x_1 + e_2$, $x_3 = -0.8x_2 + e_3$, where $e_1, e_2, e_3 \sim \text{Uniform}(-1, 1)$.

$x_1 \to x_2 \to x_3$, with ground truth adjacency matrix

$$B = \begin{pmatrix} 0 & 0 & 0 \\ 1.2 & 0 & 0 \\ 0 & -0.8 & 0 \end{pmatrix} \tag{4.4}$$

and each noise variable $e_j \sim \text{Uniform}(-1, 1)$. We first simulate $n = 1000$ samples from the ground truth model, shown in Figure 4.3.

A typical FastICA (Hyvärinen et al., 2001) estimate of the demixing matrix is[2]

$$\hat{W} = \begin{pmatrix} -0.0548 & -0.0001 & -0.0001 \\ 0.0031 & -0.046 & -0.0564 \\ -0.0663 & 0.0532 & -0.0009 \end{pmatrix} \tag{4.5}$$

Notice that $\hat{W}$ contains many small values close to zero. This is not only due to numerical error in the optimization algorithm but also because of the inevitable statistical error when learning with finite data. Since the matrix is fully dense, any row permutation will lead to nonzero diagonal. To resolve this issue, in practice ICA-LiNGAM performs bipartite

[2]Note that due to nonconvexity of ICA and randomness of the optimization algorithm, different runs may lead to different results. Nonetheless, in most cases they are just row permutations of the matrix shown here.

matching using for instance the Hungarian algorithm to find the row permutation that leads to the *most significantly* nonzero diagonals:

$$\min_{\text{all row permutations}} \quad \sum_j \frac{1}{\left|[\text{perm}(\hat{W})]_{jj}\right|} \tag{4.6}$$

In this example, the best row permutation is given by $\pi = (1, 3, 2)$, i.e. switching the last two rows:

$$\text{perm}(\hat{W}) = \begin{pmatrix} -0.0548 & -0.0001 & -0.0001 \\ -0.0663 & 0.0532 & -0.0009 \\ 0.0031 & -0.046 & -0.0564 \end{pmatrix} \tag{4.7}$$

We can now obtain unit diagonals by rescaling each row by its diagonal entry:

$$\text{scale}(\text{perm}(\hat{W})) = \begin{pmatrix} 1 & 0.0024 & 0.0026 \\ -1.2457 & 1 & -0.0164 \\ -0.0542 & 0.8145 & 1 \end{pmatrix} \tag{4.8}$$

The initial estimate for the adjacency matrix is:

$$\hat{B} = I - \text{scale}(\text{perm}(\hat{W})) = \begin{pmatrix} 0 & -0.0024 & -0.0026 \\ 1.2457 & 0 & 0.0164 \\ 0.0542 & -0.8145 & 0 \end{pmatrix} \tag{4.9}$$

However, notice that even though the estimate $\hat{B}$ appears to be close to the ground truth B in Frobenius norm, however the support of $\hat{B}$ is far from being a DAG, due to many small entries in $\hat{B}$. Hence, in practice ICA-LiNGAM further prunes edges with small weights until $\hat{B}$ becomes a DAG. In this example, the entries in the upper triangular part of $\hat{B}$ can be removed, which leads to a DAG estimate:

$$\text{threshold}(\hat{B}) = \begin{pmatrix} 0 & 0 & 0 \\ 1.2457 & 0 & 0 \\ 0.0542 & -0.8145 & 0 \end{pmatrix} \tag{4.10}$$

Although the thresholded estimate of the adjacency matrix satisfies the acyclicity, it still contains an insignificant edge $x_1 \to x_3$, which is absent in the true model. To further prune out these edges, in practice ICA-LiNGAM adopts an additional step of significance test, e.g. the Wald test, for each estimated edge. The final pruned estimate is

$$\text{prune}(\text{threshold}(\hat{B})) = \begin{pmatrix} 0 & 0 & 0 \\ 1.2457 & 0 & 0 \\ 0 & -0.8145 & 0 \end{pmatrix} \tag{4.11}$$

which correctly recovers the support of the ground truth B, and the recovered parameters are close to the true values.

As we have seen, even with $n = 1000$ samples, which is typically considered sufficient for $d = 3$ variables, many practical modifications and additional steps are required to make the algorithm applicable. Even worse is when the sample size is small, in which case the statistical error in ICA may be so large that later steps cannot correct from it. For instance, with smaller sample size $n = 100$, a bad estimate given by FastICA is[3]

$$\hat{W}_{n=100} = \begin{pmatrix} -0.1431 & -0.0253 & 0.0131 \\ 0.1128 & -0.204 & -0.1276 \\ -0.198 & 0.0287 & -0.1183 \end{pmatrix} \tag{4.12}$$

which leads to an estimate of the adjacency matrix:

$$\hat{B}_{n=100} = \begin{pmatrix} 0 & 0 & 0 \\ 0.5529 & 0 & -0.6255 \\ -1.6735 & 0 & 0 \end{pmatrix} \tag{4.13}$$

Clearly, this estimate with $n = 100$ is significantly different from the ground truth, and more importantly such error in ICA cannot be easily corrected by subsequent pruning and significance tests as in the large sample case.

Similar observation have been made for other two-step methods in the literature (Margaritis and Thrun, 1999; Tsamardinos et al., 2006; Schmidt et al., 2007; Pellet and Elisseeff, 2008). Albeit not based on ICA, these methods first estimates a superstructure such as a moralized graph or a skeleton, and then prune or orient the initial estimate. It has been argued (Huang et al., 2013; Xiang and Kim, 2013) that these methods are prone to error, since they are built on the assumption that the output of the early steps does not mislead future steps. In practice, this is often violated due to inevitable statistical error from finite data, let alone potential suboptimality of the local solution in nonconvex problems.

4.2.4 Previous works

It is not surprising that constraint-based learning methods such as PC (Spirtes and Glymour, 1991) output a Markov equivalence class – the set of graphs that satisfy the same conditional independence relations, since the algorithm is based on detecting conditional independencies in the data, which also explains the necessity of the faithfulness assumption. Many score-based learning methods like GES (Chickering, 2002) also work with Markov equivalence class under similar assumption, albeit for a seemingly different reason: they optimize score-equivalent functions that attain the same value for all graphs in the Markov equivalence class.

A number of studies have been on reducing the output space while relaxing the assumptions. One salient line of works started from the discovery of LiNGAM (Shimizu et al., 2006), where a unique DAG can be identified from observations without making faithfulness assumption. At the core of this result is the identifiability of ICA (Hyvärinen et al., 2001), which in turn follows

[3]This is by no means a typical case. Indeed, there are many more correct ICA estimates than this failure case. However, we would like to point out such error can mislead subsequent steps of ICA-LiNGAM without being corrected. In contrast, our method presented later is able to correctly estimate B on the same dataset.

from the Darmois-Skitovich theorem (Darmois, 1953; Skitovich, 1953) on the independence of linear combinations of random variables. The identifiability result of LiNGAM paved road for many subsequent works on various extensions, including latent variable models (Hoyer et al., 2008b), cyclic models (Lacerda et al., 2008), and time series models (Hyvärinen et al., 2010), just to name a few. By solving the original LiNGAM problem, we would like to improve these subsequent extensions as well.

On the algorithmic side, motivated by the ICA theory, the original discovery algorithm for LiNGAM (Shimizu et al., 2006) employs ICA as a subroutine. Later, various algorithms that do not rely on ICA have been proposed for LiNGAM. For instance, DirectLiNGAM (Shimizu et al., 2011) iteratively finds the root node in the DAG by performing regression for all variable pairs and then independence test between residuals and the input variable, in a similar manner the Kanh's algorithm work for topological sort. Pairwise DirectLiNGAM (Hyvärinen and Smith, 2013) follows the same approach, but identifies the root by pairwise likelihood ratio instead of independence test. Note that both DirectLiNGAM and Pairwise DirectLiNGAM are multi-step methods, in a sense that a mistake in the early root-finding step can significantly affect later steps without being able to be corrected. We focus on ICA-LiNGAM since it can be converted into a single optimization program as we show below.

4.3 Single-step ICA-LiNGAM

In this section, we describe how to solve ICA-LiNGAM in a single step.

4.3.1 Maximum likelihood for ICA

For completeness, let us first consider the vanilla ICA model $\mathbf{x} = A\mathbf{e}$. Let $\tilde{p}_j(e)$ be the density of the j-th independent source, and denote $W = A^{-1}$ as the demixing matrix. Since the data is a linear transformation of independent sources, the ICA negative log-likelihood for a single data point $\mathbf{x}$ can be written as

$$L_{\text{ICA}}(W; \mathbf{x}) \coloneqq -\log p(\mathbf{x}) = -\log p(\mathbf{e}) + \log|\det A| \tag{4.14}$$

$$= -\log \prod_{j=1}^{d} \tilde{p}_j(e_j) + \log|\det A| \tag{4.15}$$

$$= \sum_{j=1}^{d} -\log \tilde{p}_j([W\mathbf{x}]_j) - \log|\det W| \tag{4.16}$$

whose gradient is given by

$$\nabla L_{\text{ICA}}(W; \mathbf{x}) = -g(W\mathbf{x})\mathbf{x}^T - W^{-T} \tag{4.17}$$

where $g(\mathbf{s}) = (g_1(s_j), \ldots, g_d(s_d))$ is a vector-valued function with elements defined as the sensitivity of the log-density:

$$g_j(s) \coloneqq (\log \tilde{p}_j(s))' = \frac{\tilde{p}_j'(s)}{\tilde{p}_j(s)} \tag{4.18}$$

In order to perform gradient-based optimization, we need to choose the form of the function g_j, which in turn defines our choice of the independent source distribution. On the other hand, it has been shown that the maximum likelihood estimate is locally consistent even in the presence of small misspecification error on $\tilde{p}_j$ (Amari et al., 1997). Hence even without knowing the exact form of the true $\tilde{p}_j$, approximate distribution families can be used for g_j and $\tilde{p}_j$, as long as the model and the truth are both supergaussian or both subgaussian.

A classical choice for supergaussian distribution is given by

$$g_j(s) = -2\tanh(s) \tag{4.19}$$

$$\log \tilde{p}_j(s) = -2\log\cosh(s) + \text{constant} \tag{4.20}$$

The corresponding distribution $\tilde{p}_j$ is also known as the hyperbolic secant, which is a supergaussian distribution. It is also used in the original Infomax (Bell and Sejnowski, 1995) for blind source separation.

Another choice of g_j leads to a subgaussian distribution:

$$g_j(s) = \tanh(s) - s \tag{4.21}$$

$$\log \tilde{p}_j(s) = \log\cosh(s) - \frac{s^2}{2} + \text{constant} \tag{4.22}$$

We refer to Hyvärinen et al. (2001) for a more extensive discussion on the choice of independent component distribution and furthermore how to adaptively decide on the approximation family.

4.3.2 Maximum likelihood for LiNGAM

Having established the likelihood of ICA, we can write down the negative log-likelihood of LiNGAM w.r.t. B:

$$L(B; \mathbf{x}) \coloneqq -\log p(\mathbf{x}; B) = \sum_{j=1}^{d} -\log \tilde{p}_j(x_j - \boldsymbol{\beta}_j^T \mathbf{x}) - \log|\det(I - B)| \tag{4.23}$$

$$= \langle -\log \tilde{p}(\mathbf{x} - B\mathbf{x}), \mathbf{1}\rangle - \log|\det(I - B)| \tag{4.24}$$

where $\mathbf{1}$ is the vector of all ones. The gradient is given by

$$\nabla L(B; \mathbf{x}) = g(\mathbf{x} - B\mathbf{x})\mathbf{x}^T + (I - B)^{-T} \tag{4.25}$$

The choice of the density $\tilde{p}_j$ often assumes unit variance of independent sources in ICA, which does not hold for the noise variable e_j for general SEM. Therefore a normalized

likelihood for standardized noise variable e_j/σ_j is preferable, as noted in (Hyvärinen et al., 2010):

$$\tilde{L}(B;\mathbf{x}) = \sum_{j=1}^{d} -\log \tilde{p}_j\Big(\frac{x_j - \boldsymbol{\beta}_j^T\mathbf{x}}{\hat{\sigma}_j}\Big) + \log \hat{\sigma}_j - \log|\det(I-B)| \tag{4.26}$$

where $\sigma_j^2 = \mathrm{Var}(e_j)$ and its empirical version is $\hat{\sigma}_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j - \boldsymbol{\beta}_j^T\mathbf{x})^2$.

Let $X \in \mathbb{R}^{n\times d}$ be the collection of n samples. Using the smooth characterization of DAGs (Zheng et al., 2018), we can express the maximum likelihood estimation problem of ICA-LiNGAM (left) as a single optimization program (right):

$$\begin{array}{ll} \min\limits_{B} & \tilde{L}(B;X) \\ \text{s.t.} & G(B) \in \mathsf{DAGs} \end{array} \iff \begin{array}{ll} \min\limits_{B} & \tilde{L}(B;X) \\ \text{s.t.} & h(B) = 0 \end{array} \tag{4.27}$$

4.3.3 Optimization

The original Infomax (Bell and Sejnowski, 1995) for ICA uses stochastic gradient descent for the objective (4.16). Over the years, a number of more advanced algorithms for ICA have been proposed. For instance, instead of performing gradient descent in the Euclidean space of parameters, one can perform steepest descent in the Fisher metric. The resulting algorithm, namely natural gradient descent, is known to converge faster than ordinary gradient descent. Another algorithm is the FastICA (Hyvärinen et al., 2001), which is regarded as one of the state-of-the-art algorithms for ICA. It can be viewed as a fixed-point update for the maximum likelihood estimate.

In maximum likelihood for ICA-LiNGAM, however, we would like to solve a constrained optimization problem. Since the augmented Lagrangian algorithm solves a sequence of regularized subproblem, it is not straightforward to directly apply advanced ICA algorithms in our framework. Nonetheless, we demonstrate that the simple L-BFGS algorithm that uses the Euclidean gradient works well enough in practice. We leave the improvement on the optimization algorithm for future work.

4.4 Experiments

In this section, we demonstrate some experimental results on simulated dataset. Each experiment is repeated for 20 random instantiations, and we report the box plot as the aggregated result.

Dataset We simulate the data from the LiNGAM model $\mathbf{x} = B\mathbf{x} + \mathbf{e}$. Two types of random graph models are used: Erdös-Rényi(ER) and scale-free (SF). In ER graphs, edges are instantiated independently with the same probability p. The SF graph is generated according to the preferential attachment process in (Barabási and Albert, 1999). Given the graph structure, the edge weights β_{ji} are drawn uniformly from $(-2,-0.5)\cup(0.5,2)$. We use four different noise non-Gaussian distributions:

- Exponential distribution: $e_j \sim \text{Exp}(1)$
- Laplace distribution: $e_j \sim \text{Laplace}(0, 1)$
- Gumbel distribution: $e_j \sim \text{Gumbel}(0, 1)$
- Uniform distribution: $e_j \sim \text{Uniform}(-1, 1)$

All of the distributions are supergaussian, except for the last one which is subgaussian. For each experiment, we simulated $n = 50, 100, 200, 500, 1000$ samples of $\mathbf{x}$.

Proposed method If not explicitly mentioned, we choose $g_j(s) = -2\tanh(s)$ as the nonlinearity. The resulting algorithm is named NOTEARS-ICA. In all settings, we use an ℓ_1 regularization with parameter $\lambda_1 = 0.1$.

Baselines We use ICA-LiNGAM (Shimizu et al., 2006) and DirectLiNGAM (Shimizu et al., 2011) for comparison. In both methods, we use the default hyperparameter settings if there is any.

- ICA-LiNGAM is the original LiNGAM method that uses ICA to identify independent noise terms. We use the MATLAB implementation from the author's website: https://sites.google.com/site/sshimizu06/lingam.
- DirectLiNGAM is a new algorithm for LiNGAM that is based on simple regression and independence test instead of ICA. The implementation is available from the author's website: https://sites.google.com/site/sshimizu06/Dlingamcode.

Metrics We evaluate the structure recovery using two common graph metric: structural Hamming distance (SHD) and false discovery rate (FDR) against the ground truth graph. SHD is the total number of edge addition, removal, and reversals needed to convert one graph to another. FDR is defined as the fraction of incorrect edges among the predicted ones, where incorrect edges include extra edges and edges with opposite direction.

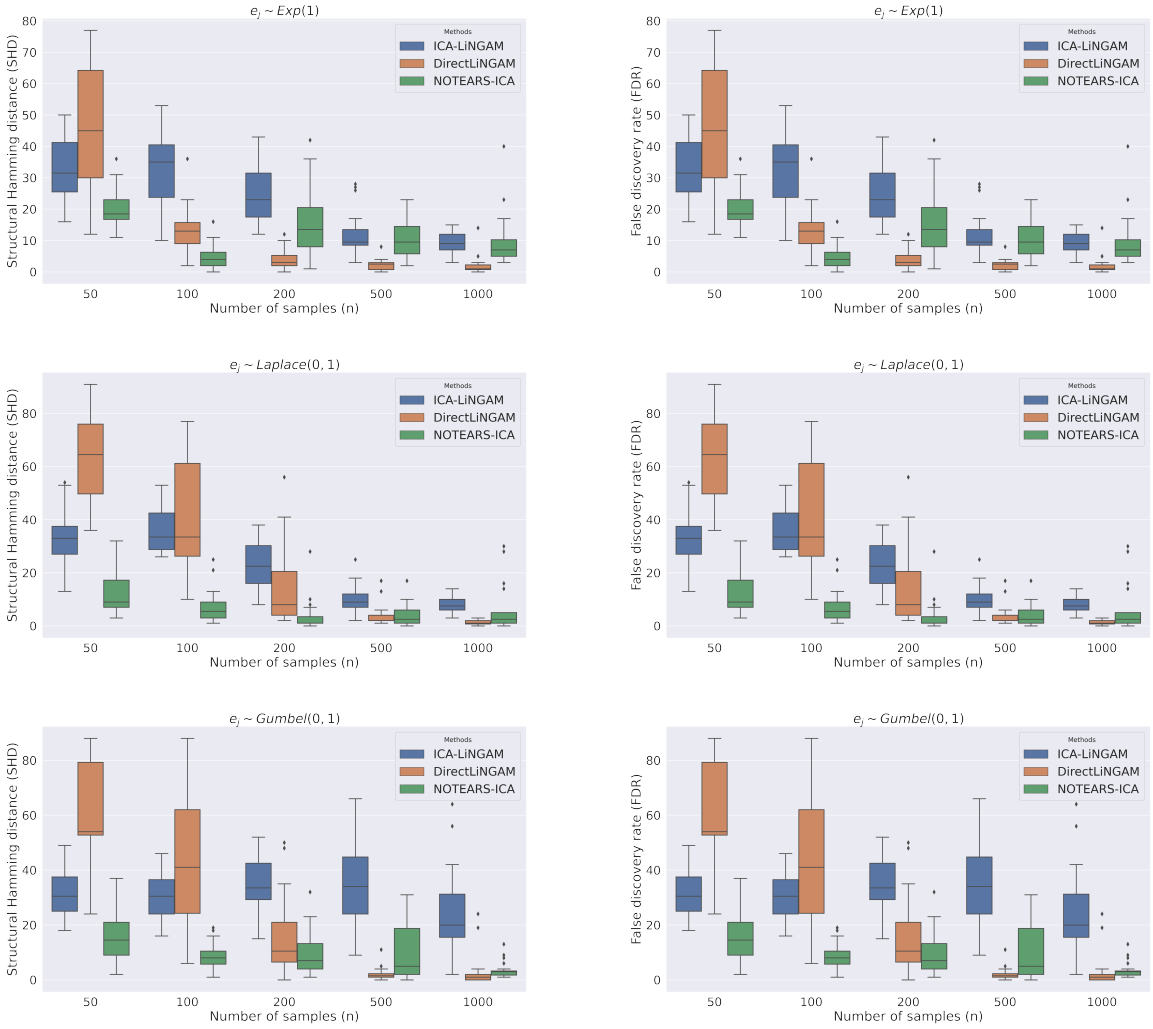


Figure 4.4: Structure recovery under different non-Gaussian noise distributions: Exponential, Laplace, and Gumbel. Lower SHD and FDR are better. The proposed NOTEARS-ICA (green) performs decently in general, especially when the sample size is small.

4.4.1 Structure recovery under different noise distributions

Figure 4.4 shows structure recovery result as the sample size increases from $n = 50$ to $n = 1000$, under different non-Gaussian noise distributions. In this experiment, we simulated ER graph with $d = 20$ variables and $s_0 = 40$ edges.

In all figures, the general trend is clear: the proposed NOTEARS-ICA performs well compared to baseline methods, especially when the sample size n is small, for instance when $n = 50$ or $n = 100$. This clearly shows the advantage of jointly solving both ICA and permutation search at the same time. By contrast, for both Exponential and Laplace noise ICA-LiNGAM is less accurate at the beginning but achieves good accuracy with sufficient

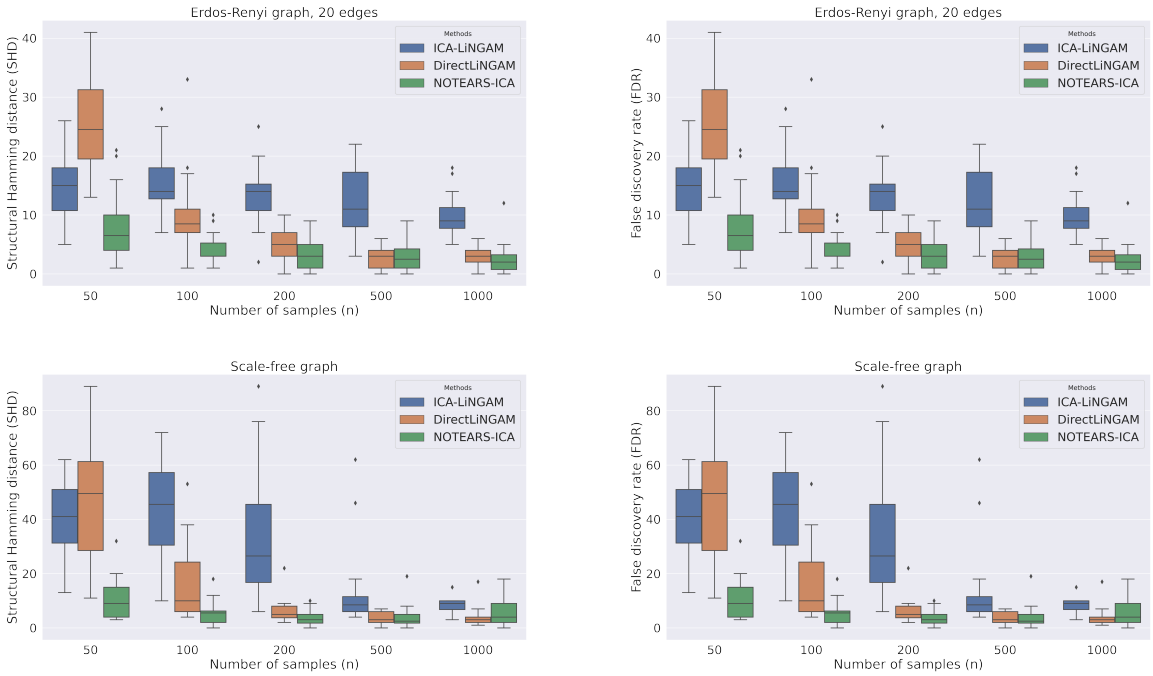


Figure 4.5: Structure recovery with different graph types: top row is ER graph with 20 edges, and bottom row is scale-free graph with 40 edges. Lower SHD and FDR are better. The proposed NOTEARS-ICA (green) performs stably for different graph types.

samples, although for Gumbel noise the improvement over increased sample size is not significant. On the other hand, DirectLiNGAM performs the best when supplied with large samples, however it also suffers the most when sample size is small, e.g. $n = 50$. We conjecture this is because the sensitivity of independence test with insufficient data. We would also like to point out that this shows a single nonlinearity $g_j(s) = -2\tanh(s)$ for approximate supergaussian family can successfully adapt to different types of non-Gaussian distributions. Overall, the proposed method exhibits nice performance especially with small sample size.

4.4.2 Structure recovery on different graph types

Figure 4.5 contains similar results but with different random graphs: sparse graphs and scale-free graphs. Both graphs use $e_j \sim \text{Exp}(1)$ as the non-Gaussian noise distribution.

In the top row we simulated a sparse ER graph with $d = 20$ variables and $s_0 = 20$ edges. We can observe that ICA-LiNGAM performance gradually improves as the sample size increases, however does not reach the ground truth even with $n = 1000$ samples. On the other hand, DirectLiNGAM can reach near optimal SHD against the ground truth in large sample regime. However, its performance suffer significantly when the sample size is small, e.g. when $n = 50$. In contrast, the proposed NOTEARS-ICA performs stably for all sample sizes.

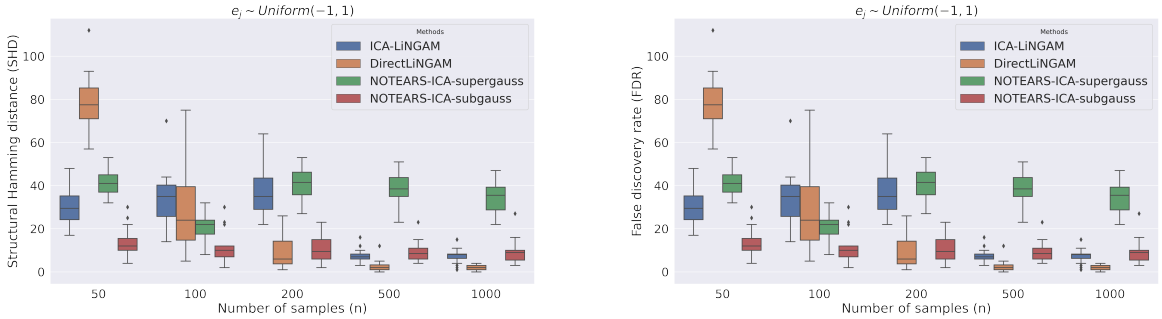


Figure 4.6: Structure recovery with misspecified noise distribution. Lower SHD and FDR are better. True noise follows Uniform distribution, which is subgaussian. Both baselines (blue and orange) perform well given enough samples. However, when the proposed model incorrectly assumes supergaussian noise (green), the performance degrades substantially. Good accuracy is achieved if we use subgaussian noise model (red) instead.

The bottom row shows results for scale-free graphs, with $d = 20$ variables and $s_0 = 40$ edges. This result can be contrasted with the top row in Figure 4.4, as they have the same graph density. Both ICA-LiNGAM and DirectLiNGAM perform worse when $n = 50$ than in the ER graph, whereas NOTEARS-ICA still performs decently in this setting. As the sample size increases, DirectLiNGAM quickly becomes accurate, and ICA-LiNGAM catches up at $n = 500$. Across all sample sizes, the proposed NOTEARS-ICA performs stably well.

4.4.3 Model misspecification: supergaussian vs subgaussian

We report a failure case in Figure 4.6. As before, we simulated ER graph with $d = 20$ and $s_0 = 40$ edges, however with a subgaussian noise distribution $e_j \sim \text{Uniform}(-1, 1)$. We experiment with two versions of NOTEARS-ICA, one assumes supergaussian noise (4.19) and the other assumes subgaussian noise (4.21).

We can first observe that both ICA-LiNGAM and DirectLiNGAM perform similar to previous experiments with supergaussian noise. This is expected because 1) ICA-LiNGAM often uses FastICA (Hyvärinen et al., 2001) algorithm as subroutine, which can adapt to supergaussian or subgaussian independent component distributions; and 2) DirectLiNGAM uses least squares regression and kernel-based nonparametric independence test, none of which makes any parametric assumptions about the noise distribution.

On the other hand, when the proposed method NOTEARS-ICA assumes supergaussian noise (4.19), the accuracy is substantially lower than other cases, and moreover does not improve with increased sample size. By contrast, if the model correctly assumes subgaussian noise (4.21), the structure recovery becomes accurate as before. To summarize, this shows that the algorithm can be affected by model misspecification.

We note a potential solution here. As suggested in Hyvärinen et al. (2001), if the noise

distribution cannot be determined a priori, one can empirically test the stability quantity

$$\mathbb{E}_s[sg_j(s) - g_j'(s)] > 0 \tag{4.28}$$

at each iteration, and change to the opposite distribution class if the sign flips. We leave it as a future work to adaptively choose the noise distribution types.

4.5 Discussion

In this chapter we presented a continuous optimization approach for the linear non-Gaussian SEM. This model class is interesting because the true graph can be uniquely identified from the joint distribution, i.e. observational data. We showed that original procedure of ICA followed by permutation-finding step can lead to unsatisfactory result due to the propagation of error from the first step to subsequent steps, especially when the sample size is small. Instead, we propose to perform joint optimization by expressing the steps in a single optimization algorithm. Moreover, the new algorithm is in fact a special case of the linear NOTEARS in Chapter 2, which makes implementation particularly straightforward. Experiments suggest that the proposed joint optimization algorithm outperforms in most cases, especially with fewer samples.

Aside from empirical performance, the proposed algorithm also has some advantages:

- It is easy to incorporate structure regularization such as the ℓ_1 penalty while respecting the DAG structure of the underlying graph.
- It is easy to incorporate prior knowledge, for instance in the form of $x_1 \not\to x_2$. These "not-parent" constraints can be expressed as a box constraint $0 \leq B_{21} \leq 0$, which can be readily handled by numerical optimization algorithms such as L-BFGS-B Byrd et al. (1995).
- It opens new possibilities in other more complicated model classes, such as latent variable models, cyclic models, and time series.

Of course, the proposed algorithm has a number of limitations as well.

- The experiments are all using the linear non-Gaussian SEM with equal noise variance. However, as we said before, this can be a restrictive setting. The normalized likelihood $\tilde{L}(B; \mathbf{x})$ makes gradient expression complicated for non-Gaussian $\tilde{p}_j$, hence it remains a challenge to address the non-equal noise variance case.
- There are a number of ICA algorithms proposed since (Bell and Sejnowski, 1995), most noticeably the FastICA algorithm (Hyvärinen et al., 2001). In this work we only used the original MLE (Bell and Sejnowski, 1995), but is is known in the ICA literature that MLE alone does not have a good convergence property. Moreover, MLE has to choose a distributional form for the noise density $\tilde{p}_j$, which could be another disadvantage compared to e.g. kurtosis minimization algorithm, which does not require distributional assumption.
- This naturally leads to another issue of MLE, which is supergaussian vs subgaussian. In the current work we simply prescribe one of the classes. Although stability theory

suggests minor misspecification of $\tilde{p}_j$ is okay within a class (Amari et al., 1997), it leads to worse result if the class is wrong, as shown in Figure 4.6. Therefore there is a need to adaptively choose between supergaussian and subgaussian distributions.

We expand the discussion on the possible extensions in the follows.

4.5.1 Latent variable models

All of the above presentation are based on the assumption that there is no unobserved confounders. In practice, it is rarely safe to assume this is the case. Fortunately, there are methods developed to deal with latent variables, for instance the FCI (Spirtes et al., 2000) which is based on independence test as in PC but considers the possibility of unobserved confounders when orienting the edges.

Later, Hoyer et al. (2008b) considered an extension of LiNGAM to hidden confounders, and used overcomplete ICA to estimate the graph. In particular, suppose the full data $\tilde{\mathbf{x}} = (x_1, \dots, x_d)$ including latent variables follow a linear non-Gaussian SEM:

$$\tilde{\mathbf{x}} = \tilde{B}\tilde{\mathbf{x}} + \mathbf{e} \tag{4.29}$$

for a directed acyclic graph $\tilde{B}$. As usual, using the non-Gaussianity, we can cast the above equation into an ICA:

$$\tilde{\mathbf{x}} = (I - \tilde{B})^{-1}\mathbf{e} = \tilde{A}\mathbf{e} \tag{4.30}$$

where $\tilde{A}$ is the total effect matrix that contains all variables. If we only focus on a subset of $[d]$ observed variables, then the resulting system is

$$\mathbf{x} = A\mathbf{e} \tag{4.31}$$

where A only contains the rows of $\tilde{A}$ corresponding to the observed variables. To estimate the graph, Hoyer et al. (2008b) uses overcomplete ICA to estimate an initial A. To find the permutation, it simply enumerates all possible hidden and observed variable split, and for each split, try to find a permutation and scaling that satisfies the condition.

We propose to avoid this permutation step by jointly optimizing overcomplete ICA and permutation:

$$\min_{B} \quad L_{\text{OICA}}(B; X) \tag{4.32}$$

$$\text{s.t.} \quad h(\text{block}(B)) = 0 \tag{4.33}$$

where L_{OICA} is the objective for overcomplete ICA, and we force lower right block of B to be a DAG. Promising objectives for the overcomplete ICA include the Reconstruction ICA (RICA) (Le et al., 2011), and the likelihood-free overcomplete ICA (Ding et al., 2019). One downside of this approach is that it requires to specify beforehand the number of hidden variables.

4.5.2 Cyclic models

Even though this thesis is almost exclusively on how to enforce acyclicity, it is also helpful to look back and examine the necessity of such constraint. Indeed, acyclicity accompanies neat theoretical properties such as allowing for a topological order, and hence many polynomial time algorithms can be developed. On the other hand, however, there are many real world networks that exhibits feedback loops, such as brain connectivity, or even simply the supply-demand relationship in economy. It turns out one can also establish notions of d-separation and Markov equivalence on cyclic graphs Richardson (1996a,b), therefore it is not at all a weird object to study.

While Richardson (1996a) extends the PC algorithm to cyclic graphs, Lacerda et al. (2008) is the first to extend ICA-LiNGAM to cyclic graphs. In particular, the SEM is given by

$$\mathbf{x} = B\mathbf{x} + \mathbf{e}, \tag{4.34}$$

where the noise terms $\mathbf{e}$ are mutually independent and non-Gaussian, with an important distinction now that B does not necessarily represent a DAG. Instead, a weaker assumption is imposed: B does not represent self-loops, i.e. $\operatorname{diag}(B) = 0$.

Regardless of the acyclicity, thanks to the independence and non-Gaussianity, one can still write the above SEM as an ICA instance:

$$\mathbf{x} = (I - B)^{-1}\mathbf{e} = A\mathbf{e} \tag{4.35}$$

for a mixing matrix A. However, problem occurs after obtaining $\hat{A}$ (or equivalently $\hat{W} = \hat{A}^{-1}$). The permutation step in ICA-LiNGAM entirely relies on the fact that there is a unique row permutation such that $\operatorname{perm}(\hat{W})$ is a DAG. If there is no such restriction, how to find the right permutation?

It turns out in the cyclic case all permutations that lead to zeroless diagonals are considered "admissible", which comprise a distribution-entailment equivalence class (Lacerda et al., 2008). In the acyclic case, the admissible permutation is unique, however it is not unique anymore for cyclic models. Therefore Lacerda et al. (2008) finds all permutations that has zeroless diagonals, and then scale each row of $\hat{W}$ accordingly, thereby forming the equivalence class. Lacerda et al. (2008) also describes a sufficient condition to further identify a single graph from the equivalence class: if the cycles are disjoint, then there is at most one graph in the equivalence class that is stable, i.e. the maximum cycle product is bounded below 1.

Another method based on ICA is from Sanchez-Romero et al. (2019), who proposes a two-step algorithm: 1) estimate undirected graph with adaptive lasso; then 2) perform ICA only on the free parameters selected by the previous step. It relies on the shrinkage estimate of adaptive lasso to select the most stable solution.

We propose to solve a similar problem as in NOTEARS-ICA, with a change in the constraint. Notice that the stability condition

$$g(B) = \max_{j \in [d]} \max_{\text{cycle } \pi_{jj}} \prod_{(p,c) \in \pi_{jj}} |\beta_{cp}| < 1 \tag{4.36}$$

is reminiscent of the acyclicity constraint

$$h(B) = \sum_{j \in [d]} \sum_{\text{cycle } \pi_{jj}} \prod_{(p,c) \in \pi_{jj}} |\beta_{cp}| = 0 \tag{4.37}$$

with the only difference that the sum is replaced with maximum. To incorporate $g(B)$ into the continuous optimization framework, we can consider a "soft" version of the stability condition:

$$\tilde{g}(B) = \operatorname*{smax}_{j \in [d]} \operatorname*{smax}_{\text{cycle } \pi_{jj}} \prod_{(p,c) \in \pi_{jj}} |\beta_{cp}| < 1 \tag{4.38}$$

where smax is the smooth max operation such as the log-sum-exp:

$$\operatorname{smax}(\mathbf{a}) = \log \sum_i \exp(a_i) \tag{4.39}$$

Since log-sum-exp upper bounds the maximum function, it is valid to restrict $\tilde{g}(B) < 1$.

The proposed optimization algorithm is therefore

$$\min_B \quad \tilde{L}(B; X) \tag{4.40}$$

$$\text{s.t.} \quad \tilde{g}(B) < 1 \tag{4.41}$$

4.5.3 Time series

There are many time series model that relaxes the iid assumption in the general SEM. We introduce one instance here.

Structural vector autoregressive model (SVAR) (Hyvärinen et al., 2010) extends LiNGAM to time series by modeling both instantaneous and lagged effects in the vector autoregressive (VAR) framework:

$$\mathbf{x}(t) = \sum_{\tau=0}^{k} B_\tau \mathbf{x}(t - \tau) + \mathbf{e}(t) \tag{4.42}$$

where k is the time delay, $\{B_\tau\}$ are the autoregressive parameters, and $\mathbf{e}(t)$ is the innovation process. The difference to conventional VAR is that the system also contains B_0, which represents the instantaneous effect just as in an ordinary SEM.

In order to make the model identifiable, it makes a few assumptions similar to LiNGAM:

- The components of innovation process $\{e_j(t)\}$ are mutually independent of each other and over time. Furthermore, they have non-Gaussian distributions.
- The instantaneous effect matrix B_0 encodes an directed acyclic graph.

Define shorthand

$$M_\tau \coloneqq (I - B_0)^{-1} B_\tau \tag{4.43}$$

$$W \coloneqq I - B_0 \tag{4.44}$$

$$\mathbf{r}(t) \coloneqq \mathbf{x}(t) - \sum_{\tau=1}^{k} M_\tau \mathbf{x}(t-\tau) \tag{4.45}$$

Then by moving the instantaneous effect on one side, we can recover a form similar to ICA:

$$W\mathbf{r}(t) = \mathbf{e}(t) \tag{4.46}$$

Hence one can write the (normalized) likelihood as

$$\tilde{L}(\{B_\tau\}; \mathbf{x}) \coloneqq -\log p(\mathbf{x}; \{B_\tau\}) \tag{4.47}$$

$$= \sum_{t=1}^{T} \sum_{j=1}^{d} -\log \tilde{p}_j \Big(\frac{\mathbf{w}_j^T [\mathbf{x}(t) - \sum_{\tau=1}^{k} M_\tau \mathbf{x}(t-\tau)]}{\hat{\sigma}_j} \Big) + \log \hat{\sigma}_j - \log |\det(W)| \tag{4.48}$$

where $\hat{\sigma}_j^2 = \frac{1}{t} \sum_{t=1}^{T} (\mathbf{w}_j^T [\mathbf{x}(t) - \sum_{\tau=1}^{k} M_\tau \mathbf{x}(t-\tau)])^2$ is the empirical variance. The MLE is solving the following constrained problem:

$$\min_{\{B_\tau\}} \quad \tilde{L}(\{B_\tau\}; \mathbf{x}) \tag{4.49}$$

$$\text{s.t.} \quad B_0 \in \text{DAGs} \tag{4.50}$$

As always, the hardest part is the combinatorial constraint of acyclicity. In Hyvärinen et al. (2010), the authors propose a two-step method that estimates a classical VAR first and then treat the estimated residual $\hat{\mathbf{r}}(t)$ as the LiNGAM samples with B_0 as the graph. Similar to ICA-LiNGAM, this procedure is asymptotically consistent. The authors also provide another algorithm that utilizes non-Gaussianity better, based on a convolutive version of ICA called multi-channel blind deconvolution (MBD). This method can be thought of as a direct ICA-LiNGAM extension to SVAR, in a sense that it replaces ICA with MBD and follows similar permutation/scaling procedures.

We propose to solve the continuous version of the original MLE problem, similar to NOTEARS-ICA. In particular, the optimization problem now becomes

$$\min_{\{B_\tau\}} \quad \tilde{L}(\{B_\tau\}; \mathbf{x}) \tag{4.51}$$

$$\text{s.t.} \quad h(B_0) = 0 \tag{4.52}$$

This avoids invoking ICA as subroutine, and it is straightforward to incorporate structural or prior knowledge in the form of additional regularizers or constraints.

Chapter 5

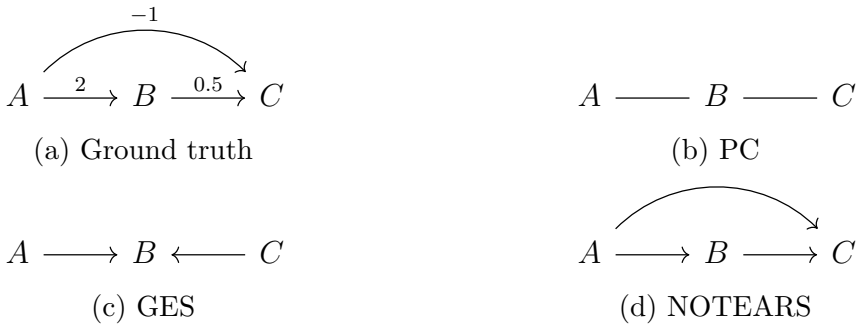
Summary and Discussion

In this thesis, we present a continuous optimization approach for the Bayesian network structure learning. We focus on three main classes of models: linear models, nonparametric additive noise models, and linear non-Gaussian models. We show a complete recipe that can convert these combinatorial problems into continuous ones. The key technical device is the smooth characterization of directed acyclic graphs (DAGs), which is not only exact but also easy to evaluate and optimize. We then generalize this to nonparametric setting, where nonparametric sparsity is extended to the nonparametric acyclicity. Lastly, we look into the special case of linear non-Gaussian models, which has nice identifiability result and serves as the basis for many other models. With minimal change, we show that our framework can be extended to non-Gaussian models, which is equivalent to a joint optimization of ICA and permutation finding.

As mentioned in the introduction, this thesis took motivation from the Markov network literature, where global search algorithms based on continuous optimization gained massive practical success in areas like bioinformatics. This thesis can hopefully serves as the first step towards achieving the same goal for the Bayesian networks. Of course, there are still many limitations and open questions left to address. We outline just a few in the follows.

5.1 Faithfulness, equal noise variance, and global optimum

One might wonder how the proposed class of methods perform for non-faithful data. We simulated an instance of such case where the ground truth has parameters that leads to cancellation in total effect, as shown in Figure 5.1(a). We use a linear Gaussian model with equal noise variance of 1, and simulated $n = 10000$ samples. Figure 5.1(b-d) shows the results on PC (Spirtes and Glymour, 1991), GES (Chickering, 2002), and the proposed NOTEARS. First notice that PC correctly identified the chain $A \to B \to C$ up to Markov equivalence, however missed the edge between $A \to C$. This is not surprising since due to the cancellation $A \perp\!\!\!\perp C$ from the data, hence the edge is removed. The remaining graph is not a v-structure, hence it returns a CPDAG $A - B - C$. On the other hand, GES not only missed the edge



(a) Ground truth (b) PC

(c) GES (d) NOTEARS

Figure 5.1: Simulation result with non-faithful data.

$A \to C$ but also incorrectly returned a v-structure. This is also expected, since this is the only DAG that satisfies the conditional independence set $A \perp\!\!\!\perp C$ shown by the data, whereas the ground truth graph has none. These are classic examples of faithfulness assumptions for PC and GES. In contrast, the proposed NOTEARS can recover the correct DAG in the absence of faithfulness.

This leads to the next question: Why does NOTEARS work? One might conjecture that NOTEARS might be implicitly exploiting additional structure in the problem, such as the equal noise variance. In fact, most of the experiments on linear SEM in this manuscript so far are based on equal noise variance, with least squares loss as the score function. Granted, equal noise variance is a favorable setting for least squares as it correctly specifies the model. However, even if we run with unequal noise variances, e.g. with $(0.5^2, 1^2, 1.5^2)$ in this example, the result stays the same as long as the ground truth parameters are identifiable (more in the next section). This is also supported by the theoretical results on the penalized least squares estimator (Aragam et al., 2019) showing high-dimensional consistency for linear Gaussian SEM without the faithfulness assumption in the case when the model is identifiable.

However it is important to note that the above result is based on the analysis on the global optimum of the penalized least squares problem. In contrast, since the smooth characterization of DAGs is exact, it also inherits the hard nonconvexity of the DAG space. By relying solely on the gradient information, our augmented Lagrangian solver is only attaining stationary points instead of global optimum. Nonetheless, the empirical results suggest that this problem might have a nicer landscape than we expected. The study on the nonconvex landscape is a popular topic in optimization and it would be exciting if we can explain the performance of NOTEARS, or even further improve the algorithm.

5.2 Scalability and real data

There two sources of computational inefficiencies in the NOTEARS framework.

First, the underlying matrix exponential operation takes $O(d^3)$ time to evaluate for a matrix of size $d \times d$. Note that evaluating least squares loss $\frac{1}{n} \|X - XW\|_F^2 = \mathrm{tr}[(I - W)^T \hat{\Sigma}(I - W)]$ for a $n \times d$ matrix X also takes $O(\min(n, d)d^2)$, if one can precompute the empirical covariance

matrix $\hat{\Sigma}$. Therefore if $O(n) \geq O(d)$, then from the complexity point of view $O(d^3)$ is not a big problem. However, often times the constant associated with matrix exponential is very large, especially when the matrix is dense. Hence there is still a need to speed up this operation. Indeed, more computationally efficient alternatives that has a better constant term than the $h(W) = \operatorname{tr} e^{W \circ W} - d$ have been proposed (Yu et al., 2019). Furthermore one can potentially take advantage of the sparsity of the matrix to greatly reduce the computational complexity.

Second source of inefficiency comes from solving a sequence of Lagrange subproblems, instead of just one. Since we are solving a constrained optimization problem with a nontrivial projection to the constraint set, we have to resort to iterative methods such as the augmented Lagrangian. However, in some special cases this might not be needed, for instance Ng et al. (2020), which solves a single (possibly just two) optimization problems by simply *regularizing* instead of constraining to the DAG. Although this work only focuses on Gaussian case, it would be interesting to see if it can be extended to other cases.

Once the scalability issue is resolved, we are ready to use the extended models to many real datasets with latent variables, cycles, and time indices such as fMRI (Sanchez-Romero et al., 2019) or gene expression data.

5.3 Identifiability

What classes of models are identifiable? Identifiability of DAG from observational data has been a long-standing problem in the literature. The conventional understanding of the identifiable model classes can be summarized as follows.

- Linear non-Gaussian models (Shimizu et al., 2006)
- Linear Gaussian models with equal noise variance (Peters and Bühlmann, 2014)
- Linear models with arbitrary noise distribution with equal noise variance (Loh and Bühlmann, 2014)
- Linear models with arbitrary noise distribution with unequal noise variance, if the noise variances are known up to a multiplicative constant or approximated close enough (Loh and Bühlmann, 2014)
- Most of nonlinear additive noise models (ANM) (Hoyer et al., 2008a; Zhang and Hyvärinen, 2009; Peters et al., 2014)

Recently, a number of interesting works try to provide milder identifiability conditions that are dependent on the parameter values. Ghoshal and Honorio (2018) provides an identifiability condition for the linear model with arbitrary noise distribution where the noise variances satisfy some minimal precision criteria. Chen et al. (2019) provides another angle to the identifiability of linear model with equal noise variance, by studying the minimal conditional variances. When identifiable, these two works are in fact dual to each other since the former identifies variables bottom-up whereas the latter is top-down. Park and Kim (2020) gives an identifiability condition for the linear Gaussian models with unequal noise variance, based on the ordering of conditional variances. This condition is strictly milder

than Loh and Bühlmann (2014) in the same setting. Park (2020) further generalizes the conditional variance condition for ANM with unequal noise variances, and show that in the linear case it includes Chen et al. (2019) and Ghoshal and Honorio (2018) as special cases.

The general trend is clear: the identifiability for linear models seem to hold in much more generality than what is traditionally known. On the other hand, however, all of the works above give sufficient conditions of identifiability. It would be interesting to find necessary conditions as well, so that we can completely characterize the identifiable region.

One way to show this is to look at the global minimum of the score. For an appetizer, consider a linear SEM with two variables $x_1 \to x_2$:

$$x_1 = z_1 \tag{5.1}$$
$$x_2 = ax_1 + z_2 \tag{5.2}$$

where z_1 and z_2 are independent noise terms with mean 0 and variance ω_1 and ω_2 respectively, i.e. the true graph and true noise covariance are given by

$$W = \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix}, \quad \Omega = \begin{bmatrix} \omega_1 & 0 \\ 0 & \omega_2 \end{bmatrix} \tag{5.3}$$

Then the true covariance of $\mathbf{x} = (x_1, x_2)$ is

$$\Sigma = \mathrm{Cov}(\mathbf{x}) = (I - W)^{-T}\Omega(I - W)^{-1} \tag{5.4}$$
$$= \begin{bmatrix} \omega_1 & a\omega_1 \\ a\omega_1 & a^2\omega_1 + \omega_2 \end{bmatrix} \tag{5.5}$$

Now let $\tilde{W}$ be any matrix with the following form:

$$\tilde{W} = \tilde{W}(b, c) = \begin{bmatrix} 0 & b \\ c & 0 \end{bmatrix} \tag{5.6}$$

Consider the global optimum of the following problem:

$$\min_{\tilde{W}} \quad R(\tilde{W}) \tag{5.7}$$
$$\text{s.t.} \quad G(\tilde{W}) \in \mathsf{DAG} \tag{5.8}$$

where $R(\tilde{W})$ is the population least squares risk

$$R(\tilde{W}) = \mathrm{tr}\left((I - \tilde{W})^T\Sigma(I - \tilde{W})\right) \tag{5.9}$$
$$= \omega_1 b^2 - 2a\omega_1 b + \omega_1 + (a^2\omega_1 + \omega_2)c^2 - 2a\omega_1 c + a^2\omega_1 + \omega_2 \tag{5.10}$$
$$= \underbrace{\omega_1(b - a)^2}_{R_b} + \underbrace{\omega_1(ac - 1)^2 + \omega_2(c^2 + 1)}_{R_c} \tag{5.11}$$

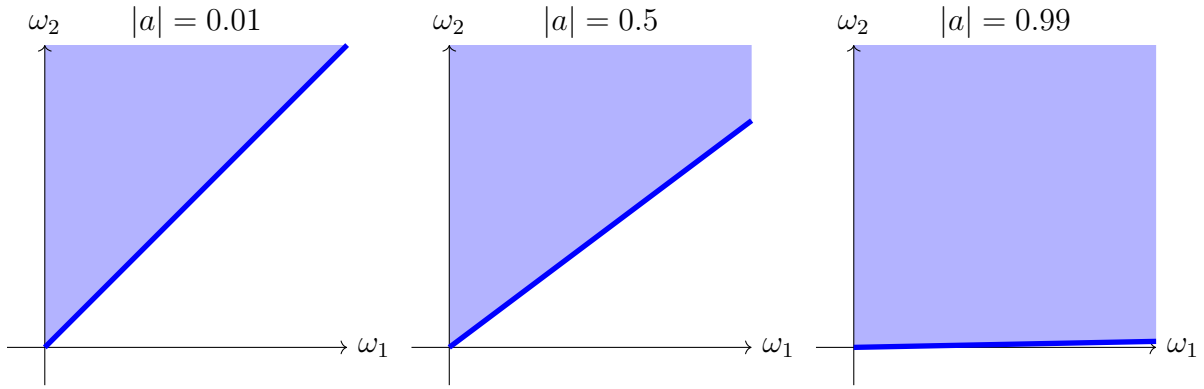


Figure 5.2: Sublevel set S (shaded) with different values of $|a|$. If the ground truth parameters (a, ω_1, ω_2) fall into the shaded region, the global minimizer of the least squares score is the ground truth. Otherwise, the global minimizer is in the wrong direction.

Notice that it decomposes into two terms, R_b and R_c, which involves only b and c respectively. On the other hand, due to the DAG constraint, one of (b, c) should be zero at the optimum. From the optimality condition, we have only one of the following holds true:

$$\partial_b R(\tilde{W}) = R_b' = 2\omega_1(b - a) = 0 \quad \Longrightarrow \quad b^* = a \tag{5.12}$$

$$\partial_c R(\tilde{W}) = R_c' = 2\omega_1 a(ac - 1) + 2\omega_2 c = 0 \quad \Longrightarrow \quad c^* = \frac{a\omega_1}{a^2\omega_1 + \omega_2} \tag{5.13}$$

Plug back in, we have the optimal population risks in each cases:

$$R_b^* = R(\tilde{W}(b^*, 0)) = \omega_1 + \omega_2 \tag{5.14}$$

$$R_c^* = R(\tilde{W}(0, c^*)) = a^2\omega_1 + \omega_2 + \frac{\omega_1\omega_2}{a^2\omega_1 + \omega_2} \tag{5.15}$$

When does the global optimum equal W? In other words, when does $R_b^* < R_c^*$? Essentially, we want the sublevel set $S = \{(a, \omega_1, \omega_2) : g(a, \omega_1, \omega_2) < 0\}$ for

$$g(a, \omega_1, \omega_2) \coloneqq R_b^* - R_c^* = \frac{-a^2\omega_1(a^2\omega_1 + \omega_2 - \omega_1)}{a^2\omega_1 + \omega_2} \tag{5.16}$$

This is a nonlinear function. Let's consider some special cases.

Example 1 (Equal variance) *If $\omega_1 = \omega_2 = \omega$, then $g(a, \omega_1, \omega_2) = -\frac{a^4\omega}{a^2+1} \leq 0$ for all $a \in \mathbb{R}$, equality holds when $a = 0$.*

Example 2 (Large a) *If $|a| \geq 1$, then $g(a, \omega_1, \omega_2) < 0$ for all $\omega_1, \omega_2 > 0$.*

Example 3 (Small a) *If $|a| < 1$, then $g(a, \omega_1, \omega_2) < 0$ for all $\omega_2 > (1 - a^2)\omega_1$. See Figure 5.2 for some examples.*

To summarize, if $\frac{\omega_2}{\omega_1} > 1 - a^2$, then $g(a, \omega_1, \omega_2) < 0$. This region is much larger than the mere equal variance case that is known to be identifiable. Therefore we can explain the nice empirical performance based on Aragam et al. (2019).

Interestingly, this condition coincides with the sufficient conditions in (Ghoshal and Honorio, 2018; Chen et al., 2019; Park and Kim, 2020; Park, 2020), hinting that these sufficient conditions have the potential to be also necessary, at least in bivariate settings, for the least squares loss. In the future, it would be interesting to generalize the above reasoning to general dimensions.

Bibliography

A. Abid, M. F. Balin, and J. Zou. Concrete Autoencoders for Differentiable Feature Selection and Reconstruction. In *International Conference on Machine Learning*, 2019.

A. H. Al-Mohy and N. J. Higham. A New Scaling and Squaring Algorithm for the Matrix Exponential. *SIAM Journal on Matrix Analysis and Applications*, 2009.

P. Alquier and G. Biau. Sparse Single-Index Model. *Journal of Machine Learning Research*, 2013.

S.-I. Amari, T.-P. Chen, and A. Cichocki. Stability analysis of learning algorithms for blind source separation. *Neural Networks*, 1997.

B. Aragam, A. A. Amini, and Q. Zhou. Learning Directed Acyclic Graphs With Penalized Neighbourhood Regression. *arXiv preprint arXiv:1511.08963*, 2015.

B. Aragam, A. A. Amini, and Q. Zhou. Globally optimal score-based learning of directed acyclic graphs in high-dimensions. In *Advances in Neural Information Processing Systems*, 2019.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *Journal of Machine Learning Research*, 2008.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 1999.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks. In *European Conference on Artificial Intelligence in Medicine*, 1989.

A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 1995.

K. Bertin and G. Lecué. Selection of variables and dimension reduction in high-dimensional non-parametric regression. *Electronic Journal of Statistics*, 2008.

P. Blöbaum, D. Janzing, T. Washio, S. Shimizu, and B. Schölkopf. Cause-Effect Inference by Comparing Regression Errors. In *International Conference on Artificial Intelligence and Statistics*, 2018.

A. Botev, H. Ritter, and D. Barber. Practical Gauss-Newton Optimisation for Deep Learning. In *International Conference on Machine Learning*, 2017.

L. Bottou and O. Bousquet. The Tradeoffs of Large Scale Learning. In *Advances in Neural*

Information Processing Systems, 2008.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *arXiv preprint arXiv:1606.04838*, 2016.

R. R. Bouckaert. Probabilistic Network Construction Using the Minimum Description Length Principle. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 1993.

P. Bühlmann, J. Peters, and J. Ernest. CAM: Causal additive models, high-dimensional order search and penalized regression. *Annals of Statistics*, 2014.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 1995.

E. Y.-J. Chen, Y. Shen, A. Choi, and A. Darwiche. Learning Bayesian networks with ancestral constraints. In *Advances in Neural Information Processing Systems*, 2016.

W. Chen, M. Drton, and Y. S. Wang. On causal discovery with an equal-variance assumption. *Biometrika*, 2019. ISSN 0006-3444.

S. Chiappa and W. S. Isaac. A causal bayesian networks viewpoint on fairness. *IFIP Advances in Information and Communication Technology*, 2019. ISSN 18684238.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*. Springer, 1996.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 2002.

D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 1997.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 2004.

J. Cussens. Bayesian network learning with cutting planes. In *Uncertainty in Artificial Intelligence*, 2011.

J. Cussens, D. Haws, and M. Studený. *Polyhedral aspects of score equivalence in Bayesian network structure learning*. 2017.

G. Darmois. Analyse générale des liaisons stochastiques. *Revue de l'Institut International de Statistique*, 1953.

A. P. Dempster. Covariance Selection. *Biometrics*, 1972.

P. Diaconis and M. Shahshahani. On Nonlinear Functions of Linear Combinations. *SIAM Journal on Scientific and Statistical Computing*, 1984.

C. Ding, M. Gong, K. Zhang, and D. Tao. Likelihood-Free Overcomplete ICA and Applications in Causal Discovery. In *Advances in Neural Information Processing Systems*, 2019.

S. Efromovich. *Nonparametric Curve Estimation*. Springer, 1999.

B. Ellis and W. H. Wong. Learning causal Bayesian network structures from experimental

data. *Journal of the American Statistical Association*, 2008.

J. Feng and N. Simon. Sparse-Input Neural Networks for High-dimensional Nonparametric Regression and Classification. *arXiv preprint arXiv:1711.07592*, 2017.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2008.

F. Fu and Q. Zhou. Learning sparse causal Gaussian networks with experimental intervention: Regularization and coordinate descent. *Journal of the American Statistical Association*, 2013.

K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel Measures of Conditional Dependence. In *Advances in Neural Information Processing Systems*, 2007.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 2011.

A. Ghoshal and J. Honorio. Learning Identifiable Gaussian Bayesian Networks in Polynomial Time and Sample Complexity. In *Advances in Neural Information Processing Systems*, 2017.

A. Ghoshal and J. Honorio. Learning linear structural equation models in polynomial time and sample complexity. In *International Conference on Artificial Intelligence and Statistics*, 2018.

O. Goudet, D. Kalainathan, P. Caillou, I. Guyon, D. Lopez-Paz, and M. Sebag. Learning Functional Causal Models with Generative Neural Networks. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer, 2018.

M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming, version 2.1, 2014. URL http://cvxr.com/cvx.

M. Gregorová, A. Kalousis, and S. Marchand-Maillet. Structured nonlinear variable selection. In *Uncertainty in Artificial Intelligence*, 2018.

A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf. Measuring Statistical Dependence with Hilbert-Schmidt Norms. In *International Conference on Algorithmic Learning Theory*, 2005.

J. Gu, F. Fu, and Q. Zhou. Penalized Estimation of Directed Acyclic Graphs From Discrete Data. *Statistics and Computing*, 2018.

P. Hall. Cross-validation and the smoothing of orthogonal series density estimators. *Journal of Multivariate Analysis*, 1987.

F. Harary and B. Manvel. On the number of cycles in a graph. *Matematický časopis*, 1971.

T. Hastie and R. Tibshirani. Generalized Additive Models: Some Applications. *Journal of the American Statistical Association*, 1987.

D. Heckerman, E. Horvitz, and B. N. Nathwani. Toward Normative Expert Systems: Part I. The Pathfinder Project. *Methods of Information in Medicine*, 1992.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 1995.

P. O. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems*, 2008a.

P. O. Hoyer, S. Shimizu, A. J. Kerminen, and M. Palviainen. Estimation of causal effects using linear non-Gaussian causal models with hidden variables. *International Journal of Approximate Reasoning*, 2008b.

C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. QUIC: Quadratic Approximation for Sparse Inverse Covariance Estimation. *Journal of Machine Learning Research*, 2014.

B. Huang, K. Zhang, Y. Lin, B. Schölkopf, and C. Glymour. Generalized Score Functions for Causal Discovery. In *International Conference on Knowledge Discovery and Data Mining*, 2018.

S. Huang, J. Li, J. Ye, A. Fleisher, K. Chen, T. Wu, and E. Reiman. A Sparse Structure Learning Algorithm for Gaussian Bayesian Network Identification from High-Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

A. Hyvärinen and S. M. Smith. Pairwise likelihood ratios for estimation of non-Gaussian structural ecuation models. *Journal of Machine Learning Research*, 2013.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc, 2001.

A. Hyvärinen, K. Zhang, S. Shimizu, and P. O. Hoyer. Estimation of a Structural Vector Autoregression Model Using Non-Gaussianity. *Journal of Machine Learning Research*, 2010.

A. M. Kagan, C. R. Rao, and Y. V. Linnik. *Characterization problems in mathematical statistics*. Wiley, 1973.

D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, and M. Sebag. SAM: Structural Agnostic Model, Causal Discovery and Penalized Adversarial Learning. *arXiv preprint arXiv:1803.04929*, 2018.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 2007.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

J. Kuipers, G. Moffa, and D. Heckerman. Addendum on the scoring of Gaussian directed acyclic graphical models. *Annals of Statistics*, 2014.

G. Lacerda, P. Spirtes, J. Ramsey, and P. O. Hoyer. Discovering Cyclic Causal Models by Independent Components Analysis. In *Uncertainty in Artificial Intelligence*, 2008.

S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien. Gradient-Based Neural DAG

Learning. *arXiv preprint arXiv:1906.02226*, 2019.

J. Lafferty and L. Wasserman. Rodeo: Sparse, greedy nonparametric regression. *Annals of Statistics*, 2008.

Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng. ICA with reconstruction cost for efficient overcomplete feature learning. In *Advances in Neural Information Processing Systems*, 2011.

A. B. Lee and R. Izbicki. A spectral series approach to high-dimensional nonparametric regression. *Electronic Journal of Statistics*, 2016.

H. Liu, J. Lafferty, and L. Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. *Journal of Machine Learning Research*, 2009.

P.-L. Loh and P. Bühlmann. High-Dimensional Learning of Linear Causal Networks via Inverse Covariance Estimation. *Journal of Machine Learning Research*, 2014.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems*, 1999.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 2006.

H. Miller and P. Hall. Local polynomial regression and variable selection. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown*. Institute of Mathematical Statistics, 2010.

R. P. Monti, K. Zhang, and A. Hyvärinen. Causal Discovery with General Non-Linear Relationships Using Non-Linear ICA. In *Uncertainty in Artificial Intelligence*, 2019.

J. M. Mooij, J. Peters, D. Janzing, J. Zscheischler, and B. Schölkopf. Distinguishing Cause from Effect Using Observational Data: Methods and Benchmarks. *Journal of Machine Learning Research*, 2016.

A. Nemirovski. Optimization II: Standard Numerical Methods for Nonlinear Continuous Optimization. 1999.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 2005.

I. Ng, A. Ghassami, and K. Zhang. On the Role of Sparsity and DAG Constraints for Learning Linear DAGs. *arXiv preprint arXiv:2006.10201*, 2020.

T. Niinimäki, P. Parviainen, and M. Koivisto. Structure Discovery in Bayesian Networks by Sampling Partial Orders. *Journal of Machine Learning Research*, 2016.

J. Nocedal and S. J. Wright. *Numerical Optimization*. 2006.

S. Ott and S. Miyano. Finding Optimal Gene Networks Using Biological Constraints. *Genome Informatics*, 2003.

G. Park. Identifiability of Additive Noise Models Using Conditional Variances. *Journal of Machine Learning Research*, 2020. ISSN 1533-7928.

G. Park and Y. Kim. Identifiability of Gaussian Structural Equation Models with Homogeneous and Heterogeneous Error Variances. *Journal of the Korean Statistical Society*, 2020.

G. Park and H. Park. Identifiability of Generalized Hypergeometric Distribution (GHD) Directed Acyclic Graphical Models. *arXiv preprint arXiv:1805.02848*, 2018.

G. Park and S. Park. High-Dimensional Poisson Structural Equation Model Learning via ℓ_1-regularized Regression. *Journal of Machine Learning Research*, 2019.

G. Park and G. Raskutti. Learning Quadratic Variance Function (QVF) DAG models via OverDispersion Scoring (ODS). *arXiv preprint arXiv:1704.08783*, 2017.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

J.-P. Pellet and A. Elisseeff. Using Markov blankets for causal structure learning. *Journal of Machine Learning Research*, 2008.

J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 2014.

J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf. Causal Discovery with Continuous Additive Noise Models. *Journal of Machine Learning Research*, 2014.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference*. The MIT Press, 2017.

S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing Features on Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.

J. Ramsey. Scaling up Greedy Causal Search for Continuous Variables. *arXiv preprint arXiv:1507.07749*, 2015.

J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour. A million variables and more: the Fast Greedy Equivalence Search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 2017.

P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society: Series B*, 2009.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using ℓ_1-regularized logistic regression. *Annals of Statistics*, 2010.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 2011.

T. S. Richardson. A discovery algorithm for directed cyclic graphs. In *Uncertainty in Artificial Intelligence*, 1996a.

T. S. Richardson. A Polynomial-Time Algorithm for Deciding Markov Equivalence of Directed Cyclic Graphical Models. In *Uncertainty in Artificial Intelligence*, 1996b.

R. W. Robinson. Counting unlabeled acyclic digraphs. *Combinatorial mathematics V*, 1977.

L. Rosasco, S. Villa, S. Mosci, M. Santoro, and A. Verri. Nonparametric sparsity and regularization. *Journal of Machine Learning Research*, 2013.

D. Rothenhäusler, J. Ernest, and P. Bühlmann. Causal inference in partially linear structural equation models. *Annals of Statistics*, 2018.

K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, 2005.

R. Sanchez-Romero, J. Ramsey, K. Zhang, M. Glymour, B. Huang, and C. Glymour. Estimating feedforward and feedback effective connections from fMRI time series: Assessments of statistical methods. *Network Neuroscience*, 2019.

A. D. Sanford and I. A. Moosa. A Bayesian network structure for operational risk modelling in structured finance operations. *Journal of the Operational Research Society*, 2012.

M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon. Learning Bayesian Networks with Thousands of Variables. In *Advances in Neural Information Processing Systems*, 2015.

M. Scanagatta, G. Corani, C. P. de Campos, and M. Zaffalon. Learning Treewidth-Bounded Bayesian Networks with Thousands of Variables. In *Advances in Neural Information Processing Systems*, 2016.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning Graphical Model Structure using L1-Regularization Paths. In *AAAI Conference on Artificial Intelligence*, 2007.

M. Schmidt, E. Berg, M. Friedlander, and K. Murphy. Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm Mark. In *International Conference on Artificial Intelligence and Statistics*, 2009.

S. C. Schwartz. Estimation of Probability Density by an Orthogonal Series. *Annals of Mathematical Statistics*, 1967.

E. Sgouritsa, D. Janzing, P. Hennig, and B. Schölkopf. Inference of Cause and Effect with Unsupervised Inverse Regression. In *International Conference on Artificial Intelligence and Statistics*, 2015.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 2006.

S. Shimizu, T. Inazumi, Y. Sogawa, A. Hyvärinen, Y. Kawahara, T. Washio, P. O. Hoyer, and K. Bollen. DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research*, 2011.

T. Silander and P. Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Uncertainty in Artificial Intelligence*, 2006.

A. P. Singh and A. W. Moore. Finding Optimal Bayesian Networks by Dynamic Programming. Technical report, 2005.

V. P. Skitovich. On a property of the normal distribution. *Doklady Akademii nauk SSSR*, 1953.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social*

Science Computer Review, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2000.

A. S. Suggala, M. Kolar, and P. Ravikumar. The Expxorcist: Nonparametric Graphical Models Via Conditional Exponential Densities. In *Advances in Neural Information Processing Systems*, 2017.

N. Tagasovska, T. Vatter, and V. Chavez-Demoulin. Nonparametric Quantile-Based Causal Discovery. *arXiv preprint arXiv:1801.10579*, 2018.

G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training Neural Networks Without Gradients: A Scalable ADMM Approach. In *International Conference on Machine Learning*, 2016.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Uncertainty in Artificial Intelligence*, 2005.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 2006.

A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2003.

P. van Beek and H.-F. Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Principles and Practice of Constraint Programming*, 2015.

S. van de Geer and P. Bühlmann. ℓ_0-Penalized maximum likelihood for sparse directed acyclic graphs. *Annals of Statistics*, 2013.

A. Voorman, A. Shojaie, and D. Witten. Graph estimation with joint additive models. *Biometrika*, 2014.

G. Wahba. Data-based Optimal Smoothing of Orthogonal Series Density Estimates. *Annals of Statistics*, 1981.

X. Wang, D. Dunson, and C. Leng. No penalty no tears: Least squares in high-dimensional linear models. In *International Conference on Machine Learning*, 2016.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.

J. Xiang and S. Kim. A* Lasso for Learning a Sparse Bayesian Network Structure for Continuous Variables. In *Advances in Neural Information Processing Systems*, 2013.

E. Yang, P. Ravikumar, G. I. Allen, and Z. Liu. Graphical Models via Univariate Exponential Family Distributions. *Journal of Machine Learning Research*, 2015.

M. Ye and Y. Sun. Variable Selection via Penalized Neural Network: a Drop-Out-One Loss Approach. In *International Conference on Machine Learning*, 2018.

Y. Yu, J. Chen, T. Gao, and M. Yu. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *International Conference on Machine Learning*, 2019.

M. Yuan. On the identifiability of additive index models. *Statistica Sinica*, 2011.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model.

Biometrika, 2007.

B. Zhang, C. Gaiteri, L. G. Bodea, Z. Wang, J. McElwee, A. A. Podtelezhnikov, C. Zhang, T. Xie, L. Tran, R. Dobrin, E. Fluder, B. Clurman, S. Melquist, M. Narayanan, C. Suver, H. Shah, M. Mahajan, T. Gillis, J. Mysore, M. E. MacDonald, J. R. Lamb, D. A. Bennett, C. Molony, D. J. Stone, V. Gudnason, A. J. Myers, E. E. Schadt, H. Neumann, J. Zhu, and V. Emilsson. Integrated systems approach identifies genetic nodes and networks in late-onset Alzheimer's disease. *Cell*, 2013.

K. Zhang and A. Hyvärinen. On the Identifiability of the Post-Nonlinear Causal Model. In *Uncertainty in Artificial Intelligence*, 2009.

K. Zhang, J. Peters, and B. Schölkopf. Kernel-based Conditional Independence Test and Application in Causal Discovery. In *Uncertainty in Artificial Intelligence*, 2011.

K. Zhang, Z. Wang, J. Zhang, and B. Schölkopf. On Estimation of Functional Causal Models: General Results and Application to Post-Nonlinear Causal Model. *ACM Transactions on Intelligent Systems and Technology*, 2016.

X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*, 2018.

X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. P. Xing. Learning Sparse Nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, 2020.

K. Zhong, I. E. H. Yen, I. S. Dhillon, and P. Ravikumar. Proximal Quasi-Newton for Computationally Intensive ℓ_1-regularized M-estimators. In *Advances in Neural Information Processing Systems*, 2014.

Q. Zhou. Multi-domain sampling with applications to structural inference of Bayesian networks. *Journal of the American Statistical Association*, 2011.

S. Zhou. Thresholding Procedures for High Dimensional Variable Selection and Statistical Estimation. In *Advances in Neural Information Processing Systems*, 2009.