Learning Sparse Nonparametric DAGs

Abstract

We develop a framework for learning sparse nonparametric directed acyclic graphs (DAGs) from data. Our approach is based on a recent algebraic characterization of DAGs that led to a fully continuous program for score-based learning of DAG models parametrized by a linear structural equation model (SEM). We extend this algebraic characterization to nonparametric 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. Unlike existing approaches that require specific modeling choices, loss functions, or algorithms, we present a completely general framework that can be applied to general nonlinear models (e.g. without additive noise), general differentiable loss functions, and generic black-box optimization routines. The code is available at https://github.com/xunzheng/notears.

1 Introduction

Learning DAGs from data is an important and classical problem in machine learning, with a diverse array of applications in causal inference (Spirtes et al., 2000), fairness and accountability (Kusner et al., 2017), medicine (Heckerman et al., 1992), and finance (Sanford and Moosa, 2012). In addition to their undirected counterparts, DAG models offer a parsimonious, interpretable representation of a joint distribution that is useful in practice. Unfortunately, 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). 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., 2013; Banerjee et al., 2008).

In this paper, 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 is based on a recent algebraic characterization of acyclicity due to Zheng et al. (2018) that recasts 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. This work relies heavily on the linear parametrization in terms of a weighted adjacency matrix \( W \in \mathbb{R}^{d \times d} \), which is a stringent restriction on the class of models. One of the key technical contributions of the current work is extending this to general nonparametric problems, where no such parametrization in terms of a weighted adjacency matrix exists.

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 from Zheng et al. (2018) to general nonparametric models, show that the linear parametrization is a special case of the general framework, and apply this to several popular examples for modeling nonlinear dependencies (Section 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 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 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.

Related work The problem of learning nonlinear and nonparametric DAGs from data has gained significant interest in recent years, including additive models (Bühlmann et al., 2014; Voorman et al., 2014; Ernest et al., 2016), generalized linear models (Park, 2018; Park and Raskutti, 2017; Park and Park, 2019; Gu et al., 2018), additive noise models (Hoyer et al., 2009; 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., 2019a; 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., 2008; Zhang et al., 2012). Another line of work uses quantile scoring (Tagasovska et al., 2018). Also of relevance is the literature on nonparametric variable selection (Bertin et al., 2008; Lafferty et al., 2008; Miller et al., 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 (2018)—as our focus is on a generic formulation of an optimization problem that can be solved with generic solvers (see Section 2 for a more detailed comparison). This also distinguishes this paper 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 (Section 5.1; Appendix D). As such, we hope that this work is able to spur future work using more sophisticated nonparametric estimators and optimization schemes.

Notation Norms will always be explicitly subscripted to avoid confusion: $||\cdot||_p$ is the $\ell_p$-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 $f : \mathbb{R} \rightarrow \mathbb{R}$ 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 $i$th element is $f(a^i)$, where $a^i$ is the $i$th row of $A$.

2 Background

Our approach is based on (acyclic) structural equation models as follows. Let $X = (X_1, \ldots, X_d)$ be a random vector and $G = (V, E)$ a DAG with $V = X$. We assume that there exist functions $f_j : \mathbb{R}^d \rightarrow \mathbb{R}$ and $g_j : \mathbb{R} \rightarrow \mathbb{R}$ such that

$$E[X_j \mid X_{pa(j)}] = g_j(f_j(X)), \quad E f_j(X) = 0 \quad (1)$$

and $f_j(u_1, \ldots, u_d)$ does not depend on $u_k$ if $X_k \notin pa(j)$, where $pa(j)$ denotes the parents of $X_j$ in $G$. Formally, the independence statement means that for any $X_k \notin pa(j)$, the function $\phi(u) := f_j(X_1, \ldots, X_{k-1}, u, X_{k+1}, X_d)$ is constant for all $u \in \mathbb{R}$. Thus, $G$ encodes the conditional independence structure of $X$. The functions $g_j$, which are typically known, allow for possible non-additive errors such as in generalized linear models (GLMs). The model (1) is quite general and includes additive noise models, linear and generalized linear models, and additive models as special cases (Section 3.3).

In this setting, the DAG learning problem can be stated as follows: Given a data matrix $X = [x_1 \mid \cdots \mid x_d] \in \mathbb{R}^{n \times d}$ consisting of $n$ i.i.d. observations of the model (1), we seek to learn the DAG $G(X)$ that encodes the dependency between the variables in $X$. Our approach is to learn $f = (f_1, \ldots, f_d)$ such that $G(f) = G(X)$ using

\[\text{The reason for writing } f_j(X) \text{ instead of } f_j(X_{pa(j)}) \text{ is to simplify notation by ensuring each } f_j \text{ is defined on the same space.}\]
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:

$$\min_f L(f) \text{ subject to } G(f) \in \text{DAG},$$

where

$$L(f) = \frac{1}{n} \sum_{j=1}^{d} \ell(x_j, f_j(X)).$$

There are two challenges in this formulation: 1) How to enforce the acyclicity constraint that $G(f) \in \text{DAG}$, 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 (1). To address this, we develop a suitable surrogate of $W$ defined for general nonparametric models, to which we can apply the trace exponential regularizer from Zheng et al. (2018).

### 2.1 Identifiability

Existing papers approach this problem as follows: 1) Assume a specific model for (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-catalogued list of identifiability results for various linear, parametric, and nonlinear models, which we review briefly below (see also Section 3.3).

When the model (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., 2009; 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, 2018). In the sequel, we assume that the model is chosen such that the graph $G$ is uniquely defined from (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 for specific examples with discussion on identifiability.

### 2.2 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 and Zhou, 2015; Gu et al., 2018; Park and Raskutti, 2017; Park, 2018; Chen et al., 2018; 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; Vornman 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 $X$, 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 dependence (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 (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 multi-layer 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 Characterizing acyclicity in nonparametric SEM

In this section, we discuss how to extend the trace exponential regularizer from Zheng et al. (2018) beyond the linear setting, and then discuss several special cases.

#### 3.1 Linear SEM and the trace exponential regularizer

We begin by briefly reviewing Zheng et al. (2018) in the linear case, i.e. $g_j(s) = s$ and $f_j(X) = w_j^T X$ for some $w_j \in \mathbb{R}^d$. This defines a matrix $W = [w_1 | \cdots | w_d] \in \mathbb{R}^{d \times d}$ that precisely encodes the graph $G(f)$, i.e. there is an edge $X_k \rightarrow X_j$ in $G(f)$ if and only if $w_{kj} \neq 0$. In this case, 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 \( h(W) = \text{tr} e^{W^T W - d} \), where \([W \circ W]_{kj} = w_{kj}^2 \). Then Zheng et al. (2018) show that (2) is equivalent to

\[
\min_{W \in \mathbb{R}^{d \times d}} L(W) \text{ subject to } h(W) = 0, 
\]

(3)

The key insight from Zheng et al. (2018) is replacing the combinatorial constraint \( G(W) \in \text{DAG} \) with the continuous constraint \( h(W) = 0 \). 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 (2).

### 3.2 A notion of nonparametric acyclicity

Unfortunately, for general models of the form (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 \( k \)th 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 (2009)). 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 matrix \( W(f) = W(f_1, \ldots, f_d) \in \mathbb{R}^{d \times d} \) with entries

\[
[W(f)]_{kj} := \| \partial_k f_j \|_{L^2}. 
\]

(4)

precisely encodes the dependency structure amongst the \( X_j \). Thus the program (2) is equivalent to

\[
\min_{f_j \in H^1(\mathbb{R}^d), X_j \in [d]} L(f) \text{ subject to } h(W(f)) = 0. 
\]

(5)

This implies an equivalent continuous formulation of the program (2). Moreover, when the functions \( f_j \) are all linear, \( W(f) \) is the same as the weighted adjacency matrix \( W \) defined in Section 3.1. Thus, (5) is a genuine generalization of the linear case (3).

### 3.3 Special cases

In addition to applying to general nonparametric models of the form (2) and linear models, the program (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., 2009; Peters et al., 2014) assumes that

\[
X_j = f_j(X) + z_j, \quad E[f_j(X)] = 0, \quad z_j \perp f_j(X). 
\]

(6)

and \( z_j \sim \mathbb{P}_j \) is the random noise. Clearly this is a special case of (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 (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 \( E[X_j | X_{pa(j)}] = g_j(w_j^T X) \) for some known link functions \( g_j : \mathbb{R} \rightarrow \mathbb{R} \) and \( w_j \in \mathbb{R}^d \). 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

\[
E[X_j | X_{pa(j)}] = g_j(f_j(X)). 
\]

(7)

Clearly, (6) is a special case of (7). Furthermore, for linear mean functions, \( [W(f)]_{kj} = 0 \) if and only if \( w_{jk} = 0 \), recovering the parametric approach in Zheng et al. (2018). Several special cases of GLMs are known to be identifiable: Linear Gaussian with equal variances (Peters and Bühlmann, 2013), 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 \( f_j(X) \) is a polynomial in \( X_1, \ldots, X_d \). More generally, given a known dictionary of functions \( \eta_1(u_1, \ldots, u_d), \ldots, \eta_n(u_1, \ldots, u_d) \), we require that \( f_j(X) = \sum_\beta \beta_\beta \eta_\beta(X) \). Then it is easy to check that \( [W(f)]_{kj} = 0 \) if and only if \( \beta_{jk} = 0 \) whenever \( \eta_\beta \) depends on \( u_k \). For each \( k \), define \( a_{jk}(u) := f_j(X_1, \ldots, X_{k-1}, u, X_{k+1}, X_d) \). 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 \( f_j(X) = \sum_{k \neq j} f_{jk}(X_k) \) for some \( f_{jk} \in H^1(\mathbb{R}) \). Then it is straightforward to show that \( \| \partial_k f_{jk} \|_{L^2} = 0 \) 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 $f_j(X) = \sum_{m=1}^{M} h_{jm}(\beta_{jm}^T X)$ for some $h_{jm} \in H^1(\mathbb{R})$ and $\beta_{jm} \in \mathbb{R}^d$. 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 $\beta_{jm} = 0$ for each $m$. In other words, $[W(f)]_{kj} = 0$ if and only if $\sum_{m=1}^{M} \beta_{jm}^2 = 0$. 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 (5) is straightforward to solve (see Section 4.3).

4 Optimization

In general, the program (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 (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 $\mathbb{E}[X_j | X_{pa(j)}]$ (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 (5) into a tractable optimization problem. By approximating the $f_j$ with a flexible family of functions parameterized by $\theta$, 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 $\theta$ is allowed to depend on $n$, although this dependence will be suppressed.

4.1 Multilayer perceptrons

We first consider the use of neural networks to approximate the $f_j$, as in an ANM (6) or GLM (7). Consider a multilayer perceptron (MLP) with $h$ hidden layers and a single activation $\sigma: \mathbb{R} \rightarrow \mathbb{R}$, given by

$$\text{MLP}(w; A^{(1)}, \ldots, A^{(h)}) = \sigma(A^{(h)} \sigma(\cdots \sigma(A^{(1)} u)))),$$

$$A^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}}, \quad m_0 = d.$$

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 $\text{MLP}(w; A^{(1)}, \ldots, A^{(h)})$ is independent of $u_k$—this is important both for enforcing acyclicity and sparsity. It is not hard to see that if the $k$th column of $A^{(1)}$ is zero (i.e. $A^{(1)}_{k:b} = 0$ for all $b = 1, \ldots, m_1$), then $\text{MLP}(w; A^{(1)}, \ldots, A^{(h)})$ 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 1.** 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 $k$th column of $A^{(1)}$ consists of all zeros. Then $\mathcal{F} = \mathcal{F}_0$.

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 $A^{(1)}_{k:b} = 0$ 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 $\|k\text{-th column}(A^{(1)}_{j:})\|_2 = 0$. 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 $j$th MLP and $\theta = (\theta_1, \ldots, \theta_d)$. Define $[W(\theta)]_{kj} = \|k\text{-th column}(A^{(1)}_{j:})\|_2$. The problem (2) thus reduces to

$$\min_{\theta} \frac{1}{n} \sum_{j=1}^{d} \ell(x_j, \text{MLP}(X; \theta_j)) + \lambda \|A^{(1)}_{j:}\|_{1,1}$$

subject to $h(W(\theta)) = 0$. (8)
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, 2008). While many techniques are valid, we adopt an approach based on Ravikumar et al. (2009). Let \( \{ \varphi_r \}_{r=1}^{\infty} \) be an orthonormal basis of \( H^1(\mathbb{R}^d) \) such that \( \mathbb{E}\varphi_r(X) = 0 \) for each \( r \). Then any \( f \in H^1(\mathbb{R}^d) \) can be written uniquely

\[
f(u) = \sum_{r=1}^{\infty} \alpha_r \varphi_r(u), \quad \alpha_r = \int_{\mathbb{R}^d} \varphi_r(u) f(u) \, du. \tag{9}
\]

As long as the coefficients \( \alpha_r \) decay sufficiently fast, \( f \) can be well-approximated by the finite series \( \sum_{r=1}^{R} \alpha_r \varphi_r \). 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 et al. (2016). Thus,

\[
f_j(u_1, \ldots, u_d) = \sum_{k \neq j} f_{jk}(u_k) = \sum_{k=1}^{R_k} \sum_{r=1}^{\infty} \alpha_{jkr} \varphi_r(u_k).
\]

Given integers \( R_k \) and assuming \( f_{jk} \) is sufficiently smooth, we have \( \| f_{jk} - f_{jk}\|_2 = O(1/R_k) \) (Efromovich, 2008), so that the overall approximation error is on the order \( O(d/\min_k R_k) \). Furthermore, \( \|W(f)\|_{kj} = 0 \iff \| f_{jk}\|_2 = 0 \iff \alpha_{jkr} = 0 \) 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 \( \theta \) denote the parameters \( \alpha_{jkr} \) 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 \( \Phi_k \) be the matrix \( [\Phi_k]_{ir} = \varphi_r(X_k^{(i)}) \). To estimate the coefficients \( \alpha_{jkr} \), we solve

\[
\min_{\theta} \frac{1}{n} \sum_{j=1}^{d} \ell(X_j, \sum_{k \neq j} \Phi_k a_{jk}) + \lambda_1 \sum_{k \neq j} \| a_{jk} \|_1 + \lambda_2 \| \Phi_k a_{jk} \|
\]

subject to \( h(W(\theta)) = 0 \). \tag{11}

This is similar to Ravikumar et al. (2009) with the addition of an explicit \( \ell_1 \) constraint.

4.3 Solving the continuous program

Having converted \( L(f) \) and \( W(f) \) to their finite-dimensional counterparts, we are now ready to solve (5) by applying standard optimization techniques. We emphasize that the hard work went into formulating the programs (8) and (11) as generic problems for which off-the-shelf solvers can be used. Importantly, since in both (8) and (11) the term \( W(\theta) \) is differentiable w.r.t. \( \theta \), the optimization program is an \( \ell_1 \)-penalized smooth minimization under a differentiable equality constraint. As in Zheng et al. (2018), the standard machinery of augmented Lagrangian can be applied, resulting in a series of unconstrained problems:

\[
\min_{\theta} F(\theta) + \lambda \| \theta \|_1.
\]

\[
F(\theta) = L(\theta) + \frac{\rho}{2} \| h(W(\theta)) \|^2 + \alpha h(W(\theta)).
\]

where \( \rho \) is a penalty parameter and \( \alpha \) is a dual variable.

A number of optimization algorithms can be applied to the above unconstrained \( \ell_1 \)-penalized smooth minimization problem. A natural choice is the L-BFGS-B algorithm (Byrd et al., 1995), which can be directly applied by casting (12) into a box-constrained form:

\[
\min_{\theta} F(\theta) + \lambda \| \theta \|_1
\]

\[
\iff \min_{\theta^+, \theta^-} F(\theta^+ - \theta^-) + \lambda 1^T(\theta^+ + \theta^-) \tag{13}
\]

where \( 1 \) is a vector of all ones. We note that as in Zheng et al. (2018), (12) 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.

5 Experiments

We study the empirical performance of two instances of the general framework: MLP (4.1) and Sobolev expansions (4.2), denoted by NOTEARS-MLP and NOTEARS-Sob. For NOTEARS-MLP we use an MLP with one hidden layer with 10 hidden units and sigmoid activation function. For NOTEARS-Sob we use Sobolev basis \( \varphi_r(u) = s_r \sin(u/s_r), \) \( s_r = 2/((2r - 1)\pi) \) \( (r = 1, \ldots, 10) \). Complete details on all baselines and simulations, including a discussion of computational complexity and runtimes, can be found in the appendix.

Baselines For comparison, the following methods are chosen as baselines: fast greedy equivalence search (FGS) (Ramsey et al., 2017), greedy equivalence search with generalized scores (GGES) (Huang et al., 2018), DAG-GNN (GNN) (Yu et al., 2019), NOTEARS (Linear) (Zheng et al., 2018) for linear SEM, and causal additive models (CAM) (Bühlmann et al., 2014).
summarize, FGS and Linear are specialized at linear models, whereas GSGES, GNN, and CAM targets general nonlinear dependencies. Comparisons with other score-based methods (KGV score (Bach and Jordan, 2003), Spearman correlation (Sokolova et al., 2014)) and constraint-based methods (PC (Spirtes et al., 2000), MM-MB (Aliferis et al., 2010)) can be found in previous work (Huang et al., 2018), hence are omitted.

Simulation The ground truth DAG is generated from two random graph models: Erdos-Renyi (ER) and scale-free (SF). We use ER2 to denote an ER graph with $s_0 = 2d$ edges, likewise for SF. Given the ground truth DAG, we simulate the SEM $X_j = f_j(X_{pa(j)}) + z_j$ for all $j \in [d]$ in topological order, and each $z_j \sim N(0, 1)$. To evaluate the performance under different data generation mechanisms, we consider four models for the $f_j$: 1) Additive models with Gaussian processes (GPs) for each $f_{jk}$, 2) Index models ($M = 3$), 3) ANM with MLPs, and 4) ANM with GPs.

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, similar to (Zheng et al., 2018).

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 NOTEARS-MLP and NOTEARS-Sob, $\lambda =$
\{0.01, 0.03\} are used for \(n = \{1000, 200\}\) respectively. Figure 1 shows the SHD in various settings; the complete set of results for the remaining metrics are deferred to the supplement. Overall, the proposed NOTEARS-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 NOTEARS-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, NOTEARS-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 NOTEARS-Sob degrades as expected. Finally, we observe that GSGES outperforms NOTEARS-MLP and NOTEARS-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 NOTEARS-MLP and NOTEARS-Sob. Also, with by properly tuning the regularisation parameter, the performance of NOTEARS-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 paper, we fix a reasonable \(\lambda\) for all settings (see Appendix D for more discussion).

With respect to runtime and scalability, we note that the computational complexity of our approach depends on the choice of nonparametric estimator. For example, NOTEARS-MLP requires \(O(nd^2m + d^2m + d^3)\) flops per iteration of L-BFGS-B. In terms of runtime, the average runtime of GSGES on ER2 with \(d = 40, n = 1000\) is over 90 minutes, whereas NOTEARS-MLP takes less than five minutes on average (see Appendix D for more discussion).

Figure 2 shows the SHD compared with CAM. We first observe that NOTEARS-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 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 NOTEARS-MLP and NOTEARS-Sob on additive GP and GP settings, denoted as NOTEARS-MLP++ and NOTEARS-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 (13), demonstrating the flexibility of the proposed approach. The performance improves in both cases, matching or improving vs. CAM.

5.2 Real data

Finally, we evaluated NOTEARS-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 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 NOTEARS-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 NOTEARS-MLP, 7 edges agree with the consensus network: raf \(\rightarrow\) mek, mek \(\rightarrow\) erk, PLCg \(\rightarrow\) PIP2, PIP3 \(\rightarrow\) PLCg, PIP3 \(\rightarrow\) PIP2, PKC \(\rightarrow\) mek, PKC \(\rightarrow\) 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 \(\rightarrow\) erk, PIP3 \(\rightarrow\) PLCg, PKC \(\rightarrow\) mek.

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.

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

We acknowledge the support of NSF via IIS-1909816, OAC-1934584, ONR via N000141812861, NSF IIS1563887 and DARPA/AFRL FA87501720152. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Defense Advanced Research Projects Agency, or Air Force Research Laboratory.

References


