Contextual Directed Acyclic Graphs

Ryan Thompson
University of New South Wales
CSIRO's Data61

Edwin V. Bonilla
CSIRO's Data61

Robert Kohn
University of New South Wales

Abstract

Estimating the structure of directed acyclic graphs (DAGs) from observational data remains a significant challenge in machine learning. Most research in this area concentrates on learning a single DAG for the entire population. This paper considers an alternative setting where the graph structure varies across individuals based on available "contextual" features. We tackle this contextual DAG problem via a neural network that maps the contextual features to a DAG, represented as a weighted adjacency matrix. The neural network is equipped with a novel projection layer that ensures the output matrices are sparse and satisfy a recently developed characterization of acyclicity. We devise a scalable computational framework for learning contextual DAGs and provide a convergence guarantee and an analytical gradient for backpropagating through the projection layer. Our experiments suggest that the new approach can recover the true context-specific graph where existing approaches fail.

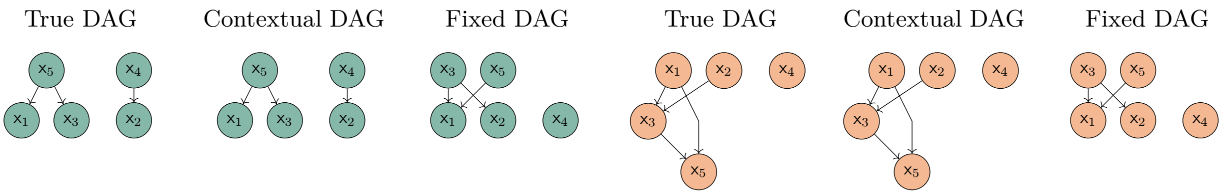
1 INTRODUCTION

Directed acyclic graphs (DAGs)—graphs with directed edges and no cycles—are a core tool for probabilistic graphical modeling (see, e.g., Murphy 2023). Their ability to represent complex multivariate relationships makes them valuable across a broad range of domains, including psychology (Foster 2010), economics (Imbens 2020), and epidemiology (Tennant et al. 2021). Though DAGs have a long and rich history in machine learning and statistics (Lauritzen and Spiegelhalter 1988; Pearl 1988), they have recently attracted significant attention due to a computational breakthrough by Zheng et al. (2018) that reframed the combinatorial DAG structure learning problem as a continuous optimization problem. This and subsequent continuous reformulations of the DAG problem have enabled scalable structure learning for applications that were considered intractable. See Vowels, Camgoz, and Bowden (2022) for a recent review of this work.

A major focus of the structure learning literature, both recent and past, is on estimating a single DAG for the entire population. Although these "fixed DAGs" are suitable in standard settings, they can fail in heterogeneous or non-stationary environments (Huang et al. 2020; Zhou, He, and Ni 2022) and in the presence of external modifying effects (Ni, Stingo, and Baladandayuthapani 2019). An important application where these types of problems arise is that of understanding recreational drug consumption patterns. The consumption patterns can be represented as a DAG whose structure is dictated according to individual personality traits. As traits differ across individuals, so does the graph structure—an edge between a pair of drugs present for one individual might be absent for another. Furthermore, an edge may point in opposite directions for different individuals. The personality traits in this example represent information that encodes context into the graph. DAGs that change with context, which we refer to as "contextual DAGs," lie beyond the capability of existing structure learning methods.

Our paper demonstrates that it is possible to learn contextual DAGs using available contextual information. Let $x = (\mathsf{x}_1, \ldots, \mathsf{x}_p)^\top$ be a vector of continuous variables (nodes) and $z = (\mathsf{z}_1, \ldots, \mathsf{z}_m)^\top$ be a vector of contextual features. We study linear DAGs on x, which can be represented via a $p \times p$ weighted adjacency matrix $W = (\mathsf{w}_{jk})$ such that w_{jk} is non-zero if and only if a directed edge exists from node j to node k. We define a fixed DAG as a graph such that W is independent of z and a contextual DAG as a graph where W is a function of z. Via the weighted adjacency representation, a contextual DAG can be expressed in

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).



(a) Graph conditional on $z = z_1$.

(b) Graph conditional on $z = z_2$.

Figure 1: Illustration of the contextual DAG. The true graph is a function of the features z. The left three graphs correspond to one realization of z, and the right three correspond to a second independent realization.

terms of a (linear) structural equation model:

$$\mathsf{x}_k = \sum_{j=1}^{p} \mathsf{x}_j \mathsf{w}_{jk}(z) + \varepsilon_k, \quad k = 1, \dots, p, \tag{1}$$

where $\varepsilon = (\varepsilon_1, \dots, \varepsilon_p)^\top$ is a zero mean stochastic noise vector. Given the contextual features z, the weighted adjacency function $W(z) := [\mathsf{w}_{jk}(z)]$ encodes the parents of x_k (the js such that $\mathsf{w}_{jk}(z)$ is non-zero) and the effects of those parents. A key property of a DAG is that its nodes can be sorted such that parents come before children, known as a topological ordering (see, e.g., Murphy 2023, §4.2). Thus, a contextual DAG allows both the topological ordering and weights to be functions of the contextual features. A fixed DAG is a special case where these functions are constant.

Estimating the structural equation model (1) is challenging, even in the fixed case, because of the combinatorial nature of the search space. Leveraging recent breakthroughs in continuous optimization routines for fixed DAGs (Zheng et al. 2018; Bello, Aragam, and Ravikumar 2022), we devise a neural network architecture that learns to map from the contextual feature space to the space of DAGs. It achieves this task using a novel layer that projects a simple directed graph[1] from a feedforward neural network onto the space of acyclic graphs. This "projection layer" simultaneously induces sparsity over the graph by constraining it to an ℓ_1 ball of appropriate size, leading to a parsimonious representation of the data with as few edges as feasible. Although the projection layer constitutes an iterative optimization algorithm, we establish its convergence properties and show that it can be solved in parallel on a GPU, allowing for efficient forward passes. We also derive an analytical form for the layer's gradients to facilitate efficient backpropagation during training.

To briefly illustrate our proposal, we consider a small-scale Gaussian model with five variables generated by a graph whose node ordering depends on a contextual feature vector $z \in \mathbb{R}^2$. The number of edges also varies with the contextual features, so graphs for different realizations of z need not have the same sparsity. Figure 1 compares the true DAG under this model with the contextual and fixed DAG estimates for two independent draws of z. The topological ordering of the true graph under z_1 is distinct from that under z_2. For instance, the direction of the edge between x_1 and x_5 changes from z_1 to z_2. The number of edges also varies—three edges under z_1 and four under z_2. By learning to predict a graph from the contextual features, the contextual DAG captures these evolving dependencies and recovers the true graph in both cases. Conversely, the fixed DAG, which assumes the structure is unchanging, fails to recover either graph.

This paper makes four contributions:

1. We introduce a new neural network architecture that learns to predict context-specific DAGs.

2. We establish a theoretical convergence guarantee for the network's projection layer and derive its analytical gradient, avoiding the expense of automatically differentiating through the optimizer.

3. We analyze our approach in numerical experiments that show it is the best available tool for structure learning in the contextual setting.

4. We provide the structure learning community with `ContextualDAG`, an open-source Julia implementation that scales well on GPUs.

Before proceeding, we remark that we do not seek to causally interpret contextual DAGs in this paper, and hence make no causal assumptions (e.g., causal sufficiency). We instead focus on the core learning problem of finding a contextual DAG that fits the data well.

[1] A simple directed graph is a directed graph with no self-loops (an edge directed from a node back to itself) and at most one edge in each direction between any pair of nodes.

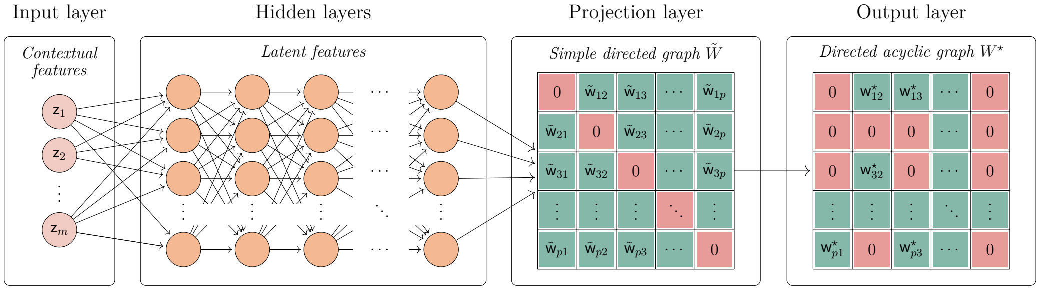

Input layer
Hidden layers
Projection layer
Output layer
Contextual features
z_1
z_2
z_m
Latent features
Simple directed graph $\tilde{W}$
0 $\tilde{w}_{12}$ $\tilde{w}_{13}$ $\cdots$ $\tilde{w}_{1p}$
$\tilde{w}_{21}$ 0 $\tilde{w}_{23}$ $\cdots$ $\tilde{w}_{2p}$
$\tilde{w}_{31}$ $\tilde{w}_{32}$ 0 $\cdots$ $\tilde{w}_{3p}$
$\tilde{w}_{p1}$ $\tilde{w}_{p2}$ $\tilde{w}_{p3}$ $\cdots$ 0
Directed acyclic graph $W^\star$
0 $w^\star_{12}$ $w^\star_{13}$ $\cdots$ 0
0 0 0 $\cdots$ 0
0 $w^\star_{32}$ 0 $\cdots$ 0
$w^\star_{p1}$ 0 $w^\star_{p3}$ $\cdots$ 0

Figure 2: Neural network architecture of the contextual DAG. The features z pass through hidden layers to produce a simple directed graph $\tilde{W}$. The projection layer makes $\tilde{W}$ acyclic and sparse, resulting in a DAG $W^\star$.

2 CONTEXTUAL DAGS

To facilitate exposition, it is helpful to rewrite the structural equation model (1) more compactly as

$$x = W(z)^\top x + \varepsilon, \tag{2}$$

where the function $W(z) : \mathbb{R}^m \to \text{DAG}$ maps the contextual features z onto the space of acyclic weighted adjacency matrices. More precisely, the sparsity pattern of the matrix produced by $W(z)$, corresponding to the set of edges, changes with z under the constraint that the pattern never contains a directed cycle.

2.1 Fixed DAG Problem

Before considering the estimation problem corresponding to the model (2), it is useful to review the learning problem for a fixed DAG where z is absent:

$$\begin{aligned} \min_{W \in \mathbb{R}^{p\times p}} \quad & \mathrm{E}_x\left[\|x - W^\top x\|_2^2\right] \\ \text{s.t.} \quad & W \in \text{DAG} \\ & \|W\|_{\ell_1} \leq \lambda. \end{aligned} \tag{3}$$

Here, $\|W\|_{\ell_1} := \|\operatorname{vec}(W)\|_1$ is the sum of absolute values of the elements of W. The first constraint in (3) states that W cannot contain cycles (i.e., it must be acyclic), while the second constraint is an ℓ_1 regularizer that promotes sparsity in W. A sparse W implies a parsimonious graph with few edges.

2.2 Contextual DAG Problem

In the contextual setting, W is no longer a fixed matrix but instead a function of the random variable z:

$$\begin{aligned} \min_{W \in \mathcal{F}} \quad & \mathrm{E}_{x,z}\left[\|x - W(z)^\top x\|_2^2\right] \\ \text{s.t.} \quad & W(z) \in \text{DAG for all } z \in \mathbb{R}^m \\ & \mathrm{E}_z\left[\|W(z)\|_{\ell_1}\right] \leq \lambda, \end{aligned} \tag{4}$$

where the set $\mathcal{F}$ is some class of functions that define feasible solutions (e.g., neural networks). The DAG constraint is now a statement that the (random) matrix $W(z)$ must correspond to a DAG over the entire support of z. Said differently, for all realizations of z, the weighted adjacency matrix must be acyclic. The ℓ_1 regularizer continues to impose sparsity, but now on the expectation of $W(z)$, meaning that the graph's sparsity can change. It may contain fewer edges for some realizations of z and more edges for other z.

Given a sample of observations $(x_i, z_i)_{i=1}^n$, the data version of (4) replaces the population expectations with their sample counterparts:

$$\begin{aligned} \min_{W \in \mathcal{F}} \quad & \frac{1}{n}\sum_{i=1}^n \|x_i - W(z_i)^\top x_i\|_2^2 \\ \text{s.t.} \quad & W(z) \in \text{DAG for all } z \in \mathbb{R}^m \\ & \frac{1}{n}\sum_{i=1}^n \|W(z_i)\|_{\ell_1} \leq \lambda. \end{aligned}$$

To realize this estimator, we take the function class $\mathcal{F} = \{W_\theta(z) : \theta \in \mathbb{R}^d\}$ with $W_\theta(z)$ a certain architecture of feedforward neural networks parameterized by weights θ. This choice results in our proposal:

$$\begin{aligned} \min_{\theta \in \mathbb{R}^d} \quad & \frac{1}{n}\sum_{i=1}^n \|x_i - W_\theta(z_i)^\top x_i\|_2^2 \\ \text{s.t.} \quad & W_\theta(z) \in \text{DAG for all } z \in \mathbb{R}^m \\ & \frac{1}{n}\sum_{i=1}^n \|W_\theta(z_i)\|_{\ell_1} \leq \lambda. \end{aligned} \tag{5}$$

Next, we present a novel neural network architecture that restricts the network's co-domain to the constraint set of (5), a difficult task in general.

2.3 Neural Network Architecture

Figure 2 illustrates our neural network architecture. The network has two main components: hidden layers and a projection layer. The hidden layers are standard linear transformations followed by non-linear activation functions (e.g., rectified linear units), except for the final hidden layer, which has no activation. The purpose of the hidden layers is to capture any nonlinear effects of the contextual features. Their output is a simple directed graph in the form of a $p \times p$ weighted adjacency matrix, denoted $\tilde{W}$. Recall that a simple directed graph has directed edges but no self-loops, so the diagonal of $\tilde{W}$ is zero. Hence, the final hidden layer has $p \times (p-1)$ neurons. The network's second major component—the projection layer—is required because it is impossible to obtain an acyclic or sparse graph using only the hidden layers.

2.4 Projection Layer

The role of the projection layer is to transform the (dense) simple directed graph $\tilde{W}$ into a (sparse) DAG $W^\star$. During training, n weighted adjacency matrices $\tilde{W}_1, \dots, \tilde{W}_n$ are projected onto the feasible set each time a forward pass is performed. This projection is a solution to a constrained optimization problem:

$$\begin{aligned} \min_{W_1,\dots,W_n \in \mathbb{R}^{p\times p}} \quad & \frac{1}{2}\sum_{i=1}^{n} \|\tilde{W}_i - W_i\|_F^2 \\ \text{s.t.} \quad & W_i \in \text{DAG}, \quad i = 1,\dots,n \\ & \frac{1}{n}\sum_{i=1}^{n} \|W_i\|_{\ell_1} \leq \lambda. \end{aligned} \tag{6}$$

The subscript F in the objective function denotes the Frobenius norm. Due to the combinatorial DAG constraint, the feasible set of (6) is non-convex. The number of possible DAGs on p nodes is super-exponential in p, limiting the scalability of exact combinatorial approaches. In our setting, where the projection operates on n weighted adjacency matrices at every forward pass, scalability issues are only further exacerbated.

Recently, Zheng et al. (2018), Bello, Aragam, and Ravikumar (2022), and others proposed continuous characterizations of acyclicity in the form of $h(W) = 0$ for some function $h(W)$ having the property

$$h(W) = 0 \iff W \in \text{DAG}.$$

In particular, Bello, Aragam, and Ravikumar (2022) showed that for all matrices W whose element-wise square $W \circ W$ has spectral radius less than $s > 0$, a well-suited choice of the function $h(W)$ is

$$h_s(W) := -\log\det(sI - W \circ W) + p\log(s). \tag{7}$$

This characterization of acyclicity, known as DAGMA, allows us to replace the combinatorial DAG constraint with its equivalent log determinant characterization:

$$\begin{aligned} \min_{W_1,\dots,W_n \in \mathbb{W}^s} \quad & \frac{1}{2}\sum_{i=1}^{n} \|\tilde{W}_i - W_i\|_F^2 \\ \text{s.t.} \quad & h_s(W_i) = 0, \quad i = 1,\dots,n \\ & \frac{1}{n}\sum_{i=1}^{n} \|W_i\|_{\ell_1} \leq \lambda, \end{aligned} \tag{8}$$

where $\mathbb{W}^s := \{W \in \mathbb{R}^{p\times p} : \rho(W \circ W) < s\}$ and $\rho(\cdot)$ is the matrix's spectral radius. Though (8) remains a non-convex problem, the constraint function $h_s(W)$ is continuous and differentiable, enabling applications of scalable first-order optimization methods.

3 PROJECTION ALGORITHMS

A forward pass through the neural network performs the projection (8) onto the intersection of a non-convex set (the log det level set) and a convex set (the ℓ_1 ball). We split this projection into two steps for efficient computation: (1) projection onto the log det level set and (2) projection onto the ℓ_1 ball. Proposition 1 states that this approach yields a feasible solution to the original problem. Its proof is in Appendix A.

Proposition 1. *Let $\tilde{W} \in \mathbb{R}^{p\times p}$, $\hat{W}$ be the projection of $\tilde{W}$ onto the* log det *level set, and $W^\star$ the projection of $\hat{W}$ onto the ℓ_1 ball. Then $W^\star$ lies on the intersection of the* log det *level set and the ℓ_1 ball.*

3.1 log det Projection

The optimization problem of the log det projection readily separates into n identical subproblems. We therefore focus on this subproblem for simplicity:

$$\begin{aligned} \min_{W \in \mathbb{W}^s} \quad & \frac{1}{2}\|\tilde{W} - W\|_F^2 \\ \text{s.t.} \quad & h_s(W) = 0. \end{aligned} \tag{9}$$

The projection (9) is a complex problem for which no analytical solution is available. Bello, Aragam, and Ravikumar (2022) devised a path-following algorithm for general log det constrained problems that shifts the constraint function to the objective and then treats the original loss function as a penalty that is progressively annealed towards zero. In our setting, the objective function corresponding to this approach is

$$f_{\mu,s}(W; \tilde{W}) := \frac{\mu}{2}\|\tilde{W} - W\|_F^2 + h_s(W), \tag{10}$$

where the coefficient $\mu > 0$ is successively decreased along the path. Lemma 6 of Bello, Aragam, and Ravikumar (2022) states that the limiting solution as $\mu \to 0$ satisfies the original log det constraint.

Algorithm 1 uses the formulation (10) to provide a method for projecting onto the log det level set.

Algorithm 1: log det projection

```
input : Adjacency matrix $\tilde{W} \in \mathbb{R}^{p \times p}$, log det
        parameter $s > 0$, path coefficient $\mu > 0$,
        decay factor $\alpha \in (0, 1)$, step count $T \in \mathbb{N}$
Set $W^{(0)} \leftarrow 0$
for $t = 0, 1, \ldots, T - 1$ do
    Initialize $W$ at $W^{(t)}$ and solve
        $W^{(t+1)} \leftarrow \underset{W \in \mathbb{W}^s}{\arg\min} \quad f_{\mu,s}(W; \tilde{W})$    (11)
    Set $\mu \leftarrow \alpha\mu$
end
output: Adjacency matrix $\hat{W} = W^{(T)}$
```

In the experiments, we set the path coefficient $\mu = 1$, decay factor $\alpha = 0.5$, and step count $T = 10$. These default values generally work well in our experience.

The performance of Algorithm 1 depends critically on efficiently solving the inner optimization problem (11) at each step along the path, i.e., minimizing the objective function (10) for a fixed value of μ. Recall that the gradient of $\log\det(X)$ is $X^{-\top}$, so that

$$\nabla h_s(W) = 2(sI - W \circ W)^{-\top} \circ W,$$

where $h_s(W)$ is defined in (7). The gradient $\nabla f_{\mu,s}(W; \tilde{W})$ of the objective function immediately follows. Using these gradients, the inner optimization problem can be solved by first-order methods such as gradient descent. To characterize the behavior of gradient descent on this problem, Theorem 1 presents its convergence properties. The proof is in Appendix B.

Theorem 1. *Let* $W^{(0)} = 0$, $\tilde{W} \in \mathbb{R}^{p \times p}$ *with* $|\tilde{\mathsf{w}}_{jk}| \leq 1$, $s \geq 1 + \max(\|\tilde{W}\|_1, \|\tilde{W}\|_\infty)$, *and* $\mu > 0$. *Define the gradient descent update*

$$W^{(k+1)} = W^{(k)} - \frac{1}{\bar{c}} \nabla f_{\mu,s}(W^{(k)}; \tilde{W}).$$

Then, for any $\bar{c} \geq c = \max(\mu/2, 2\sqrt{p} + 4p\|\tilde{W}\|_F)$, *the sequence* $\{f_{\mu,s}(W^{(k)}; \tilde{W})\}_{k \in \mathbb{N}}$ *decreases, converges, and satisfies the inequality*

$$f_{\mu,s}(W^{(k)}; \tilde{W}) - f_{\mu,s}(W^{(k+1)}; \tilde{W}) \geq \frac{\bar{c} - c}{2} \|W^{(k+1)} - W^{(k)}\|_F^2.$$

Theorem 1 establishes that gradient descent applied to the inner optimization problem (11) generates a convergent sequence of objective values. This result is not trivial to prove and relies on showing Lipschitz continuity of $f_{\mu,s}(W; \tilde{W})$ under the theorem's conditions. The condition $|\tilde{\mathsf{w}}_{jk}| \leq 1$ is readily satisfied by scaling the algorithm's inputs (and rescaling its outputs).

The step size $1/\bar{c}$ implied by Theorem 1 can be quite small in practice, slowing convergence. Likewise, the implied log det parameter s can be quite large, also slowing convergence. It is unclear if these conservative values are an artifact of the proof strategy. Our numerical experience is that a more aggressive step size with $\bar{c} = p$ and $s = 1$ typically leads to convergence.

3.2 ℓ_1 Projection

Let $\hat{W}_1, \ldots, \hat{W}_n$ be the output of the log det projection. With these matrices, the ℓ_1 projection solves

$$\begin{aligned} \min_{W_1, \ldots, W_n \in \mathbb{R}^{p \times p}} \quad & \frac{1}{2} \sum_{i=1}^{n} \|\hat{W}_i - W_i\|_F^2 \\ \text{s.t.} \quad & \frac{1}{n} \sum_{i=1}^{n} \|W_i\|_{\ell_1} \leq \lambda. \end{aligned} \tag{12}$$

It is straightforward to show that the projection (12) is solved by thresholding $\hat{W}_1, \ldots, \hat{W}_n$ element-wise using

$$S_\kappa(\mathsf{w}) := \operatorname{sign}(\mathsf{w}) \max(|\mathsf{w}| - \kappa, 0),$$

where the threshold $\kappa \geq 0$ is derived from $\hat{W}_1, \ldots, \hat{W}_n$. Duchi et al. (2008) provide a non-iterative algorithm involving only a few low-complexity operations for computing κ in the case of vector arguments. As described in Appendix C, their algorithm readily extends to matrix arguments. The computation of κ is performed only during training. For inference, the κ from the training set is used. This projection yields the neural network's final output $W_1^\star, \ldots, W_n^\star$.

4 SCALABLE COMPUTATION

Even with a first-order method for the projection layer, scaling contextual DAGs to large sample sizes requires careful treatment of the forward and backward passes through the network. We now propose some novel, efficient methods and evaluate their scalability.

4.1 Forward Pass

At each forward pass during training, the projection layer acts on all n observations in the sample. These projections, in turn, each involve inverting a $p \times p$ matrix at every gradient descent iteration, which has complexity $O(p^3)$. Though cubic complexity is typical of existing structure learning methods (e.g., Zheng et al. 2018; Bello, Aragam, and Ravikumar 2022), it makes it prohibitive to carry out the projections in sequence, as a naive implementation might do. For this reason, we implement Algorithm 1 as a batched solver that handles all n problems in parallel. Batched BLAS (Dongarra et al. 2017) and its CUDA equivalent

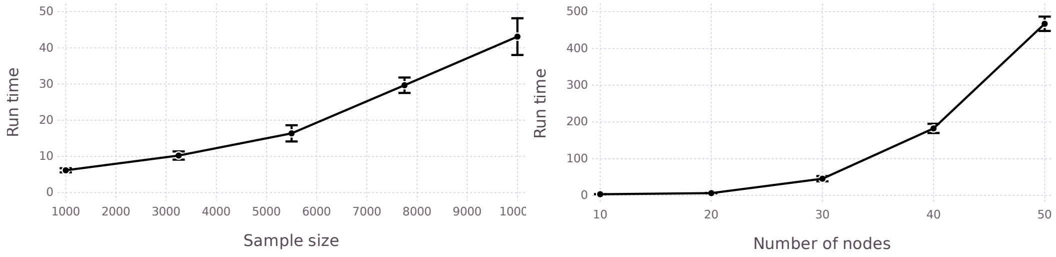


Figure 3: Run times in seconds for 10 epochs of a contextual DAG on an NVIDIA RTX 4090 over 10 synthetic datasets. The number of nodes $p = 20$ in the left plot and the sample size $n = 1000$ in the right plot. The number of contextual features $m = 2$. The solid points are averages and the error bars are one standard errors.

expose batched linear algebra routines and, critically, batched linear algebra solvers to perform parallel matrix inversions. Via these routines, our implementation can perform forward passes in reasonable run times for n in the order of thousands or tens of thousands.

4.2 Backward Pass

The second consideration during training is the backward pass, where gradients are propagated backward through the network. The presence of the projection layer complicates this step because we require the gradients of the output $W_1^\star, \ldots, W_n^\star$ from this layer with respect to the input $\tilde{W}_1, \ldots, \tilde{W}_n$ to backpropagate successfully. Despite the layer involving a complex optimization problem, Theorem 2 gives an analytical expression for its gradients. The proof is in Appendix D.

Theorem 2. *Let $\tilde{W}_1, \ldots, \tilde{W}_n \in \mathbb{R}^{p \times p}$. Define $W_1^\star, \ldots, W_n^\star \in \mathbb{R}^{p \times p}$ as a solution to the projection (8). Let $\mathcal{A} := \{(i,j,k) : \mathsf{w}_{ijk}^\star \neq 0\}$ be the active set of edges, where $\mathsf{w}_{ijk}^\star$ is the element of $W_i^\star$ at the jth row and kth column. Then, if the ℓ_1 constraint is binding (i.e., $n^{-1} \sum_{i=1}^n \|W_i^\star\|_{\ell_1} = \lambda$), the gradient of $\mathsf{w}_{ijk}^\star$ with respect to $\tilde{\mathsf{w}}_{lqr}$ is given by*

$$\frac{\partial \mathsf{w}_{ijk}^\star}{\partial \tilde{\mathsf{w}}_{lqr}} = \begin{cases} \delta_{ijk}^{lqr} - \dfrac{\operatorname{sgn}(\mathsf{w}_{lqr}^\star) \operatorname{sgn}(\mathsf{w}_{ijk}^\star)}{|\mathcal{A}|} & \text{if } (i,j,k) \in \mathcal{A} \\ 0 & \text{otherwise,} \end{cases}$$

where $\delta_{ijk}^{lqr} := 1[(i,j,k) = (l,q,r)]$ and $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$. If the ℓ_1 constraint is not binding, the gradient is given by

$$\frac{\partial \mathsf{w}_{ijk}^\star}{\partial \tilde{\mathsf{w}}_{lqr}} = \begin{cases} \delta_{ijk}^{lqr} & \text{if } (i,j,k) \in \mathcal{A} \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 2 states that the elements of $W_1^\star, \ldots, W_n^\star$ that are zero (i.e., the non-existent edges) have gradient zero, while the gradients of the remaining non-zero elements depend on their respective signs. The dependence vanishes as the number of non-zero elements grows. Importantly, the gradient only requires knowledge of the projection layer's output $W_1^\star, \ldots, W_n^\star$. Once the forward pass produces these outputs, the gradient needed for the backward pass comes with virtually no additional computation. The alternative, which is to automatically differentiate through the layer's gradient descent routine, is exceptionally expensive due the algorithm's iterative nature.

4.3 Complexity

With a fixed number of hidden-layer neurons, the forward or backward pass through the hidden layers, which input m contextual features and output a $p \times p$ matrix, takes $O(nm + np^2)$ operations. A forward pass through the projection layer takes $O(np^3)$ operations, while the backward pass only takes $O(np^2)$ operations (based on Theorem 2). The total complexity of training the network is $O(nm + np^3)$. Hence, the training time is linear in the sample size n and the number of contextual features m, and cubic in the number of nodes p. We emphasize again that cubic complexity on p is typical of continuous structure learning algorithms. Figure 3 plots the average run times for training a contextual DAG as a function of n and p. The run time as a function of n is indeed fairly linear. The run time as a function of p is non-linear but not quite cubic. Of course, cubic complexity is worst-case.

4.4 Implementation

Our algorithmic framework is provided in the `Julia` (Bezanson et al. 2017) implementation `ContextualDAG` using the deep learning library `Flux` (Innes et al. 2018). We devise a custom pathwise optimization strategy to speed up training, described in Appendix E. `ContextualDAG` is available at

`github.com/ryan-thompson/ContextualDAG.jl`.

5 RELATED WORK

To the best of our knowledge, no existing methods are available for learning DAGs with varying structure as general as our proposal. The closest work to ours is Ni, Stingo, and Baladandayuthapani (2019), whose approach also allows some components of the DAG structure to vary as a function of external features, albeit with two crucial differences. First, the approach in Ni, Stingo, and Baladandayuthapani (2019) assumes the ordering of the nodes is fixed and known by the user. In contrast, our approach implicitly learns this ordering from the data directly and does not assume it is fixed. Second, Ni, Stingo, and Baladandayuthapani (2019) learn the map from the contextual features to the DAG using splines rather than using a neural network as we do. Though splines are computationally appealing, they do not generalize well beyond low-complexity function classes (i.e., smooth functions), which can lead them to underperform relative to neural networks in practice (see, e.g., Agarwal et al. 2021).

Several other papers have explored ideas that relate more generally to contextual DAGs. Boutilier et al. (1996) studied the notion of "context-specific independence" wherein dependencies between nodes in a DAG (of fixed topological ordering) can depend on the states of other nodes. Geiger and Heckerman (1996) considered a related class of models—similarity networks and multinets—which also represent varying independencies. Oates et al. (2016) devised a method for learning subject-specific graphs whereby multiple graphs are trained simultaneously under a regularizer that penalizes distances between the graphs. A dependency network is used to describe the relations between subjects. In another line of work, Ahmed et al. (2022) proposed semantic probabilistic layers for neural networks, which could be applied to impose constraints such as acyclicity to produce context-specific DAGs. However, their approach requires the true structure during training, which is unavailable in our setting.

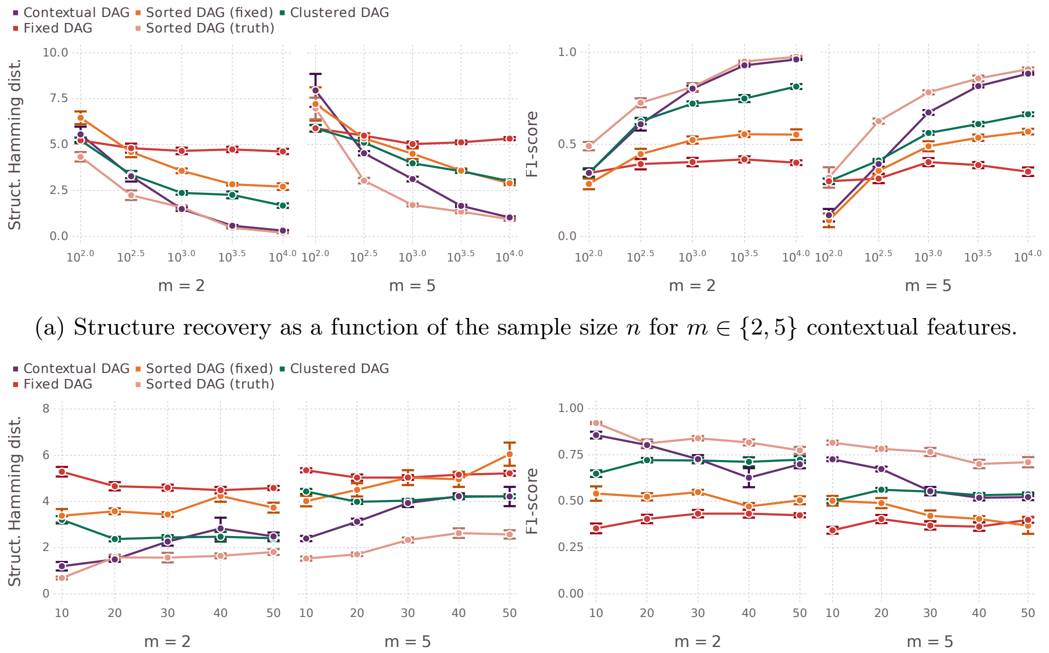
Undirected graphical models with varying structure have received more attention than their directed counterparts. See the recent papers Ni, Stingo, and Baladandayuthapani (2022), Niu et al. (2023), and Zhang and Li (2023), and references therein. Whereas we constrain our model's co-domain to acyclic matrices, undirected graphical models that vary with external features typically constrain the co-domain to positive definite matrices. This constraint is comparatively straightforward to handle as it does not require iterative algorithms that are necessary for enforcing acyclicity. These works also focus on low-complexity (linear or piecewise linear) maps from the external features to graphs, restricting their expressiveness. We are unaware of any existing work on graphical models that employ neural networks for this task. However, Thompson, Dezfouli, and Kohn (2023) showed that neural networks can successfully incorporate external features into non-graphical (sparse linear) models.

More broadly, our paper complements the literature on fixed DAG learning. Zheng et al. (2018) introduced the NOTEARS framework, which was the first to provide a continuous characterization of the DAG constraint. Subsequent work, including DAGMA (Bello, Aragam, and Ravikumar 2022)—used in this paper—as well as Ng, Ghassami, and Zhang (2020), Yu et al. (2021), and Gillot and Parviainen (2022) refined the continuous optimization approach. See also the extensions and applications in Yu et al. (2019), Lachapelle et al. (2020), Pamfil et al. (2020), Geffner et al. (2022), and Gong et al. (2023). The appeal of continuous algorithms is that they scale gracefully to large graphs and can be augmented with more complex combinatoric approaches if needed (Manzour et al. 2021; Deng et al. 2023). Our work sits parallel to this line of research. We build a neural network architecture around these algorithms to enable the otherwise fixed structure of DAGs to vary flexibly with contextual features.

6 SYNTHETIC EXPERIMENTS

Here, we evaluate our approach on synthetic data. We compare against a fixed DAG that estimates a single graph over the entire sample. We also include two "sorted DAGs" that allow the graph structure to vary but assume the ordering of the nodes is given. The first uses a fixed topological order (as in Ni, Stingo, and Baladandayuthapani 2019) taken from the estimated fixed DAG. The second uses the *true* varying topological order, representing an idealized case and a bound on what our approach can achieve. The two sorted DAGs employ the same architecture as the contextual DAG but use binary masking matrices rather than an acyclicity projection to encode the topological orderings. Finally, we include a "clustered DAG" that assigns observations on z into $\lceil n/100 \rceil$ clusters using k-means and fits a fixed DAG to each cluster. All approaches use the DAGMA acyclicity characterization of Bello, Aragam, and Ravikumar (2022). Appendix F provides the details of their implementation.

We generate synthetic datasets according to the structural equation model (2) using Erdős-Rényi and scale-free graphs where the edge weights, node ordering, and number of edges vary with the contextual features. Appendix G details the full simulation design. As structure recovery metrics, we measure the structural Hamming distance and F1-score averaged over all the graphs predicted for a testing set generated in-



(a) Structure recovery as a function of the sample size n for $m \in \{2, 5\}$ contextual features.

(b) Structure recovery as a function of the number of nodes p for $m \in \{2, 5\}$ contextual features.

Figure 4: Structure recovery performance on varying Erdős-Rényi graphs over 10 synthetic datasets. The number of nodes $p = 20$ in the top row and the sample size $n = 1000$ in the bottom row. The solid points are averages and the error bars are one standard errors. The sorted DAG (truth) uses the ground truth topological order.

dependently and identically to the training set.

Figure 4a reports the results as function of n for the Erdős-Rényi graphs. Besides the true sorted DAG, the contextual DAG is the only method to recover the context-specific graphs accurately as n grows large. Its neural network–enabled by the projection layer—is increasingly able to well-approximate the true function. The gap between the contextual DAG and true sorted DAG, representing the cost of not knowing the topological order a priori, vanishes with growing n.

The fixed sorted DAG, which allows the edge weights and sparsity of the graph to change but not the topological ordering, also improves with n but at a slower rate. However, its performance eventually plateaus well-short of the contextual DAG due to its inability to learn the varying ordering. The clustered DAG, which fits discrete DAGs to different clusters of z, likewise underperforms our continuous approach. Unsurprisingly, the fixed DAG fails to improve as n increases.

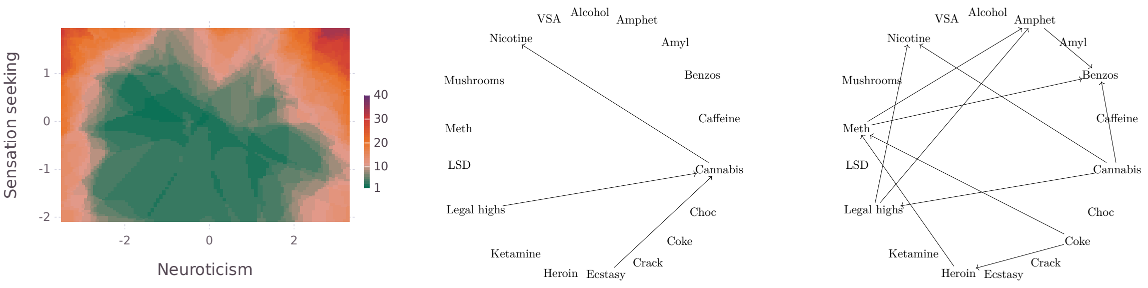
Figure 4b plots the results for Erdős-Rényi graphs as a function of p. All approaches naturally witness worsening performance as p grows. In any case, the contextual DAG remains highly competitive with both sorted DAGs, the clustered DAG, and the fixed DAG.

Appendix H reports the experiments for the scale-free graphs. Additional experiments for a setting where the contextual features are irrelevant (i.e., the fixed DAG's home court) are available in Appendix I.

7 DRUG CONSUMPTION DATASET

We consider a dataset from Fehrman et al. (2017) on the recreational drug consumption patterns of $n = 1885$ survey participants. The dataset includes measurements on the consumption of $p = 18$ illicit and non-illicit drugs in terms of recency of use. In addition to these variables, the survey also captured the participants' personality characteristics. Fehrman et al. (2017) reported neuroticism and sensation-seeking as the two most important determinants of drug consumption. Using these $m = 2$ characteristics as contextual features, we apply the contextual DAG to predict personalized graphs of consumption dependencies.

Figure 5a shows how the graph sparsity changes with the neuroticism and sensation seeking scores. Low and moderate scores yield sparser graphs than high scores, suggesting that individuals with atypical characteristics exhibit more complex and interconnected consumption patterns. Figures 5b and 5c present the actual predicted graphs for two individuals: one with low scores and another with high scores. In the high-scoring graph, "softer" drugs like cannabis act as im-



(a) Number of edges (b) Low-scoring graph (c) High-scoring graph

Figure 5: The contextual DAG from the drug consumption dataset. The left plot is the graph sparsity as a function of the neuroticism/sensation seeking scores. The other plots are the graphs with the scores at low levels (0.1 quantiles) and high levels (0.9 quantiles). The parameter λ is set to attain five edges on average over z.

portant nodes influencing the use of other substances. The edge between legal highs and cannabis is orientated differently in each graph, further indicating nuanced variation. Several edges in the high-scoring graph correspond to significant correlations identified in Fehrman et al. (2017), e.g., the edges between heroin and cocaine and methadone. The findings underscore the need for individualized risk mitigation strategies.

8 SUMMARY

Our paper introduces contextual DAGs, which relax the rigidity of fixed DAGs by allowing the graph structure to vary as a function of contextual features. A novel projection layer, for which we provide a convergence analysis and analytical gradients, allows us to learn neural networks that predict context-specific DAGs. An experimental analysis suggests that our approach can recover the true context-specific graph where other approaches fail. Our `Julia` implementation `ContextualDAG` is made publicly available.

References

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., and Hinton, G. E. (2021). "Neural additive models: Interpretable machine learning with neural nets". *Advances in Neural Information Processing Systems*. Vol. 34, pp. 4699–4711.

Ahmed, K., Teso, S., Chang, K.-W., Van den Broeck, G., and Vergari, A. (2022). "Semantic probabilistic layers for neuro-symbolic learning". *Advances in Neural Information Processing Systems*. Vol. 35, pp. 29944–29959.

Bello, K., Aragam, B., and Ravikumar, P. (2022). "DAGMA: Learning DAGs via M-matrices and a log-determinant acyclicity characterization". *Advances in Neural Information Processing Systems*. Vol. 35, pp. 8226–8239.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). "Julia: A fresh approach to numerical computing". *SIAM Review* 59 (1), pp. 65–98.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. (1996). "Context-specific independence in Bayesian networks". *Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence*, pp. 115–123.

Deng, C., Bello, K., Aragam, B., and Ravikumar, P. (2023). "Optimizing NOTEARS objectives via topological swaps". *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202, pp. 7563–7595.

Dongarra, J., Hammarling, S., Higham, N. J., Relton, S. D., Valero-Lara, P., and Zounon, M. (2017). "The design and performance of batched BLAS on modern high-performance computing systems". *Procedia Computer Science*. Vol. 108, pp. 495–504.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). "Efficient projections onto the ℓ_1-ball for learning in high dimensions". *Proceedings of the 25th International Conference on Machine Learning*, pp. 272–279.

Fehrman, E., Mirkes, E. M., Muhammad, A. K., Egan, V., and Gorban, A. N. (2017). "The five factor model of personality and evaluation of drug consumption risk". arXiv: 1506.06297.

Foster, E. M. (2010). "Causal inference and developmental psychology". *Developmental Psychology* 46 (6), pp. 1454–1480.

Geffner, T., Antorán, J., Foster, A., Gong, W., Ma, C., Kıcıman, E., Sharma, A., Lamb, A., Kukla, M., Hilmkil, A., Jennings, J., Pawlowski, N., Allamanis, M., and Zhang, C. (2022). "Deep end-to-end causal inference". *NeurIPS 2022 Workshop on Causal Machine Learning for Real-World Impact*.

Geiger, D. and Heckerman, D. (1996). "Knowledge representation and inference in similarity networks and Bayesian multinets". *Artificial Intelligence* 82.1-2, pp. 45–74.

Gillot, P. and Parviainen, P. (2022). "Learning large DAGs by combining continuous optimization and feedback arc set heuristics". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36, pp. 6713–6720.

Gong, W., Jennings, J., Zhang, C., and Pawlowski, N. (2023). "Rhino: Deep causal temporal relationship learning with history-dependent noise". *International Conference on Learning Representations*.

Huang, B., Zhang, K., Zhang, J., Ramsey, J., Sanchez-Romero, R., Glymour, C., and Schölkopf, B. (2020). "Causal discovery from heterogeneous/nonstationary data". *Journal of Machine Learning Research* 21.89, pp. 1–53.

Imbens, G. W. (2020). "Potential outcome and directed acyclic graph approaches to causality: Relevance for empirical practice in economics". *Journal of Economic Literature* 58 (4), pp. 1129–1179.

Innes, M. J., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M. C., Joy, N. M., Karmali, T., Pal, A., and Shah, V. B. (2018). "Fashionable modelling with Flux". *Workshop on Systems for ML and Open Source Software at NeurIPS 2018*.

Kingma, D. P. and Ba, J. L. (2015). "Adam: A method for stochastic optimization". *International Conference on Learning Representations*.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. (2020). "Gradient-based neural DAG learning". *International Conference on Learning Representations*.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). "Local computations with probabilities on graphical structures and their application to expert systems". *Journal of the Royal Statistical Society: Series B (Methodological)* 50 (2), pp. 157–224.

Manzour, H., Küçükyavuz, S., Wu, H.-H., and Shojaie, A. (2021). "Integer programming for learning directed acyclic graphs from continuous data". *INFORMS Journal on Optimization* 3 (1), pp. 46–73.

Meyer, C. D. (2000). *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Minsker, S. and Wang, L. (2022). "Robust estimation of covariance matrices: Adversarial contamination and beyond". arXiv: 2203.02880.

Murphy, K. P. (2023). *Probabilistic Machine Learning. Advanced Topics*. Cambridge, MA, USA: MIT Press.

Ng, I., Ghassami, A., and Zhang, K. (2020). "On the role of sparsity and DAG constraints for learning linear DAGs". *Advances in Neural Information Processing Systems*. Vol. 33, pp. 17943–17954.

Ni, Y., Stingo, F. C., and Baladandayuthapani, V. (2019). "Bayesian graphical regression". *Journal of the American Statistical Association* 114 (525), pp. 184–197.

– (2022). "Bayesian covariate-dependent Gaussian graphical models with varying structure". *Journal of Machine Learning Research* 23, pp. 1–29.

Niu, Y., Ni, Y., Pati, D., and Mallick, B. K. (2023). "Covariate-assisted Bayesian graph learning for heterogeneous data". *Journal of the American Statistical Association*, In press.

Oates, C. J., Smith, J. Q., Mukherjee, S., and Cussens, J. (2016). "Exact estimation of multiple directed acyclic graphs". *Statistics and Computing* 26 (4), pp. 797–811.

Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Beaumont, P., Georgatzis, K., and Aragam, B. (2020). "DYNOTEARS: Structure learning from time-series data". *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. Vol. 108, pp. 1595–1605.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann.

Tennant, P. W. G., Murray, E. J., Arnold, K. F., Berrie, L., Fox, M. P., Gadd, S. C., Harrison, W. J., Keeble, C., Ranker, L. R., Textor, J., Tomova, G. D., Gilthorpe, M. S., and Ellison, G. T. H. (2021). "Use of directed acyclic graphs (DAGs) to identify confounders in applied health research: Review and recommendations". *International Journal of Epidemiology* 50 (2), pp. 620–632.

Thompson, R., Dezfouli, A., and Kohn, R. (2023). "The contextual lasso: Sparse linear models via deep neural networks". arXiv: 2302.00878.

Varah, J. M. (1975). "A lower bound for the smallest singular value of a matrix". *Linear Algebra and its Applications* 11 (1), pp. 3–5.

Vowels, M. J., Camgoz, N. C., and Bowden, R. (2022). "D'ya like DAGs? A survey on structure learning and causal discovery". *ACM Computing Surveys* 55 (4), pp. 1–36.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). "DAG-GNN: DAG structure learning with graph neural networks". *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97, pp. 7154–7163.

Yu, Y., Gao, T., Yin, N., and Ji, Q. (2021). "DAGs with no curl: An efficient DAG structure learning approach". *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139, pp. 12156–12166.

Zhang, J. and Li, Y. (2023). "High-dimensional Gaussian graphical regression models with covariates". *Journal of the American Statistical Association* 118 (543), pp. 2088–2100.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018). “DAGs with NO TEARS: Continuous optimization for structure learning”. *Advances in Neural Information Processing Systems*. Vol. 31.

Zhou, F., He, K., and Ni, Y. (2022). “Causal discovery with heterogeneous observational data”. *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*. Vol. 180, pp. 2383–2393.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A PROOF OF PROPOSITION 1

Proof. For the proposition to hold, (1) the ℓ_1 projection must not introduce any cycles and (2) $W^\star$ must be an element of the set $\mathbb{W}_s$. The first requirement is satisfied because the ℓ_1 projection only modifies non-zero elements and therefore cannot add cycles. To show the second requirement is satisfied, we require Lemma 1.

Lemma 1. *Let $A = (\mathsf{a}_{jk}) \in \mathbb{R}^{p\times p}$ and $B = (\mathsf{b}_{jk}) \in \mathbb{R}^{p\times p}$ with $0 \leq \mathsf{b}_{jk} \leq \mathsf{a}_{jk}$. Then it holds $\rho(B) \leq \rho(A)$, where $\rho(\cdot)$ is the spectral radius of its argument.*

Proof. The Collatz-Wielandt formula for a non-negative matrix A (see, e.g., Meyer 2000, §8.3) gives that

$$\rho(A) = \max_{x\in\mathbb{R}^p:x\geq 0,x\neq 0}\ \min_{j:\mathsf{x}_j\neq 0} \frac{(Ax)_j}{\mathsf{x}_j}. \tag{13}$$

Now, since B is bounded elementwise by A, we have $(Bx)_j \leq (Ax)_j$ for any non-negative vector $x \neq 0$, and hence

$$\frac{(Bx)_j}{\mathsf{x}_j} \leq \frac{(Ax)_j}{\mathsf{x}_j},$$

for all $\mathsf{x}_j \neq 0$. It follows immediately from the above inequality that

$$\min_{j:\mathsf{x}_j\neq 0} \frac{(Bx)_j}{\mathsf{x}_j} \leq \min_{j:\mathsf{x}_j\neq 0} \frac{(Ax)_j}{\mathsf{x}_j}.$$

Taking the maximum over x on both sides gives

$$\max_{x\in\mathbb{R}^p:x>0,x\neq 0}\ \min_{j:\mathsf{x}_j\neq 0} \frac{(Bx)_j}{\mathsf{x}_j} \leq \max_{x\in\mathbb{R}^p:x>0,x\neq 0}\ \min_{j:\mathsf{x}_j\neq 0} \frac{(Ax)_j}{\mathsf{x}_j}.$$

By the Collatz-Wielandt formula (13), the quantity on the left-hand side is $\rho(B)$ and the quantity on the right-hand side is $\rho(A)$. Hence, $\rho(B) \leq \rho(A)$. □

With Lemma 1 in hand, we take $A = \hat{W} \circ \hat{W}$ and $B = W^\star \circ W^\star$ to get $\rho(W^\star \circ W^\star) \leq \rho(\hat{W} \circ \hat{W})$. Moreover, since $\rho(\hat{W} \circ \hat{W}) < s$, it holds $\rho(W^\star \circ W^\star) < s$ and hence $W^\star \in \mathbb{W}_s$. □

B PROOF OF THEOREM 1

B.1 Preliminary Lemmas

The proof requires several technical lemmas which we state and prove in turn. The first lemma shows that $\nabla f_{\mu,s}(W;\tilde{W})$ is Lipschitz over a compact set of W. The second lemma provides that the gradient descent sequence $\{W^{(k)}\}_{k\in\mathbb{N}}$ stays in a compact set when the step size is suitably small. The third lemma ensures descent under Lipschitz continuity.

To facilitate exposition, we write $f_{\mu,s}(W,\tilde{W})$ as $f(W) := l(W) + h(W)$, where $l(W) := \mu/2\|W - \tilde{W}\|_F^2$ and $h(W)$ is defined as in (7). The gradient $\nabla f(W) = \nabla l(W) + \nabla h(W)$, where $\nabla l(W) = -\mu/2(\tilde{W} - W)$ and $\nabla h(W) = 2(sI - W \circ W)^{-\top} \circ W$. Except for the Frobenius norm, all norms below are induced matrix norms.

Lemma 2. *Let $\tilde{W} \in \mathbb{R}^{p\times p}$ with $|\tilde{\mathsf{w}}_{jk}| \leq 1$ and $\mu \geq 0$. Define $\mathcal{W} := \{W \in \mathbb{R}^{p\times p} : |\mathsf{w}_{jk}| \leq |\tilde{\mathsf{w}}_{jk}|\}$. Then, for any $s \geq 1 + \max(\|\tilde{W}\|_1, \|\tilde{W}\|_\infty)$, the gradient $\nabla f(W) : \mathcal{W} \to \mathbb{R}^{p\times p}$ is Lipschitz continuous with respect to the Frobenius norm and has Lipschitz constant $c = \max(\mu/2, 2\sqrt{p} + 4p\|\tilde{W}\|_F)$.*

Proof. Recall that a function $g(W) : \mathcal{W} \to \mathbb{R}^{p\times p}$ is Lipschitz with respect to the Frobenius norm if, for all W_1 and W_2 in $\mathcal{W}$, it holds

$$\|g(W_1) - g(W_2)\|_F \leq c\|W_1 - W_2\|_F, \tag{14}$$

for some $c \geq 0$.

Now, if $\nabla l(W)$ and $\nabla h(W)$ are Lipschitz with constants c_1 and c_2, then $\nabla f(W)$ is also Lipschitz with constant $c = \max(c_1, c_2)$. Fix any $W_1 \in \mathcal{W}$ and any $W_2 \in \mathcal{W}$.

It is trivial to show that $\nabla l(W)$ satisfies the Lipschitz inequality (14), since

$$\begin{aligned}\|\nabla l(W_1) - \nabla l(W_2)\|_F &= \left\| -\frac{\mu}{2}(\tilde{W} - W_1) + \frac{\mu}{2}(\tilde{W} - W_2) \right\|_F \\ &= \frac{\mu}{2}\|W_1 - W_2\|_F.\end{aligned}$$

Hence, $\nabla l(W)$ is Lipschitz with constant $c_1 = \mu/2$.

Next, for $\nabla h(W)$, we have

$$\begin{aligned}\|\nabla h(W_1) - \nabla h(W_2)\|_F &= \|2(sI - W_1 \circ W_1)^{-\top} \circ W_1 - 2(sI - W_2 \circ W_2)^{-\top} \circ W_2\|_F \\ &= 2\|(sI - W_1 \circ W_1)^{-\top} \circ W_1 - (sI - W_2 \circ W_2)^{-\top} \circ W_2\|_F.\end{aligned}$$

To simplify the notation, we let $A_i = (sI - W_i \circ W_i)^\top$ for $i = 1, 2$. Observe now that

$$\begin{aligned}2\|A_1^{-1} \circ W_1 - A_2^{-1} \circ W_2\|_F &= 2\|A_1^{-1} \circ W_1 - A_1^{-1} \circ W_2 + A_1^{-1} \circ W_2 - A_2^{-1} \circ W_2\|_F \\ &\leq 2\|A_1^{-1} \circ W_1 - A_1^{-1} \circ W_2\|_F + 2\|A_1^{-1} \circ W_2 - A_2^{-1} \circ W_2\|_F \\ &\leq 2\|A_1^{-1}\|_F\|W_1 - W_2\|_F + 2\|W_2\|_F\|A_1^{-1} - A_2^{-1}\|_F.\end{aligned} \tag{15}$$

The first and second inequalities follow from the Frobenius norm being sub-additive and sub-multiplicative, respectively.

Consider the first term on the far right-hand side of (15). The requirements that $s \geq 1 + \max(\|\tilde{W}\|_1, \|\tilde{W}\|_\infty)$ and $W_1 \in \mathcal{W}$ mean that A_1 is row-wise and column-wise diagonally dominant with a dominance factor of at least one. Then Corollary 2 of Varah (1975) applies and gives $\|A_1^{-1}\|_2 \leq 1$. This result in combination with the inequality $\|A_1^{-1}\|_F \leq \sqrt{p}\|A_1^{-1}\|_2$ yields

$$\|A_1^{-1}\|_F\|W_1 - W_2\|_F \leq \sqrt{p}\|W_1 - W_2\|_F. \tag{16}$$

Consider now the second term on the far right-hand side of (15). It holds

$$\begin{aligned}\|W_2\|_F\|A_1^{-1} - A_2^{-1}\|_F &= \|W_2\|_F\|A_1^{-1}(A_1 - A_2)A_2^{-1}\|_F \\ &\leq \|W_2\|_F\|A_1^{-1}\|_F\|A_2^{-1}\|_F\|A_1 - A_2\|_F \\ &\leq p\|W_2\|_F\|A_1 - A_2\|_F \\ &= p\|W_2\|_F\|W_1 \circ W_1 - W_2 \circ W_2\|_F \\ &\leq 2p\|W_2\|_F\|W_1 - W_2\|_F.\end{aligned}$$

The last inequality follows from the fact that W_1 and W_2 have elements bounded in absolute value by one, since

$$\begin{aligned}\|W_1 \circ W_1 - W_2 \circ W_2\|_F &= \sqrt{\sum_{j=1}^{p}\sum_{k=1}^{p}(\mathsf{w}_{1jk}^2 - \mathsf{w}_{2jk}^2)^2} \\ &= \sqrt{\sum_{j=1}^{p}\sum_{k=1}^{p}(\mathsf{w}_{1jk} + \mathsf{w}_{2jk})^2(\mathsf{w}_{1jk} - \mathsf{w}_{2jk})^2} \\ &\leq \sqrt{\sum_{j=1}^{p}\sum_{k=1}^{p}4(\mathsf{w}_{1jk} - \mathsf{w}_{2jk})^2} \\ &= 2\|W_1 - W_2\|_F.\end{aligned}$$

Now, using that $\|W\|_F \leq \|\tilde{W}\|_F$ for any $W \in \mathcal{W}$, it follows

$$\|W_2\|_F\|A_1^{-1} - A_2^{-1}\|_F \leq 2p\|\tilde{W}\|_F\|W_1 - W_2\|_F. \tag{17}$$

Plugging the bounds (16) and (17) into (15), we arrive at

$$\|\nabla h(W_1) - \nabla h(W_2)\|_F \leq (2\sqrt{p} + 4p\|\tilde{W}\|_F)\|W_1 - W_2\|_F.$$

Hence, $\nabla h(W)$ is Lipschitz with constant $c_2 = 2\sqrt{p} + 4p\|\tilde{W}\|_F$.

The claim of the lemma now follows since $\nabla f(W)$ must also be Lipschitz with constant $c = \max(c_1, c_2) = \max(\mu/2, 2\sqrt{p} + 4p\|\tilde{W}\|_F)$. □

Lemma 3. *Let $\tilde{W} \in \mathbb{R}^{p\times p}$ with $|\tilde{\mathsf{w}}_{jk}| \leq 1$, $\mu \geq 0$, and $c \geq \max(\mu/2, 1)$. Define $\mathcal{W} := \{W \in \mathbb{R}^{p\times p} : |\mathsf{w}_{jk}| \leq |\tilde{\mathsf{w}}_{jk}|\}$. Then, for all $W^{(k)} \in \mathcal{W}$ and any $s \geq 1 + \max(\|\tilde{W}\|_1, \|\tilde{W}\|_\infty)$, it holds $W^{(k+1)} \in \mathcal{W}$, where*

$$W^{(k+1)} = W^{(k)} - \frac{1}{c}\nabla f(W^{(k)}).$$

Proof. Assume without loss of generality that all elements of $\tilde{W}$ are non-negative. We aim to show that for any $W^{(k)} \in \mathcal{W} = \{W \in \mathbb{R}^{p\times p} : 0 \leq \mathsf{w}_{jk} \leq \tilde{\mathsf{w}}_{jk}\}$, the gradient descent update $W^{(k+1)}$ is also in $\mathcal{W}$. We break the proof into two parts: (1) we prove that $\mathsf{w}_{jk}^{(k+1)} \leq \tilde{\mathsf{w}}_{jk}$ and (2) we prove that $\mathsf{w}_{jk}^{(k+1)} \geq 0$.

Fix any $\mathcal{W}^{(k)} \in \mathcal{W}$. For the upper bound (1), we have

$$\begin{aligned}
W^{(k+1)} &= W^{(k)} - \frac{1}{c}\nabla f(W^{(k)}) \\
&= W^{(k)} - \frac{1}{c}\left(\nabla l(W^{(k)}) + \nabla h(W^{(k)})\right) \\
&\leq W^{(k)} - \frac{1}{c}\nabla l(W^{(k)}) \\
&= W^{(k)} + \frac{\mu}{2c}(\tilde{W} - W^{(k)}) \\
&\leq W^{(k)} + \tilde{W} - W^{(k)} \\
&= \tilde{W}.
\end{aligned} \tag{18}$$

The first inequality follows from $\nabla h(W^{(k)}) \geq 0$, since $sI - W^{(k)} \circ W^{(k)}$ is a non-singular M matrix and hence its inverse is non-negative. The last inequality follows from $\mu/(2c) \leq 1$ and the elements of $\tilde{W} - W^{(k)}$ being non-negative, since $W^{(k)} \in \mathcal{W}$.

For the lower bound (2), it holds

$$\begin{aligned}
W^{(k+1)} &= W^{(k)} - \frac{1}{c}\nabla f(W^{(k)}) \\
&= W^{(k)} - \frac{1}{c}\left(\nabla l(W^{(k)}) + \nabla h(W^{(k)})\right) \\
&= W^{(k)} - \frac{1}{c}\left(\frac{\mu}{2}(W^{(k)} - \tilde{W}) + \nabla h(W^{(k)})\right) \\
&= W^{(k)} + \frac{\mu}{2c}(\tilde{W} - W^{(k)}) - \frac{1}{c}\nabla h(W^{(k)}) \\
&\geq W^{(k)} - \frac{1}{c}\nabla h(W^{(k)}) \\
&\geq W^{(k)} - \nabla h(W^{(k)}) \\
&= W^{(k)} - (sI - W^{(k)} \circ W^{(k)})^{-\top} \circ W^{(k)}.
\end{aligned}$$

The first inequality follows again from $\tilde{W} - W^{(k)} \geq 0$. The second inequality follows from $1/c \leq 1$ and $\nabla h(W^{(k)})$ having non-negative elements. Now, observe that for a matrix $B = (\mathsf{b}_{jk}) \in \mathbb{R}^{p\times p}$, it holds

$$|\mathsf{b}_{jk}| = |e^{(j)\top} B e^{(k)}| \leq \|e^{(j)}\|_2 \|Be^{(k)}\|_2 = \|Be^{(k)}\|_2 \leq \max_{x\in\mathbb{R}^p : \|x\|_2 = 1} \|Bx\|_2 = \|B\|_2,$$

where $e^{(j)}$ is a standard basis vector (one in the jth position and zero elsewhere). The last equality follows from the definition of the spectral norm. Letting $B = (sI - W^{(k)} \circ W^{(k)})^{-\top}$, we have $|\mathsf{b}_{jk}| \leq \|B\|_2 \leq 1$ along the same lines as the argument for (16) in Lemma 2. Combining this result with the previous inequality on $W^{(k+1)}$ gives

$$\begin{aligned}
W^{(k+1)} &\geq W^{(k)} - (sI - W^{(k)} \circ W^{(k)})^{-\top} \circ W^{(k)} \\
&\geq 0.
\end{aligned} \tag{19}$$

Combining the inequalities (18) and (19) gives the desired bound $0 \leq W^{(k+1)} \leq \tilde{W}$. □

Lemma 4. *Let $g(W) : \mathbb{R}^{p\times p} \to \mathbb{R}$. Suppose $\nabla g(W)$ is Lipschitz in Frobenius norm with Lipschitz constant $c > 0$. Then, for any $W \in \mathbb{R}^{p\times p}$ and any $V \in \mathbb{R}^{p\times p}$, it holds*

$$g(V) \leq g(W) + \langle \nabla g(W), V - W \rangle + \frac{c}{2}\|V - W\|_F^2,$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius inner product.

Proof. See the proof of Lemma 10 in Minsker and Wang (2022). □

B.2 Proof of Main Result

We now prove Theorem 1.

Proof. Take $\bar{c} \geq c$ as an upper bound to the Lipschitz constant c. The objective $f(W)$ can be lower bounded as

$$\begin{aligned} f(W) &= f(W) + \langle \nabla f(W), W - W \rangle + \frac{\bar{c}}{2}\|W - W\|_F^2 \\ &\geq \inf_{V \in \mathbb{R}^{p\times p}} \left(f(W) + \langle \nabla f(W), V - W \rangle + \frac{\bar{c}}{2}\|V - W\|_F^2 \right) \\ &= \inf_{V \in \mathbb{R}^{p\times p}} \left(f(W) - \frac{1}{2\bar{c}}\|\nabla f(W)\|_F^2 + \frac{\bar{c}}{2}\left\| V - \left(W - \frac{1}{\bar{c}}\nabla f(W) \right) \right\|_F^2 \right). \end{aligned}$$

Observe that the infimum is attained at the gradient descent update

$$\hat{W} = W - \frac{1}{\bar{c}}\nabla f(W).$$

Substituting $\hat{W}$ into the previous inequality gives

$$\begin{aligned} f(W) &\geq f(W) - \frac{1}{2\bar{c}}\|\nabla f(W)\|_F^2 + \frac{\bar{c}}{2}\left\| \hat{W} - \left(W - \frac{1}{\bar{c}}\nabla f(W) \right) \right\|_F^2 \\ &= f(W) + \langle \nabla f(W), \hat{W} - W \rangle + \frac{\bar{c}}{2}\|\hat{W} - W\|_F^2 \\ &= f(W) + \langle \nabla f(W), \hat{W} - W \rangle + \frac{c}{2}\|\hat{W} - W\|_F^2 + \frac{\bar{c} - c}{2}\|\hat{W} - W\|_F^2. \end{aligned} \tag{20}$$

Lemmas 2 and 3 give that $\nabla f(W)$ is Lipschitz under the conditions of the theorem. This result is sufficient to invoke Lemma 4 and lower bound the first three terms on the right-hand side as

$$f(W) + \langle \nabla f(W), \hat{W} - W \rangle + \frac{c}{2}\|\hat{W} - W\|_F^2 \geq f(\hat{W}). \tag{21}$$

Substituting (21) into (20) yields

$$f(W) \geq f(\hat{W}) + \frac{\bar{c} - c}{2}\|\hat{W} - W\|_F^2.$$

Finally, taking $W = W^{(k)}$ and $\hat{W} = W^{(k+1)}$, we arrive at

$$f(W^{(k)}) - f(W^{(k+1)}) \geq \frac{\bar{c} - c}{2}\|W^{(k+1)} - W^{(k)}\|_F^2. \tag{22}$$

Hence, the sequence $\{f(W^{(k)})\}_{k\in\mathbb{N}}$ is decreasing, and because $f(W)$ is bounded below by zero, it converges. □

C ℓ_1 PROJECTION ALGORITHM

Algorithm 2 provides the method for projecting onto the ℓ_1 ball by computing the thresholding parameter κ.

Algorithm 2: ℓ_1 projection

```
input  : Adjacency matrices $\hat{W}_1, \ldots, \hat{W}_n \in \mathbb{R}^{p \times p}$, regularization parameter $\lambda > 0$
Set $v \leftarrow [\text{vec}(\hat{W}_1), \ldots, \text{vec}(\hat{W}_n)]$
Take $u$ as the absolute values of $v$
Sort $u$ in decreasing order as $\mathsf{u}_j > \mathsf{u}_k$ for all $j < k$
Set $k_{\max} \leftarrow \max \left\{ k : \mathsf{u}_k > \left( \sum_{j=1}^{k} \mathsf{u}_j - n\lambda \right) / k \right\}$
Set $\kappa \leftarrow \left( \sum_{j=1}^{k_{\max}} \mathsf{u}_j - n\lambda \right) / k_{\max}$
Compute $W_1^\star, \ldots, W_n^\star$ as $\mathsf{w}_{ijk}^\star \leftarrow \text{sgn}(\hat{\mathsf{w}}_{ijk}) \max(|\hat{\mathsf{w}}_{ijk}| - \kappa, 0)$ for $i = 1, \ldots, n$ and $j, k = 1, \ldots, p$
output: Adjacency matrices $W_1^\star, \ldots, W_n^\star$
```

The algorithm directly extends that of Duchi et al. (2008) for projecting a vector $\hat{v} \in \mathbb{R}^p$ onto the ℓ_1 ball:

$$\min_{v \in \mathbb{R}^p : \|v\|_1 \leq \gamma} \frac{1}{2} \|\hat{v} - v\|_2^2. \tag{23}$$

Our modified algorithm simply flattens the matrices into a vector and then reshapes the result of the projection. This approach is valid since (23) is equivalent to

$$\min_{W_1, \ldots, W_n \in \mathbb{R}^{p \times p} : \frac{1}{n} \sum_{i=1}^{n} \|W_i\|_{\ell_1} \leq \lambda} \frac{1}{2} \sum_{i=1}^{n} \|\hat{W}_i - W\|_F^2$$

when $\hat{v} = [\text{vec}(\hat{W}_1), \ldots, \text{vec}(\hat{W}_n)]$, $v = [\text{vec}(W_1), \ldots, \text{vec}(W_n)]$, and $\gamma = n\lambda$.

D PROOF OF THEOREM 2

Proof. To simplify exposition of the proof, we assume $n = 1$ without loss of generality. We consider the cases where the ℓ_1 constraint is binding and non-binding in turn.

Non-Binding ℓ_1 Constraint

In the non-binding case where $\|W^\star\|_{\ell_1} < \lambda$, a solution $W^\star$ sets some elements of $\tilde{W}$ to zero via the DAG constraint and leaves the remaining elements untouched. There is no shrinkage from the ℓ_1 constraint.

The $\mathsf{w}_{jk}^\star$ that are non-zero are an identity function of the input $\tilde{\mathsf{w}}_{qr}$ when $(q, r) = (j, k)$ and are a null function otherwise. More precisely, $\mathsf{w}_{jk}^\star(\tilde{\mathsf{w}}_{qr}) = \tilde{\mathsf{w}}_{qr}$ for $(q, r) = (j, k)$ in the active set $\mathcal{A}$. The elements $\tilde{\mathsf{w}}_{qr}$ for which $(q, r) \neq (j, k)$ have no effect on $\mathsf{w}_{jk}^\star$, so $\mathsf{w}_{jk}^\star(\tilde{\mathsf{w}}_{qr}) = 0$ for $(q, r) \neq (j, k)$. It follows

$$\frac{\partial \mathsf{w}_{jk}^\star}{\partial \tilde{\mathsf{w}}_{qr}} = \delta_{jk}^{qr}, \tag{24}$$

for all $(j, k) \in \mathcal{A}$, where δ_{jk}^{qr} equals one if $(j, k) = (q, r)$ and zero otherwise.

The elements (j, k) in the inactive set $\mathcal{A}^c$, where $\mathcal{A}^c$ is the complement of $\mathcal{A}$, are a null function in $\tilde{\mathsf{w}}_{qr}$, i.e., $\mathsf{w}_{jk}^\star(\tilde{\mathsf{w}}_{qr}) = 0$, even if $(q, r) = (j, k)$. It follows

$$\frac{\partial \mathsf{w}_{jk}^\star}{\partial \tilde{\mathsf{w}}_{qr}} = 0, \tag{25}$$

for all $(j, k) \in \mathcal{A}^c$.

Combining (24) and (25) yields the gradient for the non-binding case.

Binding ℓ_1 Constraint

For the binding case where $\|W^\star\|_{\ell_1} = \lambda$, we derive the gradients by differentiating through the Karush-Kuhn-Tucker (KKT) conditions of (8), which characterize a solution in terms of $\tilde{W}$. We need not consider the DAG constraint in our analysis of these conditions since it only determines the active set and does not have any effect on the magnitude of the edge weights, unlike the ℓ_1 constraint.

Let ν be the dual variable corresponding to the constraint $\|W\|_{\ell_1} \leq \lambda$. Like the primal solution $W^\star$, the dual solution $\nu^\star$ can be treated a function of $\tilde{W}$, i.e., $\nu^\star = \nu^\star(\tilde{\mathsf{w}}_{qr})$. To simplify the presentation, we omit the dependence hereafter and write $\nu^\star$ and the same for $W^\star$ (or $\mathsf{w}^\star_{jk}$).

Recall that the ℓ_1-norm is not differentiable but subdifferentiable. We denote the subderivative of a function $f(x)$ as $\partial f(x)$. With this notation, the KKT conditions for stationarity and complementary slackness are

$$\partial\left(\frac{1}{2}\|\tilde{W} - W^\star\|_F^2 + \nu^\star(\|W^\star\|_{\ell_1} - \lambda)\right) \ni 0 \tag{26}$$

and

$$\nu^\star(\|W^\star\|_{\ell_1} - \lambda) = 0. \tag{27}$$

We consider separately the gradients on the active set $\mathcal{A}$ and the gradients on the inactive set $\mathcal{A}^c$. We begin by deriving the gradients on $\mathcal{A}$. It follows immediately from the complementary slackness condition (27) that

$$\|W^\star\|_{\ell_1} - \lambda = \sum_{(j,k)\in\mathcal{A}} |\mathsf{w}^\star_{jk}| - \lambda = 0,$$

and hence its gradient with respect to $\tilde{\mathsf{w}}_{qr}$ is

$$\frac{\partial}{\partial\tilde{\mathsf{w}}_{qr}}\left(\sum_{(j,k)\in\mathcal{A}} |\mathsf{w}^\star_{jk}| - \lambda\right) = 0.$$

Evaluating the derivative on the left-hand side yields

$$\sum_{(j,k)\in\mathcal{A}} \operatorname{sgn}(\mathsf{w}^\star_{jk})\frac{\partial \mathsf{w}^\star_{jk}}{\partial\tilde{\mathsf{w}}_{qr}} = 0. \tag{28}$$

Now, evaluating the subderivative on the left-hand side of the stationarity condition (26) leads to the equalities

$$\mathsf{w}^\star_{jk} - \tilde{\mathsf{w}}_{jk} + \nu^\star \operatorname{sgn}(\mathsf{w}^\star_{jk}) = 0,$$

which hold for all $(j,k) \in \mathcal{A}$. Differentiating this expression with respect to $\tilde{\mathsf{w}}_{qr}$ gives

$$\frac{\partial \mathsf{w}^\star_{jk}}{\partial\tilde{\mathsf{w}}_{qr}} - \delta^{qr}_{jk} + \frac{\partial\nu^\star}{\partial\tilde{\mathsf{w}}_{qr}}\operatorname{sgn}(\mathsf{w}^\star_{jk}) + \nu^\star\frac{\partial}{\partial\tilde{\mathsf{w}}_{qr}}\operatorname{sgn}(\mathsf{w}^\star_{jk}) = 0,$$

which simplifies to

$$\frac{\partial \mathsf{w}^\star_{jk}}{\partial\tilde{\mathsf{w}}_{qr}} = \delta^{qr}_{jk} - \frac{\partial\nu^\star}{\partial\tilde{\mathsf{w}}_{qr}}\operatorname{sgn}(\mathsf{w}^\star_{jk}). \tag{29}$$

Substituting (29) into (28) leads to

$$\sum_{(j,k)\in\mathcal{A}} \operatorname{sgn}(\mathsf{w}^\star_{jk})\left(\delta^{qr}_{jk} - \frac{\partial\nu^\star}{\partial\tilde{\mathsf{w}}_{qr}}\operatorname{sgn}(\mathsf{w}^\star_{jk})\right) = 0,$$

from which it follows

$$\operatorname{sgn}(\mathsf{w}^\star_{qr}) - \frac{\partial\nu^\star}{\partial\tilde{\mathsf{w}}_{qr}}\sum_{(j,k)\in\mathcal{A}} \operatorname{sgn}(\mathsf{w}^\star_{jk})^2 = 0.$$

Again, rearranging and simplifying, we have

$$\frac{\partial\nu^\star}{\partial\tilde{\mathsf{w}}_{qr}} = \frac{\operatorname{sgn}(\mathsf{w}^\star_{qr})}{\sum_{(j,k)\in\mathcal{A}}\operatorname{sgn}(\mathsf{w}^\star_{jk})^2} = \frac{1}{|\mathcal{A}|}\operatorname{sgn}(\mathsf{w}^\star_{qr}). \tag{30}$$

Substituting (30) into (29), we obtain

$$\frac{\partial \mathsf{w}_{jk}^{\star}}{\partial \tilde{\mathsf{w}}_{qr}} = \delta_{jk}^{qr} - \frac{1}{|\mathcal{A}|} \operatorname{sgn}(\mathsf{w}_{qr}^{\star}) \operatorname{sgn}(\mathsf{w}_{jk}^{\star}), \tag{31}$$

for all $(j,k) \in \mathcal{A}$. This quantity is the gradient of the projection's output $\mathsf{w}_{jk}^{\star}$ with respect to its input $\tilde{\mathsf{w}}_{qr}$ for the $\mathsf{w}_{jk}^{\star}$ that are non-zero.

Finally, for the gradients on $\mathcal{A}^c$, we see from (27) that the complementary slackness condition does not involve the $\mathsf{w}_{jk}^{\star}$ such that $(j,k) \in \mathcal{A}^c$. The stationarity condition for $(j,k) \in \mathcal{A}^c$ is

$$-\tilde{\mathsf{w}}_{jk} + \frac{\partial \nu^{\star}}{\partial \tilde{\mathsf{w}}_{qr}} s \ni 0,$$

where $s = [-1, 1]$ is the subderivative of the absolute value function at zero. Hence, the stationarity condition also does not involve $\mathsf{w}_{jk}^{\star}$. It follows that

$$\frac{\partial \mathsf{w}_{jk}^{\star}}{\partial \tilde{\mathsf{w}}_{qr}} = 0, \tag{32}$$

for all $(j,k) \in \mathcal{A}^c$. This quantity is the gradient of the projection's output $\mathsf{w}_{jk}^{\star}$ with respect to its input $\tilde{\mathsf{w}}_{qr}$ for the $\mathsf{w}_{jk}^{\star}$ that are zero.

Combining (31) and (32) yields the gradient for the binding case. □

E PATHWISE OPTIMIZATION

It is typical to compute a sequence of graphs corresponding to different sparsity levels from which the user can select. We take λ as a sequence $\{\lambda^{(t)}\}_{t=1}^{T}$, where $\lambda^{(0)}$ imposes no regularization and $\lambda^{(T)}$ imposes full regularization. Except in degenerate cases, the unregularized model with $\lambda = \lambda^{(0)}$ contains $(p^2 - p)/2$ edges, which is the maximum number of edges that an acyclic graph can have. The fully regularized model with $\lambda = \lambda^{(T)}$ contains no edges. Rather than compute these T models independently, we compute them in a pathwise manner by sequentially warm-starting the optimizer. Specifically, the model for $\lambda^{(t+1)}$ is trained using the fitted weights $\hat{\theta}^{(t)}$ from the model for $\lambda^{(t)}$ as an initialization point.

Algorithm 3 presents the details of the pathwise optimization approach.

Algorithm 3: Pathwise optimization

```
input  : Initial network weights $\hat{\theta}^{(0)} \in \mathbb{R}^d$, step size $\varphi > 0$, number of regularization parameters $T \in \mathbb{N}$
Initialize $\lambda^{(1)} \leftarrow \infty$
for $t = 1, \ldots, T$ do
    Initialize $\theta_{(0)} \leftarrow \hat{\theta}^{(t-1)}$
    Initialize $m \leftarrow 0$
    while Not converged do
        Update $\theta_{(m+1)} \leftarrow \theta_{(m)} - \varphi \nabla_{\theta} L(\theta_{(m)}; \lambda^{(t)})$
        Update $m \leftarrow m + 1$
    end
    Set $\hat{\theta}^{(t)} \leftarrow \theta_{(m)}$
    if $t$=1 then
        Set $\lambda^{(1)} \leftarrow n^{-1} \sum_{i=1}^{n} \|W_{\hat{\theta}^{(1)}}(z_i)\|_{\ell_1}$ and $\lambda^{(T)} = 0$
        Interpolate $T - 2$ points between $\lambda^{(1)}$ and $\lambda^{(T)}$
    end
end
output: Fitted network weights $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(T)}$
```

To unpack the notation, $L(\theta; \lambda) = n^{-1} \sum_{i=1}^{n} \|x_i - W_{\theta}(z_i)^{\top} x_i\|_2^2$ is the loss function evaluated at weights θ and regularization parameter λ. Its gradient is denoted $\nabla_{\theta} L(\theta; \lambda)$. The benefit of pathwise optimization is a significantly reduced overall runtime. Typically, it only takes a small number of iterations to converge if $\lambda^{(t+1)}$ and $\lambda^{(t)}$ are similar.

The initial network weights $\hat{\theta}^{(0)}$ for Algorithm 3 are computed by training a preliminary neural network that contains no projection layer. This preliminary network is initialized with random weights and is quick to train. Using the resulting trained weights as $\hat{\theta}^{(0)}$ yields better performance than directly taking $\hat{\theta}^{(0)}$ as random weights.

F IMPLEMENTATION OF METHODS

All neural networks are implemented using two hidden layers, each containing 128 neurons with rectified linear activation functions. For the drug dataset, we use 32 neurons in each hidden layer due to the smaller sample size. Adam (Kingma and Ba 2015) is used as an optimizer for learning the network weights, with a learning rate of 0.001. Convergence is monitored on a validation set with the optimizer terminated after 10 iterations without improvement.

Following Bello, Aragam, and Ravikumar (2022), we apply a final thresholding step to all methods that sets near-zero edge weights to exact zeros, where the threshold is chosen to guarantee the graphs are acyclic. This step is necessary for continuous structure learning approaches such as NOTEARS and DAGMA because gradient descent and related first-order methods seldom produce machine-precision zeros.

For *all methods*, we sweep the regularization parameter λ over a grid of 20 values and take the final λ that best approximates the correct sparsity level. This approach fairly compares graphs of the same complexity and ensures that the results are not confounded by uncertainty arising from model selection. In this regard, we follow earlier works such as Zheng et al. (2018) and Bello, Aragam, and Ravikumar (2022) that also forgo empirically estimating λ.

The experiments are performed on a Linux platform with two NVIDIA RTX 4090s.

G DESIGN OF SYNTHETIC DATASETS

We create synthetic datasets of n observations as follows. First, we generate the contextual features $z_1,\ldots,z_n$ by taking n iid draws uniformly on $[-1,1]^m$. The noise $\varepsilon_1,\ldots,\varepsilon_n$ is also generated by taking n iid draws from a p-dimensional $N(0,I)$. The variables $x_1,\ldots,x_n$ are then generated as

$$x_i=(I-W(z_i))^{-\top}\varepsilon_i,$$

which follows from rearranging terms in the model $x_i=W(z_i)^\top x_i+\varepsilon_i$. This process of sampling $z_1,\ldots,z_n$ and $x_1,\ldots,x_n$ is repeated three times to obtain independent training, validation, and testing sets.

The function $W(z)$ is constructed by randomly sampling an Erdős-Rényi or scale-free graph with 10 edges. This undirected graph is then oriented according to z_i as follows. For each node j, we sample a point c_j on $[-1,1]^m$. If the graph contains an edge between node j and node k, we direct it from j to k if

$$\|z_i-c_j\|_2-\|z_i-c_k\|_2>\phi,$$

and remove it otherwise. This scheme imposes a topological ordering where node j is higher than node k in the ordering if $\|z_i-c_j\|_2>\|z_i-c_k\|_2$. The parameter $\phi\geq 0$ controls the expected sparsity of the graph (i.e., the average number of active edges). Larger values of ϕ encourage a sparser graph (in expectation), while smaller values have the opposite effect. We set ϕ empirically so that 5 of the 10 edges are active in the graph on average.

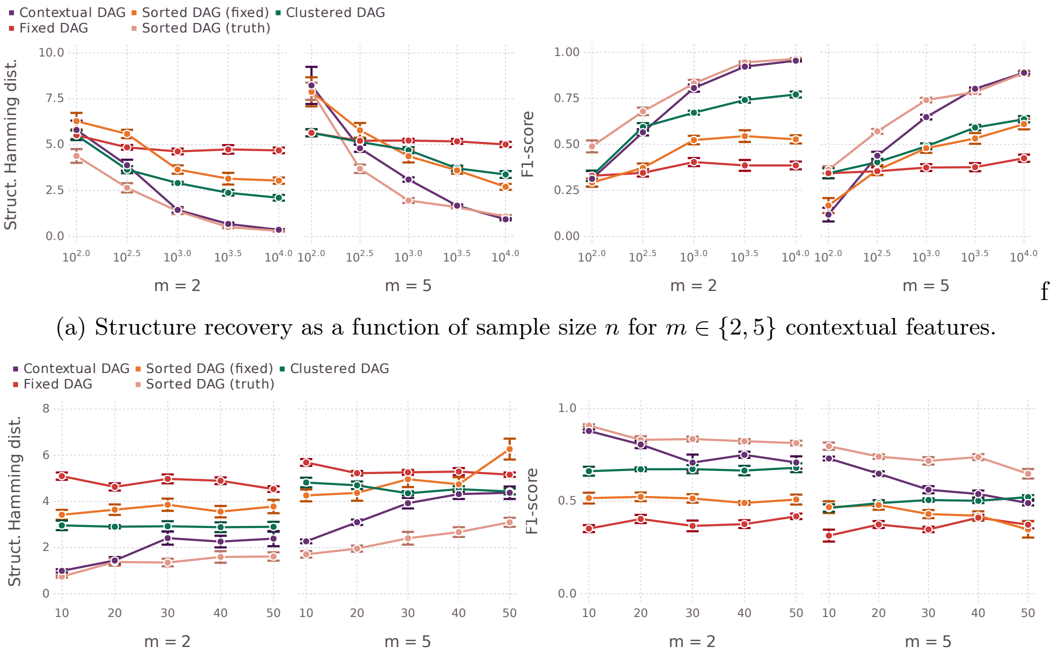
The edge weight w_{jk} is also allowed to vary with z when non-zero. Specifically, for all edges between nodes j and k, we set

$$\mathsf{w}_{jk}(z)=\begin{cases}\|z_i-c_j\|_2-\|z_i-c_k\|_2 & \text{if } \|z_i-c_j\|_2-\|z_i-c_k\|_2>\phi\\ 0 & \text{otherwise.}\end{cases}$$

For all other pairs of nodes j and k, we set $\mathsf{w}_{jk}(z)=0$.

H SCALE-FREE RESULTS

Figure 6 reports the scale-free graph results, analogous to the Erdős-Rényi graphs in Section 6. The findings are quite similar to those for the Erdős-Rényi graphs—the contextual DAG is highly competitive at structure recovery. Its consistent performance across both settings indicates that it is reliable for multiple graph types.



(a) Structure recovery as a function of sample size n for $m \in \{2, 5\}$ contextual features.

(b) Structure recovery as a function of number of nodes p for $m \in \{2, 5\}$ contextual features.

Figure 6: Structure recovery performance on varying scale-free graphs over 10 synthetic datasets. The number of nodes $p = 20$ in the top row and the sample size $n = 1000$ in the bottom row. The solid points are averages and the error bars are one standard errors. The sorted DAG (truth) uses the ground truth topological sort.

I FIXED GRAPH RESULTS

Figure 7 reports the fixed graph results where the contextual features are irrelevant, i.e., $W(z)$ is constant in z. Although the inductive bias of methods that assume a single population DAG does have an impact, the contextual DAG remains competitive and still recovers the ground truth for large n despite z having no predictive value

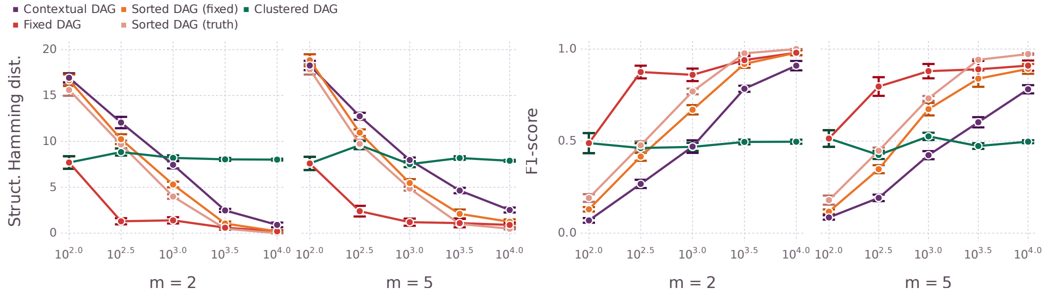


Figure 7: Structure recovery performance on fixed Erdős-Rényi graphs over 10 synthetic datasets. The x-axis is sample size n. The number of contextual features $m \in \{2, 5\}$ and the number of nodes $p = 20$. The solid points are averages and the error bars are one standard errors. The sorted DAG (truth) uses the ground truth topological sort.