DAG LEARNING ON THE PERMUTAHEDRON

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce *Daguerro*, a strategy for learning directed acyclic graphs (DAGs). In contrast to previous methods, our problem formulation (i) guarantees to learn a DAG, (ii) does not propagate errors over multiple stages, and (iii) can be trained end-to-end without pre-processing steps. Our algorithm leverages advances in differentiable sparse structured inference for learning a total ordering of the variables in the simplex of permutation vectors (the permutahedron), allowing for a substantial reduction in memory and time complexities w.r.t. existing permutation-based continuous optimization methods.

1 INTRODUCTION

Learning a directed acyclic graph (DAG) from observational data is very challenging yet useful for many applications, e.g., in biology (Sachs et al., 2005), genetics (Zhang et al., 2013), or finance (Sanford & Moosa, 2012). The challenge comes from the combinatorial nature of the solution space, whose size grows super-exponentially with the number of variables d. While exact scored-based algorithms exist for small d (Singh & Moore, 2005; Xiang & Kim, 2013; Cussens, 2011), approximate methods (Scanagatta et al., 2015; Aragam & Zhou, 2015; Ramsey et al., 2017) rely on global or local search heuristics in order to scale to problems with thousands of nodes. For instance, a large body of recent works (Zheng et al., 2018; Yu et al., 2019; Zheng et al., 2020; Ng et al., 2020; Brouillard et al., 2020; He et al., 2021) formulate DAG learning as a continuous optimization problem, where the acyclicity constraint is expressed as a smooth function and relaxed to allow efficient optimization with multi-purpose solvers. As such, the absence of cycles is no longer guaranteed and solutions often require post-processing.

Assuming identifiability of the underlying graph structure, another prominent line of works (Friedman & Koller, 2003; Gao et al., 2020; Reisach et al., 2021; Cundy et al., 2021) learns DAGs by (i) finding an ordering of the variables, and (ii) selecting the best scoring graph among (or marginalizing over) the structures that are consistent with the found ordering. This allows to work on the space of orderings which is smaller and more regular than the space of DAGs (Friedman & Koller, 2003) while guaranteeing acyclicity. Existing works, however, are based either on two-steps procedures (Gao et al., 2020; Reisach et al., 2021) that do not guarantee optimality of the solution, or on end-to-end ones that, however, involve higher computational costs (Friedman & Koller, 2003; Cundy et al., 2021). In particular, the recent work of Cundy et al. (2021) models a total ordering distribution on the Birkhoff polytope (the convex hull of permutation matrices) to obtain a differentiable operator, which has $O(d^3)$ time and $O(d^2)$ memory complexities.

In this work, we propose *Daguerro* an end-to-end score-based strategy that belongs to the latter family of works. Contrary to Cundy et al. (2021), we build our method on the SparseMAP operator (Niculae et al., 2018) for learning the total ordering of the variables. This choice of differentiable operator allows us to work in the simplex of permutation vectors (a.k.a. Permutahedron) with significant gains in time ($O(d \log d)$) and space ($O(d)$) complexities (Blondel et al., 2020). Preliminary results on synthetic data show that *Daguerro* generally recovers the topological sort of the nodes but tends to return overly dense graphs.

2 METHOD

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix consisting of n i.i.d. observations for d nodes, where we denote the j-th column of $\mathbf{X}$ by $\boldsymbol{x}_j \in \mathbb{R}^n$, and $\mathcal{G} \in \mathbb{D}$ a DAG, where $\mathbb{D}$ denotes the discrete space of DAGs

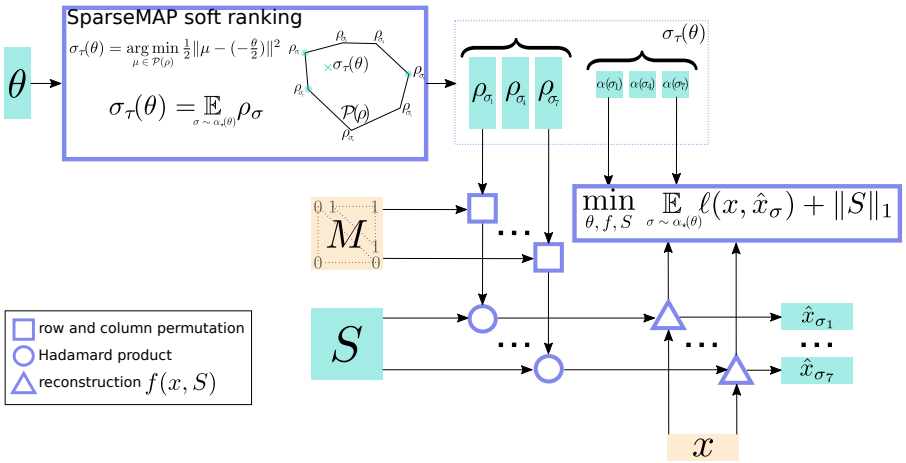

Figure 1: Overview of *Daguerro*.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on d nodes. We represent $\mathcal{G}$ by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$, where an element $\mathbf{A}_{ij} \neq 0$ iff a directed edge exists from node i to node j.

Learning DAGs via ranking and masking We formulate our score-based problem as follows

$$\min_{\boldsymbol{\theta}, \phi} \sum_{j=1}^{d} \ell\left(\mathbf{x}_j, f_j^{\phi}\left(\mathbf{X}, \mathbf{A}(\boldsymbol{\theta})_j\right)\right) \quad s.t.\ \mathbf{A}(\boldsymbol{\theta}) \in \mathbb{D}, \tag{1}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a point-wise loss function and $\{f^{\phi}\}_{j=1}^{d}$ is a set of functions parameterized by $\boldsymbol{\phi} \in \mathbb{R}^{d_{\boldsymbol{\phi}}}$ that computes $\mathbf{x}_j$ given its parents $pa(j) = \{i \in [d] \mid \mathbf{A}(\boldsymbol{\theta})_{ij} \neq 0\}$. Our strategy to find a solution for Equation (1) is to enforce the constraint $\mathbf{A}(\boldsymbol{\theta}) \in \mathbb{D}$ by learning a total ordering of the variables and masking out all edges that are not consistent with it: if $\boldsymbol{x}_j \prec \boldsymbol{x}_i$ no edge can be drawn from i to j. As this procedure returns only transitive closures, we further encourage the removal of unnecessary edges via l_1 regularization (as later defined in eq. (7)).

Let Σ_d denote the set of all d-permutations. Given a full and unconstrained weighted adjacency matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$, a parametrized permutation $\sigma(\boldsymbol{\theta}) \in \Sigma_d$, and an upper strictly triangular masking matrix $\mathbf{M} \in \{0, 1\}^{d \times d}$ such that $\mathbf{M}_{ij} = 1$ if $d < i < j$ and 0 otherwise, we decompose $\mathbf{A}(\boldsymbol{\theta})$ as:

$$\mathbf{A}(\boldsymbol{\theta}) = \mathbf{S} \circ \mathbf{M}_{\sigma(\boldsymbol{\theta})}\,, \tag{2}$$

where $\circ$ denotes the Hadamard product, and $\mathbf{M}_{\sigma(\boldsymbol{\theta})}$ indicates that the rows and columns of $\mathbf{M}$ have been permuted according to $\sigma(\boldsymbol{\theta})$.

Soft ranking on the permutahedron. In contrast to Cundy et al. (2021) that operates in the Birkhoff polytope, to learn the optimal permutation we turn to an efficient formulation of ranking as constrained optimization on the permutahedron, following Blondel et al. (2020). The ranking of a vector $\boldsymbol{\theta} \in \mathbb{R}^d$, i.e., the permutation $\sigma \in \Sigma_d$ that arranges the elements of $\boldsymbol{\theta}$ in decreasing order, can be written as (Blondel et al., 2020, Lemma 1)

$$\sigma(\boldsymbol{\theta}) = \arg\max_{\sigma \in \Sigma_d} \langle \boldsymbol{\theta}, \boldsymbol{\rho}_\sigma \rangle\,, \quad \text{where } \boldsymbol{\rho} = (d, d-1, \ldots, 1)\,. \tag{3}$$

The permutahedron of a vector $\boldsymbol{w} \in \mathbb{R}^d$ is the convex hull of all possible permutations of $\boldsymbol{w}$:

$$\mathcal{P}(\boldsymbol{w}) = \operatorname{conv}\{\boldsymbol{w}_\sigma : \sigma \in \Sigma_d\}\,. \tag{4}$$

Blondel et al. (2020) show that projection onto $\mathcal{P}(\boldsymbol{\rho})$ yields a soft ranking operator:

$$\sigma_\tau(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\mu} \in \mathcal{P}(\boldsymbol{\rho})} \frac{\tau}{2}\|\boldsymbol{\mu} - (-\boldsymbol{\theta})\|^2 = \arg\min_{\boldsymbol{\mu} \in \mathcal{P}(\boldsymbol{\rho})} \frac{1}{2}\|\boldsymbol{\mu} - (-\boldsymbol{\theta}/\tau)\|^2\,. \tag{5}$$

This mapping is differentiable w.r.t. $\boldsymbol{\theta}$ and, in the absence of ties, as the regularization parameter τ goes to zero, $\sigma_\tau(\boldsymbol{\theta}) \to \sigma(\boldsymbol{\theta})$ (Blondel et al., 2020, Proposition 2). An algorithm based on isotonic regression permits $O(d \log d)$ computation of $\sigma_\tau(\boldsymbol{\theta})$. However, soft rankings are not suitable for our goal, since we cannot replace $\mathbf{M}_{\sigma(\boldsymbol{\theta})}$ by $\mathbf{M}_{\sigma_\tau(\boldsymbol{\theta})}$. In fact, the latter is only defined for the vertices of the permutahedron, i.e., discrete permutations $\rho \in \Sigma_d$. The same problem arises when working with soft permutation matrices, as done in Cundy et al. (2021). There, the authors resort to the straight-through gradient estimator (Bengio et al., 2013), which boils down to (i) finding a hard permutation matrix in the forward pass by solving an optimal matching program via the Hungarian algorithm (Kuhn, 2010), and (ii) computing the gradients of its soft counterpart obtained through the Sinkhorn operator (Sinkhorn, 1964) for the backward pass. This strategy however induces additional bias in the training procedure.

SparseMAP distributions over permutations. To handle the issue that our loss can only be evaluated meaningfully at discrete permutations, we note that any $\boldsymbol{\mu} \in \mathcal{P}(\boldsymbol{\rho})$ can be written as a convex combination of vertices,

$$\boldsymbol{\mu} = \sum_{\sigma \in \Sigma_d} \alpha(\sigma) \boldsymbol{\rho}_\sigma = \mathbb{E}_{\sigma \sim \alpha}[\boldsymbol{\rho}_\sigma]\,, \qquad \text{where } \alpha(\cdot) \geq 0, \text{ and } \sum_{\sigma \in \Sigma_d} \alpha(\sigma) = 1\,. \tag{6}$$

SparseMAP (Niculae et al., 2018) is a strategy for differentiable structured inference that solves problems of the form of eq. (5), yielding not only the optimal solution $\sigma_\tau(\boldsymbol{\theta}) = \boldsymbol{\mu}_*$, but also a sparse decomposition $\alpha_*(\boldsymbol{\theta})$, such that $\boldsymbol{\mu}_* = \mathbb{E}_{\sigma \sim \alpha_*}[\boldsymbol{\rho}_\sigma]$. This is achieved via an active set algorithm, which maintains a sparse decomposition, and iteratively adds or removes permutations based on linear oracles of the form (3), implemented by a sorting algorithm. This iterative algorithm is more costly than the direct isotonic regression one, but provides the decomposition α_*, which allows us to compute exact marginalizations of the form $\mathbb{E}_{\sigma \sim \alpha_*(\boldsymbol{\theta})}[f(\sigma)]$ by explicitly summing over the sparse set of permutations found by SparseMAP, as proposed for structured latent variable learning by Correia et al. (2020).

DAG learning on the permutahedron. Our final bi-level optimization problem is:

$$\min_{\boldsymbol{\theta}, \mathbf{S}, \phi} \mathbb{E}_{\sigma \sim \alpha_*(\boldsymbol{\theta})} \left[\sum_{j=1}^{d} \ell(\mathbf{x}_j, f_j^{\boldsymbol{\phi}}(\mathbf{X}, \mathbf{S} \circ \mathbf{M}_\sigma)) \right] + \lambda \|\mathbf{S}\|_1 \tag{7}$$

$$s.t.\ \mathbb{E}_{\sigma \sim \alpha_*(\boldsymbol{\theta})}[\boldsymbol{\rho}_\sigma] = \operatorname*{arg\,min}_{\boldsymbol{\mu} \in \mathcal{P}(\boldsymbol{\rho})} \frac{1}{2} \|\boldsymbol{\mu} - (-\boldsymbol{\theta}/\tau)\|^2 \tag{8}$$

The inner problem eq. (8) is optimized via SparseMAP, which returns a sparse categorical distribution α_* and its gradients w.r.t. $\boldsymbol{\theta}$. We can solve the outer optimization via proximal adaptive gradient methods. A schematic of the overall approach is presented in Figure 1.

Computational details. Each SparseMAP iteration involves a length-d argsort and a Cholesky update of a s-by-s matrix, where s is the size of the active set, bounded by the iteration number. With a constant number of iterations (as in our implementation), this leads to an overall time complexity of $O(sd \log d + s^2)$ and space complexity $O(sd)$. If run until exact convergence, Carathéodory's convex hull theorem guarantees $s \leq d + 1$ leading to cubic worst-case complexity; however, in practice, we almost always reach convergence within 100 iterations. In addition, we warm-start the sorting algorithm with the last selected permutation. This is better both in theory and in practice than the $O(d^3)$ complexity of maximization over the Birkhoff polytope. A tighter analysis or specialized algorithm based on isotonic regression is a promising direction for future work.

3 Experiments

In this section, we report a set of preliminary experiments to validate the proposed method for learning DAGs from observational data. We consider synthetic settings with simulated linear additive noise models. Following common practices from the literature (Zheng et al., 2018; Gao et al., 2020), we generate a random graph with $2 * d$ expected edges according to the Erdös-Rényi, Scale-Free or BiPartite models and assign to each edge a weight uniformly drawn in $[-2, -0.5] \cup [0.5, 2]$. We then

sample data according to the linear model $XA + \varepsilon$, where $\varepsilon \in \mathbb{R}^d$ is an exogenous random variable whose elements are independently distributed as $\mathrm{Normal}(0, 0.5)$ (**Gaussian**), $\mathrm{Exponential}(0.5)$ (**Exponential**) or $\mathrm{Gumbel}(0, 0.5)$ (**Gumbel**).

As baselines, we consider state-of-the-art approximate score-based methods[1]: *NoTears* (Zheng et al., 2018), the first continuous optimization method, which optimizes the Frobenius reconstruction loss and where the DAG constrain is enforced via the Augmented Lagrangian approach; *Golem* (Ng et al., 2020), another continuous optimization method which optimizes the data likelihood (under Gaussian equal variance error assumptions) regularized by *NoTears*'s DAG penalty; *NPVAR* (Gao et al., 2020), an iterative algorithm that learns topological layers and then prunes edges based on node residual variance (with Generalized Additive Models regressor backend to estimate conditional variance); *sortnregress* (Reisach et al., 2021), another two-steps strategy that orders nodes by increasing variance and selects the parents of a node among all its predecessors using the Least Angle Regressor (Efron et al., 2004). Before evaluation, we post-process the graphs found by *NoTears* and *Golem* by iteratively removing edges ordered by increasing weight until obtaining a DAG, as they often contain cycles. For our method, we instantiate $f_j^{\phi}(\mathbf{X}, \mathbf{A}(\boldsymbol{\theta})_j) = \mathbf{X}\mathbf{A}(\boldsymbol{\theta})_j$ and optimize the data likelihood (under Gaussian equal variance error assumptions).

We set the hyper-parameters of all methods to their default values, apart from *sortnregress* that uses the Bayesian Information Criterion for model selection. In particular, for *Daguerro* we set $\lambda = 0.1$, $\tau = 1$ and remove all edges with weights ≤ 0.3 as commonly done in the literature. We additionally apply a l_2 regularization to all learned parameters (and set its hyper-parameter to 0.01) as we find it stabilizes training.

In Figure 2, we compare the methods by two metrics the Structural Hamming Distance (*SHD*) between true and estimated graphs, as standard in the literature; and the Topological Ordering Pearson Correlation (*TOPC*) that we introduce to measure how well a method is able to estimate the true topological ordering of the nodes. We believe it is essential to study both metrics because the *SHD* alone does not distinguish between wrong edges that are consistent with the variable ordering from wrong edges that are not. For instance, the transitive closure of a sparse graph has high *SHD* even though it preserves the true ordering of the nodes.

In terms of *TOPC*, we observe that *Daguerro* generally provides better orderings than those of *NoTears* and *NPVAR*; it sightly improves over *sortnregress* and it is on-par with *Golem* on two out of three noise models. In terms of *SHD*, we find that *Daguerro* never provides the best performance, although it sometimes improves over *NoTears* and *NPVAR*. Given these results, we believe that a better strategy for pruning edges is key to improve our method. Additional improvement could come from a careful tuning of our hyper-parameters.

4 Conclusion and Future Work

In this work, we presented *Daguerro*, an efficient score-based method for end-to-end learning of directed acyclic graphs and structural equation models. *Daguerro* reliably finds correct total ordering, yet tends to yield too dense graphs. We anticipate that incorporating a l_0 regularization term (Louizos et al., 2018) could help mitigate this latter issue. Future directions includes adapting our strategy to non-linear SEMs (less amenable to continuous optimization approaches) and experimenting with interventional data. While our current implementation uses SparseMAP, other recent strategies for structured discrete distributions such as SST (Paulus et al., 2020) or I-MLE (Niepert et al., 2021) could offer compelling alternatives to optimize the structure parameters.

References

Bryon Aragam and Qing Zhou. Concave penalized estimation of sparse gaussian bayesian networks. *The Journal of Machine Learning Research*, 16, 2015.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, 2013.

[1] Unfortunately, we were not able to run the code of the method most related to ours: BCDnets (Cundy et al., 2021) due to existing issues with the publicly available code-base at the time of submission.

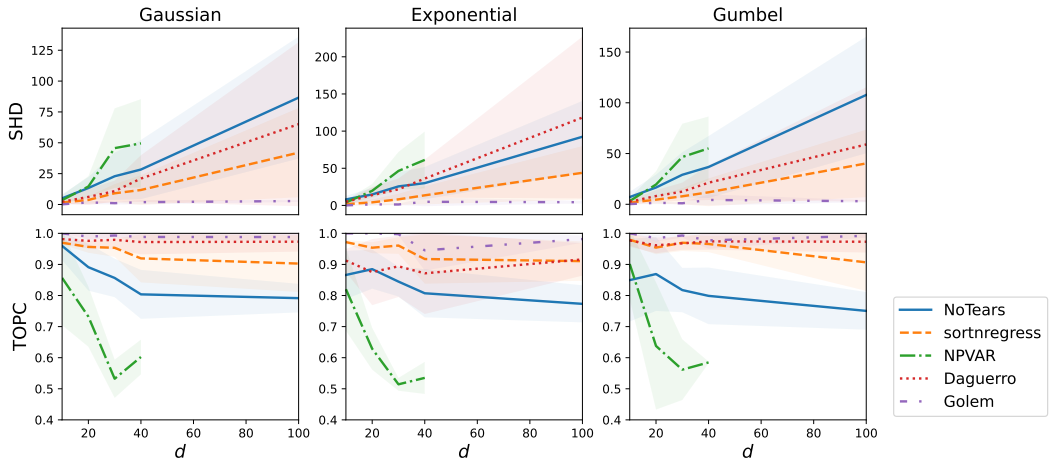

Figure 2: Structural Hamming Distance (SHD, the lower the better) and Topological Ordering Pearson Correlation (TOPC, the higher the better) as a function of the number of nodes d and for three different noise models (in column) and $n = 1000$ training points (which are not enough to run *NPVAR* with $d = 100$). We average results over the three graph models, where for each setting we ran the methods on 5 different datasets. Detailed results for specific graph models are reported in Appendix A.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pp. 950–959. PMLR, 2020.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *NeurIPS*, 2020.

Gonçalo Correia, Vlad Niculae, Wilker Aziz, and André Martins. Efficient marginalization of discrete and structured latent variables via sparsity. *Advances in Neural Information Processing Systems*, 33, 2020.

Chris Cundy, Aditya Grover, and Stefano Ermon. Bcd nets: Scalable variational approaches for bayesian causal discovery. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32, 2004.

Nir Friedman and Daphne Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50, 2003.

Ming Gao, Yi Ding, and Bryon Aragam. A polynomial-time algorithm for learning nonparametric causal graphs. In *Advances in Neural Information Processing Systems*, 2020.

Yue He, Peng Cui, Zheyan Shen, Renzhe Xu, Furui Liu, and Yong Jiang. DARING: differentiable causal discovery with residual independence. In *KDD*, 2021.

Harold W. Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming*. Springer, 2010.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *ICLR*, 2018.

Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and DAG constraints for learning linear dags. In *NeurIPS*, 2020.

Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning*, pp. 3799–3808. PMLR, 2018.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit mle: Backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34, 2021.

Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems*, 33:5691–5704, 2020.

Joseph D. Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3, 2017.

Alexander G Reisach, Christof Seiler, and Sebastian Weichwald. Beware of the simulated dag! varsortability in additive noise models. *Advances in Neural Information Processing Systems*, 34, 2021.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529, 2005. doi: 10.1126/science.1105809. URL https://www.science.org/doi/abs/10.1126/science.1105809.

Andrew D Sanford and Imad A Moosa. A bayesian network structure for operational risk modelling in structured finance operations. *Journal of the Operational Research Society*, 63(4):431–444, 2012.

Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Ajit P Singh and Andrew W Moore. *Finding optimal Bayesian networks by dynamic programming*. Citeseer, 2005.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35, 1964.

Jing Xiang and Seyoung Kim. A* lasso for learning a sparse bayesian network structure for continuous variables. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In *ICML*, volume 97, 2019.

Bin Zhang, Chris Gaiteri, Liviu-Gabriel Bodea, Zhi Wang, Joshua McElwee, Alexei A Podtelezhnikov, Chunsheng Zhang, Tao Xie, Linh Tran, Radu Dobrin, et al. Integrated systems approach identifies genetic nodes and networks in late-onset alzheimer's disease. *Cell*, 153(3):707–720, 2013.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Dags with NO TEARS: continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, 2018.

Xun Zheng, Chen Dan, Bryon Aragam, Pradeep Ravikumar, and Eric Xing. Learning sparse nonparametric dags. In *International Conference on Artificial Intelligence and Statistics*, pp. 3414–3425. PMLR, 2020.

A DETAILED RESULTS

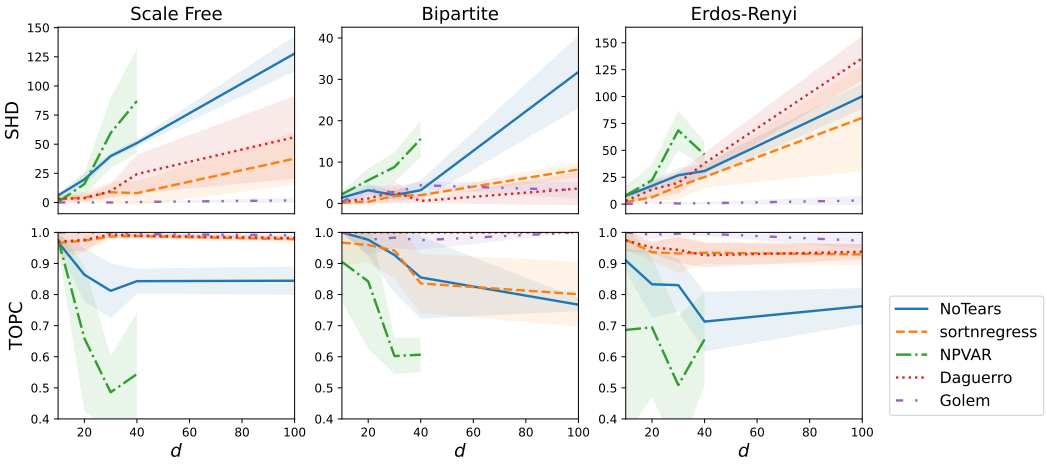

Figure 3: Detailed results with *Gaussian* noise model.

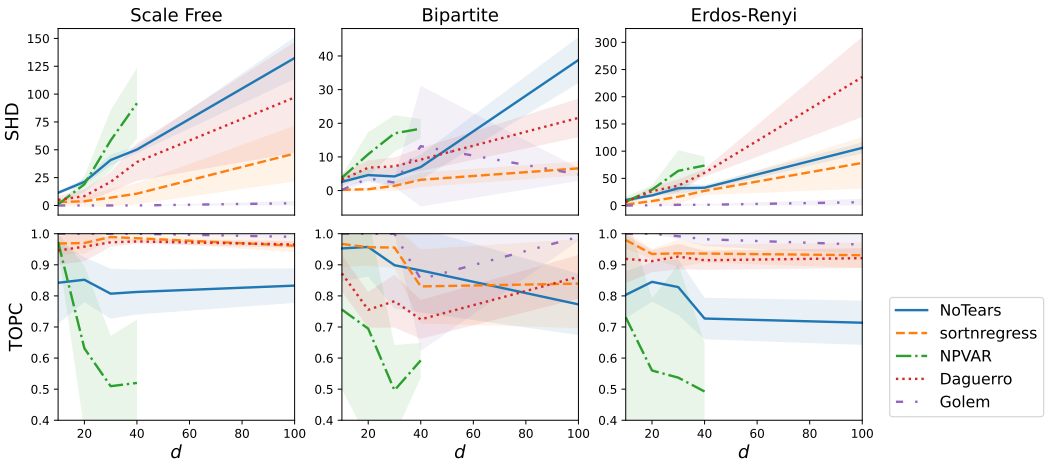

Figure 4: Detailed results with *Exponential* noise model.

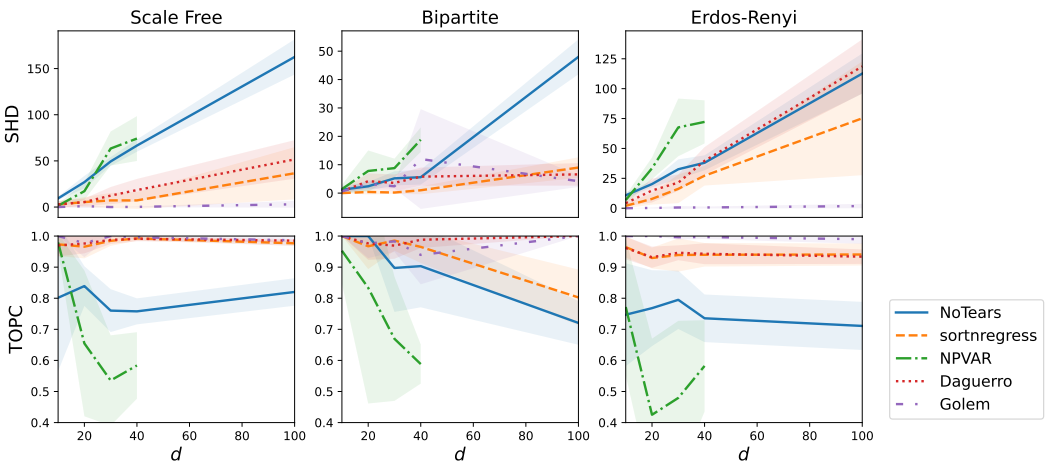

Figure 5: Detailed results with *Gumbel* noise model.