DAGs with NO TEARS: Continuous Optimization for Structure Learning

Daniel Severo, Cem Anil
University of Toronto, Vector Institute

Abstract

We introduce and explain how one can cast and solve structure learning (discovering structural causal model topologies) as a continuous optimization problem. The techniques described were proposed by Zheng et al. (2018) in the paper "DAGs with NOTEARS". The technical contribution of this work is the introduction of a novel loss function over adjacency matrices that quantify how "DAG-like" [1] a graph is, with the level set corresponding to and only to DAGs. Switching to continuous optimization using the NOTEARS loss is demonstrated to outperform more traditional discrete algorithms, despite largely relying on domain-agnostic optimization machinery.

1 What is Structure Learning

We'll begin with motivating **structural causal models** (SCM) for modelling complex, multidimensional systems, then introduce the problem of structure learning – the problem of learning the network topology underlying SCMs.

1.1 Structural Causal Models

Say you're trying to make sense of a complex, multi-dimensional system from which you have a bunch of observations, but no domain knowledge to make sense of it. Examples might include measurements of protein expressions in human cells (Sachs et al., 2005), or health stats of individuals who regularly exercise and of who don't. What would be your first step?

A reasonable first step would be to consult a domain expert about the data you have and learn which variable(s) are expected by theory to have a causal influence over the others. For example, a physician might tell you that frequent exercise strengthens the heart muscle, and that a strong heart muscle improves blood flow, thereby causing a reduction in heart rate. They might also tell you that exercise is linked with lower body mass index, which is also linked with lower heart rate as well as lower blood pressure. These causal relationships can be succinctly displayed with the help of a directed acyclic graph as done in Figure 1.

This graph simply summarizes what we know about which variables directly "cause" which other variables (or which variables are direct functions of the others):

$$E = U_E \tag{1}$$
$$S = f_S(E) + U_S \tag{2}$$
$$F = f_F(E) + U_F \tag{3}$$
$$R = f_R(S, F) + U_R \tag{4}$$
$$P = f_P(F) + U_P \tag{5}$$

[1] DAG stands for "directed acyclic graph"

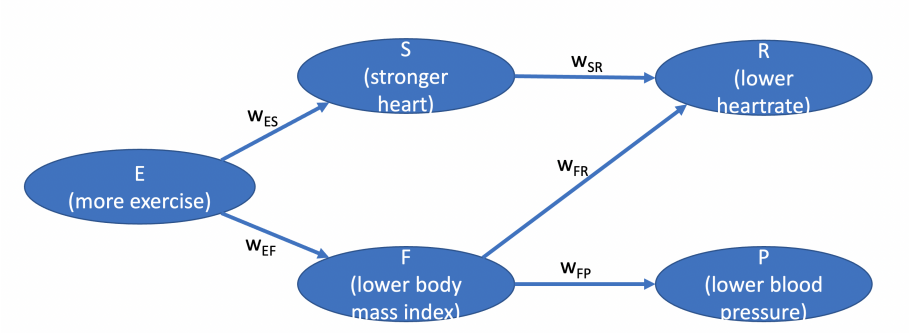


Figure 1: Example of a Structural Causal Model: A Structural Causal model defines with variables are direct (noisy) functions of which other variables. In the figure above, S (stronger heart) is a direct function of E (exercise), hence there's a link between them. On the other hand, R (lower heart rate) is only influenced by E(exercise) through the mediation of S (stronger heart); hence there's no link between the two.

In Figure 1, E, S, F, R and P are univariate random variables that represent exercise, heart strength, body mass index, heart rate and blood pressure, U_E, U_S, U_F, U_R and U_P are zero-centered (but not necessarily Gaussian) exogenous random variables and $f_S, f_F, f_R, f_P : \mathbb{R} \mapsto \mathbb{R}$ are functions that govern how causes impact their effects. The above set of variables and equations define a **Structural Causal Model** (SCM), also known as a **Structural Equation Model** – a central abstraction in causal inference.

Let's say the domain expert who gave us the above SCM also told us that the variables are linked to each other *linearly* – a very common assumption. This yields a linear SCM:

$$E = U_E \tag{6}$$
$$S = w_{ES}E + U_S \tag{7}$$
$$F = w_{EF}E + U_F \tag{8}$$
$$R = w_{SR}S + w_{FR}F + U_R \tag{9}$$
$$P = w_{FP}F + U_P \tag{10}$$

Setting $\boldsymbol{V} = [E, S, F, R, P]$ to be the concatenation of our random variables and $\boldsymbol{U} = [U_E, U_S, U_F, U_R, U_P]$ the concatenation of the exogenous noise variables, we can succinctly express the SCM using matrix-vector notation:

$$\boldsymbol{V} = \boldsymbol{W}\boldsymbol{V} + \boldsymbol{U} \tag{11}$$

The matrix $\boldsymbol{W}$ can be seen as the weighed adjacency matrix of the SCM: the nonzero entries $\boldsymbol{W}_{ij}$ are equal to the weights connecting two nodes i and j connected with an edge. Note that $\boldsymbol{W}$ is often quite sparse: $\boldsymbol{W}_{ij} = 0$ if and only if there's no edge between two variables.

What can we do with a given SCM topology? If we also know the weights, we can answer *probabilistic and causal inference* type questions: given someones body-mass index and exercise habits, can we predict their expected blood pressure? If a person is given a medicine that strengthens heart muscle without directly affecting anything else, how does this affect ones heart rate? We can also attempt to *learn* the coefficients in the linear SCM given a bunch of observations of the random variables: since the functional relationships between the variables is very clear, it's easy to set up a maximum likelihood objective and minimize it by optimizing over the SCM weights.

1.2 Structure Learning

Above, the topology of the graph was given to us by a domain expert. Structure learning refer to the problem of learning the topology of SCM from data - a very difficult, yet very important learning problem.

Let's formally state the structure learning problem following Zheng et al. (2018) building on the formalism above. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a data matrix consisting of n i.i.d. observations of the multivariable random variable $\boldsymbol{V}$. Let $\mathbb{D}$ stand for the space of all DAGs on d nodes. The purpose of structure learning is to learn a weighted adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ that not only models the dataset $\boldsymbol{X}$ via the SCM equation $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{V} + \boldsymbol{U}$, but also has the constraint that the edges defined by its nonzero entries describe a graph $G(V, E)$ that is a DAG (i.e. $G \in \mathbb{D}$)).

There are numerous ways of searching over the space of DAGs to find the most fitting one that describes the data. Until early 2010s, most approaches relied on discrete/combinatorial search algorithms (e.g Heckerman (1994)). Zheng et al. (2018) take a different approach which uses a fully continuous optimization based algorithm to solve search over DAGs. This lecture is about the key technical contribution of this work, which is a novel loss function over adjacency matrices that quantify how "DAG-like" a graph is, with the level set corresponding to and only to DAGs.

The optimization problem: We need three loss functions to set up our continuous optimization problem: (1) a loss function that quantifies how well the current $\boldsymbol{W}$ estimate fits the data, (2) a regularization term to make sure the solution is well-behaved (sparse, for example), and (3) a constraint that guarantees that the $\boldsymbol{W}$ found at the end of optimization represents a DAG.

The choice of (1) and (2) are not central to this lecture, so we'll directly state what Zheng et al. (2018) use:

$$F(W) = l(W, X) + r(W) = \frac{1}{2n}||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{W}||_F^2 + \lambda||\boldsymbol{W}||_1 \tag{12}$$

The choice of l, besides making intuitive sense [2], has solid theoretical backing: the minimizer of this loss "provably recovers a true DAG with high probability on finite samples and high-dimensions" (Zheng et al., 2018). The L1 loss on the adjacency matrix $\boldsymbol{W}$ serves to enforce sparsity. It's not hard to show that the minimizer of L1 regularized least squares is sparse. The combination of these loss functions is analogous to LASSO regression (Tibshirani, 1996).

The choice of (3) is the main contribution of (Zheng et al., 2018), and deserves a whole section of its own.

2 Quantifying "DAGness"

We're in search of a function $h : \mathbb{R}^{d \times d} \mapsto \mathbb{R}$ that have the following properties:

1. $h(\boldsymbol{W}) = 0$ iff $\boldsymbol{W}$ is acyclic.
2. High values of $h(\boldsymbol{W})$ implies that $G(\boldsymbol{W})$ has many cycles. Lower values imply not many cycles.
3. h is smooth.
4. h and it's derivatives are easy to compute.

While it's relatively easy to satisfy (1) and (2) alone, satisfying all 4 criteria is challenging.

[2] This loss has a maximum likelihood interpretation if one assumes that the exogenous noise terms are unit Gaussian.

Candidate 1: Pick $h(\boldsymbol{W})$ as the minimum L2 distance to an element of $\mathbb{D}$:

$$h_1(\boldsymbol{W}) = ||\boldsymbol{W} - \text{proj}_{\mathbb{D}} \boldsymbol{W}||_F \tag{13}$$

This loss function satisfies (1) and (2), but doesn't satisfy (3) and (4): it's derivatives are discontinuous and it's very difficult to differentiate through. (proof omitted)

Before considering $h(\boldsymbol{W})$ candidates, let's take a step back and consider what acyclicity really entails. For a graph to be a DAG, there should be no k-length cycles in it, for all values of k. If we could count the number of k-length cycles in a graph, then we could possibly use it to design a loss function to penalize k-length cycles. This insight leads to the following lemma:

Lemma 2.1. *Let $\boldsymbol{B} \in \mathbb{D}$ be the binary adjacency matrix of a directed graph G(V, E) of d nodes. The number of paths starting at node s and ending at node t, $p_k^{(st)}(\boldsymbol{B})$ is given by:*

$$p_k^{(st)}(\boldsymbol{B}) = \chi_t^T \boldsymbol{B}^k \chi_s \tag{14}$$

where χ_n is the indicator vector at node n.

Proof. We'll use proof by induction. The base case $k = 1$ works, because $\chi_t^T \boldsymbol{B} \chi_s = 1$ if s and t are neighbors (by the definition of an adjacency matrix) and 0 otherwise. This is consistent with the definition $p_1^{(st)}$. Now assume that the statement holds for $k-1$. This means that $(\boldsymbol{B}^{k-1}\chi_s)_i = p_k^{(si)}(\boldsymbol{B})$. That is, ith coordinate of $\boldsymbol{B}^{k-1}\chi_s$ contains the number of paths from s to i. The total number of length k paths that start from s and leads to t is equal to the sum of the number of all $k-1$ length paths that start at s and lead to a neighbor of t. Combining this with the definition of an adjacency matrix, we conclude the proof.

$$p_k^{(st)} = \sum_{u:(u,t)\in E} p_{k-1}^{(si)} = \sum_{i=1}^{d} \boldsymbol{B}_{ti} p_{k-1}^{(si)} = \sum_{i=1}^{d} \boldsymbol{B}_{ti} (\boldsymbol{B}^{k-1}\chi_s)_i = \chi_t^T B B^{k-1} \chi_s = \chi_t^T B^k \chi_s \tag{15}$$

□

Now that we can count the number of paths with known start and end nodes, we can count the total number of k-length cycles in a graph.

Lemma 2.2. *Let $\boldsymbol{B} \in \mathbb{D}$ be the binary adjacency matrix of a directed graph G(V, E) of d nodes. The total number of k-length cycles in G, $c_k(\boldsymbol{B})$, is given by:*

$$c_k(\boldsymbol{B}) = \text{Tr}\, \boldsymbol{B}^k \tag{16}$$

Proof. The total number of length k paths starting and ending on node s is given by $p_k^{(ss)}(\boldsymbol{B}) = \chi_s^T \boldsymbol{B}^k \chi_s$ (by Lemma 2.1). Counting length k cycles starting on all nodes, we get:

$$c_k(\boldsymbol{B}) = \sum_{i=1}^{d} p_k^{(ii)} = \sum_{i=1}^{d} (\boldsymbol{B}^k)_{ii} = \text{Tr}\, \boldsymbol{B}^k \tag{17}$$

□

We're now ready to state a new candidate loss function.

Candidate 2: Pick $h(\boldsymbol{W})$ as follows:

$$h_2(\boldsymbol{W}) = \text{Tr}(\boldsymbol{I} - \boldsymbol{W})^{-1} - d \tag{18}$$

While the inverse might look counter-intuitive, things become conceptually cleaner when one considers the (matrix version of the) identity $\frac{1}{1-x} = 1 + x + x^2 + x^3 \dots$ Now, let's justify this loss function more formally.

Lemma 2.3. *Let $\boldsymbol{B} \in \mathbb{D}$ be the binary adjacency matrix of a directed graph G(V, E) of d nodes. Further assume that $r(\boldsymbol{B}) < 1$ where $r(\boldsymbol{B})$ is the largest singular value of $\boldsymbol{B}$. Then $\boldsymbol{B}$ is a DAG if and only if*

$$\operatorname{Tr}(\boldsymbol{I} - \boldsymbol{B})^{-1} = d \tag{19}$$

Proof. $\boldsymbol{B}$ is a DAG if and only if it doesn't contain any cycles. By Lemma 2.2 this is equivalent to saying $\operatorname{Tr} \boldsymbol{B}^k = 0$ for any $k \geq 1$. By the Neumann sum identity, we can write:

$$\operatorname{Tr}(\boldsymbol{I} - \boldsymbol{B})^{-1} = \operatorname{Tr} \sum_{i=0}^{\infty} \boldsymbol{B}^k = \operatorname{Tr} \boldsymbol{I} + \sum_{i=1}^{\infty} \operatorname{Tr} \underbrace{\boldsymbol{B}^k}_{=0 \text{ for all k larger than 1}} = d \tag{20}$$

□

This loss function (or the version where infinity is replaced with d to the same effect) is quite promising, while having fatal flaws. First, we need to ensure $r(\boldsymbol{W}) < 1$ throughout training, which prevents learning to discover SCM formulas that don't satisfy these properties. Moreover, values of $\operatorname{Tr} \boldsymbol{B}^k$ can get extremely large if the graph is large enough, exceeding machine precision and preventing reliable gradients from being calculated.

Luckily, there exists a loss function that solves both of these issues.

Candidate 3: Pick h as follows:

$$h(\boldsymbol{W}) = \operatorname{Tr} \exp(\boldsymbol{W} \circ \boldsymbol{W}) - d \tag{21}$$

where $\circ$ stand for Hadamard product and exp stand for matrix exponential. Intuitively, this works because just like the inverse operation in Candidate 2, Candidate 3 also defines an infinite sum over the k-length cycles in the graph, but this time weighed by $\frac{1}{k!}$. Let's now see this more concretely.

Lemma 2.4. *Let $\boldsymbol{B} \in \mathbb{D}$ be the binary adjacency matrix of a directed graph G(V, E) of d nodes. $\boldsymbol{B}$ is a DAG iff*

$$\operatorname{Tr} \exp \boldsymbol{B} = d \tag{22}$$

Proof. Following the definition of the matrix exponential, we get:

$$\operatorname{Tr} \exp \boldsymbol{B} = \operatorname{Tr} \sum_{i=0}^{\infty} \frac{1}{k!} \boldsymbol{B}^k = \operatorname{Tr} \boldsymbol{I} + \sum_{i=1}^{\infty} \frac{1}{k!} \boldsymbol{B}^k = d + \sum_{i=1}^{\infty} \frac{1}{k!} \boldsymbol{B}^k \tag{23}$$

By Lemma 2.2, we have that the right hand of Equation 2.2 is 0 (i.e. $\operatorname{Tr} \boldsymbol{B}^k = 0$ for all $k \geq 1$) iff $\boldsymbol{B}$ is a DAG, completing the proof. □

As stated above, moving from the inverse operation to matrix exponential merely adds a $\frac{1}{k!}$ scaling to each term in the infinite sum. this serves two crucial purposes: 1) Now, we don't have any constraint on the spectral radius! This loss function works on any matrix. 2) Larger length step counts are normalized by a factor that grows exponentially with k. Moreover, it's now smooth and differentiable!

Lemma 2.5. *The gradient of the (candidate 3) loss function $h(\boldsymbol{B})$ is given by:*

$$\nabla h(\boldsymbol{W}) = 2(\exp \boldsymbol{W} \circ \boldsymbol{W}) \circ \boldsymbol{W} \tag{24}$$

The proof for this is trivial and hence omitted.

Candidate 3 satisfies all of the desiderata, hence can be used to fully define the NOTEARS optimization problem:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{d\times d}} \frac{1}{2n}||\boldsymbol{X}-\boldsymbol{X}\boldsymbol{W}||_F^2+\lambda||\boldsymbol{W}||_1 \tag{25}$$

$$\text{subject to} \quad h(\boldsymbol{W})=\operatorname{Tr}\exp(\boldsymbol{W}\circ\boldsymbol{W})-d=0 \tag{26}$$

While well defined, this objective is unfortunately non-convex, since the loss function h is nonconvex. (Nonconvexity is inevitable, as the space of DAGs is non-convex) We'll now discuss how the augmented Lagrangian machinery can be used to solve this constrained minimization problem to great empirical success.

3 Optimizing the NOTEARS Objective

As mentioned in the previous section, the optimization problem is non-convex since the constraint $h(\boldsymbol{W})=0$ defines a non-convex set. This implies that having the gradient of the Lagrangian equal to zero at some point $\boldsymbol{W}$ (i.e., a *stationary point*) is not a sufficient condition for optimality. Nonetheless, experiments show that said stationary points are close enough to the true optimum for the applications considered.

The authors apply the augmented Lagrangian method to solve the optimization problem, which adds a quadratic penalty $\frac{\rho}{2}|h(\boldsymbol{W})|^2$ to the objective function resulting in

$$\min_{\boldsymbol{W}\in\mathbb{R}^{d\times d}} \frac{1}{2n}||\boldsymbol{X}-\boldsymbol{X}\boldsymbol{W}||_F^2+\lambda||\boldsymbol{W}||_1+\alpha h(\boldsymbol{W})+\frac{\rho}{2}|h(\boldsymbol{W})|^2 \tag{27}$$

$$\text{subject to} \quad h(\boldsymbol{W})=\operatorname{Tr}\exp(\boldsymbol{W}\circ\boldsymbol{W})-d=0. \tag{28}$$

Note that $h(\boldsymbol{W})=0$ whenever $\boldsymbol{W}$ is a solution. Hence, adding the quadratic penalty will not change the solution set with respect to the original optimization problem. Intuitively, as ρ increases, any convex solver will increasingly favor solution where $|h(\boldsymbol{W})|$ is closer to zero. A full treatment of this method is beyond the scope of the write-up.

Converting the constrained optimization problem above into a non-constrained one yields the following objective

$$\max_{\alpha\in\mathbb{R}}\min_{\boldsymbol{W}\in\mathbb{R}^{d\times d}} F(\boldsymbol{W})+\frac{\rho}{2}|h(\boldsymbol{W})|^2+\alpha h(\boldsymbol{W}), \tag{29}$$

where $F(\boldsymbol{W})=\frac{1}{2n}||\boldsymbol{X}-\boldsymbol{X}\boldsymbol{W}||_F^2+\lambda||\boldsymbol{W}||_1$. As previously mentioned, it will suffice to find a local minimizer $\boldsymbol{W}_\alpha^\star$ for the inner loop at some value of α. There are a multitude of different black-box solvers that can efficiently find globally and locally optimal solutions for convex and non-convex problems, respectively. The development of such programs are completely orthogonal to the specific domain of the optimization problem and is out of the scope for this write-up and hence shall not be discussed. The outer loop can be optimized via gradient descent methods by noting that

$$\nabla_\alpha\left(F(\boldsymbol{W}_\alpha^\star)+\frac{\rho}{2}|h(\boldsymbol{W}_\alpha^\star)|^2+\alpha h(\boldsymbol{W}_\alpha^\star)\right)=h(\boldsymbol{W}_\alpha^\star). \tag{30}$$

Finally, because the chosen convex solvers will not necessarily output an estimate $\tilde{\boldsymbol{W}}$ with some entries equal to 0 (i.e., indicating an absence of an edge in the DAG), the method requires specifying a

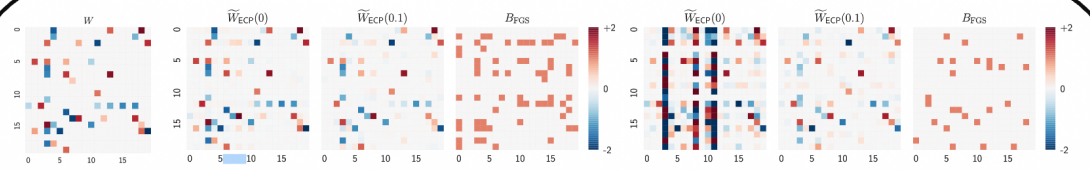
threshold value $\omega \geq 0$ where all entries in $\tilde{\boldsymbol{W}}$ smaller than ω are set to zero. Note that the thresholding step has non-trivial consequences: a hard-thresholding step with an appropriately chosen value reduces the number of false discoveries (Zhou, 2009).

4 Results and Discussion

The authors provide comparisons with existing methods on both toy examples (i.e., synthetic data) and real-world datasets.

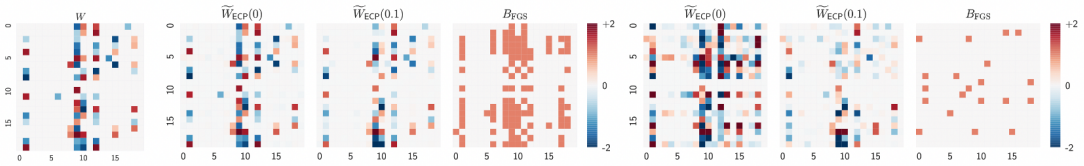
For each experiment a graph is sampled from either the Erdős–Rényi (ER) or scale-free (SF) families of random graphs along with a ground truth weight matrix $\boldsymbol{W}$. Then, $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{V} + \boldsymbol{U}$ is sampled using either a Normal, Exponential or Gumbel distribution for $\boldsymbol{U}$ to generate datasets $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for multiple values of n and d. For each experiment, the following 3 analyses are presented.

Parameter estimation: In Figures 1. and 2. [3] a visual comparison between the estimated and ground truth weight matrices, $\tilde{\boldsymbol{W}}$ and $\boldsymbol{W}^\star$, is provided for different threshold values ω. The results indicate that, at least for these simple experiments and through visual inspection, the method can recover the underlying ground truth.



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

Figure 1: Visual comparison of the learned weighted adjacency matrix on a 20-node graph with $n = 1000$ (large samples) and $n = 20$ (insufficient samples): $\widetilde{W}_{\text{ECP}}(\lambda)$ is the proposed NOTEARS algorithm with ℓ_1-regularization λ, and B_{FGS} is the binary estimate of the baseline (Ramsey et al., 2016). The proposed algorithms perform well on large samples, and remains accurate on small n with ℓ_1 regularization.

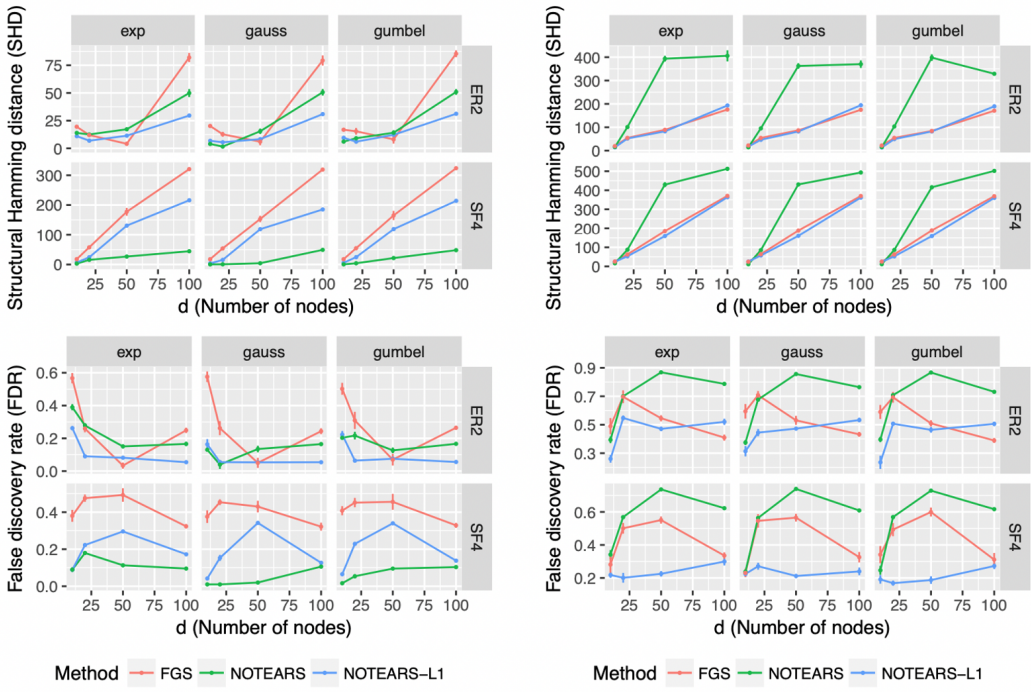


(a) true graph (b) estimate with $n = 1000$ (c) estimate with $n = 20$

Figure 2: Parameter estimates of $\widetilde{W}_{\text{ECP}}$ on a scale-free graph. Without the additional thresholding step in Algorithm 1, NOTEARS still produces consistent estimates of the true graph. The proposed method estimates the weights very well with large samples even without regularization, and remains accurate on insufficient samples when ℓ_1-regularization is introduced. See also Figure 1.

Structure Learning: A common way to compare 2 graphs is through the *Hamming Distance*, which measures the minimal number of edits (e.g., addition and removal of edges) required to transform one graph to another. The Hamming Distance is compared for varying values of n, d, distributions for $\boldsymbol{U}$, and random graph families. NOTEARS consistently outperforms other methods for large values of n and d, but under-performs for small values of n (i.e., when there is little data) – see Figure 3.

[3] These figures are screenshots from the original paper. Hence, the captions belong to the authors.



(a) SHD with $n = 1000$

(b) SHD with $n = 20$

Figure 3: Structure recovery in terms of SHD and FDR to the true graph (lower is better). Rows: random graph types, {ER,SF}-k = {Erdös-Rényi, scale-free} graphs with kd expected edges. Columns: noise types of SEM. Error bars represent standard errors over 10 simulations.

Comparison to exact global minimizer: For small DAGs it is possible to find the global optimum of the NOTEARS objective in feasible time using out-of-the-box solvers. In this setting the dimensionality is fixed to $d = 10$ and n is varied from 20 to 1000. The error of NOTEARS estimate $\tilde{\boldsymbol{W}}$ with respect to the global objective $\boldsymbol{W}^\star$ is measured as $||\tilde{\boldsymbol{W}} - \boldsymbol{W}^\star||$, which is shown to be small (compared to the norm of $\boldsymbol{W}^\star$). This indicates that the non-convexity of the NOTEARS objective does not significantly hinder the employment of convex solvers, at least for low dimensionality – Table 1.

Table 1: Comparison of NOTEARS vs. globally optimal solution. $\Delta(W_{\mathsf{G}}, \widehat{W}) = F(W_{\mathsf{G}}) - F(\widehat{W})$.

n	λ	Graph	$F(W)$	$F(W_{\mathsf{G}})$	$F(\widehat{W})$	$F(\widetilde{W}_{\mathsf{ECP}})$	$\Delta(W_{\mathsf{G}}, \widehat{W})$	$\|\widehat{W} - W_{\mathsf{G}}\|$	$\|W - W_{\mathsf{G}}\|$
20	0	ER2	5.11	3.85	5.36	3.88	-1.52	0.07	3.38
20	0.5	ER2	16.04	12.81	13.49	12.90	-0.68	0.12	3.15
1000	0	ER2	4.99	4.97	5.02	4.95	-0.05	0.02	0.40
1000	0.5	ER2	15.93	13.32	14.03	13.46	-0.71	0.12	2.95
20	0	SF4	4.99	3.77	4.70	3.85	-0.93	0.08	3.31
20	0.5	SF4	23.33	16.19	17.31	16.69	-1.12	0.15	5.08
1000	0	SF4	4.96	4.94	5.05	4.99	-0.11	0.04	0.29
1000	0.5	SF4	23.29	17.56	19.70	18.43	-2.13	0.13	4.34

References

Heckerman, D. (1994). The combination of knowledge and statistical data. In *Proceedings of the Tenth Conference on Uncertainty if Artificial Intelligence, Los Altos, 1994*. Morgan Kaufman.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. In *NeurIPS*.

Zhou, S. (2009). Thresholding procedures for high dimensional variable selection and statistical estimation. *Advances in Neural Information Processing Systems*, 22.