GRADIENT-BASED NEURAL DAG LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel score-based approach to learning a directed acyclic graph (DAG) from observational data. We adapt a recently proposed continuous constrained optimization formulation to allow for nonlinear relationships between variables using neural networks. This extension allows to model complex interactions while avoiding the combinatorial nature of the problem. In addition to comparing our method to existing continuous optimization methods, we provide missing empirical comparisons to nonlinear greedy search methods. On both synthetic and real-world data sets, this new method outperforms current continuous methods on most tasks, while being competitive with existing greedy search methods on important metrics for causal inference.

1 INTRODUCTION

Structure learning and causal inference have many important applications in different areas of science such as genetics (Koller & Friedman, 2009; Peters et al., 2017), biology (Sachs et al., 2005) and economics (Pearl, 2009). *Bayesian networks* (BN), which encode conditional independencies using *directed acyclic graphs* (DAG), are powerful models which are both interpretable and computationally tractable. *Causal graphical models* (CGM) (Peters et al., 2017) are BNs which support *interventional* queries like: *What will happen if someone external to the system intervenes on variable X?* Recent work suggests that causality could partially solve challenges faced by current machine learning systems such as robustness to out-of-distribution samples, adaptability and explainability (Pearl, 2019; Magliacane et al., 2018). However, structure and causal learning are daunting tasks due to both the combinatorial nature of the space of structures (the number of DAGs grows *super exponentially* with the number of nodes) and the question of *structure identifiability* (see Section 2.2). Nevertheless, these graphical models known qualities and promises of improvement for machine intelligence renders the quest for structure/causal learning appealing.

In this work, we propose a score-based method (Koller & Friedman, 2009) for structure learning named GraN-DAG which makes use of a recent reformulation of the original *combinatorial problem* of finding an optimal DAG into a *continuous constrained optimization problem*. In the original method named NOTEARS (Zheng et al., 2018), the directed graph is encoded as a *weighted adjacency matrix* which represents coefficients in a linear *structural equation model* (SEM) (Pearl, 2009) (see Section 2.3) and enforces acyclicity using a constraint which is both efficiently computable and easily differentiable, thus allowing the use of numerical solvers. This continuous approach improved upon popular methods while avoiding the design of greedy algorithms based on heuristics.

Our first contribution is to extend the framework of Zheng et al. (2018) to deal with nonlinear relationships between variables using neural networks (NN) (Goodfellow et al., 2016). GraN-DAG is general enough to deal with a large variety of parametric families of conditional probability distributions. To adapt the acyclicity constraint to our nonlinear model, we use an argument similar to what is used in Zheng et al. (2018) and apply it first at the level of *neural network paths* and then at the level of *graph paths*. On both synthetic and real-world tasks, we show GraN-DAG outperforms other approaches which leverage the continuous paradigm, including DAG-GNN (Yu et al., 2019), a recent nonlinear extension of Zheng et al. (2018) which uses an evidence lower bound as score.

Our second contribution is to provide a missing empirical comparison to existing methods that support nonlinear relationships but tackle the optimization problem in its discrete form using greedy search procedures, namely CAM (Bühlmann et al., 2014) and GSF (Huang et al., 2018). We show

We provide an implementation of GraN-DAG here.

that GraN-DAG is competitive on the wide range of tasks we considered, while using similar pre and post-processing steps as CAM.

2 BACKGROUND

Before presenting GraN-DAG, we review concepts relevant to structure and causal learning.

2.1 CAUSAL GRAPHICAL MODELS

We suppose the natural phenomenon of interest can be described by a random vector $X \in \mathbb{R}^d$ entailed by an underlying CGM $(P_X, \mathcal{G})$ where P_X is a probability distribution over X and $\mathcal{G} = (V, E)$ is a DAG (Peters et al., 2017). Each node $j \in V$ corresponds to exactly one variable in the system. Let $\pi_j^{\mathcal{G}}$ denote the set of parents of node j in $\mathcal{G}$ and let $X_{\pi_j^{\mathcal{G}}}$ denote the random vector containing the variables corresponding to the parents of j in $\mathcal{G}$. Throughout the paper, we assume there are no hidden variables. In a CGM, the distribution P_X is said to be *Markov* to $\mathcal{G}$, i.e. we can write the probability density function (pdf) of P_X as $p(x) = \prod_{j=1}^{d} p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ where $p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ is the conditional pdf of variable X_j given $X_{\pi_j^{\mathcal{G}}}$. A CGM can be thought of as a BN in which directed edges are given a causal meaning, allowing it to answer queries regarding *interventional distributions* (Koller & Friedman, 2009).

2.2 STRUCTURE IDENTIFIABILITY

In general, it is impossible to recover $\mathcal{G}$ given samples from P_X. It is, however, customary to rely on a set of assumptions to render the structure fully or partially *identifiable*.

Definition 1 *Given a set of assumptions A on a CGM $\mathcal{M} = (P_X, \mathcal{G})$, its graph $\mathcal{G}$ is said to be identifiable from P_X if there exists no other CGM $\tilde{\mathcal{M}} = (\tilde{P}_X, \tilde{\mathcal{G}})$ satisfying all assumptions in A such that $\tilde{\mathcal{G}} \neq \mathcal{G}$ and $\tilde{P}_X = P_X$.*

There are many examples of graph identifiability results for continuous variables (Peters et al., 2014; Peters & Bühlman, 2014; Shimizu et al., 2006; Zhang & Hyvärinen, 2009) as well as for discrete variables (Peters et al., 2011). These results are obtained by assuming that the conditional pdf $p_j \; \forall j$ belongs to a specific parametric family $\mathcal{P}$. For example, if one assumes that

$$X_j | X_{\pi_j^{\mathcal{G}}} \sim \mathcal{N}(f_j(X_{\pi_j^{\mathcal{G}}}), \sigma_j^2) \;\; \forall j \tag{1}$$

where f_j is a nonlinear function satisfying some mild regularity conditions, then $\mathcal{G}$ is identifiable from P_X (see Peters et al. (2014) for the complete theorem and its proof). We will make use of this result in Section 4.

One can consider weaker assumptions such as *faithfulness* (Peters et al., 2017). This assumption allows one to identify, not $\mathcal{G}$ itself, but the *Markov equivalence class* to which it belongs (Spirtes et al., 2000). A Markov equivalence class is a set of DAGs which encodes exactly the same set of conditional independence statements and can be characterized by a graphical object named a *completed partially directed acyclic graph* (CPDAG) (Koller & Friedman, 2009; Peters et al., 2017). Some algorithms we use as baselines in Section 4 output only a CPDAG.

2.3 NOTEARS: CONTINUOUS OPTIMIZATION FOR STRUCTURE LEARNING

Structure learning is the problem of learning $\mathcal{G}$ using a data set of n samples $\{x^{(1)}, ..., x^{(n)}\}$ from P_X. Score-based approaches cast this problem as an optimization problem, i.e. $\hat{\mathcal{G}} = \arg\max_{\mathcal{G} \in \text{DAG}} \mathcal{S}(\mathcal{G})$ where $\mathcal{S}(\mathcal{G})$ is a regularized maximum likelihood under graph $\mathcal{G}$. Since the number of DAGs is super exponential in the number of nodes, most methods rely on various heuristic greedy search procedures to approximately solve the problem (see Section 5 for a review). We now present the work of Zheng et al. (2018) which proposes to cast this combinatorial optimization problem into a continuous constrained one.

To do so, the authors propose to encode the graph $\mathcal{G}$ on d nodes as a weighted adjacency matrix $U = [u_1|\dots|u_d] \in \mathbb{R}^{d\times d}$ which represents (possibly negative) coefficients in a linear SEM of the form $X_j := u_j^\top X + N_i \;\; \forall j$ where N_j is a noise variable. Let $\mathcal{G}_U$ be the directed graph associated with the SEM and let A_U be the (binary) adjacency matrix associated with $\mathcal{G}_U$. One can see that the following equivalence holds:

$$(A_U)_{ij} = 0 \iff U_{ij} = 0 \tag{2}$$

To make sure $\mathcal{G}_U$ is acyclic, the authors propose the following constraint on U:

$$\operatorname{Tr} e^{U\odot U} - d = 0 \tag{3}$$

where $e^M \triangleq \sum_{k=0}^{\infty} \frac{M^k}{k!}$ is the *matrix exponential* and $\odot$ is the Hadamard product.

To see why this constraint characterizes acyclicity, first note that $({A_U}^k)_{jj}$ is the number of cycles of length k passing through node j in graph $\mathcal{G}_U$. Clearly, for $\mathcal{G}_U$ to be acyclic, we must have $\operatorname{Tr} {A_U}^k = 0$ for $k = 1, 2, ..., \infty$. By equivalence (2), this is true when $\operatorname{Tr}(U \odot U)^k = 0$ for $k = 1, 2, ..., \infty$. From there, one can simply apply the definition of the matrix exponential to see why constraint 3 characterizes acyclicity (see Zheng et al. (2018) for the full development).

The authors propose to use a regularized negative least square score (maximum likelihood for a Gaussian noise model). The resulting continuous constrained problem is

$$\max_U \mathcal{S}(U, \mathbf{X}) \triangleq -\frac{1}{2n}\|\mathbf{X} - \mathbf{X}U\|_F^2 - \lambda\|U\|_1 \quad \text{s.t.} \quad \operatorname{Tr} e^{U\odot U} - d = 0 \tag{4}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the design matrix containing all n samples. The nature of the problem has been drastically changed: we went from a combinatorial to a continuous problem. The difficulties of combinatorial optimization have been replaced by those of non-convex optimization, since the feasible set is non-convex. Nevertheless, a standard numerical solver for constrained optimization such has an *augmented Lagrangian method* (Bertsekas, 1999) can be applied to get an approximate solution, hence there is no need to design a greedy search procedure. Moreover, this approach is more global than greedy methods in the sense that the whole matrix U is updated at each iteration. Continuous approaches to combinatorial optimization have sometimes demonstrated improved performance over discrete approaches in the literature (see for example Alayrac et al. (2018, §5.2) where they solve the multiple sequence alignment problem with a continuous optimization method).

3 GRAN-DAG: GRADIENT-BASED NEURAL DAG LEARNING

We propose a new nonlinear extension to the framework presented in Section 2.3. For each variable X_j, we learn a fully connected neural network with L hidden layers parametrized by $\phi_{(j)} \triangleq \{W_{(j)}^{(1)}, \dots, W_{(j)}^{(L+1)}\}$ where $W_{(j)}^{(\ell)}$ is the ℓth weight matrix of the jth NN (biases are omitted for clarity). Each NN takes as input $X_{-j} \in \mathbb{R}^d$, i.e. the vector X with the jth component masked to zero, and outputs $\theta_{(j)} \in \mathbb{R}^m$, the m-dimensional parameter vector of the desired distribution family for variable X_j.[2] The fully connected NNs have the following form

$$\theta_{(j)} \triangleq W_{(j)}^{(L+1)} g(\dots g(W_{(j)}^{(2)} g(W_{(j)}^{(1)} X_{-j})) \dots) \;\; \forall j \tag{5}$$

where g is a nonlinearity applied element-wise. Note that the evaluation of all NNs can be parallelized on GPU. Distribution families need not be the same for each variable. Let $\phi \triangleq \{\phi_{(1)}, \dots, \phi_{(d)}\}$ represents all parameters of all d NNs. Without any constraint on its parameter $\phi_{(j)}$, neural network j models the conditional pdf $p_j(x_j|x_{-j}; \phi_{(j)})$. Note that the product $\prod_{j=1}^d p_j(x_j|x_{-j}; \phi_{(j)})$ does not integrate to one (i.e. it is not a joint pdf), since it does not decompose according to a DAG. We now show how one can constrain ϕ to make sure the product of all conditionals outputted by the NNs is a joint pdf. The idea is to define a new weighted adjacency matrix A_ϕ similar to the one encountered in Section 2.3, which can be directly used inside the constraint of Equation 3 to enforce acyclicity.

[2] Not all parameter vectors need to have the same dimensionality, but to simplify the notation, we suppose $m_j = m \;\; \forall j$

3.1 NEURAL NETWORK CONNECTIVITY

Before defining the weighted adjacency matrix A_ϕ, we need to focus on how one can make some NN outputs unaffected by some inputs. Since we will discuss properties of a single NN, we drop the NN subscript (j) to improve readability.

We will use the term *neural network path* to refer to a computation path in a NN. For example, in a NN with two hidden layers, the sequence of weights $(W^{(1)}_{h_1 i}, W^{(2)}_{h_2 h_1}, W^{(3)}_{k h_2})$ is a NN path from input i to output k. We say that a NN path is *inactive* if at least one weight along the path is zero. We can loosely interpret the *path product* $|W^{(1)}_{h_1 i}||W^{(2)}_{h_2 h_1}||W^{(3)}_{k h_2}| \geq 0$ as the strength of the NN path, where a path product is equal to zero if and only if the path is inactive. Note that if all NN paths from input i to output k are inactive (i.e. the sum of their path products is zero), then output k does not depend on input i anymore since the information in input i will never reach output k. The sum of all path products from input i to output k for all input i and output k can be easily computed by taking the following matrix product.

$$C \triangleq |W^{(L+1)}| \dots |W^{(2)}||W^{(1)}| \in \mathbb{R}^{m \times d}_{\geq 0} \tag{6}$$

where $|W|$ is the element-wise absolute value of W. Let us name C the *neural network connectivity matrix*. It can be verified that C_{ki} is the sum of all NN path products from input i to output k. This means it is sufficient to have $C_{ki} = 0$ to render output k independent of input i.

Remember that each NN in our model outputs a parameter vector θ for a conditional distribution and that we want the product of all conditionals to be a valid joint pdf, i.e. we want its corresponding directed graph to be acyclic. With this in mind, we see that it could be useful to make a certain parameter θ not dependent on certain inputs of the NN. To have θ independent of variable X_i, it is sufficient to have $\sum_{k=1}^{m} C_{ki} = 0$.

3.2 A WEIGHTED ADJACENCY MATRIX

We now define a weighted adjacency matrix A_ϕ that can be used in constraint of Equation 3.

$$(A_\phi)_{ij} \triangleq \begin{cases} \sum_{k=1}^{m} \left(C_{(j)}\right)_{ki}, & \text{if } j \neq i \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $C_{(j)}$ denotes the connectivity matrix of the NN associated with variable X_j.

As the notation suggests, $A_\phi \in \mathbb{R}^{d \times d}_{\geq 0}$ depends on all weights of all NNs. Moreover, it can effectively be interpreted as a weighted adjacency matrix similarly to what we presented in Section 2.3, since we have that

$$(A_\phi)_{ij} = 0 \implies \theta_{(j)} \text{ does not depend on variable } X_i \tag{8}$$

We note $\mathcal{G}_\phi$ to be the directed graph entailed by parameter ϕ. We can now write our adapted acyclicity constraint:

$$h(\phi) \triangleq \operatorname{Tr} e^{A_\phi} - d = 0 \tag{9}$$

Note that we can compute the gradient of $h(\phi)$ w.r.t. ϕ (except at points of non-differentiability arising from the absolute value function, similar to standard neural networks with ReLU activations (Glorot et al., 2011); these points did not appear problematic in our experiments using SGD).

3.3 A DIFFERENTIABLE SCORE AND ITS OPTIMIZATION

We propose solving the maximum likelihood optimization problem

$$\max_{\phi} \mathbb{E}_{X \sim P_X} \sum_{j=1}^{d} \log p_j(X_j | X_{\pi^\phi_j}; \phi_{(j)}) \quad \text{s.t.} \quad \operatorname{Tr} e^{A_\phi} - d = 0 \tag{10}$$

where π^ϕ_j denotes the set of parents of node j in graph $\mathcal{G}_\phi$. Note that $\sum_{j=1}^{d} \log p_j(X_j | X_{\pi^\phi_j}; \phi_{(j)})$ is a valid log-likelihood function when constraint (9) is satisfied.

As suggested in Zheng et al. (2018), we apply an augmented Lagrangian approach to get an approximate solution to program (10). Augmented Lagrangian methods consist of optimizing a sequence

of subproblems for which the exact solutions are known to converge to a stationary point of the constrained problem under some regularity conditions (Bertsekas, 1999). In our case, each subproblem is

$$\max_{\phi} \mathcal{L}(\phi, \lambda_t, \mu_t) \triangleq \mathbb{E}_{X \sim P_X} \sum_{j=1}^{d} \log p_j(X_j | X_{\pi_j^\phi}; \phi_{(j)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \tag{11}$$

where λ_t and μ_t are the Lagrangian and penalty coefficients of the tth subproblem, respectively. These coefficients are updated after each subproblem is solved. See Appendix A.1 for details regarding the optimization procedure.

3.4 Thresholding to ensure acyclicity

The solution outputted by the augmented Lagrangian will satisfy the constraint only up to numerical precision, thus several entries of A_ϕ might not be exactly zero and require thresholding. To do so, we mask the inputs of each NN j using a binary matrix $M_{(j)} \in \{0,1\}^{d \times d}$ initialized to have $(M_{(j)})_{ii} = 1 \ \ \forall i \neq j$ and zeros everywhere else. Having $(M_{(j)})_{ii} = 0$ means the input i of NN j has been thresholded. This mask is integrated in the product of Equation 6 by doing $C_{(j)} \triangleq |W_{(j)}^{(L+1)}| \dots |W_{(j)}^{(1)}| M_{(j)}$ without changing the interpretation of $C_{(j)}$. During optimization, if the entry $(A_\phi)_{ij}$ is smaller than the threshold $\epsilon = 10^{-4}$, the corresponding mask entry $(M_{(j)})_{ii}$ is set to zero, permanently. The masks $M_{(j)}$ are never updated via gradient descent. We also add an iterative thresholding step at the end to ensure the estimated graph $\mathcal{G}_\phi$ is acyclic (described in Appendix A.2).

3.5 Overfitting

In practice, we maximize an empirical estimate of the objective of problem (10). It is well known that this maximum likelihood score is prone to overfitting in the sense that adding edges can never reduce the maximal likelihood (Koller & Friedman, 2009). GraN-DAG gets around this issue in three ways. First, as we optimize a subproblem, we evaluate its objective on a held-out data set and declare convergence once it has stopped improving. This approach is known as *early stopping* (Prechelt, 1997). Second, once we have thresholded our graph estimate to be a DAG, we apply a final pruning step identical to what is done in CAM (Bühlmann et al., 2014) to remove spurious edges. This step performs a regression of each node against its parents and uses a significance test to decide which parents should be kept or not. Third, for graphs of 50 nodes or more, we apply a *preliminary neighbors selection* (PNS) before running the optimization procedure as was also recommended in Bühlmann et al. (2014). This procedure selects a set of potential parents for each variables. See Appendix A.3 for details on PNS and pruning. Many score-based approaches control overfitting by penalizing the number of edges in their score. For example, NOTEARS includes the L1 norm of its weighted adjacency matrix U in its objective. GraN-DAG regularizes using PNS and pruning for ease of comparision to CAM, the most competitive approach in our experiments. The importance of PNS and pruning and their ability to reduce overfitting is illustrated in an ablation study presented in Appendix A.3. The study shows that PNS and pruning are both very important for the performance of GraN-DAG in terms of SHD, but don't have a significant effect in terms of SID. In these experiments, we also present NOTEARS and DAG-GNN with PNS and pruning, without noting a significant improvement.

4 Experiments

In this section, we compare GraN-DAG to various baselines in the continuous paradigm, namely DAG-GNN (Yu et al., 2019) and NOTEARS (Zheng et al., 2018), and also in the combinatorial paradigm, namely CAM (Bühlmann et al., 2014), GSF (Huang et al., 2018), GES (Chickering, 2003) and PC (Spirtes et al., 2000). These methods are discussed in Section 5. We also report the performance of random graphs sampled using the *Erdős-Rényi* (ER) scheme described in Appendix A.4 (denoted by RANDOM). For each approach, we evaluate the estimated graph on two metrics: the *structural hamming distance* (SHD) and the *structural interventional distance* (SID) (Peters & Bühlmann, 2013). The former simply counts the number of missing, falsely detected or reversed edges. The latter is especially well suited for causal inference since it counts the number of couples (i, j) such that the interventional distribution $p(x_j | do(X_i = \bar{x}))$ would be miscalculated if we were

to use the estimated graph to form the parent adjustement set. Note that GSF, GES and PC output only a CPDAG, hence the need to report a lower and an upper bound on the SID. See Appendix A.6 for more details on SHD and SID. All experiments were ran with publicly available code from the authors website. See Appendix A.7 for the details of their hyperparameters.

4.1 SYNTHETIC DATA

We have generated different *data set types* which vary along three dimensions: number of nodes, level of edge sparsity and graph type. We consider two graph sampling schemes: *Erdős-Rényi* (ER) and *scale-free* (SF) (see Appendix A.4 for details). For each data set type, we sampled 10 data sets of 1000 examples as follows: First, a ground truth DAG $\mathcal{G}$ is randomly sampled following either the ER or the SF scheme. Then, the data is generated following $X_j|X_{\pi_j^{\mathcal{G}}} \sim \mathcal{N}(f_j(X_{\pi_j^{\mathcal{G}}}), \sigma_j^2) \ \forall j$ with the functions f_j independently sampled from a Gaussian process with unit bandwidth and with σ_j^2 sampled uniformly. As mentioned in Section 2.2, in this setup we know $\mathcal{G}$ to be identifiable from the distribution (Peters et al., 2014). This ensures that finding the correct DAG via maximum likelihood is not impossible.

In these experiments, each NN learned by GraN-DAG outputs a Gaussian mean $\hat{\mu}_{(j)}$, i.e. $\theta_{(j)} := \hat{\mu}_{(j)} \ \forall j$. The parameters $\hat{\sigma}^2_{(j)}$ are learned as well, but do not depend on the parent variables $X_{\pi_j^{\mathcal{G}}}$. Note that NOTEARS and CAM also make the correct Gaussian model assumption.

We considered graphs of 10, 20, 50 and 100 nodes. We only present results for 10 and 50 nodes in the main paper since the conclusions do not change with graphs of 20 or 100 nodes (see Appendix A.5 for the additional experiments). We consider graphs of d and $4d$ edges (respectively denoted by ER1 and ER4 in the case of ER graphs). We report the performance of the popular GES and PC in Appendix A.5 since they are almost never on par with the best methods presented in this section.

Table 1: Results for ER and SF graphs of 10 nodes

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**1.7±2.5**	**1.7±3.1**	**8.3±2.8**	**21.8±8.9**	**1.2±1.1**	**4.1±6.1**	**9.9±4.0**	**16.4±6.0**
DAG-GNN	11.4±3.1	37.6±14.4	35.1±1.5	81.9±4.7	9.9±1.1	29.7±15.8	20.8±1.9	48.4±15.6
NOTEARS	12.2±2.9	36.6±13.1	32.6±3.2	79.0±4.1	10.7±2.2	32.0±15.3	20.8±2.7	49.8±15.6
CAM	**1.1±1.1**	**1.1±2.4**	**12.2±2.7**	**30.9±13.2**	**1.4±1.6**	**5.4±6.1**	**9.8±4.3**	**19.3±7.5**
GSF	6.5±2.6	[6.2±10.8 17.7±12.3]	21.7±8.4	[37.2±19.2 62.7±14.9]	**1.8±1.7**	**[2.0±5.1 6.9±6.2]**	**8.5±4.2**	**[13.2±6.8 20.6±12.1]**
RANDOM	26.3±9.8	25.8±10.4	31.8±5.0	76.6±7.0	25.1±10.2	24.5±10.5	28.5±4.0	47.2±12.2

Table 2: Results for ER and SF graphs of 50 nodes

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**5.1±2.8**	**22.4±17.8**	**102.6±21.2**	**1060.1±109.4**	**25.5±6.2**	**90.0±18.9**	**111.3±12.3**	**271.2±65.4**
DAG-GNN	49.2±7.9	304.4±105.1	191.9±15.2	2146.2±64	49.8±1.3	182.8±42.9	144.9±13.3	540.8±151.1
NOTEARS	62.8±9.2	327.3±119.9	202.3±14.3	2149.1±76.3	57.7±3.5	195.7±54.9	153.7±11.8	558.4±153.5
CAM	**4.3±1.9**	**22.0±17.9**	**98.8±20.7**	**1197.2±125.9**	**24.1±6.2**	**85.7±31.9**	**111.2±13.3**	**320.7±152.6**
GSF	25.6±5.1	[21.1±23.1 79.2±33.5]	**81.8±18.8**	**[906.6±214.7 1030.2±172.6]**	**31.6±6.7**	[85.8±29.9 147.3±49.9]	**120.2±10.9**	[284.7±80.2 379.9±98.3]
RANDOM	535.7±401.2	272.3±125.5	708.4±234.4	1921.3±203.5	514.0±360.0	381.3±190.3	660.6±194.9	1198.9±304.6

We now examine Tables 1 and 2 (the errors bars represent the standard deviation across datasets per task). We can see that, across all settings, GraN-DAG and CAM are the best performing methods, both in terms of SHD and SID, while GSF is not too far behind. The poor performance of NOTEARS can be explained by its inability to model nonlinear functions. In terms of SHD, DAG-GNN performs rarely better than NOTEARS while in terms of SID, it performs similarly to RANDOM in almost all cases except in scale-free networks of 50 nodes or more. Its poor performance might be due to its incorrect modelling assumptions and because its architecture uses a strong form of parameter sharing between the f_i functions, which is not justified in a setup like ours. GSF performs always better than DAG-GNN and NOTEARS but performs as good as CAM and GraN-DAG only about half the time. Among the continuous approaches considered, GraN-DAG is the best performing on all our synthetic tasks.

We also considered synthetic data sets which do not satisfy the additive gaussian noise assumption present in GraN-DAG, NOTEARS and CAM. We considered two kinds of *post nonlinear causal*

models (Zhang & Hyvärinen, 2009), PNL-GP and PNL-MULT (see Appendix A.4 for details about their generation). A post nonlinear model has the form $X_j := g_j(f_j(X_{\pi_j^{\mathcal{G}}}) + N_j)$ where N_j is a noise variable. Note that GraN-DAG does not have the representational power to express these kinds of relationship (at least with the gaussian model considered in this section). This is also true about CAM and NOTEARS. However, these data sets differ from the previous additive noise setup only by the nonlinearity g_j, hence offering a case of mild model misspecification. The results are reported in Table 10 of the appendix. GraN-DAG and CAM are outperforming DAG-GNN and NOTEARS except in SID for certain data sets where all methods score similarly to RANDOM. GraN-DAG and CAM have similar performance on all data sets except one where CAM is better. GSF performs worst than GraN-DAG (in both SHD and SID) on PNL-GP but not on PNL-MULT where it performs better in SID.

4.2 Real and pseudo-real data

We have tested all methods considered so far on a well known data set which measures the expression level of different proteins and phospholipids in human cells (Sachs et al., 2005). We trained only on the $n = 853$ observational samples. This dataset and its ground truth graph proposed in Sachs et al. (2005) (11 nodes and 17 edges) are often used in the probabilistic graphical model literature (Koller & Friedman, 2009). We also consider pseudo-real data sets sampled from the SynTReN generator (Van den Bulcke, 2006). This generator was designed to create synthetic transcriptional regulatory networks and produces simulated gene expression data that approximates experimental data. See Appendix A.4 for details of the generation.

Note that in applications, it is not clear whether the DAG $\mathcal{G}$ is identifiable from the distribution. Nevertheless, we apply procedures to estimate it. This departure from identifiable setups is an occasion to explore different modelling assumptions for GraN-DAG. In addition to the model presented in Section 4.1, we consider an alternative, denoted GraN-DAG++, which allows the variance parameters $\hat{\sigma}^2_{(i)}$ to depend on the parent variables $X_{\pi_i^{\mathcal{G}}}$ through the NN, i.e. $\theta_{(i)} := (\hat{\mu}_{(i)}, \log \hat{\sigma}^2_{(i)})$.

In addition to metrics used in Section 4.1, we also report SHD-C. To compute the SHD-C between two DAGs, we first map each of them to their corresponding CPDAG and measure the SHD between the two. This metric is useful to compare algorithms which only outputs a CPDAG like GSF, GES and PC to other methods which outputs a DAG. Results are reported in Table 3.

Table 3: Results on real and pseudo-real data

	Protein signaling data set			SynTReN (20 nodes)		
	SHD	SHD-C	SID	SHD	SHD-C	SID
GraN-DAG	13	11	47	**34.0±8.5**	**36.4±8.3**	161.7±53.4
GraN-DAG++	13	10	48	**33.7±3.7**	**39.4±4.9**	127.5±52.8
DAG-GNN	16	21	44	93.6±9.2	97.6±10.3	157.5±74.6
NOTEARS	21	21	44	151.8±28.2	156.1±28.7	**110.7±66.7**
CAM	**12**	**9**	55	**40.5±6.8**	**41.4±7.1**	152.3±48
GSF	18	10	[44, 61]	61.8±9.6	63.3±11.4	**[76.7±51.1, 109.9±39.9]**
GES	26	28	**[34, 45]**	82.6±9.3	85.6±10	[157.2±48.3, 168.8±47.8]
PC	17	11	[47, 62]	**41.2±5.1**	**42.4±4.6**	[154.8±47.6, 179.3±55.6]
RANDOM	21	20	60	84.7±53.8	86.7±55.8	175.8±64.7

First, all methods perform worse than what was reported for graphs of similar size in Section 4.1, both in terms of SID and SHD. This might be due to the lack of identifiability guarantees we face in applications. On the protein data set, GraN-DAG outperforms CAM in terms of SID (which differs from the general trend of Section 4.1) and arrive almost on par in terms of SHD and SHD-C. On this data set, DAG-GNN has a reasonable performance, beating GraN-DAG in SID, but not in SHD. On SynTReN, GraN-DAG obtains the best SHD but not the best SID. Overall, GraN-DAG is always competitive with the best methods of each task.

5 Related Work

Most methods for structure learning from observational data make use of some identifiability results similar to the ones raised in Section 2.2. Roughly speaking, there are two classes of methods: *independence-based* and *score-based* methods. GraN-DAG falls into the second class.

Score-based methods (Koller & Friedman, 2009; Peters et al., 2017) cast the problem of structure learning as an optimization problem over the space of structures (DAGs or CPDAGs). Many popular algorithms tackle the combinatorial nature of the problem by performing a form of greedy search. GES (Chickering, 2003) is a popular example. It usually assumes a linear parametric model with Gaussian noise and greedily search the space of CPDAGs in order to optimize the Bayesian information criterion. GSF (Huang et al., 2018), is based on the same search algorithm as GES, but uses a generalized score function which can model nonlinear relationships. Other greedy approaches rely on parametric assumptions which render $\mathcal{G}$ fully identifiable. For example, Peters & Bühlman (2014) relies on a linear Gaussian model with equal variances to render the DAG identifiable. RESIT (Peters et al., 2014), assumes nonlinear relationships with additive Gaussian noise and greedily maximizes an independence-based score. However, RESIT do not scale well to graph of more than 20 nodes. CAM (Bühlmann et al., 2014) decouples the search for the optimal node ordering from the parents selection for each node and assumes an additive noise model (ANM) (Peters et al., 2017) in which the nonlinear functions are additive. As mentioned in Section 2.3, NOTEARS, proposed in Zheng et al. (2018), tackles the problem of finding an optimal DAG as a continuous constrained optimization program. This is a drastic departure from previous combinatorial approaches which enables the application of well studied numerical solvers for continuous optimizations. Recently, Yu et al. (2019) proposed DAG-GNN, a graph neural network architecture (GNN) which can be used to learn DAGs via the maximization of an evidence lower bound. By design, a GNN makes use of parameter sharing which we hypothesize is not well suited for most DAG learning tasks. To the best of our knowledge, DAG-GNN is the first approach extending the NOTEARS algorithm for structure learning to support nonlinear relationships. Although Yu et al. (2019) provides empirical comparisons to linear approaches, namely NOTEARS and FGS (a faster extension of GES) (Ramsey et al., 2017), comparisons to greedy approaches supporting nonlinear relationships such as CAM and GSF are missing. Moreover, GraN-DAG significantly outperforms DAG-GNN on our benchmarks. There exists certain score-based approaches which uses integer linear programming (ILP) (Jaakkola et al., 2010; Cussens, 2011) which internally solve continuous linear relaxations. Connections between such methods and the continuous constrained approaches are yet to be explored.

Methods for causal discovery using NNs have already been proposed. SAM (Kalainathan et al., 2018) learns conditional NN generators using adversarial losses but does not enforce acyclicity. CGNN (Goudet et al., 2018), when used for multivariate data, requires an initial skeleton to learn the different functional relationships.

GraN-DAG has strong connections with MADE (Germain et al., 2015), a method used to learn distributions using a masked NN which enforce the so-called *autoregressive property*. The autoregressive property and acyclicity are in fact equivalent. MADE does not learn the weight masking, it fixes it at the beginning of the procedure. GraN-DAG could be used with a unique NN taking as input all variables and outputting parameters for all conditional distributions. In this case, it would be similar to MADE, except the variable ordering would be learned from data instead of fixed *a priori*.

6 Conclusion

The continuous constrained approach to structure learning has the advantage of being more global than other approximate greedy methods (since it updates all edges at each step based on the gradient of the score but also the acyclicity constraint) and allows to replace task-specific greedy algorithms by appropriate off-the-shelf numerical solvers. In this work, we have introduced GraN-DAG, a novel score-based approach for structure learning supporting nonlinear relationships while leveraging a continuous optimization paradigm. The method rests on a novel characterization of acyclicity for NNs based on the work of Zheng et al. (2018). We showed GraN-DAG outperforms other gradient-based approaches, namely NOTEARS and its recent nonlinear extension DAG-GNN, on the synthetic data sets considered in Section 4.1 while being competitive on real and pseudo-real data sets of Section 4.2. Compared to greedy approaches, GraN-DAG is competitive across all datasets considered. To the best of our knowledge, GraN-DAG is the first approach leveraging the continuous paradigm introduced in Zheng et al. (2018) which has been shown to be competitive with state of the art methods supporting nonlinear relationships.

REFERENCES

J. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Learning from narrated instruction videos. *TPAMI*, 40(9):2194–2208, Sep. 2018.

A.-L. Barabási. Scale-free networks: a decade and beyond. *Science*, 2009.

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

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

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

D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 2003.

A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *SIAM review*, 2009.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 2006.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

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

Biwei Huang, Kun Zhang, Yizhu Lin, Bernhard Schölkopf, and Clark Glymour. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian Network Structure using LP Relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, and M. Sebag. Sam: Structural agnostic model, causal discovery and penalized adversarial learning. *arXiv preprint arXiv:1803.04929*, 2018.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. MIT Press, 2009.

S. Magliacane, T. van Ommen, T. Claassen, S. Bongers, P. Versteeg, and J.M. Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. In *Advances in Neural Information Processing Systems 31*, 2018.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

J. Pearl. The seven tools of causal inference, with reflections on machine learning. *Commun. ACM*, 2019.

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

J. Peters and P. Bühlmann. Structural intervention distance (sid) for evaluating causal graphs. *Neural Computation*, 2013.

J. Peters and P. Bühlmann. Structural intervention distance (SID) for evaluating causal graphs. *Neural Computation*, 2015.

J. Peters, D. Janzing, and B. Schölkopf. Causal inference on discrete data using additive noise models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 2014.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference - Foundations and Learning Algorithms*. MIT Press, 2017.

Lutz Prechelt. Early stopping - but when? In *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*, 1997.

J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *I. J. Data Science and Analytics*, 2017.

K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 2005.

S. Shimizu, P.O. Hoyer, A. Hyvärinen, and A. Kerminen. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 2006.

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

T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Tim Van den Bulcke, et al. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. 2006.

Y. Yu, J. Chen, T. Gao, and M. Yu. DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

K. Zhang and A. Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

Kun Zhang, Zhikun Wang, Jiji Zhang, and Bernhard Schölkopf. On estimation of functional causal models: General results and application to the post-nonlinear causal model. *ACM Trans. Intell. Syst. Technol.*, 2015.

X. Zheng, B. Aragam, P.K. Ravikumar, and E.P. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, 2018.

A APPENDIX

A.1 OPTIMIZATION

Let us recall the augmented Lagrangian:

$$\max_{\phi} \mathcal{L}(\phi, \lambda_t, \mu_t) \triangleq \mathbb{E}_{X \sim P_X} \sum_{i=1}^{d} \log p_i(X_i | X_{\pi_i^\phi}; \phi_{(i)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \tag{12}$$

where λ_t and μ_t are the Lagrangian and penalty coefficients of the tth subproblem, respectively. In all our experiments, we initialize those coefficients using $\lambda_0 = 0$ and $\mu_0 = 10^{-3}$. We approximately solve each non-convex subproblem using RMSprop (Tieleman & Hinton, 2012), a stochastic gradient descent variant popular for NNs. We use the following gradient estimate:

$$\begin{aligned} &\nabla_\phi \mathcal{L}(\phi, \lambda_t, \mu_t) \approx \nabla_\phi \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t) \\ &\text{with } \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t) \triangleq \frac{1}{|B|} \sum_{x \in B} \sum_{i=1}^{d} \log p_i(x_i | x_{\pi_i^\phi}; \phi_{(i)}) - \lambda_t h(\phi) - \frac{\mu_t}{2} h(\phi)^2 \end{aligned} \tag{13}$$

where B is a minibatch sampled from the data set and $|B|$ is the minibatch size. The gradient estimate $\nabla_\phi \hat{\mathcal{L}}_B(\phi, \lambda_t, \mu_t)$ can be computed using standard deep learning libraries. We consider a subproblem has converged when $\hat{\mathcal{L}}_H(\phi, \lambda_t, \mu_t)$ evaluated on a held-out data set H stops increasing. Let ϕ_t^* be the approximate solution to subproblem t. Then, λ_t and μ_t are updated according to the following rule:

$$\begin{aligned} \lambda_{t+1} &\leftarrow \lambda_t + \mu_t h\left(\phi_t^*\right) \\ \mu_{t+1} &\leftarrow \begin{cases} \eta \mu_t, & \text{if } h\left(\phi_t^*\right) > \gamma h\left(\phi_{t-1}^*\right) \\ \mu_t, & \text{otherwise} \end{cases} \end{aligned} \tag{14}$$

with $\eta = 10$ and $\gamma = 0.9$. Each subproblem t is initialized using the previous subproblem solution ϕ_{t-1}^*. The augmented Lagrangian method stops when $h(\phi) \leq 10^{-8}$.

A.2 THRESHOLDING TO ENSURE ACYCLICITY

The augmented Lagrangian outputs ϕ_T^* where T is the number of subproblems solved before declaring convergence. Note that the weighted adjacency matrix $A_{\phi_T^*}$ will most likely not represent an acyclic graph, even if we threshold as we learn, as explained in Section 3.4. We need to remove additional edges to obtain a DAG (edges are removed using the mask presented in Section 3.4). One option would be to remove edges one by one until a DAG is obtained, starting from the edge (i, j) with the lowest $(A_{\phi_T^*})_{ij}$ up to the edge with the highest $(A_{\phi_T^*})_{ij}$. This amounts to gradually increasing the threshold ϵ until $A_{\phi_T^*}$ is acyclic. However, this approach has the following flaw: It is possible to have $(A_{\phi_T^*})_{ij}$ significantly higher than zero while having $\theta_{(j)}$ almost completely independent of variable X_i. This can happen for at least two reasons. First, the NN paths from input i to output k might end up cancelling each others, rendering the input i inactive. Second, some neurons of the NNs might always be saturated for the observed range of inputs, rendering some NN paths *effectively inactive* without being inactive in the sense described in Section 3.1. Those two observations illustrate the fact that having $(A_{\phi_T^*})_{ij} = 0$ is only a sufficient condition to have $\theta_{(j)}$ independent of variable X_i and not a necessary one.

To avoid this issue, we consider the following alternative. Consider the function $\mathcal{L} : \mathbb{R}^d \mapsto \mathbb{R}^d$ which maps all d variables to their respective conditional likelihoods, i.e. $\mathcal{L}_i(X) \triangleq p_i(X_i \mid X_{\pi_i^{\phi_T^*}}) \ \forall i$. We consider the following expected Jacobian matrix

$$\mathcal{J} \triangleq \mathbb{E}_{X \sim P_X} \left| \frac{\partial \mathcal{L}}{\partial X} \right|^\top \tag{15}$$

where $\left| \frac{\partial \mathcal{L}}{\partial X} \right|$ is the Jacobian matrix of $\mathcal{L}$ evaluated at X, in absolute value (element-wise). Similarly to $(A_{\phi_T^*})_{ij}$, the entry $\mathcal{J}_{ij}$ can be loosely interpreted as the strength of edge (i, j). We propose removing edges starting from the lowest $\mathcal{J}_{ij}$ to the highest, stopping as soon as acyclicity is achieved. We believe $\mathcal{J}$ is better than $A_{\phi_T^*}$ at capturing which NN inputs are effectively inactive since it takes into account NN paths cancelling each others and saturated neurons. Empirically, we found that using $\mathcal{J}$ instead of $A_{\phi_T^*}$ yields better results, and thus we report the results with $\mathcal{J}$ in this paper.

A.3 PRELIMINARY NEIGHBORHOOD SELECTION AND DAG PRUNING

PNS: For graphs of 50 nodes or more, GraN-DAG performs a *preliminary neighborhood selection* (PNS) similar to what has been proposed in Bühlmann et al. (2014). This procedure applies a variable selection method to get a set of possible parents for each node. This is done by fitting an *extremely randomized trees* (Geurts et al., 2006) (using `ExtraTreesRegressor` from `scikit-learn`) for each variable against all the other variables. For each node a feature importance score based on the gain of purity is calculated. Only nodes that have a feature importance score higher than $0.75 \cdot$ `mean` are kept as potential parent, where `mean` is the mean of the feature importance scores of all nodes. Although the use of PNS in CAM was motivated by gains in computation time, GraN-DAG uses it to avoid overfitting, without reducing the computation time.

Pruning: Once the thresholding is performed and a DAG is obtained as described in A.2, GraN-DAG performs a pruning step identical to CAM (Bühlmann et al., 2014) in order to remove spurious edges. We use the implementation of Bühlmann et al. (2014) based on the R function `gamboost` from the `mboost` package. For each variable X_i, a generalized additive model is fitted against the current parents of X_i and a significance test of covariance is applied. Parents with a p-value higher than 0.001 are removed from the parent set. Similarly to what Bühlmann et al. (2014) observed, this pruning phase generally has the effect of greatly reducing the SHD without considerably changing the SID.

Ablation study: In Table 4, we present an ablation study which shows the effect of adding PNS and pruning to GraN-DAG on different performance metrics and on the negative log-likelihood (NLL) of the training and validation set. Note that, before computing both NLL, we reset all parameters of GraN-DAG except the mask and retrained the model on the training set without any acyclicity constraint (acyclicity is already ensure by the masks at this point). This retraining procedure is important since the pruning removes edges (i.e. some additional NN inputs are masked) and it affects the likelihood of the model (hence the need to retrain).

Table 4: PNS and pruning ablation study for GraN-DAG (averaged over 10 datasets from ER1 with 50 nodes)

PNS	Pruning	SHD	SID	NLL (train)	NLL (validation)
False	False	1086.8±48.8	31.6±23.6	0.36±0.07	1.44±0.21
True	False	540.4±70.3	17.4±16.7	0.52±0.08	1.16±0.17
False	True	11.8±5.0	39.7±25.5	0.78±0.12	0.84±0.12
True	True	6.1±3.3	29.3±19.5	0.78±0.13	0.83±0.12

A first observation is that adding PNS and pruning improve the NLL on the validation set while deteriorating the NLL on the training set, showing that those two steps are indeed reducing overfitting. Secondly, the effect on SHD is really important while the effect on SID is almost nonexistent. This can be explained by the fact that SID has more to do with the ordering of the nodes than with false positive edges. For instance, if we have a complete DAG with a node ordering coherent with the ground truth graph, the SID is zero, but the SHD is not due to all the false positive edges. Without the regularizing effect of PNS and pruning, GraN-DAG manages to find a DAG with a good ordering but with many spurious edges (explaining the poor SHD, the good SID and the big gap between the NLL of the training set and validation set). PNS and pruning helps reducing the number of spurious edges, hence improving SHD.

We also implemented PNS and pruning for NOTEARS and DAG-GNN to see whether their performance could also be improved. Table 5 reports an ablation study for DAG-GNN and NOTEARS. First, the SHD improvement is not as important as for GraN-DAG and is almost not statistically significant. The improved SHD does not come close to performance of GraN-DAG. Second, PNS and pruning do not have a significant effect of SID, as was the case for GraN-DAG. The lack of improvement for those methods is probably due to the fact that they are not overfitting like GraN-DAG, as the training and validation (unregularized) scores shows. NOTEARS captures only linear relationships, thus it will have a hard time overfitting nonlinear data and DAG-GNN uses a strong form of parameter sharing between its conditional densities which possibly cause underfitting in a setup where all the parameters of the conditionals are sampled independently.

Moreover, DAG-GNN and NOTEARS threshold aggressively their respective weighted adjacency matrix at the end of training (with the default parameters used in the code), which also acts as a form of heavy regularization, and allow them to remove many spurious edges. GraN-DAG without PNS and pruning does not threshold as strongly by default which explains the high SHD of Table 4. To test this explanation, we removed all edges (i, j) for which $(A_\phi)_{ij} < 0.3$[3] for GraN-DAG and obtained an SHD of 29.4±15.9 and an SID of 85.6±45.7, showing a significant improvement over NOTEARS and DAG-GNN, even without PNS and pruning.

Table 5: PNS and pruning ablation study for DAG-GNN and NOTEARS (averaged over 10 datasets from ER1 with 50 nodes)

Algorithm	PNS	Pruning	SHD	SID	Score (train)	Score (validation)
DAG-GNN	False	False	56.8±11.1	322.9±103.8	-2.8±1.5	-2.2±1.6
	True	False	55.5±10.2	314.5±107.6	-2.1±1.6	-2.1±1.7
	False	True	49.4±7.8	325.1±103.7	-1.8±1.1	-1.8±1.2
	True	True	47.7±7.3	316.5±105.6	-1.9±1.6	-1.9±1.6
NOTEARS	False	False	64.2±9.5	327.1±110.9	-23.1±1.8	-23.2±2.1
	True	False	54.1±10.9	321.5±104.5	-25.2±2.7	-25.4±2.8
	False	True	49.5±8.8	327.7±111.3	-26.7±2.0	-26.8±2.1
	True	True	49.0±7.6	326.4±106.9	-26.23±2.2	-26.4±2.4

A.4 Details on data sets generation

Post nonlinear data sets: Like the synthetic data introduced in Section 4.1, a ground truth DAG $\mathcal{G}$ is randomly sampled following the ER scheme and then the data is generated. PNL-GP is generated following $X_j|X_{\pi_j^\mathcal{G}} \sim \sigma(f_j(X_{\pi_j^\mathcal{G}}) + Laplace(0, l_j))\ \forall j$ with the functions f_j independently sampled from a Gaussian process with bandwidth one and $l_j \sim \mathcal{U}[0, 1]$. Root variables are sampled from $\mathcal{U}[-1, 1]$. In the two-variable case, this model is identifiable following the Corollary 9 from Zhang & Hyvärinen (2009). To get identifiability according to this corollary, it is important to use non-Gaussian noise, explaining our design choices. PNL-MULT is generated following $X_j|X_{\pi_j^\mathcal{G}} \sim \exp(\log(\sum_{i\in\pi_j^\mathcal{G}} X_i) + |\mathcal{N}(0, \sigma_j^2)|)\ \forall j$ where $\sigma_j^2 \sim \mathcal{U}[0, 1]$. Root variables are sampled from $\mathcal{U}[0, 2]$. This model is adapted from Zhang et al. (2015).

SynTReN: Ten datasets have been generated using the SynTReN generator (`http://bioinformatics.intec.ugent.be/kmarchal/SynTReN/index.html`) using the software default parameters except for the *probability for complex 2-regulator interactions* that was set to 1 and the random seeds used were 0 to 9. Each dataset contains 500 samples and comes from a 20 nodes graph.

Graph types: *Erdős-Rényi* (ER) graphs are generated by randomly sampling a topological order and by adding directed edges were it is allowed independently with probability $p = \frac{2e}{d^2-d}$ were e is the expected number of edges in the resulting DAG. *Scale-free* (SF) graphs were generated using the BarabsiAlbert model (Barabási & Albert, 1999) which is based on preferential attachment. Nodes are added one by one. Between the new node and the existing nodes, m edges (where m is equal to d or $4d$) will be added. An existing node i have the probability $p(k_i) = \frac{k_i}{\sum_j k_j}$ to be chosen, where k_i represents the degree of the node i. While ER graphs have a degree distribution following a Poisson distribution, SF graphs have a degree distribution following a power law: few nodes, often called *hubs*, have a high degree. Barabási (2009) have stated that these types of graphs have similar properties to real-world networks which can be found in many different fields, although these claims remain controversial (Clauset et al., 2009).

A.5 Supplementary experiments

The results for 20 and 100 nodes are presented in Table 6 and 7 using exactly the same additive gaussian data set types introduced in Section 4.1. The conclusions drawn remain similar to the 10

[3]This was the default value of thresholding used in NOTEARS and DAG-GNN.

and 50 nodes experiments. For GES and PC, the SHD and SID are respectively presented in Table 8 and 9. Their performances do not compare favorably to the GraN-DAG nor CAM. Table 10 contains the performance of GraN-DAG and other baselines on post nonlinear data described in Section 4.1. Note that GSF results are missing for two data set types in Tables 7 and 10. This is because the search algorithm could not finish inside 12 hours, even when the maximal in-degree was limited to 5. All other methods could run in less than 6 hours.

Figure 1 shows the entries of the weighted adjacency matrix A_ϕ as training proceeds in a typical run for 10 nodes.

Table 6: Results for ER and SF graphs of 20 nodes

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**4.0 ±3.4**	**17.9±19.5**	**45.2±10.7**	**165.1±21.0**	**7.6±2.5**	28.8±10.4	**36.8±5.1**	**62.5±18.8**
DAG-GNN	25.6±7.5	109.1±53.1	75.0±7.7	344.8±17.0	19.5±1.8	60.1±12.8	49.5±5.4	115.2±33.3
NOTEARS	30.3±7.8	107.3±47.6	79.0±8.0	346.6±13.2	23.9±3.5	69.4±19.7	52.0±4.5	120.5±32.5
CAM	**2.7±1.8**	**10.6±8.6**	**41.0±11.9**	**157.9±41.2**	**5.7±2.6**	**23.3±18.0**	**35.6±4.5**	**59.1±18.8**
GSF	12.3±4.6	[15.0±19.9 45.6±22.9]	**41.8±13.8**	**[153.7±49.4 201.6±37.9]**	**7.4±3.5**	**[5.7±7.1 27.3±13.2]**	**38.6±3.6**	**[54.9±14.4 86.7±24.2]**
RANDOM	103.0±39.6	94.3±53.0	117.5±25.9	298.5±28.7	105.2±48.8	81.1±54.4	121.5±28.5	204.8±38.5

Table 7: Results for ER and SF graphs of 100 nodes

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**15.1±6.0**	**83.9±46.0**	**206.6±31.5**	**4207.3±419.7**	**59.2±7.7**	**265.4±64.2**	**262.7±19.6**	**872.0±130.4**
DAG-GNN	110.2±10.5	883.0±320.9	379.5±24.7	8036.1±656.2	97.6±1.5	438.6±112.7	316.0±14.3	1394.6±165.9
NOTEARS	125.6±12.1	913.1±343.8	387.8±25.3	8124.7±577.4	111.7±5.4	484.3±138.4	327.2±15.8	1442.8±210.1
CAM	**17.3±4.5**	**124.9±65.0**	**186.4±28.8**	**4601.9±482.7**	**52.7±9.3**	**230.3±36.9**	**255.6±21.7**	**845.8±161.3**
GSF	66.8±7.3	[104.7±59.5 238.6±59.3]	$>$ 12 hours	— —	**71.4±11.2**	**[212.7±71.1 325.3±105.2]**	**275.9±21.0**	**[793.9±152.5 993.4±149.2]**
RANDOM	1561.6±1133.4	1175.3±547.9	2380.9±1458.0	7729.7±1056.0	2222.2±1141.2	1164.2±593.3	2485.0±1403.9	4206.4±1642.1

Table 8: SHD for GES and PC (against GraN-DAG for reference)

	10 nodes		20 nodes		50 nodes		100 nodes	
	ER1	ER4	ER1	ER4	ER1	ER4	ER1	ER4
GraN-DAG	1.7±2.5	8.3±2.8	4.0 ±3.4	45.2±10.7	5.1±2.8	102.6±21.2	15.1±6.0	206.6±31.5
GES	13.8±4.8	32.3±4.3	43.3±12.4	94.6±9.8	106.6±24.7	254.4±39.3	292.9±33.6	542.6±51.2
PC	8.4±3	34±2.6	20.136.4±6.5	73.1±5.8	44.0±11.6	183.6±20	95.2±9.1	358.8±20.6
	SF1	SF4	SF1	SF4	SF1	SF4	SF1	SF4
GraN-DAG	1.2±1.1	9.9±4.0	7.6±2.5	36.8±5.1	25.5±6.2	111.3±12.3	59.2±7.7	262.7±19.6
GES	8.1±2.4	17.4±4.5	26.2±7.5	50.7±6.2	73.9±7.4	178.8±16.5	190.3±22	408.7±24.9
PC	4.8±2.4	16.4±2.8	13.6±2.1	44.4±4.6	43.1±5.7	135.4±10.7	97.6±6.6	314.2±17.5

Table 9: Lower and upper bound on the SID for GES and PC (against GraN-DAG for reference). See Appendix A.6 for details on how to compute SID for CPDAGs.

	10 nodes		20 nodes		50 nodes		100 nodes	
	ER1	ER4	ER1	ER4	ER1	ER4	ER1	ER4
GraN-DAG	1.7±3.1	21.8±8.9	17.9±19.5	165.1±21.0	22.4±17.8	1060.1±109.4	83.9±46.0	4207.3±419.7
GES	[24.1±17.3 27.2±17.5]	[68.5±10.5 75±7]	[62.1±44 65.7±44.5]	[301.9±19.4 319.3±13.3]	[150.9±72.7 155.1±74]	[1996.6±73.1 2032.9±88.7]	[582.5±391.1 598.9±408.6]	[8054±524.8 8124.2±470.2]
PC	[22.6±15.5 27.3±13.1]	[78.1±7.4 79.2±5.7]	[80.9±51.1 94.9±46.1]	[316.7±25.7 328.7±25.6]	[222.7±138 256.7±127.3]	[2167.9±88.4 2178.8±80.8]	[620.7±240.9 702.5±255.8]	[8236.9±478.5 8265.4±470.2]
	SF1	SF4	SF1	SF4	SF1	SF4	SF1	SF4
GraN-DAG	4.1±6.1	16.4±6.0	28.8±10.4	62.5±18.8	90.0±18.9	271.2±65.4	265.4±64.2	872.0±130.4
GES	[11.6±9.2 16.4±11.7]	[39.3±11.2 43.9±14.9]	[54.9±23.1 57.9±24.6]	[89.5±38.4 105.1±44.3]	[171.6±70.1 182.7±77]	[496.3±154.1 529.7±184.5]	[511.5±257.6 524±252.2]	[1421.7±247.4 1485.4±233.6]
PC	[8.3±4.6 16.8±12.3]	[36.5±6.2 41.7±6.9]	[42.2±14 59.7±14.9]	[95.6±37 118.5±30]	[124.2±38.3 167.1±41.4]	[453.2±115.9 538±143.7]	[414.5±124.4 486.5±120.9]	[1369.2±259.9 1513.7±296.2]

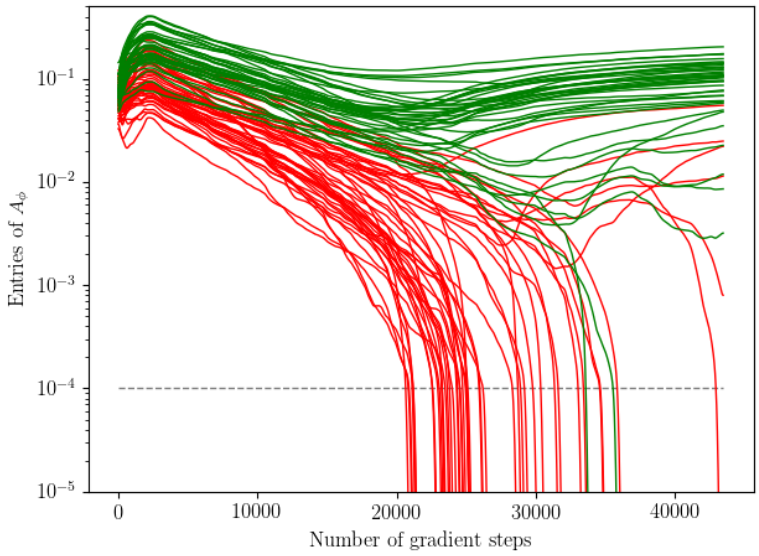

Figure 1: Entries of the weighted adjacency matrix A_ϕ as training proceeds in GraN-DAG for a synthetic data set ER4 with 10 nodes. Green curves represent edges which appear in the ground truth graph while red ones represent edges which do not. The horizontal dashed line at 10^{-4} is the threshold ϵ introduced in Section 3.4. We can see that GraN-DAG successfully recovers most edges correctly while keeping few spurious edges.

Table 10: Synthetic post nonlinear data sets

		PNL-GP SHD	PNL-GP SID	PNL-MULT SHD	PNL-MULT SID
10 nodes ER1	GraN-DAG	**1.6±3.0**	**3.9±8.0**	**13.1±3.8**	35.7±12.3
	DAG-GNN	11.5±6.8	32.4±19.3	17.900±6.2	40.700±14.743
	NOTEARS	10.7±5.5	34.4±19.1	**14.0±4.0**	38.6±11.9
	CAM	**1.5±2.6**	**6.8±12.1**	**12.0±6.4**	36.3±17.7
	GSF	**6.2±3.3**	[7.7±8.7, 18.9±12.4]	**10.7±3.0**	**[9.8±11.9, 25.3±11.5]**
	RANDOM	23.8±2.9	36.8±19.1	23.7±2.9	37.7±20.7
10 nodes ER4	GraN-DAG	**10.9±6.8**	**39.8±21.1**	32.1±4.5	77.7±5.9
	DAG-GNN	32.3±4.3	75.8±9.3	37.0±3.1	82.7±6.4
	NOTEARS	34.1±3.2	80.8±5.5	37.7±3.0	81.700±7.258
	CAM	**8.4±4.8**	**30.5±20.0**	34.4±3.9	79.6±3.8
	GSF	25.0±6.0	[44.3±14.5, 66.1±10.1]	31.3±5.4	**[58.6±8.1, 76.4±9.9]**
	RANDOM	35.0±3.3	80.0±5.1	33.6±3.5	76.2±7.3
50 nodes ER1	GraN-DAG	16.5±7.0	64.1±35.4	**38.2±11.4**	213.8±114.4
	DAG-GNN	56.5±11.1	334.3±80.3	83.9±23.8	507.7±253.4
	NOTEARS	50.1±9.9	319.1±76.9	78.5±21.5	425.7±197.0
	CAM	**5.1±2.6**	**10.7±12.4**	**44.9±9.9**	284.3±124.9
	GSF	31.2±6.0	[59.5±34.1, 122.4±32.0]	**46.3±12.1**	**[65.8±62.2, 141.6±72.6]**
	RANDOM	688.4±4.9	307.0±98.5	691.3±7.3	488.0±247.8
50 nodes ER4	GraN-DAG	**68.7±17.0**	**1127.0±188.5**	**211.7±12.6**	2047.7±77.7
	DAG-GNN	203.8±18.9	2173.1±87.7	246.7±16.1	2239.1±42.3
	NOTEARS	189.5±16.0	2134.2±125.6	**220.0±9.9**	2175.2±58.3
	CAM	**48.2±10.3**	**899.5±195.6**	**208.1±14.8**	2029.7±55.4
	GSF	105.2±15.5	[1573.7±121.2, 1620±102.8]	$>$ 12 hours	—
	RANDOM	722.3±9.0	1897.4±83.7	710.2±9.5	1935.8±56.9

A.6 METRICS

SHD takes two partially directed acyclic graphs (PDAG) and counts the number of edge for which the edge type differs in both PDAGs. There are four edge types: $i \leftarrow j$, $i \rightarrow j$, $i - j$ and $i \quad j$. Since this distance is defined over the space of PDAGs, we can use it to compare DAGs with DAGs,

DAGs with CPDAGs and CPDAGs with CPDAGs. When comparing a DAG with a CPDAG, having $i \leftarrow j$ instead of $i - j$ counts as a mistake.

SHD-C is very similar to SHD. The only difference is that both DAGs are first mapped to their respective CPDAGs before measuring the SHD.

Introduced by Peters & Bühlmann (2015), SID counts the number of interventional distribution of the form $p(x_i|\, do(x_j = \hat{x}_j))$ that would be miscalculated using the *parent adjustment formula* (Pearl, 2009) if we were to use the predicted DAG instead of the ground truth DAG to form the parent adjustment set. Some care needs to be taken to evaluate the SID for methods outputting a CPDAG such as GES and PC. Peters & Bühlmann (2015) proposes to report the SID of the DAGs which have approximately the minimal and the maximal SID in the Markov equivalence class given by the CPDAG. See Peters & Bühlmann (2015) for more details.

A.7 HYPERPARAMETERS

All GraN-DAG runs were performed using the following set of hyperparameters. We used RMSprop as optimizer with learning rate of 10^{-2} for the first subproblem and 10^{-4} for all subsequent suproblems. Each NN has two hidden layers with 10 units (except for the real and pseudo-real data experiments which uses only 1 hidden layer). Leaky-ReLU is used as activation functions. The NN are initialized using the initialization scheme proposed in Glorot & Bengio (2010) also known as *Xavier initialization*. We used minibatches of 64 samples.

For NOTEARS, DAG-GNN, and GSF, we used the default hyperparameters found in the authors code. It happens rarely that NOTEARS and DAG-GNN returns a cyclic graph. In those cases, we removed edges starting from the weaker ones to the strongest (according to their respective weighted adjacency matrices), stopping as soon as acyclicity is acheived. For GES and PC, we used default hyperparameters of the `pcalg` R package. For CAM, we used the the default hyperparameters found in the `CAM` R package, with default PNS and DAG pruning.