IDYNO: Learning Nonparametric DAGs from Interventional Dynamic Data

Tian Gao [1] Debarun Bhattacharjya [1] Elliot Nelson [1] Miao Liu [1] Yue Yu [2]

Abstract

Causal discovery in the form of a directed acyclic graph (DAG) for time series data has been widely studied in various domains. The resulting DAG typically represents a dynamic Bayesian network (DBN), capturing both the instantaneous and time-delayed relationships among variables of interest. We propose a new algorithm, IDYNO, to learn the DAG structure from potentially nonlinear time series data by using a continuous optimization framework that includes a continuous acyclicity constraint. The proposed algorithm is designed to handle both observational and interventional time series data. We demonstrate the promising performance of our method on synthetic benchmark datasets against state-of-the-art baselines. In addition, we show that the proposed method can more accurately learn the underlying structure of a sequential decision model, such as a Markov decision process, with a fixed policy in typical continuous control tasks.

1. Introduction & Related Work

Probabilistic graphical models (Pearl, 1988; Koller & Friedman, 2009; Pearl, 2009) have been widely adopted in applications of artificial intelligence since the 1990s. Dynamic probabilistic graphical models are particularly applicable in real-world problems, such as for neuroscience (Rajapakse & Zhou, 2007), molecular biology (Linzner et al., 2019) and computer vision (Meng et al., 2018) tasks, since they capture dynamics in temporal data like time series by explicitly modeling how variables change over time.

Perhaps the most popular dynamic graphical models in the literature are dynamic Bayesian networks (DBNs) (Dean & Kanazawa, 1989; Murphy, 2002), which are discrete-time models with an underlying directed acyclic graph (DAG) structure (e.g. Figure 1). While initial work in DBNs considered discrete variables, there has been plenty of subsequent literature on models with continuous variables, which are more appropriate for time series data involving continuous-valued measurements. For instance, DBNs can represent structured vector auto-regressive (SVAR) models from the statistics and econometrics literature (Reale & Wilson, 2001; 2002; Demiralp & Hoover, 2003; Swanson & Granger, 1997; Lanne et al., 2017; Kilian, 2013; Tank et al., 2019). Note that there are other related dynamic probabilistic graphical models for (discrete-time) time series data (Eichler, 1999; Dahlhaus, 2000) as well as for a parallel stream of research on continuous-time graphical models (Nodelman et al., 2002; Didelez, 2008; Gunawardana et al., 2011; Meek, 2014; Bhattacharjya et al., 2018).

Typical learning tasks for graphical models include parameter estimation and graph structure discovery. Structure discovery for both static and dynamic models aims at learning the graphical structure underlying the probabilistic model, usually in the form of a DAG, given observational data. Standard structure learning methods can be categorized as score-based, constraint-based, or hybrid methods that combine the approaches. Score-based DAG learning methods find a graphical model that best fits the data while also controlling the complexity of the DAG, according to a scoring function (Heckerman et al., 1995; Chickering, 2002). On the other hand, constraint-based methods identify a structure that conforms to conditional independencies between variables as gauged by statistical tests (Spirtes et al., 2001; Tsamardinos et al., 2006; Colombo et al., 2012; Malinsky & Spirtes, 2019). Structure learning methods of both basic types are often super-exponential in complexity due to a combinatorial search over all possible graphs.

To address computational issues, there has been a recent trend towards score-based approaches for models involving continuous variables that formulate a more tractable continuous optimization problem, with algebraic characterization of the DAG (Zheng et al., 2018). Several works have since successfully extended the initially proposed formulation to nonlinear neural models (Yu et al., 2019; Lachapelle et al., 2020; Kalainathan et al., 2018; Ng et al., 2019; Zheng

[1]IBM Research, Yorktown Heights, NY, USA [2]Department of Mathematics, Lehigh University, Bethlehem, PA, USA. Correspondence to: Tian Gao <tgao@us.ibm.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

et al., 2020). Recently, this continuous formulation has been applied to dynamic DAG structure learning and shows promising performance (Pamfil et al., 2020).

It is well known that structure discovery for DAGs with *observational* data alone, i.e. independent and identically distributed (i.i.d.) samples from a joint distribution over all random variables, does not necessarily provide the entire DAG structure (Spirtes et al., 2001; Pearl, 2009; Peters & Bühlmann, 2014; Oates et al., 2016); the structure can only be identified up to a Markov equivalence class in general, if one makes further assumptions. The gold standard for causal discovery is achieved when one can perform experiments by intervening in a system and measuring the ramifications. This results in *interventional* data, which can be seen as an operation of replacing the observational distribution with another distribution and provides additional information beyond observational data alone, which could potentially identify the underlying DAG structure. With a sufficient number of interventions, DAGs are fully identifiable (Eberhardt et al., 2005; Eberhardt, 2012). There is substantial prior work on identifiability results and structure learning algorithms that incorporate interventional i.i.d. data for (static) Bayesian networks (Hauser & Bühlmann, 2012; Shanmugam et al., 2015; Yang et al., 2018; Brouillard et al., 2020; Jaber et al., 2020; Ke et al., 2020; Squires et al., 2020). However, we are not aware of prior work in structure learning for dynamic Bayes nets that considers interventional data, potentially along with observational data.

In this paper, we propose a graph structure learning approach for time series data, in the form of dynamic Bayesian networks, while leveraging interventional data in addition to standard observational data. Similar to the case of i.i.d. data, an intervention on time series data requires modifying the process that generates the random variables over time; this is achieved when an experimenter changes conditional distributions of random variables. Note that an intervention can be made for any variable at any time slice in general.

To utilize interventional data, we formulate learning as a continuous optimization problem, extending the recent algebraic characterization of DAGs in time series datasets, known as DYNOTEARS (Pamfil et al., 2020). There are two crucial innovations: 1) While DYNOTEARS applies a new (continuous DAG) constraint to observational linear time series data, we introduce a non-linear objective through neural models, thereby allowing for nonlinear temporal dynamics. 2) We formulate a modified objective and general solution approach that can handle different distributions on intervention targets, thereby vastly expanding the scope of learning to general interventions in time series data.

Our work is closely related to aspects of reinforcement learning (RL) such as factored MDPs (Boutilier et al., 1995), which involve optimizing a target node (reward) in a dynamically evolving process with an underlying graphical model (with partially or fully observable nodes), given control over a subset of nodes (decision/action nodes). In the context of offline RL (Levine et al., 2020), action nodes cannot be actively intervened upon, as only pre-existing data from the actions (possibly of other agents) are available; however, offline data from multiple policies effectively provides access to different interventions on fixed targets.

Contributions. Our main contributions are as follows:

- We propose a general DBN structure learning algorithm called IDYNO – an interventional extension of DYNOTEARS – that is capable of utilizing both observational and interventional time series data.
- We extend the baseline IDYNO to neural models for handling potentially complex and nonlinear time series data. The resulting methods can leverage perfect (hard) and imperfect (soft) interventions with known targets.
- We present identifiability results around interventional equivalence classes for our learning approach, under some specified assumptions.
- Through synthetic as well as simulated offline RL datasets, we show that our proposed IDYNO outperforms existing score-based methods by utilizing such interventional data.

2. Background

Consider a set of independent realizations of a stationary time series, with each individual realization of size T in the form of $X_t := [x_{t,i}]_{i=1}^{d} \in \mathbb{R}^d$. Here $t \in \{0, ..., T\}$ represents the time index, X_t represents the observed values of all d number of variables in an observational or interventional time series dataset, and $x_{t,i}$ denotes the i-th component of X_t. We use lower case letters for scalars, upper case letters for vectors, bold letters for matrices. As typically assumed in such models, there may exist instantaneous or contemporaneous influences as well as a time-delayed impact among variables. This is illustrated in the simple DBN in Figure 1.

2.1. Basic Notation

A **causal graphical model** is defined by a distribution P_X over a set of random variables $X \in \mathbb{R}^d$ and a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$ with nodes V and edges E. Each node $i \in V = \{1, ..., d\}$ is associated with a random variable x_i and each edge $(i, j) \in E$ represents a direct causal relation from variable x_i to x_j. With a slight abuse of notation, we will use X and V interchangeably. We assume the distribution P_X is Markov with respect to graph $\mathcal{G}$, which enables the factorized joint distribution as $P(X) = \prod_{j=1}^{d} p_j(x_j | x_{\pi_j^{G}})$, where π_j^G is the set of parents of node j in the graph $\mathcal{G}$ and x_B denotes the instantiations of a subset of X whose indices are $B \subset V$. We also assume

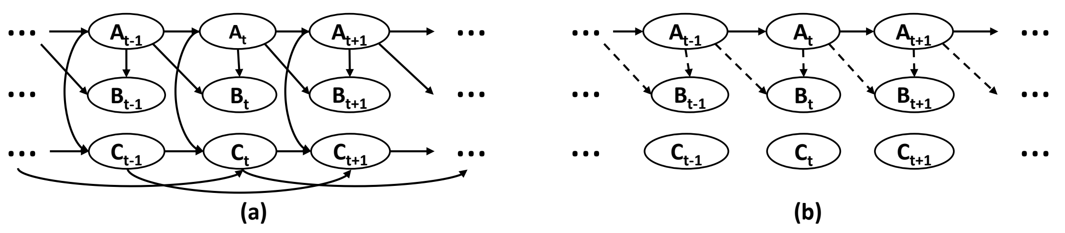

Figure 1. (a) A dynamic Bayesian network (DBN) with 3 nodes: A, B, C. The only intra-slice edges are from A to B and A to C. Note that C has inter-slice edges from the prior two periods whereas A and B have only single period inter-slice edges. (b) DBN from (a) with soft and hard interventions for variables B and C respectively.

causal sufficiency, i.e., there are no hidden common causes between any pair of variables in X (Peters et al., 2017).

In time series datasets, there are many possible ways to model P_X. We follow the typical setting in recent work (Pamfil et al., 2020), characterizing X through a standard SVAR model (Demiralp & Hoover, 2003; Swanson & Granger, 1997; Kilian, 2013):

$$X_t = X_t\mathbf{W} + X_{t-1}\mathbf{A_1} + ... + X_{t-p}\mathbf{A_p} + Z_t, \qquad (1)$$

where $t \in \{p, ..., T\}$ with horizon T, p is the autoregressive order, and $Z_t \in \mathbb{R}^d$ is a vector of noise variables drawn from any continuous distribution. We assume Z_t is independent of $Z_{t' \neq t}$ and of $X_{t'}$ for all t'. The $d \times d$ matrices $\mathbf{W}$ and $\mathbf{A}_i$, $i \in \{1, ..., p\}$, represent weighted adjacency matrices for the intra-slice and inter-slice edges in $\mathcal{G}$, respectively, and model the contemporaneous and time-lagged causal relations. As is typical in dynamic Bayesian networks, $\mathbf{W}$ and $\mathbf{A}_i$ are constrained to ensure that the underlying graph is acyclic; note that inter-slice edges are such that there are only edges from the previous time slice to the current time slice. Eq. 1 can be written in matrix form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}_1\mathbf{A_1} + ... + \mathbf{Y}_p\mathbf{A_p} + \mathbf{Z} \qquad (2)$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is a matrix whose rows are X_t, $\mathbf{Z} \in \mathbb{R}^{n\times d}$ is a matrix formed similarly by Z_t, and $\mathbf{Y_j}$, $j \in \{1, ...p\}$, are time lagged versions of $\mathbf{X}$. The number n is the effective sample size, which is equal to $T - p + 1$. Let $\mathbf{Y} = [\mathbf{Y_1}, ..., \mathbf{Y_p}]$ be a matrix with size $n \times pd$ concatenation of time-lagged data, $\mathbf{A} = [\mathbf{A_1}^T, ..., \mathbf{A_p}^T]^T$ be a matrix with size $pd \times d$, then the formulation takes the structural equation model (SEM) form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}\mathbf{A} + \mathbf{Z} \qquad (3)$$

We refer to such a data transformation as time-lagged, so that it has the same matrix form as i.i.d. data.

2.2. Interventions

An intervention on a variable x_j in a DAG $\mathcal{G}$ corresponds to replacing its factored conditional distribution $p_j(x_j|x_{\pi_j^{\mathcal{G}}})$ with another distribution $\hat{p}_j(x_j|x_{\pi_j^{\mathcal{G}}})$. The intervention can be performed on multiple variables simultaneously with a set of interventional targets $I \subseteq V$. Denote the interventional family by $\mathcal{I} := (I_1, ..., I_K)$, where K is the number of interventions. The joint likelihood for the kth intervention can be written as:

$$p^{(k)}(X) := \prod_{j \notin I_k} p_j^{(1)}(x_j|x_{\pi_j^{\mathcal{G}}}) \prod_{j \in I_k} p_j^{(k)}(x_j|x_{\pi_j^{\mathcal{G}}}) \qquad (4)$$

The typical intervention on the data from Eq. 4 is referred to as an *imperfect* (or *soft*) intervention. An intervention can also be *perfect* (or *hard*), if it completely removes the dependencies of a node x_j on its parents,[1] for example by setting $p_j^{(k)}(x_j|x_{\pi_j^{\mathcal{G}}}) = p_j^{(k)}(x_j)$, $\forall j \in I_k$. Real-world examples of these types of interventions include gene knockouts/knockdowns in biology or performing a fixed action for a decision variable with a deterministic policy in a reinforcement learning environment.

We note that while existing work can handle multiple interventional families, they focus on only one interventional target at a time (Brouillard et al., 2020). Our method in comparison allows for multiple interventions at any time.

2.3. Causal Structure Learning

The goal of typical causal structure learning tasks is to recover the DAG $\mathcal{G}$ using samples from P_X and/or from the interventional distributions. The exact recovery of the graph is typically costly, due to the super-exponential search space in number of nodes, and may not always be identifiable. In Section 1, we mentioned some prior literature on the subject.

The most relevant work here is the following continuous constrained optimization re-formulation for DAG learning that uses a continuous DAG constraint, $h(\mathbf{W}) = 0$ on the weighted adjacency matrix $\mathbf{W}$ to avoid the combinatorial

[1]A hard intervention in this sense can be stochastic, and does not necessarily imply deterministically fixing the value of a node.

search on the feasible solutions $\mathbf{W}$ (Zhang et al., 2019):

$$\min_{\theta, \mathbf{W}} L_\theta(\mathbf{X}; \mathbf{W}) - \lambda\Omega(\theta, \mathbf{W}) \quad s.t. \quad h(\mathbf{W}) = 0, \tag{5}$$

where L_θ is the loss function and θ indicates parameters other than $\mathbf{W}$, $\Omega(\theta, \mathbf{W})$ is a regularization term on model parameters and/or the edge complexity in $\mathbf{W}$ with a tunable regularization parameter λ. Zhang et al. (2019) propose $h(\mathbf{W}) = \text{Tr}(e^{\mathbf{W}}) - d$, where Tr represents the matrix trace, and shows the graph is acyclic if and only if the constraint $h(\mathbf{W}) = 0$. Typically, the loss f_θ can be the least square loss (Zhang et al., 2019) in linear structured equation models (SEM), or evidence lower bound (Yu et al., 2019), among other losses (Kalainathan et al., 2018). The problem is then approximately solved using an augmented Lagrangian procedure. Many extensions to the method have been proposed (Lachapelle et al., 2020; Ng et al., 2019).

For time series datasets, DYNOTEARS (Pamfil et al., 2020) extends the continuous optimization framework to DBNs by explicitly modeling the intra- and inter-slice adjacency matrices separately with a linear SEM model. Namely:

$$\min_{\theta, \mathbf{W}, \mathbf{A}} L_\theta(\mathbf{X}; \mathbf{W}, \mathbf{A}) - \lambda\Omega(\theta, \mathbf{W}) \;\; s.t. \;\; h(\mathbf{W}) = 0, \tag{6}$$

where $\mathbf{W}$ is the matrix for intra-slice edges and $\mathbf{A}$ is the matrix for inter-slice connections, and $L_\theta(\mathbf{X}; \mathbf{W}, \mathbf{A}) = \frac{1}{n}||\mathbf{X} - \mathbf{X}\mathbf{W} - \mathbf{Y}\mathbf{A}||_F^2$. Here n is the total sample size, and $||\cdot||_F$ is the Frobenius norm. To distinguish with the Frobenius norm, we use $||\cdot||_2$ to denote the (vector) l_2 norm.

3. IDYNO: Structure Learning from Interventional Time Series Data

We make two major improvements of the existing graph learning algorithms in the proposed algorithm, IDYNO. First, we propose a new DAG learning method and algorithm that can handle both observational and interventional time series data. Second, we propose to use a more complex nonlinear family of models to capture arbitrary distribution P_X more accurately.

3.1. Time Series Data with Interventions

First, we develop a method that can handle both interventional and observational data together, based on a similar idea from i.i.d. datasets (Brouillard et al., 2020). The main idea is to learn a DAG from interventional data by considering a separate distribution family for the intervened nodes (in the cases of perfect intervention, removing intervened nodes) in the log-likelihood objective. Specifically, in a standard SVAR model, let a binary indicator matrix $\mathbf{R}^{\mathcal{I}} = [r_{kj}^{\mathcal{I}}] \in \{0, 1\}^{K \times d}$ encode the interventional family $\mathcal{I}$, such that $r_{kj}^{\mathcal{I}} = 1$ when x_j is an intervened target in I_k and 0 otherwise. For each interventional family k, we use $\mathbf{W_{(k)}} \in \mathbb{R}^{d \times d}$ and $\mathbf{A_{(k)}} \in \mathbb{R}^{pd \times d}$ to denote the corresponding weighted adjacency matrices, and $\mathbf{W} := \{\mathbf{W_{(1)}}, ..., \mathbf{W_{(K)}}\}$, $\mathbf{A} := \{\mathbf{A_{(1)}}, ..., \mathbf{A_{(K)}}\}$ represent the collection matrices of all interventional families, with $W_{(1)}$ and $A_{(1)}$ corresponding to non-intervened data. The loss function $L_\theta(\mathbf{X}; \mathbf{W})$ in Eq (6) considers both observational and interventional data, namely:

$$\begin{aligned} L_\theta(\mathbf{X}; \mathbf{W}, \mathbf{A}) = &\sum_{k=1}^{K}\sum_{j=1}^{d} \big(L_j(\mathbf{X}; \mathbf{W_{(1)}}, \mathbf{A_{(1)}})^{1 - r_{kj}^{\mathcal{I}}} \\ &+ L_j(\mathbf{X}; \mathbf{W_{(k)}}, \mathbf{A_{(k)}})^{r_{kj}^{\mathcal{I}}}\big) \end{aligned} \tag{7}$$

where L_j denotes the loss associated with the jth component. Then, the optimization becomes:

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{A}} \frac{1}{n} &\sum_{k=1}^{K}\sum_{j=1}^{d} \Big(|| \left(\mathbf{X} - \mathbf{X}\mathbf{W_{(1)}} - \mathbf{Y}\mathbf{A_{(1)}}\right)_j ||_2^{2(1 - r_{kj}^{\mathcal{I}})} \\ &+ || \left(\mathbf{X} - \mathbf{X}\mathbf{W_{(k)}} - \mathbf{Y}\mathbf{A_{(k)}}\right)_j ||_2^{2 r_{kj}^{\mathcal{I}}} \Big) + \lambda\Omega(\theta) \\ s.t. \quad &Tr(e^{\mathbf{W}}) - d = 0 \end{aligned} \tag{8}$$

Then the final estimated graph structure would be $\mathbf{W}_{(1)}$ and $\mathbf{A}_{(1)}$, indicating the underlying graph structures under no interventions.

In the case of perfect or hard interventions, intervened nodes no longer depend on $\mathbf{W}$ or $\mathbf{A}$, and hence their loss can be removed from the objective without affecting the minimization with respect to $\mathbf{W}$ and $\mathbf{A}$.

3.2. Nonlinear Time Series Data

Nonlinear Observational Time Series Data In practice, the relationships among variables can be highly nonlinear, increasing the difficulty in modeling. To alleviate this issue, we first adapt the nonlinear relationships in the continuous constrained optimization DAG learning framework for time series data to the IDYNO framework, with a nonlinear model such as the neural network based NOTEARS method (Zheng et al., 2020). In this class of models, there may not be parameters directly representing the weighted adjacency matrices $\mathbf{W}$ and $\mathbf{A}$.

We assume that there exist functions $f_j : \mathbb{R}^d \to \mathbb{R}$ and $g_j : \mathbb{R} \to \mathbb{R}$ such that:

$$\mathbb{E}[x_j | x_{\pi_j^G}] = g_j(f_j(X), f_j^p(Y)), \quad \mathbb{E}[f_j(X)] = 0 \tag{9}$$

where $f_j(x_1, ..., x_d)$ does not depend on x_k if $x_k \notin \pi_j^G$, and f_j^p is the time-lagged function. We assume g_j follows a generalized linear model (GLM) with possible non-additive noise terms. In this work, we study the link function g_j as the sum of two terms. In this setting, we seek to learn $f = (f_1, ..., f_d)$ such that the DAG from f, $\mathbf{W}(f)$, represents the same DAG over the data X. The overall objective can be written as:

$$\min_f L_\theta(f) - \lambda\Omega(\theta) \quad s.t. \quad \mathbf{W}(f) \text{ is a DAG} \tag{10}$$

where $L_\theta(f) = \sum_{j=1}^{d} L(X_j, f_j(X), f_j^p(Y))$.

However, it is not straightforward to directly use a $\mathbf{W}$ and $\mathbf{A}$ in the parametrization of the neural network, since neural networks do not contain them, and the continuous acyclicity constraint $h(\mathbf{W})$ requires $\mathbf{W}$ explicitly. To remedy this problem, we extend the nonparametric acyclicity to time series datasets by using partial derivatives to measure the dependence of each function f_j on the qth variable.

Let $H^1(\mathbb{R}^d) \subset L_2(\mathbb{R}^d)$ be the Sobolev space of square-integrable functions whose derivatives are also square integrable. Let $f_j \in H^1(\mathbb{R}^d)$ and $\partial_q f_j$ be the partial derivative with respect to x_q. It can be shown that f_j is independent of x_q if and only if $||\partial_q f_j||_{L_2} = 0$, where $||\cdot||_{L_2}$ is the usual L_2-norm (Zheng et al., 2020). For a formal argument for this statement, please refer to Appendix A.

Then each entry of $\mathbf{W}(f) = \mathbf{W}(f_1, ..., f_d) \in \mathbb{R}^{d\times d}$ can be defined as:

$$[\mathbf{W}(f)]_{qj} := ||\partial_q f_j||_{L_2}$$

To use a neural network to approximate f_j , we define f_j, for each j, as a multi-layer perceptron (MLP) with h hidden layers and activation $\sigma : \mathbb{R} \to \mathbb{R}$, given by

$$MLP(U; \mathbf{M^{(1)}}, ..., \mathbf{M^{(h)}}) = \sigma(\mathbf{M^{(h)}}\sigma(...\mathbf{M^{(2)}}\sigma(\mathbf{M^{(1)}}U)))$$

where $\mathbf{M^{(l)}}$ is the (matrix) weight of each layer in MLP. If the qth column of the first layer weight $\mathbf{M^{(1)}}$ consists of all zeros, then $MLP(U; \mathbf{M^{(1)}}, ..., \mathbf{M^{(h)}})$ is independent of u_q, the qth component of U.

The above analysis focuses on one variable x_j at a time. Let $\theta_j = (M_j^{(1)}, ..., M_j^{(h)})$ denote parameters for the the jth MLP, and $\theta = (\theta_1, .., \theta_d)$. Let $[\mathbf{W}(\theta)]_{qj} = ||q\text{th-column}(M_j^{(1)})||_2$ and λ_a, λ_w as the regularization parameters for $\mathbf{A}$ and $\mathbf{W}$, respectively. Then the overall objective becomes:

$$\min_{\mathbf{W},\mathbf{A},\theta} \frac{1}{n}\sum_{j=1}^{d} L_j(\mathbf{X}, \mathbf{MLP}(\mathbf{X}; \theta_\mathbf{j}, \mathbf{W}, \mathbf{A}) + \lambda_\mathbf{a}||\mathbf{A}_\mathbf{j}^{(\mathbf{1})}||_{\mathbf{1},\mathbf{1}}$$
$$+ \lambda_w||W_j^{(1)}||_{1,1} \quad s.t. \quad \mathbf{W} \text{ is acyclic}$$

where $MLP(X; \theta_j, \mathbf{W}, \mathbf{A})$ produces the estimate $\hat{X}_j$. Since there are two adjacency matrices $\mathbf{W}$ and $\mathbf{A}$ to be learned, we use separate MLPs for them and concatenate the output from each MLP via a third MLP. In other words, we set $MLP(X; \theta_j, \mathbf{W}, \mathbf{A})$ as the combination of separate MLPs for $\mathbf{W}$ and $\mathbf{A}$, i.e., $MLP_1(MLP_2(X; \mathbf{A}, \theta_j^A), MLP_3(X; \mathbf{W}, \theta_j^W); \theta_j)$. $\mathbf{A^{(1)}}$ and $\mathbf{W^{(1)}}$ are first layer parameters for $\mathbf{W}$ and $\mathbf{A}$ networks.

Solving the continuous program can be done with any off-the-shelf solver. As typical in the continuous acyclic formulation, the standard augmented Lagrangian transforms the problem into a series of unconstrained objectives;

$$\min_\theta L(\theta) + \frac{\rho}{2}|h(\mathbf{W}(\theta))|^2 + \alpha h(\mathbf{W}(\theta))$$
$$+ \lambda_a||A_j^{(1)}||_{1,1} + \lambda_w||W_j^{(1)}||_{1,1} \tag{11}$$

To solve the unconstrained l_1-penalized smooth minimisation problem above, a number of possible optimizers could be used. A natural choice would be the L-BFGS-B algorithm (Byrd et al., 1995). Since Eq (11) is a nonconvex program due to the acyclicity constraint, only a stationary solution can be guaranteed.

Nonlinear Interventional Time Series Data With the above proposal nonlinear model for time series data, we propose a neural network version of IDYNO, denoted as IDYNO-nn, to handle potentially complex nonlinear interventional time series data. The main difference from the observational data is the use of interventional loss functions for each interventional family.

$$\min_{W,A,\theta} \frac{1}{n}\sum_{k=1}^{K}\sum_{j=1}^{d} L_j(\mathbf{X}, MLP(\mathbf{X}; \theta_j, \mathbf{A_{(1)}}, \mathbf{W_{(1)}}))^{1-r_{kj}^{\mathcal{I}}}$$
$$L_j(\mathbf{X}, MLP(\mathbf{X}; \theta_j, \mathbf{A_{(k)}}, \mathbf{W_{(k)}}))^{r_{kj}^{\mathcal{I}}}$$
$$+ \lambda_a||A_j^{(1)}||_{1,1} + \lambda_w||W_j^{(1)}||_{1,1} \tag{12}$$
$$s.t. \quad Tr(e^{\mathbf{W}}) - d = 0 \tag{13}$$

Eqs (12)-(13) can be solved similarly as (11), to achieve stationary point solutions.

After learning, we use a threshold on $\mathbf{W_{(1)}}$ and $\mathbf{A_{(1)}}$ to remove small coefficients and obtain the final graphs. This thresholding procedure is standard in continuous DAG learning algorithms.

3.3. Identifiability

Identifiability of the DAG structures for observational time series data has been established (Pamfil et al., 2020), where the inter-slice edges represented by $\mathbf{A}$ are identified from standard results in vector autoregressive (VAR) models, whereas the intra-slice edges $\mathbf{W}$ can be identified under two special cases: 1) the errors $\mathbf{Z}$ are non-Gaussian, as a well-known consequence of Marcinkiewicz's theorem and independent component analysis (Pamfil et al., 2020), or 2) all errors $\mathbf{Z}$ are standard Gaussian with zero mean and equal variance (Peters & Bühlmann, 2014). For linear time series interventional datasets with known targets, identifiability results have been studied in the i.i.d. case (Chen et al., 2018). Specifically, if each variable is affected by a unique set of intervened variables, the resulting model is identifiable. For time series data, under the (linear) SVAR parametric distribution assumption, one could establish the same identification results, since the reformulation with time-lagged data can be seen as equivalent to an i.i.d. formulation.

With nonlinear interventional data, the identification results above may not apply anymore. However, without further distributional assumptions, Brouillard et al. (2020) establishes identifiability of the $\mathcal{I}$-Markov equivalent class in i.i.d. data. These results can be extended to IDYNO-nn for time series data in the following fashion.

First, we define a generalized version of $\mathcal{I}$-Markov equivalence from prior work (Hauser & Bühlmann, 2012; Yang et al., 2018) for graphs within a (domain or problem-dependent) subset of DAGs rather than all DAGs:

Definition 3.1. **($(\mathcal{I}, \mathcal{D})$-Markov Equivalence Class).** Given a set of interventions $\mathcal{I}$ on a DAG $\mathcal{G}$ in a subset $\mathcal{D}$ of all possible DAGs over a set of variables, we define $(\mathcal{I}, \mathcal{D})$-MEC($\mathcal{G}$) as the subset of graphs in $\mathcal{D}$ which are $\mathcal{I}$-Markov equivalent to $\mathcal{G}$.

As such, $(\mathcal{I}, \mathcal{D})$-MEC($\mathcal{G}$) is simply the intersection of $\mathcal{D}$ with the $\mathcal{I}$-Markov equivalence class of $\mathcal{G}$, i.e. $\mathcal{D} \cap (\mathcal{I})$-MEC($\mathcal{G}$). We also define the negative score function $-\mathcal{S}^{\mathcal{I}}(\mathcal{G})$ of a model graph $\mathcal{G}$ as the expectation $\mathbb{E}_{\mathbf{X}|\{p^{(k)}\},\mathcal{G}^\star}[L_{\text{reg}}(\mathbf{X})]$ – where L_{reg} is the regularized loss being minimized in Eq. (12) with the losses L_j being negative log-likelihoods – over data $\mathbf{X}$ generated from the true graph $\mathcal{G}^\star$ with interventional distributions $p^{(k)}$ for each $I_k \in \mathcal{I}$.[2] With these definitions, the following result holds: (Please see the proof and related discussion in Appendix B.)

Theorem 3.2. *For a graph $\hat{\mathcal{G}} \in \mathcal{D}$, where $\mathcal{D} \subset DAG$, if $\hat{\mathcal{G}} \in argmax_{\mathcal{G} \in DAG} \mathcal{S}_{\mathcal{I}}(\mathcal{G})$, and furthermore if (i) the density model, i.e. functions $\{g_j, f_j, f_j^p\}$ along with noise terms, has sufficient capacity to exactly represent the ground truth distribution P_X, (ii) a given set of interventions $\mathcal{I}$ satisfies $\mathcal{I}$-faithfulness for the true graph and distributions $(\mathcal{G}^\star, P_X)$, (iii) the density models are strictly positive, and (iv) the ground truth densities $p^{(k)}(\mathbf{X})$ have finite differential entropy, then $\hat{\mathcal{G}}$ is $(\mathcal{I}, \mathcal{D})$-Markov equivalent to $\mathcal{G}^\star$, for small enough λ_a, λ_w.*

Setting $\mathcal{D}$ to be the subset $\mathcal{D}_s$ of DAGs which correspond to stationary dynamics with constant-in-time inter-slice and intra-slice conditional distributions (see Appendix B for details), Theorem 3.2 takes the following form:

Corollary 3.3. *For a graph $\hat{\mathcal{G}} \in \mathcal{D}_s$ and given the assumptions mentioned in Theorem 3.2, $\hat{\mathcal{G}}$ is $(\mathcal{I}, \mathcal{D}_s)$-Markov equivalent to $\mathcal{G}^\star$, for small enough λ_a, λ_w.*

Here, restricting to DAGs in $\mathcal{D}_s$ while allowing interventions in $\mathcal{I}$ to change over time reduces the size of the equivalence class $(\mathcal{I}, \mathcal{D}_s)$-MEC$(\mathcal{G}^\star)$.

4. Empirical Evaluation

We compare the proposed IDYNO methods, including both linear/nonlinear and soft/hard regimes, with baseline methods for time series interventional datasets to show the superior performance of the proposed methods. We first compare different methods on synthetic datasets and then apply them to control tasks to discover the underlying structures. Since there are no known structure learning algorithms for interventional dynamic data, we compare our method to implementations of various state-of-the-art observational baselines, including DYNOTEARS (Pamfil et al., 2020), standard vector autoregressive models (VAR) (Johansen, 1991), tsGFCI (Malinsky & Spirtes, 2019), and i.i.d. differential interventional method DCDI (Brouillard et al., 2020). We need to transform time series data for DCDI's usage. We use the following 2-step procedure: 1) fit VAR to compute the residual $\mathbf{e} = \mathbf{Z}(\mathbf{I} - \mathbf{W})^{-1}$ and $\mathbf{B} = \mathbf{A}(\mathbf{I} - \mathbf{W})^{-1}$ and 2) use DCDI to learn $\mathbf{W}$ over the residual data $\mathbf{e} = \mathbf{e}\mathbf{W} + \mathbf{Z}$. This two-step approach can learn both the $\mathbf{W}$ and $\mathbf{A} = \mathbf{B}(\mathbf{I} - \mathbf{W})$, although errors can propagate via estimation of earlier steps (Pamfil et al., 2020). Moreover, since tsGFCI returns a partial ancestral graph (PAG), an edge is counted as correct in our experiment if it has the correct arrow in the desired direction, regardless of the other direction. This procedure favors tsGFCI even more than the setting in DYNOTEARS (Pamfil et al., 2020) and hence may overstate the performance of tsGFCI. We use an existing implementation of tsGFCI from Tetrad [3] with default parameter settings.

All experiments are done in Python on a machine with 3.7GHz CPU and 16GB memory.

4.1. Synthetic Dynamic Datasets

We first evaluate different approaches on synthetic time series data. We simulate the data according to the SEM from Eq (3), mostly following the setup and code from Pamfil et al. (2020) for ease of comparison. The generating process consists of 3 steps:

1. Generating weighted graphs in the form $\mathbf{W}$ and $\mathbf{A}$.
2. Generating data matrices $\mathbf{X}$ and corresponding $\mathbf{Y}$ per $\mathbf{W}$ and $\mathbf{A}$.
3. Generating interventional data with random targets and different distributions.

We repeat each experiment 5 times and compare the struc-

[2] We omit the constraint, Eq. (13), since the theoretical analysis applies to the space of DAGs, and does not consider cyclic graphs.

[3] https://github.com/cmu-phil/tetrad

tural Hamming distance (SHD) between the learned graph and the estimated graph (the lower, the better). We report the mean and the standard error for each case. We generate linear datasets first and then more complex datasets under more difficult settings.

For the hyperparameter values in DYNOTEARS and our proposed method, we use a separate validation dataset to choose the best performing hyperparameters for each method per SHD. We search for the best value of each of 4 parameters sequentially, including λ_a, λ_w, the threshold to obtain final $\mathbf{W_{(1)}}$ and $\mathbf{A_{(1)}}$, and the hidden neuron size. For λ_a and λ_w, we search over a value range of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. Graph threshold search range is set to be $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3\}$, and neuron size range is searched over $\{5, 10, 30, 50\}$. For other baseline methods, we use the default parameter settings.

Table 1. SHD Results for Synthetic Linear Datasets

Dataset	DYNOTEARS	IDYNO
Observational	2.0± 0.0	2.0± 0.0
Interventional	32± 0.4	**19± 0.3**

Linear Synthetic Interventional Datasets. In the linear setting, we use the following steps and hyper-parameters to generate the data.

- For step 1), we use the Erdos-Renyi model to generate intra-slice graph $\mathbf{W}$ with degree of 3 for a total $d = 10$ nodes, with its weights sampled uniformly at random from $\mathcal{U}([-2.0, -0.5] \cup [0.5, 2.0])$. We use the same Erdos-Renyi model to generate inter-slice graph $\mathbf{A}$ with degree of 3, with its weights sampled uniformly at random from $\mathcal{U}([-0.5, -0.3] \cup [0.3, 0.5])$.
- For step 2), we focus on data with first autoregressive order, i.e., $p = 1$, where data only depends on the previous time slice. We generate 5 sequences with 500 time slices each with standard Gaussian noises. We estimate one graph from each sequence and report the average SHD along with its standard error.
- For step 3), to generate interventional data, we flip a fair coin to decide whether to intervene at any time slice, in which case we sample one node uniformly to be the intervened node. Here we deploy a perfect intervention, setting the choosing the values of intervened nodes from $\{0.25, 0.5, 0.75, 1\}$ randomly. The number of intervention families K is therefore 2. We also compare methods for purely observational data (Obs).

As shown in Table 1, the proposed IDYNO method achieves similar accuracy as DYNOTEARS on the observational dataset, validating it can handle observational data. For the interventional dataset, IDYNO significantly outperforms DYNOTEARS, achieving almost half of the SHD.

Nonlinear Interventional Datasets. Next we test IDYNO-nn, along with all other baselines, on nonlinear synthetic dynamic datasets. The data generating process is similar to the linear setting, except the underlying function between a node and its parents becomes a two-layer MLP with a sigmoid activation function. In the MLP setting, the weights of the MLPs are sampled uniformly from $\mathcal{U}([-2.0, -0.5] \cup [0.5, 2.0])$, and then weights in the first layer are updated by the parental weights from $\mathbf{W}$ and $\mathbf{A}$ structures. We also test the effect of varying sizes of node d, from 5 to 20. The interventional data is generated again with the same hard intervention regime as described previously.

As shown in Figure 2, IDYNO and DYNOTEARS suffer due to their linearity assumptions; IDYNO-nn is the most accurate, performing much better than both linear models.

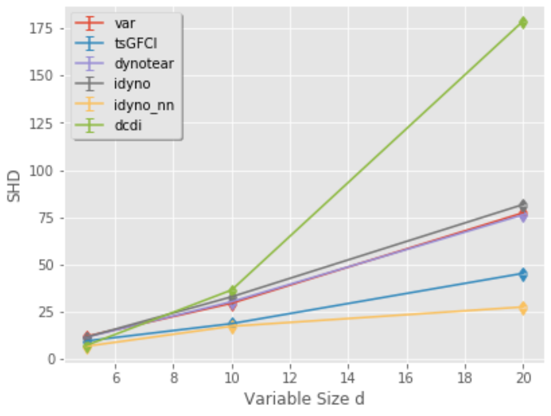

Figure 2. SHD Results for Nonlinear Hard-Intervention Datasets.

Nonlinear Soft-Interventional Datasets. Lastly, we also test the methods under soft interventional regimes. We follow the same nonlinear dynamic data generating process as above, with two exceptions: 1) we increase the number of potential interventions at each time slice, and 2) we sample intervened nodes' values from another distribution. We assume every node has a probability of 0.1 to be intervened upon at each time slice (hence up to d instead of 1), and if a node is chosen, the soft intervention comes from a different 2-layer MLP, depending on the values of its parent nodes per its graph. This soft intervention distribution can be very different from the observation distribution.

As shown by the results in Figure 3, we compare the soft version of the IDYNO-nn, or IDYNO-nn-soft, with all other methods, for $d = \{5, 10, 20\}$. The results confirm that IDYNO-nn is consistently better than DYNOTEARS and

IDYNO. IDYNO-nn-soft achieves much better performance than IDYNO-nn, showing that the soft intervention loss function might be necessary in these settings. tsGFCI ranks second, with mean SHD 42.0 as opposed to IDYNO-nn-soft's 39.0 at $d = 20$.

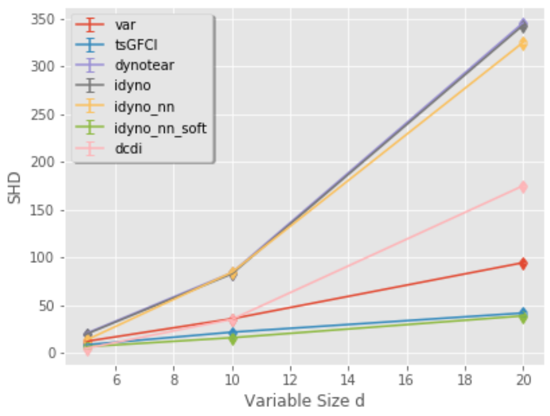

Figure 3. SHD Results for Nonlinear Soft-Intervention Datasets.

4.2. Network Administration Experiments

Problem Setting. We consider a network administration example from the factored MDP literature where a cluster of computers are connected together in some underlying network topology such that failures of these computers propagate probabilistically through network connections (Guestrin et al., 2001). The administrator can prevent failure propagation by fixing computers that have failed. The problem can be modeled as a DBN where there is a variable X_i for every computer i in the network. X_i at any epoch depends on its state at the previous period, its parent computer states in the underlying topology (representing physical dependencies in the network) as well as the binary repair action performed for the computer. At most one computer can be repaired at any epoch due to resource constraints. In the original version of the problem, variables X_i are binary such that 0 and 1 represent failure and operational states respectively. We consider the continuous state version of the problem where the states of each computer are values between 0 and 1, such that a lower value indicates a poorer condition (Hauskrecht, 2004; Kveton et al., 2006).

Data Generation. We consider the ring network topology involving d computers, similar to prior work, where there are only edges from computer i to computer $i + 1$, $i = 1, \cdots, d - 1$ as well as an edge from computer d to computer 1. We note that our data generation applies to any network topology in general. Our approach for the DBN sampling follows prior work (Hauskrecht, 2004; Kveton et al., 2006) as follows. Action A_i is 1 if the computer

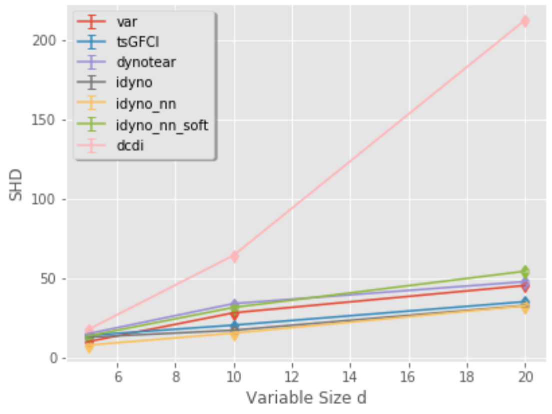

Figure 4. SHD Results for Network Administration Experiments

indexed i is fixed at any epoch, otherwise it is 0. For the experiments, we consider a realistic policy where only the worst state computer from the previous epoch is fixed. The transition model captures propagation of network failures and is encoded locally by beta distributions with parameters a and b. Specifically, when $A_i = 1$, X_i is generated from a Beta distribution with parameters $a = 20$, $b = 2$. In contrast, when $A_i = 0$, X_i depends on variables from the previous epoch with the following Beta distribution parameters:

$$a = 2 + 13x'_i - \left(2x'_i \sum_{j \in Pa(X_i)} x'_j\right),$$

$$b = 10 - 2x'_i - \left(2x'_i \sum_{j \in Pa(X_i)} x'_j\right),$$

where x'_i is the state of computer i from the previous epoch, x'_j is the state of computer j from the previous epoch, and $Pa(X_i)$ are the parents of X_i in the network topology. We also enforce non-negativity constraints on the beta parameters above. In this fashion, failures are only propagated when a computer is not fixed.

Results. Figure 4 compares results for structure learning of the DBN for a ring network topology with a varying number of computers (d). IDYNO-nn performs best on this task since it is a hard-intervention dataset; it is better than DYNOTEARS and IDYNO. tsGFCI's performance is closely behind (for example, mean SHD 35.0 against 32.4 at $d = 20$), possibly because there is only a limited amount of interventional data.

4.3. Continuous Control Experiments

For these experiments, we apply the proposed methods to a reinforcement learning environment, where data is provided by RL agent actions. We work in the offline RL setting

Table 2. Continuous Lunar Lander Experiment Results Of Various Algorithms.

Algorithm	VAR	DCDI	tsGFCI	DYNOTEAR	IDYNO	IDYNO-nn	IDYNO-nn-soft
SHD	46.4 ± 0.4	34.8 ± 0.1	27.6 ± 0.1	32.6 ± 0.2	32.2 ± 0.1	31.2 ± 0.1	$\mathbf{27.4 \pm 0.1}$

where data trajectories are generated in advance by multiple unknown policies (intervention families k) and are used for structure learning without online acquisition of more data (Sutton & Barto, 2018; Levine et al., 2020).

Problem Setting. We use the continuous version of the Lunar Lander environment in OpenAI Gym (Brockman et al., 2016), as DYNOTEARS handles only continuous variables. Lunar Lander is a mini-game where an agent tries to land a lunar lander safely in a desired location without crashing. Let us denote the state and action space dimensionalities as $(d_s, d_a) = (6, 2)$. States η_1 through η_6 are the lander's horizontal position, horizontal velocity, vertical position, vertical velocity, angular orientation, and angular velocity, respectively. Actions β_1 and β_2 are the forces from the main engine and orientation engine. We omit two binary state variables indicating whether the lander legs have landed.

State-action dynamics take the following form:[4]

$$\Delta\eta_1^{(t)} = \tau\eta_2^{(t)} \quad \Delta\eta_2^{(t)} = \tau\beta_2^{(t)}/M \tag{14}$$
$$\Delta\eta_3^{(t)} = \tau\eta_4^{(t)} \quad \Delta\eta_4^{(t)} = \tau(\beta_1^{(t)}/M - g) \tag{15}$$
$$\Delta\eta_5^{(t)} = \tau\eta_6^{(t)} \quad \Delta\eta_6^{(t)} = \tau \cdot f(\beta_1^{(t)}, \beta_2^{(t)}), \tag{16}$$

where $\Delta\eta_i^{(t)} := \eta_i^{(t+1)} - \eta_i^{(t)}$ and $f(\beta_1, \beta_2)$ is a function which evaluates angular torque on the lander.

Note that the actions' parents (which determine the policy in RL) are missing from the above dynamics, which changes depending on the different learned parents. NN-based agents typically take all states as input in policy networks, hence all states are the parents of each action. From the above equation, we can construct the ground truth DBN structure governing the MDP at a fixed policy in this setting. For completeness, we show the resulting ground truth graph in Appendix C.

Data Generation. We use one of the state-of-the-art learners, Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), as the policy learning agents. We use two DDPG agents: a randomly initialized DDPG agent (intervention family $k = 1$), and a fully trained agent with $2e5$ total time steps ($k = 2$). We then save 10 different episodes performed by both agents with 1000 time steps as the interventional data. If the lander touched down early, we then remove redundant data after 5 maximal reward values ($R = 1$). To use the time-lagged transformation, we combine one sequence from each agent, hence with 2 intervention families, to form the training data. For more details on the training and experimental setting, please refer to Appendix C.

[4] M, g, and τ are fixed parameters.

Results. We show the SHD results in Table 2. Variants of IDYNO methods perform competitive and improve upon the DYNOTEAR results. In addition, IDYNO-nn-soft performs best here, with tsGFCI slightly behind with a potentially favorable setting. Other methods are behind in term of SHD performance.

5. Conclusion

In this paper, we have proposed IDYNO – an approach for learning the DAG structure of a dynamic Bayesian network from (dynamic) time series data. IDYNO can handle linear and nonlinear dependencies, observational and interventional data, and perfect and imperfect interventions. To the best of our knowledge, IDYNO is the first such proposed algorithm. We show that under standard assumptions, the underlying graph can be identified at least up to the interventional Markov equivalent class. On various synthetic datasets as well as on a continuous control task, we show that IDYNO consistently outperforms its purely observational counterpart, exhibiting great potential to handle interventional datasets. We have considered passive interventional data in this paper but future work could potentially explore active intervention strategies as well as connections to online RL, based on the proposed approach.

Acknowledgements

We thank the anonymous reviewers for the thoughtful comments. YY would like to acknowledge support by the National Science Foundation under award DMS 1753031 and the AFOSR grant FA9550-22-1-0197.

References

Bhattacharjya, D., Subramanian, D., and Gao, T. Proximal graphical event models. In *Advances in Neural Information Processing Systems*, pp. 8136–8145, 2018.

Boutilier, C., Dearden, R., and Goldszmidt, M. Exploiting structure in policy construction. In *Proceedings of the*

International Joint Conference on Artificial Intelligence, pp. 1104–1111, 1995.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, pp. 21865–21877, 2020.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Chen, C., Ren, M., Zhang, M., and Zhang, D. A two-stage penalized least squares method for constructing large systems of structural equations. *Journal of Machine Learning Research*, 19(1):40–73, 2018.

Chickering, D. M. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3: 507–554, 2002.

Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294–321, 2012.

Dahlhaus, R. Graphical interaction models for multivariate time series. *Metrika*, 51:157–172, 2000.

Dean, T. and Kanazawa, K. A model for reasoning about persistence and causation. *Computational Intelligence*, 5: 142–150, 1989.

Demiralp, S. and Hoover, K. D. Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, 65:745–767, 2003.

Didelez, V. Graphical models for marked point processes based on local independence. *Journal of the Royal Statistical Society, Ser. B*, 70(1):245–264, 2008.

Eberhardt, F. Almost optimal intervention sets for causal discovery. In *Conference on Uncertainty in Artificial Intelligence*, pp. 161–168, 2012.

Eberhardt, F., Glymour, C., and Scheines, R. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables. In *Conference on Uncertainty in Artificial Intelligence*, pp. 178–184, 2005.

Eichler, M. *Graphical Models in Time Series Analysis*. PhD thesis, University of Heidelberg, Germany, 1999.

Guestrin, C., Koller, D., and Parr, R. Max-norm projections for factored MDPs. In *Conference on Uncertainty in Artificial Intelligence*, pp. 673–682, 2001.

Gunawardana, A., Meek, C., and Xu, P. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, pp. 1962–1970, 2011.

Hauser, A. and Bühlmann, P. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(1):2409–2464, 2012.

Hauskrecht, Milos, . K. B. Linear program approximations for factored continuous-state markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 895–902, 2004.

Heckerman, D., Geiger, D., and Chickering, D. M. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

Jaber, A., Kocaoglu, M., Shanmugam, K., and Bareinboim, E. Causal discovery from soft interventions with unknown targets: Characterization and learning. In *Advances in Neural Information Processing Systems*, pp. 9551–9561, 2020.

Johansen, S. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica: journal of the Econometric Society*, pp. 1551–1580, 1991.

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

Ke, N. R., Bilaniuk, O., Goyal, A., Bauer, S., Larochelle, H., Schölkopf, B., Mozer, M. C., Pal, C., and Bengio, Y. Learning neural causal models from unknown interventions, 2020.

Kilian, L. Structural vector autoregressions. In *Handbook of Research Methods and Applications in Empirical Macroeconomics*. Edward Elgar Publishing, 2013.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Kveton, B., Hauskrecht, M., and Guestrin, C. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, pp. 153–201, 2006.

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

Lanne, M., Meitz, M., and Saikkonen, P. Identification and estimation of non-Gaussian structural vector autoregressions. *Journal of Econometrics*, 196(2):288–304, 2017.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv: arXiv:2005.01643*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Linzner, D., Schmidt, M., and Koeppl, H. Scalable structure learning of continuous-time Bayesian networks from incomplete data. *Advances in Neural Information Processing Systems*, 32:3746–3756, 2019.

Malinsky, D. and Spirtes, P. Learning the structure of a nonstationary vector autoregression. In *International Conference on Artificial Intelligence and Statistics*, pp. 2986–2994, 2019.

Meek, C. Toward learning graphical and causal process models. In *Proceedings of Uncertainty in Artificial Intelligence Workshop Causal Inference: Learning and Prediction*, pp. 43–48, 2014.

Meng, Z., Han, S., Liu, P., and Tong, Y. Improving speech related facial action unit recognition by audiovisual information fusion. *IEEE Transactions on Cybernetics*, 49(9): 3293–3306, 2018.

Murphy, K. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California Berkeley, USA, 2002.

Ng, I., Fang, Z., Zhu, S., Chen, Z., and Wang, J. Masked gradient-based causal structure learning. *arXiv preprint arXiv:1910.08527*, 2019.

Nodelman, U., Shelton, C. R., and Koller, D. Continuous time Bayesian networks. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pp. 378–378, 2002.

Oates, C. J., Smith, J. Q., and Mukherjee, S. Estimating causal structure using conditional dag models. *Journal of Machine Learning Research*, 17(1):1880–1903, 2016.

Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Georgatzis, K., Beaumont, P., and Aragam, B. DYNOTEARS: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*, pp. 1595–1605, 2020.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

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

Peters, J. and Bühlmann, P. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014.

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

Rajapakse, J. C. and Zhou, J. Learning effective brain connectivity with dynamic Bayesian networks. *Neuroimage*, 37(3):749–760, 2007.

Reale, M. and Wilson, G. T. Identification of vector ar models with recursive structural errors using conditional independence graphs. *Statistical Methods and Applications*, 10(1-3):49–65, 2001.

Reale, M. and Wilson, G. T. The sampling properties of conditional independence graphs for structural vector autoregressions. *Biometrika*, 89(2):457–461, 2002.

Shanmugam, K., Kocaoglu, M., Dimakis, A. G., and Vishwanath, S. Learning causal graphs with small interventions. In *Advances in Neural Information Processing Systems*, pp. 3195–3203, 2015.

Spirtes, P., Glymour, C., and Scheines, R. *Causality, Prediction, and Search*. MIT Press, Cambridge, MA, USA, 2nd edition, 2001.

Squires, C., Wang, Y., and Uhler, C. Permutation-based causal structure learning with unknown intervention targets. In *Uncertainty in Artificial Intelligence*, pp. 1039–1048, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Swanson, N. R. and Granger, C. W. Impulse response functions based on a causal approach to residual orthogonalization in vector autoregressions. *Journal of the American Statistical Association*, 92(437):357–367, 1997.

Tank, A., Fox, E. B., and Shojaie, A. Identifiability and estimation of structural vector autoregressive models for subsampled and mixed-frequency time series. *Biometrika*, 106(2):433–452, 2019.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

Yang, K., Katcoff, A., and Uhler, C. Characterizing and learning equivalence classes of causal DAGs under interventions. In *International Conference on Machine Learning*, pp. 5541–5550, 2018.

Yu, Y., Chen, J., Gao, T., and Yu, M. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, pp. 7154–7163, 2019.

Zhang, A., Lipton, Z. C., Pineda, L., Azizzadenesheli, K., Anandkumar, A., Itti, L., Pineau, J., and Furlanello, T. Learning causal state representations of partially observable environments. *arXiv preprint arXiv:1906.10437*, 2019.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, pp. 9472–9483, 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P. K., and Xing, E. P. Learning sparse nonparametric DAGs. In *International Conference on Artificial Intelligence and Statistics*, pp. 3414–3425, 2020.

A. Proof on "f_j is independent of x_q iff $\left|\left|\frac{\partial f_j}{\partial x_q}\right|\right|_{L^2(\mathbb{R})} = 0$"

As a simple illustration of the argument, we take $q = 1$ without loss of generality here. For the "if" side: when $\left|\left|\frac{\partial f_j}{\partial x_1}\right|\right|_{L^2(\mathbb{R})} = 0$, for any $a, b \in \mathbb{R}$ of x_1, we have $|f_j(b, x_2 \cdots, x_d) - f_j(a, x_2 \cdots, x_d)|^2 = \left|\int_a^b \frac{\partial f_j}{\partial x_1} dx_1\right|^2 \leq (b - a)\left|\left|\frac{\partial f_j}{\partial x_1}\right|\right|^2_{L^2(\mathbb{R})} = 0$ and hence the value of f_j is independent of x_1. Here, the inequality was guaranteed by the Cauchy-Schwarz inequality. For the "only if" side: if $f_j(x_1, x_2, \cdots, x_d)$ is independent of x_1, let a test function $\phi(x_1) \in C_c^\infty(a, b)$ (an infinitely differentiable functions with compact support in (a, b)), then $\int_a^b f_j(x_1, x_2, \cdots, x_d)\frac{\partial \phi}{\partial x_1} dx_1 = f_j(x_1, x_2, \cdots, x_d)\int_a^b \frac{\partial \phi}{\partial x_1} dx_1 = f_j(x_1, x_2, \cdots, x_d)(\phi(b) - \phi(a)) = 0$. Hence we have $\frac{\partial f_j}{\partial x_1} = 0$ in the distributional sense, and the L^2 norm of $\frac{\partial f_j}{\partial x_1} = 0$.

B. Identifiability

In this section we clarify the quantities and notation used in Section 3.3 to establish conditions for identifiability of the "unrolled" temporally extended DAG which includes all variables at all timesteps. We define $\mathbf{X} \in \mathbb{R}^{n \times d}$ (where $n = T - p + 1$) – as in the main text – as the matrix whose elements form the set of random variables in a DAG $\mathcal{G}^\star = (V, E)$, where $V = \{x_{t,i}\}$ for $i \in \{1, ..., d\}$ and $t \in \{1, ..., n\}$ is the set of indices for d variables across n timesteps, and $(x_{t,i}, x_{t',j}) \in E$ if $x_{t,i}$ is an element of the parent nodes $\pi^{\mathcal{G}^\star}_{x_{t',j}}$ of $x_{t',j}$. Due to the acyclicity of intra-slice connections and the forward-in-time direction of inter-slice connections, $\mathcal{G}^\star$ is directed and acyclic.

We assume that the inter-slice and intra-slice edges in $\mathcal{G}^\star$ are constant in time, in the sense that (i) inter-slice edges connecting a given pair of variables $(x_{t,i}, x_{t',j})$ with a given time lag $p = t' - t$ are either elements of E for all $p \leq t_2 \leq n$ or for no t_2, and (ii) intra-slice edges $(x_{t,i}, x_{t,j})$ connecting a pair of variables at time t are either elements of E for all t or for no t. Furthermore, we assume that the distribution P_X over variables in this graph is invariant across time. That is, we restrict the conditional distribution $p_j(x_{t,j}|\pi^G_{x_{t,j}})$ for any x_j to be independent of the time index t. We will denote the subset of all DAGs that can be partitioned in this way into a directed sequence of a repeated subgraph as $\mathcal{D}_s$, in reference to the fact that repetition of the same conditional distributions and edges over time corresponds to stationary or fixed dynamics.

We consider a family of interventions $\mathcal{I}$ on the temporally extended graph $\mathcal{G}^\star$. An intervention $I_k \in \mathcal{I}$ need not modify conditional distributions in the same way at all times, but may modify conditional distributions for any subset of edges $(x_{t,i}, x_{t',j}) \in E$. (Our algorithm IDYNO assumes a smaller subset of interventions which are constant in time, and can thus be viewed as a special choice of $\mathcal{I}$.)

With these definitions, along with those given in Section 3.3, our dynamical graphical setting reduces to the non-dynamical graphical setting of Brouillard et al. (2020) and we can apply their Theorem 1 to the DAG $\mathcal{G}^\star$ as follows.

Proof of Theorem 1. We refer the reader to Brouillard et al. (2020) for precise statements of assumptions (i)-(iv) as stated in Theorem 3.2. When these assumptions ho9ld, Theorem 1 of Brouillard et al. (2020) applies. Thus, as long as λ_a, λ_w are sufficiently close to zero (such that the score $\mathcal{S}_\mathcal{I}(\mathcal{G})$ is equivalent to the log-likelihood score in Eq. (8) of Brouillard et al. (2020) in the $\lambda \to 0$ limit), $\hat{\mathcal{G}}$ is $\mathcal{I}$-Markov equivalent to $\mathcal{G}^\star$. Since furthermore $\hat{\mathcal{G}} \in \mathcal{D}$, then by Definition 3.1, $\hat{\mathcal{G}}$ is $(\mathcal{I}, \mathcal{D})$-Markov equivalent to $\mathcal{G}^\star$.

C. Continuous Control Examples: Data Sampling from Reinforcement Learning Agent

For the continuous control task, the data are obtained via reinforcement learning agents taking actions in the simulated RL environment. We save the action taken at each time step, along with current states, and hence the saved data can be seen as standard offline RL data. In this experiments, different intervention families in the data correspond to different RL policies from RL agents.

Specifically, we use the state of the art Deep Deterministic Policy Gradient (DDPG) as the learning agent (Lillicrap et al., 2016). DDPG is an off-policy reinforcement learning algorithm specifically designed for environments with continuous action spaces. It interleaves estimation of Q-function $Q^*(s, a)$ and solving the optimal action $a^* = \arg\max_a Q^*(s, a)$. In our experiments, we adopt MLP for policy representation and utilize the package stable baselines 3 [5] to train our DDPG

[5] https://stable-baselines3.readthedocs.io/en/master/modules/ddpg.html

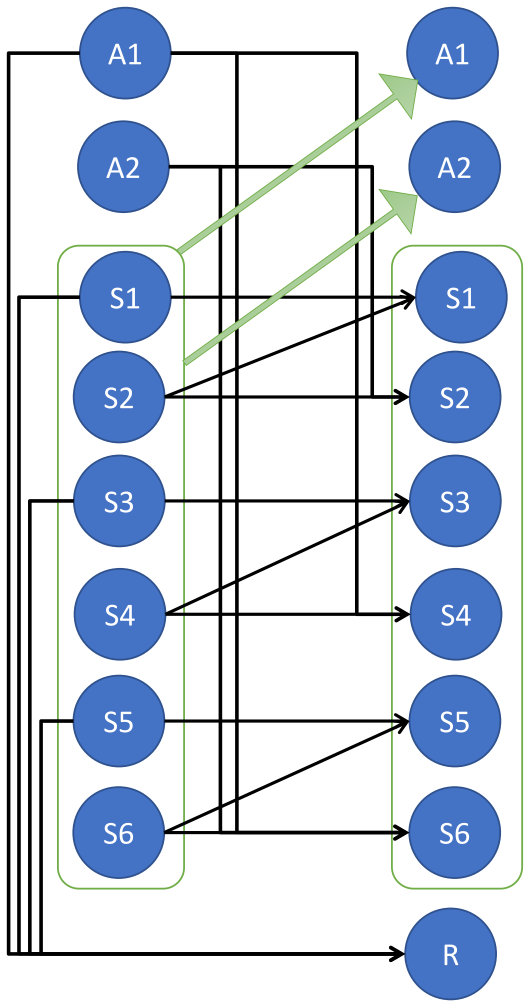

Figure 5. Ground Truth Graph for the Lunar Lander Experiment, between two consecutive time slices. We omit time index for clarity.

agent with the default hyperparameter settings.

We show the underlying graph structure as discussed in Section 4.3 in Figure 5, across two different time slices. We omit the time for clarity. A1 and A2 represent two action nodes, and there are also 6 state nodes, labelled as S1 to S6, along with one reward node R. The green arrows indicate that all state nodes are the parents of action A1 and A2.