DYNOTEARS: Structure Learning from Time-Series Data

Roxana Pamfil[1*], **Nisara Sriwattanaworachai**[1*], **Shaan Desai**[1†], **Philip Pilgerstorfer**[1], **Paul Beaumont**[1], **Konstantinos Georgatzis**[1], **Bryon Aragam**[2]
[1]QuantumBlack, a McKinsey company
causal@quantumblack.com
[2]University of Chicago
bryon@chicagobooth.edu

[*] *These authors contributed equally.*

Abstract

We revisit the structure learning problem for dynamic Bayesian networks and propose a method that simultaneously estimates contemporaneous (*intra-slice*) and time-lagged (*inter-slice*) relationships between variables in a time-series. Our approach is score-based, and revolves around minimizing a penalized loss subject to an acyclicity constraint. To solve this problem, we leverage a recent algebraic result characterizing the acyclicity constraint as a smooth equality constraint. The resulting algorithm, which we call DYNOTEARS, outperforms other methods on simulated data, especially in high-dimensions as the number of variables increases. We also apply this algorithm on real datasets from two different domains, finance and molecular biology, and analyze the resulting output. Compared to state-of-the-art methods for learning dynamic Bayesian networks, our method is both scalable and accurate on real data. The simple formulation and competitive performance of our method make it suitable for a variety of problems where one seeks to learn connections between variables across time.

1 Introduction

Graphical models are a popular approach to understanding large datasets, and provide convenient, interpretable output that is needed in today's high stakes applications of machine learning and artificial intelligence. In particular, with the growing need for interpretable models and causal insights about an underlying process, directed acyclic graphs (DAGs) have shown promise in many applications. The edges in a DAG provide users with important clues about the relationship between variables in a system. When these edges are not known based on prior knowledge, it is necessary to resort to structure learning, namely, the problem of learning the edges in a graphical model from data. Broadly speaking, structure learning can be divided into static (i.e. equilibrium) and dynamic models, the latter of which explicitly model temporal dependencies. Static models make sense for independent and identically distributed data. Many applications, however, exhibit strong temporal fluctuations that we are interested in modeling explicitly. The problem of learning graphical structures from temporal data collected from dynamic systems has received significant attention from the machine learning (Koller and Friedman, 2009), econometrics (Lütkepohl, 2005), and neuroscience (Rajapakse and Zhou, 2007) communities.

In this paper, we revisit the problem of learning *dynamic Bayesian networks* (DBNs) (Dean and Kanazawa, 1989; Murphy, 2002) from data. DBNs have been used successfully in a variety of domains such as clinical disease prognosis (Van Gerven et al., 2008; Zandonà et al., 2019), gene regulatory network (Linzner et al., 2019), facial and speech recognition (Meng et al., 2019; Nefian et al., 2002), neuroscience (Rajapakse and Zhou, 2007), among others. DBNs are the standard approach to modeling discrete-time temporal dynamics in directed graphical models. In econometrics, they are also known as *structural vector autoregressive* (SVAR) models (Demiralp and Hoover, 2003; Swanson and Granger, 1997).

We propose a simple, score-based approach for learning these models that scales gracefully to high-dimensional datasets. To accomplish this, we cast the problem as an optimization problem (i.e. score-based learning), and use standard second-order optimization schemes to solve the resulting program. Our approach is based

[†] Contributed during an internship at QuantumBlack.

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

on the recent algebraic characterization of acyclicity in directed graphs from Zheng et al. (2018), which makes the formulation simple and amenable to different modeling choices.

Contributions The main contributions of this paper are the following:

- We develop a score-based approach to learning DBNs and use standard optimization routines to optimize the resulting program. The resulting method, which we call DYNOTEARS, can be used to learn time series of arbitrary order, without any implicit assumptions on the underlying graph topologies such as bounded in-degree or treewidth.
- We validate our approach with extensive simulation experiments, exhibiting the accuracy of our approach in learning both intra-slice and inter-slice relationships in dynamic models.
- We apply our method to two real datasets: A financial dataset consisting of daily stock returns ($d = 97$) and the DREAM4 dataset ($d = 100$) (Marbach et al., 2009). These examples illustrate the importance of modeling both temporal trends as well as steady state relationships and achieve competitive accuracy among other DBN methods.

The resulting method simultaneously achieves three important goals: 1) Accuracy on high-dimensional data with $d > n$, which allows for application to real world data 2) Robustness to complex graph topologies, and 3) A simple, plug-n-play algorithm for learning based on black-box optimization.

Related work There are many methods for learning DBNs in the literature. Some approaches ignore contemporaneous dependencies and recover only time-lagged relationships (Haufe et al., 2010; Song et al., 2009). Others learn both types of relationships independently (Haufe et al., 2010; Song et al., 2009). Many methods follow a two-step approach of first learning inter-slice weights and then estimating intra-slice weights from the residuals from the first step (Chen and Chihying, 2007; Hyvärinen et al., 2010; Moneta et al., 2011). There are also hybrid algorithms that combine conditional-independence tests and local search to improve the score (Malinsky and Spirtes, 2018, 2019). While all of these methods can achieve good structure recovery on small graphs, they suffer from the curse of dimensionality. More discussion and comparison can be found in Section 2.3. There is also an extensive literature on learning SVAR models in the econometrics and statistics literature (Demiralp and Hoover, 2003; Lanne et al., 2017; Reale and Wilson, 2001, 2002; Swanson and Granger, 1997; Tank et al., 2019).

Since algorithms for learning DBNs typically rely internally on calling methods for learning static BNs, it is worth briefly reviewing this here. These methods can be classified into *constraint-based* methods and *score-based* methods, as well as hybrid methods that combine these two approaches. Constraint-based methods use conditional independence tests to recover the Markov equivalence class of DAGs under the assumption of faithfulness (e.g. Colombo et al., 2012; Spirtes and Glymour, 1991). While this yields fast algorithms, these methods are sensitive to the underlying graph structure and suffer from error propagation (Spirtes, 2010). Score-based methods, on the other hand, use a score function to find the best DAG that fits the given data (e.g. Heckerman et al., 1995; Chickering, 2002; Bouckaert, 1993). Examples of scores include the BIC, BDe, and BDeu (Spirtes et al., 2000). Score-based methods are computationally expensive due to the acyclicity constraint and the vast number of DAGs to search over (Robinson, 1977). The recent work of Zheng et al. (2018) expresses the acyclicity of a DAG by a smooth equality constraint, which makes it possible to formulate structure learning as a smooth minimization problem subject to this equality constraint.

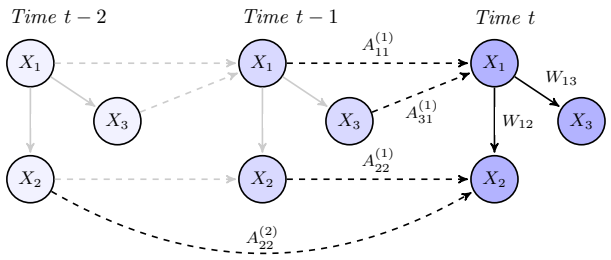


Figure 1: Illustration of *intra-slice* (solid lines) and *inter-slice* (dashed lines) dependencies in a DBN with $d = 3$ nodes and autoregression order $p = 2$. For clarity, we display edges that do not influence the variables at time t in a lighter shade.

2 Dynamic Structure Learning

2.1 Formulation

Consider M independent realizations of a stationary time series, with the mth time series given by $\{\mathbf{x}_{m,t}\}_{t\in\{0,\dots,T\}}$ for $\mathbf{x}_{m,t} \in \mathbb{R}^d$, where d represents the number of variables in the dataset. We assume that variables influence each other in both a contemporaneous and a time-lagged manner, as illustrated in Figure 1. We call these *intra-slice* and *inter-slice* dependencies, respectively.

We model the data using the following standard SVAR model (Demiralp and Hoover, 2003; Kilian, 2011; Swan-

son and Granger, 1997):

$$\mathbf{x}_{m,t}^\top = \mathbf{x}_{m,t}^\top \mathbf{W} + \mathbf{x}_{m,t-1}^\top \mathbf{A}_1 + \ldots + \mathbf{x}_{m,t-p}^\top \mathbf{A}_p + \mathbf{z}_{m,t}^\top \quad (1)$$

for $t \in \{p, \ldots, T\}$ and for all $m \in \{1, \ldots, M\}$, where p is the *autoregressive order*, and $\mathbf{z}_{m,t}$ is a vector of centered error variables, which are independent within and across time. The error variables are *not* assumed to be Gaussian. Extensions to nonlinear models are also possible, but not explored here, see Section 5 for more discussion.

The matrices $\mathbf{W}$ and $\mathbf{A}_i$ ($i \in \{1, \ldots, p\}$) represent weighted adjacency matrices with nonzero entries corresponding to the intra-slice and inter-slice edges, respectively. When $\mathbf{W}$ is acyclic, as we will assume throughout this paper, these matrices parametrize a DBN. We also assume that this network structure is constant across time. This allows us to write Equation (1) in matrix form as

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}_1\mathbf{A}_1 + \ldots + \mathbf{Y}_p\mathbf{A}_p + \mathbf{Z}, \quad (2)$$

where $\mathbf{X}$ is an $n \times d$ matrix whose rows are $\mathbf{x}_{m,t}^\top$, and the matrices $\mathbf{Y}_1, \ldots, \mathbf{Y}_p$ are time-lagged versions of $\mathbf{X}$. The number n of rows is our effective sample size, and we have $n = M(T + 1 - p)$.

Let $\mathbf{Y} = [\, \mathbf{Y}_1 \mid \cdots \mid \mathbf{Y}_p \,]$ be the $n \times pd$ matrix of time-lagged data. Additionally, let $\mathbf{A} = [\, \mathbf{A}_1^\top \mid \cdots \mid \mathbf{A}_p^\top \,]^\top$ be the $pd \times d$ matrix of inter-slice weights. With this notation, the SEM in (2) takes the compact form:

$$\mathbf{X} = \mathbf{X}\mathbf{W} + \mathbf{Y}\mathbf{A} + \mathbf{Z}\,. \quad (3)$$

This general formulation makes it possible to consider scenarios in which the time-lagged data matrix $\mathbf{Y}$ does not necessarily cover a contiguous sequence of time slices (i.e., from $t - p$ to $t - 1$). For instance, in time series that exhibit known seasonality patterns, one can include in the lagged data matrix $\mathbf{Y}$ only those time points that have an impact on the variables at time t.

Identifiability Identifiability in SVAR models is a central topic of the econometrics literature (see Kilian, 2011 for a review). Identifiability of $\mathbf{A}$ follows from standard results on vector autoregressive (VAR) models, whereas identifiability of $\mathbf{W}$ is more difficult to establish. We focus on two special cases where identifiability holds:

- The errors $\mathbf{z}_{m,t}$ are non-Gaussian. Identifiability in this model is a well-known consequence of Marcinkiewicz's theorem on the cumulants of the normal distribution (Kagan et al., 1973; Marcinkiewicz, 1939) and independent component analysis (ICA), see Hyvärinen et al. (2010); Lanne et al. (2017); Moneta et al. (2011).
- The errors $\mathbf{z}_{m,t}$ are standard Gaussian, i.e. $\mathbf{z}_{m,t} \sim \mathcal{N}(0, I)$. Identifiability of this model is an immediate consequence of Theorem 1 of Peters and Bühlmann (2013) and the acyclicity of $\mathbf{W}$.

In what follows, we assume that one of these two conditions on $\mathbf{z}_{m,t}$ holds.

2.2 Optimization problem

Given the data $\mathbf{X}$ and $\mathbf{Y}$, our goal is to estimate weighted adjacency matrices $\mathbf{W}$ and $\mathbf{A}$ that correspond to DAGs. The edges in $\mathbf{A}$ go only forward in time and thus they do not create cycles. In order to ensure that the whole network is acyclic, it thus suffices to require that $\mathbf{W}$ is acyclic. Minimizing the least-squares loss with the acyclicity constraint gives the following optimization problem:

$$\min_{\mathbf{W},\,\mathbf{A}} \ \ell(\mathbf{W}, \mathbf{A}) \ \text{ s.t. } \ \mathbf{W} \text{ is acyclic}\,, \quad (4)$$
$$\text{where } \ell(\mathbf{W}, \mathbf{A}) = \frac{1}{2n}\|\mathbf{X} - \mathbf{X}\mathbf{W} - \mathbf{Y}\mathbf{A}\|_F^2.$$

To enforce the sparsity of $\mathbf{W}$ and $\mathbf{A}$, we also introduce ℓ_1 penalties in the objective function. Let the regularized optimization problem be

$$\min_{\mathbf{W},\,\mathbf{A}} \ f(\mathbf{W}, \mathbf{A}) \ \text{ s.t. } \ \mathbf{W} \text{ is acyclic}\,, \quad (5)$$
$$\text{with } f(\mathbf{W}, \mathbf{A}) = \ell(\mathbf{W}, \mathbf{A}) + \lambda_{\mathbf{W}}\|\mathbf{W}\|_1 + \lambda_{\mathbf{A}}\|\mathbf{A}\|_1\,,$$

where $\|\cdot\|_1$ stands for the element-wise ℓ_1 norm. Regularization is especially useful in cases with much fewer samples than variables, $n \ll d$. For example, for static BNs, the least-squares loss has been shown to be consistent in high-dimensional settings for learning DAGs (van de Geer et al., 2013; Aragam et al., 2015).

The key difficulty in solving the optimization problem (5) is the acyclicity constraint on $\mathbf{W}$. To deal with this, we use an equivalent formulation of acyclicity via the trace exponential function, due to Zheng et al. (2018). They show that the function $h(\mathbf{W}) = \operatorname{tr} e^{\mathbf{W}\circ\mathbf{W}} - d$ satisfies $h(\mathbf{W}) = 0$ if and only if $\mathbf{W}$ is acyclic. Here, $\circ$ denotes the Hadamard product of two matrices. Replacing the acyclicity constraint in (5) with the equality constraint $h(\mathbf{W}) = 0$, the resulting equality constrained program can be solved as in Zheng et al. (2018), using the augmented Lagrangian method. This translates the problem to a series of unconstrained problems of the form

$$\min_{\mathbf{W},\,\mathbf{A}} F(\mathbf{W}, \mathbf{A}) \quad (6)$$

where F is the following smooth augmented objective:

$$F(\mathbf{W}, \mathbf{A}) = f(\mathbf{W}, \mathbf{A}) + \frac{\rho}{2}h(\mathbf{W})^2 + \alpha h(\mathbf{W})\,. \quad (7)$$

By writing $\mathbf{W} = \mathbf{W}_+ - \mathbf{W}_-$ such that $\mathbf{W}_+ \geq 0$ and $\mathbf{W}_- \geq 0$ (and analogously for $\mathbf{A}$), we transform (6) to

a quadratic program with non-negative constraints and twice the number of variables. The resulting problem can be solved using standard solvers such as `L-BFGS-B` (Zhu et al., 1997).

Following Zheng et al. (2018), in order to reduce the effect of numerical error from computing $h(\mathbf{W})$, we eliminate weights close to 0 by thresholding the entries of $\mathbf{W}$ and $\mathbf{A}$. Overall, DYNOTEARS has 4 hyper-parameters: the regularization constants $\lambda_{\mathbf{W}}$ and $\lambda_{\mathbf{A}}$, and the weight thresholds $\tau_{\mathbf{W}}$ and $\tau_{\mathbf{A}}$.

2.3 Alternative formulations

An alternative approach to jointly minimizing (6) over $(\mathbf{W}, \mathbf{A})$ is to use a two-stage optimization procedure, similar to those in Chen and Chihying (2007); Demiralp and Hoover (2003); Hyvärinen et al. (2010). We can rewrite Equation (2) as the structural VAR (SVAR)

$$\mathbf{X}\mathbf{A}_0 = \mathbf{Y}_1\mathbf{A}_1 + \ldots + \mathbf{Y}_p\mathbf{A}_p + \mathbf{Z}\,, \tag{8}$$

where $\mathbf{A}_0 = \mathbf{I} - \mathbf{W}$. With the acyclicity constraint, $\mathbf{W}$ can be written as an upper triangular matrix. It follows that $\mathbf{A}_0$ has full rank and is invertible. Right-multiplying (8) by $\mathbf{A}_0^{-1}$ gives

$$\mathbf{X} = \mathbf{Y}_1\mathbf{B}_1 + \ldots + \mathbf{Y}_p\mathbf{B}_p + \mathbf{e}\,, \tag{9}$$

where $\mathbf{B}_i = \mathbf{A}_i\mathbf{A}_0^{-1}$ $(i \in \{1, \ldots, p\})$ and $\mathbf{e} = \mathbf{Z}\mathbf{A}_0^{-1}$ is a noise term whose elements are correlated (for fixed t). Equation (9) is now a *reduced-form* VAR that we can fit using least-squares (Amisano and Giannini, 2012). This produces estimates of the matrices $\mathbf{B}_i$ and of the residuals $\mathbf{e}$. Recall that

$$\mathbf{e}\mathbf{A}_0 = \mathbf{Z} \Leftrightarrow \mathbf{e}(\mathbf{I} - \mathbf{W}) = \mathbf{Z} \Leftrightarrow \mathbf{e} = \mathbf{e}\mathbf{W} + \mathbf{Z}\,, \tag{10}$$

so we can estimate $\mathbf{W}$ by applying static NOTEARS to the matrix of residuals $\mathbf{e}$.

One disadvantage of the two-step approach is that enforcing sparsity via ℓ_1 regularization is no longer straightforward, as $\mathbf{B}_i$ and $\mathbf{A}_i$ do not necessarily have the same sparsity patterns. Moreover, any errors in estimating the $\mathbf{B}_i$ at the right sparsity levels propagate to the residuals $\mathbf{e}$, which directly affects the estimation of $\mathbf{W}$. In turn, this affects the final step of the process, where one calculates $\mathbf{A}_i = \mathbf{B}_i(\mathbf{I} - \mathbf{W})$. See Appendix A for additional discussion.

3 Experiments

To validate the effectiveness of the proposed method, we performed a series of simulation experiments for which the ground truth is known in advance. We focus here on the main results; more detailed results can be found in Appendix B.

3.1 Setup

To benchmark DYNOTEARS against existing approaches, we simulate data according to the SEM from (3). There are three steps to this process: 1) generating the weighted graphs $\mathcal{G}_{\mathbf{W}}$ and $\mathcal{G}_{\mathbf{A}}$, 2) generating data matrices $\mathbf{X}$ and $\mathbf{Y}$ consistent with these graphs, and 3) running all algorithms on $\mathbf{X}$ and $\mathbf{Y}$ and computing performance metrics. See Appendix B.4 for a detailed description of steps 1) and 2).

There are three algorithms that we use for benchmarking. The first algorithm is based on a general approach from Murphy (2002); here, we use static NOTEARS and Lasso regression to estimate $\mathbf{W}$ and $\mathbf{A}$ independently. The second algorithm is the SVAR estimation method based on LiNGAM (Hyvärinen et al., 2010). The third algorithm is tsGFCI (Malinsky and Spirtes, 2018). For details, see Appendix B.1.

3.2 Results

We start by illustrating the typical performance of DYNOTEARS on one simulated dataset in Figure 2. We follow the process from Appendix B.4 to generate data with Gaussian noise, $n = 500$ samples, $d = 5$ variables, and $p = 3$ autoregressive terms. The intra-slice DAG is an ER graph with mean degree equal to 4 (counting edges in either direction), and the inter-slice DAG is an ER graph with a mean out-degree equal to 1. The base of the exponential decay of inter-slice weights is $\eta = 1.5$. We run DYNOTEARS on this dataset with regularization parameters $\lambda_{\mathbf{W}} = \lambda_{\mathbf{A}} = 0.05$ and weight thresholds $\tau_{\mathbf{W}} = \tau_{\mathbf{A}} = 0.01$. The low values for the thresholds are intentional, as we want to highlight that the algorithm recovers a structure close to the ground truth even without tweaking these parameters. We consider here the case where the autoregressive order p is known, however, see Appendix B.5 for details on how to estimate p when it is unknown. As Figure 2 shows, estimated weights are close to the true weights for both $\mathbf{W}$ and $\mathbf{A}$. The only significant omission is a missing entry in the third row and third column of $\mathbf{A}_2$. Performance is similar without ℓ_1 regularization and for larger values of p.

In Figure 3, we compare the performance of DYNOTEARS to that of three other algorithms (see Appendix B.1). We consider two distributions for the noise, Gaussian and Exponential, and two choices for the number of samples, $n \in \{50, 500\}$. Within each of the four panels, each column corresponds to one choice of intra-slice graph model and mean degree; for instance, ER2 indicates that we used an Erdős–Rényi graph model with a mean degree of 2. Similarly, each row corresponds to one choice of inter-slice graph model and mean degree. For each individual plot,

Figure 2: Example results using DYNOTEARS for data with Gaussian noise, $n = 500$ samples, $d = 5$ variables, and $p = 3$ autoregressive terms. We set the regularization parameters $\lambda_{\mathbf{W}} = \lambda_{\mathbf{A}} = 0.05$ and the weight thresholds $\tau_{\mathbf{W}} = \tau_{\mathbf{A}} = 0.01$. Our algorithm recovers weights that are close to the ground truth.

we generate n data samples with autoregression order $p = 1$ for five choices of the number of variables, $d \in \{5, 10, 20, 50, 100\}$. The vertical axis indicates the performance of each algorithm in terms of the F1 score, which we calculate separately for intra-slice and inter-slice matrices. We discuss the selection of hyperparameter values for the four algorithms in Appendix B.3. The relative ranking of the four algorithms is not especially sensitive to these hyperparameters. In particular, the regularization parameters are largely irrelevant when there is sufficient data ($n \gg dp$).

DYNOTEARS is the best-performing algorithm in Figure 3, with F1 scores close to 1 for $n = 500$. DYNOTEARS is also the best-performing algorithm when the number of variables exceeds the number of samples (see panels (c) and (d) for $d = 100$). This high-dimensional case is especially difficult, yet common in applications, and we discuss one such example in Section 4.2. The second-best algorithm is tsGFCI. However, its performance tends to degrade as we add more edges to the ground-truth graphs; see Figure B.3 in the Appendix. The variance in performance is also larger for tsGFCI than for the other algorithms. As expected, learning intra-slice and inter-slice structure separately with NOTEARS + Lasso underperforms DYNOTEARS. In particular, we find that the Lasso step falsely identifies some intra-slice edges as inter-slice. LiNGAM is an algorithm designed for non-Gaussian data, so its poor performance in panels (a) and (c) of Figure 3 is not surprising. However, even on data with exponential noise (panels (b) and (d) of Figure 3), its performance degrades significantly as d increases. Appendix B.7 contains additional figures that compare the performance of the four algorithms using metrics other than the F1 score.

4 Applications

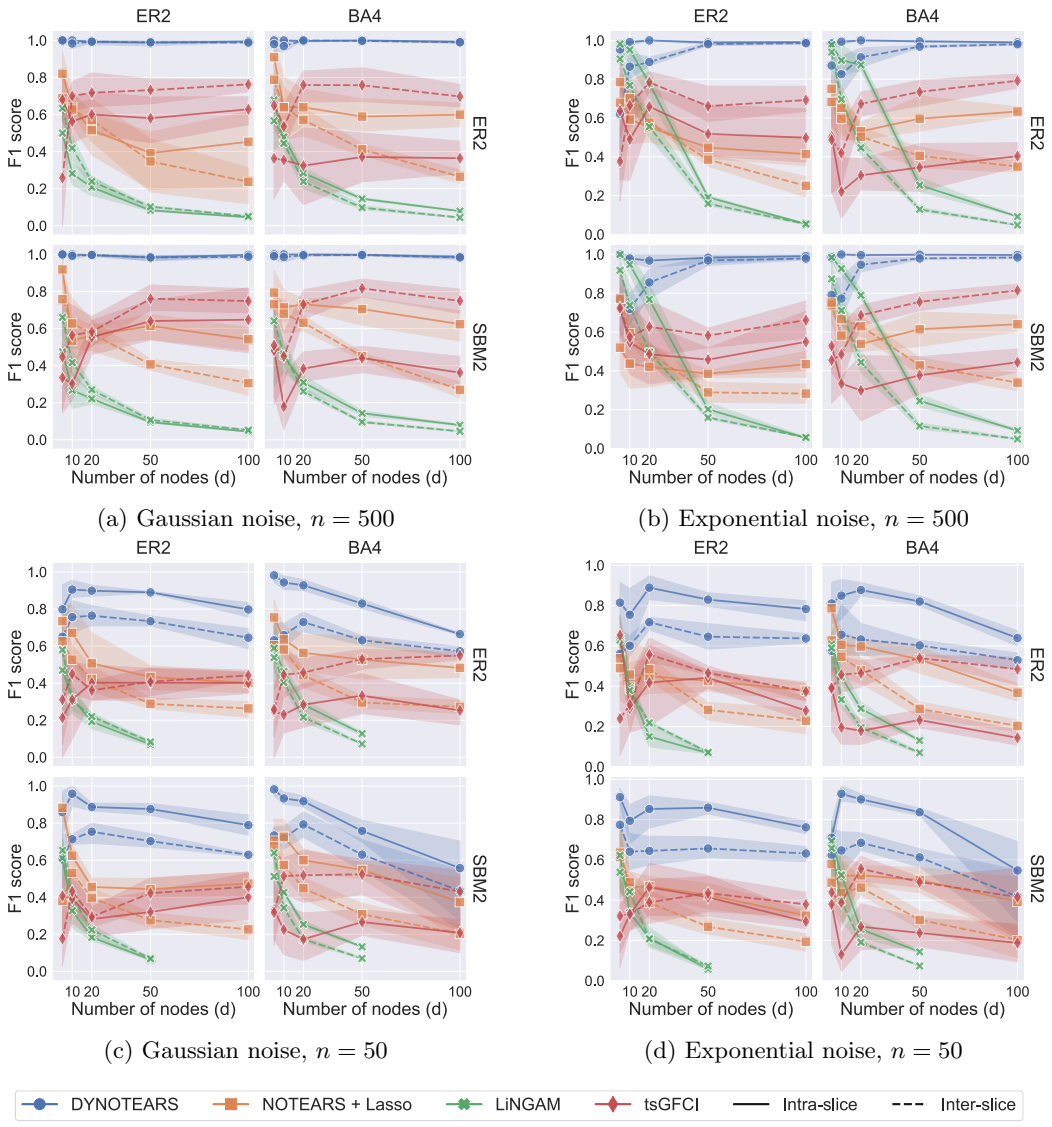
Our approach allows us to detect whether contemporaneous or time-lagged relationships are more meaningful in a given dataset. We discuss two examples for which each type of interaction is dominant.

4.1 S&P 100 stock returns

We apply DYNOTEARS to a financial dataset of daily stock returns of companies in the S&P 100. The dataset was obtained using the `yahoofinancials`[1] Python package and consists of daily closing prices from 1 July 2014 to 30 June 2019, inclusive. To account for non-stationarity in the stock price data, we use log-returns, defined as the temporal difference of the logarithms of the stock prices. We test the stationarity of the transformation using the Augmented Dick-Fuller test and reject the Null hypothesis of a unit-root with $p_{\text{value}} < 0.01$. We normalise the log-returns so that they have zero mean and unit variance. If we did not perform this step, the regularization would favor low-variance stocks, whose (linear) weights would be relatively larger than for high-risk stocks.

The resulting dataset contains $n = 1257$ samples (i.e., trading days) and $d = 97$ variables (i.e., stocks). We apply DYNOTEARS with $p = 1$, $\lambda_{\mathbf{W}} = 0.1$, and $\lambda_{\mathbf{A}} = 0.1$. The values are selected via grid search; see Figure C.1 in the supplement. We hold out the final 400 data points of the series for validation ($\approx 32\%$) and we discard 2 trading days to avoid validation-set

[1] `https://pypi.org/project/yahoofinancials/`



(a) Gaussian noise, $n = 500$

(b) Exponential noise, $n = 500$

(c) Gaussian noise, $n = 50$

(d) Exponential noise, $n = 50$

Figure 3: F1 scores for different noise models (Gaussian, Exponential) and different sample sizes ($n \in \{50, 500\}$.). Each panel contains results for two different choices of intra-slice graphs (columns) and inter-slice graphs (rows). Every marker corresponds to the mean performance across 5 algorithm runs, each on a different simulated dataset. F1 scores for intra-slice and inter-slice edges appear in continuous and dashed lines, respectively. DYNOTEARS typically outperforms the other algorithms. Note that the ICA step in LiNGAM does not work for $n < d$, so the corresponding markers for $d = 100$ are missing from panels (c) and (d).

contamination. Running DYNOTEARS on the whole dataset took 3.8 minutes on a laptop, with 25.8 minutes total CPU time.

The final graph does not contain inter-slice edges. This means that the best prediction of future returns is the current return. This could be evidence for the efficient market hypothesis, which says that asset prices contain all information available (Fama, 1970). While there are no inter-slice edges, the validation loss for DYNOTEARS is slightly smaller than for static NOTEARS with the same $\lambda_{\mathbf{W}}$ value, $167.387 < 167.784$. Comparing the two graphs, the biggest difference is that the edge between `GOOGL` and `GOOG` flips.[2] The dynamic parametrization helps to find a better local optimum.

Figure 4 provides a visualization of the estimated intra-slice weights matrix $\mathbf{W}$. When we order the stocks by industry sector, we obtain an approximately block-diagonal structure. Thus, two stocks are more likely to

[2]Both shares are issued by Alphabet, the holding company of Google, and they are equal in terms of dividends. However, only `GOOGL` includes shareholder voting rights.

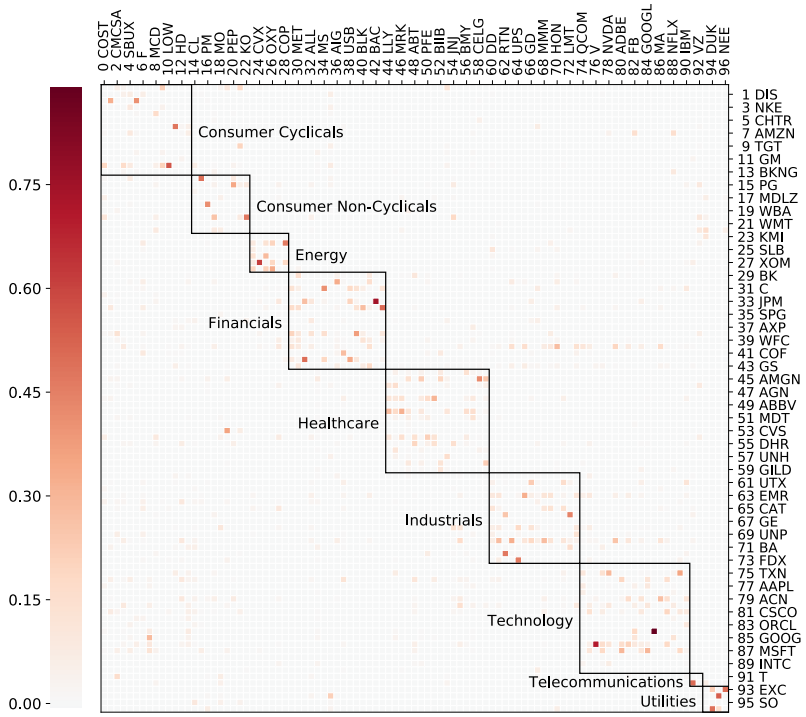


Figure 4: Estimated intra-slice matrix $\mathbf{W}$, with rows and columns sorted by sector. A row is the source stock, a column is the target stock. Stocks in the same sector tend to have larger edge weights between them. The normalized mutual information between a community-induced partition and the sector-based partition indicates agreement between the estimated network and the sectors.

influence each other if they belong to the same sector than if they are from different sectors. The weight of 96% of all identified edges is positive, so most stocks move together. This percentage is roughly the same for edges within and between sectors. One noteworthy feature in Figure 4 is that Amazon (`AMZN`), which is part of the Consumer Cyclicals sector, is connected to many of the technology stocks, including Facebook, Netflix, NVIDIA, Google, and Microsoft.

To quantify the sector relationship, we apply the Louvain method (Blondel et al., 2008), a community detection algorithm and a form of graph clustering, to $\mathbf{W}$.[3] This partitions the set of stocks into subsets that correspond to densely-connected subgraphs in the estimated Bayesian network. The normalized mutual information (NMI, Li et al., 2001) between this community-induced partition and the sector-based partition is approximately 0.79, which indicates close agreement.[4]

[3] We use the *python-louvain* package from `https://pypi.org/project/python-louvain/`.

[4] An NMI value of 1 corresponds to identical partitions, whereas a value of 0 corresponds to independent partitions.

4.2 DREAM4 gene expression data

We benchmark DYNOTEARS on the DREAM4 network inference challenge (Marbach et al., 2009), in which the objective is to learn gene regulatory networks from gene expression data. DREAM4 consists of 5 independent datasets, each with 10 different time-series recordings for 100 genes across 21 time steps. We preprocess our data and choose hyperparameters $\lambda_{\mathbf{A}}$ and $\lambda_{\mathbf{W}}$ through 10-fold cross validation, details of which can be found in Appendix D.1. In Lu et al. (2019), the authors compared different approaches to learning these networks, including methods based on mutual information, Granger causality, dynamical systems, decision trees, Gaussian processes (GPs), and DBNs. Unsurprisingly, their results indicate that flexible nonparametric methods such as GPs and decision trees perform the best. It is nonetheless instructive to compare the performance of DYNOTEARS to these methods for two reasons: 1) This gives us a sense of how much is lost in assuming the linear model (1), and 2) For an apples-to-apples comparison, we can still compare DYNOTEARS to the DBN methods (six in total) tested by Lu et al. (2019).

As in the original DREAM4 challenge, we use mean AUPR and AUROC across the 5 datasets to compare

Algorithm	Mean AUPR	Mean AUROC
DYNOTEARS	**0.173**	0.664
G1DBN	0.110	**0.676**
ScanBMA	0.101	0.657
VBSSMb	0.096	0.618
VBSSMa	0.086	0.624
Ebdbnet	0.043	0.643

Table 1: AUPR and AUROC scores of DBN algorithms on the DREAM4 dataset. Values for methods other than DYNOTEARS are from Lu et al. (2019).

DYNOTEARS to the 23 algorithms presented in Lu et al. (2019). The results are as follows:

- DYNOTEARS achieves an average AUROC of 0.664 and an average AUPR of 0.173. Among the six DBN methods tested, this ranks **1st** and **2nd** in AUPR and AUROC, respectively, as shown in Table 1.
- Furthermore, DYNOTEARS is within one standard deviation of the best performing method (G1DBN) based on AUPR, and no other method is within one standard deviation of DYNOTEARS based on AUROC.
- Overall, this ranks **4th** in AUPR and **8th** in AUROC (see Table D.2 and Table D.1, respectively). While the top performing methods were based on nonparametric models such as GPs, DYNOTEARS still outperforms several other nonparametric methods despite its use of the linear model (1).

5 Discussion

In this paper, we proposed DYNOTEARS, an algorithm for learning dynamic Bayesian networks, inspired by recent work on structure learning for static Bayesian networks using differentiable acyclicity constraints (Zheng et al., 2018). Our algorithm learns both intra-slice and inter-slice dependencies between variables simultaneously, in contrast with some existing methods that perform these estimations in succession.

An important feature of DYNOTEARS is its simplicity, both in terms of formulating an objective function and in terms of optimizing it. It performs well on simulated data across a wide range of parameter choices in the data-generation process.

We also applied DYNOTEARS to two empirical datasets from different application domains, finance and molecular biology. The results reveal insightful patterns in the data. Both of these applications have $d \approx 100$, confirming that DYNOTEARS can be applied to larger datasets than those considered in most existing work on DBNs.

To conclude, we briefly discuss some limitations and possible extensions of DYNOTEARS.

Assumptions We have assumed that the structure of the DBN is fixed through time and is identical for all time series in the data (i.e., it is the same for all $m \in \{1, \dots, M\}$). It would be useful to relax these assumptions in various ways, for example by allowing the structure to change smoothly over time (Song et al., 2009) or at discrete change points that we infer from the data (Grzegorczyk and Husmeier, 2011). Another topic for future work is to investigate the behaviour of the algorithm on nonstationary or cointegrated time series (Malinsky and Spirtes, 2019), or in situations with confounders (Huang et al., 2015; Malinsky and Spirtes, 2018). A possible approach is to apply a post-processing step to the output of DYNOTEARS so as to remove spurious relationships between variables (e.g., by using statistical tests).

Undersampling As pointed out by a reviewer, as with most DBN models, we implicitly assume that the sampling rate of the process is at least as high as the fluctuations in the underlying causal process. See for example Gong et al. (2015); Hyttinen et al. (2016); Plis et al. (2015); Cook et al. (2017). As a check on the sensitivity of DYNOTEARS to this (strong) assumption, we tested a modification of our approach for undersampled data adapted from Cook et al. (2017). In essence, by running DYNOTEARS on the data and then post-processing any intra-slice edges by making them bi-directed, we can obtain a graph which is comparable to the methods suggested in Cook et al. (2017). Although our method was not designed to handle undersampling, it achieves lower false-negative rates compared to other methods. Of course, more careful modifications to handle undersampling is an interesting direction for future work.

Nonlinear dependence Finally, we emphasize that linear assumption in (1-2) is made purely for simplicity, in order to keep the focus on the most salient dynamic and temporal aspects of this problem. For example, using the general approach outlined in Zheng et al. (2019), it is possible to model complicated nonlinear dependencies via neural networks or orthogonal basis expansions. Furthermore, it is straightforward to replace the least squares loss with the logistic loss (or more generally, any exponential family log-likelihood) to model binary data. It is also possible to go a step further and consider combinations of continuous and discrete data (Andrews et al., 2019), which is important for many real-world applications.

References

Amisano, G. and Giannini, C. (2012). *Topics in Structural VAR Econometrics*. Springer Berlin Heidelberg.

Andrews, B., Ramsey, J., and Cooper, G. F. (2019). Learning high-dimensional directed acyclic graphs with mixed data-types. In *The 2019 ACM SIGKDD Workshop on Causal Discovery*, pages 4–21.

Aragam, B., Amini, A. A., and Zhou, Q. (2015). Learning directed acyclic graphs with penalized neighbourhood regression. *arXiv preprint arXiv:1511.08963*.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Jounal of Statistical Mechanics: Theory and Experiment*, 2008(10):10008.

Bouckaert, R. R. (1993). Probabilistic network construction using the minimum description length principle. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: European Conference ECSQARU '93*, volume 747 of *Lecture Notes in Computer Science*, pages 41–48. Springer-Verlag.

Chen, P. and Chihying, H. (2007). Learning causal relations in multivariate time series data. *Economics: The Open-Access, Open-Assessment E-Journal*, 1:11.

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

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

Cook, J. W., Danks, D., and Plis, S. M. (2017). Learning dynamic structure from undersampled data. In *Proceedings of the UAI Causality Workshop*.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150.

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

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417.

Gong, M., Zhang, K., Schoelkopf, B., Tao, D., and Geiger, P. (2015). Discovering temporal causal relations from subsampled data. In *International Conference on Machine Learning*, pages 1898–1906.

Grzegorczyk, M. and Husmeier, D. (2011). Non-homogeneous dynamic Bayesian networks for continuous data. *Machine Learning*, 83(3):355–419.

Haufe, S., Müller, K.-R., Nolte, G., and Krämer, N. (2010). Sparse causal discovery in multivariate time series. In *Causality: Objectives and Assessment*, pages 97–106.

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

Huang, B., Zhang, K., and Schölkopf, B. (2015). Identification of time-dependent causal model: A Gaussian process treatment. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Hyttinen, A., Plis, S., Järvisalo, M., Eberhardt, F., and Danks, D. (2016). Causal discovery from subsampled time series data by constraint optimization.

Hyvärinen, A., Zhang, K., Shimizu, S., and Hoyer, P. O. (2010). Estimation of a structural vector autoregression model using non-Gaussianity. *Journal of Machine Learning Research*, 11(May):1709–1731.

Kagan, A. M., Rao, C. R., and Linnik, Y. V. (1973). Characterization problems in mathematical statistics.

Kilian, L. (2011). Structural vector autoregressions. *CEPR Discussion Paper No. DP8515*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

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

Li, M., Badger, J. H., Chen, X., Kwong, S., Kearney, P., and Zhang, H. (2001). An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154.

Linzner, D., Schmidt, M., and Koeppl, H. (2019). Scalable structure learning of continuous-time bayesian networks from incomplete data. *arXiv preprint arXiv:1909.04570*.

Lu, J., Dumitrascu, B., McDowell, I. C., Jo, B., Barrera, A., Hong, L. K., Leichter, S. M., Reddy, T. E., and Engelhardt, B. E. (2019). Causal network inference from gene transcriptional time series response to glucocorticoids. *bioRxiv*, 587170.

Lütkepohl, H. (2005). *New introduction to multiple time series analysis*. Springer Science & Business Media.

Malinsky, D. and Spirtes, P. (2018). Causal structure learning from multivariate time series in settings with unmeasured confounding. In *Proceedings of 2018 ACM SIGKDD Workshop on Causal Discovery*, pages 23–47.

Malinsky, D. and Spirtes, P. (2019). Learning the structure of a nonstationary vector autoregression. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2986–2994.

Marbach, D., Schaffter, T., Mattiussi, C., and Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239.

Marcinkiewicz, J. (1939). Sur une propriété de la loi de Gauss. *Mathematische Zeitschrift*, 44(1):612–618.

Meng, Z., Han, S., Liu, P., and Tong, Y. (2019). Improving speech related facial action unit recognition by audiovisual information fusion. *IEEE Transactions on Cybernetics*, 49(9):3293–3306.

Moneta, A., Chlaß, N., Entner, D., and Hoyer, P. (2011). Causal search in structural vector autoregressive models. In *NIPS Mini-Symposium on Causality in Time Series*, pages 95–114.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley. AAI3082340.

Nefian, A. V., Liang, L., Pi, X., Liu, X., and Murphy, K. (2002). Dynamic Bayesian networks for audio-visual speech recognition. *EURASIP Journal on Advances in Signal Processing*, 2002(11):783042.

Peters, J. and Bühlmann, P. (2013). Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228.

Plis, S., Danks, D., Freeman, C., and Calhoun, V. (2015). Rate-agnostic (causal) structure learning. In *Advances in neural information processing systems*, pages 3303–3311.

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

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

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

Robinson, R. W. (1977). Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V*, pages 28–43. Springer.

Song, L., Kolar, M., and Xing, E. P. (2009). Time-varying dynamic Bayesian networks. In *Advances in Neural Information Processing Systems 22*, pages 1732–1740.

Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research*, 11(May):1643–1662.

Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72.

Spirtes, P., Glymour, C. N., Scheines, R., Heckerman, D., Meek, C., Cooper, G., and Richardson, T. (2000). *Causation, prediction, and search*. MIT Press.

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

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

van de Geer, S., Bühlmann, P., et al. (2013). ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics*, 41(2):536–567.

Van Gerven, M. A. J., Taal, B. G., and Lucas, P. J. F. (2008). Dynamic Bayesian networks as prognostic models for clinical patient management. *Journal of Biomedical Informatics*, 41(4):515–529.

Zandonà, A., Vasta, R., Chiò, A., and Di Camillo, B. (2019). A dynamic bayesian network model for the simulation of amyotrophic lateral sclerosis progression. *BMC Bioinformatics*, 20(4):118.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, pages 9472–9483.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. P. (2019). Learning sparse nonparametric dags. *arXiv preprint arXiv:1909.13189*.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.