Time-Varying DAGs with NOTEARS Final Report

Maxwell Bond Wang (mbwang) [1] **Chang Shi (changshi)** [1] **Wuwei Lin (wuweil)** [1] **Michael Kronovet (mkronove)** [1]

Introduction

Many processes in the natural sciences, particularly within biological and medical applications, can be represented as dynamical networks. For example, the gene expression of cells undergoes a large amount of transformation that can be linked to processes such as the development of progenitor stem cells or the transformation of healthy tissue into cancerous tumors. Due to this, there has been considerable interest in investigating the dynamics of single-cell gene expression with the hopes of identifying diagnostic biomarkers or therapeutic targets, especially considering the advent of genome-editing technologies (Lin & Bar-Joseph, 2019). However, linking the changes of a single cell to other important dynamic processes occurring in surrounding cells/tissues is a non-trivial task. Cell development is oftentimes not synchronized within a single sample or tissue, introducing the need for flexibility in measuring temporal correlations. Furthermore, in single-cell data it is often unclear how cells have differentiated and difficult to follow the response of specific cell types over time.

In order to solve these issues, time-based inference methods using graphical models that seek to account for the biological similarities among certain cells in addition their temporal dependencies have been introduced (Lin & Bar-Joseph, 2019). A Dynamic Bayesian Network (DBN) is a specific type of graphical model that is able to model temporal dependencies as well as spacial dependencies between variables. Like all graphical models it is useful for determining relations between variables in a system, and therefore we believe it will be a good fit for modeling the time-series single-cell gene expression data.

The goal of this project is to optimize the Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning (NOTEARS) framework (Zheng et al., 2018) for learning time-varying DAGs. The NOTEARS framework is a score-based continuous optimization algorithm for DAGs, which formulates structure learning as a least-squares minimization problem. This convenient formulation allows us to easily add regularizers to the objective function. DYNOTEARS (Pamfil et al., 2020) is an adaptation of NOTEARS used to learn dynamic graph structures from time-series data. In contrast to this method, we attempt to modify the NOTEARS framework by learning independent graphs across time and then aggregating them together. We apply several different methods that rely on unique assumptions about the relationships of variables over time and multiple graph aggregation techniques. We apply our methods to time series data where the ground truth is static across time as well as time series data where the ground truth graph is dynamic over time.

We summarize our contributions as followed:

1. We adapt the NOTEARS for the estimation of the graphical models using time-varying data.
2. We introduce a theoretical modification to the existing DYNOTEARS framework to impose regularization on graph nodes that are close together in the time series.
3. We compare several graph aggregation techniques to combine independent time series graphs into a static ground truth graph.
4. We explore different time regularization techniques for combining independent, static graphs into dynamic graphs.

Related Work

Structure learning for DAG

The two common types of algorithms used to find the structure of DAGs are score-based methods and constraint-based methods. Constraint-based methods are sensitive to noise and suffer from error propagation whereas score-based methods are computationally intensive.

Zheng et al. (2018) formulates the structural learning problem for DAG as an optimization problem for a continuous function, which can be solved with existing optimization methods. Zheng et al. (2019) extends the algebraic characterization in NOTEARS to nonparametric SEM by leveraging nonparametric sparsity based on partial derivatives. One of the drawback of NOTEARS is that it uses matrix exponential to characterize the DAG, which requires

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.

10708 Class Project, Spring, 2019.

$O(d^3)$ computational complexity. Lee et al. (2020) introduces spectral radius as an alternative characterization for DAG, which can be approximated in $O(d^2)$ time. It also introduced a gradient-based optimization method to make NOTEARS scalable to handle gene regulatory data.

Learning time-varying networks

There are two key technical hurdles preventing us from an in-depth investigation of the underlying mechanisms of time-varying networks, the lack of measurements and the unavailability of serial snapshots of the time-varying networks. Kalofolias et al. (2017) indicates that depending on the speed with which the hidden structure changes, there might be only very few sample available that correspond to the same or almost the same distribution at a given point in time. Usually, only time series measurements, but not their linkage status are available.

To increase the resolvability, most models for learning time-varying networks are based on some assumptions. A particular emphasis is put on sparsistent estimation, that is, consistent estimation of the graph structure, under a setting in which the number of nodes p in the graph is larger than the sample size n, but the number of neighbors of each node is small, that is, the true graph is sparse. described in Kolar et al. (2008).

Koller & Friedman (2009) summarizes the general estimation framework for time-varying networks as, given data

$$\mathcal{D}_n = \{\mathbf{x}^t | \mathbf{x}^t \sim P(\boldsymbol{\theta}^t; G^t)\}_{t \in \mathcal{T}_n},\ \mathcal{T}_n = \{1/n, 2/n, \ldots, 1\}$$

$$\arg\max\ \ell(\mathcal{D}_n, \{\boldsymbol{\theta}^t\}) - (\{\boldsymbol{\theta}^t\})$$

While the loss $\ell(\mathcal{D}_n, \{\boldsymbol{\theta}^t\})$ measures the fit of model to data, the penalty $(\{\boldsymbol{\theta}^t\})$ balances the complexity of model and encodes structural assumptions about model class. According to the property of networks, smoothly evolving networks and networks with jumps can be learned by given different penalty, such as penalty on parameter changes, or structure changes. The score-based general framework gives strong convenience to our works on learning time-varying DAGs, since it is straight forward to adjust the penalty tern according to our demand of how to learn from time-series data.

Comparing to time invariant networks, much less has been done on modeling dynamical processes that guide topological rewiring and semantic evolution of networks over time. Hanneke & Xing (2007) introduced a new class of models to capture dynamics of networks evolving over discrete time steps, called *temporal Exponential Random Graph Models (tERGMs)*. This class of models uses a number of statistics defined on time-adjacent graphs, for example, ”edge-stability”, “reciprocity”, ”density”, ”transitivity”, etc., to construct a log-linear graph transition model that captures dynamics of topological changes. Guo et al. (2007) incorporate a hidden Markov process into the tERGMs, which imposes stochastic constraints on topological changes in graphs. Unfortunately, even though this class of model is very expressive, the sampling algorithm for posterior inference scales only to small graphs with tens of nodes. Later, Zhou et al. (2008) develops a nonparametric method for estimation of a time-varying Gaussian graphical model, while Kolar et al. (2008) discussed about a counterpart of the discrete Ising model. Both of them successfully build methods on a temporally smoothed $l1$-regularized logistic regression formalism that can be cast as a standard convex-optimization problem and solved efficiently using generic solvers scalable to large networks.

While the aforementioned works are on undirected networks. 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. There are many methods for learning DBNs in the literature. Some approaches ignore contemporaneous dependencies and recover only time-lagged relationships. Others learn both types of relationships independently. 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. There are also hybrid algorithms that combine conditional-independence tests and a local search to improve the BIC score. While all of these methods suffer from the curse of dimensionality, (Pamfil et al., 2020) largely reduces the computation complexity by applying smooth equality constraint.

In 2009, Synapse.org organized the DREAM4 (Dialogue for Reverse Engineering Assessments and Methods 4) competition in which participants were challenged to infer the structure of gene regulatory networks given simulated data. Pirgazi & Khanteymoori (2018) discusses and evaluates the best performing methods used in this challenge as well as subsequent methods that were invented years later. The best performing method on the DREAM4 challenge when it was still active was the GENIE3 algorithm. This algorithm partitioned the task of learning a gene regulatory network between p genes into p different regression problems, where in each regression the expression pattern of one of the genes was predicted from the expression patterns of all the other genes using tree-based ensemble methods. In addition to the GENIE3 algorithm, Pirgazi & Khanteymoori (2018) evaluates the BMALR algorithm, which is a commonly used method for inferring molecular interactions in biological systems. BMALR algorithm computes posterior probabilities of the edges from gene regulators to a target gene within a hybrid framework of Bayesian model averaging and linear regression methods.

We plan to use the results obtained by Pirgazi & Khanteymoori (2018) when testing these state of the art algorithms as a reference to compare them to our implementation of NOTEARS on the DREAM4 challenge dataset.

Methods

Proposing Time-Based Regularization on DYNOTEARS

DYNOTEARS identifies a dynamical Bayesian network as follows. Consider M realizations of a dynamical random variable with the m-th time series given by $\{x_{m,t}\}_{t\in\{0,\dots,T\}}$. Each $x_{m,t} \in \mathbb{R}^d$ is a d-dimensional random variable. Each time-step of this random variable, $x_{m,t}$, is affected by the P most recent time-steps: $x_{m,t-1}, \dots x_{m,t-P}$. There are also relations between the d different features of $x_{m,t}$ within the same time point. We can formulate all of this as,

$$x^T_{m,t} = x^T_{m,t}W + x^T_{m,t-1}A_1 + \dots + x^T_{m,t-P}A_P + z^T_{m,t} \quad (1)$$

$z^T_{m,t}$ is merely a noise term. W represents the interactions in x within the same time point and $A_1, \dots, A_P$ represents interactions across different time points. The goal of DYNOTEARS is to identify optimal W and As to best-fit this model. More specifically, it tries to optimize the following loss function where X is a matrix containing all the $x_{m,t}$ and Y is a matrix containing their time-lagged versions.

$$\min_{W,A} l(W,A) = \min_{W,A} \frac{1}{2n}||X - XW - YA||^2_F \quad (2)$$

Here we have concatenated all $A_1, \dots, A_P$ into a single large matrix for simplicity of notation. In DYNOTEARS, there is also an l_1 penalty on W and A to enforce sparsity.

For this project, we want to allow W and/or A to vary with time to add an additional degree of flexibility in capturing non-stationary signals. Somewhat naively, we can simply allow W and A to be assigned a different value for each time slice, t, as follows,

$$l(W,A) \propto \sum_{m=1}^{M}\sum_{t=1}^{T}\Bigg|\Bigg| x^T_{m,t} - x^T_{m,t}W_t \dots + \sum_{p=1}^{P} x^T_{m,t-p}A_{t,p}\Bigg|\Bigg|_2 \quad (3)$$

Then we can apply some form of regularization to assure that W and A are not changed too quickly. Assuming some distance measure, $d(W_t, W_{t+1})$, to compare how different two networks are from each other, this can take the form of,

$$\min_{W,A} f(W,A) = \min_{W,A}\left[l(W,A) + \lambda \sum_{t=1}^{T-1}\left(d(W_t, W_{t+1}) + \dots + \sum_{p=1}^{P} d(A_{t,p}, A_{t+1,p})\right)\right] \quad (4)$$

If $\lambda \to \infty$, this becomes the same as the original DYNOTEARS proposal.

The simplest distance measure would be to simply subtract the two matrices and perform a Froebius norm. This has the advantage of having a simple optimization path. Other options include distance measures from the graph kernel literature. One example here would be random walk kernels, which perform random walks on two graphs simultaneously and counts the number of paths that were produced by both walks. Intuitively, there should also be an interesting way to frame this specifically in the context of DAGs.

However, all of this will only work if every trial is temporally aligned with each other. Let's say we are looking at how the top five stocks of some index changes between 2007 and 2011 to model the 2008 housing market crash. Each trial comes from a different index: housing, insurance, energy, etc. For each index, it would make sense for the A's to start encouraging positive overall trends, but then switch to negative ones as the crash occurs before switching back to positive when recovery happens. However, indices involving housing started collapsing before other markers. Why should we force the timing of when W and A changes to be exactly the same for different trials? Let's try adding in a temporal offset instead,

$$l(W,A) \propto \sum_{m=1}^{M}\sum_{t=1}^{T}\Bigg|\Bigg| x^T_{m,t} - x^T_{m,t}W_{t+\tau_m} \dots + \sum_{p=1}^{P} x^T_{m,t-p}A_{t+\tau_m,p}\Bigg|\Bigg|_2 \quad (5)$$

Basically every single trial goes through the "same" evolution of its DBN, but the exact timing of "when" this evolution happens can be phase lagged. We can also apply regularizers to limit the magnitude of this phase lag subject to some application-specific intuition.

Unfortunately, we are unable to directly test out this modification for the sake of this project, because the DYNOTEARS source code has not been made publicly available. Therefore, for the rest of this paper we impose modifications on the NOTEARS framework to aggregate graphs that were computed independently across time series data.

Evaluation

We evaluate the efficacy of our predicted graphs using precision-recall and receiver operating characteristic curves. A precision-recall curve plots fraction of retrieved instances that are relevant versus the fraction of relevant instances that are retrieved. A receiver operating characteristic curve plots the true positive rate versus the false positive rate. These curves are standard for measuring how accurate edge predictions in a graph are compared to the ground truth.

In the DREAM4 challenge, contestants were evaluated on derivatives of the precision-recall and receiver operating characteristic curves. Specifically, the DREAM4 challenge organizers proposed the following evaluation framework:

We begin with the following definitions.

- AUPR as the area under the PR curve
- AUROC as the area under the ROC curve
- p_{aupr} is the probability that a given or larger AUPR is obtained by random ordering of the potential network edges.
- p_{auroc} is the probability that a given or larger AUROC is obtained by random ordering of the potential network edges.

$\bar{p}_{aupr}$ and $\bar{p}_{auroc}$ are the respective geometric means of the individual AUPR and AUROC p-values from the 5 different networks in the dataset.

$$\bar{p}_{aupr} = \sqrt[5]{p^1_{aupr} \cdot p^2_{aupr} \cdot p^3_{aupr} \cdot p^4_{aupr} \cdot p^5_{aupr}} \quad (6)$$

$$\bar{p}_{auroc} = \sqrt[5]{p^1_{auroc} \cdot p^2_{auroc} \cdot p^3_{auroc} \cdot p^4_{auroc} \cdot p^5_{auroc}} \quad (7)$$

The overall score is the log-transformed geometric mean of $\bar{p}_{aupr}$ and $\bar{p}_{auroc}$.

$$\text{score} = -\frac{1}{2}\log_{10}(\bar{p}_{aupr} \cdot \bar{p}_{auroc}) \quad (8)$$

We use these same methods when applying NOTEARS to the DREAM4 dataset to accurately compare our predictions to the best methods from this challenge. We calculate the AUPR and AUROC p-values by simulating 10,000 different 100x100 graphs, where each entry in the graph is drawn from a Uniform(0,1) distribution. The proportion of these random graphs with higher AUPR and AUROC values, yields the AUPR and AUROC p-values, respectively. We expect that random graphs would have AUPR and AUROC values of around 0.5, so if our methods are to perform well (have low p-values and high scores), they would need to have AUPR and AUROC values considerably above 0.5.

Data

We first evaluated the performance of our algorithm on simulated data generated with GeneNetWeaver (GNW) (Schaffter et al., 2011). GNW is a open source framework for the generation of detailed dynamical models of gene regulatory networks to be used as benchmarks, it also provides a network motif analysis that reveals systematic prediction errors, thereby indicating potential ways of improving inference methods. Specifically, we have setup the data of EColi gene network in the GNW dataset and benchmarked the existing methods Zheng et al. (2019); Lee et al. (2020).

The goal of the DREAM4 in silico network challenge was to reverse engineer gene regulation networks given in silico gene expression datasets. The data from the DREAM4 challenge was generated using the GNW version 2.0 (Marbach et al., 2010). The challenge was made up of 3 subchallenges called *In Silico Size 10*, *In Silico Size 100*, and *In Silico Size 100 Multifactorial*. In each of these subchallenges, contestants were evaluated on five different networks. Network topologies were obtained by extracting subnetworks from transcriptional regulatory networks of E. coli and S. cerevisiae. We evaluated our methods on the *In Silico Size 100* subchallenge data, and we compared our methods to the approaches that Pirgazi & Khanteymoori (2018) examined on this same dataset. The data corresponds to noisy measurements of messenger RNA concentration levels, which were normalized such that the maximum normalized gene expression value in the datasets of a given network is one. In the *In Silico Size 100* data, there are 100 messenger RNA features in this dataset for every datapoint. The dataset consists of 5 ground truth graphs, where each graph has 10 data samples. Each data sample has 21 time points with times from 0 to 1000 in increments of 50.

We also use the SynTReN (Synthetic Transcriptional Reulatory Networks) to synthesize expression data of 100 and 300 genes. The sample size of each simulation are 500 or 2000. Two genes, Ecoli and YeastFull are used to generate the dataset.

Our Approaches On the DREAM4 Dataset

Method 1 We treat each time point as another sample from the same data, ignoring the actual time that the datapoint was collected. This allows NOTEARS to compute a single graph from the 210 total data points (10 data samples with 21 time points each). This method ignored temporal relationships in the data and is used as a simple baseline for NOTEARS.

Method 2 We organized the data by the time variable, and then a separate graph was computed for each of the different time groupings. This left us with 21 separate graphs that we averaged together by taking the arithmetic mean of W_t to create the gene regulatory network prediction.

Method 3 Given a time t, we took data from t and $t+1$ and constructed separate features that corresponded to each time point. For example, if our graph originally had features $v1, v2, v3$ our new graph would have features $v1_t, v2_t, v3_t$ and $v1_{t+1}, v2_{t+1}, v3_{t+1}$. We join the data from each sample across the different time points so that we encode data points from t and $t+1$ into a single data point. Our idea with this approach was to encode some time dependencies between the samples using a Markov structure where features can be influenced only by the features that were at the same time or directly before them. For each time step t, the prediction for the time step is a matrix $W_t \in R^{2p\times 2p}$, which contains the estimated graph for the current time step, as well as the connection to the previous time step. Denote W_t as block matrix:

$$W_t = \begin{pmatrix} w_{t,t} & w_{t,t+1} \\ w_{t+1,t} & w_{t+1,t+1} \end{pmatrix} \tag{9}$$

where $w_{t,t}, w_{t,t+1}, w_{t+1,t}, w_{t+1,t+1} \in R^{p\times p}$. Here $w_{t,t+1}, w_{t+1,t}$ is the connection between two adjacent time t and $t+1$.

After getting these $T-1$ (20) graphs that had $2p$ (200) features each, we averaged them into a single 100x100 graph.

For averaging the results over time to produce a single 100x100 graph as the final prediction, two approaches are applied. The first approach is similar to the way we compute the average in Method 2, in this approach we only take $w_{t+1,t+1}$ of each W_t and compute the average, ignoring other parts in W_t. This approach has the assumption that NOTEARS can take the time dependency into account by using both the feature for time t and $t+1$ as the input.

Although experiment results show that this simple approach works well, it ignores the other parts of each W_t and thus fails to consider the estimated connection between time t and $t+1$.

Method 4 This was the same method that we used in method 3 except we use an alternative, "smoothing", approach for the graph agglomeration, which considers the estimated connection between time t and $t+1$ in the graph estimation. In this approach, we perform local smoothing first to produce the smoothed graph for each time t, and then compute the average over them. The local smoothing is performed as followed: for each prediction W_t, we compute the smoothed prediction for the graph $t+1$ as

$$W'_{t+1} = w_{t+1,t+1} + \frac{1}{2}(w_{t,t}w_{t,t+1} + w_{t+1,t}w_{t,t}) \tag{10}$$

And then, we compute the average of W'_{t+1} to produce the final prediction.

Method 5 We applied NOTEARS $T-1$ (20) times, where for each $t \in [T-1]$ we used all the data points from t and $t+1$ to fit our NOTEARS graph while retaining the same number of original features (so we use 20 datapoints for each graph instead of 10 like in the other methods). This method relied on a similar Markov assumption to methods 3 and 4 where the graph at $t+1$ only depends on the features at the same time and the graph at t. However, this method encodes the Markov assumption in the number of datapoints since we fit each graph on double the number of data points instead of double the number of features.

Preliminary Results

Dataset	AP (%)	ROC (%)
GNW-Ecoli	35.7	93.0
GNW-YeastFull	56.9	88.9
EColi-d100-n500	8.5	73.3
EColi-d100-n2000	9.3	74.3
EColi-d300-n500	10.5	83.6
EColi-d300-n2000	12.6	85.4
YeastFull-d100-n500	14.1	83.2
YeastFull-d100-n2000	14.6	85.8
YeastFull-d300-n500	11.6	88.5
YeastFull-d300-n2000	12.9	89.6

Table 1. Benchmark of NOTEARS as baseline on GNW and SynTRen datasets. Average precision (AP) and area under a ROC curve (AUC-ROC) are reported.

We have run NOTEARS on the gene regulatory network dataset GNW and SynTReN as the baseline. As shown in Table 1, GNW-Ecoli and GNW-YeastFull are from the GNW dataset. The other datasets in Table 1 are synthesized expression data from the SynTReN. The name of the synthesized dataset follows the convension $gene\_name - dxxx - nyyy$ where xxx and yyy are the number of edges and the number of nodes, respectively. We will benchmark our method on these datasets in the next step.

Results and Discussion on DREAM4 Dataset

Prior Results

Pirgazi & Khanteymoori (2018) reports the AUPR and AUROC p-values determined for each network in the DREAM4 *In Silico Size 100* dataset using 14 different state of the art network learning methods. In table 2, we show the total scores for each of these methods which we compare to our own approaches.

Model	Total Score
BMALAR	1.60e2
GINIE3	1.51e2
MRNET	1.06e2
ARACNE	1.31e2
BGRMI	1.50e2
CLR	1.49e2
G1DBN	1.02e2
NARROMI	1.39e2
TIGRESS	1.53e2
GENIRF	1.64e2
MIBNI	1.44e2
FBISC	1.35e2
CMI2NI	6.52e1
KFLR	2.03e2

Table 2. Total scores evaluated on the DREAM4 dataset from Pirgazi & Khanteymoori (2018).

AUPR P Values on DREAM4

	Method 1	Method 2	Method 3	Method 4	Method 5
Graph 1	0.4980	0.5012	0.5012	0.5012	0.5046
Graph 2	0.5021	0.5021	0.5139	0.3646	0.5031
Graph 3	0.4955	0.4371	0.1478	0.0998	0.2342
Graph 4	0.4942	0.4942	0.4942	0.4942	0.4993
Graph 5	0.3960	0.5017	0.4530	0.3960	0.4880

Table 3. Evaluation of our methods' area under the precision recall curve p-values on the DREAM4 dataset.

AUROC P Values on DREAM4

	Method 1	Method 2	Method 3	Method 4	Method 5
Graph 1	0.4981	0.5094	0.5155	0.5195	0.5240
Graph 2	0.5662	0.5173	0.5162	0.4306	0.5409
Graph 3	0.5582	0.4660	0.3708	0.3641	0.4192
Graph 4	0.5560	0.5056	0.5029	0.5071	0.5383
Graph 5	0.4453	0.5438	0.4743	0.4453	0.4896

Table 4. Evaluation of our methods' area under the receiver operator characteristic curve p-values on the DREAM4 dataset.

Total Scores

From these graphs we see that the AUROC and AUPR p-values for all 5 of our methods lurk around 50 percent. For a completely random assignment of edge values, we would expect to see areas of 50 percent, which indicates that our algorithm isn't really predicting edges better than random. The P_{AUROC} and P_{AUPR} values for each of our methods are also similar to 50 percent for each method, so our randomly generated graphs have very high probabilities of having larger areas under both the PR and ROC curves. We suspect that we may have received these poor results because there could be nonlinear dependencies between genes in the ground truth regulatory network, whereas we used the linear implementation of NOTEARS. Furthermore, the ground truth regulatory networks were not actually DAGs. The NOTEARS algorithm is capable of learning non-DAG graphs, which it did repeatedly during our training, but this could have also impacted the efficacy of our methods. However, it is possible that the final non-DAG ground truth graphs could be combinations of DAGs learned from each time series. This could provide insight into why our methods that incorporated the time series assumptions performed better than just training NOTEARS on the full dataset and ignoring the time feature. Our methods 3 and 4 which encoded a Markov time assumption into the features of our dataset performed a little better than our other implementations of time series assumptions, which could reveal that there are connections between features at past time points. We propose that future work could focus on utilizing non-linear implementations of NOTEARS, perhaps alternative graph aggregation techniques, or different assumptions of how time influences the data.

	P_{AUROC}	P_{AUPR}	Score
Method 1	0.5225	0.4752	0.3024
Method 2	0.5077	0.4865	0.3035
Method 3	0.4724	0.3855	0.3697
Method 4	0.4497	0.3239	0.4182
Method 5	0.5002	0.4287	0.3343

Table 5. Evaluation of our methods' AUROC and AUPR total scores on the DREAM4 dataset.

Results and Discussion on Dynamically Changing Graph

To assess our approach on a dynamically changing graph, we tested our approach on a simulated dynamic Gaussian Bayesian network. More specifically, we generated samples drawn from an acylic Gaussian Bayesian network where over each time step, each previously existing edge would have a random, independent chance of being removed in the following timestep. Furthermore, previously absent edges that would not break the DAG constraint would also have a random chance to be added. We generated this network under two conditions, one "slowly-changing" network where the change probability was 10%, and one "quickly-changing" network where the probability was 50%. We utilized method 3 as described in the previous section where the width of the temporal smoothing was varied as a hyperparameter and calculated the accuracy (AUC) of our algorithm's ability to deduce the correct network each time step. The results are shown in Figure 1.

We see a concave relationship between the temporal smoothness penalty and average AUC for each time point. As expected, for extremely high regularization penalties, we force the same graph to be deduced each time point which prevents us from capturing dynamic behavior. For low penalties, we do not take advantage of any temporal structure in the data. We also see that the increase in performance by taking advantage of this temporal structure is much higher for the slowly changing graph than the quickly changing one. This is expected since a graph from one time point utility in aiding our estimate of the graph in future time points is inversely proportional to how quickly that network is changing. Future work for this dynamic implementation of NOTEARS could experiment with alternative temporal regularization tech-

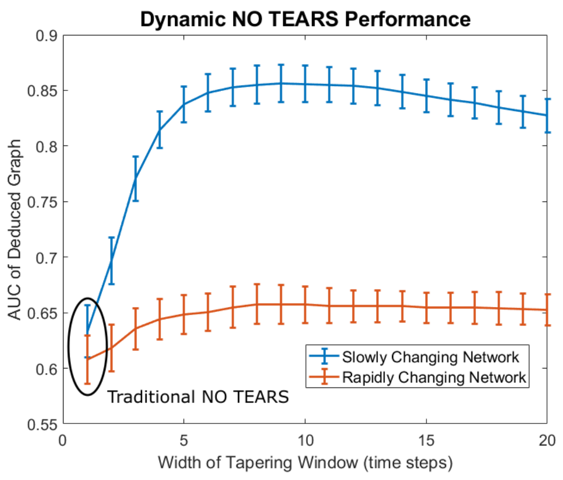

Figure 1. Accuracy of Temporally Regularized NOTEARS implementation. We implemented a dynamic Gaussian Bayesian network whose structure changed over time and assessed our algorithm's ability to deduce the correct network structure over each time point. We utilized two networks: one "slowly-changing" network where each edge had a 10% of changing each time step and one faster changing network where each edge had a 50% chance of changing. The x-axis shows how "temporally smooth" our hyperparameterization forces the deduced graphs to be. The far left points represent no temporal smoothness penalization which is identical to using the original NO TEARS implementation on each time point independently.

niques. We also think it would be useful to see how well our method predicts graph structures from real datasets that were not simulated.

References

Guo, F., Hanneke, S., fu, W., and Xing, E. Recovering temporally rewiring networks: A model-based approach. volume 227, pp. 321–328, 01 2007. doi: 10.1145/1273496.1273537.

Hanneke, S. and Xing, E. P. Discrete temporal models of social networks. In Airoldi, E., Blei, D. M., Fienberg, S. E., Goldenberg, A., Xing, E. P., and Zheng, A. X. (eds.), *Statistical Network Analysis: Models, Issues, and New Directions*, pp. 115–125, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-73133-7.

Kalofolias, V., Loukas, A., Thanou, D., and Frossard, P. Learning time varying graphs. pp. 2826–2830, 03 2017. doi: 10.1109/ICASSP.2017.7952672.

Kolar, M., Song, L., Ahmed, A., and Xing, E. Estimating time-varying networks. *The Annals of Applied Statistics*, 4, 12 2008. doi: 10.1214/09-AOAS308.

Koller, D. and Friedman, N. Probabilistic graphical models - principles and techniques. 2009.

Lee, H.-C., Danieletto, M., Miotto, R., Cherng, S. T., and Dudley, J. T. Scaling structural learning with NO-BEARS to infer causal transcriptome networks. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 25:391–402, 2020.

Lin, C. and Bar-Joseph, Z. Continuous-state HMMs for modeling time-series single-cell RNA-Seq data. *Bioinformatics*, 35(22):4707–4715, 04 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz296. URL `https://doi.org/10.1093/bioinformatics/btz296`.

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences*, 107(14):6286–6291, 2010. ISSN 0027-8424. doi: 10.1073/pnas.0913357107. URL `https://www.pnas.org/content/107/14/6286`.

Pamfil, R., Sriwattanaworachai, N., Desai, S., Pilgerstorfer, P., Beaumont, P., Georgatzis, K., and Aragam, B. Dynotears: Structure learning from time-series data. *ArXiv*, abs/2002.00498, 2020.

Pirgazi, J. and Khanteymoori, A. A robust gene regulatory network inference method base on kalman filter and linear regression. *PLoS ONE*, 13, 07 2018. doi: 10.1371/journal.pone.0200094.

Schaffter, T., Marbach, D., and Floreano, D. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16): 2263–2270, 06 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr373. URL `https://doi.org/10.1093/bioinformatics/btr373`.

Zheng, X., Aragam, B., Ravikumar, P., and Xing, E. P. DAGs with NO TEARS: Continuous Optimization for Structure Learning. 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. P. Learning Sparse Nonparametric DAGs. 2019.

Zhou, S., Lafferty, J., and Wasserman, L. Time varying undirected graphs. *Machine Learning*, 80, 03 2008. doi: 10.1007/s10994-010-5180-0.