DYNOTEARS: Structure Learning from Time-Series Data

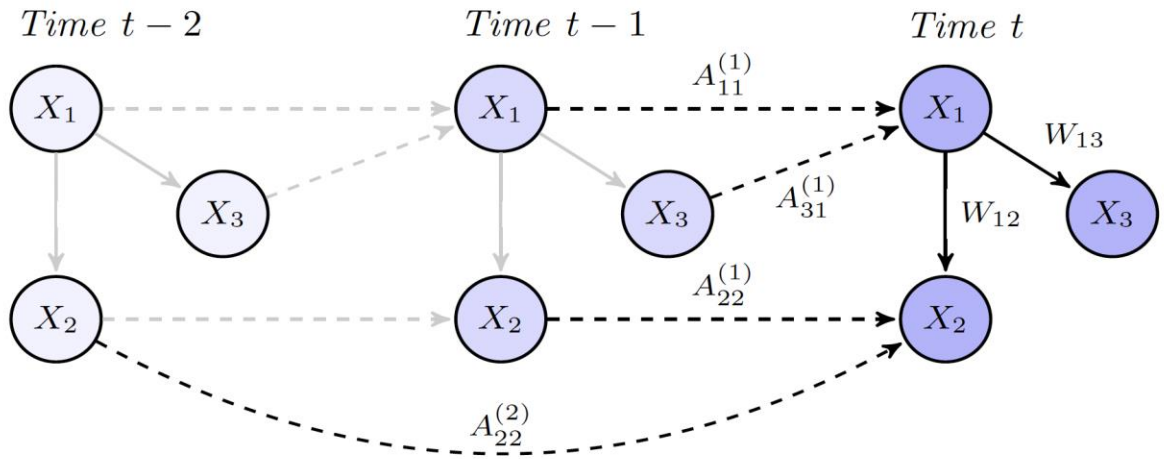
Daniele Montagnani

21 January 2024

Introduction

The Problem: Structure Learning for Dynamic Bayesian Networks

- What is structure learning?
- Static vs Dynamic
- Score-based vs Costraint-based



SOTA

- Heuristics to enforce acyclicality
- Learning relationships of differet types either independently or sequantially

Paper Contribution

- NOTEARS framework was introduced to deal with static settings
- Adapt it to deal with time series data
- Simultaneous learning of contemporaneous and time lagged dependencies

Formulation

M independent realisations of a stationary time series

Variables influenced by other variables at present time and by previous observations up to the **autoregressive order** p

Structural Equation Model:

$$\boldsymbol{X} = \boldsymbol{XW} + \boldsymbol{YA} + \boldsymbol{Z}$$

Observations: $X = [\dots | x_{m,t}^T | \dots]^T$
Time lagged observations: $Y = [Y_1 | \dots | Y_p]$
Inter Slice Adjacency Matrices: $A = [A_1^T | \dots | A_p^T]^T$
Errors: Z - respecting proper conditions for identifiability
let's say, standard Gaussian, i.e. $z_{m,t} \sim N(0, I)$

Optimisation I

$$\min_{W,A} \; f(W,A) = \frac{1}{2n}\left|\left|X - XW - YA\right|\right|_F^2 + \lambda_W \left|\left|W\right|\right|_1 + \lambda_A \left|\left|A\right|\right|_1 \quad \text{s.t.} \quad W \text{ is acyclic}$$

- Squared Error Loss
- l_1 regularization on W and A

How to Enforce Acyclicality in W?

From NOTEARS: require $h(W) = 0$, where $h(W) = tr\, e^{W \circ W} - d$

Obtain a measure with the following properties:

1. Null iff W is acyclic
2. Quantifies the "Dagness" of the graph
3. Smooth
4. Easy to compute, like its derivatives

Yes, but why?

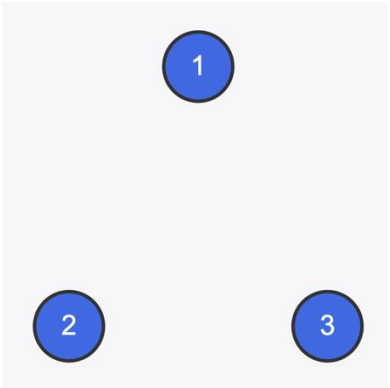
A beautiful math result

$$h(W) = tr\, e^{W \circ W} - d = 0 \quad \Rightarrow \quad \textbf{Acyclic Graph}$$

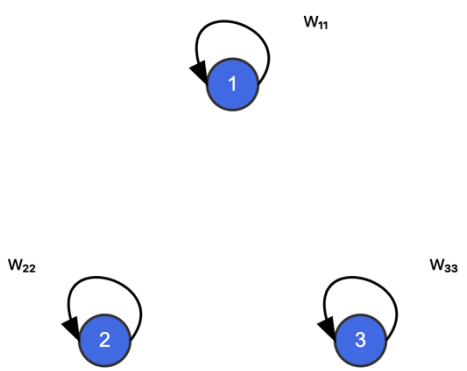
But why?

- Why exponential? $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots$
- Meaning of matrix multiplication and matrix powers
- Why trace?

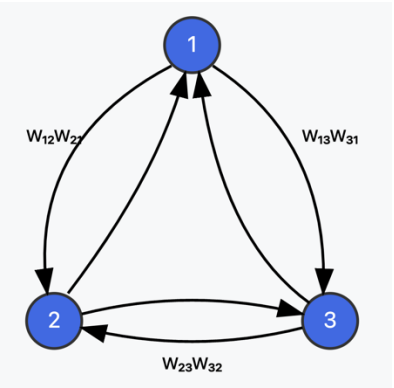
$tr(W^0) = tr(I)$



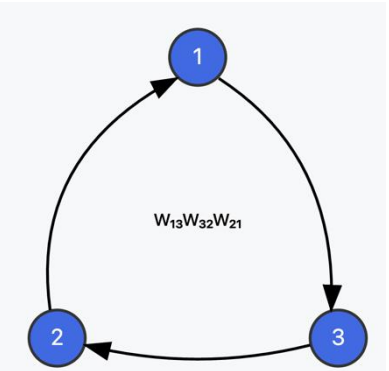
$tr(W)$



$tr\,(W^2)$



$tr\,(W^3)$



- Why the Hadamard product? Isn't contraction a problem?

Optimisation II

After having introduced the costraint $h(W) = 0$

Solve with the Augmented Lagrange Method

$$\alpha^{k+1} = \alpha^{k} + \rho h(x^{k+1})$$

Obtain a series of smooth augmented objectives of the form

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

Equivalent to a quadratic program with non-negative constraints

For numerical stability:
eliminate weights close to 0 by thresholding the entries of W and A

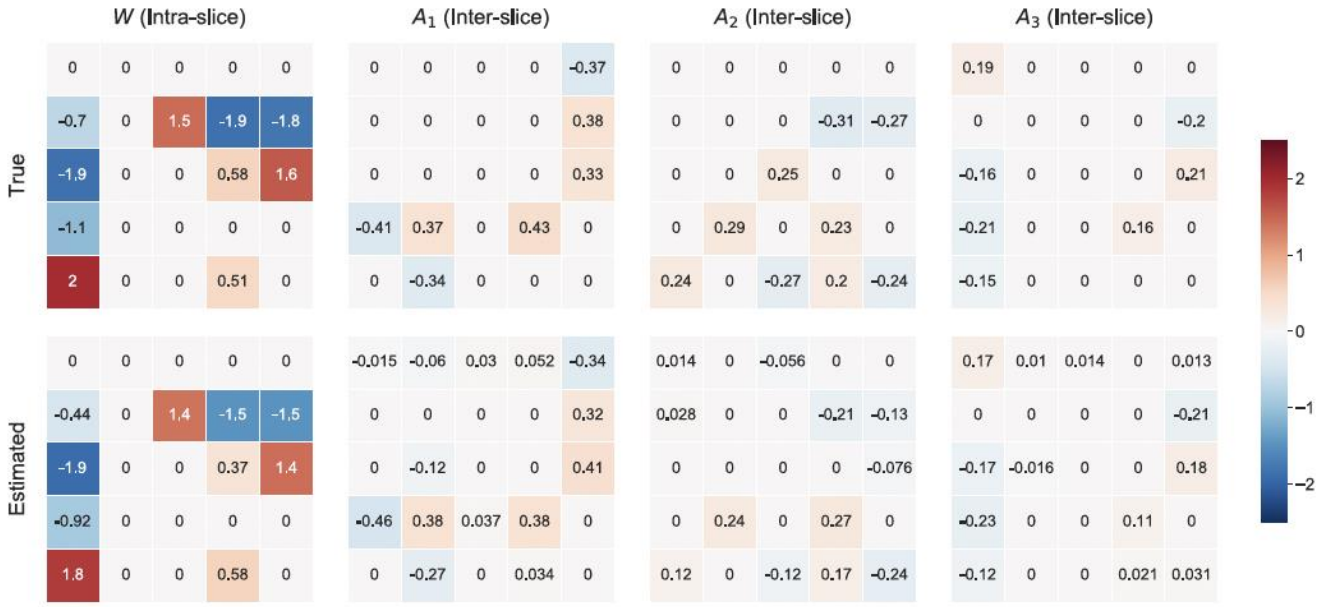
In the end, 4 total hyperparameters: $\lambda_W, \lambda_A, \tau_W, \tau_A$

Experiments on Synthetic Data

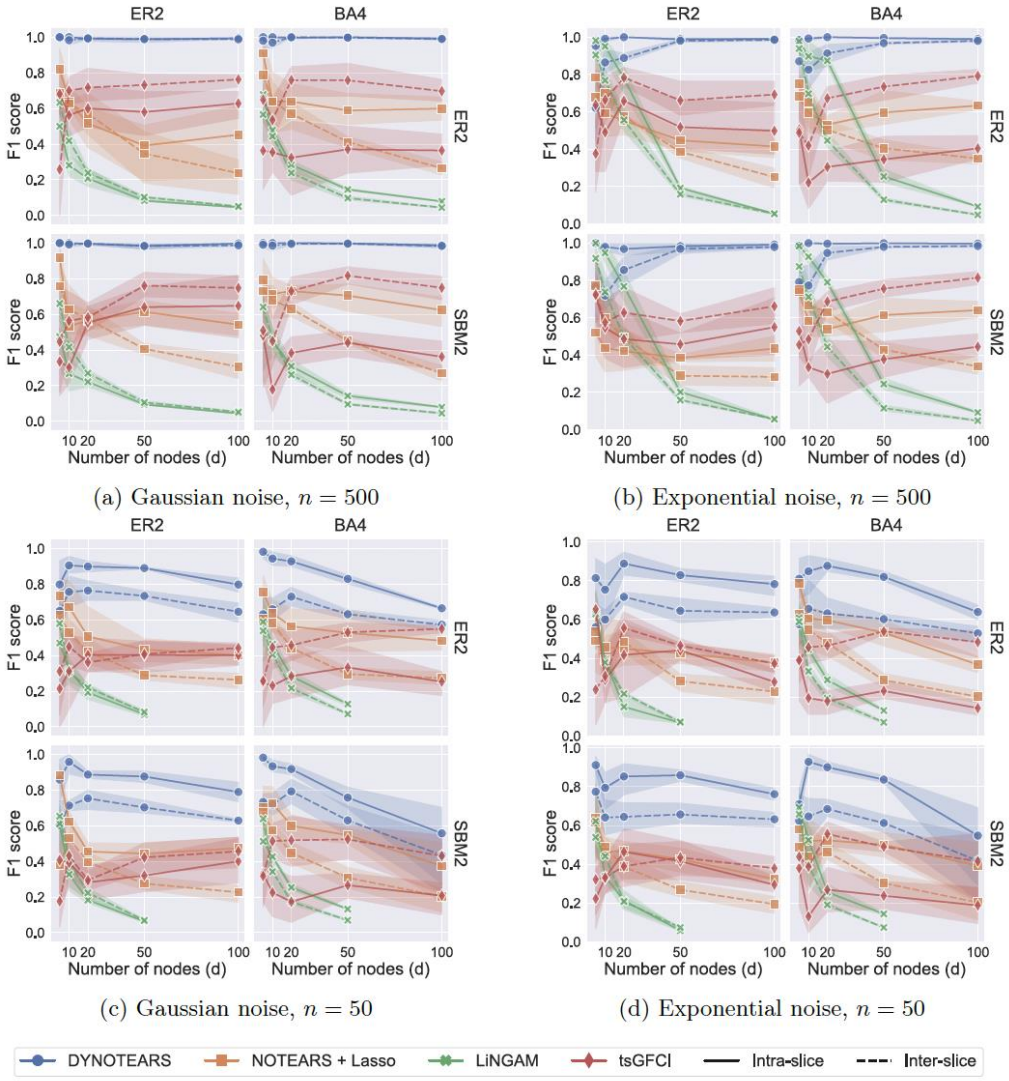
Simulation experiment:

1. Generate weigthed graphs G_A and G_W
2. Generate synthetic data X and Y from these graphs
3. Run and eval DYNOTEARS on the synthetic data
4. Inspect the estimated weights
5. Compare with benchmarks from other algorithms

Estimated Weights (Gaussian noise, n=500, d=5, p=3)



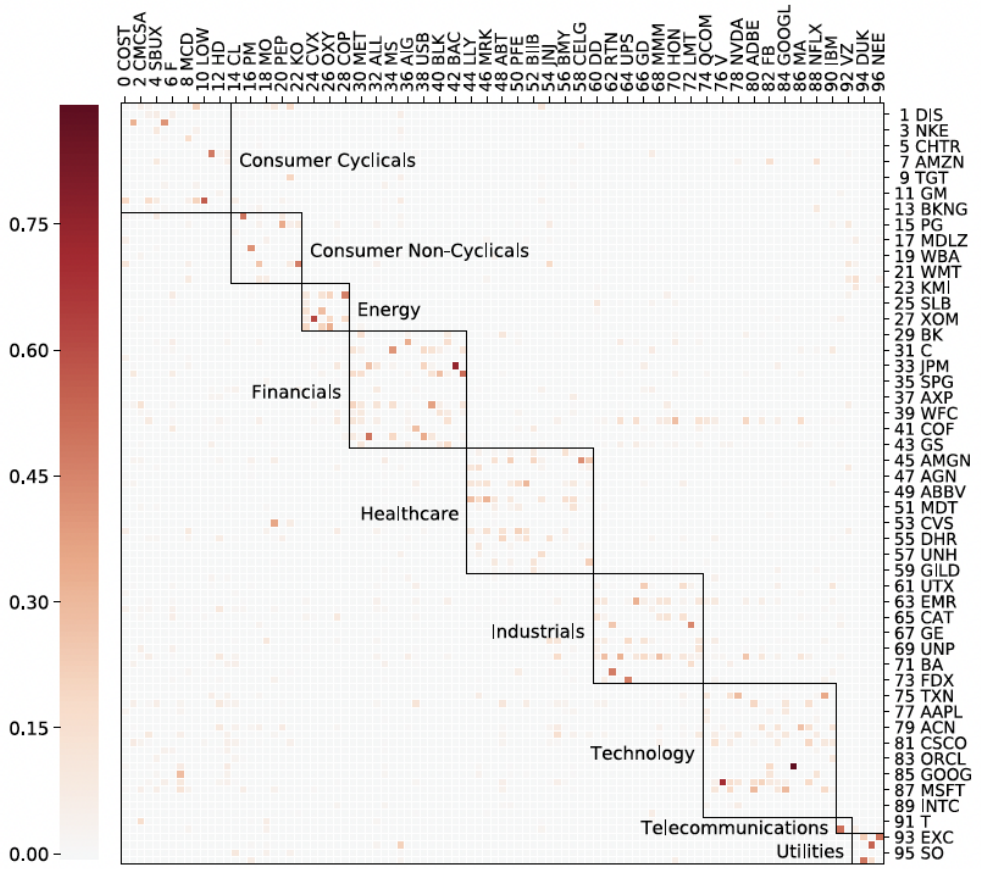
Performances against benchmarking algorithms



Applications – S&P100 and DREAM4 Datasets

S&P100

- Use Normalised log returns
- Grid search → Optimal Hyperparameters
- No inter-slice weights
- Community detection algorithm → sector relationship



DREAM4

- Learn gene regulatory networks
- Use gene expression data

- Compare with flexible non-parametric methods
- Understand validity of the linear model assumption

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

Conclusion

The core of the paper was the application of the NOTEARS framework:

- Extremely elegant approach
- Obtains an optimization-friendly condition from a tricky to check requirement

The model rests on the fundamental assumption of linearity:

- Need to test its validity in different contexts
- Future directions can investigate non-linear approaches
- For example, with neural networks
- An approach reminiscent of Graph Neural Networks