DYNAMIC NONLINEAR GAUSSIAN MODEL FOR INFERRING GRAPH STRUCTURES ON TIME SERIES

ABHINUV UPPAL — 2022 — CLAREMONT MCKENNA COLLEGE

Introduction

Graphs are incredibly powerful tools for analytics and machine learning, but the choice of how to impose a graph structure on data is often arbitrary. In this project, I propose a novel, dynamic method for inferring a graph structure on a collection of time series, where weighted edges encode information on how much the value of one node influences another in the system's time evolution.

Current methods in this field rely either on graph neural network (GNN) methods or static methods where edge weights are defined by functions of aggregating statistics are computed over a sliding window, such as the linear Pearson correlation.

I attempt to provide a new lens into this field with a maximum likelihood approach to a dynamic model, applicable to situations in which multiple time series are involved and there may be some nonlinear function governing their evolution. This alternative construction provides a new design choice for imposing graph structures on time series, where the edges explicitly encode information on how the nodes evolve at each point in time.

Equations and Methodology

Given a collection of time series indexed (1, 2, …, n) running from t = 1, 2, …, T, let $x_t \in \mathbb{R}^n$ be the state vector at time t, where the j-th entry corresponds to the value of series j. For an arbitrary, potentially nonlinear map $\mu: \mathbb{R}^n \to \mathbb{R}^n$ and noise vector η_t, assume the state vector evolves in time according to

$$x_{t+1} = \mu(Ax_t) + B\eta_t$$

For this project, I let $\mu(x) = \tanh(x)$ (element-wise) and $\eta_t \sim \mathcal{N}(0, I_n)$ is an n x 1 vector of i.i.d. Gaussian noise. It may be possible to infer μ from data with a neural network, but this is out of the scope of this project. The above equation is an affine transformation of a Gaussian distribution:

$$x_{t+1}|A, B, x_t \sim \mathcal{N}(\mu(Ax_t), BB^\top)$$

This can be used to derive a log-likelihood function:

$$\log(y) = -\frac{1}{2}\sum_{t=0}^{T-1} n\log(2\pi) + \log(|BB^\top|) + (x_{t+1} - \mu(Ax_t))^\top (BB^\top)^{-1}(x_{t+1} - \mu(Ax_t))$$

Jacobian matrices of log(y) with respect to A and B are defined element-wise by:

$$\frac{\partial \log y}{\partial A_{ij}} = -\sum_{t=0}^{T-1} x_{t,i}[\mu'(Ax_t)]_j[(BB^\top)^{-1}(x_{t+1} - \mu(Ax_t))]_j$$

$$\frac{\partial \log y}{\partial B_{ij}} = -TB_{ij}B_{jj} - \frac{1}{2}\sum_{t=0}^{T-1}\sum_{k=1}^{n}\sum_{l=1}^{n}(x_{t+1} - \mu(Ax_t))_k(x_{t+1} - \mu(Ax_t))_l\left[(BB^\top)^{-1}_{ki}[B^{-1}]_{jl} + (BB^\top)^{-1}_{il}[(B^\top)^{-1}]_{kj}\right].$$

Using the Jacobians, I perform simple, first-order gradient ascent with a learning rate of 10^{-4} in order to maximize the likelihood function on simulated data I generated from the process above. Due to computational limitations, I set $B = cI_n$, where c is a given constant determining the strength of i.i.d. noise in the system, and use numerical optimization solely to estimate the true value of A. I then determine the error incurred in this estimate by comparing the estimated value of A to the true value used to generate the data.

Results

The main two questions of the study are

1. Given different starting guesses for A, do they converge to the same estimates? This tests the likelihood function's convexity.
2. Are parameters getting closer to the true values as optimization continues? This tests the accuracy of the algorithm.

For all experiments, I set T = 1000, n = 10, and c = 1 unless otherwise stated. The code used is available at https://github.com/AbhiUppal/seniorthesis.

Figure 1: Standard Deviation of Post-Optimization Parameter Values

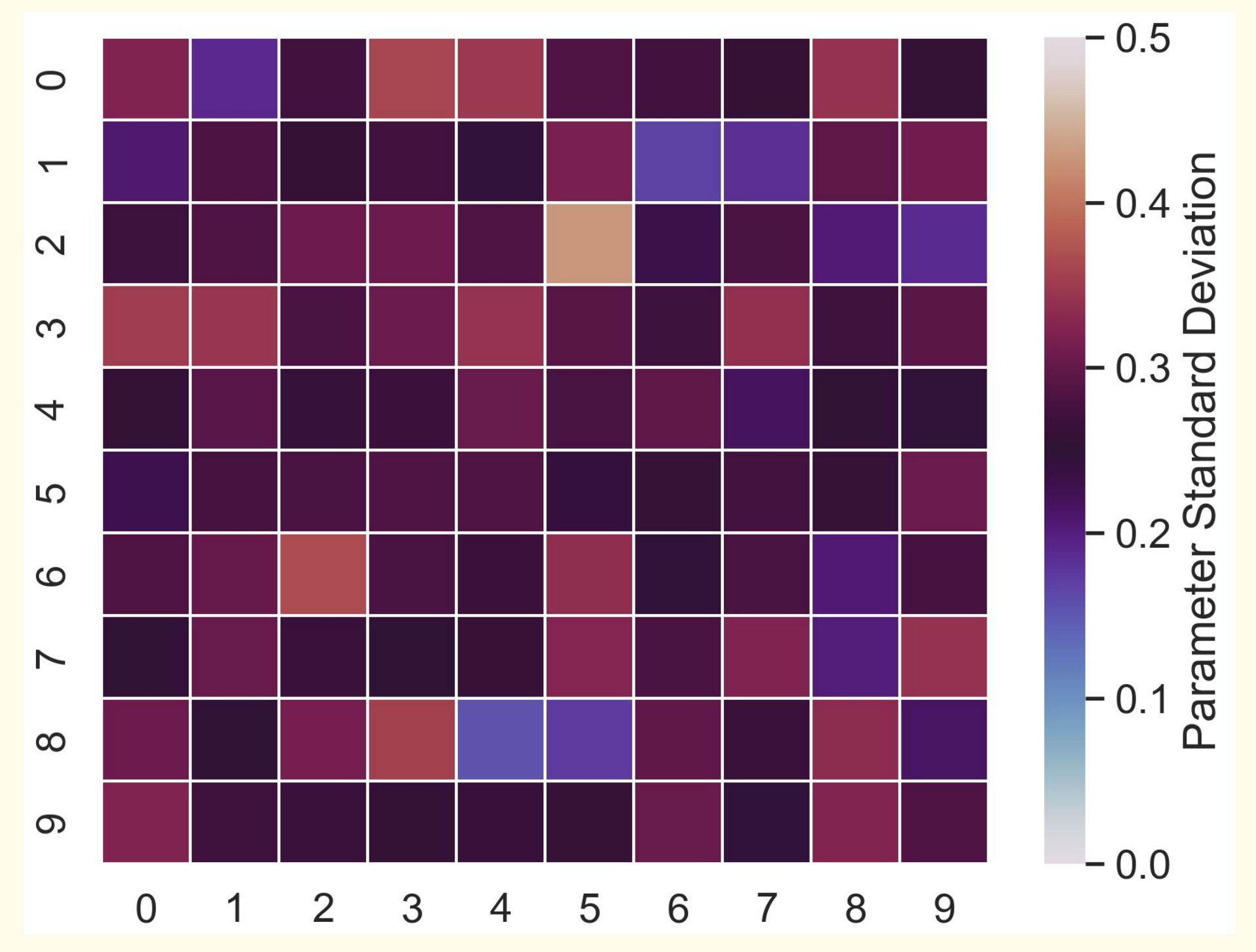

Figure 1 details the standard deviation of post-optimization parameters for 10 initial guesses performed on the same dataset. The relatively large values indicate a large dependence of the results of the optimization on the initial guess.

The large variability of results prompts an investigation into whether or not the optimization actually works.

Figure 2: Change in Error for Post-Optimization Parameter Values

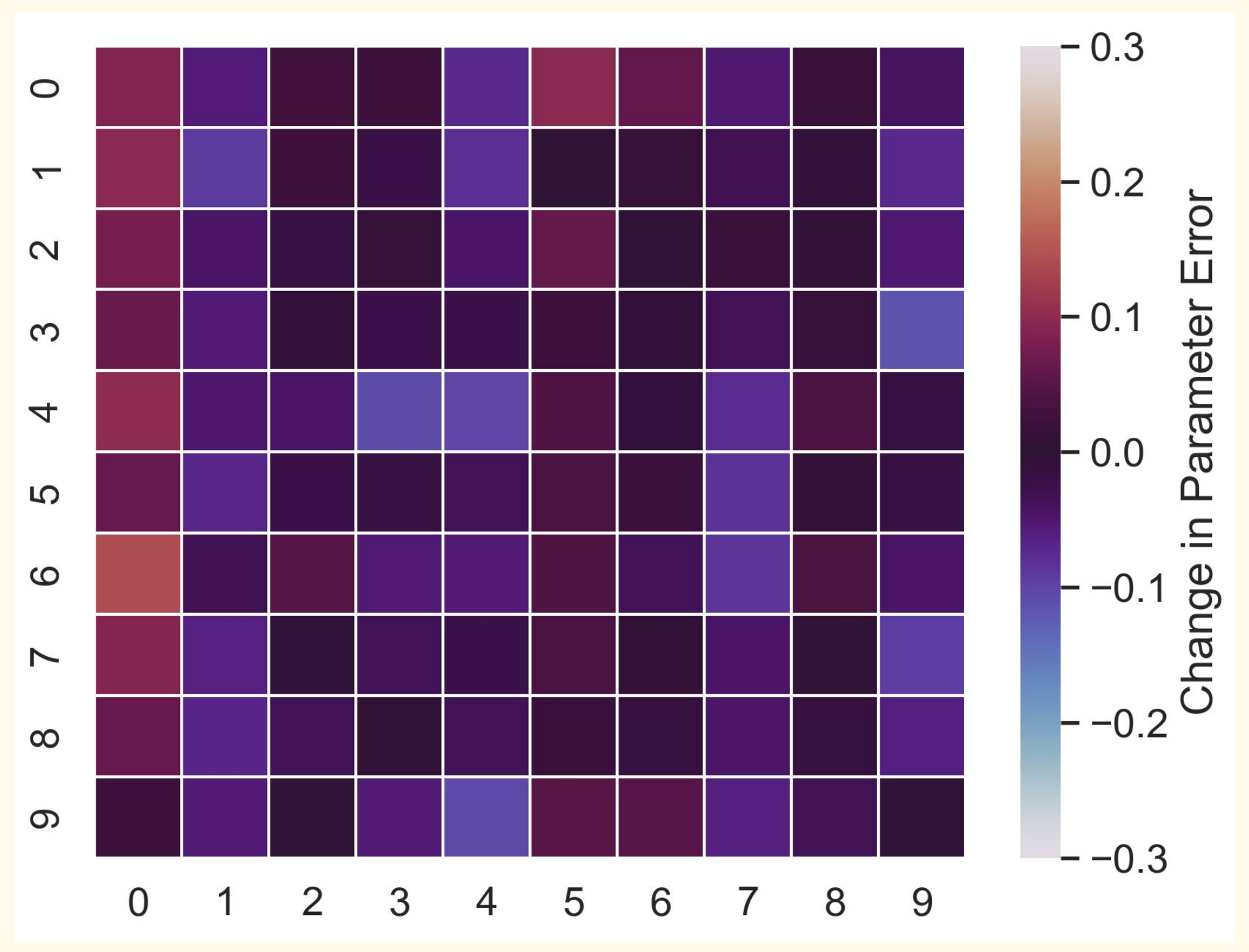

Figure 2 displays changes in each parameter's error post-optimization. The columnar pattern indicates that the errors are very likely not random—the values in each column represent the influence of the network on one node in particular. In general, each column tends to take the same color as a node's impact on itself (the diagonal elements).

I computed the mean number of improved parameters for multiple values of c.

Figure 3: Mean Number of Improved Parameters as a Function of Noise Prevalence.

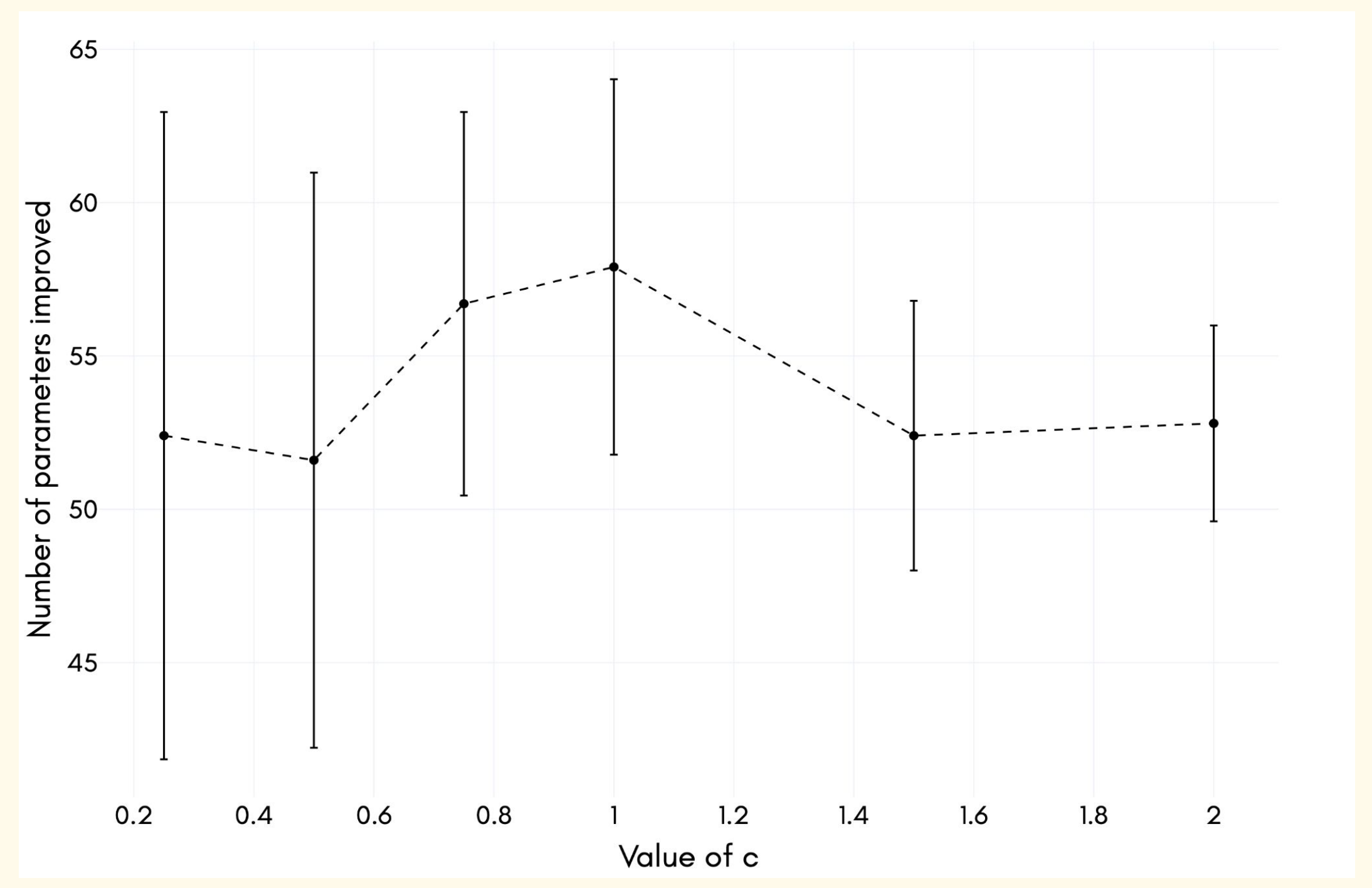

Figure 3 shows 95% confidence intervals for the mean number of parameters which were improved when the value c varied. Error bars show the margin of error of the mean for each value of c tested. 10 runs were executed for each value of c. A statistically significant improvement from random adjustment (50 improved parameters) occurs only for values of c = 0.75 and 1, indicating that the algorithm is unsuccessful in learning the adjacency matrix.

Conclusion and Discussion

The algorithm as a whole appears unsuccessful. The inconsistent results of repeated optimizations on the same dataset indicate that the objective function may be nonconvex, leading estimates to convergence to local maxima rather than the global maximum. This may be alleviated by nonconvex optimization methods, but this field is young and offers minimal guarantees of convergence.

From a modeling standpoint, attempting to learn 100 parameters on 1000 data points is highly prone to overfitting. Unfortunately, increasing T is difficult without significant compute. A possible workaround is to introduce L^1 regularization term to the objective function, enforcing A to be much more sparse than it is. This may sacrifice some accuracy, but could help capture some of the most important edges, especially since a complete graph may not be a useful abstraction in some modeling situations. Additionally, it could help the graphs generalize better out-of-sample.

Acknowledgements

I extend sincere gratitude towards Professor Sarah Marzen for assisting me in the development of the mathematical framework. Additionally, I thank Professor Adam Landsberg for his vital feedback in the revision process. This project was supervised by faculty from the Keck Science Department and the Robert Day School of Economics and Finance, from which I thank Professors Scot Gould and Julio Garín for their valuable guidance as readers. I also gratefully acknowledge Lauren Kula for her assistance in designing this poster.

Key References

- Jonathan Mei and Jose M. F. Moura. "Signal Processing on Graphs: Causal Modeling of Unstructured Data". In: *IEEE Transactions on Signal Processing* 65.8 (Apr. 2017), pp. 2077–2092. issn: 1941-0476. doi: 10.1109/tsp.2016.2634543. url: http://dx.doi.org/10.1109/TSP.2016.2634543.
- R.N. Mantegna. "Hierarchical structure in financial markets". In: *The European Physical Journal* B 11.1 (Sept. 1999), pp. 193–197. issn: 1434-6028. doi: 10.1007/s100510050929. url: http://dx.doi.org/10.1007/s100510050929.
- Antonio Ortega et al. "Graph Signal Processing: Overview, Challenges, and Applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828. doi: 10.1109/JPROC. 2018.2820126.
- K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Nov. 2012. url: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html.
- Thomas Kipf et al. *Neural Relational Inference for Interacting Systems.* 2018. arXiv: 1802.04687 [stat.ML].