Identification of gene regulatory network from gene expression time-course data

Najwa Laabid
100646115

Jan Sternagel
102177941

February 2024

1 Introduction

In this project, we deduce gene regulatory networks (GRNs) from gene expression data collected by Cantone et al. [2009]. The challenge in inferring GRNs lies in the complex and nonlinear nature of gene regulation. Recognizing that no singular method can perfectly address all aspects of this challenge, we explore three distinct approaches, each escalating in complexity. We share our code base in this GitHub Repository.

2 Methods

2.1 Relevance Networks

We initiate our exploration with coarse-grained approaches to modeling gene interactions. The goal is to identify all similarly behaving genes, and call an interaction if the strength of such a similarity is above a pre-determined threshold. Networks inferred in this way are called *relevance* networks. We note that this method generates an *undirected* network, necessitating the disregard of connection directions when evaluating this method. Various methods can be used to quantify the similarity between statistical variables, including correlation and mutual information. We explore both metrics in what follows.

Correlation The Pearson correlation coefficient is a measure of the linear correlation between two variables x and y. The metric takes a value between -1 and 1 inclusive, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. Equation 1 defines the correlation between x and y given a list of n pairs of observations, and with $\bar{x}$ and $\bar{y}$ the mean of each variable respectively

$$\operatorname{corr}(x,y) = \frac{\operatorname{cov}(x,y)}{\sqrt{\operatorname{cov}(x,x)\operatorname{cov}(y,y)}} \tag{1}$$

where $\operatorname{cov}(x,y) = \mathbb{E}\big[(x-\mathbb{E}[x])(y-\mathbb{E}[y])\big]$ is the covariance between variables x and y.

Mutual Information Mutual information measures the amount of uncertainty that is reduced from variable x by knowing variable y. It takes value 0 when the variables are entirely unrelated to each other, and it is maximal when variable y completely defines variable x. Formally, mutual information is defined as the Kullback-Liebler divergence between the joint distribution of x and y and the product of their marginals (i.e. the joint assuming independence between the two variables)

$$I(x|y) = D_{\mathrm{KL}}\Big[p(x,y)||p(x)p(y)\Big] = \int\int p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)dxdy \tag{2}$$

2.2 Linear ODE with a single integration step

A simple way to model the dynamics of the network is through a linear function of the form

$$\frac{d\mathbf{X}(t)}{dt} = \mathbf{W}\mathbf{X}(t) \tag{3}$$

where $\mathbf{X}(t)$ is a 5×1 vector containing the gene expression of each of the 5 genes x_i at time t. $\mathbf{W}$ is a weight matrix defining the rate of change of each gene as a linear transformation of the expression of all other genes at time t. We assume the dynamics do not change over time, i.e. $\mathbf{W}$ is time-independent. We do not consider the intercept term here since we assume all interactions to be determined solely by the genes considered in this system.

Euler solver and connection to linear regression One way to solve the dynamics model defined above is using a numerical approach called Euler integration (or solver). In its crudest form, the Euler integrator predicts the next step in the model by adding the rate of change to the current quantity

$$\begin{aligned}
\mathbf{X}(t+\Delta t) &= \mathbf{X}(t) + \frac{d\mathbf{X}(t)}{dt} * \Delta t && (4)\\
&= \mathbf{X}(t) + \mathbf{W}\,\mathbf{X}(t)\,\Delta t && (5)\\
&= (\mathbf{I} + \mathbf{W}\,\Delta t)\,\mathbf{X}(t) && (6)\\
&= \mathbf{W}'\,\mathbf{X}(t) && (7)
\end{aligned}$$

As can be seen from Equation 4, the single step integration simplifies to a linear regression model. This means that to try the crudest form of integration, we only need to fit a linear regression model to predict the gene quantities at time $t + \Delta t$ from measurements in time t.

We have one issue though: given the amount of data we have (measurements from 20 time steps) and the number of variable interactions we're considering (5 genes each with 5 potential regulators, meaning 25 model weights in total) our model suffers from *collinearity*. To solve this issue, we add Ridge regularization. See Appendix A for an overview of collinearity and regularization in this context.

2.3 Linear ODE with advanced integration

Now let us consider the same dynamics model as in Equation 3, but this time with an advanced integration method. One way to do this is by defining a linear ODE inside an automatic differentiation framework. This way, the integrated ODE returns a solution for the system at the following time-step, while the automatic differentiation framework takes care of optimizing the parameters of the ODE. Specifically, we use the `ODEINT` integrator from `torchdiffeq` to perform the integration and `PyTorch` as the auto-diff framework.

2.4 PySINDY library

`PySINDy` is a powerful Python library designed for the discovery of governing equations from data, particularly in the form of nonlinear ODEs de Silva et al. [2020]. Furthermore, it enables the integration of prior biological knowledge through inequality constraints, enhancing the model's expressivity and integration of prior information. At its core, `PySINDy` applies sparse regression techniques to identify the simplest possible differential equations that capture the system dynamics. The fundamental operation of `PySINDy` can be encapsulated by the following mathematical relationship:

$$\frac{d\mathbf{X}}{dt} \approx \Theta(\mathbf{X}(t))\mathbf{\Xi} \tag{8}$$

where $\frac{d\mathbf{X}}{dt}$ symbolizes the time derivative of the system state matrix $\mathbf{X}(t)$, $\Theta(\mathbf{X})$ denotes a library of candidate functions, and $\mathbf{\Xi}$ represents a sparse matrix containing coefficients that correspond to the significant features in $\Theta(\mathbf{X})$ which best characterize the system's behavior.

A diverse suite of optimizers is available within `PySINDy`, each offering distinct strategies for enforcing sparsity and incorporating domain-specific constraints. For a more comprehensive insight into the library's capabilities, we refer the reader to the detailed work by de Silva et al. [2020].

3 Experimental setup

Data We use the switch-off time series data collected by Cantone et al. [2009].

Models We consider the following models: 1) RN-Corr, the relevance network model with the correlation metric, 2) RN-MI, the relevance network model with the mutual information metric, 3) ODE-LR, for the linear regression with ridge regularization, 4) ODE-INT, for the linear ODE model with advanced integration, and 5) STSLQ, PySINDYs sequentially thresholded least squares optimizer.

Implementation details All models were implemented using `torch` on `Python`. The linear regression model used a single Linear layer. To integrate the ODE in the ODE-INT model, we used `odeint(.)` from the `torchdiffeq` package. For both ODE-LR and ODE-INT, we empirically set the hyper-parameter α to 0.01 for both models through a grid-search.

Evaluation metrics To evaluate the performance of the network, the study employs the true network depicted in Figure 5 in Cantone et al. [2009] as the benchmark model. For the models not inferring directionality (namely the relevance network models), we consider a prediction to be valid if there is an edge between two genes irrespective of its direction. For other methods, a prediction is only correct if it matches both the end points of the edge and its direction.

We use the Receiving Operator Curve (ROC) to visualize the trade-off between the false positive rate (fpr) and the true positive rate (tpr). An accompanying metric is the Area Under the Curve (AUC), which summarizes the trade-off between fpr and tpr for the model. A perfect classifier has an AUC=1, whereas a random model would have an AUC=0.5. We call a threshold optimal for calling an interaction if it manages to recover all 6 known interactions. We report the number of false positives recovered for the optimal threshold for all models. We also provide the predictions plots for our best model in Appendix D.

4 Results

All obtained results are listed in Table 4. ROC is depicted in a simplified form, with a detailed version available in the appendix at section C. The result for STLSQ are approximated over many runs.

Despite employing simpler undirected network models, the relevance network approaches underperformed in AUC compared to other methods. The best performing RN model in terms of AUC is RN-Mi. Though it strikes a balance between fpr and tpr that is better than random, the model is only able to recover all 6 true interactions when it considers all interactions to be true, which means that it is not useful for distinguishing true interactions in this context.

The linear ODE strategies yield superior outcomes, with ODE-INT achieving the highest AUC of 0.83, and the corresponding lowest number of false positives with the optimal threshold (min fp = 7). The linear regression model is the second best performing model with an AUC of 0.64, which is a surprisingly decent performance given the simplicity of the model and the complexity of the dynamics governing gene regulation. Despite its high AUC, ODE-INT only captures a coarse view of the dynamics as shown in the prediction plots in Appendix D.

STLSQ attained a lower AUC of approximately 0.55, despite the model's ability to fit more complex dynamics. This ill-fitness most likely hints to a lack of data to properly train the model. It is neverthless interesting to note the usefulness of the library for dynamics modeling using sparse differential equations.

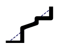
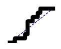
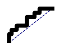
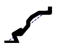
	RN-Corr	RN-Mi	ODE-LR-0.01	ODE-INT-0.01	STLSQ
Directed Graph	False		**True**		
AUC	0.59	0.62	0.64	**0.83**	$\sim$0.55
Optimal Threshold	0.12921	0.	0.02719	0.00131	0.008
Min. FP	14	19	13	7	17
ROC					

Table 1: Comparing the inferred network dynamics against the network presented in Cantone et al. [2009] as a classification task.

5 Discussion

Modeling the dynamics of the system as a linear function seems to lead to the best model fit, especially when using advanced integration methods. This means that the regulatory interactions between genes in this context can be approximated through rather simple functions, at least with the amount of data given in this experiment. To confirm this observation, more data would be necessary, especially from other experiments capturing different aspects of gene regulation, like the switch-on data shared by Cantone et al. [2009].

Other methods, including reverse-engineering approaches Cantone et al. [2009] and non-parametric function approximations Äijö and Lähdesmäki [2009], were evaluated on this dataset and yielded better results overall. This suggests that an ideal approach to this problem is one combining models with high expressivity coupled with a close integration of prior (or simplifying) assumptions about the nature of the interactions. Nonethless, it is interesting to note that a model that is simple to use like a linear ODE can capture a coarse approximation of the regulation dynamics, and could provide a simple and interesting baseline for more advanced methods.

6 Conclusion

We applied a number of methods to infer the gene regulatory network in yeast data. Our results showed that coarse-grained, non-directional methods perform poorly on this task, and that a model assuming a linear connection between all genes can approximate the interactions coarsely. More thorough evaluations of the models proposed here using data from other experiments and/or longer time series data is crucial to assess the generalization of these results.

References

Irene Cantone, Lucia Marucci, Francesco Iorio, Maria Aurelia Ricci, Vincenzo Belcastro, Mukesh Bansal, Stefania Santini, Mario di Bernardo, Diego di Bernardo, and Maria Pia Cosma. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137:172–181, 2009.

Brian M de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J Nathan Kutz, and Steven L Brunton. Pysindy: a python package for the sparse identification of nonlinear dynamics from data. *arXiv preprint arXiv:2004.08424*, 2020.

Tarmo Äijö and Harri Lähdesmäki. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22):2937–2944, 08 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp511.

A Multicollinearity and regularization

Multicollinearity happens when the regressors in a model are linearly dependent, which means there are infinitely many correct values for the parameters. One way this occurs is when we have more parameters than data. To illustrate this case, consider a simple linear model of the form $ax + b$ in 2 dimensions (i.e. x is a single dimensional variable and a and b are scalar values). Assume we have a single data point to estimate the parameters a and b, i.e. the number of data points is less than the number of parameters we are trying to estimate. This means we have one point in a Euclidean space, through which an infinite number of lines can pass, and we are trying to choose a single 'correct' line from this set. We clearly do not have enough information to reach this decision, no matter the statistical method we use to perform the estimation in practice. The same idea applies in higher dimensions, i.e. when we have a higher number of parameters than data points.

One way to bypass this issue in our case is by adding a *regularization term* forcing some of the weights to become 0. Intuitively, looking at our 2D example, the regularization term adds new constraints on the values of the parameters, which would help us 'choose' a line more easily.

$$\mathcal{L} = ||\mathbf{X}(t+1) - \hat{\mathbf{X}}||_2^2 + \alpha||W||_2^2 \tag{9}$$

where $\hat{\mathbf{X}}$ are the gene quantities at time $t + \Delta t$ estimated by our model, α is a hyperparameter controlling the strength of the regularization, and $||.||_2^2$ is the squared Euclidean norm. This regularization term is called $L2$ or *ridge* regularization. Another option is the $L1$ or *Lasso* regularizer, which only looks at the first norm of the weights.

B Relevance Networks

The relevance networks are derived from varying the threshold over the obtained correlation matrix and mutual information matrix shown in Fig. B.

Figure 1: Correlation Matrix (left) and Mutual Information Matrix (right)

C ROC curves for all models

We include ROC plots for all the models explored in this work.

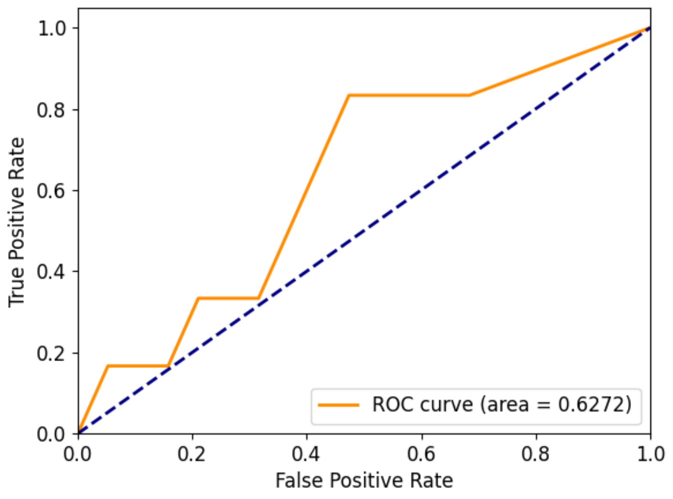


Figure 2: ROC for Relevance Network with mutual information.

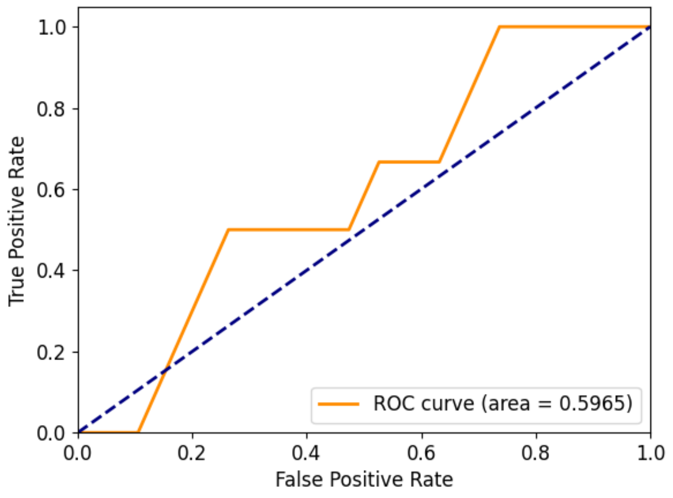


Figure 3: ROC for Relevance Network with correlation.

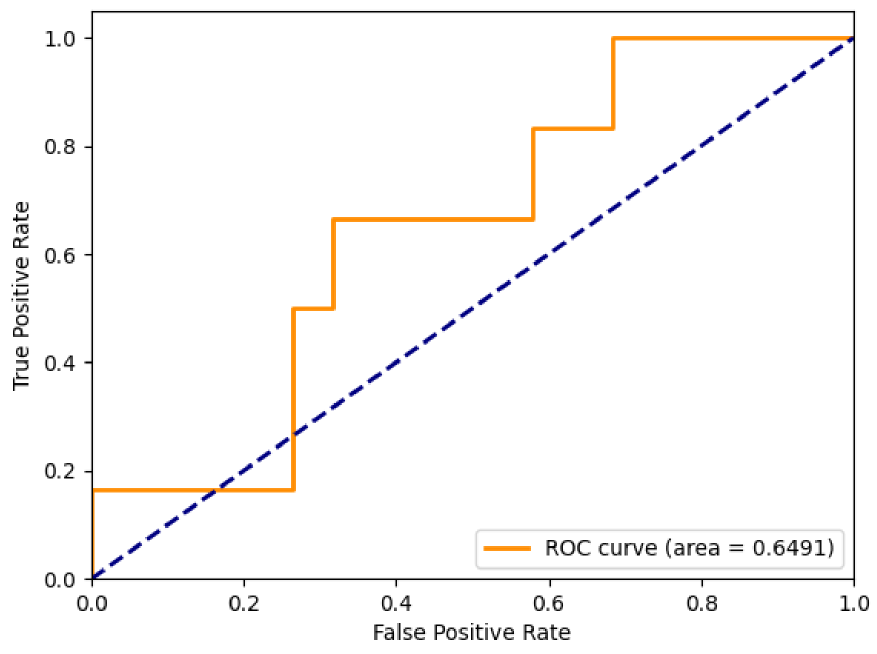


Figure 4: ROC for ODE-LR.

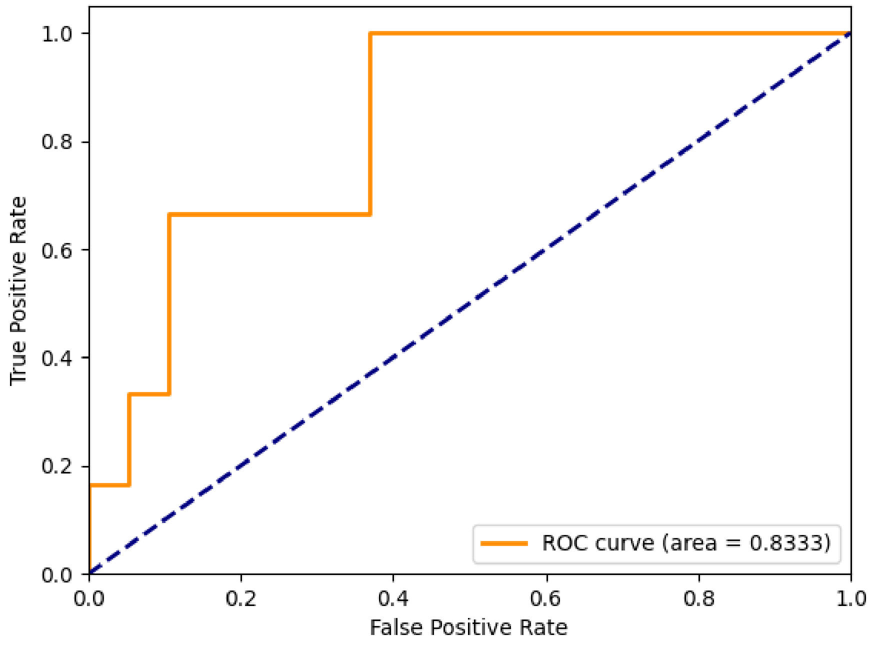


Figure 5: ROC for ODE-INT.

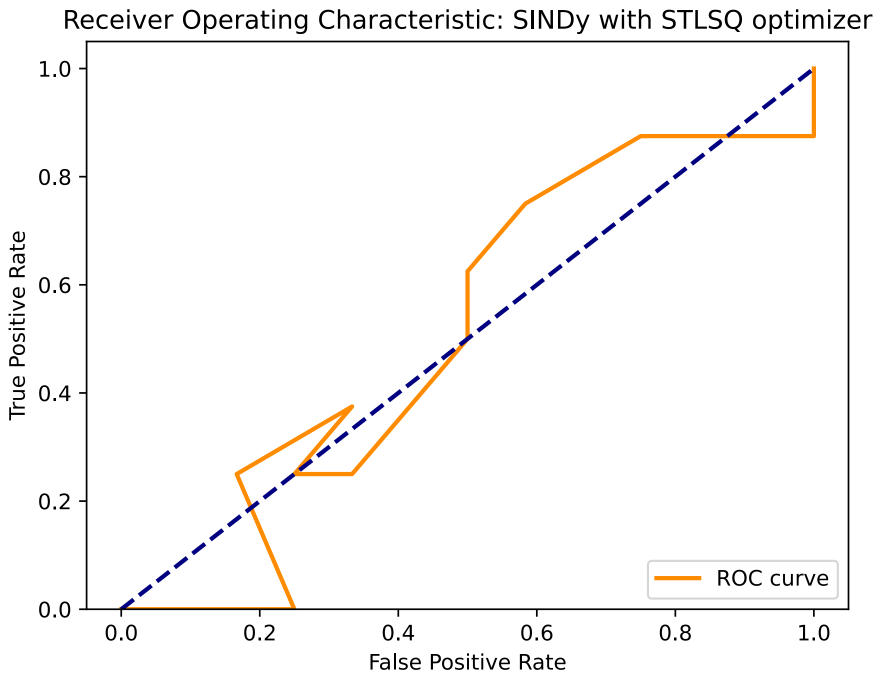


Figure 6: The STLSQ optimizer's local optimum at a 0.018 threshold results in a complex, non-monotonic ROC curve, rendering AUC calculation infeasible. However, the AUC is estimated to be greater than > 0.5, approximately ~ 0.55.

D Predictions using the Linear ODE model

We plot our best model's predictions compared to the ground-truth time series data for all genes. The plot highlights some limitations of the linear ODE system which only captures a coarse approximation of the true regulation dynamics.

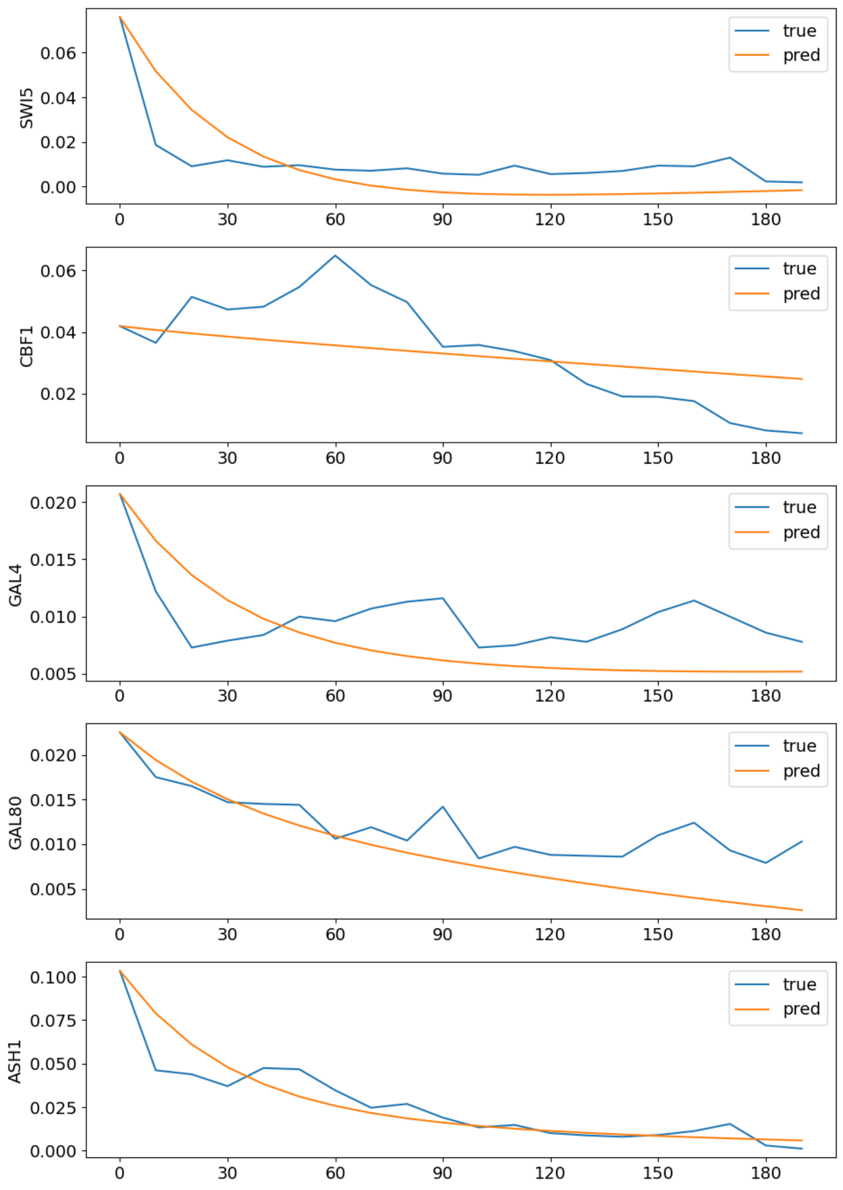


Figure 7: Predictions using our best model, ODE-INT.

E STLSQ Model Network

We include the graph learned by PySINDY in Figure E to showcase the library's ability to visualize the graphical information deduced from the learned ODEs.

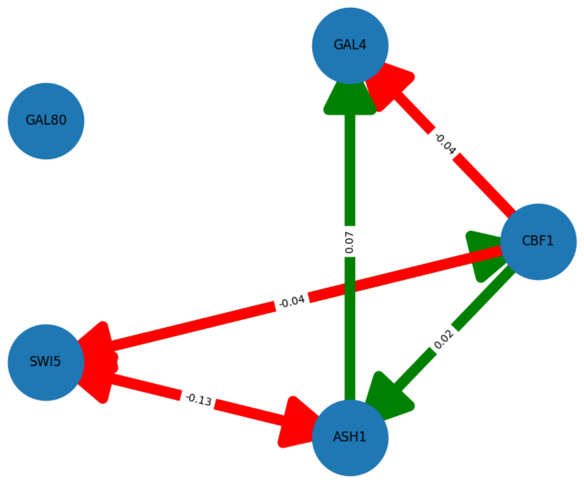


Figure 8: Visualized network graph of STLSQ optimizer at an threshold of 0.0182.