LETTERS IN BIOMATHEMATICS
An International Journal

Intercollegiate Biomathematics Alliance

RESEARCH ARTICLE

OPEN ACCESS

Building Model Prototypes from Time-Course Data

Alan Veliz-Cuba[a], Stephen Randal Voss[b], David Murrugarra[c]

[a]Department of Mathematics, University of Dayton, Dayton, OH; [b]Department of Neuroscience, Spinal Cord and Brain Injury Center, and Ambystoma Genetic Stock Center, University of Kentucky, Lexington, KY; [c]Department of Mathematics, University of Kentucky, Lexington, KY

ABSTRACT

A primary challenge in building predictive models from temporal data is selecting the appropriate model topology and the regulatory functions that describe the data. In this paper we introduce a method for building model prototypes. The method takes as input a collection of time course data. After network inference, we use our toolbox to simulate the model as a stochastic Boolean model. Our method provides a model that can qualitatively reproduce the patterns of the original data and can further be used for model analysis, making predictions, and designing interventions. We applied our method to a time-course, gene-expression data that were collected during salamander tail regeneration under control and intervention conditions. The inferred model captures important regulations that were previously validated in the research literature and gives novel interactions for future testing. The toolbox for inference and simulations is freely available at `github.com/alanavc/prototype-model`.

ARTICLE HISTORY
Received April 6, 2022
Accepted July 30, 2022

KEYWORDS
Network inference, Boolean networks, time course data, stochastic simulations

1 Introduction

The process of constructing discrete models from experimental data has several steps that have been studied in parallel. The main steps involved in this process are discretization, network inference, network selection, model interpolation, and deterministic/stochastic simulations (Dimitrova et al., 2010; Jarrah et al., 2007; Veliz-Cuba, 2012; Laubenbacher and Stigler, 2004; Stigler et al., 2007; Hinkelmann and Jarrah, 2012; Murrugarra and Laubenbacher, 2012; Wooten et al., 2021). Although some tools exist that address the global process (Dimitrova et al., 2011; Sun et al., 2020; Liang et al., 1998), either the code is unavailable, not editable, or not in a ready-to-use format.

Equation learning (EQ) methods for differential equation (DE) models start with a collection of time course data and then "recovers" the governing equations using a library of functions (Brunton et al., 2016; Lagergren et al., 2020). Many methods for EQ of DE models are based on formulating the inference problem as a parameter estimation problem that can be solved via optimization techniques (Brunton et al., 2016; Lagergren et al., 2020). Analogue methods for equation learning of discrete models that can learn both the network topology and the functions are still under development. Some of these existing methods can provide network candidates (i.e., possible wiring diagrams) that can explain the data. Other methods can provide candidate functions based on interpolating the data.

The main contribution of the paper is the combination of methods and the concrete toolbox that any user can use without familiarity with algebraic techniques. Our toolbox is modular, so that any step in the flowchart can be modified by the user without any restrictions. Importantly, it is also open-source and is freely available through a GitHub site. It works in Octave, so it is available for use in any operating system without the need of any license costs due to proprietary software. This makes our results fully reproducible.

The starting point of our method is experimental time-course data. Our focus is the construction of Boolean models, but we show with a toy model how our method also works for mixed-state models where variables can have different number of states or levels. As an application, we construct a model prototype using gene expression data for several time points which was collected during tail regeneration experiments in axolotls. We also use a synthetic network to illustrate the effect of data size, noise, and number of levels.

CONTACT David Murrugarra murrugarra@uky.edu

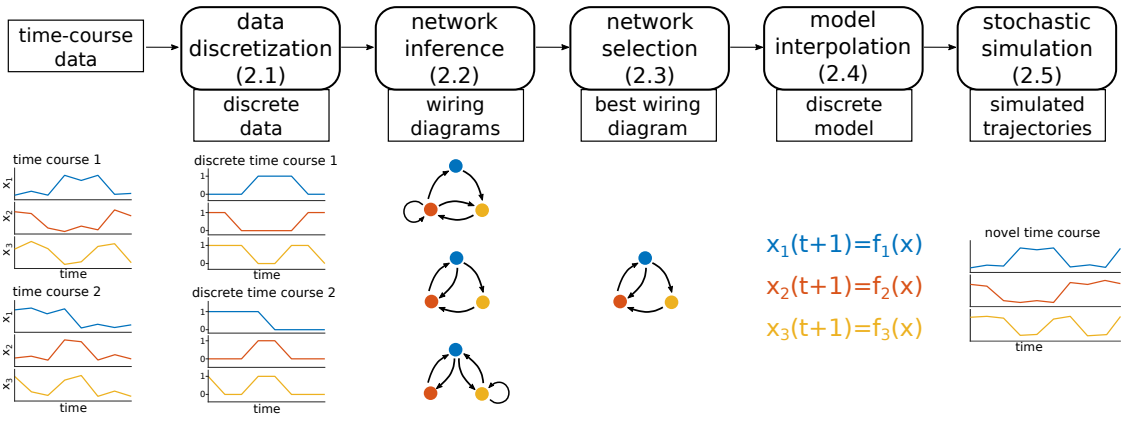


Figure 1: Flowchart showing the steps in model creation from data and the sections where each step is described. Starting from experimental time courses, we first transform the data into discrete values (in this case Boolean). Using algebraic techniques, we find wiring diagrams that explain the data. Each wiring diagram found will be consistent with all discrete time courses. We select the best wiring diagram and then find a discrete model that fits all the discrete data. This will result in a discrete model that can be simulated and compared with the original data. The model can also be run with new initial conditions or for longer time to create novel time courses that can be used to make predictions.

2 Methods

We will use "network" to refer to the correct wiring diagram and set of functions, and "model" to refer to the wiring diagram and set of functions that are obtained using data generated by the network.

Here we describe the methods for model selection (i.e., wiring diagram and regulatory functions) and the framework for simulations. We assume that we are given time courses of the form $s^1 \to s^2 \to \cdots \to s^r$, where $s^i = (s_1^i, \ldots, s_n^i) \in S = S_1 \times \cdots \times S_n$. Here S_i is a finite set of all the values that the i-th variable can take. Note that if $S_i = \{0, 1\}$, then we have a Boolean model.

Example 2.1. *To illustrate the methods, we use an example with the following four time courses.*

(1) $(0.1, 1.1, 1.9, 0.9, 0.2) \to (0.0, 0.2, 0.2, 0.1, 0.1) \to (0.0, 1.1, 0.1, 1.9, 2.1)$

(2) $(1.9, 0.1, 0.9, 0.1, 0.0) \to (0.9, 1.1, 0.1, 1.9, 2.1) \to (1.1, 0.9, 0.1, 1.9, 2.0)$

(3) $(0.2, 1.1, 1.9, 0.9, 1.1) \to (0.1, 0.0, 0.2, 0.1, 0.1)$

(4) $(0.1, 0.9, 2.1, 1.1, 2.1) \to (0.2, 0.1, 0.2, 0.1, 1.1)$

2.1 Discretization

We implemented a simple discretization method based on binning data by dividing the range of the data into equally spaced regions. The time courses suggest that the number of levels for variables x_1, x_2, x_3, x_4, x_5, are 3, 2, 3, 3, 3, respectively. For example, by plotting the values of x_1 and x_2 for each trajectory (Figure 2), we see that x_1 has 3 distinctive levels and x_2 has 2 distinctive levels. For x_1, all values below the dotted line will be mapped to 0 (low); all values between the dotted and dashed lines will get mapped to 1 (medium); and all values above the dashed line will get mapped to 2 (high). For x_2, all values below the dotted line will be mapped to 0 (low); and all values above the dotted line will get mapped to 1 (high).

Then, the discrete time courses are given below.

(1) $01210 \to 00000 \to 01022$

(2) $20100 \to 11022 \to 11022$

(3) $01211 \to 00000$

(4) $01212 \to 00001$

In this case $S = \{0, 1, 2\} \times \{0, 1\} \times \{0, 1, 2\} \times \{0, 1, 2\} \times \{0, 1, 2\}$.

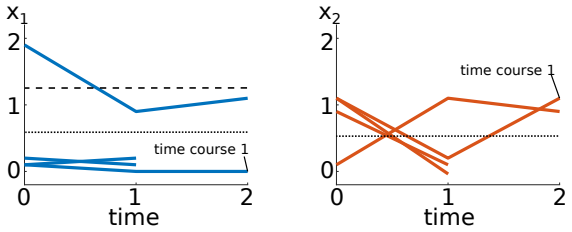


Figure 2: Values of x_1 and x_2 for the time courses. Variable x_1 can be considered as having 3 levels, whereas variable x_2 has 2 levels. The dashed lines show how the range of the data can be divided into regions (3 regions for x_1 and 2 for x_2), which will determine the discretization.

Table 1: Partial information for example.

x	$f(x)$
01210	00000
00000	01022
20100	11022
11022	11022
01211	00000
01212	00001

2.2 Network inference

To find the wiring diagrams that are consistent with a collection of time courses of the form $s^1 \to s^2 \to \cdots \to s^r$ we use the algebraic framework introduced by Veliz-Cuba (2012). This framework takes partial information about the evolution of a network $s \to f(s)$ and returns all the minimal wiring diagrams that are consistent with the data. This approach guarantees that for each minimal wiring diagram there exists a model that fits the data such that each interaction is activation or inhibition (Veliz-Cuba, 2012) .

To use the framework of Veliz-Cuba (2012), we first note that each time course $s^1 \to s^2 \to \cdots \to s^r$ implies that $s^{j+1} = f(s^j)$ for $j = 1, \ldots, r-1$, where f is the network one is trying to infer. This results in a set $D \subseteq S$ such that $f(s)$ is known for every $s \in D$. That is, D is the set of inputs for which we know the outputs.

Example 2.2. *In Example 2.1, $D = \{01210, 00000, 20100, 11022, 01211, 01212\}$. Then, the partial information we have is given in the Table 1. Then, using the algebraic techniques of Veliz-Cuba (2012), we can find all minimal wiring diagrams that are consistent with the data. For each variable x_i in the network, the algebraic framework returns $W_1, \ldots, W_k$, where each W_j is a minimal set of inputs for variable i. For our example we obtain Table 2.*

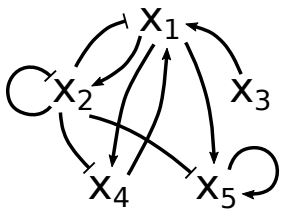
By selecting one wiring diagram for each x_i, we obtain a (global) wiring diagram that is consistent with the data. For example, if we select $\{x_2^-, x_3^+, x_4^+\}$ for x_1, $\{x_1^+, x_2^-\}$ for x_2, $\{\}$ for x_3, $\{x_1^+, x_2^-\}$ for x_4, and $\{x_1^+, x_2^-, x_5^+\}$ for x_5, we obtain the wiring diagram shown in Figure 3. To compare different wiring diagrams we can use the adjacency matrix representation.

2.3 Wiring diagram selection

The network inference described in Section 2.2 could return several minimal network candidates for each variable. That is, for a given time course data, there might be several models that explain the data and that are minimal. The method will return

Table 2: Minimal wiring diagrams. The +/− superscripts indicate activation/inhibition. For example, the set $\{x_2^-, x_3^+, x_4^+\}$ indicates that one way to explain the data is for x_2 to be an inhibitor of x_1 and x_3 and x_4 to be activators of x_1. By choosing one set for each variable, one obtains a wiring diagram that is consistent with the data. Note that no variable affects x_3 (i.e., constant function).

x_i	Minimal wiring diagrams for x_i
x_1	$\{x_1^+\}, \{x_2^-, x_3^+, x_4^+\}$
x_2	$\{x_3^-\}, \{x_2^-, x_4^+\}, \{x_1^+, x_4^-\}, \{x_1^+, x_2^-\}$
x_3	$\{\}$
x_4	$\{x_3^-\}, \{x_2^-, x_4^+\}, \{x_1^+, x_4^-\}, \{x_1^+, x_2^-\}$
x_5	$\{x_3^-, x_5^+\}, \{x_2^-, x_4^+, x_5^+\}, \{x_1^+, x_4^-, x_5^+\}, \{x_1^+, x_2^-, x_5^+\}$



$$\begin{pmatrix} 0 & -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3: Example of wiring diagram consistent with the data. Left: Wiring diagram. Right: Adjacency matrix representation.

Table 3: Frequencies of interactions on minimal wiring diagrams. The parameter q^+_{ji} (resp. q^-_{ji}) represents the frequency of regulator x^+_j (resp. x^-_j) in the minimal wiring diagrams of x_i. For instance, for variable x_1 in the first row, x^+_1 appears in one out of two wiring diagrams, therefore $q^+_{11} = 1/2$.

x_i	Frequencies of activations and inhibitions	# of WDs
x_1	$q^+_{11} = 1/2, q^-_{21} = 1/2, q^+_{31} = 1/2, q^+_{41} = 1/2$	2
x_2	$q^+_{12} = 2/4, q^-_{22} = 2/4, q^-_{32} = 1/4, q^+_{42} = 1/4, q^-_{42} = 1/4$	4
x_3	NA	0
x_4	$q^+_{14} = 2/4, q^-_{24} = 2/4, q^-_{34} = 1/4, q^-_{44} = 1/4, q^+_{44} = 1/4$	4
x_5	$q^+_{15} = 2/4, q^-_{25} = 2/4, q^-_{35} = 1/4, q^+_{45} = 1/4, q^-_{45} = 1/4, q^+_{55} = 4/4$	4

all candidate wiring diagrams. In order to select one model out of all possible options, we calculate the "best wiring diagram" by including only the most frequent interactions from the wiring diagrams found. For each variable, x_i, we quantified the frequency q^+_{ji} of positive interactions $x_j \longrightarrow x_i$ across all possible wiring diagrams and the frequency q^-_{ji} of negative interaction $x_j \longdashv x_i$ across all possible wiring diagrams for all $j = 1, \ldots, n$. That is, the parameter q^+_{ji} (resp. q^-_{ji}) represents the frequency of regulator x^+_j (resp. x^-_j) in the minimal wiring diagrams of x_i (see Example 2.3 and Table 3 for additional details). Then we construct an adjacency matrix W^* by considering the interactions with a frequency above certain threshold τ. If conflicts arise (that is, when $q^-_{ji} = q^+_{ji}$ for some j), then we discard those interactions. Subsequently, for each row of W^*, say W^*_i, we calculate the distance with each possible wiring diagram of x_i (these are represented as rows). Finally, we construct an adjacency matrix W with rows corresponding to the rows with minimum distances.

Example 2.3. *For the network in Example 2.1, we calculated the frequency of the interactions; see Table 3. For instance, for variable x_1 in the first row of Table 3, x^+_1 appears in one out of two wiring diagrams (see Table 2), therefore $q^+_{11} = 1/2$. Then, we computed an adjacency matrix W^* by including the interactions with a frequency above the threshold $\tau = 1/5$. We discarded conflicting interactions (i.e., the cases where $q^-_{ji} = q^+_{ji}$).*

$$W^* = \begin{pmatrix} 1 & -1 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 \end{pmatrix}$$

Then, for each row of W^*, say W^*_i, we calculate the distance with each possible wiring diagram of x_i (these are represented as rows). Then we construct an adjacency matrix with rows corresponding to the rows with minimum distances. Then, the matrix after the distance calculations is:

$$W = \begin{pmatrix} 0 & -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

The reason for why we take a distance approach is because there might not be a truth table satisfying the matrix W^* but there is certainly one for W as shown in Example 2.2.

2.4 Fitting model to data

After one wiring diagram has been selected from the family of minimal wiring diagrams, we proceed to construct a function that fits the data. Although there are known formulas for interpolation, we are interested in *monotone* interpolation, that is, we

Table 4: Partial information for variable x_4 with wiring diagram $\{x_1^+, x_2^-\}$.

(x_1, x_2)	output
01	0
00	2
20	2
11	2
01	0
01	0

Table 5: Incomplete truth table for variable x_4 with wiring diagram $\{x_1^+, x_2^-\}$ and the corresponding construction of a function for all inputs.

(x_1, x_2)	output	$h(x_1, x_2)$
00	2	2
01	0	0
10	?	2
11	2	2
20	2	2
21	?	2

need to find a model that not only fits the data, but one whose signs of interaction match the wiring diagram selected.

We illustrate our approach with wiring diagram $\{x_1^+, x_2^-\}$ for variable x_4. Since it is guaranteed that there is a monotone function $h(x_1, x_2)$ that fits the data for variable x_4 (Veliz-Cuba, 2012), we consider Table 4 with only x_1 and x_2 in the first column (inputs) and only x_4 in the second column (output).

We now rewrite this table as a truth table by ordering the inputs lexicographically ($x_1 \in \{0,1,2\}, x_2 \in \{0,1\}$), where some entries are unknown, Table 5.

To fill in the table, we use the fact that the function increases with respect to x_1 and decreases with respect to x_2. For example, since $h(2,1) \geq h(1,1) = 2$, it follows that $h(2,1) = 2$. Similarly, since $2 = h(0,0) \leq h(1,0) \leq h(2,0) = 2$, it follows that $h(1,0) = 2$. In this way, we obtain the value of the missing entries. This process can be done for all wiring diagrams and for all variables. Since the existence of a wiring diagram guarantees that there is at least one suitable function that fits the data, this is always possible (Veliz-Cuba, 2012). To guarantee that the fitting is unique, we implemented the algorithmic construction from Lemma 2.4 from Veliz-Cuba (2012).

2.5 Stochastic framework

For the simulations we will use the stochastic framework introduced by Murrugarra et al. (2012) referred to as Stochastic Discrete Dynamical Systems (SDDS). This framework is a natural extension of Boolean networks and is an appropriate setup to model the effect of intrinsic noise on network dynamics. Consider the discrete variables $x_1, \ldots, x_n$ that can take values in finite sets $S_1, \ldots, S_n$, respectively. Let $S = S_1 \times \cdots \times S_n$ be the Cartesian product. An *SDDS* in the variables $x_1, \ldots, x_n$ is a collection of n triplets

$$F = \{f_i, p_i^{\uparrow}, p_i^{\downarrow}\}_{i=1}^{n}$$

where

- $f_i \colon S \to S_i$ is the update function for x_i, for all $i = 1, \ldots, n$.
- $p_i^{\uparrow} \in [0,1]$ is the activation propensity.
- $p_i^{\downarrow} \in [0,1]$ is the degradation propensity.

The stochasticity originates from the propensity parameters $p_i^{\uparrow}$ and $p_i^{\downarrow}$, which should be interpreted as follows: If there would be an activation of x_k at the next time step, i.e., if $s_1, s_2 \in S_k$ with $s_1 < s_2$ and $x_k(t) = s_1$, and $f_k(x_1(t), \ldots, x_n(t)) = s_2$, then $x_k(t+1) = s_2$ with probability $p_i^{\uparrow}$. The degradation probability $p_i^{\downarrow}$ is defined similarly. SDDS can be represented as a Markov chain by specifying its transition matrix in the following way. For each variable x_i, $i = 1, \ldots, n$, the probability of changing its value is given by

$$Prob(x_i \to f_i(x)) = \begin{cases} p_i^{\uparrow}, & \text{if } x_i < f_i(x), \\ p_i^{\downarrow}, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x), \end{cases}$$

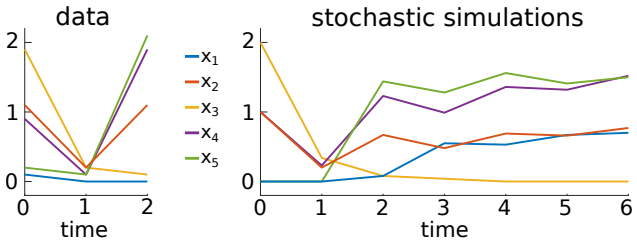


Figure 4: Comparison between data (only first time course shown) and stochastic simulations. Using the discretization of the initial condition of the data, 01210, we can use the model obtained to simulate the system for any arbitrary number of steps. Stochastic simulations are the average of 100 realizations, so they may take non discrete values between 0 and 2.

and the probability of maintaining its current value is given by

$$Prob(x_i \to x_i) = \begin{cases} 1 - p_i^{\uparrow}, & \text{if } x_i < f_i(x), \\ 1 - p_i^{\downarrow}, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x). \end{cases}$$

Let $x, y \in S$. The transition from x to y is given by

$$a_{xy} = \prod_{i=1}^{n} Prob(x_i \to y_i). \tag{1}$$

Notice that $Prob(x_i \to y_i) = 0$ for all $y_i \notin \{x_i, f_i(x)\}$.

The stochastic framework is implemented in the toolbox to give the user with simulation options including the deterministic case. By setting all propensities equal to 1, one obtains a deterministic model. Alternatively, setting all propensities equal to 0.9 gives a 90% chance of using the regulatory function for each node and a 10% chance of keeping the current state. Likewise, setting all propensities equal to 0.5 gives a 50% chance of using the regulatory function for each node and a 50% chance of keeping the current state value. Furthermore, one could use the parameter estimation techniques for computing the propensity parameters of SDDS that have been presented by Murrugarra et al. (2016).

3 Applications: Gene Expression Data from Experiments in Axolotls

In this section we apply our method to a time-course, gene expression data that were collected during salamander (axolotls – *Ambystoma mexicanum*) tail regeneration under control and intervention conditions. Chemicals that inhibit cell-signaling activities are used as intervention agents to block tail regeneration and alter gene expression. Modeling gene interactions can provide confirmatory and novel information for developing hypotheses about the actions of cell-signaling molecules and transcription factors that orchestrate tissue regeneration.

Using our method, we generated a Boolean model for a set of 10 genes that were expressed differently during axolotl tail regeneration under control and Wnt C59 treatment, a chemical that blocks the secretion of Wnt signaling molecules from cells (Ponomareva et al., 2015). We note that the Wnt C59 intervention in that study inhibited tail regeneration. Seven of the genes are ligands (*Areg*, *Fgf9*, *Bmp2*, *Inhbb*, and *Wnt5a*) or negative feedback regulators (*Dusp6*, *Nradd*) of cell signaling pathways, *Sp7* is a bone-specific transcription factor, *Hapln3* is a cell adhesion molecule and *Phlda2* is an intracellular protein. We label these genes using the following variables:

$$\begin{aligned} &x_1 = Areg, && x_2 = Phlda2, && x_3 = Fgf9, && x_4 = Bmp2, && x_5 = Nradd, \\ &x_6 = Hapln3, && x_7 = Sp7, && x_8 = Wnt5a, && x_9 = Inhbb, && x_{10} = Dusp6. \end{aligned} \tag{2}$$

In Figure 5 we show the wiring diagrams obtained using our method for both conditions, control and intervention. These wiring diagrams present gene-by-gene interactions for the given gene expression data set. For the control case, *Wnt5a* represents a key node in the network; this Wnt signaling ligand is predicted to activate ligands that function in BMP (*Bmp2*), FGF (*Fgf9*), and TGFβ (*Inhbb*) pathways, and transcriptional regulation of bone formation (Sp7) (Hojo and Ohba, 2022), consistent with Wnt signaling playing a central, integrative role in regeneration (Wehner et al., 2014) (Figure 5 a). Additionally, for the control case, we note that the well-established inhibitory effect of Dusp6 on FGF signaling is captured by this wiring diagram (Li et al., 2007), and it also implicates *Wnt5a* as an inhibitor of *Areg* during regeneration. By comparing the intervention with the control

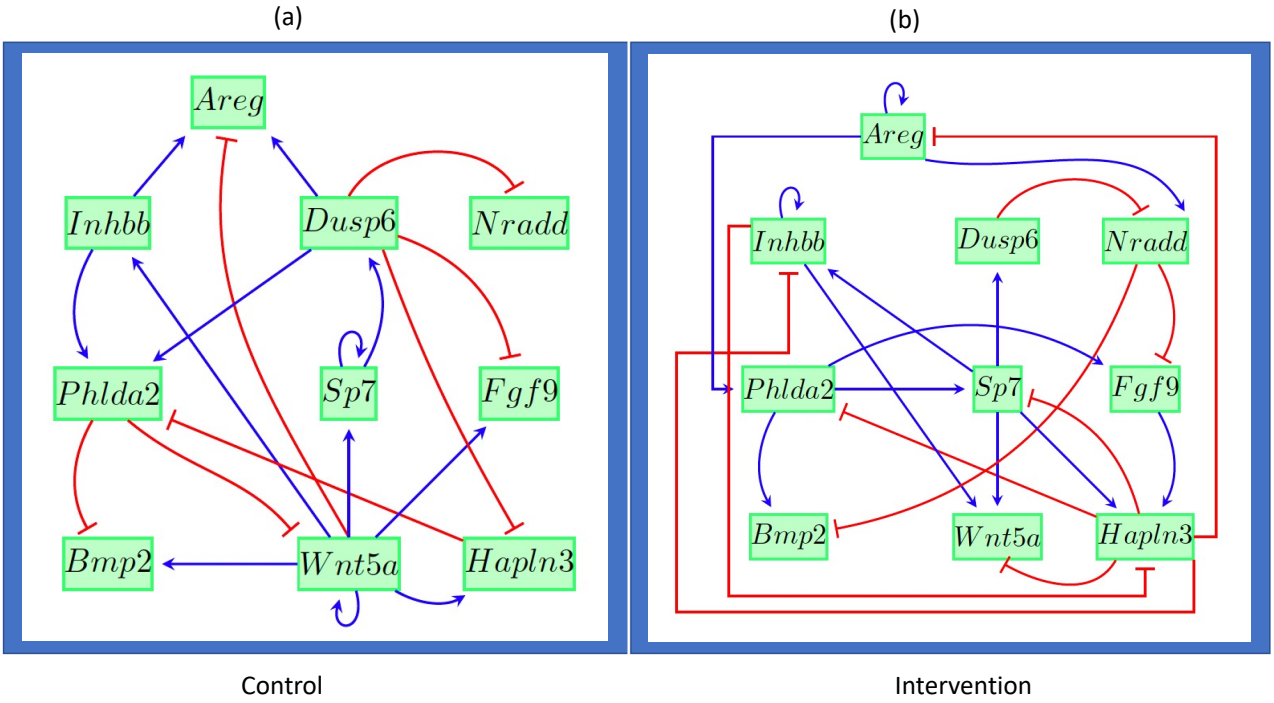


Figure 5: Wiring diagrams for the genes in Equation (2) for both conditions: (a) control and (b) intervention. Blue edges represent activation while red edges inhibition.

case, we see that Wnt C59 eliminated all of the *Wnt5a* activating edges and the inhibitory edge to *Areg*. A new activating edge from *Areg* to *Nradd* suggests a novel hypothesis that Wnt C59 blockade of Wnt ligand secretion indirectly (via Nradd) inhibits the transcription of BMP and FGF pathway ligands that are required for tissue regeneration (Wehner and Weidinger, 2015).

To further validate this model we compare the experimental data versus the simulations that are shown in Figures 6 and 7. These figures were obtained from 100 runs. Simulations using the framework SDDS (Murrugarra et al., 2012) were performed initializing the system at the initial state 1100000001. This initialization represents a discretized version of the actual data at time 0. For the simulations in Figures 6 and 7 we used propensities equal to 0.9 for all variables. To assess the quality of the predictions, we use the Mean Squared Error (MSE) between the discretized data and simulated trajectories (see Appendix D in the Supplemental Materials for details of the MSE). We also generated simulation plots using propensities equal to 0.5 for all variables, see Figures 1 and 2 of the Supplemental Materials. From the MSE values, it can be seen that the simulated trajectories with propensities equal to 0.9 give better fits of the discretized data. The propensity values can further be optimized using the method of Murrugarra et al. (2016). This model can further be used to attractor analysis, control, modularity, etc. However, the main result from this application is the potential novel interactions that can be experimentally tested.

4 Effect of Data Size, Noise, Number of Levels, and Threshold Value

Since our toolbox consists in the combination of several methods/algorithms, any advantages or disadvantages of these will affect the performance of the model created. We explore some of these effects using synthetic networks so that we can compare the model constructed with the original network.

4.1 Effect of data size

Here we illustrate the effect of data size using a synthetic example given by the Boolean network $f = (f_1, \ldots, f_5)$, where

$$f_1 = \neg x_3 \vee x_5 \qquad f_2 = x_1 \wedge x_4 \qquad f_3 = \neg x_2 \qquad f_4 = \neg x_3 \wedge x_5 \qquad f_5 = x_4 \tag{3}$$

We will generate synthetic data using this Boolean network. Then, *using the data only*, we will see if the following novel trajectories can be predicted: 01111 $\longrightarrow$ 10001 $\longrightarrow$ 10110 $\longrightarrow$ 01101 $\longrightarrow$ 10000 and 00100 $\longrightarrow$ 00100 (that is, 00100 is a steady state). That is, we will initialize the system at 00100 and 01111 in the model predicted by our toolbox. The results are summarized

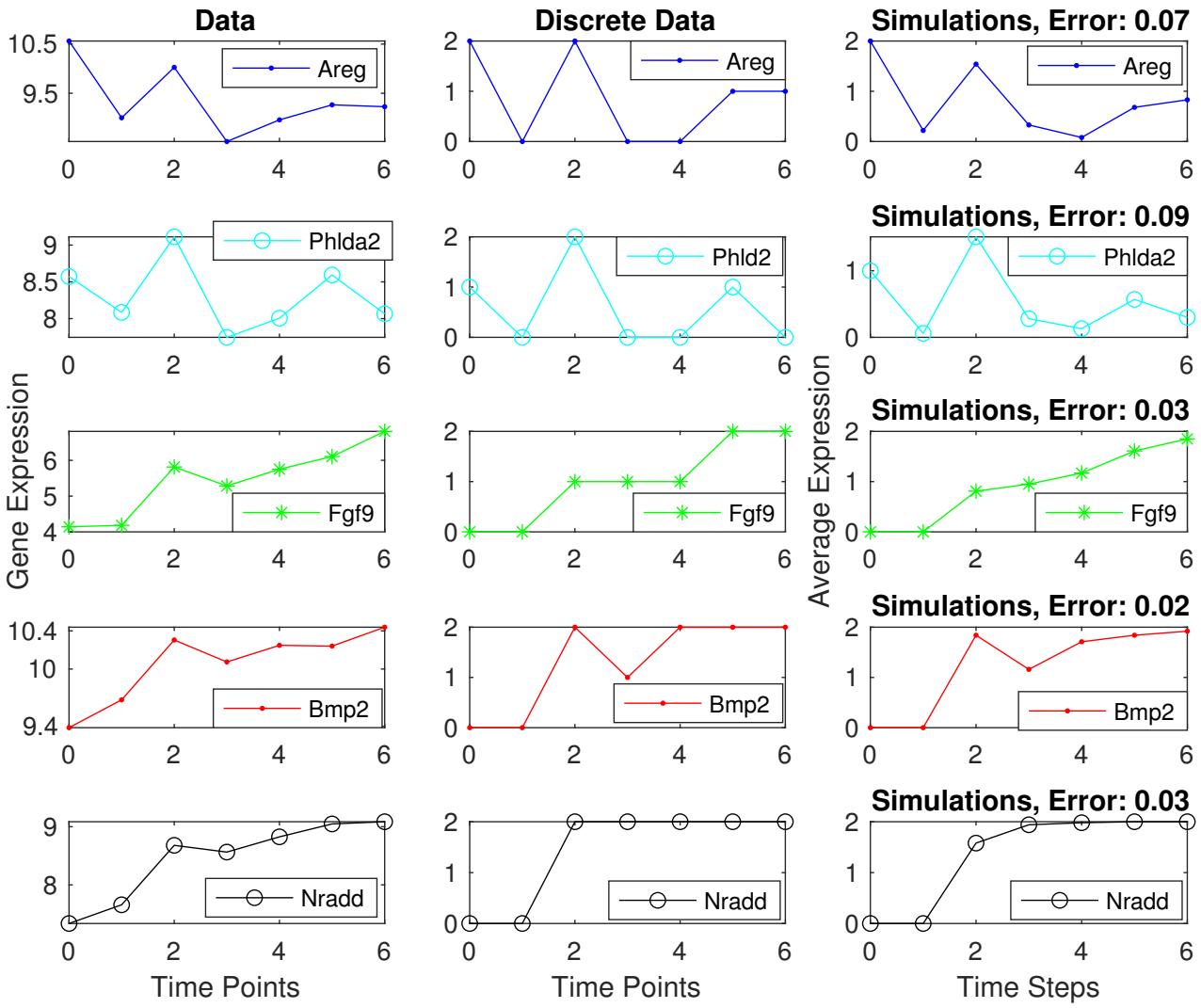


Figure 6: Gene expression data, discretized data, and simulations of the first five genes in Equation (2). The plots in the left panel are the experimental data, the ones in the middle are the discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.9 and the mean squared error is between the discretized data and simulated trajectories.

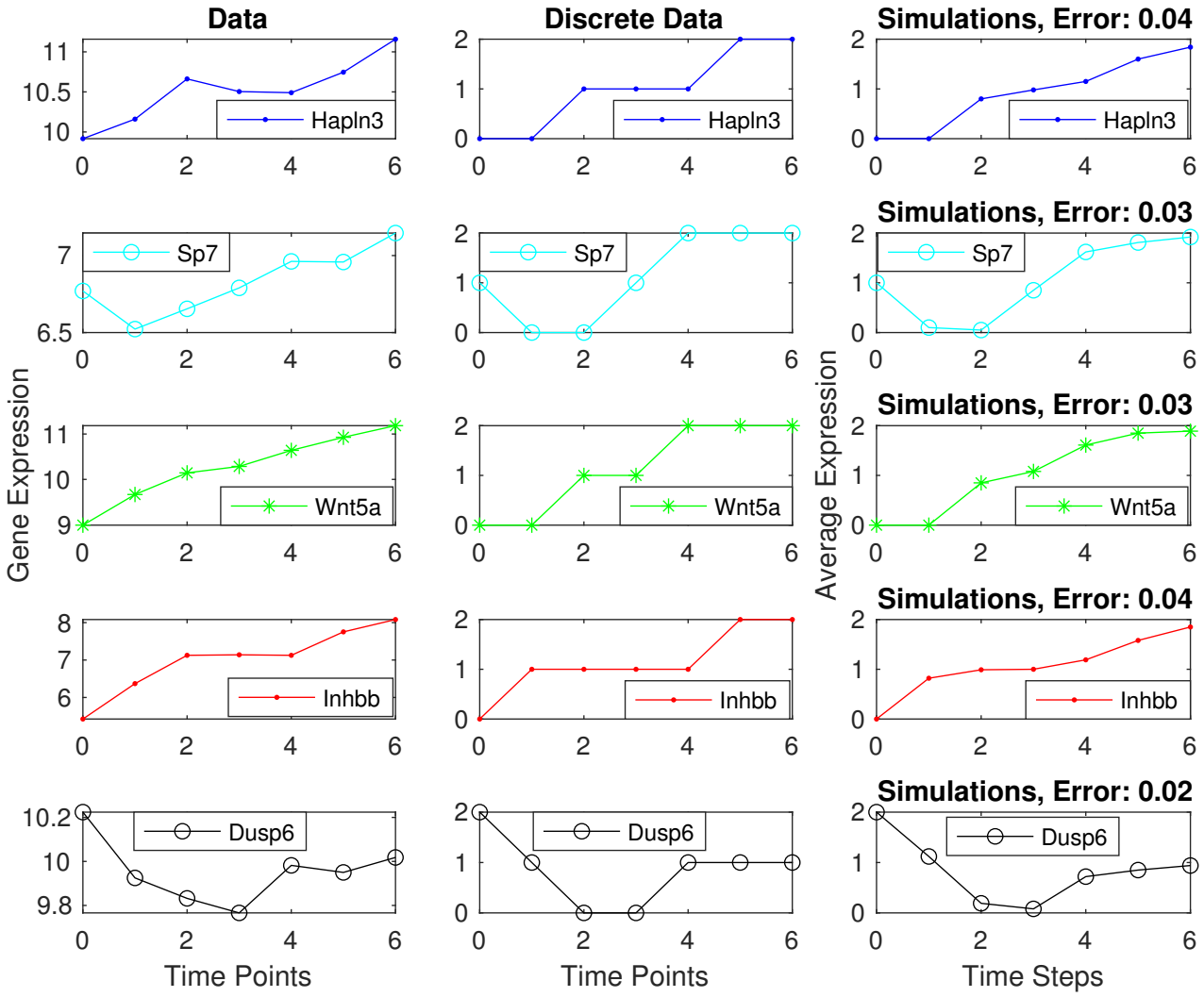


Figure 7: Gene expression data, discretized data, and stochastic simulations of the last five genes in Equation (2). The plots in the left panel are experimental data, the ones in the middle are discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.9 and the mean squared error is between the discretized data and simulated trajectories.

Table 6: Effect of increasing the data size. $T_1 = \{11011 \to 11011\}$, $T_2 = \{11010 \to 11001 \to 10010\}$, $T_3 = \{11110 \to 01001 \to 10010 \to 11101\}$, $T_4 = \{01011 \to 10011 \to 11111 \to 11001\}$. Transitions predicted correctly are indicated by bold arrows.

Data used	Novel trajectories predicted
T_1, T_2	$01111 \to 11011 \to 11011 \to 11011 \to 11011, 00100 \to 10000$
T_1, T_2, T_3	$01111 \to 01011 \to 11011 \to 11011 \to 11011, 00100 \to 00100$
T_1, T_2, T_3, T_4	$01111 \to 10001 \to 10110 \to 01101 \to 10000, 00100 \to 00100$

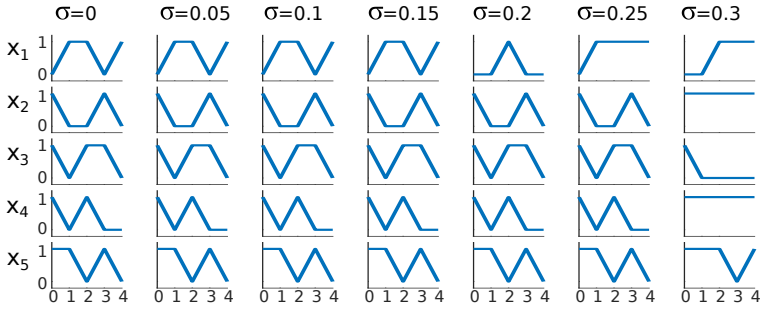


Figure 8: Effect of noise on predicted model. For small noise, the qualitative behavior is maintained, but as noise increases, the predicted model loses its predictive power.

in Table 6. To illustrate the effect of the number of data points clearer, we do not consider stochasticity in this subsection (all propensities are set equal to 1).

First, we start with the synthetic time series $T_1 = \{11011 \to 11011\}$ and $T_2 = \{11010 \to 11001 \to 10010\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 11011 \to 11011 \to 11011 \to 11011$ and $00100 \to 10000$. In this case the data was not enough to recover the trajectories.

Second, we start with the synthetic time series T_1, T_2, and also $T_3 = \{11110 \to 01001 \to 10010 \to 11101\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 01011 \to 11011 \to 11011 \to 11011$ and $00100 \to 00100$. In this case we see that with more data still the trajectory for 01111 was not recovered, but the model created by the toolbox correctly predicted that 11011 is a steady state.

Third, we start with the synthetic time series T_1, T_2, T_3, and also $T_4 = \{01011 \to 10011 \to 11111 \to 11001\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 10001 \to 10110 \to 01101 \to 10000$ and $00100 \to 00100$. That is, with the data given, the model predicted by the toolbox was able to correctly reproduce the novel trajectories. Note that T_1, T_2, T_3, T_4 represent 9 out of the $2^5 = 32$ possible transitions.

4.2 Effect of noise

Here we study the effect of noise in the model predicted by the toolbox. We use the Boolean network in Equation (3) as the truth and the trajectories T_1, T_2, T_3, T_4 as data. To this data we will add noise following a uniform distribution centered at 0 and with standard deviation σ. With the noisy data, we use the toolbox to create a model and make a prediction of the trajectory with initial condition 01111.

The results are shown in Figure 8. We see that for small noise the model is still able to make an accurate prediction of the trajectory. Since Boolean models focus on the qualitative features of he dynamics, it is robust to small noise levels. However, for large enough noise, the predicted trajectory does not match the true trajectory. A possible cause is that the first step of the discretization needs to distinguish between "low" and "high". If noise is large, it is possible that a low value plus noise is larger than a high value plus noise and may be incorrectly discretized.

4.3 Effect of number of levels

Here we use the network with 4 levels (or states) $f = (f_1, \ldots, f_5) : \{0,1,2,3\}^5 \to \{0,1,2,3\}^4$, where

$$f_1 = \max(3 - x_3, x_5) \qquad f_2 = \min(x_1, x_4) \qquad f_3 = 3 - x_2 \qquad f_4 = \min(3 - x_3, x_5) \qquad f_5 = x_4 \tag{4}$$

We generated 5 trajectories using this network: $32032 \to 33123 \to 32022 \to 32122 \to 22122, 23131 \to 22013 \to 31131 \to 23213 \to 31011, 22322 \to 22102 \to 20120 \to 22302 \to 20100, 03123 \to 30022 \to 32322 \to 22102 \to 20120,$

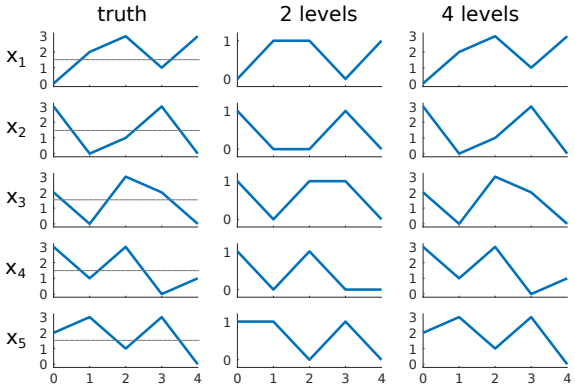


Figure 9: Effect of different levels. Using a coarser discretization, some features are lost. Using an appropriate number of levels, more features can be captured by the model.

03121 → 20012 → 31321 → 12202 → 20110. Using this data only, we use our toolbox to create a model and make a prediction for the novel trajectory with initial condition 03232. The true trajectory is shown in Figure 9 (first column).

Selecting a discretization of the data with 2 levels in the toolbox (0 and 1 become 0, and 2 and 3 become 1) will create a Boolean model. In this case, the initial condition 03232 would become 01111 and the Boolean trajectory will of course not match the true trajectory exactly. However, the Boolean trajectory does have the same qualitative features of the true trajectory, Figure 9. For instance, x_1 has the pattern $0 \to 2 \to 3 \to 1 \to 3$ in the true trajectory. If discretized, this would be $0 \to 1 \to 1 \to 0 \to 1$, just like the predicted Boolean trajectory.

We then considered a discretization that included the previous one. To achieve this we considered 4 levels. Based on the data, using 4 levels is a more natural discretization and indeed, the trajectory predicted by the toolbox matches the true trajectory, Figure 9.

4.4 Effect of threshold values

We now explore the effect of changing the threshold chosen for discretization on the predicted model. We use the network $f\colon \{0,1,2\}^3 \to \{0,1,2\}^3$ given by

$$f_1 = x_2 \qquad f_2 = \min(x_1, x_3) \qquad f_3 = \max(x_1, 1) \tag{5}$$

We use the data $111 \to 111$, $020 \to 201$, $002 \to 001$, $220 \to 202 \to 022$ and will attempt to predict the true trajectory $020 \to 201 \to 012 \to 101 \to 011$ using the model given by the toolbox. To make the comparison simpler, we choose 2 states only. Choosing different thresholds can potentially result in different models with different dynamical properties. The difference is shown in Figure 10. Different states can be mapped to the same value if they are both less than or greater than the threshold. This causes the loss of certain features, but some coarse qualitative features are preserved.

5 Discussion

Discrete models have been successfully used to model biological systems (Wooten et al., 2021; Veliz-Cuba and Stigler, 2011). Although several discrete modeling packages exist for their analysis (e.g., PlantSimLab, Ha et al., 2019; BoolNet, Müssel et al., 2010; BNReduction, Veliz-Cuba et al., 2014; GinSim, Naldi et al., 2009; CaSQ, Aghamiri et al., 2020; WebMaBoSS, Noël et al., 2021), they require an existing model or the wiring diagram to be created by the user. Few tools exist that provide an automated and easily customizable pipeline to quickly create model prototypes. Our toolbox allows the creation of model prototypes easily, which can then be used by existing modeling packages for validation, modification, or extension.

Equation learning methods in general require large amounts of data which might not be feasible in practice (Brunton et al., 2016; Lagergren et al., 2020). Furthermore, those approaches require knowledge of the form of the functions (some times called a library of functions) *a priori*, which may be unfeasible for unknown interactions. Even if the form of the functions is known for continuous modeling, the model obtained can be the result of parameter estimation being stuck in a local minimum. In contrast, our method can be used even with a limited number of time points. Although this does not guarantee predictive power, our toolbox does find all minimal wiring diagrams. This is important, because it can be seen as the discrete version of finding all local minima in parameter estimation for continuous models. Furthermore, our approach does not need to know the

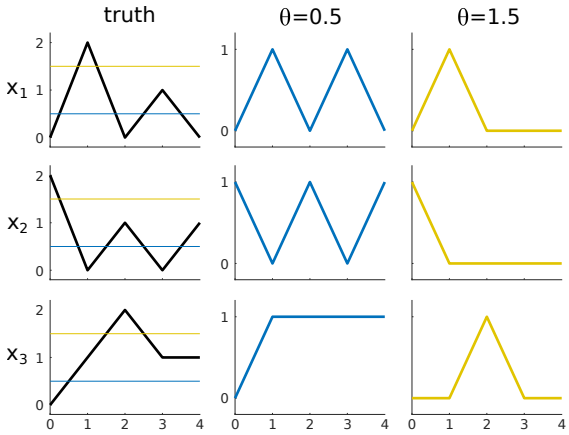


Figure 10: Effect of different thresholds. For thresholds around $\theta = 0.5$, values that should have been 1 or 2 in the true trajectory are all mapped to 1. For thresholds around $\theta = 1.5$, the values that should have been 0 or 1 are mapped to 0.

form of the functions *a priori*. We note that the discrete model resulting from our approach can be converted into a continuous model using existing approaches such as those of Wittmann et al. (2009) and of Manicka et al. (2022).

The limitations of our toolbox are those related to each component in the pipeline. Notably, if the discretization considers two instances of the same value as different due to noise (e.g., 0.7 as 0 and 0.9 as 1), this can cause overfitting. Selecting the correct number of levels of the model is also important and can cause missing some features if the number of levels is too low or overfitting if the number of levels is too large. Also, the selection of thresholds can make a difference on which features of the true dynamics are correctly predicted with the inferred model. Another limitation is that it is not known how much data is needed to guarantee that the predicted model is "close" to the true network unless the network is known *a priori*.

For the purpose of reproducibility, we provide all the data and the code that we use in our toy example and application which can be accessed through this link: `github.com/alanavc/prototype-model`.

Acknowledgments

A. VC. was partially supported by the Simons Foundation (grant 516088). S. R. V. was partially supported by NIH grant R24OD010435. D. M. was partially supported by a Collaboration grant (850896) from the Simons Foundation. The authors thank the referees for their insightful comments that have improved the manuscript.

References

Aghamiri, S. S., V. Singh, A. Naldi, T. Helikar, S. Soliman, and A. Niarakis (2020). Automated inference of Boolean models from molecular interaction maps using casq. *Bioinformatics 36*(16), 4473–4482. 117

Brunton, S. L., J. L. Proctor, and J. N. Kutz (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences 113*(15), 3932–3937. 107, 117

Dimitrova, E., L. D. García-Puente, F. Hinkelmann, A. S. Jarrah, R. Laubenbacher, B. Stigler, M. Stillman, and P. Vera-Licona (2011). Parameter estimation for Boolean models of biological networks. *Theoretical Computer Science 412*(26), 2816–2826. 107

Dimitrova, E. S., M. P. V. Licona, J. McGee, and R. Laubenbacher (2010). Discretization of time series data. *Journal of Computational Biology 17*(6), 853–868. 107

Ha, S., E. Dimitrova, D. Hoops, S.and Altarawy, M. Ansariola, D. Deb, J. Glazebrook, R. Hillmer, H. Shahin, F. Katagiri, J. McDowell, M. Megraw, J. Setubal, B. M. Tyler, and R. Laubenbacher (2019). PlantSimLab - a modeling and simulation web tool for plant biologists. *BMC Bioinformatics 20*(1), 508. 117

Hinkelmann, F. and A. S. Jarrah (2012). Inferring biologically relevant models: nested canalyzing functions. *International Scholarly Research Notices 2012*. 107

Hojo, H. and S. Ohba (2022). Sp7 action in the skeleton: Its mode of action, functions, and relevance to skeletal diseases. *International Journal of Molecular Sciences 23*(10), 5647. 112

Jarrah, A. S., R. Laubenbacher, B. Stigler, and M. Stillman (2007). Reverse-engineering of polynomial dynamical systems. *Advances in Applied Mathematics 39*(4), 477–489. 107

Lagergren, J. H., J. T. Nardini, G. Michael Lavigne, E. M. Rutter, and K. B. Flores (2020). Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A 476*(2234), 20190800. 107, 117

Laubenbacher, R. and B. Stigler (2004, Aug). A computational algebra approach to the reverse engineering of gene regulatory networks. *J Theor Biol 229*(4), 523–37. 107

Li, C., D. A. Scott, E. Hatch, X. Tian, and S. L. Mansour (2007). Dusp6 (mkp3) is a negative feedback regulator of fgf-stimulated erk signaling during mouse development. 112

Liang, S., S. Fuhrman, and R. Somogyi (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Biocomputing*, Volume 3. 107

Manicka, S., K. Johnson, D. Murrugarra, and M. Levin (2022). The nonlinearity of regulation in biological networks. *bioRxiv*. 118

Murrugarra, D. and R. Laubenbacher (2012, 5). The number of multistate nested canalyzing functions. *Physica D: Nonlinear Phenomena 241*(10), 929–938. 107

Murrugarra, D., J. Miller, and A. N. Mueller (2016). Estimating propensity parameters using google pagerank and genetic algorithms. *Frontiers in Neuroscience*, 513. 112, 113

Murrugarra, D., A. Veliz-Cuba, B. Aguilar, S. Arat, and R. Laubenbacher (2012). Modeling stochasticity and variability in gene regulatory networks. *EURASIP Journal on Bioinformatics and Systems Biology 2012*(1), 5. 111, 113

Müssel, C., M. Hopfensitz, and H. A. Kestler (2010). BoolNet - an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics 26*(10), 1378–1380. 117

Naldi, A., D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya (2009). Logical modelling of regulatory networks with GINsim 2.3. *Biosystems 97*(2), 134–139. 117

Noël, V., M. Ruscone, G. Stoll, E. Viara, A. Zinovyev, E. Barillot, and L. Calzone (2021). Webmaboss: A web interface for simulating Boolean models stochastically. *Frontiers in Molecular Biosciences 8*. 117

Ponomareva, L. V., A. Athippozhy, J. S. Thorson, and S. R. Voss (2015). Using ambystoma mexicanum (mexican axolotl) embryos, chemical genetics, and microarray analysis to identify signaling pathways associated with tissue regeneration. *Comparative Biochemistry and Physiology Part C: Toxicology & Pharmacology 178*, 128–135. 112

Stigler, B., A. Jarrah, M. Stillman, and R. Laubenbacher (2007). Reverse engineering of dynamic networks. *Annals of the New York Academy of Sciences 1115*(1), 168–177. 107

Sun, J., A. A. AlMomani, and E. Bollt (2020). Data-driven learning of Boolean networks and functions by optimal causation entropy principle (bocse). *arXiv preprint arXiv:2006.01023*. 107

Veliz-Cuba, A. (2012). An algebraic approach to reverse engineering finite dynamical systems arising from biology. *SIAM Journal on Applied Dynamical Systems 11*(1), 31–48. 107, 109, 111

Veliz-Cuba, A., B. Aguilar, F. Hinkelmann, and R. Laubenbacher (2014). Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics 15*, 221. 117

Veliz-Cuba, A. and B. Stigler (2011). Boolean models can explain bistability in the *lac* operon. *Journal of Computational Biology 18*(6), 783–794. 117

Wehner, D., W. Cizelsky, M. D. Vasudevaro, G. Özhan, C. Haase, B. Kagermeier-Schenk, A. Röder, R. I. Dorsky, E. Moro, F. Argenton, et al. (2014). Wnt/β-catenin signaling defines organizing centers that orchestrate growth and differentiation of the regenerating zebrafish caudal fin. *Cell reports 6*(3), 467–481. 112

Wehner, D. and G. Weidinger (2015). Signaling networks organizing regenerative growth of the zebrafish fin. *Trends in Genetics 31*(6), 336–343. 113

Wittmann, D. M., J. Krumsiek, J. Saez-Rodriguez, D. A. Lauffenburger, S. Klamt, and F. J. Theis (2009). Transforming Boolean models to continuous models: methodology and application to t-cell receptor signaling. *BMC Systems Biology 3*(1), 1–21. 118

Wooten, D. J., J. G. T. Zañudo, D. Murrugarra, A. M. Perry, A. Dongari-Bagtzoglou, R. Laubenbacher, C. J. Nobile, and R. Albert (2021). Mathematical modeling of the *Candida albicans* yeast to hyphal transition reveals novel control strategies. *PLoS Computational Biology 17*(3), e1008690. 107, 117