Master's thesis
Master's Programme in Data Science

Inference of Bacterial Growth Dynamics via Symbolic Regression

Karen Hovhannisyan

June 9, 2023

Supervisor(s): Professor Ville Mustonen

Examiner(s): Professor Ville Mustonen
Dr. Anthony Sun

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Karen Hovhannisyan

Työn nimi — Arbetets titel — Title

Inference of Bacterial Growth Dynamics via Symbolic Regression

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 9, 2023	39

Tiivistelmä — Referat — Abstract

Microbial growth dynamics play an important role in virtually any ecosystem. To know the underlying laws of growth would help in understanding how bacteria interact with each other and their environment. In this thesis we try to automate the process of scientific discovery of said dynamics, via symbolic regression. It has historically been implemented with genetic algorithms, and although many of the new implementations have different approaches, we stick with a highly optimized genetic-programming based package. Whatever the approach, the purpose of symbolic regression is to search for a mathematical expression that explains a response variable. We test the highly interpretable machine learning method on several datasets, each generated to mimic certain patterns of growth. Our findings confirm its ability to reverse-engineer theory from data. Even when the generating equations contain the latent nutrient variable, whose dynamics are not observable through the raw data, symbolic regression is able to find an analytically correct reparametrization and exact solution. In this thesis we discuss these results and give an overview of symbolic regression and its applications.

ACM Computing Classification System (CCS):
General and reference → Document types → Surveys and overviews
Applied computing → Life and medical sciences → Bioinformatics
Computing methodologies → Machine learning → Machine learning approaches → Bio-inspired approaches → Genetic programming

Avainsanat — Nyckelord — Keywords

Symbolic Regression, Data Visualization, Bacterial Growth, Dynamical Systems

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Classical machine learning methods (the likes of linear regression, decision trees, support vector machines, and neural networks) tend to sort themselves quite linearly when comparing their interpretability and accuracy. On one end of the spectrum would be linear regression, whose formulaic representation can give strong context to features and their relations, but whose assumed functional model is rigid and inapplicable to data with complex non-linear relations. As a result, predictive accuracy suffers. On the other end would be deep-learning neural networks, where many layers of nodes work together to produce accurate predictions but offer no greater insight into relations of features; a black box. However, methods exist that disturb this linear order depicted in figure 1.1. One such is symbolic regression (SR); a useful method to use in settings within natural science where underlying physical laws shape the data collected. With it, we look to recover a symbolic representation of a response variable with feature variables. The obtained output is an expression, similar to linear regression. However, it differs in that we only provide a library of operators and variables. The regressor then searches the state-space consisting of all possible expressions that can be made from these. It ultimately offers more flexibility than linear regression and can be used to uncover many different types of relations between features. Although, navigating the search space is an intensive and difficult task as the amount of possible expressions is enormous. To approach this, most existing SR implementations search the state-space via an evolutionary algorithm, although many variations exist [19].

In this master's thesis, we look to apply SR to bacterial growth data. Our aim is to recover equations that accurately describe the dynamics of growth. The regressor is run on many simulated datasets; each modeling bacterial growth with different dependencies. The simulations are done for sets of 1536 colonies at a time, reflecting the collection of growth data done by Zackrisson *et al.* [38]. We look to see how the regressor can be used to infer biological concepts from datasets with a limited source of information, acting as a useful tool in partnership with domain knowledge, to aid scientific discovery.

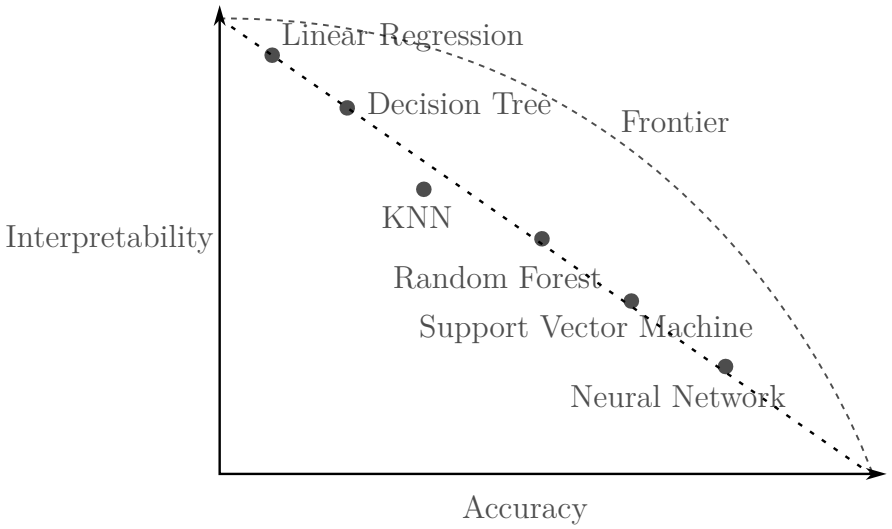

Figure 1.1: Schematic detailing the interpretability and accuracy trade-off, recreated from [23].

1.1 Biological Context

The problem we explore in this thesis is a biological one. Specifically, we try to leverage symbolic regression to better understand the growth dynamics of a plate of bacteria. The plate is an observable and controlled environment where 1536 colonies are placed in 32 rows and 48 columns.

To grow, bacteria need to consume nutrients, which are provided to them in the experiment. However, the nutrient concentration dynamics across the plate are difficult to measure, making it a latent variable within the growth dynamics [38].

Understanding how bacteria grow in relation to one another gives insight into how microbiomes evolve or how they would respond to changes in the environment. However, performing experiments with more than one bacterial strain in the same environment complicates the dynamics due to competition, making them expensive and demanding. For this reason, being able to infer a bacterium's inherent behavior from a single strain experiment is easier and also allows us to potentially extrapolate to more complex environments [26].

Growth curves of bacteria have three phases: lag, exponential, and stationary. The lag phase is the period where bacterial cells take to adjust to the environment and growth is minimal. The exponential phase is after the adjustment has been made and the cells begin to accelerate their growth. Finally, in the stationary phase growth slows due to the depletion of nutrients [26]. These phases can be represented as parameters in a differential equation. To find suitable representations for them, or other characteristics of growth, could prove to be useful in developing theory and understanding bacterial behavior in the real world.

1.2 Genetic Algorithms

Living organisms exist as expert problem solvers, with characteristics and properties that have been developed over many years of evolution. Mutations, although not always, give rise to emergent behavior that propel species to dominant positions, changing the dynamics of an ecosystem. Such genetic processes have shown to be quite effective in finding an optimal fit for conditions of life; and in this process of nature came inspiration for algorithmic approaches to complex problems. We can label these as genetic algorithms, a subset of evolutionary algorithms [15].

The evolutionary process can be reduced into selection and reproduction. Reproduction combines lineages to produce new generations that may be more, less, or equally fitted to the environment or problem at hand. However, we would hope that the current iteration improves upon their predecessor and this is made more likely by selection. A fitness equation evaluates how well a member of the generation performs and as a result, how likely it is to be selected for reproduction. In nature, fitness could be reflected by the health and vitality of an organism, also making them more likely to reproduce [15]. During the process, assuming two parties are involved, traits from each are non-trivially passed down to the offspring. However, the possibility of mutations occurring also exists. Mutations are relatively unlikely, but offer an escape from dead-ends, meaning when a species' lineage has settled into a stable state where a single profile dominates the space [15].

The inspiration of genetic algorithms is rather clear but its implementation needs more contextual information. It follows naturally that the population-analogous would be solutions to the problem we are using the algorithm to solve. These solutions should be represented in a way that allows them to breed together successfully, in order to mimic the evolutionary process. To allow for the mixing of various traits and characteristics, a modular format is conducive. Implementation details of selection and cross-breeding are then dependent on the task at hand. In this thesis we are concerned with the task of finding free-form symbolic equations to explain our data. This is typically done using expression trees [19], and can be labeled as symbolic regression.

1.2.1 Symbolic Regression

Symbolic regression is a supervised learning task where the optimization objective is across a symbolic state space. This space contains all possible symbolic representations of the feature variables and operators given to the regressor. Each representation is a mathematical expression that is evaluated with respect to how well they explain the response variable, and the objective is to find the expressions that explain it the

best. For our purposes, these will be differential equations. This is a difficult and intensive task, shown to be NP-hard [35]. In general SR is suitable for when we want to explore various functional forms at the same time to explain a signal within the data. However, the state space of possible equations is huge and not computationally reasonable to search only with brute force methods. To mitigate, it can be implemented with genetic programming, as most implementations are [19, 20]. We ourselves take this approach by using the PySR package [9]. Although other methods do exist, all guiding the search one way or another.

In the PySR SR framework, equations are represented as trees [9]. This is the standard representation in genetic programming where nodes can be operators, constants, or variables. The equation is built by combining the nodes from the bottom up [19]. This format is conducive to mutations and breeding equations with one another, by modifying nodes and swapping subtrees. Example of such representations can be found in figures 1.2 and 1.3. The corresponding fitness function is also quite intuitive, simply using a loss function that checks the equation's goodness of fit to the training data.

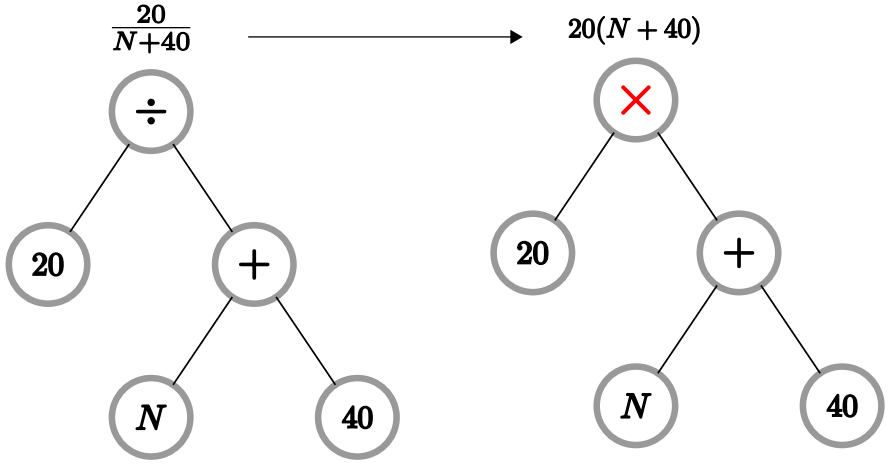

Figure 1.2: An example of how a mutation may occur in an expression tree.

AI Feynman is a symbolic regression software that takes a divide-and-conquer approach to symbolic regression, as opposed to genetic programming [32, 33]. It has been extensively tested on SR problems within physics with great success. It contains several steps in the fitting, including a brute force search step. This step is made viable by the fact that there is a dimensional analysis module and neural network aimed at simplifying the optimization objective by continuously splitting the problem into smaller ones. As a result, the smaller problems are solved independently and brought together to explain the original [33].

SR also has deep-learning and Bayesian variants. Petersen *et al.* [24] present Deep Symbolic Regression (DSR) and use generative AI and reinforcement learning

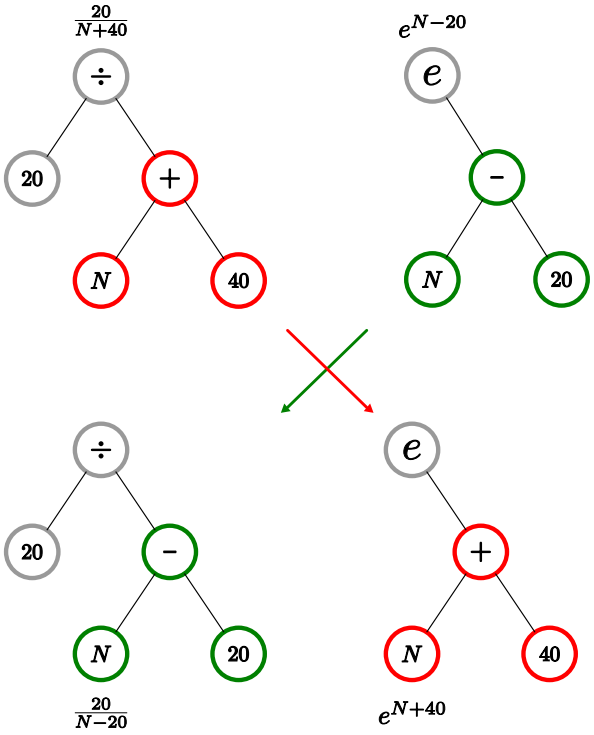

Figure 1.3: An example of how two expression trees may breed two new ones by swapping subtrees, also known as crossover.

in tandem, via a recurrent neural network and risk-seeking policy gradient. Equation Learner (EQL) is a neural network architecture trained to perform symbolic regression [29]. Kim *et al.* [17] combine this with an encoder and decoder and see significant improvements in the performance when reverse-engineering dynamical systems. Jin *et al.* [16] use assigned priors and posterior inference via Metropolis-Hastings and Monte-Carlo-Markov-Chain to create Bayesian Symbolic Regression (BSR). One advantage this has over typical GP based SR is that it reduces the memory used by the computation. In their results, it also tends to produce more concise equations, although this is because of tighter prior restrictions placed on the model. Another alternative to GP on expression trees is the use of sparse regression [28].

Genetic programming itself has not been a stagnant field. Virgolin *et al.* [34] introduce the GOMEA (Gene-pool Optimal Mixing Evolutionary Algorithm) to SR. This seeks to preserve patterns in the genotypes. The genotype in our case of SR would be the expression tree, and it would be preserved by having a rigid structure, instead of a free-form one. The result is more success with finding small concise solutions; but like BSR, has less flexibility.

In 2022, the Genetic and Evolutionary Computation Conference (GECCO) took place in the city of Boston [12]. The purpose of this competition was to compare various SR methods as there has been an emergence of new methods that look beyond

genetic programming. The package that we use for our biological case study, the open source PySR [9], also participated in GECCO.

The competition was split into 2 parts, synthetic and real world tracks. In the synthetic track, data was produced by specific functions and the methods were tested on their function reconstruction performance by several metrics [12]. This is akin to how we test PySR in our case study. The real world track gives the regressor data on COVID-19 key indicators with the task of forecasting future events. The study showed no significant separation from the top 4 methods of the synthetic track, of which PySR placed second, implying that a current state of the art does not exist [12].

The first place framework in the synthetic track, QLattice [5], also uses an evolutionary approach although in conjunction with probabilistic modeling and on graphs instead of expression trees. The first place position in the real world track, uDSR [21], uses a multi-pronged approach, including recurrent neural networks and sparse regression.

The tests exposed general challenges that SR has, mostly by the addition of noise into the dataset. Under heavier noise, the methods become worse at predicting the true underlying functional forms, but still demonstrate high accuracy with their chosen models. This was measured by the R^2 statistic [12]. Aside from accuracy, they also considered the simplicity of the output equation, and other task specific measurements. It is important to know that the computational budget was limited to 10 hours of runtime for large problems [12], whereas our environment allows us to run for much longer.

Symbolic regression, and evolutionary algorithms in general, lend themselves nicely to deep search problems. It follows then, that they could prove to be useful in the natural sciences, where tasks are presumably governed by laws, which may possibly be unknown. Tradeoffs of using SR instead of decision trees or deep learning approaches can be seen in figure 1.4, produced by Wadekar *et al.* [37] as part of their motivation to use SR for an astrophysics task.

1.2.2 Previous Applications of Symbolic Regression

Applications of SR are typically in the natural sciences, as it is not usually favored for predictive forecasting or classification. Out of the natural sciences, it is particularly useful in physics whose experiments tend to be quite controlled and relatively free of noise. We also discuss applications in the field of ecology, which more closely relates to our own work.

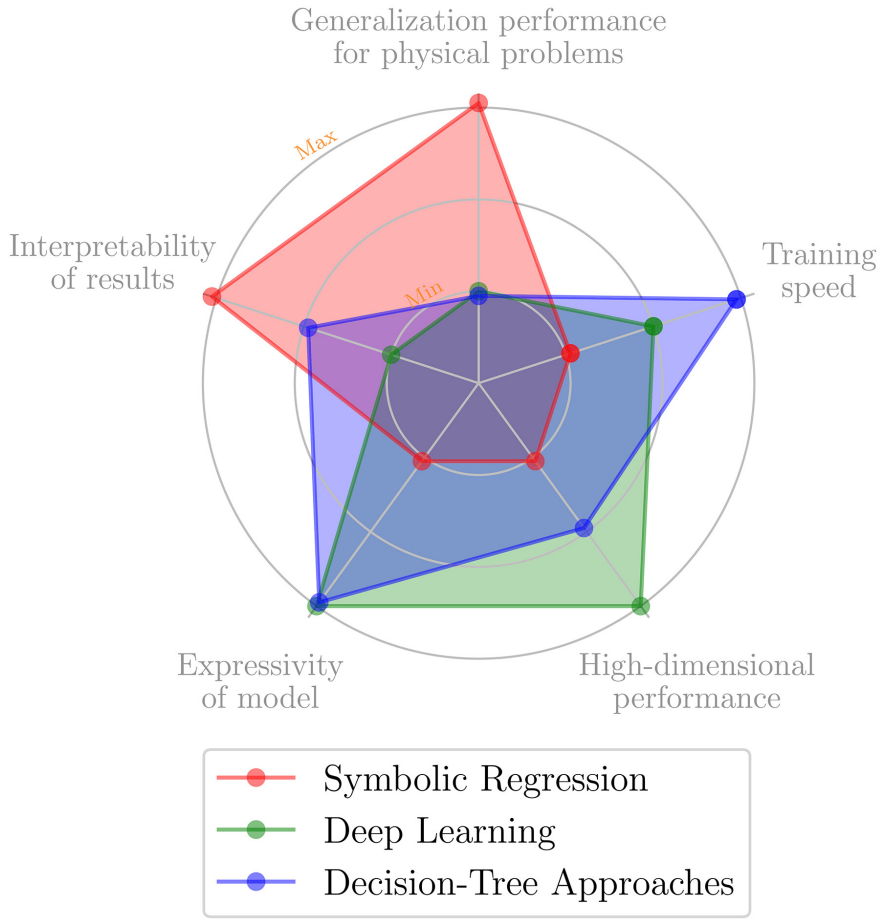

Figure 1.4: Comparison of Symbolic Regression with Decision Trees and Neural Networks. Taken from [37].

Physics

Using motion tracking sensors, time series data was collected of a chaotic double pendulum. This includes angles and angular velocities. This same data was fed into a symbolic regressor [30]. Partial derivatives of the time series data were numerically calculated and compared to the analytically calculated partial derivatives of the recovered formulas. The regressor ultimately discovered the Hamiltonian equation of the double pendulum, which is also the conservation law [30].

SR is also tested on data collected from oscillators. There is almost no noise involved and as predicted, the equations for the time derivative are almost exact when enough input information is passed [25]. In the same paper, SR is tested on a real-world situation where the prediction task is the 1-day forecast of solar power production from solar panels. In such a noisy situation it is difficult to assess the validity of any model, but the predicted equations do seem to follow the trend seen from observations [25].

Cranmer *et al.* [11] use SR not directly with the data but on components of a GNN trained to model interactions between particles. Much of physics can be generalized as

particle interactions; and in this paper they test the GNN-SR pairing on Newtonian dynamics, Hamiltonian dynamics, and a dark matter simulation. The GNN serves to reduce the dimensionality of the data, making the problem more suitable for SR. Within the dark matter simulation, they found that the resulting formula for predicting dark matter density generalized to other data better than the learned GNN itself [11].

SR has also been used to find partial differential equations with the presence of unobservable latent variables. This is not dissimilar to the problem we explore, although it is in the context of fluid dynamics, which ultimately defines the approach used by Reinbold *et al.* [27].

Ecology

In simple systems, population dynamics can be expressed by the Lotka-Volterra (LV), or predator-prey, equations, defined as the following [3]:

$$\begin{aligned} \frac{dX}{dt} &= \alpha X - \beta XY \\ \frac{dY}{dt} &= \delta XY - \gamma Y \end{aligned} \tag{1.1}$$

Here, X represents the prey population and Y the predator. α is the growth rate of the prey, or how quickly they reproduce. β represents the effect the predator has on the prey, hence why it is negative. δ is the effect the consumption of the prey has on the predator, and is a positive term. Finally, γ is the death rate of the predator [3].

It follows, then, that the equations and their parameters could be reverse engineered via SR given sufficient data. Namely, the relevant population time series [7]. This motivates us as the bacteria our group is concerned with grow under a resource-consumer model, which is conceptually similar to the Lotka-Volterra model. However, in practice, it is not a trivial task as this data is very susceptible to noise.

Bongard *et al.* [4] test SR on the recovery of 2 species Lotka-Volterra equations, with synthetically generated data (with noise). The recovered equations and coefficients are virtually exact. Gaucel *et al.* [13] also test on a simple 2 LV system with varying degrees of noise. Predictably, increasing the noise leads to less accurate equations. Although, the general structure is always found, and close to exact. Martin *et al.* [22] test on the specific 2 species LV system of *Paramecium* and *Didinium* cultures. The SR working in conjunction on the 2 sets of growth curves was able to recover the LV equations. The output is similar to the outputs we receive from PySR, the best performing equations for each level of complexity. They look for the equation along this Pareto-front, where the fit is first saturated. This tends to happen quite early for them resulting in relatively low complex equations which is desirable [22].

SR has also been extended to solve more complex ecosystems. For example, the insect *Tribolium* have cannibalistic properties and multiple life stages, resulting

in chaotic behavior [22]. SR found a 3 parameter model for this that was already conjectured by Constantino *et al.* [8]. This showcases SR's ability to discover non-linear dynamics.

Chen *et al.* [7] test SR on a 6 species synthetic ecosystem. They find when given enough prior information, that SR performs well. However, even when prior information is not passed, an accurate model is found, albeit highly complex and uninterpretable. In this case, the prior information was a dictionary of possible interaction terms that the regressor would have access to use. These constraints are reasonable to put on the regressor only if domain knowledge permits [7].

Schmidt *et al.* [31] use SR to discover dynamics of the glycolytic oscillations of yeast, although the datasets are simulated. Even still, the metabolic network of yeast is high dimensional with complex dynamics. Instead of using SR in the form of genetic programming on expression trees, they develop their own software where expressions are represented as acyclic graphs. The result is an automatic inference of 7 differential equations explaining the network reliably. The model was given no prior information and described dynamics which took years to analyze and understand previously [31]. In our experiments we also look to see how SR can accelerate our analysis of bacterial growth.

SR has also been applied to ecology problems of a larger scale. For example, the modeling of species distributions (SDM) across islands [6]. This is a particularly difficult problem due to the amount of variables and interactions that can exist in an uncontrolled large scale environment. Potential sources of noise include unpredictable weather events and interactions between different regions or ecosystems [6]. Cardoso *et al.* [6] compare SR to logistic regression and maximum entropy models for species distribution of a rare beetle and common spider on Terceira island. These models are the most widely used for SDM. They found that SR outperformed both with respect to their performance metric of true skill statistic. But perhaps more importantly, they found the equations the easiest to interpret.

1.3 Experimental Setup

All training and testing data used has been computer generated without noise, using a system of differential equations and initial values. The format mimics the setup purported by Zackrisson *et al.* [38] consisting of separate plates of bacterial colonies. Each plate has a grid projected onto it, representing 1536 colonies set up in 32 rows by 48 columns. Notably, each plate is flat, without wells, meaning colonies have the possibility to interact via the shared environment. A well is a depression in the plate that would serve to keep colonies isolated. The border around the plate is free of

bacteria (but not nutrients), meaning that the border colonies have fewer neighbor colonies and more space as well as nutrients to grow [38]. In addition, we assume this to be a mono culture experiment, meaning only one strain of bacteria is present on the plate. We produce four datasets by using different sets of generating functions, each implying characteristics of the plate dynamics. The growth curves of each can be seen in figure 1.5.

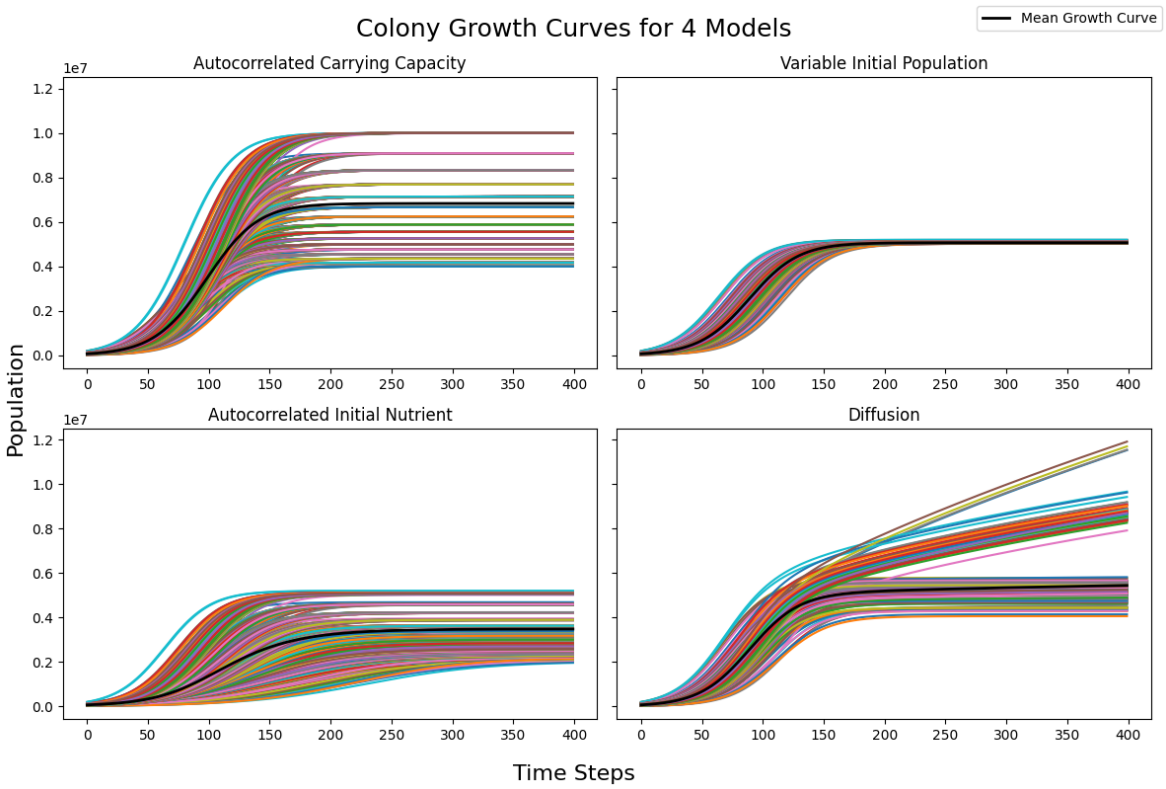

Figure 1.5: Growth curves of a single plate for each of the 4 datasets used. The black line shows the mean growth curve for each dataset. Variances between the growth curves vary greatly depending on the underlying dynamics. The lag phase refers to the beginning portion of the curve where there is no growth. The exponential phase is the middle section where the growth happens. Finally, the stationary phase is when the growth curve plateaus.

In the real laboratory experiments colonies and nutrients are initialized by a robotic pin-head holder that deposits cells and nutrients, giving us known initial values. The plates are then scanned every 20 minutes, producing a time series of population size for each colony in each plate [38]. The bacteria need to consume nutrients to grow and so as the bacterial populations grow in size, the nutrient concentration will decrease. Bacteria have 3 stages of growth. First is the lag phase, followed by the exponential phase, then finally the stationary phase [2]. These phases put together then make a familiar logistical curve typically seen in ecology. Unlike the populations, the nutrient concentration is not observable through the scans, and as a result becomes

a latent variable. We adhere to this property of the experiment when testing on our synthetic data. The proximity of the colonies means that interactions between them will occur, creating inter-colony competition. Therefore, one might expect the border colonies, who have more space, less competition, and more nutrients, to thrive. All of our datasets with the exception of variable initial population (VIN) give advantage to the border colonies.

1.3.1 Theoretical Models of Growth

Growth curves can be modeled in many different ways. In this section we explore 4 specific sets of dynamics, 3 of which are resource-consumer models.

General Model of Growth

The resource-consumer system provides a fundamental framework to study the interactions between a species and the resource it consumes.

$$\begin{cases} \frac{dN}{dt} = g(S)N \\ \frac{dS}{dt} = f(N,S) \end{cases} \tag{1.2}$$

These equations show a general system of differential equations, where N and S are the population and nutrient densities respectively [1].

Auto-correlated Carrying Capacity Model (ACK)

The ACK model is the only one we present that is not coupled with a resource equation. It is a logistic growth equation that approaches a set carrying capacity.

$$\frac{dN_{ij}}{dt} = rN_{ij}(1 - \frac{N_{ij}}{K(x)}) \tag{1.3}$$

N_{ij} is the population density of the colony at row i and column j and r is the growth rate of the colonies. r is a parameter that sets how fast the bacteria grow when population density is relatively low [26]. K is the carrying capacity as a function of position x, on a plate. This parameter controls when the growth will enter the stationary phase, because as N_{ij} approaches it, the growth rate will approach 0. x is a single value representing the distance to border, ranging from 1 to 16.

This dataset is different than the others in that there is no governance from the nutrient concentration dynamics. Instead the population is simply controlled by $K(x)$. Colonies closer to the border will be assigned larger values for K, implying that they have less competition and a larger pool of nutrients to consume. This is ultimately reflected in their higher final populations.

Parametrization via Constant of Motion

When the nutrient dynamics are simple enough, we can solve the system by using a constant of motion.

$$\begin{cases} \frac{dN}{dt} = rNS \\ \frac{dS}{dt} = -\nu\frac{dN}{dt} \end{cases} \tag{1.4}$$

These coupled equations show a simple case of bacterial growth dynamics. Let us define variable M as $M = S + \nu N$. Then

$$\frac{dM}{dt} = \frac{dS}{dt} + \nu\frac{dN}{dt}$$
$$\frac{dM}{dt} = -\nu\frac{dN}{dt} + \nu\frac{dN}{dt} = 0$$

Therefore M is a constant of motion. Now substituting $S = M - \nu N$ into $\frac{dN}{dt}$:

$$\frac{dN}{dt} = \frac{rN}{M}(1 - \frac{\nu N}{M})$$

The carrying capacity in this equation is $\frac{M}{\nu}$. Since M is a constant, we can also define it as $M = S(0) + \nu N(0)$, inferring that the carrying capacity can be set by the initial values of the nutrient concentration or population. We proceed to create 2 datasets: one where the initial nutrient concentration is autocorrelated (ACS), and another where there is no autocorrelation (VIN). For all our datasets, the initial population is varied across colonies.

This is a logistic differential equation with a closed form solution. We will not cover the derivation but the resulting N and S are as follows:

$$\begin{cases} N(t) = \dfrac{MN_0e^{Mrt}}{M - N_0\nu + N_0\nu e^{Mrt}} \\ \\ S(t) = \dfrac{MS_0}{M - N_0\nu + N_0\nu e^{Mrt}} \end{cases} \tag{1.5}$$

However, if $N(t)$ is given to us, then we can solve for $S(t)$ as an independent ODE. We first substitute $\frac{dN_{ij}}{dt}$ into $\frac{dS_{ij}}{dt}$ of system 1.4 to get $\frac{dS_{ij}}{dt} = -\nu r N_{ij}(t) S_{ij}$. Since $N_{ij}(t)$ is a known function, the solution to this differential equation is:

$$S_{ij}(T) = S_{ij}(0) * e^{-r\nu \int_0^T N_{ij}(t)dt} \tag{1.6}$$

That means that we can also implicitly define $\frac{dN_{ij}}{dt}$ as an integro-differential equation:

$$\frac{dN_{ij}}{dt} = S_{ij}(0) r N_{ij} e^{-r\nu \int_0^T N_{ij}(t)dt} \tag{1.7}$$

Autocorrelated Nutrient Initialization Model (ACS)

The initial conditions are very important when considering dynamical systems. In the ACS dataset, we use a simple framework like in system of equations 1.4, but with spatially correlated initial conditions.

$$\begin{cases} \frac{dN_{ij}}{dt} = rN_{ij}S_{ij} \\ \frac{dS_{ij}}{dt} = -\nu\frac{dN_{ij}}{dt} \end{cases} \tag{1.8}$$

$S_{ij}(0)$ is initialized according to its distance from the border, x. And ν determines the consumption rate of the nutrients.

Now we have a coupled population and nutrient equation. The change indicated in the nutrient equation is always less than 0, reflecting the dwindling nutrients. The spatial effect is induced by giving higher initial nutrient concentrations to colonies closer to the border, resulting in larger populations.

Variable Initial Population Model (VIN)

In the VIN dataset, only the initial population sizes are varied, but without any significant spatial correlation. We continue to use the simple system of equations 1.4.

$$\begin{cases} \frac{dN_{ij}}{dt} = rN_{ij}S_{ij} \\ \frac{dS_{ij}}{dt} = -\nu\frac{dN_{ij}}{dt} \end{cases} \tag{1.9}$$

$N_{ij}(0)$ is initialized randomly and $S_{ij}(0)$ is initialized to 1 across all colonies. This dataset does not give any benefits to colonies on the border, as evidenced by the growth curves in figure 1.5.

Diffusion Model

The last model we cover is the most complicated one. Here, we introduce a diffusion term into the nutrient dynamics to represent the flow of nutrients across the plate.

$$\begin{cases} \frac{dN_{ij}}{dt} = rN_{ij}S_{ij} \\ \frac{dS_{ij}}{dt} = -\nu\frac{dN_{ij}}{dt} + D(\bar{S_{ij}} - S_{ij}) \end{cases} \tag{1.10}$$

D is the diffusion rate, and $\bar{S}$ is the average nutrient concentration of the colony's neighbors. These equations state that nutrients diffuse naturally from higher to lower concentrations at a rate of D. This diffusion term is also the only distinction between the Diffusion and ACS equations. However, there is no autocorrelated initial nutrient concentration in the Diffusion dataset; all colonies start with a concentration of 1.

The diffusion term introduces interactions between the colonies, by way of the $\bar{S}$ term. This term links each colonies generating functions with their neighbors', resulting

in all 1536 colony dynamics being directly or indirectly linked with one another. The border colonies are favored in this dataset because they have fewer neighbors and so less colonies to share their nutrients with. The results are that the border colonies, and especially the corners of the plate, do not run out of resources within the simulation time and continue to grow, as seen in figure 1.5.

2. Methods

2.1 Preprocessing

We have an understanding of general growth behavior but the exact underlying form should not be assumed. We use the per-capita growth rate, $\frac{dN}{dt \cdot N}$, as our response.

Since we are just given a population vector, and our desired output is a differential equation, we must infer the derivatives ourselves for training and testing. We do this by interpolating the data with Scipy's [36] cubic spline method, and then differentiating the twice-differentiable result. This provides an avenue for computational imprecision to sneak in that may affect the precision of the equations found.

Because of the latent nutrient dynamics, we know that it may not be possible to achieve exact functional forms, although three out of four of our datasets have solutions. Instead, we are looking for equations that may offer interpretation to the dynamics at play. To allow for greater complexity in the dynamics we create several new variables. The first is the static colony variable *ring*, which represents the distance to border. The next three variables are $\bar{N}$, $N_{cumulative}$ (N_C), and $\bar{N}_{cumulative}$ ($\bar{N}_C$). $\bar{N}$ is the population analogous to $\bar{S}$, a variable which we hope codifies colony interaction. The cumulative variables are akin to integrating the population over time. So, they can also be shown as $\int N dt$ and $\int \bar{N} dt$.

2.2 Workflow

When running the regressor, it is unknown which features are the best to use. We also should not trust the regressor to pick out the best features if all are passed into it, as there may not be an exact solution to find. In which case, more features may just unnecessarily complicate the fit and take away from the desired interpretability. For this reason we use an iterative approach in our search. We first start by introducing few variables, to attempt to explain the variability simply, and then observe the fit.

We adopt a greedy approach to searching for the optimal feature set. This entails that we first search for the feature that explains everything the best by itself, and then

Table of Experiments				
	No Interactions			Interactions
Model / Features	ACK	VIN	ACS	Diffusion
N	✓	✓	✓	✓
*S			✓	✓
Ring	✓	✓	✓	✓
N(0)	✓	✓	✓	✓
*S(0)			✓	
$\overline{\mathrm{N}}$				✓
*$\overline{\mathrm{S}}$				✓
$\int \mathrm{N}$		✓	✓	✓
$\int \overline{\mathrm{N}}$				✓

Table 2.1: Table outlining which features would be expected to be relevant for each dataset.
* Not observable from real data

keep adding the feature that offers the largest improvement to the existing set; until there is no longer an improvement. We compare the fits by their loss-complexity pairing.

The output of any given PySR run is a Pareto front, the best performing expression for various complexities. PySR also includes a "score" metric for each of these that is defined as "the negated derivative of the log-loss with respect to complexity" in the PySR documentation [9]. The model will choose the best expression from the pareto front by looking at both the score and training loss. It considers the expression with the lowest training loss as a starting point, and then looks to other expressions that have a loss within 1.5 times this. Then, of all expressions within this range, the one with the highest score is chosen as the best. However, we find that often times the expression with the best score across the whole front is usually more interpretable while still maintaining performance. More complex ones already start to over-saturate the fit. We therefore consider two expressions automatically for each model run: the Model Best and the Best Score. We assign an expression its "type" based on how it was chosen. We may also hand pick one if we are not satisfied with either of the two. In addition, for an expression to be considered, it should use a different set of variables than previously found expressions of the same type. Otherwise, any improvement on the result can be attributed to the stochastic nature of the model. If adding a feature does not improve upon an existing solution, then we stop considering any feature set containing it.

It is also noted that we do have *a priori* information given the setting and biological context. Table 2.1 lists some models of bacterial growth and which of our features are presumed to be relevant. We can use this as a heuristic for our greedy approach to dramatically reduce the number of paths we would need to explore and avoid excess computation. For example, we split the greedy approach into two: one that considers population variables only if they are not cumulative; and another that considers the opposite. It is also worth to consider the top-down approach, where we pass all possible features. If an exact solution exists within the data, the regressor should be able to find it reliably from this, as exhibited at GECCO [12]. In that case, we expect to obtain a fit with an exceptionally low loss. Aside from this, it may also give insight into how the population is best codified in the equation. Since we have variables that show population by time and cumulative population over time, including both seems redundant. The top-down result may tell us which type of variable is best suited, or if indeed using both is the correct approach. We look at the top-down result only after considering the non-cumulative approach, and if it gives reason to consider further features, we also move forward with the cumulative approach.

Another thing to take into account is the operator library. Operators which may be unnecessary for some datasets could be crucial for others. In our case, this pertains to the unary exponent operator. However, it is hard to justify *a priori* what should the exact set of operators be. And for this reason we use a fixed set across all runs, consisting of: addition, subtraction, multiplication, division, and the exponent operator.

Symbolic regression also lends itself nicely to manual inspection. We take advantage of this, in that besides looking at the loss and complexity, we look for known or insightful expressions within the equation. We further emphasize SR's role as a tool to aid discovery. Its selling point is a highly interpretable output that performs well. However, to make use of the interpretation requires domain knowledge. There may be aspects of the output that do not make sense, and others that inspire further research.

We run all our fits as a job on a cluster for 80000 seconds ($\sim$22 hours) with a stop loss criteria. The jobs have 32 cores and 128 gigabytes of memory allocated for them. We set the stop loss criteria to be 2e-9. We felt that this loss was just high enough so that the run would terminate if an exact solution was found.

Figure 2.1 shows the process described for just a single dataset.

2.3 PySR

PySR is an open-source symbolic regression engine, built on a Julia backend. It allows for highly customizable runs, offers good performance as proven in GECCO [12], is

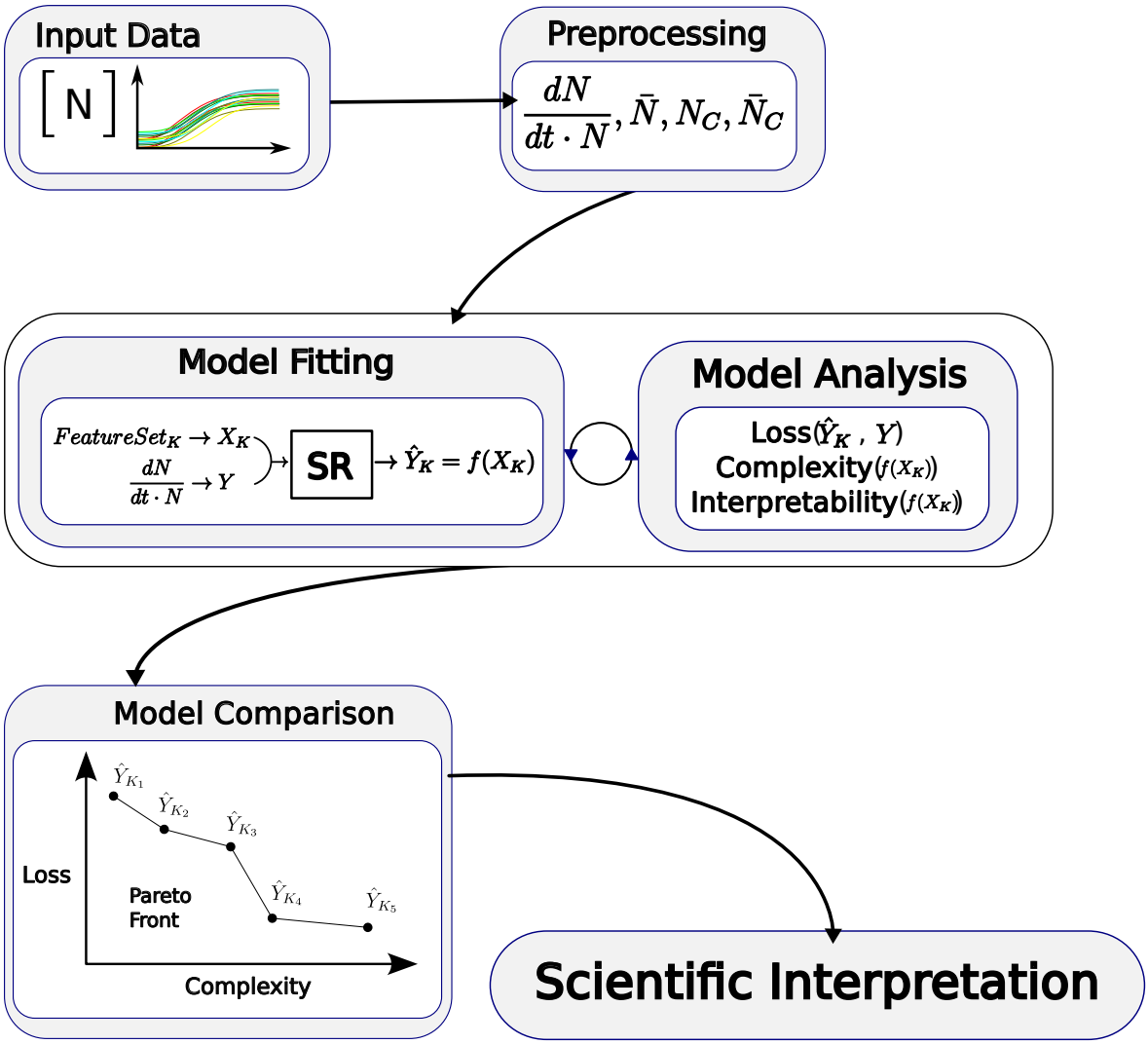

Figure 2.1: Schematic giving a high level overview of our workflow. We first start by preprocessing the given dataset, then start to fit our models with various feature sets. We then compare our models and finally, interpret the output equations.

easy to use with python, and is parellelizable [10]; making it our software of choice. The algorithm is a genetic one, using tournament selection [14] and an optimization technique called simulated annealing [18]. Although the use of annealing is optional and we restrain from using it. Tournament selection refers to how we select equation-trees for fitting and breeding.

There are 2 levels to the breeding that occurs. The first is the breeding of expressions on "islands", of which there are several. The second level occurs intermittently and is when expressions migrate across these islands. This allows for opportunity to massively parallelize the process. Further details can be found in [10].

2.3.1 Fitting

We notice a difference in predictive ability when looking for $\frac{dN}{dt}$ as opposed to $\frac{dN}{dt \cdot N}$. There are two reasons we can use to explain the gain in predictive power for $\frac{dN}{dt \cdot N}$. First, the latter response variable has a simpler expression form as N can be factored out. And second, the per-capita growth rate has a smaller scale which may contribute to greater ease of fitting. Our reasoning for this is that as the variance is smaller, the regressor will take less intermediate steps in trying to explain it. Another way to reduce the scale could be done with standardization, by dividing each observed population value by the maximum observed value in the entire plate. However, there is no need if we are using $\frac{dN}{dt \cdot N}$.

We take into account the guide for fitting given by Miles Cranmer on the PySR documentation [9]. Our configuration places constraints on the complexity of the equations, as well as how the exponent operator is used. We restrict the complexity of any term with the exponent operator to 5, and also disallow any nested exponent operators, for example e^e. The max complexity we allow is 25 and max depth of the expression tree is 10. We felt that this configuration worked well for each dataset. The full configuration code can be seen in appendix A.

We choose to train the model on 10% of the colonies of the plate. We select them with some structure, choosing randomly 10% of the colonies from each ring. We do this to make sure that any spacial effect can be captured by the regressor. Another thing to consider is that the runs are not reproducible unless a random number generator seed is set. Even when there is a seed set, it restricts the computation to one core, which undermines the parallelization of the software and reduces performance [9]. It is therefore advisable to run the algorithms for long enough in hopes that it converges to a solution.

The code for the preprocessing and fitting can be found on github.com/krnhov/InferringBacterialDynamics.

3. Results

In the following section, we will show the recovered equations of various datasets. All attempt to explain our response variable, the per-capita growth rate. We also look to see how valid they are by looking at the interpretability and considering previous domain knowledge. We compare the performances of the equations against the Null model, which we define as using the plate-mean per-capita growth rate as our predictor, given by equation 3.1

$$\hat{f}(t) = \frac{1}{1536} \sum_{i=1}^{32} \sum_{j=1}^{48} \frac{\dot{N}_{ij}(t)}{N_{ij}(t)} \tag{3.1}$$

$\frac{\dot{N}_{ij}}{N_{ij}}$ is the per-capita growth rate. This serves as a useful benchmark to show how the further models improve upon the explanation of the dynamics. The rates and null model for each dataset can be seen in figure 3.1.

We calculate the errors of our equations using the root mean squared error (RMSE), whereas the model measures training loss by using the L2 squared loss. All datasets start with the same initial population density, seen in figure 3.2.

3.1 Autocorrelated Carrying Capacity Dataset

We first begin by analyzing the results from the ACK dataset. Equation 3.2 gives the true dynamics used to generate the data.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5 * (1 - \frac{N_{ij}}{K(x)}) \tag{3.2}$$

x is the *ring* variable and $K(x)$ has values according to the plot in figure 3.3. First, we observe the heatmap of errors when using the Null model, in figure 3.5. We can see that the Null model performs well for those colonies that are neither in the middle or on the border. This follows logically, considering that these are closest to the mean of the plate.

The best equations from each feature-set are plotted in figure 3.4. After running the regressor with all of our non-cumulative feature-sets, we take the top-down ap-

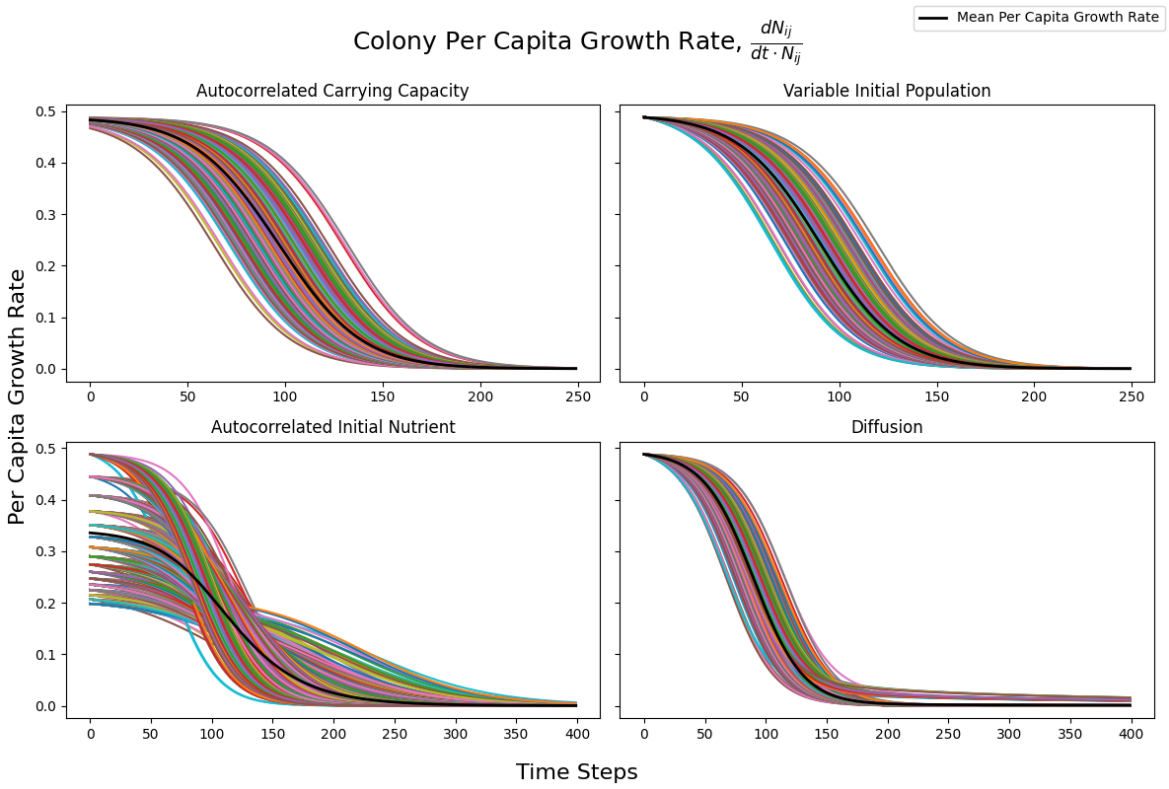

Figure 3.1: The per-capita growth rate for all colonies in each dataset. The black line is the mean curve and also serves as our null model.

proach, where all features are fed into the model. This is the purple cross in the figure. We determine that it does not offer any significant improvement on existing solutions and so do not continue to fit any models on the ACK dataset.

To the eye, it looks like the Best Score solution of the feature-set $N, Ring$ is the best combination of training loss and complexity. We can justify this further by comparing the errors of the plates of the various solutions. As a relevant point of comparison we choose the solution to the feature set consisting of just N.

By looking at colony average RMSEs in figure 3.5 we gain further insight into how the models perform compared to each other. When N is the only feature, not much of the variance is explained and it actually performs much worse than the Null model. However, once we run with both N and $Ring$ we see a dramatic reduction in error. This suggests that $Ring$ is a key variable in predicting the per-capita growth rate as well as the fact that there is positional dependency in the dataset. First, we look at the Best Score equation, 3.3, of complexity 6. We notice that it brings the error down close to an order of magnitude while still maintaining a high degree of interpretability. We also compare with the Model Best equation, using N and $Ring$. It has even lower error, although the fit seems saturated. Interpretability suffers greatly, with a quite large complexity of 23, and it is more likely to be a case of overfitting.

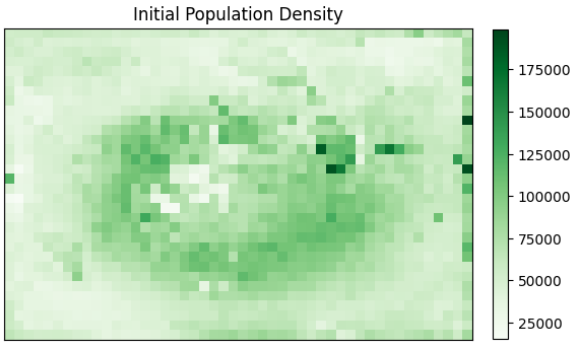

Figure 3.2: The initial population densities for each dataset.

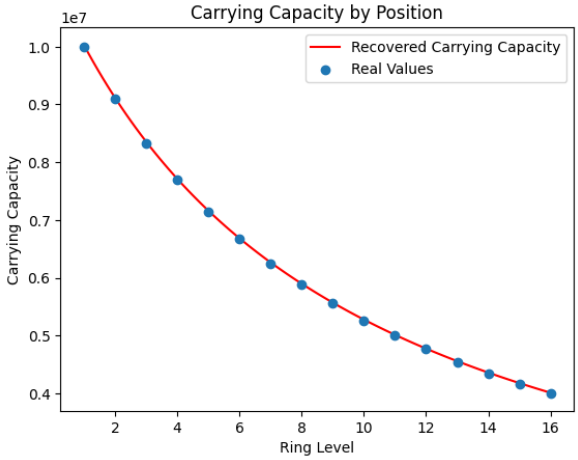

Figure 3.3: Plot of the carrying capacity by ring of the ACK dataset. The real values are blue dots, and recovered function from equation 3.3 is the red line. The recovered function performs well, matching the real values with high accuracy.

For these reasons, we feel that equation 3.3 is the best choice.

Equation 3.3 is the result after we round all constants to 4 decimal places, and manipulate the raw output to a more familiar functional form. It is a logistic growth differential equation, from which we can infer the carrying capacity as the inverse of the coefficient of N_{ij}, which is a function of $Ring$. Figure 3.3 shows a comparison between the real values of the carrying capacity and the recovered $K(x)$. We also see that it found the correct growth rate parameter.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = .4906(1 - (9.9769 \cdot 10^{-9} N_{ij}\,(Ring + 8.9917)) \tag{3.3}$$

The dramatic increase in performance when providing the $Ring$ variable, as well as the interpretable form suggests that the position of the colony is important. Of course, the dataset we are using is generated by using this exact $Ring$ variable and so this was expected. In a real system we would not expect to see such a clean correlation between the distance to border and carrying capacity.

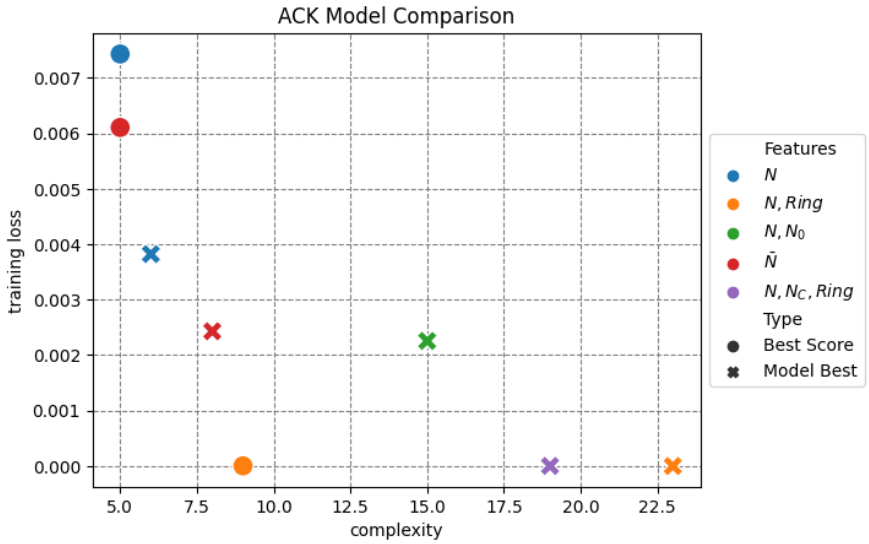

Figure 3.4: Comparison of the different models for the ACK dataset. The "type" refers to how that equation was selected from the model's output. Loss is L2 training loss and complexity is number of nodes in the expression tree.

3.2 Autocorrelated Initial Nutrient Concentration Dataset

Next, we look at the fits to the ACS dataset, whose initial nutrient values are spatially correlated. Equations 3.4 give the exact dynamics used to generate the data.

$$\begin{cases} \frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5 * S_{ij} \\ \frac{dS_{ij}}{dt} = -2 \cdot 10^{-7} \frac{dN_{ij}}{dt} \end{cases} \tag{3.4}$$

$S(0, Ring)$ can be seen in figure 3.6. The nutrient concentration is lower for colonies that are further from the border, putting them at a disadvantage.

The Null model errors in figure 3.8 show that there is a clear systemic bias based on the *Ring* of the colony. This immediately suggests that including *Ring* would be beneficial. Predictably, the 3 best performing expressions all use it, as seen in figure 3.7.

Compared to the Null model, the three best performing expressions all explain the variance very well. However, there is remaining bias in the model using $N, Ring$. After comparing to the heatmap in figure 3.2, we suspect the variation to be attributed to the fact that the recovered formula does not capture the intial population. The models using the cumulative variables, on the other hand, do capture it, as N_0 is baked into the variable. We, unfortunately, made the mistake of not running the model with the feature-set consisting of $N, N_0, Ring$, but we would expect to see a high performing equation there as well; possibly by using the M parametrization. The expression of

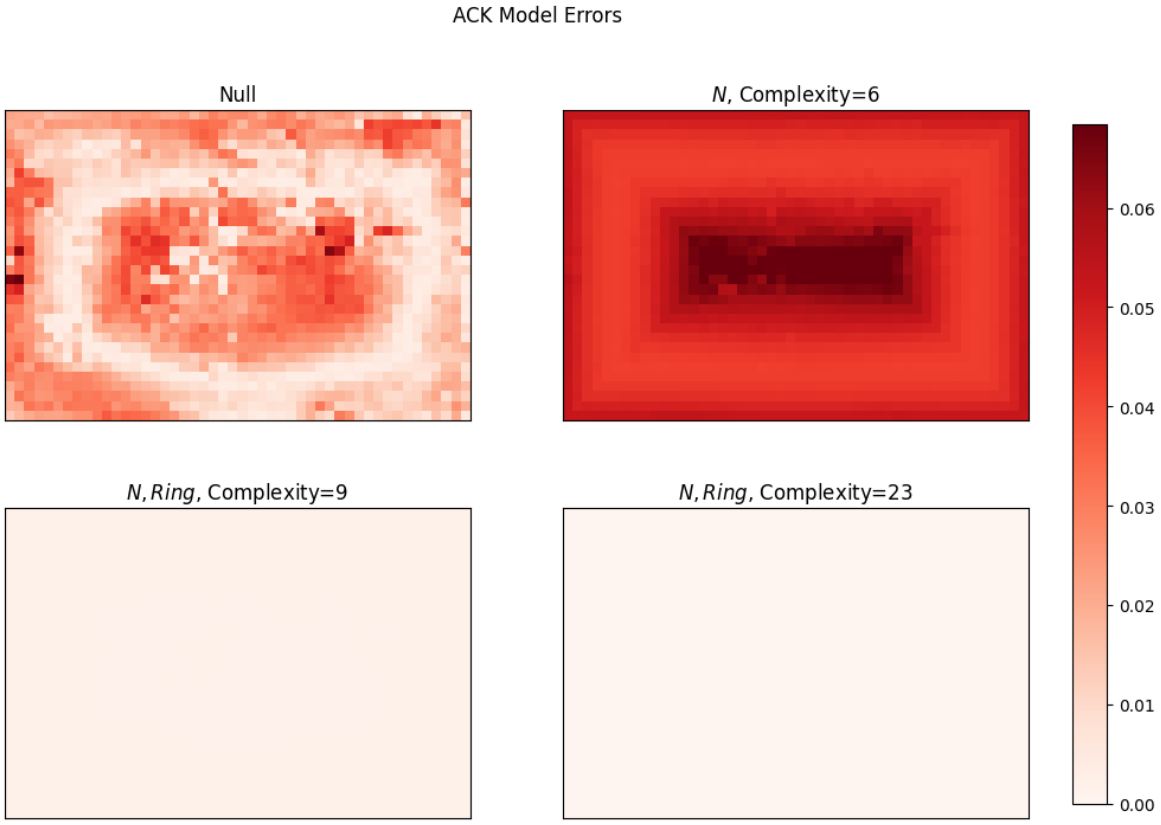

Figure 3.5: Comparison of colony average RMSE between models of the ACK dataset. We compare the models that use the feature-sets of N and $N, Ring$.

complexity 9 and features $N, Ring$ is equation 3.5. The expression of complexity 11 and features $N_C, Ring$ is equation 3.6. We do not consider the expression consisting of features $N, N_C, Ring$ as it offers no significant improvement on equation 3.6, and is also more difficult to interpret.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = -9.8375 \cdot 10^{-8} N_{ij} + \frac{5.1833}{Ring + 9.4473} \tag{3.5}$$

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = \frac{4.9819 e^{-9.6798 \cdot 10^{-8} N_C}}{Ring + 9.1440} \tag{3.6}$$

It is interesting how the regressor can parametrize the dynamics with N_C and achieve a commendable result. In fact, the form is the same as of equation 1.7, an exact solution. $S(0)$ is even expressed as a function of $Ring$ as is the case in the true dynamics.

We also note parallels between its counterpart in equation 3.5. The coefficients are all very close to each other suggesting that the parameters that can be extracted are the same. The term of approximately $\frac{5}{Ring+9}$ is presumed to explain the variance between rings. By plotting this, we notice that it is in the same shape as the real initial

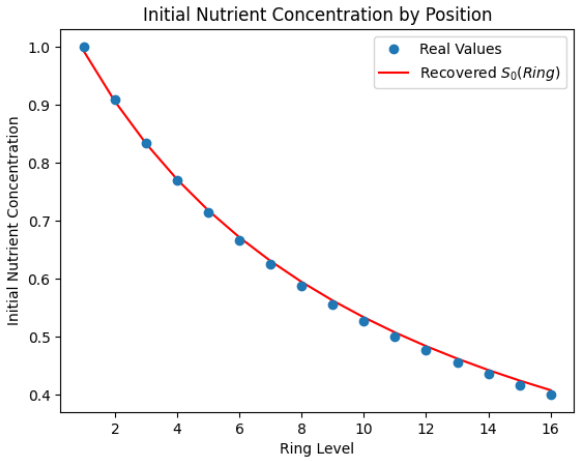

Figure 3.6: Initial nutrient concentrations of the colonies in the ACS dataset based on their distance to border or ring level. The blue dots are the real values and the recovered function from equation 3.7 is the red line.

nutrient concentration in figure 3.6 but on a twice smaller scale. Therefore, multiplying the term by one half (since the term suggests inverse relation between $Ring$ and $S(0)$) will correct the scale and also separate the parameters in the equation. The results are equation 3.7 and 3.8. We also show the recovered $S_0(x)$ function in figure 3.6.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5(-1.9747 \cdot 10^{-7} N_{ij} + \frac{10.366544}{Ring + 9.4473}) \tag{3.7}$$

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5 \frac{9.9638}{Ring + 9.1440} e^{-9.6798 \cdot 10^{-8} N_C} \tag{3.8}$$

If we use the M parametrization, equation 3.4 can be written in the form:

$$\frac{dN}{dt \cdot N} = r(M - \nu N) = r(\nu N(0,x) + S(0,x) - \nu N) \tag{3.9}$$

Comparing Equation 3.7 to this, we see that it is only missing information about $N(0,x)$, which results in the remaining bias. Ultimately, the regressor found two ways to explain the response variable in the ACS dataset.

3.3 Variable Initial Population Dataset

The next dataset we look at is the VIN data. This is a simpler case of the ACS equations, where initial nutrient concentration is uniform. Equations 3.10 give the exact generating functions.

$$\begin{cases} \frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5 * S_{ij} \\ \frac{dS_{ij}}{dt} = -2 \cdot 10^{-7} \frac{dN_{ij}}{dt} \end{cases} \tag{3.10}$$

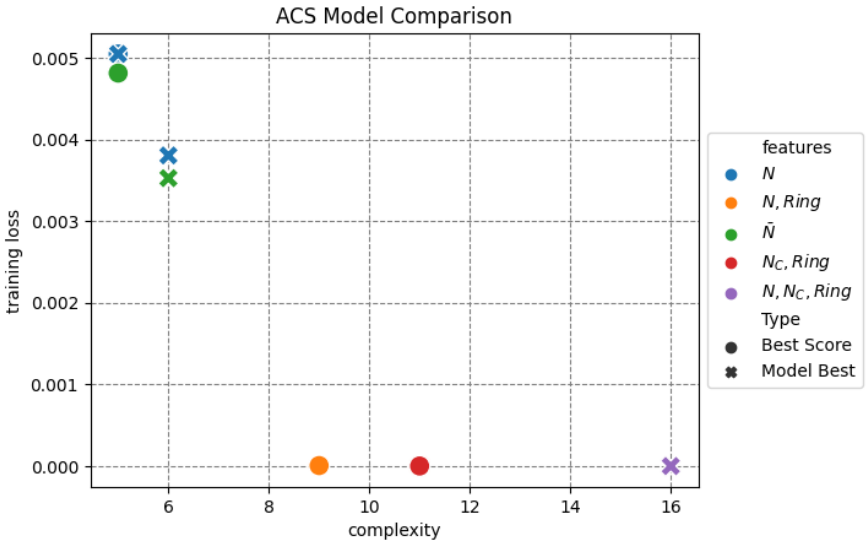

Figure 3.7: Comparison of the different models for the ACS dataset. The "type" refers to how that equation was selected from the model's output. Loss is L2 training loss and complexity is number of nodes in the expression tree.

The initial populations of the colonies can be seen in figure 3.2. The initial nutrient concentration is 1 for all colonies.

By comparing the models in figure 3.9, we notice that N already explains much of the variance in the data, as the scale is in the order of 1e−6. We also see that the Best Score expression using feature N_C performs quite well. However, we found that there is another expression using N_C that was much less complex. This is labeled as type "Hand Pick."

Although feature-set N performs well, systemic bias from initial values is still present, as seen in figure 3.10. It follows, then, that introducing the N_0 or N_C variable will reduce this since they both carry the information given by the initial value.

The regressor has parameterized the dynamics with respect to N_C in the same way it did for the ACS dataset. There is no longer a term that describes the variance between rings as it is unnecessary.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = 0.4911e^{-9.6460 \cdot 10^{-8} N_C} \tag{3.11}$$

Equation 3.11 is an exact solution to the dynamical system, as shown in equation 1.7.

3.4 Diffusion Model Dataset

We expect the Diffusion model to be the most difficult of the four to get right because of interaction dynamics. In equation 3.12, they are presented through $\bar{S}$, where the

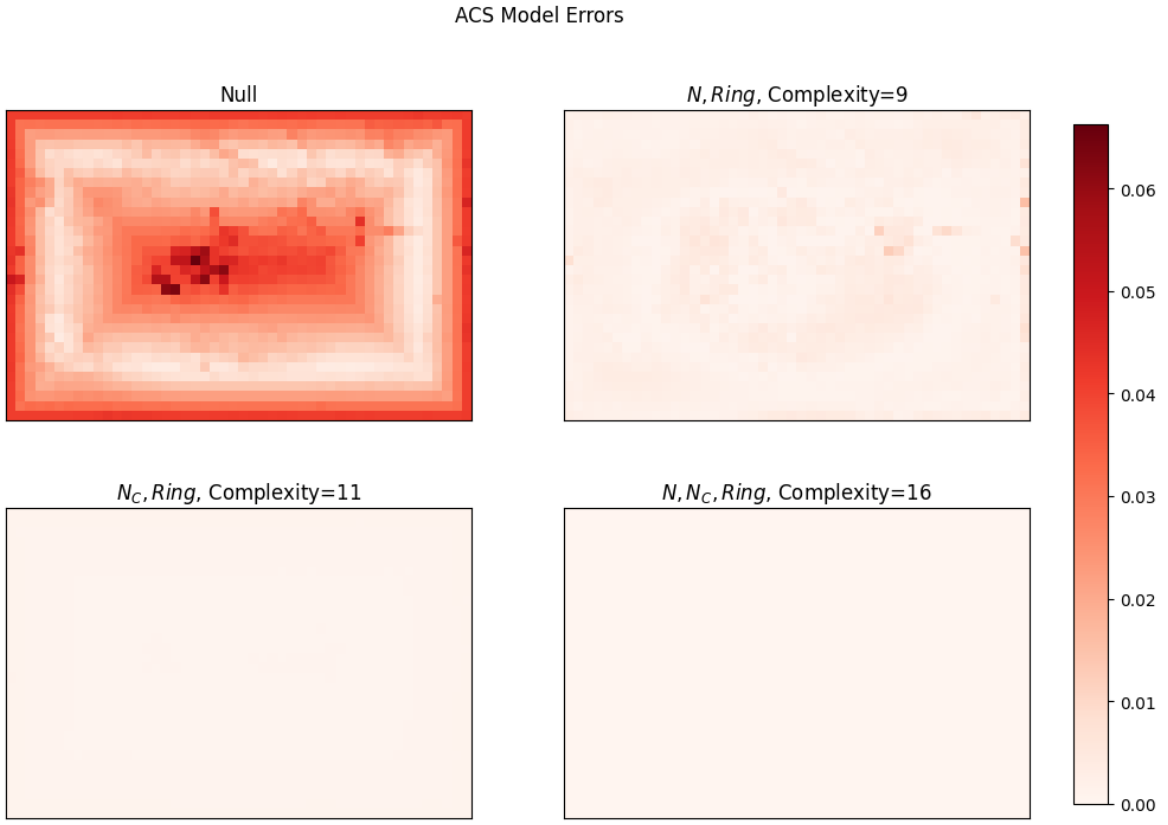

Figure 3.8: Comparison of colony average RMSE between models. We compare the models that use the feature-sets of $N, Ring$, $N_C, Ring$, and $N, N_C, Ring$. The latter uses both N and N_C as it is a result from the top-down approach.

rate of change of nutrient density is dependent on the density of the neighbors as well. Other than this term, however, the rest of the system of equations remains the same as in the VIN dataset.

$$\begin{cases} \frac{dN_{ij}}{dt \cdot N_{ij}} = 0.5 * S_{ij} \\ \frac{dS_{ij}}{dt} = -2 \cdot 10^{-7} \frac{dN_{ij}}{dt} + 0.1(\bar{S_{ij}} - S_{ij}) \end{cases} \tag{3.12}$$

After performing the greedy approach, it was clear that the cumulative variables had a large advantage. For this reason, we only compare models with a cumulative variable in the feature-set. Figure 3.11 shows the comparison of these models. We also hand picked an expression from the feature set of $N_C, \bar{N}_C$, that we thought balanced loss and complexity well.

We consider the simple N_C model of complexity 6; $N_C, \bar{N}_C$ model of complexity 13; and $N_C, \bar{N}_C$ model of complexity 18. We choose not to consider the $N, \bar{N}, N_C, \bar{N}_C$ model, even though it has the lowest training loss, as the feature set is large and difficult to interpret. Looking at figure 3.12, we see that by just using N_C, much of the variance is explained. However, there is a large amount of systemic bias towards the border

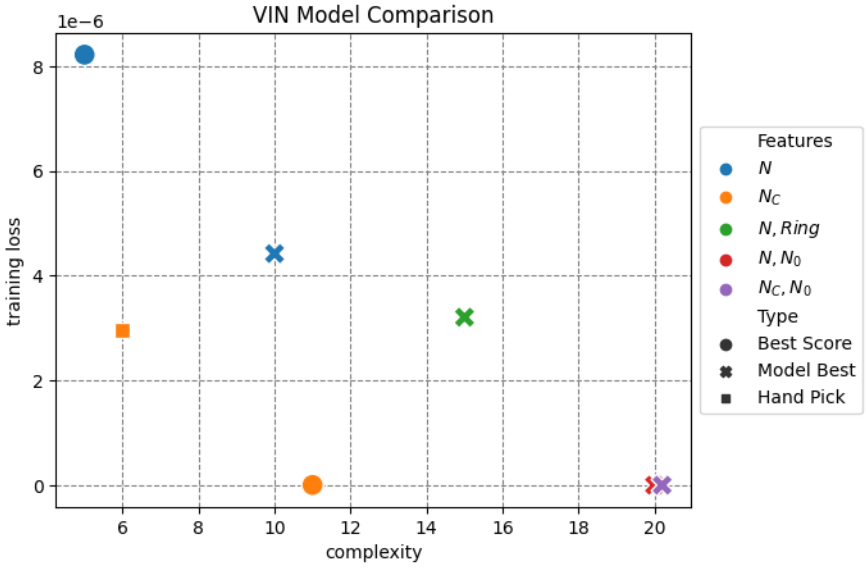

Figure 3.9: Comparison of the different models for the VIN dataset. The "type" refers to how that equation was selected from the model's output. Loss is L2 training loss and complexity is number of nodes in the expression tree

colonies. These are the colonies that benefit the most from the interaction dynamics. Introducing $\bar{N}_C$ greatly reduces the error around the border, although some colonies, including the corners, are still not explained well. This would suggest the presence of interactions as the colonies seem dependent on how fast the colonies around them are growing. We look at the 2 equations that use features $N_C, \bar{N}_C$.

Equation 3.13 is the simpler of the two that we observe. The first thing we notice is that one of the terms is virtually identical to that found in both equations 3.6 and 3.11 in the ACS and VIN datasets respectively. This leaves us able to interpret the equation better. It finds dynamics that would explain the dataset without any interactions and then corrects for the interaction by adding the second term, which uses $\bar{N}_C$. We will refer to this as the interaction term. In equation 3.13, as $\bar{N}_C$ grows larger relative to N_C the interaction term grows smaller. The use of cumulative variables here means that the interaction term is more sensitive to changes in population density in the early stages of growth. As they always hold the history of all past population densities. Moreover, it suggests that the early stages of growth will play a large role in defining the behavior of later stages. This is consistent with existing theory from [2] that places importance on the cell growth history.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = e^{-\frac{8.1950\bar{N}_C}{N_C}} + 0.4882e^{-9.5115 \cdot 10^{-8} N_C} \tag{3.13}$$

Equation 3.14 is slightly more complex but is still in the form of an interaction term plus the general growth term. It also uses the same logic that as $\bar{N}_C$ increases

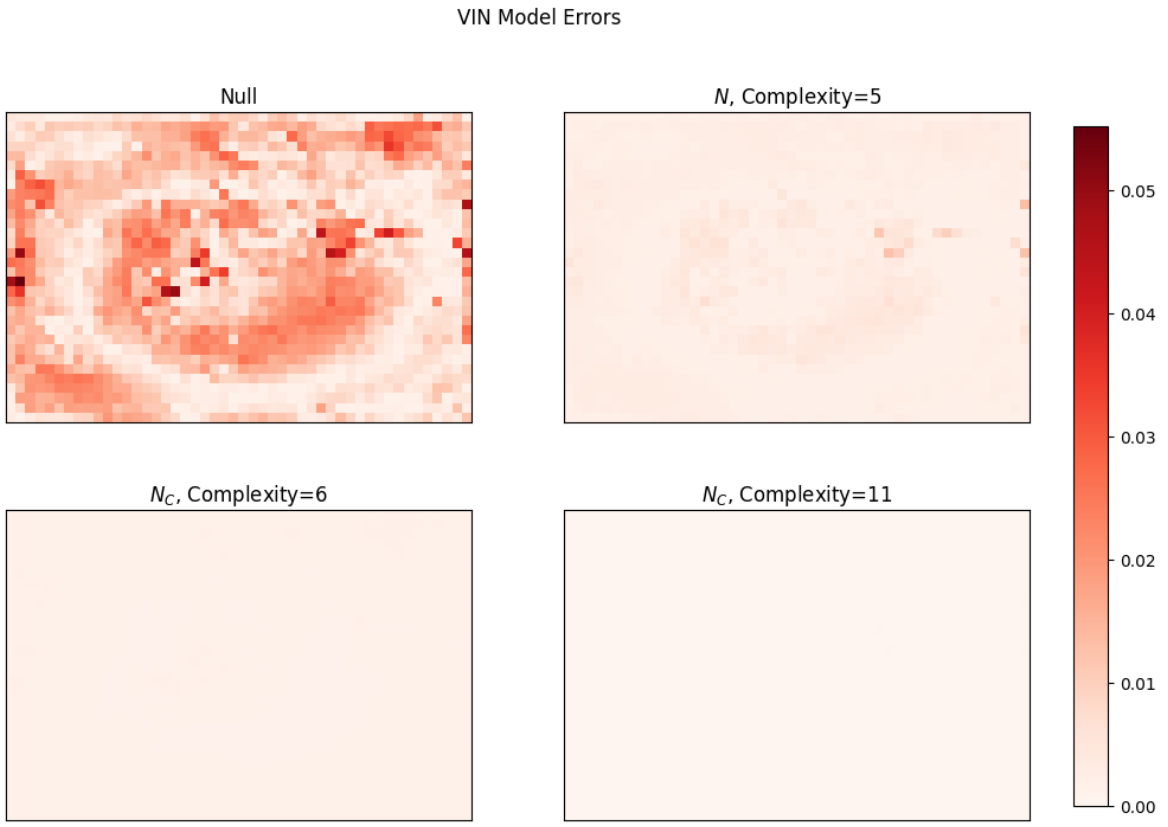

Figure 3.10: Comparison of colony average RMSE between models. We compare the models that use the feature-sets of N, and N_C.

relative to N_C the interaction term decreases. However, this function allows for negative per-capita growth rates, which is not possible under the true dynamics. For this reason, we turn back to equation 3.13 to interpret further.

$$\frac{dN_{ij}}{dt \cdot N_{ij}} = \frac{0.0195N_C - 0.0195\bar{N}_C}{\bar{N}_C + 805184.3790} + 0.4897e^{-9.5115 \cdot 10^{-8} N_C} \tag{3.14}$$

Figure 3.13 shows a heatmap of equation 3.13, where the values are given colors according to a logarithmic scale. We also plot traces of various colonies from the border and middle of the plate. These traces show how their N_C and $\bar{N}_C$ values change over time. For the corner colones, (0,0) and (31,47), the model actually predicts that their growth rate will start to increase at the end of the simulation, shown by the slight veer towards the asymptote $\frac{dN}{dt \cdot N} = 1$.

We are pleased with how the regressor was able to deduce the presence of interactions, and even suggest a diffusion mechanism via the relation between N_C and $\bar{N}_C$. We do note, however, that it is unclear how the diffusion parameter, D, in equations 3.12, presents itself in the solution. Further testing would be required to analyze its effect.

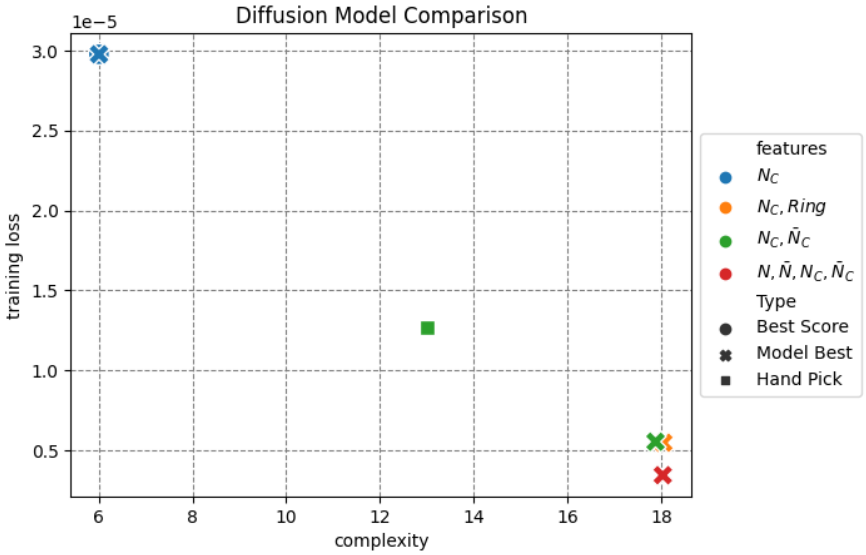

Figure 3.11: Comparison of the different models for the DIFF dataset. The "type" refers to how that expression was selected from the model's output. Loss is L2 training loss and complexity is number of nodes in the expression tree

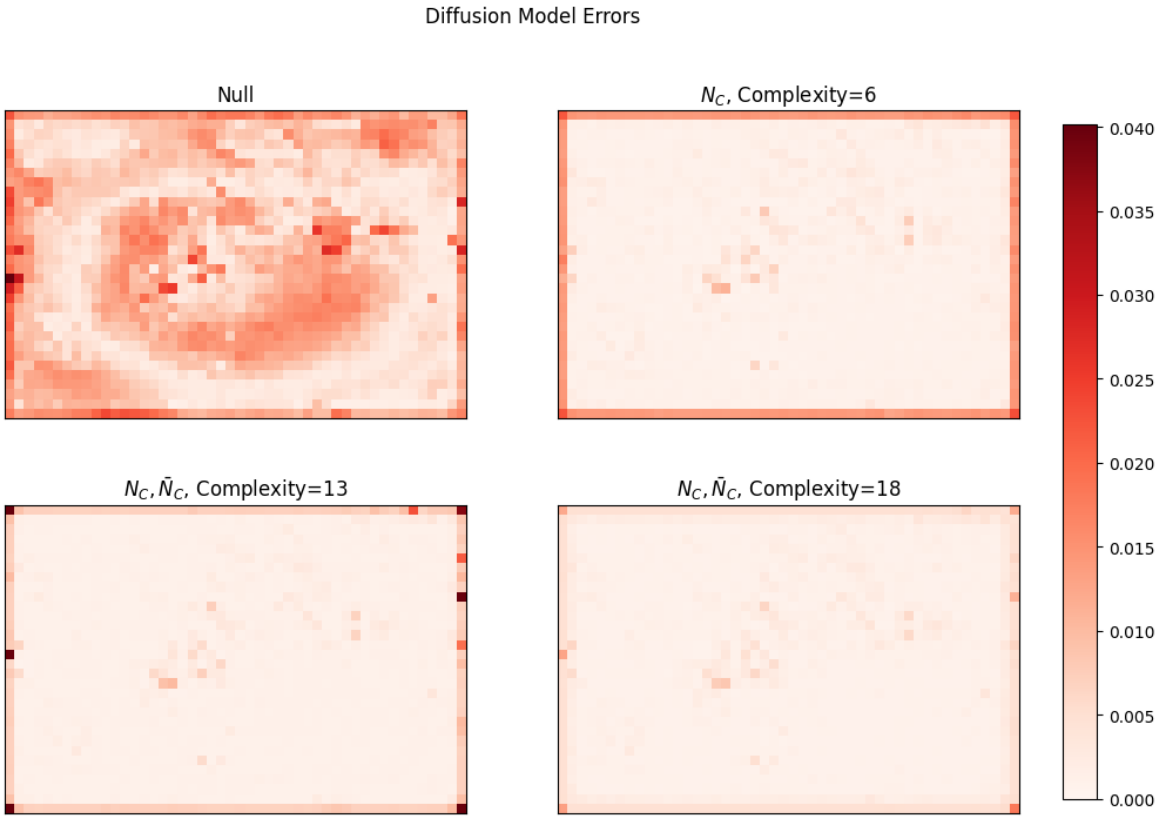

Figure 3.12: RMSE by plate for 3 different models on Diffusion dataset. Labeled by features used and complexity.

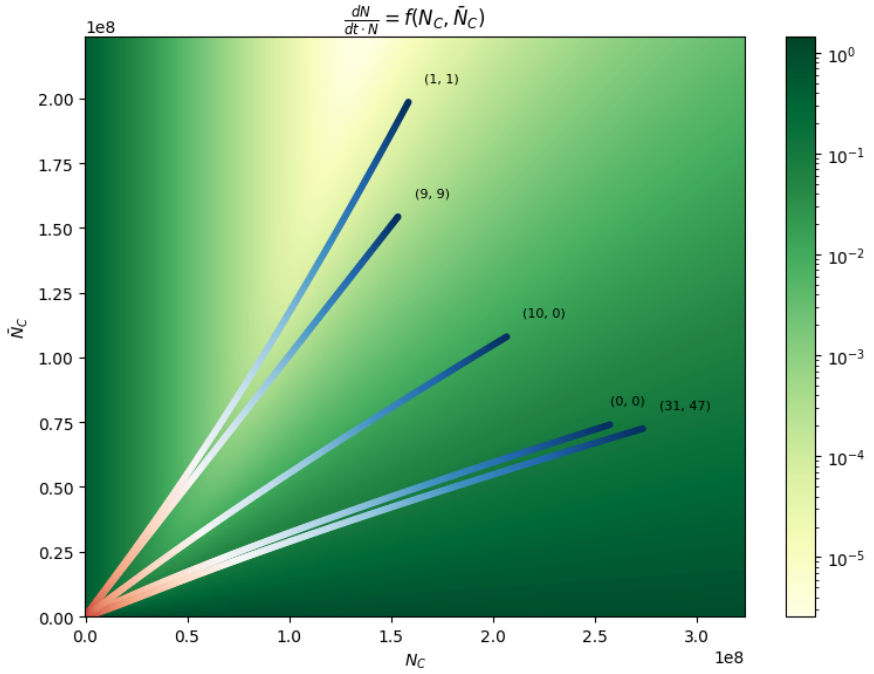

Figure 3.13: We plot a heatmap of the per-capita growth rate according to equation 3.13. The x-axis is N_C and the y-axis is $\bar{N}_C$. The rate is on a logarithmic scale to reflect all values between 0 and 1. We also show scatter plots of a few colonies to see how the function would predict them. The red-blue gradient shows the time step of the values. We annotate the traces with their position on the plate. (0,0) and (31,47) are corner colonies for which the model does not explain the variance well. (10,0) is a border colony for which the model performs better but still not perfect. (1,1) and (9,9) are non-border colonies for which the model performs very well, correctly predicting how fast their growth rate tends to 0.

4. Conclusions

Symbolic regression proved useful in finding ways to explain the dynamics of our datasets. Even with the nutrient concentration hidden, the regressor still found ways to express the underlying dynamics. Both by finding the analytical solution to our proposed ODE as in the case of the VIN dataset; and by turning to new variables that we form via transformation of the raw data as in the case of the diffusion dataset.

4.1 Challenges

The largest obstacle to overcome is the lack of data about nutrient dynamics. This means that we have to assume the dynamics will not be able to be reverse-engineered exactly from the data via SR. Instead, we must observe all the models and make our choice based on how well they perform and their interpretability. In the case of models that perform similarly but behave differently, the challenge lies in using domain knowledge to consider both of them and pick the more reasonable one. This is non-trivial, and in the diffusion dataset, we saw a few plausible terms that described the variance of the dataset well. We ultimately narrowed it down to one based on criteria that we felt made the most sense. However, even within the same feature-set it is not clear which equation performs best, as we have to consider the problem of overfitting as complexity grows (and training loss shrinks). We used two criteria, the best score and the model's pick, to automate the equation selection from a single model output. Although these tend to be the best equations, we resorted to manual selection several times to find the most interpretable ones.

In all of our datasets the position of the colony played a role. This was either done by setting the initial conditions or by using dynamics that gave an inherent advantage to some colonies because of their position (Diffusion dataset). Although the *Ring* variable served to be very useful in our ACK and ACS datasets, it did not prove to be useful in the Diffusion or VIN dataset. This follows logically as *Ring* was not used in their data generation. Codifying the position into a variable to pass into the regressor can be done in many ways, and it is uncertain *a priori* which is the best. We explored only two in this thesis, with the variables *Ring* and $\bar{N}$. It remains to be a

problem-specific challenge in how to incorporate position in the regressor.

4.2 Further Developments

In our experiments we avoided passing any nutrient data to the regressor. However, we could also take the reverse approach, by generating many sets of our own nutrient data, with different dynamics and parameters. After fitting the model to explain the per-capita growth rate for each dataset, we would look to the best performing ones and consider those parameters and dynamics as possible solutions.

None of our datasets contain added noise, which has proven to be challenging for SR [12]. In addition, noise is prevalent in microbial growth data that comes from real lab experiments. It is uncertain how the regressor would fare with this data, and which features of the data would be impacted most by the noise. This remains an open problem we hope to work on in the future.

We have also focused on only one way to do symbolic regression without real consideration of other methods. QLattice [5] has shown great results in benchmark tests [12] using GP and probabilistic machine learning. In general, it would be useful to see how modern implementations perform relative to each other as the field grows. The application of using SR on GNN components as done in [11] also looks promising considering that our experimental setup lends itself nicely to be formatted as a graph; where, for example, the colonies could be nodes and the flow of nutrients the edges.

4.3 Acknowledgments

This thesis was written as part of the Bioinformatics and Evolution Group at University of Helsinki. From which, Professor Ville Mustonen and PhD candidate Dovydas Kičiatovas were especially helpful; offering feedback and help throughout the research and writing process. The model fitting was all done on the Puhti cluster, operated by CSC. Lastly, we would like to thank Miles Cranmer for his work on the open-source PySR, our software of choice.

Bibliography

[1] R. J. Allen and B. Waclaw. Bacterial growth: a statistical physicist's guide. *Reports on Progress in Physics*, 82(1):016601, Nov. 2018.

[2] J. Baranyi and T. A. Roberts. A dynamic approach to predicting bacterial growth in food. *International Journal of Food Microbiology*, 23(3):277–294, 1994. Special Issue Predictive Modelling.

[3] A. A. Berryman. The orgins and evolution of predator-prey theory. *Ecology*, 73(5):1530–1535, 1992.

[4] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 104:9943–8, 07 2007.

[5] K. R. Broløs, M. V. Machado, C. Cave, J. Kasak, V. Stentoft-Hansen, V. G. Batanero, T. Jelen, and C. Wilstrup. An approach to symbolic regression using feyn, 2021.

[6] P. Cardoso, V. V. Branco, P. A. V. Borges, J. C. Carvalho, F. Rigal, R. Gabriel, S. Mammola, J. Cascalho, and L. Correia. Automated discovery of relationships, models, and principles in ecology. *Frontiers in Ecology and Evolution*, 8, 2020.

[7] Y. Chen, M. T. Angulo, and Y.-Y. Liu. Revealing complex ecological dynamics via symbolic regression. *BioEssays*, 41(12):1900069, 2019.

[8] R. F. Costantino, R. A. Desharnais, J. M. Cushing, and B. Dennis. Chaotic dynamics in an insect population. *Science*, 275(5298):389–391, 1997.

[9] M. Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, Sept. 2020.

[10] M. Cranmer. Interpretable Machine Learning for Science with PySR & SymbolicRegression.jl, May 2023.

[11] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17429–17442. Curran Associates, Inc., 2020.

[12] F. O. de Franca, M. Virgolin, M. Kommenda, M. S. Majumder, M. Cranmer, G. Espada, L. Ingelse, A. Fonseca, M. Landajuela, B. Petersen, R. Glatt, N. Mundhenk, C. S. Lee, J. D. Hochhalter, D. L. Randall, P. Kamienny, H. Zhang, G. Dick, A. Simon, B. Burlacu, J. Kasak, M. Machado, C. Wilstrup, and W. G. L. Cava. Interpretable symbolic regression for data science: Analysis of the 2022 competition, 2023.

[13] S. Gaucel, M. Keijzer, E. Lutton, and A. Tonda. Learning dynamical systems using standard symbolic regression. In M. Nicolau, K. Krawiec, M. I. Heywood, M. Castelli, P. García-Sánchez, J. J. Merelo, V. M. Rivas Santos, and K. Sim, editors, *Genetic Programming*, pages 25–36, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[14] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. San Francisco, CA: Morgan Kaufmann, 1991.

[15] J. H. Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[16] Y. Jin, W. Fu, J. Kang, J. Guo, and J. Guo. Bayesian symbolic regression, 2020.

[17] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9):4166–4177, 2021.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[19] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, Jun 1994.

[20] W. La Cava, P. Orzechowski, B. Burlacu, F. de Franca, M. Virgolin, Y. Jin, M. Kommenda, and J. Moore. Contemporary symbolic regression methods and their relative performance. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[21] M. Landajuela, C. Lee, J. Yang, R. Glatt, C. P. Santiago, I. Aravena, T. N. Mundhenk, G. Mulcahy, and B. K. Petersen. A unified framework for deep symbolic regression. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[22] B. T. Martin, S. B. Munch, and A. M. Hein. Reverse-engineering ecological theory from data. *Proceedings of the Royal Society B: Biological Sciences*, 285(1878):20180422, May 2018.

[23] M. Mollison and S. Stone. Interpreting machine learning models, 2017. Accessed on May 29, 2023.

[24] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

[25] M. Quade, M. Abel, K. Shafi, R. K. Niven, and B. R. Noack. Prediction of dynamical systems by symbolic regression. *Phys. Rev. E*, 94:012214, Jul 2016.

[26] Y. Ram, E. Dellus-Gur, M. Bibi, K. Karkare, U. Obolski, M. W. Feldman, T. F. Cooper, J. Berman, and L. Hadany. Predicting microbial growth in a mixed culture from growth curve data. *Proceedings of the National Academy of Sciences*, 116(29):14698–14707, 2019.

[27] P. A. K. Reinbold and R. O. Grigoriev. Data-driven discovery of partial differential equation models with latent variables. *Physical Review E*, 100(2), aug 2019.

[28] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[29] S. S. Sahoo, C. H. Lampert, and G. Martius. Learning equations for extrapolation and control. In *Proc. \35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 2018*, volume 80, pages 4442–4450. PMLR, 2018.

[30] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[31] M. D. Schmidt, R. R. Vallabhajosyula, J. W. Jenkins, J. E. Hood, A. S. Soni, J. P. Wikswo, and H. Lipson. Automated refinement and inference of analytical models for metabolic networks. *Phys. Biol.*, 8(5):055011, Oct. 2011.

[32] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[33] S.-M. Udrescu and M. Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[34] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation*, 29(2):211–237, 2021.

[35] M. Virgolin and S. P. Pissis. Symbolic regression is np-hard, 2022.

[36] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[37] D. Wadekar, L. Thiele, F. Villaescusa-Navarro, J. C. Hill, M. Cranmer, D. N. Spergel, N. Battaglia, D. Anglés-Alcázar, L. Hernquist, and S. Ho. Augmenting astrophysical scaling relations with machine learning: Application to reducing the sunyaev-zeldovich flux-mass scatter. *Proceedings of the National Academy of Sciences*, 120(12):e2202074120, 2023.

[38] M. Zackrisson, J. Hallin, L. G. Ottosson, P. Dahl, E. Fernandez-Parada, E. m, L. Fernandez-Ricaud, P. Kaferle, A. Skyman, S. Stenberg, S. Omholt, U. č, J. Warringer, and A. Blomberg. Scan-o-matic: High-Resolution Microbial Phenomics at a Massive Scale. *G3 (Bethesda)*, 6(9):3003–3014, Sep 2016.

Appendix A. Model Example

The following model configuration is what we use for all runs of the symbolic regressor.

```
Nmodel = PySRRegressor(
    equation_file = "output.csv",
    procs=32,
    early_stop_condition = 2e-09,
    timeout_in_seconds = 80000,
    model_selection="best",
    niterations=1e10,
    binary_operators=["+","-", "*", "/"],
    unary_operators=["exp"],
    constraints={"exp":5},
    nested_constraints={"exp":{"exp":0}},
    populations = 96,
    loss = "L2DistLoss()",
    ncyclesperiteration = 5000,
    maxsize = 25,
    maxdepth = 10,
    parsimony = 4e-10,
    weight_optimize = 0.001,
    turbo = True,
    progress = False
    )
```