Using Domain Knowledge on Population Dynamics Modeling for Equation Discovery

Ljupčo Todorovski, Sašo Džeroski

Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
Ljupco.Todorovski@ijs.si, Saso.Dzeroski@ijs.si

Abstract. State of the art equation discovery systems are concerned with the empirical approach to modeling of physical systems, where none or a very limited portion of the expert knowledge about the observed system is used in the modeling process. In this paper, we propose a formalism for integration of the population dynamics modeling knowledge into the process of equation discovery. The formalism allows the encoding of a high-level domain knowledge accessible to human experts. The encoded knowledge can be automatically transformed into the operational form of context dependent grammars. We present an extended version of the equation discovery system LAGRAMGE that can use these context free grammars. Experimental evaluation shows that the integration of domain knowledge in the process of equation discovery considerably improves the efficiency and noise robustness of LAGRAMGE.

1 Introduction

Most of the work in scientific discovery [4] is concerned with assisting the empirical approach to modeling of physical systems. Following this approach, the observed system is modeled on a trial-and-error basis to fit observed data. None or a very limited portion of the available domain background knowledge about the observed system is used in the modeling process. This is especially the case in domains where a limited amount of knowledge is expressed in the form of mathematical laws, such as biology, medicine and other life sciences. The empirical approach is in contrast to the theoretical approach to modeling, where the basic physical processes involved in the observed system are first identified. Then, the human expert uses the domain knowledge about the identified processes to write down a proper structure of the model in the form of differential equations. Finally, the values of the constant parameters of these equations are fitted against the observed data using system identification methods [5].

The focus of this paper is on integrating expert theoretical knowledge about the domain of interest within the process of automated modeling of population dynamics. The equation discovery system LAGRAMGE [7] has made initial steps toward this integration. It allows the user to define the space of possible equations using a context free grammar, written on the basis of the user background knowledge about the domain at hand. However, one can argue that it is difficult to encode a context free grammar from the expert domain knowledge. In this paper, we propose a formalism for encoding population dynamics modeling knowledge that is more accessible to human experts. It allows an automated generation of a grammar for equation discovery. The generated grammar is context dependent (and not context free as in Lagramge), so Lagramge 2.0 was developed that, among other improvements, allows the use of context dependent constraints in the grammar specifying the space of possible equations. The experimental evaluation of the proposed framework shows that integrating expert knowledge within the process of equation discovery considerably improves the efficiency and noise robustness of Lagramge.

The paper is organized as follows. Section 2 gives a brief introduction to equation discovery. The basics of population dynamics modeling are introduced in Section 3. The formalism for encoding the population dynamics modeling knowledge and the process of its transformation into a grammar for equation discovery are presented in Section 3. The necessary improvements of Lagramge are presented in Section 4. The experimental evaluation of Lagramge 2.0 is given in Section 5. The last Section 6 summarizes the paper and gives directions for further work.

2 Equation Discovery

Equation discovery is the area of machine learning that develops methods for automated discovery of quantitative laws, expressed in the form of equations, in collections of measured data [4]. It is related to the area of system identification. However, mainstream system identification methods work under the assumption that the structure of the model, i.e., the form of the equations, is known and are concerned with determining the values of the constant parameters in the model [5]. Equation discovery systems, on the other hand, aim at identifying both an adequate structure of the equations and appropriate values of the constant parameters.

2.1 Background Knowledge and Language Bias

Equation discovery systems search through the space of possible equation structures. Most of the equation discovery systems emulate the empirical approach to scientific discovery: different equation structures are generated and fitted against measured data. However, some of the possible equation structures may be inappropriate for modeling the observed system. For example, consider the case where the measured variables of the observed system are not dimensionless. In that case some algebraic combinations of the system variables, such as addition (or subtraction) of mass and energy, are not valid. Beyond this simple example, there are also more sophisticated inconsistencies of equation structures with a background knowledge from the domain of the observed system.

Different equation discovery systems explore different spaces of possible equation structures, or in other words they use different language biases. One possibility is to use some pre-defined (built-in) language bias that restricts the space of possible equation structures to some reasonably small class, such as polynomials or trigonometric functions, like in Lagrange [2]. In this case, the user can not influence the space of possible equations or use domain specific knowledge in the process of equation discovery. It is much better to use a declarative bias approach, where the user is allowed to influence or directly specify the space of possible equations. This approach provides users with a tool for incorporating their background knowledge about the domain at hand in the process of equation discovery. The use of background knowledge in the sense of a declarative language bias can avoid the problems of inconsistency of the discovered equations with the knowledge about the domain of the observed system, mentioned above.

Several equation discovery systems make use of domain specific knowledge. In equation discovery systems that are based on genetic programming, the user is allowed to specify a set of algebraic operators that can be used. A similar approach has been used in the EF [10] equation discovery system. The equation discovery system SDS [8] effectively uses scale-type information about the dimensions of the system variables and is capable of discovering complex equations from noisy data.

However, expert users can usually provide much more modeling knowledge about the domain at hand than merely enumerating the algebraic operators to be used or (the scale-type of) dimensions of the measured system variables. In order to incorporate this knowledge in the process of equation discovery, we should provide the user with a more sophisticated declarative bias formalism. In LAGRAMGE [7], the formalism of context free grammars has been used to specify the space of possible equations. Note here that context free grammars are a far more general and powerful mechanism for incorporating domain specific knowledge than the ones used in SDS [8] and EF [10].

The use of declarative bias in the form of context free grammar was crucial for modeling the phytoplankton growth in Lake Glumsoe in Denmark from real-world sparse noisy measurements [3]. However, one can argue that it is difficult for the users of LAGRAMGE to express their knowledge about the domain in the form of a context free grammar. In this paper, we present a formalism for encoding the domain knowledge at a higher, more user-friendly level, which can be automatically transformed to the operational form of grammars for equation discovery.

2.2 Discovery of Differential Equations

In this paper, we consider the problem of modeling dynamic systems, i.e. systems that change their state over time. Differential equations are the most common tool for modeling dynamic systems. LAGRANGE [2] was the first equation discovery system that extended the scope of equation discovery systems to ordinary differential equations. The basic idea was to introduce the time derivatives of the systems variables through numerical differentiation and then search for algebraic equations. This simple approach has a major drawback: large errors are introduced by numerical differentiation.

The problem was partly resolved in LAGRAMGE, where numerical integration is used instead of differentiation for the highest-order derivatives [7]. However, LAGRAMGE is only capable of discovering a differential equation for a single user specified system variable. In order to discover a system of simultaneous differential equations, LAGRAMGE has to be invoked once for each system variable.

3 Population Dynamics Modeling

Population ecology studies the structure and dynamics of populations, where each population is a group of individuals of the same species inhabiting the same area. In this paper we consider modeling the dynamics of populations, especially the dynamics of change of their density. We consider models of predator-prey population dynamics, where the interaction between predator and prey is antagonistic in the sense that it causes increase of the predator population and decrease of the prey population. The models are systems of differential equations [6].

3.1 Generalized Volterra-Lotka Model of Population Dynamics

Consider a simple model based on two populations, foxes and rabbits. The latter are grazing on grass and the foxes are carnivores that hunt rabbits. We assume that rabbits are the only food of foxes, rabbits have unlimited supply of grass and seasonal changes are ignored. Under these assumptions, if the rabbit population is large, the fox population grows rapidly. However, this causes many rabbits to be eaten, thus diminishing the rabbit population to the point where the food for foxes is not sufficient. Consequently, the fox population decreases, which causes faster growth of the rabbit population.

The oscillatory behavior of the two population densities can be modeled using the Volterra-Lotka population dynamics model [6]. It can be generalized using the following schema:

$$\dot{N} = \mathit{growth\_rate}(N) - \mathit{feeds\_on}(P, N)$$
$$\dot{P} = \mathit{feeds\_on}(P, N) - \mathit{decay\_rate}(P),$$

where N is the prey (rabbit) population density and P is the predator (fox) population density.

Using this model schema, we can build models of predator-prey population dynamics with different complexity. The term $\mathit{growth\_rate}(N)$ defines the model of the prey population growth in absence of predation. Two models of single population growth are usually used [6]: $(a)\ aN$; and $(b)\ aN(1 - N/K)$. The first model (a) assumes that the population growth is exponential and unlimited. However, there are real-world environments that have some carrying capacity for the population, which limits the density of the population. In such cases, an alternative logistic growth model (b) can be used, where K is a constant, determining the carrying capacity of the environment.

The second assumption made in simple population models is that the predation rate is proportional to the densities of predator and prey populations ($\mathit{feeds\_on}(N, P) = bPN$). As for population growth, this means that the predation growth is exponential and unlimited. Again, in some cases the predators have limited predation capacity. When the prey population density is small the predation rate is proportional to it, but when the prey population becomes abundant, the predation capacity saturates to some limit value ($\mathit{feeds\_on}(N, P) = bPs(N)$). Three different terms are often used to model the predator saturation response to the increase of the prey density $(a)\ AN/(N + B)$, $(b)\ AN^2/(N^2 + B)$ and $(c)\ A(1-e^{-BN})$, where A is the limit value of the predation capacity saturation, and B is the constant, which determines the saturation rate [6].

The modeling knowledge about population growth and saturation presented here can be very useful as a background knowledge for automated modeling of ecological systems with equation discovery.

3.2 Encoding of Domain and Modeling Knowledge

The knowledge about modeling population dynamics can be divided in two types. The first type is domain specific knowledge about the populations and their role in the food-chain. In the Volterra-Lotka model of population dynamics, this knowledge is represented by the single fact that foxes feed on rabbits. This type of knowledge can be expect from a biologist without any experience in mathematical modeling of population dynamics.

The second type of knowledge is domain independent knowledge about population dynamics modeling, presented in the Section 3.1. This type of knowledge can be provided by, say a mathematician with modeling experience, who is not necessary familiar with the biological ecosystem structure and the interactions involved.

Domain Specific Knowledge We first provide a specification of the food-chain in the domain: for our example rabbits and foxes domain it is given in Table 1. We use three first-order predicates: the `domain(domain_name)` predicate is used to specify the name of the domain at hand; each population in the domain is specified using the predicate `population(domain_name, population_name)`; finally, we use the predicate `feeds_on(domain_name, predator_population, prey_population)` to specify each interaction between two populations. For now, only predator-prey interactions can be specified. However, the formalism can be easily generalized to allow the specification of other types of interactions between populations, such as parasytism, competitive exclusion and symbiosis [6].

Table 1. Description of a simple Volterra-Lotka population dynamics domain consisting of two populations of foxes (predators) and rabbits (preys).

```
domain(vl).
population(vl, fox).
population(vl, rabbit).    feeds_on(vl, fox, rabbit).
```

Note that by using the predicates `population` and `feeds_on`, the user is allowed to specify an arbitrary number of populations and predator-prey interactions between them.

Domain Independent Modeling Knowledge The second part of the modeling knowledge, which is domain independent, is given in Table 2. We use the predicate `template` to specify a set of alternative models for population dynamics processes like population growth and saturation, described in Section 3. Note that the symbol `const` is used to specify a constant parameter, whose value has to be fitted against measured data. A constraint of the form `[L:U]` can be assigned to each constant parameter specifying that the value v of the constant parameter should be within the interval $\mathtt{L} \leq v \leq \mathtt{U}$. Omitting the `U` (`L`) value in this constraint means that there is no upper (lower) bound on the constant parameter value. For example, the symbol `const[0:]` means that the constant parameter should be non-negative.

Table 2. Templates with alternative sub-expressions used for modeling the processes of saturation and population growth.

```
template(saturation, X, (X)).
template(saturation, X, (X / (X + const[0:]))).
template(saturation, X, (X * X / (X * X + const[0:]))).
template(saturation, X, (1 - exp(-const[0:] * X))).
template(growth, X, (const * X)).
template(growth, X, (const * X * (1 - X / const[0:]))).
```

Transforming the Background Knowledge into a Grammar Using the definitions of the background knowledge from Tables 1 and 2, a grammar for equation discovery can be automatically generated. The process of transformation of the knowledge into grammar is automated using the predator-prey model schema presented in Section 3. The grammar for the example population dynamics domain, consisting of rabbits and foxes, is given in Table 3.

Table 3. A grammar for equation discovery constructed from the background knowledge in Tables 1 and 2.

```
vl -> time_deriv(rabbit) = growth(rabbit) - feeds_on(fox,rabbit);
    time_deriv(fox) = feeds_on(fox,rabbit) - const * fox
feeds_on(fox,rabbit) -> const * fox * nutrient(rabbit)
nutrient(rabbit) -> rabbit
nutrient(rabbit) -> rabbit / (rabbit + const[0:])
nutrient(rabbit) -> rabbit * rabbit / (rabbit * rabbit + const[0:])
nutrient(rabbit) -> 1 - exp(-const[0:] * rabbit)
growth(rabbit) -> const * rabbit
growth(rabbit) -> const * rabbit * (1 - rabbit / const[0:])
```

The starting non-terminal symbol in the grammar `vl` is used to generate the system of two differential equations of the population dynamics model, using the schema from Section 3. The growth of the rabbit population in the absence of predation is modeled using the non-terminal symbol `growth(rabbit)` with two alternative productions, reflecting the two `template` predicates for growth from Table 2. The third non-terminal symbol `feeds_on(fox,rabbit)` models the predation of foxes on rabbits. The predation rate is always proportional to the density of the fox population and the non-terminal `nutrient(rabbit)` is used to introduce the model of predator response to the increase of rabbit population density (the productions reflect the templates from Table 2). The terminal symbols `fox` and `rabbit` are used to introduce the measured system variables of the population densities.

Strictly speaking, the grammar in Table 3 is not context free. Namely, the production for the starting symbol `vl` generates two `feeds_on(fox,rabbit)` symbols, one in the first and another one in the second equation. In context free grammar, these two non-terminal symbols can generate two different expressions. In population dynamics models, however, these two expressions have to be the same. The use of context dependent constraints can overcome this limitation of the context free grammars.

4 LAGRAMGE 2.0

In order to use grammars like the one from Table 3 for equation discovery, we developed LAGRAMGE 2.0, an improved version of LAGRAMGE 1.0 [7]. Improvements were made in three directions. First, the context dependent constraints have to be checked for each expression. Second, the grammar in Table 3 generates all model equations at once, therefore a system of simultaneous equations has to be discovered, instead of a single equation for each system variable separately. Third, the constraints on the lower and upper bound of the values of the constant parameters have to be considered. The top-level algorithm of the LAGRAMGE 2.0 exhaustive search procedure is presented in Table 4.

Table 4. Exhaustive search procedure of LAGRAMGE 2.0.

procedure `LagramgeSearch`(V, V_d, $Data$, G, b)
1 $Q \leftarrow \{\}$
2 $S \leftarrow$ enumerate all derivation trees in G
3 **foreach** T **in** S **do**
4   **if** `CheckConstraints`(T, G) **then**
5   T.error $\leftarrow 0.0$
6   **foreach** v **in** V_d
7   T.error $\leftarrow$ T.error + `Fit`($\dot{v} = T.v$, $Data$)
8   **endfor**
9   **endif**
10   $Q \leftarrow Q \cup T$
12 **endwhile**
11 return the b best parse trees in Q

The search procedure of LAGRAMGE 2.0 takes as an input a set of variables V, a (sub)set of dependent variables $V_d \subseteq V$, a data set $Data$ with measured time behaviors of the variables in V, a (context dependent) grammar G, and a parameter b that determines the number of best models (systems of equations) returned as output of LAGRAMGE.

The search space of LAGRAMGE is the set of parse trees that can be derived with the user provided grammar G. The search space is ordered according to the height of the parse trees using the refinement operator defined in [7]. Starting with an empty parse tree, it can be repeatedly used to generate all parse trees.

Context Dependent Constraints Each generated parse tree is first checked to see if it satisfies the context dependent constraints in G (line 4 of the algorithm in Table 4). The user is allowed to specify an arbitrary number of context dependent constraints for each production in the grammar. Examples of productions with context dependent constraints are presented in Table 5.

Table 5. Examples of grammar productions with context dependent constraints.

```
E -> A + B, B - A { A.1 == A.2; }
E -> A + B, B - A { A.1 == A.2; B.1 == B.2; }
E -> A * E { A <= E; }
```

In the first production, a single constraint `A.1 == A.2` specifies that the first (`A.1`) and the second (`A.2`) occurrence of the symbol `A` on the right hand side of the production should generate the same sub-expression. For example, the expression `a1 + b1, b2 - a2` can not be derived using that production (`A.1 -> a1` is different from `A.2 -> a2`), whereas the expression `a1 + b1, b2 - a1` can. However, the latter expression can not be derived using the second production due to the second constraint `B.1 == B.2`. `a1 + b1, b1 - a1` is an example of an expression that can be derived using both productions.

The third production illustrates the use of a context dependent constraint to avoid redundant generation of expressions that are equivalent due to the commutativity of multiplication. The context free production `E -> A * E` can generate both `a * b` and `b * a`. On the other hand, using the context dependent constraint `A <= E` (where the operator `<=` stands for lexicographic comparison), the second expression `b * a` can not be derived.

Simultaneous Equations In order to evaluate a system of simultaneous equations for the user provided set of dependent variables V_d, the sub-trees T_v of the generated tree T are identified for each dependent variable $v \in V_d$. Then the error of each equation of the form $\dot{v} = T.v$ is evaluated (using the `Fit` function in line 7 of the algorithm in Table 4), where $T.v$ here denotes the expression derived by the sub-tree T_v. The errors of the equations for all dependent variables are added together to obtain the error of the whole parse tree T (lines 5-8).

Constraints on the Values of the Constant Parameters The function `Fit`(*equation*, *Data*) is used to fit the values of the constant parameters of the equation to the given data set *Data*. The discrepancy between the measured data *Data* and the data obtained by simulating the equation is used to evaluate the error of the equation. The nonlinear regression algorithm used in LAGRAMGE 1.0 can not impose any constraints on parameter values. Because of this, we replaced replaced it with the nonlinear regression algorithm proposed in [1]. The latter allows the use of simple constraints specifying the lower and upper bounds on the values of the constant parameters.

5 Experiments

The goal of the experiments with LAGRAMGE, presented in this section, is to evaluate the effect of using the new type of background knowledge in the process of equation discovery. For that purpose, we compared the performance of LAGRAMGE 2.0 with the performance of LAGRAMGE 1.0 on a task of reconstructing different models of a simple aquatic ecosystem consisting of three populations of inorganic nutrient, phytoplankton and zooplankton. The food-chain in the ecosystem contains two predator-prey interactions: zooplankton feeds on phytoplankton and phytoplankton feeds on inorganic nutrient.

5.1 Experimental Setup

Using the food-chain description along with the domain independent modeling templates from Table 2, the context dependent grammar presented in Table 6 has been built using the algorithm described in Section 3.2. The grammar in Table 6 generates thirty-two different models, i.e. systems of three simultaneous equations, which are used in the experiments. The experimental evaluation of

Table 6. A grammar for equation discovery in the aquatic ecosystem domain constructed from the food chain description and modeling knowledge from Table 2.

```
aquatic -> time_deriv(nut) = - feeds_on(phyto,nut);
    time_deriv(phyto) = growth(phyto) + feeds_on(phyto,nut)
                          - feeds_on(zoo,phyto);
    time_deriv(zoo) = const * zoo + feeds_on(zoo,phyto)
feeds_on(phyto,nut) -> const[0:] * phyto * nutrient(nut)
feeds_on(zoo,phyto) -> const[0:] * zoo * nutrient(phyto)
nutrient(nut) -> nut
nutrient(nut) -> nut / (nut + const[0:])
nutrient(nut) -> nut * nut / (nut * nut + const[0:])
nutrient(nut) -> 1 - exp(-const[0:] * nut)
nutrient(phyto) -> phyto
nutrient(phyto) -> phyto / (phyto + const[0:])
nutrient(phyto) -> phyto * phyto / (phyto * phyto + const[0:])
nutrient(phyto) -> 1 - exp(-const[0:] * phyto)
growth(phyto) -> const * phyto
growth(phyto) -> const * phyto * (1 - phyto / const[0:])
```

LAGRAMGE 2.0 consisted of attempting to reconstruct each of these 32 models from simulated data.

Using the grammar, we generated all thirty-two different model structures. In order to obtain simulation models, the values of the constant parameters have to be set. We used randomly generated values uniformly distributed on the $[0, 1]$ interval. We simulated each of the thirty-two obtained models from ten different randomly selected initial states (initial values of `nut`, `phyto` and `zoo`) for 100 time steps of 1. Thus, ten different behaviors were obtained of each of the thirty-two models.

In order to test the robustness of the approach to noise in the data, we added artificially generated random Gaussian noise to the behaviors. The noise was added at four different relative noise levels: 1%, 5% and 10%. The relative noise level of l% means that we multiplied the original value x with $(1+l*G/100)$ to obtain a noisy value, where G is a normally distributed random variable with mean equal to 0 and standard deviation equal to 1.

Two evaluation criteria were used for evaluating the performance of the equation (re)discovery. First, the leave-one-out procedure was applied in order to estimate the error of discovered equations on test data, unseen during the discovery process. In each iteration of the leave-one-out procedure, nine out of ten behaviors were used to discover a system of differential equations with LAGRAMGE. The obtained differential equations were then simulated using the initial state of the remaining (test) behavior. The simulation error was measured as a sum of squared differences between the simulated behavior and the test behavior. Second, the structure of the best model discovered by LAGRAMGE, was matched against the structure of the original model equations. The structure of the equations is obtained by abstracting the values of the constant parameters in them and replacing them with the generic symbol `const`.

5.2 Experimental Results

We compared the performance of (1) LAGRAMGE 2.0 using the context dependent grammar with the performance of (2) LAGRAMGE 2.0 using the grammar without constraints on the values of the constant parameters, and (3) LAGRAMGE 1.0 (where no context dependent constraints, and no constraints on the values of the constant parameters can be used). The results of the comparison are summarized in Tables 7 and 8.

Before we discuss the experimental results, we should note here that the using the context dependent constraints in the grammar reduces the space of possible models. The grammar in Table 6 generates thirty-two models when interpreted as a context dependent grammar. On the other hand, when interpreted as a context free grammar (as interpreted by LAGRAMGE 1.0) this grammar generates 512 possible models. Therefore, using context dependent constraints reduces the search space of LAGRAMGE by a factor of sixteen.

Table 7. Performance of LAGRAMGE 2.0 (L2.0), LAGRAMGE 2.0 without applying constraints on constant parameters (L2.0-NCC) and LAGRAMGE 1.0 (L1.0). Left hand side: average sum of squared errors on the test behavior. Right hand side: number of successfully reconstructed original model structures.

	AVERAGE TEST ERROR			STRUCTURE RECONSTRUCTION		
NOISE LEVEL	L2.0	L2.0-NCC	L1.0	L2.0	L2.0-NCC	L1.0
0%	0.0031	0.0020	0.0006	29	28	5
1%	0.0083	*(1)	*(10)	8	6	0
5%	0.1490	*(6)	*(13)	3	5	1
10%	0.6187	*(6)	*(13)	4	5	2

In the left hand side of Table 7 the average leave-one-out testing error of the thirty-two (re)discovered models is given. Note that the symbol *(N) means that N out of thirty-two (re)discovered models could not be simulated (and therefore the average error could not be properly evaluated) due to the singularities, such as division by zero or unstable behavior of the discovered system of differential equations. Some of the singularities were caused by inappropriate values of the constant parameters (e.g., negative saturation limit or carrying capacity): these are the reasons for the failure of LAGRAMGE 2.0 without applying the constraints on the values of the constant parameters to discover a valid model. In models discovered by LAGRAMGE 1.0, some of the simulation failures are also caused by inappropriate model structure, due to the lack of context dependent constraints.

These results shows one important aspect of the noise robustness of LAGRAMGE 2.0: at all noise levels it discovers models that can be simulated and have stable behaviors. This is due both to the context dependent constraints and the constraints on the values of the constant parameters. This is very important: in our earlier experiments on modeling phytoplankton growth in Lake Glumsoe, we had to manually filter out models, discovered by LAGRAMGE 1.0 with inappropriate values of the constant parameters [3].

In Table 8, the win-loss-tie counts for the comparison of the test error is presented. We counted the number of wins, losses and ties in the following manner. For each of the thirty-two experiments, we compared the simulation error

e_1 of LAGRAMGE 2.0 with the simulation error $e_2$of the other two algorithms (L2.0-NCC and L1.0 in Table 8). Comparisons where the relative difference of the simulation errors is less than 10% ($0.9 < e_2/e_1 < 1.1$) are considered ties.

Table 8. Win-loss-tie counts for comparison of the test error of (L2.0) with the test errors of L2.0-NCC and L1.0 (see caption of Table 7).

NOISE LEVEL	L2.0 VS. L2.0-NCC	L2.0 VS. L1.0
0%	10-9-13	26-2-4
1%	4-5-23	18-2-12
5%	7-12-13	16-6-10
10%	6-16-10	14-10-8

The comparison of simulation errors shows clear performance improvement of LAGRAMGE 2.0 over LAGRAMGE 1.0 for all noise levels. On the other hand, models discovered by LAGRAMGE 2.0 without applying the constraints on the values of the constant parameters better fits noisy data than the ones generated with LAGRAMGE 2.0. This observation shows that models with inappropriate values of the constant parameters (or inappropriate structure for models discovered by LAGRAMGE 1.0) can sometimes fit the observed data better. However, these models do not make sense from biological point of view.

6 Discussion

In the paper, a new approach to representing and using background knowledge for equation discovery is presented. The formalism for encoding knowledge about population dynamics, allows encoding two types of knowledge. The first type is domain specific knowledge about predator-prey food chain in the domain and can be provided by a biologist without any experience in mathematical modeling. The second type of knowledge is modeling knowledge in form of typical models used for modeling different population dynamic processes, such as growth of a population and saturation of predation. The modeling knowledge is provided by a population modeling expert, not necessary familiar with the domain at hand. This is high-level knowledge represented in first-order logic, which can be automatically transformed to the operational form of grammars used to guide the search for models in the process of equation discovery. This can be done for arbitrarily complex predator-prey models consisting of any number of populations and interactions between them. The proposed formalism can be easily extended with predicates for specifying other types of interactions between populations, such as parasytism, competitive exclusion and symbiosis.

The grammars generated using the presented approach are context dependent and generate context dependent and generate whole models, i.e., systems of simultaneous equations. The equation discovery system LAGRAMGE 2.0 was developed that accept such grammars. Using context dependent constraints reduces the space of possible models as compared to purely context free grammars used in LAGRAMGE 1.0. Therefore, context dependent constraints improve the efficiency of LAGRAMGE.

Experimental evaluation of LAGRAMGE 2.0 shows that both context dependent constraints and constraints on the values of the constant parameters improves noise robustness of LAGRAMGE in several ways. First, all the models (re)discovered by LAGRAMGE 2.0 at different noise levels can be simulated and generate stable behaviors. Second, all the models have clear interpretations from biological point of view. Finally, LAGRAMGE 2.0 (re)discovers the original model structure more often than LAGRAMGE 1.0. The experimental results should be further confirmed with experiments on the real-world observational data. These include modeling phytoplankton growth in Danish lake Glumsoe, predicting algae blooms in Lagoon of Venice [3] and modeling plankton population dynamics in Japanese lake Kasumigaura [9].

Currently, the presented formalism focuses on representing background knowledge for population dynamics modeling. However, the presented formalism can be extended so knowledge about dynamic processes from other areas can be encoded and used for equation discovery. The knowledge should include ontology of typical processes in the area (such as predator-prey interaction and population growth in population dynamics) and templates of models typically used for modeling these processes. Finally, knowledge from different areas can be organized in the form of libraries of background knowledge for equation discovery.

References

1. D. S. Bunch, D. M. Gay, and R. E. Welsch. Algorithm 717; subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. *ACM Transactions on Mathematical Software*, 19:109–130, 1993.
2. S. Džeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4:89–108, 1995.
3. S. Džeroski, L. Todorovski, I. Bratko, B. Kompare, and V. Križman. Equation discovery with ecological applications. In A. H. Fielding, editor, *Machine learning methods for ecological applications*, pages 185–207. Kluwer Academic Publishers, 1999.
4. P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Żythow. *Scientific Discovery.* MIT Press, 1987.
5. L. Ljung. Modelling of industrial systems. In *Proc. of Seventh International Symposium on Methodologies for Intelligent Systems*, pages 338–349. Springer, 1993.
6. J. D. Murray. *Mathematical biology.* Springer, 1993. Second, Corrected Edition.
7. L. Todorovski and S. Džeroski. Declarative bias in equation discovery. In *Proc. of the Fourteenth International Conference on Machine Learning*, pages 376–384. Morgan Kaufmann, 1997.
8. T. Washio and H. Motoda. Discovering admissible models of complex systems based on scale-types and identity constraints. In *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 810–817. Morgan Kaufmann, 1997.
9. P. A. Whigham. An inductive approach to ecological time series modelling by evolutionary computation. In *Abstract Book of the Second International Conference on Applications of Machine Learning to Ecological Modelling*, page 35. Adelaide University, 2000.
10. R. Zembowicz and J. M. Żytkow. Discovery of equations: Experimental evaluation of convergence. In *Proc. of the Tenth National Conference on Artificial Intelligence*, pages 70–75. Morgan Kaufmann, 1992.