Discovering Ecosystem Models from Time-Series Data

Dileep George,[1] **Kazumi Saito**,[2] **Pat Langley**,[1]
Stephen Bay,[1] and **Kevin R. Arrigo**[3]

[1] Computational Learning Laboratory, CSLI
Stanford University, Stanford, California 94305 USA
{dil,langley,sbay}@apres.stanford.edu

[2] NTT Communication Science Laboratories
2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan
saito@cslab.kecl.ntt.co.jp

[3] Department of Geophysics, Mitchell Building
Stanford University, Stanford, CA 94305 USA
arrigo@pangea.stanford.edu

Abstract. Ecosystem models are used to interpret and predict the interactions of species and their environment. In this paper, we address the task of inducing ecosystem models from background knowledge and time-series data, and we review IPM, an algorithm that addresses this problem. We demonstrate the system's ability to construct ecosystem models on two different Earth science data sets. We also compare its behavior with that produced by a more conventional autoregression method. In closing, we discuss related work on model induction and suggest directions for further research on this topic.

1 Introduction and Motivation

Ecosystem models aim to simulate the behavior of biological systems as they respond to environmental factors. Such models typically take the form of algebraic and differential equations that relate continuous variables, often through feedback loops. The qualitative relationships are typically well understood, but there is frequently ambiguity about which functional forms are appropriate and even less certainty about the precise parameters. Moreover, the space of candidate models is too large for human scientists to examine manually in any systematic way. Thus, computational methods that can construct and parameterize ecosystem models should prove useful to Earth scientists in explaining their data.

Unfortunately, most existing methods for knowledge discovery and data mining cast their results as decision trees, rules, or some other notation devised by computer scientists. These techniques can often induce models with high predictive accuracy, but they are seldom interpretable by scientists, who are used to different formalisms. Methods for equation discovery produce knowledge in forms that are familiar to Earth scientists, but most generate descriptive models rather than explanatory ones, in that they contain no theoretical terms and make little contact with background knowledge.

In this paper, we present an approach to discovering dynamical ecosystem models from time-series data and background knowledge. We begin by describing IPM, an algorithm for inducing process models that, we maintain, should be interpretable by Earth scientists. After this, we demonstrate IPM's capabilities on two modeling tasks, one involving data on a simple predator-prey ecosystem and another concerning more complex data from the Antarctic ocean. We close with a discussion of related work on model discovery in scientific domains and prospects for future research on the induction of ecosystem models.

2 An Approach to Inductive Process Modeling

As described above, we are interested in computational methods that can discover explanatory models for the observed behavior of ecosystems. In an earlier paper (Langley et al., in press), we posed the task of inducing process models from time-series data and presented an initial algorithm for addressing this problem. We defined a quantitative process model as a set of processes, each specifying one or more algebraic or differential equations that denote causal relations among variables, along with optional activation conditions. At least two of the variables must be observed, but a process model can also include unobserved theoretical terms.

The IPM algorithm generates process models of this sort from training data about observable variables and background knowledge about the domain. This knowledge includes generic processes that have a form much like those in models, in that they relate variables with equations and may include conditions. The key differences are that a generic process does not commit to specific variables, although it constrains their types, and it does not commit to particular parameter values, although it limits their allowed ranges. Generic processes are the building blocks from which the system constructs its specific models.

More specifically, the user provides IPM with three inputs that guide its discovery efforts:

1. A set of generic processes, including constraints on variable types and parameter values;
2. A set of specific variables that should appear in the model, including their names and types;
3. A set of observations for two or more of the variables as they vary over time.

In addition, the system requires three control parameters: the maximum number of processes allowed in a model, the minimum number of processes, and the number of times each generic process can occur. Given this information, the system first generates all instantiations of generic processes with specific variables that are consistent with the type constraints. After this, it finds all ways to combine these instantiated processes to form instantiated models that have acceptable numbers of processes. The resulting models refer to specific variables, but their parameters are still unknown. Next, IPM uses a nonlinear optimization routine to determine these parameter values. Finally, the system selects and returns the candidate that produces the smallest squared error on the training data, modulated by a minimum description length criterion.

The procedure for generating all acceptable model structures is straightforward, but the method for parameter optimization deserves some discussion. The aim is to find, for each model structure, parameters that minimize the model's squared predictive error on the observations. We have tried a number of standard optimization algorithms, including Newton's method and the Levenberg-Marquardt method, but we have found these techniques encounter problems with convergence and local optima. In response, we designed and implemented our own parameter-fitting method, which has given us the best results to date.

A nonlinear optimization algorithm attempts to find a set of parameters $\mathbf{\Theta}$ that minimizes an objective function $\mathbf{E}(\mathbf{\Theta})$. In our case, we define $\mathbf{E}$ as the squared error between the observed and predicted time series:

$$\mathbf{E}(\mathbf{\Theta}) = \sum_{t=1}^{T} \sum_{j=1}^{J} (\ln(x_j^o(t)) - \ln(x_j(t)))^2 \ , \qquad (1)$$

where x_j^o and x_j represent the observed and predicted values of J observed variables, t denotes time instants, and $\ln(\cdot)$ is the natural logarithmic function.

Standard least-squares estimation is widely recognized as relatively brittle with respect to outliers in samples that contain gross error. Instead, as shown in Equation (1), we minimize the sum of squared differences between logarithmically transformed variables, which is one approach to robust estimation proposed by Box and Cox (1964). In addition, we maintain positivity constraints on process variables by performing a logarithmic transformation on the differential equations in which they appear. Predicted values for x_j are obtained by solving finite-difference approximations of the differential equations specified in the model. The parameter vector $\mathbf{\Theta}$ incorporates all unknowns, including any initial conditions for unobserved variables needed to solve the differential equations.

In order to minimize our error function, $\mathbf{E}$, defined as a sum of squared errors, we can calculate its gradient vector with respect to a parameter vector. For this purpose, we borrowed the basic idea of error backpropagation through time (Rumelhart, Hinton, & Williams, 1986), frequently used for learning in recurrent neural networks. However, the task of process model induction required us to extend this method to support the many different functional forms that can occur. Our current solution relies on hand-crafted derivatives for each generic process, but it utilizes the additive nature of process models to retain the modularity of backpropagation and its compositional character. These in turn let the method carry out gradient search to find parameters for each model structure.

Given a model structure and its corresponding backpropagation equations, our parameter-fitting algorithm carries out a second-order gradient search (Saito & Nakano, 1997). By adopting a quasi-Newton framework (e.g., Luenberger, 1984), this calculates descent direction as a partial Broyden-Fletcher-Goldfarb-Shanno update and then calculates the step length as the minimal point of a second-order approximation. In earlier experiments on a variety of data sets, this algorithm worked quite efficiently as compared to standard gradient search methods. Of course, this approach does not eliminate all problems with local optima; thus, for each model structure, IPM runs the parameter-fitting algorithm ten times with random initial parameter values, then selects the best result.

Using these techniques, IPM overcomes many of the problems with local minima and slow convergence that we encountered in our early efforts, giving reasonable performance according to the squared error criterion. However, we anticipate that solving more complex problems will require the utilization of even more sophisticated algorithms for non-linear minimization.

However, reliance on squared error as the sole optimization criterion tends to select overly complex process models that overfit the training data. Instead, IPM computes the description length of each parameterized model as the sum of its complexity and the information content of the data left unexplained by the model. We define complexity as the number of free parameters and variables in a model and the unexplained content as the number of bits needed to encode the squared error of the model. Rather than selecting the model with the lowest error, IPM prefers the candidate with the shortest description length, thus balancing model complexity against fit to the training data.

3 Modeling Predator-Prey Interaction

Now we are ready to consider IPM's operation on an ecosystem modeling task. Within Earth science, models of predator-prey systems are among the simplest in terms of the number of variables and parameters involved, making them good starting points for our evaluation. We focus here on the protozoan system composed of the predator *P. aurelia* and the prey *D. nasutum*, which is well known in population ecology. Jost and Adiriti (2000) present time-series data for this system, recovered from an earlier report by Veilleux (1976), that are now available on the World Wide Web. The data set includes measurements for the two species' populations at 12-hour intervals over 35 days, as shown in Figure 1. The data are fairly smooth over the entire period, with observations at regular intervals and several clear cycles. We decided to use these observations as an initial test of IPM's ability to induce an ecosystem model.

3.1 Background Knowledge about Predator-Prey Interaction

A scientist who wants IPM to construct explanatory models of his observations must first provide a set of generic processes that encode his knowledge of the domain. Table 1 presents a set of processes that we extracted from our reading of the Jost and Adiriti article. As illustrated, each generic process specifies a set of generic variables with type constraints (in braces), a set of parameters with ranges for their values (in brackets), and a set of algebraic or differential equations that encode causal relations among the variables (where $d[X, t, 1]$ refers to the first derivative of X with respect to time). Each process can also include one or more conditions, although none appear in this example.

The table shows five such generic processes. Two structures, *predation_holling* and *predation_volterra*, describe alternative forms of feeding; both cause the predator population to increase and the prey population to decrease, but they differ in their precise functional forms. Two additional processes – *logistic_growth* and *exponential_growth* – characterize the manner in which a species' population

Table 1. A set of generic processes for predator-prey models.

generic process logistic_growth;
  variables $S\{species\}$;
  parameters ψ $[0, 10], \kappa$ $[0, 10]$;
  equations $d[S, t, 1] = \psi * S * (1 - \kappa * S)$;

generic process predation_volterra;
  variables $S1\{species\}, S2\{species\}$;
  parameters π $[0, 10], \nu$ $[0, 10]$;
  equations $d[S1, t, 1] = -1 * \pi * S1 * S2$;
    $d[S2, t, 1] = \nu * \pi * S1 * S2$;

generic process predation_holling;
  variables $S1\{species\}, S2\{species\}$;
  parameters ρ $[0, 1], \gamma$ $[0, 1], \eta$ $[0, 1]$;
  equations $d[S1, t, 1] = -1 * \gamma * S1 * S2/(1 + \rho * \gamma * S1)$;
    $d[S2, t, 1] = \eta * \gamma * S1 * S2/(1 + \rho * \gamma * S1)$;

generic process exponential_growth;
  variables $S\{species\}$;
  parameters β $[0, 10]$;
  equations $d[S, t, 1] = \beta * S$;

generic process exponential_decay;
  variables $S\{species\}$;
  parameters α $[0, 1]$;
  equations $d[S, t, 1] = -1 * \alpha * S$;

increases in an environment with unlimited resources, again differing mainly in the forms of their equations. Finally, the *exponential_decay* process refers to the decrease in a species' population due to natural death. All five processes are generic in the sense that they do not commit to specific variables. For example, the generic variable S in *exponential_decay* does not state which particular species dies when it is active. IPM must assign variables to these processes before it can utilize them to construct candidate models.

Although the generic processes in Table 1 do not completely encode knowledge about predator-prey dynamics, they are adequate for the purpose of evaluating the IPM algorithm on the Veilleux data. If needed, a domain scientist could add more generic processes or remove ones that he considers irrelevant. The user is responsible for specifying an appropriate set of generic processes for a given modeling task. If the processes recruited for a particular task do not represent all the mechanisms that are active in that environment, the induced models may fit the data poorly. Similarly, the inclusion of unnecessary processes can increase computation time and heighten the chances of overfitting the data.

Before the user can invoke IPM, he must also provide the system with the variables that the system should consider including in the model, along with their types. This information includes both observable variables, in this case *predator* and *prey*, both with type *species*, and unobservable variables, which do not arise in this modeling task. In addition, he must state the minimum acceptable number of processes (in this case one), the maximum number of processes (four), and the number of times each generic process can occur (two).

3.2 Inducing Models for Predator-Prey Interaction

Given this information, IPM uses the generic processes in Table 1 to generate all possible model structures that relate the two species *P. aurelia* and *D. nasutum*, both of which are observed. In this case, the system produced 228 candidate

Table 2. Process model induced for predator-prey interaction.

```
model Predator_Prey;
variables $Predator, Prey$;
observables $Predator, Prey$;
process exponential_decay;
  equations $d[Predator, t, 1] = -1 * 1.1843 * Predator$;
process logistic_growth;
  equations $d[Prey, t, 1] = 2.3049 * Prey * (1 - 0.0038 * Prey)$;
process predation_volterra;
  equations $d[Prey, t, 1] = -1 * 0.0298 * Prey * Predator$;
            $d[Predator, t, 1] = 0.4256 * 0.0298 * Prey * Predator$;
```

structures, for each of which it invoked the parameter-fitting routine described earlier. Table 2 shows the parameterized model that the system selected from this set, which makes general biological sense. It states that, left in isolation, the prey (*D. nasutum*) population grows logistically, while the predator (*P. aurelia*) population decreases exponentially. Predation leads to more predators and to fewer prey, controlled by multiplicative equations that add 0.4256 predators for each prey that is consumed.

Qualitatively, the model predicts that, when the predator population is high, the prey population is depleted at a faster rate. However, a reduction in the prey population lowers the rate of increase in the predator population, which should produce an oscillation in both populations. Indeed, Figure 1 shows that the model's predicted trajectories produce such an oscillation, with nearly the same period as that found in the data reported by Jost and Adiriti. The model produces a squared error of 18.62 on the training data and a minimum description length score of 286.68. The r^2 between the predicted and observed values is 0.42 for the prey and 0.41 for the predator, which indicates that the model explains a substantial amount of the observed variation.

3.3 Experimental Comparison with Autoregression

Alternative approaches to induction from time-series data, such as multivariate autoregression, do not yield the explanatory insight of process models. However, they are widely used in practice, so naturally we were interested in how the two methods compare in their predictive abilities. To this end, we ran the Matlab package ARFit (Schneider & Neumaier, 2001) on the Veilleux data to infer the structure and parameters of an autoregressive model. This uses a stepwise least-squares procedure to estimate parameters and a Bayesian criterion to select the best model. For the runs reported here, we let ARFit choose the best model order from zero to five.

To test the two methods' abilities to forecast future observations, we divided the time series into successive training and test sets while varying their relative sizes. In particular, we created 35 training sets of size $n = 35 \dots 69$ by selecting

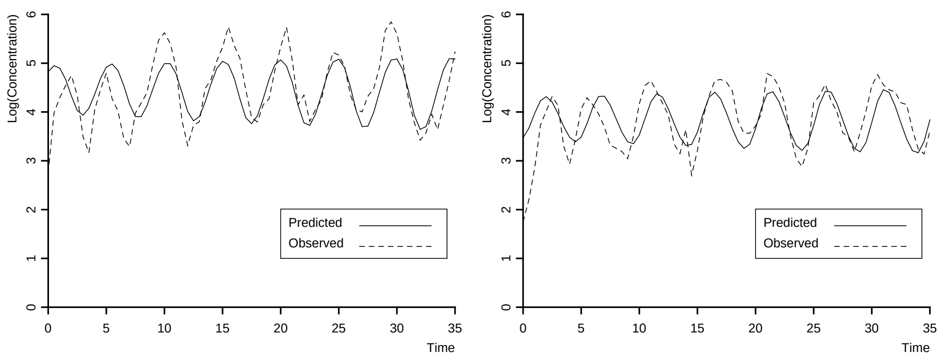

Fig. 1. Predicted and observed log concentrations of protozoan prey (left) and predator (right) over a period of 36 hours.

the first n examples of the time series, each with a corresponding test set that contained all successive observations. In addition to using these training sets to induce the IPM and autoregressive models, we also used their final values to initialize simulation with these models. Later predictions were based on predicted values from earlier in the trajectory. For example, to make predictions for $t = 40$, both the process model and an autoregressive model of order one would utilize their predictions for $t = 39$, whereas an autoregressive model of order two would draw on predictions for $t = 38$ and $t = 39$.

Figure 2 plots the resulting curves for the models induced by IPM, ARFit, and a constant approximator. In every run, ARFit selected a model of order one. Both IPM and autoregression have lower error than the straw man, except late in the curve, when few training cases are available. The figure also shows that, for 13 to 21 test instances, the predictive abilities of IPM's models are roughly equal to or better than those for the autoregressive models. Thus, IPM appears able to infer models which are as accurate as those found by an autoregressive method that is widely used, while providing interpretability that is lacking in the more traditional models.

4 Modeling an Aquatic Ecosystem

Although the predator-prey system we used in the previous section was appropriate to demonstrate the capabilities of the IPM algorithm, rarely does one find such simple modeling tasks in Earth science. Many ecosystem models involve interactions not only among the species but also between the species and environmental factors. To further test IPM's ability, we provided it with knowledge and data about the aquatic ecosystem of the Ross Sea in Antarctica (Arrigo et al., in press). The data came from the ROAVERRS program, which involved three cruises in the austral spring and early summers of 1996, 1997, and 1998. The measurements included time-series data for phytoplankton and nitrate concentrations, as shown in Figure 3.

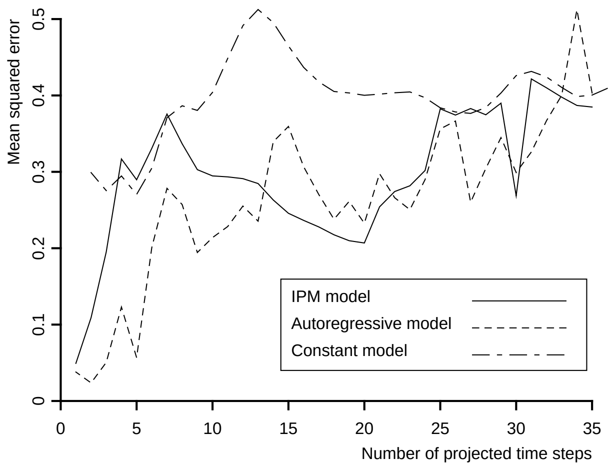

Fig. 2. Predictive error for induced process models, autoregressive models, and constant models, vs. the number of projected time steps, on the predator-prey data.

4.1 Background Knowledge about Aquatic Ecosystems

Taking into account knowledge about aquatic ecosystems, we crafted the set of generic processes shown in Table 3. In contrast to the components for predator-prey systems, the *exponential_decay* process now involves not only reduction in a species' population, but also the generation of residue as a side effect. Formation of this reside is the mechanism by which minerals and nutrients return to the ecosystem. Knowledge about the generation of residue is also reflected in the process *predation*.

The generic process *nutrient_uptake* encodes knowledge that plants derive their nutrients directly from the environment and do not depend on other species for their survival. Two other processes – *remineralization* and *constant_inflow* – convey information about how nutrients become available in ecosystems Finally, the *growth* process posits that some species can grow in number independent of predation or nutrient uptake.

As in the first domain, our approach to process model induction requires the user to specify the variables to be considered, along with their types. In this case, we knew that the Ross Sea ecosystem included two species, phytoplankton and zooplankton, with the concentration of the first being measured in our data set and the second being unobserved. We also knew that the sea contained nitrate, an observable nutrient, and detritus, an unobserved residue generated when members of a species die.

4.2 Inducing Models for an Aquatic Ecosystem

Given this background knowledge about the Ross Sea ecosystem and data from the ROAVERRS cruises, we wanted IPM to find a process model that explained the variations in these data. To make the system's search tractable, we intro-

Table 3. Five generic processes for aquatic ecosystems with constraints on their variables and parameters.

```
generic process exponential_decay;
  variables $S\{species\}, D\{detritus\}$;
  parameters $\alpha$ $[0, 10]$;
  equations $d[S, t, 1] = -1 * \alpha * S$;
            $d[D, t, 1] = \alpha * S$;

generic process constant_inflow;
  variables $N\{nutrient\}$;
  parameters $\nu$ $[0, 10]$;
  equations $d[N, t, 1] = \nu$;

generic process nutrient_uptake;
  variables $S\{species\}, N\{nutrient\}$;
  parameters $\beta$ $[0, 10]$, $\mu$ $[0, 10]$;
  conditions $N > \tau$;
  equations $d[S, t, 1] = \mu * S$;
            $d[N, t, 1] = -1 * \beta * \mu * S$;

generic process remineralization;
  variables $N\{nutrient\}, D\{detritus\}$;
  parameters $\psi$ $[0, 10]$ ;
  equations $d[N, t, 1] = \psi * D$ ;
            $d[D, t, 1] = -1 * \psi * D$;

generic process predation;
  variables $S1\{species\}, S2\{species\}, D\{detritus\}$;
  parameters $\rho$ $[0, 10]$, $\gamma$ $[0, 10]$;
  equations $d[S1, t, 1] = \gamma * \rho * S1$;
            $d[D, t, 1] = (1 - \gamma) * \rho * S1$;
            $d[S2, t, 1] = -1 * \rho * S1$;
```

duced further constraints by restricting each generic process to occur no more than twice and considering models with no fewer than three processes and no more than six. Using the four variables described above – Phyto{species}, Zoo{species}, Nitrate{nutrient}, and Detritus{residue} – IPM combined these with the available generic processes to generate some 200 model structures. Since Phyto and Nitrate were observable variables, the system considered only those models that included equations with these variables on their left-hand sides. The parameter-fitting routine and the description length criterion selected the model in Table 4, which produced a mean squared error of 23.26 and a description length of 131.88. Figure 3 displays the log values this candidate predicts for phytoplankton and nitrate, along with those observed in the field. The r^2 value is 0.51 for Phyto but only 0.27 for Nitrate, which indicates that the model explains substantially less of the variance than in our first domain.

Note that the model includes only three processes and that it makes no reference to zooplankton. The first process states that the phytoplankton population dies away at an exponential rate and, in doing so, generates detritus. The second process involves the growth of phytoplankton, which increases its population as it absorbs the nutrient nitrate. This growth happens only when the nitrate concentration is above a threshold, and it causes a decrease in the concentration of the nutrient. The final process states that the residue is converted to the consumable nitrate at a constant rate.

In fact, the model with the lowest squared error included a predation process which stated that zooplankton feeds on phytoplankton, thereby increasing the former population, decreasing the latter, and producing detritus. However,

Table 4. Induced model for the aquatic ecosystem of the Ross Sea.

```
model Aquatic_Ecosystem;
variables $Phyto, Nitrate, Detritus, Zoo$;
observables $Phyto, Nitrate$;
process exponential_decay_1;
  equations $d[Phyto, t, 1] = -1 * 1.9724 * Phyto$;
            $d[Detritus, t, 1] = 1.9724 * Phyto$;
generic process nutrient_uptake;
  conditions $Nitrate > 3.1874$;
  equations $d[Phyto, t, 1] = 3.6107 * Phyto$;
            $d[Nitrate, t, 1] = -1 * 0.3251 * 3.6107 * Phyto$;
generic process remineralization;
  equations $d[Nitrate, t, 1] = 0.032 * Detritus$;
            $d[Detritus, t, 1] = -1 * 0.032 * Detritus$;
```

IPM calculated that the improved fit was outweighed by the cost of including an additional process in the model. This decision may well have resulted from a small population of zooplankton, for which no measurements were available but which is consistent with other evidence about the Ross Sea ecosystem. We suspect that, given a more extended time series, IPM would rank this model as best even using its description length, but this is an empirical question that must await further data.

5 Discussion

There is a large literature on the subject of ecosystem modeling. For example, many Earth scientists develop their models in STELLA (Richmond et al., 1987), an environment that lets one specify quantitative models and simulate their behavior over time. However, work in this and similar frameworks has focused almost entirely on the manual construction and tuning of models, which involves much trial and error. Recently, increased computing power has led a few Earth scientists to try automating this activity. For instance, Morris (1997) reports a method for fitting a predator-prey model to time-series data, whereas Jost and Adiriti (2000) use computation to determine which functional forms best model similar data. Our approach has a common goal, but IPM can handle more complex models and uses domain knowledge about generic processes to constrain search through a larger model space.

On another front, our approach differs from most earlier work on equation discovery (e.g., Washio et al., 2000) by focusing on differential equation models of dynamical systems. The most similar research comes from Todorovski and Džeroski (1997), Bradley et al. (1999), and Koza et al. (2001), who also report methods that induce differential equation models by searching for model structures and parameters that fit time-series data. Our framework extends theirs by

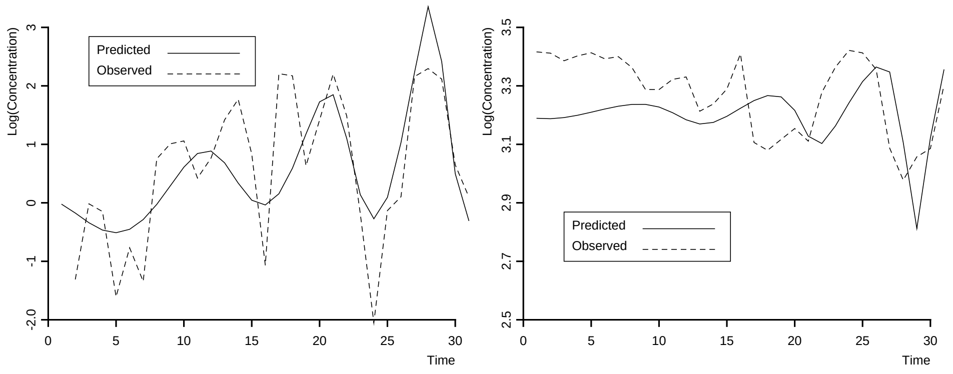

Fig. 3. Predicted and observed log concentrations of phytoplankton (left) and nitrate (right) in the Ross Sea over 31 days.

focusing on processes, which play a central role in many sciences and provide a useful framework for encoding domain knowledge that constrains search and produces more interpretable results. Also, because IPM can construct models that include theoretical terms, it supports aspects of abduction (e.g., Josephson, 2000) as well as induction.

Still, however promising our approach to ecosystem modeling, considerable work remains before it will be ready for use by practicing scientists. Some handcrafted models contain tens or hundreds of equations, and we must find ways to constrain search further if we want our system to discover such models. The natural source of constraints is additional background knowledge. Earth scientists often know the qualitative processes that should appear in a model (e.g., that one species preys on another), even when they do not know their functional forms. Moreover, they typically organize large models into modules that are relatively independent, which should further reduce search. Future versions of IPM should take advantage of this knowledge, along with more powerful methods for parameter fitting that will increase its chances of finding the best model.

In summary, we believe that inductive process modeling provides a valuable alternative to the manual construction of ecosystem models which combines domain knowledge, heuristic search, and data in a powerful way. The resulting models are cast in a formalism recognizable to Earth scientists and they refer to processes that domain experts will find familiar. Our initial results on two ecosystem modeling tasks are encouraging, but we must still extend the framework in a number of directions before it can serve as a practical scientific aid.

Acknowledgements

This work was supported by the NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation. We thank Tasha Reddy and Alessandro Tagliabue for preparing the ROAVERRS data and for discussions about ecosystem processes. We also thank Sašo Džeroski and Ljupčo Todorovski for useful discussions about approaches to inductive process modeling.

References

Arrigo, K. R., Worthen, D. L. & Robinson, D. H. (in press). A coupled ocean-ecosystem model of the Ross Sea. Part 2: Phytoplankton taxonomic variability and primary production. *Journal of Geophysical Research.*

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society*, *Series B*, *26*, 211–252.

Bradley, E., Easley, M., & Stolle, R. (2001). Reasoning about nonlinear system identification. *Artificial Intelligence*, *133*, 139–188.

Josephson, J. R. (2000). Smart inductive generalizations are abductions. In P. A. Flach & A. C. Kakas (Eds.), *Abduction and induction.* Kluwer.

Jost, C., & Adiriti, R. (2000). Identifying predator-prey processes from time-series. *Theoretical Population Biology*, *57*, 325–337.

Koza, J., Mydlowec, W., Lanza, G., Yu, J., & Keane, M. (2001). Reverse engineering and automatic synthesis of metabolic pathways from observed data using genetic programming. *Pacific Symposium on Biocomputing*, *6*, 434–445.

Langley, P., George, D., Bay, S. & Saito, K. (in press). Robust induction of process models from time-series data. *Proceedings of the Twentieth International Conference on Machine Learning.* Washington, DC: AAAI Press.

Luenberger, D.G. (1984). *Linear and nonlinear programming.* Reading, MA: Addison-Wesley.

Morris, W. F. (1997). Disentangling effects of induced plant defenses and food quantity on herbivores by fitting nonlinear models. *American Naturalist*, *150*, 299–327.

Richmond, B., Peterson, S., & Vescuso, P. (1987). *An academic user's guide to STELLA.* Lyme, NH: High Performance Systems.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing.* Cambridge: MIT Press.

Saito, K., & Nakano, R. (1997). Law discovery using neural networks. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 1078–1083). Yokohama: Morgan Kaufmann.

Schneider, T., & Neumaier, A. (2001). Algorithm 808: ARFIT – A Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, *27*, 58–65.

Todorovski, L., & Džeroski, S. (1997). Declarative bias in equation discovery. *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 376–384). San Francisco: Morgan Kaufmann.

Veilleux, B. G. (1979). An analysis of the predatory interaction between Paramecium and Didinium. *Journal of Animal Ecology*, *48*, 787–803.

Washio, T., Motoda, H., & Niwa, Y. (2000). Enhancing the plausibility of law equation discovery. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 1127–1134). Stanford, CA: Morgan Kaufmann.