Inductive Process Modeling

Pat Langley (langley@csli.stanford.edu)
Computatational Learning Laboratory, Center for the Study of Language and Information, Stanford University, Stanford, CA 94305 USA

Ljupčo Todorovski (ljupco.todorovski@ijs.si)
Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Will Bridewell (willb@csli.stanford.edu)
Computatational Learning Laboratory, Center for the Study of Language and Information, Stanford University, Stanford, CA 94305 USA

Sašo Džeroski (saso.dzeroski@ijs.si)
Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract. In this paper, we pose a novel research problem for machine learning that involves constructing a *process model* from continuous data. We claim that casting learned knowledge in terms of processes with associated equations is desirable for scientific and engineering domains, where such notations are commonly used. We also argue that existing induction methods are not well suited to this task, although some techniques hold partial solutions. In response, we describe an approach to learning process models from time-series data and illustrate its behavior in three domains. In closing, we describe open issues in process model induction and encourage other researchers to tackle this important problem.

Keywords: Scientific Discovery, Process Models, Compositional Modeling, System Identification, Ecosystem Modeling

1. Introduction and Motivation

Many scientific and engineering domains involve continuous variables that change over time. The increasing availability of data from such systems presents both an opportunity and a challenge for machine learning. Successful applications of induction methods hold obvious benefits, and there exist large literatures on computational methods for regression and time-series prediction. But however accurate the predictive models these techniques induce from data, they usually make little contact with the formalisms and concepts used by scientists and engineers. As Schwabacher and Langley (2001) have argued, domain experts will benefit more from knowledge that is cast in *communicable* form, utilizing notations already familiar to them. And as Pazzani et al. (2001) have shown, experts in some domains will reject a learning system's output, even when very accurate, unless it makes contact with their prior knowledge.

Research on discovering numeric laws (e.g., Langley, 1981, Żytkow et al., 1990, Džeroski & Todorovski, 1993, Washio et al., 2000) addresses this concern, in that many scientists find equations familiar. However, although the resulting knowledge generalizes beyond the training data, it is typically *descriptive* in that it directly relates observable variables. In contrast, a model in science and engineering often provides an *explanation* which includes variables, objects, or mechanisms that are unobserved, but that help predict the behavior of the observed variables. Moreover, explanations often make use of general concepts or relations that occur in different models. Compare Kepler's third law, which relates a planet's period to its distance from the sun, with Newton's theory of gravitation, which introduces a theoretical variable for gravitational force. Kepler's model answers a *what* question about planetary motion, whereas the latter answers a *why* question, thereby providing explanation.

We claim that scientists and engineers often state their explanations in terms of generic *processes* from some domain. A process describes one or more causal relations among system variables. We develop here a particular class of processes that are represented in terms of differential equations (for modeling change over time) and algebraic equations (for modeling instantaneous effects). Each such process may also include conditions, stated as threshold tests on its input variables, that describe when it is active. A *process model* consists of a set of processes that link observable variables with each other causally, possibly through unobserved theoretical terms.[1]

Table I shows a simple process model for a predator–prey relationship between two protists (microorganisms). In this ecosystem, *Didinium nasutum* preys upon *Paramecium aurelia*. The model includes three processes that explain the changes in the concentrations of the two species (*nasutum* and *aurelia*, respectively). The first of these states that the *aurelia* concentration increases logistically, limited by the environment's carrying capacity, while the second process states that *nasutum* decays exponentially. The third process explains the interaction between the two species using a Holling type I response. That is, the prey population decreases in relation to the size of both populations and the rate of predation, while the predator population increases correspondingly, with an additional term reflecting the efficiency of predation (intuitively, the number of predators produced by consuming one prey). Given initial conditions (e.g., $nasutum = 64.67$, $aurelia = 276.60$), this model can simulate changes in the two species'

[1] Our framework can be viewed readily as a quantitative version of Forbus' (1984) qualitative process theory, from which we have borrowed many ideas.

Table I. A quantitative process model of a protist ecosystem with one predator (*D. nasutum*) and one prey (*P. aurelia*). Prey growth is logistic, predator decay is exponential, and predation is explained using a Holling type I response. The notation $d[X, t, 1]$ indicates the first derivative of X with respect to t.

model PredatorPrey;
variables $aurelia\{prey\}, nasutum\{predator\}$;
observable $aurelia, nasutum$;
process aurelia_growth;
equations $d[aurelia, t, 1] = 1.81 * aurelia * (1 - 0.0003 * aurelia)$;
process nasutum_decay;
equations: $d[nasutum, t, 1] = -1 * 1.04 * nasutum$;
process predation_holling_1;
equations $d[aurelia, t, 1] = -1 * 0.03 * aurelia * nasutum$;
$d[nasutum, t, 1] = 0.30 * 0.03 * aurelia * nasutum$;

concentrations over time. Note that, as expected, the model predicts oscillitory behavior, but it does not state this explicitly, as would descriptive laws. For comparison, the model is equivalent to the two differential equations

$$\begin{aligned} \dot{a} &= 1.81 \cdot a \cdot (1 - 0.0003 \cdot a) - 0.03 \cdot a \cdot n \\ \dot{n} &= -1.04 \cdot n + 0.30 \cdot 0.03 \cdot a \cdot n \end{aligned}$$

where $n = nasutum$ and $a = aurelia$, which predict the same trajectories over time.

We maintain that process models of the sort in Table 1 occur frequently in science and engineering, and that inducing them from data is a worthwhile task for machine learning researchers to address. We can state the task of *inductive process modeling* as:

- *Given:* Observations for a set of continuous variables as they change over time;
- *Given:* Generic processes that specify causal relations among variables using conditional equations;
- *Given:* Constraints on the types of variables involved in each process and knowledge about these variables' types;
- *Find:* A specific process model that fits the observed data and that predicts future data accurately.

Note that this formulation distinguishes between the generic processes given as input and the specific processes in the induced model, which mention particular variables and parameter values. Later, we will see that this background knowledge provides important constraints on search

through the model space and ensures that the resulting models make contact with concepts familiar to domain scientists.

This paper takes the form of an exploratory research report in the sense described by Dietterich (1990). Following his advice, we state clearly a promising new problem for machine learning and explore its various facets. In the section that follows, we consider some challenges posed by the problem of inducing quantitative process models. Next we review a variety of established induction paradigms, concluding that none can be applied directly to this task, though some hold promising ideas on which we can build. After this, we describe a candidate approach to process model induction and illustrate it with some initial results. Finally, we close by suggesting an agenda for future research on this important topic. We do not report extensive experimental studies, but we do present results on a number of domains as evidence for the generality of both the task and our initial approach to it.

2. Challenges of Inductive Process Modeling

We have claimed that the induction of process models differs from the tasks typically studied in machine learning. Thus, before proceeding further, we should review the characteristics that distinguish this task from traditional induction problems and that pose research challenges. Some characteristics address aspects of the training data, whereas others involve constraints on the nature of acquired knowledge.

Process models are designed to characterize the behavior of dynamic systems, which may be in equilibrium, over time. The data produced by such systems differ in a variety of ways from those that arise in most induction tasks. First, the variables are primarily continuous, since they represent quantitative measurements of the system under study. Therefore we must solve a regression task rather than one that involves learning classifiers. Second, the observed values are not independently and identically distributed, since those observed at later time steps depend on those measured earlier. Thus, the data violate an assumption made by the great majority of available learning algorithms. Finally, the training data are primarily unsupervised, in that they describe a set of variables that change over time, with no variable being singled out for special attention. Such unsupervised learning is generally viewed as more difficult than learning from supervised data.

We have already noted that process models are explanatory in nature. One can observe some variables as they change over time, but the processes that cause these changes are not themselves observable, and multiple processes can interact to produce complex behavior. Moreover, process models can include theoretical variables that are also unob-

servable. These characteristics concern the knowledge acquired during learning, but their presence makes the induction task more complex, just as the introduction of hidden variables into neural or Bayesian networks requires additional learning mechanisms. As a result, inducing process models is a suitable option only for domains that are complex enough to require such explanatory accounts. Fortunately, this challenge is offset somewhat by knowledge about generic processes that can serve as components of candidate models.

Another assumption also makes process model induction more tractable than it might be otherwise: the dynamical systems that the models explain are generally viewed as deterministic. The observations themselves may well contain noise, which can complicate matters for this paradigm as it does for others. However, the framework posits that processes themselves are always active whenever their conditions are met and that their equations have the specified effects. Scientists and engineers often treat the systems they study as deterministic, and we will operate under the same assumption.

3. Limitations of Existing Approaches

According to Dietterich (1990), an exploratory research paper should not only define a novel problem, but also show the inability of existing methods to solve that problem. Thus, we should consider whether any established learning techniques can handle inductive process modeling. Our discussion will draw on comments in the previous section about the distinctive characteristics of this task.

We have argued that methods for equation discovery, although they generate knowledge in formalisms familiar to scientists, are not sufficient for our task because they produce descriptive summaries of data rather than explanations in terms of underlying processes. A few exceptions to this trend exist, such as Bradley et al.'s (2001) work on constructing differential equation models from existing components, Todorovski and Džeroski's (1997) use of context-free grammars for generating candidate equations, and Koza et al.'s (2001) method for inferring quantitative metabolic pathways. However, even these efforts do not combine known generic processes into explanatory models, and most work on equation discovery is far less relevant.

Mainstream methods for supervised learning fall short on the same front, in that they can develop accurate predictors but fail to make contact with explanatory concepts familiar to domain experts. Thus, widely used algorithms for inducing regression trees (e.g., Breiman et al., 1984) and multilayer neural networks do not, by themselves, seem sufficient for inductive process modeling, nor do other schemes

for predicting continuous variables. This grouping includes techniques for time-series analysis, such as ARIMA models (e.g., Schneider & Neumaier, 2001), which are designed to predict accurately one time step ahead, rather than matching entire trajectories. However, such methods can prove useful in the overall task, as we will see later with techniques for equation discovery.

Because we have emphasized the explanatory nature of process models, we should consider whether explanation-based learning (e.g., Mitchell et al., 1986) lends itself to constructing them. This paradigm also assumes the use of background knowledge to account for observations, but nearly all research within it has dealt with classification or problem-solving tasks, rather than with predicting continuous variables. Moreover, the typical formulation assumes that the training data are supervised, whereas inductive process modeling deals primarily with observational data in which no variable is treated as special. Most important, explanation-based methods are generally deductive in nature, and we presumably need an inductive approach that does not rely on fixed axioms. DeJong's (1995) work on explanation-based learning for motor control comes closer to our needs, but the paradigm as a whole seems poorly suited.

Research on explanation-based learning gave birth to another movement, often referred to as *theory revision*, that seems a better match to our problem in that it combines background knowledge with inductive learning. This framework often states domain theories as Horn clause programs, and the revision methods typically employ operators for adding and removing rules' conditions, as well as adding and removing entire rules (e.g., Ourston & Mooney, 1990). Moreover, these theories usually incorporate nonterminal symbols that one can view as theoretical terms. However, the paradigm assumes that one has an explanatory model at the outset, rather than constructing it from available components. More importantly, theory-revision methods emphasize the learning of classifiers from supervised training data rather than dealing with unsupervised regression. Thus, this approach also seems like a poor match, although we will return to it when we discuss open issues.

Inductive logic programming (e.g., Lavrač & Džeroski, 1994) fares somewhat better on the task of learning process models. This framework takes advantage of background knowledge, stated as Horn clauses and ground literals, to learn from training cases. The resulting knowledge is itself cast as a set of Horn clauses, possibly with nonterminal (i.e., theoretical) symbols, and thus can have an explanatory character. As usually practiced, inductive logic programming uses supervised learning for classification tasks and often generates descriptive rules

with no intermediate concepts. Moreover, the research emphasis has been on generating logical structures rather than numerical ones. However, Garrett et al. (in press) have used this approach to infer metabolic pathways that involve biochemical processes and unobserved entities, making them similar to our own models. Thus, inductive logic programming holds promise as an approach to process model construction, although both the performance and learning methods must be extended to support numeric equations and continuous time.

Since hidden Markov models (e.g., Poritz, 1988) can describe systems that change over time, we should also evaluate their relevance to our learning problem. The states in such models are unobservable, which gives them an explanatory flavor, but typically only one state can be active at a time, whereas any number of the processes in our framework can be active simultaneously. Furthermore, a hidden Markov model requires explicit links that specify which states can follow each other, rather than letting this behavior emerge from a set of processes. Finally, the probabilistic assumptions of Markov models are unnecessary for scientific and engineering domains that are typically viewed as deterministic in nature.

A final approach involves learning 'dynamic Bayesian networks', which characterize how the values of variables at one time step influence their values at the next step (e.g., Ghahramani, 1998). Such models encode the probabilistic analog for sets of differential equations, but they do not organize these equations into processes. Also, work in this paradigm has employed discrete variables and, as with hidden Markov models, the probabilistic representation seems inappropriate for deterministic process models. The situation for dynamic Bayes nets and hidden Markov models resembles that with logic programs, in that one might adapt them to support induction of process models, but they bring unnecessary assumptions and machinery to the task.

In summary, no existing learning paradigms seem appropriate for the problem of inducing process models, which indicates the need for new methods. However, many of the responses to other induction tasks will prove relevant to challenges that arise when constructing such explanatory models, as we will later see.

4. Inducing Process Models

Our primary aim here is to characterize and promote the problem of learning process models. To this end we have built a baseline system that provides evidence for the feasibility of the task and that can serve as a nontrivial comparator for future work. As with any induction system, ours can be described in terms of the formalism for communicating

the resulting models,the performance element that uses them, and the learning method that constructs the models from data.

4.1. Representing Processes and Models

We have already seen one example that involves a three-process model for a protist ecosystem. To reiterate, a process model specifies a set of processes that characterize quantitative relations among a set of observed, and possibly unobserved, variables. Each process includes zero or more conditions under which it is active, along with at least one causal equation that characterizes the influence that one or more variables exert on another. Although the processes in a given model are unordered and operate in parallel, we can organize them into a causal graph that equates the outputs of some processes with the inputs of others (Iwasaki & Simon, 1994).

Table II presents a set of processes for population dynamics, which concerns changes in species' population levels over time (Murray, 2004). The table contains *generic* processes that serve as background knowledge for learning; unlike *specific* processes, these do not commit to particular variables or parameter values, but they can indicate constraints on them. For example, the process for exponential growth states that its variable S must have type *species* and that its equation's coefficient gr must fall between zero and two. The background knowledge also contains a hierarchy of variable types that stem from the base type *number*. Note that processes in this domain include no conditions, so they are continuously active.

In addition to the generic processes, an induction system needs a set of typed variables and training data for those declared observable. For example, to construct the model in Table I, the program would require *nasutum* to have type predator and *aurelia* to have type prey. Combined with the set of generic processes, this information defines the space of model structures that the induction system will search. Since both variables are observable, two trajectories must be supplied. These data provide a means both to direct the search for parameters and to evaluate the instantiated process model.

4.2. Making Predictions with Process Models

Any induction system requires some performance element that can utilize the learned knowledge. In this case, we require an interpreter that can generate a predicted trajectory for each observable variable by carrying out forward simulation of a quantitative process model. To this end, we have implemented a module that invokes an established method (Cohen & Hindmarsh, 1996) for solving first-order differential

Table II. Five generic processes for population dynamics with constraints on their variables and parameters. The variable type constraints are denoted in braces following the local name, while parameter bounds are specified within brackets. The notation $d[S, t, 1]$ indicates the first derivative of S with respect to t.

library predpreysmall;

generic process logistic_growth;
  variables $S\{species\}$;
  parameters $gr[0, 3], ic[0, 1]$;
  equations $d[S, t, 1] = gr * S * (1 - ic * S)$;

generic process exponential_growth;
  variables $S\{species\}$;
  parameters $gr[0, 2]$;
  equations $d[S, t, 1] = gr * S$;

generic process exponential_decay;
  variables $S\{species\}$;
  parameters $dr[0, 2]$;
  equations $d[S, t, 1] = -1 * dr * S$;

generic process holling_1;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0, 1], ef[0, 1]$;
  equations $d[S1, t, 1] = -1 * ar * S1 * S2$;
  $d[S2, t, 1] = ef * ar * S1 * S2$;

generic process holling_2;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0, 1], ef[0, 1], ht[0, 1]$;
  equations $d[S1, t, 1] = -1 * ar * S1 * S2/(1 + ht * ar * S1)$;
  $d[S2, t, 1] = ef * ar * S1 * S2/(1 + ht * ar * S1)$;

type species subtypeof number;
type predator subtypeof species;
type prey subtypeof species;

equations along with simple arithmetic operations for handling algebraic equations. For this purpose, we must specify initial values for each observable variable, all values for exogenous variables, and the size of the time step, which determines the temporal resolution of the simulation.

Such an approach suffices for predicting the effects of individual differential equations, but a process model may involve chains of such equations. Thus, for each process P, the performance element solves the associated instantaneous and differential equations for the current time step to determine new values for P's output variables, uses these values to solve the equations associated with any processes that occur in the

Table III. Possible mappings of the generic processes from the population dynamics domain when given the variables *nasutum* of type *predator* and *aurelia* of type *prey*.

logistic_growth:	$S \rightarrow aurelia$	exponential_decay:	$S \rightarrow nasutum$
logistic_growth:	$S \rightarrow nasutum$	holling_1:	$S1 \rightarrow aurelia$
exponential_growth:	$S \rightarrow aurelia$		$S2 \rightarrow nasutum$
exponential_growth:	$S \rightarrow nasutum$	holling_2:	$S1 \rightarrow aurelia$
exponential_decay:	$S \rightarrow aurelia$		$S2 \rightarrow nasutum$

next step on the causal chain, and so on, until reaching the chain's final variables. The interpreter utilizes only active processes (i.e., those whose conditions are met) on each time step. When multiple active processes influence the same variable, the system makes the simplifying assumption that their effects are additive. We will see examples of the trajectories produced by the module in the next section.

4.3. Constructing a Model from Components

Once provided with background knowledge, which we assume comes from a domain expert, and time-series data about the quantitative variables one wants to explain, an induction system can carry out constrained search through the space of process models. We have implemented such a system, called IPM, wherein the search mechanism operates in three distinct stages.[2]

The first step involves finding all ways to instantiate the known generic processes with specific variables. For each generic process, IPM checks every possible assignment of observable variables to generic variables mentioned in the process, retaining only assignments that satisfy the type constraints. Given the variables *aurelia* and *nasutum*, along with the generic processes from Table II, the program finds the eight mappings indicated in Table III. From this point, IPM must determine which processes to use and what values their associated parameters should take.

In the second stage, the program uses subsets of the collection of partially instantiated processes to form *generic models*, each of which specifies an explanatory structure. To be retained as a candidate model, a set of processes must satisfy certain user-specified and system-defined constraints. In particular, the user indicates the minimal and maximal

[2] IPM serves as one component of Prometheus, an integrated environment for process modeling that we have described elsewhere (Bridewell et al., 2004).

Table IV. A generic model from the predator–prey domain wherein variables are mapped but parameters remain constrained by their bounds.

process logistic_growth;
  parameters $gr[0, 3], ic[0, 1]$;
  equations $d[aurelia, t, 1] = gr * aurelia * (1 - ic * aurelia)$;
process exponential_decay;
  parameters $dr[0, 2]$;
  equations $d[nasutum, t, 1] = -1 * dr * nasutum$;
process holling_1;
  parameters $ar[0, 1], ef[0, 1]$;
  equations $d[aurelia, t, 1] = -1 * ar * aurelia * nasutum$;
$d[nasutum, t, 1] = ef * ar * aurelia * nasutum$;

size of the model in terms of processes and states the number of times a generic process may be instantiated within that model.[3] IPM also enforces a number of constraints that define a valid model. First, exogenous variables must remain unexplained by the model, meaning that they cannot be influenced by any process. Second, all observable variables must be explained by the model in the sense that at least one process affects their values. Third, theoretical variables connected to the model must serve as both input and output to one or more processes. This last constraint ensures that the values of these terms are defined before use and that such terms affect the model's behavior. Table IV shows one valid generic model built from the mapped processes of Table III.

The third step selects parameter values for each generic model using two optimization strategies. At the core of this stage, IPM employs a nonlinear least-squares algorithm (Bunch et al., 1993) that carries out a second-order gradient descent search through the parameter space. For the first strategy, the program guides this optimization routine by simulating the full set of trajectories from initial values and providing information about the residuals for each observable variable. To avoid entrapment in local minima, IPM performs a series of restarts from random points in the parameter space.

When all the variables are observable, the second strategy augments the above search with a second strategy that uses *teacher forcing* (Williams & Zipser, 1989). Rather than carrying out a full simulation from initial values, this technique estimates parameters solely from

[3] Multiple variable mappings may apply to a generic process, and each of these becomes a candidate component. Note that a each such component can occur only once in a given model structure.

their ability to predict the values at time $i + 1$ from those at time i. Given n samples, the program performs $n - 1$ one-step simulations, forwarding the information about the residuals to the same nonlinear least-squares algorithm mentioned above. Since local minima can still cause problems, IPM restarts the teacher forcing approach at multiple randomly selected points. The system then uses the set of parameters that produce the lowest error to seed one invocation of the first strategy.

From among the results for each optimization run, IPM selects the best set of parameters according to a fitness measure. The system currently uses the sum of squared error for this step (both standard and normalized versions), but measures such as the correlation coefficient would also be reasonable. The parameters that yield the lowest error become part of the fully instantiated version of each generic model, as shown in Table I.

Notably, IPM also lets the user treat the initial values of the simulation as parameters. Often scientists and engineers can give only plausible ranges or approximate values for theoretical variables. In these cases, allowing variability in the initial values can help the program achieve a better fit to the observed trajectories. Moreover, noise can exist in the measurements of observable variables, so the system can also treat these values as parameters. Upon completion of its search, IPM returns a set of tuples, sorted by error scores, each containing a model and its associated initial values.

5. Applications and Results

In previous sections, we characterized the task of inductive process modeling and described one approach to it. In this section, we report results for our current implementation on three scientific domains. In the first we apply an extension of the generic processes given in Table II to predator–prey data. The second domain centers on the population dynamics for the aquatic ecosystem of the Ross Sea. The final modeling task involves the dynamics of Ringkøbing Fjord as its water level responds to various environmental conditions. In each case, we discuss both the inputs to IPM and the models it produces in each case. We judge success based on predictive accuracy, model comprehensibility, and visual fit to the observations.

5.1. Predator–Prey Relations

A key aspect of many ecosystem models is the relationship between predators and their prey. For instance, researchers believe that such interactions drive the evolution of the species involved, leading to the development of defense mechanisms in the prey and corresponding

Table V. Additional generic processes for the predator–prey domain. Variable type constraints are denoted in braces following the local name, while parameter bounds are specified within brackets. The notation $d[S, t, 1]$ indicates the first derivative of S with respect to time.

---

generic process ratio_dependent_2;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0,1], ef[0,1], ht[0,1]$;
  equations $d[S1,t,1] = -1 * ar * S1 * S2/(S2 + ht * ar * S1)$;
  $d[S2,t,1] = ef * ar * S1 * S2/(S2 + ht * ar * S1)$;

generic process ivlev;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0,1], ef[0,1], sat[0,1]$;
  equations $d[S1,t,1] = -1 * S2 * sat * (1 - e^{(-ar*S1)})$;
  $d[S2,t,1] = ef * S2 * sat * (1 - e^{(-ar*S1)})$;

generic process hassell_varley_1;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0,1], ef[0,1], mu[0,1]$;
  equations $d[S1,t,1] = -1 * ar * S1 * S2^{-mu} * S2$;
  $d[S2,t,1] = ef * ar * S1 * S2^{-mu} * S2$;

generic process deangelis_beddington;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0,1], ef[0,1], ht[0,1], mi[0,1]$;
  equations $d[S1,t,1] = -1 * ar * S1 * S2/(1 + ht * ar * S1 + mi * S2)$;
  $d[S2,t,1] = ef * ar * S1 * S2/(1 + ht * ar * S1 + mi * S2)$;

generic process crowley_martin;
  variables $S1\{prey\}, S2\{predator\}$;
  parameters $ar[0,1], ef[0,1], ht[0,1], mi[0,1]$;
  equations $d[S1,t,1] = -1 * ar * S1 * S2/((1 + ht * ar * S1) * (1 + mi * S2))$;
  $d[S2,t,1] = ef * ar * S1 * S2/((1 + ht * ar * S1) * (1 + mi * S2))$;

---

adaptations in the predators. In addition, understanding predation translates into knowledge about sustainability, which can determine the effects of species introduction or removal upon a particular ecosystem. Research in this area effectively began in the last century through the work of Lotka and Volterra, as Berryman (1992) has recounted.

The Lotka–Volterra model of predation assumes three fundamental processes: prey growth, predator decay, and predation. The original formulation can be stated as

$$\begin{aligned} \dot{F} &= \gamma F - \alpha FC \\ \dot{C} &= \epsilon\alpha FC - \delta C, \end{aligned}$$

where F represents the prey population and C the predator population. Here the growth rate, γ, controls the exponential increase of prey over time, whereas the decay rate, δ, similarly specifies the natural decrease of the predator population. Predation correspondingly decreases the

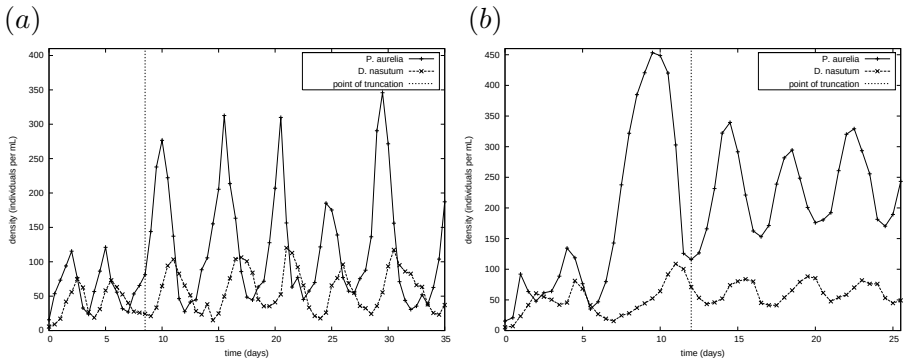

Figure 1. Trajectories for two predator–prey data sets. Samples to the left of the vertical lines were not used because they appeared to involve different regimes.

number of prey while increasing the number of predators based upon the density of both populations and a fixed attack rate, α. An efficiency factor, ϵ, corresponds to the ratio of predators produced with respect to the number of prey consumed, so smaller values slow predator growth. Since the introduction of this model, ecologists have suggested a variety of alternative forms with different growth and predation processes.

For our experiments, we combined the partial library in Table II with the generic processes in Table V and others to define two types of growth, eleven types of predation, and one type of decay. Both exponential growth and decay come from the Lotka–Volterra model, whereas logistic growth restates the Malthus–Verhulst equation. We took the predation processes from Jost and Ellner (2000), who place them into two general categories. The Holling and Ivlev processes treat predation solely as a function of prey density, while the rest also incorporate the predator density. No single form of these processes characterizes all the predatory interactions hypothesized by ecologists, so the explanation of an observed system requires a search through the space of model structures in addition to parameter fitting.

We provided IPM with the expanded set of generic processes along with data for the protist ecosystem introduced in Section 1. These data, originally collected by Vellieux (1979), consist of twice daily recordings of the densities of both *Didinium nasutum* and *Paramecium aurelia*.[4] We selected two data sets that consist of 71 and 52 recordings each. Visual inspection of the data indicated irregularities during the first few days of both experiments, which suggested that different regimes were operating at those times. As a result, we truncated the data sets to 54

[4] Jost and Ellner (2000) extracted the data from the original paper and made them publicly available.

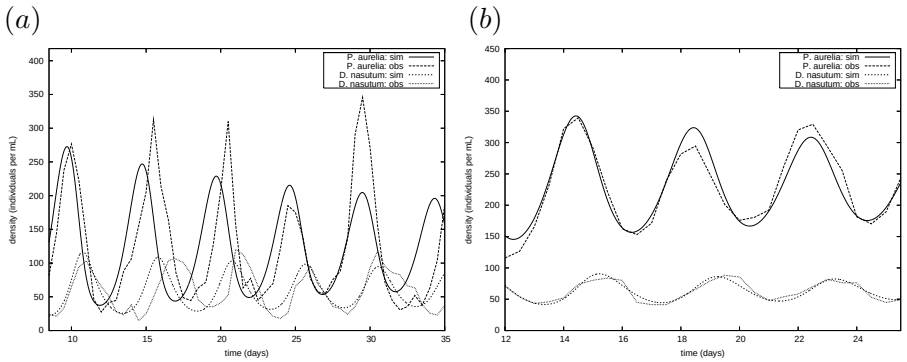

Figure 2. Simulated and observed trajectories for two predator–prey data sets. IPM induced the corresponding models from the truncated trajectories shown in Figure 1.

and 28 samples, respectively, concentrating our attention on the more regular, though still noisy, sections of the trajectories. Figure 1 shows the full trajectories, along with lines that indicate points of truncation.

To estimate the predictive accuracy of the models produced by IPM, we modified k-fold cross validation to make it applicable to time-series data. Given a data set consisting of temporal trajectories, we remove the initial time point t_0 and generate the k subsets by filling them with the remaining data selected uniformly at random and without replacement, as in standard cross validation. We then create the k folds, prepending t_0 to each training set. After IPM produces a model, it simulates the trajectories beginning at t_0 and effectively overlays each trajectory onto the appropriate test set. For each variable, the program reports both the sum of the squared error and the coefficient of determination (r^2) using the predicted and observed values.

We carried out ten-fold cross validation with IPM on both of the Vellieaux data sets independently. In each case, we told the program to search for models with three or fewer processes. The size limitation stems from domain knowledge about the nature of predation; that is, an explanatory model would typically include one process for each of prey growth, predator loss, and predatory interaction. These restrictions result in a model space that includes 470 different structures, all of which respect the structural constraints imposed by IPM.

Running the program on the data in Figure 1(a) resulted in a cross-validated r^2 of 0.59 for both variables, while giving it the data from Figure 1(b) yielded an r^2 of 0.90 for aurelia and 0.84 for nasutum. Training IPM over the complete sets of data yielded the trajectories shown in Figure 2, and led to slightly higher coefficients of determination. Notice that the relatively steep peaks in these data sets amplify the effects of even a slight phase shift, which results in lowered scores.

However, the simulated trajectories match the shape and periodicity of the observed values quite well.

Both models responsible for the trajectories in Figure 2 have one process for each of growth, loss, and predation, but they differ slightly in their functional forms. For instance, although the growth rate is roughly the same in both cases, the model for Figure 2(a) expresses growth as a logistic function that contrasts with the exponential growth present in the second model. The predation processes also differ, but the functional forms are virtually identical, with the model for Figure 2(b) introducing a very slight influence by the predator density. In general, the models were comprehensible in form, accurate in terms of r^2, and close in visual fit to the data.

5.2. The Ross Sea Ecosystem

Although studying isolated pairs of species can provide useful information about population dynamics, most ecosystems comprise complex interrelationships among multiple species and energy sources. In these environments, processes such as growth and loss exhibit more complicated forms. In particular, remineralization, which replenishes crucial nutrients, results from organism decay, while the availability of various energy sources may limit species growth. To aid our task, we worked directly with an ecologist to determine the required knowledge for developing a mechanistic model of an aquatic ecosystem.

Table VI presents a portion of the knowledge derived from our collaboration represented as a set of generic processes that relate nutrients, light, animals, and plants. Here the exponential loss process transforms dead organisms into a residue that is remineralized, thereby replenishing key nutrients. Plants (*p_species*) absorb the available nutrients that, along with the amount of light, limit their growth. Additionally, animals (*z_species*) graze on the plants at a rate determined by one of three processes (e.g., ivlev_rate). Light is exogenous and system-wide parameters are treated as variables with type *r_rate*, *n_const*, and *r_const*.

We used an extended set of the processes in Table VI to develop a model of phytoplankton growth in the Ross Sea—an environment that holds particular interest for ecologists (e.g., Arrigo et al., 2003) both for its relative simplicity and for the size of its phytoplankton blooms. We had data consisting of 188 daily measurements of nitrate, ice, and phytoplankton concentrations that cover two separate blooming seasons. Because ice in the ocean reduces the light available to the phytoplankton, we combined the measurements of ice concentration

Table VI. A partial set of generic processes for an aquatic ecosystem. Variable types that are not explicitly defined are direct subtypes of *number*.

generic process remineralization;
  variables $N\{nutrient\}, REM\{r\_rate\}, RES\{residue\}, NtoC\{n\_const\}$;
  equations $d[N,t,1] = REM * NtoC * RES$;
  $d[RES,t,1] = -1 * REM * RES$;

generic process nutrient_absorbtion;
  variables $N\{nutrient\}, GR\{phyto\_growth\}, P\{phyto\}, NtoC\{n\_const\}$;
  equations $d[N,t,1] = -1 * NtoC * GR * P$;

generic process growth_limitation;
  variables $GR\{phyto\_growth\}, RMAX\{r\_const\}, NR\{n\_rate\}, LR\{l\_rate\}$;
  equations $GR = RMAX * NR * LR$;

generic process nutrient_availability;
  variables $NR\{n\_rate\}, N\{nutrient\}$;
  parameters $r[0,10]$;
  equations $NR = N/(N+r)$;

generic process light_availability;
  variables $LR\{l\_rate\}, L\{light\}$;
  parameters $r[0,100]$;
  equations $LR = L/(L+r)$;

generic process grazing;
  variables $P\{phyto\}, Z\{zoo\}, R\{residue\}, G\{zoo\_growth\}$;
  parameters $r[0,1]$;
  equations $d[P,t,1] = -1.0 * G * Z$;
  $d[R,t,1] = r * G * Z$;
  $d[Z,t,1] = (1-r) * G * Z$;

generic process ivlev_rate;
  variables $G\{zoo\_growth\}, P\{phyto\}$;
  parameters $r[0,10], k[0,10]$;
  equations $G = k * (1 - e^{(-1*r*P)})$;

type phyto subtypeof species;
type zoo subtypeof species;
type phyto_growth subtypeof growth_rate;
type zoo_growth subtypeof growth_rate;

with the light signal to create a single exogenous variable that captures the total light entering the system.

In addition to the measured variables, we introduced several theoretical variables from ecology that could help explain the observed behaviors. For example, we told IPM to consider models that include zooplankton concentration even though we did not know whether such grazers played a meaningful role. We also included various terms, such as phytoplankton's growth rate, that correspond to concepts in current ecological theory.

We ran IPM on the two Ross Sea data sets, using the modified version of ten-fold cross validation discussed earlier, to evaluate its generalization accuracy. Since models with more than nine processes would have to include multiple characterizations of the grazing rate (i.e., more than one of ivlev_rate, monod_rate, and modified_monod_rate), we used this as an upper bound on their size. The resulting structure space consisted of 2035 different structures.

The r^2 values for the first year of data were 0.87 for the phytoplankton concentration and 0.72 for nitrate, and those for the second year were similar, with an r^2 of 0.78 for phytoplankton and 0.72 for nitrate. As with the predation data, we also ran IPM on the complete data sets to generate models. Figure 3 shows their corresponding trajectories along with the original observations. As expected, the scores were somewhat higher for these models, with r^2 above 0.92 for all variables.

The models produced by IPM for the two years differed noticeably in their form. The model for the first year suggests that phytoplankton growth is limited primarily by light and that zooplankton play the primary role in the culling of the population. The model for the second year suggests that nitrate plays the growth-limiting role and that zooplankton contribute to phytoplankton loss. In this second model, the zooplankton population grows much more slowly than in the first, which corresponds to our domain expert's observation that the size of their presence in the Ross Sea is anomalously low (Tagliabue & Arrigo, 2003). However, the abundance of nitrate in the environment favors the light-limited growth from the model for the first year. In fact, the phytoplankton require other nutrients, such as phosphorous and iron, that are also plausible limiting factors. Overall, the models had excellent quantitative and visual fits, and their comprehensibility allowed the development of meaningful hypotheses about the ecosystem.

5.3. The Ringkøbing Fjord

For our third modeling task, we focused on water-level variation in Ringkøbing Fjord, a shallow estuary on the Danish west coast. Here the water level depends on three distinct aspects of the environment: the fresh water supply, the water exchange with the North Sea, and the local wind currents. The first two factors dominate the variation in the level, with the second being controlled through a 14 part gate. However, westerly wind currents also cause a rise in the level measured at the gate. Accurate modeling of the influences on estuary height would be useful in a control system for the fjord's gate.

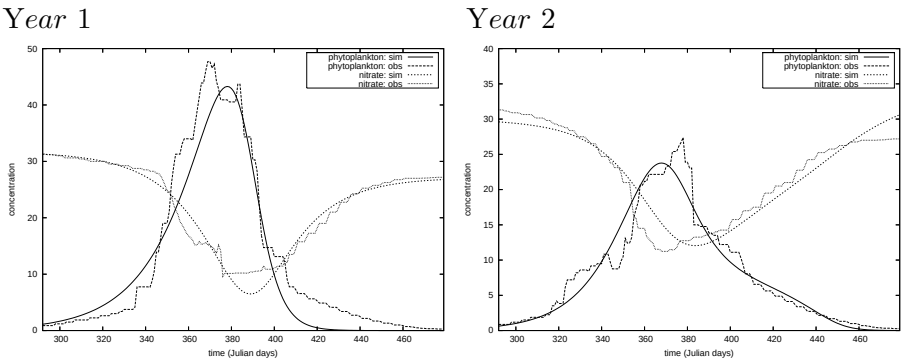

Figure 3. Simulated and observed trajectories of phytoplankton (μg/L) and nitrate (μmol/L) concentrations for two successive years in the Ross Sea.

This domain differs from the two ecological cases in that we have a partial explanation of variation in fjord height:

$$\dot{h} \;=\; \frac{f(a)}{A}(h_{sea} - h + h_0) + \frac{Q_f}{A} + g(W_{vel}, W_{dir}) \quad .$$

Here, the water level in the estuary, h, changes at a rate determined by some function of the number of open gate parts, $f(a)$, divided by the surface area of the fjord, A. When the gates are opened, the difference between the water level in the open sea, h_{sea}, h, and a constant measurement error, h_0, influence h. Additionally, fresh water accumulates in the estuary, which causes an increase of Q_f/A, and the velocity and direction of the wind, W_{vel} and W_{dir}, alter the water level according to some unknown function $g(\cdot)$. All of the variables except h and h_0 are measured and considered exogenous to the model. Thus IPM's induction task consists of identifying which functions for $f(\cdot)$ and $g(\cdot)$ most accurately model the change in the observed water level.

Table VII lists 5 of the 16 candidate processes provided to IPM as background knowledge, wherein all variable types are direct subtypes of *number*. The gate-forcing functions fill the role of $f(\cdot)$, specifying the magnitude of the effect produced by opening one or more sections of the gate. The wind-forcing functions supply forms for $g(\cdot)$ and vary based on their use of the direction and velocity of the wind. For instance, some forms used the cosine and sine of the direction to capture the influences of the wind's other components.

We ran IPM on a data set that consisted of 900 hourly samples from the Ringkøbing Fjord, again using ten-fold cross validation. Each model could contain at most three processes to account for the influence of gate openings and wind gusts on the fjord's water level. This size limitation resulted in a search space containing 696 different structures.

Table VII. A set of generic processes for modeling the Ringkøbing Fjord. Variable types not explicitly defined are direct subtypes of *number*.

generic process gate_influence_0;
  variables $GI\{gate\_influence\}$;
  parameters $r[-100000, 100000]$;
  equations $GI = r$;

generic process gate_influence_1;
  variables $GI\{gate\_influence\}, GO\{gate\_open\}$;
  parameters $r[-100000, 100000]$;
  equations $GI = r * GO$;

generic process wind_forcing_0;
  variables $WI\{wind\_influence\}$;
  parameters $r[-100000, 100000]$;
  equations $WI = r$;

generic process wind_forcing_1d_sin;
  variables $WI\{wind\_influence\}, WD\{wind\_direction\}$;
  parameters $r[-100000, 100000]$;
  equations $WI = r * sin(WD * 3.14159/180)$;

generic process wind_forcing_2dv_sin;
  variables $WI\{wind\_influence\}, WD\{wind\_direction\}, WV\{wind\_velocity\}$;
  parameters $r[-100000, 100000]$;
  equations $WI = r * WV * cos(WD * 3.14159/180)$;

As an additional constraint, we hand-coded information about mutually exclusive processes that further reduced the search space to 529 structures.

The models generated by IPM fared worse on these data than on the data from the previous domains. The resulting r^2 value for the water level was 0.39, which compares well to the value of 0.43 reported by Todorovski (2003), who both used more samples for his experiments and explored models of greater arithmetic complexity. Upon closer examination of the results we found that the model produced by one of the folds displayed erratic behavior, as if the system were thrown into a different regime as shown in Figure 5. Removing this fold from consideration results in an r^2 score of 0.66, which is considerably better than the original outcome.

Training IPM over all 900 samples yields a model with an r^2 of 0.73. Figure 4 shows the resulting trajectory along with the observed data. The model itself consists of one gate-forcing process that defines $f(\cdot)$, and two wind-forcing processes for $g(\cdot)$ that employ both the direction and velocity of the wind. Most importantly, this simple and plausible structure provides a close fit to the training data in both visual and quantitative terms.

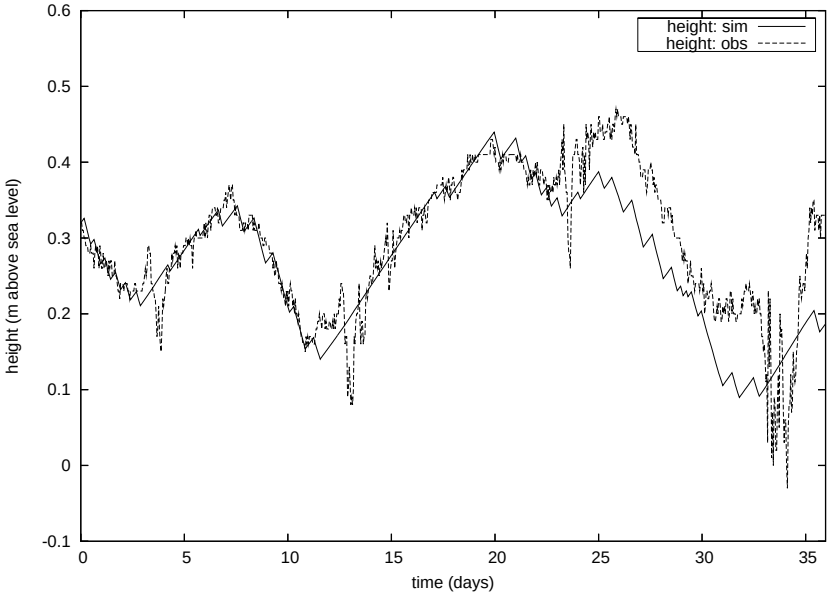

Figure 4. Simulated versus observed trajectories of the water level in Ringkøbing Fjord.

6. A Proposed Research Agenda

Although our initial results with IPM suggest the viability of inducing process models from observational data, they leave many questions unanswered. Before closing, we should discuss some issues that future work in the area should address and consider some promising approaches that should be explored within this research agenda.

6.1. Representation

Our initial foray into process modeling uses a language of somewhat limited scope that suggests numerous extensions. For instance, scientists often organize their models in terms of systems and subsystems to increase manageability. Likewise, processes may be decomposed into subprocesses, thereby creating a behavioral hierarchy that complements a scientist's structural one. In the same vein, scientists often organize related properties into groups that belong to particular objects within the system. Since the task of inductive process modeling includes the development of comprehensible models as a primary goal, researchers should investigate methods of incorporating these common forms of knowledge organization into the modeling language.

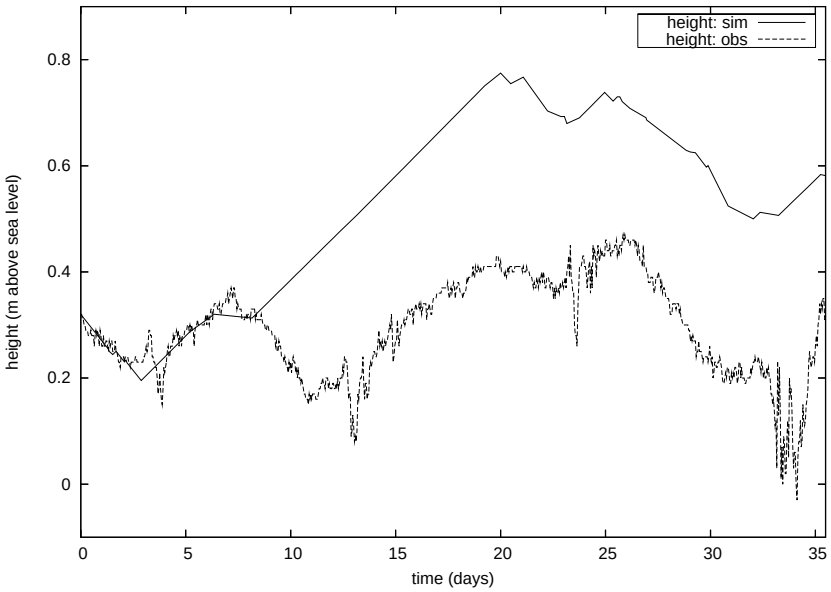

Figure 5. A trajectory produced by IPM for one cross-validation fold. Notice that after roughly eight days, the simulated trajectory diverges considerably as if the system were forced into a different regime.

Along with structural extensions to the representation, future work should explore methods for specifying and learning models with varying degrees of certainty. In particular, a model might consist of both quantitative and *qualitative* processes (Forbus, 1984), which use proportionalities to describe relations between variables. In this framework, exponential and logistic growth would map onto a single qualitative process stating that a population's growth rate is proportional to its size. Such models are appropriate for domains like molecular biology, where scientists often state their knowledge in qualitative form. Moreover, qualitative processes generally have fewer effective parameters than quantitative ones, which makes them useful for situations with few observations. Many issues that arise with quantitative models also occur with their qualitative analogs, so we also need work on this front.

6.2. Induction

In addition to dealing with representational issues, we also need research that addresses the traditional problems of model robustness and search efficiency. Overfitting the training data can arise in nearly every learning task, and we need ways to guard against this tendency, especially as we develop algorithms that generate more complex pro-

cess models. One avenue would examine analogs to methods that have proven successful in other induction paradigms. These include techniques for early halting in decision-tree construction using minimum description length and methods for postpruning using cross validation. Other techniques include ensemble methods like bagging, though this would require an adaptation of bootstrap sampling (Efron & Tibshirani, 1993) to handle time-series data and a means for generating a single, comprehensible model from the ensemble. In addition, researchers should explore other defenses against overfitting specific to process models.

While experimenting with IPM, we noticed that over 99% of the system's runtime was spent on parameter estimation, which suggests another avenue of research. Our current routines use variants of uninformed, gradient descent search. In contrast, search through the space of model structures relies heavily on domain knowledge, so it seems reasonable that one could use similar information to guide parameter estimation. For instance, our conversations with scientists indicate that they concentrate more on the high-level features of the trajectories (e.g., peak placement and height) than the residuals. Thus, developing methods for characterizing these features could lead to a more rational and more efficient means of tuning a model's parameters.

Research on inductive process modeling should also lead to extensions of the general search mechanism. For example, the current process-level constraints based on variable types could be augmented by model-level constraints based on processes or process types, so that a scientist can express both required and mutually exclusive processes. Knowledge about the dimensional units of variables would also constrain model induction, as would the introduction of knowledge that certain variables are conserved over time. Research should also continue on Bradley et al.'s (2001) use of qualitative patterns to characterize certain classes of equations. Other work should develop methods for learning both conditions on processes and new process forms.

An alternative approach to aiding model induction borrows an idea from work on theory revision. Rather than constructing a process model from scratch, one can instead start with a specific model and revise details to improve its fit to observations. Research on this topic should explore ways to revise a specific model's parameters, change the conditions on its component processes, replace these processes with others that relate the same variables, and even alter the basic structure of the initial model. Model revision will require the ability to remove components as well as add them, but otherwise the same issues arise as in the basic problem of process model induction.

6.3. Evaluation

Finally, future work should identify appropriate methods for evaluating process models. Since the data composing the trajectories fail to meet the independent and identically distributed assumption made by many classification algorithms, standard evaluation techniques such as cross validation must be adapted to better measure model performance. Additionally, dynamic domains can pass through several operating regimes, which poses additional difficulties for any evaluation method that samples from the original trajectory. Successful modeling would require the observation of each of these regimes during training. Also, some models may contain hidden variables for which the induction system identifies initial values, but we doubt that these values will apply to other data sets in which the starting state often differs from that used for training.

Although these challenges should be met, the evaluation of process models must move beyond an emphasis on predictive accuracy for the new paradigm to be useful to scientists. Research must also take into account considerations of model robustness and explanatory power. For example, in biological domains, researchers may evaluate a model based on both its sensitivity to parameters and its ability to match the general *shape* of observed trajectories. Such evaluation will require moving beyond one-to-one comparisons of observations and predictions to incorporate sensitivity analysis and to measure agreement with meaningful trends in the data.

In pursuing this research agenda, we should follow the accepted standards for established induction paradigms. Thus, papers should make explicit claims about a method's abilities and support them with experimental or theoretical evidence. Ideally, experimental studies should include a mixture of natural domains to ensure relevance and synthetic domains that let one vary dimensions of interest. However, the focus on familiarity and background knowledge recommends studies that involve collaborations with domain scientists or engineers, which remain uncommon in the machine learning literature. Finally, despite the distinctive nature of process model induction, researchers should incorporate ideas from other learning tasks and utilize existing methods as subroutines whenever sensible.

7. Concluding Remarks

In this paper, we proposed a new problem for machine learning researchers that addresses the induction of process models from observations. We defined this task as the construction of models that combine known component processes to explain time series or other continuous

data. We considered the challenges posed by process model induction and the potential of established techniques to address them, concluding that it demands research on new methods specialized to process modeling. We also presented an initial algorithm of this sort and demonstrated its functionality in multiple domains, after which we outlined a research agenda for future work on the topic.

Process models constitute a novel representation of knowledge that differs from the formalisms traditionally used in machine learning. These models are cast in the same terms as many scientific and engineering models, which should make them more communicable to practitioners in those fields. However, they have the same modularity as other formalisms that support learning, and they provide a clear facility for incorporating domain knowledge into learning mechanisms. We maintain that research on process model induction will broaden the scope of machine learning in significant ways, and we encourage others to join us in exploring computational methods that address this important new problem.

Acknowledgements

The research reported in this paper was supported by NSF Grant Number IIS-0326059. We thank Javier Sánchez, Kazumi Saito, Dileep George, Stephen Bay, and Nima Asgharbeygi for their work on early versions of the IPM system, along with Kevin Arrigo for data from and knowledge about the Ross Sea. A shorter version of this paper appeared as "Inducing Process Models from Continuous Data" in the *Proceedings of the Nineteenth International Conference on Machine Learning.*

References

Arrigo, K. R., Worthen, D. L., & Robinson, D. H. (2003). A coupled ocean-ecosystem model of the Ross Sea: 2. Iron regulation of phytoplankton taxonomic variability and primary production. *Journal of Geophysical Research*, *108*, 3231.

Berryman, A. A. (1992). The origins and evolution of predator–prey theory. *Ecology*, *73*, 1530–1535.

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

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth.

Bridewell, W., Sánchez, J. N., & Langley, P. (2004). An interactive environment for the modeling and discovery of scientific knowledge. Technical Report. Computational Learning Laboratory, CSLI, Stanford University, Stanford, CA.

Bunch, D., Gay, D., & Welsch, R. (1993). Algorithm 717: Subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. *ACM Transactions on Mathematical Software*, *19*, 109–130.

Cohen, S., & Hindmarsh, A. (1996). CVODE, a stiff/nonstiff ODE solver in C. *Computers in Physics*, *10*, 138–143.

DeJong, G. (1995). A case study of explanation-based control. In *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 167–175). Tahoe City, CA: Morgan Kaufmann.

Dietterich, T. G. (1990). Exploratory research in machine learning. *Machine Learning*, *5*, 5–9.

Džeroski, S., & Todorovski, L.(1993). Discovering dynamics. In *Proceedings of the Tenth International Conference on Machine Learning* (pp. 97–103). Amherst, MA: Morgan Kaufmann.

Efron, B., & Tibshirani, R.J. (1993). *An introduction to the bootstrap*. New York City: Chapman & Hall.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, *24*, 85–168.

Garrett, S., Coghill, G. M., Srinivasan, A., & King, R. D. (in press). Learning qualitative models of physical and biological systems. In S. D. Džeroski & L. Todorovski (Eds.), *Computational discovery of communicable scientific knowledge*. Berlin: Springer.

Ghahramani, Z. (1998). Learning dynamic bayesian networks. In C. L. Giles & M. Gori (Eds.), *Adaptive processing of sequences and data structures*. Berlin: Springer.

Iwasaki, Y. & Simon, H. A. (1994). Causality and model abstraction. *Artificial Intelligence*, *67*, 143–194.

Jost, C., & Ellner, S. (2000). Testing for predator dependence in predator-prey dynamics: A non-parametric approach. *Proceedings of the Royal Society of London B*, *267*, 1611–1620.

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

Langley, P. (1981). Data-driven discovery of physical laws. *Cognitive Science*, *5*, 31–54.

Lavrač, N. L., & Džeroski, S. D. (1994). *Inductive logic programming: Techniques and applications*. New York City: Ellis Horwood.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*, 47–80.

Murray, J. D. (2004). *Mathematical biology I: An introduction* (3rd ed.). Berlin: Springer.

Ourston, D., & Mooney, R. J. (1990). Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 815–820). Boston, MA: AAAI Press.

Pazzani, M. J., Mani, S., & Shankle, W. R. (2001). Acceptance by medical experts of rules generated by machine learning. *Methods of Information in Medicine*, *40*, 380–385.

Poritz, A. (1988). Hidden Markov models: A guided tour. In *Proceedings of the International Conference on Acoustic, Speech and Signal Processing* (pp. 7–13). New York City: IEEE Press.

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.

Schwabacher, M., & Langley, P. (2001). Discovering communicable scientific knowledge from spatio-temporal data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 489–496). Williamstown, MA: Morgan Kaufmann.

Tagliabue, A., & Arrigo, K. R. (2003). Anomalously low zooplankton abundance in the Ross Sea: An alternative explanation. *Limnology and Oceanography*, *48*, 686–699.

Todorovksi, L. (2003). *Using domain knowledge for automated modeling of dynamic systems with equation discovery.* Ph.D. Thesis, Faculty of Computer and Information Science, University of Ljubljana. Ljubljana, Slovenia.

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

Veilleux, B. G. (1979). An analysis of 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. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 1127–1134). San Francisco, CA: Morgan Kaufmann.

Williams, R., and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*, 270–280.

Żytkow, J. M., Zhu, J., & Hussam, A. (1990). Automated discovery in a chemistry laboratory. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 889–894). Boston, MA: AAAI Press.