Advances in Cognitive Systems 4 (2016) Submitted 4/2016; published 6/2016

Selective Induction of Rate-Based Process Models

Adam Arvay AARV914@AUCKLANDUNI.AC.NZ
Department of Computer Science, University of Auckland, Private Bag 92019, Auckland 1142 NZ

Pat Langley PATRICK.W.LANGLEY@GMAIL.COM
Institute for the Study of Learning and Expertise, Palo Alto, California 94306 USA

Abstract

This paper addresses the task of inductive process modeling, which involves constructing an explanation of multivariate time series in terms of background knowledge. We review earlier research on this problem, focusing on RPM, an induction system that is substantially more efficient and robust than its predecessors. We also examine three of its limitations: overeager binding of process instances, exhaustive search for component equations, and greedy search for consistent models. In response, we present SPM, a successor system that instead binds process instances dynamically, combines sampling with backward elimination to find complex equations, and uses constrained depth-first search to identify sets of consistent models. After this, we report empirical studies that demonstrate the benefits of these changes for model discovery in ecological and chemical domains. We conclude by discussing related research, along with ways to address remaining limitations.

1. Introduction

Research on computational scientific discovery has an extended and distinguished history in artificial intelligence (Shrager & Langley, 1990; Džeroski & Todorovski, 2007). Although this paradigm has features in common with work in data mining and machine learning, it differs in the aim of constructing laws and models stated in established scientific formalisms. As a result, the field has pursued separate problems, from the inference of quark models in particle physics to construction of reaction pathways in chemistry. Initial research in this area emphasized equation discovery (Żytkow & Langley, 1989), a topic important to many disciplines, but this task often arises early in a field's development, before scientists create models that explain observations in deeper terms. Explanatory model construction has received less attention, but some progress has occurred.

Langley et al. (2002) introduced one important form of such tasks – *inductive process modeling* – that uses background knowledge about processes that can occur in a domain to explain multivariate time-series data. The aim is to find a specific dynamic model, including numeric parameters, that reproduces the observed trajectories and predicts new values accurately. This model maps directly into a set of differential equations, but it also explains observations in terms of known processes. This distinguishes the paradigm from other problems that involve time series, including ones that find differential equations with no explanatory overlay. The role of background knowledge and explanation also make the task relevant to the cognitive systems community.

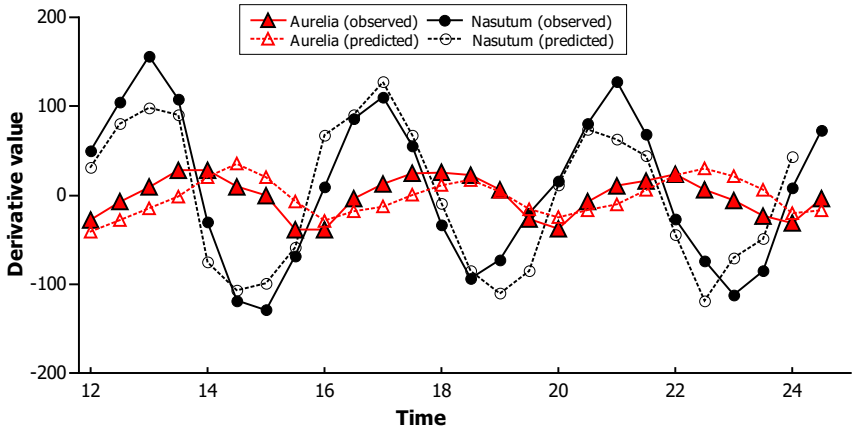

Figure 1. Observed and predicted derivatives for a predator-prey ecosystem comprising the protists *Nasutum* and *Aurelia*. The former trajectory derives from data reported by Veilleux (1979).

In this paper, we review earlier research on inductive process modeling, focusing on the RPM system, which is substantially more efficient and robust than its precursors. We also identify some remaining limitations and describe SPM, a new system that builds on RPM but incorporates changes to representation and search that address each of them in turn. Next we report experiments on ecological and chemical domains that demonstrate SPM's superiority over the its predecessor. In closing, we consider connections to other research and discuss promising directions for future efforts. Our work builds directly on previous results, making it incremental, but the impacts on ability are substantial, making them noteworthy in scientific terms.

2. Prior Work on Inductive Process Modeling

We can specify the abstract task of inductive process modeling in terms of inputs and outputs:

- *Given:* A set of typed variables and observed trajectories of their values over time;
- *Given:* A subset of these variables whose values should be explained;
- *Given:* Generic background knowledge about processes that might appear in the explanations;
- *Find:* A quantitative process model that explains variables' trajectories and predicts future values.

The trajectories in Figure 1 and the process model in Table 1 should clarify these ideas. The figure shows measurements reported by Veilleux (1979) for a simple ecosystem in which the protist *Nasutum* feeds on *Aurelia* that demonstrate a classic predator-prey cycle. The model in the table, which includes three processes, reproduces these data with low error. Each process has a name, an expression that determines its rate, one or more derivatives that are proportional to this rate, and parameters that specify the sign and amount of this influence. Process rates are directly analogous to the rates of chemical reactions, which can vary over time.

Multiple processes in a model can affect the same derivative term. If we assume that these influences are additive, then we can compile any model into a set of ordinary linked differential equations. In this case we obtain the result

Table 1. A three-process model for an ecosystem involving the predator *Nasutum (nas)* and the prey *Aurelia (aur)* that reproduces the two trajectories in Figure 1.

exponential_change[aur]	*exponential_change[nas]*
rate $r = aur$	*rate* $r = nas$
parameters $A = 1.48$	*parameters* $B = -1.12$
equations $d[aur] = A \cdot r$	*equations* $d[nas] = B \cdot r$
holling_predation[nas, aur]	
rate $r = nas \cdot aur$	
parameters $C = 0.0047$, $D = -0.023$	
equations $d[nas] = C \cdot r$, $d[aur] = D \cdot r$	

$$
\begin{aligned}
d[aur] &= 1.48 \cdot aur + -0.023 \cdot aur \cdot nas \\
d[nas] &= -1.12 \cdot nas + 0.0047 \cdot aur \cdot nas \ .
\end{aligned}
$$

However, the model moves beyond these equations to explain the observations in terms of the familiar processes of organism growth, loss, and predation. This in turn requires background knowledge about ones that may operate in the domain. This knowledge takes the form of *generic* processes with a form similar to those in models, but that replace specific variables with typed abstractions (e.g., *nas* with *x [predator]* and *aur* with *y [prey]* in *holling_predation*), rate formulae with algebraic expressions (e.g., $r = x \cdot y$), particular parameter values with ranges (e.g., $C > 0$ and $D < 0$), and equations with functional forms (e.g., $d[x] = C \cdot r$ and $d[y] = D \cdot r$).

Since its introduction (Langley et al. 2002), research on inductive process modeling has made steady progress. The framework has produced positive results in ecology (Asgharbeygi et al. 2006), biochemistry (Langley et al. 2006), and hydrology (Bridewell et al. 2008), and algorithmic extensions have added the ability to handle missing values, combine models into interpretable ensembles, and incorporate constraints to reduce search. However, most approaches to process model induction have been limited in three ways. First, although process models are inherently compositional, early systems generated complete model structures using constrained exhaustive search, then carried out gradient descent to fit each one's parameters. Second, this parameter estimation relied on repeated simulation to obtain model errors and thus revise parameters, sometimes taking hundreds of iterations to reach local optima. Finally, even these expensive operations sometimes led to poor parameter estimates that fit the trajectories inadequately.

In recent work, Langley and Arvay (2015) have reported RPM, a system that overcomes these problems. They introduced the assumptions, used in our example, that every process has an associated rate on each time step, that they have one or more associated derivatives that are proportional to its rate, and that these rates are determined by parameter-free algebraic expressions. Drawing on these assumptions, RPM estimates the derivative for each variable on each time step using 'center differencing', which averages a number of adjacent differences. The system also generates a set of process instances by binding generic processes with variables in all possible ways consistent with their type constraints. From the rate expressions associated with each one, it then calculates the rate of each process instance on each time step.

Using these derived values, RPM carries out greedy search through the space of process models. For each derivative term D in turn, it uses multiple linear regression to find an equation that predicts D as a linear function of rate terms. The system first attempts this with the rates of single processes as predictive variables. If any produces an equation with r^2 above threshold, it uses this relationship; if not, it considers all pairs of process rates, and so forth, continuing until it reaches a maximum number of combinations. RPM requires later differential equations to include processes that are consistent with those in earlier ones. For instance, if it includes *holling-predation[nas, aur]* in the *aur* equation, it must also do so in *nas* regression and vice versa. This approach to inductive process modeling combines the robustness and efficiency of multivariate regression with the use of background knowledge to ensure consistent explanations.

Experimental studies of RPM's behavior showed it was more reliable than SC-IPM (Bridewell & Langley, 2010), an earlier system for process model induction, and that it was far more efficient, running 83,000 times faster on even simple tasks. The program also dealt well with noise and it scaled reasonably with increases in the number of model equations and the number of generic processes. However, RPM's developers noted that a number of drawbacks remained, which the work reported in the next section aims to address.

3. An Extended Approach to Process Model Induction

In recent research, we have developed SPM, a system that incorporates many assumptions introduced by its predecessor, RPM, but that also addresses some of the system's remaining limitations. Here we describe the primary differences between the two programs, both of which are implemented in Common Lisp using subroutines for multivariate regression from the public domain.

3.1 Flexible Components of Process Models

One drawback of RPM was that it generated many process instances at initialization time. Not only did this lead the system to generate more process instances than necessary at the outset, but it could lead model induction astray through overeager binding of variables to processes. SPM addresses this drawback by using a more flexible notation for generic processes and by instantiating initially only with those variables mentioned in their rate expressions.

For example, in chemistry, RPM would require three distinct generic processes with two inputs, x and y, one with a single output, another with two outputs, and another with three outputs. Each might have rates that are equivalent to $x \cdot y$. Given five chemical substances, say A, B, C, D, and E, RPM would generate 49 process instances with seven possible pairs of inputs, some 21 with one output, another 21 with two outputs, and seven with three outputs. Moreover, during model construction, RPM might introduce an equation for C that incorporates one such process, say that A and B react to form C and D, when in fact the correct model involves A and B react to form C and E. The need to compose a process model from repeated choices of this sort makes it an important issue, as it increases the branching factor in an exponential search task. Without the ability to backtrack, RPM is likely to make overeager commitments to some process instances in favor of others without evidence that they are preferable.

An even more serious problem is that the matrix manipulation routine RPM used for multiple linear regression assumes that rows do not have identical entries. When this procedure encounters two or more process instances with the same rates on each time step, it exits with an error. As a re-

sult, RPM cannot find any models when some candidate processes have the same rates, which occurs whenever those rates are determined by a subset of the variables in a generic process. This includes a broad class of process modeling tasks, with ones that involve systems of chemical reactions being an important subset that has practical relevance.

In the same setting, SPM would instead encode a single generic process with two inputs, x and y, and with zero or more outputs. As a result, the system would generate only seven partially instantiated process instances at initialization time. Moreover, it would decide to incorporate non-rate variables only as justified by equations' ability to fit the data. For instance, it might decide to incorporate the process instance with A and B in its rate expression into the differential equation for C and decide later whether to add other non-rate variables, like D or E, depending on whether their changes appear to be influenced by the process's rate. In addition to reducing the number of process instances generated during search, this approach lets SPM delay commitment about including non-rate variables until it finds empirical evidence, and it avoids the problem with replicated rates that caused RPM serious issues. Thus, the new system should be able to find process models for a broad class of settings, including ones for sets of chemical reactions, while its predecessor could not.

3.2 Heuristic Search for Component Equations

Another drawback of the RPM system was its reliance on exhaustive search for individual differential equations. The system kept this tractable by ordering search from simple to complex candidates and placing a maximum on the number of processes considered. This was practical for domains that involve many variables with only few interactions, but it could not induce models with more than a few processes (rate expressions) in the right-hand side of each equation. Moreover, the number of candidate equations grew very rapidly with irrelevant processes, making the problem worse.

The natural response is to replace this exhaustive search for equations with a heuristic method. We need to identify a subset of processes that should appear in an equation, after which estimating their coefficients using regression will be straightforward. This maps directly onto the task of feature selection in supervised classification learning (Blum & Langley, 1997), which borrows from an older literature in stepwise regression. Methods for 'forward selection' seem attractive, as they search from simple to complex feature sets, but preliminary studies showed they will not work with our rate terms, which can be highly correlated. For instance, if the correct equation is $d[z] = 2xy + 3yz + 4xz$, the r^2 score for an equation that contains only one or two of these predictors is no higher than for candidates with irrelevant terms instead. Only if all three predictors are present is the score high enough to identify them as useful.

Fortunately, the converse does not hold. When given all relevant terms but also irrelevant ones, multivariate linear regression produces an equation that is just as accurate as when given only relevant variables. This suggests that we use 'backward elimination', another standard technique for feature selection, that starts with all terms and removes one variable at a time until the r^2 value drops substantially. But this method is ineffective for large numbers of terms due to round-off errors when the predictor variables are highly collinear,[1] which is common in our domains and which causes high variance in the regression estimates for coefficients. Backward elimination is usually reliable for ten or fewer terms, but our problems involve many more candidate predictors.

1. A set of variables is collinear if one can predict some of them with high accuracy as a linear combination of others.

In response, SPM combines backward elimination with repeated sampling. For each dependent variable, the equation-finding module repeatedly selects a subset of k processes, with uniform probability. Using the derivatives and process rates for each time step, it then invokes multivariate regression to construct a linear equation that predicts the former in terms of the latter. If this equation's r^2 score is lower than a user-specified threshold, then SPM abandons it. Otherwise the system ranks the processes by their coefficients' absolute values and removes them in turn, smaller values first, rerunning the regression algorithm in each case. This continues until either the r^2 score falls below threshold or the ratio between new and old r^2 scores is less than a specified fraction. The result is a differential equation with only those processes needed for high predictive accuracy. Like its predecessor, SPM checks its coefficients to ensure they satisfy the numeric constraints associated with each process. It retains the equation for future use in this case and abandons it otherwise.

Either way, SPM then selects another sample of rate terms, without replacement, and repeats the procedure. This continues until the module has either found n distinct equations with r^2 scores above threshold or it has examined a user-specified number of samples without reaching the desired number of acceptable equations. At this point, it returns those equations with adequate scores, including the processes used as their predictive terms and the coefficients associated with each one. The program repeats these steps separately for each dependent variable, removing any redundant equations that it reaches from different starting samples. This strategy should let SPM scale, unlike RPM, to domains like biochemistry, where a single metabolite can be influenced by many reactions.

3.3 Search for Consistent Process Models

As noted earlier, RPM carried out greedy search at the level of process models, with each candidate model comprising a set of linked differential equations. The system constrained this search by requiring that, if a process p that contains a derivative $d[x]$ appears in an equation for some other derivative, then that process p must also appear in the equation for $d[x]$. This ensured that processes are used consistently across different equations, which is an essential aspect of process explanations. However, RPM's greedy search meant that, if it incorporated an incorrect equation early in model induction, it could not overcome this decision later. Another drawback was that it found only one process model, even if alternative accounts could explain observations equally well.

Other approaches would carry out depth-first search, by utilizing backtracking, or beam search, by entertaining a number of partial models in parallel. Both could retain the ability to restrain later equations by taking into account processes already included, but they would also require repeated induction of the same equations during lower levels of the search tree, which would be inefficient. A more promising method would find a set of differential equations separately for each dependent variable, then attempt to combine elements of each set into one or more consistent process models. This scheme would not reduce equation-level search by constraining later equations to include processes in earlier ones, but it would still check for such consistency at combination time.

SPM incorporates this latter approach, calling repeatedly on the equation-induction module described earlier to find, for each of n dependent variables, a set of e or fewer alternative differential equations. Next it selects one equation from each set in an effort to find all models that explain the multivariate time series in a consistent manner. To this end, it carries out depth-first search, with the first level considering equations for one dependent variable, the second for another one, and so

on, to level n. However, search is not exhaustive because, like its predecessor, the system checks each equation for consistency with earlier ones in terms of shared processes. In the worst case, it may consider e^n different models, but it will typically reject many candidates at the second or lower levels, reducing the effective branching factor substantially. In summary, a key difference from RPM is that SPM finds multiple models that explain the data adequately, achieving this in a robust manner by separating induction of equations from their combination into consistent models.

4. Experimental Evaluation

Although we designed SPM to overcome its predecessor's drawbacks, whether it accomplishes this aim is an empirical question. In this section, we review our claims about the new system and report experiments that support them. First we show that SPM has broader coverage than RPM, in that it can induce not only those process models the earlier system handled but others as well. Next we compare the two programs' methods for finding individual equations, demonstrating that SPM scales better than its predecessor. Finally, we examine their relative abilities for inducing complete models that incorporate processes consistently. Because our purpose is to show that SPM's extensions are beneficial, only comparisons to RPM are appropriate; reporting results to other systems for inductive process modeling (e.g., Bridewell & Langley, 2010) would offer no scientific insights.

4.1 Basic Capabilities of the System

One difference between RPM and SPM is that the former instantiates process instances fully at initialization time, whereas the latter delays binding of variables that do not appear in the rate expression. Eager commitment can keep RPM from inducing models in domains like chemistry, where it generates multiple processes with the same rate. Instead, SPM waits until it finds evidence that a variable not yet included in a process instance benefits from appearing in its differential equation, then binds it at that point. This suggests a testable hypothesis about the systems' abilities:

- *SPM induces a superset of the models found by RPM that adequately explain the observations.*

To evaluate this claim, we first ran both systems on five different ecological time series. These included the natural data on protist interactions described earlier, noise-free synthetic data[2] for two predator-prey settings that involved six interacting organisms, and synthetic observations for an aquatic ecosystem involving zooplakton, phytoplankton, two nutrients, and detritus. For the natural data set, RPM and SPM found the same process model, with $r^2 = 0.84$ for the *d[Aurelia]* equation and $r^2 = 0.71$ for *d[Nasutum]*. For the synthetic domains, both systems successfully found the target model used to generate the data, although in some cases SPM also found models with similar r^2 scores, as we discuss later.

We also demonstrated that, like RPM, the new system can reconstruct a 20-variable predator-prey model from the multivariate trajectories shown in Figure 2. For this task, we had SPM consider differential equations with ten or fewer rate terms, find one such equation for each variable, and

2. We have used synthetic data for most of our studies, both because time series for our scientific domains are difficult to obtain and because they give control over features of the task. Langley and Arvay (2015) reported that smoothing the time series lets RPM handle up to ten percent noise, so we did not focus on that issue in our experiments.

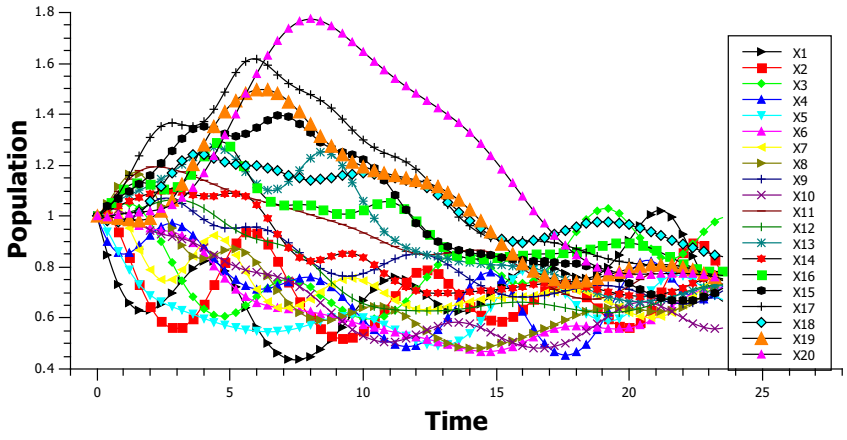

Figure 2. Observed trajectories for a 20-variable predator-prey system in which the organisms are organized in a linear food chain. Both RPM and SPM reconstruct the target process model that generated these data.

sample no more than 1,000 subsets of rate expressions. Over 50 runs, the system reliably found a single consistent model, which corresponded to the target, in 1.35 ± 1.28 CPU seconds, as compared to $0.38 \pm .018$ seconds for its predecessor. SPM also reliably reconstructed a six-variable predator-prey model when given not only two relevant generic processes, but 16 irrelevant ones with other rate expressions that produce 324 distinct process instances. Here the system required 4,000 samples to find each equation and took 2.7 ± 1.2 seconds on average compared to RPM's 9.1 ± 0.2 seconds.

In addition, we ran both systems on a number of synthetic chemical data sets. Table 2 shows one system involving six interacting chemicals connected via eight reactions (processes). One process involved a time-varying influx variable Z that keeps other variables from reaching a steady state. In another chemical data set, seven chemicals participated in 12 reactions, including a time-varying influx. SPM encountered no difficulty inducing either reaction network from multivariate trjectories with at least 80 time steps. In the first case, the system generated 22 process instances from three generic processes, then took 1,000 samples of six rate terms to identify each component equation. In the second case, it generated 46 processes from four generic processes, then took 15,000 samples of ten rate terms. Runs on the first data set required 14.7 ± 0.21 CPU seconds on average, whereas those for the second took a mean of 111.8 ± 0.6 seconds. In contrast, RPM generated 63 process instances from analogous generic structures and could not induce either target model. As anticipated, its greedy algorithm interacted with its eager binding of variables in processes, leading to inclusion of incorrect process instances it could not retract during later stages of model construction. These runs demonstrate that SPM can induce chemical process models that its predecessor cannot handle.

In summary, the new system improves on RPM by delaying the binding of variables that do not appear in rate expressions of generic processes. This lets SPM avoid the problem of replicated rates that keeps its predecessor from finding a broad class of quantitative process models, including systems of linked chemical reactions. The new approach also reduces the number of process instances generated during induction, but this efficiency gain is less important than the increased coverage of discoverable process models.

Table 2. Differential equations for a chemical system with six variables that interact through eight distinct reactions. SPM can reconstruct this model, with minor parameter differences, from time series that it generates whereas RPM cannot.

$$dX1/dt = 1.1 \cdot X2 \cdot X3 - 1.6 \cdot X1$$
$$dX2/dt = 1.8 \cdot X1 - 1.5 \cdot X2 - 1.0 \cdot X2 \cdot X3 + 0.9 \cdot X5 \cdot X6$$
$$dX3/dt = 1.9 \cdot X1 + 1.1 \cdot X2 - 1.3 \cdot X3 - 1.3 \cdot X2 \cdot X3$$
$$dX4/dt = 0.9 \cdot X2 + 0.8 \cdot X3 - 2.5 \cdot X4 \cdot X5 + 0.5 \cdot X5 \cdot X6$$
$$dX5/dt = 0.9 \cdot X3 - 1.8 \cdot X4 \cdot X5 + 0.9 \cdot Z$$
$$dX6/dt = 2.3 \cdot X4 \cdot X5 - 0.8 \cdot X5 \cdot X6 - 0.5 \cdot X6$$

4.2 Scalable Induction of Differential Equations

As noted earlier, SPM's approach to finding individual differential equations differs substantially from that of its predecessor. RPM carries out exhaustive search for the simplest equation with an acceptable r^2 score, starting with one-term candidates and adding terms until reaching a maximum number. The new system combines sampling of rate terms (processes) with backward elimination to identify subsets that are good predictors of derivatives. This suggests a second hypothesis:

- *As the number of terms in a target equation increases, their induction time for SPM grows more slowly than for RPM*.

To test this prediction, we examined the behavior of their modules for equation induction in isolation. We generated synthetic data in which derivatives were a linear function of different numbers – from one to ten – of processes with random valued rates. The random data ensured that the terms in each equation were not highly correlated, thus containing redundant information. We ran each system ten times on each equation and measured the CPU time needed to find it. We fixed the number of samples at 10,000 and the number of sampled rate terms at 13 for all SPM runs.

Figure 3 presents the results of this experiment. RPM actually finds simpler equations slightly more rapidly than SPM, as they are consistent with its simplicity bias and it avoids the cost of repeated sampling. However, this changes for equations with five processes, at which point SPM becomes faster. In fact, there were so many combinations of nine-term equations that RPM could not finish generating them, making the system unable to complete its runs. Growth in CPU time for SPM was approximately linear, as it depended on the number of samples and the number of processes in each sample, as specified by the user.

Analysis reveals that SPM's sampling approach does not guarantee it will find the appropriate equation. For this to happen, the correct set of rates must appear in the sampled set and feature selection must correctly identify them as relevant. We can calculate the probability that the correct combination of rates will appear in a sample as $\binom{T}{S}\binom{T-S}{S-R}/\binom{T}{R}$, where T is the total number of processes, R is the number of rates that appear in the target equation, and S is the size of the sample. Additional sampling increases the odds of finding an equation but increases CPU time further, which is a natural tradeoff. Nevertheless, it seems clear the new system scales better to equation complexity than its precursor.

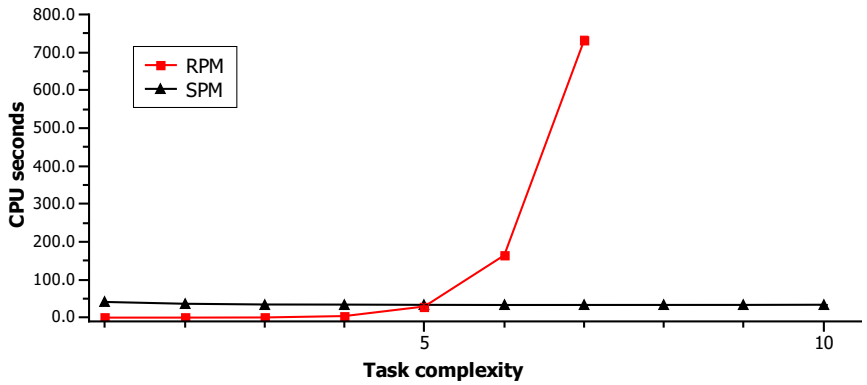

Figure 3. Average time for RPM and SPM to find target equations, in CPU seconds, with different numbers of rate terms (processes). The SPM curve is actually linear, but the slope is so low that it appears constant.

4.3 Improved Induction of Consistent Models

Another difference between our approach to process model induction and its precursor lies in their search for consistent models. Rather than relying on a greedy method aided by process constraints, SPM first finds alternative equations for each dependent variable and then uses depth-first search to find all ways to combine them into models. This suggests a final hypothesis about the two systems:

- *SPM induces a more complete set of consistent process models than RPM and has greater chances of recovering the target model.*

This claim seems straightforward to test, but we have already seen that RPM's greedy search is sufficient to find complex ecological models, and its inability to induce chemical reaction networks is due mainly to eager binding of variables in processes. However, we can modify SPM's parameters to approximate greedy search through the space of process models.

Thus, we ran a parametric study in which we compared the behavior of the multi-equation SPM with a variant that finds only one differential equation for each dependent variable. We ran both versions on the same synthetic data sets used earlier, some generated from predator-prey models and others from chemical reaction networks. For each condition, we ran the systems 20 times and recorded both the total number of consistent models induced, as well as the percentage of times they found the target model. Table 2 shows that, on the five ecosystem data sets, each variant reliably found a single model that was equivalent to the target. In contrast, on the two chemical data sets, the 'greedy' version was unable to find the correct model, whereas the full SPM generated several consistent models, in each case finding the target. Naturally, the full variant took longer to run (14.7 and 111.8 CPU seconds, respectively) than the greedy version (1.17 and 1.65 CPU seconds), but there is a natural tradeoff between time and coverage. The chemistry B data set was particularly challenging and needed more time to find consistent models reliably. We should emphasize that all additional models SPM found were internally consistent in terms of processes and had comparable r^2 scores. One cannot distinguish them given the data and the system's background knowledge.

Table 3. The probability of finding a target model by greedy and multi-equation variants of SPM on ecological and chemical data sets, along with average CPU time.

	Greedy SPM		Multi-Equation SPM	
	Percent	CPU	Percent	CPU
Nas-Aur	100	0.004±.002	100	0.004±.001
Aquatic Ecosyst	100	0.03±.012	100	0.12±.007
Predator Prey 6a	100	0.01±.003	100	0.03±.004
Predator Prey 6b	100	0.83±.004	100	2.63±.008
Predator Prey 20	100	0.81±.028	100	4.10±.100
Chemistry A	0	1.17±2.03	100	14.7±.210
Chemistry B	0	1.65±1.27	100	111.8±.610

To summarize, SPM's incorporation of depth-first search lets it find multiple explanations that are consistent with the data, as opposed to RPM's more limited capacity to return a single process model. The latter's inability to induce some classes of models ruled out a direct experimental comparison of the two systems, but studies with a greedy version of SPM revealed that devoting additional effort not only lets it uncover multiple accounts that fit the observations equally well, which is desirable in its own right, but also increases greatly the chances of inferring the underlying model that generated the multivariate time series.

5. Related Research

We have already explained how SPM builds on a long tradition of research on inductive process modeling. Our system addresses the same basic discovery task as other work in this paradigm, although it takes advantage of ideas introduced by Langley and Arvay (2015) to make the problem more tractable. We have retained RPM's assumptions that each process has an associated rate that is determined by an algebraic expression and derivatives that are proportional to this rate. This idea comes originally from Forbus's (1984) Qualitative Process Theory, which used a similar notation for qualitative models of physical systems. SPM introduces improved mechanisms for inducing quantitatve process models, but it benefits from many earlier ideas. The use of background knowledge in inductive logic programming is similar in spirit but very different in practice, as it acquires models from relational rather than numeric data and it typically relies on separate-and-conquer methods that are inappropriate for linked sets of differential equations, which produce strong interactions among the shared variables.

We should also discuss other work on inducing differential equation models that falls outside the process modeling paradigm. A number of efforts have drawn on other forms of background content to constrain search through the space of model structures. Bradley et al. (2001) report an approach that uses knowledge about the qualitative behavior of different forms of equations to limit the space of functions considered. For example, their system uses knowledge about types of trajectories that linear differential equations can produce to reject this class when it observes different behavior.

Evaluating candidates through parameter estimation occurs as a last resort, as it requires far more computation than qualitative reasoning. We should explore using similar techniques in SPM to reduce the number of functional forms that it entertains.

Džeroski and Todorovski (2008) adopt another approach that uses a context-free grammar to specify the space of functional forms examined during equation induction. The grammar includes rules that restrict the forms of algebraic expressions that can appear on the right sides of differential equations. Their system can also take into account structures that are partially known. Both forms of background knowledge reduce the size of the search space greatly. Arvay and Langley (in press) show how another relative of RPM can adapt process models induced in one setting to new data sets, but it could benefit from additional constraints on how to combine processes. The main drawback of their approach is that, without a notion of process, it remains unclear how to explain why particular elements appear in an equation, but we could borrow SC-IPM's use of constraints to guide inductive process modeling (Bridewell & Langley, 2010).

Finally, Srividhya et al. (2007) describe a technique that uses knowledge about molecular structures to constrain search for biochemical pathways. Their system generates a complete set of possible reactions that are consistent with chemical laws and creates candidates from these components. One variant starts from an empty model and adds one reaction at a time; another version starts from all reactions and discards them one after another. The authors draw on ideas from graph theory to quantify model complexity, which their system uses to determine its halting criterion. SPM already encodes domain constraints in its library of generic processes, but adding a complexity metric based on model connectivity could let it select better candidates. Srividhya et al.'s approach appears to work well for chemical systems, but some elements may not apply to other domains.

A different line of work instead utilizes extensive search, typically some variety of evolutionary algorithm, to find differential equation models that fit time series. Examples of this approach include Koza et al. (2001), Schmidt and Lipson (2009), and, most recently, Lobo and Levin (2015). These techniques are adept at inducing models that match observed trajectories, but do not lend themselves to using background knowledge during search. This means they can obtain results that humans find difficult to interpret, making them quite different in spirit from methods for inductive process modeling. We have not compared these systems to SPM empirically because they focus on a different class of problems and search a substantially larger space, which makes them incomparable. However, evolutionary search methods might be more effective at selecting rate terms for equations than our current use of random sampling.

6. Directions for Future Work

We have shown that SPM provides greater coverage, efficiency, and reliability than RPM, but there remain several areas where we can still improve the system. One drawback of the current implementation concerns its reliance on user-specified parameters. These include the maximum r^2 value needed to accept an equation, the number of processes to sample from those available, the number of equations to find for each derivative, and the number of samples taken when searching for them. SPM shares the first parameter with its predecessor, but we introduced the others when adding its new techniques. The system's behavior can be sensitive to these settings when variables are highly collinear, and we should explore ways to mitigate this dependence.

SPM system also shares two important assumptions with RPM. One is that the rates of processes are determined by parameter-free algebraic expressions. This assumption is violated in chemical models that include the Michaelis-Menten equation and in some ecological accounts. Future versions should estimate such parameters at the same time as they fit the coefficient for each process. Nonlinear least squares estimation offers one promising approach, but it requires an initial parameter value, upper and lower bounds, and stopping criteria; the technique also requires more computation and it can halt at local optima. SPM's current method treats equations separately, so that its parameter space includes only those terms that appear in a given equation, not elsewhere in the model. Moreover, once it finds a parameter for one equation, it reuses its value in later equations that include it, reducing dimensionality further. Our initial efforts with gradient descent approaches have been promising, but the results are too preliminary to discuss here.

The second shared assumption is that all variables are observed. This lets both systems transform parameter estimation for a set of linked differential equations into separate regression tasks, which is far more tractable. SPM cannot find equations for observed variables if any of the rate expressions include an unobserved variable, as it cannot calculate the latter's rates, which are needed during the regression procedure. However, the system can return a partial model that includes equations influenced only by observed variables. This offers a way to constrain the values of unobserved variables if they are influenced by multiple processes. The latter must still follow the relations specified in the generic process library, so a given rate equation may contain one unobserved variable, with the rest being observed. These serve as constraints on the unknown values that, again, we can use gradient descent techniques to estimate. Our initial studies with this approach have been encouraging, but it applies only to certain model configurations that involve observed and unobserved variables, and our results are not mature enough to report here.

A related challenge involves SPM's reliance on a library of generic processes that contain the elements needed to construct an adequate model. If some processes are missing, then the system can only induce a partial model for which some equations are incomplete. However, even such a partial model can serve as the starting point in a search for entirely new processes. Consider the simplest case, in which a target equation lacks only one unknown process. An extended SPM could then generate candidate rate expressions as algebraic combinations of observed variables, determine which of these improves the r^2 score sufficiently, and retain them as the core of new processes. If some of these rate terms also improve the fit for other equations, this offers further evidence they describe the missing process. One could apply the same idea to situations in which two or more unknown processes contribute to an equation, but this would require far more search.

A final area of investigation would expand the framework to handle qualitative process models and simulation. SPM's processes offer a qualitative overlay that provides context on differential equations, but not all scientific models or data are numeric. Qualitative models arise in fields that involve imprecise phenomena, like biology, where experiments produce results noisy enough that scientists trust only their direction, not their degree. We can replace SPM's algebraic expressions with qualitative influences to produce a variant on Forbus' (1984) framework. Given a model with such qualitative processes, we can use his techniques to generate qualitative trajectories, or *envisionments*, for each variable, which we can in turn compare to observations. The problem is that envisionments are often nondeterministic, in that they describe multiple possible trajectories. One

response is to use semi-qualitative models that retain parameter-free functional expressions for process rates, but omit parameters for the linear relations between derivatives and rates. Instead, such a model would include, for each derivative term dX/dt, an algebraic constraint that predicts when the derivative is positive or negative. If, for each such term, we label the combination of observed values on each time step as associated with positive or negative derivatives, we can use supervised learning methods to find algebraic bounds that predict the direction of change on future time steps.

7. Concluding Remarks

In this paper, we reviewed earlier work on the task of inductive process modeling, focusing on the RPM system and its limitations. We also described SPM, a new system that shares its predecessor's basic framework but that introduces three innovations. These include (1) delayed binding for variables that do not appear in processes' rate expressions until evidence suggests they would benefit, (2) combination of backward elimination with sampling to induce more complex equations, and (3) constrained depth-first search to identify process models with consistent equations. We reported experiments that demonstrated these novel elements let SPM find chemical process models that RPM cannot handle, induce complex equations in a more scalable manner, and discover multiple consistent models when they are available, rather than being limited to only one explanation.

We also discussed related research on induction of differential equation models. Most other work on this topic does not adopt a process framework, making their results less explanatory than descriptive, but we reviewed a number of earlier systems that used background knowledge to guide heuristic search through the space of models. Finally, we proposed a number of directions in which to augment our framework for inductive process modeling – reducing SPM's reliance on user-specified parameters, handling processes with parameters in their rate expressions, inferring the values of unobserved variables, inventing entirely new processes, and inferring qualitative accounts. Taken together, these extensions should improve considerably the range and reliability of computational methods for discovering explanatory process models.

Acknowledgements

The research reported in this paper was supported in part by Grant No. N00014-11-1-0107 from the US Office of Naval Research, which is not responsible for its contents. We thank Will Bridewell, Sašo Džeroski, Ruolin Jia, Thomas Lumley, and Ljupčo Todorovski for useful discussions that led to the approach and results we have described.

References

Asgharbeygi, N., Bay, S., Langley, P., & Arrigo, K. (2006). Inductive revision of quantitative process models. *Ecological Modelling*, *194*, 70–79.

Arvay, A., & Langley, P. (in press). Heuristic adaptation of quantitative process models. *Advances in Cognitive Systems*.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*, 245–271.

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

Bridewell, W. & Langley, P. (2010). Two kinds of knowledge in scientific discovery. *Topics in Cognitive Science*, *2*, 36–52.

Bridewell, W., Langley, P., Todorovski, L., & Džeroski, S. (2008). Inductive process modeling. *Machine Learning*, *71*, 1–32.

Džeroski, S., Langley, P., & Todorovski, L. (2007). Computational discovery of scientific knowledge. In S. Džeroski & L. Todorovski (Eds.), *Computational discovery of communicable scientific knowledge*. Berlin: Springer.

Džeroski, S., & Todorovski, L. (2008). Equation discovery for systems biology: Finding the structure and dynamics of biological networks from time course data. *Current Opinion in Biotechnology*, *19*, 360–368.

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

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

Langley, P., & Arvay, A. (2015). Heuristic induction of rate-based process models. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. Austin, TX: AAAI Press.

Langley, P., Sanchez, J., Todorovski, L., & Džeroski, S. (2002). Inducing process models from continuous data. *Proceedings of the Nineteenth International Conference on Machine Learning*. (pp. 347–354). Sydney: Morgan Kaufmann.

Langley, P., Shiran, O., Shrager, J., Todorovski, L., & Pohorille, A. (2006). Constructing explanatory process models from biological data and knowledge. *Artificial Intelligence in Medicine*, *37*, 191–201.

Langley, P., & Żytkow, J. M. (1989). Data-driven approaches to empirical discovery. *Artificial Intelligence*, *40*, 283–312.

Lobo, D., & Levin, M. (2015). Inferring regulatory networks from experimental morphological phenotypes: A computational method reverse-engineers planarian regeneration. *PLoS Computational Biology*, *11*, e1004295.

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, *324*, 81–85.

Shrager, J., & Langley, P. (Eds.) (1990). *Computational models of scientific discovery and theory formation*. San Francisco: Morgan Kaufmann.

Srividhya, J., Crampin, E. J., McSharry, P. E., & Schnell, S. (2007). Reconstructing biochemical pathways from time course data. *Proteomics*, *7*, 828–838.

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