From *Proceedings of the Nineteenth International Conference on Machine Learning* (2002). Sydney: Morgan Kaufmann.

Inducing Process Models from Continuous Data

Pat Langley LANGLEY@ISLE.ORG
Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306 USA

Javier Sanchez JSANCHEZ@CS.STANFORD.EDU
Computational Learning Laboratory, Center for the Study of Language and Information, Stanford University, Stanford, CA 94305 USA

Ljupčo Todorovski LJUPCO.TODOROVSKI@IJS.SI
Sašo Džeroski SASO.DZEROSKI@IJS.SI
Jožef 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 a population dynamics domain. In closing, we describe open issues in process model induction and encourage other researchers to tackle this important problem.

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. 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; 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, models in science and engineering often provide an *explanation* which includes variables, objects, or mechanisms that are unobserved, but that help predict the behavior of observed variables. Moreover, explanations often make use of general concepts or relations that occur in different models. One example is Newton's theory of gravitation, which moved beyond Kepler's descriptive laws to an explanation of planetary trajectories in terms of straight line motion and attractive force.

We claim that explanations in science and engineering are often stated in terms of generic *processes* from some domain. We will focus here on a particular class of processes that describe one or more causal relations between input variables and output variables. A process states these relations in terms of differential equations (for a process that involves change over time) or static equations (for one that involves instantaneous effects). A 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 input variables with observable output variables, possibly through unobserved theoretical terms.

Table 1 shows a simple process model for the changes in an ice-water system as a function of the heat put into it. The model includes three processes, one (ice-warming) active when the ice's mass is nonzero and the system temperature is less than zero, another (ice-melting) when the ice's mass is nonzero and the tem-

perature is zero, and a third (water-warming) when all the ice has melted and the temperature is between 100 and zero. The first and third processes influence only the temperature, whereas the second process affects only the masses of ice and water. Given initial conditions (e.g., $ice\_mass = 10$, $water\_mass = 0$, and $temp = -10$) and heat measurements, this model can simulate changes over time.[1] Note that the model predicts the ice-warming process will be followed by ice-melting and then water-warming, but it does not state this explicitly, as would empirical laws. Thus, the process model explains the system's temporal behavior.

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 our field to address. We can state this task as:

- *Given:* Observations for a set of continuous variables as they vary over time;
- *Given:* Generic processes that specify causal relations among variables using conditional equations;
- *Given:* Optionally, constraints on the types of variables involved in each process and knowledge about these variables' types;
- *Find:* A specific process model that explains the observed data and 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 values for their parameters. Also, we have allowed for type constraints on process variables, which are available for many domains.

We intend this paper as 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 one prospective approach for process model induction and illustrate it with some initial results from population dynamics. Finally, we close by suggesting an agenda for future research on this important topic. We will not report extensive experimental results, leaving such studies for those who implement this agenda.

[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 1. A quantitative process model of mass and temperature change in an ice-water system.

model WaterPhaseChange
variables: $temp, heat, ice\_mass, water\_mass$
observables: $temp, heat, ice\_mass, water\_mass$
process ice-warming
  conditions: $ice\_mass > 0, temp < 0$
  equations: $d[temp, t] = heat/(0.00206 * ice\_mass)$
process ice-melting
  conditions: $ice\_mass > 0, temp == 0$
  equations: $d[ice\_mass, t] = -(18 * heat)/6.02$,
  $d[water\_mass, t] = (18 * heat)/6.02$
process water-warming
  conditions: $ice\_mass == 0, water\_mass > 0$,
  $temp >= 0, temp < 100$
  equations: $d[temp, t] = heat/(0.004184 * water\_mass)$

2. Challenges of 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 it from traditional induction problems and that pose research challenges. Some characteristics focus on aspects of the training data, whereas others involve constraints on the nature of acquired knowledge.

Process models are designed to characterize the behavior of dynamical systems that change over time, though they can also handle systems in equilibrium. The data produced by such systems differ from those that arise in most induction tasks in a variety of ways. First, these variables are primarily continuous, since they represent quantitative measurements of the system under study. Second, the observed values are not independently and identically distributed, since those observed at later time steps depend on those measured earlier. 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.

We have already noted that process models are explanatory in nature. The processes themselves are not observable, and multiple processes can interact to produce complex behavior. Moreover, process models can include theoretical variables that are also unobservable. These characteristics involve the knowledge acquired during learning, but they have implications for task difficulty as well. In particular, inducing process models is a plausible option only for domains that are complex enough to require such explanatory accounts. Fortunately, this challenge is offset by knowl-

edge 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 they 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 may develop accurate predictors but fail to make contact with explanatory concepts familiar to domain experts. Thus, widely used algorithms for inducing regression trees and multilayer neural networks do not, by themselves, seem sufficient for inductive process modeling, nor do other schemes for predicting continuous variables, including ones for time-series analysis. However, this does not mean they cannot 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 their construction. This paradigm also uses background knowledge to account for observations, but nearly all such research has dealt with classification or problem solving, rather than with predicting continuous variables.[2] Moreover, the standard formulation assumes the training data are supervised, whereas inductive process modeling deals primarily with observational data in which no variable is treated as special. Similar points hold for research on theory revision (e.g., Ourston & Mooney, 1990), which combines inductive learning with domain knowledge, often stated as Horn clause programs that contain theoretical terms. Also, this paradigm assumes that one has an explanatory model at the outset, rather than constructing it from available components such as generic processes. Thus, both approaches seem like poor matches, though we will return to theory revision when we discuss open research issues.

Another paradigm – 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. However, as usually practiced, inductive logic programming focuses on supervised learning for classification tasks. Moreover, the research emphasis has been on generating logical structures rather than numerical ones.[3] Thus, the framework holds some promise as an approach to inductive process modeling, but it requires some revisions and extensions to this end.

Because 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 processes 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 deterministic in nature.

[2] DeJong's (1994) work on explanation-based learning for motor control comes closer to our needs, but the paradigm as a whole seems ill suited.

[3] Garrett et al. (in press) have used this approach to infer metabolic pathways involving biochemical processes, but these are qualitative rather than quantitative models.

A final approach involves learning 'dynamic Bayesian networks' that 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 focused on discrete variables and, as with hidden Markov models, the probabilistic representation seems inappropriate for deterministic process models. The situation for dynamic Bayes nets is similar to 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.

4. Inductive Process Modeling

Although our primary aim here has been to characterize the problem of learning process models, our arguments will be more convincing if we can report some initial results on this task. In this section, we describe one approach that we have implemented and report its behavior on data from the domain of population dynamics. We will present this approach in the traditional order, first discussing a formalism for representing process models, then considering the performance element that uses them, and finally describing the induction method that constructs models from data.

4.1 Representing Processes and Models

We have already seen one example that involves a three-process model for an ice-water system. 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 specifies zero or more conditions under which it is active, along with one or more causal equations that characterize the influence one or more variables exert on another. Although the processes in a given model are unordered and can occur in parallel, we can organize them into a causal graph that equates the outputs of some processes with the inputs of others.

Table 2 presents a set of processes for population dynamics, which concerns changes in species' population levels over time (Murray, 1993). 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 P must have type *population* and that its equation's coefficient must be nonnegative, with 1 as its default value. The background knowledge also specifies that certain generic processes, such as saturation and nonsaturation, cannot be instantiated with the same variables. Note that processes in this domain include no conditions, so they are continuously active.

Table 2. Seven generic processes for population dynamics with constraints on their variables and parameters.

process exponential_growth
  variables: $P\{population\}$
  equations: $d[P,t] = [0,1,\infty] * P$
process logistic_growth
  variables: $P\{population\}$
  equations: $d[P,t] = [0,1,\infty] * P * (1 - P/[0,1,\infty])$
process exponential_decay
  variables: $P\{population\}$
  equations: $d[P,t] = -[0,1,\infty] * P$
process constant_inflow
  variables: $I\{inorganic\_nutrient\}$
  equations: $d[I,t] = [0,1,\infty]$
process consumption
  variables: $P1\{population\}$, $P2\{population\}$, $nutrient\_P2\{number\}$
  equations: $d[P1,t] = [0,1,\infty] * P1 * nutrient\_P2$, $d[P2,t] = -[0,1,\infty] * P1 * nutrient\_P2$
process no_saturation
  variables: $P\{number\}$, $nutrient\_P\{number\}$
  equations: $nutrient\_P = P$
process saturation
  variables: $P\{number\}$, $nutrient\_P\{number\}$
  equations: $nutrient\_P = P/(P + [0,1,\infty])$
mutually exclusive: {exponential_growth, logistic_growth}
mutually exclusive: {no_saturation, saturation}

The induction system is also given a set of observed variables, possibly with information about their types. For example, we might observe the dynamic behavior of an aquatic ecosystem that involves zooplankton, phytoplankton, and nitrogen. Here the first two variables are populations, whereas the third is an inorganic nutrient. The training data consist of measurements for all three variables as they change over time.

The result of learning is a process model like the one shown in Table 3, which includes six specific instances of the generic processes in Table 2. The first two processes state that phytoplankton grows in an unlimited (exponential) manner, whereas zooplankton growth is limited by the environment's capacity. The next process specifies that phytoplankton consumes the inorganic nutrient nitrogen, which increases its population and decreases the amount of nutrient. The fourth process posits that the consumption capacity of phytoplanton for nitrogen is unlimited. The last two processes specify that zooplankton's predation on phytoplankton has limited capacity (that it saturates). This

Table 3. A process model for an aquatic ecosystem.

model AquaticEcosystem
variables: $nitro, phyto, zoo, nutrient\_nitro, nutrient\_phyto$
observables: $nitro, phyto, zoo$
process phyto_exponential_growth
  equations: $d[phyto, t] = 0.1 * phyto$
process zoo_logistic_growth
  equations: $d[zoo, t] = 0.1 * zoo/(1 - zoo/1.5)$
process phyto_nitro_consumption
  equations: $d[nitro, t] = -1 * phyto * nutrient\_nitro$,
  $d[phyto, t] = 1 * phyto * nutrient\_nitro$
process phyto_nitro_no_saturation
  equations: $nutrient\_nitro = nitro$
process zoo_phyto_consumption
  equations: $d[phyto, t] = -1 * zoo * nutrient\_phyto$,
  $d[zoo, t] = 1 * zoo * nutrient\_phyto$
process zoo_phyto_saturation
  equations: $nutrient\_phyto = phyto/(phyto + 0.5)$

model incorporates two unobservable variables, each related to a population's consumption capacity, and assumes a closed ecosystem with no inflow.

4.2 Making Predictions with Process Models

Any induction system requires some performance element that can utilize knowledge once it has been learned. In this case, we require some interpreter that can use a quantitative process model to carry out forward simulation that generates a predicted time series for each observable variable. To this end, we have implemented a module that invokes established methods for solving first-order differential equations, which are available in public-domain software (e.g., Cohen & Hindmarsh, 1996), along with simple arithmetic operations for handling instantaneous equations. For this purpose, we must specify initial values for each observable variable 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 next step on the causal chain, and so on, until reaching the chain's final variables. The interpreter utilizes only active processes on each time step, that is, those whose conditions are met. When multiple active processes influence the same variable, the system makes the simplifying assumption that their effects are additive.

4.3 Constructing a Model from Components

Recall that our learning task involves constructing a process model from a known set of generic processes, which we assume comes from a domain expert, and time-series data about the quantitative variables one wants to explain. We have implemented an initial system, which we will call IPM (for Inductive Process Modeler) that carries out constrained search through the space of process models for one that accounts for these observations. The search mechanism operates in four successive stages.

The first step involves finding all ways to instantiate the known generic processes with specific variables. For each generic process, IPM simply checks every possible assignment of observable variables to generic variables mentioned in the process, retaining only assignments that satisfy the type constraints. For example, the variables for zooplankton (zoo) and phytoplankton ($phyto$) have type *population*, so they match the generic variable P in the process exponential_growth, whereas nitrogen has type *inorganic_nutrient*, so it matches I in the process constant_inflow. The result is a set of *instantiated* processes that specify particular variables but still lack parameter values. Because a model can refer to theoretical variables, this step also generates instantiated processes that incorporate one or more such terms.

In the second stage, IPM combines subsets of these instantiated processes into *generic models*, each of which specifies an explanatory structure, much like a proof tree. One constraint here is that the candidate models must consist of connected graphs, on the assumption that the data are produced by a single system. Another forbids mapping any specific variable onto more than one generic variable in a given process. We also specify the maximum number of processes that can connect any two variables and the maximum number of processes in a model. Within these boundaries, IPM carries out a backward-chaining search for all generic models, since this scheme is guaranteed to find all connected graphs. This search requires some unification, in that it must link unobservable variables that are input by one process to the output of others.

The third step focuses on inducing values for the parameter in each generic model. To this end, IPM calls on LAGRAMGE (Todorovski & Džeroski, 1997), which incorporates a declarative bias to constrain search for differential equation models. This program uses established methods for optimization search to fit the parameters in equations, but, since the details have appeared elsewhere, we will not recount them here. More important, because the declarative bias takes

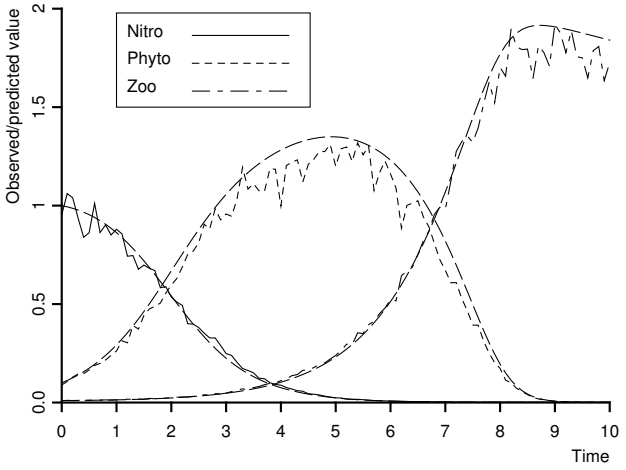

Figure 1. Observations (jagged lines) generated by the process model in Table 3, with noise added, and predictions (smooth lines) from the induced model in Table 4.

the form of a context-free grammar, IPM translates each generic model into such a grammar and passes it to LAGRAMGE. These grammars include one rewrite rule for each process-equation pair, with each nonterminal symbol occurring in only one rule. This does not take full advantage of LAGRAMGE's ability to handle more general grammars, but IPM's search through the space of generic models serves an analogous function.

For each generic model, LAGRAMGE returns a specific model with specific values for each parameter, along with a score for each such candidate. This score reflects the overall difference between the model's predictions and observations, stated as the sum of the squared errors over all observed variables. In the final step, IPM simply selects the specific model with the best score and halts upon returning it.

4.4 Learning a Population Dynamics Model

To demonstrate IPM's functionality at inducing process models, we decided to run it on synthetic data for a known system. For this purpose, we used the aquatic ecosystem model in Table 3 to generate observations for 100 time steps, using the initial values $nitrogen = 1.0$, $phyto = 0.01$, and $zoo = 0.01$ for the observable variables.

To make these data more realistic, we introduced noise by replacing each 'true' value x with $x \times (1 + r \times 0.05)$, where we sampled r from a Gaussian distribution with mean 0 and standard deviation 1. This produces noise relative in size to the actual value, giving the observations shown in Figure 1. We then ran the program on the noisy data, giving it type constraints and the generic processes in Table 2 as background knowledge.

Table 4. Ecosystem model induced by the IPM algorithm.

model AquaticEcosystem
variables: $nitro, phyto, zoo, nutrient\_phyto, nutrient\_nitro\_1, nutrient\_nitro\_2$
observables: $nitro, phyto, zoo$
process phyto_exponential_growth
  equations: $d[phyto, t] = 0.089 * phyto$
process zoo_logistic_growth
  equations: $d[zoo, t] = 0.013 * zoo/(1 - zoo/0.469)$
process phyto_nitro_consumption
  equations: $d[nitro, t] = -1.174 * phyto * nutrient\_nitro\_1$,
  $d[phyto, t] = 1.058 * phyto * nutrient\_nitro\_1$
process phyto_nitro_no_saturation
  equations: $nutrient\_nitro\_1 = nitro$
process zoo_phyto_consumption
  equations: $d[phyto, t] = -0.986 * zoo * nutrient\_phyto$,
  $d[zoo, t] = 1.089 * zoo * nutrient\_phyto$
process zoo_phyto_saturation
  equations: $nutrient\_phyto = phyto/(phyto + 0.487)$
process nitro_constant_inflow
  equations: $d[nitro, t] = 0.067$
process zoo_nitro_consumption
  equations: $d[nitro, t] = -0.470 * zoo * nutrient\_nitro\_2$,
  $d[zoo, t] = 1.089 * zoo * nutrient\_nitro\_2$
process zoo_nitro_saturation
  equations: $nutrient\_nitro\_2 = nitro/(nitro + 0.020)$

Table 4 presents the process model that resulted from this run, which has a form very similar to the ecosystem model that generated the data. The IPM algorithm selected this model structure from 2196 candidates that it considered during search. Some differences from the model in Table 3 involve parameters that appear in the shared processes, all of which have values close to the 'true' ones. But the induced model also includes three extra processes, shown at the bottom of Table 4, that do not occur in the original. One process states that nitrogen flows into the system at a constant rate. The other two claim that zooplankton consumes nitrogen with a limited capacity.

Fortunately, these extra processes have little effect on the model's overall behavior. Figure 1 shows that the predicted trajectories for the three variables are close to their observed values. In more quantitative terms, the root mean-squared error for the induced model on the training data is 0.026 for *nitro*, 0.085 for *phyto* and 0.067 for *zoo*. These compare favorably with the errors for the 'true' model on the same data, which are 0.024 for *nitro*, 0.045 for *phyto*, and 0.043 for *zoo*.

These results are encouraging, as they demonstrate that the IPM algorithm can induce a reasonable process model from noisy time-series data. However, its inclusion of unnecessary processes suggests the need

for a pruning method, along with experiments on other data sets, including ones from actual ecosystems, that use a standard division into training and test cases. Such studies may reveal other limitations and suggest improved algorithms for this important task.

5. 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 research in the area should address and consider some promising approaches that should be explored within this research agenda.

For example, the IPM algorithm assumes that the conditions on component processes are correct. Future methods should determine from training data the proper thresholds on conditions specified in generic processes or even learn which variables should occur in their conditions. We also need research that extends model representations in tractable ways. For instance, variables in scientific models are often associated with physical objects; encoding such objects and their roles explicitly could provide further constraints on acceptable models. We should even consider methods which can generate explanations that involve at least some processes with unknown forms. Overall, there remains considerable room for demonstrating new functionality in the induction of process models.

Another important issue concerns making robust algorithms for process model induction. Overfitting the training data can arise in nearly every learning task, and we need research on ways to guard against this tendency, especially as we develop algorithms that generate more complex process 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 boosting and bagging, though these would reduce the communicability of the resulting models. In addition, we should explore other defenses against overfitting specific to process models.

We also need research on ways to further direct the search for process models. Our IPM algorithm uses constraints on variable types to this end, but we should examine other ways to incorporate such knowledge, especially as we move to more difficult modeling tasks. One approach would draw on a taxonomy of process types to organize and limit further the search effort. Knowledge about the dimensional units of variables would also constrain model induction, as would the introduction of knowledge that sums of certain variables are conserved over time. We should also build on Bradley et al.'s (2001) use of qualitative patterns to focus on certain classes of equations. Future research should consider these and other methods for making the search for process models more tractable.

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.

We have focused here on process models that include numeric equations, but our research agenda should also explore techniques for inducing models composed of *qualitative* processes (Forbus, 1984). These take a similar form to quantitative processes, but use proportionalities to describe relations between variables. In this framework, the processes of exponential growth and logistic growth in Table 2 both map onto a single qualitative process which states that $d[P, t]$ is directly proportional to P. Such models are appropriate for domains like molecular biology, where scientists often state their knowledge in qualitative form. Moreover, qualitative models generally have fewer effective parameters than quantitative ones, making 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.

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. 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.

6. Concluding Remarks

In this paper, we proposed a new problem for 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 methods to address them, concluding that it demands research on new induction methods specialized to process modeling. We also presented an initial algorithm of this sort and demonstrated its functionality in a population dynamics domain, 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. They 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 methods that address this important new problem.

Acknowledgements

The research reported in this paper was supported in part by NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation, in part by Grant NCC 2-1220 from NASA Ames Research Center, and in part by EU Grant IST-2000-26469.

References

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

Cohen, S., & Hindmarsh, A. (1996). CVODE: A stiff/nonstiff ODE solver in C. *Computers in Physics*, *10*, 138–43.

DeJong, G. F. (1994). Learning to plan in continuous domains. *Artificial Intelligence*, *64*, 71–141.

Dietterich, T. G., (1990). Exploratory research in machine learning. *Machine Learning*, *5*, 5–10.

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

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

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

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

Murray, J. D. (1993). *Mathematical biology* (2nd ed.). Berlin: Springer.

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

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

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

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

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