Robust Induction of Process Models from Time-Series Data

Pat Langley LANGLEY@CSLI.STANFORD.EDU
Dileep George DIL@STANFORD.EDU
Stephen Bay SBAY@APRES.STANFORD.EDU
Computational Learning Laboratory, CSLI, Stanford University, Stanford, CA 94305 USA

Kazumi Saito SAITO@CSLAB.KECL.NTT.CO.JP
NTT Communication Science Laboratories, 2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan

Abstract

In this paper, we revisit the problem of inducing a process model from time-series data. We illustrate this task with a realistic ecosystem model, review an initial method for its induction, then identify three challenges that require extension of this method. These include dealing with unobservable variables, finding numeric conditions on processes, and preventing the creation of models that overfit the training data. We describe responses to these challenges and present experimental evidence that they have the desired effects. After this, we show that this extended approach to inductive process modeling can explain and predict time-series data from batteries on the International Space Station. In closing, we discuss related work and consider directions for future research.

1. Introduction and Motivation

Research on computational scientific discovery studies methods that generate knowledge in established scientific notations (e.g., Langley, 2000). Like other branches of machine learning, most work in this area has been content with inducing shallow, 'descriptive' models, provided they are accurate on new test cases. However, mature scientific disciplines strive for models that *explain* behavior in terms of unobserved variables, entities, or processes. Some researchers have developed methods that construct explanatory scientific models, but they have typically focused on qualitative formalisms (e.g., Valdés-Pérez, 1995) or on algebraic relations (e.g., Washio et al., 2000).

Yet many fields of science and engineering develop quantitative models of dynamic systems that change over time. Such models are often cast as sets of interacting differential equations that characterize how one or more variables produce changes in other variables. Research on *system identification* (Astrom & Eykhoff, 1971) develops methods that infer the quantitative parameters for such models, but these assume the model structure is known. A few researchers have tackled the task of determining both the model's structure and its parameters (e.g., Bradley et al., 1999; Todorovski & Džeroski, 1997), but their learned models retain a generally descriptive flavor, in that they do not refer to unobserved entities.

In a recent paper (Langley, Sanchez, Todorovski, & Džeroski, 2002), we defined the task of *inductive process modeling*. This paradigm revolves around a novel class of quantitative process models and their construction from time-series data. Process models are explanatory in that they account for observations in terms of interactions among unobserved processes, which in turn make contact with generic background knowledge. Inductive process modeling constitutes a new paradigm for computational scientific discovery that seems especially appropriate for integrative fields like Earth science, but it is relevant to any discipline that utilizes quantitative models of dynamic systems.

We begin by reviewing the notion of process models and the reasons that existing learning methods are not sufficient to induce them from time-series data. We also reexamine our earlier IPM algorithm and some challenges that arise when it is confronted with data from a realistic ecosystem modeling task. In response, we introduce a number of extensions to this algorithm, along with experimental evidence that they handle these challenges. After this, we demonstrate the extended method's behavior on telemetry data from batteries on the International Space Station. We conclude by discussing research on related problems and directions for future work on process model induction.

Proceedings of the Twentieth International Conference on Machine Learning, Washington, DC, 2003.

Table 1. A process model for a simple aquatic ecosystem.

```
model AquaticEcosystem;
variables phyto, zoo, nitro, residue;
observables phyto, nitro;
process phyto_exponential_decay;
  equations d[phyto, t, 1] = −0.307 * phyto;
            d[residue, t, 1] = 0.307 * phyto;
process zoo_exponential_decay;
  equations d[zoo, t, 1] = −0.251 * zoo;
            d[residue, t, 1] = 0.251 * zoo;
process zoo_phyto_predation;
  equations d[zoo, t, 1] = 0.615 * 0.495 * zoo;
            d[residue, t, 1] = 0.385 * 0.495 * zoo;
            d[phyto, t, 1] = −0.495 * zoo;
process nitro_uptake;
  conditions nitro > 0;
  equations d[phyto, t, 1] = 0.411 * phyto;
            d[nitro, t, 1] = −0.098 * 0.411 * phyto;
process nitro_remineralization;
  equations d[nitro, t, 1] = 0.005 * residue;
            d[residue, t, 1] = −0.005 * residue;
```

2. Induction of Process Models

Explanations in science and engineering are often stated in terms of generic processes from the domain in question. We will focus here on a class of processes that describe one or more quantitative causal relations among variables. In this framework, a process states these relations in terms of differential equations (for a process that involves change over time) or algebraic equations (if it involves instantaneous effects). A process may also include conditions, stated as threshold tests on variables, that describe when it is active. A *process model* consists of a set of processes that link observable variables with each other, possibly through unobserved theoretical terms.

2.1. Quantitative Process Models

Table 1 shows a process model for an aquatic ecosystem, based on our discussions with an Earth scientist (K. Arrigo, personal communication, 2003), that concerns changes in species' populations, nitrates, and residue over time. The table shows five processes, the top two stating that phytoplankton and zooplankton die at an exponential rate. Another process indicates that zooplankton preys on phytoplankton, increasing and decreasing their populations, respectively, but also that some killed phytoplankton is not absorbed but produces residue. A fourth process posits that phytoplankton uptakes nitrogen, which increases its population and decreases the nutrient. The final process specifies that residue returns nitrogen to the environment at a certain rate. The model also states that only phytoplankton and nitrogen are observable, and that the uptake process requires the presence of nitrogen.

Given a quantitative process model of this sort and initial values, we can simulate the model's behavior to generate a predicted time series for each variable. Because models may involve chains of such equations, the simulator must solve, for each process P, the associated algebraic and differential equations for the current time step to determine new values for P's output variables, use these to solve 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. Moreover, it must utilize only active processes (those with satisfied conditions) on each time step.[1]

Process models of this sort provide an explanation of such time-series data, not only because they offer a causal account and may include unobserved variables, but because they refer to processes and associated equations that are familiar to domain specialists. Table 2 presents some generic processes relevant to aquatic ecosystems that might serve as such background knowledge. Unlike *specific* processes, they do not commit to particular variables or parameter values, but they can indicate constraints on them. For example, the process for exponential decay states that its variable S must have type *species* and that its equations' shared coefficient, α, must fall between 0 and 1.

2.2. The Task of Inductive Process Modeling

We maintain that scientists often construct specific models (cast in their own notations) like that in Table 1 from knowledge about generic processes like those in Table 2. We can state this task of *inductive process modeling* 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:* The names and types of observable and unobservable variables to consider for incorporation;
- *Find:* A specific process model that explains the observed data and predicts future data accurately.

Developing computational techniques that handle this problem in a robust manner would constitute a significant advance in machine learning and provide many benefits to the natural sciences and engineering.

[1] Here we assume that, when multiple active processes influence the same variable, their effects are additive.

Table 2. Five generic processes for aquatic ecosystems with constraints on their variables and parameters.

```
generic process exponential_decay;
  variables $S\{species\}, D\{detritus\}$;
  parameters $\alpha$ $[0, 1]$;
  equations $d[S, t, 1] = -1 * \alpha * S$;
            $d[D, t, 1] = \alpha * S$;
generic process predation;
  variables $S1\{species\}, S2\{species\}, D\{detritus\}$;
  parameters $\rho$ $[0, 1], \gamma$ $[0, 1]$;
  equations $d[S1, t, 1] = \gamma * \rho * S1$;
            $d[D, t, 1] = (1 - \gamma) * \rho * S1$;
            $d[S2, t, 1] = -1 * \rho * S1$;
generic process nutrient_uptake;
  variables $S\{species\}, N\{nutrient\}$;
  parameters $\tau$ $[0, \infty], \beta$ $[0, 1], \mu$ $[0, 1]$;
  conditions $N > \tau$;
  equations $d[S, t, 1] = \mu * S$;
            $d[N, t, 1] = -1 * \beta * \mu * S$;
generic process remineralization;
  variables $N\{nutrient\}, D\{detritus\}$;
  parameters $\pi$ $[0, 1]$;
  equations $d[N, t, 1] = \pi * D$;
            $d[D, t, 1] = -1 * \pi * D$;
generic process constant_inflow;
  variables $N\{nutrient\}$;
  parameters $\nu$ $[0, 1]$;
  equations $d[N, t, 1] = \nu$;
```

As noted in our earlier paper, this task differs from those typically studied in machine learning. In particular, process models are designed to characterize the behavior of dynamic systems with continuous variables that change over time; thus, samples are not independently and identically distributed, since those observed later depend on those measured earlier. Moreover, process models are explanatory in nature, in that processes themselves are not observable, multiple processes can interact to produce complex behavior, and the models can include theoretical variables that are also unobservable. However, these complicating factors are partially offset by the assumption that the dynamic systems are deterministic (although the observations themselves may contain noise), since scientists often operate under this assumption.

We have also argued that existing methods for machine learning and knowledge discovery do not solve the task of inductive process modeling. For instance, although methods for equation discovery generate knowledge in formalisms familiar to scientists, they typically produce shallow descriptive summaries rather than explanations in terms of underlying processes. Standard methods for supervised learning, including algorithms for inducing regression trees, neural networks, and ARIMA models, may produce accurate predictors but make no contact with explanatory concepts familiar to scientists. Techniques for explanation-based learning use background knowledge to account for observations, but they typically assume supervised training cases and focus on classification or problem solving.

Another paradigm, Hidden Markov models, describes systems that change over time and include unobservable states, but typically only one state can be active at a time, and they require explicit links that specify state transitions. Dynamic Bayesian networks are similar in spirit to sets of differential equations, but they do not organize these equations into processes, and they typically assume discrete variables. Moreover, like hidden Markov models, they assume discrete time steps and make probabilistic assumptions that are unnecessary for many scientific models. The framework of inductive logic programming comes closest to our approach, in that it takes advantage of background knowledge and its learned rules can contain explanatory theoretical symbols. However, its focus on classification tasks and logical structures would require some extension to handle our new class of models.

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

2.3. Review of Previous Results

In our earlier paper, we described IPM, an initial algorithm that constructs a process model from known generic processes and observed time-series data. The method carries out constrained search through the space of process models, operating in four stages:

1. Find all ways to instantiate the known generic processes with specific variables. This involves finding, for each generic process, every possible assignment of variables to generic variables mentioned in the process that satisfies the type constraints. This gives a set of *instantiated processes* that specify particular variables but lack parameter values. Because a model can refer to unobserved variables, IPM also generates instantiated processes that include one or more such terms.

2. Combine subsets of these instantiated processes into *generic models*, each of which specifies an explanatory structure, much like a proof tree. These models cannot map any specific variable onto more than one generic variable in a process. IPM also includes a parameter that specifies the maximum number of processes allowed in a model.

3. Transform each generic model into a specific model by determining its parameters. To this end, IPM invokes a gradient descent search through the space of parameter values that attempts to minimize the mean squared error over all observed variables.
4. Select the specific model (the generic model with associated parameter values) that produces the lowest mean squared error on the training data.

We did not intend this initial algorithm to be especially efficient or robust, but we hoped it would show that inductive process modeling was possible in principle.

We demonstrated IPM's ability on synthetic data for a known system, similar to that in Table 1, involving four variables and six processes. To make these data more realistic, we added five percent noise to each variable. The induced model's structure was very similar to the target structure, its parameters were close to the 'true' ones, and it reproduced the observed behavior very well. These encouraging results suggested it was reasonable to explore process model induction further.

3. Robust Induction of Process Models

Our previous paper defined the task of inductive process modeling and presented an initial method, IPM, that addresses this problem. However, we also identified a number of challenges that required additional research. In this section, we describe three extensions to IPM that respond directly to these problems, along with an application to a real-world induction task.

We should also note two other changes in our current implementation. First, IPM now requires the user to specify explicitly the unobservable variables that can appear in the induced model, which further constrains the space of model structures. Second, the system now uses a gradient-descent method developed by Saito and Nakano (1997) to carry out a more efficient, second-order search through the parameter space.

3.1. Dealing with Unobservable Variables

Note that the target model in Table 1 includes four variables, but that only two of these – *phyto* and *nitro* – are observable. The remainder – *zoo* and *residue* – correspond to theoretical terms that are involved in the system, but for which measurements are not available. Our initial paper reported the induction of a model with hidden variables, but these did not appear on the left side of differential equations. The problem this raises is that, to evaluate a process model on training data, we must have initial values for each variable that appears in such a role. But we do not know the initial values for *zoo* and *residue*, since they are not observed.

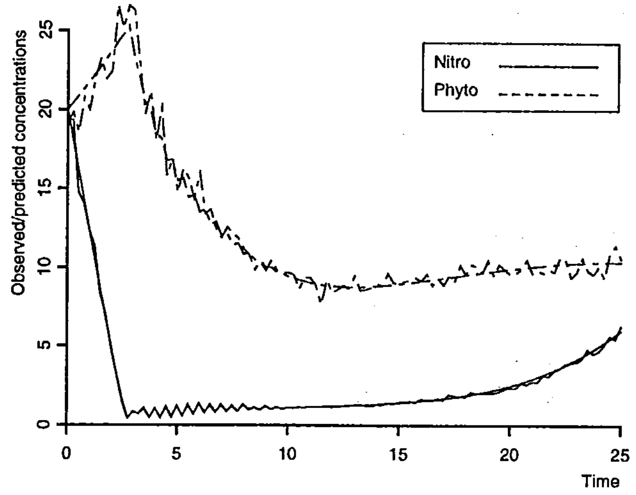

Figure 1. Observations (jagged lines) generated by the process model in Table 1, with noise added, and predictions (smooth lines) from the induced model. Concentrations for *nitro* are multiplied by ten to display curves on same scale.

Our response is to treat the initial values of unobserved variables as part of the induced model. We already use a parameter optimization method to determine the values for coefficients in models' equations, and we can utilize the same method to infer initial values. For a candidate with the same model structure as in Table 1, the need to select starting values for *zoo* and *residue* adds two parameters to the optimization search.

To determine whether this approach works, we used the target model to generate 20 sets of observations for 100 time steps, using different initial values for all four variables on each set. To make these data more realistic, we added noise by replacing each 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. Figure 1 presents the time series that results from one of these simulations. We then ran the parameter optimization method on these training data, given the same model structure as shown in the table.

The results suggested that we can indeed use parameter optimization to handle model structures with unobservable variables. In each run, this method found values that were close to both the target coefficients (the same across all runs) and the initial values (which differed across runs). The mean difference between the target and inferred values was 0.0064 ± 0.002 for *zoo*, which ranged from 0.5 to 1, and 2.116 ± 0.650 for *residue*, which ranged from 0.1 to 10. The smooth curves in Figure 1 show the predicted values for one such model, which fit the observed values very well. On average, the mean squared error was 46.76 ± 2.69 for *phyto* and 0.0859 ± 0.0113 for *nitro*. This contrasted with 45.82 ± 2.64 for *phyto* and 0.0850 ± 0.0112 for *nitro* on runs when the initial values for *zoo* and *residue* were known to the system.

3.2. Finding Conditions on Processes

Recall that our definition of a quantitative process model lets each process incorporate one or more conditions, stated as thresholds on variables. The model in Table 1 includes one such condition, which specifies that, for *nitro_uptake* to be active, the variable *nitro* must have a value greater than zero. The IPM algorithm reported in our earlier paper could not deal with such conditions, but we noted the need to extend the method to induce models containing them.

For this problem, we assume that the generic process specifies the basic form of the condition. For instance, Table 2 states that the generic process *nutrient_uptake* has one condition that involves a variable of type *nutrient*. Moreover, this variable must be greater than some constant τ, which must itself be more than zero. Based on these assumptions, we can also treat induction of conditions for a specific process, like *nitro_uptake*, as a problem in parameter optimization. We implemented this idea by recasting each condition as a sigmoid function (with value 0 on one side of the threshold and 1 on the other), then including the function as a multiplicative term in the process' equations.

To evaluate this approach to condition finding, we again generated 20 sets of time-series data for the two observed variables, each involving 100 time steps. However, in this study we held constant the initial values for all four variables and varied instead the *nitro* threshold used to generate synthetic data, which again contained five percent Gaussian noise. For each training set, we ran the parameter optimization method, given the same model structure as shown in Table 1.

As before, the results were encouraging. In each run, the system found a threshold for *nitro* that was close to the value used to generate the data, with a mean difference of 0.0053 ± 0.0009 between the inferred threshold and the target, which ranged from 0.1 to 1. The average root mean squared error for these models was 33.84 ± 0.007 on *phyto* and 0.176 ± 0.008 on *nitro*, as compared with 34.13 ± 2.78 on *phyto* and 0.170 ± 0.033 on *nitro* when the condition was known at the outset.

3.3. Reducing Overfitting of Process Models

Both extensions described above involve determining parameters for a given model structure, but a key advance over traditional system identification is that inductive process modeling searches the space of such structures. However, in our earlier paper, we noted the IPM algorithm tended to construct models with extra processes. These reduced error on the training set only slightly, but they would probably increase error on new test cases. This problem is reminiscent of early results with decision-tree induction, which overfit training data by creating overly deep trees.

One countermeasure to overfitting in decision trees, which we have adapted to inductive process modeling, attempts to minimize the complexity of the model plus the complexity of the data not explained by that model. We can encode this description length as

$$M_d = (M_v + M_c) \cdot log(n) + n \cdot log(M_e) ,$$

where M_v is the number of variables, M_c is the number of parameters (including ones for initial values and conditions), n is the number of training cases, and $log(M_e)$ is the number of bits needed to encode the mean squared error.[2] This metric is a variant of Schwartz's (1978) Bayesian Information Criterion.

Rather than selecting the candidate model with the lowest error, as did the original IPM algorithm, our new method selects the model with the lowest description length M_d. To demonstrate that this new metric has the desired effect, we ran both systems on five synthetic data sets generated from the ecosystem model in Table 1. For each one, we reused a set of initial values from Section 3.1 to produce 100 observations for *phyto* and *nitro*, then added five percent noise. We let both versions of IPM consider models that included up to six processes, with the only difference being the evaluation function used to rank models. Starting from the generic processes shown in Table 2, this led to 256 distinct model structures, each of which was transformed into a specific model by fitting its parameter values.

As expected, the description length metric fared better on these induction tasks than did the mean squared error metric. The M_d criterion ranked the correct model structure as best on all five data sets, whereas the original scheme selected an incorrect complex model in three out of five runs. These included the five processes from the target model in Table 1 that generated the time series, but they also contained one additional process. In general, both criteria produced models that matched the observed trajectories well, but the candidates selected using the new criterion did this without unnecessary processes. These results also suggest that IPM can induce the correct model structure even when it must infer initial values for unobserved variables.

We expected to observe an even greater difference between the two versions of IPM when we increased

[2] We chose not to include terms for the number of equations or processes, although this would also be reasonable.

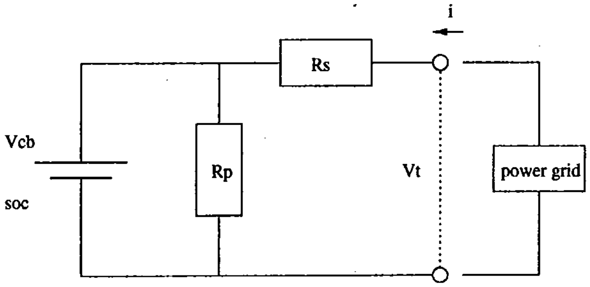

Figure 2. A simple battery and associated variables.

the noise to ten percent. However, on these training sets, the description length metric selected the correct model structure on only two runs out of five, whereas the error metric was right three times. The new system's errors involved not overly complex models, but rather overly simple ones that omitted processes for which it did not think there was sufficient evidence. These results suggest that there remains some room to improve our formulation of description length in the context of inductive process modeling.

3.4. Inducing a Dynamic Battery Model

The experiments with synthetic data reported above suggest that our approach to process model induction is promising, but results on a real data set would strengthen our claims. Lacking sufficient time-series data on aquatic ecosystems, we turned to another domain for which data were available – the behavior of batteries on the International Space Station – and which we have described elsewhere (Bay et al., 2002). Batteries are not well understood and modeling their behavior remains a challenging problem.

Figure 2 depicts a simple battery cast as an equivalent electric circuit. Here Vcb stands for the voltage of an ideal cell (the voltage source), Rs is the internal resistance to current flow associated with a resistor, Rp is the resistance to self discharge (degradation), i is the current flowing into or out of battery, Vt is the voltage at the battery's terminals, and *soc* is the total electric charge stored in the battery. Other prospective variables not shown in the figure are the battery's temperature and pressure.

The International Space Station has three battery units, each of which contains two sets of 36 nickel-hydrogen cells. We have telemetry data for these cells over 24 hours, with samples about every ten seconds. However, since only some cells have sensors, we averaged readings from 36 cells to produce a single time series for the current i and the voltage Vt at the terminals. Even with averaging, these data are low quality, with many missing values and occasional spikes. We used linear interpolation to infer missing values, but we did not attempt to remove the spikes.

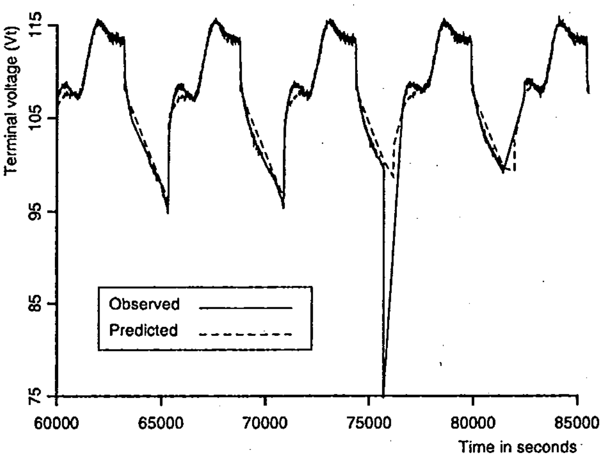

Figure 3. Telemetry test data for the terminal voltage Vt.

We provided IPM with generic processes for how Vcb and Rs affect Vt when the battery is charging ($i > 0$) and discharging ($i < 0$), as well as how i and Vcb produce changes in *soc*. In addition, we told the system about three generic processes with algebraic equations that involved a constant parameter, a linear relation, and a quadratic function. Type constraints ensured that only Vcb or Rs could occur on the left sides of these equations and that only *soc* and *temperature* could occur on the right sides. We based these processes on available knowledge about batteries, such as that *soc* affects Vcb in some manner (Chan & Sutanto, 2000), and that *soc* influences Rs, since it becomes very hard to charge a battery when *soc* is high and very hard to extract charge when it is low.

We divided the Space Station data into a training set of 6,000 instances (not shown) and a test set of 2,640 cases (shown in Figure 3), then ran the extended IPM algorithm on the former. We gave it the six generic processes just described and let it consider models with up to eight processes, with i, Vt, and *temperature* observed and *soc* unobserved. The system generated 256 model structures and fit parameters to each candidate. Table 3 shows the best model according to the M_d metric, which has six processes. These include charging, discharging, and charge transfer, which are required to predict Vt, but also a quadratic influence of *soc* on the voltage Vcb, a linear influence of *temperature* on Vcb, and a linear influence of *soc* on Rs. Nine other models had lower error on the training set, but additional processes gave them a higher description length.

Because IPM induced this model from real telemetry data, we cannot know whether its structure or parameters are correct. However, we can measure its pre-

Table 3. Induced process model for Space Station batteries.

```
model Battery;
variables Rs, Vcb, soc, Vt, i, temperature;
observables soc, Vt, i, temperature;
process voltage_charge;
  conditions i ≥ 0;
  equations Vt = Vcb + 6.105 * Rs * i;
process voltage_discharge;
  conditions i < 0;
  equations Vt = Vcb * 1.0/(Rs + 1.0);
process charge_transfer;
  equations d[soc, t, 1] = i − Vcb/179.38;
process quadratic_influence_Vcb_soc;
  equations Vcb = 41.32 * soc * soc;
process linear_influence_Vcb_temp;
  equations Vcb = 0.2592 * temperature;
process linear_influence_Rs_soc;
  equations Rs = 0.03894 * soc;
```

dictive ability on the test set we reserved for this purpose. Figure 3 shows that the model's predictions for Vt shadow the observations very closely, except for one brief anomalous period. The mean squared test error for Vt is 7.04, which is only one third the error produced by a model that has constant values for Vcb and Rs. Moreover, it is nearly identical to the error (6.99) reported in our earlier paper (Bay et al., 2002), which used LAGRAMGE (Todorovski & Džeroski, 1997) to find a simpler model.[3] However, the current model is more plausible in some ways; for example, it incorporates the notion that Rs increases with soc, which the earlier one did not.

4. Related Work and Future Research

Despite its novelty, our approach to inductive process modeling has many intellectual precursors. Although early work on computational scientific discovery (e.g., Langley, 1981) dealt with induction of shallow empirical laws, it cast knowledge in established scientific formalisms and viewed the discovery problem in terms of search. However, our approach has more in common with research by Todorovski and Džeroski (1997), Bradley et al. (1999), and Koza et al. (2001), who combine ideas from artificial intelligence and system identification to induce the structure and parameters for differential equation models. Our work extends theirs by focusing on processes, which play a central role in many sciences and provide a useful framework for encoding domain knowledge that constrains search.

Our framework also builds on early research in qualitative physics (e.g., Forbus, 1984), which was concerned with simulation of behavior over time, but emphasized qualitative reasoning about possible events rather than deterministic simulation of quantitative models. There are especially close connections with Farquhar's (1994) QPC system, which constructs models from a set of qualitative process fragments. There has also been research on induction of qualitative models from time-series data (e.g., Suc & Bratko, 2002), but this has not used processes to group causal influences or to encode domain knowledge for guiding model construction.

Finally, although we have distinguished our approach from those taken in mainstream work on machine learning, it retains many common ideas. Our view of model construction as data-guided search through a space of hypotheses follows an old tradition, and our efforts to avoid overfitting incorporate ideas like minimizing description length, which has long been used in other paradigms. However, our framework applies these ideas in novel ways to support the induction of quantitative process models from time-series data.

We hope to incorporate other ideas from traditional machine learning in our future research. For instance, another response to overfitting would prune portions of the process model after construction, much as established methods prune induced decision trees. Ensemble methods can also reduce variance, but we cannot apply them directly if we want our process models to remain comprehensible. However, we can adapt bootstrap sampling (Efron & Tibshirani, 1993) to learn distinct process models from different training sets, each produced from the original data by sampling with replacement, then retain only those processes that occur in more than a specified fraction of the learned models.

We should also develop more flexible methods for inducing conditions on processes. One approach would utilize a greedy algorithm similar to that used to find conditions in rule induction. For each process in a model, this method would consider adding a single condition that involves each possible variable, use parametric search to determine the best threshold, and select the condition that gives the best score. The method would then continue the search, first considering more conditions on the current process and then determining conditions for other processes.

A third important direction concerns increasing search efficiency. Our current implementation of IPM considers all model structures up to a specified size, but clearly we should also examine heuristic methods that search the model space more selectively. One approach would start model construction from 'output' variables

[3] We should also mention that Bay et al.'s method considered 6859 model structures, because they included the variable *pressure*, along with sigmoid functions and third-degree polynomials, in the search space.

and work backwards, first finding processes that explain their variations, then turning to inputs of those processes, finding others that explain their behavior, and continuing until forming a complete model. However, over 99 percent of our run times are due to the gradient descent method for parameter fitting, so we should also look into ways to make this more efficient.

5. Concluding Remarks

In this paper, we examined the problem of inducing quantitative process models from time-series data. This new paradigm holds special relevance for fields that must construct models of complex systems with interacting elements, but our initial work in the area made some important simplifying assumptions. After reviewing the IPM algorithm, we presented three extensions that let the method determine the initial values for unobserved variables, infer the thresholds on numeric conditions, and minimize overfitting by taking into account model complexity.

In each case, we used studies on synthetic data to demonstrate the extended method's robust behavior. These included tests of a description length metric's ability to determine the correct model structure in the presence of noisy data, which produced mixed results. Finally, we showed that our method produces accurate and plausible models when applied to telemetry data from batteries on the International Space Station. Despite this encouraging progress, we also identified some areas that require additional development, which we hope to explore in future papers on the promising paradigm of inductive process modeling.

Acknowledgements

This research was supported in part by NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation, and in part by Grant NCC 2-1220 from NASA Ames Research Center.

References

Astrom, K. J., & Eykhoff, P. (1971). System identification - A survey. *Automatica*, *7*, 123–167.

Bay, S. D., Shapiro, D. G., & Langley, P. (2002). Revising engineering models: Combining computational discovery with knowledge. *Proceedings of the Thirteenth European Conference on Machine Learning* (pp. 10–22). Helsinki, Finland.

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

Chan, H., & Sutanto, D. (2000). A new battery model for use with battery energy storage systems and electric vehicle power systems. *Proceedings of the IEEE Power Engineering Society Winter Conference.*

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

Farquhar, A. (1994). A qualitative physics compiler. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 1168–1174). AAAI Press.

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 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.

Langley, P. (2000). The computational support of scientific discovery. *International Journal of Human-Computer Studies*, *53*, 393–410.

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.

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

Schwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Suc, D., & Bratko, I. (2002). Qualitative reverse engineering. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 610–617). Sydney: Morgan Kaufmann.

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.

Valdés-Pérez, R. E. (1995). Machine discovery in chemistry: New results. *Artificial Intelligence*, *74*, 191–201.

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.