

User Guided Model Revision

Gordon Christie, Derek Sleeman and George M. Coghill

Department of Computing Science,
University of Aberdeen,
Aberdeen, AB24 3UE. UK.

{gchristie, dsleeman, gcoghill}@csd.abdn.ac.uk

Abstract. Model discovery systems are able to use collected domain data to either revise existing models (Prometheus), or create new models (Lagrange). However, these systems do not provide the ability to explicitly obtain and use feedback during model searches. Creating a system able to use this feedback explicitly would allow a user who was not entirely certain of the final form of their model to use small steps to find a (progressively) more acceptable model. This paper describes a possible method for doing this.

1. Existing Systems

1.1. Prometheus/RPM

Prometheus [1] is a visualisation tool for RPM [2], a model-revision system. RPM takes an existing model representing a "best guess" for the domain, a list of generic processes that can be used to extend the system or replace processes in the system, and a data file of example behaviour. RPM then attempts a refinement process to make the model output closer to the data provided.

Prometheus allows the user to create and edit models in a graphical environment (GUI), and allows the user to specify which processes can be altered/deleted in the course of model revision. Prometheus also provides a separate explicit simulation facility, which allows the user to generate a sample data set from a model.

1.2. Lagrange

Lagrange [3] is a system for equation discovery which uses "declarative bias" [4] to describe possible forms for equations to take. Lagrange uses a context-free grammar (CFG) to describe this bias. This context-free grammar allows the user to describe the form of equations the system searches for, reducing the search space dramatically. The grammar is either written by the user, or can be based upon existing grammars provided with Lagrange. The system uses a set of data files and a context-free grammar, and generates a set of possible equations, sorted by the smallest error between the results of the equations and the provided data files.

2. Systems Evaluation

2.1. Feature Comparison

The major difference between the two systems is that Prometheus/RPM is a model revision system, using an existing model as a starting point to find variations, whereas Lagrange is a model discovery system.

Prometheus provides the ability to create and edit RPM model files using a graphical environment; to create and edit models in RPM, the user must edit the text files directly. Lagrange, a model discovery system, works with context-free grammars.

2.2. Limitations

The systems evaluated are mostly concerned with relatively few iterations of search, where the system is provided with a set of data and constraints of some kind (an existing model or context free grammar), and is asked to produce a set of candidate models.

It would be desirable to explicitly capture user feedback (i.e. which parts of this model are “good” and “bad”? What rules can we derive from this?) and use this information as part of an iterative search process, using small steps to refine the model. This would reduce the computational complexity of the task, as well as allowing a user to guide the process more explicitly.

Prometheus comes closest to achieving this; it is possible to take models from previous iterations and use them as a starting point for further discovery. However, this does not allow one to specify which process one wants to look for in future searches, but only lets one start from a complete model. Lagrange would have difficulties in expressing this kind of user feedback. Context-free grammars, by their nature, are not expressive enough to allow the user to specify parameter variables in equations.

3. Proposed System

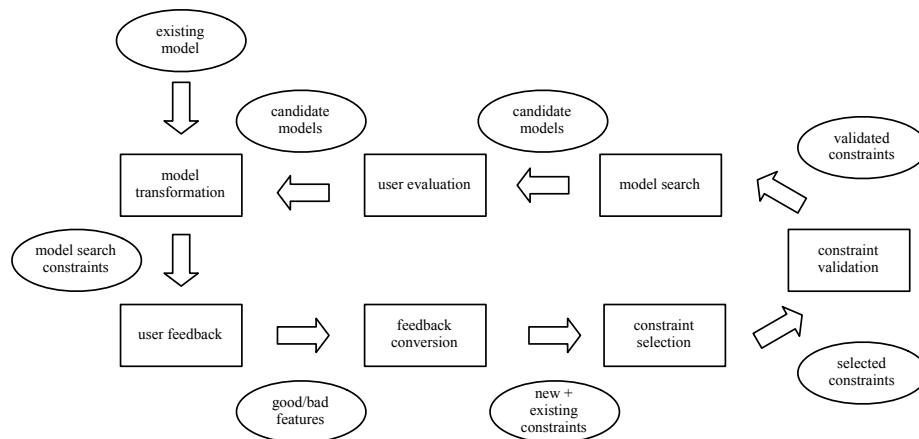


Fig. 1. Proposed System Architecture

The proposed system will start with either a partially specified model, or an existing fully specified model. The system will use this in conjunction with a target data set to produce a list of possible revised models, ordering these by the closeness of fit to the target data set. The user will then rate the components of these revised models individually as being either positive or negative. The system will take this information and form constraints that will be used to produce another list of revised models; the

user then evaluates these models. This process continues until the user finds a model that is acceptably close to the sample data provided, or abandons the process.

3.1. Model Transformation

The modelling system should be able to reuse an existing model, or allow a user to discover models using a partial model specification. One way to reuse existing models would be to take the existing model and translate it into a context-free grammar. In specifying a partial model, the user would specify parts of the model, together with a list of allowed components for building the remainder of the model. This list of possible components would take the form of a library of template processes, similar to the generic processes found in Prometheus/RPM.

3.2. User Feedback

In this stage, the user would be presented with a list of models in a similar way to Prometheus. The user would be asked to give feedback on these models. Some example questions are shown below:

- Process y has invalid parameters, but is of the correct form – can a better range for parameters be specified?
- Process y is using the wrong variables, but is of the correct form – which variables might be better?
- Process y is correct, lock it temporarily to prevent revision in future iterations

The user will have the option of using this information to constrain future searches; in this way, a model will be refined by many small modifications.

3.3. Feedback Conversion

This stage would take the reviewed good and bad processes and add them to the constraints used for the previous search iteration. New constraints would initially be given priority over the older constraints, as they should be more relevant.

3.4. Constraint Selection

The user would then be presented with a list of search strategies based upon feedback given in the previous stage. Instead of searching for all alternatives, it would be possible to select a subset and use those initially for searching. The program would keep a history of constraints generated by feedback, and a full history of previously found models, allowing the user to try alternatives without losing the ability to go back to a previous state.

3.5. Constraint Validation

Once the user has selected which parts of the model are good or bad, a conflict check is required, as contradictions may occur. Examples include:

- Have two different ranges for a particular constant been provided? Can the range be expanded or altered?

- One model says that a process is good and another says that this process is bad. Which is correct? Can both process types apply in different cases, in which case, how are conditions specified?

3.6. Model Search

This functionality could be provided by an existing system like Lagrange, which would be constrained by an appropriate context-free grammar.

3.7. User Evaluation

The user would be presented with a sorted list of models from the model search component, ordered by their fit to the example data. The user would then analyse the models presented, and determine if any of the returned models was acceptable. If not, the user would be invited to give feedback on the returned models, and use that information as input to a further iteration of the system.

4. Future Work

The system in question is currently being designed. Several issues need to be resolved:

What are valid “good/bad” points from the several candidate models? Which of these points are relevant to future searches?

How are good and bad choices obtained from the user? How are alternative candidate models presented to a user without overwhelming them and losing possibly valuable feedback? How are constraints automatically created based upon these good and bad choices? How are constraints created from existing models?

What form of model representation should be used? Should a model be directly revised, or should there be an interim stage of representation similar to a context-free grammar?

5. References

1. BRIDEWELL, W., SANCHEZ, J.N. and LANGLEY, P., 2004. An Interactive Environment for the Modelling and Discovery of Scientific Knowledge.
2. ASGHARBEGYI, N., BAY, S., LANGLEY, P. and ARRIGO, K., (in press). Inductive revision of quantitative process models. *Ecological Modelling*.
3. TODOROVSKI, L. and DZEROSKI, S., 1997. Declarative Bias in Equation Discovery, Proceedings of Fourteenth International Conference on Machine Learning, 1997, Morgan Kaufmann pp376-384.
4. NEDELLEC, C., ROUVEIROL, C., ADE, H., BERGADANO, F., & TAUSEND, B., 1996. Declarative Bias in ILP. In L.D. Raedt (Ed.), *Advances in inductive logic programming*, 82-103. Amsterdam, The Netherlands: IOS Press.