Neural-guided equation discovery

Jannis Brugger [1*], David Richter [1], Mattia Cerrato [2], Mira Mezini [1], Stefan Kramer [2]

*Correspondence: jannis.brugger@tu-darmstadt.de

[1] TU Darmstadt, Department of Computer Science, Darmstadt, 64293, Germany
[2] Johannes Gutenberg University Mainz, Faculty 08: Physics, Mathematics and Computer Science, Mainz, 55128, Germany

We address the task of discovering symbolic equations that describe the dynamics of a system of interest. An example of such a system of interest could be the experimental data from the free-falling of a body. If the distance fallen is denoted by y and the time elapsed by t, our equation discoverer should then find the equation $y = \frac{g}{2} \cdot t^2$ using the experimental data. The search for the equations is formulated as a tree search as in a game, where the possible actions are given by a *context-free grammar* and the goal is to find the equation which fits the measurements as accurately as possible. A context-free grammar is defined as a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{S})$ where $\mathcal{N}$ are non-terminals and $\mathcal{T}$ are terminals. Non-terminals can be rewritten by a production rule $r \in \mathcal{R}$. A production rule r is of the form $r = t \rightarrow \alpha$, where $t \in \mathcal{N}$ and $\alpha \in (\mathcal{T} \cup \mathcal{R})^*$ [1] Starting from a starting symbol $\mathcal{S}$, the equations are constructed as a syntax tree. As the grammar rules, based on human expert knowledge, can represent constraints on the discovered equations, the latter should be easier to understand for humans than without those constraints.

The tree search for the best-fitting equation can be performed by a Monte Carlo tree search (MCTS). In the MCTS, a decision has to be made at each node of the search tree, which child node should be explored further. The decision depends on a prior (e.g. from a uniform distribution), how often a child node has already been visited, and the results obtained so far when this child node was explored. The idea is to find a good trade-off between exploring unknown states and nodes already shown to be promising. In a first step, a recognition model, as proposed in the DreamCoder [2], is investigated. Instead of using an uniform distribution as a prior to select a rule r from the grammar to extend the syntax tree, the recognition module should guide the search. The recognition model has two different information pipelines. First, the measured values of the experiment, which are encoded by Long short-term memories (LSTMs) or multilayer perceptrons (MLP). Second, it consists of the current state of the syntax tree. In experiments, several representations (see Table 1) of the trees based on LSTM and attention mechanisms are compared. The results using both information pipelines is superior to using only one of the pipelines individually and much better than a random guessing baseline on a set of example tasks.

Table 1: Approaches to process the equation syntax trees. The synatx tree at the bottom left of Figure 1 is used as example

Approach	Pattern	Example
Hashed path [3]	left symbol, Hash(path), right symbol	Path A: 1, Hash(Number, S), Mul Path B: Product, Hash(S), Number
Full tree	Node [Child Nodes]	S[Number [1] Mul Product]
Fringe	Fringe nodes	1 Mul Product

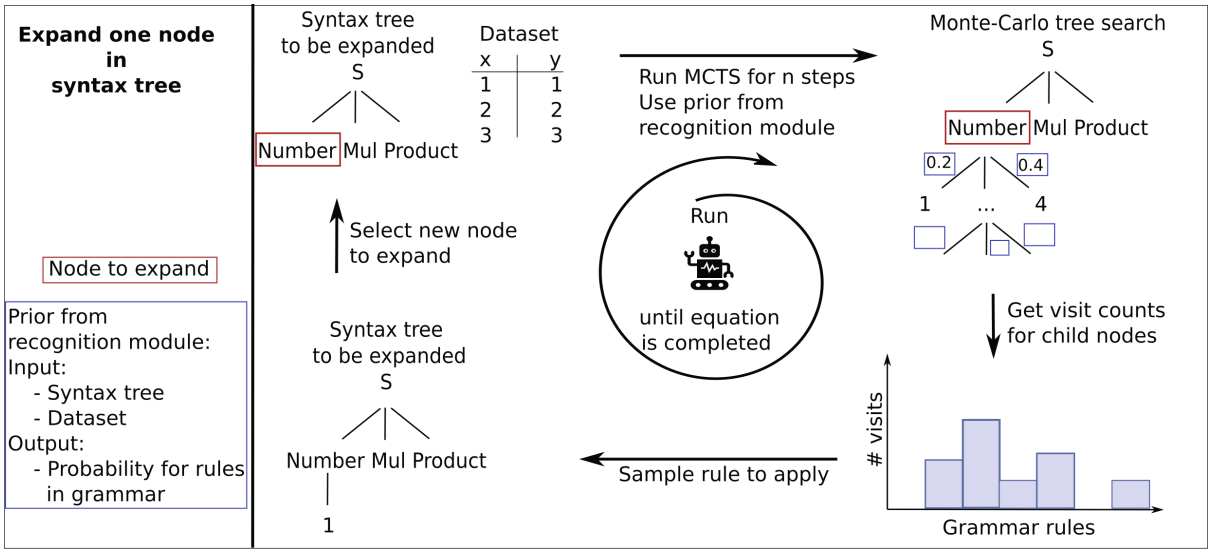

Figure 1: Sequential generation of an equation using the recognition model. The visit counts of the MCTS are used as probabilities of which production rule from the grammar should be used to expand the syntax tree. The recognition module is trained with the visit counts from the MCTS as well as the reward of the finished equation.

Using the one-step recognition model in a neural-guided Monte Carlo tree search, as in AlphaZero [4], our system is able to discover new equations on the basis of multiple related previous tasks. We evaluate the performance of the discovered equation by calculating $reward = 1 - L_2(y_{dataset}, y_{prediction})$. Figure 1 shows the usage of the recognition module to build a complete syntax tree

The next step will be to use the grammar to enrich real-world training data with self-generated examples, again inspired by the DreamCoder [2]. The grammar itself will be continuously improved by adding frequent patterns from solutions found for the real-world data. Through the mutual improvement of the recognition module, the Monte Carlo tree search, and the grammar, the system could be expected to outperform equation discovery systems based on chance or human heuristics.

References

[1] Jure Brence, Ljupčo Todorovski, and Sašo Džeroski. Probabilistic grammars for equation discovery. *Knowledge-Based Systems*, 224:107077, July 2021.

[2] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pages 835–850, 2021.

[3] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning distributed representations of code. *Proc. ACM Program. Lang.*, 3(POPL), 2019.

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.