Interactive Reinforcement Learning for Symbolic Regression from Multi-Format Human-Preference Feedbacks

Laure Crochepierre\textsuperscript{1,2}, Lydia Boudjeloud-Assala\textsuperscript{1}, Vincent Barbesant\textsuperscript{2}
\textsuperscript{1}Université de Lorraine, CNRS, LORIA, F-57000 Metz, France
\textsuperscript{2}Réseau de Transport d’Electricité (Rte) R&D, Paris, France
\{laure.crochepierre, lydia.boudjeloud-assala\}@univ-lorraine.fr,
\{laure.crochepierre,vincent.barbesant\}@rte-france.com

Abstract
In this work, we propose an interactive platform to perform grammar-guided symbolic regression using a reinforcement learning approach from human-preference feedback. To do so, a reinforcement learning algorithm iteratively generates symbolic expressions, modeled as trajectories constrained by grammatical rules, from which a user will elicit preferences. The interface gives the user three distinct ways of stating its preferences between multiple sampled symbolic expressions: categorizing samples, comparing pairs, and suggesting improvements to a sampled symbolic expression. Learning from preferences enables users to guide the exploration in the symbolic space toward regions that are more relevant to them. We provide a web-based interface testable on symbolic regression benchmark functions and power system data.

1 Introduction
Symbolic regression (SR) is the task of automatically finding a symbolic function \( f \), represented as a mathematical expression (e.g. \( f(X) = X^2 \)), that accurately models the relationship \( f(X) = y \) between \( X = (X_0, \ldots, X_{N-1}) \in \mathbb{R}^{N \times D} \) an observation set with \( N \) variables and \( D \) observations, and a target variable \( y \in \mathbb{R}^D \). Until recently, SR was mostly performed using Genetic Programming algorithms [Koza, 1990]. However, with current advancements along with the need for more interpretable models in the Deep Learning community, other methods now propose to tackle SR by encoding \( f \) in the neural network activations [Kim et al., 2021] or using Deep Reinforcement Learning to explore the symbolic search space [Petersen et al., 2021].

In addition to these algorithmic improvements, more data are now available with the rise of Big Data environments. This is notably the case in industrial and real-world applications, where SR has been of large interest [Wang et al., 2019]. However, the exact underlying formula might not be known with this type of data, leading to a scenario where SR is used as an exploratory tool to extract a symbolic representation to describe and predict a given target variable. Eventually, as the ultimate goal of SR is to find a expression that will make sense to the user, we also want to rely on human knowledge and expertise to find more relevant expressions and reduce the problem complexity.

As well as relying on human knowledge, various approaches propose to include interactivity in SR. For instance, Genetic Programming has long been combined with interactivity [Petersen et al., 2017] where it was found useful, for example, to rank individuals based on their subjective judgment. Interactivity is also a hot topic in the Reinforcement Learning (RL) community, as it can offer other interaction and learning modalities such as defining a loss function based on human preference [Christiano et al., 2017] or ranking [Kuhlman et al., 2018]. Still, few works yet focus on interactive RL (iRL) for SR. In that direction, a recent work [Kim et al., 2020] propose to control the RL algorithm by dynamic hyperparameters updates and expressions selection/removal from the batch.

However, interaction schemes such as the ones detailed above can be found sometimes exhausting [Poli et al., 1997] and even annoying or challenging for non-machine learning practitioners [Amershi et al., 2014]. For example, scoring algorithm solutions can be a demanding task and require task-specific prior knowledge [Wirth et al., 2017]. To make the user experience more pleasant and reduce the number of required interactions, we decided to implement several guidelines [Amershi et al., 2014] such as acknowledging that users want to give more than just “labels” and they like demonstrating how the algorithm should work. Thus, to avoid user boredom and favor user engagement while requesting as few interactions as possible, we focused on learning from user preferences over symbolic expression pairs. As suggested in Wirth et al. [Wirth et al., 2017], working on entire RL trajectories reduces the number of interactions and also makes sense with the sparse reward used in SR environments where expressions can only be evaluated when they are complete.

Contributions
More precisely, our contributions to handle the abovementioned challenges are the following\textsuperscript{1}:

\begin{itemize}
  \item We propose an interactive platform to perform interactive Reinforcement Based Grammar Guided SR (interactive RBG2-SR) by learning directly from human preference over expression pairs.
  \item We offer multiple ways to elicit preferences:
    \begin{itemize}
      \item 1) with preference categories, by labeling expressions as
    \end{itemize}
\end{itemize}

\textsuperscript{1}Demo video available at: https://youtu.be/_HVxkz1KzMA
“Best”, “Average”, or “Bad”, which allows generating many preference pairs in fewer interactions [Kuhlman et al., 2019]; 2) with a direct definition of preference over pairs; 3) or with solution suggestion by trying to improve an expression proposed by the algorithm with a user-created expression.

- We embed several SR tasks taken from state-of-the-art benchmarks with increasing complexity, along with their grammar and an industrial SR dataset from which to perform a more complex exploratory SR.

The rest of this paper gives a description of the system algorithm and interface (Section 2), propose a case study (Section 3), and finally draw conclusions (Section 4).

2 System Description

The training procedure is presented in Figure 1. Our algorithm is based on the RBG2-SR algorithm [Crochepierre et al., 2022], a reinforcement-based approach to SR, where grammatical rules constrain the construction of symbolic expressions. The same Partially-Observable Markov Decision Process (POMDP) [Kaelbling et al., 1998] is considered with a finite horizon and sparse reward signal. The state contains information about the partially-constructed expression parse tree. The action is defined as the selection of an accessible rule in the grammar, and the reward is the quantitative measure comparing the target $y$ and the evaluation of the mathematical expression $f(x)$, using a squashed Mean Squared Error (MSE) as cumulative reward: $R_f = \frac{1}{1+MSE(f(x), y)}$.

2.1 Training Algorithm

The SR search consists of the following stages:

Procedure P1: Training Launch To launch the training, the user has to select a dataset file, a grammar file, and a frequency of interaction with the algorithm. The dataset specifies the target variable to model $y$ and provides instances $X$ to evaluate the expression. The grammar is defined in a Backus-Naur Form (BNF) [Knuth, 1964], which is composed of a set of rules used to restrict the sampling of symbolic expressions.

Procedure P2: Trajectories sampling At each step of a trajectory creation, the Neural Agent $\pi$ generates a distribution over accessible grammatical actions $P_t$ (rules might be masked according to the current state). The action is then sampled from this distribution. A sequence of actions builds a trajectory $\tau$, then translated into a symbolic expression $f_{\tau}$. This procedure is vectorized to generate a batch of trajectories. Trajectories are evaluated using $R_{f_{\tau}}$.

Procedure P3: Preferences collection The best-in-batch expressions are displayed to the user. The preference collection mechanism is detailed in Section 2.2.

Procedure P4: Agent optimization The Neural Agent $\pi$ is trained using the REINFORCE algorithm [Williams, 1992] with risk-seeking behavior [Petersen et al., 2021]. Preferences and equivalence between trajectories are inserted into the objective function in order to maximize the probability of occurrence of preferred trajectories while avoiding non-preferred trajectories according to a weighted pairwise disagreement loss [Duchi et al., 2010]. If two trajectories are found relevant, they are both maximized.

In-between runs where interactions occur, trajectories can be re-simulated, and preference re-used. Procedures 2 to 4 are repeated in this order until a termination criterion is met: either the exact expression found, or the maximum number of training iterations reached.

Procedure P5: Grammar update Optionally, at the end of the training, the user can comment on and refine grammatical constraints. Once transposed into new rules, these comments will improve the grammar used in the next training sessions.

2.2 System Interface

At a given interaction frequency, the best-in-batch expressions are presented to the user, ordered by cumulative reward. Preferences over expressions are collected during procedure P3 (see Figure 1) according to three different mechanisms:

Preference categories As shown in Figure 2, the user is asked to classify expressions into three ordered categories: “Best”, “Average”, and “Bad”. We then generate pairs from categories, considering that each expression in the “Best” category is better than all expressions in the two lower ones, (similarly “Average” > “Bad”). This strategy, derived from the work of [Kuhlman et al., 2018], allows generating more pairs in fewer interactions than the direct preference over-pairs. We propose Regular Expressions filtering over the remaining expressions to further reduce the number of clicks.

Preference pairs The second Tab, shown in Figure 3, initially presents a list of randomly selected best expression pairs to the user. It is inspired by preference over segments strategy [Christiano et al., 2017]. The user can prefer one, another, both, or none of the two expressions in each pair. New pairs can be manually added to the list. This Tab, shown in Figure 3, is complementary to the first interaction and aims at refining the preference over categories. For example, the
user can select preference over expressions classified in the same category in Tab 1. However, as this interaction is more time-consuming [Kuhlman et al., 2019], it’s not used as the main preference collection mechanism.

**Expression suggestion** The third Tab in Figure 4 focuses on improving an expression sampled by the algorithm in compliance with the grammatical rules. Here, the user first selects an expression in the batch. Then, he creates a new expression from scratch by iteratively selecting actions-rules in the BNF grammar. The expression proposed by the user will be considered better than the one from the batch in order to create a pair. This Tab is devoted to users who have been taught how BNF grammars work. It might allow avoiding local optima when the algorithm lacks diversity.

## 3 Case Study

We implemented a web-based platform\(^2\) using Dash Plotly library in Python [Hossain et al., 2019]. The neural network architecture uses PyTorch CPU implementation [Paszke et al., 2019]. The code is freely available on Github\(^3\).

### 3.1 Exploration of a Benchmark Function

First, we propose testing our platform on a standard SR benchmark. The actual target symbolic expression is known by the research team but not to the user. The objective is here for the user to recover the exact formula for an expression selected from the Nguyen [Uy et al., 2011] SR benchmark. A grammar is provided and comprise functions such as: +, -, \times/, \exp, \log, \sin, \cos. Two cases can be studied: 1) The exact expression isn’t known a priori. The user must explore the proposed expressions through both visual and score-based comparisons to iteratively forge an opinion about the correct solution. 2) The user can have domain-related concepts in mind (or be given) which are necessary to represent the target expression. With these concepts in mind, the user can then attempt to act as a teacher [Mosqueira-Rey et al., 2021] by finding examples to represent each concept and having them taught to the algorithm iterating over explain/review steps.

### 3.2 Power System Example

This second example focuses on an industrial use case: representing a set of historical sensor measurements as a symbolic expression that respects domain-related physical properties. More precisely, we propose to perform a SR task on electrical power network simulated data to uncover an unknown simplified physical relationship on a given power line. Simulated data are obtained using the Grid2Op platform\(^4\), where power system experts have selected the power line to study to be representative of their industrial goal. A predefined grammar [Crochepierre et al., 2020] is proposed to insert physical relationships such as Ohm’s and Kirchoff’s Laws.

## 4 Conclusions and Discussion

We present an interactive interface to perform grammar-guided symbolic regression from human-preference feedback. The platform uses three preference collection mechanisms and proposes use cases with incremental complexity. It will allow to test and compare these interaction modalities when used separately and jointly. This preference-based learning approach is of particular interest to symbolically study datasets in an exploratory fashion when the user might have insights on the solution constraints to consider but does not know the exact symbolic expression beforehand.

## References


[Christiano et al., 2017] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Sys-


