Integrating top-level constraints into a symbolic regression search algorithm

Marko Djukanović[1], **Aleksandar Kartelj**[2]

[1] University of Banja Luka
Faculty of Sciences and Mathematics
Mladena Stojanovića 2, Banja Luka, Bosnia and Herzegovina
Email: marko.djukanovic@pmf.unibl.org
[2] University of Belgrade
Faculty of Mathematics
Studentski trg 16, Belgrade, Serbia
Email: aleksandar.kartelj@gmail.com

Abstract

In this paper we deal with the well-known symbolic regression problem. Previously, we have proposed the efficient metaheuristic approach called RILS-ROLS for symbolic regression that combines the iterated local search algorithm with the ordinary least squares method. Now we extend this work by integrating three top-level constraints in RILS-ROLS: 1) monotonicity, 2) local Lipschitz continuity, and 3) probability distribution difference. Their integration in RILS-ROLS (or in any other search algorithm for symbolic regression) can be used to improve the trajectory of search algorithm through the search space. We propose an implicit modification of the search trajectory by imposing subtle penalties within the nonlinear fitness function. These penalties quantify the inadmissibility of the candidate symbolic expression with respect to the imposed constraints. Empirical results have shown that this approach is promising and that the integration of other, not necessarily mathematical constraints, has high potential.

Key words: symbolic regression, iterated local search, metaheuristic, mathematical properties, monotonicity, continuity, empirical probability distribution.

1 Introduction

Symbolic Regression (SR) aims to find mathematical formulas that explain given data. More precisely, given a set of input data $\mathbf{X}$ and a set of corresponding output target variables $\mathbf{y}$, the goal is to find the exact mathematical formula f that explains the target variable in terms of the input variables, i.e., $\mathbf{y} = f(\mathbf{X})$. Classical regression models usually assume the model structure of f – for example, linear regression, polynomial regression, artificial neural networks, and many others. Symbolic regression, on the other hand, avoids the assumption of a model structure and therefore attempts to search the vast space of possible mathematical expressions by combining allowed arithmetic operations and elementary mathematical functions such as: $+$, $-$, $\cdot$, $/$, $\sqrt{x}, x^2$, sin, cos, log, exp, arcsin, arccos, etc. SR has recently been shown to be NP-hard [1].

Koza [2] introduced the SR problem as a specific application of genetic programming (GP). GP can be viewed as a nonlinear variant of a more familiar genetic algorithm (GA) – unlike GA, which uses a linear solution representation, GP uses a nonlinear (usually tree-like) solution structure. The SR problem has been tackled by many different methods. Most of them can be classified as combinatorial optimization methods, but

there are also machine learning approaches, or even combinations of optimization and ML methods. Since the list of SR methods is very long, we refer the reader to the work of La Cava et al. [3], which proposes an open source, reproducible benchmarking platform for SR called `SRBench`. Within `SRBench`, the performance of 14 relevant SR methods from the literature under the same experimental conditions are compared.

2 RILS-ROLS method for symbolic regression

Recently, Kartelj and Djukanović [4] have developed a novel approach called RILS-ROLS for solving closed-form SR problems. The method combines the popular iterated local search metaheuristic (ILS) [5] with the ordinary least squares (OLS) [6] method. The goal of ILS is to deal with combinatorial aspects of the search space, i.e., the model structure, while OLS deals with continuous aspects of the search space, i.e., the fitting of coefficients. It was shown that RILS-ROLS outperforms all 14 comparison methods from SR on the benchmarks of `SRBench` where ground-truth is known.

3 Integrating top-level constraints into RILS-ROLS

We now propose an extension of RILS-ROLS that allows the search algorithm to take into account so-called *top-level* constraints. These top-level constraints are actually properties of function f. We explain how three types of top-level constraints can be integrated: 1) monotonicity, 2) local Lipshitz continuity, and 3) empirical probability distribution difference. For example, if it is known that the function f is monotonically increasing, this information can be used to navigate the algorithm through the search space. This extension can be done in at least two ways: 1) by incorporating a *hard* constraint that prohibits the search algorithm from even considering a solution that does not satisfy this constraint, and 2) by incorporating a *soft* constraint that allows any solution to be considered, but penalized accordingly. We decided to integrate them in a soft way, which required a reconstruction of the fitness function proposed in [4].

$$fit(f) = (2 - R^2(f)) \cdot (1 + RMSE(f)) \cdot (1 + p_{size} \cdot size(f)) \tag{1}$$

As can be seen, the RILS-ROLS fitness function is a nonlinear combination of several key aspects of solution quality: R^2, $RMSE$, and solution (formula) size. Since minimization is performed, the idea behind Equation (1) is to favour small solutions with low $RMSE$ and high R^2. Using the analogous idea for imposing fitness function preferences, we propose the following alternative fitness function:

$$fit'(f) = fit(f) \cdot (1 + p_{mono} \cdot mono(f)) \cdot (1 + p_{cont} \cdot cont(f)) \cdot (1 + p_{dist} \cdot dist(f)). \tag{2}$$

The penalty parameters p_{mono}, p_{cont}, and p_{dist} are clearly positive and are meant to quantify the importance of satisfying the given constraints. In particular, if any of the constraints is not imposed, the corresponding penalty is simply set to 0. The terms $mono(f)$, $cont(f)$, and $dist(f)$ quantify the satisfiability of the imposed constraints. All of these terms are computed numerically on the training data set.

To quantify the monotonicity score, the algorithm must be informed about the input variable(s) with respect to which the monotonicity is calculated. Beside this, the algorithm must be informed about the type of the monotonicity: strictly increasing, strictly decreasing, non-increasing or non-decreasing. More precisely, the $mono(f)$ score represents a fraction of training set points that follow the imposed type of monotonicity.

Lipschitz continuity [7] measures how much the function values of $f(x)$ may be changed when x changes. More precisely, the function is Lipschitz continous iff there exists a constant $K \geq 0$ such that for all $x_1 \neq x_2$, $d(f(x_1), f(x_2))/d(x_1, x_2) \leq K$. Local Lipschitz continuity is more relaxed as it checks Lipschitz continuity condition only in a neighborhood of a given point x. The $cont(f)$ score is calculated as $RMSE$ between $d(y_1, y_2)/d(x_1, x_2)$ and $d(f(x_1), f(x_2))/d(x_1, x_2)$ for all pairs of training points $((x_1, y_1), (x_2, y_2))$ that are close enough, i.e., $d(x_1, x_2) \leq \epsilon_{cont}$, where ϵ_{cont} is parameter.

Finally, the $dist(f)$ score is based on the Bhattacharyya coefficient (BC) [8] for discrete probability distributions. More precisely, the true $\mathbf{y}$ and predicted target values $f(\mathbf{X})$ are put into m equally sized bins, which is followed by a calculation of their respective vectors of frequencies $\mathbf{p}$ and $\mathbf{q}$. The BC is then calculated as $BC(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{m} \sqrt{p_i \cdot q_i}$.

4 Experimental results

RILS-ROLS[1] equipped with the three top-level constraints has been tested on four[2] fairly complex formulae (instances). Each instance contains 200 feature vectors $\mathbf{x} \in [0, 1]^3$, with 75% of data reserved for training and 25% for testing. The termination criterion per each run was set to 100,000 fitness evaluations.

To demonstrate the effect of integrating monotonicity and/or empirical probability distribution constraints in RILS-ROLS, we tested the formula f_0 under 28 different scenarios (four different constraint settings combined with seven levels of Gaussian noise, starting from no-noise scenario to noise of 99%). Note that f_0 is increasing w.r.t. x_0 on the given dataset points. Table 1 reports the final R^2 score of each of 28 scenarios. The penalty parameters p_{mono} and p_{dist} were tuned to values 1 and 10, respectively. It can be concluded that the monotonicity and distribution constraints independently improved the accuracy of solutions in the presence of small-to-medium noise. However, imposing only monotonicity or only distribution constraint is not sufficient for obtaining an accurate solution in the presence of high noise. Fortunately, imposing both constraints improves the results in this scenario.

Table 1: R^2 comparison on formula $f_0(x_0, x_1, x_2) = 1000 \cdot x_0 + \sqrt{200 + x_1 + x_2}$ with monotonicity and distribution difference constraints.

Mono. ($p_{mono} = 1$)	Dist. ($p_{dist} = 10$)	0	0.1	0.3	0.5	0.7	0.9	0.99
		1	**0.998**	0.47	0.932	0.628	0.074	-0.096
✓		**1**	**0.998**	**0.979**	0.951	0.854	0.821	0.768
	✓	**1**	0.992	0.976	0.976	**0.945**	0.802	0.494
✓	✓	**1**	0.992	0.965	**0.999**	0.873	**0.966**	**0.915**

[1]Codes, datasets and results available at https://github.com/kartelj/rils-rols/tree/top_level_math_constraints.

[2]Due to the limited number of pages, more extensive experimental evaluation could not fit into the paper format.

Table 2: R^2 comparison on formula f_1, f_2, f_3 with Lipschitz continuity constraint.

Formula / p_{cont}	0	1	10
$f_1(x_0, x_1, x_2) = \cos(e^{x_0} + e^2) \cdot e^{x_2 \cdot \cos(x_1)}$	-0.326	0.15	**0.296**
$f_2(x_0, x_1, x_2) = \sin(x_0)^3 \cdot \cos(x_1)^3 \cdot \sin(x_2)^3 \cdot \sin(x_0 \cdot x_1 \cdot x_2)$	0.837	0.994	**0.996**
$f_3(x_0, x_1, x_2) = \cos(x_0 + x_1)^4 \cdot sin(x_1 \cdot x_2)^4$	-0.642	-76.289	**-0.204**

Table 2 reports the results of the remaining three instances (f_1, f_2 and f_3) w.r.t. local Lipschitz continuity constraint, under varying p_{cont}. Although the results are not statistically significant, there is an indication that R^2 might improve by incorporating this type of constraint. Note that formulae f_1, f_2 and f_3, unlike formula f_0, have complicated structure, and are therefore more difficult to solve.

5 Conclusions

The experimental evaluation showed that integrating top-level constraints into the SR search process plays an important role in finding more promising regions of the search space, thus obtaining more accurate solutions. Future research could include other top-level constraints, such as periodicity, auto-correlation, number of local minima/maxima, etc.

References

[1] Virgolin M, Pissis SP. Symbolic Regression is NP-hard. arXiv preprint arXiv:220701018. 2022;.

[2] Koza JR. Genetic programming as a means for programming computers by natural selection. Statistics and computing. 1994;4(2):87–112.

[3] La Cava W, Orzechowski P, Burlacu B, de França FO, Virgolin M, Jin Y, et al. Contemporary symbolic regression methods and their relative performance. arXiv preprint arXiv:210714351. 2021;.

[4] Kartelj A, Djukanović M. RILS-ROLS: Robust Symbolic Regression via Iterated Local Search and Ordinary Least Squares. Journal of Big Data. 2023;.

[5] Lourenço HR, Martin OC, Stützle T. Iterated local search: Framework and applications. New York, NY: Springer; 2010.

[6] Leng L, Zhang T, Kleinman L, Zhu W. Ordinary least square regression, orthogonal regression, geometric mean regression and their applications in aerosol science. Journal of Physics: Conference Series. 2007 jul;78(1):012084.

[7] Eriksson K, Estep D, Johnson C, Eriksson K, Estep D, Johnson C. Lipschitz continuity. Applied Mathematics: Body and Soul: Volume 1: Derivatives and Geometry in IR 3. 2004;p. 149–164.

[8] Bhattacharyya A. On a measure of divergence between two statistical populations defined by their probability distribution. Bulletin of the Calcutta Mathematical Society. 1943;35:99–110.