A Semantics based Symbolic Regression Framework for Mining Explicit and Implicit Equations from Data

Quang Nhat Huynh, Hemant Kumar Singh and Tapabrata Ray
School of Engineering and Information Technology, The University of New South Wales, Australia.
quang.huynh@student.adfa.edu.au, h.singh@adfa.edu.au, t.ray@adfa.edu.au

ABSTRACT

Symbolic Regression (SR) is commonly used to identify relationships among variables and responses in a data in the form of analytical, preferably compact expressions. Genetic Programming (GP) is one of the common ways to perform SR. Such relationships could be represented using explicit or implicit expressions, of which the former has been more extensively studied in literature. Some of the key challenges that face SR are bloat, loss of diversity, and accurate determination of coefficients. More recently, semantics and multi-objective formulations have been suggested as potential tools to build more intelligence in the search process. However, studies along both these directions have been in isolation and applied only to selected components of SR so far. In this paper, we intend to build a framework that integrates semantics deeper into more components of SR. The framework could be operated in traditional single objective as well as multi-objective mode and is capable of dealing with both explicit and implicit functions. The constituent modules utilize semantics for compaction of expressions, maintaining diversity by identifying unique individuals, crossover and local exploitation. A comparison of obtained results with those from existing semantics-based and multi-objective approach demonstrates the advantages of the proposed framework.

Keywords

Genetic Programming; Symbolic Regression; Sampling Semantics; Explicit Equations; Implicit Equations

1. INTRODUCTION

The problem of finding comprehensible relationships among variables and responses in observed data is of significant interest in data analytics. Symbolic regression (SR) is a popular method to discover such relationships, and one of the common approaches to perform SR by *evolving* expressions using Genetic Programming (GP). The expressions defining these relationships can be either explicit ($y = f(\mathbf{x})$) or implicit ($f(\mathbf{x}, y) = 0$), of which the explicit functions have been studied more comprehensively in literature. The conventional SR usually suffers limitations such as *bloat* and diversity reduction. To overcome these shortcomings, two of the emerging trends in GP are the use of semantics and multi-objective formulations. Semantics has been individually integrated into different components of SR such as initialization, crossover and mutation in [4, 1, 3]. One of the recent works reported on multi-objective GP is [5] where the tree depth and error are considered as two separate objectives to be minimized, in order to obtain compact, accurate expressions. While the methods have shown promising results, they have been applied in isolation to different components of SR.

In this paper, we integrate semantics deeper into more components of SR to improve the framework presented in [5]. The proposed framework can be executed in single objective or multi-objective mode, and is capable of dealing with both explicit and implicit problems. Semantics is used in different key components, such as compaction, uniqueness, crossover, and local exploitation. A constant optimization scheme is introduced to determine coefficients accurately. New mutation and local exploitation operations are also suggested to enhance the global exploration and local exploitation abilities. The results are compared with an existing semantics based algorithm [4] for explicit functions and a multi-objective algorithm [5] for implicit functions.

GECCO'16 Companion July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2908971

2. SAMPLING SEMANTICS

The authors of [4] defined semantics for real-value SR based on a method called *sampling semantics*, where a number of points in the problem domain are randomly sampled and used to evaluate the expressions in the population. If the difference between evaluated values of two expressions is less than a given threshold, they are considered semantically equivalent. However, in our implementation, we calculate the semantics only based on the *given fitness cases* to reduce the computations required. As a result, a large proportion of allotted computational resource is saved for other mechanisms. Two types of semantic crossover methods, *semantics aware crossover* (SAC) and *semantic similarity-based crossover* (SSC) [4], are incorporated in our framework.

3. PROPOSED FRAMEWORK

The proposed framework is referred to as semantics based symbolic regression (SSR) and summarized in Algorithm 1. The enhanced mechanisms (denoted with asterisk (*) in Algorithm 1) are subsequently described. Expressions in SSR have tree structure representation as in [4, 5]. The first fitness function used in SSR is Mean Square Error (MSE) for explicit expressions and Mean Logarithmic Error in Derivatives (MLED) [2, 5] for implicit expressions. The second fitness function is the depth of the tree.

Algorithm 1 Semantics based Symbolic Regression (SSR)

1: **Initialization:** Generate initial population; Evaluate and sort the trees in the population;
2: **while** termination condition not met **do**
3: **Crossover:** Perform semantic crossover of parents selected using binary tournament; Find the **compact*** form of the child population;
4: **Mutation:** Perform **multi-point mutation*** on the child population; Find the compact form of the child population;
5: **Evaluation:** Evaluate the fitness of the child population;
6: Remove duplicate trees through **uniqueness*** check;
7: **Sorting:** Sort the trees in child + parent population based on single or multi-objective criteria as applicable;
8: Select the best trees to carry to next generation;
9: Perform **constant optimization*** and **local exploitation***; Re-sort the population;
10: **end while**

3.1 Compactness

To find the compact form of an expression, first step is to replace the operator nodes which always return constant values by terminal constant nodes. The second step is to identify the sub-trees having the form of $(0 + expression)$, $(expression + 0)$, $(expression - 0)$, $(1 \times expression)$, $(expression \times 1)$ and $(expression \div 1)$ and shorten them. For a particular node, if all the semantic values are within a pre-set threshold with respect to the mean of these values, the operator node is considered as always returning a constant.

3.2 Uniqueness

This mechanism is used to maintain the diversity of the trees in the population. To compare if two trees are identical, the string forms of the expressions represented by the trees, number of nodes, number of operator nodes, identities of the operator nodes, number of variable nodes and identities of the variable nodes are examined (in that order). If all these conditions fail to differentiate the two trees, the semantics is made use of. If the original semantics of two trees are different, all the constant nodes of the two trees are reset to 1. Then, the two trees are evaluated again to obtain their new semantics. If new semantics are the same for both trees, it means that the structure of the trees are alike and only the values of the constant nodes are different. Between the two trees, the one having better fitness is kept.

3.3 Constant Optimization and Local Exploitation

To determine the best values of constants in an expression, a local search method is employed. For this step, the terminal variable nodes are treated as constants while the terminal constant nodes are treated as variables. The form of the expression is known at this stage so any reasonable local search method can be applied to identify the most appropriate values for the terminal constant nodes.

Local exploitation is simply a mutation operator combined with fitness checking. After the mutation, the fitness is re-evaluated and the mutated expression is kept only if the new fitness is better than the previous one.

3.4 Multi-point Mutation

A fixed percentage of nodes in a tree are selected for mutation and half of the selected nodes are forced to change to new nodes of the same class (i.e., binary is mutated to another binary node, a unary to another unary node and a terminal to another terminal node). The other half of selected nodes are forced to mutate to new nodes belonging to other classes, which are chosen randomly. This new operation is referred to as multi-point mutation and helps to enhance the exploration ability of SSR.

4. NUMERICAL EXPERIMENTS

SSR is tested with 10 explicit expressions from [4] in single objective mode and 3 implicit expressions from [5] in multi-objective mode. A population size of 500 is used, with termination criteria set to 15×10^6 node evaluations for explicit and 100 generations for implicit cases. Thirty independent runs are performed for each problem. A run is considered as *successful* in the case of explicit expression if there is at least one tree in the population having the difference between predicted and observed outputs less than 0.01 for all fitness cases [4]. For implicit expressions, a run is considered successful if it can find the exact expressions from which the input data is generated. SSR performs better than the algorithm SSC12 reported in [4] in finding explicit expression as illustrated in Table 1. It also uncovers the correct implicit expressions in all the test runs faster compared to the previously reported results [5].

Table 1: Percentage of successful runs of SSC12 and single-objective SSR for 10 explicit problems

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
SSC12	67	33	14	12	47	47	66	38	37	51
SSR	100	93	96	73	100	100	80	73	90	96

5. SUMMARY

In this paper, we present a semantics based symbolic regression (SSR) framework to mine explicit and implicit expressions from given data. The proposed approach utilizes benefits of semantics for various components, such as compactness, uniqueness, crossover and local exploitation. In addition, a new constant optimization technique and multi-point mutation are used to strengthen the approach further. These enhancements help to deal with the persistent obstacles in SR, such as bloat, diversity and coefficient identification as well as enhance the exploration and exploitation abilities of the algorithm. Numerical experiments are conduced using both single and multi-objective forms of SSR. The results obtained are compared with existing techniques for both implicit and explicit functions, which demonstrate the competence and ability of SSR to find accurate, compact expressions in relatively low computational budget.

6. REFERENCES

[1] A. Moraglio, K. Krawiec, and C. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31. 2012.

[2] M. Schmidt and H. Lipson. Symbolic regression of implicit equations. In *Genetic Programming Theory and Practice VII*, Genetic and Evolutionary Computation, pages 73–85. October 2010.

[3] N. Uy, N. Hoai, and M. ONeill. Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In *15th International Conference on Soft Computing*, 2009.

[4] N. Q. Uy, N. X. Hoai, M. OâĂŹNeill, R. I. McKay, and E. Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

[5] B. Wang, H. Singh, and T. Ray. A multi-objective genetic programming approach to uncover explicit and implicit equations from data. In *IEEE Congress on Evolutionary Computation (CEC) 2015*, pages 1129–1136, May 2015.