Neurocomputing 000 (2017) 1–8

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Block building programming for symbolic regression

Chen Chen [a,b], Changtong Luo [a,*], Zonglin Jiang [a,b]

[a] State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

*Article history:*
Received 24 May 2017
Revised 3 September 2017
Accepted 17 October 2017
Available online xxx

Communicated by Prof. H.R. Karimi

*Keywords:*
Symbolic regression
Separable function
Block building programming
Genetic programming

ABSTRACT

Symbolic regression that aims to detect underlying data-driven models has become increasingly important for industrial data analysis. For most existing algorithms such as genetic programming (GP), the convergence speed might be too slow for large-scale problems with a large number of variables. This situation may become even worse with increasing problem size. The aforementioned difficulty makes symbolic regression limited in practical applications. Fortunately, in many engineering problems, the independent variables in target models are separable or partially separable. This feature inspires us to develop a new approach, block building programming (BBP). BBP divides the original target function into several blocks, and further into factors. The factors are then modeled by an optimization engine (e.g. GP). Under such circumstances, BBP can make large reductions to the search space. The partition of separability is based on a special method, block and factor detection. Two different optimization engines are applied to test the performance of BBP on a set of symbolic regression problems. Numerical results show that BBP has a good capability of structure and coefficient optimization with high computational efficiency.



## 1. Introduction

Data-driven modeling of complex systems has become increasingly important for industrial data analysis when the experimental model structure is unknown or wrong, or the concerned system has changed [1,2]. Symbolic regression aims to find a data-driven model that can describe a given system based on observed input-response data, and plays an important role in different areas of engineering such as signal processing [3], system identification [4], industrial data analysis [5], and industrial design [6]. Unlike conventional regression methods that require a mathematical model of a given form, symbolic regression is a data-driven approach to extract an appropriate model from a space of all possible expressions $\mathcal{S}$ defined by a set of given binary operations (e.g. $+, -, \times, \div$) and mathematical functions (e.g. $\sin, \cos, \exp, \ln$), which can be described as follows:

$$f^* = \arg\min_{f\in\mathcal{S}} \sum_i \left\| f(\mathbf{x}^{(i)}) - y_i \right\|, \tag{1}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are sampling data. $f$ is the target model and $f^*$ is the data-driven model. Symbolic regression is a kind of non-deterministic polynomial (NP) problem, which simultaneously optimizes the structure and coefficient of a target model. How to use an appropriate method to solve a symbolic regression problem is considered as a kaleidoscope in this research field [7–9].

Genetic programming (GP) [10] is a classical method for symbolic regression. The core idea of GP is to apply Darwin's theory of natural evolution to the artificial world of computers and modeling. Theoretically, GP can obtain accurate results, provided that the computation time is long enough. However, describing a large-scale target model with a large number of variables is still a challenging task. This situation may become even worse with increasing problem size (increasing number of independent variables and range of these variables). This is because the target model with a large number of variables may result in large search depth and high computational costs of GP. The convergence speed of GP may then be too slow. This makes GP very inconvenient in engineering applications.

Apart from basic GP, two groups of methods for symbolic regression have been studied. The first group focused on evolutionary strategy, such as grammatical evolution [11] and parse-matrix evolution [12]. These variants of GP can simplify the coding process. Gan et al. [13] introduced a clone selection programming method based on an artificial immune system. Karaboga et al. [14] proposed an artificial bee colony programming method based on the foraging behavior of honeybees. However, these methods are still based on the idea of biological simulation processes. This helps little to improve the convergence speed when solving large-scale problems.

* Corresponding author.
*E-mail addresses:* chenchen@imech.ac.cn (C. Chen), luo@imech.ac.cn (C. Luo), zljiang@imech.ac.cn (Z. Jiang).

https://doi.org/10.1016/j.neucom.2017.10.047
0925-2312/

The second branch exploited strategies to reduce the search space of the solution. McConaghy [15] presented the first non-evolutionary algorithm, fast function eXtraction (FFX), based on pathwise regularized learning, which confined its search space to generalized linear space. However, the computational efficiency is gained at the sacrifice of losing the generality of the solution. More recently, Worm [16] proposed a deterministic machine-learning algorithm, prioritized grammar enumeration (PGE). PGE merges isomorphic chromosome presentations (equations) into a canonical form. The author argues that it could make a large reduction to the search space. However, debate still remains on how the simplification affects the solving process [17–19].

In many scientific or engineering problems, the target models are separable. Luo et al. [20] presented a divide-and-conquer (D&C) method for GP. The authors indicated that detecting the correlation between each variable and the target function could accelerate the solving process. D&C can decompose a concerned separable model into a number of sub-models, and then optimize them. The separability is probed by a special method, the bi-correlation test (BiCT). However, the D&C method can only be valid for an additively/multiplicatively separable target model (see Definition 1 in Section 2). Many practical models are out of the scope of the separable model (Eqs. (6) and (7)). This limits the D&C method for further applications.

In this paper, a more general separable model that may involve mixed binary operators, namely plus (+), minus (−), times (×), and division (÷), is introduced. In order to get the structure of the generalized separable model, a new approach, block building programming (BBP), for symbolic regression is also proposed. BBP reveals the target separable model using a block and factor detection process, which divides the original model into a number of blocks, and further into factors. Meanwhile, binary operators could also be determined. The method can be considered as a bi-level D&C method. The separability is detected by a generalized BiCT method. Numerical results show that BBP can obtain the target functions more reliably, and produce extremely large accelerations of the GP method for symbolic regression.

The presentation of this paper is organized as follows. Section 2 is devoted to the more general separable model. The principle and procedure of the BPP approach are described in Section 3. Section 4 presents numerical results, discussions, and efficiency analysis for the proposed method. In the last section, conclusions are drawn with future works.

## 2. Definition of separability

### 2.1. Examples

As previously mentioned, in many applications, the target models are separable. Below, two real-world problems are given to illustrate separability.

**Example 1.** When developing a rocket engine, it is crucial to model the internal flow of a high-speed compressible gas through the nozzle. The closed-form expression for the mass flow through a choked nozzle [21] is

$$\dot{m} = \frac{p_0 A^*}{\sqrt{T_0}} \sqrt{\frac{\gamma}{R}} \left( \frac{2}{\gamma+1} \right)^{(\gamma+1)/(\gamma-1)}, \tag{2}$$

where $p_0$ and $T_0$ represent the total pressure and total temperature, respectively. $A^*$ is the sonic throat area. $R$ is the specific gas constant, which is a different value for different gases. $\gamma = c_p/c_v$, where $c_v$ and $c_p$ are the specific heat at constant volume and constant pressure. The sub-functions of the five independent variables, $p_0$, $T_0$, $A^*$, $R$, and $\gamma$ are all multiplicatively separable in Eq. (2). That is, the target model can be re-expressed as follows

$$\begin{aligned} \dot{m} &= f(p_0, A^*, T_0, R, \gamma) \\ &= f_1(p_0) \times f_2(A^*) \times f_3(T_0) \times f_4(R) \times f_5(\gamma). \end{aligned} \tag{3}$$

The target function with five independent variables can be divided into five sub-functions that are multiplied together, and each with only one independent variable. Furthermore, the binary operator between two sub-functions could be plus (+) or times (×).

**Example 2.** In aircraft design, the lift coefficient of an entire airplane [22] can be expressed as

$$C_L = C_{L\alpha}(\alpha - \alpha_0) + C_{L\delta_e}\delta_e \frac{S_{\mathrm{HT}}}{S_{\mathrm{ref}}}, \tag{4}$$

where $C_{L\alpha}$ and $C_{L\delta_e}$ are the lift slope of the body wings and tail wings. $\alpha$, $\alpha_0$, and $\delta_e$ are the angle of attack, zero-lift angle of attack, and deflection angle of the tail wing, respectively. $S_{\mathrm{HT}}$ and $S_{\mathrm{ref}}$ are the tail wing area and reference area, respectively. Note that the sub-functions of the variable $C_{L\alpha}$, $C_{L\delta_e}$, $\delta_e$, $S_{\mathrm{HT}}$, and $S_{\mathrm{ref}}$ are separable, but not purely additively/multiplicatively separable. Variables $\alpha$ and $\alpha_0$ are not separable, but their combination $(\alpha, \alpha_0)$ can be considered separable. Hence, Eq. (4) can be re-expressed as

$$\begin{aligned} C_L &= f\left(C_{L\alpha}, \alpha, \alpha_0, C_{L\delta_e}, \delta_e, S_{\mathrm{HT}}, S_{\mathrm{ref}}\right) \\ &= f_1(C_{L\alpha}) \times f_2(\alpha, \alpha_0) + f_3\left(C_{L\delta_e}\right) \times f_4(\delta_e) \times f_5(S_{\mathrm{HT}}) \times f_6(S_{\mathrm{ref}}). \end{aligned} \tag{5}$$

In this example, the target function is divided into six sub-functions.

### 2.2. Additively/multiplicatively separable model

The additively and multiplicatively separable models introduced in [20] are briefly reviewed below.

**Definition 1.** A scalar function $f(\mathbf{x})$ with $n$ continuous variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ $(f : \mathbb{R}^n \mapsto \mathbb{R},\ \mathbf{x} \in \mathbb{R}^n)$ is additively separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{m} \alpha_i f_i\left(\mathbf{I}^{(i)}\mathbf{x}\right), \tag{6}$$

and is multiplicatively separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 \cdot \prod_{i=1}^{m} f_i\left(\mathbf{I}^{(i)}\mathbf{x}\right). \tag{7}$$

In Eqs. (6) and (7), $\mathbf{I}^{(i)} \in \mathbb{R}^{n_i \times n}$ is the partitioned matrix of the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, namely $\mathbf{I} = \left[\mathbf{I}^{(1)} \quad \mathbf{I}^{(2)} \quad \ldots \quad \mathbf{I}^{(m)}\right]^\top$, $\sum_{i=1}^{m} n_i = n$. $\mathbf{I}^{(i)}\mathbf{x}$ is the variables set with $n_i$ elements. $n_i$ represents the number of variables in sub-function $f_i$. Sub-function $f_i$ is a scalar function such that $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}$. $\alpha_i$ is a constant coefficient.

### 2.3. Partially/completely separable model

Based on the definition of additive/multiplicative separability, the new separable model with mixed binary operators, namely plus (+), minus (−), times (×), and division (÷) are defined as follows.

**Definition 2.** A scalar function $f(\mathbf{x})$ with $n$ continuous variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ $(f : \mathbb{R}^n \mapsto \mathbb{R},\ \mathbf{x} \in \mathbb{R}^n)$ is partially separable if and only if it can be rewritten as

$$f(\mathbf{x}) = \alpha_0 \otimes_1 \alpha_1 f_1\left(\mathbf{I}^{(1)}\mathbf{x}\right) \otimes_2 \alpha_2 f_2\left(\mathbf{I}^{(2)}\mathbf{x}\right) \otimes_3 \ldots \otimes_m \alpha_m f_m\left(\mathbf{I}^{(m)}\mathbf{x}\right), \tag{8}$$

Please cite this article as: C. Chen et al., Block building programming for symbolic regression, Neurocomputing (2017), https://doi.org/10.1016/j.neucom.2017.10.047