Information Sciences 430–431 (2018) 287–313

Contents lists available at ScienceDirect

Information Sciences

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

Automatic feature engineering for regression models with machine learning: An evolutionary computation and statistics hybrid

Vinícius Veloso de Melo[a,*], Wolfgang Banzhaf[b]

[a] Institute of Science and Technology (ICT), Federal University of São Paulo (UNIFESP), São José dos Campos, SP, Brazil
[b] Department of Computer Science and Engineering and BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48864, USA

ARTICLE INFO

Article history:
Received 15 April 2017
Revised 15 October 2017
Accepted 17 November 2017
Available online 21 November 2017

Keywords:
Feature engineering
Machine learning
Symbolic regression
Kaizen programming
Linear regression
Genetic programming
Hybrid

ABSTRACT

Symbolic Regression (SR) is a well-studied task in Evolutionary Computation (EC), where adequate free-form mathematical models must be automatically discovered from observed data. Statisticians, engineers, and general data scientists still prefer traditional regression methods over EC methods because of the solid mathematical foundations, the interpretability of the models, and the lack of randomness, even though such deterministic methods tend to provide lower quality prediction than stochastic EC methods. On the other hand, while EC solutions can be big and uninterpretable, they can be created with less bias, finding high-quality solutions that would be avoided by human researchers. Another interesting possibility is using EC methods to perform automatic feature engineering for a deterministic regression method instead of evolving a single model; this may lead to smaller solutions that can be easy to understand. In this contribution, we evaluate an approach called Kaizen Programming (KP) to develop a hybrid method employing EC and Statistics. While the EC method builds the features, the statistical method efficiently builds the models, which are also used to provide the importance of the features; thus, features are improved over the iterations resulting in better models. Here we examine a large set of benchmark SR problems known from the EC literature. Our experiments show that KP outperforms traditional Genetic Programming - a popular EC method for SR - and also shows improvements over other methods, including other hybrids and well-known statistical and Machine Learning (ML) ones. More in line with ML than EC approaches, KP is able to provide high-quality solutions while requiring only a small number of function evaluations.

1. Introduction

In a traditional regression task, one seeks to model the relationship between a dependent variable (the response) and one or more independent variables (also called explanatory variables). In a statistical regression approach, the practitioner employs a predetermined function f (or manually develops a variation) to combine the explanatory variables x in order to

* Corresponding author. .
E-mail addresses: vinicius.melo@unifesp.br, dr.vmelo@gmail.com (V. Veloso de Melo), banzhaf@msu.edu (W. Banzhaf).

https://doi.org/10.1016/j.ins.2017.11.041
0020-0255/

calculate output y:

$$y = f(x, \beta) + \epsilon, \tag{1}$$

where β is a set of parameters (constants, one for each variable), and ϵ is a measure of error. An optimization method has to optimize β to minimize the "lack of fit".

Symbolic Regression (SR), on the other hand, is a non-linear regression analysis technique that generates mathematical expressions to fit a given dataset. Being an optimization algorithm, SR optimizes the mathematical expressions according to some criterion, such as goodness-of-fit and/or expression complexity. Another particular aspect of SR is that it assumes no a priori model; nevertheless, one may be provided.

Therefore, an initial expression, or group of expressions, is randomly generated from the operand and operator sets provided by the user. Operands are the features of the dataset and other constants, such as π. Operators are the functions that generate data (random distributions, for instance) or functions to be applied to the operands (arithmetical, geometrical, etc.). As SR may start with random expressions, and usually there is no mechanism to avoid specific constructions, the algorithm is free to explore the search space of solutions. Thus, it may find high-quality models that would never be discovered by humans because the relationships among the variables could not make sense from a human perspective. Nonetheless, if necessary, domain knowledge and bias can be employed in grammar-based SR algorithms [34].

One may notice that SR is a more general, mixed-type problem, where not only the parameters β must be optimized but also an appropriate function f must be found. Therefore, SR must optimize both the model structure and its parameters, while traditional regression techniques optimize the parameters of a model supplied by the user. Clearly, SR solves a substantially more difficult problem and requires particular algorithms in order to work properly; consequently, many researchers are still looking for good heuristics to improve the search.

Over the last years, SR has been widely studied with Evolutionary Computation (EC) techniques able to produce computer code. As examples one may cite Genetic Programming (GP, [25,5]), Multi Expression Programming (MEP, [31]), Gene Expression Programming (GEP, [15]), Grammatical Evolution (GE, [34]), Linear Genetic Programming (LGP, [8]), Cartesian Genetic Programming (CGP, [29]), Behavioral Programming (BP, [26]), and Stack-based Genetic Programming (Stack-based GP, [32,39]). These methods evolve populations of individuals, each being a single model. Related non-EC techniques may also be found in the literature, for instance Fast Feature Extraction (FFX, [27]) and Prioritized grammar enumeration (PGE, [42]); different from the EC methods, these two are very successful in performing *feature engineering*, which is usually defined as a process of creating *relevant* features from the original features in the data in order to increase predictive power of the learning algorithm.

As previously stated, Evolutionary Computation is largely used for finding models composed of a single highly predictive feature. In EC methods, the differential survival of fitter solutions is one of the main ingredients. In most cases, a population of individuals is evolved to solve a particular task, where an individual represents a complete solution. Competition among the individuals is used to control evolution allowing the population to converge to the best individual. There is no guarantee of convergence to the optimum, of course, due to the stochasticity of the process, only of approximation. Thus, it is important to examine techniques that can provide better guidance in stochastic global optimization tasks such as SR.

An interesting proposal for better guidance is the Cooperative co-evolution algorithm (CCeA, [33]). It was proposed as a Genetic Algorithm extension to provide an environment where subcomponents could "emerge" and collaboration could automatically appear. CCeA is an evolutionary approach, therefore one expects that emergent behavior will occur. Over the years, some issues of this approach have been improved, e.g. larger populations than traditional Evolutionary Algorithms or multiple populations (a population for each subcomponent); the credit assignment problem; random selection of subcomponents for combination or based on their individual fitnesses (good subcomponents do not always produce good solutions when put together); among others. CCeAs have been applied with success in solving several tasks including SR [1,6,33,41].

De Melo [9] proposed Kaizen Programming (KP) to also search for collaboration among subcomponents. KP, different from CCeA, was proposed as an iterative approach focused on efficient problem-solving techniques that could come from Statistics, ML, Classical Artificial Intelligence (AI), Econometrics, or other related areas. For instance, KP has been used with Logistic Regression [12,10], CART decision tree [13], and Random Forests [36]. Also, a greedy approach was developed for solving a control problem known as the virtual Lawn Mower [37].

These techniques used by KP can be seen as powerful local optimizers that may need good starting points. For providing such points, KP searches the solution space through random search, recombination, variation, and sampling, among other methods. KP then uses those starting points as subcomponents and the local optimization techniques try to find the best combination of the subcomponents to get the highest-quality solution. Later, one can identify what was combined and the importance given by local optimizers to each subcomponent.

The application of KP to SR means that, instead of directly searching for a solution, KP will search for a *set* of features (mostly non-linear, but single features may be selected) for a known model; in this paper, a standard linear model optimized by Ordinary Least Squares. A relevant characteristic of such approach is the posterior use of other statistical tools for further feature and model selection (AIC, BIC, among others), to calculate prediction and confidence intervals, and to perform residual analysis, among others.

A known statistical approach related to KP for regression tasks is called basis expansion ([20], Chapter 5, Page 115):