Expert Systems With Applications 57 (2016) 76–90

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Model-driven regularization approach to straight line program genetic programming

José L. Montaña[a], César L. Alonso[b], Cruz E. Borges[c], Cristina Tîrnăucă[a,*]

[a] Universidad de Cantabria, Av. de los Castros s/n, Santander 39005, Spain
[b] Universidad de Oviedo, Calle San Francisco 1, Oviedo 33003, Spain
[c] Universidad de Deusto, Av. de las Universidades 24, Bilbao 48007, Spain

A R T I C L E   I N F O

Article history:
Received 6 November 2015
Revised 5 February 2016
Accepted 3 March 2016
Available online 19 March 2016

Keywords:
Genetic programming
Straight line program
Pfaffian operator
Symbolic regression

A B S T R A C T

This paper presents a regularization method for program complexity control of linear genetic programming tuned for transcendental elementary functions. Our goal is to improve the performance of evolutionary methods when solving symbolic regression tasks involving Pfaffian functions such as polynomials, analytic algebraic and transcendental operations like sigmoid, inverse trigonometric and radial basis functions. We propose the use of straight line programs as the underlying structure for representing symbolic expressions. Our main result is a sharp upper bound for the Vapnik Chervonenkis dimension of families of straight line programs containing transcendental elementary functions. This bound leads to a penalization criterion for the mean square error based fitness function often used in genetic programming for solving inductive learning problems. Our experiments show that the new fitness function gives very good results when compared with classical statistical regularization methods (such as Akaike and Bayesian Information Criteria) in almost all studied situations, including some benchmark real-world regression problems.

1. Introduction

Inductive inference (Angluin & Smith, 1983; Gori, Maggini, Martinelli, & Soda, 1998; Shaoning & Kasabov, 2004; Tenebaum, Griffiths, & Kemp, 2006) is one of the main fields in Machine Learning, and it can be defined as the process of hypothesizing a general rule from examples. The methods used in inductive inference span a variety of very different Machine Learning tools including neural networks, regression and decision trees, support vector machines, Bayesian networks, probabilistic finite state machines and many other statistical techniques. Given a sample set of an unknown process, the problem of finding a model capable to predict correct values for new examples has applications in a diversity of fields such as economics, electronic design, game playing, physical processes, etc. When dealing with real-world problems, the sample data are generally obtained through measures that are often corrupted by noise. In addition, the distribution according to which the examples are generated is usually unknown. This situation is very common when trying to solve inductive learning problems in the context of symbolic regression and must be considered and modeled in its whole complexity to produce reasonable models. Symbolic regression can be understood as the problem of finding a mathematical formula that fits a set of data. For a long time symbolic regression was a human task, but the advent of computers facilitated the exploration of the huge search space in which the regression process usually takes place. Genetic Programming (GP) can be seen as a symbolic regression strategy by means of evolutionary algorithms. This idea was proposed by Koza to whom both the term and the concept are due (see Koza, 1992).

In the last years, GP has been applied to a wide range of situations to solve both unsupervised and supervised learning problems, and it has become a powerful tool in the Knowledge Discovery and Data Mining domain (e.g., Freitas, 2002), including the emergent field of Big Data analysis (see Castelli, Vanneschi, Manzoni, & Popovič, 2015). Main subjects in unsupervised learning like clustering have been approached using GP (see Bezdek, Boggavarapu, Hall, & Bensaid, 1994; Falco, Tarantino, Cioppa, & Fontanella, 2004; Folino, Pizzuti, & Spezzano, 2008; Jie, Xinbo, & Li-cheng, 2003). Supervised classification by evolving selection rules is another avenue in which GP obtains a remarkable success as shown, for example, in Carreño, Leguizamón, and Wagner (2007), Cano, Herrera, and Lozano (2007), Chien, Yang, and Lin (2003), Freitas (1997), Hennessy, Madden, Conroy, and Ryder (2005) and Kuo, Hong, and Chen (2007). Singular

* Corresponding author. Tel.: +34942201529.
E-mail addresses: joseluis.montana@unican.es (J.L. Montaña), calonso@uniovi.es (C.L. Alonso), cruz.borges@deusto.es (C.E. Borges), cristina.tirnauca@unican.es (C. Tîrnăucă).

http://dx.doi.org/10.1016/j.eswa.2016.03.003
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

applications to medicine and biology problems (Aslam, Zhu, & Nandi, 2013; Bojarczuk, Lopes, & Freitas, 2000; Bojarczuk, Lopes, Freitas, & Michalkiewicz, 2004; Castelli, Vanneschi, & Silva, 2014), feature extraction methods (Krawiec, 2002; Smith & Bull, 2005), database clustering and rule extraction (Wedashwara, Mabu, Obayashi, & Kuremoto, 2015), generation of hybrid multi-level predictors for function approximation and regression analysis (Tsakonas & Gabrys, 2012) are other examples in which GP is applied. Specific applications to inductive learning problems solved via GP can be found in relatively old papers (Okley, 1994; Poli & Cagnoni, 1997).

The general procedure of GP consists in the evolution of a population of computer programs, each of them computing a certain function, with the aim of finding one that best describes the target function. These computer programs are build out from a set of functions and a set of terminals consisting of variables and constants. In the evolutionary process, the fitness function evaluates the goodness of each member of the population by measuring the empirical error over the sample set. In the presence of noise and to prevent other causes of overfitting (like, for example, the use of a very complex model), this fitness function must be regularized with some term that usually depends on the complexity of the model. Regularization is then a central problem related to generalization. By generalization we mean that the empirical error must converge to the expected true error when the number of examples increases. This notion of generalization defined here roughly agrees with the informal use of the term in GP (good performance over unseen examples) but captures the nature of the problem in the learning theory setting. The problem of generalization is focused for example in Tackett and Carmi (1994) and also in Cavaretta and Chellapilla (1999). In Keijzer and Babovic (2000), ensembles methods that can improve the generalization error in GP are discussed. The specific problem of regularization is extensively treated for polynomials in Nikolaev and Iba (2001), Nikolaev, de Menezes, and Iba (2002). Recent work regarding regularization used in GP can be found in Ni and Rockett (2015b), where a *vicinal risk* regularization technique is explored. A detailed description of other GP-regularization strategies can be found in Ni and Rockett (2015a), where Tikhonov regularization, in conjunction with node count as a general complexity measure in multiobjective GP, is proposed.

Most work describing GP strategies for solving symbolic regression problems employs trees as data structure for representing programs. There are other methodologies having an instruction-based approach, like that of matrix representation given in Li, Wang, Lee, and Leung (2008) or the more classical Cartesian Genetic Programming (CGP) of Miller and Thomson (2000). In this paper, we propose the use of straight line programs (SLPs) as data structure to evolve in GP. The SLP structure has a good performance when solving symbolic regression problem instances as shown in Alonso, Montana, and Puente (2008). The main difference between SLP GP and CGP consists in the implementation of the crossover operator, which in the case of SLPs is designed to interchange subgraphs, as described in Section 3. While GP-trees are expression trees of formulas, the SLPs correspond to expression dags (direct acyclic graphs) in which precomputed results can be reused. This makes the SLP data structure more compact and, usually, smaller than a tree, and consequently, easier to evaluate. Moreover, there is a canonical transformation mapping GP-trees onto SLPs (see Alonso, Montana, and Puente, 2009 for a detailed explanation of the relationships between SLPs and GP-trees). This transformation preserves the size and other complexity measures. For this reason, conclusions related to SLP performance can be extended to GP-tree performance at least in the context of generalization and regularization.

This paper is primarily concerned with statistical regularization methods which can be applied to a wide family of models. In particular we study a complexity measure for families of data structures representing programs named Vapnik Chervonenkis dimension (VCD). This measure, which belongs to the field of Information Theory, usually appears in conjunction with the Probably Approximately Correct approach to supervised learning, commonly referred to in the literature as the PAC model (see Vapnik, 1998 and Vapnik & Chervonenkis, 1974). We consider families of SLPs constructed from a set of Pfaffian functions (solutions of triangular systems of first order partial differential equations with polynomial coefficients - the formal definition is given in Section 6). As important examples, polynomials, exponential functions, trigonometric functions on some particular intervals and, in general, analytic algebraic functions are all Pfaffian. The main result of this paper is summarized in Theorem 1, where an upper bound for the VCD of a family of SLPs using Pfaffian functions is found. This upper bound is polynomial on some parameters, and in particular, on the non-scalar length of the SLPs. This implies that the length of the SLPs can be arbitrary if the excess is caused by addition and/or subtraction operations. Based on this result, we propose a method for selecting models in SLP GP for the case in which the admitted operations are all Pfaffian. In general, analyticity of the operators is a very desirable property in numerical computations, leading to more robust computer programs and providing a higher classification capacity. There are even well-founded proposals to replace the traditional protected division in GP by analytic operators, providing much better experimental results as shown in Ni, Drieberg, and Rockett (2013). This is another strong motivation for using Pfaffian operations.

The paper is organized as follows. Section 2 contains some basic concepts concerning statistical regression, the model selection criteria used in this paper, along with the general definition of the VCD of a family of sets. In Section 3 we describe the SLP data structure, and we introduce, via a technical lemma, the concept of an universal SLP. Section 4 includes the main traits of the SLP GP paradigm. In Section 5 we provide basic definitions for the Pfaffian operators, and we present a summary of previous technical results that are used in the proof of our main theorem, which can be found in Section 6. There, we give an upper bound for the VCD of families of SLPs that use as operators only Pfaffian functions. Section 7 shows the results of an extensive experimentation phase on a variety of symbolic regression problems. We also provide a comparative analysis between our method and two other well-known statistical regularization strategies, in which the complexity of the models is estimated by the number of free parameters. Finally, Section 8 contains some conclusions and future work.

2. Supervised learning and regression

Genetic Programming can be seen as a direct evolution method of computer programs for inductive learning. Inductive GP can be considered as a specialization of GP, in that it uses the framework of the last one in order to solve inductive learning problems. These problems are, in general, searching problems, where the aim is to find the best model from a finite set of observed data. Note that the best model might not be the one that perfectly fits the data, since this could lead to overfitting and, consequently, to a poor performance on unseen instances. The idea is to look for a model that fits well the data set, being at the same time as simple as possible. This immediately raises the question of how to measure the complexity of a model. There are many ways to do this. For example, we could prefer models with a small number of free parameters, which corresponds to simple mathematical formulas. Or, if the model is represented by a program, we could consider the length of the program as a complexity measure. Usually, for tree structures, a measure of the complexity is the height or the width of the tree. There is no universal way of measuring the complexity of