Genetic Programming Model Regularization

César L. Alonso, José Luis Montaña and Cruz Enrique Borges

Abstract We propose a tool for controlling the complexity of Genetic Programming models. The tool is supported by the theory of Vapnik-Chervonekis dimension (VCD) and is combined with a novel representation of models named straight line program. Experimental results, implemented on conventional algebraic structures (such as polynomials), show that the empirical risk, penalized by suitable upper bounds for the Vapnik-Chervonenkis dimension, gives a generalization error smaller than the use of statistical conventional techniques such as Bayesian or Akaike information criteria.

Keywords Genetic Programming · Straight Line Program · Pfaffian Operator · Symbolic Regression

1 Introduction

Inductive inference from examples is one of the most studied problems in Artificial Intelligence and has been addressed for many years using different techniques. Among them are included statistical methods such as inference techniques, regression and decision trees and other machine learning methods like neuronal networks and support vector machines [3, 12, 18, 20].

C.L. Alonso
Centro de Inteligencia Artificial, Campus de Gijón, Universidad de Oviedo,
33271 Gijón, Spain
e-mail: calonso@aic.uniovi.es

J.L. Montaña (✉)
Departamento de Matemáticas, Estad ística y Computación, Universidad de Cantabria,
39005 Santander, Spain
e-mail: montanjl@unican.es

C.E.Borges
DeustoTech (Energy Unit), Universidad de Deusto, 48007 Bilbao, Spain
e-mail: cruz.borges@alumnos.unican.es

K. Madani et al. (eds.), *Computational Intelligence*,
Studies in Computational Intelligence 613,
DOI 10.1007/978-3-319-23392-5_6

In the last two decades, genetic programming (GP) has been applied to solving problems of inductive learning with some remarkable success [15–17, 19]. The general procedure involves the evolution of populations of data structures that represent models for the target function. In the evolutive process, the fitness function for evaluating the population measures some empirical error between the empirical value of the target function and the value of the considered individual over the sample set. Usually this fitness function must be regularized with some term that depends on the complexity of the model. Identifying optimal ways to measure the complexity of the model is one of the main goals in the process of regularization.

Most of the work devoted to develop GP strategies for solving inductive learning problems makes use of the GP-trees as data structures for representing programs [14]. We have proposed a new data structure named straight line program (slp) to deal with the problem of learning by examples in the framework of genetic programming. The slp has a good performance in solving symbolic regression problem instances as shown in (see [2]). A slp consists of a finite sequence of computational assignments. Each assignment is obtained by applying some function (selected from a given set) to a set of arguments that can be variables, constants or pre-computed results. The slp structure can describe complex computable functions using a few amount of computational resources than GP-trees. The key point for explaining this feature is the ability of slp's for reusing previously computed results during the evaluation process. Another advantage with respect to trees is that the slp structure can describe multivariate functions by selecting a number of assignments as the output set. Hence one single slp has the same representation capacity as a forest of trees. We study the practical performance of ad-hoc recombination operators for slps and we apply the slp- based GP approach to regression. In addition we study the Vapnik-Chervonekis dimension of slps representing models. We consider families of slp's constructed from a set of Pfaffian functions. Pfaffian functions are solutions of triangular systems of first order partial differential equations with polynomial coefficients. As examples, polynomials, exponential functions, trigonometric functions on some particular intervals and, in general, analytic algebraic functions are Pfaffian. The main outcome of this work is a penalty term for the fitness function of a genetic programming strategy based on slp's to solve inductive learning problems. Experimental results point out that the slp structure, if suitably regularized, may result in a robust tool for supervised learning.

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 target is to construct some prediction model from a finite set of observed data. Providing a framework for studying inductive learning problems is one of the goals of Statistical

Learning Theory. In this sense the inclusion of methods from Statistical Learning Theory into the GP paradigm is a relevant contribution to inductive GP. Inductive inference process consists of three steps: to observe a phenomenon, to construct a model of that phenomenon and to make predictions by using this model. Given some sample set obtained by means of the observation step, it could seem that the best model might fit exactly the data, but this situation could lead to a poor performance on unseen instances in the presence of noise. Hence, the general 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 would prefer models with a small number of free parameters, which corresponds to simple mathematical formulas. In other cases where the model is represented by a program, we would 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 the model and the choice of a specific measure inherently depends on the problem at hand.

We will consider symbolic regression formulation under the general setting for predictive learning (see, [7, 21, 22]). The goal is to estimate an unknown real-valued function that fits a given finite sample set of data points. More formally, we consider an input space $X = \mathbb{R}^n$ and an output space $Y = \mathbb{R}$. We are given a sample of m pairs $z = (x_i, y_i)_{1 \leq i \leq m}$. These examples are drawn according to an unknown probability measure ρ on the product space $Z = X \times Y$ and they are generated according to an independent identically distributed (i.i.d.) process. As usual, probability measure ρ factorizes through its marginal distribution in X, $\rho(x)$, and the conditional distribution in Y, $\rho(y|x)$, that is:

$$\rho(x, y) = \rho(x)\rho(y|x) \tag{1}$$

The goal is to construct a function $f : X \to Y$ which predicts the value $y \in Y$ from a given $x \in X$. To choose the function f we use a criterion of a low probability of error. The best estimation of the function is the mean of the output conditional probability:

$$g(x) = \int y\rho(y|x) \tag{2}$$

A learning method selects the best model $f \in \mathcal{H}$, where $\mathcal{H}$ is some class of functions. In general, the error of the estimator f, $\varepsilon(f)$, is written as

$$\varepsilon(f) = \int Q(x, f, y)d\rho, \tag{3}$$

where Q measures some notion of loss between $f(x)$ and the target value y, and ρ is the distribution from which examples (x, y) are drawn to the learner. For regression tasks one usually takes $Q(x, f, y) = (y - f(x))^2$.

For a class of functions $\mathcal{H}$ of finite bounded complexity (for instance trees with bounded size or height), the model can be chosen minimizing the empirical error, also known as empirical risk:

$$\varepsilon_m(f) = \frac{1}{m}\sum_{i=1}^{m} Q(x_i, f, y_i) \tag{4}$$

Obviously, this method will have a good performance when the optimal model belongs to the complexity bounded class of functions considered. Nevertheless, usually we are not able to make such an assumption and a large class $\mathcal{H}$ must be considered. In this case, the problem of regression estimation requires optimal selection of model complexity in addition to model estimation via minimization of the empirical risk.

Analytical models selection criteria estimate the real error (Eq. 3) as a function of the empirical error (Eq. 4) with a penalty term related with some measure of model complexity:

$$\varepsilon(f) = \varepsilon_m(f) * pen(h, m); \quad * \in \{+, \cdot\} \tag{5}$$

where f is the model, h is the model complexity and m is the size of the sample set. In the above equation, there exists a degree of freedom which is the selection of the measure h for the model complexity. This measure always depends on the considered class $\mathcal{H}$ and more exactly on the representation structure for the models in $\mathcal{H}$. For example if the functions are described by multivariate polynomials, h is usually the number of monomials or a linear function involving the degree of the polynomial. In other cases, when the elements of $\mathcal{H}$ are represented by programs or trees, some typical complexity measures are the length of the programs or the size of the trees.

In this work, as we will see in the next sections, the classes $\mathcal{H}$ are families of programs named straight line programs, that are constructed from a set of operators F and a set of terminals T. The elements of F are Pfaffian functions, that is, a more general class of functions than polynomials or rational functions. Pfaffian functions include, for example, the analytic algebraic functions as for instance square root extraction.

Motivated by the concept of degree of polynomials, in our programs we will only consider the non-scalar instructions for measuring the complexity of the model. The non-scalar instructions are those in which the selected operator in F is different from $\{+, -\}$. The main theoretical result in this work is the computation of a polynomial upper bound for a new complexity measure of our straight line programs with Pfaffian instructions, that does not involve the length of the corresponding program, but only the number of the non-scalar instructions. A simplification of this complexity measure will be considered in the Eq. (5) for the fitness regularization in GP with our structure.

3 Straight Line Program Genetic Programming

Straight line programs have a large history in the field of Computational Algebra. A particular class of straight line programs, known in the literature as arithmetic circuits, constitutes the underlying computation model in Algebraic Complexity Theory [6]. Arithmetic circuits with the standard arithmetic operations $\{+, -, *, /\}$ are the natural model of computation for studying the computational complexity of algorithms solving problems with an algebraic flavor. They have been used in linear algebra problems [4], in quantifier elimination [13] and in algebraic geometry [9, 10]. Also, slp's constitute a promising alternative to the trees in the field of Genetic Programming (see [2]). The formal definition of the straight line program structure is as follows: Let $F = \{f_1, \ldots, f_n\}$ be a set of functions, where f_i has arity a_i, for $1 \leq i \leq n$, and let $T = \{t_1, \ldots, t_m\}$ be a set of terminals. A *straight line program (slp)* over F and T is a finite sequence of computational instructions $\Gamma = \{I_1, \ldots, I_l\}$ where

$$I_k \equiv u_k := f_{j_k}(\alpha_1, \ldots, \alpha_{a_{j_k}}); \text{ with } f_{j_k} \in F,$$

$\alpha_i \in T$ for all i if $k = 1$ and $\alpha_i \in T \cup \{u_1, \ldots, u_{k-1}\}$ for $1 < k \leq l$.

Terminal set T is of the form $T = V \cup C$, where $V = \{x_1, \ldots, x_n\}$ is a finite set of variables and $C = \{c_1, \ldots, c_q\}$ is a finite set of constants. The number of instructions l is the *length* of Γ.

Note that if we consider the slp Γ as the code of a program, then a new variable u_i is introduced at each instruction I_i. We will denote by $\Gamma = \{u_1, \ldots, u_l\}$ a slp. Each of the non-terminal variables u_i can be considered as an expression over the set of terminals T constructed by a sequence of recursive compositions from the set of functions F. The set of all slp's over F and T is denoted by $SLP(F, T)$.

An output set of a slp $\Gamma = \{u_1, \ldots, u_l\}$ is any set of non-terminal variables of Γ, that is, $O(\Gamma) = \{u_{i_1}, \ldots, u_{i_t}\}$, $i_1 < \cdots < i_t$. Provided that $V = \{x_1, \ldots, x_p\} \subset T$ is the set of terminal variables, the function computed by Γ, denoted by $\Phi_\Gamma : I^p \rightarrow O^t$, is defined recursively in the natural way and satisfies $\Phi_\Gamma(a_1, \ldots, a_p) = (b_1, \ldots, b_t)$, where b_j stands for the value of the expression over V of the non-terminal variable u_{i_j} when we replace the variable x_k with a_k; $1 \leq k \leq p$.

Example 1 Let F be the set given by the three binary standard arithmetic operations, $F = \{+, -, *\}$ and let $T = \{1, x_1, x_2\}$ be the set of terminals. In this situation any slp over F and T is a finite sequence of instructions where each instruction represents a polynomial in two variables with integer coefficients. If we consider the following slp Γ of length 5 with output set $O(\Gamma) = \{u_5\}$:

$$\Gamma \equiv \begin{cases} u_1 := x_1 + 1 \\ u_2 := u_1 * u_1 \\ u_3 := x_2 + x_2 \\ u_4 := u_2 * u_3 \\ u_5 := u_4 - u_3 \end{cases} \tag{6}$$

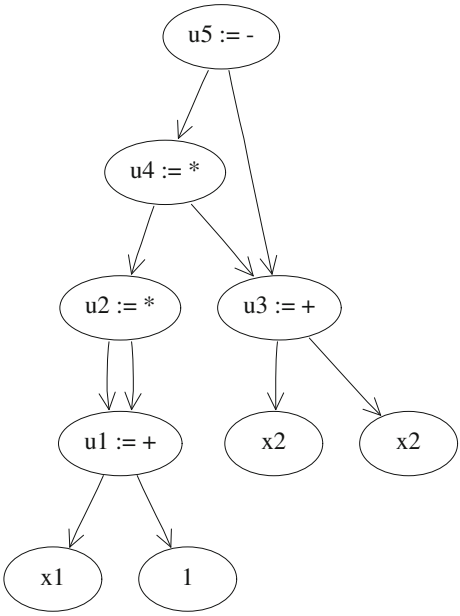

Fig. 1 Directed graph representing a slp

then the function computed by Γ is the polynomial

$$\Phi_\Gamma = 2x_2(x_1 + 1)^2 - 2x_2$$

Every slp, $\Gamma = \{u_1, \ldots, u_l\}$, over F and T can be represented by a directed acyclic graph $G_\Gamma = (V, E)$. The set of vertices is $V = T' \cup \{u_1, \ldots, u_l\}$, where T' contains all terminals involved in the computation. The set of edges E is constructed as follows: for every k, $1 \leq k \leq l$, we draw an edge (u_k, α_i) for each $i \in \{1, \ldots, a_{j_k}\}$. Note that T' is the set of leaves of G_Γ and it is a subset of the set T of terminals. Figure 1 is a directed graph representing the slp described in Eq. (6)

For computing the initial population, the well known methods for trees (see [14]) can be easily adapted to slp's. In order to compute the fitness function in a GP process to solve a particular problem, the computation of the function Φ_Γ, considering its own definition, would be often necessary.

3.1 SLP Crossover and Mutation

For slp-GP, 1-point crossover and in general k-point crossover are easily defined. However, a new specific crossover operation that produces another type of information exchange between the two selected parents, has been designed. The objective is to carry subexpressions from one parent to the other. A subexpression is captured by

an instruction u_i and all the instructions that are used to compute the expression over the set of terminals represented by u_i. Now follows a description of this crossover with a clarifying example.

Given two slp's, Γ_1 and Γ_2, first a position k en Γ_1 is randomly selected. Let S_{u_k} be the set of all instructions related to the computation of the subexpression associated to node u_k. We obtain the first offspring by randomly selecting an allowed position t in Γ_2 and making the substitution of a part of its instructions by those instructions in S_{u_k} suitably renamed. For the second offspring we symmetrically repeat the strategy.

Example: Consider $F = \{*, +\}$, $L = 5$ and $T = \{x, y\}$. Let Γ_1 and Γ_2 be the following two slp's:

$$\Gamma_1 \equiv \begin{cases} u_1 := x + y \\ u_2 := u_1 * u_1 \\ u_3 := u_1 * x \\ u_4 := u_3 + u_2 \\ u_5 := u_3 * u_2 \end{cases} \qquad \Gamma_2 \equiv \begin{cases} u_1 := x * x \\ u_2 := u_1 + y \\ u_3 := u_1 + x \\ u_4 := u_2 * x \\ u_5 := u_1 + u_4 \end{cases}$$

If $k = 3$ then $S_{u_3} = \{u_1, u_3\}$, and t must be selected in $\{2, \ldots, 5\}$. If for instance $t = 3$, then the first offspring is as follows.

$$\Gamma_1' \equiv \begin{cases} u_1 := x * x \\ \mathbf{u_2} := \mathbf{x} + \mathbf{y} \\ \mathbf{u_3} := \mathbf{u_2} * \mathbf{x} \\ u_4 := u_2 * x \\ u_5 := u_1 + u_4 \end{cases}$$

For the second offspring, if the selected position in Γ_2 is $k' = 4$, then $S_{u_4} = \{u_1, u_2, u_4\}$. Now if $t = 5$, then the offspring will be

$$\Gamma_2' \equiv \begin{cases} u_1 := x + y \\ u_2 := u_1 * u_1 \\ \mathbf{u_3} := \mathbf{x} * \mathbf{x} \\ \mathbf{u_4} := \mathbf{u_3} + \mathbf{y} \\ \mathbf{u_5} := \mathbf{u_4} * \mathbf{x} \end{cases}$$

The mutation operation in the slp structure consists of a change in one of the instructions. This change can be either the substitution of the complete instruction by another one randomly generated, or a little modification of just one of the arguments of the function in F that defines the instruction.

4 Pfaffian Functions and VCD of Formulas

In this section we introduce some tools concerning the geometry of sets defined by boolean combinations of sign conditions over Pfaffian functions (semi-Pfaffian sets in the mathematical literature). A complete survey on the subject is due to Gabrielov and Vorobjov [8].

Definition 1 Let $U \subset \mathbf{R}^n$ be an open domain. A *Pfaffian chain* of length $q \geq 1$ and degree $D \geq 1$ in U is a sequence of real analytic functions $f_1, \ldots, f_q$ in U satisfying a system of differential equations

$$\frac{\partial f_i}{\partial x_j} = P_{i,j}(\mathbf{x}, f_1(\mathbf{x}), \ldots, f_i(\mathbf{x})) \tag{7}$$

for $i = 1, \ldots q$ where $P_{i,j} \in \mathbf{R}[\mathbf{x}, y_1, \ldots, y_i]$ are polynomials of degree at most D and $\mathbf{x} = x_1, \ldots, x_n$.

A function f on U is called a Pfaffian of order q and degree (D, d) if

$$f(\mathbf{x}) = P(\mathbf{x}, f_1(\mathbf{x}), \ldots, f_q(\mathbf{x})) \tag{8}$$

where $P \in \mathbf{R}[\mathbf{x}, y_1, \ldots, y_q]$ is a polynomial of degree at most $d \geq 1$ and $f_1, \ldots, f_q$ is a Pfaffian chain of length q and degree D.

The following functions are Pfaffian: $sin(x)$, defined on the interval $(-\pi + 2\pi r, \pi + 2\pi r)$; $tan(x)$, defined on the interval $(-\pi/2 + \pi r, \pi/2 + \pi r)$; e^x defined in $\mathbf{R}$; $log\ x$ defined on $x > 0$; $1/x$ defined on $x \neq 0$. $\sqrt{x}$ defined on $x \geq 0$. More generally, analytic algebraic functions are Pfaffian.

Definition 2 Let $\mathcal{F}$ be a class of subsets of a set X. We say that $\mathcal{F}$ shatters a set $A \subset X$ if for every subset $E \subset A$ there exists $S \in \mathcal{F}$ such that $E = S \cap A$. The VCD of $\mathcal{F}$ is the cardinality of the largest set that is shattered by $\mathcal{F}$.

Next we announce an upper bound for the VCD of a family of concept classes whose membership tests are computed by straight line programs involving Pfaffian operators over the real numbers. An important new issue is that we do not consider an upper bound for the length of the slp's. In previous results about VCD of programs or families of computation trees, a time bound approximated by the number of steps of the program execution or by the height of the computation tree is needed [11]. In our case we only need a bound for the number of the non-scalar slp's instructions. Those are instructions involving operations which are not in $\{+, -\}$.

A rough estimation of the VC dimension of slps using Pfaffian operators can be obtained computing the number of free parameters in families of slps with bounded non-scalar complexity. To do this let $T = \{t_1, \ldots, t_n\}$ be a set of terminals and let $F = \{+, -*, /, sign\} \cup \{f_1, \ldots, f_q\}$ be a set of functions, where the elements f_i constitute a Pfaffian chain of length q with arities bounded by A and the $sign$ function is defined as $sign(x) = 1$ if $x > 0$ and 0 otherwise.

Let $\Gamma_{n,L}$ be the collection of $slp's$ Γ over F and T using at most L non-scalar operations and a free number of scalar operations. Then, the number of free parameters of a universal slp Γ_U that parameterizes the elements of the family $\Gamma_{n,L}$ is exactly:

$$N := L[3 + q + A(n + \frac{L-1}{2}) + 1] + n \tag{9}$$

The proof is as follows. Introduce a set of parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ taking values in Z^k for a suitable natural number k, such that each slp in the family can be obtained specializing the parameters. For this purpose we define $u_{-n+m} = t_m$, for $1 \leq m \leq n$. Note that any non-scalar assignment u_i, $1 \leq i \leq L$ in a slp Γ belonging to $\Gamma_{n,L}$ is a function of $t = (t_1, \ldots, t_n)$ that can be parameterized as follows:

$u_i = U_i(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})(t) =$

$$\gamma^i_{-n}[\alpha^i_{-n}(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j) * (\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_2} u_j) +$$

$$+(1 - \alpha^i_{-n})[\beta^i_{-n} \frac{\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j}{\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_2} u_j} +$$

$$+(1 - \beta^i_{-n}) sgn(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j)]] +$$

$$+(1 - \gamma^i_{-n})[\sum_{k=1}^{q} \gamma^i_k f_k(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j, \ldots, \sum_{j=-n+1}^{i-1} \alpha_j{}^{i_A} u_j)]$$

Now considering the last assignment as the output set of the slp Γ, this last assignment is parameterized as:

$$U = \sum_{j=-n+1}^{L} \alpha_j u_j$$

where u_j, $1 \leq j \leq L$ are the non-scalar assignments.

Finally counting the number of introduced parameters we will obtain Eq. (9). The estimation given in that equation can be converted, after certain algebraic manipulations, into an upper bound using theory of Pfaffian operators (see [8]). We omit the proof due to lack of space.

Main Theorem. Let $\Gamma_{L,n}$ the set of slps with n variables, at most L non-scalar operations, using operators in F that contains the operations $\{+, -, *, /, sign\}$ and Pfaffian operations f, where each f belongs to a fixed Pfaffian chain $\{f_1, \ldots, f_q\}$ of

length q and degree $D \geq 2$. Let N be as in Eq. (3). Then, the Vapnik-Chervonenkis dimension of $\mathcal{C}_{k,n}$ is in the class:

$$O((q(N+n))^2 + (N+n)(L+q)log_2((N+n)(L+1)(4+D))) \tag{10}$$

Simplification. If we consider as constants parameters n, q, D and d, the VCD of the class is at most $O(L^4)$. This quantity gives an idea of the asymptotic maximum order of VCD of common classes of GP-models. We point out that this quantity is an upper bound and, possibly, far from being an optimal bound, but it can be used as starting point in further experimental developments.

5 Model Selection Criterion

In supervised learning problems like regression and classification a considerable amount of effort has been done for obtaining good generalization error bounds. The results by Vapnik (see [22]) state the following error bound:

$$\varepsilon(f) \leq \varepsilon_m(f) + \sqrt{\frac{h(log(2m/h)+1) - log(\eta/4)}{m}}, \tag{11}$$

where h must be substituted by the upper bound of the VCD of the hypothesis class that contains the model f, η is the probability that the error bound is violated and m is the sample size. As usual in this context $\varepsilon(f)$ and $\varepsilon_m(f)$ stand, respectively, for the true mean square error and the empirical mean square error of the model f.

In our case, f will be represented by a straight line program $\Gamma \in SLP(F,T)$ where T contains n variables and F contains the operations on real numbers $\{+,-,*,/,sign\}$ and Pfaffian operations over the reals. Note that the sets F and T are invariants throughout the model selection process. Hence, the search space of models forms a nested structure:

$$C_1 \subset C_2 \subset \cdots \subset C_L \subset \cdots$$

where C_L represents the class of slp's in $SLP(F,T)$ that have at most L non-scalar instructions. In this situation we will finally choose the model that minimizes the right side of Eq. (5).

6 Experimental Results

In this section we present the obtained results after an experimental phase in which symbolic regression problem instances were solved using the selection criterion described in the previous section. Our proposal is to consider straight line programs

with Pfaffian instructions as the structure that represents the model. Then a GP algorithm is executed considering the recombination operators for slp's described in Sect. 2 and with fitness regularization function expressed in Eq. (11). So we propose a model estimation via structural risk minimization (SRM). For the complexity measure h of the model, we will use the VCD bound in 4.

We will consider additive gaussian noise in the sample set $z = (x_i, y_i)_{1 \leq i \leq m}$. Hence, for a target function g, the sample set verifies: $y_i = g(x_i) + \epsilon$, where ϵ is independent and identically distributed (i.i.d.) zero mean random error.

We will compare the effectiveness of the VCD fitness regularization method (VCD-SRM) with two well known representative statistical methods with different penalization terms:

- Akaike Information Criterion (AIC) which is as follows (see [1]):

$$\varepsilon(f) = \varepsilon_m(f) + \frac{2h}{m}\sigma^2 \tag{12}$$

- Bayesian Information Criterion (BIC) (see [5]):

$$\varepsilon(f) = \varepsilon_m(f) + (ln\ m)\frac{h}{m}\sigma^2 \tag{13}$$

In the above expressions h stands for the number of free parameters of the model (Eq. 9).

For measuring the quality of the final selected model, we have considered a new set of unseen points, generated without noise from the target function. This new set of examples is known as the test set or validation set. So, let $(x_i, y_i)_{1 \leq i \leq n_{test}}$ a validation set for the target function $g(x)$ (i.e. $y_i = g(x_i)$) and let $f(x)$ be the model estimated from the training data. Then the prediction risk $\varepsilon_{n_{test}}$ is defined by the mean square error between the values of f and the true values of the target function g over the validation set:

$$\varepsilon_{n_{test}} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (f(x_i) - y_i)^2 \tag{14}$$

For the first experiment we have considered a set of 500 multivariate polynomials with real coefficients whose degrees are bounded by 5. The number of variables varies from 1 to 5 with 100 polynomials for each case.

A second experiment was performed considering some well known real benchmark problems. In all cases, when the GP process finishes, the best individual is selected as the proposed model for the corresponding target function.

We shall denote the set of polynomials as $\mathrm{P_n}^R[X]$ with $X = (x_1, \ldots, x_n)$, $1 \leq n \leq 5$ and $x_i \in [-1, 1]\ \forall i$. The individuals are slp's over $F = \{+, -, *, /, sqrt, sin, cos, exp\}$. In order to avoid errors generated by divisions by zero, instead of the traditional division we will use in our computation the operation usually named "protected division", that returns 1 if the denominator is zero. Besides the variables x_i, the terminal set also includes five constants c_i, $1 \leq i \leq 5$, randomly generated

in $[-1, 1]$. Observe that although the target functions are polynomials, our set F not only contains the operators of sum, difference and product, but also contains other Pfaffian functions. This situation increments considerably the search space. Nevertheless, note that in a real problem situation usually we do not know if the target function is a polynomial or not.

The parameters for the GP process are the following: population size $M = 200$, probability of crossover $p_c = 0,9$, probability of mutation $p_m = 0,05$, and tournament selection of size 5. The real length of the slp's in the population is bounded by 40. Elitism and a particular generational replacement are used. In this sense, the offsprings do not necessarily replace their parents. After a crossover we have four individuals: two parents and two offsprings. We select the two best individuals with different fitness values. The motivation is to prevent premature convergence and to maintain diversity in the population.

As we are considering multivariate polynomials as target functions, the difficulty of the problem instance increases with the number of variables. Hence, to vary the size of the sample set as a function of the number of variables is a reasonable decission. Note that an upper bound for the number of monomials in a polynomial with n variables and degree d is $4 \cdot d^{n+1}$ and this is also a quite good estimation for a lower bound of the size of the sample set. Thus, in our case we have considered sample sets of size $4 \cdot 5^{n+1}$, $1 \leq n \leq 5$. In this experiment one execution for each strategy has been performed over the 500 generated target functions. In every execution the process finishes after 250 generations were completed. Finally, the validation set consists of a number of unseen points that is equal to two times the size of the sample set (Table 1).

Figures 2 and 3 represent the empirical distribution of the executions of the three compared strategies over the sets of polynomials. We have separated the polynomial sets by the number of variables, from one to five. These empirical distributions are displayed using standard box plot notation with marks at the best execution, 25 %,

Table 1 Values of means and variances

$P_1{}^R[X]$	μ	σ	$P_2{}^R[X]$	μ	σ
AIC	0.39	0.42	AIC	1.17	0.86
BIC	0.38	0.41	BIC	1.25	0.85
VCD	0.24	0.27	VCD	0.82	0.54
$P_3{}^R[X]$	μ	σ	$P_4{}^R[X]$	μ	σ
AIC	2.62	1.28	AIC	4.79	1.49
BIC	2.89	1.57	BIC	5.08	1.74
VCD	2.24	0.85	VCD	4.76	1.43
$P_5{}^R[X]$	μ	σ			
AIC	8.51	2			
BIC	8.63	2.16			
VCD	8.63	2.17			

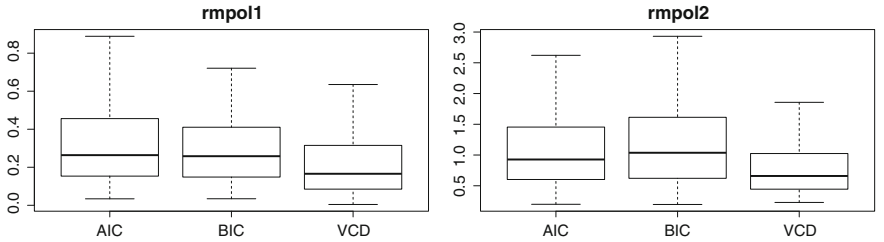

Fig. 2 Empirical distribution of the executions, for the univariate and bivariate polynomials

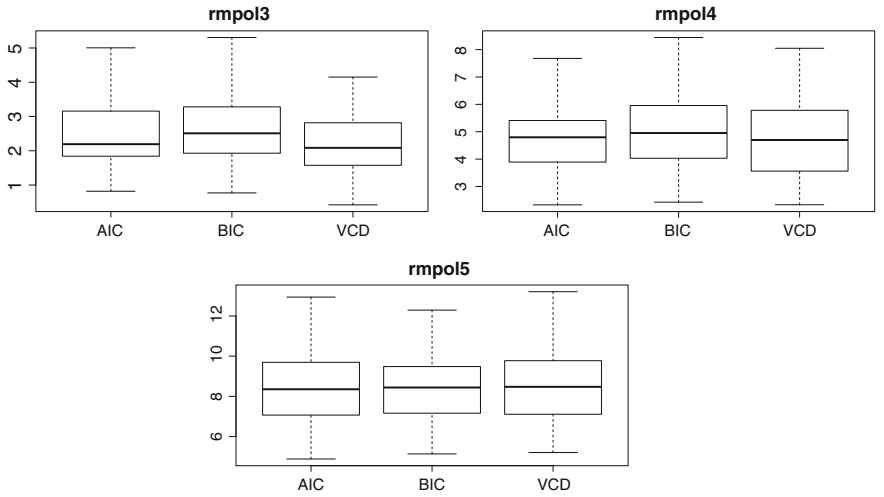

Fig. 3 Empirical distributions of the executions, for the multivariate polynomials with 3, 4, and 5 variables

50 %, 75 % and the worst execution, always considering the prediction risk of the selected model, represented on the y-axis and defined by the mean square error between the values of the model and the true values of the target function over the validation set. We also include tables that show means and variances as well as the prediction risk of the best obtained model for each method.

As we can see from the above figures and tables, it seems that VCD regularization performs better than the well known regularization methods AIC and BIC. This is more clear for the polynomials up to three variables and not so clear for the rest of the polynomial sets. This could be because for polynomials with four and five variables, as they constitute more complex problem instances, it would be necessary a large number of generations in the evolutive process. In order to confirm the comparative results of the studied strategies we have made crossed statistical hypothesis tests. The obtained results are showed in Table 2. Roughly speaking, the null-hypothesis in each test with associated pair (i, j) is that strategy i is not better than strategy j.

Table 2 Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies

$P_1{}^R[X]$	AIC	BIC	VCD
AIC		0.91	1
BIC	0.30		1
VCD	$3.93 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	
$P_2{}^R[X]$	AIC	BIC	VCD
AIC		$3.91 \cdot 10^{-2}$	1
BIC	0.96		1
VCD	$1.86 \cdot 10^{-5}$	$1.12 \cdot 10^{-7}$	
$P_3{}^R[X]$	AIC	BIC	VCD
AIC		$7.7 \cdot 10^{-2}$	0.99
BIC	0.99		1
VCD	$1.2 \cdot 10^{-2}$	$3.93 \cdot 10^{-4}$	
$P_4{}^R[X]$	AIC	BIC	VCD
AIC		0.12	0.90
BIC	0.77		0.99
VCD	0.27	$2.7 \cdot 10^{-3}$	
$P_5{}^R[X]$	AIC	BIC	VCD
AIC		0.44	0.52
BIC	0.77		0.78
VCD	0.61	0.37	

Table 3 Prediction risk of the model obtained from the best execution

Instance	AIC	BIC	VCD
$P_1{}^R[X]$	$3.40 \cdot 10^{-2}$	$3.44 \cdot 10^{-2}$	$4.32 \cdot 10^{-3}$
$P_2{}^R[X]$	0.20	0.19	0.22
$P_3{}^R[X]$	0.82	0.77	0.42
$P_4{}^R[X]$	2.33	2.43	2.33
$P_5{}^R[X]$	4.88	5.13	5.20

Hence if value a_{ij} in Table 2 is less than a significance value α, we can reject the corresponding null-hypothesis (Table 3, Fig. 4).

Taking into account the results of the crossed statistical hypothesis tests with a significance value $\alpha = 0.05$, we can confirm that our proposed regularization method based on the VC dimension of families of SLP's is the best of the studied strategies for the considered sets of multivariate polynomials.

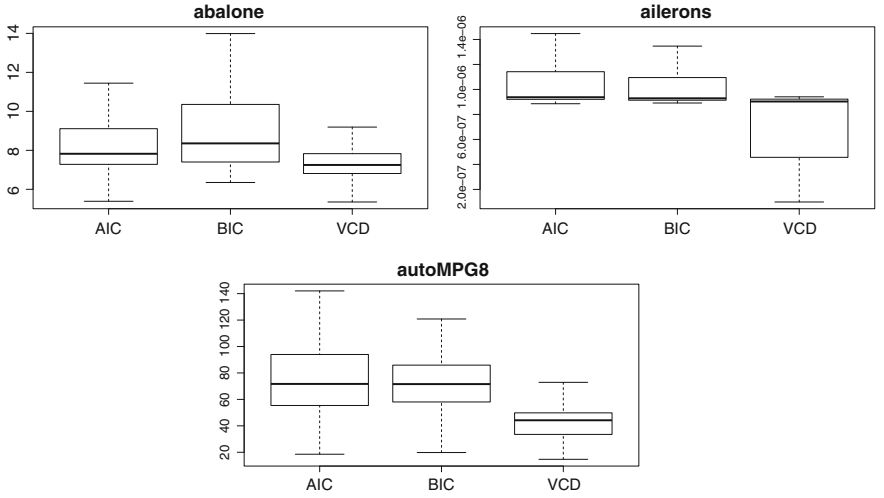

Fig. 4 Empirical distribution of the executions, for functions associated to real problems

7 Conclusions

Straight line programs constitute a promising structure for representing models in the Genetic Programming framework. Indeed, as it was published in a previous work, slp's outperform the traditional tree structure when GP strategies are applied for some kind of regression problems. In this paper we try to control the complexity of populations of slp's while they evolve in order to find good models for solving symbolic regression problem instances. The evolving structure is constructed from a set of functions that contains Pfaffian operators. We have considered the Vapnik Chervonenkis dimension as a complexity measure and we have found a theoretical upper bound of the VCD of families of slp's over Pfaffian operators as an important generalization of similar results for more simple sets of operators including rational functions. This theoretical upper bound is polynomial in the number of the non-scalar instructions of the family of the slp's. As a consequence of the main result, we propose a regularized fitness function included in a evolutionary strategy for solving symbolic regression problem instances. We have compared our fitness function based on the VCD upper bound with two well known statistical penalization criteria. The experimental results obtained after the execution of the compared strategies over two different groups of target functions, show that our proposed complexity measure and its corresponding penalization criterion is better than the others in over the group of the three real problem instances where the VCD regularization method is clearly the best.

Acknowledgments This work is partially supported by spanish grant TIN2011-27479-C04-04.

References

1. Akaike, H.: Statistical prediction information. Ann. Inst. Stat. Math. **22**, 203–217 (1970)
2. Alonso, C.L., Montaña, J.L.. Puente, J.: Straight line programs: a new linear genetic programming approach. In:Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 571–524 (2008)
3. Angluin, D., Smith, C.H.: Inductive inference: theory and methods. ACM Comput. Surv. **15**(3), 237–569 (1983)
4. Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. Inf. Process. Lett. **18**, 147–150 (1984)
5. Bernadro, J., Smith, A.: Bayesian Theory. Willey, New York (1994)
6. Burguisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Springer, New York (1997)
7. Cherkassky, V., Yunkian, M.: Comparison of model selection fo regression. Neural Comput. **15**(7), 1691–1714 (2003)
8. Gabrielov, A.N., Vorobjov, N.: Complexity of Computations with Pfaffian and Noetherian Functions, Normal Forms, Bifurcations and Finiteness Problems in Differential Equations. Kluwer, Dordrecht (2004)
9. Giusti, M., Heinz, J.: La Détermination des Points Isolés et la Dimension dúne Varieté Agebrique Peut se Faire en Temps Polynomial, Computational Algebraic Geometry and Commutative Algebra, Symposia Matematica XXXIV, Eisenbud, D, Robbiano, L. (eds.), pp. 216–256. Cambridge University Press, Cambridge (1993)
10. Giusti, M., Heintz, J., Morais, J., Morgentern, J.E., Pardo, L.M.: Straight line programs in geometric elimination theory. J. Pure Appl. algebra **124**, 121–146 (1997)
11. Goldberg, P., Jerrum, M.: Bounding the vapnik-chervonenkis dimension of concept classes parametrizes by real numbers. Mach. Learn. **18**, 131–148 (1995)
12. Gori, M., Maggini, M., Martinelli, E., Soda, G.: Inductive inference from noisy examples using the hybrid finite state filter. IEEE Trans. Neural Networks **9–3**, 571–575 (1998)
13. Heintz, J., Roy, M.F., Solerno, P.: Sur la Complexité du Principe de Tarski-Seidenberg. Bulletin de la Societé Mathematique de France **118**, 101–126 (1990)
14. Koza, J.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
15. Nikolaev, N.Y., Iba, H.: Regularization approach to inductive genetic programming. IEEE Trans. Evol. Comput. **5**(4), 359–375 (2001)
16. Okley, H.: In: Kinnear, K. (ed.) Advances in Genetic Programming. Two scientific applications of Genetic Programming: Stack filters and nonlinear fitting to chaotic data, pp. 369–389. MIT Press, Cambridge (1994)
17. Poli, R., Cagnoni, S.: In: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L. (eds.) Evolution of Pseudo-coloring Algoritms for Image Enhancement with Interactive Genetic Programming, pp. 269–277. MIT Press, Cambridge (1997)
18. Shaoning, P., Kasabov, N.: Inductive vs transductive inference. global vs local models: SVM, TSVM and SVMT for gene expression classification problems. In: Proceedings IEEE International Joint Conference on Neural Networks, vol. 2, pp. 1197–1202 (2004)
19. Tackett, W.A., Carmi, A.: In: Kinnear, K. (ed.) The Donut Problem: Scalability and Generalization in Genetic Programming, Advances in Genetic Programming. MIT Press, Cambridge (1994)
20. Tenebaum, J.B., Griffiths, T.L., Kemp, C.: Theory Based Bayesian Models of Inductive Learning and Reasoning. Trends in Cognitive Sciences, Kingston, vol. 10(7), pp. 309–318 (2006)
21. Vapnik, V., Chervonenkis, A.: Ordered risk minimization. Autom. Remote Control **34**, 1226–1235 (1974)
22. Vapnik, V.: Statistical Learning Theory. Willey, New York (1998)