

A Method for Regularization of Evolutionary Polynomial Regressions

Francisco Coelho^{a,c,*}, João Pedro Neto^{b,c}

^a*Dept. Informática, Universidade de Évora, Rua Romão Ramalho 58, 7000-671 Évora*

^b*Dept. Informática, Faculdade de Ciências da Universidade de Lisboa, Campo Grande
1749-016 Lisboa*

^c*Laboratory of Agent Modelling (LabMAg)*

Abstract

While many applications require models that have no acceptable linear approximation, the simpler nonlinear models are defined by polynomials. The use of genetic algorithms to find polynomial models from data is known as Evolutionary Polynomial Regression (EPR). This paper introduces Evolutionary Polynomial Regression with Regularization (EPRR), an algorithm that extends the EPR method and describes a set of experiences on common datasets that compare both flavors of EPR and other methods including Linear Regression, Regression Trees and Support Vector Regression.

The empiric conclusion of those experiments is that EPRR is able to achieve better fitting than other non-ensemble methods and it has shorter computation time than plain EPR.

Keywords: evolutionary polynomial regression, regularization, feature extraction, dimensionality reduction

*Corresponding author

Email addresses: `fc@di.uevora.pt` (Francisco Coelho), `jpn@di.fc.ul.pt` (João Pedro Neto)

1. Introduction

With notable exceptions (*e.g.* neural networks) machine learning regression techniques produce linear models. The linearity assumption has many advantages including reduced computational complexity and strong theoretical framework. However nonlinearity is unavoidable in many application scenarios, specially those with phase transitions or feedback loops, so common in engineering, ecology, cybernetics and other areas. The kernel trick in Support Vector Machines (SVM) (???) alleviates this problem by allowing special non-linear transformations of the feature-space. The condition such transformations must meet is known as the *kernel trick*, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$, where φ is the feature-space transformation and $\langle \cdot, \cdot \rangle$ denotes inner product. The “trick” consists on computing the kernel $k(x, x')$ while avoiding the computation of the inner product and the transformations $\varphi(x), \varphi(x')$. A special case of polynomial transformation, the *polynomial kernel*, $k(x, x') = \langle x, x' \rangle^d$ is commonly used in regression and classification tasks with SVMs. However general polynomial transformations do not verify the kernel trick.

Polynomials, one of the most studied subjects in mathematics, generalize linear functions and define, perhaps, the simplest and most used nonlinear models. Applications include colorimetric calibration (?), explicit formulæ for turbulent pipe flows (?), computational linguistics (?) and more recently analytical techniques for cultural heritage materials (?), liquid epoxy moulding process (?), B-spline surface reconstruction (?), product design (?) or forecasting cyanotoxins presence in water reservoirs (?). These examples not only illustrate the wide spectrum of applications but, additionally, each one uses, at some point, Genetic algorithms (GA).

26 Evolutionary algorithms, including GA, were, arguably, one of the hottest
27 topics of research in the recent decades and with good reason since they out-
28 line an optimization scheme easy to conceptualize and with very broad appli-
29 cation. If a nonlinear (or otherwise) model requires parameterization, GAs
30 provide a simple and often effective approach to search for locally optimal
31 parameters. Related research abound and spans from the 1950s seminal work
32 of Nils Aall Barricelli (?) in the Institute for Advanced Study of Princeton
33 to today’s principal area of study for thousands of researchers, covered in
34 hundreds of conferences, workshops and other meetings. Perhaps the key
35 impulse to GAs come from John Holland’s work and his book “Adaptation
36 in Natural and Artificial Systems” (?).

37 One interesting variation of genetic algorithms, named *genetic program-*
38 *ming* by John Koza (?), proposes the use of GAs to search the syntactic
39 structure of complex functions. Syntactic structure search is also keen to the
40 central ideas of deep learning (??), a subarea of machine learning actually
41 producing quite promising results (*e.g.* in ?). It is also related to the work
42 presented in this paper in the sense that, unlike linear models that have a
43 simple structure, $y = \sum_i \beta_i x_i$, nonlinear (in particular polynomial) models
44 pose an additional structure search problem.

45 The idea of using GAs to find a polynomial regression is not new (???)
46 but still generates original research (??). The modern formulation of the use
47 of GA to find polynomial models is known as Evolutionary Polynomial Re-
48 gression (EPR) and systematization can be traced back to the work of David-
49 son, Savic and Walters (?). Further developments include multi-objective
50 optimizations (?).

51 This paper describes an extension of the general EPR method to find a
52 regularized polynomial regression of a given dataset. The optimal regression
53 results from a cost function that accounts for both the root-mean-square
54 (error) and a regularization factor to avoid overfit by penalysing polynomial
55 complexity.

56 The next section describes the method's details and is followed by a pre-
57 sentation of some performance results. The last section draws some conclu-
58 sions and points future research tasks.

59 2. Genetic Algorithms for Polynomials

60 This section starts with a brief introduction and outline of the evolution-
61 ary polynomial regression algorithm, EPR, and proceeds into core details as
62 the encoding used to represent individual polynomial instances in the GA
63 populations and the regularization of the cost function.

An usual representation of polynomials is through expressions of the form

$$p(x_1, \dots, x_m) = \sum_i \theta_i q_i$$

64 where each $q_i = \prod_j x_j^{\alpha_{ij}}$ is a monomial, the exponents $\alpha_{ij} \in \mathbb{N}_0$ are non-
65 negative integers and the coefficients $\theta_i \in \mathbb{R}$ are real valued. For example
66 $p(x_1, x_2, x_3) = 2x_1 + x_2x_3 + \frac{1}{2}x_1^2x_3$ has monomials $q_1 = x_1, q_2 = x_2x_3$ and
67 $q_3 = x_1^2x_3$, exponents $\alpha_{1,1} = 1, \alpha_{2,2} = 1, \alpha_{2,3} = 1, \alpha_{3,1} = 2, \alpha_{3,3} = 1$ and all
68 other $\alpha_{ij} = 0$ and coefficients $\theta_1 = 2, \theta_2 = 1$ and $\theta_3 = 1/2$.

The exponents alone can be organized into a matrix $[\alpha_{ij}]$ that defines the
monomial structure of the polynomial. For the example above the matrix

representation of the monomials is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} x_1 \\ x_2 x_3 \\ x_1^2 x_3 \end{bmatrix}$$

where each row defines a monomial and each column represents a variable. Changing the order of the rows doesn't change the polynomial whereas changing the order of the columns corresponds to changing the respective variables.

This partial representation of polynomials makes the problem of structure search very clear: except for the trivial cases, the number of possible monomials given n variables and a maximum joint degree d grows exponentially with either n or d . But more importantly, by separating the set of monomials from the coefficients, the polynomial regression problem can be naturally split into two subproblems:

1. For a given set of monomials $\mathcal{Q} = \{q_1, \dots, q_k\}$ find the regression coefficients $\Theta = \{\theta_1, \dots, \theta_k\}$ that minimize the error on a given dataset;
2. Find the fittest set of monomials, *i.e.* the polynomial that minimizes the error on the same dataset;

More precisely, concerning the first problem, let \mathcal{D} be a dataset with n observations of variables Y, X_1, \dots, X_m and $\mathcal{Q} = \{q_1, \dots, q_k\}$ a set of k monomial expressions over X_1, \dots, X_m . Define the hypothesis¹

$$h_{\Theta, \mathcal{Q}}(x_1, \dots, x_m) = \sum_{j=1}^k \theta_j q_j|_{X_i=x_i, \forall 1 \leq i \leq m}$$

¹The expression " $q|_{X=x}$ " reads " q with all instances of X replaced by x ."

85 and let the error (as “cost”) be

$$J_{\text{fit}}(\Theta; \mathcal{Q}, \mathcal{D}) = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - h_{\Theta, \mathcal{Q}}(x_1^{(i)}, \dots, x_m^{(i)}) \right)^2} \quad (1)$$

86 the usual root-mean-square (error) function. Now the first problem can be
 87 stated as: *Given a dataset \mathcal{D} and a set of monomials \mathcal{Q} find parameters Θ*
 88 *that minimize the cost $J_{\text{fit}}(\Theta; \mathcal{Q}, \mathcal{D})$.*

89 This is a simple linear regression problem obtained by expanding \mathcal{D} with
 90 columns that replicate the monomials in \mathcal{Q} . The resulting dataset, $\mathcal{D} \cup \mathcal{Q}(\mathcal{D})$,
 91 adds the monomial transformations in \mathcal{Q} to the original dataset \mathcal{D} . An
 92 alternative formulation would just replace \mathcal{D} by $\mathcal{Q}(\mathcal{D})$. It turns out that the
 93 first formulation is a special case of the second (by including the variables in
 94 the monomial set) and has better error performance — what is not surprising
 95 because it uses more features.

96 The second problem is treated in the GA setting: Let \mathcal{D} be a dataset as
 97 above and \mathcal{P} a set of polynomials. For each polynomial $p \in \mathcal{P}$ let \mathcal{Q}_p be the
 98 set of monomials in p (without the coefficients) and define the (anti) fitness

$$\phi_p = \min_{\Theta} J_{\text{fit}}(\Theta; \mathcal{Q}_p, \mathcal{D})$$

99 by solving the first problem. With a fitness of every instance, the GA genetic
 100 operators (usually mutation and crossover) evolve the population \mathcal{P} until a
 101 reasonable approximation of a local minimum is found. The properties of
 102 GAs and linear regression entail that Algorithm 1 converges to a polynomial
 103 that is a local minimum of the fitness function, encapsulated in the error
 104 function J_{fit} .

Algorithm 1 This EPR algorithm uses linear regression for the calculation of the error J and the space of polynomials is searched in the GAs iteration step. At exit the error of the fittest instance is bounded by ϵ or the maximum number of allowed iterations.

```

function EPR( $D, pop_0, \epsilon, maxiter$ )
     $pop \leftarrow pop_0; err \leftarrow 1.0 + \epsilon$ 
    while  $err > \epsilon \wedge iterations < maxiter$  do
         $pop \leftarrow \text{ITERATEGA}(pop)$ 
         $pop \leftarrow \text{SORT}(pop, key = J)$   $\triangleright$  Sort population by regression error
         $err \leftarrow J(\text{FIRST}(pop))$ 
    end while
    return  $\text{FIRST}(pop)$ 
end function

```

105 Subsection 2.1 describes the encoding of individual polynomial instances
106 as chromosomes and other parameters used in the GA implementation. The
107 regularization of the cost function is discussed in subsection 2.2.

108 2.1. Polynomial Encoding

109 The specific encoding (representation) of a set of monomials is an im-
110 portant aspect in the implementation of EPR. The choice described below
111 permits active and inactive monomials for regression purposes. The active (or
112 inactive) state of a monomial might change through mutation or crossover.
113 This simple mechanism enhances variation in the complexity of polynomial
114 expressions by evolutionary operations.

115 Let $\{q_1, \dots, q_k\}$ be a set of monomials over the variables X_1, \dots, X_m . The

116 encoding of that set using d bits per exponent is a binary list such that

- 117 1. the initial segment of k bits defines the active state of each monomial;
- 118 2. the remaining bits are split into k segments of size $m \times d$, each repre-
- 119 senting a monomial;
- 120 3. the bits in each monomial segment are split into m sub-segments of
- 121 size d . The j^{th} sub-segment is the binary representation of the degree
- 122 of the variable X_j in the enclosing monomial segment;

This encoding can also be viewed as the flattening of the binary exponents in the matrix representation prefixed by the activation segment. The set $\{x_1^3x_3, x_3^7, x_1x_2\}$ (with $m = 3$) has matrix representation

$$\begin{bmatrix} 3 & 0 & 1 \\ 0 & 0 & 7 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 011 & 000 & 001 \\ 000 & 000 & 111 \\ 001 & 001 & 000 \end{bmatrix}_{(2)}$$

123 where the right matrix is in binary form using $d = 3$ bits. An encoding of this
 124 set of monomials with the extra monomial $x_1^6x_2^2x_3^5$ inactive, setting $k = 4$,
 125 would be

$$126 \quad 1110; 011, 000, 001; 000,000,111; 001,001,000; 110,010,101$$

127 where, for reading purposes, semicolons separate segments and commas sep-
 128 arate variables. The first $k = 4$ bits inform that the first, second and third
 129 monomials are active while the fourth is not.

130 While each valid encoding represents a set of monomials the map is not
 131 bijective: each set of monomials has multiple encodings, for example by
 132 changing d or the order of monomial segments. However, considering the

133 EPR task, this is a minor issue and a bijective map would add computational
 134 complexity and negative impact to the algorithm’s performance.

135 There is one final remark concerning this encoding method. As it is,
 136 the activation segment can become all zeros, representing the empty set of
 137 monomials. This situation can be avoided with a simple hack: Given an
 138 encoding, the first monomial is always considered active, thus restricting the
 139 syntactic form of encodings to binary strings starting with 1. In practice,
 140 this means that the implementation of the encoding can omit the first bit.

141 2.2. Cost Function

142 The polynomial regression error considered so far accounts for the ability
 143 to predict the transformed testset. A known problem of using a cost function
 144 based only in the dataset error (and of polynomial regressions in general) is
 145 the tendency to overfit training data. Excessive variance of the estimation
 146 method can be reduced by regularizing the error function with a penalty
 147 factor. Thus, to reduce polynomial complexity and variance by regularizing
 148 the size of the monomial set the error function from equation 1 is multiplied
 149 by a factor λ^k

$$J_{\text{reg}}(\Theta, \lambda; \mathcal{Q}, \mathcal{D}) = \lambda^k J_{\text{fit}}(\Theta; \mathcal{Q}, \mathcal{D}) \quad (2)$$

150 where k is the number of monomials in the polynomial. When $\lambda > 1$ poly-
 151 nomials with more monomials are penalized. The regularized extension of
 152 EPR is denoted by Evolutionary Polynomial Regression with Regulariza-
 153 tion (EPRR).

154 A simple exploration on the effect of the regularization parameter is de-
 155 picted in Figure 1 where it is possible to observe that the typical inflection

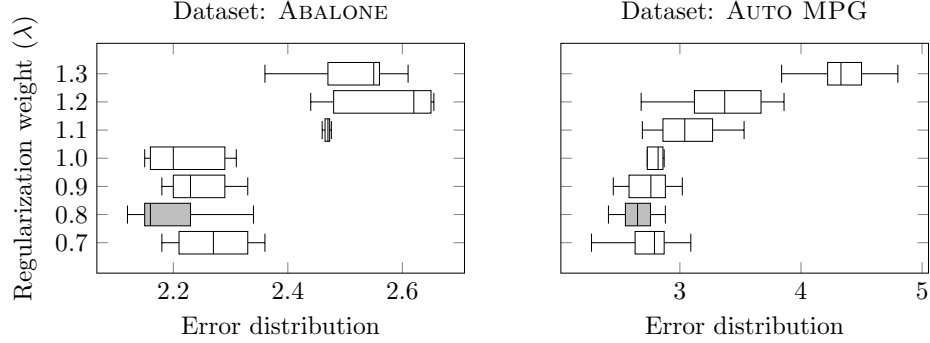


Figure 1: Error distribution by regularization exponent for two common datasets. The box plots summarize error values of ten simulations for each value of λ . The smallest overall error, in grey, is achieved in both datasets when $\lambda = 0.8$. Performance of the non-regularized EPR is plotted in the line $\lambda = 1$.

point lies around $\lambda = 0.8$. This value, favoring “larger” polynomials is justified by the balance of the data’s non-linearity and polynomial complexity: below $\lambda = 0.8$, even penalized, larger monomial sets achieve better error performance than smaller ones while above that value the size of the monomial set is excessive. Within this tension the overall error results reduced when compared to the non-regularized EPR version.

2.3. Genetic Algorithm Parameterization

In general GAs offer many possibilities with respect to the choice of genetic operators and respective application rates, population evolution, *etc.* The results found here were obtained using the package `genalg` (?) with default parameters, standard operators (crossover and mutation) and population evolution with 20% elitism between generations.

168 3. Experimental Results

169 Here is described the experiment setup used to gather and summarize the
170 empirical evidence that supports this comparative study of EPR and EPRR.
171 Evaluation is focused in error distribution and, besides EPR and EPRR, also
172 uses several common regression methods and datasets easily accessible in R,
173 the free software environment for statistical computing and graphics (?)². A
174 small consideration on the convergence speed concludes this section.

175 3.1. Regression Methods and Datasets

176 The EPRR method is ranked against several well-known learning algo-
177 rithms for regression, namely: non-regularized EPR, Linear Regression, Sup-
178 port Vector Machines (?), Regression Trees (?) and Conditional Inference
179 Trees (???). To achieve better error results the SVM and Regression Tree
180 parameters are tuned in each dataset.

181 The performance of each method is evaluated on several common datasets.
182 From each dataset 70% of the observations are reserved for training purposes
183 and the remaining observations used to estimate the error. To enhance the
184 robustness of results this process is repeated 25 times, each time with a
185 different shuffling of the samples in the train and test sets. Some datasets
186 with attribute values of different magnitudes have a pre-processing scaling
187 transformation. The box plots in figures 2 and 3 resume the test set error
188 distributions over these different runs.

²The datasets and R code used to produce the results and plots in this paper are available online at <https://github.com/jpneto/GenAlgPoly>.

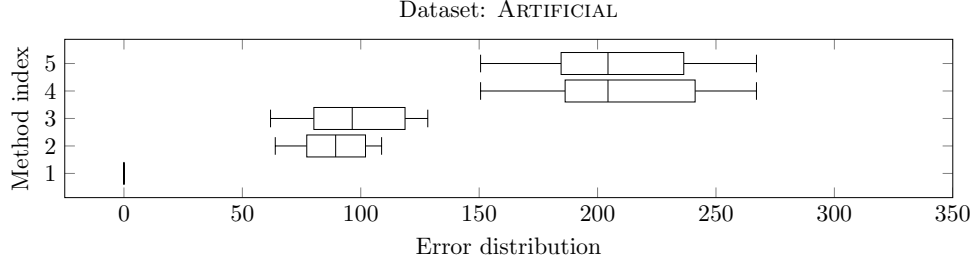


Figure 2: Testing polynomial discovery. The dataset is generated from a polynomial expression and, as shown, EPRR finds the exact generator structure: in line 1, the error box is centered in 0 and has width 0. The regression methods depicted are: 1. EPRR, 2. Linear Regression, 3. SVM, 4. Regression Trees and 5. Conditional Inference Trees

One of the used datasets, `ARTIFICIAL`, has a special role: it is used to test if EPRR is able to discover a polynomial model. The idea of this test is to generate a polynomial dependent variable and measure the EPRR error after fitting the dataset. The genetic algorithm parameterization for this dataset uses a population with size $n = 100$ and evolves for 50 generations. For the remaining datasets the population has size $n = 300$ and evolves for 100 generations.

`ARTIFICIAL` is a polynomial dataset with four numeric features, x_1, \dots, x_4 , where x_1, x_3 are outcomes from Poisson random variables, and x_2, x_4 from Normal random variables. The dependent variable is given by the polynomial expression $y = x_2x_4^2 + x_1^2x_3 + 5$. The dataset includes $n = 50$ observations;

`HOUSING` concerns the task of predicting housing values in areas of Boston.

There are $n = 506$ observations of $m = 13$ continuous attributes and

203 one dependent variable, the median value of owner-occupied homes in
204 thousands of USD;

205 ABALONE is used to predict the age of a abalone shell using $m = 8$ numeric
206 attributes concerning several physical measurements. There are $n =$
207 4177 observations;

208 AUTO MPG gathers fuel consumption in miles per gallon, based on two
209 discrete and five continuous attributes ($m = 7$). There are $n = 398$
210 observations;

211 KINEMATICS results from a realistic simulation of the forward kinematics
212 of an 8 link robot arm. The task is to predict the distance of the end-
213 effector from a target using $m = 8$ continuous attributes. There are
214 $n = 8192$ observations;

215 3.2. Convergence speed

216 Since this work is oriented to the error of the EPRR model it is necessary
217 to assess how this depends on the number of generations of the GA. As
218 illustrated in Figure 4, the error quickly drops during the initial 50 to 100
219 generations. Then, it proceeds slower achieving better solutions only with
220 marginal error reduction.

221 4. Conclusion and Future Work

222 Of the regression methods considered SVM achieves the best results in
223 three out of four datasets. However SVM and Conditional Inference Trees
224 are pre-trained, having parameters tuned for each particular dataset unlike

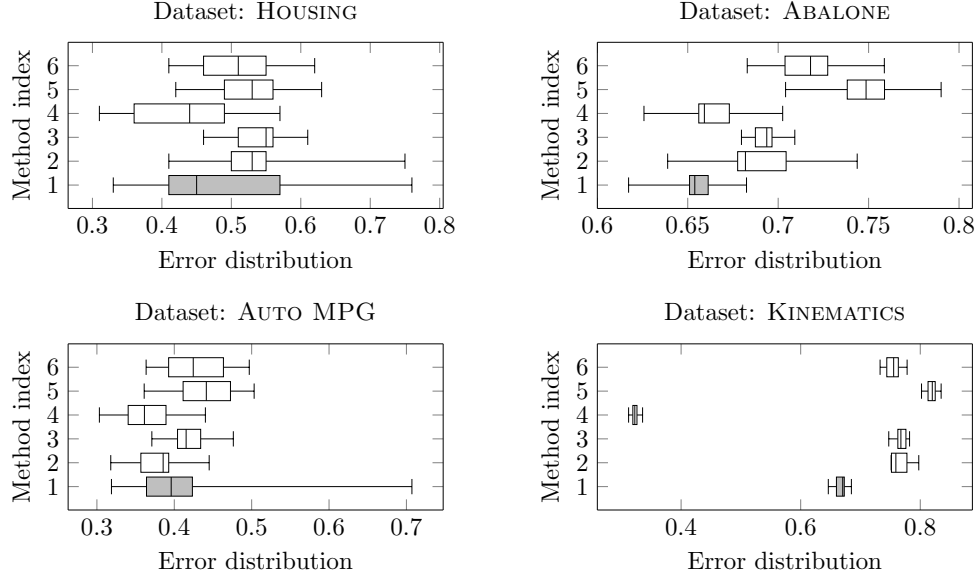


Figure 3: Summary results for different regression methods on diverse datasets. Although EPRR not always achieves the smallest expected error, performance is on-par with more sophisticated methods. The regression methods depicted in these figures are: 1. EPRR; 2. EPR; 3. Linear Regression; 4. SVM; 5. Regression Trees; 6. Conditional Inference Trees;

225 EPRR, that runs with the same parameterization on all datasets. Even so it
 226 is the best estimator for the ABALONE dataset and in the remaining datasets
 227 it outperforms most of the other estimators.

228 Comparing EPR and EPRR — the main article’s topic — the regularized
 229 version achieves much better results at ABALONE and especially KINEMAT-
 230 ICS. On the HOUSING dataset errors are improved wrt EPR in a differ-
 231 ence in means, resulted in a 95% HDI (Highest Density Interval) equal to
 232 $[0.001, 0.119]$ which, while borderline, achieves statistical significance. Only
 233 in the AUTO MPG dataset EPR achieves better results, even if not that

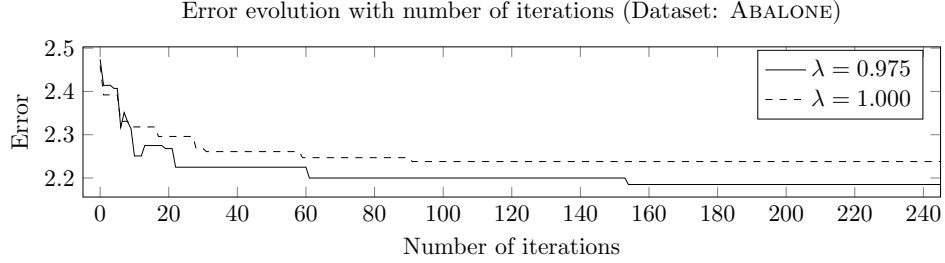


Figure 4: Learning curve: Error progress for the ABALONE dataset during a single execution of the genetic algorithm. The figure shows the fitness evolution for two different regularization values. The population for both consists of 200 polynomials. The error values seem to stabilize around iteration 250.

different from EPRR.

For complexity considerations EPR and EPRR demand some processing time. On a quad-core computer, processing the KINEMATICS dataset (with near 8K observations) takes approximately 5 minutes. Probably processing time can be reduced by one to two orders in magnitude if the algorithm is implemented with computational speed in mind. However, speed optimization is not the focus of this article.

A cross-validation procedure can be implemented to refine the appropriate parameter values to achieve better errors. Namely, the regularization parameter, λ , can be tested with several values, instead of being fixed at 0.8. Other parameters like mutation chance or the amount of elitism can also be tested. However, these type of tests need a low-level, fast implementation of EPR and are postponed to future investigation.

247 **Acknowledgements**

248 The authors are grateful to the Fundação para a Ciência e Tecnologia
249 (FCT) and the R&D laboratory LabMAg for the financial support given to
250 this work, under the strategic project PEST-OE/EEI/UI0434/2011.

251 Datasets used herein are selected from Luís Torgo's data repository, [http:](http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html)
252 [//www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html](http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html). Most can also
253 be found in the UCI ML repository at <http://archive.ics.uci.edu/ml/>.

254 The authors wish to thank professor André Falcão for motivation and
255 useful discussions around the article.

256 **Bibliography**