

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) (Schölkopf et al. (1997); Liang and Lee (2012); Bao et al. (2013)) alleviates this problem by allowing special nonlinear 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 (Mendes and Carvalho, 2005), explicit formulæ for turbulent pipe flows (Davidson et al., 1999), computational linguistics (Sánchez et al., 2009) and more recently analytical techniques for cultural heritage materials (Cséfalvayová et al., 2010), liquid epoxy moulding process (Chan et al., 2011), B-spline surface reconstruction (Gálvez et al., 2012), product design (Chan et al., 2012) or forecasting cyanotoxins

26 presence in water reservoirs (García Nieto et al., 2013). These examples not
27 only illustrate the wide spectrum of applications but, additionally, each one
28 uses, at some point, Genetic algorithms (GA).

29 Evolutionary algorithms, including GA, were, arguably, one of the hottest
30 topics of research in the recent decades and with good reason since they out-
31 line an optimization scheme easy to conceptualize and with very broad appli-
32 cation. If a nonlinear (or otherwise) model requires parameterization, GAs
33 provide a simple and often effective approach to search for locally optimal
34 parameters. Related research abound and spans from the 1950s seminal work
35 of Nils Aall Barricelli (Barricelli, 1962) in the Institute for Advanced Study
36 of Princeton to today’s principal area of study for thousands of researchers,
37 covered in hundreds of conferences, workshops and other meetings. Perhaps
38 the key impulse to GAs come from John Holland’s work and his book “Adap-
39 tation in Natural and Artificial Systems” (Holland, 1975).

40 One interesting variation of genetic algorithms, named *genetic program-*
41 *ming* by John Koza (Koza, 1992), proposes the use of GAs to search the
42 syntactic structure of complex functions. Syntactic structure search is also
43 keen to the central ideas of deep learning (Bengio, 2009; Bengio et al., 2013),
44 a subarea of machine learning actually producing quite promising results
45 (*e.g.* in Tarlow et al. (2013)). It is also related to the work presented in this
46 paper in the sense that, unlike linear models that have a simple structure,
47 $y = \sum_i \beta_i x_i$, nonlinear (in particular polynomial) models pose an additional
48 structure search problem.

49 The idea of using GAs to find a polynomial regression is not new (Maertens
50 et al., 2006; Yu and Lin, 2008; Wu et al., 2009) but still generates original

51 research (Hofwing et al., 2011; Cetisli and Kalkan, 2011). The modern formu-
52 lation of the use of GA to find polynomial models is known as Evolutionary
53 Polynomial Regression (EPR) and systematization can be traced back to the
54 work of Davidson, Savic and Walters (Davidson et al. (2003)). Further devel-
55 opments include multi-objective optimizations (Giustolisi and Savic (2009)).

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

61 The next section describes the method’s details and is followed by a pre-
62 sentation of some performance results. The last section draws some conclu-
63 sions and points future research tasks.

64 2. Genetic Algorithms for Polynomials

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

An usual representation of polynomials is through expressions of the form

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

69 where each $q_i = \prod_j x_j^{\alpha_{ij}}$ is a monomial, the exponents $\alpha_{ij} \in \mathbb{N}_0$ are non-
70 negative integers and the coefficients $\theta_i \in \mathbb{R}$ are real valued. For example
71 $p(x_1, x_2, x_3) = 2x_1 + x_2x_3 + \frac{1}{2}x_1^2x_3$ has monomials $q_1 = x_1, q_2 = x_2x_3$ and

72 $q_3 = x_1^2 x_3$, exponents $\alpha_{1,1} = 1, \alpha_{2,2} = 1, \alpha_{2,3} = 1, \alpha_{3,1} = 2, \alpha_{3,3} = 1$ and all
 73 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

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

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

77 This partial representation of polynomials makes the problem of struc-
 78 ture search very clear: except for the trivial cases, the number of possible
 79 monomials given n variables and a maximum joint degree d grows exponen-
 80 tially with either n or d . But more importantly, by separating the set of
 81 monomials from the coefficients, the polynomial regression problem can be
 82 naturally split into two subproblems:

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

87 More precisely, concerning the first problem, let \mathcal{D} be a dataset with n obser-
 88 vations of variables Y, X_1, \dots, X_m and $\mathcal{Q} = \{q_1, \dots, q_k\}$ a set of k monomial

89 expressions over X_1, \dots, X_m . Define the hypothesis¹

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

90 and let the error (as “cost”) be

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

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

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

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

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

104 by solving the first problem. With a fitness of every instance, the GA genetic
 105 operators (usually mutation and crossover) evolve the population \mathcal{P} until a

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

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

```

106 reasonable approximation of a local minimum is found. The properties of
 107 GAs and linear regression entail that Algorithm 1 converges to a polynomial
 108 that is a local minimum of the fitness function, encapsulated in the error
 109 function J_{fit} .

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

113 2.1. Polynomial Encoding

114 The specific encoding (representation) of a set of monomials is an im-
 115 portant aspect in the implementation of EPR. The choice described below
 116 permits active and inactive monomials for regression purposes. The active (or

inactive) state of a monomial might change through mutation or crossover. This simple mechanism enhances variation in the complexity of polynomial expressions by evolutionary operations.

Let $\{q_1, \dots, q_k\}$ be a set of monomials over the variables X_1, \dots, X_m . The encoding of that set using d bits per exponent is a binary list such that

1. the initial segment of k bits defines the active state of each monomial;
2. the remaining bits are split into k segments of size $m \times d$, each representing a monomial;
3. the bits in each monomial segment are split into m sub-segments of size d . The j^{th} sub-segment is the binary representation of the degree 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 $\{x_1^3 x_3, x_3^7, x_1 x_2\}$ (with $m = 3$) has matrix representation

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

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

$$1110; 011, 000, 001; 000,000,111; 001,001,000; 110,010,101$$

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

135 While each valid encoding represents a set of monomials the map is not
 136 bijective: each set of monomials has multiple encodings, for example by
 137 changing d or the order of monomial segments. However, considering the
 138 EPR task, this is a minor issue and a bijective map would add computational
 139 complexity and negative impact to the algorithm’s performance.

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

146 2.2. Cost Function

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

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

155 where k is the number of monomials in the polynomial. When $\lambda > 1$ poly-
 156 nomials with more monomials are penalized. The regularized extension of

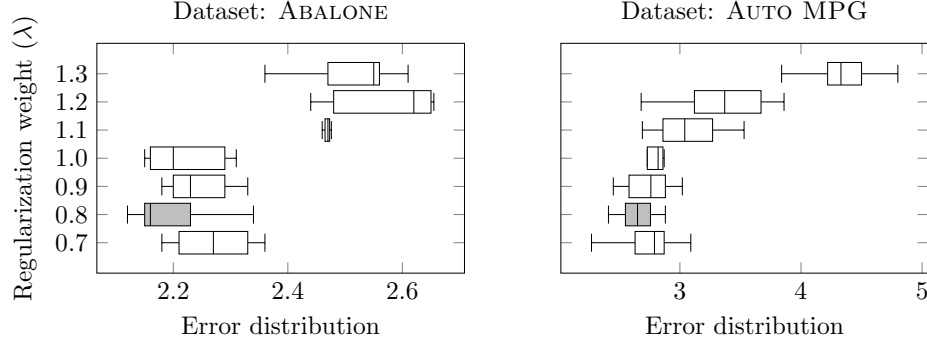


Figure 1: Error distribution by regularization exponent for two common datasets. The box plots summarize error values of ten simulations for each value of λ . The smallest overall error, in grey, is achieved in both datasets when $\lambda = 0.8$. Performance of the non-regularized EPR is plotted in the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

157 EPR is denoted by Evolutionary Polynomial Regression with Regulariza-
 158 tion (EPRR).

159 A simple exploration on the effect of the regularization parameter is de-
 160 picted in Figure 1 where it is possible to observe that the typical inflection
 161 point lies around $\lambda = 0.8$. This value, favoring “larger” polynomials is jus-
 162 tified by the balance of the data’s non-linearity and polynomial complexity:
 163 below $\lambda = 0.8$, even penalized, larger monomial sets achieve better error per-
 164 formance than smaller ones while above that value the size of the monomial
 165 set is excessive. Within this tension the overall error results reduced when
 166 compared to the non-regularized EPR version.

167 2.3. Genetic Algorithm Parameterization

168 In general GAs offer many possibilities with respect to the choice of ge-
 169 netic operators and respective application rates, population evolution, *etc.*

170 The results found here were obtained using the package `genalg` (Willigha-
171 gen, 2012) with default parameters, standard operators (crossover and mu-
172 tation) and population evolution with 20% elitism between generations.

173 3. Experimental Results

174 Here is described the experiment setup used to gather and summarize the
175 empirical evidence that supports this comparative study of EPR and EPRR.
176 Evaluation is focused in error distribution and, besides EPR and EPRR, also
177 uses several common regression methods and datasets easily accessible in `R`,
178 the free software environment for statistical computing and graphics (R Core
179 Team, 2013)². A small consideration on the convergence speed concludes this
180 section.

181 3.1. Regression Methods and Datasets

182 The EPRR method is ranked against several well-known learning algo-
183 rithms for regression, namely: non-regularized EPR, Linear Regression, Sup-
184 port Vector Machines (Meyer et al., 2012), Regression Trees (Therneau et al.,
185 2013) and Conditional Inference Trees (Hothorn et al., 2006; Strobl et al.,
186 2007, 2008). To achieve better error results the SVM and Regression Tree
187 parameters are tuned in each dataset.

188 The performance of each method is evaluated on several common datasets.
189 From each dataset 70% of the observations are reserved for training purposes
190 and the remaining observations used to estimate the error. To enhance the

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

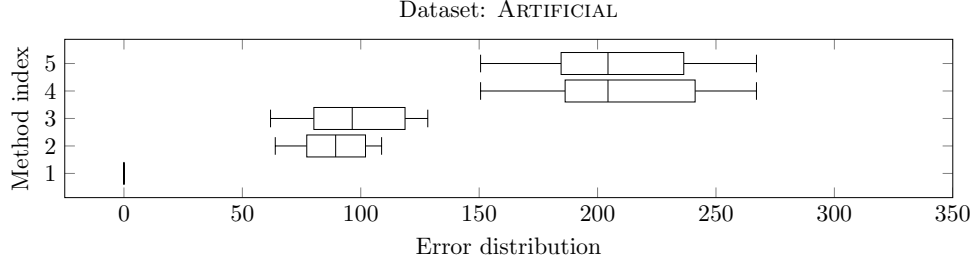


Figure 2: Testing polynomial discovery. The dataset is generated from a polynomial expression and, as shown, EPRR finds the exact generator structure: in line 1, the error box is centered in 0 and has width 0. The regression methods depicted are: 1. EPRR, 2. Linear Regression, 3. SVM, 4. Regression Trees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

robustness of results this process is repeated 25 times, each time with a different shuffling of the samples in the train and test sets. Some datasets with attribute values of different magnitudes have a pre-processing scaling transformation. The box plots in figures 2 and 3 resume the test set error distributions over these different runs.

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

206 the polynomial expression $y = x_2x_4^2 + x_1^2x_3 + 5$. The dataset includes
207 $n = 50$ observations;

208 HOUSING concerns the task of predicting housing values in areas of Boston.
209 There are $n = 506$ observations of $m = 13$ continuous attributes and
210 one dependent variable, the median value of owner-occupied homes in
211 thousands of USD;

212 ABALONE is used to predict the age of a abalone shell using $m = 8$ numeric
213 attributes concerning several physical measurements. There are $n =$
214 4177 observations;

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

218 KINEMATICS results from a realistic simulation of the forward kinematics
219 of an 8 link robot arm. The task is to predict the distance of the end-
220 effector from a target using $m = 8$ continuous attributes. There are
221 $n = 8192$ observations;

222 3.2. Convergence speed

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

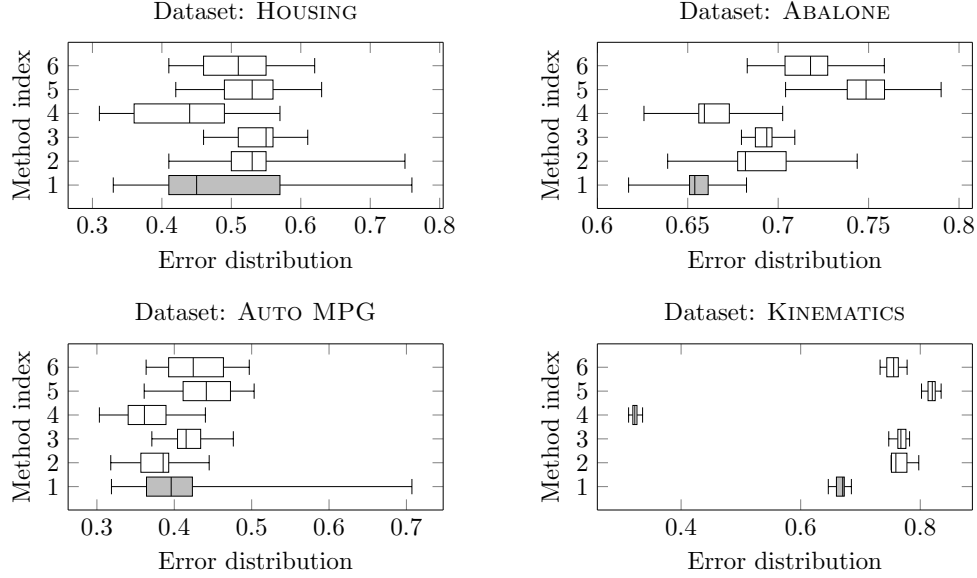


Figure 3: Summary results for different regression methods on diverse datasets. Although EPRR not always achieves the smallest expected error, performance is on-par with more sophisticated methods. The regression methods depicted in these figures are: 1. EPRR; 2. EPR; 3. Linear Regression; 4. SVM; 5.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

228 4. Conclusion and Future Work

229 Of the regression methods considered SVM achieves the best results in
 230 three out of four datasets. However SVM and Conditional Inference Trees
 231 are pre-trained, having parameters tuned for each particular dataset unlike
 232 EPRR, that runs with the same parameterization on all datasets. Even so it
 233 is the best estimator for the ABALONE dataset and in the remaining datasets
 234 it outperforms most of the other estimators.

235 Comparing EPR and EPRR — the main article’s topic — the regularized
 236 version achieves much better results at ABALONE and especially KINEMAT-

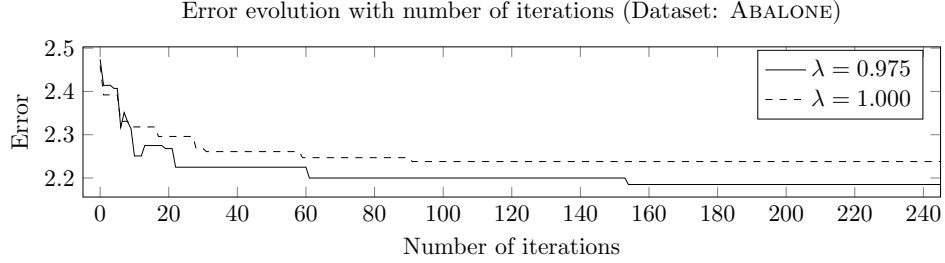


Figure 4: Learning curve: Error progress for the ABALONE dataset during a single execution of the genetic algorithm. The figure shows the fitness evolution for two different regularization values. The population for both consists of 200 polynomials. The error values seem to stabilize around iteration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

237 ICS. On the HOUSING dataset errors are improved wrt EPR in a differ-
 238 ence in means, resulted in a 95% HDI (Highest Density Interval) equal to
 239 $[0.001, 0.119]$ which, while borderline, achieves statistical significance. Only
 240 in the AUTO MPG dataset EPR achieves better results, even if not that
 241 different from EPRR.

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

248 A cross-validation procedure can be implemented to refine the appropri-
 249 ate parameter values to achieve better errors. Namely, the regularization
 250 parameter, λ , can be tested with several values, instead of being fixed at 0.8.
 251 Other parameters like mutation chance or the amount of elitism can also be

252 tested. However, these type of tests need a low-level, fast implementation of
253 EPR and are postponed to future investigation.

254 **Acknowledgements**

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

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

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

263 **Bibliography**

- 264 Bao, Y., Hu, Z., Xiong, T., 2013. A pso and pattern search based memetic
265 algorithm for svms parameters optimization. *Neurocomputing* 117, 98–106.
- 266 Barricelli, N.A., 1962. Numerical testing of evolution theories. part i: Theo-
267 retical introduction and basic tests. *Acta Biotheoretica* 16, 69–98.
- 268 Bengio, Y., 2009. Learning deep architectures for AI. *Foundations and trends*
269 *in Machine Learning* 2, 1–127.
- 270 Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A
271 review and new perspectives .
- 272 Cetisli, B., Kalkan, H., 2011. Polynomial curve fitting with varying real
273 powers. *Electronics and Electrical Engineering* 112, 117–122.

- 274 Chan, K.Y., Dillon, T.S., Kwong, C.K., 2011. Modeling of a liquid epoxy
275 molding process using a particle swarm optimization-based fuzzy regression
276 approach. *Industrial Informatics, IEEE Transactions on* 7, 148–158.
- 277 Chan, K.Y., Kwong, C., Dillon, T.S., 2012. Development of product design
278 models using fuzzy regression based genetic programming, in: *Computational Intelligence Techniques for New Product Design*. Springer, pp.
279 111–128.
- 281 Cséfalvayová, L., Pelikan, M., Kralj Cigić, I., Kolar, J., Strlič, M., 2010.
282 Use of genetic algorithms with multivariate regression for determination of
283 gelatine in historic papers based on FT-IR and NIR spectral data. *Talanta*
284 82, 1784–1790.
- 285 Davidson, J., Savic, D., Walters, G., 1999. Method for the identification of
286 explicit polynomial formulae for the friction in turbulent pipe flow. *Journal*
287 *of Hydroinformatics* 1, 115–126.
- 288 Davidson, J., Savic, D.A., Walters, G.A., 2003. Symbolic and numerical
289 regression: Experiments and applications. *Information Sciences* 150, 95–
290 117.
- 291 Gálvez, A., Iglesias, A., Puig-Pey, J., 2012. Iterative two-step genetic-
292 algorithm-based method for efficient polynomial b-spline surface recon-
293 struction. *Information Sciences* 182, 56–76.
- 294 García Nieto, P.J., Alonso Fernández, J., de Cos Juez, F., Sánchez Lasheras,
295 F., Díaz Muñoz, C., 2013. Hybrid modelling based on support vector

296 regression with genetic algorithms in forecasting the cyanotoxins presence
 297 in the trasona reservoir (northern spain). *Environmental research* .

298 Giustolisi, O., Savic, D., 2009. Advances in data-driven analyses and mod-
 299 elling using epr-moga. *Journal of Hydroinformatics* 11, 225–236.

300 Hofwing, M., Strömberg, N., Tapankov, M., 2011. Optimal polynomial
 301 regression models by using a genetic algorithm, in: *Proceedings of the*
 302 *Second International Conference on Soft Computing Technology in Civil,*
 303 *Structural and Environmental Engineering Conference,(Crete, Greece),*
 304 *2011*009.

305 Holland, J.H., 1975. *Adaptation in natural and artificial systems: An intro-*
 306 *ductory analysis with applications to biology, control, and artificial intel-*
 307 *ligence.* U Michigan Press.

308 Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A
 309 conditional inference framework. *Journal of Computational and Graphical*
 310 *Statistics* 15, 651–674.

311 Koza, J.R., 1992. *Genetic Programming: vol. 1, On the programming of*
 312 *computers by means of natural selection. volume 1.* MIT press.

313 Liang, Z., Lee, Y., 2012. Eigen-analysis of nonlinear pca with polynomial
 314 kernels .

315 Maertens, K., De Baerdemaeker, J., Babuška, R., 2006. Genetic polynomial
 316 regression as input selection algorithm for non-linear identification. *Soft*
 317 *Computing* 10, 785–795.

318 Mendes, L., Carvalho, P.d., 2005. Adaptive polynomial regression for colorimetric scanner calibration using genetic algorithms, in: Intelligent Signal
319 Processing, 2005 IEEE International Workshop on, IEEE. pp. 22–27.

321 Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2012. e1071: Misc functions of the department of statistics (e1071), tu wien URL:
322 <http://CRAN.R-project.org/package=e1071>. r package version 1.6-1.

324 R Core Team, 2013. R: A language and environment for statistical computing
325 URL: <http://www.R-project.org/>.

326 Sánchez, L., Otero, J., Couso, I., 2009. Obtaining linguistic fuzzy rule-based regression models from imprecise data with multiobjective genetic
327 algorithms. *Soft Computing* 13, 467–479.

329 Schölkopf, B., Smola, A., Müller, K.R., 1997. Kernel principal component
330 analysis, in: Artificial Neural Networks ICANN’97. Springer, pp. 583–588.

331 Strobl, C., Boulesteix, A.L., Kneib, T., Augustin, T., Zeileis, A., 2008. Conditional variable importance for random forests. *BMC Bioinformatics* 9.
332 URL: <http://www.biomedcentral.com/1471-2105/9/307>.

334 Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T., 2007. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics* 8. URL: <http://www.biomedcentral.com/1471-2105/8/25>.
335
336
337

338 Tarlow, D., Sutskever, I., Zemel, R.S., 2013. Stochastic k-neighborhood
339 selection for supervised and unsupervised learning. *Journal of Machine Learning Research* .
340

- 341 Therneau, T., Atkinson, B., Ripley, B., 2013. rpart: Recursive partitioning
342 R package version 4.1-1.
- 343 Willighagen, E., 2012. genalg: R based genetic algorithm.
- 344 Wu, C.H., Tzeng, G.H., Lin, R.H., 2009. A novel hybrid genetic algorithm for
345 kernel function and parameter optimization in support vector regression.
346 Expert Systems with Applications 36, 4725–4735.
- 347 Yu, T.L., Lin, W.K., 2008. Optimal sampling of genetic algorithms on poly-
348 nomial regression, in: Proceedings of the 10th annual conference on Ge-
349 netic and evolutionary computation, ACM. pp. 1089–1096.