

A method for regularization of evolutionary polynomial regressions

Francisco Coelho*

*Departamento de Informática, Universidade de Évora, Rua Romão Ramalho 58, 7000-671
Évora, Portugal*

João Pedro Neto*

*Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa, Campo
Grande, 1749-016 Lisboa, Portugal*

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. This paper introduces Evolutionary Polynomial Regression with Regularization, 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 EPR with regularization 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

— GAs are repeatedly mentioned as an approach to search for locally optimal parameters, this is simply incorrect and is a very surprising statement. GAs are

*Corresponding author: Tel.: +351-919-006-379

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

NOT a local optimizer in any shape or form. Our intention was merely to note that GAs give no guarantee of finding a global optimum

— What is the reference for the GA encoding used? It is ours.

— The penalty term introduces is the main contribution of this work, this should be clearly stated in the text. It is basically a complexity based penalty which is really not so novel, there are even several cases in literature of penalty based GA functions. Please dig up further in the state of the art to back up better your claim of novelty. We included references to previous regularization in GA and outlined the role and originality of our contribution.

1 — In some box plots only ten simulations are performed, in others 25. These
2 numbers are too low. The number of samples was increased.

3 1. Introduction

4 With notable exceptions (*e.g.* neural networks) machine learning regres-
5 sion techniques produce linear models. The linearity assumption has many
6 advantages including reduced computational complexity and strong theoretical
7 framework. However nonlinearity is unavoidable in many application scenarios,
8 specially those with phase transitions or feedback loops, so common in engineer-
9 ing, ecology, cybernetics and other areas. The kernel trick in Support Vector
10 Machines (SVM) ([1, 2, 3]) alleviates this problem by allowing special non-
11 linear transformations of the feature-space. The condition such transformations
12 must meet is known as the *kernel trick*, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$, where φ is
13 the feature-space transformation and $\langle \cdot, \cdot \rangle$ denotes inner product. The “trick”
14 consists on computing the kernel $k(x, x')$ while avoiding the computation of the
15 inner product and the transformations $\varphi(x), \varphi(x')$. A special case of polynomial
16 transformation, the *polynomial kernel*, $k(x, x') = \langle x, x' \rangle^d$ is commonly used in
17 regression and classification tasks with SVMs. However the kernel trick doesn’t
18 apply to general polynomial transformations.

19 Polynomials, one of the most studied subjects in mathematics, generalize li-
20 near functions and define, perhaps, the simplest and most used nonlinear mod-

els. For example, Polynomial Neural Networks [4] generalize the linear component of neural networks with a polynomial function. Applications include colorimetric calibration [5], explicit formulæ for turbulent pipe flows [6], computational linguistics [7] and more recently analytical techniques for cultural heritage materials [8], liquid epoxy moulding process [9], B-spline surface reconstruction [10], product design [11] or forecasting cyanotoxins presence in water reservoirs [12]. These examples not only illustrate the wide spectrum of applications but, additionally, each one uses, at some point, Genetic algorithms (GA).

Evolutionary algorithms, including GA, were, arguably, one of the hottest topics of research in the recent decades and with good reason since they outline an optimization scheme easy to conceptualize and with very broad application. If a nonlinear (or otherwise) model requires parameterization, GAs provide a simple and often effective approach to search for global optimal parameters (although no guarantee of global optimality can be given). Related research abound and spans from the 1950s seminal work of Nils Aall Barricelli [13] in the Institute for Advanced Study of Princeton to today’s principal area of study for thousands of researchers, covered in hundreds of conferences, workshops and other meetings. Perhaps the key impulse to GAs came from John Holland’s work and his book “Adaptation in Natural and Artificial Systems” [14].

One interesting variation of genetic algorithms, named *genetic programming* by John Koza [15], proposes the use of GAs to search the syntactic structure of complex functions. Syntactic structure search is also keen to the central ideas of deep learning [16, 17], a subarea of machine learning actually producing quite promising results (*e.g.* in [18]). It is also related to the work presented in this paper in the sense that, unlike linear models that have a simple structure, $y = \sum_i \beta_i x_i$, nonlinear (in particular polynomial) models pose an additional structure search problem.

The idea of using GAs to find a polynomial regression is not new [19, 20, 21] but still generates original research [22, 23]. The modern formulation of the use of GA to find polynomial models is known as Evolutionary Polynomial Regression (EPR) and systematization can be traced back to the work of Davidson,

52 Savic and Walters [24]. Further developments include multi-objective optimiza-
 53 tions [25].

54 Use of regularization/penalty functions is common practice in machine learn-
 55 ing in general and has some applications in GA[26]. This paper describes an
 56 extension of the general EPR method to find a regularized polynomial regres-
 57 sion of a given dataset. Herein optimal regression results from a cost function
 58 that accounts for both the root-mean-square (error) together with a novel reg-
 59 ularization factor that penalizes over-fitting by polynomial complexity.

60 The next section describes the method's details and is followed by a presen-
 61 tation of some performance results. The last section draws some conclusions
 62 and points future research tasks.

63 2. Genetic Algorithms for Polynomials

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

A usual representation of polynomials is through expressions of the form

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

68 where each $q_i = \prod_j x_j^{\alpha_{ij}}$ is a monomial, the exponents $\alpha_{ij} \in \mathbb{N}_0$ are non-
 69 negative integers and the coefficients $\theta_i \in \mathbb{R}$ are real valued. For example
 70 $p(x_1, x_2, x_3) = 2x_1 + x_2x_3 + \frac{1}{2}x_1^2x_3$ has monomials $q_1 = x_1, q_2 = x_2x_3$ and
 71 $q_3 = x_1^2x_3$, exponents $\alpha_{1,1} = 1, \alpha_{2,2} = 1, \alpha_{2,3} = 1, \alpha_{3,1} = 2, \alpha_{3,3} = 1$ and all
 72 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 rep-

resentation of the monomials is

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

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

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

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

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

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

89 and let the error (as “cost”) be

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

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

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

This is a simple linear regression problem obtained by expanding \mathcal{D} with columns that replicate the monomials in \mathcal{Q} . The resulting dataset, $\mathcal{D} \cup \mathcal{Q}(\mathcal{D})$, adds the monomial transformations in \mathcal{Q} to the original dataset \mathcal{D} . An alternative formulation would just replace \mathcal{D} by $\mathcal{Q}(\mathcal{D})$. It turns out that the first formulation is a special case of the second (by including the variables in the monomial set) and has the potential for better error performance because it uses more features. — How is this known??? Is there a reference or some experimental results backing this claim or is it simply intuition??? We reformulated the previous sentence.

The second problem is treated in the GA setting: Let \mathcal{D} be a dataset as above and \mathcal{P} a set of polynomials. For each polynomial $p \in \mathcal{P}$ let \mathcal{Q}_p be the set of monomials in p (without the coefficients) and define the minimization based fitness — say what? Do you mean something like a minimization based fitness?? Anti fitness seems odd. Adopted the reviewer suggestion.

$$\phi_p = \min_{\Theta} J_{\text{fit}}(\Theta; \mathcal{Q}_p, \mathcal{D}) \quad (3)$$

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

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

2.1. Polynomial Encoding

The specific encoding (representation) of a set of monomials is an important aspect in the implementation of EPR. The choice described below, developed

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

```

by the authors for this algorithm, permits active and inactive monomials for regression purposes — is this an original contribution???? Otherwise it needs a reference. it is ours. 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

$$\begin{bmatrix} 3 & 0 & 1 \\ 0 & 0 & 7 \\ 1 & 1 & 0 \\ 6 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 011 & 000 & 001 \\ 000 & 000 & 111 \\ 001 & 001 & 000 \\ 110 & 010 & 101 \end{bmatrix}_{(2)}$$

where the right matrix is in binary form using $d = 3$ bits. An example of this set of monomials with the forth monomial, $x_1^6 x_2^2 x_3^5$, inactive 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.

While each valid encoding represents a set of monomials the map is not bijective: each set of monomials has multiple encodings, for example by changing d or the order of monomial segments. However, considering the EPR task, this is a minor issue and a bijective map would add computational complexity and negative impact to the algorithm's performance — The encoding example should include the inactive term so that both examples are consistent (lines 113-115). Is there a reference some experimentation to back this up? We changed the text according to the suggestion..

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

2.2. Cost Function

The polynomial regression error considered so far accounts for the ability to predict the transformed testset. A known problem of using a cost function

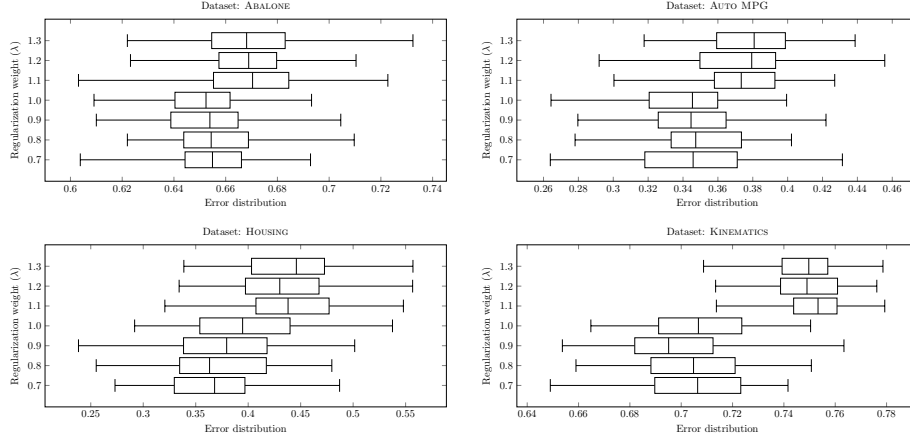


Figure 1: Error distribution by regularization exponent for common datasets. The box plots summarize error values of 75 simulations (60 iterations on a population size of 50) for each value of λ . Performance of the non-regularized EPR is plotted in the line $\lambda = 1$. Performance for $\lambda < 1$ is better than $\lambda > 1$ in all datasets.

154 based only in the dataset error (and of polynomial regressions in general) is the
 155 tendency to overfit training data. Excessive variance of the estimation method
 156 can be reduced by regularizing the error function with a penalty factor. Thus,
 157 to reduce polynomial complexity and variance by regularizing the size of the
 158 monomial set the error function from equation 2 is multiplied by a factor λ^k

$$J_{\text{reg}}(\Theta, \lambda; \mathcal{Q}, \mathcal{D}) = \lambda^k J_{\text{fit}}(\Theta; \mathcal{Q}, \mathcal{D}) \quad (4)$$

159 where k is the number of monomials in the polynomial. When $\lambda > 1$ polyno-
 160 mials with more monomials are penalized. The regularized extension of EPR is
 161 denoted by Evolutionary Polynomial Regression with Regularization (EPRR).

162 A simple exploration on the effect of the regularization parameter is depicted
 163 in Figure 1 where it is possible to observe that penalizing polynomial complexity
 164 (i.e. $\lambda < 1$) achieves better results than the opposite ($\lambda > 1$). This observation
 165 motivates further inquire, done in section 3. — please clarify this paragraph and
 166 justify this claim we rephrased the statement.

167 *2.3. Genetic Algorithm Parameterization*

168 — Please indicate all these values in detail We described the parameteriza-
169 tion in more detail.

170 — You should contrast obtained results in a table. Results are hard to
171 judge from Fig 3 alone.

172 — The number of 250 for iterations seems premature specially when one
173 looks at Fig 4. Stabilize? Really? This only seems valid in one case of
174 lambda. we changed the number of iterations to 2000, way behind stabiliza-
175 tion

176 In general GAs offer many possibilities with respect to the choice of genetic
177 operators and respective application rates, population evolution, *etc.* The re-
178 sults found here were obtained using the package `genalg` [27] with standard
179 operators (crossover and mutation) and population evolution defined by muta-
180 tion rate of 5% and 20% elitism between generations.

181 **3. Experimental Results**

182 Here is described the experiment setup used to gather and summarize the
183 empirical evidence that supports this comparative study of EPR and EPRR.
184 Evaluation is focused in error distribution and, besides EPR and EPRR, also
185 uses several common regression methods and datasets easily accessible in R, the
186 free software environment for statistical computing and graphics [28]². A small
187 consideration on the convergence speed concludes this section.

188 *3.1. Regression Methods and Datasets*

189 The EPRR method is ranked against several well-known learning algorithms
190 for regression, namely: non-regularized EPR, Linear Regression, Support Vector
191 Machines [29] with linear kernel, Regression Trees [30], Random Forest [31, 32]
192 and Conditional Inference Trees [33].

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

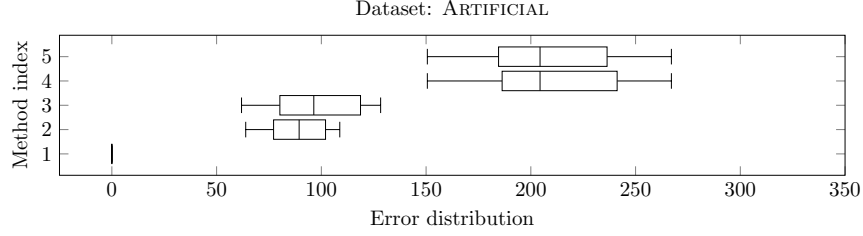


Figure 2: Testing polynomial discovery. The dataset is generated from a polynomial expression and, as shown, EPRR finds the exact generator structure: in line 1, the error box is centered in 0 and has width 0. The regression methods depicted are: 1. EPRR, 2. Linear Regression, 3. SVM, 4. Regression Trees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

The performance of each method is evaluated on several common datasets. From each dataset 70% of the observations are reserved for training purposes and the remaining observations used to estimate the error. To enhance the 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 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 one dependent variable, the median value of owner-occupied homes in thousands of USD;

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

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

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

3.2. Convergence speed

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

4. Conclusion and Future Work

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

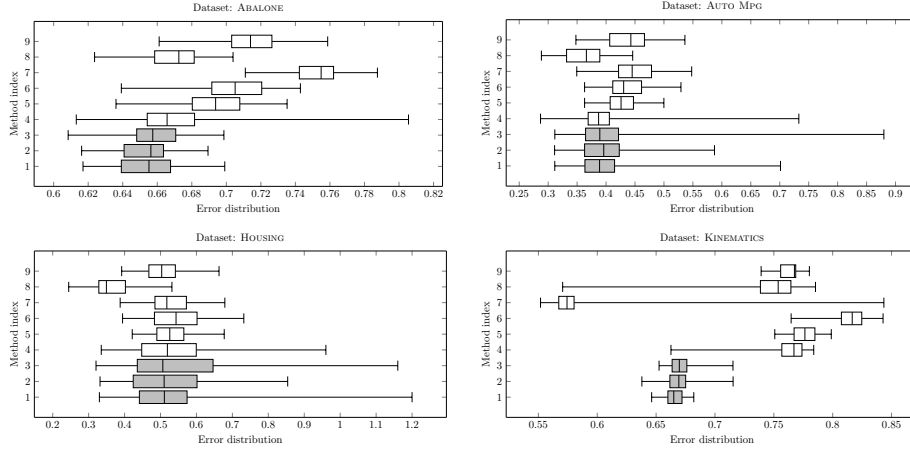


Figure 3: Summary results for different regression methods on common datasets. Although EPRR not always achieves the smallest expected error, performance is on-par with more sophisticated methods. The regression methods depicted in these figures are: 1. EPRR, $\lambda = 0.7$; 2. EPRR, $\lambda = 0.8$; 3. EPRR, $\lambda = 0.9$; 4. EPR (i.e. $\lambda = 1.0$); 5. Linear Regression; 6. SVM (linear kernel); 7. Regression Trees; 8. Random Forest (with 100 trees); 9. Conditional Inference Trees.

239 Comparing EPR and EPRR — the main article’s topic — the regularized
 240 version achieves much better results at ABALONE and especially KINEMATICS.
 241 On the HOUSING dataset errors are improved wrt EPR in a difference in means,
 242 resulted in a 95% HDI (Highest Density Interval) equal to $[0.001, 0.119]$ which,
 243 while borderline, achieves statistical significance. Only in the AUTO MPG
 244 dataset EPR achieves better results, even if not that different from EPRR.

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

251 A cross-validation procedure can be implemented to refine the appropriate
 252 parameter values to achieve better errors. Namely, the regularization parame-

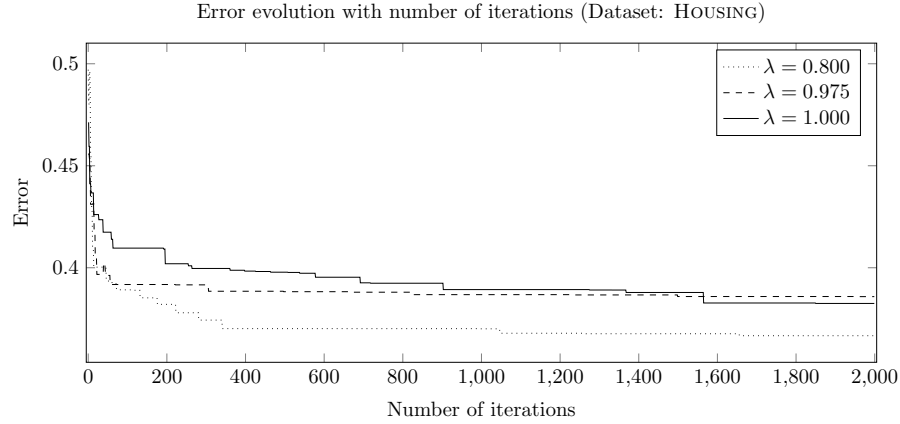


Figure 4: Learning curve: Error progress for the HOUSING dataset during a single execution of the genetic algorithm. The figure shows the fitness evolution for different regularization values. The population for both consists of 200 polynomials.

253 ter, λ , can be tested with several values, instead of being fixed at 0.8. Other
 254 parameters like mutation chance or the amount of elitism can also be tested.
 255 However, these type of tests need a low-level, fast implementation of EPR and
 256 are postponed to future investigation.

257 Acknowledgements

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

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

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

- 266 [1] B. Schölkopf, A. Smola, K.-R. Müller, Kernel principal component analysis,
 267 in: Artificial Neural Networks ICANN'97, Springer, 1997, pp. 583–588.

- 268 [2] Z. Liang, Y. Lee, Eigen-analysis of nonlinear pca with polynomial kernels.
- 269 [3] Y. Bao, Z. Hu, T. Xiong, A pso and pattern search based memetic algorithm
270 for svms parameters optimization, *Neurocomputing* 117 (2013) 98–106.
- 271 [4] S. Dehuri, B. B. Misra, A. Ghosh, S.-B. Cho, A condensed polynomial neu-
272 ral network for classification using swarm intelligence, *Applied Soft Com-*
273 *puting* 11 (3) (2011) 3106–3113.
- 274 [5] L. Mendes, P. d. Carvalho, Adaptive polynomial regression for colorimetric
275 scanner calibration using genetic algorithms, in: *Intelligent Signal Process-*
276 *ing, 2005 IEEE International Workshop on, IEEE, 2005*, pp. 22–27.
- 277 [6] J. Davidson, D. Savic, G. Walters, Method for the identification of explicit
278 polynomial formulae for the friction in turbulent pipe flow., *Journal of*
279 *Hydroinformatics* 1 (1999) 115–126.
- 280 [7] L. Sánchez, J. Otero, I. Couso, Obtaining linguistic fuzzy rule-based regres-
281 sion models from imprecise data with multiobjective genetic algorithms,
282 *Soft Computing* 13 (5) (2009) 467–479.
- 283 [8] L. Cséfalvayová, M. Pelikan, I. Kralj Cigić, J. Kolar, M. Strlič, Use of ge-
284 netic algorithms with multivariate regression for determination of gelatine
285 in historic papers based on FT-IR and NIR spectral data, *Talanta* 82 (5)
286 (2010) 1784–1790.
- 287 [9] K. Y. Chan, T. S. Dillon, C. K. Kwong, Modeling of a liquid epoxy mold-
288 ing process using a particle swarm optimization-based fuzzy regression ap-
289 proach, *Industrial Informatics, IEEE Transactions on* 7 (1) (2011) 148–158.
- 290 [10] A. Gálvez, A. Iglesias, J. Puig-Pey, Iterative two-step genetic-algorithm-
291 based method for efficient polynomial b-spline surface reconstruction, *In-*
292 *formation Sciences* 182 (1) (2012) 56–76.
- 293 [11] K. Y. Chan, C. Kwong, T. S. Dillon, Development of product design mod-
294 els using fuzzy regression based genetic programming, in: *Computational*

- Intelligence Techniques for New Product Design, Springer, 2012, pp. 111–128.
- [12] P. J. García Nieto, J. Alonso Fernández, F. de Cos Juez, F. Sánchez Lasheras, C. Díaz Muñiz, Hybrid modelling based on support vector regression with genetic algorithms in forecasting the cyanotoxins presence in the trasona reservoir (northern spain), *Environmental research*.
- [13] N. A. Barricelli, Numerical testing of evolution theories. part i: Theoretical introduction and basic tests, *Acta Biotheoretica* 16 (1-2) (1962) 69–98.
- [14] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, 1975.
- [15] J. R. Koza, *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, Vol. 1, MIT press, 1992.
- [16] Y. Bengio, Learning deep architectures for AI, *Foundations and trends in Machine Learning* 2 (1) (2009) 1–127.
- [17] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives.
- [18] D. Tarlow, I. Sutskever, R. S. Zemel, Stochastic k-neighborhood selection for supervised and unsupervised learning, *Journal of Machine Learning Research*.
- [19] K. Maertens, J. De Baerdemaeker, R. Babuška, Genetic polynomial regression as input selection algorithm for non-linear identification, *Soft Computing* 10 (9) (2006) 785–795.
- [20] T.-L. Yu, W.-K. Lin, Optimal sampling of genetic algorithms on polynomial regression, in: *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ACM, 2008, pp. 1089–1096.

- [21] C.-H. Wu, G.-H. Tzeng, R.-H. Lin, A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression, *Expert Systems with Applications* 36 (3) (2009) 4725–4735.
- [22] M. Hofwing, N. Strömberg, M. Tapankov, Optimal polynomial regression models by using a genetic algorithm, in: *Proceedings of the Second International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering Conference*, (Crete, Greece), 2011009, 2011.
- [23] B. Cetisli, H. Kalkan, Polynomial curve fitting with varying real powers, *Electronics and Electrical Engineering* 112 (6) (2011) 117–122.
- [24] J. Davidson, D. A. Savic, G. A. Walters, Symbolic and numerical regression: Experiments and applications, *Information Sciences* 150 (1) (2003) 95–117.
- [25] O. Giustolisi, D. Savic, Advances in data-driven analyses and modelling using epr-moga, *Journal of Hydroinformatics* 11 (3-4) (2009) 225–236.
- [26] R. Gupta, A. Bhunia, D. Roy, A GA based penalty function technique for solving constrained redundancy allocation problem of series system with interval valued reliability of components, *Journal of Computational and Applied Mathematics* 232 (2) (2009) 275 – 284.
doi:<http://dx.doi.org/10.1016/j.cam.2009.06.008>.
URL <http://www.sciencedirect.com/science/article/pii/S0377042709003549>
- [27] E. Willighagen, *genalg: R based genetic algorithm* (2012).
- [28] R Core Team, *R: A language and environment for statistical computing*.
URL <http://www.R-project.org/>
- [29] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, *e1071: Misc functions of the department of statistics (e1071), tu wienR package version 1.6-1*.
URL <http://CRAN.R-project.org/package=e1071>

- 349 [30] T. Therneau, B. Atkinson, B. Ripley, rpart: Recursive partitioningR pack-
350 age version 4.1-1.
- 351 [31] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, A. Zeileis, Conditional
352 variable importance for random forests, BMC Bioinformatics 9 (307).
353 URL <http://www.biomedcentral.com/1471-2105/9/307>
- 354 [32] C. Strobl, A.-L. Boulesteix, A. Zeileis, T. Hothorn, Bias in random forest
355 variable importance measures: Illustrations, sources and a solution, BMC
356 Bioinformatics 8 (25).
357 URL <http://www.biomedcentral.com/1471-2105/8/25>
- 358 [33] T. Hothorn, K. Hornik, A. Zeileis, Unbiased recursive partitioning: A
359 conditional inference framework, Journal of Computational and Graphi-
360 cal Statistics 15 (3) (2006) 651–674.