

Selection of Polynomial Features by Genetic Algorithms

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

Many applications require models that have no acceptable linear approximation and many nonlinear models are defined by polynomials. The use of genetic algorithms to find polynomial models is decades old but still poses challenges due to the complexity of the search and different definitions of optimal solution.

This paper describes a two-step method that uses genetic algorithms and linear regression to find empirical polynomial regressions. Experiments on common datasets show that, discounting the training computational effort, this method is quite competitive.

Keywords: polynomial regression, genetic algorithm, 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 are based on 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 ecology, cybernetics, robotics and other areas.

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 molding process (Chan et al., 2011), B-spline surface reconstruction (Gálvez et al., 2012), product design (Chan et al., 2012) or forecasting cyanotoxins presence in water reservoirs (García Nieto et al., 2013). These examples not only illustrate the wide spectrum of applications but, additionally, work in each one uses, at some point, a genetic algorithm.

Genetic algorithms (GA) where, 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 locally optimal parameters. Research related to genetic algorithms abound and spans from the 1950s seminal work of Nils Aall Barricelli (Barricelli, 1962) in the Institute for Ad-

26 vanced Study of Princeton to today’s principal area of study for thousands of
27 researchers, covered in hundreds of conferences, workshops and other meet-
28 ings. Perhaps the key impulse to GAs come from John Holland’s work and
29 his book “Adaptation in Natural and Artificial Systems” (Holland, 1975).

30 One interesting take on genetic algorithms, named *genetic programming*
31 by John Koza (Koza, 1992), proposed the use of GAs to search the syntactic
32 structure of complex functions. This syntactic structure search is keen to the
33 central ideas of deep learning (Bengio et al., 2013; Bengio, 2009), a subarea of
34 machine learning actually producing quite promising results (*e.g.* in Tarlow
35 et al. (2013)). It is also related to the work presented in this paper in the
36 sense that, unlike linear models that have a simple structure, $y = \sum_i \beta_i x_i$,
37 nonlinear (in particular polynomial) models pose an additional “structure”
38 search problem.

39 The idea of using GAs to find a polynomial regression is not new (Maertens
40 et al., 2006; Yu and Lin, 2008; Wu et al., 2009) but still generates original
41 research (Hofwing et al., 2011; Cetisli and Kalkan, 2011). In line with that
42 research this work describes a general method to find a polynomial regression
43 of a given dataset. The optimal regression minimizes a cost function that
44 accounts for both the root-mean-square error (error) and a regularization
45 factor to avoid over-fitting.

46 It turns out that, discarding the computational cost of training, the poly-
47 nomial regression method presented here, Genetic Algorithms for Polynomi-
48 als (GAPOLY), provides a quite competitive regression method. Indeed, it is
49 only systematically out-performed by random forests, an *ensemble* method.

50 The remainder of this paper is organized as usual: the next section de-

51 scribes the details of our method and is followed by a presentation of some
 52 performance results. The last section draws some conclusions and points
 53 future research tasks.

54 2. Genetic Algorithms for Polynomials

55 This section is dedicated to the description of an algorithm to find a
 56 polynomial regression from a given dataset. It starts with a brief introduction
 57 and outline of the algorithm and proceeds into core details as the encoding
 58 used to represent individual polynomial instances in the GA populations and
 59 the regularization of the cost function.

An usual representation of polynomials is

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

60 where each q_i is a monomial, $q_i = \prod_j x_j^{\alpha_{ij}}$, the exponents are non-negative
 61 integers, $\alpha_{ij} \in \mathbb{N}_0$, and the coefficients are real valued, $\theta_i \in \mathbb{R}$. For example
 62 $p(x_1, x_2, x_3) = 2x_1 + x_2x_3 + \frac{1}{2}x_1^2x_3$ has monomials $q_1 = x_1, q_2 = x_2x_3$ and
 63 $q_3 = x_1^2x_3$, coefficients $\theta_1 = 2, \theta_2 = 1$ and $\theta_3 = 1/2$ and exponents $\alpha_{1,1} =$
 64 $1, \alpha_{2,2} = 1, \alpha_{2,3} = 1, \alpha_{3,1} = 2, \alpha_{3,3} = 1$ and all other $\alpha_{ij} = 0$. The exponents
 65 alone are a matrix that defines the monomial structure of the polynomial,
 66 $A = [\alpha_{ij}]$.

For the example above the matrix of monomials is

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} x_1 \\ x_2x_3 \\ x_1^2x_3 \end{bmatrix}$$

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 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, the polynomial regression problem can be naturally split into two subproblems:

1. For a given set of monomials $\mathcal{P} = \{q_1, \dots, q_k\}$, find the regression coefficients $\theta_1, \dots, \theta_k$ 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 $\mathcal{P} = \{q_1, \dots, q_k\}$ a set of k monomial expressions over X_1, \dots, X_m . Define the hypothesis¹

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

and let the cost

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

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

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

86 It turns out that this problem can be solved as a usual linear regression
87 problem by expanding \mathcal{D} with columns that replicate the monomials in \mathcal{M} .

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

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

91 by solving the first problem. With a fitness of every instance, a GA will apply
92 genetic operators (usually mutation and crossover) to evolve the population
93 Q until a reasonable approximation of a local minimum is found. Notice
94 that the properties of GAs and linear regression entail that the composition
95 of GAs with linear regression, as defined in Algorithm 1, converges to a
96 polynomial that is a local minimum of the fitness function, encapsulated in
97 the error function J_{fit} .

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

101 2.1. Polynomial Encoding

102 The encoding for any polynomial will be as follows:

- 103 1. an initial segment detailing which monomials are active (the first mono-
104 mial is always active), this is represented in unary description, i.e., each
105 monomial is identified by a single bit

Algorithm 1 GAPOLY uses linear regression to find monomial coefficients that minimize the error over a dataset and GAs to explore the space of polynomials. The role of the linear regression step is to produce a fitting error that sorts the population. At exit the error of the fittest instance is bounded by ϵ .

```

function GAPOLY( $D, pop_0, \epsilon$ )
     $pop \leftarrow pop_0$ ;  $err \leftarrow 1.0 + \epsilon$ 
    while  $err > \epsilon$  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 2. the remaining bits are split into k sets of equal size, each one repre-
 107 senting a monomial
- 108 3. each monomial is split into m sets of d size each, *i.e.*, a variable
- 109 4. for each variable, the remaining bits are the binary description of the
 110 variable degree, *i.e.*, the maximum exponent is given by $2^d - 1$

111 Let's see an example: consider polynomial $x_1^3 x_3 + x_3^7 + x_1 x_2$ with $k =$
 112 4, $m = 3$ and $d = 3$. One possible encoding would be:

113 110 - 011,000,001 ; 000,000,111 ; 001,001,000 ; 110,010,101

114 (for reading purposes the semicolons separate monomials, the commas
 115 separate variables)

116 The first three bits inform that the second and third monomials are active
 117 while the fourth is not (as said, the first monomial is always active). This
 118 last monomial does not enter neither in the polynomial regression nor in the
 119 error (fitness) evaluation. However, it acts as a kind of junk DNA, becoming
 120 active when, in a future mutation or crossover, the third bit of the entire
 121 sequence flips from 0 to 1.

122 Let's interpret the first monomial description, 011,000,001. It is divided
 123 by three since $m = 3$. The first triple 011 is the binary description of the
 124 exponent of variable x_1 which is 3, so the first monomial includes x_1^3 . The
 125 second triple, 000, means that x_2 is not part of the monomial. The third
 126 triple 001 says that variable x_3 has exponent 1. The first monomial is $x_1^3x_3$.

127 Notice that all binary descriptions give rise to valid polynomials. However
 128 this is not a bijective mapping. For each polynomial there are multiple rep-
 129 resentations. For example, $x_1 + x_2$ and $x_2 + x_1$ have different representations.
 130 The authors considered that more complex mappings in order to force a one
 131 to one mapping would impact negatively in the algorithm's performance.

132 2.2. Cost Function

133 The polynomial regression error considered so far is based on the ability to
 134 predict the test set after the polynomial regression has found the appropriate
 135 coefficients θ_i for each one of the monomials q_i .

136 This error function tends to prefer more complex polynomials, namely in
 137 the number of monomials which provides the regression algorithm for more
 138 fitting possibilities. One way to balance this is to provide a regularization
 139 term into the error function. Our proposal is to include a multiplicative factor
 140 proportional to the number of monomials. Thus, the error from equation 1

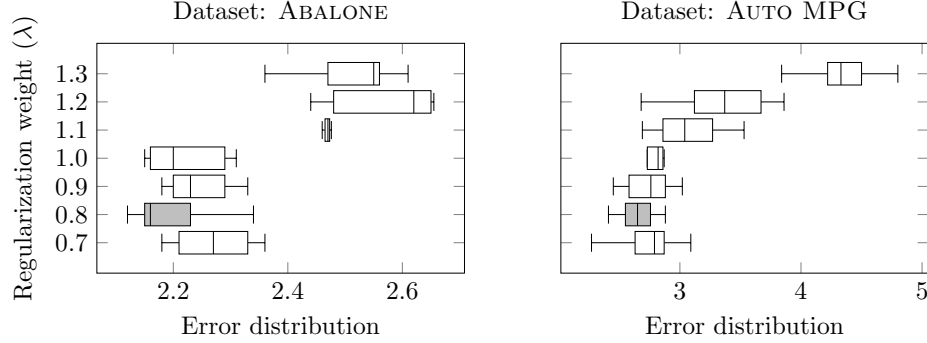


Figure 1: Error distribution by regularization exponent for the Abalone and Auto MPG datasets. The box plots summarize the error values of 10 runs for each value of λ . The smallest overall error is shown in gray and, in both cases, was achieved with $\lambda = 0.8$.

141 defines

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

142 where k is the number of monomials in the polynomial. When $\lambda > 1$ poly-
 143 nomials with more monomials are penalized.

144 Somewhat unexpectedly after some experiences it was found that lower
 145 values for λ provide better, even if marginal, results. Figure 1 shows regu-
 146 larization results for the Abalone and Auto MPG datasets with ten runs for
 147 each λ . The following section includes information about these datasets.

148 The typical inflection point lies around $\lambda = 0.8$. The dataset results for
 149 applying the proposed regression method use the regularization parameter
 150 with this value.

151 2.3. Genetic Operators

152 To perform the genetic algorithm it was used the R package genalg (Wil-
 153 lighagen, 2012). The operators were the standard ones: (a) crossover, *i.e.*,

154 a pair of solutions from the previous generations are combined by splitting
155 and mixing their respective representations, and (b) mutation, changing the
156 values of single bits; the mutation chance applied in the datasets was 5%.
157 There was also elitism between generations, *i.e.*, 20% of the best solutions
158 survive to the next generation.

159 3. Experimental Results

160 The results were found using R programming language (R Core Team,
161 2013)². To compare this paper’s proposed algorithm we applied the exact
162 same train and test samples using several well-known learning algorithms
163 for regression, namely: Linear Regression, Support Vector Machines (Meyer
164 et al., 2012), Regression Trees (Therneau et al., 2013), Conditional Infer-
165 ence Trees (Hothorn et al., 2006; Strobl et al., 2007, 2008) and Random
166 Forests (Liaw and Wiener, 2002)

167 In order to train and test the performance of GAPOLY several main-
168 stream datasets were used. For each dataset, we selected 70% for training
169 purposes and the remaining observations to make a test set in order to com-
170 pute the estimated error. To achieve more robust results, each dataset were
171 processed 25 times, each one with different samples for the train and test sets.
172 For the datasets with attribute values of different magnitudes, a preliminary
173 scaling was executed. The results below are box plots for the test set error
174 predictions over these different runs.

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

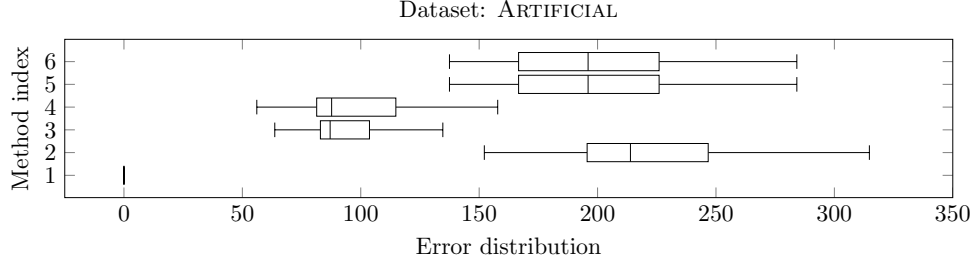


Figure 2: Results for Artificial dataset. As shown, GAPOLY was able to find the exact polynomial structure that generates the dataset, reducing the test set prediction error to zero. The regression methods depicted are: 1. GAPOLY, 2. Random Forests, 3. Linear Regression, 4. SVM, 5. Regression Trees and 6. Conditional Inference Trees

175 ARTIFICIAL this is an artificial dataset with four numeric features, x_1, \dots, x_4 ,
176 where x_1, x_3 are outcomes from Poisson random variables, and x_2, x_4
177 from Normal random variables. The dependent variable y is given by
178 expression $x_2x_4^2 + x_1^2x_3 + 5$. The dataset includes $n = 50$ observations.
179 This dataset was used in order to verify if GAPOLY was able to find
180 the polynomial relation, which the algorithm did (cf. figure 2). The
181 genetic algorithm run with a population of $n = 100$ solutions with
182 50 iterations for each run.

183 In the next datasets, the population of the genetic algorithm had
184 size $n = 250$ with 100 iterations for each run.

185 HOUSING: This data set concerns the task of predicting housing values
186 in areas of Boston. There are $m = 13$ continuous attributes and the
187 dependent variable is the median value of owner-occupied homes in
188 \$1000's. There are $n = 506$ observations.

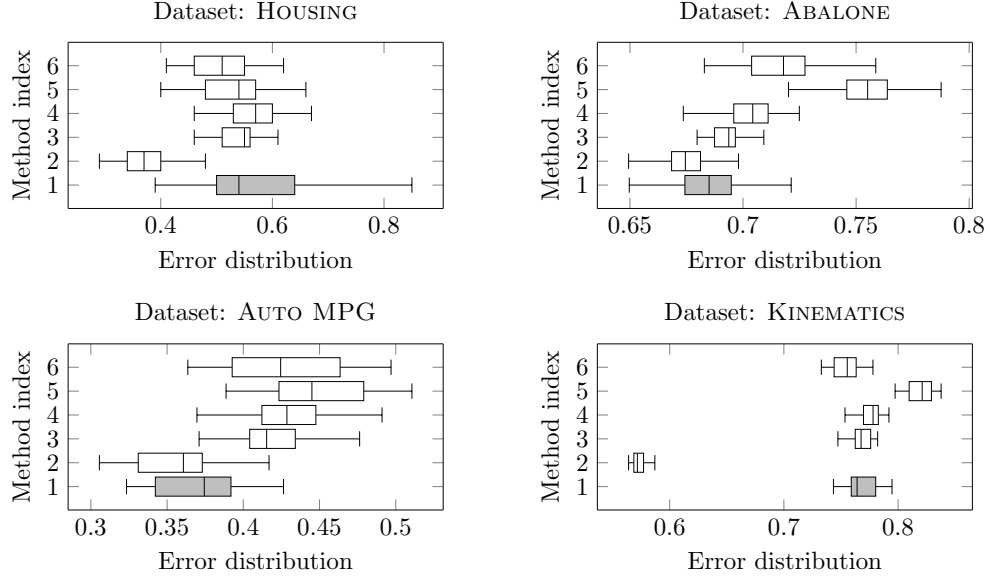


Figure 3: Summary results for different regression methods on the Housing, Abalone, Auto MPG and Kinematics datasets. Random Forests, an ensemble method (“2” in the plots), has the better results. However GAPOLY achieves similar errors in two of the datasets. For the remaining two GAPOLY produces similar errors to the other classic regression methods. The regression methods depicted in these figures are: 1. GAPOLY, 2. Random Forests, 3. Linear Regression, 4. SVM, 5. Regression Trees and 6. Conditional Inference Trees

189 Just as an example of the model the GAPOLY algorithm outputs: in
190 this dataset the best polynomial,

$$y = -0.12x_6x_9^4 + 0.78x_6 - 0.4x_9^2x_{13} + 0.17x_{13}^2 - 0.044$$

191 where the attributes mean: x_6 , RM average number of rooms per
192 dwelling; x_9 , RAD index of accessibility to radial highways; x_{13} , Lower
193 status of the population.

194 For comparison, if we access the mean decrease in accuracy found by
 195 Random Forests, the most important attributes are — in decreasing
 196 order — x_{13}, x_6, x_5 (but x_5 is already considered 4 times less important
 197 than x_6). Both algorithms agree in two of their three most important
 198 attributes.

199 **ABALONE** This dataset can be used to predict the age of a abalone shell
 200 using the given $m = 8$ numeric attributes concerning several physical
 201 measurements. There are $n = 4177$ observations.

202 **AUTO MPG** This dataset is used to predict fuel consumption in miles per
 203 gallon, based on two discrete and five continuous attributes ($m = 7$).
 204 There are $n = 398$ observations.

205 **KINEMATICS** This dataset is concerned with the realistic simulation of the
 206 forward kinematics of an 8 link robot arm. The task is to predict
 207 the distance of the end-effector from a target using $m = 8$ continuous
 208 attributes. There are $n = 8192$ observations.

209 *3.1. Convergence speed*

210 The GA quickly proceeds in the first 50 to 100 generations to reasonable
 211 error rates. Then, it proceeds slower achieving best solutions with marginal
 212 error reduction. Since the entire learning process takes some time, in the
 213 current R implementation, placing a limit between 50 to 100 generations
 214 already achieves good results, relative to higher iteration values. Figure 4
 215 shows a typical error evolution for the dataset Abalone given two different
 216 values for λ .

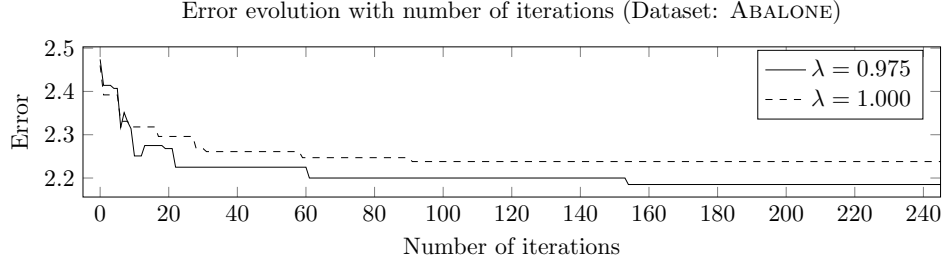


Figure 4: Error progress for Abalone dataset during a single execution of the genetic algorithm. The figure shows the fitness evolution for two different regularization values. The population for both consisted of 200 polynomials. The error values seem to stabilize around iteration 250.

217 4. Conclusion and Future Work

218 Of all the non-ensemble regression methods considered, the proposed
 219 method is the one that shows, with one exception, the lowest error. Only
 220 Random Forest outperforms GAPOLY systematically (it also outperforms
 221 all the other regression algorithms). The single non-ensemble exception —
 222 besides the artificial dataset that uses a straightforward polynomial relation
 223 — is the Auto MPG dataset where GAPOLY has comparable results. This
 224 is evidence that applying standard genetic operators for polynomial model
 225 searching is a viable tool for regression purposes.

226 For complexity considerations GAPOLY demands some processing time.
 227 On a current quad-core computer, processing the Kinematics dataset (with
 228 8k observations) takes approximately 5 minutes. The processing time can
 229 probably be speeded by one to two orders in magnitude if the process is
 230 implemented in a low level programming language like C++. However, speed
 231 optimization was not the focus of this article.

232 A cross-validation procedure can be implemented to refine the appropri-
233 ate parameter values to achieve better errors. Namely, the regularization
234 parameter, λ , can be tested with several values, instead of being fixed at 0.8.
235 Other parameters like mutation chance or the amount of elitism could also
236 be tested. However, these type of tests need a low-level, fast implementation
237 of GAPOLY.

238 Acknowledgements

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

242 The datasets used herein were selected from Luís Torgo's data repository,
243 <http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html>. Most come
244 originally from UCI ML repository, <http://archive.ics.uci.edu/ml/>.

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

247 Barricelli, N.A., 1962. Numerical testing of evolution theories. part i: Theo-
248 retical introduction and basic tests. Acta Biotheoretica 16, 69–98.

249 Bengio, Y., 2009. Learning deep architectures for AI. Foundations and trends
250 in Machine Learning 2, 1–127.

251 Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A
252 review and new perspectives .

253 Cetisli, B., Kalkan, H., 2011. Polynomial curve fitting with varying real
254 powers. Electronics and Electrical Engineering 112, 117–122.

- 255 Chan, K.Y., Dillon, T.S., Kwong, C.K., 2011. Modeling of a liquid epoxy
256 molding process using a particle swarm optimization-based fuzzy regression
257 approach. *Industrial Informatics, IEEE Transactions on* 7, 148–158.
- 258 Chan, K.Y., Kwong, C., Dillon, T.S., 2012. Development of product design
259 models using fuzzy regression based genetic programming, in: *Computational Intelligence Techniques for New Product Design*. Springer, pp.
260 111–128.
- 262 Cséfalvayová, L., Pelikan, M., Kralj Cigić, I., Kolar, J., Strlič, M., 2010.
263 Use of genetic algorithms with multivariate regression for determination of
264 gelatine in historic papers based on FT-IR and NIR spectral data. *Talanta*
265 82, 1784–1790.
- 266 Davidson, J., Savic, D., Walters, G., 1999. Method for the identification of
267 explicit polynomial formulae for the friction in turbulent pipe flow. *Journal*
268 *of Hydroinformatics* 1, 115–126.
- 269 Gálvez, A., Iglesias, A., Puig-Pey, J., 2012. Iterative two-step genetic-
270 algorithm-based method for efficient polynomial b-spline surface recon-
271 struction. *Information Sciences* 182, 56–76.
- 272 García Nieto, P.J., Alonso Fernández, J., de Cos Juez, F., Sánchez Lasheras,
273 F., Díaz Muñoz, C., 2013. Hybrid modelling based on support vector
274 regression with genetic algorithms in forecasting the cyanotoxins presence
275 in the trasona reservoir (northern spain). *Environmental research* .
- 276 Hofwing, M., Strömberg, N., Tapankov, M., 2011. Optimal polynomial
277 regression models by using a genetic algorithm, in: *Proceedings of the*

278 Second International Conference on Soft Computing Technology in Civil,
 279 Structural and Environmental Engineering Conference,(Crete, Greece),
 280 2011009.

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

284 Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A
 285 conditional inference framework. Journal of Computational and Graphical
 286 Statistics 15, 651–674.

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

289 Liaw, A., Wiener, M., 2002. Classification and regression by randomforest.
 290 R News 2, 18–22. URL: <http://CRAN.R-project.org/doc/Rnews/>.

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

294 Mendes, L., Carvalho, P.d., 2005. Adaptive polynomial regression for colori-
 295 metric scanner calibration using genetic algorithms, in: Intelligent Signal
 296 Processing, 2005 IEEE International Workshop on, IEEE. pp. 22–27.

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

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

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

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

308 Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T., 2007. Bias in ran-
 309 dom forest variable importance measures: Illustrations, sources and a so-
 310 lution. *BMC Bioinformatics* 8. URL: [http://www.biomedcentral.com/](http://www.biomedcentral.com/1471-2105/8/25)
 311 [1471-2105/8/25](http://www.biomedcentral.com/1471-2105/8/25).

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

315 Therneau, T., Atkinson, B., Ripley, B., 2013. rpart: Recursive partitioning
 316 R package version 4.1-1.

317 Willighagen, E., 2012. genalg: R based genetic algorithm.

318 Wu, C.H., Tzeng, G.H., Lin, R.H., 2009. A novel hybrid genetic algorithm for
 319 kernel function and parameter optimization in support vector regression.
 320 *Expert Systems with Applications* 36, 4725–4735.

321 Yu, T.L., Lin, W.K., 2008. Optimal sampling of genetic algorithms on poly-
322 nomial regression, in: Proceedings of the 10th annual conference on Ge-
323 netic and evolutionary computation, ACM. pp. 1089–1096.