The Controlled Random Search Algorithm in Optimizing Regression Models

Ivan KŘIVÝ and Josef TVRDÍK

University of Ostrava, Dvořá kova 7, 701 03 Ostrava, Czech Republic

Summary: This paper deals with the problems of controlled random search algorithms (CRS algorithms) and their use in regression analysis. A modified CRS algorithm of Price is described, which is more effective when compared with the original algorithm in optimizing regression models, first non-linear ones. The principal modification consists in randomizing the search for the next trial points. Some results of testing the algorithm, using both real and modeled data, are given to illustrate its possibilities when estimating the parameters of non-linear regression models.

(SSNinCSDA 20, 199-204 (1995))

Keywords: random search algorithm, non-linear regression models

Received: November 1994 *Revised: February 1995*

I. Introduction

The notion of the CRS algorithm was introduced by Price [9] for his seeking algorithm for the global minimum of a multimodal function, f, of d variables. The algorithm combines the simple random search and the simplex method [8] into a single continuous process. When minimizing $f(\mathbf{P})$ subject to $\mathbf{P} \in \Omega$, where $\mathbf{P}$ is a d-dimensional vector and Ω a bounded set in $\mathbf{R}^d$, Price's algorithm is as follows:

1. Set $k = 0$, load storage of size N by generating randomly points $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_N \in \Omega$ and store also $f(\mathbf{P}_i)$ for $i = 1, 2, \ldots, N$.
2. Choose in random $d+1$ linearly independent points ($N >> d$) from the current configuration of N points in store, generate the new trial point $\mathbf{P} \in \Omega$ by the relation

$$\mathbf{P} = 2\,\mathbf{G} - \mathbf{P}_{d+1}, \qquad (1)$$

where $\mathbf{P}_{d+1}$ is one (randomly taken) pole of the simplex $\mathbf{P}_1\ \mathbf{P}_2\ \ldots\ \mathbf{P}_{d+1}$ and $\mathbf{G}$ the centroid of the remaining d poles of the simplex, and determine $f(\mathbf{P})$.

3. If $f(\mathbf{P}) < f(\mathbf{M})$, $\mathbf{M}$ being the point with the greatest function value of the N points stored, then replace $\mathbf{M}$ with $\mathbf{P}$.
4. Set $k = k + 1$ and return to Step 2.

The algorithm was originally programmed in Basic and run on a PDP 10/20 minicomputer and on a Cyber 72 computer [9].

A FORTRAN procedure based on Price's algorithm was also presented recently by Conlon [1].

II. Modifications of Price's algorithm

Price's algorithm is very simple and easily programmed on the PC, but its convergence is usually slower compared with optimization methods based on function derivatives. It is possible to speed up its convergence in the following ways:

- by selecting $d+1$ simplex vertices not from the complete configuration of N points in store, but from some of its subset containing points with the lowest function values;
- by selecting as the simplex pole $\mathbf{P}_{d+1}$ in Eqn. (1) that point (from the set of $d+1$), which has the largest function value.

Such modifications improve convergence, but at the same time tend to increase the risk of finding a local minimum instead of a global one.

We have investigated several quite different modifications of the original Price's algorithm. Our modifications [3] are based on generating the next trial point $\mathbf{P}$ according to the formula

$$\mathbf{P} = \mathbf{G} - \Gamma(\alpha)\,(\mathbf{P}_{d+1} - \mathbf{G}) \qquad (2)$$

instead of using Eqn. (1). Regarding the multiplication factor $\Gamma(\alpha)$, the following assumptions have been tested in detail:

- $\Gamma(\alpha) = \alpha$,
- Γ is a random variable with uniform distribution on the interval $(0, \alpha)$,
- Γ is a random variable with normal distribution $N(\alpha, 1)$,

α being a positive constant. Special attention is paid to investigating the effect of the Γ-factor on the running time needed for reaching acceptable optimization results. As shown later in Section 7, the best results were obtained when considering Γ distributed uniformly with α ranging from 4 to 8.

III. Optimization criteria

When estimating the parameters of regression models, the following three functions are usually considered as optimization criteria to be minimized:

- residual sum of squares

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \; ; \tag{3}$$

- sum of absolute deviations

$$SAD = \sum_{i=1}^{n}|y_i - \hat{y}_i| \; ; \tag{4}$$

- maximum absolute deviation

$$MAD = \max|y_i - \hat{y}_i| \quad \text{for } i = 1, 2, ..., n. \tag{5}$$

where y_i ($i = 1, 2, ... , n$) denote the observed values of the dependent variable, $\hat{y}_i$ their estimates calculated from the estimates of regression parameters and n the total number of observations (sample size).

More robust criteria for optimization (for example median of squares [10], trimmed squares and S-estimators [11]) can be used as well.

A vector-type criterion is also applicable in evaluating the quality of regression model variants. This criterion enables finding values of the model parameters that correspond to the optimum with respect to all of its components.

When using the vector-type criterion, it is desirable to define a subsidiary scalar criterion for ordering the model variants according to their quality. Weinberger [12] recommends evaluating the value of the scalar criterion for a given regression model variant (point) in store by using the expression

$$\sum_{i=1}^{m} w_i D_i \; , \tag{6}$$

where w_i is a subjective weight of i-th component of the vector criterion ($i = 1, 2, ... , m$) and D_i the number of variants that are dominated in this component by the variant considered.

IV. Stop Criterion

The CRS algorithm given in Section 1 does not include any particular stop criterion. However, it is clear that the criterion has to be evaluated within each iteration of the optimizing algorithm.

We propose to stop the optimization process, when

$$\frac{f(P^N) - f(P^1)}{f_0} \le \varepsilon \tag{7}$$

ε_0 being a positive input value, $f(P^N)$ and $f(P^1)$ the greatest and the least value of the optimization criterion for the current iteration, resp. [3]. f_0 denotes an appropriate constant factor whose value is determined by the variability of the dependent model variable. For example, when using RSS as the optimization criterion, the f_0 factor is equal to the total sum of squares, i.e.

$$f_0 = \sum_{i=1}^{n}(y_i - \bar{y})^2 .$$

The stop condition (7) proves to be more useful, when compared with that defined by Conlon [1] in terms of $f(P^N) - f(P^1)$.

V. Input to the algorithm

The input parameters for our modification of the CRS algorithm consist of:

- number of points, N, to hold in the store,
- value of α in Eqn. (2),
- value of ε_0 in Eqn. (7).

VI. Implementation of the algorithm

Our CRS algorithm was implemented (as a core of a program named MOR) in Turbo Pascal, version 6.0, using the ESTAT environment [13].

The input data for the program include:

- definition of the regression model,
- boundaries for the individual regression parameters,
- regression data in a text file (no more than 10 000 data items),
- choice of the optimization criterion (RSS, SAD, MAD or vector-type criterion with the individual weights of its components),
- input for the modified CRS algorithm (given in Section 5).

The regression model is defined by writing the corresponding regression function (as an arithmetic expression in Pascal source form) on a separate input file. Therefore, the program unit containing this file together with the main program has to be compiled once again for each new regression model under consideration.

Regarding the algorithm input, the following choices are usually found to be acceptable: $N = 5d$, $\alpha = 8$, and $\varepsilon_0 = 1E{-}16$ (when minimizing RSS). All input is introduced interactively from the keyboard.

The course of the optimization process (number of iterations, running time, values of the minimized criterion, regression parameters and R-squared) is displayed on the screen every other second.

The optimization run stops as soon as the condition (7) is satisfied. The user is also allowed to stop the calculations interactively, e.g. in case when the estimates of the regression parameters remain constant for a sufficiently long time period. The final results are saved on a text file for the sake of a subsequent evaluation, if there is a need.

The MOR source program and user guide are available for the interested reader on request per E-mail to tvrdik@oudec.osu.cz or per regular mail at the address of the authors.

VII. Test Results

Both real and modelled regression data were used to illustrate the possibilities of our algorithm (MOR program) when estimating the parameters of nonlinear regression models. All the calculations were performed on PC 486DX, 33 MHz, under MS DOS, version 6.0. The results for some well-known testing

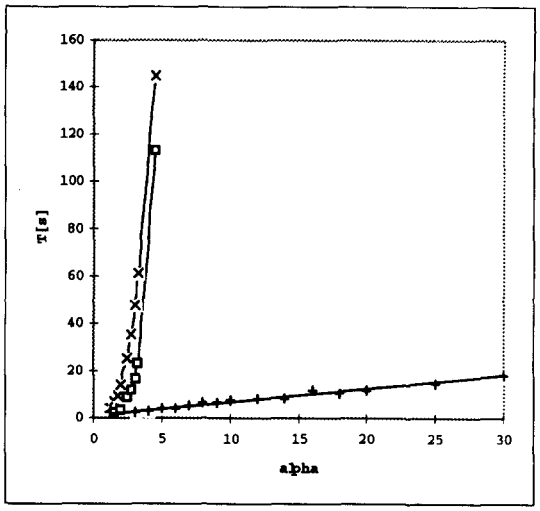

Figure 1: Empirical dependence of T on α for $\Gamma(\alpha) = \alpha$ (×), $\Gamma \sim N(\alpha, 1)$ ( ), and Γ uniform on $(0, \alpha)$ (+): data [4]-Example 7, $N = 15$, $\varepsilon_0 = 1E\text{-}16$.

examples are summarized in Figures 1, 2, and 3 as well as in Tables 1 and 2, the residual sum of squares (RSS) being chosen as an optimization criterion.

Fig. 1 shows the running time, T, as a function of the input parameter α for different distributions of Γ (see Eqn. (2)). For $\Gamma(\alpha) = \alpha$ as well as for Γ distributed normally, the running time increases dramatically with the increasing value of α. On the other hand, when considering Γ distributed uniformly on the interval $(0, \alpha)$, the values of T grow almost linearly with increasing α, the observed line having a relatively small slope. This is true for all the data tested, which indicates a privileged standing of uniform distribution among the three distributions under consideration. The use of our algorithm with Γ distributed uniformly on $(0, \alpha)$ permits reducing the value of N as compared with the proposal of Price [9] and, therefore, shortening the running time.

Figures 2 and 3 illustrate in more detail the effect of α on the values of T provided that Γ has uniform distribution on $(0, \alpha)$. It is clear that all the graphs T vs. α are very close to those of linear function, except perhaps, the parts for small α not exceeding approx. 5. Starting from our experience in optimizing nonlinear regression models, we can recommend working with α ranging from 4 to 8.

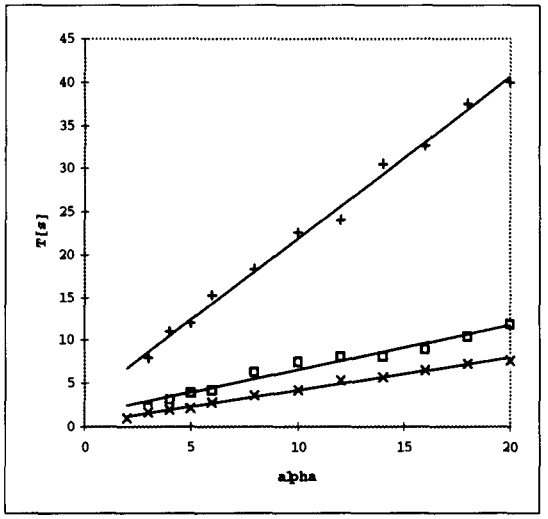

Figure 2: Graphs T vs. α. Γ distributed uniformly on (0, α), for data [2] (×), data [4]-Example 7 ( ), and data [6]-Model 5 (+): N = 15, ε_0 = 1E-16.

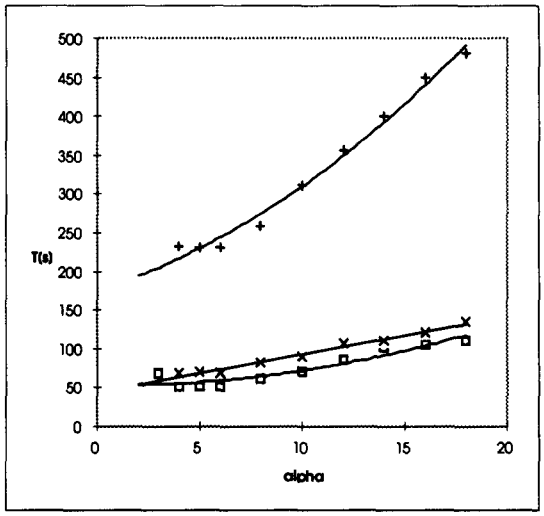

Figure 3. Graphs T vs. α. Γ distributed uniformly on (0, α), for data [4]-Example 8 ( ), data [5]-Model IV (×) and data [5]-Model V (+): N = 15, ε_0 = 1E-16.

Table 1: The results of testing the MOR program on well-known published data: N = 5d, α = 8, ε_0 = 1E-16.

Reference	Regression model	Time [s]	RSS	Parameters
[4] Example 1	$\beta_1\beta_3x_1/(1 + \beta_1x_1 + \beta_2x_2)$	4.6	4.355E-05	3.1315 15.159 0.7801
[4] Example 4	$\beta_3(\exp(-\beta_1x_1) + \exp(-\beta_2x_2))$	13.7	7.471E-05	13.241 1.5007 20.100
[4] Example 5	$\beta_3(\exp(-\beta_1x_1) + \exp(-\beta_2x_2))$	9.9	1.252	32.000 1.5076 19.920
[4] Example 7	$\beta_1 + \beta_2\exp(\beta_3x)$	5.8	5.986E-03	15.673 0.9994 0.0222
[4] Example 8	$\beta_1\exp(\beta_2/(\beta_3 + x))$	62.0	87.95	0.00561 6181.4 345.22
[2]	$\exp(\beta_1x) + \exp(\beta_2x)$	3.6	124.4	0.2578 0.2578
[5] Model IV	$\beta_1\exp(\beta_3x) + \beta_2\exp(\beta_4x)$	73.9	129.0	1655.2 3.4E07 -0.6740 -1.8160
[5] Model V	$\beta_1 x^{\beta_3} + \beta_2 x^{\beta_4}$	283.4	2.981E-05	0.00414 3.8018 2.0609 0.2229

Table 2: The results of testing the MOR program on the unpublished data of Militký [6]: $N = 5d$, $\alpha = 8$, $\varepsilon_0 = 1E\text{-}16$.

Reference	Regression model	Time [s]	RSS	Parameters
Model 2	$\exp(\beta_1 x) + \exp(\beta_2 x)$	5.4	8.896E-03	0.2807 0.4064
Model 3	$\beta_1 + \beta_2 \exp((\beta_3 + \beta_4 x)^{\beta_5})$	45.4	0.9675	9.3593 2.0292 1.3366 0.4108 0.3551
Model 4	$\beta_1 \exp(\beta_3 x) + \beta_2 \exp(\beta_4 x)$	19.3	3.179E-04	47.971 102.05 -0.2466 -0.4965
Model 5	$\beta_1 x^{\beta_2} + \beta_3^{\beta_2/x}$	17.6	4.375E-03	0.05589 3.5489 1.4822
Model 6	$\beta_1 + \beta_2 x^{\beta_3} + \beta_4 x^{\beta_5} + \beta_6 x^{\beta_7}$	275.0	1.694E-02	1.9295 2.5784 0.8017 -1.2987 0.8990 0.011915 3.0184
Model 7	$\beta_1 \ln(\beta_2 + \beta_3 x)$	44.2	7.147E-05	2.0484 18.601 1.8021

VIII. Conclusions

Our modification of the original Price's algorithm (see Eqn. (2)) with Γ distributed uniformly on $(0, \alpha)$ makes possible a significant decrease in the value of the tuning parameter N as compared with the value recommended by Price, and therefore reduces the running time.

The experiments showed that our algorithm (MOR program) provides results comparable with the best ones obtained using special techniques based on the calculation of criterial function derivatives. This algorithm proved to be successful even in the cases when, according to the results of Militký [7], most of the commercial statistical packages fail.

Because the MOR program provides no information on the accuracy of the parameter estimates, it is desirable, where possible, to complete the regression results by using commonly used regression software.

References

[1] Conlon, M. (1992): The Controlled Random Search Procedure for Function Optimization. *Commun. Statist.-Simula. Comput*, 21, 919-923.

[2] Jennrich, R.I. & P.T. Sampson (1968): Application of Stepwise Regression to Non-Linear Estimation. *Technometrics* 10, 63-72

[3] K ř ivý, I. & J. Tvrdík (1994): Multicriterial Optimization of Regression Models, in: R. Dutter and W.

Grossmann (Eds.), *COMPSTAT 1994. Short Communications and Posters.* Vienna: Univ. of Technology 25-26

[4] Meyer, R.R. & P.M. Roth (1972): Modified damped least squares: An algorithm for non-linear estimation. *J. Inst. Math. Applics.* 9, 218-233

[5] Militký, J. and M. Meloun (1994): Modus operandi of the least squares algorithm MINOPT. *Talanta* 40, 269-277

[6] Militký, J., private communication.

[7] Militký, J. (1994): Nonlinear Regression on Personal Computers, in: R. Dutter and W. Grossmann (Eds.), *COMPSTAT 1994. Proceedings in Computational Statistics.* Heidelberg: Physica -Verlag, 395-400

[8] Nelder, J.A. & R. Mead (1964): A simplex method for function minimization. *Computer J.* 7, 308-313

[9] Price, W.L. (1976): A controlled random search procedure for global optimization. *Computer J.* 20, 367-370

[10] Rousseeuw, P. (1984): Least Median of Squares Regression. *J. Amer. Statist. Assoc.* 79, 871-879

[11] Rousseeuw, P. and V. Yohai (1984: Robust Regression by Means of *S*-Estimators, in: *Robust and Nonlinear Time Series Analysis. Lecture Notes Statistics, Vol. 26.* New York: Springer Verlag, 256-272

[12] Weinberger, J. (1987): Extremization of vector criteria of simulation models by means of quasi-parallel hanling. *Comp. and Art. Intel.* 6, 71-19

[13] Žvaček, J.and Ře zanková H. (1990): ESTAT. Statistical programming environment in Turbo Pascal. In: *COMPSTAT 1990. Software Catalogue.* Heidelberg: Physica -Verlag, 19-20