Journal of Statistical Software

August 2015, Volume 66, Code Snippet 2. *http://www.jstatsoft.org/*

MLGA: A SAS Macro to Compute Maximum Likelihood Estimators via Genetic Algorithms

Francisco Juretig
The Nielsen Company

Abstract

Nonlinear regression is usually implemented in SAS either by using `PROC NLIN` or `PROC NLMIXED`. Apart from the model structure, initial values need to be specified for each parameter. And after some convergence criteria are fulfilled, the second order conditions need to be analyzed. But numerical problems are expected to appear in case the likelihood is nearly discontinuous, has plateaus, multiple maxima, or the initial values are distant from the true parameter estimates. The usual solution consists of using a grid, and then choosing the set of parameters reporting the highest log-likelihood. However, if the amount of parameters or grid points is large, the computational burden will be excessive. Furthermore, there is no guarantee that, as the number of grid points increases, an equal or better set of points will be found. Genetic algorithms can overcome these problems by replicating how nature optimizes its processes. The `MLGA` macro is presented; it solves a maximum likelihood estimation problem under normality through `PROC GA`, and the resulting values can later be used as the starting values in SAS nonlinear procedures. As will be demonstrated, this macro can avoid the usual trial and error approach that is needed when convergence problems arise. Finally, it will be shown how this macro can deal with complicated restrictions involving multiple parameters.

Keywords: genetic algorithms, likelihood, initial values, SAS.

1. Introduction

Nonlinear regression entails certain additional difficulties compared to linear regression. As either the likelihood (or the sum of squares) need to be maximized/minimized numerically, problems may appear. They may be caused by different reasons, and will in general have an impact on the final estimates. Furthermore, if the likelihood is almost flat at some areas, traditional derivative-based algorithms may encounter difficulties progressing until a maximum is found. On a different front, if the starting values are distant from this optimum, the

algorithm may not head towards the correct set of parameters. If the likelihood has multiple maxima, it may well happen that the algorithm will head towards the local maximum that is closest to the initial value used.

In addition, derivative-based methods will iterate until the norm of the gradient is close to zero, but there is no guarantee that the second order conditions are to be fulfilled when that happens. The Hessian may well be not of full rank, or positive definite (for example in saddle-points). It is evident that a good choice of the initial values can mitigate or evade these problems. On the other hand, an incorrect choice may push these algorithms towards these situations.

Most modern software implements the possibility of using a grid to choose these starting values. Nevertheless, it is to be noted that there is no guarantee that, as the number of grid points increases, a better solution can be found. A second problem is that, as the number of grid-points or variables increase, the number of grid-points that are to be evaluated by the algorithm will increase exponentially, rendering the approach useless in some scenarios.

Genetic algorithms do not need any of the assumptions required by derivative-based methods, namely that the function is continuous up to its second derivative, that the initial values are close to the maximum of the function to be optimized, that a second order approximation to the function is good, that the function does not have sudden spikes or plateaus – except for the only requirement that the function can be evaluated in all its domain. In addition, they can avoid situations with multiple maxima, since the strategy resides in evaluating the function to be optimized (in this case the log-likelihood) at multiple locations, and combining that information to breed better generations. It is thus evident that a good possibility could be using a genetic algorithm to compute these initial values, and then pass them on to some derivative based procedure. In this fashion, the final value reported by the genetic algorithm may be enhanced, the confidence intervals for each parameter would be reported, and the second order conditions could be checked.

The use of genetic algorithms in nonlinear regression is not new. Kapanoglu, Koc, and Erdogmus (2007) find out that the convergence properties of this approach are satisfactory and the parameters that have the most influence on the residual sum of squares converge faster. Dorsey and Mayer (1995) analyze the advantages of genetic algorithms in terms of econometric models focusing on how local maxima can be avoided. Pan, Chen, Kang, and Zhang (1995) highlight that these algorithms do not require auxiliary information, as traditional methods do, and that they provide a very flexible methodology with convincing results. Gulsen, Smith, and Tate (1995) study how genetic algorithms can be applied to diverse curve fitting models with multiple parameters. They find that genetic algorithms provide good results and are robust to random seeds, dataset sizes, or amount of variables. Olinsky, Quinn, Mangiameli, and Chen (2004) apply genetic algorithms to nonlinear least squares problems.

`MLGA` macro reads a model structure specified by the user, and writes the corresponding instructions to SAS (SAS Institute Inc. 2008) `PROC GA`. This macro can avoid many of the aforementioned problems and usually finds the maximum of the log-likelihood. After this, practically no further steps will be needed by a routine that maximizes the log-likelihood (such as `PROC NLMIXED`). Consequently, the trial and error approach until some set of initial values work properly is no longer required.

2. Genetic algorithms

Over the past years, the reduction of computing prices and the availability of good software implementations have led to a major increase in the use of stochastic search algorithms. These methods share the property of using random numbers for finding the maximum/minimum of a function. On most occasions, the solutions that are found using these algorithms cannot be exactly replicated unless the exact same seed (which is used in the random number generators) is used. Most of these algorithms need much less stringent assumptions than their non-stochastic counterparts, or in some occasions, they do not require any assumptions at all. Although these methods can be quite different, a specific subset of these algorithms are built on the idea of replicating how nature optimizes processes. In particular, genetic algorithms are based on the idea that each successive generation of a population can be interpreted as the outcome of an optimization process.

The process of building better biological entities can be conceptualized as selecting an initial population of individuals and then choosing usually the fittest of them iteratively. Consequently, the set of chosen individuals will be composed of a larger group of the fittest individuals, and a smaller one containing the rest of the population. This is of prime importance, and it is likely one of the few ways nature tries to avoid local maxima. If all the chosen individuals were to be part of the very best group, the population would probably fall quickly into some local maximum, as the selected members of the population would probably be almost identical. The selected population is thus chosen by weighting the genetic diversity versus choosing the best members. After this selection is done, pairs are formed in order to build offspring. Thus, the offspring carry a combination of genetic material of two individuals, with some additional mutations that will usually render them unique. In this fashion, again, the population is ensured to have some diversity. In practical terms, nature loops through this process *ad infinitum*. Genetic algorithms mimic this selection process by assuming that the members of the population are sets of parameters, using the pseudo-random number generator for computing the transition probabilities, evaluating the fit of the solutions by some mathematical function that is to be optimized, and assuming there is a terminal criterion. It is to be noted that practically no assumption is needed for applying this tool, except for the obvious assumption that the fitness function can be computed for each observation. The usual assumptions made by derivative-based methods (i.e., that the function is continuous up to its second derivative, that the Hessian can be inverted, or that the initial values are close to the optimal solutions) are not needed here. Much more detailed discussions about genetic algorithms can be found in Whitley (1994); Forrest (1996); Wright (1991).

3. Program description

The macro was tested on SAS 9.2 64-bit and the examples presented here were obtained using a SunOS 5.10 Unix machine. This macro should be included in the usual way `%include MLGA.sas`. After that, it should be called

```
%MLGA(Data = , var_list = , bounds = , expr = , restrict = , gapressure = ,
  out_table = , seed = , gen_size = );
```

`Data` specifies the dataset that has the data that will be used to estimate the model. `var_list` should be a SAS dataset having a character variable named `var` with the list of the indepen-

dent variables of the model. `expr` should be the mathematical formula for the model (the dependent variable is obtained by parsing the left-hand side of this equation). `bounds` should be a SAS dataset having a variable `lbound` and `ubound` respectively specifying the lower and upper bound for each parameter (a parameter $sd2$ associated to the variance should always be present), `gapressure` (optional) should be between 1 and 5, where 5 is the strongest evolutionary pressure and 1 the minimum. `Restrictions` is an optional parameter that should be a SAS dataset having a character variable `restrict` with the restrictions that should be enforced. `Out_table` is the name for the output dataset containing the estimated values; `seed` (optional) is the desired seed for reproducibility; and finally `gen_size` (optional) is the population size; when more power is needed this should be increased (default is 400).

The macro works by first reading the parameters specified by the user, and checking that the required datasets exist. Secondly, the log-likelihood maximization is written into SAS `PROC GA` terminology. The equation for the model is parsed through a regular expression, and the restrictions are finally incorporated into the procedure.

The genetic pressure parameter deserves some particular attention. As it has been mentioned, more genetic pressure will produce generations which are the fittest, but they will probably be very much alike. The immediate consequence of this is that the genetic algorithm may suffer from premature convergence (see Section 6).

The most relevant part of the program is where the likelihood is computed for each observation, taking into account the restrictions. Note that in case restrictions are defined and satisfied, the computation of the log-likelihood is to be carried out in the regular way. On the contrary, if restrictions are present and not satisfied, then a large negative value is returned. In this fashion, the algorithm will tend to favor solutions belonging to the relevant subspace.

```
sum1 = 0                                ;
&rt.                                    ;
%if %length(&restrict.) = 0 %then %do   ;
        do i = 1 to &obs_data.          ;
                yhat = &modif_formula.  ;
                loglik = &TY.           ;
                sum1 = sum1 + loglik    ;
        end                             ;
%end                                    ;
%else %do;
        if &list_res. then do           ;
                do i = 1 to &obs_data.  ;
                yhat = &modif_formula.  ;
                loglik = &TY.           ;
                sum1 = sum1 + loglik    ;
                end                     ;
        end                             ;
        else do                         ;
                sum1 = -10000000000     ;
        end                             ;
%end                                    ;
return(sum1)                            ;
```

where `yhat` is the predicted value using the parameters from the current iteration and `loglik` is the log-likelihood for observation i.

4. Examples

4.1. Michaelis-Menten kinetics

This is one of the fundamental equations describing enzime kinetics. It states a relationship between the reaction rate of enzymatic reactions to the concentration of a substrate. Interesting discussions on the statistical aspect of this equation can be found in Raaijmakers (1987); Ritchie and Prvan (1996). The model can be formulated as

$$v = \frac{(V_{\max} \times S)}{K_m + S} + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

SAS Institute Inc. (2008) `PROC NLIN` documentation discusses possible strategies to obtain initial values for this problem. Essentially, since the function is increasing in S, and as S increases the limit is $V_{\max}$, the largest observed value should be taken as $V_{\max}$. Regarding K_m doing some replacements and using a grid it is possible to obtain a reasonable range of values for it.

In Example 1 (in the replication files) it can be seen that when, for example $n = 30$ and the true values are: $V_{\max} = 158$, $K_m = 0.07$ and $\sigma^2 = 1$ and the starting values are $V_{\max} = 40$, $K_m = 40$ and $\sigma^2 = 1.5$, `PROC NLMIXED` reports problems with final estimates very distant from the true ones ($V_{\max} = 172590$, $K_m = -121743$, $\sigma^2 = 32387$) (also problems with the Hessian appear).

Before calling the `MLGA` macro, two simple datasets need to be created. The first one should hold a list of coefficients with a lower and upper bound. The second dataset should hold a list of the independent variables. Finally, the macro should be called.

```
data coefi ; length coefficient $50.                          ;
coefficient = "Vmax" ; lbound = 0.1 ; ubound = 1000 ; output  ;
coefficient = "Km"   ; lbound = 0.1 ; ubound = 500  ; output  ;
coefficient = "sd2"  ; lbound = 0.2 ; ubound = 10   ; output  ;
run                                                           ;

data list_var     ;
length var $50.   ;
var = "S"; output ;
run               ;

%MLGA( data       = SIMU_DATA_MM ,
       var_list   = list_var ,
       bounds     = coefi ,
       expr       = "y = ( Vmax * S) / ( Km + S )" ,
       out_table  = XDG ,
       seed       = 11
);
```

Parameter	True value	MLGA estimated	lbound	ubound
$V_{\max}$	158	158.427	0.1	1000
K_m	0.07	0.100	0.1	500
σ^2	1	0.963	0.2	10

Table 1: Michaelis-Menten model.

Furthermore, the MLGA macro solves this problem easily as can be verified in Table 1. Note that MLGA is searching coefficients in a broad space, compared to PROC NLMIXED which tends to fail when some of those initial values are = 40.

4.2. Four parameter Emax model

This model is widely used to model dose-relationships of certain drugs (for example Dette, Melas, and Wong 2005; Accorsi-Mendonça, Corrêa, Paiva, Souza, Laurindo, and De Oliveira 2012; Sunzel, Paalzow, Berggren, and Eriksson 1988). The typical formulation of the model is as follows:

$$v = E0 + \frac{Emax \times (Dose^{\beta})}{(ED50^{\beta}) + (Dose^{\beta})} + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

Similar to the previous example, two datasets need to be created.

```
data coefi ; length coefficient $40.;
coefficient = "beta" ; lbound = 0.1 ; ubound = 50   ; output;
coefficient = "E0"   ; lbound = 0.1 ; ubound = 1000 ; output;
coefficient = "ED50" ; lbound = 0.1 ; ubound = 1000 ; output;
coefficient = "Emax" ; lbound = 0.1 ; ubound = 1000 ; output;
coefficient = "sd2"  ; lbound = 0.2 ; ubound = 10   ; output;
run;

data list_var         ;
length var $20.       ;
var = "Dose" ; output ;
run                   ;

%MLGA( data       = SIMU_DATA_EM ,
       var_list   = list_var ,
       bounds     = coefi ,
       expr       = "predx = E0 + (( Emax * ( Dose**beta ))/(( ED50**beta )
       + ( Dose**beta )))" ,
       out_table  = XDG ,
       seed       = 11
);
```

PROC NLMIXED shows problems when the true values are $E0 = 183$, $\beta = 6.22$, $ED50 = 5.06$, $Emax = 37$, $s2 = 1$ and the starting values are for example $E0 = 250$, $\beta = 0.1$, $ED50 = 2.06$, $Emax = 130$, $s2 = 1$, reporting final estimates very distant from the true ones, and

Parameter	True value	MLGA estimated	lbound	ubound
$E0$	183	185.409	0.1	1000
β	6.22	5.884	0.1	50
$Emax$	37	34.673	0.1	1000
$ED50$	5.06	5.161	0.1	1000
σ^2	1	1.429	0.2	10

Table 2: Emax model.

with problems in the Hessian. Other sets of initial values such as: $E0 = 183$, $\beta = 6.22$, $ED50 = 5.06$, $Emax = 130$, $s2 = 1$, have problems needing more than 200 iterations and reporting a warning, with estimates very different from the true ones. On the contrary, the MLGA macro solves this problem easily as can be verified in Table 2. These results can be checked in Example 2 in the replication files.

5. Restrictions

Restrictions are frequently necessary in nonlinear models, mostly for two different reasons: in order to incorporate prior knowledge into the model (the magnitude or sign of a certain coefficient may already be known), or to ensure proper identification of it. A clear example of this last case is the bi-exponential model. It is used extensively to model drug concentrations as functions of time, for example in Sekimoto, Fukui, and Fujita (1997); Zhang, Sigmund, Rusinek, Chandarana, Storey, Chen, and Lee (2012); Williams, Buhl, and Mody (2012). This model is not identified, since the pairs $(A, \alpha1)$ and $(B, \alpha2)$ are exchangeable. Naturally, all algorithms are expected to fluctuate between these two pairs of coefficients.

$$z = \alpha1 \times (\exp(-A \times x)) + \alpha2 \times (\exp(-B \times x)) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

Consequently, a restriction is usually added to ensure proper identification of the model:

$$\alpha1 > \alpha2 \tag{1}$$

Even if the identification issues are resolved, initial values may be difficult to obtain. Bonate (2011, p. 114) and Pinheiro and Bates (2000, p. 514) show how using some algebra and an auxiliary linear regression in logs, reasonable initial values can be obtained. With the genetic algorithm, these calculations are no longer required.

```
data coefi ; length coefficient $40.                                ;
coefficient = "alpha1" ; lbound = 0.0 ; ubound = 100  ; output ;
coefficient = "alpha2" ; lbound = 0.0 ; ubound = 100  ; output ;
coefficient = "A"      ; lbound = 0.1 ; ubound = 1000 ; output ;
coefficient = "B"      ; lbound = 0.1 ; ubound = 1000 ; output ;
coefficient = "sd2"    ; lbound = 0.2 ; ubound = 10   ; output ;
run                                                                 ;

data list_var         ;
length var $20.       ;
```

Parameter	True value	Starting value	PROC NLMIXED estimated	MLGA restricted
A	70	3	49.817	70.513
B	50	3	70.032	50.991
$\alpha 1$	0.5	0.01	0.049	0.520
$\alpha 2$	0.05	0.01	0.495	0.051
$\sigma 2$	1	1	0.920	0.906

Table 3: Bi-exponential model with inverted coefficients for PROC NLMIXED.

```
var = "Dose" ; output ;
run                   ;

data restrictions                      ;
    restriction = "alpha1>alpha2"      ;
    output                             ;
run                                    ;

%MLGA( data       = data_biexp,
       var_list   = list_var ,
       bounds     = coefi ,
       expr       = "predx = A * exp(-alpha1 * Dose) +
       B * exp(-alpha2 * Dose)",
       out_table  = XDG ,
       restrict   = restrictions ,
       gen_size   = 800 ,
       seed       = 11
);
```

The macro is called the standard way, with the only exception that an additional restriction table needs to be passed. This example can be found in Example 3 in the replication file.

6. Genetic pressure

In general, a genetic algorithm that applies mutation operations or crosses solutions infrequently in order to build offspring will be very selective. This means that the solutions carried across iterations will have on average a very good score on the fitness function. But the problem is that they will be very much alike, with the dire consequence of having a major risk of suffering premature convergence. On the other hand, a very diverse population of parameters will explore the solution space much better, with the consequence of having a very low chance of achieving the optimum. It is thus evident that a balance needs to be achieved. A proper strategy could be to force each solution to compete in a tournament of size x and then retain the parameter set that has won that tournament. The members that compete in each tournament are selected at random.

In this macro, there is an optional parameter called genetic_pressure that should be an integer between 1 and 5 (1 being the lowest genetic pressure and 5 the highest). This parameter alters certain aspects of the genetic algorithm. For instance, it affects the delta mutation

genetic_pressure	Log-likelihood	$E0$	β	$Emax$	$ED50$	σ^2
1	−1526.331	185.409	5.884	34.673	5.161	1.429
2	−5719.118	217.437	50.000	0.236	1000.000	10.000

Table 4: Emax model with different genetic pressure.

vector (the values that are summed to each perturbed solution) – a higher genetic pressure will add a larger value equal to 0.2/`genetic_pressure` to each element. On the contrary, when this parameter is close to 5, practically nothing will be added to perturbed solutions. In addition, this parameter affects the tournament size. When the tournament size is big, there is a high probability that a good solution is present and will win the tournament, discarding the other solutions. On the other hand, in small tournament sizes it is likely that only bad solutions are present, consequently increasing the probability that a bad solution moves forward to the next generation. Finally the probability that two solutions are paired and offspring is built is modified according to this parameter as 0.65/`genetic_pressure` meaning that when the *genetic pressure* is high, solutions will most likely be passed as they are to the next generation. As an example of how relevant this impact can be, returning to Section 4.2, and running it with `genetic_pressure = 1` and `genetic_pressure = 2` yields the following results in Table 4 (see Example 4 in the replication files).

Clearly, the poor mixing induced by the large value of this macro parameter produces incorrect results in this case. The optimal value for this parameter is an unknown *a priori*, and depends on the model and the data. Naturally, it is fairly straightforward to call this macro looping through a couple of values in order to determine the optimal one.

7. Conclusion

Numerical problems may appear when estimating nonlinear regression models, thus requiring a trial and error approach until correct convergence is detected. In SAS, it is customary to use the `TO-BY` option in the `PARMS` statement, but it is generally impractical for large grids. Moreover, there is no guarantee that the log-likelihood will increase as the number of grid points augments. The `MLGA` macro is presented; it allows to search for values that maximize the log-likelihood under Gaussian residuals in a reasonable large space without needing the usual assumptions required by most maximization methods. In addition, it can also handle very complicated multi-parameter restrictions. The resulting values produced by it may be used later as starting values for derivative-based numerical maximization algorithms.

References

Accorsi-Mendonça D, Corrêa FMA, Paiva TB, Souza HPD, Laurindo FRM, De Oliveira AM (2012). "The Balloon Catheter Induces an Increase in Contralateral Carotid Artery Reactivity to Angiotensin II and Phenylephrine." *British Journal of Pharmacology*, **142**(1), 79–88.

Bonate P (2011). *Pharmacokinetic-Pharmacodynamic Modeling and Simulation*. 2nd edition. Springer-Verlag.

Dette H, Melas VB, Wong WK (2005). "Optimal Design for Goodness-of-Fit of the Michaelis-Menten Enzyme Kinetic Function." *Journal of the American Statistical Association*, **100**(472), 1370–1381.

Dorsey RE, Mayer WJ (1995). "Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features." *Journal of Business & Economic Statistics*, **13**(1), 53–66.

Forrest S (1996). "Genetic Algorithms." *ACM Computing Surveys*, **28**(1), 77–80.

Gulsen M, Smith AE, Tate DM (1995). "A Genetic Algorithm Approach to Curve Fitting." *International Journal of Production Research*, **33**(7), 1911–1923.

Kapanoglu M, Koc IO, Erdogmus S (2007). "Genetic Algorithms in Parameter Estimation for Nonlinear Regression Models: An Experimental Approach." *Journal of Statistical Computation and Simulation*, **77**(10), 851–867.

Olinsky AD, Quinn JT, Mangiameli PM, Chen SK (2004). "A Genetic Algorithm Approach to Nonlinear Least Squares Estimation." *International Journal of Mathematical Education in Science and Technology*, **35**(2), 207–217.

Pan Z, Chen Y, Kang L, Zhang Y (1995). "Parameter Estimation by Genetic Algorithms for Nonlinear Regression." In GZ Liu (ed.), *Optimization Techniques and Applications, Proceedings of International Conference on Optimization Technique and Applications '95*, pp. 946–953.

Pinheiro J, Bates D (2000). *Mixed-Effects Models in S and S-PLUS*. Springer-Verlag.

Raaijmakers J (1987). "Statistical Analysis of the Michaelis-Menten Equation." *Biometrics*, **43**(4), 793–803.

Ritchie RJ, Prvan T (1996). "Current Statistical Methods for Estimating the K_m and $V_{\max}$ of Michaelis-Menten Kinetics." *Biochemical Education*, **24**(4), 196–206.

SAS Institute Inc (2008). *SAS/STAT Software, Version 9.2*. Cary, NC. URL http://www.sas.com/.

Sekimoto M, Fukui M, Fujita K (1997). "Plasma Volume Estimation Using Indocyanine Green with Biexponential Regression Analysis of the Decay Curves." *Anaesthesia*, **52**(12), 1166–1172.

Sunzel M, Paalzow L, Berggren L, Eriksson I (1988). "Respiratory and Cardiovascular Effects in Relation to Plasma Levels of Midazolam and Diazepam." *British Journal of Clinical Pharmacology*, **25**(5), 561–569.

Whitley D (1994). "A Genetic Algorithm Tutorial." *Statistics and Computing*, **4**(2), 65–85.

Williams SR, Buhl EH, Mody I (2012). "The Dynamics of Synchronized Neurotransmitter Release Determined from Compound Spontaneous IPSCs in Rat Dentate Granule Neurones in Vitro." *The Journal of Physiology*, **510**(2), 477–497.

Wright AH (1991). "Genetic Algorithms for Real Parameter Optimization." In *Foundations of Genetic Algorithms*, pp. 205–218. Morgan Kaufmann.

Zhang JL, Sigmund EE, Rusinek H, Chandarana H, Storey P, Chen Q, Lee VS (2012). "Optimization of B-Value Sampling for Diffusion-Weighted Imaging of the Kidney." *Magnetic Resonance in Medicine*, **67**(1), 89–97.

Affiliation:

Francisco Juretig
The Nielsen Company
Global Research Analysis and Development
Nielsen House, London Road Headington
Oxford OX3 9RX, United Kingdom
E-mail: Francisco.Juretig@Nielsen.com, fjuretig@yahoo.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 66, Code Snippet 2 *Submitted:* 2012-10-15
August 2015 *Accepted:* 2014-10-20