NLP for MLE: Non-Linear Programming for Maximum Likelihood Estimation
Robert J. Gallop, West Chester University, West Chester, PA

ABSTRACT
The NLP procedure offers a series of optimization techniques for minimizing or maximizing continuous nonlinear functions. In this paper, we consider using the Newton-Raphson technique to maximize likelihood functions based on assumed probabilistic distribution on observed data. This results in estimates of the unknown parameter referred to as maximum likelihood estimates (MLE). This paper illustrates to examples of using NLP to derive MLE’s.

Keywords: PROC NLP, likelihood function, Newton-Raphson, maximum likelihood estimation.

INTRODUCTION
While SAS® software has many procedures which provide modeling platforms for common statistical models, if a researcher wants to model data which follows a likelihood distribution which is not accommodated by the current procedures, then a key question is what can be done? Many researchers will consider alternate software, which they may believe is better suited for their needs. The NLP procedure offers many optimization techniques for maximizing nonlinear functions. This paper considers application of the NLP procedure for maximum likelihood estimation under two examples. The first example considers realizations from a Bernoulli distribution with the goal to estimate the probability of success. The second example considers estimation of data acquired from a clustered binary Markov process.

MAXIMUM LIKELIHOOD ESTIMATION
The method of maximum likelihood is, by far, the most popular technique for deriving estimators. Please refer to Wackerely et al. (2002) or any comparable mathematical statistics text for a similar discussion as illustrated below. In principle, the method is very simple and intuitive. Before outlining the steps for the derivation, we first define the likelihood function. Suppose that $Y_1, Y_2, ..., Y_n$ iid $f_Y (y; \theta)$; $\theta \in \Omega$, and that $y_1, y_2, ..., y_n$ are the $n$ observed data values. The likelihood function for $\theta$ is given by

$$L(\theta | \mathbf{y}) = L(\theta | y_1, y_2, ..., y_n) = \prod_{i=1}^{n} f_Y (y_i; \theta)$$

Hence, the likelihood function is a joint distribution, which represents the probability the data fits the assumed model. As the value of the unknown parameters change, the value of the likelihood function changes; therefore, to maximize the joint probability the data fits the assumed model, we need the value of the unknown parameters that achieve that maximum probability.

Derivation of this maximum may result in some calculations which can get rather involved. The maximum likelihood estimator (MLE) for $\theta$, denoted $\hat{\theta}_{MLE}$, is simply the value of $\theta$ that maximizes the likelihood function $L(\theta | \mathbf{y})$. Its interpretation is very appealing: “The value $\hat{\theta}_{MLE}$ is more likely to have produced the data $y_1, y_2, ..., y_n$ than any other value of $\theta$.

Since the likelihood is a product, the mathematics can get quite cumbersome. We use a mathematical transformation, which may help the derivation. The mathematical transformation is a logarithmic transformation, which results in the log-likelihood. We recall for all $x>0$, $f(x) = \ln(x)$ is an increasing function. This follows immediately since the first derivative, $f'(x)$, is equal to $1/x$ which is greater than 0 for all positive $x$. In maximizing a likelihood function $L(\theta | \mathbf{y})$, we will often be able to use differentiable calculus (i.e., find first derivative, set equal to zero, solve for $\theta$, verify solution is a maximized by verifying second order conditions). However, it will often be the case that working with $\ln L(\theta | \mathbf{y})$ is easier analytically. The main advantage of the log-likelihood function is based on
the property \( \ln(ab) = \ln a + \ln b \), such that a product becomes a sum. So our likelihood equation under the log-likelihood becomes:

\[
\ln L(\theta \mid y) = \sum_{i=1}^{n} \ln f(y_i, \theta)
\]

Therefore, applying differential calculus, the derivative of a sum is the sum of the derivatives, which is not only easier for an individual, but is exactly how PROC NLP estimates MLE based on raw data.

**PROC NLP**

As discussed in Katz (2006), SAS has PROC NLP, a nonlinear operator, in the SAS/OR \({\text{\textsuperscript{\textregistered}}}\) software module. It was introduced as an experimental release with SAS 6.08 and was placed in production with SAS 6.09, and has been included with each subsequent release of SAS/OR. The NLP offers a set of optimization techniques for minimizing or maximizing a continuous nonlinear function. With our goal of implementing PROC NLP for MLE, the function will be the log-likelihood function and the optimization will be maximization for MLE.

Here is the PROC NLP template syntax we will use for MLE:

```
PROC NLP data=x covariance=h pall tech=newrap;
Max logf;
Parms theta1=0, theta2=0, ..., thetan=0;
Array a[10] a1-a10;
Logf = ****;
Quit;
```

**PROC NLP Syntax Notes:**

- COVARIANCE=h specifies the derivation of the covariance matrix which is using the Hessian matrix. Under reasonable assumptions, the approximate standard errors of the estimates are the square-root of the diagonal elements of the covariance matrix of the parameter estimates which is the same as the inverse of the Hessian matrix per the SAS documentation.
- PALL request that the covariance matrix, standard errors, the model program and variable lists, the derivative tables and the compiled program code to be displayed. To print information with regards to only the Hessian matrix use PHES instead of PALL. To print information with regards to only the Covariance matrix use PCOV instead of PALL.
- TECH=NEWRAP performs the Newton-Raphson optimization technique. The algorithm combines a line-search algorithm with ridging.
- MAX logf specifies the log-likelihood function to be maximized.
- PARAMS theta1=0, theta2=0, ..., thetan=0 – provides the list of parameters to estimate as well as initial "guesses" to the parameters to be estimated.
- ARRAY a[10] a1-a10 is similar to, but not the same as, the ARRAY statement in the SAS DATA step. The ARRAY statement is used to associate a name (of no more than eight characters) with a list of variables and constants.
- LogF is the form for the Log-Likelihood function.
- Quit ends the program.

In unconstrained maximum likelihood estimation, we must have the gradient at the solution very close to zero and the Hessian matrix at the solution must have negative eigenvalues. An eigenvalue of zero may indicate an inflection point; therefore, close inspection of the likelihood evaluated at the parameter estimates must be made to ensure a maximum is truly achieved.

**BINOMIAL DISTRIBUTION EXAMPLE**

Suppose that \( Y_1; Y_2; \ldots; Y_n \) are iid Bernoulli observations with mean \( p \), where \( 0 < p < 1 \). Our goal is to derive the MLE of \( p \). The likelihood equation for \( p \), with \( y_i = 0, 1 \) is given by:
The logarithm of \( L(p | y) \) is given by

\[
\ln L(p | y) = \sum_{i=1}^{n} y_i \ln p + \left( n - \sum_{i=1}^{n} y_i \right) \ln(1 - p)
\]

which is equal to:

\[
\ln L(p | y) = \sum_{i=1}^{n} y_i \ln(p) + \sum_{i=1}^{n} (1 - y_i) \times \ln(1 - p)
\]

and the derivative of the log-likelihood is thus

\[
\frac{\partial}{\partial p} \ln L(p | y) = \frac{\sum_{i=1}^{n} y_i}{p} - \frac{\left( n - \sum_{i=1}^{n} y_i \right)}{1 - p}
\]

Setting the derivative to 0 and solving for \( p \), one obtains \( \hat{p}_{MLE} = \bar{y} \), the sample proportion. It can be shown that the second order conditions are satisfied as well.

Consider a survey of registered voters consisted of 100 ballots of which 64 were in favor of the proposal. We can consider this a binomial setting, where each individual is a Bernoulli trial. Using the derivation above, we see \( \hat{p}_{MLE} = \bar{Y} = 0.64 \). Our NLP Syntax is:

```
PROC NLP data=x covariance=h tech=newrap phes;
Max logf;
Parms p=0.5;
Logf =y*log(p) + (1-y)*log(1-p);
Quit;
```

The data set \( x \) consist of 100 records, with 64 of the 100 with \( y=1 \) and 36 of the 100 with \( y = 0 \). The logf function, which corresponds to the log-likelihood function, is the function we want to maximize. TheParms line has an initial value of \( p \), the unknown parameter for which we want to achieve the maximum likelihood estimate. For simplistic log-likelihood PROC NLP can achieve MLE’s with even poor initial guesses of the unknown parameters, but as the log-likelihood function becomes more complex, the programmer may need to use search grids to achieve good initial guesses. The Logf function shows the individual observations contribution to the log-likelihood function. PROC NLP adds each observations contribution to logf and maximizes the sum.

Figure 1 through Figure 3 shows the output from the above PROC NLP code.
The ABSCONV indicates that convergence has been achieved. You are looking to achieve ABSCONV or GCONV to be achieved.
Figure 3 provides the MLE estimate for $p$ which as we derived by hand is 0.64. We see the gradient function is near 0 and the Hessian matrix has only a negative eigenvalue which verifies our estimate is truly the maximum likelihood estimate.

**MIXED EFFECTS MARKOV MODEL EXAMPLE**

The motivation for examining PROC NLP’s ability to perform maximum likelihood estimation as from data from a cocaine addiction trial, where it was of interest to quantify transition rates between abstinence and usage as illustrated in Figure 4.
Figure 4. Two-State Markov Application

\[
\begin{align*}
\lambda_1 & \quad \text{Usage (State 1)} \\
\lambda_2 & \quad \text{Abstinence (State 2)}
\end{align*}
\]

$\lambda_1$ is the rate of transition from USAGE to ABSTINENCE per unit time.  
$\lambda_2$ is the rate of transition from ABSTINENCE to USAGE per unit time.

To investigate the transition between states such as abstinence and usage, advanced statistical methods are needed. Markov models provide one framework for analysis of situations in which the status of individuals may be characterized by a finite number of states and in which individuals may transition among the states over time. For binary outcome data, the data can be thought of as a system with two states. Along these lines, Barnett (1999) fit a two-state Markov model for the analysis of cost-effectiveness of methadone maintenance. Within longitudinal studies involving repeated assessments, substantial subject-to-subject heterogeneity may exist in the transition from state to state. Conaway (1990) considered an extension of the two-state Markov model structure that models the subject-to-subject heterogeneity by the incorporation of random effects. Cook (1999) developed a generalization of Conaway’s model to accommodate irregular spacing of the observation times and the inclusion of covariates. This model is called a Mixed Effect Markov model (MMM). Distributions for the two random effects, $\mu_1$ and $\mu_2$, in the MMM are as independent gamma variables each with mean one and separate variances $\tau_1$ and $\tau_2$, respectively. Pulkstenis et al. (1998) and Conaway (1990) used the same random effects distribution assumption in related models for the analysis of clustered binary data. The random effects were included as multiplicative factors to algebraic transformations of the transition rates. The first transformation is denoted as $\theta_1$, with 
\[
\theta_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2} \exp(-\mu_1 e^{\beta})
\]
where $\lambda_1$ and $\lambda_2$ are the transition rate from usage to abstinence and the transition rate from abstinence to usage, respectively. The second transformation is denoted as $\theta_2$, with 
\[
\theta_2 = (\lambda_1 + \lambda_2) \mu_2 \exp(\beta_2)
\]
We assume the time between transitions per subject is s units. The number of transitions and transition type are represented by $r_{11}$, $r_{12}$, $r_{21}$, and $r_{22}$, where $r_{11}$ represents the number of transitions from state 1 to state 1 for the $i^{th}$ subject. Similar definitions correspond to the other three transition types. The four-dimensional subspace $D$ is defined as follows: 
\[
D = \{ v : 0 \leq v_1 \leq r_{22} + v_4, 0 \leq v_2 \leq r_{11} + r_{21} + v_2 + v_4, \\
0 \leq v_3 \leq r_{12}, 0 \leq v_4 \leq r_{22} \}.
\]
The likelihood per subject will be summed over this 4-dimensional subspace per gap time. Additional notation is
Cook's (1999) derivation of the likelihood for the MMM assuming s units of time between transitions is as follows:

\[
\sum_{v \in D} \exp \left( \log \left( a(v) \right) \times \exp \left( -\mu_i b(v) e^{b_i} \right) \times \exp \left( -\mu_2 c(v) e^{b_i} s \right) \right)
\]

Integrating the random effects yields the likelihood function of:

\[
\sum_{v \in D} \exp \left\{ \log(a(v)) - \log \left( 1 + (b(v) \tau_1 e^{b_i}) \right) / \tau_1 \right\} - \log \left( 1 + c(v) \tau_2 e^{b_i} s \right) / \tau_2 \right\}
\]

Now to implement the MLE of the above likelihood derivation for the Cocaine abstinence example, requires the following steps: (1) Load the Data, (2) Derive the transition types, (3) Derive the four dimensional space, D and the values for the additional notation a(v), b(v), and c(v), and (4) Formulate the likelihood within PROC NLP.

Step 1 – Load the data:

data dataA;
input patid y time prev;
cards;
1 2 0.
1 2 4 2
1 1 8 2
1 2 1 2 1
2 1 0.
2 1 3 1
2 2 6 1
3 1 0.
3 2 2 1
3 1 4 2
3 1 6 1
4 1 0.
4 1 1
;
run;

data dataA;
set dataA;
lpat = lag(patid);
ltime = lag(time);
stime = time - ltime;

if patid = lpat then logtime = log(stime);
if prev ne . ;
run;

As indicated the data consists of multiple records for each patid. The second data steps produces a final data set which consists for each patient their current state, previous state, and the amount of time taken to transition between states.

Step 2 – Derive the transition types:

data dataA;
set dataA;
r11=0; r12=0; r21=0; r22=0;
if y = 1 and prev = 1 then r11 = 1;
if y= 1 and prev = 2 then r21 = 1;
if y = 2 and prev = 1 then r12 = 1;
if y = 2 and prev = 2 then r22 = 1;
run;

proc univariate data=dataA noprint;
by patid stime
var r11 r12 r21 r22;
output out=dataB sum=r11 r12 r21 r22;
run;

The above steps produce a new data set DataB which consists of 1 record per patient with the various transition types and number of transition done per type.

Step 3 – Derive D:

data dataC;
set dataB;
do i4 = 0 to r22;
do i3= 0 to r11;
do i2 = 0 to (r12+r21+i3+i4);
do i1 = 0 to (r21+i4);

*****Create A****;
    r = mod(sum(i1+i2+i3+i4),2);
    if r= 0 then r1=1;
    if r = 1 then r1 = -1;

    a1 =comb(r21+i4,i1)*comb(r12+r21+i3+i4,i2)*comb(r11,i3)*
        comb(r22,i4)*(r1);
    b1 = r12+i1+i3;
    c1 = i2;

    output;
    end; /* i1 */
    end; /* i2 */
    end; /* i3 */
    end; /* i4 */
run;

data dataD;
set dataB;
keep patid stime; /* One Record by Patient */
run;

%let indata = dataC ; /* name of data set;*/  
%let vars = a1 b1 c1 ; /* variables */  
%let ivars = a b c ; /* Index for Transpose; */  
%macro tform;  
*******************;  
%do k=1 %to 3;  
***********************;  
%put &k.;  
%let mvar&k=%scan(&vars,&k,%str( ));  
%letivar&k=%scan(&ivars,&k,%str( ));  
%put Variable Number &k. is &&mvar&k  
***********************;  
proc transpose data=&indata prefix = &&ivar&k name = check out = Inf&&mvar&k.;  
by patid stime;  
var &&mvar&k.;  
quit;  
data inf&&mvar&k.;  
set inf&&mvar&k.;  
drop check;  
run;  
proc contents noprint out=cntnts; quit;  
data cntnts; set cntnts; obsnr = _n_; run;  
data _null_;  
set cntnts;  
call symput("VAR"||left(put(varnum,6.)),name);  
call symput("N_VAR", obsnr);  
run;  

%let varlist= ;  
%put Varlist is set to Blank  &varlist;  
%do ki = 1 %to &n_var;  
%put Variable being added on is &&var&ki.;  
%let varlist=&varlist &&var&ki;  
%put Iterative process of creating varlist: &varlist;  
%end;  
data Inf&&mvar&k.;  
set Inf&&mvar&k.;  
array info(*) &varlist;  
do i = 1 to dim(info);  
if info(i) = . then info(i) = 0;  
end;
drop i;
run;

proc sort data=Inf&&mvar&k.;
by patid;
quit;
data dataD;
merge dataD Inf&&mvar&k.;
by patid;               /* Record of ID - Patid */
run;
%end;
%mend tform;
%tform;

Step 4: Run in PROC NLP

We need to maintain one record per patient. Running the above code, produces each portion of the summation.

proc nlp data=dataD cov=2 pall tech=newrap;
max logf;
parms beta1=-1.08, beta2=0.53, gamma1=0, gamma2=0;  /* id parms and give initial ests */
/* define obj function ... */

s = log(
a1*exp(-1*(log(1+b1*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c1*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a2*exp(-1*(log(1+b2*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c2*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a3*exp(-1*(log(1+b3*exp(beta1+gamma1)))/exp(gamma1)+
           -1*(log(1+c3*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a4*exp(-1*(log(1+b4*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c4*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a5*exp(-1*(log(1+b5*exp(beta1+gamma1)))/exp(gamma1)+
           -1*(log(1+c5*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a6*exp(-1*(log(1+b6*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c6*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a7*exp(-1*(log(1+b7*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c7*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a8*exp(-1*(log(1+b8*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c8*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a9*exp(-1*(log(1+b9*exp(beta1+gamma1)))/exp(gamma1) +
           -1*(log(1+c9*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a10*exp(-1*(log(1+b10*exp(beta1+gamma1)))/exp(gamma1)+
             -1*(log(1+c10*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a11*exp(-1*(log(1+b11*exp(beta1+gamma1)))/exp(gamma1) +
             -1*(log(1+c11*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a12*exp(-1*(log(1+b12*exp(beta1+gamma1)))/exp(gamma1) +
             -1*(log(1+c12*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a13*exp(-1*(log(1+b13*exp(beta1+gamma1)))/exp(gamma1) +
             -1*(log(1+c13*stime*exp(beta2+gamma2)))/exp(gamma2)) +

Statistics and Data Analysis
NESUG 2007
a14*exp(-1*(log(1+b14*exp(beta1+gamma1)))/exp(gamma1) +
   -1*(log(1+c14*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a15*exp(-1*(log(1+b15*exp(beta1+gamma1)))/exp(gamma1) +
   -1*(log(1+c15*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a16*exp(-1*(log(1+b16*exp(beta1+gamma1)))/exp(gamma1) +
   -1*(log(1+c16*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a17*exp(-1*(log(1+b17*exp(beta1+gamma1)))/exp(gamma1) +
   -1*(log(1+c17*stime*exp(beta2+gamma2)))/exp(gamma2)) +
a18*exp(-1*(log(1+b18*exp(beta1+gamma1)))/exp(gamma1) +
   -1*(log(1+c18*stime*exp(beta2+gamma2)))/exp(gamma2)));

logf = s;
run;

We see the above formulation can be shortened using the ARRAY statement in PROC NLP as follows:

proc nlp data=dataD cov=2 pall tech=newrap;
   max logf;
   parms beta1=-1.08, beta2=0.53, gamma1=0, gamma2=0;  /* id parms and give initial ests */
   /* define obj function ... */
array a[18] a1-a18;
array b[18] b1-b18;
array c[18] c1-c18;
w = 0;
do i = 1 to 18;
   w = w + a[i]*exp(-1*(log(1+b[i]*exp(beta1+gamma1)))/exp(gamma1)
                  -1*(log(1+c[i]*stime*exp(beta2+gamma2)))/exp(gamma2));
end;
s = log(w)
logf = s
quit;

This syntax was applied to the Cocaine addiction trial with the results reported in Gallop et al. (2007).

CONCLUSIONS

This paper has illustrated that PROC NLP can provide MLE’s even under very complex likelihood functions. While it is necessary to have your data organized as one record for each individual observations contribution to the log-likelihood function, there is no need to use other software to achieve optimization in individual research. One must ensure the likelihood function and log-likelihood function is written accurately. This requires detailed observations for common errors such as missed minus or plus signs. As the likelihood or log-likelihood function becomes more complex, the programmer may need to use search grids to derive good initial guesses of the unknown parameters.

REFERENCES:


**ACKNOWLEDGMENTS**

SAS is a Registered Trademark of the SAS Institute, Inc. of Cary, North Carolina.

**CONTACT INFORMATION**

Your comments and questions are valued and encouraged. Contact the author at:

Robert J. Gallop  
West Chester University  
Department of Mathematics  
Applied Statistics Program  
323B Anderson Hall  
West Chester, PA 19383  
(610) 436-2419  
e-mail: rgallop@wcupa.edu

******************************************************************************