Maximum Likelihood in SAS

Statistics 135

Autumn 2005

Maximum Likelihood Estimation

Lets assume that that $x_1, x_2, \ldots, x_n$ are a random sample from a population with a distribution described by the density (or pmf if discrete) $f(x_i|\theta)$. The parameter θ might be a vector $\theta = (\theta_1, \theta_2, \ldots, \theta_p)$.

Then the likelihood function is

$$L(\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

The maximum likelihood estimate (MLE) of θ is

$$\hat{\theta} = \arg\sup L(\theta)$$

i.e. the value of θ that maximizes the likelihood function. One way of thinking of the MLE is that its the value of the parameter that is most consistent with the data.

One approach to maximizing the likelihood is via calculus by solving the equations

$$\frac{\partial L(\theta)}{\partial \theta_1} = 0, \quad \frac{\partial L(\theta)}{\partial \theta_2} = 0, \quad \ldots, \quad \frac{\partial L(\theta)}{\partial \theta_p} = 0$$

with respect to the parameter θ.

Note that when determining MLEs, it is usually easier to work with the log likelihood function

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(x_i|\theta)$$

It has the same optimum since $\log$ is an increasing function and it is easier to work with since derivatives of sums are usually much nicer than derivatives of products.

Thus we can solve the score equations

$$
\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta_1} &= \sum_{i=1}^{n} \frac{\partial \log f(x_i|\theta)}{\partial \theta_1} = 0 \\
\frac{\partial l(\theta)}{\partial \theta_2} &= \sum_{i=1}^{n} \frac{\partial \log f(x_i|\theta)}{\partial \theta_2} = 0 \\
&\cdots \\
\frac{\partial l(\theta)}{\partial \theta_p} &= \sum_{i=1}^{n} \frac{\partial \log f(x_i|\theta)}{\partial \theta_p} = 0
\end{aligned}
$$

for θ instead.

Key Properties of MLEs

1. For large n, the sampling distribution of an MLE is approximately normal.

2. For large n, MLEs are nearly unbiased with a variance smaller than any other estimator.

3. If $\hat{\theta}$ is the MLE of θ, then $g(\hat{\theta})$ is the MLE of $g(\theta)$, for any "nice" function $g(\cdot)$. (Transformations of MLEs are MLEs - Invariance property.

Poisson Example

Suppose we observe the number of alpha particle emissions of Carbon-14 that are counted by a Geiger counter per second. We wish to model the data of 20 counts using a Poisson distribution. What is the MLE of the parameter of the distribution? The data are

6	5	8	8	13	11	7	8	7	10
8	4	3	12	5	11	9	15	12	6

Note that $\sum x_i = 168$. The pmf for a Poisson random variable with mean λ is

$$f(x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, x = 0, 1, 2, \ldots$$

It can be shown that the MLE, $\hat{\lambda}$ satisfies

$$\hat{\lambda} = \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

So for the example $\hat{\lambda} = 8.4$

Truncated Poisson Example

Instead of modeling the data as a Poisson distribution, assume that in this experiment it is impossible to observe zero counts. So instead will be model the data with a truncated Poisson distribution with pmf

$$f(x|\lambda) = \frac{\lambda^x e^{-\lambda}}{(1 - e^{-\lambda})x!}, x = 1, 2, \ldots$$

Note that λ is not the mean of this distribution. It is in fact

$$E[X] = \frac{\lambda}{1 - e^{-\lambda}}$$

What is the MLE of λ?

It can be shown that $\hat{\lambda}$ does not have a closed form solution in this problem.

In fact, the MLE satisfies

$$r(\hat{\lambda}) = \hat{\lambda} - \bar{x}(1 - e^{-\hat{\lambda}}) = 0$$

where

$$r(\lambda) = \lambda - \bar{x}(1 - e^{-\lambda})$$

Note that

$$r(\lambda) = -\lambda(1 - e^{-\lambda})\frac{dl(\lambda)}{d\lambda}$$

Since $-\lambda(1 - e^{-\lambda}) \neq 0$ for $\lambda > 0$ (the range we are interested in) $r(\lambda)$ and $\frac{dl(\lambda)}{d\lambda}$ must have the roots so we can solve either.

Lets use **SAS** to find the solution to $r(\hat{\lambda}) = \hat{\lambda} - \bar{x}(1 - e^{-\hat{\lambda}}) = 0$

Grid Search in SAS

A brute for technique to find a value of λ which approximately solves this equation.

For different values of λ we will calculate $r(\lambda)$. Then pick the value of λ which gives $r(\lambda)$ closest to 0.

```
DATA grid_search;
  DO i = 1 to 1000;
    lambda = i/100;
    abs_r_lambda = ABS(lambda - (168/20) * (1 - EXP(-lambda)));
    OUTPUT;
  END;
  DROP i;

PROC PRINT DATA=truncated_poisson NOOBS;

RUN;
```

This code creates a data set with variables `lambda` (taking values from 0.01 to 10 by 0.01) and `abs_r_lambda`, the values of $|r(\lambda)|$ for the values of `lambda` in the dataset.

The `OUTPUT` in the `DO` loop tells **SAS** to write out the values created there. If this is omitted, no output will be stored from the `DO` loop.

```
Grid Evaluation                       16:49 Tuesday, November 22, 2005 406


           abs_r_
lambda     lambda


 0.01     0.07358
 0.02     0.14633
 0.03     0.21826
 0.04     0.28937
 0.05     0.35967

 and so on
```

Now lets sort the data from smallest to largest of $|r(\lambda)|$.

```
PROC SORT DATA = grid_search OUT = grid_sort_sorted;
  BY abs_r_lambda;

PROC PRINT DATA = grid_sort_sorted NOOBS;

RUN;
```

```
Sorted Grid Values                     16:49 Tuesday, November 22, 2005 426

           abs_r_
lambda     lambda

 8.40     0.00189
 8.39     0.00809
 8.41     0.01187
 8.38     0.01807
 8.42     0.02185
 8.37     0.02805
...
```

The first row of this resulting dataset contains the MLE as calculated by this grid search algorithm. Now lets create a final version of the data which just this row

```
DATA grid_mle;
  SET grid_search_sorted;
  IF (_N_ EQ 1);

* Print out the approximate MLE.;

PROC PRINT DATA = grid_mle NOOBS;

RUN;
```

```
MLE of Lambda by Grid Search           16:49 Tuesday, November 22, 2005 446

              abs_r_
lambda        lambda

  8.4     .001888886
```

Newton-Raphson in SAS

Another approach to finding the roots of a function $r(\lambda)$. In this method, we start with an initial value λ_0 and then the next guess is calculated by

$$\lambda_1 = \lambda_0 - \frac{r(\lambda_0)}{r'(\lambda_0)}$$

where $r'(\lambda)$ is the first derivative of $r(\lambda)$. We iterate this scheme until we are close to the root

$$\lambda_n = \lambda_0 - \frac{r(\lambda_0)}{r'(\lambda_0)}; \quad n = 2, 3, \ldots$$

In this truncated Poisson example

$$r(\lambda) = \lambda - \bar{x}(1 - e^{-\lambda})$$
$$r'(\lambda) = 1 - \bar{x}(1 - e^{-\lambda})$$

Lets start with a simple version of this algorithm that iterates the scheme 10 times.

```
DATA newton_simple;

  /* a good starting guess for lambda is the sample mean */
  lambda = (168/20);

  /* set the previous value of lambda to 0 */

  previous_lambda = 0;

  /* these variables store the current value of r(lambda)
     and the derivative r(lambda) */

  r_lambda = 0;
  r_primed_lambda = 0;
```

```
/* set the format of the variables - number are displayed
   up to 12 characters long, with 10 decimal places. */

FORMAT lambda 12.10 previous_lambda 12.10 r_lambda 12.10
       r_primed_lambda 12.10;

/* now update the Newton-Raphson step 10 times */

DO iteration = 1 to 10;
  r_lambda = (lambda - (168/20) * (1 - EXP(-lambda)));
  r_primed_lambda = (1 - (168/20) * EXP(-lambda));
  previous_lambda = lambda;
  lambda = lambda - r_lambda/r_primed_lambda;
  OUTPUT;
END;
```

```
/* Print out the resulting dataset */

PROC PRINT DATA=newton_simple NOOBS;
  ID iteration previous_lambda;
  TITLE 'MLE of Lambda by Newton-Raphson - Simple Approach';

RUN;
```

```
MLE of Lambda by Newton-Raphson - Simple Approach                              447
                                              16:49 Tuesday, November 22, 2005

               previous_                                                 r_primed_
iteration         lambda           lambda          r_lambda               lambda

     1       8.4000000000    8.3981075398    0.0018888855    0.9981111145
     2       8.3981075398    8.3981075364    0.0000000034    0.9981075365
     3       8.3981075364    8.3981075364    0.0000000000    0.9981075364
     4       8.3981075364    8.3981075364    0.0000000000    0.9981075364
     5       8.3981075364    8.3981075364    0.0000000000    0.9981075364
     6       8.3981075364    8.3981075364    0.0000000000    0.9981075364
     7       8.3981075364    8.3981075364    0.0000000000    0.9981075364
```

```
   8       8.3981075364    8.3981075364    0.0000000000    0.9981075364
   9       8.3981075364    8.3981075364    0.0000000000    0.9981075364
  10       8.3981075364    8.3981075364    0.0000000000    0.9981075364
```

One problem with this scheme is that it runs much longer that it needs to. The value of $\hat{\lambda}$ stabilizes after a couple of steps. Lets modify the scheme to allow it to stop early.

```
%LET meanx = (168/20);  /* A macro variable */
%LET maxiter = 10;
%LET converge = 1e-8;

DATA newton_better;

  /* Initialize Newton scheme */
  lambda = &meanx;
  previous_lambda = 0;
  r_lambda = 0;
  r_primed_lambda = 0;
  iteration = 1;
  /* Set formating */

  FORMAT lambda 12.10 previous_lambda 12.10 r_lambda 12.10
         r_primed_lambda 12.10;

  DO WHILE ((ABS(lambda-previous_lambda) > &converge) &
            (iteration < &maxiter));
    r_lambda = (lambda - &meanx * (1 - EXP(-lambda)));
    r_primed_lambda = (1 - &meanx * EXP(-lambda));
    previous_lambda = lambda;
    lambda = lambda - r_lambda/r_primed_lambda;
    OUTPUT;
    iteration = iteration + 1;
  END;

PROC PRINT DATA=newton_better NOOBS;
  ID iteration previous_lambda;
  TITLE 'MLE of Lambda by Newton-Raphson - Better Approach';
```

```
MLE of Lambda by Newton-Raphson - Better Approach                          448
                                           16:49 Tuesday, November 22, 2005

                previous_                                            r_primed_
iteration          lambda          lambda        r_lambda           lambda

    1        8.4000000000    8.3981075398    0.0018888855    0.9981111145
    2        8.3981075398    8.3981075364    0.0000000034    0.9981075365
```