POLS6382 Quantitative Methods III: Maximum Likelihood Estimation

Ling Zhu
Associate Professor
Department of Political Science
University of Houston
September 13, 2021

Lab 2: Estimating a Linear Regression Model Using MLE

Objectives

- Learn how to implement grid search in `R`.
- Compare OLS and ML estimators: normal regression model.
- Consider heterskedasticity in ML estimation.

1 Grid Search

The purpose of a grid search is to calculate the likelihood for a range of possible parameter values. The two primary considerations are the range from low to high parameter values and the mesh which determines how many values will be evaluated within this range. A coarse mesh will produce a fast but jagged plot of the likelihood, while a fine mesh will take longer to calculate but can produce a maximum likelihood estimate of high precision. Often a grid search is set up to be run several times with a variable mesh, starting with a coarse mesh to narrow the range in which the maximum of the likelihood falls, then an increasingly fine mesh to find the maximum with considerable precision.

While these examples are not likely to be used in practice (where other numerical methods are much quicker and more precise) there are some cases in which grid searches are still used, particularly with "difficult" likelihoods possessing several maxima or with rough likelihood surfaces. The purpose of doing a grid search for elementary problems is to get a feel for how one might find the maximum likelihood estimator to as high a precision as desired using numerical methods alone. Later we will apply analytic techniques and more sophisticated maximization methods.

1.1 A Bernoulli Example

Imagine that you observe three independent outcomes distributed as a Bernoulli process in which two outcomes are successes (the respondents voted in the last election) and the third is a failure (the respondent did not vote.) If $\pi$is the probability of success, and hence $1 - \pi$ is the probability of failure, and assuming the observations are independent, then our sample can be represented as $Y = 1, 1, 0$ and the likelihood of the sample is:

$$L(\pi|y) = \pi \times \pi \times (1 - \pi) \quad (1)$$

Since this is a Bernoulli distribution the parameter π represents the probability of a success and so must range between 0 and 1. We start the grid search by setting the range to (0, 1) and a coarse grid of .1. This search shows that the maximum of the likelihood lies between values of π between about .6 and .7. We narrow the range to these values and set the mesh to .01 and repeat the grid search over these values. We narrow the range to these values and set the mesh to .01 and repeat the grid search over these values. Finally, we find that the maximum appears to fall between .661 and .669. We search this range with a mesh of .001. The final estimate of the value of πthat maximizes the likelihood is found to be .667 using the final mesh of .001. If we plot the three searches together, we'll see that the first "trail" explores a large parameter space of π. The subsequent search just focus on the high-density territory, while the last search gets us even closer to the top of the "hill".

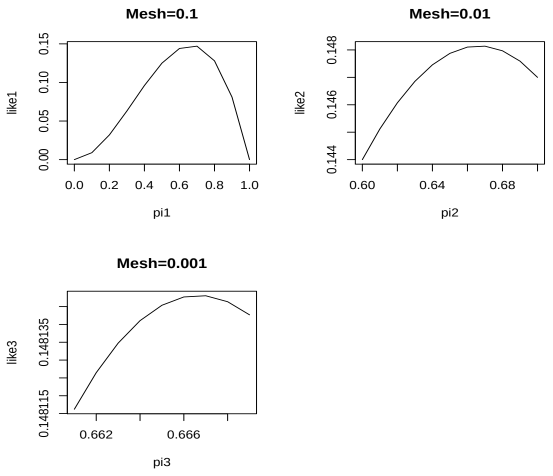
——————————————- R Code———————————————-

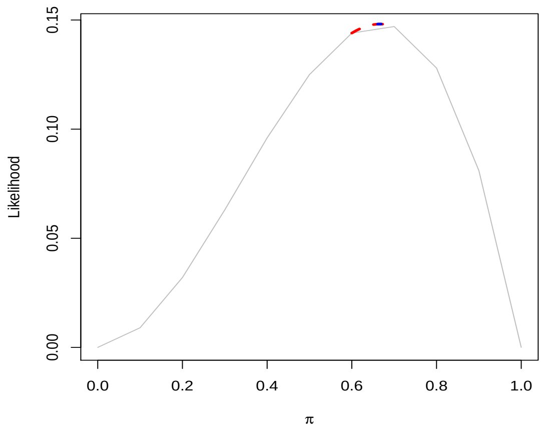
```
# Search 1
 pi1<-seq(0,1,.1)
 like1<-pi1*pi1*(1-pi1)
# Search 2
 pi2<-seq(.6,.7,.01)
 like2<-pi2*pi2*(1-pi2)
# Search 3
pi3<-seq(.661,.669,.001)
like3<-pi3*pi3*(1-pi3)

par(mfrow=c(2,2))
plot(like1~pi1,type="l")
plot(like2~pi2,type="l")
plot(like3~pi3,type="l")

# Plot three figures together
par(mfrow=c(1,1))
plot(like1~pi1,type="l",col="gray",xlab=expression(pi),ylab="Likelihood")
lines(pi2,like2, type="l",col=2,lty=2,lwd=3)
lines(pi3,like3, type="l",col="blue",lwd=3)
```

——————————————R Output———————————————-





1.2 Find the Maximum Likelihood Estimator using Grid Search

To list the full likelihood after the final search, and to find its maximum we use the functions:

——————————————- R Code————————————————-

```
print(cbind(pi3,like3))
print(cbind(pi3,like3)[like3==max(like3),])
```

——————————————R Output————————————————-

```
> print(cbind(pi3,like3))
        pi3     like3
 [1,] 0.661 0.1481162
 [2,] 0.662 0.1481265
 [3,] 0.663 0.1481348
 [4,] 0.664 0.1481411
 [5,] 0.665 0.1481454
 [6,] 0.666 0.1481477
 [7,] 0.667 0.1481480
 [8,] 0.668 0.1481464
 [9,] 0.669 0.1481427
> print(cbind(pi3,like3)[like3==max(like3),])
     pi3    like3
0.667000 0.148148
```

The function `cbind()` combines the vectors $pi3$ and $like3$ as column vectors into an $N \times 2$ matrix. The `print` function prints this matrix. The second `print` function prints only the row of the combined vectors for which $like3$ is equal to its maximum. This is, of course, the maximum of the likelihood function and the corresponding value of $pi3$ is the *maximum likelihood estimate* (for the degree of precision specified in the grid search, .001 in this case.)

2 Comparing OLS and MLE: The Case of Normal Regression Models

During Tuesday's lecture, we learned how to use analytic solutions to find the ML estimator for a normal regression. We learned that for the mean coefficient estimator, β, the OLS estimator is essentially the ML estimator. However, the two approaches produce different estimators for σ^2. Only with a large sample size, the ML estimator for σ^2 is asymptotically equal to the OLS estimator for σ^2. In this section, we'll use simulations to compare OLS and ML estimators with different samples, with varying sample sizes.

2.1 Simulation 1: N=21

In the first simulation, we simulate a vector of x to be a sequence of numbers between -20 and 20, and set the mesh to be 2. This generates 21 observations. Next step, we define the sample size ($n1 = 21$) and the true parameter values for intercept $b0$ and slope $b1$. We set these two parameter values to be 3.5 and 2, respectively. Then, we define the dependent variable $y1$ to be a linear combination of $b0$, $x1$, and an random error,e. We define that $e \sim N(0, 1)$. With this simulated data, we know that $y = 3.5 + 2x + e$ is the true underlying model.

——————————————- R Code————————————————-
```
x1<-seq(-20,20,2)
n1<-21
b0<-3.5
b1<-2
y1<-b0+b1*x1+rnorm(n1,0,1)
```

Next, we will fit both an OLS and an ML normal regression model, then compare results. Fitting an OLS regression in R is simple. We use the `lm()` in R. The function, `summary()` or `print` spell out the model output.

——————————————- R Code————————————————-
```
olsmodel<-lm(y1~x1)
summary(olsmodel)
```

—————————————R Output————————————————-

```
Call:
lm(formula = y1 ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.0439 -0.3968 -0.0503  0.1191  2.3193 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.45205    0.16842    20.5 2.04e-14 ***
x1           1.96478    0.01391   141.3  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7718 on 19 degrees of freedom
Multiple R-squared:  0.999,Adjusted R-squared:  0.999 
F-statistic: 1.996e+04 on 1 and 19 DF,  p-value: < 2.2e-16
```

Function `stargazer()` from the "`stargazer`" package can convert R model output in LaTeX table code, following the AJPS style. Past the LaTeX table code in .tex file, we produce Table **??**.

——————————- R Code——————————————-

```
require(stargazer)
stargazer(olsmodel)
```

Table 1: OLS Regression: Sample 1

	Dependent variable:
	y1
x1	1.965***
	(0.014)
Constant	3.452***
	(0.168)
Observations	21
R^2	0.999
Adjusted R^2	0.999
Residual Std. Error	0.772 (df = 19)
F Statistic	19,959.990*** (df = 1; 19)
Note:	*p<0.1; **p<0.05; ***p<0.01

Now, we want to estimate a normal regression model using the maximum likelihood method. Estimating an ML normal regression in `R` is not as easy as estimating an OLS model. It takes several steps. First, we use `function()` to program the log-likelihood function, which is to be maximized with respect to parameters later. In our example, we need to define three parameters: the intercept (α), the slope (β), and the variance (σ^2). The next line defines the log-likelihood function for a normal regression:

$$lnL = -\frac{1}{2}ln\sigma^2 - \frac{1}{2}\frac{[y - (\alpha + x\beta)]^2}{\sigma^2} \tag{2}$$

Once we write the log-likelihood function, we define data using our simulated sample (y1, x1, and sample size N). We then maximize the log-likelihood function with respect to the three defined parameters using function `maxLik()`. To start the optimization algorithm, we must set up a sequence of starting values, one for each parameter.

——————————- R Code————————————-

```
library(maxLik)
# Define parameters and log-likelihood function
mlnormal<- function(param) {
  alpha<- param[1]
  beta<- param[2]
  sigma <- param[3]
  ll <- -0.5*log(sigma^2) -(0.5*((y-(alpha+beta*x))^2/sigma^2))
  ll
}

# Define data
x<-x1
y<-y1
N<-21

# Maximizing the log-likelihood function
mlemodel1<-maxLik(mlnormal, start=c(0,0,1))
summary(mlemodel1)
```

——————————R Output————————————-

```
Maximum Likelihood estimation
Newton-Raphson maximisation, 16 iterations
Return code 1: gradient close to zero
Log-Likelihood: -4.009728
3  free parameters
Estimates:
     Estimate Std. error t value  Pr(> t)
[1,]  3.45205    0.16022  21.545  < 2e-16 ***
[2,]  1.96478    0.01323 148.530  < 2e-16 ***
[3,]  0.73414    0.11328   6.481 9.13e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
--------------------------------------------
```

Now, compare two models (OLS v. MLE), we see that the coefficient estimates are identical. Each model produces an intercept and slope coefficient, which are quite close to the true parameter values we set. But the OLS model produces slightly greater standard errors than the MLE model.

2.2 Simulation 2: N=501

Next, we simulate a second sample, using the same true parameter values, but increasing the sample size from 21 to 501. We repeat the analysis in Section 2.1 to see how OLS compares with MLE with a relatively large sample size. Because we have written the log-likelihood function, in this analysis, we only need to redefine the sample (data). How would you compare the two models with a large sample size?

————————————- R Code———————————————-

```
#2.2 Simulation 2: N=501
x2<-seq(-250,250,1)
n2<-501
y2<-b0+b1*x2+rnorm(n2,0,1)
# OLS,sample 2
olsmodel2<-lm(y2~x2)
# MLE,sample 2
x<-x2
y<-y2
N<- 501
mlemodel2<-maxLik(mlnormal, start=c(0,0,1))
summary(olsmodel2)
summary(mlemodel2)
```

————————————- R Output———————————————-

```
> summary(olsmodel2)
Call:
lm(formula = y2 ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2590 -0.6013  0.0393  0.6666  2.4329

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.4612627  0.0445986   77.61   <2e-16 ***
x2          2.0000415  0.0003084 6485.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9983 on 499 degrees of freedom
Multiple R-squared:      1,Adjusted R-squared:      1
F-statistic: 4.207e+07 on 1 and 499 DF,  p-value: < 2.2e-16

> summary(mlemodel2)
--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 22 iterations
Return code 2: successive function values within tolerance limit
Log-Likelihood: -248.621
3  free parameters
Estimates:
      Estimate Std. error t value Pr(> t)
```

```
[1,] 3.4612627  0.0445046   77.77  <2e-16 ***
[2,] 2.0000415  0.0003078 6498.80  <2e-16 ***
[3,] 0.9962566  0.0314735   31.65  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
--------------------------------------------
```

2.3 Softeware Issues: R and Stata

In this lab, we use `maxLik()` to estimate normal regression models. When using `maxLik`, we must provide log-likelihood function and choose different optimization algorithms (the default is Newton). There are lots of optimizers in *R*.

- `maxLik` package: options for Newton-Raphson, BHHH, BFGS, others.
- `optima` (in `stats`): quasi-Newton, plus others.
- `newton` (in `stats`): Newton-Raphson solver.
- `solveLP` (in `linprog`): linear programming optimizer.

In `Stata`, `ml` is the command that we can use to implement maximum likelihood estimation. The syntax is:

```
.ml model <method> <progname> <eq>...
.ml maximize
```

- An example with logistic likelihood (if y=1)

$$f(y, xb) = \frac{1}{1 + exp(-xb)} \tag{3}$$

———————————Stata Code———————————

```
sysuse auto.dta
program define mylogit
          args lnf Xb
          replace 'lnf' = -ln(1+exp(-'Xb')) if $ML_y1==1
          replace 'lnf' = -'Xb' - ln(1+exp(-'Xb')) if $ML_y1==0
  end

ml model lf mylogit(foreign=mpg weight)
ml maximize
```

———————————Stata Output———————————

```
Iteration 5:   log likelihood = -27.175156

                                                  Number of obs   =         74
                                                  Wald chi2(2)    =      17.78
Log likelihood = -27.175156                       Prob > chi2     =     0.0001

------------------------------------------------------------------------------
```

```
     foreign |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -.1685869   .0919175    -1.83   0.067    -.3487418     .011568
      weight |  -.0039067   .0010116    -3.86   0.000    -.0058894    -.001924
       _cons |   13.70837   4.518709     3.03   0.002     4.851859    22.56487
------------------------------------------------------------------------------
```