Econometrics II TA Session

Wang Xin

Oct 23, 2023

1 Correction

We consider the simple regression model

$$Y_i = \alpha + \beta X_i + u_i$$

We assume $u_i|X_i \sim \mathcal{N}(0, \sigma^2)$. As a consequence, the density is

$$p_{u;\theta}(u_i|X_i) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{u_i^2}{2\sigma^2}\right).$$

$$\begin{aligned} M_n(\theta) &= \sum_{i=1}^{n} \log p_\theta(Y_i, X_i) = \sum_{i=1}^{n} \log p_\theta(Y_i|X_i) + \sum_{i=1}^{n} \log p_\theta(X_i) \\ &= \sum_{i=1}^{n} \log p_\theta(Y_i|X_i) \end{aligned}$$

Since X_i contains no information about the parameters. Thus is can be discarded.

2 MLE Estimation in R

We start with a multiple linear model with only 2 independent variables

$$y_i = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + u_i \qquad u_i|\mathbf{x}_i \sim \mathcal{N}(0, \sigma^2)$$

where $\mathbf{x}_i = (1, x_{i1}, x_{i2})$, and the parameters should be $\boldsymbol{\theta} = (\alpha, \beta_1, \beta_2, \sigma^2)$

Now we generate some random values for the variables as the data set.

```
set.seed(243)

Nsim = 10^4
alpha0 = 1
beta10 = 2
```

```
beta20 = 3
sigma20 = 10 # set values of true parameters
theta0 = as.vector(c(alpha0, beta10, beta20, sigma20))

const = rep(1, Nsim)
x1 = rnorm(Nsim, mean = 5, sd = 1)
x2 = rnorm(Nsim, mean = 10, sd = 1)
u = rnorm(Nsim, mean = 0, sd = sqrt(sigma20))
y = alpha0*const + beta10*x1 + beta20*x2 + u

y = as.vector(y)
const = as.vector(const)
x1 = as.vector(x1)
x2 = as.vector(x2)
X = cbind(const, x1, x2)
```

In a way, the most important point is that whether the log likelihood function is correctly written or not. Before entering this part, let's review the log-likelihood function of linear model. Suppose all conditions hold, especially, we assume that (y_i, x_{i1}, x_{i2}) are iid. Then the log-likelihood function should be written as

$$M_n(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)$$

The equation in R language suppose to be

```
# Writing log-likelihood function
log_likeli = function(para) {
n = nrow(X)
k = ncol(X)
beta = para[1:k]
sigma2 = para[k+1]
R = -(n/2)*log(2*pi*sigma2) - (1/(2*sigma2))*t((y - X %*% beta)
    )
  %*%(y - X%*% beta)
# In R, %*% means product between matrices(vectors)
return(R)
}
```

2.1 Estimating by "optim" Function

In R, the default function to do minimization(maximization) is optim(optimize for one-dimensional case), so insert the log-likelihood function into optim and don't forget to

give the appropriate arguments.

```
# Estimation by optim function
n = nrow(X)
k = ncol(X)
opti1 = optim(par = c(0 ,0 ,0 ,1), fn = log_likeli , control =
  list(fnscale = -1), hessian = TRUE)
# Default method is Nelder-Mead
# For maximization, setting control = list(fnscale = -1)
#   (minimization is default)
# hessian = TRUE means returning a numerically differentiated
#   hessian matrix.
```

Comparing the true parameter $\boldsymbol{\theta}_0$ with the estimates, then we can find they are approximately equal to each other.

```
> theta0
[1]  1  2  3 10
> opti1
$par
[1]  1.098523  1.960113  3.012267 10.025663

$value
[1] -25724.5

$counts
function gradient
     353       NA

$convergence
[1] 0

$message
NULL

$hessian
                [,1]          [,2]          [,3]         [,4]
[1,] -9.974403e+02 -4.979533e+03 -9.993465e+03   0.01228636
[2,] -4.979533e+03 -2.586452e+04 -4.989133e+04   0.19930576
[3,] -9.993465e+03 -4.989133e+04 -1.011206e+05   0.08225152
[4,]  1.228636e-02  1.993058e-01  8.225152e-02 -49.93094626
```

To calculate the empirical variance, one should not forget that the hessian here is the SOC of criterion function, which is Fisher's information matrix. Therefore, the variance of each estimate should be the inverse of the elements on the diagonal.

```
> diag(abs(solve(opti1$hessian))) # variance
[1] 0.1266031767 0.0009948923 0.0010050341 0.0200276680
> diag(sqrt(abs(solve(opti1$hessian)))) # sd
[1] 0.35581340 0.03154191 0.03170227 0.14151914
```

2.2 Estimating by "bbmle::mle2" Function

Various external packages are also provided to extend the calculating ability of programming software. In R, package "bbmle" gathers tools for general maximum likelihood estimation, and "mle2" function is used to estimate MLE.

```
library(bbmle)
# Attention : The objective of mle2 function is minus log-
#   likelihood function(minimizaiton), so we need to rewrite it
log_likeli_2 = function(para) {
n = nrow(X)
k = ncol(X)
beta = para[1:k]
sigma2 = para[k+1]
R = -(n/2)*log(2*pi*sigma2) - (1/(2*sigma2))*t((y - X %*% beta)
    )
  %*%(y - X%*% beta)
return(-R)
}

# Running mle2
parnames(log_likeli_2) = c("alpha","beta1","beta2","sigma2")
# Assign names to"para"in log_likeli_2

opti2 = mle2(minuslogl = log_likeli_2, start = list(alpha = 0,
  beta1 = 0, beta2 = 0, sigma2 = 1), vecpar = TRUE)
```

The results can be checked as

```
> opti2@details
$par
```

```
    alpha      beta1      beta2     sigma2
 1.097025   1.958944   3.012995 10.044988

$value
[1] 25724.49

$counts
function gradient
      87       23

$convergence
[1] 0

$message
NULL

$hessian
                [,1]          [,2]          [,3]          [,4]
[1,]  9.955213e+02  4.969953e+03  9.974238e+03 -0.008776204
[2,]  4.969953e+03  2.581476e+04  4.979534e+04 -0.064580850
[3,]  9.974238e+03  4.979534e+04  1.009261e+05 -0.119164480
[4,] -8.776204e-03 -6.458085e-02 -1.191645e-01 49.547960958

$maxgrad
[1] 1.197005

$eratio
[1] 6.172317e-05

> summary(opti2)
Maximum likelihood estimation

Call:
mle2(minuslogl = log_likeli_2, start = list(alpha = 0, beta1
    = 0, beta2 = 0, sigma2 = 1), vecpar = TRUE)

Coefficients:
        Estimate Std. Error z value     Pr(z)
alpha    1.097025   0.356156  3.0802  0.002069 **
beta1    1.958944   0.031572 62.0463 < 2.2e-16 ***
beta2    3.012995   0.031733 94.9489 < 2.2e-16 ***
sigma2  10.044988   0.142065 70.7070 < 2.2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

-2 log L: 51448.99
```

The standard errors and p values can also be checked by using summary function. Comparing standard errors derived by two functions, they are very close to each other(standard errors from optim are a little bit smaller). And covariance can be obtained either by function vcov or hessian in this case.

```
> diag(solve(opti2@details$hessian))
[1] 0.1268472289 0.0009968098 0.0010069714 0.0201824659
> diag(sqrt(abs(solve(opti2@details$hessian))))
[1] 0.35615619 0.03157229 0.03173281 0.14206501
```

3 Discrete Choice Model - Binary Model

A binary model focuses on the dependent variable y_i which is equal to 1 or 0 exclusively, considering the explanatory variable x_i. Our approach will be to analyze each of them in the general framework of probability models:

$$P(y_i) = P(y_i = 1)^{y_i} (1 - P(y_i = 1))^{1-y_i}$$

Discrete dependent-variable models are often cast in the form of index function models. We view the outcome of a discrete choice as a reflection of an underlying regression, which represented by an unobserved variable y^* such that

$$\mathbf{y}_i^* = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i \qquad \epsilon_i \sim (0, \sigma^2)$$

Then, the probability that y equals one is

$$P(y_i = 1) = P(y_i^* > 0) = P(\epsilon_i < \mathbf{x}_i\boldsymbol{\beta}) = F(\mathbf{x}_i\boldsymbol{\beta})$$

where $F(t)$ is the cdf of the random variable, ϵ_i. This provides an underlying structural model for the probability.

The problem at this point is to devise a suitable model for the right-hand side of the equation. One possibility is to retain the familiar linear regression,

$$\cancel{F(\mathbf{x}_i\boldsymbol{\beta}) = \mathbf{x}_i\boldsymbol{\beta}}$$

The linear probability model has a number of shortcomings. A most serious flaw is that without some ad hoc tinkering with the disturbances, we cannot be assured that the predictions from this model will truly look like probabilities. We cannot constrain $\mathbf{x}_i\boldsymbol{\beta}$ to the 0-1 interval.

The normal distribution has been used in many analyses, giving rise to the probit model,

$$P\left(y_i = 1\right) = \int_{-\infty}^{\mathbf{x}_i\boldsymbol{\beta}} \phi(t)dt = \Phi(\mathbf{x}_i\boldsymbol{\beta})$$

Partly because of its mathematical convenience, the logistic distribution,

$$P\left(y_i = 1\right) = \frac{exp\left(\mathbf{x}_i\boldsymbol{\beta}\right)}{1 + exp\left(\mathbf{x}_i\boldsymbol{\beta}\right)} = \Lambda\left(\mathbf{x}_i\boldsymbol{\beta}\right)$$

has also been used in many applications.

The probability of observing y_i for an individual i can be written as

$$\begin{aligned} P\left(y_i\right) &= P\left(y_i = 1\right)^{y_i}\left(1 - P\left(y_i = 1\right)^{1-y_i}\right) \\ &= F(\mathbf{x}_i\boldsymbol{\beta})^{y_i}\left(1 - F(\mathbf{x}_i\boldsymbol{\beta})\right)^{1-y_i} \end{aligned}$$

which leads to the joint probability, or likelihood function

$$\begin{aligned} M_n(\boldsymbol{\beta}) &= \log\left(\prod_{i=1}^{n} F(\mathbf{x}_i\boldsymbol{\beta})^{y_i}\left(1 - F(\mathbf{x}_i\boldsymbol{\beta})\right)^{1-y_i}\right) \\ &= \sum_{i=1}^{n}\left(y_i \log\left(F(\mathbf{x}_i\boldsymbol{\beta})\right) + \left(1 - y_i\right)\log\left(1 - F(\mathbf{x}_i\boldsymbol{\beta})\right)\right). \end{aligned}$$

The optimal $\hat{\beta}$ can be calculated when FOC of $M_n(\beta)$ equals 0. That is

$$\frac{\partial M_n(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n}\left(y_i - F(\mathbf{x}_i\boldsymbol{\beta})\right)\frac{f(\mathbf{x}_i\boldsymbol{\beta})}{F(\mathbf{x}_i\boldsymbol{\beta})\left(1 - F(\mathbf{x}_i\boldsymbol{\beta})\right)}\mathbf{x}_i' = 0.$$