Maximum Likelihood Estimation by R

MTH 541/643

Instructor: Songfeng Zheng

In the previous lectures, we demonstrated the basic procedure of MLE, and studied some examples. In the studied examples, we are lucky that we can find the MLE by solving equations in closed form. But life is never easy. In applications, we usually don't have closed form solutions due to the complicated probability distribution or the nonlinearity of the equations obtained from maximum likelihood principles. In these situations, we can use a computer to solve the problem. Many statistics software package has MLE as a standard procedure, but for the purpose of learning MLE and for the purpose of learning programming language, let us develop the code ourselves.

In this course, we use R for our computer programming. R is free software, and you can download it from http://www.r-project.org/. R is one of the most widely used software packages in the statistics community these days, together with SAS, SPSS, SPLUS, STATA, among others.

We will introduce the R programming for MLE via an example:

The Poisson distribution has been used by traffic engineers as a model for light traffic, based on the rationale that if the rate is approximately constant and the traffic is light (so the individual cars move independently of each other), the distribution of counts of cars in a given time interval or space area should be nearly Poisson (Gerlough and Schuhl 1955). The following table shows the number of right turns during 300 3-minute intervals at a specific intersection.

n	frequency
0	14
1	30
2	36
3	68
4	43
5	43
6	30
7	14
8	10
9	6
10	4
11	1
12	1

13+	0

If we suppose Poisson model might be a good model for this dataset, we still need to find out which Poisson, that is estimate the parameter λ in the Poisson model:

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}.$$

Of course, we can use the formula to calculate MLE of the parameter λ in the Poisson model as: $\hat{\lambda} = \overline{X}$ (please check this yourselves.) For the purpose of demonstrating the use of R, let us just use this Poisson distribution as an example.

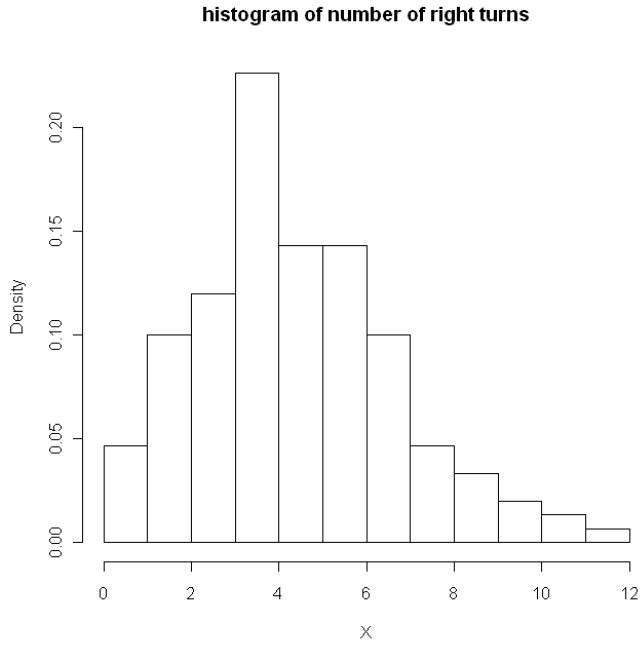
The first step is of course, input the data. If the data are stored in a file (*.txt, or in excel format), we can directly read the data from file. In this example, we can input the data directly:

X <- c(rep(0,14), rep(1,30), rep(2,36), rep(3,68), rep(4, 43), rep(5,43), rep(6, 30), rep(7,14), rep(8,10), rep(9, 6), rep(10,4), rep(11,1), rep(12,1))

Here rep(x, n) means repeat the number x for n times, and c( ) is the operation to combine all the numbers into a long vector. We can also plot out the histogram of the data, and compare to the point mass function of Poisson, to gain a rough impression whether a Poisson model is good or not.

hist(X, main="histogram of number of right turns", right=FALSE, prob=TRUE)

and the following is the graph for histogram:



If we believe the Poisson model is good for the data, we need to estimate the parameter. Let's first get the size of the sample by using the following command:

```
n <- length(X)
```

In order to obtain the MLE, we need to maximize the likelihood function or log likelihood function. The R package provides a function which can minimize an object function, therefore, we can define the negative log likelihood function as follows:

```
negloglike<-function(lam) { n* lam -sum(X) *log(lam) + sum(log(factorial(X))) }
```

Here the first line "negloglike<-function(lam)" defines a function, and the function name is called "negloglike", and the input parameter of the function is "lam", "function" is a keyword in R, which means the following is a function. The argument "lam" can also be a vector in the case of more than two unknown parameters (see the exercise). The statement inside the bracket is the body of the function. Please note that, there should not be any unknown values in the function body except for the input parameter lam. For example, here, you know X, and n. Once the function is defined in R, you can evaluate the function value by giving it a value for lam. For example, you can type in

```
negloglike(0.3)
```

to find the negative log likelihood at $\lambda = 0.3$.

After we define the negative log likelihood, we can perform the optimization as following:

```
out<-nlm(negloglike,p=c(0.5), hessian = TRUE)
```

here "nlm" is the nonlinear minimization function provided by R, and the first argument is the object function to be minimized; the second argument "p=c(0.5)", specifies the initial value of the unknown parameter, here we start at 0.5; the third argument, "hessian = TRUE" says that we want the Hessian matrix as a part of the output result. The optimization result and relevant information will be stored in a structure, and in this example, the variable "out" stores the structure.

You may get some warning messages, like the following, which does not matter:

Warning messages:

1: In log(p) : NaNs produced

2: In nlm(fn, p = c(0.5), hessian = TRUE) :

NA/Inf replaced by maximum positive value

3: In log(p) : NaNs produced

4: In nlm(fn, p = c(0.5), hessian = TRUE) :

NA/Inf replaced by maximum positive value

If you want to see the result, just type in the variable name "out" in this case, and you will have the following information:

out

$minimum

[1] 667.183

$estimate

[1] 3.893331

$gradient

[1] -2.575476e-05

$hessian

[,1]

[1,] 77.03948

$code

[1] 1

$iterations

[1] 10

Where, *out$minimum* is the negative log-likelihood, *out$estimate* are the maximum likelihood estimates of the parameters, *out$gradient* is the gradient of the negative log-likelihood function at this point, *out$hessian* is the value of the second derivative at this point, *out$iterations* is the number of iterations need to converge. The notation "$" is to take the component of the output variable "out".

For this Poisson distribution, it is well-known that the MLE is the mean value of the values, type in command to find the mean value:

mean(X)

[1] 3.893333

Which is very close to the result from the code.

Exercise: (Please fit a gamma distribution, plot the graphs, turn in the results and code! Please type your solution to a file.)

The file *gamma-arrivals.txt* contains another set of gamma-ray data, this one consisting of the times between arrivals (inter-arrival times) of 3935 photons (units are seconds). Assume the Gamma distribution is a good model for the data:

$$f(x \mid \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad \text{for } x \geq 0$$

where both alpha and beta are unknown.

1. *Make a histogram of the inter-arrival times. Does it appear that a gamma distribution would be a plausible model?*
2. *Fit the parameters by the method of moments and maximum likelihood. How do the estimate compare?*
3. *Plot the two fitted gamma densities on top of the histogram. Do the fits look reasonable?*

Notes:

a. in order to read the data, if you are connected to Internet, you need the following:

x = scan("http://people.missouristate.edu/songfengzheng/Teaching/MTH541/gamma-arrivals.txt")

The url can be replaced by the file path, if desired. For example:

x = (scan(" C:\\TEACHING\\MATH 541\\Data_Rice\\ASCII Comma\\Chapter 8\\gamma-arrivals.txt "))

This command also has the advantage of importing the data as a vector instead of a data frame, making the subsequent conversion unnecessary.

b. You don't need to use the function "nlm" to do Method of Moment, instead, you need to derive the formula for the estimator, and write the code to calculate directly.

c. to plot the density of a distribution, use the following:

curve(dgamma(x, shape=alpha_hat_MLE, scale=1.0/beta_hat_MLE))

d. to put the density on top of the histogram, you can use the following:

curve(dgamma(x, shape=alpha_hat_MLE, scale=1.0/beta_hat_MLE),col='blue',add=TRUE)