Estimating The Location Parameter of a Cauchy Distribution With a Known Scale Parameter

5361 Homework 3

Qinxiao Shi *

22 September 2018

1 Proofs

1.1 Fisher Information

The function of cauchy distribution, whose location parameter is θ, scale parameter is 1:

$$f(x;\theta) = \frac{1}{\pi[1+(x-\theta)^2]}, x \in R, \theta \in R.$$

Then we can calculate for fisher information:

$$L(\theta) = \frac{1}{\pi^n \prod_{i=1}^n [1+(\theta - X_i)^2]}$$

$$\ell(\theta) = \log(L(\theta)) = -\log(\pi^n) - \log(\prod_{i=1}^n [1+(\theta-X_i)^2])$$

$$= -n\ln\pi - \sum_{i=1}^n \ln[1+(\theta-X_i)^2]$$

$$\ell'(\theta) = 0 - 2\sum_{i=1}^n \frac{\theta - X_i}{1+(\theta-X_i)^2} = -2\sum_{i=1}^n \frac{\theta - X_i}{1+(\theta-X_i)^2}$$

$$\ell''(\theta) = -2\sum_{i=1}^n (\frac{1}{1+(\theta-X_i)^2} - \frac{\theta - X_i}{1+(\theta-X_i)^2})$$
$$= -2\sum_{i=1}^n \frac{1-(\theta-X_i)^2}{(1+(\theta-X_i)^2)^2}$$

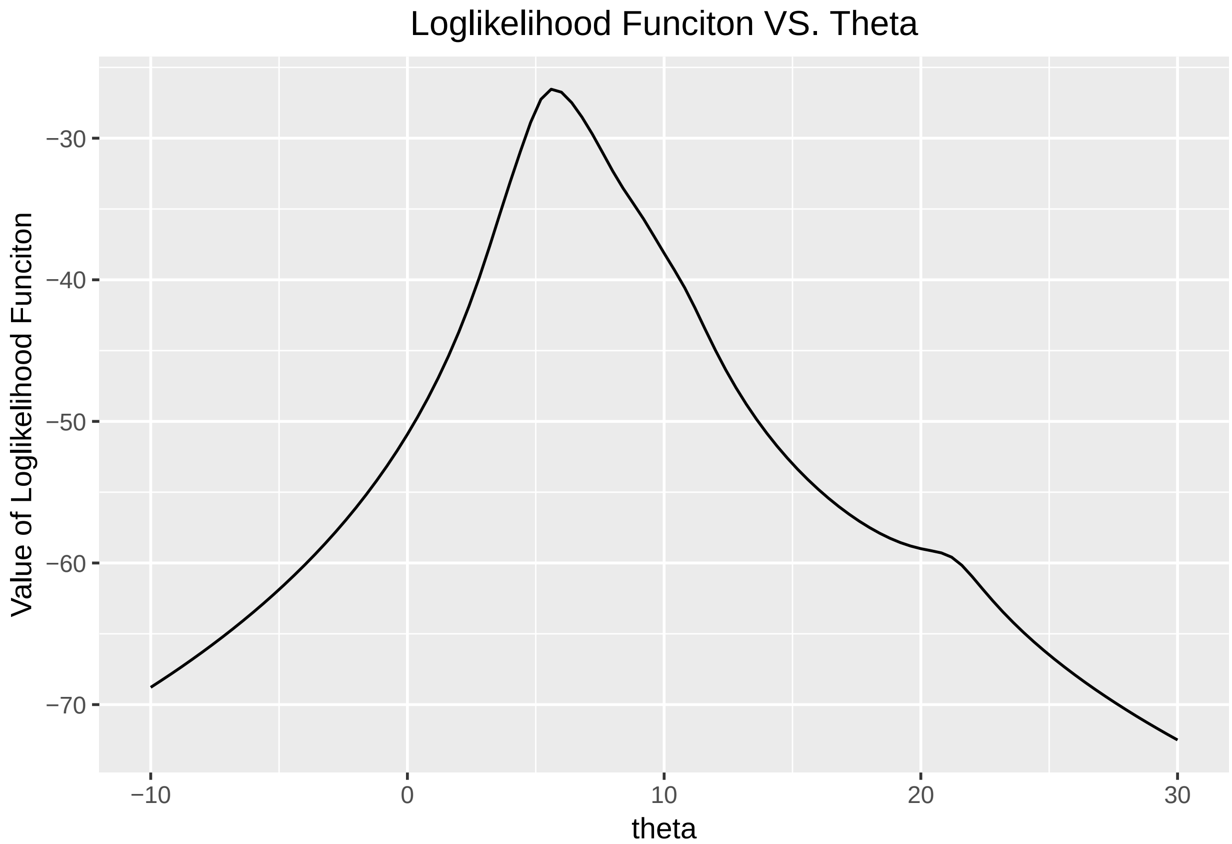
*qinxiao.shi@uconn.edu

$$
\begin{aligned}
I_n(\theta) = -E(\ell''(\theta)) = 2n \int_{-\infty}^{\infty} \frac{1-(\theta - X_i)^2}{(1+(\theta-X_i)^2)^2} \frac{1}{\pi(1+(x-\theta)^2)} dx \\
&= \frac{2n}{\pi} \int_{-\infty}^{\infty} \frac{1-x^2}{(1+x^2)^2} \frac{1}{1+x^2} dx \\
&= \frac{2n}{\pi} [\frac{1}{\frac{1}{x^2}+1} \Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty} \frac{2x^2}{(1+x^2)^3} dx] \\
&= \frac{2n}{\pi} [0 + \int_{-\infty}^{\infty} \frac{2x^2}{(1+x^2)^3} dx] = \frac{4n}{\pi} \int_{-\infty}^{\infty} \frac{x^2}{(1+x^2)^3} dx \\
&= \frac{4n}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\tan^2 t}{[1+\tan^2 t)]^3} d \tan t = \frac{4n}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (\sin^2 t \cos^2 t) dt \\
&= \frac{n}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (\sin^2 2t) dt = \frac{n}{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (1-\cos 4t) dt \\
&= \frac{n}{2\pi} \times \pi = \frac{n}{2}
\end{aligned}
$$

1.2 Loglikehood Function Plot

The plot below shows the loglikelihood function against $\theta = 5$ when sample size $n = 10$

```r
library("ggplot2")
set.seed(20180909)
cauchy <- rcauchy(n=10, location=5, scale=1)

y <- function(cauchy, x){
  y <- 0;
  for(i in 1:length(cauchy)){
  y <- y-log(pi)-log(1+(x-cauchy[i])^2)
  }
  return(y)
}

ggplot(data.frame(x=c(-10,30)), aes(x=x))+
  stat_function(fun = function(x) y(cauchy, x))+
  ggtitle("Loglikelihood Funciton VS. Theta")+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(y="Value of Loglikelihood Funciton", x="theta")
```



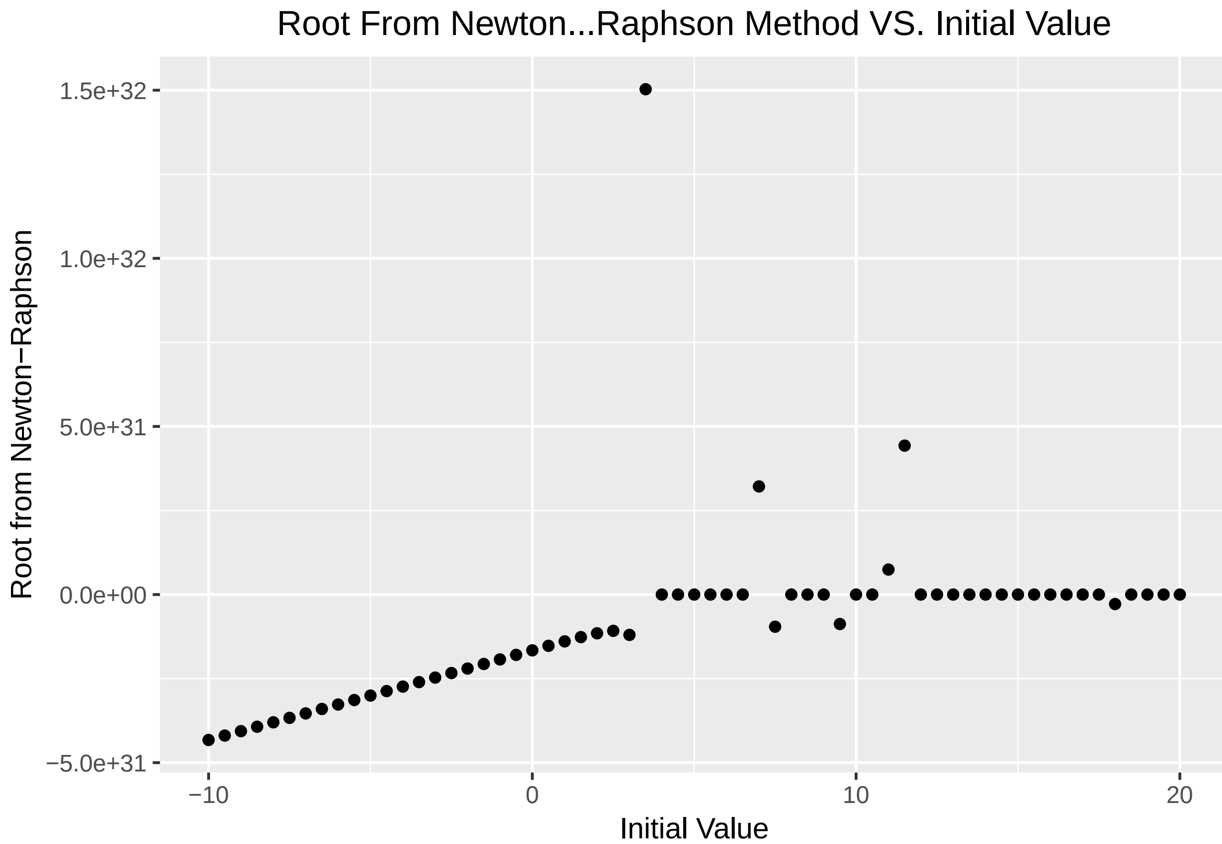
2 Newton-Raphson Method

2.1 Find MLE

The plot of likelihood function vs. θ shows that the MLE is in the range of $\theta \in (5, 10)$, and it is pretty close to $\theta = 5$. Next step is finding the MLE of θ using the Newton–Raphson method with initial values on a grid starting from -10 to 30 with increment 0.5

```
library("pracma")
library("pander")
library("gridExtra")
library("grid")
library("knitr")
library("kableExtra")


f <- function(cauchy, x){
  f <- sum(dcauchy(cauchy, location = x, scale = 1, log = TRUE))
  return(f)
}
```

```r
func1 <- function(cauchy, x){
  f <- sapply(x, FUN = function(x) f(cauchy, x))
  return(f)
}

gradient <- function(x){
  gradient <- 0
  for (i in 1:length(cauchy)) {
    gradient <- gradient-2*(x-cauchy[i])/(1+(x-cauchy[i])^2)
  }
  return(gradient)
}

hessian <- function(x){
  hessian <- 0
  for (i in 1:length(cauchy)) {
    hessian <- hessian-2*(1-(x-cauchy[i])^2)/(1+(x-cauchy[i])^2)^2
  }
  return(hessian)
}


init <- seq(-10, 20, by=0.5)

  newton <- newtonRaphson(fun=function(x) gradient(x), x0=init,
                          dfun=function(x) hessian(x))
  root <- newton$root

  raphson <- data.frame(init = init, root = root)
  colnames(raphson) <- c('Initial Value', 'Root')

ggplot(raphson, aes(x=init, y=root))+ geom_point()+
  ggtitle("Root From Newton-Raphson Method VS. Initial Value")+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(y="Root from Newton-Raphson", x="Initial Value")
```




```
kable(raphson[1:31,], booktabs = TRUE, align = 'c', row.names = 1)
```

	Initial Value	Root
1	-10.0	-4.324741e+31
2	-9.5	-4.193577e+31
3	-9.0	-4.062249e+31
4	-8.5	-3.930748e+31
5	-8.0	-3.799064e+31
6	-7.5	-3.667185e+31
7	-7.0	-3.535100e+31
8	-6.5	-3.402796e+31
9	-6.0	-3.270261e+31
10	-5.5	-3.137479e+31
11	-5.0	-3.004436e+31
12	-4.5	-2.871118e+31
13	-4.0	-2.737510e+31
14	-3.5	-2.603599e+31
15	-3.0	-2.469374e+31
16	-2.5	-2.334832e+31
17	-2.0	-2.199981e+31
18	-1.5	-2.064850e+31
19	-1.0	-1.929508e+31
20	-0.5	-1.794100e+31
21	0.0	-1.658922e+31
22	0.5	-1.524582e+31
23	1.0	-1.392396e+31
24	1.5	-1.265439e+31
25	2.0	-1.151924e+31
26	2.5	-1.079358e+31
27	3.0	-1.199750e+31
28	3.5	1.502957e+32
29	4.0	2.056366e+01
30	4.5	2.108229e+01
31	5.0	5.685422e+00

```
kable(raphson[32:61,], booktabs = TRUE, align = 'c', row.names = 1)
```

	Initial Value	Root
32	5.5	5.685422e+00
33	6.0	5.685422e+00
34	6.5	5.685422e+00
35	7.0	3.215974e+31
36	7.5	-9.558888e+30
37	8.0	1.937744e+01
38	8.5	2.108229e+01
39	9.0	5.685422e+00
40	9.5	-8.759488e+30
41	10.0	2.108229e+01
42	10.5	5.685422e+00
43	11.0	7.439560e+30
44	11.5	4.429077e+31
45	12.0	2.056366e+01
46	12.5	2.056366e+01
47	13.0	2.056366e+01
48	13.5	2.108229e+01
49	14.0	2.108229e+01
50	14.5	2.108230e+01
51	15.0	1.937743e+01
52	15.5	2.056366e+01
53	16.0	2.056366e+01
54	16.5	1.937744e+01
55	17.0	1.937743e+01
56	17.5	1.937744e+01
57	18.0	-2.825479e+30
58	18.5	2.056366e+01
59	19.0	2.056366e+01
60	19.5	2.056366e+01
61	20.0	2.056366e+01

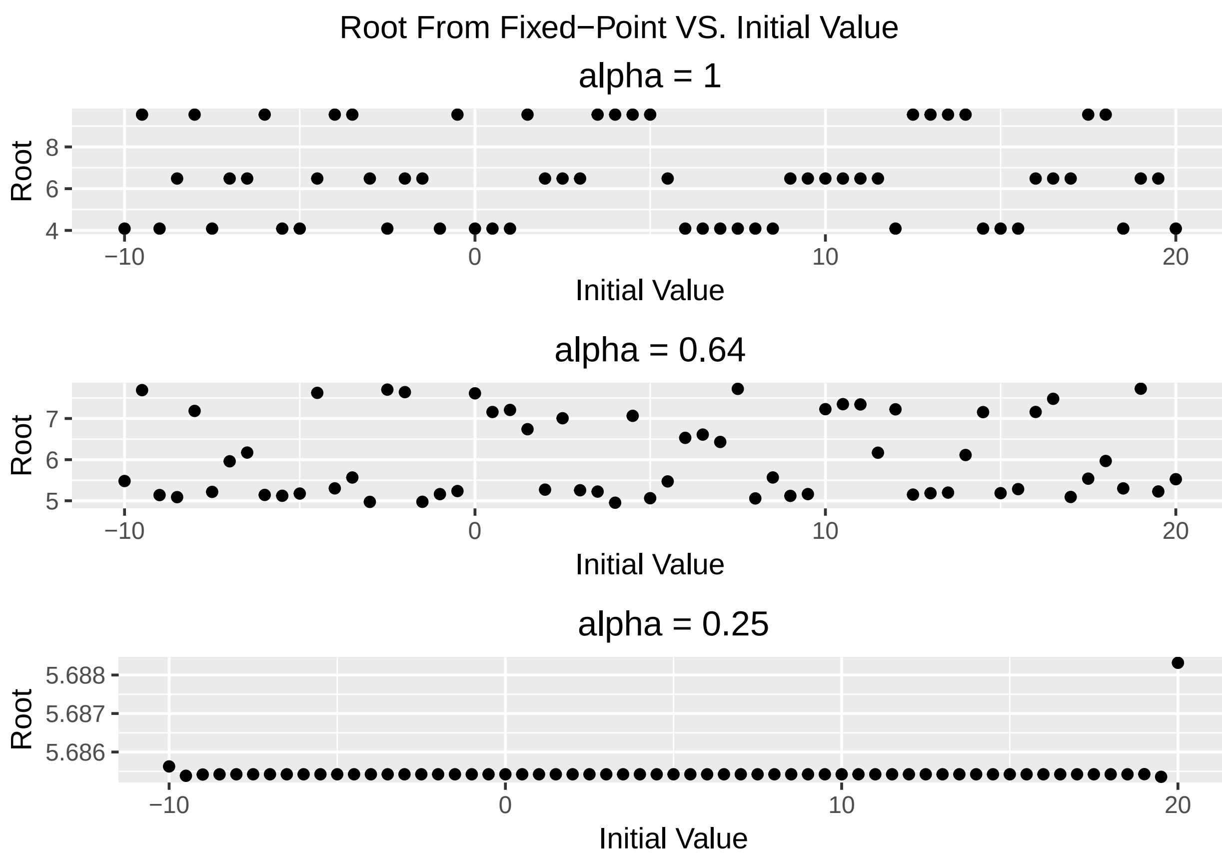
2.2 Summarization

From the table, it is obvious that the roots from Newton-Raphson method do not converge with initial value going large.

3 Fixed-Point Iteration

```
fxpt <- function(fun, init, alpha, maxiter = 100, tol = .Machine$double.eps^0.2){
  for (i in 1:maxiter) {
    init1 <- alpha*fun(init) + init
```

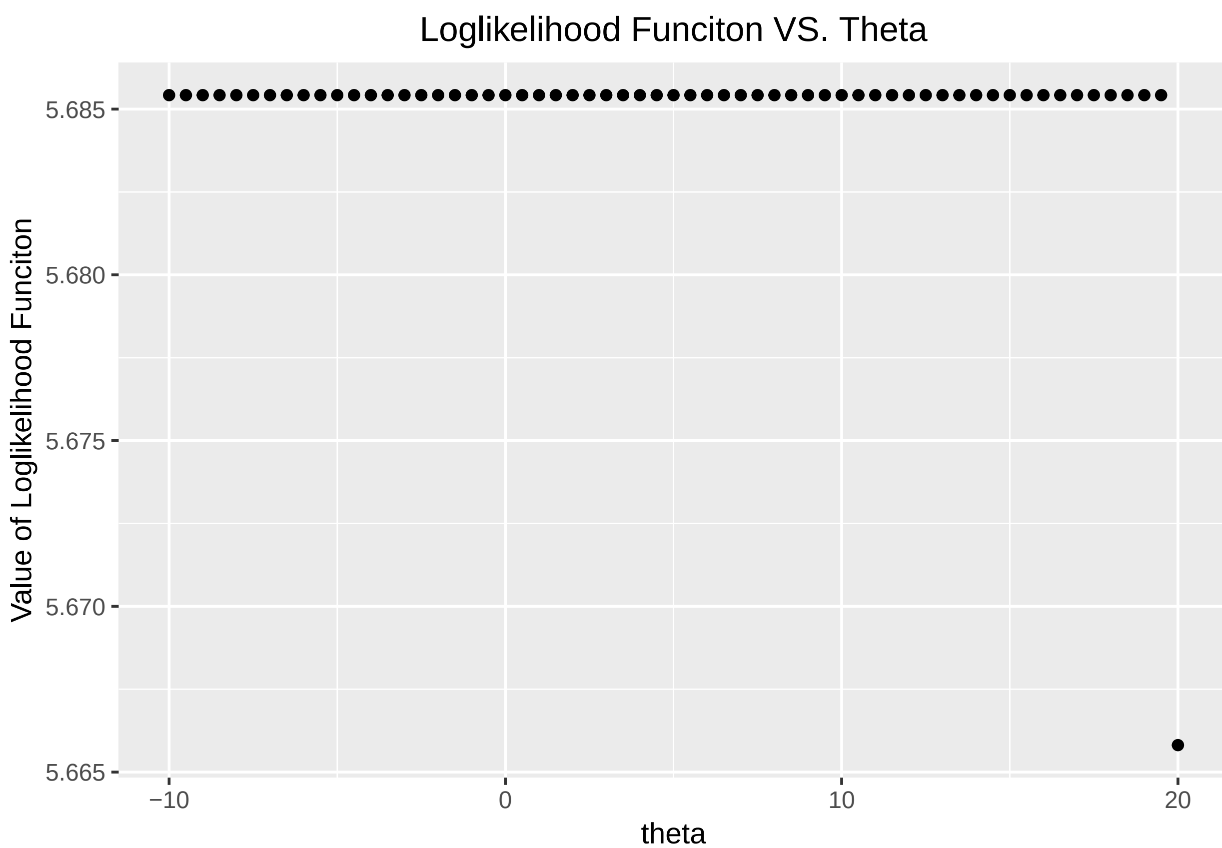
```r
    if(abs(init1 - init) < tol) break
    init <- init1
  }
  if(i == maxiter)
    warning("Reached the maximum iteration!")
  return(data.frame(root = init, niter = i))
}

root.fxpt <- matrix(NA, nrow = length(init), ncol = 4)

for (i in 1:length(init)) {
  root.fxpt[i,1] <- init[i]
}

fxptfunc1 <- fxpt(fun = function(x) gradient(x), init = init, alpha = 1)
  root.fxpt[,2] <- fxptfunc1$root

fxptfunc2 <- fxpt(fun = function(x) gradient(x), init = init, alpha = 0.64)
  root.fxpt[,3] <- fxptfunc2$root

fxptfunc3 <- fxpt(fun = function(x) gradient(x), init = init, alpha = 0.25)
  root.fxpt[,4] <- fxptfunc3$root

table2 <- as.data.frame(root.fxpt)

p1 <- ggplot(table2, aes(x = V1, y = V2))+
  geom_point()+
  labs(x = "Initial Value", y = "Root")+
  ggtitle("alpha = 1")+
  theme(plot.title = element_text(hjust = 0.5))

p2 <- ggplot(table2, aes(x = V1, y = V3))+
  geom_point()+
  labs(x = "Initial Value", y = "Root")+
  ggtitle("alpha = 0.64")+
  theme(plot.title = element_text(hjust = 0.5))

p3 <- ggplot(table2, aes(x = V1, y = V4))+
  geom_point()+
  labs(x = "Initial Value", y = "Root")+
  ggtitle("alpha = 0.25")+
  theme(plot.title = element_text(hjust = 0.5))

gridExtra::grid.arrange(p1, p2, p3, nrow=3,
                        top="Root From Fixed-Point VS. Initial Value")
```



4 Fisher Scoring and Newton-Raphson

```
options(digits = 8)
fsh <- function(fun, init, In, maxiter = 100, tol = .Machine$double.eps^0.2)
{
  for (i in 1:maxiter) {
    init1 <- init + fun(init)/In
    if(abs(init1 - init) < tol) break
    init <- init1
  }
  if(i == maxiter)
    message("Reached the maximum iteration!")

  return(data.frame(root = init1, iter = i))
}

root.fsh <- matrix(NA, nrow = length(init), ncol = 2)
fs <- fsh(fun = function(x) gradient(x), init = init, In = 5)
fsroot <- fs$root
NR <- newton(x0 = fsroot, fun = function(x) gradient(x), dfun = function(x) hessian(x))
```

```r
root.NR <- NR$root
table3 <- data.frame(init, root.NR)

ggplot(table3, aes(x = init, y = root.NR))+
  geom_point()+
  ggtitle("Loglikelihood Funciton VS. Theta")+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(y="Value of Loglikelihood Funciton", x="theta")
```



5 Comment

In conclusion, roots from Newton-Raphson method may be converging or not due to initial values, which means the method is not stable. But when we fix the iteration points, the roots can converge in a very fast speed in $\alpha = 0.25$. If using fisher scoring to find MLE of θ, the effect is much better then the Newton-Raphson method only as well.