Cauchy

Yaqiong Yao

9/21/2018

Q1

We know the density of Cauchy distribution is:

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

The likelihood function can be represented as:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} f(X_i;\theta) \\ &= \prod_{i=1}^{n} \frac{1}{\pi[1+(X_i-\theta)^2]}. \end{aligned}$$

Thus, the log-likelihood function is:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{n} \log\left(\frac{1}{\pi[1+(X_i-\theta)^2]}\right) \\ &= -n\log\pi - \sum_{i=1}^{n} \log[1+(X_i-\theta)^2]. \end{aligned}$$

The first derivative is:

$$\begin{aligned} \ell'(\theta) &= \frac{\partial \ell(\theta)}{\partial\theta} \\ &= -2\sum_{i=1}^{n} \frac{\theta - X_i}{1+(X_i-\theta)^2}. \end{aligned}$$

The second derivative is:

$$\begin{aligned} \ell''(\theta) &= \frac{\partial \ell'(\theta)}{\partial\theta} \\ &= -2\sum_{i=1}^{n} \frac{1+(X_i-\theta)^2 - 2(X_i-\theta)^2}{[1+(X_i-\theta)^2]^2} \\ &= -2\sum_{i=1}^{n} \frac{1-(X_i-\theta)^2}{[1+(X_i-\theta)^2]^2}. \end{aligned}$$

The fisher information for one sample is:

$$
\begin{aligned}
I_1(\theta) &= E\left[-\frac{\partial^2 \ell(\theta)}{\partial\theta^2}\middle|\theta\right] \\
&= E\left[\frac{2[1-(\theta-X)^2]}{(1+(\theta-X)^2)^2}\middle|\theta\right] \\
&= \frac{2}{\pi}\int_{-\infty}^{\infty}\frac{1-(\theta-X)^2}{(1+(\theta-X)^2)^3}dx \\
&= \frac{2}{\pi}\int_{-\infty}^{\infty}\frac{1-t^2}{(1+t^2)^3}dt \\
&= \frac{2}{\pi}\left(\int_{-\infty}^{\infty}\frac{2}{(1+X^2)^3}-\int_{-\infty}^{\infty}\frac{1}{(1+X^2)^2}\right)dt
\end{aligned}
$$

where $t=(\theta-X)$. Then the question becomes how to derive $M_k=\int_{-\infty}^{\infty}\frac{1}{(1+t^2)^k}$.

$$
\begin{aligned}
M_k &= \int_{-\infty}^{\infty}\frac{(1+t^2)}{(1+t^2)^{(k+1)}}dt \\
&= M_{k+1}+\int_{-\infty}^{\infty}\frac{2kt}{(1+t^2)^{k+1}}\frac{t}{2k}dt
\end{aligned}
$$

Since,

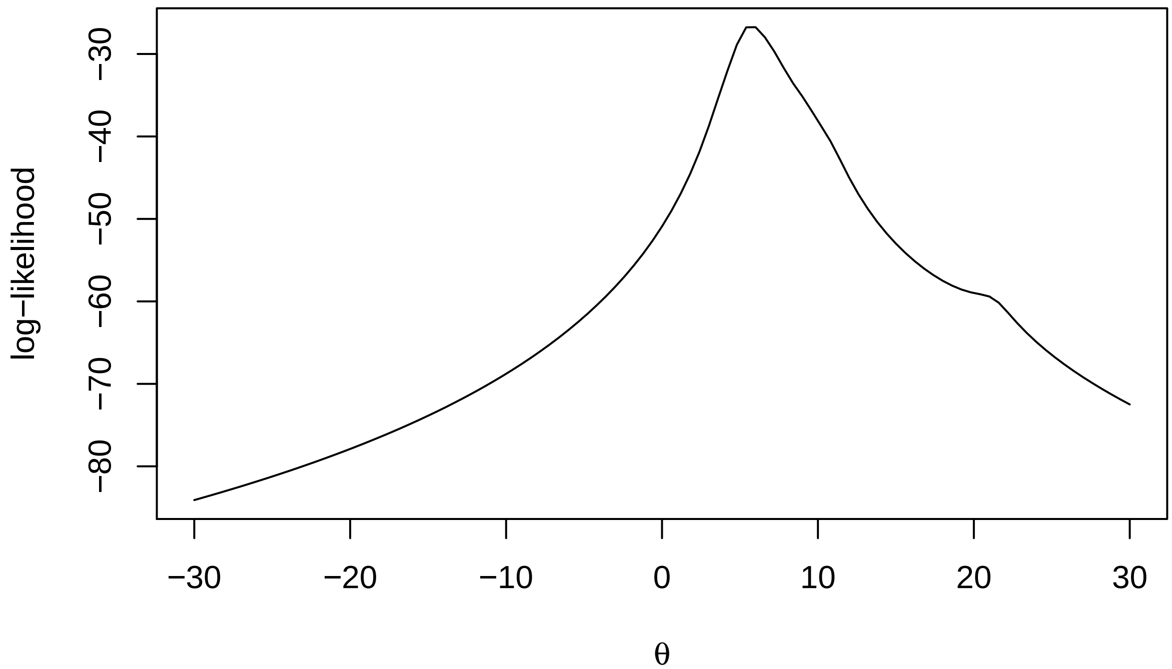
$$
\begin{aligned}
\int_{-\infty}^{\infty}\frac{2kt}{(1+t^2)^{k+1}}\frac{t}{2k}dt &= \left((-\frac{1}{(1+t^2)^k}\frac{t}{2k})\right)+\int_{-\infty}^{\infty}\frac{1}{(1+t^2)^k}\frac{1}{2k}dt \\
&= \int_{-\infty}^{\infty}\frac{1}{(1+t^2)^k}\frac{1}{2k}dt \\
&= \frac{1}{2k}M_k
\end{aligned}
$$

Since we know that $M_1=\int_{-\infty}^{\infty}\frac{1}{1+t^2}dt=\pi$, thus $M_1(\theta)=\frac{1}{2}$. Then $M_n(\theta)=\frac{n}{2}$. [1]

Q2

```r
set.seed(20180909)
n <- 10
theta_true <- 5
X <- rcauchy(n, location = theta_true, scale = 1)
# Implement the log-likelihood function
likelihood <- function(x){
  -n*log(pi) - sum(log(1+(X-x)^2))
}
hh <- Vectorize(likelihood)
curve(hh, -30, 30, ylab = "log-likelihood", xlab = expression(theta))
```

[1] This method is from https://stats.stackexchange.com/questions/145017/cauchy-distribution-likelihood-and-fisher-information



Q3

The algorithm for Newton-Raphson is:

$$\theta^{(t+1)} = \theta^{(t)} - (\ell''(\theta^{(t)}))^{-1}\ell'(\theta^{(t)})$$

```
Newton <- function(init){
  theta0 <- init
  i <- 0
  diff <- 1
  msg <- "converge"
  while(abs(diff) > 0.0000001){
    lfd <- -2*sum((theta0-X)/(1+(theta0-X)^2))
    lsd <- -2*sum((1-(theta0-X)^2)/(1+(theta0-X)^2)^2)
    diff <- (lfd/lsd)
    theta1 <- theta0 - diff
    theta0 <- theta1
    i <- i+1
    #cat(i)
    if(i >= 150){
      msg <- "Not converge"
      break
    }
  }
  return(list(theta = theta0, itr = i, msg = msg))
}
```

```
Newton_summary <- NULL
for(i in seq(-10, 20, 0.5)){
  result <- Newton(i)
```

```r
  if(result$msg == "converge"){
    Newton_summary <- rbind(Newton_summary, c(i, result$theta, result$itr))
  }
}
colnames(Newton_summary) <- c("start_point", "theta", "iteration")
Newton_summary
```

```
##      start_point    theta iteration
## [1,]         5.0 5.685422         6
## [2,]         5.5 5.685422         4
## [3,]         6.0 5.685422         5
## [4,]         6.5 5.685422         8
## [5,]         9.0 5.685422         7
## [6,]        10.5 5.685422         6
```

I omit not converge results to save space. We can see that Newton-Raphson result shows that a lot of start points result in not coverge.

Q4

Use fixed point method to estimate *theta*. The algorithm for it is:

$$\theta^{(t+1)} = \theta^t + \alpha\ell'(\theta)$$

```r
fixed_point <- function(init, alpha){
  theta0 <- init
  i <- 0
  diff <- 1
  msg <- "converge"
  alpha <- alpha
  while(abs(diff) > 0.0000001){
    lfd <- -2*sum((theta0-X)/(1+(theta0-X)^2))
    diff <- alpha*lfd
    theta1 <- theta0 + diff
    theta0 <- theta1
    i <- i+1
    if(i >= 150){
      msg <- "Not converge"
      break
    }
  }
  return(list(theta = theta0, itr = i, msg = msg))
}

fixed_point_summary <- NULL
for(alpha in c(1, 0.64, 0.25)){
  for(i in seq(-10, 20, 0.5)){
  result <- fixed_point(i, alpha)
  if(result$msg == "converge"){
    fixed_point_summary <- rbind(fixed_point_summary, c(alpha, i, result$theta, result$itr))
  }
}
```

```r
}

colnames(fixed_point_summary) <- c("alpha", "start_point", "theta", "iteration")
fixed_point_summary
```

```
##       alpha start_point    theta iteration
##  [1,]  0.25       -10.0 5.685422        45
##  [2,]  0.25        -9.5 5.685422        44
##  [3,]  0.25        -9.0 5.685422        42
##  [4,]  0.25        -8.5 5.685422        40
##  [5,]  0.25        -8.0 5.685422        39
##  [6,]  0.25        -7.5 5.685422        37
##  [7,]  0.25        -7.0 5.685422        36
##  [8,]  0.25        -6.5 5.685422        34
##  [9,]  0.25        -6.0 5.685422        32
## [10,]  0.25        -5.5 5.685422        32
## [11,]  0.25        -5.0 5.685422        28
## [12,]  0.25        -4.5 5.685422        29
## [13,]  0.25        -4.0 5.685422        27
## [14,]  0.25        -3.5 5.685422        26
## [15,]  0.25        -3.0 5.685422        25
## [16,]  0.25        -2.5 5.685422        24
## [17,]  0.25        -2.0 5.685422        23
## [18,]  0.25        -1.5 5.685422        23
## [19,]  0.25        -1.0 5.685422        22
## [20,]  0.25        -0.5 5.685422        21
## [21,]  0.25         0.0 5.685422        21
## [22,]  0.25         0.5 5.685422        19
## [23,]  0.25         1.0 5.685422        18
## [24,]  0.25         1.5 5.685422        19
## [25,]  0.25         2.0 5.685422        18
## [26,]  0.25         2.5 5.685422        18
## [27,]  0.25         3.0 5.685422        16
## [28,]  0.25         3.5 5.685422        16
## [29,]  0.25         4.0 5.685422        17
## [30,]  0.25         4.5 5.685422        16
## [31,]  0.25         5.0 5.685422        15
## [32,]  0.25         5.5 5.685422        16
## [33,]  0.25         6.0 5.685422        15
## [34,]  0.25         6.5 5.685422        16
## [35,]  0.25         7.0 5.685422        17
## [36,]  0.25         7.5 5.685422        17
## [37,]  0.25         8.0 5.685422        18
## [38,]  0.25         8.5 5.685422        19
## [39,]  0.25         9.0 5.685422        17
## [40,]  0.25         9.5 5.685422        20
## [41,]  0.25        10.0 5.685422        21
## [42,]  0.25        10.5 5.685422        20
## [43,]  0.25        11.0 5.685422        22
## [44,]  0.25        11.5 5.685422        19
## [45,]  0.25        12.0 5.685422        23
## [46,]  0.25        12.5 5.685422        24
## [47,]  0.25        13.0 5.685422        23
## [48,]  0.25        13.5 5.685422        25
```

```
## [49,]  0.25        14.0 5.685422        26
## [50,]  0.25        14.5 5.685422        27
## [51,]  0.25        15.0 5.685422        28
## [52,]  0.25        15.5 5.685422        29
## [53,]  0.25        16.0 5.685422        30
## [54,]  0.25        16.5 5.685422        31
## [55,]  0.25        17.0 5.685422        33
## [56,]  0.25        17.5 5.685422        35
## [57,]  0.25        18.0 5.685422        35
## [58,]  0.25        18.5 5.685422        38
## [59,]  0.25        19.0 5.685422        41
## [60,]  0.25        19.5 5.685422        44
## [61,]  0.25        20.0 5.685422        48
```

I omit not converge results to save space. We can see that when $alpha = 1, 0.64$. All starting points give us not converge result. But $alpha = 0.25$ works well.

Q5

The fisher information is $I_n(\theta) = \frac{n}{2}$.

```
Newton_fisher <- function(init){
  theta0 <- init
  i <- 0
  diff <- 1
  msg <- "converge"
  while(abs(diff) > 0.0000001){
    lfd <- -2*sum((theta0-X)/(1+(theta0-X)^2))
    I <- n/2
    diff <- (lfd/I)
    theta1 <- theta0 + diff
    theta0 <- theta1
    i <- i+1
    #cat(i)
    if(i >= 150){
      msg <- "Not converge"
      break
    }
  }
  return(list(theta = theta0, itr = i, msg = msg))
}

Newton_fisher_summary <- NULL
for(i in seq(-10, 20, 0.5)){
  result <- Newton_fisher(i)
  if(result$msg == "converge"){
    Newton_fisher_summary <- rbind(Newton_fisher_summary, c(i, result$theta, result$itr))
  }
}
colnames(Newton_fisher_summary) <- c("start_point", "theta", "iteration")
Newton_fisher_summary
```

```
##       start_point    theta iteration
##  [1,]       -10.0 5.685422        44
```

```
##  [2,]         -9.5 5.685422         42
##  [3,]         -9.0 5.685422         40
##  [4,]         -8.5 5.685422         38
##  [5,]         -8.0 5.685422         36
##  [6,]         -7.5 5.685422         34
##  [7,]         -7.0 5.685422         33
##  [8,]         -6.5 5.685422         30
##  [9,]         -6.0 5.685422         28
## [10,]         -5.5 5.685422         28
## [11,]         -5.0 5.685422         26
## [12,]         -4.5 5.685422         24
## [13,]         -4.0 5.685422         23
## [14,]         -3.5 5.685422         22
## [15,]         -3.0 5.685422         20
## [16,]         -2.5 5.685422         18
## [17,]         -2.0 5.685422         18
## [18,]         -1.5 5.685422         17
## [19,]         -1.0 5.685422         15
## [20,]         -0.5 5.685422         15
## [21,]          0.0 5.685422         14
## [22,]          0.5 5.685422         12
## [23,]          1.0 5.685422         11
## [24,]          1.5 5.685422         12
## [25,]          2.0 5.685422         11
## [26,]          2.5 5.685422         10
## [27,]          3.0 5.685422          9
## [28,]          3.5 5.685422          8
## [29,]          4.0 5.685422          9
## [30,]          4.5 5.685422          9
## [31,]          5.0 5.685422          8
## [32,]          5.5 5.685422          8
## [33,]          6.0 5.685422          8
## [34,]          6.5 5.685422          9
## [35,]          7.0 5.685422         10
## [36,]          7.5 5.685422         10
## [37,]          8.0 5.685422         10
## [38,]          8.5 5.685422         11
## [39,]          9.0 5.685422         13
## [40,]          9.5 5.685422         14
## [41,]         10.0 5.685422         15
## [42,]         10.5 5.685422         15
## [43,]         11.0 5.685422         14
## [44,]         11.5 5.685422         17
## [45,]         12.0 5.685422         17
## [46,]         12.5 5.685422         18
## [47,]         13.0 5.685422         18
## [48,]         13.5 5.685422         19
## [49,]         14.0 5.685422         20
## [50,]         14.5 5.685422         22
## [51,]         15.0 5.685422         24
## [52,]         15.5 5.685422         25
## [53,]         16.0 5.685422         26
## [54,]         16.5 5.685422         27
## [55,]         17.0 5.685422         30
```

```
## [56,]        17.5 5.685422        32
## [57,]        18.0 5.685422        33
## [58,]        18.5 5.685422        37
## [59,]        19.0 5.685422        40
## [60,]        19.5 5.685422        44
## [61,]        20.0 5.685422        48
```

I omit not converge results to save space. All points are coverged.

Q6

We can see that choosing starting point for Newton-Raphson is extremely important. However, when it is converged, the iteration of Newton-Raphson is relatively small than other two methods. The scaling paramter for fixed point method is significantly important. Newton's method adapted by fisher's information works well for this function.

Let's compare the speed of these three methods.

```r
set.seed(20180909)
system.time(replicate(10000, Newton(5)))
```

```
##    user  system elapsed
##   0.153   0.004   0.159
```

```r
system.time(replicate(10000, fixed_point(5, 0.25)))
```

```
##    user  system elapsed
##   0.191   0.007   0.198
```

```r
system.time(replicate(10000, Newton_fisher(5)))
```

```
##    user  system elapsed
##   0.109   0.001   0.112
```

We can see that Newton's method adapted by fisher's information is the fastest. While the fixed point method is the slowest. I think this because fixed point method requires more iteration to achieve convergence.