STAT 206 Homework 5_Lihua Xu

Due Monday, November 6, 5:00 PM

General instructions for homework: Homework must be completed as an R Markdown file. Be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

Part I - Optimization and standard errors

1. Using any optimization code you like, maximize the likelihood of the gamma distribution on the cats's hearts. Start the optimization at the estimate you get from the method of moments. (a) What command do you use to maximize the log-likelihood? Explain its arguments. (b) What is the estimate? (c) What is the log-likelihood there? The gradient?

```
##Following the instructions on "gradient.descent()" told in lecture Optimization I.
library(MASS)
library(numDeriv)
data(cats)

gamma.est <- function(data) {
  m <- mean(data)
  v <- var(data)
  scale <- v/m
  shape <- m^2/v
  return(c(shape=shape,scale=scale))}

gamma.loglike <- function(parameters) {
    return(sum(dgamma(cats$Hwt, shape=parameters[1], scale=parameters[2], log = TRUE)))}

#Optimization code on gradient.descent
gradient.descent <- function(f, x, max.iterations, step.scale,
  stopping.deriv,...) {
  for (iteration in 1:max.iterations) {
    gradient <- grad(f, x,...)
    if(all(abs(gradient) < stopping.deriv)) { break() }
    x <- x - step.scale*gradient}
  fit <- list(argmin = x, final.gradient=gradient, final.value=f(x,...),iterations=iteration)
  return(fit)}

max_loglike_estimation <- gradient.descent(f=gamma.loglike, x=gamma.est(cats$Hwt),
                                           max.iterations=1e4, step.scale=c(-1e-2,-1e-4),
                                           stopping.deriv=1e-5)
max_loglike_estimation
```

```
## $argmin
##      shape      scale
## 20.2995481  0.5236843
##
## $final.gradient
```

```
## [1]  3.869504e-06 -9.999272e-06
##
## $final.value
## [1] -325.5476
##
## $iterations
## [1] 6503
```

```
#Here I use tolerance (stopping.deriv) 1e-5,
#the step size for the step-scales for the shqpe and scale are -1e-2 and -1e-4 respectively.
#And here, I set the maximum iteration number is 10000.
#x refers to the initial value for shape and scale.
#Here we use the value from moment of method as the initail guess.
#f is the function regarding the gamma.loglike function.
#The estimate parameters for shape and scale are 20.2995481 and 0.5236843 seperately.
#The gadient are 3.869504e-06 and -9.999272e-06 respectively.
```

We need standard errors for the estimated parameters. If we believe the model is accurate, we can get standard errors by simulating from the fitted model, and re-estimating on the simulation output.

2. Write a function, `make.gamma.loglike`, which takes in a data vector `x` and returns a log-likelihood function.

```
make.gamma.loglike <- function(x) {
    gamma.loglike <- function(parameters) {
    return(sum(dgamma(x, shape = parameters[1], scale = parameters[2], log = TRUE)))}
    return(gamma.loglike)}
```

3. Write a function, `gamma.mle`, which takes in a data vector `x`, and returns a shape and a scale parameter, estimated by maximizing the log-likelihood of the gamma distribution. It should use your `make.gamma.loglike` function from the previous part. Check that if `x` is `cats$Hwt`, then `gamma.mle` matches the answer in problem 1.

```
gamma.mle <- function(x){
  temps <- gamma.est(x)
    loglike.temp <- make.gamma.loglike(x)
  mle.est<-gradient.descent(f=loglike.temp, x=temps, max.iterations=1e4,
        step.scale = c(-1e-2, -1e-4), stopping.deriv = 1e-4)$argmin
  return(mle.est)}
gamma.mle(cats$Hwt)
```

```
##      shape      scale
## 20.2993392  0.5236897
```

```
#`gamma.mle` matches the answer in problem 1.
```

4. Modify the code from homework 4 to use your `gamma.mle` function, rather than the method-of-moments estimator. In addition to giving the modified code, explain in words what you had to change, and why.

```
#Modifying the code for "gamma.est.sim"
gamma.est.sim <- function(B,x,n) {
    return(replicate(B, gamma.mle(rgamma(n, shape=20.2993392, scale = 0.5236897))))}
#I need to repleace all the estimations from method-of-moment to "gamma.mle" function
#because these change will help to change the method to what we want.
#The input "shape" and "scale"" should be the optimized parameters from "gamma.mle" method.
```

5. What standard errors do you get from running $10e4$ simulations?

```
#The standard error function
gamma.est.se <- function(B,x,n)
{shape <- sd(gamma.est.sim(B,x,n)[1,])
scale <- sd(gamma.est.sim(B,x,n)[2,])
return(c(standard_deviation_shape=shape,standard_deviation_scale=scale))}

#Running calculations:
#When I do 10e4 simulations, it take too long time, so here, I used 20 simulations instead.
#(gamma.est.se(1e4,cats$Hwt,length(cats$Hwt)))
gamma.est.se(20,cats$Hwt,length(cats$Hwt))
```

```
## standard_deviation_shape standard_deviation_scale
##               2.28689015               0.05771451
```

```
#The standard errors for shape and scale are shown above.
```

6. An alternative to using simulation is to use the jack-knife. Calculate jack-knife standard errors for the MLE of the gamma distribution. Your code should be able to work with an arbitrary data vector, not just `cats$Hwt`, and you will want to use functions from problems 1 and 2.

```
Gamma.jackmle <- function(x) {
    num <- length(x)
    jack_estimation <- matrix(0, num, ncol=2)
    for (i in 1:num) {
        temps <- gamma.est(x[-i])
        loglike.temp <- make.gamma.loglike(x[-i])
        jack_estimation[i,] <- gradient.descent(f=loglike.temp, x=temps, max.iterations=1e4,
                                                step.scale = c(-1e-2, -1e-4),
                                                stopping.deriv = 1e-5)$argmin}
    jack.v <- ((num-1)^2/num)*apply(jack_estimation, 2, var)
    jack.sde <- sqrt(jack.v)
    return(jack.sde)}
```

7. What are the jackknife standard errors for the MLE? (If you do not have two, one for the shape and one for the scale parameters, something is wrong.)

```
Gamma.jackmle(cats$Hwt)
```

```
## [1] 2.4561031 0.0644522
```

```
#The jackknife standard errors for the MLE are 2.4561031 and 0.0644522 respectively.
```

8. Do your jackknife standard errors for the MLE match those you got in problem 5? Should they?

```
#My jackknife standard errors for the MLE doesn't equal those I got in problem 5.
#But these values should within some range. In case of this, they should match.
```

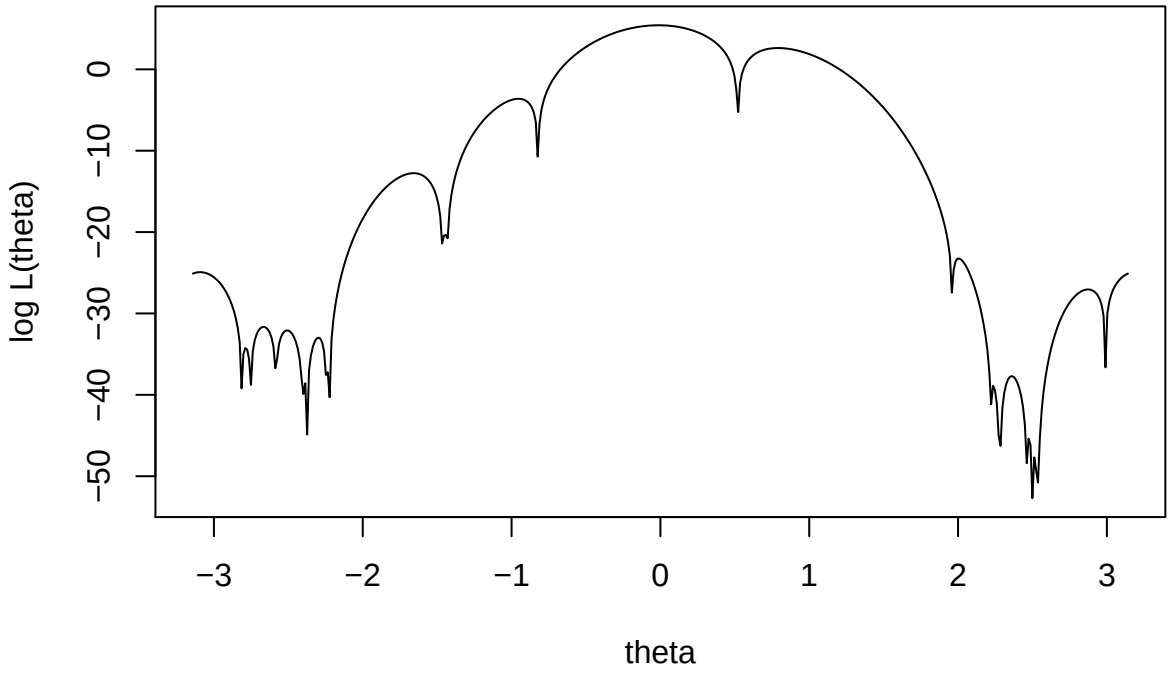
Part II - Newton's method

Consider the density $f(x) = [1 - \cos\{x - \theta\}]/2\pi$ on $0 \leq x \leq 2\pi$, where θ is a parameter between $-\pi$ and π. The following i.i.d. data arise from this density: 3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88, 2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.50. We wish to estimate θ.

9. Graph the log-likelihood function between $-\pi$ and π.

As the above is the density function, so the observations are iid according to f, the log-likelihood function should be $l(\theta) = \sum_{i=1}^{n} log(1 - cos(X_i - \theta))$, X_i is related to each element in the desity vector above.

```r
X <- c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96,
       2.53, 3.88, 2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.50)
loglik <- function(x) sum(log(1 - cos(X - x)))
A <- seq(-pi,pi, len=500)
loglikelihood.A <- sapply(A, loglik)
plot(A, loglikelihood.A, type="l", xlab="theta", ylab="log L(theta)")
```



10. Find the method of moments estimator of θ.

My solution: The mean function for is

$$\mu(\theta) = \frac{1}{2\pi}\int_0^{2\pi} x[1 - cos(x-\theta)]dx = \pi - \frac{1}{2\pi}\int_0^{2\pi} xcos(x-\theta)dx$$

As we know the integration below:

$$\int_0^{2\pi} xcos(x-\theta)dx = 2\pi sin(2\pi - \theta)$$

Then we could get:

$$\mu(\theta) = \pi - sin\theta$$

The method of moment estimator is found by setting the mean function $\mu(\theta)$ equal to the sample mean $\bar{X}$, so

$$\mu(\theta) = \bar{X}$$

Which means: $\pi - sin\theta = \bar{X}$, finally $\theta = arcsin(\pi - \bar{X})$

```r
mome <- asin(pi - mean(X))
mome
```

```
## [1] -0.05844061
```

```r
#So the the method of moments estimator is -0.05844061.
```

11. Find the MLE for θ using Newton's method, using the result from 10 as a starting value. What solutions do you find when you start at -2.7 and 2.7?

```r
#First and second derivitive of the function
loglik.firstD <- function(x) sum(sin(X - x) / (1 - cos(X - x)))
loglik.secondD <- function(x) sum(1 / (1 - cos(X - x)))
#The newton_method function
newton_method <- function(f.firstD, f.secondD, x0, tolerance=1e-06, iter.max=500) {
  x <- x0 #xo is the initial guess
  n <- 0 #n is used to count the number of iterations
  repeat {
    n <- n+1
    x_new <- x - f.firstD(x)/f.secondD(x)
    if(abs(x_new-x) < tolerance) {
      if(n >= iter.max) break}
    x <- x_new}
  solution <- list(final_optimized_value=x_new)
  return(solution)}
##start from result in 10, the estimator is -0.0119
newton_method(loglik.firstD, loglik.secondD, mome)
```

```
## $final_optimized_value
## [1] -0.011972
```

```r
##start at -2.7, the estimator is -2.6667
newton_method(loglik.firstD, loglik.secondD, -2.7)
```

```
## $final_optimized_value
## [1] -2.6667
```

```r
##start at 2.7,the estimator is 2.873095
newton_method(loglik.firstD, loglik.secondD, 2.7)
```

```
## $final_optimized_value
## [1] 2.873095
```

12. Repeat problem 11 using 200 equally spaced starting values between $-\pi$ and π. The partition the interval into sets of attraction. That is, divide the starting values into separate groups corresponding to the different local modes. Discuss your results.

```r
#between -pi to pi, we have 200 equal interval
B <- seq(-pi, pi, len=200)
#For the convenience of comparing the value, the values would be rounded to contain 3 digits
Value <- function(x_start)
  {round(newton_method(loglik.firstD, loglik.secondD, x_start)$final_optimized_value,3)}
theta_vector <- sapply(B,Value)
seperate_groups <- split(B,theta_vector)
#There are 19 different theta values as shown in "Global Environment" in Rstudio.
```

13. Find two starting values as close together as you can that converge to different solution using Newton's method.

```r
minimum <- sapply(seperate_groups, min)
maxnimum <- sapply(seperate_groups, max)
list <- cbind(minimum=round(minimum,5), maxnimum=round(maxnimum,5))
list
```

```
##         minimum maxnimum
## -3.093 -3.14159 -2.82585
## -2.786 -2.79428 -2.76271
## -2.667 -2.73113 -2.60484
## -2.508 -2.57326 -2.41540
## -2.388 -2.38382 -2.38382
## -2.297 -2.35225 -2.25753
## -2.232 -2.22595 -2.22595
## -1.658 -2.19438 -1.46818
## -1.447 -1.43661 -1.43661
## -0.953 -1.40503 -0.83671
## -0.012 -0.80513  0.48939
## 0.791   0.52097  1.94179
## 2.004   1.97336  2.19438
## 2.236   2.22595  2.25753
## 2.361   2.28910  2.44697
## 2.475   2.47854  2.47854
## 2.514   2.51012  2.51012
## 2.873   2.54169  2.98372
## 3.19    3.01530  3.14159
```

```r
value_difference <- c()
for (i in 1:18) {
  value_difference <- c(value_difference,list[i+1,2]-list[i,2])
}
value_difference
```

```
##  [1] 0.06314 0.15787 0.18944 0.03158 0.12629 0.03158 0.75777 0.03157
##  [9] 0.59990 1.32610 1.45240 0.25259 0.06315 0.18944 0.03157 0.03158
## [17] 0.47360 0.15787
```

```r
#minimum and maximum is related to the lower and upper bounds to the corresponding theta value.
#From which, we can found the two closest values:2.47854,2.44697
#2.47854-2.44697=0.03157 this is the smallest difference value.
#The difference values are shown in the vector value_difference.
```