STAT 206 Homework 5

Xin Feng(Vanessa)

11/3/2019

Due Monday, November 4, 5:00 PM

General instructions for homework: Homework must be submitted as pdf file, and 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

```
library(MASS)
data(cats)
# View(cats)
summary(cats)
```

```
##  Sex         Bwt             Hwt
##  F:47   Min.   :2.000   Min.   : 6.30
##  M:97   1st Qu.:2.300   1st Qu.: 8.95
##         Median :2.700   Median :10.10
##         Mean   :2.724   Mean   :10.63
##         3rd Qu.:3.025   3rd Qu.:12.12
##         Max.   :3.900   Max.   :20.50
```

1. Using any optimization code you like, maximize the likelihood of the gamma distribution on the cats' 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?

```
# a)
# Use `nlm` to minimize negative likelihood of gamma distribution
# i.e. maximize the likelihood of gamma distribution.
logsum_gamma <- function(p){
  ls_gamma <- -sum(dgamma(cats$Hwt, shape=p[1], scale=p[2], log=TRUE))
  return(ls_gamma)
}

# Get initial value of p(shape, scale)
gamma.est <- function(value_x){
  value_mean <- mean(value_x)
  value_var <- var(value_x)
  shape <- value_mean^2/value_var
  scale <- value_var/value_mean
  return(c(shape, scale))
}
```

```r
p <- gamma.est(cats$Hwt)
result <- nlm(logsum_gamma, p)
result
```

```
## $minimum
## [1] 325.5476
##
## $estimate
## [1] 20.2995529  0.5236842
##
## $gradient
## [1] -1.507798e-06 -5.777764e-05
##
## $code
## [1] 1
##
## $iterations
## [1] 7
```

```r
# b)
cat("Estimate is: shape =", result$estimate[1], ", scale =", result$estimate[2], "\n")
```

```
## Estimate is: shape = 20.29955 , scale = 0.5236842
```

```r
# c)
cat("Loglikelihood is: minimum =", result$minimum[1], "\n")
```

```
## Loglikelihood is: minimum = 325.5476
```

```r
cat("Gradient is: gradient =", result$gradient[1], result$gradient[2], "\n")
```

```
## Gradient is: gradient = -1.507798e-06 -5.777764e-05
```

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.

```r
p <- gamma.est(cats$Hwt)
make.gamma.loglike <- function(x){
  logsum_gamma <- function(p){
    ls_gamma <- -sum(dgamma(x, shape=p[1], scale=p[2], log=TRUE))
    return(ls_gamma)
  }
}
make.gamma.loglike(cats$Hwt)
```

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, p){
  result <- nlm(make.gamma.loglike(x), p)
  return(c(result$estimate[1], result$estimate[2]))
}
p <- c(19, 0.56)
gamma.mle(cats$Hwt, p)
```

```
## [1] 20.299929  0.523674
```

```
# The result 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.

```
gamma.est.sim.mod <- function(a, s, n, B, p){
  est_a <- c()
  est_s <- c()
  for(i in 1:B){
    value_gamma <- rgamma(n, shape=a, scale=s)
    est_gamma <- gamma.mle(value_gamma, p)
    est_a <- c(est_a, est_gamma[1])
    est_s <- c(est_s, est_gamma[2])
  }
  est_para <- rbind(est_a, est_s)
  return(est_para)
}

gamma.est.se.mod <- function(a, s, n, B, p){
  est_para <- gamma.est.sim.mod(a, s, n, B, p)
  sd_a <- sd(est_para[1,])
  sd_s <- sd(est_para[2,])
  return (cbind(sd_a=sd_a, sd_s=sd_s))
}
```

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

```
gamma.est.se.mod(a=2,s=1,n=10,B=10e4,p=c(2,1))
```

```
##          sd_a      sd_s
## [1,] 1.706875 0.4529812
```

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.

```
jk_mle <- function(x, p){
  est_a <- c()
  est_s <- c()
  for(i in 1:length(x)){
    gamma_para <- gamma.mle(x[-i], p)
```

```
    est_a <- c(est_a, gamma_para[1])
    est_s <- c(est_s, gamma_para[2])
  }
  est_a_sd <- sd(est_a)
  est_s_sd <- sd(est_s)
  return(data.frame(est_a_sd, est_s_sd))
}
```

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.)

```
jk_mle(rgamma(1000, shape=2, scale=1), c(2,1))
```

```
##      est_a_sd    est_s_sd
## 1 0.002447395 0.001400363
```

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

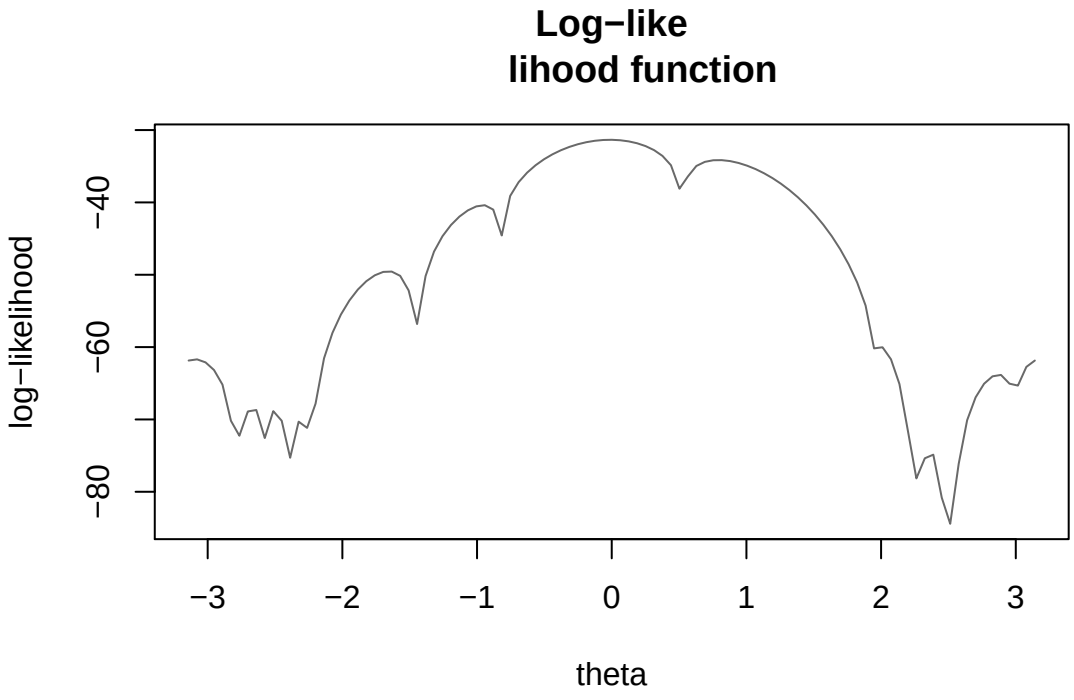
```
# Jackknife standard errors don't match the standard errors in problem 5.
# They should not equal. The jackknife standard errors should smaller than
# the one in problem 5.
```

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.

```
value_iid <- 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)
log.lh <- function(theta, data){
  sum(log(1-cos(data-theta)))-length(value_iid)*log(2*pi)
}
log.lh.gra <- function(theta, ...){
  return(sapply(theta, log.lh, ...))
}
curve(log.lh.gra(theta=x, data=value_iid), from=-pi, to=pi, main="Log-like
      lihood function", xlab="theta", ylab="log-likelihood", col="dimgrey")
```



10. Find the method of moments estimator of θ.

```
mean_iid <- mean(value_iid)
theta_hat <- asin(mean_iid-pi)
cat("Estimator of theta is:", theta_hat, "\n")
```

```
## Estimator of theta 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?

```
#  Newton's method is an iterative method for finding the roots of a
# differentiable function f, which are solutions to the equation f(x) = 0.
nt.log.lh <- function(data){
  log_lh <- function(theta){
    result <- sum(log(1-cos(data-theta)))-length(value_iid)*log(2*pi)
    return(result)
  }
  return(log_lh)
}
nt.log.lh(value_iid)
```

```
## function(theta){
##     result <- sum(log(1-cos(data-theta)))-length(value_iid)*log(2*pi)
##     return(result)
##   }
## <environment: 0x7fd72de86290>
```

```
# packageurl <- "https://cran.r-project.org/bin/macosx/el-capitan/contrib/
#             3.6/numDeriv_2016.8-1.1.tgz"
# install.packages(packageurl, repos=NULL, type="source")
library(numDeriv)
newton.func<- function(f,theta_0,e,B){
  theta<- c(theta_0)
  for (i in 2:B){
    data_deri <- genD(f,theta[i-1])
    para_p <- data_deri[[1]][1]
    para_h <- data_deri[[1]][2]
    theta_1 <- theta[i-1]-para_p/para_h
    theta <- c(theta,theta_1)
    if (abs(theta[i]-theta[i-1])< e)
    {break}
  }
  return(theta[i])
}
result_1 <- newton.func(nt.log.lh(value_iid),-2.7,1e-5,1e+3)
cat("theta starts at -2.7, result is", result_1, "\n")
```

```
## theta starts at -2.7, result is -2.6667
```

```
result_2 <-newton.func(nt.log.lh(value_iid),2.7,1e-5,1e+3)
cat("theta starts at 2.7, result is", result_2, "\n")
```

```
## theta starts at 2.7, result is 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.

```
value_rep<-seq(from=-pi,to=pi,by=2*pi/199)
estimate<- c()
for (i in 1:200){
estimate[i]<- newton.func(nt.log.lh(value_iid), value_rep[i], 1e-5, 1e+3) }
estimate
```

```
##   [1] -3.09309173 -3.09309173 -3.09309173 -3.09309173 -3.09309173
##   [6] -3.09309173 -3.09309173 -3.09309173 -3.09309173 -3.09309173
##  [11] -3.09309173 -2.78616675 -2.78616675 -2.66669993 -2.66669993
##  [16] -2.66669993 -2.66669993 -2.66669993 -2.50761323 -2.50761323
##  [21] -2.50761323 -2.50761323 -2.50761323 -2.50761323 -2.38820049
##  [26] -2.29725622 -2.29725622 -2.29725622 -2.29725622 -2.23216729
##  [31] -1.65828323 -1.65828323 -1.65828323 -1.65828323 -1.65828323
##  [36] -1.65828323 -1.65828323 -1.65828323 -1.65828323 -1.65828323
##  [41] -1.65828323 -1.65828323 -1.65828323 -1.65828323 -1.65828323
##  [46] -1.65828323 -1.65828323 -1.65828323 -1.65828323 -1.65828323
##  [51] -1.65828323 -1.65828323 -1.65828323 -1.65828323 -1.44747876
##  [56] -0.95333633 -0.95333633 -0.95333633 -0.95333633 -0.95333633
##  [61] -0.95333633 -0.95333633 -0.95333633 -0.95333633 -0.95333633
##  [66] -0.95333633 -0.95333633 -0.95333633 -0.95333633 -0.95333633
```

```
##  [71] -0.95333633 -0.95333633 -0.95333633 -0.95333633 -0.01197200
##  [76] -0.01197200 -0.01197200 -0.01197200 -0.01197200 -0.01197198
##  [81] -0.01197201 -0.01197201 -0.01197201 -0.01197199 -0.01197200
##  [86] -0.01197202 -0.01197200 -0.01197210 -0.01197200 -0.01197206
##  [91] -0.01197200 -0.01197200 -0.01197200 -0.01197196 -0.01197201
##  [96] -0.01197200 -0.01197200 -0.01197200 -0.01197199 -0.01197200
## [101] -0.01197200 -0.01197200 -0.01197203 -0.01197199 -0.01197200
## [106] -0.01197209 -0.01197201 -0.01197202 -0.01197200 -0.01197200
## [111] -0.01197187 -0.01197200 -0.01197204 -0.01197200 -0.01197199
## [116] -0.01197197  0.79060131  0.79060131  0.79060131  0.79060131
## [121]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [126]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [131]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [136]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [141]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [146]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [151]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [156]  0.79060131  0.79060131  0.79060131  0.79060131  0.79060131
## [161]  0.79060131  0.79060131  2.00364489  2.00364489  2.00364489
## [166]  2.00364489  2.00364489  2.00364489  2.00364489  2.00364489
## [171]  2.23621939  2.23621939  2.36071817  2.36071817  2.36071817
## [176]  2.36071817  2.36071817  2.36071817  2.47537363  2.51359318
## [181]  2.87309451  2.87309451  2.87309451  2.87309451  2.87309451
## [186]  2.87309451  2.87309451  2.87309451  2.87309451  2.87309451
## [191]  2.87309451  2.87309451  2.87309451  2.87309451  2.87309451
## [196]  3.19009358  3.19009358  3.19009358  3.19009358  3.19009358
```

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

```
newton.func(nt.log.lh(value_iid), -2.8, 1e-5, 1e+3)
```

```
## [1] -2.786167
```

```
newton.func(nt.log.lh(value_iid), -2.9, 1e-5, 1e+3)
```

```
## [1] -3.093092
```