PS2 rmd

Jared Yu

February 5, 2020

### Problem 1  
library(latex2exp)  
y\_sample\_data <- c(28, 33, 22, 35, 31) # Sample data  
y\_sample\_mean <- mean(y\_sample\_data) # Sample mean  
theta\_hat <- y\_sample\_mean # MLE of theta  
  
L2\_norm\_squared <- function(y, theta) { # Objective function  
 return(sum((y - theta)^2))  
}  
  
theta\_vector <- seq(theta\_hat - 10, theta\_hat + 10, length.out = 200)  
L2\_values <- sapply(theta\_vector, function(index)  
 L2\_norm\_squared(y = y\_sample\_data, theta = index))  
plot(theta\_vector, L2\_values, type = 'l', main = TeX('$s\_p(\\theta)$'),  
 ylab = 'L2 norm squared', xlab = TeX('$\\theta$'))  
  
L2\_norm\_squared\_derivative <- function(y, theta) {  
 n <- length(y)  
 return(-2 \* sum(y) + 2 \* n \* theta)  
}  
  
initialize\_variables <- function(a\_0, b\_0) {  
 x\_0 <- (a\_0 + b\_0) / 2  
 return(c(x\_0, a\_0, b\_0))  
}  
  
absolute\_convergence\_function <- function(x\_t, x\_update, epsilon) {  
 absolute\_convergence\_criterion <- abs(x\_update - x\_t)  
 bool\_flag <- ifelse(absolute\_convergence\_criterion < epsilon, FALSE, TRUE)  
 return(c(absolute\_convergence\_criterion, bool\_flag))  
}  
  
a\_b\_update <- function(a\_t, b\_t, x\_t) {  
 if(L2\_norm\_squared\_derivative(y = y, theta = a\_t) \*  
 L2\_norm\_squared\_derivative(y = y, theta = x\_t) <= 0) {  
 a\_update = a\_t; b\_update = x\_t  
 } else {  
 a\_update = x\_t; b\_update = b\_t  
 }  
 return(c(a\_update, b\_update))  
}  
  
x\_update\_function <- function(a\_update, b\_update) {  
 x\_update <- (a\_update + b\_update) / 2  
 return(x\_update)  
}  
  
bisection <- function(a, b) {  
 initial\_vector <- initialize\_variables(a\_0 = a, b\_0 = b) # Initialize variables  
 x\_t <- initial\_vector[1]; a\_t <- initial\_vector[2]; b\_t <- initial\_vector[3]  
 absolute\_convergence\_criterion <- TRUE  
 epsilon <- 0.01  
 counter <- 0  
   
 while (absolute\_convergence\_criterion) { # Loop through algorithm  
 a\_b\_update\_vector <- a\_b\_update(a\_t = a\_t, b\_t = b\_t, x\_t = x\_t)  
 a\_update <- a\_b\_update\_vector[1]; b\_update <- a\_b\_update\_vector[2]  
   
 x\_update <- x\_update\_function(a\_update = a\_update, b\_update = b\_update)  
   
 convergence\_vector <- absolute\_convergence\_function(x\_t = x\_t,  
 x\_update = x\_update, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 a\_t <- a\_update; b\_t <- b\_update; x\_t <- x\_update; counter <- counter + 1  
 }  
   
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('x\_update: ', x\_update, '\n')  
   
 theta <- seq(a, b, length.out = 200) # Zoom in to check  
 w <- sapply(theta, function(z) L2\_norm\_squared(y = y, theta = z))  
 plot(theta, w, main = TeX('$s\_p(\\theta)$'), type = 'l')  
 legend("topright", legend = c("theta = 30", "bisection update"),  
 col = c('blue', 'red'), lty = c(1,1))  
 abline(v = 30, col = 'blue')  
 abline(v = x\_update, col = 'red')  
}  
  
bisection(a = 25, b = 35) # 29.79492 (t=9)  
bisection(a = 28, b = 32) # 29.80469 (t=8)  
bisection(a = 29, b = 30) # 29.80469 (t=6)  
bisection(a = 29.6, b = 30) # 29.79375 (t=5)  
bisection(a = 29.7, b = 29.9) # 29.79375 (t=4)  
bisection(a = 29.75, b = 29.85) # 29.79375 (t=3)  
bisection(a = 29.78, b = 29.82) # 29.795 (t=2)  
bisection(a = 29.79, b = 29.81)  
bisection(a = 29.795, b = 29.805)  
bisection(a = 29.798, b = 29.802)  
# bisection(a = 29.79999, b = 29.80002) # 29.8 (t=1)  
bisection(a = 29.799999, b = 29.800001)  
bisection(a = 29.799, b = 29.801)  
  
### Problem 2  
problem\_2\_first\_derivative <- function(x) {  
 return(-x^3 + x - 1)  
}  
  
problem\_2\_second\_derivative <- function(x) {  
 return(-3 \* x^2 + 1)  
}  
  
h <- function(x) {  
 return(-problem\_2\_first\_derivative(x) / problem\_2\_second\_derivative(x))  
}  
  
nr <- function(x\_init, epsilon = 0.01) {  
 x\_new <- x\_init # Initialize variables  
 x\_old <- x\_init + 9999  
 counter <- 0  
 absolute\_convergence\_criterion <- TRUE  
   
 while(absolute\_convergence\_criterion) { # Loop through algorithm  
 x\_old <- x\_new  
 x\_new <- x\_old + h(x\_old)  
   
 convergence\_vector <- absolute\_convergence\_function(x\_t = x\_old,  
 x\_update = x\_new, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 counter <- counter + 1  
 }  
   
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('x\_update: ', x\_new, '\n')  
}  
  
nr(-1) # -1.324718 (t=4)  
nr(2) # -1.324732 (t=64)  
  
problem\_2\_function <- function(x) {  
 return(-(1/4) \* x^4 + (1/2) \* x^2 - x + 2)  
}  
  
xs <- seq(-1.35, -1.3, length.out = 200) # Plot problem 2 and check values  
plot(xs, problem\_2\_function(xs), type = 'l')  
abline(v = -1.324718)  
abline(v = -1.324732)  
  
### Problem 3  
### Newton-Raphson  
cauchy\_sample\_data <- c(1.77, -0.23, 2.76, 3.80, 3.47,  
 56.75, -1.34, 4.24, -2.44, 3.29,  
 3.71, -2.40, 4.53, -0.07, -1.05,  
 -13.87, -2.53, -1.75, 0.27, 43.21)  
  
log\_likelihood\_cauchy <- function(x, theta) {  
 n <- length(x)  
 return(-n \* log(pi) - sum(log(1 + (x - theta)^2)))  
}  
  
theta\_sequence <- seq(-50, 50, 0.01)  
log\_likelihood\_values\_cauchy <- sapply(theta\_sequence, function(index)  
 log\_likelihood\_cauchy(x = cauchy\_sample\_data, theta = index))  
plot(theta\_sequence, log\_likelihood\_values\_cauchy,  
 type = 'l', main = TeX('$Cauchy log likelihood$'),  
 ylab = 'log-likelihood theta', xlab = TeX('$\\theta$'))  
  
log\_likelihood\_cauchy2 <- function(x, theta) { # First derivative of log-likelihood  
 return(2 \* sum((x - theta) / (1 + (x - theta)^2)))  
}  
  
log\_likelihood\_cauchy3 <- function(x, theta) { # Second derivative of log-likelihood  
 squared\_difference <- (x - theta)^2  
 return(2 \* sum(((2 \* squared\_difference) /  
 (1 + squared\_difference)^2) - (1 / (1 + squared\_difference))))  
}  
  
h\_cauchy <- function(x, theta) { # h(t) function  
 return(-log\_likelihood\_cauchy2(x = x, theta = theta) /  
 log\_likelihood\_cauchy3(x = x, theta = theta))  
}  
  
nr <- function(x = cauchy\_sample\_data, theta\_init, epsilon = 0.01) {  
 theta\_new <- theta\_init # Initialize variables  
 theta\_old <- theta\_init + 9999  
 counter <- 0  
 absolute\_convergence\_criterion <- TRUE  
   
 while(absolute\_convergence\_criterion) { # Loop through algorithm  
 theta\_old <- theta\_new  
 theta\_new <- theta\_old + h\_cauchy(x = x, theta = theta\_old)  
   
 convergence\_vector <- absolute\_convergence\_function(x\_t = theta\_old,  
 x\_update = theta\_new, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 counter <- counter + 1  
 }  
  
 cat('theta\_init =', theta\_init, '\n')  
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('theta\_update: ', theta\_new, '\n')  
 return(theta\_new)  
}  
starting\_points <- c(-11, -1, 0, 1.5, 4,  
 4.7, 7, 8, 38)  
theta\_approximate <- sapply(starting\_points, function(index) nr(theta\_init = index))  
  
theta\_sequence <- seq(-0.19231, -0.19226, length.out = 200)  
log\_likelihood\_values <- sapply(theta\_sequence, function(index)  
 log\_likelihood\_cauchy(x = cauchy\_sample\_data, theta = index))  
plot(theta\_sequence, log\_likelihood\_values,  
 type = 'l', main = 'Cauchy log likelihood',  
 ylab = 'log-likelihood theta', xlab = TeX('$\\theta$'))  
abline(v = theta\_approximate[1]) # -0.1922825  
abline(v = theta\_approximate[2]) # -0.1922865 (closer)  
abline(v = theta\_approximate[3]) # -0.1922825  
abline(v = theta\_approximate[6]) # -0.1922865 (closer)  
abline(v = theta\_approximate[8]) # -0.1922825  
# -1, 4.7 closest  
  
### bisection method  
a\_b\_update\_cauchy <- function(a\_t, b\_t, x\_t) {  
 if(log\_likelihood\_cauchy2(x = cauchy\_sample\_data, theta = a\_t) \*  
 log\_likelihood\_cauchy2(x = cauchy\_sample\_data, theta = x\_t) <= 0) {  
 a\_update = a\_t; b\_update = x\_t  
 } else {  
 a\_update = x\_t; b\_update = b\_t  
 }  
 return(c(a\_update, b\_update))  
}  
  
bisection\_cauchy <- function(a, b) {  
 initial\_vector <- initialize\_variables(a\_0 = a, b\_0 = b) # Initialize variables  
 x\_t <- initial\_vector[1]; a\_t <- initial\_vector[2]; b\_t <- initial\_vector[3]  
 absolute\_convergence\_criterion <- TRUE  
 epsilon <- 0.01  
 counter <- 0  
   
 while (absolute\_convergence\_criterion) { # Loop through algorithm  
 a\_b\_update\_vector <- a\_b\_update\_cauchy(a\_t = a\_t, b\_t = b\_t, x\_t = x\_t)  
 a\_update <- a\_b\_update\_vector[1]; b\_update <- a\_b\_update\_vector[2]  
   
 x\_update <- x\_update\_function(a\_update = a\_update, b\_update = b\_update)  
   
 convergence\_vector <- absolute\_convergence\_function(x\_t = x\_t,  
 x\_update = x\_update, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 a\_t <- a\_update; b\_t <- b\_update; x\_t <- x\_update; counter <- counter + 1  
 }  
   
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('x\_update: ', x\_update, '\n')  
   
 theta <- seq(a, b, length.out = 200) # Zoom in to check  
 w <- sapply(theta, function(y)  
 log\_likelihood\_cauchy(x = cauchy\_sample\_data, theta = y))  
 plot(theta, w, main = "log likelihood function", type = 'l')  
 legend("bottomleft", legend = c("theta = 0", "bisection update"),  
 col = c('blue', 'red'), lty = c(1,1))  
 abline(v = 0, col = 'blue')  
 abline(v = x\_update, col = 'red')  
}  
  
bisection\_cauchy(a = -1, b = 1) # -0.1953125 (t=7)  
bisection\_cauchy(a = -5, b = 5) # -0.1855469 (t=9)  
bisection\_cauchy(a = -10, b = 10) # -0.1855469 (t=10)  
bisection\_cauchy(a = -20, b = 40) # 2.814941 (t=12)  
  
### fixed-point iterations  
fixed\_point <- function(theta\_init, alpha) {  
 theta\_current <- theta\_init # Initialize variables  
 absolute\_convergence\_criterion <- TRUE  
 epsilon <- 0.01  
 counter <- 0  
   
 while (absolute\_convergence\_criterion) { # Loop through algorithm  
 theta\_update <- theta\_current + alpha \*  
 log\_likelihood\_cauchy2(x = cauchy\_sample\_data, theta = theta\_current)  
  
 convergence\_vector <- absolute\_convergence\_function(x\_t = theta\_current,  
 x\_update = theta\_update, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 theta\_current <- theta\_update; counter <- counter + 1  
 }  
   
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('theta\_update: ', theta\_update, '\n')  
   
 return(theta\_update)  
}  
alpha\_vector <- c(1, 0.64, 0.25)  
sapply(alpha\_vector, function(index) fixed\_point(theta\_init = -1, alpha = index))  
# -0.1977772 (t=2065, alpha=1)  
# -0.1971533 (t=114, alpha=0.64)  
# -0.1937823 (t=5, alpha=0.25)  
  
### secant method  
secant\_update <- function(theta\_current, theta\_previous) {  
 theta\_update <- theta\_current -  
 log\_likelihood\_cauchy2(cauchy\_sample\_data, theta\_current) \*  
 ((theta\_current - theta\_previous) /  
 (log\_likelihood\_cauchy2(cauchy\_sample\_data, theta\_current) -  
 log\_likelihood\_cauchy2(cauchy\_sample\_data, theta\_previous)))  
 return(theta\_update)  
}  
secant\_method <- function(theta\_init, theta\_1) {  
 theta\_previous <- theta\_init # Initialize variables  
 theta\_current <- theta\_1  
 absolute\_convergence\_criterion <- TRUE  
 epsilon <- 0.01  
 counter <- 0  
   
 while (absolute\_convergence\_criterion) { # Loop through algorithm  
 theta\_update <- secant\_update(theta\_current, theta\_previous)  
   
 convergence\_vector <- absolute\_convergence\_function(x\_t = theta\_current,  
 x\_update = theta\_update, epsilon = epsilon)  
 absolute\_convergence\_value <- convergence\_vector[1]  
 absolute\_convergence\_criterion <- convergence\_vector[2]  
   
 theta\_previous <- theta\_current; theta\_current <- theta\_update; counter <- counter + 1  
 }  
   
 cat('iteration t =', counter, '\n') # Print results  
 cat('absolute convergence criterion: ', absolute\_convergence\_value, '\n')  
 cat('theta\_update: ', theta\_update, '\n')  
   
 return(theta\_update)  
}  
  
secant\_method(theta\_init = -2, theta\_1 = -1) # -0.1923655 (t=4)  
secant\_method(theta\_init = -3, theta\_1 = 3) # 2.817013 (t=4)