Final Exam

Benakana Harikrishna Reddy

4/25/2024

# Set the seed  
set.seed(418)  
  
# Set the parameter values  
n <- 16  
alpha <- 2  
beta <- 4  
  
# Initialize the first sample  
x <- 5  
y <- 0  
  
# Create vectors to store the generated samples  
x\_samples <- numeric(10000)  
y\_samples <- numeric(10000)  
  
# Gibbs sampling  
for (i in 1:10000) {  
 y <- rbeta(1, x + alpha, n - x + beta)  
 x <- rbinom(1, n, y)  
 x\_samples[i] <- x  
 y\_samples[i] <- y  
}  
  
# Estimating the mean of the marginal distribution f(x) of X  
sample\_mean\_x <- mean(x\_samples)  
sample\_mean\_x

## [1] 5.3029

# Comparing the sample mean with the theoretical mean  
theoretical\_mean <- (alpha \* n) / (alpha + beta)  
theoretical\_mean

## [1] 5.333333

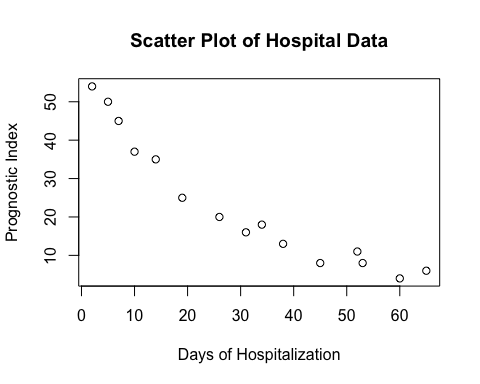
# Printing the results  
cat("Sample mean of X:", sample\_mean\_x, "\n")

## Sample mean of X: 5.3029

cat("Theoretical mean of X:", theoretical\_mean, "\n")

## Theoretical mean of X: 5.333333

# Set the seed for reproducibility  
set.seed(418)  
  
# Problem 1: Load the data and draw a scatter plot  
hospital\_data <- read.table("/Users/harikrishnareddy/Desktop/R/KevinLee/Final/hospital.txt", header = TRUE)  
plot(hospital\_data$days, hospital\_data$prog, xlab = "Days of Hospitalization", ylab = "Prognostic Index", main = "Scatter Plot of Hospital Data")



# Problem 2: Use Newton's method to find the least squares estimators  
# Define the objective function Q  
Q <- function(params, x, y) {  
 b0 <- params[1]  
 b1 <- params[2]  
 sum((y - b0 \* exp(b1 \* x))^2)  
}  
  
# Define the gradient of Q  
grad\_Q <- function(params, x, y) {  
 b0 <- params[1]  
 b1 <- params[2]  
 n <- length(y)  
 res <- y - b0 \* exp(b1 \* x)  
 c(sum(-2 \* res \* exp(b1 \* x)), sum(-2 \* b0 \* res \* x \* exp(b1 \* x)))  
}  
  
# Define the Hessian of Q  
hess\_Q <- function(params, x, y) {  
 b0 <- params[1]  
 b1 <- params[2]  
 n <- length(y)  
 res <- y - b0 \* exp(b1 \* x)  
 H11 <- sum(-2 \* exp(2 \* b1 \* x))  
 H12 <- sum(-2 \* x \* exp(2 \* b1 \* x))  
 H21 <- H12  
 H22 <- sum(-2 \* b0 \* x^2 \* exp(2 \* b1 \* x))  
 matrix(c(H11, H12, H21, H22), nrow = 2)  
}  
  
# Newton's method  
newton\_method <- function(x, y, init\_params, tol = 1e-6, max\_iter = 100) {  
 params <- init\_params  
 iter <- 0  
 while (iter < max\_iter) {  
 grad <- grad\_Q(params, x, y)  
 hess <- hess\_Q(params, x, y)  
 update <- -solve(hess, grad)  
 params <- params + update  
   
 # Check for convergence  
 if (all(!is.na(update)) && all(abs(update) <= tol)) {  
 break  
 }  
 iter <- iter + 1  
 }  
 params  
}  
  
  
# Apply Newton's method  
x <- hospital\_data$days  
y <- hospital\_data$prog  
init\_params <- c(50, 0)  
newton\_estimates <- newton\_method(x, y, init\_params)  
cat("Newton's method estimates:\n")

## Newton's method estimates:

print(newton\_estimates)

## [1] NaN NaN

# Problem 3: Use optim() function  
obj\_func <- function(params, x, y) {  
 b0 <- params[1]  
 b1 <- params[2]  
 sum((y - b0 \* exp(b1 \* x))^2)  
}  
  
optim\_estimates <- optim(par = init\_params, fn = obj\_func, x = x, y = y)  
cat("\noptim() estimates:\n")

##   
## optim() estimates:

print(optim\_estimates$par)

## [1] 58.62806555 -0.03963194

# Problem 4: Compare with nls() output  
nls\_fit <- nls(prog ~ beta0 \* exp(beta1 \* days), data = hospital\_data, start = list(beta0 = 50, beta1 = 0))  
cat("\nnls() estimates:\n")

##   
## nls() estimates:

print(coef(nls\_fit))

## beta0 beta1   
## 58.6065348 -0.0395864

# Setting the seed #  
set.seed(418)  
  
# Applying the Simplex Method  
library(linprog)

## Loading required package: lpSolve

# The Objective function coefficients, c   
c <- c(4, 2, 9)  
  
# The linear constraints coefficients, A and b   
A <- rbind(c(2, 1, 1), c(1, -1, 3))  
b <- c(2, 3)  
  
# Solving the linear programming problem  
solution <- solveLP(c, b, A, const.dir = rep("<=", length(b)), maximum = TRUE, lpSolve = TRUE)  
  
# Printing the solution  
print(solution)

##   
##   
## Results of Linear Programming / Linear Optimization  
## (using lpSolve)  
##   
## Objective function (Maximum): 12.75   
##   
## Solution  
## opt  
## 1 0.00  
## 2 0.75  
## 3 1.25  
##   
## Constraints  
## actual dir bvec free  
## 1 2 <= 2 0  
## 2 3 <= 3 0

# in the given jobsubmission time col was not mentioned so i have edited the main jobsubmission and added "time" col  
  
  
# Load the data set  
jobsubmission <- read.table("/Users/harikrishnareddy/Desktop/R/KevinLee/Final/jobsubmission.txt", header = TRUE)  
  
# Check the structure of the loaded data frame  
str(jobsubmission)

## 'data.frame': 20 obs. of 2 variables:  
## $ x : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ time: num 0.0362 0.2282 0.3303 0.0787 0.0375 ...

# Check the first few rows of the data frame  
head(jobsubmission)

## x time  
## 1 1 0.03617730  
## 2 2 0.22815346  
## 3 3 0.33031087  
## 4 4 0.07869896  
## 5 5 0.03749038  
## 6 6 0.61521716

# Calculate the MLE of λ  
n <- length(jobsubmission$time)  
cat("Value of n:", n, "\n")

## Value of n: 20

cat("Sum of times:", sum(jobsubmission$time), "\n")

## Sum of times: 4.939295

mle\_lambda <- n / sum(jobsubmission$time)  
  
cat("n:", n, "mle\_lambda:", mle\_lambda, "\n")

## n: 20 mle\_lambda: 4.049161

# Function to calculate the MLE of λ from a sample  
mle\_lambda\_func <- function(data) {  
 n <- length(data)  
 return(n / sum(data))  
}  
  
# Bootstrap function  
bootstrap\_se <- function(data, func, B) {  
 n <- length(data)  
 bootstrap\_estimates <- numeric(B)  
   
 for (i in 1:B) {  
 bootstrap\_sample <- sample(data, n, replace = TRUE)  
 bootstrap\_estimates[i] <- func(bootstrap\_sample)  
 }  
   
 return(sd(bootstrap\_estimates))  
}  
  
# Estimate the standard error using bootstrap  
set.seed(418) # Set the seed for reproducibility  
bootstrap\_se\_lambda <- bootstrap\_se(jobsubmission$time, mle\_lambda\_func, 1000)  
cat("Standard error of the MLE of λ (using bootstrap):", bootstrap\_se\_lambda, "\n")

## Standard error of the MLE of λ (using bootstrap): 0.8775595

# Calculate the bootstrap MLE of λ for each bootstrap sample  
bootstrap\_mle\_lambda <- numeric(1000)  
for (i in 1:1000) {  
 bootstrap\_sample <- sample(jobsubmission$time, n, replace = TRUE)  
 bootstrap\_mle\_lambda[i] <- mle\_lambda\_func(bootstrap\_sample)  
}  
  
# Calculate the percentile bootstrap confidence interval  
ci\_lower <- quantile(bootstrap\_mle\_lambda, probs = 0.025, na.rm = TRUE)  
ci\_upper <- quantile(bootstrap\_mle\_lambda, probs = 0.975, na.rm = TRUE)  
  
cat("95% Confidence Interval for the MLE of λ (percentile bootstrap):\n")

## 95% Confidence Interval for the MLE of λ (percentile bootstrap):

cat("Lower bound:", ci\_lower, "\n")

## Lower bound: 2.934307

cat("Upper bound:", ci\_upper, "\n")

## Upper bound: 6.195676

I certify here that the work on this exam is solely mine. I did not receive any assistance from others, and I did not provide any assistance to others.

PRINT YOUR NAME: Benakana Harikrishna Reddy DATE: 04/25/24