PS1

Boyu, Chen R11323006

2023-03-13

Question 1

Data generate process

```
b1 <- 0.5; b2 <- -0.5; n =400
x1 <- rnorm(n, 0, 1)
x2 <- rchisq(n, 1)
u1 <- rgumbel(n)
u2 <- rgumbel(n)
y <- as.numeric((b1*x1 + u1) > (b2*x2 + u2))
```

Log-likelihood function

$$
\begin{aligned}
&lnL(\beta_1, \beta_2; X_{1i}, X_{2i}) \\
&= \sum_i [y_i(\beta_1 X_{1i} - \beta_2 X_{2i}) - ln(1 + exp(\beta_1 X_{1i} - \beta_2 X_{2i})) - (1 - y_i) ln(1 + exp(\beta_1 X_{1i} - \beta_2 X_{2i}))] \\
&= \sum_i [y_i(\beta_1 X_{1i} - \beta_2 X_{2i}) - \frac{exp(\beta_1 X_{1i} - \beta_2 X_{2i})}{1 + exp(\beta_1 X_{1i} - \beta_2 X_{2i})} X_{1i}]
\end{aligned}
$$

```
loglik <- function(beta1, beta2, x1, x2){
    index <- beta1*x1 - beta2 * x2
    ll <- sum(y*(index) - log(1+exp(index)))
    return(-ll)
}
```

And we try to minimize the negative loglikelihood function, so 'loglik' return negative value.

Grid search

```
beta1_grid <- seq(from = -5, to = 5, by = 0.1)
beta2_grid <- seq(from = -5, to = 5, by = 0.1)
max_lik <- 10000000
argmax_beta <- c(0,0)
for (i in beta1_grid){
    for(j in beta2_grid){
```

```
        temp <- loglik(i,j, x1, x2)
        if (temp < max_lik){
            # try to minimize negative log-likelihood fn.
            max_lik <- temp
            argmax_beta <- c(i,j)
        }
    }
}
argmax_beta
```

```
## [1]  0.5 -0.5
```

Gradient method(BHHH method)

The gradient of the log-likelihood function is

$$\frac{\partial lnL_i}{\partial \beta} = \begin{bmatrix} X_{1i}(y_i - p_i) \\ X_{2i}(p_i - y_i) \end{bmatrix}$$

Where $p_i = \frac{exp(\beta_1 X_{1i} - \beta_2 X_{2i})}{1+exp(\beta_1 X_{1i} - \beta_2 X_{2i})}$

The H_{bhhh} matrix is

$$H_{bhhh} = \sum_i \frac{\partial lnL_i}{\partial \beta} \frac{\partial lnL_i}{\partial \beta'} = \begin{bmatrix} \sum_i X_{1i}^2 (y_i - p_i)^2 & \sum_i -X_{1i}X_{2i}(y_i - p_i)^2 \\ \sum_i -X_{1i}X_{2i}(y_i - p_i)^2 & \sum_i X_{2i}^2 (p_i - y_i)^2 \end{bmatrix}$$

The code is

```
##### BHHH #####
## R=100, N=400
R=100;N=400
X1 <- matrix(rnorm(R*N), nrow=N)
X2 <- matrix(rchisq(R*N, 1), nrow=N)
U1 <- matrix(rgumbel(R*N), nrow=N)
U2 <- matrix(rgumbel(R*N), nrow=N)
Y <- as.matrix((b1*X1 + U1) > (b2*X2 + U2)) %>% ifelse(1,0)

logistic <- function(x) {
    exp(x)/(1 + exp(x))
}

gradient <- function(beta1, beta2, X1, X2, Y) {
    p <- logistic(beta1*X1 - beta2*X2)
    g1 <- sum((Y-p)*X1)
    g2 <- sum((-Y+p)*X2)
    return(matrix(c(g1, g2), ncol=1))
}

BHHH <- function(beta1, beta2, X1, X2, Y) {
```

```
    p <- logistic(beta1*X1 - beta2*X2)
    b11 <- sum(X1^2 * (Y-p)^2)
    b12 <- sum(X1*X2*(Y-p)*(p-Y))
    b22 <- sum(X2^2 * (p-Y)^2)
    return(matrix(c(b11, b12, b12, b22),nrow=2, ncol=2))
}

beta_hat <- matrix(0, nrow=R, ncol=2)
for (i in 1:R){
    beta <- c(0, 0)
    tol <- 1e-4
    maxiter <- 1000
    for (j in 1:maxiter) {
        g <- gradient(beta[1], beta[2], X1[,i], X2[,i], Y[,i])
        bhhh <- BHHH(beta[1], beta[2], X1[,i], X2[,i], Y[,i])
        if (max(abs(g)) < tol) {
            break
        }
        beta <- beta + solve(bhhh) %*% g
    }
    beta_hat[i, 1] <- beta[1]
    beta_hat[i, 2] <- beta[2]
}
cat('The mean of beta1_hat is ', mean(beta_hat[,1]), '\n',
    'The mean of beta2_hat is', mean(beta_hat[,2]))
```

```
## The mean of beta1_hat is  0.5086533
##  The mean of beta2_hat is -0.5172834
```

```
cat('The standard error of beta1_hat is', sd(beta_hat[,1]), '\n',
    'The standard error of beta2_hat is', sd(beta_hat[,2]))
```

```
## The standard error of beta1_hat is 0.1122697
##  The standard error of beta2_hat is 0.1003795
```

Question 2

2-1

```
library(dplyr)
df <- readxl::read_xlsx('cps09mar.xlsx')
df$married <- ifelse(df$marital %in% c(1, 2, 3), 1, 0)
blk_wm_midwest <- df %>%
    filter(race==2, region == 2, female == 1)
blk_wm_midwest_logit <- glm(married ~ age + I(age^2) + education,
                            family = binomial, data = blk_wm_midwest)


summary(blk_wm_midwest_logit)$coefficients[,1:2] ## coef. and std. error
```

```
##                  Estimate   Std. Error
## (Intercept) -7.960369526 1.7670610496
## age          0.240087123 0.0772294083
## I(age^2)    -0.002503876 0.0008895825
## education    0.146292661 0.0459414202
```

2-2

```r
# setting sample size and num of repetition
n <- nrow(blk_wm_midwest)
B <- 1000

# Bootstrap standard errors saver
se_boot_vec <- matrix(NA, nrow=B, ncol=4)

# Bootstrap
for (i in 1:B) {
    sample_idx <- sample(1:n, size = n, replace = TRUE)
    sample_data <- blk_wm_midwest[sample_idx, ]
    fit <- glm(married ~ age + I(age^2) + education,
               data = sample_data, family = binomial())

    for(j in 1:4){
        se_boot_vec[i,j] <- summary(fit)$coefficients[j, 2]
    }
}

# calculate Bootstrap standard error
se_boot <- apply(se_boot_vec, 2, sd)

# print Bootstrap standard error
cat(' Intercept',se_boot[1], '\n',
    'age       ', se_boot[2], '\n',
    "I(age^2) ", se_boot[3], '\n',
    "education", se_boot[4])
```

```
##  Intercept 0.176652
##  age       0.008301609
##  I(age^2)  0.0001016893
##  education 0.002142154
```

2-3

The Delta method in multivariate case is

$$\sqrt{n}(g(\hat{\theta}) - g(\theta_0)) \xrightarrow{d} N(\partial g^T \Sigma \partial g)$$

where ∂g is the gradient column vector of g function, Σ is the asymptotic variance-covariance matrix of $\hat{\theta}$. In this case, $g = \frac{-\beta_1}{2\beta_2}$ and $\partial g = (\frac{-1}{2\beta_2} \, \frac{\beta_1}{2\beta_2^2})^T$

```r
#  - (b1) / (2*b2)
beta_hat <- coef(blk_wm_midwest_logit)
result <- unname(- beta_hat[2] / (2 * beta_hat[3]))

# partial derivitives
d1 <- -1 / (2 * beta_hat[3])
d2 <- beta_hat[2] / (2 * beta_hat[3]^2)
grad <- matrix(c(d1, d2),ncol=1)

# standard error
vcov_matrix <- vcov(blk_wm_midwest_logit)
delta_se <- t(grad) %*% vcov_matrix[2:3, 2:3] %*% grad

# result
sqrt(delta_se)
```

```
##          [,1]
## [1,] 2.668302
```

2-4

```r
# num of repetition
B <- 1000

# Bootstrap estimations saver
t_boot <- numeric(B)

# Bootstrap
for (i in 1:B) {
    sample_data_boot <- blk_wm_midwest[sample(nrow(blk_wm_midwest),
                                              replace = TRUE), ]
    fit_boot <- glm(married ~ age + I(age^2) + education,
                    data = sample_data_boot, family = binomial())
    coef_boot <- coef(fit_boot)
    t_boot[i] <- -coef_boot[2] / (2 * coef_boot[3])
}

se_boots <- sd(t_boot)

cat("Bootstrap standard error is", se_boots, "\n")
```

```
## Bootstrap standard error is 7.909103
```