Final Exam

2020324011
민동은

1. $h(x) = \{\cos(50x) + \sin(20x)\}^2$, $0 < x < 1$.

$\{f(g(x))\}' = f'(g(x)) \cdot g'(x)$. $\rightarrow h'(x) = 2\{\cos(50x) + \sin(20x)\} \cdot \{-50\sin(50x) + 20\cos(20x)\}$

$h''(x) = 2\left[\{-50\sin(50x) + 20\cos(20x)\}^2 + \{\cos(50x) + \sin(20x)\} \cdot \{-2500\cos(50x) - 400\sin(20x)\}\right]$

2. (a) Linear mixed effect model : $Y_i = X_i\beta + Z_i u_i + \varepsilon_i$

$u_i \sim N(0, D)$, $\varepsilon_i \sim N(0, \sigma^2 I_{n_i})$ $\rightarrow$ estimate $\theta = (\beta, D, \sigma^2)$

$Y_i \sim N(X_i\beta, Z_i D Z_i' + \sigma^2 I_{n_i})$. Let $\Sigma_i = Z_i D Z_i' + \sigma^2 I_{n_i}$.

Observed data log-likelihood : $\ell(\theta \mid Y_1, \cdots, Y_n) = \sum_{i=1}^{n}\left\{-\frac{1}{2}(Y_i - X_i\beta)'\Sigma_i^{-1}(Y_i - X_i\beta) - \frac{1}{2}\log|\Sigma_i| - \frac{n_i}{2}\log(2\pi)\right\}$

Given (D, σ^2), we know Σ_i, so we can obtain the β that maximizes the likelihood.

$$\therefore \frac{\partial \ell(\beta, D, \sigma^2 \mid Y_1, \cdots, Y_n)}{\partial \beta} = \sum_{i=1}^{n} X_i'\Sigma_i^{-1}(Y_i - X_i\beta) = 0.$$

$$\therefore \hat{\beta} = \left(\sum_{i=1}^{n} X_i'\Sigma_i^{-1}X_i\right)^{-1}\sum_{i=1}^{n} X_i'\Sigma_i^{-1}Y_i \rightarrow \text{need } D, \sigma^2 \text{ to maximize.}$$

Complete log-likelihood : $\begin{pmatrix} u_i \\ \varepsilon_i \end{pmatrix} \sim MVN\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} D & 0 \\ 0 & \sigma^2 I_{n_i} \end{pmatrix}\right)$.

$$\ell(\theta \mid \varepsilon_1, \cdots, \varepsilon_n, u_1, \cdots, u_n) = \sum_{i=1}^{n}\left(-\frac{1}{2}u_i' D u_i - \frac{1}{2}\log|D| - \frac{1}{2\sigma^2}\varepsilon_i\varepsilon_i' - \frac{n_i}{2}\log\sigma^2\right)$$

$$\underset{\theta}{\operatorname{argmax}}\, \ell(\theta \mid \varepsilon_1, \cdots, u_n) = \begin{cases} D = \frac{1}{N}\sum_{i=1}^{n} u_i u_i' \\ \sigma^2 = (1/\sum_{i=1}^{N} n_i)\cdot \sum_{i=1}^{n} \varepsilon_i'\varepsilon_i \\ \beta = (\sum_{i=1}^{n} X_i'X_i)^{-1}\sum_{i=1}^{N} X_i'(Y_i - Z_i u_i) \end{cases}$$

① E step : $E(u_i u_i' \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)})$

$E(\varepsilon_i'\varepsilon_i \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)})$

$E(u_i \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)})$

$\rightarrow$ Use $Var(X) = E(X^2) - \{E(X)\}^2$

$E(X^2) = Var(X) + \{E(X)\}^2$.

$$E(u_i u_i' \mid Y_i) = Var(u_i \mid Y_i) + E(u_i \mid Y_i)\,E(u_i' \mid Y_i)$$

$$\begin{pmatrix} Y_i \\ u_i \end{pmatrix} \sim MVN\left(\begin{pmatrix} X_i\beta \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_i & Z_i D \\ D Z_i' & D \end{pmatrix} \right), \text{ where } \Sigma_i = Z_i D Z_i' + \sigma^2 I_{n_i}$$

Then, $E(u_i \mid Y_i) = 0 + D Z_i' \Sigma_i^{-1}(Y_i - X_i\beta) = D Z_i' \Sigma_i^{-1}(Y_i - X_i\beta)$

$$Var(u_i \mid Y_i) = D - D Z_i' \Sigma_i^{-1} Z_i D.$$

$$\therefore E(u_i u_i' \mid Y_i) = (D - D Z_i' \Sigma_i^{-1} Z_i D) + (D Z_i' \Sigma_i^{-1}(Y_i - X_i\beta))(D Z_i' \Sigma_i^{-1}(Y_i - X_i\beta))'$$

$$E(\varepsilon_i' \varepsilon_i \mid Y_i) = E(\varepsilon_i' \mid Y_i) \cdot E(\varepsilon_i \mid Y_i) + Var(\varepsilon_i \mid Y_i)$$

$$\begin{pmatrix} Y_i \\ u_i \end{pmatrix} \sim MVN\left(\begin{pmatrix} X_i\beta \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_i & G I_{n_i} \\ G I_{n_i} & G^2 I_{n_i} \end{pmatrix} \right) \rightarrow \begin{array}{l} E(\varepsilon_i \mid Y_i) = 0 + G\Sigma_i^{-1}(Y_i - X_i\beta). \\ Var(\varepsilon_i \mid Y_i) = G^2 I_{n_i} - G^2 \Sigma_i^{-1} \end{array}$$

$$\therefore E(\varepsilon_i' \varepsilon_i \mid Y_i) = (G\Sigma_i^{-1}(Y_i - X_i\beta))'(G\Sigma_i^{-1}(Y_i - X_i\beta)) + G^2 I_{n_i} - G^2 \Sigma_i^{-1}.$$

$$\& \; E(u_i \mid Y_i) = D \cdot Z_i'(Y_i - X_i\beta) / \Sigma_i.$$

② M step : $D^{t+1} = \frac{1}{N} \sum_{i=1}^{n} E(u_i u_i' \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)})$.

$$\sigma^{2(t+1)} = \left(1 / \sum_{i=1}^{n} n_i\right) \cdot \sum_{i=1}^{n} E(\varepsilon_i' \varepsilon_i \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)}).$$

$$\beta^{(t+1)} = \left(\sum_{i=1}^{n} X_i' X_i\right)^{-1} \sum_{i=1}^{n} X_i' E(Y_i - Z_i u_i \mid Y_i, \beta^{(t)}, D^{(t)}, \sigma^{2(t)}).$$

Computing Final Exam

Dongeun Min

12/20/2020

Settings

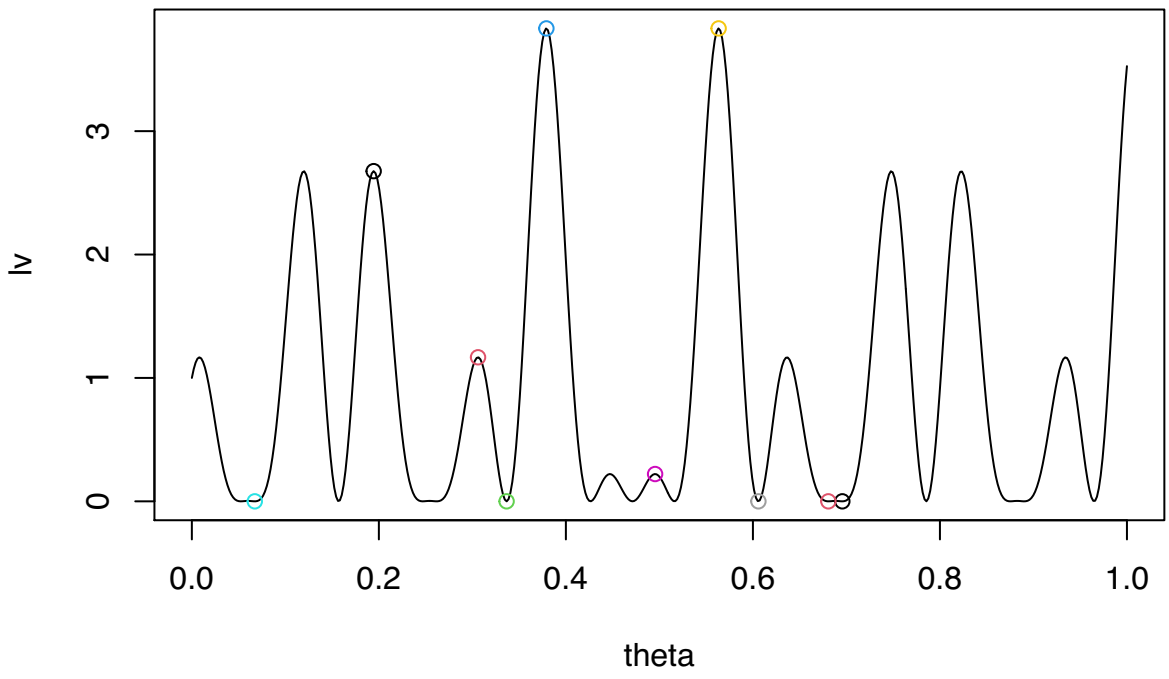
```
library("Rcpp")

# Compile C++ code and import function into R
current_path = rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
sourceCpp("Computing_Final_2020324011.cpp")
```

Question 1.(a)

```
h = function(x){
  value = (cos(50*x)+sin(20*x))^2
  return(value)
}

hprime1 = function(x){
  value = 2 * (cos(50*x)+sin(20*x)) * (-50*sin(50*x)+20*cos(20*x));
  return(value)
}

hprime2 = function(x){
  temp1 = (-50*sin(50*x)+20*cos(20*x))^2;
  temp2 = (cos(50*x)+sin(20*x))*(-2500*cos(50*x)-400*sin(20*x));
  value = 2*(temp1 + temp2);
  return(value)
}

# newton method
newton = function(init, eps){
  x = init;
  i = 0;
  diff = 1.0;
  while(abs(diff) > eps){
    diff = -hprime1(x)/hprime2(x)
    x = x + diff
    i = i + 1
  }
```

```
  return(x)
}
```

```
theta = seq(1e-7, 1-(1e-7), length=500)
lv = theta
for(i in 1:length(theta)){
  lv[i] = h(theta[i])
}
plot(theta, lv, type="l")

start = c(0.2, 0.3, 0.35, 0.38, 0.4, 0.5, 0.56, 0.6, 0.7, 0.8)
x_hat = rep(0, 10)
like = rep(0, 10)
for (i in 1:length(start)){
  x_hat[i] = newton(start[i], 1e-10)
  like[i] = h(x_hat[i])
  points(x_hat[i], h(x_hat[i]), col=i)
}
```




```
x_hat
```

```
##  [1] 0.19445072 0.30604352 0.33659921 0.37913844 0.06731984 0.49541481
##  [7] 0.56333935 0.60587858 0.69563837 0.68067841
```

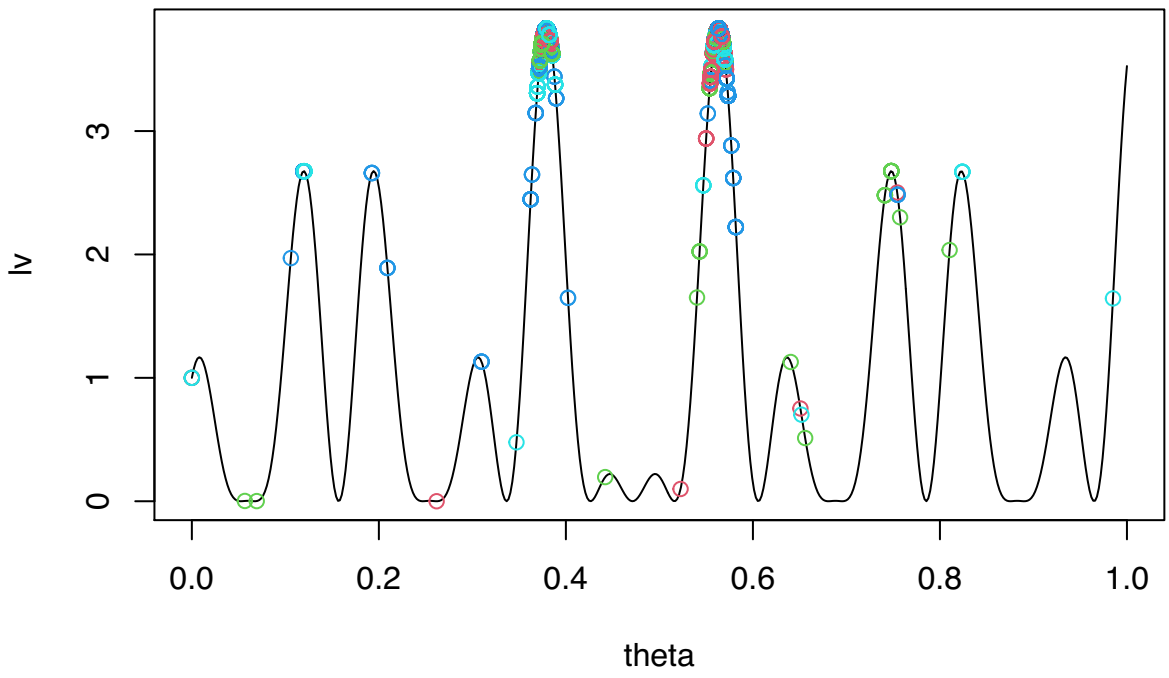
```
like
```

```
##  [1] 2.675667e+00 1.167253e+00 6.039716e-31 3.832544e+00 0.000000e+00
##  [6] 2.211757e-01 3.832544e+00 1.232595e-32 3.562200e-30 3.081488e-31
```

The maximum value is 3.832544.

Question 1.(b)

```
x = matrix(0, 2500, 4)  #initial value = 0
hval = matrix(0, 2500, 4)
r = 0.5
for (j in 1:4){
  for (i in 1:2499){
    a_t = max(x[i,j]-r, 0)
    b_t = min(x[i,j]+r, 1)
    u = runif(1, a_t, b_t)
    T_t = 1/log(i)
    ratio = min(exp((h(u)-h(x[i,j]))/T_t), 1)
    prob = runif(1, 0, 1)
    if (prob < ratio){
      x[i+1,j] = u
    } else {
      x[i+1,j] = x[i,j] }
    # update T_t and T_(t+1) ??
  }
}
for (j in 1:4){
  for (i in 1:2500){
    hval[i,j] = h(x[i,j]) }}

# Plotting
theta = seq(1e-7, 1-(1e-7), length=500)
lv = theta
for (i in 1:length(theta)){
  lv[i] = h(theta[i])
}
plot(theta, lv, type="l")
for (i in 1:2500){
  points(x[i,1], hval[i,1], col=2)
  points(x[i,2], hval[i,2], col=3)
  points(x[i,3], hval[i,3], col=4)
  points(x[i,4], hval[i,4], col=5)
}
```




```
x[2500,]
```

```
## [1] 0.5668107 0.3743752 0.5630223 0.3822225
```

```
hval[2500,]
```

```
## [1] 3.764962 3.705830 3.831980 3.779578
```

From the simulated annealing algorithm, it is found that the maximum value is found when x is around 0.38 or 0.56.
The maximum value found is 0.83251.

Question 2.(b)

```
rikz <- read.table("/Users/michelle/coursework/CS/rikz.txt", header = TRUE)
Y = rikz[,1]  #Response Variable Y: Richness
X = rikz[,3]  #Fixed Effect Variable X: NAP
Z = rikz[,4]  #Random Effect Variable Z: Beach
N = nrow(rikz)
beta = 1  #initial value of beta
D = 0.5  #initial value of D (= variance of b_i)
sig = 0.5  #initial value of sigma(small)
```

```
evec = rep(0,3*N)
estep = function(x, y, z, b, d, s){
  exp_s = rep(0,N)
  exp_e = rep(0,N)
  exp_u = rep(0,N)
  for (i in 1:nrow(rikz)) {
    sigma = d*(z[i]^2) + s^2
    exp_s[i] = (d - (d^2)*(z[i]^2)/sigma) + (d*z[i]*(y[i]-x[i]*b)/sigma)^2
    exp_e[i] = (s^2 - (s^2)/sigma) + (s*(y[i]-x[i]*b)/sigma)^2
    exp_u[i] = d*z[i]*(y[i]-x[i]*b)/sigma
  }
  evec[1:N] = exp_s
  evec[(N+1):(2*N)] = exp_e
  evec[(2*N+1):(3*N)] = exp_u
  return(evec)
}
mvec = rep(0,3)
mstep = function(vec, x){
  mvec[1] = mean(vec[1:N])
  mvec[2] = mean(vec[(N+1):(2*N)])
  sumx = 0; sume = 0
  for (i in 1:nrow(rikz)) {
    sumx = sumx + (x[i]^2)
    sume = sume + x[i]*vec[2*N+i]
  mvec[3] = sume/sumx
  }
  return(mvec)
}

iter = 100
for (i in 1:iter){
  e = estep(X,Y,Z,beta,D,sig)
  p = mstep(e,X)
}
p  ## output
```

```
## [1] 11.3333741  7.7893444 -0.6510261
```

The estimates of D, sigma^2, and beta are 11.3333741, 7.7893444, -0.6510261.

Question 3

```
data <- read.csv("/Users/michelle/coursework/CS/data.csv", header = TRUE)
v = 6
iter = 10  # get result of 100 patients: iter = 100
# Calculate utility function with trapezoidal, simpson's rule
utility1 = rep(0,iter)
utility2 = rep(0,iter)
for (i in 1:iter){
  X = data[i,4]
```

```
  W = data[i,1]
  A = data[i,2]
  utility1[i] = 0.5*u1trapezoid(X, W, A, 0, v/2, 1000) + 5*u2trapezoid(X, W, A, 0, v/2, 1000) +
  10*u1trapezoid(X, W, A, v/2, v, 1000) + 20*u2trapezoid(X, W, A, v/2, v, 1000)
  utility2[i] = 0.5*u1simpson(X, W, A, 0, v/2, 1000) + 5*u2simpson(X, W, A, 0, v/2, 1000) +
  10*u1simpson(X, W, A, v/2, v, 1000) + 20*u2simpson(X, W, A, v/2, v, 1000)
}
utility1
```

```
##  [1] 4.453321 3.743581 3.743581 4.453321 3.130399 3.130399 3.743581 3.743581
##  [9] 3.743581 3.130399
```

```
utility2
```

```
##  [1] 4.453418 3.743651 3.743651 4.453418 3.130435 3.130435 3.743651 3.743651
##  [9] 3.743651 3.130435
```

For the first 10 patients in the clinical data (data.csv), the utility function calculated with numerical integration using the trapezoidal rule is shown in the first line of the code output.
The utility function calculated with numerical integration using simpson's rule is shown in the second line of the code output.
The result of the first 100 patients can also be calculated with the code above (if 'iter' is changed to iter = 100), but it takes more time and the output becomes much longer, so only 10 patients are shown above.

Question 4

```
# data import & preprocessing
cancer = read.table("/Users/michelle/coursework/CS/UScancer.txt", colClasses = "character")
cancer = as.matrix(cancer)
cancer = strsplit(cancer, NULL)
cancer = unlist(cancer)
cancer = as.numeric(cancer)
cancer = matrix(cancer, nrow = 58, byrow = T)
cancer[cancer==0] <- -1
cancer[cancer==2] <- 0

# estimates
mple = c(-0.3205, 0.1115)
dmh = c(-0.3020, 0.1227)
aex = c(-0.3017, 0.1224)

# Bootstrap
iter = 10
table1 = cancer
bias1 = 0
s1sum1 = rep(0,iter)
s2sum1 = rep(0,iter)
s2sum11 = rep(0,iter)
for (i in 1:iter){
  table1 = bootstrap(table1, mple[1], mple[2])
```

```
  bias1 = bias1 + (sum(table1) - sum(cancer))^2
  s1sum1[i] = sum(table1)
  s2sum1[i] = (rmse_s2(cancer, table1)[2] - rmse_s2(cancer, table1)[1])^2
  s2sum11[i] = rmse_s2(cancer, table1)[2]
}
var1 = sum((s1sum1 - mean(s1sum1))^2) / iter
bias11 = sum(s2sum1) / iter
var11 = sum((s2sum11 - mean(s2sum11))^2) / iter
rmse_s1_mple = sqrt(bias1+var1)
rmse_s2_mple = sqrt(bias11+var11)

table2 = cancer
bias2 = 0
s1sum2 = rep(0,iter)
s2sum2 = rep(0,iter)
s2sum22 = rep(0,iter)
for (i in 1:iter){
  table2 = bootstrap(table2, dmh[1], dmh[2])
  bias2 = bias2 + (sum(table2) - sum(cancer))^2
  s1sum2[i] = sum(table2)
  s2sum2[i] = (rmse_s2(cancer, table2)[2] - rmse_s2(cancer, table2)[1])^2
  s2sum22[i] = rmse_s2(cancer, table2)[2]
}
var2 = sum((s1sum2 - mean(s1sum2))^2) / iter
bias22 = sum(s2sum2) / iter
var22 = sum((s2sum22 - mean(s2sum22))^2) / iter
rmse_s1_dmh = sqrt(bias2+var2)
rmse_s2_dmh = sqrt(bias22+var22)

table3 = cancer
bias3 = 0
s1sum3 = rep(0,iter)
s2sum3 = rep(0,iter)
s2sum33 = rep(0,iter)
for (i in 1:iter){
  table3 = bootstrap(table3, aex[1], aex[2])
  bias3 = bias3 + (sum(table3) - sum(cancer))^2
  s1sum3[i] = sum(table3)
  s2sum3[i] = (rmse_s2(cancer, table3)[2] - rmse_s2(cancer, table3)[1])^2
  s2sum33[i] = rmse_s2(cancer, table3)[2]
}
var3 = sum((s1sum3 - mean(s1sum3))^2) / iter
bias33 = sum(s2sum3) / iter
var33 = sum((s2sum33 - mean(s2sum33))^2) / iter
rmse_s1_aex = sqrt(bias3+var3)
rmse_s2_aex = sqrt(bias33+var33)

# Calculate RMSE
rmse_s1_mple; rmse_s1_dmh; rmse_s1_aex
```

```
## [1] 5276.548
```

```
## [1] 5699.816
```

```
## [1] 5578.075
```

```
rmse_s2_mple; rmse_s2_dmh; rmse_s2_aex
```

```
## [1] 5610.953
```

```
## [1] 5593.084
```

```
## [1] 5704.982
```

The RMSE of alpha and beta is lowest when using estimation with MPLE (maximum pseudo-likelihood estimator). The difference between each RMSE of estimates of beta is smaller than alpha.