MP HW3.4

Michael Pena

2024-02-28

J-2.2 (continued)

```
#function to square root a matrix "A"
sqrtm <- function(A){
a <- eigen(A)
sqm <- a$vectors %*% diag(sqrt(a$values)) %*% t(a$vectors)
sqm <- (sqm+t(sqm))/2
}
#function for generating data
gen <- function(n,p,mu,sigma,seed){
#generate data from a p-variate normal with mean mu and covaraince sigma
#set seed to 2024
set.seed(seed)
#generate data from normal
z <- matrix(rnorm(n*p),n,p)
datan <- z %*% sqrtm(sigma) + matrix(mu,n,p,byrow = TRUE)
datan
}

# putting in the data
sig <- matrix(c(1,0.7,0.7,0.7,1,0.7,0.7,0.7,1), nrow = 3, ncol = 3)
mu <- matrix(c(-1,1,2), nrow =3)
x <- gen(200,3,mu,sig,2025)
```

```
# initials
I3 <- diag(3)
mu_0 <- matrix(0,3,1)
abstol = 1e-05


 # turn theta into a mu and sigma
 from.theta <- function(p,theta){
   mu <- theta[1:p]
   sig <- matrix(0, nrow = p, ncol = p)

   k = p + 1

   for (i in 1:p){
     for (j in 1:i){
       sig[i,j] <- theta[k]
       sig[j,i] <- sig[i,j]
       k = k + 1
     }
   }
  list(mu = mu, sig = sig)
 }

# # compile Sigma and Mu into a single theta vector
 to.theta <- function(mu,sig){
   p <- nrow(sig)
   theta <- matrix(0,nrow = p + p*(1+p)/2,ncol = 1)
   theta[1:p] <- mu

   k = p + 1
   for(i in 1:p){
     for(j in 1:i){
       theta[k] <- sig[i,j]
       k = k + 1
     }
   }
   return(theta)
 }

# make gradient
gradient <- function(x,mu,sig){
  p <- nrow(sig)
  n <- nrow(x)
  inv.sig <- solve(sig)
  # set initials
  xi.sum <- matrix(0, p, 1)
  C.mu <- matrix(0, p, p)
  # compute sum of Xi and sum C(mu)
  for(i in 1:n){
    xi <- x[i,] - mu
    xi.sum <- xi.sum + xi
    C.mu <- C.mu + xi %*% t(xi)
  }
  # place elements into gradient mu and gradient sig
  grad.mu <- inv.sig %*% xi.sum
  A <- (n * inv.sig) - inv.sig %*% C.mu %*% inv.sig
  grad.sig <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  #gradient sig
  for(i in 1:nrow(sig)){
    grad.sig[i,i] <- -(1/2) * A[i,i]
  }
  for(i in 1:nrow(sig)-1){
    for (j in (i+1):ncol(sig)){
      grad.sig[i,j] <- -1 * A[i,j]
      grad.sig[j,i] <- grad.sig[i,j]
    }
  }
  grad.norm <- norm(to.theta(grad.mu,grad.sig), type = '2')
  list(grad.mu = grad.mu, grad.sig = grad.sig, grad.norm = grad.norm)
 }
```

```r
#likelihood function
likemvn <- function (x,mu,sig) {
  # computes the likelihood and the gradient for multivariate normal
  n = nrow(x)
  p = ncol(x)

  sig.inv <- solve(sig)
  C.mu = matrix(0,p,p) # initializing sum of (xi-mu)(xi-mu)^T
  xi.sum = matrix(0,p,1) # initializing sum of xi-mu
  for (i in 1:n){
    xi = x[i,] - mu
    C.mu = C.mu + xi %*% t(xi)
  }

  ell = -(n*p*log(2*pi)+n*log(det(sig)) + sum(sig.inv * C.mu ))/2
  return(ell)
}
```

```r
# new function to run the optim() function

# Likelihood Function the passes theta vector
theta_opt <- function(theta,data){
  x <- data
  p <- ncol(x)
  sig <- from.theta(p,theta)$sig
  mu <- from.theta(p,theta)$mu
  if(all(eigen(sig)$values>0)){
  L <- likemvn(x,mu,sig)
  } else {
  L = NaN
  }
  return(L)
}

# gradient theta vector
grad_vec_opt <- function(theta,data){
 x <- data
 p <- ncol(x)
 sig <- from.theta(p,theta)$sig
 mu <- from.theta(p,theta)$mu
 grad_sig <- gradient(x,mu,sig)$grad.sig
 grad_mu <- gradient(x,mu,sig)$grad.mu
 grad_theta <- to.theta(grad_mu,grad_sig)
 return(grad_theta)
}
```

```r
# running optim()
theta_0 <- to.theta(mu_0,I3)
optim(par = theta_0,
      fn = theta_opt,
      gr = grad_vec_opt,
      data = x,
      method = "BFGS",
      control = list(fnscale = -1, trace = 1, abstol = 1e-5),
      hessian = TRUE)
```

```
## initial  value 1461.282329
## iter  10 value 740.079678
## iter  20 value 699.166236
## iter  30 value 699.128054
## final  value 699.127438
## converged
```

```
## $par
##              [,1]
##  [1,] -0.9915896
##  [2,]  0.9938697
##  [3,]  2.0319712
##  [4,]  0.9176866
##  [5,]  0.6112404
##  [6,]  0.9727371
##  [7,]  0.6902985
##  [8,]  0.7691464
##  [9,]  1.1088348
##
## $value
## [1] -699.1274
##
## $counts
## function gradient
##      111       31
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##                  [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,] -4.464025e+02  1.345794e+02  1.845538e+02 -9.301913e-06 -1.432136e-05
##  [2,]  1.345794e+02 -4.959286e+02  2.602207e+02  2.804306e-06 -5.171035e-06
##  [3,]  1.845538e+02  2.602207e+02 -4.757652e+02  3.845661e-06  1.250248e-05
##  [4,] -9.301913e-06  2.804306e-06  3.845661e-06 -4.981954e+02  3.003905e+02
##  [5,] -1.432136e-05 -5.171035e-06  1.250248e-05  3.003905e+02 -1.197504e+03
##  [6,]  5.162981e-06 -1.902551e-05  9.982910e-06 -4.527977e+01  3.337179e+02
##  [7,] -2.927390e-05  1.540709e-05  3.778715e-06  4.119372e+02  4.566367e+02
##  [8,]  1.706485e-05 -2.681114e-05  1.054550e-06 -1.241868e+02  2.825287e+02
##  [9,]  1.369245e-05  1.930637e-05 -3.529815e-05 -8.515167e+01 -2.401267e+02
##                  [,6]          [,7]          [,8]          [,9]
##  [1,]  5.162981e-06 -2.927390e-05  1.706485e-05  1.369245e-05
##  [2,] -1.902551e-05  1.540709e-05 -2.681114e-05  1.930637e-05
##  [3,]  9.982910e-06  3.778715e-06  1.054550e-06 -3.529815e-05
##  [4,] -4.527977e+01  4.119372e+02 -1.241868e+02 -8.515167e+01
##  [5,]  3.337179e+02  4.566367e+02  2.825287e+02 -2.401267e+02
##  [6,] -6.148743e+02 -1.751036e+02  6.452751e+02 -1.692901e+02
##  [7,] -1.751036e+02 -1.232248e+03  8.001323e+01  4.390333e+02
##  [8,]  6.452751e+02  8.001323e+01 -1.518361e+03  6.190392e+02
##  [9,] -1.692901e+02  4.390333e+02  6.190392e+02 -5.658909e+02
```

G.H. 2.3

part (b)

```
#building the likelihood
likelihood_wei <- function(t,d,w,a,b0,b1){

 length(t) -> n
 sum=0
  for(i in 1:n){
   sum = sum + (w[i]*log(a)+w[i]*(a-1)*log(t[i])-(t[i]^(a))*exp(b0+d[i]*b1))
  }

 return(sum)
}

#vectorize
to_theta <- function(a,b0,b1){
th <- matrix(c(a,b0,b1),ncol=1)
return(th)
}


# building the gradient function
gradient_wei <- function(t,d,w,a,b0,b1){

 dLda = 0 # intials
 dLdb0 = 0
 dLdb1 = 0
 length(t) -> n

 for(i in 1:n){
 dLda <- dLda + (w[i]/a+w[i]*log(t[i])-(t[i]^(a))*log(t[i])*exp(b0+d[i]*b1))
 }

 for(i in 1:n){
  dLdb0 = dLdb0 - (t[i]^(a))*exp(b0+d[i]*b1)
 }

 for(i in 1:n){
  dLdb1 = dLdb1 - (t[i]^(a))*exp(b0+d[i]*b1)*d[i]
 }

vec <- matrix(c(dLda,dLdb0,dLdb1), nrow = 3)
return(vec)
}

# rendering the hessian
hessian_wei <- function(t,d,w,a,b0,b1){
  H <- matrix(0,3,3)
  length(t) -> n
  #L_aa
  for(i in 1:n){
   H[1,1] = H[1,1] - w[i]/(a^2)-2*(t[i]^(a))*log(t[i])*exp(b0+d[i]*b1)
  }
  #L_ab0
  for(i in 1:n){
   H[2,1] = H[2,1] - (t[i]^(a))*log(t[i])*exp(b0+d[i]*b1)
  }
  H[1,2] = H[2,1]
  #L_ab1
  for(i in 1:n){
   H[3,1] = H[3,1] - (t[i]^(a))*log(t[i])*exp(b0+d[i]*b1)*d[i]
  }
  H[1,3] = H[3,1]
  #L_b0b0
  for(i in 1:n){
   H[2,2] = H[2,2] - (t[i]^(a))*exp(b0+d[i]*b1)
  }
  #L_b0b1
  for(i in 1:n){
   H[3,2] = H[3,2] - (t[i]^(a))*exp(b0+d[i]*b1)*d[i]
  }
  H[2,3] = H[3,2]
  #L_b1b1
  for(i in 1:n){
   H[3,3] = H[3,3] - (t[i]^(a))*exp(b0+d[i]*b1)*d[i]^2
  }

 return(H)
}
```

```
# input data
t <- c(6,9,10,11,17,19,20,25,32,32,34,35,6,6,6,7,10,13,16,22,23,1,1,2,2,3,4,4,5,5,8,8,8,8,11,11,12,12,1
d <- c(0,0,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1,1,1,1,1 ,1 ,1 ,1 ,1 ,1,1,1,1,1,1,1,1,1,1,1,1,1,1 ,1 ,1 ,1 ,1
w <- c(1,1,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1,1,1,1,1 ,1 ,1 ,1 ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
# newtons method

newton2 <- function(t,d,w,a,b0,b1,maxit,tolerr,tolgrad){
  header = paste0("Iteration", "      halving", "     log-likelihood","      ||Gradient||")
  print(header)

 it = 1
 stop = FALSE
 while(it <= maxit & stop == FALSE){
  # first steps
  theta0 <- to_theta(a,b0,b1)
  L0 <- likelihood_wei(t,d,w,a,b0,b1)
  #grad elements
  grad_0 <- gradient_wei(t,d,w,a,b0,b1)
  grad_norm <- norm(grad_0)
  grad_a <- grad_0[1]
  grad_b0 <- grad_0[2]
  grad_b1 <- grad_0[3]
  #get direction
```

```r
hess <- hessian_wei(t,d,w,a,b0,b1)
inv_h <- solve(hess)
direc <- (-1)*(inv_h %*% grad_0)
#print
  if (it == 1 | it ==2 | it == 499 | it == 500){
    print(sprintf('%2.0f                 --          %3.4f               %.1e',it,L0,grad_norm))
  }

#get new params
theta1 = theta0 + direc
a_n <- theta1[1]
b0_n <- theta1[2]
b1_n <- theta1[3]
grad_norm1 <- gradient_wei(t,d,w,a_n,b0_n,b1_n)

if(theta1[1] > 0){
    L1 <- likelihood_wei(t,d,w,a_n,b0_n,b1_n)
} else {L1 <- NaN}

  halve <- 0
    if(it == 1 | it ==2 | it == 499 | it == 500){
        print(sprintf('%2.0f                 %2.0f          %3.4f               %.1e',it,  halve,L1, g
  }

  while(halve <= 20 & (theta1[1] <= 0 || L1 < L0)){

  theta1 = theta0 + direc/(2^halve)

  if(theta1[1] > 0){
   a_n <- theta1[1]
   b0_n <- theta1[2]
   b1_n <- theta1[3]

   L1 <- likelihood_wei(t,d,w,a_n,b0_n,b1_n)
   grad_norm1 <- norm(gradient_wei(t,d,w,a_n,b0_n,b1_n))
  }

  halve = halve + 1
    if(it == 1 | it ==2 | it == 499 | it == 500){
        print(sprintf('%2.0f                 %2.0f          %3.4f               %.1e',it,  halve,L1, g
  }

}
      if(it == 1 | it == 2 | it == 499){
    print("-----------------------------------------------------------------")
    print(header)
  }
   r.e = max(abs(theta0 - theta1)/abs(pmax(1,abs(theta0))))
  if (r.e < tolerr & grad_norm1 < tolgrad){stop == TRUE}
  a <- a_n
  b0 <- b0_n
  b1 <- b1_n
  it <- it + 1
```

```
 }
    return(list("estimator of alpha"=a, "estimator of beta_0" = b0, "estimator of beta_1" = b1, "iterat
}

newton2(t,d,w,1,1,1,500,1e-07,1e-07)
```

```
## [1] "Iteration      halving     log-likelihood      ||Gradient||"
## [1] " 1                 --          -2829.7858               1.2e+04"
## [1] " 1                  0          -1042.2027              -2.7e+03"
## [2] " 1                  0          -1042.2027              -1.0e+03"
## [3] " 1                  0          -1042.2027              -7.9e+02"
## [1] "-----------------------------------------------------------------"
## [1] "Iteration      halving     log-likelihood      ||Gradient||"
## [1] " 2                 --          -1042.2027               4.5e+03"
## [1] " 2                  0          -387.4379               -9.3e+02"
## [2] " 2                  0          -387.4379               -3.8e+02"
## [3] " 2                  0          -387.4379               -2.9e+02"
## [1] "-----------------------------------------------------------------"
## [1] "Iteration      halving     log-likelihood      ||Gradient||"
## [1] "499                 --          54.5393               1.0e+02"
## [1] "499                  0          29.4896               3.1e+01"
## [2] "499                  0          29.4896               -1.4e+01"
## [3] "499                  0          29.4896               -1.1e+01"
## [1] "499                  1          29.4896               5.6e+01"
## [1] "499                  2          45.1707               4.4e+01"
## [1] "499                  3          50.8335               6.4e+01"
## [1] "499                  4          52.9618               8.2e+01"
## [1] "499                  5          53.8238               9.2e+01"
## [1] "499                  6          54.2004               9.7e+01"
## [1] "499                  7          54.3747               1.0e+02"
## [1] "499                  8          54.4582               1.0e+02"
## [1] "499                  9          54.4991               1.0e+02"
## [1] "499                 10          54.5193               1.0e+02"
## [1] "499                 11          54.5293               1.0e+02"
## [1] "499                 12          54.5343               1.0e+02"
## [1] "499                 13          54.5368               1.0e+02"
## [1] "499                 14          54.5381               1.0e+02"
## [1] "499                 15          54.5387               1.0e+02"
## [1] "499                 16          54.5390               1.0e+02"
## [1] "499                 17          54.5391               1.0e+02"
## [1] "499                 18          54.5392               1.0e+02"
## [1] "499                 19          54.5393               1.0e+02"
## [1] "499                 20          54.5393               1.0e+02"
## [1] "499                 21          54.5393               1.0e+02"
## [1] "-----------------------------------------------------------------"
## [1] "Iteration      halving     log-likelihood      ||Gradient||"
## [1] "500                 --          54.5393               1.0e+02"
## [1] "500                  0          29.4895               3.1e+01"
## [2] "500                  0          29.4895               -1.4e+01"
## [3] "500                  0          29.4895               -1.1e+01"
## [1] "500                  1          29.4895               5.6e+01"
## [1] "500                  2          45.1707               4.4e+01"
## [1] "500                  3          50.8334               6.4e+01"
## [1] "500                  4          52.9617               8.2e+01"
## [1] "500                   5          53.8237              9.2e+01"
## [1] "500                   6          54.2004              9.7e+01"
## [1] "500                   7          54.3746              1.0e+02"
## [1] "500                   8          54.4582              1.0e+02"
## [1] "500                   9          54.4990              1.0e+02"
## [1] "500                  10          54.5192              1.0e+02"
## [1] "500                  11          54.5293              1.0e+02"
## [1] "500                  12          54.5343              1.0e+02"
## [1] "500                  13          54.5368              1.0e+02"
## [1] "500                  14          54.5380              1.0e+02"
## [1] "500                  15          54.5387              1.0e+02"
## [1] "500                  16          54.5390              1.0e+02"
## [1] "500                  17          54.5391              1.0e+02"
## [1] "500                  18          54.5392              1.0e+02"
## [1] "500                  19          54.5393              1.0e+02"
## [1] "500                  20          54.5393              1.0e+02"
## [1] "500                  21          54.5393              1.0e+02"
```

```
## $`estimator of alpha`
## [1] 2.296857
##
## $`estimator of beta_0`
## [1] -7.851381
##
## $`estimator of beta_1`
## [1] 2.05616
##
## $iteration
## [1] 501
```