MP HW3.4

Michael Pena

2024-02-28

{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)

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
I <- 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)
```

```
 }

#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)
}
```

```
# 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)
}
```

```
# running optim()
theta_0 <- to.theta(mu_0,I)
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)
```