STAT 243: Problem Set 7

Jin Rou New [jrnew]

November 14, 2014

1 Problem 4g

The Cholesky decomposition is the fastest, followed by LU decomposition. Computing the A^{-1} is the slowest by far. The difference in speeds increases as n and p increase.

```
GetMatrices <- function(n, p) {
  Z <- matrix(rnorm(n^2), n, n)
  A <- crossprod(Z) # t(Z) %*% Z
  X <- matrix(rnorm(n*p), n, p)
  B <- A %*% X
  return(list(A = A, X = X, B = B))
}

set.seed(123)
for (n in c(100, 3000)) {
  for (p in c(1, 100, 3000)) {
    cat("n = ", n, "; p = ", p, "\n")
    matrices <- GetMatrices(n = n, p = p)
    # Method 1: Solve with LU decomposition
    print(system.time({
      X1 <- solve(matrices$A, matrices$B)
    }))

    # Method 2: Solve to find the inverse followed by matrix multiplication
    print(system.time({
      X2 <- solve(matrices$A) %*% matrices$B
    }))

    # Method 3: Cholesky decomposition
    print(system.time({
      U <- chol(matrices$A)
      X3 <- backsolve(U, backsolve(U, matrices$B, transpose = TRUE))
    }))
  }
}

## n =  100 ; p =  1
##    user  system elapsed
##   0.001   0.000   0.001
##    user  system elapsed
##   0.001   0.000   0.002
##    user  system elapsed
##   0.001   0.001   0.008
## n =  100 ; p =  100
##    user  system elapsed
##   0.002   0.000   0.001
##    user  system elapsed
##   0.003   0.000   0.003
##    user  system elapsed
##   0.002   0.000   0.002
## n =  100 ; p =  3000
##    user  system elapsed
##   0.028   0.000   0.028
##    user  system elapsed
##   0.024   0.001   0.025
##    user  system elapsed
##   0.030   0.001   0.031
## n =  3000 ; p =  1
##    user  system elapsed
##   6.570   0.008   6.582
##    user  system elapsed
##  23.347   0.068  23.425
##    user  system elapsed
##   4.344   0.004   4.347
## n =  3000 ; p =  100
##    user  system elapsed
##   7.342   0.007   7.354
##    user  system elapsed
##  24.380   0.052  24.440
##    user  system elapsed
##   5.188   0.004   5.195
## n =  3000 ; p =  3000
##    user  system elapsed
##   31.33    0.04   31.39
##    user  system elapsed
##  44.826   0.103  44.931
##    user  system elapsed
##  30.917   0.048  30.982
```

2 Problem 5

```
GetBeta <- function(X, Y, A, b) {
  X_qr <- qr(X)
  C <- crossprod(qr.R(X_qr), qr.R(X_qr))
  Cinvd <- backsolve(qr.R(X_qr), crossprod(qr.Q(X_qr), Y))
  ARinv <- A %*% solve(qr.R(X_qr))
  ACinvA <- tcrossprod(AR, AR)
  ACinvd <- A %*% Cinvd
  betahat <- Cinvd + solve(C, crossprod(A, solve(ACinvA, -ACinvd + b)))
}
```

3 Problem 6

Comparing across trials with the same condition number and different magnitudes of eigenvalues, we can see that the error in estimated eigenvalues increases with the magnitude of the eigenvalues. Similarly, the error increases with the condition number.
The condition number where the matrix is not numerically positive definite varies depending on the matrix A but in this particular set of results, the relevant condition number is 10^15.

```r
set.seed(6)
n <- 100
Z <- matrix(rnorm(n), n, n)
A <- crossprod(Z)
eigen_list <- eigen(A)
gamma <- cbind(eigen_list$vectors)

CreateMatrix <- function(gamma, eigenvalues) {
  lambda <- diag(eigenvalues)
  return(gamma %*% lambda %*% solve(gamma))
}

norm2 <- function(x)
  sqrt(sum(x^2))

num_trials <- 12
results_list <- list()
eigenvalues_ij <- results_ij <- matrix(NA, num_trials, n)
A_ijk <- array(NA, c(num_trials, n, n))
condition_number_i <- rep(NA, num_trials)

# Specify sets of eigenvalues to test with
eigenvalues_ij[1, ] <- rep(0.00001, n)
eigenvalues_ij[2, ] <- rep(1, n)
eigenvalues_ij[3, ] <- rep(100000, n)
eigenvalues_ij[4, ] <- seq(1, n)/100000
eigenvalues_ij[5, ] <- seq(1, n)
eigenvalues_ij[6, ] <- seq(1, n)*100000
eigenvalues_ij[7, ] <- seq(1, n*1000000000000, length.out = n)/100000
eigenvalues_ij[8, ] <- seq(1, n*1000000000000, length.out = n)
eigenvalues_ij[9, ] <- seq(1, n*1000000000000, length.out = n)*100000
eigenvalues_ij[10, ] <- seq(1, n*10000000000000, length.out = n)/100000
eigenvalues_ij[11, ] <- seq(1, n*10000000000000, length.out = n)
eigenvalues_ij[12, ] <- seq(1, n*10000000000000, length.out = n)*100000

# Create a new matrix A given eigenvectors and eigenvalues,
# then perform an eigendecomposition and calculate condition number
for (i in 1:num_trials) {
  A_ijk[i, , ] <- CreateMatrix(gamma = gamma, eigenvalues = eigenvalues_ij[i, ])
  results_ij[i, ] <- eigen(A_ijk[i, , ])$values
  condition_number_i[i] <- abs(max(eigenvalues_ij[i, ]))/abs(min(eigenvalues_ij[i, ]))
}

# Summarise results
is_positive_definite_i <- apply(Re(results_ij), 1, function(x) all(x > 0))
error_i <- apply(results_ij - eigenvalues_ij, 1, norm2)
```

```
condition_number_i

##  [1] 1e+00 1e+00 1e+00 1e+02 1e+02 1e+02 1e+14 1e+14 1e+14 1e+15 1e+15
## [12] 1e+15

data.frame(condition_number = condition_number_i,
           magnitude = rep(c(1/100000, 1, 100000), nrow(eigenvalues_ij)/3),
           is_positive_definite = is_positive_definite_i,
           error = error_i)

##    condition_number magnitude is_positive_definite     error
## 1             1e+00     1e-05                 TRUE 3.206e-19
## 2             1e+00     1e+00                 TRUE 3.198e-14
## 3             1e+00     1e+05                 TRUE 1.599e-09
## 4             1e+02     1e-05                 TRUE 5.773e-03
## 5             1e+02     1e+00                 TRUE 5.773e+02
## 6             1e+02     1e+05                 TRUE 5.773e+07
## 7             1e+14     1e-05                 TRUE 5.832e+09
## 8             1e+14     1e+00                 TRUE 5.832e+14
## 9             1e+14     1e+05                 TRUE 5.832e+19
## 10            1e+15     1e-05                FALSE 5.832e+10
## 11            1e+15     1e+00                FALSE 5.832e+15
## 12            1e+15     1e+05                FALSE 5.832e+20
```