Problem Set 7

Cameron Adams

November 16, 2017

1 Suppose I have a statistical method that estimates a regression ...

You can compare the standard errors estimated from simulation study, $\hat{SE}_i(\hat{\beta})_i$, to the standard error of the estimated coefficients $SE(\hat{\beta}_i)$ or to the mean of the estimated standard errors, $mean(\hat{SE}(\hat{\beta}_i))$.

2 Show that $||A||_2$ is the largest of the absolute values of the eigenvalues of A for symmetric A. To do so, find the following quantity ...

First we assume that A is symmetric, with $A = \Gamma\Lambda\Gamma^{-1}$, where, Λ is a diagonal matrix of eigenvalues and Λ are the eigenvectors.

$$
\begin{aligned}
||A||_2 &= \sup_{z:||z||_2} \sqrt{(Az)^\top Az} \\
&= \sqrt{z^\top A^\top Az} \\
&= \sqrt{z^\top(\Gamma\Lambda\Gamma^{-1})^\top(\Gamma\Lambda\Gamma)z} \\
&= \sqrt{z^\top(\Gamma^{-1})^\top\Lambda\Gamma^\top\Gamma\Lambda\Gamma^{-1}z} \\
&= \sqrt{(\Gamma^{-1}z)^\top\Lambda^2(\Gamma^{-1}z)} \\
\text{where } y &= \Gamma^\top z, \text{and if } ||z||_2 = 1 \text{ then } ||y||_2 = 1 \\
&= y^\top\Lambda^2 y \\
&= \sum_{i=1}^{n}\sqrt{\sum_{i=1}^{n}\lambda_i y_i^2} \\
\text{maximized when} &\sqrt{\sum_{i=1}^{n}\lambda_i y_i^2} = \sqrt{max(|\lambda_i|)}
\end{aligned}
$$

We can use the hint $\Gamma^\top z = y$ because Γ is orthogonal columns (SVD). Therefore, if $||z||_2 = 1$ then it follows that $||y||_2 = 1$. The norms of the columns of Γ are 1 by SVD. Therefore, the linear map of Γ applied to unit norm vector will give a unit norm vector output. When we get to the sum, we know that $\sum y_i = 1$. From there we can substitute the $\sum$ into the equation, and we know that this value is maximized when we have the largest singular/eigen value, which should be the first one.

3 Some practice with matrix manipulations.

3.1

X is a matrix with n x p dimensions, with n > p. $X^\top X = M$, where M is a matrix of dimension p x p and $XX^\top = M$ and M is a matrix with dimensions n x n. Therefore, the eigen decomposition of $X^\top X$, by definition of SVD:

$$\begin{aligned}\text{by definition, SVD of } X &= UDV^\top \\ &\text{therefore,} \\ X^\top X &= (UDV^\top)^\top (UDV^\top) \\ &= (VD^\top U^\top UDV^\top \\ &= VD^\top DV^\top \\ &= VD^2V^\top\end{aligned}$$

And for $XX^\top = M$, we see something similar

$$\begin{aligned}\text{by definition, SVD of } X &= UDV^\top \\ &\text{therefore,} \\ X^\top X &= (UDV^\top)(UDV^\top)^\top \\ &= UDV^\top VD^\top U^\top \\ &= UD^2U^\top\end{aligned}$$

Therefore, we see that the left and right singular vectors of X are the eigenvalues for $XX^\top$ and $X^\top X$, respectively.

To show that $X^\top X$ is p.s.d., we need the eigenvalues of $X^\top X$ to be positive and that $X^\top X$ is symmetric. From above, we see that eigen-decomposition of $X^\top X = VD^2V^\top$. The eigenvalues are D^2, and $D^2 > 0$. $X^\top X$ will be a square matrix with dimensions p x p, and is symmetric and therefore will be positive definite.

3.2

If we know that λ is a number that is an eigenvalue of Σ and v is a non-zero vector that is an eigenvector of Σ, there, $\Sigma v = \lambda v$. So, if we want an eigenvalue for $Z = \Sigma + cI$, we can substitute Z for Σ.

$$\begin{aligned}Zv &= \lambda v \\ (\Sigma + cI)v &= \lambda v \\ (\Sigma + cI) &= \lambda\end{aligned}$$

4

4.1

First I would simplify the expression by inverting $AC^{-1}A^\top)^{-1}$.

$$\begin{aligned}\beta &= C^{-1}d + C^{-1}A^\top(AC^{-1}A^\top)^{-1}(-AC^{-1}d + b) \\ &= C^{-1}d + C^{-}1A^\top(A^\top)^{-1}CA^{-1}(-AC^{-1}d + b) \\ &= C^{-1}d + A^{-1}(-AC^{-1}d + b) \\ &= C^{-1}d + A^1AC^{-1}d + A^{-1}b \\ &= C^{-1}d - C^{-1}d + A^{-1}b \\ &= A^{-1}b\end{aligned}$$

In the above simplification, we are assuming that A is invertable. However, if we can't assume A is invertable, then we can use eigen decomposition to remove the need for taking the inverse of $E^{-1} = (AC^{-1}A^\top)^{-1}$.

$$\begin{aligned}
\beta &= C^{-1}d + C^{-1}A^\top(AC^{-1}A^\top)^{-1}(-AC^{-1}d + b) \\
&= C^{-1}d + C^{-1}A^\top(eigen(AC^{-1}A^\top))^{-1}(-AC^{-1}d + b) \\
&= C^{-1}d + C^{-1}A^\top(\Gamma\Lambda\Gamma^\top)^{-1}(-AC^{-1}d + b) \\
&= C^{-1}d + C^{-1}A^\top(\Gamma^\top)^{-1}\Lambda^{-1}\Gamma^{-1}(-AC^{-1}d + b) \\
&= C^{-1}d + C^{-1}A^\top\Gamma\Lambda^{-1}\Gamma^\top(-AC^{-1}d + b)
\end{aligned}$$

If we assume, that the eigenvalues in $\Lambda > 0$, then we can use backsolve Cholesky and backsolves for $C^{-1}d$ and $C^{-1}A$ expressions. We would then want to solve this system from right to left.

4.2

We could write up three functions and one expression to solve this matrix:

1. q3.fun.solve(): The original way using *solve*(), which is the bad way
2. q3.fun.fast(): Use cholesky decomposition and some *backsolve*()'s to reduce the number of *solve*()'s
3. q3.fun.eigen(): Use eigen decomposition, cholesky decomposition, and *backsolve*'s to remove all *solve*()'s [MY ANSWER]
4. Simply $\beta = A^{-1}b$

The third function is my answer and probably the most correct. The fourth is the most fun.

```
rm(list=ls())
gc()
require(MASS)

m = 50
p = 20
n = 100

X = matrix(rnorm(n*p), ncol = p)
d = matrix(rnorm(p), ncol = 1)
A = matrix(rnorm(m*p), ncol = p)
b = matrix(rnorm(m), ncol = 1)

#1
q3.fun.solve <- function(X, d, A, b) {

    #get C
    C = crossprod(X) #symmetric t(X) %*% X

    #caculate parts of expression
    first  = solve(C) %*% d
    second = solve(C) %*% t(A)
    third  = ginv(A %*% solve(C) %*% t(A), tol = 1 * (.Machine$double.eps))
    fourth = -A %*% solve(C) %*% d + b

    #solve from right ot left
    first + (second %*% (third %*% fourth))
```

```
}

#2
q3.fun.fast <- function(X, d, A, b) {

    #Chol decomposition for C = X^tX
    U <- chol(crossprod(X), pivot = T) # U is upper-triangular

    #caculate parts of expression
    first  = backsolve(U, d)
    second = backsolve(U, t(A))
    third  = ginv(A %*% backsolve(U, t(A)), tol = 1 * (.Machine$double.eps))
    fourth = -A %*% backsolve(U, d) + b

    #solve from right ot left
    first + (second %*% (third %*% fourth))
}

#3
q3.fun.eigen <- function(X, d, A, b) {

    #Chol decomposition for C = X^tX
    U <- chol(crossprod(X), pivot = T) # U is upper-triangular

    #calculate parts of expression
    first  = backsolve(U, d)
    second = backsolve(U, t(A))
    E      = A %*% ginv(C) %*% t(A)
    E_eigen= eigen(E)
    third = E_eigen$vectors %*% diag(1/E_eigen$values) %*% E_eigen$vectors
    fourth = -A %*% backsolve(U, d) + b

    #solve from right ot left
    first + (second %*% (third %*% fourth))
}

#4
beta <- solve(A) %*% b
```

5

5.1

We cannot compute this matrix directly, even if we use matrix decomposition simply because the resulting matrices would be very very large. Memory space would be the biggest concern, but it would also be computational difficult to perform computations with matrices that are millions x millions in size without first reducing them, or performing sparse matrix operations.

5.2

We can rewrite the equations, by expanding X an $X^\top$ in the equation for $\hat{\beta}$. From there we can simplify and perform matrix operations and will result in matrices of manageable size. See below.

If,

$$\hat{X} = Z(Z^\top Z)^{-1}Z^\top X \text{ and}$$
$$\hat{X}^\top = X^\top Z((Z^\top Z)^{-1})^\top Z^\top \text{ then,}$$
$$\beta = (\hat{X}^\top \hat{X})^{-1}\hat{X}^\top y$$
$$\beta = [(X^\top Z((Z^\top Z)^{-1})^\top Z^\top)(Z(Z^\top Z)^{-1}Z^\top X)]^{-1}(X^\top Z((Z^\top Z)^{-1})^\top Z^\top)y$$
$$\beta = (X^\top Z((Z^\top Z)^{-1})^\top X)^{-1}X^\top Z((Z^\top Z)^{-1})^\top Z^\top y$$

Now, in this form, we have a series of matrix operations and will HUGELY reduce the size of the matrices used for calculation of β

$$A = X^\top Z = Z^\top X : (630\text{x}60\text{M})(60\text{Mx}1)=630\text{x}1$$
$$B = ((Z^\top Z)^{-1})^\top : (630\text{x}60\text{M})(60\text{Mx}630)=630\text{x}630$$
$$C = Z^\top y : (630\text{x}60\text{M})(60\text{Mx}630)=630\text{x}630$$

From here, we could plug in these much smaller matrices, A, B, C, into the expanded equation from above:

$$\beta = (X^\top Z((Z^\top Z)^{-1})^\top X)^{-1}X^\top Z((Z^\top Z)^{-1})^\top Z^\top y$$

This should be fairly easy to compute even for a regular computer. If memory were of particular concern, we could also remove the original matrices, Z, X, y from memory after we compute, A, B, C.

6

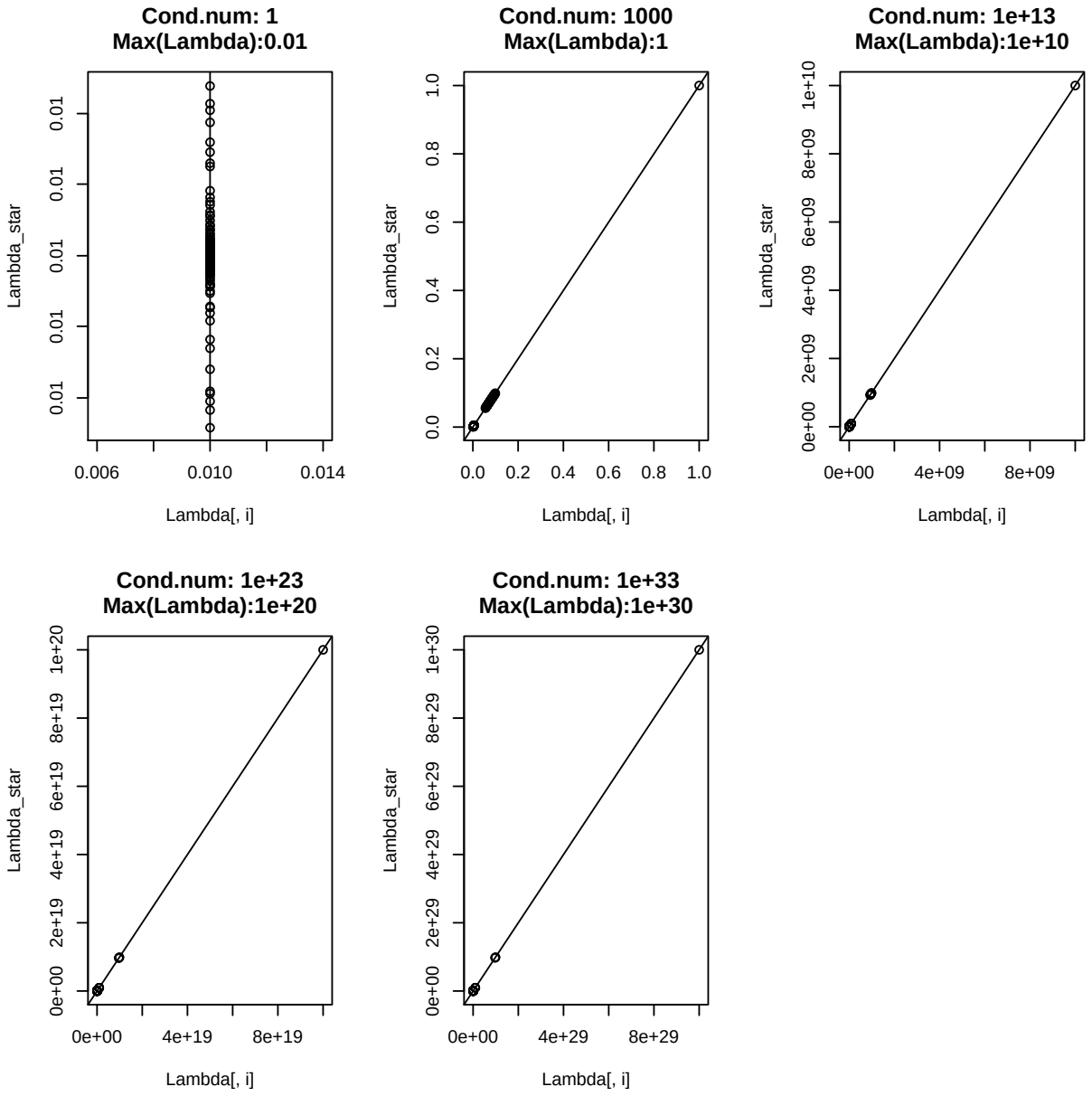
```
rm(list=ls())

#generate z matrix
n=100
z = matrix(rnorm(n * n), ncol = n)

#eigen decomp
z_eigen = eigen( crossprod(z) )
Gamma  <- z_eigen$vectors #Lambda

#generate eigenvalues
Lambda <- sapply(seq(-1, 32, 2), function(x)
    seq(0.1, 0.01, length.out = n) *
    10^floor(seq(x, -1, length.out = n)))

#add text eigen values of all the same number
Lambda <- cbind(rep(0.01, n), rep(5), rep(11), Lambda)
head(Lambda)
```

```
##      [,1] [,2] [,3]        [,4]       [,5]       [,6]       [,7]
## [1,] 0.01    5   11 0.010000000 1.00000000 100.000000 10000.0000
## [2,] 0.01    5   11 0.009909091 0.09909091   9.909091   990.9091
## [3,] 0.01    5   11 0.009818182 0.09818182   9.818182   981.8182
## [4,] 0.01    5   11 0.009727273 0.09727273   9.727273   972.7273
## [5,] 0.01    5   11 0.009636364 0.09636364   9.636364   963.6364
## [6,] 0.01    5   11 0.009545455 0.09545455   9.545455   954.5455
##            [,8]       [,9]       [,10]        [,11]        [,12]
```

```
## [1,] 1000000.00 100000000 10000000000 1.000000e+12 1.000000e+14
## [2,]   99090.91   9909091   990909091 9.909091e+10 9.909091e+12
## [3,]   98181.82   9818182   981818182 9.818182e+10 9.818182e+12
## [4,]   97272.73   9727273   972727273 9.727273e+10 9.727273e+12
## [5,]   96363.64   9636364   963636364 9.636364e+10 9.636364e+12
## [6,]   95454.55   9545455   954545455 9.545455e+10 9.545455e+12
##             [,13]        [,14]        [,15]        [,16]        [,17]
## [1,] 1.000000e+16 1.000000e+18 1.000000e+20 1.000000e+22 1.000000e+24
## [2,] 9.909091e+14 9.909091e+16 9.909091e+18 9.909091e+20 9.909091e+22
## [3,] 9.818182e+14 9.818182e+16 9.818182e+18 9.818182e+20 9.818182e+22
## [4,] 9.727273e+14 9.727273e+16 9.727273e+18 9.727273e+20 9.727273e+22
## [5,] 9.636364e+14 9.636364e+16 9.636364e+18 9.636364e+20 9.636364e+21
## [6,] 9.545455e+14 9.545455e+15 9.545455e+17 9.545455e+19 9.545455e+21
##             [,18]        [,19]        [,20]
## [1,] 1.000000e+26 1.000000e+28 1.000000e+30
## [2,] 9.909091e+24 9.909091e+26 9.909091e+28
## [3,] 9.818182e+24 9.818182e+26 9.818182e+28
## [4,] 9.727273e+24 9.727273e+26 9.727273e+28
## [5,] 9.636364e+23 9.636364e+25 9.636364e+27
## [6,] 9.545455e+23 9.545455e+25 9.545455e+27

#calculate condition  number
cond_num <- Lambda[1, ] / Lambda[100, ]
cond_num

##  [1] 1e+00 1e+00 1e+00 1e+01 1e+03 1e+05 1e+07 1e+09 1e+11 1e+13 1e+15
## [12] 1e+17 1e+19 1e+21 1e+23 1e+25 1e+27 1e+29 1e+31 1e+33

#for loop to calculate z*, lambda*, and plot(lambda, lambda*)
##calculate abs difference in lambda values
par(mfrow = c(2, 3))
Lambda_diff <- pd_check <- c()
for (i in 1:ncol(Lambda)) {

    #calcuate z* using generate Lambda's and Gamma from z
    z_star <- Gamma %*% diag(Lambda[ , i]) %*% t(Gamma)

    #eigen decomp z* and get lambda*
    Lambda_star <- eigen(z_star)$values

    #plot every fourth lambda vs lambda*
    if (i %in% c(1, 5, 10, 15, 20)) {
        plot(Lambda[ , i], Lambda_star,
         main = paste0("Cond.num: ", cond_num[i],
                       "\nMax(Lambda):", Lambda[1, i]))
        abline(a = 0, b = 1)
    } else {}

    #check P.D of z*
    pd_check <- c(pd_check,
                  if (sum(Lambda_star < .Machine$double.eps) > 0) {0} else {1}
    )

    Lambda_diff <- c(Lambda_diff,
```

```
                    mean(abs(Lambda[ , i] - Lambda_star)))
}

#combine results
result <- data.frame(cbind(pd_check, cond_num, Lambda[1, ], Lambda_diff))
colnames(result) <- c("Lambda*_i > 0", "Condition Num.", "Max(Lambda)", "avg(abs(Lambda - Lambda*))")

#print results table
print(result)
```

```
##    Lambda*_i > 0 Condition Num. Max(Lambda) avg(abs(Lambda - Lambda*))
## 1              1          1e+00     1.0e-02               5.030698e-17
## 2              1          1e+00     5.0e+00               2.517986e-14
## 3              1          1e+00     1.1e+01               5.547562e-14
## 4              1          1e+01     1.0e-02               2.469813e-18
## 5              1          1e+03     1.0e+00               3.750906e-17
## 6              1          1e+05     1.0e+02               3.427263e-15
## 7              1          1e+07     1.0e+04               3.402714e-13
## 8              1          1e+09     1.0e+06               3.256274e-11
## 9              1          1e+11     1.0e+08               3.370966e-09
## 10             1          1e+13     1.0e+10               4.383111e-07
## 11             1          1e+15     1.0e+12               3.268040e-05
## 12             1          1e+17     1.0e+14               3.530970e-03
## 13             0          1e+19     1.0e+16               3.486926e-01
## 14             0          1e+21     1.0e+18               1.717030e+01
## 15             0          1e+23     1.0e+20               1.735859e+03
## 16             0          1e+25     1.0e+22               2.821720e+05
## 17             0          1e+27     1.0e+24               2.063879e+07
## 18             0          1e+29     1.0e+26               4.690645e+09
## 19             0          1e+31     1.0e+28               3.821452e+11
## 20             0          1e+33     1.0e+30               2.056957e+13
```



At condition number $\geq 1e17$, the z* matrix is no longer positive definite.

The error in the estimated eigenvalues doesn't begin to be noticeable until condition number $\approx 1e19$ with $max(\lambda) \approx 1e16$. Until that point, the true and estimated eigenvalues are essentially the same with the $mean(|\Lambda - \Lambda *|)$ ranging from 1e-17 to 1e-5. Mean differenes less than machine epsilon are functionally 0.