Stat243: Problem Set 7

Worked with Jamie Palumbo, Mingyung Kim, Alanna Iverson

Sicun Huang

November 16, 2015

1. Reading Questions

(a) What are the goals of their simulation study and what are the metrics that they consider in assessing their method?

In this article, the authors presented an effective expectation-maximization (EM) test for testing the null hypothesis of arbitrary order m0 under a finite normal mixture model. To examine the performance of the test on finite sample, they conducted a number of simulation studies. The goals of the simulation study are to assess the accuracy of the asymptotic approximation in finite samples and to examine the power of the expectation-maximization (EM) test. The metrics they used to access the hypothesis testing method on the order of the normal mixture model are type I error and power of the EM test.

(b) What choices did the authors have to make in designing their simulation study? What are the key aspects of the data generating mechanism that likely affect the statistical power of the test?

They chose number of null models to be 12, sample sizes to be 200 and 400, number of repetitions to be 5000, and significance levels to be 5% and 1%. The power of the EM test increases as the sample size increases; also, it increases as the component means under the alternative models become far away from one another.

(c) Suggest some alternatives to how the authors designed their study. Are there data-generating scenarios that they did not consider that would be useful to consider?

The authors could increase the number of null models as well as sample sizes and replications to achieve higher precision. To do so efficiently, they could run the multiple experiments in parallel. The authors could also use a resampling approach when generating data to control for random differences between the datasets.

(d) Give some thoughts on how to set up a simulation study for their problem that uses principles of basic experimental design (see the Unit 10 notes) or if you think it would be difficult, say why.

In the paper, the authors followed the basic steps of a simulation study: specify sample size, distributions, parameters, statistic of interest, etc.; determine what inputs to vary; write code to carry out the individual experiment and return the quantity of interest; for each combination of inputs, repeat the experiment m times; summarize the results for each combination of interest, quantifying simulation uncertainty; and report the results in graphical or tabular form. The authors varied one input variable at a time. Although this makes results easy to interpret, it is inefficient. Instead, they could implement a fractional factorial design by carefully choosing which treatment combinations to omit. The goal is to achieve balance across the levels in a way that allows us to estimate lower level effects (in particular main effects) but not all high-order interactions.

(e) Do their figures/tables do a good job of presenting the simulation results and do you have any alternative suggestions for how to do this? Do the authors address the issue of simulation uncertainty/simulation standard errors and/or do they convince the reader they've done enough simulation replications?

The tables and figures presented results clearly. The simulated Type I errors figures showed that the observed levels were close to the significance levels that the authors were trying to achieve, particularly when K=3. However, it would make it more clear to the readers the accuracy of the results if the quartiles, minimum, maximum, and median values of the box and whisker plots were labeled. Comparing to the 5% significance level simulations, the values for the 1% significance level simulations seemed to be rather far from what was expected. While this may suggest that the sample sizes should be increased, it could also be due to the fact that the scales between these graphs are different. Keeping the scales constant would eliminate this problem. Although the results come out to be close to what was expected, this paper didnt totally convince me that there were enough simulation replications since the authors never explained their reasoning behind the choice of values when designing this simulation study. Also, the authors never addressed simulation standard errors.

(f) Interpret their tables on power (Tables 4 and 6) - do the results make sense in terms of how the power varies as a function of the data generating mechanism?

Overall, the power of the EM test increases with sample size. Table 4 showed that when the mixing proportions of the 3 models are equal, the power of EM test is greater. However, in the 4 component model, the opposite is true. In Table 6, we can observe that the test has greater power when the component means are further from one another. The above results make sense and are inline with the authors conclusions: as sample size increases, power of EM test increases; also, as the component means under the alternative models become further away from one another, the statistical power of the test increases.

(g) Discuss the extent to which they follow JASA's guidelines on simulation studies (see the end of the Unit 10 class notes for the JASA guidelines).

JASAs guidelines on simulation studies states: Papers reporting results based on computation should provide enough information so that readers can evaluate the quality of the results. Such information includes estimated accuracy of results, as well as descriptions of pseudorandom-number generators, numerical algorithms, computers, programming languages, and major software components that were used. In this study, the authors used R to implement the EM test and made the supplementary materials available online, which means that the results of this study should be reproducible by the reader. Although they did not elaborate on the details of the simulation process, the tables of simulated Type I errors and the powers of the test can help readers to evaluate the quality of their results.

2.

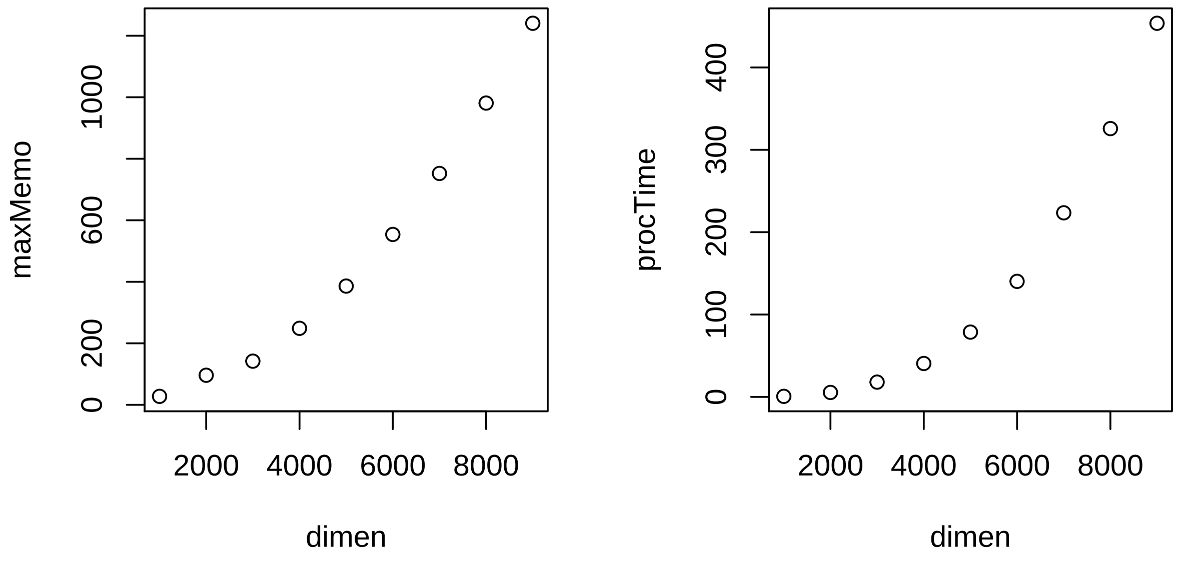
(b) As we can see from 2(a), the algorithm for computing the Cholesky upper triangular matrix U does not utilize the elements below diagonal in the original matrix; also, the ij^{th} element of the original matrix is only used to compute the ij^{th} element of U. So we can discard the elements below diagonal in the original matrix and overwrite the ij^{th} element of the original matrix with the ij^{th} element of U to save storage space.

(c) First, we consider the case where we overwrite the original matrix X with U, the upper triangular matrix in Cholesky decomposition.

```
library(pryr)

#function to compute cholesky decomposition for n by n matrix X; overwrite X with upper
#triangular matrix U
cholOverwrite <- function(i){
  n <- i*1000
  x <- crossprod( matrix( rnorm(n^2), n ) )
  x <- chol(x)
}

dimen <- seq(from=1000, to=9000, by=1000)
#initialize
maxMemo <- rep(0,9)
procTime <- rep(0,9)

#get maximum memory use and processing time separately so system.time function doesn't add
#to memory use
gc(reset = TRUE)

##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 372935 20.0     750400 40.1   372935 20.0
## Vcells 591012  4.6    1308461 10.0   591012  4.6

for (i in 1:9){
  gc(reset = TRUE)
  cholOverwrite(i)
  #2nd row 6th column of gc is max memory used in Mb
  maxMemo[i] <- gc()[2,6]
}

#3rd column of system.time is elapsed time
for (i in 1:9){
  gc(reset = TRUE)
  procTime[i] <- as.double( system.time(cholOverwrite(i))[3] )
}

dimen

## [1] 1000 2000 3000 4000 5000 6000 7000 8000 9000

maxMemo

## [1]   27.5   96.1  141.9  248.7  386.0  553.9  752.2  981.1 1240.5

procTime
```

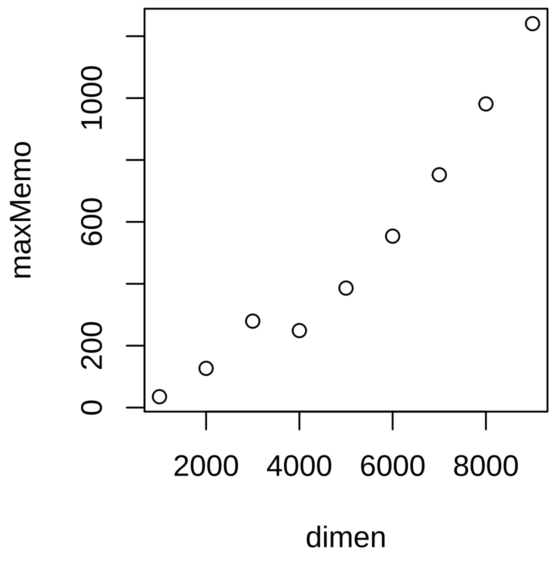
```
## [1]   0.652   5.380  17.935  40.557  78.663 140.254 223.501 325.760 453.631
```

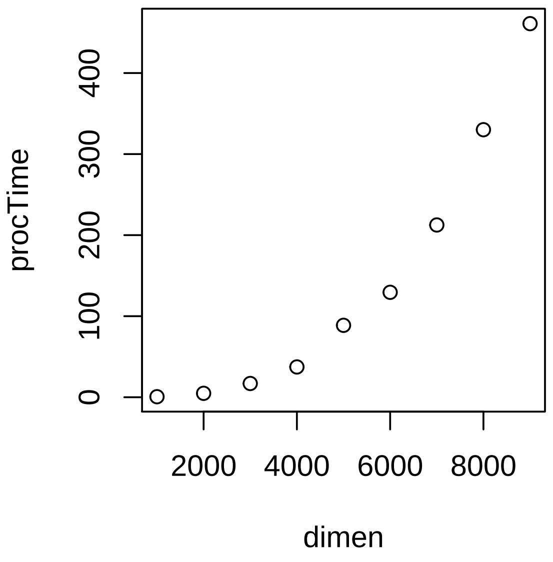
```
par(mfrow=c(1,2), pty="s")
plot(dimen,maxMemo)
plot(dimen,procTime)
```



Then, consider the case where we save the upper triangular matrix U into a new object, keeping both X and U.

```
rm(list=ls())

#function to compute cholesky decomposition for n by n matrix X; keep both X and upper
#triangular matrix U
cholKeep <- function(i){
  n <- i*1000
  x <- crossprod( matrix( rnorm(n^2),n ) )
  u <- chol(x)
}

dimen <- seq(from=1000, to=9000, by=1000)
#initialize
maxMemo <- rep(0,9)
procTime <- rep(0,9)

gc(reset = TRUE)

##          used (Mb) gc trigger   (Mb) max used (Mb)
## Ncells 380945 20.4     750400   40.1   380945 20.4
## Vcells 599820  4.6  187627703 1431.5   599820  4.6

for (i in 1:9){
  gc(reset = TRUE)
  cholKeep(i)
  maxMemo[i] <- gc()[2,6]
}

for (i in 1:9){
  gc(reset = TRUE)
  procTime[i] <- as.double( system.time(cholKeep(i))[3] )
}

dimen

## [1] 1000 2000 3000 4000 5000 6000 7000 8000 9000

maxMemo

## [1]   35.1  126.7  279.3  248.8  386.1  553.9  752.3  981.2 1240.6

procTime

## [1]   0.635   4.834  16.960  37.385  88.733 129.468 212.535 330.103 460.970

par(mfrow=c(1,2), pty="s")
plot(dimen,maxMemo)
plot(dimen,procTime)
```





As we can see from the plots, overwriting X with U significantly saves memory when dimension of the original matrix X is below 3000 by 3000; however, such obvious advantage vanishes when X gets larger. This result is partially consistent with the conclusion drawn in 2(b). Furthermore, memory use and processing time increase exponentially with n.

3. We compare the speed of solving $b = X^{-1}y$ by using (a) solve(X) followed by %*%, (b) solve(X,y), and (c) Cholesky decomposition followed by solving triangular systems, where X is a 5000 by 5000 matrix.

```
x <- crossprod( matrix( rnorm(5000^2), 5000 ) )
y <- matrix( rnorm(5000), ncol=1 )

#approach a
system.time(b1 <- solve(x)%*%y)

##    user  system elapsed
## 235.258   0.671 236.297

#approach b
system.time(b2 <- solve(x,y))

##    user  system elapsed
##  26.279   0.083  26.377

#approach c
system.time(b3 <- backsolve( chol(x), backsolve(chol(x),y,transpose=TRUE) ))

##    user  system elapsed
##  36.323   0.166  36.520
```

(a) The above result shows that elapsed time amongst the three method has $(a) > (c) > (b)$. However, their relative orderings are (a) $n^3 + n^2$ (b) $\frac{n^3}{3} + O(n^2)$ and (c) $\frac{n^3}{6} + O(n^2)$, which should give $(b) > (c)$ (We are given that the full inversion takes n^3 calculations). Interestingly, the above conclusion is not consistent with the order of computations we discussed in class.

(b)

```
all.equal(b1,b2,tolerance=.Machine$double.eps)

## [1] "Mean relative difference: 3.527693e-15"

all.equal(b1,b3,tolerance=.Machine$double.eps)

## [1] "Mean relative difference: 9.920121e-09"

all.equal(b2,b3,tolerance=.Machine$double.eps)

## [1] "Mean relative difference: 9.920121e-09"

max(abs(b1-b2),abs(b1-b3),abs(b2-b3))

## [1] 5.063931e-05

#compute condition number
norm(x)*norm(solve(x))

## [1] 67948425398
```

The results for the different methods are not the same numerically up to machine precision; however, they are fairly close. The maximum of the absolute values of the difference amongst the three β's tells us that in the worst case, the β's agree up to 5 digits. The condition number is the number of digits of accuracy lost during a computation relative to the precision of numbers on the computer. Having a condition number of approximately 10^{11} means that 11 digits of accuracy are lost during a computation. Hence the β's computed from the three different methods agree up to 5 decimal places, which supports our result.

4. Base on the above algorithm, we write gls function to compute the generalized least squares estimator. We do this by computing the Cholesky decomposition of $V = U^T U$, get $X' = (U^{-1})^T X$ and $Y' = (U^{-1})^T Y$, and then use the new X' and Y' in place of X and Y to calculate β by computing the QR decomposition of X' and performing a backsolve. To test the function, we construct n by p matrix X, n by n positive definite matrix V (sigma in the question), and n by 1 vector y. Here, we choose $n = 1000$ and $p = 100$.

```
gls <- function(x,y,v){
  u <- chol(v)
  x1 <- backsolve(u,x,transpose=TRUE)
  y1 <- backsolve(u,y,transpose=TRUE)
  b <- qr.solve(x1,y1)
  return(b)
}

x <- matrix( rnorm(1000*100), 1000 )
#make v positive definite
v <- crossprod( matrix( rnorm(1000^2), 1000 ) )
y <- matrix( rnorm(1000), ncol=1 )

head(gls(x,y,v))

##              [,1]
## [1,]  0.11665008
## [2,] -0.21130791
## [3,] -0.09719438
## [4,]  0.07530496
## [5,] -0.06823360
## [6,]  0.10885115
```