Stat243: Problem Set 6

Qingan Zhao
SID: 3033030808

1 Nov. 2017

Problem 1

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

  The goal of the simulation is to demostrate the likelihood ratio statistic is a weighted sum of indenpendent chi-squared asymptotial distribution. The use guassian mixture model in the simulation (one normal versus two components and two component versus three components). The metrics they consider is the powers of the particular tests using EM algorithm.

- 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? Are there data-generating scenarios that the authors did not consider that would be useful to consider?

  The choices they have to make are sample size, the distance between two components (D), the mixing coefficient and the number of components (k). The key aspects that are likely affect the power are the condition of the samples generated from the distribution and whether the samples are independent and identically distributed. Other scenarios which would be useful to consider are randomness and other distributions.

- Do their tables do a good job of presenting the simulation results and do you have any alternative suggestions for how to do this?

  Generally yes. The results are from top to bottom, and they controlled well in the simulation. I would say it might be more clear if using figures with comparison. Also, the accuracy of the results and be presented more clearly by introducing statistics.

- Interpret their tables on power (Tables 2 and 4) - do the results make sense in terms of how the power varies as a function of the data generating mechanism?

  Table 2 demostrates the single gussian distribution under null hypothesis. Sampling the same distribution to see if the results match and show the convergence rate. The results only make sense when D is one or two, and the sample size n is small. there is no strong evidence that the power depends on the mix proportion when D equals to 3.

- How do you think the authors decided to use 1000 simulations. Would 10 simulations be enough? How might we decide if 1000 simulations is enough?

Generally the more simulations, the better distribution, since the error will have much more impacts on the results if there are two few simulations. Hence, I would say 10 simulations are not enough. However, the number of simulations cannot be too large to operate. So we should choose an optimal number, which maybe the reason why the author decided to choose 1000. To decide if 1000 is enough, we could, for example, choose 100-2000 simulations with the stepsize of 100, and compute the error and difference to work it out.

Problem 2

This problem is to find the number of unique users who have asked only R-related questions and no Python-related questions using SQL. To get the result, first find out the columns of the related table (which are *questions* and *questions_tags*) in the database. Then join these two tables, extract the unique users based on the *ownerid*, and extract those who only asked R questions by excluding the users of Python questions. Finally, count the number of the extracted rows.

```
#connect to the database
drv <- dbDriver("SQLite")
dir <- "~/Desktop"
dbFilename <- 'stackoverflow-2016.db'
db <- dbConnect(drv, dbname = file.path(dir, dbFilename))

#check tables in the database
dbListTables(db)

## [1] "answers"          "maxRepByQuestion" "ownertag"
## [4] "questions"        "questionsAugment" "questions_tags"
## [7] "users"

#check the columns of the related tables
dbListFields(db, 'questions')

## [1] "questionid"   "creationdate" "score"        "viewcount"
## [5] "title"        "ownerid"

dbListFields(db, 'questions_tags')

## [1] "questionid" "tag"

#extract rows with unique users who have asked only R-related questions
#and no Python-related questions
uni_user <- dbGetQuery(db,"SELECT DISTINCT ownerid FROM(
                       SELECT questions.ownerid, questions_tags.tag FROM questions
                       JOIN questions_tags
                       ON questions.questionid = questions_tags.questionid)
                       WHERE tag = 'r' EXCEPT SELECT DISTINCT ownerid FROM(
                       SELECT questions.ownerid, questions_tags.tag FROM questions
                       JOIN questions_tags
                       ON questions.questionid = questions_tags.questionid)
                       WHERE tag = 'python'")

#count the number of the extracted rows, which is the result
length(unlist(uni_user))

## [1] 18611
```

Hence, **18611** unique users have asked only R-related questions and no Python-related questions.

Problem 3

Since the period of the given Wikipedia traffic data is October to December 2008 and there are many holidays in such a period. So my question in this problem is related to a specific holiday which is Thanksgiving Day. The question is that **whether there are more sites about "turkey" being viewed on Thanksgiving Day than some other days and how much are the difference**. Hence, let's focus on several dates: 10/27/2018, 11/15/2008, 11/27/2008 (Thanksgiving), 12/15/2008 and 12/27/2008.

```
#activate Spark on Savio using srun
srun -A ic_stat243 -p savio --nodes=2 -t 02:00:00 --pty bash
module load java spark
source /global/home/groups/allhands/bin/spark_helper.sh
spark-start
env | grep SPARK
module unload python
pyspark --master $SPARK_URL --executor-memory 60G
```

```
#set the first date 10/27/2008
date_interested = "20081027"

#this function filters to sites related to 'turkey' viewed on a certain date
def find(line, regex_date = date_interested, regex_word = "turkey", language = None):
vals = line.split(' ')
if len(vals) < 6:
return(False)
tmp = re.search(regex_date, vals[0]) is None or (re.search(regex_word, vals[3]) is None)
if tmp or (language != None and vals[2] != language):
return(False)
else:
return(True)

#count the number of sites
first_date = lines.filter(find).repartition(96)
first_date.count()

#result output
## 1302

#count the number of sites viewed on 11/15/2008
date_interested = "20081115"
second_date = lines.filter(find).repartition(96)
second_date.count()

#result output
## 1444

#count the number of sites viewed on 11/27/2008 (Thanksgiving)
date_interested = "20081127"
second_date = lines.filter(find).repartition(96)
second_date.count()
```

```
#result output
## 2714

#count the number of sites viewed on 12/15/2008
date_interested = "20081215"
second_date = lines.filter(find).repartition(96)
second_date.count()

#result output
## 1414

#count the number of sites viewed on 12/27/2008
date_interested = "20081227"
second_date = lines.filter(find).repartition(96)
second_date.count()

#result output
## 1473
```

Now let's plot the result:

```
dates <- c('1027', '1115', '1127', '1215', '1227')
observations <- c(1302, 1444, 2714, 1414, 1473)
plot(observations, xaxt='n', xlab='Date', ylab='Observations', type='b')
axis(1, at=1:5, labels=dates)
```

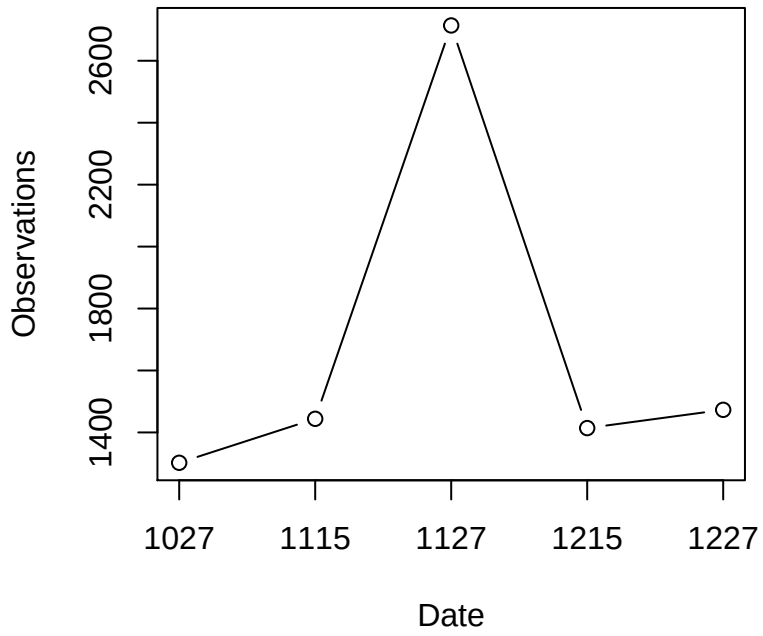

Fig.1: Observation results on five dates

As seen in Figure 1, the number of sites on 11/27 (Thanksgiving) is much more than those on other dates, which makes sense because people may pay more attention on how to cook their turkey better during the Thanksgiving day.

Problem 4

This problem is to analysis the same data as in Problem 3 on *Savio* using parallelization, and compare the effectiveness using parallelized R versus using *PySpark*.

(a)

In this part, I used *foreach* on a single Savio node. I ran the code on a quarter of the files (240) and assume the time scales accordingly. I used *sbatch* in the problem with an R script and a shell script. The results are collected into a single data frame.

Note that the *readr* package is installed first and the rest packages are loaded.

```
#!/bin/bash
# Job name:
#SBATCH --job-name=prob4a
#
# Account:
#SBATCH --account=ic_stat243
#
# Partition:
#SBATCH --partition=savio
#
#Number of tasks needed for use case:
#SBATCH --nodes=4
#
# Processors per task:
#SBATCH --cpus-per-task=1
#
# Wall clock limit (30 seconds here):
#SBATCH --time=02:00:00
#
## Command(s) to run:
R CMD BATCH --no-save prob4a.R prob4a.out
```

```
#this is the R script submitted to Savio together with the output in .output file#
library(parallel)
library(doParallel)
library(foreach)
library(stringr)
library(readr)
library(dplyr)

#access the number of the available cores
nCores <- as.numeric(Sys.getenv('SLURM_CPUS_ON_NODE'))
registerDoParallel(nCores)

#code on a quarter of the files
```

```
n <- 240

#filter the rows and collect the results into a single data frame
result <- foreach(i = 1:n,
                  .combine = rbind,
                  .verbose = TRUE) %dopar% {
                    #get the file names
                    data <- paste(
                      '/global/scratch/paciorek/wikistats_full/dated_for_R/part-00',
                      str_pad(i, 3, pad=0), sep='')
                    #read the file
                    tmp <- readr::read_delim(data, ' ', quote='', col_names=FALSE)
                    #filter the rows
                    output <- tmp %>% dplyr::filter(
                      stringr::str_detect(tmp$X4,'Barack_Obama'))
                    output
                  }

#print part of the result
#since the screen can only display 3 columns, I chose to display the 4th column
#instead of the 3rd because the 4th column contains "Barack Obama" for better view
print(result[1:5,c(1,2,4)])

#result output (in the .out file)
##        X1     X2                                                          X4
##     (int)  (chr)                                                       (chr)
##1 20081213 070000                                        Image:Barack_Obama.jpg
##2 20081105 170001                              2008_Barack_Obama_assassination_plot
##3 20081214 020000 List_of_Barack_Obama_presidential_campaign_endorsements,_2008
##4 20081214 120001                                          Barack_Obama,_Sr.
##5 20081106 070000     Image:Barack_Obama_and_supporters_5,_February_4,_2008.jpg
```

(b)

The *.out* file in Problem 4a has recorded the time to do the filtering as follow:

```
> proc.time()
##     user    system   elapsed
## 7603.518 31052.551  2202.074
```

Since I run the code on a quarter of the files and assuming scalability can de perfectly achieved, the time I used would be the same as the time spending in filtering the data in all files if I use 4 nodes. Hence, let's compare the elapsed time (2202.074 *seconds* $\approx$ 37 *minutes*) with PySpark (15 *minutes* provided in the problem).

It turns out parallelized R is less efficient than PySpark from the perspective of my experiment, and I think part of the reason is that my code is not efficicent enough.

Problem 5

(a)

Divide the operations into 3 parts:

$$U_{11} = \sqrt{A_{11}} \tag{1}$$

$$U_{1j} = \frac{A_{1j}}{U_{11}} \quad (j = 2, 3, ..., n) \tag{2}$$

$$operations\_1 = n - 1 \tag{3}$$

$$U_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} U_{ki}^2} \quad (i = 2, 3, ..., n) \tag{4}$$

$$operations\_2 = 1 + 2 + 3 + ... + (n-1) = \frac{n(n-1)}{2} \tag{5}$$

$$U_{ij} = \frac{A_{ij} - \sum_{k=1}^{i-1} U_{ki} U_{kj})}{U_{ii}} \tag{6}$$

$$operations\_3 = \sum_{i=2}^{n} (n-i)I = \frac{n^3 - 7n + 6}{6} \tag{7}$$

Hence, the total operation count is:

$$\begin{aligned} operations &= operations\_1 + operations\_2 + operations\_3 \\ &= (n-1) + \frac{n(n-1)}{2} + \frac{n^3 - 7n + 6}{6} = \frac{n^3 + 3n^2 - 4n}{6} \end{aligned} \tag{8}$$

(b)

I think we can store the Cholesky upper triangular matrix U in the storage space that is used for the original matrix, and does not have to overwrite anything needed later. As the 3 decomposition steps (parts) indicated in Problem 5a, for Step 1 and 2, they are independent and can be stored at the same time. As for Step 3, when it comes to row i, we only operate on the rows above (i.e., 1,2,...,i-1). The calculations can be done before $U_i j$ is computed. Hence, nothing needs to be overwritten for later use.