Apriori Frequent Itemsets Generations

MA Mingyu 14110562D

2/28/2017

COMP4433 Assignment 1

derek.ma@connect.polyu.hk derek.ma

This R markdown file can be found at https://github.com/derekmma/data-mining-algorithms/blob/master/apriori_simulation.Rmd

Set Up and Prepare the Database

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)
library(norm)

#Read the data from a .csv file
transactions <- read.csv("~/GoogleDrive/_DM/3_Homework/transactions.csv", stringsAsFactors = FALSE)
colnames(transactions) <- c ("id","item") #change column names

#input min_sup and min_conf provided in the question
min_sup <- 0.17
min_conf <- 0.8

#filter out all maintenance record
maintenance <- filter(transactions, item == "Maintenance")
transactions <- filter(transactions, item != "Maintenance")

#calculate minimum support count
min_sup_count <- min_sup * (max(transactions$id) - length(maintenance$id))
```

Prepare for Iterations

Write scan function and transform current data to `transactions_byID` which shown below:

```r
#Scan function: input-a itemset; output-how many times this itemset appears in the transaction record
scan <- function(itemList){
  index <- 0
  for (i in 1:nrow(transactions_byID)){
    exist <- TRUE
    for (j in 1:length(itemList)){
      if (transactions_byID[i,itemList[j]] == FALSE){
        exist <- FALSE
      }
    }
    if (exist == TRUE) {
      index <- index + 1
    }
  }
  index
}

#Another form of transactions grouped by ID.
transactions_byID <- data.frame(id=1:max(transactions$id),
                                Case=rep(FALSE,max(transactions$id)),
                                Desktop=rep(FALSE,max(transactions$id)),
                                DisplayCard=rep(FALSE,max(transactions$id)),
                                Mouse=rep(FALSE,max(transactions$id)),
                                Speaker=rep(FALSE,max(transactions$id)))

for (i in 1:nrow(transactions)){
  transactions_byID[transactions[i,1],transactions[i,2]] <- TRUE
}
transactions_byID
```

```
##    id  Case Desktop DisplayCard Mouse Speaker
## 1   1 FALSE   FALSE       FALSE FALSE    TRUE
## 2   2 FALSE    TRUE        TRUE FALSE   FALSE
## 3   3  TRUE   FALSE       FALSE FALSE    TRUE
## 4   4 FALSE   FALSE       FALSE FALSE    TRUE
## 5   5 FALSE   FALSE        TRUE  TRUE   FALSE
## 6   6 FALSE   FALSE       FALSE FALSE   FALSE
## 7   7 FALSE    TRUE       FALSE  TRUE   FALSE
## 8   8  TRUE   FALSE       FALSE FALSE   FALSE
## 9   9 FALSE    TRUE       FALSE FALSE   FALSE
## 10 10 FALSE   FALSE       FALSE  TRUE   FALSE
## 11 11 FALSE   FALSE       FALSE FALSE   FALSE
## 12 12  TRUE   FALSE       FALSE FALSE   FALSE
## 13 13  TRUE    TRUE        TRUE  TRUE    TRUE
## 14 14 FALSE   FALSE        TRUE FALSE   FALSE
## 15 15 FALSE   FALSE       FALSE FALSE   FALSE
## 16 16 FALSE   FALSE       FALSE  TRUE   FALSE
## 17 17 FALSE   FALSE       FALSE FALSE   FALSE
## 18 18 FALSE   FALSE        TRUE FALSE   FALSE
## 19 19  TRUE    TRUE        TRUE  TRUE    TRUE
## 20 20 FALSE   FALSE       FALSE FALSE   FALSE
## 21 21  TRUE   FALSE       FALSE FALSE   FALSE
## 22 22  TRUE   FALSE        TRUE FALSE   FALSE
## 23 23 FALSE    TRUE        TRUE  TRUE    TRUE
```

```
## 24 24 FALSE    TRUE        TRUE FALSE   FALSE
## 25 25  TRUE    TRUE        TRUE  TRUE    TRUE
## 26 26  TRUE   FALSE        TRUE FALSE    TRUE
## 27 27  TRUE    TRUE        TRUE FALSE   FALSE
## 28 28  TRUE    TRUE        TRUE  TRUE    TRUE
## 29 29 FALSE    TRUE       FALSE FALSE   FALSE
## 30 30 FALSE   FALSE        TRUE FALSE   FALSE
## 31 31 FALSE   FALSE        TRUE FALSE   FALSE
## 32 32 FALSE   FALSE       FALSE FALSE   FALSE
## 33 33  TRUE   FALSE       FALSE FALSE   FALSE
## 34 34 FALSE   FALSE       FALSE FALSE   FALSE
## 35 35  TRUE   FALSE       FALSE FALSE   FALSE
## 36 36 FALSE   FALSE        TRUE FALSE   FALSE
## 37 37  TRUE    TRUE       FALSE  TRUE   FALSE
## 38 38  TRUE    TRUE       FALSE FALSE   FALSE
## 39 39 FALSE   FALSE       FALSE FALSE   FALSE
## 40 40 FALSE    TRUE       FALSE FALSE   FALSE
## 41 41  TRUE   FALSE       FALSE FALSE   FALSE
## 42 42 FALSE    TRUE       FALSE FALSE   FALSE
## 43 43 FALSE   FALSE       FALSE FALSE   FALSE
## 44 44  TRUE   FALSE       FALSE FALSE   FALSE
## 45 45 FALSE   FALSE        TRUE FALSE   FALSE
## 46 46 FALSE    TRUE        TRUE  TRUE   FALSE
## 47 47  TRUE   FALSE       FALSE FALSE   FALSE
## 48 48 FALSE   FALSE       FALSE FALSE   FALSE
## 49 49  TRUE   FALSE        TRUE FALSE    TRUE
## 50 50  TRUE    TRUE        TRUE  TRUE    TRUE
```

First-Round

```
#C1
c1 <- summarize(group_by(transactions, item),sup = n())
l1 <- filter(c1, sup > min_sup_count)
```

C1:

```
c1
```

```
## # A tibble: 5 <U+00D7> 2
##          item   sup
##         <chr> <int>
## 1        Case    20
## 2     Desktop    17
## 3 DisplayCard    20
## 4       Mouse    12
## 5     Speaker    11
```

In this case, all itemsets' support count is larger than minimum support count requirement, thus L1 is:

```
l1
```

```
## # A tibble: 5 <U+00D7> 2
##          item   sup
##         <chr> <int>
## 1        Case    20
```

```
## 2     Desktop    17
## 3 DisplayCard    20
## 4       Mouse    12
## 5     Speaker    11
```

Second-Round

Self-crosing find the new itemsets in C2, because all itemsets in C1 is under requirement, so no need to pruning:

```
#C2
c2 <- data.frame()
n1 <- 0
for (itm1 in l1$item){
  n1 <- n1 + 1
  n2 <- 0
  for (itm2 in l1$item){
    n2 <- n2 + 1
    if (itm1 != itm2 && n2 > n1){
      temp <- data.frame(item1 = itm1, item2 = itm2, sup = scan(c(itm1,itm2)))
      c2 <- rbind(c2, temp)
    }
  }
}

c2 <- c2 %>%
  mutate(sup_dist = sup - min_sup_count)
c2
```

```
##          item1       item2 sup sup_dist
## 1         Case     Desktop   8      1.2
## 2         Case DisplayCard   9      2.2
## 3         Case       Mouse   6     -0.8
## 4         Case     Speaker   8      1.2
## 5      Desktop DisplayCard  10      3.2
## 6      Desktop       Mouse   9      2.2
## 7      Desktop     Speaker   6     -0.8
## 8  DisplayCard       Mouse   8      1.2
## 9  DisplayCard     Speaker   8      1.2
## 10       Mouse     Speaker   6     -0.8
```

In this case, we can found some of the records in C2 have low support count, so we should delete them from C2 to produce L2:

```
l2 <- filter(c2, sup_dist >= 0) %>%
  select(item1,item2,sup)
l2
```

```
##         item1       item2 sup
## 1        Case     Desktop   8
## 2        Case DisplayCard   9
## 3        Case     Speaker   8
## 4     Desktop DisplayCard  10
## 5     Desktop       Mouse   9
## 6 DisplayCard       Mouse   8
```

```
## 7 DisplayCard     Speaker   8
```

Third Round

First create candidates based on L2:

```
c3 <- data.frame()
for (i1 in 1:nrow(l2)){
  for (i2 in 1:nrow(l2)){
    if (i2 > i1){
      uni <- union(c(as.character(l2[i1,1]),as.character(l2[i1,2])),
                   c(as.character(l2[i2,1]),as.character(l2[i2,2])))
      if (length(uni) == 3) {
        stop <- FALSE
        for (m in 1:nrow(c3)){
          if (setequal(uni, c(as.character(c3[m,1]), as.character(c3[m,2]), as.character(c3[m,3])))) == 
            stop <- TRUE
          }
        }
        if (stop == FALSE) {
          temp <- data.frame(item1 = uni[1], item2 = uni[2], item3 = uni[3], sup = scan(uni))
          c3 <- rbind(c3, temp)
        }
      }
    }
  }
}
c3 %>% select(item1,item2,item3)
```

```
##         item1       item2       item3
## 1        Case     Desktop DisplayCard
## 2        Case     Desktop     Speaker
## 3        Case     Desktop       Mouse
## 4        Case DisplayCard     Speaker
## 5        Case DisplayCard       Mouse
## 6     Desktop DisplayCard       Mouse
## 7     Desktop DisplayCard     Speaker
## 8 DisplayCard       Mouse     Speaker
```

In this table, No. 2, 3, 5, 7, 8 contain the itemset with low support in C2, so we delete them and calculate the support:

```
c3 <- c3 %>%
  mutate(sup_dist = sup - min_sup_count) %>%
  slice(c(1,4,6))
c3
```

```
##     item1       item2       item3 sup sup_dist
## 1    Case     Desktop DisplayCard   6     -0.8
## 2    Case DisplayCard     Speaker   7      0.2
## 3 Desktop DisplayCard       Mouse   7      0.2
```

Then we delete the itemsets with low support and get L3:

```r
l3 <- filter(c3, sup_dist >= 0) %>%
  select(item1,item2,item3,sup)
l3
```

```
##     item1       item2   item3 sup
## 1    Case DisplayCard Speaker   7
## 2 Desktop DisplayCard   Mouse   7
```

Fourth Round

Will this round be the final round? Create candinates based on L3: Then I found the C4 is empty because all 4-item itemsets are pruned. The algorithm terminated. All frequent itemsets are found in L1, L2 and L3.

Result

```r
l1
```

```
## # A tibble: 5 <U+00D7> 2
##          item   sup
##         <chr> <int>
## 1        Case    20
## 2     Desktop    17
## 3 DisplayCard    20
## 4       Mouse    12
## 5     Speaker    11
```

```r
l2
```

```
##         item1       item2 sup
## 1        Case     Desktop   8
## 2        Case DisplayCard   9
## 3        Case     Speaker   8
## 4     Desktop DisplayCard  10
## 5     Desktop       Mouse   9
## 6 DisplayCard       Mouse   8
## 7 DisplayCard     Speaker   8
```

```r
l3
```

```
##     item1       item2   item3 sup
## 1    Case DisplayCard Speaker   7
## 2 Desktop DisplayCard   Mouse   7
```