实验报告

141220132 银琦 141220132@smail.nju.edu.cn

一． 实验要求

By varying different level of support and confidence, compare Apriori, FPGrowth and a dummy the baseline method that conducting exhaustive search for frequent itemsets, in terms of the number of generated frequent itemsets, the storage used for mining the rules, the computational cost (in second)

Try to discover some interesting association rules using Apriori or FPGrowth, make discussion on the insights the rules disclose.

二． 问题及数据集描述

给定数据集，挖掘出数据集中的频繁项集，通过调整支持度与置信度便可得出关联规则。

给定了两个数据集，第一个是杂货店一个月的交易记录，每行代表一条交易；第二个数据集是 8 个用户在 9 台计算机的 Unix 系统上两年内使用的指令，每组指令都由**SOF**和**EOF**分隔开。

三． 方法描述

本次实验共比较了三种方法对数据的处理：Apriori 算法、FPGrowth 算法和蛮力算法。

1. Apriori 算法

   算法使用频繁项集性质的先验知识，通过连接产生候选选项及其支持度，然后通过剪枝生成频繁项集。对于该种算法及给定的数据集，在 Weka 中跑不出结果，因为消耗大量的内存和时间，所以在 matlab 中手动实现。

2. FPGrowth 算法

   针对 Apriori 算法固有的多次扫描事务数据集的缺陷，首先将代表频繁项集的数据库压缩到一颗频繁模式树，该树扔保留项集的关联信息，然后把这种压缩后的数据库划分成一组条件数据库，每个数据库关联一个频繁项或“模式段”，并分别挖掘每个条件数据库，对于每个“模式片段”只需要考察与它相关联数据集。对于该种算法及给定数据集，在 Weka 中可得出第一个数据集的关联规则，未手动实现。

3. 蛮力算法

   未实现。

四． 方法的比较

1. Apriori 算法

   用 Java 语言对数据集做了处理，将原数据集文件中的数据提取出，存入 txt 文件

中，数据集一格式如下：

```
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
whole milk
pip fruit,yogurt,cream cheese ,meat spreads
other vegetables,whole milk,condensed milk,long life bakery product
whole milk,butter,yogurt,rice,abrasive cleaner
rolls/buns
other vegetables,UHT-milk,rolls/buns,bottled beer,liquor (appetizer)
pot plants
whole milk,cereals
tropical fruit,other vegetables,white bread,bottled water,chocolate
citrus fruit,tropical fruit,whole milk,butter,curd,yogurt,flour,bottled water,dishes
beef
frankfurter,rolls/buns,soda
chicken,tropical fruit
butter,sugar,fruit/vegetable juice,newspapers
fruit/vegetable juice
packaged fruit/vegetables
chocolate
specialty bar
other vegetables
butter milk,pastry
whole milk
tropical fruit,cream cheese ,processed cheese,detergent,newspapers
tropical fruit,root vegetables,other vegetables,frozen dessert,rolls/buns,flour,sweet spreads,salty snack,waffles,candy,bathroom cleaner
bottled water,canned beer
yogurt
sausage,rolls/buns,soda,chocolate
other vegetables
brown bread,soda,fruit/vegetable juice,canned beer,newspapers,shopping bags
yogurt,beverages,bottled water,specialty bar
hamburger meat,other vegetables,rolls/buns,spices,bottled water,hygiene articles,napkins
root vegetables,other vegetables,whole milk,beverages,sugar
pork,berries,other vegetables,whole milk,whipped/sour cream,artif. sweetener,soda,abrasive cleaner
beef,grapes,detergent
pastry,soda
fruit/vegetable juice
canned beer
root vegetables,other vegetables,whole milk,dessert
citrus fruit,zwieback,newspapers
sausage,rolls/buns,soda,canned beer,specialty bar,shopping bags
tropical fruit,root vegetables,whole milk,yogurt,domestic eggs,brown bread,pastry,sugar,cereals,coffee,soda,waffles,candy
berries,yogurt
canned beer
```

数据集二格式如下：

```
whoami, pwd, ls, dir, vi, source, source, exit
whereis, mkdir, vi, vi, ls, source
elm, log
elm, man, fg, finger, |, more, finger, fg
elm, logout
elm, cd, ls, vi, fg, exit
elm, logout
elm, exit
elm, log
cd, cd, ls, vi, vi, exit
elm, exit
lwinfo, lwinfo, lwinfo, lwinfo, lwinfo, lwinfo, exit
elm
elm, exit
elm, cd, ls, vi, fg, vi, fg, cd, elm, finger, finger, |, more, fg, finger, finger, |, more, finger, |, more, fg, finger, |, more, fg
ls, more, exit
elm, finger, fg, finger, fg, finger, fg, finger, finger, finger, |, more, fg, finger, |, more, date, fg, elm, ls, fg, ls, emacs, ls, ls, emacs, ls, cat, rm
cd, ls, vi, exit
ls, cd, ls, mkdir, cd, ls, vi, vi, exit
exit
ls, uptime, exit
ruptime, exit
elm, vi, more, fg, finger, fg, ls, cd, ls, cd, ls, cd, vi, ls, more, ls, ls, -a, cd, ls, ls, -R, |, grep, finger, talk, fg, exit
whoami, ls, uptime, netscape
who, exit
elm, uptime, elm, r
setenv, ls, more, rm, ls, cd, ls, cd, ls, maker, ruptime
elm, finger, ruptime, ruptime, |, more, home, fg, elm
elm, finger, finger, fg, finger, finger, finger
elm, logout
ls, cd, ls, more, elm, exit
elm, exit
elm, ls, cd, ls, vi, ls, wc, ls, wc, mv, ls, rm, exit, cd, fg, exit
ls, ls, cd, ls, vi, ls, mv, mv, ll, cd, whereis, <GENSYM:0>/vi, whereis, vi, exit
cd, cd, ls, more, ls, vi, <GENSYM:0>/vi, ls, rm, exit
elm, ls, ll, fg, elm, exit
cd, ls, vi, cls, man, cls, ls, vi, ls
write, exit
setenv, cd, ls, maker, ls, cd, ls, cd, ls, cd, ls, ghostview, ls, cd, ls, cd, ls, man, ls, lp, -dmsl21, -n15, lpq, lpq, -dmsl21, lpq, -Pmsl21, lpq, -dmsl21
elm, log
cancel, -u, cancel, -u, ls, cd, lp, -dmsl21, cancel, -u, lp, -d, cancel, -u, lwifo, lwinfo, ls, lwinfo, lwinfo, back, back, cd, ls, vi, ls, <GENSYM:0>/vi,
setenv, cd, ls, vi, ls, vi, ls, vi, ls, lwinfo, lwinfo, lwinfo, lwinfo, ls, cancel, -u, lp, -dmsl22, lp, -dmsl22, lpq, -Pmsl22, lpq, -Pmsl23, cancel, -u, l
elm, lwinfo, lwinfo, lwinfo, lwinfo, lwinfo, lpq, -Pmsl23, cancel, -u, cancel, -uclin, lpq, -Pmsl22, lpq, -Pmsl23, cancel, -u, cancel, -u, man, ls, lpq, -P
setenv, ls, cd, maker, ls, ls, ls, ls, vi
```

在 matlab 中读入文件，将其转化成 01 矩阵，用 Apriori 算法得出关联规则，一开始设置支持度为 0.001，置信度为 0.9，但是运行时间过长，没有得出结果，之后将支持度改为 0.01，置信度改为 0.5，得出了结果。

2. FPGrowth 算法

用 Java 语言对数据集做了处理，将原数据集文件中的数据提取出，存入 arff 文件中，数据集一格式如下：

```
@attribute 'mayonnaise' {0,1}
@attribute 'rubbing alcohol' {0,1}
@attribute 'syrup' {0,1}
@attribute 'liver loaf' {0,1}
@attribute 'baby cosmetics' {0,1}
@attribute 'organic products' {0,1}
@attribute 'nut snack' {0,1}
@attribute 'kitchen towels' {0,1}
@attribute 'frozen chicken' {0,1}
@attribute 'light bulbs' {0,1}
@attribute 'ketchup' {0,1}
@attribute 'jam' {0,1}
@attribute 'decalcifier' {0,1}
@attribute 'nuts/prunes' {0,1}
@attribute 'liqueur' {0,1}
@attribute 'organic sausage' {0,1}
@attribute 'cream' {0,1}
@attribute 'toilet cleaner' {0,1}
@attribute 'specialty vegetables' {0,1}
@attribute 'baby food' {0,1}
@attribute 'pudding powder' {0,1}
@attribute 'tidbits' {0,1}
@attribute 'whisky' {0,1}
@attribute 'frozen fruits' {0,1}
@attribute 'bags' {0,1}
@attribute 'cooking chocolate' {0,1}
@attribute 'sound storage medium' {0,1}
@attribute 'kitchen utensil' {0,1}
@attribute 'preservation products' {0,1}
@data
1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

数据集二格式如下：

```
@attribute -type {0,1}
@attribute lastcomm {0,1}
@attribute <GENSYM:1>/split.pl {0,1}
@attribute nfrom {0,1}
@attribute <GENSYM:1>/thud {0,1}
@attribute <GENSYM:40>/ {0,1}
@attribute <GENSYM:1>/testit.pl {0,1}
@attribute mn {0,1}
@attribute chs_parser {0,1}
@attribute scan {0,1}
@attribute <GENSYM:41>/talk {0,1}
@attribute 5<GENSYM:42> {0,1}
@attribute stats.pl {0,1}
@attribute php {0,1}
@attribute df-k {0,1}
@attribute drift_test.pl {0,1}
@attribute <GENSYM:12>/csh_parser {0,1}
@attribute -50037 {0,1}
@attribute memory {0,1}
@attribute <GENSYM:43> {0,1}
@attribute '-1,' {0,1}
@attribute 'from,' {0,1}
@attribute wnewmail {0,1}
@attribute <GENSYM:11>/tr {0,1}
@attribute olbiff {0,1}
@attribute end {0,1}
@attribute rehast {0,1}
@attribute tok_count.sh {0,1}
@attribute printf {0,1}
@attribute -guxawwww {0,1}
@attribute <GENSYM:1>/build {0,1}
@attribute kll {0,1}

@data
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

用 Weka 打开 arff 文件，选择 FPGrowth 算法进行挖掘，对数据集一分别测试了支持度为 0.001，置信度为 0.9 和支持度为 0.01，置信度为 0.5 的情况并得出结果，对数据集二仅测试了支持度为 0.01，置信度为 0.5 的情况并得出结果。

五． 结果分析

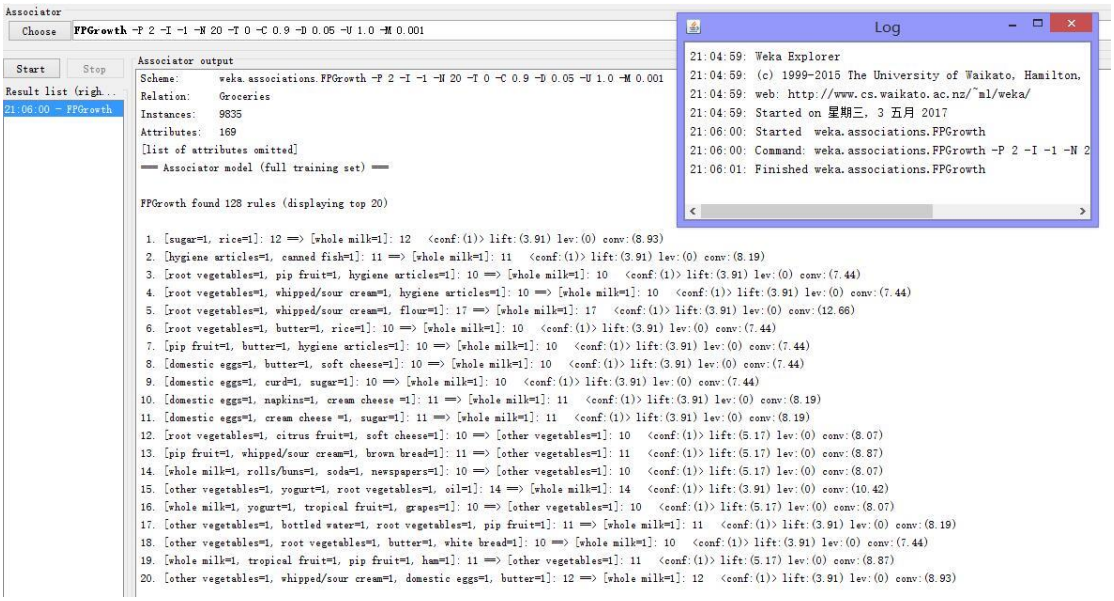
1. Apriori 算法

```
>> apriori
关联规则算法完成，规则数为：14
存储规则到文件‘rules.txt’完成
Apriori算法挖掘菜品订单关联规则完成！
用时：

ans =

   35.9590
```

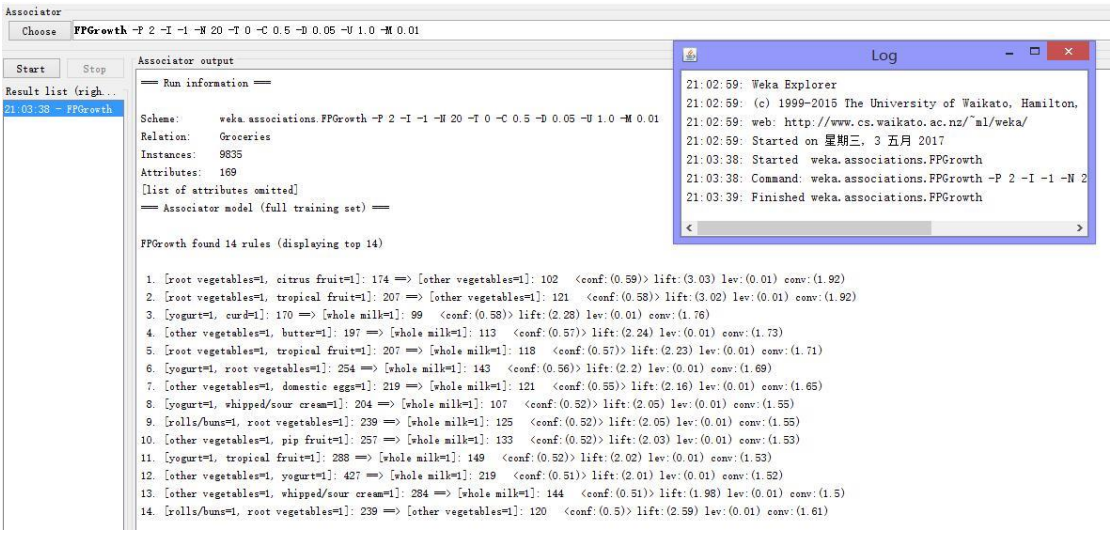
```
>> apriori
关联规则算法完成，规则数为：100
存储规则到文件‘rules2.txt’完成
Apriori算法挖掘菜品订单关联规则完成！
用时：

ans =

   88.8780
```

由上图可以得出，当支持度为 0.01，置信度为 0.5 时，该算法对于数据集一可以得出 14 条关联规则，耗时大约 36 秒，对数据集二可得出 100 条规则，耗时大约 89 秒。

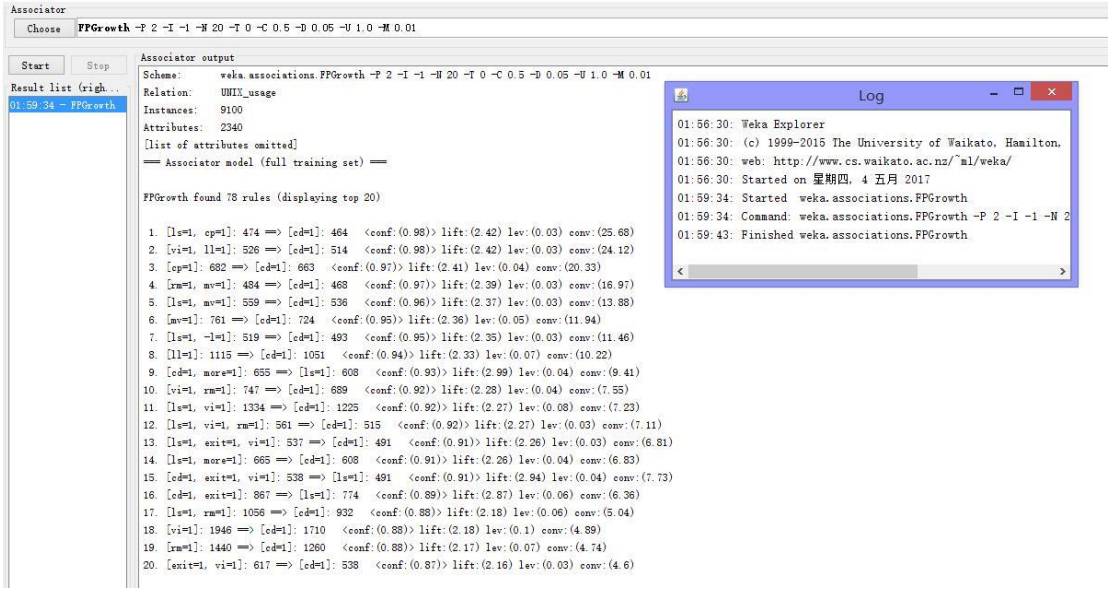
2. FPGrowth 算法



由上图可以得出，当支持度为 0.01，置信度为 0.5 时，该算法对数据集一可以得出 14 条关联规则，耗时大约 1-2 秒；



由上图可知，当支持度为 0.001，置信度为 0.9 时，该算法对数据集一可以得出 128 条关联规则，耗时大约 1-2 秒。



由上图可知，当支持度为 0.01，置信度为 0.5 时，该算法对数据集二可得出 78 条关联规则，耗时大约 9 秒。分析与 Apriori 算法不一致的原因可能是在处理数据时出现一条乱码，于是手动将乱码删除，可能导致数据集不一致。

具体的规则详见附件中的输出文件。

六． 结论

在理论上，Apriori 算法比 FPGrowth 算法要消耗更多的时间与空间，因为 Apriori 算法在每一步产生侯选项目集时循环产生的组合过多，没有排除不应该参与组合的元素，耗费大量空间；每次计算项集的支持度时，都对数据集中的全部记录进行了一遍扫描比较，耗费大量时间。而 FPGrowth 不使用候选集而且只进行两次数据集扫描。这两种算法与蛮力算法相比都有提升。

在实验中得出的结论也的确如此。处理好数据集后，一开始使用 Weka 进行计算，但是对于给定的两个数据集，Apriori 算法长时间没有得出结果，而 FPGrowth 算法则是只需较短时间便得出结果，后来在 matlab 中手动实现了 Apriori 算法，然而当支持度为 0.001，置信度为 0.5 时，两个数据集仍然均得不出结果，增大支持度到 0.01，减小置信度到 0.5

后才得出结果，并且耗时远大于 FPGrowth 算法的耗时。

七． 参考文献

1. http://blog.csdn.net/lfdanding/article/details/50755919
2. http://www.it165.net/pro/html/201502/33715.html
3. 数据挖掘概念与技术 原书第三版