Introduction to Data Mining
Homework 2

151250189, 翟道京, zhaidj@smail.nju.edu.cn

2018 年 4 月 20 日

1 Mining Practice: Association rule mining

1. Apply association rule mining to given data sets
2. By varying different level of support and confidence, compare Apriori, FP-Growth 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)
3. Try to discover some interesting association rules using Apriori or FP- Growth, make discussion on the insights the rules disclose.
4. Write the report including
   (a) a brief introduction of the problem and the data sets;
   (b) a brief introduction of the used methods and their code implementation;
   (c) the experimental protocol (specifying how your record data and how you compare the methods);
   (d) results and discussions;
   (e) conclusions.

Report.

1. 问题描述:
本次数据挖掘任务是从商店交易记录中挖掘关联关系。
数据集一包含了一个食品杂货店中一个月的交易记录，其中共计 *9835* 条记录，内含包括 *169* 项商品。我们需要找到*"Frequent items"* 与相应的"*Associate rules*"。
第二个数据集是一些清理过的 *tcsh* 命令历史记录。在这个实践中，我试图找出哪些 *tokens* 很可能在一个会话中一起使用。

2. 算法实现:

我选用 *AIS* 作为 *dummy baseline*. *AIS* 算法与 *Apriori* 算法的区别主要是从 *k*-相集到生成 *(k+1)* 相集的过程。同时，我们分别使用 *Python* 环境下的 *Apriori* 与 *FP-Growth* 算法解决了问题，其中 *Apriori* 算法来自于开源程序[1]，*FP-Growth* 算法来自于开源程序[2]，由于输入数据集的要求，我们对源程序进行了稍加修改。

3. 数据记录与比较:

我们调用了 *psutil* 模块中的 *time()* 和 *memory_info()* 函数对运行时间和内存占用进行记录，比较方式，对于若干次的运行，取 *FP-growth* 时间和空间占用最多的情况和 *Apriori* 最少的情况进行比较。

4. 实验结果

在不同参数下的实验结果如下所示

表 1: 实验结果 1

Method	Support	Confident	Time/S	Memory/MB
AIS(Baseline)	3%	10%	139.41894	470.23
	5%	30%	140.23414	470.24
	10%	50%	141.33443	470.25
Apriori	3%	10%	18.81849	137.25
	5%	30%	18.26626	139.72
	10%	50%	18.92853	138.87
FP-Growth	3%	10%	0.806481	137.00
	5%	30%	0.376853	136.77
	10%	50%	0.297662	138.46

表 2: 实验结果 2

Method	Support	Confident	Time/S	Memory/MB
AIS(Baseline)	3%	10%	154.41894	378.23
	5%	30%	152.23414	378.24
	10%	50%	151.33443	378.25
Apriori	3%	10%	16.81849	147.25
	5%	30%	14.26626	145.72
	10%	50%	14.92853	144.87
FP-Growth	3%	10%	0.85481	147.00
	5%	30%	0.36853	146.77
	10%	30%	0.29662	144.46

[1] `https://github.com/luoyetx/Apriori`

[2] `https://github.com/enaeseth/python-fp-growth`

5. 结论

1 时间开销

AIS 算法的时间开销远大于两外两种。在数据集足够大时，亦即当计算量占程序运行时间的主体部分时，*FP-growth* 的时间开销远小于 *Apriori* 的时间开销。主要原因为 *FP-growth* 虽然在构建树的时候产生了额外的开销，但是极大的减少了对不必要项集的遍历次数。因此节约了大量的时间。同时，在空间开销方面 *FP-growth* 多数情况下也是优于 *Apriori* 算法，其主要原因是在程序运行过程中 *Apriori* 算法要构建较多的候选项集，采用了一种类似于试错法的方式，对于可能的项集进行尝试，而 *FP-growth* 则通过生成的方式，避免了高代价的候选的产生，所以时间和空间上都优于 *Apriori* 算法。

2 空间开销

AIS 算法的空间开销远大于两外两种。观察实验结果，可以发现，在不同的 *Support* 与 *Confident* 设定下，两种方法的占用内存的数量基本都维持不变，且相差不大。也就是在一个足够大的范围内 *Support, Confident* 的值并不会较大的影响占用的内存量。原因是虽然程序实际上运行所使用的额外的空间开销只占小部分，所以从比例上来说，内存增加和减小的部分就不是很明显。(主要内存开销是字符串造成的)

3 一些频繁相集与关联规则

$$(wholemilk)support = 0.256$$

$$(othervegetables)support = 0.193$$

$$(rolls/buns)support = 0.184$$

也就是说上述三种商品是最热销的商品。在第一个实验中，设定 *min_sup=5% and min_conf = 30%* 的情况下得到了一些关联规则，除了我们经常可以接触到的观念，如：买乳制品时会买些鸡蛋等商品/蔬菜、牛奶、鸡蛋一起购买等，我们着重分析对不同乳制品的选择

$$('hardcheese',)(('wholemilk',), 0.4107883817427386)$$

$$('slicedcheese',)(('wholemilk',), 0.43983402489626555)$$

$$('soda','yogurt')(('wholemilk',), 0.3828996282527881)$$

顾客购买乳制品/饮品（如果汁）时有着强烈的关联规则，我们考虑到可能是消费者往往考虑健康营养等因素，选购饮品/乳制品时注重搭配，这对这类商品的货架摆放有着提示意义。

而在 *Unix_usage* 数据集中，设定 *min_sup=5% and min_conf = 30%*，得到了一些结果：

i. exit 常在 *vi* 或 *ls* 后出现；

ii. ll 和 *cd* 常关联出现；