《数据挖掘导论》Assignment 2

Introduction to the problem and the data sets

刘潇远 161220083 liuxy@smail.nju.edu.cn

一、问题及数据集介绍

本次实验着眼于关联规则挖掘，希望在数据集中找出满足支持度和置信度的关联(共现)规则。实验在两个数据集上进行挖掘，第一个数据集 GroceryStore 记录了一段时间内，某杂货店顾客消费情况，每条数据都是某位顾客在一次结账时购买的全部物品的集合；第二个数据集 UNIX_usage 记录了 9 位学生在使用 UNIX 时输入的命令行指令，每条记录代表在一个命令行窗口从打开到关闭输入的所有命令，每条记录以***SOF***开头，每一行记录一个命令，之后以***EOF***结尾

实验一共实现了三个方法，分别是 naïve 方法，Apriori 方法和 FPGrowth 方法。其中，naïve 方法是没有进行“剪枝”的 Apriori 方法，其余两种方法完全按照课本伪代码进行实现，原理简单讲解清楚，在此不做赘述。

2、方法实现

软件入口放置在 Main.java 中，运行时根据控制台引导，分别进行：输入支持度，输入置信度(以小数形式)，数据集选择，挖掘方法选择。输入完毕后程序执行挖掘，将频繁项集和关联规则分别输出到 frequentSet.txt 文件和 rules.txt 文件，并在控制台输出找出频繁项集所用时间，程序结束。

3、结果

注：naïve 方法运行时间过长，因此并没有等到他运行结束，因此只记录 Apriori 和 FPGrowth 方法的运行时间

方法	数据集	时间

Apriori	Grocery(支持度 100，置信度 0.4)	2.7354 秒
	UNIX_usage(仅 USER0，支持度 20，置信度 0.4)	6.9879 秒
FPGrowth	Grocery(支持度 100，置信度 0.4)	1.3799 秒
	UNIX_usage(仅 USER0，支持度 20，置信度 0.4)	5.3277 秒

4、结论

由此可以得出，在运行速度上 FPGrowth 优于 Apriori 远优于 naïve 方法。这是因为 Apriori 在每次 k 频繁项集挖掘时进行了"剪枝"，去掉了子集为非频繁项集的 k 候选频繁项集，但是每一步仍然产生大量候选频繁项集，需要对其进行剪枝。而 FPGrowth 更进一步，直接利用 FP 树的结构，消除了 Apriori 方法的缺点，因此速度更快。