

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： 校交1801**

**学 号： U201814132**

**姓 名： 张浚哲**

**指导教师： 王蔚**

**报告日期： 2020年12月30日**

**计算机科学与技术学院**

**目录**

[实验三 关系挖掘实验 1](#_Toc58793746)

[**3.1 实验内容** 1](#_Toc58793747)

[**3.2 实验过程** 1](#_Toc58793748)

[3.2.1 编程思路 1](#_Toc58793749)

[3.2.2 遇到的问题及解决方式 1](#_Toc58793750)

[3.2.3 实验测试与结果分析 1](#_Toc58793751)

[**3.3 实验总结** 2](#_Toc58793752)

# 实验三 关系挖掘实验

## **3.1 实验内容**

**必做：**

1. 实验内容

编程实现Apriori算法，要求使用给定的数据文件进行实验，获得频繁项集以及关联规则。

2. 实验要求

以Groceries.csv作为输入文件

输出1~3阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数

固定参数以方便检查，频繁项集的最小支持度为0.005，关联规则的最小置信度为0.5

**加分项：**

1. 实验内容

在Apriori算法的基础上，要求使用pcy或pcy的几种变式multiHash、multiStage等算法对二阶频繁项集的计算阶段进行优化。

2. 实验要求

以Groceries.csv作为输入文件

输出1~4阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数

输出pcy或pcy变式算法中的vector的值，以bit位的形式输出

 参数不变，频繁项集的最小支持度为0.005，关联规则的最小置信度为0.5

## **3.2 实验过程**

### 3.2.1 编程思路

本实验的程序分成：

1.候选频繁k项集产生函数

2.筛选函数

3. 规则产生函数

4. pcy设计

具体解释如下：

1. 候选频繁k项集产生函数：

Ck\_make

入口参数：set\_dict（上一阶的频繁k项集），frozen\_set\_data（整个数据集的数据），k\_times（当前处理的阶数），bitmap=None（传入的bitmap 默认为None），basket=None（传入的bitmap对应的通的值，默认为None）

返回值：生成的这一阶的候选频繁k项集

功能：通过上一阶的频繁k项集，构造并计算这一阶的频繁k项集。

实现：对set\_dict中的每一个候选频繁集的两两组合进行一次遍历，set\_dict是一个字典，k为对应的frozenset（上一阶的候选频繁集），v对应的是对应候选频繁集出现的次数。对set\_dict中的两两组合进行筛选，如果其组合后的长度不等于此阶数，则不要，否则则记录下来，并将其加入到字典中，并将其初始值设置为0.然后遍历整个数据集froze\_set\_data对forze\_set\_data中的每一行来说，遍历当前的字典，只要当前字典中有集合在此行中，就将对应集合的计数加1.这样将froze\_set\_data全部遍历一遍之后就能得到对应下一阶的候选频繁k项集了。

1. 筛选函数：

filtr

入口参数：Ck\_dict（候选频繁k项集），all\_line\_num（所有记录的数目）

返回值：返回筛选之后的频繁k项集。

功能：将Ck\_dict（候选频繁k项集）按照设置的支持度进行筛选，得到对应阶数的频繁k项集。

实现：遍历候选频繁k项集，分别比较器其计数和所有记录数乘以支持度的大小，如果大于则保留，否则丢弃。这样筛选候选频繁k项集得到的是对应阶数的频繁k项集。

1. 规则产生函数

rule\_make\_all函数

入口参数：C\_list（对应1~k阶的频繁项集组成的列表），k\_times（生成的规则的阶数-1）

返回值：产生的k阶规则列表。形式为[[(b,c), (a)], p]意思为(b,c)->a这个规则的置信度是p。

功能：根据前几项的频繁项集产生对应的规则。

实现：首先，将阶数最高的那个频繁项集设为主要处理集，然后对主要处理集中的每一个频繁集项进行子集切分，找到对应集合中的所有可能的真子集，然后对于每个子集获取其长度，其长度-1即为对应的阶数频繁项集对应存储列表的下标位置。通过长度-1定位到对应的频繁项集的字典，然后获取其中的计数，然后将特定频繁项除以定位的对应的频繁项集得到概率p。如果这个概率大于置信概率，则将其加入对应的规则列表中。

1. pcy设计：

本次的pcy要求是对二阶频繁项集的计算进行优化，因此bitmap过程只设计在了产生二阶频繁项集的函数中。

设计32个bitvector，每个bitvector为32位，定义hash函数Hash1(i, j)：(i\*j)%1024.在第一次遍历数据的时候，对其中每一个字典进行编号，然后对每一条数据，产生其每一种2项集的组合，根据其二项集中对应单词的编号进行hash得到对应桶的下标，然后将对应下标的桶进行计数。然后遍历对应1024个桶，如果1024个桶中对应的通的个数比支持度\*总记录数要小，就不变，否则将对应位置的bit\_vector的位置为1.在产生候选频繁k项集的时候，将得到的bit\_map传入，并将之前对单词的索引字典也传入。在加入了bit\_map后的候选频繁k项集的产生函数时，在之前的对长度要求的基础上加上对bit\_map对应位置为1的条件的要求，只有对应两个集合编号的hash值对应的bit\_map值为1的时候才进行操作。

### 遇到的问题及解决方式

1. 候选频繁集的产生函数，这个函数是最基础最重要的一个函数，首先要解决对应k阶候选频繁集的产生问题，我们可以利用频繁k项集的一个性质，就是其所有子集均为频繁集，因此我们可以让k-1项集的集合两两组合，可以这样理解，对于一个k项频繁集，其k-1项子集一定为频繁集，而我们拥有所有的k-1项频繁集，便可用所有的k-1项频繁集两两组合，因为只要一个k项集是频繁集，那么其对应的k-1项频繁集一定在我们找到的所有k-1项频繁集中，因此是一定能组合出来的。使用这样的规则，可以降低程序运行的时间。
2. 频繁集及候选频繁集的数据结构的设计

由于需要经常性的使用到集合操作，又经常需要根据对应的集合找到对应集合的数目，因此，这里使用了字典的数据结构，其中字典的k为对应的集合，是frozenset的类型，v是对应集合出现的次数。

1. 规则产生函数

难点主要是为了定位对应规则的频繁项集的数目，因为一个长的频繁项集可能会产生多个长度不同的规则。比如(a, b, c, d)可能会产生(a, b, c)->d或者(a, b)->(c, d)。因此对于k阶的频繁集，需要传入1~k阶的频繁集才能确定能够生成对应的规则。其中对应阶数的规则使用频繁项集的长度来定位。

1. 子集产生函数

需要对一个集合产生不重复的所有真子集。这里采用的是将对应的元素编号，然后不重复的遍历两边列表中的元素，获得其对应的真子集元素编号组成，然后再通过编号映射回来，得到对应的所有二项真子集。

### 实验测试与结果分析

1. 首先观察对应加载的数据集，froze\_set\_data，其数据信息如图1.1所示：

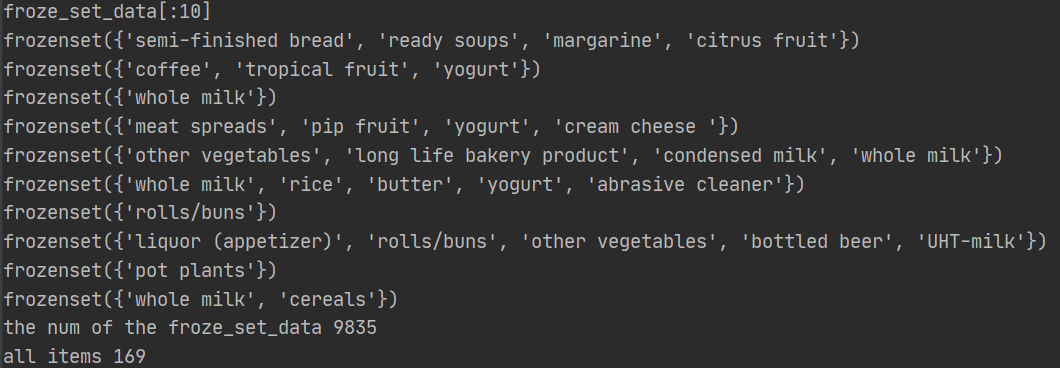


图1.1

可见，一共有9835条记录，所有的项目一共有169种，froze\_set\_data的数据中，每一此购物记录是由一共frozenset记录下来的。

1. 在产生1阶频繁集的时候，进行bitvector的生成，最后出来的对应的bitmap和频繁1项集合如图1.2和1.3所示：

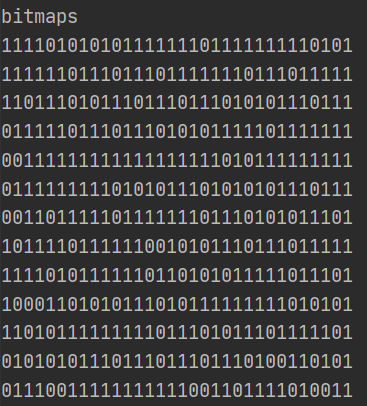


图1.2

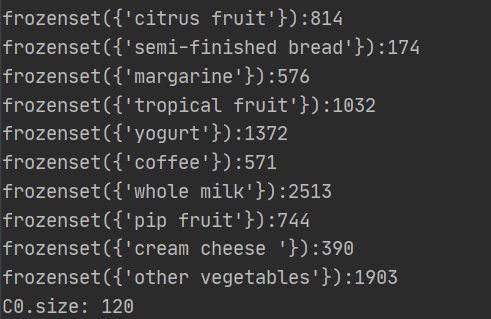


图1.3

可以看到当使用的hash桶的数目达到了1024个的时候，就会出现一定比例的达不到支持度的桶，这样就能达到降维的一部分目的。对应的频繁k项集的数据结构如图1.3所示。此处生成的bitvector可以用在二阶频繁子集的生成里。1阶频繁集的个数为120.

1. 使用候选频繁k项集产生函数产生的2项候选频繁集如图1.4所示：

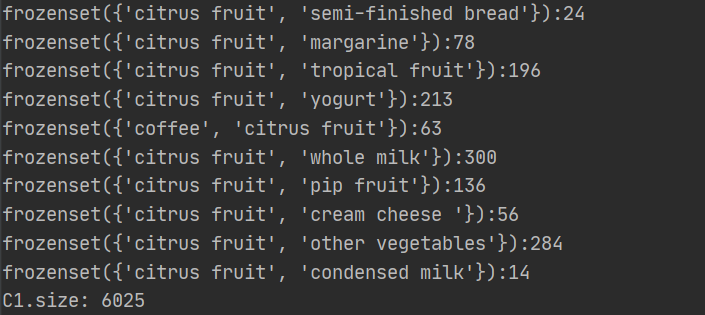


图1.4

一共是9835条记录，支持度为0.005时对应最少的条数应该是49.175因此上面的候选2项集应该会筛选掉一部分。此时的二项候选频繁集的个数是6025.

4．使用filtr函数将2项候选频繁集筛选成2项频繁集后，如图1.5所示：

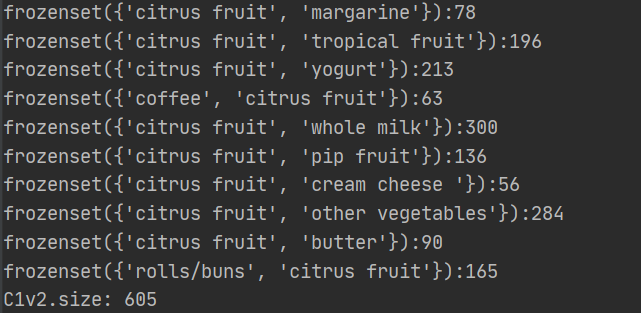


图1.5

可以看到相较于图1.4，凡是对应数量比49.175小的2项集都被去除。可以看到，现在2项频繁集的个数605远比候选集的个数小。

同理运行3阶4阶的候选频繁集产生函数和频繁集筛选函数得到的3阶频繁集合4阶频繁集的个数如图1.6所示：

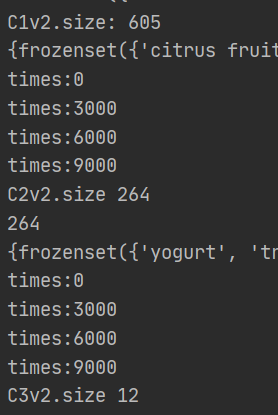


图1.6

1. 利用规则产生函数分别得到对应阶数下的规则如图1.7,1.8,1.9所示：



图1.7

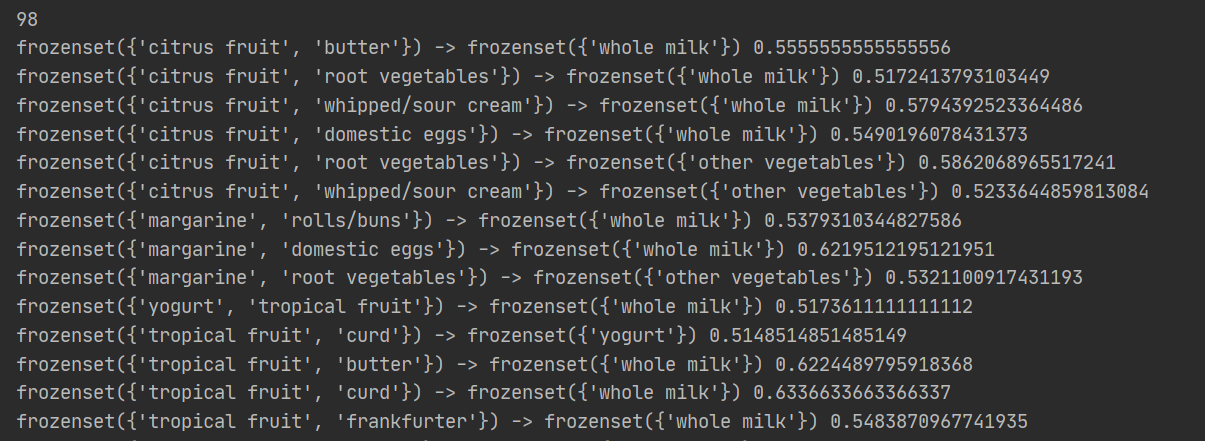


图1.8

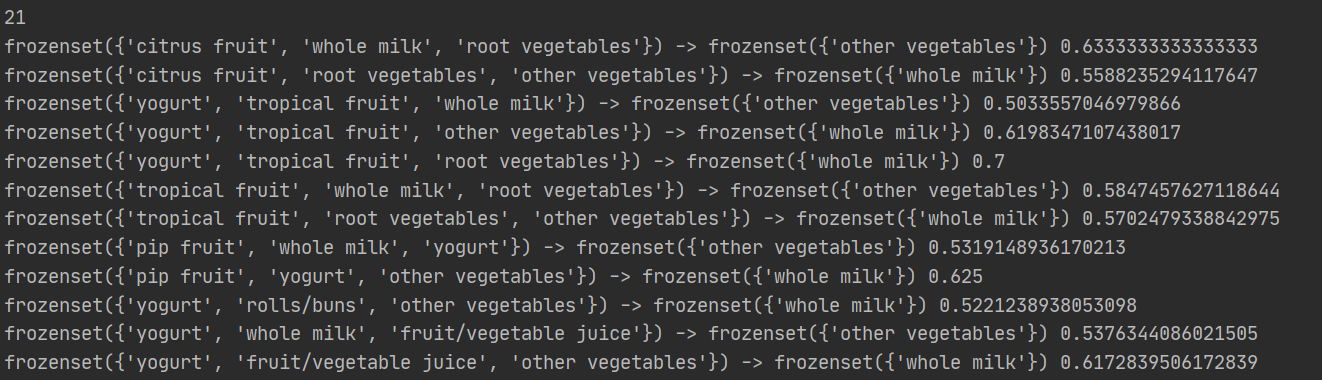


图1.9

输出的格式是(x, y, z)->e p 表示(x, y, z)退出e的置信度为p，本次实验的置信度为0.5，所以可以看到这里的置信度大于0.5才能被定义为规则。

分别以2阶，3阶，4阶为对应的规则产生源，分别产生了1,98,21条规则，符合结果1,99,120.

## **3.3 实验总结**

个人认为本系列实验中，此次实验的难度最高，首先需要对关联关系挖掘算法有一定的熟悉，其次，需要对算法进行一定的优化，否则算法执行一遍的时间过长，在检查和测试的时候都不方便，需要使用一些频繁集的性质来加快程序运行的速度，并且还要考虑周全，不能漏掉其中的一些频繁集，并且需要按阶数进行计算和统计，不可打乱顺序。同时还要注意pcy算法的设计，注意hash桶数目的设计，桶数目多的时候，空桶的数目会很多，这样会造成一些不必要的浪费，而哈希桶的数目少的时候，空桶的数目就比较少，这样就达不到pcy的效果，同时还要设计性质比较好的hash函数，不然容易将相邻的映射到同一个桶。同时对于规则的产生也是需要注意的，对于高阶频繁集可能会产生同一阶的不同条规则，也有可能产生不同阶的不同规则。因此，对应产生规则的结束不确定就需要将之前的所有的频繁集的计数都传入函数，根据对应规则的阶数（长度）在支持度字典列表中定位其支持度存在的字典，然后找到其支持度，然后想除和置信度进行比较，大于则产生对应的规则。