

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： ACM1901**

**学 号： U201914965**

**姓 名： 卫云泽**

**指导教师： 崔金华**

**报告日期： 2021年12月21日**

**计算机科学与技术学院**

**目录**

[实验三 关系挖掘实验 1](#_Toc58398474)

[**3.1 实验内容** 1](#_Toc58398475)

[**3.2 实验过程** 1](#_Toc58398476)

[3.2.1 编程思路 1](#_Toc58398477)

[3.2.2 遇到的问题及解决方式 3](#_Toc58398478)

[3.2.3 实验测试与结果分析 4](#_Toc58398479)

[**3.3 实验总结** 6](#_Toc58398480)

# 实验三 关系挖掘实验

## **3.1 实验内容**

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

## **3.2 实验过程**

### 3.2.1 编程思路

**数据集输入与预处理**

实验中使用的数据集为Groceries.csv，该数据集包含9835个“购物车”，每个“购物车”是一个花括号包围的商品集合，不同商品之间以“,”分割。由于购物车信息是以字符串的形式出现，故数据集输入后首先需要对字符串进行合理分割，提取购物车中商品信息存入需要的数据结构中。实验中使用list存放购物车集合，其中每个购物车是一个包含若干商品字符串的set；使用了一个set存放所有出现过的商品。数据集的输入和预处理部分代码如下，其中第5行中的[1：-1]实现了“购物车”描述字符串两端花括号的去除。

1. g\_list = []     # 所有购物车的集合
2. g\_set = set()   # 所有商品的集合
3. **for** i **in** range(0, g\_size):
4. groceryi = groceries['items'][i][1:-1].split(",")
5. gi\_set = set()
6. **for** item **in** groceryi:
7. gi\_set.add(item)
8. g\_set.add(item)
9. g\_list.append(gi\_set)

**初始化1项集候选集列表**

基于Apriori算法，在求解每一阶频繁项集之前，都需要根据上一阶频繁项集的结果构造候选集。由于1阶频繁项集是第一次计算，故所有商品的列表即为1阶频繁项集候选集列表。遍历商品集合g\_set中元素并将其加入k项集中间结果列表中。本部分代码实现较为简单，故不再单独展示。

**求频繁项集**

实验中使用for循环依次求出1~3阶频繁项集。求频繁项集的过程即遍历当前候选项集，对于候选项集中的每一项，遍历购物车，统计该项在购物车中出现的频数，用该频数除以购物车总数即可得到该项置信度，将获得的置信度和设定的最小置信度进行比较，若不小于最小值置信度，则该项是频繁项，可加入结果列表中。求k阶频繁项集的核心代码如下，其中列表new\_k\_item\_sets存放本次结果，用于后续下一阶频繁项集候选集的生成；列表k\_item\_sets\_res[k]存放k阶频繁项集的最终结果。

1. new\_k\_item\_sets = []    # 初始化本次结果列表
2. k\_item\_sets\_res[k] = [] # 初始化最终结果列表
3. **for** item\_set **in** k\_item\_sets:
4. count = 0
5. **for** grocery **in** g\_list:  # 遍历购物车，统计商品在购物车中的数目
6. isIn = True
7. **for** item **in** item\_set:
8. **if** item **not** **in** grocery:
9. isIn = False
10. **break**
11. **if** isIn:
12. count +=1
13. support = count / g\_size  # 求置信度，g\_size是购物车总数
14. **if** support >= minimum\_support:
15. new\_k\_item\_sets.append(item\_set)
16. k\_item\_sets\_res[k].append((item\_set, support))

**求关联规则**

在求频繁项集的循环中，当求出二阶及以上频繁项集后，可根据本次频繁项集结果求出相应关联规则，即对于每个频繁项集，依次去掉该项集中的某个商品项，获得一个差集，统计该差集在购物车中出现的次数count\_set\_j，再结合求频繁项集中获得的该频繁项集出现次数count求出当前关联规则的置信度conf = count / count\_set\_j，若置信度不小于阈值，则关联规则可被记录。求关联规则的核心代码如下。

1. **if** k >= 2:
2. **for** item **in** item\_set:
3. count\_set\_j = 0  # 统计不含某item的子集的计数
4. item\_set\_j = item\_set.copy()
5. item\_set\_j.remove(item)  # item\_set\_j是item\_set去掉item的差集
6. **for** grocery **in** g\_list:
7. isIn = True
8. **for** it **in** item\_set\_j:
9. **if** it **not** **in** grocery:
10. isIn = False
11. **break**
12. **if** isIn:
13. count\_set\_j += 1
14. conf = count / count\_set\_j  # 计算置信度，其中count是求频繁项集的结果
15. **if** conf >= minimum\_confidence:
16. association\_rules.append((item\_set\_j, item, conf))

**生成k+1项集候选集**

在每一轮执行循环结束之前，需要根据当前已生成的k项集生成k+1项集的候选集。若直接对候选集和商品集合求笛卡尔积，很容易出现大量重复集合，实验中使用了将候选项转换为有序字符串，结合集合set实现去重，set2string、string2set等函数具体实现细节见3.2.2节。生成k+1项候选集的具体实现如下。

1. k\_item\_sets.clear() # 新候选项集
2. k\_item\_sets\_candidates = set() # 存放候选集对应字符串，以去重
3. **for** item\_set **in** new\_k\_item\_sets:
4. **for** item **in** g\_set:
5. **if** item **not** **in** item\_set:
6. new\_item\_set = item\_set.copy()
7. new\_item\_set.add(item)  # 执行笛卡尔积
8. k\_item\_sets\_candidates.add(set2string(new\_item\_set))  # 字符串去重
9. **for** item\_set **in** k\_item\_sets\_candidates:
10. k\_item\_sets.append(string2set(item\_set))  # 获得去重后的候选集

### 3.2.2 遇到的问题及解决方式

**候选集出现大量重复集合问题：**

初版代码编写完毕后，在测试中发现候选集出现大量重复元素，以至于候选集数目呈指数级增长，无法正确获得后续结果。该问题是由于生成高阶候选集时直接对低阶结果和商品集合做笛卡尔积的缘故，候选集中会出现大量重复集合（包括顺序不同、元素相同的集合），重复的集合是应该被去除的。

**解决方式：**

实验中使用字符串+集合的方式实现去重，编写了集合转换为字符串的函数set2string和字符串转换成集合的函数string2set实现集合和有序字符串的相互转换，用于去重过程。两个函数的具体实现如下。

1. # 集合转有序字符串 #
2. **def** set2string(set):
3. tmp = []
4. res = ""
5. **for** item **in** set:
6. tmp.append(item)
7. tmp.sort()
8. **for** item **in** tmp:
9. res = res + item + ","
10. **return** res[:-1]
11. # 字符串转集合 #
12. **def** string2set(str):
13. res = set()
14. tmp = str.split(",")
15. **for** item **in** tmp:
16. **if** len(item) != 0:
17. res.add(item)
18. **return** res

set2string函数将输入的集合中的商品名（字符串）按字母序排序，再以“,”作为分隔符连接成字符串，形成商品集合的有序字符串，保证了含有相同商品的集合生成相同的字符串；string2set函数则是将这个字符串转换成集合的形式供之后后续频繁项集和关联规则挖掘过程使用。

生成无重复候选集的过程首先将每一次新生成的候选集转换为上述有序字符串，加入set中实现去重，最后遍历获得的候选集set，将有序字符串形式的候选集转换回集合形式，即可实现无重复候选集的生成。

**关联规则缺失：**

在解决了候选集出现大量重复集合问题后，程序可以正常运行并获得1~3阶频繁项集和关联规则的结果，但统计发现关联规则仅求得92项，相比之下参考结果的关联规则为99项。

**解决方式：**

审查缺失的关联规则，发现缺失的关联规则置信度均为0.5，猜测是边界条件出现问题。重新审查代码中的判断条件，发现初版代码中的最小值支持度和最小置信度均使用“>”运算符进行比较，实际应为“不小于”，故将“>”改为“>=”即可获得正确的频繁项集和关联规则。

### 3.2.3 实验测试与结果分析

实验结果中的L1频繁项集部分展示如图3-1所示。

文本

描述已自动生成

图3-1 L1频繁项集展示（部分）

实验结果中的L2频繁项集部分展示如图3-2所示。

文本

描述已自动生成

图3-2 L2频繁项集展示（部分）

实验结果中的L3频繁项集部分展示如图3-3所示。

文本

描述已自动生成

图3-3 L3频繁项集展示（部分）

实验结果中的关联规则部分展示如图3-4所示。

文本

描述已自动生成

图3-4 关联规则展示（部分）

经与参考结果比对，实验结果正确地求出了所有L1~L3频繁项集（及对应支持度）和关联规则（及对应置信度），实验成功实现了基于Apriori算法的频繁项集和关联规则的挖掘。

## **3.3 实验总结**

在本次实验中，我深入理解了频繁项集、关联规则、支持度和置信度等重要概念，学习会使用Apriori算法求频繁项集的方法，同时对关联规则的挖掘原理也有了实践意义上的认知。

在实验开展过程中，我遇到了候选项集去重等问题，基于排序再去重的思路，将候选项集生成为有序字符串，再使用python的set容器实现去重，正确地获得了没有重复集合的候选项集。

当然，由于Apriori算法复杂度较高，在实验过程中我也感受到挖掘速度较慢的问题，基于Apriori算法还有许多优化策略，如PCY算法、FP-tree等，还有待今后进一步的学习、研究与探索。