**杭州电子科技大学计算机学院**

**数据仓库与数据挖掘**

**实验2：关联规则挖掘**

时间：2019年10月28日，学号：17061833 姓名：於文卓

**注意：**

**1）实验开始前，填写学号和姓名。**

**2）将文件名由“关联规则挖掘”更改为“学号+姓名+关联规则挖掘”。**

**3）作业做完后，验收之后通过作业提交系统提交。提交网址是：**

https://www.wjx.cn/jq/46478225.aspx

# 一、实验目的

1、了解关联规则的基本思想，理解频繁项集挖掘的一般过程和基本原理；

2、巩固Apriori算法的算法思想，能够进行关联规则分析。

# 二、实验原理

1、频繁项集

设I={i1, i2,…,im}是项的集合，D是数据库事务的集合，其中每个事务T是一个非空项集，使得T包含于I。每一个事务具有唯一的事务标识符TID。设A是一个项集，事务T包含A，当且仅当A包含于T。如果项集A中包含k个项，则称其为k项集。项集的出现频度是包含项集的事务数，简称为项集的频度、支持度计数或计数。如果项集I的相对支持度满足预定义的最小支持度阈值，则I是频繁项集。

2、关联规则、关联规则的挖掘

关联规则是形如A=>B的逻辑蕴含式，其中A包含于I，B包含于I，A不等于空集，B不等于空集，并且A交B等于空集。如果事务数据库D中有s%的事务包含A并B，则称关联规则X=>Y的支持度为s%，实际上，支持度是一个概率值。若项集A的支持度记为support(A)，规则的置信度为support(A并B)／support (A)。这是一个条件概率P(B|A)。也就是：

support(A=>B)=P(A并B)

confidence(A=>B)=P(B|A)=support(A并B)／support(A)

同时满足最小支持度阈值（min\_sup）和最小置信度阈值（min\_conf）的规则称为强规则。

为方便计算，用0%~100%之间的值，而不是0.0~1.0之间的值来表示支持度和置信度。

SouthEast

该式表明，规则A=>B的置信度容易从A并B的支持度计数推出。也就是说，一旦得到A、B和A并B的支持度计数，则导出对应的关联规则A=>B和B=>A，并检查它们是否是强规则是直截了当的。因此，挖掘关联规则的问题可以归结为挖掘频繁项集。

一般而言，关联规则的挖掘是一个两步的过程：

（1）找出所有的频繁项集：根据定义，这些项集的每一个频繁出现的次数至少与预定义的最小支持度计数min\_sup一样。

（2）有频繁项集产生强关联规则：根据定义，这些规则必须满足最小支持度和最小置信度。

由于第二步的开销远低于第一步，因此挖掘关联规则的总体性能由第一步决定。

3、Apriori算法

**（1）Apriori 算法概述**

Apriori算法是一种挖掘关联规则的频繁项集算法，其核心思想是通过候选集生成和情节的向下封闭检测两个阶段来挖掘频繁项集。而且算法已经被广泛的应用到商业、网络安全等各个领域。

**（2）问题的引入**

购物篮分析：引发性例子

Question：哪组商品顾客可能会在一次购物时同时购买？

关联分析

Solutions：

1：经常同时购买的商品可以摆近一点，以便进一步刺激这些商品一起销售。

2：规划哪些附属商品可以降价销售，以便刺激主体商品的捆绑销售。

**（3）Apriori算法**过程分为两个步骤：

* 第一步通过迭代，检索出事务数据库中的所有频繁项集，即支持度不低于用户设定的阈值的项集；
* 第二步利用频繁项集构造出满足用户最小信任度的规则。

具体做法就是：

首先找出频繁1-项集，记为L1；然后利用L1来产生候选项集C2，对C2中的项进行判定挖掘出L2，即频繁2-项集；不断如此循环下去直到无法发现更多的频繁k-项集为止。每挖掘一层Lk就需要扫描整个数据库一遍。算法利用了一个性质：

Apriori 性质：任一频繁项集的所有非空子集也必须是频繁的。意思就是说，生成一个k-itemset的候选项时，如果这个候选项有子集不在(k-1)-itemset(已经确定是frequent的)中时，那么这个候选项就不用拿去和支持度判断了，直接删除。具体而言：

1） 连接步

为找出Lk（所有的频繁k项集的集合），通过将Lk-1（所有的频繁k-1项集的集合）与自身连接产生候选k项集的集合。候选集合记作Ck。设l1和l2是Lk-1中的成员。记li[j]表示li中的第j项。假设Apriori算法对事务或项集中的项按字典次序排序，即对于（k-1）项集li，li[1]<li[2]<……….<li[k-1]。将Lk-1与自身连接，如果(l1[1]=l2[1])&&( l1[2]=l2[2])&&……..&& (l1[k-2]=l2[k-2])&&(l1[k-1]<l2[k-1])，那认为l1和l2是可连接。连接l1和l2 产生的结果是{l1[1],l1[2],……,l1[k-1],l2[k-1]}。

2） 剪枝步

CK是LK的超集，也就是说，CK的成员可能是也可能不是频繁的。通过扫描所有的事务（交易），确定CK中每个候选的计数，判断是否小于最小支持度计数，如果不是，则认为该候选是频繁的。为了压缩Ck,可以利用Apriori性质：任一频繁项集的所有非空子集也必须是频繁的，反之，如果某个候选的非空子集不是频繁的，那么该候选肯定不是频繁的，从而可以将其从CK中删除。

# 三、实验内容：

实验要求：利用Apriori算法求出事务表中的频繁项集和关联规则。

实验数据：10个购物篮事务构成的下列事务表，事务表由项目集I={ I1，I2，I3，I4，I5，I6}组成（项目集I中个项目对应的信息为：I1-牛奶，I2-啤酒，I3-尿布，I4-面包，I5-黄油，I6-饼干）。 其中，最小支持度设定为3，最小置信度设定为70%。

|  |  |
| --- | --- |
| TID | 商品 |
| T100 | 牛奶，啤酒，尿布 |
| T200 | 牛奶，面包，黄油 |
| T300 | 牛奶，尿布，饼干 |
| T400 | 面包，黄油，饼干 |
| T500 | 啤酒，尿布，饼干 |
| T600 | 牛奶，尿布，面包，黄油 |
| T700 | 尿布，面包，黄油 |
| T800 | 啤酒，尿布 |
| T900 | 牛奶，尿布，面包，黄油 |
| T1000 | 啤酒，饼干 |

（另： 如果上述数据完成后，有兴趣的同学可以选做Online Retail数据集（数据代表了2010-2011年英国零售商的交易数据），其中这个数据集的处理部分可以参考<http://www.360doc.com/content/17/0724/14/22146031_673760063.shtml>）

# 四、实验步骤

（代码）

*# -\*- coding: utf-8 -\*-*

*# @Author: TD21forever*

*# @Date: 2019-10-26 17:16:09*

*# @Last Modified by: TD21forever*

*# @Last Modified time: 2019-11-08 21:23:30*

MINSUPPORT = 3

MINCONF = 0.7

dataset = [[1, 3, 4], [2, 3, 5], [1, 2, 3, 5], [2, 5]]

dataFun = {

1: "牛奶", 2: "啤酒", 3: "尿布", 4: "面包", 5: "黄油", 6: "饼干",

}

data = [

[1, 2, 3],

[1, 4, 5],

[1, 3, 6],

[4, 5, 6],

[2, 3, 6],

[1, 3, 4, 5],

[3, 4, 5],

[2, 3],

[1, 3, 4, 5],

[2, 6]

]

**def** MakeC1(dataset):

*'''*

*找到dataset中不重复的元素,并且把每个元素置为frozenset*

*'''*

C1 = []

**for** row **in** data:

**for** item **in** row:

**if** **not** [item] **in** C1:

C1.append([item])

C1.sort()

C1 = list(map(frozenset, C1))

**return** C1

C1 = MakeC1(dataset)

*# print(C1)*

*# 其实就是减少Ck中元素的个数*

**def** MakeLk(D, Ck, minsupport=MINSUPPORT):

*'''*

*得出Ck中的每个元素在数据集D中所占的比例*

*如果达到最小支持度的要求,就将元素放入Lklist中*

*'''*

cnt = {}

**for** D\_data **in** D:

**for** C\_data **in** Ck:

*# 如果Ck中的某个元素是D中某个元素的子集*

**if** C\_data.issubset(D\_data):

cnt[C\_data] = 1 **if** C\_data **not** **in** cnt **else** cnt[C\_data] + 1

*# if not C\_data in cnt:*

*# cnt[C\_data] = 1*

*# else:*

*# cnt[C\_data] += 1*

*# 保存了所有Ck中元素的支持度*

supportValue = {}

*# 保存返回的L列表,舍弃一些不符合要求的元素后的Ck列表*

LkList = []

numItems = float(len(D))

**for** key **in** cnt:

**if** cnt[key] >= MINSUPPORT:

LkList.insert(0, key)

support = cnt[key] / numItems

supportValue[key] = support

**return** LkList, supportValue

a, b = MakeLk(dataset, C1)

*# print(a, b)*

**def** apriorGen(LK, k):

*'''*

*根据k的大小,相互组合*

*k=2 {0},{1},{2}--->{0,1},{0,2},{1,2}*

*k=3 {0,1},{1,2},{0,2}--->{0,1,2}*

*如果(l1[1]=l2[1])&&( l1[2]=l2[2])&&…&& (l1[k-2]=l2[k-2])&&(l1[k-1]<l2[k-1])，*

*那认为l1和l2是可连接。连接l1和l2 产生的结果是{l1[1],l1[2],……,l1[k-1],l2[k-1]}。*

*比如k=3 那么l1[0]和l2[0]就应该相等*

*'''*

retList = []

lenLK = len(LK)

**for** i **in** range(lenLK):

**for** j **in** range(i + 1, lenLK):

one = list(LK[i])[:k - 2]

two = list(LK[j])[:k - 2]

one.sort()

two.sort()

*# 通过树状图找规律发现,保证前几位一样,在并集*

*# 不然的话就得 计算所有情况然后去重 比较麻烦*

**if** one == two:

retList.append(LK[i] | LK[j])

**return** retList

*# print(apriorGen(a, 3))*

**def** apriori(dataset, k=2):

*'''*

*找到一个dataset的所有频繁项集*

*1. 求所有单个物品的项集列表 C*

*2. 去掉不满足最支持度的集合 L*

*3. 组合 C*

*4. goto 2*

*元数据去重--->C1--->筛选-->L1-->组合(apriori算法)--->C2--->筛选-->L2-->组合...*

*'''*

C1 = MakeC1(data)

D = list(map(set, dataset))

L1, supportValue = MakeLk(D, C1)

L = [L1]

index = 0

**while**(len(L[index]) > 0):

*# 组合*

Ck = apriorGen(L[index], k)

*# 筛选*

Lk, supportk = MakeLk(D, Ck, k)

supportValue.update(supportk)

L.append(Lk)

k += 1

index += 1

**return** L, supportValue

*# 最后找出所有的频繁项集*

L, supportData = apriori(data)

print(L)

sub\_set\_list = []

big\_rule = []

**for** each **in** range(len(L)):

**for** fre\_set **in** L[each]:

**for** sub\_set **in** sub\_set\_list:

**if** sub\_set.issubset(fre\_set):

start = fre\_set - sub\_set

end = sub\_set

conf = supportData[start | end] / supportData[start]

startName = []

endName = []

**for** one **in** start:

startName.append(dataFun[one])

**for** one **in** end:

endName.append(dataFun[one])

*# print("{}-->{},conf={}".format(start, end, conf))*

*# print("{}-->{},conf={}".format(startName, endName, conf))*

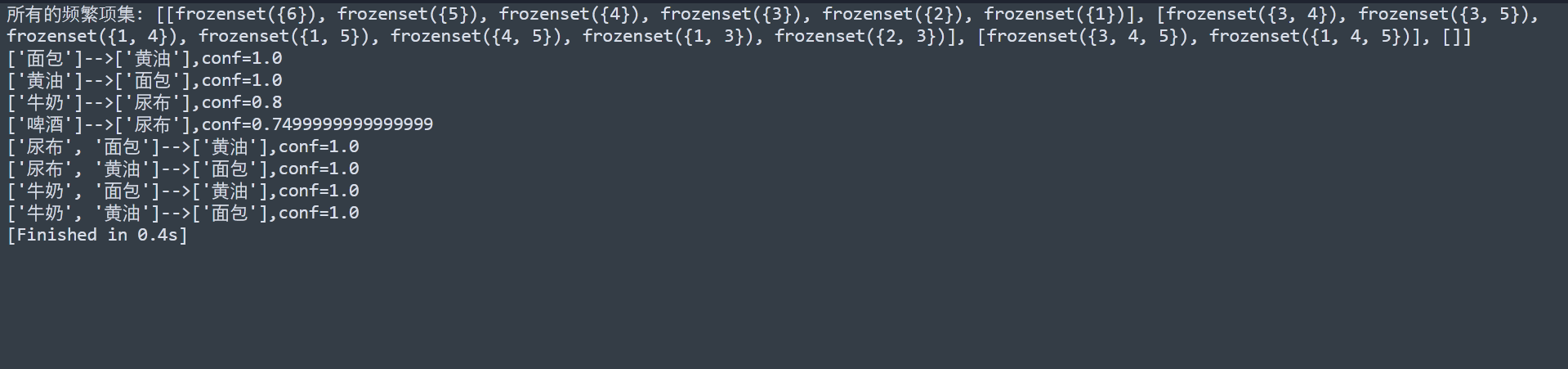
**if** conf >= MINCONF:

big\_rule.append((startName, endName, conf))

print("*{}*-->*{}*,conf=*{}*".format(startName, endName, conf))

sub\_set\_list.append(fre\_set)

# 五、实验结果



# 六、心得体会

这次实验有点数据挖掘的意思了,一些生活中的现象通过算法可以量化出来,找到一定的规律和依据.

总结下整个流程:

1. 得到数据集中没有重复的项 取名C1
2. 去掉C1中不符合条件的值,得到L1
3. 对L1使用apriori算法,根据k的大小进行组合,apriori说白了就是一种剪枝的方法,以为暴力遍历的话数据量一大复杂度太高了,但其实这个算法对于大型数据及复杂度还是很高..
4. 根据得到的所有频繁项集 获得强关联集,bigRule