# 云南大学软件学院
# 实 验 报 告

姓名：王子陈 学号：20171050008 专业：电子科学与技术 日期：2019/11/27 成绩：_____
任课教师：谢仲文

## 数据挖掘技术 实验六

## 一、实验目的

1. 掌握关联规则、频繁项集的概念，熟练运用 Apriori 算法进行关联规则分析。
2. 选择一种编程语言实现 Apriori 算法。
3. 熟悉 FP-growth 算法。

## 二、实验内容

1. 在一个简单的、虚拟的数据集（训练集）上应用相关算法进行频繁项集的计算。该数据集如下：

| 交易号 TID | 顾客购买的商品 | 交易号 TID | 顾客购买的商品 |
|---|---|---|---|
| T1 | 面包, 牛奶, 茶 | T6 | 面包, 牛奶, 啤酒, 尿布, 茶 |
| T2 | 面包, 尿布, 啤酒, 茶 | T7 | 啤酒, 牛奶, 茶 |
| T3 | 牛奶, 尿布, 啤酒 | T8 | 面包, 茶 |
| T4 | 面包, 牛奶, 尿布, 茶 | T9 | 面包, 尿布, 牛奶, 啤酒, 茶 |
| T5 | 面包, 尿布, 牛奶 | T10 | 面包, 牛奶 |

2. 基本要求 1：使用 Apriori 算法。（必做，相对支持度 minsup=35%）
3. 基本要求 2：从频繁项集中挖掘关联规则，并计算其置信度和提升度。（必做）
4. 提高要求：使用 FP-growth 算法。（选做，相对支持度 minsup=25%）

## 三、实验要求

1. 完成实验内容，源码作为实验报告附件一起打为一个压缩包提供。该压缩包要包含实验报告、代码文件。
2. 关键部分要求有注释，注释量不低于 20%
3. 要求独立完成，不得抄袭代码。

## 四、关键实验步骤（请粘贴关键步骤、代码、实验结果）

**Apriori 算法**

**1. 先设置数据集的格式: 每一条购买记录是一个元组, 每一个商品是一个集合, 集合之间可以做并, 交, 补的运算**

```
Dataset=[({'面包'},{'牛奶'},{'茶'}),
         ({'面包'},{'尿布'},{'啤酒'},{'茶'}),
         ({'牛奶'},{'尿布'},{'啤酒'}),
```

```
({'面包'},{'牛奶'},{'尿布'},{'茶'}),
({'面包'},{'尿布'},{'牛奶'}),
({'面包'},{'牛奶'},{'啤酒'},{'尿布'},{'茶'}),
({'啤酒'},{'牛奶'},{'茶'}),
({'面包'},{'茶'}),
({'面包'},{'尿布'},{'牛奶'},{'啤酒'},{'茶'}),
({'面包'},{'牛奶'})]
```

2.

```
#统计支持度个数
candidate_dict={} #商品字典：5 种商品
for set_origin in Dataset: #统计每种商品的支持度
    for thing in set_origin:
        if tuple(thing) not in candidate_dict.keys():
            candidate_dict[tuple(thing)]=0
        candidate_dict[tuple(thing)]+=1
prim_goods_dict=candidate_dict #置信度的分母
#可以得到字典: {('面包',): 8, ('牛奶',): 8, ('茶',): 7, ('尿布',): 6, ('啤酒',): 5}

all_frequent_dict={} #以字典的形式存放所有的频繁项集
while(len(candidate_dict)>1): #迭代到只剩一个候选项集
    #剪枝
    frequent_list=[]        #频繁项集
    non_frequent_list=[]    #非频项集
    for item in candidate_dict.items():#每一个 item 是一个元组(('面包')，8)
        if (item[1]>0.35*len(Dataset)):
            frequent_list.append(set(item[0])) #大于支持度的项集放入频繁列表
            if(len(item[0])>1):#不要 1 项集
                all_frequent_dict[item[0]]=item[1]
        else:
            non_frequent_list.append(set(item[0]))#把达不到 minsup 的项集放到非频繁项集里

 #连接
    new_compose_list=[] #连接商品组合
    copy_frequent_list=copy.copy(frequent_list) #拷贝一份以便删除
    #频繁项集里的每一项集与其他的项集组合,当他们两个补集等于 2 时可以连接
    for element in frequent_list:
        copy_frequent_list.remove(element)#和剩下的搭配
        for goods in copy_frequent_list:
            if len(goods^element)==2:
                union_set=element|goods
                new_compose_list.append(union_set)
```

```
    #再检查一次，看候选项集里有没有重新组合出已经被刷掉的非频繁项集
    for compose_set in new_compose_list:
        for non_frequent_set in non_frequent_list:
            if non_frequent_set.issubset(compose_set):
                new_compose_list.remove(compose_set)

    #清除重复项集(比如{'牛奶","面包","茶"}出现了 2 次)
    clean_compose_list=[]
    for compose in new_compose_list:
        if compose not in clean_compose_list:
            clean_compose_list.append(compose)

    #统计支持度个数
    new_candidate_dict={}
    for new_combine in clean_compose_list: #new_combine 是集合
        #小集合变大集合:2 个商品是 2 个集合,这里要合成一个集合
        for original_set in [[list(Dataset[y][x])[0] for x in range(0,len(Dataset[y]))]for y in range(0,len(Dataset))]:
            #判断 new_combine 是不是子集，要转化为集合类型并用 issubset 方法判断
            if set(new_combine).issubset(original_set):
                if tuple(new_combine) not in new_candidate_dict.keys():
                    new_candidate_dict[tuple(new_combine)]=0
                new_candidate_dict[tuple(new_combine)]+=1
    candidate_dict=new_candidate_dict #候选项集更新
```

**可以得到所有的频繁项集:**

{('牛奶', '面包'): 6, ('茶', '面包'): 6, ('面包', '尿布'): 5, ('牛奶', '茶'): 5, ('牛奶', '尿布'): 5, ('牛奶', '啤酒'): 4, ('茶', '尿布'): 4, ('茶', '啤酒'): 4, ('尿布', '啤酒'): 4, ('牛奶', '茶', '面包'): 4, ('牛奶', '面包', '尿布'): 4, ('茶', '面包', '尿布'): 4}

**根据频繁项集计算置信度和提升度:**

```
for key in all_frequent_dict.keys():
    if(len(key)==2): #频繁 2 项集
        for good in key:
            confidence=all_frequent_dict[key]/prim_goods_dict[tuple({good})] #置信度
            lift=confidence/(prim_goods_dict[tuple({key[key.index(good)-1]})]/10) #支持度
    if(len(key)==3):    #频繁 2 项集
        for good in key:
            copyKey=list(copy.copy(key))
            copyKey.remove(good)
            if tuple(copyKey)in all_frequent_dict: #如果不是频繁项集,就不要
                confidence=all_frequent_dict[key]/all_frequent_dict[tuple(copyKey)] #置信度
            lift=confidence/(prim_goods_dict[tuple({good})]/10) #支持度
```

得到结果:

('牛奶', '面包')
置信度: 牛奶 | 面包 = 0.75
置信度: 面包 | 牛奶 = 0.75
提升度: 面包 | 牛奶 = 0.9375
('面包', '茶')
置信度: 面包 | 茶 = 0.75
提升度: 面包 | 茶 = 1.0714285714285714
置信度: 茶 | 面包 = 0.8571428571428571
**提升度: 茶 | 面包 = 1.0714285714285714**
('面包', '尿布')
置信度: 面包 | 尿布 = 0.625
提升度: 面包 | 尿布 = 1.0416666666666667
置信度: 尿布 | 面包 = 0.8333333333333334
**提升度: 尿布 | 面包 = 1.0416666666666667**
('牛奶', '茶')
置信度: 牛奶 | 茶 = 0.625
提升度: 牛奶 | 茶 = 0.8928571428571429
置信度: 茶 | 牛奶 = 0.7142857142857143
提升度: 茶 | 牛奶 = 0.8928571428571428
('牛奶', '尿布')
置信度: 牛奶 | 尿布 = 0.625
提升度: 牛奶 | 尿布 = 1.0416666666666667
置信度: 尿布 | 牛奶 = 0.8333333333333334
**提升度: 尿布 | 牛奶 = 1.0416666666666667**
('牛奶', '啤酒')
置信度: 牛奶 | 啤酒 = 0.5
提升度: 牛奶 | 啤酒 = 1.0
置信度: 啤酒 | 牛奶 = 0.8
提升度: 啤酒 | 牛奶 = 1.0
('尿布', '茶')
置信度: 尿布 | 茶 = 0.6666666666666666
提升度: 尿布 | 茶 = 0.9523809523809524
置信度: 茶 | 尿布 = 0.5714285714285714
提升度: 茶 | 尿布 = 0.9523809523809523
('啤酒', '茶')
置信度: 啤酒 | 茶 = 0.8
提升度: 啤酒 | 茶 = 1.142857142857143
置信度: 茶 | 啤酒 = 0.5714285714285714
**提升度: 茶 | 啤酒 = 1.1428571428571428**
('啤酒', '尿布')
置信度: 啤酒 | 尿布 = 0.8
提升度: 啤酒 | 尿布 = 1.3333333333333335
置信度: 尿布 | 啤酒 = 0.6666666666666666
**提升度: 尿布 | 啤酒 = 1.3333333333333333**
('牛奶', '面包', '茶')
先买 ['面包', '茶']
置信度: ('牛奶', '面包', '茶') | ['面包', '茶'] = 0.6666666666666666
提升度: 0.8333333333333333
先买 ['牛奶', '茶']
置信度: ('牛奶', '面包', '茶') | ['牛奶', '茶'] = 0.8
提升度: 1.0
先买 ['牛奶', '面包']
置信度: ('牛奶', '面包', '茶') | ['牛奶', '面包'] = 0.6666666666666666
提升度: 0.9523809523809524

('牛奶', '面包', '尿布')
先买 ['面包', '尿布']
置信度: ('牛奶', '面包', '尿布') | ['面包', '尿布'] = 0.8
提升度: 1.0
先买 ['牛奶', '尿布']
置信度: ('牛奶', '面包', '尿布') | ['牛奶', '尿布'] = 0.8
提升度: 1.0
**先买 ['牛奶', '面包']**
**置信度: ('牛奶', '面包', '尿布') | ['牛奶', '面包'] = 0.6666666666666666**
**提升度: 1.1111111111111112**

('尿布', '面包', '茶')
先买 ['面包', '茶']
置信度: ('尿布', '面包', '茶') | ['面包', '茶'] = 0.6666666666666666
提升度: 1.1111111111111112
先买 ['尿布', '茶']
置信度: ('尿布', '面包', '茶') | ['尿布', '茶'] = 1.0
提升度: 1.25
**先买 ['尿布', '面包']**
**置信度: ('尿布', '面包', '茶') | ['尿布', '面包'] = 1.0**
**提升度: 1.428571428571428**6

**可以得到关联规则:**

提升度=1 意味着两者的购买互相独立没有关联, 如: (啤酒,牛奶), (面包, [牛奶,茶]), (牛奶, [面包,尿布]), (面包，[牛奶,尿布])

1. 面包→茶的置信度 confidence=85.7%,意味着购买面包的顾客中有 85.7%也购买了茶, 但其提升度为 1.071,两者关联性较弱,很多没有买面包的人也买了茶.

2. 面包→尿布的置信度 confidence=83.3%,意味着购买面包的顾客中有 83.3%也购买了尿布, 但其提升度为 1.041,两者关联性较弱,很多没有买面包的人也买了尿布. 牛奶→尿布也是同样的情况

3. 茶→啤酒的置信度 confidence=80%,意味着购买茶的顾客中有 80%也购买了啤酒, 其提升度为 1.143,两者关联性较弱,很多没有买茶的人也买了啤酒.

4. 尿布→啤酒的置信度 confidence=80%,意味着购买尿布的顾客中有 80%也购买了啤酒, 其提升度为 1.333,两者关联性较强,买啤酒的人里很多是先买尿布的

5. 先买[尿布,面包]再买茶的置信度在本数据集中为 1, 提升度也较高,可以看出有很强的关联性.