

数据挖掘实验报告

Apriori频繁集挖掘

学 院：电子信息与通信学院

班 级： 提高2101班

姓 名： 杨筠松

学 号： U202115980

实验时间： 2024年 4月

**Apriori 算法**是一种发掘事物内在关联关系的算法，它可以加快关联分析的速度，从而让我们更有效的进行关联分析。

## 1，关联分析

**关联分析**用于发掘大规模数据集中的内在关系。

关联分析一般要分析数据集中的**频繁项集**（frequent item sets）和**关联规则**（association rules）：

* 频繁项集：是数据集中**频繁项**的集合，集合中可以有**一项或多项**物品。
* 关联规则：暗示了两种物品之间可能存在很强的内在关系。

假设，我们收集了一家商店的交易清单：

| **交易编号** | **购物清单** |
| --- | --- |
| 1 | 牛奶，面包 |
| 2 | 牛奶，面包，火腿 |
| 3 | 面包，火腿，可乐 |
| 4 | 火腿，可乐，方便面 |
| 5 | 面包，火腿，可乐，方便面 |

频繁项集是一些经常出现在一起的物品集合。比如：{牛奶，面包}，{火腿，方便面，可乐}都是频繁项集的例子。关联规则意味着有人买了一种物品，还会买另一种物品。比如方便面->火腿，就是一种关联规则，表示如果买了方便面，还会买火腿。

## 2，三个重要概念

关联分析中有三个重要的概念，分别是：

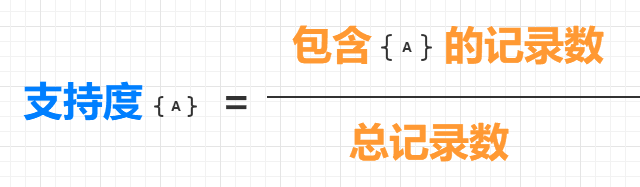
* 支持度
* 可信度 **/** 置信度
* 提升度

***支持度***

要进行关联分析，首先要寻找**频繁项**，也就是频繁出现的物品集。那么怎样才叫频繁呢？我们可以用**支持度**来衡量频繁。

**支持度**是针对**项集**来说的，一个项集的支持度就是该项集的记录占总记录的比例。通常可以定义一个**最小支持度**，从而只保留满足最小支持度的项集。

一个项集{A} 的支持度的定义如下：

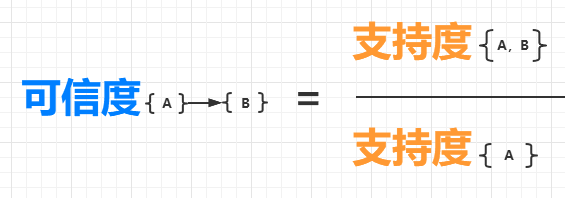


比如，在上面表格中的5 项记录中，{牛奶} 出现在了两条记录中，所以{牛奶} 的支持度为 2/5；而{面包，火腿} 出现在了三条记录中，所以{面包，火腿}的支持度为3/5。

***可信度***

**可信度**又叫**置信度**，它是针对**关联规则**来说的，比如{火腿}->{可乐}。

一个关联规则{A}->{B} 表示，如果购买了**物品A**，会有多大的概率购买**物品B**？它的可信度的定义如下：

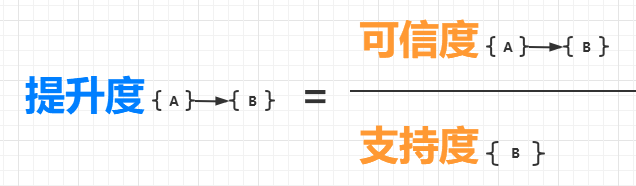


所以，在上面的表格中，{火腿，可乐} 的支持度是 3/5，{火腿} 的支持度是 4/5，所以{可乐}->{火腿} 的可信度为 3/5 除以 4/5，等于 0.75。这意味着，如果购买了火腿，有 75% 的可能性会购买可乐。

***提升度***

提升度也是针对关联规则来说的，它表示的是“如果购买物品A，会对购买物品B 的概率**提升**多少”。

一个关联规则{A}->{B} 的提升度的定义如下：



提升度会有三种情况：

* 提升度{A}->{B} > 1：表示购买物品A 对购买物品B 的概率有提升。
* 提升度{A}->{B} = 1：表示购买物品A 对购买物品B 的概率没有提升，也没有下降。
* 提升度{A}->{B} < 1：表示购买物品A 对购买物品B 的概率有下降。

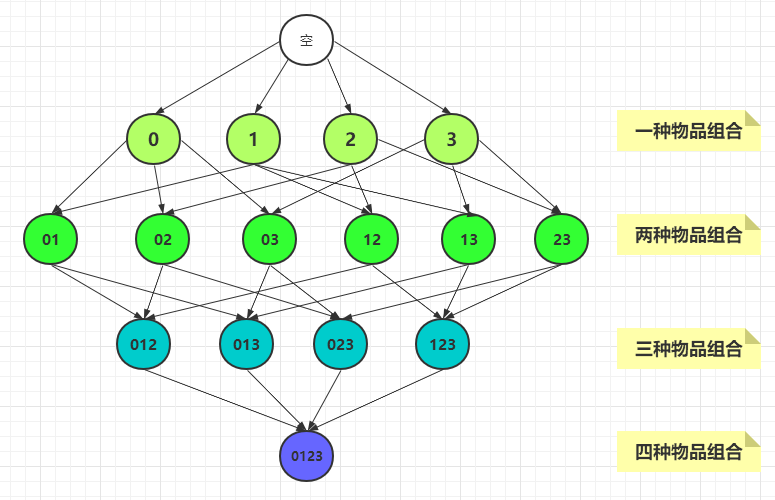
### 3，如何寻找频繁项

寻找频繁项的一个简单粗暴的方法是，**对所有的物品进行排列组合**，然后计算所有组合的支持度，这种算法也可以叫做**穷举法**。

***穷举法***

穷举法就是列出所有物品的组合，然后计算每种组合的支持度。

比如，我们有一个物品集{0，1，2，3}，其中有四个物品，那么所有的物品组合如下：



从图中可以看到一共有**15** 种组合，计算每一种组合的支持度都需要遍历一遍所有的记录，检查每个记录中是否包含该组合。因此有多少种组合，就需要遍历多少遍记录，时间复杂度则会很大。

可以总结出：包含**N** 种物品的数据集，共有 **2N - 1** 种组合。为了计算每种组合的支持度，则需要遍历 **2N - 1** 次记录。

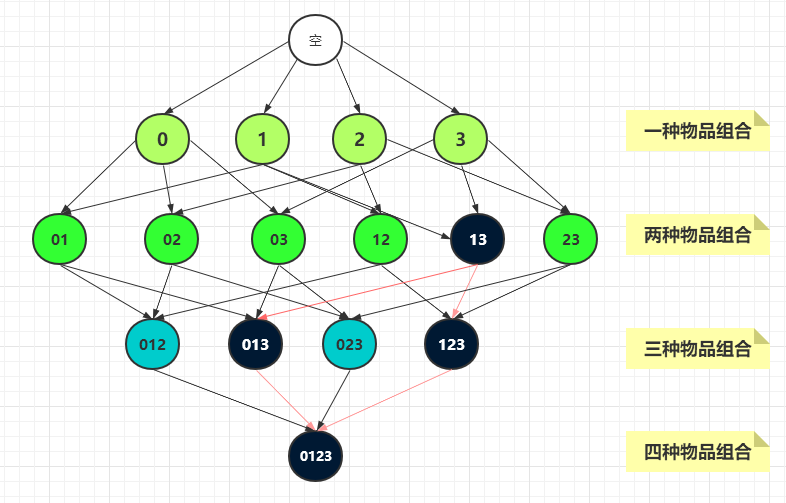
如果一个商店中有100 款商品，将会有**1.26\*1030** 种组合，这是一个非常庞大的数字。而普通商店一般都会有成千上万的商品，那么组合数将大到无法计算。

### 4，Apriori 算法

为了降低计算所需的时间，1994 年 **Agrawal** 提出了著名的 **Apriori 算法**，该算法可以有效减少需要计算的组合的数量，避免组合数量的指数增长，从而在合理的时间内计算出频繁项集。

**Apriori 原理**是说：如果一个项集是**非频繁集**，那么它的所有超集也是**非频繁的**。

比如下图中的项集{1，3} 是非频繁集，那么{0，1，3}，{1，2，3}，{0，1，2，3} 就都是非频繁项集。这就大大减少了需要计算的项集的数量。



### 5，Apriori 算法的实现

| **交易编号** | **购物清单** |
| --- | --- |
| 1 | 牛奶，面包 |
| 2 | 牛奶，面包，火腿 |
| 3 | 面包，火腿，可乐 |
| 4 | 火腿，可乐，方便面 |
| 5 | 面包，火腿，可乐，方便面 |

这里，我们使用**Apriori 算法**来寻找上文表格中的购物清单的频繁项集（为了方便查看，我把表格放在这里）。

***efficient\_apriori 模块***

[Efficient-Apriori](https://pypi.org/project/efficient-apriori/) 包是**Apriori 算法**的稳定高效的实现，该模块适用于 Python 3.6+。

使用**Apriori 算法**要先安装：pip install efficient-apriori

**efficient\_apriori** 包中有一个 apriori 函数，原型如下（这里只列出了常用参数）：

1. apriori(data,
2. min\_support = 0.5,
3. min\_confidence = 0.5)

参数的含义：

* **data**：表示数据集，是一个列表。列表中的元素可以是元组，也可以是列表。
* **min\_support**：表示最小支持度，小于最小支持度的项集将被舍去。
  + 该参数的取值范围是 **[0, 1]**，表示一个百分比，比如**0.3** 表示**30%**，那么支持度小于**30%** 的项集将被舍去。
  + 该参数的默认值为**0.5**，常见的取值有**0.5，0.1，0.05**。
* **min\_confidence**：表示最小可信度。
  + 该参数的取值范围也是 **[0, 1]**。
  + 该参数的默认值为**0.5**，常见的取值有**1.0，0.9，0.8**。

***使用 apriori 函数***

首先，将表格中的购物清单转化成 **Python** 列表，如下：

1. data = [
2. ('牛奶', '面包'),
3. ('牛奶', '面包', '火腿'),
4. ('面包', '火腿', '可乐'),
5. ('火腿', '可乐', '方便面'),
6. ('面包', '火腿', '可乐', '方便面')
7. ]

挖掘频繁项集和频繁规则：

1. itemsets, rules = apriori(data, min\_support=0.5,  min\_confidence=1)

查看频繁项集和频繁规则：

1. >>> itemsets *# 频繁项集*
2. {1: { *# 只有一个元素的项集*
3. ('面包',): 4, *# 4 表示记录数*
4. ('火腿',): 4,
5. ('可乐',): 3
6. },
7. 2: { *# 有两个元素的项集*
8. ('火腿', '面包'): 3,
9. ('可乐', '火腿'): 3
10. }
11. }
12. >>> rules *# 频繁规则*
13. [{可乐} -> {火腿}]

### 6，总结

本篇实验报告中主要介绍了什么是关联分析，关联分析中三个重要的概念，以及 **Apriori** 算法。

**Apriori** 算法用于加快关联分析的速度，但它也需要多次扫描数据集。其实除了**Apriori** 算法，还有其它算法也可以加快寻找频繁项集的速度。