关联分析：Apriori与FP-Growth

一、什么是关联分析?

- 一个事件发生时，另一个事件也随之发生，则这两个事件是关联的。关联分析本质上是求条件概率。关联分析的典型案例是购物篮分析，通过挖掘购物篮中各商品之间的关系，构成关联规则（AssociationRule），从而有针对性地制定营销策略。
- 举例：零售店考察顾客购买的商品之间的关联性，网络流量分析，...

二、关联分析的基础概念

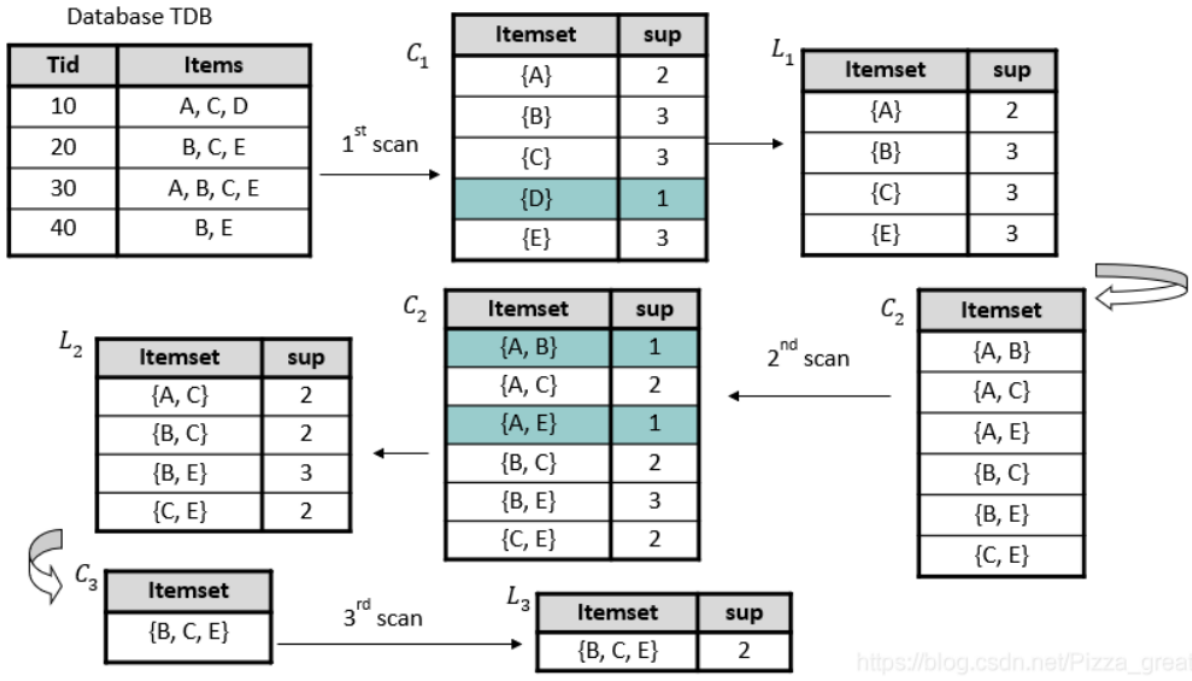
- 频繁项集
    - 项集：包含0个或多个项的集合被称为项集；如果一个项集包含k个项，则称它为k项集。
    - 频繁项集：**经常出现在一块的物品的集合**（项集的支持度满足预设置的最小支持度阈值）。
- 关联规则：**暗示两种物品之间可能存在很强的关系**，形式为P->H
- 支持度（Support）————最小支持度
    - 意义：确定规则可以用于给定数据集的频繁程度（频率）。
    - $Support = \frac{\sigma(X \cup Y \cup ...)}{\text{总记录数}N}$，其中$\sigma(\cdot)$指计数
- 置信度（Confidence）、左手项与右手项
    - 意义：确定Y在包含X的事务中出现的频繁程度。
    - $Confidence(P->H) = P(P|H) = \frac{Support(P \cup H)}{Support(P)}$
- 提升度（List）：比较两个概率，一个是在已知购买了左手项情况下购买右手项的概率（即置信度），另一个是任意情况下购买右手项的概率（即右手项的支持度）。提升度大于1，说明依据关联规则的交叉销售能产生更大的商业价值。
- 热门 VS 冷门：可以设置阈值来区分热门与冷门商品。为热门商品设置较大的最小支持度，而为冷门商品设置较小的最小支持度。

三、Apriori算法步骤

1. 设定最小支持度和置信度，比如最小支持度为30%，最小置信度为70%。
2. 找出满足最小支持度的频繁项集，比如我们找到了[啤酒]，[尿布]，[口罩]，[啤酒、尿布]，[啤酒、口罩]，[啤酒，尿布，口罩]
3. 对于第二步中的每一个频繁项集，找到满足最小置信度的关联规则，此时我们共有10个规则需要判断：[A->B], [B->A], [A->C], [C->A], [A,B->C], [A,C->B], [B,C->A],[A->B,C],[B->A,C],[C->A,B]，然后一一计算其置信度，淘汰小于最小置信度的规则。
4. 验证关联规则的有效性，即对于每一条规则计算提升度（Lift），大于1则证明这种交叉销售具有商业价值。

四、Apriori

- A Priori：“一个先验”，先验可以是其他领域已有的知识，也可以是**先前的测量结果**
- 原始计算Support方法：
  - 对于给定Data Set，遍历以确认是否存在某一个特定集合，有则+1
  - 缺陷：笨
- Apriori原则：
  - 频繁项集的子集也是频繁的，
  - 非频繁项集的母集也是非频繁的
- Apriori与频繁项集



1、首先，通过扫描数据集，产生一个大的候选数据项集,并计算每个候选数据项发生的次数，然后基于预先给定的最小支持度生成频繁1项集的集合，该集合记作L1；

2、然后基于L1和数据集中的数据，产生频繁2项集L2；

3、用同样的方法，直到生成频繁n项集，其中已不再可能生成满足最小支持度的（N+1）项集（即只剩下了一项）；

4、最后，从大数据项集中导出规则。

- Apriori算法使用产生-测试策略来发现频繁项集。在每次迭代之后，新的候选项集都由前一次迭代发现的频繁项集产生，然后对每个候选的支持度进行计数，并于最小支持度阈值进行比较。

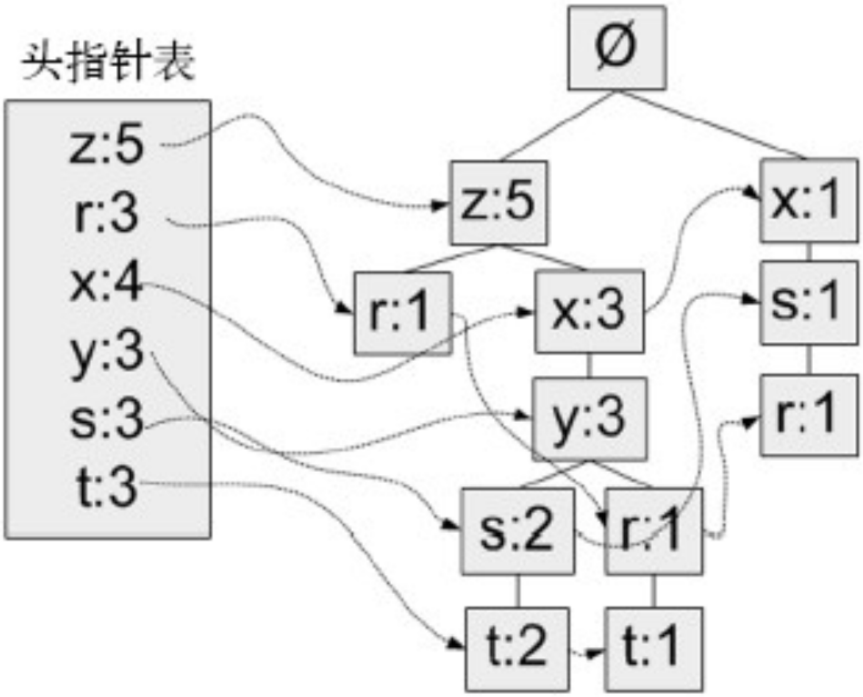
- Apriori与关联规则

    - 找到某个频繁项集的所有非空子集，按左手项和右手项组成规则，因为任意左手项和右手项的支持频度必然在Apriori算法中被计算过，因此无须扫描数据库就可以直接计算置信度。再以最小置信度要求将不符合要求的规则过滤掉即可。

五、FP-Growth（Frequent Pattern）

- intro：在较大数据集上Apriori需要花费大量的运算开销，而FP-Growth却不会有这个问题。因为FP-Growth只扫描整个数据库两次。这种做法（FP）使得算法的执行速度要快于Apriori，通常性能要好两个数量级以上。
- Aiming：将数据集存储在一个特定的称作FP树的结构之后发现频繁项集或者频繁项对，即常在一块出现的元素项的集合FP树。
- 步骤：
    1. 构建FP tree
    2. 从FP tree中挖掘频繁项集
- FP tree:

001	r, z, h, j, p
002	z, y, x, w, v, u, t, s
003	z
004	r, x, n, o, s
005	y, r, x, z, q, t, p
006	y, z, x, e, q, s, t, m

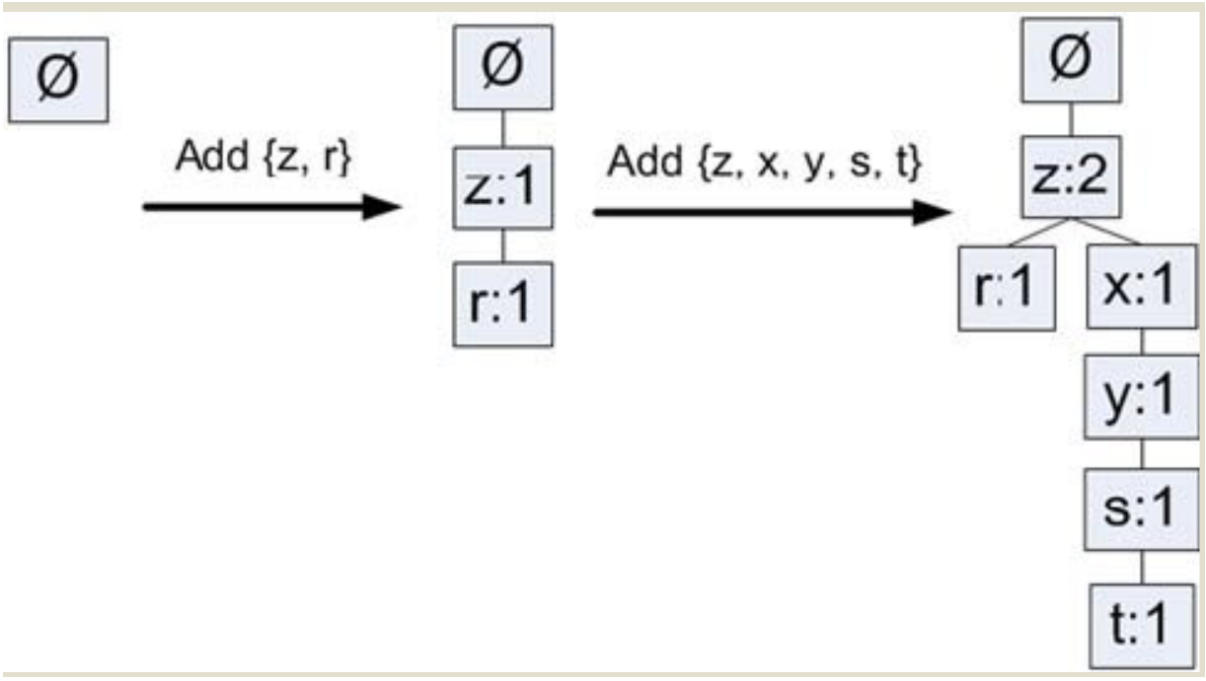



```
class treeNode:
    def __init__(self, nameValue, numOccur, parentNode):
        self.name = nameValue
        self.count = numOccur
        self.nodeLink = None
        self.parent = parentNode
        self.children = {}

    def inc(self, numOccur):
        self.count += numOccur

    def disp(self, ind=1):
        print ' '*ind, self.name, ' ', self.count
        for child in self.children.values():
        child.disp(ind+1)
```

- 第一次遍历数据集会获得每个元素项的出现频率。接下来，去掉不满足最小支持度的元素项。再下一步构建FP树。在构建时，读入每个项集并将其添加到一条已经存在的路径中。
- 如果该路径不存在，则创建一条新路径。每个事务就是一个无序集合。假设有集合{z,x,y}和{y,z,r}，那么在FP树中，相同项会只表示一次。为了解决此问题，在将集合添加到树之前，需要对每个集合进行排序。
- 排序基于元素项的绝对出现频率来进行。第一次遍历数据集会获得每个元素项的出现频率。接下来，去掉不满足最小支持度的元素项。
- 再下一步构建FP树。

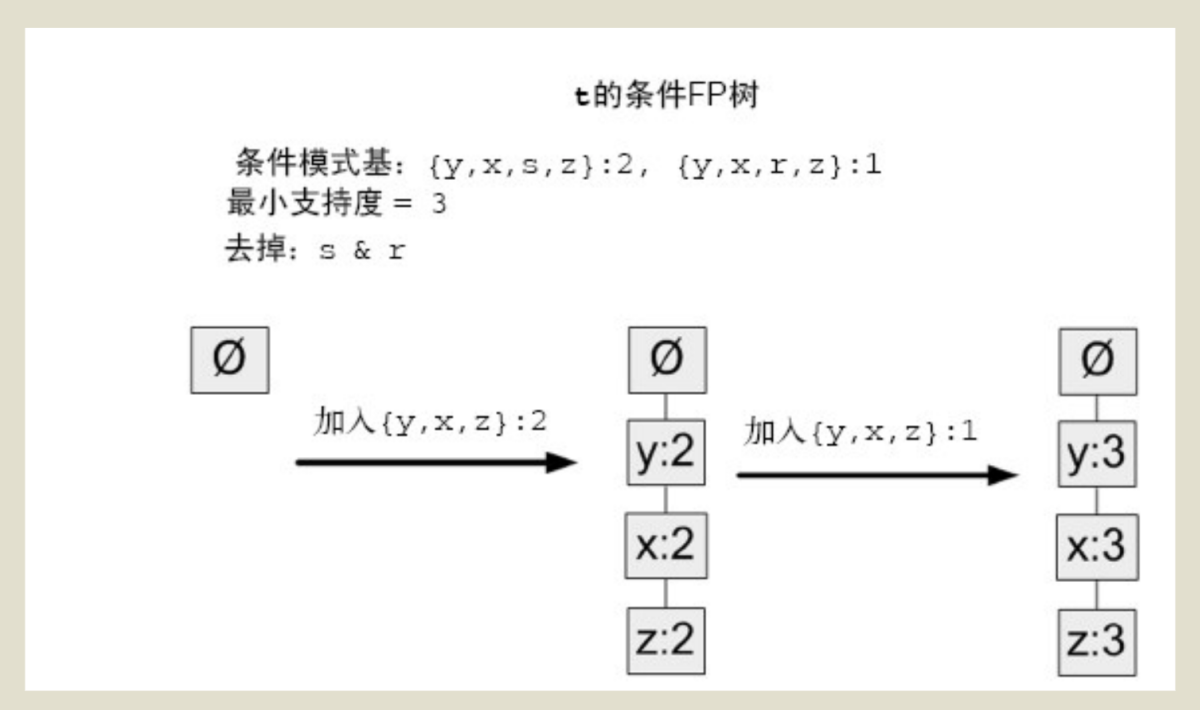


- 从FP tree中挖掘频繁项集

1. 从FP树中获得条件模式基：
   - 条件模式基是以所查找元素项为结尾的路径集合。每一条路径其实都是一条前缀路径（prefix path）。简而言之，一条前缀路径是介于所查找元素项与树根节点之间的所有内容。

频繁项	前缀路径
z	{}5
r	{x,s}1, {z,x,y}1, {z}1
x	{z}3, {}1
y	{z,x}3
s	{z,x,y}2, {x}1
t	{z,x,y,s}2, {z,x,y,r}1

   - 为了获得这些前缀路径，可以对树进行穷举式搜索，直到获得想要的频繁项为止，或者使用一个更有效的方法来加速搜索过程。可以利用先前创建的头指针表来得到一种更有效的方法。头指针表包含相同类型元素链表的起始指针。一旦到达了每一个元素项，就可以上溯这棵树直到根节点为止。
2. 利用条件模式基，构建一个条件FP树；



- note：t的条件FP树的创建过程。最初树以空集作为根节点。接下来，原始的集合{y,x,s,z}中的集合{y,x,z}被添加进来。因为不满足最小支持度要求，字符s并没有加入进来。类似地，{y,x,z}也从原始集合{y,x,r,z}中添加进来

4. 迭代重复步骤1和步骤2，直到树包含一个元素项为止。

六、Reference

- 《金融商业算法建模：基于Python和SAS》
- 《机器学习实战》
- https://blog.csdn.net/qq_18298439/article/details/110492483?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216607808711678239538042 9%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166078087116782395380429&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-2-110492483-null-null.142^v41^pc_rank_34_2,185^v2^control&utm_term=apriori&spm=1018.2226.3001.4187
- https://blog.csdn.net/Pizza_great/article/details/101224098?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-1-101224098-blog-110492483.pc_relevant_vip_default&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-1-101224098-blog-110492483.pc_relevant_vip_default&utm_relevant_index=1