博雅大数据学院
BOYA BIG DATA COLLEGE

第七部分
关联规则挖掘

何向南
hexn@ustc.edu.cn

10 April 2020

目录

- 关联规则概述
- Apriori算法
- FP-Growth算法
- 辛普森悖论

关联规则概述

- 面对海量数据，发现数据特征之间隐藏的关系
  - 使用统计学中的相关性检验，验证数据特征是否存在相关性
    - 皮尔森相关性稀疏反映数据特征之间的线性相关性，取值范围 $[-1, 1]$，绝对值越大相关性越强
  - 使用**关联规则（association rule）**显式表明数据中的特征之间的关联关系
    - 关联规则形如表达式 $A \Rightarrow B$
    - 关联规则学习是挖掘数据特征之间关联关系的技术
    - 应用领域：购物篮分析、医疗手段关联关系分析、优化交通道路设计与管理

关联规则指标

- 重要性
  - 使用关联规则中包含的所有特征的支持度 $S(A \cup B)$ 来衡量
  - 含义为关联规则被多少比例的数据覆盖
  - 数学表达式

$$S(A \Rightarrow B) = S(A \cup B) = \frac{\text{itemset including} \quad A \cup B}{\text{total itemset}}$$

关联规则指标

- 置信度（confidence）
  - 衡量规则的可信程度
  - 数学表达式为 $C(A \Rightarrow B) = \dfrac{S(A \cup B)}{S(A)}$

- 以规则 { 男性，尿布 } ⇒ { 啤酒 } 为例，支持度表示所有的购物记录中，有多少比例的购物记录中的购物者是男性，且同时购买了尿布和啤酒；置信度则表示在购物者为男性且购买了尿布的记录中，有多大的比例同时也购买了啤酒

关联规则概述

- 根据支持度和置信度指标，给定数据集
  - 寻找所有不小于预置支持度和置信度阈值的规则集合
  - 满足上述最小条件的规则，也称为**强关联规则（strong association rule）**
  - 寻找满足条件规则的过程主要分为两个步骤
    ① 找到满足支持度阈值条件的规则，即频繁规则
    ② 筛选频繁规则，过滤掉不满足置信度阈值条件的规则

博雅大数据学院
BOYA BIG DATA COLLEGE

- 从关联规则学习的角度来看
  - 寻找满足支持度阈值条件的规则，等价于找出满足支持度阈值的特征集（项集）
  - 满足支持度阈值的特征集称为**频繁项集**

- 关联规则学习的核心为寻找数据集中的频繁项集，关联规则学习也称为**频繁项集挖掘**

关联规则概述

- 关联规则学习的一般步骤
  ① 寻找所有的频繁项集
  ② 根据频繁项集，生成频繁规则
  ③ 根据置信度阈值条件过滤筛选规
- 关联规则学习一般只能处理布尔型数据
  - 如果数据为连续型或度类型，需要进行离散化和数据编码等预处理操作

关联规则实例

- 给定手机评论数据集

	x_1	x_2	x_3	x_4	x_5
评论编号	功能	速度	屏幕	手感	客服
1	1	1	1	1	0
2	0	1	0	0	1
3	0	1	1	0	0
4	1	0	1	1	0
5	0	0	1	1	1

- 数据集每列表示对应的词语在一片评论中是否出现
- 使用特征集合来表示样本，集合中的元素为样本值取值为1的所有特征

关联规则实例

博雅大数据学院 BOYA BIG DATA COLLEGE

- 手机评论数据集的集合表示形式
  - 样本的每个特征称为项（item）
  - 每个样本是一个项集的集合，称为项集（itemset）
  - 如果一个项集的数量为 k ， 则称为 k 项集

评论编号	项集
1	$\{x_1, x_2, x_3, x_4\}$
2	$\{x_2, x_5\}$
3	$\{x_2, x_3\}$
4	$\{x_1, x_3, x_4\}$
5	$\{x_3, x_4, x_5\}$

关联规则实例

博雅大数据学院 BOYA BIG DATA COLLEGE

评论编号	项集
1	$\{x_1, x_2, x_3, x_4\}$
2	$\{x_2, x_5\}$
3	$\{x_2, x_3\}$
4	$\{x_1, x_3, x_4\}$
5	$\{x_3, x_4, x_5\}$

- 给定支持度阈值为0.4，置信度阈值为0.6
- 项集 $\{x_2, x_3\}$ 在评论1和3中出现，支持度为
$$S(\{x_2, x_3\}) = 2/5 = 0.4$$
- $\{x_2\}$ 在前3条评论中都出现，支持度为 $S(\{x_2\}) = 3/5 = 0.6$
- 判断 $\{x_2\} \Rightarrow \{x_3\}$ 是否为强规则

$$C(\{x_2\} \Rightarrow \{x_3\}) = \frac{S(\{x_2, x_3\})}{S(\{x_2\})} = 2/3 \approx 0.67$$

评论编号	功能 x_1	速度 x_2	屏幕 x_3	手感 x_4	客服 x_5
1	1	1	1	1	0
2	0	1	0	0	1
3	0	1	1	0	0
4	1	0	1	1	0
5	0	0	1	1	1

目录

博雅大数据学院
BOYA BIG DATA COLLEGE

- 关联规则概述
- **Apriori算法**
- FP-Growth算法
- 辛普森悖论

Apriori算法

- Apriori算法
  - 1994年由Rakesh Agrawal等提出的挖掘频繁项集的经典算法
  - 思路：首先产生候选集，然后验证候选集是否满足频繁的要求

- Apriori算法（A Priori）得名的原因
  - 基于频繁1项集，发现频繁2项集，一次类推
  - 频繁 n 项集的发现基于频繁 $(n+1)$ 项集

Apriori算法

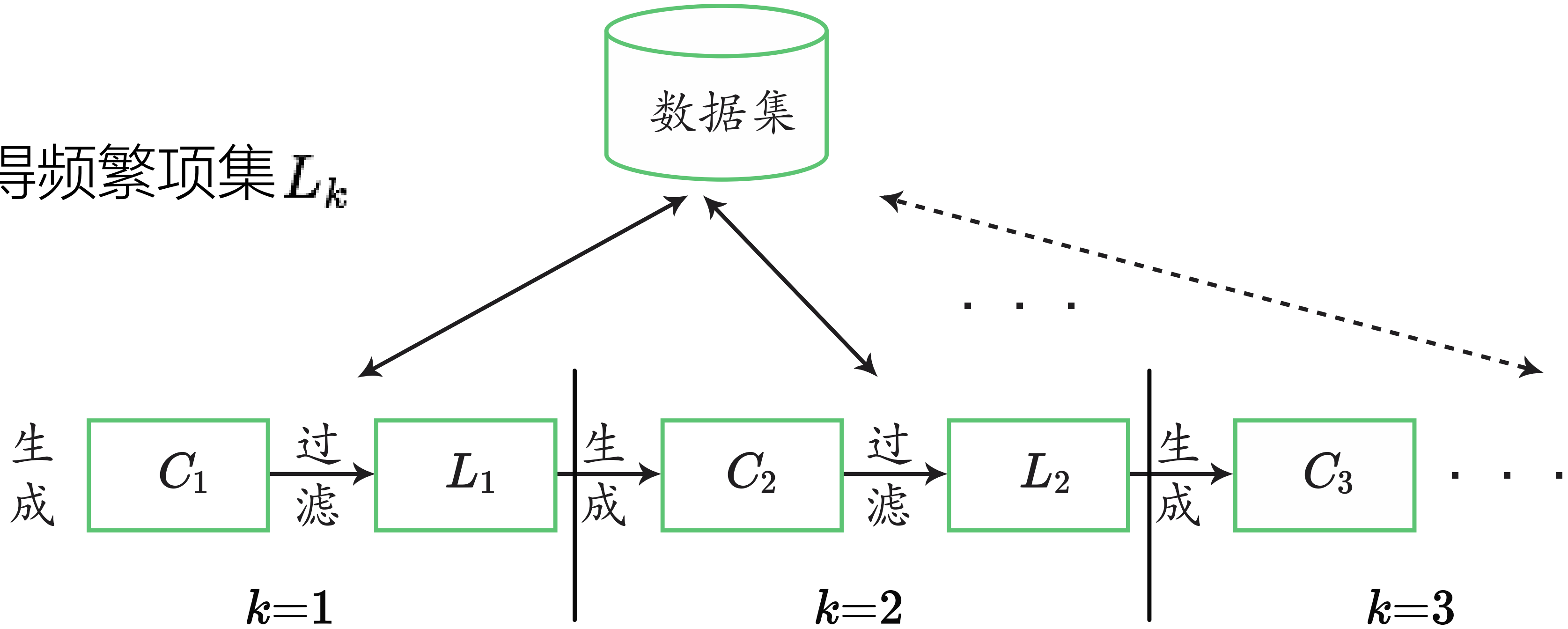
- Apriori算法使用项集的支持度性质，避免穷举所有候选集，提高计算效率
- Apriori算法的性质：
  - 如果一个项集不是频繁项集，那么它的所有超集都不可能是频繁项集
- 依据该性质，对于某 $(n+1)$ 项集，只要存在一个 n 项子集不是频繁项集，则可以直接判定该项集不是频繁项集

Apriori算法

- 令 $k=1,2,3,\cdots,n,\cdots$，代表Apriori算法的迭代步数
- 从 $k=1$ 开始，第 k 步的任务是发现所有频繁 k 项集，具体包括

  ① 生成候选集C_k

  ② 过滤候选集，获得频繁项集L_k



生成候选集

- 将两个 k 项集求并集，得到一个大小不小于 k 的项集

$$\{x_1, x_2\}, \{x_2, x_4\} \rightarrow \{x_1, x_2\}$$
$$\{x_1, x_2\}, \{x_3, x_4\} \rightarrow \{x_1, x_2, x_3, x_4\}$$

- 基于 k 项集，Apriori算法只需要生成 k 项集
  - 为提高效率，将相机中的特征按照特名进行排序
  - 如果两个 k 项集的前 $(k-1)$ 个元素相同，只有第 k 个元素不同才将二者合并，生成候选集 C_{k+1}

过滤候选集

博雅大数据学院 BOYA BIG DATA COLLEGE

- 生成候选集C_{k+1}后，重新遍历数据集，计算候选集中每个项集的支持度
- 根据预置的支持度阈值
  - 过滤实际支持度达不到要求（小于阈值）的$(k+1)$项集
  - 获得频繁$(k+1)$ 项集 L_{k+1}

Apriori实例

- 在如下所示的手机评论数据集上运行Apriori算法，挖掘频繁项集
- 给定支持度阈值为0.6，置信度阈值为0.5
- 由单个特征构成的项集

$$C_1 = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$$

- 根据支持度阈值，过滤掉$\{x_1\}, \{x_5\}$

  得到频繁1项集

$$L_1 = \{\{x_1\}, \{x_3\}, \{x_4\}\}$$

评论编号	项集
1	$\{x_1, x_2, x_3, x_4\}$
2	$\{x_2, x_5\}$
3	$\{x_2, x_3\}$
4	$\{x_1, x_3, x_4\}$
5	$\{x_3, x_4, x_5\}$

Apriori算法实例

- 根据频繁1项集生成候选集

$$C_2 = \{\{x_2, x_3\}, \{x_3, x_4\}, \{x_2, x_4\}\}$$

- 遍历数据集，候选集中各个项集的支持度

项集	支持度计数
$\{x_2, x_3\}$	2
$\{x_3, x_4\}$	3
$\{x_2, x_4\}$	1

- C_2 中只有 $\{x_3, x_4\}$ 的支持度符合要求，由此得到频繁2项集 $L_2 = \{\{x_3, x_4\}\}$
- 由于 L_2 中只有一个项集，不再生成候选集，Apriori算法结束

Apriori算法实例

博雅大数据学院 BOYA BIG DATA COLLEGE

- 通过生成的所有频繁项集，生成频繁规则，进而发现强关联规则
- 依据 $\{x_3, x_4\}$，得到规则 $x_3 \Rightarrow x_4, x_4 \Rightarrow x_3$

项集	支持度计数
$\{x_1\}$	2
$\{x_2\}$	3
$\{x_3\}$	4
$\{x_4\}$	3
$\{x_5\}$	2

项集	支持度计数
$\{x_2, x_3\}$	2
$\{x_3, x_4\}$	3
$\{x_2, x_4\}$	1

- 计算置信度

$$C(x_3 \Rightarrow x_4) = \frac{S(\{x_3, x_4\})}{S(\{x_3\})} = 3/4 = 0.75$$

$$C(x_4 \Rightarrow x_3) = \frac{S(\{x_3, x_4\})}{S(\{x_4\})} = 4/4 = 1$$

$x_3 \Rightarrow x_4, x_4 \Rightarrow x_3$ 为强关联规则

Apriori算法小结

博雅大数据学院
BOYA BIG DATA COLLEGE

- Apriori算法是数据挖掘领域最有代表性的成果之一
- 频繁项集的性质：如果某项集不是频繁项集，那么它的所有超集也不是频繁项集
- Apriopri算法的主要步骤
  - 生成候选集
  - 过滤候选集

目录

FP-Growth算法

博雅大数据学院
BOYA BIG DATA COLLEGE

- Apriori算法
  - 产生大量无用的候选项集
    - 1000个频繁1项集生成至少 $\binom{1000}{2}$ 个候选2项集
  - 需要多次遍历数据集
- 有没有一种能够发现频繁项集而不产生候选项集的方式?
  - 频繁模式增长（Frequent Pattern Growth，FP-Growth）

- FP-Growth算法
  - 是关联规则挖掘的经典算法之一，于2000年由韩家炜、裴健等提出
  - 依靠一种能够压缩原始数据的**频繁模式树**（Frequent Pattern Tree，FP-tree）
  - 采用分治的思想，算法过程主要分为两部分

    ① FP-tree的构建

    ② 基于FP-tree进行递归挖掘频繁项集

韩家炜

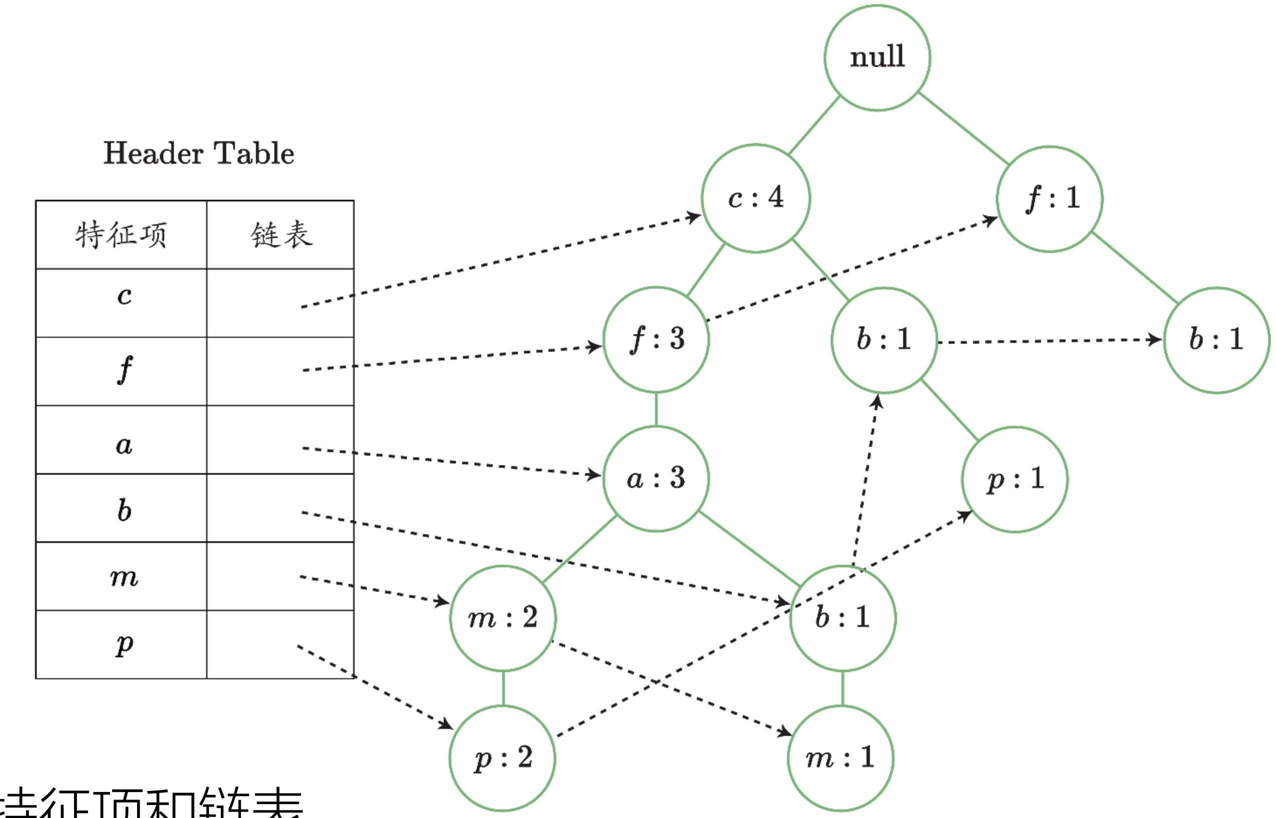
http://hanj.cs.illinois.edu

裴健

http://www.cs.sfu.ca/~jpei/

FP-tree构建过程

- FP-tree
  - 将数据集以树结构的方式进行存储
    - 根节点为null
    - 其他节点包含特征项和支持度信息
  - 快速高效进行频繁项集挖掘
    - 引入频繁项表（header table），包含特征项和链表
      - 频繁项表中的记录按照特征项支持度将序排列
      - 链表连接树中含相同特征项的节点



FP-tree构建过程

- 以商品交易数据为例，展示FP-tree的建树过程
- 数据集如下所示，给定支持度的阈值为0.6

交易记录	项
001	$\{a, c, f, d, g, m, i, p\}$
002	$\{a, c, b, m, o, f, l\}$
003	$\{j, o, b, h, f\}$
004	$\{b, c, s, k, p\}$
005	$\{a, c, e, f, l, n, m, p\}$

FP-tree构建过程

博雅大数据学院
BOYA BIG DATA COLLEGE

交易记录	项
001	$\{f, a, c, d, g, m, i, p\}$
002	$\{a, c, b, m, o, f, l\}$
003	$\{j, o, b, h, f\}$
004	$\{b, c, s, k, p\}$
005	$\{a, c, e, f, l, n, m, p\}$

特征项	支持度
c	4
f	4
a	3
b	3
m	3
p	3
l	2
o	2
d	1
e	1
g	1
h	1
i	1
j	1
k	1
n	1

- 首先，遍历数据集，统计每个特征项的支持付信息，并将特征项按照降序排列

FP-tree构建过程

- 数据集包含5行数据记录，支持度阈值为0.6，因此淘汰支持度小于3的特征项
- 数据集的频繁1项集为

$$L_1 = \{c:4, f:4, a:3, b:3, m:3, p:3\}$$

特征项	支持度
c	4
f	4
a	3
b	3
m	3
p	3

FP-tree构建过程-建树

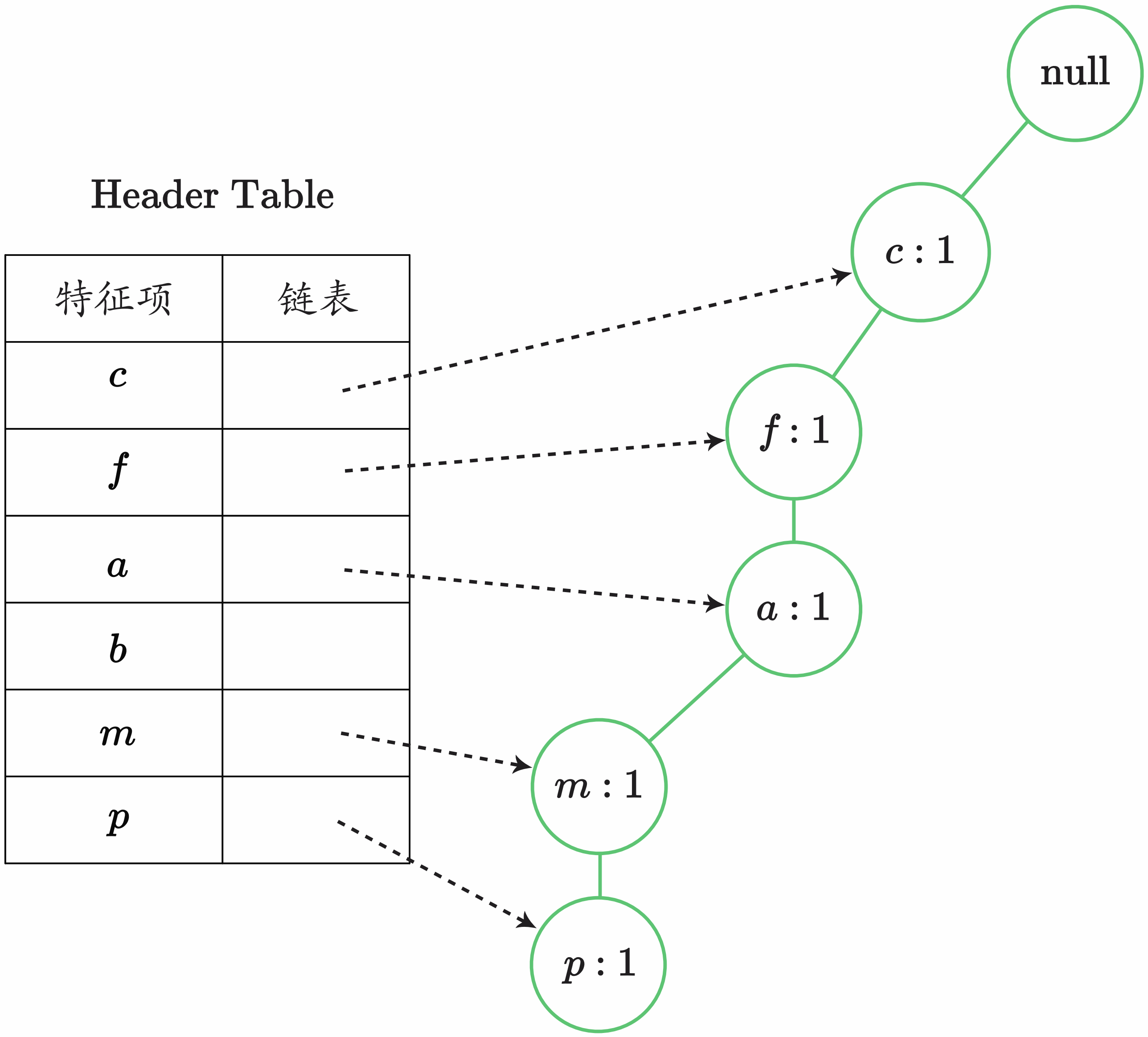
Header Table

特征项	链表
c	
f	
a	
b	
m	
p	

① 创建FP-tree

- 构造根节点null
- 根据频繁1项集建立对应的频繁项表，其中链表列内容为空

FP-tree构建过程-插入记录



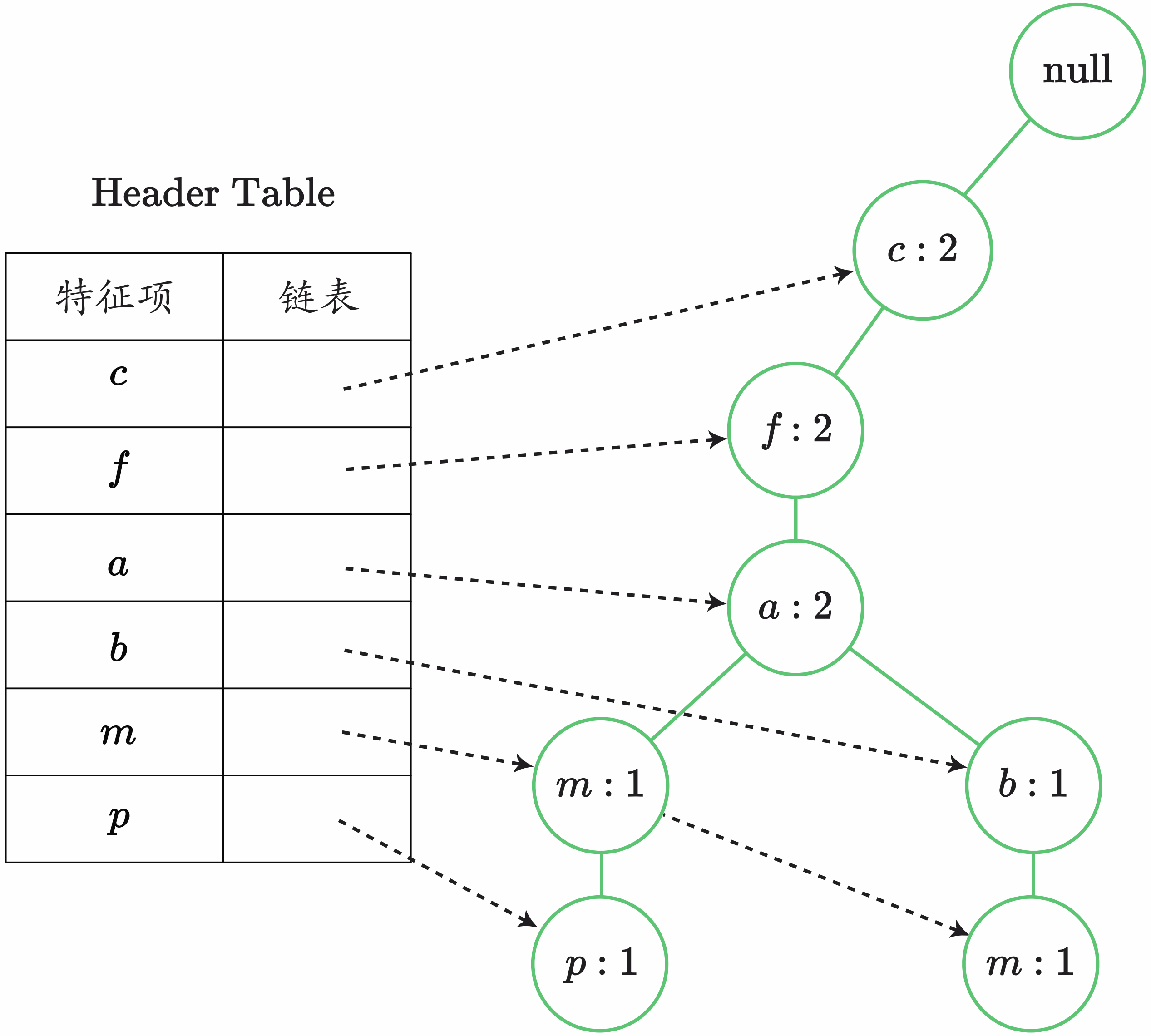
② 针对ID为001的交易记录

- 根据 L_1 提取频繁特征项并降序排列

$$\{f, a, c, d, g, m, i, p\} \rightarrow \{c, f, a, m, p\}$$

- 将特征项依次插入FP-tree中
  - 树不包含要插入的特征项节点，需创建新的节点，支持度取值初始化为1

FP-tree构建过程-插入记录



③ 针对ID为002的交易记录

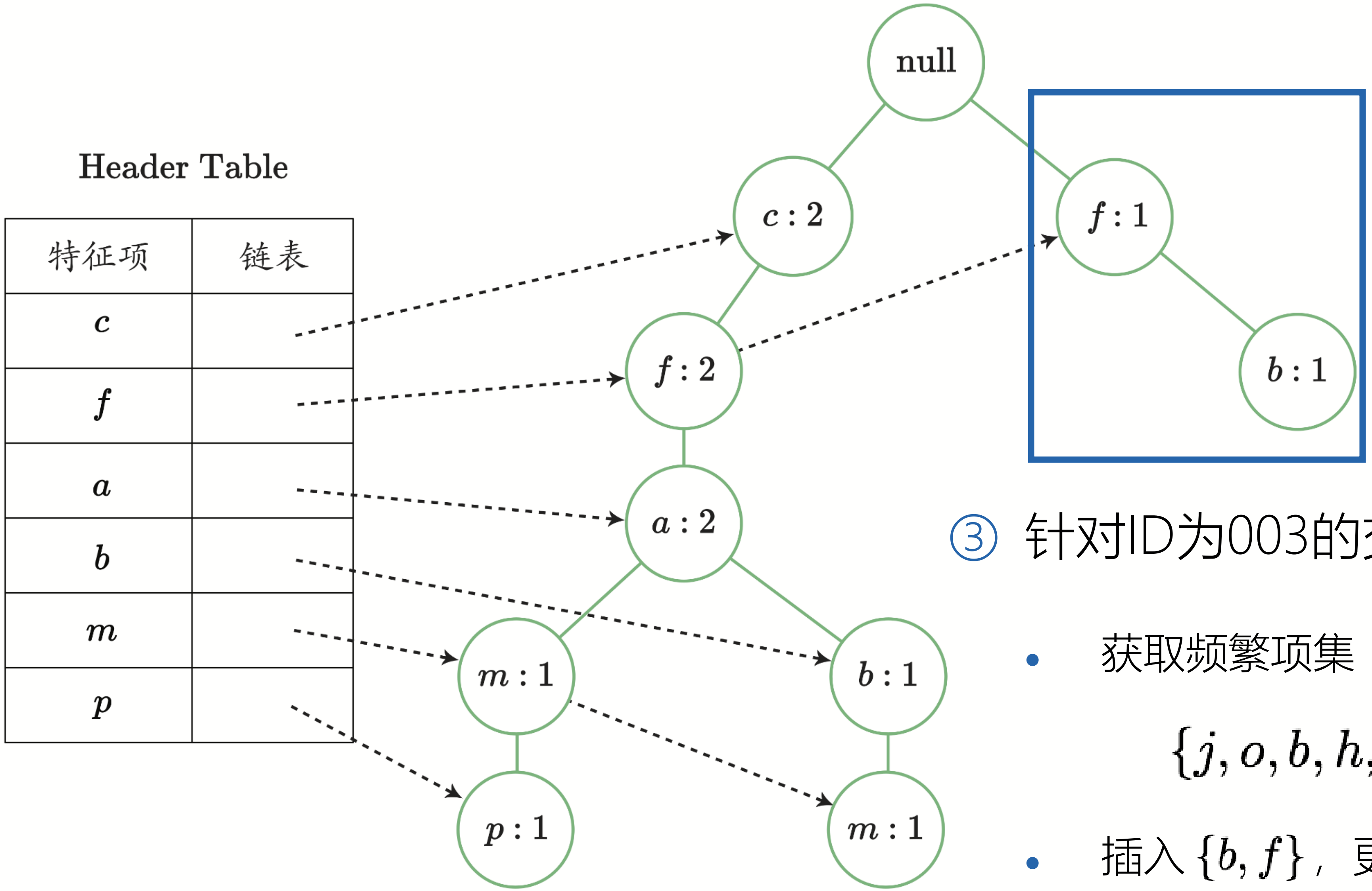
- 获取将序排列的频繁项集

$$\{a, c, b, m, o, f, l\} \rightarrow \{c, f, a, b, m\}$$

- 与001交易记录共享前缀cfa，更新共享节点的支持度

- 对于bm节点，重新创建树中节点，更新频繁项表的链接

FP-tree构建过程-插入记录



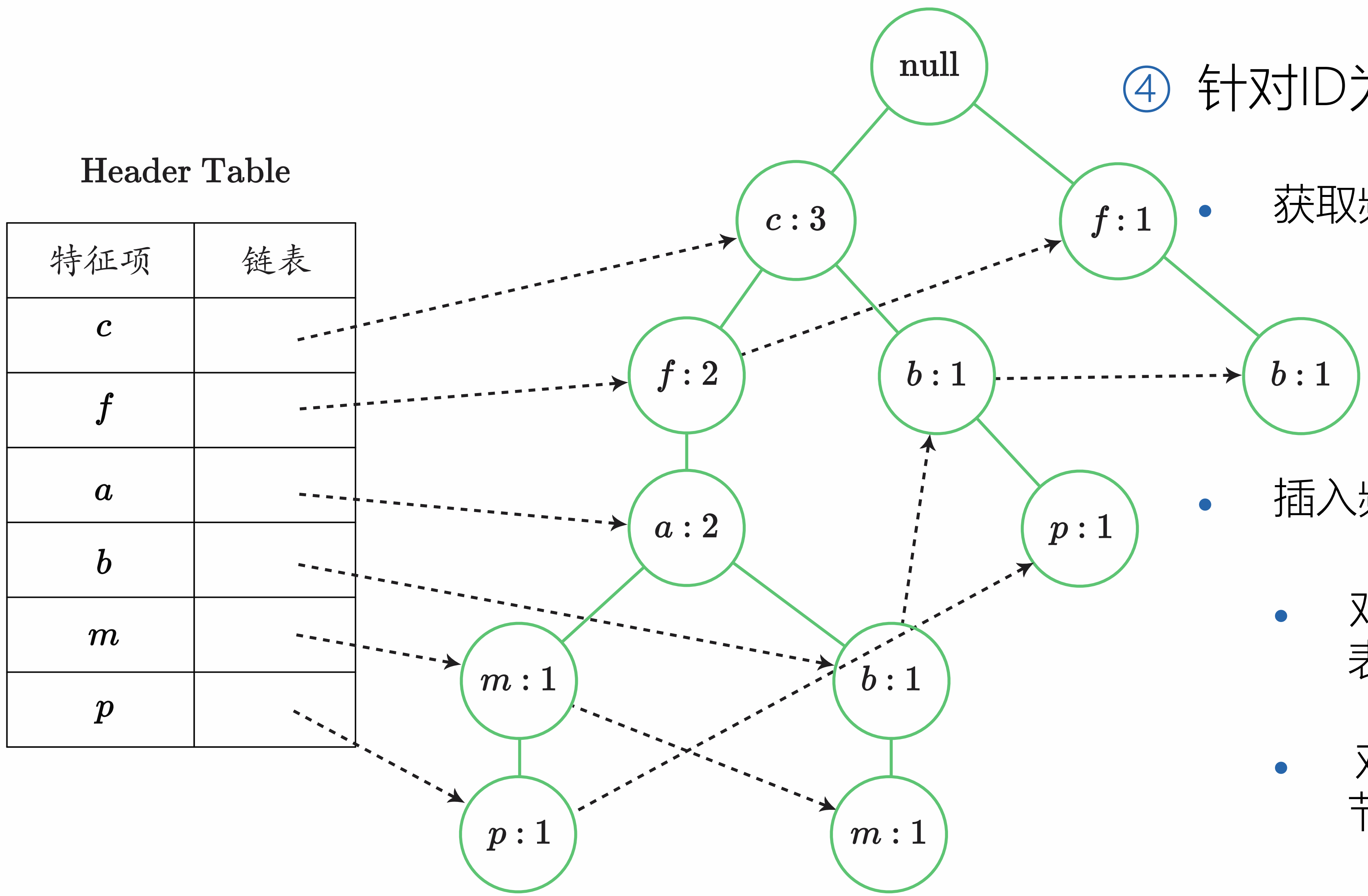
③ 针对ID为003的交易记录

- 获取频繁项集

$$\{j, o, b, h, f\} \rightarrow \{b, f\}$$

- 插入 $\{b, f\}$，更新节点 f 的链表

FP-tree构建过程



④ 针对ID为004的交易记录

- 获取频繁项集

$$\{b, c, s, k, p\} \rightarrow \{c, b, p\}$$

- 插入频繁项集
  - 对于共享前缀 c，更新频繁项表中的支持度
  - 对于 bp 创建新节点，更新节点的链表内容

FP-tree构建-插入记录

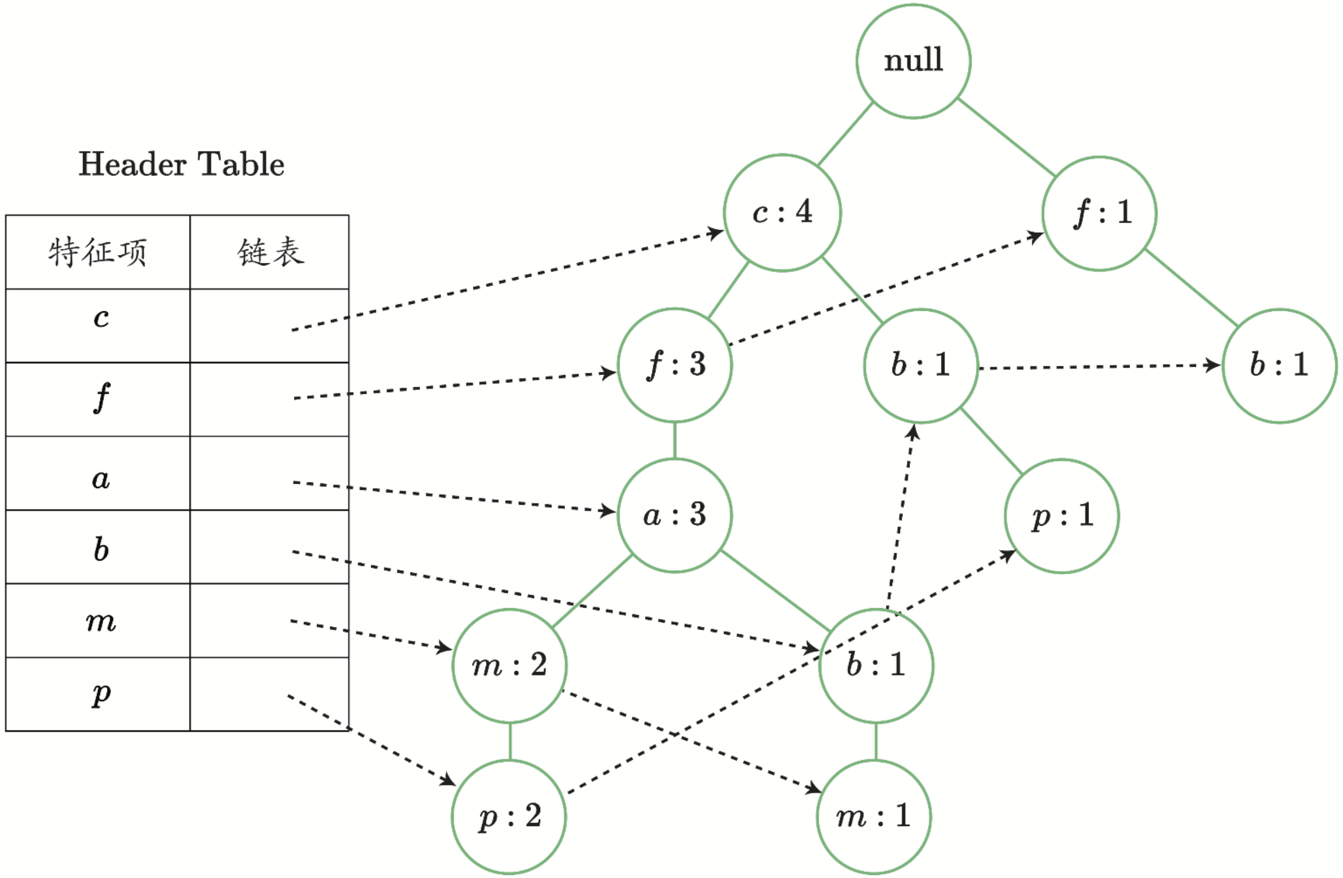
- 插入ID为005的交易记录后，FP-tree构建完成

④ 针对ID为005的交易记录

- 频繁项集

$$\{a, f, c, e, l, p, m, n\} \rightarrow \{c, f, a, m, p\}$$

- 更新FP-tree相关节点的支持度计数



FP-tree构建算法

- 从前面的具体实例中，我们发现FP-tree的如下特点

  ① 由于建树过程中，数据记录共享前缀节点，树的大小远小于原始数据集

  ② FP-tree的节点存储了支持度技术的信息，有利于挖掘频繁项集

- 因此，使用FP-tree进行频繁项集挖掘，避免频繁访问遍历数据集

FP-tree构建算法

博雅大数据学院
BOYA BIG DATA COLLEGE

- 通过数据集示例，给出FP-tree的构建算法（给定数据集 D 和支持度阈值 α）：

  ① 遍历数据集，计算特征项的支持度并降序排列，通过支持度阈值α淘汰不频繁的项，获得频繁1项集 L_1

  ① 创建FP-tree和对应的频繁项表，并按照如下方法遍历数据集

  - 对于D 中的样本，执行如下步骤：
    - 按照L_1 生成样本中的频繁项，并排序
    - 将排序过的频繁项集插入到树中，更新节点的支持度计数
    - 更新频繁项表中的链表内容

基于FP-tree挖掘频繁项集

博雅大数据学院
BOYA BIG DATA COLLEGE

- 基于构建完成的FP-tree，FP-Growth算法采用分治的思想
  - 将FP-tree划分为多个条件模式库（Conditional Pattern Base， CPB）
  - 依据每个CPB，构建条件频繁模式树(Conditional FP-tree)
    - 与特定后缀结尾的项集有关联
  - 以递归方式对条件FP-tree独立挖掘频繁项集
    - 将挖掘频繁项集的任务拆分为一些子频繁项集挖掘任务

- FP-tree如右图所示，生成以p 为后缀的CPB
  - 根据频繁项表中p 的链表结构，寻找路径

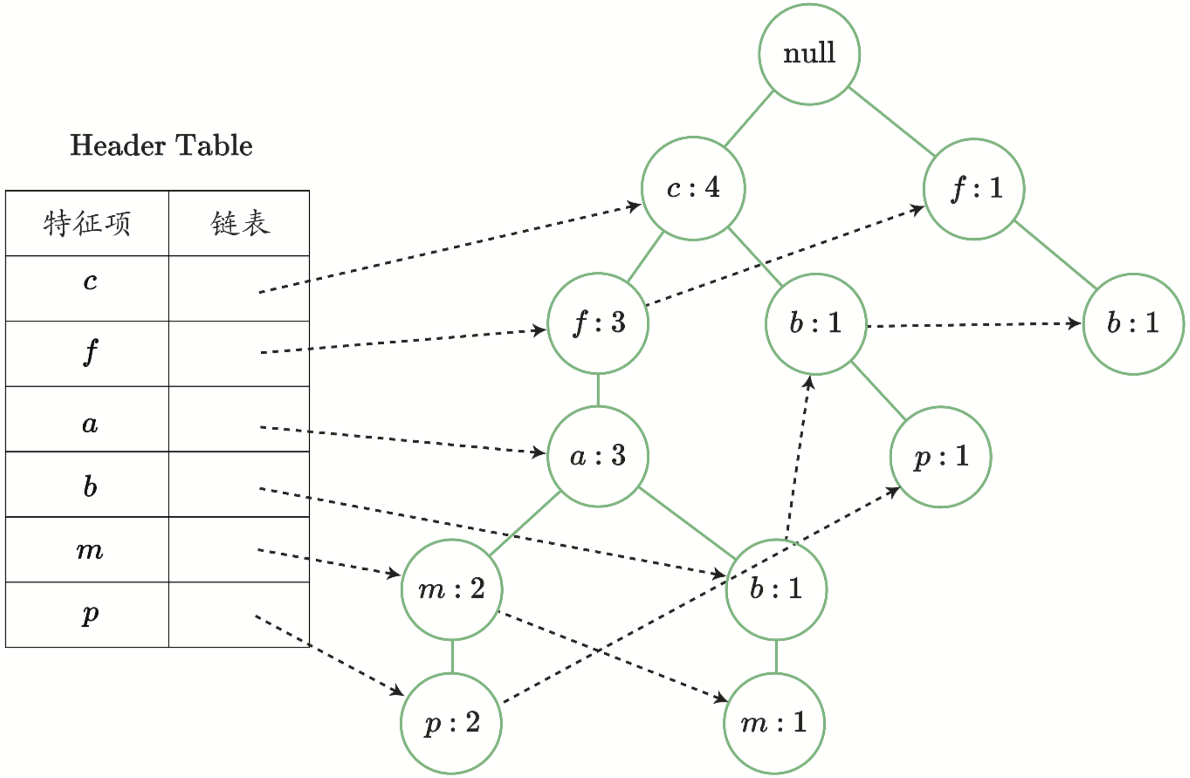
$$\{cfamp, cbp\}$$

  - 去除路径中的节点p，剩余节点组成新的项集

$$\{cfam, cb\}$$

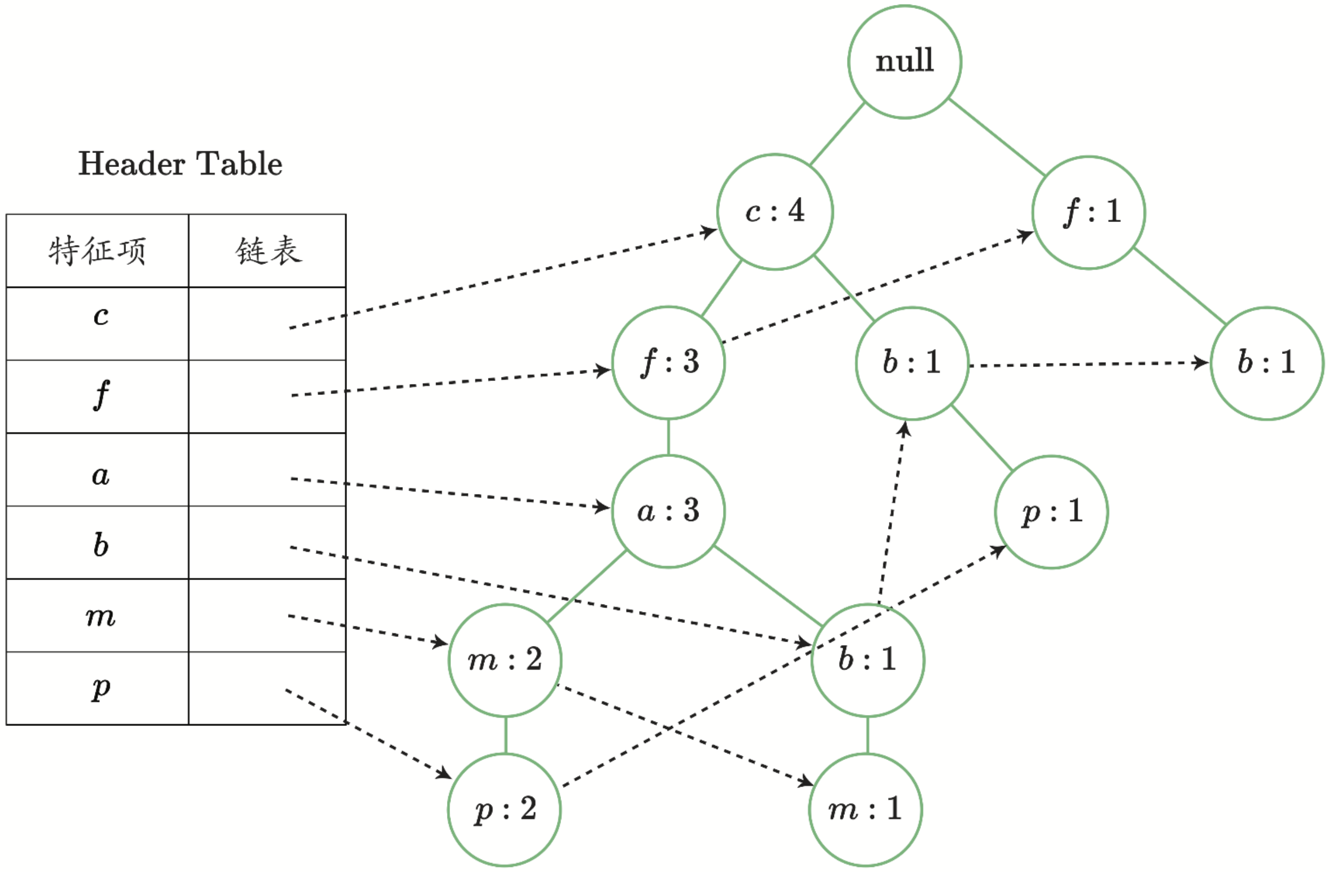
  - 项集的支持度为节点p 的支持度，那么CPB为

$$\{cfam:2, cb:1\}$$



条件模式库

- 现在，我们获取频繁项表中特征项的CPB



条件项	条件模式库(CPB)
c	$\{\}$
f	$\{c: 3\}$
a	$\{cf: 3\}$
b	$\{cfa: 1, c: 1$
m	$\{cfa: 2, cfab: 1\}$
p	$\{cfam: 2, cb: 1\}$

条件频繁模式树

博雅大数据学院 BOYA BIG DATA COLLEGE

- 基于条件模式库，使用FP-tree构建算法构造条件频繁模式树

- 例如，节点p 的条件模式库为 $\{cfam : 2, cb : 1\}$
  - 使用支持度阈值过滤不符合条件的项集，最终的结果为 $\{c : 3\}$
  - 构建完成的树结构仅包含有一个根节点和节点c

FP-Growth算法过程

博雅大数据学院
BOYA BIG DATA COLLEGE

算法 FP-Growth 算法

输入: 数据集 D 对应的 FP-tree, 支持度阈值 min_suppt.

输出: 频繁项集.

procedure FP-GROWTH(Tree, α) ▷ Tree 为 FP-tree, α 为当前的后缀
  if Tree 包含一个单独路径的前缀 **then** ▷ 挖掘单路径
    假设 P 为 Tree 的单独前缀路径部分,
    将 Tree 第一个分叉节点标记为 "null" 后作为根节点的子树记为 Q,
    for 路径 P 中特征项组成的每一个非空组合 β **do**
      生成一个频繁项集 $\beta \cup \alpha$, 其支持度计数等于 α 的支持度计数.
    将通过 P 生成的所有频繁项集记为 set(P).
  else
    将 Tree 记为 Q.
  for Q 中的每一个特征项 a_i **do** ▷ 挖掘多路径
    生成频繁项 $\beta = a_i \cup \alpha$, 将其支持度计数设置为 β 节点的最小支持度计数,
    构建 β 的条件频繁模式树 Tree_β.
    if Tree_β 不为空 **then**
      调用 $\text{FP} - \text{GROWTH}(\text{Tree}_\beta, \beta)$.
  将通过 Q 生成的频繁项集记为 set(Q).
  return $\text{set}(P) \cup \text{set}(Q) \cup (\text{set}(P) \times \text{set}(Q))$

关联规则生成

- 与Apriori算法相同，FP-Growth算法的结果是**频繁项集**，而不是**关联规则**
- 为了进一步生成关联规则

  ① 根据频繁项集生成关联规则

  ② 根据预置的置信度指标，**过滤**第一步的关联规则，生成**强**关联规则

FP-Growth算法过程

算法 6 FP-Growth 算法

输入: 数据集 D 对应的 FP-tree, 支持度阈值 min_suppt.

输出: 频繁项集.

procedure FP-GROWTH(Tree, α) ▷ Tree 为 FP-tree, α 为当前的后缀
  if Tree 包含一个单独路径的前缀 **then** ▷ 挖掘单路径
    假设 P 为 Tree 的单独前缀路径部分,
    将 Tree 第一个分叉节点标记为 “null” 后作为根节点的子树记为 Q,
    for 路径 P 中特征项组成的每一个非空组合 β **do**
      生成一个频繁项集 $\beta \cup \alpha$, 其支持度计数等于 α 的支持度计数.
    将通过 P 生成的所有频繁项集记为 set(P).
  else
    将 Tree 记为 Q.
  for Q 中的每一个特征项 a_i **do** ▷ 挖掘多路径
    生成频繁项 $\beta = a_i \cup \alpha$, 将其支持度计数设置为 β 节点的最小支持度计数,
    构建 β 的条件频繁模式树 Tree_β.
    if Tree_β 不为空 **then**
      调用 FP − GROWTH(Tree_β, β).
  将通过 Q 生成的频繁项集记为 set(Q).
  return $\text{set}(P) \cup \text{set}(Q) \cup (\text{set}(P) \times \text{set}(Q))$

FP-Growth算法小结

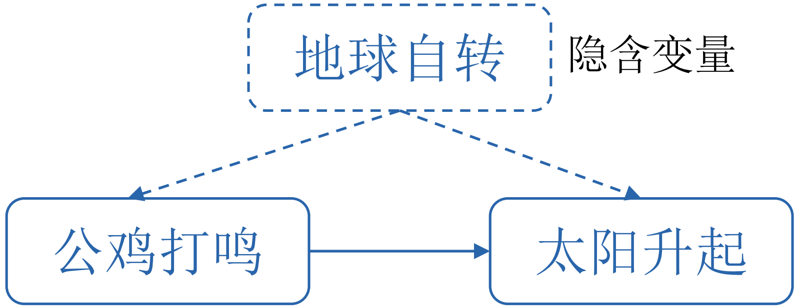
- FP-Growth算法
  - 遍历2次数据集，将原始数据集压缩成树型结构FP-tree
  - 采用分治的思想，对构造的条件FP-tree进行递归挖掘
- 与Apriori算法相同，FP-Growth算法的输出结果是频繁项集，而不是关联规则
- FP-Growth算法不产生无用的候选项集

目录

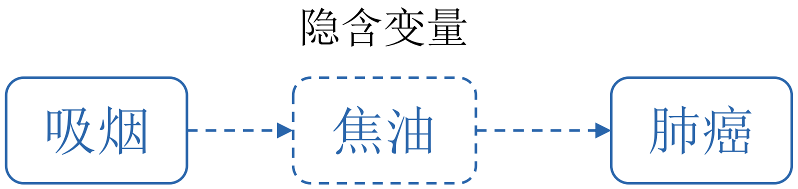
- 关联规则概述
- Apriori算法
- FP-Growth算法
- **辛普森悖论**

相关关系与因果

- 相关关系 != 因果关系，但相关关系的背后可能蕴含着某种因果
  - E.g., 公鸡打鸣 => 太阳升起（相关但非因果）
  - 总有些“分析师”会拿相关关系作为因果解释忽悠人，e.g., 金融危机->房价暴涨
- 解释变量之间的关联时要特别小心，因为观察到的联系可能受其他混淆因素的影响。（推荐因果科普书 Judea Pearl《The Book of Why》）
- 辛普森悖论：没有包括在分析中的隐藏变量，可能导致观察到的一对变量之间的相关联系**消失或逆转方向**



英国统计学家E.H.辛普森于1951年提出的悖论



辛普森悖论-举例

博雅大数据学院 BOYA BIG DATA COLLEGE

• 辛普森悖论：没有包括在分析中的隐藏变量，可能导致观察到的一对变量之间的相关联系**消失或逆转方向**

买 HDTV	买健身器		
	是	否	
是	99	81	180
否	54	66	120
	153	147	300

数据总体统计：

{买HDTV=是}→{买健身器=是}: 99/180 = 55%

>

{买HDTV=否}→{买健身器=是}: 54/120 = 45%

顾客组	买 HDTV	买健身器		总数
		是	否	
大学生	是	1	9	10
	否	4	30	34
在职人员	是	98	72	170
	否	50	36	86

对于大学生：

{买HDTV=是}→{买健身器=是}: 1/10 = 10%

<

{买HDTV=否}→{买健身器=是}: 4/34 = 11.8%

对于在职人员：

{买HDTV=是}→{买健身器=是}: 98/170 = 57.7%

<

{买HDTV=否}→{买健身器=是}: 50/86 = 58.1%

辛普森悖论-举例

- **辛普森悖论：虽然$a/b<c/d$ 且 $p/q<r/s$，**
  但 $(a+p)/(b+q)>=(c+r)/(d+s)$

即在分组比较中都占优势的一方，会在总评中反而是失势的一方。

- **适当的数据分层有助于避免辛普森悖论，比如：**
  - **比如：大型超市的购物篮数据应依据商店的位置分层；**
  - **学校录取数据应依据专业分层等**

辛普森悖论-实际例子

- UC Berkeley was sued for bias against women applying to graduate school.

Men		Women	
#Applicants	%Admitted	#Applicants	%Admitted
832	44%	366	11%

- In fact, most departments had a small bias against men

Major	Men		Women	
	#Applicants	%Admitted	#Applicants	%Admitted
B	560	63%	25	68%
F	272	6%	341	7%

适当的数据分层有助于避免辛普森悖论

- **Adapted from the example at http://en.wikipedia.org/wiki/Simpson's_paradox . See the following paper for more details: P.J. Bickel, E.A. Hammel and J.W. O'Connell (1975). "Sex Bias in Graduate Admissions: Data From Berkeley". Science 187 (4175): 398–404.**
- **https://en.wikipedia.org/wiki/Simpson%27s_paradox**