第五章 挖掘频繁模式、关联和相关

张静
（Jingzhang@ecust.edu.cn）

主要内容

- 基本概念
- 频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

频繁模式分析

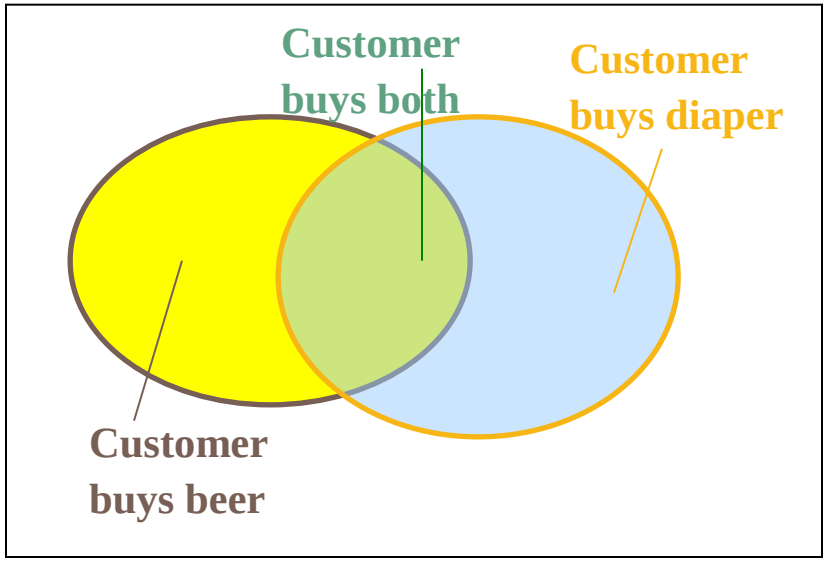
- 频繁模式: 数据集中频繁出现的模式 (一组项的集合，子序列，子结构等。)
- 最先由Agrawal, Imielinski, and Swami [AIS93] 在描述频繁项集和关联规则挖掘的相关文档中提出
- 动机: 找到数据之间的内在规律
  - 哪些产品经常一起出售?— 啤酒和纸尿裤?!
  - 用户在买了PC之后接下来很有可能买什么?
  - 哪种DNA对新药物敏感?
  - 我们能够自动地对网络文档分类么?
- 应用
  - 购物篮分析，交叉市场分析，Web日志分析以及DNA序列分析等。

频繁模式挖掘的重要性

- 频繁模式: 数据集本质的并且重要的性质
- 很多重要数据挖掘工作的基础
  - 关联，相关以及因果分析
  - 序列的, 结构化的 (例如，子图) 模式
  - 在时空、多媒体、时间序列以及流数据上的模式分析
  - 分类：区分性，频繁模式分析
  - 聚类分析：基于频繁模式的聚类
  - 数据仓库：冰立方以及立方体聚集
  - 语义数据压缩

基本概念：频繁项集

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



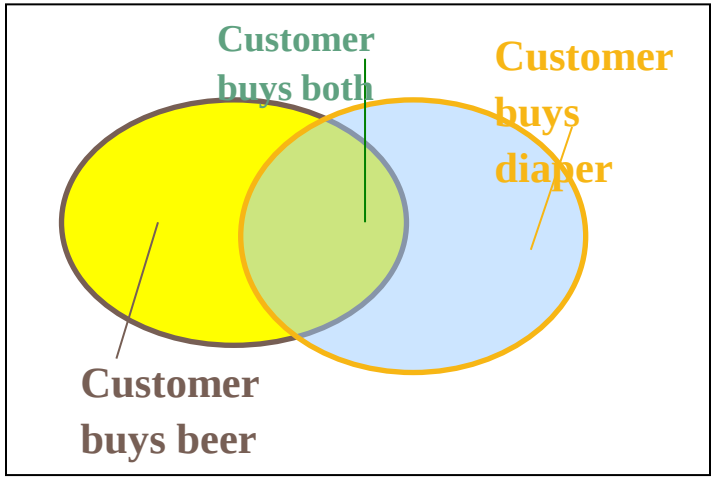
- 项集: 一个或多个项的集合
- **k-itemset** $X = \{x_1, \ldots, x_k\}$
- ***(absolute) support***, or, ***support count*** of X: 项集 X出现的频率
- ***(relative) support***, s, 事务中包含 X的部分 (即，一个事务包含 X的概率)
- 项集X 是频繁的（ ***frequent***），如果X的支持度不小于支持度阈值

基本概念：关联规则

- 假设: (1) 事务数据库, (2) 每一个事务是一个项目集列表(例如：顾客一次购买的商品集)
- 找出：所有使目前的项集与另一个项集相关联的规则
  - 例子：
    - 规则形式: “Body →Head [支持度,置信度]”.
    - buys(x, “diapers”) →buys(x, “beers”) [0.5%, 60%]
    - major(x, “CS”) ^ takes(x, “DB”) →grade(x, “A”) [1%, 75%]
- 应用
  - * ⇒维修协议(商店应该怎样做才能提升维修协议的销售)
  - 家电⇒*(商店应该增加其它那些产品的存储量?)

基本概念：关联规则

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- 找到所有满足最小支持度和置信度阈值的规则 $X \rightarrow Y$
  - **support**, s, 一个事务包含 $X \cup Y$ 的概率
  - **confidence**, c, 一个事务既包含X也包含Y的条件概率
- 令 minsup = 50%, minconf = 50%
  频繁模式： Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3
  - 关联规则:
    - *Beer → Diaper* (60%, 100%)
    - *Diaper → Beer* (60%, 75%)

极大项集与频繁闭项集（Closed Patterns and Max-Patterns）

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{1}{1}\binom{0}{0}\binom{0}{0} = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* $Y \supset X$, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern $Y \supset X$ (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

极大项集与频繁闭项集

- 极大项集和频繁闭项集
  - 极大项集：频繁项集**p**，使得**p**的任何超项集都不是频繁的。
  - 频繁闭项集：一个频繁的闭的项集，其中项集**c**是闭的，如果不存在**c**的真超集**c'**，使得每个包含**c**的事务也包含**c'**。

频繁模式挖掘的分类体系

- 根据挖掘的模式的完全性分类
  - 频繁项集的完全集、闭频繁项集、极大频繁项集
  - 被约束的频繁项集、近似的频繁项集、接近匹配的频繁项集、最频繁的*k*个项集
- 根据规则中所处理的值类型分类
  - 布尔关联和量化关联
    - buys(x, “SQLServer”) ^ buys(x, “DMBook”) →buys(x, “DBMiner”) [0.2%, 60%]
    - age(x, “30..39”) ^ income(x, “42..48K”) →buys(x, “PC”) [1%, 75%]
- 根据规则中所涉及的数据维数分类
  - 单维关连和多维关联

频繁模式挖掘的分类体系

- 根据规则集所涉及的抽象层分类
  - 单层关联规则和多层关联规则
    - age(x, “30..39”) →buys(x, “laptop computer”)
    - age(x, “30..39”) →buys(x, “computer”)
- 根据所挖掘的规则类型分类
  - 关联规则、相关规则、强梯度联系（与父母、子女或兄妹之间的关系）
  - 相关性和因果关系分析
    - 关联并不一定必须意味着相关性和因果性
- 根据所挖掘的模式类型分类
  - 频繁项集挖掘、序列模式挖掘、结构模式挖掘

关联规则基本模型

- 关联规则就是支持度和置信度分别满足用户给定阈值的规则。
- 发现关联规则需要经历如下两个步骤：
  - 找出所有频繁项集。
  - 由频繁项集生成满足最小置信度阈值的规则。

例子

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule $A \Rightarrow C$:

support = support($\{A\} \cup \{C\}$) = 50%

confidence = support($\{A\} \cup \{C\}$)/support($\{A\}$) = 66.6%

主要内容

- 基本概念
- 频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

Apriori算法：通过限制候选产生发现频繁项集

- Apriori算法是R. Agrawal和R. Srikant于1994年提出的为布尔关联规则挖掘频繁项集的原创性算法。
- 算法的思想
  - 使用频繁项集性质的先验知识
  - 使用逐层搜索的迭代方法产生频繁项集
- Apriori性质
  - 频繁项集的所有非空子集也必须是频繁的

频繁项集

- 为了避免计算所有项集的支持度（实际上频繁项集只占很少一部分），Apriori算法引入候选频繁项集的概念。
- 若候选频繁k项集的集合记为C_k，频繁k项集的集合记为L_k，m个项构成的k项集的集合为C_m^k，则三者之间满足关系$L_k \subseteq C_k \subseteq C_m^k$。
- 可以根据Apriori性质生成潜在的频繁项集。

Apriori算法

- 连接步：通过连接产生C_k
- 剪枝步：如果一个候选k-项集的(k-1)-子集不在(k-1)-的频繁项集中,则该候选项集也不可能是频繁的,从而由C_k删除
- 伪代码
  - C_k：大小为k的候选集
  - L_k：大小为k的频繁项集
  - L_1= {频繁项集};
    - for(k= 1; L_k!=Ø; k++) do begin
      - C_{k+1}= 从L_k中产生的候选集;
    - for each transaction t∈D do
      - 对于包含在t中的属于C_{k+1}的所有候选集的计数加一
    - L_{k+1}= C_{k+1}中满足最小支持度的候选集
  - end
  - return $\cup_k L_k$;

实例 (support : 50%)

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

C_1

1st scan

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

如何产生候选集

- 假设L_{k-1}中的项是按顺序列出的
- 第一步：自我连接L_{k-1}
  - Insert into C_k
    - select p.item$_1$, p.item$_2$, …, p.item$_{k-1}$, q.item$_{k-1}$
  - from L_{k-1}p, L_{k-1} q
    - where p.item$_1$=q.item$_1$, …, p.item$_{k-2}$=q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$
- 第二步：剪枝步
  - 对于C_k中的所有项集c do
    - 对于c 的所有(k-1)子集s do
      - if (s 不在L_{k-1}中) then 从C_k中删除c

产生候选项集的例子

- L_3={abc, abd, acd, ace, bcd}
- 自我连接: L_3*L_3
  - 由abc和abd产生abcd
  - 由acd和ace产生acde
- 剪枝:
  - 删除acde因为ade不在L_3
  - C_4={abcd}

由频繁项集产生关联规则

- 方法
  - 对于每个频繁项集**l**，产生 **l** 的所有非空子集。
  - 对于**l** 的每个非空子集**S**，如果 $\frac{\sup port\_count(l)}{\sup port\_count(s)} \geq \min\_conf$ 则输出规则“ $s \Rightarrow (l-s)$ ”。其中，***min_conf*** 是最小置信度阈值。
- 例子
  - 频繁项集 $l=\{I1, I2\}$，可以由**l**产生哪些关联规则？
  - **L**的非空子集：**{I1}**，**{I2}**
  - 关联规则如下：$I1 \Rightarrow I2$

    $I2 \Rightarrow I1$
  - 计算置信度，大于置信度阈值就是强关联规则

提高Apriori算法的效率

- 基于散列的技术
  - 如果一个k-项集对应的哈希桶计数低于支持度阈值,则它不可能是频繁的
- 事务压缩
  - 不包含任何频繁k-项集的事务在以后的扫描中是无用的
- 划分
  - 任何项集如果在DB中是潜在频繁的,那么它至少在DB中的一个划分中是频繁的
- 抽样
  - 在给定数据的一个子集上进行挖掘,低支持阈值+好的策略可以得到完整的频繁项集
- 动态项集计数
  - 如果一个项集的所有子集已被确定为频繁的,则添加它作为新的候选

不产生候选项集挖掘频繁集

- Apriori算法存在的问题
  - 大量的候选集
    - 10^4个频繁1-项集将生成 10^7个候选2-项集
    - 为了发现一个大小为100的频繁模式，如 $\{a_1, a_2, \ldots, a_{100}\}$，需要产生 $2^{100} \approx 10^{30}$个候选集
  - 数据库的多遍扫描
    - 假定n是最长项集的长度， 则需要(n+1)遍扫描数据库。

频繁模式增长

- 把大型数据库压缩成一棵**Frequent-Pattern tree (FP-tree) [Han, Pie and Yin, SIGMOD2000]**
  - 高度浓缩，同时对频繁集的挖掘又完备的
  - 避免代价较高的数据库扫描
- 开发一种高效的基于**FP-tree**的频繁集挖掘算法
  - 分而治之的策略：分解数据挖掘任务为小任务
  - 避免候选项集的产生：只检测子数据库

建立 FP-tree

TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

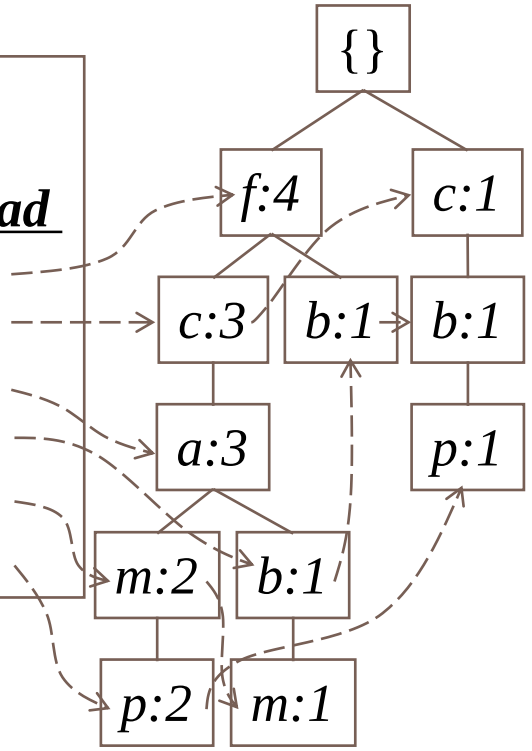
最小支持度 = 0.5

步骤:

1. 扫描数据库一次，得到频繁**1-**项集
2. 把项按支持度递减排序
3. 再一次扫描数据库，建立**FP-tree**

头表

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



FP-tree 结构的优点

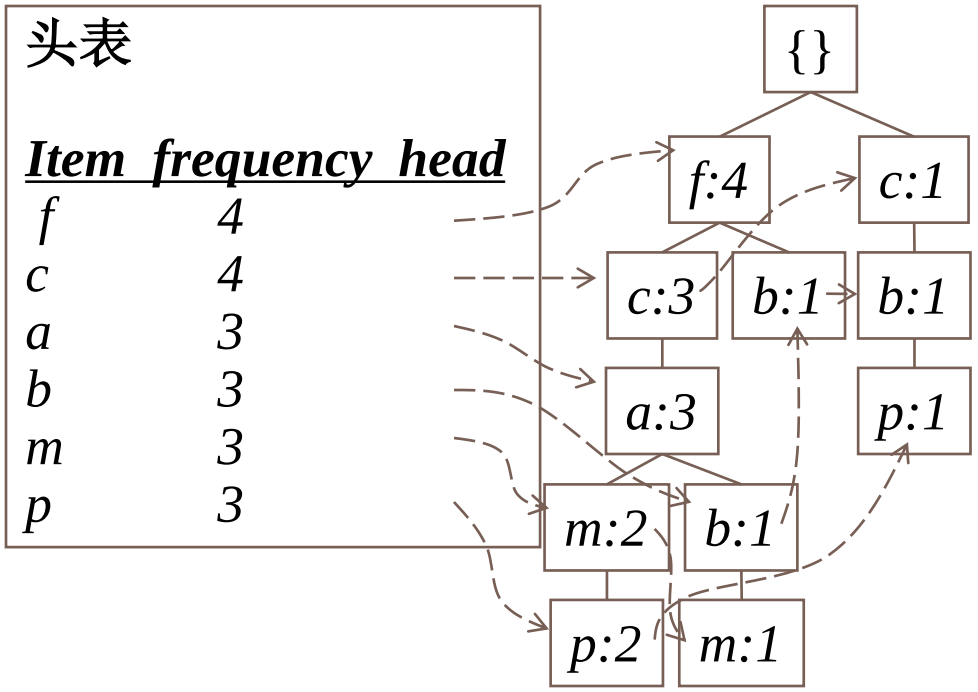
- 完整性
  - 不会破坏交易中的任何模式
  - 包含了频繁模式挖掘所需的全部信息
- 简洁性
  - 减少了不相关的信息-----非频繁项集被删掉
  - 频繁项集按支持度递减顺序排列:越是频繁的项集越有可能被共享
  - 不会比原数据库大(如果不算节点链和计数)

用FP-tree挖掘频繁项集

- 基本思想 (分而治之)
  - 用FP-tree递归增长频繁集
- 主要步骤
  - 为FP-tree中的每个节点生成条件模式基
  - 用条件模式基构造对应的条件FP-tree
  - 递归构造条件 FP-trees 同时生成其包含的频繁集
    - 如果条件FP-tree只包含一个路径，则直接生成所包含的频繁集。
    - 如果条件FP-tree包含多个路径，则采用混合的方法

步骤1: 从 FP-tree 到条件模式基

- 从FP-tree的头表开始
- 按照每个频繁项的连接遍历 FP-tree
- 列出能够到达此项的所有前缀路径，得到条件模式基



条件模式基

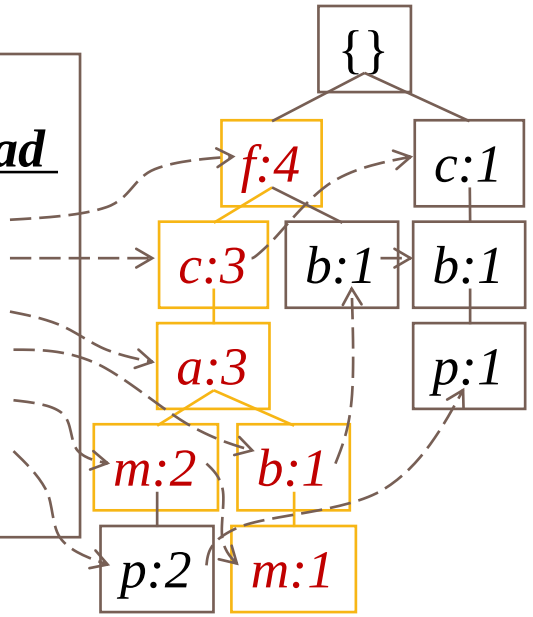
item	*cond. pattern base*
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

步骤2: 建立条件 FP-tree

- 对任意模式基
  - 计算其中每个项的支持度
  - 为模式基中的频繁项建立条件FP-tree

Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



m-条件模式基:
fca:2, fcab:1

设最小支持度计数为3

***m-conditional* FP-tree**

All frequent patterns concerning *m*

***m*,**

fm, cm, am,

fcm, fam, cam,

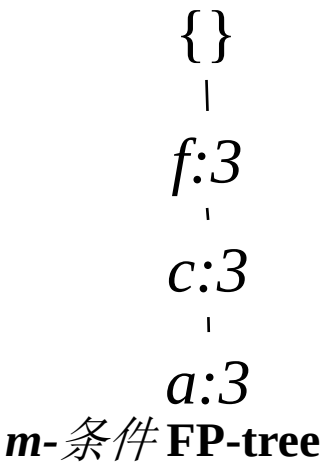
fcam

步骤3：生成条件FP-tree

设最小支持度计数为 3

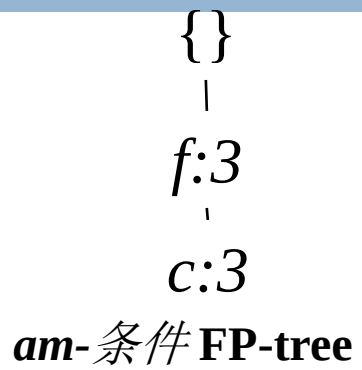
项	条件模式基	条件FP-tree
p	{(fcam:2), (cb:1)}	Empty
m	{(fca:2), (fcab:1)}	{(f:3, c:3, a:3)}|m
b	{(fca:1), (f:1), (c:1)}	Empty
a	{(fc:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Empty	Empty

步骤4：递归挖掘条件FP-tree



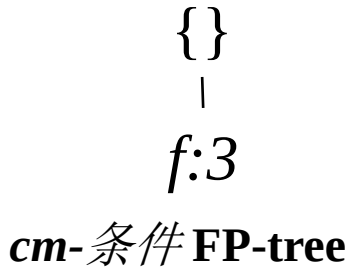
***m*-条件 FP-tree**

"am"的条件模式基: (fc:3)



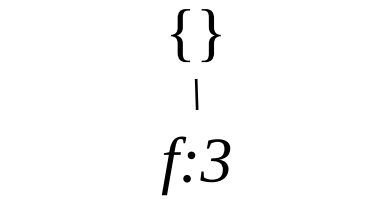
***am*-条件 FP-tree**

"cm"的条件模式: (f:3)



***cm*-条件 FP-tree**

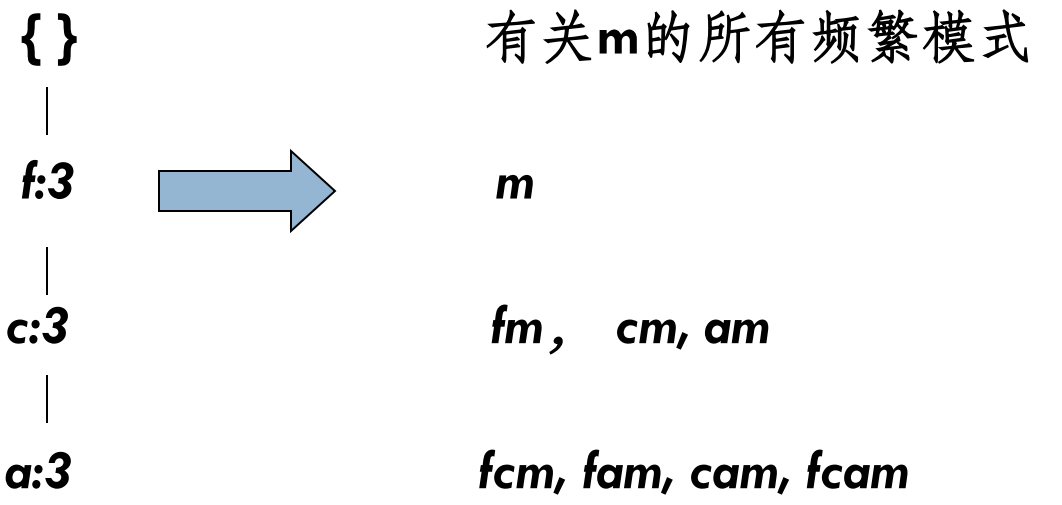
"cam"条件模式基: (f:3)



***cam*-条件 FP-tree**

单一FP-tree路径的产生

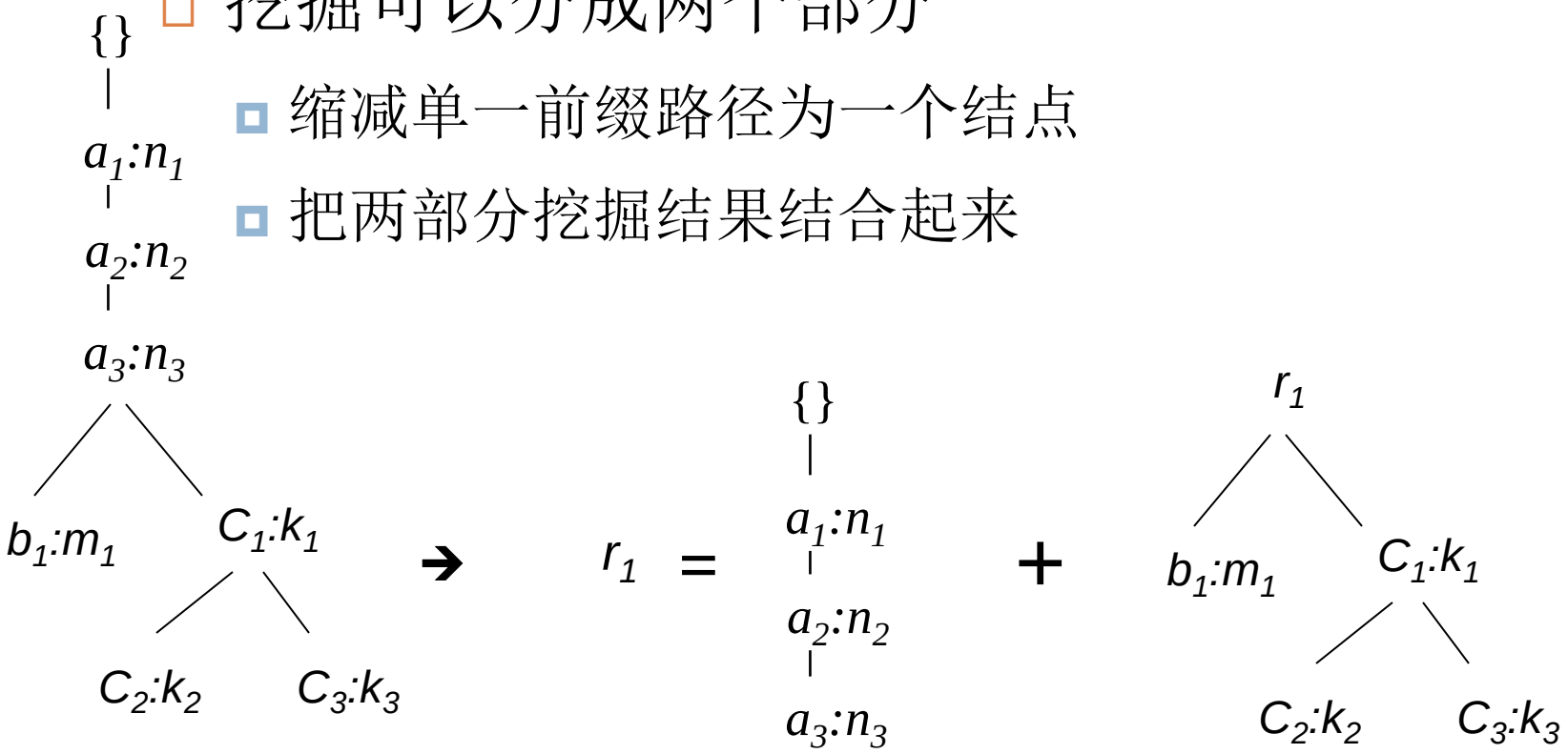
- 假设一个FP-tree T 有一个单一的路径P
  - T的完全频繁模式集可以通过列举P的子路径的所有组合来产生



m-条件FP-树

拥有单一前缀路径的FP-tree

- 假定一个条件FP-tree T 有一个共享的单一前缀路径 P
- 挖掘可以分成两个部分
  - 缩减单一前缀路径为一个结点
  - 把两部分挖掘结果结合起来



频繁模式增长的性能分析

- 性能研究表明
  - FP-树比Apriori算法快一个数量级
- 原因
  - 没有候选集的产生，没有候选测试
  - 使用简洁的数据结构
  - 除去了重复的数据库扫描
  - 基本操作是计数和FP-树构造

主要内容

- 基本概念
- 频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

挖掘各种类型的关联规则

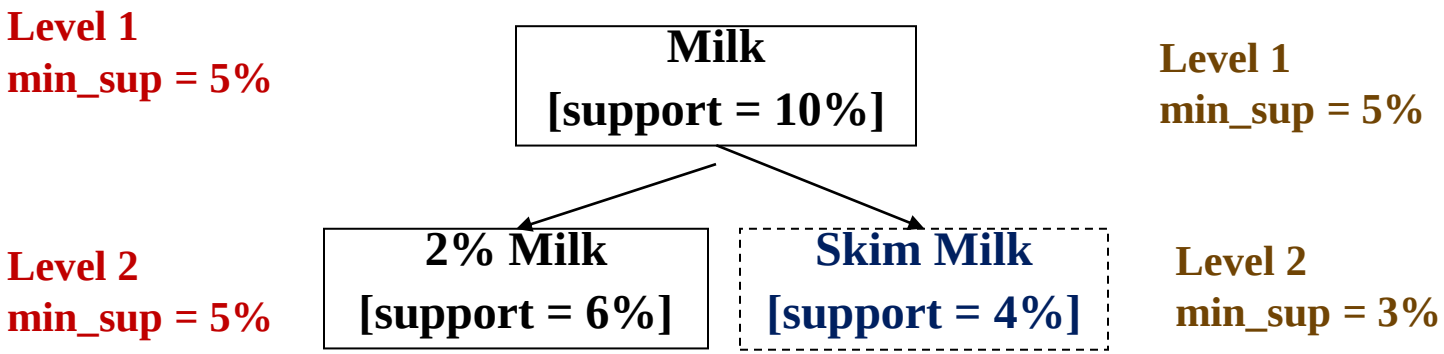
- 多层关联规则
  - 涉及不同抽象层中的概念
- 多维关联规则
  - 涉及多个维或谓词（如涉及顾客买什么和顾客年龄的规则）
- 量化关联规则
  - 涉及维之间具有隐含排序数值属性（如年龄）

挖掘多层关联规则

- 项集通常构成层次关系
  - 在多个抽象层上挖掘数据产生的关联规则称多层关联规则
- 灵活的支持度阈值
  - 对于所有层使用一致的最小支持度
  - 在较低层使用递减的最小支持度
  - 使用基于项或基于分组的最小支持度

uniform support

reduced support



多层关联规则：灵活的支持度和冗余过滤

- 灵活的支持度阈值: 一些项很有价值但却不频繁
  - 利用非一致的，基于组的支持度阈值
  - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …
- 冗余过滤: 因为项之间的父子关系，一些规则可能是冗余的
  - milk ⇒ wheat bread [support = 8%, confidence = 70%]
  - 2% milk ⇒ wheat bread [support = 2%, confidence = 72%]
  - 第一个规则是第二个规则的父规则
  - 基于该规则的父规则，如果它的支持度和置信度接近期望值，则这个规则是冗余的

挖掘多维关联规则

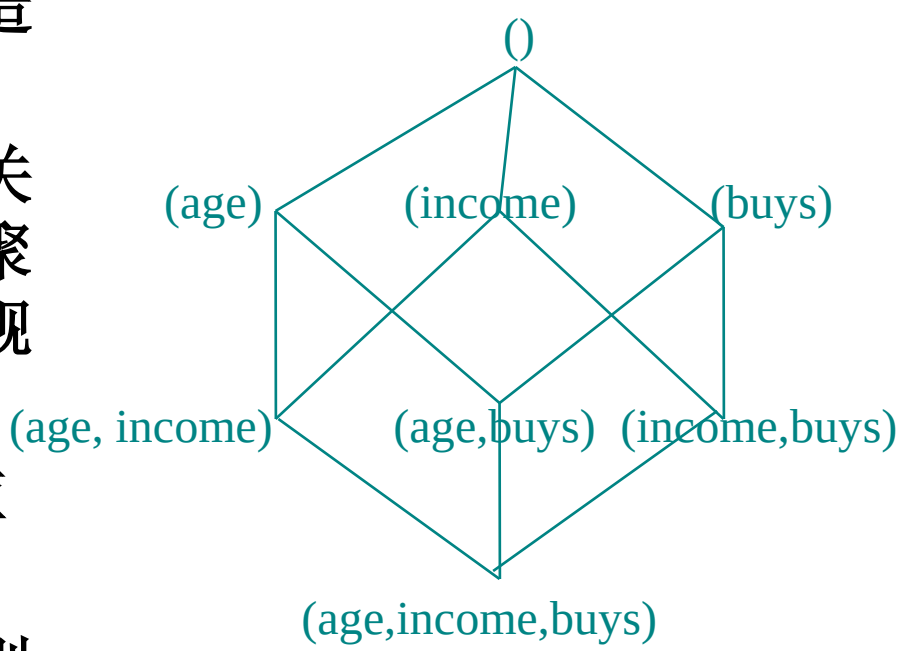
- 单维关联规则
  - buys(X, “milk”) ⇒ buys(X, “bread”)
- 多维关联规则: 大于等于两个维或谓词的关联规则
  - 维间关联规则：无重复谓词
    - age(X,”19-25”) ∧ occupation(X,“student”) ⇒ buys(X, “coke”)
  - 混合维关联规则：重复谓词
    - age(X,”19-25”) ∧ buys(X, “popcorn”) ⇒ buys(X, “coke”)
  - 在多维关联规则挖掘中，搜索频繁谓词集。*k*谓词集是包含*k*个合取谓词的集合。例如：{age, occupation, buys} 是一个3谓词集。

挖掘量化关联规则

- 属性
  - 分类属性: 具有有限个可能值，值之间无序，如：职业、品牌
  - 量化属性: 数值的，在值之间有蕴含的序，如：年龄，收入
- 挖掘量化关联规则
  - 数据立方体方法
  - 基于聚类的方法
  - 揭示异常行为的统计学方法

量化关联规则的基于数据立方体挖掘

- 在挖掘之前，用概念层次将量化属性离散化
- 数值属性的值用区间标号替代
- 变换后的多维数据可以用来构造数据立方体。
- 数据立方体非常适合挖掘多维关联规则：它们在多维空间存储聚集信息，这对于计算多维关联规则的支持度和置信度是重要的
- 可以用n维方体的单元存放对应的n谓词集的支持度计数
- 从数据立方体挖掘多维关联规则速度更快



挖掘基于聚类的量化关联规则

- 假定，有趣的频繁模式或关联规则通常在量化属性相对稠密的簇中发现。
- 自顶向下方法
  - 对于每个量化维，使用聚类算法发现该维上满足最小支持度阈值的簇。
  - 对于每个这样的簇，考察该簇与另一维的一个簇或标称属性值组合生成的二维空间，看这一组合是否满足最小支持度阈值。如果满足，进一步考察更高维。
- 自底向上方法
  - 先在高维空间聚类，然后投影并合并较少维组合上的簇。

使用统计学理论发现异常行为

- 发现揭示异常行为的量化关联规则，其中“异常”的定义建立在统计学理论的基础上。
- 比如：

Sex=female=>meanwage=7.9$/h(overall_mean_wage=9.02$/h)

（注：美国**1985**年）

主要内容

- 基本概念
- 频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

兴趣度度量

- 客观度量
  - 两个流行的度量方法
    - 支持度
    - 置信度
- 主观度量
  - 一个规则(模式)是有趣的,如果
    - 它是非预期的(令用户吃惊的)
    - 可控制的(用户可以用它来做一些事情)

支持度和置信度的缺点

- 例1:
  - 在5000个学生中
    - 3000 个打篮球
    - 3750 个吃谷类食品
    - 2000 个既打篮球又吃谷类食品
  - 打篮球⇒吃谷类食品[40%, 66.7%]
    - 是一个误导,因为吃谷类食品的学生的可能性是75% ,比66.7%要高
  - 打篮球⇒不吃谷类食品[20%, 33.3%]
    - 虽然具有较低的支持度和置信度,但比以上的规则要“有趣”得多

	打篮球	不打篮球	行和
吃谷类食品	2000	1750	3750
不吃谷类食品	1000	250	1250
列和	3000	2000	5000

支持度和置信度的缺点(续)

- 例2:
  - X 和Y: 正相关
  - X 和Z, 负相关
  - 无论是支持度还是置信度，规则 X=>Z 都有较高的值

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

- 我们需要一个依赖或相关事件的度量----提升度(Lift)

$$\text{lift}(A,B)=\frac{P(A\cup B)}{P(A)P(B)}$$

Rule	Support	Confidence
X=>Y	25%	50%
X=>Z	37.50%	75%

- P(B|A)/P(B) 也被称作规则 A => B的“提升”

其它的兴趣度量: 提升度

- 兴趣度(相关,提升)
- 考虑P(A) 和P(B)

$$\text{lift}(A,B) = \frac{P(A \cup B)}{P(A)P(B)}$$

  - 如果A和B 是独立事件, P(AB)=P(B)*P(A)
  - 如果Lift(A,B)的值小于1，则A和B是负相关；大于1，A和B是正相关的；等于1，A和B不相关。

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Itemset	Support	Interest
X,Y	25%	2
X,Z	37.50%	0.9
Y,Z	12.50%	0.57

主要内容

- 基本概念和路线图
- 有效的和可伸缩的频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

基于约束的关联挖掘

- 能否自动地找到数据库中的所有模式？----显然是不现实的！
  - 模式太多并且也不是所有模式都是用户感兴趣的！
- 数据挖掘应该是一个交互的过程
  - 用户用数据挖掘查询语言（或者图形化的用户界面）指导挖掘
- 基于约束的挖掘
  - 用户灵活性：对挖掘的内容进行限制
  - 优化: 利用约束规则优化查询过程（注: 在满足约束条件下仍然能够找到所有的结果，而不是在启发式搜索下找到一些结果）

基于约束的挖掘--约束的定义

- 知识类型约束
  - 关联规则, 相关规则，分类、聚类等
- 数据约束— 指定任务相关的数据集
  - 比如：找到本年度在芝加哥商店一起售卖的商品
- 维/层约束—指定挖掘所期望的数据维，或概念分层结构的层次
  - 比如：挖掘 region, price, brand, customer category
- 规则 (或 模式) 约束---指定要挖掘的规则模式
  - 比如：售价低的商品 (price < $10) 触发大的销售额 (sum > $200)
- 兴趣度约束
  - 比如：强规则: min_support ≥ 3%, min_confidence ≥ 60%

基于约束的频繁模式挖掘

- 对于频繁项集挖掘，规则约束分类如下
  - 反单调的：如果违反了规则c，则进一步的挖掘将被终止
  - 单调的: 如果已经满足了规则c，则后续挖掘不必再检验该规则
  - 简洁的: c必须被满足，因此可以从满足c的数据集开始挖掘
  - 可转变的: c不是单调的或反单调的，但可以被转换成单调的或反单调的，如果事务中的项可以被正确的排序
  - 不可转变的：c不是单调的或反单调的，且不可以转换。

反单调的约束

- 约束C是反单调的，如果超模式满足C，则其所有的子模式也满足。
- 换句话说，如果一个项集不满足该规则约束，它的任何超集也不可能满足该规则约束。
- Ex. 1. *sum(S.price)* ≤ *v* is anti-monotone
- Ex. 2. *range*((S.profit) ≤ 15 is anti-monotone
  - Itemset *ab* violates C
  - So does every superset of *ab*
- Ex. 3. *sum(S.Price)* ≥ *v* is not anti-monotone
- Ex. 4. 支持度计数是反单调的: Apriori算法的核心属性

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

单调的约束

- 约束C是单调的，如果模式满足C，则在后续的挖掘中不必再检查是否满足约束C
- 换句话说, 如果一个项集S满足约束，则其所有的超集也满足。
- **Ex. 1. *sum(S.Price)* ≥ *v* is monotone**
- **Ex. 2. *min(S.Price)* ≤ *v* is monotone**
- **Ex. 3. C: range(S.profit) ≥ 15**
  - **Itemset *ab* satisfies C**
  - **So does every superset of *ab***

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

简洁的约束

- 简洁的
  - 给定 A_1, 项的集合满足一个简洁约束C，则任何满足C的集合S基于A_1 , 即，S 包含一个属于A_1的子集
  - 基本思想: 不必查看事务数据库，一个项集S是否满足约束C可以由其选择的项来决定。
  - $min(S.Price) \leq v$ is succinct
  - $sum(S.Price) \geq v$ is not succinct
- 优化: 如果C是简洁的，则C 是计数前可剪枝的

可转变的约束

- 通过把项以特定次序排序把难处理的约束转变成反单调的或单调的约束。
- 例如：约束C: avg(S.profit) ≥ 25
  - Order items in value-descending order
    - <a, f, g, d, b, h, c, e>
  - If an itemset *afb* violates C
    - So does *afbh, afb**
    - It becomes anti-monotone!

TDB (min_sup=2)

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

不可转变的约束

- 不是每种约束都是可转变的
- 不可转变的约束
  - 如：“sum(S) θ v”其中 θ∈{⩽，⩾}并且S中的每个元素可以是任意实数。
- 尽管有一些难处理的约束是不可转变的，但是大部分使用SQL内部聚集的简单SQL表达式都属于前四类之一，对于它们可以使用有效的约束挖掘方法。

主要内容

- 基本概念和路线图
- 有效的和可伸缩的频繁项集挖掘方法
- 挖掘各种类型的关联规则
- 由关联挖掘到相关分析
- 基于约束的关联挖掘
- 小结

小结

- 关联规则挖掘
  - Apriori算法
  - 频繁模式增长
  - 多层关联规则
  - 基于约束的规则挖掘
- 相关分析
  - 相关规则
  - 提升度

参考文献

- Agrawal R, Imielinski T, and Swami A. *Mining association rules between sets of items in large databases*. SIGMOD, 207-216, 1993.
- Agrawal R, and Srikant R. *Fast algorithms for mining association rules in large databases*. VLDB, 478-499, 1994.
- Han J W, Pei J, Yin Y W. *Mining frequent patterns without candidate generation*. SIGMOD, 1-12, 2000.
- Han J W, Pei J, Yin Y W, and Mao R Y. *Mining frequent patterns without candidate generation: a frequent-pattern tree approach*. Data Mining and Knowledge Discovery. 8, 53-87, 2004

习题一

- 数据库有4个事务。设min_sup=60%,min_conf=80%

TID	Itmes_bought
T100	{K,A,D,B}
T200	{D,A,C,E,B}
T300	{C,A,B,E}
T400	{B,A,D}

- 分别使用Apriori和FP-增长算法找出频繁项集。
- 列出所有的强关联规则（带支持度s和置信度c），它们与下面的元规则匹配，其中，X是代表顾客的变量，itmei是表示项的变量（例如：A、B等）

$$\forall x \in \text{transaction}, buys(X, itme_1) \wedge buys(X, itme_2) \Rightarrow buys(X, itme_3) \quad [s, c]$$

习题二

- 下面的相依表总汇了超级市场的事务数据。其中，**hot dogs**表示包含热狗的事务，**non-hotdogs**表示不包含热狗的事务，**hamburgers**表示包含汉堡的事务，**non-hamburgers**表示不包含汉堡包的事务。

	hotdogs	Non-hotdogs	行汇总
hamburgers	2000	500	2500
Non-hamburgers	1000	1500	2500
列汇总	3000	2000	5000

- 假定发现关联规则”**hot dog => hamburgers**”。给定最小支持度阈值**25%**，最小置信度阈值**50%**，该关联规则是强的么？
- 根据给定的数据，买**hot dog**独立于买**hamburgers**么？如果不是，二者存在何种“相关”关系？

思考题

- 商店里每种商品的价格都是非负的。对于以下每种情况，识别他们提供的约束类型，并简略讨论如何有效地挖掘这种关联规则。
  - 至少包含一件任天堂游戏。
  - 包含一些商品，它们的价格和小于150美元。
  - 包含一件免费商品，并且其他商品的价格和至少是200美元。
  - 所有商品的平均价格在100美元-500美元之间。

END