数据仓库与数据挖掘

Data warehouse and data mining

丁钰

yuding@live.com

南京农业大学人工智能学院

第六章:挖掘频繁模式、关联和相关性：基本概念和方法

第6章：挖掘频繁模式、关联和相关性：基本概念和方法

- 基本概念
- 高效的模式挖掘方法
- 模式评估
- 摘要

模式发现：基本概念

- 什么是模式发现？ 为什么它很重要？
- 基本概念：频繁模式和关联规则
- 压缩表示：闭模式和极大模式

什么是模式发现？

- 什么是模式？
  - 模式:一组项目、子序列或子结构，在数据集中经常一起出现（或强烈相关）。
  - 模式代表了数据集的内在和重要属性
- 模式发现:从海量数据集中发现模式
- 例子：
  - 哪些产品经常是一起购买的？
  - 购买iPad后的后续购买行为有哪些？
  - 在这个语料库中，哪些词的序列可能构成短语？

模式发现：为什么它很重要？

- 在数据集中寻找内在的规律性
- 是许多基本数据挖掘任务的基础
  - 关联性、相关性和因果关系分析
  - 挖掘顺序性、结构性（如子图）模式
  - 时空、多媒体、时间序列和流数据的模式分析
  - 分类：基于判别模式的分析
  - 聚类分析：基于模式的子空间聚类
- 广泛的应用
  - 购物篮分析、交叉营销、目录设计、销售活动分析、网络日志分析、生物序列分析

基本概念：K项集（K-itemsets）和支持度

- 项集：一个或多个项的集合
- k-项集： $X = \{x_1, ..., x_k\}$
  - 例：{Beer, Nuts, Diaper}是一个3个项的集合
- X的*（绝对）支持度*，sup{X}。频率或一个项集X的出现次数
  - 例如：sup{Beer}=3
  - 例：sup{Diaper}=4
  - 例如：sup{Beer, Diaper}=3
  - 例如：sup{Beer, Eggs}=1

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

- *(相对)支持度*，s{X}。 含有X的交易比例（即一个交易含有X的概率）
  - 例如，s{啤酒}=3/5=60%。
  - 例如：s{尿布}=4/5=80%。
  - 例如：s{啤酒，鸡蛋}=1/5=20%。

基本概念：频繁项集（模式）

- 如果一个项集（或模式）X的支持度不低于一个*Minsup*阈值σ，那么它就是*频繁的。*
- 让σ=*50%*（σ：*Minsup*阈值）。
  对于给定的5个交易的数据集
  - 所有经常出现的1-项集。
    - 啤酒：3/5（60%）；坚果：3/5（60%）。
    - 纸尿裤：4/5 (80%); 鸡蛋: 3/5 (60%)
  - 所有经常出现的2-项集。
    - {啤酒，尿布}。3/5 (60%)
  - 所有经常出现的3个项目组合？
    - 无

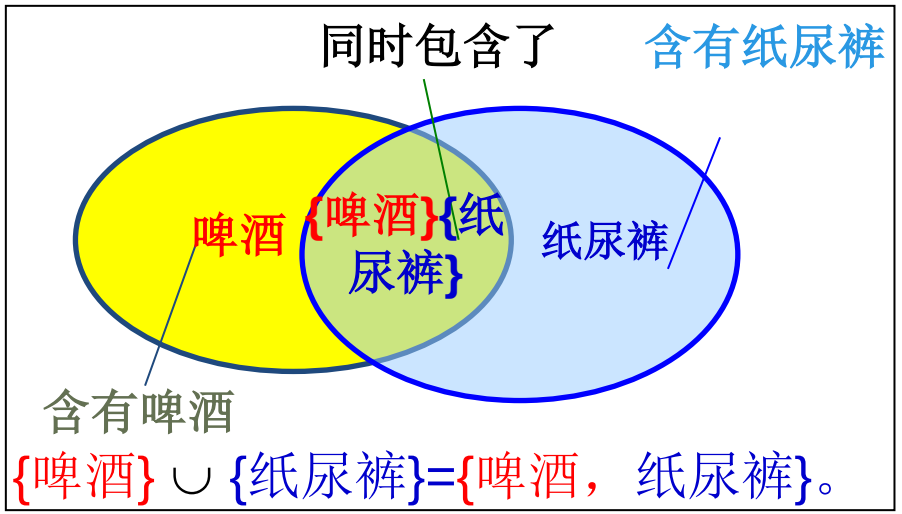
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- 为什么这些项集（如左图所示）会形成任何*K*的频繁*K*项集（模式）的完整集合？
- **观察：** 我们可能需要一种有效的方法来挖掘完整的频繁模式集

从频繁项集到关联规则

- 与项集相比，规则可以更有说服力
  - 例：*尿布* → *啤酒*
    - *购买尿布可能会导致购买啤酒*
- 这个规则有多强？ (support, confidence)
  - 衡量关联规则： $X \rightarrow Y$ (s, c)
    - *X*和*Y*都是项集
  - Support， *S*:一项交易包含X ∪ Y的概率
    - 例如：s{尿布，啤酒}=3/5=0.6（即60%）。
  - confidence，*c*:包含X的交易也包含*Y*的*条件概率*
    - 计算方法：$C = \sup(X \cup Y) / \sup(X)$
    - 例：$c = \sup\{Diaper, Beer\}/\sup\{Diaper\} = ¾ = 0.75$

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



注：X ∪ Y：两个项目集的联合。

- 该集合同时包含X和Y

挖掘频繁项集和关联规则

- **关联规则挖掘**
  - 给出两个阈值：*Minsup, minconf*
  - 找到所有的规则，*X → Y* (s, c)
    - 这样，s≥*minsup*，*c*≥*minconf*

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

- 让*minsup = 50%*
  - 频率。1-项集。啤酒: 3, 坚果: 3, 尿布:4, 鸡蛋: 3
  - 频率：2项集。 {啤酒，尿布}：3

- 让*minconf = 50%*
  - *啤酒 → 尿布* (60%, 100%)
  - *尿布 → 啤酒*（60%，75%）。

(问：这些都是规则吗？)

- **观察到的情况:**
  - 挖掘关联规则和挖掘频繁模式是非常接近的问题
  - 挖掘大型数据集需要可扩展的方法

挑战：频繁模式太多!

- 一个长的模式包含一子列的子模式
- 下面的TDB_1包含多少个频繁项集？
  - TDB_1: T1: {a1, …, a50}; T2: {a1, …, a100}
  - 假设（绝对）*Minsup*=1
  - 让我们试一试

1-itemsets: $\{a_1\}$:2, $\{a_2\}$:2, ..., $\{a_{50}\}$:2, $\{a_{51}\}$:1, ..., $\{a_{100}\}$:1,

2-itemsets：$\{a_1, a_2\}$:2, ..., $\{a_1, a_{50}\}$:2, $\{a_1, a_{51}\}$:1 ..., ..., $\{a_{99}, a_{100}\}$:1,

..., ..., ..., ...

99-itemsets:$\{a_1, a_2, ..., a_{99}\}$:1，…，$\{a_2, a_3, ..., a_{100}\}$：1

100-itemset:$\{a_1, a_2, ..., a_{100}\}$:1

一个巨大的集合，任何人都无法计算或储存!

- 频繁项集的总数：

$$\binom{100}{1} + \binom{100}{2} + \binom{100}{3} + \cdots + \binom{100}{100} = 2^{100} - 1$$

以压缩的形式表达模式：闭合模式

- 如何处理这样的挑战？
- 解决方案1：**闭模式**。 如果一个模式（项集）X是*频繁的*，并且*不存在与X支持度相同的超级模式*Y ⊃X，那么这个模式就是封闭的。
  - 让交易数据库TDB:$_1$ T_1: {a_1, ..., a_{50}}; T_2: {a_1, ..., a_{100}}
  - 假设*minsup*=1，TDB_1包含多少个封闭模式？
    - 2：P_1："{a_1，...，a_{50}}。2"; P_2: "{a_1, ..., a_{100}}:1"
- 闭合模式是一种对频繁模式的无损压缩
  - 减少了模式的数量，但并没有失去支持信息!
  - 你仍将能够说"{a_2, ..., a_{40}}:2"，"{a_5, a_{51}}:1"

以压缩的形式表达模式：最大模式

- 解决方案2：最大**模式**。 如果一个模式X是频繁的，并且不存在频繁的超级模式 Y⊃X，那么它就是一个最大模式。
- 与闭模式的区别？
  - 不关心一个最大模式的子模式的实际支持度
  - 让交易数据库TDB:$_1$ T_1: {a_1, ..., a_{50}}; T_2: {a_1, ..., a_{100}}
  - 假设*minsup*=1，TDB_1包含多少个最大模式？
    - 1：P："{a_1，...，a_{100}}。1"
- Max-pattern是一种有损失的压缩!
  - 我们只知道{a_1，...，a_{40}}是频繁的
  - 但我们不知道{a_1, ..., a_{40}}, ..., 的真正支持度了!
- 因此，在许多应用中，挖掘闭模式比挖掘最大模式更可取。

第6章：挖掘频繁模式、关联和相关性：基本概念和方法

- 基本概念
- 高效的模式挖掘方法
- 模式评估
- 摘要

高效的模式挖掘方法

- 频繁模式的向下闭合特性
- Apriori算法
- Apriori的扩展或改进
- 通过探索垂直数据格式挖掘频繁模式
- FPGrowth: 频繁模式增长的算法
- 挖掘封闭模式

频繁模式的向下闭合特性

- 观察：从TDB_1：T_1：{a_1，…，a_{50}}；T_2：{a_1，…，a_{100}}。
  - 我们得到一个频繁项集： {a_1, …, a_{50}}
  - 而且，它的子集都是频繁的。{a_1}, {a_2}, …, {a_{50}}, {a_1, a_2}, …, {a_1, …, a_{49}}, …。
  - 频繁的模式之间一定存在着一些隐藏的关系!
- 频繁模式的向下封闭（也称为 "Apriori"）属性
  - 如果**{啤酒、尿布、坚果}**是频繁的，那么**{啤酒、尿布}**也是。
  - 每笔包含{啤酒、尿布、坚果}的交易也包含{啤酒、尿布}。
  - Apriori：频繁项集的任何子集都必须是频繁的。
- 高效的挖掘方法
  - 如果一个项目集S的任何一个子集是不频繁的，那么S就没有机会是频繁的，我们为什么还要考虑S呢！？

一把修剪用的尖刀!

Apriori修剪和可扩展的挖掘方法

- Apriori修剪原则：如果有任何子项集是不频繁的，它的超集甚至不应该被生成(Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- 可扩展的挖掘方法： 三种主要方法
  - 层次分明、基于连接的方法。 Apriori (Agrawal & Srikant@VLDB'94)
  - 垂直数据格式方法。Eclat (Zaki, Parthasarathy, Ogihara, Li @KDD'97)
  - 频繁的模式投射和增长。FPgrowth (Han, Pei, Yin @SIGMOD'00)

Apriori：一个候选项集生成和测试的方法

- Apriori的概要（按级别划分，候选项集的生成和测试）。
  - 最初，扫描一次DB，以获得频繁的1-项集
  - 重复进行
    - 从长度为k的频繁项集生成长度为(k+1)的候选项集
    - 根据数据库测试候选者，找到频繁的(k+1)项集
    - 设置k := k +1
  - 直到不能生成频繁集或候选集
  - 返回所有衍生的频繁项集

Apriori算法(伪代码)

C_k: Candidate itemset of size k

F_k : Frequent itemset of size k

K := 1;
F_k := {frequent items}; // frequent 1-itemset
While (F_k != ∅) **do {** // when F_k is non-empty
    C_{k+1} := candidates generated from F_k; // candidate generation
    Derive F_{k+1} by counting candidates in C_{k+1} with respect to *TDB* at minsup;
    k := k + 1
    }
return $\cup_k F_k$ // return F_k generated at each level

Apriori算法--一个例子

Database TDB

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

minsup = 2

C_1

1st scan

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

F_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

F_2

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

C_3

Itemset
{B, C, E}

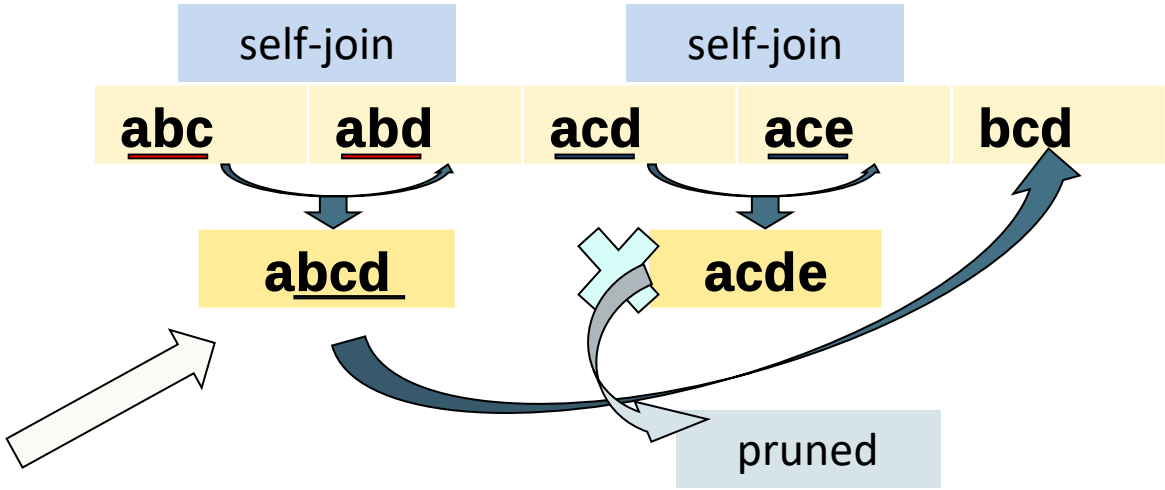
3rd scan

F_3

Itemset	sup
{B, C, E}	2

Apriori：实现技巧

- 如何产生候选？
  - 第1步：自我连接F_k
  - 第2步：修剪
- 候选生成的例子
  - F_3 = {*abc, abd, acd, ace, bcd*}
  - 自我连接。F_3*F_3
    - *abcd*来自*abc*和*abd*
    - Acde来自*acd*和*ace*
  - 修剪。
    - *acde*被删除，因为*ade*不在F_3
  - C_4 = {*abcd*}



候选者的产生：一个SQL的实现

- Suppose the items in F_{k-1} are listed in an order
- Step 1: self-joining F_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

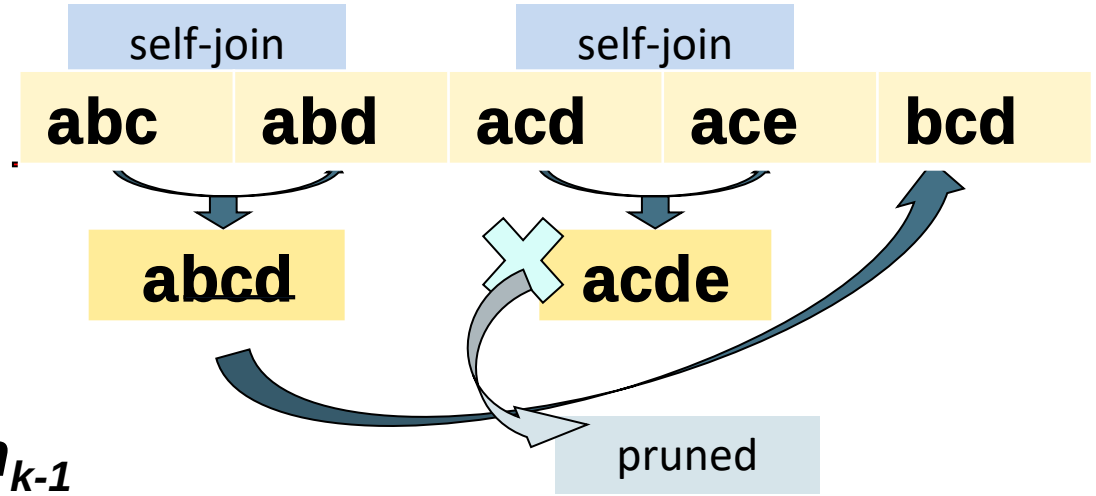
  from F_{k-1} as p, F_{k-1} as q

  where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  for all *itemsets c in* C_k do

  for all *(k-1)-subsets s of c* do

  if (*s is not in* F_{k-1}) **then delete** *c* **from** C_k



Apriori：改进和替代方案

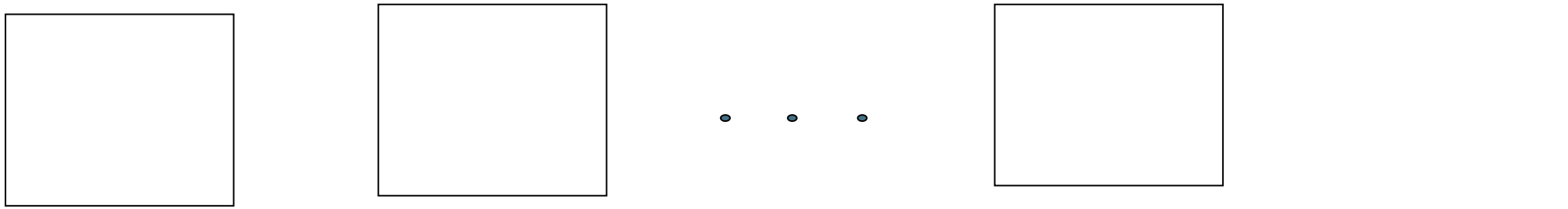
- 减少交易数据库的扫描次数
  - 分区（例如，Savasere et al，1995）。
  - 动态项目集计数（Brin, et al., 1997）。
- 缩减候选的数量
  - Hashing （如DHP：Park, et al., 1995）。
  - 通过支持下限修剪（例如，Bayardo 1998）。
  - 抽样（例如，Toivonen，1996）。
- 探索特殊的数据结构
  - 树状投影（Agarwal, et al，2001）。
  - H-miner (Pei, et al., 2001)
  - 超立方体分解（例如，LCM：Uno, et al.）

将在随后的幻灯片中讨论

将在随后的幻灯片中讨论

分区：只扫描两次数据库

- 定理:*任何可能在TDB中频繁出现的项目集，在TDB的至少一个分区中是频繁的。*



- 方法:扫描DB两次（A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)
  - 扫描1：对数据库进行分区，使每个分区都能容纳在主内存中（为什么？）
    - 在这个分区中挖掘局部频繁模式
  - 扫描2：巩固全局频繁的模式
    - 寻找全局频繁项集候选者（那些在至少一个分区中频繁出现的项集）。
    - 通过再一次扫描TDB_i，找到这些候选者的真实频率

直接哈希和修剪（DHP）。

- DHP（ Direct Hashing and Pruning ）:(J. Park, M. Chen, and P. Yu, SIGMOD'95)
- 哈希：不同的项集可能有相同的哈希值：v = *hash*(itemset)
- 1st扫描。当计算1-项集时，哈希2-项集来计算桶的数量。
- 观察: 如果一个*k项集*的对应散列桶数低于*Minsup*阈值，则该*k*项集不可能是频繁的。
- 例:在TDB的第1st次扫描中，计数1-项集，并且
  - 将交易中的2项集哈希到其桶中
    - {ab, ad, ce}.
    - {bd，be，de}
    - ...
  - 在第一次扫描结束时。
    - *如果minsup=80，移除ab、ad、ce，因为count{ab、ad、ce}*<80。

项目集	计数
{*ab, ad, ce*}.	35
{*bd，be，de*}	298
.....	...
{*yz, qs, wt*}	58

哈希表

On the fly C_1 count L_1

C_1	count	L_1
{A}	2	{A}
{B}	3	{B}
{C}	3	{C}
{D}	1	{E}
{E}	3	

minimum support, s = 2

Making a hash table

100 {A C}, {A D}, {C D}
200 {B C}, {B E}, {C E}
300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
400 {B E}

h{{x y}) = ((order of x)*10 + (order of y)) mod 7;

0	1	2	3	4	5	6
{C E} {C E} {A D}	{A E}	{B C} {B C}		{B E} {B E} {B E}	{A B}	{A C} {C D} {A C}
3	1	2	0	3	1	3

Hash table H_2

Hash address

The number of items hashed to bucket 2

Generating C_2

in a bucket with the itemset

$L_1 \times L_1$

		C_2
{A B}	1	
{A C}	3	{A C}
{A E}	1	{B C}
{B C}	2	{B E}
{B E}	3	{C E}
{C E}	3	

探索垂直数据格式(Exploring Vertical Data Format):ECLAT

- ECLAT（等价类转换）：一种使用集合相交的深度优先搜索算法[Zaki等人，@KDD'97]
- Tid-List:含有一个项目集的交易-ID的列表
- 垂直格式：T(e)={T_{10}, T_{20}, T_{30}}；T(a)={T_{10}, T_{20}}；T(ae)={T_{10}, T_{20}}。
- Tid-List的属性
  - t(X) = t(Y)。X和Y总是一起发生（例如，t(ac}=t(d}）。
  - t(X) ⊂ t(Y)：有X的交易总是有Y（例如，t(ac) ⊂ t(ce)）。
- 根据垂直交叉推导出频繁模式
- 使用diffset来加速挖掘
  - 只记下不同种类的数据
  - t(e) = {T_{10}, T_{20}, T_{30}}, t(ce) = {T_{10}, T_{30}} → Diffset (ce, e) = {T_{20}}

一个水平数据格式的交易数据库

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

垂直数据格式的交易数据库

项目	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

为什么要通过模式增长来挖掘频繁模式？

- Apriori：一种*广度优先的搜索*挖掘算法
  - 首先找到完整的频繁K项集的集合
  - 然后推导出频繁的(k+1)-项集候选者
  - 再次扫描数据库，找到真正频繁的(k+1)项集
- 采用不同挖掘方法的动机
  - 我们能否开发一种*深度优先搜索的*挖掘算法？
  - 对于一个频繁的项集ρ，后续的搜索能否只限于包含ρ的那些交易？
- 这种想法产生了frequent pattern growth。
  - FPGrowth (J. Han, J. Pei, Y. Yin, "Mining Frequent Patterns without Candidate Generation, " SIGMOD 2000)

例子：从交易数据库中构建FP-树

TID	Items in the Transaction	Ordered, frequent itemlist
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, w}	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

在插入频繁1st的项列表。"F、C、A、M、P"

1. 扫描一次数据库，找到单项的频繁模式。

   min_support = 3

   F：4，A：3，C：4，B：3，M：3，P：3

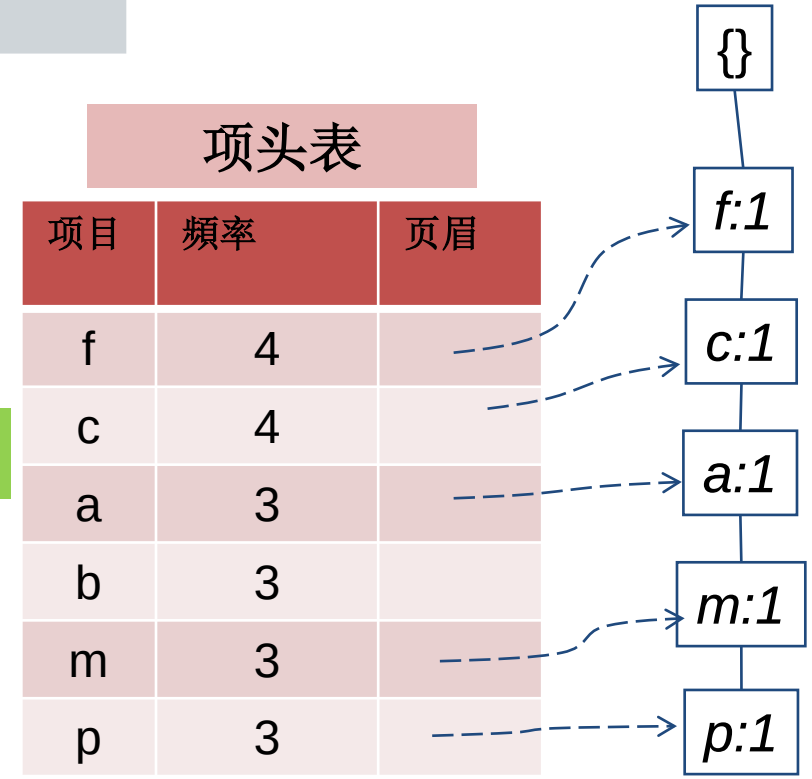
2. 按频率降序排列频繁出现的项目，f-list
3. 再次扫描数据库，构建FP树

   F-list = f-c-a-b-m-p

   - 每个交易的频繁项表被作为一个分支插入，共享的子分支被合并，计数被累积。

项头表

项目	频率	页眉
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



例子：从交易数据库中构建FP-树

TID	Items in the Transaction	Ordered, frequent itemlist
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, w}	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

在插入2nd个频繁的项列表 "F、C、A、B、M"

1. 扫描一次数据库，找到单项的频繁模式。

让min_support = 3

F：4，A：3，C：4，B：3，M：3，P：3

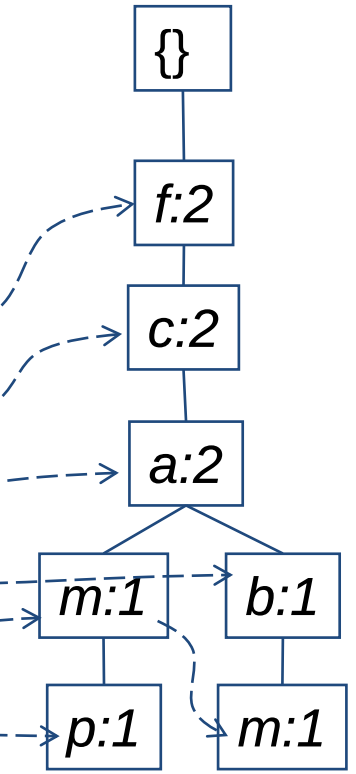
2. 按频率降序排列频繁出现的项目，f-list
3. 再次扫描数据库，构建FP树

F-list = f-c-a-b-m-p

- 每个交易的频繁项目表被作为一个分支插入，共享的子分支被合并，计数被累积。

项头表

项目	频率	页眉
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



例子。从交易数据库中构建FP-树

TID	Items in the Transaction	Ordered, frequent itemlist
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, w}	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

1. 扫描一次数据库，找到单项的频繁模式。

   让min_support = 3

   F：4，A：3，C：4，B：3，M：3，P：3

2. 按频率降序排列频繁出现的项目，f-list

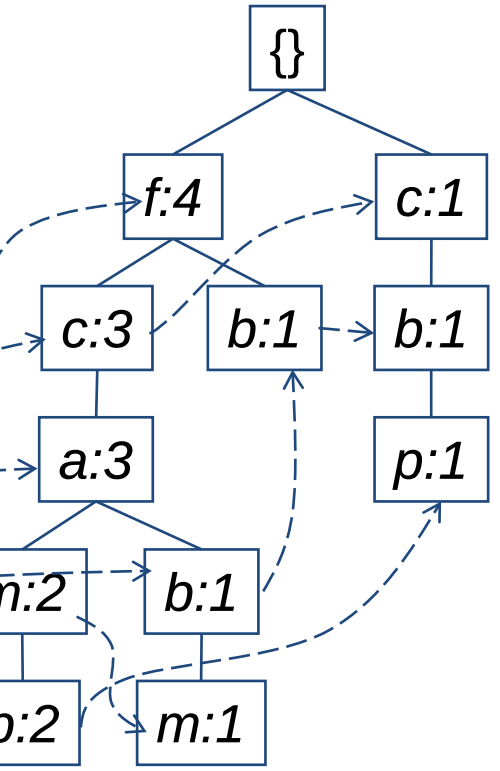
   F-list = f-c-a-b-m-p

3. 再次扫描数据库，构建FP树
   - 每个交易的频繁项表被作为一个分支插入，共享的子分支被合并，计数被累积。

项头表

项目	频率	页眉
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

在插入所有频繁的项表后

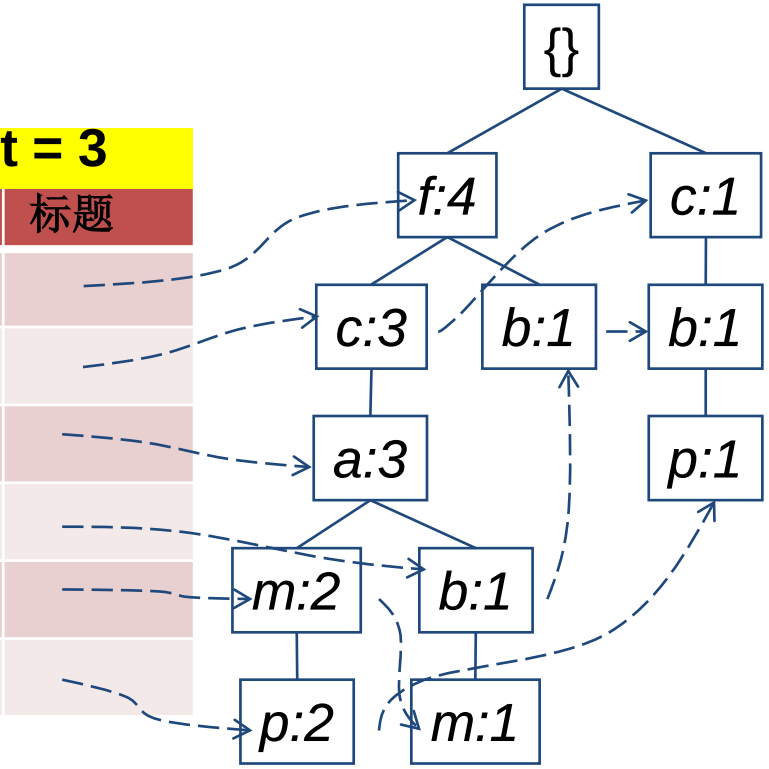


挖掘FP-树：基于模式和数据

- 模式挖掘可以根据当前模式进行划分
  - 包含*p*的模式：*p*的条件模式基：*FCAM:2, CB:1*
    - *p*的条件模式基（即*p*存在的条件下的数据库）。
      - 项目*p*的*前缀路径*
  - 有m但没有p的模式：m的条件数据库：*FCA：2，FCAB：1*
  - ....

min_support = 3

项目	频率	标题
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



Conditional database of each pattern

Item	*Conditional database*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

递归地挖掘每个条件模式基

min_support = 3

- 对于每个条件模式基
  - 挖掘单项模式
  - 构建其条件FP-树并进行挖掘

Conditional Data Bases

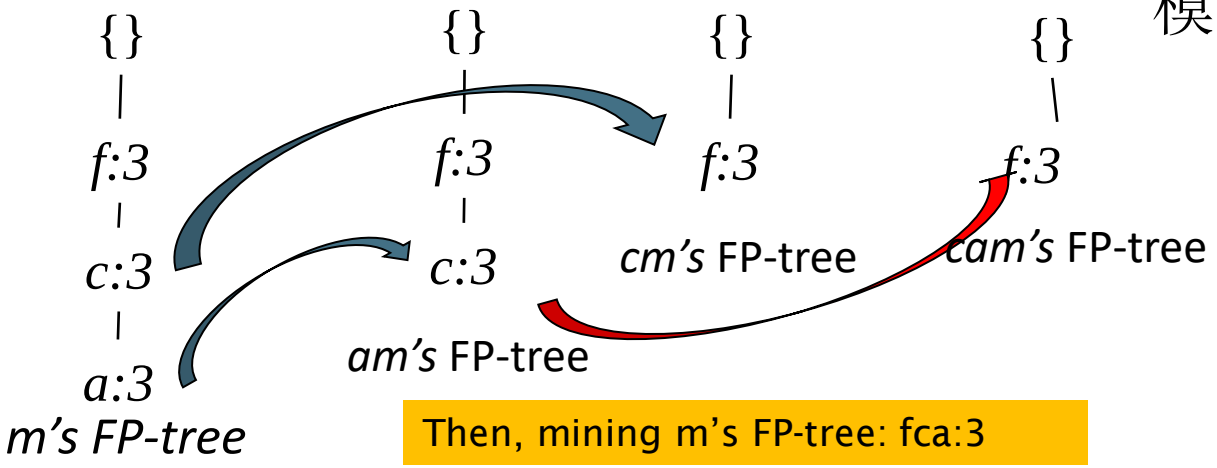
item	*cond. data base*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

p's conditional DB: ***fcam:2, cb:1 → c: 3***

m's conditional DB: ***fca:2, fcab:1 → fca: 3***

b's conditional DB: ***fca:1, f:1, c:1 → ϕ***

实际上，对于单分支的FP-树，所有的频繁模式都可以一次生成。

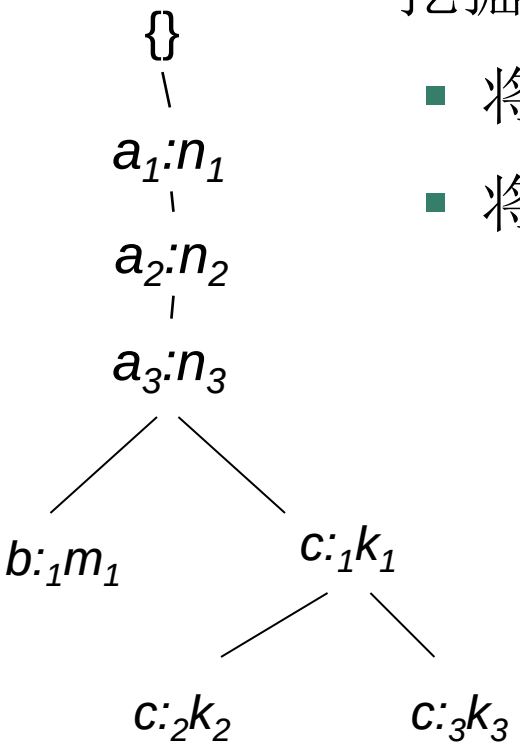


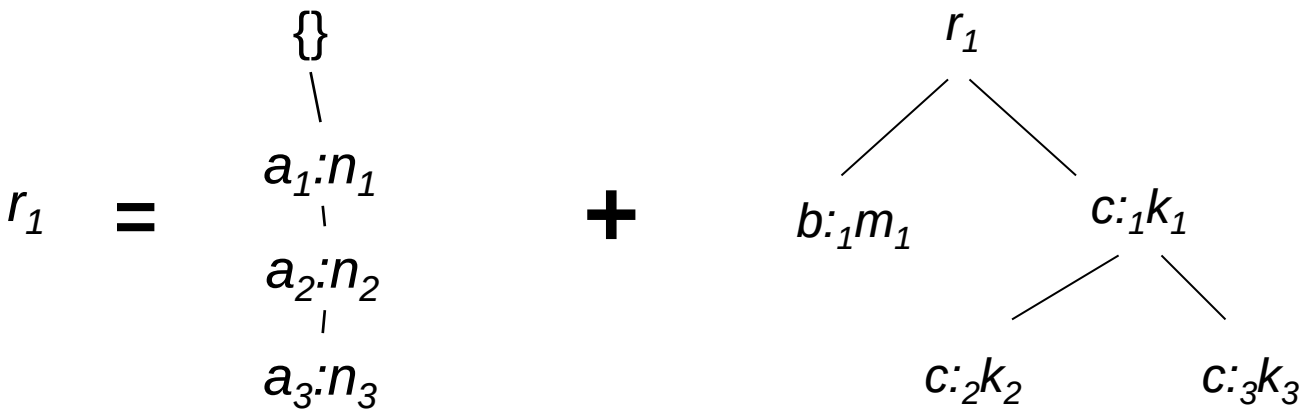
Then, mining m's FP-tree: fca:3

m: 3
fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3
fcam: 3

一个特例：FP-树中的单前缀路径

- 假设一个（有条件的）FP树T有一个共享的单一前缀路径P
- 挖掘可以分解为两个部分
  - 将单一前缀路径缩减为一个节点
  - 将两部分的挖掘结果串联起来





FPGrowth:通过模式增长挖掘频繁模式

- 频繁模式增长（FPGrowth）方法的本质
  - 寻找频繁出现的单项，并根据每个这样的单项模式来划分数据库
  - 通过对每个分区*的数据库*（也称为模式的*条件数据库*）进行上述操作，递归地增长频繁模式
  - 为了便于高效处理，可以构建一个高效的数据结构，即FP-树。
- 挖掘
  - 递归地构建和挖掘（有条件的）FP-树
  - 直到产生的FP-树是空的，或者直到它只包含一条路径--单一路径将产生其子路径的所有组合，其中每一条都是一个频繁的模式。

Trans	Items
1	b,d,f,g,l
2	f,g,h,l,m,n
3	b,f,h,k,m
4	a,f,h,j,m
5	d,f,g,j,m

1.设relative min_sup=0.6,使用Apriori算法寻找频繁模式。

2.设relative min_sup=0.4,使用fp-growth算法寻找频繁模式

第6章：挖掘频繁模式、关联和相关性：基本概念和方法

- 基本概念
- 高效的模式挖掘方法
- 模式评估
- 摘要

模式评估

- 支持-置信框架的局限性
- 有趣的度量：提升和相关χ^2
- 无变量度量
- 趣味性度量的比较

如何判断一条规则/模式是否有趣？

- 模式挖掘将产生一大套模式/规则
  - 并非所有生成的模式/规则都是有趣的
- 趣味性的测量。客观的与主观的
  - 客观的趣味性措施
    - 支持、置信、相关、...
  - 主观的趣味性衡量。
      - 不同的用户对趣味性的判断可能不同
    - 让用户指定
      - 基于查询： 与用户的特定请求有关
    - 根据自己的知识基础进行判断
      - 出乎意料，新鲜感，及时性

支持-信任框架的局限性

- 在关联规则中，*s*和*c*是否有趣。"A ⇒ B" [*s*, *c*]?

要小心!

- 例子。 假设一个学校可能有以下关于可能打篮球和/或吃麦片的学生人数的统计。

	打篮球	不打篮球	和(行)
吃麦片	400	350	750
不吃麦片	200	50	250
sum(col.)	600	400	1000

2-way contingency table

- 关联规则挖掘可能会产生以下结果。
  - *打篮球* ⇒ *吃麦片* [40%, 66.7%] (更高的S & C)
- 但这个强关联规则是误导性的。吃麦片的学生总体比例是75%>66.7%，这是一个更有说服力的规则。
  - ¬ *打篮球* ⇒ *吃麦片* [35%, 87.5%] (高S & C)

趣味性的度量：提升度（lift）

- 依赖/相关事件的衡量标准：**lift**

$$lift(B,C)=\frac{c(B\rightarrow C)}{s(C)}=\frac{s(B\cup C)}{s(B)\times s(C)}$$

*提升*比S & C更有说服力

	B	¬B	$\sum_{row}$
C	400	350	750
¬C	200	50	250
$\sum_{col.}$	600	400	1000

- Lift(B, C)可以说明B和C是如何关联的
  - Lift(B, C) = 1: B和C是独立的
  - >1：正相关
  - < 1: 负相关
- 对于我们的例子。

$$lift(B,C)=\frac{400/1000}{600/1000\times 750/1000}=0.89$$

$$lift(B,\neg C)=\frac{200/1000}{600/1000\times 250/1000}=1.33$$

- 因此，B和C是负相关的，因为lift（B，C）<1。
  - B和¬C是正相关的，因为lift(B, ¬C) > 1。

趣味性度量： χ^2

- 检验相关事件的另一个措施： χ^2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- 对于右边的表格。

	B	¬B	$\sum_{row}$
C	400 (450)	350 (300)	750
¬C	200 (150)	50 (100)	250
$\sum_{col}$	600	400	1000

预期值

观察值

$$\chi^2 = \frac{(400-450)^2}{450} + \frac{(350-300)^2}{300} + \frac{(200-150)^2}{150} + \frac{(50-100)^2}{100} = 55.56$$

- 通过查阅χ^2分布的临界值表，可以得出结论：B和C独立的可能性非常低（<0.01）。
- χ^2检验显示B和C是负相关的，因为预期值是450，但观察值只有400。
- 因此，χ^2也比支持-置信框架更能说明问题

lift和 χ^2：它们总是好的度量吗？

- 零事务： 既不包含B也不包含C的交易
- 让我们检查一下新的数据集D
  - BC(100)比B¬C(1000)和¬BC(1000)稀少得多，但有许多¬B¬C(100000)。
  - 不太可能B和C一起发生!
- 但是，Lift(B, C) = 8.44 >> 1（Lift显示B和C是强正相关的！）。
- χ^2 = 670:观察值（BC）>>预期值（11.85）
- *太多的无效交易可能会"坏了一锅汤"!*

	B	¬B	$\sum_{row}$
C	100	1000	1100
¬C	1000	100000	101000
$\sum_{col.}$	1100	101000	102100

空交易

加入预期值的应急表

	B	¬B	$\sum_{row}$
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\sum_{col.}$	1100	101000	102100

趣味性度量和无差异性

- *零不变度量*。不随零事务的数量而变化的值
- 一些有趣的度量。 有些是零不变的

Measure	Definition	Range	Null-Invariant?
$\chi^2(A,B)$	$\sum_{i,j} \frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0, \infty]$	No
$Lift(A,B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$Allconf(A,B)$	$\frac{s(A \cup B)}{max\{s(A), s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A,B)$	$\frac{s(A \cup B)}{s(A)+s(B)-s(A \cup B)}$	$[0, 1]$	Yes
$Cosine(A,B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)})$	$[0, 1]$	Yes
$MaxConf(A,B)$	$max\{\frac{s(A \cup B)}{s(A)}, \frac{s(A \cup B)}{s(B)}\}$	$[0, 1]$	Yes

X^2 *和lift不是零不变的*

Jaccard、consine、AllConf、MaxConf和Kulczynski都是零不变的度量。

模式评估比较（几种度量）

- 全置信度： $all\_conf(A,B)=\min\{P(A|B),P(B|A)\}$
- 最大置信度：$max\_conf(A,B)=\max\{P(A|B),P(B|A)\}$
- **Kulczynski（Kulc）度量：** $Kulc(A,B)=\frac{1}{2}(P(A|B)+P(B|A))$
- 余弦度量：$\cos ine(A,B)=\sqrt{P(A|B)\times P(B|A)}$

零不变性：一个重要的属性

- 为什么零不变性对分析大量交易数据至关重要？
  - 许多交易可能既不含牛奶，也不含咖啡!

牛奶与咖啡相依表

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

- Lift和χ^2不是零不变的：不好评估包含过多或过少零事务的数据!
- 许多度量都不是零不变的!

Null-transactions w.r.t. m and c

Data set	mc	$\neg mc$	$m\neg c$	$m\neg c$	χ^2	$Lift$
D_1	10,000	1,000	1,000	100,000	90557	9.26
D_2	10,000	1,000	1,000	100	0	1
D_3	100	1,000	1,000	100,000	670	8.44
D_4	1,000	1,000	1,000	100,000	24740	25.75
D_5	1,000	100	10,000	100,000	8173	9.18
D_6	1,000	10	100,000	100,000	965	1.97

零不变性度量的比较

- 并非所有的零不变性度量都是平等的
- 哪一个更好？
  - D_4-D_6的零不变性度量是有区别的

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

所有5个都是零不变的

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	$AllConf$	Jaccard	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

微妙：他们在这些数据集上意见不一致

Imbalance Ratio with Kulczynski Measure

- IR（Imbalance Ratio）：衡量两个项目集A和B在规则影响中的不平衡性。

$$IR(A,B) = \frac{|s(A)-s(B)|}{s(A)+s(B)-s(A \cup B)}$$

- Kulczynski和Imbalance Ratio（IR）一起对所有三个数据集D_4到D呈现出清晰的画面[6]
  - D_4 不中立和平衡； D_5 不中立但不平衡
  - D_6 中立，但非常不平衡

Data set	mc	$\neg mc$	$m\neg c$	$\neg m\neg c$	Jaccard	$Cosine$	$Kulc$	IR
D_1	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
D_2	10,000	1,000	1,000	100	0.83	0.91	0.91	0
D_3	100	1,000	1,000	100,000	0.05	0.09	0.09	0
D_4	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
D_5	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
D_6	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

选择什么度量进行有效的模式评估？

- 在许多大型数据集中，空值情况占主导地位
  - 大多数篮子里既没有牛奶也没有咖啡；大多数论文中，迈克和吉姆都不是作者；……
- *零不变*性是一个重要的属性
- 如果空交易不占主导地位，则提升、χ^2和余弦是很好的测量方法
  - 否则，应使用*库尔钦斯基+失衡*率来判断模式的有趣程度。

第6章：挖掘频繁模式、关联和相关性：基本概念和方法

- 基本概念
- 高效的模式挖掘方法
- 模式评估
- 摘要

摘要

- 基本概念
  - 什么是模式发现？ 为什么它很重要？
  - 基本概念。频繁模式和关联规则
  - 压缩表示。封闭模式和最大模式
- 高效的模式挖掘方法
  - 频繁模式的向下闭合特性
  - Apriori算法
  - Apriori的扩展或改进
  - 通过探索垂直数据格式挖掘频繁模式
  - FPGrowth: 频繁模式增长的方法
  - 挖掘封闭模式
- 模式评估
  - 模式挖掘中的趣味性度量
  - 有趣的度量。提升和χ^2
  - 零不变的度量
  - 趣味性度量的比较

Recommended Readings (Basic Concepts)

- R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, in Proc. of SIGMOD'93
- R. J. Bayardo, “Efficiently mining long patterns from databases”, in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules”, in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery, 15(1): 55-86, 2007

Recommended Readings (Efficient Pattern Mining Methods)

- R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, “An efficient algorithm for mining association rules in large databases”, VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, “An effective hash-based algorithm for mining association rules”, SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, “Integrating association rule mining with relational database systems: Alternatives and implications”, SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, “Parallel algorithm for discovery of association rules”, Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD’00
- M. J. Zaki and Hsiao, “CHARM: An Efficient Algorithm for Closed Itemset Mining”, SDM'02
- J. Wang, J. Han, and J. Pei, “CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets”, KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, “Frequent Pattern Mining Algorithms: A Survey”, in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014

Recommended Readings (Pattern Evaluation)

- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS’98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE’03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010