日期: /

关联分析

一. 频繁项集

1. 购物篮问题

事务 (Transactions)	项 (Items)
1	Bread, Jelly, Peanut, Butter
2	Bread, Butter
3	Bread, Jelly
4	Bread, Milk, Butter
5	Chips, Milk
6	Bread, Chips
7	Bread, Milk
8	Chips, Jelly

I为所有项的集合.

I={Bread, Jelly, Peanut, Butter, Milk, Chips}

2. 基础概念:

① 项 item: 购物篮中的每一个商品.

[例]: Bread, Milk, Chocolate]

② 项集 itemset: 购物篮中多个项组成的集合.

[例]: {Bread, Jelly}

- 每个顾客购买的所有商品都是一个项集.
- 项集的长度: 项集中项的个数.
- k-项集: 含有k个项的项集.

[例] {Bread, Jelly} 为2-项集]

③ 事务 (transaction): 包含一个标识, TID 和购买的商品集合.

- I的每个非空子集都称为一个事务.
- 事务数据库: 所有事务构成事务数据库.
- 事务标识 TID: 每一个事务有唯一的标识.

日期: /

二. 关联规则

1. 目的: 找出事务数据库中多次重复出现的项之间的关联

2. 基本概念:

设 $I=\{i_1, i_2, \dots, i_m\}$ 是所有项 (item) i_j 的集合

T 是所有事务构成的集合: $T=\{t_1, t_2, \dots, t_n\}$, T 为 I 的一个非空子集. 每一个事务 t_i 是一个项集, 即 $t_i \subseteq I$

关联规则形式化为 $P \rightarrow Q$ 的蕴含式, 其中 $P \subset I$, $Q \subset I$, 且 $P \cap Q = \varnothing$

3. 例: 关联规则 Bread → Butter.

4. 关联规则兴趣度(有效性)度量: 支持度和置信度.

5. 支持度 Support: 度量规则在事务中出现频率

① 项 X 的支持度计数: 数据库 D 中包含项 X 的事务的数目.

② 项集 X 的支持度计数: 数据库 D 中包含项集 X 的事务的数目.

标识 (TID)	项 (Items)
1	Bread, Jelly, Peanut, Butter
2	Bread, Butter
3	Bread, Jelly
4	Bread, Milk, Butter
5	Chips, Milk
6	Bread, Chips
7	Bread, Milk
8	Chips, Jelly

项 (Item)	支持度计数 (Support Number)
Bread	6
Butter	3
Chips	2
Jelly	3
Milk	3
Peanut	1

项集 (Itemset)	支持度计数(Support Number)
Bread, Butter	3
...	
Bread, Butter, Chips	0
...	
Bread, Butter, Chips, Jelly	0
...	
Bread, Butter, Chips, Jelly, Milk	0
...	
Bread, Butter, Chips, Jelly, Milk, Peanut	0

日期: /

③ 项(或项集)X的支持度：包含项(项集)X的事务数与总事务数(#D)的比值.

$$sup(X) = \frac{\#X}{\#D}$$

标识 (TID)	项 (Items)
1	Bread, Jelly, Peanut, Butter
2	Bread, Butter
3	Bread, Jelly
4	Bread, Milk, Butter
5	Chips, Milk
6	Bread, Chips
7	Bread, Milk
8	Chips, Jelly

⇒

项 (Item)	支持度 (Support)
Bread	6/8
Butter	3/8
Chips	2/8
Jelly	3/8
Milk	3/8
Peanut	1/8

项集 (Itemset)	支持度 (Support)
Bread, Butter	3/8
...	
Bread, Butter, Chips	0/8
...	
Bread, Butter, Chips, Jelly	0/8
...	
Bread, Butter, Chips, Jelly, Milk	0/8
...	
Bread, Butter, Chips, Jelly, Milk, Peanut	0/8

④ 关联规则 $X \to Y$ 的支持度计数：同时包含项集X和Y的事务数

⑤ 规则的支持度(sup)：D中包含项集X和项集Y的事务数与数据库D中的事务数的比.

$$sup(X \to Y) = P(X \cup Y) = \frac{\#(X \cup Y)}{\#D}$$

6. 规则置信度 confidence：度量了规则的强度.

① 定义：指在数据库D中同时包含两个项集X,Y的事务数与包含项集X的事务数的比率

$$conf(X \to Y) = \frac{\#(X \cup Y)}{\#(X)}$$

条件概率：$conf(X \to Y) = P(Y|X) = \frac{\#(X \cup Y)}{\#(X)}$

日期: /

特定的例子：

标识 (TID)	项 (Items)
1	Bread, Jelly, Peanut, Butter
2	Bread, Butter
3	Bread, Jelly
4	Bread, Milk, Butter
5	Chips, Milk
6	Bread, Chips
7	Bread, Milk
8	Chips, Jelly

Bread → Milk.

$Sup(Bread \to Milk) = \frac{\#(Bread \cup Milk)}{\#D}$

$= \frac{2}{8} = \frac{1}{4}$

$Conf(Bread \to Milk) = \frac{\#(Bread \cup Milk)}{\#(Bread)}$

$= \frac{2}{6} = \frac{1}{3}$

同理 Milk → Bread $Sup = \frac{2}{8}$, $Conf = \frac{2}{3}$.

因此买牛奶的人会去买面包的概率比买面包的人会再买牛奶的概率更强。

日期: /

三 频繁项集和强规则

1. 频繁项集：是支持度大于 θ 的项集，即

$$sup(X \to Y) > minsup = \theta.$$

频繁 k 项集的集合通常记为 L_k.

2. 强关联规则：同时满足最小支持度 (min sup) 和最小置信度 (min conf) 的规则

$$\begin{cases} sup(X \to Y) > min\ sup \\ conf(X \to Y) > min\ conf. \end{cases}$$

3. 关联规则挖掘

① 关联规则问题.

给定所有商品 I，数据库 D，两个阈值 θ，ϕ，求满足 $X \to Y$ 的所有强规则，即为关联规则挖掘

② 关联规则挖掘大致流程.

step 1：求所有频繁项集.

step 2：使用频繁项集去产生关联规则.

对每一个频繁项集 f，生成 f 的所有非空子集.

对 f 的每一个非空子集 s，输出 $s \to (f-s)$，

若 $\frac{sup(f)}{sup(s)} > \phi$

※ 说明 1：当数据库中某商品出现概率很大，可以说阿果出现概率少的商品的人一定会阿果

日期: /

出现概率大的商品，X 错误观点.

※误区：冰淇淋消费量越多，犯罪率越高.

相关≠因果.

2个变量之间存在关联，不代表存在因果关系.

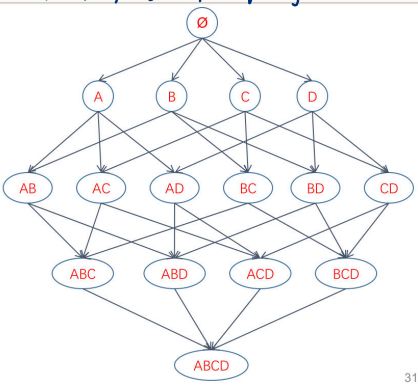
原始不可取方法

- **挖掘关联规则的一种原始方法是：暴力法(Brute-force approach):**
  - 计算每个可能规则的支持度和置信度
  - 这种方法计算代价过高，因为可以从数据集提取的规则的数量达指数级
  - 从包含d个项的数据集提取的可能规则的总数 $R=3^d-2^{d+1}+1$，如果d等于6，则R=602

③ 生成频繁项集 (Frequent Itemset Generation)

a) 目标是发现满足最小支持度阈值的所有项集.

b) 蛮力法：枚举所有可能项集. 一个项集包含k个项的数据集可以产生除空集外的2^k-1

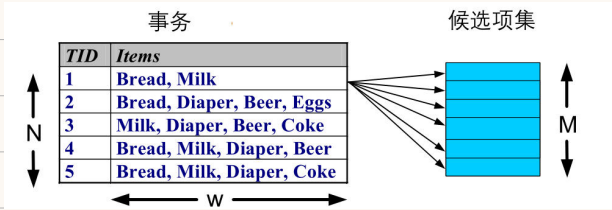


日期: /

3, 暴力方法:

把格结构中每个项集作为候选项集

将每个候选项集和每个事务进行比较,确定每个候选项集的支持度计数



时间复杂度 ~ O(NMw)

4, 减少候选项集的数量: 先验原理 Apriori:

- **关键思路 (基于先验原理 (Apriori principle)):**
  - **如果一个项集 X 是频繁的，则它的所有非空子集 $Y \subset X$ 一定也是频繁的**
    - {Milk, Bread} 是频繁的 → {Milk}, {Bread} 是频繁的
  - **如果一个项集 X 是非频繁的，则它的所有超集 $Y \supset X$ 也一定是非频繁的**
    - {Coke} 是非频繁的 → {Milk, Coke} 是非频繁的

基于支持度的剪枝 (support-based pruning)

性质: 一个项集的支持度不会超过它子集的支持度。[支持度度量的反单调性]

$$support(X \cup Y) \leq support(X)$$

日期: /

Apriori 算法:

step 1: 产生一个特定大小的项集.

step 2: 扫描数据库一次以查看它们中的哪些是频繁的

step 3: 使用频繁项集来生成 size = size + 1 的候选项集

step 4: 迭代地找到基数从 1 到 k 的频繁项集.

C_k: 大小为 k 的候选项集

L_k: 大小为 k 的频繁项集

$L_1 \leftarrow \{\text{频繁项集}\}$ (找出所有单个项的频繁项集)

```
for (k=1; L_k≠∅; k++)
    C_{k+1} ← candidate(L_k)   连接步          候选(candidates)
    for 每一个事务 t (扫描数据库中的每条记录)
        Q ← {c | c ∈ C_{k+1} ∧ c ⊆ t}
        count[c] ← count[c] + 1,   ∀c ∈ Q       计数(counting)
    end for
    L_{k+1} ← {c | c ∈ C_{k+1} ∧ count[c]/N ≥ σ}   过滤(filtering)
end for
return ⋃_k L_k
```

- **连接步**: 候选k-项集C_k由频繁(k-1)-项集 L_{k-1}与其自身连接生成
- **剪枝步**: 任意非频繁(k-1)-项集不可能是一个频繁k-项集的子集
- Apriori 算法的频繁项集产生的部分有两个重要的特点:
  - 它是一个逐层算法。即从频繁1-项集到最长的频繁项集，它每次遍历项集格中的一层
  - 它使用产生-测试策略来发现频繁项集。在每次迭代，新的候选项集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，并与最小支持度阈值进行比较。
  - 该算法需要的总迭代次数是 k_{max}+1，其中 k_{max} 是频繁项集的最大长度

日期: /

候选集生成举例：

频繁项集 $L_1=\{1,2,3,4,5\}$ $L_2=\{\{1,2\},\{2,3\}\}$

C_3 候选项集产生方法1：频繁1-项集与频繁2-项集进行连接

$$\{X \cup p \mid X \in L_k, p \in L_1, p \notin X\}$$

候选项集 $C_3=\{\{1,2,3\},\{1,2,4\},\{1,2,5\},\{2,3,4\},\{2,3,5\}\}$ 不紧凑

非频繁 {1,3}不在L_2中　　非频繁 {1,5}不在L_2中

C_3 候选项集产生方法2：频繁2-项集与其自身进行连接

条件：X 和 Y 只有一位不同

$$\{X \cup Y \mid X, Y \in L_k, \mid X \cap Y \mid = k-1\}$$

候选项集 $C_3=\{\{1,2,3\}\}$ 不紧凑

非频繁 (因{1,3}非频繁)

Apriori 真正采用的候选项集生成策略

(1) Apriori算法假设事务或项集里的各项已经预先按字典顺序进行排列

(2) 只对 L_k 中那些 X 和 Y 前 k-1 项相同且第 k 项不同的频繁项集进行连接，生成 C_{k+1} 的候选项集，连接方法如下：

候选项集产生 $\{X \cup Y \mid X, Y \in L_k, X_i = Y_i, \forall i \in [1, k-1], X_k \neq Y_k\}$ 有序列表

为 k-1 相同 第k不同

$L_2=\{\{1,2\},\{2,3\}\}$ $C_3=\{\}$

$L_2=\{\{1,2\},\{1,3\},\{2,3\}\}$ $C_3=\{\{1,2,3\}\}$

$L_3=\{\{1,2\},\{1,3\}\}$ $C_3=\{\{1,2,3\}\}$ 不符合

日期: /

4.3.1

挖掘频繁项集 (支持度 ≥ 50% sup ≥ 2)

数据库 D

TID	项集
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

扫描D

C_1 项集 — 支持度计数

项集	支持度计数
{1}	2
{2}	3
{3}	3
{4}	1 X
{5}	4

剪枝 → L_1 项集 — 支持度计数

项集	支持度计数
{1}	2
{2}	3
{3}	3
{5}	4

连接 → C_2 项集 — sup

项集	sup
{1,2}	1 X
{1,3}	2
{1,5}	1 X
{2,3}	2
{2,5}	3
{3,5}	2

→ L_2 项集 — sup

项集	sup
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

→ C_3 itemsets — sup

itemsets	sup
{2,3,5}	2

频繁项集	支持度计数
{1}	2
{2}	3
{3}	3
{5}	3
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2
{2,3,5}	2

sup ≥ 50%
conf ≥ 70%

{1,3}产生规则: **1→3(sup=2/4=50%, conf=2/2=1)** ✓
3→1(sup=2/4=50%, conf=2/3≈66.7%)
{2,3}产生规则: 2→3(sup=2/4=50%, conf=2/3≈66.7%)
3→2(sup=2/4=50%, conf=2/3≈66.7%)
{2,5}产生规则: **2→5(sup=3/4=75%, conf=3/3=100%)** ✓
5→2(sup=3/4=75%, conf=3/3=100%) ✓
{3,5}产生规则: 3→5(sup=2/4=50%, conf=2/3≈66.7%)
5→3(sup=2/4=50%, conf=2/3≈66.7%)
{2,3,5}产生规则: 2→3∪5(sup=2/4=50%, conf=2/3≈66.7%)
3→5∪2(sup=2/4=50%, conf=2/3≈66.7%)
5→2∪3(sup=2/4=50%, conf=2/3≈66.7%)
2∪3→5(sup=2/4=50%, conf=2/2=100%) ✓
2∪5→3(sup=2/4=50%, conf=2/3≈66.7%)
3∪5→2(sup=2/4=50%, conf=2/3=100%) ✓

日期: /

例 2.

数据库 D

TID	项集
10	A B C
20	A C
30	A D
40	B E F

(1) 根据给定的数据库D计算所有的频繁项集
(2) 根据频繁项集给出满足最小支持度和最小置信度的强关联规则

频繁项集最小支持度=50%(即支持度计数≥2)

关联规则支持度≥ 50%(即支持度计数≥2), 置信度≥70%

数据库 D

TID	项集
10	A B C
20	A C
30	A D
40	B E F

扫描 D

C_1

项集	支持度计数
{A}	3
{B}	2
{C}	2
{D}	1
{E}	1
{F}	1

L_1

项集	支持度计数
{A}	3
{B}	2
{C}	2

C_2

项集
{A B}
{A C}
{B C}

扫描 D

C_2

项集	支持度计数
{A B}	1
{A C}	2
{B C}	1

L_2

项集	支持度计数
{A C}	2

频繁项集	支持度
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

要求最小支持度=50%, 最小置信度=70%
对于规则 A→C:
支持度: Support(A∪C)=50%
置信度: Support(A∪C)/Support(A)=66.6%
对于规则 C→A:
支持度: Support(C∪A)=50%
置信度: Support(C∪A)/Support(C)=100%

例 3: sup ≥ 20% conf ≥ 50%

Transaction	Items	Transaction	Items
t_1	Blouse	t_{11}	TShirt
t_2	Shoes, Skirt, TShirt	t_{12}	Blouse, Jeans, Shoes, Skirt, TShirt
t_3	Jeans, TShirt	t_{13}	Jeans, Shoes, Shorts, TShirt
t_4	Jeans, Shoes, TShirt	t_{14}	Shoes, Skirt, TShirt
t_5	Jeans, Shorts	t_{15}	Jeans, TShirt
t_6	Shoes, TShirt	t_{16}	Skirt, TShirt
t_7	Jeans, Skirt	t_{17}	Blouse, Jeans, Skirt
t_8	Jeans, Shoes, Shorts, TShirt	t_{18}	Jeans, Shoes, Shorts, TShirt
t_9	Jeans	t_{19}	Jeans
t_{10}	Jeans, Shoes, TShirt	t_{20}	Jeans, Shoes, Shorts, TShirt

日期: /

I = {Blouse, Shoes, Skirt, TShirt, Jeans, Shorts}

C_1 项集	支持度计数	→	L_1 项集	支持度计数 →
Blouse	3 X		Shoes	10
Shoes	10		Skirt	6
Skirt	6		Tshirt	14
Tshirt	14		Jeans	14
Jeans	14		Shorts	5
Shorts	5			

C_2 项集	支持度计数	→ L_2 项集	
{Shoes, Skirt}	3 X	{Jeans, shoes}	7
{Shoes, Tshirt}	10	{Jeans, short}	5
{Shoes, Jeans}	7	{Jeans, Tshirt}	9
{Shoes, shorts}	4	{shoes, Short}	4
{Skirt, Tshirt}	4	{shoes, Tshirt}	10
{Skirt, Jeans}	3 X	{short, Tshirt}	4
{Skirt, shorts}	0 X	{Skirt, Tshirt}	4
{Tshirt, Jeans}	9		
{Tshirt, shorts}	4		
{Jeans, shorts}	5		

→ C_3 项集.. sup 计数 → L_3 sup计数

{Jeans, shoes, short} 4 {Jeans, shoes, short} 4

日期: /

{Jeans, shoes, Tshirt} ✓ {Jeans, shoes, Tshirt} ✓
{Jeans, short, Tshirt} ✓ {Jeans, short, Tshirt} ✓
{shoes, short, Tshirt} ✓ {shoes, short, Tshirt} ✓

→ L4 C_4 项集. sup

{Jeans, shoes, short, Tshirt} 1 ✓

无疑规则. 略

四 序列模式挖掘

1. 序列模式寻找事务时间上的相关性

2. 基本概念:

I = {$i_1, i_2, i_3, \cdots, i_m$} 是所有项集合

① 序列: 元素的有序列表. , $S = \langle s_1, s_2, \cdots s_n \rangle$

其中. s_i 为一个或多个项的集合 $s_i = \{x_1, x_2, \cdots x_k\}$, $x_i \in I$.

② 项按字典序排列

③ 大小: 序列的大小是序列中元素的个数.

④ 长度: 一个序列长度是序列中所有项的个数

长度为 k ⇒ k-序列

⑤ 包含: 若 $t = \langle t_1, t_2, \cdots t_m \rangle$ 是 $S = \langle s_1, s_2, \cdots s_n \rangle$ 的子序列, 则 存在整数 $1 \le j_1 < j_2 < \cdots < j_m \le n$

日期: /

使得 $t_1 \subseteq s_{j_1}$, $t_2 \subseteq s_{j_2} \cdots t_m \subseteq s_{j_m}$, S为t的超序列 $t \subset S$

例1

- 设 I ={1, 2, 3, 4, 5, 6, 7, 8, 9}
- 序列<{3} {4,5} {8}>包含在<{6} {3,7} {9} {4,5,8} {3,8}> (或者说前者是后者的子序列)
  - {3} ⊆ {3,7}, {4,5} ⊆ {4,5,8}, {8} ⊆ {3,8}
  - 但是<{3},{8}>并不包含在<{3,8}>中, 反之也成立
  - 序列<{3} {4,5} {8}>的大小是3, 序列长度是4

s	t	Y/N
<{2, 4} {3, 6, 5} {8}>	<{2} {3, 6} {8}>	Yes
<{2, 4} {3, 6, 5} {8}>	<{2} {8}>	Yes
<{1, 2} {3, 4}>	<{1} {2}>	No
<{2, 4} {2, 4} {2, 5}>	<{2} {4}>	Yes

例2:

- 假定没有时限约束，列举包含在下面的数据序列中的所有4-子序列 设有4个

<{1,3}, {2}, {2,3}, {4}>

共有 $C_6^4 = \frac{6\times5\times4\times(6-4+1)}{4\times3\times2\times1} = 15$ 个. 从6中取2个去掉

{{1,3},{2},{3}} {{1,3},{2},{4}} {{1,3},{2,3},{4}} {{1,3},{3},{4}} {{1},{2},{2,3},{4}}

略.

- 列举如下:
- < {1, 3} {2} {2}>, <{1, 3} {2} {3} >, < {1, 3} {2} {4}>,
- <{1, 3}{2, 3} >, < {1, 3} {3} {4}>, <{1} {2} {2, 3} >,
- < {1} {2} {2} {4}>, <{1} {2} {3} {4} >, < {1} {2, 3} {4}>,
- <{3} {2} {2, 3} >, < {3} {2} {2} {4}>, <{3} {2} {3} {4} >,
- < {3} {2, 3} {4}>,<{2} {2, 3} {4} >

日期: /

例3:

- 假定没有时限约束，列举包含在下面的数据序列中的所有3个元素 (项集) 的子序列

<{1,3}, {2}, {2,3}, {4}>

$C_4^3 \Rightarrow 4$

① <{1},{2,3},{4}> <{2},{2,3},{4}> <{2},{3},{4}>

② <{1,3},{2,3},{4}> <{1},{2,3},{4}>

<{1},{2},{4}> <{1},{3},{4}>

<{3},{2,3},{4}> <{3},{2},{4}>

<{3},{3},{4}> <{1,3},{2},{4}>

<{1,3},{3},{4}>

③ ~~<{1,3},{2},{4}> <{1},{2},{4}> <{3},{2},{4}>~~

④ <{1,3},{2},{2,3}> <{1},{2},{2}>

<{1},{2}{3}> <{1},{2},{2,3}>

<{3},{2},{2}> <{3},{2}{3}>

<{3},{2},{2,3}> <{1,3},{2}{2}>

<{1,3}{2}{3}>

去重

3. 序列挖掘目标

① 数据序列：与单个对象相关联的事件的有

日期: /

序列表.

② S包含一个或多个数据序列的数据集.

③ 序列s支持度包含S的所有数据序列所占比例.

④ 频繁序列：若序列S在数据库S中支持度大于或等于用户指定的最小支持度阈值，则称序列S为频繁序列.

⑤ 序列支持度计数：S中包含该序列个数.

⑥ 长度为1（项数为1）的序列模式记为1-模式.

- 给定输入数据序列集(或序列数据库) S，挖掘序列模式的问题是求满足用户指定最小支持度的所有序列，即该子序列在序列集中的出现频率大于或等于用户指定的最小支持度阈值

例

区分不同的客户

CID	时间戳	项
A	1	1, 2, 4
A	2	2, 3
A	3	5
B	1	1, 2
B	2	2, 3, 4
C	1	1, 2
C	2	2, 3, 4
C	3	2, 4, 5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

$S_1 = \langle \{1,2,4\}, \{2,3\}, \{5\} \rangle$

$S_2 = \langle \{1,2\}, \{2,3,4\} \rangle$

$S_3 = \langle \{1,2\}, \{2,3,4\}, \{2,4,5\} \rangle$

$S_4 = \langle \{2\}, \{3,4\}, \{4,5\} \rangle$

$S_5 = \langle \{1,3\}, \{2,4,5\} \rangle$

$S = \langle S_1, S_2, S_3, S_4, S_5 \rangle$

日期: /

$supp(\langle \{1,2,4\} \rangle) = \frac{\#\langle \{1,2,4\} \rangle}{\#S} = \frac{3}{5}$ (A, B, C)

$supp(\langle \{1,3\},\{2,4\} \rangle) = \frac{\#\langle \{1,3\},\{2,4\} \rangle}{\#S} = \frac{4}{5}$ (A, B, C, E)

4. 先生候选序列空间

① • 先验原理 (Apriori 算法) 对序列数据成立

• 如果一个k-序列是频繁的，则它的所有(k-1)-子序列也一定是频繁的 。

② **算法 7.1** 序列模式发现的类 *Apriori* 算法

```
1:  k = 1。
2:  F_k = {i | i ∈ I ∧ σ({i})/N ≥ minsup}。   {找出所有的频繁 1-序列。}
3:  repeat
4:    k = k + 1。
5:    C_k = apriori-gen(F_{k-1})。            {产生候选 k-序列。}
6:    for 每个数据序列 t ∈ T do
7:      C_t = subsequence(C_k, t)。           {识别包含在 t 中的所有候选。}
8:      for 每个候选 k-序列 c ∈ C_t do
9:        σ(c) = σ(c) +1。                    {支持度计数增值。}
10:     end for
11:   end for
12:   F_k = {c | c∈ C_k ∧ σ(c)/N ≥ minsup}。  {提取频繁 k-序列。}
13: until F_k = ∅。
14: Answer = ∪ F_k。
```

[长为k-1项数列]

Apriori中当前k-1项相同且第k项不同时才合并一对频繁k-项集.

③合并方法：2种情况

日期: /

序列 $s^{(1)}$与另一个序列 $s^{(2)}$合并，仅当从 $s^{(1)}$中去掉第一个事件得到的子序列与从 $s^{(2)}$中去掉最后一个事件得到的子序列相同。结果候选是序列 $s^{(1)}$与 $s^{(2)}$的最后一个事件的连接。$s^{(2)}$的最后一个事件可以作为最后一个事件合并到 $s^{(1)}$的最后一个元素中，也可以作为一个不同的元素，取决于如下条件：

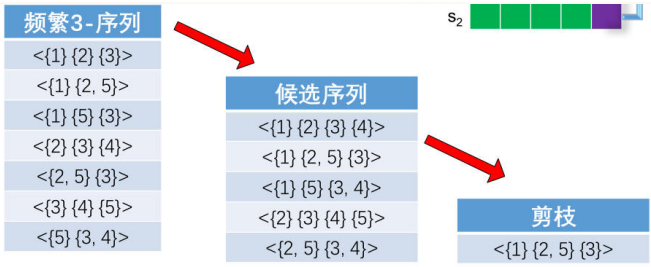
(1) 如果 $s^{(2)}$的最后两个事件属于相同的元素，则 $s^{(2)}$的最后一个事件在合并后的序列中是 $s^{(1)}$的最后一个元素的一部分。

(2) 如果 $s^{(2)}$的最后两个事件属于不同的元素，则 $s^{(2)}$的最后一个事件在合并后的序列中成为连接到 $s^{(1)}$的尾部的单独元素。

- 例子
  - <{1} {2} {3} {4}>是通过合并<{1} {2} {3}>和<{2} {3} {4}>得到 。由于事件3和事件4属于第二个序列的不同元素，它们在合并后序列中也属于不同的元素。
  - <{1} {5} {3,4} >通过合并<{1} {5} {3}>和<{5} {3,4}>得到 。由于事件3和事件4属于第二个序列的相同元素，4被合并到第一个序列的最后一个元素中。

④候选剪枝：

- 一个候选k-序列被剪枝，如果它的(k-1)-序列最少有一个是非频繁的。
- 例如，假设<{1} {2} {3} {4}>是一个候选4-序列。我们需要检查<{1} {2} {4}>和<{1} {3} {4}>是否是频繁3-序列。由于它们都不是频繁的，因此可以删除候选<{1} {2} {3} {4}>。



日期: /

□考虑以下频繁3-序列: < {1, 2, 3} >, < {1, 2}{3} >, < {1}{2, 3} >, < {1, 2}{4} >, < {1, 3}{4} >, < {1, 2, 4} >, < {2, 3}{3} >, < {2, 3}{4} >, < {2}{3}{3} >, 和 < {2}{3}{4} >

- (1) 列举出候选生成步骤产生的所有候选4-序列

  所有候选4-序列列举如下:

  < {1, 2, 3} {3} >, < {1, 2, 3} {4} >, < {1, 2} {3} {3} >, < {1, 2} {3} {4} >, < {1} {2, 3} {3} >, < {1} {2, 3} {4} >
- (2) 列出候选剪枝步骤剪掉的所有候选4-序列(假定没有时限约束)。如果没有时间限制，则所有候选子序列都必须频繁。 因此，经过修剪的候选子序列为:

  < {1, 2, 3} {3} >, < {1, 2} {3} {3} >, < {1, 2} {3} {4} >, < {1} {2, 3} {3} >, < {1} {2, 3} {4} >

  剪枝后的候选序列为: < {1, 2, 3} {4} >