日期: /

一 关联规则

- Step 1: 求所有频繁项集. 先验原则
- 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 → (f-s) 如果 support (f) / support (s) > Φ (找出所有可能的关联规则, 然后再进行校验)

step1:

(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 $\{X \cup Y | X, Y \in L_k, X_i = Y_i, \forall i \in [1, k-1], X_k \neq Y_k\}$ 有序列表

举例:

数据库 D

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

最小支持度 = 50% ⟺ 2.
置信度 ≥ 70%.

解: 构造频繁项集如下

C_1:

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

⇒ 删去 {4}

L_1:

项集	sup num
{1}	2
{2}	3
{3}	3
{5}	3

日期: /

C_2

⇒ 项集	sup num	⇒ L_2 项集	sup num
{1,2}	1	删 {1,2}	{1,3} 2
{1,3}	2	{1,5}	{2,3} 2
{1,5}	1	{2,3}	{2,5} 3
{2,3}	2		{3,5} 2
{2,5}	3		
{3,5}	2		

⇒ C_3 项集 sup num ⇒ L_3 项集 sup num

{2,3,5} 2 {2,3,5} 2

新集项集

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

{1,3} 如何 ☆ 1→3 sup=$\frac{2}{4}$ conf=$\frac{2}{2}$

3→1 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

{2,3} 2→3 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

3→2 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

{2,5} ☆ 2→5 sup=$\frac{3}{4}$ conf=$\frac{3}{3}$

☆ 5→2 sup=$\frac{3}{4}$ conf=$\frac{3}{3}$

{3,5} 3→5 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

5→3 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

{2,3,5} 2→3^5 sup=$\frac{2}{4}$ conf=$\frac{2}{3}$

日期: /

3 U 5 → 2 ~ conf = $\frac{2}{2}$ ☆

3 → 2 U 5 ~ conf = $\frac{2}{3}$

2 U 5 → 3 ~ conf = $\frac{2}{3}$

5 → 2 U 3 ~ conf = $\frac{2}{3}$

2 U 3 → 5 ~ conf = $\frac{2}{2}$ ☆

日期: /

二. 序列模型

- **大小**(size): 序列的大小是序列中元素(或项集)的个数
- **长度**: 一个序列的长度是序列中所有项的个数 k-序列
- 称 $t=<t_1\ t_2\ \cdots\ t_m>$ 是 $s=<s_1\ s_2\ \cdots\ s_n>$ 的一个**子序列**如果存在整数 $1 \le j_1<j_2< \cdots <j_m \le n$ 使得 $t_1 \subseteq s_{j_1}, t_2 \subseteq s_{j_2}, \cdots, t_m \subseteq s_{j_m}$. 我们也称 s 是 t 的**超序列**, 或 s **包含** t, 记为 $t \subset s$

举例 计算子序列

- 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}>

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

- 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

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

15-9

- 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: /

- **序列 s 的支持度**是包含 s 的所有数据序列所占的比例
- **频繁序列 (或序列模式):** 如果序列 s 在序列数据库 S 中的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** (或**序列模式**)
- **序列的支持度计数**是 S 中包含该序列的总数据序列个数
- 长度为1的序列模式记为1-模式

举例：支持度计算

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

数据集 S 含5个数据序列，
A、B、C、D、E各一个

支持度	
<{1, 2}>	60%
<{2, 3}>	60%
<{2, 4}>	80%
<{3} {5}>	80%
<{1} {2}>	80%
<{2} {2}>	60%
<{1} {2, 3}>	60%
<{2} {2, 3}>	60%
<{1, 2} {2, 3}>	60%

$$supp(<\{1,2\}>) = \frac{\#<\{1,2\}>}{\#S} = \frac{3}{5} = 60\%$$

$$supp(<\{1\},\{2\}>) = \frac{\#<\{1\},\{2\}>}{\#S} = \frac{4}{5} = 80\%$$

60

$S_1 = <\{1,2,4\},\{2,3\},\{5\}>$

日期: /

序列合并：

序列合并过程

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。

日期: /

- 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}>。
- 支持度计数
  - 在支持度计数期间，算法将枚举属于一个特定数据序列的所有候选k-序列。
  - 计数之后，算法将识别出频繁k-序列，并可以丢弃其支持度计数小于最小支持度阈值 minsup 的候选。

□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} >

日期: /