第七章：关联分析

相似性度量，相异性（距离）度量：**欧几里得距离考！Kmeans 里用到，层次聚类用到**
常见的度量有哪些
聚类分析：簇距离度量
数据类型：不考；数据预处理：不考；特征选择：不考；连续离散属性：不考
最重要的：强关联规则挖掘（频繁项集的挖掘-先验原理 2 个；强关联规则构造-支持度和置信度大于阈值，PPT 例子）序列模式挖掘（序列模式强关联规则计算未讲，关键是如何构造频繁序列，未讲的不考）

一、强关联规则挖掘
1、频繁项集和强关联规则

- **频繁项集**是**支持度**大于 σ 的项集 (满足最小支持度阈值 σ 的所有项集或者说出现的**频率**大于 σ), 即满足

  $$sup(X \rightarrow Y) > minsup$$

  的项集。频繁 k 项集的集合通常记为 L_k
- **强关联规则(强规则)**是一种同时满足最小支持度($minsup$)和最小置信度($minconf$)的规则 或者说条**件概率**大于Φ, 即满足

  $$sup(X \rightarrow Y) > minsup \text{ 且 } \mathrm{conf}(X \rightarrow Y) > minconf$$

  的规则

2、强关联规则挖掘流程

- 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 → (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)

3、频繁项集计算方法：减少候选项集的数目（先验原理），减少比较次数

- **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
  - 这种基于支持度度量修剪指数搜索空间的策略称为**基于支持度的剪枝**（support-based pruning）
  - 这种剪枝策略依赖于支持度度量的一个关键性质，即一个项集的支持度决不会超过它的子集的支持度。这个性质也称为**支持度度量的反单调性**(anti-monotone), 即 $\mathrm{support}(X \cup Y) \leq \mathrm{support}(X)$

- **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

真正使用的办法:

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

二、序列模式挖掘：序列模式寻找的是事务(项)之间时间上的相关性

1、基础概念:

- 设 I ={i_1,i_2,…,i_m} 是所有项 (Item) i_j (j=1,2,…,m) 的集合。
- 序列: 元素(项集，事务)的有序列表
- 元素/项集: 非空项集 X⊂ I, 表示序列 s 为 s=<s_1 s_2 … s_n>, 其中 s_i 是一个或多个项(事件)的集族, 也叫作 s 的一个元素 (element)
- 序列的一个元素(或项集)表示为{x_1, x_2,…,x_k}, 其中 x_j∈ I 是一个项
- 不失一般性, 假设序列中元素的项是按**字典顺序**排列的
- **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
  - 长度为k 的序列称为***k*-序列**
- 称 t=<t_1 t_2 … t_m> 是 s=<s_1 s_2 … s_n> 的一个**子序列**如果存在整数 1 ≤ j_1<j_2< … <j_m≤n 使得 t_1⊆ s_{j_1}, t_2⊆ s_{j_2},…, t_m⊆ s_{j_m}. 我们也称 s 是 t 的**超序列**, 或 s **包含** t, 记为 t ⊂ s

2、序列模式

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

- 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 。

候选产生

- 一对频繁(k-1)-序列合并，产生候选k-序列。
- 为了避免重复产生候选，传统的Apriori算法仅当前k-1项相同且第看k项不同时才合并一对频繁k-项集。类似的方法可以用于序列。
- 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。

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。

候选生成 (Candidate Generation)

- 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。