第六章 挖掘频繁模式，关联和相关性：基本概念和方法

笔记本：	数据挖掘：概念与技术
创建时间：	2017/12/20 15:43	更新时间：	2017/12/26 11:25
作者：	Passero

频繁模式是频繁地出现在数据集中的模式（如项集，子序列或子结构）
---频繁项集
e.g. 频繁的同时出现在交易数据集中的商品（如牛奶和面包）的集合
---频繁的序列模式
e.g. 一个子序列，如首先购买PC，然后是数码相机，然后是内存卡，如果它频繁的出现在购物历史数据库中，则称它为一个频繁的序列模式
---频繁的结构模式
e.g. 一个子结构可能涉及不同的结构形式，如子图，子树或子格，它可能与项集或子序列结合在一起。如果一个子结构频繁的出现，则称它为频繁的结构模式

6.1基本概念

频繁模式挖掘搜索给定数据集中反复出现的联系

购物篮分析：一个诱发例子

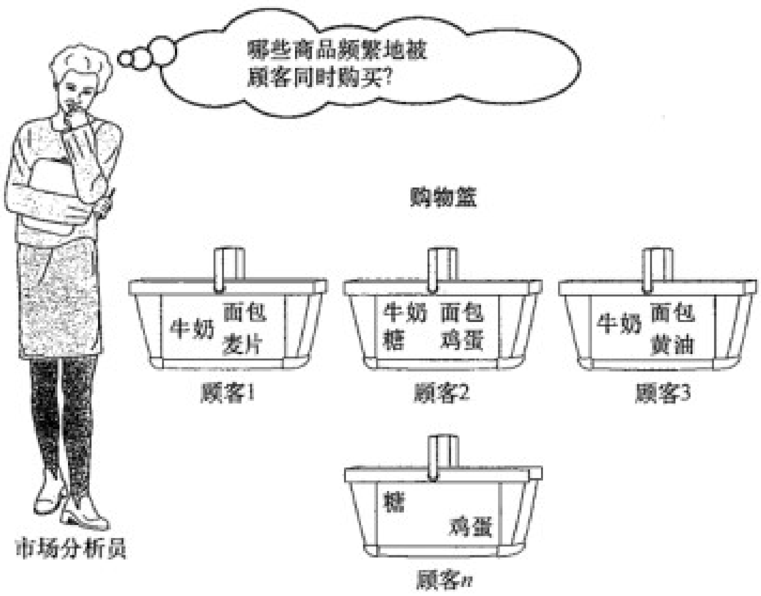

图 6.1 购物篮分析

例 6.1 购物篮分析。假定作为 AllElectronics 的部门经理，你想更多地了解顾客的购物习惯。尤其是，你想知道"*顾客可能会在一次购物同时购买哪些商品?*"为了回答问题，可以在商店的顾客事务零售数据上运行购物篮分析。分析结果可以用于营销规划、广告策划，或新的分类设计。例如，购物篮分析可以帮助你设计不同的商店布局。一种策略是：经常同时购买的商品可以摆放近一些，以便进一步刺激这些商品同时销售。例如，如果购买计算机的顾客也倾向于同时购买杀毒软件，则把硬件摆放离软件陈列近一点，可能有助于增加这两种商品的销售。

另一种策略是：把硬件和软件摆放在商店的两端，可能诱发买这些商品的顾客一路挑选其他商品。例如，在决定购买一台很贵的计算机后，去看软件陈列，购买杀毒软件，途中看到销售安全系统，可能会决定也买家庭安全系统。购物篮分析也可以帮助零售商规划什么商品降价出售。如果顾客趋向于同时购买计算机和打印机，则打印机的降价出售可能既促使购买打印机，又促使购买计算机。 ■

如果我们想象全域是商店中商品的集合，则每种商品有一个布尔变量，表示该商品是否出现。每个购物篮可用一个布尔向量表示。可以分析布尔向量，得到反映商品频繁关联或同时购买的购买模式。这些模式可以用**关联规则**（association rule）的形式表示。例如，购买计算机也趋向于同时购买杀毒软件，可以用以下关联规则（6.1）表示：

$$computer \Rightarrow antivirus\_software[support = 2\%; confidence = 60\%] \qquad (6.1)$$

规则的**支持度**（support）和**置信度**（confidence）是规则兴趣度的两种度量。它们分别反映所发现规则的有用性和确定性。关联规则（6.1）的支持度为 2%，意味所分析的所有事务的 2% 显示计算机和杀毒软件被同时购买。置信度 60% 意味购买计算机的顾客 60% 也购买了杀毒软件。在典型情况下，关联规则被认为是有趣的，如果它满足**最小支持度阈值**和**最小置信度阈值**。这些阈值可以由用户或领域专家设定。还可以进行其他分析，揭示关联项之间有趣的统计相关性。

频繁项集、闭项集和关联规则

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence(A \Rightarrow B) = P(B \mid A)$$

其中，A非空，B非空，且A和B的交集非空。

同时满足最小支持度阈值(min_sup)和最小置信度阈值(min_conf)的规则称为强规则。通常，为方便计算，用0% ~ 100%之间的值，而不是0~1.0之间的值表示支持度和置信度。

项的集合称为项集，包含k个项的项集称为k项集。项集的出现频度是包含项集的事务数，简称为项集的频度，支持度计数或计数。

attention：

support(A=>B) = P(A∪B)，定义的项集支持度有时称为相对支持度，而出现频度称为绝对支持度。

若项集I的相对支持度满足预定义的最小阈值支持度，则I是频繁项集。频繁k项集的集合通常记为Lk。

$$confidence(A \Rightarrow B) = P(B \mid A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}$$

一般而言，关联规则的挖掘是一个两步的过程：

（1）**找出所有的频繁项集**：根据定义，这些项集的每一个频繁出现的次数至少与预定义的最小支持计数 min_sup 一样。

（2）**由频繁项集产生强关联规则**：根据定义，这些规则必须满足最小支持度和最小置信度。

项集 X 在数据集 D 中是**闭的**（closed），如果不存在真超项集 Y[⊖]使得 Y 与 X 在 D 中具有相同的支持度计数。项集 X 是数据集 D 中的**闭频繁项集**（closed frequent itemset），如果 X 在 D 中是闭的和频繁的。项集 X 是 D 中的**极大频繁项集**（maximal frequent itemset）或**极大项集**（max-itemset），如果 X 是频繁的，并且不存在超项集 Y 使得 $X \subset Y$ 并且 Y 在 D 中是频繁的。

6.2频繁项集挖掘方法

Apriori算法：通过限制候选产生发现频繁项集
链接：https://wizardforcel.gitbooks.io/dm-algo-top10/content/apriori.html
算法总述：该算法使用一种称为逐层搜索的迭代方法，其中k项集用于探索k+1项集。
步骤：
首先，通过扫描数据库，累积每个项的计数，并收集满足最小支持度的项，找出频繁1项集的集合，该集合记为L1。
然后，使用L1找出频繁2项集的集合L2，使用L2找出L3，如此下去，直到不能再找到频繁k项集。
（找出每个Lk需要一次数据库的完整扫描）

提高效率：先验性质：频繁项集的所有非空子集也一定是频繁的。该性质属于一类特殊的性质，即反单调性。
[反单调性：如果一个集合不能通过测试，则它的所有超集也都不能通过相同的测试。之所以是反单调，是因为在通不过测试的意义下，该性质是单调的。]

如何在算法中使用先验性质?
---如何使用Lk-1找出Lk(k>=2)

（1）**连接步**：为找出 L_k，通过将 L_{k-1} 与自身连接产生**候选** k 项集的集合。该候选项集的集合记为 C_k。设 l_1 和 l_2 是 L_{k-1} 中的项集。记号 $l_i[j]$ 表示 l_i 的第 j 项（例如，$l_1[k-2]$ 表示 l_1 的倒数第 2 项）。为了有效地实现，Apriori 算法假定事务或项集中的项按字典序排序。对于（$k-1$）项集 l_i，这意味把项排序，使得 $l_i[1] < l_i[2] < \cdots < l_i[k-1]$。执行连接 $L_{k-1} \bowtie L_{k-1}$；其中 L_{k-1} 的元素是可连接的，如果它们前（$k-2$）个项相同。即，L_{k-1} 的元素 l_1 和 l_2 是可连接的，如果 $(l_1[1]=l_2[1]) \wedge (l_1[2]=l_2[2]) \wedge \cdots \wedge (l_1[k-2]=l_2[k-2]) \wedge (l_1[k-1]<l_2[k-1])$。条件 $(l_1[k-1]<l_2[k-1])$ 是简单地确保不产生重复。连接 l_1 和 l_2 产生的结果项集是 $\{l_1[1], l_1[2], \cdots, l_1[k-1], l_2[k-1]\}$。

（2）**剪枝步**：C_k 是 L_k 的超集，也就是说，C_k 的成员可以是也可以不是频繁的，但所有的频繁 k 项集都包含在 C_k 中。扫描数据库，确定 C_k 中每个候选的计数，从而确定 L_k（即根据定义，计数值不小于最小支持度计数的所有候选都是频繁的，从而属于 L_k）。然而，C_k 可能很大，因此所涉及的计算量就很大。为了压缩 C_k，可以用以下办法使用先验性质。任何非频繁的（$k-1$）项集都不是频繁 k 项集的子集。因此，如果一个候选 k 项集的（$k-1$）项子集不在 L_{k-1} 中，则该候选也不可能是频繁的，从而可以从 C_k 中删除。这种**子集测试**可以使用所有频繁项集的散列树快速完成。

Apriori算法

算法 6.2.1 Apriori。使用逐层迭代方法基于候选产生找出频繁项集。

输入：

- D：事务数据库。
- min_sup：最小支持度阈值。

输出：L，D 中的频繁项集。

方法：

```
(1)   L_1 = find_frequent_1_itemsets(D);
(2)   for(k = 2; L_{k-1} ≠ ∅; k++) {
(3)      C_k = aproiri_gen(L_{k-1});
(4)      for each 事务 t ∈ D {              // 扫描 D，进行计数
(5)          C_t = subset(C_k, t);          // 得到 t 的子集，它们是候选
(6)          for each 候选 c ∈ C_t
(7)              c.count++;
(8)      }
(9)      L_k = {c ∈ C_k | c.count ≥ min_sup}
(10) }
(11) return L = ∪_k L_k;

procedure apriori_gen(L_{k-1}: frequent(k-1)itemset)
(1)   for each 项集 l_1 ∈ L_{k-1}
(2)      for each 项集 l_2 ∈ L_{k-1}
(3)         if(l_1[1] = l_2[1]) ∧ … ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-2]) then{
(4)            c = l_1 ⋈ l_2;               // 连接步：产生候选
(5)            if has_infrequent_subset(c, L_{k-1}) then
(6)                delete c;                // 剪枝步：删除非频繁的候选
(7)            else add c to C_k;
(8)   }
(9)   return C_k;

procedure has_infrequent_subset (c: candidate k itemset; L_{k-1}: frequent(k-1)itemset)
// 使用先验知识
(1)  for each(k-1)subset s of c
(2)     if s ∉ L_{k-1} then
(3)         return TRUE;
(4)  return FALSE;
```

由频繁项集产生关联规则

一旦由数据库 D 中的事务找出频繁项集，就可以直接由它们产生强关联规则（强关联规则满足最小支持度和最小置信度）。对于置信度，可以用（6.4）式计算。为完整起见，这里重新给出该式

$$confidence(A \Rightarrow B) = P(A \mid B) = \frac{support\_count(A \cup B)}{support\_count(A)}$$

条件概率用项集的支持度计数表示，其中，$support\_count(A \cup B)$ 是包含项集 $A \cup B$ 的事务数，而 $support\_count(A)$ 是包含项集 A 的事务数。根据该式，关联规则可以产生如下：

- 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 $\frac{support\_count(l)}{support\_count(s)} \geqslant min\_conf$，则输出规则"$s \Rightarrow (l-s)$"。

  其中，min_conf 是最小置信度阈值。

由于规则由频繁项集产生，因此每个规则都自动地满足最小支持度。频繁项集和它们的支持度可以预先存放在散列表中，使得它们可以被快速访问。

提高Apriori算法的效率

---基于散列的技术
---事务压缩
---划分
---抽样
---动态项集计数

挖掘频繁项集的模式增长方法

频繁模式增长（Frequent-Pattern Growth, FP-growth）

算法：FP-Growth。使用 FP 树，通过模式增长挖掘频繁模式。

输入：

- D：事务数据库。
- min_sup：最小支持度阈值。

输出：频繁模式的完全集。

方法：

1. 按以下步骤构造 FP 树：

   (a) 扫描事务数据库 D 一次。收集频繁项的集合 F 和它们的支持度计数。对 F 按支持度计数降序排序，结果为频繁项列表 L。

   (b) 创建 FP 树的根结点，以"null"标记它。对于 D 中每个事务 $Trans$，执行：

   选择 $Trans$ 中的频繁项，并按 L 中的次序排序。设 $Trans$ 排序后的频繁项列表为 $[p \mid P]$，其中 p 是第一个元素，而 P 是剩余元素的列表。调用 **insert_tree**($[p \mid P], T$)。该过程执行情况如下。如果 T 有子女 N 使得 $N.item\text{-}name = p.item\text{-}name$，则 N 的计数增加 1；否则，创建一个新结点 N，将其计数设置为 1，链接到它的父结点 T，并且通过结点链结构将其链接到具有相同 $item\text{-}name$ 的结点。如果 P 非空，则递归地调用 **insert_tree**(P, N)。

2. FP 树的挖掘通过调用 **FP_growth**(FP_tree, $null$) 实现。该过程实现如下。

```
procedure FP_growth(Tree, α)
(1) if Tree 包含单个路径 P then
(2)   for 路径 P 中结点的每个组合(记作 β)
(3)     产生模式 β∪α,其支持度计数 support_count 等于 β 中结点的最小支持度计数;
(4) else for Tree 的头表中的每个 a_i {
(5)     产生一个模式 β = a_i ∪α,其支持度计数 support_count = a_i.support_count;
(6)     构造 β 的条件模式基,然后构造 β 的条件 FP 树 Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)       调用 FP_growth(Tree_β, β); }
```

使用垂直数据格式挖掘频繁项集

Apriori 算法和 FP-growth 算法都从 *TID* 项集格式（即 {*TID*：*itemset*}）的事务集中挖掘频繁模式，其中 *TID* 是事务标识符，而 *itemset* 是事务 *TID* 中购买的商品。这种数据格式称为**水平数据格式**（horizontal data format）。或者，数据也可以用项 - *TID* 集格式（即 {*item*：*TID_set*}）表示，其中 *item* 是项的名称，而 *TID_set* 是包含 *item* 的事务的标识符的集合。这种格式称为**垂直数据格式**（vertical data format）。

e.g.

项集	TID - 集	项集	TID - 集
I1	{T100，T400，T500，T700，T800，T900}	I4	{T200，T400}
I2	{T100，T200，T300，T400，T600，T800，T900}	I5	{T100，T800}
I3	{T300，T500，T600，T700，T800，T900}		

项集	*TID* - 集	项集	*TID* - 集
{I1，I2}	{T100，T400，T800，T900}	{I2，I3}	{T300，T600，T800，T900}
{I1，I3}	{T500，T700，T800，T900}	{I2，I4}	{T200，T400}
{I1，I4}	{T400}	{I2，I5}	{T100，T800}
{I1，I5}	{T100，T800}	{I3，I5}	{T800}

项集	*TID* - 集
{I1，I2，I3}	{T800，T900}
{I1，I2，I5}	{T100，T800}

挖掘闭模式和极大模式

项合并：如果包含频繁项集 *X* 的每个事务都包含项集 *Y*，但不包含 *Y* 的任何真超集，则 $X \cup Y$ 形成一个闭频繁项集，并且不必再搜索包含 *X* 但不包含 *Y* 的任何项集。

子项集剪枝：如果频繁项集 *X* 是一个已经发现的闭频繁项集 *Y* 的真子集，并且 $support\_count(X) = support\_count(Y)$，则 *X* 和 *X* 在集合枚举树中的所有后代都不可能是闭频繁项集，因此可以剪枝。

项跳过：在深度优先挖掘闭项集时，每一层都有一个与头表和投影数据库相关联的前缀项集 *X*。如果一个局部频繁项 *p* 在不同层的多个头表中都具有相同的支持度，则可以将 *p* 从较高层头表中剪裁掉。

6.3哪些模式是有趣的：模式评估方法

强规则不一定是有趣的

e.g.

例 6.7 一个误导的“强”关联规则。假设我们对分析涉及购买计算机游戏和录像的 AllElectronics 的事务感兴趣。设 *game* 表示包含计算机游戏的事务，而 *video* 表示包含录像的事务。在所分析的 10 000 个事务中，数据显示 6000 个顾客事务包含计算机游戏，7500 个事务包含录像，而 4000 个事务同时包含计算机游戏和录像。假设发现关联规则的数据挖掘程序在该数据上运行，使用最小支持度 30%，最小置信度 60%。将发现下面的关联规则：

$$buys(X, \text{“computer games”}) \Rightarrow buys(X, \text{“videos”})$$
$$[support = 40\%, confidence = 66\%] \tag{6.6}$$

规则（6.6）是强关联规则，因为它的支持度为 $\frac{4000}{10\,000} = 40\%$，置信度为 $\frac{4000}{6000} = 66\%$，分别满足最小支持度和最小置信度阈值。然而，规则（6.6）是误导，因为购买录像的概率是 75%，比 66% 还高。事实上，计算机游戏和录像是负相关的，因为买一种实际上降低了买另一种的可能性。不完全理解这种现象，容易根据规则（6.6）做出不明智的商务决定。 ■

例 6.7 也表明规则 $A \Rightarrow B$ 的置信度有一定的欺骗性。它并不度量 A 和 B 之间相关和蕴涵的实际强度（或缺乏强度）。因此，寻求支持度 - 置信度框架的替代，对挖掘有趣的数据联系可能是有用的。

从关联分析到相关分析

$A \Rightarrow B [support, confidence, correlation]$

提升度（lift）是一种简单的相关性度量，定义如下。项集 A 的出现**独立**于项集 B 的出现，如果 $P(A \cup B) = P(A)P(B)$；否则，作为事件，项集 A 和 B 是**依赖的**（dependent）和**相关的**（correlated）。这个定义容易推广到两个以上的项集。A 和 B 出现之间的**提升度**可以通过计算下式得到

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)} \tag{6.8}$$

如果（6.8）式的值小于 1，则 A 的出现与 B 的出现是负相关的，意味一个出线可能导致另一个不出现。如果结果值大于 1，则 A 和 B 是正相关的，意味每一个的出现都蕴涵另一个的出现。如果结果值等于 1，则 A 和 B 是独立的，它们之间没有相关性。

（6.8）式等价于 $P(B|A)/P(B)$ 或 $conf(A \Rightarrow B)/sup(B)$，也称关联（或相关）规则 $A \Rightarrow B$ 的提升度。换言之，它评估一个的出现“提升”另一个出现的程度。例如，如果 A 对应于计算机游戏的销售，B 对应于录像的销售，则给定当前行情，游戏的销售把录像销售的可能性增加或“提升”了一个（6.8）式返回值的因子。

模式评估度量比较

给定两个项集 A 和 B，A 和 B 的**全置信度**（all_confidence）定义为：

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\} \tag{6.9}$$

其中，$max\{sup(A), sup(B)\}$ 是 A 和 B 的最大支持度。因此，$all\_conf(A, B)$ 又称两个与 A 和 B 相关的关联规则“$A \Rightarrow B$”和“$B \Rightarrow A$”的最小置信度。

给定两个项集 A 和 B，A 和 B 的**最大置信度**（max_confidence）定义为：

$$max\_conf(A,B) = max\{P(A|B), P(B|A)\} \tag{6.10}$$

max_conf 是两个关联规则“$A \Rightarrow B$”和“$B \Rightarrow A$”的最大置信度。

给定两个项集 A 和 B，A 和 B 的 **Kulczynski**（**Kulc**）度量定义为：

$$Kulc(A,B) = \frac{1}{2}(P(A|B) + P(B|A)) \tag{6.11}$$

该度量是波兰数学家 S. Kulczynski 于 1927 年提出的。它可以看做两个置信度的平均值。更确切地说，它是两个条件概率（给定项集 A，项集 B 的概率；给定项集 B，项集 A 的概率）的平均值。

最后，给定两个项集 A 和 B，A 和 B 的**余弦**度量定义为：

$$cosine(A,B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} = \sqrt{P(A|B) \times P(B|A)} \tag{6.12}$$

余弦度量可以看做调和提升度度量：两个公式类似，不同之处在于余弦对 A 和 B 的概率的乘积取平方根。然而，这是一个重要区别，因为通过取平方根，余弦值仅受 A、B 和 $A \cup B$ 的支持度的影响，而不受事务总个数的影响。

上面介绍的 4 种度量都具有如下性质。度量值仅受 A、B 和 $A \cup B$ 的支持度的影响，更准确地说，仅受条件概率 $P(A|B)$ 和 $P(B|A)$ 的影响，而不受事务总个数的影响。另一个共同性质是，每个度量值都遍取 0 ~ 1，并且值越大，A 和 B 的联系越紧密。

总结：6种模式评估度量

表 6.9　使用不同数据集的相依表比较 6 种模式评估度量

数据集	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	χ^2	提升度	全置信度	最大置信度	$Kluc$	余弦
D_1	10 000	1000	1000	100 000	90 557	9.26	0.91	0.91	0.91	0.91
D_2	10 000	1000	1000	100	0	1	0.91	0.91	0.91	0.91
D_3	100	1000	1000	100 000	670	8.44	0.09	0.09	0.09	0.09
D_4	1000	1000	1000	100 000	24 740	25.75	0.50	0.50	0.50	0.50
D_5	1000	100	10 000	100 000	8173	9.18	0.09	0.91	0.50	0.29
D_6	1000	10	100 000	100 000	965	1.97	0.01	0.99	0.50	0.10

零事务是不包含任何考察项集的事务。

“对于指示有趣的模式联系，全置信度、最大置信度、Kulczynski 和余弦哪个最好？”

为了回答该问题，引进**不平衡比**（Imbalance Ratio，IR），评估规则蕴含式中两个项集 A 和 B 的不平衡程度。它定义为：

$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)} \tag{6.13}$$

其中，分子是项集 A 和 B 的支持度之差的绝对值，而分母是包含项集 A 或 B 的事务数。如果 A 和 B 的两个方向的蕴含相同，则 $IR(A, B)$ 为 0；否则，两者之差越大，不平衡比就越大。这个比率独立于零事务的个数，也独立于事务的总数。