BIG DATA

第3章 关联分析

关联分析用于分析对象之间的关联性、相关性。关联规则是关联分析的技术之一。关联规则概念的提出最早源自超市购物篮分析(market basket analysis)。它用于分析顾客一次购买的商品之间的关联性,即哪些商品经常被一起购买,这从一定程度上反映了顾客的购买行为模式。这种模式可以被用于辅助零售运营商进行运营管理,例如,货品的布局安排、促销策略的制定、进货管理等方面。然而它的应用远不局限于购物篮分析,还可以用于发现许多其他应用领域数据之间的关联并进而研究其相关性,也可以用于分类、聚类等其他挖掘任务中。

本章将首先介绍关联规则相关的基本概念,如频繁项集、关联规则、闭合项集等,然后介绍几种经典的关联规则和频繁项集的挖掘算法,最后介绍关联分析的兴趣度度量。

3.1 频繁模式与关联规则

以购物篮分析为例,关联规则用于分析顾客一次购买的商品的关联性。为了进行此类分析,需要记录顾客每次购买的所有商品,记录该信息的数据库称为**交易数据库**(transaction database),如表3.1所示。表中第一列记录的是交易号(transaction identifier,TID),对应于购物小票的编号,具有唯一标识一个顾客一次购物行为的作用。第二列记录的是一个顾客一次购物所包含的所有商品,用集合变量 t_i 表示,其中 i 是交易号。表中的一行对应一次交易(transaction)。实际购物小票中通常一种商品在一行中表示,以便记录购买数量、价格和折扣等详细信息。在关联规则的分析中暂且忽略这些信息,并且为了表示的简洁性忽略了每个商品的具体品牌和规格等信息。

表3.1 交易数据库示例

交易号(TID)	商品(item)	交易号(TID)	商品(item)
1	beer, diaper, nuts	4	beer, cheese, diaper, nuts
2	beer, biscuit, diaper	5	beer, butter, cheese, nuts
3	bread, butter, cheese		

在关联规则挖掘中,被研究的对象称为项(item)。例如在购物篮分析应用背景下,顾客购买的每个商品称为一个项。所有项的集合由 I 表示,$I=\{i_1,i_2,\cdots,i_n\}$。例如,在表3.1中,$I=\{\text{beer},\text{biscuit},\text{bread},\text{butter},\text{cheese},\text{diaper},\text{nuts}\}$。由此,每个交易 t_i 对应的项的集合是 I 的子集,即 $t_i\subseteq I$。例如,$t_1=\{\text{beer},\text{diaper},\text{nuts}\}\subseteq I$。交易数据库 D 则是交易的集合,即 $D=\{t_1,t_2,\cdots,t_m\}$。I 的任何一个子集称为**项集**(itemset)。一个项集 X 包含的项的个数称为该项集的**长度**。包含 k 个项的项集称为 k **项集**。一个项集 X 在数据库 D 中出现

的次数，称为频数，记为 count(X)，等于包含该项集的交易的个数。例如，若 X={beer，diaper}，则 count(X)=3，因为它被 t_1、t_2 和 t_4 所包含。一个项集 X 的**支持度**(support)记为 support(X)，由公式(3.1)定义。

$$\text{support}(X)=\frac{\text{count}(X)}{|D|}\times 100\% \tag{3.1}$$

其中，$|D|$代表 D 中交易的个数。若 X={beer，diaper}，则 support(X)=3/5=60%。

一个项集 X 的支持度如果大于用户给定的一个最小支持度(minimum support，minsup)阈值，则 X 被称为**频繁项集(或频繁模式)**，或称 X **是频繁的**。

给定两个项集 X 和 Y，关联规则是形如 $X\rightarrow Y$ 的蕴含式，其中 $X\subseteq I$ 称为规则的前件，$Y\subseteq I$ 称为规则的后件，且 $X\cap Y=\varnothing$。一个规则 $X\rightarrow Y$ 在数据库 D 中的**支持度**，记为 support($X\rightarrow Y$)，定义为项集 $X\cup Y$ 的支持度，即 support($X\rightarrow Y$)=support($X\cup Y$)。它的**置信度**记为 confidence($X\rightarrow Y$)，定义为项集 $X\cup Y$ 的支持度除以项集 X 的支持度，即如公式(3.2)定义。

$$\text{confidence}(X\rightarrow Y)=\frac{\text{support}(X\rightarrow Y)}{\text{support}(X)}\times 100\% \tag{3.2}$$

一个规则 $X\rightarrow Y$ 如果同时满足 support($X\rightarrow Y$)$\geqslant$minsup 和 confidence($X\rightarrow Y$)$\geqslant$minconf，则称该规则在数据库 D 中成立，其中 minsup 和 minconf 分别是用户给定的最小支持度和最小置信度的阈值。例如，给定 minsup=40%和 minconf=60%，则规则{beer，diaper}→nuts(为了表示的简洁性，将包含一个元素的集合的大括号省略，以下同)在表 3.1 所示的数据库中成立，因为 support({beer，diaper}→nuts)=40%，confidence({beer，diaper}→nuts)=66.7%。

在实际应用中，给定 minsup 和 minconf 两个阈值后，一个数据库中存在的频繁项集的数目可能巨大，这是因为频繁项集存在如下性质。

性质 3.1 给定最小支持度阈值 minsup，一个频繁项集的所有非空子集都是频繁的。

例如，若 minsup=40%，则项集{beer，diaper，nuts}是频繁的，因为它出现在 t_1 和 t_4 两个交易中，则这两个交易必然同时包含了该项集的所有非空子集，如 beer、diaper、nuts、{beer，diaper}、{beer，nuts}和{diaper，nuts}，共 6 个项集。那么这些项集至少出现在这两个交易中，因此它们都是频繁的。根据该性质，可以知道“如果一个项集是不频繁的，则其所有超集都是不频繁的”也是成立的。

由于该性质的存在，如果一个数据库中可以发现的频繁项集的最大长度为 20，则最终输出的频繁项集个数至少为($2^{20}-1$)个，其中($2^{20}-2$)个是该项集的子集。众多的项集给后续的结果分析与评价带来问题，因此，如何减少输出的项集个数，同时又不损失任何信息是需要解决的问题，为此，Pasquier 等人首次提出了闭合频繁项集(closed frequent itemset)的概念。

一个频繁项集 X 被称为**闭合频繁项集**当且仅当不存在任一个项集 Y 满足 $X\subset Y$ 且 support(Y)=support(X)。闭合频繁项集 X 被称为是闭合的。

例如，在表 3.1 中，假设 minsup=40%，项集 diaper 是频繁的，但不是闭合的，因为存在项集{beer，diaper}是 diaper 的超集且具有和它相同的支持度 60%。而{beer，diaper}是闭合频繁项集，因为不存在一个更长的超集具有与它相同的支持度。利用闭合频繁项集，输

出的最终满足条件的项集个数将大大降低。举个极端的例子，如果一个长度为 20 的频繁项集 X 的所有非空子集的支持度都与该项集相同，则原来结果中的$(2^{20}-1)$个项集都由该项集 X 所涵盖了。如果只有部分子集具有与 X 相同的支持度，则这些项集不必输出的同时，又可以通过 X 推导出它们的存在，所有未出现在结果中的那些 X 的子集一定是频繁的且具有与 X 相同的支持度。因此，闭合频繁项集的集合是频繁项集集合的一种信息无损的压缩。

3.2 频繁项集的典型挖掘方法

关联规则的挖掘一般分为两个步骤：第一步发现所有的频繁项集；第二步从频繁项集中发现关联规则。本节重点介绍频繁项集的生成方法，3.3 节介绍关联规则的生成方法。

关联规则自 1993 年提出以来，至今已有非常多的相关研究。典型的挖掘算法包括 Agrawal 和 Srikant 1994 年提出的 Apriori 算法以及 J. Han、J. Pei 等提出的 FP-Growth 算法和 Zaki 提出的 Eclat 算法等。挖掘闭合频繁项集的典型算法包括 CLOSET＋以及 CHARM 算法等。在本节中将对上面提到的 Apriori、FP-Growth 两个算法进行介绍。

3.2.1 逐层发现算法 Apriori

逐层发现算法 Apriori 发现频繁项集的过程是按照项集的长度由小到大逐级进行的，即首先发现频繁 1 项集，然后是频繁 2 项集，……，最后是频繁 N 项集。在发现过程中为了减少需要检查的项集的个数，提高发现效率，该算法充分利用了 3.1 节中给出的性质 3.1，即一个项集如果是不频繁的，则其所有超集一定也是不频繁的。

Apriori 算法的主要步骤如下所示。

算法 3.1：Apriori
输入：交易数据库 D，最小支持度阈值 minsup
输出：D 中的所有频繁项集的集合 F
主要步骤：

(1) Find all of frequent items from D and save them in F_k, $k=1$;
(2) if F_k is not empty then begin
(3) C_{k+1}=gen_candidate(F_k);
(4) for each transaction t in D begin
(5) for any candidate $c \in C_{k+1}$
(6) if itemset c occurs in t then
(7) c.count++;
(8) end for
(9) $F_{k+1}=\{c \in C_{k+1} \mid \text{support}(c) \geqslant \text{minsup}\}$;
(10) $k=k+1$;
(11) end if
(12) return $F=\bigcup_k F_k$;