北京大学信息管理系

《数据挖掘导论》讲义

第三章 关联规则挖掘

北京大学信息管理系

2016 年秋

目录

第三章 关联规则挖掘

现实生活中，许多企业在运营的过程中积累了大量的数据。例如，某超市的收银台每天都能收集大量的顾客购物数据。假设你是某超市的销售经理，正在与一位刚在超市购买了微波炉和厨具的顾客交谈，你应该向他推荐什么产品？这时候如果知道其他同时购买微波炉和厨具的顾客又频繁购买什么产品，将会对你的推荐起很大帮助。

本章主要介绍一种关联分析（association analysis）的方法，用于发现隐藏在大型数据集中有意义的联系。

3.1 相关概念

本节通过一个超市购物篮的迷你数据集来解释关联规则挖掘的基本概念。

3.1.1 购物篮分析

“啤酒与尿布”的故事是关联规则挖掘的一个经典案例。超市拥有大量的商品，如牛奶、面包等，顾客将所购买的商品放入到自己的购物篮中，即为购物篮事务（market basket transaction）。

例 3.1 购物篮分析。表 3-1 给出了一个超市购物篮的例子，其中每一行对应一个事务，包含一个唯一标识 TID 和某顾客购买商品的集合。超市管理者可通过发现顾客放入购物篮中的不同商品之间的联系，分析顾客的购买习惯（例如，哪些物品经常被顾客购买；同一次购买中，哪些商品经常会被一起购买；一般用户的购买过程中是否存在一定的购买时间序列等），以实现利润最大化。

表 3-1 购物篮事务的例子

TID	项集
1	{面包，牛奶}
2	{面包，尿布，啤酒，鸡蛋}
3	{牛奶，尿布，啤酒，可乐}
4	{面包，牛奶，尿布，啤酒}
5	{面包，牛奶，尿布，可乐}

购物篮数据还可以用二元形式表示。表 3-1 对应的购物篮事务可表示为：

表 3-2 购物篮事务的二元表示

TID	面包	牛奶	尿布	啤酒	鸡蛋	可乐
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

其中，每行对应一个事务，每列对应一个项。项用二元变量表示，如果项在事务中出现，则它的主值为 1，否则为 0。

从以上购物篮数据中可以看出许多购买尿布的顾客也购买啤酒，因此可提取出如下规则：{尿布}→{啤酒}。该规则表明尿布和啤酒的销售之间存在很强的联系，超市管理者可以使用这类规则于营销规划、广告策划或超市布局中。例如用该规则指导超市布局，一种策略是将啤酒和尿布相邻摆放，以便进一步刺激二者的同时销售；另一种策略是把啤酒和尿布摆放在

超市的两端，诱发买这两种商品的顾客一路挑选其它商品。

3.1.2 支持度、置信度与关联规则

设 $I=\{i_1, i_2, \ldots, i_n\}$是购物篮数据中所有项的集合，$T=\{t_1, t_2, \ldots, t_m\}$是所有事务的集合。在例 3.1 中，I={面包，牛奶，尿布，啤酒，鸡蛋，可乐}，其中{面包}、{牛奶}等为**项**（item）；$T=\{t_1, t_2, t_3, t_4, t_5\}$。

事务 t（transaction）是项的集合（$t \subseteq I$），每一个事务都对应一个唯一的标识（如交易号，记作 TID）；事务 t 的宽度定义为事务 t 中包含的项的个数；如果一个项集包含 k 个项，则称其为 **k 项集**，如例 3.1 中 t_1 事务的项集为{面包，牛奶}，是一个 2 项集。

若项集 X 是 I 中一些项的集合，且 X 是事务 t 的子集，则称事务 t 包含项集 X。例 3.1 中，t_2 事务包含项集{啤酒，尿布}，但不包含项集{面包，牛奶}。

项集的一个重要性质是它的**支持度计数**，为某项集的出现频率，即包含该项集的事务个数。如例 3.1 中，{面包，牛奶}项集的支持度计数为 3，因为该项集同时出现在了 t_1，t_4 和 t_5 三个事务中。

关联规则（association rule）是所有形如 X→Y 的蕴涵式，其中 X 和 Y 是不相交的项集。关联规则的有趣性可以用支持度与置信度来度量，**支持度**（support）用来描述给定项集的频繁程度，**置信度**（confidence）用来确定 Y 在包含 X 的事务中出现的频繁程度。二者的度量公式如下。

- 支持度

$$\text{support}(X \Rightarrow Y) = \frac{\text{support\_count}(X \cup Y)}{m}$$

其中 support_count（$X \cup Y$）为项集{X, Y}的支持度计数，即包含项 X 和 Y 的交易数；m 为总交易数。

- 置信度

$$\text{confidence}(X \Rightarrow Y) = \frac{\text{support\_count}(X \cup Y)}{\text{support\_count}(X)}$$

support_count（X）为购买商品 X 的交易数。

支持度可反映规则的有用性，因为低支持度的规则可能只是偶尔出现。从超市管理者的角度去看，顾客很少同时购买的商品可能对促销无益，因此支持度通常用来删去那些无意义的规则。置信度可反映规则的确定性，如果顾客在购买 X 时一定会购买 B，那么就可以将这两种商品进行捆绑销售。

关联规则的基本表现形式如下：

前提条件→结论[支持度，置信度]

以上节提到的购物篮数据为例，可得到如下关联规则：

规则一：*{尿布}→{啤酒} [60%, 75%]*

规则二：*{牛奶，尿布}→{啤酒} [40%, 67%]*

规则一指 60%的人同时购买了尿布和啤酒，买了尿布的人中有 75%的人同时购买了啤酒；规则二指 40%的人同时购买了牛奶、尿布和啤酒，同时买了牛奶和尿布的人中有 67%的人还购买了啤酒。

3.1.3 关联规则挖掘

关联规则挖掘就是发现大量数据中项集之间有趣的关联。一般情况下，有趣的关联规则是指满足最小支持度阈值和最小置信度阈值的关联规则。这些阈值可以由用户或某领域专家来设定。

挖掘关联规则的一个原始方法是计算每个可能规则的支持度与置信度，但该方法的代价太高，达到指数级；具体地说，从包含 d 个项的数据集中提取的可能的规则总数为 $R=3^d-2^{(d+1)}+1$。即使对于例 3.1 中的小数据集，这种方法也需要计算 602 条规则的支持度和置信度。如果设定最小支持度阈值为 20%，最小置信度为 50%，则 80%以上的规则都将被丢弃，使得大部分计算是无用的开销。为了避免进行不必要的计算，应事先对规则剪枝。

提高关联规则挖掘算法性能的第一步是拆分支持度和置信度，由二者的度量公式可看出，规则 X→Y 的支持度仅依赖于其对应项集 X∪Y 的支持度。例如，下面的规则有相同的支持度，因为它们都涉及到同一个项集{啤酒，尿布，牛奶}。如果该项集是非频繁的，则可以立即剪掉这 6 个候选规则，而不必计算它们的置信度。

{啤酒，尿布}→{牛奶}，{啤酒，牛奶}→{尿布}

{尿布，牛奶}→{啤酒}，{啤酒}→{尿布，牛奶}

{牛奶}→{啤酒，尿布}，{尿布}→{啤酒，牛奶}

因此，大多数关联规则挖掘算法采用的是一种两步的策略：

（1）**频繁项集的产生**：其目标是发现满足最小支持度阈值的所有项集，这些项集称作频繁项集（frequent itemset）。

（2）**规则的产生**：其目标是从上步所产生的频繁项集中提取满足最小置信度的规则。

关联规则挖掘所花费的时间主要是在生成频繁项集上，因为找出的频繁项集往往不会很多，利用频繁项集生成规则也就不会花太多的时间，而生成频繁项集需要测试很多的备选项集，如果不加优化，所需的时间是 $O(2^n)$。

3.2 频繁项集的产生

3.2.1 格结构

图 3-1 为项集 I={a, b, c, d, e}的格结构（lattice structure），用来枚举所有可能的项集。

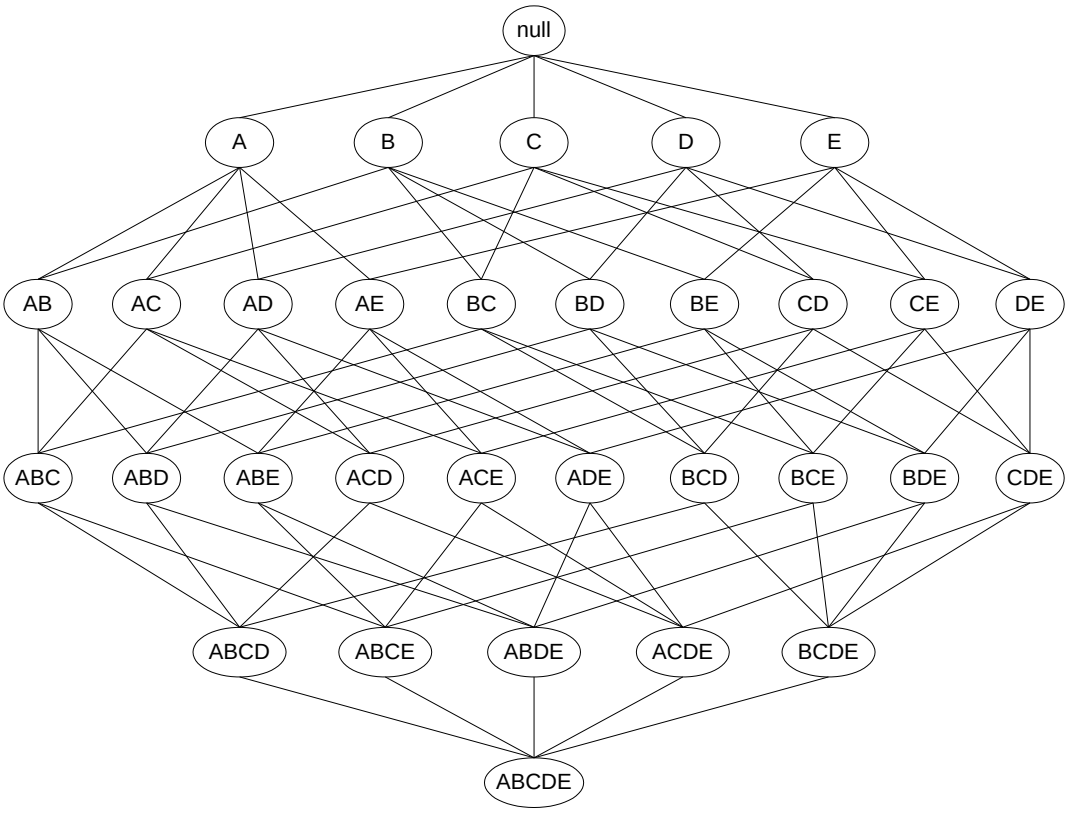

图 3-1 项集的格结构

一般来说，一个包含 k 个项的数据集可能产生 2^k-1 个频繁项集（不包括空集）。发现频繁项集的一种原始方法是确定格结构中每个候选项集的支持度计数。为了完成这一任务，必须将每个候选项集与每个事务进行比较，如果候选项集包含在事务中，则候选项集的支持度计数增加。

有两种方法可以降低产生频繁项集的计算复杂度：（1）减少候选项集的数目，如 apriori 原理，是一种不用计算支持度值而删除某些候选项集的有效方法；（2）减少比较次数，可使用更高级的数据结构。

3.2.2 Apriori 性质

为了减少频繁项集的生成时间，应尽早消除一些完全不可能是频繁项集的集合，Apriori 的性质（又称先验原理）可起到该作用。

先验原理 1：任何频繁项集的非空子集均为频繁项集。例如，假设{A, B, C}是频繁项集，则{A, B}、{A, C}、{B, C}均为频繁项集。

先验原理 2：如果一个集合不是频繁项集，则它的所有超集都不是频繁项集。例如，假设集合{A,B}不是频繁项集，即 A 和 B 同时出现的次数小于最小支持度阈值，则它的任何超集如{A,B,C}出现的次数必定小于最小支持度阈值，因此其超集必定也不是频繁项集（图 3-2）。

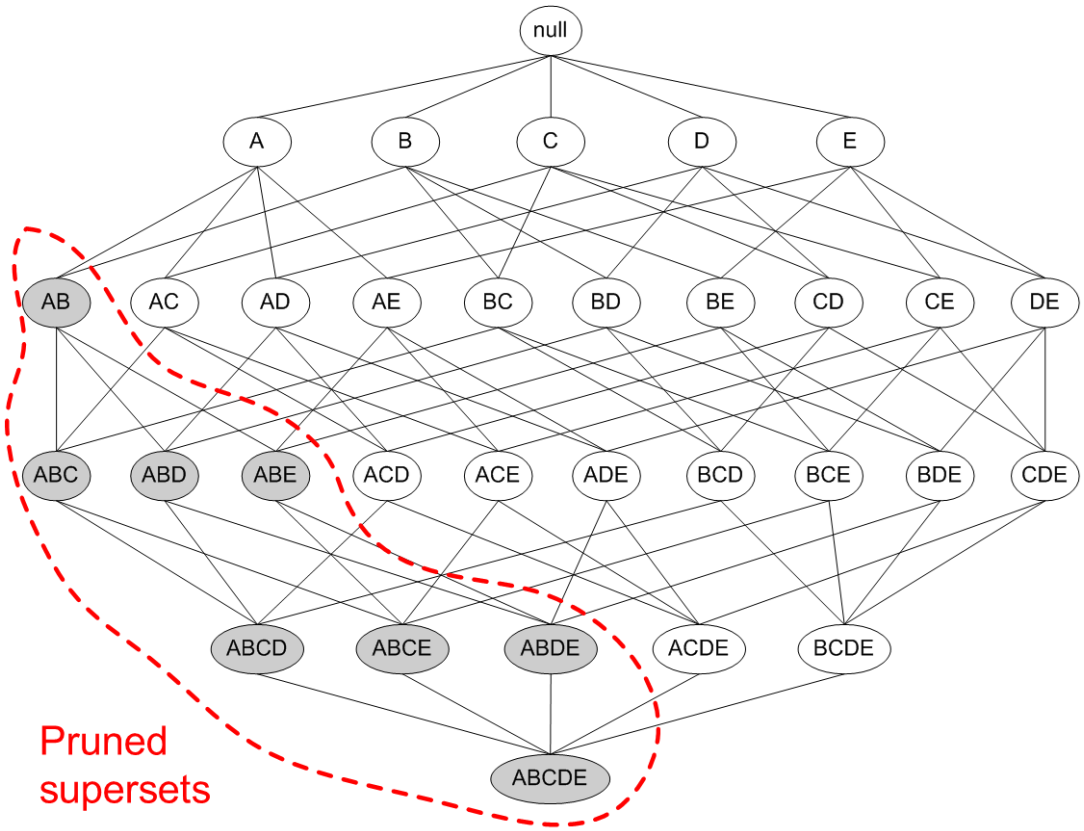

图 3-2 基于支持度的剪枝

3.2.2 Apriori 算法

Apriori 算法是 Agrawal 和 R.Srikant 于 1994 年提出的，也是第一个被提出的关联规则挖掘算法。该算法使用一种逐层搜索的迭代方法，用 k 项集探求（k+1）项集；并合理运用先验知识，基于支持度的剪枝技术系统地控制候选项集的指数增长。

首先，通过扫描数据库，得到每个项的支持度计数，并找出满足最小支持度阈值的频繁

1-项集，该集合记为 L_1；然后用 L_1 找出频繁 2-项集的集合 L_2；使用 L_2 找出 L_3，如此继续下去，直到找到最大频繁项集。

该方法主要由连接和剪枝两步构成，连接指从频繁 k-项集产生候选的频繁(k+1)-项集；剪枝指删除候选频繁（k+1）-项集中某一 k-项子集是非频繁的项集，具体步骤如图 3-3。

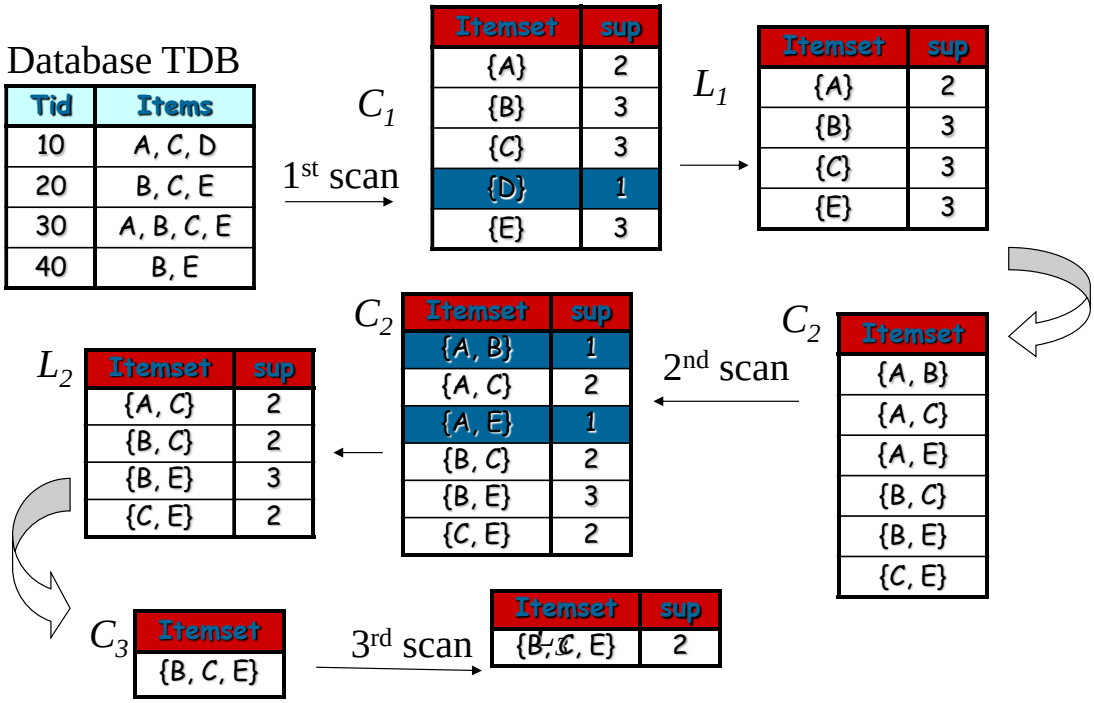

图 3-3 Apriori 算法的具体步骤

数据库中存有 4 个事务，所有项的集合为{A,B,C,D,E}。假设最小支持度阈值为 50%，由于事务个数为 4，因此需要某项集的支持度计数大于等于 2，才可称为频繁项集。第一次扫描时，计算出所有 1-项集的支持度计数，发现项集{D}的支持度计数为 1，小于 2，因此将其删除（剪枝）。由频繁 1-项集通过连接生成 2-项集，即 C2，重新计算 C2 中各项集的支持度计数，发现{A,B}和{A,E}的支持度计数为 1，小于 2，将其删除，得到 L2；在对 L2 进行连接时，本可得到{A,B,C}，{A,B,E}，{A,C,E}，{B,C,E}四个 3-项集，但由于{A,B}和{A,E}不是频繁项集，因此包含{A,B}的{A,B,C}，{A,B,E}以及包含{A,E}的{A,C,E}被剪枝，得到 C3；通过扫描数据库，得到{B,C,E}的支持度计数为 2，支持度为 50%，由于无法再连接，因此{B,C,E}即为该数据的最大频繁项集。

3.3 规则的产生

3.3.1 由频繁项集产生关联规则

从数据库的事务当中找出频繁项集后，即可直接由它们生成强关联规则（同时满足最小支持度阈值和最小置信度阈值的关联规则）。忽略包含空集的规则（⌀→Y 或 Y→⌀），每个频繁 k-项集能够产生多达 2^k-2 个关联规则。由于频繁项集已经满足支持度阈值，因此只要将项集 Y 划分为两个非空的子集 X 和 Y-X，使得 X→Y-X 满足置信度阈值即可。

例 3.2 由频繁项集产生关联规则。设 U={B,C,E}是频繁项集，则可以由 X 产生 6 个候选关联规则：{B}→{C,E}，{C}→{B,E}，{E}→{B,C}，{B,C}→{E}，{B,E}→{C}和{C,E}→{B}。由于它们的支持度都等于 X 的支持度，因此这些规则一定满足支持度阈值。由于这些项集的支持度计数已经在频繁项集的产生时得到，因此在计算以上关联规则的置信度时并不需要重新扫描数据库。