一种挖掘频繁闭项集的改进算法

An Improved Algorithm for Mining Frequent Closed Itemsets

吴春旭　陈家耀　刘博文 （中国科学技术大学 管理学院 安徽 合肥 230052）

摘　要: 频繁闭项集的挖掘是近年来频繁项集挖掘研究的热点。本文引入了共生项集的概念,从一个新的角度看待频繁闭项集的挖掘问题。利用共生项集的性质,本文提出了一种新的无需遍历结果集的闭合性检查方法,并在此基础上对 CLOSET 算法进行改进,实验证明,取得了良好的效果。

关键词: 数据挖掘 频繁闭项集 闭合性检查 共生项集 CLOSET

1　引言

作为关联规则挖掘的关键步骤,频繁项集的挖掘在许多数据挖掘任务中具有重要的作用。然而,当支持度阈值较小时,频繁项集的数量往往大得惊人,从而严重影响了挖掘的效率。

为了解决这一问题,Pasquier 等人提出了频繁闭项集的概念[1]。频繁闭项集可以说是频繁项集的一种无损压缩形式,它在数量上往往比频繁项集小几个数量级,但又同时保留了频繁项集的所有有用信息。因此,挖掘频繁闭项集是一种比挖掘频繁项集更有效,更具操作性的数据挖掘方式。对频繁闭项集的挖掘算法也成为近年来数据挖掘领域的研究热点之一,许多有效的算法被提出来,如 CLOSET[2]、CHARM[3]、CLOSET+[4]等。

这些算法大多数都是直接从频繁闭项集的性质出发,对传统频繁项集挖掘算法加入剪枝策略及闭合性检查方法后得到。在进行闭合性检查时,大部分算法都是基于超集检查或子集检查的思想,即每挖掘出一个新的频繁项集时,要遍历已得到的结果集,检查新项集是否是已有项集的超集或子集,且支持度相同,若是,则将非闭合项集丢弃。这样的方法不仅需要将结果集一直保存在内存中,而且每次检查都需要遍历结果集,计算支持度。当支持度阈值较小,结果集较大时,内存和 CPU 的开销将会变得很大。

针对上述问题,本文通过归纳概括共生项集的概念,总结了共生项集的性质与作用,提出了一种基于共生项集的无需遍历结果集的闭合性检查方法,并应用该方法改进 CLOSET 算法,取得较好的结果。

2　相关概念

2.1　频繁闭项集

给定项集 $I = \{i_1, i_2, \cdots i_m\}$。一个事务可表示为 $T = <tid, X>, X \subseteq I$。对于项集 Y,若 $Y \subseteq X$,则称事务 T 支持 Y。一个事务数据库 TDB 中支持 Y 的事务数目叫做 Y 的支持数,记作 sup_count(Y)。如果给定一个支持数阈值 min_sup,支持数大于该阈值的项集称为频繁项集。

而频繁闭项集的定义如下：

定义 1(频繁闭项集) 设 X 是一个频繁项集,若不存在任何真超集 Y,$X \subset Y$,使得 sup_count(Y) = sup_count(X),则称 X 是频繁闭项集[3]。

为了有效挖掘频繁闭项集,一些剪枝技术和搜索策略,如项目合并、子集剪枝等[2],被提了出来,并得到广泛应用。

所谓项目合并,就是:若 X 是一个频繁项集,如果包含 X 的每个事务均包含 Y 但不包含任何 Y 的超集,那么 $Y \cup X$ 可以合并成一个频繁闭合项目集,而且没有必要去挖掘任何包含 X 但不包含 Y 的项目集。

所谓子集剪枝,就是:假设 X 是当前考虑下的频繁项集,如果 X 是一个已经发现的频繁闭项集的子集,并且两者支持度相同,那么在 FP－树中 X 和所有 X 的子孙均不可能是频繁闭项集,可以将其剪枝掉。

2.2　共生项集

在研究中我们发现,有一类项集与频繁闭项集挖

掘关系密切,以上的剪枝技术实际上都或多或少地利用了这种项集的一些性质,为了便于对这种项集进行研究,我们将其归纳定义如下:

定义 2(共生项集) 设 X,Y 是项集,$X \subseteq I, Y \subseteq I$。若 $X \cap Y = \varnothing$,且 $\forall T = <tid, I_t> \in TDB, X \subset I_t \Rightarrow Y \subset I_t$,则称 Y 是 X 的共生项集。

简单来说,X 的共生项集就是在某个事务集中,每个支持 X 的事务均支持的项集。如在表 1 的事物数据库中,任何出现了 g 的事务同时都出现了 c,e,所以 {c,e} 就是 {g} 的共生项集。

表 1　事务数据库 TDB

tid	ItemSet
T1	c d e g
T2	b c e g
T3	a c f
T4	a e f
T5	b g h

3　共生项集的性质与作用

3.1　共生项集的性质

经过研究,我们发现共生项集具有以下几个有趣的性质:

引理 设 X 是一个项集,Y 是 X 的共生项集,∀ 频繁项集 Z,若 $X \subseteq Z$,则 $Z \cup Y$ 也是频繁项集,且 sup_count(Z) = sup_count($Z \cup Y$)。

证明:∵ Y 是 X 的共生项集,

∴ $\forall T = <tid, I_t> \in TDB, Z \subset I_t \Rightarrow X \subset I_t \Rightarrow Y \subset I_t$

即 $\forall T = <tid, I_t> \in TDB, Z \subset I_t \Rightarrow (Z \cup Y) \subset I_t$。

∴ sup_count($Z \cup Y$) sup_count(Z)。

又∵ $Z \cup Y$ 是 Z 的超集,

∴ sup_count(Z) sup_count($Z \cup Y$)。

∴ sup_count(Z) = sup_count($Z \cup Y$)。

推论 1 设 X 是一个项集,Y 是 X 的非空共生项集,∀ 频繁闭项集 Z,若 $X \subseteq Z$,则有 $Y \subseteq Z$。

证明:反证法。设有频繁闭项集 Z,$X \subseteq Z, Y \not\subset Z$。

由上面引理可知 sup_count(Z) = sup_count($Z \cup Y$),而由于 Y 非空,故 $Z \cup Y$ 是 Z 的真超集。这与频繁闭项集的定义矛盾。命题得证。

推论 2 设 X 是频繁项集,Y 是 X 的所有共生项集的并集,则 $Y = \varnothing \Leftrightarrow X$ 是频繁闭项集。

证明 先证⇒。反证法。假设 X 不是频繁闭项集,则存在 X 的真超集 Z,使得 sup_count(Z) = sup_count(X),即 $\forall T = <tid, I_t> \in TDB, X \subset I_t \Rightarrow (Z - X) \subset I_t$

∴ (Z − X) 是 X 的共生项集,且 $(Z - X) \neq \varnothing$,

这与 $Y = \varnothing$ 矛盾。

再证⇐。反证法。假设 $Y \neq \varnothing$,由上面的引理可知,sup_count(X) = sup_count($X \cup Y$),$X \cup Y$ 是 X 的真超集,这与 X 是频繁闭项集矛盾。

命题得证。

观察以上定理,不少剪枝技术都可以从共生项集的角度,对其原理进行阐释。例如“项目合并”技术,就是利用推论 2 的性质得出“$X \cup Y$ 是一个频繁闭合项目集”,再利用推论 1,得出“没有必要去挖掘任何包含 X 但不包含 Y 的项集”。而“子集剪枝”技术则是利用了推论 2 得出“X 和所有 X 的子孙不可能是频繁闭合项目集”的结论,从而实现剪枝。

因此,可以说,这些剪枝技术实际上都是利用共生项集的性质进行的。事实上,利用推论 2,不仅能进行剪枝,还能进行闭合性判断。

3.2　基于共生项集的闭合性检查

CLOSET 算法及大多数频繁闭项集挖掘算法所使用的闭合性检查方法是“子集检查”的方法。这种方法的本质是直接利用频繁闭项集的定义进行检查的,因此,需要在内存中保存并不断遍历结果集。

从推论 2 我们可以看出,挖掘频繁闭项集实质上就是挖掘共生项集为空的频繁项集。因此,我们完全可以从共生项集的角度进行闭合性判断。

基于共生项集的闭合性判断的基本思想是:在得到一个新的频繁项集时,检查该项集的共生项集是否为空,若是为空则是频繁闭项集,反之则不是。

为了利用共生项集,我们需要收集共生项集的信息。对于不同的数据结构,应有不同的收集方法。CLOSET 算法挖掘频繁闭项集的成熟而有效的算法。下面我们提出一种应用共生项集进行闭合性检查,以对 CLOSET 进行改进的算法。

4　改进 CLOSET 算法

4.1　数据结构的扩展

为了收集存储共生项集的信息,我们对 FP − 树结

点的结构进行扩展：

原来的 FP－树结点的结构是(Item :sup_count)，Item 表示项目，sup_count 表示节点项目在该节点的分支中的支持数。现在我们增加一个存储单元，节点结构扩展为(Item[Local_BCOIS] :sup_count)，其中 Local_BCOIS 表示在该分支所表示的事务集中与节点项目共生的，且在头表中位置在节点项目之后的项目的集合(头表中项目按支持数降序排列)，称为局部后共生项集。

例如，在图 1 所示的 FP－树中，e 节点的 Local_BCOIS 为 {g}，而 c 节点的 Local_BCOIS 为空集。

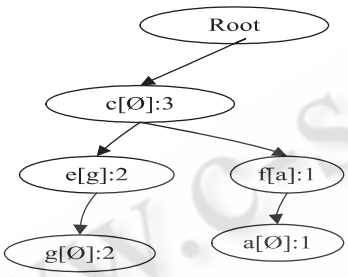

图 1　带 Local_BCOIS 的 FP－树

引入局部后共生项集的原因在于：条件 FP－树的构造是前向性的，也就是说当我们考虑挖掘头表中的某个项目时，其条件数据库中只包含头表中位于该项目之前的项目，而忽略了头表中位于该项目之后的项目。例如图 1 中，当我们挖掘项目 e 时，构造的条件 FP－树将只包含 c，而不会包含 g。因此，我们将共生项集分为两部分，在头表中位于当前挖掘项目之前的叫做前共生项集，之后的称为后共生项集。前共生项集可以通过扫描条件数据库得到。而后共生项集，在引入了节点局部后共生项集结构后，可以通过对每个节点的局部后共生项集扫描归纳后得到。

事实上，由于 CLOSET 算法中引入项目合并技术，前共生项集会被提取出来进行项目合并。因此在进行了项目合并的前提下，我们实际上只需考查后共生项集是否为空，就能判断项目是否闭合。

4.2　挖掘过程

挖掘频繁闭项集的步骤与 CLOSET 算法基本一致，只是一些步骤的实现过程有所不同，我们以表 1 的 TDB 为例(min_sup 为 2)，对其中实现过程不同的步骤作说明如下：

(1) 全局 FP－树的生成。与原算法不同的是，本算法在将事务记录插入 FP－树时，除了要计算节点的支持数，还要对节点的 Local_BCOIS 作初始化。具体是：在插入一个节点时，若该节点不是当前事务记录的最后一个项目，就将其 Local_BCOIS 置为 NULL，否则置为∅。例如，插入第一条记录 {c e g}，将依次在全局 FP－树中插入了三个节点，分别为(c[NULL] :1)，(e[NULL] :1)，(g[∅] :1)。最后生成的全局 FP－树如图 2 所示。

(2) 条件数据库的生成。在 FP－树生成过程中初始化得到的 Local_BCOIS，并不是节点真正的局部后共生项集。因此，在生成条件模式基时，我们要计算节点的 Local_BCOIS，并得到带 Local_BCOIS 的条件模式基。

FP 节点的 Local_BCOIS 的计算方法是这样的：若该节点的 Local_BCOIS 的初始化值为 NULL，则提取每个子节点或该子节点的 Local_BCOIS 中都出现的项目，作为父节点的 Local_BCOIS；否则，提取父节点的 Local_BCOIS 和子节点的 Local_BCOIS 都出现的项目，作为父节点的 Local_BCOIS。

得到节点的 Local_BCOIS 后，我们将其带入该节点生成的条件模式基中。

例如，在图 2 中，当我们考虑挖掘 a 时，我们先计算每个 a 节点的 Local_BCOIS，然后得到带 Local_BCOIS 的条件模式基：(cf [∅] :1)，(ef [∅] :1)。同理，当考虑挖掘 f 时，得到的两条条件模式基是：(c[a] :1)，(e[a] :1)。

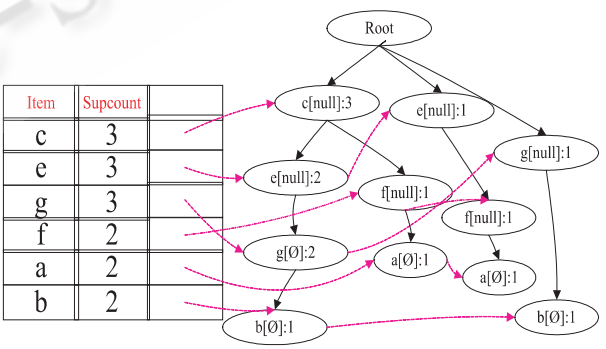

图 2　初始化的全局 FP－树

(3) 闭合性判断。在得到条件模式基后，我们对条件模式基中的 Local_BCOIS 进行扫描，提取所有的 Local_BCOIS 中都出现的项目作为后共生项集。若后共生项集不为空，则说明当前模式不是闭合的，将其丢

弃,并停止其后续挖掘。

例如,对于 a 的条件模式基:(cf [∅] :1),(ef [∅] :1),提取出的后共生项集为∅,于是我们利用项目合并提取前共生项集 {f},得到频繁闭项集(af :2),并递归构造 a 的条件 FP – 树继续挖掘。而对于 f 的两条条件模式基:(c[a] :1),(e[a] :1)中都出现了 a,即后共生项集是 {a},不为空,故(f :2)不是频繁闭合项,将其丢弃,并停止对 f 的挖掘。

(4)条件 FP – 树的生成。生成条件 FP – 树时同样要对节点的 Local_BCOIS 作初始化,但与全局初始化不同的是,为了使共生信息在递归构造条件 FP – 树时不被丢失,初始化时要带入条件模式基的 Local_BCOIS。

具体来说是:将一条条件模式基插入条件 FP – 树时,当插入最后一个项目时,若该项目的节点已存在且其 Local_BCOIS 不为 NULL,则将节点的 Local_BCOIS 与条件模式基的 Local_BCOIS 的交集作为节点新的 Local_BCOIS;否则将节点的 Local_BCOIS 置为条件模式基的 Local_BCOIS。

从以上挖掘过程,我们的算法的基本思路可以总结如下:自下而上地在全局和条件 FP – 树中各节点的中计算和保存局部后共生项集;当挖掘某模式时,从该模式对应的各节点的局部后共生项集中提取全局后共生项集,利用全局后共生项集进行闭合性判断,而无需保存和遍历结果集。

4.3 算法描述

结合以上的挖掘过程,我们将算法简要概括如下:

(1)扫描 TDB,生成头表;

(2)扫描 TDB,生成 FP – 树,并对节点的局部后共生项集作初始化;

(3)从头表底部的项目开始,计算其各节点的局部后共生项集,生成条件模式基,从中提取后共生项集进行闭合性判断。若通过,进行项目合并得到频繁闭项集,递归构造并挖掘条件 FP – 树。

5 实验结果

我们的测试环境是:CPU 为 AMD Sempron 2800 + 1.61GHz,内存 512MB,硬盘 160GB,操作系统是 Windows XP。算法用 c#语言编写。数据集由生成器生成,特性是:总事务数为 10000,总项目数为 500,平均记录长度为 11.3。

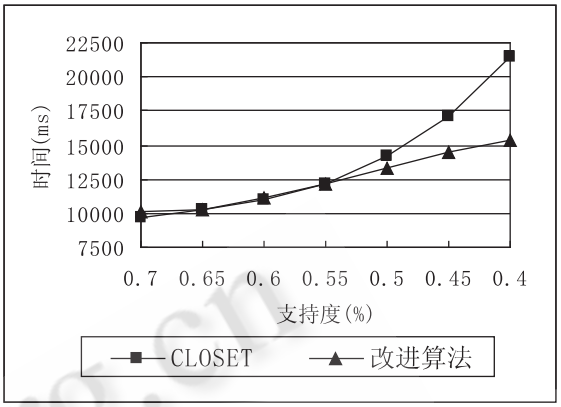

图 3　算法比较

实验结果如图 3 所示。从实验结果可以看出,当支持度阈值下降,结果集快速增大时,CLOSET 算法时间增长较大,而改进算法增长较小。

6 总结与展望

本文将共生项集的概念引入到频繁闭项集的挖掘中,阐述了共生项集的一些已被无形中应用,却未得到很好归纳研究的有趣性质,并利用这些性质提出了一种无需遍历结果集的闭合性检查方法,以此为基础对 CLOSET 算法进行改进,实验表明,当随着支持度阈值减小结果集快速增大时,改进算法时间效率的稳定性更好,效率更高。

但是,对共生项集的研究现在还处于比较初步的阶段。在今后的工作中,我们在以下几方面还可以继续深入研究:(1)频繁项集的挖掘任务现在存在许多变形,如挖掘 Top – k 的频繁闭项集[5],基于约束的频繁闭项集挖掘[6]等。如何将共生项集应用其中需要进一步研究。(2)是否有更合适的应用共生项集的数据结构,如何更有效地利用共生项集,也有待进一步研究。

参考文献

1 Pasquier N, Bastide Y, Taouil R, Lakhal L. Discovering frequent closed itemsets for association rules. In: Proceeding of the 7th International Conference on Database Theory (ICDT99). Jerusalem, 1999. 398 – 416.

2 Pei J, Han J. CLOSET: An efficient algorithm for mining frequent closed itemsets. (下转第 46 页)

图 3　灰度方差法分割效果图

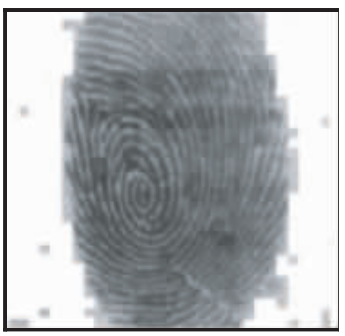

图 4　角部灰度均值法分割效果图

图 5　复合法分割效果图

通过实验发现,方向图法在绝大多数情况下都很有效,但当图像中方向信息不能准确提取时,则会失去作用;灰度方差法对于高对比度图像有较好的效果,但对于低对比度或增强后的图像则不适用;角部灰度均值法分割较为准确,而且处理速度很快。复合法结合了方向图和角部灰度均值两种方法的优点,可以快速有效地对指纹图像分割,与人的视觉分割要求接近,具有较强的稳定性和较高的分割正确率。

4　总结

本文提出了一种新的基于方向图和角部灰度均值的复合分割方法。本算法的优点是充分利用了指纹图像特有的方向性和灰度特性,对于方向图法和方差方法单独应用时无效的情况都适用,能准确地将背景区域从指纹图像中分离出来,从而提高了分割的精确度。实验结果表明,本算法分割图像较为彻底全面,为指纹的进一步识别提供条件,具有较强的可靠性和实用性。

参考文献

1　尹桂萍. 指纹识别技术中预处理技术各算法特点比较. 福建电脑,2006(11),51 – 52.

2　詹小四,尹义龙. 一种改进的指纹图像分割算法. 广西师范大学学报,2006,24(4):207 – 210.

3　陈明. 一种指纹图像分割方法. 计算机技术与应用进展,2004.

4　祁兵,景晓军等. 基于三角模融合算子的指纹图像分割方法. 计算机工程,2004.

(上接第 35 页)

In: Proceeding of the 2000 ACM – SIGMOD International Workshop Data Mining and Knowledge Discovery (DMKD00). Dallas, 2000. 11 – 20.

3　Zaki, Hsiao. CHARM: An efficient algorithm for closed itemset mining. In: Proceeding of the 2002 SIAM International Conference on Data Mining (SDM'02). Arlington, 2002. 457 – 473.

4　J Wang, J Han, J Pei. CLOSET +: Searching for the best strategies for mining frequent closed Itemsets. In: Proceeding of the 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD03). Washington, 2003. 236 – 245.

5　Wang J, Han J, Lu Y, Tzvetkov P. TFP: An Efficient Algorithm for Mining Top – K Frequent Closed Itemsets. IEEE Trans Knowl Data Eng, 2005,17(5):652 – 664.

6　崔立新, 苑森淼, 赵春喜. 约束性相联规则发现方法及算法. 计算机学报, 2000,23(2): 216 – 220.