Journal of Computer Applications
计算机应用,2018,38(4):995 - 1000
ISSN 1001-9081
CODEN JYIIDU
2018-04-10
http://www.joca.cn

文章编号:1001-9081(2018)04-0995-06
DOI:10.11772/j.issn.1001-9081.2017092389

基于数据集稀疏度的频繁项集挖掘算法性能分析

肖　文*,胡　娟
(河海大学文天学院 电气信息工程系,安徽 马鞍山 243031)
(*通信作者电子邮箱 cyees@163.com)

摘　要:频繁项集挖掘(FIM)是最基础的数据挖掘任务之一,被挖掘数据集的特征对FIM算法的性能有着显著影响。数据集稀疏度是体现数据集本质特征的属性之一,不同类型的FIM算法对数据集稀疏度的可扩展性有着很大的不同。针对如何量化度量数据集稀疏度及稀疏度对不同类型FIM算法性能影响等问题,首先回顾并讨论了已有的度量方法,然后提出两种新的量化度量数据集稀疏度的方法(基于事务差异度的度量方法和基于FP-Tree的度量方法)。这两种度量方法均考虑了FIM任务背景下最小支持度对数据集稀疏度的影响,反映的是事务频繁项集之间的差异度。最后通过实验验证了不同类型FIM算法对数据集稀疏度的可扩展性。实验结果表明,数据集稀疏度与最小支持度成反比,基于垂直格式的FIM算法在三类典型FIM算法中具有最佳的稀疏度可扩展性。

关键词:数据挖掘;频繁项集挖掘;稀疏度;可扩展性
中图分类号:TP311.5　　**文献标志码**:A

Performance analysis of frequent itemset mining algorithms based on sparseness of dataset

XIAO Wen*, HU Juan
(*Department of Electrical and Information Engineering, Hohai University Wentian College, Maanshan Anhui* 243031, *China*)

Abstract: Frequent Itemset Mining (FIM) is one of the most important data mining tasks. The characteristics of the mined datasets have a significant effect on the performance of FIM algorithms. Sparseness of datasets is one of the attributes that characterize the essential characteristics of datasets. Different types of FIM algorithms are very different in the scalability of dataset sparseness. Aiming at the measurement of sparseness of datasets and influence of sparsity on the performance of different types of FIM algorithms, the existing measurement methods were reviewed and discussed, then two methods were proposed to quantify the sparseness of the datasets: the measurement based on transaction difference and the measurement based on FP-Tree method, both of which considered the influence of the minimum support degree on the sparseness of the datasets in the background of the FIM task, and reflected the difference between the frequent itemsets of the transaction. The scalability of different types of FIM algorithms for sparseness of datasets was studied experimentally. The experimental results show that the sparseness of datasets is inversely proportional to the minimum support, and the FIM algorithm based on vertical format has the best scalability in three kinds of typical FIM algorithms.

Key words: data mining; Frequent Itemset Mining (FIM); sparseness; scalability

0　引言

频繁项集挖掘(Frequent Itemset Mining, FIM)是最基础的数据挖掘任务之一,是关联规则、分类、聚集、离群点分析等众多数据挖掘任务的重要组成部分,自它被提出以来[1],受到了越来越多的关注。经典的FIM算法可以分为三类:"产生-计数"类方法如Apriori[2]、DHP(Direct Hashing and Pruning)[3-4]、DIC(Dynamic Itemset Counting)[5]等;"模式增长"类方法如FP-Growth(Frequent Pattern Growth)[6]、LPTree(Linear Prefix Tree)[7]、FIUT(Frequent Items Ultrametric Trees)[8]、IFP(Improved FP-Growth)[9]、FPL/TPL(Frequent Pattern List/Transcation Pattern List)[10]等,上述两类算法都是基于水平数据格式的。第三类算法是基于垂直数据格式的,如Eclat(Equivalence Class Transformation)以及使用位向量优化的类Eclat算法等。

所有的FIM算法都包含两个基本计算步骤:产生(候选)项集及对项集进行计数。"产生-计数"类的方法在项枚举树上通过广度优先搜索(Breadth First Search, BFS)产生候选项集,并通过扫描整个数据集对每一个候选项集进行计数,从而得到频繁项集。"模式增长"及基于垂直数据格式的算法均通过深度优先搜索(Depth First Search, DFS)产生候选项集,可以尽可能早地将非频繁的项集进行剪枝,效率比BFS方法更高。"模式增长"方法中项集的计数通过构造条件数据库来实现,设有前缀项集p,其条件数据库为D_p(由数据集中所有包含项集p的事务组成),找到D_p中i个频繁项$\{x_1, x_2, \cdots, x_i\}$,可以直接得到$i$个频繁项集$\{p \cup x_1, p \cup x_2, \cdots, p \cup x_i\}$。基于垂直数据格式的FIM算法通过计算两个项集*tidset*的交集来确定生成项集的支持度,设有两个*tidset*:$t_a = \{1,2,3,4\}$,$t_b = \{2,3\}$分别表示项a在事务号为$\{1,2,3,4\}$的事务中出现,项b在事务号为$\{2,3\}$的事务中出现,则$t_{ab} = t_a \cap t_b =$

收稿日期:2017-10-13;**修回日期**:2017-11-06。　**基金项目**:安徽省高校自然科学研究项目(KJ2016A623)。
作者简介:肖文(1984—),男,安徽黄山人,讲师,硕士研究生,主要研究方向:分布式计算、数据挖掘;　胡娟(1985—),女,江苏海门人,讲师,硕士研究生,主要研究方向:软件工程、数据库系统。

{2,3},项集{a,b}的支持度即为 t_{ab} 的长度。可以将项集的 tidset 转换为 bitmap 来进一步提高计算交集的速度[11]。

被挖掘数据集的特征对 FIM 算法的效率有着显著影响。数据集的特征主要包括事务数、项数、平均事务长度、数据尺寸等,有不少研究都关注了不同 FIM 算法针对上述特征的扩展性[12]。但上述属性只能反映了数据集的一般特征,没有从根本上描述数据集的本质特征。数据集的稀疏度描述的是数据集中事务之间的差异度及信息的冗余量,是数据集的本质特征之一,对 FIM 算法的效率有着显著的影响,很少研究关注了 FIM 算法对于稀疏度的可扩展性问题。

为了了解不同 FIM 算法对于稀疏度的可扩展性,首先要对数据集稀疏度进行定量的分析。不少研究以数据集的一般特征为基础,文献[13-17]中提出了一些数据集稀疏度的度量方法;闫珍等[18]提出一种将事务数据库转换为二进制矩阵,借用工程数学中"稀疏矩阵"的概念来描述数据集的稀疏度;还有一些方法首先将数据集转换为特殊的数据格式(一般为类似 FP-Tree 的树结构),通过对特定数据结构的分析来得出数据集的稀疏度[19-20];Shepard[21]在 2013 年给出了一种基于等价类及等价类内部冗余度的数据稀疏度度量方法。虽然上述稀疏度度量方法一定程度上解决了数据集稀疏度度量问题,但没有兼顾到 FIM 任务背景下决定稀疏度的所有要素,且没有研究关注数据稀疏度对不同类型 FIM 算法的效率影响。

本文主要工作为:

1)对已有的数据集稀疏度量化度量方法进行了综述,讨论了每种类型方法的优缺点并进行了比较。

2)提出了一种基于事务频繁项集之间差异度的数据集稀疏度度量方法,这种度量方法有如下三个特点:考虑了最小支持度对数据稀疏度的影响(不同的支持度会导致频繁项在数据集中的不同分布),考虑了事务之间的差异(关联)度,数据集的整体稀疏度由各个局部稀疏度组成。

3)提出了一种基于 FP-Tree(Frequent Pattern Tree)的稀疏度的近似度量方法,这种方法同样考虑了最小支持度以及事务之间部分的差异(关联)度,虽然是一种近似的稀疏度度量方法,但计算效率比第一种方法更高。

4)通过实验研究了数据稀疏度对于不同类型 FIM 算法的性能影响,比较了不同类型 FIM 算法对数据稀疏度的可扩展性。

1 符号及相关定义

设 $I=\{i_1,i_2,\cdots,i_n\}$ 是项的集合,X 是一个项集,$X\subseteq I$,包含 k 个项的项集称为 k-项集;一个事务 $T=(tid,X)$,其中 tid 为事务标识,X 为一个项集;事务数据库 $D=\{t_1,t_2,\cdots,t_n\}$ 是 T 的集合。项集 X 在事务数据库 D 中的支持度 $Sup(X)$ 为 D 中包含 X 事务的数量。最小支持度 $minsup$ 是用户给定的一个阈值,如果 $Sup(x)\geqslant minsup$,则称项集 x 是频繁的。

定义 1 频繁项的集合 I'。一个事务数据库 D 频繁项的集合由所有频繁的项组成。

$$I'=\{i\mid i\in I \&\& Sup(i)\geqslant minsup\}$$

定义 2 事务的频繁项集 T'。一个事务 $T=(tid,X)$ 的频繁项集 T' 由事务中所有频繁的项组成。

$$T'=\{i\mid i\in X \&\& i\in I'\}$$

定义 3 项集的长度 $|X|$。一个项集的长度定义为项集中包含的项的个数。

定义 4 设有两个项集 X、Y,其中长度较大的一个项集表示为 $\max(X,Y)$。

2 已有度量方法

为了量化度量数据集的稀疏度,已提出的方法可以大致分为三类:第一类是使用数据集的基本特征作为参数来计算数据集的稀疏度;第二类是将数据集转换为特定的数据结构(大多是类似于 FP-Tree 的树结构),通过对特定数据结构特征的分析来体现数据集的稀疏度;第三类是利用等价类(闭项集)的概念,将事务的频繁项集划分为若干个等价类,通过计算每一个等价类的稀疏度来得到整个数据集的稀疏度度量。

从 FIM 挖掘的背景来看,数据集的稀疏度度量方法应考虑三个方面的要素:一是要考虑最小支持度的影响,根据先验性质[1],FIM 挖掘只关注数据集中所有频繁的项,不同的支持度会导致事务中包含不同的频繁项;二是要考虑事务频繁项集之间的差异度,全面地反映一个数据集内事务之间的关联程度;三是可以将整个数据集划分为若干部分,整个数据集的稀疏度由所有局部稀疏度构成。

2.1 数据集基本特征的度量方法

Bayardo 等[13]提出了一种数据集稀疏度度量的粗略描述,将稀疏数据集描述为"数据集中有大量的项,但每条事务中项的数量(平均事务长度)是很小的。"稀疏度的计算方法如下所示:

$$\text{数据集稀疏度}=|\overline{T_i}|/|I|$$

数据稀疏度使用事务平均长度和数据集中项总数的比值体现,这种计算方法没有涵盖上述三个要素的任何一个,只能说是一种十分粗略的稀疏度定量描述,其优点是计算十分方便。

Gouda 等[14]中提出一种利用频繁项集长度大致估计数据集稀疏度的方法。如果频繁项集的平均长度很长,则数据是密集的;反之是稀疏的。这种方法只能给出一个粗略的定性描述,且没有明确频繁项集的平均长度的阈值是多少。

有不少研究利用频繁项的平均支持度与最小支持度的比值作为量化估计数据集的方法,其计算公式为:

$$\text{数据集密集度}=\overline{Sup(I')}/minsup$$

数据越密集,则密集度越大;数据越稀疏,则密集度越小,稀疏度越大。这种方法虽然考虑了最小支持度对数据稀疏度的影响,计算也十分简便,但只考虑了数据集中频繁的项,而没有考虑事务频繁项集之间的关联关系,不能全面反映一个数据集的特征。

文献[15-17]中提出的度量方法考虑了项在整个数据集中的分布情况,使用分布比例来衡量数据集的稀疏度。设数据集中所有项的集合 $I=\{i_1,i_2,\cdots,i_n\}$,数据集稀疏度计算方法如下所示:

$$\text{项的分布比例} = \frac{\sum_{j=1}^{n} i_j}{|I| * |D|}$$

分布比例越高,则数据集越密集;分布比例越低,则数据集越稀疏。这种方法计算也十分简单,但存在着明显的缺点:没有体现最小支持度对数据集稀疏度的影响;没有反映事务频繁项集之间的关联程度;在某些情况下,两个数据集可能拥有相同的项分布比例,但两个数据集实际上存在着重大的差异。

闫珍等[18]提出的方法与分布比例的思想类似,首先将事务数据库转换为二进制矩阵,通过矩阵中非零元素的占比来得到数据集的稀疏度,其计算公式如下:

$$\text{稀疏度} = t/(m \times n)$$

其中:t 为矩阵中非零元素的个数;$m \times n$ 为矩阵的尺寸。一般而言,稀疏度小于等于 0.05 为稀疏数据集[22]。

2.2　基于特殊数据结构的度量方法

Grahne 等[19]中提出一种基于 FP-Tree 的稀疏度粗略的估计方法,首先将数据集构造成一棵 FP-Tree,然后以树层次的方法对该 FP-Tree 进行检查:若 FP-Tree 上 1/4 部分包含的节点少于 15% 的总结点,则数据集是密集的;否则数据集是稀疏的。这种方法考虑了最小支持度的影响,也考虑了事务频繁项集之间的关联程度,但只能得出定性的稀疏度度量结论(密集或稀疏),无法量化一个数据集的稀疏程度。在文献[23]中说明,这种度量方法有时会产生错误的结论。

为了克服稀疏度不能准确量化度量的缺点,Salleb-Aouissi 等[20]提出一种精确的度量方法,首先将数据集构成一个类似 FP-Tree 的数据结构二元决策图(Binary Decision Diagram, BDD),通过检查 BDD 的特征来得出数据集量化的稀疏度,具体计算公式如下:

$$SP_{\text{BDD}} = \frac{\text{BDD 中节点数}}{|I| * |D|}$$

若 SP_{BDD} 的比值很小,说明 BDD 中的节点很少,数据集是密集的;反之数据集为稀疏的。这种方法可以得到一个精确量化的稀疏度度量,但没有考虑最小支持度的因素。

上述两种基于树的度量方法实际上主要关注点是事务之间的差异(关联)程度。如果事务之间的差异度低,则在 FP-Tree 或 BDD 中一个节点可以代表多个事务共同项,整个数据结构中的节点就相应减少;反之则 BDD 中的节点数量更多。

2.3　基于等价类划分的度量方法

Shepard[21]中提出一种基于等价类的数据集稀疏度度量方法,主要借助了闭项集及等价类中的最小项集(Mininal Generators, MG)的概念[24]。其计算方法主要描述为:首先将数据集的所有事务划分为 n 个等价类 $\{Y_1, Y_2, \cdots, Y_n\}$,同一等价类中的事务具有相同的闭项集,整个数据集的稀疏度是所有等价类稀疏度的平均数:

$$Sp_{\text{数据集}} = \frac{1}{n}\sum_{i=1}^{n} Sp(Y_i)$$

每一个等价类稀疏度的计算方法为:

$$Sp(Y_i) = y/x^2$$

其中:y 为等价类中无冗余的 mg 的个数;x 为等价类中所有的项集数。

这种度量方法虽然可以给出一个量化精确的稀疏度度量,也考虑了部分事务之间的关联(同一个等价类内部的事务),但很明显,它没有考虑最小支持度对数据集稀疏度的影响,且没有考虑不同等价类之间的关联关系。

可以将上述度量方法从三个维度对比如表 1 所示。

表 1　六类稀疏度度量方法的比较

Tab. 1　Comparison of six methods of sparseness measurement

度量方法	类型	最小支持度	事务之间关联
基于平均长度的方法	定量	不支持	不支持
基于平均支持度的方法	定量	支持	不支持
基于项出现次数的方法	定量	不支持	不支持
基于 FP-Tree 的粗略方法	定性	支持	支持
基于 BDD 的方法	定量	不支持	支持
基于等价类划分的方法	定性	不支持	部分支持

3　稀疏度度量方法

本文首先提出一种定量的稀疏度度量方法,考虑最小支持度对数据集稀疏度的影响,全面关注事务之间的差异度(关联程度),整个数据集的稀疏度由局部稀疏度组成。

3.1　基于事务差异度的稀疏度度量方法

由于不同最小支持度下事务的频繁项集是完全不同的,因此在 FIM 任务背景下,数据集的稀疏度是一个与最小支持度紧密关联的指标。设数据集 $D = \{T_1, T_2, \cdots, T_n\}$,则所有事务之间的关系可用 R 来表示:

$$R = \{r \mid r = (T_i, T_j)\};\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant n,\ i \neq j$$

在 FIM 任务背景下,只需要考虑事务频繁项集之间的关系,可以得到 R':

$$R' = \{r' \mid r' = (T_i', T_j')\};\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant n, i \neq j$$

其中:T_i' 为 T_i 的频繁项集。

整个数据集的稀疏度可由任意两个事务之间的差异度定义而来。可以对任意事务之间的差异度求平均数得到数据集的稀疏度,即如下定义:

定义 5　设数据集 D 中共有 n 条事务,其关系集合 $R' = \{r_1, r_2, \cdots, r_n\}$,数据集 D 的稀疏度定义为 $SP(D)$,计算公式为:

$$SP(D) = \frac{1}{n}\sum_{i=1}^{i=n} dif(r_i)$$

下面讨论任意两个事务频繁项集之间差异度的计算问题。本文引入数据挖掘领域中“标称属性”的概念[25],将一个事务看成是由若干个标称属性刻画的对象,其频繁项集中的每一个项可以看成是一个标称属性。如有两个事务的频繁项集 $X' = \{a,b,c\}$ 和 $Y' = \{b,c,d,e\}$,则事务 X' 有 3 个标称属性,Y' 有 4 个标称属性。计算两个使用标称属性刻画的实体 i、j 之间差异度 $dif(i, j)$ 可使用如下计算公式[25]:

$$dif(i, j) = (p - m)/p$$

其中:m 为 i、j 中相同的属性数;p 为刻画实体属性的总数。如将 X' 和 Y' 均看成使用标称属性刻画的实体,它们具有共同的标称属性 $\{b,c\}$,刻画实体所有的属性数为 4,则两者的差异度为 $(4-2)/4 = 0.5$。

因此,可以正式将两个事务的频繁项集之间的差异度定义如下:

定义 6　两个事务的频繁项集 T_i' 和 T_j' 之间的差异度定义为 $dif(T_i', T_j')$,其计算方法为:

$$dif(T_i',T_j') = \frac{|\max(T_i',T_j')| - |\{i \mid i \in T_i' \& i \subseteq T_j'\}|}{|\max(T_i',T_j')|}$$

下面讨论数据集稀疏度的最大值和最小值。根据定义 5，$Sp(D)$ 由任意两条事务的频繁项集之间的差异度求平均值所得。对于 $dif(T_i',T_j')$，若 T_i' 和 T_j' 的长度相等且每一个项均相等，则 $dif(T_i',T_j') = 0$；若 T_i' 和 T_j' 的长度相等且每一个项均不相等，则 $dif(T_i',T_j') = 1$；除了上述两种极端情况之外，$dif(T_i',T_j')$ 是一个在范围(0,1) 内的小数。因此对于 $SP(D)$ 而言，其取值必然在区间[0,1] 内。为了便于查看和书写，可以将 $SP(D) \times 100\%$ 得到的一个百分比作为数据集稀疏度的表现方式。

3.2 基于 FP-Tree 的稀疏度度量方法

3.1 节提出的稀疏度度量方法支持最小支持度考虑了所有事务之间的差异度，具有较好的代表性；但有一个明显的缺陷，即计算成本很高。设数据集 D 中有 n 个事务，则 R' 中有 $[n(n-1)]/2$ 个元素，计算一个关系差异度的时间复杂度为 $O(1)$，则计算 $SP(D)$ 的时间复杂度为 $O(n^2)$，当数据集中包含海量事务时，计算成本很高。为了降低计算成本，可以将基于 FP-Tree 的方法和计算 SP_{BDD} 的方法进行结合，提出一种新的基于 FP-Tree 的稀疏度计算方法。其主要思想是将数据集转换为对应的 FP-Tree 后，对 FP-Tree 中的节点进行检查来代表数据集的稀疏度。设数据集 D 的 $I' = \{i_1,i_2,\cdots,i_n\}$，基于 FP-Tree 的稀疏度计算方式如下：

$$FPSP(D) = \frac{FPTree\ 中节点数}{\sum_{j=1}^{n} Sup(i_j)}$$

这种计算方法在构造 FP-Tree 时支持最小支持度，通过 FP-Tree 的节点共享体现事务之间的差异度。设数据集中有 n 个事务，若任意两个事务的频繁项集 T_i' 和 T_j' 的长度相等且每一个项均相等，则数据集最密集，$FPSP(D) = 1/n$；若任意两个事务的频繁项集 T_i' 和 T_j' 的长度相等且每一个项均不相等，则数据集最稀疏，$FPSP(D) = 1$；因此对于任何数据集 D，$FPSP(D)$ 的取值范围为[1/n, 1]。

4 实验结果分析

实验选取 FIM 挖掘标准数据集[26] 中的四个数据集用于测试本文提出的两种稀疏度计算方法及三种典型 FIM 算法对于稀疏度的可扩展性。其中：mushroom 和 chess 为两个传统的密集数据集，retail 和 T10I4D100K 为两个传统的稀疏数据集。数据集的基本属性如表 2 所示。

表 2 四个测试数据集的基本属性

Tab. 2 Basic properties of four test datasets

数据集	事务数	项数	事务平均长度
mushroom	8 124	119	29
chess	3 196	75	37
retail	88 162	16 470	10
T10I4D100K	100 000	870	10

4.1 稀疏度度量结果对比

使用本文第 3 章中提出的两种稀疏度量化度量方法，对四个标准数据集分别计算其量化稀疏度，第一种方法度量值记为 SP，第二种方法度量值记为 FPSP。具体结果见图 1。

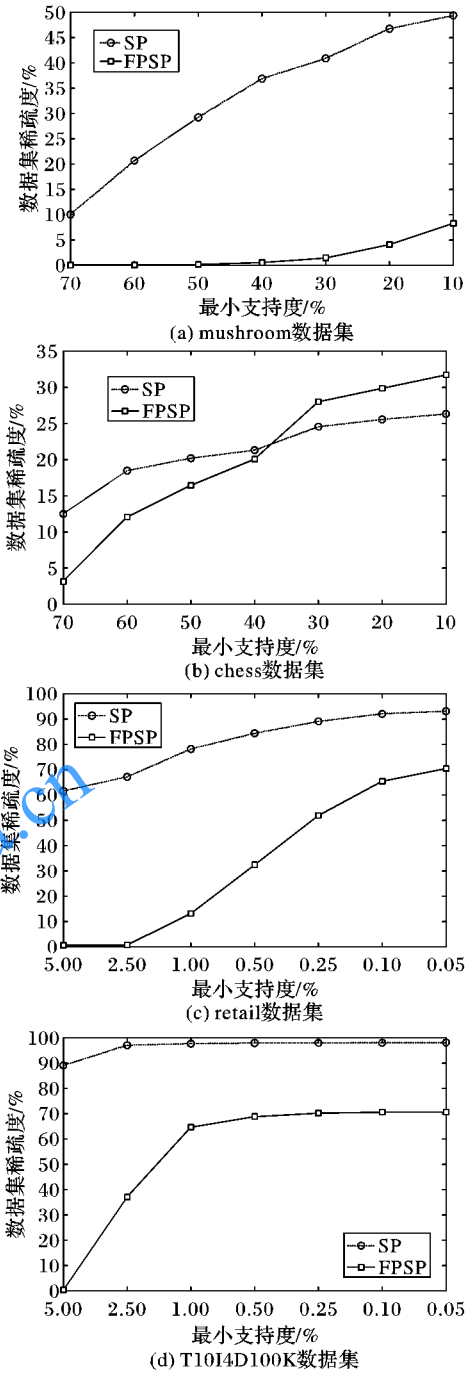

图 1 本文提出的两种方法对四个标准数据集的度量结果

Fig. 1 Sparseness of four standard datasets using two proposed methods

从图 1 可以看出：

1)在 FIM 任务背景下，数据集稀疏度(SP)是一个与最小支持度(minsup)相关联的一个数值，且与最小支持度成反比。

2)本文提出的两种稀疏度度量方法都具有良好的区分度，即对于一个数据集而言，不存在支持度不同而稀疏度相同的情况。从文献[21]中可以看出，基于等价类的稀疏度度量方法在 T40I0D100K 数据集上，当最小支持度分别为 1%、1.5%、2%、2.5%、5%、10% 时，其稀疏度值均为 100%，基于等价类的稀疏度度量方法在某些数据集上区分度较差。

3)对于相对密集的数据集来说，如 mushroom 和 chess，SP 的数值与 minsup 成近似线性关系，而 FPSP 增长得比较慢(主要原因是事务的频繁项集之间差别较小，可以共享很多前

缀);而对于较为稀疏的数据集来说,如 retail 和 T10I4D100K,SP 的增长率较小,而 FPSP 的增长率较大,这也说明对于稀疏度很高的数据集,不适合使用基于 FP-Tree 的 FIM 算法(如 FP-Growth),会构造大量的条件数据库,影响算法效率。

将文献[17,21]中提出的稀疏度计算方法分别记为 SP_{mg} 和 CD,选择标准数据集中较为密集的数据集 mushroom 和稀疏数据集 T10I4D100K,分别计算四种稀疏度如表 3 所示。

表 3　不同度量方法在两个数据集上的比较　%

Tab. 3　Comparison of different measurement methods on two data sets　%

数据集	minsup	SP	FPSP	SP_{mg}[17]	CD[21]
mushroom	20.00	46.762	4.105	46.89	44.80
	10.00	49.390	8.297	45.54	37.73
	5.00	50.932	12.278	44.00	30.51
	3.00	51.152	13.690	42.60	27.63
	2.00	51.180	14.028	43.81	25.64
	1.00	51.190	14.270	39.63	23.86
T10I4D100K	0.20	98.090	70.480	99.78	1.35
	0.15	98.100	70.580	99.52	1.31
	0.10	98.100	70.660	99.47	1.26
	0.05	98.190	70.730	98.21	1.20
	0.03	98.100	70.740	98.19	1.18
	0.02	98.100	70.740	98.04	1.18

从表 3 可以看到,本文提出的度量方法与已有方法在度量值上具有一定的一致性。但本文提出的两种度量方法均以事务之间的差异性为基础,度量值与最小支持度之间成明显的反比关系,而 SP_{mg} 以等价类中的冗余度为基础,CD 主要考虑的是项在数据集中的分布情况,相对来说其度量值和最小支持度之间的关联性不明显。

4.2　三种类型算法效率对比

从三类 FIM 算法中各选取一种经典的算法,研究其对于数据稀疏度的扩展性。“产生-测试”类选择经典的 Apriori 算法;“模式增长”类选择经典的 FP-Growth 算法;基于垂直数据格式的算法选择基于 bitmap 优化的 Eclat 算法(Eclat-bitmap 算法)。

三种算法在两个较密集的数据集上执行时间如图 2 所示,从中可以看出:

1)Apriori 算法的性能随着数据集稀疏度的增长降低得十分明显。其主要原因是随着稀疏度的增长,数据集中频繁项的数量急剧增长,基于 BFS 的候选项集产生策略会导致候选项集数量相对于频繁项的数量呈指数级别增长,极大地影响 Apriori 算法的性能。

2)FP-Growth 算法的性能在数据集稀疏度较小的情况下具有较好的性能,但随着数据集稀疏度的增加会产生明显的恶化。其主要原因是随着稀疏度的增长,事务频繁项集之间的差异度越来越大,导致构造的 FP-Tree 中事务间共享的节点数量减少,在挖掘过程中构造的条件数据库数量急剧增加,影响了 FP-Growth 算法的性能。

3)Eclat-bitmap 算法在数据集稀疏度较低的情况下具备十分良好的性能,在候选项集产生及计数两个阶段都优于上述两种算法。

由于 Apriori 算法在数据集稀疏度高时性能恶化十分剧烈,因此对于两个较为稀疏的算法已不再将 Apriori 算法作为参照对象,只考虑 FP-Growth 和 Eclat-bitmap 两种类型的算法。

FP-Growth 算法和 Eclat-bitmap 算法在两个较稀疏的数据集上执行时间如图 3 所示,从中可以看出:

1)相比较而言,Eclat-bitmap 算法在稀疏度较低的情况下,性能比 FP-Grwoth 算法要出色。但当稀疏度上升到一定高度时,性能与 FP-Growth 相当。

2)在稀疏度极高时,已有的三种典型算法的性能都不能

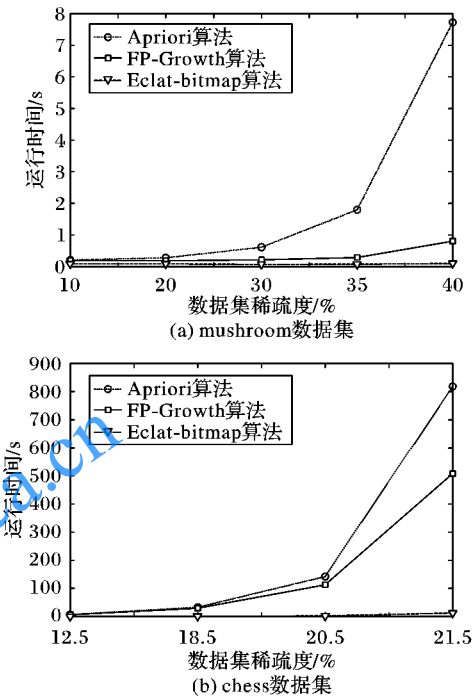

图 2　三种算法在密集据集上的执行时间

Fig. 2　Execution time of three algorithms on dense datasets

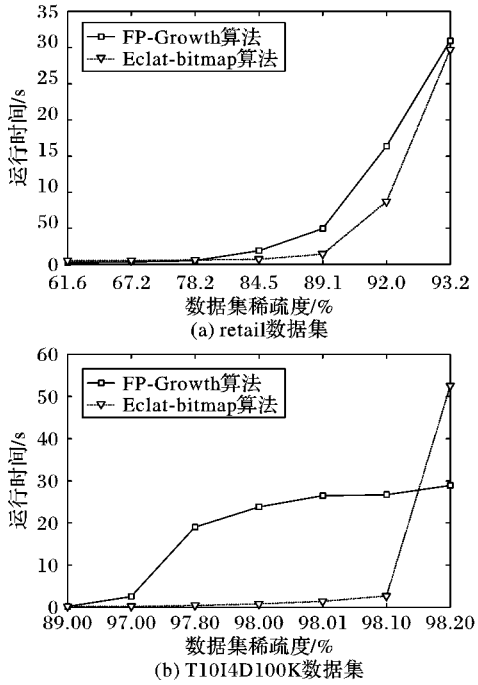

图 3　两种算法在稀疏数据集上的执行时间

Fig. 3　Execution time of two algorithms on sparse datasets

令人十分满意,不能很好地解决极度稀疏数据集中的 FIM 问题。

5 结语

数据集的特征对 FIM 算法的性能有着显著的影响,除了事务数、项数及平均事务长度等数据集的一般特征外,稀疏度是描述数据集本质特征最重要的属性之一。本文提出了两种量化度量数据集稀疏度的方法,考虑了最小支持度对数据集稀疏度的影响和事务之间的差异性:基于事务差异度的度量方法全面反映了所有事务之间的差异性(关联性);而基于 FP-Tree 的度量方法具有更好的计算性能。从实验结果来看,三种类型算法中"产生-测试"类的方法(如 Apriori)对稀疏度扩展性最差,基于数据垂直格式的方法(如 Eclat-bitmap)对稀疏度的扩展性最好,但当数据稀疏度增大到一定程度时,性能仍然出现了极度的恶化。因此,对于极度稀疏的数据集,可以数据垂直格式方法的设计思路,优化设计新的 FIM 算法,来提高挖掘的效率。

参考文献(References)

[1] AGRAWAL R, IMIELINSKI T, SWAMI A N. Mining association rules between sets of items in large databases[C]// Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. New York: ACM, 1993: 207 – 216.

[2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules in large databases[EB/OL]. [2017-05-10]. http://www.cs.uu.nl/docs/vakken/adm/agrawalfast.pdf.

[3] PARK J S, CHEN M S, YU P S. Using a hash-based method with transaction trimming for mining association rules[J]. IEEE Transactions on Knowledge & Data Engineering, 1997, 9(5): 813 – 825.

[4] OZEL S A, GUVENIR H A. An algorithm for mining association rules using perfect hashing and database pruning[C]// Proceedings of the 10th Turkish Symposium on Artificial Intelligence and Neural Networks. Berlin: Springer, 2001: 257 – 264.

[5] BRIN S, MOTWANI R, ULLMAN J D, et al. Dynamic itemset counting and implication rules for market basket data[J]. ACM Sigmod Record, 2001, 26(2): 255 – 264.

[6] HAN J, PEI J, YIN Y, et al. Mining frequent patterns without candidate generation: a frequent-pattern tree approach[J]. Data Mining & Knowledge Discovery, 2015, 8(1): 53 – 87.

[7] PYUN G, YUN U, RYU K H. Efficient frequent pattern mining based on linear prefix tree[J]. Knowledge-Based Systems, 2014, 55: 125 – 139.

[8] TSAY Y J, HSU T J, YU J R. FIUT: a new method for mining frequent itemsets[J]. Information Sciences, 2009, 179(11): 1724 – 1737.

[9] LIN K C, LIAO I E, CHEN Z S. An improved frequent pattern growth method for mining association rules[J]. Expert Systems with Applications, 2011, 38(5): 5154 – 5161.

[10] TSENG F C. An adaptive approach to mining frequent itemsets efficiently[J]. Expert Systems with Applications, 2012, 39(18): 13166 – 13172.

[11] BURDICK D, CALIMLIM M, FLANNICK J, et al. MAFIA: a maximal frequent itemset algorithm[J]. IEEE Transactions on Knowledge & Data Engineering, 2005, 17(11): 1490 – 1504.

[12] GOETHALS B, ZAKI M J. Advances in frequent itemset mining implementations: report on FIMI'03[J]. ACM Sigkdd Explorations Newsletter, 2003, 6(1): 109 – 117.

[13] BAYARDO R J J, AGRAWAL R, GUNOPULOS D. Constraint-based rule mining in large, dense databases[J]. Data Mining & Knowledge Discovery, 2000, 4(2/3): 217 – 240.

[14] GOUDA K, ZAKI M J. Efficiently mining maximal frequent itemsets[C]// ICDM 2001: Proceedings of the 2001 IEEE International Conference on Data Mining. Washington, DC: IEEE Computer Society, 2001: 163 – 170.

[15] PALMERINI P, ORLANDO S, PEREGO R. Statistical properties of transactional databases[C]// SAC 2004: Proceedings of the 2004 ACM Symposium on Applied Computing. New York: ACM, 515 – 519.

[16] STEINBACH M, TAN P N, KUMAR V. Support envelopes: a technique for exploring the structure of association patterns[C]// Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2004: 296 – 305.

[17] YAN H, CHEN K, LIU L, et al. SCALE: a scalable framework for efficiently clustering transactional data[J]. Data Mining & Knowledge Discovery, 2010, 20(1): 1 – 27.

[18] 闫珍, 皮德常, 吴文昊. 高维稀疏数据频繁项集挖掘算法的研究[J]. 计算机科学, 2011, 38(6): 183 – 186. (YAN Z, PI D C, WU W H. Research on frequent itemsets mining algorithm for high-dimensional sparse data[J]. Computer Science, 2011, 38(6): 183 – 186.)

[19] GRAHNE G, ZHU J F. Efficiently using prefix-trees in mining frequent itemsets[EB/OL]. [2017-05-10]. http://ceur-ws.org/Vol-90/grahne.pdf.

[20] SALLEB-AOUISSI A, VRAIN C. A Contribution to the Use of Decision Diagrams for Loading and Mining Transaction Databases[M]. Amsterdam: IOS Press, 2007: 220 – 242.

[21] SHEPARD T H. Looking for a structural characterization of the sparseness measure of(frequent closed) itemset contexts[J]. Information Sciences, 2013, 222(3): 343 – 361.

[22] 严蔚敏, 吴伟民. 数据结构(C 语言版)[M]. 北京: 清华大学出版社, 2007: 96 – 96. (YAN W M, WU W M. Data Structure (C Language Edition)[M]. Beijing: Tsinghua University Press, 2007: 96 – 96.)

[23] YAHIA S B, HAMROUNI T, NGUIFO E M. Frequent closed itemset based algorithms[J]. ACM SIGKDD Explorations Newsletter, 2006, 8(1): 93 – 104.

[24] PASQUIER N, BASTIDE Y, TAOUIL R, et al. Discovering frequent closed itemsets for association rules[C]// ICDT 1999: Proceedings of the 7th International Conference on Database Theory, LNCS 1540. Berlin: Springer, 1999: 398 – 416.

[25] 韩家炜, 范明. 数据挖掘: 概念与技术[M]. 北京: 机械工业出版社, 2012: 27 – 46. (HAN J W, FAN M. Data Mining: Concepts and Techniques[M]. Beijing: China Machine Press, 2012: 27 – 46.)

[26] IEEE computer society. Frequent itemset mining dataset repository[DB/OL]. [2017-11-01]. http://fimi.ua.ac.be/data/.

This work is partially supported by the Natural Science Foundation of the Colleges and Universities in Anhui Province (KJ2016A623).

XIAO Wen, born in 1984, M. S., lecturer. His research interests include distributed computing, data mining.

HU Juan, born in 1985, M. S., lecturer. Her research interests include software engineering, database system.