文章编号: 2095-2163(2021)01-0143-05 中图分类号:TP301.6 文献标志码:A

基于矩阵二进制编码遗传算法的频繁项集挖掘

杜嘉伟，余 粟
（上海工程技术大学 机械与汽车工程学院，上海 201620）

摘 要: 提出一种基于矩阵二进制编码的改进遗传算法 MGA（Matrix Genetic Algorithm），应用于挖掘关联规则中的频繁项集。通过对初始种群的编码以及降维保证了合理的初始适应度，并对遗传算法中交叉算子和变异算子生成新个体与筛选的过程进行优化，使算法有优良的全局和局部搜索能力。实验结果显示，MGA 算法的整体挖掘效率与质量良好。

关键词: 遗传算法；关联规则；频繁项集

Frequent itemsets mining based on matrix binary code Genetic Algorithm

DU Jiawei, YU Su
(School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

【Abstract】 This paper proposes an improved genetic algorithm MGA (Matrix Genetic Algorithm) based on matrix binary coding, which is applied to mining frequent itemsets in association rules. By encoding the initial population and reducing the dimensionality, a reasonable initial fitness is ensured, and then the process of generating new individuals and screening by the crossover operator and mutation operator in the genetic algorithm is optimized, so that the algorithm has excellent global and local search capabilities. Experimental results show that the overall mining efficiency and quality of the MGA algorithm are good.

【Key words】 Genetic Algorithm; association rules; frequent itemsets

0 引 言

频繁项集是那些出现在二进制或者定量数据集中的项目集,其出现频率超过指定的最小支持度,这是关联规则挖掘的基础和重要的部分[1]。由于多数挖掘定量频繁项目集的方法都是先将数据集转换成二进制的表现方法[2],本文拟重点关注二进制频繁项目集的挖掘。

为了挖掘到最大频繁项目集,通常会选择使用精确算法。其中使用率较高的 Apriori 算法[3]和 FPGrowth 算法[4]能够找到目标事务数据集中最大频繁项集,但是随着挖掘问题变得越来越复杂,也带来了一些缺点,如大量的时间和存储的成本。因此,更多的研究者开始运用遗传算法等启发式算法,虽然无法如精确算法一样得到问题的全局最优解,但却能在基于时间和空间的条件约束下求得问题的近似解,这也将具有更大的现实意义。

将传统遗传算法运用到长频繁项集的挖掘过程,但是并未将算法应用到大体量的数据集[5]。侯燕等人[6]研究优化了遗传算法对频繁项集的全局搜索能力,来挖掘关联规则并分析其性能,有着良好的挖掘效率,但对于频繁项集的提取率有限。在应用方面,已将遗传算法应用于挖掘医疗信息,并常见于健康医疗推荐中[7]。有关联规则研究针对弱关联分析,其中即使用遗传算法优化了数据挖掘的过程[8]。

本文提出一种基于矩阵二进制编码的遗传算法 MGA(Matrix Genetic Algorithm)来从事务数据集中挖掘出频繁项集。在 MGA 算法中共执行 k 步挖掘过程,交叉算子对前一次挖掘得到的频繁 $k-1$ 项集进行交叉计算生成长度为 k 的候选项集[9],变异算子则在候选项集中进行剪枝筛选挖掘出频繁 k 项集,其中在交叉变异计算过程中建立了一种编码模式将剪枝、判别候选项集是否为频繁项集、挖掘频繁项集三个部分有机地融合在一起,还提出一种基于降维的方法来提高挖掘性能。

1 基本概念

1.1 频繁项集

采用向量和矩阵的概念建立频繁项集(Frequent itemsets)的挖掘模型[10]。令 $I=\{item_1, item_2 \ldots, item_n\}$ 是 n 个数据项的集合,且 $item_j(1 \leqslant j \leqslant n)$ 称为项, $D=\{t_1, t_2, \cdots, t_m\}$ 是 m 个事务组成的数据集, $t_i(1 \leqslant i \leqslant m)$ 表示为一条事务,并且 t_i 是

作者简介:杜嘉伟(1995-),男,硕士研究生,主要研究方向:大数据挖掘; 余 粟(1962-),女,教授,主要研究方向:机电控制、计算机视觉。
通讯作者:余 粟 Email:suyu_sh@ hotmail. com

收稿日期: 2020-11-02

I 的子集。这里涉及的重要概念可做阐释分述如下。

(1)项集(*Itemset*):就是项的集合。包含 k 个项的项集称为 k 项集。当一个 k 项集计算所得的支持度满足设定的支持度阈值,就将该项集定义为频繁 k 项集。

(2)支持度(*Support*):描述了多个项集在所有事务中同时出现的概率。事务数据集中的支持度用 $sup(x)$ 来表示,支持度阈值用 min_sup 来表示,在挖掘频繁项集的过程中, min_sup 能够筛选并枚举出所有重要的项集。基于以上描述可推得定义如下[1]:

定义 1:给定 min_sup ,若 $sup(X) > min\_sup$,那么项目集 X 是频繁项目集。

定义 2:存在项目集 X_1、X_2,而且 $X_1 \subseteq X_2$,则 $sup(X_1) \geqslant sup(X_2)$

1.2 遗传算法

遗传算法(Genetic Algorithm,GA)是一种随机搜索方法,算法通过模拟生物进化的过程,由染色体的交叉和变异生成新个体,以适者生存为目的,淘汰无法适应环境的个体,逐步找到最优解。与传统搜索算法不同,遗传算法的初始种群是随机挑选的,其中个体再根据适应度函数进行标记筛选,通过交叉和变异计算生成后代种群,如此循环找到最优个体即为最优解[11]。对研究中主要用到的 2 个概念拟做简述如下。

(1)交叉算子:对 2 个个体按定义的规则相互交换其部分基因,产生 2 个继承到父代有效模式的新个体。为了得到有着多样性特性的候选项集,采用多点交叉的方式,即在 2 个不同的基因中的不同位置进行交换。

(2)变异算子:对交叉计算产生的新个体上的基因按定义的规则进行变异,形成新个体。变异计算的过程增加了后代种群的多样性,能够提高局部搜索能力。

2 MGA 算法

2.1 编码搜索空间、项目集、事务

给定具有 m 个项目 n 条事务的数据集 D ,将 MGA 的搜索空间视为 m x n 的矩阵 $\boldsymbol{M}$,文中用矩阵二进制(bit string) 来编码,即对一事务 T,若 $t \in T$,则令码串的第 i 位为1,否则为0[5]。比如数据集 I 是由 5 个项目集所构成,那么{10010} 则表示对应的事务 $\{t_1, t_4\}$,同时也表示一个二项集。

给定事务数据集 D 映射关系 $f: D \rightarrow \boldsymbol{M}$,即 $f(D) = (a_{ij})_{nxm} = \boldsymbol{M}$,并且,

$$r_{ij} = \begin{cases} 1, & I_j \in T_i, \\ 0, & I_j \notin T_i. \end{cases} \quad i = 1,2,\cdots,n, j = 1,2,\cdots,m. \tag{1}$$

其中, T_i 为 D 中事务的个数,m 为 I 中数据项的个数[12]。

根据公式(1),通过一次数据库的扫描,事务数据集 D 能映射为矩阵 $\boldsymbol{M}$。项集支持度可以通过二进制编码后的项集与矩阵 $\boldsymbol{M}$ 内积所得,项集支持度计算方法为:存在矩阵二进制编码的事务集合 $\boldsymbol{M}$,即:

$$\boldsymbol{M} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{bmatrix}, 1 \leq i \leq D_l, 1 \leqslant j \leqslant L. \tag{2}$$

其中, D_l 为 D 中事务的总个数,L 为 I 中数据项的总个数。

若经过交叉计算后得到的候选项集中存在 $\boldsymbol{B} = (b_1, b_2, \cdots, b_j)$,且 $\boldsymbol{B}$ 为 k 项集。

令向量 $\boldsymbol{C} = \boldsymbol{M} \cdot \boldsymbol{B}^{\mathrm{T}} = (c_1, c_2, \cdots, c_j)^{\mathrm{T}}$,其中 $c_j = a_{j1}b_1 + a_{j2}b_2 + \ldots + a_{jj}b_j (1 \leqslant j \leqslant L)$,遍历向量 $\boldsymbol{C}$ 中元素,当向量中有 n 个元素等于 k ,则 $sup(\boldsymbol{B}) = k$ 。当此候选项集的支持度满足支持度阈值,则此候选项集为频繁 k 项集。

2.2 初始种群及其降维

初始种群很大程度上影响着挖掘效率和质量。在 MGA 算法中,首先由遗传算法随机创建初始种群,然后根据以下推论通过降维来优化其中的一部分。

给定一个项集 X,存在 X_1, X_2 是 X 的真子集,即 $X_1, X_2 \subset X$,则可得:

$$\begin{cases} sup(X - X_1) \geqslant sup(X) \\ sup(X - X_2) \geqslant sup(X) \end{cases}$$

同时,根据先验启发式原理:

$$\begin{cases} sup(X_1) \geqslant sup(X) \\ Sup(X_2) \geqslant sup(X) \end{cases}$$

从中可以推论出:

$$\begin{cases} sup(X - X_1) + sup(X_1) \geqslant 2sup(X) \\ sup(X - X_2) + sup(X_2) \geqslant 2sup(X) \end{cases}$$

假设 $X - X_1$ 与 $X - X_2$ 的支持度为最大,则当 $sup(X_1) > sup(X_2)$,那么 $sup(X - X_1) \geqslant sup(X - X_2)$ 即项集 X 中去除支持度较低的 X_2 更有可能是

频繁项集。由此假设可得，在处理初始种群时，对矩阵 $\boldsymbol{M}$ 每列求内积得出事务数据集中每个项集的支持度，并根据 min_sup 筛选中不满足要求的项集 X，最终得到频繁一项集。

2.3　交叉和变异

根据先验启发式与剪枝原理，在挖掘频繁 k 项集的过程中，经过频繁 $k-1$ 项集交叉变异计算生成的 2 个长度为 k 的候选项集，当此集合去除了包含前 $k-1$ 步已判断为不频繁项的项集，那么集合中剩余项集更可能为频繁 k 项集。

基于以上的推论，MGA 算法对进行交叉计算的项集有所约束，要求进行运算的 2 个以二进制编码的项集要同时为 n 项集，且这 2 个项集要进行交叉计算的元素位置需为 01 对位，例如要进行交叉计算的其中一个项集为 {11000}，则要求进行交叉计算的另外一个项集为 2 项集，且项集元素需要为 01 对位，即另一项集为{00110}或是{00011}。

变异算子则是作用于交叉算子计算后得到的候选项集，首先判别当前项集是否包含非频繁 $k-1$ 项集，若是包含则对其剪枝去除，若不包含则根据公式(2)，判断当前候选项集是否为频繁项集，若为非频繁项集则进行变异计算，将 k 项集元素中的 01 元素翻转生成一个新的 k 项集之后再进行计算判别，不断循环直到得出一个频繁项集，此过程将剪枝、判别候选项集是否为频繁项集、挖掘频繁项集三个过程有机地融合在一起，避免了不断扫描数据库的过程，减少了运行时间和内存消耗。

2.4　适应度函数和候选结果

适应度函数用于评估个体的质量。在 MGA 中，个体的适应度等于该个体的支持度，即：

$$fit(x) = sup(x), \tag{3}$$

但是，为了避免候选项集中存在相同个体，调整适应度函数为：

$$fit(x) = \begin{cases} sup(x), sup(x) \geq min\_sup \ \&x \notin H, \\ 0, \quad sup(x) < \min, \\ 0, \quad x \notin H. \end{cases} \tag{4}$$

式中，数据集 H 是前一次挖掘得到的候选项集。

2.5　MGA 算法的伪代码

算法 1　基于矩阵二进制编码的改进遗传算法

输入：事务数据集 D，min_sup

输出：输出挖掘的所有频繁项集集合 FI

```
//初始化集合，FI 为频繁项集
CS ←∅, CSet ←∅, MSet ←∅;
//将事务数据集 D 以矩阵表示
F : D →M;
For (each x in D) do
    CSet ← CSet ∪ get Freq One(x);
  //矩阵 M 去除不频繁列
M = mat_proc(M);
FI←FI∪CSet;
  //p1,p2 为 CSet 中任意两个父项目集
For each (p1,p2) ∈ Cset do
    MSet←MSet∪Cross_cal(p1,p2);
End for
//p3 为 MSet 中任意一个项目集
For each p3 ∈ MSet do
    While ! Judg_Freq(p3) do
        MSet←MSet∪Mutation_cal(p3)
    End while
End for
FI←FI∪MSet;
End For
Return A
```

3　实验结果与分析

在本节中，使用了某零售商店数据集来评估 MGA 算法，其中实验数据包含有 65 000 多条商品交易信息和 2 241 个交易项目。实验环境为 64 位的 Win10 的系统，i7-9700KF，CPU 3.6 GHz 的计算机。

以下从不同支持度阈值的条件下，从挖掘频繁项集的平均时间、内存消耗量、提取率三个方面的性能指标来评价 MGA 和 Apriori 这两个算法并进行分析。

图 1 和图 2 分别表示了在不同的 min_sup 条件下，MGA 和 Apriori 算法挖掘相同电商数据集中的频繁项集的运行时间和内存消耗量。由实验结果可知，随着支持度的提高，2 个算法的运行时间和内存消耗量都处于下降的趋势，这是因为 MGA 在运行过程中避免了多次扫描数据库，只需要在挖掘频繁一项集时扫描一次数据库并将其编码为矩阵表示存入内存中，随后对初始种群进行了降维的操作，这减少了冗余候选项集的产生。而在挖掘频繁项集时，只需要通过公式(2)进行矩阵计算判别当前项目集是否为频繁项集，这也减少了挖掘搜索的时间。

图 3 表示了在不同的支持度阈值的条件下，MGA 算法、Apriori 算法的提取率，图 4 表示了迭代

次数和提取率的关系。由实验结果可知,Apriori 算法能够精确地挖掘出所有的最大频繁项集,提取率保持为 100%。MGA 算法随着支持度的升高,提取率越高,但是作为启发式近似算法在有限的迭代下不能挖掘所有的频繁项集。由图 4 分析可知,MGA 算法的迭代次数越多,频繁项集的提取率越高。

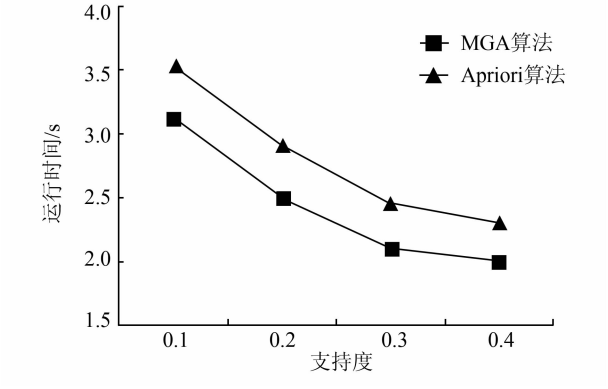

图 1 MGA 算法和 Apriori 算法运行时间比较

Fig. 1 Comparison of running time between MGA algorithm and Apriori algorithm

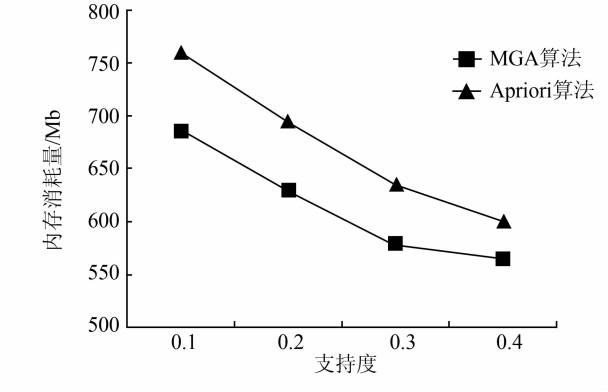

图 2 MGA 算法和 Apriori 算法内存消耗量比较

Fig. 2 Comparison of memory consumption between MGA algorithm and Apriori algorithm

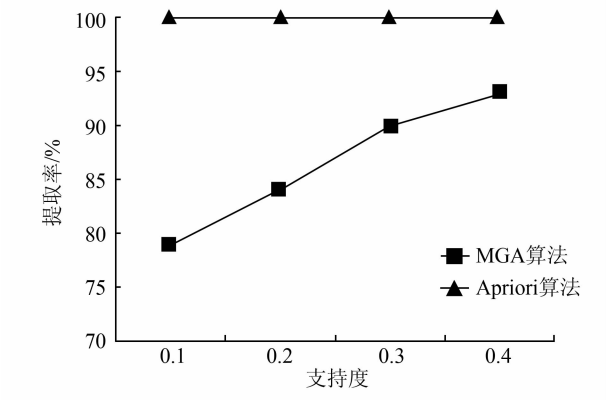

图 3 MGA 算法和 Apriori 算法的提取率比较

Fig. 3 Comparison of extraction rate between MGA algorithm and Apriori algorithm

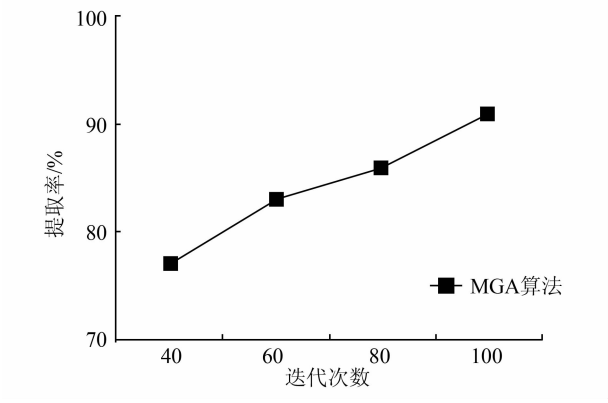

图 4 MGA 算法迭代次数与提取率的关系

Fig. 4 The relationship between iteration algebra and extraction rate of MGA algorithm

4 结束语

本文提出一种基于二进制编码的遗传算法 MGA 从数据集中挖掘频繁项集。对于繁杂的数据集,改进的遗传算法 MGA 通过对事务集进行矩阵二进制编码并通过矩阵计算挖掘频繁项集,避免了多次扫描数据库,而对初始种群的降维也降低了矩阵维度,从而减少了挖掘的运行时间和内存消耗。MGA 继承了遗传算法优良的全局搜索能力,通过优化算法中交叉和变异生成新个体的过程,改进了算法的局部搜索能力,进而提高了频繁项集整体挖掘效率与质量。接下来要考虑在高维数据集的情况下,从事在算法过程中动态地削减数据集的维度和频繁项集的搜索空间,以达到进一步优化整体挖掘质量与效率的研究。

参考文献

[1] HAN Jiawei, KAMBER M P J. Data mining: Concepts and techniques [M]. Third Edition. USA:Elsevier, 2011.

[2] ALATAS B, AKIN E. Rough particle swarm optimization and its applications in data mining [J]. Soft Computation, 2008, 12 (12): 1205-1218.

[3] GRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]//Proceedings of the 20th VLDB International Conference on Very high Database. San Francisco, CA: Morgan Kaufmann, 1994:487-499.

[4] HAN Jiawei, PEI Jian, YIN Yiwen. Mining frequent patterns without candidate generation[C]//Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data. Dallas, Texas, USA:ACM Press,2000:1-12.

[5] 王伟,高亮,吴涛. 基于遗传算法的长频繁项集挖掘方法[J]. 计算机技术与发展,2008,18(4):19-21.

[6] 侯燕,刘辛. 基于主从架构和 GA 的模糊关联规则挖掘算法[J]. 控制工程,2017,24(2):276-282.

(下转第 151 页)