Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 91 (2016) 324 – 333

Procedia Computer Science

Information Technology and Quantitative Management (ITQM 2016)

A Heuristic Rule based Approximate Frequent Itemset Mining Algorithm

Haifeng Li[a,*], Yuejin Zhang[a], Ning Zhang[a], Hengyue Jia[a]

[a]School of Information, Central University of Finance and Economics

Abstract

In this paper, we focus on the problem of mining the approximate frequent itemsets. To improve the performance, we employ a sampling method, in which a heuristic rule is used to dynamically determine the sampling rate. Two parameters are introduced to implement the rule. Also, we maintain the data synopsis in an in-memory data structure named SFIHtree to speed up the runtime. Our proposed algorithm SFIH can be efficiently performed over this tree. We conducted extensive experiments and showed that the mining performance can be improved significantly with a high accuracy when we used reasonable parameters.

Peer-review under responsibility of the Organizing Committee of ITQM 2016

Keywords: Frequent Itemset, Sampling, Data Mining, Heuristic Rule

1. Introduction

In data mining, frequent itemset mining is very important. An itemset is frequent if its support is not less than a minimum support specified by users. Traditional frequent itemset mining approaches mainly considered the problem of mining transaction databases. In these methods, transactions should be stored in secondary storage so that multiple scans over the data can be performed. Also, certain methods try to compact the transactions into the memory with different data structure, such as the FP-tree and the vertical bitmap. In [28], Han et.al presented a comprehensive survey of frequent itemset mining and discussed research directions. We will introduce the detailed in our related works.

In this paper, we will improve the performance with sampling technique. In this process, a balance between the performance and the accuracy will be considered. Generally, when we set a lower sampling rate, the performance will be improved, but the accuracy is low; on the other hand, when we set a higher sampling rate, the algorithm will be inefficient, but the accuracy will be much high. Thus, we will consider use a novel method to improve the performance with a low sampling rate.

Our contributions are as follows. 1) We present an in-memory data structure to quickly index the itemsets, which can speed up the itemset searching and retrieving. 2) We use a heuristic rule to improve the accuracy

Peer-review under responsibility of the Organizing Committee of ITQM 2016
doi:10.1016/j.procs.2016.07.087

while using the sampling method. 3) We propose an algorithm to maintain the mining results. 4) We conduct the experiments and show the effectiveness of our method.

The rest of this paper is organized as follows: In Section 2 we present the preliminaries. Section 3 presents the data structures, and illustrates our algorithm in detail. Section 4 introduces the related works. Section 5 evaluates the performance with theoretical analysis and experimental results. Finally, Section 6 concludes this paper.

2. Preliminaries

Given a set of distinct items $\Gamma = \{i1,i2,\ldots,in\}$ where $|\Gamma| = n$ denotes the size of Γ, a subset $X \subseteq \Gamma$ is called an itemset; suppose $|X| = k$, we call X a k-itemset. A concise expression of itemset $X = \{x1,x2,\ldots,xm\}$ is x1x2…xm. A database $D = \{T1,T2,\ldots,Tv\}$ is a collection wherein each transaction is a subset of Γ, namely an itemset. Each transaction Ti(i =1… v) is related to an id, i.e., the id of Ti is i. The absolute support (AS) of an itemset X, also called the weight of X , is the number of transactions which cover X, denoted $\Lambda(X)= \{|T \| T \in D \wedge X \subseteq T\}$; the relative support (RS) of an itemset X is the ratio of AS with respect to |D|, denoted $\Lambda r(X)= \Lambda(X)$. Given a relative imum support λ ($0 \leqslant \lambda \leqslant 1$), itemset X is frequent if $\Lambda r(X) \geqslant \lambda$. Table 1 is a simple database.

Example 1. Given a simple database D as shown in Table 1 and the minimum support λ=0.4, the frequent itemsets are {a, b, c, d, e, ab, ac, ad, bc, bd, be, cd, ce, de, abc, abd, acd, bcd, cde, abcd}.

Table 1. Simple Database

ID	Itemsets
1	a b c d e
2	a b c d
3	b c d
4	b e
5	c d e

3. Sampling Frequnt Itemset Mining Algorithm based on Heuristic Rule(SFIH)

In this section, we first present the data structure, and then we introduce the algorithm based on our heuristic rules.

3.1. SFIHtree

How to record the itemsets is very important since it has effect on the runtime and the running memory. In this section, we proposed an in-memory data structure name SFIHtree. For each itemset X, we use a tree node n_X, which is indeed a 2-tuple < item, sup >. item is the last item of X, and it is sorted by the support order under the a parent node; sup is the support of X. In our data structure, we can see that if node nX is the parent of node n_Y , then itemset Y is the superset of itemset X; also, all the nodes denote the frequent itemsets, and the infrequent nodes are deleted. Note in our data structure, the data can be mined incrementally. We show the SFIHtree of the database in Table 1 in Figure 1.