Mining Maximal Frequent Itemsets over Sampling Databases

Haifeng Li[1,a]

[1]School of Information Central University of Finance and Economics Beijing, China 100081

[a]mydlhf@cufe.edu.cn

Keywords: Maximal Frequent Itemset, Sampling, Data Mining.

Abstract. Maximal frequent itemset is an important representation of frequent itemset. This paper focuses on how to achieve the maximal frequent itemsets over databases by sampling technique. We use a tree-based data synopsis to maintain the frequent itemsets, based on which, an efficient algorithm SMFI is proposed. Our extensive experimental studies over a dataset show that sampling is effective when mining the maximal frequent itemsets.

Introduction

Frequent itemset mining is a traditional and important problem in data mining. An itemset is frequent if its support is not less than a minimum support specified by users. Traditional frequent itemset mining approaches have mainly considered the problem of mining static transaction databases. In these methods, transactions are stored in secondary storage so that multiple scans over the data can be performed. [15] reviewed the method of frequent itemset mining and discussed the research directions.

Frequent Itemsets are huge when the given threshold is low; consequently, the condensed representations of frequent itemsets including closed frequent itemsets[19], free frequent itemsets[12], approximate frequent k-sets[10], weighted frequent itemsets[21], and non-derivable frequent itemsets[14] were proposed; in addition, [20] focused on discovering a minimal set of unexpected itemsets.

Maximal frequent itemsets are one of the condensed representations, which only store the non-redundant cover of frequent itemsets, resulting in a space cost reduction. The concept of maximal frequent itemsets(MFI) was first proposed in 1998, an itemset is maximal frequent itemset if its support is frequent and it is not covered by others, we will introduce the details in Section 2.

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

Many maximal frequent itemset mining algorithms were proposed to improve the performance. The main considerations focused on developing new data retrieving method, new data pruning strategy and new data structure. Yang used directed graphs in [11] to obtain maximal frequent itemsets and proved that maximal frequent itemset mining is a #p problem. The basic maximal frequent itemset mining method is based on the a priori property of a itemset. The implementations were separated into two types: One type is an improvement of the a priori mining method, a bread first search[7], with utilizing data pruning, nevertheless, the candidate results are huge when an itemset is large; a further optimization was the down-top method, which counted the weight from the largest itemset to prevent super-itemset check, also, the efficiency was low when the threshold was small. Another one used depth first search[8] to prune most of the redundant candidate results, which, generally, is better than the first type. In these algorithms, many optimized strategies were proposed[9][6]: The candidate group built a head itemset and a tail itemset, which can quickly built different candidate itemsets; the super-itemset pruning could immediately locate the right frequent itemset; the global itemset pruning deleted all the subitemsets according to the sorted itemsets; the item dynamic sort strategy built heuristic rules to directly obtain the itemsets with high support,

which was extended by a further pruning based on tail itemset; the local check strategy got the related maximal frequent itemsets with the current itemset.

Preliminaries

Given a set of distinct items $\Gamma = \{i_1, i_2, \ldots, i_n\}$ 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 $X = \{x_1, x_2, \ldots, x_m\}$ is $x_1x_2\ldots x_m$. A database $D = \{T_1, T_2, \ldots, T_v\}$ is a collection wherein each transaction is a subset of Γ, namely an itemset. Each transaction $T_i (i = 1\ldots v)$ is related to an id, i.e., the id of T_i 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 $\Lambda_r(X) = \Lambda(X)/|D|$. Given a relative minimum support λ $(0 \leq \lambda \leq 1)$, itemset X is frequent if $\Lambda_r(X) \geq \lambda$. Table 1 is a simple database.

A maximal itemset is a largest itemset in a database D, that is, it is not covered by other itemsets. A maximal frequent itemset is both maximal and frequent in D, i.e., given an relative support λ, an itemset X is maximal frequent itemset if $\Lambda_r(X) \geq \lambda \wedge !Y|Y \supset X$.

Example 1. Given a simple database D as shown in Table 1 and an absolute support 2, the frequent itemsets are {a, b, c, d, e, ab, ac, ad, bc, bd, be, cd, ce, de, abc, abd, acd, bcd, cde, abcd}. The maximal frequent itemsets are {abcd,be,cde}.

Sampling Maximal Frequent Itemset Mining Algorithm

SMFITree Our aim is to to quickly search the itemsets when we perform mining. Consequently, we design a tree-based index named SMFITree(Sampling Maximal Frequent Itemset Tree). In the SMFITree, each node nX denotes an itemset X. n_X is a 2-tuple <item, sup>, in which item denotes the last item of the current itemset X, and it is sorted by the support order under the same parent; sup is the support of X. In our implementation, we use a pointer to link the child node to its parent node. From 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. We show the SMFITtree of the database of Table I in Figure 1. The bold rectangle represents the maximal frequent itemsets.

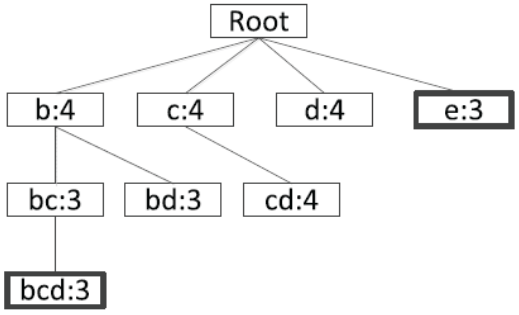

Figure 1 SMFITree for $\lambda = 3$

Sampling the Databases We use a simple sampling method to get the samples from the database, that is, we randomly take the samples without putting back, until we get the transactions of specialized count. During this process, we make sure that the sampling is normal distributed. The count will be specified by the users. In the experiments, we will use different counts to evaluate our algorithm.

SMFI Algorithm We propose a depth-first algorithm to conduct the mining. In this algorithm, we first generate a root node, and then we create the children nodes of the root, which represent the distinct items. For each child, we recursively generate $X \cup Y$ for itemset X with its sibling itemset Y, and we compute the support, if the support is larger than the minimum support, we will generate a child node $n_{X \cup Y}$ for n_X. We also use a collection to maintain the maximal frequent itemsets. Once a new node is generated, the itemset will be compared to the itemsets in the collection, if it is not

covered, it will be append in the collection.

Experiments

We conducted the experiments to evaluate the performance of SMFI. In the experiments, we use the minimum support and sampling rate as the major elements to do the evaluation.

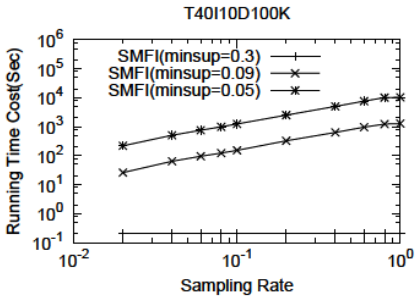

Fig. 2 Runtime Cost

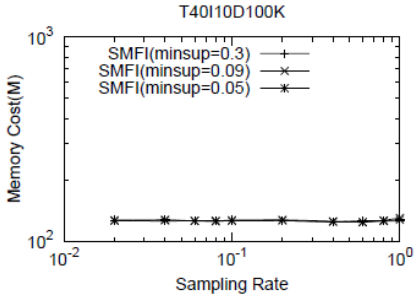

Fig. 3 Memory Cost

Running Environment and Datasets All the algorithms were implemented with Python, compiled with **Wingide** running on Microsoft Windows 7 and performed on a PC with a 3.60GHZ Intel Core i7-4790M processor and 12GB main memory.

Table 2 Dataset Characteristics

DataSet	Trans Count	Average Size	Min Size	Max Size	Items Count	Trans Correlation
T40I10D100K	100 000	4	77	40	1000	24

We employed a synthetic dataset named T40I10D100K as the evaluation dataset, which was generated by the IBM data generator. The detailed data characteristics are shown in Table 2.

Table 3 Precision and Recall over T40I10D100K Dataset

Sampling Rate	Pre(ms=0.09)	Re(ms=0.09)	Pre(ms=0.05)	Re(ms=0.05)
0.8	99.09%	100%	99.34%	100%
0.6	100%	100%	99.01%	100%
0.4	98.18%	100%	98.01%	99.66%
0.2	98.18%	99.08%	98.01%	99%
0.1	94.55%	99.05%	97.02%	98.32%
0.08	96.36%	99.07%	96.03%	96.99%
0.06	96.36%	97.25%	96.36%	97%
0.04	96.36%	95.5%	96.03%	95.39%
0.02	92.73%	96.23%	94.7%	93.77%

We evaluated the SMFI algorithm over different sampling rate; plus, the minimum support was set to various values. As can be seen from Figure 2, when we decreased the sampling rate, that is, from 0.9 to 0.01, the mining efficiency increased significantly over the dataset. Note when the minimum support is high, we can see that the sampling method has almost no effect over the performance, which is reasonable since little frequent itemsets were generated for such a parameter. Furthermore, we can see from Figure 3, the memory cost was almost unchanged when we alter the sampling rate. We argue that it is reasonable since the index we need to maintain in the memory are almost the same. Furthermore, we compared the accuracy of SMFI algorithm over different sampling rate and different minimum support. As can be seen from Table 3, the precision and recall decreased when we reduce the sampling rate. On the other hand, we noticed that there was almost no relation between the accuracy and the minimum support. As a result, we can find that the performance is inversely proportional to the accuracy.

Conclusions

In this paper we made a study over the algorithm of mining the maximal frequent itemsets on a

database with sampling method. An effective data structure, the SMFITree, was used to record all the frequent itemsets. An additional collection is to maintain the maximal frequent itemsets for quickly pruning. We also proposed an algorithm SMFI to maintain the data synopsis in SMFITree. In the algorithm, we use the sampling method to more quickly compute the support. Our extensive experimental results showed that our sampling method can significantly improve the performance with a high accuracy.

Acknowledgement

This research is supported by the National Natural Science Foundation of China (61100112, 61309030), Beijing Higher Education Young Elite Teacher Project (YETP0987),). Key project of National Social Science Foundation of China(13AXW010), 121 of CUFE Talent project Young doctor Development Fund in 2014 (QBJ1427).

Refereneces

[1] F.Afrati, A.Gionis, and H.Mannila, Approximating a Collection of Frequent Sets, in Proc. SIGKDD'2004

[2] G.Yang. The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns. in Proc. SIGKDD'2004.

[3] J.Boulicaut, A.Bykowski, and C.Rigotti, Free-sets: a condensed representation of boolean data for the approximation of frequency queries, Data Mining and Knowledge Discovery, 7 (2003) 5-22

[4] R.Agrawal, and R.Srikant, Fast algorithms for mining association rules, in: Proc. VLDB'1994.

[5] T.Calders, and B.Goethals, Mining All Non-Derivable Frequent Itemsets, in: Proc. PKDD'2002

[6] J.Han, H.Cheng, D.Xin, and X.Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery, 17 (2007) 55-86

[7] J.Han, and J.Pei, Mining frequent patterns by pattern-growth: methodology and implications, in: Proc. SIGKDD'2000

[8] S.Kevin, and R.Ramakrishnan, Bottom-Up Computation of Sparse and Iceberg CUBEs, in: Proc. SIGMOD'1999.

[9] G.Mao, X.Wu, X.Zhu, and G.Chen, Mining Maximal Frequent Itemsets from Data Streams, Journal of Information Science 33 (3) (2007) 251-262

[10] N.Pasquier, Y.Bastide, R.Taouil, and L.Lakhal, Discovering frequent closed itemsets for association rules, in: Proc. ICDT'1999