Top Down FP-Growth for Association Rule Mining

Ke Wang, Liu Tang, Jiawei Han, and Junqiang Liu

School of Computing Science, Simon Fraser University
{wangk, llt, han, jliui}@cs.sfu.ca

Abstract. In this paper, we propose an efficient algorithm, called **TD-FP-Growth** (the shorthand for Top-Down FP-Growth), to mine frequent patterns. **TD-FP-Growth** searches the FP-tree in the top-down order, as opposed to the bottom-up order of previously proposed FP-Growth. The advantage of the top-down search is not generating conditional pattern bases and sub-FP-trees, thus, saving substantial amount of time and space. We extend **TD-FP-Growth** to mine association rules by applying two new pruning strategies: one is to push multiple minimum supports and the other is to push the minimum confidence. Experiments show that these algorithms and strategies are highly effective in reducing the search space.

1 Introduction

Association rule mining has many important applications in real life. An association rule represents an interesting relationship written as $A \Rightarrow B$, read as "if A occurs, then B likely occurs". The probability that both A and B occur is called the *support*, and written as *count*(AB). The probability that B occurs given that A has occurred is called the *confidence*. The association rule mining problem is to find all association rules above the user-specified minimum support and minimum confidence. This is done in two steps: step 1, *find all frequent patterns*; step 2, *generate association rules from frequent patterns*.

This two-step mining approach suffers from several drawbacks. First, only a single uniform minimum support is used, though the distribution of data in reality is not uniform. Second, the two-step process does not consider the confidence constraint at all during the first step. Pushing the confidence constraint into the first step can further reduce search space and hence improve efficiency.

In this paper, we develop a family of algorithms, called **TD-FP-Growth**, for mining frequent patterns and association rules. Instead of exploring the FP-tree in the bottom-up order as in [5], **TD-FP-Growth** explores the FP-tree in the top-down order. The advantage of the top-down search is not constructing conditional pattern bases and sub-trees as in [5]. We then extend **TD-FP-Growth** to mine association rules by applying two new pruning strategies: **TD-FP-Growth(M)** pushes multiple minimum supports and **TD-FP-Growth(C)** pushes the minimum confidence.

2 Related Work

Since its introduction [1], the problem of mining association rules has been the subject of many studies [8][9][10][11]. The most well known method is the Apriori's *anti-monotone* strategy for finding frequent patterns [3]. However, this method suffers

M.-S. Chen, P.S. Yu, and B. Liu (Eds.): PAKDD 2002, LNAI 2336, pp. 334-340, 2002.

from generating too many candidates. To avoid generating many candidates, [4] proposes to represent the database by a *frequent pattern tree* (called the FP-tree). The FP-tree is searched recursively in a bottom-up order to grow longer patterns from shorter ones. This algorithm needs to build conditional pattern bases and sub-FP-trees for each shorter pattern in order to search for longer patterns, thus, becomes very time and space consuming as the recursion goes deep and the number of patterns goes large.

As far as we know, [7][8] are the only works to explicitly deal with non-uniform minimum support. In [7], a minimum item support (MIS) is associated with each item. [8] bins items according to support and specifies the minimum support for combinations of bins. Unlike those works, our specification of minimum support is associated with the consequent of a rule, not with an arbitrary item or pattern.

3 TD-FP-Growth for Frequent Pattern Mining

As in [5], **TD-FP-Growth** first constructs the FP-tree in two scans of the database. In the first scan, we accumulate the count for each item. In the second scan, only the frequent items in each transaction are inserted as a node into the FP-tree. Two transactions share the same upper path if their first few frequent items are same. Each node in the tree is labeled by an item. An *I_node* refers to a node labeled by item *I*. For each item *I*, all *I*_nodes are linked by a *side-link*. Associated with each node *v* is a count, denoted by *count(v)*, representing the number of transactions that pass through the node. At the same time, a header table H(Item, count, side-link) is built. An entry (*I, H(I), ptr*) in the header table records the total count and the head of the side-link for item *I*, denoted by *H*(*I*) and *ptr* respectively. Importantly, the items in each transaction are lexicographically ordered, and so are the labels on each path in FP-tree and the entries in a header table. We use Example 3.1 to illustrate the idea of **TD-FP-Growth**.

Example 3.1 A transaction database is given as in the following Figure 3.1. Suppose that the minimum support is 2. After two scans of transaction database, the FP-tree and the header table H is built as Figure 3.1.

The *top-down mining* of FP-tree is described below. First, entry *a* at the top of H is frequent. Since *a*_node only appears on the first level of the FP-tree, we just need to output {*a*} as a frequent pattern.

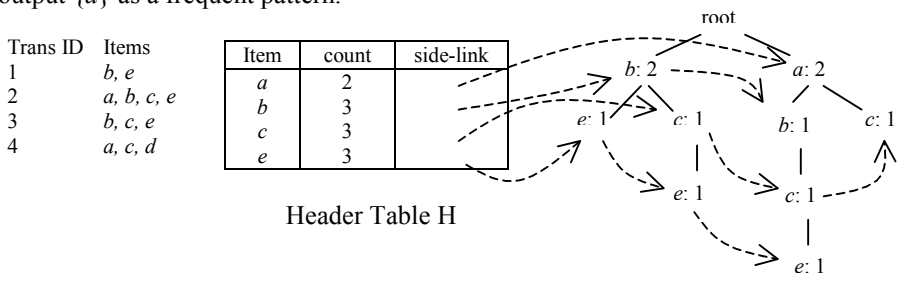

Figure 3.1 Transaction table, FP-tree and H

Then, for entry *b* in H, following the side-link of *b*, we walk up the paths starting from *b*_node in the FP-tree once to build a sub-header-table for *b*, denoted H_*b*.