Summarization in Pattern Mining

Mohammad Al Hasan
Rensselaer Polytechnic Institute, USA

INTRODUCTION

The research on mining interesting patterns from transactions or scientific datasets has matured over the last two decades. At present, numerous algorithms exist to mine patterns of variable complexities, such as set, sequence, tree, graph, etc. Collectively, they are referred as **Frequent Pattern Mining** (FPM) algorithms. FPM is useful in most of the prominent knowledge discovery tasks, like classification, clustering, outlier detection, etc. They can be further used, in database tasks, like indexing and hashing while storing a large collection of patterns. But, the usage of FPM in real-life knowledge discovery systems is considerably low in comparison to their potential. The prime reason is the lack of interpretability caused from the enormity of the output-set size. For instance, a moderate size graph dataset with merely thousand graphs can produce millions of frequent graph patterns with a reasonable support value. This is expected due to the combinatorial search space of pattern mining. However, classification, clustering, and other similar Knowledge discovery tasks should not use that many patterns as their knowledge nuggets (features), as it would increase the time and memory complexity of the system. Moreover, it can cause a deterioration of the task quality because of the popular "*curse of dimensionality*" effect. So, in recent years, researchers felt the need to summarize the output set of FPM algorithms, so that the summary-set is *small*, *non-redundant* and *discriminative*. There are different summarization techniques: lossless, profile-based, cluster-based, statistical, etc. In this article, we like to overview the main concept of these summarization techniques, with a comparative discussion of their strength, weakness, applicability and computation cost.

BACKGROUND

FPM had been the core research topic in the field of data mining for the last decade. Since, its inception with the seminal paper of mining association rules by Argrawal *et al* (Agrawal & Srikant, 1994), it has matured enormously. Currently, we have very efficient algorithms for mining patterns with higher complexity, like sequence (Zaki, 2001), tree (Zaki, 2005) and graph (Yan & Han, 2002; Hasan, Chaoji, Salem & Zaki, 2005). The objective of FPM is as follows: Given a database D, of a collection of events (an event can be as simple as a set or as complex as a graph) and a user defined support threshold π^{min}; return all patterns (patterns can be set, tree, graph, etc. depending on D) that are frequent with respect to π^{min}. Sometimes, additional constraints can be imposed besides the minimum support criteria. For details on FPM, see data mining textbooks (Han & Kamber, 2001).

FPM algorithms search for patterns in a combinatorial search space, which is generally very large. But, the ***anti-monotone*** property allows fast pruning: which states, *"If a pattern is frequent, so is all it's sub-pattern; if a pattern is infrequent, so is all its super-pattern."* Efficient data structure and algorithmic techniques on top of this basic principle enable FPM algorithms to work efficiently on database of millions events. However, the usage of frequent patterns in knowledge discovery tasks requires the analysts to set a reasonable support value for the mining algorithms to obtain interesting patterns, which, unfortunately, is not that straightforward. Experience suggests that if the support value is set too high, only few common-sense patterns are obtained. For example, if the database events are recent movies watched by different subscribers, using a very high support will only return the set of super-hit movies which are liked by anybody. On the other hand, setting low support value returns enormously large number of frequent patterns that are difficult to interpret; many of those are redundant too. So, ideally one would like to set the support value at a comparably lower threshold and then adopt a summarization or compression technique to obtain a smaller *FP*-set, comprising interesting, non-redundant, and representative patterns.

Figure 1: An itemset database of 6 transactions (left). Frequent, Maximal and Closed patterns mined from the dataset in 50% support (right)

Transaction Database, D
1. A C T W
2. C D W
3. ACTW
4. ACDW
5. ACDTW
6. CDT

Frequent Patterns in D (Minimum Support = 3)			
Support	**Frequent Pattern**	**Maximal Pattern**	**Closed Pattern**
6	C		C
5	W, CW		CW
4	A, D, T, AC, AW, CD, CT, ACW		CD, CT, ACW
3	AT, DW, TW, ACT, ATW, CDW, CTW, ACTW	CDW, ACTW	CDW, ACTW

MAIN FOCUS

The earliest attempt to compress the FPM result-set was to mine Maximal Patterns (Bayardo, 1998). A frequent pattern is called maximal, if it is not a sub-pattern of any other frequent pattern (see the example in figure 1). Depending on the dataset, maximal pattern can reduce the result-set substantially; especially for dense dataset, the compression ratio can be very high. And, maximal patterns can be mined in the algorithmic framework of FPM processes without a post-processing step. The limitation of maximal pattern mining is that the compression also loses the support information of the non-maximal patterns; for example, in figure 1, from the list of maximal patterns we can deduce that the pattern *CD* is also frequent (since *CDW* is frequent), but its support value is lost (which is 4, instead of 3). Support information is critical if the patterns are used for rule generation, as it is essential for confidence computation. To circumvent that, Closed Frequent Pattern Mining was proposed by Zaki (Zaki, 2000). A pattern is called closed, if it has no super-pattern with the same support. The compressibility of closed frequent mining is smaller than the maximal pattern mining, but for the earlier, all frequent patterns and also, their support information can be immediately retrieved (without further scan of the database). Closed frequent pattern can also be mined within the FPM process.

Pattern compression offered by maximal or closed mining framework is not sufficient, as the result-set size is still too large for human interpretation. So, many pattern summarization techniques have been proposed lately, each with different objective preference. It is difficult and sometimes, not fair, to compare them. For instance, in some cases, the algorithms try to preserve the support value of the frequent patterns; whereas, in other cases the support value is completely ignored and more emphasis is given in controlling the redundancy in patterns. In the following few paragraphs we discuss the main ideas of some of the major compression approaches, with their benefits and limitations. At the end of this section (see table 1), we show the benefits/limitations of these algorithms in tabular form for quick references.

Top-*k* Patterns

If the analyst has a predefined number of patterns in mind that (s)he wants to employ in the knowledge discovery tasks, *top-k patterns* is one of the best summarization technique. Han et al. (Han, Wang, Lu & TzVetkov, 2002) proposed one of the earliest algorithms that falls in this category. Their top-*k* patterns are *k* most frequent closed patterns with a user-specified minimum-length, *min_l*. Note that, minimum-length constraint is essential, since without it only length-1 patterns (or their corresponding closed super-pattern) will be reported, since they always have the highest frequency. The authors proposed efficient implementation of their proposed algorithm using FP-Tree; un-