Mining Frequent Patterns via Pattern Decomposition

Qinghua Zou
University of California - Los Angeles, USA

Wesley Chu
University of California - Los Angeles, USA

INTRODUCTION

Pattern decomposition is a data-mining technology that uses known frequent or infrequent patterns to decompose a long itemset into many short ones. It finds frequent patterns in a dataset in a bottom-up fashion and reduces the size of the dataset in each step. The algorithm avoids the process of candidate set generation and decreases the time for counting supports due to the reduced dataset.

BACKGROUND

A fundamental problem in data mining is the process of finding frequent itemsets (FI) in a large dataset that enable essential data-mining tasks, such as discovering association rules, mining data correlations, and mining sequential patterns. Three main classes of algorithms have been proposed:

- **Candidates Generation and Test (Agrawal &Srikant, 1994; Heikki, Toivonen &Verkamo, 1994; Zaki et al., 1997):** Starting at k=0, it first generates candidate *k+1* itemsets from known frequent *k* itemsets and then counts the supports of the candidates to determine frequent *k+1* itemsets that meet a minimum support requirement.
- **Sampling Technique (Toivonen, 1996):** Uses a sampling method to select a random subset of a dataset for generating candidate itemsets and then tests these candidates to identify frequent patterns. In general, the accuracy of this approach is highly dependent on the characteristics of the dataset and the sampling technique that has been used.
- **Data Transformation:** Transforms an original dataset to a new one that contains a smaller search space than the original dataset. FP-tree-based (Han, Pei & Yin, 2000) mining first builds a compressed data representation from a dataset, and then, mining tasks are performed on the FP-tree rather than on the dataset. It has performance improvements over Apriori (Agrawal &Srikant, 1994), since infrequent items do not appear on the FP-tree, and, thus, the FP-tree has a smaller search space than the original dataset. However, FP-tree cannot reduce the search space further by using infrequent 2-item or longer itemsets.

What distinguishes pattern decomposition (Zou et al., 2002) from most previous works is that it reduces the search space of a dataset in each step of its mining process.

MAIN THRUST

Both the technology and application will be discussed to help clarify the meaning of pattern decomposition.

Search Space Definition

Let $N=X{:}Y$ be a transaction where X, called the head of N, is the set of required items, and Y, called the tail of N, is the set of optional items. The set of possible subsets of Y is called the power set of Y, denoted by $P(Y)$.

Definition 1

For $N=X{:}Y$, the set of all the itemsets obtained by concatenating X with the itemsets in $P(Y)$ is called the *search space* of N, denoted as $\{X{:}Y\}$. That is,

$$\{X : Y\} = \{X \cup V \mid V \in P(Y)\}.$$

For example, the search space $\{b{:}cd\}$ includes four itemsets *b*, *bc*, *bd*, and *bcd*. The search space $\{{:}abcde\}$ includes all subsets of *abcde*.

By definition 1, we have $\{X{:}Y\}=\{X{:}Z\}$, where $Z=Y-X$ refer to the set of items contained in Y but not in X. Thus, we will assume that Y does not contain any item in X, when $\{X{:}Y\}$ is mentioned in this article.

Definition 2

Let S, S_1, and S_2 be search spaces. The set $\{S_1, S_2\}$ is a *partition* of S if and only if $S=S_1 \cup S_2$ and $S_1 \cap S_2=\phi$. The relationship is denoted by $S=S_1+S_2$ or $S_1=S-S_2$ or $S_2=S-S_1$. We say S is partitioned into S_1 and S_2. Similarly, a set $\{S_1, S_2, ..., S_k\}$ is a partition of S if and only if $S=S_1 \cup S_2 \cup ... \cup S_k$ and $S_i \cap S_j=\phi$ for $i,j \in [1..k]$ and $i \neq j$. We denote it as $S=S_1+S_2+...+S_k$.

Let a be an item where aX is an itemset by concatenating a with X.

Theorem 1

For $a \notin X,Y$, the search space $\{X{:}aY\}$ can be partitioned into $\{Xa{:}Y\}$ and $\{X{:}Y\}$ by item a (i.e., $\{X{:}aY\}=\{Xa{:}Y\}+\{X{:}Y\}$).

Proof

It follows from the fact that each itemset of $\{X{:}aY\}$ either contains a (i.e., $\{Xa{:}Y\}$) or does not contain a (i.e., $\{X{:}Y\}$). For example, we have $\{b{:}cd\}=\{bc{:}d\}+\{b{:}d\}$.

Theorem 2

- **Partition Search Space:** Let $a_1, a_2, ..., a_k$ be distinct items and $a_1a_2...a_k$Y be an itemset; the search space of $\{X{:}\ a_1a_2...a_k\text{Y}\}$ can be partitioned into

$$\sum_{i=1}^{k}\{Xa_i : a_{i+1}\dots a_kY\}+\{X : Y\}, \text{ where } a_i \notin X, Y.$$

Proof

It follows by partitioning the search space via items $a_1, a_2, ..., a_k$ sequentially as in theorem 1.

For example, we have $\{b{:}cd\}=\{bc{:}d\}+\{bd{:}\}+\{b{:}\}$ and $\{a{:}bcde\}=\{ab{:}cde\}+\{ac{:}de\}+\{a{:}de\}$.

Let $\{X{:}Y\}$ be a search space and Z be a known frequent itemset. Since Z is frequent, all subsets of Z will be frequent (i.e., every itemset of $\{{:}Z\}$ is frequent). Theorem 3 shows how to prune the space $\{X{:}Y\}$ by Z.

Theorem 3

- **Pruning Search Space:** If Z does not contain the head X, the space $\{X{:}Y\}$ cannot be pruned by Z (i.e., $\{X{:}Y\}-\{{:}Z\}=\{X{:}Y\}$). Otherwise, the space can be pruned as

$$\{X{:}Y\}-\{{:}Z\}=\sum_{i=1}^{k}\{Xa_i : a_{i+1}...a_k(Y\cap Z)\}, \ a_1a_2...a_k=Y-Z.$$

Proof

If Z does not contain X, no itemset in $\{X{:}Y\}$ is subsumed by Z. Therefore, knowing that Z is frequent, we cannot prune any part of the search space $\{X{:}Y\}$.

Otherwise, when X is a subset of Z, we have $\{X{:}Y\}=\sum_{i=1}^{k}\{Xa_i : a_{i+1}...a_kV\}+X:V$, where $V=\text{Y}\cap\text{Z}$. The head in the first part is Xa_i where a_i is a member of Y-Z. Since Z does not contain a_i, the first part cannot be pruned by Z. For the second part, we have $\{X{:}V\}-\{{:}Z\}=\{X{:}V\}-\{X{:}(Z-X)\}$. Since $\text{X}\cap\text{Y}=\phi$, we have $V\subseteq Z-X$. Therefore, $\{X{:}V\}$ can be pruned away entirely.

For example, we have $\{{:}bcde\}-\{{:}abcd\}=\{{:}bcde\}-\{{:}bcd\}=\{e{:}bcd\}$. Here, a is irrelevant and is removed in the first step. Another example is $\{e{:}bcd\}-\{{:}abe\}=\{e{:}bcd\}-\{{:}be\}=\{e{:}bcd\}-\{e{:}b\}=\{ec{:}bd\}+\{ed{:}b\}$.

Pattern Decomposition

Given a known frequent itemset Z, we are able to decompose the search space of a transaction $N=X{:}Y$ to $N'=Z{:}Y'$, if X is a subset of Z, where Y' is the set of items that appears in Y but not in Z, denoted by PD($N=X{:}Y|$Z)= Z:Y'.

For example, if we know that an itemset abc is frequent, we can decompose a transaction $N=a{:}bcd$ into $N'=abc{:}d$; that is, PD($a{:}bcd|abc$)=$abc{:}d$.

Given a known infrequent itemset Z, we also can decompose the search space of a transaction $N=X{:}Y$. For simplicity, we use three examples to show the decomposition by known infrequent itemsets and leave out its formal mathematic formula in general cases. Interested readers can refer to Zou, Chu, and Lu (2002) for details. For example, if $N=d{:}abcef$ and a known infrequent itemsets, then we have:

- For infrequent 1-itemset ~a, PD($d{:}abcef|$~a) = $d{:}bcef$ by dropping a from its tail.
- For infrequent 2-itemset ~ab, PD($d{:}abcef|$~ab) = d:$bcef$+$da{:}cef$ by excluding ab.