Efficient Mining of Partial Periodic Patterns in Time Series Database

Jiawei Han, Guozhu Dong, Yiwen Yin

Presented by ***Xin Tu***

Abstract

- Previous studies on periodicity search mainly consider finding **full periodic patterns**,*where every point in time contributes to the periodic-ity*
- Now we will discuss **efficient mining of partial periodic patterns**, by exploring some interesting properties related to partial periodicity

Outline

- Definitions related to partial periodicity
- Algorithms for mining partial periodicity in regard to both single and multiple periods
- Implementation of the max-subpattern tree
- Comparison of the performance of the algorithms above
- Conclusion

Outline

- **Definitions related to partial periodicity**
- Algorithms for mining partial periodicity in regard to both single and multiple periods
- Implementation of the max-subpattern tree
- Comparison of the performance of the algorithms above
- Conclusion

Related Concepts

- For each **time instant** i, let D_i be a set of features of dataset at that instant, the **time series** of features is represented as $S=D_1, D_2,\ldots, D_n$
- Define a **pattern** $s=s_1 \ldots s_p$ as a nonempty sequence over $(2^L - \{\phi\}) \cup \{*\}$
- $|s|$ denotes the length of s, called the **period** of s
- A **subpattern** of a pattern $s=s_1 \ldots s_p$ is a pattern $s'=s'_1 \ldots s'_p$ such that s and s' have the same length, and $s'_i \subseteq s_i$ for every position i where $s'_i \neq *$

Problem Definition

- The **frequency_count** and **confidence** of a pattern s in a time series $S=D_1, D_2,\ldots, D_n$ are defined as

$frequency\_count(s) = |\{i \mid 0 \le i < m,$ *and the string s is true in* $D_{i|s|+1},\ldots, D_{i|s|+|s|}\}|$

$$conf(s) = \frac{frequency\_count(s)}{m}$$

m is the maximum number of periods of length $|s|$ contained in the time series (i.e., m is the positive integer such that $m|s| \le n < (m+1)|s|$)

Outline

- Definitions related to partial periodicity
- **Algorithms for mining partial periodicity in regard to both single and multiple periods**
- Implementation of the max-subpattern tree
- Comparison of the performance of the algorithms above
- Conclusion

Single-period Apriori Method

- **Apriori Property:** If one subset of an itemset is not frequent, then the itemset itself cannot be frequent. (This allows us to use frequent itemsets of size i as filters for candidate itemsets of size i+1)
- **Property 3.1 [Apriori on Periodicity]:** Each sub-pattern of a frequent pattern of period *p* is itself a frequent pattern of period *p*

Single-period Apriori Method

- **Algorithm 3.1:** Find all partial periodic patterns for a given period ***p*** satisfying a given confidence threshold ***min_conf*** in time-series ***S***, based on the Apriori property 3.1
  - Find F_1, the set of frequent **1-patterns** of period ***p***, by accu-mulating the frequency count for each 1-pattern in each whole period segment and selecting among them whose frequency count is no less than **min_conf x m**, where *m* is the maximum number of periods
  - Repeat the same procedure as the first step to find all frequent ***i*-patterns** of period ***p***, for ***i*** from **2** to ***p***, until the candidate frequent ***i*-pattern** set is empty

Concepts of Single-period Max-subpattern Hit Set Method

- **Candidate frequent max-pattern** $(C_{\max})$ is the maximal pattern which derive from F_1

  For example: $C_{\max} = a\{b_1, b_2\}cd*$
- A subpattern of $C_{\max}$ is **hit** in a period segment S_i of *S* if it is the maximal subpattern of $C_{\max}$ in S_i; the **hit set**, ***H***, of a time series *S* is the set of all hit subpatterns of $C_{\max}$ in *S*
- **Property 3.2** [The bound of hit set] The bound for the size of *H* is $|H| \le \min\{m, 2^{|F1|} - 1\}$, where *m* is the total number of periods in *S*

Single-period Max-subpattern Hit Set Method

- **Algorithm 3.2:** Find all the partial periodic patterns for a given period *p* in a time-series *S*, based on the max-subpattern hit-set, for a given ***min_conf*** threshold
  - Using **Step 1** of Algorithm 3.1 to find F_1 of period ***p***; form the candidate max-pattern $C_{\max}$ from F_1
  - Scan ***S*** once again; during the scan, for each period seg-ment, do: If there is no **max-subpattern**, then add it into the **hit set buffer**; otherwise, **add one** to the count of the max-subpattern
  - After the scan, derive the **frequent patterns** from the **hit set**; how to implement this procedure will be discussed later

Comparison between the Algorithms 3.1 and 3.2

- Scan
  - **Algorithm 3.1** requires to scan ***S*** up to ***p*** times in the worst case
  - **Algorithm 3.2** only requires to scan it **2** times
- Space
  - **Algorithm 3.1** need $2^{|F1|} - 1$
  - **Algorithm 3.2** need $\min\{m, 2^{|F1|} - 1\}$

Question

- Can we extend the idea of **Apriori** to computing partial periodicity **among different periods**, that is, to use the patterns of small periods ***p*** as filters for candidate patterns of periods of the form ***kp*** for an integer k>1?
- Then the most direct way is to repeatedly apply the **single-period** algorithm for **each period** in the range

Mining Partial Periodicity with Multiple Periods

- **Algorithm 3.3** [Looping over single period computation]: Find all the partial periodic patterns for a set of periods in a given range of interest, p_1, …, p_k, in the time-series ***S***, with the given min_conf threshold
  - Apply algorithm 3.2 on **each period** P_j in the range of interest (p_1, …, p_k)
- This algorithm require to scan the time-series *S* for **2** x ***k*** times, so when the number of periods ***k*** is large, we still need a good number of scans; how to improve it?

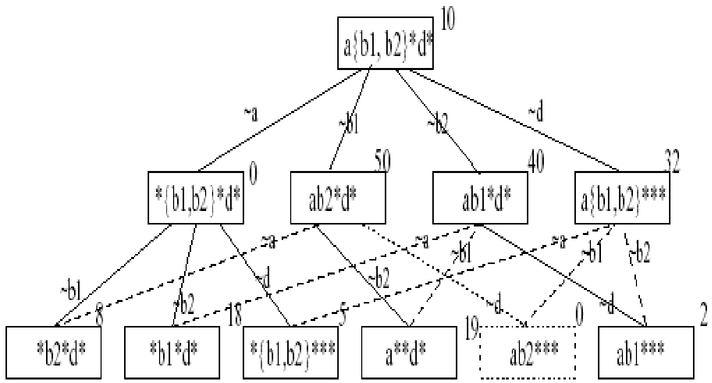
Mining Partial Periodicity with Multiple Periods

- **Algorithm 3.4** [Shared mining of multiple periods]: Shared mining of all the partial periodic patterns for a set of periods in a given *min_conf* threshold
  - For **all** periods p_j in the range of interest, scan ***S*** once first, then find $F_1(p_j)$ of period p_j, using the same step 1 as in Algorithm 3.1. For each set of frequent 1-patterns of period p_j, form the candidate max-pattern, $C_{\max}(p_j)$, from $F_1(p_j)$
  - For all periods p_j in the range of interest, scan ***S*** once again, then do the same step 2 as Algorithm 3.2
- The total number of time-series scans is 2 for multiple periods; but it will require more space in the processing

Outline

- Definitions related to partial periodicity
- Algorithms for mining partial periodicity in regard to both single and multiple periods
- **Implementation of the max-subpattern tree**
- Comparison of the performance of the algorithms above
- Conclusion

Implementation of The Max-Subpattern Tree



Build a Max-Subpattern Tree

- Take the candidate max-pattern $C_{\max}$ as the root node, where each subpattern of $C_{\max}$ with one non-$*$ letter missing is a direct child node of the root
- Each node has a “count” field (registers the number of hits of the current node), a parent link (nil for root), and a set of child links; each child link points a child and is associated with a corresponding missing letter.
- A node with only 2 non-$*$ letters will not have any children

Insertion in the Max-sp tree

- **Algorithm 4.1:** Insert a max-sp *w* found during the scan of *S* into the max-sp tree *T*
  - Starting from the root of the tree, find the corresponding node by checking the missing non-$*$ letters in order
  - If the node w is found, increase its count by 1. Otherwise, create a new node *w* (with count 1) and its missing ancestor nodes (only those on the path to *w*, with count 0), and insert them into the corresponding places of the tree

    For example 4.1

Derivation of Frequent Patterns from Max-sp tree

- **Algorithm 4.2:** The derivation of the frequent *k*-patterns for all *k*, given a max-sp tree *T*, by an Apriori-like technique
  - The set of frequent 1-patterns F_1 is derived in the first scan of Algorithm 3.2
  - After the second scan of Algorithm 3.2, we get the max-sp tree *T*. The set of frequent *k*-patterns ($k > 1$) is derived by for i:= 2 to $|F_1|$ do {
    - Derive candidate patterns with L-length i from frequent patterns with L-length (i-1)
    - Scan tree T to find frequent counts of these candidate patterns and eliminate the non-frequent ones.

    Frequency count=count of node+counts of reachable ancestors

Outline

- Definitions related to partial periodicity
- Algorithms for mining partial periodicity in regard to both single and multiple periods
- Implementation of the max-subpattern tree
- **Comparison of the performance of the algorithms above**
- Conclusion

Performance of The Algorithms

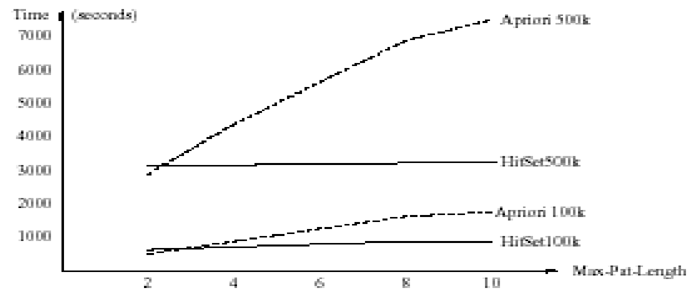


Figure 2. Performance gain when MAX-PAT-LENGTH increases: $p = 50$, $|F_1| = 12$.

Performance of The Algorithms

- The running time of max-sp hit-set is almost constant for the length of the time series being 100,000 and the other being 500,000; Apriori is almost linear in the same conditions
- No matter for Mining partial periodicity with single or multiple periods, max-sp hit-set requires much less times of scans

Outline

- Definitions related to partial periodicity
- Algorithms for mining partial periodicity in regard to both single and multiple periods
- Implementation of the max-subpattern tree
- Comparison of the performance of the algorithms above
- **Conclusion**

Conclusion

- By exploring several interesting properties Apriori property, the max-sp hit-set property, and shared mining of multiple periods, a set of partial periodicity mining algorithms are proposed. The study shows that the max-subpattern hit-set method offers excellent performance

Thank You