Pattern-Growth Methods for Frequent Pattern Mining

OPEN ACCESS

Volume : 6

Special Issue : 1

Month : September

Year: 2018

ISSN: 2321-788X

Impact Factor: 3.025

Citation:
Abirami, S. (2018). Pattern-Growth Methods for Frequent Pattern Mining. *Shanlax International Journal of Arts, Science and Humanities*, 6(S1), pp. 76–81

DOI:
https://doi.org/10.5281/zenodo.1410989

S.Abirami
M.Phil. Research Scholar, Department of Computer Science
Morappur Kongu College of Arts &Science

Abstract

Mining frequent patterns from large databases play an essential role in many data mining tasks and has broad applications. Most of the previously proposed methods adopt Apriori-like candidate-generation-and-test approaches. However, those methods may encounter serious challenges when mining datasets with prolific patterns and long patterns.

In this work, to develop a class of novel and efficient pattern-growth methods for mining various frequent patterns from large databases. Pattern-growth methods adopt a divide-and-conquer approach to decompose both the mining tasks and the databases. Then, they use a pattern fragment growth method to avoid the costly candidate generation-and-test processing completely. The effective data structures are proposed to compress crucial information about frequent patterns and avoid expensive, repeated database scans. A comprehensive performance study shows that pattern-growth methods, FP-growth and H-mine, are efficient and scalable. They are faster than some recently reported new frequent pattern mining methods.

Interestingly, pattern growth methods are not only efficient but also effective. With pattern growth methods, many interesting patterns can also be mined efficiently, such as patterns with some tough non-anti-monotonic constraints and sequential patterns. These techniques have strong implications to many other data mining tasks.

Introduction

"The universe is full of magical things patiently waiting for our wits to grow sharper." Data mining is to find valid, novel, potentially useful, and ultimately understandable patterns in data. In general, there are many kinds of patterns (knowledge) that can be discovered from data. For example, association rules can be mined for market basket analysis; classification rules can be found for accurate classifiers, clusters and outliers can be identified for customer relation management.

Frequent pattern mining plays an essential role in many data mining tasks, such as mining association rules, correlations, causality, sequential patterns, episodes, multi-dimensional patterns, max-patterns partial periodicity, and emerging patterns. Frequent pattern mining techniques can also be extended to solve many other problems, such as iceberg-cube computation and classification. Thus, effective and effcient frequent pattern mining is an important and interesting research problem.

Motivation

Most of the previous studies on frequent pattern mining adopt an Apriori-like approach, which is based on an anti-monotone Apriori heuristic if any length k pattern is not frequent in the database, its length (k + 1) super-pattern can never be frequent. The essential idea is to iteratively generate the set of candidate patterns of length (k +1) from the set of frequent patterns of length k (for k ¸ 1), and check their corresponding occurrence frequencies in the database.

The Apriori heuristic achieves good performance gain by (possibly significantly) reducing the size of candidate sets. However, in situations with prolific frequent patterns, long patterns, or quite low minimum support thresholds, an Apriori-like algorithm may still suffer from the following two nontrivial costs:

1. It is costly to handle a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 length-2 candidates and test their occurrence frequencies.
2. It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for long mining patterns.

Contributions

In this thesis, to study the problem of efficient and effective frequent pattern mining, as well as some of its extensions and applications. In particular, we make the following contributions.

- Systematically develop a pattern-growth method for frequent pattern mining. A novel algorithm, FP-growth, is proposed for efficiently mining frequent patterns from large dense datasets. Furthermore, to achieve efficient frequent pattern mining in various situations, we design H-mine, which is highly scalable and space preserving for very large databases.
- As an inherent problem, frequent pattern mining may return too many patterns. Constraint-based data mining is an important approach to solve the problem of effective data mining. Study the problem of constraint-based frequent pattern mining using pattern-growth methods. Our study shows that pattern-growth methods can push constraints deeper into the mining process, even including such constraints using aggregate AVG( ) and SUM( ), which other methods cannot handle.
- To extend the pattern-growth method to allow the mining of sequential patterns. Our study shows that pattern-growth methods are more effcient in mining large sequence databases. Interesting techniques are developed to solve the sequential pattern mining problem effectively.

Problem Definition and Related Work

In this chapter, first define the problem of frequent pattern mining, then we revisit the Apriori heuristic and algorithm. Several improvements of the Apriori algorithm are also discussed.

Frequent Pattern Mining Problem

The frequent pattern mining problem was first introduced by R. Agrawal, et al. in as mining association rules between sets of items.

Let I = {i1; : : : ; im} be a set of items. An itemset X ≤ I is a subset of items. Hereafter, we write itemsets as X = ij1 ... ijn, i.e., omitting set brackets. Particularly, an itemset with l items is called an l-itemset.

A transaction T = (tid, X) is a tuple where tid is a transaction-id and X is an itemset. A transaction T = (tid, X) is said to contain itemset Y if Y ≤ X.

A transaction database T DB is a set of transactions. The support of an itemset X in transaction

database T DB denoted as Supt DB(X) or sup(X), is the number of transactions in T DB containing X, i.e.,

sup(X) = {(tid, Y) ((tid, Y) ϵT DB) ^ (X ≤ Y)

Problem statement. Given a user-specified support threshold min sup, X is called a frequent itemset or frequent pattern if sup(X) ¸ min sup. The problem of frequent mining itemsets is to find the complete set of frequent itemsets in a transaction database T DB with respect to a given support threshold min sup.

Association rules can be derived from frequent patterns. An association rule is an implication of the form X =) Y, where X and Y are item sets and X ∩Y = φ. The rule X =) Y has support s in a transaction database T DB if Supt DB(X [Y) = s. The rules =) Y holds in the transaction database T DB with confidence c where c = sup(X U Y).

sup(X)

Apriori Heuristic and Algorithm

To achieve effcient mining frequent patterns, an anti-monotonic property of frequent item-sets, called the Apriori heuristic.

Theorem 2.1 (Apriori) Any superset of an infrequent itemset cannot be frequent. In other words, every subset of a frequent itemset must be frequent.

Proof. To prove the theorem, we only need to show sup(X) ≤sup(Y) if X ≥ Y.

Given a transaction database T DB. Let X and Y be two itemsets such that X≥Y. For each transaction T containing itemset X, T also contains Y, which is a subset of X. Thus, we have sup(X) ≤sup(Y).

The Apriori heuristic can prune candidates dramatically. Based on this property, a fast, frequent itemset mining algorithm, called Apriori, was developed. It is illustrated in the following example.

Example 2.1 (Apriori) Let the transaction database, T DB, be Table 2.1 and the mini-mum support threshold be 3.

tid	Itemset	(Ordered) Frequent Items
100	f; a; c; d; g; i; m; p	a; c; f; m; p
200	a; b; c; f; l; m; o	a; b; c; f; m
300	b; f; h; j; o	b; f
400	b; c; k; s; p	b; c; p
500	a; f; c; e; l; p; m; n	a; c; f; m; p

Table 2.1: A transaction database T DB.

Apriori finds the complete set of frequent itemsets as follows.

1. Scan T DB once to find frequent items, i.e., items appearing in at least three transactions. They are a, b, c, f, m, p. Each of these six items forms a length-1 frequent itemset. Let L1 be the complete set of length-1 frequent itemsets.
2. The set of length-2 candidates, denoted as C2, is generated from L1. Here, we use the Apriori heuristic to prune the candidates. Only those candidates that consist of frequent subsets can be potentially frequent. An itemset xy ϵC2 if and only if x,yϵL1.Thus,C2={ab,ac,..... ap,bc....,mp}.
3. Scan T DB once more to count the support of each itemset in C2. The itemsets in C2 passing the support threshold form the length-2 frequent itemsets, L2. In this example, L2 contains itemsets ac, af, am, cf, cm, and fm.

4. Then, we form the set of length-3 candidates. Only those length-3 itemsets for which every length-2 sub-itemset is in L2 are qualified as candidates. For example, ACF is a length-3 candidate since ac, af and cf are all in L2.

One scan of T DB identifies the subset of length-3 candidates passing the support threshold and form the set L3 of length-3 frequent itemsets. A similar process goes on until no candidate can be derived or no candidate is frequent.

One can verify that the above process eventually finds the complete set of frequent itemsets in the database T DB.

The Apriori algorithm is presented as follows.

Algorithm 1 (Apriori)

Input: transaction database TDB and support threshold min_sup

Output: the complete set of frequent patterns in TDB with respect to support threshold min_sup
Method:

1. scan transaction database TDB once to find L1, the set of frequent 1-itemsets;
2. for (k = 2; Lk¡-1 ≠ φ;k + +) do
   (a) Generate Ck, the set of length-k candidates. A k-itemset X is in Ck if and only if every length-(k - 1) subset of X is in Lk-1;
   (b) if Ck = φ then go to Step 3;
   (c) scan transaction database T DB once to count the support for every itemset in Ck;
   (d) Lk = {X(X ε Ck) ^ (sup(X) ≥min _sup)}
3. return Uki=1 Li

Improvements over Apriori

In the past several years, many improvements over the Apriori algorithm have been proposed. In this section, we review some important proposals.

The major bottlenecks in Apriori algorithm are in three aspects.

- The Apriori algorithm needs to scan the database multiple times. When mining a huge database, multiple database scans are costly. One feasible strategy to improve the effciency of Apriori algorithm is to reduce the number of database scans.
- The Apriori algorithm has to generate a huge number of candidates. Storing and counting these candidates are tedious. To attack this problem, some studies focus on reducing the number of candidates.
- One dominant operation in the Apriori algorithm is support counting. To speed up the Apriori-like algorithms, some facilities are proposed.

Discussion

Pattern-growth methods for effcient and effective frequent pattern mining. In this chapter, first, summarize the major characteristics of pattern-growth methods, then discuss some interesting extensions and applications of pattern-growth methods.

Characteristics of Pattern-growth Methods

To developed a new class of pattern-growth methods for effective and effcient data mining. Summarize the major characteristics of pattern-growth methods here.

- Pattern-growth methods adopt a divide-and-conquer methodology and partition both the data sets and patterns into subsets recursively. In general, data mining has to search a very huge space. The divide-and-conquer methodology enables the search algorithms to focus on reduced subsets of goals within much smaller sub-spaces. That makes sharper pruning feasible.
- Pattern-growth methods avoid candidate-generation-and-test. Instead, pattern-growth methods take the patterns found as seeds and explore extensions of current patterns. The benefits are from two aspects. On the one hand, pattern-growth methods search much less than candidate-generation-and-test methods do, since the number of candidates could be huge. On the other hand, pattern-growth methods avoid most of the expensive pattern matching operations. Instead, they search for frequent local items, which is much cheaper.
- Pattern-growth methods employ effective data structures to fully utilize the available space. For example, both FP-tree and H-block try to fully use the available memory and reduce irrelevant information. With highly condensed and well-indexed data structures, the search space is presented in a well-organized way so that the pattern-growth search can be much more effcient.

Extensions and Applications of Pattern-growth Methods

That pattern-growth methods are effective and effcient in frequent pattern mining. Interestingly and surprisingly, pattern-growth methods are also applicable to mining other kinds of knowledge and solving some other interesting data processing problems. In this section, we discuss some examples.

Mining Closed Association Rules

Association mining may often derive an undesirably large set of frequent itemsets and association rules. There is an interesting alternative, proposed recently by Pasquier, et al. [PBTL99]: instead of mining the complete set of frequent itemsets and their associations, association mining only needs to find frequent closed itemsets and their corresponding rules. An important implication is that mining frequent closed itemsets has the same power as mining the complete set of frequent itemsets, but it substantially reduces the redundant rules generated and increases both effciency and effectiveness of mining.

Conclusion

As our world is now in its information era, a huge amount of data is accumulated every day. A real universal challenge is to find actionable knowledge from a large amount of data. Data mining is an emerging research direction to meet this challenge. Many kinds of knowledge (patterns) can be mined from various data. In this thesis, we focus on the problem of frequent mining patterns effciently and effectively and develop a new class of pattern-growth methods.

A comprehensive performance study shows that pattern-growth methods, FP-growth and H-mine, are efficient and scalable. They are faster than some recently reported new frequent pattern mining methods. Pattern growth methods are not only efficient but also effective. With pattern growth methods, many interesting patterns can also be mined efficiently, such as patterns with some tough non-anti-monotonic constraints and sequential patterns. These techniques have strong implications to many other data mining tasks.

Final Thoughts

"Discovery consists of seeing what everybody has seen and thinking what nobody has thought." Data mining is towards an effective and effcient tool for discovery. By mining, we can see the patterns hidden behind the data more accurately, more systematically and more effciently. However, it is the data miner's responsibility to distinguish the gold from the dust.

"Every science begins as philosophy and ends as art." So does data mining.

References

Agarwal, R, Aggarwal, C., & Prasad, V. V. V., (2000), "A tree projection algorithm for the generation of frequent itemsets." *In Journal of Parallel and Distributed Computing (Special Issue on High-Performance Data Mining).*

Agrawal, R, Imielinski, T., & Swami, A, (1993), "Mining association rules between sets of items in large databases". In Proc. 1993 ACM-SIGMOD Int.Conf. Management of Data (SIGMOD'93), pages 207–216, Washington, DC.

Agrawal, R., & Srikant, R, (1994), "Fast algorithms for mining association rules". In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94), pages 487–499, San- tiago, Chile.

Agrawal, R.,& Srikant, R, (1995), "Mining sequential patterns." In Proc. 1995 Int. Conf. Data Engineering (ICDE'95), pages 3–14, Taipei, Taiwan.

Bayardo, R. J., (1998)"Effciently is mining long patterns from databases". In Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), pages 85–93, Seattle, WA.

Bayardo, R. J., Agrawal, R., & Gunopulos, D, (1999), "Constraint-based rule mining on large, dense data sets." In Proc. 1999 Int. Conf. Data Engineering (ICDE'99), Sydney, Australia.

Beyer, K., & Ramakrishnan, R, (1999), "Bottom-up computation of sparse and ice-berg cubes". In Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'99), pages 359–370, Philadelphia, PA.

Brain, D, Motwani, R.,& Silverstein, C, (1997), "Beyond market basket: Generalizing association rules to correlations." In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'97), pages 265–276, Tucson, Arizona.

Han, J, Pei, J, Mortazavi-Asl, B, Chen, Q, Dayal, U., & M.-C. Hsu, (2000), "FreeSpan: Frequent pattern-projected sequential pattern mining". In Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00), pages 355–359, Boston, MA.

Han, J, Pei, J., & Yin. Y, (2000), "Mining frequent patterns without candidate generation". In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD'00), pages 1–12, Dallas, TX.

Web Sources

http://www.ijcsiet.com/pdf/01072013-003.pdf
https://www.cs.sfu.ca/~jpei/publications/jianpei_thesis.pdf
http://hanj.cs.illinois.edu/pdf/dami04_fptree.pdf
https://www.cs.sfu.ca/~jpei/publications/jianpei_thesis.pdf
http://shodhganga.inflibnet.ac.in/bitstream/10603/4332/9/09_chapter%204.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.649.8476&rep=rep1&type=pdf
http://www.aun.edu.eg/journal_files/186_J_1152.pdf