Chapter 5: Mining Frequent Patterns, Association and Correlations

Dong-Kyu Chae

PI of the Data Intelligence Lab @HYU
Department of Computer Science & Data Science
Hanyang University

Data
Intelligence
LAB

Contents

- Basic concepts and a road map
- Efficient and scalable frequent pattern (itemset) mining methods
- Mining association rules
- From association mining to correlation analysis
- Constraint-based frequent pattern and association mining
- Summary

Frequent Pattern Mining

- **Frequent pattern:** a pattern (a set of co-purchased items, subsequences, substructures, etc.) that occurs frequently in a data set
  Data types :
  Sequential data   graph data
- First proposed in 1993 in the context of frequent itemsets and association rule mining
- **Motivation: Finding inherent patterns in data**
  - **What products were often purchased together? (this will be our main example)** — Beer and diapers   item set data
  - What are the subsequent purchases after buying a digital camera?
    Sequential data
  - What kinds of DNA are sensitive to this new drug?
    Graph data
- **Applications**
  - Basket data analysis, DNA sequence analysis

Basic Concepts

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

2^6-1 개 가능 (combination) ← 6 items (A-F)

- **Itemset X** = $\{x_1, \ldots, x_k\}$

  각 itemset의 frequency count → 특정수보다 높으면 그 itemset은 frequent pattern

  - Frequent pattern is defined on an itemset

  $X = \{A, D\} : 3$

- **Association rules** $X \rightarrow Y$
  - It is defined on two itemsets X and Y, where X U Y must be a frequent pattern.

  Association rules can be originated from frequent pattern Including more than 2 item sets

- **Support** and **Confidence:**
  - **Support**, *s*, is probability (or, frequency) that a transaction contains X. → $\frac{3}{5} = 60\%$
    - **Minimum support**: a threshold that decides whether X is a frequent pattern or not, based on its support

    % Integer

  - **Confidence**, *c*, conditional probability that a transaction having X also contains *Y*
    - **Minimum confidence:** it is also a threshold

  $X = \{A, D\}$ $Y = \{B\}$

  $X \rightarrow Y : \frac{X \cup Y}{X} = \frac{1}{3} = 33\%$ ↓ < $conf_{min}$ 이므로 Association rules 아님

Let sup_{min} = 50%, $conf_{min}$ = 50%, then:

- **Q: Find all freqent patterns.** **A:** {A:3, B:3, D:4, E:3, AD:3}
- **Q: Find all association rules.** **A:**

size1) {A} {B} ~~{C}~~ {D} {E} ~~{F}~~

{A}	{B}	~~{C}~~	{D}	{E}	~~{F}~~
3/5	3/5	~~2/5~~	4/5	3/5	~~2/5~~
60%	60%	~~40%~~	80%	60%	~~40%~~

size2) {AB} {AC} ··· {AD} $\frac{3}{5}$

$\frac{3}{5}$ {A}, {D}은 각각 & {AD}는 frequent pattern

$A \rightarrow D$ (60%, 100%) $\frac{A \cup D}{A} = \frac{3}{3}$

$D \rightarrow A$ (60%, 75%) $\frac{D \cup A}{D} = \frac{3}{4}$

sup conf

⇐ association rules 조건 만족

Data Intelligence LAB

Closed Patterns and Max-Patterns

⇒ Can Ignore huge amount of useless pattern and only focus on the most meaningful, important patterns

- A long pattern contains *too many* number of **sub-patterns,** e.g., $\{a_1, \ldots, a_{100}\}$ contains $\mathbf{2^{100} - 1}$ sub-patterns!

  이미 frequent pattern이면 subsets도 frequent pattern

- Solution: Mine closed patterns and max-patterns instead, which can be representatives of those sub-patterns
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ⊃ X, *with* ***the same support*** as X

  X is the superset

- An itemset X is a max-pattern if X is frequent and there exists no **frequent** super-pattern Y ⊃ X

  $sup_{min} = 7$ → X, Y, Z : frequent pattern
  Z : Max pattern

- Closed pattern is a lossless compression of freq. patterns

  $sup_{min} = 9$ → X, Y : frequent pattern
  Y : Max pattern

  - Reducing the # of redundant patterns and rules

Z, Y는 동일하게 중요
둘다 closed pattern

Support (frequency)

X = {a, b, c} : 10

Y = {a, b, c, d} : 10 (times - Integer)

Y is superset
Y is closed pattern

Z = {a, b, c, d, e} : 8

Z is Superset of Y
But Support 감소
→ Closed pattern 관점에서 Z는 Y를 표현할수X

⇒ which one is more important? Y! Becuz Y is Including X, So Y is more informative

Closed Patterns and Max-Patterns

- **Exercise. DB = {<a_1, …, a_{100}>, < a_1, …, a_{50}>} including only two transactions and 100 items**

  transaction 1 transac. 2

  - Let Min_sup = 1.

    Integer 쓰면 Frequency 나타냄
    probability 아님

- **Questions:**
  - How many frequent patterns are there?

    all the sigle items is frequent
    so, all possible combinations of items are frequent pattern

    - $2^{100} - 1$
  - What is the set of closed patterns? (write each one's support as well)
    - <a_1, …, a_{100}>: 1
    - < a_1, …, a_{50}>: 2
  - What is the set of max-patterns? (write each one's support as well)
    - <a_1, …, a_{100}>: 1

    Superset

Scalable Methods for Mining Frequent Patterns

- **The downward closure property of frequent patterns**
  - Any subset of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}** also frequent.
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}

If {a, b, c} is not frequent, there's no way that {a, b, c, d} is frequent

- **Scalable mining methods: Three major approaches**
  - **Apriori** (Agrawal & Srikant@VLDB' 94)
  - Freq. pattern growth (**FP-growth**, @SIGMOD' 00)
  - Vertical data format approach (**Charm**, @SDM' 02)

Apriori: A Candidate Generation-and-Test Approach

- **Apriori pruning principle:** If there is any itemset which is infrequent, its superset should not be generated/tested!

- **Method:**
  - Initially, scan DB once to get frequent 1-itemset
  - **Repeat with index [k]:**

    self-Joining Lk
    pruning (before candidate generation)

    - **Generate candidate** itemsets of length (k+1) from frequent itemsets of length k
    - **Test** the candidates against DB
    - **Terminate** when no frequent or candidate set can be generated

    Pruning after candidate generation
    – Supmin 보다 작으면 제거

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan →

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→ L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

← 2nd scan

Candidate

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

← Test

List

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

~~{A, B, C}~~ {B, C, E}
~~{A, C, E}~~

C_3

Itemset
{B, C, E}

3rd scan →

L_3

Itemset	sup
{B, C, E}	2

L_1 L_2 L_3
→ output

The Apriori Algorithm: Pseudo-code

Pseudo-code:

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items}; terminates when there's no additional candidate generation step k+1

for (k = 1; L_k !=∅; k++) **do begin**

C_{k+1} = candidates generated from L_k

test **for each** transaction t in database **do**

increment the count of all candidates in C_{k+1} that are contained in t

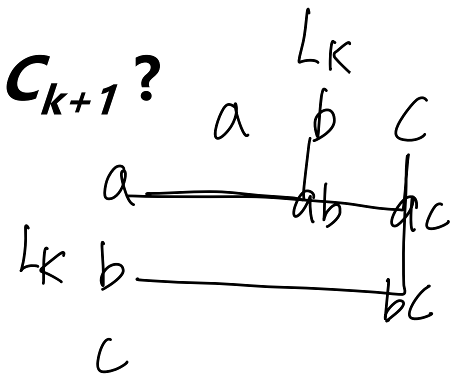
L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$ $L_1 \cup L_2 \cup L_3 \cdots \cup L_k$: Result of frequent pattern mining

Important Details of Apriori

- **How to generate candidates, from L_k to C_{k+1} ?**
  - Step 1: self-joining L_k
  - Step 2: pruning : Before candidate generation
    → infrequent item set 포함하면제거



- **Example of candidate generation via self-joining and pruning**
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - ***abcd*** can be a candidate from ***abc***, ***abd***, ***bcd***, all of which are frequent
  - Pruning:
    - ***acde*** cannot be included because ***ade*** is not in L_3 , i.e., not frequent!
  - C_4={*abcd*}

Challenges of Frequent Pattern Mining

- **Challenges**
  - Multiple scans of a transaction database (about k times)
  - Huge number of candidates
  - Tedious workload of support counting for candidates

- **General ideas of improving efficiency of frequent pattern mining**
  - Reduce the number of transaction database scans
  - Reduce the number of candidates
  - Facilitate support counting of candidates

Thank You