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

Frequent Pattern Mining

Frequently occuring items in a DB

- **Frequent pattern:** a pattern (a set of co-purchased items, subsequences, substructures, etc.) that occurs frequently in a data set
- 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
  - What are the subsequent purchases after buying a digital camera?

    SD memory
  - What kinds of DNA are sensitive to this new drug?
- **Applications**
  - Basket data analysis, DNA sequence analysis

DB 6 items 5 transactions

Basic Concepts

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

6 items -> 2^6 - 1 possible combinations

- **Itemset X** = $\{x_1, \ldots, x_k\}$
  - Frequent pattern is defined on an itemset
- **Association rules** $X \rightarrow Y$ If someone purchases X, then there's a good chance he'll buy Y
  - It is defined on two itemsets X and Y, where X U Y must be a frequent pattern.

Minimum confidence 를 넘어야 association rule이라고 부를 수 있음

- **Support** and **Confidence:** Computed on each item
  - **Support**, *s*, is probability (or, frequency) that a transaction contains X. Ex) x = {A, D} -> 3/5
    - **Minimum support:** a threshold that decides whether X is a frequent pattern or not, based on its support
  - **Confidence**, *c*, conditional probability that a transaction having X also contains *Y*
    - **Minimum confidence:** it is also a threshold Ex) X = {A, D} Y = {B} -> 1/3

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

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

size 1 item sets size 2 item sets …

- Q: Find all association rules. **A:**

$A \rightarrow D$ (60%, 100%)

$D \rightarrow A$ (60%, 75%)

sup *conf*

1) defined on two item sets A and D
2) A U D is frequent

Closed Patterns and Max-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! Too many patterns!!
  Frequent patterns
- 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
- An itemset X is a max-pattern if X is frequent and there exists no **frequent** super-pattern Y ⊃ X
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of redundant patterns and rules

support
X = {a,b,c} : 10
Y = {a,b,c,d}: 10

Y is more informative (includes X)
-> Y is closed.

X = {a,b,c} : 10
Y = {a,b,c,d} : 10
Z = {a,b,c,d,e} : 8

Y is still closed
because support of Z is 8.

Sup-min : 8
Y is not a max-pattern
because Z is frequent.

Sup-min : 9
Y is a max-pattern
because Z is not frequent.

Closed Patterns and Max-Patterns

- **Exercise. DB = $\{\langle a_1, \ldots, a_{100}\rangle, \langle a_1, \ldots, a_{50}\rangle\}$ including only two transactions and 100 items**
  - Let Min_sup = 1.

X = {a1 ··· a49} : 2
Y = {a1 ··· a50} : 2
-> Only Y is closed.

Z = {a1 ··· a51} : 1
-> Y is still closed.

Q = {a1 ···a100}
-> Y and Q is closed

- **Questions:**
  - How many frequent patterns are there?
    - $2^{100} - 1$
  - What is the set of closed patterns? (write each one's support as well)
    - $\langle a_1, \ldots, a_{100}\rangle$: 1
    - $\langle a_1, \ldots, a_{50}\rangle$: 2
  - What is the set of max-patterns? (write each one's support as well)
    - $\langle a_1, \ldots, a_{100}\rangle$: 1

X = {a1 ··· a50} : 2
Y = {a1 ··· a100} : 1

-> Y is a max-pattern.
(Y is frequent and is a superset of X)

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}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}

  If {abc} is not frequent, {abcd} cannot be 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]:**
    - **Generate candidate** itemsets of **Size** (k+1) from frequent itemsets of length k
    - **Test** the candidates against DB
    - **Terminate** when no frequent or candidate set can be generated

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

C_2

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

L_2

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

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

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};

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

  C_{k+1} = candidates generated from L_k

  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 $\cup_k L_k$

Important Details of Apriori

- **How to generate candidates, from L_k to C_{k+1} ?**
  - Step 1: self-joining L_k
  - Step 2: pruning

- **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