CS570 Introduction to Data Mining

Frequent Pattern Mining and Association Analysis

Cengiz Gunay

Partial slide credits: Li Xiong, Jiawei Han and Micheline Kamber
George Kollios
Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
What Is Frequent Pattern Analysis?

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - Frequent sequential pattern
  - Frequent structured pattern

- **Motivation**: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

- **Applications**
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
Frequent Itemset Mining

- Frequent itemset mining: frequent set of items in a transaction data set

- First proposed by Agrawal, Imielinski, and Swami in SIGMOD 1993
  - SIGMOD Test of Time Award 2003

  “This paper started a field of research. In addition to containing an innovative algorithm, its subject matter brought data mining to the attention of the database community … even led several years ago to an IBM commercial, featuring supermodels, that touted the importance of work such as that contained in this paper. ”


Basic Concepts: Frequent Patterns and Association Rules

- Itemset: $X = \{x_1, \ldots, x_k\}$ (k-itemset)
- Frequent itemset: $X$ with minimum support count
  - Support count (absolute support): count of transactions containing $X$
- Association rule: $A \rightarrow B$ with minimum support and confidence
  - Support: probability that a transaction contains $A \cup B$
    - $s = P(A \cup B)$
  - Confidence: conditional probability that a transaction having $A$ also contains $B$
    - $c = P(B \mid A)$

Association rule mining process
- Find all frequent patterns (more costly)
- Generate strong association rules

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, D</td>
</tr>
<tr>
<td>20</td>
<td>A, C, D</td>
</tr>
<tr>
<td>30</td>
<td>A, D, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
<tr>
<td>50</td>
<td>B, C, D, E, F</td>
</tr>
</tbody>
</table>
Illustration of Frequent Itemsets and Association Rules

- Frequent itemsets (minimum support count = 3) ?
  \{A:3, B:3, D:4, E:3, AD:3\}
- Association rules (minimum support = 50%, minimum confidence = 50%) ?
  \[ A \rightarrow D \ (60\%, \ 100\%) \]
  \[ D \rightarrow A \ (60\%, \ 75\%) \]
  \[ A \rightarrow C \ ? \]

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, D</td>
</tr>
<tr>
<td>20</td>
<td>A, C, D</td>
</tr>
<tr>
<td>30</td>
<td>A, D, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
<tr>
<td>50</td>
<td>B, C, D, E, F</td>
</tr>
</tbody>
</table>
Mining Frequent Patterns, Association and Correlations

- Basic concepts
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary
Scalable Methods for Mining Frequent Patterns

- Scalable mining methods for frequent patterns
  - Apriori (Agrawal & Srikant@VLDB’94) and variations
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD’00)
  - Algorithms using vertical format
- Closed and maximal patterns and their mining methods
- FIMI Workshop and implementation repository
Apriori – Apriori Property

- Apriori: use *prior knowledge* to reduce search by pruning unnecessary subsets
- The apriori property of frequent patterns
  - Any nonempty subset of a frequent itemset must be frequent
    - If \{beer, diaper, nuts\} is frequent, so is \{beer, diaper\}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Bottom up search strategy
Apriori: Level-Wise Search Method

- Level-wise search method:
  - Initially, scan DB once to get frequent 1-itemset (L1) with minimum support
  - Generate length (k+1) candidate itemsets from length k frequent itemsets (e.g., find L2 from L1, etc.)
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
The Apriori Algorithm

- **Pseudo-code:**
  $C_k$: Candidate $k$-itemset
  $L_k$: frequent $k$-itemset

  $L_1 =$ frequent 1-itemsets

  for $(k = 2; L_{k-1} \neq \emptyset; k++)$
    $C_k =$ generate candidate set from $L_{k-1}$
    for each transaction $t$ in database
      find all candidates in $C_k$ that are subset of $t$
      increment their count;
    $L_k =$ candidates in $C_k$ with min_support
  return $\cup_k L_k$
The Apriori Algorithm—An Example

min\_support = 2

Transaction DB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

1st scan

$C_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{D}</td>
<td>1</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

$L_1$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

2nd scan

$C_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td>1</td>
</tr>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

$L_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{A, E}</td>
<td>1</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

3rd scan

$C_3$

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
</tr>
</tbody>
</table>

$L_3$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

$C_2$

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td></td>
</tr>
<tr>
<td>{A, C}</td>
<td></td>
</tr>
<tr>
<td>{A, E}</td>
<td></td>
</tr>
<tr>
<td>{B, C}</td>
<td></td>
</tr>
<tr>
<td>{B, E}</td>
<td></td>
</tr>
<tr>
<td>{C, E}</td>
<td></td>
</tr>
</tbody>
</table>

$C_3$

<table>
<thead>
<tr>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
</tr>
</tbody>
</table>
Important Details of Apriori

- How to generate candidate sets?
- How to count supports for candidate sets?
Candidate Set Generation

- Step 1: self-joining $L_{k-1}$: assuming items and itemsets are sorted in order, joinable only if the first $k-2$ items are in common

  \[
  \text{insert into } C_k \\
  \text{select } p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1} \\
  \text{from } L_{k-1} p, L_{k-1} q \\
  \text{where } p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, \\
  p.item_{k-1} < q.item_{k-1};
  \]

- Step 2: pruning: prune if it has infrequent subset

Example: Generate $C_4$ from $L_3=\{abc, abd, acd, ace, bcd\}$

- Step 1: Self-joining: $L_3 \times L_3$
  - $abcd$ from $abc$ and $abd$; $acde$ from $acd$ and $ace$
- Step 2: Pruning:
  - $acde$ is removed because $ade$ is not in $L_3$

$C_4=\{abcd\}$
How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be huge
  - Each transaction may contain many candidates

- Method:
  - Build a hash-tree for candidate itemsets
    - Leaf node contains a list of itemsets
    - Interior node contains a hash function determining which branch to follow
  - Subset function: for each transaction, find all the candidates contained in the transaction using the hash tree
Prefix Tree (Trie)

- Prefix tree (trie from retrieval)
  - Keys are usually strings
  - All descendants of one node have a common prefix

- Advantages
  - Fast looking up
  - Less space with a large number of short strings
  - Help with longest-prefix matching

- Applications
  - Storing dictionary
  - Approximate matching algorithms, including spell checking
Example: Counting Supports of Candidates

hash function
1, 4, 7

3, 6, 9

2, 5, 8

Transaction: 2 3 5 6 7

Diagram showing a tree structure with nodes labeled with numbers and arrows indicating supports.
Improving Efficiency of Apriori

- Bottlenecks
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates

- Improving Apriori: general ideas
  - Shrink number of candidates
  - Reduce passes of transaction database scans
  - Reduce number of transactions
  - Facilitate support counting of candidates
DHP: Reduce the Number of Candidates

- DHP (Direct hashing and pruning): hash $k$-itemsets into buckets and a $k$-itemset whose bucket count is below the threshold cannot be frequent
- Especially useful for 2-itemsets
  - Generate a hash table of 2-itemsets during the scan for 1-itemset
  - If the count of a bucket is below minimum support count, the itemsets in the bucket should not be included in candidate 2-itemsets

DHP: Reducing number of candidates

Database $D$

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>A C D</td>
</tr>
<tr>
<td>200</td>
<td>B C E</td>
</tr>
<tr>
<td>300</td>
<td>A B C E</td>
</tr>
<tr>
<td>400</td>
<td>B E</td>
</tr>
</tbody>
</table>

Making a hash table

100  {A C}, {A D}, {C D}
200  {B C}, {B E}, {C E}
300  {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
400  {B E}

$h\{x \ y\} = ((\text{order of } x) \times 10 + (\text{order of } y)) \mod 7$;

<table>
<thead>
<tr>
<th>C_1</th>
<th>count</th>
<th>L_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
<td>{A}</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
<td>{B}</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
<td>{C}</td>
</tr>
<tr>
<td>{D}</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
<td>{E}</td>
</tr>
</tbody>
</table>

minimum support, $s = 2$

The number of items hashed to bucket 2

Hash table $H_2$

Hash address

Generating $C_2$

$\# \text{ in a bucket with the itemset}$

$L_1 \times L_1$

{A B}  1
{A C}  3
{A E}  1
{B C}  2
{B E}  3
{C E}  3

{A C}
{B C}
{B E}
{C E}
DHP: Reducing the transactions

- If an item occurs in a frequent \((k+1)\)-itemset, it must occur in at least \(k\) candidate \(k\)-itemsets (necessary but not sufficient)
- Discard an item if it does not occur in at least \(k\) candidate \(k\)-itemsets during support counting

DIC: Reduce Number of Scans

- DIC (Dynamic itemset counting): add new candidate itemsets at partition points
  - Once both A and D are determined frequent, the counting of AD begins
  - Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins

Partitioning: Reduce Number of Scans

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database in $n$ disjoint partitions and find local frequent patterns (minimum support count?)
  - Scan 2: determine global frequent patterns from the collection of all local frequent patterns

A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB’95*
Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within samples using Apriori
- Scan database once to verify frequent itemsets found in sample
- Use a lower support threshold than minimum support
- Tradeoff accuracy against efficiency

H. Toivonen. *Sampling large databases for association rules*. In *VLDB’96*
Scalable Methods for Mining Frequent Patterns

- Scalable mining methods for frequent patterns
  - Apriori (Agrawal & Srikant@VLDB’94) and variations
  - Frequent pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD’00)
- Algorithms using vertical format
- Closed and maximal patterns and their mining methods
- FIMI Workshop and implementation repository