Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?
  - Pattern Evaluation Methods (next week)
- Summary
What Is Pattern Discovery?

- What are patterns?
  - A set of items, subsequences, or substructures, that occur frequently together (or strongly correlated) in a data set
  - Patterns represent intrinsic and important properties of datasets
- Pattern discovery
  - Uncovering patterns from massive data sets
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a notebook?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

Beer and Diapers

A 26-year-old legend:
beer and diaper sales spike
between the hours of 5 p.m. and 7 p.m.

Original NCR (now TeraData) study in 1992 by Thomas Blischok (MindMeld Inc.) for American retail chain Osco Drugs.

- examined 1.2 million market baskets in 25 stores
- NCR identified 30 different shopping experiences, such as a correlation between fruit juice and cough medication sales.
- Osco removed approximately 5,000 slow-moving items from its inventory
- by re-arranging merchandise, consumers actually thought that Osco’s selection had increased.
- putting the right merchandise in the right quantities at the right time
Why Is Pattern Discovery Important?

- Finding inherent regularities in a data set

- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Mining sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative pattern-based analysis
  - Cluster analysis: pattern-subspace clustering

- Many Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log analysis, and biological sequence analysis

Basic Concepts: Frequent Patterns

- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- (absolute) support (count) of $X$: Frequency or the number of occurrences of an itemset $X$
- (relative) support, $s$: The fraction of transactions that contains $X$ (i.e., the probability that a transaction contains $X$)
- An itemset $X$ is frequent if the support of $X$ is no less than a $\text{minsup}$ threshold ($\sigma$)

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Beer, Nuts, Diaper</td>
</tr>
<tr>
<td>20</td>
<td>Beer, Coffee, Diaper</td>
</tr>
<tr>
<td>30</td>
<td>Beer, Diaper, Eggs</td>
</tr>
<tr>
<td>40</td>
<td>Nuts, Eggs, Milk</td>
</tr>
<tr>
<td>50</td>
<td>Nuts, Coffee, Diaper, Eggs, Milk</td>
</tr>
</tbody>
</table>

- Let $\text{minsup} = 50\%$
- Freq. 1-itemsets:
  - Beer: 3 (60%); Nuts: 3 (60%)
  - Diaper: 4 (80%); Eggs: 3 (60%)
- Freq. 2-itemsets:
  - {Beer, Diaper}: 3 (60%)
### From Frequent Itemsets to Association Rules

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Beer, Nuts, Diaper</td>
</tr>
<tr>
<td>20</td>
<td>Beer, Coffee, Diaper</td>
</tr>
<tr>
<td>30</td>
<td>Beer, Diaper, Eggs</td>
</tr>
<tr>
<td>40</td>
<td>Nuts, Eggs, Milk</td>
</tr>
<tr>
<td>50</td>
<td>Nuts, Coffee, Diaper, Eggs, Milk</td>
</tr>
</tbody>
</table>

- **Association rules**: \( X \rightarrow Y \) with \( (s, c) \)
  - **Support**, \( s \): The probability that a transaction contains \( X \cup Y \)
  - **Confidence**, \( c \): The conditional probability that a transaction containing \( X \) also contains \( Y \)
  - \( c = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)} \)

- **Association rule mining**: Find all of the rules, \( X \rightarrow Y \), with minimum support and confidence

- **Frequent itemsets**: Let \( \text{minsup} = 50\% \)
  - Freq. 1-itemsets: Beer: 3, Nuts: 3, Diaper: 4, Eggs: 3
  - Freq. 2-itemsets: \{Beer, Diaper\}: 3

- **Association rules**: Let \( \text{minconf} = 50\% \)
  - Beer \( \rightarrow \) Diaper (60%, 100%)
  - Diaper \( \rightarrow \) Beer (60%, 75%)

### Challenge: There Are Too Many Frequent Patterns!

- A long pattern contains a combinatorial number of sub-patterns
- **How many frequent itemsets does the following TDB\(_1\) contain?**
  - TDB\(_1\): \( T_1: \{a_1, \ldots, a_{50}\}; \ T_2: \{a_1, \ldots, a_{100}\} \)
  - Assuming (absolute) \( \text{minsup} = 1 \)
  - Let’s have a try
    - 1-itemsets: \( \{a_1\}: 2, \{a_2\}: 2, \ldots, \{a_{50}\}: 2, \{a_{51}\}: 1, \ldots, \{a_{100}\}: 1 \)
    - 2-itemsets: \( \{a_1, a_2\}: 2, \ldots, \{a_1, a_{50}\}: 2, \{a_1, a_{51}\}: 1, \ldots, \{a_{99}, a_{100}\}: 1, \ldots, \ldots, \ldots \)
    - 99-itemsets: \( \{a_1, a_2, \ldots, a_{99}\}: 1, \ldots, \{a_2, a_3, \ldots, a_{100}\}: 1 \)
    - 100-itemsets: \( \{a_1, a_2, \ldots, a_{100}\}: 1 \)
  - In total: \( \binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \) sub-patterns!
Expressing Patterns in Compressed Form: Closed Patterns

- How to handle such a challenge?
- **Solution 1: Closed patterns**: A pattern (itemset) $X$ is *closed* if $X$ is frequent, and there exists no super-pattern $Y \supset X$, with the same support as $X$
  
  - Let Transaction DB TDB: $T_1: \{a_1, \ldots, a_{50}\}; T_2: \{a_1, \ldots, a_{100}\}$
  
  - Suppose $\text{minsup} = 1$. How many closed patterns does TDB contain?
    - Two: $P_1: "\{a_1, \ldots, a_{50}\}: 2"$; $P_2: "\{a_1, \ldots, a_{100}\}: 1"$
  
  - Closed pattern is a *lossless compression* of frequent patterns
  
  - Reduces the # of patterns but does not lose the support information!
  
  - You will still be able to say: "{$a_2, \ldots, a_{40}$}: 2", "{$a_5, a_{51}$}: 1"

Expressing Patterns in Compressed Form: Max-Patterns

- **Solution 2: Max-patterns**: A pattern $X$ is a *max-pattern* if $X$ is frequent and there exists no frequent super-pattern $Y \supset X$

- Difference from close-patterns?
  
  - Do not capture the real support of the sub-patterns of a max-pattern
  
  - Let Transaction DB TDB: $T_1: \{a_1, \ldots, a_{50}\}; T_2: \{a_1, \ldots, a_{100}\}$
  
  - Suppose $\text{minsup} = 1$. How many max-patterns does TDB contain?
    - One: P: "{$a_1, \ldots, a_{100}$}: 1"

- Max-pattern is a *lossy compression*!
  
  - We only know {$a_1, \ldots, a_{40}$} is frequent
  
  - But we do not know the real support of {$a_1, \ldots, a_{40}$}, ..., any more!

- Thus in many applications, mining close-patterns is more desirable than mining max-patterns
Chapter 6: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

- Basic Concepts
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?
  - Pattern Evaluation Methods (next week)
- Summary

Scalable Frequent Itemset Mining Methods

- The Downward Closure Property of Frequent Patterns
- The Apriori Algorithm
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- FPGrowth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns
The Downward Closure Property of Frequent Patterns

- Observation: From TDB₁, \( T₁ \): \{\( a₁ \), ..., \( a_{50} \)\}; \( T₂ \): \{\( a_{1} \), ..., \( a_{100} \)\}
  - We get a frequent itemset: \{\( a₁ \), ..., \( a_{50} \)\}
  - Also, its subsets are all frequent: \{\( a₁ \)\}, \{\( a₂ \)\}, ..., \{\( a_{50} \)\}, \{\( a₁ \), \( a₂ \)\}, ..., \{\( a₁ \), ..., \( a_{49} \)\}, ...
  - There must be some hidden relationships among frequent patterns!

- The downward closure (also called "Apriori") property of frequent patterns
  - If \{\textit{beer}, \textit{diaper}, \textit{nuts}\} is frequent, so is \{\textit{beer}, \textit{diaper}\}
  - Every transaction containing \{\textit{beer}, \textit{diaper}, \textit{nuts}\} also contains \{\textit{beer}, \textit{diaper}\}
  - 

Apriori: Any subset of a frequent itemset must be frequent

- Efficient mining methodology
  - If any subset of an itemset \( S \) is infrequent, then there is no chance for \( S \) to be frequent—why do we even have to consider \( S! \)?

A sharp knife for pruning!

Apriori Pruning and Scalable Mining Methods

- **Apriori pruning principle:** If there is any itemset which is infrequent, its superset should not even be generated! (Agrawal & Srikant @VLDB’94, Mannila, et al. @ KDD’94)

- **Scalable mining Methods:** Three major approaches
  - Level-wise, join-based approach: Apriori (Agrawal & Srikant@VLDB’94)
  - Vertical data format approach: Eclat (Zaki, Parthasarathy, Ogihara, Li @KDD’97)
  - Frequent pattern projection and growth: FPgrowth (Han, Pei, Yin @SIGMOD’00)
Apriori: A Candidate Generation & Test Approach

- **Apriori pruning principle**: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD'94)

- Outline of Apriori (level-wise, candidate generation and test)
  - Initially, scan DB once to get frequent 1-itemset
  - Repeat
    - Generate length-(k+1) candidate itemsets from length-k frequent itemsets
    - Test the candidates against DB to find frequent (k+1)-itemsets
    - Set k := k + 1
  - Until no frequent or candidate set can be generated
  - Return all the frequent itemsets derived

The Apriori Algorithm—An Example

- **minsup = 2**
- Database TDB
  
<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>

1\(^{st}\) scan

\(C_1\)  \(L_1\)

\begin{align*}
\{A\} & : 2 \\
\{B\} & : 3 \\
\{C\} & : 3 \\
\{D\} & : 1 \\
\{E\} & : 3 \\
\end{align*}

2\(^{nd}\) scan

\(C_2\)

\begin{align*}
\{A, B\} & : 1 \\
\{A, C\} & : 2 \\
\{A, E\} & : 1 \\
\{B, C\} & : 2 \\
\{B, E\} & : 3 \\
\{C, E\} & : 2 \\
\end{align*}

3\(^{rd}\) scan

\(C_3\)

\{B, C, E\}

\begin{align*}
\{B, C, E\} & : 2 \\
\end{align*}

\(L_i\)
The Apriori Algorithm (Pseudo-Code)

$C_k$: Candidate itemset of size $k$
$F_k$: frequent itemset of size $k$

$k := 1$;
$F_1 = \{\text{frequent items}\}$;
while ( $F_k \neq \emptyset$ ) do

$C_{k+1} = \text{candidates generated from } F_k$;
for each transaction $t$ in database do

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

$F_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}$

$k := k + 1$;
od
return $\cup_k F_k$;

Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining $F_k$
  - Step 2: pruning
- Example of Candidate-generation
  - $F_3 = \{abc, abd, acd, ace, bcd\}$
  - Self-joining: $F_3 \ast F_3$
    - $abcd$ from $abc$ and $abd$
    - $acde$ from $acd$ and $ace$
  - Pruning:
    - $acde$ is removed because $ade$ is not in $F_3$
  - $C_4 = \{abcd\}$
How to Count Supports of Candidates?

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

- Method:
  - Candidate itemsets are stored in a hash-tree
  - Leaf node of hash-tree contains a list of itemsets and counts
  - Interior node contains a hash table
  - Subset function: finds all the candidates contained in a transaction

Counting Supports of Candidates Using Hash Tree

Items: 1, 2, 3, 4, 5, 6, 7, 8, 9

Subset function
1.4.7 3.6.9
2.5.8

Transaction: 1 2 3 5 6

Leafs: Candidate itemsets
Candidate Generation: An SQL Implementation

- SQL Implementation of candidate generation
  - Suppose the items in $F_{k-1}$ are listed in an order
  - Step 1: self-joining $F_{k-1}$
    insert into $C_k$
    select $p$.item$_1$, $p$.item$_2$, ..., $p$.item$_{k-1}$, $q$.item$_{k-1}$
    from $F_{k-1}$ $p$, $F_{k-1}$ $q$
    where $p$.item$_1$=$q$.item$_1$, ..., $p$.item$_{k-2}$=$q$.item$_{k-2}$, $p$.item$_{k-1}$ < $q$.item$_{k-1}$
  - Step 2: pruning
    forall itemsets $c$ in $C_k$ do
    forall (k-1)-subsets $s$ of $c$ do
      if ($s$ is not in $F_{k-1}$) then delete $c$ from $C_k$

- Use object-relational extensions like UDFs, BLOBs, and Table functions for efficient implementation [S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD’98]

Scalable Frequent Itemset Mining Methods

- The Downward Closure Property of Frequent Patterns
- The Apriori Algorithm
- Extensions or Improvements of Apriori
- Mining Frequent Patterns by Exploring Vertical Data Format
- FP-Growth: A Frequent Pattern-Growth Approach
- Mining Closed Patterns
Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates

Apriori: Improvements and Alternatives

- Reduce passes of transaction database scans
  - Partitioning (e.g., Savasere, et al., 1995)
  - Dynamic itemset counting (Brin, et al., 1997)
- Shrink the number of candidates
  - Hashing (e.g., DHP: Park, et al., 1995)
  - Pruning by support lower bounding (e.g., Bayardo 1998)
  - Sampling (e.g., Toivonen, 1996)
- Exploring special data structures
  - Tree projection (Agarwal, et al., 2001)
  - H-miner (Pei, et al., 2001)
  - Hypercube decomposition (e.g., LCM: Uno, et al., 2004)
Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*

\[
\text{DB}_1 + \text{DB}_2 + \ldots + \text{DB}_k = \text{DB} \\
\text{sup}_1(X) < \sigma \text{DB}_1 \\
\text{sup}_2(X) < \sigma \text{DB}_2 \\
\text{sup}_k(X) < \sigma \text{DB}_k \\
\text{sup}(X) < \sigma \text{DB}
\]

DHP: Reduce the Number of Candidates

- A \( k \)-itemset whose corresponding hashing bucket count is below the support threshold cannot be frequent
- Candidates: a, b, c, d, e
- Hash entries
  - \{ab, ad, ae\}
  - \{bd, be, de\}
  - ...
- Frequent 1-itemset: a, b, d, e
- ab is not a candidate 2-itemset if the sum of count of \{ab, ad, ae\} is below the support threshold

<table>
<thead>
<tr>
<th>Itemsets</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ab, ad, ae}</td>
<td>35</td>
</tr>
<tr>
<td>{bd, be, de}</td>
<td>298</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>{yz, qs, wt}</td>
<td>58</td>
</tr>
</tbody>
</table>

Hash Table

- J. Park, M. Chen, and P. Yu. *An effective hash-based algorithm for mining association rules. SIGMOD'95 (Direct Hashing and Pruning (DHP))*
Exploring Vertical Data Format: ECLAT

- ECLAT (Equivalence Class Transformation): A depth-first search algorithm using set intersection [Zaki et al. @KDD’97]

- **Tid-List**: List of transaction-ids containing the itemset(s)
- **Vertical format**: \( t(e) = \{T_{10}, T_{20}, T_{30}\}; t(a) = \{T_{10}, T_{30}\}; t(ae) = \{T_{10}, T_{30}\} \)

- **Properties of Tid-Lists**
  - \( t(X) = t(Y) \): \( X \) and \( Y \) always happen together (e.g., \( t(ac) = t(d) \) )
  - \( t(X) \subseteq t(Y) \): transaction having \( X \) always has \( Y \) (e.g., \( t(ac) \subseteq t(ce) \) )

- **Deriving frequent patterns based on vertical intersections**
- **Using diffset to accelerate mining**
  - Only keep track of differences of tids
  - \( t(e) = \{T_{10}, T_{20}, T_{30}\}, t(ce) = \{T_{10}, T_{30}\} \rightarrow \text{Diffset} \ (ce, e) = \{T_{20}\} \)

Sampling for Frequent Patterns

- **Select a sample of the original database**, mine frequent patterns within the sample using Apriori
- **Scan database once to verify frequent itemsets found in sample. Here only borders of closure of frequent patterns are checked**:
  - Example: check \( abcd \) instead of \( ab, ac, \ldots, \) etc. (why?)
- **Scan database again to find missed frequent patterns.**
- **H. Toivonen. Sampling large databases for association rules. In VLDB’96**
Frequent Itemset Mining

Ref: Apriori and Its Improvements

References (II) Efficient Pattern Mining Methods

- J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules", SIGMOD'95
- S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating association rule mining with relational database systems: Alternatives and implications", SIGMOD'98
- J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", SIGMOD'00
- M. J. Zaki and Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining", SDM'02
- J. Wang, J. Han, and J. Pei, "CLOSEST+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, "Frequent Pattern Mining Algorithms: A Survey", in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014

References (III) Pattern Evaluation

- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010