Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining

Summary
Chapter 5: Mining Frequent Patterns, Association and Correlations

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

- **Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- 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.
Why Is Freq. Pattern Mining Important?

- Discloses an intrinsic and important property of data sets
- Forms the foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
- Classification: associative classification
- Cluster analysis: frequent pattern-based clustering
- Data warehousing: iceberg cube and cube-gradient
- Semantic data compression: fascicles
- Broad applications
Basic Concepts: Frequent Patterns and Association Rules

- Itemset $X = \{x_1, ..., x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - **support**, $s$, probability that a transaction contains $X \cup Y$
  - **confidence**, $c$, conditional probability that a transaction having $X$ also contains $Y$

Let $\text{sup}_{\text{min}} = 50\%$, $\text{conf}_{\text{min}} = 50\%$

Freq. Pat.: \{A:3, B:3, D:4, E:3, AD:3\}

Association rules:
- $A \rightarrow D$ (60%, 100%)
- $D \rightarrow A$ (60%, 75%)

<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>
Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., \{a_1, \ldots, a_{100}\} contains \(\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27 \times 10^{30}\) sub-patterns!

- Solution: Mine **closed patterns and max-patterns instead**

- An itemset \(X\) is **closed** if \(X\) is frequent and there exists no super-pattern \(Y \supset X\), with the same support as \(X\) (proposed by Pasquier, et al. @ ICDT’99)

- An itemset \(X\) is a **max-pattern** if \(X\) is frequent and there exists no frequent super-pattern \(Y \supset X\) (proposed by Bayardo @ SIGMOD’98)

- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules
Closed Patterns and Max-Patterns

- Exercise. DB = \{<a_1, \ldots, a_{100}>, <a_1, \ldots, a_{50}>)\}
  - Min_sup = 1.

- What is the set of closed itemset?
  - <a_1, \ldots, a_{100}>: 1
  - <a_1, \ldots, a_{50}>: 2

- What is the set of max-pattern?
  - <a_1, \ldots, a_{100}>: 1

- What is the set of all patterns?
  - !!
Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- 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

- 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\}

- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB’ 94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD’ 00)
  - Vertical data format approach (Charm—Zaki & Hsiao @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! (Agrawal & Srikant @VLDB’94, Mannila, et al. @ KDD’94)

- **Method:**
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated
The Apriori Algorithm—An Example

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>

Sup$_{\text{min}} = 2$

$C_1$

1$^{\text{st}}$ scan

<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>

$C_2$

2$^{\text{nd}}$ scan

<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>{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>

$L_2$

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

$C_3$

3$^{\text{rd}}$ scan

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

Sup$_{\text{min}} = 2$
The Apriori Algorithm

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

  $L_1 = \{\text{frequent items}\};$

  **for** ($k = 1; L_k \neq \emptyset; k++$) **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** $\bigcup_k L_k$;
Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining $L_k$
  - Step 2: pruning

- How to count supports of candidates?

- Example of Candidate-generation
  - $L_3=\{abc, abd, acd, ace, bcd\}$
  - Self-joining: $L_3 \times L_3$
    - $abcd$ from $abc$ and $abd$
    - $acde$ from $acd$ and $ace$
  - Pruning:
    - $acde$ is removed because $ade$ is not in $L_3$
  - $C_4=\{abcd\}$
How to Generate Candidates?

- Suppose the items in $L_{k-1}$ are listed in an order
- Step 1: self-joining $L_{k-1}$
  
  insert into $C_k$
  
  select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
  
  from $L_{k-1} p, L_{k-1} q$
  
  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 $L_{k-1}$) then delete $c$ from $C_k$
How to Count Supports of Candidates?

- Why 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
Example: Counting Supports of Candidates

Subset function

$1,4,7 \rightarrow 3,6,9$
$2,5,8$

Transaction: $1 \ 2 \ 3 \ 5 \ 6$

$1 + 2 \ 3 \ 5 \ 6$

$1 \ 3 + 5 \ 6$

$1 \ 2 + 3 \ 5 \ 6$

$1 \ 4 \ 5$

$1 \ 2 \ 4 \ 5 \ 7$

$1 \ 3 \ 6$

$1 \ 5 \ 9$

$4 \ 5 \ 8$

$2 \ 3 \ 4 \ 5 \ 6$

$1 \ 3 \ 6 \ 5 \ 7 \ 6 \ 8 \ 9$

$3 \ 6 \ 7 \ 3 \ 6 \ 8$
Efficient Implementation of Apriori in SQL

- Hard to get good performance out of pure SQL (SQL-92) based approaches alone
- Make use of object-relational extensions like UDFs, BLOBs, Table functions etc.
  - Get orders of magnitude improvement
- S. Sarawagi, S. Thomas, and R. Agrawal. *Integrating association rule mining with relational database systems: Alternatives and implications*. In *SIGMOD’ 98*
Challenges of Frequent Pattern Mining

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

- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates
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

DHP: Reduce the Number of Candidates

- A $k$-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Candidates: a, b, c, d, e
- 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 support threshold

J. Park, M. Chen, and P. Yu. \textit{An effective hash-based algorithm for mining association rules}. In \textit{SIGMOD’95}
Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check $abcd$ instead of $ab, ac, ..., etc.$
- Scan database again to find missed frequent patterns

H. Toivonen. *Sampling large databases for association rules.* In *VLDB’96*
DIC: Reduce Number of Scans

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.

Apriori

Transactions

1-itemsets

2-itemsets

...
Bottleneck of Frequent-pattern Mining

- Multiple database scans are costly
- Mining long patterns needs many passes of scanning and generates lots of candidates
  - To find frequent itemset $i_1i_2...i_{100}$
    - # of scans: 100
    - # of Candidates: $1 + 100 + ... + (100\choose100) = 2^{100}-1 = 1.27 \times 10^{30}$!
- Bottleneck: candidate-generation-and-test
- Can we avoid candidate generation?
Mining Frequent Patterns Without Candidate Generation

- Grow long patterns from short ones using local frequent items
  - “abc” is a frequent pattern
  - Get all transactions having “abc”: DB|abc
  - “d” is a local frequent item in DB|abc → abcd is a frequent pattern
Construct FP-tree from a Transaction Database

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

F-list = f-c-a-b-m-p

TID  Items bought  (ordered) frequent items
100  \{f, a, c, d, g, i, m, p\}  \{f, c, a, m, p\}
200  \{a, b, c, f, l, m, o\}  \{f, c, a, b, m\}
300  \{b, f, h, j, o, w\}  \{f, b\}
400  \{b, c, k, s, p\}  \{c, b, p\}
500  \{a, f, c, e, l, p, m, n\}  \{f, c, a, m, p\}

Header Table

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

min\_support = 3
Benefits of the FP-tree Structure

- **Completeness**
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction

- **Compactness**
  - Reduce irrelevant info—in frequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)
  - For Connect-4 DB, compression ratio could be over 100
Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list=f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - ... Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency
**Find Patterns Having \( P \) From \( P \)-conditional Database**

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item \( p \)
- Accumulate all of *transformed prefix paths* of item \( p \) to form \( p \)’s conditional pattern base

### Header Table

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>( c )</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>( a )</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>( b )</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>( m )</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

### Conditional pattern bases

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c )</td>
<td>( f:3 )</td>
</tr>
<tr>
<td>( a )</td>
<td>( fca:1, f:1, c:1 )</td>
</tr>
<tr>
<td>( m )</td>
<td>( fca:2, fcab:1 )</td>
</tr>
<tr>
<td>( p )</td>
<td>( fcam:2, cb:1 )</td>
</tr>
</tbody>
</table>
From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

**Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**m-conditional pattern base:**

- $fca:2$, $fcab:1$

All frequent patterns relate to $m$

- $\{\}$
- $f:3$
- $c:3$
- $a:3$

**m-conditional FP-tree**
Recursion: Mining Each Conditional FP-tree

Cond. pattern base of “am”: (fc:3)

Cond. pattern base of “cm”: (f:3)

cam-conditional FP-tree

m-conditional FP-tree

am-conditional FP-tree
A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree $T$ has a shared single prefix-path $P$
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts

\[
\emptyset \quad a_1 : n_1 \quad a_2 : n_2 \quad a_3 : n_3
\]

\[
\begin{align*}
\text{b}_1 : m_1 & \quad \text{c}_1 : k_1 & \quad \text{c}_2 : k_2 & \quad \text{c}_3 : k_3 \\
\end{align*}
\]

\[
\Rightarrow r_1 = \emptyset + \begin{array}{c}
\begin{align*}
\text{a}_1 : n_1 & \\
\end{align*}
\end{array} + 
\begin{array}{c}
\begin{align*}
\text{a}_2 : n_2 & \\
\end{align*}
\end{array} + 
\begin{array}{c}
\begin{align*}
\text{a}_3 : n_3 & \\
\end{align*}
\end{array} + 
\begin{array}{c}
\begin{align*}
\text{b}_1 : m_1 & \quad \\
\text{c}_1 : k_1 & \quad \text{c}_2 : k_2 & \quad \text{c}_3 : k_3
\end{align*}
\end{array}
\]

March 11, 2014

Data Mining: Concepts and Techniques
Mining Frequent Patterns With FP-trees

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern
Scaling FP-growth by DB Projection

- FP-tree cannot fit in memory?—DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. Partition projection techniques
  - Parallel projection is space costly
Partition-based Projection

- Parallel projection needs a lot of disk space
- Partition projection saves it

**Parallel projection**

- Parallel projection needs a lot of disk space
- Parallel projection saves it

**Partition projection**

- Partition projection needs a lot of disk space
- Partition projection saves it

**Diagram**

- **Tran. DB**
  - fcamp
  - fcabm
  - fb
  - cbp
  - fcamp

- **p-proj DB**
  - fcamp
  - cb
  - fcam

- **m-proj DB**
  - fcab
  - fca
  - fca

- **b-proj DB**
  - f
  - cb
  - ...

- **a-proj DB**
  - fc
  - ...

- **c-proj DB**
  - f
  - ...

- **f-proj DB**
  - ...

- **am-proj DB**
  - fc
  - fc
  - fc

- **cm-proj DB**
  - f
  - f
  - f

...
FP-Growth vs. Apriori: Scalability With the Support Threshold

Data set T25I20D10K
FP-Growth vs. Tree-Projection: Scalability with the Support Threshold

Data set T25I20D100K

- D2 FP-growth
- D2 TreeProjection
Why Is FP-Growth the Winner?

- Divide-and-conquer:
  - decompose both the mining task and DB according to the frequent patterns obtained so far
  - leads to focused search of smaller databases

- Other factors
  - no candidate generation, no candidate test
  - compressed database: FP-tree structure
  - no repeated scan of entire database
  - basic ops—counting local freq items and building sub FP-tree, no pattern search and matching
Implications of the Methodology

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD’00)
- Mining sequential patterns
  - FreeSpan (KDD’00), PrefixSpan (ICDE’01)
- Constraint-based mining of frequent patterns
  - Convertible constraints (KDD’00, ICDE’01)
- Computing iceberg data cubes with complex measures
  - H-tree and H-cubing algorithm (SIGMOD’01)
MaxMiner: Mining Max-patterns

- 1st scan: find frequent items
  - A, B, C, D, E

- 2nd scan: find support for
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE, DE,

- Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

- R. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD’98*
Mining Frequent Closed Patterns: CLOSET

- **Flist**: list of all frequent items in support ascending order
  - Flist: d-a-f-e-c
- **Divide search space**
  - Patterns having d
  - Patterns having d but no a, etc.
- **Find frequent closed pattern recursively**
  - Every transaction having d also has cfa → cfad is a frequent closed pattern

J. Pei, J. Han & R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets", DMKD'00.

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>20</td>
<td>a, b, e</td>
</tr>
<tr>
<td>30</td>
<td>c, e, f</td>
</tr>
<tr>
<td>40</td>
<td>a, c, d, f</td>
</tr>
<tr>
<td>50</td>
<td>c, e, f</td>
</tr>
</tbody>
</table>
CLOSET+: Mining Closed Itemsets by Pattern-Growth

- Itemset merging: if $Y$ appears in every occurrence of $X$, then $Y$ is merged with $X$
- Sub-itemset pruning: if $Y \subseteq X$, and $\text{sup}(X) = \text{sup}(Y)$, $X$ and all of $X$’s descendants in the set enumeration tree can be pruned
- Hybrid tree projection
  - Bottom-up physical tree-projection
  - Top-down pseudo tree-projection
- Item skipping: if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels
- Efficient subset checking
CHARM: Mining by Exploring Vertical Data Format

- Vertical format: $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving closed patterns based on vertical intersections
  - $t(X) = t(Y)$: X and Y always happen together
  - $t(X) \subset t(Y)$: transaction having X always has Y
- Using **diffset** to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$
- Eclat/MaxEclat (Zaki et al. @KDD’97), VIPER (P. Shenoy et al. @SIGMOD’00), CHARM (Zaki & Hsiao@SDM’02)
Further Improvements of Mining Methods

- AFOPT (Liu, et al. @ KDD’03)
  - A “push-right” method for mining condensed frequent pattern (CFP) tree

- Carpenter (Pan, et al. @ KDD’03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining
Visualization of Association Rules: Plane Graph

Promotions = [No Promotion] ==> Gender = [M]; [support: 37.06% , confidence: 50.55%]
Visualization of Association Rules: Rule Graph
Visualization of Association Rules (SGI/MineSet 3.0)
Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary
Mining Various Kinds of Association Rules

- Mining multilevel association
- Mining multidimensional association
- Mining quantitative association
- Mining interesting correlation patterns
Mining Multiple-Level Association Rules

- Items often form hierarchies
- Flexible support settings
  - Items at the lower level are expected to have lower support
- Exploration of *shared* multi-level mining (Agrawal & Srikant@VLB’95, Han & Fu@VLDB’95)

**uniform support**

- Level 1
  - min_sup = 5%

- Level 2
  - min_sup = 5%

**reduced support**

- Level 1
  - min_sup = 5%

- Level 2
  - min_sup = 3%

- Milk [support = 10%]
- 2% Milk [support = 6%]
- Skim Milk [support = 4%]
Multi-level Association: Redundancy Filtering

- Some rules may be redundant due to “ancestor” relationships between items.
- Example
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]
- We say the first rule is an ancestor of the second rule.
- A rule is redundant if its support is close to the “expected” value, based on the rule’s ancestor.
Mining Multi-Dimensional Association

- Single-dimensional rules:
  \[ \text{buys}(X, \text{“milk”}) \Rightarrow \text{buys}(X, \text{“bread”}) \]

- Multi-dimensional rules: \( \geq 2 \) dimensions or predicates
  - Inter-dimension assoc. rules (no repeated predicates)
    \[ \text{age}(X, \text{“19-25”}) \land \text{occupation}(X, \text{“student”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]
  - hybrid-dimension assoc. rules (repeated predicates)
    \[ \text{age}(X, \text{“19-25”}) \land \text{buys}(X, \text{“popcorn”}) \Rightarrow \text{buys}(X, \text{“coke”}) \]

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach

- Quantitative Attributes: numeric, implicit ordering among values—discretization, clustering, and gradient approaches
Mining Quantitative Associations

- Techniques can be categorized by how numerical attributes, such as age or salary are treated
- Static discretization based on predefined concept hierarchies (data cube methods)
- Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
- Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
  - one dimensional clustering then association
- Deviation: (such as Aumann and Lindell@KDD99)
  Sex = female => Wage: mean=$7/hr (overall mean = $9)
Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require $k$ or $k+1$ table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.

Diagram:

```
  (age)
  / \  \
(age, income) /   \ (income, buys)
/     \     
(age, buys) /       \ (income, buys)
```

Example predicate sets:
- (income)
- (age)
- (age, income)
- (age, buys)
- (income, buys)
- (age, income, buys)
Quantitative Association Rules

- Proposed by Lent, Swami and Widom ICDE’ 97
- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized
- 2-D quantitative association rules: \( A_{\text{quan1}} \land A_{\text{quan2}} \Rightarrow A_{\text{cat}} \)
- Cluster *adjacent* association rules to form general rules using a 2-D grid
- Example

\[
\text{age}(X, "34-35") \land \text{income}(X, "30-50K") \Rightarrow \text{buys}(X, "high resolution TV")
\]
Mining Other Interesting Patterns

- Flexible support constraints (Wang et al. @ VLDB’02)
  - Some items (e.g., diamond) may occur rarely but are valuable
  - Customized $\text{sup}_{\text{min}}$ specification and application

- Top-K closed frequent patterns (Han, et al. @ ICDM’02)
  - Hard to specify $\text{sup}_{\text{min}}$, but top-$k$ with $\text{length}_{\text{min}}$ is more desirable
  - Dynamically raise $\text{sup}_{\text{min}}$ in FP-tree construction and mining, and select most promising path to mine
Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary
Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: \( \text{lift} \)

\[
\text{lift} = \frac{P(A \cup B)}{P(A)P(B)}
\]

\[
\text{lift}(B, C) = \frac{2000 / 5000}{3000 / 5000 \times 3750 / 5000} = 0.89
\]

\[
\text{lift}(B, \neg C) = \frac{1000 / 5000}{3000 / 5000 \times 1250 / 5000} = 1.33
\]

<table>
<thead>
<tr>
<th></th>
<th>Basketball</th>
<th>Not basketball</th>
<th>Sum (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cereal</td>
<td>2000</td>
<td>1750</td>
<td>3750</td>
</tr>
<tr>
<td>Not cereal</td>
<td>1000</td>
<td>250</td>
<td>1250</td>
</tr>
<tr>
<td>Sum(col.)</td>
<td>3000</td>
<td>2000</td>
<td>5000</td>
</tr>
</tbody>
</table>
Are \textit{lift} and $\chi^2$ Good Measures of Correlation?

- “Buy walnuts $\Rightarrow$ buy milk [1\%, 80\%]” is misleading
  - if 85\% of customers buy milk
- Support and confidence are not good to represent correlations
- So many interestingness measures? (Tan, Kumar, Sritastava @KDD’02)

\[
\text{lift} = \frac{P(A \cup B)}{P(A)P(B)}
\]

\[
\text{all\_conf} = \frac{\text{sup}(X)}{\max_{\text{item}} \text{sup}(X)}
\]

\[
\text{coh} = \frac{\text{sup}(X)}{|\text{universe}(X)|}
\]

<table>
<thead>
<tr>
<th>DB</th>
<th>m, c</th>
<th>$\sim m$, c</th>
<th>m$\sim$c</th>
<th>$\sim m$$\sim$c</th>
<th>lift</th>
<th>all-conf</th>
<th>coh</th>
<th>$\chi^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>1000</td>
<td>100</td>
<td>100</td>
<td>10,000</td>
<td>9.26</td>
<td>0.91</td>
<td>0.83</td>
<td>9055</td>
</tr>
<tr>
<td>A2</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
<td>100,000</td>
<td>8.44</td>
<td>0.09</td>
<td>0.05</td>
<td>670</td>
</tr>
<tr>
<td>A3</td>
<td>1000</td>
<td>100</td>
<td>10000</td>
<td>100,000</td>
<td>9.18</td>
<td>0.09</td>
<td>0.09</td>
<td>8172</td>
</tr>
<tr>
<td>A4</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1</td>
<td>0.5</td>
<td>0.33</td>
<td>0</td>
</tr>
</tbody>
</table>
Which Measures Should Be Used?

- **lift** and **$\chi^2$** are not good measures for correlations in large transactional DBs.
- **all-conf** or **coherence** could be good measures (Omiecinski@TKDE’03).
- Both **all-conf** and **coherence** have the downward closure property.
- Efficient algorithms can be derived for mining (Lee et al. @ICDM’03sub).

<table>
<thead>
<tr>
<th>symbol</th>
<th>measure</th>
<th>range</th>
<th>formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$\phi$-coefficient</td>
<td>$-1 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$Q$</td>
<td>Yule’s Q</td>
<td>$-1 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$Y$</td>
<td>Yule’s Y</td>
<td>$-1 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$k$</td>
<td>Cohen’s</td>
<td>$-1 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$PS$</td>
<td>Pietetsky-Shapiro’s</td>
<td>$-0.25 \ldots 0.25$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$F$</td>
<td>Certainty factor</td>
<td>$-1 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$AV$</td>
<td>added value</td>
<td>$-0.5 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$K$</td>
<td>Klosgen’s Q</td>
<td>$-0.33 \ldots 0.38$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$g$</td>
<td>Goodman-kruskal’s</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}_j P(A_j</td>
</tr>
<tr>
<td>$M$</td>
<td>Mutual Information</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\sum_j \text{max}_k P(A_j</td>
</tr>
<tr>
<td>$J$</td>
<td>J-Measure</td>
<td>$0 \ldots 1$</td>
<td>( \frac{P(B</td>
</tr>
<tr>
<td>$G$</td>
<td>Gini index</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}(P(A</td>
</tr>
<tr>
<td>$s$</td>
<td>support</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}(\frac{NP(A</td>
</tr>
<tr>
<td>$c$</td>
<td>confidence</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}(\sum_j P(A_j</td>
</tr>
<tr>
<td>$L$</td>
<td>Laplace</td>
<td>$0 \ldots 1$</td>
<td>( \frac{P(A)P(\overline{B}) + P(B</td>
</tr>
<tr>
<td>$IS$</td>
<td>Cosine</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}(P(A</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>coherence (Jaccard)</td>
<td>$0 \ldots 1$</td>
<td>( \frac{P(A</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>all_confidence</td>
<td>$0 \ldots 1$</td>
<td>( \frac{\text{max}(\frac{NP(A</td>
</tr>
<tr>
<td>$o$</td>
<td>odds ratio</td>
<td>$0 \ldots \infty$</td>
<td>( \frac{\text{max}(\sum_j P(A_j</td>
</tr>
<tr>
<td>$V$</td>
<td>Conviction</td>
<td>$0.5 \ldots \infty$</td>
<td>( \frac{\text{max}(\frac{NP(A</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>lift</td>
<td>$0 \ldots \infty$</td>
<td>( \frac{P(A,B)P(B</td>
</tr>
<tr>
<td>$S$</td>
<td>Collective strength</td>
<td>$0 \ldots \infty$</td>
<td>( \frac{\text{max}(\sum_j P(A_j</td>
</tr>
<tr>
<td>$\chi^2$</td>
<td>$\chi^2$</td>
<td>$0 \ldots \infty$</td>
<td>( \frac{\text{max}(\frac{NP(A</td>
</tr>
</tbody>
</table>
Chapter 5: Mining Frequent Patterns, Association and Correlations

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

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - System optimization: explores such constraints for efficient mining—constraint-based mining
Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.

- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago in Dec.’02

- Dimension/level constraint
  - in relevance to region, price, brand, customer category

- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)

- Interestingness constraint
  - strong rules: min_support ≥ 3%, min_confidence ≥ 60%
Constrained Mining vs. Constraint-Based Search

- Constrained mining vs. constraint-based search/reasoning
  - Both are aimed at reducing search space
  - Finding all patterns satisfying constraints vs. finding some (or one) answer in constraint-based search in AI
  - Constraint-pushing vs. heuristic search
  - It is an interesting research problem on how to integrate them

- Constrained mining vs. query processing in DBMS
  - Database query processing requires to find all
  - Constrained pattern mining shares a similar philosophy as pushing selections deeply in query processing
Anti-Monotonicity in Constraint Pushing

- Anti-monotonicity
  - When an itemset $S$ violates the constraint, so does any of its superset
    - $\text{sum}(S.\text{Price}) \leq v$ is anti-monotone
    - $\text{sum}(S.\text{Price}) \geq v$ is not anti-monotone
  - Example. C: $\text{range}(S.\text{profit}) \leq 15$ is anti-monotone
    - Itemset $ab$ violates C
    - So does every superset of $ab$

### TDB (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

### Item Profit Table

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Monotonicity for Constraint Pushing

- **Monotonicity**
  - When an itemset $S$ **satisfies** the constraint, so does any of its superset
  - $\text{sum}(S.\text{Price}) \geq v$ is monotone
  - $\text{min}(S.\text{Price}) \leq v$ is monotone

- Example. C: range(S.profit) $\geq 15$
- Itemset $ab$ satisfies C
- So does every superset of $ab$

---

**TDB (min_sup=2)**

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Succinctness:

- Given $A_1$, the set of items satisfying a succinctness constraint $C$, then any set $S$ satisfying $C$ is based on $A_1$, i.e., $S$ contains a subset belonging to $A_1$.

- Idea: Without looking at the transaction database, whether an itemset $S$ satisfies constraint $C$ can be determined based on the selection of items.

- $min(S.Price) \leq v$ is succinct

- $sum(S.Price) \geq v$ is not succinct

- Optimization: If $C$ is succinct, $C$ is pre-counting pushable.
The Apriori Algorithm — Example

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

$C_1$

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

Scan D

$L_1$

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

$L_2$

- C3: {2 3 5}

$L_3$

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2 3 5}</td>
<td>2</td>
</tr>
</tbody>
</table>

$L_4$
Naïve Algorithm: Apriori + Constraint

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

\[ \text{Scan } D \]

\[ C_2 \]

itemset \sup
\{1 2\} \quad 1
\{1 3\} \quad 2
\{1 5\} \quad 1
\{2 3\} \quad 2
\{2 5\} \quad 3
\{3 5\} \quad 2

\[ \text{Scan } D \]

\[ C_3 \]

itemset
\{2 3 5\}

\[ \text{Scan } D \]

\[ L_3 \]

itemset \sup
\{2 3 5\} \quad 2

\[ \text{Scan } D \]

\[ L_1 \]

itemset \sup
\{1\} \quad 2
\{2\} \quad 3
\{3\} \quad 3
\{4\} \quad 1
\{5\} \quad 3

Constraint:
\text{Sum}\{S\text{.price}\} < 5
The Constrained Apriori Algorithm: Push an Anti-monotone Constraint Deep

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Scan D

$L_2$

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup} \\
\hline
\{1,3\} & 2 \\
\{2,3\} & 2 \\
\{2,5\} & 3 \\
\{3,5\} & 2 \\
\hline
\end{array}
\]

Scan D

$L_1$

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup} \\
\hline
\{1\} & 2 \\
\{2\} & 3 \\
\{3\} & 3 \\
\{5\} & 3 \\
\hline
\end{array}
\]

Scan D

$L_3$

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup} \\
\hline
\{2,3,5\} & 2 \\
\hline
\end{array}
\]

Constraint:
\[
\text{Sum}\{S.\text{price}\} < 5
\]
The Constrained Apriori Algorithm: Push a Succinct Constraint Deep

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Scan D

C₁

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1\} & 2 \\
\{2\} & 3 \\
\{3\} & 3 \\
\{4\} & 1 \\
\{5\} & 3 \\
\hline
\end{array}
\]

L₁

C₂

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1 2\} & 1 \\
\{1 3\} & 2 \\
\{1 5\} & 1 \\
\{2 3\} & 2 \\
\{2 5\} & 3 \\
\{3 5\} & 2 \\
\hline
\end{array}
\]

Scan D

C₃

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{2 3 5\} & 2 \\
\hline
\end{array}
\]

L₃

constraint: \( \min\{S.\text{price}\} \leq 1 \)

not immediately to be used
Converting “Tough” Constraints

- Convert tough constraints into anti-monotone or monotone by properly ordering items
- Examine C: \( \text{avg}(S.\text{profit}) \geq 25 \)
  - Order items in value-descending order
    - \(<a, f, g, d, b, h, c, e>\)
  - If an itemset \( afb \) violates C
    - So does \( afbh, afb^* \)
  - It becomes anti-monotone!

TDB (min_sup=2)

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>a, c, d, e, f</td>
</tr>
<tr>
<td>40</td>
<td>c, e, f, g</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Strongly Convertible Constraints

- \( \text{avg}(X) \geq 25 \) is convertible anti-monotone w.r.t. item value descending order \( R: <a, f, g, d, b, h, c, e> \)
  - If an itemset \( af \) violates a constraint \( C \), so does every itemset with \( af \) as prefix, such as \( afd \)

- \( \text{avg}(X) \geq 25 \) is convertible monotone w.r.t. item value ascending order \( R^{-1}: <e, c, h, b, d, g, f, a> \)
  - If an itemset \( d \) satisfies a constraint \( C \), so does itemsets \( df \) and \( dfa \), which having \( d \) as a prefix

Thus, \( \text{avg}(X) \geq 25 \) is strongly convertible

<table>
<thead>
<tr>
<th>Item</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Can Apriori Handle Convertible Constraint?

- A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm
  - Within the level wise framework, no direct pruning based on the constraint can be made
  - Itemset df violates constraint C: \( \text{avg}(X) \geq 25 \)
  - Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned
- But it can be pushed into frequent-pattern growth framework!

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
</tbody>
</table>
Mining With Convertible Constraints

- C: \( \text{avg}(X) \geq 25, \min\text{_sup}=2 \)
- List items in every transaction in value descending order \( R: \langle a, f, g, d, b, h, c, e \rangle \)
  - C is convertible anti-monotone w.r.t. \( R \)
- Scan TDB once
  - remove infrequent items
    - Item h is dropped
  - Itemsets a and f are good, ...
- Projection-based mining
  - Imposing an appropriate order on item projection
  - Many tough constraints can be converted into (anti)-monotone

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>40</td>
</tr>
<tr>
<td>f</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>20</td>
</tr>
<tr>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>-10</td>
</tr>
<tr>
<td>c</td>
<td>-20</td>
</tr>
<tr>
<td>e</td>
<td>-30</td>
</tr>
</tbody>
</table>

TDB (\( \min\text{_sup}=2 \))

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, f, d, b, c</td>
</tr>
<tr>
<td>20</td>
<td>f, g, d, b, c</td>
</tr>
<tr>
<td>30</td>
<td>a, f, d, c, e</td>
</tr>
<tr>
<td>40</td>
<td>f, g, h, c, e</td>
</tr>
</tbody>
</table>
Handling Multiple Constraints

- Different constraints may require different or even conflicting item-ordering

- If there exists an order $R$ s.t. both $C_1$ and $C_2$ are convertible w.r.t. $R$, then there is no conflict between the two convertible constraints

- If there exists conflict on order of items
  - Try to satisfy one constraint first
  - Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database
### What Constraints Are Convertible?

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Convertible anti-monotone</th>
<th>Convertible monotone</th>
<th>Strongly convertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{avg}(S) \leq , \geq v )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{median}(S) \leq , \geq v )</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>( \text{sum}(S) \leq v ) (items could be of any value, ( v \geq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v ) (items could be of any value, ( v \leq 0 ))</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>( \text{sum}(S) \geq v ) (items could be of any value, ( v \geq 0 ))</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

......
## Constraint-Based Mining—A General Picture

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Antimonotone</th>
<th>Monotone</th>
<th>Succinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v \in S$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$S \subseteq V$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{min}(S) \leq v$</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{max}(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$\text{max}(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{count}(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>$\text{count}(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>$\text{sum}(S) \leq v \ (a \in S, \ a \geq 0)$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{sum}(S) \geq v \ (a \in S, \ a \geq 0)$</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$\text{range}(S) \leq v$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{range}(S) \geq v$</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>$\text{avg}(S) \theta v, \ \theta \in {=, \leq, \geq}$</td>
<td>convertible</td>
<td>convertible</td>
<td>no</td>
</tr>
<tr>
<td>$\text{support}(S) \geq \xi$</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\text{support}(S) \leq \xi$</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
A Classification of Constraints

- Convertible anti-monotone
- Antimonotone
- Inconvertible
- Strongly convertible
- Monotone
- Convertible monotone

Succinct
Chapter 5: Mining Frequent Patterns, Association and Correlations

- Basic concepts and a road map
- Efficient and scalable frequent itemset mining methods
- Mining various kinds of association rules
- From association mining to correlation analysis
- Constraint-based association mining
- Summary
Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Projection-based (FPgrowth, CLOSET+, ...)
  - Vertical format approach (CHARM, ...)
- Mining a variety of rules and interesting patterns
- Constraint-based mining
- Mining sequential and structured patterns
- Extensions and applications
Frequent-Pattern Mining: Research Problems

- Mining fault-tolerant frequent, sequential and structured patterns
  - Patterns allows limited faults (insertion, deletion, mutation)
- Mining truly interesting patterns
  - Surprising, novel, concise, ...
- Application exploration
  - E.g., DNA sequence analysis and bio-pattern classification
  - “Invisible” data mining
Ref: Basic Concepts of Frequent Pattern Mining

- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95
Ref: Apriori and Its Improvements

Ref: Depth-First, Projection-Based FP Mining

- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD’00.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
Ref: Vertical Format and Row Enumeration Methods

- Zaki and Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM'02.

J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95.


Ref: Mining Correlations and Interesting Rules


- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02.

- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE’03.

- Y. K. Lee, W.Y. Kim, Y. D. Cai, and J. Han. CoMine: Efficient Mining of Correlated Patterns. ICDM’03.
Ref: Mining Other Kinds of Rules

- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT’02.
Ref: Constraint-Based Pattern Mining

- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang. Exploratory mining and pruning optimizations of constrained association rules. SIGMOD’ 98.
- J. Pei, J. Han, and L. V. S. Lakshmanan. Mining Frequent Itemsets with Convertible Constraints. ICDE'01.
- J. Pei, J. Han, and W. Wang, Mining Sequential Patterns with Constraints in Large Databases, CIKM'02.
Ref: Mining Sequential and Structured Patterns

- J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE'01.
- M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. ICDM'01.
- X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM'03.
- X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. KDD'03.
Ref: Mining Spatial, Multimedia, and Web Data

- K. Koperski and J. Han, Discovery of Spatial Association Rules in Geographic Information Databases, SSD’95.

- O. R. Zaiane, M. Xin, J. Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. ADL'98.

- O. R. Zaiane, J. Han, and H. Zhu, Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. ICDE'00.

- D. Gunopulos and I. Tsoukatos. Efficient Mining of Spatiotemporal Patterns. SSTD'01.
Ref: Mining Frequent Patterns in Time-Series Data

- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98.
- J. Han, G. Dong and Y. Yin, Efficient Mining of Partial Periodic Patterns in Time Series Database, ICDE'99.
- H. Lu, L. Feng, and J. Han. Beyond Intra-Transaction Association Analysis: Mining Multi-Dimensional Inter-Transaction Association Rules. TOIS:00.
- B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-Evolving Time Sequences. ICDE'00.
Ref: Iceberg Cube and Cube Computation


- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD'97.


- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98.

Ref: Iceberg Cube and Cube Exploration

- J. Han, J. Pei, G. Dong, and K. Wang, Computing Iceberg Data Cubes with Complex Measures. SIGMOD’01.
- G. Dong, J. Han, J. Lam, J. Pei, and K. Wang. Mining Multi-Dimensional Constrained Gradients in Data Cubes. VLDB'01.
- L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient Cube: How to Summarize the Semantics of a Data Cube. VLDB'02.
- D. Xin, J. Han, X. Li, B. W. Wah. Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration. VLDB'03.
Ref: FP for Classification and Clustering

- B. Liu, W. Hsu, Y. Ma. Integrating Classification and Association Rule Mining. KDD’98.
- W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. ICDM'01.
- J. Yang and W. Wang. CLUSEQ: efficient and effective sequence clustering. ICDE’ 03.
- B. Fung, K. Wang, and M. Ester. Large Hierarchical Document Clustering Using Frequent Itemset. SDM’ 03.
- X. Yin and J. Han. CPAR: Classification based on Predictive Association Rules. SDM'03.
Ref: Stream and Privacy-Preserving FP Mining

- G. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. VLDB’02.
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-Dimensional Regression Analysis of Time-Series Data Streams. VLDB'02.
- C. Giannella, J. Han, J. Pei, X. Yan and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities, Next Generation Data Mining:03.
Ref: Other Freq. Pattern Mining Applications


- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.