Chapters 6 & 7, Frequent Pattern Mining

Young-Rae Cho
Associate Professor
Department of Computer Science
Baylor University

- Market Basket Problem
- Apriori Algorithm
- CHARM Algorithm
- Advanced Frequent Pattern Mining
- Advanced Association Rule Mining
- Constraint-Based Association Mining
Market Basket Problem

- **Example**
  - "Customers who bought beer also bought diapers."

- **Motivation**
  - To promote sales in retail by cross-selling

- **Required Data**
  - Customers’ purchase patterns
    (Items often purchased together)

- **Applications**
  - Store arrangement
  - Catalog design
  - Discount plans

Solving Market Basket Problem

- **Basic Terms**
  - Transaction:
    a set of items (which are bought by one person at one time)
  - Frequent itemset:
    a set of items (as a subset of a transaction) which occur frequently across transactions
  - Association rule:
    one-direction relationship between two sets of items (e.g., A → B)

- **Process**
  - Step 1, Generation of frequent itemsets
    e.g. \{ beer, nuts, diapers \}
  - Step 2, Generation of association rules
    e.g. \{ beer \} → \{ nuts, diapers \} → expected output knowledge
Frequent Itemsets

- **Transaction Table**

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bread, eggs, milk, diapers</td>
</tr>
<tr>
<td>2</td>
<td>coke, beer, nuts, diapers</td>
</tr>
<tr>
<td>3</td>
<td>eggs, juice, beer, nuts</td>
</tr>
<tr>
<td>4</td>
<td>milk, beer, nuts, diapers</td>
</tr>
<tr>
<td>5</td>
<td>milk, beer, diapers</td>
</tr>
</tbody>
</table>

- **Support**
  - Frequency of a set of items across transactions
  - \{ milk, diapers \}, \{ beer, nuts \}, \{ beer, diapers \} → 60% support
  - \{ milk, beer, diapers \}, \{ beer, nuts, diapers \} → 40% support

- **Frequent Itemsets**
  - Itemsets having support greater than (or equal to) a user-specified **minimum support**

Association Rules

- **Frequent Itemsets**
  - (min sup = 60%, size ≥ 2)
    - \{ milk, diapers \}
    - \{ beer, nuts \}
    - \{ beer, diapers \}

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bread, eggs, milk, diapers</td>
</tr>
<tr>
<td>2</td>
<td>coke, beer, nuts, diapers</td>
</tr>
<tr>
<td>3</td>
<td>eggs, juice, beer, nuts</td>
</tr>
<tr>
<td>4</td>
<td>milk, beer, nuts, diapers</td>
</tr>
<tr>
<td>5</td>
<td>milk, beer, diapers</td>
</tr>
</tbody>
</table>

- **Confidence**
  - For \( A \rightarrow B \), percentage of transactions containing \( A \) that also contain \( B \)
  - \{milk\} → \{diapers\}, \{nuts\} → \{beer\} : 100% confidence
  - \{diapers\} → \{milk\}, \{beer\} → \{nuts\}, \{beer\} → \{diapers\}, and \{diapers\} → \{beer\} : 75% confidence

- **Association Rules**
  - Rules having confidence greater than (or equal to) a user-specified **minimum confidence**
Generalized Formulas

- **Association Rules**
  - \( I = \{ I_1, I_2, \ldots, I_m \}, \ T = \{ T_1, T_2, \ldots, T_n \}, \ T_k \subseteq I \) for \( k \) for \( k \)
  - \( A \rightarrow B \) where \( A \subseteq I (A \neq \emptyset), B \subseteq I (B \neq \emptyset), A \subseteq T_i \) for \( i \), \( B \subseteq T_j \) for \( j \), and \( A \cap B = \emptyset \)

- **Computation of Support**
  - \( \text{support} (A \rightarrow B) = \frac{P(A \cup B)}{n} \) where \( P(X) = |\{T_i \mid X \subseteq T_i\}| / n \)

- **Computation of Confidence**
  - \( \text{confidence} (A \rightarrow B) = \frac{P(B \mid A)}{P(A)} = \frac{P(A \cup B)}{P(A)} \)

Problem of Support & Confidence

- **Support Table**

<table>
<thead>
<tr>
<th></th>
<th>Tea</th>
<th>Not Tea</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>20</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>Not Coffee</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>SUM</td>
<td>30</td>
<td>70</td>
<td>100</td>
</tr>
</tbody>
</table>

- **Association Rule, \{Tea\} \rightarrow \{Coffee\}**
  - Support (\{Tea\} \rightarrow \{Coffee\}) ?
  - Confidence (\{Tea\} \rightarrow \{Coffee\}) ?

- **Problems in this Dataset ?**
Alternative Measures

- **Lift**

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

  - The association rule \( A \rightarrow B \) is interesting if \( \text{lift}(A \rightarrow B) > 1 \)
  - However, it is the same to \text{correlation} between \( A \) and \( B \)

- **Correlation**

  \[
  \text{lift } (A \rightarrow B) = \frac{P (A \cup B)}{P (A) \times P (B)} = \text{correlation } (A, B)
  \]

  - Positive correlation if \( \text{correlation}(A,B) > 1 \)
  - Negative correlation if \( \text{correlation}(A,B) < 1 \)
  - \( A \leftrightarrow B \)

---

**Alternative Measures – cont’d**

- **\( \chi^2 \) Test (Chi-Square Test)**

  - Evaluates whether an observed distribution in a sample differs from a theoretical distribution (i.e., hypothesis).
  - Where \( E_i \) is an expected frequency and \( O_i \) is an observed frequency,
    \[
    \chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}
    \]
  - The larger \( \chi^2 \), the more likely the variables are related (positively or negatively).

- **Coverage**

  \[
  \text{coverage } (A \rightarrow B) = P(A)
  \]
Chapters 6 & 7, Frequent Pattern Mining

- Market Basket Problem
  - Apriori Algorithm
- CHARM Algorithm
- Advanced Frequent Pattern Mining
- Advanced Association Rule Mining
- Constraint-Based Association Mining

Frequent Itemset Mining

- Process
  1. Find frequent itemsets → computational problem
  2. Find association rules

- Brute Force Algorithm for Frequent Itemset Generation
  - Enumerate all possible subsets of the total itemset, $I$
  - Count frequency of each subset
  - Select frequent itemsets

- Problem
  - Enumerating all candidates is not computationally acceptable
    → Efficient & scalable algorithm is required.
Apriori Algorithm

- **Motivations**
  - Efficient frequent itemset analysis
  - Scalable approach

- **Process**
  - Iterative increment of the itemset size
    1. Candidate itemset generation → computational problem
    2. Frequent itemset selection

- **Downward Closure Property**
  - Any superset of an itemset \( X \) cannot have higher support than \( X \).
  - If an itemset \( X \) is frequent (support of \( X \) is higher than min. sup.),
    then any subset of \( X \) should be frequent.

---

Candidate Itemset Generation

- **Process**
  - Two steps: (1) **selective joining** and (2) **a priori pruning**

- **Selective Joining**
  - Each candidate itemset with size \( k \) is generated by joining two frequent itemsets with size \((k-1)\)
  - The frequent itemsets with size \((k-1)\) which share a frequent sub-itemset with size \((k-2)\) are joined

- **A priori Pruning**
  - A frequent itemset with size \( k \) which has any infrequent sub-itemsets with size \((k-1)\) is pruned
Detail of Apriori Algorithm

- **Basic Terms**
  - $C_k$: Candidate itemsets of size $k$
  - $L_k$: Frequent itemsets of size $k$
  - $\text{sup}_{\min}$: Minimum support

- **Pseudo Code**
  
  $$k \leftarrow 1$$
  
  $$L_k \leftarrow \text{frequent itemsets with size 1}$$
  
  while $L_k \neq \emptyset$ do
    
    $$k \leftarrow k + 1$$
    
    $$C_k \leftarrow \text{candidate itemsets by selective joining & a priori pruning from } L_{(k-1)}$$
    
    $$L_k \leftarrow \text{frequent itemsets using } \text{sup}_{\min}$$
  
  end while

  return $U_k L_k$

Example of Apriori Algorithm

- $\text{sup}_{\min} = 2$

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
<th>Sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
<td>3</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
<td>3</td>
</tr>
</tbody>
</table>

<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>

<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>

<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,C,E}</td>
<td>3</td>
</tr>
<tr>
<td>{B,E}</td>
<td>2</td>
</tr>
<tr>
<td>{C,E}</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B,C,E}</td>
<td>2</td>
</tr>
</tbody>
</table>
Summary of Apriori Algorithm

**Features**
- An iterative approach of a level-wise search
- Reducing search space by downward closure property

**References**

Challenges of Apriori Algorithm

**Challenges**
- Multiple scan of transaction database
- Huge number of candidates
- Tedious workload of support counting

**Solutions**
- Reducing transaction database scans
- Shrinking number of candidates
- Facilitating support counting
Chapters 6 & 7, Frequent Pattern Mining

- Market Basket Problem
- Apriori Algorithm
  - CHARM Algorithm
- Advanced Frequent Pattern Mining
- Advanced Association Rule Mining
- Constraint-Based Association Mining

Association Rule Mining

- Process
  1. Find frequent itemsets → computational problem
  2. Find association rules → redundant rule generation

- Example 1
  - \{ beer \} → \{ nuts \} (40% support, 75% confidence)
  - \{ beer \} → \{ nuts, diapers \} (40% support, 75% confidence)
  - The first rule is not meaningful.

- Example 2
  - \{ beer \} → \{ nuts \} (60% support, 75% confidence)
  - \{ beer, diapers \} → \{ nuts \} (40% support, 75% confidence)
  - Both rules are meaningful.
Frequent Closed Itemsets

**General Definition of Closure**
- A frequent itemset $X$ is **closed** if there exists no superset of $X$, $Y \supseteq X$, with the same support as $X$.
- Different from frequent **maximal** itemsets

**Frequent Closed Itemsets with Min. Support of 40%**
- { milk, diapers } 60%
- { milk, beer } 60%
- { beer, nuts } 60%
- { beer, diapers } 60%
- { nuts, diapers } 60%
- { milk, beer, diapers } 40%
- { beer, nuts, diapers } 40%

### T-ID Items

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bread, eggs, milk, diapers</td>
</tr>
<tr>
<td>2</td>
<td>coke, beer, nuts, diapers</td>
</tr>
<tr>
<td>3</td>
<td>eggs, juice, beer, nuts</td>
</tr>
<tr>
<td>4</td>
<td>milk, beer, nuts, diapers</td>
</tr>
<tr>
<td>5</td>
<td>milk, beer, diapers</td>
</tr>
</tbody>
</table>

Mapping of Items and Transactions

**Mapping Functions**
- $I = \{ I_1, I_2, \ldots, I_m \}$, $T = \{ T_1, T_2, \ldots, T_n \}$, $X \subseteq I$, $Y \subseteq T$
- $i: T \rightarrow I$, $i(Y)$: itemset that is contained in all transactions in $Y$
- $t: I \rightarrow T$, $t(X)$: set of transactions (tidset) that contain all items in $X$

**Properties**
- $X_1 \subseteq X_2 \rightarrow t(X_1) \supseteq t(X_2)$
  - (e.g.) $\{ACW\} \subseteq \{ACTW\} \rightarrow \{1345\} \supseteq \{135\}$
- $Y_1 \subseteq Y_2 \rightarrow i(Y_1) \supseteq i(Y_2)$
  - (e.g.) $\{245\} \subseteq \{2456\} \rightarrow \{CDW\} \supseteq \{CD\}$
- $X \subseteq i(t(X))$, $Y \subseteq t(i(Y))$
  - (e.g.) $t(\{AC\}) = \{1345\}$, $i(\{1345\}) = \{ACW\}$
  - (e.g.) $i(\{134\}) = \{ACW\}$, $t(\{ACW\}) = \{1345\}$
Definition of Closure

- **Closure Operator**
  - \( c_i(X) = i( t(X) ) \), \( c_o(Y) = t( i(Y) ) \)

- **Formal Definition of Closure**
  - An itemset \( X \) is **closed** if \( X = c_i(X) \)
  - A tid-set \( Y \) is **closed** if \( Y = c_o(Y) \)

Examples of Closed Itemsets

- **Examples**
  - \( X = \{ACW\} \)  
    - \( t(X) = \{1345\} \), \( i( t(X) ) = \{ACW\} \)  
    - \( X \) is closed.
  - \( X = \{AC\} \)  
    - \( t(X) = \{1345\} \), \( i( t(X) ) = \{ACW\} \)  
    - \( X \) is not closed.
  - \( X = \{ACT\} \)  
    - \( t(X) = \{135\} \), \( i( t(X) ) = \{ACTW\} \)  
    - \( X \) is not closed.
  - \( X = \{CT\} \)  
    - \( t(X) = \{1356\} \), \( i( t(X) ) = \{CT\} \)  
    - \( X \) is closed.
CHARM Algorithm

- **Motivations**
  - Efficient frequent closed itemset analysis
  - Non-redundant rule generation

- **Property**
  - Simultaneous exploration of itemset space and tid-set space
  - Not enumerating all possible subsets of a closed itemset
  - Early pruning strategy for infrequent and non-closed itemsets

- **Process**
  - for each itemset pair
    1. computing the frequency of their union set
    2. pruning all infrequent and non-closed branches

---

Frequency Computation

- **Operation**
  - Tid-set of the union of two itemsets, $X_1$ and $X_2$
  - Intersection of two tid-sets, $t(X_1)$ and $t(X_2)$

\[
 t(X_1 \cup X_2) = t(X_1) \cap t(X_2)
\]

- **Example**
  - $X_1 = \{AC\}$, $X_2 = \{D\}$
  - $t(X_1 \cup X_2) = t(\{ACD\}) = \{45\}$
  - $t(X_1) \cap t(X_2) = \{1345\} \cap \{2456\} = \{45\}$

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A, C, T, W</td>
</tr>
<tr>
<td>2</td>
<td>C, D, W</td>
</tr>
<tr>
<td>3</td>
<td>A, C, T, W</td>
</tr>
<tr>
<td>4</td>
<td>A, C, D, W</td>
</tr>
<tr>
<td>5</td>
<td>A, C, D, T, W</td>
</tr>
<tr>
<td>6</td>
<td>C, D, T</td>
</tr>
</tbody>
</table>
Pruning Strategy

**Pruning**
- Suppose two itemsets $X_1 \leq X_2$

1. $t(X_1) = t(X_2) \quad \rightarrow \quad t(X_1) \cap t(X_2) = t(X_1) = t(X_2) \\
   \quad \rightarrow \quad$ Replace $X_1$ with $(X_1 \cup X_2)$, and prune $X_2$

2. $t(X_1) \subset t(X_2) \quad \rightarrow \quad t(X_1) \cap t(X_2) = t(X_1) \neq t(X_2) \\
   \quad \rightarrow \quad$ Replace $X_1$ with $(X_1 \cup X_2)$, and keep $X_2$

3. $t(X_1) \supset t(X_2) \quad \rightarrow \quad t(X_1) \cap t(X_2) = t(X_2) \neq t(X_1) \\
   \quad \rightarrow \quad$ Replace $X_2$ with $(X_1 \cup X_2)$, and keep $X_1$

4. $t(X_1) \neq t(X_2) \quad \rightarrow \quad t(X_1) \cap t(X_2) \neq t(X_1) \neq t(X_2) \\
   \quad \rightarrow \quad$ Keep $X_1$ and $X_2$

Example of CHARM Algorithm

**Subset Lattice**

T-ID | Items  
---|---
1 | A, C, T, W  
2 | C, D, W  
3 | A, C, T, W  
4 | A, C, D, W  
5 | A, C, D, T, W  
6 | C, D, T
Summary of CHARM Algorithm

➢ Advantages
  ▪ No need multiple scan of transaction database
    → Revision and enhancement of Apriori algorithm
  ▪ No loss of information

➢ References

CSI 4352, Introduction to Data Mining

**Chapters 6 & 7, Frequent Pattern Mining**

➢ Market Basket Problem
➢ Apriori Algorithm
➢ CHARM Algorithm
  ➢ Advanced Frequent Pattern Mining
➢ Advanced Association Rule Mining
➢ Constraint-Based Association Mining
Frequent Pattern Mining

Definition
- A frequent pattern: A pattern (a set of items, sub-sequences, sub-structures, etc.) that occurs frequently in a data set

Motivation
- Finding inherent regularities in data
  - e.g., What products were often purchased together?
  - e.g., What are the subsequent purchases after buying a PC?
  - e.g., What kinds of DNA sequences are sensitive to this new drug?
  - e.g., Can we find web documents similar to my research?

Applications
- Market basket analysis, DNA sequence analysis, Web log analysis

Why Frequent Pattern Mining?

Importance
- A frequent pattern is an intrinsic and important property of data sets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (sub-graph) pattern analysis
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Data pre-processing: data reduction and compression
  - Data warehousing: iceberg cube computation
Sampling Approach

- **Motivation**
  - Problem: Typically huge data size
  - Mining a subset of the data to reduce candidate search space
  - Trade-off some degree of accuracy against efficiency

- **Process**
  1. Selecting a set of random samples from the original database
  2. Mining frequent itemsets with the set of samples using Apriori
  3. Verifying the frequent itemsets on the border of closure of frequent itemsets

- **Reference**

Partitioning Approach

- **Motivation**
  - Problem: Typically huge data size
  - Partitioning data to reduce candidate search space

- **Process**
  1. Partitioning database and find local frequent patterns
  2. Consolidating global frequent patterns

- **Reference**
Hashing Approach

**Motivation**
- Problem: A very large number of candidates generated
- The process in the initial iteration (e.g., size-2 candidate generation) dominates the total execution cost
- Hashing itemsets to reduce the size of candidates

**Process**
1. Hashing itemsets into several buckets in a hash table
2. If a $k$-itemset whose corresponding hashing bucket count is below the min support, it cannot be frequent, thus should be removed

**Reference**

Pattern Growth Approach

**Motivation**
- Problem: A very large number of candidates generated
- Finding frequent itemsets without candidate generation
- Grows short patterns to long ones using local frequent items only
- Depth-first search
  (Apriori: Breadth-first search, Level-wise search)

**Example**
- "abc" is a frequent pattern
- "d" is a frequent item $\rightarrow$ "abcd" is a frequent pattern

**Reference**
- Han, J., Pei, J. and Yin, Y. "Mining frequent patterns without candidate generation." *In Proceedings of SIGMOD* (2000)
FP-Frequent Pattern)-Tree

- **FP-Tree Construction**
  - Scan DB once to find all frequent 1-itemsets
  - Sort frequent items in a descending order of support, called f-list
  - Scan DB again to construct FP-tree

- **Example**

<table>
<thead>
<tr>
<th>TID</th>
<th>items bought</th>
<th>ordered frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f, a, c, d, g, i, m, p}</td>
<td>{f, c, a, m, p}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c, f, l, m, o}</td>
<td>{f, c, a, b, m}</td>
</tr>
<tr>
<td>300</td>
<td>{b, f, h, j, a, w}</td>
<td>{f, b}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, k, s, p}</td>
<td>{c, b, p}</td>
</tr>
<tr>
<td>500</td>
<td>{a, f, c, e, l, p, m, n}</td>
<td>{f, c, a, m, p}</td>
</tr>
</tbody>
</table>

  Item Support Link
  - min_support = 3

- **Benefits of FP Tree Structure**
  - **Compactness**
    - Reduce irrelevant (infrequent) items
    - Reduce common prefix items of patterns
    - Order items in the descending order of support
      (The more frequent occurring, the more likely to be shared.)
    - Never be larger than the original database
  - **Completeness**
    - Preserve complete information of frequent patterns
    - Never break any long patterns
Conditional Pattern Bases

- **Conditional Pattern Base Construction**
  - Traverse the FP-tree by following the link of each frequent item \( p \)
  - Accumulate all prefix paths of \( p \) to form \( p \)'s conditional pattern base

- **Example**

```
Item | Support | Link
-----|---------|------
f    | 4       |      
c    | 4       |      
a    | 3       |      
b    | 3       |      
m    | 3       |      
p    | 3       |      

\{ \}                      \{ \}
\{f\}                     \{f, p\}
c:3                      c:1
b:1                      b:1
p:1

item    conditional pattern base
f      -
c      f:3
a      fc:3
b      fca:1, f, c:1
m      fca:2, fca:b:1
p      fcam:2, cb:1
```

Conditional FP-Trees

- **Conditional FP-Tree Construction**
  - For each pattern base, accumulate the count for each item
  - Construct the conditional FP-tree with frequent items of the pattern base

- **Example**

```
Item | Support | Link
-----|---------|------
f    | 4       |      
c    | 4       |      
a    | 3       |      
b    | 3       |      
m    | 3       |      
p    | 3       |      

\{ \}                      \{ \}
\{m\}                      \{f\}
f:3                       f:3
m:2                       m:1
b:1
p:1

frequent patterns
m, fm, cm, am,
fcm, fam, cam,
fcam
```

BAYLOR
Algorithm of Pattern Growth Mining

- **Algorithm**
  1. Construct FP tree
  2. For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  3. Repeat (2) recursively on each newly created conditional FP-tree until the resulting FP-tree is empty, or it contains only a single path
  4. The single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

- **Advanced Techniques**
  - To fit an FP-tree in memory, partitioning a database into a set of projected databases
  - Mining the FP-tree for each projected database

Mining Frequent Maximal Patterns

- **1st Round**
  - A, B, C, D, E

- **2nd Round**
  - AB, AC, AD, AE, ABCDE
  - BC, BD, BE, BCDE
  - CD, CE, CDE
  - DE

- **3rd Round**
  - AED

- **Reference**
Chapters 6 & 7, Frequent Pattern Mining

- Market Basket Problem
- Apriori Algorithm
- CHARM Algorithm
- Advanced Frequent Pattern Mining
  - Advanced Association Rule Mining
- Constraint-Based Association Mining

Mining Multi-Level Association Rules

**Motivation**
- Items often form hierarchies
- Setting flexible supports
  - Items at the lower level are expected to have lower support

**Example**

<table>
<thead>
<tr>
<th>(uniform support)</th>
<th>(reduced support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1&lt;br&gt;min_sup = 5%&lt;br&gt;Milk [support = 10%]&lt;br&gt;Level 2&lt;br&gt;min_sup = 5%&lt;br&gt;2% Milk [support = 6%]&lt;br&gt;Skim Milk [support = 4%]</td>
<td>Level 1&lt;br&gt;min_sup = 5%&lt;br&gt;Milk [support = 10%]&lt;br&gt;Level 2&lt;br&gt;min_sup = 3%&lt;br&gt;Skin Milk [support = 4%]</td>
</tr>
</tbody>
</table>
Redundant Rule Filtering

- **Motivation**
  - Some rules may be redundant due to "ancestor" relationships between items

- **Example**
  - \{ milk \} → \{ bread \}  [support=10%, confidence=70%]
  - \{ 2% milk \} → \{ bread \}  [support=6%, confidence=72%]
  - \{ milk \} → \{ wheat bread \}  [support=5%, confidence=38%]

Redundant if its support and confidence are close to "expected" value based on the rule's ancestor.

Mining Multidimensional Association Rules

- **Single Dimensional Association Rules**
  - buys("milk") → buys("bread")

- **Multi-Dimensional Association Rules**
  - Rules with more than 2 dimensions
  - Inter-dimensional association rules:
    - ages("18-25") ^ occupation("student") → buys("coke")
  - Hybrid-dimensional association rules:
    - ages("18-25") ^ buys("popcorn") → buys("coke")

- **Attributes in Association Rules**
  - Categorical attributes
  - Quantitative attributes → discretization
Quantitative Rule Mining

**Motivation**
- The range of numeric attributes can be changed dynamically to maximize the confidence

**Example**
- `age("32-38") ^ income("40k-50K") → buys("HDTV")`
- Binning to partition the range
- Grouping the adjacent ranges
- The combination of grouping and binning
- New rule:
  - `age("34-35") ^ income("30k-50k")`
  - `→ buys("HDTV")`

---

**CSI 4352, Introduction to Data Mining**

**Chapters 6 & 7, Frequent Pattern Mining**

- Market Basket Problem
- Apriori Algorithm
- CHARM Algorithm
- Advanced Frequent Pattern Mining
- Advanced Association Rule Mining
- **Constraint-Based Association Mining**
Constraint-based Mining

- **Motivation**
  - Finding all the patterns (association rules) in a database?
    - Too many, diverse patterns
  - Users can give directions (constraints) for mining patterns

- **Properties**
  - User flexibility
    - Users can provide any constraints on what to be mined
  - System optimization
    - It reduces the search space for efficient mining

Constraint Types

- **Knowledge Type Constraints**
  - Association, Classification, etc.

- **Data Constraints**
  - Selects data having specific values using SQL-like queries
    - ex, sales in Waco on Sep

- **Dimension/Level Constraints**
  - Selects specific dimensions or levels of the concept hierarchies

- **Interestingness Constraints**
  - Uses interestingness measures, ex, support, confidence, correlation

- **Rule Constraints**
  - Specifies rules (or meta-rules) to be mined
Meta-Rule Constraints

- **Definition**
  - Rule templates using the maximum or minimum number of predicates occurring in a rule, or the relationships among attributes, attribute values or aggregates

- **Meta-Rule-Guided Mining**
  - Finding rules between the customer attributes (age, address, credit rate) and the item purchased
  - $P_1(X) \land P_2(Y) \rightarrow \text{buys("coke")}$
  - Sample result: $\text{age("18~25") \land income("30k~40k") \rightarrow \text{buys("coke")}}$

Rule Constraint Types

- **Anti-monotonic Constraints**
  - If a constraint $c$ is violated, then its further mining is terminated

- **Monotonic Constraints**
  - If a constraint $c$ is satisfied, then its further mining is redundant

- **Succinct Constraints**
  - The itemsets satisfying a constraint $c$ can be directly generated

- **Convertible Constraints**
  - A constraint $c$ is not monotonic nor anti-monotonic, but it can be converted if items are properly ordered
Anti-Monotonicity in Constraints

**Definition**

- A constraint \( c \) is *anti-monotonic*,
  if a pattern satisfies \( c \),
  all of its sub-patterns satisfy \( c \) too
- If an itemset \( S \) violates \( c \),
  so does any of its supersets

**Examples**

- \( \text{count}(S) < 3 \) → Anti-monotonic
- \( \text{count}(S) \geq 4 \) → Not anti-monotonic
- \( \text{sum}(S.\text{price}) \leq 100 \) → Anti-monotonic
- \( \text{sum}(S.\text{price}) \geq 150 \) → Not anti-monotonic
- \( \text{sum}(S.\text{profit}) \leq 80 \) → Not anti-monotonic
- \( \text{support}(S) \geq 2 \) → Anti-monotonic

<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>Price</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>60</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>100</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>95</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>100</td>
<td>-10</td>
</tr>
</tbody>
</table>

Monotonicity in Constraints

**Definition**

- A constraint \( c \) is *monotonic*,
  if a pattern satisfies \( c \),
  all of its super-patterns satisfy \( c \) too
- If an itemset \( S \) satisfies \( c \),
  so does any of its supersets

**Examples**

- \( \text{count}(S) \geq 2 \) → Monotonic
- \( \text{sum}(S.\text{price}) \leq 100 \) → Not monotonic
- \( \text{sum}(S.\text{price}) \geq 150 \) → Monotonic
- \( \text{sum}(S.\text{profit}) \geq 100 \) → Not monotonic
- \( \text{min}(S.\text{price}) \leq 80 \) → Monotonic
- \( \text{min}(S.\text{price}) > 70 \) → Not monotonic

<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>Price</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>60</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>100</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>95</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>100</td>
<td>-10</td>
</tr>
</tbody>
</table>
Succinctness in Constraints

**Definition**
- A constraint $c$ is succinct, if an itemset satisfying $c$ can be generated before support counting.
- All and only the itemsets satisfying $c$ can be enumerated.

**Examples**
- $\min(S.price) < 80 \rightarrow \text{succinct}$
- $\sum(S.price) \geq 150 \rightarrow \text{Not succinct}$

<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>Price</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>60</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>100</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>95</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>100</td>
<td>-10</td>
</tr>
</tbody>
</table>

Converting Constraints

**Definition**
- A constraint $c$ is convertible, if $c$ is not anti-monotonic nor monotonic, but $c$ becomes anti-monotonic or monotonic when items are properly ordered.

**Example**
- $\text{avg}(S.price) > 80$
- $\rightarrow$ Neither anti-monotonic nor monotonic
- $\rightarrow$ if items are in a value-descending order $<a, e, h, g, d, f, c, b>$, then anti-monotonic
- $\rightarrow$ if items are in a value-ascending order $<b, c, f, d, g, a, e, h>$, then monotonic
- $\rightarrow$ (strongly) Convertible

<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>Price</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>b</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>60</td>
<td>-20</td>
</tr>
<tr>
<td>d</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>e</td>
<td>100</td>
<td>-30</td>
</tr>
<tr>
<td>f</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>95</td>
<td>20</td>
</tr>
<tr>
<td>h</td>
<td>100</td>
<td>-10</td>
</tr>
</tbody>
</table>
Anti-Monotonic Constraints in Apriori

- Handling Anti-monotonic Constraints
  - Can apply apriori pruning
  - Example: \( \text{sum(S.price)} < 5 \)

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>20</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>30</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>40</td>
<td>2, 5</td>
</tr>
</tbody>
</table>

C_f

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

L_f

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

C_2

Monotonic Constraints in Apriori

- Handling Monotonic Constraints
  - Cannot apply apriori pruning
  - Example: \( \text{sum(S.price)} \geq 3 \)

<table>
<thead>
<tr>
<th>T-ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>20</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>30</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>40</td>
<td>2, 5</td>
</tr>
</tbody>
</table>

C_f

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

L_f

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

C_2

\[
\begin{array}{|c|c|}
\hline
\text{Itemset} & \text{Sup.} \\
\hline
\{1, 2\} & 1 \\
\{1, 3\} & 2 \\
\{2, 3\} & 2 \\
\{1, 5\} & 1 \\
\{2, 5\} & 2 \\
\{3, 5\} & 1 \\
\hline
\end{array}
\]
### Examples of Constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Anti-Monotone</th>
<th>Monotone</th>
<th>Succinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>v ∈ S</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>S ⊆ V</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>S ⊊ V</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>min(S) ≤ v</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>min(S) ≥ v</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>max(S) ≤ v</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>max(S) ≥ v</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>count(S) ≤ v</td>
<td>yes</td>
<td>no</td>
<td>weakly</td>
</tr>
<tr>
<td>count(S) ≥ v</td>
<td>no</td>
<td>yes</td>
<td>weakly</td>
</tr>
<tr>
<td>sum(S) ≤ v (a ∈ S, a ≥ 0)</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>sum(S) ≥ v (a ∈ S, a ≥ 0)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>range(S) ≤ v</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>range(S) ≥ v</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>avg(S) θ v, θ ∈ {=, ≤, ≥}</td>
<td>convertible</td>
<td>convertible</td>
<td>no</td>
</tr>
<tr>
<td>support(S) ≥ ξ</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>support(S) ≤ ξ</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### Classification of Constraints

![Classification of Constraints Diagram](image)
Questions?

- Lecture Slides on the Course Website,
  "www.ecs.baylor.edu/faculty/cho/4352"