Frequent pattern mining: association rules

CS434
What Is Frequent Pattern Mining?

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

• 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?

• Broad applications
  – Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  – Web log (click stream) analysis
  – DNA sequence analysis
Association rules

Data: Market-Basket transactions

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

Example of Association Rules

\{\text{Diaper}\} \rightarrow \{\text{Beer}\},
\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\},
\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\},

Implication means \text{co-occurrence, not causality!}

Given a set of transactions, find rules that will \text{predict the occurrence of an item based on the occurrences of other items} in the transaction
Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: \{Milk, Bread, Diaper\}
  - k-itemset
    - An itemset that contains k items

- **Support count (\(\sigma\))**
  - Frequency of occurrence of an itemset
  - E.g. \(\sigma(\{\text{Milk, Bread, Diaper}\}) = 2\)

- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. \(s(\{\text{Milk, Bread, Diaper}\}) = 2/5\)

- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a \textit{mins}up threshold

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>
Definition: Association Rule

- **Association Rule**
  - An implication expression of the form \( X \rightarrow Y \), where \( X \) and \( Y \) are itemsets
  - Example:
    \( \{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\} \)

- **Rule Evaluation Metrics**
  - **Support (s)**
    - Fraction of transactions that contain both \( X \) and \( Y \): \( P(X \land Y) \)
  - **Confidence (c)**
    - Measures how often items in \( Y \) appear in transactions that contain \( X \): \( P(Y|X) \)

**Example:**
\( \{\text{Milk, Diaper}\} \Rightarrow \text{Beer} \)

\[
s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4
\]

\[
c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67
\]
Problem definition: Association Rules Mining

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, C</td>
</tr>
<tr>
<td>20</td>
<td>A, C</td>
</tr>
<tr>
<td>30</td>
<td>A, D</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
</tbody>
</table>

- **Inputs:**
  - Itemset $X=\{x_1, \ldots, x_k\}$,
  - thresholds: $\text{min\_sup}$, $\text{min\_conf}$

- **Output:**
  - All the rules $X \rightarrow Y$ having:
    - support $(P(X^Y)) \geq \text{min\_sup}$
    - confidence $(P(Y|X)) \geq \text{min\_conf}$

*Let $\text{min\_sup} = 50\%$, $\text{min\_conf} = 50\%$:
  - $A \rightarrow C$ (50\%, 66.7\%)
  - $C \rightarrow A$ (50\%, 100\%)*
Brute-force solution

• List all possible association rules
• Compute the support and confidence for each rule
• Prune rules that fail the $min\_sup$ and $min\_conf$ thresholds

⇒ Computationally prohibitive!
Mining Association Rules

Example of Rules:

{Milk, Diaper} → {Beer} (s=0.4, c=0.67)
{Milk, Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper, Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk, Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk, Beer} (s=0.4, c=0.5)
{Milk} → {Diaper, Beer} (s=0.4, c=0.5)

Observations:

• All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}

• Rules originating from the same itemset have identical support but can have different confidence

• Thus, we may decouple the support and confidence requirements

• We can first find all frequent itemsets that satisfy the support requirement
Mining Association Rules

• Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq \text{minsup}$
  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

• Frequent itemset generation is still computationally expensive
Given $d$ items, there are $2^d$ possible candidate itemsets.
Frequent Itemset Generation

• Brute-force approach:
  – Each itemset in the lattice is a candidate frequent itemset
  – Count the support of each candidate by scanning the database

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

– Match each transaction against every candidate
– Complexity ~ O(NMw) => Expensive since M = 2^d !!!
Reducing Number of Candidates

• **Apriori principle:**
  – If an itemset is frequent, then all of its subsets must also be frequent
  – If \{**beer**, **diaper**, **nuts**\} is frequent, so is \{**beer**, **diaper**\}
  – i.e., every transaction having \{beer, diaper, nuts\} also contains \{beer, diaper\}

• Apriori principle holds due to the following property of the support measure:

\[
\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)
\]

  – Support of an itemset never exceeds the support of its subsets
  – This is known as the **anti-monotone** property of support
Illustrating Apriori Principle

Found to be Infrequent

Pruned supersets
Illustrating Apriori Principle

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>4</td>
</tr>
<tr>
<td>Coke</td>
<td>2</td>
</tr>
<tr>
<td>Milk</td>
<td>4</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Diaper</td>
<td>4</td>
</tr>
<tr>
<td>Eggs</td>
<td>1</td>
</tr>
</tbody>
</table>

Min Support count = 3

If every subset is considered,
\[ C_1^6 + C_2^6 + C_3^6 = 41 \]
With support-based pruning,
\[ 6 + 6 + 1 = 13 \]

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread, Milk}</td>
<td>3</td>
</tr>
<tr>
<td>{Bread, Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Bread, Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk, Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Milk, Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Beer, Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Pairs (2-itemsets)
(No need to generate candidates involving Coke or Eggs)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread, Milk, Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Triplets (3-itemsets)
The Apriori Algorithm

• Identify all frequent itemsets (with given \textit{minsup})

• Method:
  – Let $k=1$
  – Generate frequent itemsets of length 1
  – Repeat until no new frequent itemsets are identified
    • Generate length $(k+1)$ candidate itemsets from length $k$
      frequent itemsets
    • Prune candidate itemsets containing subsets of length $k$
      that are infrequent
    • Count the support of each candidate by scanning the DB
    • Eliminate candidates that are infrequent, leaving only those
      that are frequent
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**
  
  \[ C_{k+1} = \text{candidates generated from } L_k; \]
  
  **for each** transaction \( t \) in database **do**
  
  increment the count of all candidates in \( C_{k+1} \)
  
  that are contained in \( t \)
  
  \[ L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support} \]
  
  **end**

  return \( \bigcup_k L_k \);
How to Generate Candidates?

• Suppose the items in $L_k$ are listed in an order (e.g., alphabetic ordering)

• Step 1: self-joining $L_k$
  For all itemsets $p$ and $q$ in $L_k$ such that
  $$p.item_i = q.item_i \text{ for } i = 1, 2, \ldots, k-1 \text{ and } p.item_k < q.item_k$$
  Add to $C_{k+1}$
  $$p.item_1, p.item_2, \ldots, p.item_k, q.item_k$$

• Step 2: pruning
  For all itemsets $c$ in $C_{k+1}$ do
    For all (k)-subsets $s$ of $c$ do
      if ($s$ is not in $L_k$) then delete $c$ from $C_{k+1}$
Important Details of Apriori

**Self-joining rule:**
1. We join two itemsets if and only if they only differ by their last item.
2. When joining, the items are always ranked based on a fixed ordering of the items (e.g., alphabetic ordering).

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

Why not abd, and acd -> abcd?
Why should this work?

• How can we be sure we are not missing any possible itemset?

• This can be seen by proving that for every possible frequent k+1-itemset, it will be included using this self-joining process.

Proof
For any k +1 item set S (with items ranked), it will be included by joining the following two subsets:

1. $S_k = \{\text{the first } k \text{ items of } S\}$
2. $S'_k = S \text{ with the } k\text{-th item removed}$

Clearly $S_k$ and $S'_k$ are frequent, and differ by only the last item. So they must satisfy the self-join condition and $S_k \cap S'_k = S$. 

The Apriori Algorithm—An Example

Sup$_{\text{min}} = 2/4$

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>

$C_1$: 1st 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>{D}</td>
<td></td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

2nd scan

$C_2$

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

$C_3$

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

3rd scan

$C_3$

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

• Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
     – Enumerate all possible rules from the frequent itemset and out these of high confidence
Example: Generating rules

• \textbf{Min\_conf = 80%}

\begin{itemize}
  \item \textbf{Database TDB}
  \begin{tabular}{|c|c|}
    \hline
    Tid & Items \\
    \hline
    10 & A, C, D \\
    20 & B, C, E \\
    30 & A, B, C, E \\
    40 & B, E \\
    \hline
  \end{tabular}
\end{itemize}

\begin{itemize}
  \item \textbf{L2}
  \begin{tabular}{|c|c|}
    \hline
    Itemset & sup \\
    \hline
    \{A, C\} & 2 \\
    \{B, C\} & 2 \\
    \{B, E\} & 3 \\
    \{C, E\} & 2 \\
    \hline
  \end{tabular}
\end{itemize}

\begin{itemize}
  \item \textbf{L3}
  \begin{tabular}{|c|c|}
    \hline
    Itemset & sup \\
    \hline
    \{B, C, E\} & 2 \\
    \hline
  \end{tabular}
\end{itemize}

\begin{itemize}
  \item \textbf{Itemset sup}
  \begin{tabular}{|c|c|}
    \hline
    \{A\} & 2 \\
    \{B\} & 3 \\
    \{C\} & 3 \\
    \{E\} & 3 \\
    \hline
  \end{tabular}
\end{itemize}

- A $\rightarrow$ C: 100%
- C $\rightarrow$ A: 66.7%
- B $\rightarrow$ C: 66.7%
- C $\rightarrow$ B: 66.7%
- B $\rightarrow$ E: 100%
- E $\rightarrow$ B: 100%
- C $\rightarrow$ E: 66.7%
- E $\rightarrow$ C: 66.7%

- B, C $\rightarrow$ E: 100%
- B, E $\rightarrow$ C: 66.7%
- C, E $\rightarrow$ B: 100%
Frequent-Pattern Mining: Summary

• Frequent pattern mining—an important task in data mining
• “Scalable” frequent pattern mining methods
  – Apriori (Candidate generation & test)
    ▪ The Apriori property has also been used in mining other type of patterns such as sequential and structured patterns
    ▪ Problem: frequent patterns are not necessarily interesting patterns
      ▪ Bread -> milk is not really interesting although it has high support and confidence
      ▪ Many other measures of interestingness exist to address this problem
        ▪ Such as “unexpectedness”
Comparing Association rule with Supervised learning

• Supervised learning
  – Have predefined class variable
  – Focus on difference one class from another

• Association rule mining
  – Do not have predefined target class variable
  – Right hand side of the rule can have many items
  – We could place the class variable C on the right hand side of a rule, but it does not focus on differentiating classes, but more on characterizing a class
What you need to know

• What is an association rule?
• What are the support and confidence of a rule?
• The apriori property
• How to find frequent itemset using the apriori property
  – The Candidate Generation: self-join, and prune
  – Why is it correct?
• How to produce association rules based on frequent itemsets?