Association Pattern Mining

Lijun Zhang  
zlj@nju.edu.cn  
http://cs.nju.edu.cn/zlj
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
Introduction

- **Transactions**
  - Sets of items bought by customers

- **The Goal**
  - Determine associations between groups of items bought by customers

- **Quantification of the Level of Association**
  - Frequencies of sets of items

- **The Discovered Sets of Items**
  - Large itemsets, frequent itemsets, or frequent patterns
Applications

☐ Supermarket Data
  ■ Target marketing, shelf placement

☐ Text Mining
  ■ Identifying co-occurring terms

☐ Generalization to Dependency-oriented Data Types
  ■ Web log analysis, software bug detection

☐ Other Major Data Mining Problems
  ■ Clustering, classification, and outlier analysis
Association Rules

- Generated from Frequent Itemsets
- Formulation $X \Rightarrow Y$
  - \{Beer\} $\Rightarrow$ \{Diapers\}
  - \{Eggs,Milk\} $\Rightarrow$ \{Yogurt\}
- Applications
  - Promotion
  - Shelf placement
- Conditional Probability

$$P(Y|X) = \frac{P(X \cap Y)}{P(X)}$$
Outline

- Introduction
- **The Frequent Pattern Mining Model**
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
The Frequent Pattern Mining Model

- $U$ is a set of $d$ items
- $T$ is a set of $n$ transactions $T_1, \ldots, T_n$
  - $T_i \subseteq U$
- Binary Representation of $T_1, \ldots, T_n$
  - $U = \{\text{Bread, Butter, Cheese, Eggs, Milk, Yogurt}\}$

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
</tbody>
</table>
The Frequent Pattern Mining Model

- $U$ is a set of $d$ items
- $T$ is a set of $n$ transactions $T_1, ..., T_n$
  - $T_i \subseteq U$
- Binary Representation of $T_1, ..., T_n$
  - $U = \{\text{Bread, Butter, Cheese, Eggs, Milk, Yogurt}\}$

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
</tbody>
</table>

- Itemset, $k$-itemset
  - A set of items, A set of $k$ items
Definitions

☐ Support

Definition 4.2.1 (Support) The support of an itemset $I$ is defined as the fraction of the transactions in the database $\mathcal{T} = \{T_1 \ldots T_n\}$ that contain $I$ as a subset.

- Denoted by $\text{sup}(I)$

☐ Frequent Itemset Mining

Definition 4.2.2 (Frequent Itemset Mining) Given a set of transactions $\mathcal{T} = \{T_1 \ldots T_n\}$, where each transaction $T_i$ is a subset of items from $U$, determine all itemsets $I$ that occur as a subset of at least a predefined fraction $\text{minsup}$ of the transactions in $\mathcal{T}$.

- $\text{minsup}$ is the minimum support

Definition 4.2.3 (Frequent Itemset Mining: Set-wise Definition) Given a set of sets $\mathcal{T} = \{T_1 \ldots T_n\}$, where each element of the set $T_i$ is drawn on the universe of elements $U$, determine all sets $I$ that occur as a subset of at least a predefined fraction $\text{minsup}$ of the sets in $\mathcal{T}$.
An Example

A Market Basket Data Set

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
<tr>
<td>4</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>5</td>
<td>{Cheese, Milk, Yogurt}</td>
<td>001011</td>
</tr>
</tbody>
</table>

- **support** of \{Bread, Milk\} is \(\frac{2}{5} = 0.4\)
- **support** of \{Cheese, Yogurt\} is \(\frac{1}{5} = 0.2\)

- **minsup** = 0.3
  - \{Bread, Milk\} is a frequent itemset
Properties

- The smaller \( \text{minsup} \) is, the larger the number of frequent itemsets is.

- **Support Monotonicity Property**

  Property 4.2.1 (Support Monotonicity Property) \( \text{The support of every subset } J \text{ of } I \text{ is at least equal to that of the support of itemset } I. \)

  \[
  \text{sup}(J) \geq \text{sup}(I) \quad \forall J \subseteq I
  \]  

  - When an itemset \( I \) is contained in a transaction, all its subsets will also be contained in the transaction.

- **Downward Closure Property**

  Property 4.2.2 (Downward Closure Property) \( \text{Every subset of a frequent itemset is also frequent.} \)
Maximal Frequent Itemsets

Definition 4.2.4 (Maximal Frequent Itemsets) A frequent itemset is maximal at a given minimum support level \( \text{mins}up \), if it is frequent, and no superset of it is frequent.

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
<tr>
<td>4</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>5</td>
<td>{Cheese, Milk, Yogurt}</td>
<td>001011</td>
</tr>
</tbody>
</table>

- Maximal frequent patterns at \( \text{mins}up = 0.3 \)
  - \{Bread,Milk\}, \{Cheese,Milk\}, \{Eggs,Milk, Yogurt\}

- Frequent Patterns at \( \text{mins}up = 0.3 \)
  - The total number is 11
  - Subsets of the maximal frequent patterns
Maximal Frequent Itemsets

Definition 4.2.4 (Maximal Frequent Itemsets) A frequent itemset is maximal at a given minimum support level minsup, if it is frequent, and no superset of it is frequent.

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
<tr>
<td>4</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>5</td>
<td>{Cheese, Milk, Yogurt}</td>
<td>001011</td>
</tr>
</tbody>
</table>

- The maximal patterns can be considered **condensed** representations of the frequent patterns.
- However, this condensed representation does not retain information about the support values of the subsets.
The Itemset Lattice

- Contain $2^{|U|}$ nodes and represents search space
The Itemset Lattice

- Contain $2^{|U|}$ nodes and represent search space.

All itemsets above this border are frequent.
The Itemset Lattice

- Contain $2^{|U|}$ nodes and represents search space

All itemsets below this border are infrequent.
The Itemset Lattice

- Contain $2^{|U|}$ nodes and represents search space.
The Itemset Lattice

- Contain \(2^{\vert U \vert}\) nodes and represents search space.

Any valid border always respects the downward closure property.
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
Definitions

- The confidence of a rule \( X \Rightarrow Y \)

Definition 4.3.1 (Confidence) Let \( X \) and \( Y \) be two sets of items. The confidence \( \text{conf}(X \Rightarrow Y) \) of the rule \( X \Rightarrow Y \) is the conditional probability of \( X \cup Y \) occurring in a transaction, given that the transaction contains \( X \). Therefore, the confidence \( \text{conf}(X \Rightarrow Y) \) is defined as follows:

\[
\text{conf}(X \Rightarrow Y) = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)}. \tag{4.2}
\]

- \( X \) and \( Y \) are said to be the antecedent and the consequent
- In the previous table:

\[
\text{conf} \left( \{\text{Eggs, Milk}\} \Rightarrow \{\text{Yogurt}\} \right) = \frac{\text{sup}(\{\text{Eggs, Milk, Yogurt}\})}{\text{sup}(\{\text{Eggs, Milk}\})} = \frac{0.4}{0.6} = \frac{2}{3}
\]
Definitions

- The confidence of a rule $X \Rightarrow Y$

Definition 4.3.1 (Confidence) Let $X$ and $Y$ be two sets of items. The confidence $conf(X \Rightarrow Y)$ of the rule $X \Rightarrow Y$ is the conditional probability of $X \cup Y$ occurring in a transaction, given that the transaction contains $X$. Therefore, the confidence $conf(X \Rightarrow Y)$ is defined as follows:

$$conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}.$$  \hspace{1cm} (4.2)

- Association Rules

Definition 4.3.2 (Association Rules) Let $X$ and $Y$ be two sets of items. Then, the rule $X \Rightarrow Y$ is said to be an association rule at a minimum support of $\text{minsup}$ and minimum confidence of $\text{minconf}$, if it satisfies both the following criteria:

1. The support of the itemset $X \cup Y$ is at least $\text{minsup}$.

2. The confidence of the rule $X \Rightarrow Y$ is at least $\text{minconf}$.

- A sufficient number of transactions are relevant
- A sufficient strength in terms of conditional probabilities
The Overall Framework

1. In the first phase, all the frequent itemsets are generated at the minimum support of $\textit{minsup}$.
   - The most difficult step

2. In the second phase, the association rules are generated from the frequent itemsets at the minimum confidence level of $\textit{minconf}$.
   - Relatively straightforward
Implementation of 2\textsuperscript{nd} Phase

\begin{itemize}
  \item A Straightforward Implementation
    \begin{itemize}
      \item Given a frequent itemset \( I \)
      \item Generate all possible partitions \( X \) and \( Y = I - X \)
      \item Examine the confidence of each \( X \Rightarrow Y \)
    \end{itemize}
  \item Reduce the Search Space
\end{itemize}

Property 4.3.1 (Confidence Monotonicity) Let \( X_1, X_2, \) and \( I \) be itemsets such that \( X_1 \subseteq X_2 \subseteq I \). Then the confidence of \( X_2 \Rightarrow I - X_2 \) is at least that of \( X_1 \Rightarrow I - X_1 \).

\[
\text{conf}(X_2 \Rightarrow I - X_2) \geq \text{conf}(X_1 \Rightarrow I - X_1)
\] (4.3)

\[
\text{sup}(X_2) \leq \text{sup}(X_1) \Rightarrow \frac{\text{sup}(I)}{\text{sup}(X_2)} \geq \frac{\text{sup}(I)}{\text{sup}(X_1)}
\]
Implementation of 2\textsuperscript{nd} Phase

- A Straightforward Implementation
  - Given a frequent itemset $I$
  - Generate all possible partitions $X$ and $Y = I - X$
  - Examine the confidence of each $X \Rightarrow Y$

- Reduce the Search Space

Property 4.3.1 (Confidence Monotonicity) Let $X_1$, $X_2$, and $I$ be itemsets such that $X_1 \subseteq X_2 \subseteq I$. Then the confidence of $X_2 \Rightarrow I - X_2$ is at least that of $X_1 \Rightarrow I - X_1$.

$$conf(X_2 \Rightarrow I - X_2) \geq conf(X_1 \Rightarrow I - X_1) \quad (4.3)$$

- Techniques for frequent itemsets mining can also be applied here
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
Frequent Itemset Mining Algorithms

- Brute Force Algorithms
- The Apriori Algorithm
- Enumeration-Tree Algorithms
- Recursive Suffix-Based Pattern Growth Methods
Brute Force Algorithms (1)

□ The Naïve Approach

■ Generate all these candidate itemsets
  ✓ For a universe of items $U$, there are a total of $2^{|U|} - 1$ distinct subsets
  ✓ When $U = 1000$, $2^{1000} \geq 10^{300}$

■ Count their support against the transaction database

□ Observation

■ no $(k + 1)$-patterns are frequent if no $k$-patterns are frequent.
Brute Force Algorithms (2)

- A Improved Approach
  - Generate all candidate $k$-itemsets with $k$
  - Count their support against the transaction database
  - If no frequent itemsets are found, then stop; Otherwise, $k++$ and continue;

- A Significant Improvement
  - Let $l$ be the final value of $k$
    \[ \sum_{i=1}^{l} \binom{|U|}{i} \ll 2^{|U|} \]
  - $|U| = 1000$ and $l = 10$, it is $O(10^{23})$
A very minor application of the downward closure property made the algorithm much faster.

To Further Improve the Efficiency

1. Reducing the size of the explored search space (lattice of Fig. 4.1) by pruning candidate itemsets (lattice nodes) using tricks, such as the downward closure property.

2. Counting the support of each candidate more efficiently by pruning transactions that are known to be irrelevant for counting a candidate itemset.

3. Using compact data structures to represent either candidates or transaction databases that support efficient counting.
Frequent Itemset Mining Algorithms

- Brute Force Algorithms
- The Apriori Algorithm
- Enumeration-Tree Algorithms
- Recursive Suffix-Based Pattern Growth Methods
The Apriori Algorithm

The Basic Idea

- Use the downward closure property to prune the candidate search space
The Apriori Algorithm

- **The Basic Idea**
  - Use the downward closure property to prune the candidate search space

- **The Overall Procedure (level-wise)**
  - Using the frequent $k$-itemsets to generate $(k + 1)$-candidates
  - Prune the candidates before counting
  - Counts the supports of the remaining $(k + 1)$-candidates
  - Stop if there is no frequent $(k + 1)$-itemsets
The pseudocode

Algorithm Apriori(Transactions: $\mathcal{T}$, Minimum Support: $\text{mins}up$)
begin
    $k = 1$;
    $\mathcal{F}_1 = \{\text{All Frequent 1-itemsets}\}$;
    while $\mathcal{F}_k$ is not empty do begin
        Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
        Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
        Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining itemsets from $\mathcal{C}_{k+1}$ with support at least $\text{mins}up$;
        $k = k + 1$;
    end;
return ($\bigcup_{i=1}^{k} \mathcal{F}_i$);
end
Candidates Generation (1)

- A Naïve Approach
  - Check all the possible combination of frequent \( k \)-itemsets
  - Keep all the \( (k + 1) \)-itemsets

- An Example of the Naïve Approach
  - \( k \)-itemsets: \{abc\} \{bcd\} \{abd\} \{cde\}
  - \{abc\} + \{bcd\} = \{abcd\}
  - \{bcd\} + \{abd\} = \{abcd\}
  - \{abd\} + \{cde\} = \{abcde\}
  - ......
Candidates Generation (1)

- A Naïve Approach
  - Check all the possible combination of frequent $k$-itemsets
  - Keep all the $(k+1)$-itemsets

- An Example of the Naïve Approach
  - $k$-itemsets: $\{abc\} \ {bcd\}$
  - $\{abc\} + \{bcd\} = \{abcd\}$
  - $\{bcd\} + \{abd\} = \{abcd\}$
  - $\{abd\} + \{cde\} = \{abcde\}$
  - ......
Candidates Generation (2)

- **Introduction of Ordering**
  - Items in $U$ have a lexicographic ordering
  - Itemsets can be ordered as strings

- **A Better Approach**
  - Order the frequent $k$-itemsets
  - Merge two itemsets if the first $k - 1$ items of them are the same
Candidates Generation (3)

- Examples of the New Methods
  - \(k\)-itemsets: \{abc\} \{abd\} \{bcd\}
  - \{abc\} + \{abd\} = \{abcd\}
  - \(k\)-itemsets: \{abc\} \{acd\} \{bcd\}
  - No \((k + 1)\)-candidates
  - Early stop is possible
    - Do not need to check \{abc\} + \{bcd\} after checking \{abc\} + \{acd\}
  - Do we miss \{abcd\}?
    - No, due to the Downward Closure Property
Level-wise Pruning Trick

- Let $F_k$ be the set of frequent $k$-itemsets
- Let $C_{k+1}$ be the set of $(k + 1)$-candidates
- For an $I \in C_{k+1}$, it is frequent only if all the $k$-subsets of $I$ are frequent
- Pruning
  - Generate all the $k$-subsets of $I$
  - If any one of them does not belong to $F_k$, then remove $I$
Support Counting (1)

☐ A Naïve Approach

■ For each candidate \( I_i \in C_{k+1} \)
  ✓ For each transaction \( T_j \) in the transaction database \( T \)
  ■ Check whether \( I_i \) appears in \( T_j \)

☐ The Limitation

■ Inefficient if both \( |C_{k+1}| \) and \( |T| \) are very large
Support Counting (2)

A Better Approach

- Organize the candidate patterns in $C_{k+1}$ with a hash tree
  - Hash tree construction

- Use the hash tree to accelerate counting
  - Each transaction $T_i$ is examined with a small number of candidates in $C_{k+1}$
Hash Tree

- A tree with a fixed degree of the internal nodes
- Each internal node is associated with a random hash function that maps an item to one of its children
- A leaf node contains a list of lexicographically sorted itemsets
- Every itemset in $C_{k+1}$ is contained in exactly one leaf node of the hash tree.
Hash Tree of $C_3$

The Maximum Depth is $3 + 1$

Hash on the 1st item

Hash on the 2nd item

Hash on the 3rd item
Counting based on Hash Tree

- For each $T_j$, identify leaves in the hash tree that might contain subset items

- The Procedure
  - Root node – hash on all items in $T_j$
    - Suppose the $i$-th item of $T_j$ is hashed to one node, then pass this position $i$ to that node
  - If we are at a leaf – find all itemsets contained in $T_j$
  - If we are at an interior node – hash on each item after the given position
    - Suppose the $i$-th item of $T_j$ is hashed to one node, then pass this position $i$ to that node
Frequent Itemset Mining Algorithms

- Brute Force Algorithms
- The Apriori Algorithm
- Enumeration-Tree Algorithms
- Recursive Suffix-Based Pattern Growth Methods
**Enumeration-Tree**

- **Lexicographic Tree**
  - A node exists in the tree corresponding to each frequent itemset.
  - The root of the tree corresponds to the null itemset.
  - Let \( I = \{i_1, \ldots, i_k\} \) be a frequent itemset, where \( i_1, \ldots, i_k \) are listed in lexicographic order. The parent of the node \( I \) is the itemset \( \{i_1, \ldots, i_{k-1}\} \).
An Example

Frequent Tree Extension

- An item that is used to extend a node
Enumeration Tree Algorithms

**Algorithm** GenericEnumerationTree(Transactions: $\mathcal{T}$, Minimum Support: $\text{minsup}$)

begin
  Initialize enumeration tree $\mathcal{E}\mathcal{T}$ to single *Null* node;
  while any node in $\mathcal{E}\mathcal{T}$ has not been examined do begin
    Select one of more unexamined nodes $\mathcal{P}$ from $\mathcal{E}\mathcal{T}$ for examination;
    Generate candidates extensions $C(P)$ of each node $P \in \mathcal{P}$;
    Determine frequent extensions $F(P) \subseteq C(P)$ for each $P \in \mathcal{P}$ with support counting;
    Extend each node $P \in \mathcal{P}$ in $\mathcal{E}\mathcal{T}$ with its frequent extensions in $F(P)$;
  end
  return enumeration tree $\mathcal{E}\mathcal{T}$;
end

- Let $Q$ be the parent of $P$
- Let $F(Q)$ be the frequent extensions of $Q$
- Then, $C(P) \subseteq F(Q)$
Enumeration-Tree-Based Interpretation of Apriori

- Apriori constructs the enumeration tree in breadth-first manner
- Apriori generates candidate \((k + 1)\)-itemsets by merging two frequent \(k\)-itemsets of which the first \(k-1\) items of are the same

- Extend \(\{ab\}\) with \(\{cdf\} \subseteq F(\{a\})\)

```
          a
         / \  \
        /   \ /
       /     /
      b     c
     / \   / \ \
    /   \ /   /
   /     \     \
  a    b    c
 / \   / \   / \ \
/   \ /   \ /   \
\       \       \\
ab ac ad af
```
TreeProjection (1)

- **The Goal**
  - Reuse the counting work that has already been done before

- **Projected Databases**
  - Each projected transaction database is specific to an enumeration-tree node.
  - Transactions that do not contain the itemset $P$ are removed.
  - Projected database at node $P$ can be expressed only in terms of the items in $C(P)$
The Algorithm

Algorithm ProjectedEnumerationTree(Transactions: $\mathcal{T}$,
Minimum Support: $\minsup$)

begin
    Initialize enumeration tree $\mathcal{ET}$ to a single ($Null$, $\mathcal{T}$) root node;
    while any node in $\mathcal{ET}$ has not been examined do begin
        Select an unexamined node $(P, \mathcal{T}(P))$ from $\mathcal{ET}$ for examination;
        Generate candidates item extensions $C(P)$ of node $(P, \mathcal{T}(P))$;
        Determine frequent item extensions $F(P) \subseteq C(P)$ by support counting
        of individual items in smaller projected database $\mathcal{T}(P)$;
        Remove infrequent items in $\mathcal{T}(P)$;
        for each frequent item extension $i \in F(P)$ do begin
            Generate $\mathcal{T}(P \cup \{i\})$ from $\mathcal{T}(P)$;
            Add $(P \cup \{i\}, \mathcal{T}(P \cup \{i\}))$ as child of $P$ in $\mathcal{ET}$;
        end
    end
    return enumeration tree $\mathcal{ET}$;
end
Vertical Counting Methods (1)

- **Vertical Representation of Market Basket Data Set**

<table>
<thead>
<tr>
<th>Item</th>
<th>Set of tids</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>{1, 3}</td>
<td>10100</td>
</tr>
<tr>
<td>Butter</td>
<td>{1}</td>
<td>10000</td>
</tr>
<tr>
<td>Cheese</td>
<td>{3, 5}</td>
<td>00101</td>
</tr>
<tr>
<td>Eggs</td>
<td>{2, 3, 4}</td>
<td>01110</td>
</tr>
<tr>
<td>Milk</td>
<td>{1, 2, 3, 4, 5}</td>
<td>11111</td>
</tr>
<tr>
<td>Yogurt</td>
<td>{2, 4, 5}</td>
<td>01011</td>
</tr>
</tbody>
</table>

- Intersection of two item \textit{tid} list gives a new list
  - The length is the support of the 2-itemset
Vertical Counting Methods (2)

☐ The Algorithm

Algorithm VerticalApriori(Transactions: \( T \), Minimum Support: \( \text{minsup} \))
begin
\( k = 1; \)
\( \mathcal{F}_1 = \{ \text{All Frequent 1-itemsets} \}; \)
Construct vertical \( tid \) lists of each frequent item;
while \( \mathcal{F}_k \) is not empty do begin
    Generate \( C_{k+1} \) by joining itemset-pairs in \( \mathcal{F}_k \);
    Prune itemsets from \( C_{k+1} \) that violate downward closure;
    Generate \( tid \) list of each candidate itemset in \( C_{k+1} \) by intersecting
    \( tid \) lists of the itemset-pair in \( \mathcal{F}_k \) that was used to create it;
    Determine supports of itemsets in \( C_{k+1} \) using lengths of their \( tid \) lists;
    \( \mathcal{F}_{k+1} = \) Frequent itemsets of \( C_{k+1} \) together with their \( tid \) lists;
    \( k = k + 1; \)
end;
return(\( \bigcup_{i=1}^{k} \mathcal{F}_i \));
end
Frequent Itemset Mining Algorithms

- Brute Force Algorithms
- The Apriori Algorithm
- Enumeration-Tree Algorithms
- Recursive Suffix-Based Pattern Growth Methods
Generic Recursive Suffix Growth Algorithm

☐ $T$ is expressed in terms of only frequent 1-itemset

Algorithm $RecursiveSuffixGrowth$(Transactions in terms of frequent 1-items: $T$, Minimum Support: $minsup$, Current Suffix: $P$)

begin
  for each item $i$ in $T$ do begin
    report itemset $P_i = \{i\} \cup P$ as frequent;
    Extract all transactions $T_i$ from $T$ containing item $i$;
    Remove all items from $T_i$ that are lexicographically $\geq i$;
    Remove all infrequent items from $T_i$;
    if ($T_i \neq \emptyset$) then $RecursiveSuffixGrowth$($T_i$, $minsup$, $P_i$);
  end
end
Relationship Between FP-Growth and Enumeration-Tree Methods

- They are Equivalent

(a) Prefix extensions with ordering of $a, b, c, d, e, f$
(Enumeration Tree Prefixes shown)

(b) FP-growth with ordering of $f, e, d, c, b, a$
(Recursion Tree Suffixes shown)
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- **Alternative Models: Interesting Patterns**
- Useful Meta-algorithms
- Summary
Motivations (1)

- Advantages of Frequent Itemsets
  - Very simple and intuitive
    - Raw frequency for the support
    - Conditional probabilities for the confidence
  - Downward Closure Property
    - Enable efficient algorithms
Motivations (2)

- Disadvantages of Frequent Itemsets
  - Patterns are not always significant from an application-specific perspective

<table>
<thead>
<tr>
<th>tid</th>
<th>Set of items</th>
<th>Binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Bread, Butter, Milk}</td>
<td>110010</td>
</tr>
<tr>
<td>2</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>3</td>
<td>{Bread, Cheese, Eggs, Milk}</td>
<td>101110</td>
</tr>
<tr>
<td>4</td>
<td>{Eggs, Milk, Yogurt}</td>
<td>000111</td>
</tr>
<tr>
<td>5</td>
<td>{Cheese, Milk, Yogurt}</td>
<td>001011</td>
</tr>
</tbody>
</table>

- \textit{Milk} can be appended to any set of items, without changing its frequency
- For any set of items \(X\), the association rule \(X \Rightarrow \{\text{Milk}\}\) has 100% confidence
Motivations (2)

☐ Disadvantages of Frequent Itemsets
- Patterns are not always significant from an application-specific perspective
- Cannot adjust to the skew in the individual item support values
  - Support of \{Milk, Butter\} is very different from \{¬Milk, ¬Butter\}
  - But the statistical coefficient of correlation is exactly the same in both cases

☐ Bit Symmetric Property
- Values of 0 in the binary matrix are treated in a similar way to values of 1
Statistical Coefficient of Correlation

- **Pearson Coefficient**
  \[ \rho = \frac{E[X \cdot Y] - E[X] \cdot E[Y]}{\sigma(X) \cdot \sigma(Y)} \]

- **Estimated Correlation**
  \[ \rho_{ij} = \frac{\text{sup}\{i, j\} - \text{sup}(i) \cdot \text{sup}(j)}{\sqrt{\text{sup}(i) \cdot \text{sup}(j) \cdot (1 - \text{sup}(i)) \cdot (1 - \text{sup}(j))}}. \]

- **Properties**
  - Lies in the range \([-1,1]\)
  - Satisfies the bit symmetric property
  - Intuitively hard to interpret
\( \chi^2 \) Measure

- Given a set \( X \) of \( k \) items, there are \( 2^k \) possible states
  - \( k = 2 \) items \( \{\text{Bread}, \text{Butter}\} \), the \( 2^2 \) states are \( \{\text{Bread}, \text{Butter}\}, \{\text{Bread, } \neg\text{Butter}\}, \{\neg\text{Bread, Butter}\}, \text{ and } \{\neg\text{Bread, } \neg\text{Butter}\} \)
**$\chi^2$ Measure**

- Given a set $X$ of $k$ items, there are $2^k$ possible states.
- The $\chi^2$-measure for set of items $X$

$$\chi^2(X) = \sum_{i=1}^{2^{|X|}} \frac{(O_i - E_i)^2}{E_i}.$$ 

- $O_i$ and $E_i$ be the observed and expected values of the absolute support of state $i$. 
\( \chi^2 \) Measure

- Given a set \( X \) of \( k \) items, there are \( 2^k \)-possible states
- The \( \chi^2 \)-measure for set of items \( X \)

Properties
- Larger values of this quantity indicate greater dependence
- Do not reveal whether the dependence between items is positive or negative
- Is bit-symmetric
- Satisfies the upward closure property
- High computational complexity
Interest Ratio

- **Definition**

\[ I(\{i_1 \ldots i_k\}) = \frac{\sup(\{i_1 \ldots i_k\})}{\prod_{j=1}^{k} \sup(i_j)} \]

- **Properties**
  - When the items are statistically independent, the ratio is 1.
  - Value greater than 1 indicates that the variables are positively correlated.
  - When some items are extremely rare, the interest ratio can be misleading.
  - Donot satisfy the downward closure property.
Symmetric Confidence Measures

☐ Confidence Measure is Asymmetric

\[ \text{conf}(X \Rightarrow Y) \neq \text{conf}(Y \Rightarrow X) \]

☐ Let X and Y be two 1-itemsets

- Minimum of \( \text{conf}(X \Rightarrow Y) \) and \( \text{conf}(Y \Rightarrow X) \)
- Maximum of \( \text{conf}(X \Rightarrow Y) \) and \( \text{conf}(Y \Rightarrow X) \)
- Average of \( \text{conf}(X \Rightarrow Y) \) and \( \text{conf}(Y \Rightarrow X) \)
  ✓ Geometric mean is the cosine measure

☐ Can be generalized to k-itemsets

☐ Do not satisfy the downward closure property
Cosine Coefficient on Columns

Definition

\[ \text{cosine}(i, j) = \frac{\text{sup}(\{i, j\})}{\sqrt{\text{sup}(i)} \cdot \sqrt{\text{sup}(j)}}. \]

Interpretation

- Cosine similarity between two columns of the data matrix

A Symmetric Confidence Measure
Jaccard Coefficient and the Min-hash Trick

- Jaccard coefficient $J(S_1, S_2)$ between the two sets
  \[ J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \]

- Jaccard coefficient between multiway sets
  \[ J(S_1 \ldots S_k) = \frac{|\cap S_i|}{|\cup S_i|} \]

- Properties
  - Satisfy the downward closure property
  - Speed up by min-hash trick
Collective Strength (1)

- **Violation**
  - If some of the items of \( I \) are present in the transaction, and others are not.

- **Violation Rate \( v(I) \)**
  - The fraction of violations of the itemset \( I \) over all transactions.
Collective Strength (2)

Collective Strength

\[ C(I) = \frac{1 - v(I)}{1 - E[v(I)]} \cdot \frac{E[v(I)]}{v(I)}. \]

- The expected value of \( v(I) \) is calculated assuming statistical independence of the individual items.

\[ E[v(I)] = 1 - \prod_{i \in I} p_i - \prod_{i \in I} (1 - p_i). \]

- 0 indicates a perfect negative correlation
- \( \infty \) indicates a perfectly positive correlation
Collective Strength (3)

- Interpretation of Collective Strength

\[ C(I) = \frac{\text{Good Events}}{\text{E[Good Events]}} \cdot \frac{\text{E[Bad Events]}}{\text{Bad Events}}. \]

- Strongly Collective Itemsets

Definition 4.5.1: An itemset \( I \) is denoted to be strongly collective at level \( s \), if it satisfies the following properties:

1. The collective strength \( C(I) \) of the itemset \( I \) is at least \( s \).

2. Closure property: The collective strength \( C(J) \) of every subset \( J \) of \( I \) is at least \( s \).

- The closure property is enforced.
Relationship to Negative Pattern Mining

- **Motivation**
  - Determine patterns between items or their absence

- **Satisfy Bit Symmetric Property**
  - Statistical coefficient of correlation
  - $\chi^2$ measure
  - Jaccard Coefficient, Strongly Collective strength
    - Also satisfy downward closure property
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
Useful Meta-algorithms

- **Definition**
  - An algorithm that uses a particular algorithm as a subroutine
    - either to make the original algorithm more efficient (e.g., by sampling)
    - or to gain new insights

- **Sampling Methods**
- **Data Partitioned Ensembles**
- **Generalization to Other Data Types**
Sampling Methods

- **The Procedure**
  - Sample a subset of the transactions
  - Apply mining algorithm to sampled data

- **Challenges**
  - False positives: These are patterns that meet the support threshold on the sample but not on the base data.
    - Post-processing
  - False negatives: These are patterns that do not meet the support threshold on the sample, but meet the threshold on the data.
    - Reduce the support threshold
Data Partitioned Ensembles

- **The Procedure**
  - The transaction database is partitioned into $k$ disjoint segments
  - The mining algorithm is independently applied to each of these $k$ segments
  - Post-processing to remove false positives

- **Property**
  - No false negatives
Generalization to Other Data Types

- **Quantitative Data**
  - Rules contain quantitative attributes
    
    
    \[(Age = 90) \Rightarrow \text{Checkers.} \quad Age[85, 95] \Rightarrow \text{Checkers.}\]

  - Discretize and convert to binary form

- **Categorical Data**
  - Rules contain mixed attributes
    
    
    \[(Gender = Male), \quad Age[20, 30] \Rightarrow \text{Basketball.}\]

  - Transform to binary values
Outline

- Introduction
- The Frequent Pattern Mining Model
- Association Rule Generation Framework
- Frequent Itemset Mining Algorithms
- Alternative Models: Interesting Patterns
- Useful Meta-algorithms
- Summary
Summary

- **Frequent Pattern Mining**
  - Support, Downward Closure Property

- **Association Rule**
  - Support, Confidence

- **Frequent Itemset Mining Algorithms**
  - Brute Force Algorithms, Apriori, Enumeration-Tree Algorithms, Recursive Suffix-Based Pattern Growth Methods

- **Alternative Models: Interesting Patterns**
  - Pearson coefficient, $\chi^2$ Measure, Interest Ratio, Symmetric Confidence Measures, ...

- **Useful Meta-algorithms**
  - Sampling, Data Partitioned Ensembles, Generalization