Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

**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**

- (Diaper) → (Beer), (Milk, Bread) → (Eggs, Coke), (Beer, Bread) → (Milk).

Implication means co-occurrence, not causality!

**Definition: Frequent Itemset**

- **Itemset**
  - A collection of one or more items

  - **k-itemset**
    - An itemset that contains k items

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

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

- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a minsup threshold

**Definition: Association Rule**

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where $X$ and $Y$ are itemsets

  - Example: 
    - {Milk, Diaper} → {Beer}

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

  - Example: 
    - [Milk, Diaper] $\Rightarrow$ Beer
    - $s$ = $\sigma$(Milk, Diaper, Beer) / $\sigma$(Milk, Diaper) = 2/5 = 0.4
    - $c$ = $\sigma$(Milk, Diaper, Beer) / $\sigma$(Milk, Diaper) = 2/3 = 0.67

**Mining Association Rules**

<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 Rules:**

- [Milk, Diaper] $\Rightarrow$ [Beer] ($s$=0.4, $c$=0.67)
- [Milk, Beer] $\Rightarrow$ [Diaper] ($s$=0.4, $c$=1.0)
- [Diaper, Beer] $\Rightarrow$ [Milk] ($s$=0.4, $c$=0.67)
- [Diaper] $\Rightarrow$ [Milk, Beer] ($s$=0.4, $c$=0.5)
- [Milk] $\Rightarrow$ [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

**Association Rule Mining Task**

- Given a set of transactions $T$, the goal of association rule mining is to find all rules having:
  - Support $\geq$ minsup threshold
  - Confidence $\geq$ minconf threshold

- **Brute-force approach:**
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the minsup and minconf thresholds

  $\Rightarrow$ Computationally prohibitive!

**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

- Frequent itemset generation is still computationally expensive
Frequent Itemset Generation

![Frequent Itemset Generation Diagram]

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

- Match each transaction against every candidate
- Complexity ~ O(NMw) => Expensive since M = 2^d !!!

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: M=2^d
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
  - 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

- Minimum Support = 3
- Items (1-itemsets)
  - Bread: 4
  - Coke: 2
  - Milk: 4
  - Beer: 3
  - Diaper: 4
  - Eggs: 1
- Pairs (2-itemsets)
  - (Bread, Milk): 4
  - (Bread, Coke): 3
  - (Milk, Beer): 3
  - (Diaper, Eggs): 3
  - (Bread, Diaper): 3
  - (Milk, Coke): 3
  - (Beer, Eggs): 3

- Triplet (3-itemsets)
  - (Bread, Milk, Diaper): 3

If every subset is considered, \( |C_1| + |C_2| + |C_3| = 41 \)
With support-based pruning, \( 6 + 6 + 1 = 13 \)
Frequent Itemset Mining

- 2 strategies:
  - Breadth-first: Apriori
    - Exploit monotonicity to the maximum
  - Depth-first strategy: Eclat
    - Prune the database
    - Do not fully exploit monotonicity

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, C, D, E\} \\
\text{Candidates} &= \{\{A\}, \{C\}, \{D\}, \{E\}\} \\
&\Rightarrow \{A\}, \{C\}, \{D\}, \{E\}
\end{align*}
\]

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, C, B, D\} \\
\text{Candidates} &= \{\{A\}, \{C\}, \{B\}, \{D\}\} \\
&\Rightarrow \{A\}, \{C\}, \{B\}, \{D\}
\end{align*}
\]

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, B, C, D\} \\
\text{Candidates} &= \{\{A\}, \{B\}, \{C\}, \{D\}\} \\
&\Rightarrow \{A\}, \{B\}, \{C\}, \{D\}
\end{align*}
\]

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, B, C, D\} \\
\text{Candidates} &= \{\{A\}, \{B\}, \{C\}, \{D\}\} \\
&\Rightarrow \{A\}, \{B\}, \{C\}, \{D\}
\end{align*}
\]

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, B, C, D\} \\
\text{Candidates} &= \{\{A\}, \{B\}, \{C\}, \{D\}\} \\
&\Rightarrow \{A\}, \{B\}, \{C\}, \{D\}
\end{align*}
\]

Apriori

Candidates:

\[
\begin{align*}
\text{minsup} &= 2 \\
&= \{A, B, C, D\} \\
\text{Candidates} &= \{\{A\}, \{B\}, \{C\}, \{D\}\} \\
&\Rightarrow \{A\}, \{B\}, \{C\}, \{D\}
\end{align*}
\]
Apriori Algorithm

- **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
**Frequent Itemset Mining**

- **2 strategies:**
  - **Breadth-first: Apriori**
    - Exploit monotonicity to the maximum
  - **Depth-first strategy: Eclat**
    - Prune the database
    - Do not fully exploit monotonicity

**Depth-First Algorithms**

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Depth-First Algorithms

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]

Find all frequent itemsets

```
1 R, C
2 A, C, D
3 A, C, D
4 A, C, D
5 A, C, D
```

\[ \text{minsup}=2 \]
Depth-First Algorithm

1. B, C
2. B, C
3. A, C, D
4. A, B, C, D
5. B, D

A: 2
B: 4
C: 4
D: 3
Depth-First Algorithm

- **DB**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **DB[C]**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **A:** 2
- **B:** 4
- **C:** 4
- **D:** 3

Depth-First Algorithm

- **DB**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **DB[E]**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **A:** 2
- **B:** 4
- **C:** 4
- **D:** 3

Depth-First Algorithm

- **DB**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **DB[B]**
  - 1: A, B, C
  - 2: A, B, C
  - 3: A, C, D
  - 4: A, B, C, D
  - 5: B, D

- **A:** 2
- **B:** 4
- **C:** 4
- **D:** 3

ECLAT

- For each item, store a list of transaction ids (tids)

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>Horizontal Data Layout</th>
<th>Vertical Data Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A,B,E</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B,C,D</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>C,E</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>A,C,D</td>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>A,B,C,D</td>
<td>E</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>A,E</td>
<td>A</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>A,B</td>
<td>B</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>A,B,C</td>
<td>C</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>A,C,D</td>
<td>D</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>E</td>
<td>9</td>
</tr>
</tbody>
</table>

- Final set of frequent itemsets
ECLAT

- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

- Depth-first traversal of the search lattice
- Advantage: very fast support counting
- Disadvantage: intermediate tid-lists may become too large for memory

Rule Generation

- Given a frequent itemset L, find all non-empty subsets f ⊂ L such that f → L – f satisfies the minimum confidence requirement
  - If (A,B,C,D) is a frequent itemset, candidate rules:
    - ABC → D, ABD → C, ACD → B, BCD → A,
    - A → BCD, B → ACD, C → ABD, D → ABC
    - AB → CD, AC → BD, AD → BC, BC → AD,
    - BD → AC, CD → AB,
  - If |L| = k, then there are $2^k - 2$ candidate association rules (ignoring L → Ø and Ø → L)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property
    - c(ABC → D) can be larger or smaller than c(AB → D)
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:
    - c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)
  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm

- Lattice of rules
- Low Confidence Rule
- Pruned Rules
- ABCD → L