Mining Association Rules in Large Databases

What Is Association Rule Mining?

- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transactional databases, relational databases, and other information repositories

- Motivation (market basket analysis):
  - If customers are buying milk, how likely is that they also buy bread?
  - Such rules help retailers to:
    - plan the shelf space: by placing milk close to bread they may increase the sales
    - provide advertisements/recommendation to customers that are likely to buy some products
    - put items that are likely to be bought together on discount, in order to increase the sales
What Is Association Rule Mining?

- **Applications:**
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

- **Rule form:** “Body $\rightarrow$ Head [support, confidence]”.

- **Examples.**
  - buys(x, “diapers”) $\rightarrow$ buys(x, “beers”) [0.5%, 60%]
  - major(x, “CS”) $\wedge$ takes(x, “DB”) $\rightarrow$ grade(x, “A”) [1%, 75%]

---

Association Rules: Basic Concepts

- **Given:** (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)

- **Find:** all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
What are the components of rules?

- In data mining, a set of items is referred to as an **itemset**
- Let D be database of **transactions**
  - e.g.: 
    | Transaction ID | Items Bought |
    |----------------|--------------|
    | 2000           | A,B,C        |
    | 1000           | A,C          |
    | 4000           | A,D          |
    | 5000           | B,E,F        |
- Let I be the set of items that appear in the database, e.g., I={A,B,C,D,E,F}
- A **rule** is defined by \( X \Rightarrow Y \), where \( X \subseteq I \), \( Y \subseteq I \), and \( X \cap Y = \emptyset \)
  - e.g.: \( \{B,C\} \Rightarrow \{E\} \) is a rule

Are all the rules interesting?

- The number of potential rules is huge. We may not be interested in all of them.
- We are interesting in rules that:
  - their items appear frequently in the database
  - they hold with a high probability
- We use the following thresholds:
  - the **support** of a rule indicates how frequently its items appear in the database
  - the **confidence** of a rule indicates the probability that if the left hand side appears in a \( T \), also the right hand side will.
Rule Measures: Support and Confidence

Find all the rules \( X \Rightarrow Y \) with minimum confidence and support
- **Support**, \( s \), probability that a transaction contains \( X \cup Y \)
- **Confidence**, \( c \), conditional probability that a transaction having \( X \) also contains \( Y \)

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>A,B,C</td>
</tr>
<tr>
<td>1000</td>
<td>A,C</td>
</tr>
<tr>
<td>4000</td>
<td>A,D</td>
</tr>
<tr>
<td>5000</td>
<td>B,E,F</td>
</tr>
</tbody>
</table>

Let minimum support 50\%, and minimum confidence 50\%, we have
- \( A \Rightarrow C \) (50\%, 66.6\%)
- \( C \Rightarrow A \) (50\%, 100\%)

Example

<table>
<thead>
<tr>
<th>TID</th>
<th>date</th>
<th>items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10/10/99</td>
<td>{F,A,D,B}</td>
</tr>
<tr>
<td>200</td>
<td>15/10/99</td>
<td>{D,A,C,E,B}</td>
</tr>
<tr>
<td>300</td>
<td>19/10/99</td>
<td>{C,A,B,E}</td>
</tr>
<tr>
<td>400</td>
<td>20/10/99</td>
<td>{B,A,D}</td>
</tr>
</tbody>
</table>

Remember: \( \text{conf}(X \Rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X)} \)

- What is the **support** and **confidence** of the rule: \( \{B,D\} \Rightarrow \{A\} \)
- **Support**:
  - percentage of tuples that contain \( \{A,B,D\} = 75\% \)
- **Confidence**:
  \[
  \frac{\text{number of tuples that contain } \{A,B,D\}}{\text{number of tuples that contain } \{B,D\}} = 100\% 
  \]
Association Rule Mining

- **Boolean vs. quantitative associations** (Based on the types of values handled)
  - buys(x, “SQLServer”) ^ buys(x, “DMBook”) \(\rightarrow\) buys(x, “DBMiner”) [0.2%, 60%]
  - age(x, “30..39”) ^ income(x, “42..48K”) \(\rightarrow\) buys(x, “PC”) [1%, 75%]

- **Single dimension vs. multiple dimensional associations** (see ex. Above)
- **Single level vs. multiple-level analysis**
  - age(x, “30..39”) \(\rightarrow\) buys(x, “laptop”)
  - age(x, “30..39”) \(\rightarrow\) buys(x, “computer”)

- **Various extensions**
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - **Maximal frequent itemsets**: no frequent supersets
  - **frequent closed itemsets**: no superset with the same support

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
Mining Association Rules—An Example

For rule $A \Rightarrow C$:
- $\text{support} = \text{support}(\{A \cup C\}) = 50\%$
- $\text{confidence} = \frac{\text{support}(\{A \cup C\})}{\text{support}(\{A\})} = 66.6\%$

The Apriori principle:
- Any subset of a frequent itemset must also be frequent!

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
<th>Frequent Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>A,B,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>A,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>A,D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>B,E,F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Min. support 50%
Min. confidence 50%

Mining Frequent Itemsets: the Key Step

- Find the frequent itemsets: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if $\{AB\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$ should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to $m$ ($m$-itemset): Use frequent $k$-itemsets to explore $(k+1)$-itemsets.

- Use the frequent itemsets to generate association rules.
Visualization of the level-wise process:

Level 4 (frequent quadruples): \{\}

Level 3 (frequent triplets): \{ABD, BDF\}

Level 2 (frequent pairs): \{AB, AD, BF, BD, DF\}

Level 1 (frequent items): \{A, B, D, F\}

Remember:
All subsets of a frequent itemset must be frequent

Question: Can ADF be frequent?
NO: because AF is not frequent

The Apriori Algorithm (the general idea)

1. Find frequent items and put them to \(L_k\) (k=1)
2. Use \(L_k\) to generate a collection of candidate itemsets \(C_{k+1}\) with size \((k+1)\)
3. Scan the database to find which itemsets in \(C_{k+1}\) are frequent and put them into \(L_{k+1}\)
4. If \(L_{k+1}\) is not empty
   - k=k+1
   - GOTO 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
    - $C_{k+1} = \text{candidates generated from } L_k$;
    - for each transaction $t$ in database do
      - increment the count of all candidates in $C_{k+1}$
      - that are contained in $t$
    - $L_{k+1} = \text{candidates in } C_{k+1} \text{ with min_support (frequent)}$
  end
  - return $\cup_k L_k$;

- **Important steps in candidate generation:**
  - **Join Step:** $C_{k+1}$ is generated by joining $L_k$ with itself
  - **Prune Step:** Any $k$-itemset that is not frequent cannot be a subset of a frequent $(k+1)$-itemset

---

The Apriori Algorithm — Example

<table>
<thead>
<tr>
<th>Database D</th>
<th>C1</th>
<th>L1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TID</td>
<td>Items</td>
<td>itemset</td>
</tr>
<tr>
<td>100</td>
<td>1 3 4</td>
<td>{1}</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
<td>{2}</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
<td>{3}</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
<td>{4}</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
<td>{5}</td>
</tr>
</tbody>
</table>

Scan D

<table>
<thead>
<tr>
<th>C1</th>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Scan D

<table>
<thead>
<tr>
<th>C2</th>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 3}</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>{2 3}</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>{2 5}</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>{3 5}</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Scan D

<table>
<thead>
<tr>
<th>C3</th>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2 3 5}</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

min_supp=2=50%
How to Generate Candidates?

- Suppose the items in $L_k$ are listed in an order
- Step 1: self-joining $L_k$ \textit{(IN SQL)}
  
  \begin{verbatim}
  insert into $C_{k+1}$
  select $p$.item$_1$, $p$.item$_2$, ..., $p$.item$_k$, $q$.item$_k$
  from $L_k$ p, $L_k$ q
  where $p$.item$_1$=q.item$_1$, ..., $p$.item$_{k-1}$=q.item$_{k-1}$, $p$.item$_k$ < $q$.item$_k$
  \end{verbatim}
- Step 2: pruning
  
  \begin{verbatim}
  forall itemsets $c$ in $C_{k+1}$ do
    forall $k$-subsets $s$ of $c$ do
      if ($s$ is not in $L_k$) then delete $c$ from $C_{k+1}$
  \end{verbatim}

Example of Candidates 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 Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be 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 of the hash-tree for $C_3$

Hash function: mod 3

Hash on 1st item

Hash on 2nd item

Hash on 3rd item
Example of the hash-tree for C₃

Hash function: mod 3

12345

Hash on 1st item

2345
look for 2XX

Hash on 2nd item

345
look for 3XX

Hash on 3rd item

12345
look for 1XX

12345
look for 2XX

12345
look for 3XX

124 457
125 458
159

Example of the hash-tree for C₃

Hash function: mod 3

12345

Hash on 1st item

2345
look for 2XX

Hash on 2nd item

345
look for 3XX

Hash on 3rd item

12345
look for 1XX

12345
look for 2XX

12345
look for 3XX

124 457
125 458
159
AprioriTid: Use D only for first pass

- The database is not used after the 1st pass.
- Instead, the set $C_k'$ is used for each step, $C_k' = \langle \text{TID}, \{X_k\} \rangle$: each $X_k$ is a potentially frequent itemset in transaction with id=TID.
- At each step $C_k'$ is generated from $C_{k-1}'$ at the pruning step of constructing $C_k$ and used to compute $L_k$.
- For small values of $k$, $C_k'$ could be larger than the database!
Methods to Improve Apriori’s Efficiency

- **Hash-based itemset counting**: A $k$-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- **Transaction reduction**: A transaction that does not contain any frequent $k$-itemset is useless in subsequent scans
- **Partitioning**: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- **Sampling**: mining on a subset of given data, lower support threshold + a method to determine the completeness
- **Dynamic itemset counting**: add new candidate itemsets only when all of their subsets are estimated to be frequent

Frequent Itemset Generation
Illustrating Apriori Principle

Found to be Infrequent

Another Representation of Frequent Itemsets

<table>
<thead>
<tr>
<th>TID</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
<th>C10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Factors Affecting Complexity

- **Choice of minimum support threshold**
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets

- **Dimensionality (number of items) of the data set**
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase

- **Size of database**
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions

- **Average transaction width**
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Rule Generation

- Given a frequent itemset $L$, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If $\{A,B,C,D\}$ is a frequent itemset, candidate rules:
    
    \[
    \begin{aligned}
    &ABC \rightarrow D, \quad ABD \rightarrow C, \quad ACD \rightarrow B, \quad BCD \rightarrow A, \\
    &A \rightarrow BCD, \quad B \rightarrow ACD, \quad C \rightarrow ABD, \quad D \rightarrow ABC \\
    &AB \rightarrow CD, \quad AC \rightarrow BD, \quad AD \rightarrow BC, \quad BC \rightarrow AD, \\
    &BD \rightarrow AC, \quad CD \rightarrow AB,
    \end{aligned}
    \]

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)
Rule Generation

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

Rule Generation for Apriori Algorithm

- Lattice of rules
- Pruned rules
- Low confidence rule
Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent.

- $\text{join}(CD \Rightarrow AB, BD \Rightarrow AC)$ would produce the candidate rule $D \Rightarrow ABC$.

- Prune rule $D \Rightarrow ABC$ if its subset $AD \Rightarrow BC$ does not have high confidence.