Example: Market Basket Data

• Items frequently purchased together:
  
  Bread ⇒ PeanutButter

• Uses:
  – Product placement
  – Advertising - Amazon
  – Sales
  – Coupons
Association Rule Definitions

- **Set of items**: $I = \{I_1, I_2, \ldots, I_m\}$
- **Transactions**: $D = \{t_1, t_2, \ldots, t_n\}$, $t_j \subseteq I$
- **Itemset**: $\{I_{i1}, I_{i2}, \ldots, I_{ik}\} \subseteq I$
- **Support of an itemset**: Percentage of transactions which contain that itemset.
- **Large (Frequent) itemset**: Itemset whose number of occurrences is above a threshold.
Association Rule Definitions

• **Association Rule (AR):** implication \( X \Rightarrow Y \) where \( X,Y \subseteq I \) and \( X \cap Y = \emptyset \);

• **Support of AR (s)** \( X \Rightarrow Y \): Percentage of transactions that contain \( X \cup Y \)

• **Confidence of AR (\( \alpha \))** \( X \Rightarrow Y \): Ratio of number of transactions that contain \( X \cup Y \) to the number that contain \( X \)
Association Rule Techniques

1. Find Large Itemsets.
2. Generate rules from frequent itemsets.
Algorithm to Generate ARs

Input:

\[ D \] // Database of transactions
\[ I \] // Items
\[ L \] // Large itemsets
\[ s \] // Support
\[ \alpha \] // Confidence

Output:

\[ R \] // Association Rules satisfying \( s \) and \( \alpha \)

ARGen Algorithm:

\[ R = \emptyset; \]
for each \( l \in L \) do
  for each \( x \subseteq l \) such that \( x \neq \emptyset \) and \( x \neq l \) do
    if \( \frac{\text{support}(l)}{\text{support}(x)} \geq \alpha \) then
      \[ R = R \cup \{x \Rightarrow (l - x)\}; \]
Apriori

Large Itemset Property:
Any subset of a large itemset is large.

Contrapositive:
If an itemset is not large, none of its supersets are large.
Apriori Algorithm

1. \(C_1\) = Itemsets of size one in \(I\);
2. Determine all large itemsets of size 1, \(L_1\);
3. \(i = 1\);
4. Repeat
   5. \(i = i + 1\);
   6. \(C_i = \text{Apriori-Gen}(L_{i-1})\);
   7. Count \(C_i\) to determine \(L_i\);
   8. until no more large itemsets found;
Apriori-Gen

• Generate candidates of size i+1 from large itemsets of size i.
• Approach used: join large itemsets of size i if they agree on i-1
• May also prune candidates who have subsets that are not large.
## Apriori-Gen Example

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>Blouse</td>
</tr>
<tr>
<td>$t_2$</td>
<td>Shoes,Skirt,TShirt</td>
</tr>
<tr>
<td>$t_3$</td>
<td>Jeans,TShirt</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Jeans,Shoes,TShirt</td>
</tr>
<tr>
<td>$t_5$</td>
<td>Jeans,Shorts</td>
</tr>
<tr>
<td>$t_6$</td>
<td>Shoes,TShirt</td>
</tr>
<tr>
<td>$t_7$</td>
<td>Jeans,Skirt</td>
</tr>
<tr>
<td>$t_8$</td>
<td>Jeans,Shoes,Shorts,TShirt</td>
</tr>
<tr>
<td>$t_9$</td>
<td>Jeans</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>Jeans,Shoes,TShirt</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>TShirt</td>
</tr>
<tr>
<td>$t_{12}$</td>
<td>Blouse,Jeans,Shoes,Skirt,TShirt</td>
</tr>
<tr>
<td>$t_{13}$</td>
<td>Jeans,Shoes,Shorts,TShirt</td>
</tr>
<tr>
<td>$t_{14}$</td>
<td>Shoes,Skirt,TShirt</td>
</tr>
<tr>
<td>$t_{15}$</td>
<td>Jeans,TShirt</td>
</tr>
<tr>
<td>$t_{16}$</td>
<td>Skirt,TShirt</td>
</tr>
<tr>
<td>$t_{17}$</td>
<td>Blouse,Jeans,Skirt</td>
</tr>
<tr>
<td>$t_{18}$</td>
<td>Jeans,Shoes,Shorts,TShirt</td>
</tr>
<tr>
<td>$t_{19}$</td>
<td>Jeans</td>
</tr>
<tr>
<td>$t_{20}$</td>
<td>Jeans,Shoes,Shorts,TShirt</td>
</tr>
</tbody>
</table>
### Apriori-Gen Example (cont’d)

<table>
<thead>
<tr>
<th>Scan</th>
<th>Candidates</th>
<th>Large Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{Blouse}, {Jeans}, {Shoes}, {Shorts}, {Skirt}, {TShirt}</td>
<td>{Jeans}, {Shoes}, {Shorts},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Skirt}, {Tshirt}</td>
</tr>
<tr>
<td>2</td>
<td>{Jeans,Shoes}, {Jeans,Shorts}, {Jeans,Skirt}, {Jeans,TShirt},</td>
<td>{Jeans,Shoes}, {Jeans,Shorts},</td>
</tr>
<tr>
<td></td>
<td>{Shoes,Shorts}, {Shoes,Skirt}, {Shoes,TShirt}, {Shorts,Skirt},</td>
<td>{Jeans,TShirt}, {Shoes,Shorts},</td>
</tr>
<tr>
<td></td>
<td>{Shorts,TShirt}, {Skirt,TShirt}</td>
<td>{Shoes,TShirt}, {Shorts,TShirt},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Skirt,TShirt}</td>
</tr>
<tr>
<td>3</td>
<td>{Jeans,Shoes,Shorts}, {Jeans,Shoes,TShirt}, {Jeans,Shorts,TShirt},</td>
<td>{Jeans,Shoes,Shorts},</td>
</tr>
<tr>
<td></td>
<td>{Jeans,Shorts,TShirt}, {Jeans,Skirt,TShirt}, {Shoes,Shorts,TShirt},</td>
<td>{Jeans,Shoes,TShirt},</td>
</tr>
<tr>
<td></td>
<td>{Shoes,Skirt,TShirt}, {Shoes,Skirt,TShirt}, {Shorts,Skirt,TShirt},</td>
<td>{Jeans,Shorts,TShirt},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{Shoes,Shorts,TShirt}</td>
</tr>
<tr>
<td>4</td>
<td>{Jeans,Shoes,Shorts,TShirt}</td>
<td>{Jeans,Shoes,Shorts,TShirt}</td>
</tr>
<tr>
<td>5</td>
<td>∅</td>
<td>∅</td>
</tr>
</tbody>
</table>
Apriori Adv/Disadv

• **Advantages:**
  – Uses large itemset property.
  – Easily parallelized. How?
  – Easy to implement.

• **Disadvantages:**
  – Assumes transaction database is memory resident.
  – Requires up to m database scans.
Improving over Basic Apriori
Sampling

• Large databases
• Sample the database and apply Apriori to the sample.

• **Potentially Large Itemsets (PL):** Large itemsets from sample

• **Negative Border (BD \(-\)):**
  – Generalization of Apriori-Gen applied to itemsets of varying sizes.
  – Minimal set of itemsets which are not in PL, but whose subsets are all in PL.
Negative Border Example

PL

PL \cup B D^-(PL)
Sampling Algorithm

1. $D_s = \text{sample of Database } D$;
2. $PL = \text{Large itemsets in } D_s \text{ using smalls}$;
3. $C = PL \cup BD^{-}(PL)$;
4. Count $C$ in Database using $s$;
5. $ML = \text{large itemsets in } BD^{-}(PL)$;
6. If $ML = \emptyset$ then done
7. else $C = \text{repeated application of } BD^{-}$
8. Count $C$ in Database;
Sampling Example

- Find AR assuming $s = 20\%$
- $D_s = \{ t_1, t_2 \}$
- Smalls = $10\%$
- $PL = \{ \{\text{Bread}\}, \{\text{Jelly}\}, \{\text{PeanutButter}\}, \{\text{Bread},\text{Jelly}\}, \{\text{Bread},\text{PeanutButter}\}, \{\text{Jelly}, \text{PeanutButter}\}, \{\text{Bread},\text{Jelly},\text{PeanutButter}\} \}$
- $BD^-(PL)=\{ \{\text{Beer}\}, \{\text{Milk}\} \}$
- $ML = \{ \{\text{Beer}\}, \{\text{Milk}\} \}$
- Repeated application of $BD^-$ generates all remaining itemsets
Sampling Adv/Disadv

- **Advantages:**
  - Reduces number of database scans to one in the best case and two in worst.
  - Scales better.

- **Disadvantages:**
  - Potentially large number of candidates in second pass
Partitioning

- Divide database into partitions $D^1, D^2, \ldots, D^p$
- Apply Apriori to each partition
- Any large itemset must be large in at least one partition.
Partitioning Algorithm

1. Divide $D$ into partitions $D^1, D^2, \ldots, D^p$;
2. For $I = 1$ to $p$ do
3. $L^i = \text{Apriori}(D^i)$;
4. $C = L^1 \cup \ldots \cup L^p$;
5. Count $C$ on $D$ to generate $L$;
Partitioning Example

\[ D_1 = \{\{\text{Bread}\}, \{\text{Jelly}\}, \{\text{PeanutButter}\}, \{\text{Bread}, \text{Jelly}\}, \{\text{Bread, PeanutButter}\}, \{\text{Jelly, PeanutButter}\}, \{\text{Bread, Jelly, PeanutButter}\}\} \]

\[ D_2 = \{\{\text{Bread}\}, \{\text{Milk}\}, \{\text{Bread, Milk}\}, \{\text{Bread, PeanutButter}\}, \{\text{Milk, PeanutButter}\}, \{\text{Beer, Milk}\}\} \]

\[ S = 10\% \]
Partitioning Adv/Disadv

- **Advantages:**
  - Adapts to available main memory
  - Easily parallelized
  - Maximum number of database scans is two.

- **Disadvantages:**
  - May have many candidates during second scan.
Parallelizing AR Algorithms

• Based on Apriori
• Techniques differ:
  – What is counted at each site
  – How data (transactions) are distributed
• Data Parallelism
  – Data partitioned
  – Count Distribution Algorithm
• Task Parallelism
  – Data and candidates partitioned
  – Data Distribution Algorithm
Count Distribution Algorithm (CDA)

1. Place data partition at each site.
2. In Parallel at each site do
3. \( C_1 = \) Itemsets of size one in \( I \);
4. Count \( C_1 \);
5. Broadcast counts to all sites;
6. Determine global large itemsets of size 1, \( L_1 \);
7. \( i = 1 \);
8. Repeat
9. \( i = i + 1 \);
10. \( C_i = \) Apriori-Gen(\( L_{i-1} \));
11. Count \( C_i \);
12. Broadcast counts to all sites;
13. Determine global large itemsets of size \( i \), \( L_i \);
14. until no more large itemsets found;
CDA Example

\[ D^1: \]
\[ t_1, t_2 \]
Counts:
- Beer 0
- Bread 2
- Jelly 1
- Milk 0
- PeanutButter 2

\[ D^2: \]
\[ t_3, t_4 \]
Counts:
- Beer 1
- Bread 2
- Jelly 0
- Milk 1
- PeanutButter 1

\[ D^3: \]
\[ t_5 \]
Counts:
- Beer 1
- Bread 0
- Jelly 0
- Milk 1
- PeanutButter 0

Broadcast Local Counts
Data Distribution Algorithm (DDA)

1. Place data partition at each site.
2. In Parallel at each site do
3. Determine local candidates of size 1 to count;
4. Broadcast local transactions to other sites;
5. Count local candidates of size 1 on all data;
6. Determine large itemsets of size 1 for local candidates;
7. Broadcast large itemsets to all sites;
8. Determine \( L_1 \);
9. \( i = 1 \);
10. Repeat
11. \( i = i + 1 \);
12. \( C_i = \text{Apriori-Gen}(L_{i-1}) \);
13. Determine local candidates of size \( i \) to count;
14. Count, broadcast, and find \( L_i \);
15. until no more large itemsets found;
DDA Example

Broadcast Database Partition

D1:
- t1, t2
  Counts:
  - Beer 0
  - Bread 2

D2:
- t3, t4
  Counts:
  - Jelly 0
  - Milk 1

D3:
- t5
  Counts:
  - PeanutButter 0
Comparing AR Techniques

- Target
- Type
- Data Type
- Data Source
- Technique
- Itemset Strategy and Data Structure
- Transaction Strategy and Data Structure
- Optimization
- Architecture
- Parallelism Strategy
## Comparison of AR Techniques

<table>
<thead>
<tr>
<th>Partitioning</th>
<th>Scans</th>
<th>Data Structure</th>
<th>Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>$m+1$</td>
<td>hash tree</td>
<td>none</td>
</tr>
<tr>
<td>Sampling</td>
<td>2</td>
<td>not specified</td>
<td>none</td>
</tr>
<tr>
<td>Partitioning</td>
<td>2</td>
<td>hash table</td>
<td>none</td>
</tr>
<tr>
<td>CDA</td>
<td>$m+1$</td>
<td>hash tree</td>
<td>data</td>
</tr>
<tr>
<td>DDA</td>
<td>$m+1$</td>
<td>hash tree</td>
<td>task</td>
</tr>
</tbody>
</table>
Sequential Association Rules
Advanced AR Techniques

- Generalized Association Rules
  - Need is-a hierarchy
- Multiple-Level Association Rules
- Quantitative Association Rules
- Using multiple minimum supports

Figure 1: Example of a Taxonomy

Figure 1: A taxonomy for the relevant data items
Measuring Quality of Rules

- Support (Joint probability)
- Confidence (Conditional probability)
- Interest (Essentially a measure of independence)
- Conviction (Asymmetrical interest measure)
  - $A \rightarrow B$ rewritten using the implication elimination of P.L?
- Chi Squared Test
  - Create a contingency table, test for independence
What Is Sequential Pattern Mining?

- **Sequential pattern mining:**
  - Frequent temporal sequential patterns in the database.
  - Association rule --- intra-transaction
  - Sequential rule --- inter-transaction

- **Example (Video stores can use this)**
  - 80% of customers typically rent “star wars”, then “Empire strikes back”, and then “Return of Jedi”.

- **Applications.**
  - Change experimental unit from a set to a sequence
  - Web-access pattern.
  - Predict onset of disease from a sequences of symptoms, etc
Sequential Association Rules

Given:
A set of objects with associated event occurrences.

Actual Pattern
\{1\} \rightarrow \{2\}

Possible patterns
\{1,4\} \rightarrow \{3,2\}

Note arrow means “some time after” \textbf{not} directly after

Only counts once
## Input Database

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>TransactionTime</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10,20</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>40,60,70</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>30,50,70</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>40,70</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>90</td>
</tr>
</tbody>
</table>

MinSupp=40%, i.e. 2 customers:

- \(<30><90>\) (1,4)
- \(<30><40,70>\) (2,4)

CSI 661 - Data Mining Lecture 1
Seq. Association Rule Defn

- Given is a set of objects, with each object associated with its own timeline of events, find rules that predict strong sequential dependencies among different events.

(A B) (C) → (D E)

- Rules are formed by first discovering patterns. Event occurrences in the patterns are governed by timing constraints.

(A B) (C) (D E)

<= xg > ng <= ws

<= ms
Complexity

- Much **higher computational complexity** than association rule discovery.
  - $O(m^k 2^{k-1})$ number of possible sequential patterns having $k$ events, where $m$ is the total number of possible events.

- Time constraints offer some pruning. Further use of **support based pruning contains complexity**.
  - A subsequence of a sequence occurs at least as many times as the sequence.
  - A sequence has no more occurrences than any of its subsequences.
  - Build sequences in increasing number of events. [GSP algorithm by Agarwal & Srikant]
Sequential APriori

\[ S_1 = \{ \text{frequent 1-item sequences} \}; \]
\[ k = 2; \]
\[ \text{while}( S_{k-1} \text{ is not empty } \} \{ \]
\[ C_k = \text{Sequential}_\text{Apriori}_\text{generate}( S_{k-1} ); \]
\[ \text{for all object timelines } o \text{ in } T \{ \]
\[ \quad \text{Count}( C_k, o ); \]
\[ \} \]
\[ S_k = \{ s \text{ in } C_k \text{ s.t. } s.\text{count} \geq \text{minimum_support} \}; \]
\[ \} \]
\[ \text{Answer} = \text{union of all sets } S_k; \]
Sequential Apriori-Gen

Sequential_Apriori_generate( S_{k-1} ) {
    join S_{k-1} with S_{k-1} such that,
        c_1 and c_2 join together if subsequences formed by
        deleting first event of c_1 and last event of c_2 are the same.
    and then new candidate, c, has a form
        c = (c_1)(e) or (c_1 e) if e, the last event of c_2, is the only event
        in last event-set of c_2 or otherwise, respectively.
    c is then added to a hash-tree structure.
}

Examples of join:
    (2 3) (4 6) (7) joins with (3) (4 6) (7) (8) to create (2 3) (4 6) (7) (8)
    (1 2) joins with (2 3) to create (1 2 3)
Special case (k=2): (1) joins with (2) to create (1)(2) and (1,2)
Count Operation

- Various possibilities for counting occurrences of a sequence in an object’s timeline
  - Count only one occurrence per object.
  - Count the number of span-size windows the sequence occurs in.
  - Count the number of distinct occurrences of a sequence:
    - Each event-timestamp pair considered at most once.
    - Each counted occurrence has at least one new event-timestamp pair.
Apriori-All: Problem Decomposition.

- 1. Sort phase:
  - CID: major key, TID: secondary key
- 2. Litemset Phase:
  - Litemset: an itemset with min support, adapted from association rule mining algorithm.
- 3. Transform phase
  - Map litemset into contiguous integers, transform database
- 4. Sequence phase: AprioriAll, find large sequences S
- 5. Maximal phase: delete all sub-sequences in S
Transform Phase.

<table>
<thead>
<tr>
<th>itemset</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>{30}</td>
<td>1</td>
</tr>
<tr>
<td>{40}</td>
<td>2</td>
</tr>
<tr>
<td>{70}</td>
<td>3</td>
</tr>
<tr>
<td>{90}</td>
<td>4</td>
</tr>
<tr>
<td>{40,70}</td>
<td>5</td>
</tr>
</tbody>
</table>

Litemsets
Transform Phase.

CID=1

CID=2

CID=3

CID=4

CID=5

<table>
<thead>
<tr>
<th>itemset</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>{30}</td>
<td>1</td>
</tr>
<tr>
<td>{40}</td>
<td>2</td>
</tr>
<tr>
<td>{70}</td>
<td>3</td>
</tr>
<tr>
<td>{90}</td>
<td>4</td>
</tr>
<tr>
<td>{40,70}</td>
<td>5</td>
</tr>
</tbody>
</table>

Litemsets

CSI 661 - Data Mining Lecture 1
The AprioriAll Algorithm

• **Pseudo-code:**

  \( C_k \): Candidate sequence of size \( k \)
  
  \( L_k \): frequent or large sequence of size \( k \)

  \( L_1 = \{ \text{large 1-sequence} \}; \) //result of litemset phase

  \textbf{for} (\( k = 2; \ L_k \neq \emptyset; \ k++ \)) \textbf{do begin}

  \hspace{1cm} \( C_k \) = candidates generated from \( L_{k-1} \);

  \hspace{1cm} \textbf{for each} customer sequence \( c \) in database \textbf{do}

  \hspace{2cm} Increment the count of all candidates in \( C_k \)
  \hspace{2cm} that are contained in \( c \)

  \hspace{1cm} \textbf{end}

  \textbf{Answer} = \text{Maximal sequences in } \bigcup_k L_k;
Candidate generation

- **Join Step:**
  - $C_k$ is generated by joining $L_{k-1}$ with itself
  - Insert into $C_k$,
  - Select $p.litemset_1, \ldots, p.litemset_{k-1}, q.litemset_{k-1}$
  - From $L_{k-1} p, L_{k-1} q$
  - Where $p.litemset_1 = q.litemset_1, \ldots,$
  - $p.litemset_{k-2} = q.litemset_{k-2}$
- **Prune Step:** Any $(k-1)$-subsequences of $s$ (length $k$) that is not frequent cannot be a subsequence of a frequent $k$-sequence.

For example: $\{1,2,3\} \times \{1,2,4\} = \{1,2,3,4\}$ and $\{1,2,4,3\}$